第1回 : 多変量解析の概要と線形回帰の復習

Code

1. 多変量解析の概要

多変量解析（multivariate analysis）という言葉は、複数の変数からなるデータを分析するために考案された様々な手法の総称です。続・初級統計学では、この「多変量解析」について詳しく学んでいきます。

1.1 データ

一般にデータは、各行に複数の変数を観測した結果を表にまとめた形で表されます。データの一例として、講義資料の data ディレクトリにある salary.csv という名前のcsvファイルを開けてみてください。このデータはある会社の社員に関する情報を記録したデータです。

Rに読み込んでから確認することもできます。Rにデータを読み込むときは

1. 作業ディレクトリの指定
2. 以下のスクリプトを実行

という流れをおこないます。作業ディレクトリの指定は、RStudioの場合、 Session -> Set Working Directory -> Choose Directory と進むことで可能です。今回は、 lecture01 ディレクトリを作業ディレクトリに指定してください。ここまで出来たら、以下のスクリプトを実行してみましょう。

Hide

```
# データが表の形で表される様子
dat <- read.csv(file = "./data/salary.csv",
                fileEncoding = "utf-8")
head(x = dat, n = 5)    # 先頭5行を確認する。
```

	月給 <int>	勤続年数 <int>	仕事の達成度 <int>	欠勤日数 <int>	特殊免許の有無 <int>
1	245803	5	144	6	0
2	241428	1	234	5	1
3	272966	6	240	5	1
4	326961	19	207	4	1
5	240608	1	189	5	1

5 rows

Remark : エラーが出力され、 ファイル './data/salary.csv' を開くことができません: No such file or directory という警告メッセージが出力されることがあります。この場合、

- ファイル名に綴りのミスがある。
- ディレクトリの変更に失敗している。

などが主な失敗にあげられます。ファイル名に綴りのミスが見当たらない場合、作業ディレクトリを getwd() 関数で出力し、 lecture01 になっているか確認してみてください。■

データの各行を**標本点**（sample point）といい、標本点の個数を**標本サイズ**（sample size）といいます。また各列を **変数** （variable）といい、変数の個数を**データの次元**（dimension）といいます。

問題 : 先ほどRに読み込んだデータ salary.csv について、標本サイズと次元を答えてください。また、変数の名前をすべて列挙してください。Hint : str 関数を用いると便利です。

解答 :
以下のスクリプトを実行してみましょう。

Hide

```
str(dat)
```

```
'data.frame':   100 obs. of  5 variables:
 $ 月給          : int  245803 241428 272966 326961 240608 261362 275045 236078 258850 280603 ...
 $ 勤続年数      : int  5 1 6 19 1 5 6 4 3 9 ...
 $ 仕事の達成度  : int  144 234 240 207 189 212 224 173 263 214 ...
 $ 欠勤日数      : int  6 5 5 4 5 5 4 7 6 4 ...
 $ 特殊免許の有無: int  0 1 1 1 1 1 1 0 1 0 ...
```

この出力から、標本サイズは 100 、次元は 5 、変数の名前は
月給 、 勤続年数 、 仕事の達成度 、 欠勤日数 、 特殊免許の有無 だとわかります。■

1.2 多変量解析の様々な課題と手法

問題 : salary.csv は、ある会社の社員に関する 月給 、 勤続年数 、 仕事の達成度 、 欠勤日数 、 特殊免許の有無 の情報が記録されたデータです。このデータから分析できる課題を考えてみましょう。（できれば複数考えてみてください。）

解答 : 例を4つほど掲げます。

1. 社員の 月給 の傾向が 特殊免許の有無 によって異なるかを検討する。
2. 社員の 月給 の傾向を 勤続年数 、 仕事の達成度 、 欠勤日数 、 特殊免許の有無 の4変数から説明できないか検討する。
3. 勤続年数 、 仕事の達成度 、 欠勤日数 、 特殊免許の有無 の4変数を用いて、社員の「熟練度」を1つの数値で表すような指標を作れないか検討する。
4. 仕事の達成度 、 欠勤日数 、 特殊免許の有無 の3変数を用いて、似たような勤務態度の社員をグルーピングできないか検討する。

以上は一例です。他にも様々な課題を考えることができます。■

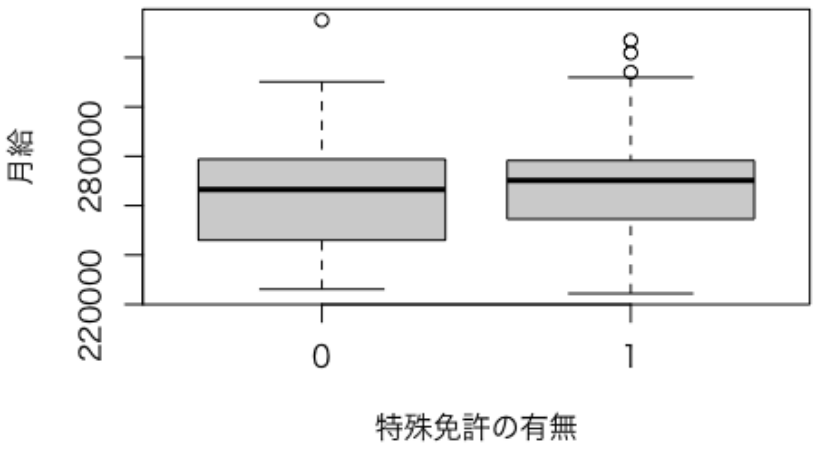
解答 (1) は、初級統計学で習う手法によって検討できます。例えば、

1. 横軸を 特殊免許の有無 、縦軸を 月給 とした「箱ひげ図」をかいて両群の月給を比較する。
2. 統計的仮説検定、今回の場合は母平均の差の検定を行う。

という流れが考えられます。箱ひげ図から、両群の月給の分布には、あまり差がないことが見てとれます。また、母平均の差の検定からも帰無仮説は棄却されず、両群の月給の母平均に差があるとは言えないという結論になります。

Hide

```
# 箱ひげ図
par(family = "ヒラギノ角ゴシック W3")    # Macユーザーのみ
boxplot(月給 ~ 特殊免許の有無, data = dat)
```



Hide

```
# 母平均の差の検定（両側検定、有意水準5%に設定した。）
t.test(月給 ~ 特殊免許の有無, data = dat,
         alternative = "two.sided", conf.level = 0.95)
```

```
    Welch Two Sample t-test

data:  月給 by 特殊免許の有無
t = -0.87961, df = 97.957, p-value = 0.3812
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -12553.174   4842.607
sample estimates:
mean in group 0 mean in group 1
       265917.4        269772.7
```

一方、(2), (3), (4)のように3つ以上の変数を対象とするような検討課題に対しては、初級統計学では触れてきませんでした。このような課題に対して有効な手法が「多変量解析」です。

さて、多変量解析と一口に言っても、想定される課題は様々あることがわかります。解答の(2)は予測や説明を目的とした課題であるのに対し、解答(3)や解答の(4)は変数間の共通性や標本点間の類似性の抽出といったデータの構造の理解を目指しています。また、各課題に対応する多変量解析の手法も一つとは限りません。様々な観点から、たくさんの手法が提案されています。なお続・初級統計学では、以下に掲げる多変量解析の手法に絞って解説していきます。

- 線形回帰
- 累乗モデル
- ロジスティック回帰
- 判別分析
- 主成分分析
- 因子分析
- k-means法
- Ward法
- 決定木とアンサンブル学習
- サポートベクトルマシン
- ニューラルネットワークと深層学習
- 多次元尺度構成法
- コレスポンデンス分析
- 共分散構造分析（構造方程式モデリング）

Remark : このように、多変量解析は多様性に富んだ一大分野です。多変量解析の勉強に挫折する方の多くは、この多様性に圧倒されているようにも思います。初学者の方は、まず様々な多変量解析の手法について、その概要をおさえることを目指してみてください。特に、

- どのような課題に対する手法なのか。
- どのような結果を計算する手法なのか。
- どのような仕組みになっているのか。

この3点をよく把握することをお勧めします。

2. 線形回帰の復習

初級統計学では「回帰分析」、正確には線形回帰とよばれる手法について勉強しました。その復習から始めていきましょう。

2.1 線形回帰の概要

A. 課題設定

データに D 個の変数 $x_1, \cdots, x_D$ と y が記録されているとき、変数 y を変数 $x_1, \cdots, x_D$ の1次式

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_D x_D + \text{誤差}$$

で表現することで説明・予測できると仮定します。例えば、社員の 月給 の傾向を 勤続年数 、 仕事の達成度 、 欠勤日数 、 特殊免許の有無 の4変数から説明できないか検討したい場合、y を 月給 、$x_1, \cdots, x_4$ を 勤続年数 、…、 特殊免許の有無 と置き換え

id	y	x_1	x_2	x_3	x_4
1	245803	5	144	6	0
2	241428	1	234	5	1
3	272966	6	240	5	1
⋮	⋮	⋮	⋮	⋮	⋮

この場合、

$$\text{月給} = \beta_0 + \beta_1 \times \text{勤続年数} + \beta_2 \times \text{仕事の達成度} + \beta_3 \times \text{欠勤日数} + \beta_4 \times \text{特殊免許の有無} + \text{誤差}$$

という式を考えています。$x_1, \cdots, x_D$ を**説明変数**（explanatory variable）、y を**目的変数**（objective variable） といいます。

B. 出来ること

$\beta_0, \beta_1 \cdots, \beta_D$ を**偏回帰係数**（coefficient）、特に β_0 を**切片**（intercept）といいます。これらの係数を推定することで

- 説明変数 x_1 だけが 1 異なり、他の説明変数 $x_2, \cdots, x_D$ が同じ場合、目的変数 y の値の差はどの程度か？
- 各説明変数の値が決まっているとき、目的変数の値はいくらと予測できるか？

を推定することができます。また、偏回帰係数の区間推定や統計的仮説検定を行うことができます。これについてはデモで例を交えながら解説します。

C. 偏回帰係数の推定の仕組み

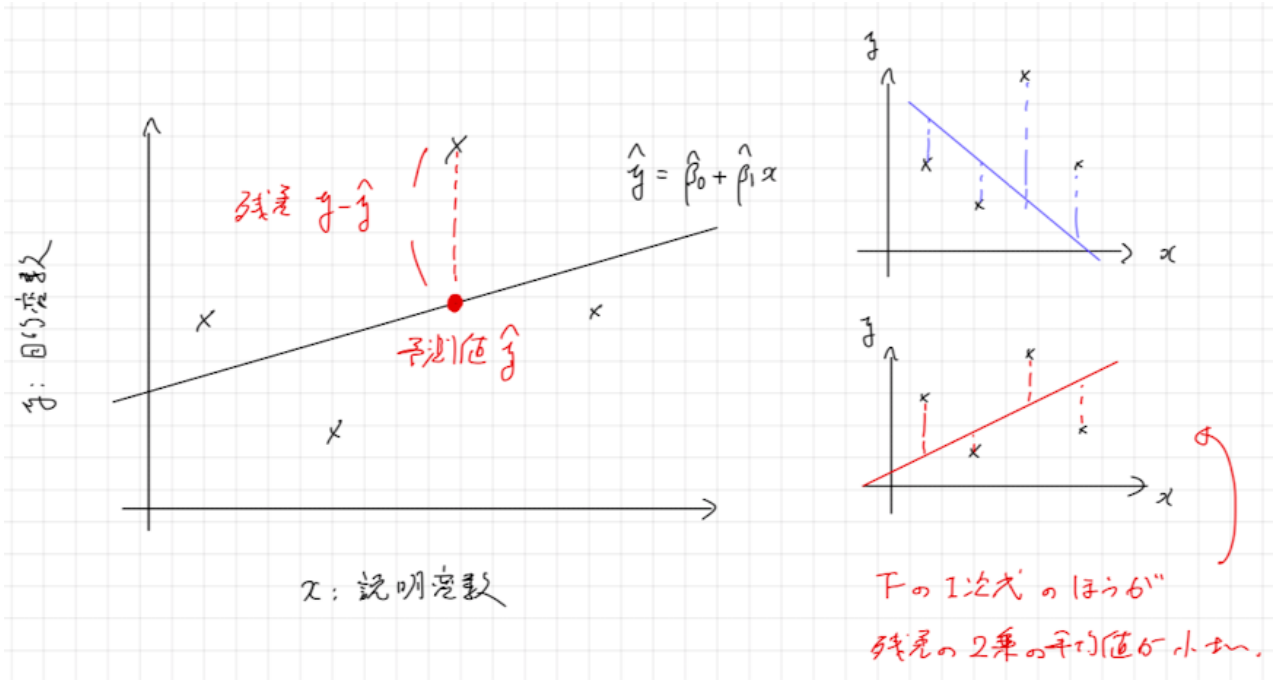
偏回帰係数の推定値 $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_D$ は、データ全体で

- 偏回帰係数から決まる各標本点の予測値 $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_D x_D$
- 実測値 y

の差の2乗（**残差**といいます） $(y - \hat{y})^2$ の平均値

$$l(\beta_0, \beta_1, \cdots, \beta_D) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

が最小になるように決めます。



Remark : 2乗は符号を打ち消すためについています。それなら、絶対値 $|y_i - \hat{y}_i|$ でも良いではないかと思う方がいるかもしれません。実際に、l^1-損失といってこちらを使うこともあるのですが、今回は詳しく触れないことにします。■

2.2 線形回帰のデモ

salary.csv は、ある会社の社員に関する 月給 （y）、 勤続年数 （x_1）、 仕事の達成度 （x_2）、 欠勤日数 （x_3）、 特殊免許の有無 （x_4） の情報が記録されたデータです。今回、社員の 月給 の傾向を 勤続年数 、 仕事の達成度 、 欠勤日数 、 特殊免許の有無 の4変数から説明できないか検討していきます。今回は、月給 y を

$$\text{月給} = \beta_0 + \beta_1 \times \text{勤続年数} + \beta_2 \times \text{仕事の達成度} + \beta_3 \times \text{欠勤日数} + \beta_4 \times \text{特殊免許の有無} + \text{誤差}$$

で説明するモデル、つまり線形回帰を考えます。

step 1. 線形性の確認

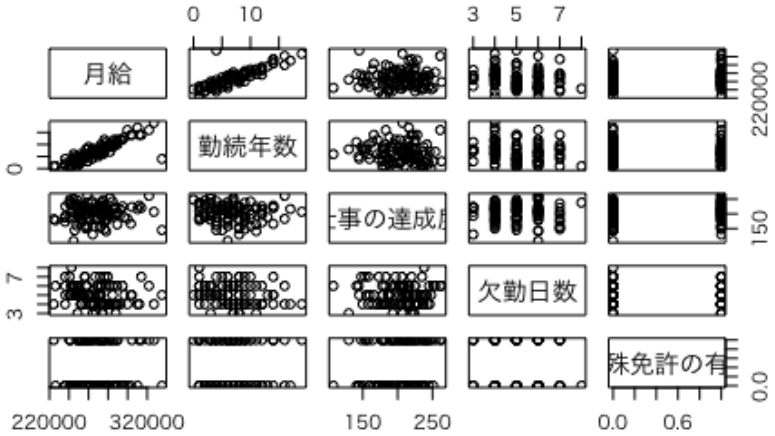
まず、以下のような1次式（線形性）を仮定することが妥当なのかを検討しておきましょう。

$$\text{月給} = \beta_0 + \beta_1 \times \text{勤続年数} + \beta_2 \times \text{仕事の達成度} + \beta_3 \times \text{欠勤日数} + \beta_4 \times \text{特殊免許の有無} + \text{誤差}$$

これは散布図をかくことで検討することができます。

Hide

```
# 散布図
par(family = "ヒラギノ角ゴシック W3")    # Macユーザーのみ
plot(dat)
```



問題 : 散布図から「線形性」を仮定しても問題ないかを検討してください。

解答 : 検討の一例を紹介します。散布図から目的変数と説明変数の間には、それぞれ線形の関係がみられることが確認できます。これより「線形性」を仮定してもよいと考えられます。■

B. 偏回帰係数の推定

切片および偏回帰係数 $\beta_0, \beta_1, \cdots, \beta_4$ の推定値を求めます。R言語では、 lm 関数を用いて線形回帰を計算できます。以下のスクリプトを確認してください。

Hide

```
# 線形回帰はlm関数で計算できます。
result <- lm(formula = 月給 ~ ., data = dat)
summary(result)
```

```
Call:
lm(formula = 月給 ~ ., data = dat)

Residuals:
   Min     1Q Median     3Q    Max 
-15413  -3884     -5   3004  88154 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    213082.90    9511.38  22.403  < 2e-16 ***
勤続年数         4761.56     271.51  17.538  < 2e-16 ***
仕事の達成度       93.07      35.25   2.640  0.00969 ** 
欠勤日数          262.71     975.75   0.269  0.78833    
特殊免許の有無   3977.66    2150.76   1.849  0.06751 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10630 on 95 degrees of freedom
Multiple R-squared:  0.7746,	Adjusted R-squared:  0.7651 
F-statistic: 81.61 on 4 and 95 DF,  p-value: < 2.2e-16
```

偏回帰係数の推定値は Coefficients の Estimate に書かれています。 Intercept は切片という意味です。計算の結果、$\hat{\beta}_0 = 213083, \hat{\beta}_1 = 4762, \hat{\beta}_2 = 93, \hat{\beta}_3 = 263, \hat{\beta}_4 = 3978$ だとわかります。具体的に1次式で表すと、

$$\text{月給} = 213083 + 4762 \times \text{勤続年数} + 93 \times \text{仕事の達成度} + 263 \times \text{欠勤日数} + 3978 \times \text{特殊免許の有無} +$$

が得られたことになります。

問題 : 以下の空欄（ア）,（イ）にあてはまる値を答えてください。

- 勤続年数 ・ 仕事の達成度 ・ 特殊免許の有無 が同じ二人の社員について、 欠勤日数 が1日多い社員は、そうでない社員に比べて 月給 が（ア）円だけ高い傾向にあると推定される。
- 勤続年数 2年、 仕事の達成度 200、 欠勤日数 3日、 特殊免許 を持っていない人の 月給 は（イ）円と予測できる。

解答 : （ア）の値は 263、（イ）の値は

$$213083 + 4762 \times 2 + 93 \times 200 + 263 \times 3 + 3978 \times 0 = 241996$$

とわかります。■

とこで、欠勤日数の偏回帰係数の推定値 $\hat{\beta}_3$ は 263 と正の値になっていることに違和感を覚えた方はいるでしょうか。よく読むと、欠勤日数の多い人の方が月給が高い傾向にあるという計算結果になっています。これは直感と異なる結果でしょう。直感や事前知識と異なる結果に気づくことは、

- データのとり方に不備があったのではないか？
- 説明変数に不足があり、欠勤日数に相関のある別の変数の影響が $\hat{\beta}_3$ の値に現れているのではないか？

など、データを理解するきっかけになります。「そうか！欠勤日数が多いほど月給が多い傾向にあるのか！これは発見だ！」とモデルを過信せず、結果から読み取れることに納得いくまで、その結果が出る理由をよく考察することが、多変量解析では大切です。

C. 偏回帰係数の区間推定・統計的仮説検定

データ分析を担当しているSさんは、特殊免許を持っている人は持っていない人に比べて、月給を 3000円より多く貰う傾向にあるのではないかと考えていました。この仮説を検討するための統計的仮説検定を行ってみましょう。有意水準は 5% とします。

帰無仮説 $H_0 : w_d = w$ に対して、検定統計量を (Estimate-w)/Std.Error とします。帰無仮説が正しいとき、この検定統計量は自由度 $n - (D + 1)$ の t 分布に従います。この事実を用いて、偏回帰係数の統計的仮説検定や区間推定を行うことができます。

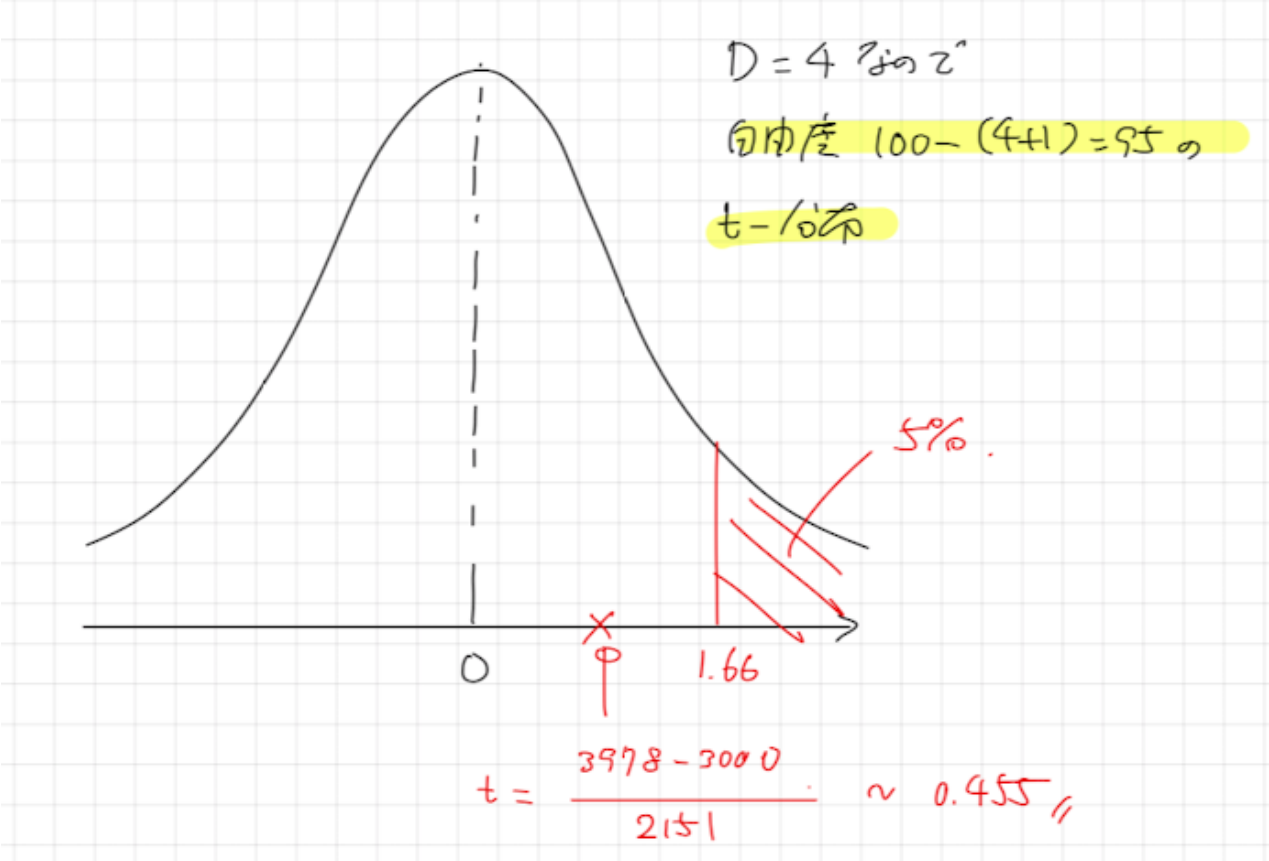
今回、帰無仮説と対立仮説をそれぞれ

$$H_0 : \beta_3 = 3000 \text{ v.s. } H_1 : \beta_3 > 3000$$

とします。このとき、検定統計量の値は

$$t = \frac{3978 - 3000}{2151} \sim 0.455$$

です。片側検定であることに注意すると、有意水準 5% のとき棄却域は$t \geq 1.66$ （R言語を用いて qt(p=0.95, df=100-5) と計算できます）です。



すなわち、帰無仮説は棄却されません。すなわち「特殊免許を持っている人は持っていない人に比べて、月給を3000円より多く貰う傾向にあるとは言えない」という結果になります。■

Remark : もし対立仮説が $H_1 : w_3 \neq 3000$ の場合、両側検定であることに注意すると、棄却域は $|t| \geq 1.985$ になります。

第2回：線形回帰のモデル選択・回帰診断・外挿

Code

1. 前準備

今回は、線形回帰にまつわる話題から

- モデル選択（変数選択）
- 残差分析
- 外挿

を解説していきます。デモデータには前回同様 salary.csv 、ある会社の社員に関する
月給 、 勤続年数 、 仕事の達成度 、 欠勤日数 、 特殊免許の有無 の情報が記録されたデータを用います。

社員の 月給 の傾向を 勤続年数 、 仕事の達成度 、 欠勤日数 、 特殊免許の有無 の4変数で説明する線形回帰は

$$\text{月給} = \beta_0 + \beta_1 \times \text{勤続年数} + \beta_2 \times \text{仕事の達成度} + \beta_3 \times \text{欠勤日数} + \beta_4 \times \text{特殊免許の有無} + \text{誤差}$$

と表すことができ、偏回帰係数 $\beta_0, \beta_1, \cdots, \beta_4$ の推定値はR言語の lm 関数を用いて次のように推定することができるのでした。（復習されたい方は、第1回の資料をご覧ください。）

Hide

```
# 線形回帰の計算結果
dat <- read.csv(file = "./data/salary.csv",
                fileEncoding = "utf-8")
result <- lm(formula = 月給 ~ ., data = dat)
summary(result)
```

```
Call:
lm(formula = 月給 ~ ., data = dat)

Residuals:
   Min     1Q Median     3Q    Max 
-15413  -3884     -5   3004  88154 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    213082.90    9511.38  22.403  < 2e-16 ***
勤務年数         4761.56     271.51  17.538  < 2e-16 ***
仕事の達成度       93.07      35.25   2.640  0.00969 ** 
欠勤日数          262.71     975.75   0.269  0.78833    
特殊免許の有無   3977.66    2150.76   1.849  0.06751 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10630 on 95 degrees of freedom
Multiple R-squared:  0.7746,	Adjusted R-squared:  0.7651 
F-statistic: 81.61 on 4 and 95 DF,  p-value: < 2.2e-16
```

2. 線形回帰のモデル選択

変数 $x_1, \cdots, x_D$ を用いて変数 y を説明するような線形回帰

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_D x_D + \text{誤差}$$

を作るとき、説明変数 $x_1, \cdots, x_D$ の組み合わせは事前知識によって入念に検討されて決まることが一般的です。しかし実際には、変数 x_d を説明変数に含めた方が良いかいなか悩ましい変数もあるでしょう。そこで、何らかの基準に基づいて変数を選択する方法がいくつか提案されてきました。

- 偏回帰係数の統計的仮説検定
- 自由度調整済み決定係数
- 赤池情報量規準（AIC）

これらの手法を、以下の疑問に答える形で解説します。

Remark : 他にもMallows' Cp規準やベイズ情報量規準、正則化を用いる方法などがよく知られています。

問題 : 変数 `欠勤日数` を説明変数に含めるかいなか、上記の3通りの方法で検討してみてください。

A. 偏回帰係数の統計的仮説検定

`欠勤日数` を説明変数に含めるかいなかを検討する方法の一つに、 `欠勤日数` の偏回帰係数が 0 かいなかを検討するという考え方があります。これは、偏回帰係数の統計的仮設検定によって実現できます。

解答 : 前回紹介した偏回帰係数の統計的仮説検定を、モデル選択に用いてみましょう。ここで考える帰無仮説と対立仮説は、

$$H_0 : \beta_3 = 0 \text{ v.s. } H_1 : \beta_3 \neq 0$$

です。このとき、検定統計量の値（t値）は

$$t = \frac{263 - 0}{976} \sim 0.269$$

です。

Hide

```
# Rでt値を計算してみよう。
t_value <- 263 / 976
t_value
```

```
[1] 0.2694672
```

両側検定であることに注意すると、有意水準 5% のとき棄却域は$|t| \geq 1.99$ （R言語を用いて `qt(p=0.975, df=100-(4+1))` と計算できます）です。

Hide

```
# Rで両側検定の棄却限界を計算してみよう。
critical <- qt(p = 0.975, df = 100-(4+1))
critical
```

```
[1] 1.985251
```

すなわち、帰無仮説は棄却されません。すなわち「欠勤日数は説明変数として含めなくて良いだろう」という結果になります。なおこの結果は、 `summary` 関数の出力 `t-value` , `Pr(>|t|)` によって確認することもできます。

B. 自由度調整済み決定係数

欠勤日数 を説明変数に含めるかいなかを検討するもう一つの方法に、目的変数を精度よく予測する式が作れるかいなかという考え方があります。偏回帰係数を推定して得られる式の予測の精度を測る指標に、自由度調整済み決定係数があります。

自由度調整済み決定係数（adjusted R-squared）は、目的変数に対する予測の精度を、線形回帰によるものと標本平均によるものとで比較したものです。$\hat{y}_i$ を標本点 i の目的変数の値 y_i に対する線形回帰の予測値、$\bar{y}$ を目的変数の標本平均とします。このとき、自由度調整済み決定係数は次のように定義されます。

$$R^{*2} = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2/\{n-(D+1)\}}{\sum_{i=1}^{n}(\bar{y} - y_i)^2/(n-1)}$$

自由度調整済み決定係数は、1 以下の値をとります。なお、負の数になることもあります。この値が 1 に近い線形回帰のほうが予測の精度が高いと推定できます。

Remark : 自由度調整済み決定係数の分子は線形回帰モデルが正しいと仮定したときの誤差の分散の不偏推定量、分母はなんら説明変数を用いなかったときの誤差の分散の不偏推定量になっています。このため、自由度調整済み決定係数は決定係数に比べて、母集団の決定係数を偏りなく推定することができます。つまり、決定係数は偏回帰係数の推定に用いたデータへのあてはまりの良さ、自由度調整済み決定係数は未知のデータへのあてはまりの良さの指標だと解釈できます。

解答 : 自由度調整済み決定係数は、 summary 関数の Adjusted R-squared に書かれています。今回、変数 欠勤日数 を含む線形回帰の自由度調整済み決定係数は 0.7651 です。一方、 欠勤日数 を含まない自由度調整済み決定係数は、以下のスクリプトから 0.7674 とわかります。

Hide

```
result2 <- lm(formula = 月給 ~ 勤務年数+仕事の達成度+特殊免許の有無,
              data = dat)
summary(result2)
```

```

Call:
lm(formula = 月給 ~ 勤務年数 + 仕事の達成度 + 特殊免許の有無,
    data = dat)

Residuals:
   Min     1Q Median     3Q    Max 
-15764  -4305    115   2984  88058 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)    214630.76    7540.68  28.463  < 2e-16 ***
勤務年数         4744.69     262.89  18.048  < 2e-16 ***
仕事の達成度       92.59      35.04   2.643  0.00961 ** 
特殊免許の有無   3936.54    2134.94   1.844  0.06829 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10570 on 96 degrees of freedom
Multiple R-squared:  0.7744,	Adjusted R-squared:  0.7674 
F-statistic: 109.9 on 3 and 96 DF,  p-value: < 2.2e-16
```

変数 `欠勤日数` を含む線形回帰より、含まない線形回帰のほうが自由度調整済み決定係数が高く、 `欠勤日数` を含まない場合の線形回帰のほうが未知のデータに対する予測の精度が高いと推定されます。すなわち「欠勤日数は説明変数として含めなくて良いだろう」という結果になります。

C. 赤池情報量規準

未知のデータへの予測の精度を推定する方法には、自由度調整済み決定係数の他にも**赤池情報量規準**（Akaike Information Criterion, AIC）を使う方法があります。赤池情報量規準は、次のように定義されます。

$$AIC = n \log \sum_{i=1}^{n} (\hat{y_i} - y_i)^2 + 2(D+1)$$

この値が小さい線形回帰のほうが予測の精度が高いと推定できます。

Remark : 赤池情報量規準は、Kullback-Leibler divergenceとよばれる母集団と推定によって得た線形回帰の間の近さを測る指標の（漸近）不偏推定量になっています。

解答 : 赤池情報量規準は、 `AIC` 関数を用いて計算することができます。

Hide

```
# 欠勤日数を含む場合のAICと、含まない場合のAIC
AIC(result); AIC(result2)
```

```
[1] 2144.875
[1] 2142.952
```

この結果、欠勤日数を含む場合の赤池情報量規準の値として 2144.875、含まない場合の赤池情報量規準の値として 2142.952 が得られ、 `欠勤日数` を含まない場合の線形回帰のほうが未知のデータに対する予測の精度が高いと推定されます。すなわち「欠勤日数は説明変数として含めなくて良いだろう」という結果になります。

3. 回帰診断

線形回帰モデルの偏回帰係数を最小2乗法で推定することは、以下の場合に「適切な」推定になっていることが数理統計学で確認されてきました。

- 誤差 ϵ の分散が説明変数によらず一定な場合。
- 誤差が正規分布に従っていると仮定する。

深くは立ち入りませんが、前者の場合はGauss-Markovの定理とよばれるもの、後者の場合は最尤法と最小2乗法が等しくなるというものです。（詳しくは数理統計学の講座を受講されてみてください。）

また、最小2乗法は残差の「二乗和」を小さくするように偏回帰係数を求めるため、**外れ値**に敏感であることが知られています。そこで、線形回帰を計算した後は

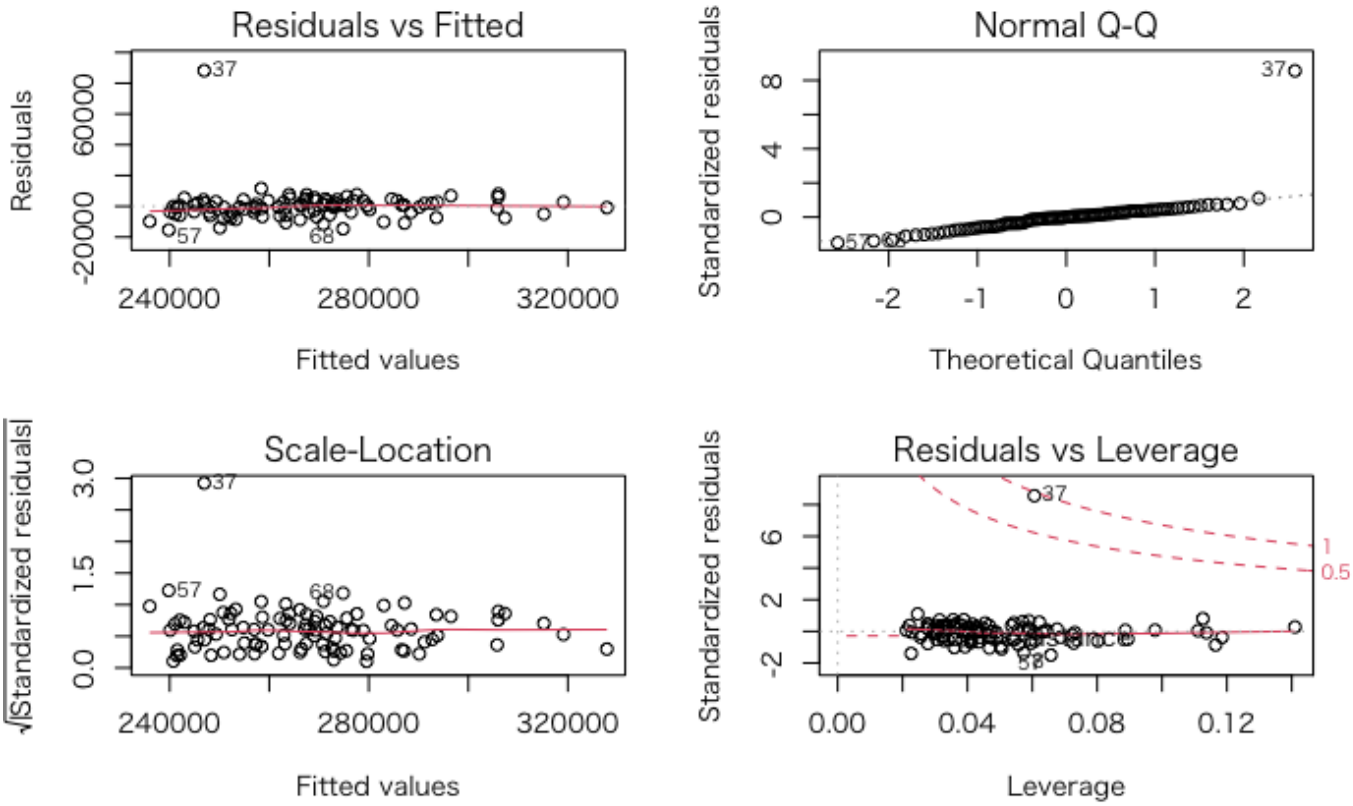
- 誤差が説明変数の値によらず一定か？
- 誤差が正規分布に従っているか？
- 外れ値がないか？

