

Least Squares with Matrix 行列で最小二乗法

ryamada

2016年12月23日

- 1 最小二乗法を使うとき When to use least squares
- 2 $b = 0$ の場合と、そうでない場合
- 3 どうやるか
 - 3.1 式を眺める
- 4 Exercise 1
 - 4.1 Exercise 1-1
- 5 固有値分解とは違う分解、QR分解、特異値分解
- 6 Exercise 2
 - 6.1 Exercise 2-1

1 最小二乗法を使うとき When to use least squares

1 個の説明変数、1 個の被説明変数があつて、線形回帰を行うとき。

$$Y \sim aX + b$$

$$\sum_{i=1}^n (y_i - \hat{y})^2 = \sum_{i=1}^n (y_i - (ax_i + b))^2$$

を最小にするような、 a, b の値を求める。

複数の説明変数があつて、1 個の被説明変数があるときには、説明変数の数だけの係数からなるベクトル \mathbf{a} を用いて

$$y \sim X\mathbf{a} + b$$

というモデルになる。

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (x_i^T \mathbf{a} + b))^2 = \|y - (X\mathbf{a} + b)\|^2$$

を最小にするような \mathbf{a}, b を求める。

2 $b = 0$ の場合と、そうでない場合

今、 $b = 0$ と固定したモデルの場合には

$$y \sim X\mathbf{a}$$

$$\|y - X\mathbf{a}\|^2$$

を最小にするような \mathbf{a} を求めることが課題である。

$b = 0$ でない場合には、 X に1列足して、 X' とする。

加えた列は、すべて値を1とする。

また、**a**にも、1つ要素を増やしてやる。増やした分の推定値が**b**になる。

3 どうやるか

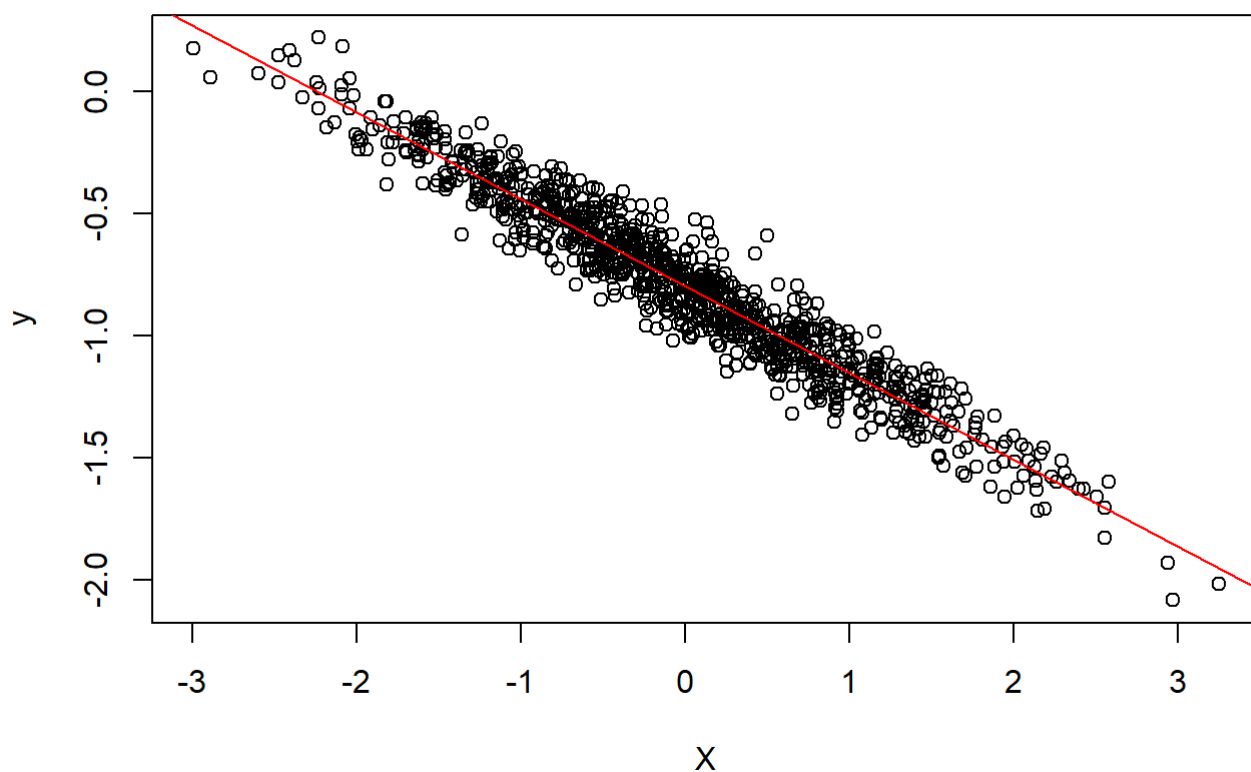
$$a = (X^T X)^{-1} X^T y$$

とにかく、計算してみる。

```
d <- 1
n <- 1000
X <- matrix(rnorm(d*n), ncol=d)
a <- rnorm(d)
b <- rnorm(1)
a
```

```
## [1] -0.3547459
```

```
y <- X %*% a + b + rnorm(n)*0.1
plot(X, y)
abline(b, a, col=2)
```



```
X. <- cbind(X, rep(1, n))
a. <- solve(t(X.) %*% X.) %*% t(X.) %*% y
a.
```

```
##           [, 1]  
## [1,] -0.3563211  
## [2,] -0.7967197
```

```
print(c(a, b))
```

```
## [1] -0.3547459 -0.7970394
```

確かにうまく行っている。

3.1 式を眺める

$$a = (X^T X)^{-1} X^T y$$

どうしてこの式でよいのか、というのは、ふつうの統計学の教科書に書いてある。

それよりは、この式の行列のサイズを確認することに時間を使ってみよう。

- X は $n \times d$ 行列。
- X^T は $d \times n$ 行列。
- $X^T X$ は $d \times d$ 正方行列。
- $(X^T X)^{-1}$ も $d \times d$ 正方行列。
- $(X^T X)^{-1} X^T$ は $d \times n$ 行列。
- $(X^T X)^{-1} X^T y$ は $d \times 1$ 行列。

求めたい a のそれと一致している。

```
dim(X.)
```

```
## [1] 1000    2
```

```
dim(t(X.))
```

```
## [1]    2 1000
```

```
dim(t(X.)%*%X.)
```

```
## [1] 2 2
```

```
dim(solve(t(X.)%*%X.))
```

```
## [1] 2 2
```

```
dim(solve(t(X.)%*%X.)%*%t(X.))
```

```
## [1] 2 1000
```

```
dim(solve(t(X.)*X.)*t(X.)*y)
```

```
## [1] 2 1
```

