```
# 一日目
#  正規線形モデリング

#  2013 年 8 月 28 最終改訂
#  執筆者 馬場真哉（北大水産 M2）
#  ウェブサイト http://logics-of-blue.com/
#  ミスなどのご連絡は logics.of.blue★gmail.com までお願いします。
#  （星を@に変更）

#====================================
# R による統計モデリング
# この章は全てのプログラムを記述～実行する
#====================================

# ★★★★★★★★★★★★★★★★★★
# ★          写経はじめ          ★
# ★★★★★★★★★★★★★★★★★★


# こんにちは世界
print("Hallow World")

#  print は省略可能
"Hallow World"

#  ""を抜かすとエラー
Hallow World

#  簡単な計算
1 + 2
5*6
3 + 4/2 - 1
2^10

# コメント部の説明
# 1 + 2
```

```
37
38
39  #  変数の定義
40  # 「<-」の左右にはスペースを必ずあけること。
41  #  見やすいプログラムを書く癖をつけた方がよい
42  x
43  x <- 3
44  x
45  x*2
46
47
48  # データの読み込みと表示
49  #  データをコピーしてから実行する
50  data.0.clip <- read.delim("clipboard")
51  data.0.clip
52  head(data.0.clip)
53  pairs(data.0.clip)
54
55  # データの読み込み方法その 2
56  #  作業ディレクトリを変更してから以下を実行
57  getwd()
58
59  # Not Run
60  # setwd()
61  # 任意のワーキングディレクトリを指定
62  #  コンソールをクリック→ファイル→ディレクトリの変更　から指定できる
63  #  getwd()で場所を確認したうえで setwd()に値をコピペで渡すと楽。
64
65  #  読み込みはこの一行で OK
66  data.0 <- read.csv("data0.csv")
67  data.0
68  head(data.0)
69  pairs(data.0)
70
71
72  # データの中身の取出し
```

```
# attach()関数は"絶対に"使わないこと。
food           # エラーが出る
names(data.0) # data.0 に格納されているモノの名前を調べる
data.0$food   # OK


# プロット
#  興味の対象を「〜」の左側におく
#  興味の対象 length はえさの量によってどう変わるか？
plot(data.0$length ~ data.0$food)

#  興味の対象 length は薬の有無によってどう変わるか？
plot(data.0$length ~ data.0$medicine)


#  ちょっと複雑なプロット
#  長いプログラムは分かりやすく改行するとよい
#  インデント（左端にスペースを空ける）するとなお見やすい。
#  Google 基準ではインデントはスペース 2 つ分
#  ただし丸かっこで囲まれている場合は、一行目に合わせてインデントを付ける
#  ↑ただし、これはめんどくさいので私は使っていない。スペース 2 つで十分と思う。
#  無理に Google に合わせる必要はないが、改行&インデントは実務上必須。
#  私はタブを使うこともよくあるが、Google 基準では使ってはいけないらしい？

#  見づらいコードを書き続けると、いつか"必ず"後悔することになるので注意。
#  Google に合わせる必要は皆無だが、自分なりにルールを決める必要はある。

plot(
  data.0$length ~ data.0$food,
  col = c(2,3)[data.0$medicine],
  pch=16,
  ylab = "Length",
  xlab = "food",
  main = "薬の有無別、体長と餌の量の関係",
  cex.main = 1.5,
  font.lab = 2
```

```
109 )
110
111 # 凡例
112 legend(
113   "topleft",
114   legend = c("薬あり", "薬なし"),
115   col = c(2,3),
116   bty = "n",
117   pch = 16
118 )
119
120 # ヘルプ
121 ?legend
122
123 #  モデルの作成
124 # 応答変数：length
125 #  説明変数：food  &  medicine
126 #  興味のある対象（応答変数 length）は説明変数によってどれくらい変化するか？  の
127 モデリング
128 lm.model.0 <- lm(length ~ food + medicine, data = data.0)
129 lm.model.0
130
131 #  ANOVA による検定結果
132 #  説明は後程
133 anova(lm.model.0)
134
135 # より詳しい結果の表示→最初は飛ばす
136 summary(lm.model.0)
137
138 #  予測
139 predict(
140   lm.model.0,
141   newdata = data.frame(food = 50, medicine = "medicine"))
142
143 # 予測区間つき
144 predict(
```

```
  lm.model.0,
  newdata = data.frame(food = 50, medicine = "medicine"),
  interval = "prediction", level = 0.95)

#  newdata の指定なし
predict(lm.model.0)

# データの型の確認
class(data.0)

# ベクトルデータ
vec <- c(1, 2, 3, 4.5, 5, 6, 7.2, 8, 9, 9.9)
vec
int <- 1:10
int

#  等差数列
seq(from = 0.1, to = 1, by = 0.1)

# データフレームに格納する
d <- data.frame(
    vec = vec,
    int = int,
    seq = seq(from = 0.1, to = 1, by = 0.1)
)

#  データの取り出し方
d
d[1,]
d[,1]
d[2,3]
names(d)
d$vec
d[,c("vec")]
d[,c("vec", "seq")]

```

```
# 予測の図示
# newdata の作成
newfood <- seq(from = min(data.0$food), to = max(data.0$food), by = 1)
newfood

# 薬があるときのえさと体長の関係の予測のためのデータセット
new.1 <- data.frame(
  food = newfood,
  medicine = "medicine")

#  薬がない時のえさと体長の関係の予測のためのデータセット
new.2 <- data.frame(
  food = newfood,