を確認し、偏回帰係数の推定の適切さを見積もったり、外れ値と考えた標本点を除去して偏回帰係数の推定値が適切な値になるかを実際にやってみて検討したりします。これを**回帰診断**（regression diagnostics）といいます。

残差分析はR言語の場合、 `plot` 関数に線形回帰の計算結果を渡すことで実行できます。

Hide

```
# 残差分析
par(family = "ヒラギノ角ゴシック W3")
par(mfrow = c(2, 2))
plot(result)
```



問題 : 上の図を参考に、以下の事項について検討してください。

1. 除去するべき外れ値はあるか？
2. 誤差は説明変数によらず一定か？
3. 誤差は正規分布に従っているとみなせるか？

解答 : 上の図には、 Fitted values , Residual , Standardized residual , Theoretical Quantities , Leverage , Cook's distance という6つの単語が出てくるため、これを解説します。

- Fitted values ：目的変数の予測値のこと。
- Residuals ：目的変数の実測値と予測値の差のこと。
- Standardized residuals ：各標本点における誤差 ϵ_i のz得点の推定値のこと。
- Theoretical Quantities ： Standardized residuals を小さい順に並べたとき、対応する標準正規分布の分位点の値のこと。
- Leverage ：その標本点の目的変数の実現値を変化させたとき、予測値がどれだけ変化するか。数式で表すと $\partial \hat{y}_i / \partial y_i$ のこと。日本語では「てこ比」という。
- Cook's distance ：次のように定義される外れ値の評価指標のひとつ。 Standardized Residual が大きいほど、また Leverage が大きいほど外れ値だと考えることができるため、このような式になっている。

$$\text{Cook's distance} = \text{Standardized Residual} * \frac{\text{Leverage}}{1 - \text{Leverage}}$$

1. 除去するべき外れ値はあるか？: 4番目の図から、37行目の標本点のCook距離が大きく、外れ値であろうと推測できます。また、1番目の図と3番目の図からも、37行目の標本点だけが大きな誤差の実現値を持っていると推定できる。ゆえに、37行目の標本点が外れ値であろうと考えられる。

2. 誤差は説明変数によらず一定か？：1番目の図と3番目の図はいずれも、誤差の推定値と予測値の間の関係を表したものです。いずれも、予測値によらず誤差の値は同じ範囲を散布していることから、誤差は説明変数によらず一定であろうと推測できます。
3. 誤差は正規分布に従っているとみなせるか？：2番目の図は誤差のz得点の推定値と対応する標準正規分布の推定値を比較したものです。37行目を除いて、直線 $y = x$ に点が並んでいることからから、誤差は正規分布に従っていると仮定して良いだろうことがわかります。

以上で解答は終わりです。■

なお、今回の問題で外れ値と考えた37行目を除去して改めて線形回帰を行うと、次のような結果になります。回帰診断の結果、外れ値は十分に除去できており、また偏回帰係数の推定も適切に行えているだろうことが推測できます。

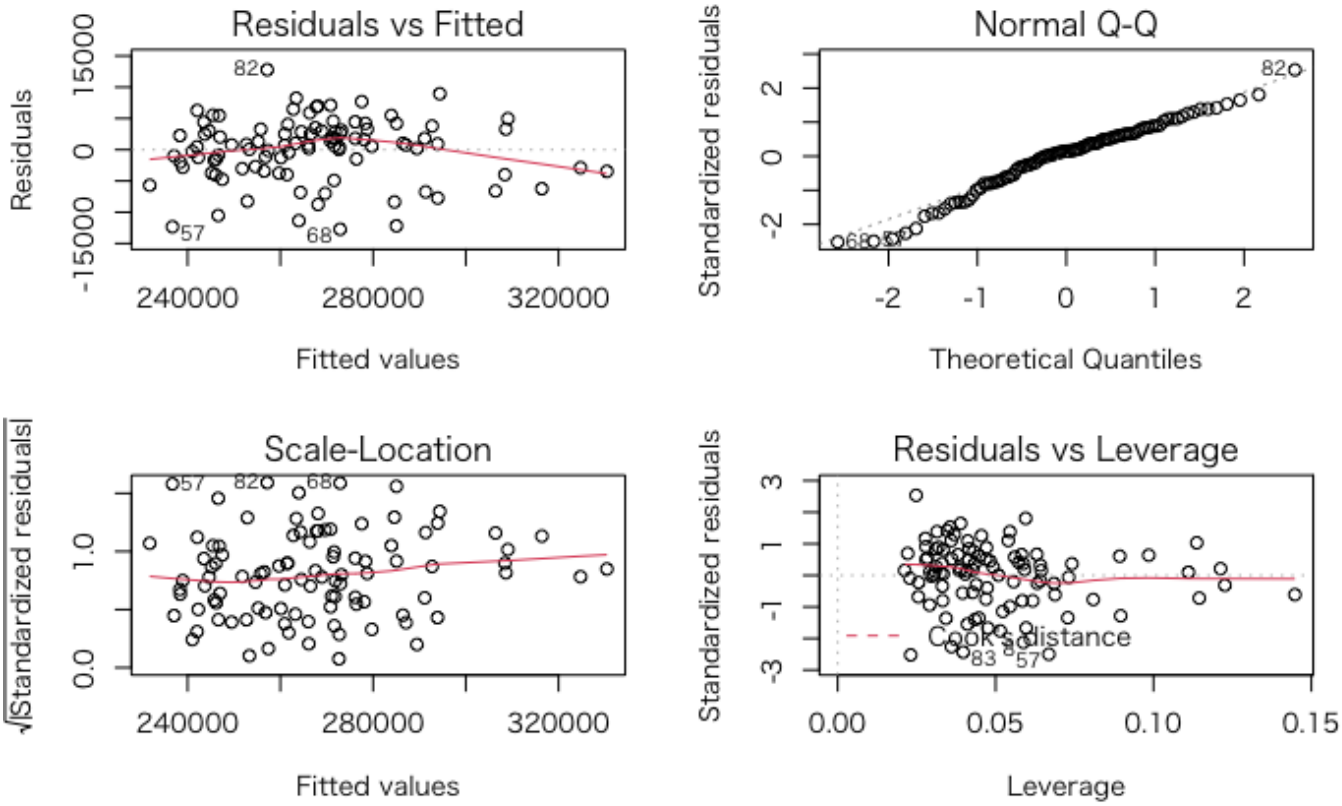
Hide

```
result3 <- lm(formula = 月給 ~ .,
              data = dat[-37, ])
summary(result3)
```

```

Call:
lm(formula = 月給 ~ ., data = dat[-37, ])

Residuals:
     Min       1Q   Median       3Q      Max 
-12731.8  -2960.8    739.8   3174.7  12787.0 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)     196501.81    4667.92  42.096  < 2e-16 ***
勤務年数          4994.37     131.22  38.061  < 2e-16 ***
仕事の達成度       151.81      17.27   8.790 6.88e-14 ***
欠勤日数           551.78     469.52   1.175    0.243    
特殊免許の有無    5399.52    1037.38   5.205 1.14e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5110 on 94 degrees of freedom
Multiple R-squared:  0.9429,	Adjusted R-squared:  0.9405 
F-statistic: 388.3 on 4 and 94 DF,  p-value: < 2.2e-16
```

Hide

```
# 残差分析
par(family = "ヒラギノ角ゴシック W3")
par(mfrow = c(2, 2))
plot(result3)
```



4. 外挿

早速ですが、以下の問題を考えてみましょう。

問題 :

1. 勤続年数・仕事の達成度・特殊免許の有無の3変数を用い、また37行目を除去したデータを用いて、月給を説明する線形回帰を計算してください。
2. 1の結果を用いて、勤続年数が40年、仕事の達成度が200、特殊免許を持っている社員の月給を予測してください。
3. 2の結果をみた分析者は、一般にこのような高い月給が発生することはなく、予測が妥当ではないと考えました。ではなぜ、このような現実的ではない予測が得られたのか、その理由を考えてください。

解答 : 線形回帰は以下のように計算できます。

Hide

```
result4 <- lm(formula = 月給 ~ 勤務年数+仕事の達成度+特殊免許の有無,
              data = dat[-37, ])
summary(result4)
```

```
Call:
lm(formula = 月給 ~ 勤務年数 + 仕事の達成度 + 特殊免許の有無,
    data = dat[-37, ])

Residuals:
     Min       1Q   Median       3Q      Max 
-13074.5  -3025.8    598.1   3333.9  12692.1 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)     199786.79    3745.98  53.334  < 2e-16 ***
勤務年数          4958.44     127.86  38.779  < 2e-16 ***
仕事の達成度       150.65      17.28   8.719 9.00e-14 ***
特殊免許の有無    5310.01    1036.65   5.122 1.58e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5120 on 95 degrees of freedom
Multiple R-squared:  0.9421,	Adjusted R-squared:  0.9403 
F-statistic: 515.2 on 3 and 95 DF,  p-value: < 2.2e-16
```

この結果、勤続年数が40年、仕事の達成度が200、特殊免許を持っている社員の月給は次のように予測できます。

$$199787 + 4958 \times 40 + 151 \times 200 + 5310 \times 1 = 433617$$

さて、分析者はこの予測について「一般にこのような高い月給が発生することはなく、予測が妥当ではない」と考察しています。このような現実的ではない予測が生じた理由は、データに含まれる説明変数の値の範囲を確認するとわかります。

Hide

```
# 説明変数の値の範囲
summary(dat)
```

```
      月給            勤務年数       仕事の達成度       欠勤日数 特殊免許の有無
 Min.   :224399   Min.   : 0.00   Min.   :108.0   Min.   :3   Min.   :0.00  
 1st Qu.:251698   1st Qu.: 4.00   1st Qu.:182.8   1st Qu.:4   1st Qu.:0.00  
 Median :267124   Median : 6.50   Median :207.5   Median :5   Median :0.00  
 Mean   :267768   Mean   : 6.84   Mean   :203.0   Mean   :5   Mean   :0.48  
 3rd Qu.:278390   3rd Qu.: 9.00   3rd Qu.:224.2   3rd Qu.:6   3rd Qu.:1.00  
 Max.   :335092   Max.   :19.00   Max.   :263.0   Max.   :8   Max.   :1.00  
```

勤務年数は最大で19年の社員しかデータには含まれていません。つまり今回、月給が線形回帰で予測できることを確かめることができたのは、勤務年数が19年までの社員です。実際、勤続年数が長くなるに従って、昇給額は小さくなり線形回帰があてはまらなくなる可能性があります。今回もこのような理由で、勤続年数が40年の社員には線形回帰の予測値が妥当な額にならない現象が発生したと考えられます。■

このように、線形回帰の計算に用いたデータに含まれる説明変数の値の範囲を超えた値をもつような標本点に対して、得られた線形回帰による予測を行うことを**外挿**（extrapolation）といいます。

第3回：非線形回帰

Code

1. 前準備

これまで解説した線形回帰は、目的変数を説明変数の1次式で表すような統計モデルのことでした。しかし、一般には目的変数と説明変数の間の関係が1次式でない（非線形である）ような場合も考えられます。このような場合に用いられる統計モデルが非線形回帰（nonlinear regression）です。今回は、非線形回帰にまつわる話題から、

- 非線形回帰の概要
- 多項式回帰
- べき乗回帰
- モデル選択

を解説します。

デモデータには、 R 言語にデフォルトで読み込まれている mtcars データセットを用います。このデータセットは、自動車の馬力と燃費の関係を記録したものです。

Hide

```
# mtcarsデータセットの確認
dat <- mtcars[, c("hp", "mpg")]
head(x = dat, n = 5)
```

	hp <dbl>	mpg <dbl>
Mazda RX4	110	21.0
Mazda RX4 Wag	110	21.0
Datsun 710	93	22.8
Hornet 4 Drive	110	21.4
Hornet Sportabout	175	18.7

5 rows

標本サイズと次元、変数の名前を確認しておきましょう。

Hide

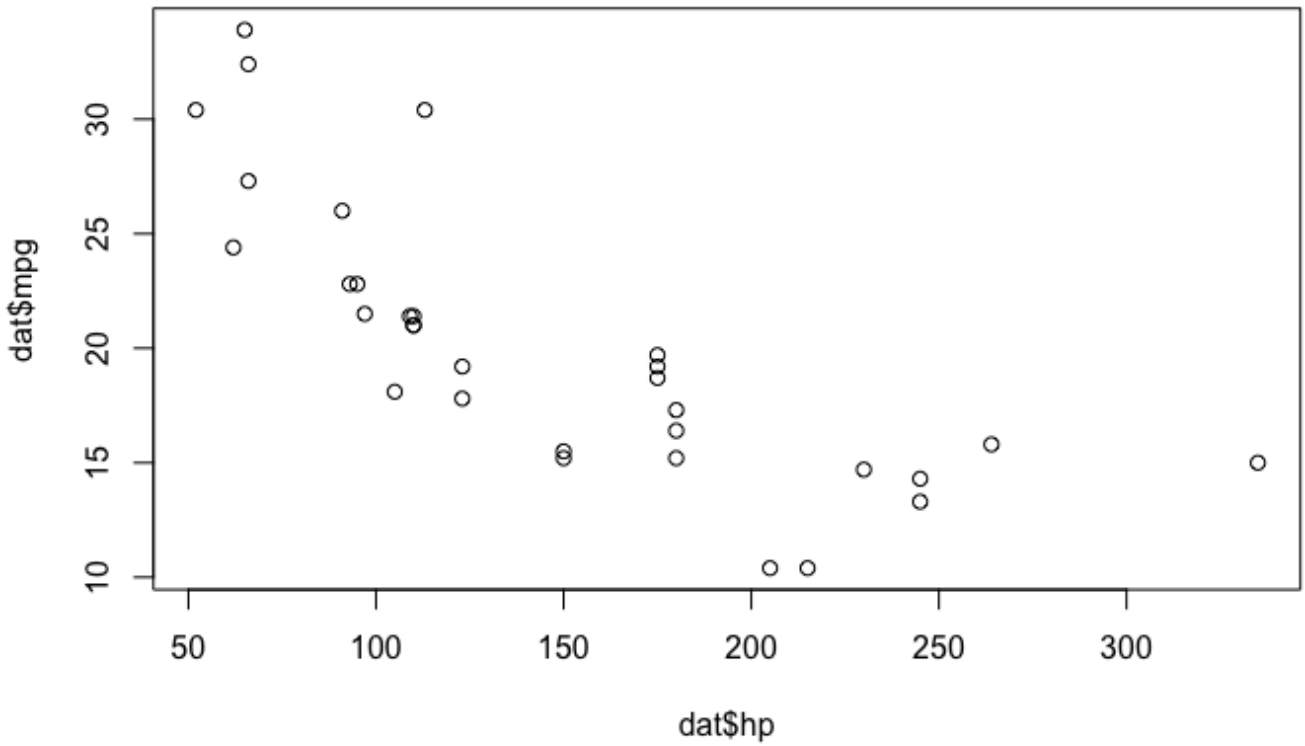
```
# 標本サイズ、次元、変数の名前の確認
str(dat)
```

```
'data.frame':	32 obs. of  2 variables:
 $ hp : num  110 110 93 110 175 105 245 62 95 123 ...
 $ mpg: num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
```

今回は、馬力 hp を用いて燃費 mpg を説明するモデルを作ろうと思います。2変数の関係を把握するために散布図をかいておくことで、 馬力 と 燃費 との間は非線形な関係にあることが確認できます。

Hide

```
# 散布図
plot(dat$hp, dat$mpg)
```



2. 非線形回帰

この節では、非線形回帰の概要を説明します。また、非線形回帰の具体例として、

- 多項式回帰
- べき乗回帰

を紹介します。

2.1 非線形回帰の概要

A. 課題設定

データに D 個の変数 $x_1, \cdots, x_D$ と y が記録されているとき、変数 y を変数 $x_1, \cdots, x_D$ の1次式ではない（非線形な）関数

$$y = f(x_1, \cdots, x_D; \beta) + \text{誤差}$$

で表現することで説明・予測できると仮定します。例えば、燃費を馬力で説明する式として

$$\text{燃費} = \beta_0 + \beta_1 \times \text{馬力} + \beta_2 \times \text{馬力}^2 + +\text{誤差}$$

という2次式や、

$$\text{燃費} = \beta_1 \times \text{馬力}^{\beta_2} + \text{誤差}$$

という式で説明できないかを検討していきましょう。前者のように説明変数の多項式で目的変数のとりうる値の傾向を表現するようなモデルを**多項式回帰**（polynomial regression）、後者のように説明変数の累乗とその定数倍で目的変数のとりうる値の傾向を表現するようなモデルを**べき乗回帰**（power regression）といいます。

B. 出来ること

モデルに含まれる値が未知のパラメータ $\beta_1, \cdots, \beta_p$ を**偏回帰係数**（coefficient）といいます。これらの係数を推定することで、各説明変数の値が決まっているとき、目的変数の値はいくらと予測できるかを推定することができます。また、偏回帰係数の区間推定や統計的仮説検定を行うことができます。これについてはデモで例を交えながら解説します。

C. 偏回帰係数の推定の仕組み

偏回帰係数の推定値 $\hat{\beta}$ は、データ全体で

- 偏回帰係数から決まる各標本点の予測値 $\hat{y} = f(x_1, \cdots, x_D; \hat{\beta})$
- 実測値 y

の差（**残差**といいます）の2乗 $(y - \hat{y})^2$ の平均値

$$l(\beta_0, \beta_1, \cdots, \beta_D) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

が最小になるように決めます。これを**最小2乗法**（least squares method）といいます。非線形回帰とは、目的変数と説明変数の関係に偏回帰係数 β で特徴付けられる関数 $f(x_1, \cdots, x_D; \beta)$ を仮定したとき、これを最小2乗法で推定する手法のことです。

2.2 多項式回帰のデモ

燃費 mpg を馬力 hp の2次式で表現するモデル、つまり

$$\text{燃費} = \beta_0 + \beta_1 \times \text{馬力} + \beta_2 \times \text{馬力}^2 + \text{誤差}$$

を考えます。多項式回帰では、説明変数の値をそのまま用いると、大きい次数の項の値がとても大きくなったり小さくなったりすることで、計算機が数値的に不安定になることがあります。そのため、説明変数を事前に正規化（例えば標準化やmin-max正規化）することが一般的です。今回は、min-max正規化を採用します。

Hide

```
# `hp`のmin-max正規化を`hp_scale`という名前で作る。
min_hp <- min(dat$hp)
max_hp <- max(dat$hp)
dat$hp_scale <- (dat$hp-min_hp) / (max_hp-min_hp)
head(x = dat, n = 5)
```

	hp <dbl>	mpg <dbl>	hp_scale <dbl>
Mazda RX4	110	21.0	0.2049470
Mazda RX4 Wag	110	21.0	0.2049470
Datsun 710	93	22.8	0.1448763
Hornet 4 Drive	110	21.4	0.2049470

	hp <dbl>	mpg <dbl>	hp_scale <dbl>
Hornet Sportabout	175	18.7	0.4346290

5 rows

多項式回帰の場合、 R 言語では次のように偏回帰係数を計算することができます。

Hide

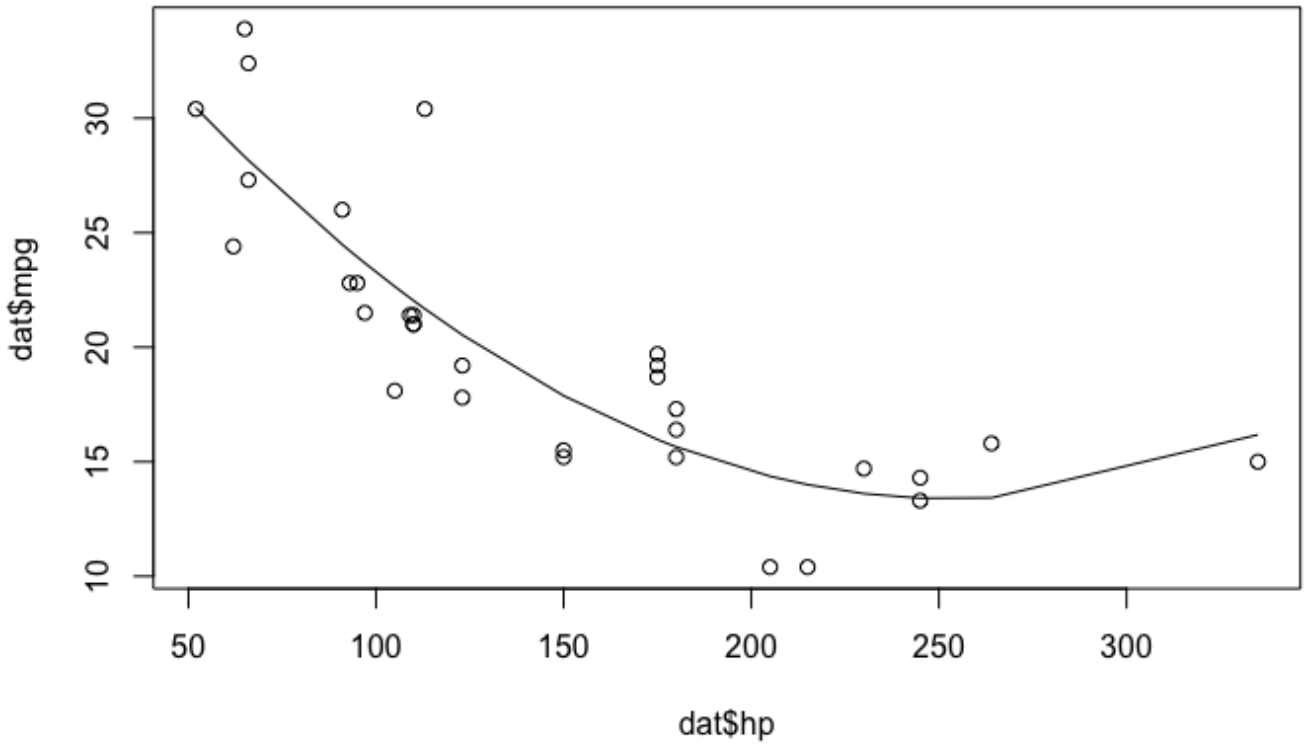
```
# 多項式回帰
result_deg2 <- lm(mpg ~ poly(hp_scale, degree = 2, raw = TRUE),
                  data = dat)
summary(result_deg2)
```

```

Call:
lm(formula = mpg ~ poly(hp_scale, degree = 2, raw = TRUE), data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.5512 -1.6027 -0.6977  1.5509  8.7213 

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)                           30.455      1.309  23.271  < 2e-16 ***
poly(hp_scale, degree = 2, raw = TRUE)1  -47.981      7.085  -6.772 1.96e-07 ***
poly(hp_scale, degree = 2, raw = TRUE)2   33.703      7.884   4.275 0.000189 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.077 on 29 degrees of freedom
Multiple R-squared:  0.7561,	Adjusted R-squared:  0.7393 
F-statistic: 44.95 on 2 and 29 DF,  p-value: 1.301e-09
```

また、得られた2次関数を散布図上に重ねてかくと、次のようになります。

Hide

```
plot(dat$hp, dat$mpg)
lines(sort(dat$hp), predict(result_deg2)[order(dat$hp)])
```



問題 : 以下の問いに答えてください。

1. 推定によって得られた2次式を答えてください。
2. 推定された結果に不自然な点があるかを検討してください。

解答 :

1. 得られた3次式は以下の通りです。なお、式の中に現れる「馬力」は、min-max正規化を施した後の変数とします。

$$\text{燃費} = -30.45 + 2.38 \times \text{馬力} + 0.08 \times \text{馬力}^2 + \text{誤差}$$

2. 馬力が大きいほど燃費は必ず悪くなるとなると想定されます。しかしグラフからは、馬力が 250 より大きくなると燃費が悪くなるという予測を与える式になっていることが確認できます。これが、このモデルの不自然な点です。■

2.3 べき乗回帰のデモ

燃費 mpg を馬力 hp の3次式で表現するモデル、つまり

$$\text{燃費} = \beta_1 \times \text{馬力}^{\beta_2} + \text{誤差}$$

を考えます。べき乗回帰では、偏回帰係数を推定するために、事前に初期値を決めておく必要があります。べき乗回帰における最小2乗法は**非線形最小2乗法**といって、予め決めた偏回帰係数の初期値から、より残差の2乗和が小さくなるような偏回帰係数の値へと更新していくような計算方法で、偏回帰係数の値を求めています。

この初期値を求める方法の一つとして、次のような線形回帰を先に計算しておきます。

Hide

```
# 偏回帰係数の初期値の決定
lm(log(mpg) ~ log(hp), data = dat)
```

```
Call:
lm(formula = log(mpg) ~ log(hp), data = dat)

Coefficients:
(Intercept)      log(hp)
     5.5454      -0.5301
```

問題 : 以下の問いに答えてください。

1. べき乗回帰の式 $\text{燃費} = \beta_1 \times \text{馬力}^{\beta_2} + \text{誤差}$ の「誤差」を無視したとき、両辺の対数をとって得られる式を答えてください。
2. 上の線形回帰の結果から、偏回帰係数の初期値をどう決めると良いかを考えてみてください。

解答 :

1. $\log \text{燃費} = \log \beta_1 + \beta_2 \log \text{馬力}$
2. Intercept は5.5454、 log(hp) の偏回帰係数は-0.5301でした。(1)の結果から $\log \beta_1 = 5.5454$ とおいたとき $\beta_1 = \exp(5.5454)$、$\beta_2 = -0.5301$ と決めることができます。■

Remark : なお目的変数を $\log$ 燃費、説明変数を $\log$ 馬力 とした線形回帰を式で表すと、
$\log \text{燃費} = \alpha_1 + \alpha_2 \log \text{馬力} + \text{誤差}$ になります。指数をとると

$$\begin{aligned} \text{燃費} &= \exp \alpha_1 \times \text{馬力}^{\alpha_2} \times \exp(\text{誤差}) \\ &= \beta_1 \times \text{馬力}^{\beta_2} \times \exp(\text{誤差}) \qquad \beta_1 = \exp \alpha_1, \beta_2 = \alpha_2 \text{とおく。} \end{aligned}$$

になるので、誤差のモデリングがべき乗回帰とは異なることが確認できます。つまり、 lm 関数で得られた推定結果はべき乗回帰そのものの結果ではなく、べき乗回帰は改めてやり直す必要があります。■

得られた初期値を用いて、べき乗回帰を計算してみましょう。 R 言語では nls 関数を用いて、べき乗回帰の偏回帰係数を計算することができます。

Hide

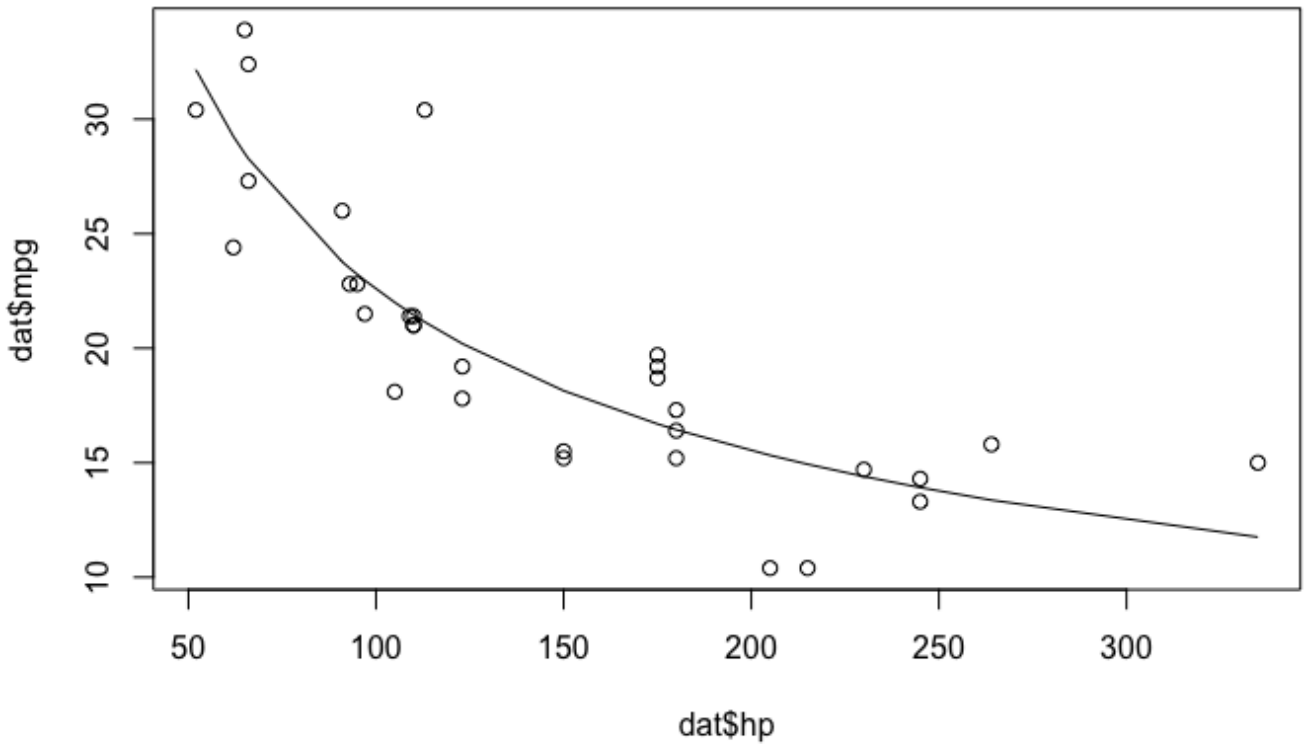
```
# 多項式回帰
result_power <- nls(mpg ~ a*hp^b,
                    data = dat,
                    start = list(a = exp(5.5454), b = -0.5301),
                    control = nls.control(maxiter = 100))
summary(result_power)
```

```
Formula: mpg ~ a * hp^b

Parameters:
   Estimate Std. Error t value Pr(>|t|)
a 272.11754   74.03692   3.675 0.000924 ***
b  -0.54043    0.05826  -9.277 2.55e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.102 on 30 degrees of freedom

Number of iterations to convergence: 2
Achieved convergence tolerance: 3.411e-06
```

また、得られた関数を散布図上に重ねてかくと、次のようになります。

Hide

```
plot(dat$hp, dat$mpg)
lines(sort(dat$hp), predict(result_power)[order(dat$hp)])
```



問題 : 推定によって得られた関数を答えてください。

解答 :

得られた関数は以下の通りです。

$$\text{燃費} = 272.12 \times \text{馬力}^{-0.54} + \text{誤差}$$

2.3 モデル選択

A. 赤池情報量規準

今回は、燃費 mpg を馬力 hp で表現する式を、多項式回帰とべき乗回帰の2通りで作りました。複数のモデルを作ったら、どちらのモデルを採用するかを検討しましょう。多項式回帰の節で出題した問題のように、各モデルの長所・短所を検討してモデルを選択することが大切です。しかし、場合によっては、それでもモデル選択に悩むことがあるでしょう。このようなとき、モデル選択の際にヒントになる指標の一つとして、未知のデータへの予測の精度を推定する方法だった赤池情報量規準を用いることができます。（第2回資料も参考にしてみてください。）

Hide

```
# 多項式回帰とべき乗回帰の比較
AIC(result_deg2); AIC(result_power)
```

```
[1] 167.6023
[1] 167.1958
```

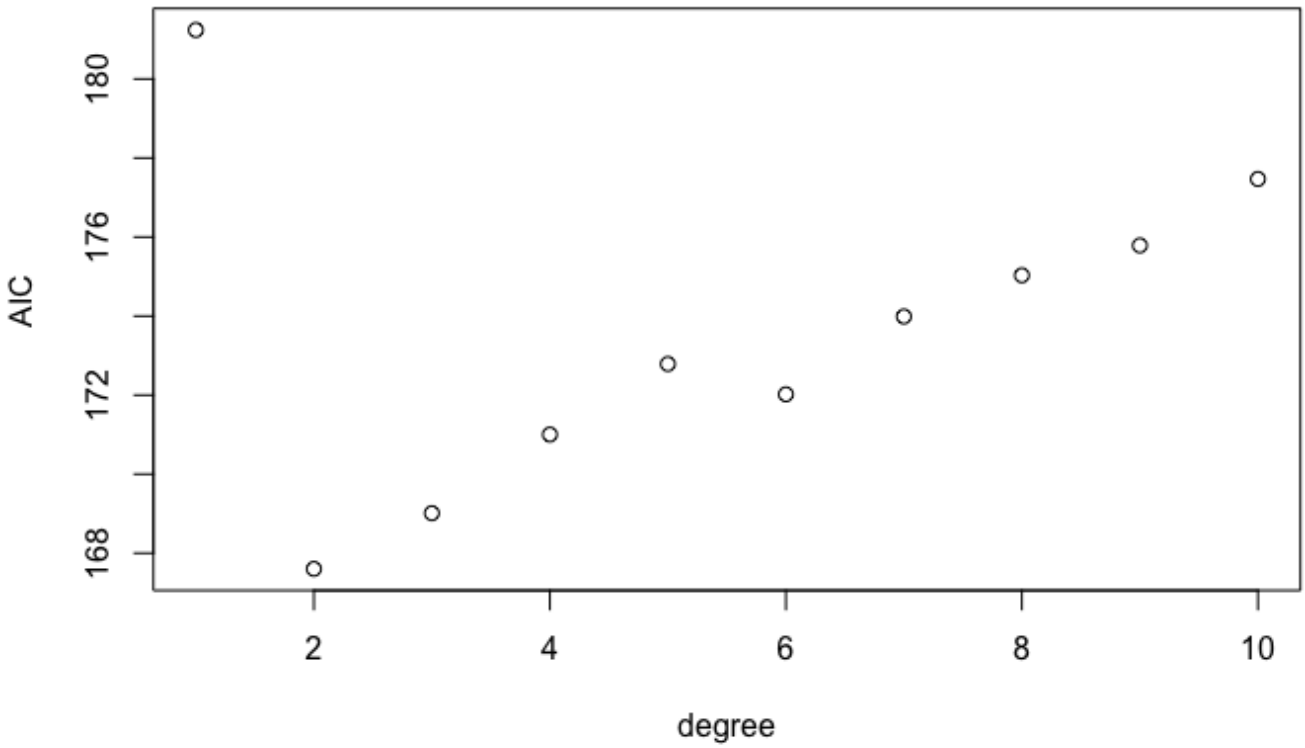
今回は、赤池情報量規準がより小さいべき乗回帰を採用するほうが良いのではないかと推定できます。

