Partial Least Squares（PLS; 部分最小二乗法，偏最小二乗法）

PLSってどのくらい使えるのか？

> 1. PLSは$(Xu)'Yv$を最大化するようにu^*とv^*を決める．
> 2. このu^*とv^*は，$X'Y$の特異値分解から得られる．
> 3. 推定の考え方がOLSとは違うので，XとYにかく乱項が加わった場合，係数の不偏性は怪しい．
> 4. PLSの実際の利用は，多数の変数を持つXからYの説明力の高いXの主成分を抽出し，それと相関の高いXを特定すること．
> 5. 変数間の係数に関心がある場合は，PLSで特定したXでPCRをするのがよさそう．
> 6. 実際のPLSは，$X'Y$の特異値分解という単純なものでなく，主成分を 1 つずつ算出していく複雑なアルゴリズムが複数ある．
> 7. Rのパッケージplsにある`plsr()`関数で出力されるYのスコア（`Yscores`）はXのスコアに合わせてスケールを調整しているので要注意．

説明変数XでYに回帰させる一般的な方法がOLSなのだが，Xに相関があると回帰がうまくいかない．

そこで，Xを主成分分析して，相関する変数をまとめた主成分を抽出する主成分回帰分析（PCR）というのがある．しかし，このXの主成分分析にはYの情報を含まないので，必ずしも主成分分析の結果が意味のあるものでないかもしれない．

そこでそこで，Yの情報も含めて主成分分析的な回帰をやろうというのがPLS．

もともと心理学の分野で開発されたらしいが，最近は化学の分野で多用されているらしい．特定の反応を説明するのに，サンプル数が数十個しかないのに対して，説明変数の候補が100個以上あったりする場合に，説明変数の候補を探し出すのに使われているようだ．

我々が行う購買行動調査のようなアンケートも，性別や年齢などの属性で説明できるような多純なものではなく，様々な消費性向のようなことを重ねて聞くから，どれとどれがどれと関連しているかを突き止めるのが大変．

PLSはあくまで連続変数間の関係なので，アンケート分析にどれだけ使えるかわからないが，なんかの参考にはなるだろう．PLSは被説明変数に複数の変数をとれるというのも魅力的だ．

気になったので勉強しようとしてみた．ところが，どうも教科書が無いのか？ネット上の解説をあさってみた．上記のような話が出てきたが，具体的に何をしているかがよくわからない．たくさん抽出できる主成分の中からどうやって主成分を抽出するかという**手順**が説明してある．しかし，その前に**そもそもどんな回帰をしているのか？**その説明が飛んでいる．PLSの場合，この手順が醍醐味なのはよくわかったが，言ってしまえば，そこは計算機の仕事でしょ，ということになる．それよりも，結果として出力される値の意味が知りたい．少なくとも分かった気になりたい．

それで，自分でデータを作成してやってみた．ネットの説明は，Rのパッケージに添付されているデータを使っているので，出力された値の理解がむずかしい．推定モデルというのは，まずは真の関係を想定する．そこにいろんなノイズが入り込む．なかなか真の関係にはたどり着かないのだけれど，できるだけそれに近い，あるいはもっともらしい解を見つけようとする．それはそれでよいのだが，しかし，例題で使われているのは，現実のデータであったりするから，真の関係が果たしてどうなのかというのは本当のところは分からない．だから推定された値が妥当なのか，的外れたものなのかがついぞわからん．

そこで，ひとまず**ノイズを含まない真の関係にあるデータ**を人為的に作り出して，それを分析にかけてみて，分析手法で何を明らかにできるのか，推定される値が何を意味しているのかをを確かめてみよう．

それからちょっとずつノイズを加えていき，その推定方法の信頼性や強みを見ていったらよいのではないか．

以下，最初に，ノイズを含まない被説明変数が２つ，説明変数が２つで，それぞれきつい相関があるデータを作成し，それを分析して，PLSの意味を理解していく．

次に，２×２変数はそのまま，攪乱項を加えた場合に，PLSがどう作動するかを確認する．

そして，さらに余計な説明変数をたくさん加えて，そうした雑音の中から，真の関係をどの程度抽出できるものなのかについて検証してみる．

```
library(tidyverse)
library(MASS)
library(pls)
library(scatterplot3d)
```

2×2変数でノイズなし

まず，Xが２変数，Yが2変数の対応を確認する

データの作成

Xの作成

PLSでは全ての変数を標準化（平均0，分散1に）する．
平均0，分散1,共分散0.8の強めの相関がある51個の2変数`x1`を作成する．

```
x1_0<-c(0,0,
1,1,
-1,-1,
1,-1,
-1,1)%>%
  matrix(.,ncol=2,byrow = T)
set.seed(123)
x1_1<-mvrnorm(46,
              mu=c(0,0),
              Sigma=matrix(c(46/45,0.8888889,
                             0.8888889,46/45),
                           nrow=2),
              empirical=T)
```

```
x1←rbind(x1_0,x1_1)
```

平均は２変数とも0

```
apply(x1_1,2,mean)%>%
  round(4)
```

```
## [1] 0 0
```

分散・共分散は

```
var(x1)
```

```
##      [,1] [,2]
## [1,]  1.0  0.8
## [2,]  0.8  1.0
```

・・・で，Xのでーたは，うまいことできた！

プロット図に書いてみよう

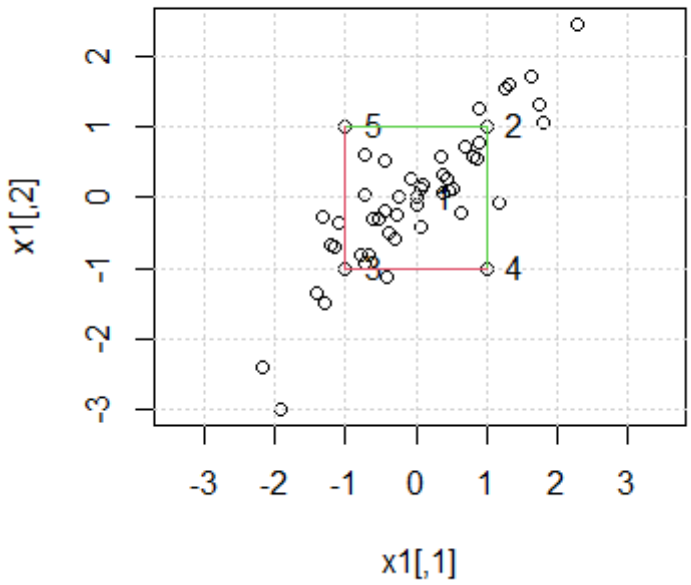
データの変換をプロット図で追いかけられるように，2~5行目のデータは，特殊な点を無理やり入れてある．そのをつなぐ線を描きたい．

その関数を定義

```
f_grid1←function(x1){
  segments(x1[2,1],x1[2,2],x1[4,1],x1[4,2],col=3)
  segments(x1[2,1],x1[2,2],x1[5,1],x1[5,2],col=3)
  segments(x1[3,1],x1[3,2],x1[4,1],x1[4,2],col=2)
  segments(x1[3,1],x1[3,2],x1[5,1],x1[5,2],col=2)
}
```

Xのプロット図．1が原点，2が(1,1)，3が(-1,-1)，4が(1,-1)，5が(-1,1)．

```
plot(x1,asp=1)
grid()
text(x=x1[1:5,1],y=x1[1:5,2],1:5,pos = 4)
f_grid1(x1[1:5,])
```



Xの主成分のスコアの計算

Xの主成分分析をすると・・・

```
prc_x1←prcomp(x1)
```

固有ベクトルは，第１主成分が右上がりの45度線，第２主成分がその直交（右下がり45度線）

```
prc_x1$rotation
```

```
##            PC1        PC2
## [1,] 0.7071068  0.7071068
## [2,] 0.7071068 -0.7071068
```

固有値は

```
prc_x1$sdev^2%>%
  round(4)
```

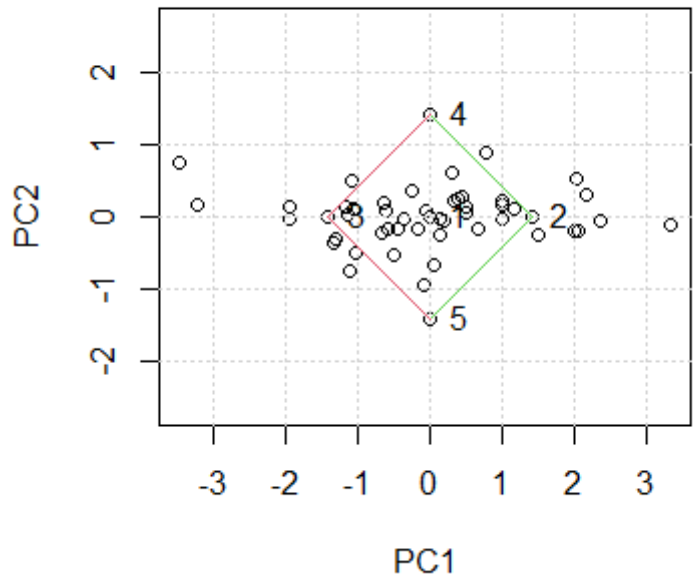
```
## [1] 1.8 0.2
```

スコアを求める．

```
Px1←prc_x1$x
```

スコアのプロット図：横軸が第１主成分，縦軸が第２主成分．
Xのプロットを45度度マイナスにした形となる．

```r
plot(Px1,asp=1)
grid()
text(x=Px1[1:5,1],y=Px1[1:5,2],1:5,pos=4)
f_grid1(Px1)
```



Xのプロット図の45度線でスコアを測っていることがわかる.
Xの2変数とも標準化してしまうと，どうあがいても，固有ベクトルの方向は，45度線になってしまう.

第１主成分が(x_1, x_2)の方のプロット図の右上がり45度線.

第２主成分が同図の右下がり45度線

つまり，固有ベクトルは，45度($\pi/4$)線で，長さ（ノルム）1なので・・・

$$\begin{pmatrix} \cos\frac{\pi}{4}, \cos\frac{-\pi}{4} \\ \sin\frac{\pi}{4}, \sin\frac{-\pi}{4} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \end{pmatrix}$$

ということ.

各主成分で符号が逆になる場合があるが，±どっち向きにスコアを測るかが違うだけで，本質的に同じ.

なので，$1/\sqrt{2} = 0.7071... = a$と置いておくと・・・

```r
a<-1/sqrt(2)
```

スコアは，$ax_1 + ax_2$（第１主成分），$ax_1 - ax_2$（第２主成分）

```
cbind(a*x1[,1]+a*x1[,2],a*x1[,1]-a*x1[,2])%>%
  round(4)%>%
  head()
```

```
##         [,1]    [,2]
## [1,]  0.0000  0.0000
## [2,]  1.4142  0.0000
## [3,] -1.4142  0.0000
## [4,]  0.0000  1.4142
## [5,]  0.0000 -1.4142
## [6,] -1.0460  0.1174
```

ちゃんと主成分分析のスコアと一致しますね.

```
Px1%>%
  round(4)%>%
  head()
```

```
##          PC1     PC2
## [1,]  0.0000  0.0000
## [2,]  1.4142  0.0000
## [3,] -1.4142  0.0000
## [4,]  0.0000  1.4142
## [5,]  0.0000 -1.4142
## [6,] -1.0460  0.1174
```

固有値は，それぞれ**スコアの分散**なんですね，これが！

さらに，$Var(x_1) = Var(x_2) = 1$なので，
第１主成分の固有値は，この場合，$1 + Cov(x_1, x_2) = 1.8$
第２主成分の固有値は，この場合，$1 - Cov(x_1, x_2) = 0.2$

```
var(prc_x1$x)%>%
  round(4)
```

```
##     PC1 PC2
## PC1 1.8 0.0
## PC2 0.0 0.2
```

Yを作成する

方針

XとYのダイレクトな規定関係ではなく，複数のXのスコアと複数のYのスコアとの関係があるものと考えよう.

観測できる変数（顕在変数）XとYは，観測できない変数（潜在変数，XのスコアとYのスコア）がノイズをもって観測されたものと考える.

最も簡単な例として，Xが上記2変数で，Yも2変数.
（Yが１変数という場合もあるが，PLSが２変数以上を被説明変数にできるというのだから，そこは2変数としよう．）

XとYは，それぞれ２変数とも平均0，標準偏差1に標準化．

ひとまず，xとYにノイズは一切入らないとする．

YのスコアをP_{y1}, P_{y2}，XのスコアをP_{x1}, P_{x_2}と表し・・・

$P_{y1} = 0.7P_{x1} + 1.3P_{x2}$

$P_{y2} = 0.346489P_{x1} - 1.679139P_{x2}$

・・・でYのスコアが決まるとする．

Yは，スコアから45度逆回転させればよいのだから．

$P_{y1} = \frac{1}{\sqrt{2}}y_1 + \frac{1}{\sqrt{2}}y_2$

$P_{y2} = \frac{1}{\sqrt{2}}y_1 - \frac{1}{\sqrt{2}}y_2$

で，$Y = (y_1, y_2)$ を求めることができる．

手順

まずy_1の2つの主成分のスコアP_{y1}, P_{y2}を求める．

第1主成分P_{y1}が，X$の主成分のスコアと次のような関係にあるとすると・・・（0.7, 1.3という値は適当に決めた）

$P_{y1} = 0.7P_{x1} + 1.3P_{x2}$

```
Py1_1<-0.7*Px1[,1]+1.3*Px1[,2]
```

この時，P_{y1}の分散は

```
var(Py1_1)
```

```
## [1] 1.22
```

Yの第2主成分

$P_{y2} = b_1P_{x1} + b_2P_{x2}$

は，これと直交するから

$P'_{y1}P_{y2} = 0$

となるようにP_{y2}を決めたい．

$(0.7P_{x1} + 1.3P_{x2})'(b_1P_{x1} + b_2P_{x2}) = 0$

$$\begin{pmatrix} 0.7 & 1.3 \end{pmatrix} \begin{pmatrix} P'_{x1} \\ P'_{x2} \end{pmatrix} \begin{pmatrix} P_{x1} & P_{x2} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = 0$$

$$\begin{pmatrix} 0.7 & 1.3 \end{pmatrix} \begin{pmatrix} 90 & 0 \\ 0 & 10 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = 0$$

$0.7 \times 90b_1 + 1.3 \times 10b_2 = 0$

さらに，２つのスコアの分散の合計を2にしたいから

$Var(P_{y2}) = 2 - 1.22 = 0.78$

$1.8b_1^2 + 0.2b_2^2 = 0.78$

b_2/b_1は

```
b2pb1<--0.7*90/(1.3*10)
```

b_1は

```
b1<-sqrt(0.78/(1.8+0.2*b2pb1^2))
```

$b2$は

```
b2<-b2pb1*b1
```

つまり，(b_1, b_2)は

```
c(b1,b2)
```

```
## [1]  0.346489 -1.679139
```

P_{y1}の係数と合わせて`b0`に代入しておこう．

```
b0<-matrix(c(0.7,1.3,b1,b2),nrow=2)
```

$P_{y2} = b_1 P_{x1} + b_2 P_{x2}$が完成！

```
Py1_2<-b1*Px1[,1]+b2*Px1[,2]
```

P_{y1}と合わせて，$P_y = (P_{y1}, P_{y2})$.

```
Py1<-cbind(Py1_1,Py1_2)
```

直交しているか確認．

```
var(Py1)%>%
  round(4)
```

```
##       Py1_1 Py1_2
## Py1_1  1.22  0.00
## Py1_2  0.00  0.78
```

ちゃんと直交しているし，分散も，合わせて2になっている．
P_yができた．

次に，Yを求める．

これは簡単．P_yを45度逆回転させればよいだけ．

```
a1←matrix(c(a,a,a,-a),ncol=2)
y1←Py1%*%solve(a1)
```

Yの平均は，

```
apply(y1,2,mean)%>%
  round(4)
```

```
## [1] 0 0
```

Yの分散・共分散

```
var(y1)%>%
  round(4)
```

```
##      [,1] [,2]
## [1,] 1.00 0.22
## [2,] 0.22 1.00
```

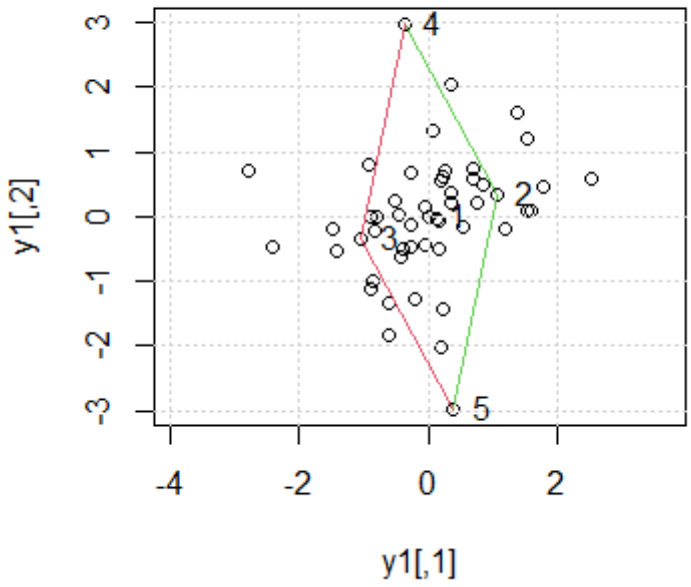
ちゃんと標準化されてる！

Yのデータ完成．

Yのプロット図

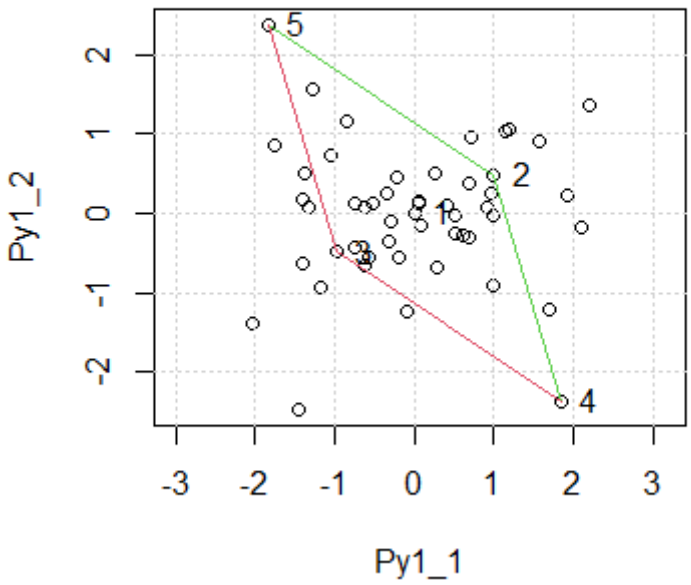
Yの分布を確認する

```
plot(y1,asp=1)
grid()
text(x=y1[1:5,1],y=y1[1:5,2],1:5,pos = 4)
f_grid1(y1)
```



Yのスコアのプロット．Yのプロットを45度マイナスにした形となる．

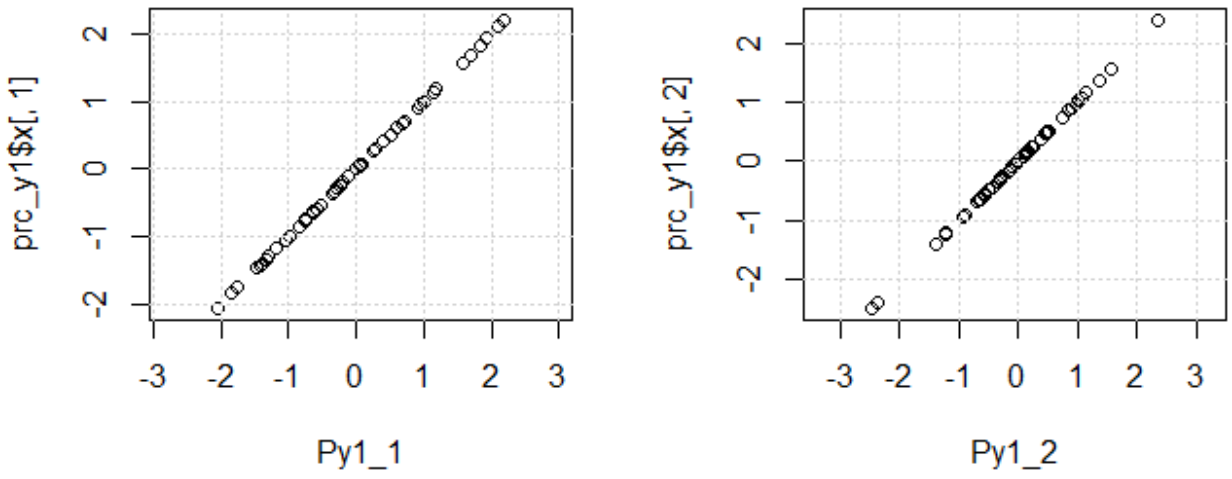
```
plot(Py1,asp=1)
grid()
text(x=Py1[1:5,1],y=Py1[1:5,2],1:5,pos=4)
f_grid1(Py1)
```



・・・という感じ．

(P_{y1}, P_{y2})もYの主成分となっていることを確認しておく．

```r
prc_y1←prcomp(y1)
par(mfrow=c(1,2))
plot(prc_y1$x[,1]~Py1_1,asp=1)
grid()
plot(prc_y1$x[,2]~Py1_2,asp=1)
grid()
```



・・・と，まあ，データの作成に手間取ったが，通常は，「こんなデータがありました！」ということで，ここからが分析のスタートとなる．

OLS回帰とPCR

関係式

XとYの関係式は以下

X→主成分P_x

$$P_{x1} = \frac{1}{\sqrt{2}}x_1 + \frac{1}{\sqrt{2}}x_2$$

$$P_{x2} = \frac{1}{\sqrt{2}}x_1 - \frac{1}{\sqrt{2}}x_2$$

$$A = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

とすると，

$$P_x = XA$$

（P_xは51行×2列，Xも51行2列，Aは2行×2列）

Y→主成分P_y

$$P_{y1} = \frac{1}{\sqrt{2}}y_1 + \frac{1}{\sqrt{2}}y_2$$

$$P_{y2} = \frac{1}{\sqrt{2}}y_1 - \frac{1}{\sqrt{2}}y_2$$

これも・・・

$$P_y = YA$$

（P_yは51行×2列，yも51行2列，Aは2行×2列）

主成分P_x→主成分P_y

$$P_{y1} = 0.7P_{x1} + 1.3P_{x2}$$

$$P_{y2} = 0.346489P_{x1} - 1.679139P_{x2}$$

$$B = \begin{pmatrix} 0.7 & 1.3 \\ 0.346489 & -1.679139 \end{pmatrix}$$

とすると，

$$P_y = P_x B$$

（P_yは51行×2列，P_xも51行2列，Bは2行×2列）

OLS回帰

作成したXとYでOLS回帰してみる．

y_1への回帰は・・・

```
lm(y1[,1]~x1-1)%>%
  summary()
```

```
## Warning in summary.lm(.): essentially perfect fit: summary may be unreliable

##
## Call:
## lm(formula = y1[, 1] ~ x1 - 1)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -4.951e-16 -6.125e-17  2.570e-18  9.978e-17  1.918e-15
##
## Coefficients:
##      Estimate Std. Error   t value Pr(>|t|)
## x11 3.337e-01  7.162e-17 4.659e+15   <2e-16 ***
## x12 7.128e-01  7.162e-17 9.953e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.039e-16 on 49 degrees of freedom
```

```
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 2.708e+32 on 2 and 49 DF,  p-value: < 2.2e-16
```