  medicine = "na")

#  薬の有無別に予測
pred.1 <- predict(
  lm.model.0, newdata = new.1,
  interval = "prediction", level = 0.95)
pred.2 <- predict(
  lm.model.0, newdata = new.2,
  interval = "prediction", level = 0.95)

#  図示
# ちょっと複雑なプロット
# これは以前作ったものと同じ。
# コピペする方が早いし、正確。
# 意地を張ると遅くなるうえに不正確になるので注意。ぜひコピペしてください。
plot(data.0$length ~ data.0$food,
  col = c(2,3)[data.0$medicine],
  pch=16,
  ylab = "Length",
  xlab = "food",
  main = "薬の有無別、体長と餌の量の関係",
  cex.main = 1.5,
  font.lab = 2
```

```
)
# 予測値
lines(pred.1[,1] ~ newfood, col = 2, lwd = 2)
lines(pred.1[,2] ~ newfood, col = 2, lwd = 1, lty = 2)
lines(pred.1[,3] ~ newfood, col = 2, lwd = 1, lty = 2)
lines(pred.2[,1] ~ newfood, col = 3, lwd = 2)
lines(pred.2[,2] ~ newfood, col = 3, lwd = 1, lty = 2)
lines(pred.2[,3] ~ newfood, col = 3, lwd = 1, lty = 2)

# 凡例
legend("topleft",
  legend = c("薬あり", "薬なし"),
  col = c(2,3),
  bty = "n",
  pch = 16)


# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
# ♪         写経終わり           ♪
# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
```

```
#=====================================
# 統計の基本とｔ検定
#=====================================

#  大阪データ
osaka <- c(19,19,20,20,20,20,20,21,21,21)
#  期待値
19*(2/10) + 20*(5/10) + 21*(3/10)
mean(osaka)

#  標本分散
((19-20.1)^2)*(2/10) + ((20-20.1)^2) *(5/10) +
  ((21-20.1)^2) *(3/10)

#  普遍分散
length(osaka)
((19-20.1)^2)*(2/9) + ((20-20.1)^2) *(5/9) + ((21-20.1)^2) *(3/9)
var(osaka)


# 東京データ
tokyo <- c(-50, 0, 0, 20, 20, 20, 20, 20, 70, 70)

# 期待値
mean(tokyo)

#  標本分散
(-50-19)^2 * (1/10) + (0-19)^2 * (2/10) +
  (20-19)^2 *(5/10) + (70-19)^2 * (2/10)

#  不遍分散
(-50-19)^2 * (1/9) + (0-19)^2 * (2/9) +
  (20-19)^2 *(5/9) + (70-19)^2 * (2/9)
var(tokyo)


```

```
#============================
# t 検定
#============================

# ★★★★★★★★★★★★★★★★★★★
# ★          写経はじめ          ★
# ★★★★★★★★★★★★★★★★★★★

#  サンプルデータの作成
d <- c(-1, -1, 0, 0, 1, 3, 5, 6, 7, 7)

# 平均値
mean(d)

#  標準偏差
sd(d)

#  サンプルサイズ
length(d)

# 標準誤差
sd(d)  /  sqrt(length(d))

#  t 検定
mean <- mean(d)
std.error <- sd(d)  /  sqrt(length(d))

t.value <- mean / std.error
t.value
(1-pt(t.value, df=length(d)-1))*2

# R に入っている関数を使う
t.test(d)
```

```
#====================
# 統計モデルなｔ検定
#====================

# データの読み込みと表示
#  作業ディレクトリを変更してから以下を実行

# NOT RUN
# setwd()

data.0 <- read.csv("data0.csv")
lm.model.0 <- lm(length ~ food + medicine, data = data.0)
lm.model.0


# より詳しい結果の表示
#  ここにｔ value とｔ検定の結果が出ている。
summary(lm.model.0)


# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
# ♪          写経終わり          ♪
# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪


```

```
#==========================================
# 分散分析 ANOVA
#==========================================

# グラフを描くだけ

#  サンプルデータの作成
d2 <- data.frame(
  Y = c(c(1,2,3,4,5), c(4,5,6,7,8), c(7,8,9,10,11)),
  option = rep(c("A", "B", "C"), each=5)
)

d2

#  データの可視化
plot(d2$Y ~ d2$option)

# スライドに載せたような散布図形式のグラフを作る
par(mar=c(5,  6,  3,  3))
plot.default(
  d2$Y ~ d2$option,
  ylim=c(0,12), xlim=c(0.5,3.5),
  ylab="結果", xlab="選択肢",
  cex=2,
  cex.lab=2,  cex.main=3,
 xaxt="n")
axis(side=1,  1:3,  LETTERS[1:3])


# ★★★★★★★★★★★★★★★★★★
# ★          写経はじめ          ★
# ★★★★★★★★★★★★★★★★★★

#  サンプルデータの作成
d2 <- data.frame(
  Y = c(c(1,2,3,4,5), c(4,5,6,7,8), c(7,8,9,10,11)),
```

```
  option = rep(c("A", "B", "C"), each=5)
)

d2

#  データの可視化
plot(d2$Y ~ d2$option)