B. 多項式回帰の次数の決定と過剰適合

多項式回帰の次数を決定する際、以下のように赤池情報量規準を各次数で計算し図にまとめることがあります。例えば、以下の図からは次数が2の場合が最も赤池情報量規準が小さいことが確認できます。

Hide

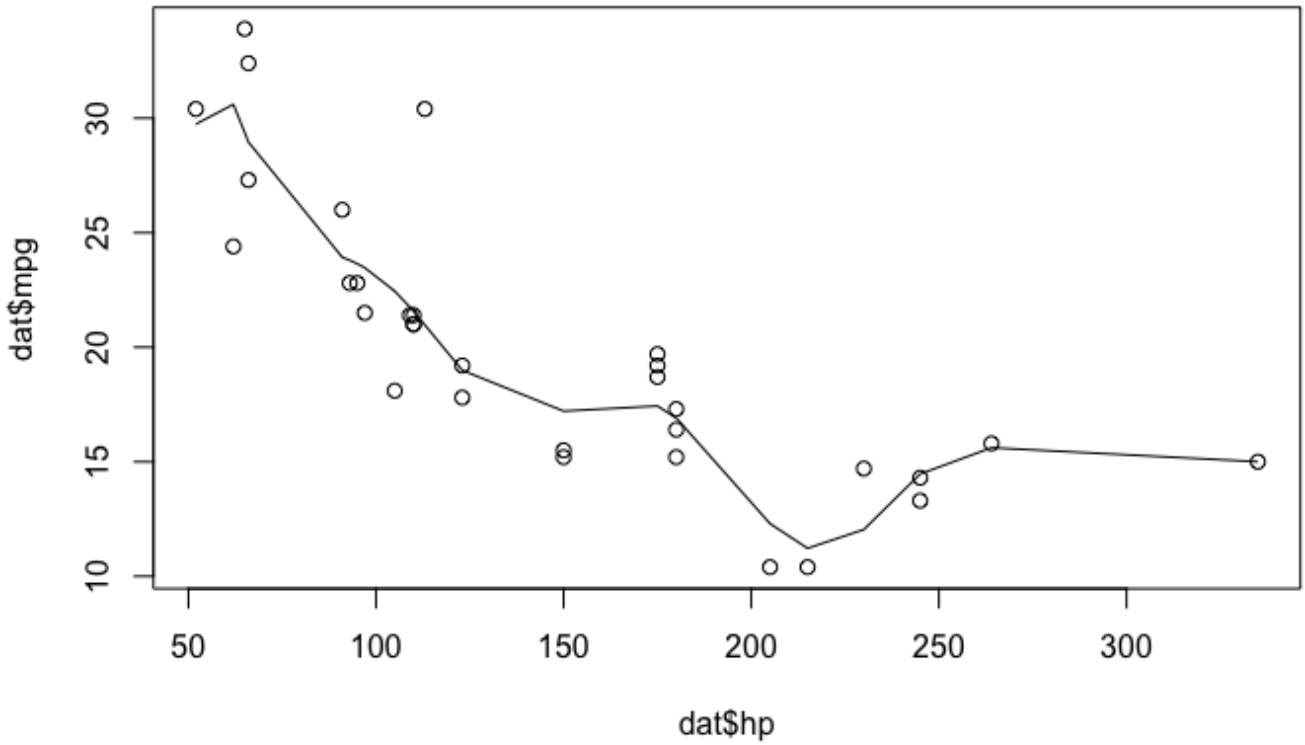
```
# 多項式回帰の次数の決定
aic <- c()
for (d in 1:10) {    # 次数1から10までの多項式回帰を逐次計算する。
  result_poly <- lm(mpg ~ poly(hp_scale, degree = d, raw = TRUE),
                    data = dat)
  aic[d] <- AIC(result_poly)
}

plot(1:10, aic, xlab = "degree", ylab = "AIC")
```



ところで、試しに次数10の多項式回帰を実行すると、以下のようにデータによくあてはまっているように見えます。それにも関わらず赤池情報量規準が次数2の場合より高い値になっているのはなぜでしょうか。

Hide

```
# 次数10の多項式回帰（10次回帰ともいいます。）
result_deg10 <- lm(mpg ~ poly(hp_scale, degree = 10, raw = TRUE),
                   data = dat)
plot(dat$hp, dat$mpg)
lines(sort(dat$hp), predict(result_deg10)[order(dat$hp)])
```



理由は、赤池情報量規準が「未知のデータへの予測の精度」を推定する方法だからです。手元のデータによくあてはまることと、まだ得られていない未知のデータによくあてはまることは異なります。前者が起こって後者が起こらないような現象を**過剰適合**（過学習, overfitting）、その逆を**過少適合**（未学習, underfitting）といいます。これらは特に、続・初級統計学で扱う後半の話題で、とても重要な話になっていきます。

第4回：ロジスティック回帰

Code

1. 前準備

これまで解説した線形回帰や非線形回帰は、目的変数が**量的**（numeric）の場合の統計モデルです。しかし、目的変数として**質的**（categorical）を考えたいこともあります。以下は、ある会社の訪問時刻・担当者年齢と契約結果の関係を記録したデータです。

Hide

```
# keiyakuデータセットの確認
dat <- read.csv("./data/keiyaku.csv",
                row.names = "会社",
                fileEncoding = "cp932")
head(dat, n = 5)
```

	訪問時刻 <chr>	担当者年齢 <int>	契約結果 <int>
A	午前	42	1
B	午前	22	0
C	午後	24	0
D	午前	36	0
E	午前	35	0

5 rows

このデータを用いて、訪問時刻・担当者年齢を用いて契約結果を説明するモデルを作ることを考えてみましょう。説明変数は訪問時刻と担当者年齢、目的変数は契約結果です。契約結果は「契約成立」の場合 1 、「契約不成立」の場合 0 が記録された2値の質的変数になっています。そこで今回は、このような2値の質的変数に対する統計モデルとして有名な**ロジスティック回帰**（logistic regression）を紹介します。

標本サイズと次元、変数の名前を確認しておきましょう。

Hide

```
# 標本サイズ、次元、変数の名前の確認
str(dat)
```

```
'data.frame':	22 obs. of  3 variables:
 $ 訪問時刻  : chr  "午前" "午前" "午後" "午前" ...
 $ 担当者年齢: int  42 22 24 36 35 42 36 39 36 42 ...
 $ 契約結果  : int  1 0 0 0 0 1 0 1 0 1 ...
```

2. ロジスティック回帰

この節では、ロジスティック回帰の概要を説明します。

2.1 ロジスティック回帰の概要

A. 課題設定

データに D 個の変数 $x_1, \cdots, x_D$ と y が記録されているとします。ここで変数 y は 0 か 1 のいずれかをとる2値の質的変数とします。変数 y が 1 になる確率を変数 $x_1, \cdots, x_D$ を用いて

$$\mathbb{P}[y = 1] = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_D x_D)}}$$

で表現することで説明・予測できると仮定します。例えば、訪問時刻・担当者年齢を用いて契約結果を説明するロジスティック回帰は

$$\mathbb{P}[\text{契約結果} = 1] = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \times \text{訪問時刻} + \beta_2 \times \text{担当者年齢})}}$$

という式を検討していることになります。

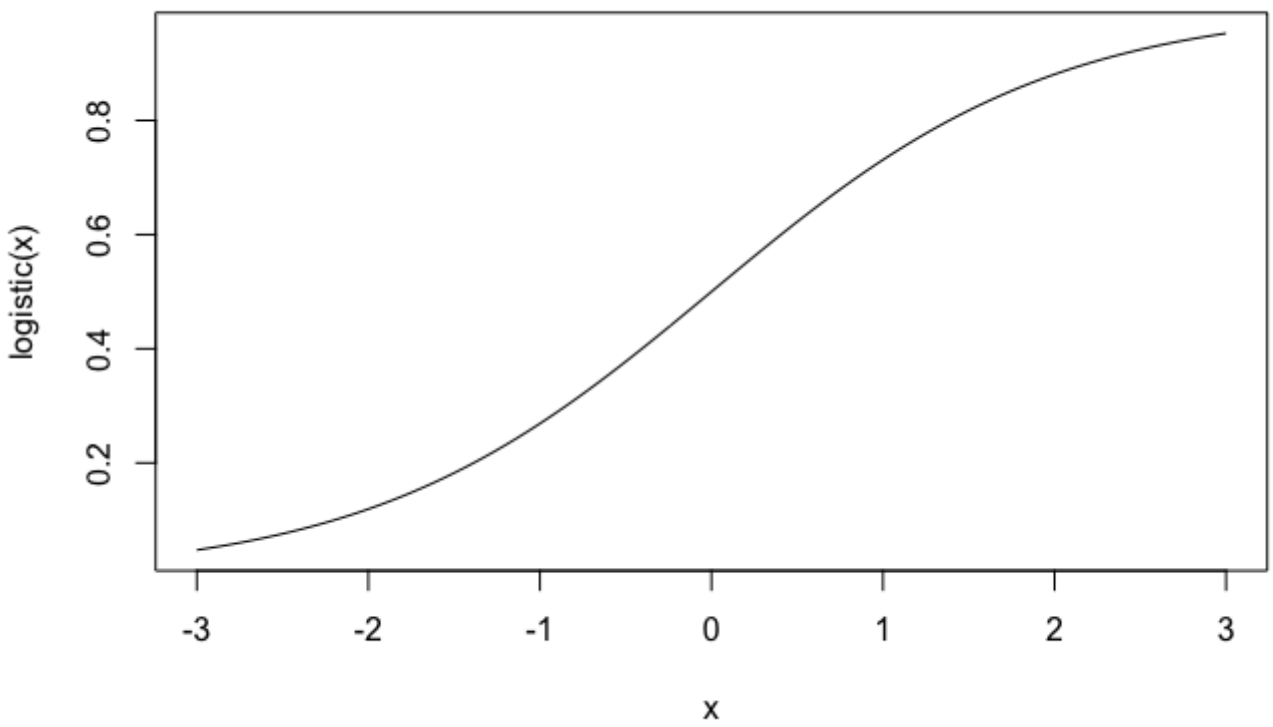
ここで、ロジスティック回帰を理解する上で役に立つ**ロジスティック関数**（logistic function）について紹介します。関数

$$f(x) = \frac{1}{1 + e^{-x}}$$

をロジスティック関数といいます。ロジスティック関数のグラフは次のようになります。

Hide

```
# ロジスティック関数
logistic <- function(x){return(1.0/(1.0+exp(-x)))}
curve(logistic, from = -3.0, to = 3.0)
```



ロジスティック関数の大きな特徴は、以下の3点です。

1. 0 より大きく 1 より小さい値をとる。
2. x に対して単調増加な関数である。
3. $f(0) = 0.5$

ロジスティック回帰のなかで特に大切になるのは、1番目と2番目の性質です。1番目の性質は、説明変数の1次式によって得られた値がロジスティック関数によって確率の値に変換できることを意味しています。2番目の性質は、偏回帰係数の値が正（resp. 負）の説明変数の値が大きくなればなるほど、確率が大きく（resp. 小さく）なるような統計モデルになっていることを意味しています。これを**線形性**といいます。

B. 出来ること

モデルに含まれる値が未知のパラメータ $\beta_1, \cdots, \beta_D$ を**偏回帰係数**（coefficient）といいます。これらの係数を推定することで、各説明変数の値が決まっているとき、目的変数の値はいくらと予測できるかを推定することができます。また、偏回帰係数の区間推定や統計的仮説検定を行うことができます。これについてはデモで例を交えながら解説します。

ここで、偏回帰係数の解釈の仕方に役立つ**オッズ**（odds）を紹介します。確率 p に対して、

$$\frac{p}{1-p}$$

をオッズといいます。

問題 : 以下の確率について、オッズを計算してください。

1. $p = 0.5$
2. $p = 0.4$
3. $p = 0.6$

解答 : (1) $p/(1-p) = 1$, (2) $p/(1-p) = 2/3$, (3) $p/(1-p) = 3/2$■

確率 $p = 0.4$ と確率 $p = 0.5$ との間でオッズは 3/2 倍、また確率 $p = 0.5$ と確率 $p = 0.6$ との間でもオッズは 3/2 倍になるところに注目すると、オッズという概念を受け入れやすいかもしれません。ところで、ロジスティック回帰は $y = 1$ の確率 p をオッズで表すとき

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_D x_D}$$

と仮定していることに対応しています。このことから、オッズを用いると説明変数 x_d の偏回帰係数 β_d は「説明変数 x_d の値のみが 1 増えたとき、$y = 1$ のオッズが e^{β_d} 倍増える」と解釈できます。

C. 偏回帰係数の推定の仕組み

偏回帰係数は、**最尤推定**（maximum likelihood estimation）という方法で推定されます。最尤法とは何か、考え方を単純な例で紹介しましょう。

問題 : 表が確率 p で出るコインを5回投げ、その結果から p の値を推定することを考えます。仮に、コインを投げて「表裏表表裏」という結果を得た場合、p の値はいくらと推定できるでしょうか。

解答 : 「表裏表表裏」という結果は確率 $p^3(1-p)^2$ で得られます。そこで、この確率が最も大きくなるような p の値を推定値として用いるのが最尤推定です。本来は**微分**によってわかることですが、今回は関数のグラフをかくことで $p = 3/5$ とわかります。

Hide

```
# 最尤推定
lik <- function(x){return(x^3*(1-x)^2)}
curve(lik, from = 0.0, to = 1.0)
```

ゆえに、p の最尤推定値は $\hat{p} = 3/5$ です。■

さて、契約結果を説明するロジスティック回帰の場合、次のような関数が最大になるような $\beta_0, \beta_1, \beta_2$ の値を偏回帰係数の推定値に用います。

$$\frac{1}{1+e^{-(\beta_0+\beta_1\times 0+\beta_2\times 42)}}\times\frac{1}{1+e^{-(\beta_0+\beta_1\times 0+\beta_2\times 22)}}\times\cdots\times\frac{1}{1+e^{-(\beta_0+\beta_1\times 0+\beta_2\times 21)}}$$

2.2 ロジスティック回帰のデモ

訪問時刻・担当者年齢を用いて契約結果を説明するモデル、つまり

$$\mathbb{P}[\text{契約結果}=1]=\frac{1}{1+e^{-(\beta_0+\beta_1\times\text{訪問時刻}+\beta_2\times\text{担当者年齢})}}$$

を考えます。

A. 線形性の確認

ロジスティック回帰は、説明変数と目的変数との間に線形性が仮定されていました。説明変数と目的変数との間の関係を可視化・数値要約することで、これを確認しましょう。

Hide

```
# 訪問時刻と契約結果の間の関係
table(dat$訪問時刻, dat$契約結果)
```

Hide

```
# 担当者年齢と契約結果の間の関係：線形性が確認できる。
par(family = "ヒラギノ角ゴシック W3")
plot(dat$担当者年齢, dat$契約結果)
```

B. ロジスティック回帰の計算

R 言語では次のようにロジスティック回帰を計算することができます。

Hide

```
# ロジスティック回帰
result <- glm(契約結果 ~ 訪問時刻 + 担当者年齢,
              data = dat)
summary(result)
```

問題 : 以下の問いに答えてください。

1. 推定によって得られた式を答えてください。
2. 担当者年齢の偏回帰係数の値が意味することを「オッズ」という言葉を用いて説明してください。

解答 :

1. 得られた式は以下の通りです。

$$\mathbb{P}[\text{契約結果}=1]=\frac{1}{1+e^{-(-0.34-0.45\times\text{訪問時刻}+0.03\times\text{担当者年齢})}}$$

2. 担当者年齢が1歳増えると、契約結果のオッズが$e^{0.03}\sim 1.03$倍増える。■

2.3 モデル選択

A. 赤池情報量規準

今回は、訪問時刻・担当者年齢を用いて契約結果を説明する式を作りました。しかし、次のように訪問時刻を含まないモデルも考えることができます。

Hide

```
# 訪問時刻を抜いたモデル
result2 <- glm(契約結果 ~ 担当者年齢, data = dat)
summary(result2)
```

複数のモデルを作ったら、どちらのモデルを採用するかを検討しましょう。今回も、モデル選択の際にヒントになる指標の一つとして、未知のデータへの予測の精度を推定する方法だった赤池情報量規準を用いることができます。（第2回資料も参考にしてみてください。）

Hide

```
# モデル比較
AIC(result); AIC(result2)
```

```
[1] 23.31996
[1] 28.80944
```

今回は、赤池情報量規準がより小さい訪問時刻を含んだロジスティック回帰を採用するほうが良いのではないかと推定できます。

第5回：多変量解析における行列の基礎

Code

1.　行列

数を長方形に並べたものを**行列**（matrix）といいます。特に縦に m 個、横に n 個だけ数を並べた行列をサイズ (m, n) の行列といいます。例えば、

$$\begin{pmatrix} 1 & -1 & 5 \\ 2 & 6 & 3 \end{pmatrix}$$

は、サイズ $(2, 3)$ 行列です。また行の数と列の数が同じ行列を**正方行列**といいます。また、m 行 1 列と数を縦に一列並べたものを**列ベクトル**、1 行 n 列と数を横に一行並べたものを**行ベクトル**といいます。

多変量解析には行列がたくさん登場します。みなさんがよく扱っているデータフレームは、まさに行列の代表例です。データフレームを**データ行列**とよんだりします。

Hide

```
# データの読み込み
dat <- read.csv("./data/scores.csv", fileEncoding = "utf-8")
dat
```

国語 <int>	数学 <int>	英語 <int>
90	30	155
95	35	115
100	50	200
90	45	150
85	20	120

5 rows

またデータの分散・共分散を一つにまとめて表現する**分散共分散行列**や、相関係数を一つにまとめて表現する**相関行列**も重要な役割を果たします。例えば先ほどのデータの分散共分散行列は、次のようなものです。

$$\begin{pmatrix} \text{国語の分散} & \text{国語と数学の共分散} & \text{国語と英語の共分散} \\ \text{数学と国語の共分散} & \text{数学の分散} & \text{数学と英語の共分散} \\ \text{英語と国語の共分散} & \text{英語と数学の共分散} & \text{英語の分散} \end{pmatrix}$$

R 言語では cov 関数を使って、分散共分散行列を計算することができます。

Hide

```
# 分散共分散行列
cov(dat)
```

```
       国語    数学     英語
国語  32.50  53.75  123.75
数学  53.75 142.50  296.25
英語 123.75 296.25 1157.50
```

この結果から「国語の分散は32.50」や「英語と数学の共分散は296.25」などがわかります。また相関行列とは、次のようなものです。

$$\begin{pmatrix} 1 & \text{国語と数学の相関係数} & \text{国語と英語の相関係数} \\ \text{数学と国語の相関係数} & 1 & \text{数学と英語の相関係数} \\ \text{英語と国語の相関係数} & \text{英語と数学の相関係数} & 1 \end{pmatrix}$$

R 言語では cor 関数を使って、分散共分散行列を計算することができます。

Hide

```
# 相関行列
cor(dat)
```

```
          国語       数学       英語
国語 1.0000000 0.7898222 0.6380328
数学 0.7898222 1.0000000 0.7294414
英語 0.6380328 0.7294414 1.0000000
```

この結果から「国語と数学の相関係数は0.79」などがわかります。

「行列を使わなくても何とかなるではないか」と思うかもしれませんが、行列を使うことで様々な数値をまとめて扱えるようになり、様々な統計量の計算に役立つわけです。

2. 行列・ベクトルの演算

行列・ベクトルの足し算、引き算、掛け算を説明します。

2.1 行列の足し算・引き算

同じサイズの行列どうしでは、各要素ごとに足し算・引き算をすることで、行列の足し算・引き算ができます。例を示します。

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 11 & 22 \\ 33 & 44 \end{pmatrix} = \begin{pmatrix} 12 & 24 \\ 36 & 48 \end{pmatrix}$$

$$\begin{pmatrix} 11 & 22 \\ 33 & 44 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix}$$

2.2 行列の掛け算

行列の掛け算の説明は容易でないので、例を使って説明します。

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 1 \times 5 + 2 \times 7 & 1 \times 6 + 2 \times 8 \\ 3 \times 5 + 4 \times 7 & 3 \times 6 + 4 \times 8 \end{pmatrix}$$
$$= \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

このように行列の掛け算は、足し算や引き算と異なり、サイズ (l, m) の行列とサイズ (m, n) の行列を掛けることで、サイズ (l, n) の行列を得ることができます。

2.3 演習

行列の足し算・引き算・掛け算の練習をやってみましょう。

問題 : 以下の行列の計算を行ってください。ただし、計算ができないものもあります。できないものについては、理由を述べてください。

1. $$\begin{pmatrix} 3 & 1 & -2 \\ 2 & -3 & 2 \end{pmatrix} + \begin{pmatrix} -1 & -5 & 2 \\ 2 & -3 & -1 \end{pmatrix}$$
2. $$\begin{pmatrix} 1 & 5 & -1 \\ -1 & 1 & 2 \end{pmatrix} + \begin{pmatrix} -1 & -5 \\ 2 & 2 \\ 3 & 3 \end{pmatrix}$$
3. $$\begin{pmatrix} 3 & 1 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} -1 & -5 \\ 2 & -3 \end{pmatrix}$$
4. $$\begin{pmatrix} -1 & -5 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 2 & -3 \end{pmatrix}$$
5. $$\begin{pmatrix} 1 & 5 & -1 \end{pmatrix} \begin{pmatrix} -1 & -5 & 1 \end{pmatrix}$$

解答 :

1. $$\begin{pmatrix} 3 & 1 & -2 \\ 2 & -3 & 2 \end{pmatrix} + \begin{pmatrix} -1 & -5 & 2 \\ 2 & -3 & -1 \end{pmatrix} = \begin{pmatrix} 2 & -4 & 0 \\ 4 & -6 & 1 \end{pmatrix}$$
2. 足し算は同じサイズの行列どうしでしか計算できません。ところがこの問題の式は、左の行列のサイズが $(2, 3)$、右の行列のサイズが $(3, 2)$ になっています。つまり、足し算ができません。
3. $$\begin{aligned} \begin{pmatrix} 3 & 1 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} -1 & -5 \\ 2 & -3 \end{pmatrix} &= \begin{pmatrix} 3 \times (-1) + 1 \times 2 & 3 \times (-5) + 1 \times (-3) \\ 2 \times (-1) + (-3) \times 2 & 2 \times (-5) + (-3) \times (-3) \end{pmatrix} \\ &= \begin{pmatrix} -1 & -18 \\ -8 & -1 \end{pmatrix} \end{aligned}$$
4. $$\begin{pmatrix} -1 & -5 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 2 & -3 \end{pmatrix} = \begin{pmatrix} -13 & 14 \\ 0 & 11 \end{pmatrix}$$
5. 行列の掛け算では、サイズ (l, m) の行列とサイズ (m, n) の行列を掛けることで、サイズ (l, n) を得ることができるのでした。この問題の式では左の行列のサイズが $(1, 3)$、右の行列のサイズが $(1, 3)$ であることから、行列の掛け算を行うことができません。

2.4 単位行列と逆行列

正方行列には**単位行列**といって、数の掛け算でいうところの 1 の役割を果たす行列があります。例えば、サイズ $(2, 2)$ の行列の場合

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

が成り立ちます。同様にして、対角成分が 1 でその他の成分が 0 であるようなサイズ (n, n) の正方行列

$$I_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

は、サイズ (n, n) のどんな正方行列に対しても $AI_n = A$ かつ $I_nA = A$ をみたします。正方行列 I_n を**単位行列**といいます。

またサイズ (n, n) の正方行列 A に対して、掛け算すると単位行列 I_n になるような正方行列 B を行列 A の**逆行列**といいます。逆行列はどんな正方行列にも存在するとは限りません。$(2, 2)$ の正方行列の場合、次のような公式が知られています。

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \Rightarrow A^{-1} = \frac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

この公式を証明しておきましょう。$AA^{-1} = I_2$ と $A^{-1}A = I_2$ を確認する必要がありますが、$AA^{-1} = I_2$ のみ示します。

$$\begin{aligned} AA^{-1} &= \frac{1}{ad - bc}\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \\ &= \frac{1}{ad - bc}\begin{pmatrix} ad - bc & 0 \\ 0 & ad - bc \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{aligned}$$

問題 : 以下の行列に逆行列があれば求めてください。逆行列がない場合は理由を説明してください。

1. $\begin{pmatrix} 3 & 1 & -2 \\ 2 & -3 & 2 \end{pmatrix}$
2. $\begin{pmatrix} 3 & 1 \\ -8 & -3 \end{pmatrix}$
3. $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

解答 :

1. 正方行列でないので、逆行列は存在しません。
2. サイズ $(2, 2)$ の正方行列に対する逆行列の公式を用います。$A^{-1} = \begin{pmatrix} 3 & 1 \\ -8 & -3 \end{pmatrix}$
3. サイズ $(2, 2)$ の正方行列に対する逆行列の公式を用います。$A^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

3. 行列と連立一次方程式

連立一次方程式は行列を用いて表現することができます。この事実は主成分分析の仕組みを理解するときに役に立ちます。例えば、連立一次方程式

$$\begin{cases} 2x & + & 4y & = 10 \\ x & + & 3y & = 6 \end{cases}$$

は行列を用いると次のように表現することができます。

$$\begin{pmatrix} 2 & 4 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 10 \\ 6 \end{pmatrix}$$

なお、逆行列を用いてこの連立一次方程式の解を求めることもできます。$A = \begin{pmatrix} 2 & 4 \\ 1 & 3 \end{pmatrix}$ の逆行列は

$$A^{-1} = \frac{1}{2} \begin{pmatrix} 3 & -4 \\ -1 & 2 \end{pmatrix}$$

なので、以下のように解を得ることができます。

$$\begin{aligned} \begin{pmatrix} x \\ y \end{pmatrix} &= A^{-1} \begin{pmatrix} 10 \\ 6 \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} 3 & -4 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 10 \\ 6 \end{pmatrix} \\ &= \begin{pmatrix} 3 \\ 1 \end{pmatrix} \end{aligned}$$

問題 : 以下の連立一次方程式を行列を用いて表してください。

1. $\begin{cases} x & + & 3y & = 4 \\ 2x & + & 3y & = 5 \end{cases}$
2. $\begin{cases} -x & + & 2y & = 3 \\ 2x & + & y & = 4 \end{cases}$

解答 :

1. $\begin{pmatrix} 1 & 3 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$
2. $\begin{pmatrix} -1 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$

4. 固有値問題

主成分分析の仕組みを理解するときに**固有値問題**が現れるので、ここで紹介します。固有値問題とは、正方行列 A に対して $Av = \lambda v$ をみたすような 0 でないベクトル v と値 λ の組 (v, λ) を求める問題のことです。ベクトル v のことを**固有ベクトル**、値 λ のことを**固有値**といいます。この問題は線形代数で解き方が知られているのですが、ここでは R 言語の関数 eigen を用いて解く方法を紹介します。

例題 : 行列 $A = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}$ の固有値と固有ベクトルを求めてください。

解答 :

Hide

```
A <- matrix(c(3, 1,
              1, 2), nrow = 2, byrow = TRUE)
eigen(A)
```

```
eigen() decomposition
$values
[1] 3.618034 1.381966

$vectors
           [,1]       [,2]
[1,] -0.8506508  0.5257311
[2,] -0.5257311 -0.8506508
```

この結果から固有値 3.618 の固有ベクトル $\begin{pmatrix} -0.851 \\ -0.526 \end{pmatrix}$ と固有値 0.526 の固有ベクトル $\begin{pmatrix} 0.526 \\ -0.851 \end{pmatrix}$ の二つがあることがわかります。■

問題 : 行列 $A = \begin{pmatrix} 4 & 1 & 5 \\ 1 & 9 & 3 \\ 5 & 3 & 1 \end{pmatrix}$ の固有値と固有ベクトルを求めてください。

解答 :

Hide

```
A <- matrix(c(4, 1, 5,
              1, 9, 3,
              5, 3, 1), nrow = 3, byrow = TRUE)
eigen(A)
```

```
eigen() decomposition
$values
[1] 11.179454  5.828534 -3.007988

$vectors
           [,1]       [,2]       [,3]
[1,] -0.4167586  0.7170824  0.5586636
[2,] -0.7958462 -0.5848058  0.1569428
[3,] -0.4392506  0.3792031 -0.8144102
```

この結果から固有値 11.179 の固有ベクトル $\begin{pmatrix} -0.417 \\ -0.796 \\ -0.439 \end{pmatrix}$、固有値 5.829 の固有ベクトル $\begin{pmatrix} 0.717 \\ -0.585 \\ 0.379 \end{pmatrix}$、固有値 -3.008 の固有ベクトル \begin{pmatrix}0.559\\0.157\\-0.814\end{pmatrix} の三つがあることがわかります。■

第6回 主成分分析（前半）

Code

1. 前準備

複数の変数からなるデータをより少ない変数で表現するタスクを**次元削減**（dimensionality reduction） といいます。一般論として、変数の多いデータを分析することは容易ではありません。散布図による可視化でデータを理解しようとしても、かかれる散布図の数は多くなり、その考察は難しくなります。しかし、次元削減の手法によって少ない変数でデータを表現することができれば、この問題を解決できる可能性があります。

例 : Iさんは、ラーメン春木屋の店員です。彼は、春木屋のラーメンの味が周辺にあるラーメン店と比較してどんな特徴があるのか調べてみようと思いました。そこで、麺・具・スープの味を5段階（1:低〜5:高）で評価したデータをとり、以下のような結果を得ました。

Hide

```
# データの読み込み
dat <- read.csv("./data/ramen.csv", fileEncoding = "cp932")
dat
```

店名 <chr>	麺 <int>	具 <int>	スープ <int>
春樹屋	2	4	5
もりまる	1	5	1
ラーメン宇宙	5	3	4
ラーメン三郎	2	2	3
大山屋	3	5	5
ばつね	4	3	2
ラーメン十七番	4	4	3
ぐうのね	1	2	1
きらきら星	3	3	2
麺屋・小次郎	5	5	3

1-10 of 10 rows

今回は、**主成分分析**（Principal Component Analysis）とよばれる次元削減の手法を用いて、この問いを考えていきたいと思います。まずは、主成分分析とは何かを説明しましょう。

2. 主成分分析

2.1 主成分分析の概要

A. 課題設定

主成分分析は、d 個の変数からなるデータを、より少ない $d' < d$ 個の変数で表現する（いわゆる次元削減とよばれる）手法のひとつです。データの変数を用いた一次式

$$\text{第1主成分} = a_1 \times \text{麺} + a_2 \times \text{具} + a_3 \times \text{スープ}$$

を考え、分散が一番大きくなるように係数 a_1, a_2, a_3 を決めたものを**第1主成分**（First Principal Component）といいます。ただこれだと、係数をとりあえず大きくするだけで、分散は意味もなく大きくなってしまいます。そのようなことを防ぐため、一般には

$$a_1^2 + a_2^2 + a_3^2 = 1$$

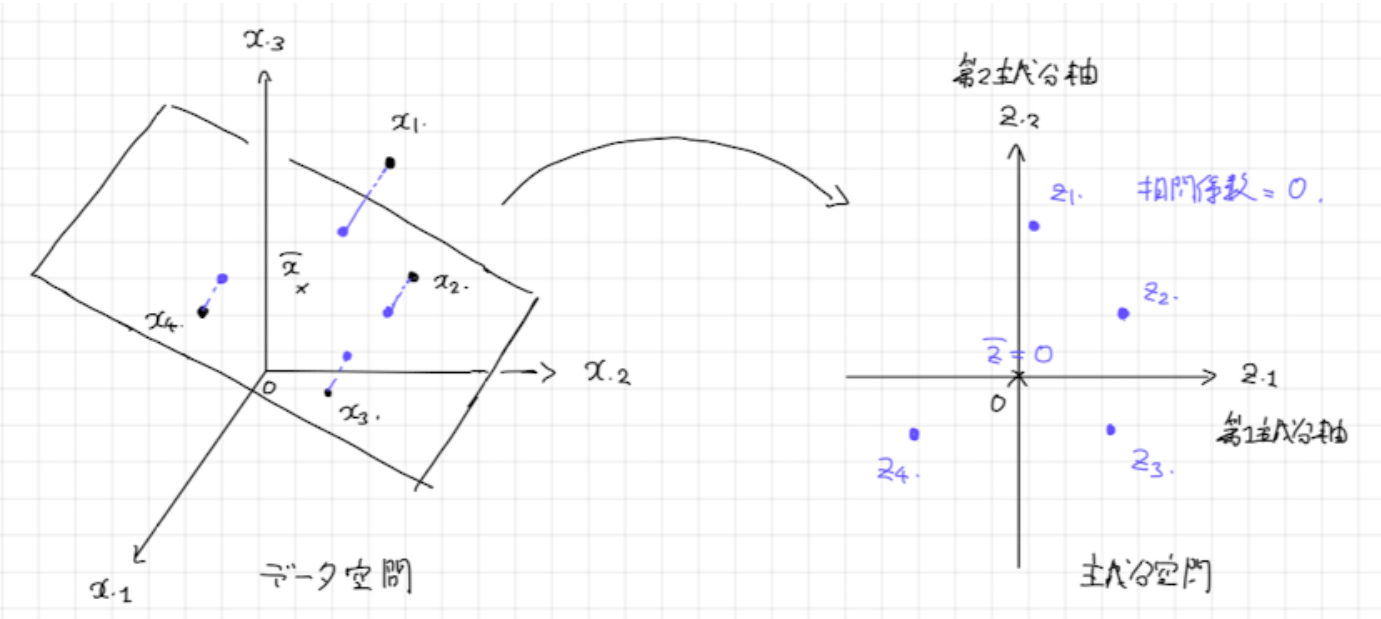
という係数の大きさの制約を設けます。

さらに、第1主成分とは直交するような係数 b_1, b_2, b_3（すなわち $a_1 b_1 + a_2 b_2 + a_3 b_3 = 0$）を持つ一次式

$$\text{第2主成分} = b_1 \times \text{麺} + b_2 \times \text{具} + b_3 \times \text{スープ}$$

