第 1 章

とりあえず使ってみる

R は汎用的な統計処理ツールとして、統計分析をやる人にはたいへん重宝されています。グラフなどの描画も簡単にできます。たとえば、R には plot 関数というのがありますが、特に指定しなくても、データに応じて適切なグラフを描き分けてくれたりします。ところが、こうしたグラフを、もうちょっと見栄え良く仕上げたくて、細かい設定をしようとすると、途端にたいへんだったりします。

そこに ggplot2 というパッケージが出てきました。これは Hadley Wickham さんが作成したもので、シンプルできれいなグラフを描けて、報告書や論文のように、フォーマットが決まったグラフを描くのにも都合がよいようです。そんなめでたいパッケージがあるなら、こっちを使ってみようということで、ここでは主に、ggplot2 を使ったグラフ作成の方法を解説していきます。

ところでこの「パッケージ」とは何かですが、R には、合計したり、乱数を発生させたり、グラフを描いたりする様々な関数が用意されています。これらの関数はいくつかのパッケージ（あるいはライブラリとも呼びます）に分けて入れられていて、R を起動した時には、base や stats、graphics　など、基本的なパッケージが読み込まれて、これらに含まれる関数は起動後すぐに使えるようになています。それ以外の関数は、必要なパッケージを読み込んで使います。

このパッケージは、やろうと思えば自分でも作成できます。多くの人が実に様々なパッケージを作成し、無償で提供してくれています。ggplot2 もそうしたパッケージのひとつです。

Wickham さんは、plyr というパッケージも提供してくれています。ggplot2 のグラフを描く前に、データを前処理をする必要があるのですが、これに plyr が役立ちます。同じ作者なので相性は言わずもがなです。R はあらゆる形にデータを加工できますが、やってみると意外と地味で面倒な処理が多かったりします。plyr は、こうした面倒な処理をうまいことやってくれます。これもつかってみましょう。

ここの解説は、自分の研究室の学生向けに作成したものです。私はパソコンは全く基礎ができていないので、もっと整然とした使い方もあるように思います。ggplot2 の詳しい解説は、Chang 2013 などを参照していただき、拙いところは補いながら読んでください。

1.1　帯グラフを描く

次のような帯グラフを描いてみましょう。

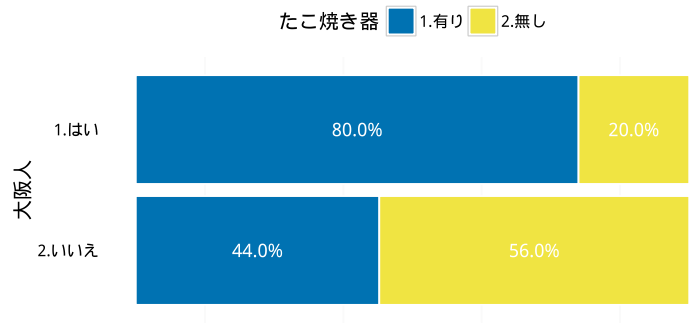


図 1.1: 帯グラフの出力例

この帯グラフは、アンケートで、大阪人か否か、たこ焼き器を持っているかどうかを聞いた結果をクロス集計したものです。

1.1.1　準備

まず、エクセルに入力したアンケート調査の結果を R に読み込みます。

1. csv ファイルを作成する
   エクセルで作成したアンケート調査結果を、(xlsx 形式ではなく）CSV 形式で保存してください。（ここでは takoyaki.csv と名前をつけました。）　決まりとしては、1 行目が変数名、2 行目以降がデータです。データは、数字でも文字でも構いません。

	A	B	C
1	大阪人	たこ焼き器	
2	1.はい	1.有り	
3	2.いいえ	1.有り	
4	2.いいえ	1.有り	
5	1.はい	1.有り	
6	2.いいえ	1.有り	
7	2.いいえ	2.無し	

図 1.2: エクセルの入力イメージ

完全なデータは以下です。

表 1.1: 練習用データセット

	大阪人	たこ焼き器
1	1. はい	1. 有り
2	2. いいえ	1. 有り
3	2. いいえ	1. 有り
4	1. はい	1. 有り
5	2. いいえ	1. 有り
6	2. いいえ	2. 無し
7	2. いいえ	2. 無し
8	1. はい	1. 有り
9	1. はい	2. 無し
10	2. いいえ	2. 無し
11	2. いいえ	2. 無し
12	2. いいえ	1. 有り
13	2. いいえ	2. 無し
14	2. いいえ	2. 無し
15	2. いいえ	1. 有り
16	1. はい	1. 有り
17	1. はい	2. 無し
18	2. いいえ	2. 無し
19	2. いいえ	1. 有り
20	1. はい	1. 有り
21	2. いいえ	1. 有り
22	1. はい	1. 有り
23	2. いいえ	2. 無し
24	2. いいえ	2. 無し
25	2. いいえ	1. 有り
26	2. いいえ	1. 有り
27	1. はい	1. 有り
28	2. いいえ	2. 無し
29	2. いいえ	2. 無し
30	2. いいえ	1. 有り
31	2. いいえ	2. 無し
32	1. はい	1. 有り
33	2. いいえ	2. 無し
34	2. いいえ	2. 無し
35	2. いいえ	1. 有り

2. 作業フォルダの変更

CSV ファイルを保存したフォルダを作業フォルダに変更します(例えば、`c:\Users\hogehoge\Documents` を作業フォルダに指定するとします)。R でのファイルの入出力は、このフォルダで行われることにに

なります。変更には `setwd()`（set workdrive）関数を使います。フォルダの区切りが \（バックスラッシュ）や ¥（円マーク）でははなく、/（スラッシュ）であることに注意してください。[*1]

```
> setwd("c:/Users/hogehoge/Documents")
```

3. CSV ファイルの読み込み
   CSV ファイルの読み込みには、read.csv 関数を使います。読み込んだデータは d01 に入れられ、画面上には何も表示されません。

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

4. 最初のいくつかのデータだけ表示するとこんな感じ。

```
> head(d01)
    大阪人 たこ焼き器
1   1. はい     1. 有り
2 2. いいえ     1. 有り
3 2. いいえ     1. 有り
4   1. はい     1. 有り
5 2. いいえ     1. 有り
6 2. いいえ     2. 無し
```

5. 以下の練習用に d01 を作成する場合は、以下を R のコンソールにコピペしてください。

```
d01<-data.frame(
        大阪人=c("1. はい","2. いいえ","2. いいえ","1. はい","2. いいえ",
                 "2. いいえ","2. いいえ","1. はい","1. はい","2. いいえ",
                 "2. いいえ","2. いいえ","2. いいえ","2. いいえ","2. いいえ",
                 "1. はい","1. はい","2. いいえ","2. いいえ","1. はい",
                 "2. いいえ","1. はい","2. いいえ","2. いいえ","2. いいえ",
                 "2. いいえ","1. はい","2. いいえ","2. いいえ","2. いいえ",
                 "2. いいえ","1. はい","2. いいえ","2. いいえ","2. いいえ"),
    たこ焼き器=c("1. 有り","1. 有り","1. 有り","1. 有り","1. 有り","2. 無し",
                 "2. 無し","1. 有り","2. 無し","2. 無し","2. 無し","1. 有り",
                 "2. 無し","2. 無し","1. 有り","1. 有り","2. 無し","2. 無し",
                 "1. 有り","1. 有り","1. 有り","1. 有り","2. 無し","2. 無し",
                 "1. 有り","1. 有り","1. 有り","2. 無し","2. 無し","1. 有り",
                 "2. 無し","1. 有り","2. 無し","2. 無し","1. 有り"))
```

[*1] Windows のフォルダの区切りは 円マーク ¥ですが、UNIX 系では/を使います。UNIX 系の作法でつくられたインターネットも URL の記述は http://www.eesog.ges.kyoto-u.ac.jp/emm/ のように、 /ですね。R で円マーク ¥は特別な意味に使うので、ファイルパスを表すのに ¥を使うとエラーになります。どうしても ¥を使いたいときは、C:¥¥Users¥¥Documents のように円マークを 2 つ重ねて使います。ちなみに、¥は環境によって \（バックスラッシュ）で表されることがあります。キーボードに ¥が刻印されていると思いますが、これは日本語版だけであって、US 版のキーボードでは ¥の代わりに \ が刻印されています。パソコンにとっては ¥も \ も全く同じ文字です。自分のパソコンでどちらが表示されているかわかりませんが、適宜読み替えてください。

1.1.2 plot 関数で描いてみる

最初に、R で従来から使われてきた標準の plot() 関数を使ったクロス集計についても知っておいた方がよいかもしれません。[*2]

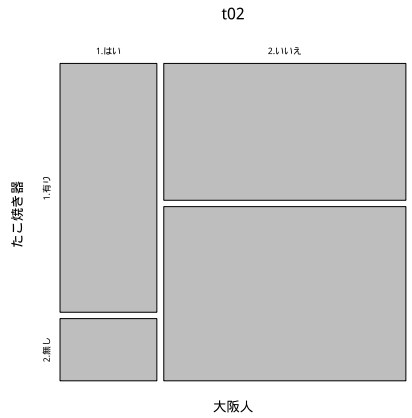
まず、クロス集計をします。R でクロス集計を行うには、table() 関数や xtabs() 関数を使うなど、様々な方法がありますが、ここでは、。xtabs() 関数を使ってみましょう。

```
> (t02<-xtabs(~大阪人 + たこ焼き器,d01))
          たこ焼き器
大阪人      1. 有り 2. 無し
  1. はい         8       2
  2. いいえ      11      14
```

xtabs() 関数に、データフレーム d01 の変数**大阪人**と**たこ焼き器**を使ってクロス集計することを、引数で指定してします。ここでは結果を t02 に入れていますが、そうすると画面上に結果が表示されないので、最初と最後に（）をつけて、結果も一緒に表示するようにしています。

クロス集計のグラフを表示します。

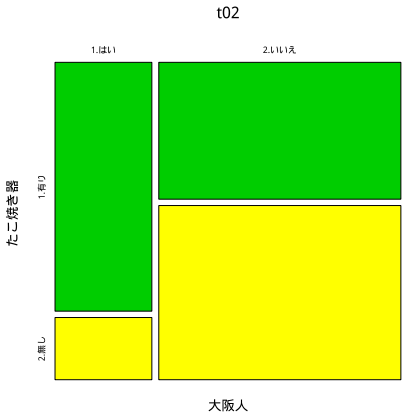
```
> plot(t02)
```



あっけないぐらい簡単ですね。色をつけてみましょう。col=c(3,7) が色の指定です。1 が黒、2 が赤、3 が緑、4 が青、5 がシアン、6 がマゼンタ、7 が黄色、8 がグレーです。

```
> plot(t02,col=c(3,7))
```

[*2] 従来の作図については、以下のサイトの第 47 節～を参照してください。
http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html



横軸に大阪人の割合が出ています。縦軸にたこ焼き器の所有率が出ています。R って簡単ですね。しかし、さきほどの ggplot2 のグラフのように加工しようとしたら、ちょっとたいへんです。

1.2　パッケージの読み込み

続いて、先に描いた ggplot2 の書き方の解説をしていきましょう。

ggplot2 は標準ではインストールされていません。あらかじめインストールしてください。インストールは、RStudio [*3] を使っていれば、右下の枠「Package」のタブで、「install」をクリックし、「Packages」のところに「ggplot2」と入力するだけです。

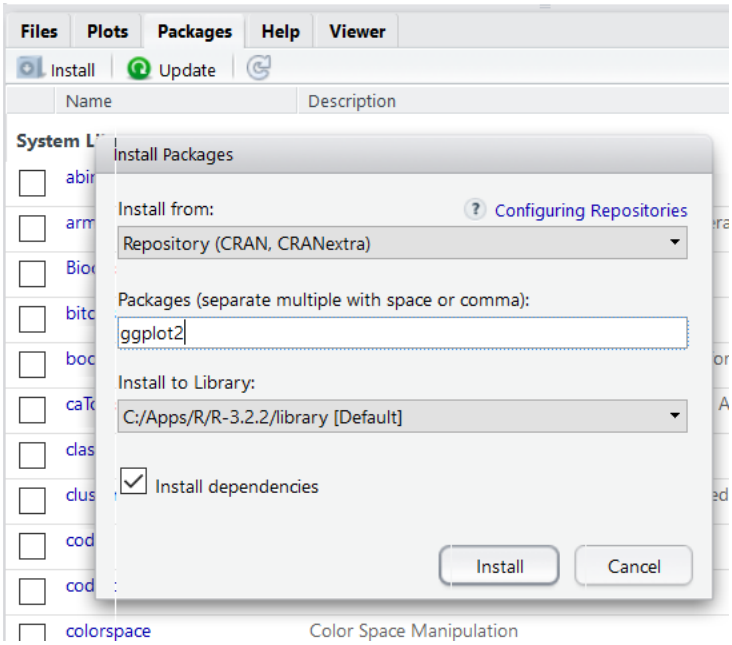


図 1.3: RStudio でのパッケージのインストール

RStudio を使っていなくても、R のコンソール上でもいけます。

[*3] RStudio は R に素晴らしい操作環境を与えてくれます。これを使わない手はないです。以下からダウンロードできます。
https://www.rstudio.com/products/rstudio/download/

```
> install.packages("ggplot2")
```

で OK のはずです。

この作業は、1 回やれば、R のバージョンアップをしない限り、やらなくていいです。

ggplot2 をインストールしたら、それを利用できるように `library()` 関数を使って読み込みます。この作業が以下です。一緒に、plyr、scales といった他の 2 つのパッケージも読み込みます。

```
> library(ggplot2)
> library(plyr)
> library(scales)
```

この作業は、RStudio だと「Packages」のところで、読み込みたいライブラリにチェックマークをつけるだけでできます。

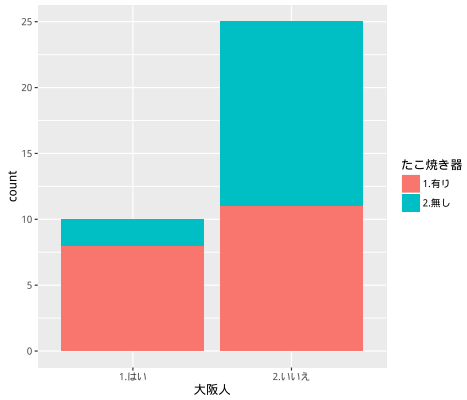
これで、ggplot2 と plyr と scales が使えるようになります。この作業は、R を起動する度に行います。R の起動後は、1 回やれば、終了するまで大丈夫です。

1.3 個票データからそのまま描く

1.3.1 積み重ね棒グラフ

はじめに、以下のように、積み重ね棒グラフを描きます。

```
> ggplot(data=d01,aes(x=大阪人,fill=たこ焼き器),stat="count")+geom_bar()
```



解説しましょう。このコマンドは、`ggplot()` と `geom_bar()` の 2 つのパートでできています。前者の `ggplot()` というのは、グラフを描くよという宣言みたいなもので、実際の描画は `geom_bar()` が行っています。`geom_bar()` は棒グラフですが、他に `geom_plot()`（散布図）とか、`geom_line()`(折れ線グラフ) などもあります。[*4]

[*4] ggplot2 で基本的なグラフを書くための関数は以下
棒グラフ：`geom_bar()`
散布図：`geom_plot()`
折れ線グラフ：`geom_line()`
ヒストグラム：`geom_hist()`
箱ひげ図：`geom_boxplot`
円グラフ：`geom_bar()` の棒グラフを `coord_polar("y")` で変換（Y 値のプロットを長さではなく角度にして描画）

描画に使うデータは、引数に data=d01 として指定しています。具体的にどんなふうに描画をしたらよいかというのは、aes() の中に記述します。*5

ここでは、x=大阪人 として、グラフの横軸は、変数「大阪人」の値を使うように指定しています。「大阪人」の値は「はい」か「いいえ」しかないので、この場合、グラフは 2 つの棒になります。

また、fill=たこ焼き器 は、たこ焼き器の値「有り」「無し」で塗り分けよ（fill、塗りつぶせ）、と意味です。

stat="count" というのは、データの数をカウントしてくださいということです。

data=d01 も aes(x=... も、ggplot() の中に記述してありますが、じつはこれ、ggplot() と geom_bar() のどっちに書いても構いません。全部 ggplot() に入れてもいいです。

```
ggplot(data=d01,aes(x=大阪人,fill=たこ焼き器),stat="count")+geom_bar()
```

全部 geom_bar() に入れてもいいです。

```
ggplot()+geom_bar(data=d01,aes(x=大阪人,fill=たこ焼き器),stat="count")
```

aes() については、外に出しても構いません。

```
ggplot(data=d01)+aes(x=大阪人,fill=たこ焼き器)+geom_bar(stat="count")
```

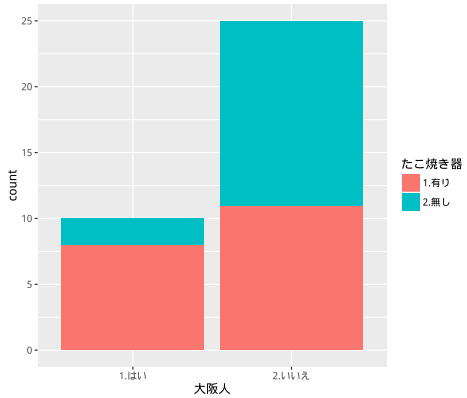
いずれも全く同じグラフが描けます。こうすることによって、さまざまなグラフを描き分けるときに、同じ部分を共用したり、条件に応じて描き分けたりすることができます。

たとえば、ggplot() の部分を g02 に入れてしまいましょう。

```
> g02<-ggplot(data=d01,aes(x=大阪人,fill=たこ焼き器),stat="count")
```

そのうえで、geom_bar() を付け足せば、同じグラフが書けます。

```
> g02+geom_bar()
```

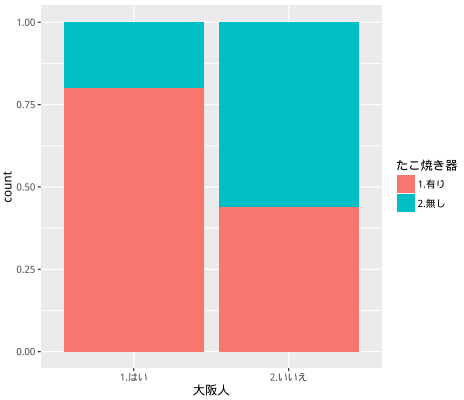


1.3.2　100 %積み上げグラフにする

position="fill"で、棒グラフを 100 %積み上げグラフにすることができます。棒の位置（position）をいっぱいまで (”fill”) ということでしょうか。

*5 aes というのは、asethetic の略で、いわゆるエステですね。グラフをきれいに書くための指定ということなのですが、実際に指定しているのは、グラフの記述に絶対必要な X や Y にどのデータを使うかなどで、このあたりの感性は私にはよくわかりません。

```
> g02+geom_bar(position="fill")
```



ここで、g02 は、

```
> ggplot(data=d01,aes(x=大阪人,fill=たこ焼き器),stat="count")
```

でしたね。ですから、これは

```
> ggplot(data=d01,aes(x=大阪人,fill=たこ焼き器),stat="count")+
+         geom_bar(position="fill")
```

と同じことです。くどいようですが・・・。

ところで、引数 position="fill"を記述する位置ですが、調子に乗って

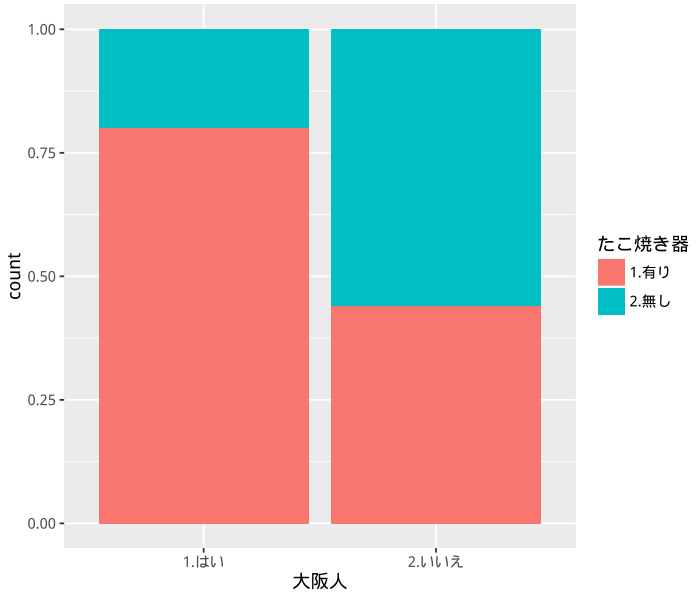
```
ggplot(data=d01,aes(x=大阪人,fill=たこ焼き器),stat="count",position="fill")+
   geom_bar()
```

としてもうまくいきません。さすがに、棒グラフ特有の引数は、geom_bar() に入れておく必要があるようです。便利な ggplot2 ですが、どの引数をどの関数に入れてよいか、そのあたりの判断が、私にはちょっとわかりにくところもあります。

1.3.3 横棒グラフにする

coord_flip() でグラフを横向きにできます。

```
> g02+geom_bar(position="fill")+
+   coord_flip()                                    #グラフを横向きに※
```

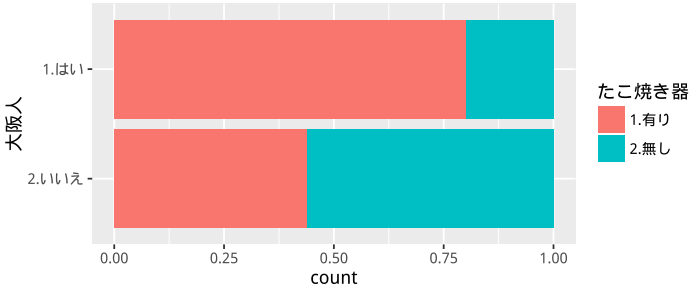


この関数の “coord” は coordinate の意味のようで、軸の位置関係をいじる関数に使われているようです。coord と名のつく関数は、ほかに `coord_fix()` があります。これは X 軸と Y 軸のスケールを同じに調整します。`coord_poler` は Y 軸を長さではなく角度で表すことで、棒グラフを円グラフに変換します。`coord_flip` は、その名のとおり、X 軸と Y 軸の位置関係を flip（ひっくり変え）させます。

1.3.4　X 軸の項目の順番を入れ替える

`scale_x_discrete()` 関数に `limits=`という引数を与えることで、X 軸が項目（discrete）の場合の順番を入れ替えることができます。

```
> g02+geom_bar(position="fill")+
+   coord_flip()+                                          #グラフを横向きに
+   scale_x_discrete(limits=c("2. いいえ","1. はい")) #X 軸の並びを指定※
```

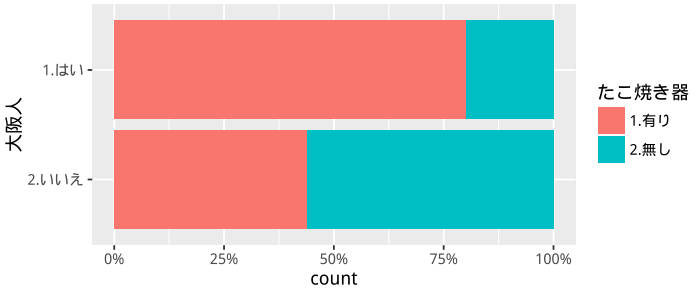


棒グラフを横向きにした場合に、右から左に並んだ棒が、下から上の順番となる。これを上から下に並んだものにしたい。エクセルなんかは、軸そのものが逆転できて、ggplot2 でも、X 軸が数値なら `scale_x_reverse()` 関数でできるのですが、X 軸が項目の場合に、軸を逆転する方法は見つけられませんでした。しかたないので、`scale_x_discrete()` 関数で、項目の並び方自体を変えています。

1.3.5 Y 軸の目盛をパーセント表記にする

scale_y_continuous() 関数に引数 labels=percent を加えることで、Y 軸の目盛をパーセント表記にすることができます。

```
> g02+geom_bar(position="fill")+
+   coord_flip()+                                          #グラフを横向きに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+ #X 軸の並びを指定
+   scale_y_continuous(labels=percent)                     #Y 軸の目盛を%表記※
```

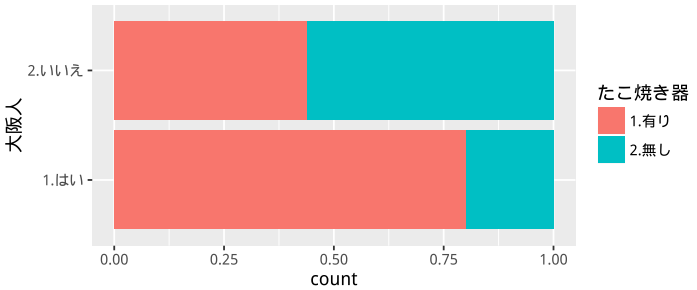


1.3.6 凡例を上に配置する

theme() 関数に引数 legend.position="top"を与えることて、凡例を上に配置することができます。"top"の代わりに"bottom"で下に配置、"right"で右、"left"で左です。

ここで使っている theme() 関数というのは、グラフの体裁を決めるためにに使われます。

```
> g02+geom_bar(position="fill")+
+   coord_flip()+                                          #グラフを横向きに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+ #X 軸の並びを指定
+   scale_y_continuous(labels=percent)+                    #Y 軸の目盛を%表記
+   theme(legend.position="top")                           #凡例を上に配置※
```

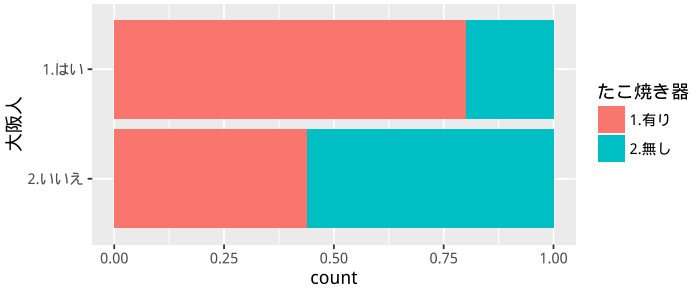


1.3.7 Y 軸のラベルを消す

theme 関数の引数@axis.title.x=に element_blank() を与えることで、横軸のラベルを消すことができます。横軸は Y 軸ではないのかと思うのですが、グラフを横向きにしているので、themex() 関数では X 軸として扱うようです。しかし、scale_y_continuous() 関数では横軸が Y 軸扱いなので、そこんとこはよくわ

かりません。

```
> g02+geom_bar(position="fill")+
+   coord_flip()+                                          #グラフを横向きに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+ #X 軸の並びを指定
+   scale_y_continuous(labels=percent)+                    #Y 軸の目盛を%表記
+   theme(legend.position="top",                           #凡例を上に配置
+         axis.title.x=element_blank())                    #横軸のラベルを消す※
```



ちなみに、Y 軸のラベルを消す方法として、`ylab("")` を加える方法もあります。ただし、`yalab()` 関数は、そもそも Y 軸のラベルを書き換える時に使うもので、`ylab("たこ焼き器の所有率")` といった形で使います。`ylab("")` は Y 軸のラベルを消したのではなくて、空白（""）で書き換えているだけです。そのため、Y 軸のラベルは消えていても、Y 軸のラベルを記入するためのスペースが確保されたままで、余計な空白ができます。上記の方法だと、そうしたスペースも含めて消去してくれます。

1.3.8　値ラベルを追加する

値ラベルは、`geom_text()` 関数で与えることができます。引数 `data=`で、表示したい値の入ったデータフレームを指定した後、同じく引数として `aes()` 関数を使い、その中の引数として、位置を `y=` 、表示したい値を `label=` で与えます。

しかし、問題は、この `data=`に与えるデータフレームをどうやって作成するかです。例えば、大阪人「はい」のたこ焼き器「有り」の人の割合は 80 %ですが、個票を使っているので、この値そのものは d01 には入っていません。それでも、実際にグラフを描いているのだから、どこのかの段階でその値は計算されているはずです。しかし、その値が格納されているところを見つけることができませんでした。

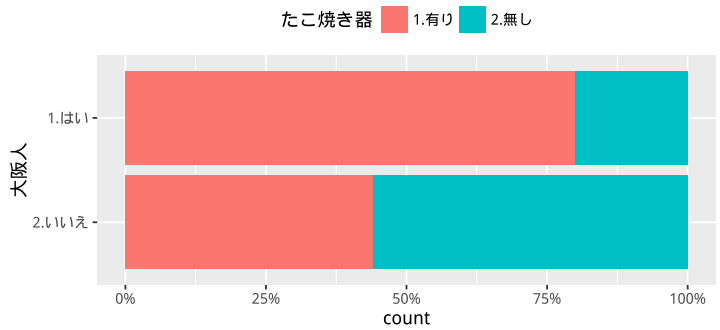
仕方ないので、直接、値を与えることとにします。d01 から計算する方法もあるのですがこれについては後述します。とりあえええず、どんな扱いをしているかを理解するために、手動で値を与えます。

```
> (p01<-data.frame(大阪人=c("1. はい","1. はい","2. いいえ","2. いいえ"),
+                  たこ焼き器=c("1. 有り","2. 無し","1. 有り","2. 無し"),
+                  prop=c(.80,.20,.44,.56),
+                  lposition=c(.4,.9,.22,.72)))
    大阪人 たこ焼き器 prop lposition
1   1. はい     1. 有り 0.80      0.40
2   1. はい     2. 無し 0.20      0.90
3 2. いいえ     1. 有り 0.44      0.22
4 2. いいえ     2. 無し 0.56      0.72
```

ここで、lposition というのは、値ラベルを表示したい位置（label position）で、各グラフの真ん中の位置を計算して与えています。例えば、大阪人のグラフの場合、0.8、0.2 で並んでいますが、0.8 は $0.8 - 0.8/2 = 0.4$、

0.2 は $0.8 + 0.2 - 0.2/2 = 0.9$ という具合です。

```
> g02+geom_bar(position="fill")+
+   coord_flip()+                                        #グラフを横並びに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+ #X 軸の並びを指定
+   scale_y_continuous(labels=percent)+                  #Y 軸の目盛を%表示に
+   theme(legend.position="top",                         #凡例を上に配置
+         axis.title.x=element_blank())+                 #横軸のラベルを消す
+   geom_text(data=p01,aes(y=lposition,label=prop))      #値ラベルを加える※
```



ラベルをパーセント表示したい場合は、引数 label=の部分を、paste() 関数を使って、以下のようにすればよいです。

```
> paste(p01$prop*100,"%",sep="")
[1] "80%" "20%" "44%" "56%"
```

paste() 関数は、引数の値や文字をつなげて、文字列として出力してくれます。引数に sep=""とあるのは、値や文字をつなぐときに間に何も入れるな、という意味になります。これを指定しないと半角のスペースが入ってしまいます。

```
> paste(p01$prop*100,"%")
[1] "80 %" "20 %" "44 %" "56 %"
```

sprintf() 関数を使うと、数値を文字列として出力する場合、表示形式を指定（string print with format）することができます。小数点以下 1 桁で出力したい場合は、表示形式として引数に、"%.1f"と指定します。ここで %

```
> sprintf("%.1f",p01$prop*100)
[1] "80.0" "20.0" "44.0" "56.0"
```

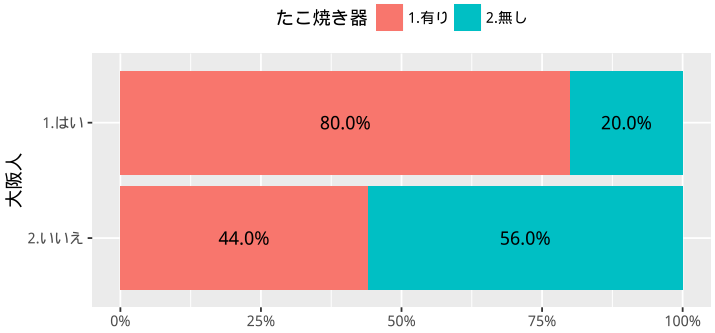
%をつけたい場合はしっぽに %% をつけます。

```
> sprintf("%.1f%%",p01$prop*100)
[1] "80.0%" "20.0%" "44.0%" "56.0%"
```

結局、%の値ラベルをつけるには以下のようにすればよいです。

```
> g02+geom_bar(position="fill")+
+   coord_flip()+                                        #グラフを横並びに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+ #X 軸の並びを指定
+   scale_y_continuous(labels=percent)+                  #Y 軸の目盛を%表示に
```

```
+   theme(legend.position="top",                          #凡例を上に配置
+         axis.title.x=element_blank())+                  #横軸のラベルを消す
+   geom_text(data=p01,                                   #値ラベル加える
+             aes(y=lposition,                                #値ラベルの位置
+                 label=sprintf("%.1f%%",p01$prop*100))) #値ラベルを%表記※
```



それにしても、手動で与えた p01 というデータフレームは、なんとかできないのか？次は、ライブラリ plyr を使って、これをなんとかしてみましょう。

1.4　一旦集計してから描画する

1.4.1　集計表を作成する

クロス集計表を作成する時、私はこれまで `xtabs()` 関数を使ってきました。こんな感じです。

```
> (t01_0<-xtabs(~大阪人 + たこ焼き器,d01))
          たこ焼き器
大阪人      1. 有り 2. 無し
  1. はい         8       2
  2. いいえ      11      14
```

ライブラリ plyr では少し違った形で集計します。とりあえず、ライブラリ plyr を読み込みましょう。

```
> library(plyr)
```

ライブラリ plyr にある `count()` 関数を使って集計します。ただし、他のライブラリにも `count` という名前の関数があることがあり、これと混乱することがあるので、ライブラリ plyr にある `count()` 関数であるということを示すために、`plyr::count()` としています。

```
> (t01<-plyr::count(d01))
    大阪人 たこ焼き器 freq
1   1. はい     1. 有り    8
2   1. はい     2. 無し    2
3 2. いいえ     1. 有り   11
4 2. いいえ     2. 無し   14
```

`xtabs()` 関数は、行列として出力しますが、`count()` 関数はデータフレームを出力します。1 列と 2 列めに、大阪人か否か、たこ焼き器を持っているかどうかで場合分けして、3 列めに `freq` という名前で、それぞれの頻度が出力されています。ggplot2 は、基本的にデータとして、データフレームを使いますので、`count()` 関数の方が都合がいいのです。

str() 関数を使うと、引数のデータがどんなデータかを確かめることができます。

```
> str(t01_0)
 int [1:2, 1:2] 8 11 2 14
 - attr(*, "dimnames")=List of 2
  ..$ 大阪人     : chr [1:2] "1. はい" "2. いいえ"
  ..$ たこ焼き器: chr [1:2] "1. 有り" "2. 無し"
 - attr(*, "class")= chr [1:2] "xtabs" "table"
 - attr(*, "call")= language xtabs(formula = ~大阪人 + たこ焼き器, data = d01)
```

xtabs() 関数で集計したものは、2 次元の行列です。R の行列は、1 つの並びのベクトルを行と列に区切って使っています。str() 関数の出力にもそう書いてあります。1 行目を読むと、8 11 2 14 という整数（int）を [1:2,1:2]（行番号 1〜2、列番号 1〜2）に配置、と表記されています。

ただし、この行列はただの行列ではなくて、2 つの次元に大阪人とたこ焼き器という名前（dimnamees）がつけられていたり、xtabs 関数（とそれが呼び出す table 関数）の出力だといった属性 (attr) も加わっています。たとえば、単純に作成した行列の情報はこれだけです。比べてみてください。

```
> d09<-matrix(1:8,nrow=4,ncol=2)
> str(d09)
 int [1:4, 1:2] 1 2 3 4 5 6 7 8
```

一方、count() 関数で集計したものは、データフレームです。

```
> str(t01)
'data.frame':        4 obs. of  3 variables:
 $ 大阪人     : Factor w/ 2 levels "1. はい","2. いいえ": 1 1 2 2
 $ たこ焼き器: Factor w/ 2 levels "1. 有り","2. 無し": 1 2 1 2
 $ freq       : int  8 2 11 14
```

上の出力の 1 行目に data.frame だと書いてあります。このデータフレームは、3 つの変数に 4 つの観測値（obs.）があるとあります。そして、3 つの変数についての説明が続きます。

ここで行列と配列の違いについて確認しましょう

R で行列とデータフレームは似ているように思いますが、全然違います。行列は、ひとつのベクトルを行と列に区切ったものなので、行列に入れられるデータの種類は 1 種類です。整数（int）に 1 つでも実数（num）が混じっていれば行列全体が実数になります。実数のつもりで作成した行列の要素に 1 つでも文字（chr）が混じると、文字の行列となります。

整数の行列に実数をひとつ加えると・・・

```
> d09[2,2]<-0.1
> str(d09)
 num [1:4, 1:2] 1 2 3 4 5 0.1 7 8
```

実数の行列に文字をひとつ加えると・・・

```
> d09[4,1]<-"a"
> str(d09)
 chr [1:4, 1:2] "1" "2" "3" "a" "5" "0.1" "7" "8"
```

ひとつのベクトルは、3 次元以上に区切ることも可能で、それは配列（array）と呼ばれます。

```
> (d09<-array(1:8,dim=c(2,2,2)))
, , 1
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

     [,1] [,2]
[1,]    5    7
[2,]    6    8
> str(d09)
 int [1:2, 1:2, 1:2] 1 2 3 4 5 6 7 8
```

ベクトル、行列、配列の 3 つは、もともと同じデータの型で、次元に関する情報が違うだけです。行列は 2 次元の配列で、ベクトルは 1 次元の配列であって、配列の特殊型と考えることもできます。ですから、これらを総称して配列（array）と呼ぶこともあります。

データフレームは、整数、実数、文字など、種類の違うベクトルを各変数として、ひとつのデータセットとして扱うことができます。ただし、各変数はベクトルなので、その中に違う種類のデータを混在させることはできません。また、各変数のベクトルの長さ（これが obs. の数）は同じでないといけません。

```
> d09<-data.frame(x=1:3,y=c(1,0.2,3),z=c(1,0.2,"a"))
> str(d09)
'data.frame':        3 obs. of  3 variables:
 $ x: int  1 2 3
 $ y: num  1 0.2 3
 $ z: Factor w/ 3 levels "0.2","1","a": 2 1 3
```

ggplot2 は、データとしてデータフレームの形を要求します。

1.4.2　構成比を計算する

`ddply()` 関数を使うと、大阪人か否かでグループ分けして、それぞれのグループごとに構成比を計算することができます。

```
> (p01<-ddply(t01,"大阪人",transform,prop=freq/sum(freq)))
    大阪人 たこ焼き器 freq prop
1   1. はい     1. 有り    8 0.80
2   1. はい     2. 無し    2 0.20
3 2. いいえ     1. 有り   11 0.44
4 2. いいえ     2. 無し   14 0.56
```

`ddply()` 関数の最初の 2 文字 dd というのは、「データフレーム（d）からデーターフレーム（d）へ」という意味です。つまり、データフレームをいじって、その結果をデータフレームとして出力します。ライブラリ plyr には、他に adply 関数というのがありますが、これは、配列（array）をいじって、結果をデータフレーム (d) に出力する関数です。同じく、配列（a）から配列（a）は aaply 関数です。つまり、最初の 2 文字で、変換前と返還後のデータの形式がわかります。

`ddply()` 関数の最初の引数は、いじりたいデータフレームで、この場合 `t01` ですね。

次に、`"大阪人"`とありますが、これでグループ化する値を指定します。「データフレーム `t01` を`"大阪人"`の値でグループ化して、次の引数で与える関数をグループごとに計算してくれ」ということになります。

その次の引数 transform は、実際に計算するための関数で、さらにその次にある prop=... とあるのは、transform() 関数の引数です。

この transform() 関数は、引数で指定した計算をして、もとのデータフレームにくっつけてくれます。例えば、「t01 の各 freq の値の構成比を計算して、prop という名前で新しい列をくっつけろ」だったら、次のようにします。

```
> transform(t01,prop=freq/sum(freq))
    大阪人 たこ焼き器 freq       prop
1   1. はい     1. 有り    8 0.22857143
2   1. はい     2. 無し    2 0.05714286
3 2. いいえ     1. 有り   11 0.31428571
4 2. いいえ     2. 無し   14 0.40000000
```

prop の合計が 1 となっていますか？。

この計算を"大阪人"の値でグループ化して、それぞれのグループごとに実行したい場合は、ddply() 関数を使って、以下のようにします。

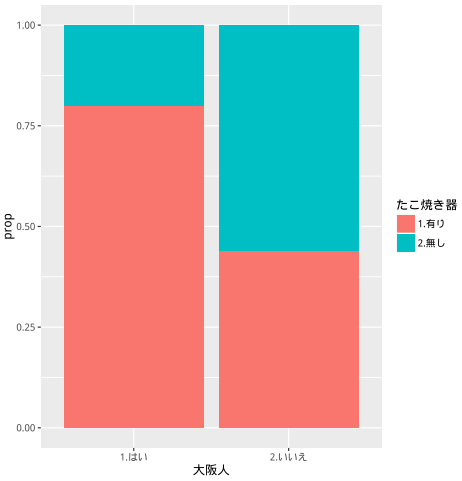
```
> ddply(t01,"大阪人",transform,prop=freq/sum(freq))
    大阪人 たこ焼き器 freq prop
1   1. はい     1. 有り    8 0.80
2   1. はい     2. 無し    2 0.20
3 2. いいえ     1. 有り   11 0.44
4 2. いいえ     2. 無し   14 0.56
```

大阪人の「1. はい」「2. いいえ」毎に、prop の合計が 1 となっているはずです。

1.4.3　一旦グラフを描いてみる

作成したデータフレーム p01（集計結果と構成比が入っている）を使って、グラフを描いてみましょう。やり方は、個票の時とほぼ同じです。

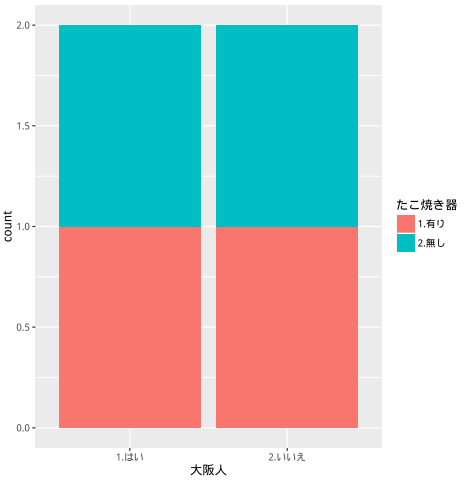
```
> ggplot(data=p01,aes(x=大阪人,y=prop,fill=たこ焼き器))+geom_bar(stat="identity")
```



個票でグラフを描いた時との違いは、引数 stat="identity"です。引数 stat="identity"は、グラフに表示する値は、y= で指定したそのまんまの値（つまり、identity てこと）。個票でグラフを描いたときは、stat=

という引数は指定しませんでした。これは、デフォルトが stat="count"となっているからです。もし、上のグラフで、stat= を指定しなかったら、p01 のデータは大阪人か否かのグループごとに、たこ焼き器があるかないかの 2 種類しかないので、それぞれ 1 個ずつと数えて、以下のようなグラフになってしまいます。

```
> ggplot(data=p01,aes(x=大阪人,fill=たこ焼き器))+geom_bar()
```

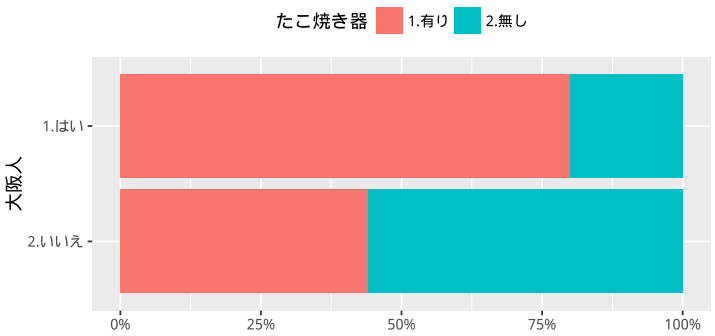


ちなみに、stat="count" ではで指定した変数の数を数えるので、y= の引数は用いません。

1.4.4　グラフの形を整える

ここまでできたら、あとは個票から描いた時と同じです。まとめていきましょう。

```
> ggplot(p01,                                          #データは p01
+        aes(x=大阪人,                                 #X 軸のデータは大阪人
+            y=prop,                                   #Y 軸のデータは prop
+            fill=たこ焼き器))+                        #たこ焼き器の値で塗り分ける
+   geom_bar(stat="identity",                           #棒の長さは prop の値そのまんま
+            position="fill")+                          #100 %積上げグラフに
+   coord_flip()+                                       #グラフを横向きに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+ #X 軸の下からの順番
+   scale_y_continuous(labels=percent)+                 #Y 軸の目盛は 100 %表記に
+   theme(legend.position="top",                        #凡例を上に
+         axis.title.x=element_blank())                 #X 軸のラベルを消す
```



1.4.5　データラベルの位置を計算する

構成比は p01 に代入していました。

```
> p01
    大阪人 たこ焼き器 freq prop
1   1. はい     1. 有り    8 0.80
2   1. はい     2. 無し    2 0.20
3 2. いいえ     1. 有り   11 0.44
4 2. いいえ     2. 無し   14 0.56
```

