社会統計学 B

R を使って自習する

京都大学大学院地球環境学堂
吉野 章

2014 年 7 月 31 日

目次

第 1 章

解題

1.1　はじめに

この資料は、2013 年度後期で行った全学共通科目「社会統計学 B」の内容を、統計解析ソフト R を用いて自習できるように解説したものです。講義内容に、講義で出てきた例や図表を R で再現するためのスクリプトおよびその出力を記述しています。

統計学は、講義をぼーと聞いているだけではなかなか身につきません。実際に自分でデータをいじって、試行錯誤しながら、ようやく理解できることも多いです。

この資料に沿って自習することで、R によるデータの扱い方に慣れ、統計学を学習する上での利用の仕方がわかるようになるはずです。この講義では、回帰分析の直感的な理解を目指していますが、そうした自習によって、より深く理解ができると思います。さらに、それ統計学についても同じような方法で自習することで、学習の助けを得ることができるようになるはずです。存分に活用してください。

1.2　講義の目標と進め方

1.2.1　講義の目的

社会科学における実証的研究を行う場合、おそらく避けて通れないであろう、最も基本的で多用される回帰分析の初歩的理解を目標とします。

ただし、数理統計学の厳密な証明などはほどんど出てきません。あくまで、実際の分析に必要な基礎的な理解と作法について解説します。

しかしながら、回帰分析を機械的に当てはめて、統計の罠にはまらないように、回帰分析とは一体何をやっているのか、その原理は何かまで理解できることを目指してます。あくまで、直感的にではありますが・・・

1.2.2　授業の進め方

最初に、とりあえず回帰分析とはどういうものかを理解してもらうために、最も基本的な回帰分析である通常最小二乗法（OLS）を使ってみます。中古車価格を例として、それが走行距離や車種などでどう変わるかを推定します。

次に、OLS 以外のいろいろな回帰分析をやってます。その背後にある理論はそれぞれで、理解するには少し勉強が必要ですが、データから結果を導くまでの考え方も手順もだいたい同じであることを理解できると思います。

ある程度、回帰分析がどうようものかわかってもらったところで、次に統計学の基本を勉強します。最もよく使われる代表値である標本平均と標本分散、確率分布の意味、統計学を理解するために避けて通れない大数の法則と中心極限定理について直感的に理解します。

続いて、はじめにやってみた OLS 回帰分析を、再びとりあげて、もう少ししっかり理解します。OLS 回帰係

数とは何なのか、どのような仮定でどのような性質があるのか、推定値の分布や検定方法についても学びます。

さらに、OLS 回帰分析の発展形として、OLS 回帰で前提としているいくつかの仮定を外した推定方法についても紹介します。

授業に必要な知識として、少し心配なのは、ひとつは偏微分です。ちょこっとだけ出てくるし、そんなに難しいものでもないので、これについては、出てきたところで解説します。

もう一つは、行列計算です。結構大きな行列、逆行列、転地行列、それらのかけ算などが出てきますが、これについてもひとつひとつ解説しながら進めます。

1.2.3 R について

R とは

R は統計解析ソフトの一つで、AT&T(当時) のベル研究所で開発された S 言語をベースに、オープンソースで開発された無料のソフトウエアです。無料ですが、多種多様な統計手法やグラフィックスが利用可能です。特殊な手法もパッケージまたはライブラリとして提供されている場合が多く、関連書籍も多数出版されています。

ただし、それらは多数の開発者たちによるもので、完全無保証ですので、推定結果の利用には留意が必要です。

R のインストール

R には、Windows 版、Mac OS X 版、Linux 版があり、以下から入手可能です。

http://cran.r-project.org/

インストール作業は簡単で、Windows 版は、ダウンロードしてきたファイルをダブルクリックするだけで、あとは指示に従うだけでインストールできます。

具体的なインストール手順については参考文献や以下のようなサイトにいくらでもありますので、そちらを参考にしてください。

`http://www.kkaneko.com/rinkou/r/rinstall.html`

R の使い方

R を起動させたたら、Console と呼ばれる画面で出ます。味もそっけもない画面ですが、ここに文字や数字を入力することで R を動かすことができます。

ひとつひとつの動作は、この資料の解説をはじめからやれば習得できるようにしてありますので、その都度覚えていてもらえば大丈夫です。

それでも、もう少し基本的なところや、それ以外についても勉強したいという人は、R をチュートリアル形式で解説した本として、以下がよいかもしれません。

竹内俊彦『はじめての S-PLUS/R 言語プログラミング』オーム社、2005 年、2800 円 + 税

R のレファレンスとしては、以下が充実しています。

`http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html`

それを書籍化したものもあります。

舟尾暢男『The R Tips －データ解析環境 R の基本技（第二版)』オーム社、2009

第 2 章

とりあえず OLS：中古車価格を例に

2.1　中古車価格の OLS 推定

2.1.1　データを読み込み、整形する

作業フォルダの指定： setwd 関数

エクセルのデータを CSV 形式で保存したファイル (CSV ファイル）を（例えばデスクトップなど）の適当な位置に置く。

そのファイルを置いたフォルダを作業フォルダに指定すると、ファイルパスをいちいち指定しなくていいので便利。

フォルダの区切りが￥でなく / であることに注意！

```
> setwd("C:/Users/yoshino/Documents/Rwd/statistics")
```

csv ファイルの読み込み： read.csv 関数

エクセルで CSV 形式で保存したデータを R のデータフレーム形式に読み込む

```
> d01<-read.csv("prius.csv")
```

データの最初の部分の表示： head 関数

ちゃんと読み込んだか最初のいくつかのレコードを表示

```
> head(d01)
```

```
  nenshiki kyori         kizu price grade selection shaken    color NV SR kawa
1       24   0.6 すごくきれい 241.0     S       TSG     21       白  1  0    0
2       23   3.0         普通 124.5     S       TSG     21       白  1  0    0
3       22   3.7       きれい 128.5     S       non     12       紺  1  0    0
4       22   3.6 すごくきれい 146.5     L       non      0       黒  0  0    0
5       22   5.1 すごくきれい 160.5     S       non      0 シルバー  1  0    0
6       24   0.5       きれい 171.0     S        TS      5       白  1  0    0
```

要約統計量を表示する： summary 関数

データの要約を見てみる。

summary 関数で読み込んだデータ d01 の要約を表示すると・・・

データの形式には項目（カテゴリー）変数の factor と数値変数 numerinc がある。

数値（例えば、 nenshiki ）は、最小値（ Min. ）、小さい方から 1/4 番目（ 1stQu. ）、中央値（ Median ）、

平均値（ Mean ）、3/4 番目 (3rd Qu. 、最大値（Max.）が表示されている。

項目変数（例えば、 kizu ）は、各項目の度数が表示されている。

```
> summary(d01)
```

```
   nenshiki         kyori                   kizu         price       grade
Min.   :21.00   Min.   : 0.000   きれい       :306   Min.   : 76.5   G:151
1st Qu.:22.00   1st Qu.: 2.000   すごくきれい:414   1st Qu.:125.5   L:105
Median :22.00   Median : 3.600   修復歴あり   :158   Median :144.5   S:744
Mean   :22.11   Mean   : 4.643   普通         :122   Mean   :145.3
3rd Qu.:23.00   3rd Qu.: 6.400                       3rd Qu.:162.0
Max.   :24.00   Max.   :22.500                       Max.   :256.0

selection     shaken            color           NV              SR
LED: 65   Min.   : 0.000   白       :372   Min.   :0.00   Min.   :0.000
non:653   1st Qu.: 0.000   シルバー:261   1st Qu.:0.00   1st Qu.:0.000
TS :274   Median : 5.000   黒       :228   Median :1.00   Median :0.000
TSG:  8   Mean   : 7.014   青       : 71   Mean   :0.65   Mean   :0.032
          3rd Qu.:13.000   赤       : 24   3rd Qu.:1.00   3rd Qu.:0.000
          Max.   :31.000   パープル: 15   Max.   :1.00   Max.   :1.000
                           (Other) : 29
     kawa
Min.   :0.000
1st Qu.:0.000
Median :0.000
Mean   :0.044
3rd Qu.:0.000
Max.   :1.000
```

2.1.2　走行距離の価格との関係をプロットしてみる

その前に、よく使うデータフレームを登録しておく： attach 関数

データフレーム d01 の中の変数 grade を指定するには d01$grade としなければならない。しかし、 $d01 をいちいち入力するのは面倒。

attach 関数を使う。

```
> attach(d01)
```

これで、たとえば summary(grade) というように、 summary(d01$grade) としなくても、 d01$ 無しで d01 の中の変数を扱えるようになる。

```
> summary(grade)
```

```
  G   L   S
151 105 744
```

解除する場合は detach(d01) というように指定する。これで d01$grade が見つけられなくなる。（以下は、エラーとなるので summary(grade) は#でコメントアウトしているが、やってみられたい。）

```
> detach(d01)
> # summary(grade)
```

もう一回 attach 関数を使うと、復活！

```
> attach(d01)
> summary(grade)
```

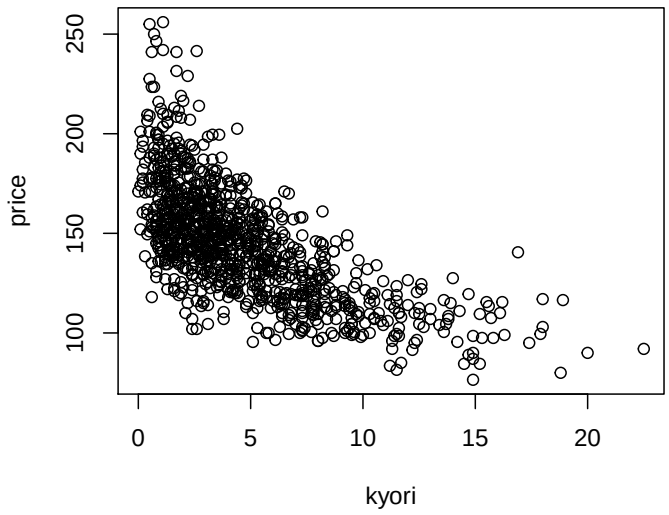
```
  G   L   S
151 105 744
```

散布図を描く： plot 関数

変数 x と y の散布図は、 plot(x,y) とすると、横軸に x 、縦軸に y を描いてくれる。
あるいは、 plot(y~x) でもよい。(x と y の順番が入れ替わっているのに注意せよ！)

データフレーム d01 の変数 price （車両価格）と kyori （走行距離）との散布図は、 plot(price~kyori,d01) だが d01 は attach 関数で登録してあるので、 d01 無しでいける。

```
> plot(price~kyori)
```



2.1.3 OLS 回帰を行う

OLS 回帰とは・・・

この分布の”ど真ん中”を通る直線を描きたい！

price を y と書く。

price は 1000 個の値があるから、y はベクトルで、

$$y = (y_1, y_2, y_3, ..., y_{1000})$$

同様に、 kyori を x_1 と書く。

kyori も対応する 1000 個の値があるから、x_1 はベクトルで、

$$x_1 = (x_{11}, x_{12}, x_{13}, ..., x_{1,1000})$$

ここで、直線を

$$y = a + bx_1$$

で表すと、この直線が分布の”ど真ん中”を通るように、a と b を決める。ここで、何を”ど真ん中”とするかだが、1000 個の各データ (x_{1i}, y_i) について、

$$\hat{y}_i = a + bx_{1i}$$

と y と差の二乗和を最小とするような、a と b_1 を探す。

$$\min_{a.b_1} \sum_{i=1}^{1000} (y_i - \hat{y}_i)^2 = \min_{a,b_i} \sum_{i=1}^{1000} (y_i - (a + bx_i))^2$$

これを通常最小二乗法（Ordinary Least Square）と言う

OLS 推定を行う： lm 関数

データ x と y の OLS 推定は lm(y~x) で R がやってくれる。

変数 x と y がデータフレーム d01 に入っていたら、 lm(y~x,d01) である。

したがって今回は kyori と price の回帰だから、 lm(price~kyori,d01) だが、 d01 は attach 関数で登録したので、 d01 は省略できる。

```
> o01<-lm(price~kyori)
```

結果は o01 に格納したので表示されない。次で表示できる。

```
> o01
```

```

Call:
lm(formula = price ~ kyori)

Coefficients:
(Intercept)        kyori
    169.858       -5.299
```

もっと、詳しく結果を見たい場合は、 summary 関数を使う。

```
> summary(o01)
```

```

Call:
lm(formula = price ~ kyori)

Residuals:
    Min      1Q  Median      3Q     Max
-55.140 -14.451  -1.665  11.879  91.971

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 169.8578     1.0886  156.03   <2e-16 ***
kyori        -5.2992     0.1855  -28.57   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.06 on 998 degrees of freedom
Multiple R-squared:  0.4499,        Adjusted R-squared:  0.4494
F-statistic: 816.3 on 1 and 998 DF,  p-value: < 2.2e-16
```

OLS 推定結果の係数の推定値を表示する： coef 関数

summary 関数で出力される数値の意味については、順を追って説明していくつもりだが、今の段階で、こんなにたくさん情報を与えられても困るだろう。とりあえず、a と b_1 の推定値が欲しい。 o01 から coef 関数を使って取り出す。(ちなみに、 coef 関数は、正式には coefficients 関数の省略形。)

```
> coef(o01)
```

```
(Intercept)       kyori
 169.857773   -5.299245
```

これは o01 の要素だから、次のようにして取り出すこともできる。

```
> o01$coef
```

```
(Intercept)       kyori
 169.857773   -5.299245
```

このうち、 (Intercept) というのが直線の切片で a の推定値で、169.858。

kyori のところが b の推定値で、-5.299。

つまり、（最小二乗法の意味で）分布図の”ど真ん中”に引いた線は、

$$y = 169.858 - 5.299x_1$$

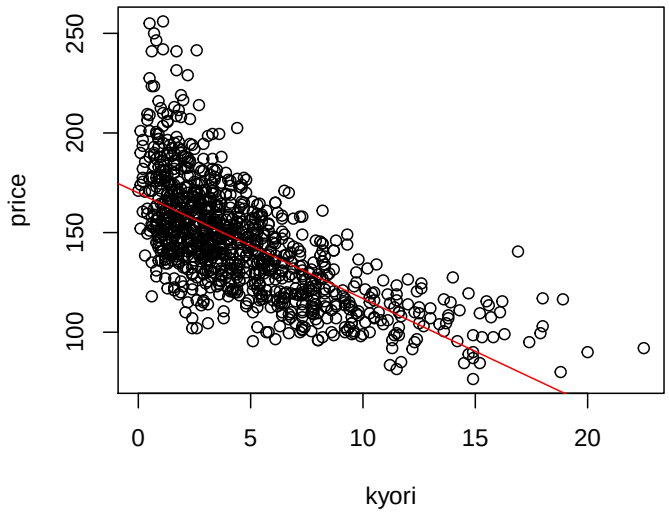
ということになる。

ちなみに、この”ど真ん中”に引いた線のことを「回帰直線」と呼ぶ。

回帰直線を分布図に書く： abline 関数

abline 関数は、既に描いてある分布図に、いろんな方法で直線を描くことができる関数だが、今回のような x と y だけの OLS 推定を行った場合は、推定結果 o01 を使って abline(o01) で描き入れることができる。ただ、これだと直線が黒色で味気ないし、引数 col=”red” を加えて赤で表示しよう。

```
> plot(price~kyori)
> abline(o01,col="red")
```



結果をどう解釈するか？

ほとんど走行距離がなかったら（新古車？）、この型のプリウスは、期待値としてだいたい 170 万円ぐらい。それが、走行距離 1 万 km ごとにだいたい 5.3 万円ぐらい価格が下がるようだ。

しかし、それにしても同じ走行距離でも、価格差がありすぎる。例えば、5 万 km 走った車だと、100 万円から 170 万円ぐらいの幅がある。

走行距離以外の情報も使えないのか？（つづく）

2.1.4　練習問題

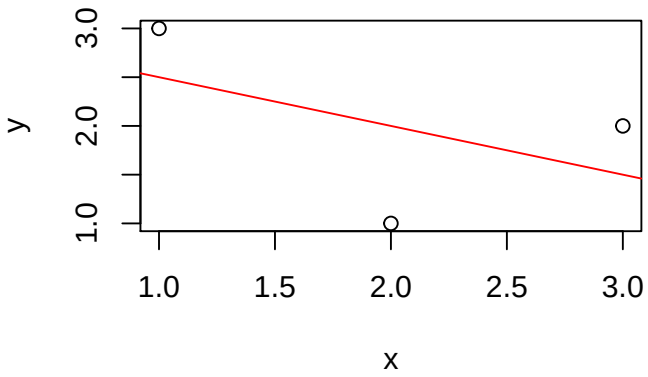
問題

次の 3 つだけのデータセットについて

	x	y
1	1	3
2	2	1
3	3	2

次の回帰直線を自力による計算と R を使った 2 とおりで、OLS 推定しなさい。

$$y = a + bx$$



解答例

自力計算は省略するので、各自一生に一回はやてみること。あとは、R を使った例だけやります。

エクセルでデータを入力して、CSV ファイルとして保存して、それを読み込むというのもありだが、たった 3 x 2 しかデータがないので、直接入力する。

3 x 2 の行列をつくる： matrix 関数

d51 という名前で、データフレームをつくる。まず、 matrix 関数で行列のデータをつくる。

matrix(v,nrow=3,ncol=2) と指定すると、データの集まり（ベクトル）v を、行数 (nrow=)3、列数 (nrow=) 2 に分割する。但し、分割は、最初に、1 列目に割り当て、続いて、2 列目・・・と言う具合。行ごとに割り当てたい場合は、引数 byrow = T を加えて、 matrix(v,nrow=3,ncol=2,byrow) と指定する。

matrix(v,nrow=3,ncol=2) は、そのままの順番で引数を指定するなら、 nrow= と ncol= を省略して、matrix(v,3,2) と書ける。

```
> (d51<-matrix(c(3,1,2,1,2,3),3,2))
```

```
     [,1] [,2]
[1,]    3    1
```

```
[2,]    1    2
[3,]    2    3
```

データフレーム形式への変換： data.frame 関数

この行列をデータフレーム形式に変換する。

この場合、行列のままでもよいが、データフレームの方が、各列のデータを別々の変数として扱うので、回帰分析には適している。ちなみに、行列の場合、データの 1 つが文字だったら、その他のデータが数字でも、R は行列全体のデータを文字として扱う。データフレームの場合は、ひとつのデータが文字だったら、そのデータが入っている列のみについて、その列のデータ全体を文字として扱う。

データフレームに変換すると、列の名前が X1 、 X2 と変わる

```
> (d51<-data.frame(d51))
```

```
  X1 X2
1  3  1
2  1  2
3  2  3
```

データフレームの変数名を指定： colnames 関数

データフレームの変数名（列名）を指定したい場合は、 colnames 関数でデータフレームの列名を指定し、そこに文字列のベクトルを代入すればよい。

colnames 関数そのものは、データフレームの列名を指定する関数

```
> colnames(d51)
```

```
[1] "X1" "X2"
```

そこに文字列を代入すると、列名が入れ代わる。文字列のベクトルは、各要素を ” ” で囲わなくてはいけないので、要注意。

```
> colnames(d51)<-c("y","x")
> d51
```

```
  y x
1 3 1
2 1 2
3 2 3
```

ちなみに、同じようにして rownames 関数を使って、レコード名（行名）をつけることができるが、ここではやらない。

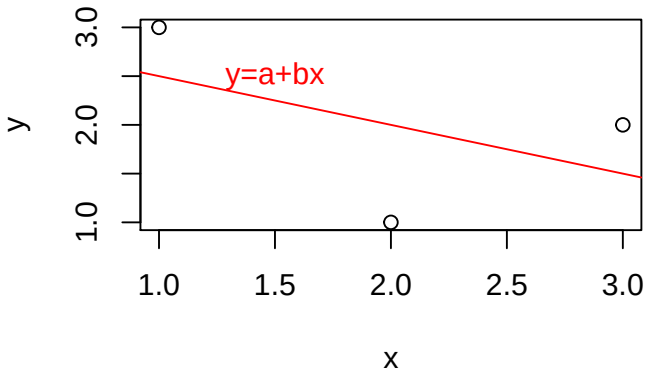
OLS 推定： lm 関数

データフレーム d51 に変数 x と y があり、 y に x を回帰させる場合、 lm(y~x,d51) で OLS 推定できる。

```
> lm(y~x,d51)
```

```
Call:
lm(formula = y ~ x, data = d51)

Coefficients:
(Intercept)            x
        3.0         -0.5
```

coefficients: の (Intercept) のところの値が、直線 $y = a + bx$ の切片 a の推定値。 x のところが、x の係数 b の推定値。



2.2 ダミー変数を加えた重回帰分析

2.2.1 残差の散布図を描く

前回の作業結果

念のため、今回の作業に必要な前回の作業をもう一回やっておく（前回に引き続いて作業する場合はここは不要）

```
> d01<-read.csv("prius.csv")
> attach(d01)
> o01<-lm(price~kyori)
```

残差を求める： resid 関数

残差（residuals）は回帰直線で説明できない y の値。

$$e_i = y_i - \hat{y}_i, (i = 1, ..., 1000)$$

resid 関数を使うと lm 関数を使って推定した OLS 推定の結果のリスト o01 から残差の値（1000 個）を取り出すことができる。

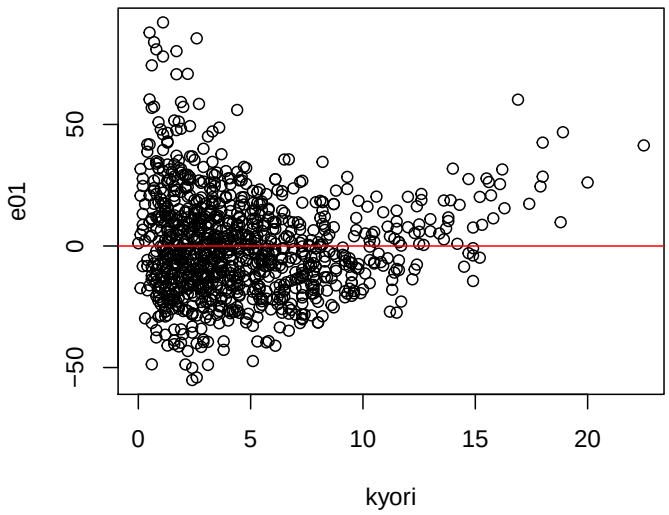
resid 関数を使って残差を取り出し、 e01 に代入する。

```
> e01<-resid(o01)
```

残差の散布図を描く： plot 関数と abline 関数

距離によって残差がどうなるかを、 plot 関数で散布図を描いて確認する。見にくいので、$e = 0$ のところに赤線を引く。abline 関数は、（前回は lm 関数による OLS 推定の結果 o01 を使って abline(o01) として利用したが）、切片 a と傾き b を与えて abline(a,b) と描くこともできる。この場合、切片 0、傾き 0 だから abline(0,0) ということになる。

```
> plot(e01~kyori)
> abline(0,0,col="red")
```



2.2.2　カーナビの有無の価格差を推定する

カーナビ変数を項目変数に変換： factor 関数

残差の散布図を見ると、走行距離によっては 100 万円近い差がある。カーナビが付いているかいないかでも価格差はあるのではないか？

カーナビが付いているかどうかは、変数 NV に記録されている。変数 NV が 1 と記録されている車はナビ付き、 0 と記録されているのはナビ無し

```
> summary(NV)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    1.00    0.65    1.00    1.00
```

変数 NV は 1,0 の数値で入力されているので、R は数値として扱っている。これを factor 関数を使って項目変数に変換しておこう。[*1]

```
> NV<-factor(NV)
> summary(NV)
```

[*1] ここでちょっと注意！　ここで、元のデータの変更を行った。元のデータ d01 を attach 関数で登録していたから、上の作業では d01$NV とは指定しなかった。しかし、実は、 attach 関数で登録したデータは、 d01 を読み込んだもので、 d01 そのものではない。上のように d01$ 無しでいじられても、 d01 自体は変更されていない。 d01$NV を summary 関数で確認してみると、数値として扱われていることがわかる。

```
> summary(d01$NV)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    1.00    0.65    1.00    1.00
```

一時的に変更したい場合は、元のデータが崩れたりしないので便利だが、そのまま変えたいときは、改めて d01$ を付けて代入しておく必要がある。

```
> d01$NV<-factor(NV)
```

もう一度 summary 関数で d01$NV を確認してみると、項目変数となっている。

```
> summary(d01$NV)
```

```
  0   1
350 650
```

```
  0   1
350 650
```

ナビの有無で中古車価格の分布を色分けする： plot 関数の引数 bg= と pch=

plot(price~kyori) で描いた散布図を、カーナビ付きかどうかで色分けしたい。 plot 関数は、どんな点の形や色でプロットするかを pch= や bg= で指定することができる。 pch=21 というのは円で、 bg="red" とすると赤く塗りつぶす。色分けするには、 c("red","blue") のように、ベクトルで複数の色を指定して、その後塗り分ける項目変数を指定すればよい。

ナビ無しを白 white 、ナビ付きを青 blue で塗り分けてみた。[*2]

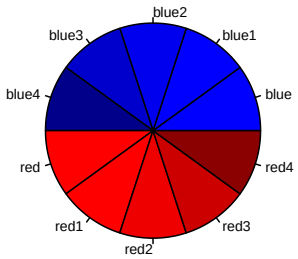
```
> plot(price~kyori,pch=21,bg=c("white","blue")[NV])
```

[*2] ここでちょっと横道に！ 直接色の名前を指定したいとき、どんな名前が利用できるかは、 colors() とすると一覧が出力されます。ただし、657 個も出力されて、長いので、ここでは [1:30] をつけて、最初の 30 個だけ表示しました。

```
> colors()[1:30]
 [1] "white"           "aliceblue"       "antiquewhite"    "antiquewhite1"
 [5] "antiquewhite2"   "antiquewhite3"   "antiquewhite4"   "aquamarine"
 [9] "aquamarine1"     "aquamarine2"     "aquamarine3"     "aquamarine4"
[13] "azure"           "azure1"          "azure2"          "azure3"
[17] "azure4"          "beige"           "bisque"          "bisque1"
[21] "bisque2"         "bisque3"         "bisque4"         "black"
[25] "blanchedalmond"  "blue"            "blue1"           "blue2"
[29] "blue3"           "blue4"
```

色の名前を直接指定して円グラフを描いてみる。円グラフを描く pie 関数の中で色を指定する引数 col= に青系と赤系の 10 色の名前（長いので、その前に cl01 に代入しています）を与えて、1 が 10 個の円グラフを描いた。ちらみに、 rep(1,10) は、1 を 12 回繰り返し出力させる関数で、 c(1,1,1,1,1,1,1,1,1,1) と同じ。 labels は、データラベルを指定する引数で、与えた色の名前のベクトル cl01 をそのまま与えてある。

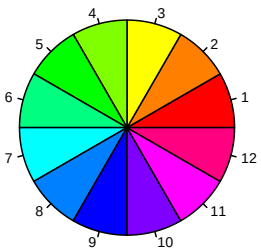
```
> cl01<-c("blue","blue1","blue2","blue3","blue4","red","red1","red2","red3","red4")
> par(mar=c(0,2,0,2))
> pie(rep(1,10),col=cl01,labels=cl01,cex=.6)
```



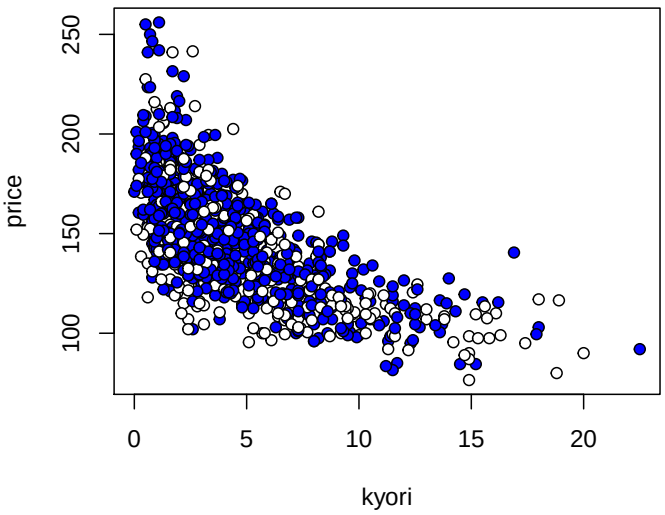
虹色を順に指定できる rainbow 関数や、暖色系を順に指定する heas.colors 、寒色系を順に指定する cm.colors などもある。

rainbow 関数の出力例として、円グラフを描く pie 関数の中で色を指定する引数 col= に rainbow 関数で 12 色を与えて、1 が 12 個の円グラフを描いた。

```
> par(mar=c(0,2,0,2))
> pie(rep(1,12),col=rainbow(12),cex=.6)
```



ちなみに、R でプロット点の指定や色の指定は、http://cse.naro.affrc.go.jp/takezawa/r-tips/r/53.html を参照してください。



NV を説明変数を加えた回帰はどういうこと？

ナビが付いていない車 (白丸）だけの回帰直線を次で引く。

$$y_i = a + b_1 x_{1i}$$

ナビが付いてる車（青丸）については次の回帰曲線を引く。NV が付いていることで高くなるぶんを b_2 で表す。

$$y_i = a + b_1 x_{1i} + b_2$$

ナビが付いていないのを 0、ナビが付いているのを 1 とした変数を $x_2 = (x_{21}, x_{22}, x_{23}, ..., x_{2,1000})$ とすると、上の２つの式を次のひとつの式で表すことができる。

$$y_i = a + b_1 x_{1i} + b_2 x_{2i}$$

	price:y	kyori:x_1	NV:x_2
1	241.0	0.6	1
2	124.5	3.0	1
3	128.5	3.7	1
4	146.5	3.6	0

ちなみに、２つ以上の説明変数を使った回帰分析を「重回帰分析」と言うことがあります。一方、説明変数１つだけの回帰分析は「単回帰分析」と言われます。（と言っても、社会科学で単回帰分析でうまくいくことはほとんどなく、まずほとんどが重回帰分析となるので、単回帰、重回帰と呼び分けるのは、最初に回帰分析を習った時ぐらいでしょうか）

また、説明変数が項目変数の場合、上のように 0,1 のデータに変換して回帰分析を行います。このように 0,1 に変換された項目変数のことを「ダミー変数」と呼びます。

２変数で OLS 回帰する： lm 関数

ナビ無し（白丸）とナビ付（青丸）それぞれの回帰直線を OLS で推定する。

$y = a + b_1 x_1 + b_2 x_2$ の OLS 回帰を行うには、lm 関数の最初の引数は、 fromula=y x1+x2 というように指定する。被説明変数 y を ~ の前に、説明変数を後に書く。被説明変数が複数ある場合は x1+x2 という具合に足せばよい。

本来なら 1+x_1+x_2 だが、 1+ は省略できる。

formula= は省略できるので、次のように OLS 回帰を行い、結果を o01_NV に代入した。

```
> o01_NV<-lm(price~kyori+NV)
```

回帰係数を表示する: coef 関数

coef 関数については前述した。OLS 回帰の結果 o01_NV から推定した回帰係数を取り出す。ここではそれを o01_NVc に代入している。

ちなみに、前後が括弧 ( ) で囲まれているが、こうすると、代入すると同時に結果を表示する。

```
> (o01_NVc<-coef(o01_NV))
```

```
(Intercept)       kyori         NV1
 166.383875   -5.244711    4.954958
```

３つの係数が推定された。順に、a、b_1、b_2 の推定値。

価格と距離との散布図に２つの回帰直線を引く： plot 関数と abline 関数

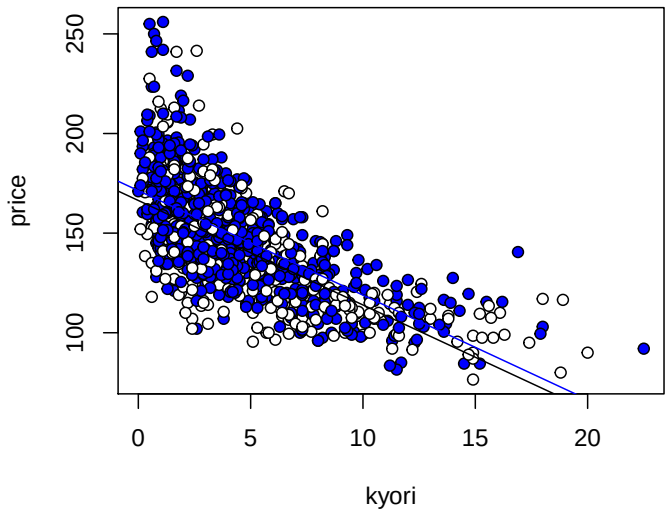
ナビ無しの回帰直線の切片は a で 166.38、傾きは b_1 で -5.245。

$$y = 166.38 - 5.247x_1$$

ナビ付きの回帰直線の切片は $a + b_2$ で $166.38 + 4.955 = 171.34$、傾きは b_1 で -5.245。

$$y = 171.34 - 5.247x_1$$

```
> plot(price~kyori,pch=21,bg=c("white","blue")[NV])
> abline(o01_NVc[1],o01_NVc[2])
> abline(o01_NVc[1]+o01_NVc[3],o01_NVc[2],col="blue")
```



ナビが付いていることで、約 5 万円弱高くなっているようだが、それを差し引いても、まだまだ価格のばらつきは大きい。

残差のばらつきを確認する: resid 関数

各回帰直線からの残差をプロットすると、以下のようになる。

```
> plot(resid(o01_NV)~kyori,pch=21,bg=c("white","blue")[NV])
> abline(0,0,col="red")
```

ナビ付きかどうか NV を説明変数に加えることにより、走行距離 kyori だけでは説明できない値段 price のバラツキを、ちょっとだけ説明できた。でも、ちょっとだけだ。他に、もっと説明できる変数はないのか？（つづく）

2.3　３項目以上のダミー変数への変換

2.3.1　年式の価格差を推定する

前回の作業結果

ここからはじめる人のために、今回の作業に必要な作業をここでやっておく。

```
> d01<-read.csv("prius.csv")
> d01$NV<-factor(d01$NV)
```

年式の変数を項目変数に変換： factor 関数

中古車価格は、当然、年式にも左右されるはず。年式は、車が最初に販売された年であり、年式が新しいと車も新しいし、ちょっとした装備も代わっていたりする。

今回の車種の場合、平成 21〜24 年の 4 種類が中古車市場に出ていて、データとしては、 nenshiki という名前で、 21,22,23,24 という具合に、数値で入力されている。 summary 関数で要約すると、最小値、1/4 分位値、中央値、、、、と言う具合に、出力され、数値データであることがわかる。

```
> summary(d01$nenshiki)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  21.00   22.00   22.00   22.11   23.00   24.00
```

これを数値のまま扱うか、項目変数に変換するかは判断の分かれるところ。平成 21 年式から平成 24 年式まで、1 年式ごとに中古車価格が直線的に高くなるのであれば、数値のままでよいが、たとえば、平成 21 年式と 22 年式との差と、平成 23 年式と 24 年式との差が全然違うのであれば、項目データとしておいた方がよい。

ここでは、ひとまず項目データとして扱うことにする。項目データに変換するには factor 関数だった。

```
> d01$nenshiki<-factor(d01$nenshiki)
```

データ d01 を登録する: attach 関数

d01$ といちいち書くのは面倒くさいので、 attach 関数で d01 を登録しておく。d01 が変更されているので、前回から引き続いて作業している場合は、一旦、 detach 関数で登録を解除してから、再び attach 関数で登録を行ってください。(予め登録されていない場合は、 detach 関数のところで Error が表示されるが、無視してよい。下では、#を入れてコメントアウトしているので、必要な人は#を外してください。)

```
> # detach(d01)
> attach(d01)
```

d01 が登録されて、 nenshiki が項目変数に変換されているか確認してみる。 summary 関数で d01$ 無しで要約を表示させると、ちゃんと出力され、年式（21,22,23,24）ごとに度数が表示され、項目変数に変換されていることがわかる。

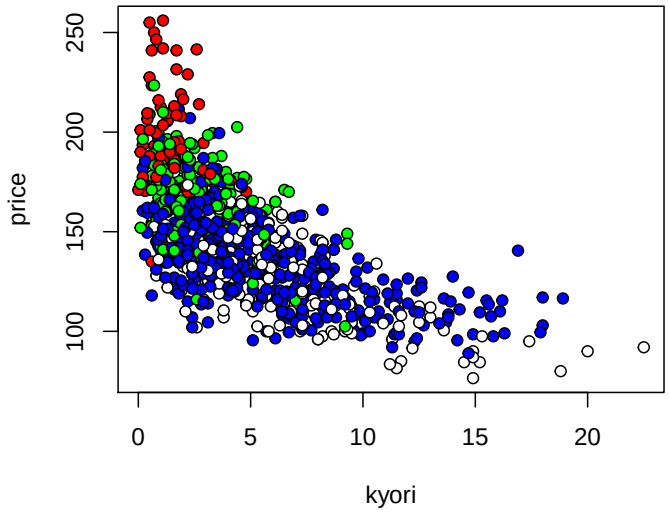
```
> summary(nenshiki)
```

```
 21  22  23  24
226 521 172  81
```

年式による中古車価格の分布を色分けする： plot 関数と引数 bg、pch

plot(price~kyori) で描いた散布図に、今度は年式が入力された変数 nenshiki で色分けする。引数 pch=21 で丸でプロットを指定し、bg= でその丸の色を指定した。nenshiki が 21 なら白（ white ）、22 なら青（ blue ）、23 なら緑（ green ）、24 なら赤（ red ）で色分けするように指定されている。

```
> plot(price~kyori,pch=21,bg=c("white","blue","green","red")[nenshiki])
```



今度は、だいぶ差があるようだ！

nenshiki を説明変数に加える

21 年式の車 (白丸）だけの回帰直線を次で引く。y_i は i 番目の車の価格 price で、x_i は i 番目の車の走行距離 kyori として以下で表す。

$$y_i = a + b_1 x_{1i}$$

22 年式の車（青丸）については次の回帰曲線を引く。22 年式であることによって、21 年式よりも高くなるぶんを b_{31} で表す。

$$y_i = a + b_1 x_{1i} + b_{31}$$

23 年式の車（緑色の丸）については次の回帰曲線を引く。23 年式であることによって、21 年式よりも高くなるぶんを b_{32} で表す。

$$y_i = a + b_1 x_{1i} + b_{32}$$

24 年式の車（緑色の丸）については次の回帰曲線を引く。24 年式であることによって、21 年式よりも高くなるぶんを b_{33} で表す。

$$y_i = a + b_1 x_{1i} + b_{33}$$

i 番目の車が、

21 年式だったら $x_{31i} = 0, x_{32i} = 0, x_{33i} = 0$、

22 年式だったら $x_{31i} = 1, x_{32i} = 0, x_{33i} = 0$、

23 年式だったら $x_{31i} = 0, x_{32i} = 1, x_{33i} = 0$、

24 年式だったら $x_{31i} = 0, x_{32i} = 0, x_{33i} = 1$、

とすると、上の４つの式を次のひとつの式で表すことができる。

$$y_i = a + b_1 x_{1i} + b_{31} x_{31i} + b_{32} x_{32i} + b_{33} x_{33i}$$

	price:y	kyori:x_1	nenshiki	x_{31}	x_{32}	x_{33}
1	241.0	0.6	21	0	0	0
2	124.5	3.0	22	1	0	0
3	128.5	3.7	23	0	1	0
4	146.5	3.6	24	0	0	1

nenshiki を加えて OLS 回帰する： lm 関数

21〜24 年式（白丸、青丸、緑丸、赤丸）それぞれの回帰直線を OLS で推定する。

$y = a + b_1 x_1 + b_{31} x_{31} + b_{32} x_{32} + b_{33} x_{33}$ の OLS 回帰を行うには、lm 関数の最初の引数 fromula は y~x1x31+x32+x33+ というように指定するべきなのだが、R で項目変数を説明変数にした場合、上の表の nenshiki から x_{31}, x_{32}, x_{33} への変換を勝手にやって、出力として、３つのパラメータ b_{31}, b_{32}, b_{33} を推定してくれる。したがって、以下で大丈夫！

```
> o01_nenshiki<-lm(price~kyori+nenshiki)
```

推定結果を出力していないが、 o01_nenshiki に代入されている。

回帰係数を表示する: coef 関数

前述したのと同様に coef 関数を用いて、OLS 回帰の結果 o01_nehshiki から推定した回帰係数を取り出す。ここではそれを o01_nenshikic に代入している。

ここでも、前後を括弧 ( ) で囲んで、代入すると同時に結果も表示する。

```
> (o01_nenshikic<-coef(o01_nenshiki))
```

```
(Intercept)       kyori  nenshiki22  nenshiki23  nenshiki24
 153.186381  -3.898778    4.714905   22.920950   46.552076
```

５つの係数が推定された。順に、a、b_1、b_{31}、b_{32}、b_{33} の推定値。

価格と距離との散布図に４つの回帰直線を引く： plot 関数と abline 関数

21 年式の回帰直線の切片は a で 153.2、傾きは b_1 で -3.899。

$$y = 153.2 - 3.899 x_1$$

22 年式の回帰直線の切片は $a + b_{31}$ で $153.2 + 4.7 = 157.9$、傾きは同じ b_1 で -3.899。

$$y = 157.9 - 3.899 x_1$$

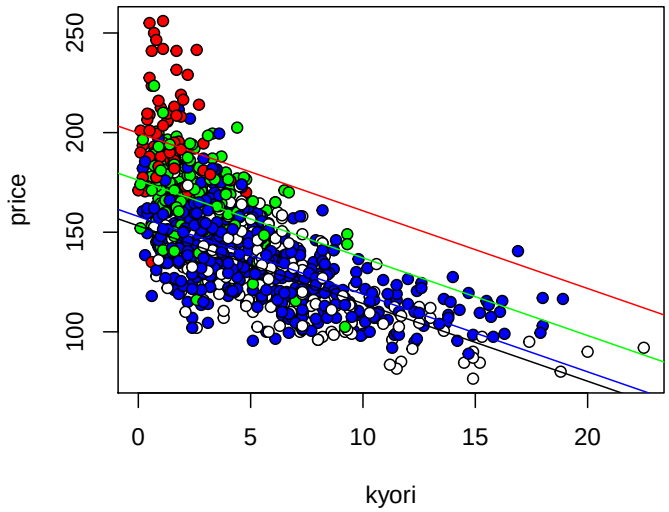
23 年式の回帰直線の切片は $a + b_{32}$ で $153.2 + 22.9 = 176.1$、傾きは同じ b_1 で -3.899。

$$y = 176.1 - 3.899 x_1$$

24 年式の回帰直線の切片は $a + b_{33}$ で $153.2 + 46.5 = 199.7$、傾きは b_1 で -3.899。

$$y = 199.7 - 3.899 x_1$$

```
> plot(price~kyori,pch=21,bg=c("white","blue","green","red")[nenshiki])
> abline(o01_nenshikic[1],o01_nenshikic[2])
> abline(o01_nenshikic[1]+o01_nenshikic[3],o01_nenshikic[2],col="blue")
> abline(o01_nenshikic[1]+o01_nenshikic[4],o01_nenshikic[2],col="green")
> abline(o01_nenshikic[1]+o01_nenshikic[5],o01_nenshikic[2],col="red")
```



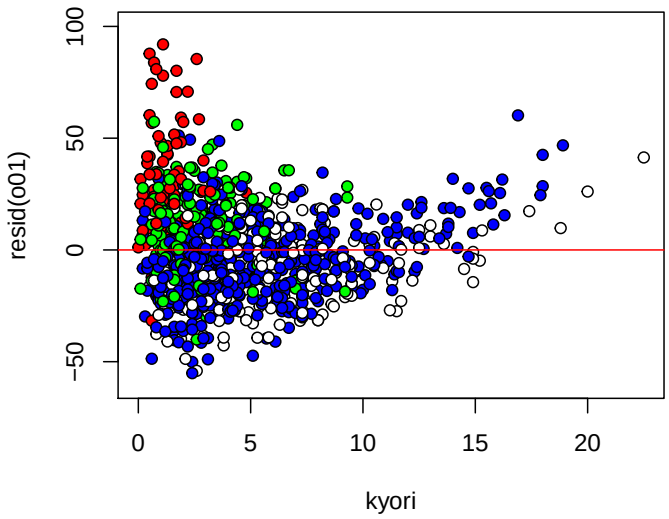
21 年式と比べて、22 年式は 4.7 万円、23 年式は 22.9 万円、24 年式は 46.5 万円、それぞれ高くなる。

残差のばらつきを確認する: resid 関数

各回帰直線からの残差をプロットすると、以下のようになる。

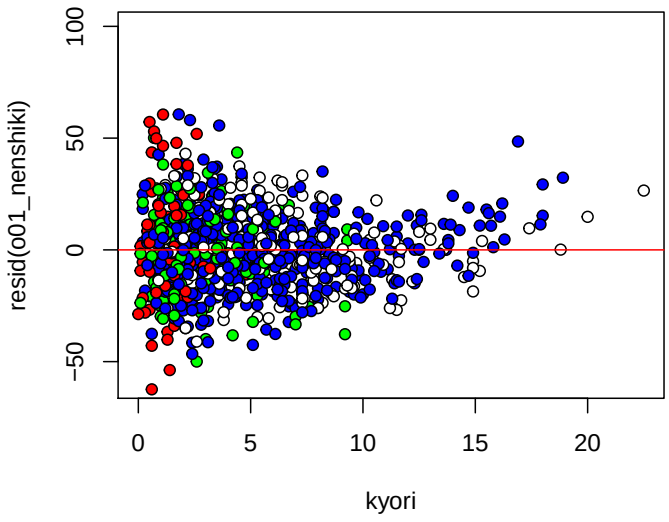
最初に、 nenshiki を加えずに、 kyori だけを回帰させた場合、それだけで説明できない価格差（残差）を resid 関数で求めて、それと距離との関係を plot 関数で描いてみた。ただし、次との比較ができるように縦軸のスケールを引数 ylim= で -60 から 100 の間に固定した。

```
> o01<-lm(price~kyori)
> plot(resid(o01)~kyori,
+      pch=21,bg=c("white","blue","green","red")[nenshiki],
+      ylim=c(-60,100))
> abline(0,0,col="red")
```



次に、nenshiki を加えて回帰した場合の４つの回帰線からの差（残差）を, 同じく resid 関数で求めて、 plot 関数で kyori との関係を描いた。縦軸は、上の分布図と同じ -60 から 100 の間に固定した。

```
> plot(resid(o01_nenshiki)~kyori,
+      pch=21,bg=c("white","blue","green","red")[nenshiki],
+      ylim=c(-60,100))
> abline(0,0,col="red")
```



年式の違いで説明できる部分が差し引かれて、だいぶ散らばりが小さくなったように見える。もっと他の変数でもやってみよう（つづく）

2.4　ダミー変数で基準となる項目を指定する

ダミー変数で基準となる項目を指定する

2.4.1　グレードによる価格差を推定する

前回の作業結果

ここからはじめる人のために、今回の作業に必要な作業をここでやっておく。

```
> d01<-read.csv("prius.csv")
> d01$NV<-factor(d01$NV)
> d01$nenshiki<-factor(d01$nenshiki)
> attach(d01)
```

グレードのデータ grade を確認する: summary 関数

プリウスのグレードは、S、G、L の３種類があり、データフレーム d01 にある grade の名前でデータが入力されている。

summary 関数で要約を出力する。

```
> summary(grade)
```

```
  G   L   S
151 105 744
```

grade は G 、 S 、 L と文字で入力されているので、R では、この変数は項目変数としてと扱っている。項目変数なので、 summary 関数による要約の出力は、それぞれの度数となる。

プリウスのグレードは、基本モデルが S でこれが大半（744)。 G は、足回りを固めたもの（たぶん、、、)。 L は軽量化して、燃費性能を高めたもの。

グレードのデータ grade を加えて OLS で回帰する： lm 関数

車体価格 price を走行距離 kyori と grade で回帰し、結果を o01_grade に代入する。ついでに、o01_gadec に推定係数を代入し、全体を ( ) で囲って同時に出力もする。

```
> o01_grade<-lm(price~kyori+grade)
> (o01_gradec<-coef(o01_grade))
```

```
(Intercept)       kyori      gradeL      gradeS
 177.580120   -5.162194  -23.463670   -7.923300
```

(Ingercept) が切片、 kyori が走行距離の推定係数、 gradeL はグレード L である場合のグレード G に比べて高く／安くなる値である。推定係数が -23.463670 と、マイナスがついているので、グレード L はグレード G よりも 23.464 万円安くなっていることを示している。 gradeS の係数は -7.923300 となっているので、グレード G よりも 7.923 万円安くなっていることがわかる。

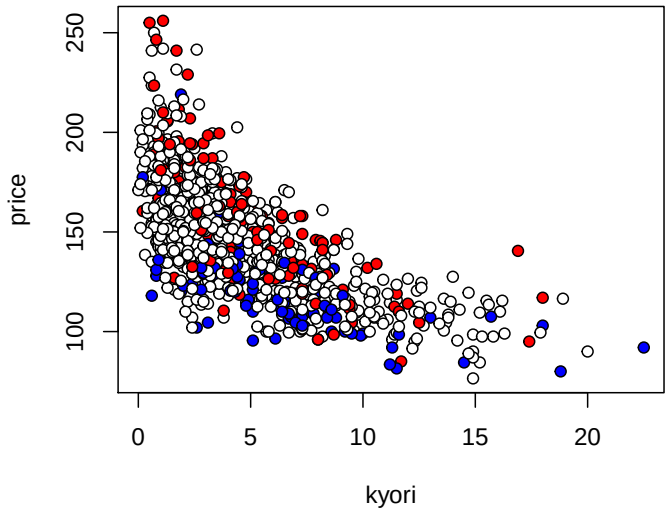
grade で散布図を色分けする： plot 関数の引数 pch= と bg=

グレード S を白（ white)、 G を赤（ red)、 L を青（ blue ）で塗り分ける。

ちなみに、色は plot 関数の引数 bg=c("red","blue","white")[grade] と指定しているが、これは項目変数 grade を構成する項目（ G 、 L 、 S ）の順番で、それぞれ c("red","blue","white") と色分けして

くれ、という意味となる。項目の順番（ G 、 L 、 S ）は、 特に指定しない限り 、アルファベット順に並べられる。（正確には、ASCII コード順。ASCII コードがわからない人は、ネットで調べてね）

```
> plot(price~kyori,pch=21,bg=c("red","blue","white")[grade])
>
```

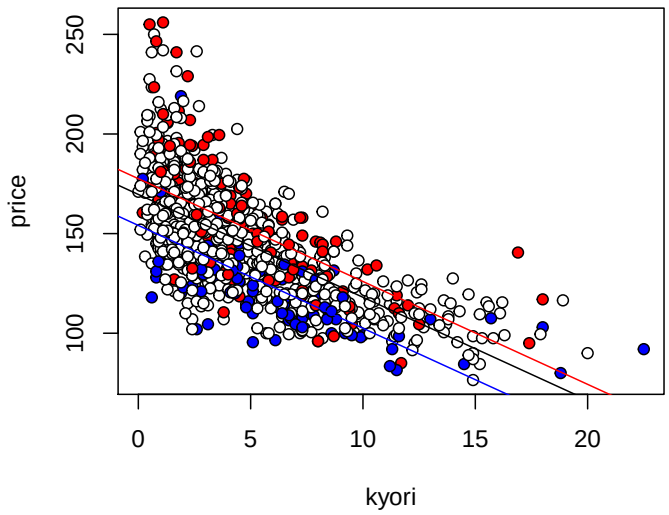


回帰直線を書き込む: abline 関数

グレード S を黒線で、グレード G を赤線で、グレード L を青線で、それぞれの散布図に回帰直線を書き込んだ。（ただし、傾きは３本とも同じ）

abline(a,b) で、切片 a 、傾き b の直線を書き込む。回帰係数の推定値は、 o01_gradec に代入されており、その 1 番目がグレード G の切片、2 番目が傾き（変数 kyori の係数)、3 番目がグレード L の切片のグレード G との差分、4 番目がグレード S とグレード G との差分。

```
> plot(price~kyori,pch=21,bg=c("red","blue","white")[grade])
> abline(o01_gradec[1],o01_gradec[2],col="red")                #Grade G
> abline(o01_gradec[1]+o01_gradec[3],o01_gradec[2],col="blue")#Grade L
> abline(o01_gradec[1]+o01_gradec[4],o01_gradec[2])            #Grade S
```



この回帰は、基準をグレード G に置いて、それよりもグレード L とグレード S がどれだけ安いかを推測したことになる。

項目変数 grade で回帰分析を行うには、本来なら、0 と 1 からなるダミー変数 x_{41} と x_{42} を使って、ダミー変数 x_{41} には、グレード L だったら 1、そうでなかったら 0 を代入し、ダミー変数 x_{42} には、グレード S だったら 1、そうでなかったら 0 を代入し、以下の関係式で回帰分析を行う必要がある。(a_1 は切片、b_1 は kyori の係数)

$$y = a + b_1 x_1 + b_{41} x_{41} + b_{42} x_{42}$$

R は、この面倒な作業を自動でやってくれるが、この場合、アルファベット順で最初の項目（この場合 G ）を、自動時に基準として選択する。つまりこの場合には、$x_{41} = 0$, $x_{42} = 0$ の回帰式となる。

しかし、プリウスはグレード S が基本モデルで、件数もこれが最も多い。これを基準に回帰分析したい。特殊なグレード G と比べて基本モデルの S は 7.9 万円安い、というのはどうも日本語的にもうれしくない。さらに、特殊なグレード G と比べて特殊なグレード L は 23.5 万円安いというのは、何のことかわからない。

項目変数の項目順を変更する： factor 関数の引数 levels=

基本モデルのグレード S を最初に持ってきて、(あとはどうでもよいが) S 、 G 、 L の順番で項目変数 grade を扱いたい。

```
> grade<-factor(grade,levels=c("S","G","L"))
```

summary 関数で度数を出力してみると、 S 、 G 、 L の順番で出力される。

```
> summary(grade)
```

```
  S   G   L
744 151 105
```

もう一度 grade を加えて OLS で回帰する： lm 関数

OLS 回帰分析の結果を o01_grade に代入し、そのうちの推定係数のみを o01_gradec に代入し、表示した。

```
> o01_grade<-lm(price~kyori+grade)
> (o01_gradec<-coef(o01_grade))
```

```
(Intercept)       kyori      gradeG      gradeL
 169.656820   -5.162194    7.923300  -15.540371
```

そうすると、gradeS というのが表示されていない。つまり、グレード S が基準となっている。

推定結果は、足回りを固めたグレード G は、基本モデルのグレード S よりも 7.9 万円安い。軽量化して燃費を高めたグレード L は基本モデルのグレード S より 15.5 万円安いということになった。（わかりやすい！）

2.5　別の数値変数を説明変数に加える

2.5.1　車検の残りの月数による価格差を推定する

前回の作業結果

ここからはじめる人のために、今回の作業に必要な作業をここでやっておく。

```
> d01<-read.csv("prius.csv")
> attach(d01)
```

車検の残りの月数 shaken を確認する: summary 関数

車検は新車の場合、最初に車検があり、3 年間有効で、その後は 2 年ごとに車検を受けなければならない。車検の費用は、自賠責保健料と自動車重量税は絶対取られて、これがだいたい 5 万円ぐらい。あとは、不具合の修理とか消耗品の交換なんかが一緒に行われて 10 万円ぐらいかかる。

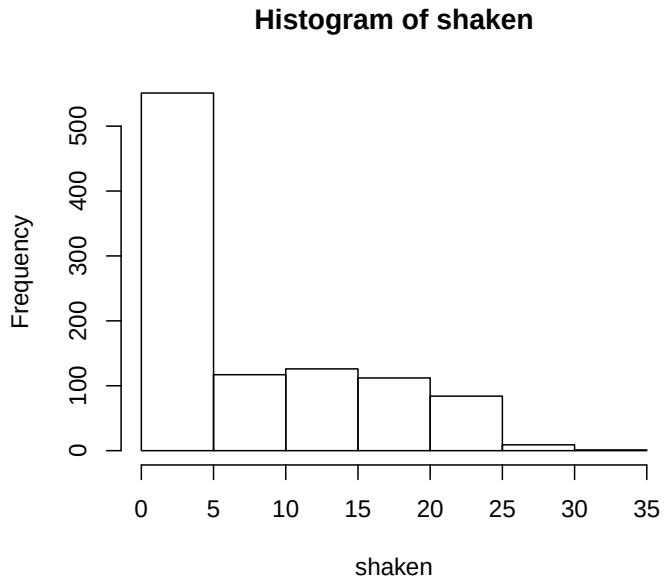
データフレーム d01 には、次回車検の月とオークション時の月の差が shaken の名前で代入されている。summary 関数で要約を見ると、これは数値変数なので、最小値 Min. 、四分の１位（小さい方から 4 分の１番目の値） 1st Qu. 、中央値 Median 、平均値 Mean 、 3rd Qu. 四分の３位（小さい方から 4 分の３番目の値)、最大値 Max. が表示される。

```
> summary(shaken)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   5.000   7.014  13.000  31.000
```

ヒストグラムを表示する： hist 関数

```
> hist(shaken)
```



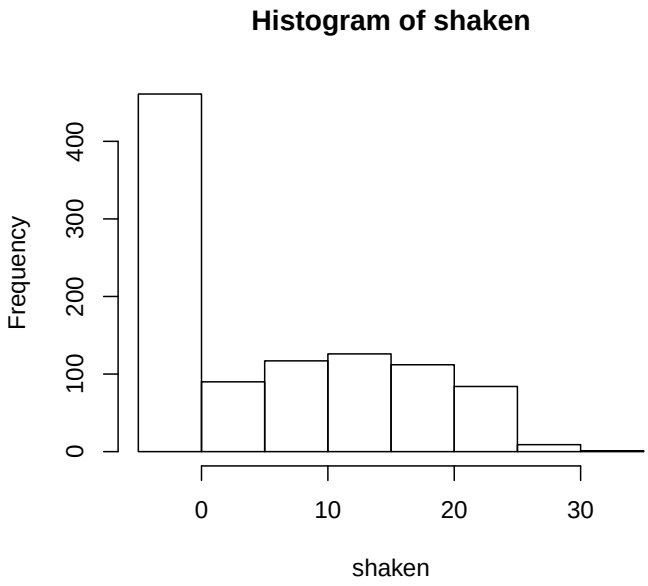
ほとんどが 0 のようだから、0 とそれ以外を分離する。

hist 関数では区切り位置を引数 breaks= （省略形 br= ）で指定することができる。 br=c(0,10,20,30) といった具合に指定すると、0 より大きく 10 以下、10 より大きく 20 以下、20 より大きく 30 以下に分けてくれ

る。実際には shaken にはマイナスの値はないが、0 だけを分離するために、区切りを-5,0,15,10,..., とした。ただ br=c(-5,0,5,10,15,... と書くのは面倒くさいので、数列を発生させる関数 seq を使った。 seq(a,b,c) とすると a ではじまり、 b で終わる数を c ごとに発生させる。たとえば、 seq(0,9,3) は 0,3,6,9。

引数 freq= は、相対度数（割合）ではなく、度数そのものを出力するように指定するもの。

```
> hist(shaken,br=c(-5,seq(0,35,5)),freq=T)
```



半分ぐらい 0、つまり車検が残っていないが、20 ヶ月を超えるものもある。

車検残月数 shaken を加えて OLS 回帰： lm 関数

数値変数を説明変数に加えること自体は、さして難しいことはない。

```
> o01_shaken<-lm(price~kyori+shaken)
```

とするだけ。

推定係数も、 coef 関数で取り出せる。（ついでに、 o01_shakenc の名前で保存しておく）

```
> (o01_shakenc<-coef(o01_shaken))
```

```
(Intercept)       kyori      shaken
161.6918575  -4.8371336   0.8583574
```

車検１ヶ月につき 0.8584、つまり 8600 円ぐらい高くなるということで、解釈にいても何の問題もない。（ただ、推定額自体は、ちょっとお高すぎるような気がするが、これつについてはまたあとで）

プロット図を描く： scatterplot3d 関数（ライブラリ scatterplot3d が必要）

説明変数として数値変数を 2 つ使うこと自体は、R で分析する上で何ら問題ではない。ダミー変数への変換などを考えなくていいので、むしろ、こちらの方が素直で、基本的な分析。

ただし、説明変数が数値変数 1 つだけだったら回帰直線が 1 本だけ、散布図に書き込めたし、数値変数 1 つと、項目変数 1 つだったら、項目変数の項目数だけの回帰直線が直線が何本か引けた。しかし、数値変数が 2 つになると、こういうことはできずに、回帰は以下の式で表される 平面 となる。

$$y = a + b_1 x_1 + b_5 x_5$$

これを散布図に書き込もうとすると、3 次元になってちょっと描きにくい。

でも、せっかくだから、やってみるか・・・

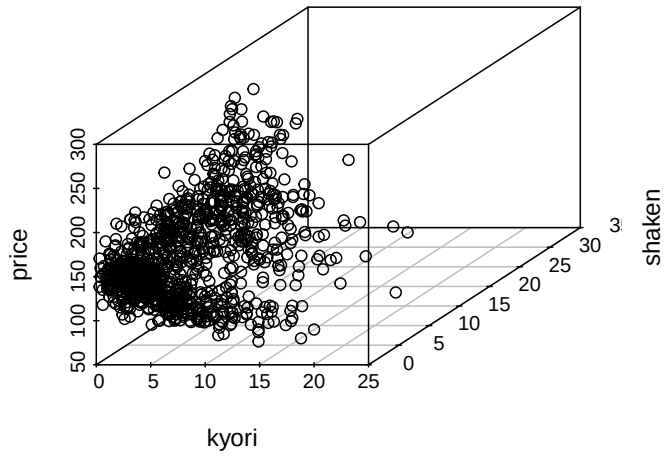
3 次元の分布を描くには、 scatterplot3d 関数を使う。ただし、これは R をインストールした段階では使えない。同じ名前のライブラリ scatterplot3d というのを読み込む必要がある。[*3]

ライブラリ scatterplot3d を読み込む

```
> library(scatterplot3d)
```

縦軸に price 、横軸に kyori と shaken のプロットを描いてみる。

```
> scatterplot3d(price~kyori+shaken)
```



回帰平面を描き入れる： scatterplot3d 関数の出力結果の plane3d

ちょっと特殊な関数の使い方だが、まず先ほど描いた散布図を適当な名前 g01 という名前で保存して、それの出力 plane3d を使う。

[*3] R のライブラリについて

R にはたくさんのライブラリがある。どんなライブラリ（パッケージ）がインストールされているかは library() と入力すると見ることができる。

インストールされていないライブラリ（パッケージ）でも簡単にインストールできる。

1. メニューの「パッケージ」から
2.「パッケージのインストール」を選択し、
3. どこかのサイト（Japan(Hyogo) とか）を選び、
4. パッケージのリストから目的のリストを選び、
5. [OK] を押す、だけ。

目的のライブラリがインストールされていたら

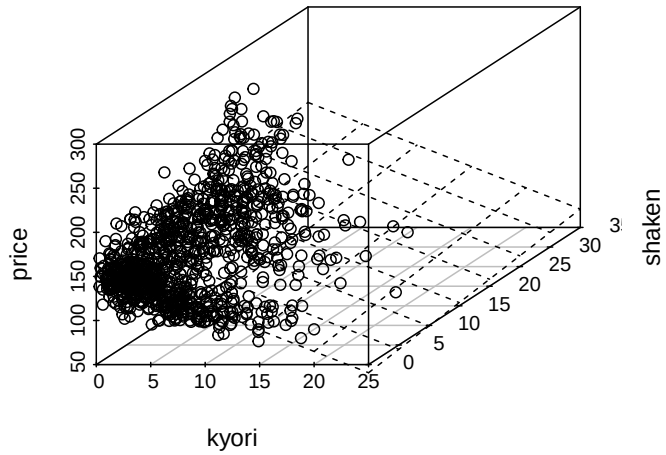
ibrary(ライブラリ名)

で読み込むことができる。

以下でもうまくいくことがある。

```
install.packages("scatterplot3d")
```

```
> g01<-scatterplot3d(price~kyori+shaken)
> g01$plane3d(o01_shakenc)
```



この図を描いたからといって、何がわかったというわけではないが、一応、イメージということで・・・

2.6　直線ではなく「曲線」をあてはめたい

2.6.1　対数をとった説明変数

前回の作業結果

ここからはじめる人のために、今回の作業に必要な作業をここでやっておく。

```
> d01<-read.csv("prius.csv")
> attach(d01)
```

改めて散布図を描く： plot 関数、 lm 関数、 abline 関数

もう一回、走行距離 kyori だけの回帰を考える。

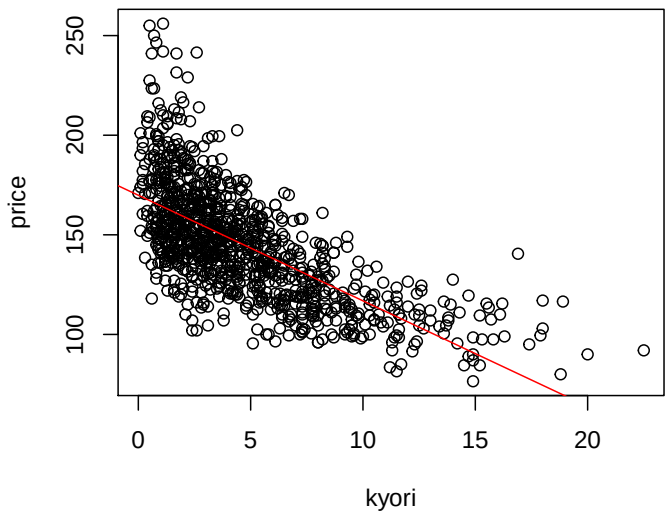
車体価格 price を y、走行距離 kyori を x_1 として、これまで、

$$y = a + b_1 x_1$$

という回帰直線のあてはめを考えてきた。

以前描いたプロット図をもう一回描いてみよう。

```
> plot(price~kyori)
> o01<-lm(price~kyori)
> abline(o01,col="red")
```



走行距離が短いと極端に価格が高くなるが、10 万 km 以上とかになると、ちょっとぐらい走行距離が伸びても、あんまり価格は変わっていないような気がする。

走行距離 kyori を対数変換する： log 関数

回帰分析で直線に無理がある場合は、変数の対数をとって表すことがある。

kyori の対数をとって

$$y = a + b_1 \ln x_1$$

を考えよう。ln というのは、底が $e = 2.718281828459045$ である対数のことで、 自然対数 と呼ばれる。

R で log 関数といえばこの自然対数のことで、たとえば、0.5 という値の対数をとると

```
> log(.5)
```

```
[1] -0.6931472
```

1 の対数は 0

```
> log(1)
```

```
[1] 0
```

0 の対数をとると、マイナス無限大となる。

```
> log(0)
```

```
[1] -Inf
```

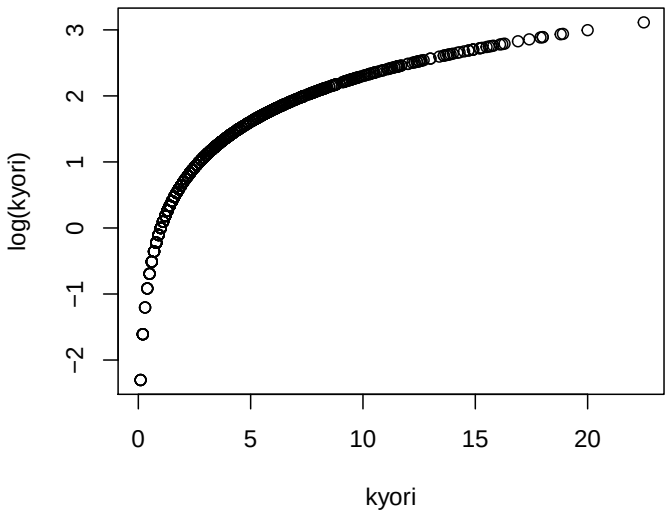
0,1,2,3,10 の対数をいっぺんに求めることもできる。

```
> log(c(0,1,2,3,10))
```

```
[1]       -Inf 0.0000000 0.6931472 1.0986123 2.3025851
```

kyori の対数変換をすると、

```
> plot(kyori,log(kyori))
```

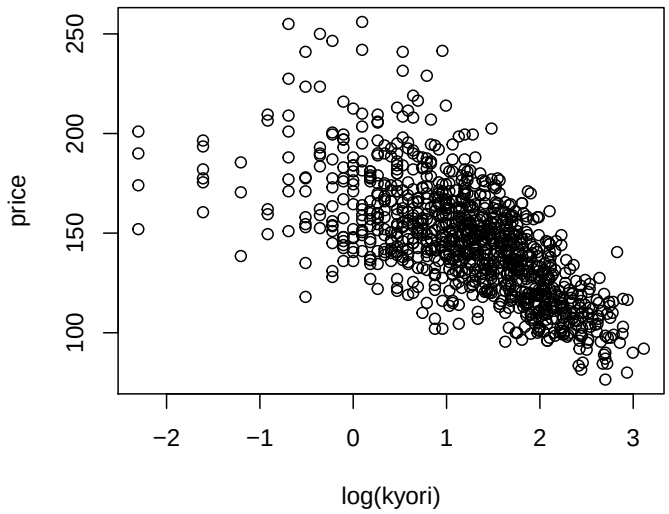


対数をとると、走行距離が小さいところ（1 万 km 以下とか）では、わずかな差を大きく見積もってくれて、走行距離が大きいところ（10 万 km 以上とか）では、大きな差を小さく見積もってくれるから都合がよい。

散布図を対数軸で表示する: plot 関数の引数 log=

実際、 price と kyori の対数値の散布図をプロットすると、

```
> plot(price~log(kyori))
```

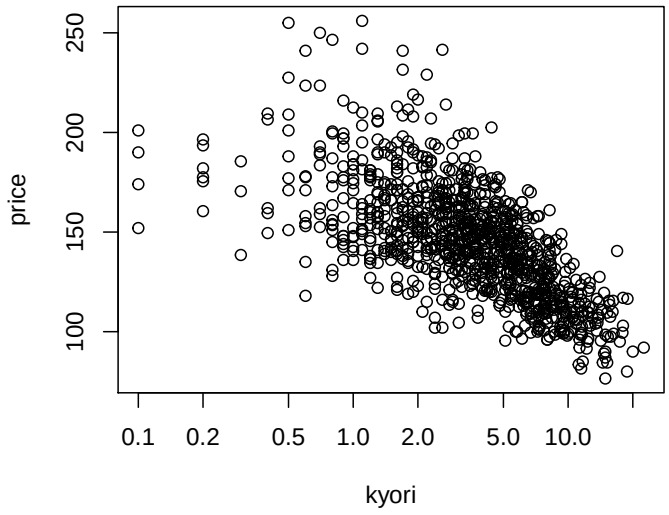


kyori の対数をとったことで price との関係が直線的となった。

kyori を対数変換しないで、軸そのものを対数スケールにしても同じ図が書ける。

x 軸（横軸）を対数表示するには引数に log=”x” とする。（ちなみに、y 軸（縦軸）を対数表示にするには log=”y” 、両方とも対数表示にするには log=”xy” とする。）

```
> plot(price~kyori,log="x")
```

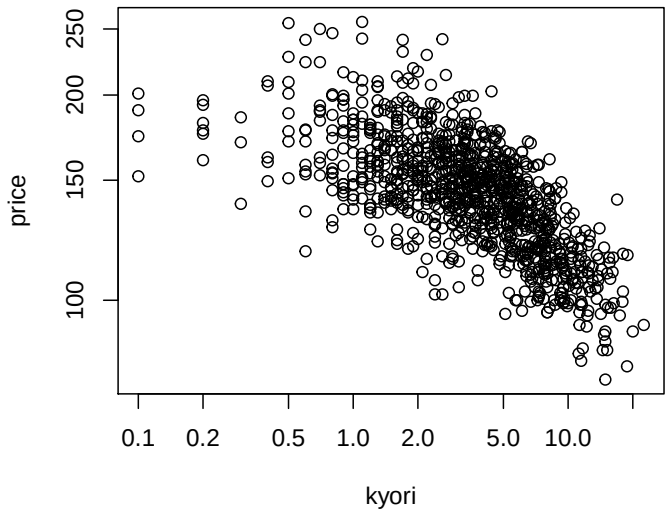


さっきの図と分布は全く同じだが、目盛り表示が違うことに注意！

y 軸も対数とってみたらどうなるか、やってみよう。

`plot(log(price)~log(kyori))` ということなんだが、軸そのものを対数表示しても同じことだし、そっちの方が簡単。

```
> plot(price~kyori,log="xy")
```



price も対数とると、逆に右上にわん曲してしまった。対数とるのは、 kyori だけでよさそうだ。

欠損値のあるデータを除く: na.omit

ところで、さっきから何か warning が出て、怒られている。$>$ 1 個の x の値が 0 以下なもんで、対数表示のグラフからはずした

、と。

kyori の中に 0 をとるデータ（走行距離が 0 の中古車？）が含まれているようだ。summary 関数で確認してみると、最小値（Min.）が 0 となっている。

```
> summary(kyori)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   2.000   3.600   4.643   6.400  22.500
```

この場合、 kyori の対数をとると、この値がマイナス無限大 -Inf となるので、グラフ描いたり、回帰分析するのに邪魔になる。

このような場合は、 kyori[kyori==0]<-0.0000001 とか、分析に影響がないような小さな値を入れて対処することができる。

しかし、だいたいこういうデータは、おかしなデータであることも多いので、確認してみる。

kyori が 0.1 以下のデータを表示してみる。

```
> d01[kyori<=0.1,]
```

```
    nenshiki kyori         kizu price grade selection shaken    color NV SR
418       24   0.1 すごくきれい   201     S       non     25       白  1  0
530       24   0.0   修復歴あり   171     S       non     24       黒  1  0
672       24   0.1 すごくきれい   190     S       non     21       黒  1  0
888       23   0.1 すごくきれい   174     S       non     17       白  1  0
991       23   0.1   修復歴あり   152     S       non     17 シルバー  0  0
    kawa
418    0
530    0
```

```
672    0
888    0
991    0
```

530 番目のデータの kyori が 0.0 となっている。

しかし、この車、走行距離が 0 なのに「修理歴あり」？　・・・なんか怪しい。

取引データのように細かいデータには、こういう怪しげなデータ（異常値 と言います）が含まれることがよくあり、分析の途中で気づくことがある。

そのような場合は、これを欠損値（NA）としたり、該当するデータセットそのものを除外して分析すべきだ。

d01 に NA を代入すると、これを欠損値として扱ってくれる

```
> d01$kyori[kyori==0]<-NA
```

もう一回、 kyori が 0.1 以下のデータを表示してみる

```
> d01[kyori<=0.1,]
```

```
    nenshiki kyori         kizu price grade selection shaken    color NV SR
418       24   0.1 すごくきれい   201     S       non     25       白  1  0
530       24    NA   修復歴あり   171     S       non     24       黒  1  0
672       24   0.1 すごくきれい   190     S       non     21       黒  1  0
888       23   0.1 すごくきれい   174     S       non     17       白  1  0
991       23   0.1   修復歴あり   152     S       non     17 シルバー  0  0
    kawa
418    0
530    0
672    0
888    0
991    0
```

530 番目のデータの kyori が NA となっている。

R の場合、NA が入っていると、関数によっては、自動的にこれを省いて分析したり、グラフを描いたりすることができるので、そのままにしておくことも多い。（NA が含まれるデータセット（行）全体を削除するのは、NA 以外のデータが使えのでもったいない）

それでも、このデータのように、データセット全体を除外したほうがよさそうな場合は、関数 na.omit を使う。

```
> d01<-na.omit(d01)
```

こうすると欠損値 NA が含まれるデータセット（行）全体を削除する。

d01 を書き換えたので、 attach し直す。

```
> detach(d01)
> attach(d01)
```

もう一回、 kyori が 0.1 以下のデータを表示してみると、530 番目のデータが削除されている。

```
> d01[d01$kyori<=0.1,]
```

```
    nenshiki kyori         kizu price grade selection shaken    color NV SR
418       24   0.1 すごくきれい   201     S       non     25       白  1  0
672       24   0.1 すごくきれい   190     S       non     21       黒  1  0
888       23   0.1 すごくきれい   174     S       non     17       白  1  0
991       23   0.1   修復歴あり   152     S       non     17 シルバー  0  0
    kawa
418    0
672    0
```

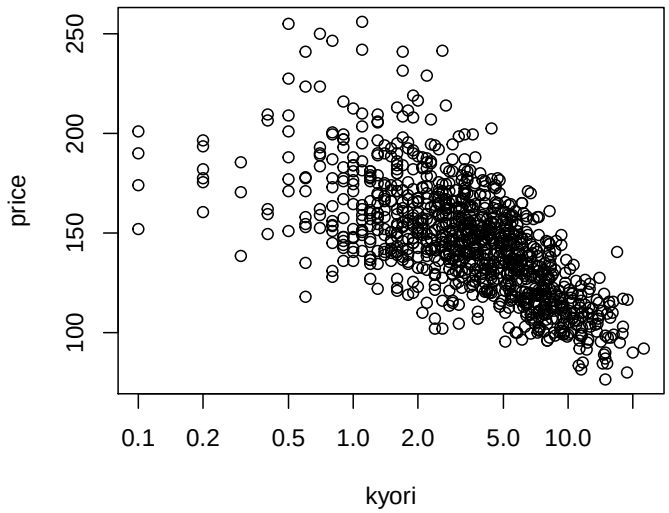
```
888     0
991     0
```

１個ヘンなデータが入っていたために、手間どってしまったが、話をもとに戻そう・・・(実際の統計分析では、こうした欠損値とか、異常値の処理を行う手間が結構ある！)

横軸が対数の散布図を描く： plot 関数で引数 log="x"

もう一回、 kyori を対数軸で表した散布図を描く。

```
> plot(price~kyori,log="x")
```



こんどは Warning で怒られない。

対数をとった説明変数で OLS 回帰する: lm 関数

kyori を対数変換して説明変数として使うが、 log(kyori) とするだけ

OLS 回帰の結果を o01_ln という名前で保存した。

```
> o01_ln<-lm(price~log(kyori))
```

この回帰で推定された係数を o01_lnc に代入して、結果も表示する

```
> (o01_lnc<-coef(o01_ln))
```

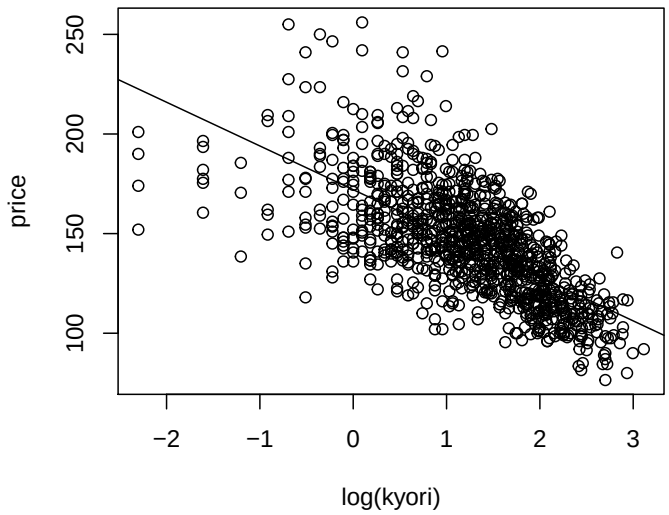
```
(Intercept)  log(kyori)
  172.04481   -21.95006
```

$$y = 172.04 - 21.95 \ln x$$

という回帰線が推定された。

散布図に回帰線を描き入れよう

```
> plot(price~log(kyori))
> abline(o01_ln)
```

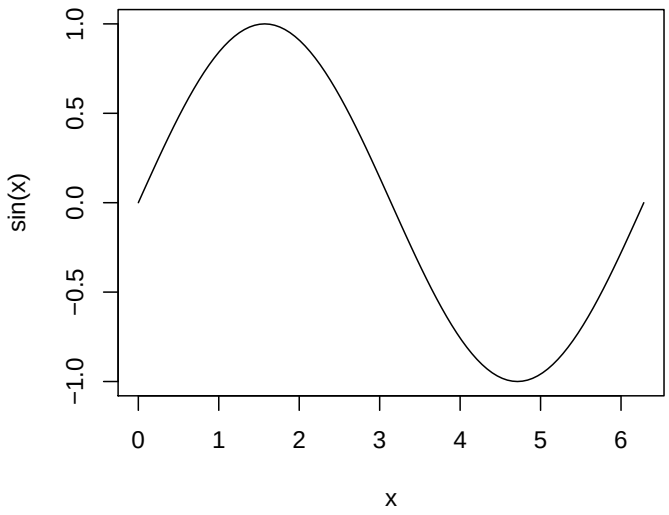


ちょっと、 kyori の小さい値にひっぱられ過ぎている気もするが、その話はしばらく置いておいて・・・
これ、対数でない軸で、グニュっと曲がった $y = a + b \ln x$ の形の回帰線を書き込みたいんですが・・・
ちょっと、ややこしいけど、やるか・・・・

1 変数関数の曲線を描く： curve 関数

例えば、$y = \sin x$ のグラフを $[0, 2\pi]$ の間で描きたければ、R で、$\sin x$ は、 sin(x)

```
> curve(sin,0,2*pi)
```



とすればよい。

$y = \ln x$ だと、$[0.1, 20]$ の区間で、次で描ける。

```
> curve(log,0.1,20)
```

それでは、$y = a + b\ln x$ はどうするかというと、新しく $f(x) = a + b\ln(x)$ という関数をつくってやればよい。

新しい関数をつくる: function 関数

R では function という関数を使って、勝手に自分で関数をつくることができる。1行で書ける 1 変数（引数）の関数から、数百行にわたる何個も引数のある関数まで、自由自在に定義できる。R を使う上で、function 関数は大変基本的で、しっかり理解してほしいが、ここでは、ひとまず簡単な使い方だけ。

例えば、$f(x) = 1 - 2\ln x$ という関数を、 f01 という名前で定義したければ、次のようにする。

関数名 f01 に function 関数の定義を代入するという形をとる。 { } の中に定義したい式を書き、その式の中で、外から与えたい変数を引数として function の後の ( ) 内に示しておく。

```
> f01<-function(x){
+       1-2*log(x)
+ }
```

このように定義した関数 f01 は、普通の関数のように $f01(x)$ という具合に使うことができる。つまり、$1 - 2 * \ln 3$ は？

```
> f01(3)
```

```
[1] -1.197225
```

で計算できる。

```
> 1-2*log(3)
```

```
[1] -1.197225
```

とちゃんと一致している。

ちなみに、$f(x, a, b) = a + b\ln x$ で、a も b も外から与えたかったら、

```
> f01<-function(x,a,b){
+   a+b*log(x)
+ }
```

とする。新しくつくった関数 f01 で、$1 - 2\ln 3$ を計算するには、次のように引数を与える。

```
> f01(3,1,-2)
```

```
[1] -1.197225
```

正式には、 f01(x=3,a=1,b=-2) とするのが正しい。このように、ちゃんと引数名まで指定すると、特に引数を与える順番は気にしなくてもよい。

```
> f01(b=-2,a=1,x=3)
```

```
[1] -1.197225
```

定義した順番で、引数を与える限り、引数名の一部またはすべてを省略できる。次は x= のみ順番が一致しているので引き数名を省略しているが、 a= と b= は順番が違うので、引数名まで記入してある。

```
> f01(3,b=-2,a=1)
```

```
[1] -1.197225
```

さて、話を本題に戻そう・・・

推定された回帰線 $Y = a + b \ln x$ を、を散布図に書き込むのであった。

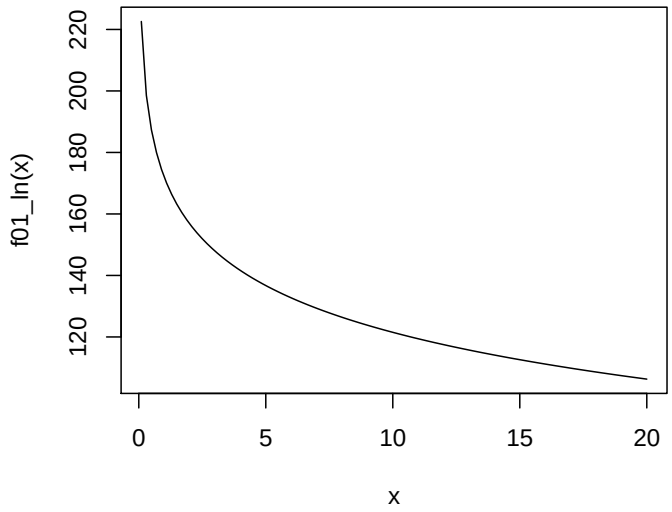
a の推定値は o01_lnc[1] に入っているし、b の推定値は o01_lnc[2] に入っているから、この関数を、

```
> f01_ln<-function(x){
+          o01_lnc[1]+o01_lnc[2]*log(x)
+ }
```

で定義しておく。

ちょっとこの関数の形を $[0.1, 20]$ の範囲でを描いてみよう。

```
> curve(f01_ln,0.1,20)
```



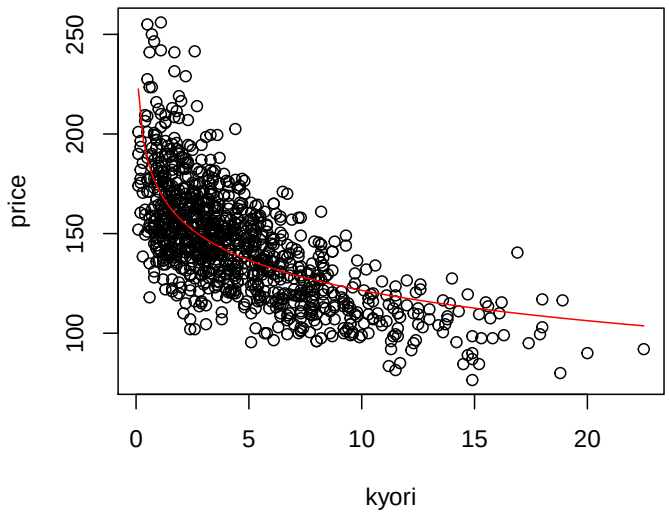
曲線を散布図に重ね描きする： curve 関数の引数 add=T

関数 curve で、散布図に重ね描きするには、 plot 関数で散布図を描いた後に、引数 add=T"を加えて描けばよい。

ついでに, 線の色も引数 col="red" で赤くした。

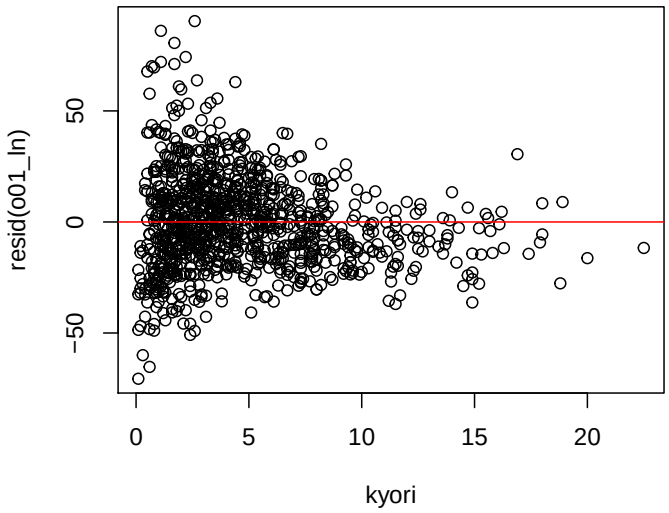
ここで、 curve(f01_ln,0.1,20) のように、区間の指定がない。これは、正確には curve(f01_ln,from=0.1,to=20) と表現すべきところを、省略した表現なのだが、既存の散布図に重ね描きする場合は、その x 軸の範囲で曲線を描いてくれて、区間指定の from=0.1,to=20 という引数が必要ない。

```
> plot(price~kyori)
> curve(f01_ln,add=T,col="red")
```



やっと描けたが、やはりちょっと、走行距離の小さいところのデータにひっぱられ過ぎている？
残差も表示してみる

```
> plot(resid(o01_ln)~kyori)
> abline(0,0,col="red")
```



他の曲線も試してみよう・・・

2.6.2 説明変数を２次式で表してみる

曲線といえば、$y = a + bx + cx^2$ といった２次式でも表されるだろう。これで OLS 回帰してみよう。

二次式で OLS 回帰： lm 関数の引数 fromula= で I(x2̂)

$y = a + bx + cx^2$ といった回帰線を推定する場合、 `lm(y~x+I(x^2))` と指定する。[*4]

kyori の２次式で OLS 回帰した結果を `o01_2` に代入する

```
> o01_2<-lm(price~kyori+I(kyori^2))
```

推定した係数を `o01_2c` に代入し、同時に結果も表示。

```
> (o01_2c<-coef(o01_2))
```

```
(Intercept)       kyori  I(kyori^2)
178.8428894  -9.2869968   0.2768463
```

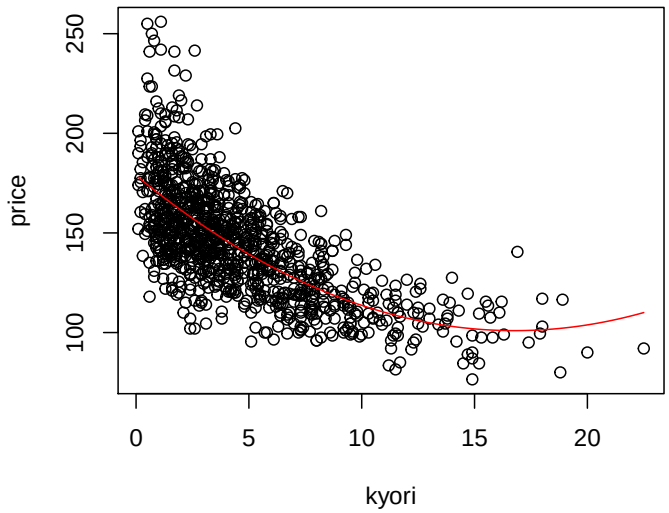
推定した２次式を１引数の関数で表す： function 関数

散布図に回帰した２次曲線を書き込むために、対数変換のときと同じように、２次式についても、推定した係数の値 `o01_2c` を使って、新しい関数 `f01_2` を定義する。

```
> f01_2<-function(x){
+   o01_2c[1]+o01_2c[2]*x+o01_2c[3]*x^2
+ }
```

散布図に回帰した２次曲線を描き込む： curve 関数

```
> plot(price~kyori)
> curve(f01_2,add=T,col="red")
```

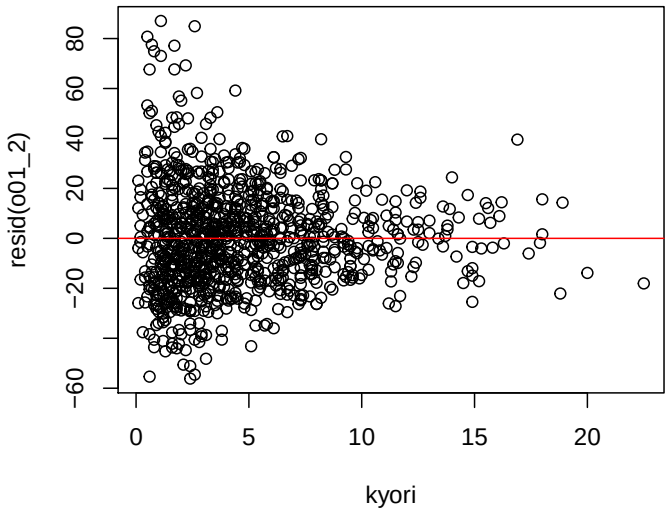


15 万 km 超えたぐらいから右上がりになってしまったが、こっちがまだましか？

残差もプロットしてみよう。

[*4] lm 関数の formula= の部分（ `y~x1+x2` とかの部分）では、 `+`、 `-`、 `*`、 `^`、それに : に特別な意味を持たせており、普通に四則演算にはならない。四則演算したい場合は `I(x+z)` のように `I()` で囲む。

```
> plot(resid(o01_2)~kyori)
> abline(0,0,col="red")
```



残差が 0 のまわりにバランス良くちらばている。

2.6.3 いろんな曲線のあてはまりを比較する

kyori で price を回帰したが、直線、対数、二次式の３種類で回帰してみた。そのうちのどれがよいかをどう判断するか？

最もよく使われる判断基準に、赤池の情報量基準（AIC) というのがある。この値が最も小さいものがあてはまりがよいとされる。

AIC が何者かについての説明は、後の方にまわしておいて、今回はひとまず、計算だけしてみる。

AIC を計算する： AIC 関数

AIC を計算するには、OLS 回帰の出力結果を引数とした AIC 関数が使える。

直線（$y = a + bx$）で回帰した結果は、 o01 に出力していた。

```
> AIC(o01)
```

```
[1] 8936.651
```

対数（$y = a + b \ln x$ で回帰した結果は、 o01_ln に出力していた。

```
> AIC(o01_ln)
```

```
[1] 8938.754
```

2 次式（$y = a + bx + cx^2$）で回帰した結果は、 o01_2 に出力していた。

```
> AIC(o01_2)
```

```
[1] 8865.703
```