#  Y の総平均
mean(d2$Y)

#  option ごとの Y の期待値の算出
tapply(d2$Y, d2$option, mean)

# R で分散分析を実装する
lm.model.anova <- lm(Y ~ option, data=d2)

#  係数の確認
lm.model.anova
summary(lm.model.anova)

#  分散分析による予測
predict(lm.model.anova, data.frame(option=c("A", "B", "C")))

# 予測値は option 別の期待値に等しい
tapply(d2$Y, d2$option, mean)


#=======================================
# 分散分析モデルにおける検定
# 検定スライドを見せてから実施
#=======================================

# 予測値の変化の大きさ
d2.effect <- data.frame(
  Y = c(rep(3,5), rep(6,5), rep(9,5)),
```

```
  option = rep(c("A", "B", "C"), each=5)
)

d2.effect

#  偏差平方和（分散の分子の部分）
sum((d2.effect$Y-mean(d2.effect$Y))^2)

#  偏差平方和を自由度で割る
#  自由度　＝　自由に動ける値
#  d2.effect の中身は 3,6,9 の 3 種類のみ。
#  また d2.effect の期待値は元データの期待値と等しくなるという制約がある
#  自由に動ける値は　　「3　-　1　=　2」
#  予測値の差の大きさを表す分散
sum((d2.effect$Y-mean(d2.effect$Y))^2) / 2

effect <- sum((d2.effect$Y-mean(d2.effect$Y))^2) / 2
effect

# ノイズの大きさ
#  単に実データから予測値を引いただけ
#  ↓これが予測誤差(ノイズ)
d2$Y - d2.effect$Y

#  データフレームにまとめる
d2.noise <- data.frame(
  Y = d2$Y - d2.effect$Y,
  option = rep(c("A", "B", "C"), each=5)
)

d2.noise

#  偏差平方和（分散の分子の部分）
sum( (d2.noise$Y-mean(d2.noise$Y))^2 )

#  自由度
```

```
#  誤差というモノの制約＝合計値は 0 になる
#  そうしないと、単なるノイズ以上の意味を持ってしまうから。
#  ゆえに、option ごとに合計値が 0 にならなければならない。
#  サンプルサイズから option の個数（この場合 3）を引く。
length(d2.noise$Y)-3

# ノイズの大きさを表す分散
sum( (d2.noise$Y-mean(d2.noise$Y))^2 ) / 12

noise <- sum( (d2.noise$Y-mean(d2.noise$Y))^2 ) / 12
noise

# F 比
#  Effect は option の数-1 で割っている
#  Noise はサンプルサイズ-option 数で割っている
#  ゆえに、サンプルサイズが多くなるほど Effect と Noise の比は大きくなる。
effect
noise

# F 比
F.ratio <- effect/noise
F.ratio

#  検定
#  option によって有意に予測は変わらないことがわかっているデータを集める部分は
#  数式を使って省略する。
#  ここにも自由度が使われている。サンプルサイズが効いているということ。
1 - pf(F.ratio, 2, 12)


# 関数を使って計算
lm.model.anova <- lm(Y ~ option, data=d2)
anova(lm.model.anova)
summary(lm.model.anova)
```

```
480  # ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
481  # ♪          写経終わり            ♪
482  # ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
483
484  #====================================
485  #  ナイーブな予測(NULL.model)との比較
486  #  ナイーブ予測と比べて予測誤差は減少したのか？
487  #====================================
488
489  # NULL.model
490  lm.model.NULL <- lm(Y ~ 1, data=d2)
491  lm.model.NULL
492
493  # ナイーブ予測の結果
494  predict(lm.model.NULL)
495
496  #  ナイーブ予測残差の計算
497  #  ↓これが予測誤差(ノイズ)
498  d2$Y - 6
499
500  #  データフレームにまとめる
501  d2.noise.NULL <- data.frame(
502    Y = d2$Y - 6,
503    option = rep(c("A", "B", "C"), each=5)
504  )
505
506  d2.noise.NULL
507
508
509  #  ナイーブ予測の残差平方和
510  sum( (d2.noise.NULL$Y - mean(d2.noise$Y))^2 )
511
512  # F 比
513  #  予測残差の減少度  ÷  option 付き ANOVA 予測の予測残差の大きさ
514  sum.sq.ANOVA <- sum( (d2.noise$Y - mean(d2.noise$Y))^2 )
515  sum.sq.NULL <- sum( (d2.noise.NULL$Y - mean(d2.noise$Y))^2 )
```

```
516
517 sum.sq.ANOVA
518 sum.sq.NULL
519
520 # 自由度の差
521 3 - 1
522
523 # 分散分析モデルの予測誤差の分散　（再掲）
524 noise.ANOVA <- sum( (d2.noise$Y-mean(d2.noise$Y))^2 ) / 12
525 noise.ANOVA
526
527 # 見やすいように括弧を増やしている。本来は(sum.sq.NULL - sum.sq.ANOVA )/ 2
528 / Noise.ANOVA
529 # 残差の差分を自由度の差分で割ったものが分子
530 # option 入りの ANOVA モデルの予測残差の分散が分母
531 F.ratio2 <- ( (sum.sq.NULL - sum.sq.ANOVA )/ 2 ) / noise.ANOVA
532 F.ratio2
533
534 1 - pf(F.ratio2, 2, 12)
535
536
537 # ナイーブ予測と分散分析モデル予測との予測誤差の比較
538 anova(lm.model.NULL, lm.model.anova)
539
540 # 普通の ANOVA
541 anova(lm.model.anova)