データにかく乱項を全く入れなかったので推定誤差は0．以下同様なので，係数だけ見る．

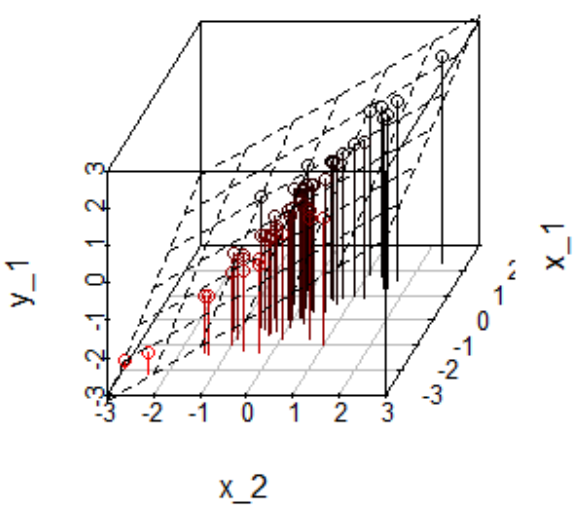
```
lm(y1~x1-1)%>%
  coef()%>%
  round(4)
```

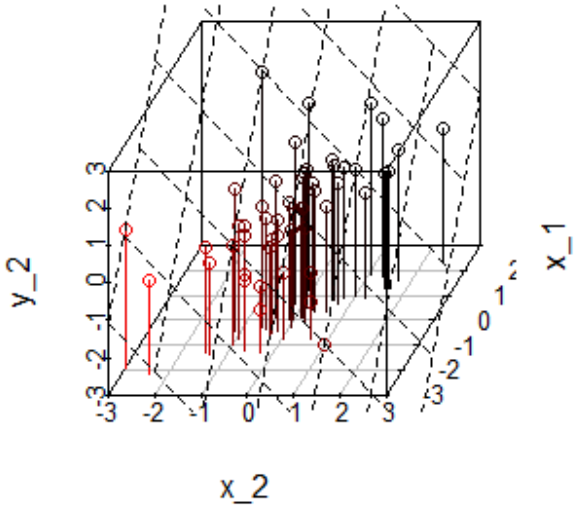
```
##       [,1]    [,2]
## x11 0.3337  1.6663
## x12 0.7128 -1.3128
```

（被説明変数を2変数にすると，それぞれの回帰をいっぺんにやってくれる． 1列目が`y1[,1]に対する回帰の結果， 2列目が`y1[,2]`に対する回帰の結果）

一見，y_1に対してx_1が1.38，x_2がその半分ぐらいの強さで影響しているように見えるが，プロットしてみると・・・

```
d1_1_OLS<-data.frame(x_2=x1[,2],
                     x_1=x1[,1],
                     y_1=y1[,1])

d1_2_OLS<-data.frame(x_2=x1[,2],
                     x_1=x1[,1],
                     y_2=y1[,2])

par(mfrow=c(1,2))

g3d1_1_OLS<-scatterplot3d(d1_1_OLS,
              type="h",
              angle=60,
              highlight.3d = T)
fit1_1_OLS<-lm(y_1~x_2+x_1-1,d1_1_OLS)
g3d1_1_OLS$plane3d(fit1_1_OLS)

g3d1_2_OLS<-scatterplot3d(d1_2_OLS,
              type="h",
              angle=60,
              highlight.3d = T)
fit1_2_OLS<-lm(y_2~x_2+x_1-1,d1_2_OLS)
g3d1_2_OLS$plane3d(fit1_2_OLS)
```





Xの相関が強く，平面を回帰するのは無理な感じがするが，誤差はないので，これはこれ．

$$P_x = XA$$

$$P_y = YA$$

$$P_y = P_xB$$

という関係にあるので，YにXを直接回帰したら，

$$Y = P_yA^{-1} = P_xBA^{-1} = XABA^{-1}$$

であり，Xの係数はABA^{-1}ということだ．

これを上の例で計算してみると・・・

```
a1%*%b0%*%solve(a1)%>%
  round(4)
```

```
##        [,1]    [,2]
## [1,] 0.3337  1.6663
## [2,] 0.7128 -1.3128
```

OLSの結果と一致する．

主成分回帰（PCR）

XのスコアとYとの回帰で，主成分回帰にあたる．

```
lm(y1~Px1-1)%>%
  coef()
```

```
##             [,1]    [,2]
## Px1PC1  0.7399795 0.249970
## Px1PC2 -0.2680918 2.106569
```

$Y = P_y A^{-1} = P_x B A^{-1}$ であり，P_xの係数はBA^{-1}.

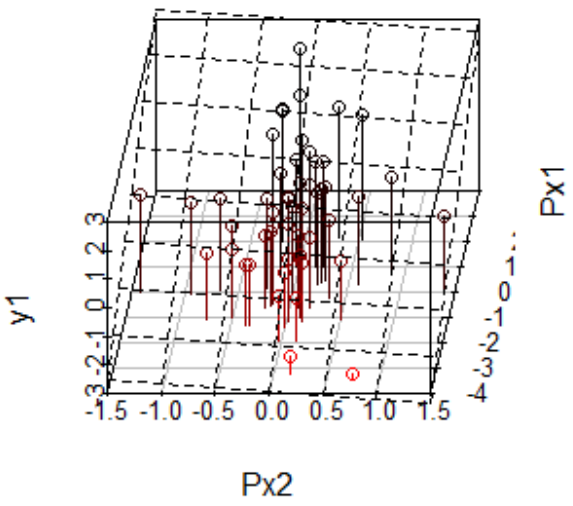
上の例だと・・・

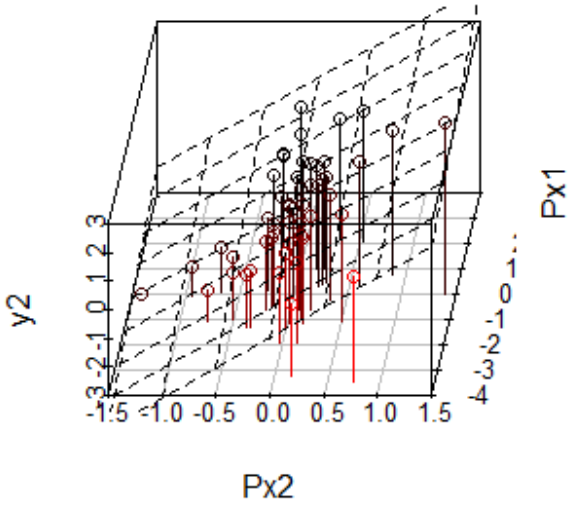
```
b0%*%solve(a1)
```

```
##           [,1]    [,2]
## [1,]  0.7399795 0.249970
## [2,] -0.2680918 2.106569
```

一致する.

```
d1_1_PCR<-data.frame(Px2=Px1[,2],
                     Px1=Px1[,1],
                     y1=y1[,1])
d1_2_PCR<-data.frame(Px2=Px1[,2],
                     Px1=Px1[,1],
                     y2=y1[,2])

par(mfrow=c(1,2))

g3d1_1_PCR<-scatterplot3d(d1_1_PCR,
              type="h",
              angle=80,
              highlight.3d = T)
fit1_1_PCR<-lm(y1~Px2+Px1-1,d1_1_PCR)
g3d1_1_PCR$plane3d(fit1_1_PCR)

g3d1_2_PCR<-scatterplot3d(d1_2_PCR,
              type="h",
              angle=80,
              highlight.3d = T)
fit1_2_PCR<-lm(y2~Px2+Px1-1,d1_2_PCR)
g3d1_2_PCR$plane3d(fit1_2_PCR)
```





この場合，y_1は‘$Px1$‘方向の傾きだけで，y_2$は`Px2`方向の傾きだけで説明できるような形となっている．

PLS的な回帰

主成分分析のスコアどうしでOLS回帰してみる．

```
lm(Py1~Px1-1)%>%
  coef()%>%
  round(4)
```

```
##        Py1_1   Py1_2
## Px1PC1   0.7  0.3465
## Px1PC2   1.3 -1.6791
```

$P_y = P_x B$ であり，Xの係数は定義通りで，B．　上の例だと・・・

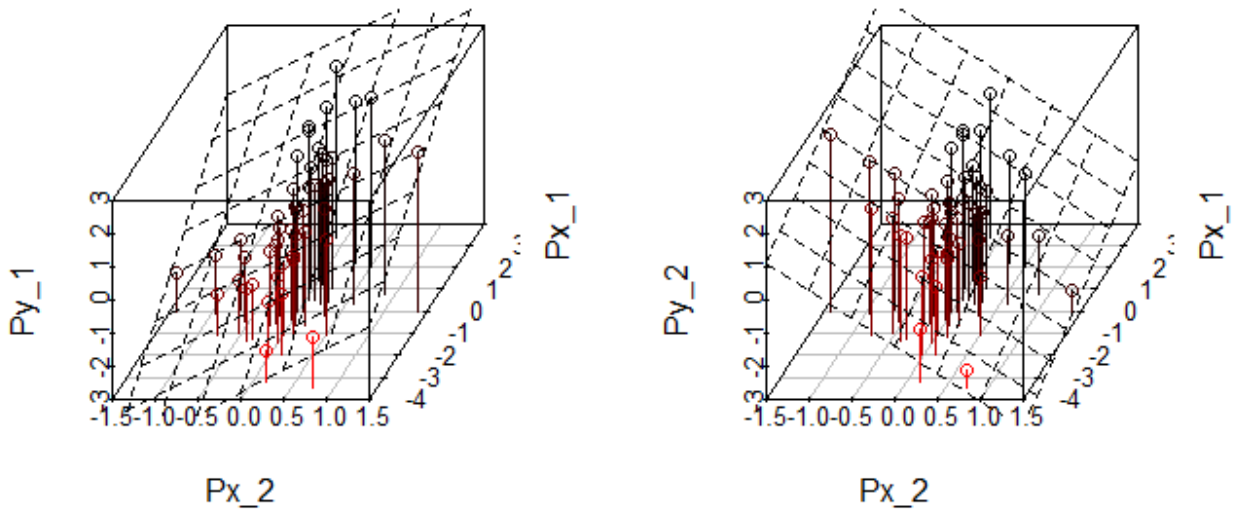
```
b0
```

```
##      [,1]      [,2]
## [1,]  0.7  0.346489
## [2,]  1.3 -1.679139
```

Yの第１主成分のスコアをXの第１主成分のスコアのみで説明していて，Yの第２主成分のスコアをXの第２主成分のスコアのみで説明している．

```
d1_1_scr<-data.frame(Px_2=Px1[,2],
                     Px_1=Px1[,1],
                     Py_1=Py1[,1])
d1_2_scr<-data.frame(Px_2=Px1[,2],
                     Px_1=Px1[,1],
                     Py_2=Py1[,2])
```

```r

par(mfrow=c(1,2))

g3d1_1_scr<-scatterplot3d(d1_1_scr,
              type="h",
              angle=60,
              highlight.3d = T)
fit1_1_scr<-lm(Py_1~Px_2+Px_1-1,d1_1_scr)
g3d1_1_scr$plane3d(fit1_1_scr)


g3d1_2_scr<-scatterplot3d(d1_2_scr,
              type="h",
              angle=60,
              highlight.3d = T)
fit1_2_scr<-lm(Py_2~Px_2+Px_1-1,d1_2_scr)
g3d1_2_scr$plane3d(fit1_2_scr)
```



しかし，PLSはこうした**主成分分析のスコアどうしのOLS回帰ではない**！

PLS回帰

PLSの考え方

YのスコアとXのスコアの内積$(Xu^*)'Yv^*$が最大化するように各スコアを決める．

YのスコアとXのスコアの空間で主成分分析しているのと同じ考え方．

しかし，このスコアはXとYをそれぞれ別々に主成分分析した場合のスコアとは違う．

$(Xu^*)'Yv^*$が最大化するようにu^*とv^*を決めた場合のスコア．

ただし，$u'u=1, v'v=1$に制約しておく．

式で表すと以下ということになる.

$\max_{u,v}(Xu)'Yv$

s.t. $u'u=1, v'v=1$

なので，ラグランジュ乗数法で

$\max_{u,v} L=(Xu)'Yv-\lambda(u'u-1)-\xi(v'v-1)$

最大化の一階の条件は，uで偏微分して・・・

$X'Yv=2\lambda u$

vで偏微分して・・・

$(X'Y)'u=2\xi v$

$\lambda=\xi$

であるなら，おお！これは特異値分解の必要条件そのものではないか！！！

・・・ということで，u,vは$X'Y$の特異値分解から得られる？

$X'Y$は・・・

```
xy1←t(x1)%*%y1
round(xy1,4)
```

```
##         [,1]    [,2]
## [1,] 45.1963 30.8037
## [2,] 48.9877  1.0123
```

$X'Y$の特異値分解をすると・・・

```
svd_xy1←svd(xy1)
print(svd_xy1)
```

```
## $d
## [1] 70.43287 20.77510
##
## $u
##            [,1]       [,2]
## [1,] -0.7518087 -0.6593812
## [2,] -0.6593812  0.7518087
##
## $v
##            [,1]       [,2]
## [1,] -0.9410458  0.3382792
## [2,] -0.3382792 -0.9410458
```

u^*とv^*が得られる・・・はず

```
u1←svd_xy1$u
v1←svd_xy1$v
d1←svd_xy1$d
```

（符号が逆方向で気持ち悪いので，逆にしておく）

```
u1←-u1
v1←-v1
```

ちなみに `$d` はXuとYvの内積

```
t(x1%*%u1)%*%y1%*%v1%>%
  round(4)
```

```
##         [,1]    [,2]
## [1,] 70.4329  0.0000
## [2,]  0.0000 20.7751
```

（$n-1=50$で割ると共分散）．

$X'Yv=2\lambda u$の2λの値は？

`$d` （つまり，XuとYvの内積）に一致する．

$X'Yv$が・・・

```
t(x1)%*%y1%*%v1%>%
  round(4)
```

```
##         [,1]    [,2]
## [1,] 52.9520  13.6987
## [2,] 46.4421 -15.6189
```

$2\lambda u$が・・・

```
u1%*%diag(d1)
```

```
##         [,1]     [,2]
## [1,] 52.95205  13.69871
## [2,] 46.44211 -15.61890
```

ぴったり．

$(X'Y)'u$が・・・

```
t(t(x1)%*%y1)%*%u1
```

```
##         [,1]     [,2]
## [1,] 66.28055 -7.027784
## [2,] 23.82597 19.550319
```

$2\xi v$が・・・

```
v1%*%diag(d1)
```

```
##          [,1]     [,2]
## [1,] 66.28055 -7.027784
## [2,] 23.82597 19.550319
```

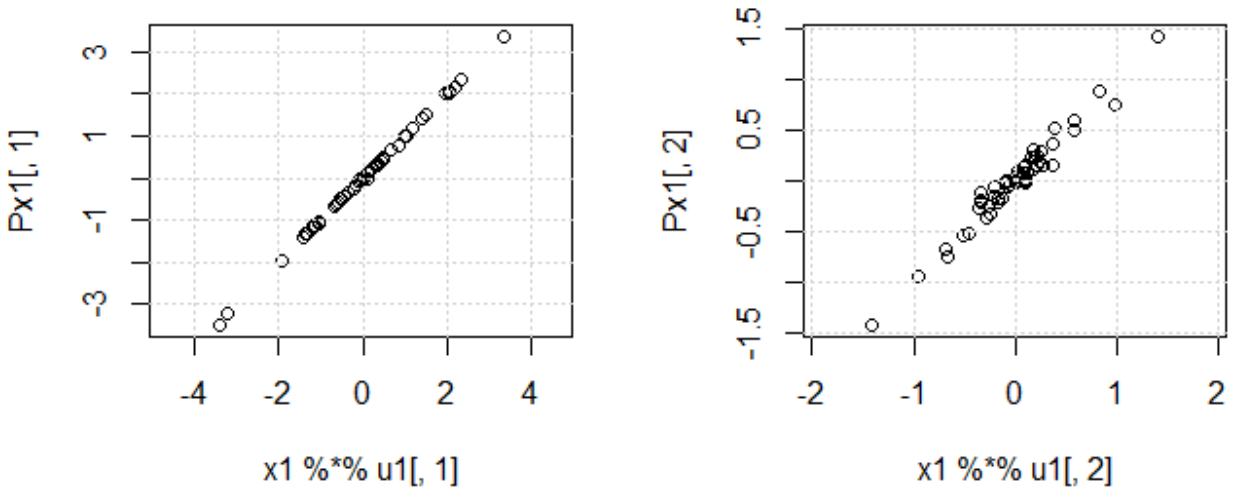
ぴったり．つまり，

$2\lambda = 2\xi = d$ ということでもある．

PCRとPLS

ノイズ（かく乱項）を入れていないので，XとYを，それぞれ主成分分析したスコアと，PLSの考え方で得られたスコアは一致するのか？

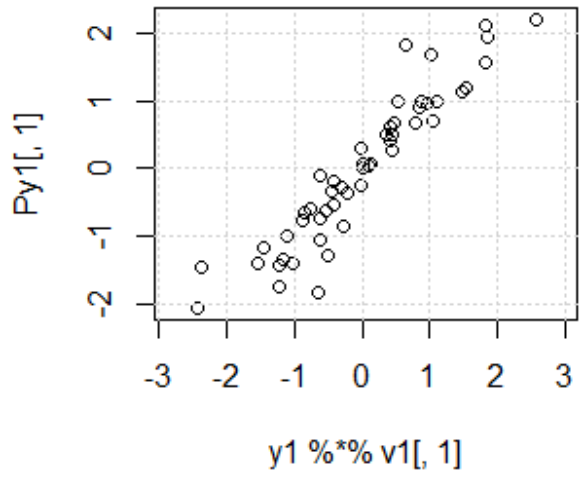
```
par(mfrow=c(1,2))
plot(Px1[,1]~x1%*%u1[,1],asp=1)
grid()
plot(Px1[,2]~x1%*%u1[,2],asp=1)
grid()
```

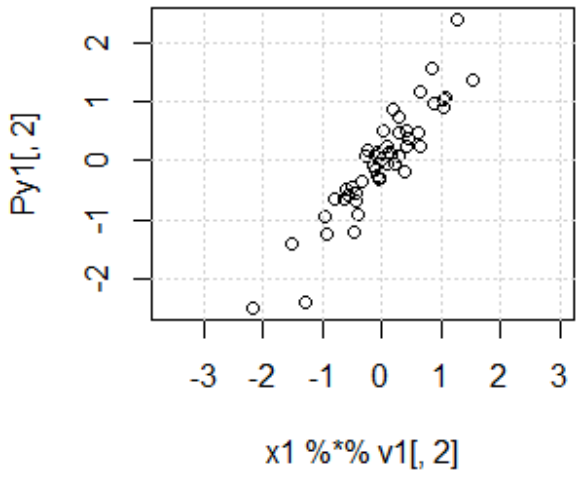


おおよそは一致しているが，**ぴったりとは一致していない**．

Yではさらにぶれる．

```
par(mfrow=c(1,2))
plot(Py1[,1]~y1%*%v1[,1],asp=1)
grid()
plot(Py1[,2]~x1%*%v1[,2],asp=1)
grid()
```





PLSの考え方で最大化された$(Xu^*)'Yv^*$が・・・

```
xyPx1←x1%*%u1
xyPy1←y1%*%v1
t(xyPx1)%*%xyPy1%>%
  round(4)
```

```
##         [,1]    [,2]
## [1,] 70.4329  0.0000
## [2,]  0.0000 20.7751
```

主成分分析で，XとYで個別に抽出した主成分の内積は・・・

```
t(Px1)%*%Py1%>%
  round(4)
```

```
##      Py1_1    Py1_2
## PC1     63  31.1840
## PC2     13 -16.7914
```

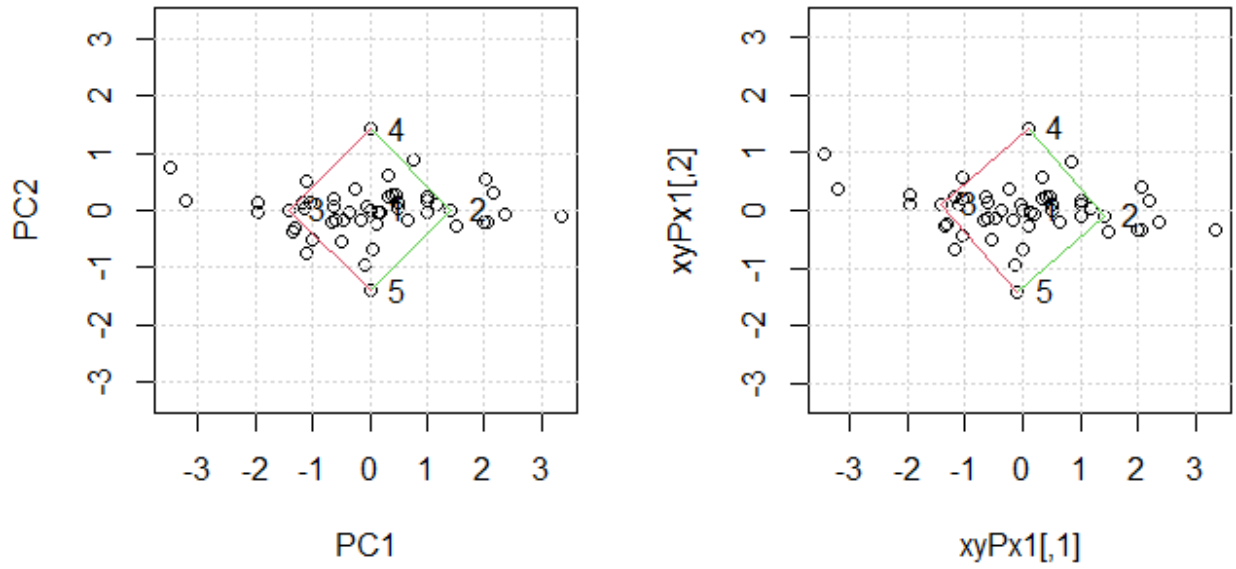
PLSの内積よりも小さい！

主成分分析では，XとYそれぞれに，最も分散の大きな主成分を抽出しているはずだが，XとYとの関係についての情報を含んでいないので，全体としては（PLSの意味での）最適ではないということ．