ダントツで 2 次式のあてはまりのよさがわかる。

2.6.4　回帰係数の意味

どんな線のあてはまりがよいかは場合による。

直線：$y = a + bx$

解釈が最もわかりやすのは、直線 b は、x が 1 単位大きくなった場合の y の増加だ。

price と kyori の関係であれば、b は、走行距離が 1 万 km 伸びたときの車体価格の下落幅となる。

あてはまりでいうと、今回はあまりうまくいかなかったが、対数がよい場合が多い。

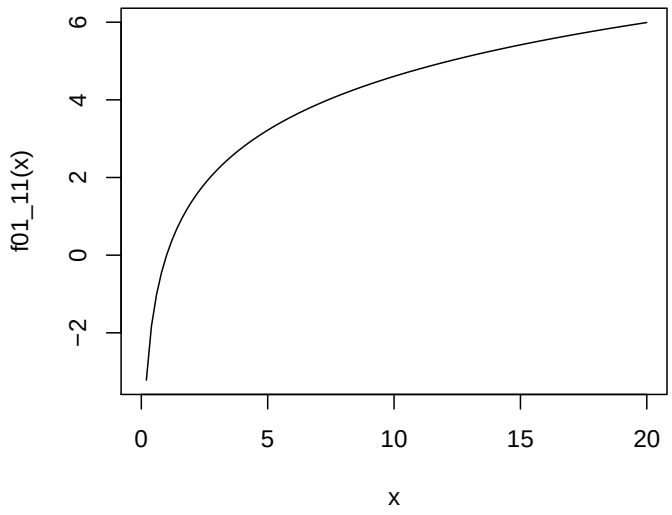
片対数：$y = a + b \ln x$

対数は 3 パターンあって、今回は この形の 片対数 を使った。

この関数形は、b が正のとき傾きが逓減する増加関数で、b がマイナスのときは、やはり傾きが逓減する減少関数となる。

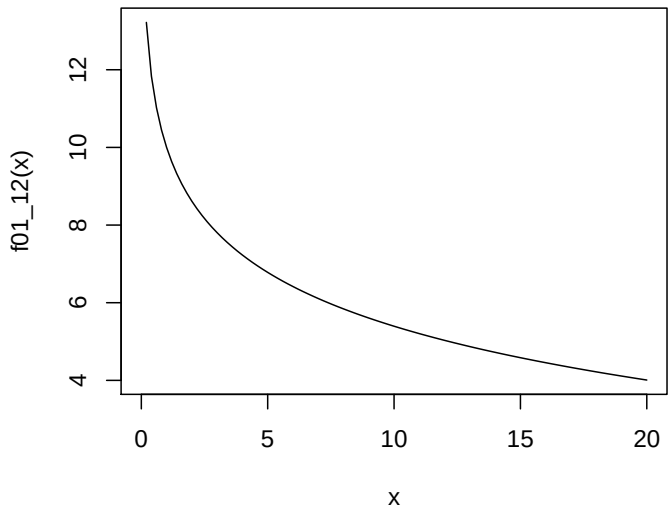
$b > 0$ の場合

```
> f01_11<-function(x){0+2*log(x)}
> curve(f01_11,0,20)
```



$b < 0$ の場合

```
> f01_12<-function(x){10-2*log(x)}
> curve(f01_12,0,20)
```

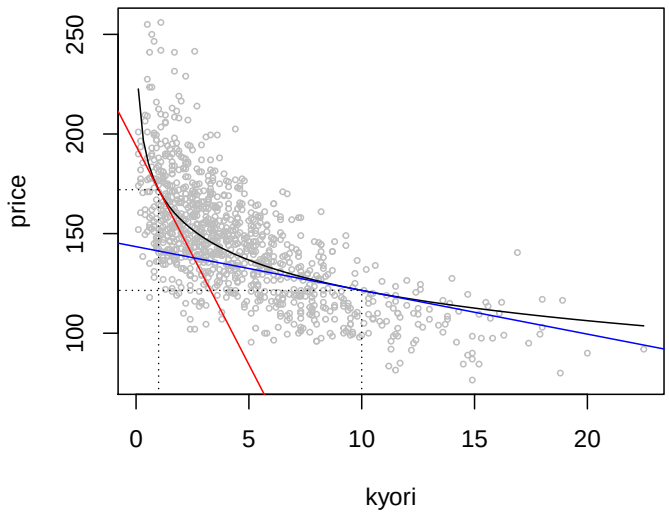


この場合、x が 1 単位大きくなった場合、y は約 b/x だけ大きくなる。($\ln x$ の傾きが $1/x$ だから・・・。「約」と書いたのは、その傾きが各 x の近くだけで成り立つから)

price と kyori との関係で言うと、走行距離が 1 万 km から 2 万 km に伸びた場合、車体価格は約 b 円だけ変化する。しかし、10 万 km が 11 万 km に伸びた場合の車体価格の変化は $b/10$ 円でしかない。[*5]

```
> f01_1b<-function(x){o01_lnc[1]+o01_lnc[2]*log(x)}
> plot(price~kyori,cex=.5,col="gray")
> curve(f01_1b,add=T)
> abline(f01_1b(1)-o01_lnc[2]/1*1,o01_lnc[2]/1,col="red")
> segments(1,0,1,f01_1b(1),lty="dotted")
> segments(-1,f01_1b(1),1,f01_1b(1),lty="dotted")
> abline(f01_1b(10)-o01_lnc[2]/10*10,o01_lnc[2]/10,col="blue")
> segments(10,0,10,f01_1b(10),lty="dotted")
> segments(-1,f01_1b(10),10,f01_1b(10),lty="dotted")
```

[*5] 上の図を描くのにまだ説明していない関数や引数を使ったが、とりあえず、ここでは傾きの様子を示したかったので、ここは気にしなくてよい。それでも、あえて説明しておくと、plot 関数の引数 cex= は、散布図のプロットの大きさを指定する。cex=.5 は、通常の 0.5 倍の大きさのプロットとなる。segments 関数は線分を描く。segments(x0,y0,x1,y1) で点 (x0,y0) と点 (x1,y1) を通る線分を描き入れる。その引数 lty="dotted" は、線の種類を点線にする。この辺りは、 http://cse.naro.affrc.go.jp/takezawa/r-tips/r/51.html を参照してください。



片対数：$\ln y = a + bx$

被説明変数だけの対数をとった片対数のパターンもある。これは、次のように変換できる。

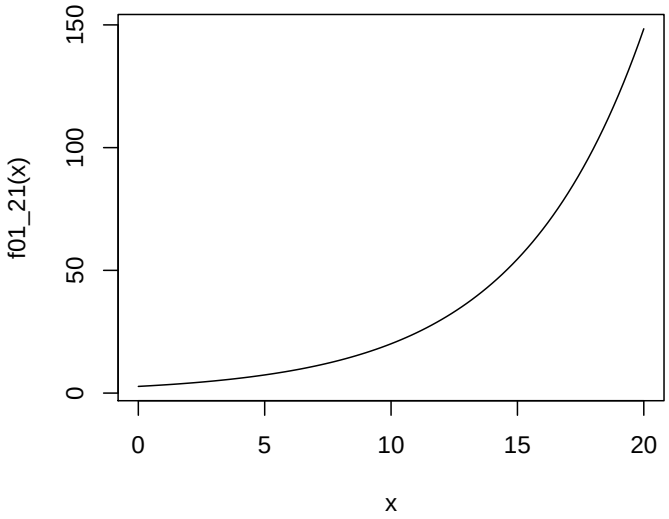
$$y = e^{a+bx} = Ae^{bx}$$

ただし、$A = e^a$。つまり、指数関数。

この関数形は、b が正のとき傾きが逓増する増加関数で、b がマイナスのときは、やはり傾きが逓増する減少関数となる。

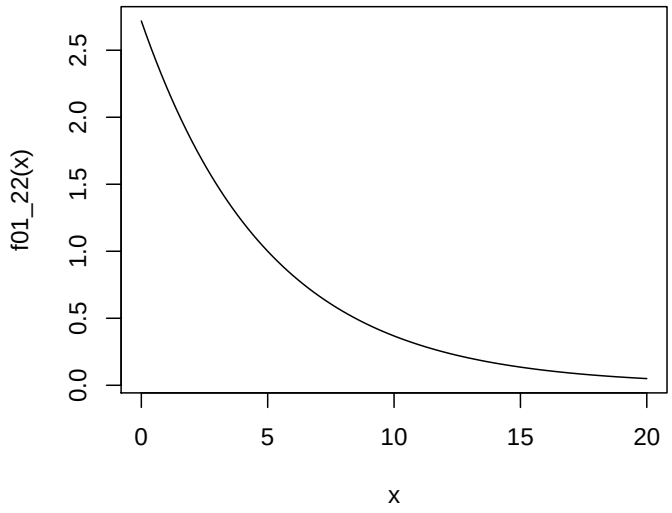
$b > 0$ の場合

```
> f01_21<-function(x){exp(1+0.2*x)}
> curve(f01_21,0,20)
```



$b < 0$ の場合

```
> f01_22<-function(x){exp(1-0.2*x)}
> curve(f01_22,0,20)
```



この場合、b は、x が１単位大きくなった場合の y の変化率（$\Delta y/y$）となる。

price と kyori との関係で言うと、走行距離が 1 万 km 伸びた場合、車体価格は約 $b \times 100$ %変化する。

両対数：$\ln y = a + b \ln x$

説明変数と被説明変数の両方の対数をとった、両対数 の回帰式もある。

これは、次のように変換できる。

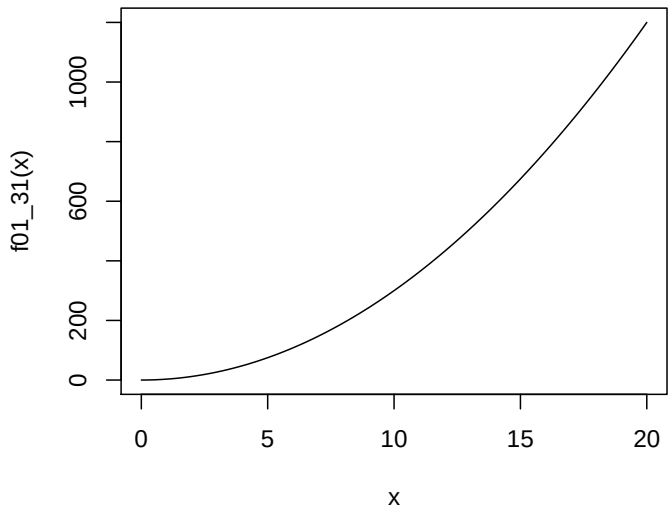
$$y = Ax^b$$

ただし、$A = e^a$。

この関数形は、$b > 1$ のとき、傾きが低増する増加関数で、$0 < b < 1$ ときは、傾きが逓減する増加関数、$b < 0$ のときは、傾きが逓減する減少関数となる。

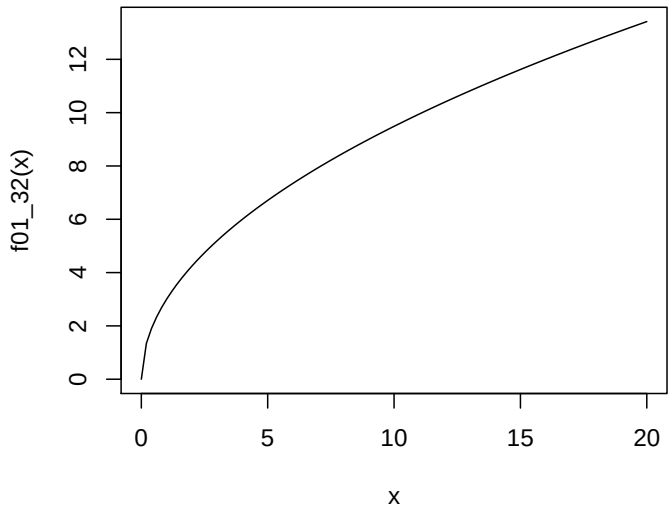
$b > 1$ の場合

```
> f01_31<-function(x){3*x^2}
> curve(f01_31,0,20)
```



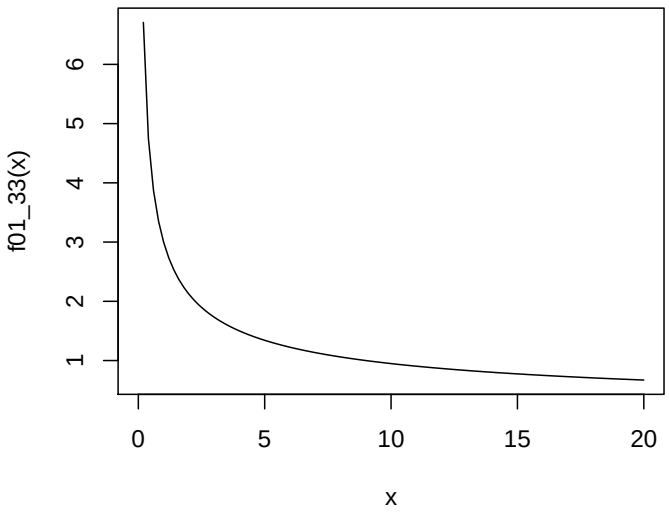
$0 < b < 1$ の場合

```
> f01_32<-function(x){3*x^0.5}
> curve(f01_32,0,20)
```



$b < 0$ の場合

```
> f01_33<-function(x){3*x^(-0.5)}
> curve(f01_33,0,20)
```



この場合、b は、x が 1 %変化したとき y が何%変化するかを示す。

price と kyori との関係で言うと、走行距離が 1 %伸びた場合、車体価格は約 b %変化する。

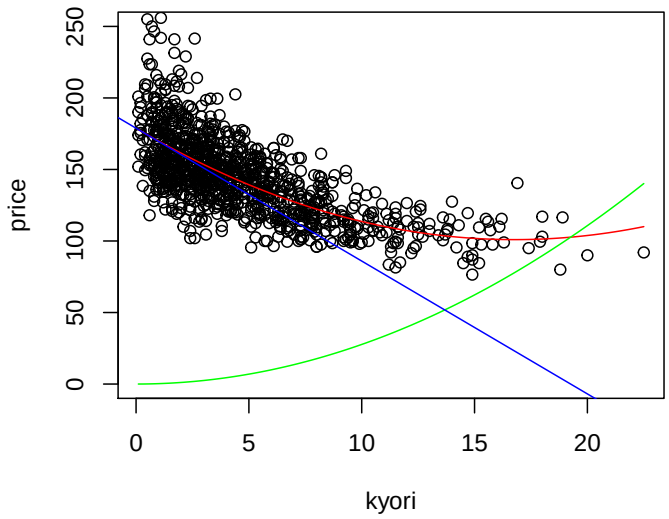
2次式：$y = a + bx + cx^2$

y は、x が1単位大きくなると、b だけ変化するが、その変化は、cx^2 で補正される。

price と kyori との関係で言うと、車体価格は、走行距離1単位に応じて直線的に b 円だけ下がるのではなく、走行距離が大きく成ると、その下がり方が（距離の2乗の c 倍分だけ）小さくなる。

下の図は、赤い線が回帰直線（$y = a + bx + cx^2$）だが、これは青の直線（$y = a + bx$）と緑の曲線（$y = cx^2$）の合計。

```
> f01_20<-function(x){o01_2c[3]*x^2}
> plot(price~kyori,ylim=c(0,250))
> curve(f01_2,add=T,col="red")
> curve(f01_20,add=T,col="green")
> abline(o01_2c[1],o01_2c[2],col="blue")
```

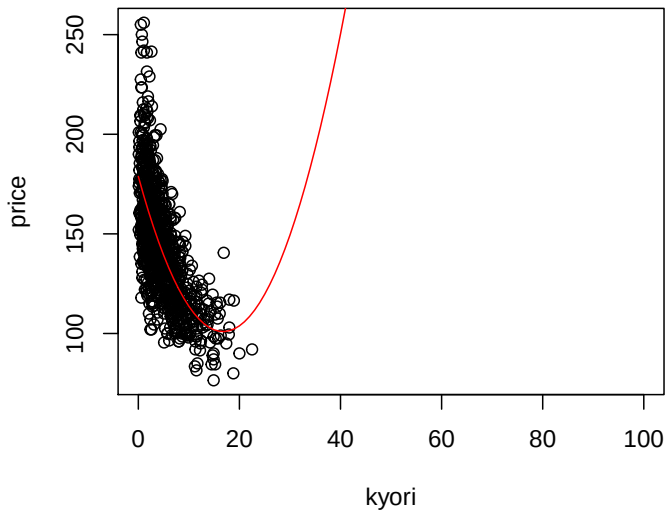


ちょっと先走るが、この回帰は、 kyori の最大値と最小値の間で（かろうじて？）成立している。この関係でうまく説明できているからといって、その他の範囲でもうまくいくとは全く限らない。

この２次式の回帰だと、100 万 km 走れば、車体価格はべらぼうに高くなってしまいます。

こうした推定に使用した説明変数の範囲を超えて、結果を予測することは、外挿 と呼ばれており、注意が必要です。

```
> plot(price~kyori,xlim=c(0,100))
> curve(f01_2,add=T,col="red")
```



2.7 係数推定値の信頼性を測る

係数の推定値の信頼性を測る

2.7.1 推定値のばらつき

前回の作業結果

念のため、今回の作業に必要な前回の作業をもう一回やっておく（前回に引き続いて作業する場合、ここは不要）

```
> d01<-read.csv("prius.csv")
> attach(d01)
```

車体の色 color の種類と度数を確認する: summary 関数

```
> summary(color)
```

```
シルバー パープル   ワイン       黒       紺       青       赤       白
     261       15       11      228       13       71       24      372
    緑
      5
```

シルバー、パープル、ワイン、黒、紺、青、赤、白、緑の９色。白が最も多くて、シルバー、黒、青と続く。そしてこれらがほとんどで、その他の色は少ない。

color で回帰する: lm 関数

color という項目変数で回帰する場合、９色もあるので、0,1 のダミー変数は８個必要。しかし、R は、これを自動的にやってくれる。その場合、 summary 関数とかで一番最初に表示される項目が基準となる。気に入らなければ facotr 関数の引数 levels= で指定できた（(６）を参照)。しかし、ここはシルバーでよいか・・・

そのまま回帰しよう。

price を kyori と color で回帰して、出力結果を o01_col に代入する。

```
> o01_col<-lm(price~kyori+color)
```

OLS 回帰で推定した係数： coef 関数

OLS 回帰の結果 o01_col から coef 関数を使って、推定された係数を取り出し、o01_colc に代入する。最初と最後を ( ) で括って、結果も同時に表示する。

```
> (o01_colc<-coef(o01_col))
```

```
  (Intercept)         kyori color パープル    color ワイン        color 黒
157.4468850    -5.1294633    13.8385885    17.1737701    18.3255317
    color 紺       color 青       color 赤       color 白       color 緑
  0.3521494    -0.9032524     6.4031487    18.7038701    -0.8025127
```

色別の回帰直線を描く： plot 関数と abline 関数

色別の回帰直線は、９本もあるので、しんどいが、やりますか・・・

色の塗り分けの指定を何回か使うので、ここで一旦 col01 という名前で保存しておく。ただし、白は、直線

が見えないので yellow としている。（ちなみに、この色の順番は、 sumary(color) で表示される色の順番で、シルバー、パープル、ワイン、黒、紺、青、赤、白、緑）

```
> col01<-c("gray","purple","violet","black","darkblue","blue","red","yellow","green")
```

散布図は plot 関数を使って、 price を縦軸に kyori を横軸に描いた。プロットの形は引数 pch=21 （枠付き丸）で、変数 color によって col01 で塗り分けた。

回帰直線は、 abline 関数で描き込む。 abline(a,b) で切片 a 、傾き b の直線を描く。

基準となったグレーの切片は、o01_colc の 1 番目に入っている。傾きは 2 番目。したがって、グレーの abline(o01_colc[1],o01_colc[2]) 。さらに、色は引数 col= で ”gray” を指定するが、これは col01 の 1 番目に入っていたので、 col=col01[1] と与えた。

その他の色の車は、

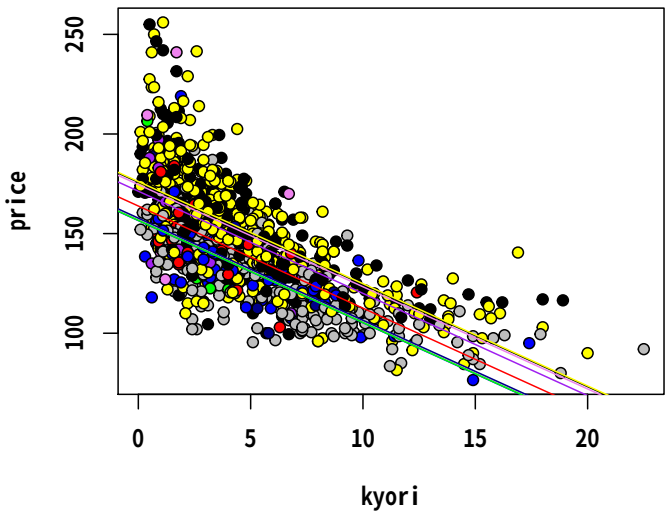
```
abline(o01_col[1]+o01_col[3],o01_col[2],col=col01[2])
abline(o01_col[1]+o01_col[4],o01_col[2],col=col01[3])
abline(o01_col[1]+o01_col[5],o01_col[2],col=col01[4])
abline(o01_col[1]+o01_col[6],o01_col[2],col=col01[5])
...
```

と書けばよいのだが、めんどうなので for 関数を使った。[*6]

```
> plot(price~kyori,pch=21,bg=col01[color])
> abline(o01_colc[1],o01_colc[2],col=col01[1])
> for(i in 2:9){
+   abline(o01_colc[1]+o01_colc[i+1],o01_colc[2],col=col01[i])
+ }
```

[*6] **for** 関数順番に繰り返し、同じような作業を実行したい場合は、for 関数を用いる。次は、ただし、 a01 （最初は 0 が入っている）に i を足すという作業を、 i に 1:5 つまり 1、2、3、4、5 を順番に代入して、繰り返し実行しなさい、という意味になる。

```
> a01<-0
> for(i in 1:5){
+   a01<-a01+i
+ }
> a01
[1] 15
```



せっかくプロット図を描いたが、色分けが９つもあって、回線も接近していたりするので、なんのことかあんまりわからないが・・・

もう一回、各色のグレーとの価格差（回帰線の切片の差）を見てみよう。

```
> o01_colc
```

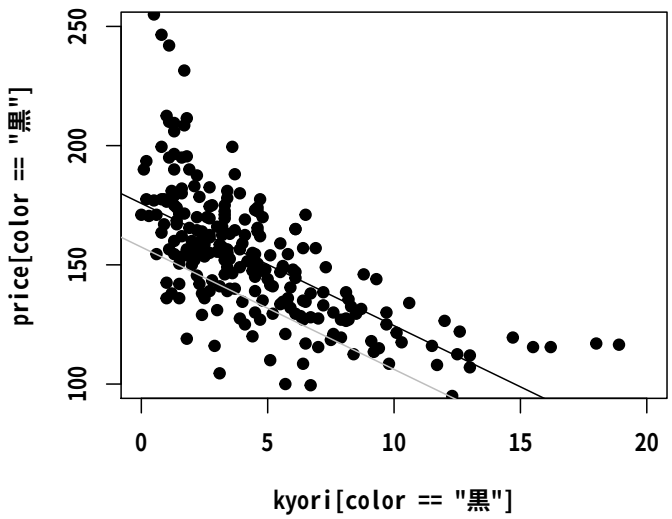
```
  (Intercept)         kyori color パープル   color ワイン        color 黒
157.4468850    -5.1294633    13.8385885    17.1737701    18.3255317
    color 紺      color 青       color 赤      color 白      color 緑
  0.3521494    -0.9032524     6.4031487    18.7038701    -0.8025127
```

白、黒、ワイン、パープル、... の順で、値段が高くなっている。

本当だろうか？

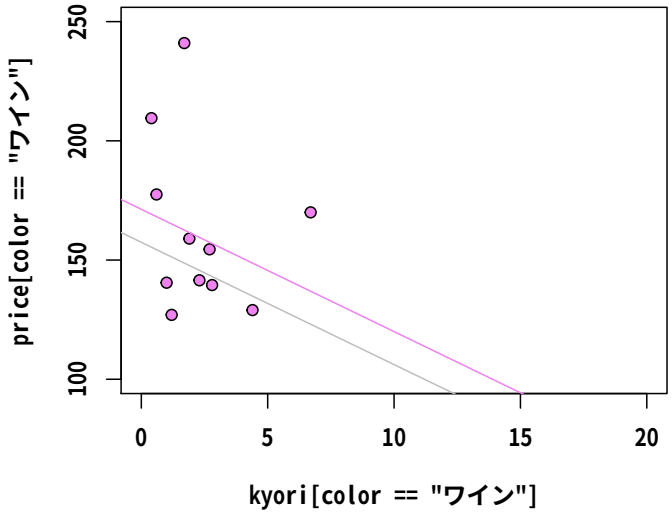
黒はデータがたくさんある。

```
> plot(price[color=="黒"]~kyori[color=="黒"],
+      pch=21,bg="black",xlim=c(0,20),ylim=c(100,250))
> abline(o01_colc[1]+o01_colc[5],o01_colc[2],col="black")
> abline(o01_colc[1],o01_colc[2],col="gray")
```



ワインはデータがちょっとしかない。

```
> plot(price[color=="ワイン"]~kyori[color=="ワイン"],
+      pch=21,bg="violet",xlim=c(0,20),ylim=c(100,250))
> abline(o01_colc[1]+o01_colc[3],o01_colc[2],col="violet")
> abline(o01_colc[1],o01_colc[2],col="gray")
```



黒は、データが１個２個入れ替わっても、おそらく回帰直線の位置は変わらないだろう。しかし、ワインは一番高いのが無くなっただけで、だいぶ回帰直線の位置は変わるはずだ。

推定係数の信頼度をどう評価するか？

推定係数のばらつきも表示する： lm 関数の出力に対する summary' 関数

lm 関数を使って行った OLS 回帰の結果 `o01_col` を summary 関数で表示すると、OLS 回帰分析結果として欲しい情報がひととおり表示される。

```
> summary(o01_col)
```

```
Call:
lm(formula = price ~ kyori + color)

Residuals:
    Min      1Q  Median      3Q     Max
-55.379 -12.077  -1.309  10.214  85.492

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   157.4469     1.5019 104.833  < 2e-16 ***
kyori          -5.1295     0.1711 -29.977  < 2e-16 ***
color パープル  13.8386     5.1131   2.707  0.00692 **
color ワイン    17.1738     5.9268   2.898  0.00384 **
color 黒        18.3255     1.7446  10.504  < 2e-16 ***
color 紺         0.3521     5.4552   0.065  0.94854
color 青        -0.9033     2.5746  -0.351  0.72579
color 赤         6.4031     4.1038   1.560  0.11901
color 白        18.7039     1.5555  12.024  < 2e-16 ***
color 緑        -0.8025     8.6723  -0.093  0.92629
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 19.18 on 990 degrees of freedom
Multiple R-squared:  0.5474,  Adjusted R-squared:  0.5432
F-statistic:   133 on 9 and 990 DF,  p-value: < 2.2e-16
```

この Coefficients: のところを見てもらいたい。 Estimate の列に `coef(o01_col)` で表示した係数の推定値が表示されている。

次の Std. Error の箇所は、その推定値のバラツキを示す。 Estimate の列に表示されたのは、あくまでそのバラツキの中のひとつの代表値であって、サンプルの取り方によってその値は変わる可能性がある。

区間推定

推定値がどのくらいばらつくかというと、そのバラツキの幅が Std. Error で判断することができる。それは、 Estimate のバラツキの 1 標準偏差であって、標準誤差 と呼ばれる。

Estimate の値プラス・マイナス Std. Error の範囲に、バラツキの約 2/3 が収まる。 Estimate の値のプラス・マイナス 2 × Std. Error の範囲に、バラツキの約 90 %が収まる。

そう理解して、各色の係数を見ると、例えば、黒だと、 Estiate が 18.326、つまり黒はグレーより 18 万 3 千円高くなると推定されるが、それは確実なものではなくて、その Std. Error の 2 倍（1.745 × 2）のプラス・マイナスで、本当の推定値を考えるべきだ、ということになる。つまり、14.8 万円〜21.8 万円ぐらい。

一方、ワイン色だと、 Estiate が 17.174、つまり黒はグレーより 17 万 2 千円ほど高くなると推定されているが、 Std. Error が大きく、その 2 倍は 5.9265 × 2 である。その幅のプラス・マイナスに、本当の推定値があると考えるべきだから、本当のワイン色のプレミアは、5.3 万円〜29 万円の間にあると予想され、だいぶ幅がある。

ちなみに、これまでやってきた１つだけの推定値を求めることを 点推定 と呼び、90 %、95 %、99 %といった一定の確率以上で本当の推定値がある範囲を予想することを 区間推定 と呼ぶ。そして、その範囲を、90 %の 信頼区間 と呼ぶ。

t 値と p 値

t value というのは、日本語では **t 値** と呼ばれる（そのまんまだが）。

係数の推定値 Estimate を標準誤差 Std. Error で割った値（ Estimate ／ Std. Error ）のこと。t 値の絶対値が大きければ、 Estimate の値が 0 でない可能性が高い。

逆に、t 値の絶対値が小さければ、（ Estimate としてどんな大きな値が出ていたとしても）、本当の係数は

0、すなわち、被説明変数を何も説明していない、関係ない、可能性があるということになる。

その可能性はどのくらいかというのが、 Pr(>|t|) の列に出ていて、この値を **p 値** と呼ぶ。

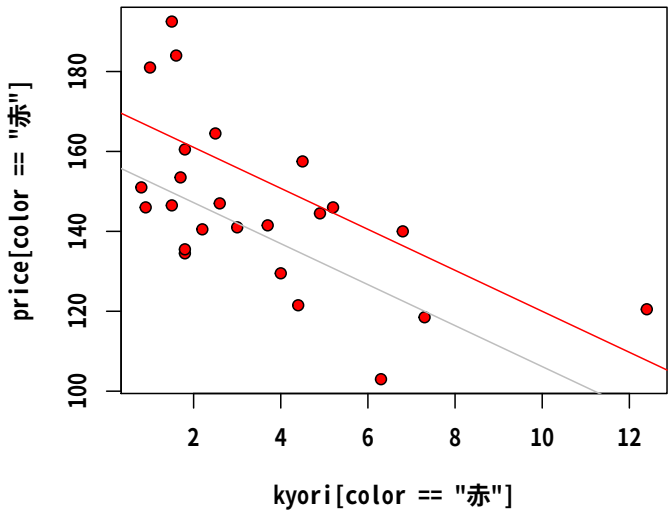
例えば、黒色だと、本当の係数が 0 である可能性は 2e-16 （2x10 のマイナス 16 乗のこと）未満、つまりほぼ 0 ということ。ワイン色だと 0.0069 で、0.69 %。

ワイン色は、黒色と比べて、サンプル数も少ないわりにバラツキも大きかったので、信頼区間が広くなってしまった。ワイン色のシルバーに対するプレミアが 5 万円〜29 万円の間だと言われても、だいぶ開きがあるな、という感じだ。

しかし、それでも、少なくとも、シルバーに対してプレミアがあるのは確かで、プレミアがない（ワイン色の係数が 0 である）確率は 0.69 %でしかない。可能性は 0 ではないが、ほとんどない。

一方、赤色は、Estimate が 6.403 万円で、点推定としてはある程度のプレミアがついているが、Std. Error が 4.104 もあり、90 %信頼区間は約 14 万円〜-2 万円である。このとき t 値は 1.56 で、シルバーに対する赤のプレミアが付かない（赤色の係数が 0 である）確率は 0.1190、つまり 11.9 %と結構ある。だから、赤色にプレミアムがあるとの確証は得られない。

```
> plot(price[color=="赤"]~kyori[color=="赤"],pch=21,bg="red")
> abline(o01_colc[1]+o01_colc[3],o01_colc[2],col="red")
> abline(o01_colc[1],o01_colc[2],col="gray")
```



つまり、t 値（とその大きさを評価した p 値）を見て、説明変数の係数が 0 である可能性が低いものについては、推定値とその標準誤差で、その係数がどの程度であるかを確認し、係数が 0 である可能性が高いもの変数については、その推定値は評価しない。場合によっては、説明変数から抜く場合もある。

ところで、係数が 0 である可能性が低い、高いは、どこで判断するかというと、統計学では、慣例的に、p 値を、10 %、5 %、1 %、0.1 %で区切り、それよりも小さいか大きいかで判断することが多い。

p 値が 0.1 %未満というのはかなり厳しい基準で、 color で推定値が 0 より大きい（プレミアがある）と判断される色は、黒と白だけだ。

p 値が 1 %未満ならば、パープルとワインのプレミアも「ある」と言えることになる。

p 値が 10 %未満というのは、かなり甘い基準で、自然科学の実験データなどではほとんど使われないようだが、社会科学のように、曖昧なな関係を、限られたデータで確認するときには使われることもある。

ちなみに R では、p 値が 0.1 %未満のときは、p 値の横に *** と表示し、1 %未満なら ** 、5 %未満なら * 、10 %未満なら . と表示する。

つまり、p 値の横に何も表示されていない変数については、その係数があんまり意味ないと考えてよい。

ただし、注意点！ として、ある変数について p 値が 0.1 以上だったとしても、その変数の「係数が 0」であるということを実証したわけではないことを留意してほしい。

赤色はの p 値は 0.1 より大きいが、これはサンプル数が少ないことが原因している。もっとサンプル数が多ければ、赤色にプレミアがついているか、ついていないかがはっきりするはずだ。「これだけでははっきりしない」というだけ。

2.7.2　実験してみよう

関係式

変数 x と y との間に、

$$y = 10 - x$$

という関係があるとする。しかし、いろいろな攪乱があり、実際に得られるデータは、

$$y = 10 - x + \varepsilon$$

である。

この ε がほとんど 0 だったら、x と y の散布図は、一直線に並ぶはずである。

ε がランダムに大きくばらつくなら、x と y の散布図も、この関係式のまわりに大きくばらつく。

以下では、ε という攪乱項が入った x と y のデータセットについて OLS 回帰を行い、もともとの関係式 $y = 10 - x$ がどのくらいのうまく再現できるかを見てみる。

攪乱項 ε を人為的に発生させる： rnorm 関数

攪乱項を、自分で適当に与えてもよいのだが面倒だ。

rnorm 関数を使うと、正規分布 というお行儀のよい分布から、指定した個数だけ、ランダムに抜き出して返してくれる。こういう値を 乱数 と言う。

以下で 10 個の値が得られる。

```
> rnorm(10)
```

```
 [1] -0.75978740  0.03105043  0.18718992 -1.24655178 -0.59843437 -1.65820660
 [7]  1.79674372 -0.69809962  0.58753036 -1.32240197
```

100 個だって簡単に得られる。

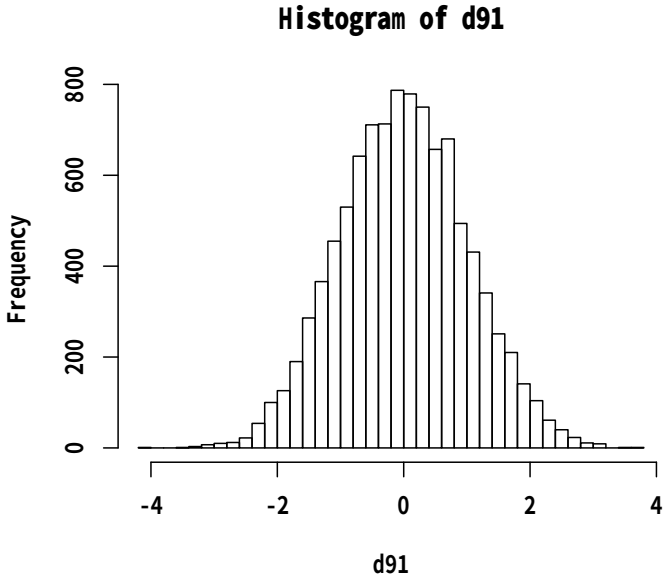
```
> rnorm(100)
```

```
  [1]  0.23907773 -0.58486295 -1.27604702 -0.07209034 -0.34996917  0.33065295
  [7]  0.03413513 -2.34936227 -0.12458878 -1.01845762  0.08023313  1.52810436
 [13]  1.26962299 -0.35745194 -0.87742658 -1.33484892  1.12579984 -0.73407001
 [19] -0.31455030  0.24376201 -0.43814832 -1.11969847  0.72602168 -0.08101855
 [25] -0.58768423  2.54546542 -0.47108887 -1.88436725 -0.91410454  0.83614849
 [31]  0.23048797  0.67828603  0.59872942 -0.60406026  1.03988485  0.57179323
 [37] -0.05776843 -0.24616156 -0.83421953 -0.67390783 -1.58948569 -2.35858946
 [43]  0.55450744  1.03902390 -0.11134721  0.46541868  0.23937367 -1.88982227
 [49]  1.86458060 -0.57758284 -0.69386073  0.60968971  0.12105819 -1.14526038
 [55]  0.77067893  0.62896271  0.48205237 -0.29385753 -0.57260575  0.33102246
 [61]  2.69112312  0.82831372  1.70228802  0.04644748 -1.31055188  0.18307335
 [67]  0.55025694 -0.23332060 -1.14001763 -0.20170514  0.16352075 -0.03728902
 [73]  1.18192382 -0.50678658  0.34094924 -0.84925683 -1.06364356 -1.55659119
 [79]  1.20437121 -0.55380201  1.03569919 -0.48240590 -0.52411982  0.23225313
 [85]  1.73763427 -1.16130944 -0.77089082 -0.46312681  0.51095903  0.54058288
 [91] -1.26222795 -1.30212483  0.97670554  1.16892365  0.12971954  0.15297106
 [97] -0.44365146  0.57558291 -1.49492068 -2.20327878
```

1000 個もとると、これらの値がどんなバラツキとなるかを確認することができる。

10000 個の値を d91 に代入して、 hist 関数でヒストグラムを描いた。(引数 br="scott" は、ヒストグラムの階級を指定する 1 種で、これがないと、分布がだいぶおおざっぱになるので加えた。)

```
> d91<-rnorm(10000)
> hist(d91,br="scott")
```



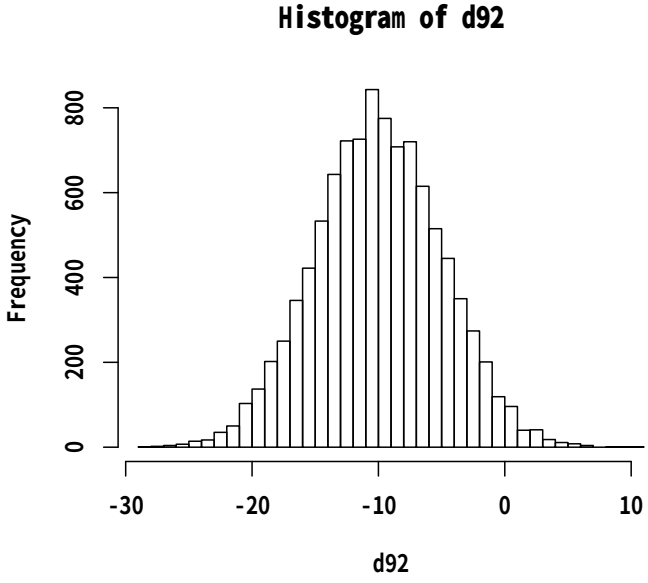
横軸が rnorm 関数で発生させた値で、縦軸が各階級（0〜0.1、0.1〜0.2 とかの区切り）に何個あるかを示す。

これが正規分布の形。

rnomr(10) とした場合、この中からランダムに 10 個持ってきたと考えても差し支えない。

正規分布からランダムに取り出した値（乱数）は 正規乱数 とも呼ばれる。平均や標準偏差（ばらつき）を変えたかったら引数 mean= 、 sd= で与えることができる。

```
> d92<-rnorm(10000,mean=-10,sd=5)
> hist(d92,br="scott")
```



この形はおおよそ平均-10、標準偏差 5 の正規分布を表している。分布の形は全然変わっていないように見えるが、横軸の位置とスケールが全然違うことに注意。

10 個のデータセットで OLS 回帰してみる

x が 1,2,...,10 の 10 個のデータだとして、これを 91 に代入。

```
> (x91<-seq(1,10,1))
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

$y = 10 - x$ に平均平均 0、標準偏差 1 の正規乱数を 10 個発生させて足す。[*7]

```
> set.seed(123)
> (y91<-10-x91+rnorm(10))
```

```
 [1]  8.4395244  7.7698225  8.5587083  6.0705084  5.1292877  5.7150650
 [7]  3.4609162  0.7349388  0.3131471 -0.4456620
```

lm 関数 x90 を y91 に OLS 回帰させて、結果を summary 関数で示した。

```
> o91<-lm(y91~x91)
> summary(o91)
```

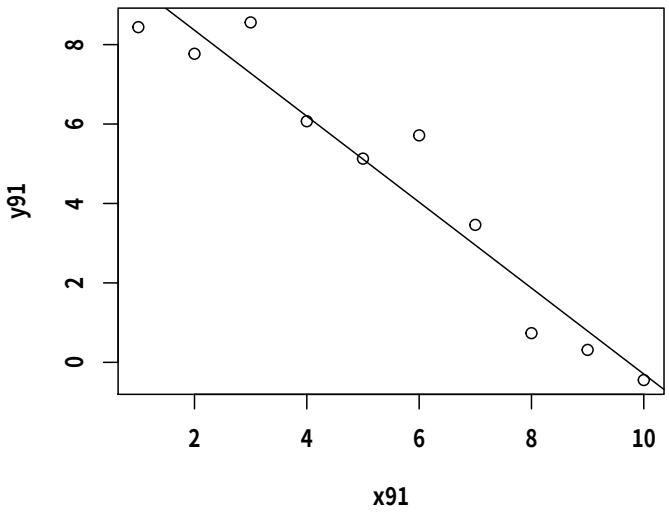
```
Call:
lm(formula = y91 ~ x91)

Residuals:
    Min      1Q  Median      3Q     Max
-1.1348 -0.5624 -0.1393  0.3854  1.6814

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.5255     0.6673   15.77 2.61e-07 ***
x91          -1.0820     0.1075  -10.06 8.11e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9768 on 8 degrees of freedom
Multiple R-squared:  0.9268,        Adjusted R-squared:  0.9176
F-statistic: 101.2 on 1 and 8 DF,  p-value: 8.111e-06
```

ほぼ、$y = 10 - x$ に近い値が推定され、 x90 の係数の標準誤差 Std. Error が 0.1 程度で、信頼区間は-1.2〜-0.8 ぐらいに留まっている。t 値の絶対値も大きな値で p 値が小さい。

散布図も描く

```
> plot(y91~x91) #散布図
> abline(o91)   #回帰直線を描き入れる
```

[*7] set.seed(123) というのは、特定の乱数を発生させるためのおまじない。本当は必要ないのだが、乱数は毎回違うものが出てくるので、このレジュメの後の細かい数値の解説が狂ってしまうから、固定させてもらった。



10 個のプロットが回帰直線のまわりに集まっている。

10 個のデータセットのバラツキを大きくしてみる。

x はそのまま x91 で、誤差項の標準偏差を 10 に増やしてみる。

```
> set.seed(123)
> (y92<-10-x91+rnorm(10,sd=10))
```

```
 [1]   3.395244   5.698225  22.587083   6.705084   6.292877  21.150650
 [7]   7.609162 -10.650612  -5.868529  -4.456620
```

lm 関数 x91 を y92 に OLS 回帰させて、結果を summary 関数で示した。

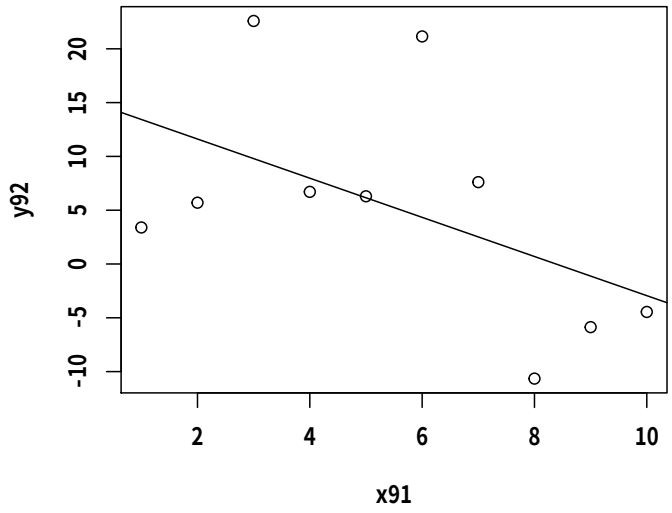
```
> o92<-lm(y92~x91)
> summary(o92)
```

```
Call:
lm(formula = y92 ~ x91)

Residuals:
    Min      1Q  Median      3Q     Max
-11.348  -5.624  -1.393   3.853  16.814

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   15.255      6.673   2.286   0.0516 .
x91           -1.820      1.075  -1.692   0.1291
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.768 on 8 degrees of freedom
Multiple R-squared:  0.2636,        Adjusted R-squared:  0.1715
F-statistic: 2.863 on 1 and 8 DF,  p-value: 0.1291
```

ほぼ、$y = 10 - x$ とはだいぶ離れた値となり、 x91 の係数の標準誤差 Std. Error も大きい。信頼区間の中に本来推定したい-1 という値が含まれてはいるが、信頼区間が広すぎる。そのため、t 値の絶対値が小さく、p 値も大きく、 x91’ と y92 に相関があるとは言えない。

散布図も描く

```
> plot(y92~x91) #散布図
> abline(o92) #回帰直線を描き入れる
```



10 個のプロットと回帰直線の隔たりがだいぶある。

データセットの規模を 1000 に増やす

標準偏差を 10 と大きいままにしておいて、データセットの規模を 1000 にしてみる。

x が 0.01,0.02,...,10 の 1000 個のデータだとして、これを x93 に代入。

```
> x93<-seq(0.01,10,0.01)
```

x93 のデータに対して $y = 10 - x$ を計算し、それに標準偏差 10 の誤差項を加える。

```
> set.seed(123)
> y93<-10-x93+rnorm(1000,sd=10)
```

lm 関数 x93 を y93 に OLS 回帰させて、結果を summary 関数で示した。

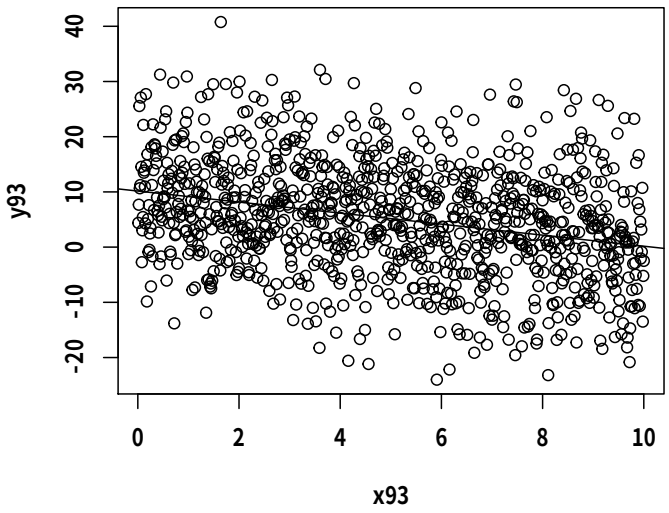
```
> o93<-lm(y93~x93)
> summary(o93)
```

```
Call:
lm(formula = y93 ~ x93)

Residuals:
    Min      1Q  Median      3Q     Max
-28.260  -6.442  -0.068   6.482  32.253

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.1562     0.6280  16.173   <2e-16 ***
x93          -0.9990     0.1087  -9.191   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.922 on 998 degrees of freedom
Multiple R-squared:  0.07804,        Adjusted R-squared:  0.07712
```

```
F-statistic: 84.48 on 1 and 998 DF,  p-value: < 2.2e-16
```

散布図も描く

```
> plot(y93~x93) #散布図
> abline(o93) #回帰直線を描き入れる
```



分布だけ見ているとほとんど関係はないように見えるが、しっかりと関係式が推定されているのがわかる。

今回は項目変数なので、特定の項目だけ説明変数から落とすという訳にはいかないが、「その他」という項目をつくり、その中に、推定値が不確かなものをまとめるというのは一つの方法かもしれない。

数値変数で p 値の小さいものがあったら、それは変数自体を説明変数からはずすという方法もあるが、実際には、他の変数との関係から、p 値だけで判断するのはまずい場合もある。

変数の取捨選択については、次回へ続く。

2.8 AIC による説明変数の取捨選択と決定係数

2.8.1 説明変数の取捨選択

前回の作業結果

ここからはじめる人のために、今回の作業に必要な作業をここでやっておく。

```
> d01<-read.csv("prius.csv")
```

前回までは、１つか２つの変数だけを説明変数として加えてきた。これはあくまで説明の都合。通常は、使えそうな変数をすべて加えて、そこから説明力のない変数を抜く、という作業を行う。

データを確認する： summary 関数

ひとまず、d01 のデータがどんなんかを再確認

```
> summary(d01)
```

```
   nenshiki         kyori                  kizu         price       grade
Min.   :21.00   Min.   : 0.000   きれい       :306   Min.   : 76.5   G:151
1st Qu.:22.00   1st Qu.: 2.000   すごくきれい:414   1st Qu.:125.5   L:105
Median :22.00   Median : 3.600   修復歴あり   :158   Median :144.5   S:744
Mean   :22.11   Mean   : 4.643   普通         :122   Mean   :145.3
3rd Qu.:23.00   3rd Qu.: 6.400                       3rd Qu.:162.0
Max.   :24.00   Max.   :22.500                       Max.   :256.0

selection     shaken            color           NV              SR
LED: 65   Min.   : 0.000   白       :372   Min.   :0.00   Min.   :0.000
non:653   1st Qu.: 0.000   シルバー:261   1st Qu.:0.00   1st Qu.:0.000
TS :274   Median : 5.000   黒       :228   Median :1.00   Median :0.000
TSG:  8   Mean   : 7.014   青       : 71   Mean   :0.65   Mean   :0.032
          3rd Qu.:13.000   赤       : 24   3rd Qu.:1.00   3rd Qu.:0.000
          Max.   :31.000   パープル: 15   Max.   :1.00   Max.   :1.000
                           (Other) : 29
     kawa
Min.   :0.000
1st Qu.:0.000
Median :0.000
Mean   :0.044
3rd Qu.:0.000
Max.   :1.000
```

次に、項目変数への変換＆基準となる項目の設定をする

nenshiki （年式）を項目変数に： factor 関数

```
> d01$nenshiki<-factor(d01$nenshiki)
> summary(d01$nenshiki)
```

```
 21  22  23  24
226 521 172  81
```

基準となる年式は 21 でよいから、これで問題ない。

kizu （傷）の基準を「普通」に： factor 関数の引数 levels=

基準にしたい項目を、 factor 関数の引数 levels= で最初にもってくればよい。あとの順番は適当に・・・

```
> d01$kizu<-factor(d01$kizu,
+                  levels=c("普通","きれい","すごくきれい","修復歴あり"))
```

```
> summary(d01$kizu)
```

```
      普通       きれい  すごくきれい    修復歴あり
       122          306          414          158
```

grade （グレード）の基準を「S」に： factoro 関数の引数 levels=

```
> d01$grade<-factor(d01$grade,levels=c("S","G","L"))
> summary(d01$grade)
```

```
  S   G   L
744 151 105
```

selection （特別仕様）の基準を「non（なんにも無し）」に： factoro 関数の引数 levels=

```
> selection<-factor(d01$selection,
+                   levels=c("non","TS","TSG","LED"))
> summary(d01$selection)
```

```
LED non  TS TSG
 65 653 274   8
```

color （色）の基準を「シルバー」に： factoro 関数の引数 levels=

```
> color<-factor(d01$color,
+               levels=c("シルバー","白","黒","青","紺",
+                        "赤","パープル","ワイン","緑"))
> summary(d01$color)
```

```
シルバー パープル   ワイン       黒       紺       青       赤       白
     261       15       11      228       13       71       24      372
      緑
       5
```

NV （カーナビ）、 SR （サンルーフ）、 kawa ` （革）については、無しが 0、付きが 1 で入力されており、項目変数に変換してもよいが、OLS 回帰をやる分には、何の意味もないので、そのまま数値でよい。

すべての変数を説明変数に OLS 回帰を行う： lm 関数の「 . 」

‘ c lm 関数の引数 formula= で、 lm(y~.,d01) と指定すると、データフレーム d01 に入った y 以外のすべての変数で回帰してくれる。

```
> lm(price~.,d01)
```

```
Call:
lm(formula = price ~ ., data = d01)

Coefficients:
     (Intercept)        nenshiki22        nenshiki23        nenshiki24
        141.9501            5.9515           18.9692           39.4475
           kyori        kizu きれい  kizu すごくきれい    kizu 修復歴あり
          -3.2209            4.1928           12.4744          -10.8663
          gradeG            gradeL      selectionnon       selectionTS
           1.6659           -7.8442           -9.4022            0.7282
    selectionTSG             shaken     color パープル       color ワイン
          20.9866            0.1819           -1.7540            5.0850
        color 黒           color 紺           color 青           color 赤
```

```
    10.5209          -2.3053          -0.3884           1.8629
color 白         color 緑               NV               SR
     9.7176           3.4146           2.8602          12.8986
       kawa
    16.3337
```

2 次の項を加える： lm 関数の「 + 」

さらに 2 次の項を加えたければ、普通に足せばよい。

kyori の 2 次の項（cx^2）を加えた OLS 回帰の結果を o01_all に代入する。

```
> o01_all<-lm(price~.+I(kyori^2),d01)
```

OLS の結果の要約を表示する： summary 関数

lm 関数の結果に summary 関数を使うと、OLS 回帰分析の結果として欲しい情報がひととおり表示される。

```
> summary(o01_all)
```

```
Call:
lm(formula = price ~ . + I(kyori^2), data = d01)

Residuals:
    Min      1Q  Median      3Q     Max
-62.281  -5.818  -0.138   5.545  44.982

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      145.79574    2.17147  67.141  < 2e-16 ***
nenshiki22         5.88592    0.84610   6.957 6.39e-12 ***
nenshiki23        18.02483    1.12565  16.013  < 2e-16 ***
nenshiki24        36.78065    1.55571  23.642  < 2e-16 ***
kyori             -5.32875    0.28606 -18.628  < 2e-16 ***
kizu きれい        5.88737    1.11103   5.299 1.44e-07 ***
kizu すごくきれい 13.45969    1.11503  12.071  < 2e-16 ***
kizu 修復歴あり   -9.82569    1.24698  -7.880 8.75e-15 ***
gradeG             2.02205    1.07185   1.886 0.059525 .
gradeL            -7.59860    1.10838  -6.856 1.26e-11 ***
selectionnon      -9.18652    1.43219  -6.414 2.20e-10 ***
selectionTS        1.28906    1.51340   0.852 0.394553
selectionTSG      21.25518    3.95467   5.375 9.60e-08 ***
shaken             0.17485    0.04664   3.749 0.000188 ***
color パープル    -1.93503    2.74084  -0.706 0.480358
color ワイン       4.45768    3.15014   1.415 0.157367
color 黒          10.71118    0.96679  11.079  < 2e-16 ***
color 紺          -1.88008    2.88260  -0.652 0.514415
color 青          -0.70980    1.36553  -0.520 0.603323
color 赤           1.49643    2.17797   0.687 0.492198
color 白           9.75786    0.84932  11.489  < 2e-16 ***
color 緑           3.49534    4.60823   0.758 0.448336
NV                 2.99499    0.68894   4.347 1.52e-05 ***
SR                12.69537    1.86782   6.797 1.86e-11 ***
kawa              15.93545    1.86320   8.553  < 2e-16 ***
I(kyori^2)         0.14075    0.01779   7.912 6.87e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.04 on 974 degrees of freedom
Multiple R-squared:  0.878,        Adjusted R-squared:  0.8749
F-statistic: 280.4 on 25 and 974 DF,  p-value: < 2.2e-16
```

selection と color の中に、p 値が 0.1 以上となる不安定な推定値があるが、その他は有意な値が得られた。

説明変数の組を最適化する： step 関数

被説明変数と関係が確認されないような変数を説明変数に入れていると、推定全体の効率を落とすので、(今回は、そういう変数はなかったが)、そうした変数は、説明変数から外したほうがよい場合もある。

どの変数を説明変数に加え、どの変数を説明変数から外した方がよいかのかを判断する方法として、ステップワイズ法とう方法がある。

ステップワイズ法は、たくさんある説明変数の候補の中から、いろいろな組み合わせを試してみて、その中で最も AIC の小さい組み合わせを選択する。

作業としては、たいへんな作業だが、ありがたいことに、R には step 関数というのがあり、これを勝手にやってくれる。

使い方は、とにかく候補になりそうな変数をすべて説明変数に加えて OLS 回帰を行い、その結果を step 関数の引数に与えるだけで、不要な変数を抜いてくれる。

```
> o01_stp01<-step(o01_all)
```

```
Start:  AIC=4638.75
price ~ nenshiki + kyori + kizu + grade + selection + shaken +
    color + NV + SR + kawa + I(kyori^2)

             Df Sum of Sq    RSS    AIC
<none>                    98175 4638.8
- shaken      1      1417  99592 4651.1
- NV          1      1905 100080 4656.0
- SR          1      4657 102832 4683.1
- grade       2      5473 103648 4689.0
- I(kyori^2)  1      6309 104485 4699.0
- kawa        1      7373 105548 4709.2
- color       8     22844 121019 4831.9
- selection   3     24279 122454 4853.7
- kyori       1     34976 133151 4941.5
- kizu        3     61622 159798 5119.9
- nenshiki    3     64650 162825 5138.7
```

結果を summary 関数で確認する

```
> summary(o01_stp01)
```

```
Call:
lm(formula = price ~ nenshiki + kyori + kizu + grade + selection +
    shaken + color + NV + SR + kawa + I(kyori^2), data = d01)

Residuals:
    Min      1Q  Median      3Q     Max
-62.281  -5.818  -0.138   5.545  44.982

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      145.79574    2.17147  67.141  < 2e-16 ***
nenshiki22         5.88592    0.84610   6.957 6.39e-12 ***
nenshiki23        18.02483    1.12565  16.013  < 2e-16 ***
nenshiki24        36.78065    1.55571  23.642  < 2e-16 ***
kyori             -5.32875    0.28606 -18.628  < 2e-16 ***
kizu きれい         5.88737    1.11103   5.299 1.44e-07 ***
kizu すごくきれい  13.45969    1.11503  12.071  < 2e-16 ***
kizu 修復歴あり    -9.82569    1.24698  -7.880 8.75e-15 ***
gradeG             2.02205    1.07185   1.886 0.059525 .
gradeL            -7.59860    1.10838  -6.856 1.26e-11 ***
selectionnon      -9.18652    1.43219  -6.414 2.20e-10 ***
selectionTS        1.28906    1.51340   0.852 0.394553
selectionTSG      21.25518    3.95467   5.375 9.60e-08 ***
shaken             0.17485    0.04664   3.749 0.000188 ***
color パープル    -1.93503    2.74084  -0.706 0.480358
```

```
color ワイン         4.45768    3.15014   1.415 0.157367
color 黒            10.71118    0.96679  11.079  < 2e-16 ***
color 紺            -1.88008    2.88260  -0.652 0.514415
color 青            -0.70980    1.36553  -0.520 0.603323
color 赤             1.49643    2.17797   0.687 0.492198
color 白             9.75786    0.84932  11.489  < 2e-16 ***
color 緑             3.49534    4.60823   0.758 0.448336
NV                   2.99499    0.68894   4.347 1.52e-05 ***
SR                  12.69537    1.86782   6.797 1.86e-11 ***
kawa                15.93545    1.86320   8.553  < 2e-16 ***
I(kyori^2)           0.14075    0.01779   7.912 6.87e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.04 on 974 degrees of freedom
Multiple R-squared:  0.878,        Adjusted R-squared:  0.8749
F-statistic: 280.4 on 25 and 974 DF,  p-value: < 2.2e-16
```

予想どおり、不要な変数はないようだ。

せっかくなので、まったく関係ない変数を加えて、回帰してみよう。

rnorm 関数を使って、平均 0、標準偏差 1 の正規乱数を 1000 個発生させて x01 という名前で保存し、これを説明変数に加えてみる。

さらに、 seq 関数を使って、0〜1 の間を等間隔に区切った 1000 個の数値を発生させて、これも x02 という名前で説明変数に加えてみる。

```
> x01<-rnorm(1000)
> x02<-seq(0,1,length=1000)
> o01_all_x<-lm(price~.+I(kyori^2)+x01+x02,d01)
```

結果を summary 関数で確認する

```
> summary(o01_all_x)
```

```
Call:
lm(formula = price ~ . + I(kyori^2) + x01 + x02, data = d01)

Residuals:
    Min      1Q  Median      3Q     Max
-61.602  -5.948  -0.224   5.425  45.713

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     145.27521    2.20973  65.744  < 2e-16 ***
nenshiki22        5.88902    0.84623   6.959 6.29e-12 ***
nenshiki23       18.01044    1.12604  15.994  < 2e-16 ***
nenshiki24       36.80140    1.55603  23.651  < 2e-16 ***
kyori            -5.31867    0.28633 -18.575  < 2e-16 ***
kizu きれい       5.85002    1.11290   5.257 1.80e-07 ***
kizu すごくきれい 13.37468    1.11710  11.973  < 2e-16 ***
kizu 修復歴あり   -9.91364    1.24898  -7.937 5.67e-15 ***
gradeG            2.02023    1.07543   1.879 0.060608 .
gradeL           -7.63450    1.10906  -6.884 1.04e-11 ***
selectionnon     -9.29058    1.43463  -6.476 1.49e-10 ***
selectionTS       1.14308    1.51777   0.753 0.451556
selectionTSG     21.16661    3.95632   5.350 1.10e-07 ***
shaken            0.17520    0.04664   3.756 0.000183 ***
color パープル   -2.03986    2.74315  -0.744 0.457286
color ワイン      4.48337    3.15078   1.423 0.155075
color 黒         10.71401    0.96695  11.080  < 2e-16 ***
color 紺         -2.02403    2.88594  -0.701 0.483258
color 青         -0.75498    1.36623  -0.553 0.580664
color 赤          1.25999    2.18609   0.576 0.564500
color 白          9.72278    0.85011  11.437  < 2e-16 ***
color 緑          3.50023    4.60898   0.759 0.447775
NV                2.95838    0.68976   4.289 1.97e-05 ***
```

```
SR                12.62145    1.86899   6.753 2.49e-11 ***
kawa              15.82034    1.86801   8.469  < 2e-16 ***
I(kyori^2)         0.14008    0.01780   7.870 9.46e-15 ***
x01                0.02311    0.33369   0.069 0.944791
x02                1.44710    1.11963   1.292 0.196498
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.04 on 972 degrees of freedom
Multiple R-squared:  0.8782,        Adjusted R-squared:  0.8748
F-statistic: 259.6 on 27 and 972 DF,  p-value: < 2.2e-16
```

変数 x01 、 x02 の推定値の p 値は大きく、有意な説明変数とはいえないことがわかるはずだ。

この結果に step 関数をあてはめてみる。

```
> o01_stp02<-step(o01_all_x)
```

```
Start:  AIC=4641
price ~ nenshiki + kyori + kizu + grade + selection + shaken +
    color + NV + SR + kawa + I(kyori^2) + x01 + x02

             Df Sum of Sq    RSS    AIC
- x01         1         0  98004 4639.0
- x02         1       168  98172 4640.7
<none>                    98003 4641.0
- shaken      1      1422  99426 4653.4
- NV          1      1855  99858 4657.8
- SR          1      4598 102601 4684.9
- grade       2      5515 103518 4691.7
- I(kyori^2)  1      6244 104248 4700.8
- kawa        1      7232 105235 4710.2
- color       8     22895 120898 4834.9
- selection   3     24214 122218 4855.8
- kyori       1     34790 132793 4942.8
- kizu        3     61554 159558 5122.4
- nenshiki    3     64666 162670 5141.7

Step:  AIC=4639.01
price ~ nenshiki + kyori + kizu + grade + selection + shaken +
    color + NV + SR + kawa + I(kyori^2) + x02

             Df Sum of Sq    RSS    AIC
- x02         1       171  98175 4638.8
<none>                    98004 4639.0
- shaken      1      1422  99426 4651.4
- NV          1      1854  99858 4655.8
- SR          1      4599 102603 4682.9
- grade       2      5517 103520 4689.8
- I(kyori^2)  1      6245 104249 4698.8
- kawa        1      7243 105247 4708.3
- color       8     22907 120910 4833.1
- selection   3     24215 122219 4853.8
- kyori       1     34829 132833 4941.1
- kizu        3     61561 159565 5120.5
- nenshiki    3     64677 162680 5139.8

Step:  AIC=4638.75
price ~ nenshiki + kyori + kizu + grade + selection + shaken +
    color + NV + SR + kawa + I(kyori^2)

             Df Sum of Sq    RSS    AIC
<none>                    98175 4638.8
- shaken      1      1417  99592 4651.1
- NV          1      1905 100080 4656.0
- SR          1      4657 102832 4683.1
- grade       2      5473 103648 4689.0
- I(kyori^2)  1      6309 104485 4699.0
- kawa        1      7373 105548 4709.2
- color       8     22844 121019 4831.9
- selection   3     24279 122454 4853.7
- kyori       1     34976 133151 4941.5
```

```
- kizu        3     61622 159798 5119.9
- nenshiki    3     64650 162825 5138.7
```

```
> summary(o01_stp02)
```

```
Call:
lm(formula = price ~ nenshiki + kyori + kizu + grade + selection +
    shaken + color + NV + SR + kawa + I(kyori^2), data = d01)

Residuals:
    Min      1Q  Median      3Q     Max
-62.281  -5.818  -0.138   5.545  44.982

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      145.79574    2.17147  67.141  < 2e-16 ***
nenshiki22         5.88592    0.84610   6.957 6.39e-12 ***
nenshiki23        18.02483    1.12565  16.013  < 2e-16 ***
nenshiki24        36.78065    1.55571  23.642  < 2e-16 ***
kyori             -5.32875    0.28606 -18.628  < 2e-16 ***
kizu きれい        5.88737    1.11103   5.299 1.44e-07 ***
kizu すごくきれい 13.45969    1.11503  12.071  < 2e-16 ***
kizu 修復歴あり   -9.82569    1.24698  -7.880 8.75e-15 ***
gradeG             2.02205    1.07185   1.886 0.059525 .
gradeL            -7.59860    1.10838  -6.856 1.26e-11 ***
selectionnon      -9.18652    1.43219  -6.414 2.20e-10 ***
selectionTS        1.28906    1.51340   0.852 0.394553
selectionTSG      21.25518    3.95467   5.375 9.60e-08 ***
shaken             0.17485    0.04664   3.749 0.000188 ***
color パープル    -1.93503    2.74084  -0.706 0.480358
color ワイン       4.45768    3.15014   1.415 0.157367
color 黒          10.71118    0.96679  11.079  < 2e-16 ***
color 紺          -1.88008    2.88260  -0.652 0.514415
color 青          -0.70980    1.36553  -0.520 0.603323
color 赤           1.49643    2.17797   0.687 0.492198
color 白           9.75786    0.84932  11.489  < 2e-16 ***
color 緑           3.49534    4.60823   0.758 0.448336
NV                 2.99499    0.68894   4.347 1.52e-05 ***
SR                12.69537    1.86782   6.797 1.86e-11 ***
kawa              15.93545    1.86320   8.553  < 2e-16 ***
I(kyori^2)         0.14075    0.01779   7.912 6.87e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.04 on 974 degrees of freedom
Multiple R-squared:  0.878,        Adjusted R-squared:  0.8749
F-statistic: 280.4 on 25 and 974 DF,  p-value: < 2.2e-16
```

変数 x01 、 x02 が外されていることがわかる。

2.8.2 残差のバラツキを読む

結局、 setp 関数をかけても O01_all のままでよいことがわかったが、それでどのくらいの予測の精度が得られたか、バラツキを確認してみる。

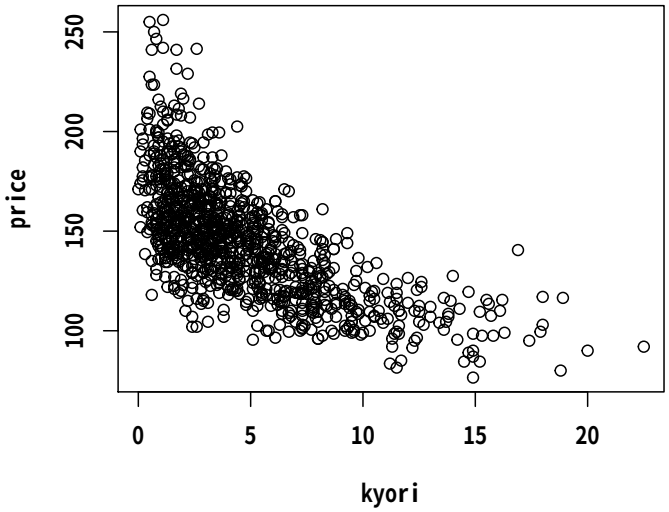
kyori の 2 次式だけの回帰の残差のバラツキ： resid 関数と plot 関数

何度もやったが、 plot 関数で、 kyori を横軸に price のバラツキを散布図にプロットする。

```
> attach(d01)
```

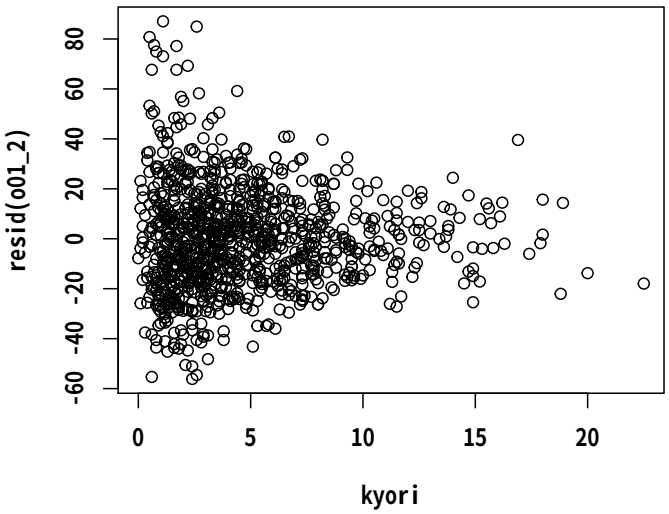
```
以下のオブジェクトはマスクされています (_by_ .GlobalEnv) :

    color, selection
```

```
> plot(price~kyori)
```



kyori の２次式だけの OLS 回帰の残差をプロットする。

```
> o01_2<-lm(price~kyori+I(kyori^2))
> plot(resid(o01_2)~kyori)
```



このバラツキが、 kyori の２次式だけでは説明できない price のバラツキということになる。

OLS 回帰の予測値: predict 関数

たとえば、走行距離 3 万 km の車だといくらぐらいになるだろうか。

回帰線上の y の値は、$\hat{y} = a + bx_1 + cx_1^2$ を計算すればよいのだが、OLS の推定結果から coef 関数を使って係数の推定値を取り出して、説明変数に掛ける・・・なんてことは、ちょっと面倒くさい。

これを predict 関数がやってくれる。

まず、予測に使うデータセットをつくる（といっても、 kyori だけが説明変数だけだが・・・）

データフレーム形式で、kyori と言う名前で 3 という数字を保存。(複数個データあったら、kyori=c(4,5,6) と言う具合に指定する。

```
> (d01_2<-data.frame(kyori=3))
```

```
  kyori
1     3
```

これを OLS 回帰の結果 o01_2 に当てはめる。

```
> predict(o01_2,d01_2)
```

```
       1
153.4623
```

153.5 万円という結果が出たが、それでは走行距離 3 万 km の車はだいたい 153 万円ぐらいと予想できるかというと、全然そうではない。

予測区間： predict 関数の引数 interval="prediction"

先ほどの散布図を見る限り、走行距離が 3 万 km で残差がプラスマイナス 40 万円前後の開きがあるようだ。

走行距離 3 万 km で実際にどのくらいの価格にばらつくのかは、 predict 関数に interval="prediction" という引数を与えることで求めることができる。

```
> predict(o01_2,d01_2,interval="prediction")
```

```
       fit      lwr     upr
1 153.4623 113.4106 193.514
```

fit が回帰線上の値で、 lwr が下限、 upr が上限の値を示す。

このように、実際にどのくらいの値にばらつくかを 予測区間 と言う。

「走行距離 3 万 km でいくらぐらいですか」？ と聞かれて「え〜と、113 万円〜193 万円ぐらい」と答えるしかない。ありがたいか？ あんまりありがたくない。[*8]

すべての変数で OLS 回帰した結果の残差のバラツキ： resid 関数と plot 関数

すべての変数を説明変数とした OLS 回帰の結果 o01_all の残差をプロットしてみる。kyori だけによる回帰のプロット比較するために、縦軸の範囲を合わせる。縦軸のスケールを 10〜20 に固定したかったら引数 ylim=c(10,20) という具合に指定する。

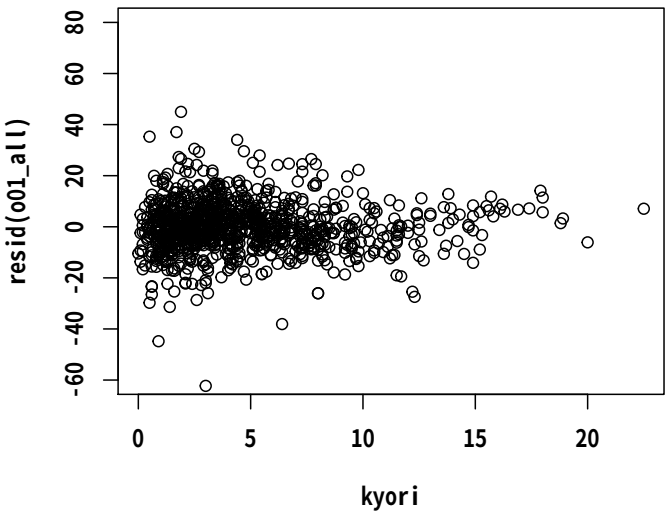
```
> plot(resid(o01_all)~kyori,ylim=c(-60,80))
```

[*8] ちなみに、関数 predict の引数 interval= には、 "confidence" という値も与えることができる。この場合は、$\hat{y}$ のバラツキであって、データの取り方でふらつく回帰線だが、その線上の点がだいたいどの範囲にあるのかを示している。これは 信頼区間 と言う。

```
> predict(o01_2,d01_2,interval="confidence")
```

```
       fit      lwr      upr
1 153.4623 152.0652 154.8593
```

当然、信頼区間の方が予測区間よりも狭くなる。



だいぶバラツキが小さくなっている。

この OLS 回帰に使用した説明変数を与えると、車体価格のかなりの部分が説明できることになる。

年式がお得な 22 年式、走行距離 3 万 km、傷は一番の多い「すごくきれい」で、グレードは低燃費仕様の L, セレクションは不要で、車検は残っていなくてよい。色はお高い白や黒は避けて無難なシルバーで、ナビもサンルーフも革も要らない。さて車の値段はいくらぐらいか？

OLS 回帰の予測値: predict 関数

まず、説明変数の値をいれたデータセットをつくる。

説明変数が多いのでいちいち指定するのは面倒だが、これは指定するしかない。

```
> (d01_all<-data.frame(nenshiki="22",
+                      kyori=3,
+                      kizu="すごくきれい",
+                      grade="L",
+                      selection="non",
+                      shaken=0,
+                      color="シルバー",
+                      NV=0,
+                      SR=0,
+                      kawa=0))
```

```
  nenshiki kyori         kizu grade selection shaken    color NV SR kawa
1       22     3 すごくきれい     L       non      0 シルバー  0  0    0
```

OLS 回帰の結果 o01_all を使ってここで指定した車の値段の期待値（回帰直線上の点）を求める。

```
> predict(o01_all,d01_all)
```

```
       1
133.6368
```

133.6 万円という結構お得な値段が出た。

予測区間： predict 関数の引数 interval="prediction"

実際に売られている同種の車はどのくらいの幅でばらつくかを予測する。

```
> predict(o01_all,d01_all,interval="prediction")
```

```
       fit      lwr      upr
1 133.6368 113.7816 153.4919
```

だいたい 113〜153 万円の間で売られている。

ここまで絞れれば、使える情報となる。

2.8.3 決定係数

バラツキを比較してみよう。

分散： var 関数

バラツキを分散で評価する。分散は、平均値と各値との差の二乗和。

たとえば、x_1, x_2, x_3 平均値は

$$\bar{x} = \frac{x_1 + x_2 + x_3}{3}$$

分散は、

$$V = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + (x_3 - \bar{x})^2}{3 - 1}$$

データのバラツキが大きいほど、分散は大きくなる。

```
> var(c(1,2,3))
```

```
[1] 1
```

```
> var(c(1,3,5))
```

```
[1] 4
```

OLS 回帰の分散は分解できる

そもそも説明変数として何も情報がない場合、車体価格のバラツキを分散で評価する。

```
> (rt01<-var(price))
```

```
[1] 805.4955
```

kyori の２次式だけで回帰した場合に、回帰式で説明できる、いわゆる推定値 $\hat{y}$ の分散（回帰線上の点だけの分散）

推定値は OLS 回帰の結果 o01_2 の中に入って、o01_2$fitted.values で取り出すことができる。

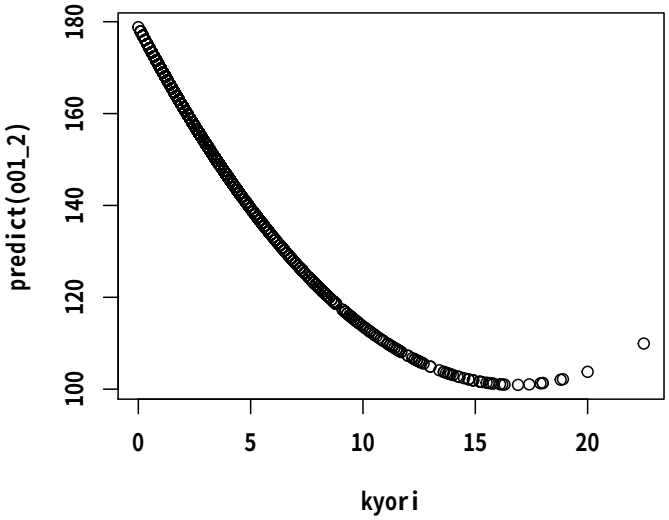
predict 関数でも求めることができる。 predict 関数は、引数に OLS 回帰の結果と予測に使う説明変数のデータを求めるが、OLS 回帰に使った説明変数そのものの予測値を使う場合はこれ（ d01 ）を省略できる。

```
> (rr01_2<-var(predict(o01_2)))
```

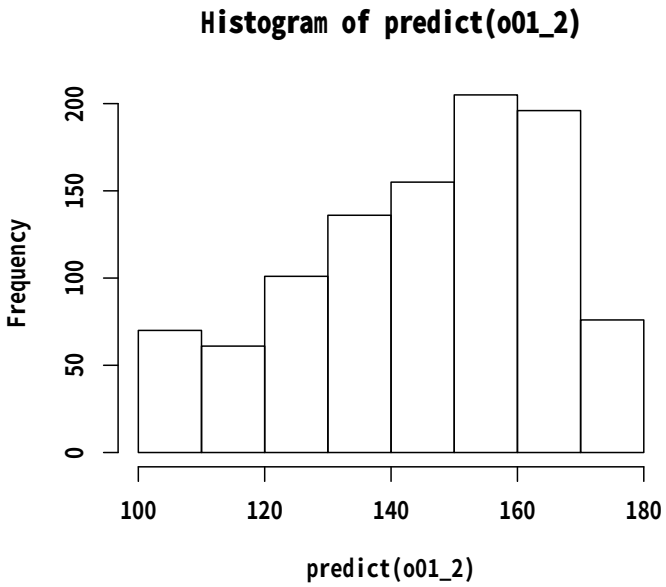
```
[1] 390.2625
```

分布の様子を確認するために、散布図とヒストグラムを描いておく。

```
> plot(predict(o01_2)~kyori)
```




```
> hist(predict(o01_2))
```



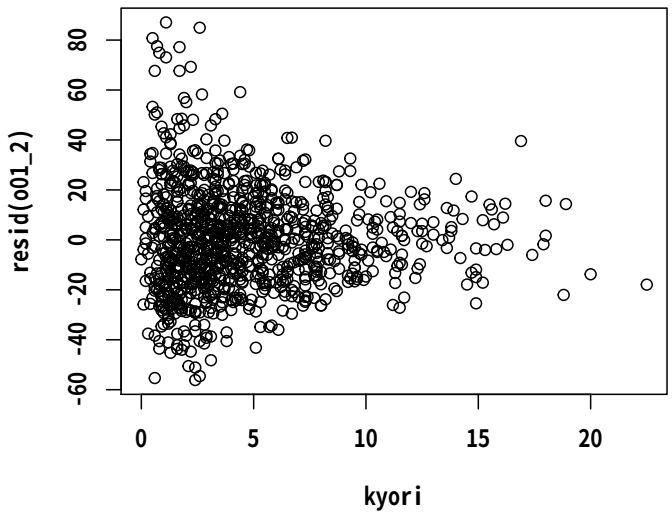
kyori の２次式だけで回帰した場合の残差（ kyori の２次式だけでは説明できない部分）の分散は

```
> (re01_2<-var(resid(o01_2)))
```

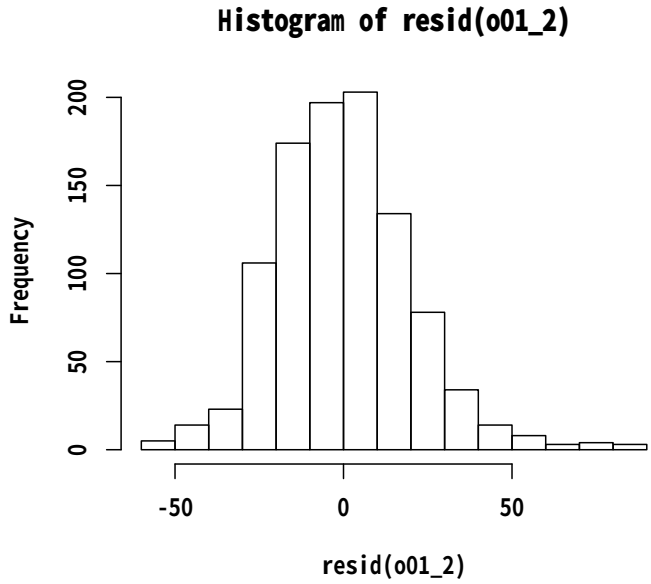
```
[1] 415.233
```

分布の様子を確認するために、これも散布図とヒストグラムを描いておく。

```
> plot(resid(o01_2)~kyori)
```




```
> hist(resid(o01_2))
```



ここで気づいて欲しいが、推定値の分散と残差の分散の和が、そのまま車体価格全体の分散と一致している。

```
> rt01
```

```
[1] 805.4955
```

```
> rr01_2+re01_2
```

```
[1] 805.4955
```

つまり、プリウスの中古車価格はいくらぐらいかと聞かれたときに、何も情報がなければ、分散 805.5 でしか答えられないが、走行距離がわかれば、そのうち情報があれば、価格のバラツキを押さえて答えることができる。その分散が 390.3 ということになる。ただし、残りの 415.2 の分散は答えられない。

決定係数：OLS 回帰の結果に対する summary 関数（とその要素 r.squared ）

何も情報がない場合のプリウスの中古車価格の分散 805.5 のうち、390.3 が走行距離の 2 次式で予測できる。割合で言うと、以下のとおり。

```
> rr01_2/rt01
```

```
[1] 0.4844999
```

つまり、被説明変数の分散の 48.45 %を、 kyori の 2 次式で説明したことになる。

この値を 決定係数 と言う。つまり、決定係数とは、被説明変数の分散のうち、回帰式で説明できる分散の割合である。

回帰式で説明できる分散と残差の分散の和が被説明変数の分散と一致するので、決定係数は、分散残差の分散／被説明変数の分散を 1 から引いた値として定義す場合もある。

```
> 1-re01_2/rt01
```

```
[1] 0.4844999
```

決定係数は、英語では **Multiple R-squared** と呼ばれ、そのまま **R** 二乗値 と呼ばれることもある。

決定係数は、OLS 回帰の結果を summary 関数で表示した時、係数の推定値の下に表示される。

```
> summary(o01_2)
```

```
Call:
lm(formula = price ~ kyori + I(kyori^2))

Residuals:
    Min      1Q  Median      3Q     Max
-56.132 -14.208  -1.227  12.159  87.068

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 178.79940    1.51887 117.719  < 2e-16 ***
kyori        -9.27416    0.51816 -17.898  < 2e-16 ***
I(kyori^2)    0.27615    0.03377   8.179 8.69e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.4 on 997 degrees of freedom
Multiple R-squared:  0.4845,        Adjusted R-squared:  0.4835
F-statistic: 468.5 on 2 and 997 DF,  p-value: < 2.2e-16
```

Multiple R-squard のところに表示されている。

決定係数だけ取り出したい場合は、$r.squared 。

```
> summary(o01_2)$r.squared
```

```
[1] 0.4844999
```

全ての説明変数を使った OLS 回帰の結果 o01_all の場合だと、決定係数は 0.878 と、 kyori の 2 次式だけで回帰した場合よりも高い。

```
> summary(o01_all)$r.squared
```

```
[1] 0.8779962
```

決定係数は、被説明変数の分散のうちの、回帰式で説明できる分散の割合だから、最大で 1、最小で 0 となる。回帰式で説明できる部分が大きいほど、決定係数は 1 に近づき、回帰式で説明できる部分が少ないほど決定係数は 0 に近くなる。[*9]

自由度調整済決定係数:OLS 回帰の結果に対する summary 関数の adj.r.squared

ところで、回帰式の予測値の精度を表す決定係数だが、実は説明変数を増やせば増やすほど、大きくなりやすい。

さきほど、全く無関係の変数 x01 、 x02 を説明変数に加えて OLS 回帰した結果 o01_all_x の決定係数を見てみよう。

```
> summary(o01_all_x)$r.squared
```

```
[1] 0.8782097
```

少しだが、決定係数が増えている。全く関係ない説明変数が増えることで決定係数が増えることはあまりうれしくない。

決定係数の定義は、回帰式で説明できる分散／被説明変数の分散だった。

たとえば。p 個の説明変数をもつ回帰式の場合、$\hat{y} = a + b_1x_1 + b_2x_2 + b_3x_3 + ...$ である。このとき、N 個の $\hat{y}$ の平均値を $\bar{\hat{y}}$ で表すと、決定係数は以下であらわすことができる。

$$R^2 = \frac{\frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}{N-1}}{\frac{\sum_{i=1}^{N}(y_i - \bar{y})^2}{N-1}}$$

さらに、回帰式で説明できる分散＋残差の分散＝被説明変数の分散、だったから、残差を $e_i = y_i - \hat{y}_i$ で表して、決定係数を以下で表すこともできる。

$$R^2 = 1 - \frac{\frac{\sum_{i=1}^{N} e_i^2}{N-1}}{\frac{\sum_{i=1}^{N}(y_i - \bar{y})^2}{N-1}}$$

残差の分散のところで、e_i の平均値を引いていないが、これは 0 になるから。

しかし、この $\hat{y}_i$ は、$p+1$ 個の係数 $a, b_1, b_2, b_3, ...$ を推定して求めたものなので、データの 自由度 が減っている。この部分を差し引いて、以下のような係数を定義した。

$$adj.R^2 = 1 - \frac{\frac{\sum_{i=1}^{N} e_i^2}{N-p-1}}{\frac{\sum_{i=1}^{N}(y_i - \bar{y})^2}{N-1}}$$

これを 自由度調整済決定係数 と呼ぶ。

自由度調整済決定係数は、決定係数を以下で変換できる。

$$adj.R^2 = 1 - (1 - R^2)\frac{N-1}{N-p-1}$$

さきほどの kyori の 2 次式だけで OLS 回帰した結果だと、以下で計算できる。

```
> 1-(1-summary(o01_2)$r.squared)*(1000-1)/(1000-1-2)
```

```
[1] 0.4834658
```

[*9] よく、決定係数を回帰式の「あてはまりのよさ」と表現する人がいるが、これは間違い。あくまで被説明変数のバラツキのどのくらいの割合を、回帰式で説明できたか、その割合ということで、回帰式に使った説明変数の値を指定した場合に、その予測値の幅がどのくらい狭く言えるかということ。

走行距離によって、車体価格がどのくらい下がるかということだけに関心がある場合、決定係数は 0.48 でも全然問題ない。その他の部分は、他の変数で説明してください、それについては私は関心ありません、という場合もあります。

もちろん、OLS 回帰の結果 o01_2 に対する summary 関数でも、 Adjusted R-squared というところに表示されている。

```
> summary(o01_2)
```

```
Call:
lm(formula = price ~ kyori + I(kyori^2))

Residuals:
    Min      1Q  Median      3Q     Max
-56.132 -14.208  -1.227  12.159  87.068

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 178.79940    1.51887 117.719  < 2e-16 ***
kyori        -9.27416    0.51816 -17.898  < 2e-16 ***
I(kyori^2)    0.27615    0.03377   8.179 8.69e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.4 on 997 degrees of freedom
Multiple R-squared:  0.4845,        Adjusted R-squared:  0.4835
F-statistic: 468.5 on 2 and 997 DF,  p-value: < 2.2e-16
```

自由度調整済決定係数だけを取り出したかったら、$adj.r.squared で。

```
> summary(o01_2)$adj.r.squared
```

```
[1] 0.4834658
```

全ての説明変数による回帰の結果 o01_all だと

```
> summary(o01_all)$adj.r.squared
```

```
[1] 0.8748647
```

余計な説明変数を加えた回帰の結果 o01_all_x だと

```
> summary(o01_all_x)$adj.r.squared
```

```
[1] 0.8748267
```

2.8.4　とりあえず OLS 回帰するための手順（まとめ）

変数の整理

1. 項目変数にするか、数値のままか決める： factor 関数
2. 項目変数の場合、基準とする項目をどれにするかを決める: factor 関数の引数 levels=
3. 欠損値を除く： na.omit 関数

OLS 回帰

4. 関係してそうな変数を全て入れて回帰する： lm 関数
5. 数値変数をどうするか決める（そのまま？ 片対数？ 両対数？ 2 次式？ 3 次式？）： AIC 関数
6. 変数の取捨選択を行う： step 関数

OLS 回帰の結果を評価する

7. 結果の要約を表示する: summary 関数
8. 特定の説明変数の効果・影響を確認する： coefficients と Std. Error 、並びに p 値の確認
9. 特定のデータセットの予測区間を確認する： predict 関数の引数 interval=”prediction”

第 3 章

いろいろな回帰

3.1　二項ロジット

OLS 以外の回帰分析を、とりあえず使い方だけ紹介する。最初は、非説明変数が 0,1 のデータの場合の回帰分析。

3.1.1　前回までの作業

前回までの作業に続いて、ここの作業は不要。次の「二項ロジット」へ飛んでください。

データの読み込み： read.csv 関数

引き続き prius.csv を使う。

このファイルを作業フォルダに置く。

作業フォルダを変更したい場合は、setwd("c:/Users/yoshino/.../") という具合に設定する。

ただし、その場合、ファイルパスのの区切りが￥ではなくて / であることに注意する。

そして、 read.csv 関数で、 prius.csv を読み込み、 d01 という名前に代入する。

```
> d01<-read.csv("prius.csv")
```

次に、項目変数への変換＆基準となる項目の設定をする

nenshiki （年式）を項目変数に： factor 関数

```
> d01$nenshiki<-factor(d01$nenshiki)
> summary(d01$nenshiki)
```

```
 21  22  23  24
226 521 172  81
```

kizu （傷）の基準を「普通」に： factor 関数の引数 levels=

```
> d01$kizu<-factor(d01$kizu,
+                  levels=c("普通","きれい","すごくきれい","修復歴あり"))
> summary(d01$kizu)
```

```
      普通      きれい すごくきれい    修復歴あり
       122         306         414         158
```

grade （グレード）の基準を「S」に： factoro 関数の引数 levels=

```
> d01$grade<-factor(d01$grade,levels=c("S","G","L"))
> summary(d01$grade)
```

```
  S   G   L
744 151 105
```

selection （特別仕様）の基準を「non（なんにも無し）」に： factoro 関数の引数 levels=

```
> d01$selection<-factor(d01$selection,
+                       levels=c("non","TS","TSG","LED"))
> summary(d01$selection)
```

```
non  TS TSG LED
653 274   8  65
```

color （色）の基準を「シルバー」に： factoro 関数の引数 levels=

```
> d01$color<-factor(d01$color,
+                   levels=c("シルバー","白","黒","青","紺",
+                            "赤","パープル","ワイン","緑"))
> summary(d01$color)
```

```
シルバー       白       黒       青       紺       赤 パープル    ワイン
     261      372      228       71       13       24       15       11
      緑
       5
```

NV （カーナビ）、 SR （サンルーフ）、 kawa ` （革）については、無しが 0、付きが 1 で入力されており、項目変数に変換してもよいが、OLS 回帰をやる分には、何の意味もないので、そのままにしておく。

3.1.2　二項ロジット

OLS は、最もよく使われる回帰分析だが、これは被説明変数が制限のない連続変数の場合にしか使えない。

被説明変数が連続変数以外の場合というのに、どんなのがあるかというと・・・

1. アンケートの回答で「はい」「いいえ」のように、数値にすると、0,1 の場合
2. 「どこどこへ何回行ったか」といったような、0,1,2,3,4,・・・といった値
3. 値段が「安い」「普通」「高い」というように、実際には連続変数かもしれないが、データにする段階で、カテゴリー化されてしまっている値
4. 途中までは連続変数だが、「130 万円以上」というように、ある値以上はひとくくりにされているデータ

などなど、OLS では無理のある回帰分析がいろいろある。

今回は、ひとまず、これらの理論的な基礎は飛ばして、使い方だけやってみましょう。

オークションで、ある中古車店がどの中古車に対して「買い」を入れたか（実際に買えたかどうかはわからない）のデータがある（ dealer.csv ）。

prius.csv の中古車プリウス 1000 台のデータについて、件の中古車店が「買い」を入れた場合は 1、そうで

なければ 0 が入力されている。

つまり、0 と 1 だけの 1000 個のデータ。

このファイルを読み込み、 dA に代入する

```
> dA<-read.csv("dealer.csv")
```

head 関数で最初のいくつかのデータを確認してみる

```
> head(dA)
```

```
  dealer
1      0
2      0
3      0
4      0
5      0
6      0
```

dealer という名前で 0,1 のデータが並んでいる。

summary 関数で要約を出力する

```
> summary(dA)
```

```
    dealer
 Min.   :0.000
 1st Qu.:0.000
 Median :0.000
 Mean   :0.089
 3rd Qu.:0.000
 Max.   :1.000
```

数値なので、分布が表示される。0 が何個、1 が何個かを表示するには, いったん factor 関数で項目変数にしておく。