542
543 # 分散分析とは予測残差の違いを検定するものだ、とご理解ください
544
545
546
547
548
549
```

```
#=================
# 回帰分析
#=================

# ★★★★★★★★★★★★★★★★★★★
# ★          写経はじめ          ★
# ★★★★★★★★★★★★★★★★★★★

#  サンプルデータ
d3 <- data.frame(
  Y = c(3, 5, 4, 6, 7),
  X = 1:5
)

d3
nrow(d3)
plot(d3$Y ~ d3$X)

summary(lm(Y ~ X, data=d3))
lm(Y ~ X, data=d3)
anova(lm(Y ~ X, data=d3))


# optim で最小二乗法

# 自作関数の作成
# 予測するための関数
yosoku <- function(a, b, X){
  Yhat <- a*X + b
  return(Yhat)
}

#  2*3 + 1
yosoku(a=2, b=1, X=3)


```

```
#  予測結果の残差平方を出す関数
zansa <- function(Y, Yhat){
  zansa <- (Y - Yhat)^2
  return(zansa)
}

# (2-3)^2
zansa(2, 3)


#  残差平方和計算
# for  ループを使用する（本当は使わなくていい。勉強用）
zansa2 <- numeric()
a <- 2
b <- 2
for ( i in 1:nrow(d3)){
  Yhat <- yosoku(a, b, d3$X[i])
  zansa2[i] <- zansa(d3$Y[i], Yhat)
}
zansa2
sum.zansa2 <- sum(zansa2)
sum.zansa2

plot(d3$Y ~ d3$X,ylim=c(0, 15),main=c("a=2, b=2 の時の予測結果"))
abline(a=2, b=2)


# 一つの関数にまとめる
OLS <- function(para){
  zansa2 <- numeric()
  for ( i in 1:nrow(d3)){
    Yhat <- yosoku(para[1], para[2], d3$X[i])
    zansa2[i] <- zansa(d3$Y[i], Yhat)
  }
  sum.zansa2 <- sum(zansa2)
  return(sum.zansa2)
```

```
}

OLS(c(2, 2))

# optim で最適化
optim(c(2, 2), OLS)

#  答え合わせ

lm.model <- lm(Y ~ X, data=d3)
lm.model

# 検定
anova(lm.model)


################################
# 実はこの関数を使った方が早い
# これを写経する必要はありません
#  しかし、実務上はこちらのプログラムを使った方が計算速度が速くなります。

OLS2 <- function(para){
  Yhat <- yosoku(para[1], para[2], d3$X)
  zansa2 <- zansa(d3$Y, Yhat)
  sum.zansa2 <- sum(zansa2)
  return(sum.zansa2)
}

OLS2(c(2, 2))

optim(c(2, 2), OLS2)

# 関数の仕組み

bai <- function(x) {
  return(2*x)
```

```
}

# 実はこの書き方でも OK
#  bai <- function(x) return(2*x)

bai(2)
sample.data <- 1:10
bai(sample.data)

#  このやり方はエラー
bai(1,2,3,4,5,6,7,8,9,10)

################################

# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
# ♪          写経終わり          ♪
# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪


#-----------------------------
# 回帰分析における分散分析
#-----------------------------

# 係数
lm.model <- lm(Y ~ X, data=d3)
lm.model

lm.model$coef

# 予測値の計算
yosokuti <- yosoku(lm.model$coef[2], lm.model$coef[1], d3$X)
yosokuti

#  予測値の変動の大きさ＝X が Y に与える影響の大きさの計算
heihou <- zansa(mean(yosokuti), yosokuti)
heihou
```

```
694
695 Effect <- sum(heihou) / 1
696 Effect
697
698 #  予測残差の大きさ  ＝  ノイズの大きさ
699 gosa <- zansa(d3$Y, yosokuti)
700 gosa
701
702 Error <- sum(gosa) / 3
703 Error
704
705 #  分散分析
706 F.value <-Effect / Error
707 F.value
708
709 1 - pf(F.value, 1, 3)
710
711 anova(lm.model)
712
713
714 # おまけ  NULL モデル(ナイーブ予測)との比較
715 lm.model.NULL <- lm(Y ~ 1, data=d3)
716 predict(lm.model.NULL)
717 mean(d3$Y)
718
719 anova(lm.model.NULL, lm.model)
720 anova(lm.model)
721
722
723
```

```
#================================
# パラメトリックブートストラップ検定
#================================

# ★★★★★★★★★★★★★★★★★★★
# ★          写経はじめ          ★
# ★★★★★★★★★★★★★★★★★★★

#  回帰分析の時に使ったデータの再掲
d3 <- data.frame(
  Y = c(3, 5, 4, 6, 7),
  X = 1:5
)

# モデルの作成
lm.model.NULL <- lm(Y ~ 1, data=d3)
predict(lm.model.NULL)

lm.model <- lm(Y ~ X, data=d3)
anova(lm.model)


#  シミュレーションデータの作成
# NULL モデルが正しいと仮定してシミュレーションする

simulate(lm.model.NULL, 1)

sim.data <- cbind(
  simulate(lm.model.NULL,1),
  1:5)

colnames(sim.data) <- c("Y","X")
sim.data
mean(sim.data$Y)
plot(sim.data$Y ~ sim.data$X)

```

```
760
761