次のプロットは，左がPCRのスコア．右がPLSのスコア．

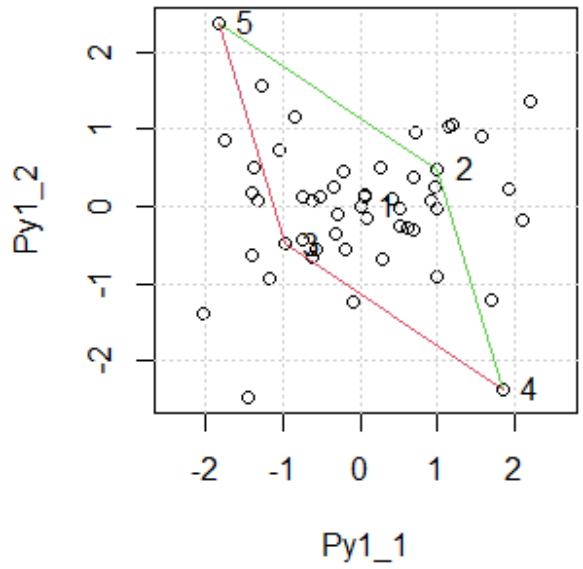
Xのスコアはあまり変わらないが・・・

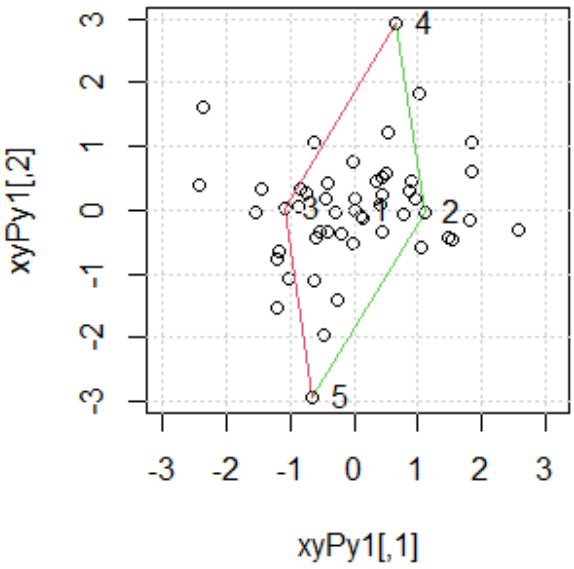
```
par(mfrow=c(1,2))
plot(Px1,asp=1)
grid()
text(x=Px1[1:5,1],y=Px1[1:5,2],1:5,pos=4)
f_grid1(Px1)

plot(xyPx1,asp=1)
grid()
text(x=xyPx1[1:5,1],y=xyPx1[1:5,2],1:5,pos=4)
f_grid1(xyPx1)
```



Yのスコアは，よりXに直交する形でとられているように見える．

```
par(mfrow=c(1,2))
plot(Py1,asp=1)
grid()
text(x=Py1[1:5,1],y=Py1[1:5,2],1:5,pos=4)
f_grid1(Py1)

plot(xyPy1,asp=1)
grid()
text(x=xyPy1[1:5,1],y=xyPy1[1:5,2],1:5,pos=4)
f_grid1(xyPy1)
```





$X'Y$ののスコアは直交していない

```
var(xyPx1)%>%
  round(4)
```

```
##         [,1]    [,2]
## [1,]  1.7932 -0.1043
## [2,] -0.1043  0.2068
```

```
var(xyPy1)%>%
  round(4)
```

```
##        [,1]   [,2]
## [1,] 1.1401 0.1696
## [2,] 0.1696 0.8599
```

スコアとX,Yとの関係も，45度回転ではない.

```
xya1_x<-lm(xyPx1~x1-1)%>%
  coef()
xya1_x
```

```
##          [,1]       [,2]
## x11 0.7518087  0.6593812
## x12 0.6593812 -0.7518087
```

```
xya1_y<-lm(xyPy1~y1-1)%>%
  coef()
xya1_y
```

```
##          [,1]       [,2]
## y11 0.9410458 -0.3382792
## y12 0.3382792  0.9410458
```

スコアどうしの関係も定義したBとは異なる

```
xyb1←lm(xyPy1~xyPx1-1)%>%
  coef()
xyb1
```

```
##             [,1]      [,2]
## xyPx11 0.8093295 0.1204335
## xyPx12 0.4083002 2.0696220
```

```
b0
```

```
##      [,1]      [,2]
## [1,]  0.7  0.346489
## [2,]  1.3 -1.679139
```

XとYの主成分どうしの関係はこちら側で定義して作成したのに・・・
$X \to X$のスコア，$Y \to Y$のスコア，$X \to Y$のスコア
もOLSで評価すると間違いないのに，PLSで一緒に分析すると，（PLSの意味で）もっとよい解が見つかった！・・・ということ．

ただし，PLSの係数を使ってYに直接Xを回帰したXの係数にあたるABA^{-1}を計算すると，・・・

```
xya1_x%*%xyb1%*%solve(xya1_y)
```

```
##           y11       y12
## x11 0.3336750  1.666325
## x12 0.7128141 -1.312814
```

（途中が違っても）PCRの場合と（OLSとも）一致する．

```
lm(y1~x1-1)%>%
  coef()
```

```
##          [,1]      [,2]
## x11 0.3336750  1.666325
## x12 0.7128141 -1.312814
```

（ノイズが一切ないので当然だが）

Rの`pls`パッケージを使ってみる．

データ

データは，data.frameにするが，XとYをそれぞれmatrixとして，2変数data.frameにするという荒業を使う．

matrixの`x1`と`y1`をそのままdata.frameに突っ込むには，`I()`関数（そのまんまにしてね！という関数）を使う．

それぞれ`x`と`y`と名前を付けよう．

```
d01←data.frame(x=I(x1),y=I(y1))
```

データ構造を見てみると・・・

```
str(d01)
```

```
## 'data.frame':    51 obs. of  2 variables:
##  $ x: 'AsIs' num [1:51, 1:2] 0 1 -1 1 -1 ...
##  $ y: 'AsIs' num [1:51, 1:2] -1.06e-18 1.05 -1.05 -3.79e-01 3.79e-01 ...
```

51行の2変数で，`x`と`y`があり，それぞれ，`[1:51,1:2]`の変数であると説明にある．

pls回帰

回帰には，`plsr()`関数を使う． 引数は，`formula=`の後に・・・
`ncomp=`は抽出する主成分の数（この場合，2変数しかないので，それ以上は無理）
`data=`はさっきつくったdata.frame
`validation=`は，クロスバリデーション（CV）の方法．“LOO”がLeave-one-out法．この場合，ややこしいので`="none"`（CV）にした．

```
fit01←plsr(y~x,
           ncomp=2,
           data=d01,
           validation="none")
```

要約を表示しても，主成分の寄与率しか表示してくれない．

```
summary(fit01)
```

```
## Data:    X dimension: 51 2
##  Y dimension: 51 2
## Fit method: kernelpls
## Number of components considered: 2
## TRAINING: % variance explained
##    1 comps  2 comps
## X    89.96      100
## Y1   98.00      100
## Y2   12.66      100
```

PLS回帰を使う人が，回帰係数よりも，たくさんの説明変数の中で**どの変数との関係が強いか**に関心がある場合に使われているということか？

適合度とモデル選択

適合度は，Root Mean Squared Error of Prediction (RMSEP) で表す.
なんか長いが，要は，実測値y_iと予測値$\hat{y}_i$との差の２乗平均の平方根をとったもの．当然小さい方がよいですね.

$$RMSEP = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y})^2}$$

`RMSEP`関数が用意されている.

```
RMSEP(fit01)
```

```
##
## Response: Y1
## (Intercept)      1 comps      2 comps
##   9.901e-01    1.401e-01    1.871e-16
##
## Response: Y2
## (Intercept)      1 comps      2 comps
##   9.901e-01    9.253e-01    8.450e-16
```

`(Intercept)`は，何も説明変数を使わなかったYのRMSEP.ほぼ標準偏差ということになるが，$n-1$ではなく，nで割られるので，1となっていない.

```
sd(y1[,1])*sqrt(50/51)
```

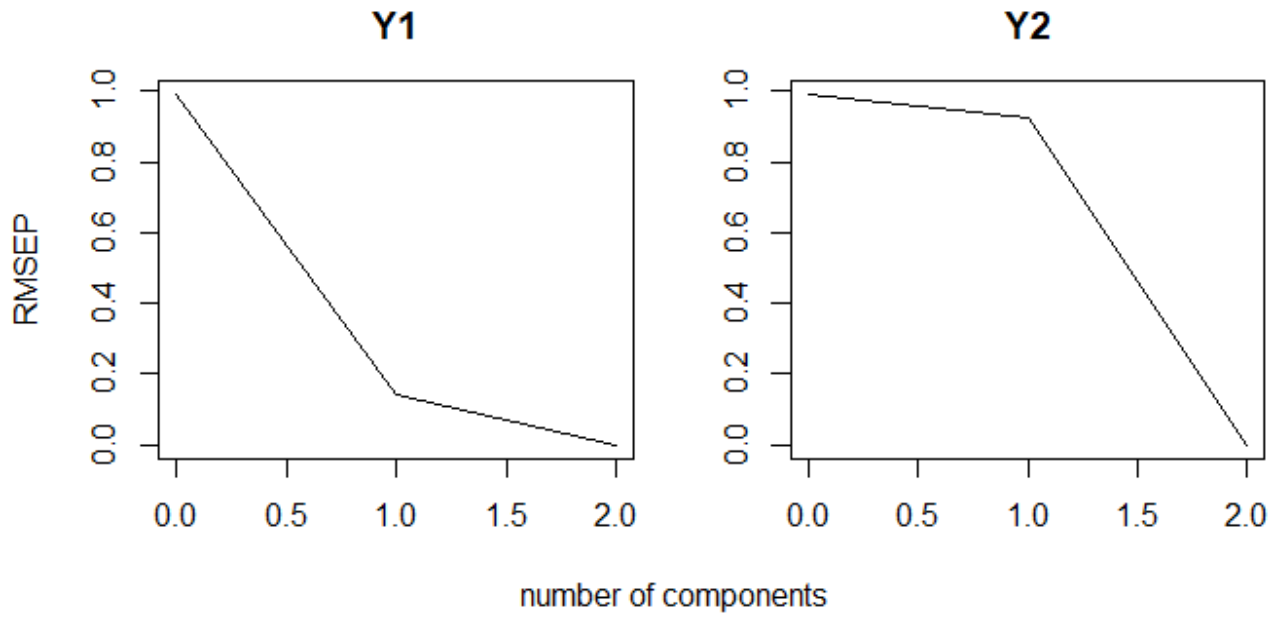
```
## [1] 0.9901475
```

`1 comps`は，第１主成分だけを使った場合のRMSEP.

`2 comps`は，第２主成分まで使った場合のRMSEP.　２変数しかないので，当然ここで0となる.

プロットもできる.

```
plot(RMSEP(fit01))
```



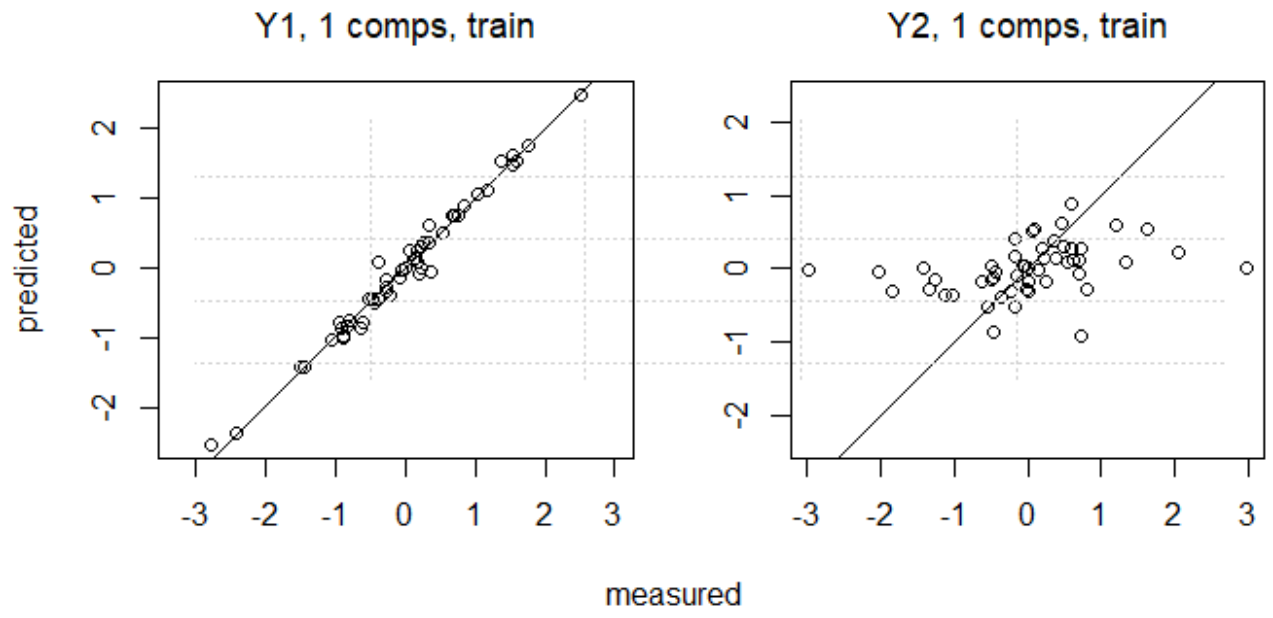
フィットの様子を確認することもできる.

横軸は抽出した主成分の数だが，1.2とか，余計な目盛がついているので，それは無視．主成分名を数字に置き換えてあるので，主成分の数が少ないとこうなるみたいだ.

そのまま，`plot()`関数を使うと，Yの実際値と予測値との相関を出力する.

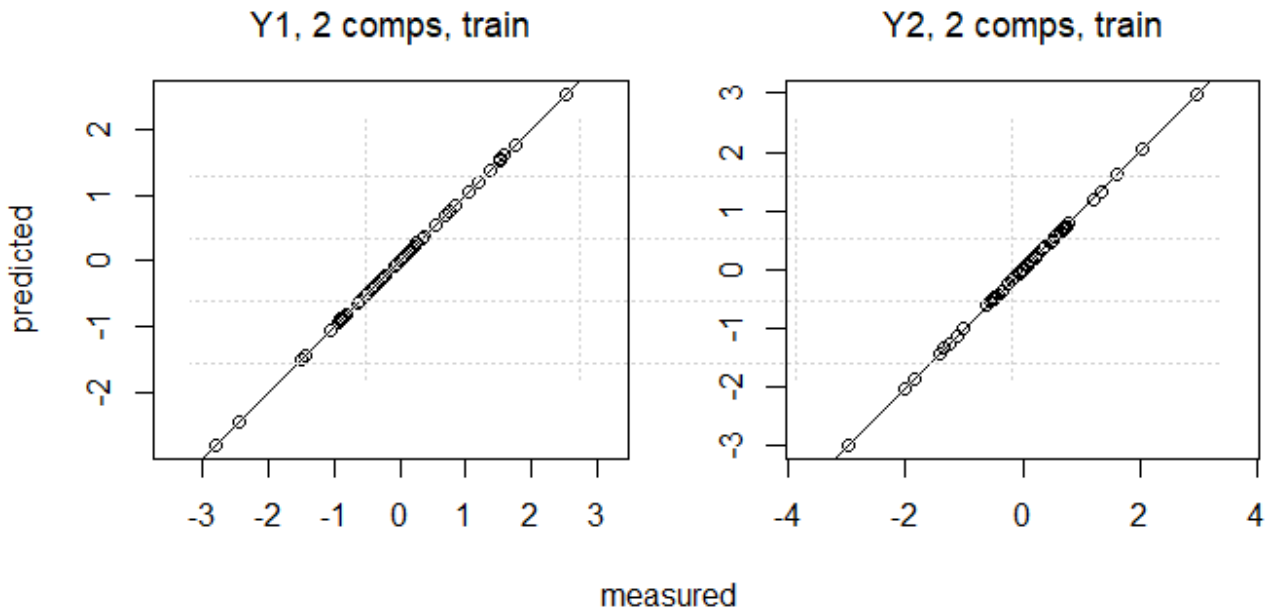
第１主成分だけで予測した場合

```
plot(fit01,ncomp=1,line=TRUE,asp=1)
grid()
```



第２主成分も使って予測した場合

```
plot(fit01,ncomp=2,line=TRUE,asp=1)
grid()
```



ちなみに，予測値は`predict`関数で出力できる．

```
predict(fit01,comps=1)%>%
  round(.,4)%>%
  head()
```

```
##        Y1      Y2
## 1  0.0000  0.0000
## 2  1.0432  0.3750
## 3 -1.0432 -0.3750
## 4  0.0683  0.0246
## 5 -0.0683 -0.0246
## 6 -0.7659 -0.2753
```

主成分のスコア

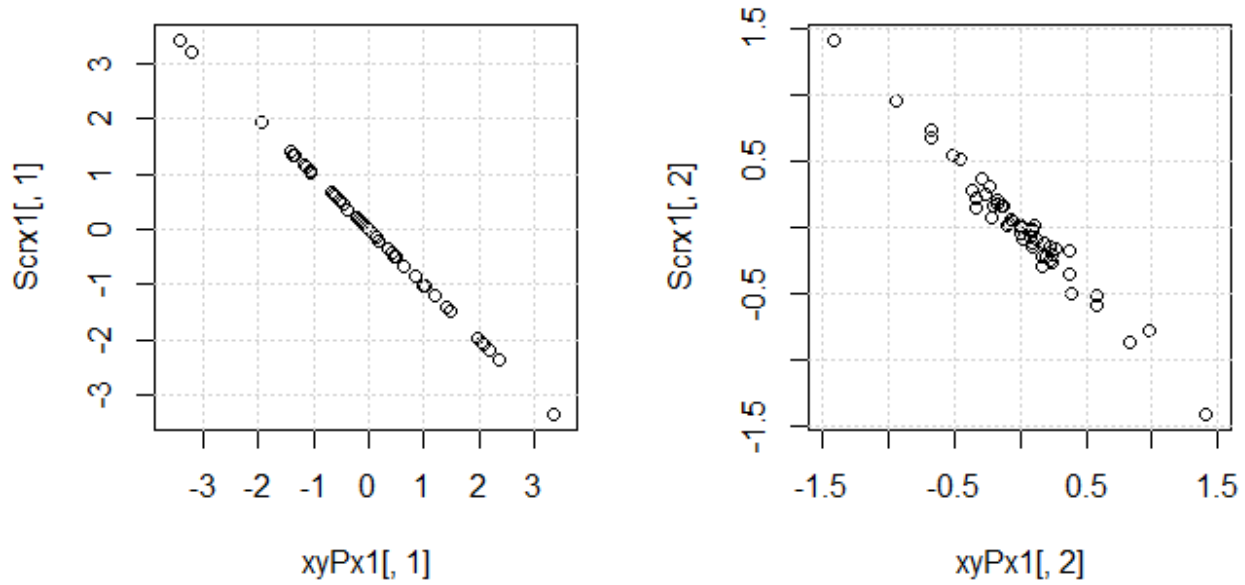
Xのスコアは`scores()`関数で

```
Scrx1←scores(fit01)
Scrx1%>%
  round(4)%>%
  head()
```

```
##    Comp 1  Comp 2
## 1  0.0000  0.0000
## 2 -1.4112  0.0103
## 3  1.4112 -0.0103
## 4 -0.0924 -1.4166
## 5  0.0924  1.4166
## 6  1.0361 -0.1252
```

XのスコアはXu^*から出した`xyScrx1`に（符号を除いて）一致するはずだが，

```
par(mfrow=c(1,2))
plot(Scrx1[,1]~xyPx1[,1],asp=1)
grid()
plot(Scrx1[,2]~xyPx1[,2],asp=1)
grid()
```



第２主成分で多少の違いが・・・

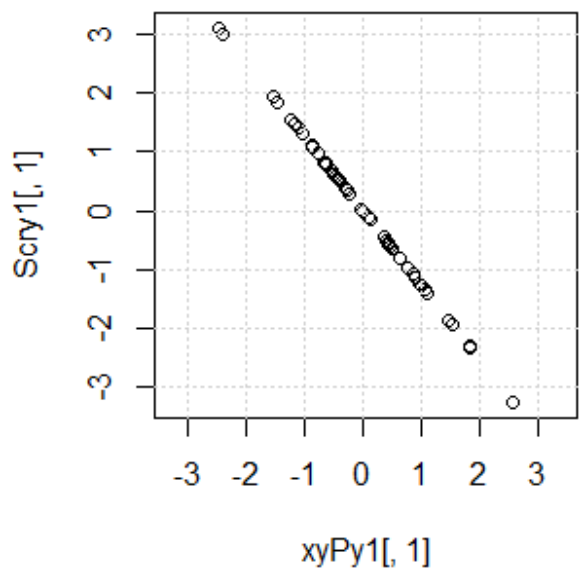
Yのスコアは`Yscores()`関数で

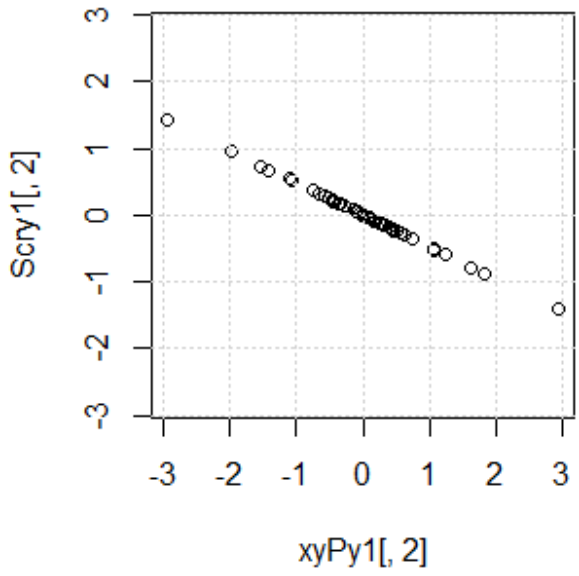
```
Scry1<-Yscores(fit01)
Scry1%>%
  round(4)%>%
  head()
```

```
##    Comp 1  Comp 2
## 1  0.0000  0.0000
## 2 -1.4058  0.0103
## 3  1.4058 -0.0103
## 4 -0.8287 -1.4166
## 5  0.8287  1.4166
## 6  0.9710 -0.1252
```