これに、値ラベル用の位置を計算して、lposition という名前で加えます。値ラベルの位置は当該のグラフの真ん中を指定します。大阪人「はい」でたこ焼き器「有り」の場合だと、0 から 0.8 の範囲がグラフの該当箇所なので、$0.8 - 0.8/2 = 0.4$ の位置がグラフの真ん中です。大阪人でたこ焼き器無しのグラフは、有りの人 0.8 に 0.2 が積み重ねられているので、その真ん中は、$0.8 + 0.2 - 0.2/2 = 0.9$ です。大阪人以外の人についても同様に計算できます。

ということは、大阪人か否かで分けたグループごとに、累計をとって、そこから当該の値の半分を差し引けば、積み重ねグラフの当該グラフの真ん中が計算できます。たった 2 つずつのグラフなので、そんなに法則だって考える必要もないかもしれませんが、この計算方法だと、たくさん値があっても、そのまま応用できますね。

ちなみに、累計は `cumsum()` 関数でできます。例えば、1, 2, ..., 10 の累計は、以下で計算できます。

```
> cumsum(1:10)
 [1]  1  3  6 10 15 21 28 36 45 55
```

`transform()` 関数を使うと、与えられたデータフレームに対して、指定した計算の結果を別の変数としてもとのデータフレームにくっつけてくれます。例えば、以下のように、x と y からなる d05 といったデータフレームをつくってみます。

```
> (d05<-data.frame(x=c(1,1,1,2,2),y=c(5,15,30,20,30)))
  x  y
1 1  5
2 1 15
3 1 30
4 2 20
5 2 30
```

y の構成比を計算して、z の名前でデータフレームにくっつけます。

```
> transform(d05,z=y/sum(y))
  x  y    z
1 1  5 0.05
2 1 15 0.15
3 1 30 0.30
4 2 20 0.20
5 2 30 0.30
```

`ddply()` 関数は、指定した関数をグループごとに計算してくれます。上の d05 の y の構成比 z を、x の値ごとにグループ化して求める場合は、以下のようにすればよいのです。

```
> ddply(d05,"x",transform,z=y/sum(y))
```

```
  x  y  z
1 1  5 0.1
2 1 15 0.3
3 1 30 0.6
4 2 20 0.4
5 2 30 0.6
```

x が 1 のグループと、x が 2 のグループで、それぞれ z の合計が 1 になっているはずです。

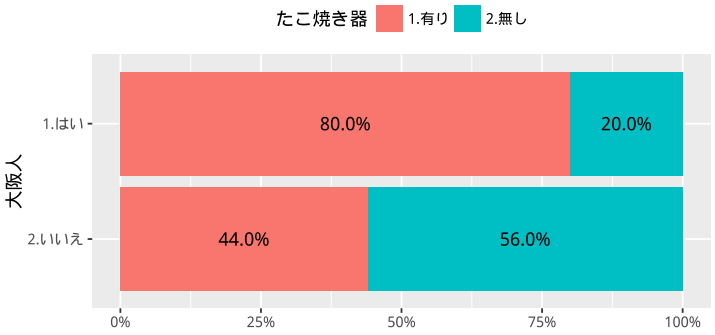
以上を使って、ラベル位置が計算できます。

```
> (p01<-ddply(p01,"大阪人",transform,lposition=cumsum(prop)-prop/2))
    大阪人 たこ焼き器 freq prop lposition
1   1. はい     1. 有り    8 0.80      0.40
2   1. はい     2. 無し    2 0.20      0.90
3 2. いいえ     1. 有り   11 0.44      0.22
4 2. いいえ     2. 無し   14 0.56      0.72
```

データフレーム p01 の"大阪人"の値でグループ化して、そのグループごとに、prop の累計から自身の prop の値の半分を引いた値を計算して、それを lposition の名前で p01 に追加しています。

このデータフレーム p01 を用いて、グラフを描きます。geom_text() 関数を使って、さきほど作成したグラフにデータラベルを追加します。

```
> ggplot(p01,                                          #データは p01
+        aes(x=大阪人,                                 #X 軸のデータは大阪人
+            y=prop,                                   #Y 軸のデータは prop
+            fill=たこ焼き器))+                        #たこ焼き器の値で塗り分ける
+   geom_bar(stat="identity",                           #棒の長さは prop の値そのまんま
+            position="fill")+                          #100 %積上げグラフに
+   coord_flip()+                                       #グラフを横向きに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+ #X 軸の下からの順番
+   scale_y_continuous(labels=percent)+                 #Y 軸の目盛は 100 %表記に
+   theme(legend.position="top",                        #凡例を上に
+         axis.title.x=element_blank())+                #X 軸のラベルを消す
+   geom_text(aes(y=lposition,                          #値ラベルの位置指定※
+                 label=sprintf("%.1f%%",prop*100)))    #小数点 1 桁で prop を 100 %表記※
```



引数 y=で、ラベルの位置を指定しています。表示するラベルは、label= として、%で表示しています。%表示には、表示形式を指定して文字列出力する sprintf() 関数を使っています。引数の"%.1f%%"の意味は、%.1f が、小数点以下 1 桁で出力するということであり、そのあとの %% が、後ろに%をくっつけるという意味です。

このやり方は、個票からグラフを描いた時にすでに説明してあります。ほとんど同じですが、個票から描いた時は、値を別のデータフレームとして与えているので、geom_text() 関数の中に、改めてそのデータフレーム名を指定しなければなりませんでしたが、今回は、描画データと同じデータフレームの中に、データラベルの値も位置情報もあるので、ここで指定する必要はありません。

1.5　グラフのデザインを変える

ggplot2 のグラフのデザインは、そこそこいいと思いますが、それでも報告書の書式などに応じて、ある程度はカスタマイズしたいものです。

何回も同じところを表示するのもうっというしいので、ここでいったん、作成したグラフの書式に関わる部分を g03 という名前で保存しておきましょう。

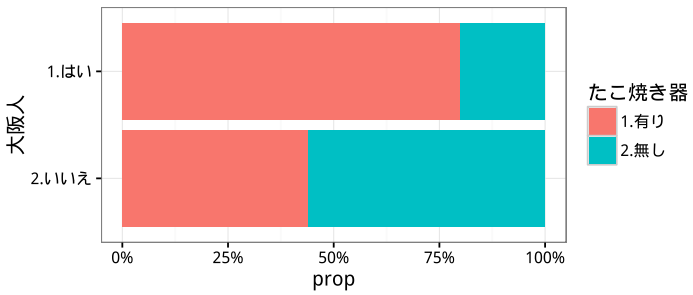
```
> g03<-ggplot(p01,aes(x=大阪人,y=prop,fill=たこ焼き器), #データと X と Y の指定
+             position="fill")+                          #100 %積上げグラフ
+   coord_flip()+                                        #グラフを横向きに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+   #X 軸の並びを指定
+   scale_y_continuous(labels=percent)                   #Y 軸目盛を%表記
```

1.5.1　組み込みのテーマを使う

ggplot2 では、ある程度見栄えのよいグラフの形式がテーマという形で与えられていています。theme_grey、theme_bw、theme_linedraw、theme_dark、theme_classic 等々。そのまま使うと、theme_grey が標準となっています。これまで描いてきたグラフは、これを使っていました。背景がグレイで、軸線が白い感じのグラフです。

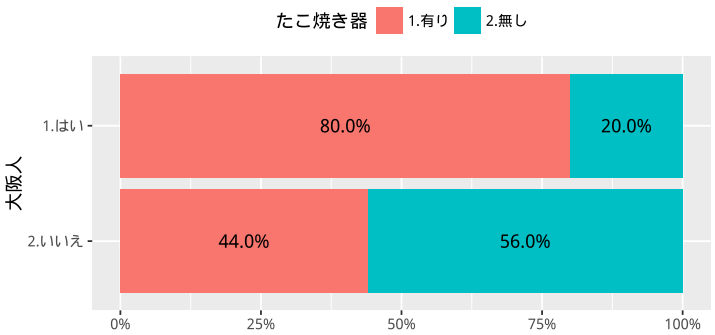
theme_bw の bw は、Black and White ということで、白黒基調のグラフとなっています。以下のように使います。

```
> g03+geom_bar(stat="identity")+
+   theme_bw()                                           #theme_bw を使用
```



theme_linedoraw だとこんな感じです。

```
> g03+geom_bar(stat="identity")+
+   theme_linedraw()                                     #theme_bw を使用
```



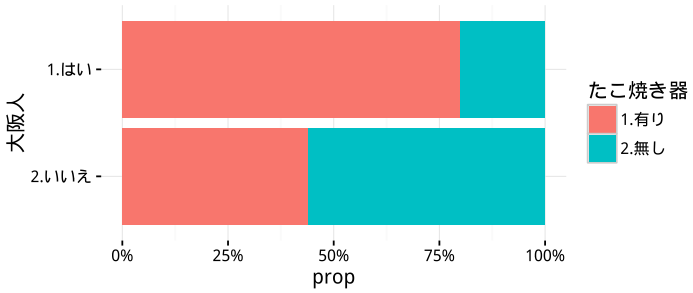
以降、theme_bw のテーマを基本に、自分好みのグラフに変更してみましょう。

1.5.2　パネルの枠線を消す

100 %積み上げグラフに軸目盛とかいりませんよね。最初は、グラフ要素を描く領域（ggplot2 は、ここここを “panel” と呼んでいるようです）のまわりの四角い枠線を消します。

`theme()` 関数に `panel.border=element_blank()` と指定します。

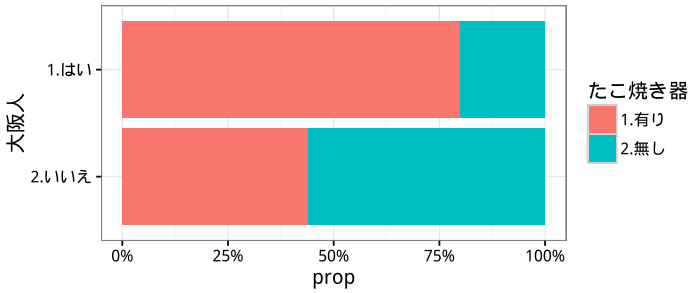
```
> g03+geom_bar(stat="identity")+
+   theme_bw()+                                  #theme_bw を使用
+   theme(panel.border=element_blank())          #パネルの枠線を消す※
```



1.5.3　目盛線を消す

`theme()` 関数に `panel.grid.major=element_blank()` と指定します。

```
> g03+geom_bar(stat="identity")+
+   theme_bw()+                                  #theme_bw を使用
+   theme(panel.border=element_blank(),          #パネルの枠線を消す
+         panel.grid.major=element_blank())      #目盛線を消す※
```



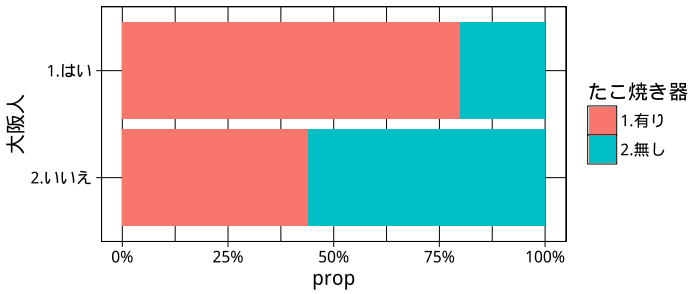
X 軸か Y 軸か片方だけ消したい場合は、panel.grid.major.x=element_blank() または panel.grid.major.y=element_blank() でいけます。

1.5.4　目盛と目盛ラベルを消す

軸の目盛を消したい場合は axis.ticks=element_blank()、目盛のラベルを消したい場合は、axis.text=element_blank())。

大阪人の「はい」「いいえ」は残したいので、目盛ラベルは Y 軸の方だけ消したい。その場合は、axis.text.y=element_blank()) とすればよいはずですが、そうするとなぜか大阪人の方のラベルが消えてしまいます。axis.text.x=element_blank()) とするとうまくいきます。横棒グラフにしたら、横軸はすでに X 軸ということか？

```
> g03+geom_bar(stat="identity")+
+   theme_bw()+                                  #theme_bw を使用
+   theme(panel.border=element_blank(),          #パネルの枠線を消す
+         panel.grid.major=element_blank(),      #目盛線を消す
+         axis.ticks=element_blank(),            #軸の目盛を消す※
+         axis.text.x=element_blank())           #目盛ラベルを消す※
```



だいたい書式は出そろったので、theme() 関数の部分を、g03t という名前で保存しておきましょう。

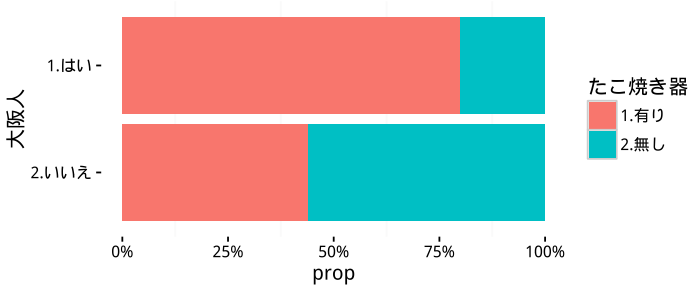
```
> g03t<-theme_bw()+                            #theme_bw を使用
+   theme(panel.border=element_blank(),          #パネルの枠線を消す
+         panel.grid.major=element_blank(),      #目盛線を消す
+         axis.ticks=element_blank(),            #軸の目盛を消す
+         axis.text.x=element_blank(),           #目盛ラベルを消す
+         axis.title.x=element_blank(),          #X 軸のタイトルを消す※
+         legend.position="top")                 #凡例を上に配置※
```

ここまで、なにげに省略していた、X 軸のタイトルを消す axis.title.x=element_blank() と、凡例を上に配置する legend.position="top"も、ここで再び加えてあります。

1.5.5　グラフ要素を白線で囲む

グラフ要素を白線で囲うと、境界がはっきりして見やすいように思います。`geom_bar()` 関数の中で、引数 `colour="white"`を指定すると、グラフ要素のまわりに白線を引いてくれます。

```
> g03+g03t+geom_bar(stat="identity",
+                   colour="white")                    #線の色指定※
```



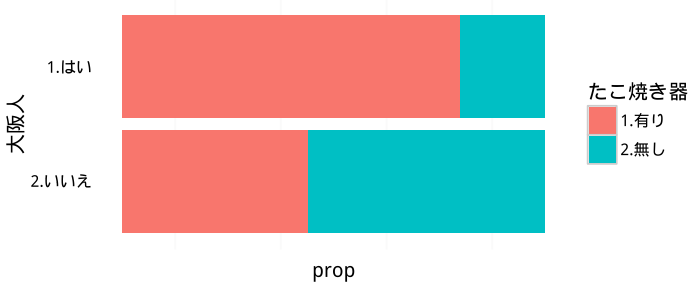
ちなみに、ggplot2 では、原則として、“colour” といえば線の色で、塗りつぶしは “fill” という用語を使うようです。

1.5.6　グラフ要素の色を変える

`scale_fill_manual()` 関数を使うと、グラフの要素の色を好きな色に塗り替えることができます。どんな色を使うかは、引数 `values=`に与えます。

色の指定の仕方は、`"red"`とか、`"blue"`とか、R で使える色の名前で指定することもできますが[*6]、ここでは直接 RGB を指定して、微妙な色を指定しています。

```
> g03+g03t+geom_bar(stat="identity",
+             colour="white")+                          #線の色指定
+   scale_fill_manual(values=c("#0072B2","#F0E442")) #塗りつぶしの色の指定※
```



`#0072B2` といったのが色の指定です。これは RGB といって、＃に続く最初の 2 桁が赤（R)、次の 2 桁が緑（G)、最後の 2 桁が青（B）を表します。この数が大きくなるほど、その色が強く出ます。

ただし、この値は 16 進数で表されていて、一番小さい値が 00（10 進数で 0）で、一番大きい値が FF（10 進数で 255）です。

[*6] R でどんな色の名前が使えるかは、`colors()` で表示させることができます。657 種類の名前があるようです。

#FF0000 が赤、#00FF00 が緑、#0000FF が青、#FFFF00 が黄、#FFFFFF が白、#000000 が黒です。#909090 とするとグレーだし、#9030C0 とすると青みの強い微妙な色になります。

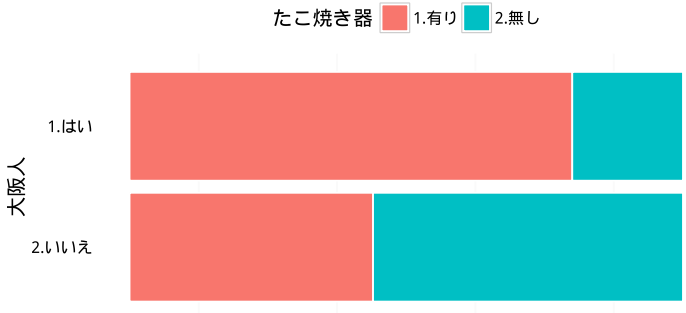
10 進数から 16 数への変換は、以下を参照してください。L0 の列が 16×0 ～ の 16 個の数、L1 の列が 16×1 ～ の 16 個の数、L14 の列が 16×14 ～ の 16 個の数、L15 が 16×15 ～ の 16 個の数です。

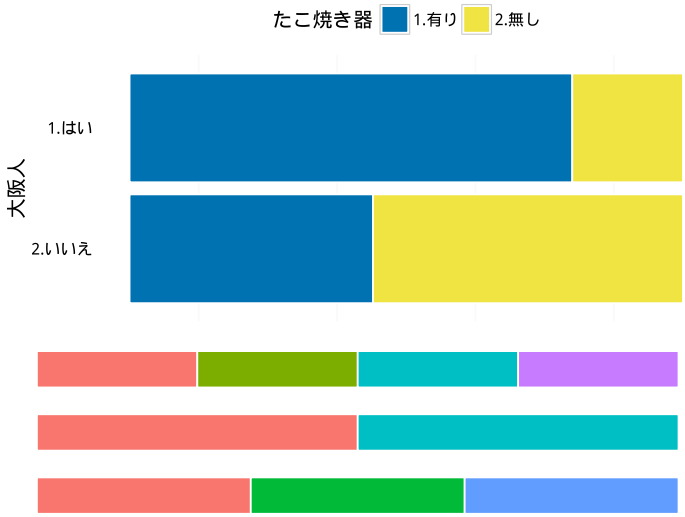
	L0	L1	L2	L14	L15
1	0 -> 0	16 -> 10	32 -> 20	224 -> E0	240 -> F0
2	1 -> 1	17 -> 11	33 -> 21	225 -> E1	241 -> F1
3	2 -> 2	18 -> 12	34 -> 22	226 -> E2	242 -> F2
4	3 -> 3	19 -> 13	35 -> 23	227 -> E3	243 -> F3
5	4 -> 4	20 -> 14	36 -> 24	228 -> E4	244 -> F4
6	5 -> 5	21 -> 15	37 -> 25	229 -> E5	245 -> F5
7	6 -> 6	22 -> 16	38 -> 26	230 -> E6	246 -> F6
8	7 -> 7	23 -> 17	39 -> 27	231 -> E7	247 -> F7
9	8 -> 8	24 -> 18	40 -> 28	232 -> E8	248 -> F8
10	9 -> 9	25 -> 19	41 -> 29	233 -> E9	249 -> F9
11	10 -> A	26 -> 1A	42 -> 2A	234 -> EA	250 -> FA
12	11 -> B	27 -> 1B	43 -> 2B	235 -> EB	251 -> FB
13	12 -> C	28 -> 1C	44 -> 2C	236 -> EC	252 -> FC
14	13 -> D	29 -> 1D	45 -> 2D	237 -> ED	253 -> FD
15	14 -> E	30 -> 1E	46 -> 2E	238 -> EE	254 -> FE
16	15 -> F	31 -> 1F	47 -> 2F	239 -> EF	255 -> FF

ちなみに、ggplot2 には、グラフを塗り分けるためのカラーパレットが用意されていて、それを切り替えて使うことができます。デフォルトは、`scale_fill_hue()` で、色相を等間隔に選択します。これまで描いたグラフでは、このカラーパレットが使われていました（といっても、２色しか使っていませんが）。他に、`scale_fill_discrete()` というののもありますが、これも同じ配色です。

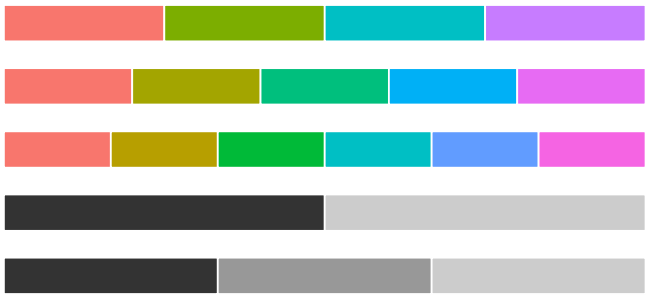
以下に、配色を見てみましょう。塗り分ける要素の数によって、色の配分が変わるようなので、それも確認してください。

- `scale_fill_hue()`（デフォルト）または `scale_fill_discrete()`

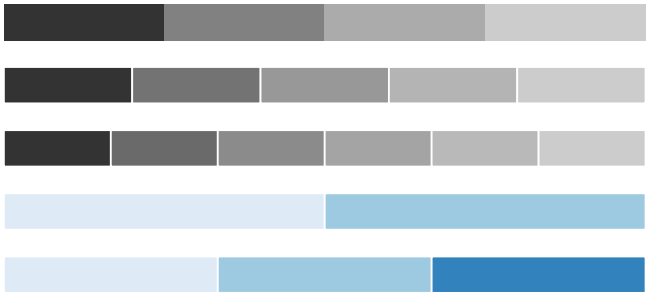




- scale_fill_grey()



- scale_fill_brewer()



scale_fill_manual() を使うと、自分でカラーパレットを作って使うこともできます。以下は、Chang 2013 に紹介されていた配色で、色覚異常に配慮した色なのだそうです。

```
> cb_palette<-c("#0072B2","#F0E442","#009E73","#56B4E9",
+               "#CC79A7","#D55E00","#E69F00","#999999")
```

- scale_fill_manual(values=cb_palette)

1.5.7　完成

あとは、データラベルを加えて、完成です。

最初から、まとめておきましょう。まず、データ指定と基本的な書式設定です。

```
> g03<-ggplot(p01,aes(x=大阪人,y=prop,fill=たこ焼き器), #データと X と Y の指定
+             position="fill")+                          #100 %積上げグラフ
+   coord_flip()+                                        #グラフを横向きに
+   scale_x_discrete(limits=c("2. いいえ","1. はい"))+   #X 軸の並びを指定
+   scale_y_continuous(labels=percent)                   #Y 軸目盛を%表記
```

次に、テーマを指定していじります。

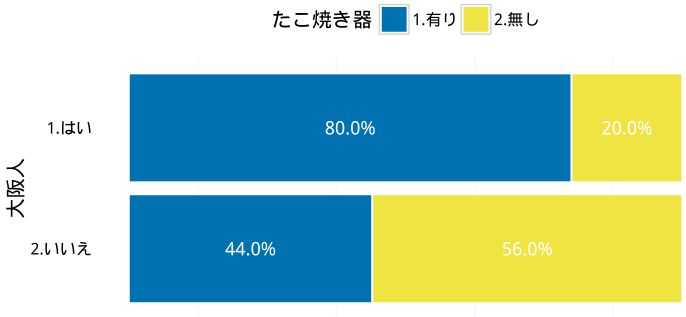
```
> g03t<-theme_bw()+                                    #theme_bw を使用
+   theme(panel.border=element_blank(),                #パネルの枠線を消す
+         panel.grid.major=element_blank(),            #目盛線を消す
+         axis.ticks=element_blank(),                  #軸の目盛を消す
+         axis.text.x=element_blank(),                 #目盛ラベルを消す
+         axis.title.x=element_blank(),                #X 軸のタイトルを消す
+         legend.position="top")                       #凡例を上に配置
```

カラーパレットを作成します。

```
> cb_palette<-c("#0072B2","#F0E442","#009E73","#56B4E9",
+               "#CC79A7","#D55E00","#E69F00","#999999")
```

色を指定し、データラベルを加えて、完成です。

```
> g03+g03t+  geom_bar(stat="identity",
+                     colour="white")+                 #線の色指定
+   scale_fill_manual(values=cb_palette)+              #塗りつぶしの色指定
+   geom_text(aes(y=lposition,                         #データら別の位置
+                 label=sprintf("%.1f%%",prop*100)),   #データラベルを%小数点 1 桁に
+             color="white")                           #文字色を白に
```



第 2 章

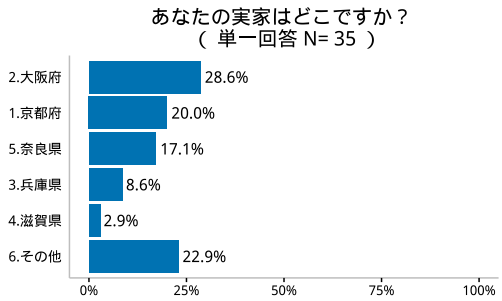
単純集計のグラフを描く

この章では、アンケート集計の実践方法についてしっかり解説します。アンケート調査票に記入された回答をエクセルに転記して、それを R に読み込み、ggplot2 で単純集計のグラフを描き、報告書を作成します。

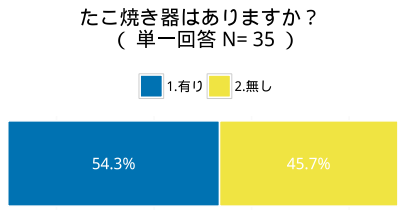
そんなことしなくてもエクセルでも同じことができると思うかもしれませんが、エクセルではひとつひとつグラフを描く必要があります。ところが R だと、グラフの描画を関数化することができます。100 個でも 1000 個でも自動的に同じフォーマットでグラフを描くことができます。しかも報告書の形で、ワードファイルや PDF ファイルに出力することもできます。

例題として、３つ以上の選択肢がある単一回答を棒グラフに、選択肢が２個だけの単一回答を帯グラフに、複数回答を棒グラフに、そして数値記入項目を階級を与えて棒グラフにします。

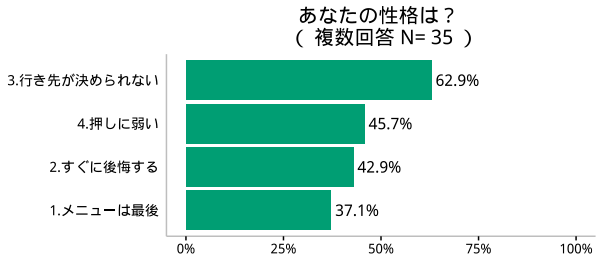
この章で説明した関数をそのまま使うこともできますが、もっと見やすいグラフにしたり、もっとスマートな関数にしたり、自分で工夫してみてください。



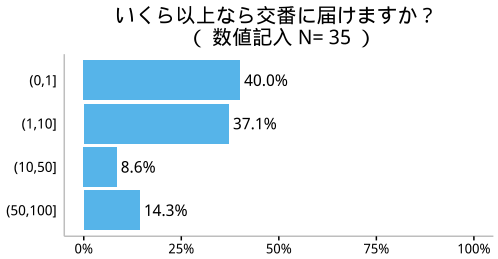
(a) 単一回答（棒グラフ）



(b) 単一回答（帯グラフ）



(c) 複数回答



(d) 複数回答

図 2.1: 本章で描くグラフ

2.1　エクセルでデータ入力

2.1.1　アンケート調査票

例題としたのは次のアンケート調査票です。問 1、2、3 が単一回答、問 4 が数値記入、問 5 が複数回答です。

回答してもらった結果は、エクセルを使ってデータとして入力します。そして、そのデータを R に読み込みます。

この手順を前提とすれば、調査票を作成する段階から、以下のルールに従っておいた方がよいと思います。

- 設問番号は、問 1、問 2、... でもよいし、設問数が多くなる場合は、設問の流れをわかりやすくするために、問 1、問 2、... を「・・・についてお聞きします。」というような漠然とした聞き方にして、具体的な質問は (1)、(2)、... として聞いてもよい。
- 問 1、(1)、(2)、... とする場合でも、(1)、(2)、... の番号は、調査票全体を通じてふった方がよい。例えば、問 1 に 3 つ設問があって、問 2 に 2 つ設問がある場合、問 1 （1)、(2)、(3) 、問 2 (4)、(5) という具合。設問の ID 管理がしやすくなります。
- 選択肢には必ず選択肢番号を振ります。これがないとデータ入力も集計も面倒になることがあります。
- 選択肢番号は、1,2,3,... の順番で振ります。データ処理に便利です。

問 1　あなたの実家はどこですか。次のいずれか 1 つに○印をつけてください。ただし、転居が多い方はこれまで最も長く住んでいた方をお答えください。

1. 京都府
2. 大阪府
3. 兵庫県
4. 滋賀県
5. 奈良県
6. その他

問 2 あなたの実家またはあなたの下宿にはたこ焼き器はありますか。次のいずれか 1 つに○印をつけてください。

1. 有り
2. 無し

問 3 もしあなたが現金で 1 万円札を拾ったら、あなたはそれを交番に届けますか？次のいずれか 1 つに○印をつけてください。

1. 届ける
2. 届けない

問 4 現金でお金が落ちていたとき、いくら以上ならあなたはそれを交番に届けますか？

（　　）円以上

問 5 あなたの性格についてお聞きします。以下の 1.〜4. についてあてはまるもの全てに○印をつけてください。

1. 友達とご飯やさんに入って、メニューを決めるのが最後になることが多い。
2. メニューを決めた後、「やっぱり別のにすればよかった」と後悔することが多い。
3. 友達と遊びに行くとき、どこに行くか自分できめずに、相手に決めさせることの方が多い。
4. ほんとうはイヤでも「どうしてもお願い」と言われると、つい「はい」と言ってしまい、自分でも「そんなにいやじゃなかったかも」と思い込んでしまうことが多い。

2.1.2 回答を選択肢番号で入力する

回収した調査票を、エクセルを使って入力します。まず、回答（response）を dset_r.csv という名前で保存しましょう。

CSV ファイルは、カンマ区切りのテキストファイルです。エクセルでファイルを保存するとき、ファイルの種類の指定を、「Excel ブック (*.xlsx)」ではなくて、「CSV (カンマ区切り)(*.csv) にすればよいですね。

入力のルールとしては、以下に従ってください。

- 回収した調査票にはすべて通し番号をふり、その番号をデータにもつけておく（これが ID の列に入力

	A	B	C	D	E	F	G	H	I
1	ID	Q01	Q02	Q03	Q04	Q05_01	Q05_02	Q05_03	Q05_04
2	1	2	1	2	10	0	0	1	1
3	2	6	1	2	50	0	0	1	1
4	3	1	1	NA	1	1	1	0	0
5	4	2	1	2	100	1	0	0	0
6	5	6	1	1	2	0	1	1	1
7	6	6	2	1	0.1	0	1	0	0

図 2.2: 回答結果の入力：dset_r.csv

された数字になります）

- 単一回答、数値記入、（ここにはないけど自由記入項目）には Q01、Q02、Q03 と ID をつけて、1 行目に入力する。（Q1、Q2、とはしない。設問数が 10 超えると、並べ替えなどで、Q1, Q10, Q2, ..., とかになってしまう）
- 複数回答には、Q05_01、Q05_02、というように選択肢ごとに列を確保して ID をつける。
- 単一回答の場合、回答は選択肢番号で入力する。
- 複数回答の場合、選択された場合 1、選択されていない場合 0 と入力する。
- 無回答の場合は、「NA」と入力する。
- 数値記入の場合、そのまま数字を入力する。ただし、桁数が多いとグラフに書いたとき目盛などがごちゃごちゃするので、単位を調整しておいたほうがよい場合もある（ここでは万円単位にしています）。

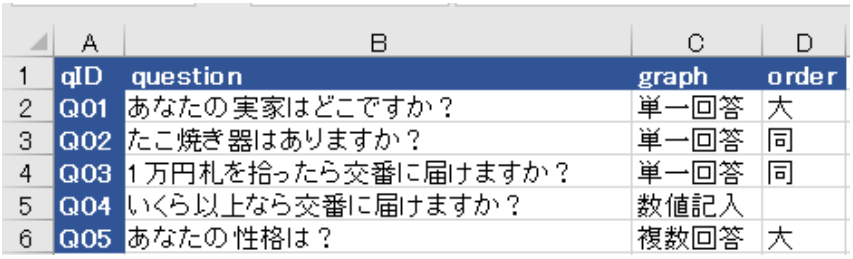
全てのデータを以下に示します。

表 2.1: 演習用データセット：回答

	ID	Q01	Q02	Q03	Q04	Q05_01	Q05_02	Q05_03	Q05_04
1	1	2	1	2	10.00	0	0	1	1
2	2	6	1	2	50.00	0	0	1	1
3	3	1	1		1.00	1	1	0	0
4	4	2	1	2	100.00	1	0	0	0
5	5	6	1	1	2.00	0	1	1	1
6	6	6	2	1	0.10	0	1	0	0
7	7	6	2	1	1.00	0	0	1	0
8	8	2	1	2	100.00	0	0	1	0
9	9	2	2	1	0.50	1	1	1	0
10	10	5	2	1	1.00	0	1	1	1
11	11	5	2	1	1.00	0	1	1	1
12	12	5	1	1	0.10	1	1	0	1
13	13	1	2	2	25.00	0	0	1	1
14	14	1	2	2	10.00	0	0	1	0
15	15	5	1	1	0.01	1	1	1	1
16	16	2	1	1	0.50	0	1	0	1
17	17	2	2	1	0.10	0	0	0	1
18	18	4	2	2	10.00	1	1	1	1
19	19	6	1	1	0.10	1	0	0	1
20	20	2	1	2	6.00	0	1	1	1
21	21	3	1	2	100.00	1	0	1	0
22	22	2	1	2	100.00	0	1	0	0
23	23	6	2	1	5.00	1	1	1	1
24	24	6	2	1	5.00	0	0	1	0
25	25	5	1	2	18.00	0	0	1	0
26	26	1	1	1	1.00	0	0	0	0
27	27	2	1	1	5.00	0	0	0	0
28	28	6	2	1	1.00	1	0	1	0
29	29	1	2	1	5.00	0	0	1	1
30	30	1	1	2	1.50	1	0	0	0
31	31	1	2	2	2.00	1	0	0	0
32	32	2	1	2	100.00	1	0	1	0
33	33	3	2	1	0.01	0	1	0	0
34	34	5	2	2	6.00	0	1	1	1
35	35	3	1	2	1.50	0	0	1	0

2.1.3　設問文とグラフの種類を入力する

設問（questions）の文言は別のワークシートにまとめて、dset_q.csv という名前で保存しましょう。ついでに、各設問に対して、どんなグラフを描くのか、それについても指定します。



	A	B	C	D
1	qID	question	graph	order
2	Q01	あなたの実家はどこですか？	単一回答	大
3	Q02	たこ焼き器はありますか？	単一回答	同
4	Q03	1 万円札を拾ったら交番に届けますか？	単一回答	同
5	Q04	いくら以上なら交番に届けますか？	数値記入	
6	Q05	あなたの性格は？	複数回答	大

図 2.3: 設問文とグラフの種類の指定：dset_q.csv

- 1 列目（qID）に、設問を識別するために、dset_r.csv で使った Q01、Q02、Q03 といった名前を記入します。
- qID について、dset_r.csv では、複数回答の場合 Q05_01, Q05_02, ... と分けて名前をつけましたが、設問文については一括して Q05 で大丈夫です。
- question の列には、設問文を入力します。そのままグラフの表題に使うので、もとの表現から大きく離れないように、しかも簡潔にします。
- question の列の設問文には、単一回答か複数回答か、数値記入かなども加えます。
- graph の列で、グラフの種類を指定します。以下のように記入します。

  単一　単一回答で選択肢が 3 つ以上の場合。横棒グラフを描きます。

  単一帯　単一回答で選択肢が 2 つだけの場合。帯グラフにします。

  複数　複数回答の場合。横棒グラフを描きます。

  数値　数値記入の場合。ヒストグラムを描きます。
- order の列は、選択肢の並び方を指定します。以下のように記入します。

  同　選択肢番号のとおり並べます。

  大　値が大きい順に並べます。

  任　任意の順に並べます。

完全なデータは以下です。

表 2.2: 演習用データセット：設問文とグラフの種類の指定

	qID	question	graph	order
1	Q01	あなたの実家はどこですか？	単一回答	大
2	Q02	たこ焼き器はありますか？	単一回答	同
3	Q03	1 万円札を拾ったら交番に届けますか？	単一回答	同
4	Q04	いくら以上なら交番に届けますか？	数値記入	
5	Q05	あなたの性格は？	複数回答	大

2.1.4　選択肢を入力する

選択肢（alternatives）の文言も、dset_a.csv という名前で保存しましょう。以下のような形式でまとめます。問 5 などは、選択肢の文言が長いので、それとわかるような別の表現をとっています。その場合は、グラフを描いたときに、もとの選択肢の文言がわかるように説明を加えておく必要があります。

選択肢には、「1. 京都府」のように、選択肢番号を一緒に書き込んでおきます。集計結果を回答の多い順で並べ替えたりするような場合、これがないと、もとの順番がわからなくなります。

設問 4 のような数値記入項目の場合は、回答の値の区切りを与えてください（「階級」と言いますね）。この例の場合は、0 より大きく 1 万円以下、1 万円より大きく 5 万円以下、...、50 万円より大きく 100 万円以下に区切ることにしました。回答の値を確認して、最小値と最大値がカバーできるように注意してください。

	A	B	C	D	E
1	Q01	Q02	Q03	Q04	Q05
2	1.京都府	1.有り	1.届ける	0	1.メニューは最後
3	2.大阪府	2.無し	2.届けない	1	2.すぐに後悔する
4	3.兵庫県			10	3.行き先が決められない
5	4.滋賀県			50	4.押しに弱い
6	5.奈良県			100	
7	6.その他				

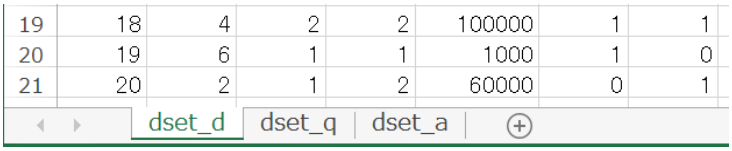
図 2.4: 選択肢の入力：dset_a.csv

完全なデータは以下です。

表 2.3: 演習用データセット：選択肢

	Q01	Q02	Q03	Q04	Q05
1	1. 京都府	1. 有り	1. 届ける	0	1. メニューは最後
2	2. 大阪府	2. 無し	2. 届けない	1	2. すぐに後悔する
3	3. 兵庫県			10	3. 行き先が決められない
4	4. 滋賀県			50	4. 押しに弱い
5	5. 奈良県			100	
6	6. その他				

以上、３つのワークシートを３つの csv ファイルとして保存しましたが、データの管理としては、これらをまとめてエクセルファイル (xlsx 形式）で保存しておき、修正等あった場合は、該当するワークシートを CSV 形式で上書き保存したほうがよいかと思います。



19	18	4	2	2	100000	1	1
20	19	6	1	1	1000	1	0
21	20	2	1	2	60000	0	1

dset_d　dset_q　dset_a

図 2.5: データをまとめて入れたエクセル・ファイル

2.2　単一回答の棒グラフ

単一回答について以下のようなグラフを描いてみましょう。

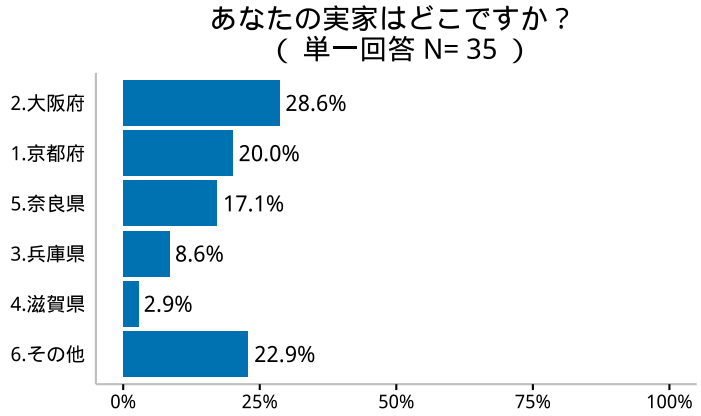


図 2.6: 単一回答の棒グラフ

2.2.1　準備

最初に、作業フォルダを設定します。dset_r、dset_q、dset_a の 3 つのファイルを保存したフォルダを作業フォルダとしましょう。

```
> setwd("c:/Users/hogehoge/Documents")
```

以後に必要とするライブラリを読み込みます。インストールされていない！と怒られる場合は、インストールしてください。RStudio の「Packages」のところの「Install」を使うか、`install.packages("ggplot2")`、`install.packages("plyr")`、... 等でできるはずです。

```
> library(ggplot2)
> library(plyr)
> library(scales)
> library(dplyr)
> library(gridExtra)
> library(showtext)
```

2.2.2　データの読み込み

1. 回答が入力されたデータ dset_r.csv を R に読み込みます。

   ```
   > d01r<-read.csv("dset_r.csv")
   ```

   演習用のサンプルデータが用意されていない場合は、以下を R コンソールにコピペして作成してください。

   ```
   d01r<-data.frame(ID=1:35,
          Q01=c(2, 6, 1, 2, 6, 6, 6, 2, 2, 5, 5, 5, 1, 1, 5, 2, 2, 4, 6, 2,
                3, 2, 6, 6, 5, 1, 2, 6, 1, 1, 1, 2, 3, 5, 3),
          Q02=c(1, 1, 1, 1, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1, 1, 2, 2, 1, 1,
   ```

```
      1, 1, 2, 2, 1, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1),
Q03=c(2, 2, NA, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 1, 2,
      2, 2, 1, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 2, 2),
Q04=c(10, 50, 1, 100, 2, 0.1, 1, 100, 0.5, 1, 1, 0.1, 25, 10,
      0.01,0.5, 0.1, 10, 0.1, 6, 100, 100, 5, 5, 18, 1, 5, 1, 5,
      1.5, 2, 100,0.01, 6, 1.5),
Q05_01=c(0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1,
         0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0),
Q05_02=c(0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0,
         1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0),
Q05_03=c(1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
         1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1),
Q05_04=c(1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
         1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0))
```

この回答データの最初の方はこんな感じです。

```
> head(d01r)
  ID Q01 Q02 Q03   Q04 Q05_01 Q05_02 Q05_03 Q05_04
1  1   2   1   2  10.0      0      0      1      1
2  2   6   1   2  50.0      0      0      1      1
3  3   1   1  NA   1.0      1      1      0      0
4  4   2   1   2 100.0      1      0      0      0
5  5   6   1   1   2.0      0      1      1      1
6  6   6   2   1   0.1      0      1      0      0
```

`str()` 関数で、R に読み込んだデータの構成を見てみましょう。

```
> str(d01r)
'data.frame':        35 obs. of  9 variables:
 $ ID    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Q01   : int  2 6 1 2 6 6 6 2 2 5 ...
 $ Q02   : int  1 1 1 1 1 2 2 1 2 2 ...
 $ Q03   : int  2 2 NA 2 1 1 1 2 1 1 ...
 $ Q04   : num  10 50 1 100 2 0.1 1 100 0.5 1 ...
 $ Q05_01: int  0 0 1 1 0 0 0 0 1 0 ...
 $ Q05_02: int  0 0 1 0 1 1 0 0 1 1 ...
 $ Q05_03: int  1 1 0 0 1 0 1 1 1 1 ...
 $ Q05_04: int  1 1 0 0 1 0 0 0 0 1 ...
```

d01r はデータフレームで、9 個の変数に 35 の observation（観測値）があると書いてあります。9 個の変数は、いずれも integer（整数）です。

2. 次に、設問文 dset_q.csv を読み込みます。英数文字とかな漢字が混在すると、縦の並びがガタガタになる場合がありますが、気にせず読んでください。

```
> (d01q<-read.csv("dset_q.csv",as.is=T))
  qID                              question    graph order
1 Q01           あなたの実家はどこですか？ 単一回答     大
2 Q02             たこ焼き器はありますか？ 単一回答     同
3 Q03 1 万円札を拾ったら交番に届けますか？ 単一回答     同
4 Q04     いくら以上なら交番に届けますか？ 数値記入
5 Q05                       あなたの性格は？ 複数回答     大
```

引数に `as.is=T` とあるのは、「そのまま読み込んで」、つまり、level とかつけないで、という意味です。これ無しで CSV ファイルから文字列を読み込むと、因子（factor）になってしまいます。そうすると、各文字列ごとに levels の属性が加えられてしまいます。levels 属性は、文字列に序列を与えて、それが便利なときもあれば、扱いに困ることもあります。ここでは不要なので、無くしています。

演習用のサンプルデータが用意されていない場合は、以下を R のコンソールにコピペです。

```
d01q<-data.frame(qID=c("Q01", "Q02", "Q03", "Q04", "Q05"),
       question=c("あなたの実家はどこですか？",
                  "たこ焼き器はありますか？",
                  "1 万円札を拾ったら交番に届けますか？",
                  "いくら以上なら交番に届けますか？",
                  "あなたの性格は？（複数回答）"),
       graph=c("単一回答", "単一回答", "単一回答", "数値記入", "複数回答"),
       order=c("大", "同", "同", "", "大"),
       stringsAsFactors = FALSE)
```