762  # パラメトリックブートストラップ検定
763  # 本来はN.sim=10000くらいで行うことが望ましいが
764  # PCのスペックに自信のない方はN.sim=1000に変えてください
765  # apply系の関数を使うともっと早くできます。興味がある方は試してみてください。
766
767  set.seed(1)
768  Nsim <- 10000
769  sim <- simulate(lm.model.NULL, Nsim)
770  sim.F.value <- numeric()
771  for(i in 1:Nsim){
772    sim.data <- cbind(sim[i], 1:5)
773    colnames(sim.data) <- c("Y", "X")
774    model <- lm(Y ~ X, data=sim.data)
775    sim.F.value[i] <- summary(model)$fstatistic[1]
776  }
777
778  sim.F.value
779  max(sim.F.value)
780
781  # 見やすいヒストグラムを作る
782  F.value <- subset(sim.F.value, sim.F.value < 20)
783
784  # F比のヒストグラム
785  hist(F.value, xlim=c(0, 20))
786
787  #  0~1の範囲内にあるF比は10000個中6000個くらい
788  #  1~2の範囲内にあるF比は10000個中1400個くらい
789  #  2~3の範囲内にあるF比は10000個中650個くらい
790
791  #  ヒストグラムの数字バージョン
792  stem(F.value)
793
794  # 実際のF値と比較
795  #  このF比よりも大きなF比が出た回数を数えればよい
```

```
lm.model <- lm(Y ~ X, data=d3)
lm.model
summary(lm.model)
summary(lm.model)$fstatistic[1]

#  数え方
# 論理演算
2<3
2>3

#  相手がたくさんあっても大丈夫
c(1,2,3,4,5,6,7,8,9,10)>6

#  TRUE の個数がわかる
sum(c(1,2,3,4,5,6,7,8,9,10)>6)


# 実データから計算された F 比
summary(lm.model)$fstatistic[1]

#  PB 検定の p 値
sum(sim.F.value>=summary(lm.model)$fstatistic[1]) / Nsim

# 理論的に計算された p 値とほぼ等しい事を確認
anova(lm.model)


# 偉人の数式とシミュレーションの比較
# pf は F 比がある値を下回る確率を表す
#  df は F 比の確率分布を表す
#  確率分布とは、面積が 1 になるように標準化されたヒストグラムのこと
x <- seq(0, 20, by=0.1)
F <- df(x, 1, 3)
F

hist(F.value,xlim=c(0,20),ylim=c(0,1),prob=T,breaks=50)
```

```
lines(x, F, col=2, lwd=2)

# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
# ♪          写経終わり          ♪
# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪

#=======================================
#  スライドのグラフを作るためのスクリプト
#=======================================

# 正規分布
x <- seq(-10,10,by=0.01)
N1 <- dnorm(x, mean=0, sd=1)
N2 <- dnorm(x, mean=2, sd=1)
N3 <- dnorm(x, mean=0, sd=2)

plot(0,0, type="n", xlim=c(-7,7), ylim=c(0,0.5),
  xlab="",ylab="確率",main="正規分布",cex.main=2)
lines(N1~x, col=1, lwd=4)
lines(N2~x, col=2, lwd=4)
lines(N3~x, col=3, lwd=4)
abline(v=0)
legend(
  "topleft",
  lwd=4,
  col=c(1,2,3),
  bty="n",
  legend=c("平均 0、分散 1","平均 2、分散 1","平均 0、分散 4"),
  cex=1.5
)
```

```
#============================
# 正規線形モデルの再確認
#============================

# ★★★★★★★★★★★★★★★★★★★★
# ★          写経はじめ          ★
# ★★★★★★★★★★★★★★★★★★★★

# データの読み込み
# setwd()
data0 <- read.csv("data0.csv")
head(data0)

# 分散分析・回帰分析で解説したように、両者はやってることがほぼ同じ
#  説明変数がカテゴリデータか定量データかということは気にする必要がない
summary(data0)
pairs(data0)

#  モデルの作成
# 応答変数：length
#  説明変数：food  &  medicine
#  興味のある対象（応答変数 length）は説明変数によってどれくらい変化するか？
model0 <- lm(length ~ food + medicine, data = data0)
model0

# 分散分析結果の再確認
#  すべての出力の意味が分かるはず
#  変数を加えることによって予測残差は"有意に"減ったか？
anova(model0)

# 要約結果の再確認
summary(model0)

#  Call:モデルの式
#  Residuals：残差
#  Coefficients：係数。t 検定によるパラメタの有意性の検定結果
```

```
# →パラメタは有意に 0 と異なるか？
#  Residual standard error  残差の標準誤差
#  Multiple R-squared:
#  予測の変動の大きさ/(予測の変動の大きさ＋残差の変動の大きさ)
#  →モデルの当てはまりの良さの指標
#  Adjusted R squared:自由度調整済み R2。
#  →変数が多いと R2 が増えて当たり前。これを補正したもの。
#  F-statistic  ナイーブ予測と比べて予測残差はどれだけ減ったか？


# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
# ♪          写経終わり            ♪
# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪

#---------------------------
# シミュレーションと予測区間
#---------------------------
set.seed(1)
x.sim <- rnorm(mean=5, sd=10, n=100)
x.sim