Yについては，比例はするが，スケールが全然ちがう． なぜ？？？？

```
par(mfrow=c(1,2))
plot(Scry1[,1]~xyPy1[,1],asp=1)
grid()
plot(Scry1[,2]~xyPy1[,2],asp=1)
grid()
```





`plsr()` がYのスコアのばらつきをXのスコアのばらつきに対応させている？

```
lm(Yscores(fit01)~scores(fit01)-1)%>%
  coef()%>%
  round(4)
```

```
##                     Comp 1 Comp 2
## scores(fit01)Comp 1 1.0000      0
## scores(fit01)Comp 2 0.5198      1
```

Yのスコアの分散は

```
var(Scry1)%>%
  round(4)
```

```
##        Comp 1 Comp 2
## Comp 1 1.8474 0.1043
## Comp 2 0.1043 0.2008
```

$X'Y$から求めたYのスコアの分散は

```
var(xyPy1)%>%
  round(4)
```

```
##        [,1]   [,2]
## [1,] 1.1401 0.1696
## [2,] 0.1696 0.8599
```

`plsr` で求めたYの第１主成分のスコアは，約1.27倍されている．

```
sqrt(1.8474/1.1401)
```

```
## [1] 1.272943
```

第2主成分のスコアは約0.48倍されている

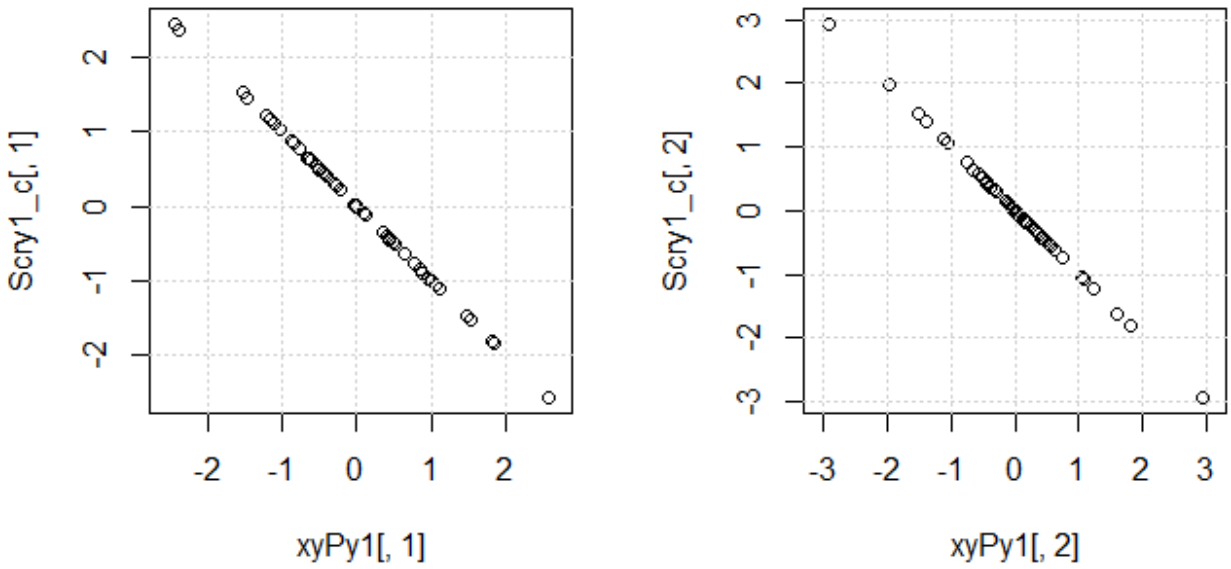
```
sqrt(0.2008/0.8599)
```

```
## [1] 0.4832344
```

`plsr`の2つのスコアを，それぞれ割ってやると，X‘Yの最大化から求めたYのスコアとほぼ一致する．

```
Scry1_c<-sweep(Scry1,2,c(sqrt(1.8474/1.1401),sqrt(0.2008/0.8599)),FUN="/")
```

```
par(mfrow=c(1,2))
plot(Scry1_c[,1]~xyPy1[,1],asp=1)
grid()
plot(Scry1_c[,2]~xyPy1[,2],asp=1)
grid()
```

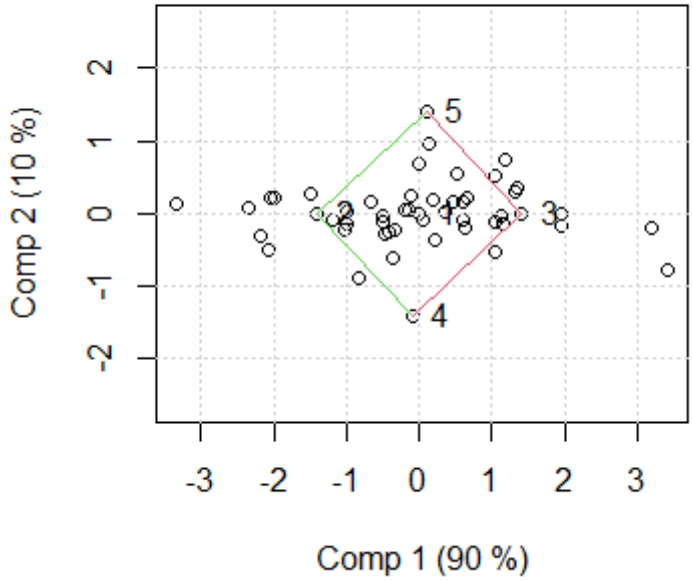


なんでそんなことするかは謎？

Xのスコアのプロット

`plot`関数に引数`plottype="scores"`で．　`scoreplot()`関数でもOK

```
plot(fit01,plottype="scores",comps=1:2,asp=1)
grid()
text(x=Scrx1[1:5,1],y=Scrx1[1:5,2],1:5,pos=4)
f_grid1(Scrx1)
```



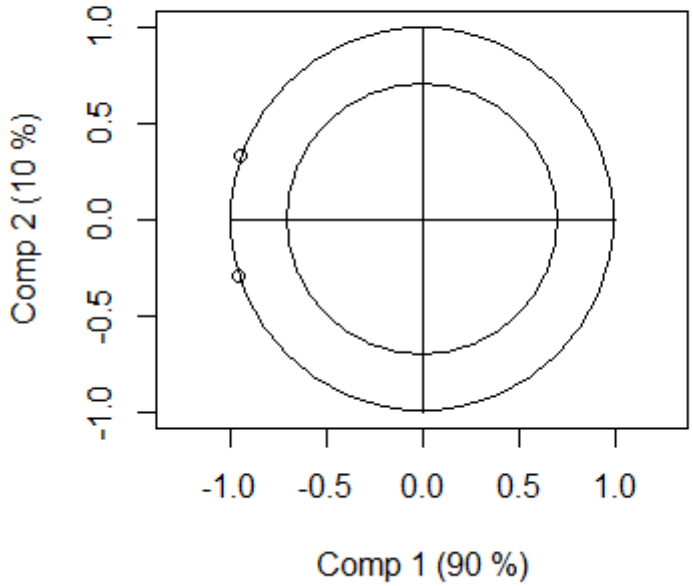
多変数の場合は，`pairs()`の図が出力されるようだ・・・

Yのスコアは関数が用意されていないが，この場合Xのスコアと全く同じなので，あんまり意味ない．

スコアとXとの相関のプロット

`plot()`関数の`plottype="correlation"`引数でも同じ．

```
1 corrplot(fit01)
```



```
cor(x1,Scrx1)%>%
  round(4)
```

```
##       Comp 1  Comp 2
## [1,] -0.9554 -0.2954
## [2,] -0.9416  0.3369
```

・・・というのをプロットしているようだ．左下がx_1で，左上がx_2．

x_1もx_2も，第１主成分の方に相関が高いようだ・・・ということはわかる．

PLSのOLSやPCRとの違い

以上は全くノイズの無い世界の話．ノイズがなければ，OLSでXを直接Yに回帰させても，あるいは，主成分回帰(PCR)で，いったんXの主成分を抽出して，主成分をYに回帰させても，それはどう解釈するかが違うだけで，情報の損失はない．

しかし，Xは2変数よりもすごく多いと，そうはいかなくなる．

まず，OLSだとXの変数間でマルチコが生じる可能性が高まる．

だからといって，PCRを使うとすると，Xから主成分を抽出して，**主な**主成分を２個とか3個とか選んで使うことになる．

ところが，この主成分は，あくまでもXの中だけの情報で抽出されるし，どれがどれが**主な**主成分かもXの中だけで決まる．これにはYの情報は入らない．

しかし，PLSだとYの情報も入れて主成分を抽出するので，PCRよりもうまく主成分を抽出できる・・・という話．

以下，ノイズを入れてやってみよう．

2×2変数にノイズあり

データ作成

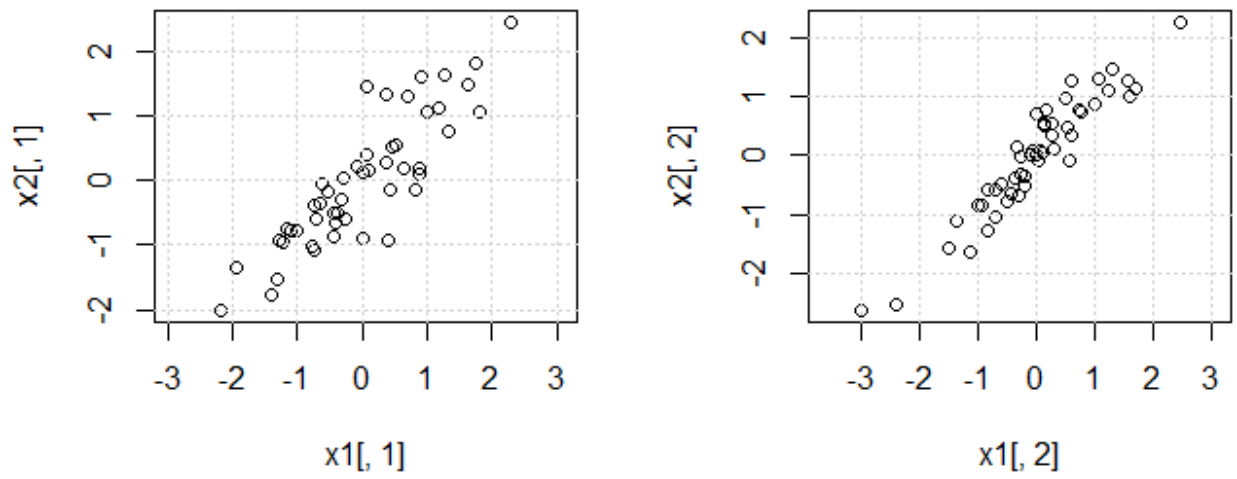
攪乱項を加える

2変数×2変数というのは変えないで，上記で作成した`x1`や`y1`にちょっとだけかく乱項を入れてみる．

```
x2←x1
set.seed(111)
x2[6:51,1]←x1[6:51,1]+rnorm(46,mean=0,sd=0.5)
set.seed(124)
x2[6:51,2]←x1[6:51,2]+rnorm(46,mean=0,sd=0.5)
x2←apply(x2,2,scale)
```

Xにどのくらいノイズが入ったかプロットしてみる

```
par(mfrow=c(1,2))
plot(x2[,1]~x1[,1],asp=1)
grid()
plot(x2[,2]~x1[,2],asp=1)
grid()
```



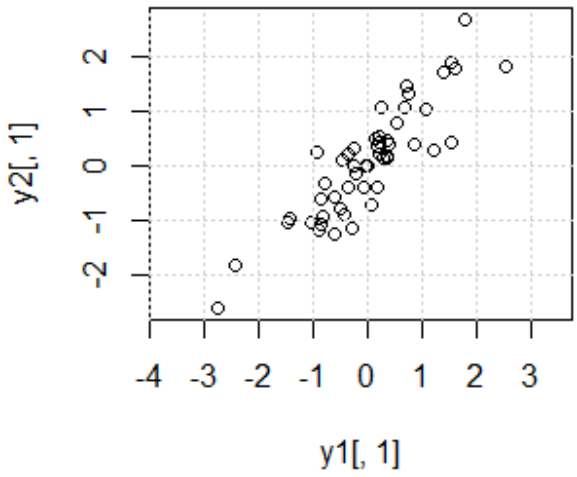
このぐらい・・・

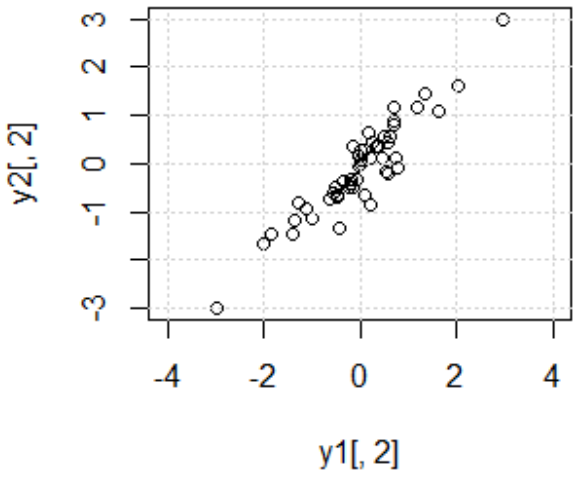
Yにもノイズを加える

```
y2←y1
set.seed(125)
y2[6:51,1]←y2[6:51,1]+rnorm(46,0,0.5)
set.seed(126)
y2[6:51,2]←y2[6:51,2]+rnorm(46,0,0.5)
```

Yにどれだけノイズが入ったか見てみよう

```
par(mfrow=c(1,2))
plot(y2[,1]~y1[,1],asp=1)
grid()
plot(y2[,2]~y1[,2],asp=1)
grid()
```





このぐらい.

OLSやPCRとの比較

OLS回帰

作成したXとYでOLS回帰してみる.

```
lm(y2~x2-1)%>%
  coef()
```

```
##          [,1]       [,2]
## x21 0.3545037  0.9186370
## x22 0.5574486 -0.6727718
```

真の値はこれ・・・

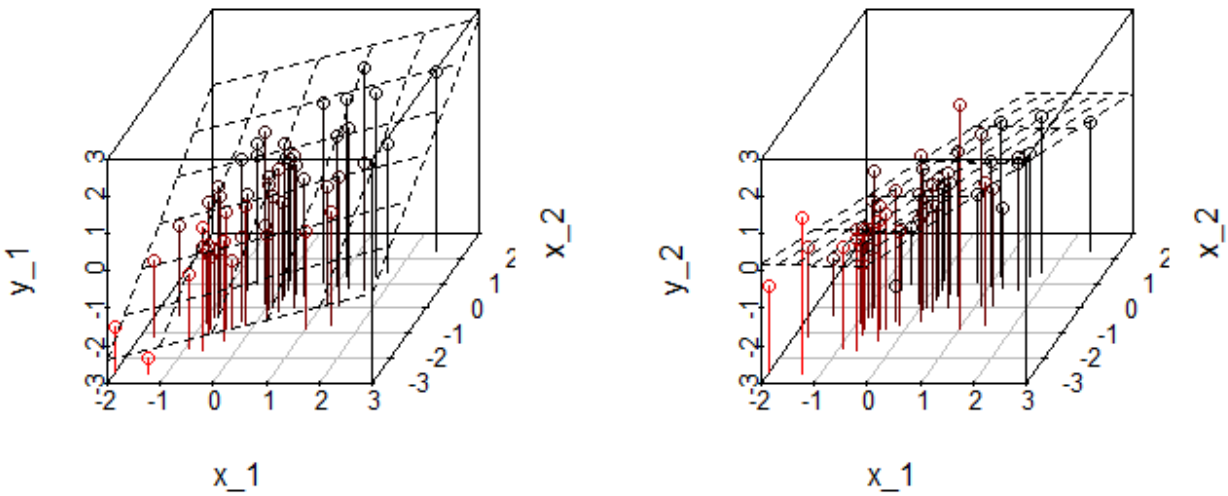
```
a1%*%b0%*%solve(a1)
```

```
##           [,1]      [,2]
## [1,] 0.3336750  1.666325
## [2,] 0.7128141 -1.312814
```

y_1のx_1の係数以外はあてはまりがいまいち．こういうのは，乱数を多数発生させて，推定値の分布を見て判断すべきで，１つの例でどうのこうの言えることではないのだが，y_1をプロットしてみると・・・

```
d2_1_OLS<-data.frame(x_1=x2[,1],
                     x_2=x2[,2],
                     y_1=y2[,1])
d2_2_OLS<-data.frame(x_1=x2[,1],
                     x_2=x2[,2],
                     y_2=y2[,2])
par(mfrow=c(1,2))
g3d1_1_OLS<-scatterplot3d(d2_1_OLS,
```

```
              type="h",
              angle=60,
              highlight.3d = T)
fit2_1_OLS<-lm(y_1~x_1+x_2-1,d2_1_OLS)
g3d1_1_OLS$plane3d(fit2_1_OLS)

g3d1_2_OLS<-scatterplot3d(d2_2_OLS,
              type="h",
              angle=60,
              highlight.3d = T)
fit2_2_OLS<-lm(y_2~x_1+x_2-1,d2_2_OLS)
g3d1_2_OLS$plane3d(fit2_2_OLS)
```



Xの相関に対して回帰平面が不安定そうなのは確認できる.

係数のt値では，そんな不安定であるようには見えないのが問題.

```
lm(y2~x2-1)%>%
  summary()%>%
  coefficients()
```

```
## Response Y1 :
##      Estimate Std. Error  t value     Pr(>|t|)
## x21 0.3545037  0.1118888 3.168358 2.640143e-03
## x22 0.5574486  0.1118888 4.982167 8.223860e-06
##
## Response Y2 :
##       Estimate Std. Error   t value     Pr(>|t|)
## x21  0.9186370  0.1255169  7.318831 2.130580e-09
## x22 -0.6727718  0.1255169 -5.360010 2.226719e-06
```

PCR 回帰

Xを主成分分析

```
prc_x2←prcomp(x2)
prc_x2
```

```
## Standard deviations (1, .., p=2):
## [1] 1.2830922 0.5947054
##
## Rotation (n x k) = (2 x 2):
##             PC1        PC2
## [1,] -0.7071068  0.7071068
## [2,] -0.7071068 -0.7071068
```

主成分分析のスコア（方向を逆に）

```
Px2←prc_x2$x
Px2[,1]←-prc_x2$x[,1]
```

YをXのスコアで回帰

```
lm(y2~Px2-1)%>%
  coef()%>%
  round(4)
```

```
##          [,1]   [,2]
## Px2PC1  0.6448 0.1739
## Px2PC2 -0.1435 1.1253
```

真の値は・・・

```
b0%*%solve(a1)
```

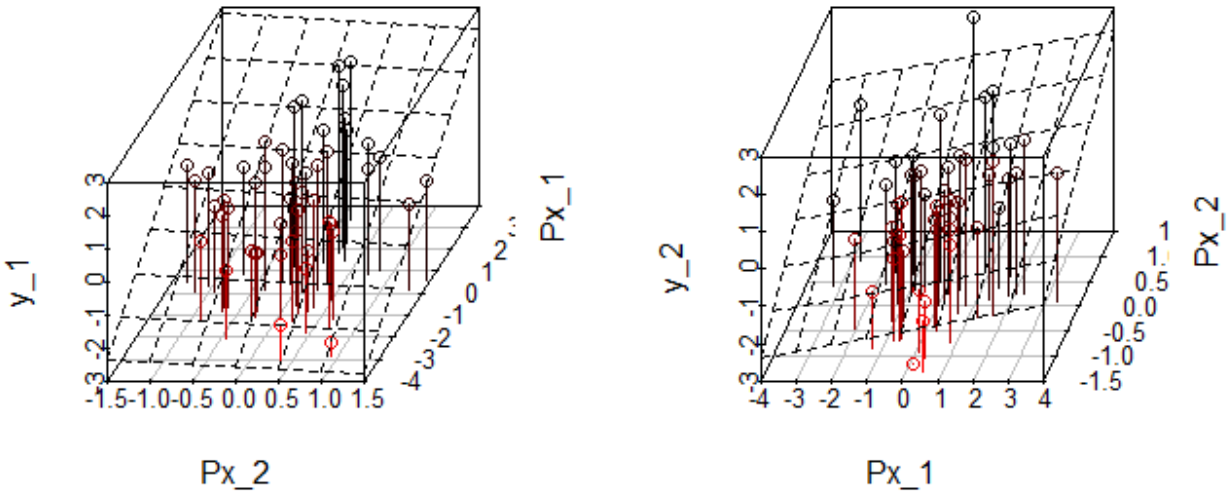
```
##           [,1]     [,2]
## [1,]  0.7399795 0.249970
## [2,] -0.2680918 2.106569
```

これも１つの乱数だけで云々できないが，y_2に対する係数がだいぶ外れているようだ．

プロットしてみると・・・

```
d2_1_PCR←data.frame(Px_2=Px2[,2],
                    Px_1=Px2[,1],
                    y_1=y2[,1])
d2_2_PCR←data.frame(Px_1=Px2[,1],
                    Px_2=Px2[,2],
                    y_2=y2[,2])
par(mfrow=c(1,2))

g3d1_1_PCR←scatterplot3d(d2_1_PCR,
              type="h",
              angle=60,
```

```
              highlight.3d = T)
fit2_1_PCR<-lm(y_1~Px_2+Px_1-1,d2_1_PCR)
g3d1_1_PCR$plane3d(fit2_1_PCR)

g3d1_2_PCR<-scatterplot3d(d2_2_PCR,
              type="h",
              angle=60,
              highlight.3d = T)
fit2_2_PCR<-lm(y_2~Px_1+Px_2-1,d2_2_PCR)
g3d1_2_PCR$plane3d(fit2_2_PCR)
```



説明変数の相関がましになったことは確認できる.

```
lm(y1~Px2-1)%>%
  summary()%>%
  coefficients()
```

```
## Response Y1 :
##          Estimate Std. Error   t value     Pr(>|t|)
## Px2PC1  0.7360053 0.03174716 23.183344 4.577494e-28
## Px2PC2 -0.2756515 0.06849531 -4.024385 1.975324e-04
##
## Response Y2 :
##         Estimate Std. Error  t value     Pr(>|t|)
## Px2PC1 0.2334963 0.07711698 3.027820 3.921328e-03
## Px2PC2 1.1032709 0.16638187 6.630956 2.475341e-08
```