```
> summary(factor(dA$dealer))
```

```
  0   1
911  89
```

1 が 78 個。つまり、この中古車屋さんは、1000 台のうち、89 台を落札した。

この中古車屋さんは、どんな中古車を落札したでしょうか？ ーーーこれを、二項ロジットでつきとめる！

データフレームをくっつけてひとつにする: data.frame 関数

データがが d01 と dA の 2 つに分かれているので、これをひとつのデータフレームにくっつけておくと便利。

d01 も dA は対応する同じ行数のデータフレーム。これらをくっつけるには、 data.frame 関数の引数に、カンマ（ , ）で区切ってつなぐだけ。

これを d02 に代入する。

```
> d02<-data.frame(d01,dA)
```

変数の要約を出力する: summary 関数

summary 関数で d02 を確認する。

```
> summary(d02)
```

```
 nenshiki     kyori                   kizu         price       grade    selection
 21:226   Min.   : 0.000   普通         :122   Min.   : 76.5   S:744   non:653
 22:521   1st Qu.: 2.000   きれい       :306   1st Qu.:125.5   G:151   TS :274
 23:172   Median : 3.600   すごくきれい:414   Median :144.5   L:105   TSG:  8
 24: 81   Mean   : 4.643   修復歴あり   :158   Mean   :145.3           LED: 65
          3rd Qu.: 6.400                       3rd Qu.:162.0
          Max.   :22.500                       Max.   :256.0

    shaken            color           NV              SR              kawa
Min.   : 0.000   白       :372   Min.   :0.00   Min.   :0.000   Min.   :0.000
1st Qu.: 0.000   シルバー:261   1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.000
Median : 5.000   黒       :228   Median :1.00   Median :0.000   Median :0.000
Mean   : 7.014   青       : 71   Mean   :0.65   Mean   :0.032   Mean   :0.044
3rd Qu.:13.000   赤       : 24   3rd Qu.:1.00   3rd Qu.:0.000   3rd Qu.:0.000
Max.   :31.000   パープル: 15   Max.   :1.00   Max.   :1.000   Max.   :1.000
                 (Other) : 29
    dealer
Min.   :0.000
1st Qu.:0.000
Median :0.000
Mean   :0.089
3rd Qu.:0.000
Max.   :1.000
```

二項ロジットで回帰する： glm 関数の引数 family=binomial

被説明変数が連続数ではなく、0,1 の二値の場合は、OLS で回帰するとうまく推定できない。そこで、二項ロジット分析というのを使う。

OLS で、y に対して x_1, x_2 を回帰する場合、は以下を最小化する a, b_1, b_2 を求めた。

$$\sum_{i=1}^{N}(y_i - a - b_1 x_{1i} - b_2 x_{2i})^2$$

二項ロジットで、y に対して x_1, x_2 を回帰する場合、は以下を最<大>化する a, b_1, b_2 を求める。

$$\prod_{i=1}^{N} p_i^{y_i}(1-p_i)^{1-y_i}$$

ただし、

$$p_i = \frac{1}{1+\exp(-a - b_1 x_{1i} - b_2 x_{2i})}$$

で、$\prod$ は、和 ($\sum$) ではなく、積を表す。$\exp(x)$ は e^x のこと。

二項ロジット分析のカラクリについては、ひとまず置いておいて、R による推定方法だけ説明しておく。

R で二項ロジットを行う場合は、 glm 関数を使い、その引数に family=binomial と入れておく。あとは、OLS の場合と同じ。

formula=dealer . で、 dealer 以外のすべての変数を説明変数に入れることを意味する。

```
> o02<-glm(dealer~.,d02,family=binomial)
> summary(o02)
```

```
Call:
glm(formula = dealer ~ ., family = binomial, data = d02)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4091  -0.4080  -0.3201  -0.2242   2.7696
```

```
Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)       -4.638065   1.834880  -2.528   0.0115 *
nenshiki22        -0.187263   0.329752  -0.568   0.5701
nenshiki23        -0.569318   0.505673  -1.126   0.2602
nenshiki24        -0.753247   0.766378  -0.983   0.3257
kyori             -0.053865   0.063405  -0.850   0.3956
kizu きれい        0.201748   0.427844   0.472   0.6373
kizu すごくきれい  -0.200320   0.453871  -0.441   0.6590
kizu 修復歴あり    -0.820634   0.585360  -1.402   0.1609
price              0.014870   0.012805   1.161   0.2456
gradeG             0.417968   0.381449   1.096   0.2732
gradeL             0.762446   0.400981   1.901   0.0572 .
selectionTS       -0.530255   0.342271  -1.549   0.1213
selectionTSG     -14.624931 837.326216  -0.017   0.9861
selectionLED       0.161192   0.566344   0.285   0.7759
shaken             0.007369   0.018272   0.403   0.6867
color 白          -0.032296   0.383396  -0.084   0.9329
color 黒          -0.070622   0.446344  -0.158   0.8743
color 青           2.787685   0.381935   7.299  2.9e-13 ***
color 紺         -13.777216 655.341412  -0.021   0.9832
color 赤           0.555959   0.811313   0.685   0.4932
color パープル     0.214208   1.109292   0.193   0.8469
color ワイン       1.237040   0.873355   1.416   0.1567
color 緑           1.780710   1.234308   1.443   0.1491
NV                 0.318093   0.289519   1.099   0.2719
SR                -1.734062   1.101657  -1.574   0.1155
kawa              -0.466671   0.765753  -0.609   0.5422
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 600.44  on 999  degrees of freedom
Residual deviance: 484.14  on 974  degrees of freedom
AIC: 536.14

Number of Fisher Scoring iterations: 15
```

結果の出力も、t 値の代わりに z 値が表示されているだけで、あとは OLS と同じ。z 値も t 値と同じと考えて問題ない。

p 値が大きい変数も多く、取捨選択が必要だ。

ステップワイズ法で説明変数の取捨選択: step 関数

AIC に基づいて説明変数を取捨選択するには、OLS の時と同様に step 関数が使える。引数に分析結果 o02 を入れるだけ。

```
> o02s<-step(o02)
```

```
Start:  AIC=536.14
dealer ~ nenshiki + kyori + kizu + price + grade + selection +
    shaken + color + NV + SR + kawa

            Df Deviance    AIC
- nenshiki   3   485.62 531.62
- shaken     1   484.30 534.30
- selection  3   488.39 534.39
- kawa       1   484.53 534.53
- kyori      1   484.88 534.88
- kizu       3   489.30 535.30
- NV         1   485.38 535.38
- price      1   485.49 535.49
<none>           484.14 536.14
- grade      2   488.35 536.35
- SR         1   487.80 537.80
- color      8   564.29 600.29
```

```
Step:  AIC=531.62
dealer ~ kyori + kizu + price + grade + selection + shaken +
    color + NV + SR + kawa

            Df Deviance    AIC
- selection  3   488.92 528.92
- shaken     1   485.75 529.75
- kawa       1   485.82 529.82
- price      1   485.95 529.95
- kyori      1   487.12 531.12
- NV         1   487.27 531.27
- grade      2   489.42 531.42
<none>           485.62 531.62
- kizu       3   492.06 532.06
- SR         1   488.54 532.54
- color      8   565.41 595.41

Step:  AIC=528.92
dealer ~ kyori + kizu + price + grade + shaken + color + NV +
    SR + kawa

         Df Deviance    AIC
- price   1   488.93 526.93
- kawa    1   489.05 527.05
- shaken  1   489.15 527.15
- NV      1   490.32 528.32
- grade   2   492.85 528.85
<none>        488.92 528.92
- SR      1   491.01 529.01
- kyori   1   491.70 529.70
- kizu    3   496.15 530.15
- color   8   566.45 590.45

Step:  AIC=526.93
dealer ~ kyori + kizu + grade + shaken + color + NV + SR + kawa

         Df Deviance    AIC
- kawa    1   489.05 525.05
- shaken  1   489.21 525.21
- NV      1   490.34 526.34
<none>        488.93 526.93
- grade   2   492.95 526.95
- SR      1   491.02 527.02
- kizu    3   496.63 528.63
- kyori   1   493.94 529.94
- color   8   570.22 592.22

Step:  AIC=525.05
dealer ~ kyori + kizu + grade + shaken + color + NV + SR

         Df Deviance    AIC
- shaken  1   489.34 523.34
- NV      1   490.42 524.42
- grade   2   492.96 524.96
<none>        489.05 525.05
- SR      1   491.21 525.21
- kizu    3   496.71 526.71
- kyori   1   494.00 528.00
- color   8   571.50 591.50

Step:  AIC=523.34
dealer ~ kyori + kizu + grade + color + NV + SR

        Df Deviance    AIC
- NV     1   490.82 522.82
- grade  2   493.25 523.25
<none>       489.34 523.34
- SR     1   491.41 523.41
- kizu   3   496.86 524.86
- kyori  1   494.88 526.88
- color  8   571.56 589.56

Step:  AIC=522.82
dealer ~ kyori + kizu + grade + color + SR
```

```
        Df Deviance    AIC
- grade  2   494.65 522.65
- SR     1   492.76 522.76
<none>       490.82 522.82
- kizu   3   498.98 524.98
- kyori  1   496.80 526.80
- color  8   575.37 591.37

Step:  AIC=522.65
dealer ~ kyori + kizu + color + SR

        Df Deviance    AIC
- SR     1   496.63 522.63
<none>       494.65 522.65
- kizu   3   502.96 524.96
- kyori  1   499.55 525.55
- color  8   581.37 593.37

Step:  AIC=522.63
dealer ~ kyori + kizu + color

        Df Deviance    AIC
<none>       496.63 522.63
- kyori  1   501.19 525.19
- kizu   3   505.92 525.92
- color  8   583.33 593.33
```

ずいぶん試行錯誤が続いたが、結果は以下のとおり

```
> summary(o02s)
```

```
Call:
glm(formula = dealer ~ kyori + kizu + color, family = binomial,
    data = d02)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4011  -0.3935  -0.3474  -0.2386   2.7218

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -2.38599    0.49730  -4.798 1.60e-06 ***
kyori             -0.08949    0.04389  -2.039   0.0414 *
kizu きれい        0.32720    0.41699   0.785   0.4326
kizu すごくきれい  0.04432    0.42065   0.105   0.9161
kizu 修復歴あり   -1.01106    0.57277  -1.765   0.0775 .
color 白          -0.04056    0.34902  -0.116   0.9075
color 黒          -0.15079    0.40096  -0.376   0.7069
color 青           2.66920    0.36435   7.326 2.37e-13 ***
color 紺         -13.90031  656.16512  -0.021   0.9831
color 赤           0.27259    0.79274   0.344   0.7310
color パープル     0.01758    1.08012   0.016   0.9870
color ワイン       0.97053    0.83676   1.160   0.2461
color 緑           1.72882    1.20215   1.438   0.1504
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 600.44  on 999  degrees of freedom
Residual deviance: 496.63  on 987  degrees of freedom
AIC: 522.63

Number of Fisher Scoring iterations: 15
```

走行距離（ kyori ）と車体の傷（ kizu ）と色 (color) が残った。色は青色の係数が大きく、p 値も小さい。

説明変数を二次関数にする

kyori を二次関数にしたかったら、OLS と同様に二次の項を I(kyori^2) として加えればよい。

```
> o02_2<-glm(dealer~.+I(kyori^2),d02,family=binomial)
```

ステップワイズ法も行う。

ステップワイズ法の経過の出力は引数に trace=0 とすることで、止めることがきる。

```
> o02s_2<-step(o02_2, trace=0)
> summary(o02s_2)
```

```
Call:
glm(formula = dealer ~ kyori + kizu + color + I(kyori^2), family = binomial,
    data = d02)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3537  -0.3949  -0.3594  -0.2248   3.2940

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -3.00294    0.58211  -5.159 2.49e-07 ***
kyori              0.24302    0.16357   1.486   0.1373
kizu きれい        0.29629    0.42508   0.697   0.4858
kizu すごくきれい  0.05505    0.42091   0.131   0.8959
kizu 修復歴あり   -0.98606    0.57880  -1.704   0.0884 .
color 白          -0.04060    0.34970  -0.116   0.9076
color 黒          -0.17433    0.40175  -0.434   0.6643
color 青           2.66669    0.36673   7.272 3.55e-13 ***
color 紺         -13.92331  657.83975  -0.021   0.9831
color 赤           0.30516    0.79353   0.385   0.7006
color パープル     0.14550    1.07982   0.135   0.8928
color ワイン       1.07151    0.83728   1.280   0.2006
color 緑           1.76125    1.18856   1.482   0.1384
I(kyori^2)        -0.03027    0.01552  -1.950   0.0512 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 600.44  on 999  degrees of freedom
Residual deviance: 491.05  on 986  degrees of freedom
AIC: 519.05

Number of Fisher Scoring iterations: 15
```

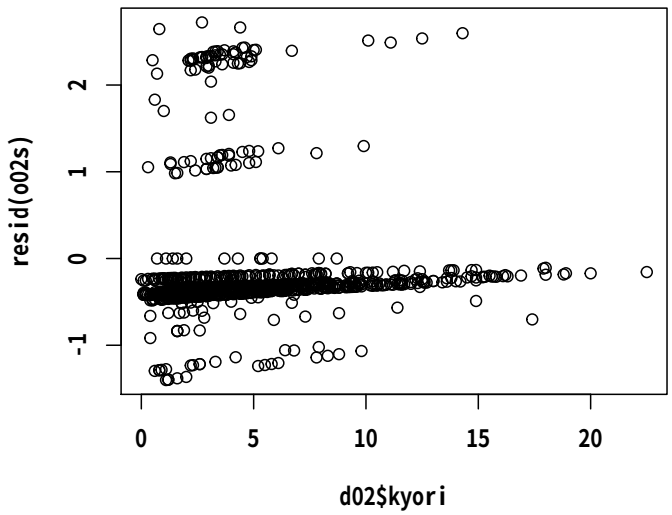
散布図

ここまでは、OLS とほとんど同じだが、被説明変数が 0 ,1 なので OLS のように散布図が描けないは難点だ。

残差も OLS のように resid(o02s) という具合に得ることができない。

無理矢理描くと、以下のようになる。

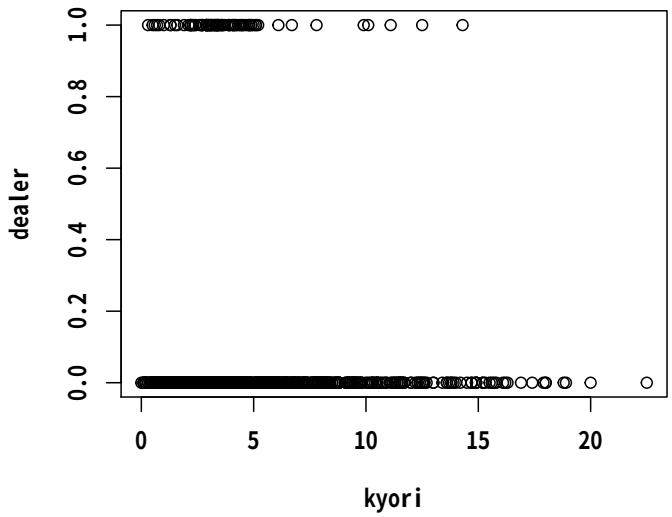
```
> plot(resid(o02s)~d02$kyori)
```



そもそも残差の概念がちょっと違う。

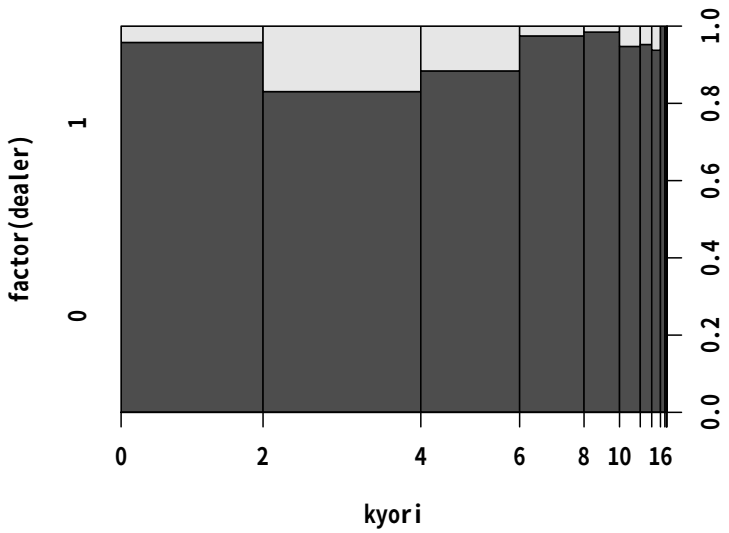
回帰する前の被説明変数と説明変数との関係もわかりにくい！！

```
> plot(dealer~kyori,d02)
```



factor 関数を使って dealer を項目変数として扱えばまだまし。

```
> plot(factor(dealer)~kyori,d02)
```



走行距離 2〜6 万 km の間で選択される (dealer が 1) 確率が高いことがわかる。

この距離とそれ以外に分けて kyori を扱ったらどうか？

連続変数を指定した区間に分割する: cut 関数

kyori を 0 2、2 6、6〜の 3 区間に分けるには, cut 関数を使い、引数 breaks= （省略形 br= ）に区切りとなる区間を指定すればよい。

```
> kyori_b<-cut(d02$kyori,br=c(0,2,6,30))
> summary(kyori_b)
```

```
 (0,2]  (2,6] (6,30]   NA's
   259    470    270      1
```

１個 NA （欠損値）が出てしまった。そういえば、走行距離 0 が 1 個あった。設定した区間が、 (0,2] と、0 より大きいところから始まっている。

最小値も含むように、区間を設定するには、引数 inclued.lowest=T をつければよい。

```
> kyori_b<-cut(d02$kyori,br=c(0,2,6,30),include.lowest=T)
> summary(kyori_b)
```

```
[0,2]  (2,6] (6,30]
  260    470    270
```

kyori_b は３つの区間に区切られた項目変数となった。

これを kyori の代わりに使う。

データフレーム d02 の変数 kyori に kyori_b を上書きしたものを、 d02b として保存する。

```
> d02b<-d02 #d02 をまるごとコピー
> d02b$kyori<-kyori_b #kyori のデータを入れ替える
```

あとは、さきほどと同じ手順。

二項ロジット分析： glm 関数に引数 family=binomial

全ての変数を説明変数とした二項ロジット分析。用いるデータフレームが d02b であることだけが異なる。

```
> o02_b<-glm(dealer~.,family=binomial,d02b)
```

ステップワイズ法で説明変数の取捨選択： step 関数

AIC に基づいて説明変数を選ぶ。（引数 trace=0 で、経過は出力しない）

```
> o02s_b<-step(o02_b,trace=0)
> summary(o02s_b)
```

```
Call:
glm(formula = dealer ~ kyori + price + selection + color + NV +
    SR, family = binomial, data = d02b)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9450  -0.4356  -0.2261  -0.1493   2.8828

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)   -7.090e+00  1.251e+00  -5.668 1.45e-08 ***
kyori(2,6]     2.187e+00  4.168e-01   5.247 1.55e-07 ***
kyori(6,30]    6.263e-01  6.039e-01   1.037  0.29964
price          1.958e-02  7.283e-03   2.688  0.00718 **
selectionTS   -9.444e-01  3.337e-01  -2.830  0.00465 **
selectionTSG  -1.558e+01  1.332e+03  -0.012  0.99066
selectionLED  -4.123e-01  5.379e-01  -0.767  0.44333
color 白      -3.406e-01  3.762e-01  -0.905  0.36531
color 黒      -2.748e-01  4.335e-01  -0.634  0.52608
color 青       3.113e+00  4.090e-01   7.610 2.73e-14 ***
color 紺      -1.480e+01  1.029e+03  -0.014  0.98853
color 赤       6.734e-01  8.250e-01   0.816  0.41438
color パープル 2.945e-01  1.118e+00   0.263  0.79226
color ワイン   1.678e+00  8.997e-01   1.865  0.06217 .
color 緑       9.838e-01  1.173e+00   0.838  0.40177
NV             4.443e-01  2.941e-01   1.510  0.13093
SR            -1.514e+00  1.072e+00  -1.412  0.15781
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 600.44  on 999  degrees of freedom
Residual deviance: 447.73  on 983  degrees of freedom
AIC: 481.73

Number of Fisher Scoring iterations: 16
```

AIC で計測モデルを比較: AIC 関数

kyori をそのまま (o02s)、２次式 (o02s_2)、それと 2〜6 万 km を切り分けた (o02s_b)、３種類を説明変数として二項ロジットによる回帰を行った。

どれがよいかは AIC で判断する。

```
> AIC(o02s)
```

```
[1] 522.6327
```

```
> AIC(o02s_2)
```

```
[1] 519.0491
```

```
> AIC(o02s_b)
```

```
[1] 481.7275
```

結局、2〜6 万 km を切り分けた o02s_b の当てはまりが最もよかった。

結局、この中古車屋さんは、走行距離が 2〜6 万 km ぐらい走った価格がやや高めて、青色の車を探している。さらに、ツーリングセレクションとかがつくと選択確率が下がるようだ。

3.1.3　例題：Titanic 号の生存

R には、Tainanic 号の生存数データが入っている。練習問題として、これを使った二項ロジット分析をしてみよう

データを見てみよう

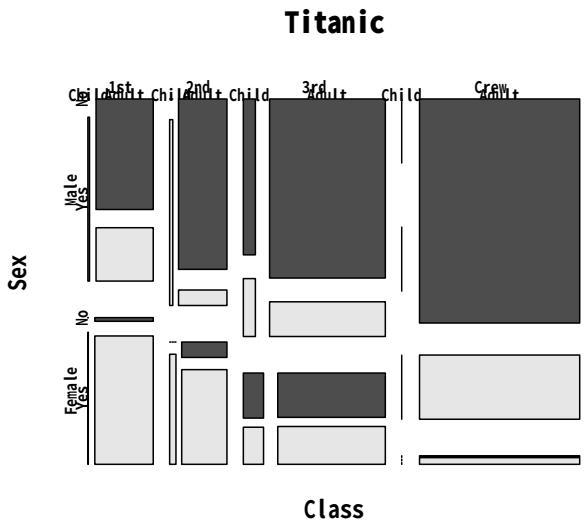
このデータは集計データ。

```
> Titanic
```

```
, , Age = Child, Survived = No

      Sex
Class  Male Female
  1st     0      0
  2nd     0      0
  3rd    35     17
  Crew    0      0

, , Age = Adult, Survived = No

      Sex
Class  Male Female
  1st   118      4
  2nd   154     13
  3rd   387     89
  Crew  670      3

, , Age = Child, Survived = Yes

      Sex
Class  Male Female
  1st     5      1
  2nd    11     13
  3rd    13     14
  Crew    0      0

, , Age = Adult, Survived = Yes

      Sex
Class  Male Female
  1st    57    140
  2nd    14     80
  3rd    75     76
  Crew  192     20
```

グラフで表すと次のように表せる

```
> plot(Titanic,color=T)
```



集計データを個別データに変換する

クロス集計表をデータフレームにすると、度数 (Freq) のついたデータに変換する。

```
> t01fd<-data.frame(Titanic)
> head(t01fd)
```

```
  Class    Sex   Age Survived Freq
1   1st   Male Child       No    0
2   2nd   Male Child       No    0
3   3rd   Male Child       No   35
4  Crew   Male Child       No    0
5   1st Female Child       No    0
6   2nd Female Child       No    0
```

これを個票にするには、 t01fd の各変数を対応する Freq の数だけ繰り返せばよい。

lapply 関数をデータフレーム適用すると、データフレームの列ごとに指定した関数を当てはめてくれる。

以下は、 i を t01fd$Freq 個発生させよという関数を定義して、 lapply の中に入れた。

```
> d01p<-lapply(t01fd,function(i) rep(i,t01fd$Freq))
> lapply(d01p,head)
```

```
$Class
[1] 3rd 3rd 3rd 3rd 3rd 3rd
Levels: 1st 2nd 3rd Crew

$Sex
[1] Male Male Male Male Male Male
Levels: Male Female

$Age
[1] Child Child Child Child Child Child
Levels: Child Adult

$Survived
[1] No No No No No No
```

```
Levels: No Yes

$Freq
[1] 35 35 35 35 35 35
```

これをデータフレームに変換

```
> d01<-data.frame(d01p)
> head(d01)
```

```
  Class  Sex   Age Survived Freq
1   3rd Male Child       No   35
2   3rd Male Child       No   35
3   3rd Male Child       No   35
4   3rd Male Child       No   35
5   3rd Male Child       No   35
6   3rd Male Child       No   35
```

Freq の列はもう不要なので削除する。

なぜか知らないが d01[,”Freq”] （ d01 から Freq という名前の列だけ抽出）という指定は出来るが、d01[,-”Freq”] （ d01 から Freq という名前の列だけ削除）という指定ができないので、ちょっと回りくどいが、 which 関数を使って、d01 で列の名前が Freq というのは何番目にあるのかというのを指定する形をとっている。

```
> d01<-d01[,-which(colnames(d01)=="Freq")]
> head(d01)
```

```
  Class  Sex   Age Survived
1   3rd Male Child       No
2   3rd Male Child       No
3   3rd Male Child       No
4   3rd Male Child       No
5   3rd Male Child       No
6   3rd Male Child       No
```

すべて項目変数だが、基準となる項目を指定する。「1st クラス」の「大人」の「男」を基準として、その他の人が、それよりもどのくらい多く（または少なく）「死んだ」かどうかを見ることにした。

```
> d01$Class<-factor(d01$Class,levels=c("1st","2nd","3rd","Crew"))
> d01$Sex<-factor(d01$Sex,levels=c("Male","Female"))
> d01$Age<-factor(d01$Age,levels=c("Adult","Child"))
> d01$Survived<-factor(d01$Survived,levels=c("Yes","No"))
> summary(d01)
```

```
 Class         Sex          Age       Survived
1st :325   Male  :1731   Adult:2092   Yes: 711
2nd :285   Female: 470   Child: 109   No :1490
3rd :706
Crew:885
```

二項ロジット分析

全ての変数について（授業ではまだやっていないが）交差項も含めて説明変数とした（ .^2 で指定）

```
> o01<-glm(Survived~.^2,family=binomial,data=d01)
> summary(o01)
```

```
Call:
glm(formula = Survived ~ .^2, family = binomial, data = d01)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2293  -0.2374   0.5952   0.7099   2.6771

Coefficients: (1 not defined because of singularities)
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)           0.72763    0.16130   4.511 6.45e-06 ***
Class2nd              1.67026    0.32240   5.181 2.21e-07 ***
Class3rd              0.91330    0.20478   4.460 8.20e-06 ***
ClassCrew             0.52215    0.18088   2.887  0.00389 **
SexFemale            -4.28298    0.53213  -8.049 8.36e-16 ***
AgeChild            -16.99213  920.38640  -0.018  0.98527
Class2nd:SexFemale    0.06801    0.67120   0.101  0.91929
Class3rd:SexFemale    2.79995    0.56875   4.923 8.52e-07 ***
ClassCrew:SexFemale   1.13608    0.82048   1.385  0.16616
Class2nd:AgeChild    -0.84881 1005.81952  -0.001  0.99933
Class3rd:AgeChild    16.34159  920.38647   0.018  0.98583
ClassCrew:AgeChild         NA         NA      NA       NA
SexFemale:AgeChild    0.68679    0.52541   1.307  0.19116
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2769.5  on 2200  degrees of freedom
Residual deviance: 2097.5  on 2189  degrees of freedom
AIC: 2121.5

Number of Fisher Scoring iterations: 15
```

ステップワイズ法で変数を選択

```
> o01s<-step(o01)
```

```
Start:  AIC=2121.49
Survived ~ (Class + Sex + Age)^2

           Df Deviance    AIC
- Sex:Age   1   2099.2 2121.2
<none>          2097.5 2121.5
- Class:Age 2   2134.8 2154.8
- Class:Sex 3   2162.5 2180.5

Step:  AIC=2121.18
Survived ~ Class + Sex + Age + Class:Sex + Class:Age

           Df Deviance    AIC
<none>          2099.2 2121.2
- Class:Age 2   2143.4 2161.4
- Class:Sex 3   2174.4 2190.4
```

```
> summary(o01s)
```

```
Call:
glm(formula = Survived ~ Class + Sex + Age + Class:Sex + Class:Age,
    family = binomial, data = d01)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2293  -0.2374   0.6057   0.7099   2.6771

Coefficients: (1 not defined because of singularities)
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)         0.72763    0.16130   4.511 6.45e-06 ***
Class2nd            1.67026    0.32240   5.181 2.21e-07 ***
Class3rd            0.87507    0.20176   4.337 1.44e-05 ***
```

```
ClassCrew             0.52215    0.18088   2.887  0.00389 **
SexFemale            -4.28298    0.53213  -8.049 8.36e-16 ***
AgeChild            -16.85008  858.44949  -0.020  0.98434
Class2nd:SexFemale    0.06801    0.67120   0.101  0.91929
Class3rd:SexFemale    2.89768    0.56364   5.141 2.73e-07 ***
ClassCrew:SexFemale   1.13608    0.82048   1.385  0.16616
Class2nd:AgeChild    -0.77411  933.63527  -0.001  0.99934
Class3rd:AgeChild    16.51217  858.44953   0.019  0.98465
ClassCrew:AgeChild         NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2769.5  on 2200  degrees of freedom
Residual deviance: 2099.2  on 2190  degrees of freedom
AIC: 2121.2

Number of Fisher Scoring iterations: 15
```

1st クラスよりも 2nd クラスや 3rd クラス、乗務員の人が多く死んだ。

特に 2nd クラスの人が多く死んだ。

女性はだいぶ助かっているが、その中で、3rd クラスの女性はあまり助かっていない。

3.2 ポアソン回帰

OLS 以外の回帰分析を紹介だけ。ポアソン回帰について

3.2.1 前回までの作業

「中古車価格の OLS 推定 (８)」をやって、連続してこれをやる場合は、ここの作業は不要。

```
> d01<-read.csv("prius.csv")
> d01$nenshiki<-factor(d01$nenshiki)
> d01$kizu<-factor(d01$kizu,
+                  levels=c("普通","きれい","すごくきれい","修復歴あり"))
> d01$grade<-factor(d01$grade,levels=c("S","G","L"))
> d01$selection<-factor(d01$selection,
+                       levels=c("non","TS","TSG","LED"))
> d01$color<-factor(d01$color,
+                   levels=c("シルバー","白","黒","青","紺",
+                            "赤","パープル","ワイン","緑"))
```

3.2.2 ポアソン回帰

オークションで、各中古車に対して何件の「買い」が入ったか（入札価格がいくらだったか、誰かかったかは関係無く、何人が買う意志を示したか）のデータがある（ bit.csv)。

prius.csv の中古車プリウス 1000 台のデータについて、１人が「買い」を入れた場合は 1、２人が「買い」を入れた場合は 2、誰も買いを入れなかった場合は 0 が入力されている。

つまり、0、1、2、3、... といった 1000 個のデータ。

このファイルを読み込み、 dB に代入する

```
> dB<-read.csv("bit.csv")
```

head 関数で最初のいくつかのデータを確認してみる

```
> head(dB)
```

```
  bit
1   3
2   4
3   5
4   4
5   4
6   5
```

summary 関数で要約を出力する。

数値なので、分布が表示される。各値がが何個かを表示するには, いったん factor ｀で項目変数にしておく。

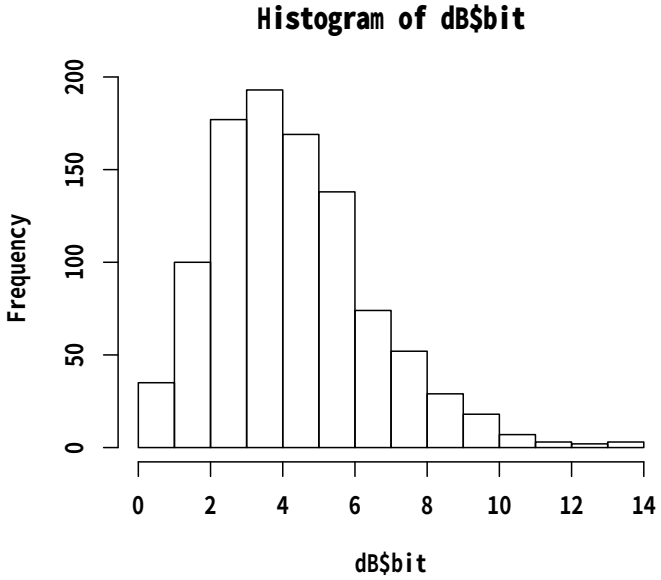
```
> summary(factor(dB$bit))
```

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
 35 100 177 193 169 138  74  52  29  18   7   3   2   1   2
```

ヒストグラムも描いておく。

引数 right=F は、階級が [0,1),[1,2),... ということであって、大きい方を階級の区分にに含まない。

```
> hist(dB$bit,right=F)
```



誰も買いを入れなかった（0）が 35 台、1 人だけ「買い」を入れたのが 100 台、2 人は 177 台、... と 14 人が買いを入れたののも 2 台ある。

どんな車が人気がある（「買い」がたくさん入った）か、それとも人気がないか（「買い」があまり入らないのか）を調べてみる。

データフレームをくっつけてひとつにする: data.frame 関数

データがが d01 と dB の 2 つに分かれているので、これをひとつのデータフレームにくっつけておく。

これを d03 に代入する。

```
> d03<-data.frame(d01,dB)
```

変数の要約を出力する: summary 関数

summary 関数で d03 を確認する。

```
> summary(d03)
```

```
nenshiki     kyori                    kizu          price        grade    selection
21:226   Min.   : 0.000   普通          :122   Min.   : 76.5   S:744   non:653
22:521   1st Qu.: 2.000   きれい        :306   1st Qu.:125.5   G:151   TS :274
23:172   Median : 3.600   すごくきれい:414   Median :144.5   L:105   TSG:  8
24: 81   Mean   : 4.643   修復歴あり    :158   Mean   :145.3           LED: 65
         3rd Qu.: 6.400                        3rd Qu.:162.0
         Max.   :22.500                        Max.   :256.0

    shaken            color          NV              SR              kawa
Min.   : 0.000   白       :372   Min.   :0.00   Min.   :0.000   Min.   :0.000
1st Qu.: 0.000   シルバー:261   1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.000
Median : 5.000   黒       :228   Median :1.00   Median :0.000   Median :0.000
Mean   : 7.014   青       : 71   Mean   :0.65   Mean   :0.032   Mean   :0.044
3rd Qu.:13.000   赤       : 24   3rd Qu.:1.00   3rd Qu.:0.000   3rd Qu.:0.000
Max.   :31.000   パープル: 15   Max.   :1.00   Max.   :1.000   Max.   :1.000
                 (Other) : 29
     bit
```

```
 Min.   : 0.000
 1st Qu.: 2.000
 Median : 3.000
 Mean   : 3.769
 3rd Qu.: 5.000
 Max.   :14.000
```

ポアソン回帰を行う： glm 関数の引数 family=poisson

被説明変数が連続数ではなく、0,1,2,3 といった回数の場合は、OLS で回帰するとうまく推定できない。そこで、ポアソン回帰 というのを使う。

OLS で、y に対して x_1, x_2 を回帰する場合、は以下を最小化する a, b_1, b_2 を求めた。

$$\sum_{i=1}^{N}(y_i - a - b_1 x_{1i} - b_2 x_{2i})^2$$

ポアソン回帰で、y に対して x_1, x_2 を回帰する場合、は以下を最<小大>化する a, b_1, b_2 を求める。

$$\prod_{i=1}^{N} \frac{\lambda_i^{y_i} e^{-\lambda_i}}{y_i!}$$

ただし、

$$\lambda_i = \exp(a + b_1 x_{1i} + b_2 x_{2i})$$

で、$\prod$ は積を表す。$\exp(x)$ は e^x のこと。

これがどういうことを意味するかは、ひとまず置いておいて、R による推定方法だけ説明しておく。

R では、ポアソン回帰も glm 関数で推定できる。その引数に `family=poisson` と入れておくだけ。あとは、OLS や二項ロジットと同じ。

formula=dealer . で、 dealer 以外のすべての変数を説明変数に入れる。

```
> o03<-glm(bit~.,d03,family=poisson)
> summary(o03)
```

```
Call:
glm(formula = bit ~ ., family = poisson, data = d03)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-3.06433  -0.82091  -0.09833   0.57556   2.73322

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.0078633  0.2288170   4.405 1.06e-05 ***
nenshiki22        0.0015677  0.0453454   0.035   0.9724
nenshiki23        0.0251593  0.0639916   0.393   0.6942
nenshiki24       -0.1576516  0.0999559  -1.577   0.1147
kyori            -0.0119682  0.0077444  -1.545   0.1222
kizu きれい      -0.0640199  0.0586747  -1.091   0.2752
kizu すごくきれい 0.0680287  0.0612852   1.110   0.2670
kizu 修復歴あり   0.0237917  0.0674101   0.353   0.7241
price             0.0011490  0.0015999   0.718   0.4726
gradeG           -0.1314628  0.0580107  -2.266   0.0234 *
gradeL           -0.0008772  0.0613011  -0.014   0.9886
selectionTS       0.0585235  0.0435317   1.344   0.1788
selectionTSG     -0.0582832  0.1918967  -0.304   0.7613
selectionLED     -0.1542458  0.0770964  -2.001   0.0454 *
shaken            0.0015228  0.0023859   0.638   0.5233
color 白          0.2185843  0.0484552   4.511 6.45e-06 ***
color 黒          0.2526146  0.0535268   4.719 2.37e-06 ***
color 青         -0.1120836  0.0804015  -1.394   0.1633
color 紺          0.0149275  0.1615542   0.092   0.9264
color 赤          0.8752234  0.0831351  10.528  < 2e-16 ***
```

```
color パープル      0.1575441  0.1433547   1.099   0.2718
color ワイン       -0.0646796  0.1781001  -0.363   0.7165
color 緑            0.1431526  0.2360339   0.606   0.5442
NV                  0.0234388  0.0358432   0.654   0.5132
SR                  0.0704467  0.0937765   0.751   0.4525
kawa                0.2001441  0.0952607   2.101   0.0356 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1355.6  on 999  degrees of freedom
Residual deviance: 1148.2  on 974  degrees of freedom
AIC: 4192.2

Number of Fisher Scoring iterations: 5
```

z 値は t 値と同じと考えて問題ない。

ステップワイズ法で説明変数の取捨選択: step 関数

引数に分析結果 o02 を入れ、 step 関数で説明変数を選択する。

```
> o03s<-step(o03)
```

```
Start:  AIC=4192.23
bit ~ nenshiki + kyori + kizu + price + grade + selection + shaken +
    color + NV + SR + kawa

            Df Deviance    AIC
- shaken     1   1148.6 4190.6
- NV         1   1148.6 4190.7
- price      1   1148.7 4190.7
- SR         1   1148.8 4190.8
- nenshiki   3   1153.2 4191.2
<none>           1148.2 4192.2
- kyori      1   1150.6 4192.6
- grade      2   1153.5 4193.6
- selection  3   1156.2 4194.2
- kawa       1   1152.6 4194.6
- kizu       3   1157.8 4195.8
- color      8   1267.0 4295.1

Step:  AIC=4190.64
bit ~ nenshiki + kyori + kizu + price + grade + selection + color +
    NV + SR + kawa

            Df Deviance    AIC
- NV         1   1149.1 4189.1
- SR         1   1149.2 4189.2
- nenshiki   3   1153.2 4189.2
- price      1   1149.3 4189.3
<none>           1148.6 4190.6
- kyori      1   1151.0 4191.0
- grade      2   1154.0 4192.0
- selection  3   1156.7 4192.7
- kawa       1   1152.9 4192.9
- kizu       3   1158.3 4194.3
- color      8   1268.5 4294.6

Step:  AIC=4189.12
bit ~ nenshiki + kyori + kizu + price + grade + selection + color +
    SR + kawa

            Df Deviance    AIC
- SR         1   1149.7 4187.7
- price      1   1149.9 4187.9
- nenshiki   3   1154.0 4188.0
<none>           1149.1 4189.1
- kyori      1   1151.4 4189.4
- grade      2   1154.3 4190.3
```

```
- selection  3   1157.2 4191.2
- kawa       1   1153.4 4191.4
- kizu       3   1158.6 4192.6
- color      8   1268.6 4292.6

Step:  AIC=4187.67
bit ~ nenshiki + kyori + kizu + price + grade + selection + color +
    kawa

            Df Deviance    AIC
- price      1   1150.8 4186.9
- nenshiki   3   1155.0 4187.1
<none>           1149.7 4187.7
- kyori      1   1151.7 4187.7
- grade      2   1154.6 4188.6
- selection  3   1157.3 4189.3
- kawa       1   1153.9 4189.9
- kizu       3   1158.8 4190.8
- color      8   1269.6 4291.6

Step:  AIC=4186.87
bit ~ nenshiki + kyori + kizu + grade + selection + color + kawa

            Df Deviance    AIC
- nenshiki   3   1155.0 4185.1
<none>           1150.8 4186.9
- grade      2   1155.5 4187.5
- selection  3   1159.0 4189.0
- kawa       1   1156.7 4190.7
- kizu       3   1162.2 4192.3
- kyori      1   1159.8 4193.9
- color      8   1280.3 4300.3

Step:  AIC=4185.07
bit ~ kyori + kizu + grade + selection + color + kawa

            Df Deviance    AIC
<none>           1155.0 4185.1
- grade      2   1159.5 4185.5
- selection  3   1162.1 4186.1
- kawa       1   1160.4 4188.4
- kizu       3   1166.3 4190.3
- kyori      1   1163.8 4191.9
- color      8   1285.3 4299.3
```

いろいろ試行錯誤をやった後に、次の結果が得られている。

```
> summary(o03s)
```

```
Call:
glm(formula = bit ~ kyori + kizu + grade + selection + color +
    kawa, family = poisson, data = d03)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0330  -0.8045  -0.1196   0.5943   2.8215

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.192006   0.069766  17.086  < 2e-16 ***
kyori            -0.015617   0.005313  -2.939  0.00329 **
kizu きれい      -0.055194   0.058083  -0.950  0.34198
kizu すごくきれい 0.083323   0.057637   1.446  0.14828
kizu 修復歴あり   0.011524   0.064901   0.178  0.85907
gradeG           -0.118661   0.056978  -2.083  0.03729 *
gradeL           -0.016325   0.059807  -0.273  0.78488
selectionTS       0.071268   0.039369   1.810  0.07025 .
selectionTSG     -0.096655   0.179792  -0.538  0.59086
selectionLED     -0.104344   0.071997  -1.449  0.14726
color 白          0.230924   0.045341   5.093 3.52e-07 ***
color 黒          0.262487   0.049928   5.257 1.46e-07 ***
```

```
color 青          -0.108329   0.080137  -1.352  0.17644
color 紺           0.023364   0.160980   0.145  0.88460
color 赤           0.887728   0.082435  10.769  < 2e-16 ***
color パープル     0.127046   0.141528   0.898  0.36936
color ワイン      -0.091792   0.177103  -0.518  0.60425
color 緑           0.143692   0.233869   0.614  0.53894
kawa               0.212914   0.091214   2.334  0.01958 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1355.6  on 999  degrees of freedom
Residual deviance: 1155.0  on 981  degrees of freedom
AIC: 4185.1

Number of Fisher Scoring iterations: 5
```

走行距離（ kyori ）と車体の傷（ kizu ）とグレード（ grade)、色 (color) などが残った。色は白、黒、赤の係数が大きく、p 値も小さい。

説明変数を二次関数にする

kyori を二次関数にしたい場合も、OLS と同様に二次の項を `I(kyori^2)` として加えればよい。

```
> o03_2<-glm(bit~.+I(kyori^2),d03,family=poisson)
```

ステップワイズ法も行う。

ステップワイズ法の経過の出力は、今回は、引数 `trace=0` で止める。

```
> o03s_2<-step(o03_2, trace=0)
> summary(o03s_2)
```

```
Call:
glm(formula = bit ~ kizu + grade + selection + color + kawa +
    I(kyori^2), family = poisson, data = d03)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0394  -0.8062  -0.1047   0.5951   2.8388

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.1597870  0.0646476  17.940  < 2e-16 ***
kizu きれい      -0.0655503  0.0584804  -1.121  0.26233
kizu すごくきれい 0.0810205  0.0577724   1.402  0.16079
kizu 修復歴あり   0.0068871  0.0651067   0.106  0.91576
gradeG           -0.1218701  0.0569472  -2.140  0.03235 *
gradeL           -0.0200167  0.0596892  -0.335  0.73736
selectionTS       0.0680282  0.0393059   1.731  0.08350 .
selectionTSG     -0.0826835  0.1795818  -0.460  0.64521
selectionLED     -0.1014498  0.0719372  -1.410  0.15846
color 白          0.2326537  0.0453123   5.134 2.83e-07 ***
color 黒          0.2641326  0.0498886   5.294 1.19e-07 ***
color 青         -0.1053440  0.0800663  -1.316  0.18827
color 紺          0.0238471  0.1609749   0.148  0.88223
color 赤          0.8916004  0.0823396  10.828  < 2e-16 ***
color パープル    0.1339664  0.1414587   0.947  0.34362
color ワイン     -0.0809078  0.1768033  -0.458  0.64723
color 緑          0.1481668  0.2338112   0.634  0.52627
kawa              0.2159693  0.0911794   2.369  0.01785 *
I(kyori^2)       -0.0010373  0.0003603  -2.879  0.00399 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)
```

```
    Null deviance: 1355.6  on 999  degrees of freedom
Residual deviance: 1155.2  on 981  degrees of freedom
AIC: 4185.2

Number of Fisher Scoring iterations: 5
```

連続変数を指定した区間に分割する: cut 関数

kyori を 0〜2、2〜5、5〜10、10〜の 4 区間に分けてみる。

cut 関数を使い、引数 breaks= （省略形 br= ）に区切りとなる区間を指定すればよい。

最小値（0）も含むように、区間を設定するには、引数 inclued.lowest=T をつければよい。

```
> kyori_b<-cut(d03$kyori,br=c(0,2,5,10,30),include.lowest=T)
> summary(kyori_b)
```

```
 [0,2]  (2,5]  (5,10] (10,30]
   260    381     272      87
```

これを kyori の代わりに使う。

データフレーム d03 の変数 kyori に kyori_b を上書きしたものを、 d03b として保存する。

```
> d03b<-d03 #d02 をまるごとコピー
> d03b$kyori<-kyori_b #kyori のデータを入れ替える
```

あとは、さきほどと同じ手順。

ポアソン回帰： glm 関数に引数 family=poisson

全ての変数を説明変数としたポアソン回帰。用いるデータフレームが d03b であることだけが異なる。

```
> o03_b<-glm(bit~.,family=poisson,d03b)
```

ステップワイズ法で説明変数の取捨選択： step 関数

AIC に基づいて説明変数を選ぶ。（引数 trace=0 で、経過は出力しない）

```
> o03s_b<-step(o03_b,trace=0)
> summary(o03s_b)
```

```
Call:
glm(formula = bit ~ kyori + kizu + grade + selection + color +
    kawa, family = poisson, data = d03b)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2083  -0.7987  -0.1185   0.6158   2.6351

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      1.04437    0.06971  14.981  < 2e-16 ***
kyori(2,5]       0.18615    0.04147   4.489 7.16e-06 ***
kyori(5,10]     -0.07623    0.04919  -1.550   0.1212
kyori(10,30]    -0.03308    0.07326  -0.452   0.6516
kizu きれい     -0.02503    0.05820  -0.430   0.6671
kizu すごくきれい  0.11807    0.05715   2.066   0.0388 *
kizu 修復歴あり   0.05013    0.06485   0.773   0.4395
gradeG          -0.12893    0.05701  -2.262   0.0237 *
gradeL          -0.01580    0.05978  -0.264   0.7915
selectionTS      0.06659    0.03954   1.684   0.0921 .
selectionTSG    -0.04058    0.18017  -0.225   0.8218
```

```
selectionLED      -0.10777    0.07200  -1.497   0.1344
color 白           0.22317    0.04532   4.925 8.45e-07 ***
color 黒           0.25366    0.04998   5.075 3.87e-07 ***
color 青          -0.11294    0.08017  -1.409   0.1589
color 紺           0.08670    0.16129   0.538   0.5909
color 赤           0.89147    0.08254  10.800  < 2e-16 ***
color パープル     0.16263    0.14155   1.149   0.2506
color ワイン      -0.06134    0.17711  -0.346   0.7291
color 緑           0.07439    0.23418   0.318   0.7507
kawa               0.20509    0.09132   2.246   0.0247 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1355.6  on 999  degrees of freedom
Residual deviance: 1117.4  on 979  degrees of freedom
AIC: 4151.4

Number of Fisher Scoring iterations: 5
```

AIC で計測モデルを比較: AIC 関数

kyori をそのまま (o03s)、2 次式 (o03s_2)、それと 0〜2 万 km、2〜5 万 km、... を切り分けた (o03s_b)、3 種類を説明変数としてポアソン回帰を行った。

どれがよいかは AIC で判断する。

```
> AIC(o03s)
```

```
[1] 4185.066
```

```
> AIC(o03s_2)
```

```
[1] 4185.18
```

```
> AIC(o03s_b)
```

```
[1] 4151.392
```

結局、2〜6 万 km を切り分けた o01s_b の当てはまりがちょっとよかった。

走行距離 2〜5 万 km、車体の色が白、黒、赤が人気があり、特に赤の人気は高いようだ。

社会統計学 B

3.3 順序ロジット分析

被説明変数が数値を区切った項目変数しか得られない場合の回帰

3.3.1 前回までの作業

```
> d01<-read.csv("prius.csv")
> d01$nenshiki<-factor(d01$nenshiki)
> d01$kizu<-factor(d01$kizu,
+                  levels=c("普通","きれい","すごくきれい","修復歴あり"))
> d01$grade<-factor(d01$grade,levels=c("S","G","L"))
> d01$selection<-factor(d01$selection,
+                       levels=c("non","TS","TSG","LED"))
> d01$color<-factor(d01$color,
+                   levels=c("シルバー","白","黒","青","紺",
+                            "赤","パープル","ワイン","緑"))
```

3.3.2 順序ロジット分析

被説明変数が、実際には連続変数なのに、得られた情報がそれを階級区分したものだけである場合、順序ロジット分析という方法を使って回帰分析を行う。

基本的な考え方

大学で教員が、レポートの成績をつける場合は、0〜100 点で採点して、成績としては、0〜59 点を不可、60〜69 点を可、70〜79 点を良、80 点以上を優と置き換えて、「不可」「可」「良」「優」という情報だけ報告すればよいことになっている。

しかし、どうせ、不可」「良」「優」で報告するのに、わざわざ 100 点満点で採点することもないので、適当な点数を付けて、「この点を超えたら可」という具合に評価しているかもしれない。

ある教員が、レポートとし課したことの何割やっているか（x_1）を評価して、全てやっていたら b_1 点、おもしろい（$x_2 = 1$)、おもしろくない（$x_2 = 0$）の配点を b_2 点としている場合、学生 i さんの採点は

$$y_i^* = b_1 x_{1i} + b_2 x_{2i}$$

となるはず。

ちなみに、定数項 a がない。相対評価なので、レポートを出した人全員にあげる点数は意味がない。

このとき、

$y_i^* < \zeta_1$ なら、、、、、、、y_i=”不可”

$\zeta_1 \leq y_i^* < \zeta_2$ なら、、、y_i =”可”

$\zeta_2 \leq y_i^* < \zeta_3$ なら、、、y_i =”良”

$\zeta_3 \leq y_i$ なら、、、、、、、y_i =”優”

で成績を出しているとする。

$i = 1, ...N$ の学生について、x_1, x_2 の評価と「不可」「可」「良」「優」の評価のデータが得られれば、この教員が、どのように a, b_1, b_2 に配点し、成績の区切り $\zeta_1, \zeta_2, \zeta_3$ をどこに定めているかを推定することができる。

こういう問題を扱うのが、順序ロジット。

プリウスの例

価格のデータが

～120 万円

120～140 万円

140～160 万円

160 万円～

の４区分でしか手に入らなかったと考えよう。

price のデータを、この４区分に変換して、順序ロジット用に使おう。ひとまず、もともとのデータ d01 は触らずに、これを d04 にまるごとコピーしていじる。

```
> d04<-d01
```

連続変数を区切る: cut 関数

連続変数を、指定した値で区切って、項目変数に変えるには cut 関数を使い、区切りは、引数 breaks=c(0,10,20,30) という具合に与える。

引数 breaks= は省略形 br= でもよい。

引数 br=c(0,10,20,30) の場合、値を (0,10],(10,20],(20,30]（つまり 0 より大きく 10 以下、10 より大きく 20 以下、20 より大きく 30 以下）に区分する。この範囲に入らないものは欠損値（ NA ）となる。

最大値と最小値を具体的に入れてもよいが、マイナス無限大（ -Inf ）と無限大（ Inf ）を使っても構わない

```
> d04$price<-cut(d04$price,br=c(-Inf,120,140,160,Inf))
```

どんなデータになったか、 head 関数で最初のいくつかを出力してみる。

```
> head(d04$price)
```

```
[1] (160, Inf] (120,140]  (120,140]  (140,160]  (160, Inf] (160, Inf]
Levels: (-Inf,120] (120,140] (140,160] (160, Inf]
```

値が、数値ではなく、項目名に変換されている。

最後に、 Levels： で、どの順序で並んでいるかも表示されている。

summary 関数で各項目にいくつずつ区分されたかを見てみる。

```
> summary(d04$price)
```

```
(-Inf,120]  (120,140]  (140,160] (160, Inf]
       204         232         300         264
```

データ準備ができた。順序ロジットをやってみよう。

ライブラリ MASS を読み込む： library 関数

順序ロジットは、ライブラリ MASS にある polr 関数を使う。しかし、このライブラリ MASS は、R を標準でインストールした場合、インストールはされているが、起動時に読み込んでくれていない。

だから、 polr 関数を使う前に、以下のように、一度ライブラリ MASS を読み込まなければならない。

```
> library(MASS)
```

順序ロジット分析： polr 関数

データフレーム d04 にある price 以外のすべての変数を説明変数として、順序ロジット分析を行うには以下のように指定する。

```
> o04<-polr(price~.,d04,method="logistic")
```

分析結果の要約を出力する： summary 関数

summary 関数で結果を出力するのは OLS と同じ。

```
> summary(o04)
```

```
Call:
polr(formula = price ~ ., data = d04, method = "logistic")

Coefficients:
                    Value Std. Error    t value
nenshiki22       1.010423    0.20546   4.917788
nenshiki23       3.795108    0.31689  11.976155
nenshiki24       5.913458    0.58836  10.050707
kyori           -0.798941    0.04564 -17.505513
kizu きれい       1.205600    0.26697   4.515899
kizu すごくきれい 2.512397    0.27920   8.998617
kizu 修復歴あり  -2.154929    0.31832  -6.769606
gradeG           0.320418    0.25676   1.247938
gradeL          -1.617258    0.28206  -5.733795
selectionTS      2.125941    0.21321   9.971293
selectionTSG    -2.865460    1.20881  -2.370479
selectionLED     2.028567    0.40628   4.993040
shaken           0.026081    0.01117   2.335298
color 白         1.877613    0.21530   8.721111
color 黒         2.050145    0.24220   8.464648
color 青        -0.424172    0.31181  -1.360367
color 紺        -0.165408    0.67538  -0.244909
color 赤         0.654533    0.51221   1.277865
color パープル  -0.891463    0.77591  -1.148924
color ワイン    -0.003593    0.77558  -0.004633
color 緑         1.386216    1.05083   1.319163
NV               0.737889    0.17126   4.308621
SR               2.721580    0.47351   5.747707
kawa             1.771785    0.51912   3.413048

Intercepts:
                      Value    Std. Error t value
(-Inf,120]|(120,140]  -2.8520   0.4140    -6.8882
(120,140]|(140,160]    0.8763   0.3765     2.3272
(140,160]|(160, Inf]   5.0379   0.4375    11.5149

Residual Deviance: 1161.44
AIC: 1215.44
```

モデルの後に、係数の推定値とそのバラツキが出力される（Coefficients:）。

OLS の場合と違って、ここに定数項 (Intercept) というのがない。

かわりに、その下に Intercepts: というのがあり、これが各項目の区分する値の推定値とそのバラツキである。

係数の推定値を出力する： coef 関数

順序ロジットの分析結果 o04 から係数の推定値だけを出力するには、 `coef(o04)` とするか `o04$coef` 。

```
> coef(o04)
```

```
     nenshiki22      nenshiki23      nenshiki24           kyori
    1.010422514     3.795108169     5.913457917    -0.798940901
     kizu きれい kizu すごくきれい   kizu 修復歴あり          gradeG
    1.205600454     2.512397165    -2.154929456     0.320418317
         gradeL      selectionTS     selectionTSG     selectionLED
   -1.617258441     2.125941461    -2.865460499     2.028566951
          shaken         color 白         color 黒         color 青
    0.026081100     1.877612720     2.050145429    -0.424172452
       color 紺         color 赤    color パープル      color ワイン
   -0.165407622     0.654533445    -0.891462817    -0.003593477
       color 緑              NV              SR            kawa
    1.386215534     0.737889281     2.721579773     1.771785490
```

区切りの推定値だけ出力するには o04$zeta 。

```
> o04$zeta
```

```
(-Inf,120]|(120,140]  (120,140]|(140,160] (140,160]|(160, Inf]
         -2.8519699            0.8762675            5.0378508
```

(-Inf,120] と (120,140] の区切りが-2.8520 で、 (120,140] と (140,160] との区切りが 0.8763、 (140,160] と (160, Inf] との区切りが 5.0379。

(-Inf,120] と (120,140] の区切りは 120 じゃないのか！ というツッコミもあるかもしれないが、被説明変数は項目変数であって、具体的な数値の情報は何一つない。 (-Inf,120] という名はついているが、これはそういう名前をつけたというだけのことであって、「 A 」でも「安いやつ」といった名前でも分析結果は変わらない。

順序ロジットで、推定した区切りは絶対的なものではなく、相対的なものである。

本当は 120,140,160 とその区切り位置は等しく 20 万円の差であるが、順序ロジットの区切り位置は、-2.8520、0.8763、5.0379 と、その差はほぼ等しく 4 前後。

つまり、実際の値の 5 分の 1 程度のスケールで、実際の推定値を再現していることになる。

区切りの推定値を 5 倍して、（最初の区切りが 120 ぐらいになるように）135 を加えてみる。

```
> o04$zeta*5+135
```

```
(-Inf,120]|(120,140]  (120,140]|(140,160] (140,160]|(160, Inf]
           120.7402              139.3813             160.1893
```

区切りがほぼ再現できている！！！

じつは、係数の推定値も、ほとんど OLS の結果の 5 倍前後となっている（さすがに t 値の小さいものはガタガタだが）。

OLS の推定値から切片を除いたものと、順序ロジットの推定値を 5 倍したものを並べて表示してみる。

ちなみに、 coef(o01)[-1] の [-1] は、 coef(o01) の１番目の要素を＜除く＞という意味。

```
> o01<-lm(price~.,d01)
> data.frame(coef(o01)[-1],coef(o04)*5)
```

```
                 coef.o01...1. coef.o04....5
nenshiki22           5.9515378    5.05211257
nenshiki23          18.9692196   18.97554084
nenshiki24          39.4474994   29.56728959
kyori               -3.2208603   -3.99470450
kizu きれい          4.1928162    6.02800227
kizu すごくきれい   12.4744222   12.56198583
kizu 修復歴あり    -10.8663251  -10.77464728
gradeG               1.6659023    1.60209159
gradeL              -7.8442331   -8.08629221
selectionTS         10.1304202   10.62970731
selectionTSG        30.3887444  -14.32730249
selectionLED         9.4021883   10.14283476
```

```
shaken                0.1818520    0.13040550
color 白              9.7176104    9.38806360
color 黒             10.5208771   10.25072714
color 青             -0.3884378   -2.12086226
color 紺             -2.3052652   -0.82703811
color 赤              1.8629030    3.27266722
color パープル       -1.7540063   -4.45731408
color ワイン          5.0850184   -0.01796738
color 緑              3.4146157    6.93107767
NV                    2.8602270    3.68944641
SR                   12.8986293   13.60789887
kawa                 16.3336887    8.85892745
```

OLS では被説明変数に千円単位の細かい数字を使い、順序ロジットでは４区分の項目しか使わなかったにもかかわらず、結構近い結果を出している。

同様に、標準誤差も比較してみよう。

OLS の結果 o01 を summary 関数で出力すると、 coefficients のところの Std. Error の列に標準誤差があるから、以下のように指定する。

```
> o01se<-summary(o01)$coefficients[,"Std. Error"]
```

順序ロジットの結果 o04 も同様。

```
> o04se<-summary(o04)$coefficients[,"Std. Error"]
```

それらをデータフレームにまとめて表示する。ただし、OLS の係数は、最初の要素として切片が含まれるので [-1] をつけてこれを抜く。

順序ロジットの方は、最後の３つは区切りの値なので、推定係数 27 個のうち最初の 24 個だけ [1:24] とするか最後の 3 個を抜く [-c(25:27)] 指定する必要がある。

順序ロジットの方は、係数がだいたい５分の１のスケールなので、標準誤差も 5 倍しておく。

```
> data.frame(o01se[-1],o04se[1:24]*5)
```

```
                  o01se..1. o04se.1.24....5
nenshiki22       0.87237076      1.02731395
nenshiki23       1.15411573      1.58444350
nenshiki24       1.56599362      2.94181180
kyori            0.10740186      0.22819694
kizu きれい      1.12410070      1.33483986
kizu すごくきれい 1.14251533      1.39599079
kizu 修復歴あり   1.27859043      1.59162096
gradeG           1.10421600      1.28379082
gradeL           1.14241087      1.41028626
selectionTS      0.82371012      1.06603099
selectionTSG     3.84053461      6.04405360
selectionLED     1.47646961      2.03139483
shaken           0.04807864      0.05584104
color 白         0.87571894      1.07647568
color 黒         0.99655192      1.21100456
color 青         1.40737758      1.55903643
color 紺         2.97174761      3.37691467
color 赤         2.24520739      2.56104332
color パープル   2.82599746      3.87955462
color ワイン     3.24709792      3.87789261
color 緑         4.75155598      5.25414716
NV               0.71015195      0.85629393
SR               1.92573376      2.36753530
kawa             1.92045201      2.59560575
```

ステップワイズ法による変数選択： step 関数

順序ロジットの結果は、やや特殊なものになるが、OLS 同様、 step 関数で、ステップワイズ法による変数選択が可能である。

いろいろな変数の組み合わせで AIC を比較して最適な変数選択を行ってくれる。

結果は o04s に代入する。

```
> o04s<-step(o04)
```

```
Start:  AIC=1215.44
price ~ nenshiki + kyori + kizu + grade + selection + shaken +
    color + NV + SR + kawa

            Df    AIC
<none>         1215.4
- shaken     1 1218.9
- kawa       1 1225.3
- NV         1 1232.3
- SR         1 1246.1
- grade      2 1249.8
- selection  3 1339.7
- color      8 1349.0
- nenshiki   3 1469.1
- kizu       3 1550.7
- kyori      1 1772.6
```

```
> summary(o04s)
```

```
Call:
polr(formula = price ~ nenshiki + kyori + kizu + grade + selection +
    shaken + color + NV + SR + kawa, data = d04, method = "logistic")

Coefficients:
                    Value Std. Error    t value
nenshiki22       1.010423    0.20546   4.917788
nenshiki23       3.795108    0.31689  11.976155
nenshiki24       5.913458    0.58836  10.050707
kyori           -0.798941    0.04564 -17.505513
kizu きれい       1.205600    0.26697   4.515899
kizu すごくきれい 2.512397    0.27920   8.998617
kizu 修復歴あり  -2.154929    0.31832  -6.769606
gradeG           0.320418    0.25676   1.247938
gradeL          -1.617258    0.28206  -5.733795
selectionTS      2.125941    0.21321   9.971293
selectionTSG    -2.865460    1.20881  -2.370479
selectionLED     2.028567    0.40628   4.993040
shaken           0.026081    0.01117   2.335298
color 白         1.877613    0.21530   8.721111
color 黒         2.050145    0.24220   8.464648
color 青        -0.424172    0.31181  -1.360367
color 紺        -0.165408    0.67538  -0.244909
color 赤         0.654533    0.51221   1.277865
color パープル  -0.891463    0.77591  -1.148924
color ワイン    -0.003593    0.77558  -0.004633
color 緑         1.386216    1.05083   1.319163
NV               0.737889    0.17126   4.308621
SR               2.721580    0.47351   5.747707
kawa             1.771785    0.51912   3.413048

Intercepts:
                     Value    Std. Error t value
(-Inf,120]|(120,140]  -2.8520   0.4140    -6.8882
(120,140]|(140,160]    0.8763   0.3765     2.3272
(140,160]|(160, Inf]   5.0379   0.4375    11.5149

Residual Deviance: 1161.44
```

```
AIC: 1215.44
```

すべての変数が残った。

説明変数に二次式を加える：引数 formula= に +I(x^2)

被説明変数に二次式を加えるのは、OLS と変わらない。

```
> o04_2<-polr(price~.+I(kyori^2),d04,method="logistic")
```

ステップワイズ法で変数を選択する

ステップワイズ法による変数選択： step 関数

再び変数を選択する。

```
> o04s_2<-step(o04_2)
```

```
Start:  AIC=1216.47
price ~ nenshiki + kyori + kizu + grade + selection + shaken +
    color + NV + SR + kawa + I(kyori^2)

             Df    AIC
- I(kyori^2)  1 1215.4
<none>          1216.5
- shaken      1 1220.0
- kawa        1 1226.7
- NV          1 1233.5
- SR          1 1247.0
- grade       2 1250.1
- kyori       1 1279.8
- selection   3 1340.8
- color       8 1351.0
- nenshiki    3 1464.8
- kizu        3 1552.7

Step:  AIC=1215.44
price ~ nenshiki + kyori + kizu + grade + selection + shaken +
    color + NV + SR + kawa

            Df    AIC
<none>         1215.4
- shaken     1 1218.9
- kawa       1 1225.3
- NV         1 1232.3
- SR         1 1246.1
- grade      2 1249.8
- selection  3 1339.7
- color      8 1349.0
- nenshiki   3 1469.1
- kizu       3 1550.7
- kyori      1 1772.6
```

```
> summary(o04s_2)
```

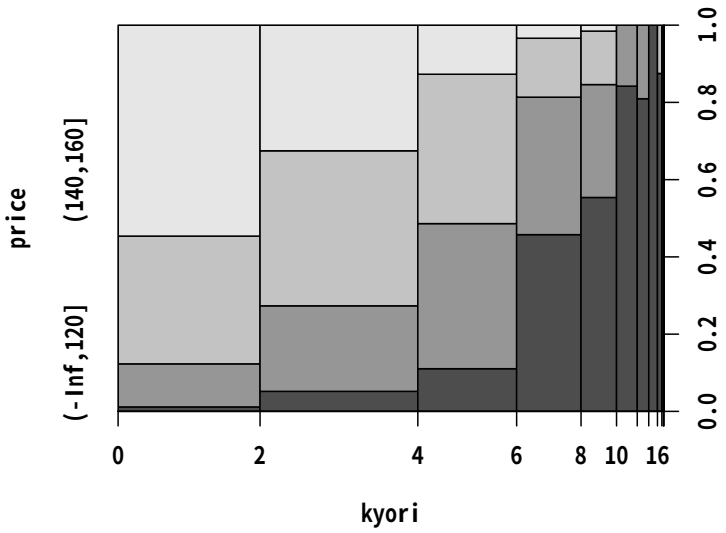
```
Call:
polr(formula = price ~ nenshiki + kyori + kizu + grade + selection +
    shaken + color + NV + SR + kawa, data = d04, method = "logistic")

Coefficients:
                    Value Std. Error    t value
nenshiki22       1.010423    0.20546   4.917788
nenshiki23       3.795108    0.31689  11.976155
nenshiki24       5.913458    0.58836  10.050707
kyori           -0.798941    0.04564 -17.505513
kizu きれい       1.205600    0.26697   4.515899
```

```
kizu すごくきれい   2.512397    0.27920   8.998617
kizu 修復歴あり    -2.154929    0.31832  -6.769606
gradeG              0.320418    0.25676   1.247938
gradeL             -1.617258    0.28206  -5.733795
selectionTS         2.125941    0.21321   9.971293
selectionTSG       -2.865460    1.20881  -2.370479
selectionLED        2.028567    0.40628   4.993040
shaken              0.026081    0.01117   2.335298
color 白            1.877613    0.21530   8.721111
color 黒            2.050145    0.24220   8.464648
color 青           -0.424172    0.31181  -1.360367
color 紺           -0.165408    0.67538  -0.244909
color 赤            0.654533    0.51221   1.277865
color パープル     -0.891463    0.77591  -1.148924
color ワイン       -0.003593    0.77558  -0.004633
color 緑            1.386216    1.05083   1.319163
NV                  0.737889    0.17126   4.308621
SR                  2.721580    0.47351   5.747707
kawa                1.771785    0.51912   3.413048

Intercepts:
                      Value    Std. Error t value
(-Inf,120]|(120,140]  -2.8520   0.4140    -6.8882
(120,140]|(140,160]    0.8763   0.3765     2.3272
(140,160]|(160, Inf]   5.0379   0.4375    11.5149

Residual Deviance: 1161.44
AIC: 1215.44
```

せっかく入れた I(kyori^2) だが、AIC による変数選択で落とされている。

被説明変数 kyori と kyori との関係を描いてみると以下のよう

```
> plot(price~kyori,d04)
```



さすがに２次の項まで識別できる情報は得られないか・・・

3.4 トービットモデル

被説明変数の値が途中で打ち切られている場合

3.4.1 前回までの作業

```
> d01<-read.csv("prius.csv")
> d01$nenshiki<-factor(d01$nenshiki)
> d01$kizu<-factor(d01$kizu,
+                  levels=c("普通","きれい","すごくきれい","修復歴あり"))
> d01$grade<-factor(d01$grade,
+                   levels=c("S","G","L"))
> d01$selection<-factor(d01$selection,
+                       levels=c("non","TS","TSG","LED"))
> d01$color<-factor(d01$color,
+                   levels=c("シルバー","白","黒","青","紺",
+                            "赤","パープル","ワイン","緑"))
```

3.4.2 トービットモデル

データというものは、すべて完璧にとってくるというものでもない。たとえば、市民マラソンの記録というのは最初の方は、ちゃんんととってくれるかもしれないが、一定時間以降は記録してくれない（失格？）

計測機器などは、大きすぎたり、小さすぎたら、計測できない場合があり、その場合もデータは途中で打ち切られる。

観察なども、いつまでも待ってられない、などの理由で、打ち切られたデータが発生します。そのような場合、打ち切られたデータは欠損地ではなく、最大値以上、といった情報を持っています。これを捨てるのはもったいない。

ただ、しかし、かといってそのまま計測すると、おかしなことになります。

そこで、使うのがトービットモデルという分析方法です。

打ち切りのあるデータをつくる

価格が 160 万円より大きな値はすべて 160 としか記録されていないと仮定して、これまで使ってきたデータフレーム d01 の変数 price の 160 より大きなものにすべて 160 を代入する。

ひとまず、 d01 をコピーして d05 という名前で保存する。

```
> d05<-d01
```

このうちの変数 price を 160 以上に 160 と入力する

```
> d05$price[d05$price>160]<-160
> summary(d05$price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   76.5   125.5   144.5   139.7   160.0   160.0
```

最大値が 160 となっている。

price と kyori で散布図を描いている。

```
> plot(price~kyori,d05,ylim=c(80,260))
```

とりあえず OLS で計測してみる： lm 関数

打ち切りのない被説明変数での OLS 推定は、以前やったとおり。

d01 の price を使った kyori で回帰。結果は o01_kyori に。推定係数は o01_kyoric に。

```
> o01_kyori<-lm(price~kyori,d01)
> o01_kyoric<-coef(o01_kyori)
```

打ち切りのある被説明変数で OLS 推定もやってみる。

d05 の price を使って同じように回帰する。結果は o05_kyori に。推定係数は o05_kyoric に。

```
> o05_kyori<-lm(price~kyori,d05)
> o05_kyoric<-coef(o05_kyori)
```

散布図を描いてみる： plot 関数の引数 ylim= と par(new=T)

切断された被説明変数で OLS 回帰をすると、本来の回帰とどのくらいずれてしまうのか、直感的に理解するために、散布図に回帰線を書き込んでみる。

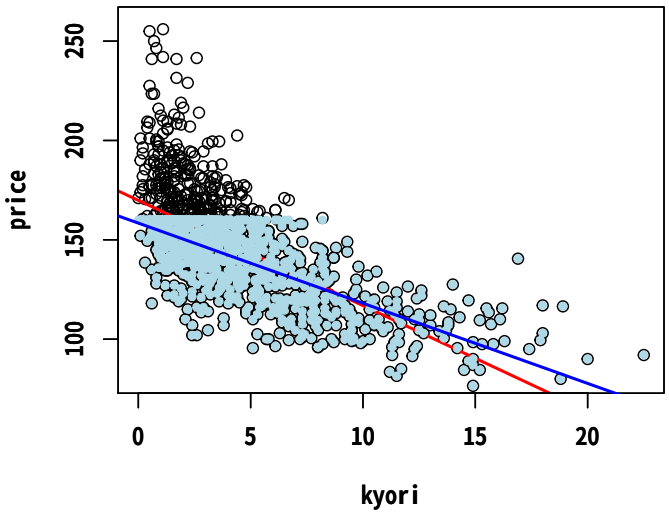
切断のない被説明変数の回帰線を赤で、切断のある回帰線を青で描く。

plot 関数で重ね描きする場合は、 curve 関数の引数 add=T のように気軽に描けない。 par(new=T) とすることで、同じ描画エリアに新しく書き込め、という命令をする。

par(new=T) だと新しく別のグラフを書き込んでいるので、軸の目盛を揃えてやる必要がある。x 軸は同じなので、そのままでよいが、y 軸は違うので ylim= で上限、下限を指定する。

lwd=2 という引数は、線の太さの指定。（そのままではちょっと見にくかったので）

```
> plot(price~kyori,d01,ylim=c(80,260))
> abline(o01_kyori,col="red",lwd=2)
> par(new=T)
> plot(price~kyori,d05,ylim=c(80,260),pch=20,col="lightblue")
> abline(o05_kyori,col="blue",lwd=2)
```



打ち切りのあるデータをそのまま OLS で回帰した場合（青線）は、全ての情報が利用できる場合（赤）もの

よりも傾きが小さくなってしまっている。

ひとまず、このような場合、OLS ではまずいということだけはわかった。

トービットモデルで計測する： vglm 関数の引数 tobit

R でトービットモデルは、vglm 関数を使うが、これはライブラリ VGAM が必要。これは、R には標準ではインストールされていないので、一回インストールしておかなければならない。

やり方は簡単で、メニューからパッケージを選択するだけ。 `install.packages("VGAM")` でも。

インストールしたら、 library 関数で登録する。

```
> library(VGAM)
```

これで準備 OK。あとは以下で回帰できる。

```
> o05t_kyori<-vglm(price~kyori,family=tobit(Upper=160),d05)
```

結果を summary 関数で表示する

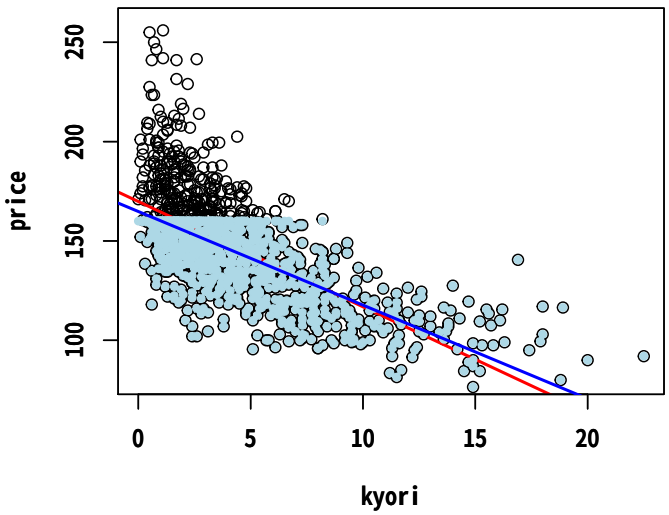
```
> summary(o05t_kyori)
```

```
Call:
vglm(formula = price ~ kyori, family = tobit(Upper = 160), data = d05)

Pearson residuals:
              Min       1Q    Median      3Q    Max
mu       -4.3619 -0.84858 -0.063766 0.98345 3.2290
log(sd) -1.1939 -0.67545 -0.137884 0.30342 7.9638

Coefficients:
              Estimate Std. Error z value
(Intercept):1 164.8381    0.96309  171.16
(Intercept):2   2.8468    0.02694  105.67
kyori          -4.7123    0.15771  -29.88

Number of linear predictors:  2

Names of linear predictors: mu, log(sd)

Dispersion Parameter for tobit family:   1

Log-likelihood: -3334.536 on 1997 degrees of freedom

Number of iterations: 5
```

散布図を描いてみる: plot 関数と abline 関数

打ち切りのないデータに対する OLS による回帰直線を赤線で、打ち切りのあるデータを使ったトービットモデルによる回帰直線を青線で描く。

```
> plot(price~kyori,d01,ylim=c(80,260))
> abline(o01_kyori,col="red",lwd=2)
> par(new=T)
> plot(price~kyori,d05,ylim=c(80,260),pch=20,col="lightblue")
> abline(coef(o05t_kyori)[c(1,3)],col="blue",lwd=2)
```



打ち切られたデータの部分を、いくらか補正していることがわかる。

全ての変数で回帰

全ての変数で回帰するには、OLS と同じように、 `formula=price~.` とすればよい。

ただし、 vglm では説明変数が多くなると、最適解を探すのがしんどくなり、以下のようなメッセージを出して、途中でギブアップする場合がある。

```
Error: NA/NaN/Inf in 'y'
```

そのような場合は、ひるまずに何度か繰り返しているうちに、ちゃんと解を出してくれる場合がある（そもそもデータに問題がある場合など、いくらがんばっても解がでない場合もあるけど）

```
> o05t<-vglm(price~.+I(kyori^2),family=tobit(Upper=160),d05)
```

summary 関数で結果を表示する。

```
> summary(o05t)
```

```
Call:
vglm(formula = price ~ . + I(kyori^2), family = tobit(Upper = 160),
    data = d05)

Coefficients:
                  Estimate Std. Error   z value
(Intercept):1    138.69401   1.575314  88.04214
(Intercept):2      2.19574   0.026306  83.46832
nenshiki22         5.40918   0.780670   6.92890
nenshiki23        18.43132   1.190277  15.48490
nenshiki24        27.16823   2.483600  10.93905
kyori             -5.41353   0.278994 -19.40378
kizu きれい        4.98170   1.032613   4.82436
kizu すごくきれい 11.76930   1.072878  10.96984
kizu 修復歴あり   -10.24412   1.156348  -8.85902
gradeG             1.56401   1.060711   1.47449
gradeL            -8.36275   1.023319  -8.17218
selectionTS        9.44787   0.830839  11.37148
selectionTSG     -16.72612   5.011725  -3.33740
selectionLED       9.76425   1.657538   5.89081
shaken             0.15905   0.045697   3.48042
```

```
color白             9.61037   0.809718  11.86879
color黒            11.06543   0.938774  11.78710
color青            -1.32916   1.251408  -1.06213
color紺            -1.92444   2.761223  -0.69695
color赤             1.49034   2.094851   0.71143
colorパープル      -2.19301   2.979131  -0.73612
colorワイン         0.37917   3.322845   0.11411
color緑             2.31028   4.530502   0.50994
NV                  3.41819   0.674824   5.06530
SR                  8.82419   1.948873   4.52784
kawa               11.15895   2.205056   5.06062
I(kyori^2)          0.14275   0.017024   8.38493

Number of linear predictors:  2

Names of linear predictors: mu, log(sd)

Dispersion Parameter for tobit family:   1

Log-likelihood: -2734.164 on 1973 degrees of freedom

Number of iterations: 7
```

係数の推定値を OLS の結果と比べる

トービットモデルの推定係数は、2 番目に OLS にはない係数が入るので、 [-2] をつけて、これを省く。

```
> o01<-lm(price~.+I(kyori^2),d01)
> data.frame(coef(o01),coef(o05t)[-2])
```

```
                  coef.o01. coef.o05t...2.
(Intercept)      136.6092148    138.6940068
nenshiki22         5.8859179      5.4091782
nenshiki23        18.0248308     18.4313211
nenshiki24        36.7806492     27.1682254
kyori             -5.3287495     -5.4135294
kizuきれい         5.8873734      4.9816961
kizuすごくきれい  13.4596901     11.7693004
kizu修復歴あり    -9.8256934    -10.2441196
gradeG             2.0220462      1.5640124
gradeL            -7.5985963     -8.3627494
selectionTS       10.4755843      9.4478731
selectionTSG      30.4417026    -16.7261154
selectionLED       9.1865209      9.7642454
shaken             0.1748549      0.1590450
color白            9.7578557      9.6103731
color黒           10.7111787     11.0654285
color青           -0.7097966     -1.3291581
color紺           -1.8800846     -1.9244449
color赤            1.4964321      1.4903375
colorパープル     -1.9350314     -2.1930087
colorワイン        4.4576833      0.3791733
color緑            3.4953352      2.3102757
NV                 2.9949909      3.4181875
SR                12.6953657      8.8241894
kawa              15.9354542     11.1589520
I(kyori^2)         0.1407539      0.1427457
```

だいたい同じ（推定が不安定な係数はやや食い違っているが・・・）

係数の標準誤差を OLS の結果と比べる

標準誤差も比べてみよう。

OLS の標準誤差を o01se に代入。

```
> o01se<-summary(o01)$coef[,"Std. Error"]
```

トービットモデルの標準誤差を o05tse に代入。

vglm の出力結果は、 lm や glm と要素の取り出し方が違う（後者は S3 クラスというが、前者は S4 クラスという新しい定義）

```
> o05tse<-summary(o05t)@coef3[,"Std. Error"]
```

これをデータフレームでくっつけて表示。ただし、o05tse の 2 番目の要素は、さきほどと同様に省く。

```
> data.frame(o01se,o05tse[-2])
```

```
                    o01se o05tse..2.
(Intercept)      1.68248649 1.57531398
nenshiki22       0.84609570 0.78066960
nenshiki23       1.12564766 1.19027735
nenshiki24       1.55570986 2.48359956
kyori            0.28606242 0.27899361
kizu きれい       1.11103152 1.03261325
kizu すごくきれい 1.11502650 1.07287799
kizu 修復歴あり   1.24697698 1.15634843
gradeG           1.07185215 1.06071098
gradeL           1.10838410 1.02331912
selectionTS      0.80005264 0.83083900
selectionTSG     3.72468803 5.01172525
selectionLED     1.43219024 1.65753819
shaken           0.04663670 0.04569713
color 白          0.84931745 0.80971773
color 黒          0.96678944 0.93877441
color 青          1.36552720 1.25140791
color 紺          2.88260360 2.76122294
color 赤          2.17797163 2.09485136
color パープル    2.74084464 2.97913112
color ワイン      3.15014485 3.32284531
color 緑          4.60823303 4.53050248
NV               0.68894028 0.67482411
SR               1.86781926 1.94887313
kawa             1.86320016 2.20505578
I(kyori^2)       0.01779044 0.01702409
```

標準誤差も（この場合は）ほとんど変わらない

第 4 章

分布の特徴をつかむ

4.1　度数分布から確率分布

プリウスの価格の「分布」をどう表現するか？

4.1.1　データの分布の様子を知りたい

生データ

1000 件のプリウスのオークション価格を集めた。

```
> d01<-read.csv("prius.csv")
> attach(d01)
```

とりあえず、最初から 100 件だけ表示してみる。

```
> price[1:100]
```

```
  [1] 241.0 124.5 128.5 146.5 160.5 171.0 149.0 104.5 174.0 153.5 205.5 127.0
 [13]  85.0 138.5 159.0 149.0 110.0 103.0 169.0 112.5 124.5 143.5 130.5 158.0
 [25] 132.0 146.0 104.5 114.5 130.0 114.5 116.0 136.0 142.0 148.0 137.5 192.0
 [37] 156.5 156.5 121.0 175.0 122.5 192.5 131.0 219.0 170.0 172.5 151.0 148.0
 [49] 145.5 155.5 132.5 111.0 138.5 139.0 116.0 141.5 141.0 121.5 118.5 138.5
 [61] 156.5 141.5  97.5 157.5 147.5 114.0  90.0 135.5 131.5  95.0 145.5 165.0
 [73] 138.0 116.0 143.0 159.5 190.0 108.5 128.0 154.5 108.5 139.0 118.0 145.0
 [85] 155.0 100.0 145.0 126.0 136.5 139.0 160.0 148.5 184.0 149.0 167.0  91.5
 [97] 134.0 117.5 153.5 184.0
```

これだけ見ていても仕方がない。

データの要約を確認する: summary 関数

R の要約を見てみよう

```
> summary(price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   76.5   125.5   144.5   145.3   162.0   256.0
```

左から、最小値、第１四分位数、中央値、平均、第３四分位数、最大値。

これで、そのようなデータのバラツキかをある程度知ることができる。

平均値と分散（標準偏差）： mean 関数と var 関数（ sd 関数)、 lenght 関数

データを知るために平均値と分散を示す場合もある。

平均値は、 mean 関数で求めることができる。

```
> mean(price)
```

```
[1] 145.2555
```

分散は、 var 関数で計算できる。

```
> var(price)
```

```
[1] 805.4955
```

分散の平方根は、標準偏差と呼ばれる。
平方根は sqrt 関数で。

```
> sqrt(var(price))
```

```
[1] 28.38125
```

sd 関数を使うこともできる。

```
> sd(price)
```

```
[1] 28.38125
```

同じ結果が出る。
平均と分散（または標準偏差）を示す場合は、同時にいくつのデータであるかを示した方がよい。
データがいくつあるかは、 length 関数で求める。

```
> length(price)
```

```
[1] 1000
```

これらは、以下のようにまとめることができる。

データ数	平均	分散（標準偏差）
1000	145.3	805.5 (28.4)

これで十分か？
答えは、十分ではない場合もあるし、十分な場合もある、だ。
この数字で何がわかるか、今回は、その意味を考えてみる。

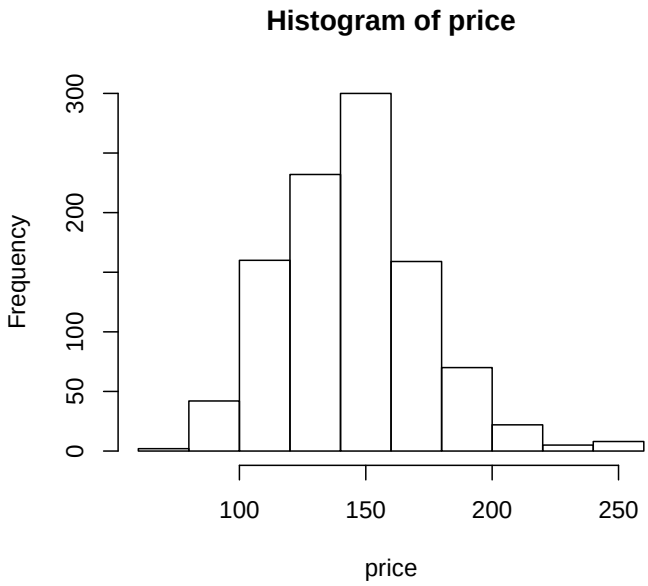
4.1.2　度数分布とヒストグラム

ヒストグラムを描く: hist 関数

データの分布を確認するには、度数分布や、それをグラフに表したヒストグラムを利用するのは、ひとつの方法だ。

R でヒストグラムは、 hist 関数で簡単に描ける。（あとで、中身を使うので、一旦 g01 として保存しておく。）

```
> g01<-hist(price)
```



この関数は、グラフだけでなく、度数分布も取り出せる。

階級の区切りは $breaks に入っている。

```
> g01$breaks
```

```
[1]  60  80 100 120 140 160 180 200 220 240 260
```

60 から 260 までを 20 間隔で区切って階級が設定されている。

度数は $count に入っている。

```
> g01$count
```

```
[1]   2  42 160 232 300 159  70  22   5   8
```

各階級にいくつ入っているかが、数値としてわかる。

ヒストグラムの階級の幅を変える: hist 関数の引数 br=

ヒストグラムの階級の幅は、引数 br= で、階級の区切りを指定することができる。

たとえば、何も指定しなかった場合は、R が勝手に 60〜260 までを 20 ごとに区切っていたが、これを 10 ごとに区切るように指示してみる。

seq 関数は、規則正しい数列を発生させるのに便利な関数。いろいろな指定が可能だが、60 から 260 まで、10 ごとに数列を発生させてくれ、というのは以下のように指定する。

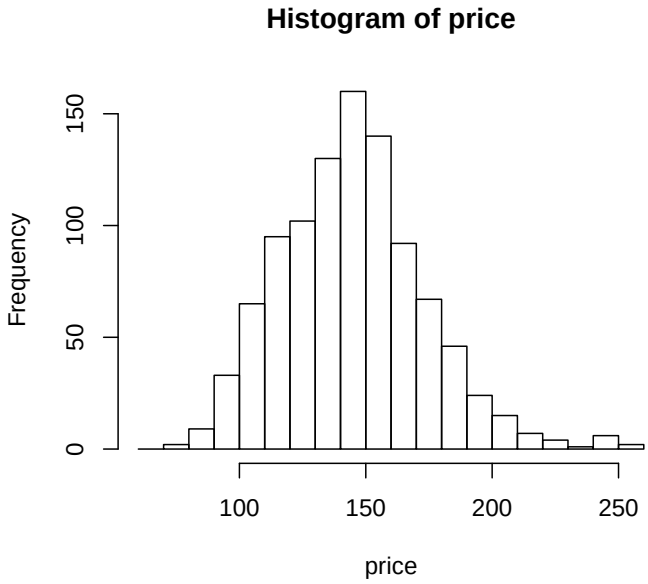
```
> seq(from=60,to=260,by=10)
```

```
 [1]  60  70  80  90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240
[20] 250 260
```

これを hist 関数の br= に与える。

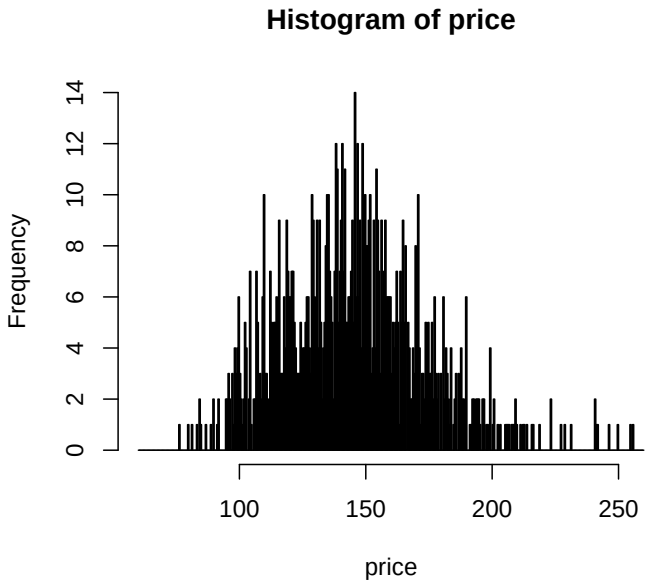
階級の幅が小さいと、ヒストグラムの様子がちょっと変わってくる。

```
> hist(price,br=seq(60,260,by=10))
```



さらに小さく、階級を 0.5 ごとに区切ると、だいぶ違った分布となる。

```
> hist(price,br=seq(60,260,by=.5))
```



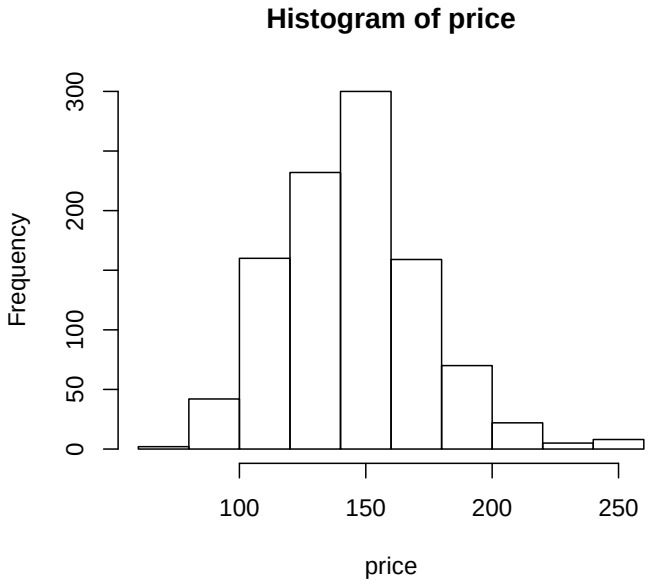
ヒストグラムは、分布の様子を描いてくれるが、階級の幅をどのくらいにするかで様子が変わってしまうので、やや扱いに困る。

最適な階級の幅についての主張があり、たとえば、R で何も指定せずに hist 関数を使うと Sturges が主張した方法でヒストグラムを描く。

他にも、Scott の方法とか、Freedam-diaconis という方法で、適切と思われる階級の幅を指定できる。

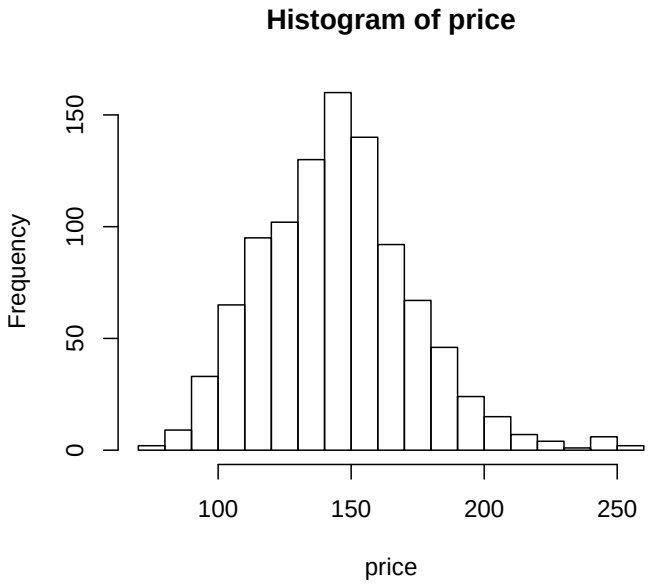
Sturges の方法

```
> hist(price,br="sturges")
```