のうち分散が最大になるものを**第2主成分**といいます。この場合も、$b_1^2 + b_2^2 + b_3^2 = 1$ という制約は設けておきます。

これまでの説明を図で表現して、イメージをつかんでおきましょう。



なお、主成分はデータの変数の個数（ラーメンのデータであれば3個）まで作ることができます。

B. 出来ること

主成分分析では、主成分とその寄与率を求めることができます。主成分によって、データに含まれる変数の要約を得られる可能性があります。例えば、主成分分析によって

$$\text{主成分} = 0.58 \times \text{麺} + 0.58 \times \text{具} + 0.58 \times \text{スープ}$$

という主成分が得られたとします。この主成分は「味の総合得点」を表していると解釈できるでしょう。

問題 : 次のような式が主成分として得られたとき、この主成分はどんな意味を表していると解釈できるでしょうか。

$$\text{主成分} = 0.71 \times \text{麺} + 0.00 \times \text{具} - 0.71 \times \text{スープ}$$

解答 : この主成分が正の値のラーメン店はスープより麺重視、負の値のラーメン店は麺よりスープ重視の店です。このことから「麺派・スープ派」といった解釈がつけられるでしょう。■

Remark : 主成分分析から得られるのは、あくまで主成分を定義する1次式の係数のみです。得られた主成分に意味があるのか、また意味があるならどう解釈するのかは、分析者が考える必要があります。主成分分析を闇雲に信じるのではなく、意味のある結果が得られているかを必ず考察するように注意しましょう。■

また主成分をできる限り全て求めたとき、主成分の分散の合計はデータの変数の分散の合計に一致することが知られています。つまり、

$$\text{第1主成分の分散} + \text{第2主成分の分散} + \text{第3主成分の分散} = \text{麺の分散} + \text{具の分散} + \text{スープの分散}$$

が成り立ちます。このことから、

$$\text{第}i\text{主成分の寄与率} = \frac{\text{第}i\text{主成分の分散}}{\text{麺の分散} + \text{具の分散} + \text{スープの分散}}$$

を主成分の**寄与率**とよび、データのどれくらいの情報をその主成分で要約できたかを表す指標として用いることが一般的です。

3. 主成分の係数を求める仕組み

主成分分析を行うことで、主成分の係数を求めることができます。ここでは、その仕組みの解説として、第1主成分の係数の求め方を説明することにしましょう。

A. 問題設定

簡単のため、次のようなデータを考えます。

id	x_1	x_2
1	p_1	q_1
2	p_2	q_2
平均	0	0

なお、簡単のため x_1, x_2 の平均値は 0 に中心化されているものとします。なお、x_1 の分散が $(p_1^2 + p_2^2)/2$、x_2 の分散が $(q_1^2 + q_2^2)/2$、x_1 と x_2 の共分散が $(p_1 q_1 + p_2 q_2)/2$、すなわち分散共分散行列が

$$\text{分散共分散行列} = \begin{pmatrix} \frac{p_1^2+p_2^2}{2} & \frac{p_1q_1+p_2q_2}{2} \\ \frac{p_1q_1+p_2q_2}{2} & \frac{q_1^2+q_2^2}{2} \end{pmatrix}$$

になります。この話は後で役に立つので、念頭に置いておくと良いでしょう。

さて、このデータの第1主成分 z を求めることは、

$$z = a_1 x_1 + a_2 x_2$$

の分散が最も大きくなるような係数 a_1, a_2 を求めることです。なお、係数を大きくすれば主成分の分散を無意味に大きくできてしまうので、$a_1^2 + a_2^2 = 1$ という制約を設けるのでした。

B. 主成分の分散を式で表現する。

主成分の分散を $a_1, a_2, p_1, p_2, q_1, q_2$ を用いて表すことで、どんな式を最大化するような a_1, a_2 を求めれば良いのかを明らかにしましょう。

主成分 $z = a_1 x_1 + a_2 x_2$ について、 id=1 の得点は $z_1 = a_1 p_1 + a_2 q_1$、 id=2 の得点は $z_2 = a_1 p_2 + a_2 q_2$ になっています。x_1, x_2 の平均値は 0 ですから、主成分 z の平均値もまた 0 です。このことから主成分 z の分散は

$$\text{主成分}z\text{の分散} = \frac{(a_1 p_1 + a_2 q_1)^2 + (a_1 p_2 + a_2 q_2)^2}{2}$$

と求めることができます。つまり、主成分の係数 a_1, a_2 を求めることは、関数

$$f(a_1, a_2) = \frac{(a_1 p_1 + a_2 q_1)^2 + (a_1 p_2 + a_2 q_2)^2}{2}$$

を制約条件 $a_1^2 + a_2^2 = 1$ をみたすように最大化する a_1, a_2 を求めなさいという数学の問題を解くことだとわかるわけです。

C. ラグランジュの未定乗数法

一般に、関数 $f(a_1, a_2)$ を制約条件 $g(a_1, a_2) = 0$ をみたすように最大化する a_1, a_2 の値について以下の事実が成り立つことが知られています：関数 $F(a_1, a_2, \lambda) = f(a_1, a_2) - \lambda g(a_1, a_2)$ としたとき、関数 $f(a_1, a_2)$ を最大化する値 a_1, a_2 は連立方程式

$$\begin{aligned} \frac{\partial F}{\partial a_1} &= 0 \\ \frac{\partial F}{\partial a_2} &= 0 \\ \frac{\partial F}{\partial \lambda} &= 0 \end{aligned}$$

の解である。これを**ラグランジュの未定乗数法**といいます。

ラグランジュの未定乗数法を用いれば、主成分分析の係数 a_1, a_2 を求めるための問題を解くことができます。関数 $F(a_1, a_2, \lambda)$ は

$$F(a_1, a_2, \lambda) = \frac{(a_1 p_1 + a_2 q_1)^2 + (a_1 p_2 + a_2 q_2)^2}{2} - \lambda(a_1^2 + a_2^2 - 1)$$

なので、解く連立方程式は

$$\begin{aligned} (a_1 p_1 + a_2 q_1) p_1 + (a_1 p_2 + a_2 q_2) p_2 - 2\lambda a_1 &= 0 \\ (a_1 p_1 + a_2 q_1) q_1 + (a_1 p_2 + a_2 q_2) q_2 - 2\lambda a_2 &= 0 \\ a_1^2 + a_2^2 - 1 &= 0 \end{aligned}$$

です。式を a_1, a_2 に注目して整理すると、連立方程式を次のように書き換えることができます。

$$\begin{aligned} \frac{(p_1^2 + p_2^2)}{2} a_1 + \frac{(p_1 q_1 + p_2 q_2)}{2} a_2 &= \lambda a_1 \\ \frac{(p_1 q_1 + p_2 q_2)}{2} a_1 + \frac{(q_1^2 + q_2^2)}{2} a_2 &= \lambda a_2 \\ a_1^2 + a_2^2 &= 1 \end{aligned}$$

D. 行列で表現してみる。

C. ラグランジュの未定乗数法 で得られた連立方程式を行列で表現してみましょう。

$$\begin{pmatrix} \frac{p_1^2+p_2^2}{2} & \frac{p_1q_1+p_2q_2}{2} \\ \frac{p_1q_1+p_2q_2}{2} & \frac{q_1^2+q_2^2}{2} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \lambda \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$a_1^2 + a_2^2 = 1$$

すると、主成分の係数 a_1, a_2 は「分散共分散行列の固有ベクトル」で、特に大きさが 1 のものだということがわかります。

Remark : さらに固有値は主成分の分散の値に等しくなることもわかります。この証明は容易ではないので省略させてください。

以上で、主成分分析で主成分の係数を求める仕組みは、分散共分散行列の固有値問題を解くことだということを解説しました。次回は、 R 言語の固有値問題を解く関数である eigen 関数を用いて、ラーメン店のデータの主成分分析を行います。また、 R 言語の主成分分析を行う prcomp 関数を用いて主成分分析を行い、結果を eigen 関数で得られたものと比較します。

第7回 主成分分析（後半）

Code

1. 前回の復習

前回は、主成分分析とその仕組みについて解説しました。一度、復習しておきましょう。

第1主成分 : 主成分分析は、d 個の変数 $x_1, \cdots, x_d$ からなるデータを、より少ない $d' < d$ 個の変数 $z_1, \cdots, z_{d'}$ で表現する（いわゆる次元削減とよばれる）手法のひとつでした。z_1 は $x_1, \cdots, x_d$ の重みつき和

$$z_1 = a_1 x_1 + \cdots + a_d x_d$$

です。係数 $a_1, \cdots, a_d$ は $a_1^2 + \cdots + a_d^2 = 1$ をみたすもののなかで変数 z_1 の分散が最も大きくなるように決めるのでした。変数 z_1 を第1主成分といいます。実は、この係数 $a_1, \cdots, a_d$ は分散共分散行列の大きさ1の固有ベクトルであることがわかるのでした。

主成分と寄与率 : 第1主成分と直交する一次式 $z_2 = b_1 x_1 + \cdots + b_d x_d$ のうち分散が最大になるようなものを第2主成分といいます。第3主成分以降も同様の考え方で定義することができます。実は、次の事実が成り立ちます。

1. 主成分は、データに含まれる変数の個数の分だけ作ることができます。
2. 主成分の係数は分散共分散行列の大きさ1の固有ベクトルとして求めることができます。
3. 固有値は主成分の分散に等しくなります。

なお、主成分の分散の和はデータの変数の分散の和に等しくなることから、第1主成分の寄与率を 第1主成分の分散/データの変数の分散の和 と定義するのでした。これは、第2主成分以降でも同様です。

今回はこの話に基づいて、実際のデータに主成分分析を計算してみましょう。

2. eigen 関数を用いた主成分分析

A. ラーメンデータの紹介

Iさんは、ラーメン春木屋の店員です。彼は、春木屋のラーメンの味が周辺にあるラーメン店と比較してどんな特徴があるのか調べてみようと思いました。そこで、麺・具・スープの味を5段階（1:低〜5:高）で評価したデータをとり、以下のような結果を得ました。

Hide

```
# データの読み込み
dat <- read.csv("./data/ramen.csv", fileEncoding = "cp932", row.names = "店名")
dat
```

	麺 <int>	具 <int>	スープ <int>
春樹屋	2	4	5
もりまる	1	5	1
ラーメン宇宙	5	3	4
ラーメン三郎	2	2	3
大山屋	3	5	5

	麺 <int>	具 <int>	スープ <int>
ばつね	4	3	2
ラーメン十七番	4	4	3
ぐうのね	1	2	1
きらきら星	3	3	2
麺屋・小次郎	5	5	3

1-10 of 10 rows

このデータに対して、まず固有値問題を解く eigen 関数によって主成分分析を行い、主成分分析を計算する prcomp 関数を紹介して結果が同じになっていることを確認します。

B. 前処理（データの標準化）

最初に主成分分析を行うにあたっての注意点を一つ紹介します。主成分分析では、データをまずは標準化することが一般的です。

問題 : 主成分分析の前にデータを標準化する目的を考えてみましょう。

解答 : 大きな分散を持つ変数がデータに含まれていたとき、出来る主成分がその変数に大きく依存してしまうため。■

標準化したデータの分散共分散行列は相関行列に等しいことが知られています。以下のスクリプトでこのことを確認してみましょう。

Hide

```
# 標準化したデータの分散共分散行列
X_scaled <- scale(dat)
covmat_scaled <- cov(X_scaled)
covmat_scaled
```

```
             麺         具     スープ
麺      1.0000000 0.1905002 0.3600411
具      0.1905002 1.0000000 0.3004804
スープ 0.3600411 0.3004804 1.0000000
```

Hide

```
# 相関行列
cormat <- cor(dat)
cormat
```

```
             麺         具     スープ
麺      1.0000000 0.1905002 0.3600411
具      0.1905002 1.0000000 0.3004804
スープ 0.3600411 0.3004804 1.0000000
```

C. 主成分と寄与率を求める

R 言語では eigen 関数を用いることで、大きさ1の固有ベクトルとその固有値を求めることができます。つまり、B節で求めた相関行列 cormat を eigen 関数に渡すと、主成分の係数と寄与率を求めることができるはずです。

Hide

```
# 相関行列の固有値分解
result <- eigen(cormat)
result
```

```
eigen() decomposition
$values
[1] 1.5728539 0.8140083 0.6131378

$vectors
          [,1]       [,2]       [,3]
[1,] 0.5715110  0.6044710  0.5549685
[2,] 0.5221161 -0.7896069  0.3223595
[3,] 0.6330639  0.1055260 -0.7668731
```

この結果から第1主成分の式は

$$\text{第1主成分} = 0.57 \times \text{麺の}z\text{得点} + 0.52 \times \text{具の}z\text{得点} + 0.63 \times \text{スープの}z\text{得点}$$

で、第1主成分の分散は1.57であることがわかります。同様に第2主成分の式は

$$\text{第2主成分} = 0.60 \times \text{麺の}z\text{得点} - 0.79 \times \text{具の}z\text{得点} + 0.11 \times \text{スープの}z\text{得点}$$

で、第2主成分の分散は0.81であることがわかります。第3主成分も同様にして得ることができます。

Remark : 主成分を求めるときデータを標準化したので、1次式は麺・具・スープのz得点に対して作られることに注意してください。■

また第1主成分・第2主成分・第3主成分の寄与率も次のように求めることができます。

Hide

```
# 寄与率の計算
result$values / sum(result$values)
```

```
[1] 0.5242846 0.2713361 0.2043793
```

ところで主成分分析では、主成分の式を求めたあと、その式に分析者が解釈を与えることが大切です。

問題 : 第1主成分・第2主成分の解釈を与えてください。

解答 : 第1主成分の係数はどれもほぼ等しいので「全体的なうまさ」を表していて、正の値に大きくなるほどうまいことがわかります。第2主成分の係数は麺と具の符号が逆になっていて、スープの係数がほぼ0に近いので「麺重視か具重視か」を表しています。そして、正の値に大きくなるほど麺重視、負の値に大きくなるほど具重視であることがわかります。■

D. 主成分得点を求める

主成分が各データ点についていくらであるかを求めたものを**主成分得点**といいます。例えば「春木屋の全体的なうまさ（第1主成分得点）と麺重視か具重視か（第2主成分得点）」に興味があるわけです。

これも行列の掛け算を使って計算することができます。次の通りです。

Hide

```
# 主成分得点
X_scaled <- scale(dat)
X_scaled %*% result$vectors
```

```
                    [,1]       [,2]         [,3]
春樹屋          0.7119408 -0.5216497 -1.373736133
もりまる       -0.9740499 -1.8911205  0.645382316
ラーメン宇宙    0.9804158  1.2947047 -0.002322692
ラーメン三郎   -1.0513965  0.6781104 -0.864614382
大山屋          1.5401350 -0.7888582 -0.726820118
ばつね         -0.2766766  0.7435735  0.683778524
ラーメン十七番  0.6049920  0.1436935  0.429217649
ぐうのね       -2.3084890  0.1269792 -0.178513165
きらきら星     -0.6600579  0.3380821  0.311494336
麺屋・小次郎    1.4331863 -0.1235150  1.076133664
```

この結果から、春木屋の全体的なうまさ（第1主成分得点）は 0.71 、麺重視か具重視か（第2主成分得点）は -0.52 です。春木屋は全体的なうまさの得点が10店舗中4番目であることから、味は「そこそこ美味しい」くらいなんだろうなということがわかります。また、麺重視か具重視かの得点が負であることから、具重視の店であることがわかります。

3. prcomp 関数を用いた主成分分析

A. 主成分・寄与率・主成分得点を求める

prcomp 関数は、これまでの主成分分析の計算を一気に行うことができる R 言語の関数です。

Hide

```
# prcompによる主成分分析
result <- prcomp(dat, scale. = TRUE)
result
```

```
Standard deviations (1, .., p=3):
[1] 1.2541347 0.9022241 0.7830312

Rotation (n x k) = (3 x 3):
            PC1        PC2        PC3
麺     0.5715110 -0.6044710  0.5549685
具     0.5221161  0.7896069  0.3223595
スープ 0.6330639 -0.1055260 -0.7668731
```

result の結果が固有値分解で得られた結果と同じになっていることが確認できるでしょう。細かい違いを列挙しておきます。 * prcomp の Standard deviation には固有値のルート（主成分の標準偏差）が出力されている。 * 第2主成分の解釈が eigen の結果と逆になっている。 また、主成分得点は次のようにして得られます。

Hide

```
# 主成分得点
result$x
```

```
                 PC1        PC2          PC3
春樹屋        0.7119408  0.5216497 -1.373736133
もりまる      -0.9740499  1.8911205  0.645382316
ラーメン宇宙    0.9804158 -1.2947047 -0.002322692
ラーメン三郎   -1.0513965 -0.6781104 -0.864614382
大山屋        1.5401350  0.7888582 -0.726820118
ばつね       -0.2766766 -0.7435735  0.683778524
ラーメン十七番  0.6049920 -0.1436935  0.429217649
ぐうのね      -2.3084890 -0.1269792 -0.178513165
きらきら星     -0.6600579 -0.3380821  0.311494336
麺屋・小次郎    1.4331863  0.1235150  1.076133664
```

B. 主成分負荷量

データの各変数と主成分得点との相関係数を**主成分負荷量**といいます。主成分負荷量は、 係数*主成分の標準偏差 に等しくなることが知られています。

問題 : 第1主成分の麺・具・スープに対する主成分負荷量を求めてください。

解答 : 第1主成分の麺に対する主成分負荷量は $0.57 \times 1.25 = 0.71$、具に対する主成分負荷量は 0.65、スープに対する主成分負荷量は 0.79 とわかります。■

この事実は次のように、麺と主成分の相関係数を直接計算することで確認することができます。

Hide

```
# 麺と主成分の相関係数
cor(dat, result$x)
```

```
           PC1        PC2        PC3
麺     0.7167518 -0.5453683  0.4345576
具     0.6548039  0.7124024  0.2524175
スープ 0.7939474 -0.0952081 -0.6004855
```

1列めに第1主成分の麺・具・スープに対する主成分得点の計算結果が示されていますが、問題の解答と値が一致していることがわかります。主成分負荷量は主成分の係数のかわりに、主成分の意味を解釈するヒントとして用いられます。

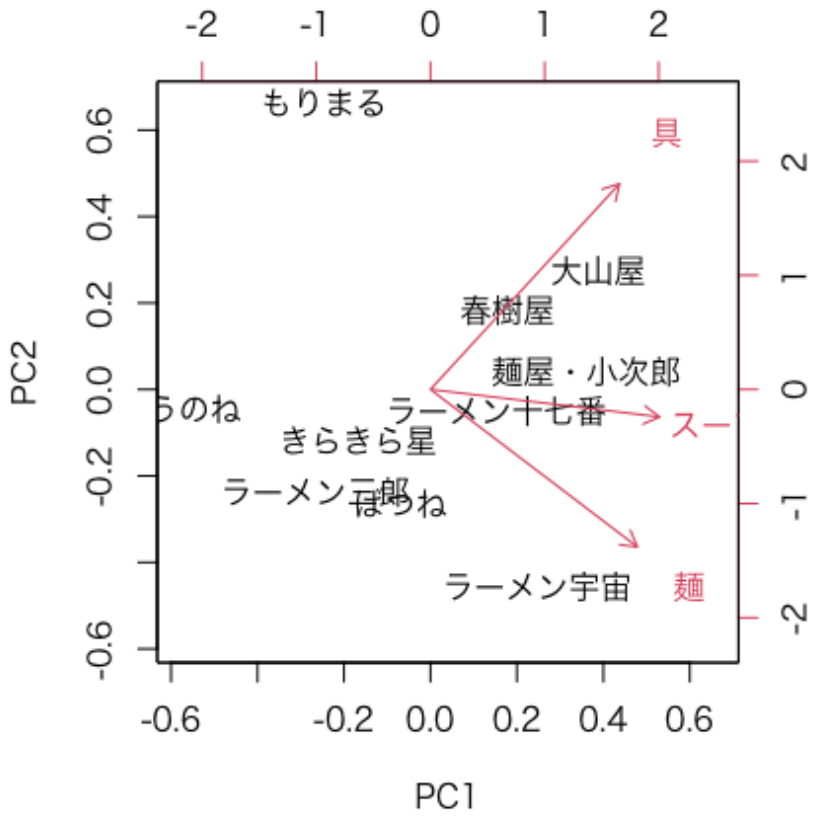
C. biplot

主成分得点と主成分負荷量を示したものに、biplotがあります。biplotはとても便利な図なので、主成分分析を行った際にはぜひかいてみることをお勧めします。さまざまなことをこのグラフから読み取ることができますが、例えば次のようなことがわかります。

- 第2主成分は、正の値に大きいほど具重視、負の値に大きいほど麺重視。
- うまい店トップ3は大山屋、麺屋・小次郎、ラーメン宇宙。

Hide

```
par(family = "ヒラギノ角ゴシック W3")    # Macのみ
biplot(result)
```


もりまる
具
大山屋
春樹屋
麺屋・小次郎
らのね
ラーメン十七番
スー
きらきら星
ラーメン二郎
はうね
ラーメン宇宙
麺
PC1
PC2
-2
-1
0
1
2
-0.6
-0.2
0.0
0.2
0.4
0.6

第8回 探索的因子分析

Code

1. 前準備

主成分分析と同様に、複数の変数からなるデータをより少ない変数で表現する手法として**探索的因子分析**（exploratory factor analysis）があります。今回は、ラーメンデータを用いながら探索的因子分析のイメージをつかんでみましょう。

2. 探索的因子分析

A. 概要

探索的因子分析は、第1因子・第2因子といったデータには観測されていない変数によって、データに観測されている麺や具やスープの味を説明できるのではないかと考えるモデルです。第1因子や第2因子のことを**共通因子**といいます。また、共通因子は2個に限らず、さまざまな個数を考えることができます。

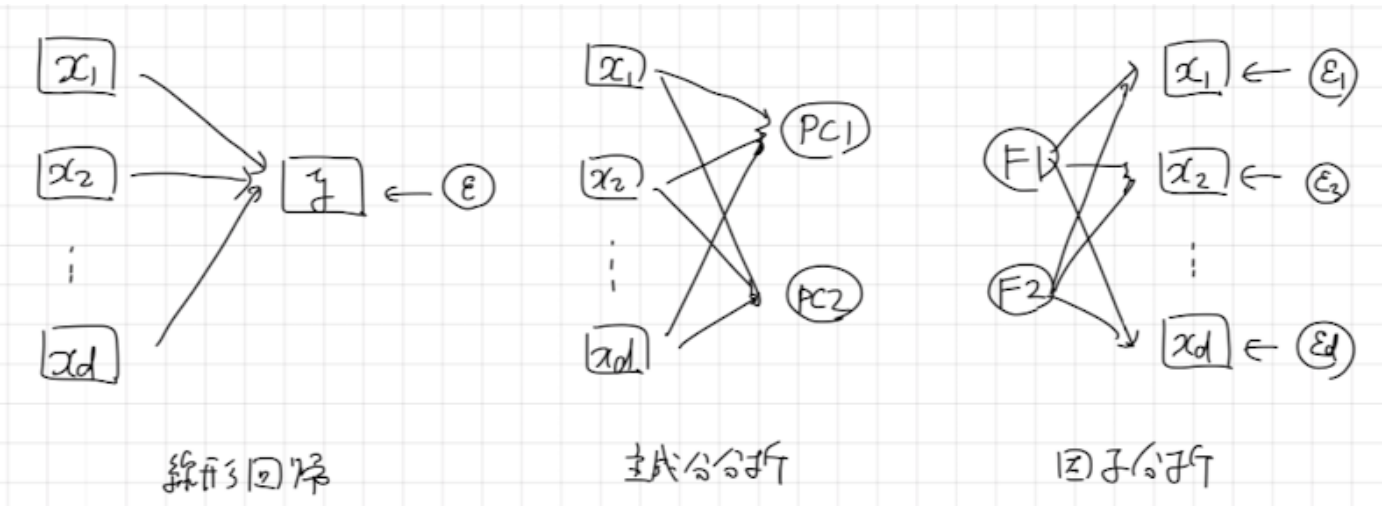
$$
\begin{aligned}
\text{麺の標準化得点} &= a_{11}\,\text{第1因子} + a_{12}\,\text{第2因子} + \text{麺の独自因子} \\
\text{具の標準化得点} &= a_{21}\,\text{第1因子} + a_{22}\,\text{第2因子} + \text{具の独自因子} \\
\text{スープの標準化得点} &= a_{31}\,\text{第1因子} + a_{32}\,\text{第2因子} + \text{スープの独自因子}
\end{aligned}
$$

係数 a_{ij} を**因子負荷量**（loadings）、各データ点の因子の値を**因子得点**（factor score）といいます。また、データに含まれる変数それぞれに対して、因子で説明できなかった誤差に対応する項を**独自因子**（unique factor）といいます。

各共通因子は期待値 0, 分散 1 とします。独自因子の期待値は 0 とし、各変数の独自因子間には相関はないものとします。なお、因子と独自因子との間の相関はないものとします。

問題 : パス図とよばれるものを使うことで因子分析の式をイメージしやすくすることができます。講師の説明を聞きながら、線形回帰・主成分分析・因子分析のパス図をかいてみましょう。

解答 :



B. できること

探索的因子分析では、

- **因子負荷量** : 得られた因子を解釈するためのヒント
- **共通性・独自性** : 各変数が因子によってどれだけ説明されているか

- **因子得点** : 各データ点の因子の値

の3つが得られます。また、**回転**（rotation）とよばれるものを選択することで、因子の間に相関を許すかを決めることができます。許した場合は、**因子間相関**が得られます。

C. 仕組み

探索的因子分析では、まず因子負荷量と独自因子の分散の2つを求めます。求め方には最尤法や最小二乗法といった方法が知られています。この具体的な話は難しいので説明を省略させてください。探索的因子分析では、あらかじめ標準化した変数を考えます。そのため変数の分散は 1 であり 1-独自因子の分散 は因子によって変数がどれだけ説明されたかを意味していると解釈できます。 1-独自因子の分散 を共通性、 独自因子の分散 を独自性といいます。

Remark : 共通性はいわゆる「データの変数に対する因子の決定係数」と解釈することができます。■

さて次に、因子負荷量から因子の解釈をおこないます。ところで、因子負荷量と因子得点には無数の解があることが知られています。以下の問題を確認してみてください。

問題 : 以下の問いに答えてください。

1. 行列を用いて因子分析の式を表してみてください。
2. 回転行列を用いて、因子負荷量と因子得点には無数の解があることを示してください。

解答 :

1. $A = (a_{ij})$、$e = (\text{麺の独自因子}, \text{具の独自因子}, \text{スープの独自因子})$ とします。また、データを X、因子得点を表す行列を F で行列で表すことにします。このとき、$X = AF + e$ となります。
2. 回転行列とは、次のような行列のことです。

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin(-\theta) \\ \sin\theta & \cos\theta \end{pmatrix}$$

   回転行列には $R(\theta)R(-\theta) = I_2$ という性質があります。これを用いると、$A' = AR(\theta), F' = R(-\theta)F$ とおくことで

$$AF + e = A'F' + e$$

   が成り立つので、因子負荷量と因子得点には無数の解があることがわかります。■

そこで、ひとまず解の候補を一つ求めておいて、**回転**（rotation）という操作により解釈しやすい結果を探していきます。ひとまず求めた一つの解を**無回転解**といいます。回転の代表例には**バリマックス回転**（varimax rotation）と**プロマックス回転**（promax rotation）があり、共通因子の間に相関を認めない場合にはバリマックス回転、認める場合にプロマックス回転が用いられます。バリマックス回転とプロマックス回転の詳細については、3節でデモを示してから、改めて4節で説明します。

最後に因子得点を予測します。一般には、データの変数の1次式

$$\begin{aligned} \text{第1因子の予測値} &= w_{11}\text{麺の標準化得点} + w_{12}\text{具の標準化得点} + w_{13}\text{スープの標準化得点} \\ \text{第2因子の予測値} &= w_{21}\text{麺の標準化得点} + w_{22}\text{具の標準化得点} + w_{23}\text{スープの標準化得点} \end{aligned}$$

を考え、係数 w_{ij} を最小二乗法で求めます。

Remark : 因子得点は観測されていない値であり、また因子分析の中で推定される値でもありません。このことから、最小二乗法を行うときの正解が見当たらず、「最小二乗法」と言われてもどうやってやるのか不可解に感じる方もいるでしょう。実は、最小化したい誤差の分散は

- 因子得点の分散共分散行列（対角行列 or 因子相関の推定値）
- 因子負荷量の推定値

- データの分散共分散行列

さえわかっていれば計算できることを確認することができます。ここで「因子得点の分散共分散行列」が既知であることから、因子得点の真の値を知る必要はないという仕組みになっています。

3. fa関数を用いた探索的因子分析

A. パッケージの読み込み

因子分析は psych パッケージの fa 関数を用いて計算することができます。 psych パッケージをインストールして、読み込んでおきましょう。

Hide

```
# faパッケージの読み込み
# install.packages("psych")
library(psych)
```

B. ラーメンデータの紹介

主成分分析で用いたラーメンデータに因子分析を行っていきましょう。Iさんは、ラーメン春木屋の店員です。彼は、春木屋のラーメンの味が周辺にあるラーメン店と比較してどんな特徴があるのか調べてみようと思いました。そこで、麺・具・スープの味を5段階（1:低〜5:高）で評価したデータをとり、以下のような結果を得ました。

Hide

```
# データの読み込み
dat <- read.csv("./data/ramen.csv", fileEncoding = "cp932", row.names = "店名")
dat
```

	麺 <int>	具 <int>	スープ <int>
春樹屋	2	4	5
もりまる	1	5	1
ラーメン宇宙	5	3	4
ラーメン三郎	2	2	3
大山屋	3	5	5
ばつね	4	3	2
ラーメン十七番	4	4	3
ぐうのね	1	2	1
きらきら星	3	3	2
麺屋・小次郎	5	5	3

1-10 of 10 rows

C. 因子負荷量の初期値と共通性を求める

R 言語では fa 関数を用いることで、探索的因子分析のさまざまな計算を実行できます。まずは因子負荷量の無回転解を求め、その共通性を確認します。今回は、2因子モデルを想定して、無回転解を求めてみましょう。

Hide

```
# 因子負荷量の初期値
result_none <- fa(dat, nfactors = 2, fm = "ml", rotate = "none")
print(result_none$loadings, cutoff = 0)
```

```

Loadings:
       ML1    ML2
麺       0.505  0.383
具       0.372  0.007
スープ  0.811 -0.131

                 ML1   ML2
SS loadings    1.052 0.164
Proportion Var 0.351 0.055
Cumulative Var 0.351 0.405
```

Hide

```
# 共通性
result_none$communalities
```

```
       麺        具    スープ
0.4021030 0.1380670 0.6753162
```

問題 : 得られた結果を因子分析の式で表してください。

解答 :

$$
\begin{aligned}
\text{麺の標準化得点} &= 0.505\text{第1因子} + 0.383\text{第2因子} + \text{麺の独自因子} \\
\text{具の標準化得点} &= 0.372\text{第1因子} + 0.007\text{第2因子} + \text{具の独自因子} \\
\text{スープの標準化得点} &= 0.811\text{第1因子} - 0.131\text{第2因子} + \text{スープの独自因子}
\end{aligned}
$$

D. 回転して解釈可能な因子を探す

バリマックス回転解

共通因子の間の相関は 0 だと仮定して、解釈可能な因子負荷量を探していきましょう。これには、一つの方法としてバリマックス回転解を求めるという方法があります。

Hide

```
result_varimax <- fa(dat, nfactors = 2, fm = "ml", rotate = "varimax")
print(result_varimax$loadings, cutoff = 0)
```

```
Loadings:
       ML1   ML2
麺      0.248 0.584
具      0.319 0.191
スープ 0.769 0.291

                 ML1   ML2
SS loadings    0.754 0.462
Proportion Var 0.251 0.154
Cumulative Var 0.251 0.405
```

Hide

```
result_varimax$communalities
```

```
       麺         具     スープ
0.4021030 0.1380670 0.6753162
```

プロマックス回転解

バリマックス回転解はあまりしっくりした解釈を行うことができません。そこで今度は、共通因子の間に相関があってもよいとして、解釈可能な因子負荷量を探していきましょう。これには、一つの方法としてプロマックス回転解を求めるという方法があります。

Hide

```
result_promax <- fa(dat, nfactors = 2, fm = "ml", rotate = "Promax")
print(result_promax$loadings, cutoff = 0)
```

```
Loadings:
       ML1    ML2
麺     -0.029  0.654
具      0.296  0.097
スープ  0.816  0.008

                 ML1   ML2
SS loadings    0.754 0.438
Proportion Var 0.251 0.146
Cumulative Var 0.251 0.397
```

問題 : プロマックス回転解の解釈を与えてください。

解答 : 第1因子は具とスープの美味しさ、第2因子は麺の美味しさ■

Hide

```
result_promax$communalities
```

```
       麺         具     スープ
0.4021030 0.1380670 0.6753162
```

共通因子間の相関、因子間相関の値を確認しておきましょう。

Hide

```
# 因子間相関
matrix(result_promax$Phi, ncol = 2)
```

```
          [,1]      [,2]
[1,] 1.0000000 0.7089695
[2,] 0.7089695 1.0000000
```

問題 : なお統計検定準1級では、共通性の値と因子負荷量・因子間相関の間に成り立つ関係が問われることがあります。

$$\textbf{麺の標準化得点} = a_1\,\textbf{第1因子} + a_2\,\textbf{第2因子} + \textbf{独自因子}$$

という関係が成り立つことに注目し、以下の問いに答えてください。

1. $\textbf{共通性} = 1 - \textbf{独自性}$ という定義に注意して、$\textbf{共通性} = a_1^2 + a_2^2 + 2a_1a_2\,\textbf{因子間相関}$ が成り立つことを証明してください。
2. 麺の因子負荷量がそれぞれ、$a_1 = -0.029, a_2 = 0.654$、因子間相関が 0.709 のとき、麺の共通性を求めてください。

解答 :

1. 共分散の公式・共通因子と独自因子の無相関性から

   $$\textbf{麺の標準化得点の分散} = a_1^2\,\textbf{第1因子の分散} + a_2^2\,\textbf{第2因子の分散} + 2a_1a_2\,\textbf{因子相関} + \textbf{独自性}$$

   が従いますが、麺の標準化得点・第1因子・第2因子の分散はすべて 1 なので
   $\textbf{共通性} = 1 - \textbf{独自性} = a_1^2 + a_2^2 + 2a_1a_2\,\textbf{因子間相関}$ が従います。
2. Rを用いて計算します。

Hide

```
(-0.029)^2 + 0.654^2 + 2*(-0.029)*0.654*0.709
```

```
[1] 0.4016632
```