PLS回帰

主成分の抽出

$X'Y$の特異値分解

```
svd_xy2<-svd(t(x2)%*%y2)
svd_xy2
```

```
## $d
## [1] 55.05462 19.84587
##
## $u
##            [,1]       [,2]
## [1,] -0.7462628 -0.6656515
## [2,] -0.6656515  0.7462628
##
## $v
##            [,1]       [,2]
## [1,] -0.9599648  0.2801207
## [2,] -0.2801207 -0.9599648
```

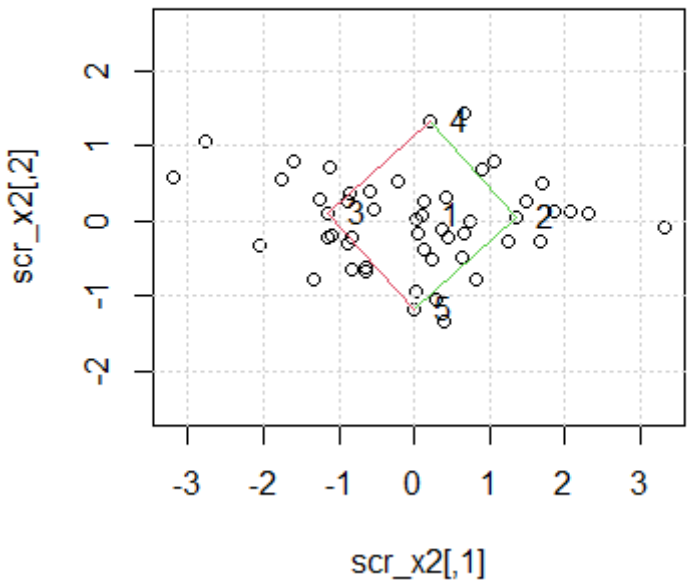
```
d2←svd_xy2$d
u2←-svd_xy2$u
v2←-svd_xy2$v
```

XとYの主成分

```
scr_x2←x2%*%u2
scr_y2←y2%*%v2
```

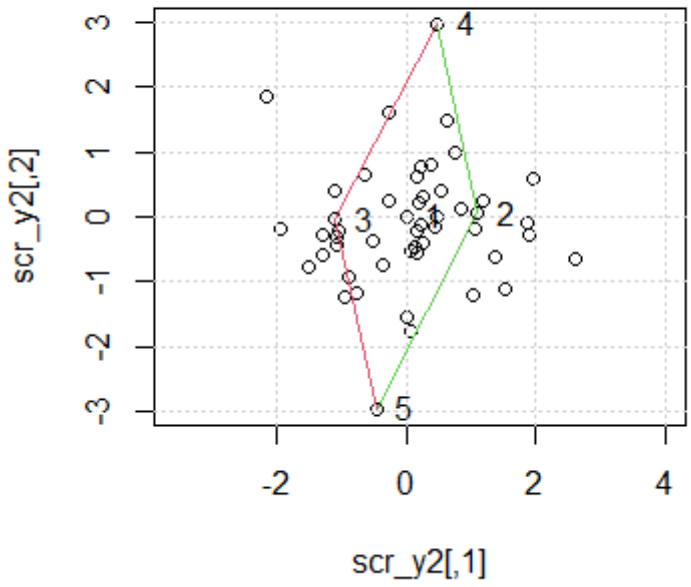
Xの主成分のスコア

```
plot(scr_x2,asp=1)
grid()
text(x=scr_x2[1:5,1],y=scr_x2[1:5,2],1:5,pos=4)
f_grid1(scr_x2)
```



Yの主成分のスコア

```
plot(scr_y2,asp=1)
grid()
text(x=scr_y2[1:5,1],y=scr_y2[1:5,2],1:5,pos=4)
f_grid1(scr_y2)
```

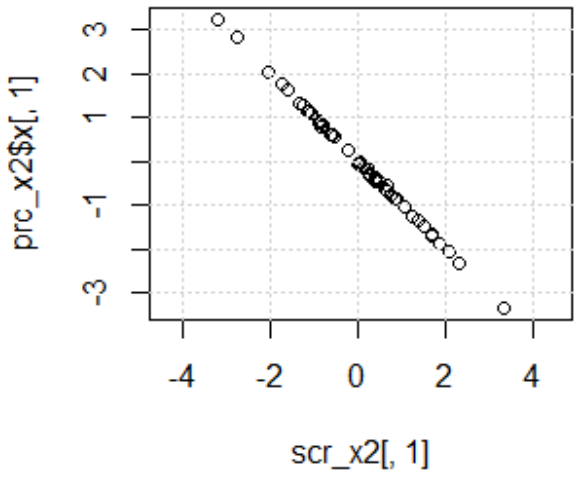


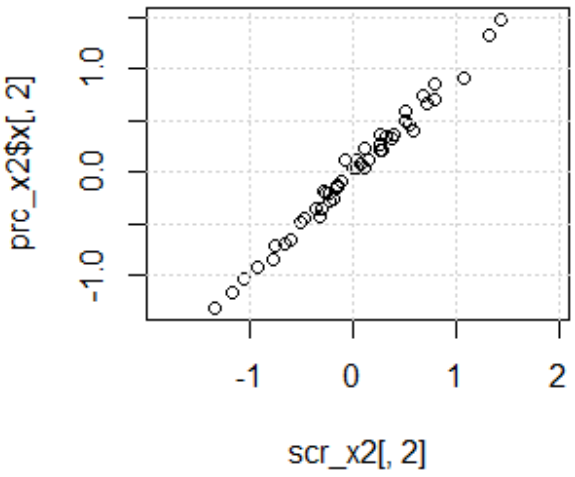
ちなみに，XとY，別々に主成分分析して主成分を抽出すると．．．

```
prc_x2←prcomp(x2)
prc_y2←prcomp(y2)
```

X‘Yからの主成分と比較すると，

```
par(mfrow=c(1,2))
plot(prc_x2$x[,1]~scr_x2[,1],asp=1)
grid()
plot(prc_x2$x[,2]~scr_x2[,2],asp=1)
grid()
```

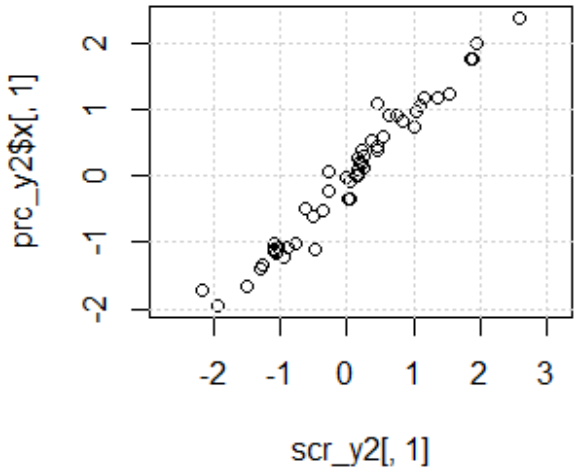


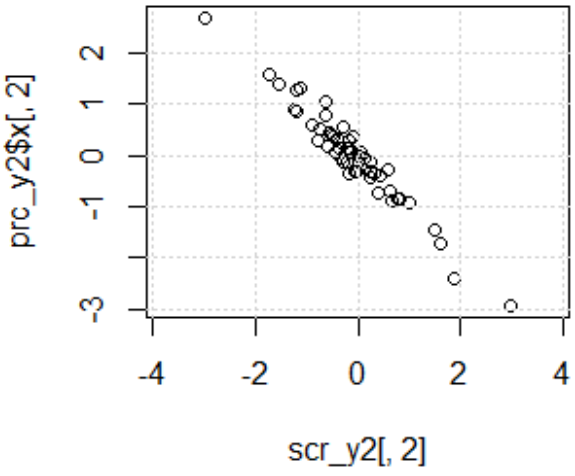


Xの主成分分析は若干ずれていることがわかる.

Yの主成分については，もっとずれてくる.

```
par(mfrow=c(1,2))
plot(prc_y2$x[,1]~scr_y2[,1],asp=1)
grid()
plot(prc_y2$x[,2]~scr_y2[,2],asp=1)
grid()
```





スコアどうしで回帰する

```
lm(scr_y2~scr_x2-1)%>%
  coef()
```

```
##              [,1]      [,2]
## scr_x21 0.6767605 0.05014599
## scr_x22 0.1391105 1.11940449
```

真値は

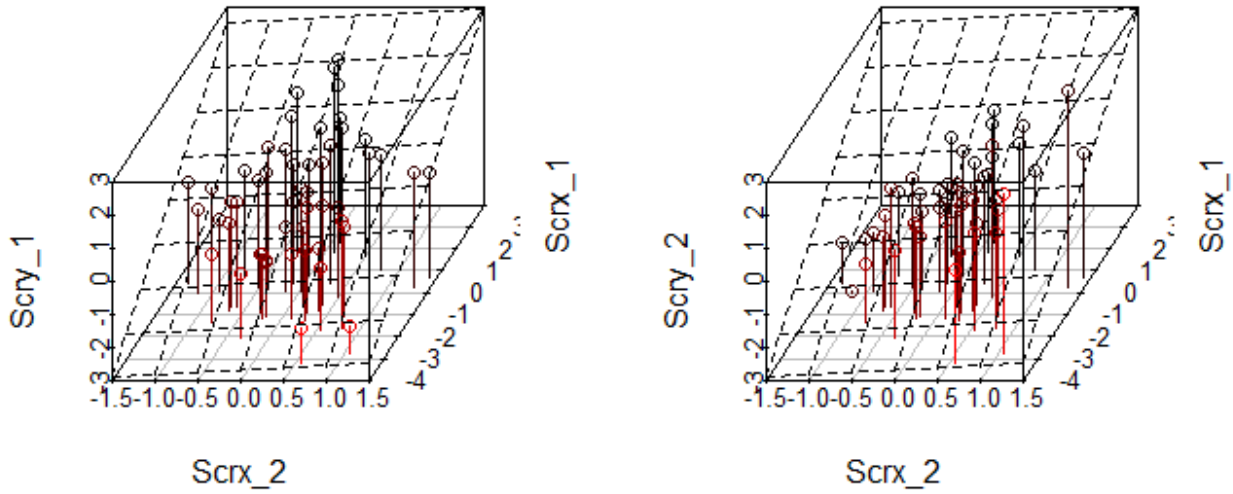
```
xyb1
```

```
##            [,1]     [,2]
## xyPx11 0.8093295 0.1204335
## xyPx12 0.4083002 2.0696220
```

第２主成分が今一つだが，第１主成分は安定している．

プロットしてみると・・・

```
d2_1_PLS<-data.frame(Scrx_2=scr_x2[,2],
                     Scrx_1=scr_x2[,1],
                     Scry_1=scr_y2[,1])
d2_2_PLS<-data.frame(Scrx_2=scr_x2[,2],
                     Scrx_1=scr_x2[,1],
                     Scry_2=scr_y2[,2])

par(mfrow=c(1,2))

g3d2_1_PLS<-scatterplot3d(d2_1_PLS,
              type="h",
              angle=60,
              highlight.3d = T)
fit2_1_PLS<-lm(Scry_1~Scrx_2+Scrx_1-1,d2_1_PLS)
g3d2_1_PLS$plane3d(fit2_1_PLS)

g3d2_2_PLS<-scatterplot3d(d2_2_PLS,
              type="h",
              angle=60,
              highlight.3d = T)
fit2_2_PLS<-lm(Scry_2~Scrx_2+Scrx_1-1,d2_2_PLS)
g3d2_2_PLS$plane3d(fit2_1_PLS)
```



被説明変数がある程度ばらついており，Yの傾向もPCRより確認できやすい気がする？

```
lm(scr_y2~scr_x2-1)%>%
  summary()%>%
  coefficients()
```

```
## Response Y1 :
##          Estimate Std. Error    t value     Pr(>|t|)
## scr_x21 0.6767605 0.06498591 10.4139573 5.147398e-14
## scr_x22 0.1391105 0.13920572  0.9993159 3.225513e-01
##
## Response Y2 :
##           Estimate Std. Error   t value     Pr(>|t|)
## scr_x21 0.05014599 0.07678037 0.6531095 5.167386e-01
## scr_x22 1.11940449 0.16447051 6.8061106 1.324866e-08
```

Yの第1主成分について，Xの第１主成分がだけが効いており，Yの第2主成分について，Xの第２主成分だけが効いている．

Yの主成分が２つXの主成分が２つの場合，内積を最大化しようとすると，主成分１対１で対応するような気がする（要確認）．

Rの`plsr()`関数で

データ作成

```
d02←data.frame(x=I(x2),y=I(y2))
```

PLS回帰

```
fit02<-plsr(y~x,
            ncomp=2,
            data=d02,
            validation = "none")
```

要約

```
summary(fit02)
```

```
## Data:    X dimension: 51 2
##  Y dimension: 51 2
## Fit method: kernelpls
## Number of components considered: 2
## TRAINING: % variance explained
##     1 comps  2 comps
## X    82.271   100.00
## Y1   65.185    66.29
## Y2    6.181    53.09
```

適合度とモデル選択

$$RMSEP = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y})^2}$$

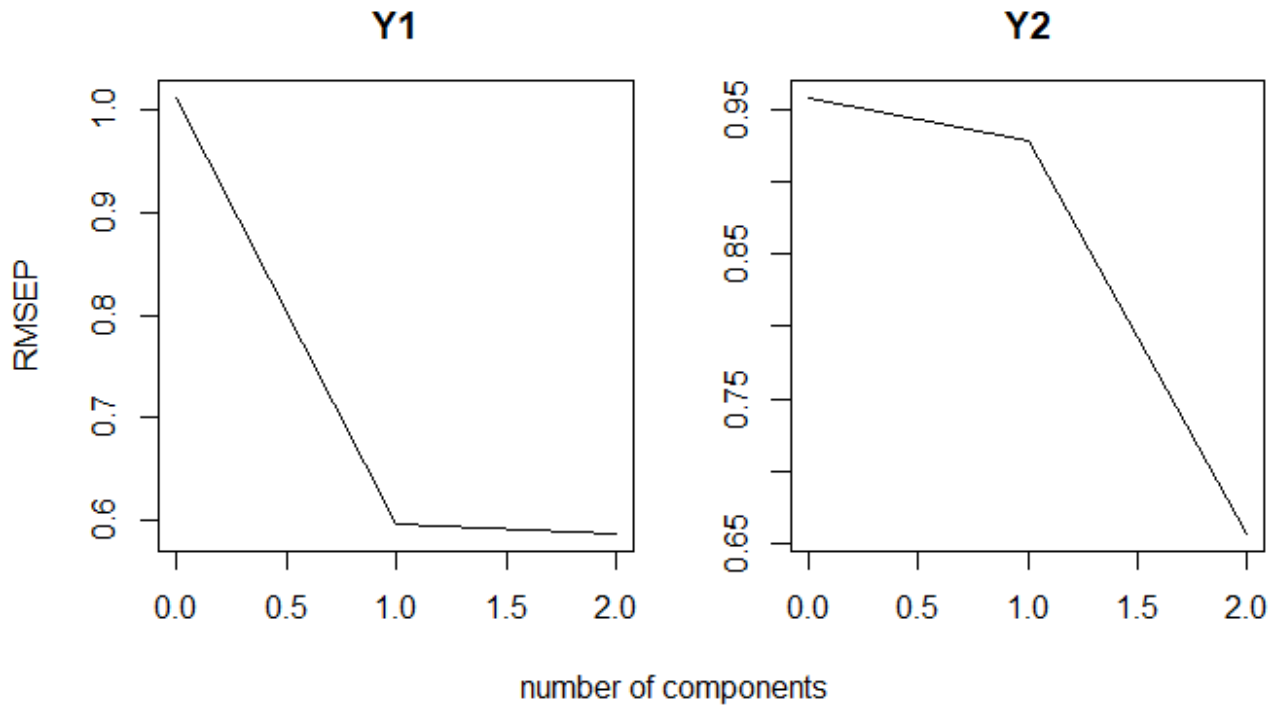
```
RMSEP(fit02)
```

```
##
## Response: Y1
## (Intercept)      1 comps      2 comps
##      1.0116       0.5969       0.5874
##
## Response: Y2
## (Intercept)      1 comps      2 comps
##      0.9586       0.9285       0.6565
```

`(Intercept)` は，何もXを使わなかったときのRMSEP.
`1 comps` は，第１主成分だけを使った場合のRMSEP.
`2 comps` は，第２主成分まで使った場合のRMSEP． ２変数しかないが，ノイズがあるので0にはならない.

プロット

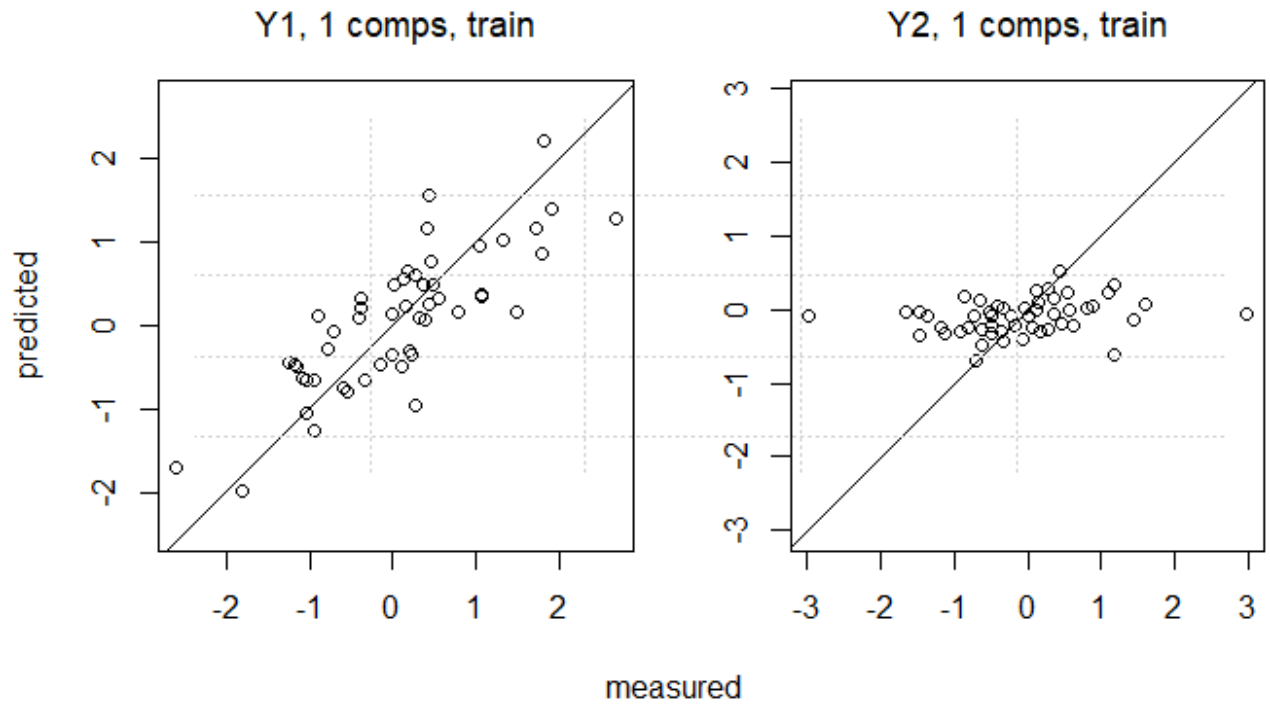
```
plot(RMSEP(fit02))
```



フィットの様子

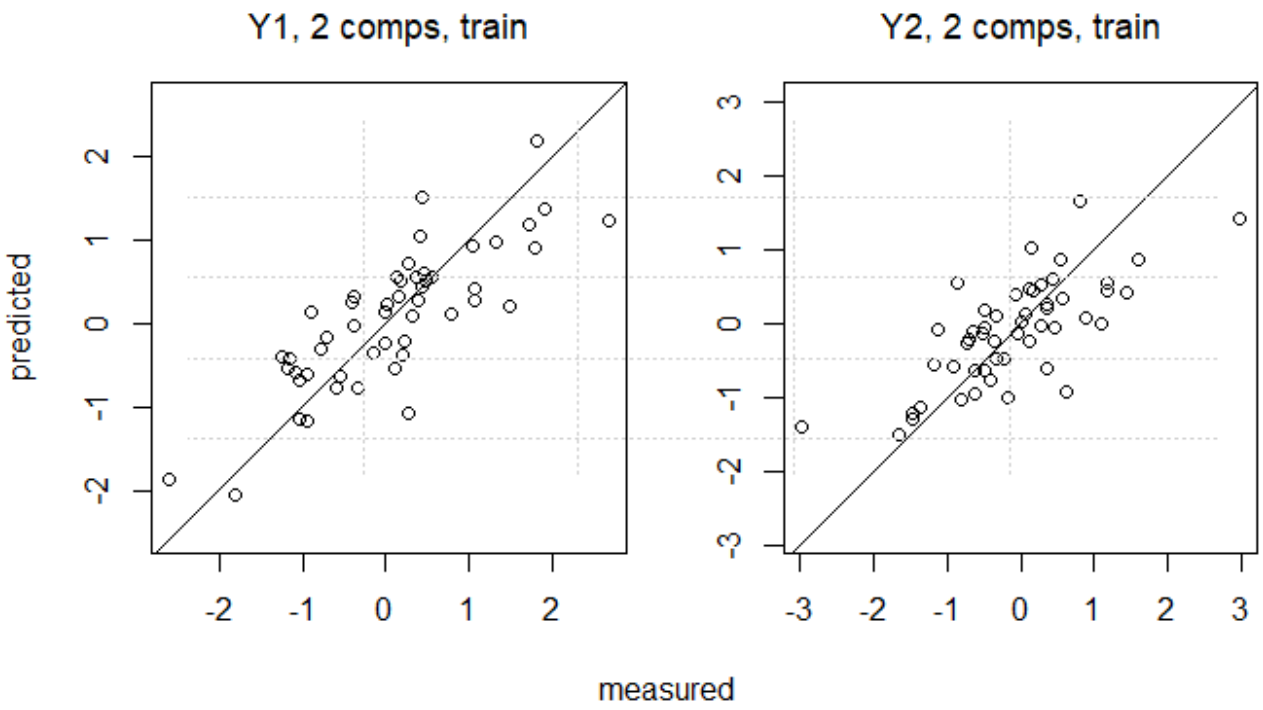
第１主成分だけで予測した場合

```
plot(fit02,ncomp=1,line=TRUE,asp=1)
grid()
```