hist(x.sim)
y.sim <- x.sim * 0.8 + 5 + rnorm(mean=0, sd=10, n=100)

plot(y.sim ~ x.sim)

model.sim <- lm(y.sim ~ x.sim)
summary(model.sim)

# 予測
new.sim <- data.frame(x.sim=seq(min(x.sim), max(x.sim), by=0.1))
new.sim

# 信頼区間
#  同じ確率分布から得られたデータを例えば 100 データセット集める。
#  一つのデータセットにつき 100 サンプルあるから、データは 1 万個
#  全部回帰分析で線を引っ張る
```

```
#  引っ張られた100本の線のうち、95本の線が中に入る区間
sinrai <- predict(
  model.sim, newdata=new.sim, se.fit=T, interval="confidence")
sinrai

plot(y.sim ~ x.sim)
lines(sinrai$fit[,1] ~ new.sim$x.sim, lwd=2)
lines(sinrai$fit[,2] ~ new.sim$x.sim, lwd=2, col=4)
lines(sinrai$fit[,3] ~ new.sim$x.sim, lwd=2, col=2)

# 予測区間
#  線を引く
#  そのあと、同じ確率分布から得られたデータを例えば100個集める。
#  その100個のデータのうち、95このデータはこの範囲内に収まる
# 予測値から左右対称になっていることに注意

yosoku <- predict(
  model.sim, newdata=new.sim, se.fit=T, interval="predict")
yosoku
yosoku$residual.scale
# ↑この値がYsimのノイズの標準偏差（10）に近いことを確認

plot(y.sim ~ x.sim)
lines(yosoku$fit[,1] ~ new.sim$x.sim, lwd=2)
lines(yosoku$fit[,2] ~ new.sim$x.sim, lwd=2, col=4)
lines(yosoku$fit[,3] ~ new.sim$x.sim, lwd=2, col=2)


# 両方載せたグラフ
plot(y.sim ~ x.sim)
lines(sinrai$fit[,1] ~ new.sim$x.sim, lwd=2)
lines(sinrai$fit[,2] ~ new.sim$x.sim, lwd=2, col=4)
lines(sinrai$fit[,3] ~ new.sim$x.sim, lwd=2, col=2)
lines(yosoku$fit[,2] ~ new.sim$x.sim, lwd=2, col=4, lty=2)
lines(yosoku$fit[,3] ~ new.sim$x.sim, lwd=2, col=2, lty=2)

```

```
976
977  #================================
978  # 中心極限定理
979  #================================
980
981  # サイコロの合計値は正規分布になるか？
982  N.sample <- 1
983  saikoro <- sample(1:6, size=N.sample, replace = T)
984  saikoro
985
986  # 一回サイコロを投げることを 1000 回実行
987  N.sim <- 1000
988  kekka <- numeric()
989  for(i in 1:N.sim){
990    kekka[i] <- sample(1:6, size=N.sample, replace = T)
991  }
992  kekka
993  hist(kekka, breaks=0:6)
994
995
996  # 五回サイコロを投げることを 1000 回実行
997  N.sample <- 5
998  N.sim <- 1000
999  kekka <- numeric()
1000 for(i in 1:N.sim){
1001   kekka[i] <- sum(sample(1:6, size=N.sample, replace = T))
1002 }
1003 kekka
1004 hist(kekka, breaks=min(kekka):max(kekka))
1005
1006
1007 # 十回サイコロを投げることを 1000 回実行
1008 N.sample <- 10
1009 N.sim <- 1000
1010 kekka <- numeric()
1011 for(i in 1:N.sim){
```

```
  kekka[i] <- sum(sample(1:6, size=N.sample, replace = T))
}
kekka
hist(kekka, breaks=min(kekka):max(kekka))


# 十回サイコロを投げることを 10000 回実行
N.sample <- 10
N.sim <- 10000
kekka <- numeric()
for(i in 1:N.sim){
  kekka[i] <- sum(sample(1:6, size=N.sample, replace = T))
}
kekka
hist(kekka, breaks=min(kekka):max(kekka))

#  単体では全く正規分布ぽくないサイコロの目が
#  合計値をとると正規分布によく似た確率分布（ヒストグラム）になっている。
```

```
1034
1035  #================================
1036  # モデル選択と AIC
1037  #================================
1038
1039  # ★★★★★★★★★★★★★★★★★★
1040  # ★          写経はじめ          ★
1041  # ★★★★★★★★★★★★★★★★★★
1042
1043  # データの読み込み
1044  #  ディレクトリを変更した後で実行する
1045  data1 <- read.csv("data1.csv")
1046
1047  #  データの確認
1048  head(data1)
1049  names(data1)
1050  levels(data1$option1)
1051  levels(data1$option2)
1052  summary(data1)
1053
1054  pairs(data1, panel=panel.smooth)
1055
1056
1057  # 4 つの説明変数を全て加えたモデルを作成
1058  lm.model.1 <- lm(Y ~ ., data=data1)
1059  summary(lm.model.1)
1060  anova(lm.model.1)
1061
1062  #  option2 が不要なようなので、これを削除したモデルを作成
1063  lm.model.2 <- update(lm.model.1, ~ .-option2)
1064  lm.model.2
1065
1066  #  変数を切った時、予測誤差は有意に増加したか？
1067  anova(lm.model.1, lm.model.2)
1068