なお、この計算はプロマックス回転解の結果を参照したものです。実際、プロマックス回転解の麺の共通性は 0.402 になっています。■

E. 因子得点を求める

各共通因子が各データ点についていくらであるかを求めたものを因子得点といいます。例えば「春木屋の具とスープの美味しさ（第1因子得点）と麺の美味しさ（第2因子得点）」に興味があるわけです。

Hide

```
# 因子得点
result_promax$scores
```

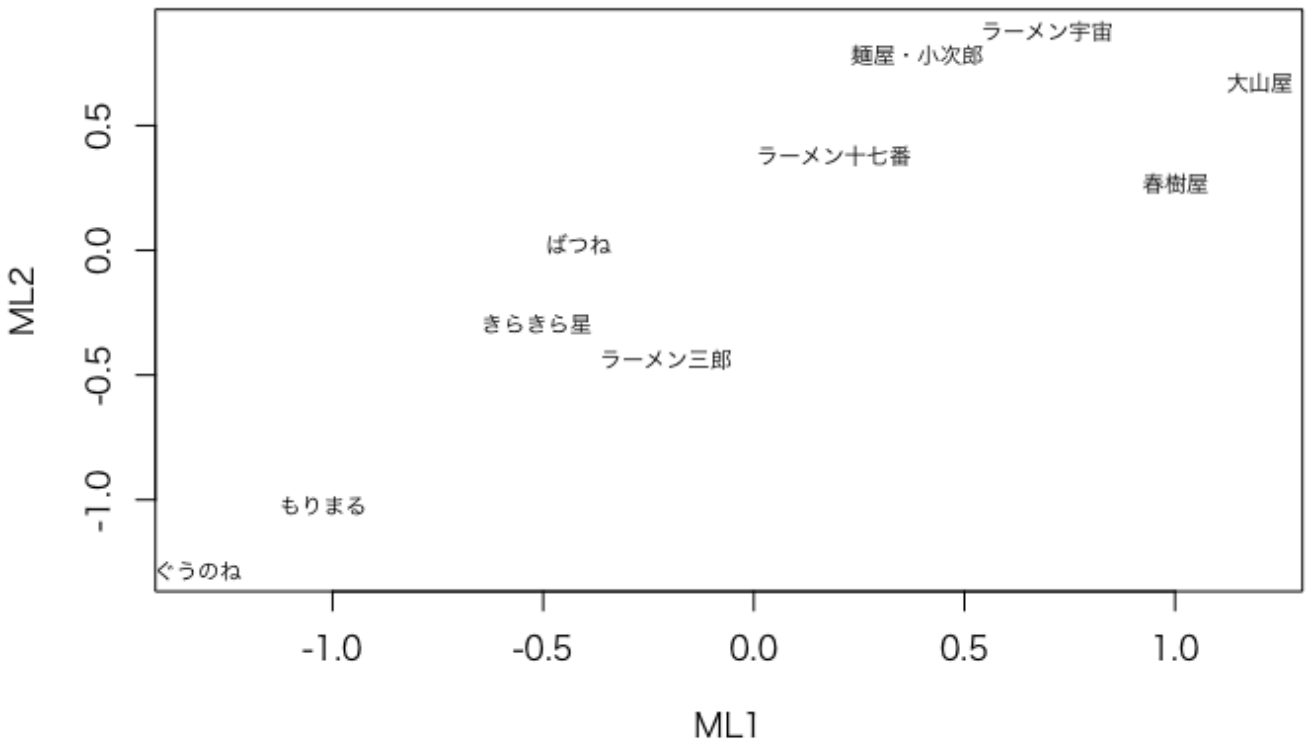
```
                ML1         ML2
春樹屋        1.0024076  0.27292478
もりまる     -1.0218746 -1.02281623
ラーメン宇宙   0.6965441  0.87914009
ラーメン三郎  -0.2080422 -0.43036389
大山屋        1.2014756  0.67823590
ばつね       -0.4151310  0.02861402
ラーメン十七番 0.1900939  0.38025835
ぐうのね     -1.3197173 -1.28088996
きらきら星    -0.5149181 -0.29067253
麺屋・小次郎   0.3891620  0.78556947
```

F. 因子得点を散布図に可視化する

E節で得られた因子得点を散布図に可視化することで、春木屋が他の店に比べて美味しいのか、またどういう特徴があるのかを考察することができます。

Hide

```
# 因子得点の散布図
par(family = "ヒラギノ角ゴシック W3")   # Macユーザーのみ
plot(result_promax$scores, type = "n")
text(result_promax$scores, labels=rownames(dat), cex = 0.7)
```



右上の店ほど全体的に味が美味しい店です。春木屋は右上にあるので味が美味しい店であることがわかります。また右に行くほど麺に、上に行くほど具やスープに重点をおいた店です。春木屋は味が美味しい店のなかでも麺に重点を置いていることがわかります。

4. 解釈可能な因子負荷量を求めるための工夫

A. サーストンの単純構造

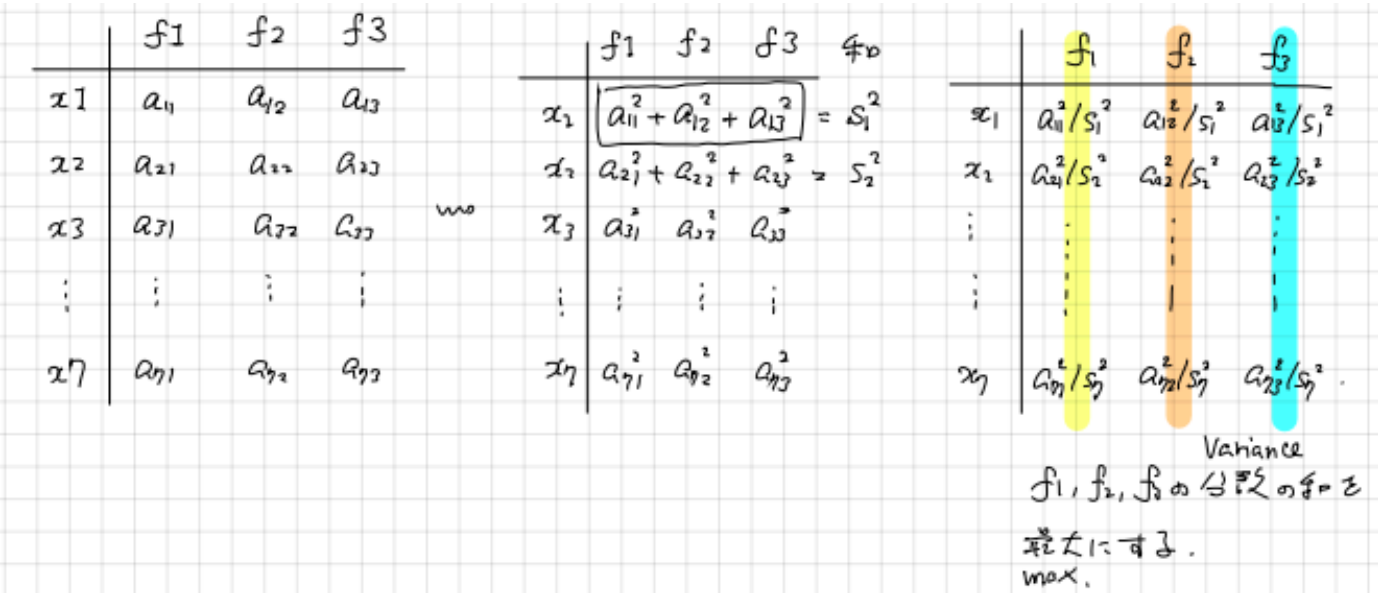
解釈可能な因子負荷量とは何かを考える上で、1947年に提案された**サーストンの単純構造**という基準が参考になります。ここで、m を因子の数とします。

1. 因子負荷行列の各行には、少なくとも一つの0の要素があること。
2. 因子負荷行列の各列には、m 個以上の0の要素があること。
3. 2つの列に注目したとき、一方の列の要素が 0 で他方の列の要素は 0 でない行があること。
4. $m \geq 4$ のとき、2つの列に注目したときどちらの要素も 0 である行があること。
5. 二つの列に注目したとき、どちらの要素も 0 でない行が少ないこと。

以下にのべるバリマックス回転やプロマックス回転は、このサーストンの単純構造をなるべくみたすような因子負荷行列を求める仕組みです。

B. バリマックス回転

以下のような条件をみたすように因子負荷行列を求めることで、サーストンの単純構造をなるべく持つような因子負荷行列を求めようという考え方を、**（規準化）バリマックス回転**といいます。



なおバリマックス回転は数式で紹介されることが多いので、ここでもその説明を与えておきます。$\lambda_{ij} = a_{ij}/s_i$ とおいて、因子負荷行列の各列 i に対して λ_{ij}^2 の分散を計算し、その和をとった式

$$\sum_{i=1}^{\text{因子数}} \left(\sum_{j=1}^{\text{変数の個数}} \lambda_{ij}^4 - \frac{1}{\text{変数の個数}} \left(\sum_{j=1}^{\text{変数の個数}} \lambda_{ij}^2 \right)^2 \right)$$

を最小にするように a_{ij} の値を求めたものがバリマックス回転解です。表すにあたっては分散の公式を用いました。

C. プロマックス回転

バリマックス回転によって得られた因子負荷行列の各要素を4乗して、より因子間の差を強調した行列を**目標行列**（target matrix）といいます。4乗はハイパーパラメータで、実際には3乗を使うこともあります。

Hide

```
# 目標行列を求める
x <- result_varimax$loadings
Q <- x * abs(x)^3     # 符号を保つためにabs^3にした。
print(Q, cutoff = 0)
```

```
Loadings:
       ML1   ML2
麺     0.004 0.116
具     0.010 0.001
スープ 0.349 0.007

               ML1   ML2
SS loadings    0.122 0.014
Proportion Var 0.041 0.005
Cumulative Var 0.041 0.045
```

この目標行列 Q の作り方は行列の掛け算で表すことができない（非線形な変換の）ため、目標行列をそのまま因子負荷行列に用いることは一般にできません。そこで、バリマックス回転解に正則な行列を掛けることで実現できるような行列のなかで、目標行列 Q に近いものを探します。

Hide

```
# バリマックス回転解に掛ける行列の求め方（仕組み）
u <- data.frame(f1_load = x[, 1], f2_load = x[, 2], q = Q[,1])
lm(q ~ 0+f1_load+f2_load, data = u)
```

```
Call:
lm(formula = q ~ 0 + f1_load + f2_load, data = u)

Coefficients:
f1_load  f2_load
 0.4908  -0.2270
```

Hide

```
# バリマックス回転解に掛ける行列の求め方（結果）
U <- lm.fit(x, Q)$coefficients    # Qの各列に対して切片0の線形回帰を行う。
U
```

```
           ML1         ML2
ML1  0.4907770 -0.08575812
ML2 -0.2270104  0.23148546
```

Hide

```
# x%*%Uを因子負荷行列にすると因子得点の標準偏差が1にならない。
d <- diag(solve(t(U)%*%U))    # 因子得点の標準偏差を1にするためのスケーリングを求める。
x %*% U %*% diag(sqrt(d))
```

```
              [,1]        [,2]
麺     -0.02896465 0.654321914
具      0.29617901 0.097324763
スープ  0.81601616 0.008096616
```

以上でプロマックス回転解が得られます。得られた結果を、 fa 関数で得られたプロマックス回転解と比較してみましょう。

Hide

```
# 結果を`fa(rotate="Promax")`と比較してみよう
print(result_promax$loadings, cutoff = 0)
```

```
Loadings:
       ML1    ML2
麺     -0.029  0.654
具      0.296  0.097
スープ  0.816  0.008

                 ML1   ML2
SS loadings    0.754 0.438
Proportion Var 0.251 0.146
Cumulative Var 0.251 0.397
```

第9回 探索的因子分析（後半）

Code

1. 探索的因子分析の仕組み

前回は探索的因子分析に入門しました。今回は、その探索的因子分析の仕組みについて、なるべく具体的に説明していこうと思います。

A. 前準備 : パッケージの読み込み

因子分析は psych パッケージの fa 関数を用いて計算することができるのでした。 psych パッケージを読み込んでおきましょう。

Hide

```
# psychパッケージの読み込み
library(psych)
```

B. 因子負荷量と独自因子の分散の推定

探索的因子分析で推定されるのは、因子負荷量と独自因子の分散の値です。推定の方法には最尤法と最小2乗法があります。最尤法は難しいため、ここでは最小2乗法を説明します。

問題 : データに3変数 x_1, x_2, x_3 が記録されているとします。このデータに対して1因子モデルを想定したとき、因子負荷量 w_1, w_2, w_3 の値、各変数の独自性 $\sigma_1^2, \sigma_2^2, \sigma_3^2$ を最小2乗法で推定してみましょう。

解答 : 最小2乗法では、因子分析モデルの相関行列と標本の相関行列の各要素の差の2乗の和が最も小さくなるように、$w_1, w_2, w_3, \sigma_1^2, \sigma_2^2, \sigma_3^2$ の値を決定します。

具体的に考えてみましょう。今回の場合、因子分析モデルの相関行列は次のようになります。

$$\begin{pmatrix} w_1^2 + \sigma_1^2 & w_1 w_2 & w_1 w_3 \\ w_1 w_2 & w_2^2 + \sigma_2^2 & w_2 w_3 \\ w_1 w_3 & w_2 w_3 & w_3^2 + \sigma_3^2 \end{pmatrix}$$

ここで標本から得られる相関行列が次のようであったとします。

$$\begin{pmatrix} 1 & r_{13} & r_{13} \\ r_{12} & 1 & r_{23} \\ r_{13} & r_{23} & 1 \end{pmatrix}$$

このとき、各要素の差の2乗の和は次のようになります。

$$L(w, \sigma^2) = \sum_{i=1}^{3} (w_i^2 + \sigma_i^2 - 1)^2 + 2 \sum_{1 \leq i < j \leq 3} (w_i w_j - r_{ij})^2$$

この関数 $L(w, \sigma^2)$ を最小にするような w, σ^2 の値を求めるわけです。

なお、σ_1^2, w_1 でこの関数 $L(w, \sigma^2)$ を微分すると

$$\frac{\partial L}{\partial \sigma_1^2} = 2(w_i^2 + \sigma_i^2 - 1)$$
$$\frac{\partial L}{\partial w_1} = 4w_i(w_i^2 + \sigma_i^2 - 1) + w_2(w_1 w_2 - r_{12}) + w_3(w_1 w_3 - r_{13})$$

が得られます。関数 $L(w, \sigma^2)$ を最小にする w, σ^2 の値が $\frac{\partial L}{\partial \sigma_1^2} = 0, \frac{\partial L}{\partial w_1} = 0$ を満たすことに注意すると、w_1, w_2, w_3 の値を

$$\begin{aligned} w_2(w_1 w_2 - r_{12}) + w_3(w_1 w_3 - r_{13}) = 0 \\ w_1(w_1 w_2 - r_{12}) + w_3(w_2 w_3 - r_{23}) = 0 \\ w_1(w_1 w_3 - r_{13}) + w_2(w_2 w_3 - r_{23}) = 0 \end{aligned}$$

を満たすもののなかで関数 $L(w, \sigma^2)$ が最小になるように求め、$w_i^2 + \sigma_i^2 - 1 = 0$ によって σ_i^2 の値を決めればよいことがわかります。

例えば、$r_{12} = r_{23} = r_{32} = 1/2$ の場合、上記の連立方程式をみたす w の値は $(w_1, w_2, w_3) = (0, 0, 0), (1/\sqrt{2}, 1/\sqrt{2}, 1/\sqrt{2}), (-1/\sqrt{2}, -1/\sqrt{2}, -1/\sqrt{2})$ の3通りがあります。関数 $L(w, \sigma^2)$ は $w_1, w_2, w_3) = (1/\sqrt{2}, 1/\sqrt{2}, 1/\sqrt{2}), (-1/\sqrt{2}, -1/\sqrt{2}, -1/\sqrt{2})$ の場合で最小値 0 になるので、解はこの2種類のどちらかを選択すればよいです。

Remark : $w_i^2 + \sigma_i^2 - 1 = 0$ は前回紹介した 共通性=1-独自性 の話に対応しています。■

B. 回転

前回用いたラーメンデータを例に、回転とはどんなことをしている操作かを説明していきましょう。

Hide

```
# データの読み込み
dat <- read.csv("./data/ramen.csv", fileEncoding = "cp932", row.names = "店名")
dat
```

	麺 <int>	具 <int>	スープ <int>
春樹屋	2	4	5
もりまる	1	5	1
ラーメン宇宙	5	3	4
ラーメン三郎	2	2	3
大山屋	3	5	5
ばつね	4	3	2
ラーメン十七番	4	4	3
ぐうのね	1	2	1
きらきら星	3	3	2
麺屋・小次郎	5	5	3

1-10 of 10 rows

サーストンの単純構造

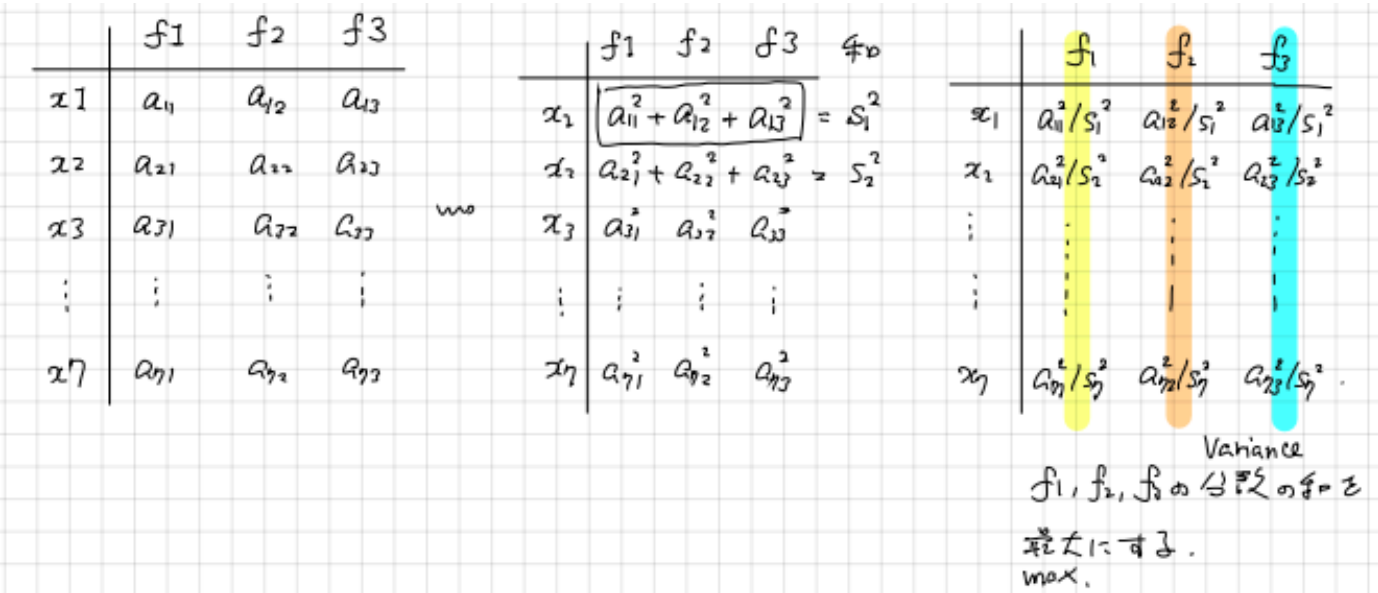
解釈可能な因子負荷量とは何かを考える上で、1947年に提案された**サーストンの単純構造**という基準が参考になります。ここで、m を因子の数とします。

1. 因子負荷行列の各行には、少なくとも一つの0の要素があること。
2. 因子負荷行列の各列には、m 個以上の0の要素があること。
3. 2つの列に注目したとき、一方の列の要素が 0 で他方の列の要素は 0 でない行があること。
4. $m \geq 4$ のとき、2つの列に注目したときどちらの要素も 0 である行があること。
5. 二つの列に注目したとき、どちらの要素も 0 でない行が少ないこと。

以下にのべるバリマックス回転やプロマックス回転は、このサーストンの単純構造をなるべくみたすような因子負荷行列を求める仕組みです。

バリマックス回転解

以下のような条件をみたすように因子負荷行列を求めることで、サーストンの単純構造をなるべく持つような因子負荷行列を求めようという考え方を、**（規準化）バリマックス回転**といいます。



なおバリマックス回転は数式で紹介されることが多いので、ここでもその説明を与えておきます。$\lambda_{ij} = a_{ij}/s_i$ とおいて、因子負荷行列の各列 i に対して λ_{ij}^2 の分散を計算し、その和をとった式

$$\sum_{i=1}^{\text{因子数}} \left(\sum_{j=1}^{\text{変数の個数}} \lambda_{ij}^4 - \frac{1}{\text{変数の個数}} \left(\sum_{j=1}^{\text{変数の個数}} \lambda_{ij}^2 \right)^2 \right)$$

を最小にするように a_{ij} の値を求めたものがバリマックス回転解です。表すにあたっては分散の公式を用いました。

前回は、次のようにしてバリマックス回転解を求めました。

Hide

```
result_varimax <- fa(dat, nfactors = 2, fm = "ml", rotate = "varimax")
print(result_varimax$loadings, cutoff = 0)
```

```
Loadings:
       ML1   ML2
麺     0.248 0.584
具     0.319 0.191
スープ 0.769 0.291

               ML1   ML2
SS loadings    0.754 0.462
Proportion Var 0.251 0.154
Cumulative Var 0.251 0.405
```

プロマックス回転解

バリマックス回転によって得られた因子負荷行列の各要素を4乗して、より因子間の差を強調した行列を**目標行列**（target matrix）といいます。4乗はハイパーパラメータで、実際には3乗を使うこともあります。

Hide

```
# 目標行列を求める
x <- result_varimax$loadings
Q <- x * abs(x)^3    # 符号を保つためにabs^3にした。
print(Q, cutoff = 0)
```

```
Loadings:
       ML1   ML2
麺     0.004 0.116
具     0.010 0.001
スープ 0.349 0.007

               ML1   ML2
SS loadings    0.122 0.014
Proportion Var 0.041 0.005
Cumulative Var 0.041 0.045
```

この目標行列 Q の作り方は行列の掛け算で表すことができない（非線形な変換の）ため、目標行列をそのまま因子負荷行列に用いることは一般にできません。そこで、バリマックス回転解に正則な行列を掛けることで実現できるような行列のなかで、目標行列 Q に近いものを探します。

Hide

```
# バリマックス回転解に掛ける行列の求め方（仕組み）
u <- data.frame(f1_load = x[, 1], f2_load = x[, 2], q = Q[,1])
lm(q ~ 0+f1_load+f2_load, data = u)
```

```
Call:
lm(formula = q ~ 0 + f1_load + f2_load, data = u)

Coefficients:
f1_load  f2_load
 0.4908  -0.2270
```

Hide

```
# バリマックス回転解に掛ける行列の求め方（結果）
U <- lm.fit(x, Q)$coefficients     # Qの各列に対して切片0の線形回帰を行う。
U
```

```
           ML1         ML2
ML1  0.4907770 -0.08575812
ML2 -0.2270104  0.23148546
```

Hide

```
# x%*%Uを因子負荷行列にすると因子得点の標準偏差が1にならない。
d <- diag(solve(t(U)%*%U))     # 因子得点の標準偏差を1にするためのスケーリングを求める。
x %*% U %*% diag(sqrt(d))
```

```
             [,1]        [,2]
麺     -0.02896465 0.654321914
具      0.29617901 0.097324763
スープ  0.81601616 0.008096616
```

以上でプロマックス回転解が得られます。得られた結果を、 fa 関数で得られるプロマックス回転解と比較してみましょう。

Hide

```
result_promax <- fa(dat, nfactors = 2, fm = "ml", rotate = "Promax")
print(result_promax$loadings, cutoff = 0)
```

```

Loadings:
       ML1    ML2
麺     -0.029  0.654
具      0.296  0.097
スープ  0.816  0.008

                 ML1   ML2
SS loadings    0.754 0.438
Proportion Var 0.251 0.146
Cumulative Var 0.251 0.397
```

2. mtcarsデータセットによる演習

問題： mtcars_numeric.csv は、1974年の雑誌 “Motor Trend US” から32台の自動車モデルについて、

- mpg ：燃費
- cyl ：シリンダーの数
- disp ：排気量
- hp ：馬力
- drat ：rear axle 比
- wt ：重量
- qsec ：ドラッグレースのタイム
- gear ：ギア数
- carb ：キャブレーターの数

を抜粋し、記録したデータです。

Hide

```
dat <- read.csv("./data/mtcars_numeric.csv",
                fileEncoding = "utf-8",
                row.names = "model")
head(dat, n = 5)
```

	mpg	cyl	disp	hp	drat	wt	qsec	gear	carb
	<dbl>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<int>	<int>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	3	2

5 rows

探索的因子分析を用いて、各自動車モデルの特徴や自動車モデルの間の類似性を把握してみてください。

解答：まずはデータの各変数間の相関関係を確認します。相関係数を確認する上で、ヒートマップを用いると便利です。

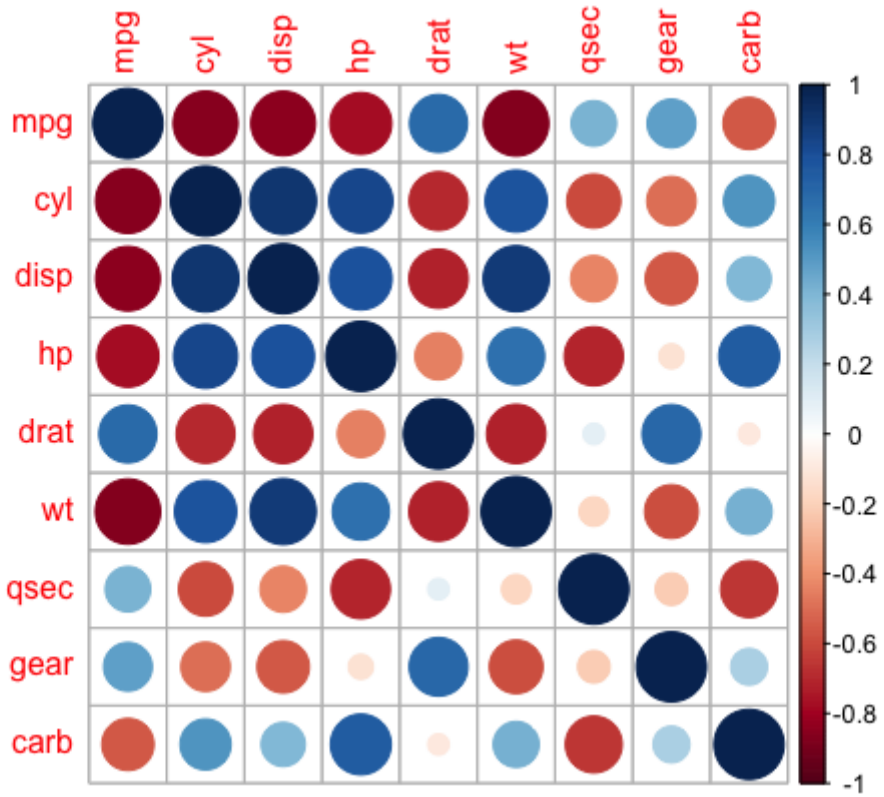
Hide

```
library(corrplot)
```

```
corrplot 0.88 loaded
```

Hide

```
cormat <- cor(dat)     # 相関行列の計算
corrplot(cormat)     # ヒートマップの作成
```



相関係数のパターンをみて、 mpg から wt および gear の7変数が一つに、 qsec と carb の2変数が一つにまとまる2因子モデルで、因子間には相関があるのではないかと考えました。

そこで、最尤法・プロマックス回転による2因子モデルを行ってみましょう。

Hide

```
# 2因子モデル
result <- fa(dat, nfactors = 2, fm = "ml", rotate = "Promax")
result <- fa.sort(result)    # 因子負荷量の絶対値が高い順にソート
result$loadings    # サーストンの単純構造が得られている
```

```
Loadings:
     ML1    ML2
gear -0.984  0.535
drat -0.874  0.104
wt    0.815  0.182
disp  0.791  0.316
mpg  -0.693 -0.386
cyl   0.688  0.445
carb -0.128  0.911
qsec  0.109 -0.848
hp    0.324  0.790

                ML1   ML2
SS loadings    4.109 2.950
Proportion Var 0.457 0.328
Cumulative Var 0.457 0.784
```

変数 hp については想定とは異なりましたが、第1因子が正に大きいほど操作が煩雑で本格的な自動車、第2因子が正に大きいほどスピードが出せそうな自動車であることがわかります。

因子であまり説明されていないような変数は、除いてから改めて探索的因子分析を行なった方が良いかもしれません。そこで、変数の共通性を確認しておきましょう。今回は、共通性の低い変数は確認できませんでした。

Hide

```
result$communalities
```

```
      mpg       cyl      disp        hp      drat        wt      qsec 
0.8314176 0.9032550 0.9139571 0.9213768 0.7062105 0.8106713 0.6616233 
     gear      carb 
0.8563825 0.7575501 
```

なお、得られた因子の間には正の相関が確認できます。

Hide

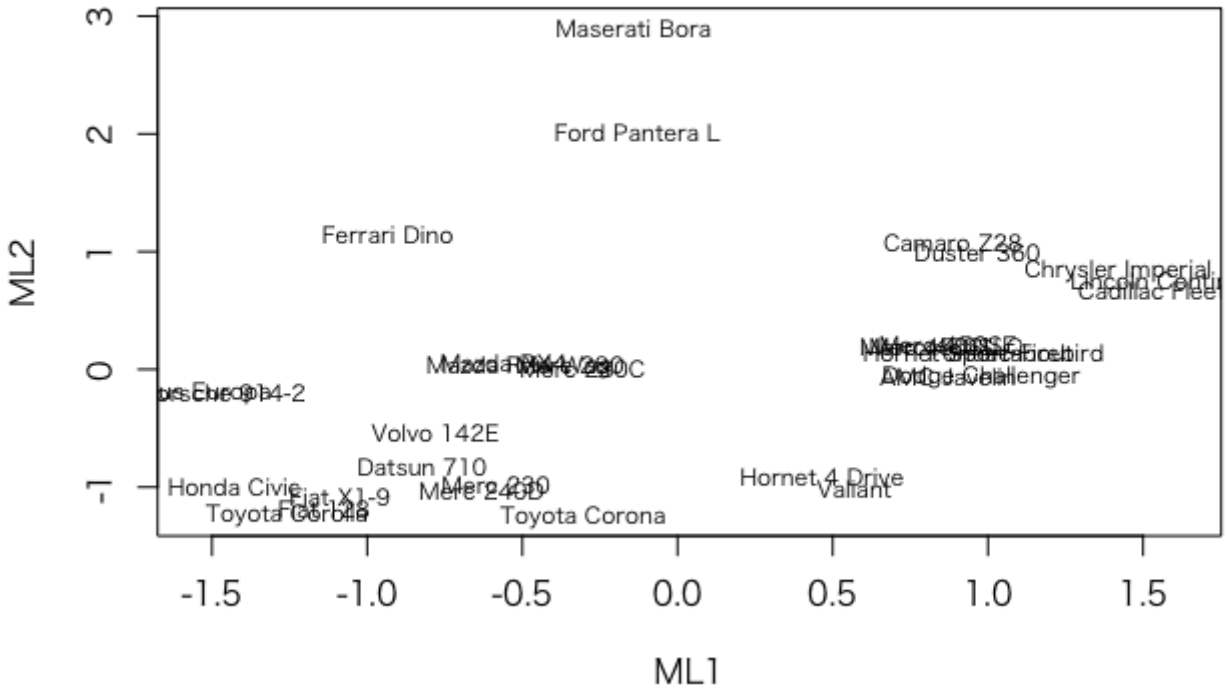
```
# 因子間相関
result$Phi
```

```
          ML1       ML2
ML1 1.0000000 0.3784488
ML2 0.3784488 1.0000000
```

各自動車の特徴を抑えるために、因子得点を散布図で可視化しておきましょう。例えば、とにかくスピードを求めている人にはMaserati BoraやFord Pantera Lを勧めるとよいだろうことがわかります。また、スピードをあまり求めず、操作性の難しくない自動車を求めている人には Honda CivicやToyota Corollaを提案するとよいでしょう。

Hide

```
# 因子得点の散布図
par(family = "ヒラギノ角ゴシック W3")   # Macユーザーのみ
plot(result$scores, type = "n")
text(result$scores, labels=rownames(dat), cex = 0.7)
```


Maserati Bora
Ford Pantera L
Ferrari Dino
Camaro Z28
Duster 360
Chrysler Imperial
Cadillac Fleet
Volvo 142E
Datsun 710
Honda Civic
Toyota Corona
Hornet 4 Drive
Valiant
ML2
ML1
3
2
1
0
-1
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5

第10回 k-means法

Code

1. 概要

k-means法は、データを複数個のクラスターに分割する方法の一つです。ここでは、k-means法のデモ・問題設定と仕組み・長所と注意点についてそれぞれ説明します。

2. k-means法のデモ

各国の乳製品の消費量のデータを持ちいて、消費量による国のクラスタリングをやってみましょう。

Hide

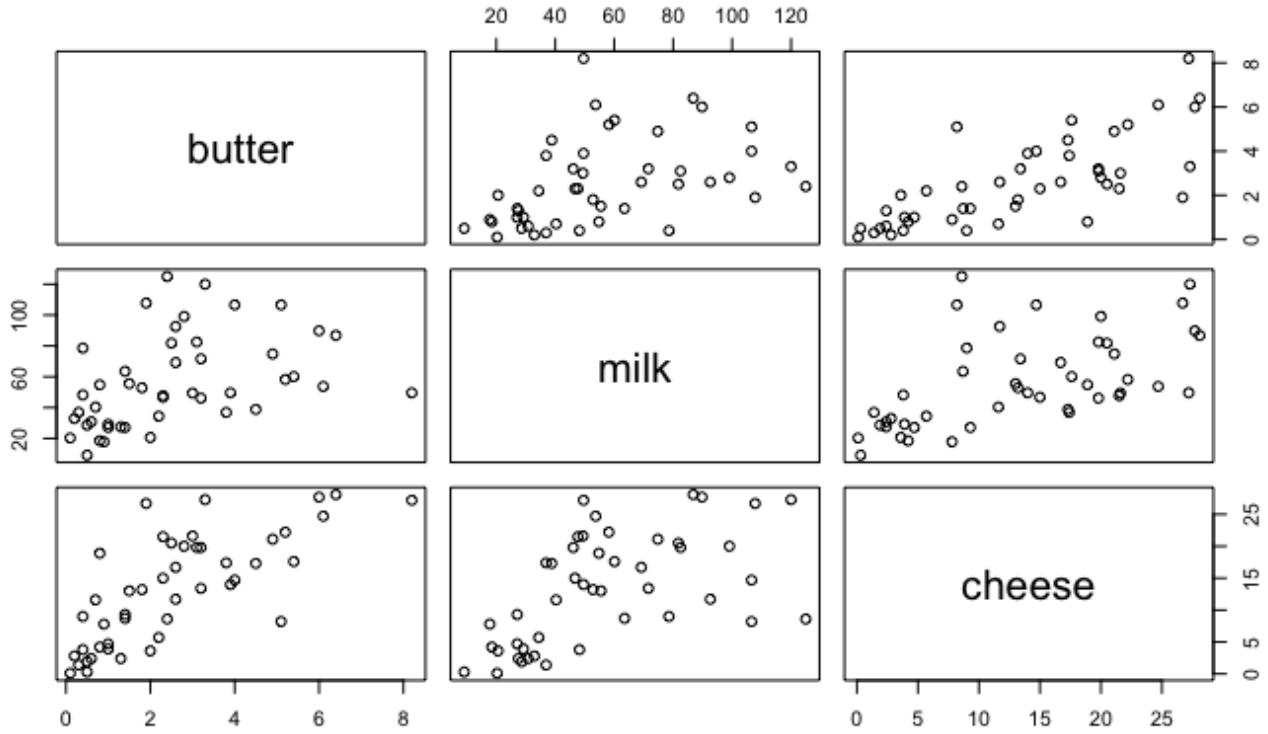
```
dat <- read.csv("./data/dairy_products.csv",
                fileEncoding = "utf-8",
                row.names = "country")
head(dat, n = 5)
```

	butter <dbl>	milk <dbl>	cheese <dbl>
中国	0.1	20.3	0.1
日本	0.6	30.8	2.4
韓国	0.2	32.9	2.8
イラン	1.0	27.1	4.7
トルコ	0.9	17.8	7.8