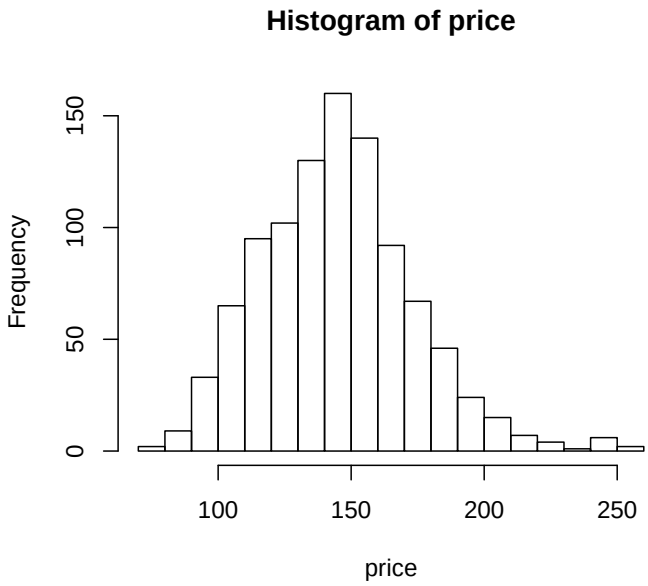
Scott の方法

```
> hist(price,br="scott")
```



Freedam-diaconis の方法

```
> hist(price,br="fd")
```



しかし、どれがいいというわけではなく、見え方の問題。

階級の幅が大きすぎれば、おおざっぱな情報しか得られないし、階級の幅がちいさ過ぎれば分布の様子がよくわからない。

price のデータは数値データとはいえ、.5 単位で区切られたデータなので、階級を細かくしても分布の形がそれほど大きく崩れることはないが、もっと精度の高いデータをヒストグラムに描くと、ややこしいグラフになる。

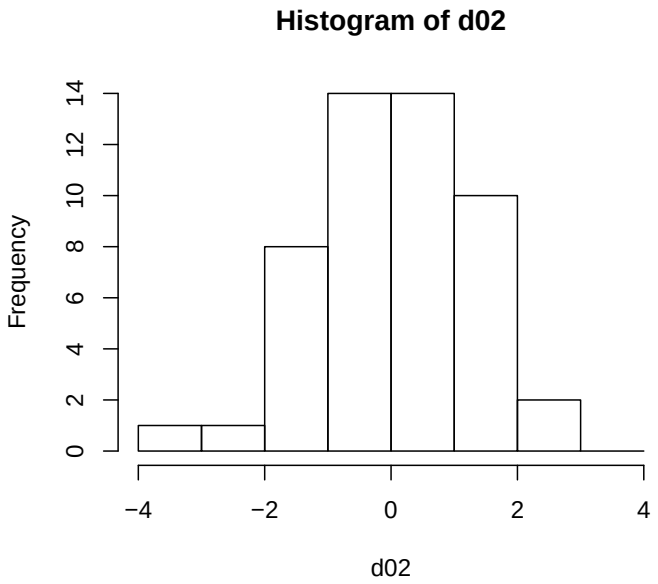
正規乱数 50 個をつくってみる（その意味はまだ説明してないけど、一種の乱数とだけ理解してください）

```
> (d02<-rnorm(50))
```

```
 [1]  1.001567729  1.497887292 -0.678921672 -0.281457677 -0.670076063
 [6]  0.592590886 -1.458126806  0.148314028  1.274842738  0.092536203
[11] -1.137488380  1.105972893  0.192156029  1.345284453 -1.538054158
[16] -1.082382480  1.221633802  0.833031644  0.814863344 -0.102597798
[21] -0.002667149  1.271591636 -0.649352031 -0.643086351  0.651114814
[26] -2.393951738  0.952214556  0.116220600 -0.283034432 -1.206937192
[31] -1.303645316 -0.058976923 -0.511384457 -0.970693460  2.021942663
[36]  0.318875680 -0.555930647 -1.810657631  1.702348435 -1.899906527
[41]  0.361179853  1.699374338  0.236242539  1.487980120  2.587914855
[46] -0.554310308 -3.394387859 -0.462637115  0.754042149  0.353653036
```

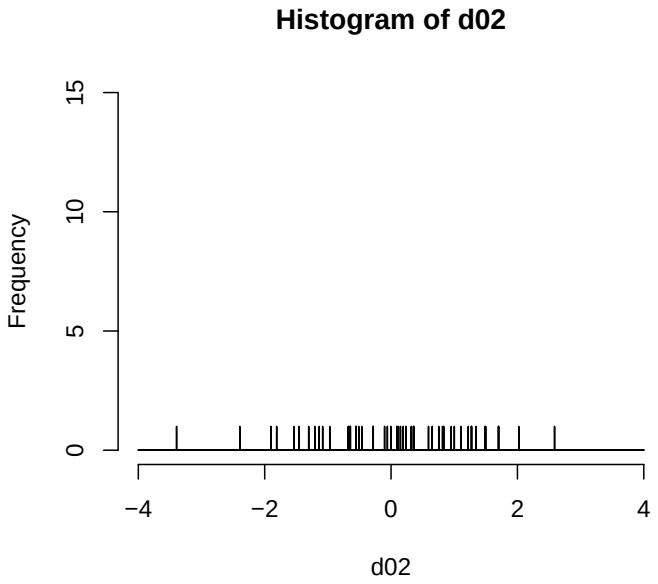
これを１単位でヒストグラムに描く。

```
> hist(d02,br=seq(-4,4,by=1))
```



これを 0.001 単位でヒスとグラムに描く。

```
> hist(d02,br=seq(-4,4,by=.001),ylim=c(0,15))
```



あたり前だが、すべての階級で度数が１となって、まったく様子が違う分布となる。

4.1.3 累積分布

累積度数を求める

累積度数というのは、度数を階級の小さい方から足しあわせて、最後は総数になるようにしたもの。

階級	度数	累積度数
$[0, 10)$	2	2
$[10, 20)$	4	6
$[20, 30)$	3	9
$[30, 40)$	1	10

累積度数を求める関数を探したが見つからないので、適当に累積度数を求める関数 f01 をつくった。
関数の中身は、（そんなに難しいことはしてないけど、）ひとまず理解できなくてもよい。

br01 に seq 関数で、階級の区切り点を与え、
x01 に br01 でした階級にデータ x を分類
den01 は、x01 の度数分布表
crm01 は、x01 の累積度数
crm01 を表示

```
> f01<-function(x,b01=1,fr01=round(min(x)-1,0),to01=round(max(x)+1,0)){
+       br01<-seq(from=fr01,to=to01,by=b01)
+       x01<-cut(x,br=br01)
+       den01<-table(x01)
+       crm01<-cumsum(den01)
+       crm01
+ }
```

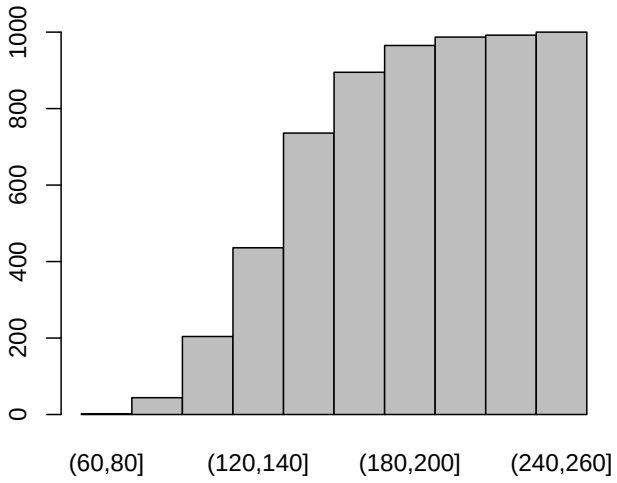
関数の使い方は、引数にもとのデータ（ベクトル）と、階級の幅を b01= で指定する（何も指定しなければ 1 となる）。階級の最小値はデータの最小値よりも小さい整数値、階級の最大値はデータの最大値よりも大きい整数値にしてしてある。自分で指定したい場合は、最小値を引数 fr01= で、最大値を to01= で与える。
階級が 20 の累積度数を求めてみる。

```
> f01(price,b01=20,fr01=60,to01=260)
```

```
  (60,80]  (80,100] (100,120] (120,140] (140,160] (160,180] (180,200] (200,220]
        2        44       204       436       736       895       965       987
(220,240] (240,260]
      992      1000
```

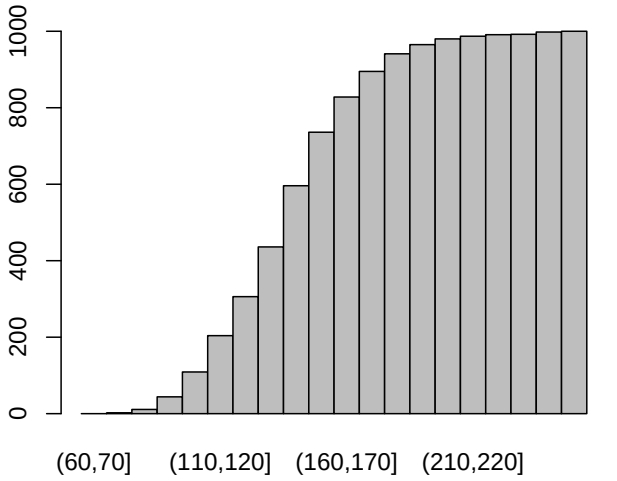
グラフに描く
階級の幅を変えて、累積度数の様子を確認してみる。
階級の幅 **20**

```
> barplot(f01(price,20,60,260),space=0)
```



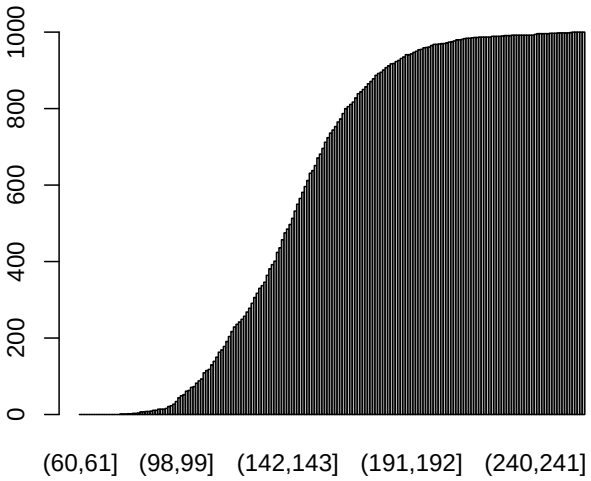
階級の幅 **10**

```
> barplot(f01(price,10,60,260),space=0)
```



階級の幅 **1**

```
> barplot(f01(price,1,60,260),space=0)
```



累積度数分布は、階級の幅に左右されない、、、というより、階級が小さいほど、きれいに分布の様子が確認できる。

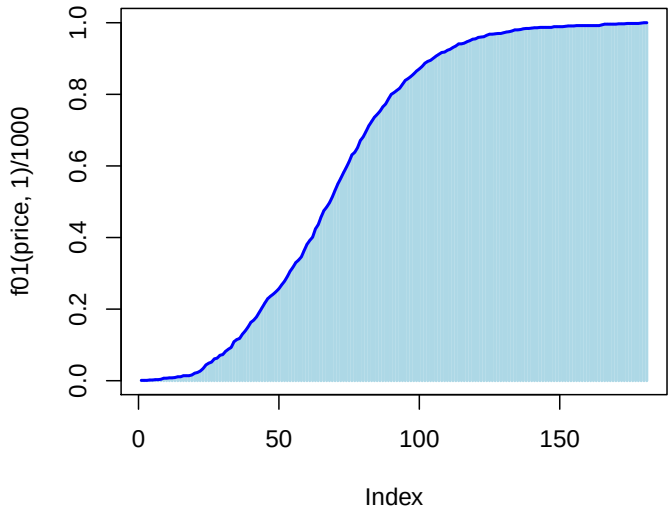
（累積）確率分布： plot 関数と ecdf 関数

これは、「度数」ではなく、度数を総数で割った「相対度数」でも同じこと。

以下は、階級の幅を小さくとった相対累積度数の分布。

軸のスケールが変わるだけで、同じ形が描ける。

```
> plot(f01(price,1)/1000,type="h",col="lightblue",lwd=2)
> lines(f01(price,1)/1000,col="blue",lwd=2)
```



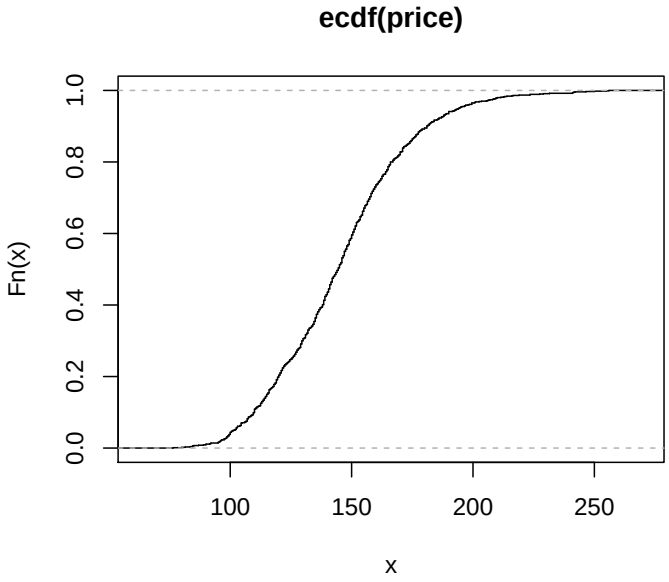
（ただし、このグラフの x 軸のラベルは階級の値ではないので無視してください。）

線だけなら、 ecdf で相対累積度数の分布を求め、これを plot 関数を使ってグラフに描くことができる。

引数の verticals=T は、階段状のグラフの垂直部分にも線を引くように指定する。

引数の do.p=F は、階級ごとに丸が打たれるのを止めて、線だけ描く。

```
> plot(ecdf(price),verticals=T,do.p=F)
```



0 から１まで増える滑らかな曲線となっている。

それでもちょっとだけデコボコしているけれど、さらにたくさんのデータがあれば、こうしたデコボコもなくなり、より滑らかな曲線となるはず。それを「(累積) 確率分布」と呼ぶ。

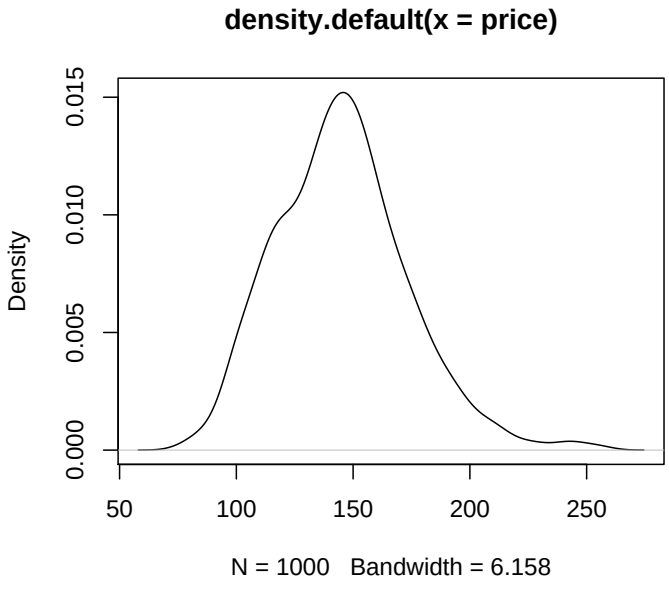
確率密度： density 関数

確率分布の傾きを描くこともできる。

確率分布の傾きは「確率密度」と呼ばれる。

これは density 関数で求め、 plot 関数を使ってグラフに描くことができる。

```
> plot(density(price))
```

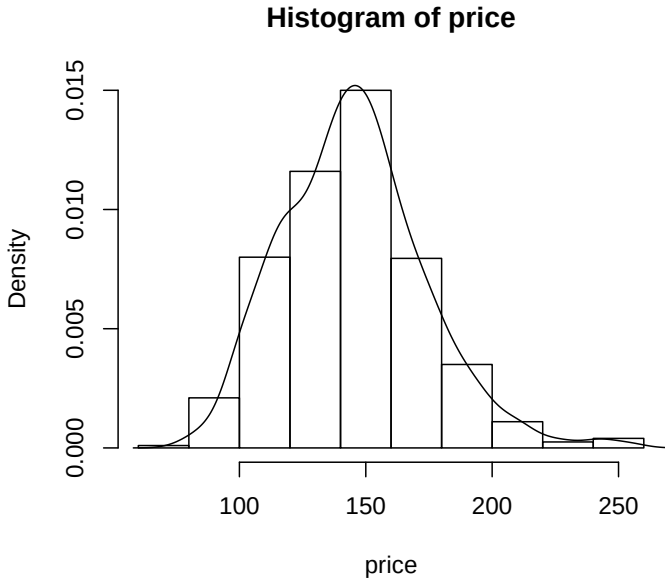


度数分布に重ねあわせてみる。

hist 関数の引数に prob=T とあるのは、度数のスケールを確率密度に合わせてある。

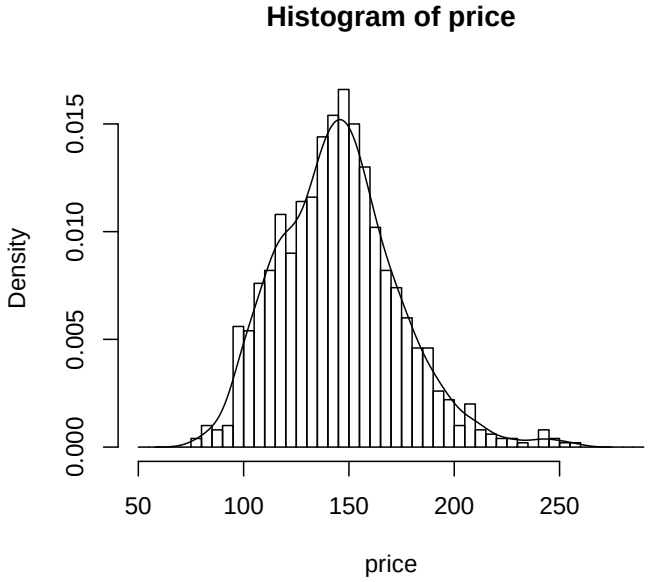
lines 関数を使うと、一旦描いたヒストグラムに重ね描きできる。

```
> hist(price,prob=T)
> lines(density(price))
```



ヒストグラムの階級の幅を、もう少し小さくしてみる。

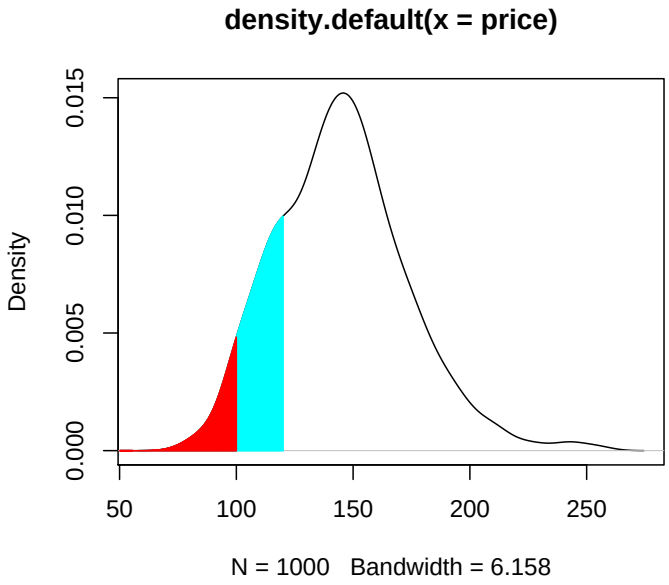
```
> hist(price,br=seq(50,290,by=5),prob=T)
> lines(density(price))
```



確率密度は（累積）確率分布の傾き（微分したもの）であるから、確率密度を小さい方から積分した面積が累積分布の値となる。

たとえば、確率密度のを小さい方から 100 まで積分した面積は、累積確率分布の 100 の値となる。

```
> plot(density(price))
> lines(density(price,from=50,to=120),type="h",col=5)
> lines(density(price,from=50,to=100),type="h",col=2)
```



赤い部分の面積は、価格が 100 万円以下の車の割合。青い部分の面積は 100〜120 万円の車の割合を表す。
これを計算してみると、赤い面積と青い面積を表すと、赤が 0.038（3.8 %）、青い面積が 0.198（19.8 %）

```
> (y1<-length(price[price<100])/1000)
```

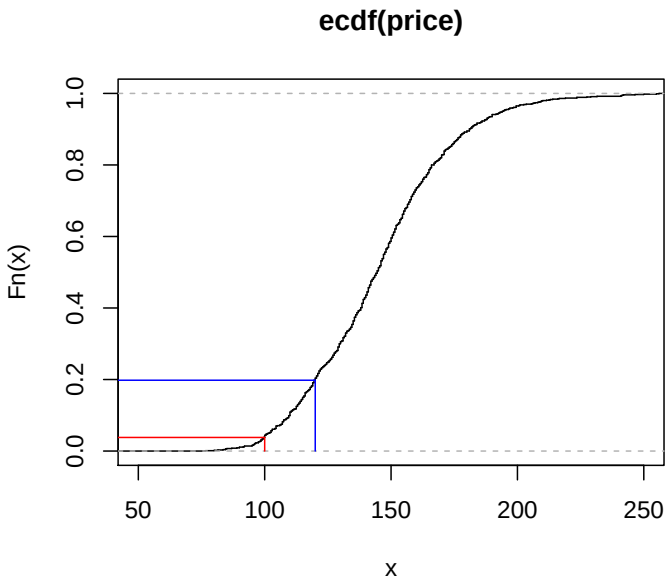
```
[1] 0.038
```

```
> (y2<-length(price[price<120])/1000)
```

```
[1] 0.198
```

これと対応する累積分布の位置を表示してみる。
累積分布は ecdf 関数を使って以下のように描く。
segments(x0,y0,x1,y1) で (x0,y0),(x1,y1) を通る線分が描ける。

```
> plot(ecdf(price),verticals=T,do.p=F,xlim=c(50,250))
> segments(100,0,100,y1,col="red")
> segments(0,y1,100,y1,col="red")
> segments(120,0,120,y2,col="blue")
> segments(0,y2,120,y2,col="blue")
```

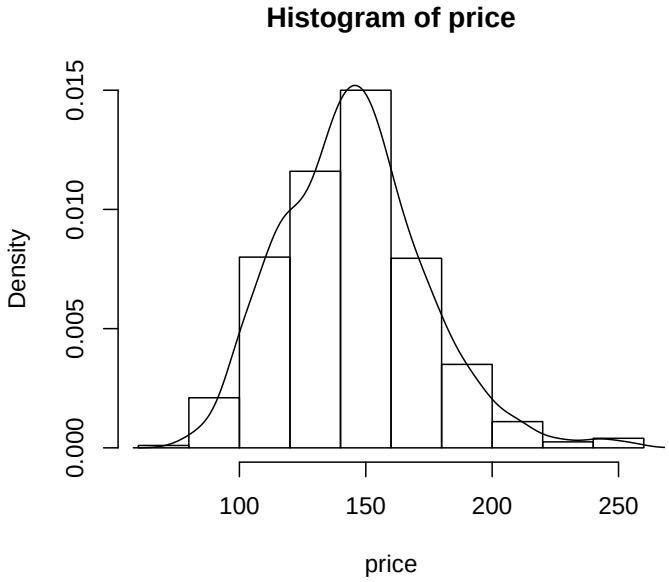


補足：R で確率密度を描く

確率密度のグラフを描くのに、 plot 関数、 lines 関数、 curve 関数の３種類を使った。

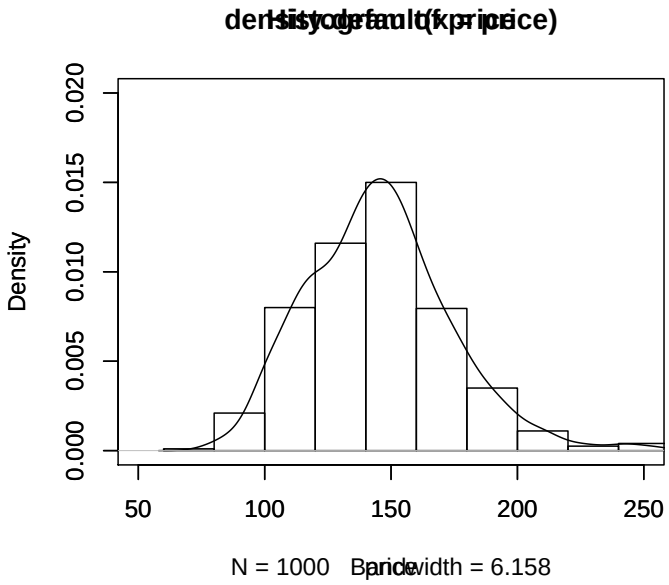
density 関数を出力するのに、 plot 関数と lines 関数の両方が使えて、同じような図を描くが、 plot 関数は単独で描けるのに対して、lines 関数は、すでに描いてあるグラフに重ね書きするだけで、単独では出力できない。基本的には次のような使い方をする。

```
> hist(price,prob=T)
> lines(density(price))
```



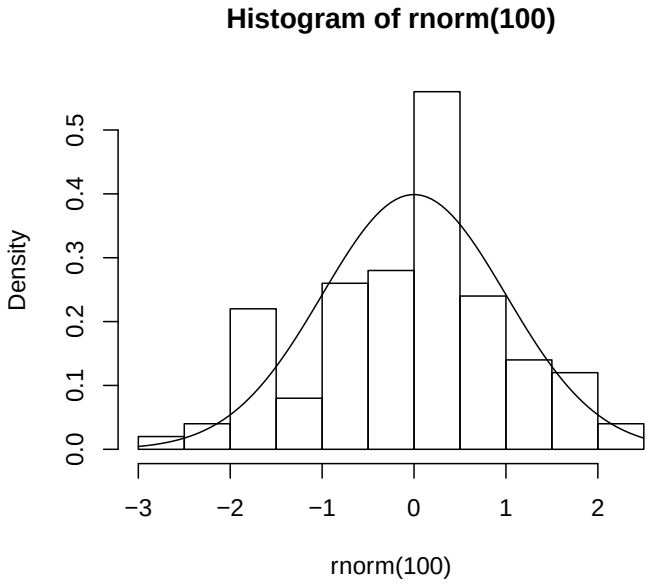
一方、 lines 関数は ヒストグラムや他のグラフに重ね描きできるが、 plot 関数は基本的に重ね描きできない。 plot 関数は、軸やらタイトルやら、改めて描き直してしまう。どうしても plot 関数で重ね描きしたい場合は、 par(new=T) として、前に描いたグラフを残しておく必要がある。その場合でも、引数 xlim= や ylim= で、軸のスケールを合わせる必要がある。そうしないと軸の目盛がズレてしまう。

```
> hist(price,prob=T,xlim=c(50,250),ylim=c(0,.02))
> par(new=T)
> plot(density(price),xlim=c(50,250),ylim=c(0,.02))
```



curve 関数は、引数に関数を与えてグラフを描くもので、単独でも描けるし、引数 add=T を与えて、重ね描きすることもできる。

```
> hist(rnorm(100),prob=T)
> curve(dnorm,add=T)
```



4.2　分布関数

関数形が知られた確率分布について。

確率分布は、一般に、ひとつの関数で書き表せるとは、全く 限らない。しかし、なんらかの関数形で書き表せるものもある。それをいくつか紹介しよう。

4.2.1　正規分布

身長の分布、平均値の分布、計測誤差の分布などは、正規分布に近いと言われている。

（累積）確率分布を表す関数を「（累積）分布関数」と言い、確率密度を表す関数を「密度関数」、分布する変数そのもの x を「確率変数」と呼ぶ。

平均値 μ、標準偏差 σ の正規分布の累積分布関数は、

$$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x} \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) du$$

正規分布の密度関数は、

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

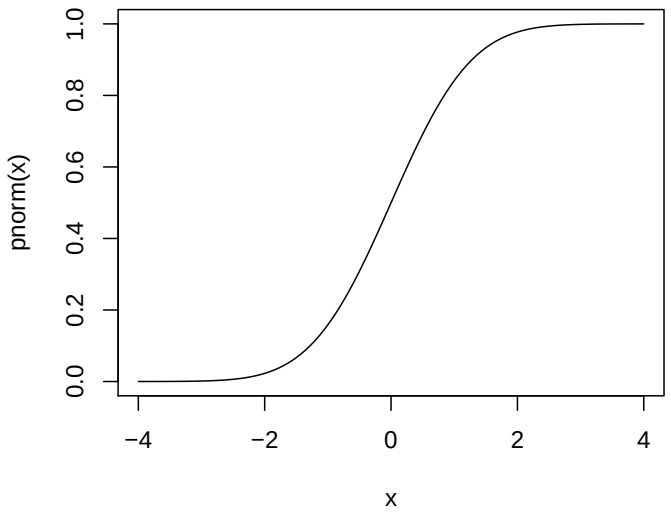
ずいぶん難しそうな数式だが、ひとまずここは、こんな感じの関数で表されていると思ってもらえればそれでよい。（ちょっとだけ余裕があれば、密度関数をマイナス無限大から x まで積分したのが累積分布関数となっていることぐらいは確認してほしいけど。）

（累積) 分布関数: pnorm 関数

これをグラフに描くのは簡単。

平均値 0、標準偏差 1 の正規分布（標準正規分布と呼ぶ）の分布関数は pnorm で描くことができる。

```
> curve(pnorm,-4,4)
```



標準正規分布で-2 以下の確率は、0.02275

```
> pnorm(-2,mean=0,sd=1)
```

```
[1] 0.02275013
```

標準正規分布の場合、引数 mean=0 と sd=1 は省略できる。

```
> pnorm(-2)
```

```
[1] 0.02275013
```

標準正規分布で 0 以下となる確率は 0.5

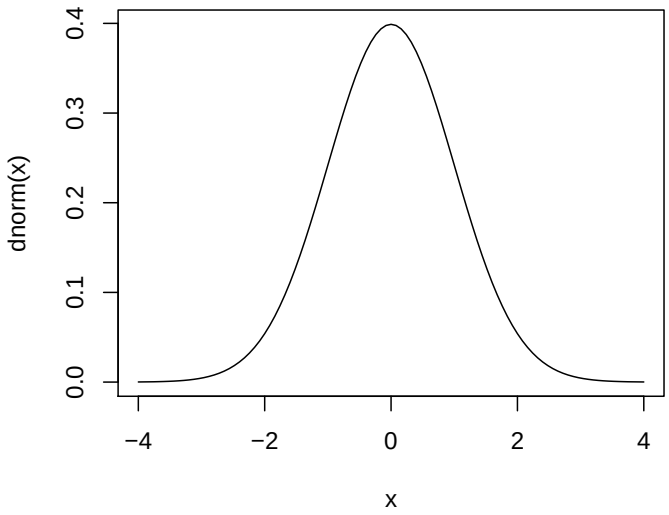
```
> pnorm(0)
```

```
[1] 0.5
```

密度関数： dnorm 関数

標準正規分布の密度関数は、 dnorm 関数で求める。グラフに描くと次のとおり

```
> curve(dnorm,-4,4)
```



標準正規分布で-2 の確率密度、0.05399

```
> dnorm(-2,mean=0,sd=1)
```

```
[1] 0.05399097
```

標準正規分布の場合、引数 mean=0 と sd=1 は省略できる。

```
> dnorm(-2)
```

```
[1] 0.05399097
```

標準正規分布で 0 の確率密度は 0.3989

```
> dnorm(0)
```

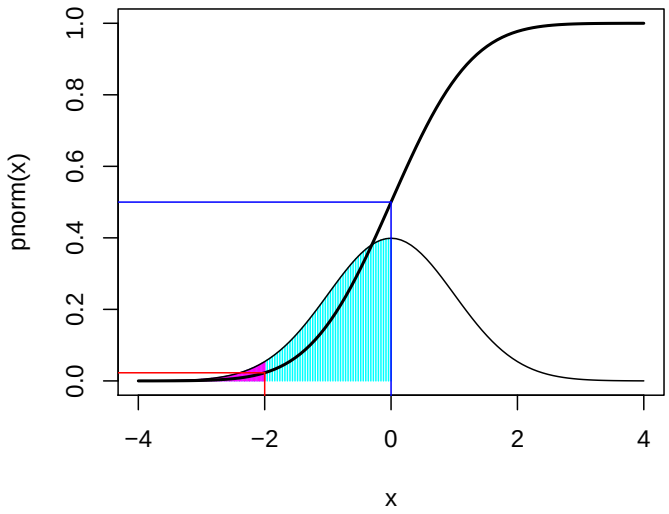
```
[1] 0.3989423
```

分布関数と密度関数の対応

注意して欲しいのは、たとえば、標準正規分布で-2 の確率密度が 0.05399 といっても、決して-2 が出る確率が 0.05399 というわけではないということ。あくまでも、累積分布関数の-2 のところでの傾き。

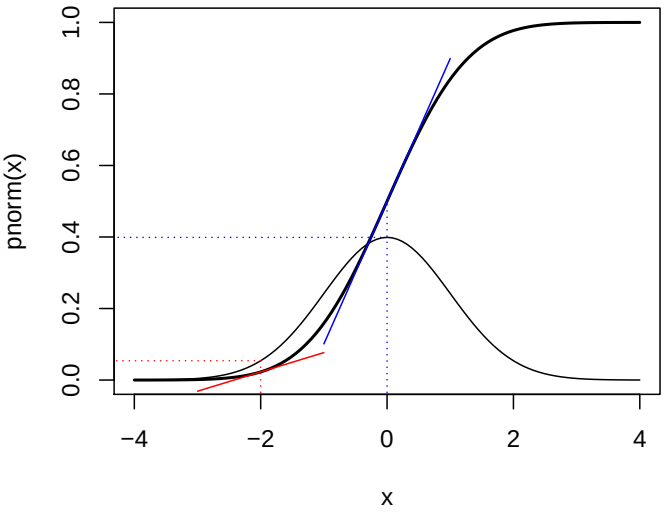
累積分布関数と密度関数を重ね描きしてみる。

```
> curve(pnorm,-4,4)
> curve(dnorm,xlim=c(-4,0),add=T,type="h",col=5)
> curve(dnorm,xlim=c(-4,-2),add=T,type="h",col=6)
> curve(dnorm,add=T)
> curve(pnorm,add=T,lwd=2)
> segments(-2,-1,-2,pnorm(-2),col=2)
> segments(-2,pnorm(-2),-5,pnorm(-2),col=2)
> segments(0,-1,0,pnorm(0),col=4)
> segments(0,pnorm(0),-5,pnorm(0),col=4)
```



密度関数の赤いところの面積が、累積分布関数の赤い線分が指す値（0.02275）。密度関数の青いところの面積が、累積分布関数の青い線分が指す値（0.5）

```
> curve(pnorm,-4,4,lwd=2)
> curve(dnorm,add=T)
> segments(-3,pnorm(-2)-dnorm(-2),-1,pnorm(-2)+dnorm(-2),col=2)
> segments(-2,-1,-2,dnorm(-2),col=2,lty="dotted")
> segments(-2,dnorm(-2),-5,dnorm(-2),col=2,lty="dotted")
> segments(-1,pnorm(0)-dnorm(0),1,pnorm(0)+dnorm(0),col=4)
> segments(0,-1,0,pnorm(0),col=4,lty="dotted")
> segments(0,dnorm(0),-5,dnorm(0),col=4,lty="dotted")
```



横軸が-2 のところ累積分布関数の傾きが、同じ位置の密度関数の傾き（0.05399)。横軸が 0 のところの累積分布関数の傾きが、同じ位置の密度関数の傾き（0.3989）

4.2.2 一様分布

どこかに偏りがあるわけではない、ある範囲で一定の確率で事象が起こりうるような確率分布。
宝くじで、いつ買うか、どこで買うかなどは一様分布？
$[0,1]$ の分布関数は、最小値 0、最大値 1 の場合、

$$F(x) = x, \text{ for } 0 \leq x \leq 1$$

密度関数は、

$$f(x) = 1, \text{ for } 0 \leq x \leq 1$$

分布関数： punif 関数

R で一様分布は punif 関数を用いる。$[a,b]$ の範囲の一様分布は `punif(x,min=a,max=b)`
$[0,10]$ の範囲の一様分布で、3 以下となる確率は 0.3

```
> runif(3,0,10)
```

```
[1] 5.915376 9.423451 7.903060
```

$[0,1]$ の範囲の一様分布関数は、引数 min=a,max=b を省略できる。
$[0,1]$ の範囲の一様分布で 0.5 以下となる確率は

```
> punif(0.5)
```

```
[1] 0.5
```

密度関数

範囲 $[a,b]$ の密度関数は、 `dunif(x,min=a,max=b)` で求める。
[0,10] の範囲で一様分布の 3 における傾きは以下で求める・・・といってもどの範囲でも同じはずだが・・・

```
> dunif(3,0,10)
```

```
[1] 0.1
```

$[0,1]$ の範囲の 0.5 における傾きは以下。

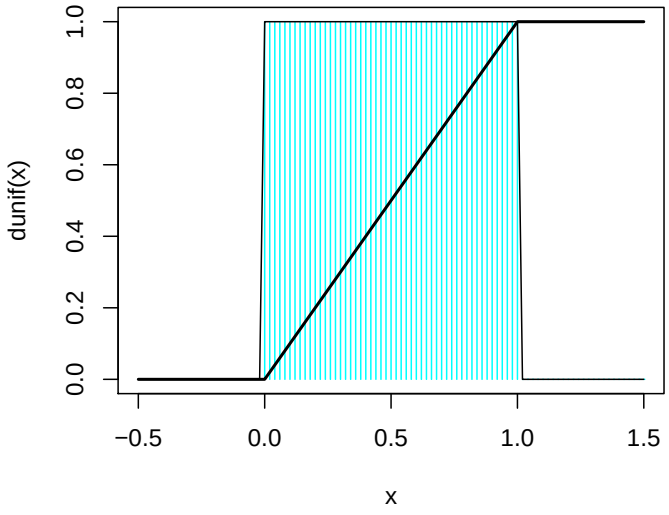
```
> dunif(.5)
```

```
[1] 1
```

比較

$[0,1]$ の範囲の一様分布をグラフで比較する。

```
> curve(dunif,-.5,1.5,type="h",col=5)
> curve(dunif,add=T)
> curve(punif,add=T,lwd=2)
```



4.2.3　指数関数

平均して θ 時間の寿命の蛍光灯が、x 時間で切れる確率や、丸太渡り競争なんかで、平均到達距離が θ である場合に、x の距離で落っこちる確率。

$\theta = 1/\lambda$ とするならば、$x \geq 0$ の範囲で・・・

分布関数は、

$$F(x) = 1 - e^{-\lambda x}$$

密度関数は、

$$f(x) = \lambda e^{-\lambda x}$$

この関数がどうやって導かれるかはともかく、分布関数の微分が密度関数になっていることぐらいは確認してほしい。

分布関数： pexp 関数

たとえば、丸太渡り競争で平均記録が 20m であるような場合は、$\lambda = 1/20$ となる。

このとき 10m までに落っこちる確率は 0.3935。すでに平均記録の半分の地点で、100 人中 39 人ぐらいは落っこちる。

```
> pexp(10,rate=1/20)
```

```
[1] 0.3934693
```

平均記録が 10m であるような場合は、$\lambda = 1/10$ で、この時 10m までに落っこちる確率は、0.632 にアップする。

```
> pexp(10,rate=1/10)
```

```
[1] 0.6321206
```

当然と言えば当然だが、0.5 よりも大きくて、100 人中 63 人以上が平均記録に達する前に落っこちる。

$\lambda = 1/10$ のとき、20m 地点までに落っこちる確率は 0.865 にアップして、残っているのは 100 人中 13〜14 人ぐらい。

```
> pexp(20,rate=1/10)
```

```
[1] 0.8646647
```

密度関数： dexp 関数

この場合の密度関数は dexp 関数で求めることができる。

たとえば、$\lambda = 1/10$ のとき、10m 地点での確率密度は 0.03679。

```
> dexp(10,rate=1/10)
```

```
[1] 0.03678794
```

10m 地点から次の 1m で、落っこちる人が 100 人中 4 人弱増える

同じ $\lambda = 1/10$ で、2m 地点だと確率密度は 0.08187 で、次の 1m 落っこちる人が 100 人中 8 人増える。

```
> dexp(2,rate=1/10)
```

```
[1] 0.08187308
```

地点によって落っこちる人の割合が減ってくように見えるが、どの地点でも、次の 1m で落っこちる人の、残った人に占める割合は同じ。

例えば、10m 地点で残った人の割合は $1 - pexp(10, 1/10)$ で求めることができる。このとき、次の 1m で落っこちる人の割合は $dexp(10, 1/10)$ だから・・・10m 地点で残った人の、次の 1m うちどの割合の人が落っこちるかは次で計算できる。

```
> dexp(10,1/10)/(1-pexp(10,1/10))
```

```
[1] 0.1
```

つまり、残った人の 10 %が次の 1m で落っこちる。

2m、5m、20m 地点でも同じように計算したら・・・

```
> dexp(2,1/10)/(1-pexp(2,1/10))
```

```
[1] 0.1
```

```
> dexp(5,1/10)/(1-pexp(5,1/10))
```

```
[1] 0.1
```

```
> dexp(20,1/10)/(1-pexp(20,1/10))
```

```
[1] 0.1
```

どの地点でも 0.1。残った人の 10 %が次の 1m で落っこちる。

この 0.1 というのは $\lambda = 1/10$ と一致する！！！

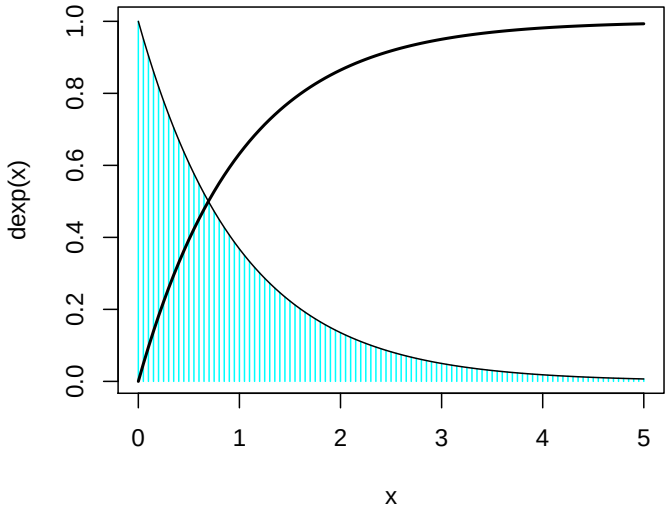
とびっくりするまでもなく、そもそも指数分布というのが、そのように定義して導いた分布だからなのだけど。

分布関数と密度関数をグラフに描く

指数分布の分布関数と密度関数をグラフに描いておく。

$\lambda = 1$ なら、 pexp 関数も dexp 関数も引数 rate= が不要なので、これで描く。($\lambda = 1/10$ なら、グラフの横軸のメモリを 10 倍すればよいだけ)[*1]

```
> curve(dexp,0,5,type="h",col=5)
> curve(dexp,add=T)
> curve(pexp,add=T,lwd=2)
```



[*1] どうしても $\lambda = 1$ 以外が描きたければ・・・(たとえば $\lambda = 1/10$)

```
> f01p<-function(x) pexp(x,rate=1/10)
> f01d<-function(x) dexp(x,rate=1/10)
```

という具体に新たに関数を定義して、 pexp の部分を f01p 、 dexp の部分を f01d に書き換えればよい。

その他の関数

R には、ほかにもいろいろな分布関数が用意されている。

分布	分布関数	密度関数
t 分布	pt	dt
カイ二乗分布	pchisq	dchisq
F 分布	pf	df
ガンマ分布	pgamma	dgamma

などなど。

パターンとして、分布名の頭に p をつけると（累積）分布関数、 d をつけると密度関数。

用意されていない確率分布でも、すぐ作れる場合もある。

4.2.4 第一種極値分布（ガンベル分布）

丸太渡り競争で指数分布を説明したが、その競争で最も距離が出た人の記録の分布。

ただし、参加人数 n が多くなると、記録の最大値も大きくなりやすいので、そのぶんは差し引いてある。[*2]

分布関数は、

$$F(x) = \exp(-\exp(-\lambda x))$$

ただし、λ は、平均到達距離の逆数（$\lambda = 1/\theta$）で、平均 10m だったら、$1/10$。

指数関数の中に指数関数が入っている二重指数関数の一種である。

密度関数は、その微分。

$$f(x) = \lambda \exp(-\lambda x) \exp(-exp(-\lambda x))$$

分布関数

R には第一種極値分布を求める関数は容易されていないが、関数形が簡単なので、自分でつくればよい。

たとえば平均到達距離が 10m だったら、$\lambda = 1/10$ で、

```
> lamb<-1/10
> f02c<-function(x) exp(-exp(-lamb*x))
```

100 人ぐらいの参加者がいたら、最高記録は期待値で 50m 程度となるはずだが、実際にやってみた場合に、それ以下となる確率は次で計算される。

```
> f02c(0)
```

```
[1] 0.3678794
```

最高記録が期待値よりも 20m 以上記録が伸びる確率は、次でわかる。

[*2] 具体的には、参加人数が $n-1$ 人から n 人に増えると、θ/n だけ増えてしまう（ただし、$theta = 1/\lambda$）。平均記録が 10m の場合に、参加者が 1 人の場合の最高記録の期待値は 10m。2 人だったら 10+10/2=15m、3 人だったら、10+10/2+10/3=18.3m、・・・100 人だったら、51.8m。

これは参加者 n が大きくなると $\theta \ln(n) + \gamma$ に近づく（ただし、$\gamma = 0.57721$ ：オイラー定数）ので、$theta \ln(n)$ だけ、到達距離 x から差し引いて、分布を求めている。

```
> 1-f02c(20)
```

```
[1] 0.126577
```

密度関数

密度関数も自分でつくればよい。

```
> f02d<-function(x) lamb*exp(-lamb*x)*exp(-exp(-lamb*x))
```

たとえば、平均到達距離が 10m の場合、最高記録の平均値から 1m 記録が伸びる確率は、以下。

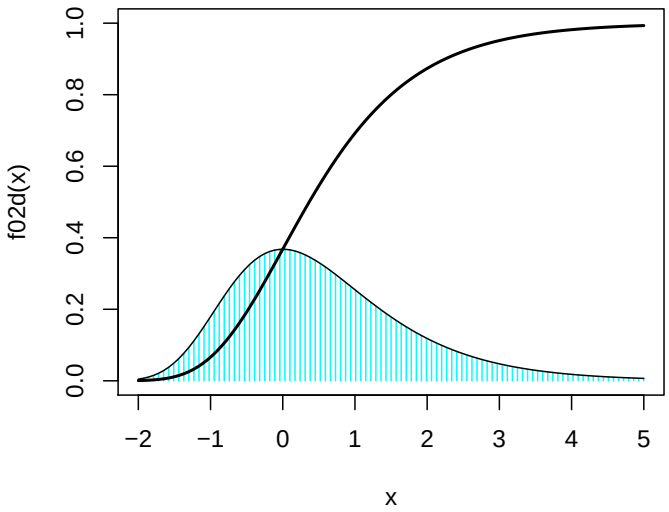
```
> f02d(0)
```

```
[1] 0.03678794
```

比較

平均到達距離が 1 だと、$\lambda = 1/1$ で、これをグラフに描く。

```
> lamb=1/1
> curve(f02d,-2,5,type="h",col=5,ylim=c(0,1))
> curve(f02d,add=T)
> curve(f02c,lwd=2,add=T)
```



4.3 標本抽出

標本の取り方をマスターする。

4.3.1 準備

プリウスのデータを使うので・・・

```
> d01<-read.csv("prius.csv")
> attach(d01)
```

4.3.2 母集団からの標本抽出

すべてのデータを使わなくても、一部のデータでもとのデータの分布を推し測りたい。

もとのデータが「母集団」、その分布を推し測るために抽出された一部のデータが「標本」。

母集団のデータは得られないのが普通だから、通常は、標本を使って議論する。中古車プリウスの価格だって、1000 個もあるが、もっともっとたくさん取引されているもののわずかでしかない。中古車プリウスの取引データすべてが「母集団」で、得られたデータが「標本」である。「母集団」には、取引される可能性も含めて考えることもできるので、実際に取引されたデータすべてを収集しても、それでもそれは「標本」という考え方もある。「確かに、これまでの取引データではそうかもしれないが、もう少し取引数が多くなると、そうした傾向は偶然ということがわかります」なんて言い方もされる。

標本は母集団そのものではないが、これが全体のデータの特徴をある程度代表していたりする。そのことを確かめるために、統計学があり、多くの定理や分析手法が存在する。そのほとんどが数式を使った厳密な定義と証明に基づいて築かれている。

そうした統計学ももちろん勉強してほしいのだが、本講では、数式を使わない直感的な理解を目指している。その方法は、まず、人為的に母集団のデータを発生させて、そこから標本をとる。そして、特定の統計的処理が、うまく母集団の情報を引き出せているかどうかを確認してみる。この方法は、直感的な理解を助けてくれる。

R を使うと、そうした試行が手軽にできる。授業でも、この後、こうした説明を多用する。ここで、その基本をマスターしよう。

母集団からの標本抽出： sample 関数

母集団のデータがあって、そこから無作為に標本を抽出したい。

中古プリウスの価格データ prius には 1000 個のデータがあった。ひとまずこれを「母集団」と考えよう。

この中から、無作為 に 3 個のデータを抽出する。

母集団からの無作為抽出は、 sample 関数を使う。

```
> sample(price,3)
```

```
[1]  96.5 162.0 136.0
```

もう一回やってみる。

```
> sample(price,3)
```

```
[1] 134.5 157.5 169.0
```

今度は違うデータが抽出された。

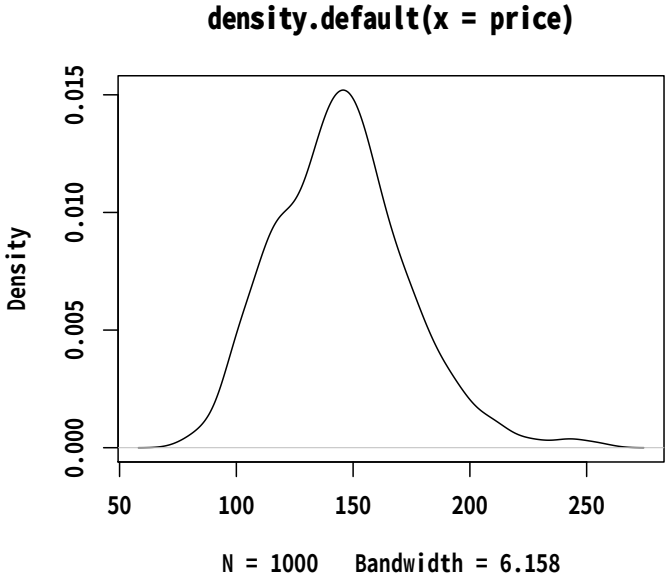
sample 関数は、 sample(x,size=a) で、ベクトル x から無作為に a 個を抽出する。

これが本当に 無作為 に選ばれたものかどうかは、これだけではわからないが、ひとまずここは放っておく。

標本の分布

price の 1000 個のデータから推定した確率密度は、以下のようであった。

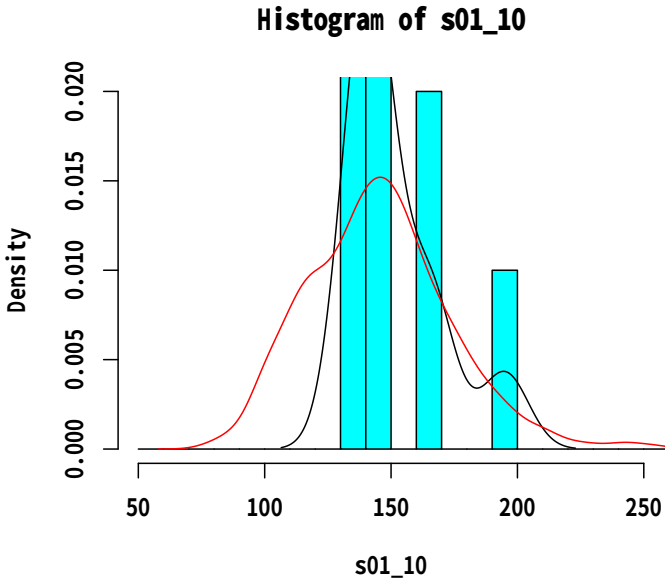
```
> plot(density(price))
```



この中から 10 個のデータを無作為に取り出し、この確率密度に重ねてみる。

hist 関数に引数 xlim=c(50,250),ylim=c(0,.02) と与えることで、x 軸を 50〜250 の範囲に固定し、y 軸を 0〜0.015 の範囲に固定した（比較のために表示のスケールを同じにしたいから）

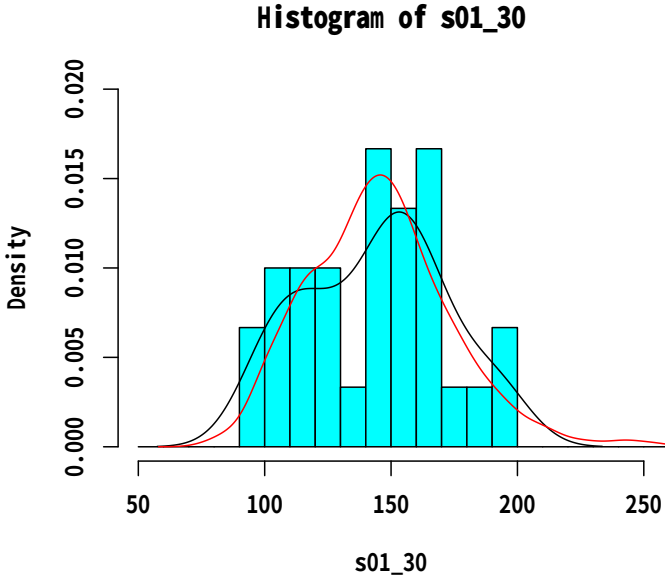
```
> s01_10<-sample(price,10) #price から 10 個の標本抽出
> b01<-seq(50,260,by=10)
> hist(s01_10,prob=T,xlim=c(50,250),ylim=c(0,.02),br=b01,col=5) #抽出した標本のヒストグラム
> lines(density(s01_10)) #抽出した標本の確率密度
> lines(density(price),col=2) #price の確率密度を赤線で
```



この標本が母集団の分布（赤線）を再現しているとは言いがたい。

標本の規模を 30 個に増やしてみる。

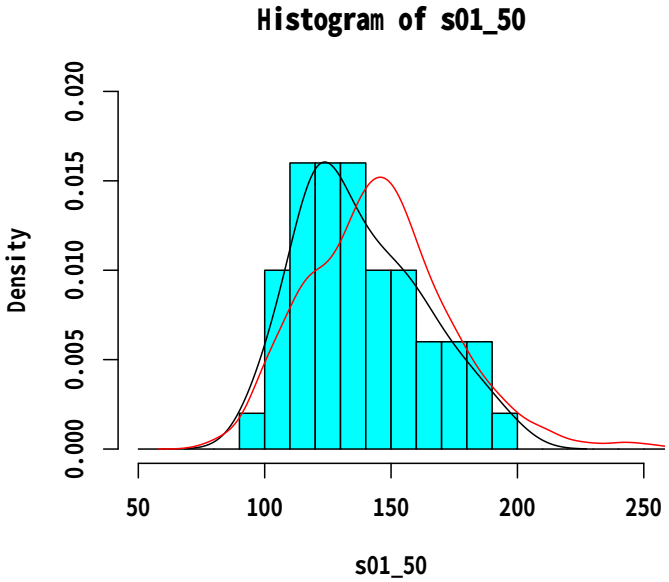
```
> s01_30<-sample(price,30)
> hist(s01_30,prob=T,xlim=c(50,250),ylim=c(0,.02),br=b01,col=5)
> lines(density(s01_30))
> lines(density(price),col=2)
```



だいぶ近づいてきた。

さらに、標本の規模を 50 個に増やしてみる。

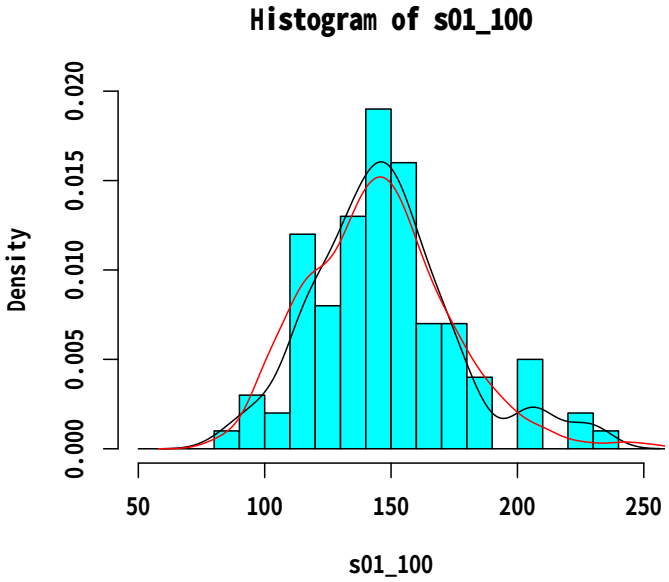
```
> s01_50<-sample(price,50)
> hist(s01_50,prob=T,xlim=c(50,250),ylim=c(0,.02),br=b01,col=5)
> lines(density(s01_50))
> lines(density(price),col=2)
```



だいぶもとの分布を再現したように見える。

もっと増やしてみよう。100 個ではどうか。

```
> s01_100<-sample(price,100)
> hist(s01_100,prob=T,xlim=c(50,250),ylim=c(0,.02),br=b01,col=5)
> lines(density(s01_100))
> lines(density(price),col=2)
```



標本の規模が少なすぎると、母集団の分布は再現できないが、標本の規模を増やすことで、母集団の分布に近づくことができる。

しかも、その規模というのは、意外と小さい？

4.3.3 特定の分布関数に従った標本作成

具体的に母集団のデータはない。しかし、データの確率分布はわかっている。その確率分布で標本抽出を行う。

「特定の確率分布に従って乱数を発生させる」という言い方もする。

実際に分析するときにはあまり考えられないケースだが、統計的手法を検証してみるときには頻繁に使う方法だ。

一様分布に従う乱数: runif 関数

$[0, 1]$ の一様分布に従う乱数を 10 個発生させる。

```
> runif(10)
```

```
 [1] 0.75152134 0.52820275 0.51380385 0.30501309 0.87410159 0.04800163
 [7] 0.87061600 0.89347141 0.98659831 0.47158798
```

もう一回

```
> runif(10)
```

```
 [1] 0.1216415 0.6096680 0.7015536 0.1570893 0.3248387 0.9664978 0.4311805
 [8] 0.1059342 0.8138896 0.4911441
```

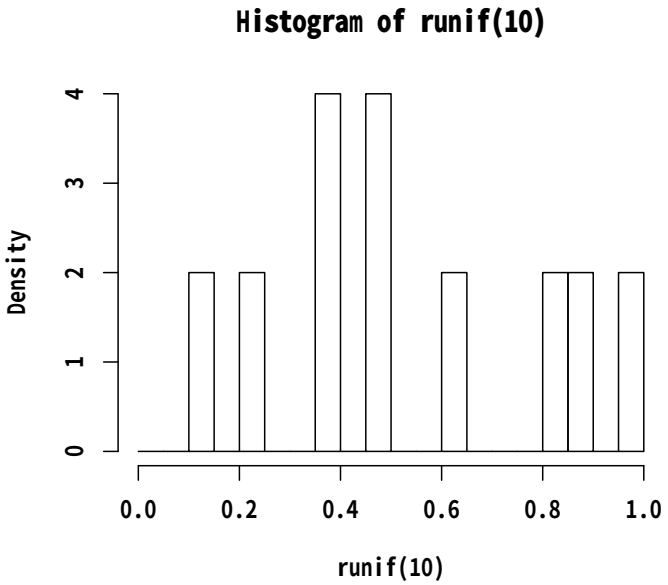
違う値が出た。

ヒストグラムを描いてみる。

```
> (b01=seq(0,1,by=.05)) #0.05 刻みで数列を 0 から 1 までの数列をつくり、これを階級に使う。
```

```
 [1] 0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70
[16] 0.75 0.80 0.85 0.90 0.95 1.00
```

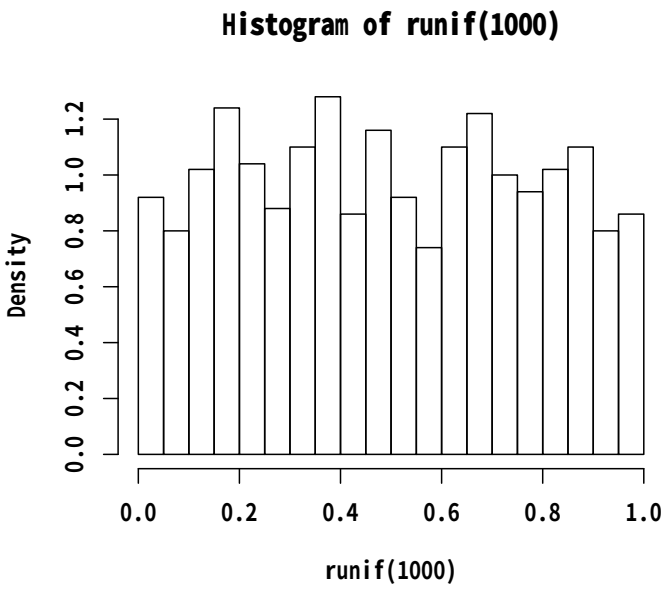
```
> hist(runif(10),br=b01,prob=T)
```



これだけではどんな分布かわからない。

1000 個ぐらいに増やしてみる。

```
> hist(runif(1000),br=b01,prob=T)
```



この乱数が一様分布に従って発生されたものであることが確認できる。

ちなみに、[a,b] の範囲の一様分布に従う乱数を 10 個発生させる場合は、 runif(10,frome=a,to=b) と指定する。[2,5] の範囲だったら、

```
> runif(10,2,5)
```

```
 [1] 4.769393 4.700607 4.832171 2.179640 2.507406 2.228279 2.035189 3.984160
 [9] 3.402504 4.596658
```

正規分布に従う乱数: rnorm 関数

平均 0、標準偏差 1 の正規分布（標準正規分布）に従う乱数を 10 個発生させる。

```
> rnorm(10)
```

```
 [1] -1.01059229  1.00111677 -0.11248332 -0.68259367  1.39075482  0.18725954
 [7] -0.71846449 -1.03526331  0.08557915  1.13367268
```

もう一回。

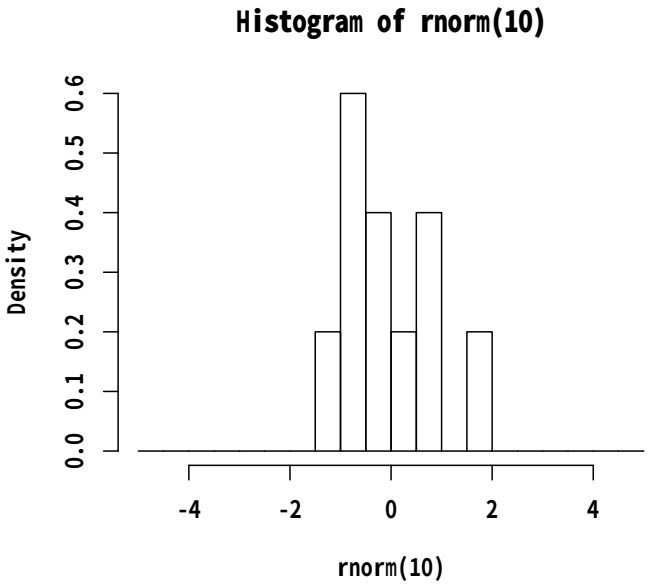
```
> rnorm(10)
```

```
 [1] -0.71968719  0.83879268 -1.21343376  1.56690768  1.71860318  0.09668423
 [7]  0.64286173 -0.30912695  0.32960728 -0.02624830
```

違う値が出た。

ヒストグラムを描いてみる。

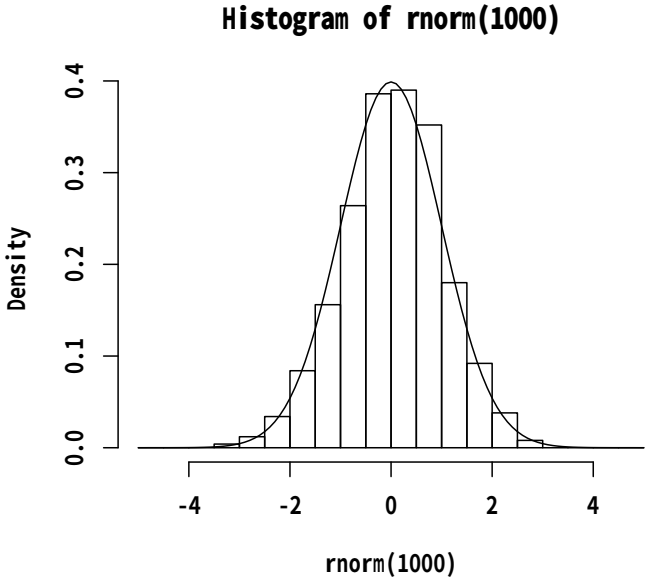
```
> b02<-seq(-5,5,by=.5) #-5 から 0.5 刻みで 5 までの数列を発生させ、これを階級に使う
> hist(rnorm(10),prob=T,br=b02)
```



これではどんな分布かわからない。

1000 個ぐらい乱数を発生させると、様子がわかる。

```
> hist(rnorm(1000),prob=T,br=b02)
> curve(dnorm,add=T) #標準正規分布の密度関数を重ね描きしてみる。
```



ちなみに、平均 a、標準偏差 b の正規分布に従う乱数を 10 個発生させる場合は、 rnorm(10,mean=a,sd=b)
。

平均 5, 標準偏差 0.5 の正規分布に従う乱数を 10 個発生させる。

```
> rnorm(10,5,0.5)
```

```
[1] 4.909926 5.354725 5.589311 4.857643 5.065217 4.520479 4.829455 5.431097
[9] 5.412836 5.087294
```

指数分布に従う乱数： rexp 関数

平均 1（$\lambda = 1/1$）の指数分布に従う乱数を発生させる。

```
> rexp(10)
```

```
 [1] 0.04647605 1.61738715 2.41538473 0.21969223 0.19245917 1.12998349
 [7] 0.01798373 1.53743059 0.77325086 0.91371865
```

もう一回

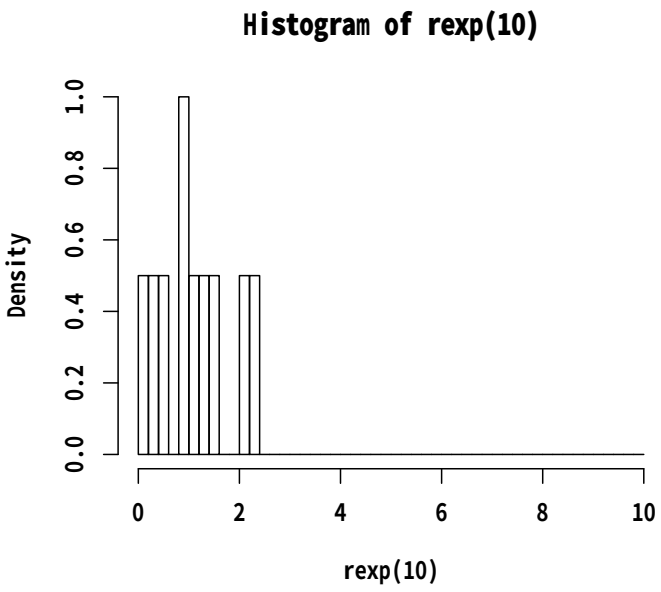
```
> rexp(10)
```

```
 [1] 1.4218606 4.4049258 3.1651887 3.1308655 0.2700161 2.5713330 0.2629726
 [8] 0.9890741 1.1959463 0.2143055
```

違う値が出た。

ヒストグラムに描いてみる。

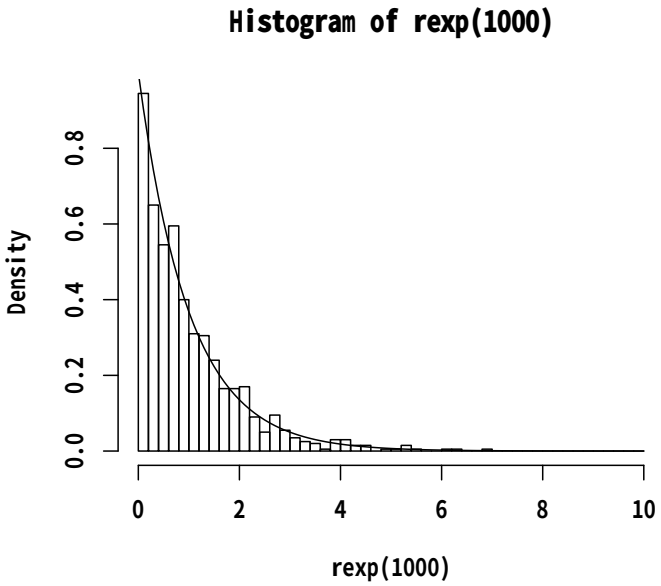
```
> b03=seq(0,10,by=.2) #0 から 10 の間を 0.2 ごとに区切った数列を発生させ、これを階級に使う
> hist(rexp(10),br=b03,prob=T)
```



これだけではよくわからない。

1000 個の乱数を発生させてみる。

```
> hist(rexp(1000),br=b03,prob=T)
> curve(dexp,add=T) #lambda=1 の指数分布の密度関数を重ね描きしてみる
```



ちなみに、平均 a（$\lambda = 1/a$）の指数分布に従う乱数を 10 個発生させるには、 rexp(10,rate=1/a) 。
平均 5 の指数分布に従う乱数を 10 個発生させる。

```
> rexp(10,1/5)
```

```
 [1]  2.7794694  6.7563077  3.4843575  2.0987525  0.3321348  9.2853374
 [7] 11.1455233  3.8172277  1.7448908  6.5049560
```

その他の分布に従う乱数

R には、ほかにもいろいろな分布に従う乱数を発生させる関数が用意されている。

分布	分布関数	密度関数	乱数
一様分布	punif	dunif	runif
正規分布	pnorm	dnorm	rnorm
指数分布	pexp	dexp	rexp
t 分布	pt	dt	rt
カイ二乗分布	pchisq	dchisq	rchisq
F 分布	pf	df	rf
ガンマ分布	pgamma	dgamma	rgamma

などなど。

第一種極値分布に従う乱数

R に用意されていない分布関数に従う乱数も、分布関数の逆関数がわかれば、$[0, 1]$ の一様分布に従う乱数を使って乱数を発生させることができる。

第一種極値分布の分布関数は、

$$y = F(x) = exp(-exp(-\lambda x))$$

だった。この逆関数は、

$$x = -\frac{1}{\lambda} ln(-ln(y))$$

で計算できるから、第一種極値分布に従う乱数を発生させるには、この y に $[0,1]$ の一様分布に従う乱数をあてめればよい。

$\lambda = 1$ の第一種極値分布に従う乱数を 10 個発生させる。

```
> -log(-log(runif(10)))
```

```
 [1] -0.6305530  1.3802120 -0.9431984 -0.5805300 -1.0094690  2.5345467
 [7]  3.1080989 -0.9138797  2.3121082  1.4376024
```

もう一回。

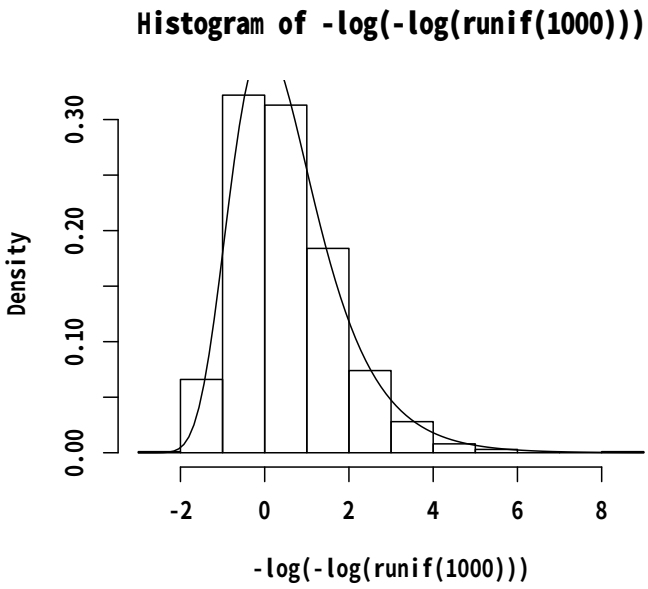
```
> -log(-log(runif(10)))
```

```
 [1] -0.12286257  0.86290635 -0.81166864  1.25660556  1.79951043 -0.08128705
 [7]  1.08096396  0.34455861  1.28084198 -0.67171351
```

ちがう値が出た。

1000 個乱数を発生させて分布を確認する。

```
> hist(-log(-log(runif(1000))),prob=T)
> curve(exp(-x)*exp(-exp(-x)),add=T) #lambda=1 の指数分布を重ね描きする
```



4.3.4　確率分布を比較する

実際にとられたデータの分布が、どの確率分布に近いかを調べる。

現実に発生したデータというのは、純粋に正規分布であるとか、一様分布であるとか、分布関数が知られた分布のいずれかそのものというのは、一部の例外を除いて少ない。

しかし、そのものではないにしろ、ある程度近似できる場合は少なくない。

近似が難しい場合でも、例えば正規分布よりも裾野が広いとか狭いとか、左に偏っているとか、特定の分布関数を基準に、分布の様子を表現することができる。

例えば、中古プリウスの価格の分布はどんなだろうか？

確率密度を描いてみて比較する

どの分布に近いかは、分布を重ね合わせてみればよいではないか。

例えば、t 分布というものがあるが、これは一見、正規分布に似ているが、分布を重ね合わせてみると、すそ野が広いことがわかる。

例えば自由度 3 の t 分布に従う乱数を 1000 個発生させて、その確率密度を、平均 0、標準偏差 1 の正規分布（標準正規分布）に従う乱数 d01_n1 と比較する。

平均 0、標準偏差 1 の正規乱数を 1000 個発生させる

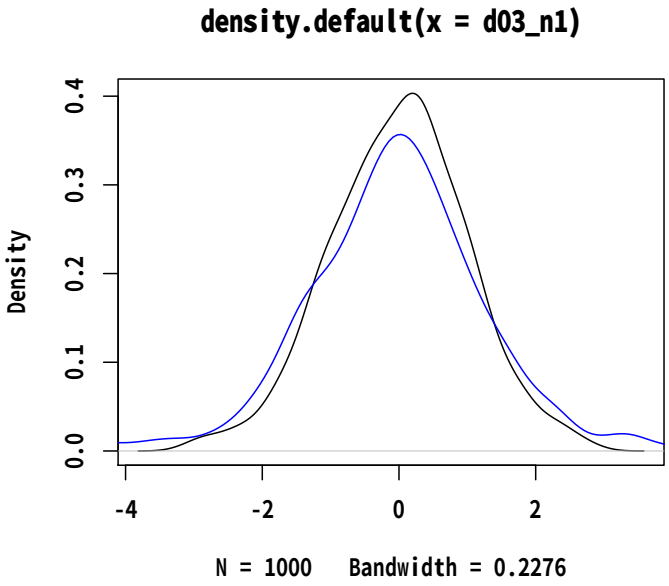
```
> d03_n1<-rnorm(1000)
```

自由度 3 の t 分布に従う乱数を 1000 個発生させる。

```
> d03_t1<-rt(1000,3)
```

２つの確率密度を density 関数で推定し、 plot 関数で分布を描いてみる。

```
> plot(density(d03_n1))
> lines(density(d03_t1),col=4)
```



t 分布（青線）の方が、すそ野が広い。

第一種極値分布は、やや右にすそ野が広く偏っている。

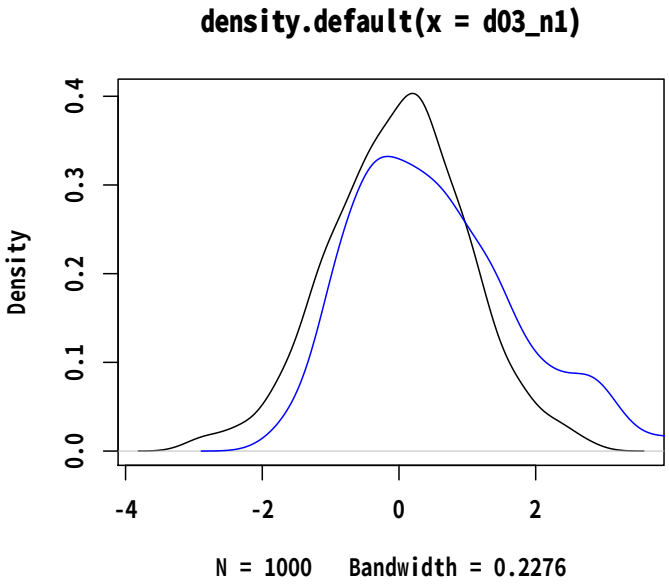
例えば、パラメータ $\lambda = 1$ の第一種極値分布に従う乱数を 1000 個発生させて、その確率密度を、標準正規乱数 d01_n1 と比較してみる。

パラメータ $\lambda = 1$ の第一種極値分布に従う乱数を 1000 個発生させる。

```
> d03_ex1<--log(-log(runif(1000)))
```

この確率密度を density 関数で推定してグラフに描き、 d01_n1 と比較する。

```
> plot(density(d03_n1))
> lines(density(d03_ex1),col=4)
```



直感的には、そちらが間違いないかもしれないが、これでは判断を誤ることもある。

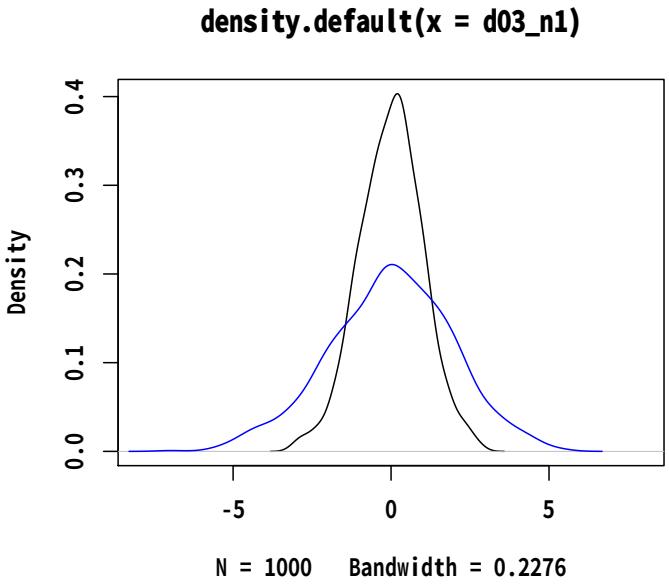
たとえば、平均 0、標準偏差 1 の正規分布（標準正規分布）と、平均 0、標準偏差 2 の正規分布を比較する。

平均 0、標準偏差 2 の正規乱数を 1000 個発生させる

```
> d03_n2<-rnorm(1000,0,2)
```

d03_n1 と d03_n2 の 2 つのデータの確率密度を比較する。

```
> plot(density(d03_n1),xlim=c(-8,8))
> lines(density(d03_n2),col=4)
```

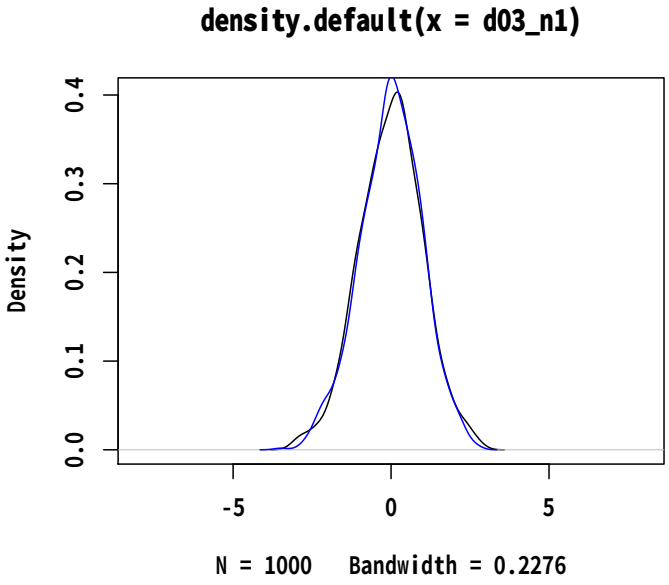


標準偏差が 2 の方（青線）の方が、幅広に見えるが、これはバラツキが大きいというだけであって、正規分布に変わりない。

d03_n2 を標準偏差 2 で割ると、 d02_n1 とほぼ同じ分布となる。

```
> plot(density(d03_n1),xlim=c(-8,8))
```

```
> lines(density(d03_n2/2),col=4)
```



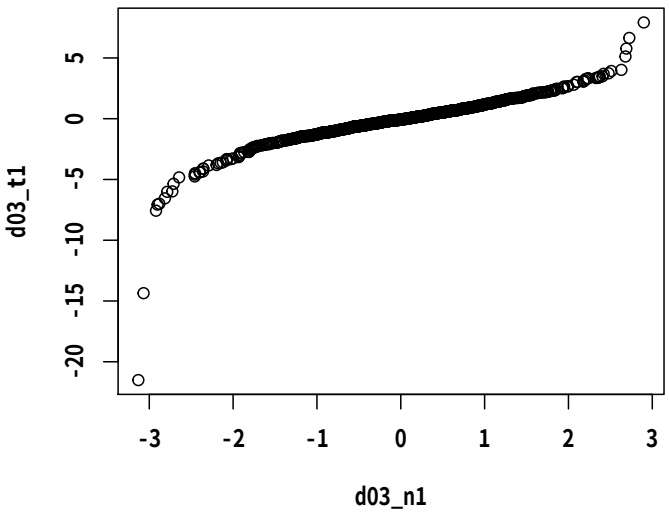
Q-Q プロットで比較する： qqplot 関数

２つの分布を小さい方から順に並べてプロットして、分布の形状を比較するという方法がある。

t 分布を正規分布と比較する

横軸に標準正規分布に従う乱数 d03_n1 を、縦軸に t 分布に従う乱数 d03_t1 を、小さい順に並べてプロットする。

```
> qqplot(d03_n1,d03_t1)
```

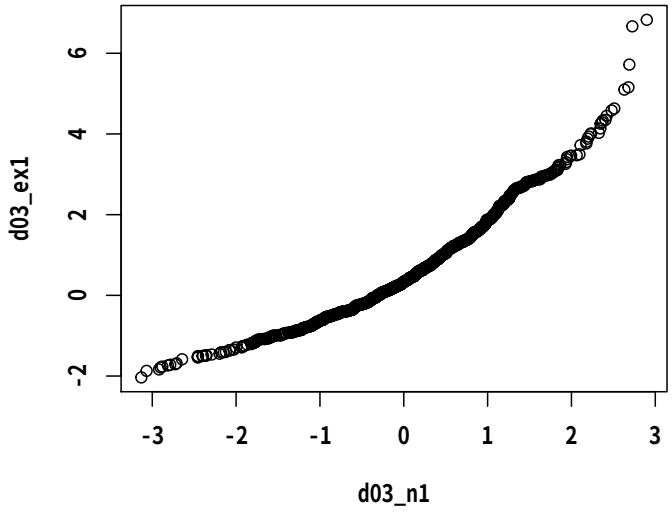


横軸の分布に比べて、縦軸の分布の方がすそ野が広いと、Q-Q プロットは、逆 S 字となる。（最初は、どういうことかよくわからないかもしれないが、２つのデータを小さい方から並べたものであることをよく考えると、わかるはずだ）

第一種極値分布を正規分布と比較してみる

さきほど発生させた第一種極値分布に従う乱数 d03_ex1 を縦軸に、正規乱数 d03_n1 を横軸にプロットして比較する。

```
> qqplot(d03_n1,d03_ex1)
```

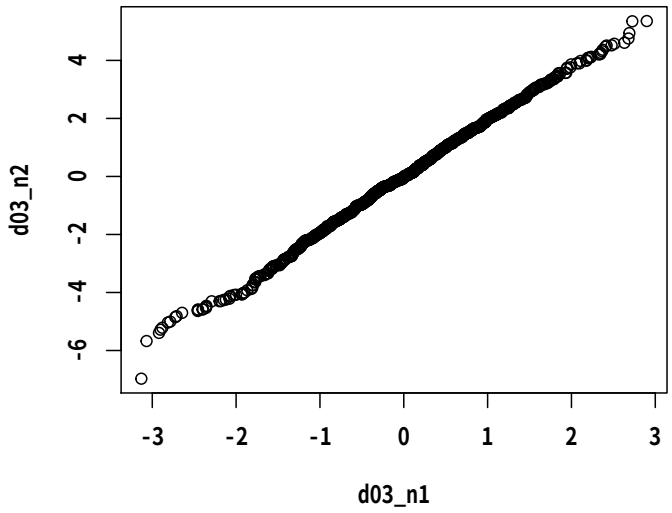


正規分布に比べて、縦軸の分布の方が左にすそ野が広いと、Q-Q プロットは、凹型の曲線となる。

正規分布どおしを比較してみる

標準偏差 2 の正規分布に従う乱数 d03_n2 を縦軸に、標準正規分布に従う乱数を横軸にプロットして、２つの分布を比較する。

```
> qqplot(d03_n1,d03_n2)
```

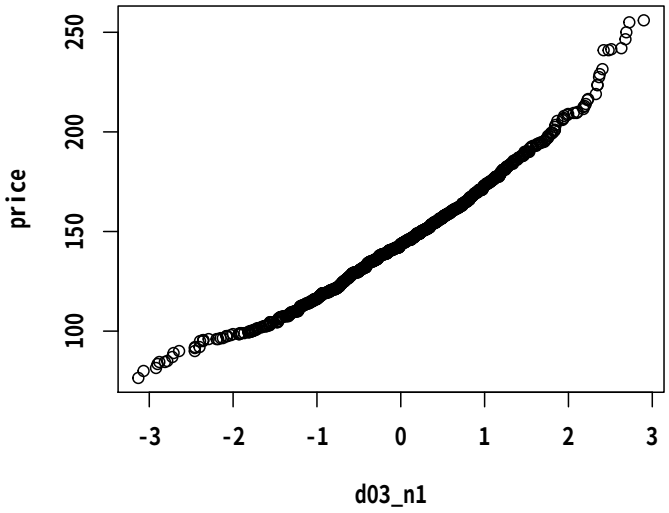


平均や標準偏差が違っても、同じ確率分布の形ならば、Q-Q プロットは直線に近くなる。

プリウスの価格の分布はどうか？

プリウスの価格 price を縦軸に、正規分布に従う乱数 d03_n1 を横軸にプロットして、二つのと比較してみる。

```
> qqplot(d03_n1,price)
```

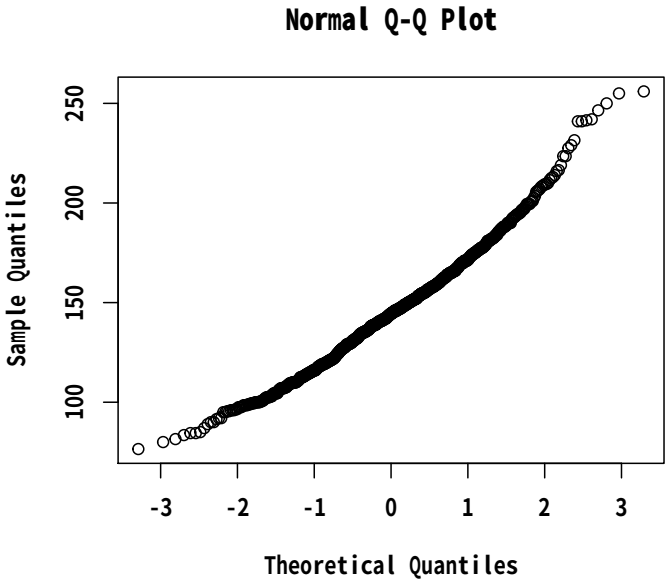


Q-Q プロットが若干凹型をしているので、正規分布よりいくらか左にすそ野が広い分布であることがわかる。

特に正規分布と比較する： **qqnorm** 関数

比較の対象が正規分布である場合は、 qqnorm 関数を使うと同じ数の正規乱数を発生させなくても、目的のデータだけを引数に与えることで、分布形を評価することができる。

```
> qqnorm(price)
```



正規分布と比較する場合は、正規乱数を発生させた場合のバラツキの影響がないので、 qqnorm 関数で評価した方が安定的な評価ができる（特にデータの規模が小さいときは、ありがたい)。

4.4　平均と分散（標準偏差）

確率分布でも確率密度でも、分布の形状を示せば、問題としているデータがどのように分布しているか知ることができる。

しかし、全国 47 都道府県の所得の分布を比較したい、などという時、分布を 47 個並べるというのも面倒。

そこで分布の代表値として、平均や分散（または、その平方根をとった標準偏差）が用いられる。

たとえば中古車プリウスの価格について、以下である。

データ数	平均	分散（標準偏差）
1000	145.3	805.5 (28.4)

4.4.1　母平均

単純平均： mean 関数

通常の平均値は算術平均とも言われる。

x_1, x_2, x_3 の３つの値があったら、その平均値は以下。

$$\mu = \frac{x_1 + x_2 + x_3}{3}$$

たとえば、$(x_1, x_2, x_3) = (1, 2, 3)$ だったら、

```
> mean(c(1,2,3))
```

```
[1] 2
```

加重平均： weighted.mean 関数

今更、と言われるかもしれないが、それでは以下の平均値はどうか？

i	値 x_i	個数 n_i
1	1	5
2	3	2
3	5	3

こういう場合は、各値に対応する個数を掛けて総数で割るという操作が必要。これを「加重平均」という。加重平均との対比でて、先ほどの平均を「単純平均」または「単純算術平均」と言うことがある。

$$\mu = \frac{n_1 x_1 + n_2 x_2 + n_3 x_3}{n_1 + n_2 + n_3}$$

R で加重平均を求めるには、 weighted.mean 関数の引数 w= に個数を与えればよい。

```
> weighted.mean(c(1,3,5),w=c(5,2,3))
```

```
[1] 2.6
```

確率が与えらた加重平均： weighted.mean 関数

各値の個数ではなく、構成比が与えられていたらどうか？

i	値 x_i	構成比 $\Pr(x_i)$
1	1	0.5
2	3	0.2
3	5	0.3

その場合の平均値は以下で求める。

$$\mu = \Pr(x_1)x_1 + \Pr(x_2)x_2 + \Pr(x_3)x_3$$

R では、 weighted.mean 関数がそのまま使える。

```
> weighted.mean(c(1,3,5),w=c(.5,.2,.3))
```

```
[1] 2.6
```

これは、$x_1 = 1$ が 50 %を占めると読めるが、$x_1 = 1$ が 0.5 の確率で生じた、と理解することもできる。
この場合、平均値は「期待値」とも呼ばれる。一般には以下。

$$\mu = \sum_i^N \Pr(x_i)x_i$$

確率分布が与えられた場合の平均

確率変数 x が [0,1] の範囲の一様分布に従う場合、期待値は、

$$\mu = \int_0^1 f(x)xdx$$

[0,1] の範囲で $f(x) = 1$ なので

$$\mu = \int_0^1 xdx = \frac{1}{2}$$

R で積分計算は直接できないが、たくさん乱数を発生させて平均をとればよい。
runif 関数で [0,1] の範囲の一様乱数を発生させて、 mean 関数で平均をとるだけ。

```
> d04_r1<-runif(10000)
> mean(d04_r1)
```

```
[1] 0.4994271
```

確率密度で加重平均するという方法もある（どうせ確率密度は、どれでも 1 だが）。

```
> weighted.mean(d04_r1,dunif(d04_r1))
```

```
[1] 0.4994271
```

確率変数 x が平均 μ 標準偏差 σ の正規分布に従うとき、期待値は

$$\mu = \int_{-\infty}^{\infty} f(x)xdx$$

正規分布の密度関数の定義を $f(x)$ に代入して、

$$\mu = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x -^m u)^2}{2\sigma^2}\right) xdx$$

難しそうな数式だが、R では数値的に計算できる。

たとえば、平均 3、標準偏差 2 の正規分布に従う確率変数の期待値は、とりあえず、 rnorm 関数で乱数をたくさん発生させて平均をとればよい。

```
> mean(rnorm(10000,3,2))
```

```
[1] 2.996478
```

あるいは、広い範囲の一様乱数をたくさん発生させて、対応する確率密度を dnorm 関数で求めて、それによって加重平均することもできる。

```
> d04_r2<-runif(10000,-100,100)
> weighted.mean(d04_r2,dnorm(d04_r2,3,2))
```

```
[1] 2.917954
```

4.4.2 標本平均

ここまで計算した平均は、データそのものの平均だった。たとえ、データが３個しかなくても、それがすべてのデータだったら、それが母集団となる。逆に、1000 個あっても、本当に知りたいデータの一部であれば、それは標本。