データの構成は以下です。

```
> str(d01q)
'data.frame':        5 obs. of  4 variables:
 $ qID     : chr  "Q01" "Q02" "Q03" "Q04" ...
 $ question: chr  "あなたの実家はどこですか？" "たこ焼き器はありますか？" "1 万円札を拾ったら交番に届
 $ graph   : chr  "単一回答" "単一回答" "単一回答" "数値記入" ...
 $ order   : chr  "大" "同" "同" "" ...
```

d01q はデータフレームで、4 つの変数に 5 の観測値があります。すべては charactor（文字）です。

3. 選択肢 dset_a.csv も読み込みます。こちらは、順序づけてほしいので、そのまま読み込みます。

```
> (d01a<-read.csv("dset_a.csv"))
       Q01    Q02         Q03 Q04                    Q05
1 1. 京都府 1. 有り   1. 届ける   0       1. メニューは最後
2 2. 大阪府 2. 無し 2. 届けない   1       2. すぐに後悔する
3 3. 兵庫県                      10 3. 行き先が決められない
4 4. 滋賀県                      50           4. 押しに弱い
5 5. 奈良県                     100
6 6. その他                      NA
```

選択肢は設問によって数が違うので、選択肢が一番多い設問が行数となっています。この場合は、Q01、Q04 の選択肢が 6 つで一番多く、その他は 2〜4 個です。（Q04 は選択肢ではなく、階級の区切りですが・・・）

演習用のサンプルデータが用意されていない場合は、以下を R のコンソールに。

```
d01a<-data.frame(Q01=c("1. 京都府", "2. 大阪府", "3. 兵庫県", "4. 滋賀県",
                       "5. 奈良県", "6. その他"),
                 Q02=c("1. 有り", "2. 無し", "", "", "", ""),
                 Q03=c("1. 届ける", "2. 届けない", "", "", "", ""),
                 Q04=c(0,1,10,50,100,NA),
                 Q05=c("1. メニューは最後",
                       "2. すぐに後悔する",
                       "3. 行き先が決められない",
                       "4. 押しに弱い", "", ""))
```

データの構成は以下です。

```
> str(d01a)
'data.frame':        6 obs. of  5 variables:
 $ Q01: Factor w/ 6 levels "1. 京都府","2. 大阪府",..: 1 2 3 4 5 6
 $ Q02: Factor w/ 3 levels "","1. 有り","2. 無し": 2 3 1 1 1 1
 $ Q03: Factor w/ 3 levels "","1. 届ける",..: 2 3 1 1 1 1
 $ Q04: int  0 1 10 50 100 NA
 $ Q05: Factor w/ 5 levels "","1. メニューは最後",..: 2 3 4 5 1 1
```

d01a もデータフレームで、5 つの変数に 6 つの観測値があると書かれています。

Q01〜Q03 と Q05 の 4 つの変数はすべて因子（Factor）で、Q04 は整数（Integer）です。Q01 だと、6

因子があり、レベルは、「1. 京都府」、「2. 大阪府」、・・・の順番に設定してあり、入力されているデータをレベル番号で表すと、1, 2, 3, 4, 5, 6 だと書いてあります。
Q02 だと、3 因子があり、レベルは、空白（""）、「1. 有り」「2. 無し」の順番に設定してあり、データは、レベルの番号で、2, 3, 1, 1, 1, 1 と入力されている（つまり、後ろ 4 つは空白（""）ということ）です。
ちなみにこの空白部分は、ひとつの設問を抽出する際に取り除いて使います。

4. 設問文 d01q から設問文の ID（qID）だけ抽出します。

```
> (qID<-d01q[,"qID"])
[1] "Q01" "Q02" "Q03" "Q04" "Q05"
```

ここからは設問ごとの設定です。

2.2.3　i 番目のデータだけを抽出する

i 番目の設問の集計をします。ひとまず、i が 1 の場合についてやってみます。

1. i に 1 を代入します。

```
> i<-1
```

2. 回答データから i 番目の設問についての回答だけを抽出します。

```
> d02r<-select(d01r,contains(qID[i]))
```

select() 関数は、ライブラリ dplyr に入っている関数で、1 番目の引数にあるデータフレームから、2 番目の引数で与えた条件でデータを抽出します。
最初の 6 つのデータはこんな感じです。

```
> head(d02r)
  Q01
1   2
2   6
3   1
4   2
5   6
6   6
```

データフレームが返されています。qID[1] は Q01 なので、ここは d01r[,qID[i]] でよいはずなのですが、そうしてしまうと、この場合、1 列しかないので、ベクトルが返されてしまいます。

```
> d01r[,qID[i]]
 [1] 2 6 1 2 6 6 6 2 2 5 5 5 1 1 5 2 2 4 6 2 3 2 6 6 5 1 2 6 1 1 1 2 3 5 3
```

だから、データフレームとして使う場合は、data.frame(qID[i]=d01r[,qID[i]]) のように、もうひと手間加えなければなりません。（ggplot2 は、データを、データフレームで与えなければならないのです。）
２番目の引数に contains(qID[i]) とあるのは、変数名に qID[i]（i が 1 の場合は、"Q01"ですね）を含む変数だけを抽出してくれ、ということになります。qID[i] は"Q01"なので、select(d01r,qID[i]) でもよいのですが、複数回答の qID[5] は"Q05"で、d01r の変数名は、"Q05_01", ..., "Q05_04"の４つなので、一致しません。でも、contents("Q05") ならこの４つとも選択してくれて、大丈夫です。

ちなみに、contains() 関数は便利ですが、select 関数の中だけで使えて、単独では使えないようです。

3. 質問文とかも抽出しておきます。

```
> (d02q<-d01q[i,"question"])
[1] "あなたの実家はどこですか？"
```

4. グラフの種類を抽出します。

```
> (d02g<-d01q[i,"graph"])
[1] "単一回答"
```

5. 選択肢も抽出しておきます。

```
> (d02a_0<-d01a[,qID[i]])
[1] 1. 京都府 2. 大阪府 3. 兵庫県 4. 滋賀県 5. 奈良県 6. その他
Levels: 1. 京都府 2. 大阪府 3. 兵庫県 4. 滋賀県 5. 奈良県 6. その他
```

6. これは空白も含んでいるので、それは除きます。(Q01 は、一番選択肢の多い設問なので、ここで除かれるものはありませんが、このプロセスは他の設問で必要になります。)

```
> (d02a_0<-d02a_0[d02a_0!=""])
[1] 1. 京都府 2. 大阪府 3. 兵庫県 4. 滋賀県 5. 奈良県 6. その他
Levels: 1. 京都府 2. 大阪府 3. 兵庫県 4. 滋賀県 5. 奈良県 6. その他
```

2.2.4　選択肢番号を選択肢の文言に変換する

全体の回答データから設問 i について抽出した d02r は、2, 6, 1, ... といった選択肢番号が入力されています。これを、もともとの選択肢 d02a_0 の文字に変換してくっつけます。

1. 後の処理のために、まず、d02_d の変数名を aID にします。

```
> colnames(d02r)<-"aID"
```

最初の 6 個のデータはこんな感じ

```
> head(d02r)
  aID
1   2
2   6
3   1
4   2
5   6
6   6
```

選択肢番号が入力されていますね。単一回答なので、 1 列しかありません。

2. d02a という名前で、選択肢番号と選択肢の文言を対応させたデータフレームをつくります。
1 列目が aID という名前で選択肢番号、2 列目が xtext という名前で d02a_0 という名前で選択肢の文言です。

```
> na01<-length(d02a_0)
> (d02a<-data.frame(aID=1:na01,xtext=d02a_0))
  aID     xtext
1   1 1. 京都府
```

```
2   2 2. 大阪府
3   3 3. 兵庫県
4   4 4. 滋賀県
5   5 5. 奈良県
6   6 6. その他
```

3. d02r の aID と、d02a の aID を対応させて、d02r の 2, 6, 1, ... といった選択肢番号で入力されたデータに、対応する選択肢の文言がくっつけて、新たなデータフレーム d02 を作成します。

```
> d02<-join(d02r,d02a,by="aID")
```

最初の方のデータはこんな感じです。

```
> head(d02)
  aID    xtext
1   2 2. 大阪府
2   6 6. その他
3   1 1. 京都府
4   2 2. 大阪府
5   6 6. その他
6   6 6. その他
```

join() 関数は、ライブラリ plyr に含まれている関数で、1 番目の引数のデータフレームの中の by=で指定した変数を、2 番目の引数のデータフレームの by=で指定した変数で参照して、その値をくっつけてくれます。
d02r には、aID の列（しかありませんが）の最初に行「2」と入力されているので、d02a の aID で「2」の行の他の列の値「2. 大阪府」がくっついています。
エクセルを使っている人ならば、VLOOK 関数に対応する関数だと言えばわかるかもしれません。

2.2.5 集計して構成比を求める

ライブラリ plyr を使えば、集計は簡単です。

1. 集計します。

```
> (t02<-plyr::count(d02))
  aID    xtext freq
1   1 1. 京都府    7
2   2 2. 大阪府   10
3   3 3. 兵庫県    3
4   4 4. 滋賀県    1
5   5 5. 奈良県    6
6   6 6. その他    8
```

count() 関数は、ライブラリ plyr に含まれる関数で、データフレームにある因子（factor）変数について、因子ごとに頻度（frequency）を数えて、freq という名前で出力してくれます。
count() 関数の前に plyr::とあるのは、**ライブラリ plyr の中の count() 関数**ですということを明示するためです。じつは、同時に読み込んでいるライブラリ dplyr にも count() 関数があって、これはライブラリ plyr の count() 関数とは違う働きをします。ですから、このようにライブラリを明示しておかないと、別の計算をしてしまいます。

2. 構成比を出します。

```
> (p02<-transform(t02,prop=freq/sum(freq)))
  aID    xtext freq       prop
1   1 1. 京都府    7 0.20000000
2   2 2. 大阪府   10 0.28571429
3   3 3. 兵庫県    3 0.08571429
4   4 4. 滋賀県    1 0.02857143
5   5 5. 奈良県    6 0.17142857
6   6 6. その他    8 0.22857143
```

transfer() 関数もライブラリ plyr の中の関数で、これは、1 番目の引数にあるデータフレームについて、2 番目以降の引数に指定した計算をして、新しい変数としてくっつけてくれます。
ここでは、データフレーム t02 の変数 freq の合計をとって（sum(freq)）、それで freq の各値を割った値を prop という名前でくっつけています。
これでデータはそろいました。

2.2.6　グラフのフォーマットを決める

1. グラフのタイトルを決めます。

```
> (ttl01<-paste(d02q,"\n"," (",d02g,"N=",sum(p02$freq),") "))
[1] "あなたの実家はどこですか？ \n  (  単一回答  N= 35  ) "
```

paste() 関数は、引数の値を全てくっつけて、文字列にしてくれます。d02q や sum(p02$freq) など、R に入っているベクトルや計算式などは、出力した形でくっつけます。" "で囲ったものは、文字列として扱いそのままくっつけます。特殊な表現として"\n"というのが入っていますが、これは改行を意味します。

2. カラーパレットを作成します。

```
> cb_palette<-c("#0072B2","#F0E442","#009E73","#56B4E9",
+               "#CC79A7","#D55E00","#E69F00","#999999")
```

単一回答は 1 色しか使いませんが、あとで使いまわせるように、ここで 8 色用意しておきます。これは、色覚が弱い人でも識別しやすい色だそうです。

3. グラフのテーマをいじります。

```
> gfbars<-theme_bw()+                              #白黒基調の組み込みテーマを使う
+   theme(panel.border=element_blank(),            #描画領域の枠線を消す
+         panel.grid=element_blank(),              #目盛線を消す
+         axis.title=element_blank(),              #軸タイトルを消す
+         axis.ticks.y=element_blank(),            #Y 軸（縦軸？）の目盛を消す
+         axis.text.y=element_text(size=10),       #Y 軸（縦軸？）ラベルの文字サイズ
+         axis.line.x=element_line(colour="grey"),#軸線を grey で書き足す
+         axis.line.y=element_line(colour="grey"))
```

できるだけシンプルなグラフにします。

2.2.7　グラフを描画する：選択肢番号どおり

1.　X 軸の並びを決めます。まず、選択肢暗号どおりに並べましょう。

```
> xord<- -p02$aID
```

横棒グラフは、x 軸に指定した値が数値の場合は、値が小さい順に下から積上げていきます。X 軸に指定した値が因子（factor）の場合は、levels の番号順です。

```
> str(xord)
 int [1:6] -1 -2 -3 -4 -5 -6
```

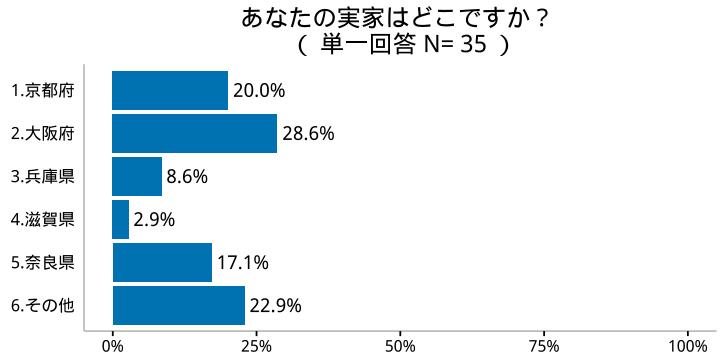
注意点として、選択肢番号の若い方が上にくるように、選択肢番号（p02$aID）にマイナスを与えています。

2. XY のデータを与えます

```
> ggbars<-ggplot(p02,                              #データを指定
+              aes(x=reorder(x=xtext,X=xord),  #X 軸は xtext を xord の順番で
+                  y=prop))+                   #Y 軸は prop のデータを
+   coord_flip()+                               #横棒グラフに
+   scale_y_continuous(labels=percent,          #Y 軸の目盛は%表記
+                      limits=c(0,1))+          #Y 軸の範囲は 0〜1
+   geom_text(                                  #データラベルを記入
+     aes(y=prop,                               #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),    #prop*100 を、数点以下 1 桁で%表記
+     size=4,                                   #ラベルの文字サイズは 4
+     hjust=-0.1)+                              #ラベルの位置調整、0.1 大きく
+   ggtitle(ttl01)+                             #グラフタイトル
+   guides(fill=FALSE)                          #凡例は非表示
```

3. グラフを表示

```
> ggbars+gfbars+geom_bar(stat="identity",fill=cb_palette[1])
```



グラフを描きます。ggbars で与えたグラフの定義に gfbars で与えた書式を足した後、geom_bar() 関数で、棒グラフを描くように指示します。棒の値（stat=）は、aes() 関数の中の y=で与えた値そのままで（"identitiy"）と指示しています。fill=cb_palette[1] は、棒の塗りつぶしを、さきほど定義したカラーパレット cb_palette の 1 番目の色を使ってくれという意味です。

2.2.8　グラフを描画する：値の大きい順

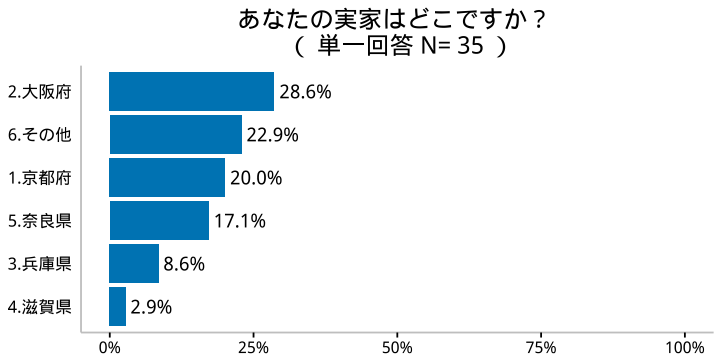
1.　X 軸の並びとして、構成比の大きさ（p02$prop）そのものを与えています。（rank() 関数などで大きさの順位を求めてもよいが、結果は同じこと）

```
> xord<-p02$prop
```

横棒グラフは、下から上へ、値が小さい順に並ぶので、最大値が一番上に表示されます。

2. グラフを表示

```
> ggbars+gfbars+geom_bar(stat="identity",fill=cb_palette[1])
```



ggbars に与えた xord は、そのまま入れ替えてくれえるようなので、ggbars を再定義しなくても、そのまま再描画できるようです。

2.2.9　グラフを描画する：「その他」を最後に

値が大きい順といっても、「その他」は寄せ集めなので、割合が大きくても最後に持ってきたほうがよいでしょう。

1. 選択肢の文（p01 の変数 xtext）が「その他」という文字を含んでいる場合、そこに一番小さい値として-1 を代入しておきます（集計値は全て 0 以上のはずだから）。

```
> xord<-p02$prop
> xord[grepl("その他",p02$xtext)]<--1
```

grepl() 関数というのは、指定した文字列を含んでいる行探し出してくれます。p02$xtext というのは、こんなんでした。

```
> p02$xtext
[1] 1. 京都府 2. 大阪府 3. 兵庫県 4. 滋賀県 5. 奈良県 6. その他
Levels: 1. 京都府 2. 大阪府 3. 兵庫県 4. 滋賀県 5. 奈良県 6. その他
```

grepl() 関数を使うと・・・

```
> grepl("その他",p02$xtext)
[1] FALSE FALSE FALSE FALSE FALSE  TRUE
```

上のコマンドは、このうち TRUE の箇所だけに-1 を代入しなさいという意味になります。

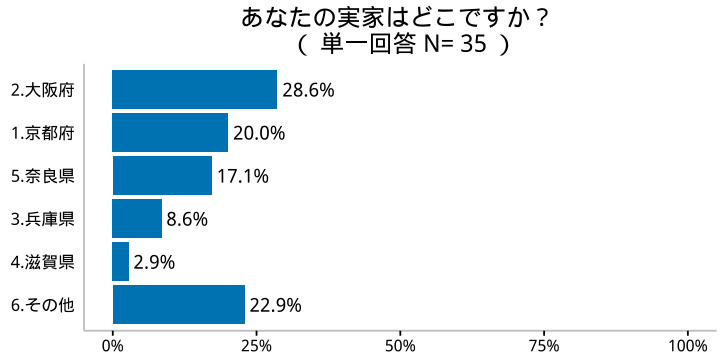
```
> xord
[1]  0.20000000  0.28571429  0.08571429  0.02857143  0.17142857 -1.00000000
```

これで、「その他」という文字を含んでいる選択肢が最後にまわされることになります。しかし、（あまりないと思いますが）通常の選択肢文に「その他」が含まれている場合も、強制的に最後にまわされて

しまうことになるので、選択肢文の表現を変えるとかして対処してください。

2. グラフを表示

```
> ggbars+gfbars+geom_bar(stat="identity",fill=cb_palette[1])
```



構成比が大きい順で、「その他」だけが、値に関わらず、一番下に配置されています。

2.2.10　グラフを描画する：任意の並べ方で

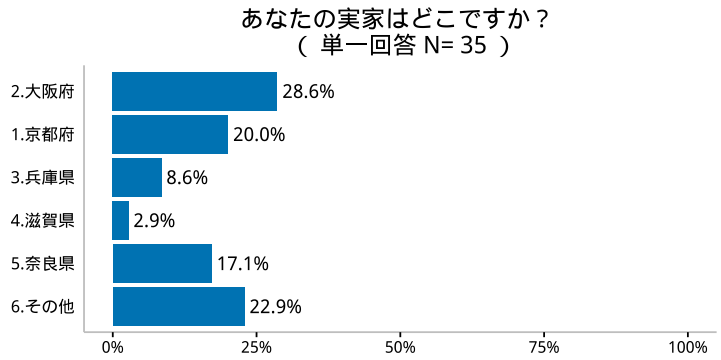
選択肢番号順でも、大きい順でもなく、任意に並べ方を与えても構いません。

1. xord に並べたい順を指定するだけです。「1. 京都府」と「2. 大阪府」を入れ替えたい場合は、以下のように指定します。ただし、下から積上げるので、マイナスをつけます。

```
> xord<--c(2,1,3,4,5,6)
```

2. グラフを表示

```
> ggbars+gfbars+geom_bar(stat="identity",fill=cb_palette[1])
```



2.2.11　指示に従って並べ方変える

以上、3 つ以上の選択肢を持つ単一回答のグラフについて、選択肢順、大きい順、任意の並べ方の 3 とおりのグラフを書いてみました。しかし、描き分けるといっても xord を変えるだけで、あとはすべて同じでした。

しかも、d01q には、その指示を与えていました。

```
> d01q
  qID                                question    graph order
1 Q01           あなたの実家はどこですか？ 単一回答      大
2 Q02             たこ焼き器はありますか？ 単一回答      同
3 Q03 1 万円札を拾ったら交番に届けますか？ 単一回答      同
4 Q04     いくら以上なら交番に届けますか？ 数値記入
5 Q05                       あなたの性格は？ 複数回答      大
```

もし、d01q の order が「同」だったら、選択肢順なので、xord に選択肢番号のマイナスを与えればよいです。

```
> (xord<--p02$aID)
[1] -1 -2 -3 -4 -5 -6
```

そうではなくて、もし、d01q の order が「大」だったら、大きい順なので、構成比をそのまま xord に与えます。

```
> (xord<-p02$prop)
[1] 0.20000000 0.28571429 0.08571429 0.02857143 0.17142857 0.22857143
```

ただし、選択肢に「その他」が含まれている場合は、それを最後に配置したいので、xord の「その他」にあたる部分に一番小さい値として-1 を与えます（構成比 p02$prop は必ずプラスなので）。

```
> xord<-p02$prop
> (xord[grepl("その他",p02$xtext)]<--1)
[1] -1
```

いずれでもなくて、任意に順序を与えたかったら、あらかじめ xord01 という値を用意しておいて、そのマイナス値を与えます。

```
> (xord<--c(2,1,3,4,5,6))
[1] -2 -1 -3 -4 -5 -6
```

以上を、条件分岐でやってみましょう。R で条件分岐は `if` と `else` を使います。もし、`x>=0` だったらその平方根 `sqrt(x)` を計算し、そうでなかったら、「値がマイナスです！！！」と表示しなさい、という命令は以下のように表します。

まず、x が正だったら、普通に平方根をとります。

```
> x<-2
> if(x>=0){
+   sqrt(x)
+ }else{
+   print("値がマイナスです！！！")
+ }
[1] 1.414214
```

x がマイナスだったら、出力が変わります。

```
> x<--2
> if(x>=0){
+   sqrt(x)
+ }else{
+   print("値がマイナスです！！！")
+ }
[1] "値がマイナスです！！！"
```

if,else の中に if,else を使って、ややこしい条件分岐をすることもできます。

xord の書き分けをやってみましょう。まず、d01q から、i 番目のものだけ取り出します。

```
> d02_ord<-d01q[i,"order"]
```

ちょっと複雑ですが、3 つのケースを振り分けています。

```
> if(d02_ord=="同"){
+   xord<--p02$aID
+ }else if(d02_ord=="大"){
+   xord<-p02$prop
+   xord[grepl("その他",p02$xtext)]<--1
+ }else{
+   xord<--xord01
+ }
```

3 行目で、}else if(... とあるのは、本来なら、}else{if(... としなければならないところですが、このように簡略化することもできます。

ちなみに、i=1 の場合、

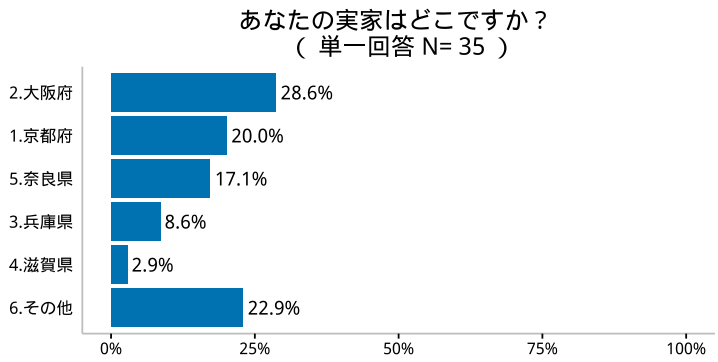
```
> d02_ord
[1] "大"
```

なので、

```
> xord
[1]  0.20000000  0.28571429  0.08571429  0.02857143  0.17142857 -1.00000000
```

です。ですから、グラフを描くと、以下のようになります。

```
> ggbars+gfbars+geom_bar(stat="identity",fill=cb_palette[1])
```



2.2.12　一旦、手順のまとめ

ちょっと、ごちゃごちゃしたので、最初から手順をもう一回まとめておきます。

ひとまず、データとは関係ないグラフのフォーマット関係の定義

```
> cb_palette<-c("#0072B2","#F0E442","#009E73","#56B4E9",
+               "#CC79A7","#D55E00","#E69F00","#999999") #カラーパレット
> #グラフのフォーマット
> gfbars<-theme_bw()+                                  #白黒基調の組み込みテーマ
```

```
+   theme(panel.border=element_blank(),          #描画領域の枠線を消す
+         panel.grid=element_blank(),            #目盛線を消す
+         axis.title=element_blank(),            #軸タイトルを消す
+         axis.ticks.y=element_blank(),          #Y 軸目盛を消す
+         axis.text.y=element_text(size=10),     #Y 軸ラベルの文字サイズ
+         axis.line.x=element_line(colour="grey"), #軸線を grey で書き足す
+         axis.line.y=element_line(colour="grey"))
```

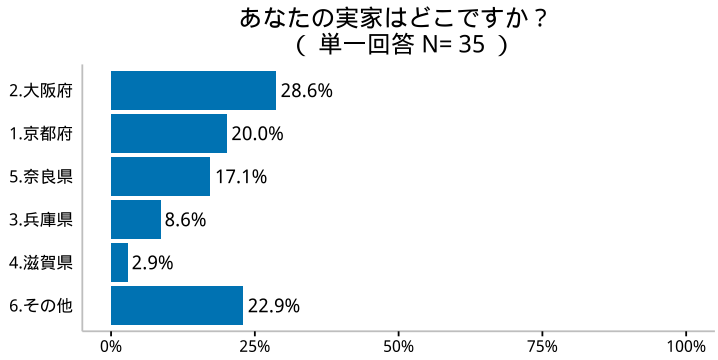
次に、i=1 についてグラフを作成します。まず i に 1 を代入。

```
> i<-1
```

さらに、以下を実行。

```
> #データ抽出
> d02r<-select(d01r,contains(qID[i]))     #i 番目の回答データ
> ttl01<-paste(d02q,"\n"," (",d02g,"N=",sum(p02$freq),") ")
> d02q<-d01q[i,"question"]                #i 番目の設問文
> d02g<-d01q[i,"graph"]                   #i 番目のグラフの種類
> d02_ord<-d01q[i,"order"]                #i 番目の x の並べ方
> d02a_0<-d01a[,qID[i]]                   #i 番目の選択肢
> d02a_0<-d02a_0[d02a_0!=""]              #選択肢から空白を削除
> na01<-length(d02a_0)                    #選択肢の数をカウント
> #xy を与えるデータを作成
> colnames(d02r)<-"aID"
> d02a<-data.frame(aID=1:na01,xtext=d02a_0)
> d02<-join(d02r,d02a,by="aID")            #選択肢番号を文に変換
> t02<-plyr::count(d02)                    #集計
> p02<-transform(t02,prop=freq/sum(freq)) #構成比を求める
> #X 軸の並べ方を指定
> if(d02_ord=="同"){
+   xord<--p02$aID
+ }else if(d02_ord=="大"){
+   xord<-p02$prop
+   xord[grepl("その他",p02$xtext)]<--1
+ }else{
+   xord<--xord01
+ }
> #グラフの定義
> ttl01<-paste(d02q,"\n"," (",d02g,"N=",sum(p02$freq),") ")#グラフタイトル
> ggbars<-ggplot(p02,                          #データを指定
+              aes(x=reorder(x=xtext,X=xord),#X 軸は xtext を xord の順番で
+                  y=prop))+                  #Y 軸は prop のデータを
+   coord_flip()+                             #横棒グラフに
+   scale_y_continuous(labels=percent,        #Y 軸の目盛は%表記
+                      limits=c(0,1))+        #Y 軸の範囲は 0〜1
+   geom_text(                                #データラベルを記入
+     aes(y=prop,                             #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)), #prop*100 を、数点以下 1 桁で%表記
+     size=4,                                 #ラベルの文字サイズは 4
+     hjust=-0.1)+                            #ラベルの位置調整、0.1 大きく
+   ggtitle(ttl01)+                           #グラフタイトル
+   guides(fill=FALSE)                        #凡例は非表示
> #グラフを表示

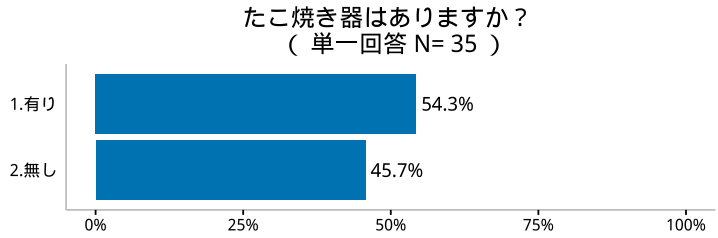
> ggbars+gfbars+geom_bar(stat="identity",fill=cb_palette[1])
```



i=2 について描画する場合は、i<-2 として、もう一度、上記の #データ抽出 以下を実行すればよいです。

```
> i<-2
```

＜続いて、上記の #データ抽出 以下を R のコンソールにコピペしてください。＞



2.2.13　関数として定義する

i<-1 としたり、i<-2 とするだけで、グラフを描き分けられるのはありがたいですね。でも、それ以下の長々としたコマンドを毎回読み込むのも面倒です。R は、繰り返して使うコマンドのかたまりを関数として定義することができます。

たとえば、$x = (1, 2, 3, 4, 5)$ というベクトルの 5 つの要素に、$ax^2 + bx + c$ という計算をさせたいときは、次のような関数を定義してあげます。

```
> f2ji<-function(x,a,b,c){
+   a*x^2+b*x+c
+ }
```

そうすると次のように計算できます。a, b, c にはそれぞれ 1, 2, 3 を設定しましょう。

```
> x01<-1:5
> f2ji(x=x01,a=1,b=2,c=3)
[1]  6 11 18 27 38
```

関数の定義で、a=1,b=2,c=3 として、デフォルトとして値を与えることもできます。

```
> f2ji<-function(x,a=1,b=2,c=3){
+   a*x^2+b*x+c
+ }
```

このように指定しておくと、f2ji(x) のように、a, b, c に何も指定しなかったら、その値が使われます。

```
> f2ji(x=x01)
[1]  6 11 18 27 38
```

指定したものについては指定した値が使われます。

```
> f2ji(x=x01,a=0)
[1]  5  7  9 11 13
```

x には x01 の値を使い a は 0、b，c はデフォルトの値 2，3 を使っています。

x にはデフォルトの値がないので、f2ji(a=0) というように、これを指定しなかったらエラーになります。

以上がわかれば、関数は定義できるはずです。・・・ということで、単純集計のグラフを描く関数を定義してみましょう。いくら長くても、原理は同じです。

3 つの原データの読み込みと i を引数に要求する関数として fgsingle_bar という名前を付けました。関数の中身は、ほぼすべて、前述したものをそのまま使っています。

変更点は、d01、d01q、d01a、i を引数の意味がわかりやすいように、dr、、dq、da、qNo に変えました。

また、gfbars の表記のところを theme に変えました。デフォルトは gfbars ですが、グラフのテーマをそれ以外にも指定できます。カラーパレットも cb_palette 以外も使えるように、これをデフォルトとして palette と変更してあります。

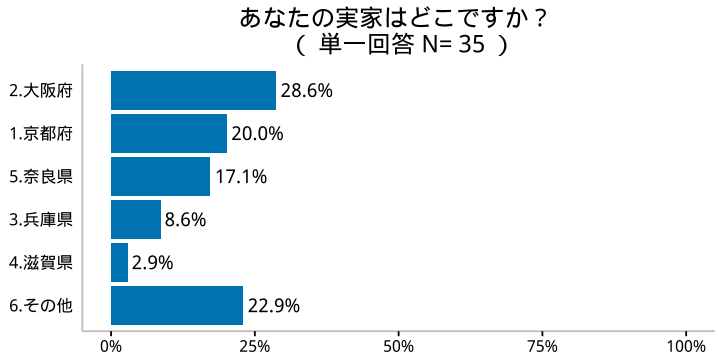
xord01=NULL とあるのは、x の並べ方を外から与える場合（d01q の order が「任」の場合、選択肢に並べる順位を指定できるようにしています。「任」以外の場合、これは使わないので、デフォルトでは NULL として、何も入れていません。

```
> fgsingle_bar<-function(dr,dq,da,qNo,
+                        xord01=NULL,theme=gfbars,palette=cb_palette[1]){
+ #データ抽出
+ d02r<-select(dr,contains(qID[qNo]))  #i 番目の回答データ
+ d02q<-dq[qNo,"question"]             #i 番目の設問文
+ d02g<-dq[qNo,"graph"]                #i 番目のグラフの種類
+ d02_ord<-dq[qNo,"order"]             #i 番目の x の並べ方
+ d02a_0<-da[,qID[qNo]]                #i 番目の選択肢
+ d02a_0<-d02a_0[d02a_0!=""]           #選択肢から空白を削除
+ na01<-length(d02a_0)                 #選択肢の数をカウント
+
+ #xy を与えるデータを作成
+ colnames(d02r)<-"aID"
+ d02a<-data.frame(aID=1:na01,xtext=d02a_0)
+ d02<-join(d02r,d02a,by="aID")            #選択肢番号を文に変換
+ t02<-plyr::count(d02)                    #集計
+ p02<-transform(t02,prop=freq/sum(freq)) #構成比を求める
+
+ #X 軸の並べ方を指定
+ if(d02_ord=="同"){
+   xord<--p02$aID
+ }else if(d02_ord=="大"){
+   xord<-p02$prop
+   xord[grepl("その他",p02$xtext)]<--1
+ }else{
+   xord<--xord01
+ }
+
+ #グラフの定義
```

```
+ ttl01<-paste(d02q,"\n"," (",d02g,"N=",sum(p02$freq),") ")#グラフタイトル
+ ggbars<-ggplot(p02,                              #データを指定
+              aes(x=reorder(x=xtext,X=xord),#X 軸は xtext を xord の順番で
+                  y=prop))+                      #Y 軸は prop のデータを
+   coord_flip()+                                 #横棒グラフに
+   scale_y_continuous(labels=percent,            #Y 軸の目盛は%表記
+                      limits=c(0,1))+            #Y 軸の範囲は 0〜1
+   geom_text(                                    #データラベルを記入
+     aes(y=prop,                                 #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)), #prop*100 を、数点以下 1 桁で%表記
+     size=4,                                     #ラベルの文字サイズは 4
+     hjust=-0.1)+                                #ラベルの位置調整、0.1 大きく
+   ggtitle(ttl01)+                               #グラフタイトル
+   guides(fill=FALSE)                            #凡例は非表示
+
+ #グラフを表示
+ ggbars+theme+geom_bar(stat="identity",fill=palette)
+ }
```

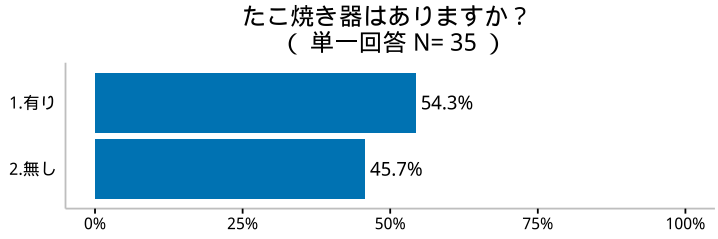
ここまで定義したら、あとは簡単です。i=1 の場合だと以下です。

```
> fgsingle_bar(dr=d01r,dq=d01q,da=d01a,qNo=1)
```



引数の順番が同じだったら、dr=といった表記は省略できます。i=2 の場合だと以下です。

```
> fgsingle_bar(d01r,d01q,d01a,2)
```



2.3　単一回答の帯グラフ

前節で、単一回答を横棒グラフにとって回答の分布をみました。選択肢が多いと、これを円グラフや帯グラフにすると、いまひとつ大きさがわかりにくくなるので、これはこれでよいと思うのですが、「はい」「いいえ」

とかの 2 つしかない場合は、どちらかの回答に関心がある場合が多く 2 つの棒は不要です。思い切って、帯グラフにした方が見やすいと思います。やってみましょう。

2.3.1　データの読み込み

i=2（たこ焼き器を持っているか？）についてやってみましょう。
データを抽出して、集計して、構成比を求めるところまでやります。ここの手順は、前節と全く同じです。

```
> i<-2
> #データ抽出
> d02r<-select(d01r,contains(qID[i]))     #i 番目の回答データ
> d02q<-d01q[i,"question"]                #i 番目の設問文
> d02g<-d01q[i,"graph"]                   #i 番目のグラフの種類
> d02_ord<-d01q[i,"order"]                #i 番目の x の並べ方
> d02a_0<-d01a[,qID[i]]                   #i 番目の選択肢
> d02a_0<-d02a_0[d02a_0!=""]              #選択肢から空白を削除
> na01<-length(d02a_0)                    #選択肢の数をカウント
> #xy を与えるデータを作成
> colnames(d02r)<-"aID"
> d02a<-data.frame(aID=1:na01,xtext=d02a_0)
> d02<-join(d02r,d02a,by="aID")           #選択肢番号を文に変換
> t02<-plyr::count(d02)                   #集計
> p02<-transform(t02,prop=freq/sum(freq))#構成比を求める
```

計算した p02 を表示してみます。

```
> p02
  aID  xtext freq      prop
1   1 1. 有り   19 0.5428571
2   2 2. 無し   16 0.4571429
```

aID が選択肢番号、xtext が選択肢の文言、freq が回答数、prop が構成比です。

2.3.2　データラベルの位置を計算する

帯グラフの各棒の真ん中に、データラベルを張り付ける準備として、その位置を計算します。棒グラフは、データラベルを棒の右側に表示するので、棒の値そのものがデータラベルの位置のデータとして使えました。しかし、帯グラフの場合、各棒の真ん中に表示しなければなりません。

帯グラフの各棒の真ん中の位置を計算するには、各棒の累和を計算して、そこから自分の値の半分を引くという計算になります。たとえば、10, 20, 30 という帯グラフの場合、$10-10/2$, $10+20-20/2$, $10+20+30-30/2$ が各棒の真ん中になります。

この値を lposition と名前をつけて p02 にくっつけます。これは、transform() 関数でできます。以下のコマンドは、p02 の各データについて、2 番目の引数の計算をして、lposition と名前でくっつけよ、という意味になります。

```
> (p03<-transform(p02,lposition=cumsum(prop)-prop/2))
  aID  xtext freq      prop lposition
1   1 1. 有り   19 0.5428571 0.2714286
2   2 2. 無し   16 0.4571429 0.7714286
```

p02 に、lposition という変数が新たに加わっています。

2.3.3　グラフのテーマを決める

グラフを描く前に、データを与えなくても指定できるような属性をあらかじめ定義しておきます。前節とほぼ同じですが、今回は凡例を表示するので、legend.position="top"で、その位置が指定されています。また、100 %のスケールはわかっているので、X 軸と Y 軸を消してみました。

```
> gfobis<-theme_bw()+                          #白黒基調の組み込みテーマを使う
+   theme(panel.border=element_blank(),        #描画領域の枠線を消す
+         panel.grid.major=element_blank(),    #目盛線を消す
+         axis.ticks=element_blank(),          #目盛を消す
+         axis.text.x=element_blank(),         #目盛ラベルを消す
+         axis.title=element_blank(),          #軸タイトルを消す
+         legend.position="top")               #凡例を上に配置する
```

2.3.4　グラフを定義する

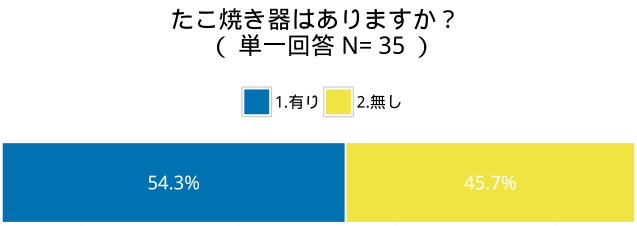
グラフを描く前に、データを与えて定義するような属性を定義しておきます。前章で描いた帯グラフと同じですが、1 本だけの帯グラフにしたいので、x=""（X 軸で表示するのは 1 個だけの要素で、名前は空白）としています。

```
> #グラフの定義
> ttl01<-paste(d02q,"\n"," (",d02g,"N=",sum(p02$freq),") ") #グラフタイトル
> ggobis<-ggplot(p03,                            #データを指定
+              aes(x="",                         #X 軸は 1 個だけなので空白
+                  y=prop,                       #Y 軸は prop のデータを
+                  fill=xtext),                  #棒を xtext ごとに塗り分ける
+              position=fill)+                   #100% 積み上げグラフに
+   coord_flip()+                                #横棒グラフに
+   scale_fill_manual(values=cb_palette)+        #棒の色
+   ggtitle(ttl01)+                              #グラフタイトル
+   guides(fill=guide_legend(title=NULL))        #凡例のタイトルを消す
```

2.3.5　グラフを描く

以上の定義を合わせて、geom_bar() 関数でグラフを描画します。データラベルは、棒の中に書くので、最後に書き込みます。でないと、棒の塗りつぶしで上書きされてしまい、データラベルが隠れてしまいます。

```
>   ggobis+gfobis+geom_bar(stat="identity",colour="white")+
+     geom_text(                                 #データラベルを記入
+       aes(y=lposition,                         #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),#prop*100 を数点以下 1 桁で%表記
+       size=4,                                  #ラベルの文字サイズは 4
+       colour="white")                          #ラベル文字の色
```



こんな感じです。どうですか？

2.3.6　関数として定義する

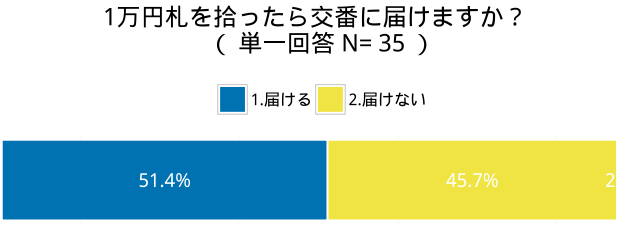
関数化しましょう。関数の名前は `fgsingle\_obi` にしてみました。関数の中で使う変数名を若干変更します。前節同様、回答番号のデータ d01r は `dr`、設問文などの d01q は `dq`、選択肢 d01a は `da`、設問番号 i は `qNo` に変えてあります。gfobis は `theme` と変えて、デフォルトを gfobis としています。cb_palette も `palette` に変えてデフォルトを cb_palette としています。

```
> fgsingle_obi<-function(dr,dq,da,qNo,
+                        theme=gfobis,palette=cb_palette){
+ #データ抽出
+ d02r<-select(dr,contains(qID[qNo])) #i 番目の回答データ
+ d02q<-dq[qNo,"question"]            #i 番目の設問文
+ d02g<-dq[qNo,"graph"]               #i 番目のグラフの種類
+ d02_ord<-dq[qNo,"order"]            #i 番目の x の並べ方
+ d02a_0<-da[,qID[qNo]]               #i 番目の選択肢
+ d02a_0<-d02a_0[d02a_0!=""]          #選択肢から空白を削除
+ na01<-length(d02a_0)                #選択肢の数をカウント
+
+ #xy を与えるデータを作成
+ colnames(d02r)<-"aID"
+ d02a<-data.frame(aID=1:na01,xtext=d02a_0)
+ d02<-join(d02r,d02a,by="aID")           #選択肢番号を文に変換
+ t02<-plyr::count(d02)                   #集計
+ p02<-transform(t02,prop=freq/sum(freq)) #構成比を求める
+
+
+ p03<-transform(p02,lposition=cumsum(prop)-prop/2) #データラベルの位置
+
+ #グラフの定義
+ ttl01<-paste(d02q,"\n"," (",d02g,"N=",sum(p02$freq),") ") #グラフタイトル
+ ggobis<-ggplot(p03,                          #データを指定
+              aes(x="",                       #X 軸は 1 個だけなので空白
+                  y=prop,                     #Y 軸は prop のデータを
+                  fill=xtext),                #棒を xtext ごとに塗り分ける
+              position=fill)+                 #100% 積み上げグラフに
+   coord_flip()+                              #横棒グラフに
+   scale_fill_manual(values=palette)+         #棒の色
+   ggtitle(ttl01)+                            #グラフタイトル
+   guides(fill=guide_legend(title=NULL))      #凡例のタイトルを消す
+
+   ggobis+theme+geom_bar(stat="identity",colour="white")+
+     geom_text(                               #データラベルを記入
+       aes(y=lposition,                       #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),#prop*100 を数点以下 1 桁で%表記
```

```
+       size=4,                          #ラベルの文字サイズは 4
+       colour="white")                  #ラベル文字の色
+ }
```

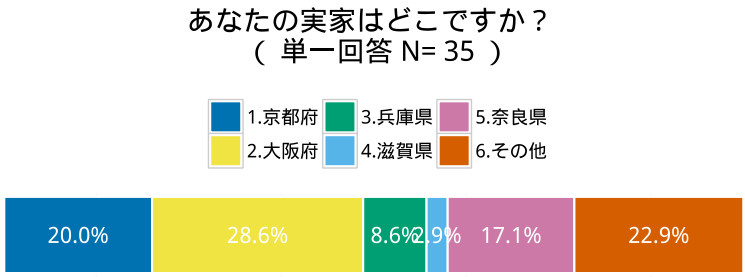
定義した関数 fgsingle_obi() を使ってみます。デフォルトで与えたテーマ theme=gfobis、palette=cb_palette については、そのまま使うので引数は省略しています。

```
> fgsingle_obi(dr=d01r,dq=d01q,da=d01a,qNo=3)
```