第２主成分も使って予測した場合

```
plot(fit02,ncomp=2,line=TRUE,asp=1)
grid()
```



主成分のスコア

Xの主成分のスコアは（最初の6個だけ）

```
px2←scores(fit02)
px2%>%
  round(4)%>%
  head()
```

```
##    Comp 1  Comp 2
## 1 -0.1106 -0.0836
## 2 -1.3586 -0.1048
## 3  1.1375 -0.0624
## 4 -0.2181 -1.3323
## 5 -0.0030  1.1652
## 6  1.1188 -0.6712
```

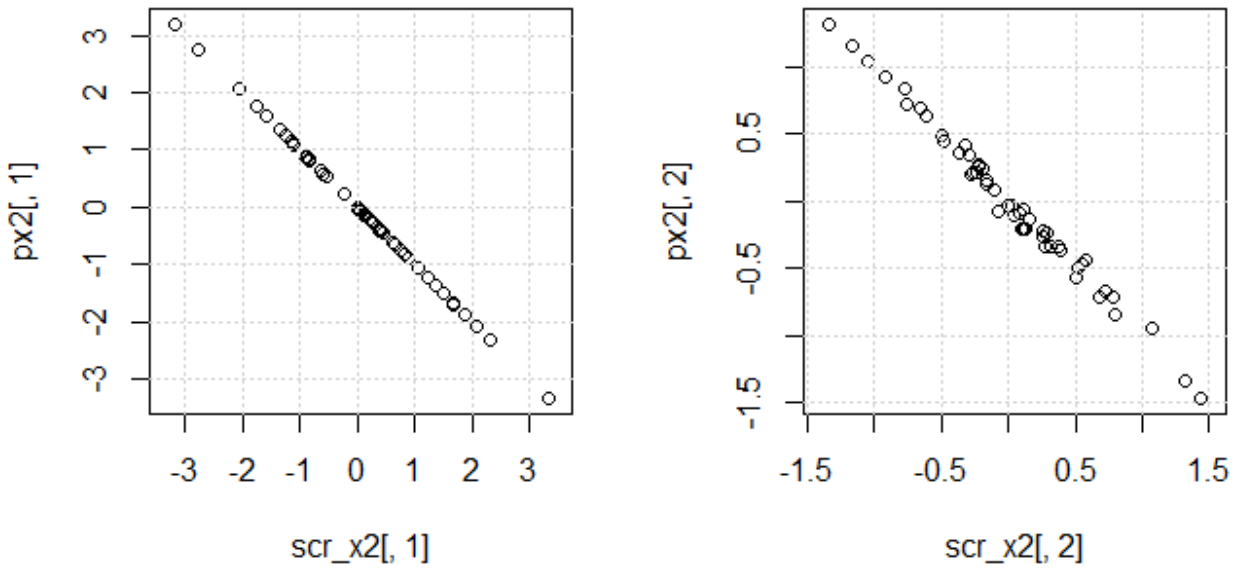
Yの主成分のスコアは（最初の6個だけ）

```
py2←Yscores(fit02)
py2%>%
  round(4)%>%
  head()
```

```
##    Comp 1  Comp 2
## 1  0.0619 -0.0882
## 2 -1.5840 -0.1580
## 3  1.7078 -0.0184
## 4 -0.6398 -2.7412
## 5  0.7637  2.5647
## 6  0.4764 -0.3743
```

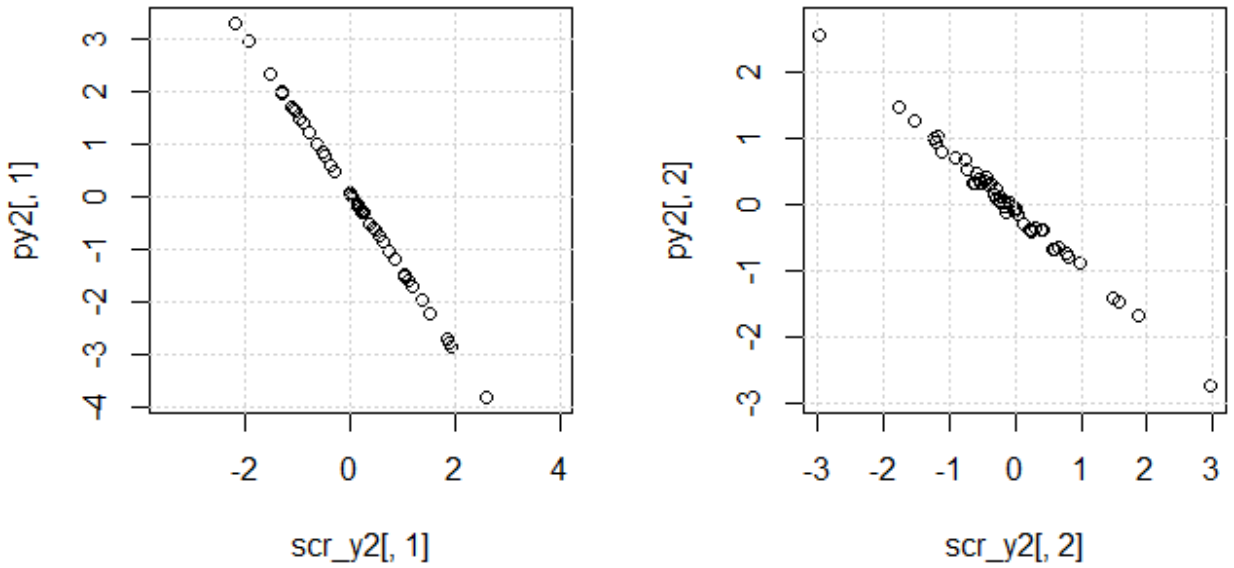
XのスコアはXu^*から出した`scr_x2`に（符号を除いて）ほぼ一致する

```
par(mfrow=c(1,2))
plot(px2[,1]~scr_x2[,1],asp=1)
grid()
plot(px2[,2]~scr_x2[,2],asp=1)
grid()
```



しかし，YのスコアはYv^*とスケールが一致しない！

```
par(mfrow=c(1,2))
plot(py2[,1]~scr_y2[,1],asp=1)
grid()
plot(py2[,2]~scr_y2[,2],asp=1)
grid()
```



やはりYのスコアの分散の調整が行われている．

```
lm(Yscores(fit02)~scores(fit02)-1)
```

```
##
## Call:
## lm(formula = Yscores(fit02) ~ scores(fit02) - 1)
##
## Coefficients:
##                       Comp 1      Comp 2
## scores(fit02)Comp 1   1.000e+00  -8.270e-17
## scores(fit02)Comp 2   2.075e-01   1.000e+00
```

Yのスコアの分散は

```
py2_v<-var(py2)
py2_v%>%
  round(4)
```

```
##        Comp 1 Comp 2
## Comp 1 2.4023 0.1101
## Comp 2 0.1101 0.7093
```

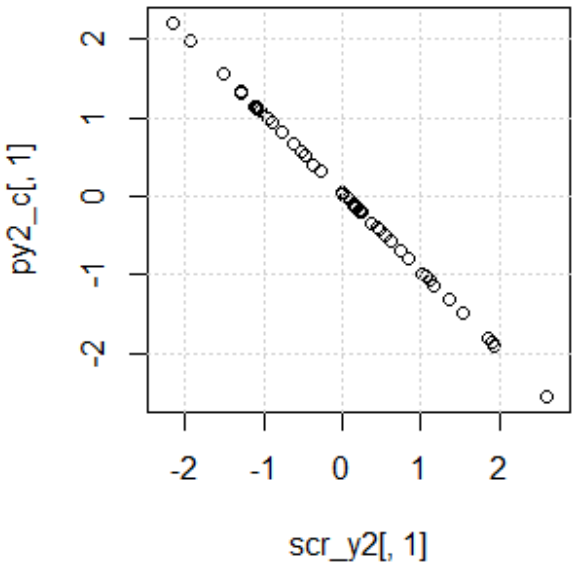
X‘Yの最大化から求めたYのスコアの分散は

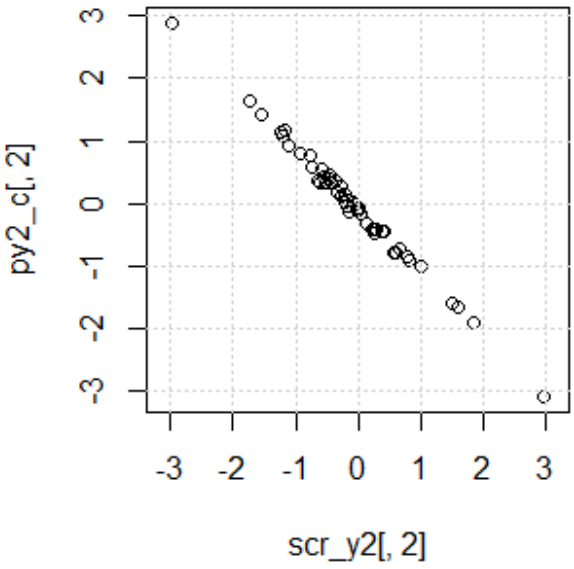
```
xy2_v<-var(scr_y2)
xy2_v%>%
  round(4)
```

```
##        [,1]   [,2]
## [1,] 1.0801 0.0414
## [2,] 0.0414 0.9009
```

`plsr()`の2つのスコアを，それぞれ割ってやると，X‘Yの最大化から求めたYのスコアと一致する．

```
py2_c<-sweep(py2,2,
      c(sqrt(py2_v[1,1]/xy2_v[1,1]),sqrt(py2_v[2,2]/xy2_v[2,2])),
      FUN="/")%>%
  round(4)
par(mfrow=c(1,2))
plot(py2_c[,1]~scr_y2[,1],asp=1)
grid()
plot(py2_c[,2]~scr_y2[,2],asp=1)
grid()
```

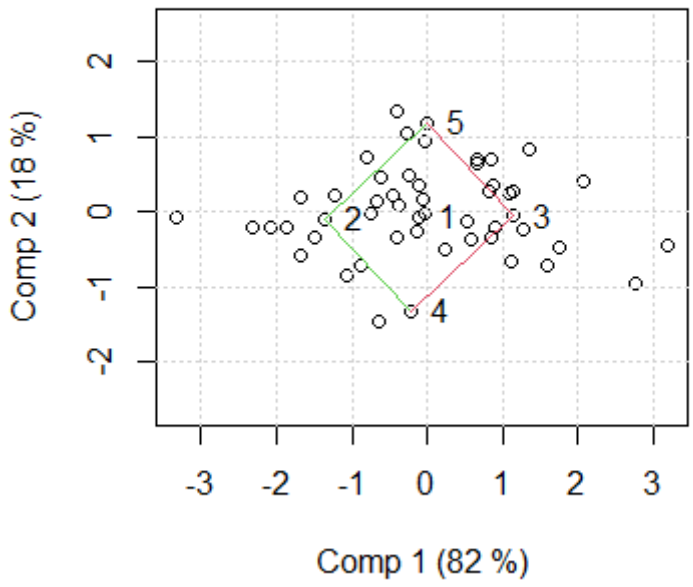




Xのスコアのプロット

`scoreplot()` でもOK

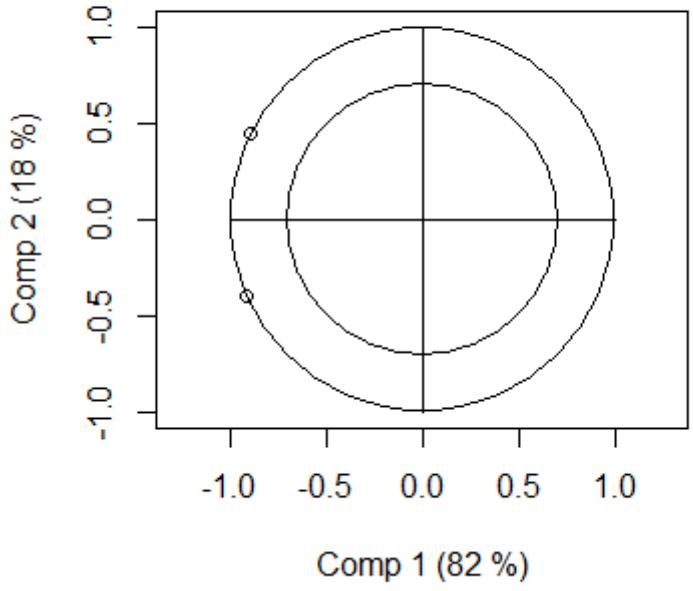
```
plot(fit02,plottype="scores",comps=1:2,asp=1)
grid()
text(x=px2[1:5,1],y=px2[1:5,2],1:5,pos=4)
f_grid1(px2)
```



スコアとXとの相関のプロット

`plot()` 関数の `plottype="correlation"` 引数でも同じ．

```
corrplot(fit02)
```




```
cor(x2,px2)
```

```
##          Comp 1     Comp 2
## [1,] -0.9180898 -0.3963724
## [2,] -0.8958415  0.4443737
```

・・・というのをプロットしているようだ．左下がx_1で，左上がx_2．

x_1もx_2も，第１主成分の方に相関が高そう・・・ということはわかる．

uの出力は `loadings()` で．

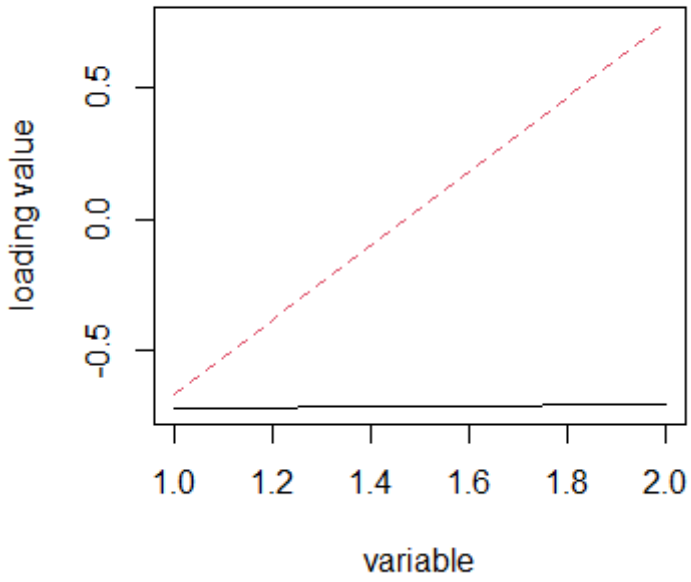
```
loadings(fit02)
```

```
##
## Loadings:
##    Comp 1 Comp 2
## x1 -0.716 -0.666
## x2 -0.699  0.746
##
##                Comp 1 Comp 2
## SS loadings     1.002  1.000
## Proportion Var  0.501  0.500
## Cumulative Var  0.501  1.001
```

`Comp 1` は，`-0.716*x1+0.699*x2` だよと言っている．
`Comp 2` は，`-0.666*x1+0.746*x2` だよと言っている．

２変数だけなので，図示する意味はあまりないが，`loadingplot()` でいける

```
loadingplot(fit02)
```



x_1, x_2の1，2を数字として扱っているため，途中に`1.2`とか余計な目盛が入っているが，これは無視する．

vは，`Yloadings()` で得られるはずだが

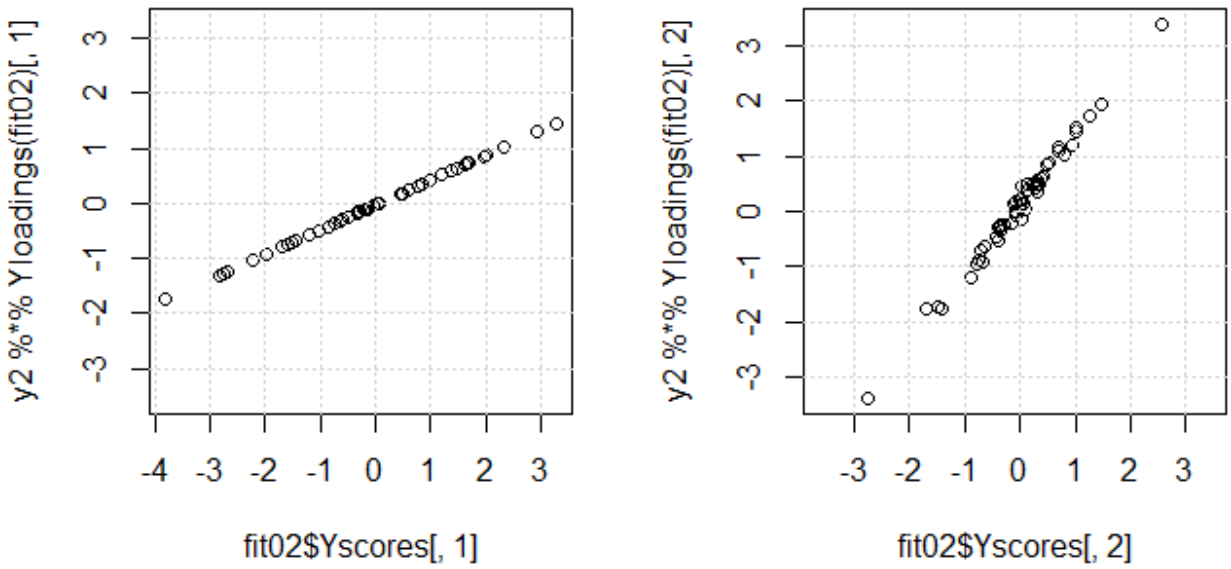
```
Yloadings(fit02)
```

```
##
## Loadings:
##    Comp 1 Comp 2
## Y1 -0.644  0.180
## Y2 -0.188 -1.114
##
##                Comp 1 Comp 2
## SS loadings     0.450  1.272
## Proportion Var  0.225  0.636
## Cumulative Var  0.225  0.861
```

・・・と，loadingが1を超えていたりとよくわからない？？？

Yとの積をとれば，ちゃんとスコアと比例はするが，スケールが全然違う．

```
par(mfrow=c(1,2))
plot(y2%*%Yloadings(fit02)[,1]~fit02$Yscores[,1],asp=1)
grid()
plot(y2%*%Yloadings(fit02)[,2]~fit02$Yscores[,2],asp=1)
grid()
```



$X'Y$によるYのスコアYv^*の分散は

```
svd_xy2←t(x2)%*%y2%>%
  svd()
v_scr_xy2←y2%*%svd_xy2$v%>%
  var()
v_scr_xy2
```

```
##           [,1]      [,2]
## [1,] 1.08010171 0.04140877
## [2,] 0.04140877 0.90089439
```

`plsr()` で出したスコアの分散は，Yが調整されている分だけ変わっている．

```
v_scr_pls2←fit02$Yscores%>%
  var()
v_scr_pls2
```

```
##           Comp 1    Comp 2
## Comp 1 2.4023141 0.1100564
## Comp 2 0.1100564 0.7093003
```

PLSのLoadingsに，第１主成分の分散の比の平方根を掛けてやると，v^*の１列目に一致する．

```
sqrt(v_scr_pls2[1,1]/v_scr_xy2[1,1])*Yloadings(fit02)[1,1]
```

```
## [1] -0.9599648
```

```
sqrt(v_scr_pls2[1,1]/v_scr_xy2[1,1])*Yloadings(fit02)[2,1]
```

```
## [1] -0.2801207
```

しかし，第２主成分の分散の比の平方根を掛けても，v^*の２列目と近くはなるが，全くは一致しない．

```
sqrt(v_scr_pls2[2,2]/v_scr_xy2[2,2])*Yloadings(fit02)[1,2]
```

```
## [1] 0.159741
```

```
sqrt(v_scr_pls2[2,2]/v_scr_xy2[2,2])*Yloadings(fit02)[2,2]
```

```
## [1] -0.9880762
```

```
v2
```

```
##           [,1]       [,2]
## [1,] 0.9599648 -0.2801207
## [2,] 0.2801207  0.9599648
```

無相関の説明変数を加える

以上，２×２変数の場合で説明したが，PLSの強みは，数多くの説明変数の中から相関する主成分を抽出してくること．

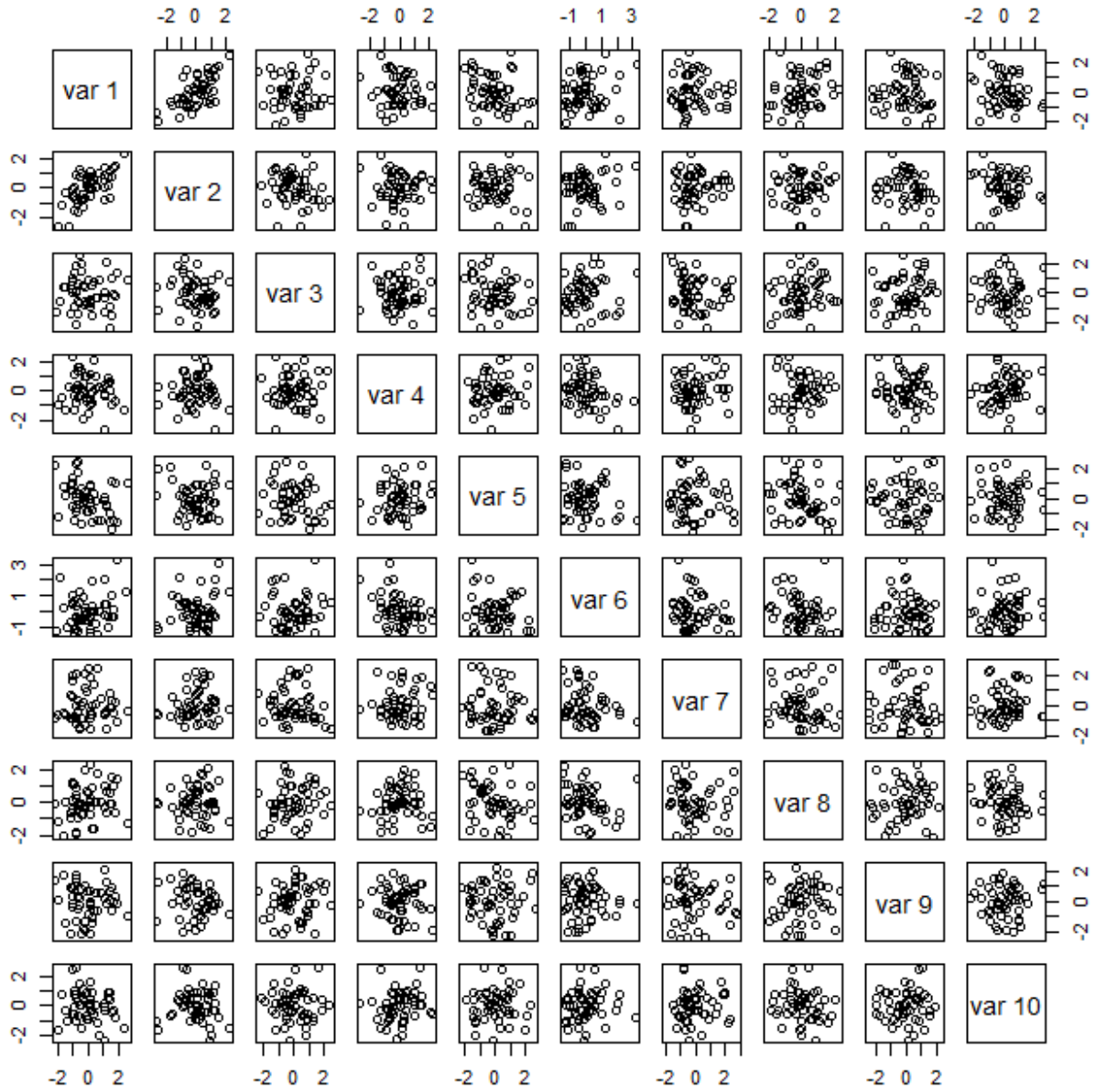
8つのランダムな変数をXに加えてみる．

データ作成

```
set.seed(123)
x3_c<-rnorm(51*8)%>%
  matrix(ncol=8)%>%
  scale()
x3<-cbind(x2,x3_c)
```

相関を確認してみる

```
pairs(x3,asp=1)
```



x_1, x_2だけが相関していることを確認.