5 rows

Hide

```
plot(dat)
```



k-means法を行う際には、事前に正規化をおこないます。理由は5節で説明します。今回は正規化として標準化を選択しました。

Hide

```
dat_scaled <- scale(dat)
```

k-means法は、 kmeans 関数に以下の値を指定することで計算できます。

- x ：データ
- centers ：クラスター数
- iter.max ：繰り返しの最大数

今回は、4つのクラスターに分割してみます。

Hide

```
# k-means法の計算
result <- kmeans(x = dat_scaled, centers = 3, iter.max = 100)
result$centers
```

```
      butter        milk      cheese
1 -0.8046707 -0.68201079 -0.7930154
2  1.1583954  0.01339385  0.9687856
3  0.2403286  1.34819249  0.4411023
```

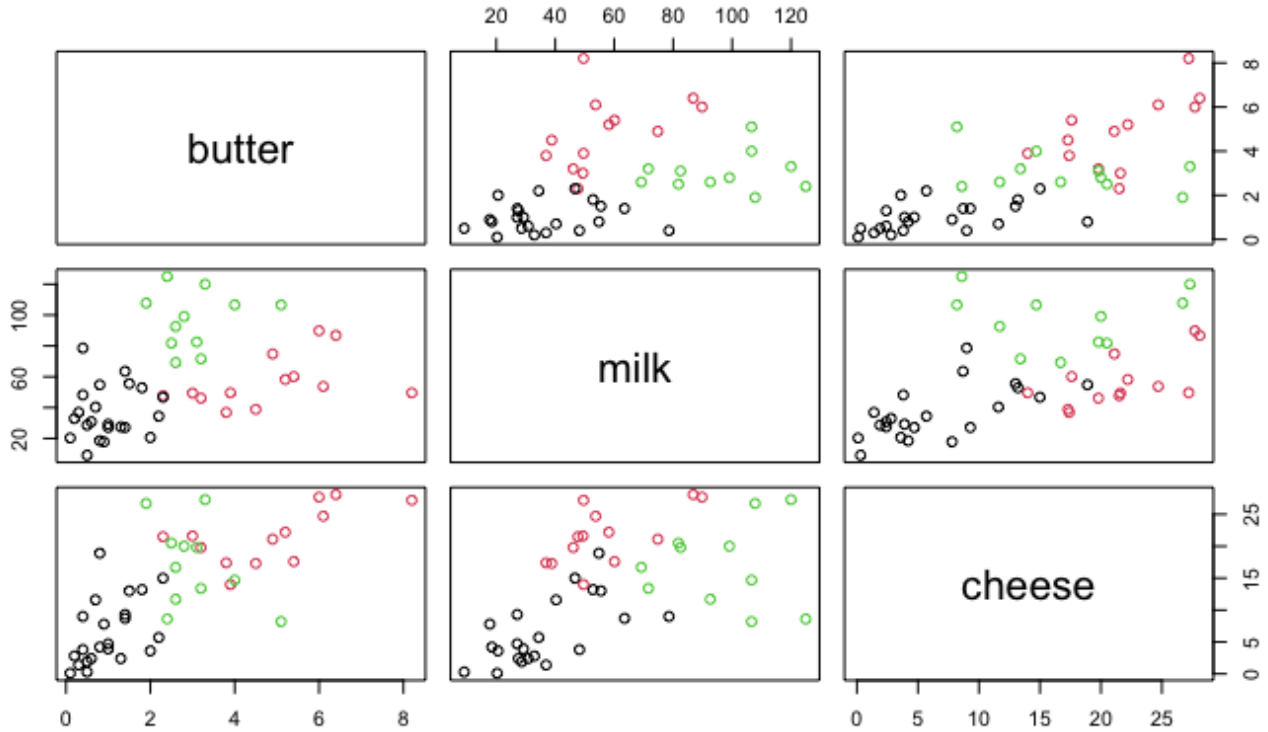
k-means法を行うと、それぞれのクラスターの中心と各データ点がどのクラスターに所属するかを求めることができます。クラスター中心は、得られたクラスターを解釈するヒントになります。今回の場合、

cluster 1 : 乳製品をあまり消費しない国 cluster 2 : butter, cheeseよりmilkでしょ！ cluster 3 : milkよりbutter, cheeseでしょ！

と解釈できます。また、クラスターは次のように分布しています。

Hide

```
# クラスターの可視化
plot(dat, col = result$cluster)
```



3. k-means法の問題設定と仕組み

A. 問題設定

数式による問題設定の説明

k-means法は、1956年にSteinhausが次のような問題を考えたことに始まります。

問題 : K をある正の整数に決めてください。d個の変数で表現される N 個の点の集合

$$D = \{x_n = (x_{n1}, \cdots, x_{nd}) \mid n = 1, 2, \cdots, N\}$$

を、以下の関数が最小になるように k 個の交わりのない集合 $C_1, \cdots, C_K$ に分割するにはどうすれば良いでしょうか。

$$L(C_1, \cdots, C_k) = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - c_k\|_2^2$$

ここで、$\|v\|_2 = \sqrt{v_1^2 + \cdots + v_d^2}$（いわゆるユークリッド距離）、$c_k = \frac{1}{\#C_k} \sum_{x \in C_k} x$（集合$C_k$に属する点の重心）です。

術語 : $C_1, \cdots, C_K$ をクラスター、c_k をクラスター C_k のcentroidと言います。

数式の解釈

問題の本質は N 個のデータ点 $x_1, \cdots, x_N$ が手元にあった時、これを自分で予め決めた個数 K のクラスター $C_1, \cdots, C_K$ に分割したいというものです。さて、どのようなクラスターを作りたいかは関数 $L(C_1, \cdots, C_K)$ に表現されています。注目してほしいのは、

$$\sum_{x \in C_k} \|x - c_k\|_2^2$$

という式です。この式は、クラスター C_k に所属するデータ点がその重心からどれだけ離れているかを表している式に読めます。そのクラスターが小さく纏まっているほど、この値は小さくなることが期待されるでしょう。いわゆるクラスター C_k の「分散」です。要するに、この問題はクラスターの分散の合計が最も小さくなるように、与えられた N 個のデータ点を K 個のクラスターに分割しなさいという問題を考えていることがわかります。

B. 得られるもの

k-means法の課題設定を解いた結果得られるものとして、

- クラスター中心
- 各データ点が所属するクラスターのid

の2つがあります。クラスターを作ったらそのクラスターの意味を解釈することが大切ですが、クラスター中心はこの解釈のヒントになります。

C. 仕組み

k-means法の課題設定をコンピュータで正確に解くことは、とても時間のかかる作業だということが知られています。そこでLloyd(1957)とForgy(1965)は独立に、この問題の近似的な解を導く以下のようなアルゴリズムを提案しました。

【Forgy-Lloyd algorithm】

1. **クラスターの初期値** : 各データポイントを何らかの方法でクラスターに割り振る。
2. 以下の操作を繰り返す。
    - **centoroidの更新** : 各クラスターに所属する点の重心を計算する。
    - **クラスターの再配置** : データポイントそれぞれについて、最近傍のcentroidを探し、対応するクラスターに再配置する。

Remark : R言語では、Hartigan-Wong algorithmがデフォルトになっている。こちらのアルゴリズムで得られる解の集合の方が、Forgy-Lloyd algorithmで得られる解の集合より真に小さい（包含されている）ことが知られており、砕けた表現をするなら、より正確に解が得られやすいからである。なお、 algorithm 引数に Forgy か Lloyd を渡すことで、Forgy-Lloyd algorithmを用いることができる。 Forgy , Lloyd と区別されているが、どちらを選択しても全く同じアルゴリズムが動く。■

4. Forgy-Lloyd algorithmのデモ

試しにこのアルゴリズムによるクラスターの更新の様子を見てみましょう。まず、デモデータを次のように生成します。可視化した結果から確認できますが、今回は2次元の散布図上で2群（左下と右上）に分かれているデータを準備しました。

Hide

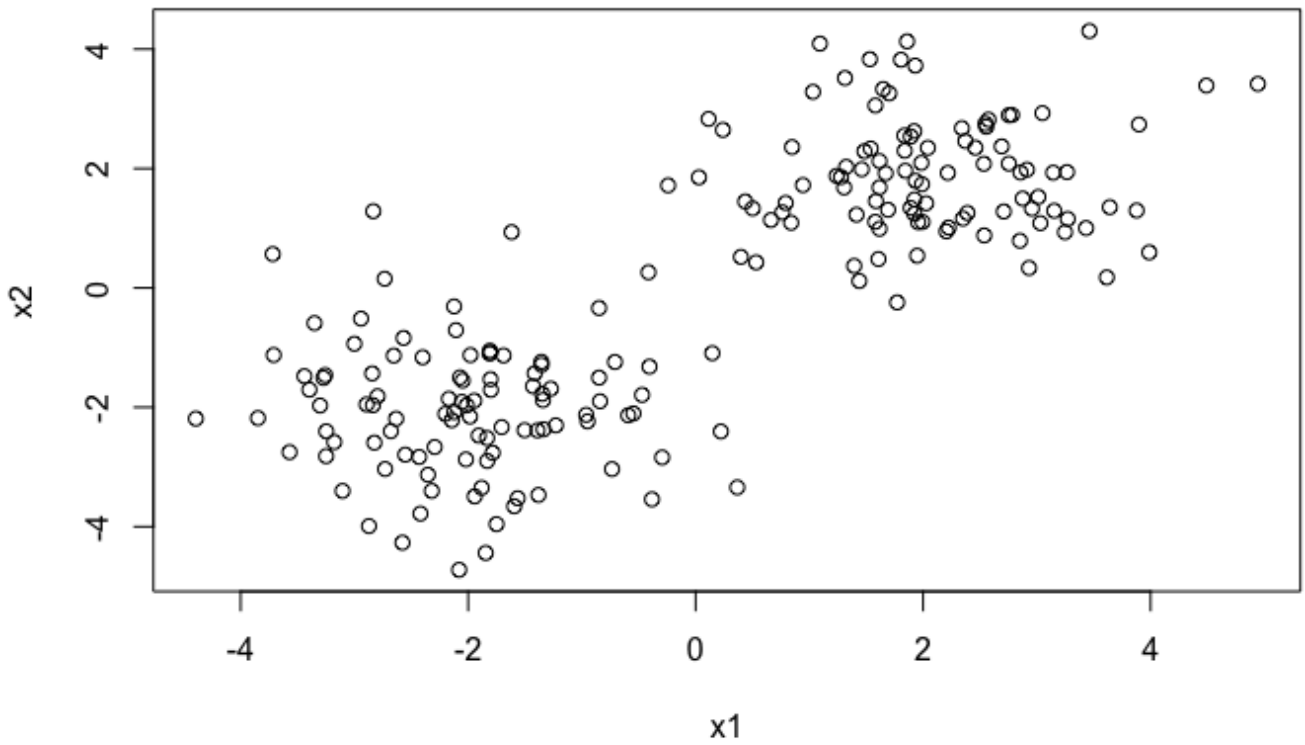
```
# デモデータの生成
library(MASS)
cluster_1 <- mvrnorm(n = 100, mu = c(-2, -2), Sigma = diag(c(1, 1)))
cluster_2 <- mvrnorm(n = 100, mu = c(2, 2), Sigma = diag(c(1, 1)))
X <- rbind(cluster_1, cluster_2)
dat <- data.frame(x1 = X[ , 1], x2 = X[, 2])
head(dat, n = 5)
```

	x1 <dbl>	**x2** <dbl>
1	-1.3422282	-1.8800616
2	-2.9416343	-0.5146741
3	-2.8716507	-3.9900049
4	0.1466697	-1.0934696
5	-2.0191856	-2.8731611

5 rows

Hide

```
plot(dat)
```

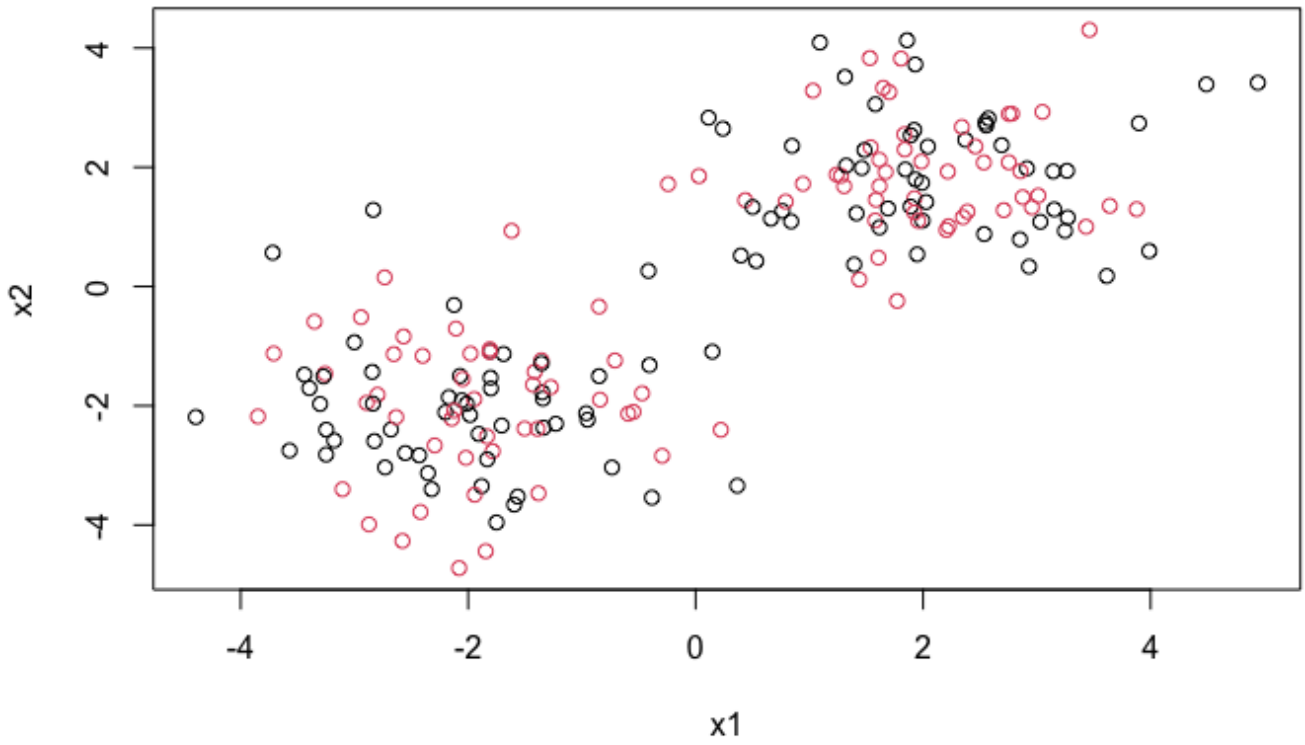


このデータを、クラスター数K=2のk-means法のアルゴリズムによってクラスタリングする様子を見ていきます。

step 1. クラスターの初期値 : 各データポイントを何らかの方法でクラスターに割り振る。今回は、各クラスターに対して等確率に割り振ることにします。

Hide

```
y <- sample(x = c(1, 2), size = 200, replace = TRUE)
plot(dat, col = y)
```



step 2. centroidの更新 : 各クラスターの重心を計算する。

Hide

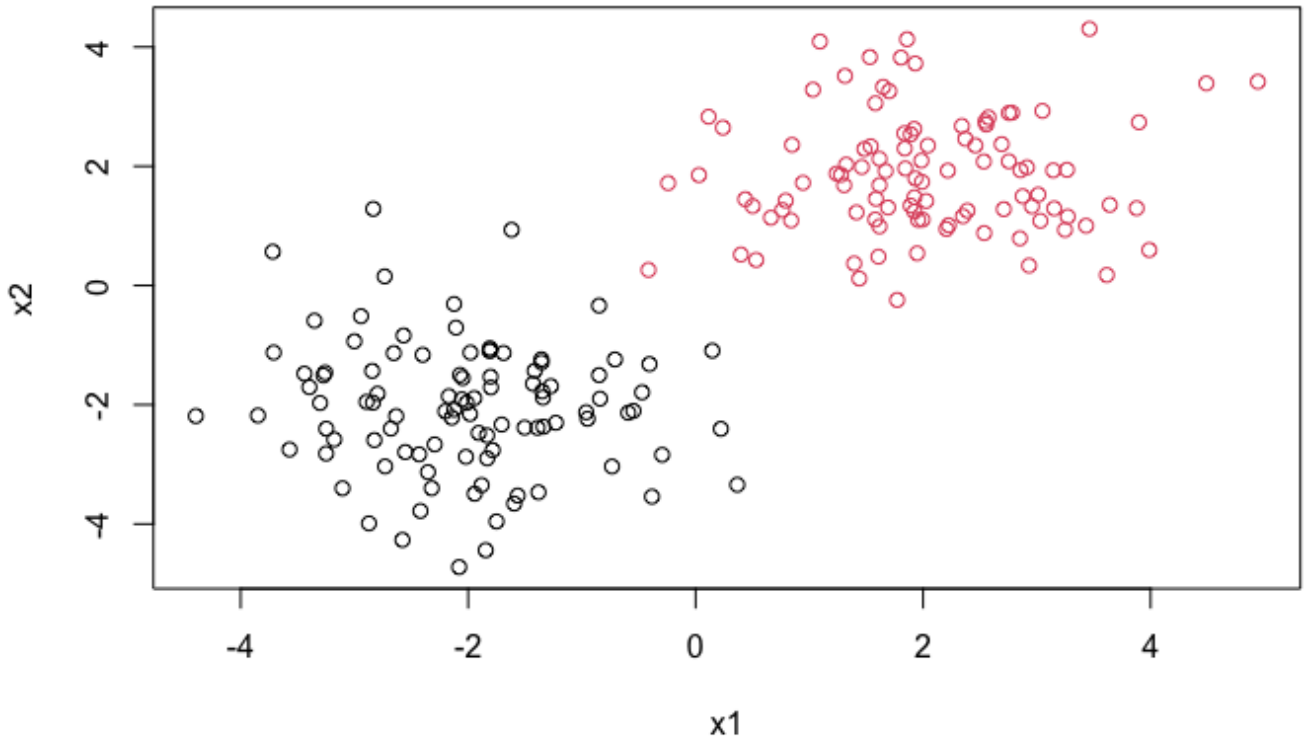
```
center_1 <- colMeans(dat[y==1, ])
center_2 <- colMeans(dat[y==2, ])
c(center_1, center_2)
```

```
         x1          x2          x1          x2
-0.022538714 -0.130764046  0.065717687 -0.009468612
```

step 3. クラスターの再配置 : データポイントそれぞれについて、最近傍のcentroidを探し、対応するクラスターに再配置する。

Hide

```
dist_1 <- rowSums(sweep(dat, MARGIN = 2, center_1)^2)
dist_2 <- rowSums(sweep(dat, MARGIN = 2, center_2)^2)
new_cluster_id <- 1*(dist_1 < dist_2) + 2*(dist_1 >= dist_2)
plot(dat, col = new_cluster_id)
```



5. クラスターの初期値の決め方

クラスターの初期値は、解の精度に影響を与えることが知られています。例えば、クラスターの初期値の決め方として最初に思いつくのは、各データ点をランダムにクラスターに割り振る方法でしょう。しかしこの初期値を採用した場合、Forgy-Lloyd algorithmによって得られる解は、関数

$$L(C_1, \cdots, C_K) = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - c_k\|_2^2$$

の値が真の最適な分割を与えるクラスターによる値よりいくらでも悪くなり得る例があることが知られています。以下のデータセットに対して、繰り返しk-means法を実行してみてください。

Hide

```
# デモデータの生成
cluster1 <- mvrnorm(n = 100, mu = c(6, 10), Sigma = diag(c(2, 2)))
cluster2 <- mvrnorm(n = 100, mu = c(13, 10), Sigma = diag(c(2, 2)))
cluster3 <- mvrnorm(n = 100, mu = c(10, -10), Sigma = diag(c(2, 2)))
cluster4 <- mvrnorm(n = 100, mu = c(-10, -10), Sigma = diag(c(2, 2)))
cluster5 <- mvrnorm(n = 50, mu = c(-10, 10), Sigma = diag(c(1, 1)))
X <- rbind(cluster1, cluster2, cluster3, cluster4, cluster5)
dat <- data.frame(x1 = X[ , 1], x2 = X[ , 2])
head(dat, n = 5)
```

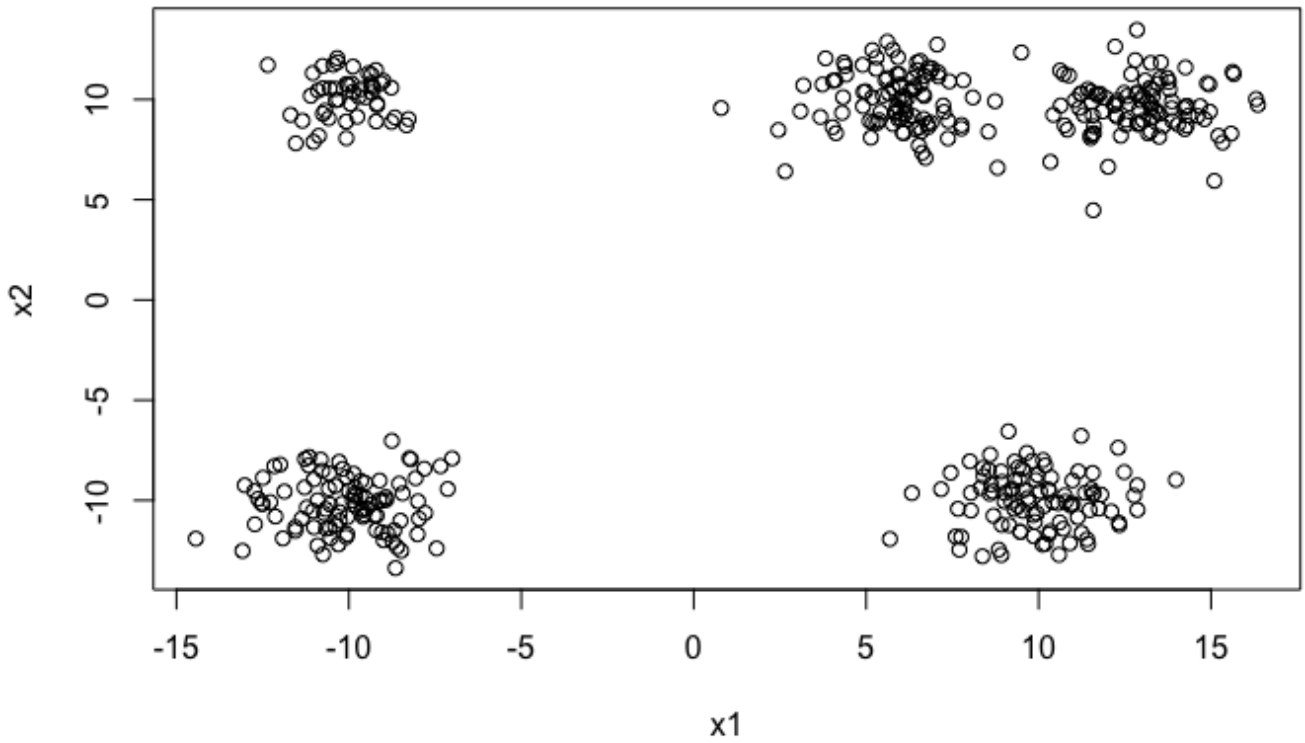
	x1 <dbl>	x2 <dbl>
1	5.610039	11.082842
2	4.908799	9.659212

	x1 <dbl>	x2 <dbl>
3	5.377743	8.966763
4	5.963044	9.576589
5	2.650137	6.411516

5 rows

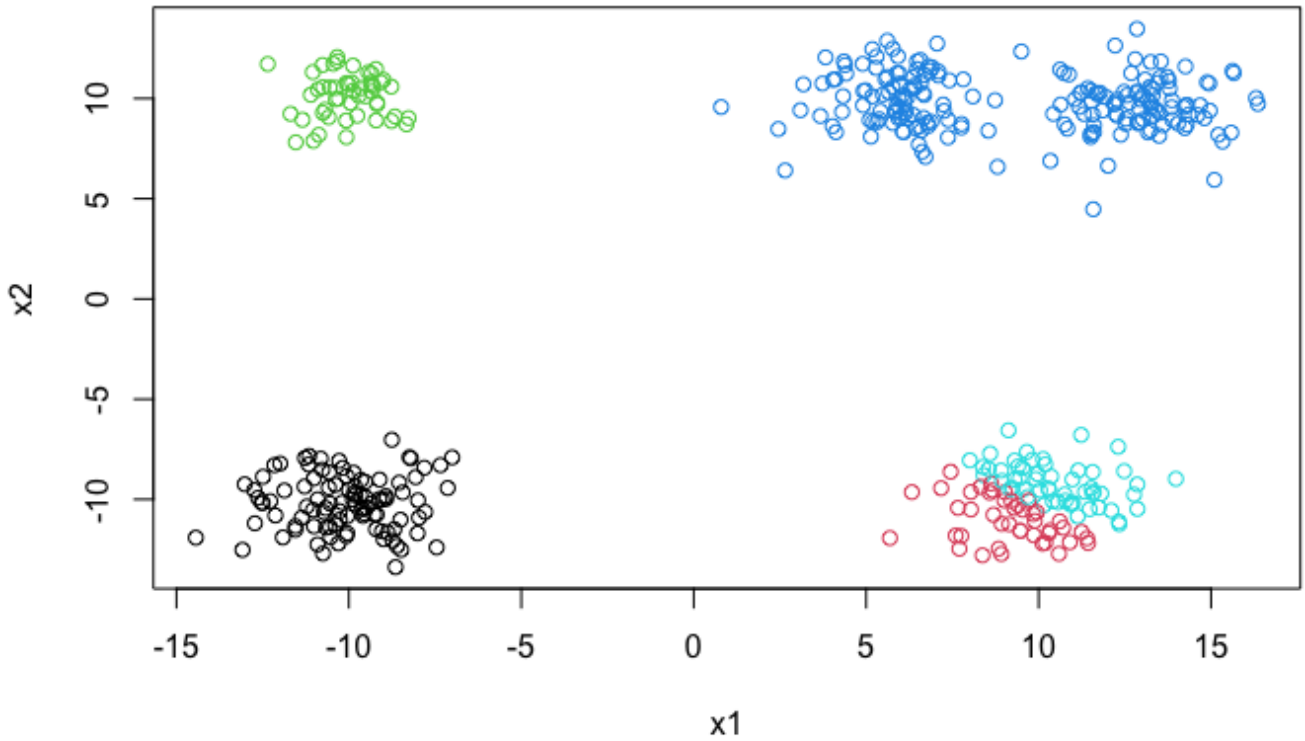
Hide

```
# 散布図 : 5つのクラスターが確認できる。
plot(dat)
```



Hide

```
# 20回くらい何度も実行して、結果を比較してみてください。
result <- kmeans(x = dat, centers = 5, iter.max = 100)
plot(dat, col = result$cluster)
```



そこで、2007年にArthurとVassilvistskiiは、この問題を緩和する初期値の決め方として**k-means++**を発表しました。

【k-means++ algorithm】

1. c_1 はデータ点の中から等確率でランダムに選ぶ。
2. 以下の計算を繰り返す。（tはステップ数とする。）
    - 各データ点に対して、$D(x) := \min_{k=1,\cdots,t} \|x - c_k\|$ を計算する。
    - c_{t+1} を確率 $D(x)^2 / \sum_x D(x)^2$ でランダムに選ぶ。

定理 : 関数 $L(C_1, \cdots, C_K)$ の最小値を L_{opt} , k-means++を初期値に採用したForgy-Lloyd algorithmによって得られるクラスターが与える関数 L の値を L_{++} と書くことにしましょう。このとき、以下のような精度の評価が成り立ちます。

$$\mathbb{E}[L_{++}] \leq 8(\log K + 2)L_{opt}$$

k-means++を自分で実装してみることにしましょう。実装してみたら、先程のデモデータに対するクラスターの結果がどうなるかを確認します。

Hide

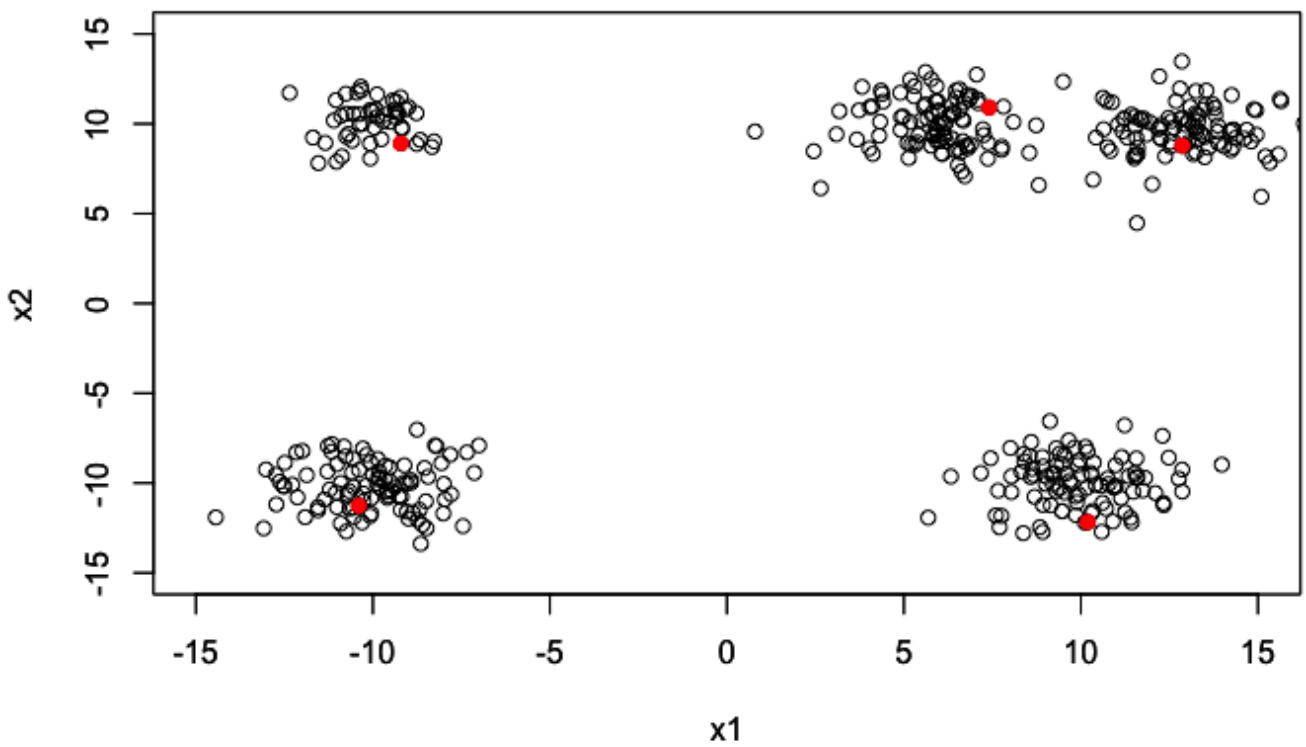
```
# kmeans++の実装
init_kmeanspp <- function(x, centers) {
  x <- as.matrix(x)
  dist_centers <- matrix(Inf, nrow = nrow(x), ncol = centers-1)

  center_pts <- sample(x = 1:nrow(x), size = 1)    # ランダムに一点選ぶ。

  for (i in 1:(centers-1)) {
    dist_centers[ , i] <- rowSums(sweep(x, 2, x[center_pts[i], ])^2)
    min_dist <- apply(dist_centers, MARGIN = 1, FUN = min)
    prob <- min_dist / sum(min_dist)
    center_pts[i+1] <- sample(x = 1:nrow(x), size = 1, prob = prob)
  }

  return(center_pts)
}
```

Hide

```
# どんなクラスターの初期値が選ばれているかを確認する。（複数回）
initial_centers <- init_kmeanspp(x = dat, centers = 5)
plot(dat, xlim = c(-15, 15), ylim = c(-15, 15))
par(new = TRUE)
```