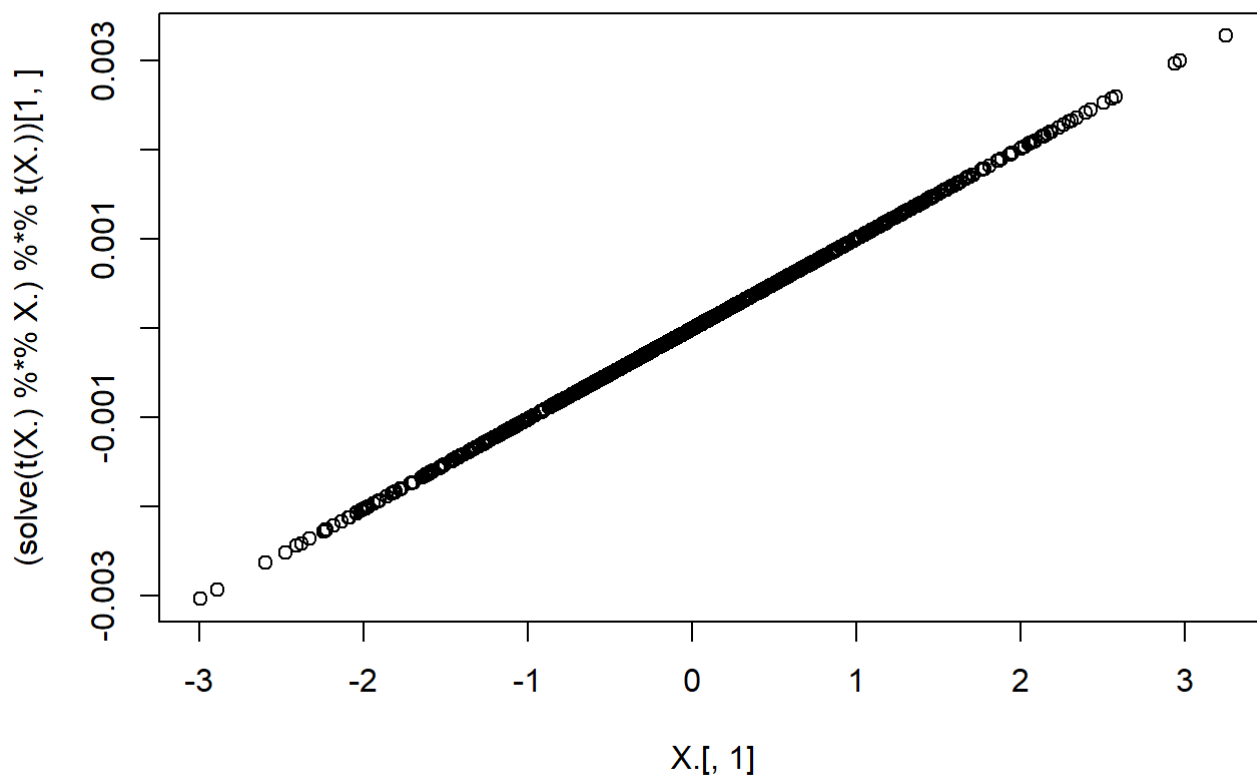
この要素の意味合いはなんだろうか？

- X は d 個の変数の値を列ベクトルとして持つ行列。
- X^T は n サンプルごとの変数の値を列ベクトルとして持つ行列。
- $X^T X$ は変数のペアワイズな内積を要素とする行列。異なる二つのベクトルの内積が大きいということは、それらが近い関係にあることを意味し、内積が0に近いということは、相互に独立性が強いことを意味する。
- $(X^T X)^{-1}$ は、ペアワイズな内積の逆数のような行列。逆にしたので、相互に独立な間柄を大きくとり扱い、相互に近い関係は軽く扱おうとする行列
- $(X^T X)^{-1} X^T$ は変数を重く扱うか軽く扱うかを考慮して、各サンプルの変数値を加減しなおした値を格納した行列
- $(X^T X)^{-1} X^T y$ は変数の軽重を考慮した上で、それが y の値を大きくする方に働いているか、小さい方に働いているかで重みづけをして足し合わせた値。
- それが係数。

もし、複数の説明変数があり、相互に相関が強いとすると、変数の重みづけの際に、両者の重みを小さ目にするることになり、結果として、個々の説明変数の係数は小さ目になることになる。

以下のプロットを見ると、説明変数の値と、重みづけを勘案しなおした値とは線形な関係にあることが解る。

```