この関数は、選択肢が 2 つの場合のグラフとして定義しましたが、i=1 のように選択肢が 6 個ある場合でも、問題なく描けます。分布がわかりにくいというだけです。

```
> fgsingle_obi(d01r,d01q,d01a,1)
```



2.4　複数回答のグラフ

複数回答の場合は、各設問についての指摘率を棒グラフにします。5 番目の設問（性格についての設問で、複数回答）を例にやてみましょう。

まずは、i に 5 を代入しましょう。

```
> i<-5
```

データを抽出する

複数回答は、設問は 1 つですが、選択肢ごとに 0, 1 のデータが入力されています。

設問の ID を確認してみます。Q05 のはずです。

```
> qID[i]
[1] "Q05"
```

回答データは、選択肢ごとに Q04_01, Q04_02, Q04_03, Q04_04 の 4 つの変数に分けて入力されています。

それらをすべて選んで、d02r に入れます。

```
> d02r<-select(d01r,contains(qID[i]))  #i 番目の回答データ
```

ライブラリ dplyr に含まれる select() 関数の中で、このように contains() 関数を使うと、qID[i]（この場合、Q05）という文字列を含む変数をすべて選んで、複数列のデータフレームを出力してくれます。

d02r 最初の 6 つだけ表示すると、以下のような感じです。

```
> head(d02r)
  Q05_01 Q05_02 Q05_03 Q05_04
1      0      0      1      1
2      0      0      1      1
3      1      1      0      0
4      1      0      0      0
5      0      1      1      1
6      0      1      0      0
```

2.4.1　設問文や選択肢データの読み込み

次に、設問文や選択肢を読み込みます。これについては単一回答の時と全く同じです。

```
> d02q<-d01q[i,"question"]          #i 番目の設問文
> d02g<-d01q[i,"graph"]             #i 番目のグラフの種類
> d02_ord<-d01q[i,"order"]          #i 番目の x の並べ方
> d02a_0<-d01a[,qID[i]]             #i 番目の選択肢
> d02a_0<-d02a_0[d02a_0!=""]        #選択肢から空白を削除
> na01<-length(d02a_0)              #選択肢の数をカウント
```

2.4.2　各選択肢の指摘率を求める

選択肢ごとに何%の人がそれを選択したか、指摘率を求めます。d02r は、選択肢ごとに 0, 1 のデータが入っていました。それぞれの指摘率は、各選択肢の選択数を回答数で割れば出ます。

まず、各選択肢の選択数を求めます。選択された回答には 1 が入力されていて、選択されていない回答には 0 が入っていますので、選択肢ごとに合計すれば、各選択肢の選択数がわかります。

これをやるには、ライブラリ dplyr に入っている summarise_each() 関数を使うと便利です。データフレームの列ごとに、指定した計算をしてくれます。最初の引数にデータフレーム d02r、次の引数に funs(sum) とすると、各列の合計をとります。

```
> (d02_p1<-summarise_each(d02r,funs(sum)))
  Q05_01 Q05_02 Q05_03 Q05_04
1     13     15     22     16
```

次に、各列の回答数を数えます。回答が 0 か 1 かに関わらず、データが何個あるかを数えます。summarise_each() 関数の中で、length() 関数を使います。

```
> (d02_p0<-summarise_each(d02r,funs(length)))
  Q05_01 Q05_02 Q05_03 Q05_04
1     35     35     35     35
```

回答数は 35 だとわかっているので、わざわざこんなことする必要はないと思うかもしれませんが、回答に無回答が含まれる場合、この処理が必要になります。無回答の処理については後で説明します。

指摘率を求めましょう。

```
> (prop_1<-d02_p1/d02_p0)
     Q05_01    Q05_02    Q05_03    Q05_04
1 0.3714286 0.4285714 0.6285714 0.4571429
```

選択肢番号、選択肢の文、選択数、それとこの指摘率の値で、データフレームを作成します。選択数 d02_p1 と指摘率 prop_1 は横長なので、`t()` 関数を使って転地します。

```
> (p02<-data.frame(aID=1:na01,
+                  xtext=d02a_0,
+                  freq=t(d02_p1),
+                  prop=t(prop_1)))
       aID                   xtext freq      prop
Q05_01   1        1. メニューは最後   13 0.3714286
Q05_02   2        2. すぐに後悔する   15 0.4285714
Q05_03   3 3. 行き先が決められない   22 0.6285714
Q05_04   4            4. 押しに弱い   16 0.4571429
```

ここで作成した p02 は、単一回答の棒グラフを書くときに使ったデータと全く同じ形式なので、以前定義したコマンドがそのまま使えます。（なお、`freq` の値は、グラフを描くのには不要なのですが、同じにするためにこれも一応入れておきました。）

2.4.3　X 軸の並べ方を指定する

以前定義したのと全く同じです。

```
> if(d02_ord=="同"){
+   xord<--p02$aID
+ }else if(d02_ord=="大"){
+   xord<-p02$prop
+   xord[grepl("その他",p02$xtext)]<--1
+ }else{
+   xord<--xord01
+ }
```

グラフの並べ方 `d02_ord` は`"大"`で、この場合選択肢に「その他」はありませんね。

```
> d02_ord
[1] "大"
```

だから、xord には構成比 `p02$prop` がそのまま入ります。

```
> xord
[1] 0.3714286 0.4285714 0.6285714 0.4571429
```

2.4.4　グラフの定義

前節で定義したグラフの書式 gfbars はそのまま使えます。その他のグラフの定義 ggbars もそのままですが、グラフのタイトルに使っている N=35 といったときの数字の拾い方が単一回答の場合と違うので、そこだけ変えてあります。単一回答の場合は、各選択肢の回答 `p02$freq` を全て足せばよかったですが、複数回答の場合は、そうすると回答者の数を超えてしますので、選択肢ごとにデータがいくつあるかを数えた d02_p0 の

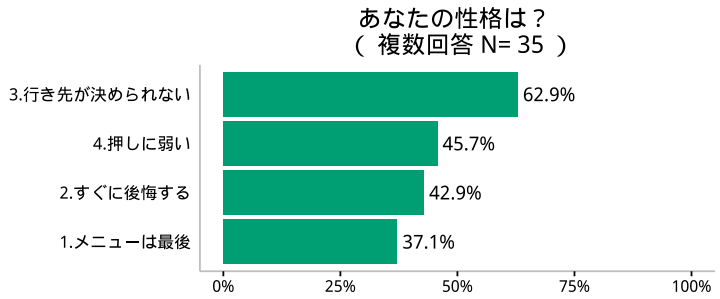
ひとつの値を使います。

```
> ttl01<-paste(d02q,"\n"," (",d02g,"N=",d02_p0[1],") ")  #グラフタイトル
> ggbars<-ggplot(p02,                                    #データを指定
+              aes(x=reorder(x=xtext,X=xord), #X 軸は xtext を xord の順番で
+                  y=prop))+                  #Y 軸は prop のデータを
+   coord_flip()+                             #横棒グラフに
+   scale_y_continuous(labels=percent,        #Y 軸の目盛は%表記
+                      limits=c(0,1))+        #Y 軸の範囲は 0〜1
+   geom_text(                                #データラベルを記入
+     aes(y=prop,                             #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),  #prop*100 を、数点以下 1 桁で%表記
+     size=4,                                 #ラベルの文字サイズは 4
+     hjust=-0.1)+                            #ラベルの位置調整、0.1 大きく
+   ggtitle(ttl01)+                           #グラフタイトル
+   guides(fill=FALSE)                        #凡例は非表示
```

2.4.5　グラフを表示

グラフを表示しましょう。色だけ少し変えてみました。

```
> ggbars+gfbars+geom_bar(stat="identity",fill=cb_palette[3])
```



2.4.6　関数として定義する

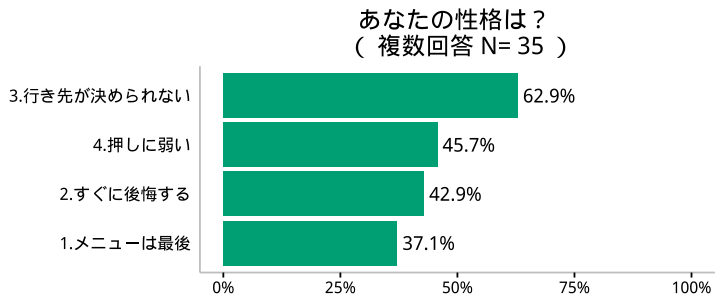
関数として定義します。要領は、これまでと同じです。

```
> fgmulti<-function(dr,dq,da,qNo,xord01=NULL,
+                   theme=gfbars,palette=cb_palette[3]){
+   #データの読み込み
+   d02r<-select(dr,contains(qID[qNo]))   #i 番目の回答データ
+   d02q<-dq[i,"question"]                #i 番目の設問文
+   d02g<-dq[i,"graph"]                   #i 番目のグラフの種類
+   d02_ord<-dq[i,"order"]                #i 番目の x の並べ方
+   d02a_0<-da[,qID[i]]                   #i 番目の選択肢
+   d02a_0<-d02a_0[d02a_0!=""]            #選択肢から空白を削除
+   na01<-length(d02a_0)                  #選択肢の数をカウント
+
+   #各選択肢の指摘率
+   d02_p1<-summarise_each(d02r,funs(sum))
+   d02_p0<-summarise_each(d02r,funs(length))
+   prop_1<-d02_p1/d02_p0
```

```
+   p02<-data.frame(x=1:na01,
+                   xtext=d02a_0,
+                   freq=t(d02_p1),
+                   prop=t(prop_1))
+
+   #X 軸の並べ方を指定する
+   if(d02_ord=="同"){
+     xord<--p02$aID
+   }else if(d02_ord=="大"){
+     xord<-p02$prop
+     xord[grepl("その他",p02$xtext)]<--1
+   }else{
+     xord<--xord01
+   }
+
+   #グラフの定義
+   ttl01<-paste(d02q,"\n"," (",d02g,"N=",d02_p0[1],") ")  #グラフタイトル
+   ggbars<-ggplot(p02,                                    #データを指定
+              aes(x=reorder(x=xtext,X=xord), #X 軸は xtext を xord の順番で
+                  y=prop))+                  #Y 軸は prop のデータを
+   coord_flip()+                             #横棒グラフに
+   scale_y_continuous(labels=percent,        #Y 軸の目盛は%表記
+                      limits=c(0,1))+        #Y 軸の範囲は 0〜1
+   geom_text(                                #データラベルを記入
+     aes(y=prop,                             #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),  #prop*100 を、数点以下 1 桁で%表記
+     size=4,                                 #ラベルの文字サイズは 4
+     hjust=-0.1)+                            #ラベルの位置調整、0.1 大きく
+   ggtitle(ttl01)+                           #グラフタイトル
+   guides(fill=FALSE)                        #凡例は非表示
+
+   #グラフを表示
+   ggbars+theme+geom_bar(stat="identity",fill=palette)
+ }
```

グラフを描いてみます。デフォルトが設定してある xord01=NULL、theme=gfbars、palette=cb_palette[3] については省略してあります。

```
> fgmulti(dr=d01r,dq=d01q,da=d01a,qNo=5)
```



2.5　数値記入項目のグラフ

最後に、数値記入項目のグラフです。アンケートで数値を直接記入させることは、値の信頼性の点から、また、回答者の負担という点から、できるだけ避けた方がよいです。しかし、致し方ない場合もあります。

サンプルデータでは 4 番目の設問で、「現金がいくら以上落ちてたら交番に届けるか？」と聞いて、数値記入を求めています。

i に 4 を代入し、データを読み込みます。選択肢がないので、回答と設問文だけでよいはずなのですが、あとで値を階級に区切る方法もやるので、その値も読み込みます。これは選択肢のところに入力されています。

2.5.1　データの読み込み

```
> i<-4
> #データ抽出
> d02r<-select(d01r,contains(qID[i]))  #i 番目の回答データ
> d02q<-d01q[i,"question"]              #i 番目の設問文
> d02g<-d01q[i,"graph"]                #i 番目のグラフの種類
> d02a<-d01a[,qID[i]]                  #i 番目の選択肢（この場合、階級の区切り）
```

d02r の最初の方だけ表示してみます。

```
> head(d02r)
    Q04
1  10.0
2  50.0
3   1.0
4 100.0
5   2.0
6   0.1
```

単位は「万円」です。みなさんそれぞれ勝手な金額を答えています（苦笑）。

d02q はこんな感じです。

```
> d02q
[1] "いくら以上なら交番に届けますか？"
```

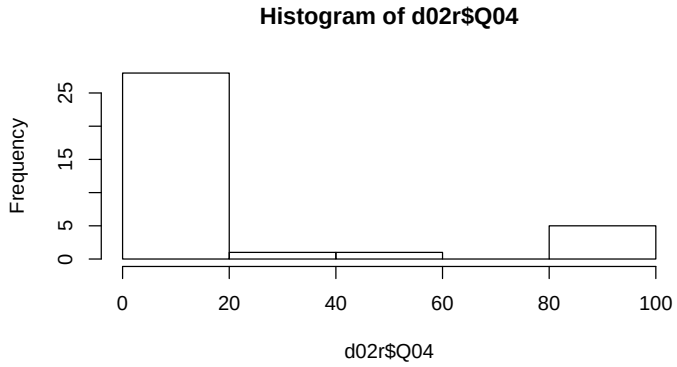
d02a はこんな感じです。ここは選択肢ではなく、データを区切る階級を指定しています。

```
> d02a
[1]   0   1  10  50 100  NA
```

2.5.2　ヒストグラムについて

数値データのグラフ化はヒストグラムとして描くのが定石です。一旦、R の旧来の方法でヒストグラムを描いてみましょう。

```
> hist(d02r$Q04)
```

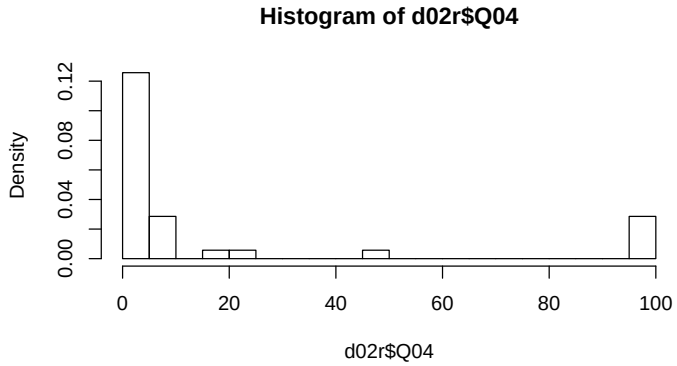


ヒストグラムは、度数分布表をグラフに表したもので、目的の数値を等間隔の階級に区分し、各階級に区分される個数を数えます。

Y 軸を、個数でなく、割合（正確には密度 density）を知りたかったら、引数に `probability=TRUE`（`prob=T` と略することも可）を付けます。

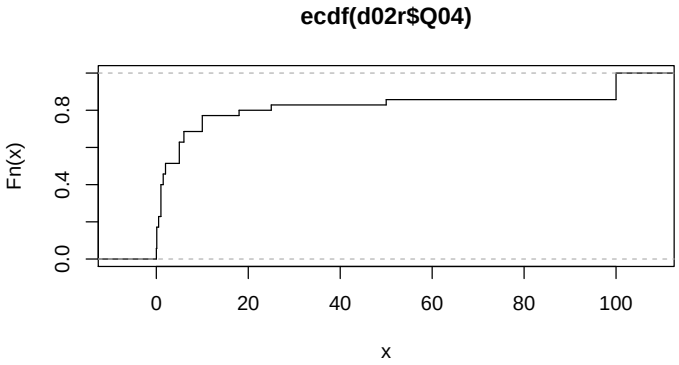
階級の幅は、`breaks=`で与えますが、デフォルトとして Sturges 法で計算されたものが設定されます。このデータについて Stureges 法で計算された区分は粗すぎるので、他の方法（FD 法）などを使います。

```
> hist(d02r$Q04,breaks="fd",probability=TRUE)
```



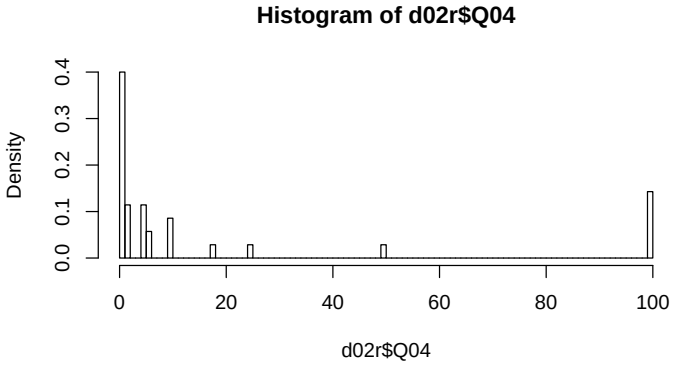
1 つの棒が 5 万円単位で描かれています。縦軸は、Density（密度）とありますが、これはその棒が全体の何%の構成比かではなく、確率分布の考え方に基づく密度分布を表しています。密度分布は、以下のような累積分布の傾きです。

```
> plot(ecdf(d02r$Q04),verticals= TRUE, do.p = FALSE)
```



縦軸を密度にとったヒストグラムは、階級の幅を 1 にとると、密度と構成比は一致します。

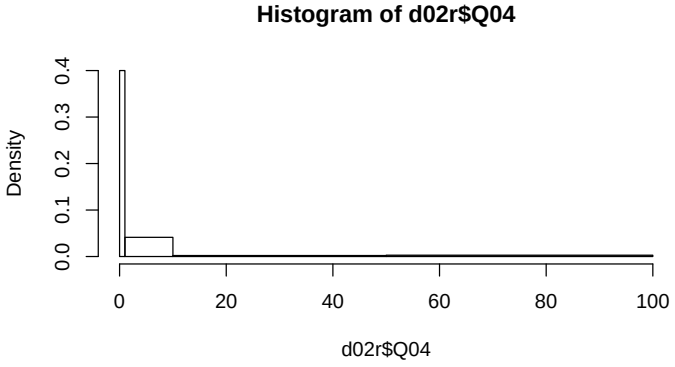
```
> hist(d02r$Q04,breaks=0:100,probability=TRUE)
```



1 万円未満に 40 %が集中していることがわかります。しかし、階級の幅が細かすぎると、分布の様子がわかりにくくなることもあります。

このような場合、グラフとしては階級を不均一に定義することもできます。階級を、d02a で指定した階級、$(0,1]$（0 円より大きく 1 万円以下）,$(1,5]$、$(5,10]$、$(10,50]$、$(50,100]$ に分けてみましょう。

```
> br01<-na.omit(d02a)
> hist(d02r$Q04,breaks=br01,probability=TRUE)
```



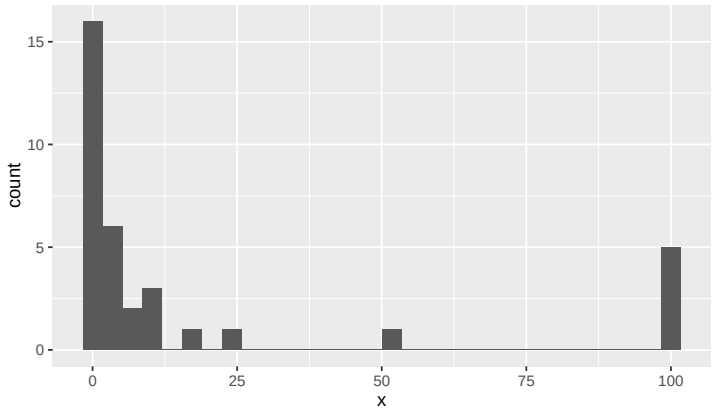
2.5.3　ggplot2 でヒストグラムを描く

ggplot2 を使って簡単なヒストグラムを描いておきましょう。
d02r の変数名は、設問番号によって違って面倒なので、x で統一しておきます。

```
> colnames(d02r)<-"x"
> head(d02r)
      x
1  10.0
2  50.0
3   1.0
4 100.0
5   2.0
6   0.1
```

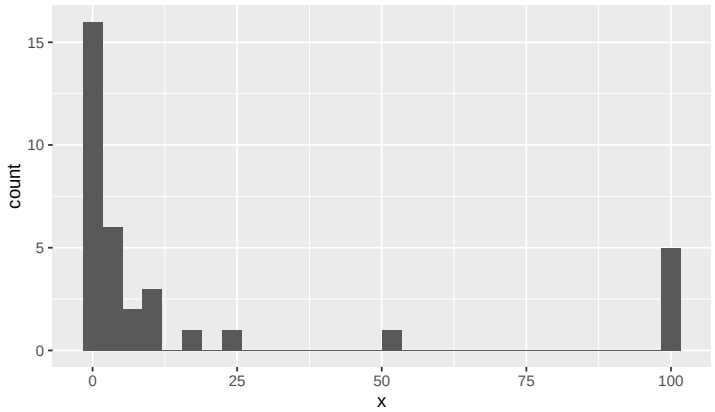
ggplot2 でヒストグラムを描く関数は geom_histgram() 関数です。

```
> ggplot(d02r,aes(x=x))+geom_histogram()
```



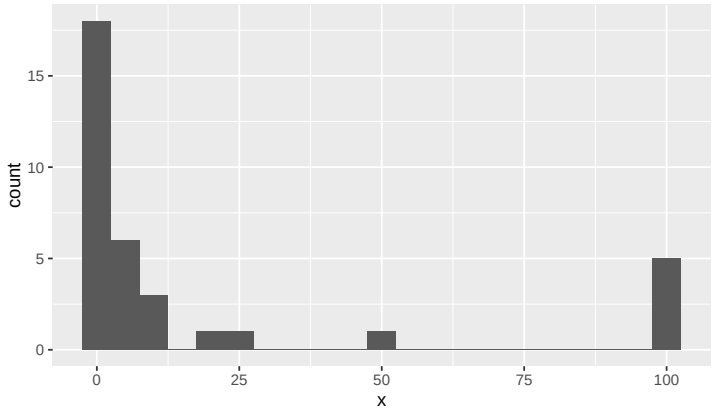
「引数 binwidth=が無い」と怒られます。これはヒストグラムの階級の幅です。デフォルトでは、メッセージにあるように、bins=30 つまり、データ範囲を 30 個に分割します。

```
> ggplot(d02r,aes(x=x))+geom_histogram(bins=30)
```



階級の幅を直接指示したい場合は、引数 binwidth=を使います。5 万円単位に区切ってみましょう。

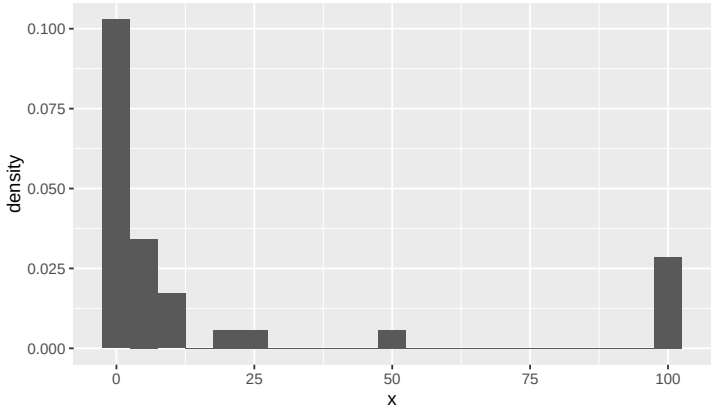
```
> ggplot(d02r,aes(x=x))+geom_histogram(binwidth=5)
```



R 旧来の `hist()` 関数で描いたのと形が少し違うようです。これは階級の右側を含むか含まないかの違いによります。例えば、0〜5 万円の階級で、`hist()` 関数は 5 万円を含みますが、ggplot2 の `geom_histogram()` 関数は 5 万円を含みません。同じにしたければ、引数に `closed="right"`を加えます・・・とこれまで書いていたのですが、ggplot2 の Update をしたら、階級の区切り方が変わったようです。たとえば、最初の棒は、以前は $[0, 5)$ でしたが、現在は、$(-2.5, 2.5]$ となっていますね。

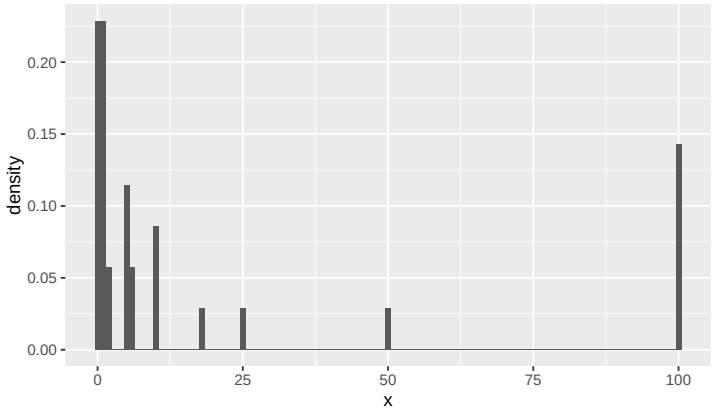
密度分布にしたい場合は、`ase()` 関数の中に `y=..density..` を加えます。

```
> ggplot(d02r,aes(x=x,y=..density..))+
+   geom_histogram(binwidth=5)
```



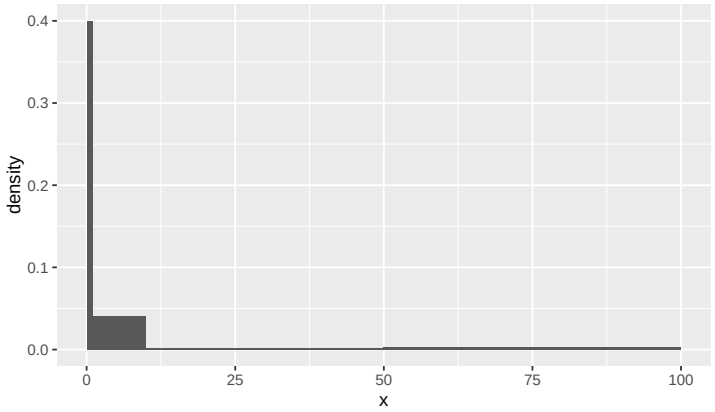
階級の幅を 1 にすると、構成比と一致します。

```
> ggplot(d02r,aes(x=x,y=..density..))+
+   geom_histogram(binwidth=1)
```



階級の幅を与えることもできます。

```
> br01<-na.omit(d02a)
> ggplot(d02r,aes(x=x,y=..density..))+
+   geom_histogram(breaks=br01)
```

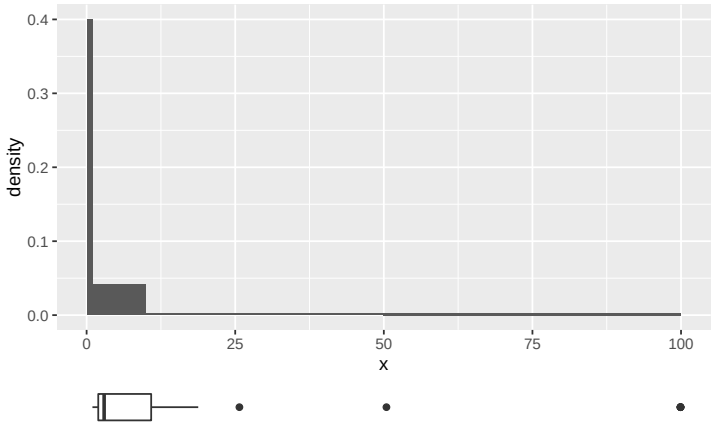


下に箱ひげ図（Boxplot）を加えることもできます。
箱ひげ図の書式を指定します。

```
> gf04<-theme_bw()+                          #白黒基調の組み込みテーマを使う
+   theme(panel.border=element_blank(),      #描画領域の枠線を消す
+         panel.grid=element_blank(),        #目盛線を消す
+         axis.ticks=element_blank(),        #目盛を消す
+         axis.text=element_text(color="white")) #目盛の文字を白色に
```

箱ひげ図を加えます。

```
> gh01<-ggplot(d02r,aes(x=x,y=..density..))+
+   geom_histogram(breaks=br01)
> gb01<-ggplot(d02r,aes(x=1,y=x))+    #データを指定,x はないので適当に 1 を
+   gf04+                             #指定したフォーマット
+   geom_boxplot()+                   #箱ひげ図の描画
+   xlab("")+ylab("")+                #軸ラベルを空白に
+   coord_flip()                      #グラフを横向きに
> grid.arrange(gh01,gb01,             #ヒストグラムと箱ひげ図を描画
+              nrow=2,ncol=1,         #描画エリアを 2 行 1 列に分割
+              heights=c(4,1))        #描画エリアの高さを 4:1 に
```



箱ひげ図の箱の下限は、全データを小さい方から数えて 25 %にあたるところ（第 1 四分位と言います）。箱の中の線が中央値（第 2 四分位）、箱の上限が 75 %のところ（第 3 四分位）です。通常は、ひげ（線）の一番小さいところが最小値で、大きいところが最大値ですが、第 1 四分位よりも箱の長さ（四分位範囲あるいは IQR と言います）の 1.5 倍小さい値がある場合は、その手前のところでひげが途切れ、あとは外れ値として点々で表示されます。同じく、第 3 四分位よりも IQR の 1.5 倍大きい値がある場合は、ひげはその手前の地点で途切れ、あとは点々で表示されます。

ggplot2 のグラフは目盛などを消していますが、目盛の文字や軸ラベルは、`element_blank()` を使わずに、`""`（空白）や `color="white"`などで見えなくしているだけです。これは、目盛の文字や軸そのものを無くすと、2 つのグラフの軸のスケールがずれてしまうからです。2 つのグラフから位置情報を抽出してスケールを合わせるという方法もあることはあるのですが、面倒なので、ここでは簡単な方法を使っています。

2.5.4　単一回答に変換してグラフ化

ヒストグラムは、統計学の確率分布の考え方になじむので、よく用いられますが、もう少し単純に、特定の階級に何%の人がいるのかを知りたい時には、R の描画は少し面倒です。そこで、回答を特定の階級に分けて、あたかも各階級を選択肢として選んでもらった形にしてしまえば、最初にやった単一回答のグラフとして処理できます。（なら、最初っからこちらでやればよかった？　まあ、そう言わんと・・・勉強、勉強）

以下に、関数化しておきましたが、手順は fgsingle_bar とほぼ同じです。違いは、選択肢を数値として読み込むということと、回答の値をその区切りで仕分けする必要があることぐらいです。

選択肢の代わりに階級の区切りが入力されています。

```
> (d02a_0<-d01a[,qID[i]])
[1]   0   1  10  50 100  NA
```

回答は、連続変数として数値で入力されています。

```
> d02r[,1]
 [1]  10.00  50.00   1.00 100.00   2.00   0.10   1.00 100.00   0.50   1.00   1.00   0.10
[13]  25.00  10.00   0.01   0.50   0.10  10.00   0.10   6.00 100.00 100.00   5.00   5.00
[25]  18.00   1.00   5.00   1.00   5.00   1.50   2.00 100.00   0.01   6.00   1.50
```

`cut()` 関数を使って、これを指定した区切りに割り振ることができます。

```
> cut(d02r[,1],d02a_0)
 [1] (1,10]   (10,50]  (0,1]    (50,100] (1,10]   (0,1]    (0,1]    (50,100] (0,1]
```

```
 [10] (0,1]    (0,1]    (0,1]    (10,50]  (1,10]   (0,1]    (0,1]    (0,1]    (1,10]
 [19] (0,1]    (1,10]   (50,100] (50,100] (1,10]   (1,10]   (10,50]  (0,1]    (1,10]
 [28] (0,1]    (1,10]   (1,10]   (1,10]   (50,100] (0,1]    (1,10]   (1,10]
Levels: (0,1] (1,10] (10,50] (50,100]
```

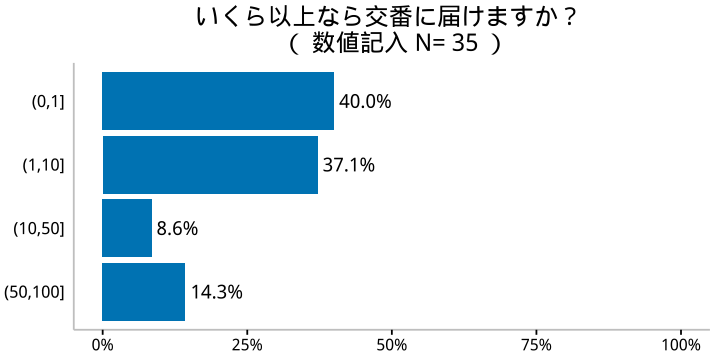
最初の値は 10.0 ですが、これには (1,10] が割り当てられています。

ここだけ気をつけて、あとは単一回答と同じ形式のデータセット（データフレーム p02）をつくるだけです。

```
> fgnumeric3<-function(dr,dq,da,qNo,
+                      theme=gfbars,palette=cb_palette[1]){
+ #データ抽出
+ d02r<-select(dr,contains(qID[qNo]))    #i 番目の回答データ
+ d02q<-dq[qNo,"question"]               #i 番目の設問文
+ d02g<-dq[qNo,"graph"]                  #i 番目のグラフの種類
+ d02a_0<-da[,qID[qNo]]                  #i 番目の選択肢（階級の区切り）
+ d02a_0<-na.omit(d02a_0)                #階級の区切りから空白を削除
+ na01<-length(d02a_0)-1                 #階級の数をカウント
+
+ #xy を与えるデータを作成
+ d02r1<-d02r[,1]                              #数値で回答されたデータ
+ d02<-data.frame(xtext=cut(d02r1,d02a_0))     #階級に区分
+ t02<-plyr::count(d02)                        #集計
+ p02<-transform(t02,prop=freq/sum(freq))      #構成比を求める
+ p02<-data.frame(aID=1:na01,p02)              #選択肢番号をつける
+
+ #X 軸の並べ方を指定
+ xord<--p02$aID
+
+ #グラフの定義
+ ttl01<-paste(d02q,"\n"," (",d02g,"N=",sum(p02$freq),") ") #グラフタイトル
+ ggbars<-ggplot(p02,                           #データを指定
+               aes(x=reorder(x=xtext,X=xord),#X 軸は xtext を xord の順番で
+                   y=prop))+                 #Y 軸は prop のデータを
+   coord_flip()+                             #横棒グラフに
+   scale_y_continuous(labels=percent,        #Y 軸の目盛は%表記
+                      limits=c(0,1))+        #Y 軸の範囲は 0〜1
+   geom_text(                                #データラベルを記入
+     aes(y=prop,                             #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)), #prop*100 を、数点以下 1 桁で%表記
+     size=4,                                 #ラベルの文字サイズは 4
+     hjust=-0.1)+                            #ラベルの位置調整、0.1 大きく
+   ggtitle(ttl01)+                           #グラフタイトル
+   guides(fill=FALSE)                        #凡例は非表示
+
+ #グラフを表示
+ ggbars+theme+geom_bar(stat="identity",fill=palette)
+ }
```

グラフを表示してみます。

```
> fgnumeric3(dr=d01r,dq=d01q,da=d01a,qNo=4)
```



と、まあ、こういう具合に、数値回答の集計はあたふたしますが、おおよその分布が予想できるようだったら、最初っからこの階級区分を選択肢として与えておけば、回答する方も回答しやすいし、集計もしやすいですね。

2.6　1 つの関数でグラフを描き分ける

2.6.1　関数の定義

これまで、アンケートの単純集計を、選択肢が 3 つ以上ある単一回答、選択肢が 2 個だけの単一回答、複数回答、それと数値記入の 4 パターンについて説明してきました。その方法は、共通する部分もあるし、違う部分もありました。これを一括して、関数として処理できないでしょうか。

やってみましょう。まず、利用するライブラリを読み込みます。

```
> library(ggplot2)
> library(plyr)
> library(scales)
> library(dplyr)
> library(gridExtra)
```

次にカラーパレットを定義します。色が気に入らなければ、ここで変えましょう。

```
> cb_palette<-c("#0072B2","#F0E442","#009E73","#56B4E9",
+               "#CC79A7","#D55E00","#E69F00","#999999")
```

グラフの x や y の値を計算する関数の定義

グラフの x や y の値を計算します。単純集計は、棒グラフを描く場合も、帯グラフを描く場合も、構成比を求めるまでは同じなので、この部分を関数として定義します。名前は、関数ということで最初に f、xy を与えるということで xy、単一回答の single をくっつけて、`fxysingle` としました。

```
> fxsingle<-function(dr=d02r,da=d02a_0,na01=na01){
+   colnames(dr)<-"aID"
+   d02a<-data.frame(aID=1:na01,xtext=da)
+   d02<-join(dr,d02a,by="aID")        #選択肢番号を文に変換
+   t02<-plyr::count(d02)               #集計
+   transform(t02,prop=freq/sum(freq)) #構成比を求める
+ }
```

複数回答は各設問の指摘率を求める部分を関数として定義します。`fxymulti` という名前にしました。

```
> fxymulti<-function(dr=d02r,da=d02a_0,na01=na01){
+   d02_p1<-summarise_each(dr,funs(sum))     #各選択肢の指摘数
+   d02_p0<-summarise_each(dr,funs(length)) #各選択肢の回答者数
+   prop_1<-d02_p1/d02_p0                    #各選択肢の指摘率
+   data.frame(x=1:na01,
+              xtext=da,
+              freq=t(d02_p1),
+              prop=t(prop_1))
+ }
```

数値記入項目は、d01a で指定した階級を読み込んだ d02a_0 で階級を作成し、これで数値を区分して、単一回答に変換します。

階級の数 na01 は階級の区切りより 1 つ少ない数です。これはわかりますか。階級の区切りが 0,10,100 の 3 つなら、階級は (0,10],(10,100] の 2 つですもんね。階級が与えられたら、cut() 関数で、数値記入項目の回答をその階級に割り振ります。あとは、単純集計と同じように集計して構成比を求めるだけです。ただし、この場合、単純集計の場合につけられている選択肢の番号（aID）がないので、データの形式を同じにするためにくっつけておきました。

```
> fxynumeric<-function(dr=d02r,da=d02a_0){
+   d02a_1<-na.omit(da)                          #階級の区分から NA を削除
+   na01<-length(d02a_1)-1                       #階級の数
+   d02r1<-dr[,1]                                #回答の値をベクトルに
+   d02<-data.frame(xtext=cut(d02r1,d02a_1))     #階級の区切りで回答の値を割り振る
+   t02<-plyr::count(d02)                        #集計
+   p02<-transform(t02,prop=freq/sum(freq))      #構成比を求める
+   data.frame(aID=1:(na01),p02)                 #単純集計のデータと同じ形式に
+ }
```

グラフのフォーマットを定義する

グラフのフォーマットは、gfbars（横棒グラフ用）と gfobis（帯グラフ用）の 2 種類しか使っていませんでした。ここは関数化する必要がないので、そのまま使います。

横棒グラフ用です。

```
> gfbars<-theme_bw()+                              #白黒基調の組み込みテーマを使う
+   theme(panel.border=element_blank(),            #描画領域の枠線を消す
+         panel.grid=element_blank(),              #目盛線を消す
+         axis.title=element_blank(),              #軸タイトルを消す
+         axis.ticks.y=element_blank(),            #Y 軸（縦軸？）の目盛を消す
+         axis.text.y=element_text(size=10),       #Y 軸（縦軸？）ラベルの文字サイズ
+         axis.line.x=element_line(colour="grey"), #軸線を grey で書き足す
+         axis.line.y=element_line(colour="grey")) #軸線を grey で書き足す
```

帯グラフ用です。

```
> gfobis<-theme_bw()+                              #白黒基調の組み込みテーマを使う
+   theme(panel.border=element_blank(),            #描画領域の枠線を消す
+         panel.grid.major=element_blank(),        #目盛線を消す
+         axis.ticks=element_blank(),              #目盛を消す
+         axis.text.x=element_blank(),             #目盛ラベルを消す
+         axis.title=element_blank(),              #軸タイトルを消す
+         legend.position="top")                   #凡例を上に配置する
```

グラフを定義して描画する関数

指定したデータに基づいて具体的にグラフを定義して描画するまでのプロセスです。これも横棒グラフと帯グラフの 2 種類だけです。

横棒グラフ用です。

```
> fgbars<-function(p02=p02,xord=xord,ttl01=ttl01,
+                  theme=gfbars,palette=cb_palette[1]){
+   #グラフの定義
+   ggbars<-ggplot(p02,                          #データを指定
+              aes(x=reorder(x=xtext,X=xord),#X 軸は xtext を xord の順番で
+                  y=prop))+                     #Y 軸は prop のデータを
+     coord_flip()+                              #横棒グラフに
+     scale_y_continuous(labels=percent,         #Y 軸の目盛は%表記
+                        limits=c(0,1))+         #Y 軸の範囲は 0〜1
+   geom_text(                                   #データラベルを記入
+     aes(y=prop,                                #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)), #prop*100 を、数点以下 1 桁で%表記
+     size=4,                                    #ラベルの文字サイズは 4
+     hjust=-0.1)+                               #ラベルの位置調整、0.1 大きく
+   ggtitle(ttl01)+                              #グラフタイトル
+   guides(fill=FALSE)                           #凡例は非表示
+   #グラフを表示
+   ggbars+theme+geom_bar(stat="identity",fill=palette)
+ }
```

帯グラフ用です。

```
> fgobis<-function(p02=p02,ttl01=ttl01,
+                  theme=gfbars,palette=cb_palette){
+   #グラフの定義
+   p03<-transform(p02,lposition=cumsum(prop)-prop/2) #データラベルの位置
+   ggobis<-ggplot(p03,                          #データを指定
+              aes(x="",                         #X 軸は 1 個だけなので空白
+                  y=prop,                       #Y 軸は prop のデータを
+                  fill=xtext),                  #棒を xtext ごとに塗り分ける
+              position=fill)+                   #100% 積み上げグラフに
+   coord_flip()+                                #横棒グラフに
+   scale_fill_manual(values=palette)+           #棒の色
+   ggtitle(ttl01)+                              #グラフタイトル
+   guides(fill=guide_legend(title=NULL))        #凡例のタイトルを消す
+   #グラフを表示
+   ggobis+theme+geom_bar(stat="identity",colour="white")+
+     geom_text(                                 #データラベルを記入
+       aes(y=lposition,                         #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),#prop*100 を数点以下 1 桁で%表記
+       size=4,                                  #ラベルの文字サイズは 4
+       colour="white")                          #ラベル文字の色
+ }
```

グラフを描き分ける

d01q には question という変数があり、ここに「単一」「単一帯」「複数」「数値」の 4 種類のグラフの種類が指定されています。その指定に従って、グラフを描き分ける関数を作成します。

ある変数に入っている文字が何かで処理を分岐させるには switch() 関数を使います。上で定義した関数等を組み合わせて、４種類のグラフを描き分けます。関数名は fgsimple としました。

なお、plts3〜pltn はグラフの色指定で、plts3 が単一回答の棒グラフ（１色）、plts2 が単一回答の帯グラフ（２色以上）、pltm が複数回答（１色）、pltn が数値記入項目（１色）です。

```
> fgsimple<-function(dr,dq,da,qNo,obi=NULL,
+                plts3=cb_palette[1],
+                plts2=cb_palette,
+                pltm=cb_palette[3],
+                pltn=cb_palette[4]){
+   #設問番号
+   qID<-dq[,"qID"]
+   #データ抽出
+   d02r<-select(dr,contains(qID[qNo]))  #i 番目 (qNo) の回答データ
+   d02r<-na.omit(d02r)                  #無回答を削除!!!
+   d02q<-dq[qNo,"question"]             #i 番目の設問文
+   d02g<-dq[qNo,"graph"]                #i 番目のグラフの種類
+   d02_ord<-dq[qNo,"order"]             #i 番目の x の並べ方
+   d02a_0<-da[,qID[qNo]]                #i 番目の選択肢
+   d02a_0<-d02a_0[d02a_0!=""]           #選択肢から空白を削除
+   na01<-length(d02a_0)                 #選択肢の数をカウント
+   nr01<-length(d02r[,1])               #有効回答の数をカウント
+
+   #グラフ描画のための x と y の値を求める
+   p02<-switch(d02g,
+         "単一回答"=fxsingle(d02r,d02a_0,na01),
+         "複数回答"=fxymulti(d02r,d02a_0,na01),
+         "数値記入"=fxynumeric(d02r,d02a_0),
+         print("グラフ指定は「単一回答」「複数回答」「数値記入」で！"))
+
+   #X 軸の並べ方を指定
+   if(d02_ord=="同"){
+     xord<--p02$aID
+   }else if(d02_ord=="大"){
+     xord<-p02$prop
+     xord[grepl("その他",p02$xtext)]<<--1
+   }else{
+     xord<-NULL
+   }
+
+   #X 軸の並べ方を指定（数値記入項目用）
+   if(d02g=="数値記入"){
+     xord<--p02$aID
+   }
+
+   #グラフタイトル
+   ttl01<-paste(d02q,"\n"," (",d02g,"N=",nr01,") ") #グラフタイトル
+
+   #obi=を指定しなかったら、選択肢数が 2 の場合は obi=に TRUE を、3 以上は FALSE を
+   obi<-ifelse(is.null(o01),ifelse(na01==2,TRUE,FALSE),obi)
+   #obi=TRUE だったら、グラフの種類を「単一回答帯」に
+   d02gr<-paste(d02g,ifelse(obi,"帯",""),sep="")
+
+   #グラフの定義
+   switch(d02gr,
```

```
+         "単一回答"=fgbars(p02,xord,ttl01,gfbars,palette=plts3),
+         "単一回答帯"=fgobis(p02,ttl01,gfobis,palette=plts2),
+         "複数回答"=fgbars(p02,xord,ttl01,gfbars,palette=pltm),
+         "数値記入"=fgbars(p02,xord,ttl01,gfbars,palette=pltn),
+         print("グラフ指定は「単一回答」「複数回答」「数値記入」で！"))
+ }
```