1069  #  option2 を切ったモデルで再度分散分析
```

```
anova(lm.model.2)


#  x1 もいらなさそうだったので削除する
lm.model.3 <- update(lm.model.2, ~ .-x1)

#  変数を切った時、予測誤差は有意に増加したか？
anova(lm.model.2, lm.model.3)

#  二つの変数を削除した後のモデルでもう一度分散分析
anova(lm.model.3)


#  念のため x2 も削除してみる
lm.model.4 <- update(lm.model.3, ~ .-x2)

#  x2 という変数がなくなった時、予測残差は有意に増えたか？
anova(lm.model.3, lm.model.4)

#  Best なモデルは lm.model.3 に決定
best.model.anova <- lm.model.3
summary(best.model.anova)

#  ベストじゃない変数全部入りモデルとの比較
summary(lm.model.1)


#  AIC によるモデル選択
#  install.packages("MuMIn")  をした後で実行
library(MuMIn)

#  変数を抜き差しして、すべての変数の組み合わせにおいて AIC を算出して比較する
list <- dredge(lm.model.1, rank="AIC")
list

all.model <- get.models(list)
```

```
best.model.AIC <- all.model[1]

# AIC と検定の比較
# 両方同じ結果になった
best.model.AIC
best.model.anova


# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
# ♪          写経終わり            ♪
# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
```

```
#==========================
# 8章  Type II ANOVA
#==========================

# ★★★★★★★★★★★★★★★★★★★★
# ★          写経はじめ          ★
# ★★★★★★★★★★★★★★★★★★★★


# データの読み込み
#  ディレクトリを変更した後で実行する
data2 <- read.csv("data2.csv")

# データの確認
#  売上（sell）を上げるためにはどうすればよいかを知りたい、という設定
#  experience：チラシを配る店員さんの経験年数
#  n.sheets：配れたチラシの枚数
#  time：チラシを配った時間帯（昼か夜か）
#  sex：チラシを配った店員さんの性別
head(data2)
pairs(data2,panel=panel.smooth)
summary(data2)

#  やってはいけない解析方法
#  統計モデルを使わずに、一つ一つ検定していく

#  経験・チラシ配布枚数は売り上げに貢献しているか？  @t 検定
summary(lm(sell ~ experience, data=data2))
summary(lm(sell ~ n.sheets, data=data2))

#経験・チラシ配布枚数は売り上げに貢献しているか？  @分散分析
anova(lm(sell ~ experience, data=data2))
anova(lm(sell ~ n.sheets, data=data2))

#  チラシを配った人の性別と、配った時間は売り上げに貢献しているか？  @t 検定
t.test(data2$sell ~ data2$sex)
```

```
t.test(data2$sell ~ data2$time)


#  4 つの変数を一つにまとめて統計モデリングする
sell.model1 <- lm(sell ~., data = data2)

#  係数が変数を一つずつ入れた時と大きく異なっていることに注意
#  特に experience の値がおかしい
sell.model1$coef

#  分散分析する。
anova(sell.model1)

#  各係数を t 検定する
summary(sell.model1)

# 説明変数の関係性
par(mfrow=c(1,2))
plot(data2$n.sheets ~ data2$experience)
plot(data2$sex ~ data2$time)
par(mfrow=c(1,1))
```

```
#=======================
# Type II ANOVA
#=======================

#  install.packages("car")  をしたあとに実行
library(car)

#  普通の分散分析(Type I ANOVA)
anova(sell.model1)

#  Type II ANOVA
Anova(sell.model1,type=c("II"))

#  性別はいらなさそうだったので切る
sell.model2 <- update(sell.model1, ~.-sex)
anova(sell.model1, sell.model2)

Anova(sell.model2)

#  さらに経験もいらなさそうだったので切る
sell.model3 <- update(sell.model2, ~.-experience)
anova(sell.model2, sell.model3)

Anova(sell.model3)

#  これがベストモデル
summary(sell.model3)


# AIC

library(MuMIn)
sell.model.list <- dredge(sell.model1, rank="AIC")
sell.model.list

all.model.sell <- get.models(sell.model.list)
```

```
best.sell.AIC <- all.model.sell[1]
best.sell.AIC


# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
# ♪          写経終わり          ♪
# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪


```

```
#============================
# 交互作用
#============================

# ★★★★★★★★★★★★★★★★★★★
# ★          写経はじめ          ★
# ★★★★★★★★★★★★★★★★★★★


# データの読み込み
#  ディレクトリを変更した後で実行する
data3 <- read.csv("data3.csv")


# データの確認
head(data3)
summary(data3)
pairs(data3, panel=panel.smooth)

#  気温や天気がビールの利益に与える影響は？
# beer:ビールによる利益
# temperature:気温
# weather:天気

#  ダメな分析手法、変数を別個に入れて検定
summary(lm(beer ~ temperature, data=data3))