標本平均の計算

標本平均の計算方法は、母平均の計算方法と同じ。

N 個の母集団から、x_1, x_2, x_3 の３つの標本を抜き出した。その場合の標本平均は以下。

$$\bar{x} = \frac{x_1 + x_2 + x_3}{3}$$

母平均は μ、標本平均は $\bar{x}$ で表していることに注意！

母平均は一つしかないが、標本平均は、標本の取り方で変わってきてしまう。

もちろん、$\mu = \bar{x}$ でもない。

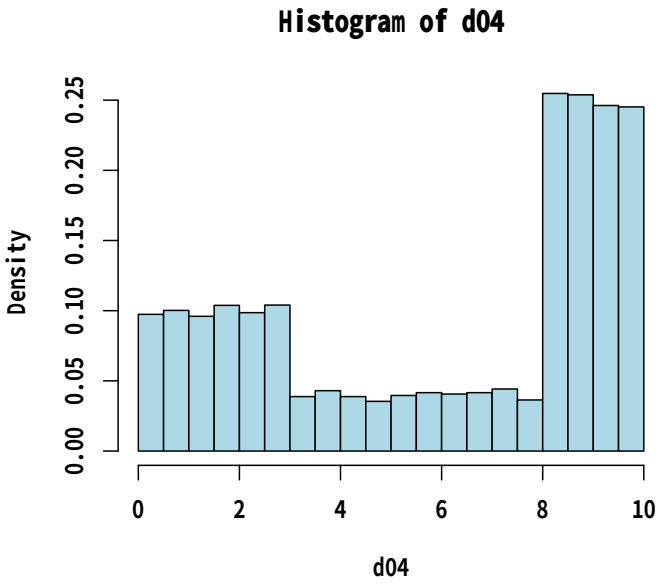
それでは、標本平均だけ見ていても意味がないかというと、そういうわけでもない。

そのことを見るために、架空の母集団を作成して、そこから標本を抽出し、その標本平均と母平均との関係を見ることにする。

母集団

架空の母集団としてだいぶヘンな分布を作成してみる。

```
> set.seed(210) #説明の都合上、乱数のパターンを固定した。
> d04_1<-runif(3000,0,3) #[0,3] の範囲の一様乱数を 3000 個
> d04_2<-runif(2000,3,8)  #[3,8] の範囲の一様乱数を 2000 個
> d04_3<-runif(5000,8,10) #[8,10] の範囲の一様乱数を 50000 個
> d04<-c(d04_1,d04_2,d04_3) #以上の３パターンの乱数をつないだ
> hist(d04,col="lightblue",prob=T) #ヒストグラム
```



母集団の平均（「母平均」と呼ぶ）は、

```
> mean(d04)
```

```
[1] 6.053284
```

母集団の分散（「母分散」と呼ぶ）は、

```
> var(d04)
```

```
[1] 11.35773
```

母集団の標準偏差はその平方根。

```
> sd(d04)
```

```
[1] 3.370123
```

5 個の標本平均をとる

母集団 10000 のデータから 5 個だけ標本を抽出する。

```
> (d04s5_01<-sample(d04,5))
```

```
[1] 2.469269 8.124595 1.178544 9.613013 2.085397
```

標本平均を計算する

```
> (m5_01<-mean(d04s5_01))
```

```
[1] 4.694164
```

一つの標本平均が得られた。

同じ作業をもう一回。

母集団 10000 のデータから 5 個だけ標本を抽出する。

```
> (d04s5_02<-sample(d04,5))
```

```
[1] 7.3240699 8.7587261 9.2494670 0.6955080 0.9100573
```

標本平均を計算する

```
> (m5_02<-mean(d04s5_02))
```

```
[1] 5.387566
```

同じ作業をしたが、さっきとは違う標本平均が得られた。
さらにもう一回。
母集団 10000 のデータから 5 個だけ標本を抽出する。

```
> (d04s5_03<-sample(d04,5))
```

```
[1] 5.769856 8.256815 9.832331 2.766881 1.736355
```

標本平均を計算する

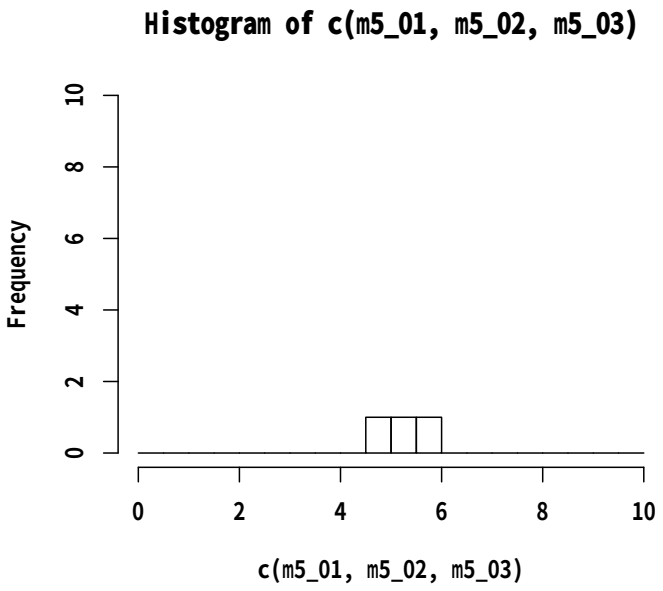
```
> (m5_03<-mean(d04s5_03))
```

```
[1] 5.672447
```

今度も違う標本平均が得られた。

標本平均の分布を確認する
得られた３つの標本平均を使ってヒストグラムを描いてみる。

```
> hist(c(m5_01,m5_02,m5_03),xlim=c(0,10),ylim=c(0,10),br=seq(0,10,by=.5))
```



標本をとるたびに標本平均は変わるということだけはわかるが、たった３つでははなんとも・・・

もうちょっとたくさんの５個標本平均を集める

いちいち、 m5_04<-mean (... と代入していくのもなんなので、ここは繰り返しができる for 関数を使う

とりあえず 50 個の標本平均をとってみよう。

まず、結果を代入する空（NA）50 個のベクトル m5 を作成する。

```
> m5<-rep(NA,50)
```

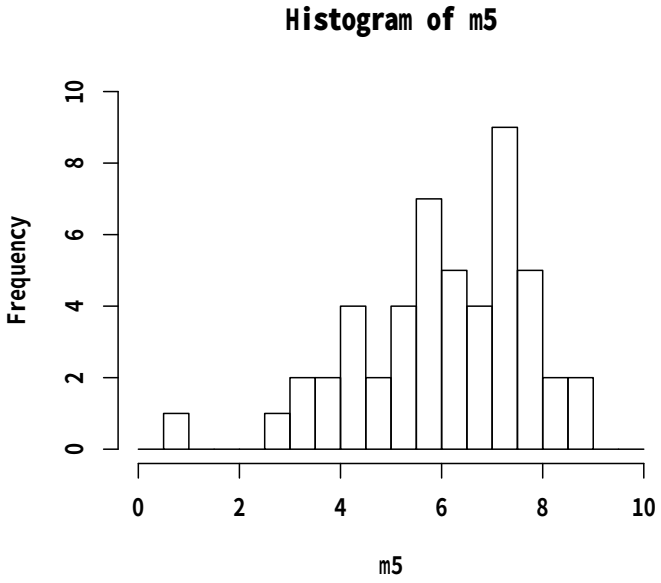
for 関数を使って、 m5 の要素に 1 から 50 まで、繰り返し標本平均を代入する。

```
> for(i in 1:50) m5[i]<-mean(sample(d04,5))
```

d04 から５個の標本を抜き出して平均をとった値を、ベクトル m5 の i 番目の要素に代入する。これを i=1 から i=50 まで繰り返す、、、という意味になる。

得られた 50 個の標本平均をヒストグラムに描いてみる。

```
> hist(m5,xlim=c(0,10),ylim=c(0,10),br=seq(0,10,by=.5))
```



標本平均はさまざまで、2 に近いものから 10 に近いものまで幅広い。

4.4.3　標本平均の不偏性

しかしながら、標本平均のさらに平均をとってみると母平均にかなり近い値が得られる。

```
> mean(m5)
```

```
[1] 6.04623
```

ちなみに、母平均は

```
> mean(d04)
```

```
[1] 6.053284
```

標本平均 $\bar{x}$ は母平均 μ に必ずしも一致しないし、むしろ大きくずれることもある。

しかしながら、標本平均 $barx$ をたくさん集めてその平均をとると、それは母平均 μ に一致する。

この標本平均 $\bar{x}$ をたくさん集めて平均をとったものは、標本平均 $\bar{x}$ の期待値であり、$E[\bar{x}]$ と表す。つまり、

$$E[\bar{x}] = \mu$$

ということであり、このことを、標本平均 $\bar{x}$ は「不偏性」を持つと言う。

4.4.4 標本平均の一致性：大数の法則

標本平均をいろいろな標本規模でとってみる

5 個より大きな規模の標本平均をとったらどうなるか？

標本の規模が 5 個、20 個、50 個、100 個の 4 種類の標本をそれぞれ 1000 回とり、それぞれ 1000 パターンの標本平均を求め、それぞれの確率密度を描いてみる。

5 個標本の平均を 1000 パターン、m5 に代入

```
> m5<-rep(NA,1000)
> for(i in 1:1000) m5[i]<-mean(sample(d04,5))
```

20 個標本の平均を 1000 パターン、m20 に代入

```
> m20<-rep(NA,1000)
> for(i in 1:1000) m20[i]<-mean(sample(d04,20))
```

50 個標本の平均を 1000 パターン、m50 に代入

```
> m50<-rep(NA,1000)
> for(i in 1:1000) m50[i]<-mean(sample(d04,50))
```

100 個標本の平均を 1000 パターン、m100 に代入

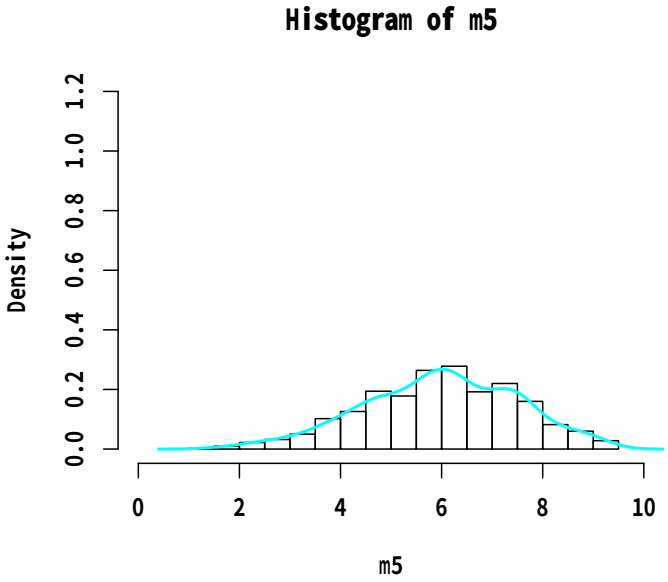
```
> m100<-rep(NA,1000)
> for(i in 1:1000) m100[i]<-mean(sample(d04,100))
```

標本平均の分布を描いてみる

5 個標本の平均の分布

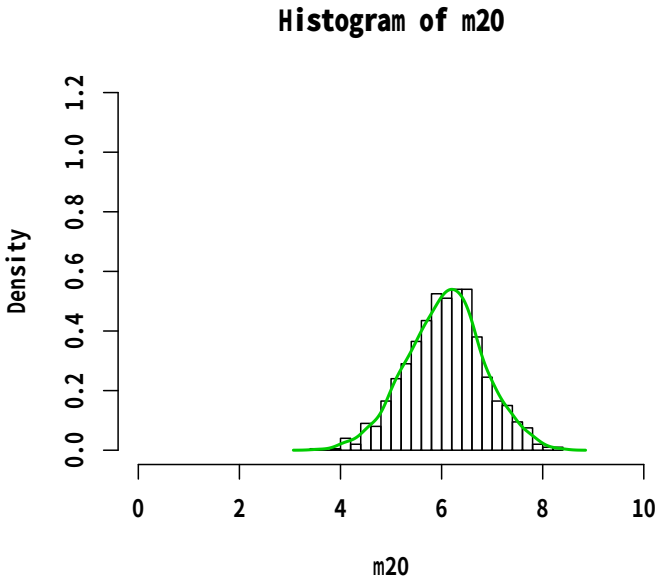
hist 関数の引数 `br="FD"` は、ヒストグラムのところで紹介した階級の与え方のひとつ。

```
> hist(m5,prob=T,br="FD",xlim=c(0,10),ylim=c(0,1.2))
> lines(density(m5),col=5,lwd=2)
```



20 個標本の平均の分布

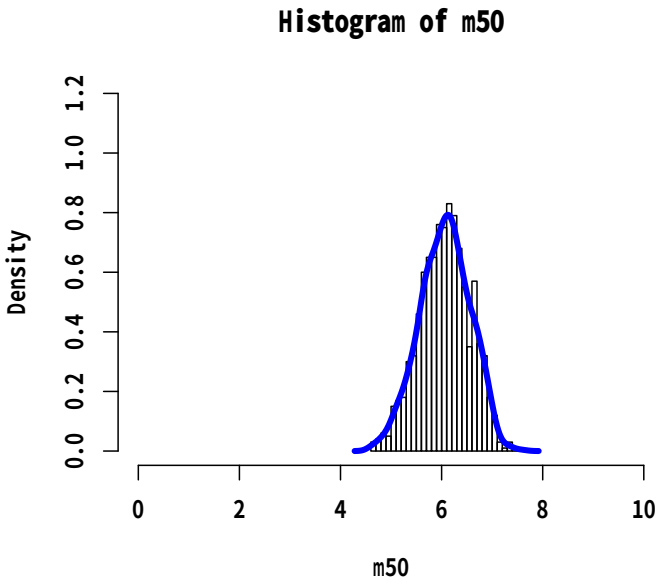
```
> hist(m20,prob=T,br="FD",xlim=c(0,10),ylim=c(0,1.2))
> lines(density(m20),col=3,lwd=2)
```



5 個標本よりも分布が狭くなった。

50 個標本の平均の分布

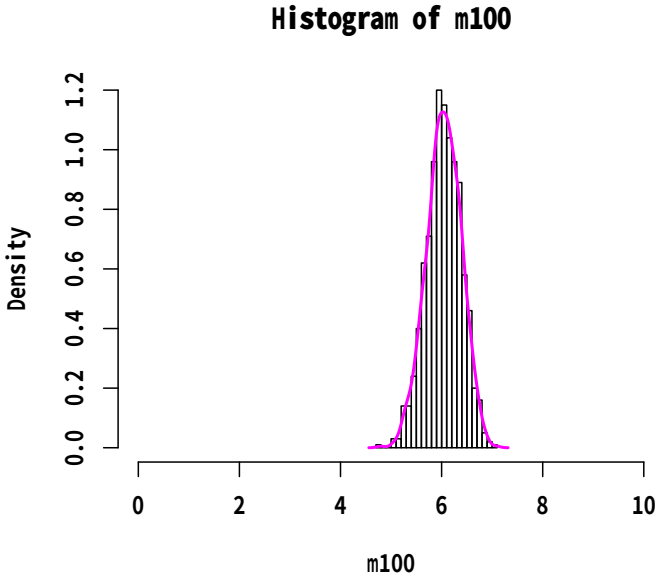
```
> hist(m50,prob=T,br="FD",xlim=c(0,10),ylim=c(0,1.2))
> lines(density(m50),col=4,lwd=4)
```



20 個標本よりもさらに、分布が狭くなった

100 個標本の平均の分布

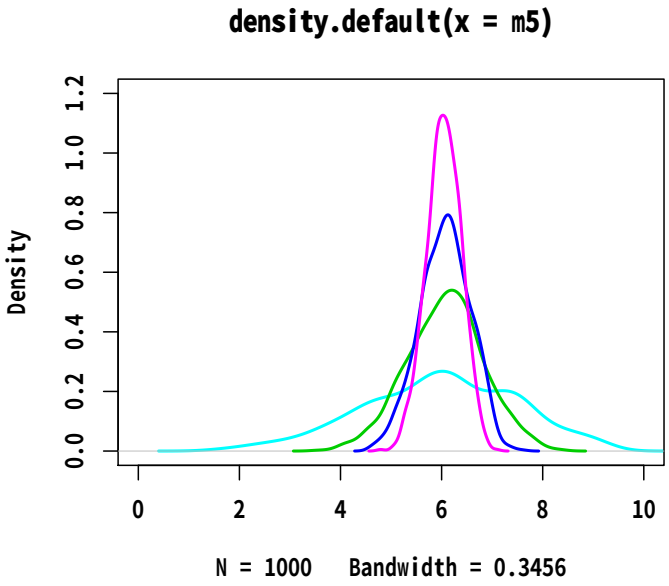
```
> hist(m100,prob=T,br="FD",xlim=c(0,10),ylim=c(0,1.2))
> lines(density(m100),col=6,lwd=2)
```



さらに分布は狭くなった。

４種類の標本平均の分布の比較を行う。確率密度だけ４種類を重ね描きしてみる。

```
> plot(density(m5),xlim=c(0,10),ylim=c(0,1.2),col=5,lwd=2)
> lines(density(m20),col=3,lwd=2)
> lines(density(m50),col=4,lwd=2)
> lines(density(m100),col=6,lwd=2)
```



４種類の標本平均のそのまた平均をとってみる

別々に平均をとるのは面倒なので、いったんデータ・フレームに括ってから、 lapply 関数で一気に平均をとった。

lapply 関数は、リストやデータ・フレームの要素ごとに、２番目の引数で指定した関数を適用してくれる。

```
> m04<-data.frame(m5,m20,m50,m100)
> lapply(m04,mean)
```

```
$m5
[1] 5.97226

$m20
[1] 6.071203

$m50
[1] 6.063544

$m100
[1] 6.043208
```

母平均が 6.042 だから、いずれも母平均に近い値となっていること（不偏性）が確認できた。

４種類の標本平均の分散をとってみる

```
> lapply(m04,var)
```

```
$m5
[1] 2.337221

$m20
[1] 0.5748148

$m50
[1] 0.2476723

$m100
[1] 0.1166553
```

標本の規模が大きいほど、分散は小さくなっている。

母分散 σ^2 を標本規模 n で割った値 σ^2/n を求めてみる。

母分散 $\mathrm{var(d04)}$ はひとつの値だが、４つの数からなるベクトルで割ると、それぞれの値で割った値を返してくれる。

```
> var(d04)/c(5,20,50,100)
```

```
[1] 2.2715453 0.5678863 0.2271545 0.1135773
```

左から、母分散を 5 で割った値、20 で割った値、50 で割った値、100 で割った値。

標本規模 5、20、50、100 の４種類の標本平均の分散とほぼ一致している。

大数の法則

つまり、標本規模 n の標本平均の分散は、母集団の分散を標本規模で割った値である。

$$V(\bar{x}) = \frac{\sigma^2}{n}$$

・・・ということは、標本規模 n を大きくしていくと、標本平均の分散は限りなく小さくなる。

標本平均のそのまた平均（期待値）は、母平均と同じだから・・・

標本平均の分布は、標本規模を大きくすると母平均のまわりに集中することになる。

つまり、標本規模が大きいほど、標本平均が母平均から離れる可能性は小さくなるということ。

これを「大数の法則」と言う！

一致性

大数の法則のように、なんらかの推定量が、標本規模 n を大きくすることで、特定の数値に収束することを、一致性を持つという。

つまり、標本平均は、「不偏性」をもち、かつ「一致性」を有する母平均の推定量である。

4.4.5 中心極限定理

標本平均の正規化

標本規模 n の標本平均を、母平均と母分散を使って以下のように正規化する。

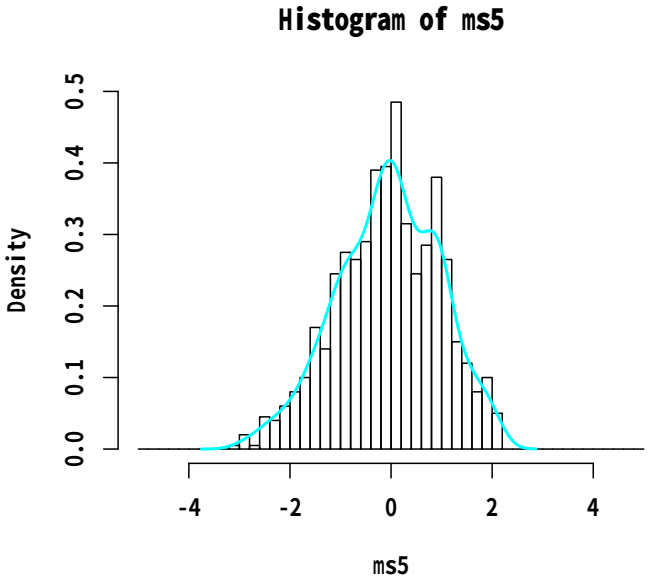
$$\frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

5 個標本の標本平均の正規化は、

```
> ms5<-(m5-mean(d04))/(sd(d04)/sqrt(5))
```

ヒストグラムと確率密度は、

```
> b02<-seq(-5,5,by=0.2)
> hist(ms5,prob=T,br=b02,ylim=c(0,.5))
> lines(density(ms5),col=5,lwd=2)
```

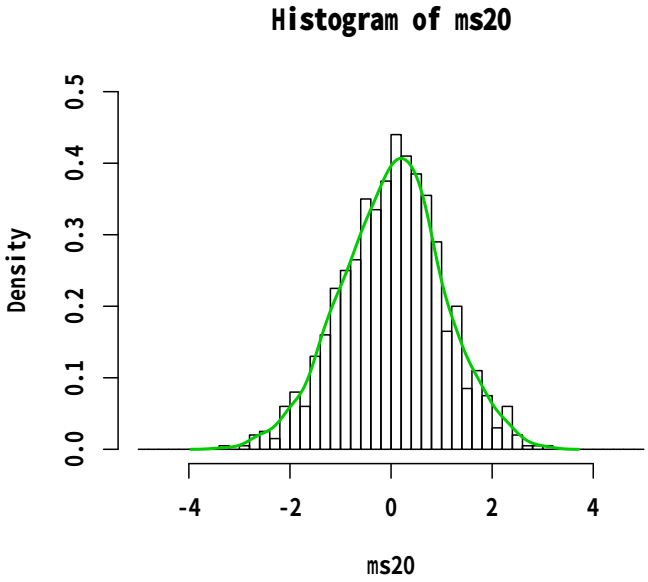


20 個標本の標本平均の正規化は、

```
> ms20<-(m20-mean(d04))/(sd(d04)/sqrt(20))
```

ヒストグラムと確率密度は、

```
> hist(ms20,prob=T,br=b02,ylim=c(0,.5))
> lines(density(ms20),col=3,lwd=2)
```



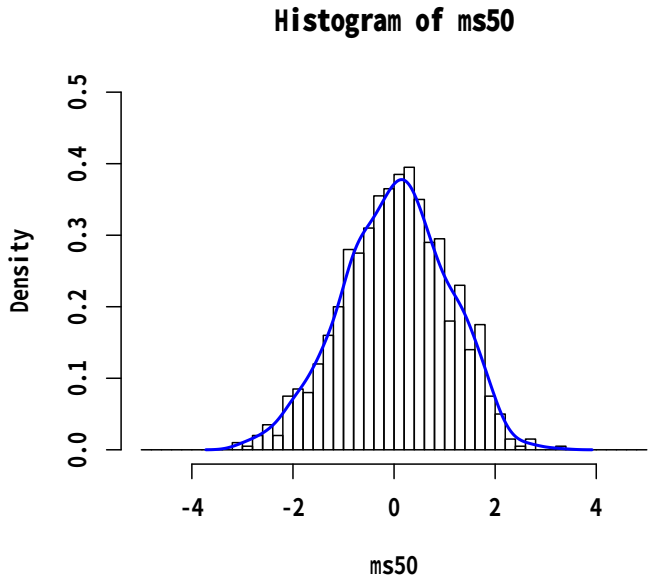
50 個標本の標本平均の正規化は、

```
> ms50<-(m50-mean(d04))/(sd(d04)/sqrt(50))
```

ヒストグラムと確率密度は、

```
> hist(ms50,prob=T,br=b02,ylim=c(0,.5))
```

```
> lines(density(ms50),col=4,lwd=2)
```

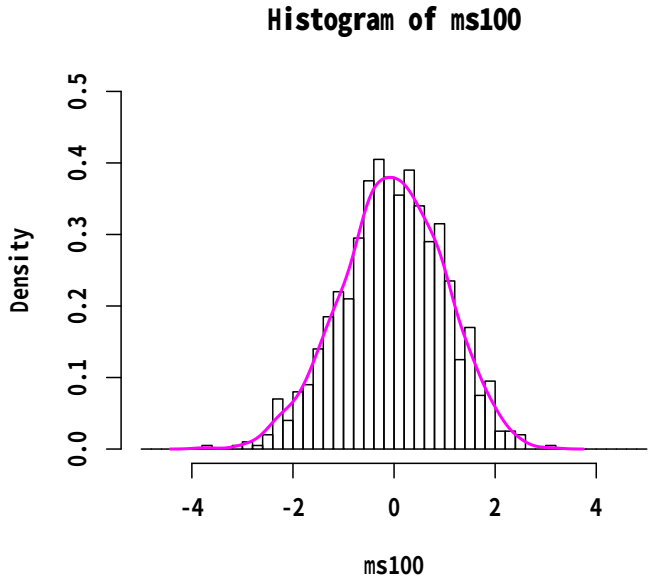


100 個標本の標本平均の正規化は、

```
> ms100<-(m100-mean(d04))/(sd(d04)/sqrt(100))
```

ヒストグラムと確率密度は、

```
> hist(ms100,prob=T,br=b02,ylim=c(0,.5))
> lines(density(ms100),col=6,lwd=2)
```



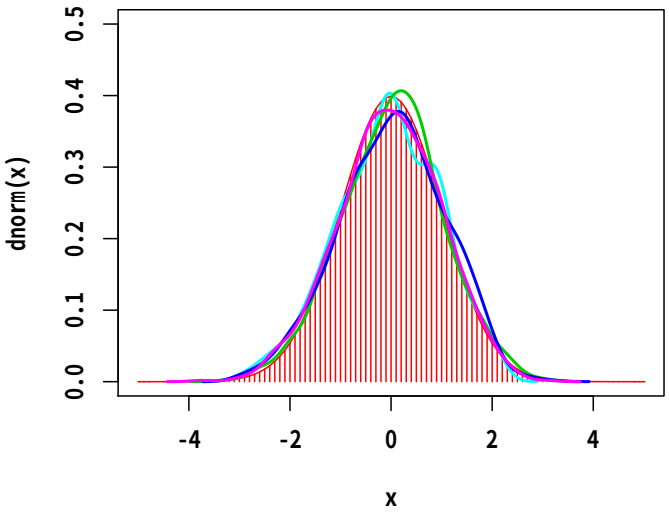
どれも同じように見える。

標準正規分布と比べる

標準正規分布に、各確率密度を重ね描きしてみよう。

背後に赤く標準正規分布を描いた。

```
> curve(dnorm,-5,5,type="h",col=2,ylim=c(0,.5))
> curve(dnorm,col=2,add=T)
> lines(density(ms5),col=5,lwd=2)
> lines(density(ms20),col=3,lwd=2)
> lines(density(ms50),col=4,lwd=2)
> lines(density(ms100),col=6,lwd=2)
```



標本規模が大きいほど、標準正規分布に近い分布となっている。

中心極限定理

つまり、標本規模が大きいと、標本平均を母平均と母標準偏差で正規化した値は、標準正規分布に近似できる。

$$\frac{\bar{x}-\mu}{\frac{\sigma}{\sqrt{n}}} \sim N(0,1)$$

これが「中心極限定理」。

母平均 μ と標準偏差 $\sigma/\sqrt{n}$ で正規化された、標本規模 n の標本平均が標準正規分布に従うならば、もとの標本平均 $\bar{x}$ は、平均 μ、標準偏差 $\sigma/\sqrt{n}$ の正規分布に従う確率変数とみてよいことになる。

つまり、以下のように表すこともできる。

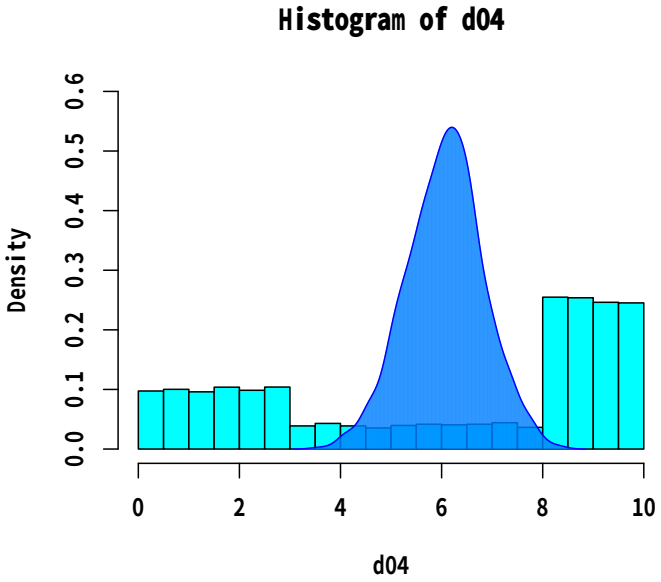
$$\bar{x} \sim N(\mu, \sigma/\sqrt{n})$$

中心極限定理の意味を考える

この話は、標本規模 n が大きければ・・・という前提だが、実際やってみると、標本規模 5 個の標本平均の分布ですら、それほど標準正規分布と違わないことがわかる。

ちょっとこれはすごいことで、もとの分布は、以下のような、なんとも言い表しがたい、だいぶ変な分布だったのに、その標本平均がきれいな正規分布となるというのは想像できただろうか？

```
> hist(d04,prob=T,col=5,ylim=c(0,.6))
> lines(density(m20),col=rgb(0,.5,1,alpha=.5),type="h")
> lines(density(m20),col=4)
```



青色の曲線は、標本規模 20 の標本平均の分布。

中心極限定理が成り立つ条件

中心極限定理は、母集団の分布の形に何も制約がない！

ただし、注意点としては、２つほど満たさなければならない条件がある。

２つとも、「そりゃそうだろう」という程度の仮定だが・・・

独立性の仮定

ひとつは、標本が無作為に取られていること。標本どおしがなんらかの関連性をもってとられたものであってはいけない。

ひとつ標本をとったら小さかったので、次は大きめのを狙ってとる・・・なんてことはしてはいけない。

同一分布の仮定

もうひとつは、同じ母集団からとられたものでなければならない。

交通量調査を、途中まで東京でやっていたのに、途中から北海道に変えました、なんてことはしてはいけない。

母平均の推定

n 個の標本で計算した標本平均は、それが母平均と同じかどうかはわからないが、それは上の正規分布に従うので、95 %の確率で、母平均プラスマイナス標準偏差の 1.96 倍の範囲にその値があるはずだ。

$$\mu - 1.96\frac{\sigma}{\sqrt{n}} \leq \bar{x} \leq \mu + 1.96\frac{\sigma}{\sqrt{n}}$$

ということは、逆に考えると、n 個の標本の標本平均 $\bar{x}$ が得られた場合、95 %の確率で、母平均 μ がそのプラスマイナス $\sigma/\sqrt{n}$ の範囲にあると言えることになる。

$$\bar{x} - 1.96\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{x} + 1.96\frac{\sigma}{\sqrt{n}}$$

ここで、σ はどうなんだ？ 実際にはわからないだろ？ ということになるが、ここはひとまず考えない。（t 分布のところで出てくる）

母平均の推定を確かめてみる

たとえば、20 個標本を１パターンだけとってみる　（、、、というより、実際の調査は１回きりが普通のことだけど。）

```
> (d04s_20<-sample(d04,20))
```

```
 [1] 2.0874659 3.0527021 9.4748927 4.7914142 9.8294067 2.1819908 8.8957769
 [8] 2.0110015 9.3427725 0.1042681 2.8564764 1.3357621 9.3139539 6.8110555
[15] 7.8846054 9.0501788 9.8843837 4.1053133 9.0156237 1.7811754
```

この標本の標本平均は、

```
> (m20_01<-mean(d04s_20))
```

```
[1] 5.690511
```

この場合、母平均 μ は 6.053 だから、この標本平均は母平均そのものではない！

しかし、中心極限定理によると、母平均 μ は、下限が

```
> m20_01+qnorm(.025)*(sd(d04)/sqrt(20))
```

```
[1] 4.213517
```

95 %の確率で上限が

```
> m20_01+qnorm(.975)*(sd(d04)/sqrt(20))
```

```
[1] 7.167505
```

の範囲に入っている、と 95 %の確からしさで言えるはず。

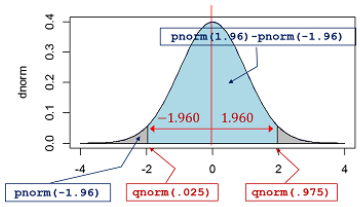
実際、この範囲に母平均 6.054 が入っている（標本抽出のパターンによっては、そうではない可能性も 5 %あるが・・・）。

*3

*3 qnorm 関数は、引数に確率 p= と平均 mean= 、標準偏差 sd= を与えることで、指定した平均と標準偏差の正規分布における指定した確率の確率変数の値を返す。

引数 mean= と sd= を省略すると、 mean=0 、 sd=1 の標準正規分布の場合の確率変数の値を返す。

つまり、 qnorm(.025) は、標準正規分布の小さい方から 2.5 %の点で -1.96 、標準正規分布 qnorm(.975) は大きい方から 2.5 %の点で 1.96 。ちなみに、上で求めた下限と上限は、以下でも求めることができる。



95 %下限が

```
> qnorm(.025,m20_01,sd(d04s_20)/sqrt(20))
```

```
[1] 4.14334
```

95 %下限が、

```
> qnorm(.975,m20_01,sd(d04s_20)/sqrt(20))
```

```
[1] 7.237682
```

なんでそうなるかは、正規分布の図を描いて、自分で確かめてみてください。

改めて中古車プリウス価格の代表値を読む

ということで、なんで平均値かということだが・・・

冒頭で、プリウスの中古車価格の分布として以下の代表値が示されていた。

データ数	平均	分散（標準偏差）
1000	145.3	805.5 (28.4)

1000 個もデータがあるが、それでも、あくまで数多ある中古プリウスのオークション価格の一部であって、標本規模 1000 の標本平均であり、標本分散である。

だから、この平均値が中古車プリウスの取引価格の平均値（母平均）だと思ったら間違いである。

95 %の確率で、母平均が以下の範囲にあると言えるはず。

（今回日は、 qnorm 関数の引数 p= をひとつずつ与えずに、いっぺんに与えたが、同じこと）

```
> qnorm(p=c(.025,.975),mean=145.3,sd=28.4/sqrt(1000))
```

```
[1] 143.5398 147.0602
```

1000 個もデータがあると、かなり狭い。

だから、表にある平均値が母平均だと思ったら間違いである、と言ったが、確かに間違いではあるが、「大きな間違い」ではない。

すなわち、上の表からわかることは、中古プリウスのオークション価格の分布は、正規分布に近いのか、それとも安いのと高いのの両極端に分布しているのか、どうなっているかわからないが、少なくとも平均値は、95 %の確率で 143.5〜147.1 の範囲にあるだろうということ。

4.4.6 標本分散の不偏性

ところで、標本分散はなぜ $n-1$ で割るのか？

ここでは証明はせず、実際に標本分散がどうなるか、やってみよう。

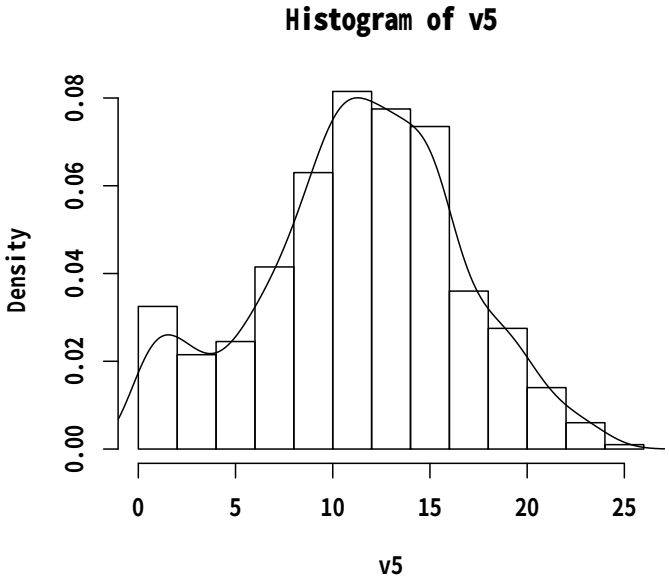
標本分散の分布

例のへんな分布 d04 から 5 個だけの標本を 1000 パターン取り出し、1000 個の標本分散を求める。

```
> v5<-rep(NA,1000)
> for(i in 1:1000) v5[i]<-var(sample(d04,5))
```

求めた 1000 個の標本分散の分布を描く。

```
> hist(v5,prob=T)
> lines(density(v5))
```



所詮 5 個しか標本をとっていなので、標本の取り方によっては、母分散とだいぶ違った標本分散となっている。

しかし、この標本分散の分布の平均を求めてみると、

```
> mean(v5)
```

```
[1] 11.33491
```

母分散は 11.36 で、標本分散の分布の平均値と一致する。

```
> var(d04)
```

```
[1] 11.35773
```

つまり、

$$V(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

は、母分散 σ^2 の不偏推定量で、

$$E[V(x)] = \sigma^2$$

であることが確認できた。

母分散を計算する関数をつくる

標本分散を母分散と同じように計算したらどうなるか？

$$V_0(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

母集団のデータが得られることはほとんどないが、R にはそもそも母分散を計算する関数が用意されていない。

とはいえ、標本分散に $(n-1)/n$ を掛けるだけだから、簡単に作れる。var0 という名前にしよう。

```
> var0<-function(x) var(x)*(length(x)-1)/(length(x))
```

ベクトル x を関数の引数として与えると、標本分散 var(x) と x の要素数 length(x) を使って母分散の計算を行う。

1,2,3 の分散を計算してみる。

標本分散として計算すると

```
> var(c(1,2,3))
```

```
[1] 1
```

母分散として計算すると

```
> var0(c(1,2,3))
```

```
[1] 0.6666667
```

ちゃんとできている。

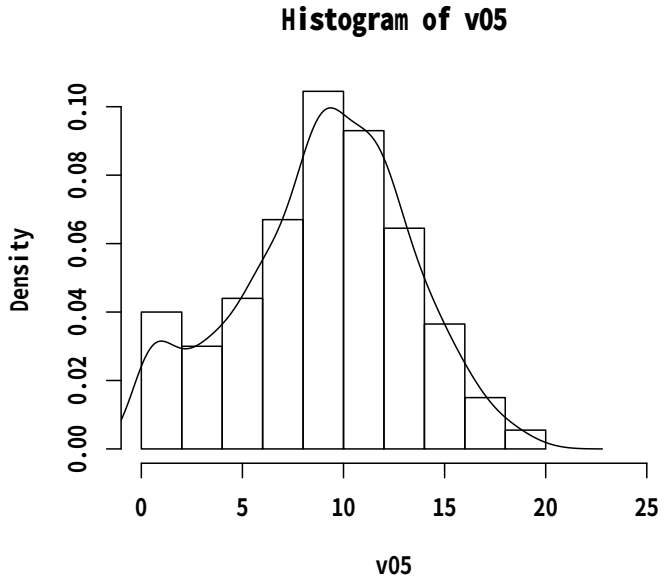
母分散として計算した標本分散の分布

さきほどと同様、 d04 から 5 個標本を 1000 パターンとり、1000 個の分散を計算する。ただし、そのときの分散は $n-1$ でなく、母平均同様 n で割った分散である。

```
> v05<-rep(NA,1000)
> for(i in 1:1000) v05[i]<-var0(sample(d04,5))
```

散布図を描いてみる。

```
> hist(v05,prob=T,xlim=c(0,25))
> lines(density(v05))
```



この分布の期待値は、

```
> mean(v05)
```

```
[1] 9.031514
```

母分散よりもだいぶ小さいことが確認できる。

つまり、標本分散として n で割ってしまうと、母分散を過小評価する、つまり、推定値のバラツキが実際よりも小さいと評価する可能性が高い。

他の標本規模の標本分散の分布を確認する

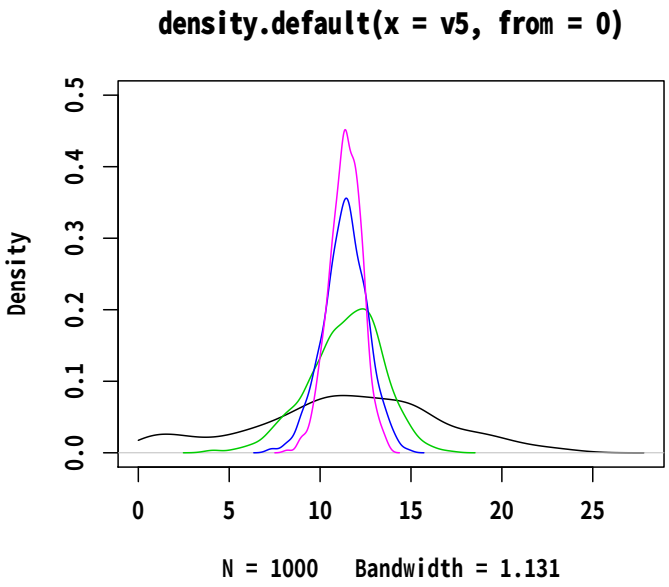
標本規模 20、50、100 の標本分散の分布も見てみよう。

まず、それぞれの規模の標本をとって、それぞれ 1000 個の標本分散を得る。

```
> v20<-rep(NA,1000)
> for(i in 1:1000) v20[i]<-var(sample(d04,20))
> v50<-rep(NA,1000)
> for(i in 1:1000) v50[i]<-var(sample(d04,50))
> v100<-rep(NA,1000)
> for(i in 1:1000) v100[i]<-var(sample(d04,100))
```

さきほど求めた標本規模 5 の標本分散の確率密度に、標本規模 20（緑線）、50（青線）、100（紫線）の確率密度を重ねてみる

```
> plot(density(v5,from=0),ylim=c(0,.5))
> lines(density(v20),col=3)
> lines(density(v50),col=4)
> lines(density(v100),col=6)
```



標本規模が大きくなるに従い、分布が母分散のまわりに集まっている。

その分散は、標本分散の場合、標本平均のように単純な数式ではないが、とりあえず、標本規模 n が大きくなると、標本分散の分散は小さくなるとだけ、覚えておいてほしい。

4.5 正規分布系の分布

補足として、統計分析で頻繁に使う正規分布系の分布、カイ二乗分布、t 分布、F 分布について解説しておく。

4.5.1 正規分布の和

正規分布に従う確率変数を足すと、正規分布となる。

確率変数 x_1 が以下の独立同一の正規分布に従うとする。

$$x_1 \sim N(\mu_1, \sigma_1^2)$$

別の確率変数 x_2 が以下の独立同一の正規分布に従うとする。

$$x_2 \sim N(\mu_2, \sigma_2^2)$$

このとき、確率変数 $x_1 + x_2$ は、以下の正規分布に従う。

$$x_1 + x_2 \sim N(\mu_1 + \mu_2, \sigma^2 + \sigma^2)$$

正規分布に従う確率変数

平均-2、分散 1（標準偏差 1）の正規分布に独立に従う確率変数を 1000 個つくって x1 に代入する。

```
> x1<-rnorm(1000,-2,1)
```

x1 の最初のいくつかを表示してみる。

```
> head(x1)
```

```
[1] -2.739299 -2.103693 -3.046138 -3.389371 -3.451722 -1.848741
```

こんな感じ。

平均 5、分散 4（標準偏差 2）の正規分布に従う確率変数を 1000 個つくって x2 に代入する。

```
> x2<-rnorm(1000,5,2)
```

x2 の最初のいくつかを表示してみる。

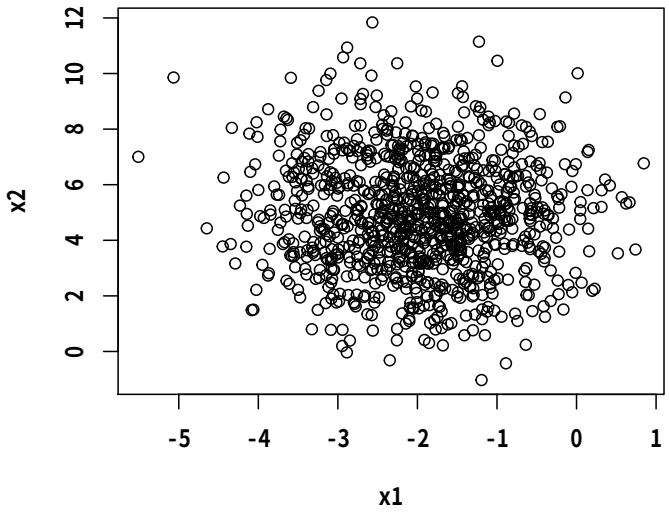
```
> head(x2)
```

```
[1] 3.921245 1.179840 7.270239 3.402171 5.612128 1.666402
```

こんな感じ。

２つをプロットしてみる。

```
> plot(x1,x2)
```

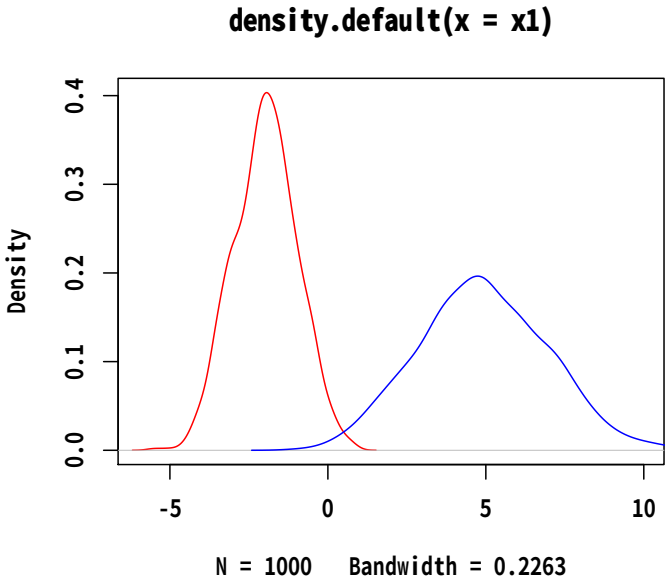


こんな感じ。

x1 と x2 も独立で、お互い関連性が見られない。独立ではなかったら、どっちかに分布が傾く。

密度関数を重ね描きしてみる。

```
> plot(density(x1),col=2,xlim=c(-6,10))
> lines(density(x2),col=4)
```



正規分布の和の分布

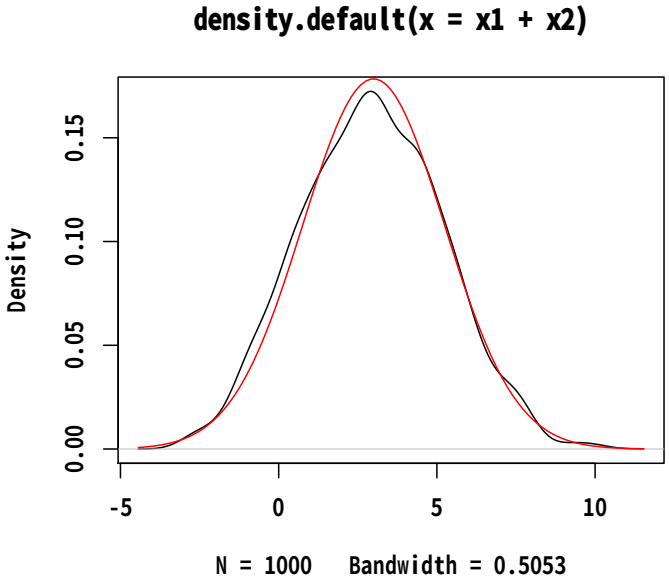
二つの確率変数の和の分布を求める。

これに、平均-2+5 分散 1+4 の正規分布の密度関数を重ね描きしてみる（赤線）。

ただし、正規分布の密度関数 dnorm で、標準正規分布以外を描こうとしたら、引数に平均と標準偏差を与えなければならない。curve 関数は、引数が１つだけの関数しか受けつかないから、 dnorm に平均と標準偏差をあらかじめ与えた関数を自分でつくる必要がある。ここでは f01n という名前で作成した。

ここで function(x) dnorm(x,-2+5,sqrt(1+4)) として関数を定義してもよいのだが、別の値も入れたいときがあるので、以下のように m01 、 sd01 としておけば、これを書き換えることで、関数 f01n がそのまま使える。

```
> plot(density(x1+x2))
> m01<--2+5
> sd01<-sqrt(1+4)
> f01n<-function(x) dnorm(x,m01,sd01)
> curve(f01n,add=T,col=2)
```



平均値は

```
> mean(x1+x2)
```

```
[1] 2.933972
```

２つの確率変数の平均の和 (-2+5) となっている。

分散は

```
> var(x1+x2)
```

```
[1] 4.995463
```

ふたつの確率変数の平均の和（1+4）となっている。

4.5.2 カイ二乗分布

標準正規分布に従う独立の確率変数 $x_1, ..., x_k$ の 2 乗の和は、自由度 k のカイ二乗分布に従う。

$$x_1^2 + ... + x_k^2 \sim \chi(k)$$

自由度１のカイ二乗分布

確率変数 x_1 が標準正規分布に従うとする。

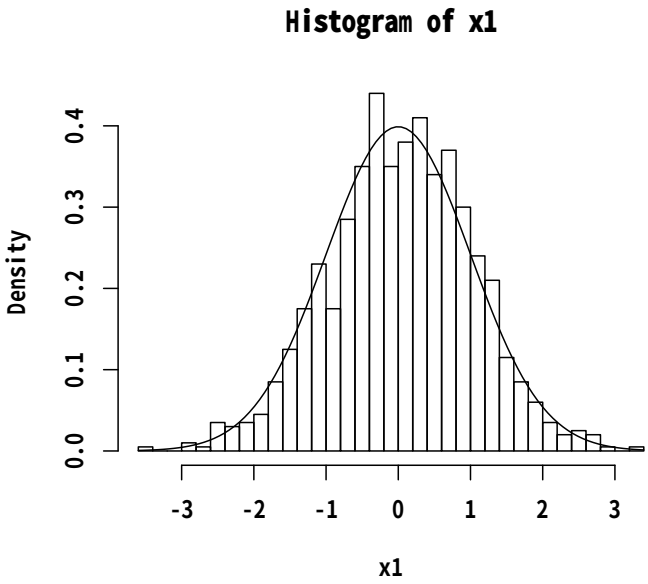
$$x_1 \sim N(0, 1)$$

R で、標準正規分布に従う乱数を 1000 個発生させる。

```
> x1<-rnorm(1000)
```

グラフを描いてみる。

```
> hist(x1,prob=T,br="FD")
> curve(dnorm,add=T)
```



x1 の最初の方をいくつか見てみると、

```
> head(x1)
```

```
[1] -2.0151184  0.8662669 -1.0419734 -2.0784037  0.1507166 -0.6618765
```

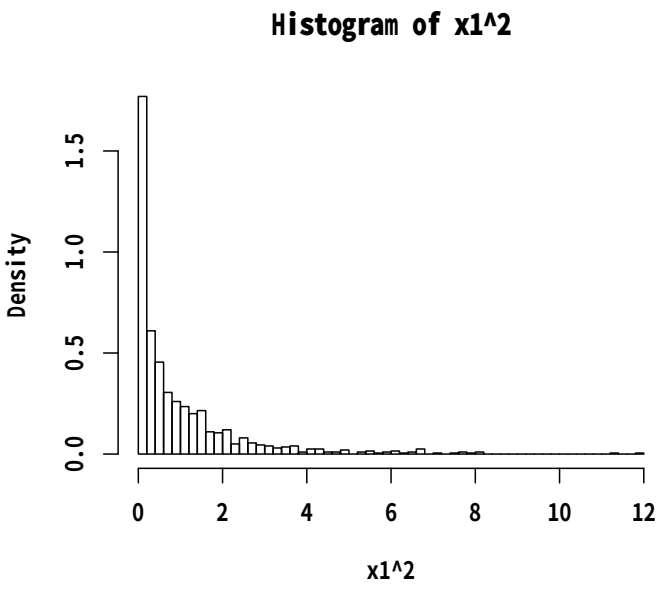
x1^2 とするとそれぞれの値を二乗してくれる。

```
> head(x1^2)
```

```
[1] 4.06070201 0.75041833 1.08570864 4.31976183 0.02271548 0.43808048
```

x^2 の分布を見てみる。

```
> hist(x1^2,prob=T,br="FD")
```



二乗しているから、当然 0 以上となる。

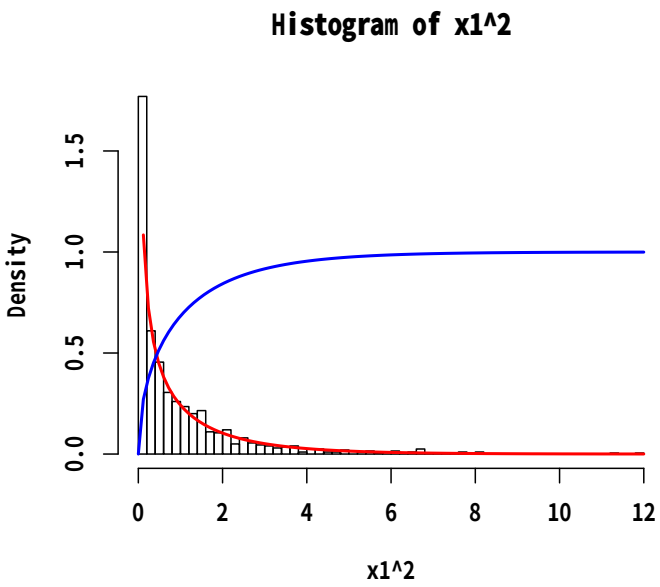
これが自由度１のカイ二乗分。

R では、（累積）確率分布は、pchisq(x,1) で計算できる。確率密度は dchisq(x,1) である。

さきほどのヒストグラムに重ね描きしてみよう。

pchisq 関数も dchisq 関数も、引数を２つとるので、引数がひとつだけを前提としている curve 関数にそのまま入れられない。一旦、 function 関数を使って、別の関数として定義して使う。ここでは、確率分布を f01p 、確率密度を f01d という名前で関数を作成した。 df01<-1 としてあるが、 df01<-2 と書き換えると、自由度 2 のカイ二乗分布にも使える。

```
> df01<-1
> f01p<-function(x) pchisq(x,df01)
> f01d<-function(x) dchisq(x,df01)
> hist(x1^2,prob=T,br="FD")
> curve(f01d,col=2,lwd=2,add=T,from=0)
> curve(f01p,col=4,lwd=2,add=T)
```



自由度 1 のカイ二乗分布の密度関数（赤線）は、0 を頂点として、右下がりの曲線として描かれる。

自由度２のカイ二乗分布

確率変数 x_1 と x_2 が独立に標準正規分布に従うとする。

$$x_1, x_2 \sim N(0,1)$$

このとき、確率変数 $x_1^2 + x_2^2$ は自由度 2 のカイ二乗分布に従う。

$$x_1^2 + x_2^2 \sim \chi^2(2)$$

x1 に 1000 個の標準正規乱数、 x2 にも 1000 個の標準正規乱数を、それぞれ独立に発生させて代入。

```
> x1<-rnorm(1000)
> x2<-rnorm(1000)
```

2 つの乱数最初のちょっとだけ表示してみる。

```
> head(x1)
```

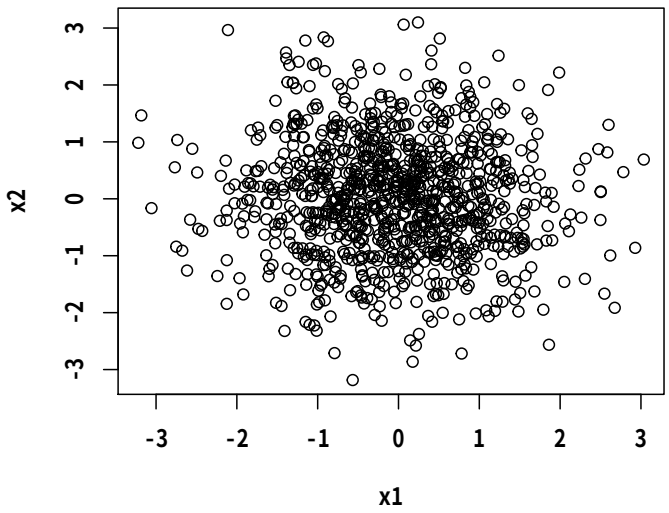
```
[1] -0.2905228  0.5205546 -0.1446555 -0.6308890  0.5402412 -1.9202113
```

```
> head(x2)
```

```
[1]  1.5068908  0.4921600  0.1577565  0.1988853 -0.5754961 -1.6803314
```

散布図にプロットしてみる。

```
> plot(x1,x2)
```



こんな感じ。

$$x_1^2 + x_2^2$$

を計算する。

```
> c2<-x1^2+x2^2
```

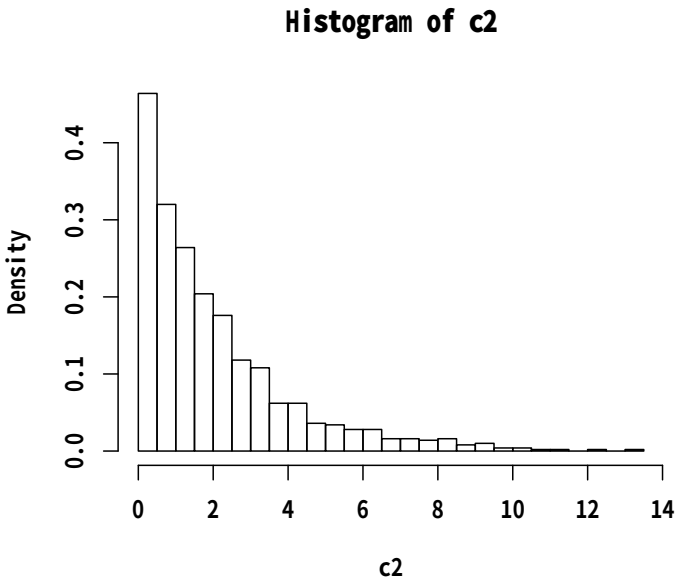
最初のちょっとだけ表示してみる。

```
> head(c2)
```

```
[1] 2.35512349 0.51319863 0.04581234 0.43757631 0.62305637 6.51072539
```

ヒストグラムを描いてみる。

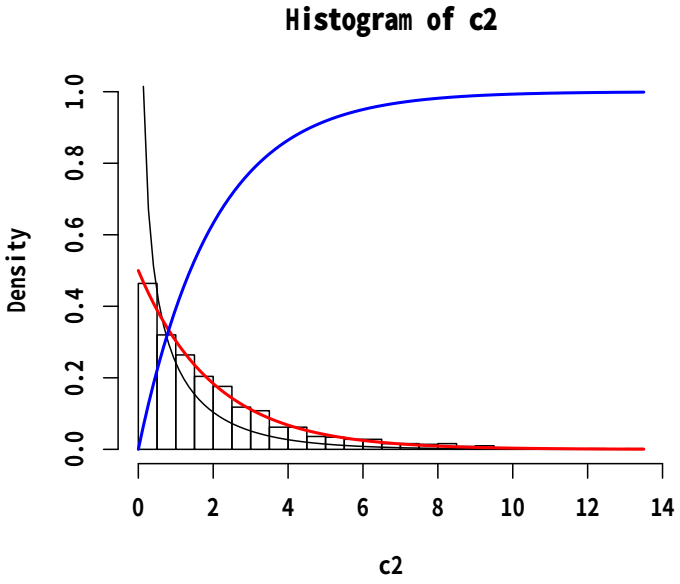
```
> hist(c2,prob=T,br="FD")
```



これは自由度 2 のカイ二乗分布。

さきほどつくった自由度 df01 のカイ二乗分布を描く関数 f01p と f01d を使って、自由度２のカイ二乗分布の（累積）確率分布と確率密度を重ね描きしてみる。

```
> hist(c2,prob=T,br="FD",ylim=c(0,1))
> df01<-1
> curve(f01d,add=T,from=0)
> df01<-2
> curve(f01d,add=T,lwd=2,col=2,from=0)
> curve(f01p,add=T,lwd=2,col=4)
```



自由度２のカイ二乗分布の密度関数は、自由度１のカイ二乗分布（黒線）と同じく、0 を頂点とするが、0 近辺でそれよりもなだらか曲線となる。

自由度 5 のカイ二乗分布

５つの確率変数 x_1、x_2、x_3、x_4、x_5 が独立に標準正規分布に従うとする。

$$x_1, x_2, x_3, x_4, x_5 \sim N(0,1)$$

このとき、$x_1, ..., x_5$ を二乗して足し合わせた確率変数は、自由度 5 のカイ二乗分布に従う

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \sim \chi(5)$$

x1,...,x5 の５つの確率変数について、それぞれ 1000 パターンの標準正規乱数を発生させる。さきほどと同じように、 x1<-rnorm(1000) 、 x2<-rnorm(1000) という具合に、一つずつ代入していくのもよいが、面倒なので、今回は、 5000 の正規乱数を発生させて、それを 1000×5 の行列にして使ってみる。

```
> d055<-matrix(rnorm(5000),1000,5)
> d055<-data.frame(d055)
> colnames(d055)<-c("x1","x2","x3","x4","x5")
```

最初のいくつかを表示してみる。

```
> head(d055)
```

```
           x1         x2         x3         x4         x5
1 -2.077095048 -0.3639033  0.5528105 -0.2208133 -0.6854299
2 -0.037621496 -1.1634653 -0.6647169 -0.2495090 -0.4896183
3  1.729456592  0.6643767  0.7846042 -0.2135772  0.2241917
4 -0.003366841 -0.2113594 -0.9185446  0.4034920 -0.8671635
5 -0.695553902 -1.9007569  1.2700449 -0.2353534 -0.8636268
6  0.299665610  1.7264421 -0.7896033  1.7401031  0.2507805
```

すべての値を二乗する

```
> d055sq<-d055^2
```

どうなったか、最初のいくつかを表示する。

```
> head(d055sq)
```

```
              x1         x2        x3         x4         x5
[1,] 4.314324e+00 0.13242564 0.3055994 0.04875852 0.46981419
[2,] 1.415377e-03 1.35365160 0.4418486 0.06225474 0.23972609
[3,] 2.991020e+00 0.44139642 0.6156037 0.04561522 0.05026191
[4,] 1.133562e-05 0.04467279 0.8437241 0.16280579 0.75197256
[5,] 4.837952e-01 3.61287679 1.6130140 0.05539123 0.74585122
[6,] 8.979948e-02 2.98060232 0.6234734 3.02795882 0.06289084
```

$x_1^2 + ... + x_5^2$ を求めるには、これを行ごとに足せばよい。

行ごとに足す、という関数は、 apply 関数というのを使って、２番目の引数 MARGIN=1 （行ごとにという意味になる）と３番目の引数 FUN=sum を与えればよい。

```
> c05<-apply(d055sq,1,sum)
```

これで、d055sq を行ごとに sum 関数を適用せよ、という意味になる。（ちなみに、２番目の引数 MARGIN= を、1 でなく、2 にすると列ごとに、そのあとの引数で指定した関数をあてはめてくれる。）

どうなったか、最初のいくつかを表示する。

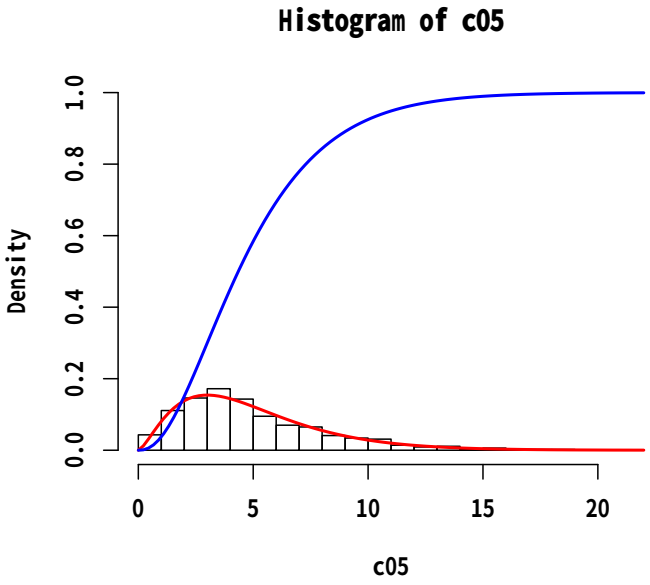
```
> head(c05)
```

```
[1] 5.270922 2.098896 4.143897 1.803187 6.510928 6.784725
```

ちゃんとできているはず。暇な人は確かめて。

　これは自由度 5 のカイ二乗分布に従う。

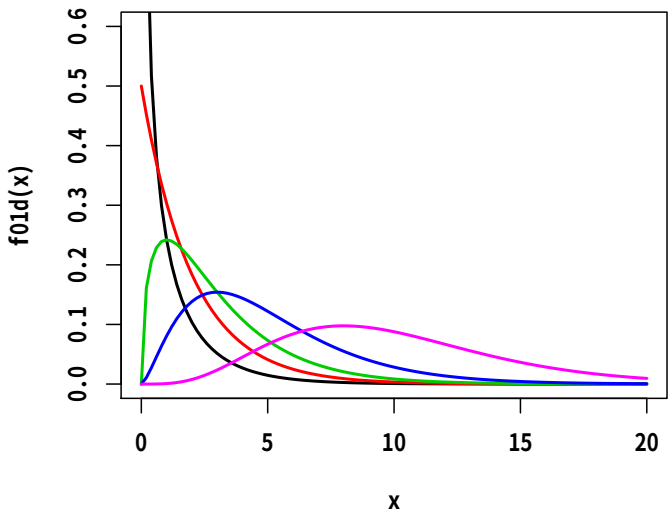
```
> hist(c05,prob=T,br="FD",ylim=c(0,1))
> df01<-5
> curve(f01d,col=2,lwd=2,add=T)
> curve(f01p,col=4,lwd=2,add=T)
```



いろいろな自由度のカイ二乗

　ちなみに、自由度 1、2、3、5、10 のカイ二乗分布（密度関数）を重ね描きしてみた。自由度 1 が黒、自由度 2 が赤、自由度 3 が緑、自由度 5 が青、自由度 10 が紫。

```
> df01<-1
> curve(f01d,col=1,lwd=2,from=0,to=20,ylim=c(0,.6))
> df01<-2
> curve(f01d,col=2,lwd=2,add=T)
> df01<-3
> curve(f01d,col=3,lwd=2,add=T)
> df01<-5
> curve(f01d,col=4,lwd=2,add=T)
> df01<-10
> curve(f01d,col=6,lwd=2,add=T)
```



一般の正規分布に従う確率変数

標準席分布ではなく、平均が 0 でない、分散も 1 ではない、普通の正規分布に従う確率変数を考える。

$x_1, ..., x_5$ の５つの確率変数が、独立同一に正規分布に従うとする。

$$x_1, ..., x_5 \sim N(\mu, \sigma^2)$$

この５つの確率変数を、その標本平均 $\bar{x} = (x_1 + ... + x_5)/5$ と母標準偏差 σ で正規化する。

$$z_1 = \frac{x_1 - \bar{x}}{\sigma}, ..., z_5 = \frac{x_5 - \bar{x}}{\sigma}$$

このとき、これら５つの確率変数の二乗和は、自由度 5-1 のカイ二乗分布に従う。

$$z_1^2 + ... + z_5^2 \sim \chi(5 - 1)$$

確かめてみよう。

平均 3、標準偏差 4 の正規分布に従う正規乱数 5000 個を作成し、1000 行× 5 列の行列にする。

```
> d055g<-matrix(rnorm(5000,3,4),1000,5)
```

頭の方だけ確認してみる。

```
> head(d055g)
```

```
          [,1]       [,2]       [,3]       [,4]       [,5]
[1,]  1.878546  1.0048777 -4.3441052 11.010028  3.2165330
[2,]  6.176959 -0.2843062  8.1885774 -1.841293  0.7014493
[3,] -5.090176 -2.8789973 -1.0893428  0.816928  1.8262983
[4,]  2.833086 -2.0504110 -0.9528485  2.149985  5.6154682
[5,]  3.255907  0.3723712 -4.6462265 -2.038576 -1.0674751
[6,] -8.306091  3.9838513  4.7944291  1.404928 -0.2829540
```

行ごとに標本平均をとる

```
> m055g<-apply(d055g,1,mean)
```

これも最初の方だけ確認する

```
> head(m055g)
```

```
[1]  2.5531758  2.5882773 -1.2830579  1.5190559 -0.8248000  0.3188327
```

もとのデータから、行ごとの標本平均を引き、母標準偏差 4 で割る。

行ごとに違った数値を引くという操作は、 sweep 関数でできる。２番目の引数に 1 とあるのは、「行ごとに」という意味。最初の引数にある行列またはデータフレームから３番目にあるベクトルを行ごとに引く。デフォルトは引き算だが、４番めの引数に、たとえば FUN=”+” とすると、行ごとに足すこともできる。

```
> z055<-sweep(d055g,1,m055g)/4
```

最初のいくつかを確認してみる

```
> head(z055)
```

```
           [,1]       [,2]        [,3]       [,4]        [,5]
[1,] -0.1686575 -0.3870745 -1.72432025  2.1142130  0.16583932
[2,]  0.8971704 -0.7181459  1.40007503 -1.1073926 -0.47170699
[3,] -0.9517795 -0.3989848  0.04842878  0.5249965  0.77733905
[4,]  0.3285074 -0.8923667 -0.61797610  0.1577323  1.02410308
[5,]  1.0201766  0.2992928 -0.95535664 -0.3034440 -0.06066877
[6,] -2.1562310  0.9162546  1.11889910  0.2715239 -0.15044667
```

ちゃんと正規化できている？

標本平均と母標準偏差で正規化されたデータをすべて二乗する。

```
> z055sq<-z055^2
> head(z055sq)
```

```
           [,1]       [,2]        [,3]       [,4]       [,5]
[1,] 0.02844536 0.14982669 2.973280317 4.46989655 0.02750268
[2,] 0.80491473 0.51573349 1.960210092 1.22631832 0.22250748
[3,] 0.90588416 0.15918890 0.002345347 0.27562130 0.60425600
[4,] 0.10791713 0.79631835 0.381894458 0.02487948 1.04878711
[5,] 1.04076034 0.08957618 0.912706303 0.09207827 0.00368070
[6,] 4.64933193 0.83952257 1.251935197 0.07372522 0.02263420
```

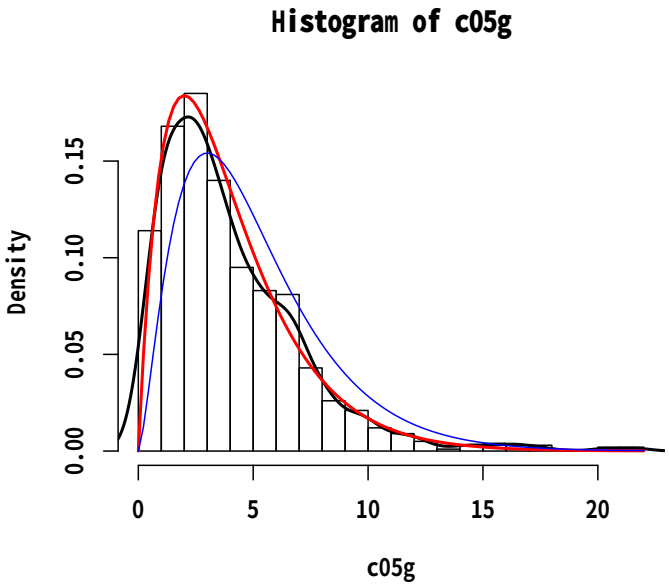
行ごとの合計をとる

```
> c05g<-apply(z055sq,1,sum)
> head(c05g)
```

```
[1] 7.648952 4.729684 1.947296 2.359797 2.138802 6.837149
```

ヒストグラムを描き、確率密度を推定し（黒線）、それに自由度 4 のカイ二乗分布の密度関数（赤線）を重ね描きしてみる。ちなみに、青い線は、自由度 5 のカイ二乗分布。

```
> hist(c05g,prob=T,br="FD")
> lines(density(c05g),lwd=2)
> df01<-4
> curve(f01d,add=T,col=2,lwd=2)
> df01<-5
> curve(f01d,add=T,col=4)
```



基準化するのに、標本平均を使っているため、$z_1, ..., z_5$ の５つの確率変数のうち、ひとつは他の４つで拘束されている（たとえば、$z_5 = -(z_1 + ... + z_4)$。

・・・しかし、この計算では、母標準偏差 σ を使っていた。これは普通はわからないから、標本分散を使わざるをえないのでではないのか？

このあたりは、次の t 分布で議論しよう。

4.5.3 t 分布

t 分布は、スチューデント分布、スチューデントの t 分布とも言う。

x_0 が標準正規分布に従う確率変数だとする。

$$x_0 \sim N(0,1)$$

$> x_1, ..., x_5$ も独立に標準正規分布に従う確率変数だとする。

$$x_1, ..., x_5 \sim N(0,1)$$

このとき、次の確率変数は、自由度 5 の t 分布に従う。

$$\frac{x_0}{\sqrt{\frac{x_1^2+...+x_5^2}{5}}}$$

t 分布の形

t 分布の密度関数の定義はちゃんとあるが、ガンマ関数から説明しなければならず、面倒なので、ここは形だけ。

t 分布は標準正規分布に似ているが、標準正規分布よりもすそ野が広い。

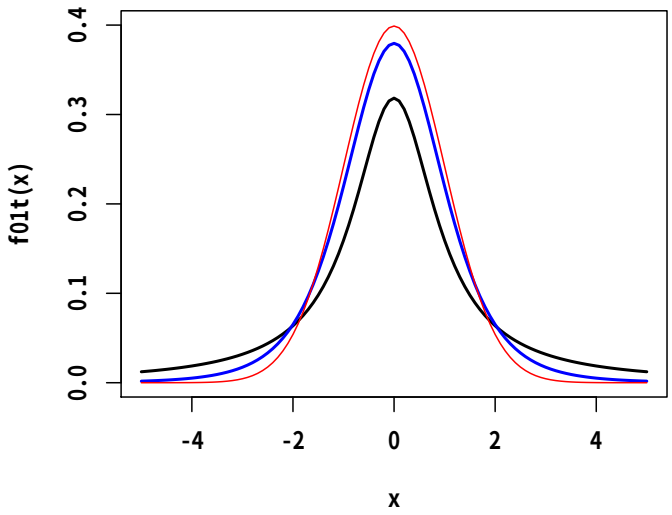
t 分布は自由度を持っていて、自由度が大きくなると、標準正規分布に近づく。

自由度 1 の t 分布を描いてみる。

R で t 分布の密度関数は dt 関数。引数を２つとるので、１つの引数しか許してくれない curve 関数でグラフを描こうとしたら、１つしか引数をとらない関数として再定義しなければならない。ここでは f01t という名前で再定義した。

黒線が自由度 1 の t 分布、青線が自由度 10 の t 分布。赤線は標準正規分布。

```
> #自由度 1 の t 分布
> df01<-1
> f01t<-function(x) dt(x,df01)
> curve(f01t,-5,5,lwd=2,ylim=c(0,.4))
> #自由度 5 の t 分布
> df01<-5
> curve(f01t,lwd=2,col=4,add=T)
> curve(dnorm,add=T,col=2)
```



標準正規分布から t 分布

標準正規分布に従う乱数を 1000 個作成し、x0 に代入

```
> x0<-rnorm(1000)
```

$x_1^2 + ... + x_5^2$ を 1000 個作成したいが、これはすでにカイ二乗分布のところで作成していたので、これを使いまわす。

```
> head(c05)
```

```
[1] 5.270922 2.098896 4.143897 1.803187 6.510928 6.784725
```

x0 を c05/5 の平方根で割って、 t05 に代入する。

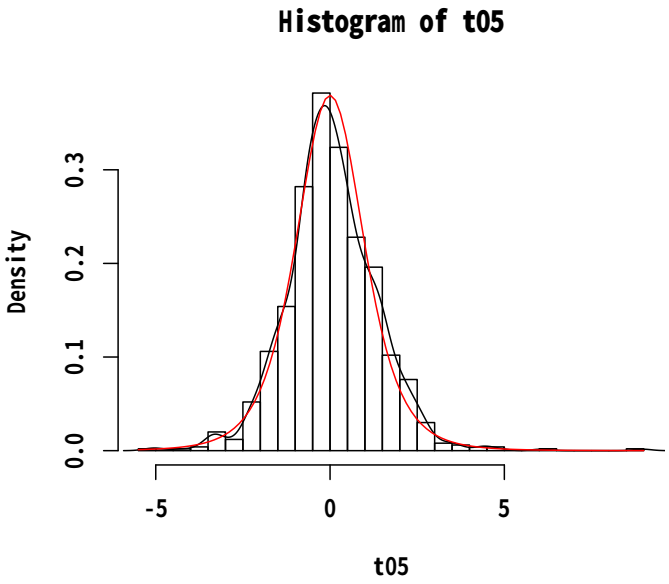
```
> t05<-x0/sqrt(c05/5)
```

t05 の分布を描く。

ヒストグラムに確率密度を重ね描きする（黒線）。

さらに、自由度 5 の t 分布の密度関数を赤線で描く。t 分布の密度関数 dt は、引数を 2 つとるので、１つの引数をとる関数しか受け付けない curve 関数の中で使うには、関数 dt の引数のひとつである自由度をあらかじめ与えた形で、再定義する。この関数を f01t と名付けよう。

```
> hist(t05,prob=T,br="FD") #ヒストグラム
> lines(density(t05)) #密度関数
> df01<-5 #自由度
> f01t<-function(x) dt(x,df01) #自由度をあらかじめ与えた t 分布の密度関数を定義
> curve(f01t,add=T,col=2) #自由度 df01 の密度関数を重ね描き。色は赤。
```



t05 の確率密度は、自由度 5 の確率分布にきれいに一致している。

標本平均の分布

標準席分布ではなく、平均が 0 でない、分散も 1 ではない、普通の正規分布に従う確率変数を考える。

$x_1, ..., x_5$ の５つの独立同一に正規分布に従うとする。

$$x_1, ..., x_5 \sim N(\mu, \sigma^2)$$

この５つの確率変数の標本平均

$$\bar{x} = \frac{x_1 + ... + x_5}{5}$$

は、中心極限定理により、

$$\frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{5}}} \sim N(0, 1)$$

この中の母分散 σ^2 を標本分散

$$s^2 = \frac{(x_1 - \bar{x})^2 + ... + (x_5 - \bar{x})^2}{5 - 1}$$

で置き換えた値

$$t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{5}}}$$

は、自由度 5-1 の t 分布に従う。

これは、以下のように変形して確かめることができる。

$$t = \frac{\frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{5}}}}{\sqrt{\frac{\left(\frac{x_1 - \bar{x}}{\sigma}\right)^2 + ... + \left(\frac{x_5 - \bar{x}}{\sigma}\right)^2}{5-1}}}$$

このうち、分子は標準正規分布に従う。

$$\frac{\bar{x}-\mu}{\frac{\sigma}{\sqrt{5}}} \sim N(0,1)$$

また、分子の

$$\frac{x_1-\bar{x}}{\sigma},...,\frac{x_5-\bar{x}}{\sigma}$$

もそれぞれ、標準正規分布に従う。

したがって、この値が、t 分布に従うことがわかる。

ただし、分母は、$\bar{x}$ を使っており、5 つのうち 1 つは、他の 4 つに縛られているので、自由度は 5-1 である。

つまり、母平均がわかっている、またはある値に固定できる場合、標本規模 n の標本平均を、標本平均の標本標準偏差で割った値は、自由度 $n-1$ の t 分布に従う。

確かめてみよう！

カイ二乗分布のところで、平均 3、標準偏差 4 の正規分布に従う正規乱数 5000 個を作成して、1000 行× 5 列の行列とした。これを使おう。

```
> head(d055g)
```

```
          [,1]       [,2]       [,3]       [,4]       [,5]
[1,]  1.878546  1.0048777 -4.3441052 11.010028  3.2165330
[2,]  6.176959 -0.2843062  8.1885774 -1.841293  0.7014493
[3,] -5.090176 -2.8789973 -1.0893428  0.816928  1.8262983
[4,]  2.833086 -2.0504110 -0.9528485  2.149985  5.6154682
[5,]  3.255907  0.3723712 -4.6462265 -2.038576 -1.0674751
[6,] -8.306091  3.9838513  4.7944291  1.404928 -0.2829540
```

各行の標本平均 $\bar{x}$ も計算済みだった。

```
> head(m055g)
```

```
[1]  2.5531758  2.5882773 -1.2830579  1.5190559 -0.8248000  0.3188327
```

各行の標本分散 $s^2=((x_1-\bar{x})^2+...+(x_5-\bar{x})^2)/(5-1)$ を計算する。

```
> s205<-apply(d055g,1,var)
> head(s205)
```

```
[1] 30.595806 18.918736  7.789183  9.439186  8.555207 27.348596
```

t 値 $t=(\bar{x}-\mu)/(s/\sqrt{n})$ を計算する。

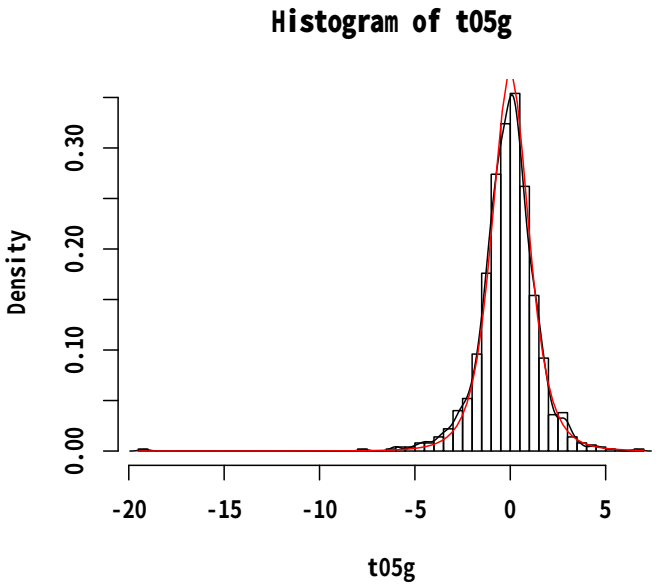
```
> t05g<-(m055g-3)/(sqrt(s205/5))
```

プロットする。

黒線が、発生させた正規乱数から計算した t 値の分布。

赤線が、自由度 4 の t 分布。

```
> hist(t05g,prob=T,br="FD")
> lines(density(t05g))
> df01<-4
> curve(f01t,add=T,col=2)
```



4.5.4　F 分布

5 つの確率変数 $x_1, ..., x_5$ が独立に標準正規分布に従うとする。

$$x_1, ..., x_5 \sim N(0,1)$$

＞ 同じく、3 つの確率変数 $x_6, ..., x_8$ も独立に標準正規分布に従うとする。

$$x_6, x_7, x_8 \sim N(0,1)$$

このとき、次の確率変数は自由度 (5,3) の F 分布に従う

$$F = \frac{\frac{x_1^2+...+x_5^2}{5}}{\frac{x_6^2+...+x_8^2}{3}} \sim F(5,3)$$

確かめてみよう！

標準正規分布を 1000 行× 5 列作成

```
> d055<-matrix(rnorm(5000),1000,5)
> head(d055)
```

```
           [,1]       [,2]        [,3]       [,4]       [,5]
[1,] -0.6807642  1.1225737  0.86046380 -2.2126601 -1.3774845
[2,]  1.5759340  0.3886105 -0.34794788  0.5021634 -1.0703697
[3,] -0.0429053 -0.2786727  0.68412372  0.4847711 -1.4477483
[4,]  0.8847623  0.7325835  0.30927546 -0.3737863  0.9985649
[5,] -0.5634098 -1.2319909  0.86669256 -0.6423988 -0.3519131
[6,] -0.4827641 -0.1106882 -0.09423499 -0.4868535  2.0651199
```

標準正規分布を 1000 行× 3 列作成

```
> d053<-matrix(rnorm(3000),1000,3)
> head(d053)
```

```
           [,1]       [,2]        [,3]
[1,] -0.0265765 -0.4097591 -1.62805457
[2,] -1.4727573 -0.5302286  0.88420114
```

```
[3,] -0.9328979 -0.4265692  1.24025597
[4,]  0.2116955  0.2007082  1.78544192
[5,] -0.8328774 -0.3960977 -0.53793777
[6,] -0.8422797 -0.6984917  0.04869549
```

それぞれを二乗する

```
> d055sq<-d055^2
> head(d055sq)
```

```
            [,1]       [,2]        [,3]      [,4]      [,5]
[1,] 0.463439951 1.26017164 0.740397957 4.8958645 1.8974636
[2,] 2.483567955 0.15101809 0.121067731 0.2521681 1.1456913
[3,] 0.001840865 0.07765848 0.468025271 0.2350030 2.0959753
[4,] 0.782804301 0.53667865 0.095651307 0.1397162 0.9971319
[5,] 0.317430653 1.51780162 0.751155996 0.4126762 0.1238428
[6,] 0.233061221 0.01225187 0.008880232 0.2370264 4.2647200
```

```
> d053sq<-d053^2
> head(d053sq)
```

```
             [,1]       [,2]        [,3]
[1,] 0.0007063106 0.16790252 2.650561692
[2,] 2.1690141995 0.28114236 0.781811655
[3,] 0.8702985081 0.18196129 1.538234877
[4,] 0.0448149715 0.04028377 3.187802846
[5,] 0.6936847559 0.15689337 0.289377044
[6,] 0.7094350390 0.48789070 0.002371251
```

各行の合計をとる

```
> c055<-apply(d055sq,1,sum)
> head(c055)
```

```
[1] 9.257338 4.153513 2.878503 2.551982 3.122907 4.755940
```

```
> c053<-apply(d053sq,1,sum)
> head(c053)
```

```
[1] 2.819171 3.231968 2.590495 3.272902 1.139955 1.199697
```

f 値を計算

```
> fv53<-(c055/5)/(c053/3)
> head(fv53)
```

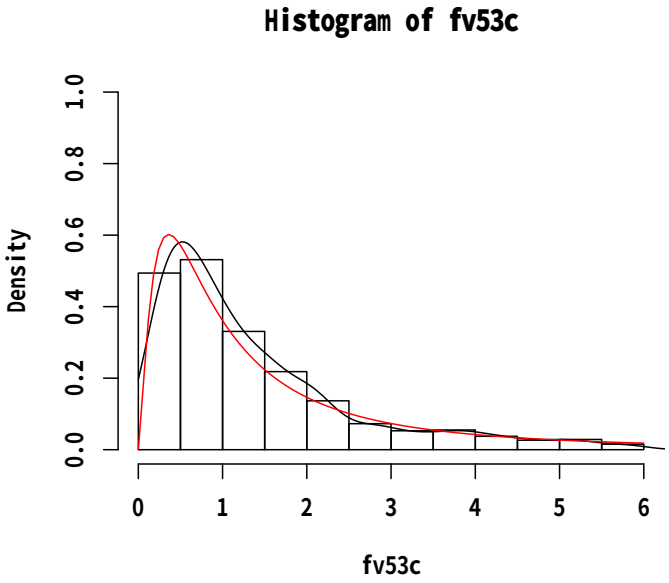
```
[1] 1.9702258 0.7710806 0.6667073 0.4678385 1.6437001 2.3785705
```

ヒストグラム、確率密度、それに自由度 (5,3) の密度関数を重ね描きする。F 分布の密度関数 df は、引数に確率変数の他に、２つの自由度を要求するので、ひとつだけを引数にとる関数しか受け付けない curve 関数の中で使うには、関数の再定義が必要。ここでは、自由度を最初から df01 と df02 に固定した f01f という名前で定義した。

この f 値は、二乗の割り算だなので、たまにだが、分母がたまたま極端に小さい値をとると、ものすごく大きな値になることがある。

あまり値の幅が広いと、よく見たい範囲 0〜4 近辺の確率密度の推定が雑になるので、計算した f 値を 6 未満で切って表示する。

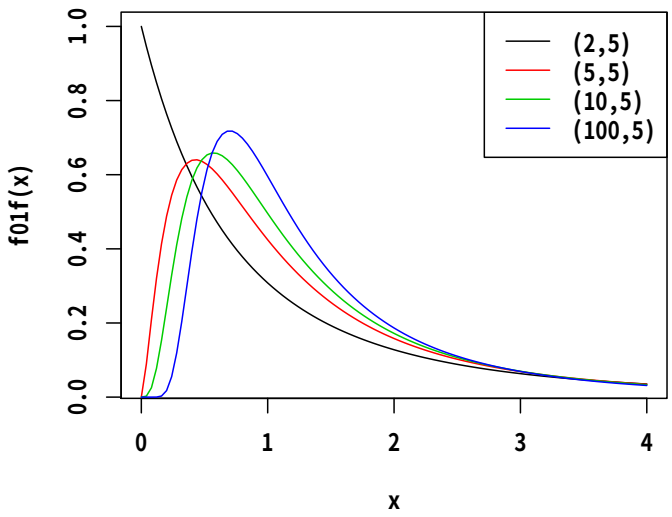
```
> fv53c<-fv53[fv53<6]
> hist(fv53c,prob=T,xlim=c(0,6),ylim=c(0,1))
> lines(density(fv53c,from=0))
> df01<-5
> df02<-3
> f01f<-function(x) df(x,df01,df02)
> curve(f01f,add=T,col=2)
```



いろいろな自由度の F 分布

自由度を分母の方は 5 のまま、分子の方だけ動かして $(2,5),(5,5),(10,5),(100,5)$ の 4 タイプの F 分布を描いてみる。

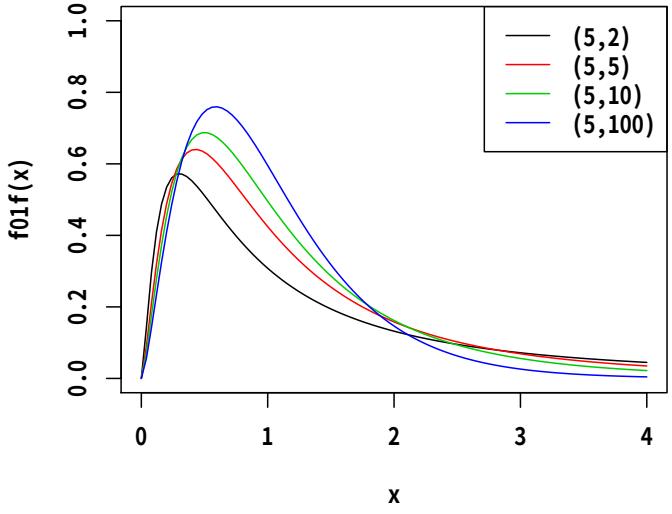
```
> df01<-2
> df02<-5
> curve(f01f,0,4)
> df01<-5
> curve(f01f,add=T,col=2)
> df01<-10
> curve(f01f,add=T,col=3)
> df01<-100
> curve(f01f,add=T,col=4)
> legend("topright",legend=c("(2,5)","(5,5)","(10,5)","(100,5)"),col=1:4,lty=1)
```



スケールは違うが、自由度 2,5,10,100 のカイ二乗分布の横軸のスケールを変えたよううな形になる。

今度は、分子は 5 のまま、分母の方を動かして、自由度 (5,2),(5,5),(5,10),(5,100) の４タイプの F 分布を描いてみる。

```
> df01<-5
> df02<-2
> curve(f01f,0,4,ylim=c(0,1))
> df02<-5
> curve(f01f,add=T,col=2)
> df02<-10
> curve(f01f,add=T,col=3)
> df02<-100
> curve(f01f,add=T,col=4)
> legend("topright",legend=c("(5,2)","(5,5)","(5,10)","(5,100)"),col=1:4,lty=1)
```



自由度 5 のカイ二乗分布の横軸のスケールが、分母の自由度に従い動いたような形となる。

標本分散の比

自由度の違う標本分散の比は、そのまま F 分布。ただし、母分散は同じ必要がある。

$x_1, ..., x_5$ の５つの確率変数が独立同一に正規分布に従うとする。

$$x_1, ..., x_5 \sim N(\mu_a, \sigma^2)$$

$x_1, ..., x_3$ の３つの確率変数が独立同一に正規分布に従うとする。

$$x_1, ..., x_3 \sim N(\mu_b, \sigma^2)$$

このとき、それぞれの標本平均を

$$\bar{x}_a = \frac{x_1 + ... + x_5}{5}$$

$$\bar{x}_b = \frac{x_1 + ... + x_3}{3}$$

と表すと、それぞれの標本分散の比は、以下のように表され、この値は、自由度 (5-1,3-1) の F 分布に従う。

$$f = \frac{\frac{(x_1-\bar{x}_a)^2+...+(x_5-\bar{x}_a)^2}{5-1}}{\frac{(x_1-\bar{x}_b)^2+...+(x_3-\bar{x}_b)^2}{3-1}} \sim F(5-1, 3-1)$$

このことは、分母と分子を母分散 σ^2 で割ることで確認できる。

$$f = \frac{\frac{\left(\frac{x_1-\bar{x}_a}{\sigma}\right)^2+...+\left(\frac{x_5-\bar{x}_a}{\sigma}\right)^2}{5-1}}{\frac{\left(\frac{x_1-\bar{x}_b}{\sigma}\right)^2+...+\left(\frac{x_3-\bar{x}_b}{\sigma}\right)^2}{3-1}}$$

確かめてみよう！

平均 3、分散 4 の正規分布を 1000 行× 5 列作成

```
> d055g<-matrix(rnorm(5000,3,2),1000,5)
> head(d055g)
```

```
         [,1]      [,2]        [,3]      [,4]      [,5]
[1,] 4.512986 0.1071208  3.02730623 0.8520786 4.5130745
[2,] 5.550292 4.5899306 -0.03748782 5.8281882 5.1693095
[3,] 6.964820 4.8906035  1.79795354 4.8190973 3.0387409
[4,] 3.797110 3.6391032  4.46154467 4.3196441 0.8230425
[5,] 2.462829 2.7075302  3.31576321 1.8970748 1.2881322
[6,] 5.371645 3.3547464  3.56912869 0.1290349 3.6619620
```

平均-2、分散 4 の正規分布を 1000 行× 3 列作成

```
> d053g<-matrix(rnorm(3000,-2,2),1000,3)
> head(d053g)
```

```
           [,1]       [,2]      [,3]
[1,] -1.7887182 -0.8163052 -1.887184
[2,] -2.4804559 -1.4881749  1.024241
[3,] -6.3146768  1.4081741 -4.106822
[4,] -3.4959489 -1.3189463 -2.333155
[5,] -2.1626032 -6.5683606  1.953076
[6,] -0.6384173 -4.1699310 -2.820220
```

それぞれの行ごとの標本分散をとる。

```
> v055g<-apply(d055g,1,var)
> head(v055g)
```

```
[1] 4.1929009 5.8800464 3.8926313 2.2069678 0.6011809 3.6297462
```

```
> v053g<-apply(d053g,1,var)
> head(v053g)
```

```
[1]  0.3503441  3.2632919 15.8220388  1.1866749 18.1607310  3.1755954
```