2.6.2　関数定義を別ファイルで扱う

以上のように定義した関数は、それを使う前に、R コンソールにコピペして、一旦実行させる必要があります。しかし、それを毎回行うのは面倒ですので、この部分だけ別ファイルに保存しておいて、それを読み込むことにしましょう。

以下を、テキストエディタまたは RStudio の「File」→「New File」→「R Script」に記述し、simple01.R という名前で、R の作業フォルダに保存します。エンコードは UTF-8 です。

```
library(ggplot2)
library(plyr)
library(scales)
library(dplyr)

cb_palette<-c("#0072B2","#F0E442","#009E73","#56B4E9",
              "#CC79A7","#D55E00","#E69F00","#999999")


#グラフの x や y の値を計算する関数の定義
fxsingle<-function(dr=d02r,da=d02a_0,na01=na01){
  colnames(dr)<-"aID"
  d02a<-data.frame(aID=1:na01,xtext=da)
  d02<-join(dr,d02a,by="aID")              #選択肢番号を文に変換
  t02<-plyr::count(d02)                    #集計
  transform(t02,prop=freq/sum(freq))       #構成比を求める
}

#複数回答で設問の指摘率を求める部分
fxymulti<-function(dr=d02r,da=d02a_0,na01=na01){
  d02_p1<-summarise_each(dr,funs(sum))     #各選択肢の指摘数
  d02_p0<-summarise_each(dr,funs(length)) #各選択肢の回答者数
  prop_1<-d02_p1/d02_p0                    #各選択肢の指摘率
  data.frame(x=1:na01,
             xtext=da,
             freq=t(d02_p1),
             prop=t(prop_1))
}

#数値記入項目で指定した階級の構成比を求める部分
fxynumeric<-function(dr=d02r,da=d02a_0){
  d02a_1<-na.omit(da)                          #階級の区分から NA を削除
  na01<-length(d02a_1)-1                       #階級の数
  d02r1<-dr[,1]                                #回答の値をベクトルに
  d02<-data.frame(xtext=cut(d02r1,d02a_1))     #階級の区切りで回答の値を割り振る
  t02<-plyr::count(d02)                        #集計
  p02<-transform(t02,prop=freq/sum(freq))      #構成比を求める
  data.frame(aID=1:(na01),p02)                 #単純集計のデータと同じ形式に
}


#グラフのフォーマットを定義する
```

```
#横棒グラフ用
gfbars<-theme_bw()+                              #白黒基調の組み込みテーマを使う
  theme(panel.border=element_blank(),            #描画領域の枠線を消す
        panel.grid=element_blank(),              #目盛線を消す
        axis.title=element_blank(),              #軸タイトルを消す
        axis.ticks.y=element_blank(),            #Y 軸（縦軸？）の目盛を消す
        axis.text.y=element_text(size=10),       #Y 軸（縦軸？）ラベルの文字サイズ
        axis.line.x=element_line(colour="grey"), #軸線を grey で書き足す
        axis.line.y=element_line(colour="grey")) #軸線を grey で書き足す

#帯グラフ用
gfobis<-theme_bw()+                             #白黒基調の組み込みテーマを使う
  theme(panel.border=element_blank(),           #描画領域の枠線を消す
        panel.grid.major=element_blank(),       #目盛線を消す
        axis.ticks=element_blank(),             #目盛を消す
        axis.text.x=element_blank(),            #目盛ラベルを消す
        axis.title=element_blank(),             #軸タイトルを消す
        legend.position="top")                  #凡例を上に配置する

#グラフを定義して描画する関数

#横棒グラフ用
fgbars<-function(p02=p02,xord=xord,ttl01=ttl01,
                 theme=gfbars,palette=cb_palette[1]){
  #グラフの定義
  ggbars<-ggplot(p02,                            #データを指定
                 aes(x=reorder(x=xtext,X=xord),#X 軸は xtext を xord の順番で
                     y=prop))+                     #Y 軸は prop のデータを
    coord_flip()+                                #横棒グラフに
    scale_y_continuous(labels=percent,           #Y 軸の目盛は%表記
                       limits=c(0,1))+           #Y 軸の範囲は 0〜1
    geom_text(                                     #データラベルを記入
      aes(y=prop,                                  #ラベルの位置
          label=sprintf("%.1f%%",prop*100)),       #prop*100 を、数点以下 1 桁で%表記
      size=4,                                      #ラベルの文字サイズは 4
      hjust=-0.1)+                                 #ラベルの位置調整、0.1 大きく
    ggtitle(ttl01)+                              #グラフタイトル
    guides(fill=FALSE)                           #凡例は非表示
  #グラフを表示
  ggbars+theme+geom_bar(stat="identity",fill=palette)
}


#帯グラフ用
fgobis<-function(p02=p02,ttl01=ttl01,
                 theme=gfbars,palette=cb_palette){
  #グラフの定義
  p03<-transform(p02,lposition=cumsum(prop)-prop/2) #データラベルの位置
  ggobis<-ggplot(p03,                           #データを指定
                 aes(x="",                        #X 軸は 1 個だけなので空白
                     y=prop,                      #Y 軸は prop のデータを
                     fill=xtext),                 #棒を xtext ごとに塗り分ける
                 position=fill)+                  #100% 積み上げグラフに
    coord_flip()+                               #横棒グラフに
    scale_fill_manual(values=palette)+          #棒の色
    ggtitle(ttl01)+                             #グラフタイトル
    guides(fill=guide_legend(title=NULL))       #凡例のタイトルを消す
  #グラフを表示
  ggobis+theme+geom_bar(stat="identity",colour="white")+
    geom_text(                                  #データラベルを記入
      aes(y=lposition,                          #ラベルの位置
```

```
          label=sprintf("%.1f%%",prop*100)),#prop*100 を数点以下 1 桁で%表記
      size=4,                              #ラベルの文字サイズは 4
      colour="white")                      #ラベル文字の色
}



#グラフを描き分ける
fgsimple<-function(dr,dq,da,qNo,obi=NULL,
                   plts3=cb_palette[1],
                   plts2=cb_palette,
                   pltm=cb_palette[3],
                   pltn=cb_palette[4]){
  #設問番号
  qID<-dq[,"qID"]
  #データ抽出
  d02r<-select(dr,contains(qID[qNo]))  #i 番目（qNo）の回答データ
  d02r<-na.omit(d02r)                  #無回答を削除!!!
  d02q<-dq[qNo,"question"]             #i 番目の設問文
  d02g<-dq[qNo,"graph"]                #i 番目のグラフの種類
  d02_ord<-dq[qNo,"order"]             #i 番目の x の並べ方
  d02a_0<-da[,qID[qNo]]                #i 番目の選択肢
  d02a_0<-d02a_0[d02a_0!=""]           #選択肢から空白を削除
  na01<-length(d02a_0)                 #選択肢の数をカウント
  nr01<-length(d02r[,1])               #有効回答の数をカウント

  #グラフ描画のための x と y の値を求める
  p02<-switch(d02g,
              "単一回答"=fxsingle(d02r,d02a_0,na01),
              "複数回答"=fxymulti(d02r,d02a_0,na01),
              "数値記入"=fxynumeric(d02r,d02a_0),
              print("グラフ指定は「単一回答」「複数回答」「数値記入」で！"))

  #X 軸の並べ方を指定
  if(d02_ord=="同"){
    xord<--p02$aID
  }else if(d02_ord=="大"){
    xord<-p02$prop
    xord[grepl("その他",p02$xtext)]<--1
  }else{
    xord<-NULL
  }

  #X 軸の並べ方を指定（数値記入項目用）
  if(d02g=="数値記入"){
    xord<--p02$aID
  }

  #グラフタイトル
  ttl01<-paste(d02q,"\n"," (",d02g,"N=",nr01,") ") #グラフタイトル

  #obi=を指定しなかったら、選択肢数が 2 の場合は obi=に TRUE を、3 以上は FALSE を
  obi<-ifelse(is.null(o01),ifelse(na01==2,TRUE,FALSE),obi)
  #obi=TRUE だったら、グラフの種類を「単一回答帯」に
  d02gr<-paste(d02g,ifelse(obi,"帯",""),sep="")

  #グラフの定義
  switch(d02gr,
         "単一回答"=fgbars(p02,xord,ttl01,gfbars,palette=plts3),
         "単一回答帯"=fgobis(p02,ttl01,gfobis,palette=plts2),
         "複数回答"=fgbars(p02,xord,ttl01,gfbars,palette=pltm),
         "数値記入"=fgbars(p02,xord,ttl01,gfbars,palette=pltn),
         print("グラフ指定は「単一回答」「複数回答」「数値記入」で！"))
```

```
}


#棒グラフの選択肢文の幅を統一して（最大に合わせて)、グラフの幅を同じにする

#全設問で最大の選択肢文の幅を抽出する
maxWidth_s<-function(dr,dq,da){
  #選択肢が最大の設問 ID を調べる
  qNo_amax<-da%>%
    as.matrix %>%                 #data.frame だと次の nchar が受け付けないので
    nchar(type="width")%>%        #文字列の幅を測る
    as.data.frame %>%             #data.frame に戻す
    summarise_each(funs(max))%>% #各列の最大値
    which.max                     #最大の設問番号

  g01_amax<-fgsimple(dr,dq,da,qNo_amax) #グラフを出力
  gp01_amax<-ggplotGrob(g01_amax)       #グラフのオブジェクト化
  gp01_amax$widths[3]                    #選択肢文の部分の幅
}

#最大の設問文の幅を外挿してグラフを描きなおす
fgsimple02<-function(dr,dq,da,qNo,obi=NULL,maxw=maxw01,
                plts3=cb_palette[1],
                plts2=cb_palette,
                pltm=cb_palette[3],
                pltn=cb_palette[4]){
  g01_adj<-fgsimple(dr,dq,da,qNo,obi)      #とりあえず目的のグラフを描く
  gp01_adj<-ggplotGrob(g01_adj)            #そのグラフをオブジェクト化
  gp01_adj$widths[3]<-maxw                 #選択肢文の最大幅を外挿
  grid.draw(gp01_adj)                      #グラフの描画
}
```

この関数定義を使いたいときは、source() 関数を使って、保存したファイルを読み込むだけです。

```
> source("simple01.R",encoding = "utf-8")
```

これで、グラフを描き分ける関数 fgsimple() が使えるはずです。

2.6.3　グラフを描いてみる

データの読み込む最初の手順から初めて、5 つの設問に対応する 5 つのグラフを描いてみましょう。

作業フォルダを指定します。たとえば　C:\Users\hogehoge\Documents など。

```
> setwd("c:/Users/hogehoge/Documents")
```

ここにデータや関数定義のスクリプト simple01.R も置いてあるものとします。データを読み込みます。

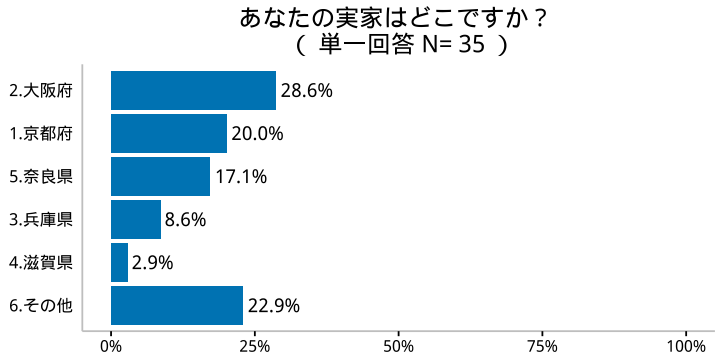
```
> d01r<-read.csv("dset_r.csv")
> d01q<-read.csv("dset_q.csv",as.is=T)
> d01a<-read.csv("dset_a.csv")
```

関数を定義して保存したファイルを読み込みます。

```
> source("simple01.R",encoding = "utf-8")
```

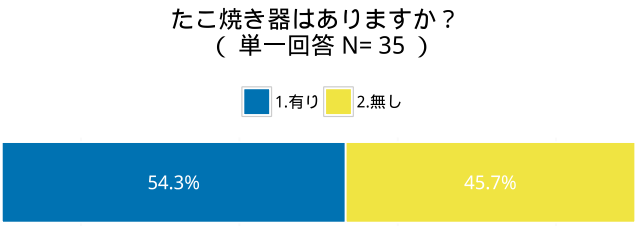
1 番目の設問の単純集計グラフです。単一回答について棒グラフを指定します。

```
> fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=1,obi=FALSE)
```



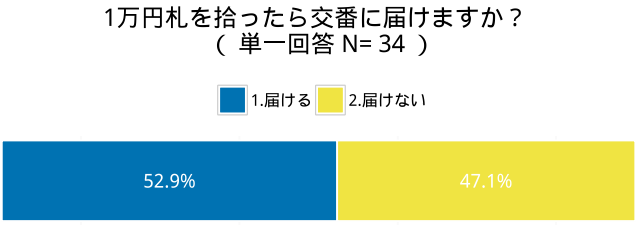
2 番目の設問のグラフです。qNo=2 だけ変えて、帯グラフを指定します。

```
> fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=2,obi=TRUE)
```



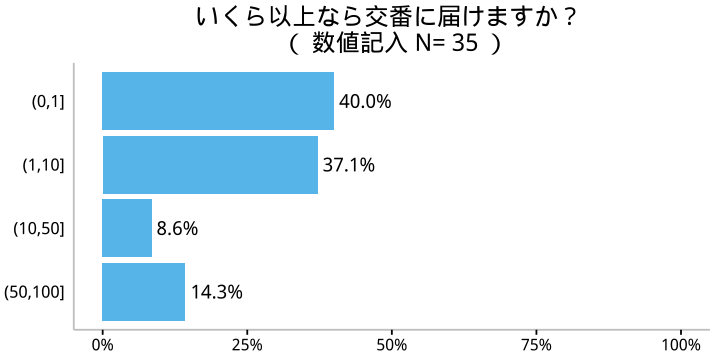
3 番目の設問のグラフです。dr=等の表記を省略しました。

```
> fgsimple(d01r,d01q,d01a,3,T)
```



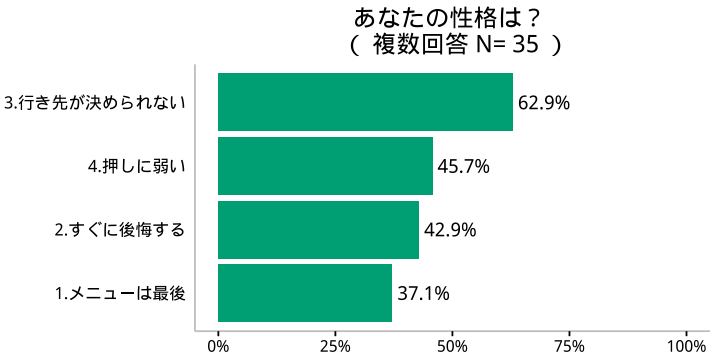
4 番目の設問のグラフです。数値記入項目を指定した階級に区切って集計してあります。

```
> fgsimple(d01r,d01q,d01a,4)
```



5 番目の設問のグラフです。複数回答です。

```
> fgsimple(d01r,d01q,d01a,5)
```



2.7　報告書を作成する

2.7.1　Wondows の拡張メタファイルとしてグラフを保存

もし、あなたが LaTeX を利用しているなら、報告書の作成には、Sweave を利用するととても便利です。しかし、LaTeX って何？という人には、少々敷居が高いかもしれませんので、それについては後回しにして、一般によく使われる MS-Word について説明しましょう。

グラフを拡張メタファイルとして保存して貼り付ける

R で作成したグラフは、いろいろな形式で保存することができます。PDF, EPS, JPEG, TIFF, PNG, BMP, SVG 等々。ワードに貼り付けるなら、Windows メタファイル WMF か、その拡張版である拡張メタファイル EMF がよいでしょう。PNG 等の画像ファイルは、ラスタ画像といって、文字などのスケールの変化に対応しません（ガタガタになる）。その点、PDF や EPS は大丈夫なのですが、ワードに読み込むのが大変です。EMF ファイルは、マイクロソフトが推奨しているベクター画像で、Windows 独特ですが、Windows で使うなら、使いやすい画像形式ということになります。

R のグラフを拡張メタファイルに保存するために、`win.metafile()` 関数が用意されています。しかし、ライブラリ ggplot2 を使っていたら、`ggsave()` 関数を使った方が勝手がよいかもしれません。`ggsave()` 関数は、引数に指定したファイルの拡張子を明示するだけでそのファイル形式で保存してくれます。`.png` なら PNG 画像として、`.emf` なら EMF 画像として保存してくれます。`.emf` の場合、結局は `win.metafile()` 関

数を呼び出しているのですが、画像の形式で関数を使い分ける必要がないし、柔軟にいろいろ指定できます。

1 番目の設問の単純集計グラフを EMF 画像として保存してみます。

```
> p01<-fgsimple(d01r,d01q,d01a,1,obi=TRUE)
> ggsave("plot-01.emf",plot=p01,width=6,height=2,units="in",pointsize=20)
```

グラフを p01 に保存して、`plot-01.emf` というファイル名で出力します。幅は 6 インチ高さが 2 インチで指定してあります。`units="in"`は長さの単位としてインチを選択しています。cm にしたかったら、`units="cm"` とすればよいです。

R 上では何も出力されませんが、作業フォルダに plot-01.emf というファイルができているはずです。ワードにこのファイル（図）を挿入すれば完成です。

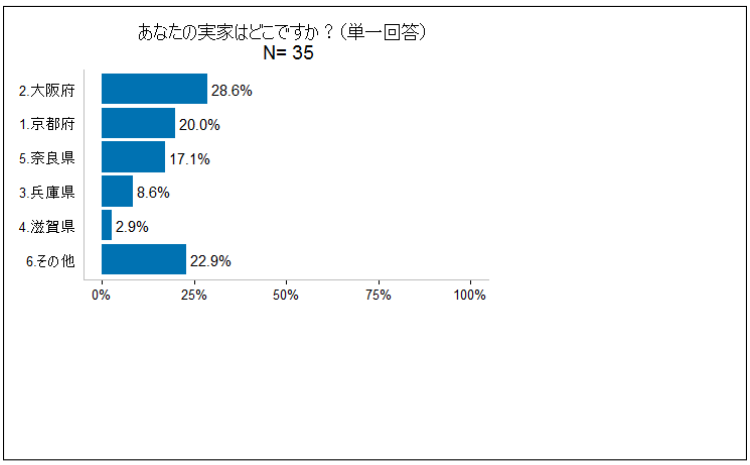


図 2.7: Windows の 150 %拡大表示で emf ファイルに無駄な余白が入る

ただし、ここで注意です。あなたが作成したメタファイルは、やたらと右と下に余白が入っていませんか？もしそうなら、ディスプレイの表示を 100 %以上に設定していることが原因だと思います。「ディスプレイのカスタマイズ」のところで、「テキスト、アプリ、その他の項目のサイズを変更する」が、100 %より大きくなっていないか確認してみてください。Windows 10 なら、「すべての設定」＞「システム」＞「ディスプレイのカスタマイズ」の順です。最近は、Windows PC も高解像度となって、この設定を 150 %にしている場合が多いです。その場合は、この値を 100 %に戻すと、出力されるメタファイルは、無駄な余白の無いものとなます。

図 2.8: emf ファイルを出力する時は Windows の画面は 100 %で

全てののグラフをいっぺんに作成する

5 つのグラフをいっぺんに出力できないでしょうか。できます。`for()` 関数を使います。

`for()` 関数は、次のように使いまず。たとえば、$1^2, 2^2, 3^2, 4^2, 5^2$ を計算したかったら、

```
> for(i in 1:5){
+   print(i^2)
+ }
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```

$1^2, 3^2, 5^2, 7^2, 9^2$ を計算したかったら、

```
> for(i in c(1,3,5,7,9)){
+   print(i^2)
+ }
[1] 1
[1] 9
[1] 25
[1] 49
[1] 81
```

全く同じように、グラフのファイル出力は以下のようにします。

```
> for(i in 1:5){
+   p01<-fgsimple(d01r,d01q,d01a,i)
+   fname01<-sprintf("plot-%02d.emf",i)
+   ggsave(fname01,plot=p01,width=6,height=3,units="in")
+ }
```

`sprintf("plot-%02d.emf",i)` の部分がわかりにくいかもしれませんが、これで、plot-01.emf, plot-02.emf, ..., plot-05.emf というファイル名で 5 つのグラフを出力してくれます。`sprintf()` 関数は、文字列の中に変

数に入っている数字を埋め込むときに、形式を指定してくっつけることができます。%02d というところに、２桁で i の数字を入れてくれます。

2.7.2　R から直接ワードファイルを作成する

R Markdown と Pandoc の利用

R でグラフを描いてメタファイルとして出力できるのはわかりましたが、そのファイルをいちいちワードに貼り付けるのも面倒です。R でワードのファイルそのものを作成できないでしょうか？

できます。rmarkdown というライブラリと Pandoc というソフトウエアを使います。rmarkdown は、通常の R パッケージのインストールでできます。

```
> install.packages("rmarkdown")
```

rmarkdown をインストールしたら読み込みます。

```
> library("rmarkdown")
```

Pandoc は、以下の公式サイトから「installing」を選んで、ダウンロードしてください。

http://pandoc.org/installing.html

いくつか種類がありますが、Windows の場合は、「.msi」の拡張子のついたものを選ぶとよいでしょう。

ダウンロードしたら、そのまま実行させるとインストールできます。このソフトウエアは、rmarkdown の背後で働くので、特に実行させてみる必要はありません。そのままほっといてください。Pandoc がちゃんとインストールできているか確認するには、以下を実行して TRUE と出れば OK です。

```
> pandoc_available()
[1] TRUE
```

次に、Rmd ファイルを作成します。RStuio で新規作成する場合に、「R Markwown」を選びます。

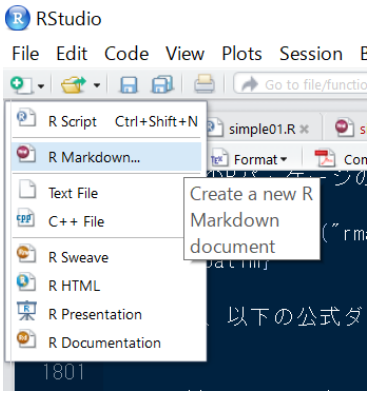


図 2.9: RStudio で Rmd ファイルを新規作成する

Default Output Format で Word を選びます。

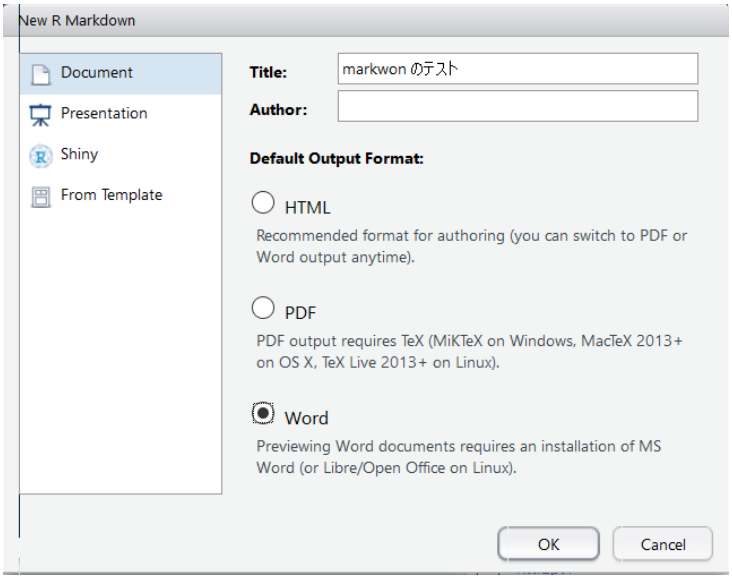


図 2.10: Word ファイルとして出力する場合

そこでできたサンプルの Rmd ファイルを書き換えて、次のように記述してみましょう。

````
---
title: "markdown のテスト"
output: word_document
---

# R でワードファイル

## Rmd ファイル

R からワードのファイルを作成することができます。まず、Rmd ファイルというのを作成します。作成の仕方は、

1. Emacs や秀丸、メモ帳といったテキストエディタを使う。
2. RStudio を使う。

Rmd ファイルは、ファイルの最初に、``title: "ほげほげ"``と ``output: word_document``を``---``と``---``の行で挟んで書きます。

あとの文章はそのまま書けばいいです。

## 見出し
見出しは、見出しにしたい g 行頭に「#」とか「##」を入れるだけです。

## R の出力
r のコマンドは、`` ```{r} ``と`` ``` ``の行で囲みます（ここを「Chunk」と呼んで言います）。すると、コマンドと出力結果を表示してくれます。
```{r}
(x01<-data.frame(x1=c("a","b","c"),y=c(1,3,2)))
```

## グラフの出力
グラフも出力できます。その際、コマンドを出力したくなかったら``ecoh=FALSE`` としてやれば、グラフだけ表示します。``fig.width=``はグラフの幅、``fig.height=``はグラフの高さを指定します。
```{r,ecoh=FALSE,fig.width=7,fig.height=3,dev="png"}
barplot(x01$y,names.arg=x01$x1)
```
````

これを、mkdwntest01.Rmd というファイル名で作業フォルダに保存します。エンコードは、「utf-8」を選

んでください。

次に、

```
> render("cross02.Rmd",encoding="utf-8")
```

とすると、作業フォルダに、mkdwntest01.docx というワードファイルができているはずです。

markdown のテスト

R でワードファイル

Rmd ファイル

R からワードのファイルを作成することができます。まず、Rmd ファイルというのを作成します。作成の仕方は、

1. Emacs や秀丸、メモ帳といったテキストエディタを使う。
2. RStudio を使う。

Rmd ファイルは、ファイルの最初に、`title: "ほげほげ"` と `output: word_document` を`---`と`---`の行で挟んで書きます。

あとの文章はそのまま書けばいいです。

見出し

見出しは、見出しにしたい g 行頭に「#」とか「##」を入れるだけです。

R の出力

r のコマンドは、```{r}と```の行で囲みます（ここを「Chunk」と呼んで言います）。すると、コマンドと出力結果を表示してくれます。

```
(x01<-data.frame(x1=c("a","b","c"),y=c(1,3,2)))
```

```
##   x1 y
## 1  a 1
## 2  b 3
## 3  c 2
```

グラフの出力

図 2.11: R Markdown でワードファイルに出力

文字の大きさが 12pt だったり、図の大きさが多きすぎたり、見出しの書式が気に入らなかったり、いろいろあるかと思いますが、それはワードなので、スタイルの定義を変更することでなんとでもなります。

markdown のテスト

1. R でワードファイル

（1） Rmd ファイル

R からワードのファイルを作成することができます。まず、Rmd ファイルというのを作成します。作成の仕方は、

1. Emacs や秀丸、メモ帳といったテキストエディタを使う。
2. RStudio を使う。

Rmd ファイルは、ファイルの最初に、`title: "ほげほげ"`と`output: word_document`を`---`と`---`の行で挟んで書きます。

あとの文章はそのまま書けばいいです。

（2） 見出し

見出しは、見出しにしたい g 行頭に「#」とか「##」を入れるだけです。

（3） R の出力

r のコマンドは、```` ```{r} ````と```` ``` ````の行で囲みます（ここを「Chunk」と呼んで言います）。すると、コマンドと出力結果を表示してくれます。

```
(x01<-data.frame(x1=c("a","b","c"),y=c(1,3,2)))
```

```
##   x1 y
## 1  a 1
## 2  b 3
## 3  c 2
```

（4） グラフの出力

グラフも出力できます。その際、コマンドを出力したくなかったら`ecoh=FALSE`としてやれば、グラフだけ表示します。`fig.width=`はグラフの幅、`fig.height=`はグラフの高さを指定します。

```
barplot(x01$y,names.arg=x01$x1)
```

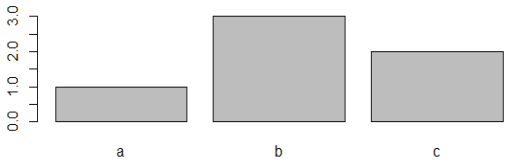


図 2.12: R Markdown でワードファイルに出力して、書式等を変更

問題は、図が png だということです。拡張メタファイル (.emf) で貼り付けられないこともないのですが、現状ではサイズ指定がうまくなく、ちっちゃなヘンな形で貼り付けられてしまいます。そのうち改善されるとは思いますが・・・

日本語表示の問題

しかし、それ以上にどうしようもない問題があります。Windows では日本語の扱いに難があります。本文中の日本語は大丈夫なのですが、Chank 内の日本語がだめです。

以下を、mkdowntest02.Rmd という名前で保存し（エンコードは utf-8）て、ワードファイルを作成してみてください。

````
---
title: "markdown のテスト"
output: word_document
---
Chunk 内で日本語を使うとエラー！
```{r}
x01<-data.frame(x1=c("漢字","ひらがな","カタカナ"),y=c(1,3,2)))
```
````

ワードファイルの作成は以下。

```
> render("mkdwntest02.Rmd",encoding="utf-8")
```

`invalid multibyte string`（へんな全角文字が入っとる！）というエラーが出て、変換作業が止まるはずです。Mac や Linux では大丈夫なのですが、Windows はどうしようもないです。どうかしようがあるのかもしれませんが、私が知る限りはどうしようもないです。

日本語入り Rmd を無理やり使う

ただし、Chank 内で日本語を使わないようにすれば、この問題は回避できます。方法は、一旦、R のコンソール上で、日本語を含む変数や関数の定義をやってしまって、その変数や関数を Rmd 内で使えばよいです。

例えば、先ほどの例だと、変数 x01 の定義が日本語を含ので、前もって、R のコンソール上で、x01 を定義しておきます。

```
> x01<-data.frame(x1=c("漢字","ひらがな","カタカナ"),y=c(1,3,2))
```

それから、次のような Rmd ファイルを作成し、mkdwntest02.Rmd という名前で、R の作業フォルダに保存しましょう。

````
---
title: "markdown のテスト"
output: word_document
---
日本語入りのグラフを出力します。
```{r,fig.width=7,fig.height=3}
showtext.begin()
barplot(x01$y,names.arg=x01$x1)
showtext.end()
```
````

ここで `showtexet.begin()` 関数と `showtext.end()` 関数で囲んだのは、Windows 環境でうまく日本語を表示させるためです。これらはライブラリ showtext を必要としますので、前もってインストールしておいてください。RStuido の Packages のところか、`install.packages("showtext")` でインストールできるはずです。

そして、ライブラリ `showtext` を読み込みます。

```
> library(showtext)
```

続いて、以下を実行します。

```
> render("mkdwntest02.Rmd",encoding="utf-8")
```

以上で、難なく日本語入りのグラフの入ったワードファイルが作成できるはずです。

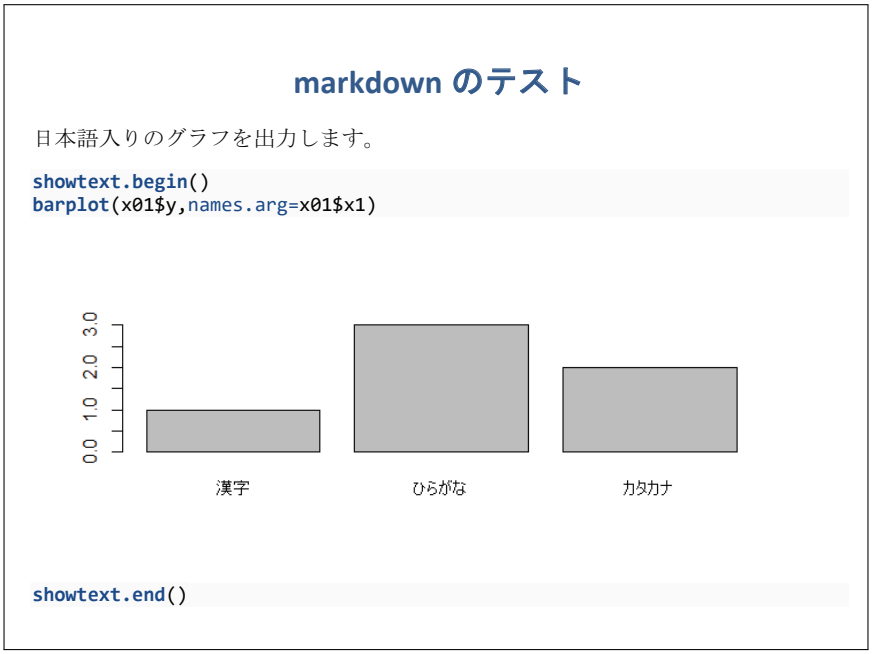


図 2.13: R Markdown で日本語入りのグラフを出力する

R のコマンドを出力しない

R から直接ワードファイルに出力できることはわかりましたが、そのための R のコマンドも一緒に出力されています。自分の作業記録としては便利ですが、報告書を作成する場合は不要です。消しましょう。

R のコマンドを出力しないようにするには、```{r}のところのオプションとして echo=FALSE を追加します。

````
---
title: "markdown のテスト"
output: word_document
---
日本語入りのグラフを出力します。
```{r,fig.width=7,fig.height=3,echo=FALSE}
showtext.begin()
barplot(x01$y,names.arg=x01$x1)
showtext.end()
```
````

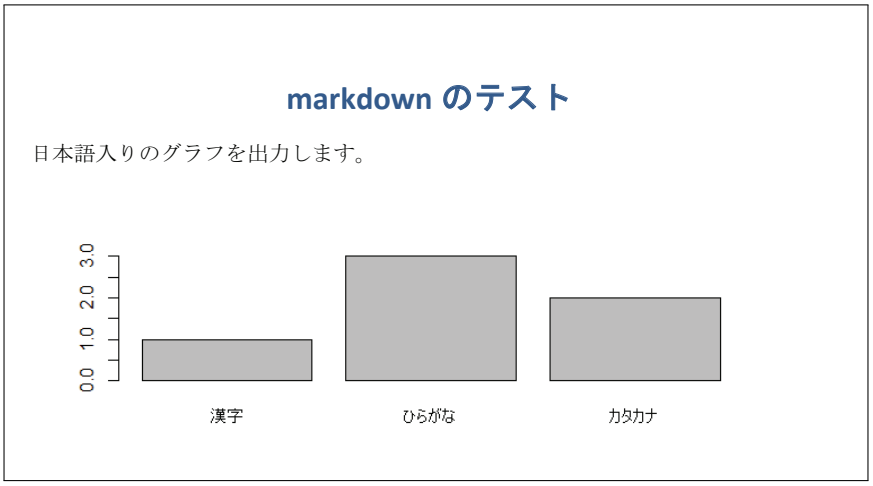


図 2.14: R のコマンドを出力しない

R Markdown で報告書作成

次に、以下のような Rmd ファイルを作成し、simple01.Rmd という名前で保存しましょう。毎回同じですが、エンコードは UTF-8 です。

```
---
title: "調査票と単純集計結果"
output: word_document
---
```{r,echo=FALSE}
source("simple01.R",encoding = "utf-8")
d01r<-read.csv("dset_r.csv")
d01q<-read.csv("dset_q.csv",as.is=T)
d01a<-read.csv("dset_a.csv")
```
問１　あなたの実家はどこですか。次のいずれか 1 つに○印をつけてください。ただし、転居が多い方はこれまで最も長く住んでいた方をお答えください。

1．京都府
2．大阪府
3．兵庫県
4．滋賀県
5．奈良県
```{r,fig.width=7,fig.height=3,echo=FALSE}
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=1)
showtext.end()
```
問２　あなたの実家またはあなたの下宿にはたこ焼き器はありますか。次のいずれか 1 つに○印を つけてください。

1．有る
2．無い
```{r,fig.width=7,fig.height=2,echo=FALSE}
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=2)
showtext.end()
```
問３　もしあなたが現金で 1 万円札を拾ったら、あなたはそれを交番に届けますか?次のいずれか 1 つに○印をつけてください。
```

```
1．届けない
2．届ける
```{r,fig.width=7,fig.height=2,echo=FALSE}
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=3)
showtext.end()
```
問４　現金でお金が落ちていたとき、いくら以上ならあなたはそれを交番に届けますか?

(　　)円以上

```{r,fig.width=7,fig.height=3,echo=FALSE}
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=4)
showtext.end()
```
問５　あなたの性格についてお聞きします。以下の 1.~4．についてあてはまるもの全てに○印をつ けてください。

1．友達とご飯やさんに入って、メニューを決めるのが最後になることが多い。
2．メニューを決めた後、「やっぱり別のにすればよかった」と後悔することが多い。
3．友達と遊びに行くとき、どこに行くか自分できめずに、相手に決めさせることの方が多い。
4．ほんとうはイヤでも「どうしてもお願い」と言われると、つい「はい」と言ってしまい、自分でも「そんなにいやじゃなかったかも」と思い込んでしまうことが多い。

```{r,fig.width=7, fig.height=3,echo=FALSE}
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=5)
showtext.end()
```
```

あとは以下を実行するだけです。

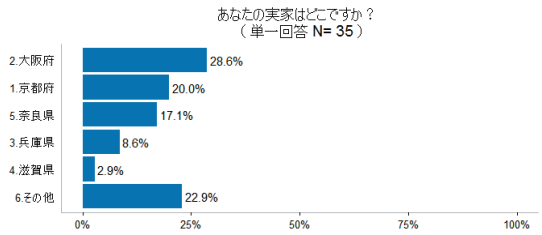
```
> render("simple01.Rmd",encoding="utf-8")
```

これで、simple01.docx という名前のワードファイルが出力されるので、あとは適当に書式を調整してやれば、まあまあ見栄えのする報告書が出来上がります。

調査票と単純集計結果

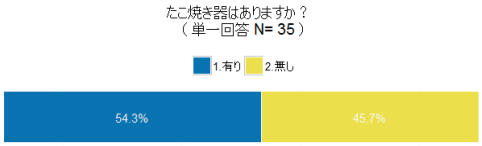
問 1　あなたの実家はどこですか。次のいずれか 1 つに○印をつけてください。ただし、転居が多い方はこれまで最も長く住んでいた方をお答えください。

1．京都府
2．大阪府
3．兵庫県
4．滋賀県
5．奈良県



問 2 あなたの実家またはあなたの下宿にはたこ焼き器はありますか。次のいずれか 1 つに○印を つけてください。

1．有る
2．無い



問 3　もし、あなたが現金で 1 万円札を拾ったら、あなたはそれを交番に届けますか?次のいずれか 1 つに○印をつけてください。

1．届けない

図 2.15: R Markdown を使って R から単純集計の報告書を出力する:Page1

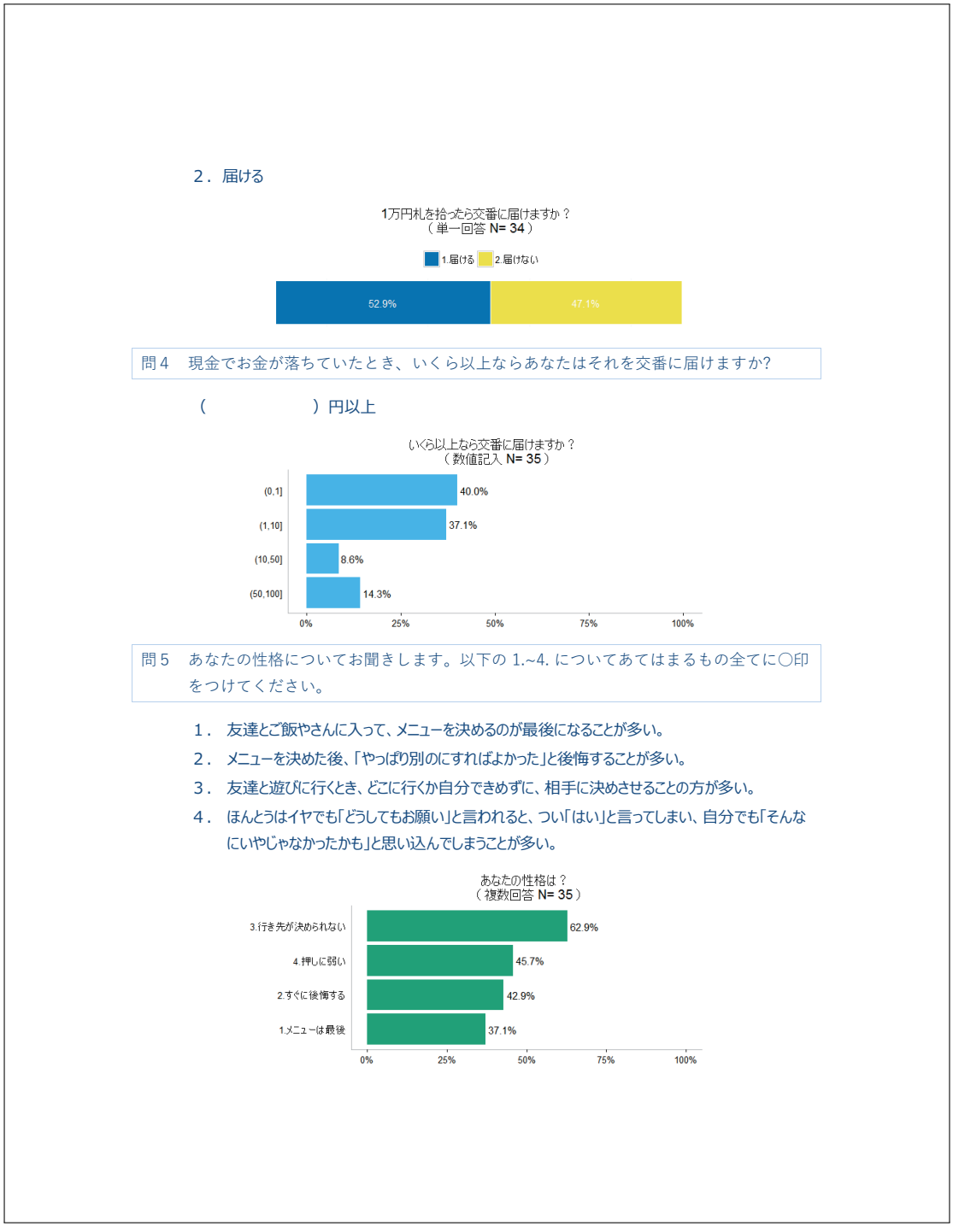


図 2.16: R Markdown を使って R から単純集計の報告書を出力する:Page2

2.7.3　R から直接 LaTeX ファイルを作成する

Sweave というのを使うと、R から直接 PDF の報告書を出力することができます。Sweave の方が、RMarkdown よりも細かい設定ができます。ただし、そのためには LaTeX の知識が必要になります。

LaTeX のインストール

ここでは LaTeX には深入りしません。最小限の情報だけ提供しておきます。インストールは TeX Live というのを使えば比較的簡単かと思います。Windows の場合、以下を参照していくください。

TeX Wiki のインストール解説

RStudio の設定

テキストエディタなどで、以下を記述して、.Rprofile という名前で、作業フォルダに保存しておいてください。

```
Sys.setenv(RSTUDIO_PDFLATEX="..../texlive/2015/bin/win32/ptex2pdf.exe")
```

ただし、`..../texlive/2015` の部分は、自分が TeX Live をインストールしたフォルダに応じて書き換えてください。

Rnw ファイルの作成

RStudio のファイルの新規作成の際、「R Sweave」を選び、以下の記述をしてください。他のテキストエディタでも構いません。

```
\documentclass[uplatex]{jsarticle}
\usepackage[dvipdfmx]{graphicx}
\usepackage[T1]{fontenc}
\usepackage[dvipdfmx]{xcolor}
\usepackage{ascmac}

\begin{document}


\section*{調査項目と単純集計結果}
<<echo=FALSE>>=
setwd("c:/Users/hogehoge/Documents")
d01r<-read.csv("dset_r.csv")
d01q<-read.csv("dset_q.csv",as.is=T)
d01a<-read.csv("dset_a.csv")
@

<<echo=FALSE>>=
source("simple01.R",encoding="utf-8")
@

\begin{itembox}[l]{問１}
あなたの実家はどこですか。次のいずれか 1 つに○印をつけてください。ただし、転居が多い方はこれまで最も長く住んでいた方をお答えください。
\end{itembox}

\begin{enumerate}
  \item 京都府
  \item 大阪府
  \item 兵庫県
  \item 滋賀県
  \item 奈良県
\end{enumerate}

\setkeys{Gin}{width=0.6\textwidth}
<<fig=TRUE,height=3>>=
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=1)
```