Yはそのまま使う

```
y3<-y2
```

`plsr()`関数用のデータセット

```
d03<-data.frame(x=I(x3),y=I(y3))
```

`plsr()`関数で

```
fit03<-plsr(y~x,
            ncomp=10,
            data=d03,
            validation="CV")
```

要約

```r
summary(fit03)
```

```
## Data:    X dimension: 51 10
##  Y dimension: 51 2
## Fit method: kernelpls
## Number of components considered: 10
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##
## Response: Y1
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.032   0.7149   0.6844   0.6857   0.7112   0.6963   0.7012
## adjCV        1.032   0.7059   0.6780   0.6798   0.6996   0.6886   0.6935
##        7 comps  8 comps  9 comps  10 comps
## CV      0.7109   0.7188   0.7215    0.7220
## adjCV   0.7029   0.7100   0.7125    0.7129
##
## Response: Y2
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.9777   0.9649   0.8875   0.8269   0.8100   0.7647   0.7612
## adjCV       0.9777   0.9642   0.8801   0.8096   0.8024   0.7559   0.7520
##        7 comps  8 comps  9 comps  10 comps
## CV      0.7596   0.7603   0.7599    0.7594
## adjCV   0.7506   0.7513   0.7509    0.7506
##
## TRAINING: % variance explained
##     1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    19.526    34.28    40.26    50.45    60.00    70.42    76.90    85.01
## Y1   60.092    66.74    67.06    69.49    70.04    70.06    70.07    70.09
## Y2    9.682    38.86    58.05    59.00    61.19    61.63    61.69    61.69
##     9 comps  10 comps
## X     94.23    100.00
## Y1    70.09     70.09
## Y2    61.69     61.69
```

主成分数をいくつにするか決める

その前にRMSEPって？

Yの実際値と推定値との誤差の平方平均の平方根がRMSEP.
しかし，計算してみると一致しない.

第１主成分までのRMSEPは

```r
RMSEP(fit03,ncomp=1)
```

```
##
## Response: Y1
##        (Intercept)  1 comps
## CV           1.032   0.7149
## adjCV        1.032   0.7059
##
## Response: Y2
##        (Intercept)  1 comps
## CV          0.9777   0.9649
## adjCV       0.9777   0.9642
```

ところが，残差の平方平均の平方根は

```
fit03$residuals[,,1]^2%>%
  apply(.,2,mean)%>%
  sqrt()
```

```
##        Y1        Y2
## 0.6390451 0.9109793
```

なんかちょっと違う．
実際には，クロスバリデーションで得られたRMSEPを計算している．

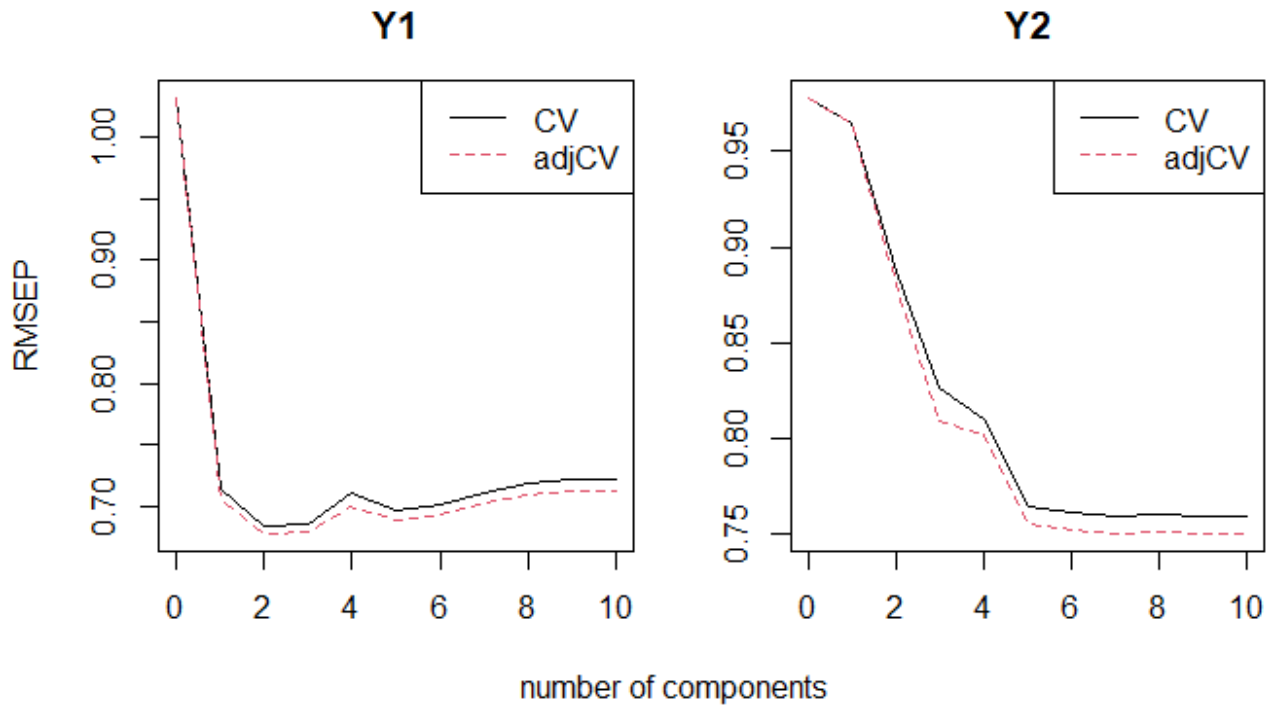
すべてのサンプルを使って推定すると（引数 `esimate="all"` で， `train` の `1 comp` のところを見ると・・・），単純に計算したRMSEPと一致する．

```
RMSEP(fit03,ncomp=1,estimate="all")
```

```
##
## Response: Y1
##        (Intercept)  1 comps
## train        1.012   0.6390
## CV           1.032   0.7149
## adjCV        1.032   0.7059
##
## Response: Y2
##        (Intercept)  1 comps
## train       0.9586   0.9110
## CV          0.9777   0.9649
## adjCV       0.9777   0.9642
```

RMSEPで

```
plot(RMSEP(fit03), legendpos = "topright")
```



成分数を選ぶヒューリスティックスとして，`selectNcomp()`関数が用意されている．
使てみたいが，これはYが1変数のときしか使えないようだ．

どんなものか知るだけのために，y_1だけにPLS回帰してみよう．

```
fit03_1←plsr(y[,1]~x,ncomp=10,data=d03,validation="CV")
```

`selectNcomp()`関数を使ってみる．判断基準は2通りあり，引数`method="onsigma"`と引数`method="randomization"`で使い分ける．

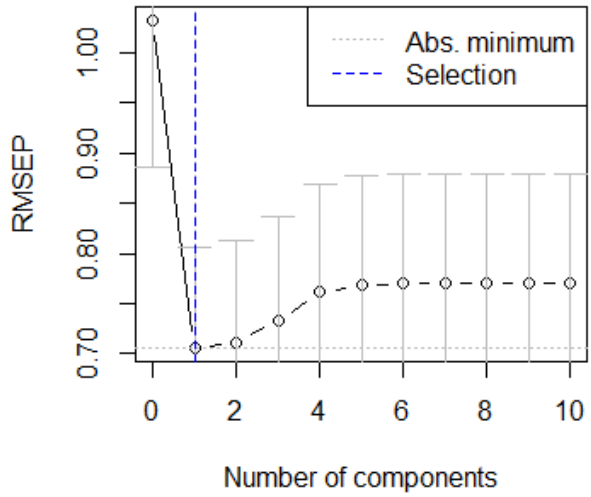
`"onesigma"`は，まず，RMSEPが最低の点を求め，そこから1標準誤差以内で最も少ない主成分数を採用するという方法．

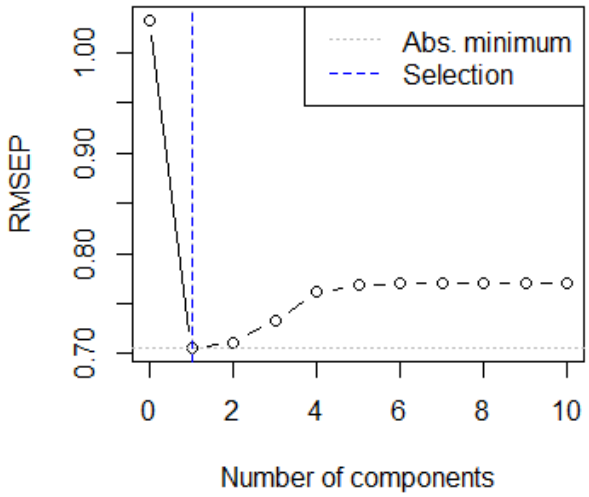
`"randomization"`は，やはりRMSEPが最低の点を求め，そこから有意差がない主成分数を採用するという方法．

```
par(mfrow=c(1,2))
selectNcomp(fit03_1, method = "onesigma",plot=TRUE)
```

```
## [1] 1
```

```
selectNcomp(fit03_1, method = "randomization",plot=TRUE)
```






```
## [1] 1
```

両方とも1主成分が最適と出してきた.

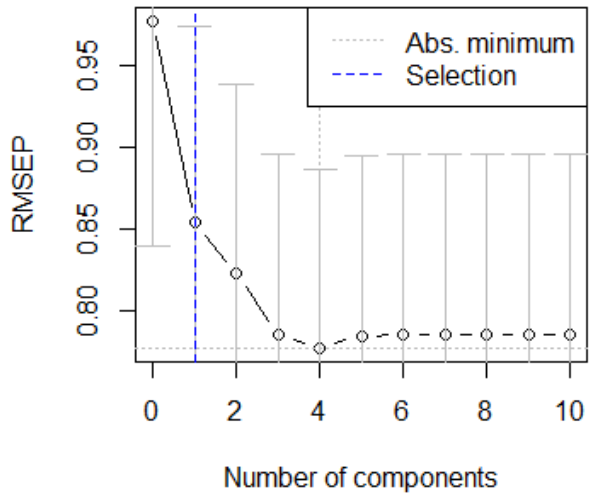
y_2についてもやってみよう.

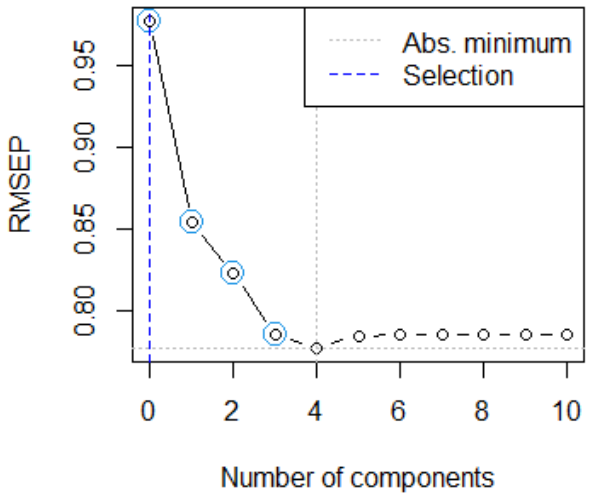
```
fit03_2←plsr(y[,2]~x,ncomp=10,data=d03,validation="CV")
```

```
par(mfrow=c(1,2))
selectNcomp(fit03_2, method = "onesigma",plot=TRUE)
```

```
## [1] 1
```

```
selectNcomp(fit03_2, method = "randomization",plot=TRUE)
```





```
## [1] 0
```

1か0なのだそうだ.

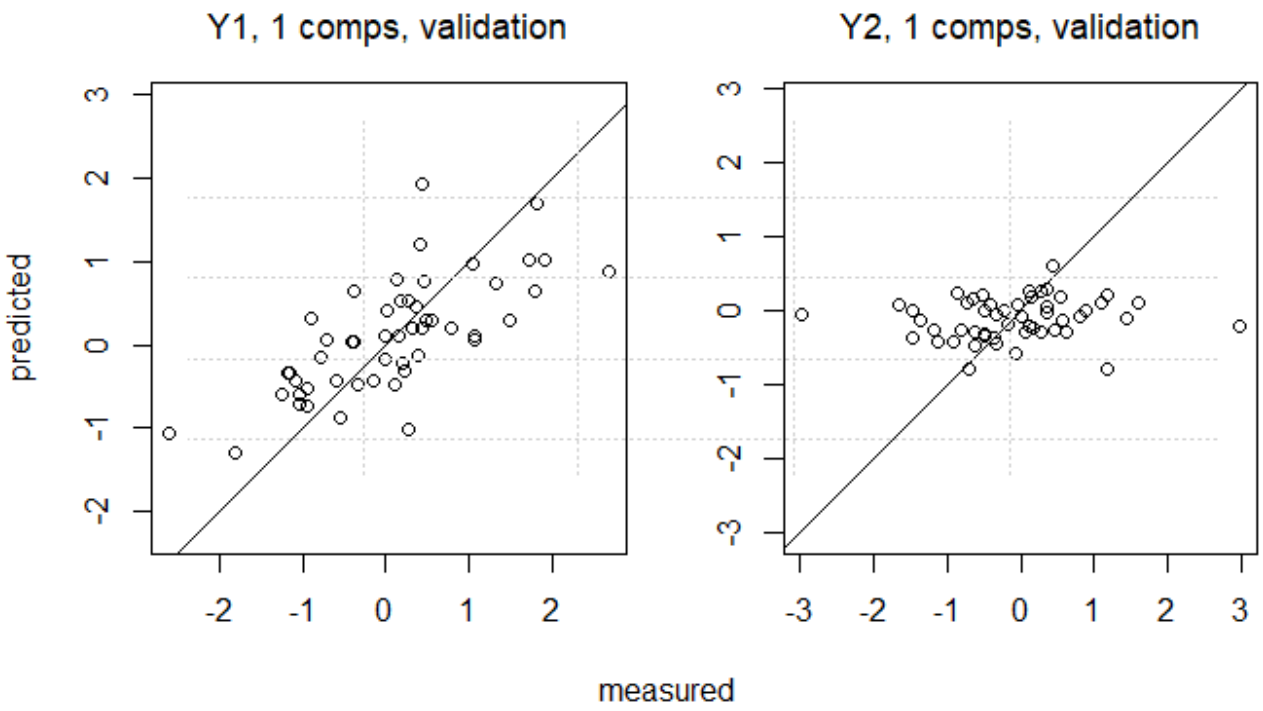
しかし，これはあくまで，Yを直接対応させた場合の結果.

予測値で

もとのYと予測値とをプロットして確認

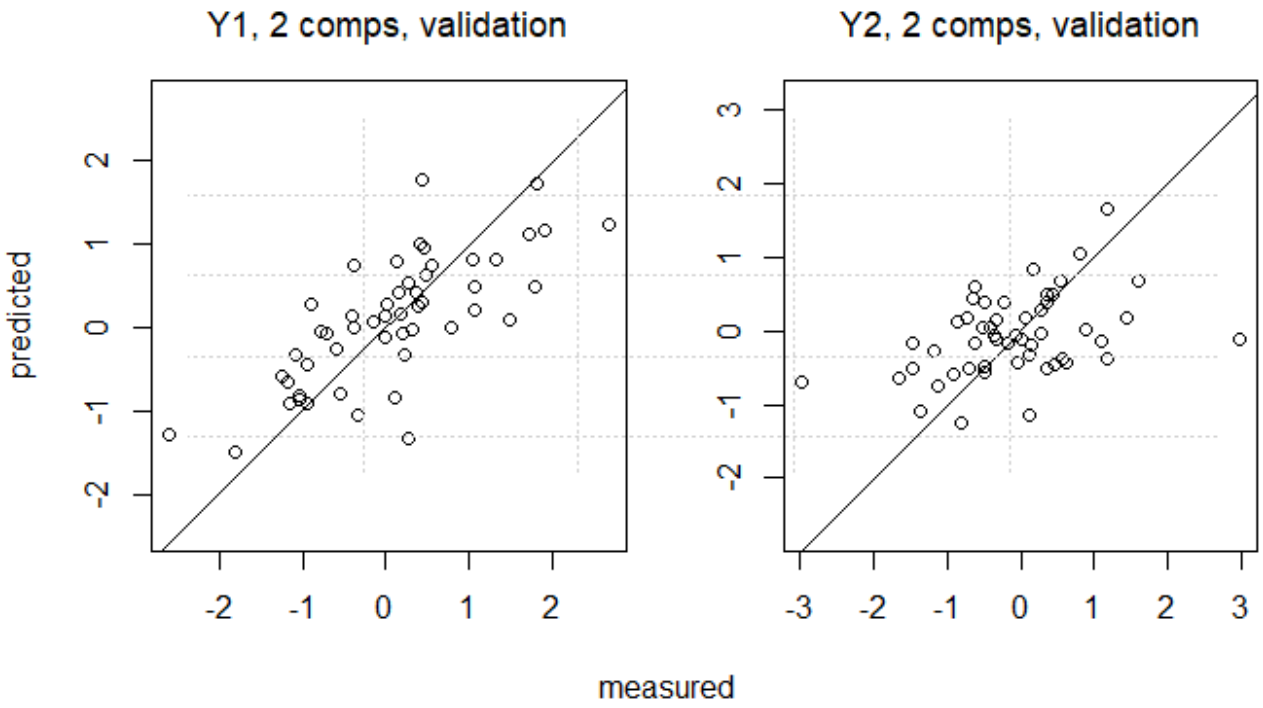
１主成分だけだとy_1はいい感じでfitしているが，y_2はいまいち.

```
plot(fit03, ncomp = 1, asp = 1, line = TRUE)
grid()
```



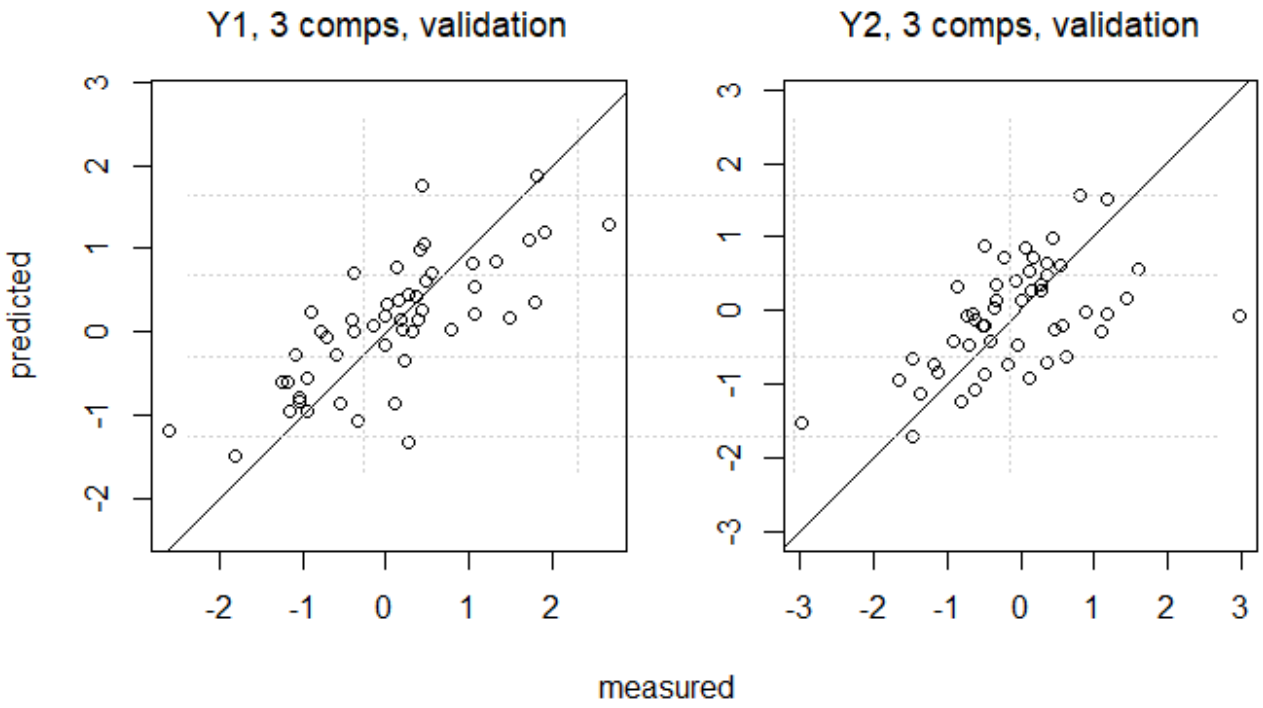
第２主成分まで使うとy_1のfitはあんまりかわらんが，y_2はだいぶよくなった.

```
plot(fit03, ncomp = 2, asp = 1, line = TRUE)
grid()
```



第３主成分まで使うと，両者ともあんまり変わらない．

```
plot(fit03, ncomp = 3, asp = 1, line = TRUE)
grid()
```