plot(X[,1], (solve(t(X.)*X.)*t(X.))[1,])
```



```
#plot(X[, 2], (solve(t(X) %*% X) %*% t(X))[2, 1])
```

```
library(mvtnorm)
d <- 3
n <- 1000
X <- rmvnorm(n, mean=rep(0, d), sigma=diag(rep(1, d)))
a <- rnorm(d)
b <- rnorm(1)
a
```

```
## [1] 1.2031908 -0.8852727 0.4874919
```

```
y <- X %*% a + b + rnorm(n)*0.1
X. <- cbind(X, rep(1, n))
a. <- solve(t(X.) %*% X.) %*% t(X.) %*% y
a.
```

```
##           [, 1]
## [1,] 1.2020769
## [2,] -0.8865770
## [3,] 0.4856535
## [4,] -1.0588932
```

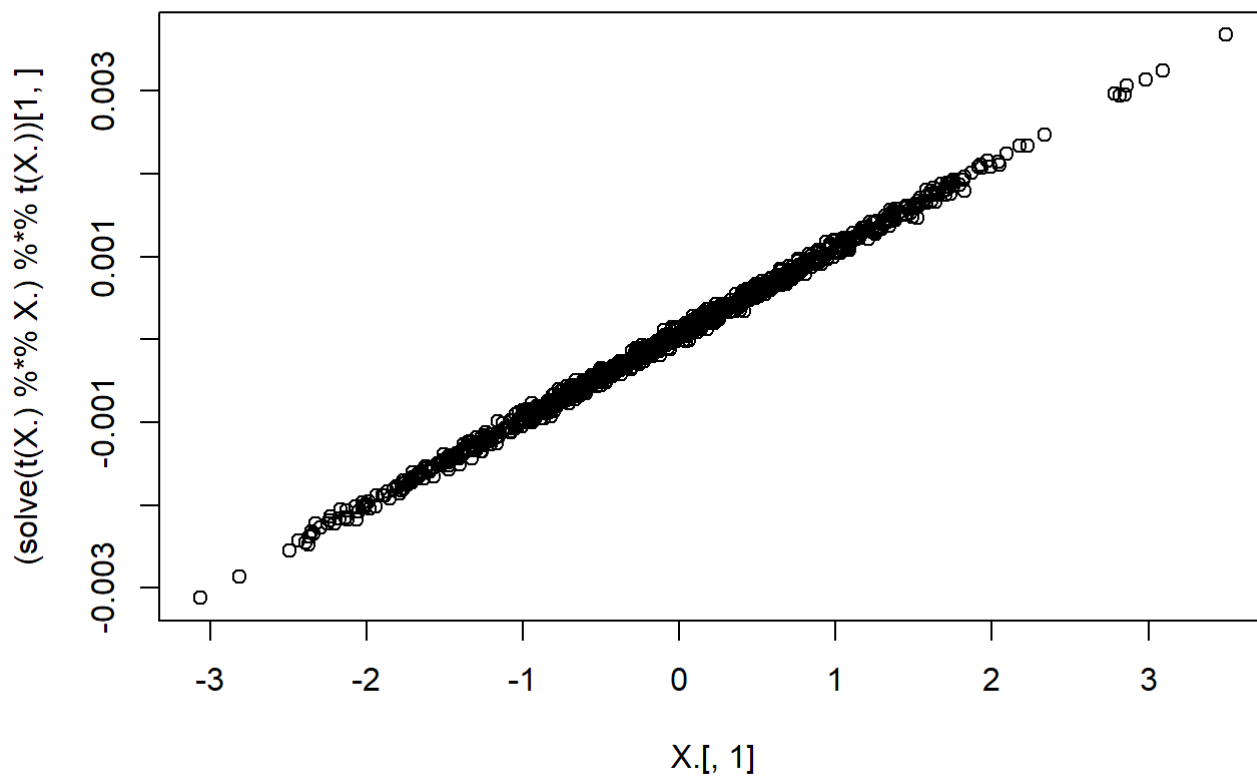
```
print(c(a, b))
```

```
## [1] 1.2031908 -0.8852727 0.4874919 -1.0584267
```