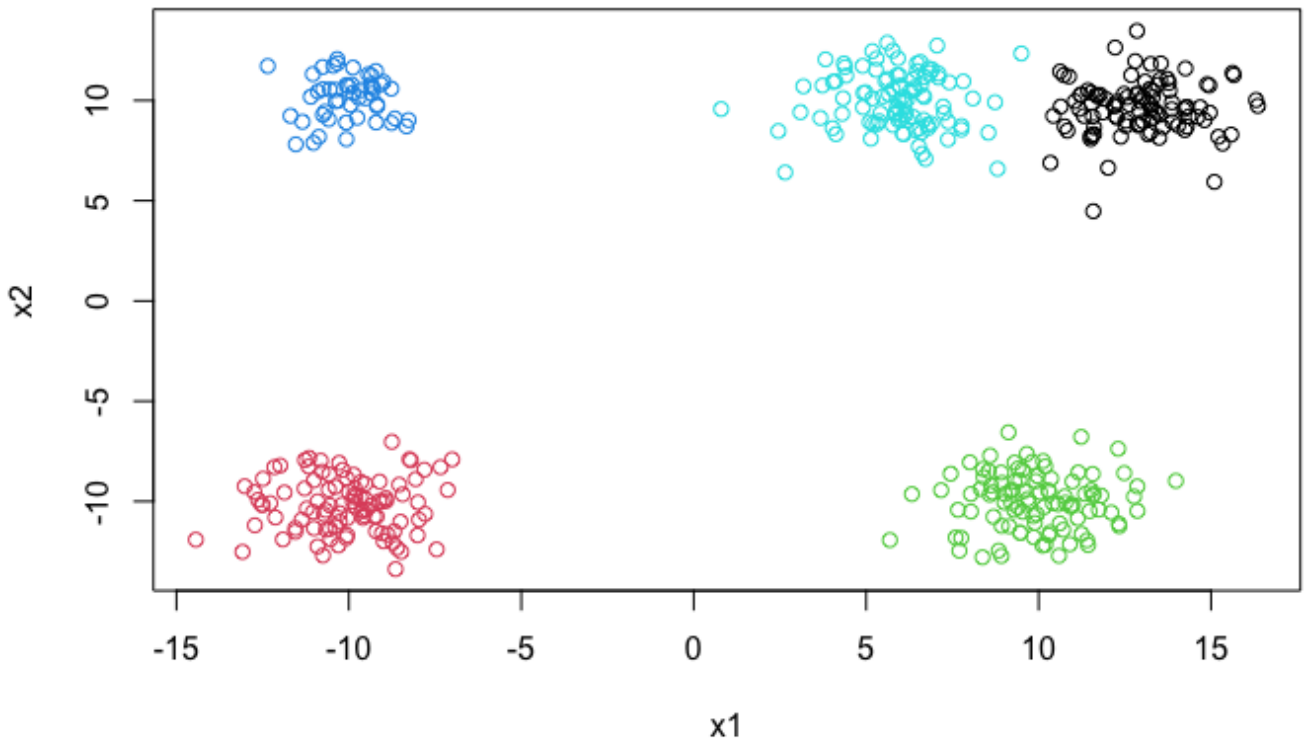
Hide

```
plot(dat[initial_centers, ], xlim = c(-15, 15), ylim = c(-15, 15),
     col = "red", pch = 19)
```



Hide

```
# 同様に20回くらい試してみてください。
initial_centers <- init_kmeanspp(x = dat, centers = 5)
result <- kmeans(dat, centers = dat[initial_centers, ], iter.max = 100)
plot(dat, col = result$cluster)
```



上記の実験を20回ほど試してみると、ランダムに初期値を決める場合より、k-means++のほうが適切なクラスターが行われやすい様子を確認できるのではないでしょうか。

5. k-means法の長所と注意点

k-means法は他のクラスタリング手法に比べて、データポイントの数に対してスケールする（計算量のオーダーが低い）という性質が知られています。また、アイディアの分かりやすさもあって、他のクラスタリング手法と比較して最初に試される手法です。

一方で、k-means法には2つの注意点があります。

1. データを正規化しておく必要性
2. 各クラスターのサンプルサイズの不均一性に対する弱さ

A. 正規化の必要性

k-means法のアルゴリズムは距離に依存しているので、ある変数の値が10倍されると、クラスタリングの結果がその変数に引っ張られてしまいます。実際に、以下のスクリプトを実行してみてください。

Hide

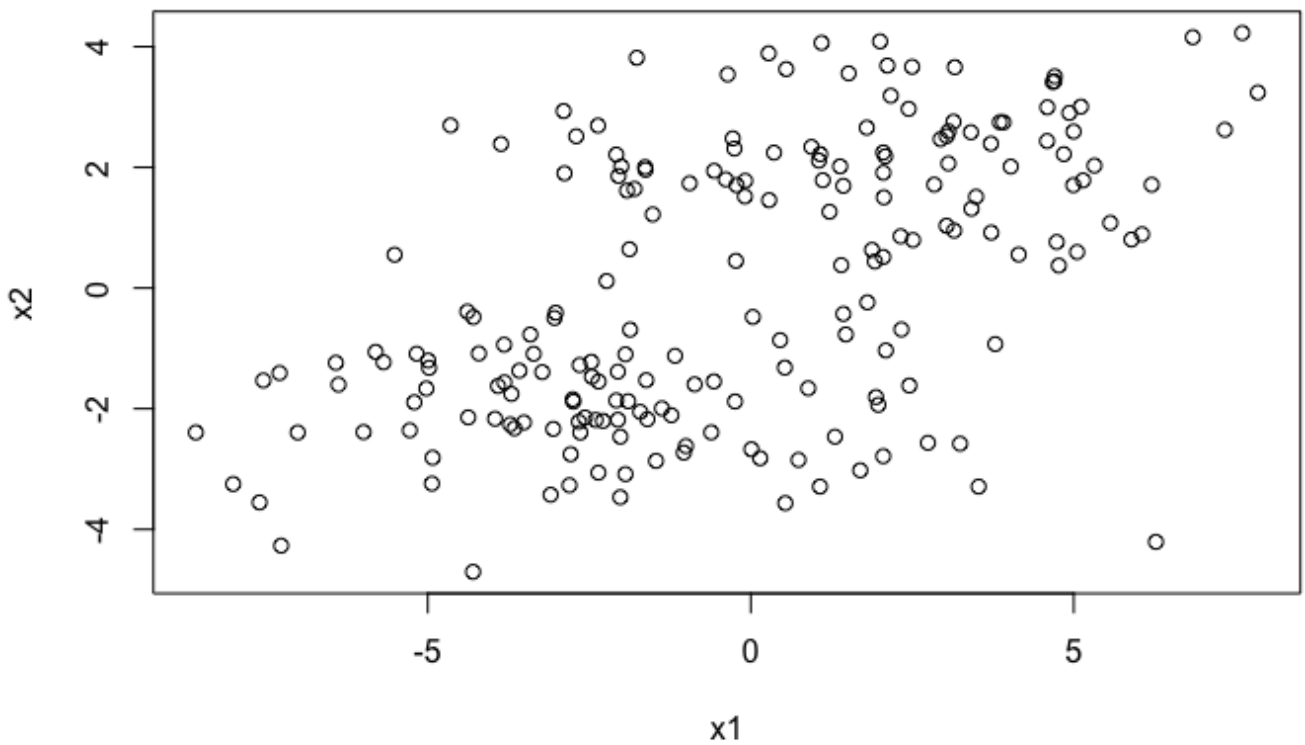
```
# デモデータの生成
cluster_1 <- mvrnorm(n = 100, mu = c(-2, -2), Sigma = diag(c(10, 1)))
cluster_2 <- mvrnorm(n = 100, mu = c(2, 2), Sigma = diag(c(10, 1)))
X <- rbind(cluster_1, cluster_2)
dat <- data.frame(x1 = X[ , 1], x2 = X[, 2])
head(dat, n = 5)
```

	x1 <dbl>	x2 <dbl>
1	-3.098442	-3.4257766
2	1.973203	-1.9363948
3	-1.867907	-0.6898445
4	-5.685633	-1.2301688
5	-7.604448	-3.5528874

5 rows

Hide

```
plot(dat)
```



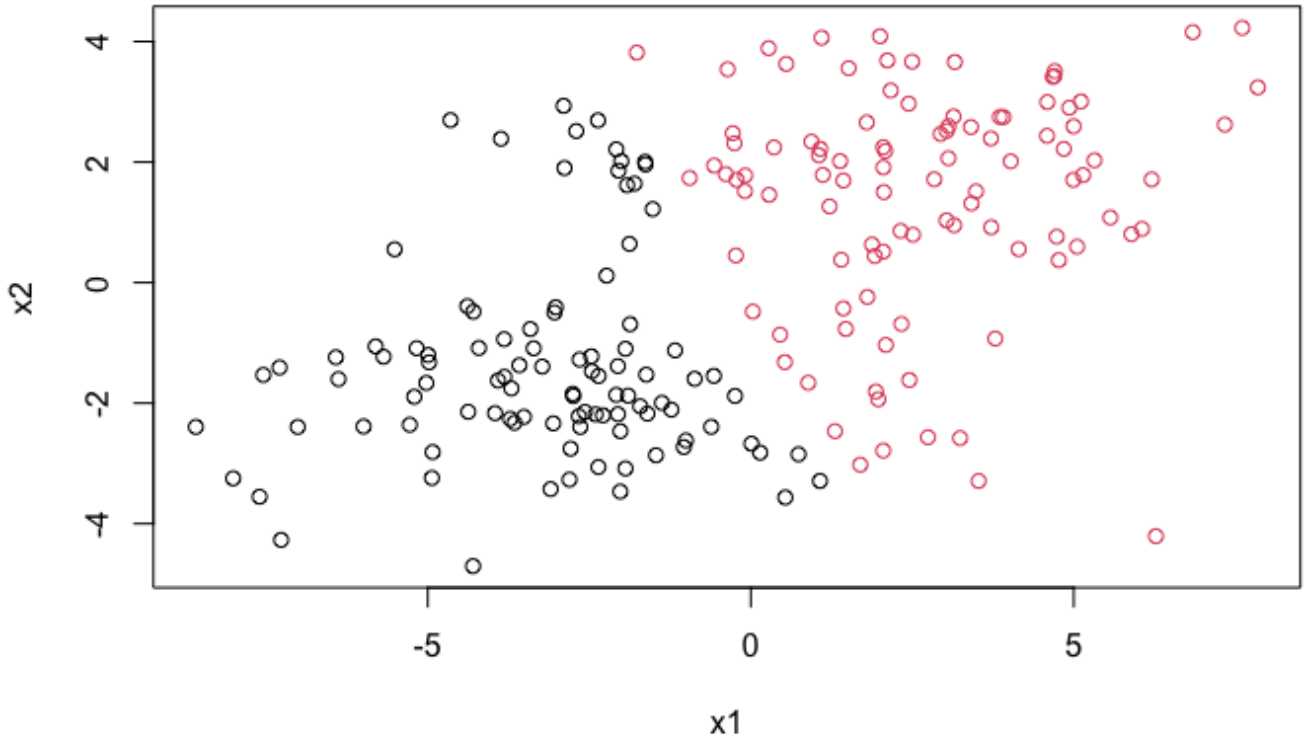
Hide

```
# k-means法によるクラスタリング
result <- kmeans(x = dat, centers = 2, iter.max = 100)
result$centers
```

```
        x1        x2
1 -3.142386 -1.366410
2  2.674283  1.400258
```

Hide

```
# クラスタリングの結果の可視化
plot(dat, col = result$cluster)
```



B. サンプルサイズの不均一性

また、k-means法は各データポイントのサンプルサイズがおおよそ同じであるという仮定を暗に置いています。実際、以下のスクリプトを実行してみてください。

Hide

```
# デモデータの生成
cluster_1 <- mvrnorm(n = 200, mu = c(-2, -2), Sigma = diag(c(1, 1)))
cluster_2 <- mvrnorm(n = 10, mu = c(2, 2), Sigma = diag(c(1, 1)))
cluster_3 <- mvrnorm(n = 20, mu = c(2, -2), Sigma = diag(c(1, 1)))
X <- rbind(cluster_1, cluster_2, cluster_3)
dat <- data.frame(x1 = X[ , 1], x2 = X[, 2])
head(dat, n = 5)
```

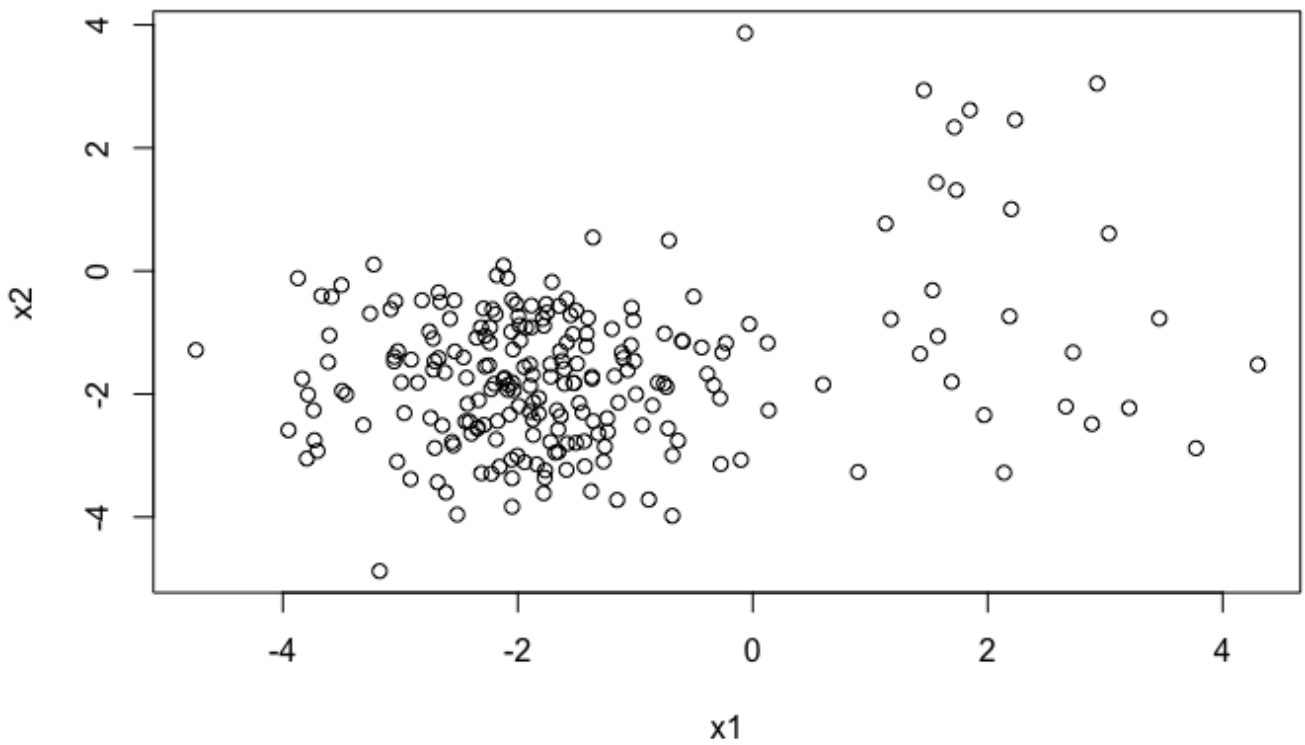
	x1 <dbl>	**x2** <dbl>
1	-1.2590592	-2.85520688
2	-2.2474017	-1.53237654
3	-2.1214116	0.08793986

	x1 <dbl>	x2 <dbl>
4	-0.7209452	-2.55823774
5	-3.7387405	-2.26274485

5 rows

Hide

```
plot(dat)
```



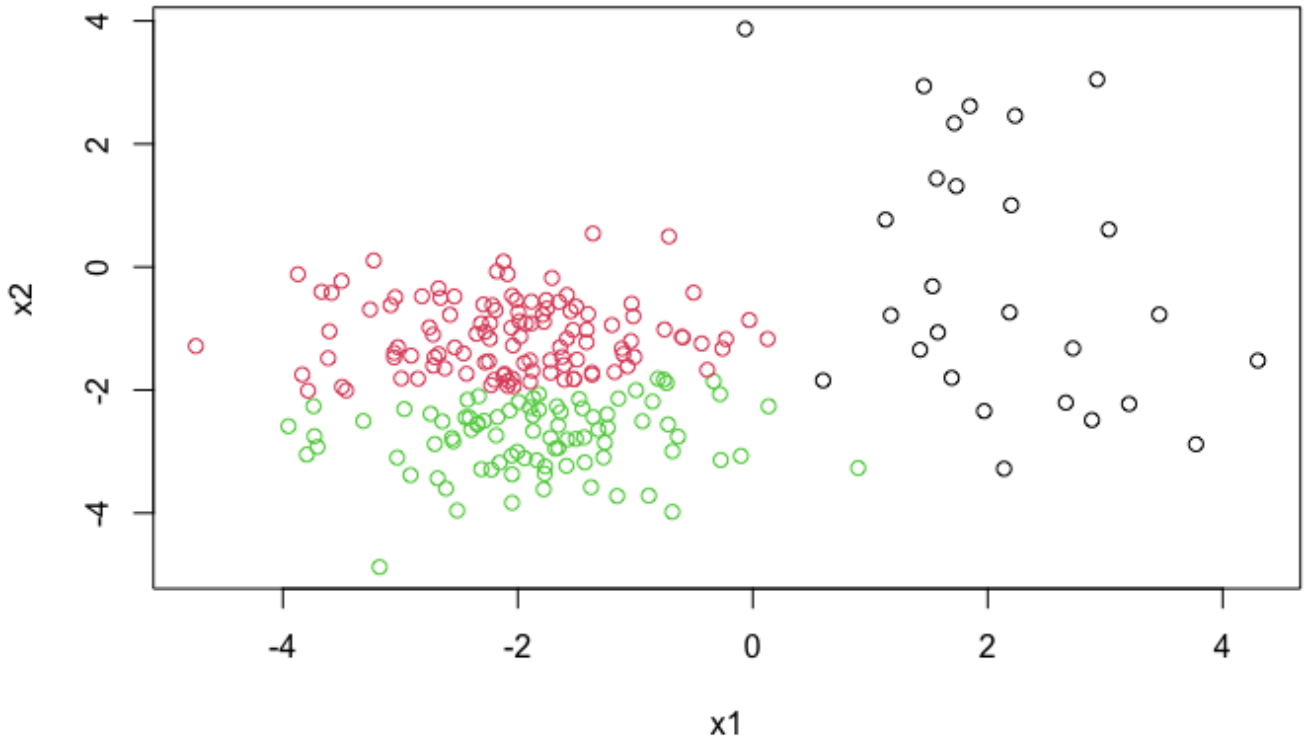
Hide

```
# k-means法によるクラスタリング
result <- kmeans(x = dat, centers = 3, iter.max = 100)
result$centers
```

```
        x1         x2
1  2.114349 -0.1677053
2 -2.042958 -1.1106878
3 -1.819332 -2.7713581
```

Hide

```
# クラスタリングの結果の可視化
plot(dat, col = result$cluster)
```


x2
4
2
0
-2
-4
-4
-2
0
2
4
x1

第11回 階層クラスタリング

Code

1. 概要

階層クラスタリングは、k-means法と同様に類似したデータ点をまとめてあげるクラスタリングとよばれる手法の一つです。

2. 階層クラスタリングのデモ

2.1 必要なライブラリの読み込み

クラスタリングを計算するためのclusterライブラリ、デンドログラムを整形するためのdendextendライブラリが必要になります。

Hide

```
# 必要なライブラリのロード
library(cluster)
library(dendextend)
```

```
Registered S3 methods overwritten by 'tibble':
  method     from
  format.tbl pillar
  print.tbl  pillar

---------------------
Welcome to dendextend version 1.13.4
Type citation('dendextend') for how to cite the package.

Type browseVignettes(package = 'dendextend') for the package vignette.
The github page is: https://github.com/talgalili/dendextend/

Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
Or contact: <tal.galili@gmail.com>

    To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
---------------------


 次のパッケージを付け加えます: ‘dendextend’

 以下のオブジェクトは ‘package:stats’ からマスクされています:

     cutree
```

2.2 データの読み込み

各国の乳製品の消費量のデータを持ちいて、消費量による国のクラスタリングをやってみましょう。

Hide

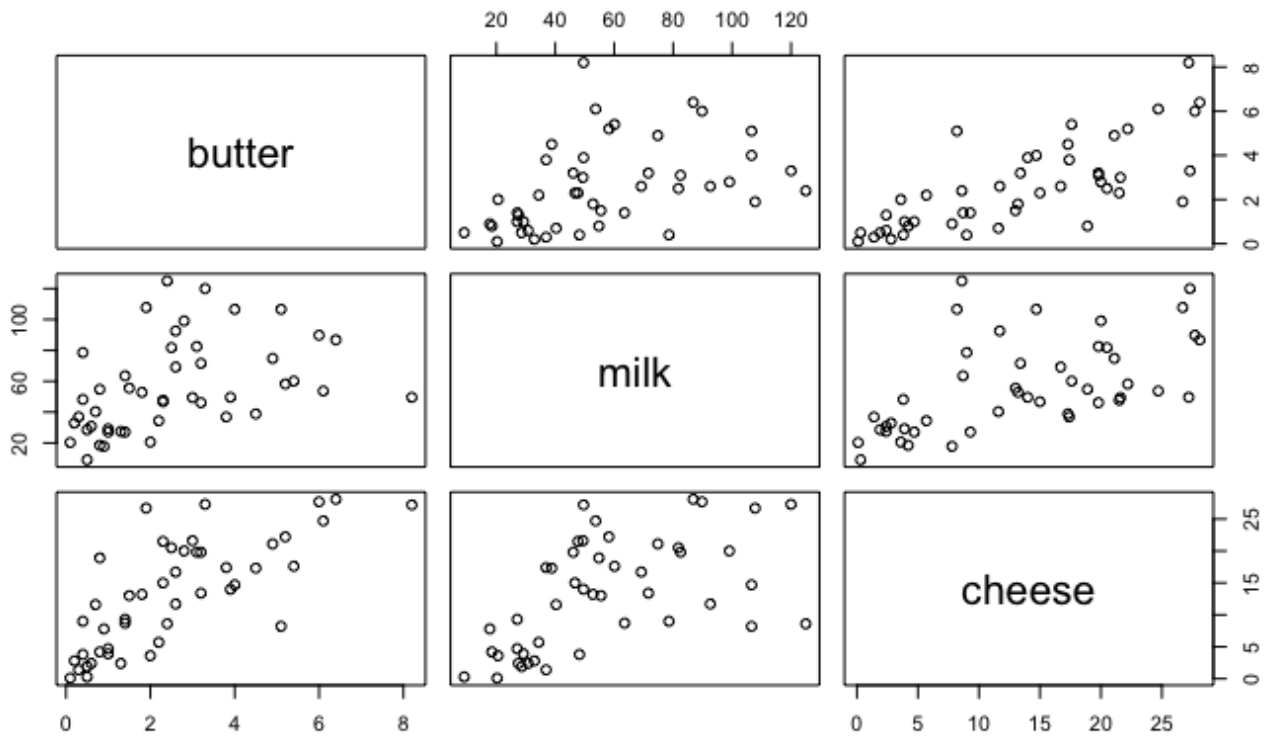
```
dat <- read.csv("./data/dairy_products.csv",
                fileEncoding = "utf-8",     # WindowsではUTF-8-BOM
                row.names = "country")
head(dat, n = 5)
```

	butter <dbl>	milk <dbl>	cheese <dbl>
中国	0.1	20.3	0.1
日本	0.6	30.8	2.4
韓国	0.2	32.9	2.8
イラン	1.0	27.1	4.7
トルコ	0.9	17.8	7.8

5 rows

Hide

```
plot(dat)
```



2.3 データの正規化

階層クラスタリングを行う際には、事前に正規化をおこないます。理由は3.3節で説明します。今回は正規化として標準化を選択しました。

Hide

```
dat_scaled <- scale(dat)
```

2.4 階層クラスタリングの計算

階層クラスタリングは、 cluster パッケージの agnes 関数に以下の値を指定することで計算できます。

- x ：データ
- metric ：距離
- method ：クラスター間距離

今回は、4つのクラスターに分割してみます。

Hide

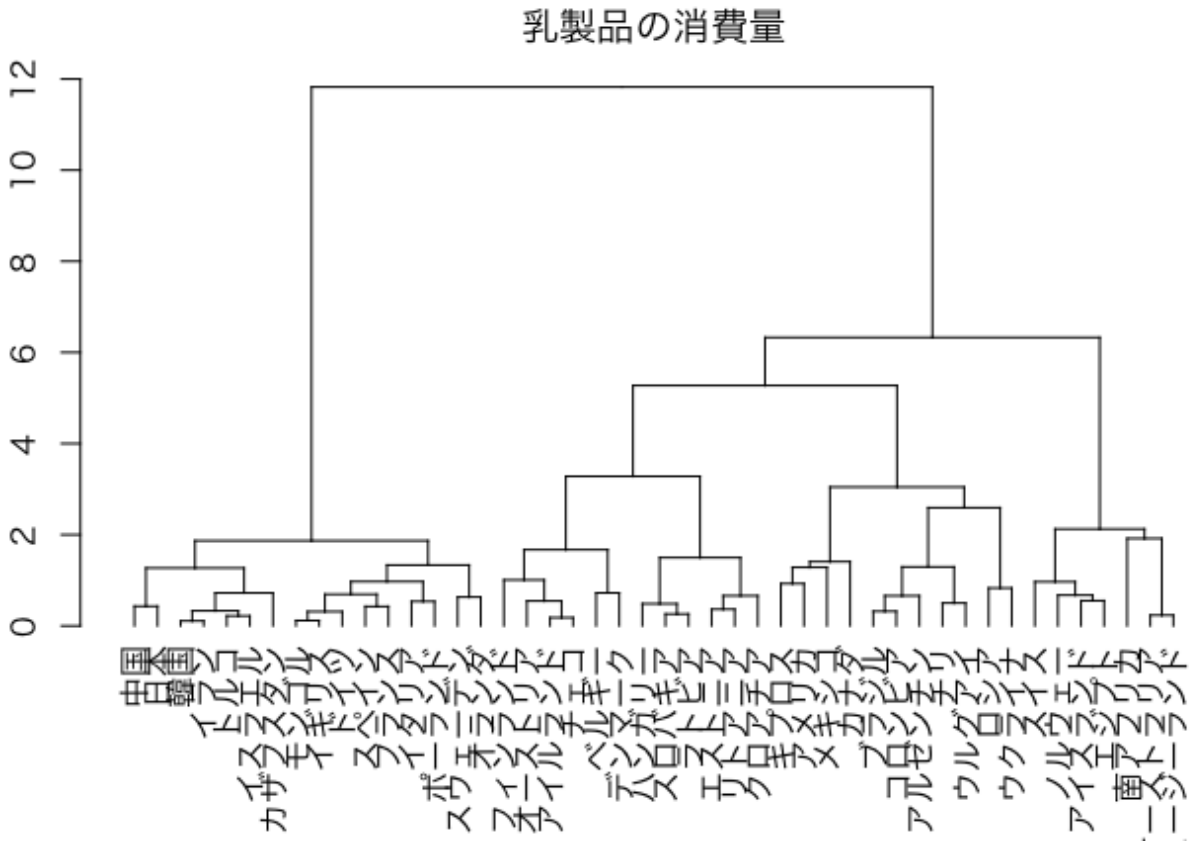
```
# 階層クラスタリングの計算
result <- agnes(x = dat_scaled, metric = "euclidian",method = "ward")
```

2.5 デンドログラムでクラスターを決める

階層クラスタリングを計算するとデンドログラムとよばれるものが出来ます。デンドログラムを参考に、 * クラスター間距離の閾値をどこで切れば適切なクラスターができそうか。 * 各クラスターは他のクラスターと比べてをどういう違いがあるのか。 を検討して、適切なクラスターを決定しましょう。

Hide

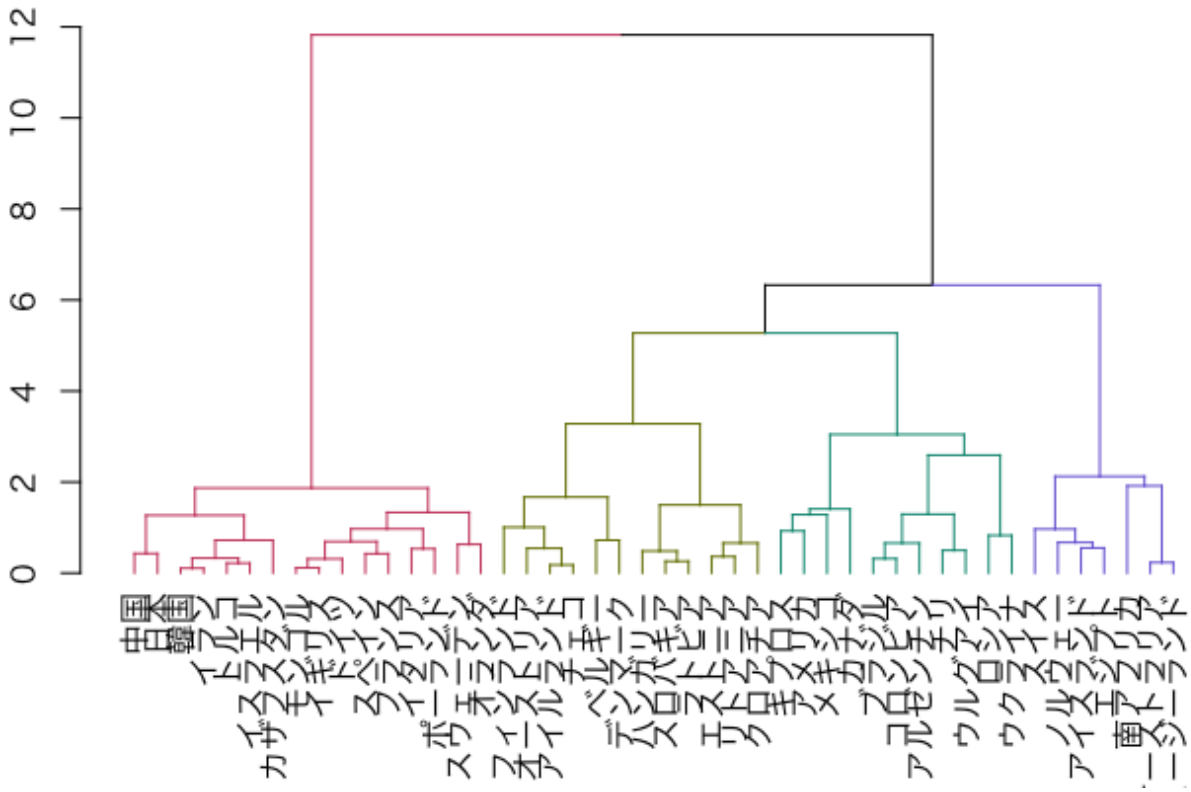
```
par(family = "ヒラギノ角ゴシック W3")    # Macユーザーのみ実行
dend <- as.dendrogram(result)    # クラスタリング結果をデンドログラムに書きなおす。
labels(dend) <- row.names(dat)   # デンドログラムにラベルを付ける。
plot(dend, main = "乳製品の消費量")
```



クラスター数を決定したら、 * デンドログラムを色付けして、各クラスターに所属する国を明確にする。 * 各クラスターの違いを可視化や数値要約で明確にする。 などを行って、クラスターをしっかり理解しましょう。

Hide

```
# 作ったクラスターを可視化する（クラスター数k = 4と決めた場合）
par(family = "ヒラギノ角ゴシック W3")
dend <- color_branches(dend, k = 4)
plot(dend)
```



Hide

```
# 作ったクラスターの様子を確認する。
clusters <- cutree(dend, k = 4)    # クラスターのラベルを計算する。
dat[clusters == 1, ]
```

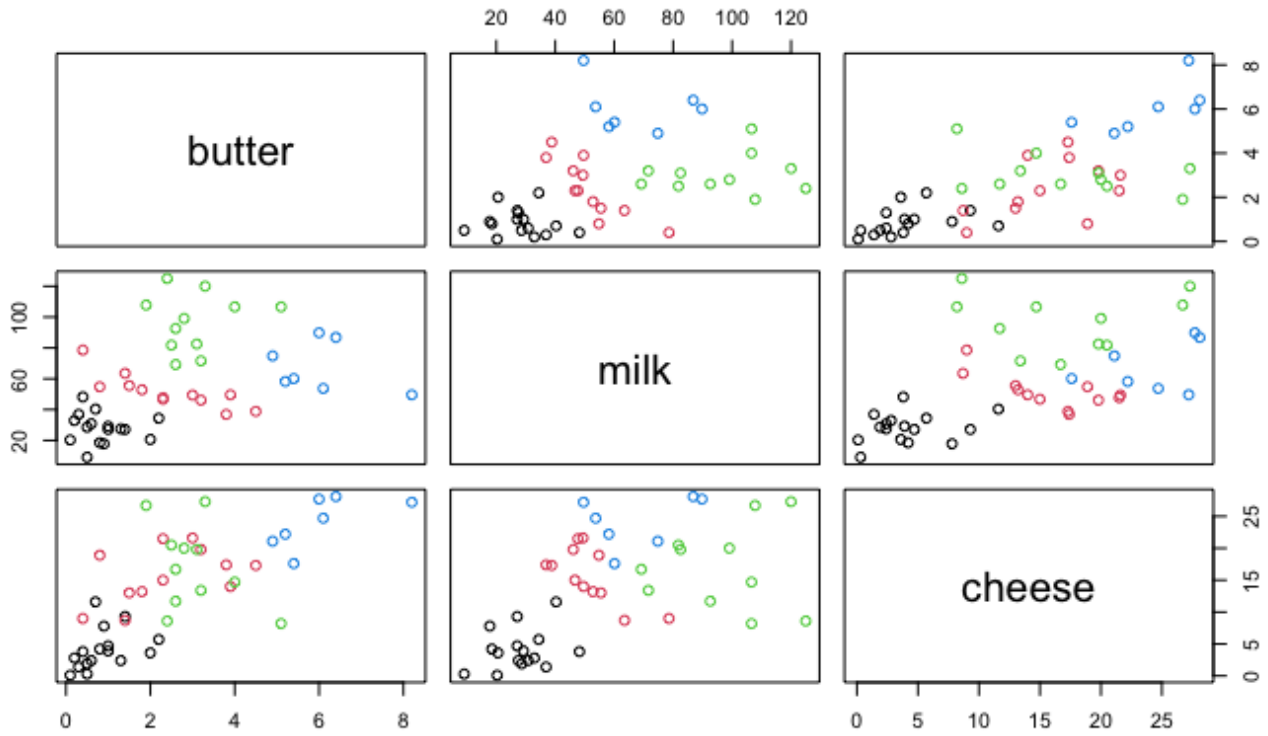
	butter <dbl>	milk <dbl>	cheese <dbl>
中国	0.1	20.3	0.1
日本	0.6	30.8	2.4
韓国	0.2	32.9	2.8
イラン	1.0	27.1	4.7
トルコ	0.9	17.8	7.8
カザフスタン	1.3	27.5	2.4
モンゴル	0.5	9.1	0.3
メキシコ	1.0	29.2	3.9
ブラジル	0.4	48.2	3.8
コロンビア	0.3	36.9	1.4

1-10 of 16 rows

Previous **1** 2 Next

Hide

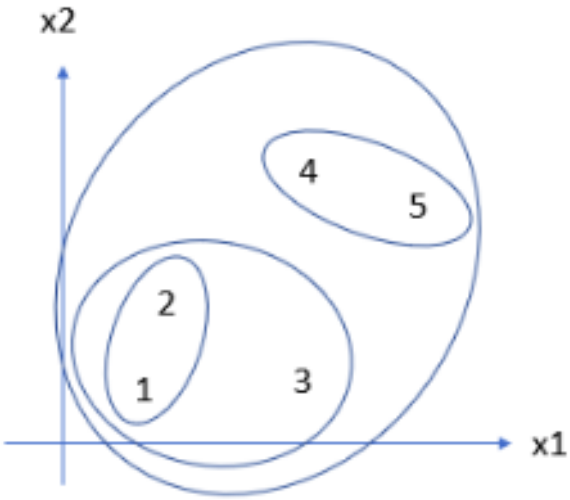
```
plot(dat, col = clusters)
```



3.階層クラスタリングの仕組み

3.1 アルゴリズムの概要

凝集型階層的クラスタリングでは、事前に2つのクラスター C, C′ のクラスター間距離 d(C,C′) を1つ指定しておいて、距離の近いクラスターどうしを併合していきます。