```
showtext.end()
@

\begin{itembox}[l]{問２}
あなたの実家またはあなたの下宿にはたこ焼き器はありますか。次のいずれか 1 つに○印を つけてくださ
い。
\end{itembox}

\begin{enumerate}
  \item 有る
  \item 無い
\end{enumerate}

<<fig=TRUE,height=2>>=
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=2)
showtext.end()
@

\begin{itembox}[l]{問３}
もしあなたが現金で 1 万円札を拾ったら、あなたはそれを交番に届けますか?次のいずれか 1 つに○印を
つけてください。
\end{itembox}

\begin{enumerate}
  \item 届けない
  \item 届ける
\end{enumerate}

<<fig=TRUE,height=2>>=
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=3)
showtext.end()
@

\begin{itembox}[l]{問４}
現金でお金が落ちていたとき、いくら以上ならあなたはそれを交番に届けますか?
\end{itembox}

(                    ) 円以上\\

<<fig=TRUE,height=3>>=
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=4)
showtext.end()
@

\begin{itembox}[l]{問５}
あなたの性格についてお聞きします。以下の 1.~4. についてあてはまるもの全てに○印をつ けてくださ
い。
\end{itembox}

\begin{enumerate}
  \item 友達とご飯やさんに入って、メニューを決めるのが最後になることが多い。
  \item メニューを決めた後、「やっぱり別のにすればよかった」と後悔することが多い。
  \item 友達と遊びに行くとき、どこに行くか自分できめずに、相手に決めさせることの方が多い。
  \item ほんとうはイヤでも「どうしてもお願い」と言われると、つい「はい」と言ってしまい、自分でも
「そんなにいやじゃなかったかも」と思い込んでしまうことが多い。
\end{enumerate}

<<fig=TRUE,height=3>>=
showtext.begin()
fgsimple(dr=d01r,dq=d01q,da=d01a,qNo=5)
showtext.end()
@
```

```
\end{document}
```

これを、smp_report01.Rnw という名前で保存しましょう。エンコードは UTF-8 です。

Rnw ファイルを tex ファイルに変換する

作成した smp_report01.Rnw ファイルに書かれている R コマンドを実行し、結果を含めた tex ファイルを作成します。R コンソールから、以下を実行します。

smp_report01.tex というファイルができてているはずです。

PDF ファイルを作成する

次に、Windows だったら CMD や Windows PowerShell などのターミナルを起動し、カレントディレクトリを作業フォルダに移動しておきます。

```
> cd c:\Users\hogehoge\Documents
```

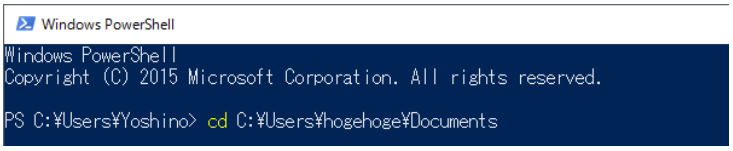


図 2.17: ターミナルでのカレントディレクトリの変更

続いて、ターミナル上で以下のコマンドを実行します。

```
> ptex2pdf -u -l -ot "-kanji=utf8 -no-guess-input-enc -synctex=1" smp_report01.tex
```

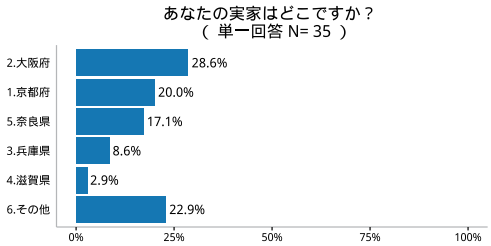
以上でエラーが出ていなければ、次のような PDF ファイルができているはずです。なかなかすんなりうまくいくとは思いませんが、エラーメッセージに注意しながら、格闘してみてください。

調査項目と単純集計結果

問 1

あなたの実家はどこですか。次のいずれか 1 つに○印をつけてください。ただし、転居が多い方はこれまで最も長く住んでいた方をお答えください。

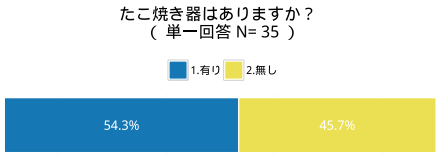
1. 京都府
2. 大阪府
3. 兵庫県
4. 滋賀県
5. 奈良県



問 2

あなたの実家またはあなたの下宿にはたこ焼き器はありますか。次のいずれか 1 つに○印を つけてください。

1. 有る
2. 無い



問 3

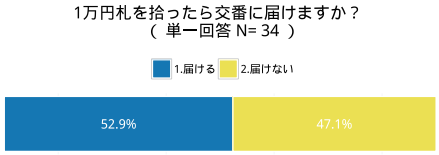
もしあなたが現金で 1 万円札を拾ったら、あなたはそれを交番に届けますか?次のいずれか 1 つに○印をつけてください。

1. 届けない

1

図 2.18: Sweave による PDF の出力:Page1

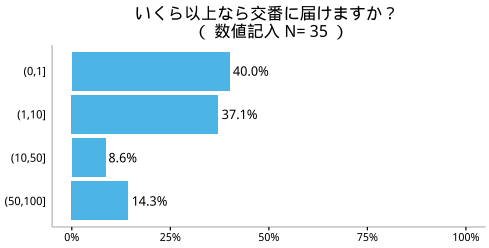
2. 届ける



問 4

現金でお金が落ちていたとき、いくら以上ならあなたはそれを交番に届けますか?

（　　）円以上



問 5

あなたの性格についてお聞きします。以下の 1. 4. についてあてはまるもの全てに○印をつ けてください。

1. 友達とご飯やさんに入って、メニューを決めるのが最後になることが多い。
2. メニューを決めた後、「やっぱり別のにすればよかった」と後悔することが多い。
3. 友達と遊びに行くとき、どこに行くか自分できめずに、相手に決めさせることの方が多い。
4. ほんとうはイヤでも「どうしてもお願い」と言われると、つい「はい」と言ってしまい、自分でも「そんなにいやじゃなかったかも」と思い込んでしまうことが多い。

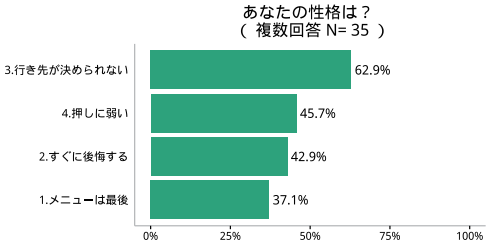
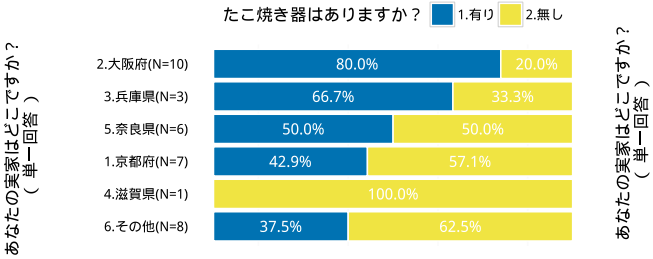


図 2.19: Sweave による PDF の出力:Page2

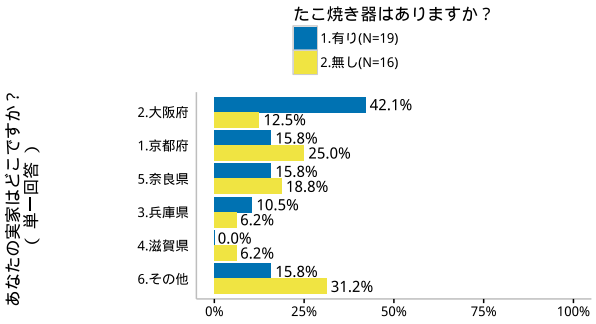
第 3 章

クロス集計のグラフを描く

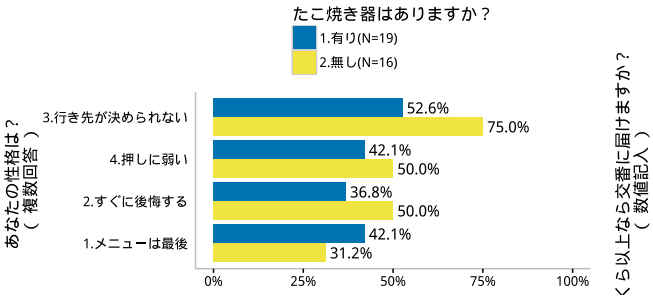
クロス集計のグラフは前々章で例としてやってみましたが、ここではそれを関数化して、報告書の作成ができるまでやってみます。クロス集計の種類も以下の 4 種類です。



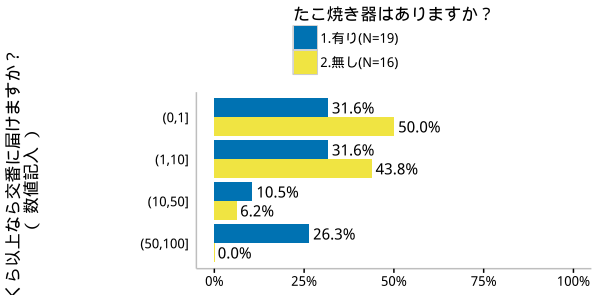
(a) 単一回答×単一回答の帯グラフ



(b) 単一回答×単一回答の棒グラフ



(c) 複数回答×単一回答の棒グラフ



(d) 数値項目×単一回答の棒グラフ

図 3.1: 本章で描くグラフ

3.1 単一回答×単一回答の帯グラフ

この節では、単一回答と単一回答のクロス集計を帯グラフとして出力します。横棒グラフの 100 %積上げで、構成比を描く棒の真ん中に表示しています。前々章で例題としてやったのと同じです。ここではさらに、X 軸をならびかえます。 1 つめの要素の構成比が大きい順に並び替えて、「その他」は最後に持ってきています。

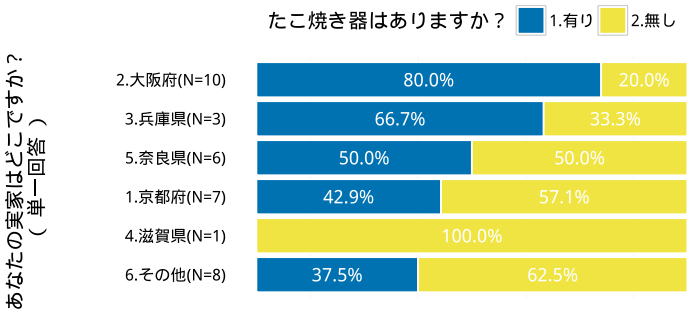


図 3.2: 単一回答×単一回答の帯グラフの例

3.1.1　データの読み込みと抽出

あらかじめ必要となるライブラリを読み込んでおきます。

```
> library(ggplot2)
> library(plyr)
> library(scales)
> library(dplyr)
> library(reshape2)
> library(showtext)
```

データを読み込みます。前の章で使ったデータをそのまま使います。

```
> d01r<-read.csv("dset_r.csv")
> d01q<-read.csv("dset_q.csv",as.is=T)
> d01a<-read.csv("dset_a.csv")
```

クロス集計を行う 2 つの設問の設問番号を指定します。データの設問は d01q に読み込みました。

```
> d01q
  qID                                question    graph order
1 Q01           あなたの実家はどこですか？ 単一回答      大
2 Q02             たこ焼き器はありますか？ 単一回答      同
3 Q03 1 万円札を拾ったら交番に届けますか？ 単一回答      同
4 Q04     いくら以上なら交番に届けますか？ 数値記入
5 Q05                       あなたの性格は？ 複数回答      大
```

設問番号は、この d01q の行番号にあたります。とりあえず例として、Q01 を X 軸用にに選び、Q02 を分類（塗り分け：fill）のための変数とて選びましょう。そうすると設問番号は 1 と 2 です。

```
> qNo_x<-1
> qNo_f<-2
```

後の便利のために、d01q から設問の ID だけ抽出しておきましょう。

```
> #設問番号
> (qID<-d01q[,"qID"])
[1] "Q01" "Q02" "Q03" "Q04" "Q05"
```

続いて、選択した設問に対する個々の回答データを抽出します。

```
> #データ抽出
> d03r_x<-dplyr::select(d01r,contains(qID[qNo_x])) #X 軸に使う設問（X）の回答データ
> d03r_f<-dplyr::select(d01r,contains(qID[qNo_f])) #分類に使う設問（f）の回答データ
```

ただし、これには無回答（NA）が含まれている可能性がありますので、ここで除いておきます。

```
> #無回答の処理
> d03r<-data.frame(d03r_x,d03r_f) #2 つの変数を結合したデータフレーム
> d03r<-na.omit(d03r)             #NA を含む行を削除（しちゃう！）
```

2 つの変数をくっつけて、どちらかの変数に NA があった行は 2 つとも削除しています。片方ずつ削除すると、片方には存在するけど、片方には存在しない行が出てきて、2 つの変数のデータ数が異なるということになるかもしれません。NA を残したまま集計するという方法もありますが、今回はそこには触れません。

d03r_x と d03r_y も再定義しておきます。

```
> d03r_x<-dplyr::select(d03r,contains(qID[qNo_x])) #X の回答データ（NA 無し）
> d03r_f<-dplyr::select(d03r,contains(qID[qNo_f])) #f の回答データ（NA 無し）
```

その他の情報も抽出します。

```
> #その他情報の読み込み
> d03q_x<-d01q[qNo_x,"question"]  #設問文
> d03q_f<-d01q[qNo_f,"question"]
> d03ord_x<-d01q[qNo_x,"order"]  #選択肢の並べ方
> d03ord_f<-d01q[qNo_f,"order"]
> d03gk_x<-d01q[qNo_x,"graph"]   #グラフの種類
> d03gk_f<-d01q[qNo_f,"graph"]
> d03a_x0<-d01a[,qID[qNo_x]]     #選択肢の文言
> d03a_f0<-d01a[,qID[qNo_f]]
> d03a_x0<-d03a_x0[d03a_x0!=""]  #選択肢から空白を削除
> d03a_f0<-d03a_f0[d03a_f0!=""]
> if(is.factor(d03a_x0)){        #（もし因子だったら・・・）
+   d03a_x0<-droplevels(d03a_x0) #選択肢から空白のレベルを削除
+ }
> d03a_f0<-droplevels(d03a_f0)
> na_x0<-length(d03a_x0)         #選択肢の数をカウント
> na_f0<-length(d03a_f0)
> lab_x<-paste(d03q_x,"\n（",d03gk_x,"）")#X 軸ラベル
```

3.1.2　データセットの作成

分類 f の回答データ（選択肢番号で入力）を選択肢の文言へ変換

抽出した回答データ d03r_x と d03r_f は、それぞれ選択肢番号で入力されています。これらを対応する選択肢の文言に変換します。分類のための変数 f については、単純集計と同じです。選択肢の番号と文言の参照表を作成します。

```
> d03a_f<-data.frame(1:na_f0,d03a_f0)      #選択肢の番号と文言の参照表
> colnames(d03a_f)<-c(qID[qNo_f],"ftext")  #参照表に変数名をつける
> d03a_f
  Q02  ftext
1   1 1.有り
2   2 2.無し
```

回答番号を選択肢の文言に変換。

```
> d03_f<-join(d03r_f,d03a_f,by=qID[qNo_f])  #回答データ（番号）を文言へ変換
> head(d03_f)
  Q02  ftext
1   1 1. 有り
2   1 1. 有り
3   1 1. 有り
4   1 1. 有り
5   1 1. 有り
6   2 2. 無し
```

分類項目 f の方のデータについてはこれでできました。

X の回答データ（選択肢番号で入力）を選択肢の文言（N=付き）へ変換

次に、X の回答データを選択肢の文言に変換します。分類 f と同じような作業ですが、X については、各項目ごとに（N=50）といったような標本規模を付け加えたい・・・ということとでちょっと工夫がいります。

最初に各選択肢の合計をカウントしなければなりません。とりあえず、f と同様に標本規模無しで選択肢の文言に変換します。

```
> #X の回答データ（番号で入力）を選択肢の文言へ変換
> d03a_x<-data.frame(1:na_x0,d03a_x0)       #選択肢の番号と文言の参照表
> colnames(d03a_x)<-c(qID[qNo_x],"xtext")  #参照表に変数名をつける
> d03_x<-join(d03r_x,d03a_x,by=qID[qNo_x]) #回答データ（番号）を文言へ変換
> head(d03_x)
  Q01    xtext
1   2 2. 大阪府
2   6 6. その他
3   1 1. 京都府
4   2 2. 大阪府
5   6 6. その他
6   6 6. その他
```

これを使って各選択肢の合計（N=）をカウントします。

```
> (t03_x<-plyr::count(d03_x))
  Q01    xtext freq
1   1 1. 京都府    7
2   2 2. 大阪府   10
3   3 3. 兵庫県    3
4   4 4. 滋賀県    1
5   5 5. 奈良県    6
6   6 6. その他    8
```

選択肢の文言（xtext）と標本規模（freq）をくつけた新たな変数 xtextN を作成します。

```
> (d03_xN<-transform(t03_x,xtextN=paste(xtext,"(N=",freq,")",sep="")))
  Q01    xtext freq          xtextN
1   1 1. 京都府    7  1. 京都府 (N=7)
2   2 2. 大阪府   10 2. 大阪府 (N=10)
3   3 3. 兵庫県    3  3. 兵庫県 (N=3)
4   4 4. 滋賀県    1  4. 滋賀県 (N=1)
5   5 5. 奈良県    6  5. 奈良県 (N=6)
6   6 6. その他    8  6. その他 (N=8)
```

qID と xtextN だけのデータフレーム d03a_xN を作成します。これが選択肢の番号と文言（N=付き）の参照表となります。

```
> (d03a_xN<-subset(d03_xN,select=c(qID[qNo_x],"xtextN")))
  Q01          xtextN
1   1  1. 京都府 (N=7)
2   2 2. 大阪府 (N=10)
3   3  3. 兵庫県 (N=3)
4   4  4. 滋賀県 (N=1)
5   5  5. 奈良県 (N=6)
6   6  6. その他 (N=8)
```

この参照表を使って、X の回答データ d03_rx（番号で入力） を xtextN に対応させます。

```
> d03_xN<-join(d03r_x,d03a_xN,by=qID[qNo_x])
> head(d03_xN)
  Q01          xtextN
1   2 2. 大阪府 (N=10)
2   6  6. その他 (N=8)
3   1  1. 京都府 (N=7)
4   2 2. 大阪府 (N=10)
5   6  6. その他 (N=8)
6   6  6. その他 (N=8)
```

これで、X のデータもできました。

回答データ（選択肢の文言で入力）の作成

これらをくっつけて、描画に使う個票データの抽出が完成です。

```
> d03<-data.frame(xtextN=d03_xN[,"xtextN"],ftext=d03_f[,"ftext"])
> head(d03)
          xtextN  ftext
1 2. 大阪府 (N=10) 1. 有り
2  6. その他 (N=8) 1. 有り
3  1. 京都府 (N=7) 1. 有り
4 2. 大阪府 (N=10) 1. 有り
5  6. その他 (N=8) 1. 有り
6  6. その他 (N=8) 2. 無し
```

帯グラフのためのデータセットの作成

続いて、d03 から帯グラフを描くために構成比などを求めたデータセットを作成します。ライブラリ plyr の `count()` 関数で集計しましょう。

```
> (t03<-plyr::count(d03))
           xtextN  ftext freq
1   1. 京都府 (N=7) 1. 有り    3
2   1. 京都府 (N=7) 2. 無し    4
3  2. 大阪府 (N=10) 1. 有り    8
4  2. 大阪府 (N=10) 2. 無し    2
5   3. 兵庫県 (N=3) 1. 有り    2
6   3. 兵庫県 (N=3) 2. 無し    1
7   4. 滋賀県 (N=1) 2. 無し    1
8   5. 奈良県 (N=6) 1. 有り    3
```

```
9    5. 奈良県 (N=6) 2. 無し    3
10   6. その他 (N=8) 1. 有り    3
11   6. その他 (N=8) 2. 無し    5
```

これでクロス集計ができています！！　便利ですね。

この t03 の変数 xtextN ごとに構成比を求めます。これもライブラリ plyr の ddply() 関数を使います。

```
> (p03<-ddply(t03,"xtextN",transform, prop=freq/sum(freq)))
           xtextN  ftext freq      prop
1    1. 京都府 (N=7) 1. 有り    3 0.4285714
2    1. 京都府 (N=7) 2. 無し    4 0.5714286
3   2. 大阪府 (N=10) 1. 有り    8 0.8000000
4   2. 大阪府 (N=10) 2. 無し    2 0.2000000
5    3. 兵庫県 (N=3) 1. 有り    2 0.6666667
6    3. 兵庫県 (N=3) 2. 無し    1 0.3333333
7    4. 滋賀県 (N=1) 2. 無し    1 1.0000000
8    5. 奈良県 (N=6) 1. 有り    3 0.5000000
9    5. 奈良県 (N=6) 2. 無し    3 0.5000000
10   6. その他 (N=8) 1. 有り    3 0.3750000
11   6. その他 (N=8) 2. 無し    5 0.6250000
```

グラフにデータラベルを貼るので、その位置を計算して新たな変数 lposition としてくっつけます。帯グラフなので、構成比 prop の大きさが順に並んでいます。それぞれの真ん中の位置を計算したい。prop を累計して（cumsum(prop)）、自身の値の半分（prop/2）を引けばよい。

この計算も、ddply() 関数で、xtextN の値ごとに行います。

```
> (p03<-ddply(p03,"xtextN",transform, lposition=cumsum(prop)-prop/2))
           xtextN  ftext freq      prop lposition
1    1. 京都府 (N=7) 1. 有り    3 0.4285714 0.2142857
2    1. 京都府 (N=7) 2. 無し    4 0.5714286 0.7142857
3   2. 大阪府 (N=10) 1. 有り    8 0.8000000 0.4000000
4   2. 大阪府 (N=10) 2. 無し    2 0.2000000 0.9000000
5    3. 兵庫県 (N=3) 1. 有り    2 0.6666667 0.3333333
6    3. 兵庫県 (N=3) 2. 無し    1 0.3333333 0.8333333
7    4. 滋賀県 (N=1) 2. 無し    1 1.0000000 0.5000000
8    5. 奈良県 (N=6) 1. 有り    3 0.5000000 0.2500000
9    5. 奈良県 (N=6) 2. 無し    3 0.5000000 0.7500000
10   6. その他 (N=8) 1. 有り    3 0.3750000 0.1875000
11   6. その他 (N=8) 2. 無し    5 0.6250000 0.6875000
```

帯グラフを描くためのデータフレーム p03 の完成です。

3.1.3　帯グラフを描画する

p03 の値を帯グラフに描画します。ここは前々章でやったので、簡単に解説します。

まず、X 軸の並びを指定します。ここでは、X に選んだ設問の選択肢番号そのままの順番で描きます。横棒グラフは、X 軸を下から積上げていくので、X 軸の項目の順番を逆転させます。前章でもやったように、xord に選択肢項目の名前の順番の逆番号を代入しておきます。

```
> xord<- -order(p03$xtextN)
```

グラフの定義を行う

```
> ggobic<-ggplot(p03,
+              aes(x=reorder(x=xtextN,X=xord),        #X の値と順番を指定
+                  y=prop,                            #Y の値を指定
+                  fill=ftext),                       #塗り分ける変数を指定
+              position="fill")+                      #100 %積上げグラフ
+   coord_flip()+                                     #グラフを横向きに
+   xlab(lab_x)+                                      #X 軸のラベルを指定
+   labs(fill=d03q_f)                                 #凡例のラベルを指定
```

グラフのフォーマットを指定

前々章のフォーマットをそのまま使います。

```
> gfobic<-theme_bw()+                               #theme_bw を使用<=
+   theme(panel.border=element_blank(),               #パネルの枠線を消す
+         panel.grid.major=element_blank(),           #目盛線を消す
+         axis.ticks=element_blank(),                 #軸の目盛を消す
+         axis.text.x=element_blank(),                #目盛ラベルを消す
+         axis.title.x=element_blank(),               #X 軸のタイトルを消す
+         legend.position="top")                      #凡例を上に配置
```

カラーパレットを指定

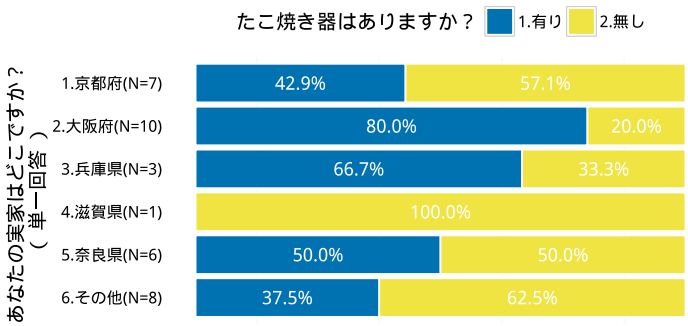
これは前の章で使ったものです。

```
> #カラーパレット
> cb_palette<-c("#0072B2","#F0E442","#009E73","#56B4E9",
+               "#CC79A7","#D55E00","#E69F00","#999999")
```

帯グラフを描画する

データラベルは、棒の中に書き込むので、上から塗りつぶされないように、最後に定義します。

```
> ggobic+gfobic+  geom_bar(stat="identity",
+                      colour="white")+                  #線の色指定
+   scale_fill_manual(values=cb_palette)+                #塗りつぶしの色指定
+   geom_text(aes(y=lposition,                           #データら別の位置
+                 label=sprintf("%.1f%%",prop*100)),     #データラベルを%小数点 1 桁に
+             color="white")                             #文字色を白に
```



X を 1 番目の選択肢の prop が大きい順に並べ替える

X が大きい順に並べてみましょう。帯グラフの場合、何の「大きい順」がよいのか、判断に迷います。X の（N=）の値で並べ替えるというのも一つの方法かもしれません。ここでは、Y 軸の最初の選択肢 **1. 有り**の割合の大きさで並べ替えてみましょう。

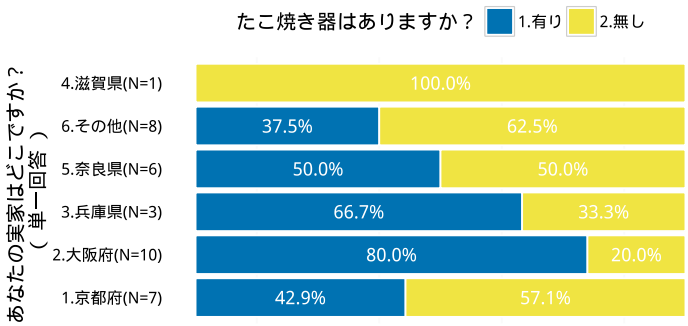
p03 の値を再確認してみましょう。

```
> p03
           xtextN  ftext freq      prop lposition
1   1. 京都府 (N=7) 1. 有り    3 0.4285714 0.2142857
2   1. 京都府 (N=7) 2. 無し    4 0.5714286 0.7142857
3  2. 大阪府 (N=10) 1. 有り    8 0.8000000 0.4000000
4  2. 大阪府 (N=10) 2. 無し    2 0.2000000 0.9000000
5   3. 兵庫県 (N=3) 1. 有り    2 0.6666667 0.3333333
6   3. 兵庫県 (N=3) 2. 無し    1 0.3333333 0.8333333
7   4. 滋賀県 (N=1) 2. 無し    1 1.0000000 0.5000000
8   5. 奈良県 (N=6) 1. 有り    3 0.5000000 0.2500000
9   5. 奈良県 (N=6) 2. 無し    3 0.5000000 0.7500000
10  6. その他 (N=8) 1. 有り    3 0.3750000 0.1875000
11  6. その他 (N=8) 2. 無し    5 0.6250000 0.6875000
```

この例では 6 本の帯グラフが描かれますが、データは 2 倍の 11 行です（「4. 滋賀県」が「2. 無し」だけだから、1 つ少ない）。X 軸の並びに、そのまま構成比 prop を与えてみると・・・

```
> (xord<-p03$prop)
 [1] 0.4285714 0.5714286 0.8000000 0.2000000 0.6666667 0.3333333 1.0000000 0.5000000
 [9] 0.5000000 0.3750000 0.6250000
```

```
> ggobic+gfobic+  geom_bar(stat="identity",
+                      colour="white")+                  #線の色指定
+   scale_fill_manual(values=cb_palette)+                #塗りつぶしの色指定
+   geom_text(aes(y=lposition,                           #データら別の位置
+                 label=sprintf("%.1f%%",prop*100)),     #データラベルを%小数点 1 桁に
+             color="white")                             #文字色を白に
```



・・・と、なんだかよくわからない並びになりました。どうも、X 軸の項目 1 つに 2 つ以上の Y 軸の値がある場合、（たぶんですが）その合計の順番が使われるようです。この場合、全て 1 なので、任意の並びになってしまうようです。

ですから、X 軸の並びとして、Y 軸にあてた選択肢（p03$ftext）が「1. 有り」であるもの以外には 0 を与えましょう。

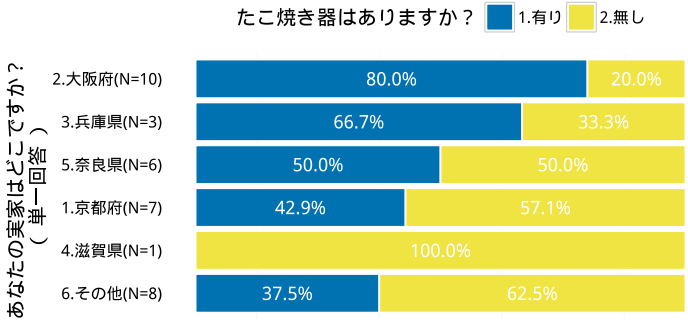
```
> xord[p03$ftext!=d03a_f0[1]]<-0
> xord
 [1] 0.4285714 0.0000000 0.8000000 0.0000000 0.6666667 0.0000000 0.0000000 0.5000000
 [9] 0.0000000 0.3750000 0.0000000
```

ただし、X 軸の項目に「その他」という選択肢がある場合は、最後に持ってきたいので、これには最小値となるマイナスの値を与えます。

```
> xord[grepl("その他",p03$xtextN)]<- -1
> xord
 [1]  0.4285714  0.0000000  0.8000000  0.0000000  0.6666667  0.0000000  0.0000000  0.5000000
 [9]  0.0000000 -1.0000000 -1.0000000
```

グラフを描いてみましょう。

```
> ggobic+gfobic+  geom_bar(stat="identity",
+                        colour="white")+                    #線の色指定
+   scale_fill_manual(values=cb_palette)+                     #塗りつぶしの色指定
+   geom_text(aes(y=lposition,                                #データら別の位置
+                 label=sprintf("%.1f%%",prop*100)),          #データラベルを%小数点 1 桁に
+             color="white")                                  #文字色を白に
```



うまくいきました。

3.2　単一回答×単一回答の棒グラフ

今度は、たこ焼き器がある人とない人で実家の分布を見たい。使うデータは前節と同様で、X が Q01（実家）で、分類 f が Q02（たこ焼き器の有無）です。ただし、求める分布が、X ごとの f の分布ではなく、f ごとの X の分布です。

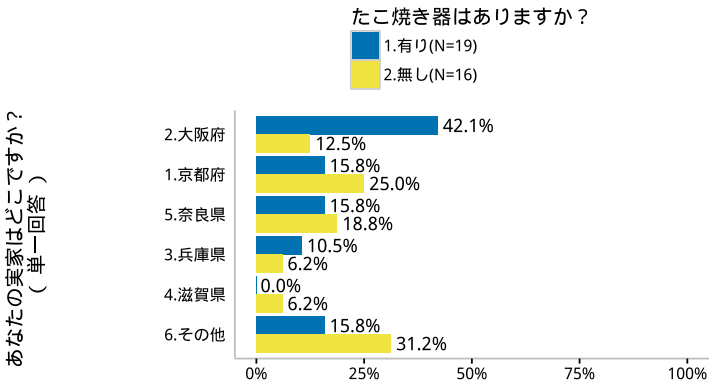


図 3.3: 単一回答×単一回答の棒グラフの例

3.2.1　データセットの作成

X の回答データ（選択肢番号で入力）を選択肢の文言へ変換

ここは前節と全く同じです。

```
> d03a_x<-data.frame(1:na_x0,d03a_x0)          #選択肢の番号と文言の参照表
> colnames(d03a_x)<-c(qID[qNo_x],"xtext")      #参照表に変数名をつける
> d03_x<-join(d03r_x,d03a_x,by=qID[qNo_x])     #回答データ（番号）を文言へ変換
```

分類 f の回答データ（選択肢番号で入力）を選択肢の文言（N=付き）へ変換

今回は f の方を標本規模（N=）付きの文言に変換します。最初は N=無しで変換します。

```
> d03a_f<-data.frame(1:na_f0,d03a_f0)          #選択肢の番号と文言の参照表
> colnames(d03a_f)<-c(qID[qNo_f],"ftext")      #参照表に変数名をつける
> d03_f<-join(d03r_f,d03a_f,by=qID[qNo_f])     #回答データ（番号）を文言へ変換
```

次に、f の文言を（N=付き）にします。これも前節で X について N=付きにしたのと同じことを f についてやります。

```
> t03_f<-plyr::count(d03_f)                          #集計して N=を求める
> d03_fN<-transform(t03_f,
+           ftextN=paste(ftext,"(N=",freq,")",sep="")) #(N=) を X の文言に付ける
>   d03a_fN<-subset(d03_fN,
+                   select=c(qID[qNo_f],"ftextN"))#選択肢の番号と N=付き文言の参照表
>   d03_fN<-join(d03r_f,d03a_fN,by=qID[qNo_f])     #回答データ（番号）を N=付き文言へ
```

データセット作成

X と f の回答データを文言に変換したものをくっつけてクロス集計して構成比を計算すれば、データセットの完成です。

```
> d03<-data.frame(xtext=d03_x[,"xtext"],
+                 ftextN=d03_fN[,"ftextN"])   #X（文言）と f（N=付き文言）のデータ
> t03_0<-table(d03)                           #クロス集計
> t03<-melt(t03_0)                            #クロス集計結果を縦に並べ替え
> colnames(t03)<-c("xtext","ftextN","freq")   #変数名の変更
> p03<-ddply(t03,"ftextN",transform,
+            prop=freq/sum(freq))             #構成比計算で、データセット完成
```

クロス集計 t03 を得るのに、ライブラリ plyr の count() 関数が便利なのですが、あえて従来からある table() 関数を使っています。これは、クロス集計した際に、2 つの棒グラフのどちらかが 0 の場合、0 をカウントするようにです。count() 関数は、0 のデータを出力せず、グラフを描くと不細工になってしまいます。p03 はこんな感じ。

```
> p03
      xtext       ftextN freq      prop
1  1. 京都府 1. 有り (N=19)    3 0.1578947
2  2. 大阪府 1. 有り (N=19)    8 0.4210526
3  3. 兵庫県 1. 有り (N=19)    2 0.1052632
4  4. 滋賀県 1. 有り (N=19)    0 0.0000000
5  5. 奈良県 1. 有り (N=19)    3 0.1578947
6  6. その他 1. 有り (N=19)    3 0.1578947
7  1. 京都府 2. 無し (N=16)    4 0.2500000
8  2. 大阪府 2. 無し (N=16)    2 0.1250000
9  3. 兵庫県 2. 無し (N=16)    1 0.0625000
10 4. 滋賀県 2. 無し (N=16)    1 0.0625000
11 5. 奈良県 2. 無し (N=16)    3 0.1875000
12 6. その他 2. 無し (N=16)    5 0.3125000
```

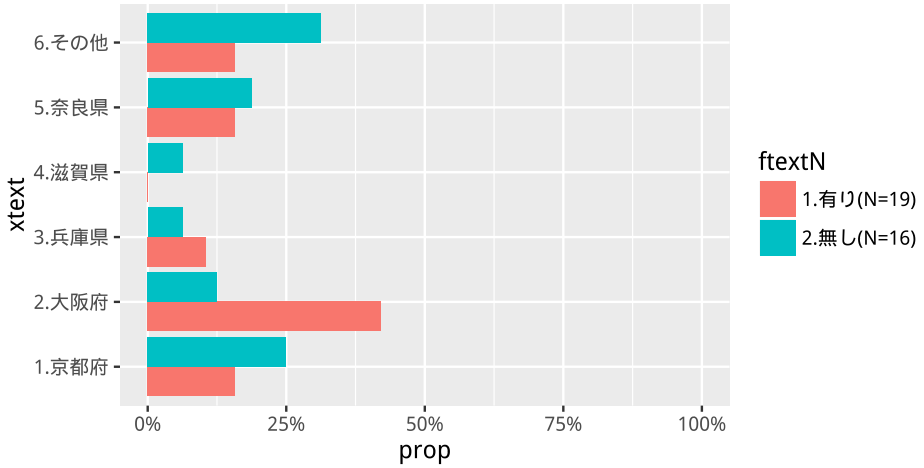
4 行目の「滋賀県」「有り」が 0 となっています。

グラフを描いていきましょう。まず、基本的な設定を定義します。ここは、これまで描いた横棒グラフと同じです。

```
> ggbarc<-ggplot(p03,
+             aes(x=xtext,                          #X の値を指定と順番を指定
+                 y=prop,                           #Y の値を指定
+                 fill=ftextN))+                    #塗り分ける変数を指定
+   coord_flip()+                                   #グラフを横向きに
+   scale_y_continuous(labels=percent,              #y 軸の目盛は%表記
+                      limits=c(0,1))               #y 軸の範囲は 0〜1
```

次に、グラフを描きます。

```
> ggbarc+geom_bar(stat="identity",position="dodge")
```



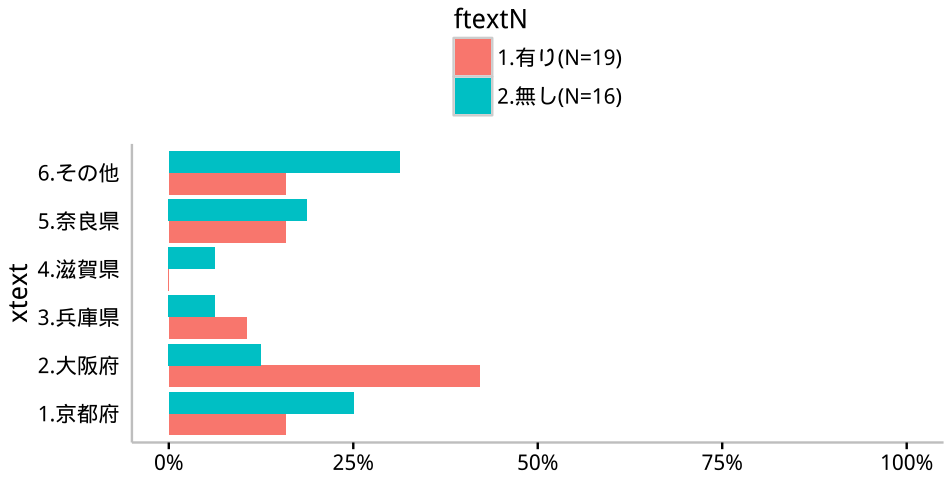
これまでとの違いは、引数 `position="dodge"`（ずらす）です。これがないと、デフォルトが`"stack"`なので、積み重ねグラフとなります。

フォーマットを変更しましょう。これも従来どおりです。

```
> gfbarc<-theme_bw()+                                  #白黒基調の組み込みテーマを使う
+   theme(panel.border=element_blank(),                 #描画領域の枠線を消す
+         panel.grid=element_blank(),                   #目盛線を消す
+         axis.title.x=element_blank(),                 #横軸タイトルを消す
+         axis.ticks.y=element_blank(),                 #縦軸目盛を消す
+         axis.line.x=element_line(colour="grey"),      #軸線を grey で書き足す
+         axis.line.y=element_line(colour="grey"),      #軸線を grey で書き足す
+         legend.position="top",                        #凡例を上に配置
+         legend.direction="vertical")                  #凡例を縦に並べる
```

再描画してみます。

```
> ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
```

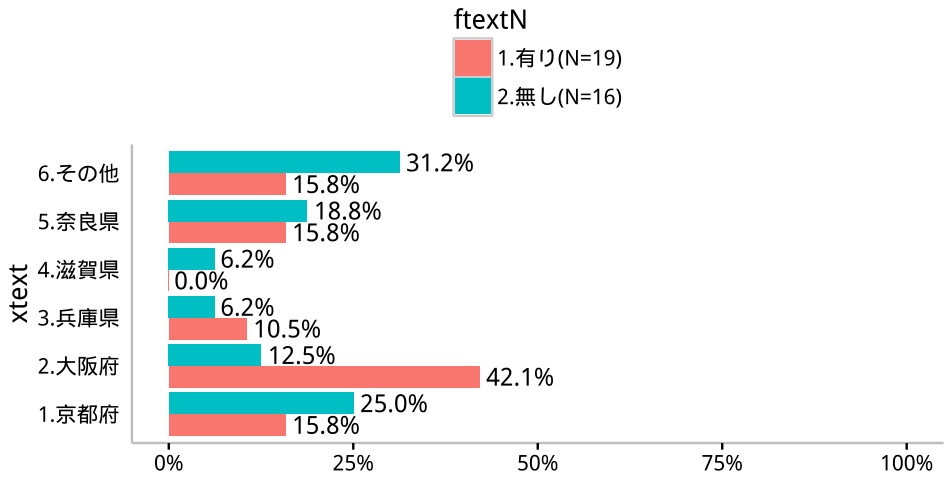


データラベルを追加しましょう。水平方向の位置は棒の右側なので、`y=prop` で位置を与えて、そこから少し水平方向にずらせばよいです。これは `hjust=-0.1` で指定しています。縦位置は、棒のずれに応じてずらす（dodge）必要があります。

```
> ggbarc<-ggplot(p03,
+             aes(x=xtext,                          #X の値と順番を指定
+                 y=prop,                           #Y の値を指定
+                 fill=ftextN))+                    #塗り分ける変数を指定
+   coord_flip()+                                   #グラフを横向きに
+   scale_y_continuous(labels=percent,              #y 軸の目盛は%表記
+                      limits=c(0,1))+              #y 軸の範囲は 0〜1
+   geom_text(                                      #データラベルを記入
+     aes(y=prop,                                   #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),        #ラベルは prop*100 を 00.0% で表記
+     hjust=-0.1,                                   #ラベルの位置調整 0.1 右に
+     position=position_dodge(0.9))                 #ラベル t の位置調整  縦方向微調整
```

グラフを確認してみましょう。

```
> ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
```



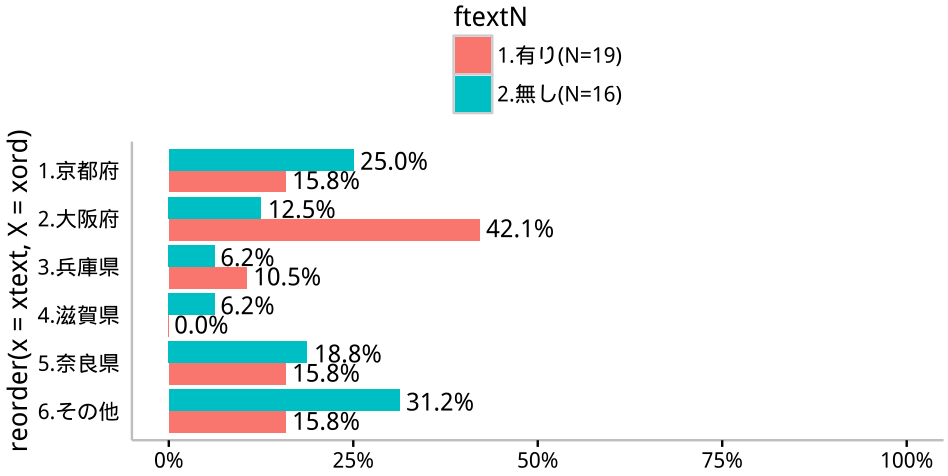
X 軸の並び替えをしましょう。デフォルトでは下から上に積み重ねられているので、これを逆にします。p03 の変数 **xtext** の順位 (order) にマイナスを掛けて、逆の順番にします。

```
> (xord<--rank(p03$xtext))
 [1]  -1.5  -3.5  -5.5  -7.5  -9.5 -11.5  -1.5  -3.5  -5.5  -7.5  -9.5 -11.5
```

グラフの X の指定を変更します。

```
> ggbarc<-ggplot(p03,
+             aes(x=reorder(x=xtext,X=xord),        #X の値と順番を指定
+                 y=prop,                           #Y の値を指定
+                 fill=ftextN))+                    #塗り分ける変数を指定
+   coord_flip()+                                   #グラフを横向きに
+   scale_y_continuous(labels=percent,              #y 軸の目盛は%表記
+                      limits=c(0,1))+              #y 軸の範囲は 0〜1
+   geom_text(                                      #データラベルを記入
+     aes(y=prop,                                   #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),        #ラベルは prop*100 を 00.0% で表記
+     hjust=-0.1,                                   #ラベルの位置調整 0.1 右に
+     position=position_dodge(0.8))                 #ラベル t の位置調整  縦方向微調整
```