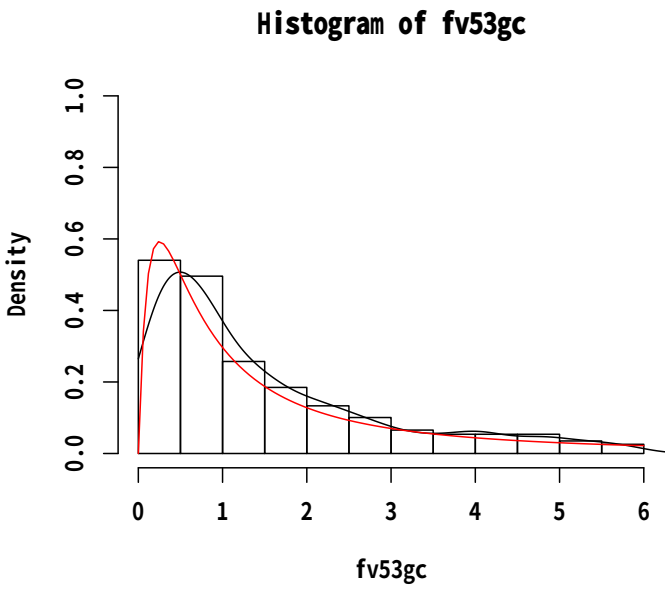
f 値を計算

```
> fv53g<-v055g/v053g
> head(fv53g)
```

```
[1] 11.96795219  1.80187570  0.24602590  1.85979142  0.03310334  1.14301280
```

ヒストグラム、確率密度、それに自由度 (5,3) の密度関数を重ね描きする。

```
> fv53gc<-fv53g[fv53g<6]
> hist(fv53gc,prob=T,xlim=c(0,6),ylim=c(0,1))
> lines(density(fv53gc,from=0))
> df01<-4
> df02<-2
> f01f<-function(x) df(x,df01,df02)
> curve(f01f,add=T,col=2)
```



正規分布に従う確率変数の標本平均と母平均との差を標本平均の標準偏差で割った値は、t 分布となるはずである。

・・・ということは、母平均を特定の値（たとえば 0）に仮定して、値標本平均とそれとの差を標本分散で割った値と、t 分布を比較して、そももそも、最初に仮定した母平均の値についての仮定（たとえば、「母平均が 0 である」）に無理があるかどうかを評価することができる。

また、２種類の正規分布に従う確率変数が、同じ母分散をもつならば、それらの標本分散の比の分布が F 分布となるはずである。

・・・ということは、ひとつだけ得られた標本分散の比と、この分布を比較して、その標本分散が同じ母分散の母集団から得られたものかどうか、その確率を計算できることになる。

このあたりは、次回の統計的検定で説明する。

第 5 章

推定と検定の考え方

5.1　統計的推定の考え方

5.1.1　経験的分布から推定する

経験的に母集団の分布がわかっていることがあります。

母集団の分布

リョウ君はゴルフが好きで、毎日練習しています。

リョウ君は毎日一生懸命練習しているので、リョウ君が現在どのくらいの精度で球を打てるかというデータはたくさんあります。

リョウ君が $100\mathrm{m}$ 先のピンを狙った 200 回の記録は以下です。

```
> d01<-c(97.6,91.9,103.2,88.8,109.9,115.4,105.5,96.4,102.1,97.9,100.5,101.6,89.4,
+ 99.5,100.5,105.2,98.3,100.6,105.5,106.2,94,100.1,98,98.1,94.7,99,
+ 97.7,97.9,105.9,104.5,97.9,93.3,98.7,104.5,91.6,101.7,94.5,102.2,105.1,
+ 101.3,96.6,89.5,100.8,102.6,101.3,100.3,95.8,104.7,99.9,99.6,95.3,91.2,
+ 99.4,97.8,103.6,107.5,100.5,95.4,103,103.1,100,99,101.5,98.1,103,
+ 89.3,91.4,97.2,107.1,95,101,93,100.8,97,93.7,92.6,104.9,96.9,
+ 93.6,98,110,102,101,97.6,92.4,96.7,91.1,97.3,97.6,97.1,99.3,
+ 103.5,107.2,101.3,97.9,97.6,97.5,93.7,91.8,106.6,105.1,101.3,103.1,109.3,
+ 101.6,113.5,101.4,114.1,99.2,112.8,89.4,109.2,93.6,92.8,102.7,105,91.3,
+ 99.8,103.3,97.4,103.7,97.8,101.6,100.1,105.9,97.1,103.3,103.3,105.8,89.5,
+ 99.9,104.5,105.4,104.3,97.7,101.4,99.5,93,101.1,99.3,94,98.1,100.9,
+ 101.9,102.1,93.7,98.1,101.9,97,95.4,105.5,104.3,110.1,103.5,102.9,98.3,
+ 96.1,92,106.6,106.9,88,103.2,103.5,91.3,100.3,106.7,103.4,100.3,100.6,
+ 110.9,94.8,97.9,97.2,103.8,94.8,101,102.6,100.9,99.7,93.7,105.9,100.8,
+ 98.5,97.7,91.7,90.1,101.2,95.6,103.1,93.4,104.5,102.7,108.1,100.2,102.6,
+ 92.6,94.9,92.2,104.1,102.1)
```

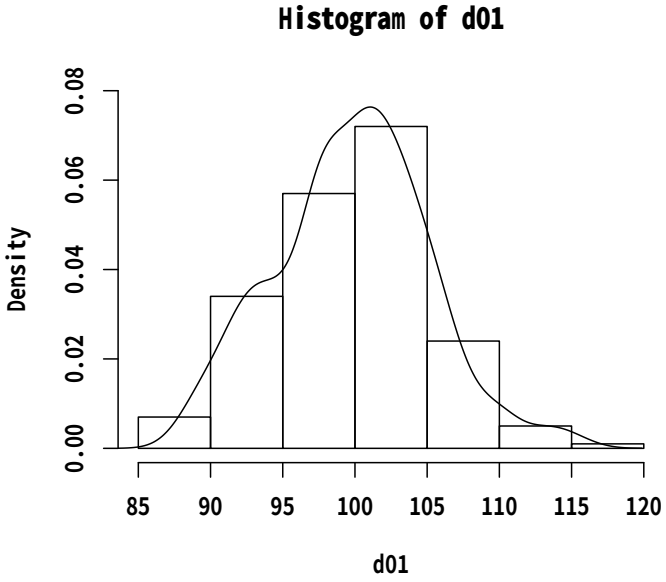
（自分でやる場合は、以下をコピペしてください。）

```
d01<-c(97.6,91.9,103.2,88.8,109.9,115.4,105.5,96.4,102.1,97.9,100.5,101.6,89.4,
99.5,100.5,105.2,98.3,100.6,105.5,106.2,94,100.1,98,98.1,94.7,99,
97.7,97.9,105.9,104.5,97.9,93.3,98.7,104.5,91.6,101.7,94.5,102.2,105.1,
101.3,96.6,89.5,100.8,102.6,101.3,100.3,95.8,104.7,99.9,99.6,95.3,91.2,
99.4,97.8,103.6,107.5,100.5,95.4,103,103.1,100,99,101.5,98.1,103,
89.3,91.4,97.2,107.1,95,101,93,100.8,97,93.7,92.6,104.9,96.9,
93.6,98,110,102,101,97.6,92.4,96.7,91.1,97.3,97.6,97.1,99.3,
103.5,107.2,101.3,97.9,97.6,97.5,93.7,91.8,106.6,105.1,101.3,103.1,109.3,
```

```
101.6,113.5,101.4,114.1,99.2,112.8,89.4,109.2,93.6,92.8,102.7,105,91.3,
99.8,103.3,97.4,103.7,97.8,101.6,100.1,105.9,97.1,103.3,103.3,105.8,89.5,
99.9,104.5,105.4,104.3,97.7,101.4,99.5,93,101.1,99.3,94,98.1,100.9,
101.9,102.1,93.7,98.1,101.9,97,95.4,105.5,104.3,110.1,103.5,102.9,98.3,
96.1,92,106.6,106.9,88,103.2,103.5,91.3,100.3,106.7,103.4,100.3,100.6,
110.9,94.8,97.9,97.2,103.8,94.8,101,102.6,100.9,99.7,93.7,105.9,100.8,
98.5,97.7,91.7,90.1,101.2,95.6,103.1,93.4,104.5,102.7,108.1,100.2,102.6,
92.6,94.9,92.2,104.1,102.1)
```

その分布を確認してみると

```
> hist(d01,prob=T,ylim=c(0,.08))
> lines(density(d01))
```



母集団で 90 %が含まれる範囲： quantile 関数

100 地点のピンから 90 %の確率でどのくらいの範囲に入るだろうか。

その場合、この記録の短かったほうから 5 %となる地点と、遠い方 5 %（短いほうから 95 %）となる地点を求めればよい。

d01 の値で小さいほうから 5 %となる点は、データの小さい方から指定した割合にある値を返す関数 quantile を使って、以下のように求めることができる。

```
> quantile(d01,.05)
```

```
    5%
91.295
```

d01 の値で大きい方から 5 %となる地点は、

```
> quantile(d01,.95)
```

```
    95%
108.155
```

つまり、91.3m〜108.2m の間に、90 %の玉が落ちるということになる。

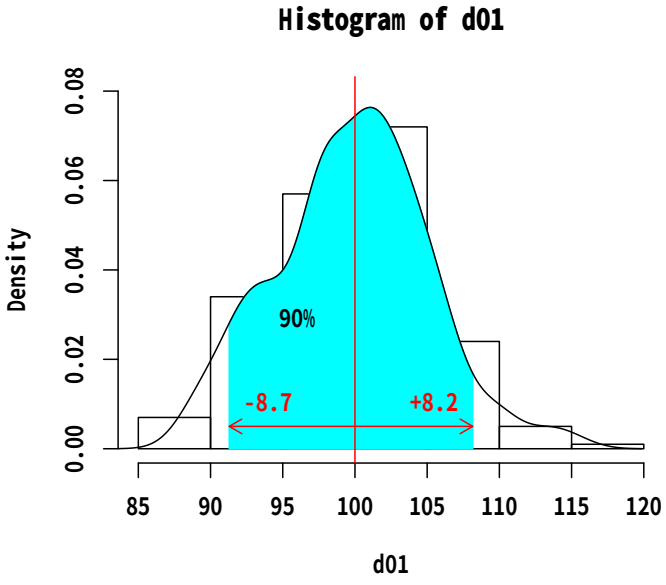
これは、狙った距離 100m から

```
> c(91.3,108.2)-100
```

```
[1] -8.7  8.2
```

の間にある。

```
> hist(d01,prob=T,ylim=c(0,.08)) #ヒストグラム
> #90 %区間を塗りつぶす
> lines(density(d01,from=quantile(d01,.05),to=quantile(d01,.95)),type="h",col=5)
> lines(density(d01)) #確率密度の曲線
> abline(v=100,col=2) #100m 地点に赤の垂直線
> #90 %区間に赤の左右の矢印
> arrows(quantile(d01,.05),0.005,quantile(d01,.95),0.005,code=3,length=.1,col=2)
> #指定した位置に「-8.7」という文字を書き込む
> text(quantile(d01,.05),.01, "-8.7",col=2,pos=4)
> #指定した位置に「+8.2」という文字を書き込む
> text(quantile(d01,.95),.01, "+8.2",col=2,pos=2)
> text(96,.03, "90%") #「90 %」という文字を書き込む。
```



点推定

・・・で、ここで問題です。

リョウ君が、別のコースで１回だけ打ったら、飛距離は 110m でした。

飛距離のバラツキは 100m を狙った時と変わらないとします。

さて、リョウ君は、何メートルの地点を狙ったでしょうか？

それはリョウ君に聞いてみないと分かりませんが、

とりあえず、狙った地点（リョウ君の場合少しオーバー気味？）が最もありそうなところなので、ズバリ、

「110m でしょう」

と答えるのが妥当と言えます。

こうした１点で答えるのを、「点推定」と言います。

区間推定

しかし、100m 狙った場合も、ピンより 8.7m 以上 手前 に落ちることが 5 %ばかりありました。

その確率でいうと、110m に落ちたとしても、実は 8.7m 先 に狙ったピンがある可能性も 5 %ばかしあります。

また、100m 狙った場合に、ピンより 8.2m 以上 先 に落ちることが、やはり 5 %ありました。

だとすると、110m に落ちたとしても、実は 8.2m 手前 に狙ったピンがある可能性も 5 %はあります。

しかし、それでも、その 2 つを合わせても 10 %の確率でしかありません。残りの 90 %は、その間にあるはずです。

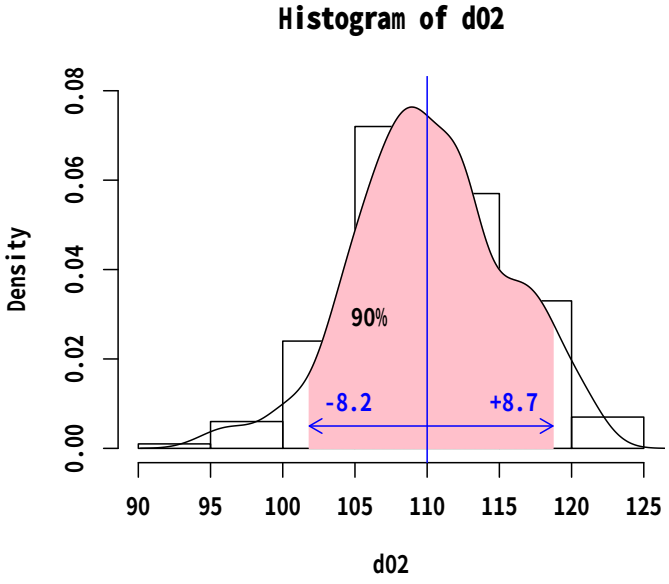
ですから、リョウ君の打った球が 110m に落ちたとしたら、リョウ君は **90** %の確率で 110-8.2〜110+8.7m の範囲にあると推定できることになります。

こうした考え方を「区間推定」と言います。

以下の図は、狙ったピンがある位置の可能性を示しています。

先ほど描いた、100m を狙った時の落ちた球の分布を左右ひっくり返した状態になっていることを確認してください。

```
> d02<-110-(d01-100)
> hist(d02,prob=T,ylim=c(0,.08))
> lines(density(d02,from=quantile(d02,.05),to=quantile(d02,.95)),type="h",col="pink")
> lines(density(d02))
> abline(v=110,col=4)
> arrows(quantile(d02,.05),0.005,quantile(d02,.95),0.005,code=3,length=.1,col=4)
> text(quantile(d02,.05),.01, "-8.2",col=4,pos=4)
> text(quantile(d02,.95),.01, "+8.7",col=4,pos=2)
> text(106,.03, "90%")
```



区間推定は、100 %で推定できることは少なく、90 %、95 %、99 %など、区切りのよい信頼水準で推定することが多いです。

当然、信頼水準を上げると区間推移の範囲（信頼区間）は広くなります。

5.1.2 分布関数から推定する

経験的に母集団の確率分布がわからないことがある・・・というよりも、ほとんどがそう。

その場合でも、理論的に分布関数の 形 を導くことができる場合がある。

たとえば、丸太渡り競争の記録は、指数分布だと仮定でる。

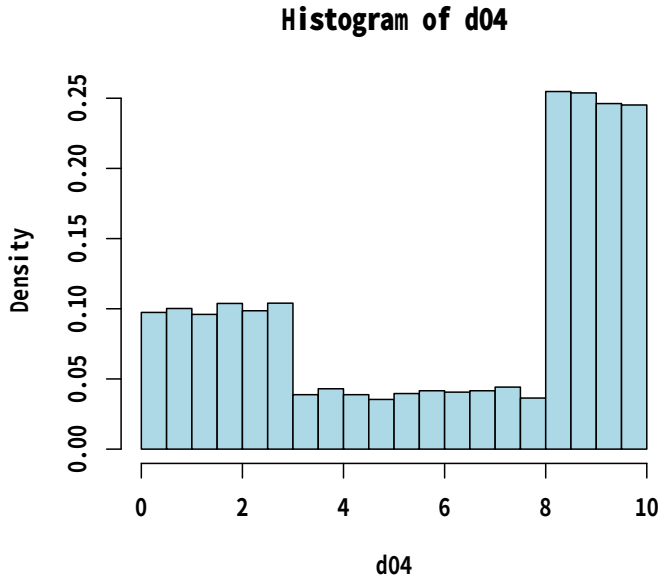
その優勝記録の分布から第一種極値分布を導くこともできる。

ここでは、中心極限定理で、標本平均の分布は、正規分布であり、母分散を標本分散に置き換えると t 分布となることを使ってみる。

準備

以前使ったヘンな分布を使って話をすすめよう。

```
> set.seed(210) #説明の都合上、乱数のパターンを固定した。
> d04_1<-runif(3000,0,3) #[0,3] の範囲の一様乱数を 3000 個
> d04_2<-runif(2000,3,8)  #[3,8] の範囲の一様乱数を 2000 個
> d04_3<-runif(5000,8,10) #[8,10] の範囲の一様乱数を 50000 個
> d04<-c(d04_1,d04_2,d04_3) #以上の３パターンの乱数をつないだ
> hist(d04,col="lightblue",prob=T) #ヒストグラム
```



母平均は、

```
> mean(d04)
```

```
[1] 6.053284
```

母分散は

```
> var(d04)
```

```
[1] 11.35773
```

母標準偏差は

```
> sd(d04)
```

```
[1] 3.370123
```

推定

例のへんな分布をする 10000 個のデータ d04 を母集団として考える。

しかも、この母集団全体は直接観察できないものとする。

だから母平均も母分散もわからないハズだ！　たった今、計算したが、これは神様だけが知っていることにしておく。

いくつかの標本を抽出して、この母集団の平均を探す。

母平均の点推定

調査したところ 10 個の標本が得られた。

```
> (d04s10<-sample(d04,10))
```

```
 [1] 2.469269 8.124595 1.178544 9.613013 2.085397 5.282984 8.429800 8.627716
 [9] 1.997970 1.063596
```

標本平均をとってみる。

```
> mean(d04s10)
```

```
[1] 4.887289
```

これを母集団の推定値として採用する。これが点推定。

え、ちょっと待て！ 標本平均はいろいろな値をとるから、標本平均は母平均と同じではないだろう、とツッコみたくなるのはわかる。

しかし、これしかないんだから仕方ない。

しかも、標本平均は不偏だ。その期待値は母平均に一致する。

$$E(\bar{x}) = \mu$$

ということは、確かに、標本平均は母平均と同じではないかもしれないが、いろいろな値をとる可能性のある標本平均の、その可能性を均すと、母平均あたりで落ち着く。だからそれでよいではないか？

それに、標本平均は、大数の法則により、一致性を持つ。ある程度標本数が大きければ、それほど母集団から離れた値は出ない。

$$\lim_{n\to\infty} \Pr(|\bar{x}^{(n)} - \mu| > \varepsilon) = 0$$

($\bar{x}^{(n)}$ は n 個の規模の標本であることを表す。ε は、任意の数。)

さらに、さらに、中心極限定理により、標本平均の分布は正規分布となる.

$$\bar{x}^{(n)} \sim N(\mu, \sigma^2/n)$$

標本平均として出てきた値は、母平均あたりが最も高く、そこから離れる値が出る可能性はだんだん低くなるはず。

これだけの理由があるから、標本平均を母集団の推定値として採用してもよいのではないか。

「でも、実はだいぶ違うよ」という神様の声も聞こえる。

期待値が一緒でも、このたった一つの標本平均はどうなのよ？

一致性があるといっても、標本規模 10 個でそれがいえるのか？

中心極限定理があるなら、標本規模である程度バラツキが評価できるだろう

・・・ということで、母平均は「これ！」ではなくて、「だいたいこれくらい」で推定してみよう。

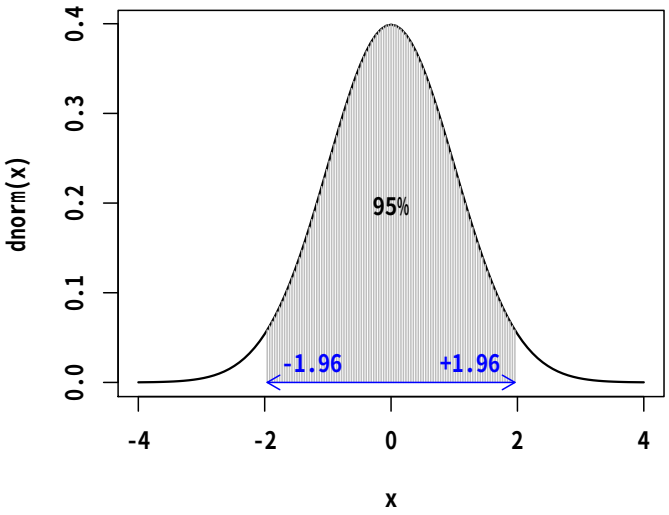
母平均の区間推定

中心極限定理により、

$$\frac{\bar{x}-\mu}{\frac{\sigma}{\sqrt{n}}} \sim N(0,1)$$

図に描くとこんな感じ

```
> curve(dnorm,-4,4,lwd=1.5)
> curve(dnorm,from=qnorm(.025),to=qnorm(.975),add=T,type="h",col="gray")
> arrows(qnorm(.025),0,qnorm(.975),0,code=3,length=.1,col=4)
> text(-1.96,.02, "-1.96",col=4,pos=4)
> text(1.96,.02, "+1.96",col=4,pos=2)
> text(0,.2, "95%")
```



であり、これは、つまり、標本平均は、母平均 μ、母分散 σ^2/n の正規分布に従うということ。

$$\bar{x} \sim N(\mu, \sigma^2/n)$$

ということは、標本平均は、95 %の確率で母標準偏差の 1.96 倍の間にあるはず。

$$\mu - 1.96\frac{\sigma}{\sqrt{n}} \leq \bar{x} \leq \mu + 1.96\frac{\sigma}{\sqrt{n}}$$

だから、標本平均が 1 つ得られたら、母平均は、95 %の確率で、母標準偏差の 1.96 倍の間にあるはずだ。

$$\bar{x} - 1.96\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{x} + 1.96\frac{\sigma}{\sqrt{n}}$$

しかし、母標準偏差はわからない。どうするか？

標本標準偏差 s で母標準偏差 σ を置き換える。

ところが、n 個の規模の標本平均を、標本標準偏差で正規化した値は、前回やったように、自由度 $n-1$ の t 分布に従う。

$$\frac{\bar{x}^{(n)}-\mu}{\frac{s}{\sqrt{n}}} \sim t(n-1)$$

t 分布で、0 を中心に 95 %を含む区間は、関数 qt を使って、小さい方から 2.5 %の点と 97.5 %の点を求めればよい。

10 個の規模の標本平均の場合は、自由度 9 の t 分布なので、小さい方から 2.5 %の点は、

```
> qt(.025,9)
```

```
[1] -2.262157
```

小さい方から 97.5 %の点は、

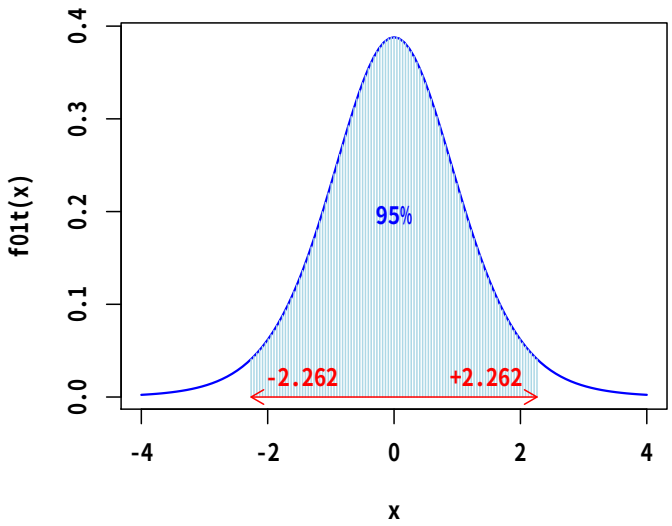
```
> qt(.975,9)
```

```
[1] 2.262157
```

標準正規分布より広くなる。

図に描くとこんな感じ

```
> df01<-9
> f01t<-function(x) dt(x,df01)
> curve(f01t,-4,4,col=4,lwd=1.5)
> curve(f01t,from=qt(.025,df01),to=qt(.975,df01),add=T,type="h",col="lightblue")
> arrows(qt(.025,df01),0,qt(.975,df01),0,code=3,length=.1,col=2)
> text(-2.262,.02, "-2.262",col=2,pos=4)
> text(2.262,.02, "+2.262",col=2,pos=2)
> text(0,.2, "95%",col=4)
```



つまり、

$$\mu - 2.262\frac{s}{\sqrt{n}} \leq \bar{x} \leq \mu + 2.262\frac{s}{\sqrt{n}}$$

ということであり、

$$\bar{x} - 2.262\frac{s}{\sqrt{n}} \leq \mu \leq \bar{x} + 2.262\frac{s}{\sqrt{n}}$$

ということになる。

標本標準偏差 s が母標準偏差 σ に同じだとしたら、母平均 μ があるだろうと推定される区間は、正規分布で予想するよりも広くなる。

つまり予測精度が悪くなる、ということ。

たまたま得られた１つの標本平均と標本標準偏差は

```
> mean(d04s10)
```

```
[1] 4.887289
```

```
> sd(d04s10)
```

```
[1] 3.494923
```

したがって、母平均の 95 %信頼区間は、下限が

```
> mean(d04s10)+qt(.025,9)*sd(d04s10)/sqrt(10)
```

```
[1] 2.387171
```

上限が

```
> mean(d04s10)+qt(.975,9)*sd(d04s10)/sqrt(10)
```

```
[1] 7.387406
```

95 %の確率で、この間のどこかに母平均があるだろうということ。実際、そうだよ、という神様の声も聞こえる？

このように、１点ではなくて、一定の幅をもって目的の値を推定することを「区間推定」と言う。

区間推定は、「100 %の確率で・・・」ということはあまりなく、大概、95 %とか、99 %とか、あるいは、90 %の確率で推定される。

90 %信頼区間となると、区間推定の幅はもう少し狭くなる。

自由度が 9 の場合は、標本平均からプラスマイナス標本標準偏差の 1.833 倍の範囲となる。

```
> qt(.05,9)
```

```
[1] -1.833113
```

```
> qt(.95,9)
```

```
[1] 1.833113
```

同じく自由度が 9 の場合、99 %信頼区間は、標本平均からプラスマイナス標本標準偏差の 3.25 倍の範囲に広がる。

```
> qt(.005,9)
```

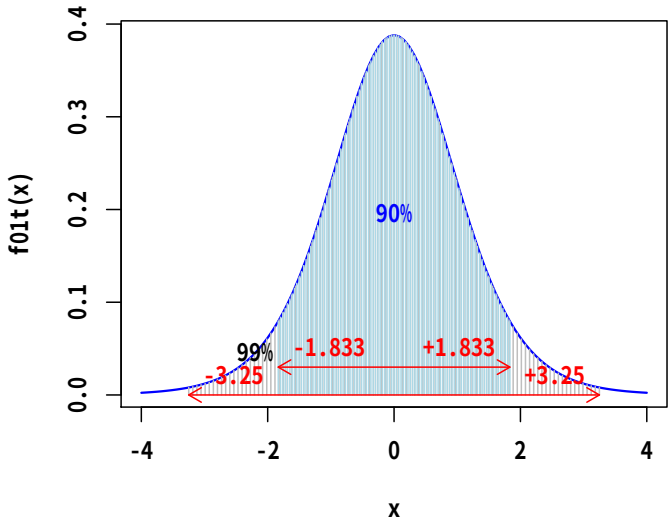
```
[1] -3.249836
```

```
> qt(.995,9)
```

```
[1] 3.249836
```

図に描くと次のようになる。

```
> curve(f01t,-4,4,col=4,lwd=1.5)
> curve(f01t,from=qt(.005,df01),to=qt(.995,df01),add=T,type="h",col="gray")
> curve(f01t,from=qt(.05,df01),to=qt(.95,df01),add=T,type="h",col="lightblue")
> arrows(qt(.005,df01),0,qt(.995,df01),0,code=3,length=.1,col=2)
> arrows(qt(.05,df01),.03,qt(.95,df01),.03,code=3,length=.1,col=2)
> text(-3.25,.02, "-3.25",col=2,pos=4)
> text(3.25,.02, "+3.25",col=2,pos=2)
> text(-1.833,.05, "-1.833",col=2,pos=4)
> text(1.833,.05, "+1.833",col=2,pos=2)
> text(0,.2, "90%",col=4)
> text(-2.2,.05, "99%",col=1)
```



つまり、母集団の信頼区間は、90 %の確率だと、下限が

```
> mean(d04s10)+qt(.05,9)*sd(d04s10)/sqrt(10)
```

```
[1] 2.861347
```

上限が

```
> mean(d04s10)+qt(.95,9)*sd(d04s10)/sqrt(10)
```

```
[1] 6.91323
```

99 %の確率だと、下限が

```
> mean(d04s10)+qt(.005,9)*sd(d04s10)/sqrt(10)
```

```
[1] 1.295597
```

上限が

```
> mean(d04s10)+qt(.995,9)*sd(d04s10)/sqrt(10)
```

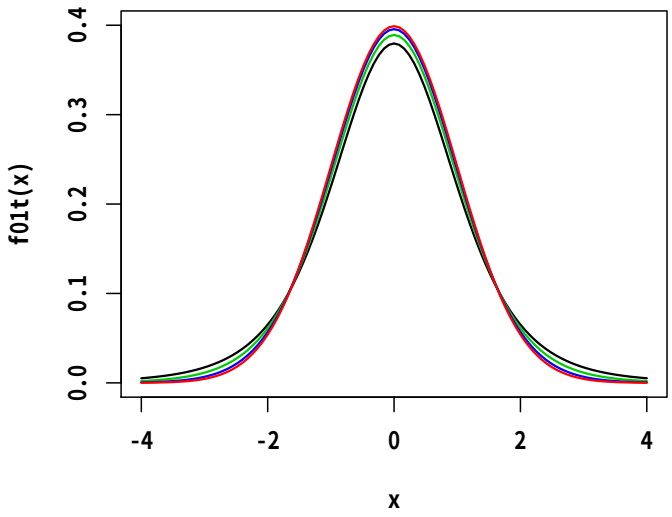
```
[1] 8.47898
```

自由度が大きい母平均の区間推定

しかし、t 分布というのは、自由度に左右される。

自由度 5,10,30 の t 分布を描いてみる。黒い線が自由度 5、緑が自由度 10、青が自由度 30、赤線は、標準正規分布の密度関数。

```
> df01<-5
> curve(f01t,-4,4,lwd=1.5,col=1,ylim=c(0,.4))
> df01<-10
> curve(f01t,add=T,lwd=1.5,col=3)
> df01<-30
> curve(f01t,add=T,lwd=1.5,col=4)
> curve(dnorm,add=T,lwd=1.5,col=2)
```



微妙でわかりにくいかもしれないが、自由度が 30 ぐらいに大きくなると、t 分布も、ほとんど標準正規分布と変わらない。

95 %信頼区間を求めるための、2.5 %の点と、97.5 %の点を求めてみる。

自由度 5 の t 分布はプラスマイナス 2.571

```
> qt(c(.025,.975),5)
```

```
[1] -2.570582  2.570582
```

自由度 10 の t 分布はプラスマイナス 2.228

```
> qt(c(.025,.975),10)
```

```
[1] -2.228139  2.228139
```

自由度 30 の t 分布はプラスマイナス 2.042

```
> qt(c(.025,.975),30)
```

```
[1] -2.042272  2.042272
```

標準正規分布はプラスマイナス 1.96

```
> qnorm(c(.025,.975))
```

```
[1] -1.959964  1.959964
```

グラフでは微妙に見えるが、自由度 5 の t 分布から導かれる信頼区間は、正規分布によるよりもだいぶ広くなってしまう。しかし、自由度 30 ぐらいになるとあまり変わらない。

したがって、自由度が大きい場合は、t 分布を使わずに、正規分布で区間推定をしてしまう場合もある。

5.2 仮説検定の考え方

仮説検定というのは、統計学ではさまざまな場面で使われる。やや独特の考え方で、まわりくどい判断をするので、ピンとこない人もいるかもしれない。

要するに、ある分布を仮定して、ある値が、その分布から得られたかどうかを判定する、ということだが、この辺り、少しかみ砕いて説明してみる。

5.2.1 可能性ゼロと判断できる場合

問題

たとえば・・・

太郎君が、今日、学校から下宿に帰ったら財布がない。

財布を落としたのか？

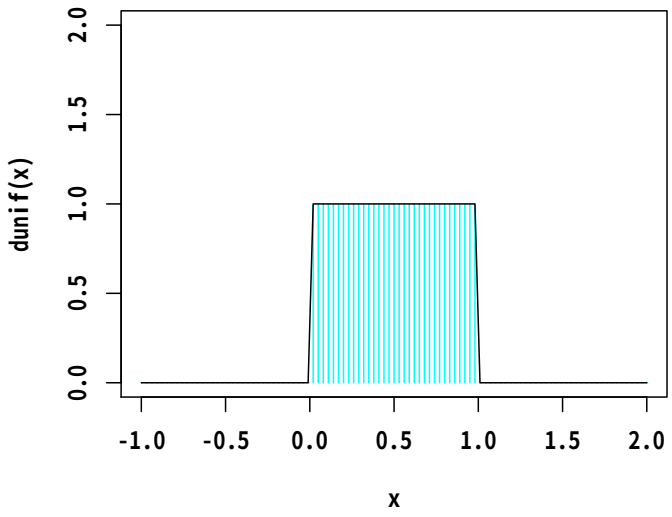
学校にいたとき、財布は確かにあったので、落としたとしたら、学校から下宿までの間だ。

その他の場所には行っていないので、それ以外は考えられない。

学校を 0 地点、下宿を 1 地点として、そのうちのどこで落としたかはわからないので、$[0,1]$ の間のどこで落としたかは、一様分布と考えるしかない。

ただし、それ以外の区間の可能性は 0 である。

```
> curve(dunif,-1,2,type="h",col=5,ylim=c(0,2))
> curve(dunif,add=T)
```



その区間を一生懸命探していたら、友達の J 郎君が、駄菓子屋でなにやら大人買いしていた。

J 郎君に声をかけたら、びっくりした様子。

そして、なんと手には、自分の財布が・・・

そこで太郎君にひとつの疑念が浮かんだ。

そういえば、J 郎のやつ、最近俺の財布をちらちら見ていたな・・・

俺、じつは財布落としてはいない？

それで聞いてみた。

「それ、どうしたの」
「落ちてた・・・。今、君に届けようと探していたところだ」
と、うまい棒を２〜３個手から落としながら釈明した。
「どこに落ちていたの？」
と聞くと、学校の向こう側にある道端と言う。
先ほどの数直線で表すと、-1 の地点となる。

帰無仮説と対立仮説

まず、可能性をふたつに分ける。
ひとつは・・・
［仮説１］J 郎君が、どこかで財布を拾った。その財布を届けようとしていたかどうかはともかく、どこかで財布を拾った。
もうひとつは、［仮説１］以外の方法。・・・
［仮説２］J 郎君は、実は財布を拾っていない。拾っていないということは・・・
このうち、［仮説１］なら、J 郎君がどこで拾う可能性があるかについて、確率分布を確定することができる。
J 郎君がありえない場所を答えたとしたら、J 郎君はウソをついていると識別することができる。
この場合、［仮説１］を帰無仮説、［仮説２］を対立仮説と呼ぶ。

帰無仮説の棄却

仮説検定は、ひとまず（確率分布が確定できる）帰無仮説が正しいと考え、事象が起きた確率を計算する。
ひとまず、J 郎君は財布を拾ったのだ、と信じよう。
しかし、拾った場所は-1 の地点。
その確率は 0！
「ありえね〜」
・・・ということで、J 郎君逮捕！
つまり、帰無仮説を「棄却」し、対立仮説を「採択」した。

帰無仮説が棄却できない場合

しかし、J 郎君が 0.5 の地点を答えていたらどうなるだろうか？
J 郎君は、本当に財布を拾った可能性は否定できない。
かといって、そうではない可能性も否定できない。
識別できないということ。
この場合は「帰無仮説は棄却できない」と言い、グレー判定となる。

仮説の妥当性は？

仮説検定は、本当に識別したい事象が生じる可能性を、直接判定できないので、一旦、確率分布が確定できる状況を帰無仮説として設定してそれを否定することでお目当ての事象が生じたと判定しようというもの。
だから、帰無仮説以外の状況がお目当ての状況でなければならない。
帰無仮説を否定しても、お目当ての状況以外の可能性が残っていれば、帰無仮説を否定しても、対立仮説を実証したことにはならない。
太郎君が実証したのは、J 郎君の答えが事実である可能性は 0 ということ、だけ。つまり、J 郎君はウソの回答をした、というのが正確な帰無仮説。
なぜウソの回答をしたか・・・
それは拾っていないから、どこに落ちる可能性があるか分からなかったからだ・・・
つまり・・・

という推論。
なぜウソの回答をしたか・・・
J 郎君は極度の方向音痴で、自分がどこにいるかもわからないからだ・・・
という可能性もあるかもしれない。
人を疑うときは、よくよくご注意を！

5.2.2 経験的分布に基づく検定

帰無仮説が正しい可能性がゼロの場合は、簡単に帰無仮説を棄却できるが、ほとんどの場合、帰無仮説が正しい可能性もいくらか残る。「可能性がゼロでない」。

その場合、帰無仮説が正しい状況での母集団の分布が経験的に分かっている場合ならば、それに従って仮説検定が可能となる。

問題
花子さんは、菊ちゃんと待ち合わせをしている。
花子さんは、菊ちゃんとはこれまで何百回と待ち合わせした。
だから、菊ちゃんが待ち合わせ時間にどのくらい遅れてくるか、その確率分布が花子さんの頭にはある。
菊ちゃんは、＜何もない＞ ならば、100 回待ち合わせしたら、待ち合わせ時間から、だいたい以下の時間内には待ち合わせ場所に来ている。

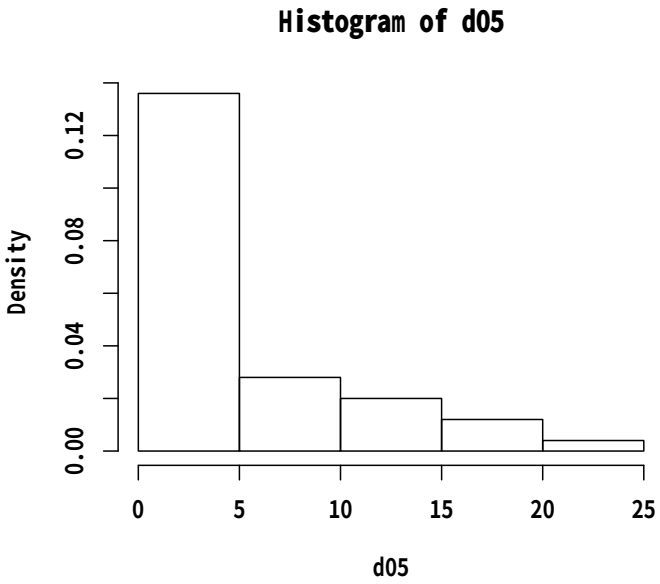
```
> d05<-c(24,4,2,1,4,0,0,17,0,14,0,6,10,5,4,4,13,4,1,2,16,5,3,11,
+ 3,5,8,1,1,4,7,5,1,2,6,1,3,4,2,12,7,3,15,4,12,0,0,5,
+ 0,14,1,0,19,5,8,2,6,0,5,6,3,2,3,2,0,6,1,10,3,16,5,3,
+ 17,8,3,22,18,6,13,4,13,4,5,1,13,2,4,0,1,2,5,6,3,0,0,1,
+ 2,1,1,3)
```

（自分でやる場合は、以下をコピペして使ってください）

```
d05<-c(24,4,2,1,4,0,0,17,0,14,0,6,10,5,4,4,13,4,1,2,16,5,3,11,
3,5,8,1,1,4,7,5,1,2,6,1,3,4,2,12,7,3,15,4,12,0,0,5,
0,14,1,0,19,5,8,2,6,0,5,6,3,2,3,2,0,6,1,10,3,16,5,3,
17,8,3,22,18,6,13,4,13,4,5,1,13,2,4,0,1,2,5,6,3,0,0,1,
2,1,1,3)
```

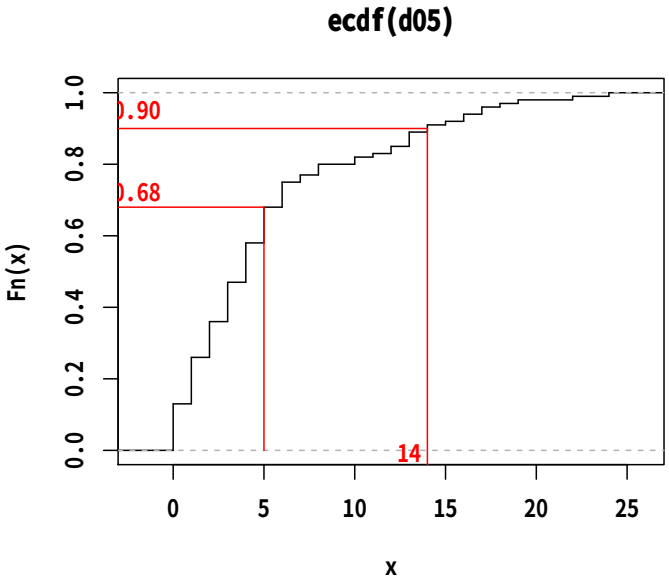
ヒストグラムに描いてみる

```
> hist(d05,prob=T)
```



累積分布を描くと次のようになる。

```
> plot(ecdf(d05),verticals=T,col.points=F)
> # 以下は赤線、赤字部分
> segments(5,0,5,0.68,col=2)
> segments(-5,.68,5,.68,col=2)
> text(-2,.73,"0.68",col=2)
> segments(14,-2,14,0.90,col=2)
> segments(-5,.90,14,.90,col=2)
> text(13,0,"14",col=2)
> text(-2,.96,"0.90",col=2)
```



菊ちゃんは、＜何もない＞ ならば、68 %（3 回に 2 回）は 5 分以内に到着している。
＜何もない＞ ならば、15 分以上遅れることは 10 回に 1 回もない。
ある日、待ち合わせていたら、5 分待っても菊ちゃんは来ない。
10 分待っていても菊ちゃんは来ない。

15 分待っても菊ちゃんは来ない。
菊ちゃんに＜何かあった＞？

帰無仮説と対立仮説

始めに帰無仮説と対立仮説を設定する。
帰無仮説：菊ちゃんの身に何も起こっていない（普段どおり）
対立仮説：菊ちゃんの身に何か起こった。何が起こったかはわからないが、何かが起こって、待ち合わせに来られない。

帰無仮説の棄却

ひとまず、帰無仮説が正しいと考える。
＜何もない＞ 場合、上のグラフのように、菊ちゃんが来る時間についての確率分布は経験的に確定できる。
これからすると、5 分待った段階で、菊ちゃんの身に何かあったと判定するのは拙速だ。
＜何もない＞状況だと、確かに、3 回に 2 回は 5 分以内に来ているが、それでも 3 回に 1 回は 5 分以上遅れることがある。
10 分待った段階でも、10 回に 2 回ぐらいは来ていない。
しかし、15 分も待って来ないというのは 10 回に 1 回もない。
つまり、15 分待っても菊ちゃんが来ないのは、10 回に 1 回未満の珍しい状況が生じた・・・と考えるよりも、＜何かあった＞という別の状況が生じたと考えた方が妥当だと言える。
つまり帰無仮説を棄却して、対立仮説を採択したことになる。
ただし、それでも＜何もなかった＞可能性が 10 %未満だが残っている。
この場合、「有意水準 10 %で対立仮説を棄却した」と言う。
有意水準をどの点に設定するかは、問題に求められる精緻さによるが、通常、切りのよいところで、10 %、5 %、1 %などが利用される。
10 %より 5 %、5 %より 1 %の有意水準で検定を行った方が、検定の確実性は増すが、1 %になるまで待っていたら、菊ちゃんを救えないかもしれない。
確実性と判断の必要性・・・そこは折り合いだ。

帰無仮説が棄却されない場合

花子さんが、5 分待った段階では、まだ、菊ちゃんの身に＜何かあった＞とは判断できないが、だからといって、菊ちゃんが無事かどうかはわからない。
菊ちゃんの身に＜何かあった＞と判断する積極的な証拠がない、というだけである。
実際には、菊ちゃんは 5 分遅れた段階で、ちくわを喉に詰めてもがき苦しんでいたかもしれないのだ・・・
しかし、そのような状況と、普段の状況とが５分待っただけではわからないというだけ。
逆に 20 分待って来なかったので、花子さんが慌てて下宿に走っていったら、（たいへん珍しいことだが）花子さんは、普段どおり出かけようとしていて、「ごめんごめん、ちょっとおそくなっちゃった！」と言うだけかもしれないのだ。

第一種の過誤と危険率、第二種の過誤と検出力

帰無仮説は、棄却されてもなお正しい可能性は残されている場合がほとんどである。
花子さんが＜何かあった＞と判断したとしても、実際には菊ちゃんの身に＜何もない＞かもしれない。
そうした、本当は帰無仮説が正しいのに、対立仮説を採択するという誤りを「第一種の過誤（タイプＩエラー）」と呼ぶ。
第一種の過誤をおかす確率が「有意水準」であり、これは「危険率（α）」とも呼ばれる。
一方、まあそんなこともあるわな・・・なんて言っている間に、実際には菊ちゃんは苦しんでいるかもしれ

ない。

そうした、本当は対立仮説が正しいのに、帰無仮説を棄却できないままでいるという誤りを「第二種の過誤（タイプ II エラー）」と呼ばれる。

第二種の過誤をおかす確率を β で表したときに、$1-\beta$ は「検出力（power）」と呼ばれる。

検出力は、帰無仮説が有意水準以下となる点以降で対立仮説が生じる確率であり、危険率がはっきりと定めることができるのに対し、検出力は、そのそも対立仮説の確率分布自身が定義しにくいので、どれだけとは言いにく場合も多い。

実際、菊ちゃんの例だと、危険率（有意水準）は 10 %だったが、菊ちゃんの身に＜何かあった＞確率分布は定義しにくいので、検出力はどうも表現しにくい。

曲りなりにでも、菊ちゃんの身に＜何かあった＞確率分布を定義したら、はてさて、どうなるのか、自分なりに考えてみて、検出力はどこになるか考えて見てください。

5.2.3　分布関数が定義できる場合の検定

経験的に母集団の確率分布がわかっている場合はよいが、通常、そううまくいくとは限らない。

経験的な確率分布がわからなくても、分布関数が仮定できる場合はそれが使える。

たとえば、丸太競争の場合、平均記録さえ分かっていれば、だいたいどのくらいの記録が出そうなのか、指数分布として予想することができる。

平均記録が 10m なのに、40m とか記録がポンポン出るようだったら、「すごいな〜」と感心するよりも、なにかカラクリを疑うべきだ。

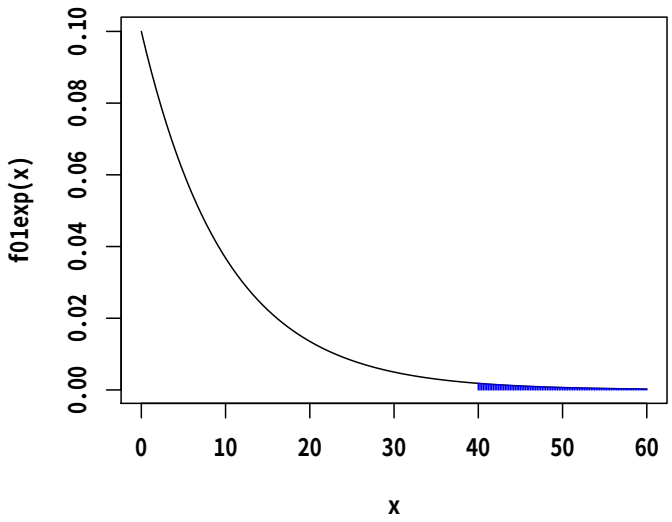
平均値 10 の指数分布だと、40 以上の記録が出るのは、100 回に 1〜2 回。

```
> 1-pexp(40,1/10)
```

```
[1] 0.01831564
```

図に描くと、

```
> lmd01<-1/10
> f01exp<-function(x) dexp(x,lmd01)
> curve(f01exp,0,60)
> curve(f01exp,40,60,type="h",col=4,add=T)
```



ほとんどありえない状況が起こってる。
標本平均の分布関数はわかっていたので、これで説明しよう。

問題
ある村で、松茸を採っている。
最近、松枯れがひどく、なかなか松茸も採れない。
誰が採りに行ってもせいぜい１日 5〜6 本というところ。
しかし、最近、松茸採りを始めたゴン太は、20 本ぐらい採ってくることがある。
村の人は、あんなトーシロが、そんなに採れるはずがない、何か悪さをしているはずだ・・・と噂している。
ゴン太に聞いても「たまたまだ」と答えるだけだ。
ゴン太はこれまで 5 回行って、以下の記録だ。

```
> (d06<-c(10,5,20,9,16))
```

```
[1] 10  5 20  9 16
```

標本平均は

```
> mean(d06)
```

```
[1] 12
```

この村で一人一日で採れる松茸の量の母平均を 6（$\mu = 6$）としよう。
その差は 6 本もある。
標本標準偏差は

```
> sd(d06)
```

```
[1] 5.958188
```

ゴン太がたくさんとれたのは「たまたま」か？

帰無仮説と対立仮説

ひとまず、ゴン太は何も悪さはしていないと考え、これを帰無仮説とする。

帰無仮説：ゴン太が松茸をたくさん採れたのは「たまたま」だ。

対立仮説：ゴン太が松茸をたくさん採れたのは「たまたま」ではない。何かやっている。

帰無仮説のもとでの確率分布

母平均が μ だけが分かっているとき、その母集団から抽出した n 個の標本の分布は、標本平均が $\bar{x}$、標本標準偏差が s のとき、以下の値が、自由の $n-1$ の t 分布に従う。

$$\frac{\bar{x}-\mu}{\frac{s}{\sqrt{n}}} \sim t(n-1)$$

母平均は　$\mu = 6$、

標本平均は　$\bar{x} = 12$、

標本標準偏差は　$s = 5.958$、

標本規模は　$n = 5$

・・・ということは、t 値が、

$$t = \frac{12-6}{\frac{5.958}{\sqrt{5}}}$$

以上となる自由度 $4-1$ の t 分布における確率を求めればよい。

一回計算したが、標本平均、標本標準偏差は mean 関数と sd 関数で書く。

t 値は

```
> (t01<-(mean(d06)-6)/(sd(d06)/sqrt(5)))
```

```
[1] 2.25176
```

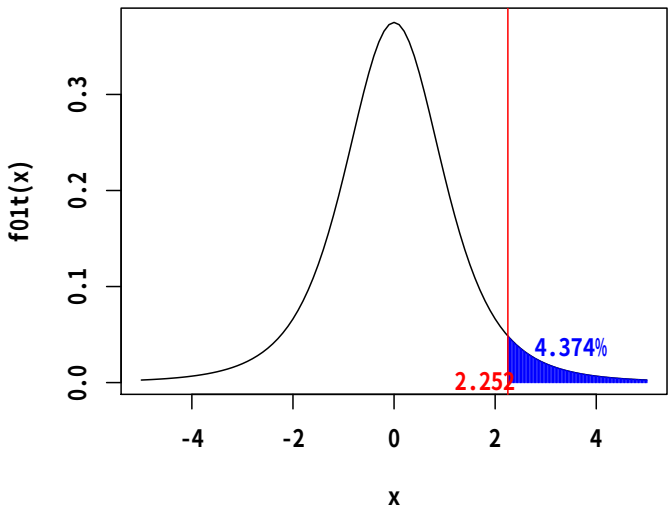
pt 関数は、２番目の引数に与えた値を自由度として、小さい方からの累積確率を求めるので、１番目の引数に与えた t 値（ t01 ）以上となる確率は、1 からその値を引けばよい。

```
> 1-pt(t01,5-1)
```

```
[1] 0.04373726
```

図示すると以下のようになる。

```
> df01<-5-1
> f01t<-function(x) dt(x,df01)
> curve(f01t,-5,5)
> curve(f01t,t01,5,add=T,type="h",col=4)
> abline(v=t01,col=2)
> text(1.8,0.005,round(t01,3),col=2)
> text(3.5,0.04,"4.374%",col=4)
```



帰無仮説の棄却

帰無仮説「ゴン太が普通に松茸をとりに行って「たまたま」たくさん採れた」が正しい確率は、まだ 4.374 %残っている。全くありえないというわけではない。

しかし、その確率は 5 %未満。この可能性を「小さい」とみなせば、有意水準（危険率）5 %未満で帰無仮説は棄却される。つまり、対立仮説が採択されて、ゴン太が松茸をたくさん採ったのは「たまたま」でなく、何か小細工をしたのだと判定される。

しかし、実際にはゴン太が何もしていないのだとしよう。この場合、ゴン太は、たまたまうまくいっただけなのに、あんまりうまくいきすぎたので濡れ衣を着せられたことになる。

このように、帰無仮説が正しいのに、たまたま偏った結果が出て、帰無仮説を棄却してしまうことを、「第一種の過誤（タイプ I エラー）」と言う。

逆に、実際にゴン太は何か小細工しているとしたら、どうだろうか？ この場合、正しい判定だから問題ない。

しかし、場合によっては、過って帰無仮説を棄却しないこともありうる。たとえば、人を疑うのに有意水準 5 %未満はないだろう、やはり 1 %未満にすべきだ、とか言い出してたら、帰無仮説は棄却できない。

「もしかしたら、ゴン太は何かしたかもしれないが、ゴン太がとった松茸の本数は、100 人採りに入ったら 4 人ぐらいは経験する採れ方であって、それだけでゴン太を疑うのは無理がある」

・・・ということになる。ゴン太が実際には何か小細工していても、判断が慎重すぎて、なかなか断罪できない。

このように、帰無仮説は正しくないのに、帰無仮説を棄却を棄却しないことを「第二種の過誤（タイプ II エラー」と言う。

検出力

ここで村人の間で、２つの説が浮上した。

ひとつは、松茸の採り方をタメ助じいさんに聞いたのではないか、という説。

タメ助じいさんは松茸採りの名人だが、その採り方は門外不出で、誰にも教えてくれなかった。

しかし、ゴン太はタメ助じいさんにかわいがられていた。

タメ助じいさんも、もう長くないので、ゴン太に松茸の採り方を教えたのではないのか、と。

ところが、ここで反論が。

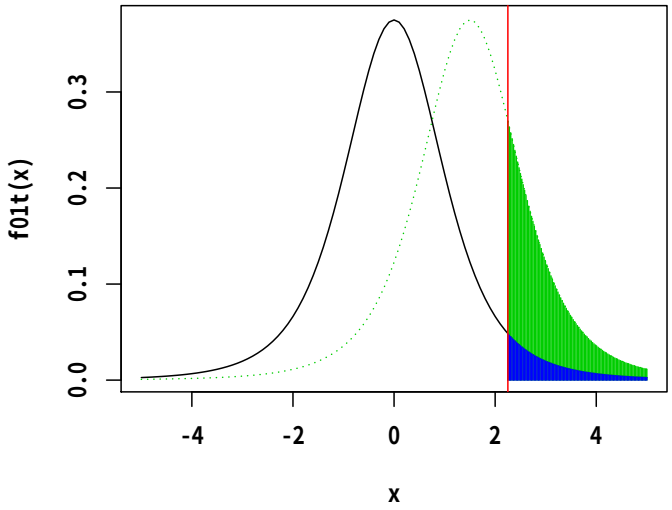
「それでも、20 本もとれることがあるのはおかしいだろう。タメ助じいさんでも、高々 12〜13 本、平均する

と、10 本ぐらいだった。」

ゴン太がも、タメ助じいさんと同じじくらい採れるとするとそのぶん分布はずれて、緑色の点線となるはずだ。

この場合描ける t 値に対する確率分布は、（ほんとうはわからないのだが）、だいたい次のようになるだろう。

```
> df01<-5-1
> f01t<-function(x) dt(x,df01)
> f02t<-function(x) dt(x-4/(sd(d06)/sqrt(5)),df01)
> curve(f01t,-5,5)
> curve(f02t,t01,5,add=T,type="h",col=3)
> curve(f02t,-5,5,add=T,lty=3,col=3)
> curve(f01t,t01,5,add=T,type="h",col=4)
> abline(v=t01,col=2)
```



確かに、村人の説が正しいと考えたほうが、確率的には高いが、それでも、判断は微妙である。

もうひとつの説は、御法度の場所から採ってきたのではないか、というもの。

「隣村に、なんとかという NPO が入って、アカマツの林の手入れをしているらしい。松茸は養育中で、採ってはいけないことになっているそうだが、ゴン太め、あそこ入りおったな」

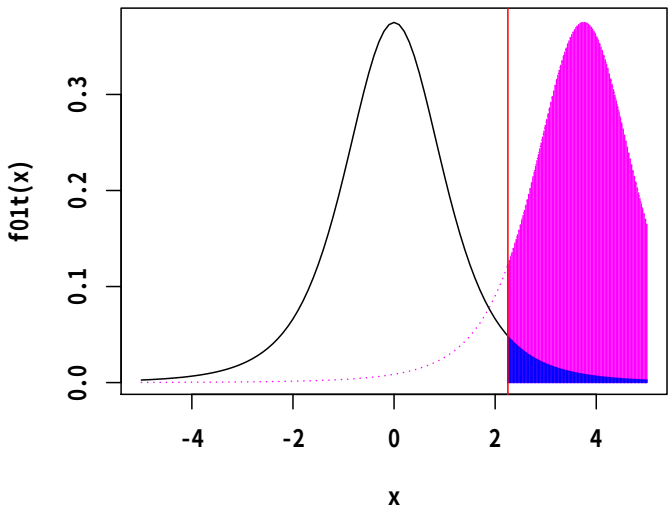
「たしかに、そうなると１日で本や 20 本はざらだろう」

「しかし、そうだとすると、逆にちょっと少ないのではないか」

なんてことも言われかねない。

この場合の t 値がどのような値をとるか、その確率分布を（本当はわからないが）便宜的に描くと、次のようになる。

```
> df01<-5-1
> f01t<-function(x) dt(x,df01)
> f03t<-function(x) dt(x-10/(sd(d06)/sqrt(5)),df01)
> curve(f01t,-5,5)
> curve(f03t,t01,5,add=T,type="h",col=6)
> curve(f03t,-5,5,add=T,lty=3,col=6)
> curve(f01t,t01,5,add=T,type="h",col=4)
> abline(v=t01,col=2)
```



これだけ違うと、逆に、標本平均 12 ぐらいではちょっと少ないのではないかということになってしまうこともある。

しかし、いずれにせよ、こっちの方が、どちらの山から採った松茸かは識別しやすいのも事実で、第一種の過誤も第二種の過誤も起きにくくなる。

以上のように、対立仮説の分布が確定できたとき、緑やピンクの部分の確率が「検出力（power）」と呼ばれ、$1-\beta$ で表される。

検出力が大きいと、第二種の過誤は起きにくくなる。

第 6 章

最小二乗推定量

6.1　導出と性質

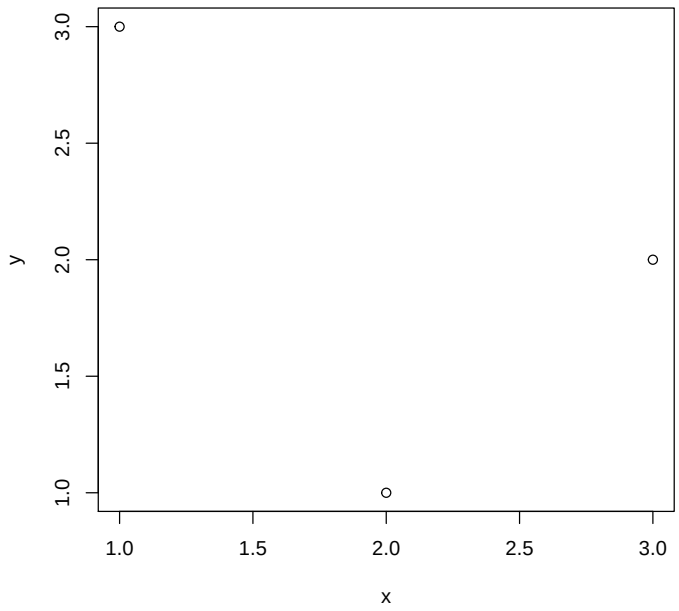
OLS 回帰を R でやるのは簡単だった。

6.1.1　OLS 回帰の例（復習）

サンプルデータ

第２項でやった、たった３つのデータを使った OLS 回帰を例に説明しよう。

```
> x<-c(1,2,3)
> y<-c(3,1,2)
> plot(y~x)
```



ちょっと寂しいが、説明を簡単にするためだ。

OLS 回帰: lm 関数

これで OLS 回帰には lm 関数を使った。

```
> (o01<-lm(y~x))
```

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
        3.0         -0.5
```

これだけだと最小限の情報しか返さない。

OLS 回帰の結果の詳細: lm 関数の結果に対する summary 関数

summary 関数を使う。

```
> summary(o01)
```

```
Call:
lm(formula = y ~ x)

Residuals:
   1    2    3
 0.5 -1.0  0.5

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.000      1.871   1.604    0.355
x             -0.500      0.866  -0.577    0.667

Residual standard error: 1.225 on 1 degrees of freedom
Multiple R-squared:   0.25,        Adjusted R-squared:   -0.5
F-statistic: 0.3333 on 1 and 1 DF,  p-value: 0.6667
```

この出力の意味を解説する。

```
Call:
 lm(formula = y ~ x)
```

とあるのは OLS 回帰式の確認

```
Residuals:
    1    2    3
  0.5 -1.0  0.5
```

とあるのは、回帰の残差を示す。

ここではデータが３つしかなかったので、そのまま出力されているが、もっとデータがたくさんある場合は、分布の summary が出力される。

```
      Min       1Q   Median       3Q      Max
-1.02206 -0.48553 -0.00657  0.48987  1.04287
```

といった感じ。

```
 Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     3.000      1.871   1.604    0.355
x              -0.500      0.866  -0.577    0.667
```

定数項 (Intercept) の推定値 Estimate は 3.00、x の係数の推定値は-0.50。

それぞれの標準誤差 Std. Error は 1.871 と 0.866。

ということは、t 値（t value）は、それぞれ 1.604 と-0.577。

t 値は Estimate を Std. Error で割った値。

帰無仮説を 「Estimate で推定されている各説明変数の真の係数が 0」 としたとき、この帰無仮説が正しければ、t 値は自由度 1 の t 分布に従う（ただし、自由度は分析によって違う。この場合データセットが 3 個、推定すべきパラメータが 2 つなので 3-2=1 で、自由度 1 ということ）。

これによって計算した p 値、つまり、上記の帰無仮説が正しい場合に、t 値の絶対値が推定された値よりも大きくなる確率が `Pr(>|t|)` 。

すなわち、帰無仮説が正しい場合（Estimate が 0 の場合）に、それぞれの t 値の絶対値が 1.604 以上、0.577 以上となる確率となる。

`Pr(|t|>t)` が、それぞれ 0.355 と 0.667 なので、定数項と x の係数が、3.00 と-0.50 となったのはたまたまで、本当は 0 である可能性が、それぞれ 35.5 %、66.7 %も残っていることを示す。

つまり、この場合、この OLS 回帰係数の推定値が 0 である（x で回帰する意味がない）可能性が非常に高いということになる。

```
Residual standard error: 1.225 on 1 degrees of freedom
```

残差の標準誤差で、 Residuals の二乗和を自由度で割った値の平方根。

自由度は、データの数から定数項を含む推定パラメータの数で、この場合データが 3 つで、推定係数が定数項と x の係数で 2 つだから、3-2=1。

残差の標準誤差は、`Residual:` のところにある 3 つの残差を使って、

```
(0.5^2+(-1.0)^2+0.5^2)/1
```

の平方根を求めれば、ちゃんと一致していることが確認できる。

```
Multiple R-squared:   0.25
```

これは、決定係数 R^2 のこと。

この例では、被説明変数の分散のうち回帰式の $a + b_1x_1 + b_2x_2 + ...$ の部分の分散の割合。

被説明変数の分散

```
> var(y)
```

```
[1] 1
```

回帰式の残差以外の分散

```
> var(-0.5*x)
```

```
[1] 0.25
```

残差の分散

```
> var(resid(o01))
```

```
[1] 0.75
```

被説明変数の分散＝回帰式の残差以外の分散＋残差の分散、となっているので、決定係数は、

```
> var(-0.5*x)/var(y)
```

```
[1] 0.25
```

でもよいし、

```
> 1-var(resid(o01))/var(y)
```

```
[1] 0.25
```

でもよい。

```
    Adjusted R-squared:   -0.5
```

自由度調整済み決定係数。

決定係数は、説明変数の数を増やしさえすれば増えるので、その分の自由度の減り方を差し引いて決定係数を評価したもの。

上で出てきた残差の標準誤差 Residual standard error は summary(o01)$sigma で取り出せるので、これを二乗して使えばよい。

```
> 1-summary(o01)$sigma^2/var(y)
```

```
[1] -0.5
```

自由度調整済み決定係数は、説明度が低いと、マイナスになってしまうこともある。

```
    F-statistic: 0.3333 on 1 and 1 DF, p-value: 0.6667
```

帰無仮説が「回帰式の係数がすべて 0 である」、つまり「回帰式に意味がない」とき、例題の回帰が推定される可能性を F 検定で判断する。

この場合、 F 統計量 0.3333 は、自由度 (1,1) の F 分布に従うので、帰無仮説が正しい確率は 0.6667。（説明変数が１個しかないので、 x の係数の p 値と同じになる）

この p 値では、帰無仮説を棄却できない。つまり、この回帰分析では、回帰式が無意味である可能性を否定できない。

「OLS 回帰を行う」というのはこれだけのことだが、以下では、その意味を考え、評価してみる。

6.1.2　回帰式の意味を理解する

誤差項

因果律は、一切のものには原因があるという。

y が特定の値をとるということは、それを説明する x という原因があるはずだ。

ただし、それはひとつの原因とは限らないので、説明する変数は、$x_1, x_2, x_3, x_4, ..., x_p$ と、たくさんあってもよい。

p は 10 かもしれないし、20 かもしれないし、100 かも、1000 かもしれない。

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + ... + \beta_p x_p$$

しかし、そのすべての変数 $x_1, x_2, ..., x_p$ のデータ得られるとは限らない。

すべての変数にデータが得られたとしても、この回帰式を推定するためには、各 $x_1, x_2, ..., x_p$ と y について、p 個よりも大きな規模のデータセットが必要である。

第２講で、中古車プリウスのオークション価格を分析したが、その説明変数は、年式、走行距離、傷、車体の色、装備等々、たくさんあった。

しかし、それでもなお、説明しきれていない価格差があった。たとえば、傷がある無しでも、それが衝突による傷か、かすり傷かによっても違うだろう。それまで乗っていたのが、１人のオーナー化、２人以上に乗り

継がれた車か、それとも営業車か等々、データセットに入っていない情報がまだまだあるはずだ。

また、車体の色はデータセットにあるが、1000 ものデータセットがありながら、例えば紫色なんかは、数台しかデータとしては得られていないので、統計的に確からしい推定ができない。

その一方で、たくさんの説明変数の中には、分析目的について、それほど重要でないと判断される変数もある。その車に以前乗っていた人の血液型が何か、とか。

したがって、被説明変数を説明するのに重要で、データとして得られる変数に限って推定式を定義することになり、残りは、「誤差項（error term）」として ε として扱われる。攪乱項（disturbance term）と呼ばれることもある。

たとえば、得られる説明変数が、x_1 しかなく、当面、他の変数には興味がなかったら、上の式は、以下のような回帰式となる。

$$y = \alpha + \beta_1 x_1 + \varepsilon$$

ε は、誤差項と呼ばれるので、まるで、主要なのは $\alpha + \beta_1 x_1$ の部分で、ε の部分は「ほんのちょっとの差」と思われがちだが、こっちの方がはるかにたくさんの説明変数が含まれ、y の違いのほとんどを説明している場合もあるので、決してあなどってはいけない。

たとえば、以下のような方程式で y の違いが説明できるとしよう。

$$y = 1 + 2x_1 - 1x_2 + x_3 - 2x_4 + 2x_5 - x_6 + 3x_7$$

これについて 10000 個のデータセットが得られたとしよう。

x_1 は、1,2,...,10 の整数のいずれか。

$x_2, ..., x_7$ は、それぞれ独立に [0,1] の範囲の一様分布であるとしよう。

これは、データ発生を簡単にするためだけのもので、本質的な縛りではない。（しかし、「独立に」という部分は必要）

```
> set.seed(321)
> x1<-sample(1:10,10000,replace=T)
> x2<-runif(10000,0,10)
> x3<-runif(10000,0,10)
> x4<-runif(10000,0,10)
> x5<-runif(10000,0,10)
> x6<-runif(10000,0,10)
> x7<-runif(10000,0,10)
> y<-1+2*x1-x2+x3-2*x4+2*x5-x6+3*x7
```

回帰式が、

$$y = 1 + 2x_1 + \varepsilon$$

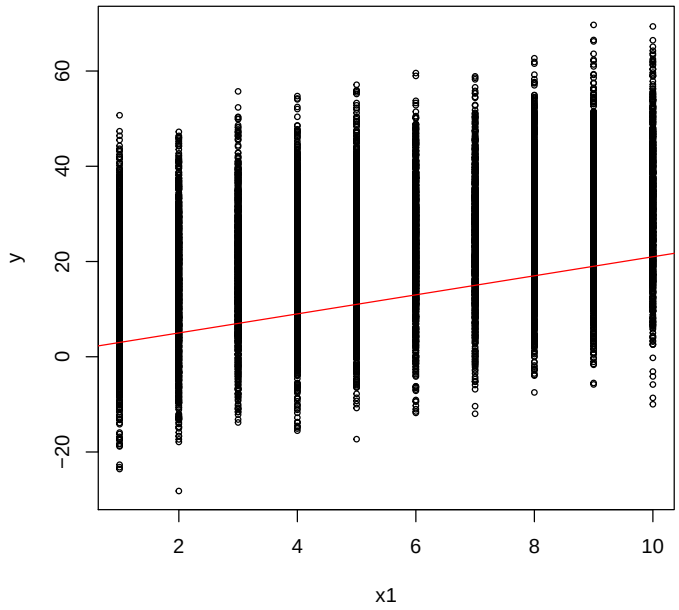
ならば、誤差項は、

$$\varepsilon = -x_2 + x_3 - 2x_4 + 2x_5 - x_6 + 3x_7$$

である。

y を縦軸、x_1 を横軸に散布図を描くと、以下のようになる。

```
> plot(y~x1,cex=.6)
> abline(1,2,col=2)
```



赤い線が $y = 1 + 2x_1$、この直線と各点の垂直方向の距離が ε である。

y の分散は、

```
> var(y)
```

```
[1] 195.9474
```

$1 + 2x_1$ の部分の分散は、

```
> var(1+2*x1)
```

```
[1] 33.32649
```

ε の部分の分散は

```
> er<--x2+x3-2*x4+2*x5-x6+3*x7
> var(er)
```

```
[1] 164.2869
```

y の分散のほとんどを ε が説明している。

（いつもそうだとは限らない。もちろん、x_1 だけでほとんどを説明している場合もある。しかし、そういう場合もあるということ。）

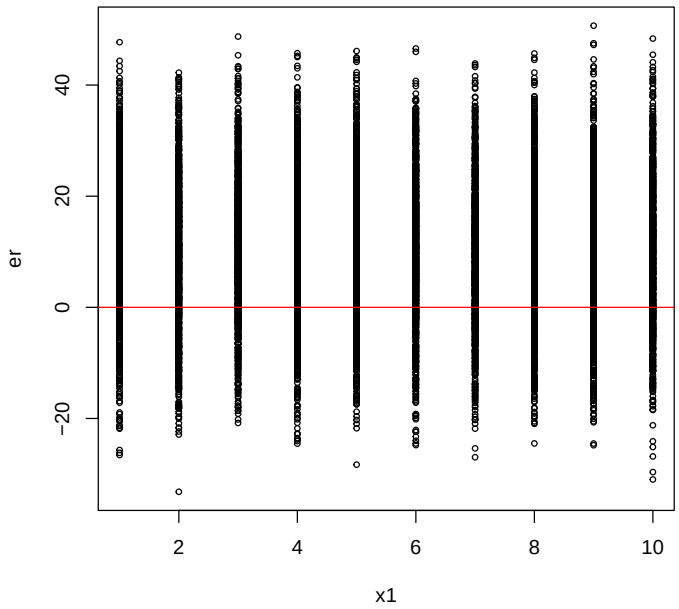
ε だけの分布を描いてみよう。

```
> plot(er~x1,cex=.6)
> abline(h=0,col=2)
> hist(er,br="fd",prob=T)
> lines(density(er))
> mean(er)
```

```
[1] 10.14138
```

```
> sd(er)
```

```
[1] 12.81745
```



平均がほぼ 10、標準偏差が 12.82。分布は x を（便宜上）同じ一様分布からとったので、中心極限定理が効いてしまって、正規分布となってしまったが、いつもそうとは限らない。

ここまで、誤差項の中身がわかっているとして分布を描いたが、通常は誤差項はわからない。

データが得られないばかりか、どんな変数かもわからない場合がある。

そのような場合、誤差項はただ ε と表すしかない。

しかし、それだけでは、あとの推定に困るので、いくつか都合のよい仮定を加えておく。

誤差項に関する仮定（1）：誤差項の期待値は 0

$$E[\varepsilon] = 0$$

これは、本質的な仮定ではない。

しかし、さきほどの例では $E[\varepsilon] = 10$ だっただろ？ という疑問も出されようが、この部分は、定数項に回せばよい。

$$y = \alpha + 10 + x_1\beta_1 + \varepsilon - 10$$

この $\alpha + 10$ を （あえて記号を乱用するが） 新たな α とし、$\varepsilon - 10$ を新たな ε とすればよいだけだ。

そして、改めて、

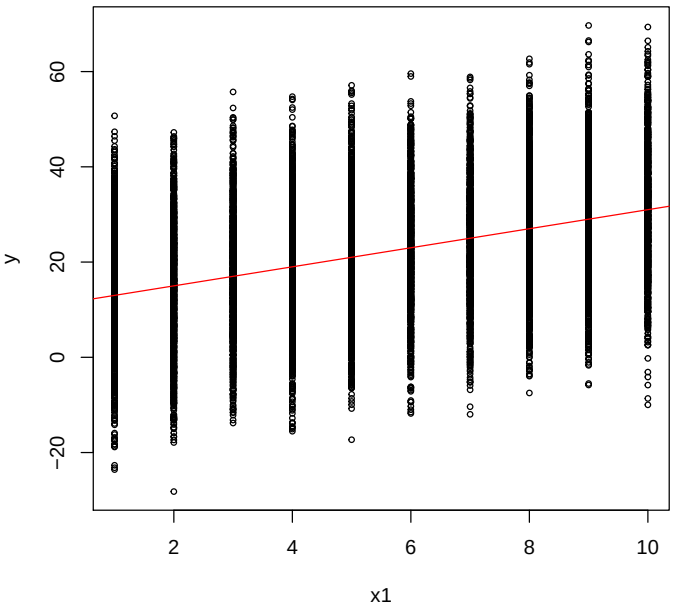
$$y = \alpha + \beta_1 x_1 + \varepsilon$$

であると考えればよい。

こうしても、$\beta_1 x_1$ の部分には全く影響しない。

また、どうせ、α は推定しなければわからなし、その水準自身にはあまり意味はないことが多いので、かまわないだろ。

```
> plot(y~x1,cex=.6)
> abline(11,2,col=2)
```



回帰直線が分布のど真ん中を通っているように見える。

誤差項に関する仮定（2）：誤差項間の独立

誤差項を任意に 2 つだけ取り出したときに、その二つの誤差項 $\varepsilon_1, \varepsilon_2$ は独立である。

これは式で表すと以下のように表すことができる。[*1]

$$E[\varepsilon_1 \varepsilon_2] = 0$$

さきほどの誤差項 er はデータとして 10000 個あったが、前後の積の期待値をとってみよう。

er の 1 番目の値から 9999 番目の値を er1 に代入する。

同じく、2 番目の値から 10000 番目の値を er2 に代入する。

そして、それぞれ掛け合わせる。そうすると、1 番目の値× 2 番目の値、2 番目の値× 3 番目の値、という数値が 9999 個できあがる。

これを er1er2 に代入する。

おっと、 er の期待値は 0 でなかったので、期待値を引いておかなければならなかった。

```
> er0<-er-mean(er)
> er1<-er0[1:9999]
> er2<-er0[2:10000]
> er1er2<-er1*er2
```

分布を確認してみる。

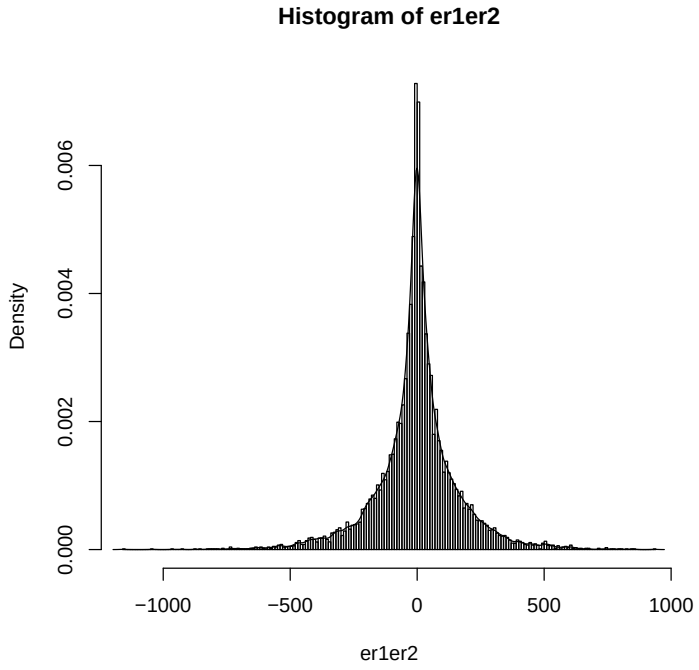
[*1] ちなみに・・・確率変数 x, y が独立であるかどうかは、

$$E[xy] = E[x]E[y]$$

で定義される。

$E[x] = 0$ または $E[y] = 0$ だとしたら、定義により $E[xy] = 0$ だ。

```
> hist(er1er2,prob=T,br="fd")
> lines(density(er1er2))
```



この期待値をとってみるとほぼ 0 になるはず。

```
> mean(er1er2)
```

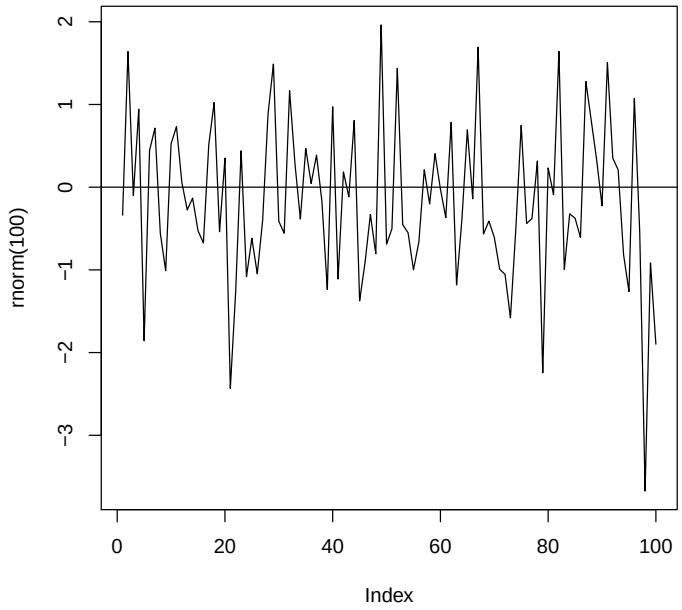
```
[1] -1.102748
```

・・・0 ではないが、100、200 の大きなバラツキを考えると、ほぼ 0 だ！

参考まで、誤差項間が独立でないものに、酔っ払いの千鳥足がある。

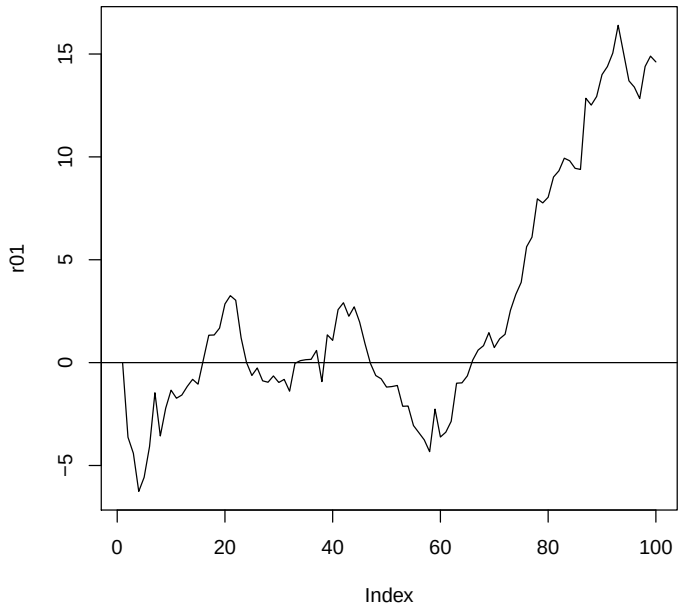
飲み屋からお家までの直線距離。この直線からのずれはランダムであったとしても、酔っ払いの歩き方は、以下のようにはならない。

```
> plot(rnorm(100),type="l")
> abline(h=0)
```



実際には、次の１歩は、その前に踏み外した１歩がどこにあるかに左右されるので、独立ではないのだ！
グラフに描くと以下のようになる。

```
> r01<-rep(NA,100)
> r01[1]<-0
> for(i in 1:99) r01[i+1]<-r01[i]+rnorm(1)
> plot(r01,type="l")
> abline(h=0)
```



どこぞで見た経済時系列のグラフのようだ。
こういう動きにいろいろ理屈つける人がいますが、意外と酔っぱらいの歩き方でしかない場合もあります。
こういうのを英語で random walk と言います。和訳は「酔歩」。

誤差項に関する仮定（3）：誤差項の等分散

$x_1 = 1$ のときの誤差項を ε_1 $x_1 = 2$ のときの誤差項を ε_2 ..... $x_1 = 10$ のときの誤差項を ε_{10} と表す。

このとき、

$$E[\varepsilon_1^2] = E[\varepsilon_2^2] = ... = E[\varepsilon_{10}^2] = \sigma^2$$

さきほどの誤差項 er からその期待値を引いて期待値 0 とした er0 で見てみる。

x1 の値ごとに er0 の分散 var をとるには、 tapply 関数を使って、以下のようにする。

```
> tapply(er0,x1,var)
```

```
       1        2        3        4        5        6        7        8
161.4675 177.0393 160.1131 174.1419 169.4937 160.8374 153.4445 158.8849
       9       10
161.1862 166.3787
```

ほぼ等分散が成立している。

しかし、これは結構大きな仮定。

たとえば、中古車プリウスのオークション価格は、走行距離が小さいときは、かなり価格にバラツキがあったが、走行距離が大きくなると、あまり価格差がない。

こういうのは、等分散の仮定に反している。

仮定（2）と仮定（3）を合わせて以下で表すこともある。

$$\begin{bmatrix} E[\varepsilon_1^2] & E[\varepsilon_1\varepsilon_2] & \cdots & E[\varepsilon_1\varepsilon_n] \\ E[\varepsilon_2\varepsilon_1] & E[\varepsilon_2^2] & \cdots & E[\varepsilon_2\varepsilon_n] \\ \vdots & \vdots & \ddots & \vdots \\ E[\varepsilon_n\varepsilon_1] & E[\varepsilon_n\varepsilon_2] & \cdots & E[\varepsilon_n^2] \end{bmatrix} = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} = \sigma^2 I^{(n)}$$

ただし $I^{(n)}$ は、$n \times n$ の単位行列

誤差項に関する仮定（4）：説明変数と独立

説明変数と誤差項が独立だということは、式で表すと以下のようになる。

$$E[x_1\varepsilon] = 0$$

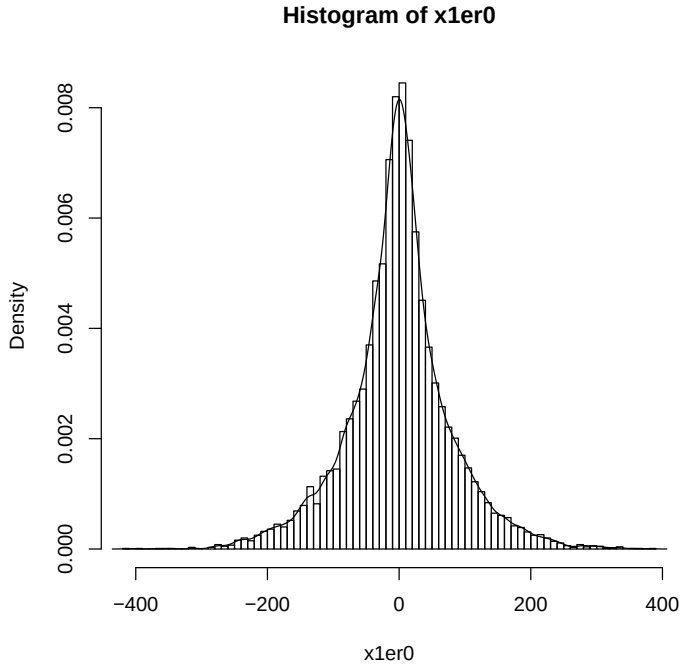
独立だということは、定義により、

$$E[x_1\varepsilon] = E[x_1]E[\varepsilon]$$

であって、$E[\varepsilon] = 0$ なので。

x1 と er0 との積を求め、その分布を確認してみる。

```
> x1er0<-x1*er0
> hist(x1er0,prob=T,br="fd")
> lines(density(x1er0))
```



期待値は

```
> mean(x1er0)
```

```
[1] -0.4164681
```

わずかに 0 からずれているが、分布のバラツキからいって、ほぼ 0 だ！

説明変数と誤差項が独立でないのは、被説変数の値が独立変数に影響を与えているような場合がある（こういうのを「内生性」という）

たとえば、犯罪が多い地域では、警官の投入人数が多いが、これをプロットすると、まるで警官の投入人数が多いほど、犯罪が多く起こっているように見える。

実際には、同じ程度の犯罪の多さの地域では、投入人数が多いほど犯罪は少ないはずなのだが、それでも、もともと犯罪の少ない地域よりは多いということ。

犯罪の発生数を y、警官の投入数を x_1 とするとき、犯罪の発生数は、

$$y = 14 - x_1 + \varepsilon$$

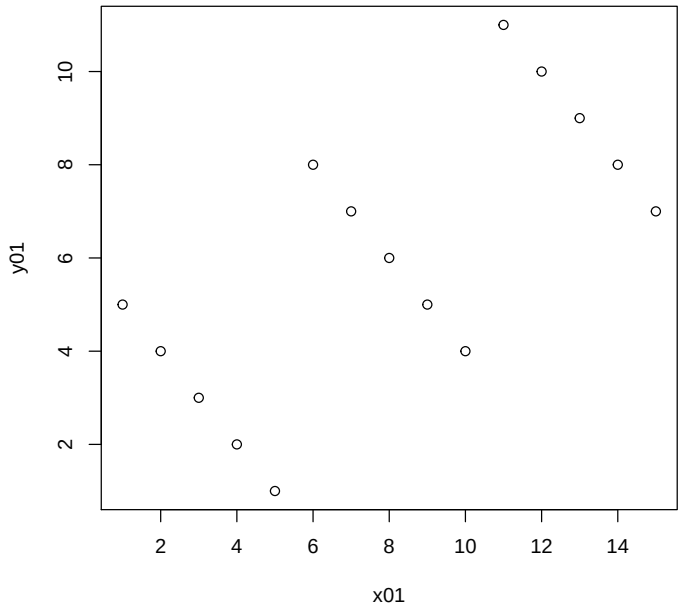
で減るものとしよう。

しかし、その水準はもともとの発生数に依存する。

具体的には以下のような関係にあるとする。

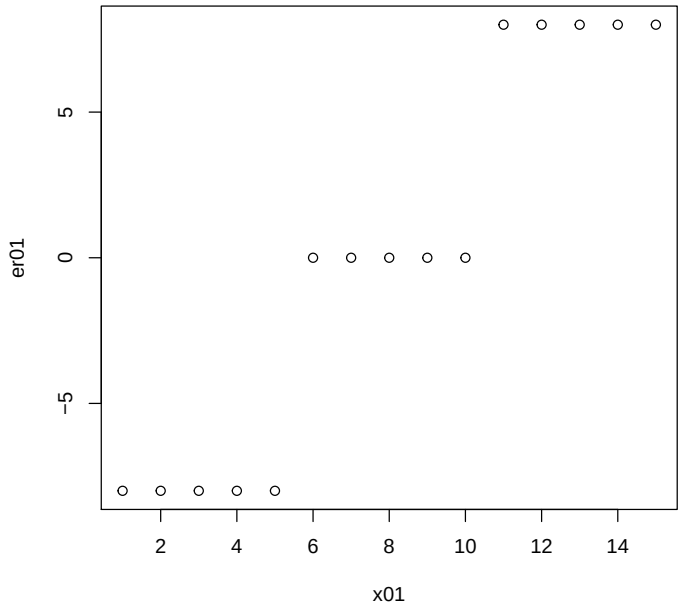
横軸が、一定エリア面積当たりの警官の投入人数、縦軸が一日当たりの犯罪の派生件数（もちろん架空のデータ）。

```
> x01<-c(1:15)
> y01<-c(5,4,3,2,1,8,7,6,5,4,11,10,9,8,7)
> plot(y01~x01)
```



この場合、誤差項 ε の分布は、

```
> er01<-y01-(14-x01)
> plot(er01~x01)
```



確かに、期待値は 0 だが、x_1 と独立ではない。

```
> mean(er01)
```

```
[1] 0
```

```
> mean(x01*er01)
```

```
[1] 26.66667
```

線形性

OLS で回帰できる方程式は線形である。

線形というのは、推定すべきパラメータについて線形であるということで、説明変数についてではない。

たとえば、次は当然線形。

$$y = \alpha + \beta_1 x_1 + \varepsilon$$

これも線形

$$y = \alpha + \beta_1 x_1^2 + \varepsilon$$

これも線形

$$y = \alpha + \beta_1 \ln x_1 + \varepsilon$$

これも線形

$$\ln y = \alpha + \beta_1 x_1 + \varepsilon$$

これは線形じゃない

$$y = \frac{\beta_1 x_1}{\alpha + x_1} + \varepsilon$$

これは線形ではないが、線形に変換できる。

$$y = A x^{\beta_1} x_2^{\beta_2} u$$

$$\ln y = \alpha + \beta_1 \ln x_1 + \beta_2 \ln x_2 + \varepsilon$$

6.1.3　標本データ

以上、1000 個という母集団、あるいは母集団に近いデータで、話をしてきたが、実際には、限られたデータで分析しなければならないことが多い。

説明変数が確率変数の場合

回帰分析では、２とおりの標本の取り方を想定する。

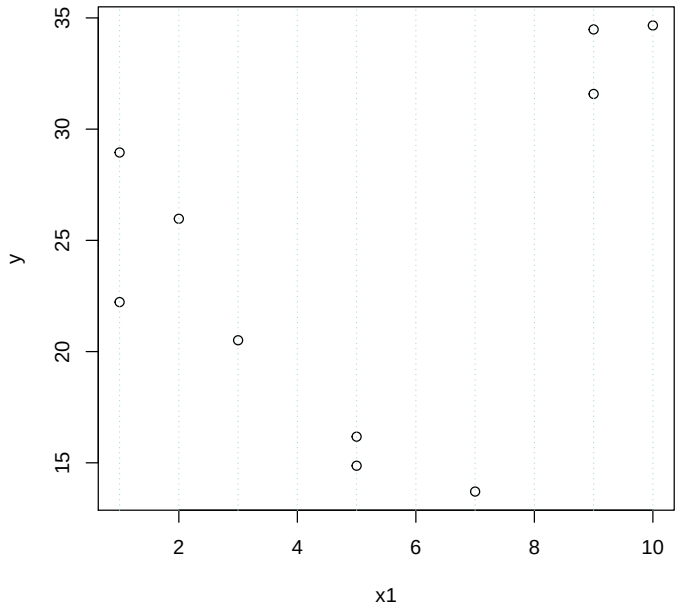
ひとつは、たくさんある x_1 と y のデータセットからランダムにいくつかのデータを抽出する方法。

たとえば、10 個だけデータを無作為抽出するには以下のようにする。

```
> d01<-data.frame(y,x1)
> (s01s<-d01[sample(1:10000,10),])
```

```
            y x1
2516 13.70755  7
6646 34.66039 10
1237 16.17510  5
4847 28.94980  1
9313 25.97013  2
3132 22.22568  1
9944 14.86829  5
505  20.50947  3
3927 31.58149  9
3219 34.48164  9
```