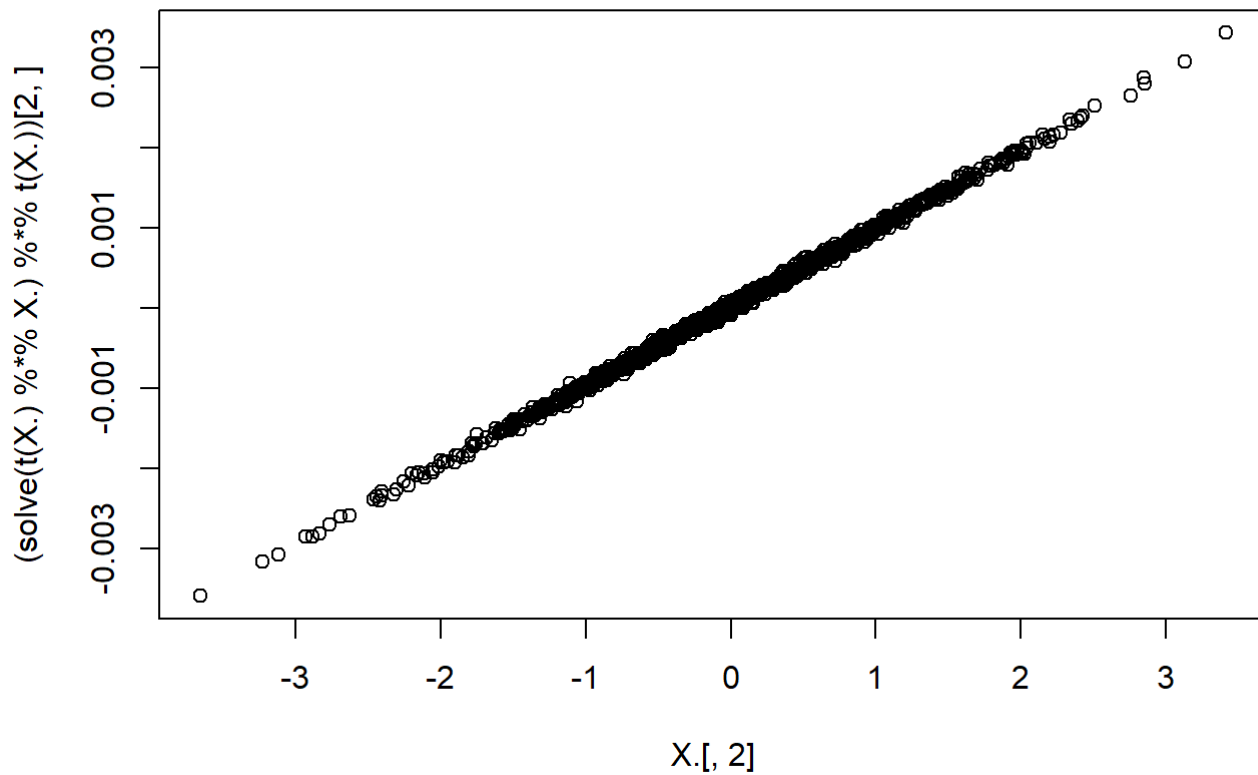
重み勘案前と後の値の関係をプロットしてみる。

これは3つの説明変数が相互に独立な場合である。なぜなら、行列 X を作ったときに3変数の分散共分散行列を単位行列で与えたからである。

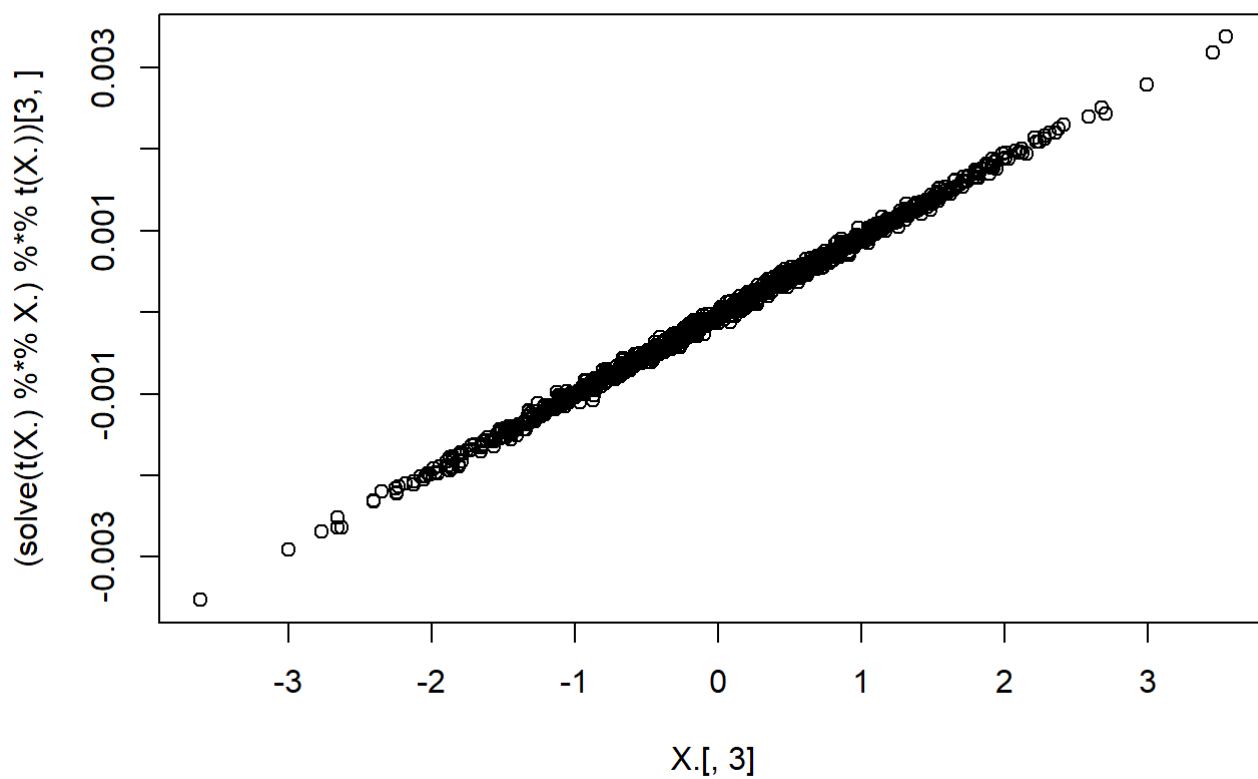
```
plot(X[, 1], (solve(t(X.)*X.)*t(X.))[1,])
```



```
plot(X[, 2], (solve(t(X.)*X.)*t(X.))[2,])
```



```
plot(X. [, 3], (solve(t(X.) %*% X.) %*% t(X.)) [3, 1])
```