```
> ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
```



X 軸の並びを大きい順に変更しましょう。棒が 2 つずつあるのでなにをもって「大きい順」かわかりませんが、p03 の prop の値をそのまま順番として与えてみました。

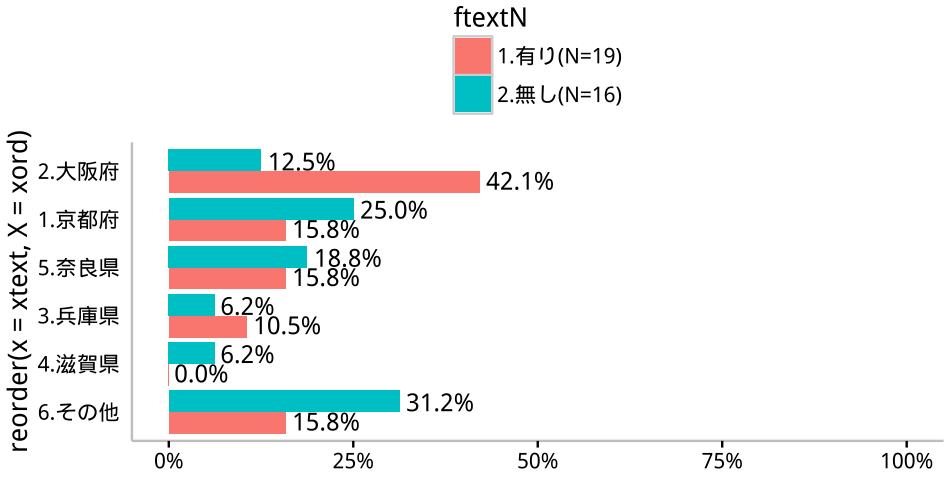
```
> (xord<-p03$prop)
 [1] 0.1578947 0.4210526 0.1052632 0.0000000 0.1578947 0.1578947 0.2500000 0.1250000
 [9] 0.0625000 0.0625000 0.1875000 0.3125000
```

ただし、「その他」は最後に持ってきたいので、一番小さいマイナスの値を与えておきます。

```
> xord[grepl("その他",p03$xtext)]<- -1              #「その他」は一番小さい値
> xord
 [1]  0.1578947  0.4210526  0.1052632  0.0000000  0.1578947 -1.0000000  0.2500000  0.1250000
 [9]  0.0625000  0.0625000  0.1875000 -1.0000000
```

グラフを確認します。

```
> ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
```



けっこううまくいっていいるように思うので、これでよしとしましょう。どうも 2 つの prop の合計で並び替えているように思いますが・・・

あと気になるのは、「1. 有り (N=19)」の棒が「2. 無し (N=16)」の棒の下に位置しています。これをひっくり返しましょう。現段階では、p03 の変数 ftextN の Levels は以下のようになっています。

```
> (yord<-levels(p03$ftextN))
[1] "1. 有り (N=19)" "2. 無し (N=16)"
```

この順序が逆になるように、factor() 関数を使って levels(p03$ftextN) を再定義しましょう。transform() 関数で、データフレームの特定の変数をいじることもできます。

```
> p03<-transform(p03,ftextN=factor(ftextN,levels=rev(yord)))
```

これで Levels の順番がひっくり返りました。

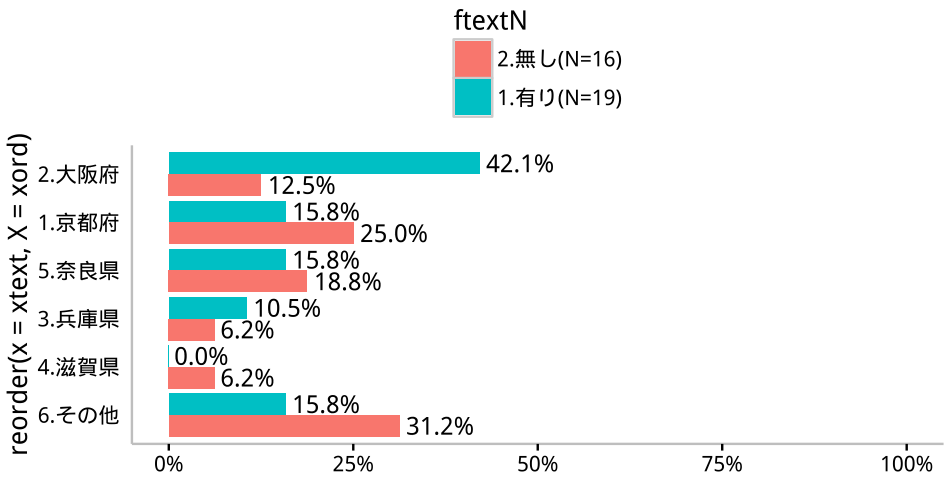
```
> levels(p03$ftextN)
[1] "2. 無し (N=16)" "1. 有り (N=19)"
```

ggbarc を再び読み込んで・・・

```
> ggbarc<-ggplot(p03,
+              aes(x=reorder(x=xtext,X=xord),        #X の値と順番を指定
+                  y=prop,                           #Y の値を指定
+                  fill=ftextN))+                    #塗り分ける変数を指定
+   coord_flip()+                                    #グラフを横向きに
+   scale_y_continuous(labels=percent,               #y 軸の目盛は%表記
+                      limits=c(0,1))+               #y 軸の範囲は 0〜1
+   geom_text(                                       #データラベルを記入
+     aes(y=prop,                                    #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),         #ラベルは prop*100 を 00.0% で表記
+     hjust=-0.1,                                    #ラベルの位置調整 0.1 右に
+     position=position_dodge(0.9))                  #ラベル t の位置調整  縦方向微調整
```

描画すれば、棒の並びが逆転しているはずです。

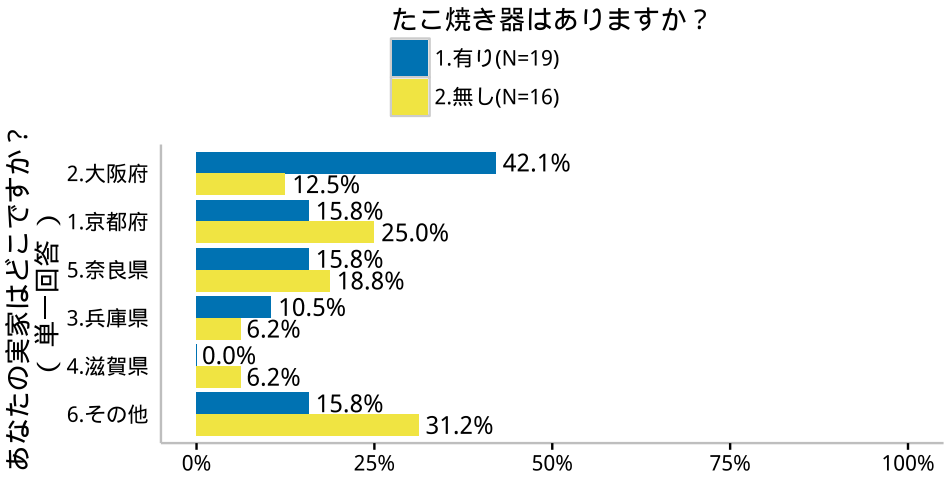
```
> ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
```



あとは凡例の並び替え、X 軸のタイトル、分離 f のタイトル、棒の色を変更して完成です。

```
> #グラフの定義
> ggbarc<-ggplot(p03,
+             aes(x=reorder(x=xtext,X=xord),        #X の値と順番を指定
+                 y=prop,                           #Y の値を指定
+                 fill=ftextN))+                    #塗り分ける変数を指定
+   coord_flip()+                                   #グラフを横向きに
+   scale_y_continuous(labels=percent,              #y 軸の目盛は%表記
+                      limits=c(0,1))+              #y 軸の範囲は 0〜1
+   geom_text(                                      #データラベルを記入
+     aes(y=prop,                                   #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),        #ラベルは prop*100 を 00.0% で表記
+     hjust=-0.1,                                   #ラベルの位置調整 0.1 右に
+     position=position_dodge(0.9))+                #ラベル t の位置調整  縦方向微調整
+   guides(fill=guide_legend(reverse = TRUE))+      #凡例の並び替え
+   xlab(lab_x)+                                    #X 軸のラベルを指定
+   labs(fill=d03q_f)+                              #凡例のラベルを指定
+   scale_fill_manual(values=cb_palette[na_f0:1])   #棒の色


> ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
```



3.3　複数回答×単一回答の棒グラフ

複数回答のクロス集計もやってみましょう。複数回答の集計は、各選択肢の指摘率を求めることになりますが、これを他の単一回答の設問のへの回答の違いで比較することになります。横棒グラフで描いたほうが見やすいかもしれません。この場合、X が複数回答、分類 f が単一回答、各 X の指摘率が Y になります。例として、複数回答の Q05（優柔不断かどうかを聞いた設問）を X に、Q02（たこ焼き器の有無）を f に指定してみましょう。あまり意味のあるクロス集計ではありませんが、勘弁。

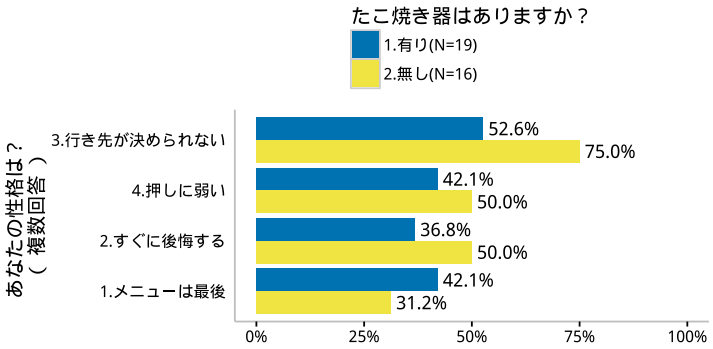


図 3.4: 単一回答×単一回答の帯グラフの例

3.3.1　データの抽出

クロス集計に使う設問の番号を指定します。
データの抽出は前節までと同じです。

```
> qNo_x<-5
> qNo_f<-2
```

選択した設問の回答データ（設問番号で入力）を抽出します。

```
> #データ抽出
> d03r_x<-dplyr::select(d01r,contains(qID[qNo_x])) #X 軸にする設問の回答データ
> d03r_f<-dplyr::select(d01r,contains(qID[qNo_f])) #Y 軸にする設問の回答データ
```

無回答が含まれるデータは削除します。

```
> #無回答の処理
> d03r<-data.frame(d03r_x,d03r_f)                    #2 つの変数を結合したデータフレーム
> d03r<-na.omit(d03r)                                #NA を含む行を削除
> d03r_x<-dplyr::select(d03r,contains(qID[qNo_x])) #X 軸にする設問の回答データ
> d03r_f<-dplyr::select(d03r,contains(qID[qNo_f])) #Y 軸にする設問の回答データ
```

その他の情報も抽出します。

```
> d03q_x<-d01q[qNo_x,"question"]                    #設問文
> d03q_f<-d01q[qNo_f,"question"]
> d03ord_x<-d01q[qNo_x,"order"]                    #選択肢の並べ方
> d03ord_f<-d01q[qNo_f,"order"]
> d03a_x0<-d01a[,qID[qNo_x]]                       #選択肢の文言
> d03a_f0<-d01a[,qID[qNo_f]]
> d03a_x0<-d03a_x0[d03a_x0!=""]                    #選択肢から空白を削除
> d03a_f0<-d03a_f0[d03a_f0!=""]
> if(is.factor(d03a_x0)){                          #（もし因子だったら・・・）
+   d03a_x0<-droplevels(d03a_x0)                   #選択肢から空白のレベルを削除
+ }
> d03a_f0<-droplevels(d03a_f0)
> na_x0<-length(d03a_x0)                           #選択肢の数をカウント
> na_f0<-length(d03a_f0)
> lab_x<-paste(d03q_x,"\n（",d03gk_x,"）")#X 軸ラベル
```

3.3.2　データセットの作成

f（分類値）ごとの X（複数回答）の指摘率を計算

d03_r には、複数回答の設問（Q05）に対する回答は複数列にわたって入力してあります。こんな感じ。

```
> head(d03r)
  Q05_01 Q05_02 Q05_03 Q05_04 Q02
1      0      0      1      1   1
2      0      0      1      1   1
3      1      1      0      0   1
4      1      0      0      0   1
5      0      1      1      1   1
6      0      1      0      0   2
```

これを 1 つのデータとするには、d03r の X 軸の方のデータを縦に並べます。これにはライブラリ resahpe2 の `melt()` 関数を使います。

```
> d03rm<-melt(data=d03r,id.vars=qID[qNo_f])
```

引数 `data=d03r` で縦に並べるデータフレームを指定しますが、引数 `id.vars=`で指定した変数についてはそのまんまいじりません。とはいえ、この「縦に並べる」というのがちょっとわかりにくいかもしれません。

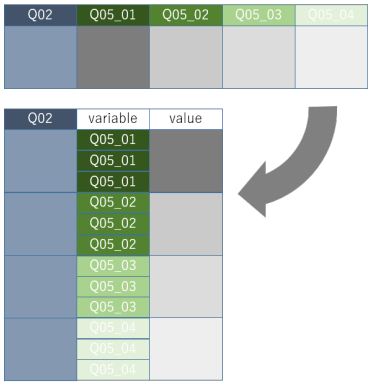


図 3.5: ライブラリ reshape2 の melt 関数でデータフレームを縦に並べる

35 行の d03_r を縦に並べ替えた d03_rm の最初の 6 行はこんな感じです。

```
> d03rm[1:6,]
  Q02 variable value
1   1   Q05_01     0
2   1   Q05_01     0
3   1   Q05_01     1
4   1   Q05_01     1
5   1   Q05_01     0
6   2   Q05_01     0
```

変数名 `Q05_01` が、`variable` という変数名の下にデータとして並んでいます。そして、その横に `value` という変数名で、もともと変数 `Q05_01` のデータだった値が並んでいます。

続く 36 行目から 6 行分を表示すると、こんな感じです。

```
> d03rm[36:41,]
```

```
   Q02 variable value
36   1   Q05_02     0
37   1   Q05_02     0
38   1   Q05_02     1
39   1   Q05_02     0
40   1   Q05_02     1
41   2   Q05_02     1
```

変数名 variable のデータが Q05_02 に変わり、その横の変数 value には、もともと変数 Q05_02 のデータだった値が並んでいます。

変数 Q02 は引数 id.vars=で指定してあるので、そのまんま残されています。

このように、5 列 35 行だった d03_r が、melt() 関数で 3 列の $35 \times 4 = 140$ 行に変わりました。これでライブラリ plyr の count() 関数が使えます。

集計します。ライブラリ plyr の count() ではなく、あえて table() 関数を使います。

```
> (t03_0m<-table(d03rm))                    #クロス集計
, , value = 0

   variable
Q02 Q05_01 Q05_02 Q05_03 Q05_04
  1     11     12      9     11
  2     11      8      4      8

, , value = 1

   variable
Q02 Q05_01 Q05_02 Q05_03 Q05_04
  1      8      7     10      8
  2      5      8     12      8
```

これで d02, variable, value の 3 変数のクロス集計ができました。ggplot2 で扱えるデータにするため、縦に展開したデータフレームに変形します。

```
> t03_m<-melt(t03_0m)                       #クロス集計結果を盾に並べる
```

変数名も変更します。

```
> colnames(t03_m)<-c(qID[qNo_f],
+                    "variable","value","freq") #変数名を変更
```

最初の方はこんな感じです。

```
> head(t03_m)
  Q02 variable value freq
1   1   Q05_01     0   11
2   2   Q05_01     0   11
3   1   Q05_02     0   12
4   2   Q05_02     0    8
5   1   Q05_03     0    9
6   2   Q05_03     0    4
```

1 行目を解説すると、設問 Q02 で 1 と回答した人のうち、設問 Q05 の最初の選択肢（Q05_01）に○をつけなかった人（変数 value が 0）の数は 11 人です。

Q02 と Q05 の設問ごとに構成比を求めます。（qID[qNo_f] は、この例では"002"のことですね。念のため）

```
> (p03_m0<-ddply(t03_m,c(qID[qNo_f],"variable"),
+                transform,prop=freq/sum(freq)))  #構成比
   Q02 variable value freq      prop
1    1   Q05_01     0   11 0.5789474
2    1   Q05_01     1    8 0.4210526
3    1   Q05_02     0   12 0.6315789
4    1   Q05_02     1    7 0.3684211
5    1   Q05_03     0    9 0.4736842
6    1   Q05_03     1   10 0.5263158
7    1   Q05_04     0   11 0.5789474
8    1   Q05_04     1    8 0.4210526
9    2   Q05_01     0   11 0.6875000
10   2   Q05_01     1    5 0.3125000
11   2   Q05_02     0    8 0.5000000
12   2   Q05_02     1    8 0.5000000
13   2   Q05_03     0    4 0.2500000
14   2   Q05_03     1   12 0.7500000
15   2   Q05_04     0    8 0.5000000
16   2   Q05_04     1    8 0.5000000
```

ddply() 関数は、2 番目の引数で指定した変数の値ごとに、そのあとの引数でしてした関数を当てはめるわけですが、c("Q02","variable") というように、2 つの変数を指定することもできます。ここでは、transform() 関数で、データフレームにその次の引数で指定した構成比を求める計算をして、prop という名前でくっつけろと指示しています。これが、変数 Q02 と variable の値ごとにあてはめられています。

しかし、欲しい値は Q05 で指摘した人（変数 value の値が 1）の割合だけで、指摘しなかった人（変数 value の値が 0）の割合は不要です。データフレーム p03_m0 から変数 value の値が 1 の行だけを抽出します。これにはいろいろな方法がありますが、ライブラリ dplyr の filter() 関数が簡単です。

```
> (p03_m1<-dplyr::filter(p03_m0,value==1))        #指摘率だけ残す
  Q02 variable value freq      prop
1   1   Q05_01     1    8 0.4210526
2   1   Q05_02     1    7 0.3684211
3   1   Q05_03     1   10 0.5263158
4   1   Q05_04     1    8 0.4210526
5   2   Q05_01     1    5 0.3125000
6   2   Q05_02     1    8 0.5000000
7   2   Q05_03     1   12 0.7500000
8   2   Q05_04     1    8 0.5000000
```

これで指摘率の計算ができました。

分類 f の選択肢番号を N=付き選択肢の文言に変換

分類 f の選択肢番号と選択肢の文言 (N=付き）との参照表を作成するところまでは前節と同じです。

```
> #回答データ（選択肢番号で入力）選択肢の文言に変換
> d03a_f<-data.frame(1:na_f0,d03a_f0)              #選択肢の番号と文言の参照表
> colnames(d03a_f)<-c(qID[qNo_f],"ftext")          #参照表に変数名をつける
> d03_f<-join(d03r_f,d03a_f,by=qID[qNo_f])         #回答データ（番号）を文言に変換
> #f の文言を（N=付き）にする
> t03_f<-plyr::count(d03_f)                        #集計して N=を求める
```

```
> d03_fN<-transform(t03_f,
+    ftextN=paste(ftext,"(N=",freq,")",sep=""))  #(N=) を X の文言に付ける
> (d03a_fN<-subset(d03_fN,
+                  select=c(qID[qNo_f],"ftextN")))#選択肢の番号と N=付き文言の参照表
  Q02       ftextN
1   1 1. 有り (N=19)
2   2 2. 無し (N=16)
>
```

この参照表を使って、さきほど計算した指摘率の表の分類 f に使っている変数（この場合 Q02）の選択肢番号を選択肢の文言（N=付き）に変換します。

```
> (p03_m2<-join(p03_m1,d03a_fN,by=qID[qNo_f]))   #回答データ（番号）を N=付き文言へ
  Q02 variable value freq      prop       ftextN
1   1   Q05_01     1    8 0.4210526 1. 有り (N=19)
2   1   Q05_02     1    7 0.3684211 1. 有り (N=19)
3   1   Q05_03     1   10 0.5263158 1. 有り (N=19)
4   1   Q05_04     1    8 0.4210526 1. 有り (N=19)
5   2   Q05_01     1    5 0.3125000 2. 無し (N=16)
6   2   Q05_02     1    8 0.5000000 2. 無し (N=16)
7   2   Q05_03     1   12 0.7500000 2. 無し (N=16)
8   2   Q05_04     1    8 0.5000000 2. 無し (N=16)
```

X の選択肢番号を N=付き選択肢の文言に変換

続いて、X に使う変数 variable の Q05_01、Q05_02、... も選択肢の文言に変換します。

選択肢の番号と文言の参照表を作成します。

```
> (d03a_x<-data.frame(variable=colnames(d03r_x),xtext=d03a_x0))
  variable                  xtext
1   Q05_01       1. メニューは最後
2   Q05_02       2. すぐに後悔する
3   Q05_03 3. 行き先が決められない
4   Q05_04           4. 押しに弱い
```

指摘率の表を変数 variable を選択肢の文言で変換します。

```
> (p03<-join(p03_m2,d03a_x,by="variable"))
  Q02 variable value freq      prop       ftextN                  xtext
1   1   Q05_01     1    8 0.4210526 1. 有り (N=19)       1. メニューは最後
2   1   Q05_02     1    7 0.3684211 1. 有り (N=19)       2. すぐに後悔する
3   1   Q05_03     1   10 0.5263158 1. 有り (N=19) 3. 行き先が決められない
4   1   Q05_04     1    8 0.4210526 1. 有り (N=19)           4. 押しに弱い
5   2   Q05_01     1    5 0.3125000 2. 無し (N=16)       1. メニューは最後
6   2   Q05_02     1    8 0.5000000 2. 無し (N=16)       2. すぐに後悔する
7   2   Q05_03     1   12 0.7500000 2. 無し (N=16) 3. 行き先が決められない
8   2   Q05_04     1    8 0.5000000 2. 無し (N=16)           4. 押しに弱い
```

これで、データセットができました。

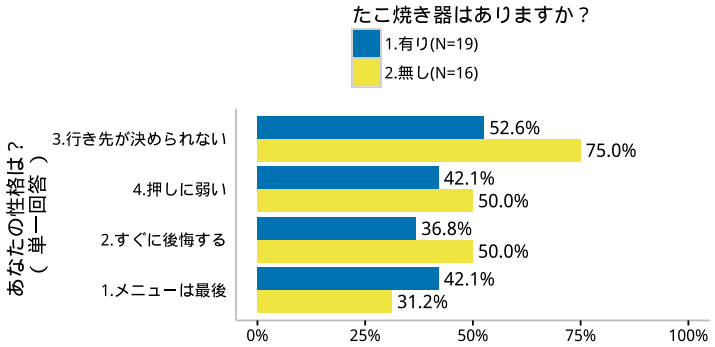
3.3.3 グラフを描く

作成した p03 は、前節で作成した棒グラフのデータがすべてそろっているので、前節の手順がそのまま使えます。

```
> #X の並びを指摘率が大きい順に
> xord<-p03$prop                                    #X は prop の大きさで並べる
> #横棒グラフを上から下に描くための工夫
> yord<-levels(p03$ftextN)
> p03<-transform(p03,
+                ftextN=factor(ftextN,levels=rev(yord))) #Y の順番を逆転
> #グラフの定義
> ggbarc<-ggplot(p03,
+             aes(x=reorder(x=xtext,X=xord),        #X の値と順番を指定
+                 y=prop,                           #Y の値を指定
+                 fill=ftextN))+                    #塗り分ける変数を指定
+   coord_flip()+                                   #グラフを横向きに
+   scale_y_continuous(labels=percent,              #y 軸の目盛は%表記
+                      limits=c(0,1))+              #y 軸の範囲は 0〜1
+   geom_text(                                      #データラベルを記入
+     aes(y=prop,                                   #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),        #ラベルは prop*100 を 00.0% で表記
+     hjust=-0.1,                                   #ラベルの位置調整 0.1 右に
+     position=position_dodge(0.9))+                #ラベル t の位置調整  縦方向微調整
+   guides(fill=guide_legend(reverse = TRUE))+      #凡例の並び替え
+   xlab(lab_x)+                  #X 軸のラベルを指定
+   labs(fill=d03q_f)+                              #凡例のラベルを指定
+   scale_fill_manual(values=cb_palette[na_f0:1])   #棒の色
```

描画します。グラフのフォーマットは前節で定義した gfbarc をそのまま使います。

```
> ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
```



できあがりです。

3.4　数値記入項目×単一回答の棒グラフ

数値記入項目といっても、階級を与えて単一回答に変換するので、その手順を除けば、あとは単一回答×単一回答の棒グラフと同じです。Q04（いくら以上なら交番に届けますか？）と Q02（たこ焼き器はありますか？）のクロス集計をやってみましょう。

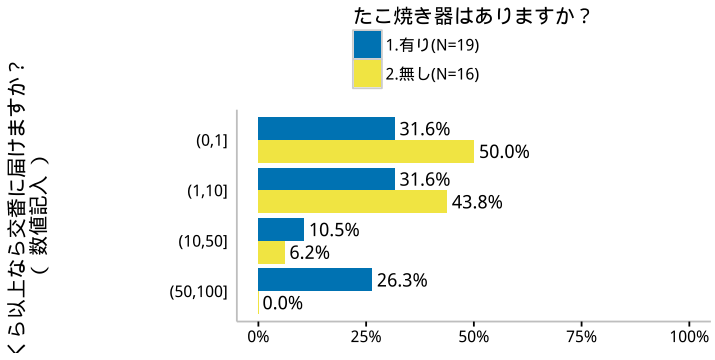


図 3.6: 単一回答×単一回答の帯グラフの例

3.4.1　データの抽出

データの抽出については、ここも前と同じです。まず、クロス集計に用いる設問番号を指定します。

```
> qNo_x<-4
> qNo_f<-2
```

選択した設問の回答データ（設問番号で入力）を抽出します。

```
> #データ抽出
> d03r_x<-dplyr::select(d01r,contains(qID[qNo_x])) #X 軸にする設問の回答データ
> d03r_f<-dplyr::select(d01r,contains(qID[qNo_f])) #Y 軸にする設問の回答データ
> #無回答の処理
> d03r<-data.frame(d03r_x,d03r_f)                   #2 つの変数を結合したデータフレーム
> d03r<-na.omit(d03r)                               #NA を含む行を削除
> d03r_x<-dplyr::select(d03r,contains(qID[qNo_x])) #X 軸にする設問の回答データ
> d03r_f<-dplyr::select(d03r,contains(qID[qNo_f])) #Y 軸にする設問の回答データ
> #その他の情報の抽出
> d03q_x<-d01q[qNo_x,"question"]                    #設問文
> d03q_f<-d01q[qNo_f,"question"]
> d03ord_x<-d01q[qNo_x,"order"]                     #選択肢の並べ方
> d03ord_f<-d01q[qNo_f,"order"]
> d03a_x0<-d01a[,qID[qNo_x]]                        #選択肢の文言
> d03a_f0<-d01a[,qID[qNo_f]]
> d03a_x0<-d03a_x0[d03a_x0!=""]                     #選択肢から空白を削除
> d03a_f0<-d03a_f0[d03a_f0!=""]
> if(is.factor(d03a_x0)){                           #（もし因子だったら・・・）
+   d03a_x0<-droplevels(d03a_x0)                    #選択肢から空白のレベルを削除
+ }
> d03a_f0<-droplevels(d03a_f0)
> na_x0<-length(d03a_x0)                            #選択肢の数をカウント
> na_f0<-length(d03a_f0)
> lab_x<-paste(d03q_x,"\n（",d03gk_x,"）")#X 軸ラベル
```

3.4.2　データセットの作成

X の回答データ (数値で入力）を指定した階級に割り振る

数値記入項目で、単一回答×単一回答の棒グラフと異なるのはここだけです。

```
> d03r_x1<-d03r_x[,1]                                  #回答の値をベクトルに
> xtext<-cut(d03r_x1,d03a_x0)                          #階級の区切りで回答の値を割り振る
```

分類 f の回答データも文言に変換して構成比を求める

残りは同じです。

```
> #f の回答データ（選択肢番号で入力）を選択肢の文言へ変換
> d03a_f<-data.frame(1:na_f0,d03a_f0)                  #選択肢の番号と文言の対応表
> colnames(d03a_f)<-c(qID[qNo_f],"ftext")              #参照表に変数名をつける
> d03_f<-join(d03r_f,d03a_f,by=qID[qNo_f])             #回答データ（番号）を文言へ変換
> #f の文言を（N=付き）にする
> t03_f<-plyr::count(d03_f)                            #集計して N=を求める
> d03_fN<-transform(t03_f,
+   ftextN=paste(ftext,"(N=",freq,")",sep="")) #(N=) を X の文言に付ける
> d03a_fN<-subset(d03_fN,
+                 select=c(qID[qNo_f],"ftextN"))#選択肢の番号と N=付き文言の参照表
> d03_fN<-join(d03r_f,d03a_fN,by=qID[qNo_f])           #回答データ（番号）を N=付き文言へ
> #データセットの作成
> d03<-data.frame(xtext,ftextN=d03_fN$ftextN)          #X（階級名）と f（N=付き文言）のデータ
> t03_0<-table(d03)                                    #クロス集計
> t03<-melt(t03_0)                                     #クロス集計結果を縦に並べ替える
> colnames(t03)<-c("xtext","ftextN","freq")            #変数名を変更
> p03<-ddply(t03,"ftextN",
+            transform,prop=freq/sum(freq))            #構成比を求める
```

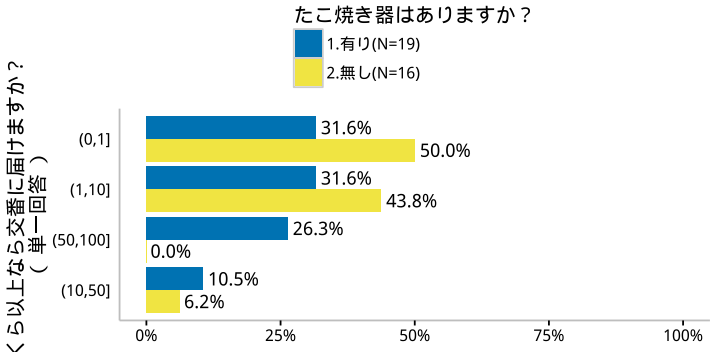
3.4.3　グラフの描画

```
> #X の並びを指摘率が大きい順に
> xord<-p03$prop                                       #X は prop の大きさで並べる
> #横棒グラフを上から下に描くための工夫
> yord<-levels(p03$ftextN)
> p03<-transform(p03,
+                ftextN=factor(ftextN,levels=rev(yord))) #Y の順番を逆転
> #グラフの定義
> ggbarc<-ggplot(p03,
+             aes(x=reorder(x=xtext,X=xord),           #X の値と順番を指定
+                 y=prop,                              #Y の値を指定
+                 fill=ftextN))+                       #塗り分ける変数を指定
+   coord_flip()+                                      #グラフを横向きに
+   scale_y_continuous(labels=percent,                 #y 軸の目盛は%表記
+                      limits=c(0,1))+                 #y 軸の範囲は 0〜1
+   geom_text(                                         #データラベルを記入
+     aes(y=prop,                                      #ラベルの位置
+         label=sprintf("%.1f%%",prop*100)),           #ラベルは prop*100 を 00.0% で表記
+     hjust=-0.1,                                      #ラベルの位置調整 0.1 右に
+     position=position_dodge(0.9))+                   #ラベル t の位置調整  縦方向微調整
```

```
+    guides(fill=guide_legend(reverse = TRUE))+      #凡例の並び替え
+    xlab(lab_x)+                                    #X 軸のラベルを指定
+    labs(fill=d03q_f)+                              #凡例のラベルを指定
+    scale_fill_manual(values=cb_palette[na_f0:1])  #棒の色
```

描画します。グラフのフォーマットは、前の節で定義した gfbarc そのままです。

```
> ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
```



3.5　関数化する

3.5.1　関数

以上、４種類のグラフを描きましたが、これを設問の種類に応じて描き分けられるように関数化しましょう。データの読み込み等は共用していますが、あとはほとんど４種類別々に定義して、swich() 関数で分岐させているだけです。X や f の回答データを文言に変換するところなどを共用して、もう少しエレガントにすっきりしたものに書き換えられるような気もしますが、こんなところで勘弁してください。

定義した関数とその流れ以下です。

1. ライブラリの読み込みとカラーパレットの定義
2. 棒グラフの描画部分の関数定義：fgcross_bar()
3. 帯グラフの描画部分の関数定義：fgcross_obi()
4. 単一回答×単一回答の棒グラフを描く関数定義： fgcross_ss3()
5. 単一回答×単一回答の帯グラフを描く関数定義： fgcross_ss2()
6. 複数回答×単一回答の棒グラフを描く関数定義： fgcross_ms()
7. 数値記入×単一回答の棒グラフを描く関数定義： fgcross_ns()
8. ４種類のグラフを描き分ける関数定義：fgcross()

```
#ライブラリの読み込み
library(ggplot2)
library(plyr)
library(scales)
library(dplyr)
library(reshape2)
library(showtext)
library(grid)
```

```
#カラーパレットの作成
cb_palette<-c("#0072B2","#F0E442","#009E73","#56B4E9",
              "#CC79A7","#D55E00","#E69F00","#999999")

######## グラフ描画の共用部分 ########

#-------------------------------------------------------------------------------

# 棒グラフ描画

fgcross_bar<-function(propd=p03,plt=plt,
                      qID, dordx=d03ord_x,
                      lab_x=lab_x,lab_f=d03q_f){
  #x 軸の並べ方を指定
  if(dordx=="同"){
    xord<- -rank(propd$xtext)                          #もし「同」なら Levels の逆順で
  }else if(dordx=="大"){
    xord<-propd$prop                                   #もし「大」なら prop の大きさで
    xord[grepl("その他",propd$xtext)]<- -1             #「その他」は一番下へ
  }else{
    xord<-NULL
  }

  #横棒グラフを上から下に描くための工夫
  yord<-levels(propd$ftextN)
  p03<-transform(propd,
                 ftextN=factor(ftextN,levels=rev(yord))) #Y の順番を逆転

  #グラフの定義
  ggbarc<-ggplot(p03,
                 aes(x=reorder(x=xtext,X=xord),        #X の値と順番を指定
                     y=prop,                           #Y の値を指定
                     fill=ftextN))+                    #塗り分ける変数を指定
    coord_flip()+                                      #グラフを横向きに
    scale_y_continuous(labels=percent,                 #y 軸の目盛は%表記
                       limits=c(0,1))+                 #y 軸の範囲は 0〜1
    geom_text(                                         #データラベルを記入
      aes(y=prop,                                      #ラベルの位置
          label=sprintf("%.1f%%",prop*100)),           #ラベルは prop*100 を 00.0% で表記
      hjust=-0.1,                                      #ラベルの位置調整 0.1 右に
      position=position_dodge(0.9))+                   #ラベル t の位置調整 縦方向微調整
    guides(fill=guide_legend(reverse = TRUE))+         #凡例の並び替え
    xlab(lab_x)+                                       #X 軸のラベルを指定
    labs(fill=lab_f)+                                  #凡例のラベルを指定
    scale_fill_manual(values=plt)                      #棒の色

  #棒グラフ用フォーマット
  gfbarc<-theme_bw()+                                  #白黒基調の組み込みテーマを使う
    theme(panel.border=element_blank(),                #描画領域の枠線を消す
          panel.grid=element_blank(),                  #目盛線を消す
          legend.position="top",                       #凡例をトップに
          legend.direction="vertical",                 #凡例を縦に並べる
          axis.title.x=element_blank(),                #横軸タイトルを消す
          axis.ticks.y=element_blank(),                #縦軸目盛を消す
          axis.line.x=element_line(colour="grey"),     #軸線を grey で書き足す
          axis.line.y=element_line(colour="grey"))     #軸線を grey で書き足す

  #グラフの描画
  ggbarc+gfbarc+geom_bar(stat="identity",position="dodge")
}

#-------------------------------------------------------------------------------
```

```
# 帯グラフ描画

fgcross_obi<-function(propd=p03,plt=plt,
                      qID,dordx=d03ord_x,alt1=d03a_f0[1],
                      lab_x=lab_x,lab_f=d03q_f){
  #x 軸の並べ方を指定
  if(dordx=="同"){
    xord<- -rank(propd$xtextN)                          #もし「同」なら Levels の逆順で
  }else if(dordx=="大"){
    xord<-propd$prop                                    #もし「大」なら prop の大きさで、
    xord[propd$ftext!=alt1]<-0                          #f の 1 番め選択肢の prop のみ採用
    xord[grepl("その他",propd$xtextN)]<- -1            #「その他」は一番下へ
  }else{
    xord<-NULL
  }

  #グラフの定義 帯グラフ
  ggobic<-ggplot(propd,
                 aes(x=reorder(x=xtextN,X=xord),        #X の値と順番を指定
                     y=prop,                            #Y の値を指定
                     fill=ftext),                       #塗り分ける変数を指定
                 position="fill")+                      #100 %積上げグラフ
    coord_flip()+                                       #グラフを横向きに
    xlab(lab_x)+                                        #X 軸のラベルを指定
    labs(fill=lab_f)                                    #凡例のラベルを指定


  #帯グラフ用フォーマット
  gfobic<-theme_bw()+                                  #theme_bw を使用
    theme(panel.border=element_blank(),                 #パネルの枠線を消す
          panel.grid.major=element_blank(),             #目盛線を消す
          axis.ticks=element_blank(),                   #軸の目盛を消す
          axis.text.x=element_blank(),                  #目盛ラベルを消す
          axis.title.x=element_blank(),                 #X 軸のタイトルを消す
          legend.position="top")                        #凡例を上に配置


  #描画  帯グラフ
  ggobic+gfobic+geom_bar(stat="identity",
                         colour="white")+               #線の色指定
    scale_fill_manual(values=plt)+                      #塗りつぶしの色指定
    geom_text(aes(y=lposition,                          #データら別の位置
                  label=sprintf("%.1f%%",prop*100)),#データラベルを%小数点 1 桁に
              color="white")                            #文字色を白に
}

##### 単一回答（棒、帯)、複数回答（棒)、数値回答（棒）の 4 種類のグラフ描画 ####

# 単一回答（X）の構成比（Y）の分布を単一回答（f）で分類： 棒グラフ
#-------------------------------------------------------------------------------
fgcross_ss3<-function(qNo_x,  qNo_f,
                      d03r_x, d03r_f,
                      d03q_x, d03q_f,
                      d03a_x0,d03a_f0,
                      na_x0,  na_f0,
                      qID,    d03ord_x,
                      lab_x,
                      plt){
  #X の回答データ（選択肢番号で入力）を選択肢の文言へ変換
  d03a_x<-data.frame(1:na_x0,d03a_x0)             #選択肢の番号と文言の参照表
  colnames(d03a_x)<-c(qID[qNo_x],"xtext")         #参照表に変数名をつける
  d03_x<-join(d03r_x,d03a_x,by=qID[qNo_x])        #回答データ（番号）を文言へ変換
```

```
  #f の回答データ（選択肢番号で入力）を選択肢の文言へ変換
  d03a_f<-data.frame(1:na_f0,d03a_f0)            #選択肢の番号と文言の参照表
  colnames(d03a_f)<-c(qID[qNo_f],"ftext")        #参照表に変数名をつける
  d03_f<-join(d03r_f,d03a_f,by=qID[qNo_f])       #回答データ（番号）を文言へ変換

  #f の文言を（N=付き）にする
  t03_f<-plyr::count(d03_f)                      #集計して N=を求める
  d03_fN<-transform(t03_f,
                    ftextN=paste(ftext,"(N=",freq,")",sep="")) #(N=) を X の文言に付ける
  d03a_fN<-subset(d03_fN,
                  select=c(qID[qNo_f],"ftextN"))#選択肢の番号と N=付き文言の参照表
  d03_fN<-join(d03r_f,d03a_fN,by=qID[qNo_f])     #回答データ（番号）を N=付き文言へ

  #データセット作成
  d03<-data.frame(xtext=d03_x[,"xtext"],
                  ftextN=d03_fN[,"ftextN"])      #X（文言）と f（N=付き文言）のデータ
  t03_0<-table(d03)                              #クロス集計
  t03<-melt(t03_0)                               #クロス集計結果を縦に並べ替え
  colnames(t03)<-c("xtext","ftextN","freq")      #変数名の変更
  p03<-ddply(t03,"ftextN",transform,
             prop=freq/sum(freq))                #構成比計算で、データセット完成

  #作図
  fgcross_bar(p03,plt=plt,qID=qID,
              dordx=d03ord_x,lab_x=lab_x,lab_f=d03q_f)
}

# 単一回答（X）ごとに分類（f）の構成比（Y）の分布を描く： 帯グラフ
#------------------------------------------------------------------------------
fgcross_ss2<-function(qNo_x,  qNo_f,
                      d03r_x, d03r_f,
                      d03q_x, d03q_f,
                      d03a_x0,d03a_f0,
                      na_x0,  na_f0,
                      qID,    d03ord_x,
                      lab_x,
                      plt){
  #X の回答データ（選択肢番号で入力）を選択肢の文言へ変換
  d03a_x<-data.frame(1:na_x0,d03a_x0)            #選択肢の番号と文言の参照表
  colnames(d03a_x)<-c(qID[qNo_x],"xtext")        #参照表に変数名をつける
  d03_x<-join(d03r_x,d03a_x,by=qID[qNo_x])       #回答データ（番号）を文言へ変換

  #X の文言を（N=付き）にする
  t03_x<-plyr::count(d03_x)                      #集計して N=を求める
  d03_xN<-transform(t03_x,
                    xtextN=paste(xtext,"(N=",freq,")",sep=""))  #(N=) を X の文言に付ける
  d03a_xN<-subset(d03_xN,
                  select=c(qID[qNo_x],"xtextN"))#選択肢の番号と N=付き文言の参照表
  d03_xN<-join(d03r_x,d03a_xN,by=qID[qNo_x])     #回答データ（番号）を N=付き文言へ

  #f の回答データ（選択肢番号で入力）を選択肢の文言へ変換
  d03a_f<-data.frame(1:na_f0,d03a_f0)            #選択肢の番号と文言の参照表
  colnames(d03a_f)<-c(qID[qNo_f],"ftext")        #参照表に変数名をつける
  d03_f<-join(d03r_f,d03a_f,by=qID[qNo_f])       #回答データ（番号）を文言へ変換

  #データセット作成
  d03<-data.frame(xtextN=d03_xN[,"xtextN"],
                  ftext=d03_f[,"ftext"])         #X（N=付き文言）と f（文言）のデータ
  t03<-plyr::count(d03)                          #集計
  p03<-ddply(t03,"xtextN",transform,
             prop=freq/sum(freq))                #構成比計算
  p03<-ddply(p03,"xtextN",transform,
             lposition=cumsum(prop)-prop/2)      #データラベルの位置計算
```

```
  #作図
  fgcross_obi(propd=p03,plt=plt,qID=qID,
              d03ord_x,d03a_f0[1],lab_x,d03q_f)
}

#-------------------------------------------------------------------------------

#  複数回答（X）の構成比（Y）の分布を単一回答（X）で分類： 棒グラフ

fgcross_ms<-function(qNo_f,
                     d03r,   d03r_x, d03r_f,
                     d03q_x, d03q_f,
                     d03a_x0,d03a_f0,
                     na_f0,
                     qID,   d03ord_x,
                     lab_x,
                     plt){
  #f（分類値）ごとの X（複数回答）の指摘率を計算
  d03rm<-melt(data=d03r,id.vars=qID[qNo_f])      #複数回答のデータを縦に並べる
  t03_0m<-table(d03rm)                           #クロス集計
  t03_m<-melt(t03_0m)                            #クロス集計結果を盾に並べる
  colnames(t03_m)<-c(qID[qNo_f],
                     "variable","value","freq") #変数名を変更
  p03_m0<-ddply(t03_m,c(qID[qNo_f],"variable"),
                transform,prop=freq/sum(freq))  #構成比
  p03_m1<-dplyr::filter(p03_m0,value==1)         #指摘率だけ残す

  #f の回答データ（選択肢番号で入力）を選択肢の文言に変換
  d03a_f<-data.frame(1:na_f0,d03a_f0)            #選択肢の番号と文言の参照表
  colnames(d03a_f)<-c(qID[qNo_f],"ftext")        #参照表に変数名をつける
  d03_f<-join(d03r_f,d03a_f,by=qID[qNo_f])       #回答データ（番号）を文言に変換

  #f の文言を（N=付き）にする
  t03_f<-plyr::count(d03_f)                      #集計して N=を求める
  d03_fN<-transform(t03_f,
                    ftextN=paste(ftext,"(N=",freq,")",sep="")) #(N=) を X の文言に付ける
  d03a_fN<-subset(d03_fN,
                  select=c(qID[qNo_f],"ftextN"))#選択肢の番号と N=付き文言の参照表
  p03_m2<-join(p03_m1,d03a_fN,by=qID[qNo_f])     #指摘率の表の番号を N=付き文言へ

  #X の値（複数回答の変数名 Q05_1,Q05_2, 等）が入っている）を選択肢の文言に変換
  d03a_x<-data.frame(variable=colnames(d03r_x),
                     xtext=d03a_x0)              #変数名と選択肢の文言の参照表
  p03<-join(p03_m2,d03a_x,by="variable")         #変数名を選択肢の文言に変換して完成

  #作図
  fgcross_bar(p03,plt=plt,qID=qID,
              dordx=d03ord_x,lab_x=lab_x,lab_f=d03q_f)
}

#-------------------------------------------------------------------------------

# 数値記入項目（X）の構成比（Y）の分布を単一回答（f）で分類：棒グラフ

fgcross_ns<-function(qNo_f,
                     d03r_x, d03r_f,
                     d03q_x, d03q_f,
                     d03a_x0,d03a_f0,
                     na_f0,
                     qID,
                     lab_x,
                     plt){
  #X の回答データ（数値で入力）を指定した階級に割り振る
  d03r_x1<-d03r_x[,1]                            #回答の値をベクトルに
```