```
> plot(y~x1,s01s)
> abline(v=1:10,lty=3,col="lightblue")
```



x1 には 1,2,...10 の 10 種類のデータがあるが、選ばれていない数もあるし、同じ数でも 2 回選ばれているものもある。

データの取り方としては、こちらの方が素直だろう。

ただし、この場合、説明変数 x_1 を確率変数として扱う必要がある。

そうなると、証明とかが急にややこしくなる。

説明変数が確定変数の場合

もうひとつは、最初に分析すべき x_1 をあらかじめ定め、それに対応する（たくさんある）y の中から標本を抽出する方法。

例えば、x1 が 1 であるデータセットの中からランダムに 1 つサンプルを選び、次に x1 が 2 であるデータセットの中らかも 1 つサンプルを選び、... x1 が 10 であるデータセットの中から 1 つのサンプルを選び、合計 10 個のデータセットを得ることをしてみよう。

一つずつ、sample(d01[x1==1,],1) 、sample(d01[x1==2,],1) としてもよいのだが、面倒くさいので tapply 関数を使う。

ただし、 sample 関数は 2 つの引数をとるので、標本の個数 nko を予め入れた新たな関数 f01s として再定義して使っている。

3 行目は、データフレーム d01 のうち、 unlist(by(c(1:10000),x1,f01s)) で指定された行数を抽出している。

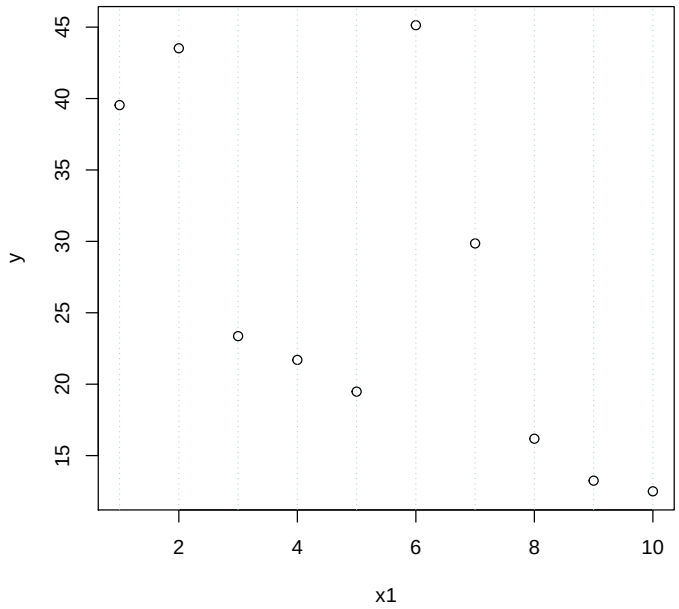
by(c(1:10000),x1,f01s) は、 x1 の値ごとに、 c(1:10000) のベクトルに関数 f01s を施せ、という意味で、この場合、 nko で指定した標本の個数（1 個）を選べということになる。

ただし、 by 関数は出力がリスト形式なので、 unlist 関数を使ってベクトル形式に変換しなければならない。

```
> nko<-1
> f01s<-function(x) sample(x,nko)
> (s01c<-d01[unlist(by(c(1:10000),x1,f01s)),])
```

```
            y x1
5285 39.53952  1
4173 43.51964  2
5327 23.36424  3
4880 21.69942  4
9041 19.47991  5
7845 45.13433  6
2929 29.85372  7
9661 16.18185  8
6419 13.24416  9
3176 12.49946 10
```

```
> plot(y~x1,s01c)
> abline(v=1:10,lty=3,col="lightblue")
```



x1 は 1〜10 の整数だが、それぞれに 1 個ずつデータが選ばれている。

こうした標本抽出の方法であれば、説明変数は確定変数であり、推定量の性質の証明などで扱いやすくなる。

確定変数ということは、x_1 と決まれば、それ以上ばらつかないということ。$E[x_1] = x_1$。

確率変数ということは、サンプルによってばらつく可能性があるということ。期待値は x_1 とは別の値の可能性がある。$E[x_1] = \bar{x}$

多少違和感を感じるデータかもしれないが、たとえば定点調査などではこのようなデータとなる。

念のため確認しておくが、ここでは説明を簡単にするために、説明変数を整数にしたが、実数値ならなんでもよいし、こんなにきれいに並ばなくてもよい。説明変数が確定的に決まっているということだけが大事。

しばらくは、得られたデータがこちらの方法で抽出されたものとして話を進める。

6.1.4　最小二乗推定量

標本データの例

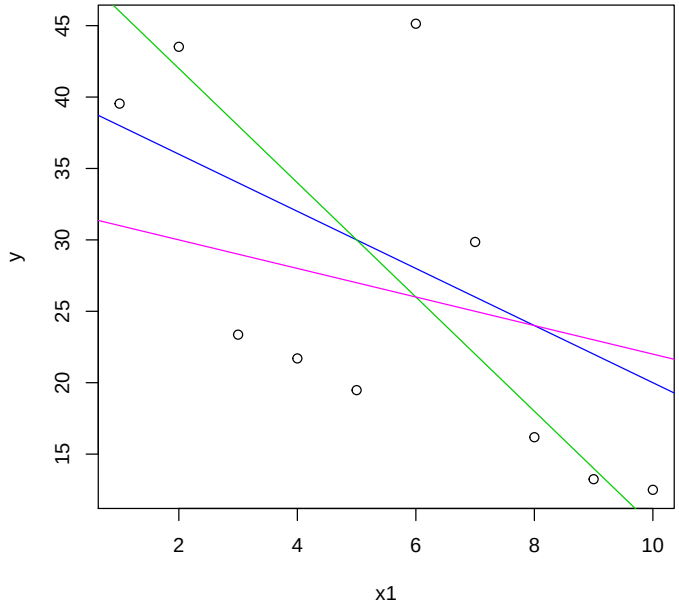
さきほど抽出した 10 個のデータ s01c を使って話をすすめよう。

```
> s01c
```

```
            y x1
5285 39.53952  1
4173 43.51964  2
5327 23.36424  3
4880 21.69942  4
9041 19.47991  5
7845 45.13433  6
2929 29.85372  7
9661 16.18185  8
6419 13.24416  9
3176 12.49946 10
```

このデータの散布図を説明する直線として、いろいろな可能性が考えられる。

```
> plot(y~x1,s01c)
> abline(40,-2,col=4)
> abline(50,-4,col=3)
> abline(32,-1,col=6)
```



最小二乗推定量の導出

この直線を一般に

$$\hat{y} = a + bx_1$$

と表す。

これは a と b 次第でなんとでもなる。

その a と b の中から、y と $\hat{y}$ の差（残差）の二乗和

$$S = \sum_{i=1}^{n} (y_i - a - bx_{1i})^2$$

を最小にするものを選ぶ。（ここで、n はデータセットの数で、上の例だと $n = 10$）。

すなわち、a と b で S を偏微分したものが 0 となる a と b を選択すればよい。（偏微分というのは、微分すべき変数が２つ以上ある場合、他の変数をただの数値と見なして微分するだけのこと）

$$\frac{\partial S}{\partial a} = -2 \sum_{i=1}^{n} (y_i - a - bx_{1i}) = 0$$

$$\frac{\partial S}{\partial b} = -2\sum_{i=1}^{n} x_{1i}(y_i - a - bx_{1i}) = 0$$

こうして得られた a、b が最小二乗推定量で、これを $\hat{\alpha}, \hat{\beta}_1$ で表す。

書き換えると、

$$\sum_{i=1}^{n}(y_i - \hat{\alpha} - \hat{\beta}_1 x_{1i}) = 0$$

$$\sum_{i=1}^{n} x_{1i}(y_i - \hat{\alpha} - \hat{\beta}_1 x_{1i}) = 0$$

となる。

これらは行列で表記すると、以下のようになる。

$$\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ \vdots & \vdots \\ 1 & x_{1n} \end{bmatrix} \begin{bmatrix} \hat{\alpha} \\ \hat{\beta}_1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \end{bmatrix} \begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ \vdots & \vdots \\ 1 & x_{1n} \end{bmatrix} \begin{bmatrix} \hat{\alpha} \\ \hat{\beta}_1 \end{bmatrix} = 0$$

さらに、

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, x = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ \vdots & \vdots \\ 1 & x_{1n} \end{bmatrix}, x' = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{11} & x_{12} & \dots & x_{1n} \end{bmatrix}$$

$$\hat{\beta} = \begin{bmatrix} \hat{\alpha} \\ \hat{\beta}_1 \end{bmatrix}$$

と表すと、

$$x'y = x'x\hat{\beta}$$

これは、以下のように解くことができる。

$$\hat{\beta} = (x'x)^{-1}x'y$$

（ここで、$(x'x)^{-1}$ は、行列 $x'x$ の逆行列。）

これが最小二乗推定量。

R での行列計算の基本

確認してみよう。

行列の積や逆行列を自分で計算しようとすればたいへんだが、R でも行列計算ができる。

さきほど抽出した 10 個のデータセット s01c から行列 y と x を作成する。

```
> (y<-matrix(s01c$y,,1))
```

```
          [,1]
 [1,] 39.53952
 [2,] 43.51964
 [3,] 23.36424
 [4,] 21.69942
 [5,] 19.47991
 [6,] 45.13433
 [7,] 29.85372
 [8,] 16.18185
 [9,] 13.24416
[10,] 12.49946
```

```
> (x<-cbind(rep(1,10),s01c$x1))
```

```
      [,1] [,2]
 [1,]    1    1
 [2,]    1    2
 [3,]    1    3
 [4,]    1    4
 [5,]    1    5
 [6,]    1    6
 [7,]    1    7
 [8,]    1    8
 [9,]    1    9
[10,]    1   10
```

行列 x の転置は t 関数を使う。x' は、

```
> t(x)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    1    1    1    1    1    1    1     1
[2,]    1    2    3    4    5    6    7    8    9    10
```

行列の積は

```
> t(x)%*%x
```

```
     [,1] [,2]
[1,]   10   55
[2,]   55  385
```

逆行列は solve 関数で行う。$(x'x)^{-1}$ は、

```
> solve(t(x)%*%x)
```

```
            [,1]        [,2]
[1,]  0.46666667 -0.06666667
[2,] -0.06666667  0.01212121
```

行列計算で最小二乗推定量

最小二乗推定量は、

$$\hat{\beta} = (x'x)^{-1}x'y$$

これを R で計算するには、上の方法をつなげて、以下で計算できる。

```
> solve(t(x)%*%x)%*%t(x)%*%y
```

```
          [,1]
[1,] 41.154410
[2,] -2.673233
```

lm 関数で確認してみる。

```
> lm(y~x1,s01c)
```

```
Call:
lm(formula = y ~ x1, data = s01c)

Coefficients:
(Intercept)           x1
     41.154       -2.673
```

一致している！

最小二乗推定量の不偏性

$$E[\hat{\beta}] = \beta$$

であることを示す。

真の回帰式を行列標記で表すと、

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ \vdots & \vdots \\ 1 & x_{1n} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

これを以下のように表す。

$$y = x\beta + \varepsilon$$

ということは、両辺に左から x' を掛けて

$$x'y = x'x\beta + x'\varepsilon$$

さらに、左から $(x'x)^{-1}$ を掛けると、β の前は $(x'x)^{-1}x'x$ となり、これは単位行列になるから、

$$(x'x)^{-1}x'y = \beta + (x'x)^{-1}x'\varepsilon$$

ここで、左辺は $\hat{\beta}$ そのもの！

$$\hat{\beta} = \beta + (x'x)^{-1}x'\varepsilon$$

誤差項に関する仮定（1）から、$E[\varepsilon_1] = ... = E[\varepsilon_n] = 0$ で、かつ x は確定した変数なので、

$$E[\hat{\beta}] = \beta$$

つまり、最小二乗推定量 $\hat{\beta}$ は、（本当の係数 β とは別物かもしれないが） 期待値が β と一致する。

ちなみに、x が確定した変数ではなくて、確率変数だとしても、誤差項に関する仮定（4）が成り立てば、$E(x'\varepsilon) = 0$ となり、この関係は成り立つ。

最小二乗推定量の分布

$\hat{\beta}$ で、期待値 $E[\hat{\beta}]$ を云々しているということは、$\hat{\beta}$ にバラツキがあるということ。

それではどんな分布をするのか、確認してみよう。

s01c は 10000 個のデータから、各 $x_1 = (x_11, x_12, ..., x_{10})$ の 10 個に対応する $y = (y_1, y_2, ..., y_{10})$ を取り出したものだった。

```
> s01c
```

```
            y x1
5285 39.53952  1
4173 43.51964  2
5327 23.36424  3
4880 21.69942  4
9041 19.47991  5
7845 45.13433  6
2929 29.85372  7
9661 16.18185  8
6419 13.24416  9
3176 12.49946 10
```

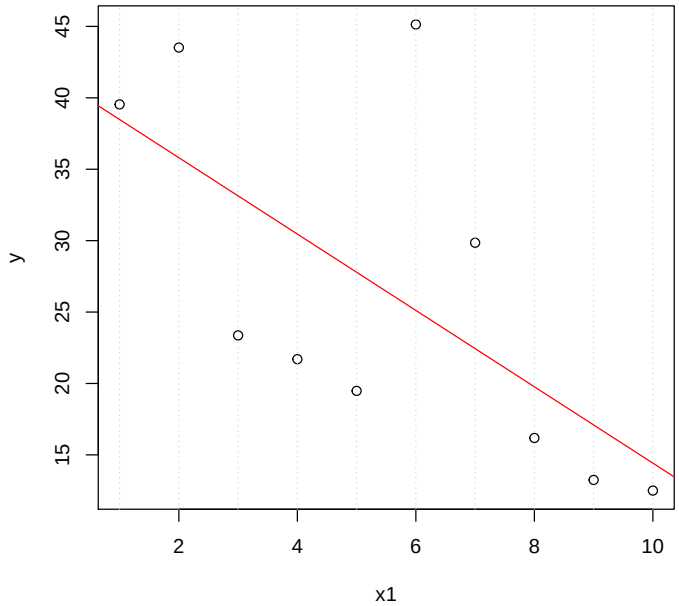
これで最小二乗推定量を計算すると、

```
> (o01c<-lm(y~x1,s01c))
```

```
Call:
lm(formula = y ~ x1, data = s01c)

Coefficients:
(Intercept)           x1
     41.154       -2.673
```

だった。

```
> plot(y~x1,s01c)
> abline(v=1:10,lty=3,col="lightblue")
> abline(o01c,col=2)
```



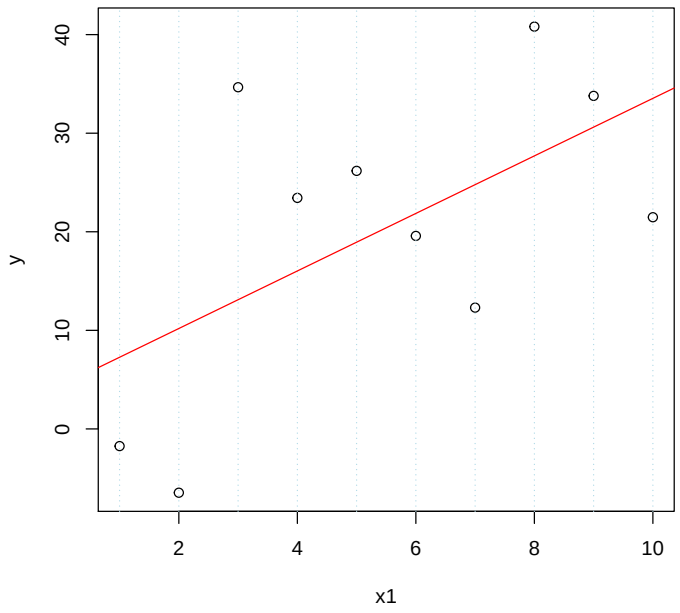
しかし、これはあくまで「たまたま」とった標本のデータでしかない。
別の取り方をすれば別の推定値がでるはずだ。
やってみよう。
$(x_{11}, x_{12}, ..., x_{1,10})$ ごとに、別のデータ $(y_1, y_2, ..., y_{10})$ を取り出す。

この関数は f01s という名前で勝手に定義していた。

```
> nko<-1
> (s01c1<-d01[unlist(by(c(1:10000),x1,f01s)),])
```

```
            y x1
5676 -1.741963  1
6830 -6.470519  2
2969 34.662377  3
5529 23.438510  4
6978 26.179282  5
4097 19.579334  6
4100 12.305589  7
6858 40.811585  8
8112 33.785793  9
1129 21.474674 10
```

```
> plot(y~x1,s01c1)
> abline(v=1:10,lty=3,col="lightblue")
> o01c1<-lm(y~x1,s01c1)
> abline(o01c1,col=2)
```



おなじ母集団から同じように抽出したのに、全然別の最小二乗推定量が出てきた。

母集団 10000 個のデータセットからなり、各 $x_1 = (x_1 1, x_1 2, ..., x_{10})$ について 1 個ずつの標本を取り出したとして、(x_i にデータが均等にあったとすれば)、1000 の 10 乗通りという天文学的組み合わせがある。

それに応じていろいろな $\hat{\beta}$ もありうる。

さすがに全部は計算できないので、このうちの 1000 パターンについて $\hat{\beta}$ を計算する。

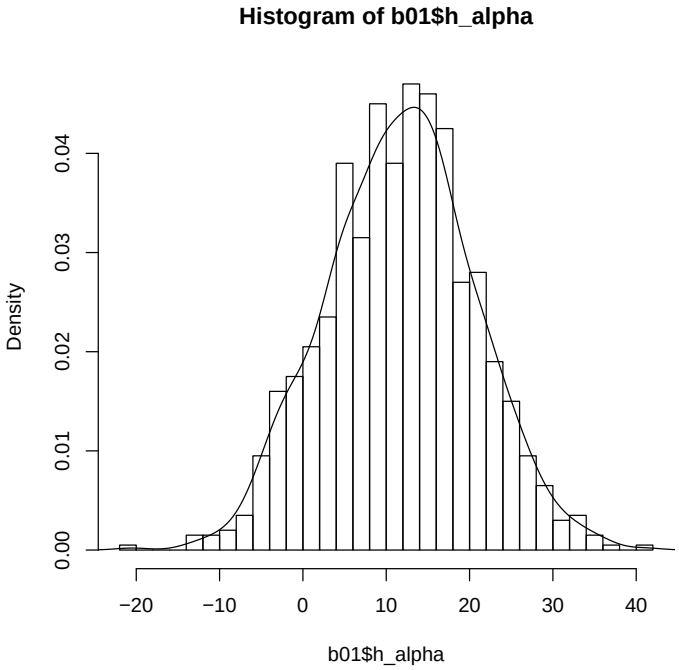
```
> b01<-matrix(NA,1000,2)
> nko<-1
> for(i in 1:1000){
+   s01r<-d01[unlist(by(c(1:10000),x1,f01s)),]
+   o01r<-lm(y~x1,s01r)
+   b01[i,]<-coef(o01r)
+ }
```

```
> b01<-data.frame(b01)
> names(b01)<-c("h_alpha","h_beta1")
> summary(b01)
```

```
    h_alpha           h_beta1
Min.   :-20.486   Min.   :-2.7783
1st Qu.:  5.607   1st Qu.: 0.9957
Median : 11.973   Median : 1.9929
Mean   : 11.674   Mean   : 1.9560
3rd Qu.: 17.427   3rd Qu.: 2.9192
Max.   : 41.064   Max.   : 6.9042
```

$\hat{\alpha}$ の推定値の分布は、

```
> hist(b01$h_alpha,prob=T,br="fd")
> lines(density(b01$h_alpha))
```



平均は

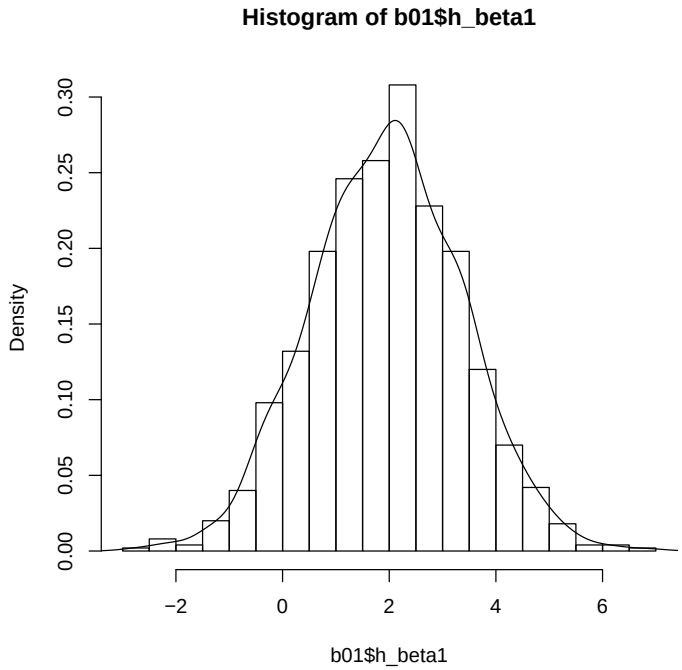
```
> mean(b01$h_alpha)
```

```
[1] 11.67358
```

$\alpha = 11$ だったので、$E[\hat{\alpha}] = \alpha$ はほぼ成立している。

$\hat{\beta}_1$ の推定値の分布は、

```
> hist(b01$h_beta1,prob=T,br="fd")
> lines(density(b01$h_beta1))
```



平均は

```
> mean(b01$h_beta1)
```

```
[1] 1.956027
```

$\beta_1 = 2$ だったので、$E[\hat{\beta}_1] = \beta_1$ はほぼ成立している。

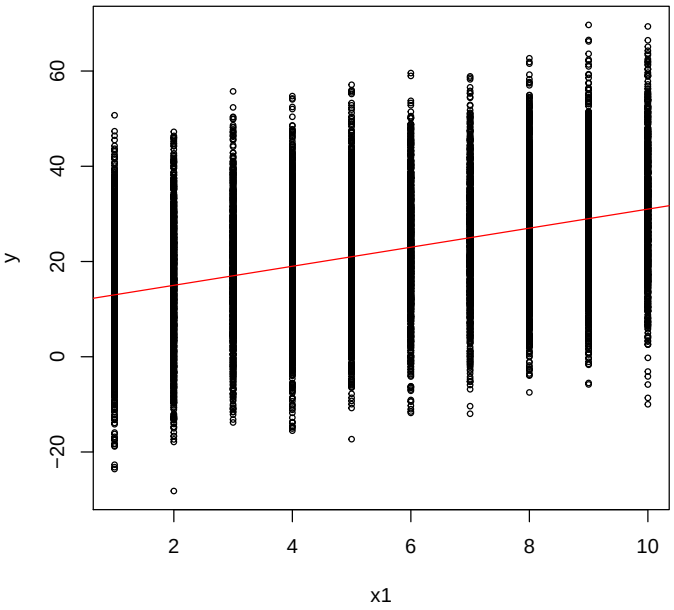
最小二乗推定量の一致性

たしかに最小二乗推定量は不偏だが、それにしても推定量のバラツキが大きい。

たとえば、β_1 の推定値は、確かに平均は 2 前後で β_1 に一致するかもしれませんが、中には正負が逆の推定値までである。

確かに、もとの分布を見たら、変数 x_1 で説明できる部分はわずかで、ほとんどが誤差項によるものだった。

```
> plot(y~x1,cex=.6,d01)
> abline(11,2,col=2)
```



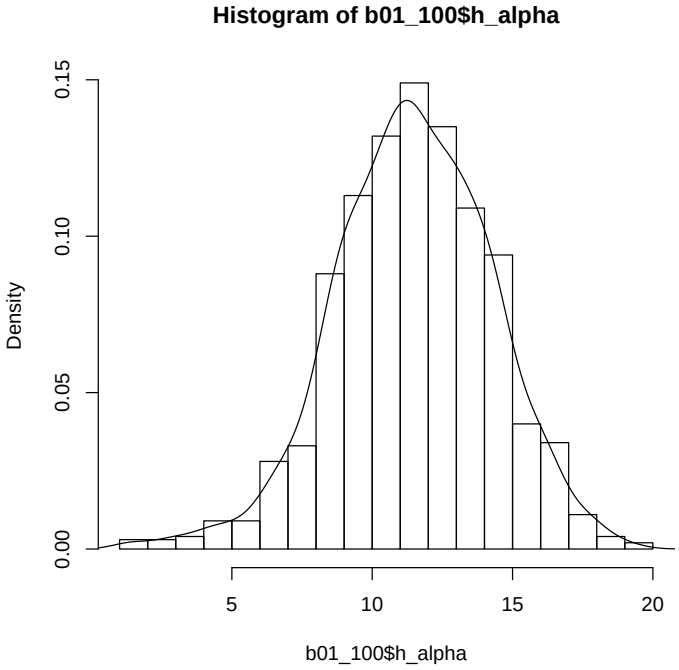
こんなバラツキがあるのに、10 個だけのデータで x_1 の係数を推定するというのが難しい話だ。
もうちょっとデータを増やしてみよう！
100 個データを集めたらどうなるか？
各 $x_1 = (x_1 1, x_1 2, ..., x_{10})$ について 10 個ずつ（でなくてもよいのだが）集めてみよう。

```
> b01_100<-matrix(NA,1000,2)
> nko<-10
> for(i in 1:1000){
+   s01r<-d01[unlist(by(c(1:10000),x1,f01s)),]
+   o01r<-lm(y~x1,s01r)
+   b01_100[i,]<-coef(o01r)
+ }
> b01_100<-data.frame(b01_100)
> names(b01_100)<-c("h_alpha","h_beta1")
> summary(b01_100)
```

```
    h_alpha          h_beta1
Min.   : 1.299   Min.   :0.6436
1st Qu.: 9.651   1st Qu.:1.6231
Median :11.462   Median :1.9459
Mean   :11.457   Mean   :1.9431
3rd Qu.:13.399   3rd Qu.:2.2205
Max.   :19.169   Max.   :3.4589
```

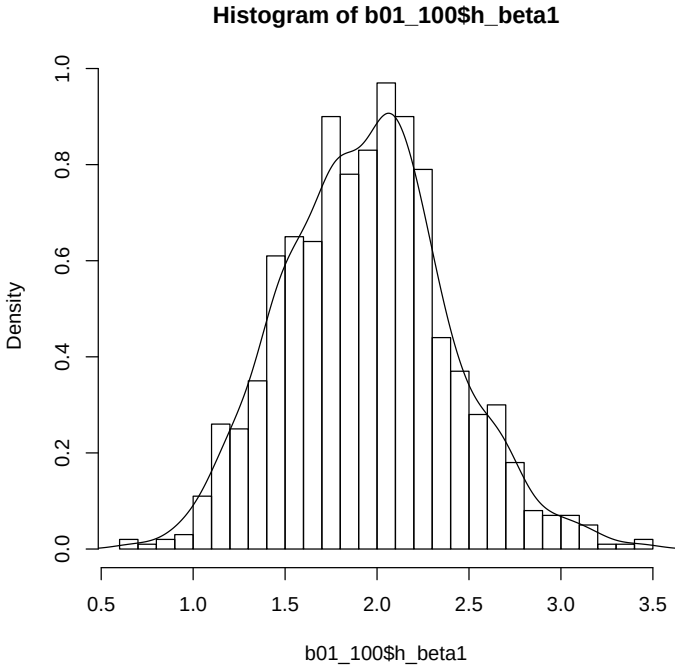
$\hat{\alpha}$ の推定値の分布は、

```
> hist(b01_100$h_alpha,prob=T,br="fd")
> lines(density(b01_100$h_alpha))
```



$\hat{\beta}_1$ の推定値の分布は、

```
> hist(b01_100$h_beta1,prob=T,br="fd")
> lines(density(b01_100$h_beta1))
```



1000 個データを集めたらどうなるか？

各 $x_1 = (x_1 1, x_1 2, ..., x_{10})$ について 100 個ずつ（でなくてもよいのだが）集めてみよう。

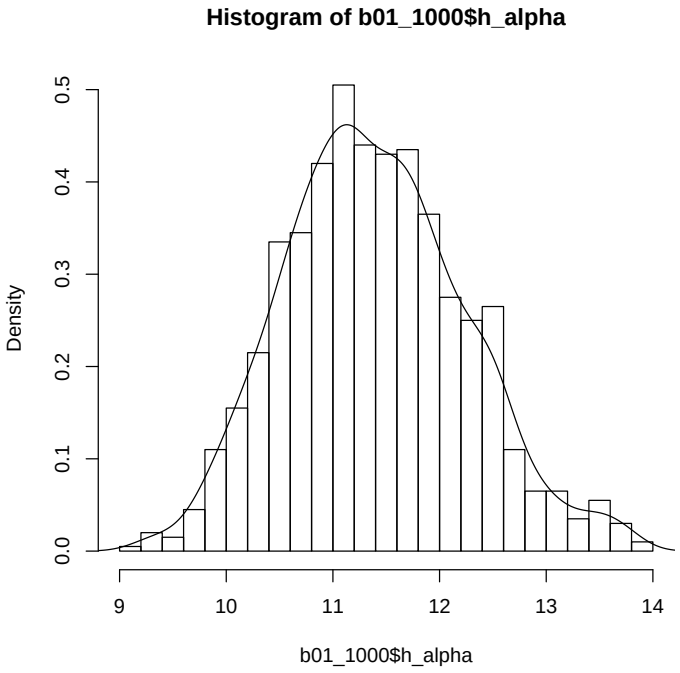
```
> b01_1000<-matrix(NA,1000,2)
> nko<-100
> for(i in 1:1000){
+   s01r<-d01[unlist(by(c(1:10000),x1,f01s)),]
```

```
+   o01r<-lm(y~x1,s01r)
+   b01_1000[i,]<-coef(o01r)
+ }
> b01_1000<-data.frame(b01_1000)
> names(b01_1000)<-c("h_alpha","h_beta1")
> summary(b01_1000)
```

```
    h_alpha          h_beta1
Min.   : 9.188   Min.   :1.437
1st Qu.:10.804   1st Qu.:1.867
Median :11.341   Median :1.963
Mean   :11.395   Mean   :1.952
3rd Qu.:11.954   3rd Qu.:2.043
Max.   :13.812   Max.   :2.367
```

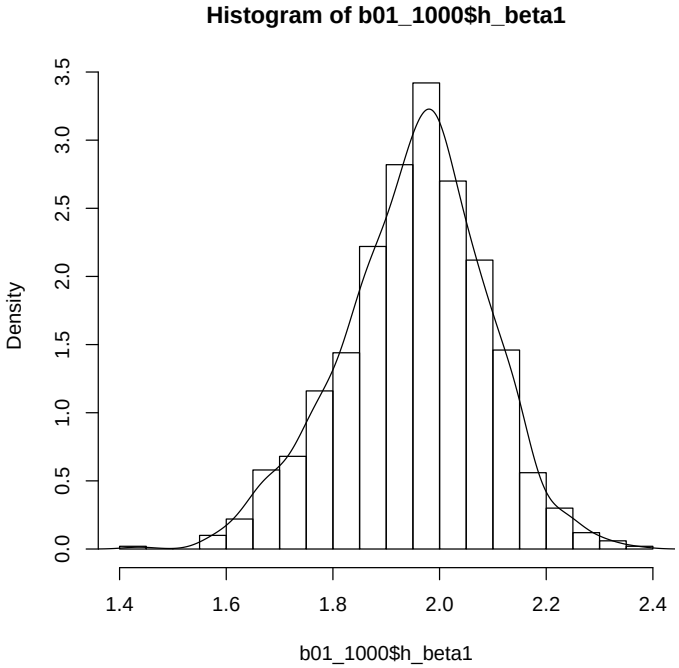
$\hat{\alpha}$ の推定値の分布は、

```
> hist(b01_1000$h_alpha,prob=T,br="fd")
> lines(density(b01_1000$h_alpha))
```



$\hat{\beta}_1$ の推定値の分布は、

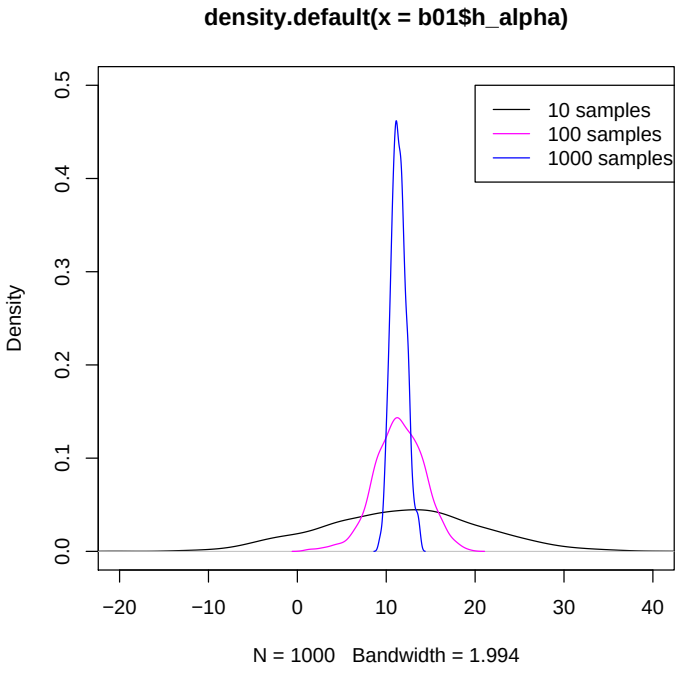
```
> hist(b01_1000$h_beta1,prob=T,br="fd")
> lines(density(b01_1000$h_beta1))
```



以上の３種の確率密度をまとめて描いてみると、以下のようになる。

$\hat{\alpha}$ の分布は

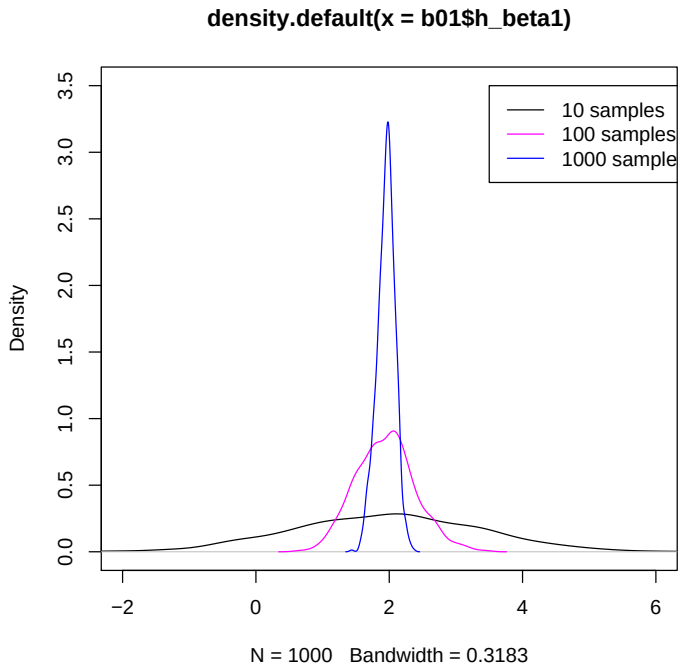
```
> plot(density(b01$h_alpha),xlim=c(-20,40),ylim=c(0,0.5))
> lines(density(b01_100$h_alpha),col=6)
> lines(density(b01_1000$h_alpha),col=4)
> legend(20,0.5,lty=1,legend=c("10 samples","100 samples","1000 samples"),col=c(1,6,4))
```



$\hat{\beta}_1$ の分布は

```
> plot(density(b01$h_beta1),xlim=c(-2,6),ylim=c(0,3.5))
> lines(density(b01_100$h_beta1),col=6)
```

```
> lines(density(b01_1000$h_beta1),col=4)
> legend(3.5,3.5,lty=1,legend=c("10 samples","100 samples","1000 samples"),col=c(1,6,4))
```



標本規模を大きくすることによって、$\hat{\alpha}$、$\hat{\beta}_1$ の分布が、$\alpha = 11$、$\beta = 2$ の周辺に集まってくる。

すなわち、最小二乗推定量 $\hat{\beta}$ は一致性を有する、ということ。

最小分散線形不偏推定量

最小二乗推定量は、線形不偏推定量の中では、最も分散が小さい推定量である。

回帰分析というのは、平たく言うと、散布図に回帰線を引く作業であるから、いろいろな線の引き方ができるはずで、最小二乗法というのはそのひとつである。

回帰式は、パラメータについて線形である場合と非線形である場合がある。これについては先述した。

最小二乗法は、線形の場合の回帰に使われる。

しかし、回帰を線形に限定しても、最小二乗法意外にも散布図に線を引く方法はある。(たとえば、残差の絶対値を最小にするなど)

その中で、なぜ最小二乗法が使われるかというと、最小二乗法推定量が不偏であるからである。

これは誤差項についての仮定(1)と(4)

$$E[\varepsilon] = 0$$

と、説明変数が確定変数、または誤差項と独立

$$E[x\varepsilon] = 0$$

で確保される。

つまり、最小二乗推定量は「線形不偏推定量」である。

しかし、他の方法でも線形不偏推定量を導くことはできる。

その中で、なぜ最小二乗法かというと、最小二乗推定量が線形不偏推定量の中で最も小さい分散だからである。

分散が小さいということは、真の値 β から大きく外れる可能性が小さいということであり、推定の効率性がよいということである。

ただし、それには条件があり、（証明は省くが）誤差項についての仮定（２）と（３）を満たす必要がある。

すなわち、誤差項が独立

$$E[\varepsilon_i \varepsilon_j] = 0 (i \neq j)$$

であり、等分散

$$E[\varepsilon_1^2] = ... = E[\varepsilon_n^2] = \sigma^2$$

であることが求められる。

すなわち、誤差項について仮定（１）～（４）が満たされる限り、最小二乗推定量は、「最小分散線形不偏推定量（BLUE）」である。

ただし、誤差項についての仮定（１）～（４）のいずれかが満たされない場合はこの限りではない。

つまり、誤差項間の相関、不等分散、説明変数との相関がある場合、なんらかの補正を加えた最小二乗法を用いるか、最小二乗法以外の方法で推定を行うことになる。（もちろん、回帰式が線形でない場合も）

6.2 推定量の分布

6.2.1 前回の作業の引き継ぎ

y は以下の方程式で。

$$y = 1 + 2x_1 - x_2 + x_3 - 2x_4 + 2x_5 - x_6 + 3x_7$$

x_1 は、1,2,...,10 の整数について観察可能で、$x_2, ..., x_7$ は、それぞれ独立に [0,1] の範囲の一様分布であるとする。

母集団として、以下の合計 10000 組のデータセットを発生させよう。

```
> set.seed(321)
> x1<-sample(1:10,10000,replace=T)
> x2<-runif(10000,0,10)
> x3<-runif(10000,0,10)
> x4<-runif(10000,0,10)
> x5<-runif(10000,0,10)
> x6<-runif(10000,0,10)
> x7<-runif(10000,0,10)
> y<-1+2*x1-x2+x3-2*x4+2*x5-x6+3*x7
```

このとき、

$$y = \alpha + \beta_1 x_1 + \varepsilon$$

という回帰式で OLS を行うことを考える。

このとき、

$$\alpha = 1, \beta_1 = 2$$

$$\varepsilon = -x_2 + x_3 - 2x_4 + 2x_5 - x_6 + 3x_7$$

となるはずだが、$E[\varepsilon] = 0$ となるように定数項を調整する。

```
> er<--x2+x3-2*x4+2*x5-x6+3*x7
> er0<-er-mean(er)
> sd(er0)
```

```
[1] 12.81745
```

mean(er) がほぼ 10 なので、推定すべき回帰式は、

$$y = 11 + 2x_1 + \varepsilon$$

ということ。

データセットの母集団は x_1 と y だけ。

これを d01 という名前のデータフレーム形式で保存する。

```
> d01<-data.frame(x1,y)
```

6.2.2 OLS 推定量の分布の形

前回、OLS 推定量は、誤差項が独立で期待値が 0 ならば不偏であり一致性を持つことがわかった。

さらに、誤差項が等分散で説明変数と独立ならば、最小分散であることもわかった。

それでは、OLS 推定量の分布はどうか？

10 個のサンプリングによる OLS 推定量

10 個の標本を 1000 通りとってみて、それぞれの標本ごとに異なるはずの OLS 推定量がどのような分布をするのか確かめてみよう。

説明変数 x_1 は、10 パターンの値をとる変数だった。それぞれに対応する y があるが、これは、同じ値をとる x_1 でも、さまざまな値をとる。

10 種類の x_1 のレベルごとに y を 1 個ずつとって、計 10 個の規模のデータセットとしよう。

そしてこのデータセットを使って OLS 回帰を行い、ひとつずつの OLS 推定量の組 $\hat{\alpha}, \hat{\beta}_1$ を得る。

これを 1000 回繰り返して、1000 組の $\hat{\alpha}, \hat{\beta}_1$ を得る。

ただし、この作業は、すでに前回、一致性の確認のところでやった。

```
> b01_10<-matrix(NA,1000,2)
> b01_10sd<-matrix(NA,1000,2)
> nko<-1
> f01s<-function(x) sample(x,nko)
> for(i in 1:1000){
+   s01r<-d01[unlist(by(c(1:10000),x1,f01s)),]
+   o01r<-lm(y~x1,s01r)
+   b01_10[i,]<-coef(o01r)
+   b01_10sd[i,]<-summary(o01r)$coeff[,"Std. Error"]
+ }
> b01_10<-data.frame(b01_10)
> b01_10sd<-data.frame(b01_10sd)
> names(b01_10)<-c("h_alpha","h_beta1")
> names(b01_10sd)<-c("h_alpha","h_beta1")
> summary(b01_10)
```

```
    h_alpha           h_beta1
 Min.   :-14.938   Min.   :-2.555
 1st Qu.:  5.432   1st Qu.: 0.913
 Median : 11.705   Median : 1.881
 Mean   : 11.498   Mean   : 1.963
 3rd Qu.: 17.759   3rd Qu.: 2.979
 Max.   : 41.116   Max.   : 6.670
```

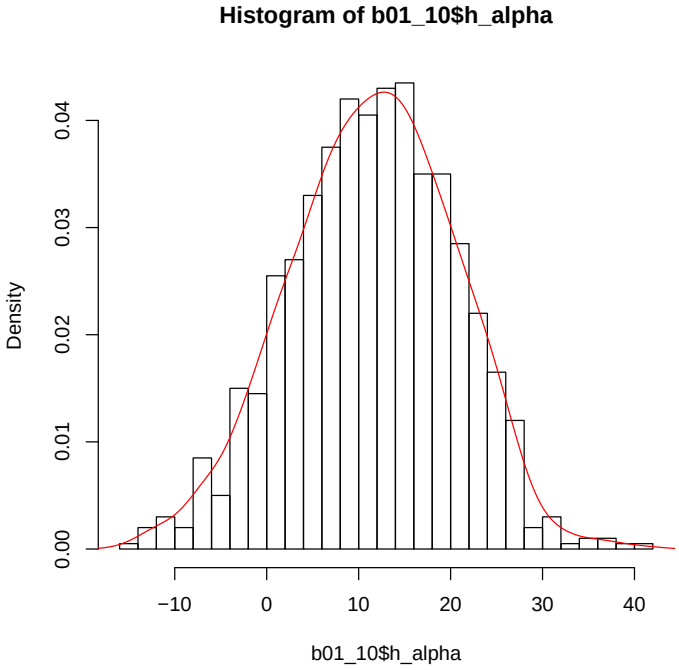
```
> summary(b01_10sd)
```

```
    h_alpha          h_beta1
 Min.   : 2.892   Min.   :0.4661
 1st Qu.: 7.117   1st Qu.:1.1470
 Median : 8.462   Median :1.3638
 Mean   : 8.505   Mean   :1.3707
 3rd Qu.: 9.891   3rd Qu.:1.5940
 Max.   :14.701   Max.   :2.3693
```

分布を確認する

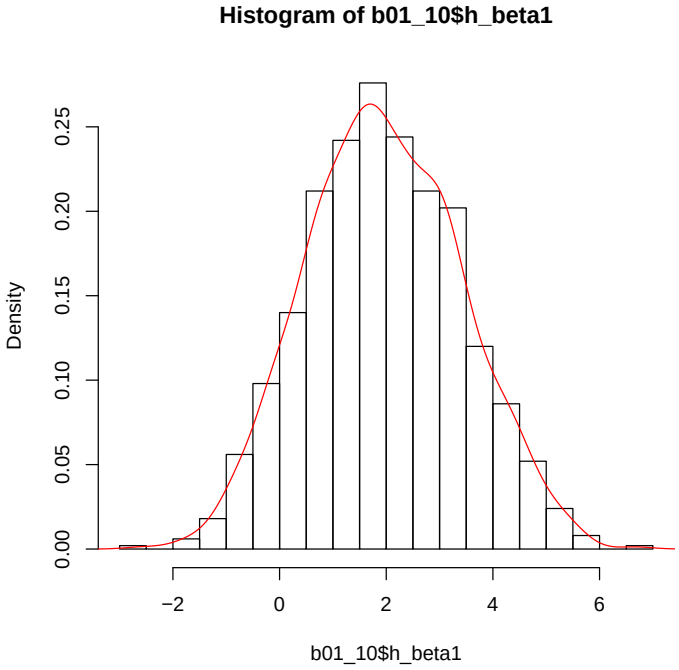
$\hat{\alpha}$ の分布は

```
> hist(b01_10$h_alpha,prob=T,br="fd")
> lines(density(b01_10$h_alpha),col=2)
```



$\hat{\beta}_1$ の分布は

```
> hist(b01_10$h_beta1,prob=T,br="fd")
> lines(density(b01_10$h_beta1),col=2)
```



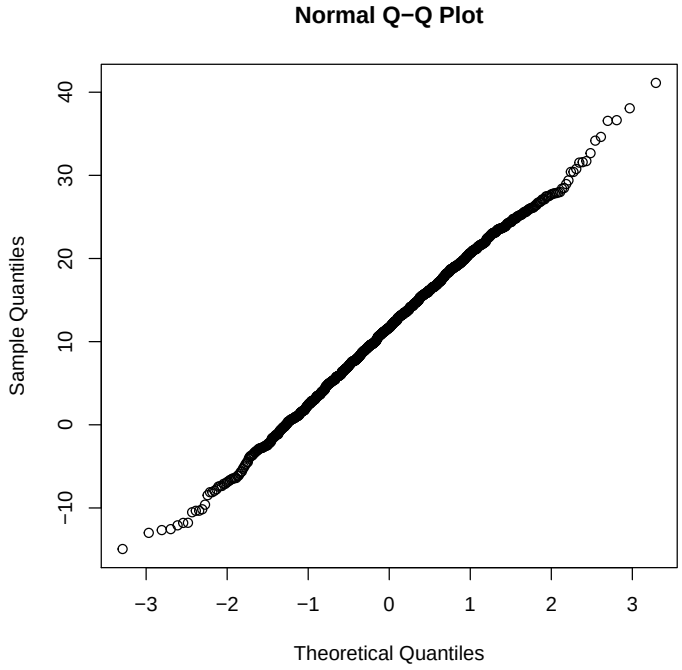
正規分布に近いように見える。

qqnorm 関数で確認してみる。

qqnorm 関数で表示される分布が直線に近いほど、分布は正規分布に近い。

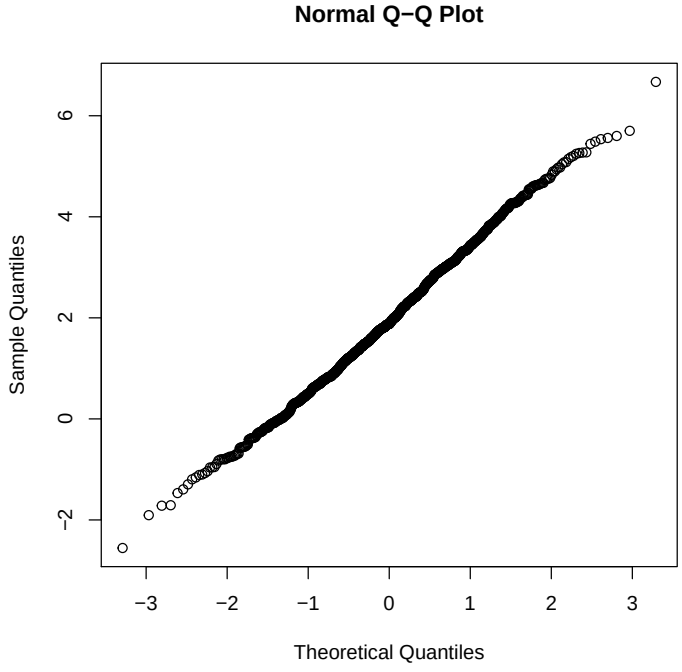
$\hat{\alpha}$ の分布は

```
> qqnorm(b01_10$h_alpha)
```



$\hat{\beta}_1$ の分布は

```
> qqnorm(b01_10$h_beta1)
```



ほぼ正規分布のようだ。

OLS 推定量は正規分布をするのか？

OLS 推定量の分散を確認する

前回見たように、この OLS 推定量は不偏性をもつ。

1000 パターンの $\hat{\alpha}, \hat{\beta}_1$ の平均をとると・・・

```
> apply(b01_10,2,mean)
```

```
  h_alpha   h_beta1
11.498127  1.963346
```

α, β_1 の値とほとんど同じだ。

次に、1000 パターンの $\hat{\alpha}, \hat{\beta}_1$ の標準偏差をとると・・・

```
> apply(b01_10,2,sd)
```

```
 h_alpha  h_beta1
8.897584 1.437893
```

となる。

この値は、1000 回 OLS 推定をやって経験的にられた値であるが、それとは別に、OLS 推定を１回やっただけで係数推定値の標準誤差（ Std. Error ）というのが報告される。

回帰係数に対して誤差項の分散がかなり大きく回帰式で、標本規模も 10 個しかないという、かなり厳しい推定だが、けっこう経験的に得られた推定値の標準誤差を言い当てているはずだ。

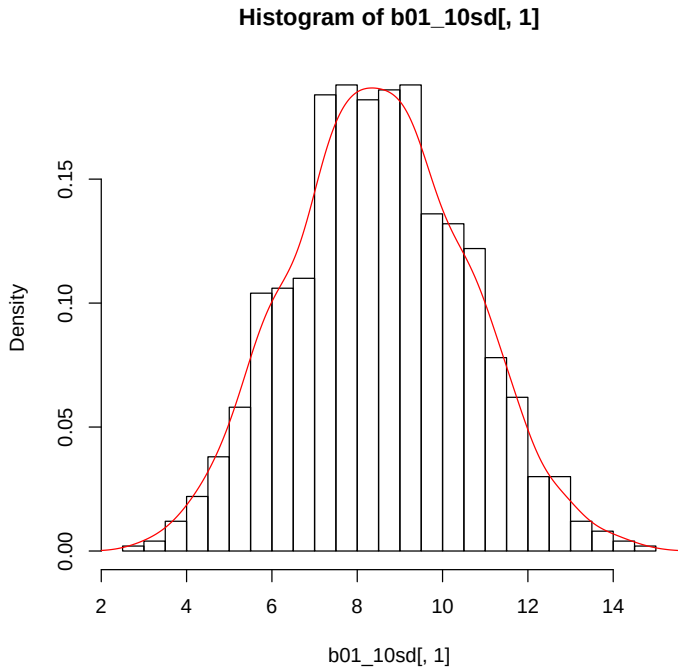
```
> summary(o01r)$coeff
```

```
            Estimate Std. Error   t value  Pr(>|t|)
(Intercept) 4.653879   9.481907 0.4908168 0.6367354
x1          3.542332   1.528148 2.3180562 0.0490681
```

とはいえ、これは理論的に出された OLS 推定値の標準偏差であり、標本のデータセットごとに違ってしまう。

理論的に出された OLS 推定値の標準誤差の分布を確認してみる。

```
> hist(b01_10sd[,1],br="fd",prob=T)
> lines(density(b01_10sd[,1]),col=2)
> hist(b01_10sd[,2],br="fd",prob=T)
> lines(density(b01_10sd[,2]),col=2)
```



理論的に推定された標準誤差は、かなりばらついていいるが、その期待値は、実際の分布の標準誤差に近いものになっている。

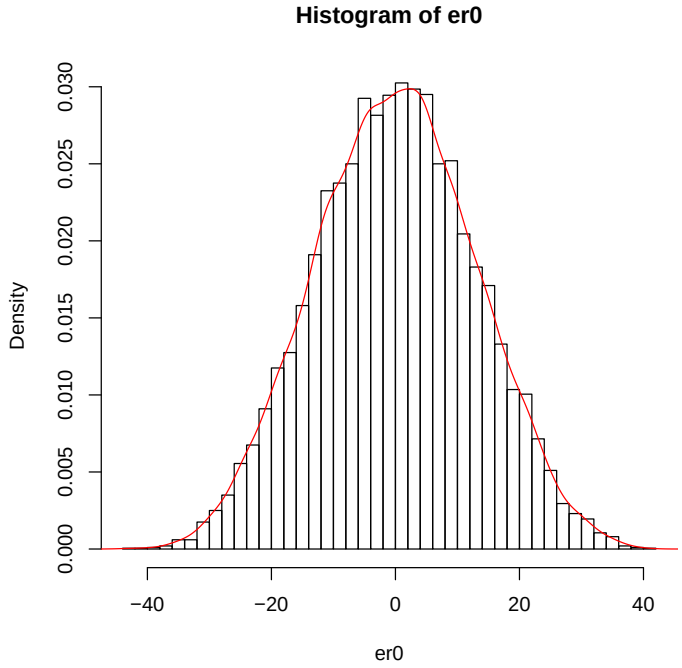
```
> apply(b01_10sd,2,mean)
```

```
 h_alpha  h_beta1
8.505145 1.370728
```

6.2.3　誤差項が正規分布からずれた場合

しかし、この推定、そもそも誤差項の分布がほぼ正規分布だった。

```
> hist(er0,prob=T,br="fd")
> lines(density(er0),col=2)
> qqnorm(er0)
```

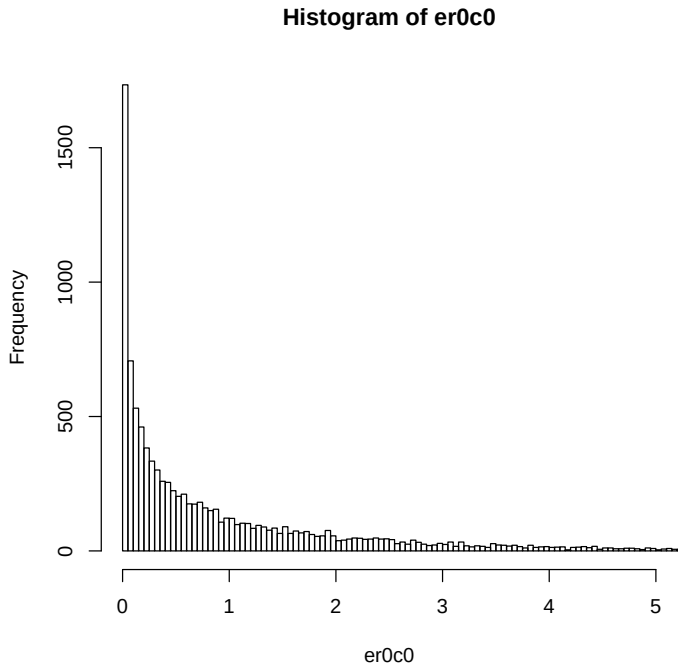


正規分布とはだいぶ違う分布を発生させてみよう。

誤差項が自由度１のカイ二乗分布に従う場合

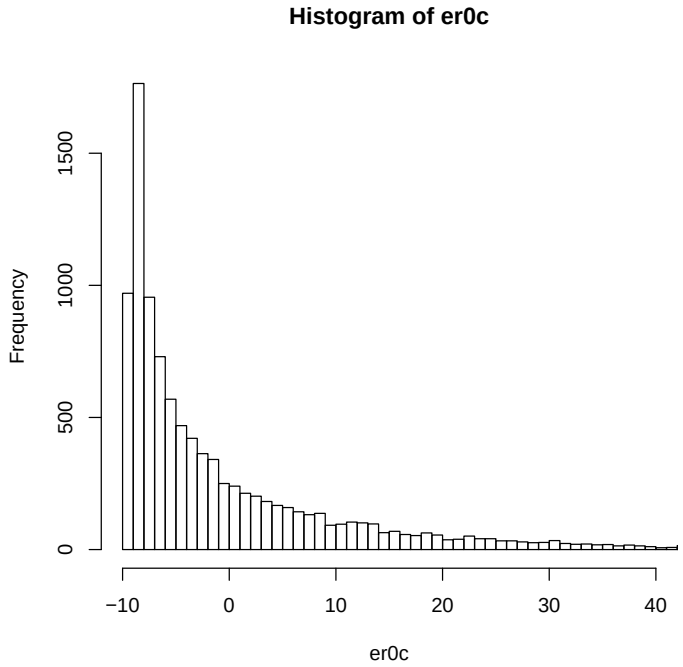
自由度１のカイ二乗分布に従う誤差項を 10000 個発生させて、これを誤差項として、先ほどの $\mathrm{er0}$ と入れ替えてみる。

```
> er0c0<-rchisq(10000,1)
> hist(er0c0,br=seq(0,20,by=0.05),xlim=c(0,5))
```



ただ、誤差項なので平均を 0 とし、上記の誤差項 $\mathrm{er0}$ と標準偏差を同じにする。

```
> er0c<-scale(er0c0)*sd(er0)
> hist(er0c,br=seq(-10,230,by=1),xlim=c(-10,40))
```



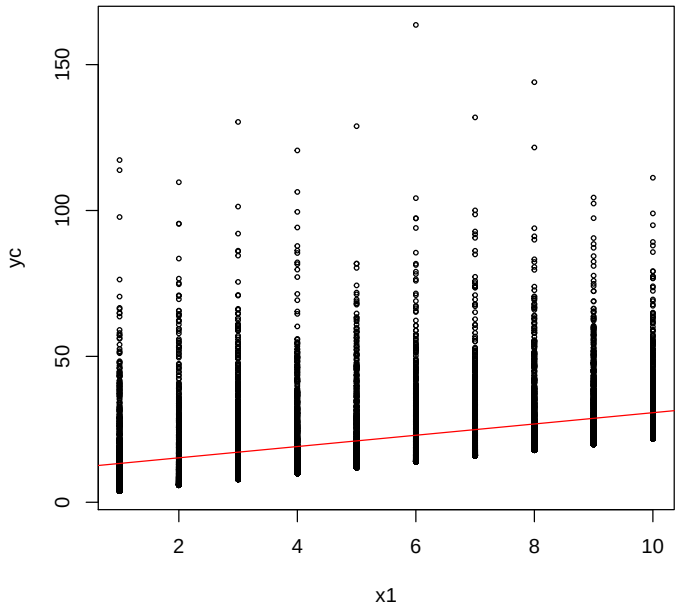
この誤差項を使って、以下の関係を満たす新たなデータセットをつくる。

$$y_c < -\alpha + \beta_1 x_1 + \varepsilon_c$$

α と β_1 はさきほどと同じだが、誤差項が入れ替わっている。

データセット (x1,yc) の母集団は以下で得られる。

```
> yc<-11+2*x1+er0c
> d01c<-data.frame(x1,yc)
> plot(yc~x1,d01c,cex=.5)
> abline(lm(yc~x1,d01c),col=2)
```



これも同様に 10 個ずつサンプリングを 1000 回繰り返して、1000 パターンの推定係数 $\hat{\alpha}_c, \hat{\beta}_{1c}$ を得る。

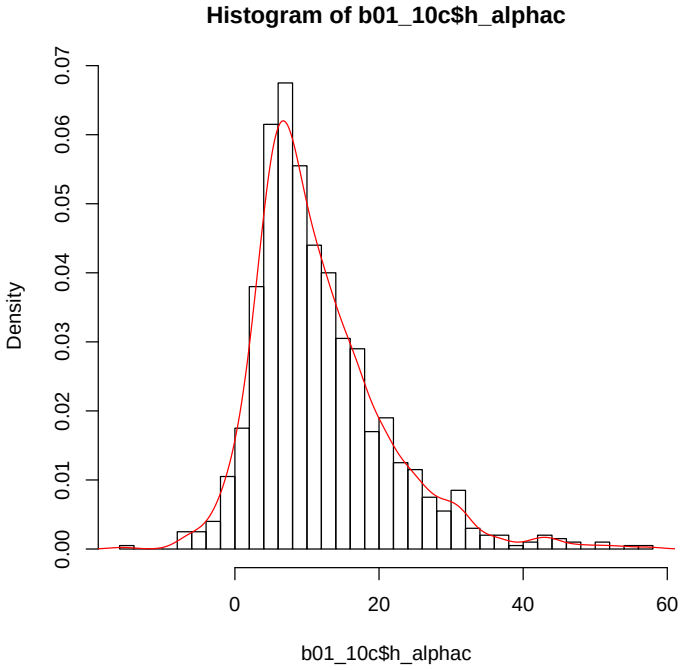
```
> b01_10c<-matrix(NA,1000,2)
> nko<-1
> for(i in 1:1000){
+   s01r<-d01c[unlist(by(c(1:10000),x1,f01s)),]
+   o01r<-lm(yc~x1,s01r)
+   b01_10c[i,]<-coef(o01r)
+ }
> b01_10c<-data.frame(b01_10c)
> names(b01_10c)<-c("h_alphac","h_beta1c")
> summary(b01_10c)
```

```
   h_alphac          h_beta1c
Min.   :-15.606   Min.   :-4.280
1st Qu.:  5.556   1st Qu.: 1.147
Median :  9.621   Median : 1.979
Mean   : 11.664   Mean   : 1.878
3rd Qu.: 16.015   3rd Qu.: 2.679
Max.   : 57.646   Max.   : 7.731
```

分布を確認する

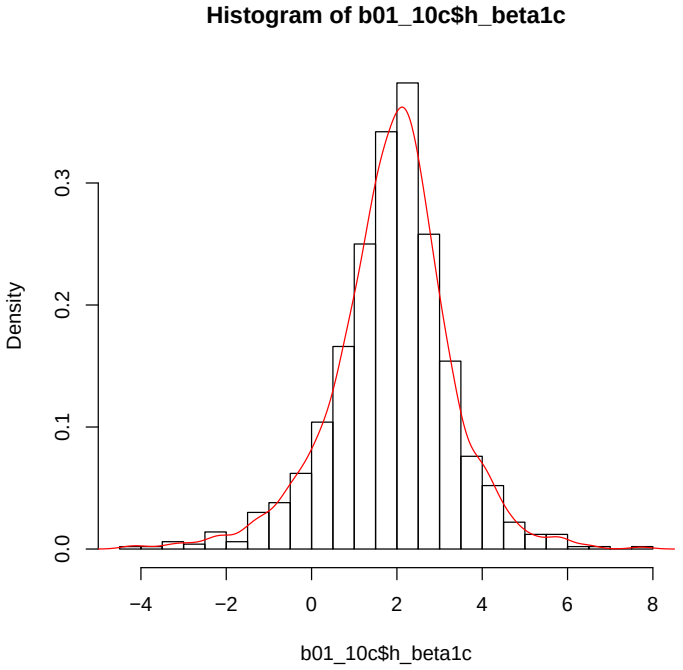
$\hat{\alpha}_c$ の分布は

```
> hist(b01_10c$h_alphac,prob=T,br="fd")
> lines(density(b01_10c$h_alphac),col=2)
```



$\hat{\beta}_{1c}$ の分布は

```
> hist(b01_10c$h_beta1c,prob=T,br="fd")
> lines(density(b01_10c$h_beta1c),col=2)
```

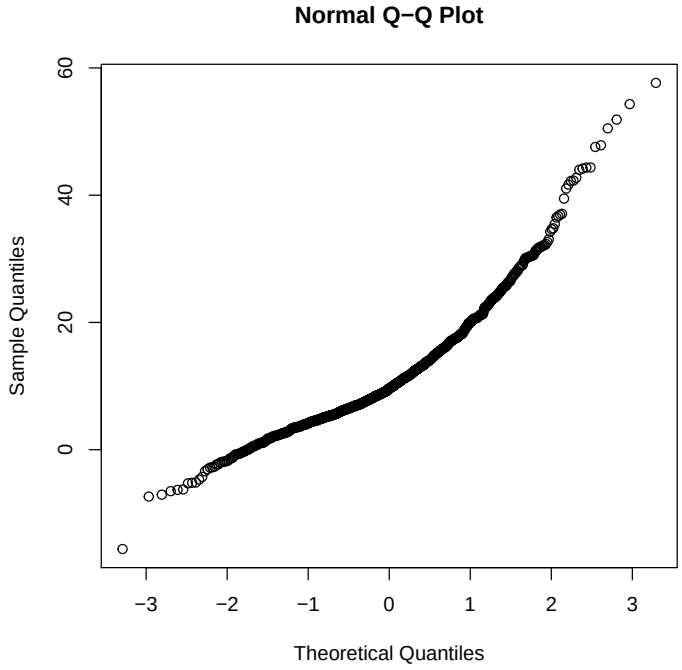


これも正規分布に近いようにも見えるが、やや歪だ。

qqnorm 関数で確認してみる。

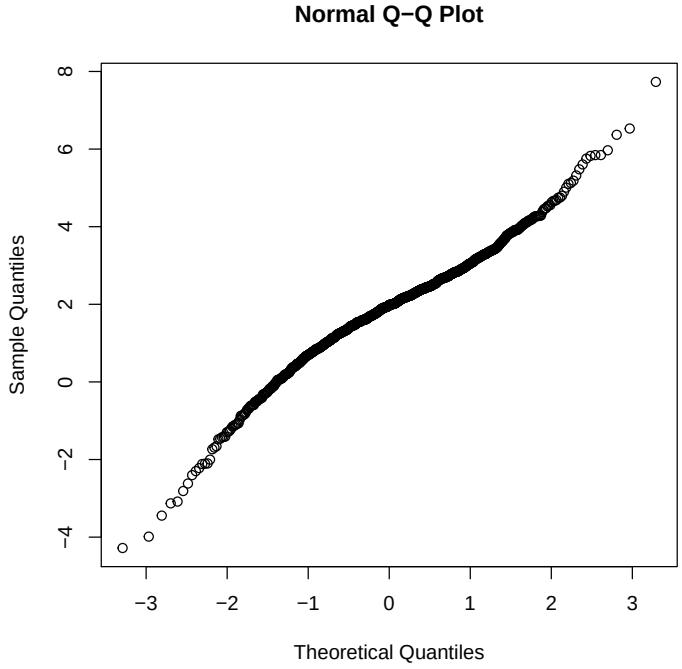
$\hat{\alpha}_c$ の分布は

```
> qqnorm(b01_10c$h_alphac)
```



$\hat{\beta_{1c}}$ の分布は

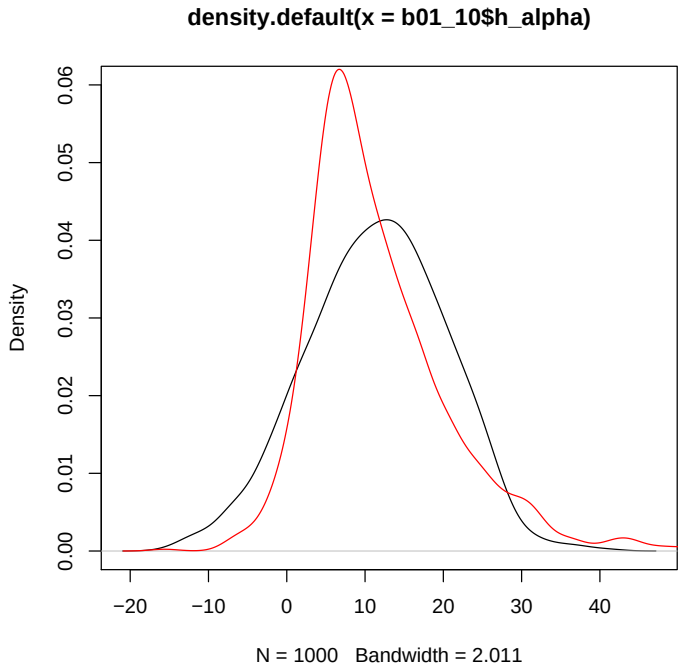
```
> qqnorm(b01_10c$h_beta1c)
```



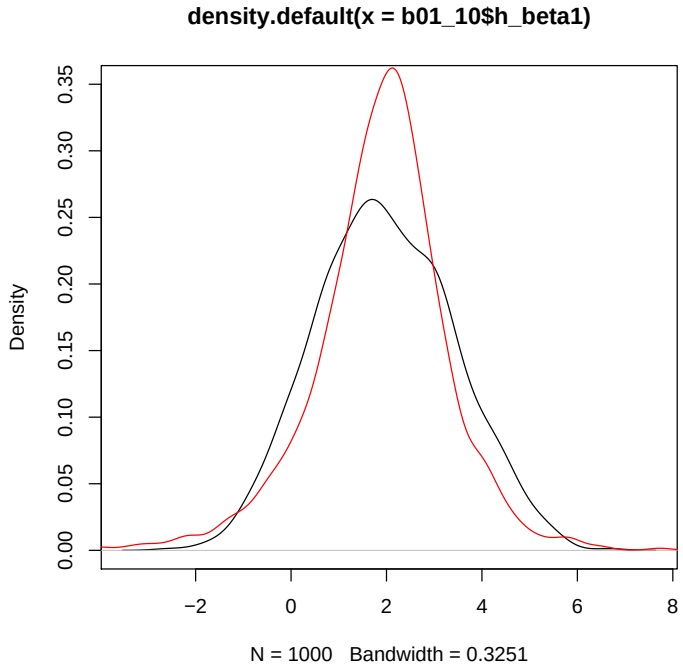
正規分布の見えたが、やや歪んでいる。

さきほどの推定値の分布と比べてみる

```
> plot(density(b01_10$h_alpha),ylim=c(0,0.06))
> lines(density(b01_10c$h_alphac),col=2)
```




```
> plot(density(b01_10$h_beta1),ylim=c(0,0.35))
> lines(density(b01_10c$h_beta1c),col=2)
```



しかし、これを１回の標本数を増やすと、かなり正規分布に近くなる。

これも同様に 100 個ずつサンプリングを 1000 回繰り返して、1000 パターンの推定係数 $\hat{\alpha}_c, \hat{\beta}_{1c}$ を得る。

```
> b01_100c<-matrix(NA,1000,2)
> nko<-10
> for(i in 1:1000){
+   s01r<-d01c[unlist(by(c(1:10000),x1,f01s)),]
+   o01r<-lm(yc~x1,s01r)
+   b01_100c[i,]<-coef(o01r)
```

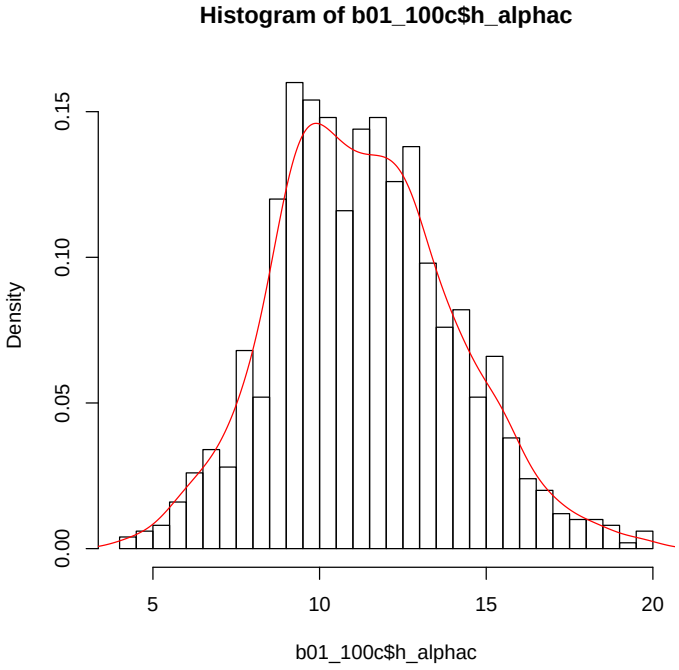
```
+ }
> b01_100c<-data.frame(b01_100c)
> names(b01_100c)<-c("h_alphac","h_beta1c")
> summary(b01_100c)
```

```
    h_alphac         h_beta1c
 Min.   : 4.028   Min.   :0.5984
 1st Qu.: 9.448   1st Qu.:1.6550
 Median :11.138   Median :1.9274
 Mean   :11.339   Mean   :1.9342
 3rd Qu.:13.027   3rd Qu.:2.2243
 Max.   :19.834   Max.   :3.3949
```

分布を確認する

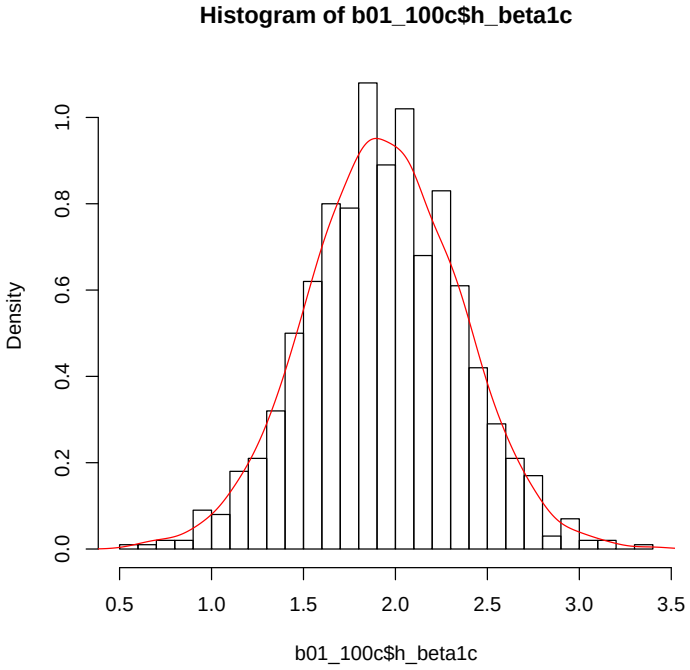
$\hat{\alpha}_c$ の分布は

```
> hist(b01_100c$h_alphac,prob=T,br="fd")
> lines(density(b01_100c$h_alphac),col=2)
```



$\hat{\beta}_{1c}$ の分布は

```
> hist(b01_100c$h_beta1c,prob=T,br="fd")
> lines(density(b01_100c$h_beta1c),col=2)
```

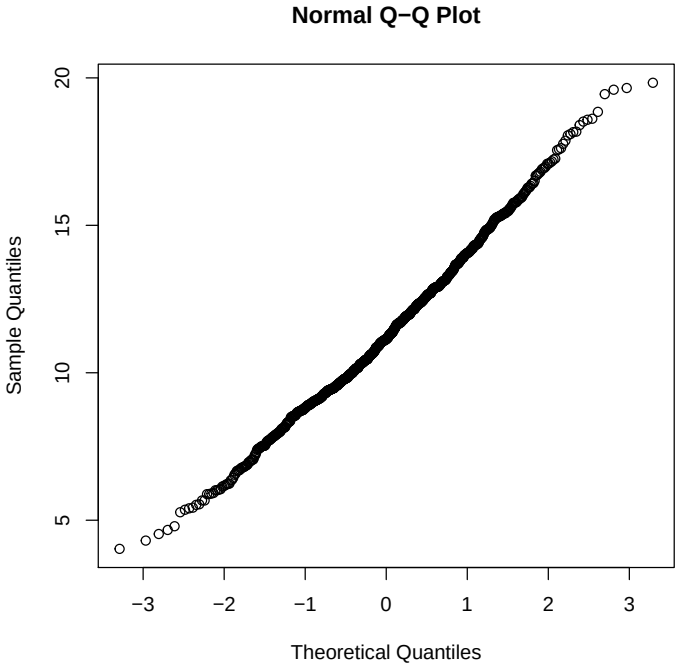


これも正規分布に近いようにも見える。

qqnorm 関数で確認してみる。

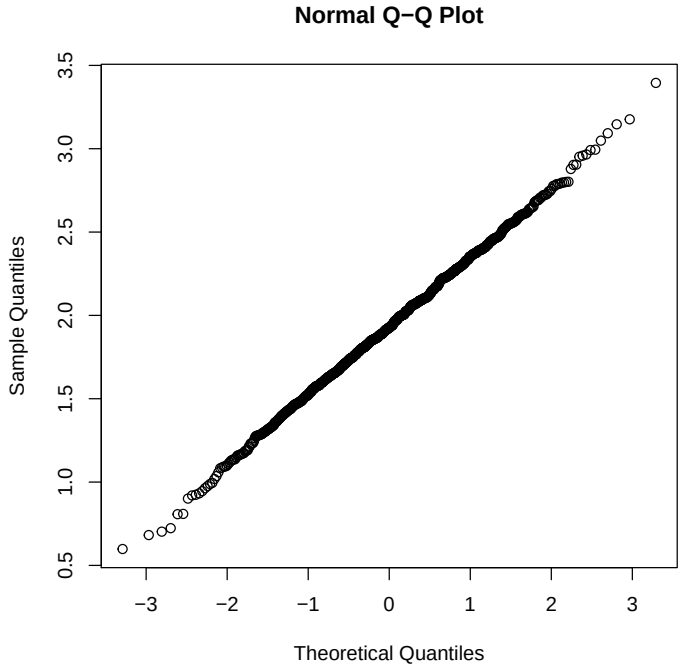
$\hat{\alpha}_c$ の分布は

```
> qqnorm(b01_100c$h_alphac)
```



$\hat{\beta}_{1c}$ の分布は

```
> qqnorm(b01_100c$h_beta1c)
```



OLS 推定量は、

$$\hat{beta} = \beta + (x'x)^{-1}x'\varepsilon$$

であり、結局は ε の線形結合である。

したがって、標本数が大きければ（x と ε との相関がない、つまり独立である限り）中心極限定理が効いてくる。

6.2.4 OLS 推定量の標準誤差の推定

ここまでは、母集団からの標本抽出を 1000 回繰り返して、推定係数の標準偏差を調べた。
しかし、１回の推定でも、標準誤差として、理論的にある程度推定できることも説明した。
この理論的な OLS 推定量の標準誤差はどうやって求めているのか？
やってみよう。

OLS 推定量の標準誤差

OLS 推定量の標準誤差は、以下の分散の平方根として定義される。

$$E[(\hat{\alpha} - \alpha)^2]$$

$$E[(\hat{\beta}_1 - \beta_1)^2]$$

これを「標準偏差」と呼ばないのは、標準偏差なら以下の分散の平方根として定義されるからだ。

$$E[(\hat{\alpha} - E[\hat{\alpha}])^2]$$

$$E[(\hat{\beta}_1 - E[\hat{\beta}_1])^2]$$

もっとも、OLS 推定量は不偏だから $E[\hat{\beta}] = \beta$ で、同じことだが。
OLS 推定量の分散・共分散行例は以下の行列の期待値。

$$\begin{bmatrix} (\hat{\alpha} - \alpha)^2 & (\hat{\alpha} - \alpha)(\hat{\beta}_1 - \beta_1) \\ (\hat{\alpha} - \alpha)(\hat{\beta}_1 - \beta_1) & (\hat{\beta}_1 - \beta_1)^2 \end{bmatrix}$$

$$= (\hat{\beta} - \beta)(\hat{\beta} - \beta)'$$

$$= (x'x)^{-1}x'\varepsilon((x'x)^{-1}x'\varepsilon)'$$

$$= (x'x)^{-1}x'\varepsilon\varepsilon'x(x'x)^{-1}$$

ちなみに、ここの計算は行列 A, B について、次が成り立つことを使っている。
すなわち、転置行列の転置行列はもとの行列

$$(A')' = A$$

行列の積の転置行列は、それぞれ転置して掛ける順番を入れ替えた行列の積

$$(AB)' = B'A'$$

逆行列の転置は、転置行列の逆行列と同じ

$$(A^{-1})' = (A')^{-1}$$

誤差項は互いに独立で等分散であるという２つの仮定が成り立てば

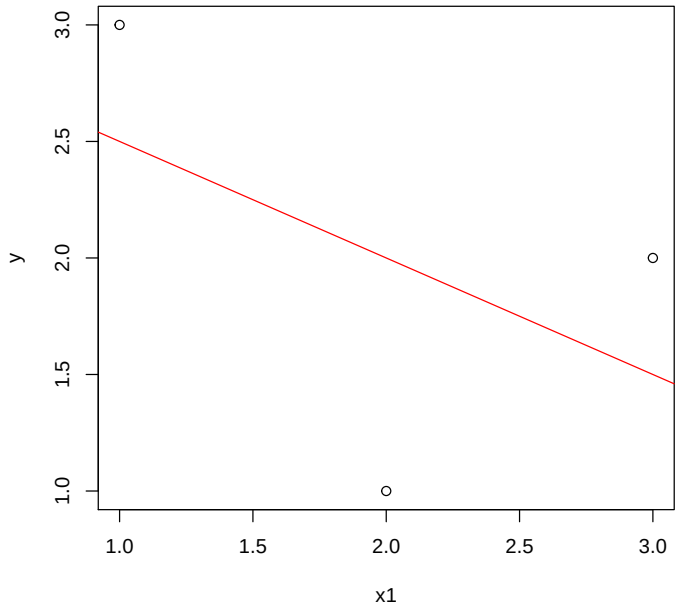
$$E[\varepsilon\varepsilon'] = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} = \sigma^2 I^{(n)}$$

x は確率変数ではないので・・・

$$\begin{aligned} E[(\hat{\beta} - \beta)(\hat{\beta} - \beta)'] &= (x'x)^{-1}x'E[\varepsilon\varepsilon']x(x'x)^{-1} \\ &= \sigma^2(x'x)^{-1}x'x(x'x)^{-1} \\ &= \sigma^2(x'x)^{-1} \end{aligned}$$

これが OLS 推定量 $\hat{\beta}$ の分散・共分散行列
前回のはじめにやった３つだけのデータによる OLS 回帰で確認してみる。

```
> x1<-c(1,2,3)
> y<-c(3,1,2)
> o01<-lm(y~x1)
> plot(y~x1)
> abline(o01,col=2)
```



推定結果の要約を表示すると

```
> summary(o01)
```

```
Call:
lm(formula = y ~ x1)

Residuals:
   1    2    3
 0.5 -1.0  0.5

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.000      1.871   1.604    0.355
x1            -0.500      0.866  -0.577    0.667

Residual standard error: 1.225 on 1 degrees of freedom
Multiple R-squared:   0.25,        Adjusted R-squared:   -0.5
F-statistic: 0.3333 on 1 and 1 DF,  p-value: 0.6667
```

Std. Error' の箇所が OLS 推定量の標準誤差。

OLS 推定量の分散の理論値 $\sigma^2(x'x)^{-1}$ を計算する。

ただし σ^2 はわからないので、残差の標準誤差を使う。

残差は $e_i = y_i - \hat{\alpha} - \hat{\beta}_1 x_{1i}$。

残差の標準誤差は、この場合データ数が 3 つしかないので

$$\frac{\sum_{i=1}^{3} e_i^2}{3-2}$$

分母の $3-2$ というのは自由度で、データが 3 つで、推定すべきパラメータが 2 つだから。

R の場合、残差の標準誤差は、OLS の結果の要約に Residual standard error として報告され、この場合、`summary(o01)$sigma` で取り出せる。

```
> (x<-cbind(rep(1,3),x1))
```

```
     x1
[1,] 1  1
```

```
[2,] 1  2
[3,] 1  3
```

```
> (v01<-solve(t(x)%*%x)*summary(o01)$sigma^2)
```

```
           x1
   3.5 -1.50
x1 -1.5  0.75
```

このうち、$\hat{\alpha}$ の標準誤差は、

```
> sqrt(v01[1,1])
```

```
[1] 1.870829
```

β_1 の標準誤差は、

```
> sqrt(v01[2,2])
```

```
[1] 0.8660254
```

いずれも summary(o01) で報告されている推定係数の Std. Error の値そのもの。

6.3 推定と検定

OLS 推定量の区間推定と仮説検定を行う。

6.3.1 区間推定

OLS 推定量の分布

前回まで、OLS 推定量は不偏であることを示した。

$$E[\hat{\beta}] = \beta$$

また、OLS 推定量の分散・共分散行列は以下であることを示した。

$$V[\hat{\beta}] = \sigma^2(x'x)^{-1}$$

しかも、標本規模がある程度大きいならば、$\hat{\beta}$ は正規分布に近かった。

つまり、

$$v[\hat{\beta}] = \begin{bmatrix} \sigma_\alpha^2 \\ \sigma_{\beta_1}^2 \end{bmatrix}$$

と表記すると、

$$\hat{\alpha} \sim N(\alpha, \sigma_\alpha^2)$$

$$\hat{\beta}_1 \sim N(\beta_1, \sigma_{\beta_1}^2)$$

ということ。

ここまでわかれば、β の推定と検定は簡単だ。

区間推定

つまり、OLS 推定量 $\hat{\alpha}$ は、95 %の確率で、

$$\alpha - 1.960\sigma_\alpha \sim \alpha + 1.960\sigma_\alpha$$

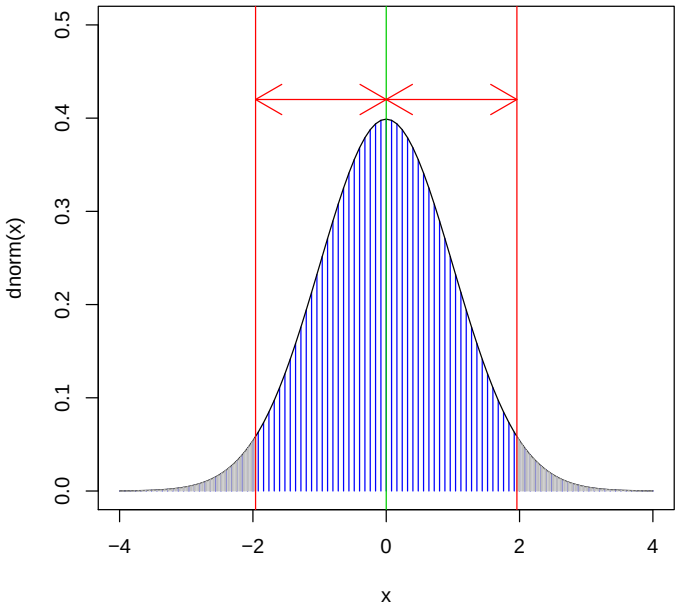
の間にあるはずだ。

また、OLS 推定量 $\hat{\beta}_1$ は、95 %の確率で、

$$\beta_1 - 1.960\sigma_{\beta_1} \sim \beta_1 + 1.960\sigma_{\beta_1}$$

の間にあるはずだ。

```
> curve(dnorm,-4,4,type="h",col=4,ylim=c(0,0.5))
> curve(dnorm,add=T)
> curve(dnorm,add=T,xlim=c(-4,-1.960),type="h",col="gray")
> curve(dnorm,add=T,xlim=c(1.960,4),type="h",col="gray")
> abline(v=-1.960,col=2)
> abline(v=1.960,col=2)
> abline(v=0,col=3)
> arrows(0,0.42,-1.96,0.42,col=2,code=3)
> arrows(0,0.42,1.96,0.42,col=2,code=3)
```



ということは、ひとつの OLS 推定量 $\hat{\alpha}, \hat{\beta}_1$ が得られた場合・・・
本当の α は 95 %の確率で、

$$\hat{\alpha} - 1.960\sigma_\alpha \sim \hat{\alpha} + 1.960\sigma_\alpha$$

の間にあるはずだ。

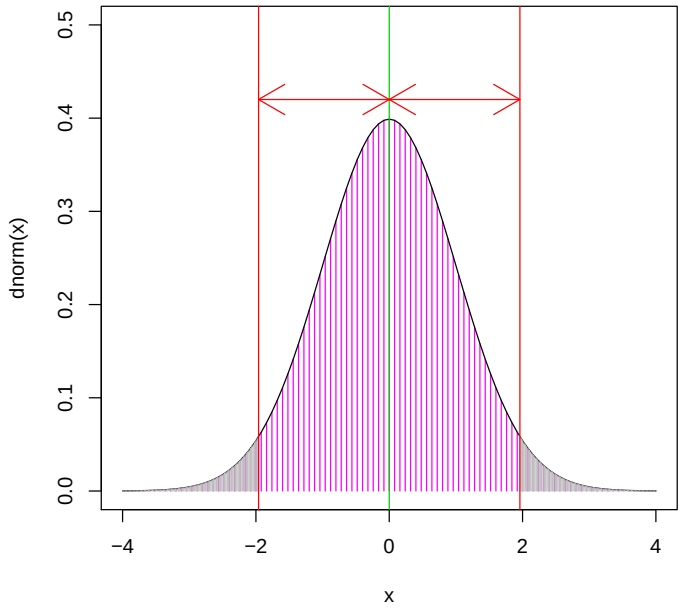
また、本当の β_1 は、95 %の確率で、

$$\hat{\beta}_1 - 1.960\sigma_{\beta_1} \sim \hat{\beta}_1 + 1.960\sigma_{\beta_1}$$

の間にあるはずだ。

図示すると以下となる。

```
> curve(dnorm,-4,4,type="h",col=6,ylim=c(0,0.5))
> curve(dnorm,add=T)
> curve(dnorm,add=T,xlim=c(-4,-1.960),type="h",col="gray")
> curve(dnorm,add=T,xlim=c(1.960,4),type="h",col="gray")
> abline(v=-1.960,col=2)
> abline(v=1.960,col=2)
> abline(v=0,col=3)
> arrows(0,0.42,-1.96,0.42,col=2,code=3)
> arrows(0,0.42,1.96,0.42,col=2,code=3)
```

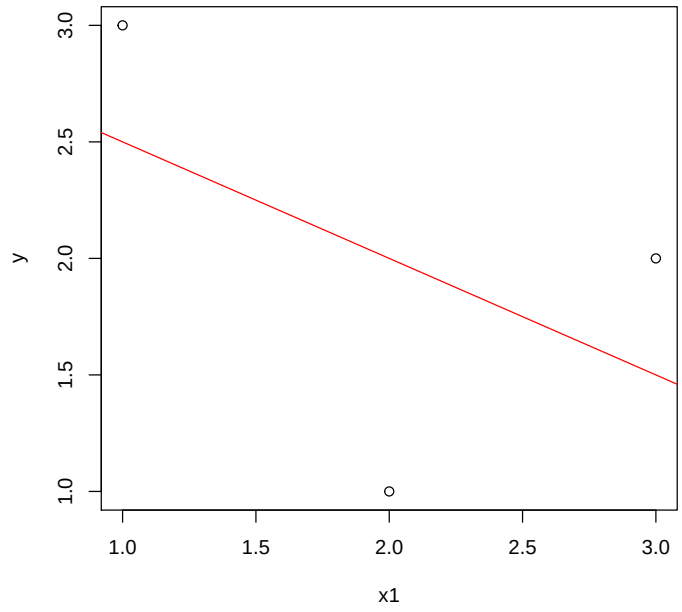


これが OLS による区間推定。

例題

3 つだけのデータによる OLS 回帰で、実際に区間推定をやってみる。

```
> x1<-c(1,2,3)
> y<-c(3,1,2)
> o01<-lm(y~x1)
> plot(y~x1)
> abline(o01,col=2)
```



OLS 推定量は

```
> (c01<-summary(o01)$coef)
```

```
            Estimate Std. Error    t value  Pr(>|t|)
(Intercept)      3.0  1.8708287  1.6035675 0.3549784
x1              -0.5  0.8660254 -0.5773503 0.6666667
```

α の区間推定は、OLS 推定量が 1 行 1 列目、標準誤差が 1 行 2 列目にあるので、

```
> qnorm(c(0.025,0.975),c01[1,1],c01[1,2])
```

```
[1] -0.6667569  6.6667569
```

β_1 の区間推定は、OLS 推定量が 2 行 1 列目、標準誤差が 2 行 2 列目にあるので、

```
> qnorm(c(0.025,0.975),c01[2,1],c01[2,2])
```

```
[1] -2.197379  1.197379
```

ただし、この区間推定は、標準偏差に、OLS 推定量の標準誤差 σ_β ではなく、その推定値 s_β を使っている！
標本数が小さいときは、この後示すように、自由度 $n-p$ の t 分布で推定すべきだ。

区間推定の注意点

注意しなければならないのは、この区間推定が、OLS 推定量が正規分布をするという仮定に依存していることだ。

OLS 推定量が正規分布しているということは・・・
誤差項の分布が正規分布である場合か、
あるいは、そうでなくても、標本数が多くて、中心極限定理で OLS 推定量が正規分布に近似できる場合か、のどちらかである。
どちらにもあてはまらない場合は、なんらかの方法で OLS 推定量の分布を確認しなければならない。(「ブートストラップ法」と呼ばれる方法などがあります。)

6.3.2　仮説検定

帰無仮説と対立仮説

OLS 推定量の仮説検定というのは、何を検定しているかというと、各 OLS 推定量の絶対値が有意に 0 より大きいかどうかを確かめるというもの。

つまり、被説明変数 y の説明変数として意味があるかどうか、である。
この場合、帰無仮説は、$\beta = 0$ 対立仮説は、$|\beta| > 0$ となる。
(対立仮説の方は、マイナスの可能性もあるので、絶対値がついている)

OLS 推定量の t 値

つまり、α についての仮説検定はあまり意味がないので、以降は β_1 だけで話をすすめる。
このとき、OLS 推定量 $\hat{\beta}_1$ は、平均 β_1、標準偏差 σ_{β_1} の正規分布でばらつくはずだ。
ということは、

$$\frac{\hat{\beta}_1 - \beta_1}{\sigma_{\beta_1}} \sim N(0,1)$$

は標準正規分布でばらつくことになる。
ひとまず、帰無仮説が正しいと考えて、$\beta_1 = 0$ であるとしよう。

ということは、

$$\frac{\hat{\beta}_1}{\sigma_{\beta_1}} \sim N(0,1)$$

ということになる。

しかし、σ_{β_1} を計算するのに必要な誤差項の標準偏差 σ はわからないので、これを残差の標準誤差 s に代える。

前回、説明したとおり、データセットの標本規模が n 個で、推定すべきパラメータの数が p 個のとき、残差の標準誤差 s は、

$$s^2 = \frac{\sum_{i=1}^{n} e_i^2}{n-p}$$

だった。

s は、 lm 関数の summary で、 Residual standard error として報告される。

その s を使った OLS 推定量の標準誤差が、推定係数の Std. Error のところに報告される。これを s_{β_1} と表記しよう。

その場合、s_{β_1} も確率変数であるから、

$$\frac{\hat{\beta}_1}{s_{\beta_1}} \sim t(n-p)$$

で、自由度 $n-p$ の t 分布に従う。

もうちょっと細かく言うと、

$$\frac{\sum_{i=1}^{n} e_i^2}{\sigma^2}$$

が自由度 $n-p$ のカイ二乗分布に従う、というところからこのことが言えるのだが、このあたりは、ちょっとややこしくなってくるので、一旦飛ばす。

気になる人は、次回の残差の性質のところを参照して欲しい。

とにかく、OLS 推定量 $\hat{\beta}_1$ をその標準誤差の推定値 s_{β_1} で割った値が t 値である。

OLS 推定量の仮説検定

この t 値は、$\beta_1 = 0$ であるとき、自由度 $n-p$ の t 分布に従うから、t 値の絶対値の大きさで、$\beta_1 = 0$ が正しい確率を確認することができる。

上記で使った OLS 推定量の例では、以下の値が得られていた。

```
> c01
```

```
            Estimate Std. Error    t value  Pr(>|t|)
(Intercept)      3.0  1.8708287  1.6035675 0.3549784
x1              -0.5  0.8660254 -0.5773503 0.6666667
```

2行目の Estimate を Std. Error の値で割った値が t value である。