4 Exercise 1

4.1 Exercise 1-1

説明変数 3 個の場合で、その 3 変数の分散共分散行列が単位行列でないように作成せよ。正定値行列の作成の仕方を思い出すこと。

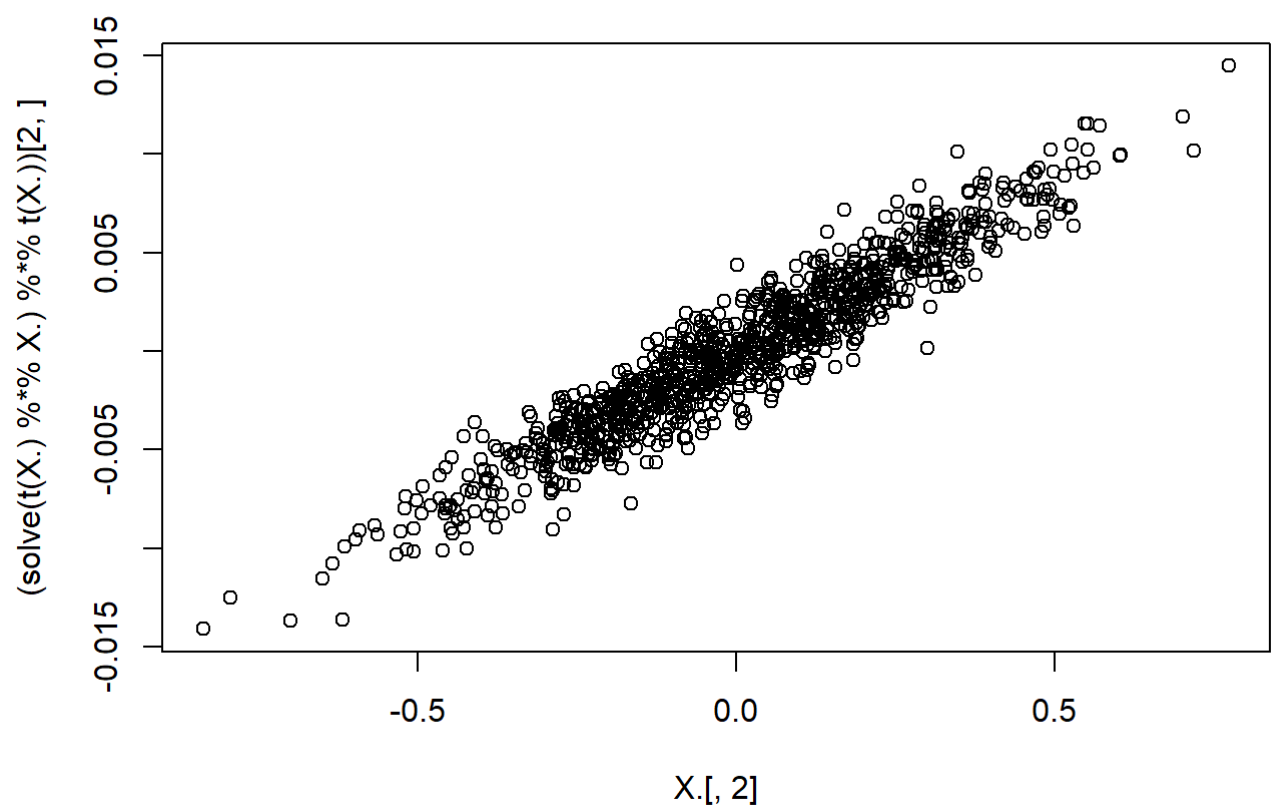
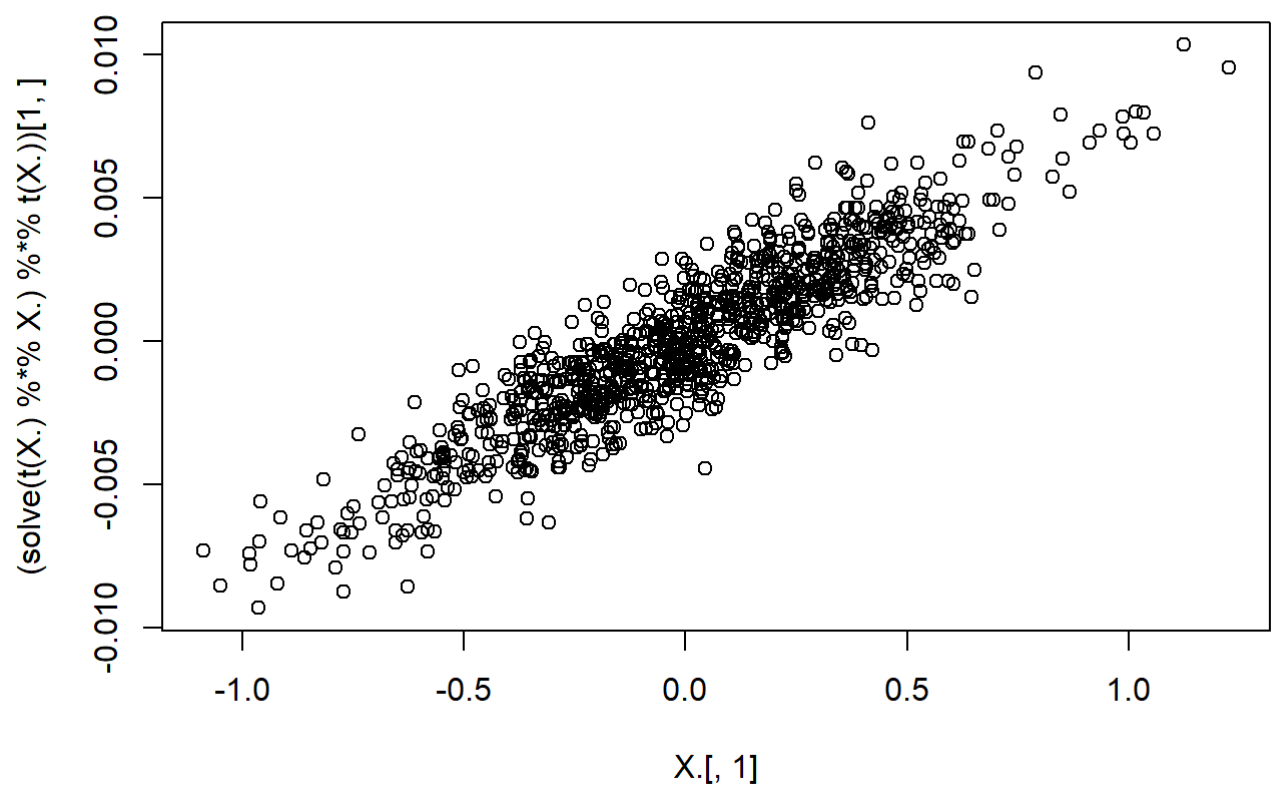
そのうえで、勘案前後のプロットをせよ。