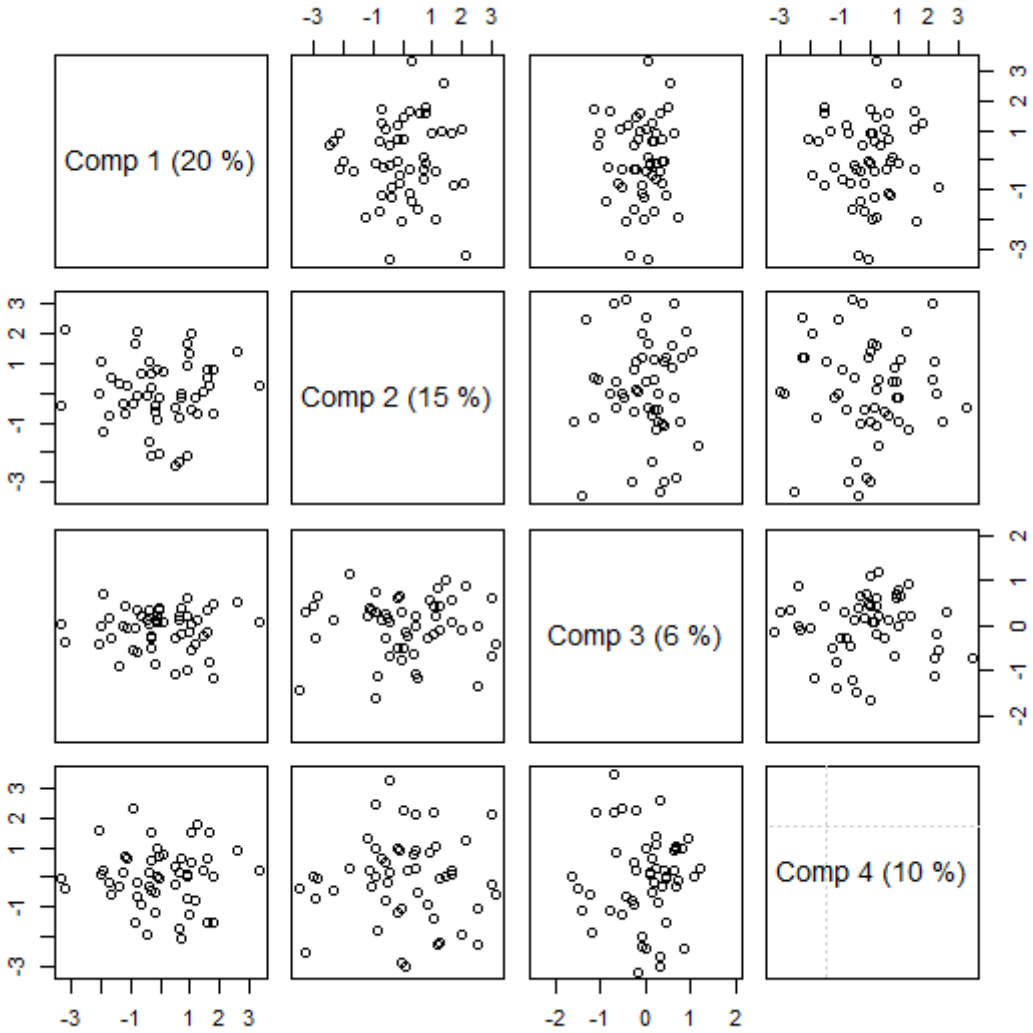


2主成分ということか？

Xのスコアのプロット

Xの第4主成分までスコアのプロットしてみると．．．

```
scoreplot(fit03,asp=1,comps = 1:4)
grid()
```



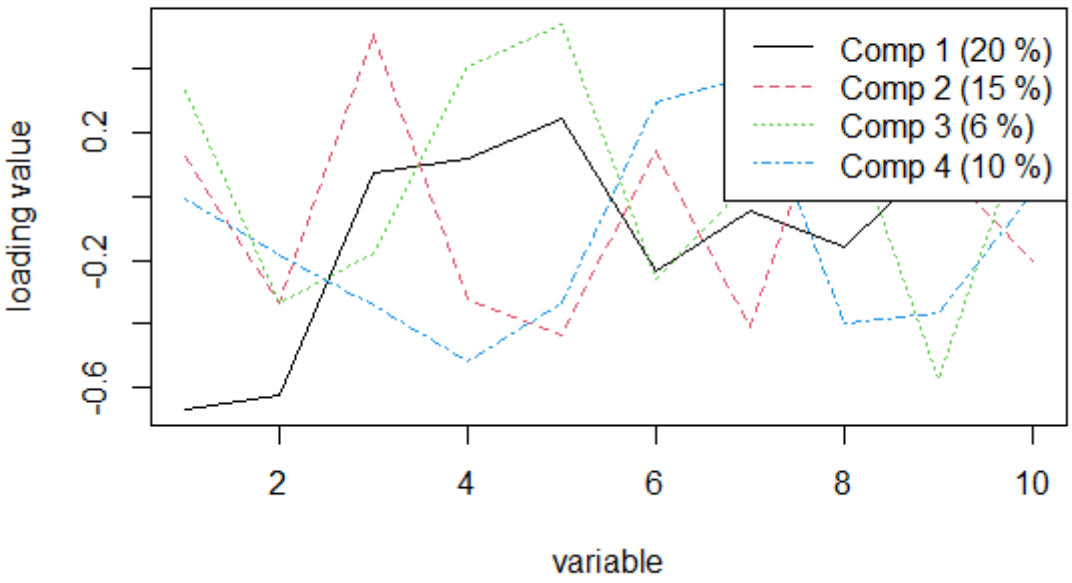
う〜ん，この情報はどうなんだろう．

説明力の高いXの変数を選択する

第１主成分だけでいくとして，Xのうち，どの変数を説明変数として採用するか・・・

u（loading value）を確認する

```
loadingplot(fit03,legendpos = "topright",comps=1:4)
```



`Comp 1` の `loading value` を見ると，変数１と2だけが大きくぶれている．

`Comp 2` の `loading value` は，特に大きな値はないなあ．

これを見る限り，x_1とx_2だけで１つの主成分を抽出し，y_1, y_2を説明するのが適切か？という仮説が出される・・・たぶん．

ダメ押しで，スコアとXとの相関も．

次は，第１主成分と第2主成分のスコアとXの各変数との相関

loading
valueのノルムで，左端のx_1, x_2は1に近いが，他は0.5あたりか，それ未満．あんまり相関していないということ．

補足:シミュレーション

係数の不偏性を確かめる．

51×2のX, Yについて，OLS($Y \leftarrow X$)とPCR（$Y \leftarrow X$のスコア），それに$X'Y$の特異値分解から求めたスコア間のOLS回帰の結果を比較する．

３通りの係数を計算する関数

とりあえず１回の推定をする関数を定義する．

攪乱項はXとYの両方に与え，その標準偏差は両者で同じとする．
各推定4つずつ，3種類で，12個の推定係数が出力．

```
f_pls01<-function(x1,y1,sd1=0.1){
  x2<-x1+matrix(rnorm(nrow(x1)*2,mean=0,sd=sd1),ncol=2)
```

```
  y2←y1+matrix(rnorm(nrow(y1)*2,mean=0,sd=sd1),ncol=2)
  fit_OLS←lm(y2~x2-1)%>%
    coef()
  prc01←prcomp(x1)
  Px←prc01$x
  fit_PCR←lm(y2~Px-1)%>%
    coef()
  svd_xy←svd(t(x2)%*%y2)
  Scrx←x2%*%svd_xy$u
  Scry←y2%*%svd_xy$v
  fit_PLS←lm(Scry~Scrx-1)%>%
    coef()
  coef01←c(fit_OLS,fit_PCR,fit_PLS)
  coef01
}
```

シミュレーション

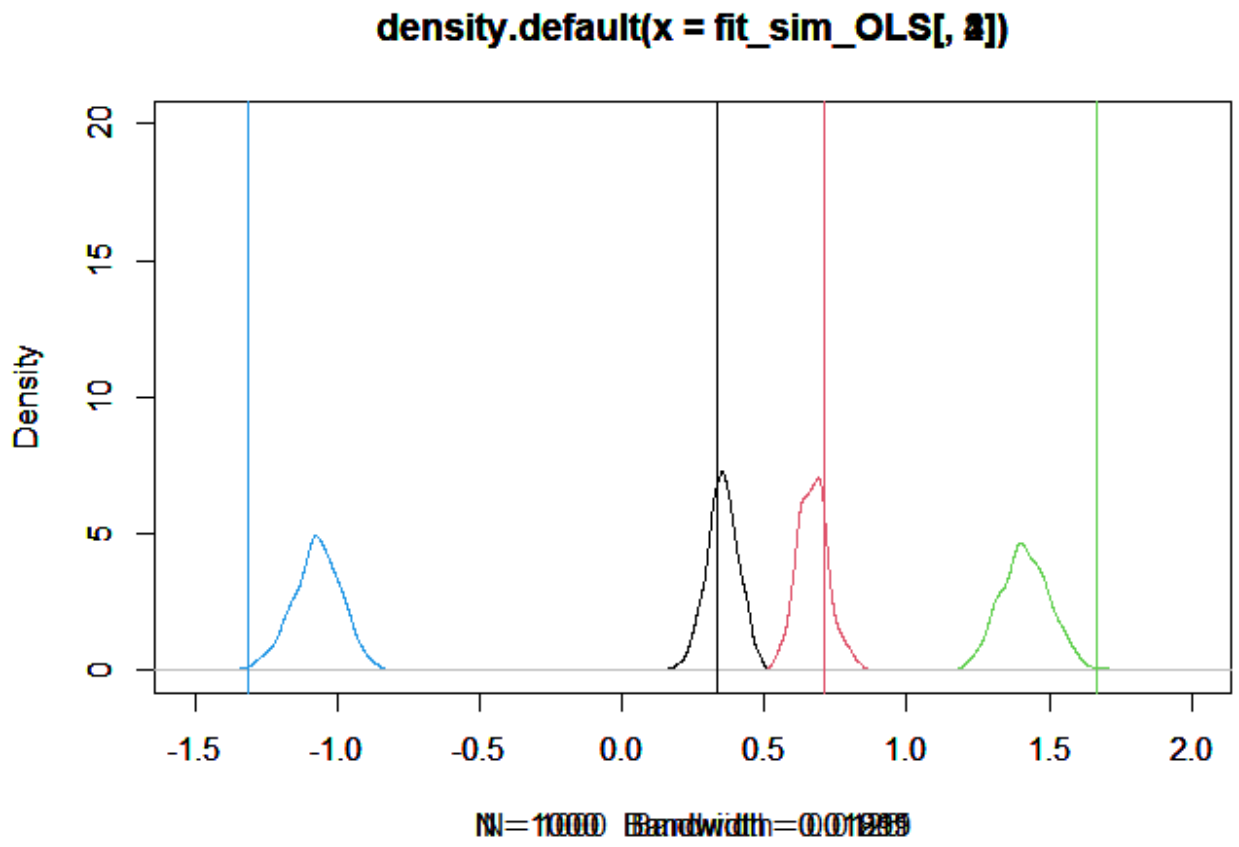
撹乱項の標準偏差を0.2として，1000回推定

```
fit_sim01←matrix(nrow=1000,ncol=12)
for(i in 1:1000){
  fit_sim01[i,]←f_pls01(x1,y1,sd1=0.2)
}

fit_sim_OLS←fit_sim01[,1:4]
fit_sim_PCR←fit_sim01[,5:8]
fit_sim_PLS←fit_sim01[,9:12]
```

OLS

Y←XのOLS推定

```
fit_OLS0←a1%*%b0%*%solve(a1)
plot(density(fit_sim_OLS[,1]),xlim=c(-1.5,2),ylim=c(0,20),col=1)
par(new=T)
plot(density(fit_sim_OLS[,2]),xlim=c(-1.5,2),ylim=c(0,20),col=2)
par(new=T)
plot(density(fit_sim_OLS[,3]),xlim=c(-1.5,2),ylim=c(0,20),col=3)
par(new=T)
plot(density(fit_sim_OLS[,4]),xlim=c(-1.5,2),ylim=c(0,20),col=4)
abline(v=fit_OLS0[1,1],col=1)
abline(v=fit_OLS0[2,1],col=2)
abline(v=fit_OLS0[1,2],col=3)
abline(v=fit_OLS0[2,2],col=4)
```

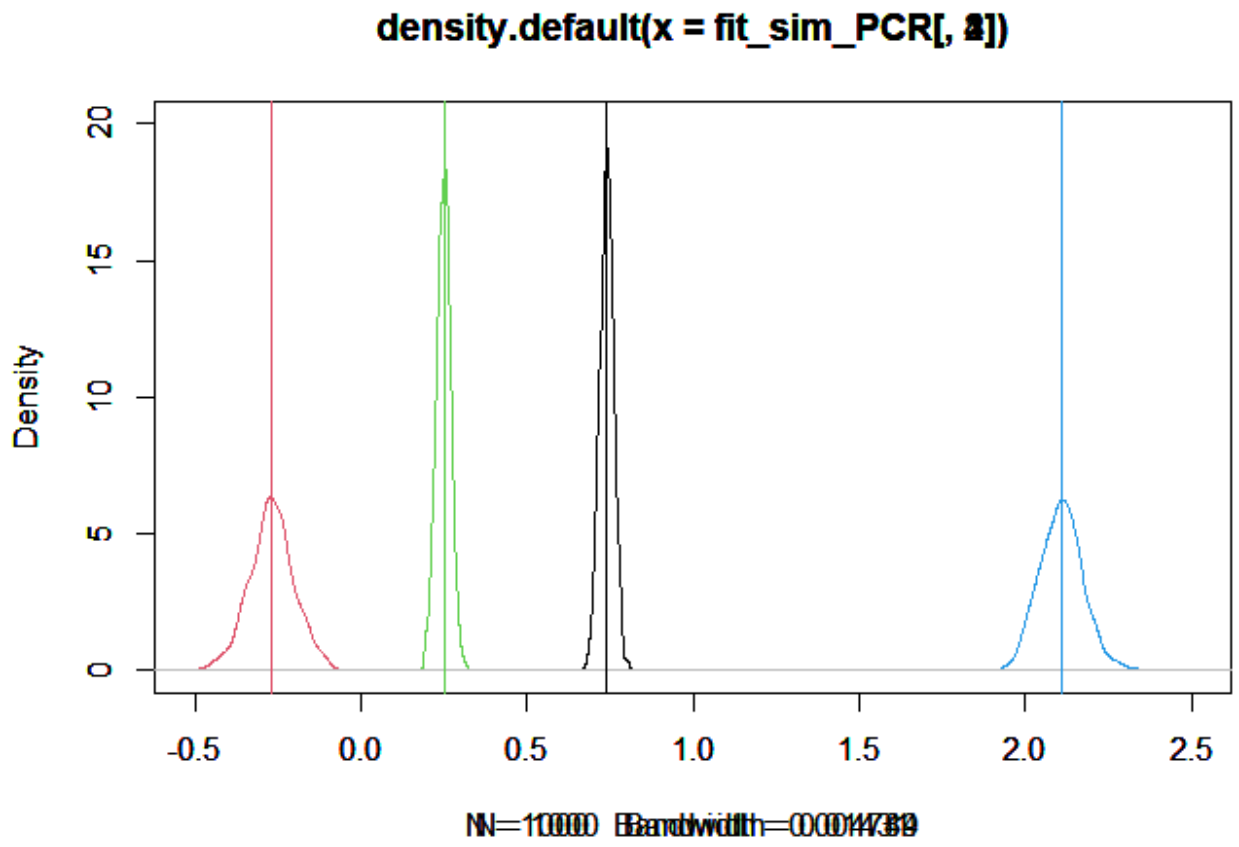


垂直線が真値．　不偏性がない！ 推定値の分散も大きい．

PCR

$Y \leftarrow X$の主成分分析のスコア

```
fit_PCR0←b0%*%solve(a1)
plot(density(fit_sim_PCR[,1]),xlim=c(-0.5,2.5),ylim=c(0,20),col=1)
par(new=T)
plot(density(fit_sim_PCR[,2]),xlim=c(-0.5,2.5),ylim=c(0,20),col=2)
par(new=T)
plot(density(fit_sim_PCR[,3]),xlim=c(-0.5,2.5),ylim=c(0,20),col=3)
par(new=T)
plot(density(fit_sim_PCR[,4]),xlim=c(-0.5,2.5),ylim=c(0,20),col=4)
abline(v=fit_PCR0[1,1],col=1)
abline(v=fit_PCR0[2,1],col=2)
abline(v=fit_PCR0[1,2],col=3)
abline(v=fit_PCR0[2,2],col=4)
```



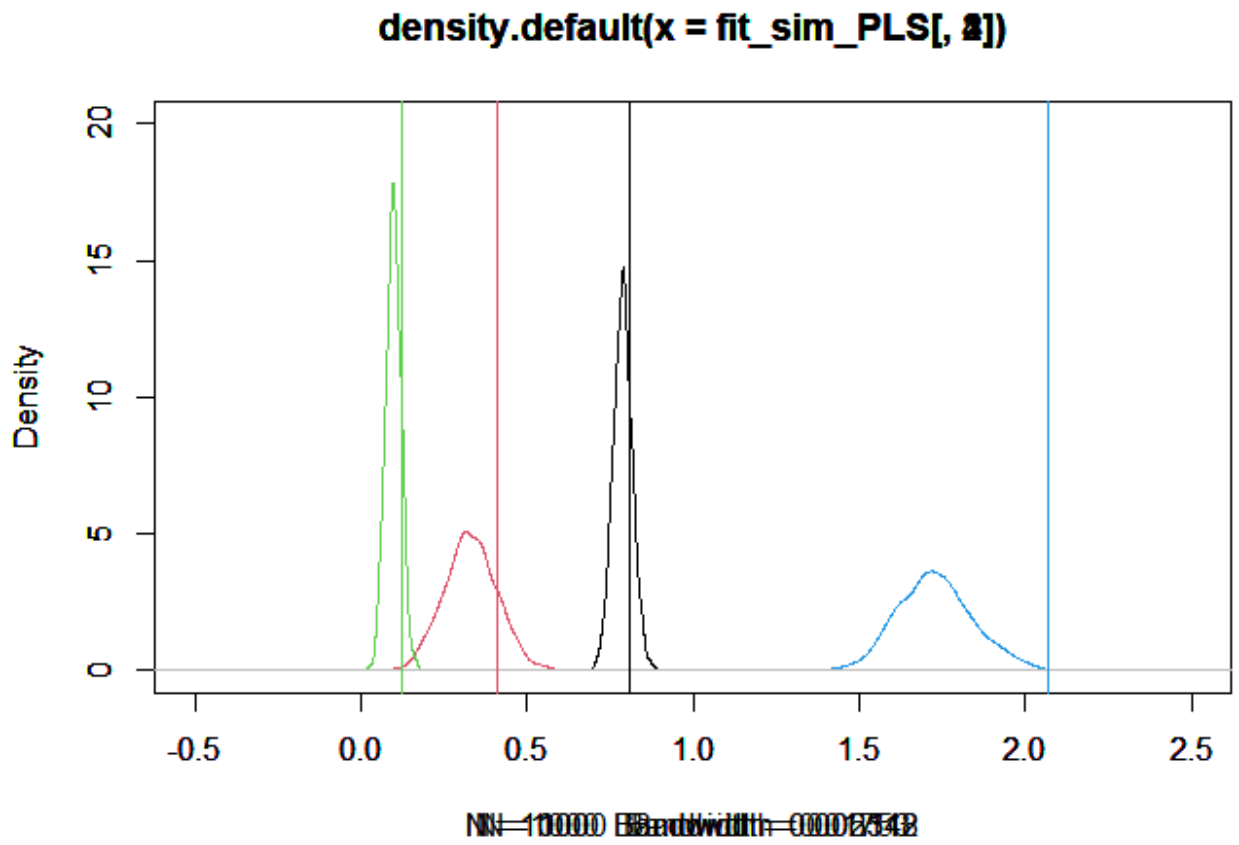
不偏性がありそうだ.

Xの第１主成分の係数（黒と緑）は標準誤差が小さいが，第２主成分の標準が長が大きくなっている.

PLS

$X'Y$の特異値分解から，スコア間の関係$Yv \leftarrow Xu$

```
fit_PLS0←xyb1
plot(density(fit_sim_PLS[,1]),xlim=c(-0.5,2.5),ylim=c(0,20),col=1)
par(new=T)
plot(density(fit_sim_PLS[,2]),xlim=c(-0.5,2.5),ylim=c(0,20),col=2)
par(new=T)
plot(density(fit_sim_PLS[,3]),xlim=c(-0.5,2.5),ylim=c(0,20),col=3)
par(new=T)
plot(density(fit_sim_PLS[,4]),xlim=c(-0.5,2.5),ylim=c(0,20),col=4)
abline(v=fit_PLS0[1,1],col=1)
abline(v=fit_PLS0[2,1],col=2)
abline(v=fit_PLS0[1,2],col=3)
abline(v=fit_PLS0[2,2],col=4)
```



Xuの第１主成分（黒と緑）については，だいぶいいところまで来ているが，不偏性は怪しい．

第２主成分（赤と青）はだいぶ怪しい？？？

実際のPLSでは，このように一気にXuとYvを算出するのではなく，まず，第１主成分Xu_1とYv_1を算出し，次いで，それを前提にXu_2とYv_2を算出・・・というような手順を踏むようだ．（そっから先は，一般的なPLSの解説がたくさんある）

そもそも，PLSの関心が，係数の値そのものにあるのではなく，説明力の高い主成分と，その主成分と相関の高いXの特定にある．`plsr()`関数などは，Xに合わせてYのスケールそのものを変えてしまう．

複数のYを主成分として扱えるところに魅力を感じたが，そこはあんまり意味なさそう．

たくさんのXの中から，PLSで主成分とそれに相関の高いXを抽出し，PCRで回帰するのが妥当な使い方のような気がする．

結論

1. PLSは$(Xu)'Yv$を最大化するようにu^*とv^*を決める．
2. このu^*とv^*は，$X'Y$の特異値分解から得られる．
3. 推定の考え方がOLSとは違うので，XとYにかく乱項が加わった場合，係数の不偏性は怪しい．
4. PLSの実際の利用は，多数の変数を持つXからYの説明力の高いXの主成分を抽出し，それと相関の高いXを特定すること．
5. 変数間の係数に関心がある場合は，PLSで特定したXでPCRをするのがよさそう．
6. 実際のPLSは，$X'Y$の特異値分解という単純なものでなく，主成分を１つずつ算出していく複雑なアルゴリズムが複数ある．

7. Rのパッケージplsにある `plsr()` 関数で出力されるYのスコア（ `Yscores` ）はXのスコアに合わせてスケールを調整しているので要注意.