```
> c01[2,1]/c01[2,2]
```

```
[1] -0.5773503
```

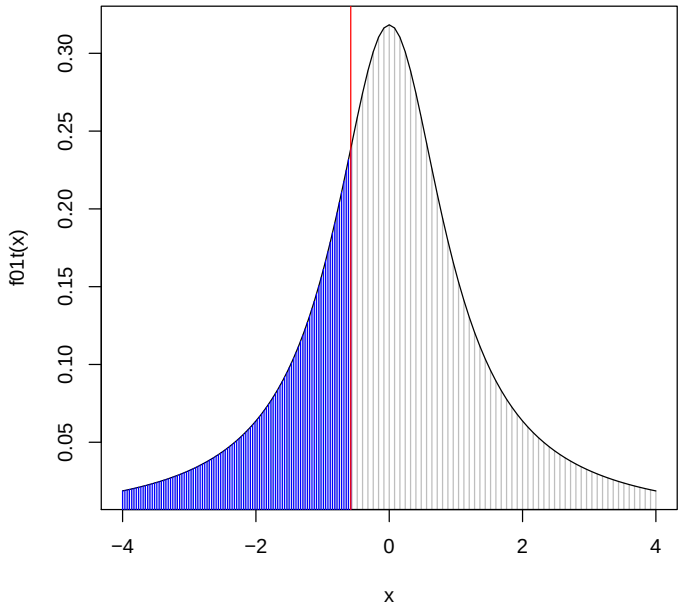
この場合の t 値はマイナスで、自由度 $3-2=1$ の t 分布に従うから、t 値がこの値以上に 0 から離れる確率は、以下で求めることができる。

```
> pt(c01[2,3],1)
```

```
[1] 0.3333333
```

図示すると、以下のようになる。

```
> df01<-1
> f01t<-function(x) dt(x,df01)
> curve(f01t,-4,4,type="h",col="gray")
> curve(f01t,add=T)
> curve(f01t,-4,c01[2,3],type="h",col=4,add=T)
> abline(v=c01[2,3],col=2)
```



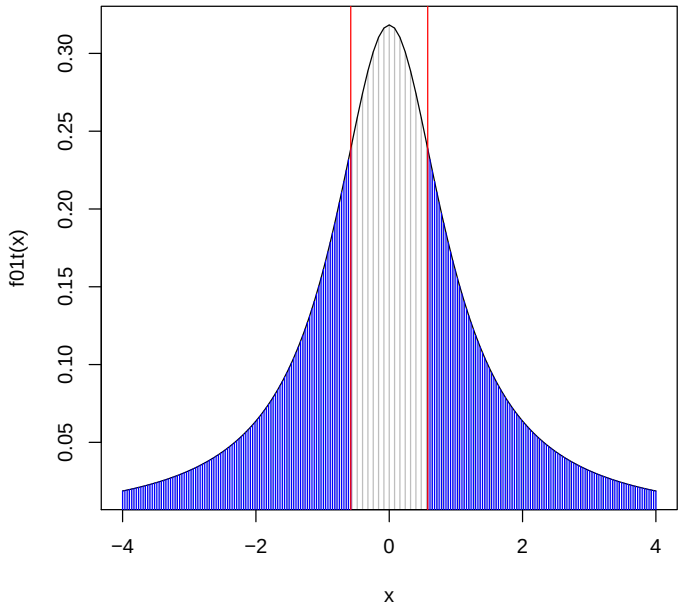
これが p 値であり、帰無仮説 $\beta_1 = 0$ が正しいとき、OLS 回帰推定量が実際に得られた推定値以上に 0 から離れる確率となる。

片側検定と両側検定

しかし、この値は、 summary 関数で報告される p 値 $\Pr(>|t|)$ の値の半分しかないではないか！？

実は、上で出した p 値は、「片側検定」と言って、 lm 関数の summary で報告されるのは「両側検定」。図示すると以下のような値で、片側検定の２倍となる。

```
> curve(f01t,-4,4,type="h",col="gray")
> curve(f01t,add=T)
> curve(f01t,-4,c01[2,3],type="h",col=4,add=T)
> curve(f01t,-c01[2,3],4,type="h",col=4,add=T)
> abline(v=c01[2,3],col=2)
> abline(v=-c01[2,3],col=2)
```



なんで反対側まで？ と思うかもしれないが、$\beta = 0$ からそのくらい大きくブレると考え、その確率を求めたと考えることも可能だ。

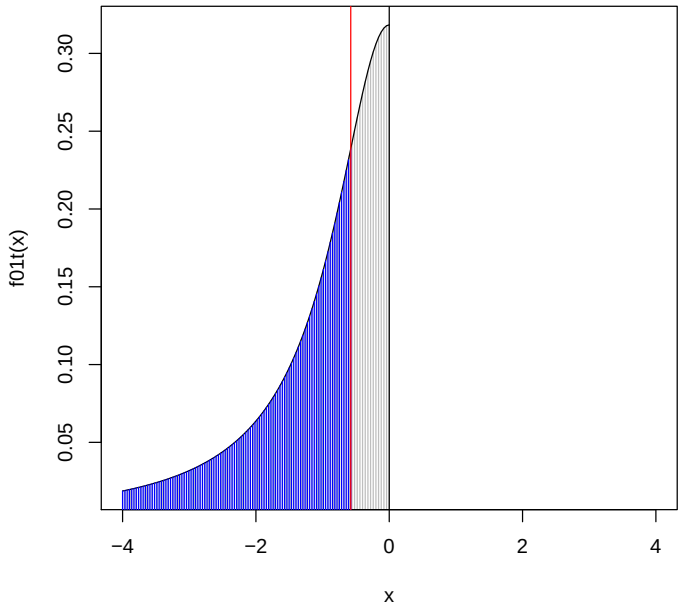
しかし、回帰分析においては、回帰係数の符号というのは、予め決まっていることも多い。

例えば、価格が上がると需要が小さくなるなど・・・。

その場合、推定係数が 0 より大きくなる可能性はない。

つまり、とりうる t 値の値は、以下のような半分の可能性となる。

```
> curve(f01t,-4,0,type="h",col="gray",xlim=c(-4,4))
> curve(f01t,add=T, from=-4,to=0)
> curve(f01t,-4,c01[2,3],type="h",col=4,add=T)
> abline(v=c01[2,3],col=2)
> abline(v=0)
```



そうなると、p 値とすべきはは、プラス側も可能性ある場合の 2 倍の値だ。

結局、これは、両側検定として考えたのと同じことになる。

この例題だと、p 値は

```
> pt(c01[2,3],1)*2
```

```
[1] 0.6666667
```

で計算しなければならない。

これが 0.6667 であるということは、$\beta_1 = 0$ だったとしても、この程度の t 値以上が出る確率は 66.7 %もあるということで、$\beta_1 = 0$ であることを否定できない、ということになる。

すなわち、帰無仮説は棄却されない、ということだ。（まあ、標本の規模が３つしかないからな・・・）

逆に、t 値が大きくて、p 値が小さい場合、$\beta_1 = 0$ と考えるのは無理がある、ということで、帰無仮説は棄却される。

その場合、推定した OLS 推定量は、説明変数として意味があるということになる。

p 値がどの程度小さいなら、それを「小さい」と判断するかだが、慣例的に、10 %、5 %、1 %未満かどうかで判断する。

当然だが、片側検定より、両側検定の方が p 値は大きくなるので、帰無仮説は棄却されにくくなる。

仮説検定の注意点

これも、OLS 推定量が正規分布をするという仮定に基づいている。

その条件は、区間推定の時に説明したのと同じ、誤差項が正規分布するか、大標本かのどちらかである。

そうでない場合は、他の方法が必要となる。

また、仮説検定で帰無仮説が棄却できなかった説明変数をどうするか、という問題がある。

よく、p 値の大きい説明変数を推定から省く・・・などということをする人を見ることがあるが、それはまずい。

p 値が大きいというのは、OLS 推定量が 0 である可能性が 否定できない と言っているに過ぎない。

たとえば、中古車プリウスの例だと、珍しい車の色がどのくらい中古車価格を左右しているかは、台数が少

ないので、はっきりとは言えない、もしかしたら 0 だったかもしれない。

しかし、だからといって、この説明変数を無視すると、他の車の色に交じることで、他の車の色の効果の推定を歪める可能性がある。

t 値はあくまで、OLS 推定量の推定の確からしさを判断するもので、変数選択は、t 値ではなく、AIC などに基づいて行った方がましな結果となる。

6.4　残差の性質

残差の和はゼロ、説明変数と残差の積の和もゼロ、被説明変数の理論値と説明変数との積の和もゼロ、ということを示す。

6.4.1　残差とは

残差は誤差項ではない。

OLS 推定の残差（residual）は、

$$e = y - x\hat{\beta}$$

で定義され、あくまで、OLS 推定量次第で決まる。

誤差項（error）は

$$\varepsilon = y - x\beta$$

であり、x 以外の要因で決まる、y の変動である。

しかし、$\hat{\beta}$ は、β の不偏推定量だから、残差 e も誤差項 ε の推定値ではある。

回帰分析は、特定の説明変数 x_k の影響を推定したり、y そのものの変動を予測したりするとともに、残差 e に注目した分析を行うことがある。特定のプリウスの価格が、回帰式で期待される価格よりも割安か割高かを評価する、など。

その場合、OLS 残差 e の性質について知っておくことは大事だ。

6.4.2　残差の和はゼロ

OLS 回帰に定数項が含まれる場合、残差の和は常に 0 である。

$$\sum_{i=1}^{n} e_i = 0$$

とういうことは、当然平均もゼロだ。

誤差項の場合、期待値がゼロだった（$E[\varepsilon_i] = 0$）。

この違いは、しっかり理解するように。

証明しよう。

OLS 推定量の求め方を思い出せば、そのままだ。

OLS 回帰は、残差平方和を以下で定義して

$$S^2 = \sum_i (y_i - a - x_{1i}b_1)^2$$

これを最小にする a, b を求めるというものだった。

そのためには、S^2 を a, b で偏微分して、それぞれが 0 となる点を探せばよい。

このうち a による偏微分は、

$$\frac{\partial S^2}{\partial a} = -\sum_i (y_i - a - b_1 x_{1i}) = 0$$

となるから、これを満たす a, b を $\hat{\alpha}, \hat{\beta}_1$ と表すと、

$$\sum_i (y_i - \hat{\alpha} - \hat{\beta}_1 x_{1i}) = 0$$

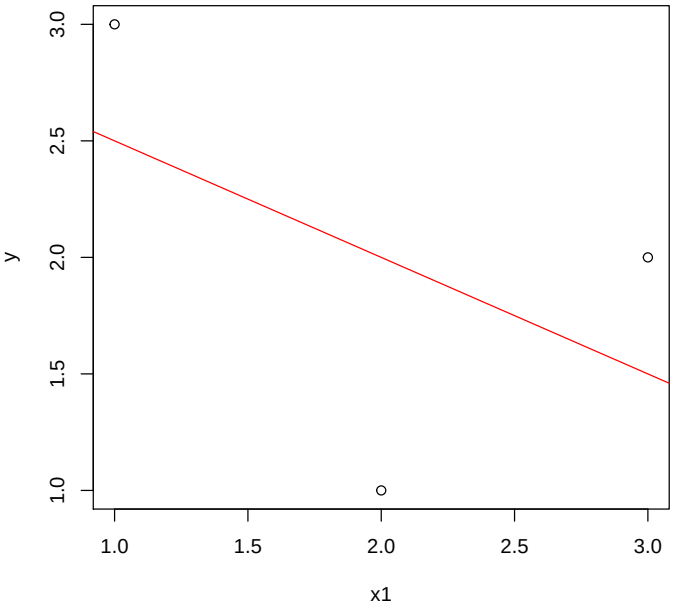
であり、

$$\sum_i e_i = 0$$

と、そのまんまである。

前回やった 3 つだけのデータによる OLS 回帰で確認してみる。

```
> x1<-c(1,2,3)
> y<-c(3,1,2)
> o01<-lm(y~x1)
> plot(y~x1)
> abline(o01,col=2)
```



残差は

```
> resid(o01)
```

```
  1   2   3
0.5 -1.0 0.5
```

その合計は確かに 0 である。

```
> sum(resid(o01))
```

```
[1] 1.110223e-16
```

PC の浮動点小数演算のせいで、やたらと小さい値が表示されるが、これはゼロである。

6.4.3 残差と説明変数との積の和はゼロ

$$\sum_{i}^{n} x_{1i} e_i = 0$$

行列表示すると、

$$x'e = 0$$

「説明変数と誤差項は独立」」という仮定（$E[x\varepsilon] = 0$）に似ているが、こっちは期待値がゼロで、あくまで仮

定だった。残差と説明変数との積の和は常にゼロだ。・・・というよりも、そうなるように $\hat{\alpha}$ と $\hat{\beta}_1$ を決めているからなのだが。

証明もそのまんま。

OLS 回帰は、さきほどの続きで、残差平方和 S^2 を b で偏微分したものが 0 となるように $\hat{\alpha}, \hat{\beta}_1$ を決める。つまり、

$$\frac{\partial S^2}{\partial b_1} = -\sum_i x_i(y_i - a - b_1 x_{1i}) = 0$$

となるように $\hat{\alpha}, \hat{beta}_1$ を決めるということで、

$$\sum_i x_{1i}(y_i - \hat{\alpha} - \hat{\beta}_1 x_{1i}) = 0$$

となる。

これは、

$$\sum_i x_{1i} e_i = 0$$

で、そのまんまだ。

さきほどの標本規模が３つの例で確かめてみる。

```
> x1
```

```
[1] 1 2 3
```

```
> resid(o01)
```

```
  1   2   3
0.5 -1.0 0.5
```

```
> sum(x1*resid(o01))
```

```
[1] -1.110223e-16
```

6.4.4　行列表示での証明

この証明は、行列表記で以下のようにすることもできる。

OLS 推定量 $\hat{\beta}$ は、以下で定義された（すぐに導出できますか？）

$$\hat{\beta} = (x'x)^{-1}x'y$$

従って、

$$\begin{aligned} x'e &= x'(y - x\hat{\beta}) \\ &= x'(y - x(x'x)^{-1}x'y) \\ &= (x' - x'x(x'x)^{-1}x')y \\ &= (x' - x')y = 0 \end{aligned}$$

ところで、$x'e = 0$ というのは、

$$x' = \begin{bmatrix} 1 & \cdots & 1 \\ x_{11} & \cdots & x_{1n} \end{bmatrix}$$

だから、

$$x'e = \begin{bmatrix} \sum_i e_i \\ \sum_i x_{1i} e_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

と、残差の和がゼロと、説明変数と残差との積の和がゼロの両方が証明できる。

例題でやってみる。

```
> (x<-cbind(rep(1,3),x1))
```

```
     x1
[1,] 1  1
[2,] 1  2
[3,] 1  3
```

```
> (e<-matrix(resid(o01)))
```

```
     [,1]
[1,]  0.5
[2,] -1.0
[3,]  0.5
```

```
> t(x)%*%e
```

```
           [,1]
   1.110223e-16
x1 0.000000e+00
```

6.4.5 被説明変数の理論値と残差との積の和はゼロ

被説明変数の理論値 $\hat{y}_i = \hat{\alpha} + \hat{\beta}_1 x_{1i}$ に対して、

$$\sum_i \hat{y}_i e_i = 0$$

ということだが、これは、そのまんま

$$\sum_i (\hat{\alpha} + \hat{\beta}_1 x_{1i}) e_i$$

ということであり、

$$\hat{\alpha} \sum_i e_i + \hat{\beta}_1 \sum_i x_{1i} e_i$$

だから、$\sum_i e_i = 0, \sum_i x_{1i} e_i = 0$ から、

$$\sum_i \hat{y}_i e_i = 0$$

$\sum_i y_i e_i$ ではないので、要注意！

6.4.6 残差平方和の分布

前回、OLS 推定量 $\hat{\beta}_1$ をその標準誤差の推定値 s_{β_1} で割った値は、標本規模が n 個、推定すべきパラメータ数 p 個の場合で、$\beta_1 = 0$ という帰無仮説が下で、$n - p$ の t 分布をすると言った。

$$\frac{\hat{\beta}_1}{s_{\beta_1}} \sim t(n-p)$$

この証明は、残差平方和 $s^2 = e'e$ の分布がわからないとできないのだが、ちょっとややこしいので、ここは興味ある人だけ読んでもらえればよい。

6.4.7　残差平方和の分散

ひとまず、次のことを示す。

σ^2 の不偏推定量は $e'e/(n-p)$、つまり、

$$\frac{E(e'e)}{n-p} = \sigma^2$$

ということ。

残差 e と誤差項 ε との関係は，

$$\begin{aligned} e &= y - x\hat{\beta} \\ &= y - x(x'x)^{-1}x'y \\ &= (I - x(x'x)^{-1}x')y \\ &= (I - x(x'x)^{-1}x')(x\beta + \varepsilon) \\ &= (I - x(x'x)^{-1}x')\varepsilon \end{aligned}$$

したがって，残差の期待値は 0

$$E(e) = (I - x(x'x)^{-1}x')E(\varepsilon) = 0$$

残差の分散・共分散行列は，

$$\begin{aligned} Var[e] &= (I - x(x'x)^{-1}x')Var(\varepsilon)(I - x(x'x)^{-1}x')' \\ &= (I - x(x'x)^{-1}x')(\sigma^2 I)(I - x(x'x)^{-1}x')' \\ &= \sigma^2(I - x(x'x)^{-1}x')(I - x(x'x)^{-1}x') \\ &= \sigma^2(I - x(x'x)^{-1}x') \end{aligned}$$

残差平方和の期待値は，$Var[e]$ の対角成分の和

$$Var[e] = \begin{pmatrix} E[e_1^2] & \cdots & E[e_1 e_n] \\ \vdots & \ddots & \vdots \\ E[e_n e_1] & \cdots & E[e_n^2] \end{pmatrix}$$

だから，

$$E[e_1^2 + \cdots + e_n^2] = \sum_{i=1}^{n} E(e_i^2) = \sigma^2 tr(I - x(x'x)^{-1}x') = \sigma^2(n-p)$$

ただし，$tr()$ は対角成分の和で，

$$tr(A+B) = tr(A) + tr(B),\ tr(AB) = tr(BA)$$

であるから，

$$\begin{aligned} tr(I - x(x'x)^{-1}x') &= tr(I) - tr(x(x'x)^{-1}x') \\ &= tr(I) - tr(x'x(xx')^{-1}) \\ &= n - p \end{aligned}$$

したがって，

$$\sigma^2 = \frac{E[e_1^2 + \cdots + e_n^2]}{n-p} = \frac{E[e'e]}{n-p}$$

6.4.8　例題

$Var[e]$ を求める。

$y = 5 + 2x_1 + \varepsilon$（誤差項は $\sigma^2 = 1$ の正規分布）の回帰で、５個の標本をとり残差を求めるという作業を 1000 回繰り返して、1000 パターンの残差を得る。

```
> x1<-1:5
> e01<-list(NULL)
> for(i in 1:1000){
+   y<-5+2*x1+rnorm(5)
+   o04<-lm(y~x1)
+   e01[[i]]<-resid(o04)
+ }
```

そのうちの 5 個だけ表示すると

```
> head(e01)
```

```
[[1]]
         1          2          3          4          5
 0.6768298 -0.8593167 -0.5276790  0.9259888 -0.2158230

[[2]]
         1          2          3          4          5
 0.11767962 -0.02191959  0.22353498 -0.85202969  0.53273467

[[3]]
         1          2          3          4          5
 0.3760909  0.1132138 -1.6180518  1.3920987 -0.2633516

[[4]]
         1          2          3          4          5
 0.01902128  1.52829874 -1.62413990 -1.41270153  1.48952141

[[5]]
         1          2          3          4          5
-0.04251978 -0.07827642 -0.36655780  1.13802399 -0.65066999

[[6]]
         1          2          3          4          5
 0.08339541 -0.11502781 -0.40038203  0.81226584 -0.38025141
```

といった感じ。

ee' を求める

```
> e01e01<-lapply(e01,function(x) matrix(x,,1)%*%matrix(x,1,))
```

1000 とおりの ee' ができるが、これも最初の 5 個だけ表示すると、

```
> head(e01e01)
```

```
[[1]]
           [,1]       [,2]       [,3]       [,4]        [,5]
[1,]  0.4580986 -0.5816111 -0.3571489  0.6267368 -0.14607541
[2,] -0.5816111  0.7384251  0.4534434 -0.7957176  0.18546026
[3,] -0.3571489  0.4534434  0.2784451 -0.4886248  0.11388524
[4,]  0.6267368 -0.7957176 -0.4886248  0.8574553 -0.19984965
[5,] -0.1460754  0.1854603  0.1138852 -0.1998496  0.04657955

[[2]]
             [,1]          [,2]         [,3]        [,4]         [,5]
[1,]  0.013848494 -0.0025794885  0.026305512 -0.10026653  0.06269202
[2,] -0.002579489  0.0004804682 -0.004899794  0.01867614 -0.01167732
[3,]  0.026305512 -0.0048997940  0.049967886 -0.19045844  0.11908483
[4,] -0.100266532  0.0186761373 -0.190458437  0.72595459 -0.45390576
[5,]  0.062692015 -0.0116773230  0.119084833 -0.45390576  0.28380623

[[3]]
           [,1]        [,2]       [,3]       [,4]        [,5]
[1,]  0.14144434  0.04257867 -0.6085345  0.5235556 -0.09904413
```

```
[2,]  0.04257867  0.01281736 -0.1831858  0.1576048 -0.02981503
[3,] -0.60853448 -0.18318578  2.6180915 -2.2524878  0.42611652
[4,]  0.52355560  0.15760478 -2.2524878  1.9379388 -0.36661142
[5,] -0.09904413 -0.02981503  0.4261165 -0.3666114  0.06935406

[[4]]
             [,1]       [,2]        [,3]        [,4]       [,5]
[1,]  0.000361809  0.0290702 -0.03089322 -0.02687139  0.0283326
[2,]  0.029070196  2.3356970 -2.48217095 -2.15902996  2.2764337
[3,] -0.030893217 -2.4821710  2.63783041  2.29442492 -2.4191912
[4,] -0.026871389 -2.1590300  2.29442492  1.99572561 -2.1042492
[5,]  0.028332602  2.2764337 -2.41919115 -2.10424917  2.2186740

[[5]]
             [,1]         [,2]        [,3]        [,4]        [,5]
[1,]  0.001807932  0.003328296  0.01558596 -0.04838853  0.02766634
[2,]  0.003328296  0.006127199  0.02869283 -0.08908045  0.05093212
[3,]  0.015585957  0.028692834  0.13436462 -0.41715157  0.23850816
[4,] -0.048388529 -0.089080449 -0.41715157  1.29509860 -0.74047805
[5,]  0.027666344  0.050932120  0.23850816 -0.74047805  0.42337143

[[6]]
             [,1]         [,2]        [,3]        [,4]        [,5]
[1,]  0.006954795 -0.009592792 -0.03339002  0.06773924 -0.03171122
[2,] -0.009592792  0.013231397  0.04605507 -0.09343316  0.04373949
[3,] -0.033390024  0.046055068  0.16030577 -0.32521664  0.15224583
[4,]  0.067739244 -0.093433161 -0.32521664  0.65977580 -0.30886523
[5,] -0.031711223  0.043739487  0.15224583 -0.30886523  0.14459114
```

$Var[e] = E[e'e]$ を求める

```
> e01e01a<-array(unlist(e01e01),dim=c(5,5,1000))
> (e01e01m<-apply(e01e01a,1:2,mean))
```

```
             [,1]        [,2]       [,3]         [,4]        [,5]
[1,]  0.370796061 -0.37905126 -0.1695216 -0.006987322  0.18476409
[2,] -0.379051263  0.68944853 -0.2274867 -0.097167098  0.01425655
[3,] -0.169521568 -0.22748671  0.7895414 -0.218536396 -0.17399673
[4,] -0.006987322 -0.09716710 -0.2185364  0.756523371 -0.43383256
[5,]  0.184764091  0.01425655 -0.1739967 -0.433832556  0.40880864
```

$E[e'e] = E[e_1^2 + \cdots + e_n^2]$ は

```
> sum(diag(e01e01m))
```

```
[1] 3.015118
```

これは $\sigma^2(n-p) = 3$ と一致するはず（標本が５個しかないのでやや粗いが）。

6.4.9　残差平方和の分布

残差平方和が自由度 $n-p$ のカイ二乗分布をすることを示す。

残差平方和は自由度 $n-p$ のカイ二乗分布に従う

$$\frac{e'e}{\sigma^2} \sim \chi^2(n-p)$$

このことを示すために、ひとまず次の定理を使う。

確率変数 u が標準正規分布 $\mathrm{N}(0,1)$ に従うとき，対称なべき等行列 A に対して，

$$u'Au \sim \chi^2(tr(A))$$

べき等行列：$A^2 = A$ となるような行列

$M = (I - x(x'x)^{-1}x)$ はべき等行列

$$(I - x(x'x)^{-1}x')(I - x(x'x)^{-1}x') = (I - x(x'x)^{-1}x')$$

3×3 の対称なべき等行列 dum23 と 3 個の標準正規乱数 dum24 でつくった積 t(dum24) dum23 dum24 は自由度 tr(dum23) のカイ二乗 分布に従うことを示す.

適当な 3×3 のべき等行列をつくる

```
> dum23<-c(.5,0,-.5, 0,1,0, -.5,0,.5)
> dum23<-matrix(dum23,3,3)
> dum23
```

```
     [,1] [,2] [,3]
[1,]  0.5    0 -0.5
[2,]  0.0    1  0.0
[3,] -0.5    0  0.5
```

```
> dum23%*%dum23
```

```
     [,1] [,2] [,3]
[1,]  0.5    0 -0.5
[2,]  0.0    1  0.0
[3,] -0.5    0  0.5
```

このとき tr(dum23) は

```
> sum(diag(dum23))
```

```
[1] 2
```

3×1000 個の標準正規乱数

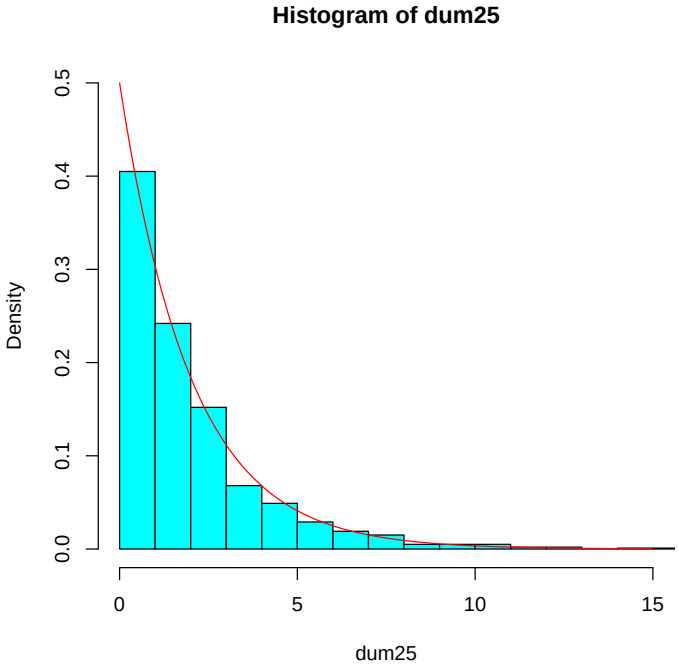
```
> dum24<-matrix(rnorm(3*1000),3,1000)
```

t(dum24) dum23 dum24 を 1000 個計算

```
> dum25<-c(length=25)
> for(i in c(1:1000)) dum25[i]<-t(dum24[,i])%*%dum23%*%dum24[,i]
```

これは自由度 tr(dum23) のカイ二乗分布

```
> hist(dum25,br="Scott",xlim=c(0,15),ylim=c(0,.5),prob=T,col=5)
> f01c<-function(x) dchisq(x,2)
> curve(f01c,col=2,add=T)
```



正規分布のこの性質を使って，$e'e/\sigma^2$ が自由度 $n-p$ のカイ二乗分布に従うことを示す．
$e = (I - x(x'x)^{-1}x')\varepsilon = M\varepsilon$ だったので，

$$e'e = (M\varepsilon)'(M\varepsilon) = \varepsilon' M M \varepsilon = \varepsilon' M \varepsilon$$

ここで，$\varepsilon \sim N(0, \sigma^2)$ (i.i.d.) を仮定すると，M は対称なべき等行列なので，

$$\frac{\varepsilon'}{\sigma} M \frac{\varepsilon}{\sigma} = \frac{1}{\sigma^2}\varepsilon M \varepsilon \sim \chi^2(tr(M)))$$

すなわち，

$$\frac{e'e}{\sigma^2} \sim \chi^2(n-p)$$

ちょっとむずかしいか？

6.5　多重共線性

6.5.1　2 変数の重回帰

OLS が最小分散であるための条件として、説明変数と誤差項とが（説明変数が確率変数であったとしても）独立であると仮定していた。

しかし、説明変数が 2 つ以上あった時に、それら説明変数どおしの関係には何も制約を課していなかった。

実際、回帰式

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

において、x_1, x_2 の間に相関があっても問題ない。

たとえば、

$$y = 10 + 2x_1 + 3x_2 + \varepsilon$$

だとしよう。

これで実際に OLS 回帰をやってみる。

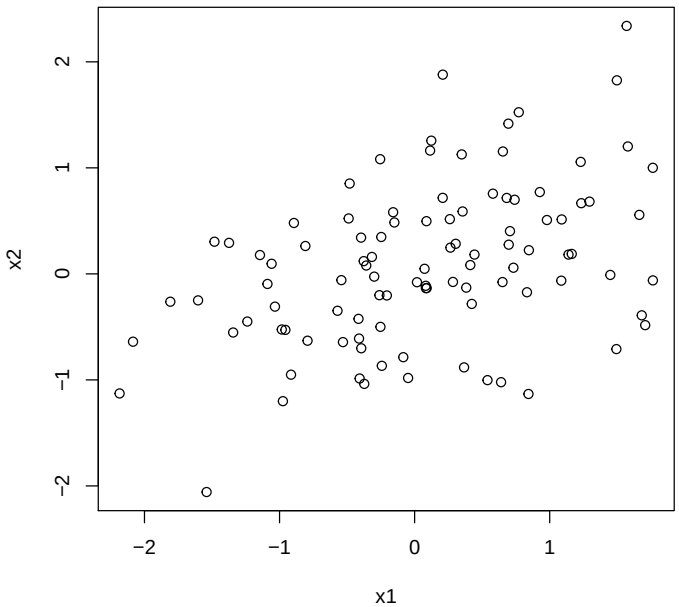
6.5.2　説明変数間にやや相関がある場合

データを発生させる。

```
> x1<-rnorm(100)
> x2<-0.3*x1+0.7*rnorm(100)
> y<-10+2*x1+3*x2+rnorm(100)
```

x_1, x_2 をプロットする。

```
> plot(x1,x2)
```



独立ではないが、それほどきつい相関があるわけではない。

OLS 回帰をやってみる。

```
> o041<-lm(y~x1+x2)
> summary(o041)
```

```
Call:
lm(formula = y ~ x1 + x2)

Residuals:
     Min       1Q   Median       3Q      Max
-2.84659 -0.67590  0.01533  0.69702  2.15671

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.9143     0.1028   96.43   <2e-16 ***
x1            2.0805     0.1226   16.97   <2e-16 ***
x2            2.9155     0.1509   19.32   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.02 on 97 degrees of freedom
Multiple R-squared:  0.9222,        Adjusted R-squared:  0.9206
F-statistic: 574.6 on 2 and 97 DF,  p-value: < 2.2e-16
```

まあまあ、うまいこと推定している。

ただ、１回きりではわからないので 1000 パターン発生させて不偏性を確認する。

```
> b041<-matrix(NA,nrow=1000,ncol=3)
> for(i in 1:1000){
+   y<-10+2*x1+3*x2+rnorm(100)
+   o041<-lm(y~x1+x2)
+   b041[i,]<-coef(o041)
+ }
> b041<-data.frame(b041)
> colnames(b041)<-c("a","b1","b2")
```

各係数の推定値の平均をとる。

```
> apply(b041,2,mean)
```

```
       a       b1       b2
9.998004 2.002857 2.991132
```

ほぼ不偏性が確認できた。

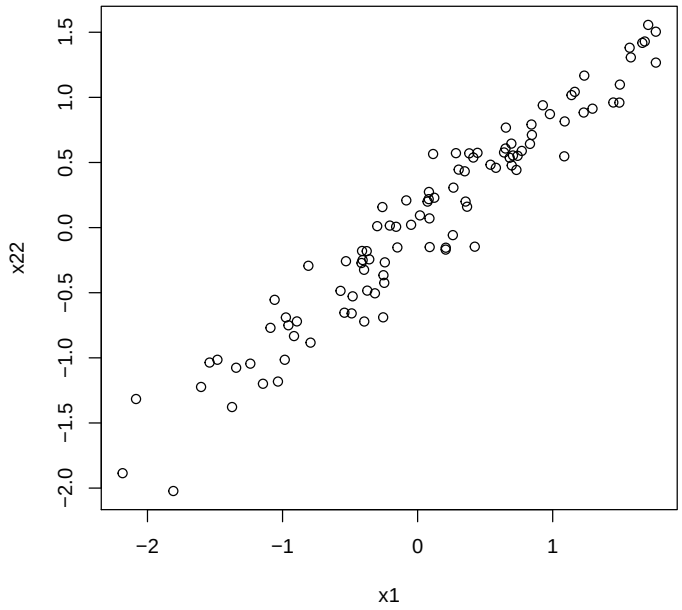
6.5.3　説明変数間にきつい相関がある場合

もう少し、x_1, x_2 の相関をきつくしてみる。

```
> x22<-0.8*x1+0.2*rnorm(100)
> y2<-10+2*x1+3*x22+rnorm(100)
```

x_1, x_2 をプロットする。

```
> plot(x1,x22)
```



完全に相関している。

OLS 回帰をやってみる。

```
> o042<-lm(y2~x1+x22)
> summary(o042)
```

```
Call:
lm(formula = y2 ~ x1 + x22)

Residuals:
    Min      1Q  Median      3Q     Max
-2.3647 -0.8262  0.0680  0.7793  3.2348

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.9556     0.1132  87.974  < 2e-16 ***
x1            2.1840     0.4704   4.643 1.08e-05 ***
x22           2.8131     0.5548   5.071 1.90e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.127 on 97 degrees of freedom
Multiple R-squared:  0.9338,        Adjusted R-squared:  0.9324
F-statistic: 683.8 on 2 and 97 DF,  p-value: < 2.2e-16
```

ちょっとずれたか？

これも１回きりではわからないので 1000 パターン発生させて不偏性を確認する。

```
> b042<-matrix(NA,nrow=1000,ncol=3)
> for(i in 1:1000){
+   y2<-10+2*x1+3*x22+rnorm(100)
+   o042<-lm(y2~x1+x22)
+   b042[i,]<-coef(o042)
+ }
> b042<-data.frame(b042)
> colnames(b042)<-c("a","b1","b2")
```

各係数の推定値の平均をとる。

```
> apply(b042,2,mean)
```

```
       a       b1       b2
9.999717 1.999628 2.997699
```

これだけ説明変数間に相関があっても不偏性が確認できる。

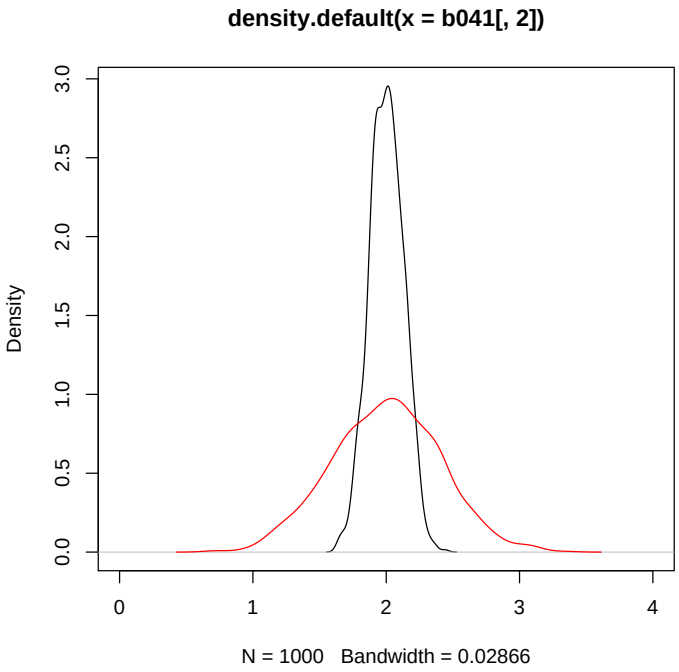
ただ、説明変数間の相関が強いほど、OLS 推定量の分散は大きくなる。

6.5.4　説明変数の相関の強さによる OLS 推定値のバラツキの違い

x_1 の係数で比較してみる。

黒い線が x_1 と x_2 の相関が弱い場合。赤い線は相関がきつい場合。

```
> plot(density(b041[,2]),xlim=c(0,4))
> lines(density(b042[,2]),col=2)
```

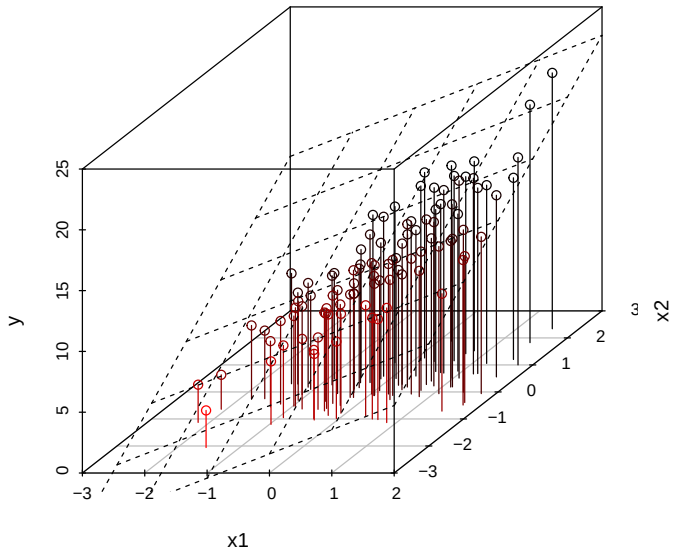


いずれも不偏性はあるが、相関のきつい方が OLS 推定値の分布が広くなっている

6.5.5　回帰平面は説明変数の平面上のバラツキが大きい方が安定する

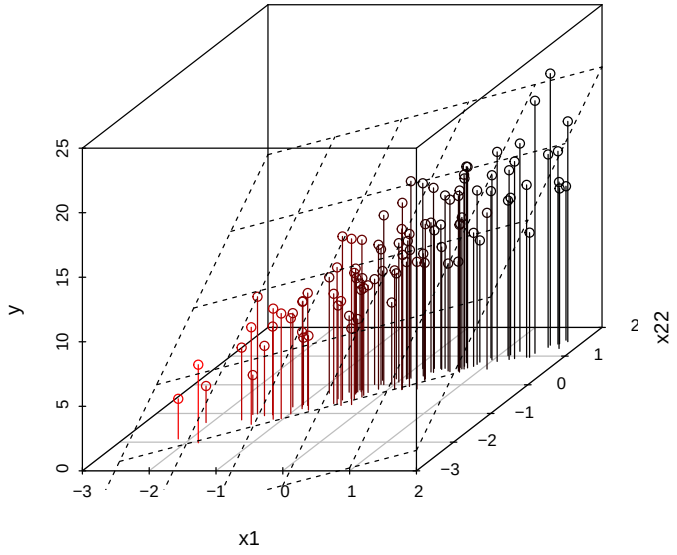
２変数による回帰の場合、平面で回帰しているので、回帰平面に対して、x_1 と x_2 が広くばらついているほうが、安定した推定ができる。

```
> library(scatterplot3d)
> g041<-scatterplot3d(y~x1+x2,type="h",highlight=T,pch=21)
> g041$plane3d(o041)
```



しかし、回帰平面に対して x_1 と x_2 があまりばらついていなかったら、回帰平面は不安定となる。

```
> g042<-scatterplot3d(y~x1+x22,type="h",highlight=T,pch=21)
> g042$plane3d(o042)
```



6.5.6　多重共線性

完全に x_1 と x_2 とが完全に線形関係にあったら、回帰ができない。

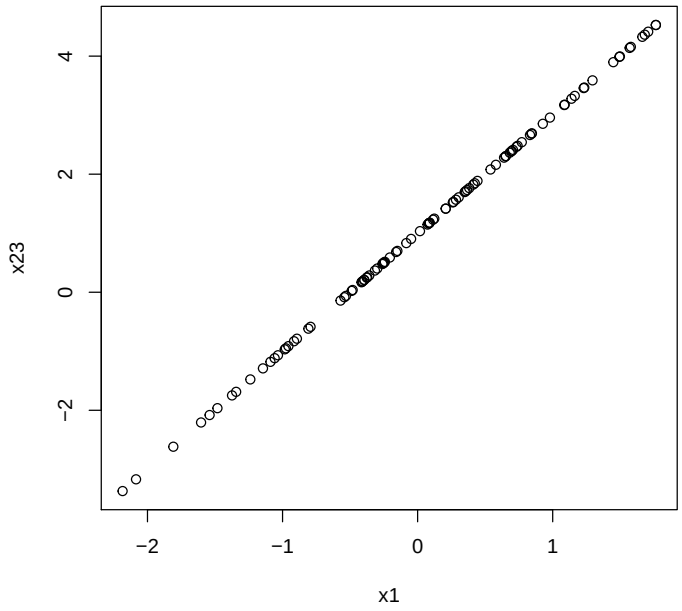
$x_2 = 1 + 2x_1$ だったとする。

```
> x23<-1+2*x1
```

```
> y3<-10+2*x1+3*x23+rnorm(100)
```

x_1, x_2 をプロットする。

```
> plot(x1,x23)
```



かなりきつい相関がある。

OLS 回帰をやってみる。

```
> o043<-lm(y3~x1+x23)
> summary(o043)
```

```
Call:
lm(formula = y3 ~ x1 + x23)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7867 -0.8003  0.0539  0.7690  2.4638

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  12.8035     0.1229  104.17   <2e-16 ***
x1            7.8980     0.1330   59.36   <2e-16 ***
x23               NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.224 on 98 degrees of freedom
Multiple R-squared:  0.9729,        Adjusted R-squared:  0.9727
F-statistic:  3524 on 1 and 98 DF,  p-value: < 2.2e-16
```

完全相関にある 1 変数が削除されて、$\beta_1 + 2\beta_2 = 8$ に近い値が、$\hat{\beta}_1$ として推定されている。

こういうのを多重共線性という。

多重共線性は、本当に x_1, x_2 が線形関係だったら、そもそも、その二つを別変数と考えることがおかしいので、どちらか片方だけで回帰しても問題ないはずだ。

ところが、実際には x_1, x_2 が線形関係ではないのに、たまたまデータをとったら、そうなった場合、片方の

変数が削除されて、あたかも x_1, x_2 の両方の効果を、もう片方の効果のように見誤る可能性がある。

たとえば、x_1 が男女、x_2 が関東か関西かの 0,1 のデータだとして、サンプリングの都合から、男は全て関東、女は全て関西となってしまったら、この場合多重共線性が生じてしまい、OLS 推定しても x_1 か x_2 のどちらかが省かれて、例えば x_1 だけが説明変数として採用されることになる。しかし、だからといって推定された係数は、性別の違いによるものであるとは全く言えない。性別によるものかもしれないし、関東か関西かによるものなのかもしれないし、それともその両方なのかもしれない。しかし、２つの変数が完全に相関している限り、そのようなデータではわからない、ということ。ちょっとだけばらけていたら１つの変数が捨てられることはなくなるが、それでも推定は不安定となる。関西男、関東男、関西女、関東女のそれぞれ両方がたくさんいて、すなわち説明変数がばらけてはじめて、有用な情報が得られる、ということ。

第 7 章

最小二乗法の仮定を緩める

7.1 不均一分散

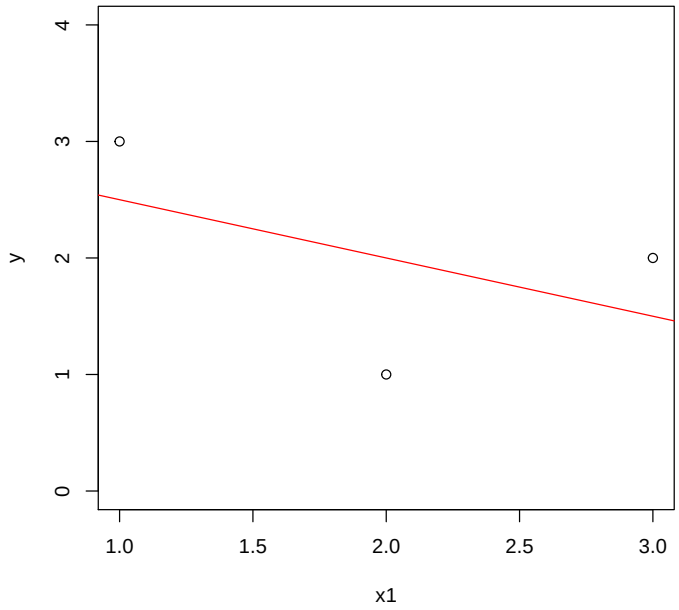
7.1.1 重み付き最小二乗推定

集計データの回帰

これまで何度かやった 3 つだけのデータによる OLS 回帰

```
> x1<-c(1,2,3)
> y<-c(3,1,2)
> o01<-lm(y~x1)
> plot(y~x1,ylim=c(0,4))
> abline(o01,col=2)
> summary(o01)
```

```
Call:
lm(formula = y ~ x1)

Residuals:
   1    2    3
 0.5 -1.0  0.5

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.000      1.871   1.604    0.355
x1            -0.500      0.866  -0.577    0.667

Residual standard error: 1.225 on 1 degrees of freedom
Multiple R-squared:   0.25,        Adjusted R-squared:   -0.5
F-statistic: 0.3333 on 1 and 1 DF,  p-value: 0.6667
```



このデータ、じつは、次の集計データだったとしたらどうなる？

x_1	y	n
1	3	1
2	1	10
3	2	5

つまり、$x_1 = 1$ についてとられた y のデータは 1 個しかないが、$x_1 = 2$ についてとられた y の値は 10 個あり、公表されているのはその平均値ということ。

$x_1 = 3$ の場合、y の値は 5 個のデータの平均値。

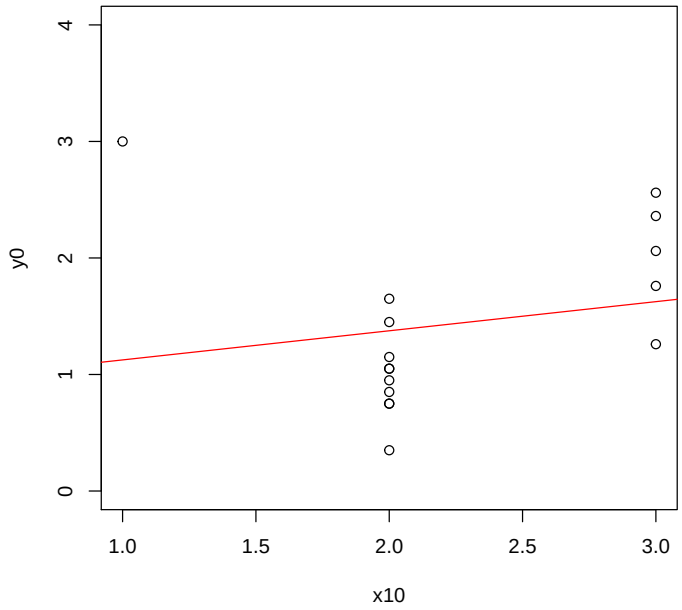
つまり、本当は、以下のようなデータだ。

x_1	y
1	3.00
2	0.85
2	0.75
2	1.05
2	1.05
2	0.35
2	1.65
2	0.75
2	1.15
2	0.95
2	1.45
3	2.36
3	2.56
3	1.26
3	1.76
3	2.06

これで OLS 回帰を行うと、さきほとどは違ったものとなる。

```
> x10<-c(1 ,2 ,2 ,2 ,2 ,2 ,2 ,2 ,2 ,2 ,3 ,3 ,3 ,3 ,3)
> y0<-c(3 ,0.85 ,0.75 ,1.05 ,1.05 ,0.35 ,1.65 ,0.75 ,
+       1.15 ,0.95 ,1.45 ,2.36 ,2.56 ,1.26 ,1.76 ,2.06)
> o010<-lm(y0~x10)
> plot(y0~x10,ylim=c(0,4))
> abline(o010,col=2)
> summary(o010)
```

```
Call:
lm(formula = y0 ~ x10)

Residuals:
   Min     1Q Median     3Q    Max
-1.025 -0.450 -0.275  0.315  1.875

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8750     0.7789   1.123    0.280
x10           0.2500     0.3360   0.744    0.469

Residual standard error: 0.7513 on 14 degrees of freedom
Multiple R-squared:  0.03804,        Adjusted R-squared:  -0.03067
F-statistic: 0.5537 on 1 and 14 DF,  p-value: 0.4691
```



具体的にはどんなケースか？

都道府県別や市町村別の全数調査の集計データなどは、人口が違うので、要注意でである。特に市町村ごとの人口はだいぶ違う場合が多い。

標本調査の場合は、集計単位の標本規模が問題となる。

たとえば、レポートの例題でもやったように、県庁所在地および政令指定都市の家計調査結果がある。しかし、じつはこれ、都市別に標本規模が違う！

家計調査結果を用いて回帰分析を行う場合、本来なら、この標本規模の違いを加味しなければならない。

社会統計というのは、ほとんどが集計データとして公表され、個々のデータが手に入ることはあまりない。

集計データを利用する場合は、集計単位ごとの標本規模を気にする必要がある。

重み付き最小二乗法

それでは、どうしたらよいか？

さきほどやったように、集計する前の、もともとのデータを見せてもらえればよいのだが、そう都合のよいことはないので、集計データで我慢するしかない。

仕方がないので、以下のように考えてはどうか。

x_1	y	n
1	3	1
2	1	10
3	2	5

というのを、

x_1	y
1	3
2	1
2	1
2	1
2	1
2	1
2	1
2	1
2	1
2	1
2	1
3	2
3	2
3	2
3	2
3	2

と考える。最小化する残差の二乗和を

$$S_w^2 = (y_1 - a - b_1 x_{11})^2 + (y_2 - a - b_1 x_{12})^2 + ... + (y_2 - a - b_1 x_{12})^2 + (y_3 - a - b_1 x_{13})^2 + ... + (y_3 - a - b_1 x_{13})^2$$

としてもよいが、$n_1 = 1, n_2 = 10, n_3 = 5$ と書き換えて、

$$S_w^2 = n_1(y - a - b_1 x_{11})^2 + n_2(y_2 - a - b_1 x_{12})^2 + n_3(y_3 - a - b_1 x_{13})^2$$

と表すこともできる。

これを最小化する a, b_1 を探すために、それぞれで偏微分する。

$$\frac{\partial S_w^2}{\partial a} = -\sum_i n_i(y_1 - a - b_1 x_{1i})$$

$$\frac{\partial S_w^2}{\partial b_1} = -\sum_i n_i x_{1i}(y_1 - a - b_1 x_{1i})$$

この２つをゼロにする a, b_1 が、S_w^2 を最小化するので、それを $\hat{\alpha}, \hat{\beta}_1$ と表すと、以下の関係式が成り立つ。

$$\begin{bmatrix} n_1 & n_2 & n_3 \\ n_1 x_{11} & n_2 x_{12} & n_3 x_{13} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} n_1 & n_2 & n_3 \\ n_1 x_{11} & n_2 x_{12} & n_3 x_{13} \end{bmatrix} \begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ 1 & x_{13} \end{bmatrix} \begin{bmatrix} \hat{\alpha} \\ \hat{\beta}_1 \end{bmatrix}$$

さらにこれは、以下のように変形できる。

$$\begin{bmatrix} 1 & 1 & 1 \\ x_{11} & x_{12} & x_{13} \end{bmatrix} \begin{bmatrix} n_1 & 0 & 0 \\ 0 & n_2 & 0 \\ 0 & 0 & n_3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_{11} & x_{12} & x_{13} \end{bmatrix} \begin{bmatrix} n_1 & 0 & 0 \\ 0 & n_2 & 0 \\ 0 & 0 & n_3 \end{bmatrix} \begin{bmatrix} 1 & x_{11} \\ 1 & x_{12} \\ 1 & x_{13} \end{bmatrix} \begin{bmatrix} \hat{\alpha} \\ \hat{\beta}_1 \end{bmatrix}$$

これは普通の OLS に対角成分が $\{n_1, n_2, n_3\}$ の正方行例が加わったもので、これを w と表して、以下のように書き換える。（その他の表記は、前回までの OLS の説明と同じ）

$$x'wy = x'wx\hat{\beta}$$

つまり、$\hat{\beta}$ は、

$$\hat{\beta} = (x'wx)^{-1}x'wy$$

ということである。

これは重み付き最小二乗法（Weighted least square: WLS）と呼ばれる。

確かめてみる

```
> (x<-cbind(rep(1,3),x1))
```

```
     x1
[1,] 1  1
[2,] 1  2
[3,] 1  3
```

```
> (w<-diag(c(1,10,5)))
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0   10    0
[3,]    0    0    5
```

```
> solve(t(x)%*%w%*%x)%*%t(x)%*%w%*%y
```

```
    [,1]
   0.875
x1 0.250
```

この推定値は、集計されていないもとのデータで行った OLS 推定値と同じだ！

```
> o010
```

```
Call:
lm(formula = y0 ~ x10)

Coefficients:
(Intercept)          x10
      0.875        0.250
```

重み付き最小二乗推定量の分散

$$y = x\hat{\beta} - \varepsilon$$

なので、

$$\hat{\beta} - \beta = (x'wx)^{-1}x'w\varepsilon$$

したがって、$\hat{\beta}$ の分散・共分散行列は、

$$\begin{aligned}
Var[\hat{\beta}] &= E[\left[(x'wx)^{-1}xw\varepsilon\left((x'wx)^{-1}xw\varepsilon\right)'\right] \\
&= E\left[(x'wx)^{-1}xw\varepsilon\varepsilon'\left((x'wx)^{-1}xw\right)'\right] \\
&= (x'wx)^{-1}xwVar[\varepsilon]\left((x'wx)^{-1}xw\right)' \\
&= \sigma^2(x'wx)^{-1}xw\left((x'wx)^{-1}xw\right)' \\
&= \sigma^2(x'wx)^{-1}xwwx'\left((x'wx)^{-1}\right)'
\end{aligned}$$

$$= \sigma^2 (x'wx)^{-1} xwwx' (x'wx)^{-1}$$

ということになる。

確かめてみる

ひとまず $(x'wx)^{-1}xwwx'(x'wx)^{-1}$ だけ計算してみると

```
> A<-solve(t(x)%*%w%*%x)%*%t(x)%*%w
> A%*%t(A)
```

```
                x1
    5.34375 -2.0625
x1 -2.06250  0.8750
```

σ がわからないので、3 つだけのデータでやった OLS 回帰の残差の標準誤差で代用する。

```
> summary(o01)$sigma
```

```
[1] 1.224745
```

つまり、$\hat{\beta}$ の分散・共分散行例は

```
> (B<-summary(o01)$sigma^2*A%*%t(A))
```

```
                 x1
    8.015625 -3.09375
x1 -3.093750  1.31250
```

$\hat{\beta}$ の標準誤差は、

```
> sqrt(diag(B))
```

```
              x1
2.831188 1.145644
```

集計していないもともとのデータで行った OLS 回帰の結果は？

```
> summary(o010)$coeff
```

```
            Estimate Std. Error   t value  Pr(>|t|)
(Intercept)    0.875  0.7789485 1.1233092 0.2802003
x10            0.250  0.3359847 0.7440815 0.4691328
```

これについては、WLS の方がだいぶ大きい。

利用できるデータが少ないので、それは仕方ないか・・・

R で簡単に WLS はできないのか？

できます！

lm 関数に引数 weights= を加えるだけです。

```
> o01w<-lm(y~x1,weights=c(1,10,5))
> summary(o01w)
```

```
Call:
lm(formula = y ~ x1, weights = c(1, 10, 5))
```

```
Weighted Residuals:
      1       2       3
 1.8750 -1.1859  0.8385

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.875      2.459   0.356    0.782
x1             0.250      1.061   0.236    0.853

Residual standard error: 2.372 on 1 degrees of freedom
Multiple R-squared:  0.05263,        Adjusted R-squared:  -0.8947
F-statistic: 0.05556 on 1 and 1 DF,  p-value: 0.8526
```

集計データの何が問題なのか？

OLS 推定量は、不偏で最小分散の線形推定量（BLUE）だったのではないのか？

それには、以下の４つの条件が必要だった。

1. $E[\varepsilon] = 0$
2. $E[\varepsilon_i \varepsilon_j] = 0$
3. $E[\varepsilon_1^2] = ... = E[\varepsilon_n^2] = \sigma^2$
4. $E[x_k \varepsilon] = 0$

集計データは、このうち３番目の均一分散の条件に反する。

分散 σ^2 の確率変数の n 個標本の平均値の分散は σ^2/n だった！

$x_{12} = 2$ に対して $y_2 = 1$ とあるのは、$n_2 = 10$ 個のデータの平均値であって、その個々のデータの誤差項の分散が σ^2 だったとしても、その平均値の誤差項の分散は $\sigma^2/10$ となるはずだ。

ただし、不均一分散は、最小分散の条件だった。

だから、この場合に OLS 推定したとしても、不偏性は保たれる。意外かもしれないが、不偏性の条件は、１番目と４番目の条件だけ。

改めて重み付き最小二乗推定について

一般に、次のような不均一分散を考える。

$$\Omega = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n^2 \end{bmatrix}$$

このとき、重み付き最小二乗推定量は、Ω の逆行列を w として与えたものとなる。

$$w = \Omega^{-1} = \begin{bmatrix} 1/\sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/\sigma_n^2 \end{bmatrix}$$

じつは、この Ω、非対角成分がゼロでない（２番目の条件を満たさない）次のようなものでも同じことが言える。

$$\Omega = \begin{bmatrix} \sigma_1^2 & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_n^2 \end{bmatrix}$$

この Ω を使った最小二乗推定を、一般化最小二乗法（generalized least square: GLS）というが、これ自身の推定が容易ではないので、使える場面は限られる。

不均一分散かどうかの確認

詳細は省くが、誤差項が不均一分散かどうかは、なかなか判断が難しい。

Goldfeld-Quant 検定や Breuseh-Pagan 検定というのがある。

```
> library(lmtest)
> gqtest(o010,order=x10)
```

```
        Goldfeld-Quandt test

data:  o010
GQ = 1.2351, df1 = 6, df2 = 6, p-value = 0.4021
```

```
> bptest(o010)
```

```
        studentized Breusch-Pagan test

data:  o010
BP = 4.0183, df = 1, p-value = 0.04501
```

不均一分散の影響を調整する White の方法というのもある。

しかし、集計データだとどの程度に不均一となるか、標本規模で予め推測することができる。

これは、不均一分散が、どんな具合に不均一かを予測できる、めずらしいケースなのだ。

7.2　誤差項間の相関

7.2.1　パネルデータ

n 人についての t 個のデータ

n 人の学生に、学期の半ば（$t = 0$）で中間テスト、学期終わり（$t = 1$）に期末テストを実施し、２つの成績 y_{i0}, y_{i1} を得た。

$$y_{i0} = \alpha + \beta_1 x_{1i0} + \gamma_i + \varepsilon_{i0}$$

$$y_{i1} = \alpha + \beta_1 x_{1i1} + \gamma_i + \varepsilon_{i1}$$

x_{1i0} は、学生 i の前半（$t = 0$）の出席率、x_{1i1} は後半（$t = 1$）の出席率。

成績というのは、個人差が大きいが、それを表すデータは得られない。

そこで、これを γ_i という変数で表してある。

こうした、n 人の t 期分のデータをパネルデータという。

パネルデータの難しいところ

上の回帰式で、実際には γ_i が観察できないからといって、これを誤差項に含めて、

$$y_{i0} = \alpha + \beta_1 x_{1i0} + \xi_{i0}$$

$$y_{i1} = \alpha + \beta_1 x_{1i1} + \xi_{i1}$$

としていまうと、ちょっとややこしい。

OLS 回帰が BLUE である４つの条件のうち、まず、

$$E[\xi_{it}] \neq 0$$

で、不偏ではなくなる。

もっともこれは、この個人能力が期待値ゼロの iid（独立同一分布）で分布すると仮定できれば不偏となる。

しかし、

$$E[\xi_{i0}\xi_{11}] \neq 0$$

はどうしようもないので、OLS 推定量は最小分散ではなくなる。

サンプルデータ

A 君（$i = 1$)、B 君（$i = 2$)、C 君（$i = 3$）３人の中間テストの成績 $y_0 = (y_{10}, y_{20}, y_{30})$ と前半出席率 $x_{10} = (x_{110}, x_{120}, x_{130})$ 、期末テストの成績 $y_1 = (y_{11}, y_{21}, y_{31})$ と後半出席率 $x_{11} = (x_{111}, x_{121}, x_{131})$ 、は以下だった。

i	y_{i0}	x_{1i0}	y_{i1}	x_{1i1}
1	93	0.7	81	0.4
2	46	0.6	66	0.9
3	56	0.9	22	0.3

A 君は優秀で、前半 7 割ぐらいしか出席しなかったけど 93 点だった。

しかし、後半気が緩んで主席率が４割に下がった。しかしそれでも 81 点とれた。

B君は、あんまり賢くないのに、前半６割しか出席せずに 46 点だった。
これではやばいと思い、後半９割出席して 66 点にアップした。
C君は、前半９割も出席したのに 56 点しかとれなかった。
すっかりやる気をなくしてしまい、後半は３割出席して 22 点しかとれなかった。

プールデータ OLS で計算してみる

データを入力

```
> x10<-c(0.7,0.6,0.9)
> x11<-c(0.4,0.9,0.3)
> y0<-c(93,46,56)
> y1<-c(81,66,22)
```

誰のデータかは無視して、６個のデータを使って OLS 回帰をしてみる。
使うデータは次の右側２列。

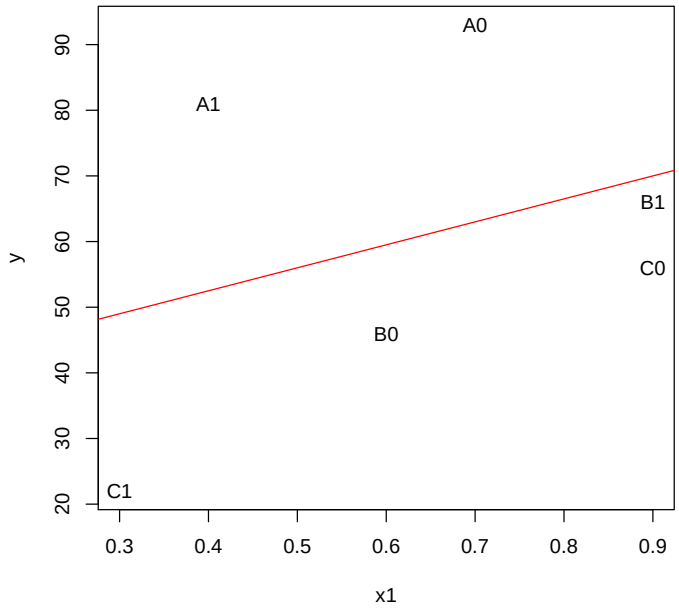
i	t	y_i	x_{1i}
1	0	93	0.7
2	0	46	0.6
3	0	56	0.9
1	1	81	0.4
2	1	66	0.9
3	1	22	0.3

これを使って OLS 回帰を行う。

```
> x1<-c(x10,x11)
> y<-c(y0,y1)
> o02p<-lm(y~x1)
> summary(o02p)
```

```
Call:
lm(formula = y ~ x1)

Residuals:
    1     2     3     4     5     6
 30.0 -13.5 -14.0  28.5  -4.0 -27.0

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    38.50      32.03   1.202    0.296
x1             35.00      47.56   0.736    0.503

Residual standard error: 26.62 on 4 degrees of freedom
Multiple R-squared:  0.1192,        Adjusted R-squared:  -0.101
F-statistic: 0.5415 on 1 and 4 DF,  p-value: 0.5026
```

```
> plot(y~x1,typ="n")
> text(x1,y,c("A0","B0","C0","A1","B1","C1"))
> abline(o02p,col=2)
```



階差によるモデル

2 つの式の辺々を引き算してみよう。

$$y_{i1} - y_{i1} = \beta_1(x_{1i1} - x_{1i0}) + \varepsilon_{i1} - \varepsilon_{i0}$$

α と γ_i が相殺されて消えている。

$$y_i = y_{i1} - y_{i0},\ x_{1i} = x_{1i1} - x_{1i0},\ \varepsilon_i = \varepsilon_{i1} - \varepsilon_{i0}$$

と書き換えると、

$$y_i = \beta_1 x_{1i} + \varepsilon_i$$

というシンプルな回帰式となる。

たとえば、$\varepsilon_{it} \sim N(0, \sigma^2)$ だとすると、$\varepsilon_i \sim N(0, 2\sigma^2)$。

定数項もなくなってしまった。

つまり、成績の絶対値は個人差（γ_i）があるが、それはさて置いて、β_1 どれだけ出席率が伸びれば成績がどれだけ上がるか（逆に、どれだけ出席率が下がれば成績がどれだけ下がるか）は、個人間で差がないということを仮定したモデル。

階差モデルを例題でやってみる

使うデータは、以下

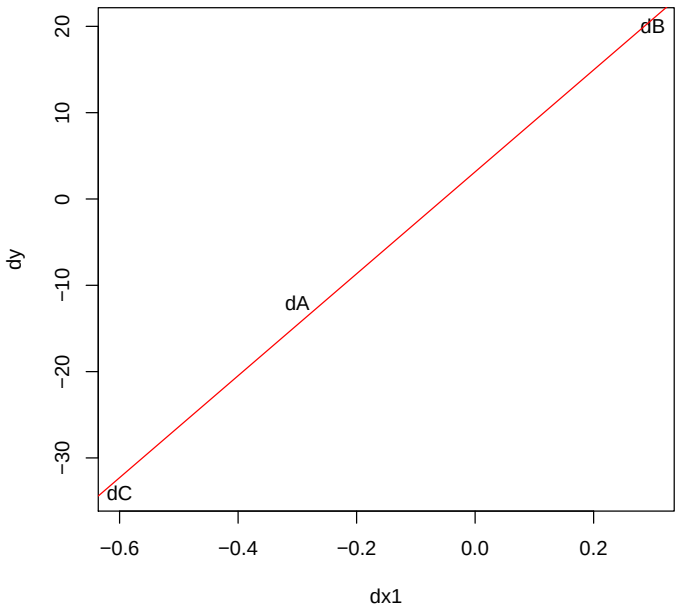
i	y_i	x_{1i}
1	-12	-0.3
2	20	0.3
3	-34	-0.6

これを使って回帰分析をやってみる

```
> dx1<-x11-x10
> dy<-y1-y0
> o02d<-lm(dy~dx1)
> summary(o02d)
```

```
Call:
lm(formula = dy ~ dx1)

Residuals:
      1       2       3
 2.5714 -0.8571 -1.7143

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.143      2.100   1.497   0.3749
dx1           59.048      4.949  11.932   0.0532 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.207 on 1 degrees of freedom
Multiple R-squared:  0.993,        Adjusted R-squared:  0.9861
F-statistic: 142.4 on 1 and 1 DF,  p-value: 0.05323
```

```
> plot(dy~dx1,typ="n")
> text(dx1,dy,c("dA","dB","dC"))
> abline(o02d,col=2)
```



結構うまいこと推定できているようだ。

有効推定量

ただし、階差をとったことで、個人間の能力の差という情報は抜け落ちてしまった。

このような推定量は「有効」ではない、と言われる。

有効な推定量を得るには、一般化最小二乗法（GLS）というのを使えばよいのだが、ここまでには踏み込まない。

（このあたり気になる人は、浅野晳『計量経済学』有斐閣,2000 年あたりを参照してください）

個人ダミーを入れてみる

たとえば、A君を基準に、B君を 1、それ以外を 0 とするダミー変数 D_B を新たに加える。

さらに、C君を 1、それ以外を 0 とするダミー変数 D_C を新たに加える。

	i	D_B	D_C
A君	1	0	0
B君	2	1	0
C君	3	0	1

すなわち、以下のように表すと、

$$\gamma_1 = \gamma_A$$

$$\gamma_2 = \gamma_A + \gamma_B$$

$$\gamma_3 = \gamma_A + \gamma_C$$

先の回帰式は以下のように書き換えられる。

$$y_{i0} = \alpha + \beta_1 x_{1i0} + \gamma_A + \gamma_B D_B + \gamma_C D_C + \varepsilon_{i0}$$

$$y_{i1} = \alpha + \beta_1 x_{1i1} + \gamma_A + \gamma_B D_B + \gamma_C D_C + \varepsilon_{i1}$$

つまり、A君なら、定数項 $\alpha + \gamma_A$ となる。

B君なら、A君の定数項よりも γ_B だけ高い、または低い。

C君なら、A君の定数項よりも γ_C だけ高い、または低い。

ダミー変数モデルを例題でやってみる

データセットは、次の右側４列

i	t	y_i	x_{1i}	D_B	D_C
1	0	93	0.7	0	0
2	0	46	0.6	1	0
3	0	56	0.9	0	1
1	1	81	0.4	0	0
2	1	66	0.9	1	0
3	1	22	0.3	0	1

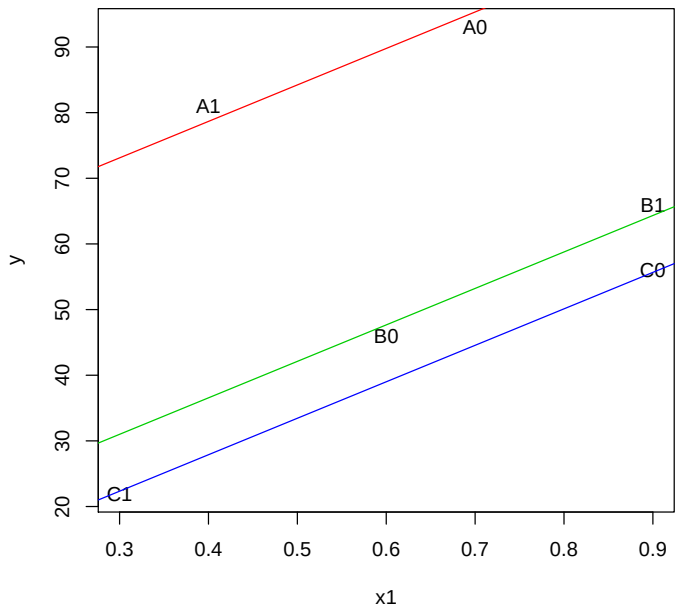
ただし、R では、このようにダミー変数をわざわざ作成する必要はなく、項目変数を説明変数とした場合、勝手にダミー変数として処理してくれる。

説明変数として、新たに x2 を代入したが、これはダミー変数ではなくA君か、B君か、C君かを示した A 、 B 、 C をデータとして与えている。

```
> x2<-c("A","B","C","A","B","C")
> y<-c(y0,y1)
> x1<-c(x10,x11)
> o02i<-lm(y~x1+x2)
> summary(o02i)
```

```
Call:
lm(formula = y ~ x1 + x2)

Residuals:
      1       2       3       4       5       6
-2.3333 -1.6667  0.3333  2.3333  1.6667 -0.3333

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   56.444      3.675   15.36  0.00421 **
x1            55.556      5.556   10.00  0.00985 **
x2B          -42.111      3.093  -13.61  0.00535 **
x2C          -50.778      2.900  -17.51  0.00325 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.887 on 2 degrees of freedom
Multiple R-squared:  0.9948,        Adjusted R-squared:  0.9871
F-statistic: 128.1 on 3 and 2 DF,  p-value: 0.007756
```

```
> plot(y~x1,typ="n")
> text(x1,y,c("A0","B0","C0","A1","B1","C1"))
> abline(coef(o02i)[1:2],col=2)
> abline(coef(o02i)[1]+coef(o02i)[3],coef(o02i)[2],col=3)
> abline(coef(o02i)[1]+coef(o02i)[4],coef(o02i)[2],col=4)
```



定数項の推定値が 56.4。これは、３人共通の定数項とＡ君の能力の和 $\alpha+\gamma_1=\alpha+\gamma_A$ の値の推定値。

Ｂ君の $\alpha+\gamma_2$ は、それよりも x2B の係数（42.1 点）だけ小さいと推定されている。

同じく C 君の $\alpha+\gamma_3$ は、A 君のよりも x2C の係数（50.8）だけ小さいと推定される。

出席率が 1（=100 %）の場合の点数が x1 の係数と推定されているが、これが 56.4 点で、階差モデルに近い。

このように、個人ダミーを入れたモデルは、階差モデルでは無くなってしまった個人ごとの能力差の情報も組み入れることができる。

ただし、個人ダミーモデルは、個人の数が多くなれば、推定しなければならないダミー変数の係数が増えれば、それだけ多くなってしまう。

で、結局どれがよいのか？

何も考えずに全てのデータをプールして OLS 回帰した結果 o02p と、階差をとって OLS 回帰した結果 o02d、それに個人ダミーを導入した回帰の結果 o02i を、AIC で比較してみる。

```
> AIC(o02p)
```

```
[1] 59.97655
```

```
> AIC(o02d)
```

```
[1] 18.21006
```

```
> AIC(o02i)
```

```
[1] 33.15717
```

この場合、 AIC が最も小さく、o02d の階差モデルがもっともあてはまりがよい、ということになる。

しかし、これはケースバイケースであって、いつもそうとは限らない。

いずれもうまくいかないケース

ところで、C 君は、前半、一生懸命授業に出ていたのに、中間テストの成績が芳しくなく、後半はほとんど授業に出なくなってしまいました。

これは、能力 γ_i が、他の説明変数 x_{1i} と相関しているということで、この γ_i が観測できないからといって誤差項として処理してしまったら、OLS 回帰がうまくいくための条件 $E[X_{1i}\varepsilon] = 0$ に反する。

このような場合、どうしたらよいかというのは、次回で説明しましょう。

7.3 説明変数との相関

7.3.1 内生性の何が問題か？

例題

賃金（y 万円／年）と学歴（教育年数:x_1）との関係を考える。

学歴が 1 年伸びると賃金が 25 万円／年増加するとしよう。

ただし、総賃金は本人の能力 $z_1 \in [-5, 5]$ と性別 $z_2 \in \{-1, 1\}$ によって就ける業種や職種が違うため、賃金は大きく異なる。

すなわち、能力 $z_1 = -5$ に比べて $z_1 = 5$ の人は平均で 200 万円ほど高い仕事に就けるし、男性（$z_2 = 1$）の方が女性（$z_2 = -1$）よりも平均で 200 万円高い賃金を得ている。

この関係式は次のようになるものとしよう。

$$y = 300 + 25x_1 + 20z_1 + 100z_2 + \varepsilon$$

ただし、$\varepsilon \sim N(0, 20)$。

ところが、ここで注意すべきは、学歴 x_1 の選択が内生的に決まるということ。

すなわち、z_1 や z_2 によって期待できる賃金 y が違うし、学歴 x_1 年にするには相応の教育コストがかかる。

したがって、z_1 の高い人ほど長い学歴を選択するだろうし、男性ほど学歴が長くなる可能性が高い。

もちろん、学歴は家庭環境（$K \sim N(0, 2)$）にもよるから、その関係は、

$$x_1 = 10 + 0.5z_1 + 5z_2 + K + \eta$$

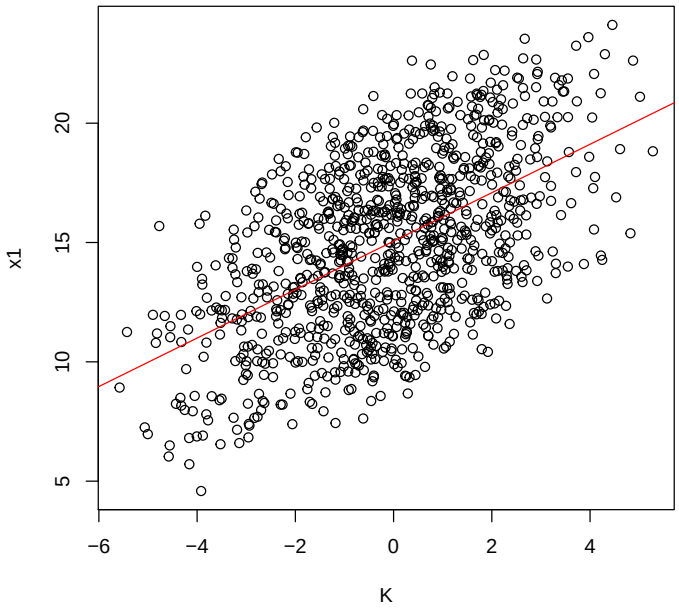
と表せる。ただし、$\eta \sim N(0, 1)$。

サンプルデータの生成

```
>  N<-1000
>   z1<-runif(N,-5,5)
>   z2<-sample(c(-1,1),N,T)
>   K<-rnorm(N,0,2)
>   eta<-rnorm(N,0,1)
>   x1<-15+0.5*z1+2.5*z2+K+eta
```

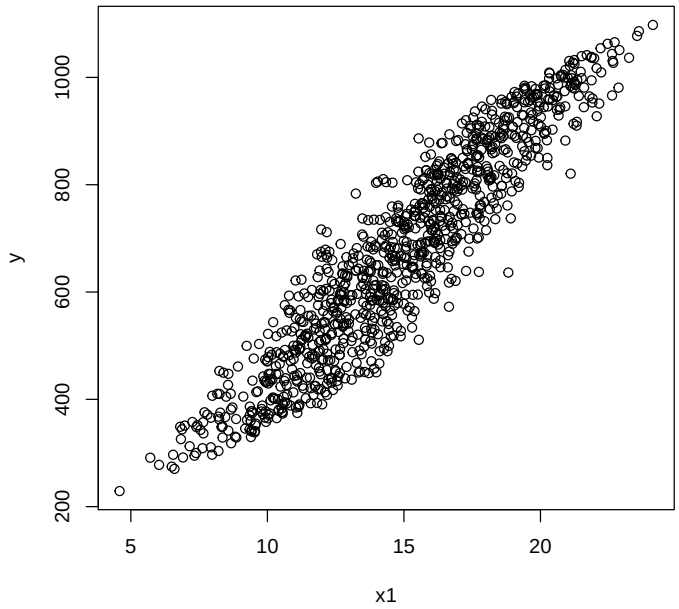
学歴と家庭環境との相関を見る。

```
> plot(x1~K)
> abline(lm(x1~K),col=2)
```



y を発生させ、学歴 x_1 との相関を確認する。

```
>   eps<-rnorm(N,0,5)
>   y<-300+25*x1+20*z1+100*z2+eps
>   plot(y~x1)
```



OLS 推定

内生性があろうとなかろうと、$E[\varepsilon] = 0$ であり、かつ $E[X'\varepsilon] = 0$ でありさえすれば、OLS 推定量は不偏である。

```
> o01<-lm(y~x1+z1+z2)
> summary(o01)
```

```
Call:
lm(formula = y ~ x1 + z1 + z2)

Residuals:
     Min       1Q   Median       3Q      Max
-17.3921  -3.2741  -0.0701   3.0545  15.2640

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 300.02815    1.04901   286.0   <2e-16 ***
x1           25.00248    0.06966   358.9   <2e-16 ***
z1           19.96126    0.06348   314.5   <2e-16 ***
z2          100.16834    0.23527   425.8   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.899 on 996 degrees of freedom
Multiple R-squared:  0.9994,        Adjusted R-squared:  0.9994
F-statistic: 5.205e+05 on 3 and 996 DF,  p-value: < 2.2e-16
```

問題なのは、z_1 や z_2 が抜け落ちた場合。

z_1 や z_2 のデータが得られなくても、それらが x_1 と相関していなければ、誤差項が $20z_1 + 200z_2 + \varepsilon$ となるだけで、$E[x'(20z_1 + 200z_2 + \varepsilon)] = 0$ だから不偏である。

ところが、x_1 が z_1 や z_2 と相関していると、これが欠落することで、$E[x'(20z_1 + 200z_2 + \varepsilon) \neq 0$ となり不偏性ではなくなる！（ココがみそ！）

```
> o01<-lm(y~x1)
> summary(o01)
```

```
Call:
lm(formula = y ~ x1)

Residuals:
     Min       1Q   Median       3Q      Max
-231.712  -49.273    5.043   50.187  192.752

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -66.6935     9.2766  -7.189 1.27e-12 ***
x1           49.6572     0.6038  82.245  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 69.51 on 998 degrees of freedom
Multiple R-squared:  0.8714,        Adjusted R-squared:  0.8713
F-statistic:  6764 on 1 and 998 DF,  p-value: < 2.2e-16
```

7.3.2 内生性への対応

操作変数法

一般的に、回帰方程式

$$y = x\beta + z\gamma + \varepsilon$$

を考える。ただし、x は内生変数、z は外生変数で $E(z) = 0$。

ここで、z が観測できなかった場合を考える。

この場合、$z\gamma + \varepsilon$ を誤差項として扱わなければならないことになる。

この場合、OLS 推定量は、

$$\hat{\beta} = (x'x)^{-1}x'y$$

となる。
つまり、

$$\hat{\beta} = \beta + (x'x)^{-1}x'(z\gamma + \varepsilon)$$

ということであり、x と z の間に相関がなければ、

$$E[\hat{\beta}] = \beta$$

で、$\hat{\beta}$ は不偏である。
しかし、$E[x'\varepsilon] = 0$ はありうるが、$E[x'z] = 0$ とは限らない。
$E[x'z] \neq 0$ なら、そのぶんがバイアスとなる。
x と z に相関があるのなら、x を z で回帰してみる。

$$x = z\delta_1 + K\delta_2 + \eta$$

ただし、K は y と無関係な x の説明変数。
z は観察できないが、K なら観察できるとすると、操作変数法が使える。
x を K で OLS 回帰して予測値 $\hat{x}$ を得る。

$$\hat{x} = K\hat{\delta}$$

これを使って

$$\hat{\beta}_{IV} = (\hat{x}'x)^{-1}\hat{x}'y$$

が得られる。このとき、

$$E[\hat{\beta}_{IV}] = \beta + E[(\hat{x}'x)^{-1}\hat{x}'z\gamma] + E[(\hat{x}'x)^{-1}\hat{x}'\varepsilon]$$

であるから、z と $\hat{x}$（つまり K）が無相関で、ε と $\hat{x}$ も無相関なら、不偏推定量が得られる。
（ただし、$\hat{x}$ の位置に注意！！！）

不偏性を確認する
x_1 を決める変数の中で、家庭環境 K だけは賃金 y に影響しないから、これを操作変数に用いる。
K で x_1 を回帰

```
>  o02<-lm(x1~K)
```

$\hat{x}$ を得る

```
>  x<-cbind(rep(1,N),x1)
>  xp<-predict(o02)
>  xh<-cbind(rep(1,N),xp)
```

$\hat{x}$ と $z\gamma + \varepsilon$ との相関を確認する

```
>  cor(xh[,2],0.5*z1+5*z2+eta)
```

```
[1] -0.002313662
```

相関はほとんどない！

操作変数法による推定値 $\hat{\beta}_{IV}$

```
>  solve(t(xh)%*%x)%*%t(xh)%*%y
```

```
         [,1]
   297.46490
x1  25.26107
```

うまく推定できているようだ！

実際、

$$\hat{\beta}_{IV} = (\hat{x}'x)^{-1}\hat{x}'y$$

において、

$$y = \alpha + \beta_1 x_1 + \gamma_1 z_1 + \gamma_2 z_2 + \varepsilon$$

なので、それぞれの項に y を分解してみると、

$(\hat{x}'x)^{-1}\hat{x}'(\alpha + \beta_1 x_1)$ は

```
> solve(t(xh)%*%x)%*%t(xh)%*%(300+25*x1)
```

```
    [,1]
     300
x1    25
```

$(\hat{x}'x)^{-1}\hat{x}'(\gamma_1 z_1)$ は

```
> solve(t(xh)%*%x)%*%t(xh)%*%(20*z1)
```

```
          [,1]
   -16.977746
x1   1.170646
```

$(\hat{x}'x)^{-1}\hat{x}'(\gamma_2 z_2)$ は

```
> solve(t(xh)%*%x)%*%t(xh)%*%(100*z2)
```

```
          [,1]
   13.3688740
x1 -0.8420287
```

$(\hat{x}'x)^{-1}\hat{x}'\varepsilon$ は

```
> solve(t(xh)%*%x)%*%t(xh)%*%eps
```

```
           [,1]
    1.07377319
x1 -0.06754325
```

となり、$z\gamma + \varepsilon$ のところは、かなり小さく、不偏性が得られそうである。

ちなみに、後の考察のために、次の値も確認しておく。

```
> solve(t(xh)%*%x)
```

```
                        xp
    0.057829104 -0.0038071617
x1 -0.003807162  0.0002550538
```

```
> t(xh)%*%(20*z1)
```

```
          [,1]
     496.3638
xp 11998.9704
```

```
> t(xh)%*%(100*z2)
```

```
       [,1]
    800.00
xp 8640.14
```

```
> t(xh)%*%eps
```

```
         [,1]
    65.56212
xp 713.81934
```

弱相関の問題

ただし、K と x の相関が弱い場合は、微妙になる。

このことを、x が定数項＋ 1 変数の場合で考えてみる。

$$(\hat{x}'x)^{-1} = \left(\begin{bmatrix} 1 & \cdots & 1 \\ \hat{x}_{11} & \cdots & \hat{x}_{1n} \end{bmatrix} \begin{bmatrix} 1 & x_{11} \\ \vdots & \vdots \\ 1 & x_{1n} \end{bmatrix} \right)^{-1}$$

$$= \begin{bmatrix} n & \sum_i x_{1i} \\ \sum_i \hat{x}_{1i} & \sum_i \hat{x}_{1i} x_{1i} \end{bmatrix}^{-1}$$

$$= \frac{1}{n\sum_i \hat{x}_{1i}x_{1i} - \sum_i x_{1i} \sum_i \hat{x}_{1i}} \begin{bmatrix} \sum_i \hat{x}_{1i}x_{1i} & -\sum_i x_{1i} \\ -\sum_i \hat{x}_{1i} & n \end{bmatrix}$$

X と $\hat{X}$ の相関が 0 に近いと、分母の

$$n\sum_i \hat{x}_{1i}x_{1i} - \sum_i x_{1i} \sum_i \hat{x}_{1i}$$

が 0 に近いということになる。

ということは、$(\hat{x}'x)^{-1}$ の値が大きくなってしまう。

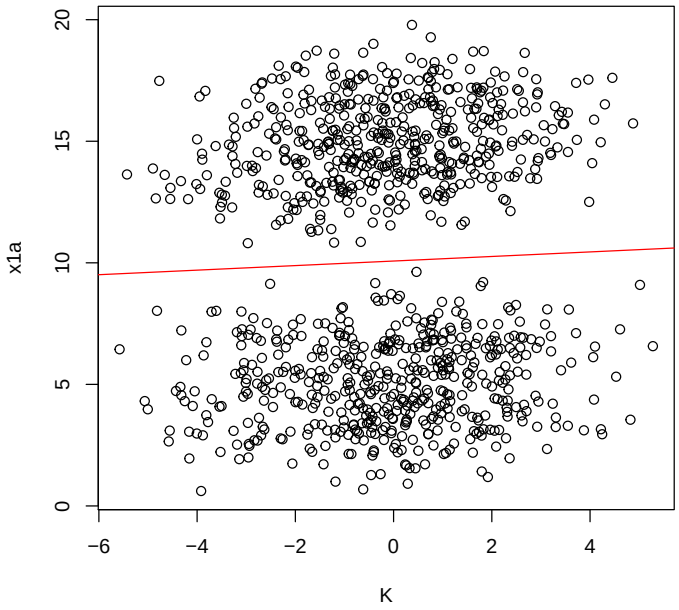
したがって、$\hat{x}'z\gamma$ や $\hat{x}'\varepsilon$ が小さくても、$E[(\hat{x}'x)^{-1}\hat{x}'z\gamma + \varepsilon]$ が大きな値をとりやすくなり、それがそのままバイアスとなる。

弱相関の問題を確かめてみる

操作変数 K と x_1 との相関が弱かった場合、不偏性が危うくなることを確かめる。

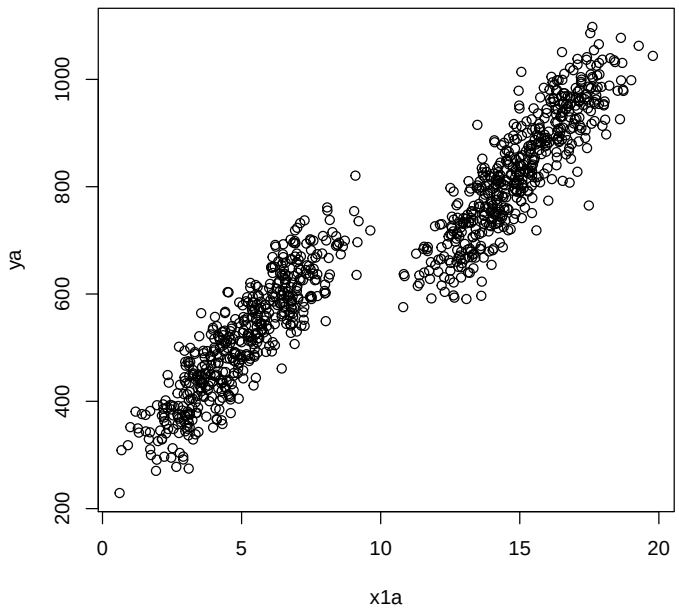
x_1 を発生させるが、K の係数が 1 から 0.1 に小さくなっている。その他は同じ。

```
> x1a<-10+0.5*z1+5*z2+0.1*K+eta
> plot(x1a~K)
> abline(lm(x1a~K),col=2)
```



この x_1 を使って y を発生させる。x_1 の値は同じだが、その他は同じ。

```
> ya<-300+25*x1+20*z1+100*z2+eps
> plot(ya~x1a)
```



さきほどと同じように $\hat{x}$ を求める。

```
> xa<-cbind(rep(1,N),x1a)
> o02a<-lm(x1a~K)
> xhma<-predict(o02a)
> xha<-cbind(rep(1,N),xhma)
```

$\hat{\beta}_{IV}$ を求めると、

```
> solve(t(xha)%*%xa)%*%t(xha)%*%ya
```

```
          [,1]
   -2078.2083
x1a   273.5988
```

バイアスが生じている。y を分解してみると、

$(\hat{x}'x)^{-1}\hat{x}'(\alpha + \beta_1 x_1)$ は

```
> solve(t(xha)%*%xa)%*%t(xha)%*%(300+25*x1a)
```

```
    [,1]
     300
x1a   25
```

となり、これは β そのもの

$(\hat{x}'x)^{-1}\hat{x}'(\gamma_1 z_1)$ は

```
> solve(t(xha)%*%xa)%*%t(xha)%*%(20*z1)
```

```
          [,1]
   -127.07094
x1a   12.67909
```

となり結構大きな値となっている。

$(\hat{x}'x)^{-1}\hat{x}'(\gamma_2 z_2)$ も

```
> solve(t(xha)%*%xa)%*%t(xha)%*%(100*z2)
```

```
         [,1]
   92.557313
x1a -9.119882
```

で大きい。

$(\hat{x}'x)^{-1}\hat{x}'\varepsilon$ は

```
> solve(t(xha)%*%xa)%*%t(xha)%*%eps
```

```
          [,1]
    7.4258665
x1a -0.7315505
```

ε の項のバイアスもそこそこ生じている。

特に、$(\hat{x}'x)^{-1}$ の部分を、さきほどと比べてみると、

```
> solve(t(xha)%*%xa)
```

```
                    xhma
    3.0297222 -0.30102876
x1a -0.3010288  0.02991965
```

であり、ずいぶん大きくなっていることがわかる。

後段は以下である。

```
> t(xha)%*%(20*z1)
```

```
          [,1]
     496.3638
xhma 5417.8059
```

```
> t(xha)%*%(100*z2)
```

```
          [,1]
     800.000
xhma 7744.178
```

```
> t(xha)%*%eps
```

```
          [,1]
     65.56212
xhma 635.18566
```

二段階最小二乗法: tsls 関数

R の tsls 関数を使う。

二段階最小二乗法（two step least square: TSLS）を行う tsls 関数は、ライブラリ sem に入っている。

操作変数法は１変数で x_1 を回帰して $\hat{x}_1$ を求めるが、二段階最小二乗法は、２変数以上でも回帰するが、全く同じことをやっている。

１変数で回帰して $\hat{x}_1$ を求めた二段階最小二乗法は、操作変数法と同じ。

```
> library(sem)
> summary(tsls(y~x1,~K))
```

```
 2SLS Estimates

Model Formula: y ~ x1

Instruments: ~K

Residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-210.1000  -91.2700    0.1192    0.0000   94.1200  201.4000

             Estimate Std. Error  t value   Pr(>|t|)
(Intercept) 297.464901  27.136709 10.96172 < 2.22e-16 ***
x1           25.261074   1.802187 14.01690 < 2.22e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 112.8454226 on 998 degrees of freedom
```

性別 z_2 も使えるなら、以下のように x_1 を z_2 と K で回帰して使うことができる。

z_2 は、y と $\hat{x}_1$ の両方の回帰に使われている。

```
> library(sem)
> summary(tsls(y~x1+z2,~K+z2))
```

```
 2SLS Estimates

Model Formula: y ~ x1 + z2
```

```
Instruments: ~K + z2

Residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-112.7000  -48.1800    0.7996    0.0000   50.4200  104.8000

               Estimate  Std. Error  t value   Pr(>|t|)
(Intercept) 284.8596251  13.5690589 20.99332 < 2.22e-16 ***
x1           26.0550081   0.9019773 28.88654 < 2.22e-16 ***
z2           94.2882383   2.8676866 32.87955 < 2.22e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 57.6531119 on 997 degrees of freedom
```