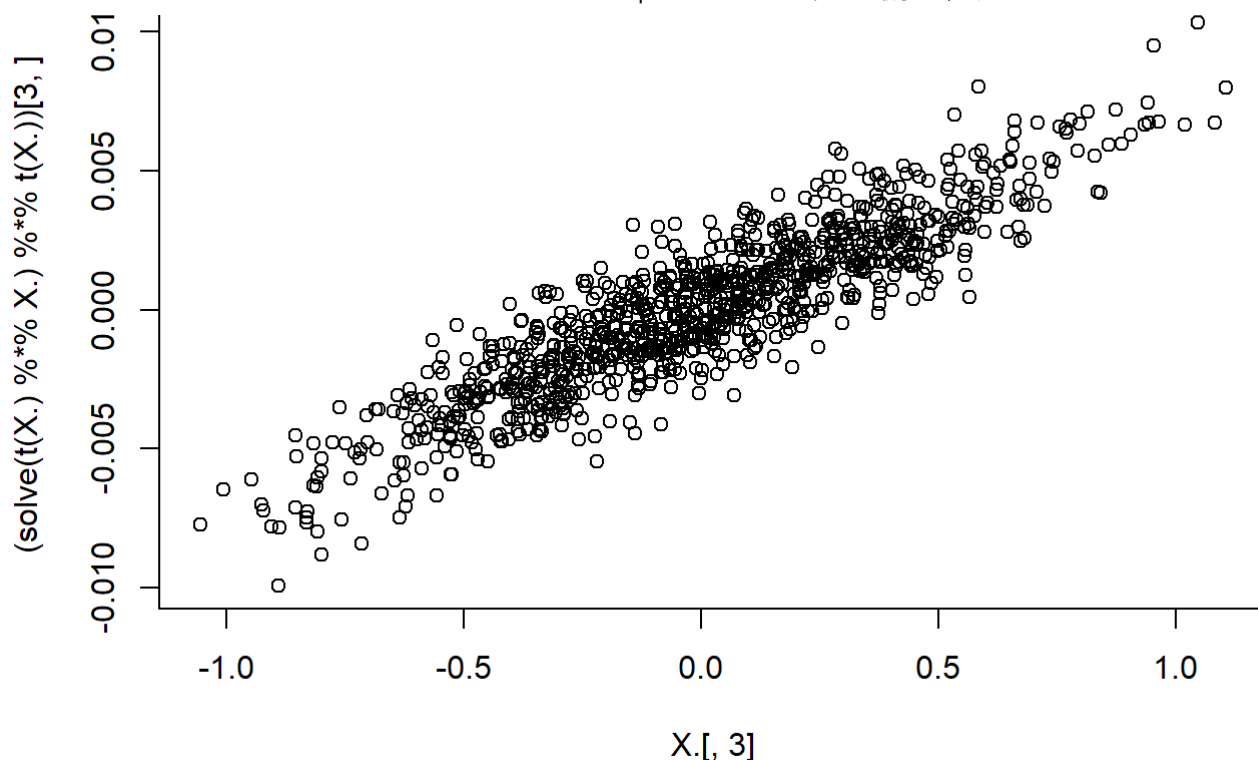
各変数の間に相関が生じているので、勘案のされ方は、各サンプルの 3 変数の値の取り方によって差が生じるため、勘案前後の関係にばらつきが生じることを確認せよ。

```
## [1] 0.2335561 -0.4809744 1.0113803
```

```
##           [, 1]  
## [1, ] 0.2530522  
## [2, ] -0.4905754  
## [3, ] 1.0101959  
## [4, ] -0.3671873
```

```
## [1] 0.2335561 -0.4809744 1.0113803 -0.3688956
```