```
  xtext<-cut(d03r_x1,d03a_x0)                          #階級の区切りで回答の値を割り振る

  #f の回答データ（選択肢番号で入力）を選択肢の文言へ変換
  d03a_f<-data.frame(1:na_f0,d03a_f0)                  #選択肢の番号と文言の対応表
  colnames(d03a_f)<-c(qID[qNo_f],"ftext")              #参照表に変数名をつける
  d03_f<-join(d03r_f,d03a_f,by=qID[qNo_f])             #回答データ（番号）を文言へ変換

  #f の文言を（N=付き）にする
  t03_f<-plyr::count(d03_f)                            #集計して N=を求める
  d03_fN<-transform(t03_f,
                    ftextN=paste(ftext,"(N=",freq,")",sep="")) #(N=) を X の文言に付ける
  d03a_fN<-subset(d03_fN,
                  select=c(qID[qNo_f],"ftextN"))#選択肢の番号と N=付き文言の参照表
  d03_fN<-join(d03r_f,d03a_fN,by=qID[qNo_f])     #回答データ（番号）を N=付き文言へ

  #データセットの作成
  d03<-data.frame(xtext,ftextN=d03_fN$ftextN)    #X（階級名）と f（N=付き文言）のデータ
  t03_0<-table(d03)                              #クロス集計
  t03<-melt(t03_0)                               #クロス集計結果を縦に並べ替える
  colnames(t03)<-c("xtext","ftextN","freq")      #変数名を変更
  p03<-ddply(t03,"ftextN",
             transform,prop=freq/sum(freq))      #構成比を求める

  #作図
  fgcross_bar(p03,plt=plt,qID=qID,
              dordx="同",lab_x=lab_x,lab_f=d03q_f)
}


#-------------------------------------------------------------------------------

##### メイン関数：クロス集計のグラフを描き分ける #######

# 設問番号は設問データ（例えば d01q）の行番号
# qNo_x： X 軸に使う設問の設問番号
# qNo_f：分類（塗り分け：fill）に使う設問の設問番号

fgcross<-function(dr,dq,da,qNo_x,qNo_f,obi=TRUE,plt=cb_palette){
  #設問番号の読み込み
  qID<-dq[,"qID"]

  #データ抽出
  d03r_x<-dplyr::select(dr,contains(qID[qNo_x])) #X 軸に使う設問（X）の回答データ
  d03r_f<-dplyr::select(dr,contains(qID[qNo_f])) #分類に使う設問（f）の回答データ

  #無回答の処理
  d03r<-data.frame(d03r_x,d03r_f)                  #2 つの変数を結合したデータフレーム
  d03r<-na.omit(d03r)                              #NA を含む行を削除（しちゃう！）
  d03r_x<-dplyr::select(d03r,contains(qID[qNo_x]))#X の回答データ（NA 無し）
  d03r_f<-dplyr::select(d03r,contains(qID[qNo_f]))#f の回答データ（NA 無し）

  #その他情報の抽出
  d03q_x<-dq[qNo_x,"question"]                     #設問文
  d03q_f<-dq[qNo_f,"question"]
  d03ord_x<-dq[qNo_x,"order"]                      #選択肢の並べ方
  d03ord_f<-dq[qNo_f,"order"]
  d03gk_x<-dq[qNo_x,"graph"]                       #グラフの種類
  d03gk_f<-dq[qNo_f,"graph"]
  #分類 f に用いる設問が単一回答以外の場合エラーを発生させる
  if(substr(d03gk_f,1,2)!="単一"){
    stop(message="qNo_f には単一回答の設問を指定してください")
  }
  d03a_x0<-da[,qID[qNo_x]]                         #選択肢の文言
  d03a_f0<-da[,qID[qNo_f]]
```

```
  d03a_x0<-d03a_x0[d03a_x0!=""]                    #選択肢から空白を削除
  d03a_f0<-d03a_f0[d03a_f0!=""]
  if(is.factor(d03a_x0)){                          #（もし因子だったら・・・）
    d03a_x0<-droplevels(d03a_x0)                   #選択肢から空白のレベルを削除
  }
  if(is.factor(d03a_f0)){                          #（もし因子だったら・・・）
    d03a_f0<-droplevels(d03a_f0)
  }
  na_x0<-length(d03a_x0)                           #選択肢の数をカウント
  na_f0<-length(d03a_f0)
  lab_x<-paste(d03q_x,"\n (",d03gk_x,") ")         #X 軸ラベル

  #グラフの種類を描き分ける
  switch(substr(d03gk_x,1,2),
         "単一"=if(obi){
           fgcross_ss2(qNo_x,  qNo_f,
                       d03r_x, d03r_f,
                       d03q_x, d03q_f,
                       d03a_x0,d03a_f0,
                       na_x0,  na_f0,
                       qID,    d03ord_x,
                       lab_x,
                       plt=plt[1:na_f0])
         } else{
           fgcross_ss3(qNo_x,  qNo_f,
                       d03r_x, d03r_f,
                       d03q_x, d03q_f,
                       d03a_x0,d03a_f0,
                       na_x0,  na_f0,
                       qID,    d03ord_x,
                       lab_x,
                       plt=plt[na_f0:1])
         },
         "複数"=fgcross_ms(qNo_f,
                         d03r,   d03r_x, d03r_f,
                         d03q_x, d03q_f,
                         d03a_x0,d03a_f0,
                         na_f0,
                         qID,    d03ord_x,
                         lab_x,
                         plt=plt[na_f0:1]),
         "数値"=fgcross_ns(qNo_f,
                         d03r_x, d03r_f,
                         d03q_x, d03q_f,
                         d03a_x0,d03a_f0,
                         na_f0,
                         qID,
                         lab_x,
                         plt=plt[na_f0:1]))
}
```

3.5.2 関数の使い方

メインとなる `fgcross()` 関数の使い方について解説しておきます。まず、上記の関数定義を「cross01.R」というファイル名で保存しておきます。RStudio の新規作成（New File）の R Script を選んで貼り付ければいいですね。文字コードは UTF-8 にしましょう。

この関数定義を読み込む場合は、`source()` 関数を使って、以下のようにします。

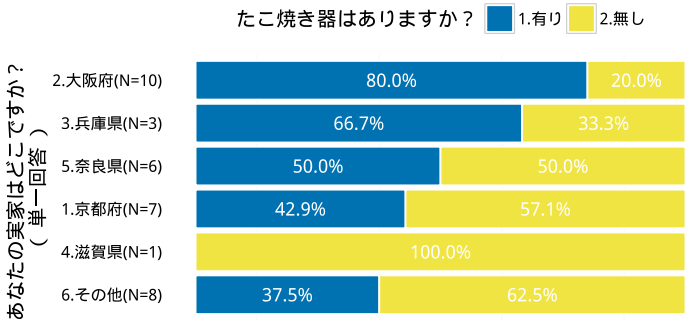
エクセルで作成して CSV 形式で保存した回答データ（dset_r.csv）、設問のリスト（dest_q.csv）、選択肢のリスト（dset_a.csv）を読み込みます。

```
> #データの読み込み
> d01r<-read.csv("dset_r.csv")
> d01q<-read.csv("dset_q.csv",as.is=T)
> d01a<-read.csv("dset_a.csv")
```

グラフを描きます。設問 1 と設問 2 のクロス集計をしてみます。まず、帯グラフから描いてみましょう。
引数は以下のとおりです。

dr　回答データ（選択肢番号で入力）
dq　設問のリスト
da　選択肢のリスト
qNo_x　X 軸に並べる設問の番号（設問リストの行番号）
qNo_y　クロス集計の分類に用いる設問の番号（〃）、単一回答だけが指定できる。
obi　帯グラフを描きたいとき TRUE、棒グラフを描きたいとき FALSE（qNo_x で指定した設問が複数回答または数値記入の場合は無視）。
plt　カラーパレットを指定する。デフォルトは関数内で作成する cb_palette。

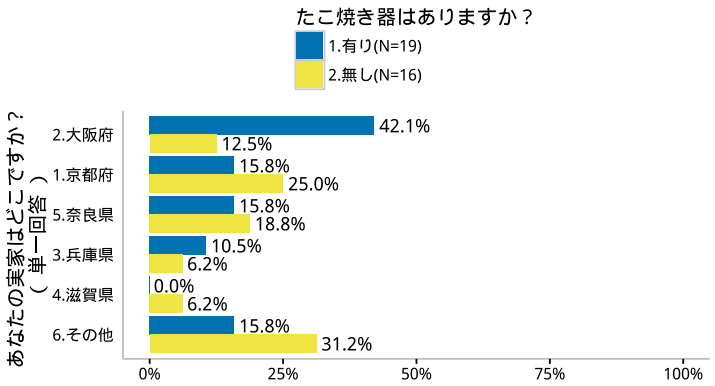
```
> fgcross(dr=d01r,dq=d01q,da=d01a,qNo_x=1,qNo_f=2,obi=TRUE,plt=cb_palette)
```



設問 1 では、d01q に並び順（order）が「大」と指定されているので、設問 2 の 1 番目の選択肢が大きい順に並び替えられています。

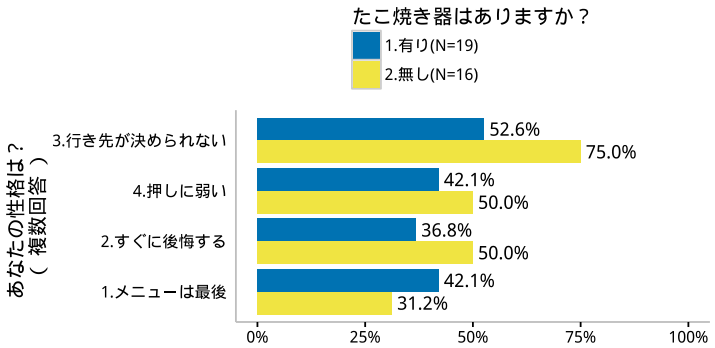
同じく、設問 1 と設問 2 のクロス集計の棒グラフを描いてみましょう。引数 obi=FALSE とするだけです。引数 plt=はデフォルトを使うので省略します。

```
> fgcross(dr=d01r,dq=d01q,da=d01a,qNo_x=1,qNo_f=2,obi=FALSE)
```



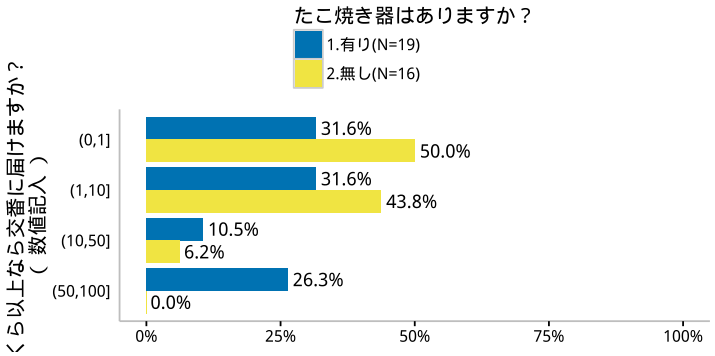
複数回答の設問 5 と設問 2 のクロス集計です。

```
> fgcross(dr=d01r,dq=d01q,da=d01a,qNo_x=5,qNo_f=2)
```



数値記入項目の設問 4 と設問 2 のクロス集計です。

```
> fgcross(dr=d01r,dq=d01q,da=d01a,qNo_x=4,qNo_f=2)
```



3.5.3　選択肢を統合する

クロス集計の結果を見て、選択肢をまとめたい時があります。そのような場合は、読み込んだデータそのものをいじってやればいいだけです。

たとえば、実家（Q01）とたこ焼き器の有無 (Q02）のクロス集計で実家の分類が多すぎるので、実家が大阪

府の場合とその他に 2 分して大阪人の特異性を強調したいと思います。まず、回答データ d01r を変更します。

```
> d02r<-transform(d01r,Q01=ifelse(Q01==2,1,2))
```

回答データを読み込んだデータフレーム d01r の変数 Q01 を iflesle() 関数を使って変換します。もし Q01==2 ならば 1 を代入し、そうでなければ 2 を代入したものを Q01 という変数名で生成します。

変換したデータの最初の方は以下です。

```
> head(d02r)
  ID Q01 Q02 Q03   Q04 Q05_01 Q05_02 Q05_03 Q05_04
1  1   1   1   2  10.0      0      0      1      1
2  2   2   1   2  50.0      0      0      1      1
3  3   2   1  NA   1.0      1      1      0      0
4  4   1   1   2 100.0      1      0      0      0
5  5   2   1   1   2.0      0      1      1      1
6  6   2   2   1   0.1      0      1      0      0
```

Q01 が 1 と 2 だけのデータに置き換えられています。

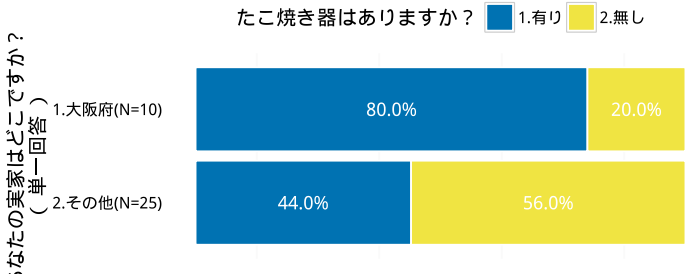
設問の文言は、そのままでも構いませんので、d01q はいじりません。選択肢は、「1. 大阪府」と「2. その他」に変更しなければなりません。

```
> (d02a<-transform(d01a,Q01=c("1. 大阪府","2. その他","","","","")))
      Q01     Q02         Q03 Q04                    Q05
1 1. 大阪府 1. 有り    1. 届ける   0       1. メニューは最後
2 2. その他 2. 無し 2. 届けない   1      2. すぐに後悔する
3                              10 3. 行き先が決められない
4                              50           4. 押しに弱い
5                             100
6                              NA
```

ちょっとガタガタしていますが、Q01 の選択肢が入れ替わっています。

あとは、新しく生成した d02r と d02a を指定すれば、グラフを書き直すことができます。

```
> fgcross(dr=d02r,dq=d01q,da=d02a,qNo_x=1,qNo_f=2,obi=TRUE)
```



問 5（Q05）の複数回答は優柔不断かどうかを具体的に聞いています。たとえば、これがいくつあてはまっているかをカウントしましょう。これが多くあてはまっている方が優柔不断度は高いと判断できるでしょう。

いくつあてはまるかどうかは、Q05_1～Q05_4 のデータが 0 か 1 で入力されているので、それを足し算すればわかります。

```
> d03r<-transform(d02r,yujufudan=Q05_01+Q05_02+Q05_03+Q05_04+1)
```

最後に 1 を足しているのは、定義した関数が 1, 2, 3, ... という自然数を回答データとして要求するからです。

変換したデータの最初の方は以下です。

```
> head(d03r)
  ID Q01 Q02 Q03   Q04 Q05_01 Q05_02 Q05_03 Q05_04 yujufudan
1  1   1   1   2  10.0      0      0      1      1         3
2  2   2   1   2  50.0      0      0      1      1         3
3  3   2   1  NA   1.0      1      1      0      0         3
4  4   1   1   2 100.0      1      0      0      0         2
5  5   2   1   1   2.0      0      1      1      1         4
6  6   2   2   1   0.1      0      1      0      0         2
```

yujufudan という新しい変数が作成されて、最後にくっついています。

新しい変数をつくったので、これに対応する設問の文言もつくる必要があります。

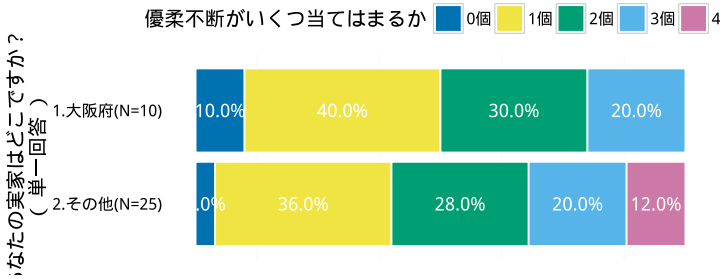
```
> (d03q<-rbind(d01q,c("yujufudan","優柔不断がいくつ当てはまるか","単一","同")))
       qID                                question    graph order
1      Q01           あなたの実家はどこですか？ 単一回答      大
2      Q02             たこ焼き器はありますか？ 単一回答      同
3      Q03 1 万円札を拾ったら交番に届けますか？ 単一回答      同
4      Q04     いくら以上なら交番に届けますか？ 数値記入
5      Q05                        あなたの性格は？ 複数回答      大
6 yujufudan          優柔不断がいくつ当てはまるか       単一      同
```

選択肢も加えなければなりません。

```
> (d03a<-cbind(d02a,yujufudan=c("0 個","1 個","2 個","3 個","4 個","")))
      Q01     Q02         Q03 Q04                      Q05 yujufudan
1 1. 大阪府 1. 有り    1. 届ける   0       1. メニューは最後       0 個
2 2. その他 2. 無し 2. 届けない   1       2. すぐに後悔する       1 個
3                              10 3. 行き先が決められない       2 個
4                              50           4. 押しに弱い       3 個
5                             100                               4 個
6                              NA
```

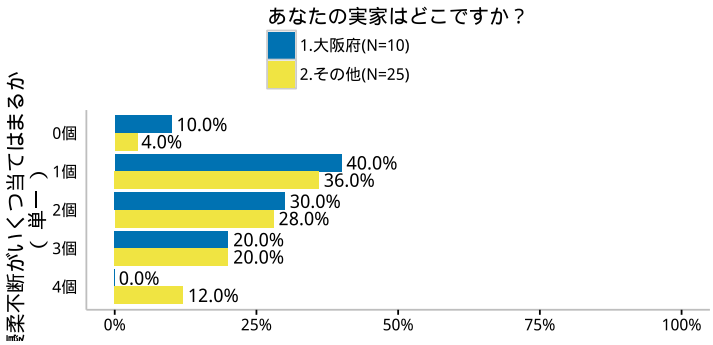
これで大阪人か否かと優柔不断度とのクロス集計をしてみましょう。

```
> fgcross(dr=d03r,dq=d03q,da=d03a,qNo_x=1,qNo_f=6,obi=T)
```



大阪人には優柔不断がやや少ないように見えます。棒グラフでも描いてみましょう。

```
> fgcross(dr=d03r,dq=d03q,da=d03a,qNo_x=6,qNo_f=1,obi=F)
```



どちらがわかりやすいですか？

問 5 の選択肢にあてはまる数が 3 以上だったら間違いなく優柔不断ということて、3 以上か 2 以下かで 2 分してみましょう。もし 3 以上だったら 1、そうでなかったら 2 を代入したものを yujufudan という名前の変数で保存します。

```
> d03r<-transform(d02r,yujufudan=ifelse(Q05_01+Q05_02+Q05_03+Q05_04>=3,1,2))
```

変換したデータの最初の方は以下です。

```
> head(d03r)
  ID Q01 Q02 Q03   Q04 Q05_01 Q05_02 Q05_03 Q05_04 yujufudan
1  1   1   1   2  10.0      0      0      1      1         2
2  2   2   1   2  50.0      0      0      1      1         2
3  3   2   1  NA   1.0      1      1      0      0         2
4  4   1   1   2 100.0      1      0      0      0         2
5  5   2   1   1   2.0      0      1      1      1         1
6  6   2   2   1   0.1      0      1      0      0         2
```

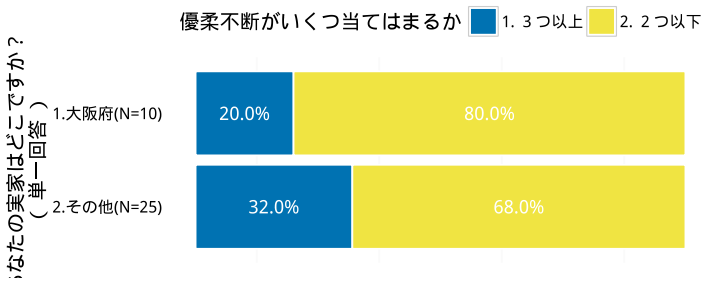
変数 yujufudan が 1, 2 の 2 変数にとなっています。

設問の文言はそのままで良しとして、選択肢を変更します。

```
> (d03a<-cbind(d02a,yujufudan=c("1. ３つ以上","2. ２つ以下","","","","")))
      Q01    Q02         Q03 Q04                    Q05   yujufudan
1 1.大阪府 1.有り    1.届ける   0       1.メニューは最後 1. ３つ以上
2 2.その他 2.無し 2.届けない   1       2.すぐに後悔する 2. ２つ以下
3                             10 3.行き先が決められない
4                             50           4.押しに弱い
5                            100
6                             NA
```

これで大阪人か否か、優柔不断か否かのクロス集計をしてみましょう。

```
> fgcross(dr=d03r,dq=d03q,da=d03a,qNo_x=1,qNo_f=6,obi=TRUE)
```



3.6　報告書の作成

3.6.1　R Markwown で Word ファイルとして出力する

ライブラリ rmarkdown を読み込みます。

```
> library(rmarkdown)
```

Pandoc がインストールされていることも確認します。

```
> pandoc_available()
[1] TRUE
```

以下のような Rmd ファイルを用意します。エンコードは UTF-8 です。ファイル名は「cross02.Rmd」とでもつけておきましょう。

````
---
title: "クロス集計をしてみる"
output: word_document
---

```{r,echo=FALSE}
source("cross01.R",encoding = "utf-8")
```

# 大阪人の特徴

## たこ焼きが好き？

大阪人はたこ焼きが好きなようです。大阪人か否かとたこ焼き器の有無をクロス集計してみました。

```{r,echo=FALSE,,fig.width=6,fig.height=2,dev="png"}
d01r<-read.csv("dset_r.csv")
d01q<-read.csv("dset_q.csv",as.is=T)
d01a<-read.csv("dset_a.csv")
d02r<-transform(d01r,Q01=ifelse(Q01==2,1,2))
d02a<-transform(d01a,Q01=alt02)
fgcross(dr=d02r,dq=d01q,da=d02a,qNo_x=1,qNo_f=2,obi=TRUE)
```

## 優柔不断ではない？

大阪人の優柔不断度に対する設問に対する各選択肢の回答率は以下でした。
```{r,echo=FALSE,,fig.width=6,fig.height=3,dev="png"}
fgcross(dr=d02r,dq=d01q,da=d02a,qNo_x=5,qNo_f=1)
```

優柔不断かどうかを判断する４つの選択肢のうち、３つ以上に回答した人と２つ以下の人の割合を大阪人とその他で比較してみます。
````

```
```{r,echo=FALSE,,fig.width=6,fig.height=2,dev="png"}
d03r<-transform(d02r,yujufudan=ifelse(Q05_01+Q05_02+Q05_03+Q05_04>=3,1,2))
d03q<-rbind(d01q,q03)
d03a<-cbind(d02a,yujufudan=alt03)
fgcross(dr=d03r,dq=d03q,da=d03a,qNo_x=1,qNo_f=6,obi=TRUE)
```
```

ただし、R Markdownd では、コードチャンク内（R のコードが書かれている部分）にかな漢字があるとエラーになるので、そこの部分は実行前に別に代入しておきます。

```
> alt02<-c("1. 大阪府","2. その他","","","","")
> q03<-c("yujufudan","優柔不断がいくつ当てはまるか","単一","同")
> alt03<-c("1. ３つ以上","2. ２つ以下","","","","")
```

その上で、以下を実行すると、ワードファイルが作成できるはずです。

```
> render("cross02.Rmd",encoding="utf-8")
```

書式がいまひとつなので変更して、次のようなワードファイルができました。

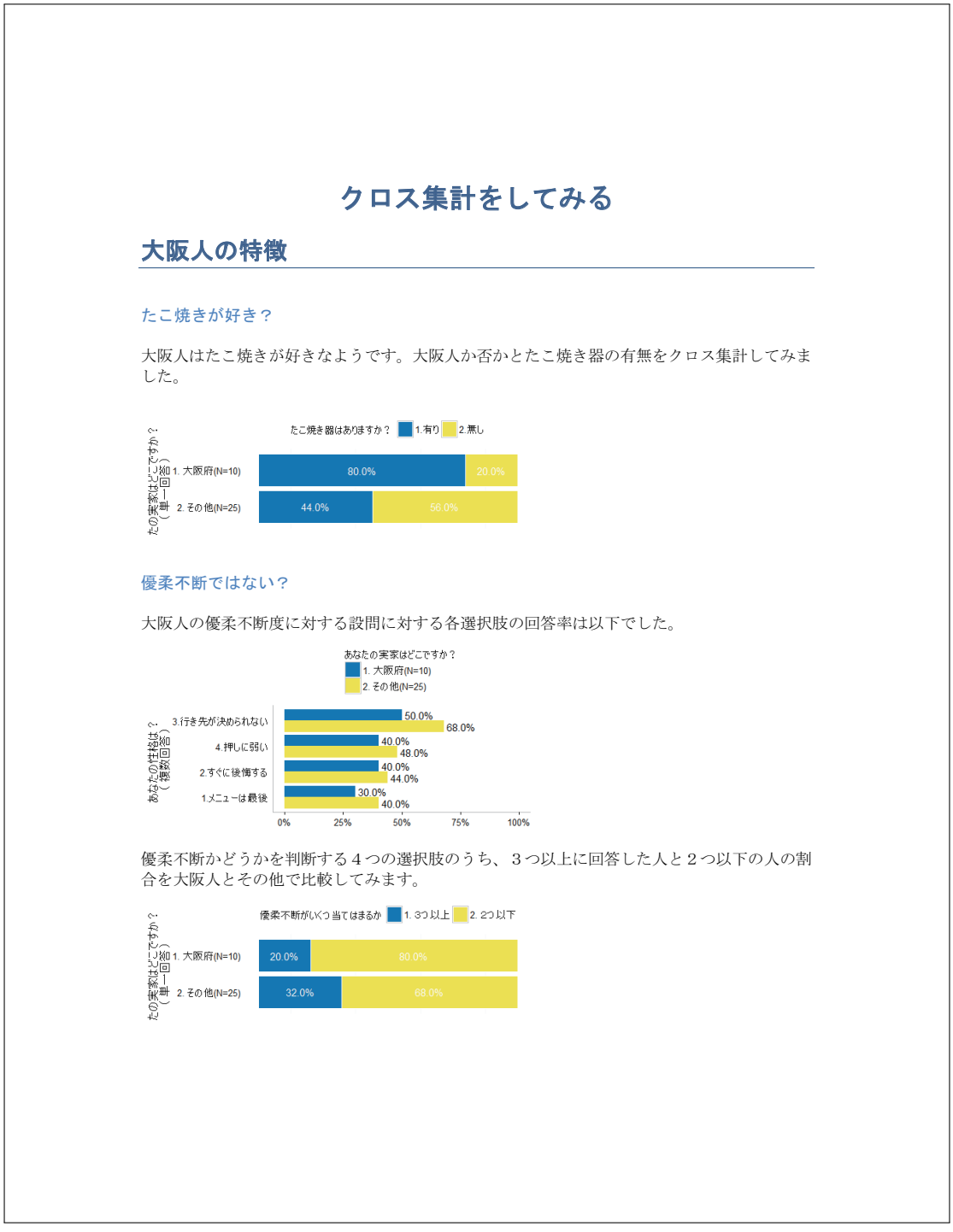


図 3.7: R Markdown でワードファイルに出力

3.6.2　Sweave で LaTeX ファイルを出力する

R Markdown と同じレポートを、Sweave を使って作成してみます。まずは、RStudio を使って、以下のような Rnw ファイルを作成します。作業フォルダに「cross_report01.Rnw」とでも名前をつけて保存しましょ

う。エンコードはあいかわらず UTF-8 です。

Chank と呼ばれる R のコマンドを実行する部分が、R Markdown では、```{r}・・・''' だったのに対し、Sweave では、<<>>=・・・@ に変わります。先頭部分は、LaTeX のプリアンブルと呼ばれるなんのかんのと定義するためのコマンドとなっています。節・小節など、Chank 以外の部分も LaTeX の文法に従います。

```
\documentclass[uplatex]{jsarticle}
\usepackage[dvipdfmx]{graphicx}
\usepackage[T1]{fontenc}
\usepackage[dvipdfmx]{xcolor}
\usepackage{ascmac}

\begin{document}


\section*{クロス集計をしてみる}

\subsection*{大阪人の特徴}


\subsubsection*{たこ焼きが好き？}

大阪人はたこ焼きが好きなようです。大阪人か否かとたこ焼き器の有無をクロス集計してみました。

\setkeys{Gin}{width=0.6\textwidth}
<<fig=TRUE,height=2.5>>=
setwd("c:/Users/yoshino/OneDrive/home/Documents/Rwd/statA")
source("cross01.R",encoding = "utf-8")
d01r<-read.csv("dset_r.csv")
d01q<-read.csv("dset_q.csv",as.is=T)
d01a<-read.csv("dset_a.csv")
d02r<-transform(d01r,Q01=ifelse(Q01==2,1,2))
showtext.begin()
d02a<-transform(d01a,Q01=c("1. 大阪府","2. その他","","","",""))
fgcross(dr=d02r,dq=d01q,da=d02a,qNo_x=1,qNo_f=2,obi=TRUE)
showtext.end()
@

\subsubsection*{優柔不断ではない？}

大阪人の優柔不断度に対する設問に対する各選択肢の回答率は以下でした。

<<fig=TRUE,height=3.1>>=
showtext.begin()
fgcross(dr=d02r,dq=d01q,da=d02a,qNo_x=5,qNo_f=1)
showtext.end()
@

優柔不断かどうかを判断する４つの選択肢のうち、３つ以上に回答した人と２つ以下の人の割合を大阪人とその他で比較してみます。

<<fig=TRUE,height=2.5>>=
showtext.begin()
d03r<-transform(d02r,yujufudan=ifelse(Q05_01+Q05_02+Q05_03+Q05_04>=3,1,2))
q03<-c("yujufudan","優柔不断がいくつ当てはまるか","単一","同")
alt03<-c("1. ３つ以上","2. ２つ以下","","","","")
d03q<-rbind(d01q,q03)
d03a<-cbind(d02a,yujufudan=alt03)
fgcross(dr=d03r,dq=d03q,da=d03a,qNo_x=1,qNo_f=6,obi=TRUE)
showtext.end()
@

\end{document}
```

作成した Rnw に対して、Sweave を実行します。

```
> Sweave("cross_report01",encoding = "utf-8")
```

すると、作業フォルダに cross_report01.tex というファイルができているはずなので、これを LaTeX で PDF ファイルに変換します。

CMD や Windows PowerShell などをのターミナルを起動して、まず作業フォルダに移動します。

```
cd c:/Users/hogehoge/Documents
```

続いて、LaTeX を実行します。

```
ptex2pdf -u -l -ot "-kanji=utf8 -no-guess-input-enc -synctex=1" cross_report01.tex
```

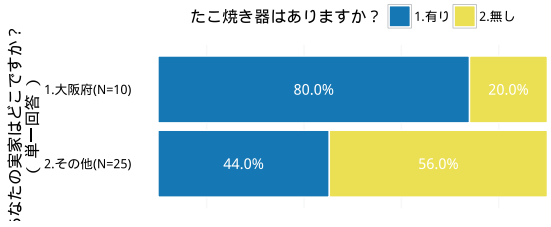
これで cross_report01.pdf という名前で次のようなファイルができているはずです。

クロス集計をしてみる

大阪人の特徴

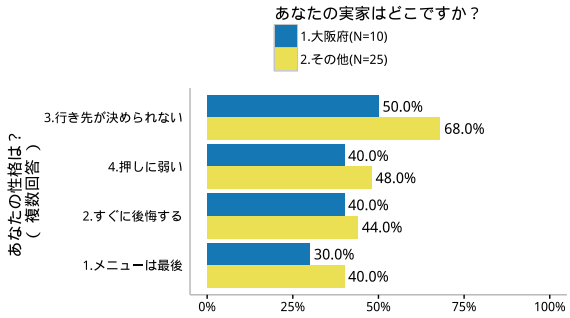
たこ焼きが好き？

大阪人はたこ焼きが好きなようです。大阪人か否かとたこ焼き器の有無をクロス集計してみました。



優柔不断ではない？

大阪人の優柔不断度に対する設問に対する各選択肢の回答率は以下でした。



優柔不断かどうかを判断する４つの選択肢のうち、３つ以上に回答した人と２つ以下の人の割合を大阪人とその他で比較してみます。

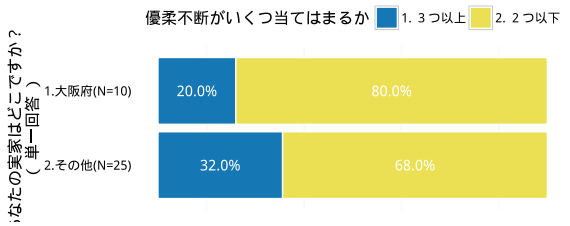


図 3.8: Sweave による PDF の出力

3.7　補足：グラフの幅を揃える

3.7.1　クロス集計について

だいたいできましたが、レポートを出力して、若干気になる点が・・・

グラフの幅は揃えられんのかと・・・

軸ラベルなども含めたグラフ全体の幅は同じでも、選択肢文の幅が違うので棒グラフの 0〜100 %の長さがグラフごとにまちまちです。これを同じにしてみましょう。

これはちょっとややこしいです。

最初に、グラフを描くための関数などを保存してある「cross01.R」の最後に、以下を挿入して、上書き保存しましょう。

```
library{grid}

#棒グラフの選択肢文の幅を統一して（最大に合わせて)、グラフの幅を同じにする

#全設問で最大の選択肢文の幅を抽出する
maxWidth_c<-function(dr,dq,da,qNo_f){
  #選択肢が最大の設問 ID を調べる
  qNo_amax<-da%>%
      as.matrix %>%                 #data.frame だと次の nchar が受け付けないので
      nchar(type="width")%>%        #文字列の幅を測る
      as.data.frame %>%             #data.frame に戻す
      summarise_each(funs(max))%>% #各列の最大値
      which.max                     #最大の設問番号

  g03_amax<-fgcross(dr,dq,da,qNo_amax,qNo_f,obi=F) #グラフを出力
  gp03_amax<-ggplotGrob(g03_amax)                  #グラフのオブジェクト化
  gp03_amax$widths[3]                              #選択肢文の最大幅
}

#最大の設問文の幅を外挿してグラフを描きなおす
fgcross02<-function(dr,dq,da,qNo_x,qNo_f,maxw=maxw03,obi=TRUE,plt=cb_palette){
  g03_adj<-fgcross(dr,dq,da,qNo_x,qNo_f,obi)       #とりあえず目的のグラフを出力
  gp03_adj<-ggplotGrob(g03_adj)                    #そのグラフをオブジェクト化
  gp03_adj$widths[3]<-maxw                         #選択肢文のとこに最大幅をつっこむ
  grid.draw(gp03_adj)                              #グラフの描画
}
```

使い方は、まず、「cross01.R」を再読み込みしましょう。

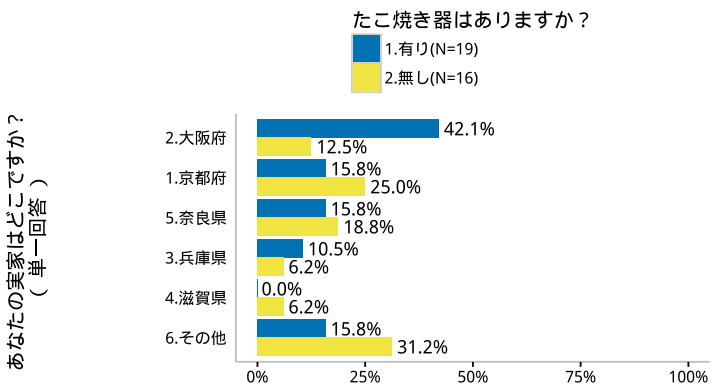
```
> source("cross01.R",encoding = "utf-8")
```

元データの読み込みはこれまでと同じように終わっているものとして・・・、次に、関数 maxWidth_c() でアンケート全体で、設問文の幅の最大値を求めます。引数 qNo_f=は、クロスさせる選択肢が 2 つしかない設問の設問番号です。選択肢が 2 つなら、どの設問でもいいです。

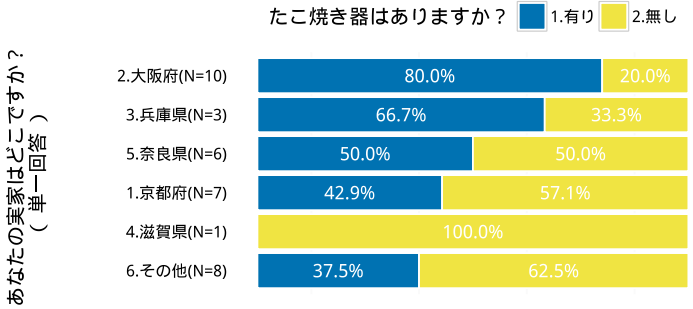
```
> maxw03<-maxWidth_c(dr=d01r,dq=d01q,da=d01a,qNo_f=2)
```

これを関数 fgcross02() 関数に入れると、グラフを描画します。

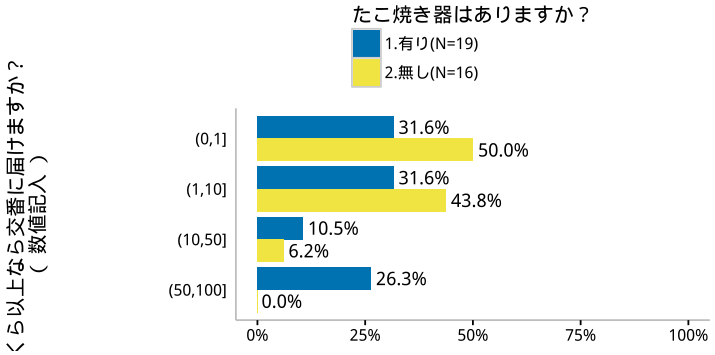
```
> fgcross02(dr=d01r,dq=d01q,da=d01a,qNo_x=1,qNo_f=2,maxw=maxw03,obi=FALSE)
```



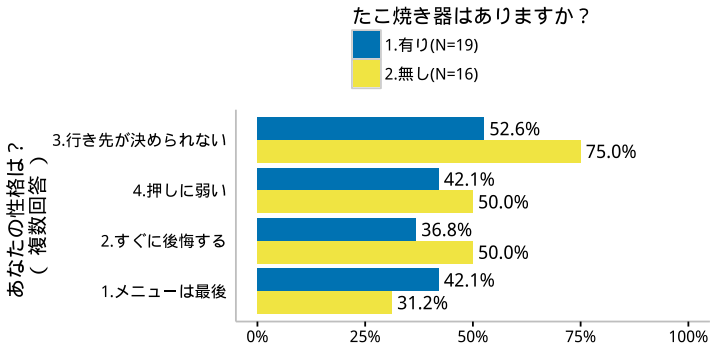

```
> fgcross02(dr=d01r,dq=d01q,da=d01a,qNo_x=1,qNo_f=2,maxw=maxw03,obi=TRUE)
```




```
> fgcross02(dr=d01r,dq=d01q,da=d01a,qNo_x=4,qNo_f=2,maxw=maxw03,obi=FALSE)
```




```
> fgcross02(dr=d01r,dq=d01q,da=d01a,qNo_x=5,qNo_f=2,maxw=maxw03,obi=FALSE)
```



こんな感じです。あとは X 軸の方のラベルが、若干はみ出す場合がありますが、これについては、設問分に \n を入れて 2 行に分けるなどの方法もあります。手っ取り早くは、文言を短くするなどの工夫で対応してもよいでしょう。

3.7.2　ついでに単純集計も

単純集計の方も同じようにできます。前章で作成した「simple01.R」の最後に、以下を追加しましょう。

```
library{grid}

#棒グラフの選択肢文の幅を統一して（最大に合わせて）、グラフの幅を同じにする

#全設問で最大の選択肢文の幅を抽出する
maxWidth_s<-function(dr,dq,da){
  #選択肢が最大の設問 ID を調べる
  qNo_amax<-da%>%
    as.matrix %>%                 #data.frame だと次の nchar が受け付けないので
    nchar(type="width")%>%        #文字列の幅を測る
    as.data.frame %>%             #data.frame に戻す
    summarise_each(funs(max))%>% #各列の最大値
    which.max                     #最大の設問番号

  g01_amax<-fgsimple(dr,dq,da,qNo_amax) #グラフを出力
  gp01_amax<-ggplotGrob(g01_amax)       #グラフのオブジェクト化
  gp01_amax$widths[3]                    #選択肢文の部分の幅
}

#最大の設問文の幅を外挿してグラフを描きなおす
fgsimple02<-function(dr,dq,da,qNo,obi=NULL,maxw=maxw01,
                plts3=cb_palette[1],
                plts2=cb_palette,
                pltm=cb_palette[3],
                pltn=cb_palette[4]){
  g01_adj<-fgsimple(dr,dq,da,qNo,obi)     #とりあえず目的のグラフを描く
  gp01_adj<-ggplotGrob(g01_adj)           #そのグラフをオブジェクト化
  gp01_adj$widths[3]<-maxw                #選択肢文の最大幅を外挿
  grid.draw(gp01_adj)                     #グラフの描画
}
```

そして、「cross01.R」を再読み込みしましょう。

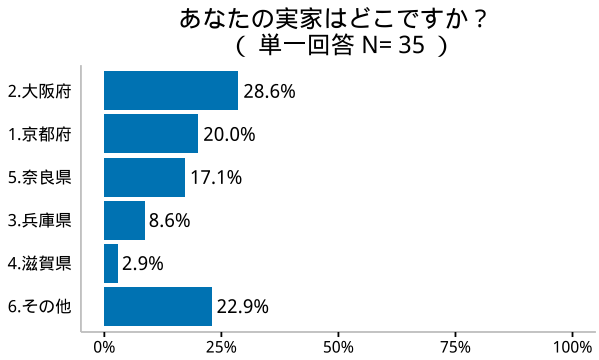
```
> source("simple01.R",encoding = "utf-8")
```

元データの読み込みは終わっているものとして・・・、次に、関数 maxWidth_s() でアンケート全体で、設問文の幅の最大値を求めます。

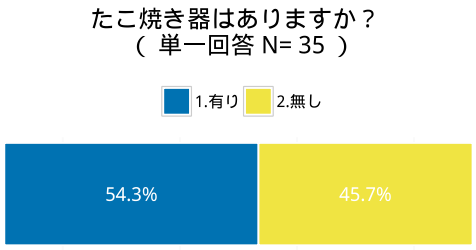
```
> maxw01<-maxWidth_s(dr=d01r,dq=d01q,da=d01a)
```

これを関数 fg02() 関数に入れると、グラフを描画します。

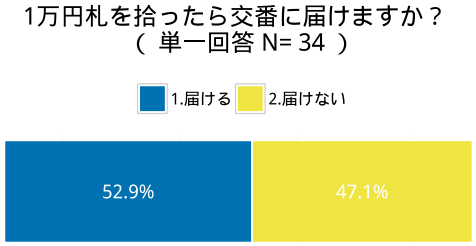
```
> fgsimple02(dr=d01r,dq=d01q,da=d01a,qNo=1,maxw=maxw01)
```



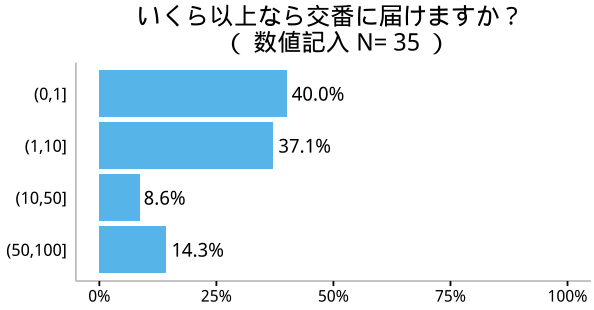

```
> fgsimple02(dr=d01r,dq=d01q,da=d01a,qNo=2,maxw=maxw01)
```



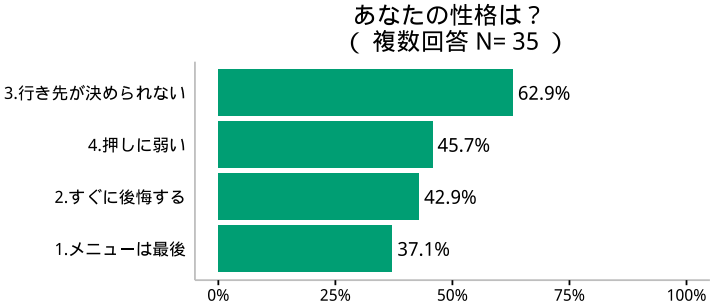

```
> fgsimple02(dr=d01r,dq=d01q,da=d01a,qNo=3,maxw=maxw01)
```




```
> fgsimple02(dr=d01r,dq=d01q,da=d01a,qNo=4,maxw=maxw01)
```




```
> fgsimple02(dr=d01r,dq=d01q,da=d01a,qNo=5,maxw=maxw01)
```



参考文献

Chang, Winston (2013). *R グラフィックスクックブック : ggplot2 によるグラフ作成のレシピ集*. 訳者: 石井弓美子 河内崇, 瀬戸山雅人, 古畠敦. 東京: オライリー・ジャパン. ISBN: 9784873116532. URL: http://m.kulib.kyoto-u.ac.jp/webopac/BB04703135.