t.test(data3$beer ~ data3$weather)

#  変数を一緒にしてモデリング
model.beer0 <- lm(beer~., data=data3)
library(car)
Anova(model.beer0)

#  晴れの日、雨の日でデータを分割
fine <- subset(data3, weather=="fine")
rain <- subset(data3, weather=="rain")
```

```
1260
1261  #  お天気別気温の影響
1262  par(mfrow=c(1,2))
1263  plot(fine$beer~fine$temperature, main="晴れ")
1264  plot(rain$beer~rain$temperature, main="雨")
1265  par(mfrow=c(1,1))
1266
1267  #  交互作用を入れる
1268  # model の formula
1269  #  どれを使っても、今回は、結果は同じ
1270  #  ＋  で区切られたのが主効果（交互作用じゃないもの）
1271  #  :  でつながったものが交互作用
1272  lm(beer ~ temperature + weather + temperature:weather, data=data3)
1273  lm(beer ~ temperature * weather , data = data3)
1274  lm(beer ~ (.)^2, data = data3)
1275
1276  #  交互作用入りのモデルを使って再度検定
1277  model.beer <- lm(beer ~ (.)^2, data = data3)
1278  summary(model.beer)
1279
1280  # anova
1281  anova(model.beer)
1282
1283  #  Type II ANOVA
1284  Anova(model.beer, type="II")
1285
1286  # weather の部分が Type I ANOVA と全くいっしょ
1287  #  主効果の検定をする際に、交互作用を無視してしまっている。
1288
1289  #  Type III ANOVA
1290  #  交互作用が入っている時はこっちを使った方がよいかも
1291  #  交互作用に着目する場合は Type III ANOVA を使うべき
1292  #  主効果の影響を主に見ているのならば、Type II ANOVA のままでよいという説もある。
1293  Anova(model.beer, type="III")
1294
1295  #  3 つの ANOVA の比較
```

```
anova(model.beer)
Anova(model.beer, type="II")
Anova(model.beer, type="III")

# 交互作用入りモデルの係数
model.beer$coef

#  AIC でモデル選択
library(MuMIn)
dredge(model.beer, rank="AIC")


#  結果の図示
plot(data3$beer ~ data3$temperature,
  col=c(1,2)[data3$weather],
  pch = 16,
  xlab = "気温",
  ylab = "ビールによる利益",
  main = "お天気別ビールによる利益",
  font.lab = 2
)
legend("bottomright", pch = 16, col = c(1,2), legend = c("fine","rain"))

#  回帰直線を引く
#  各係数の使い方に注意
abline(
  model.beer$coef[1],
  model.beer$coef[2],
  lwd = 2)
abline(
  model.beer$coef[1] + model.beer$coef[3],
  model.beer$coef[2] + model.beer$coef[4],
  lwd = 2, col = 2)

#  晴れの日の傾きと切片
model.beer$coef[1]
```

```
model.beer$coef[2]

#  雨の日の傾きと切片
model.beer$coef[1] + model.beer$coef[3]
model.beer$coef[2] + model.beer$coef[4]


# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪
# ♪          写経終わり            ♪
# ♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪♪

#========================
#  おまけ
# 両方カテゴリデータの時
#========================

data <- data.frame(Y = c(0,5,3,10,1,6,2,13),
  sisaku_A = rep(c("a.not","act","a.not","act"),2),
  sisaku_B = rep(c("a.not","a.not","act","act"),2))

data

model <- lm(Y ~ (.)^2, data=data)
model$coef

model.2 <- lm(Y ~ sisaku_A, data=data, subset=(sisaku_B == "a.not"))
model.2$coef

model.3 <- lm(Y ~ sisaku_A, data=data, subset=(sisaku_B == "act"))
model.3$coef


model$coef
model.2$coef
model.3$coef
```

```
# subset の意味
data
subset(data, data$sisaku_B == "a.not")
subset(data, data$sisaku_B == "act")

################
#  もっとわかりやすい（が、ほんとはダメな）例

data <- data.frame(Y = c(1,5,3,10),
  sisaku_A = rep(c("a.not","act","a.not","act"),1),
  sisaku_B = rep(c("a.not","a.not","act","act"),1))

data

model <- lm(Y ~ (.)^2, data=data)
model$coef

model.2 <- lm(Y ~ sisaku_A, data=data, subset=(sisaku_B == "a.not"))
model.2$coef

model.3 <- lm(Y ~ sisaku_A, data=data, subset=(sisaku_B == "act"))
model.3$coef


model$coef
model.2$coef
model.3$coef

subset(data, data$sisaku_B == "a.not")
subset(data, data$sisaku_B == "act")

# 本当はこんなデータ使ってはいけない。あくまで、わかりやすさを優先した解説
anova(model)

#  お疲れ様でした
```