5 固有値分解とは違う分解、QR分解、特異値分解

実際に

$$\mathbf{a} = (X^T X)^{-1} X^T$$

をこのまま解こうとすると、 $X^T X$ の逆行列を計算する必要がある。逆行列の計算は負荷が大きく誤差が出やすいという性質もあるので、逆行列を算出せずに、 \mathbf{a} を計算する方法がいくつか知られている。

いずれも、対角化・三角化行列を活用する方法である。

Rの線形回帰関数`lm()`では、デフォルトの計算方法としてQR法を取っていることが、`lm()`関数のソースを読むことで確認できる。

6 Exercise 2

6.1 Exercise 2-1

Rの関数は、関数名をプロンプトに書き込むことで、そのコードが読めることがある。`lm()`関数もそのようにしてコードが読める関数である。`lm()`関数のコードを表示させ、QR法がデフォルトであることを確認し、どのようにしてそれを確認したかを説明せよ。

```
lm
```

```

## function (formula, data, subset, weights, na.action, method = "qr",
##     model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
##     contrasts = NULL, offset, ...)
## {
##     ret.x <- x
##     ret.y <- y
##     cl <- match.call()
##     mf <- match.call(expand.dots = FALSE)
##     m <- match(c("formula", "data", "subset", "weights", "na.action",
##         "offset"), names(mf), 0L)
##     mf <- mf[c(1L, m)]
##     mf$drop.unused.levels <- TRUE
##     mf[[1L]] <- quote(stats::model.frame)
##     mf <- eval(mf, parent.frame())
##     if (method == "model.frame")
##         return(mf)
##     else if (method != "qr")
##         warning(gettextf("method = '%s' is not supported. Using 'qr'",
##             method), domain = NA)
##     mt <- attr(mf, "terms")
##     y <- model.response(mf, "numeric")
##     w <- as.vector(model.weights(mf))
##     if (!is.null(w) && !is.numeric(w))
##         stop("'weights' must be a numeric vector")
##     offset <- model.offset(mf)
##     mlm <- is.matrix(y)
##     ny <- if (mlm)
##         nrow(y)
##     else length(y)
##     if (!is.null(offset)) {
##         if (!mlm)
##             offset <- as.vector(offset)
##         if (NROW(offset) != ny)
##             stop(gettextf("number of offsets is %d, should equal %d (number of observation
## s)",
##                 NROW(offset), ny), domain = NA)
##     }
##     if (is.empty.model(mt)) {
##         x <- NULL
##         z <- list(coefficients = if (mlm) matrix(NA_real_, 0,
##             ncol(y)) else numeric(), residuals = y, fitted.values = 0 *
##             y, weights = w, rank = 0L, df.residual = if (!is.null(w)) sum(w !=
##             0) else ny)
##         if (!is.null(offset)) {
##             z$fitted.values <- offset
##             z$residuals <- y - offset
##         }
##     }
##     else {
##         x <- model.matrix(mt, mf, contrasts)
##         z <- if (is.null(w))
##             lm.fit(x, y, offset = offset, singular.ok = singular.ok,
##                 ...)
##         else lm.wfit(x, y, w, offset = offset, singular.ok = singular.ok,
##             ...)
##     }
##     class(z) <- c(if (mlm) "mlm", "lm")

```

```
##      z$na.action <- attr(mf, "na.action")
##      z$offset <- offset
##      z$contrasts <- attr(x, "contrasts")
##      z$xlevels <- .getXlevels(mt, mf)
##      z$call <- cl
##      z$terms <- mt
##      if (model)
##          z$model <- mf
##      if (ret.x)
##          z$x <- x
##      if (ret.y)
##          z$y <- y
##      if (!qr)
##          z$qr <- NULL
##      z
## }
## <bytecode: 0x0000000022303280>
## <environment: namespace:stats>
```