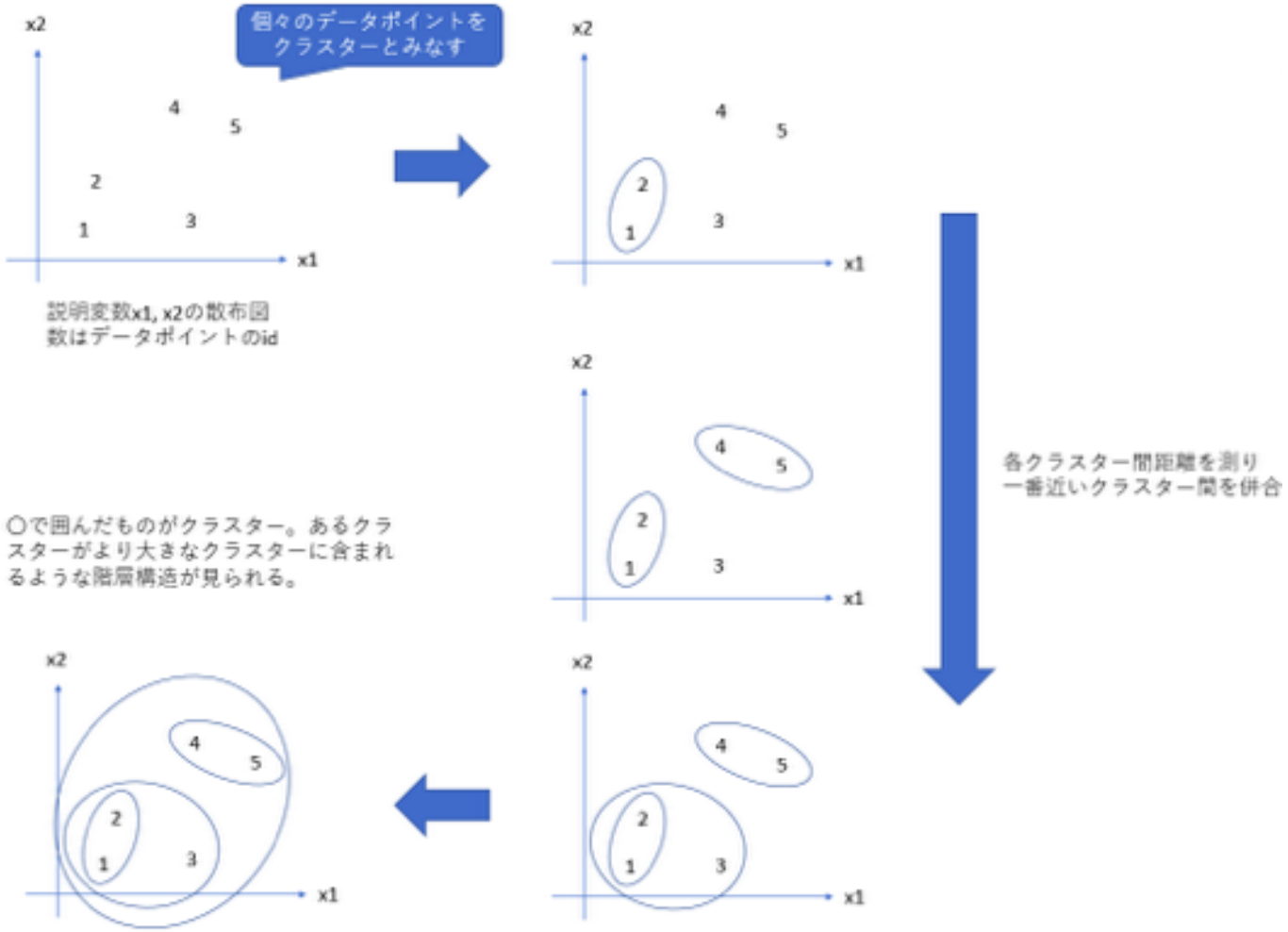
〇で囲んだものがクラスター。あるクラスターがより大きなクラスターに含まれるような階層構造が見られる。

なお、クラスター間距離 d(C, C') には様々な種類がありますが、3.3節で詳しく紹介するWard法を使うのが一般的です。

さて具体的には、以下のステップを繰り返してクラスターを求めます。

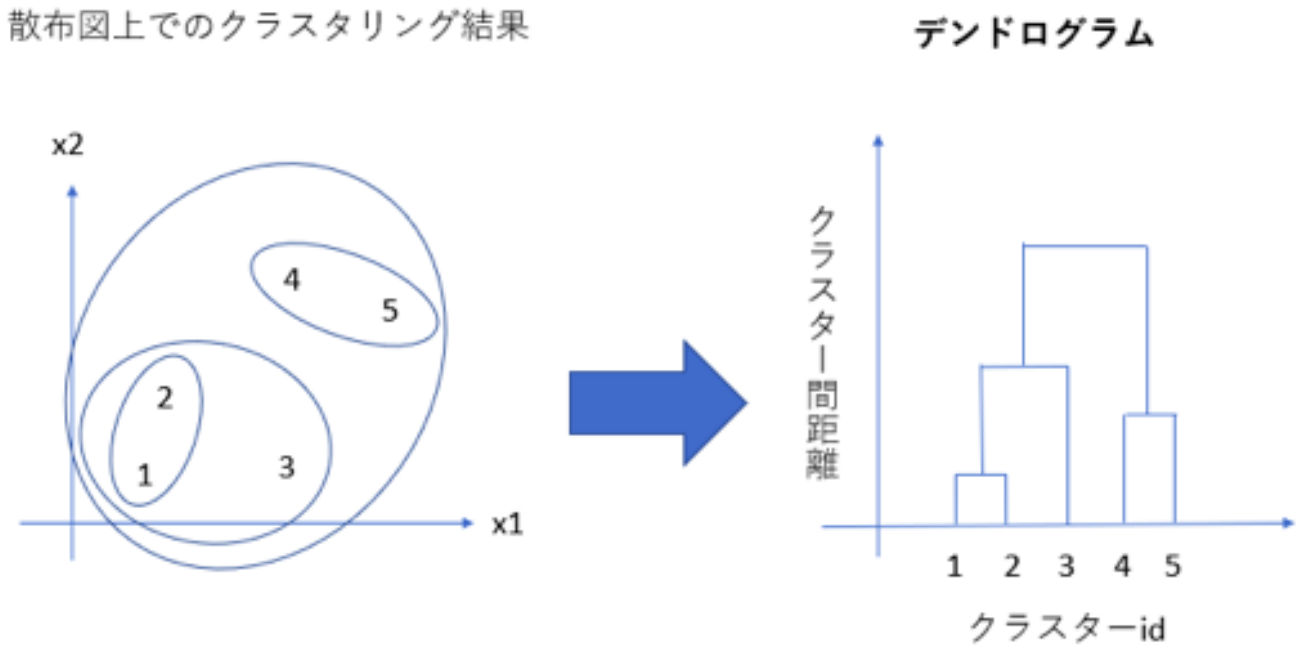
1. 個々のデータポイントを1つのクラスターとみなす。

2. 最初は全てのクラスター間の距離を計算する。
3. 以下の計算を繰り返す。
    a. 最も近接している2つのクラスター C, C′ をみつけて併合する。
    b. データポイントすべてが1つのクラスターとしてまとまったら、この計算を終了する。
    c. 新しく作ったクラスターと既存のクラスターとの距離を計算する。



3.2 結果の可視化

凝集型階層的クラスタリングの結果はデンドログラムという図によって可視化できます。



デンドログラムは、横軸に各データポイントのラベル、縦軸に併合時のクラスター間距離をとり、木グラフによってクラスターの併合の様子を表現したものです。階層的クラスタリングでは、このデンドログラムを頼りに解釈可能なクラスターを見つけ出すことが重要な作業になります。

3.3 クラスター間距離の例：Ward法

クラスター間距離は一般にWard法が用いられます。Ward法の距離は、クラスター C, C' との距離を併合後の偏差平方和と併合前の偏差平方和の合計の差で表現する手法です。

$$d(C, C') = \sum_{i \in C \cup C'} \|x_i - \bar{x}_{C \cup C'}\|^2 - \left(\sum_{i \in C} \|x_i - \bar{x}_C\|^2 + \sum_{i \in C'} \|x_i - \bar{x}_{C'}\|^2 \right)$$

この距離が最も小さいようなクラスターどうしを一つに併合します。Ward法は偏差平方和を用いていることから、説明変数の各列を標準化しておいた方がよいことがわかります。分散の大きい列があると、Ward法の距離がその列に依存して決まってしまい、各列を平等に評価できないためです。

Remark：分散分析をご存知の方は、分散分析で2つの群に有意な差があるとはどういう状況を指すことだったかを思い出してみてください。分散分析はWard法によれば、2群間A,Bのクラスター間距離が離れていると有意に差があると判断する手法だと説明できることが分かります。■

4. 他のクラスター間距離との比較

Ward法にはしばしば、よりシンプルなクラスター間距離で十分なのではないかという話が聞こえます。実際、他のクラスター間距離の例として

- **重心法**：クラスター中心間の距離をクラスター間距離とする。
- **最短距離法**：最も近い点どうしの距離をクラスター間距離とする。

があり、Ward法に比べてシンプルな定義であることは言うまでもないでしょう。しかし、ここでは重心法や最短距離法には欠点が知られています。この欠点を紹介するなかで、Ward法が階層的クラスタリングを行う際の初手として使いやすい距離であることを説明しましょう。

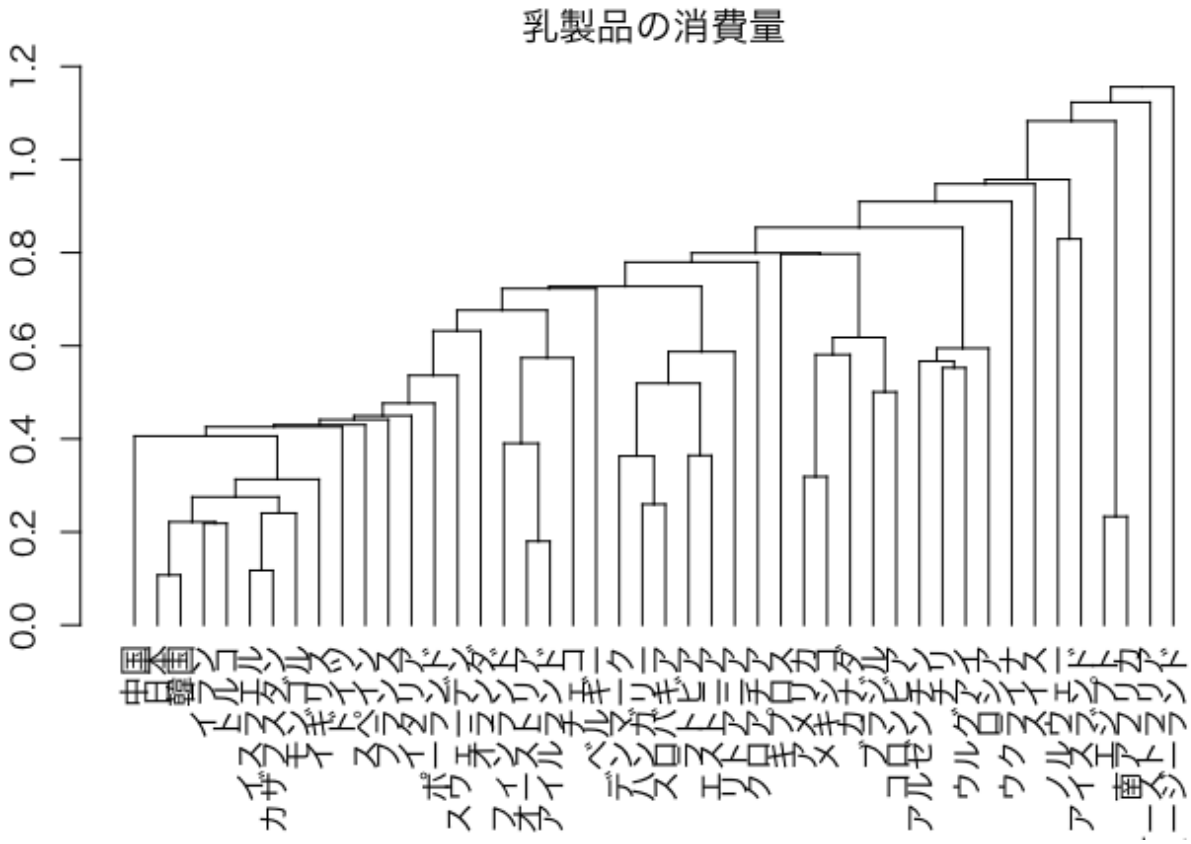
4.1 最短距離法と空間拡散

最短距離法は、最も近い点どうしの距離をクラスター間距離とする凝集型階層的クラスタリングです。この手法には、空間拡散という欠点があります。空間拡散とは、一度併合されて出来たクラスターが他のデータポイントを加速度的に併合していく性質のことです。

空間拡散という性質は、解釈に困るクラスタリングを生成する要因になり、特にデンドログラムでは鎖状構造としてこの現象を確認できます。今回のdairy_products.csvデータセットで最短距離法を試して、デンドログラムを確認してみましょう。

Hide

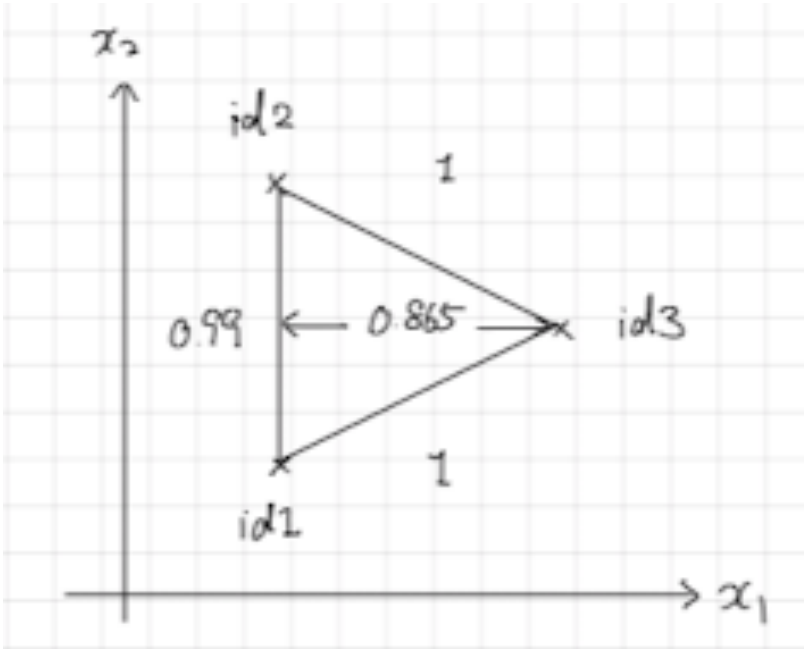
```
# 最短距離法の計算
result <- agnes(x = dat_scaled, metric = "euclidean", method = "single")
# デンドログラム：クラスタリング結果の可視化
dend <- as.dendrogram(result)    # クラスタリング結果をデンドログラムに書きなおす。
labels(dend) <- row.names(dat)    # デンドログラムにラベルを付ける。
par(family = "ヒラギノ角ゴシック W3")
plot(dend, main = "乳製品の消費量")
```



4.2 重心法と単調性

重心法はクラスター中心間の距離をクラスター間距離とする凝集型階層的クラスタリングです。この手法には、デンドログラムの単調性がないという欠点があります。デンドログラムの単調性とは、クラスター併合後によりクラスター間距離の小さいクラスター対ができる現象のことです。

問題：次のデータを重心法で階層クラスタリングしたときのデンドログラムをかいてみましょう。


x_2
id2
1
0.99
0.865
id3
1
id1
x_1

第12回：Fisherの線形判別分析

Code

1. 概要

Fisherの線形判別分析は、 男性/女性 や clickする/clickしない のような2種類のラベルのいずれかがデータ点に与えられるとき、このラベルを説明変数からうまく予測するような式を作る方法の一つです。同様の問題を解く手法には、以前紹介したロジスティック回帰がありますが、考え方が大きく異なってきます。

2. Fisherの線形判別分析のデモ

2.1 データセットの準備

R 言語にもともと備え付けられている iris データセットをデモに用いましょう。話を簡単にするため、事前に以下の処理を行っておいてください。

- 説明変数は Petal.Length と Petal.Width の2変数のみとする。
- 目的変数は Species とし、ラベルは versicolor と virginica の2種類のみを対象とする。

Hide

```
# デモデータセットの作成
data(iris)    # irisデータの読み込み
dat <- iris[iris$Species %in% c("versicolor", "virginica"),
            c("Petal.Length", "Petal.Width", "Species")]
dat$Species <- factor(dat$Species, levels = c("versicolor", "virginica"))
head(dat, n = 5)
```

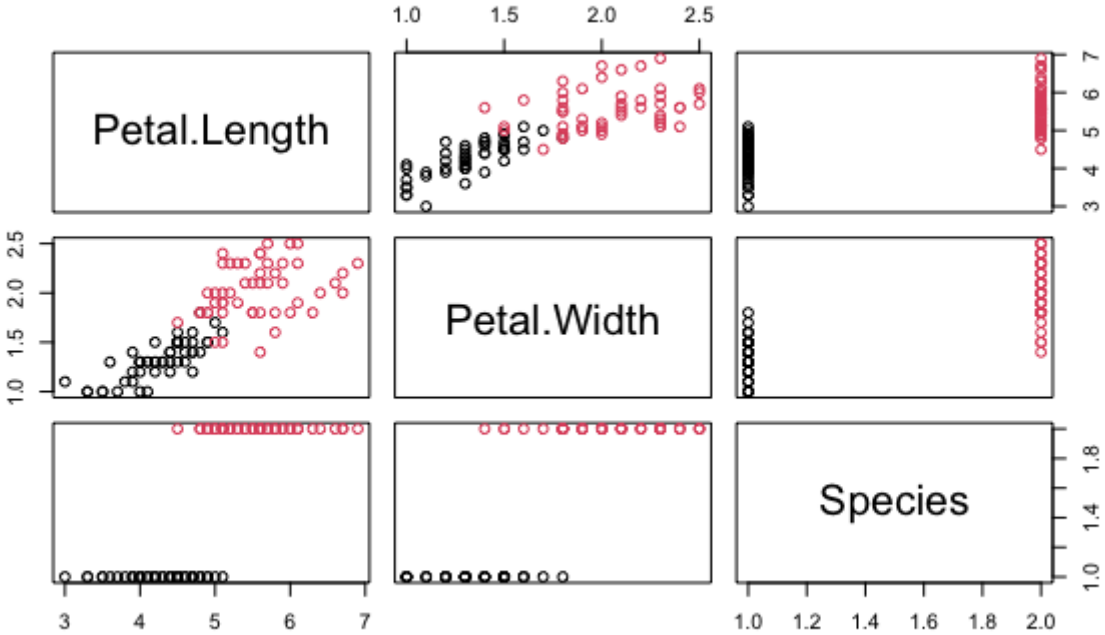
	Petal.Length	Petal.Width	Species
	<dbl>	<dbl>	<fctr>
51	4.7	1.4	versicolor
52	4.5	1.5	versicolor
53	4.9	1.5	versicolor
54	4.0	1.3	versicolor
55	4.6	1.5	versicolor

5 rows

散布図をかいてデータの様子を確認しておきましょう。

Hide

```
plot(dat, col = dat$Species)
```



2.2 課題設定

次の課題を考えます。

課題：Petal.Length と Petal.Width の値から Species が versicolor と virginica のどちらであるかを予測する式を作りましょう。

2.3 Fisherの線形判別分析

Fisherの線形判別分析によって、そのデータが versicolor と virginica のどちらなのかを予測するための判別関数

$$f(Petal.Length, Petal.Width) = w_1 \times Petal.Length + w_2 \times Petal.Width$$

を求めることができます。具体的にどう求めるのか、その考え方は第3章で説明します。ひとまず R 言語を用いて判別関数を求めてみましょう。Fisherの線形判別分析は MASS パッケージの lda 関数によって計算することができます。

Hide

```
library(MASS)
result <- lda(Species ~ Petal.Length + Petal.Width, data = dat)
result$scaling     # w1とw2
```

```
                LD1
Petal.Length 0.8712821
Petal.Width  2.9247035
```

この結果、判別関数が次のように得られました。

$$f(Petal.Length, Petal.Width) = 0.871 \times Petal.Length + 2.925 \times Petal.Width$$

ラベルを予測したいデータ点の説明変数 Petal.Length , Petal.Width の値を判別関数に代入すると、ラベルを予測する上でヒントになる**判別得点**が得られます。例えば、データ dat 一行めに記録されているデータ点の判別得点は

Hide

```
sum(result$scaling * dat[1 , c("Petal.Length", "Petal.Width")])
```

```
[1] 8.189611
```

と計算できます。判別得点を用いて実際にどうラベルの予測を決めるかは次の節で解説しましょう。

2.4 ラベルの予測

ラベルの予測を決める一つのやり方として、判別得点を数直線上に表したとき、 versicolor の判別得点の平均値と virginica の判別得点の平均値を計算しておき、興味のあるデータ点の判別得点がどちらに近いかを参考にする方法があります。

versicolor の判別得点の平均値と virginica の判別得点の平均値を計算すると次のようになります。

Hide

```
# 説明変数の群平均
group_mean <- aggregate(.~ Species, dat, mean)
group_mean
```

Species <fctr>	Petal.Length <dbl>	Petal.Width <dbl>
versicolor	4.260	1.326
virginica	5.552	2.026

2 rows

Hide

```
# 群平均から計算される各群の判別得点平均値
group_center <- as.matrix(group_mean[, c("Petal.Length", "Petal.Width")]) %*% result$scaling
group_center
```

```
          LD1
[1,]  7.589818
[2,] 10.762807
```

今回の場合、 versicolor のほうが判別得点が 8.19 に近いので、先ほどのデータ点を versicolor と予測することになります。なお、正負でラベルを予測できるように、判別関数を判別得点の群平均の中点で引いておくことがあります。今回の場合

Hide

```
mean(group_center)
```

```
[1] 9.176313
```

なので、判別関数を次のように修正するということです。

$$f(Petal.Length, Petal.Width) = 0.871 \times Petal.Length + 2.925 \times Petal.Width - 9.176$$

実際、 lda 関数ではこの判別関数を利用しています。 predict 関数を用いて判別関数を計算できるので、確認してみましょう。

Hide

```
# R言語のpredict関数で計算した判別得点
pred <- predict(result, dat[1, c("Petal.Length", "Petal.Width")])
pred$x
```

```
         LD1
51 -0.9867023
```

Hide

```
# 解説した内容に沿って計算した判別得点
sum(result$scaling * dat[1 , c("Petal.Length", "Petal.Width")]) - mean(group_center)
```

```
[1] -0.9867023
```

結果が一致しましたね。

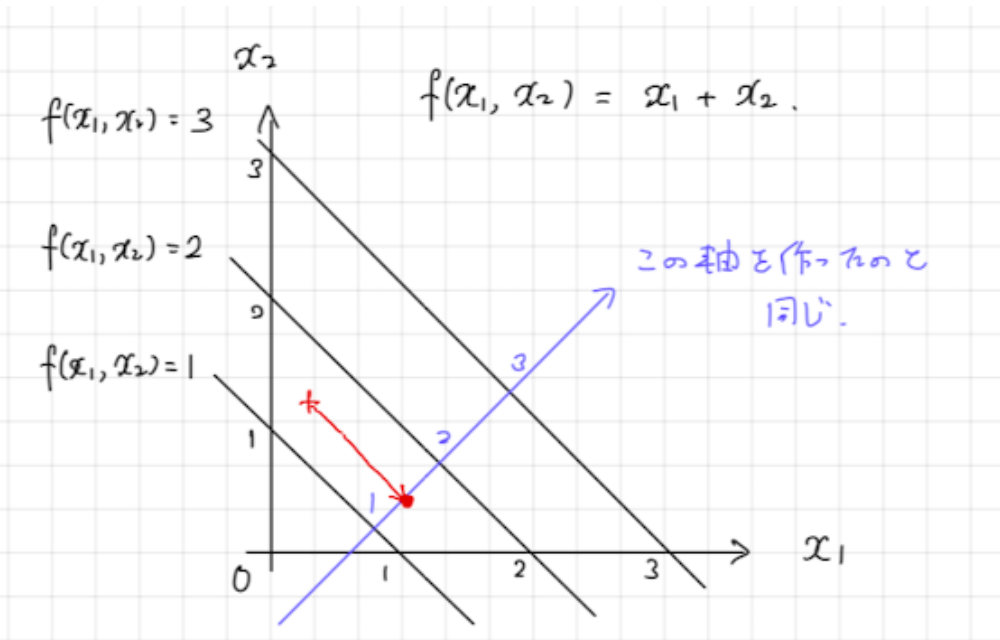
3. Fisherの線形判別分析の仕組み

3.1 基本的な考え方

判別関数は、判別するのに適した軸を新しく作っているようなイメージです。次の問題を解くことで、このイメージを伝えることができるでしょうか。

問題 : 関数 $f(x_1, x_2) = x_1 + x_2$ が (x_1, x_2)-座標上に定める 等高線を求めてください。

解答 :

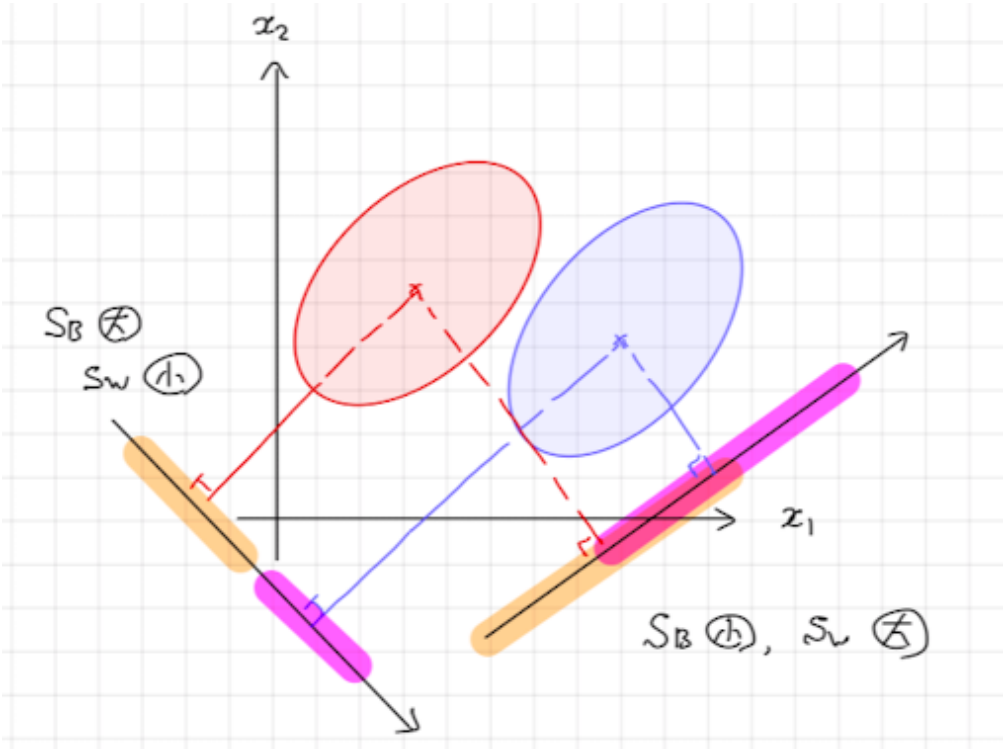


問題では一次式の係数を $(w_1, w_2) = (1, 1)$ と決めていましたが、Fisherの線形判別分析ではこの係数を「判別に適した」値にしたいわけです。では、この「判別に適している」という性質をどのように定式化するのでしょうか。

x_1, x_2 を説明変数とします。このとき、判別関数 $f(x_1, x_2) = w_1 x_1 + w_2 x_2$ が判別に適しているとは、次の二つの性質が成り立っていることと考えることができます。

- 判別得点の群平均がなるべく離れている。（**群間分散**を大きくする。）
- 各群の判別得点の分散をなるべく小さくする。（**群内分散**を小さくする。）

図で表すなら次のようになります。



そこで、Fisherの線形判別分析では

$$\lambda = \frac{\text{群間分散}}{\text{群内分散}}$$

をなるべく大きくするような判別関数（の係数）を求めようと考えます。

3.2 数式による説明

以下では簡単のため次のようなデータを考えます。

id	x_1	x_2	y	f:判別得点
1	$x_{11}^{(-1)}$	$x_{12}^{(-1)}$	-1	$f_1^{(-1)} = w_1 x_{11}^{(-1)} + w_2 x_{12}^{(-1)}$
2	$x_{21}^{(-1)}$	$x_{22}^{(-1)}$	-1	$f_2^{(-1)} = w_1 x_{21}^{(-1)} + w_2 x_{22}^{(-1)}$
3	$x_{11}^{(1)}$	$x_{12}^{(1)}$	1	$f_1^{(1)} = w_1 x_{11}^{(1)} + w_2 x_{12}^{(1)}$
4	$x_{21}^{(1)}$	$x_{22}^{(1)}$	1	$f_2^{(1)} = w_1 x_{21}^{(1)} + w_2 x_{22}^{(1)}$

群間分散

群間分散は、判別得点の群平均の離れ具合のことでした。これを数式で表しておきましょう。まずは群平均 $\bar{f}^{(-1)}, \bar{f}^{(1)}$ を数式で表します。

$$
\begin{aligned}
\bar{f}^{(-1)} &= \frac{1}{2}\left(f_1^{(-1)} + f_2^{(-1)}\right) \\
&= \frac{1}{2}\left\{w_1\left(x_{11}^{(-1)} + x_{21}^{(-1)}\right) + w_2\left(x_{12}^{(-1)} + x_{22}^{(-1)}\right)\right\} \\
&= w_1\bar{x_1}^{(-1)} + w_2\bar{x_2}^{(-1)}
\end{aligned}
$$

ここで $\bar{x_1}^{(-1)}, \bar{x_2}^{(-1)}$ は x_1, x_2 の $y = -1$ における群平均です。同様にして、

$$
\bar{f}^{(1)} = w_1\bar{x_1}^{(1)} + w_2\bar{x_2}^{(1)}
$$

が成り立ちます。これより群間分散は次のように表すことができます。

$$
(\bar{f}^{(-1)} - \bar{f}^{(1)})^2 = \left\{w_1(\bar{x_1}^{(-1)} - \bar{x_1}^{(1)}) + w_2(\bar{x_2}^{(-1)} - \bar{x_2}^{(1)})\right\}^2
$$

なお $w = (w_1, w_2)^T$ とし、また

$$
S_B = \begin{pmatrix} (\bar{x_1}^{(-1)} - \bar{x_1}^{(1)})^2 & (\bar{x_1}^{(-1)} - \bar{x_1}^{(1)})(\bar{x_2}^{(-1)} - \bar{x_2}^{(1)}) \\ (\bar{x_1}^{(-1)} - \bar{x_1}^{(1)})(\bar{x_2}^{(-1)} - \bar{x_2}^{(1)}) & (\bar{x_2}^{(-1)} - \bar{x_2}^{(1)})^2 \end{pmatrix}
$$

とおくと、群間分散は $(\bar{f}^{(-1)} - \bar{f}^{(1)})^2 = w^T S_B w$ と表すことができます。

デモに用いたデータについて、 R 言語で行列 S_B を計算するなら次のようになります。

Hide

```
# 説明変数の群平均
group_mean <- aggregate(.~ Species, dat, mean)
group_mean <- group_mean[ , c("Petal.Length", "Petal.Width")]
group_mean    # 1行目がversicolor, 2行目がvirginica
```

Petal.Length <dbl>	Petal.Width <dbl>
4.260	1.326
5.552	2.026

2 rows

を用いれば、

Hide

```
# S_Bを計算する。
x_diff <- as.numeric(group_mean[1, ] - group_mean[2, ])
SB <- x_diff %*% t(x_diff)
SB
```

```
         [,1]   [,2]
[1,] 1.669264 0.9044
[2,] 0.904400 0.4900
```

と求めることができます。

群内分散

群内分散は、各群の判別得点の分散のことでした。数式で表すと次のようになります。

$$\frac{1}{4-2}\left\{(f_1^{(-1)}-\bar{f}^{(-1)})^2+(f_2^{(-1)}-\bar{f}^{(-1)})^2+(f_1^{(1)}-\bar{f}^{(1)})^2+(f_2^{(1)}-\bar{f}^{(1)})^2\right\}$$

実はこの式が、母平均の差の検定などにあわられる**poolされた分散**であることに気づけると理解が進むと思います。群内分散も行列を使って表すことができます。$y=-1$の群の分散共分散行列を$S^{(-1)}$、$y=1$の群の分散共分散行列を $S^{(1)}$ と表すとき、$S_W=\frac{1}{4-2}\{(2-1)S_1+(2-1)S_2\}$ とおけば群間分散は

$$w^T S_W w$$

と表すことができます。

デモに用いたデータについて、 R 言語で行列 S_W を計算するなら次のようになります。

Hide

```
Sm <- cov(dat[dat$Species == "versicolor", c("Petal.Length", "Petal.Width")])
Sp <- cov(dat[dat$Species == "virginica", c("Petal.Length", "Petal.Width")])
SW <- 1/(100-2) * ((50-1)*Sm + (50-1)*Sp)
SW
```

```
             Petal.Length Petal.Width
Petal.Length   0.26270204  0.06096327
Petal.Width    0.06096327  0.05726939
```

群間分散/群内分散の最大化

$\lambda=$ **群間分散/群内分散** を最大化する w を求めるのがFisherの線形判別分析でした。行列による式の表現を用いると λ は

$$\lambda=\frac{w^T S_B w}{w^T S_W w}$$

です。これに商の微分公式を用いれば、

$$\frac{d\lambda}{dw}=\frac{1}{(w^T S_W w)^2}\left(2(w^T S_W w)S_B w-2(w^T S_B w)S_W w\right)$$

から、(λ, w) の解は $S_B w=\lambda S_W w$ をみたすことがわかります。特に S_W が逆行列を持つ場合は、

$$S_W^{-1}S_B w=\lambda w$$

が得られますから、(λ, w) の解は行列 $S_W^{-1}S_B$ の固有値問題を解いて、最大の固有値の固有ベクトルが求めたかった w だとわかるわけです。

R 言語で今回のデモデータに対してこの計算をするなら、次のようになります。

Hide

```
w <- eigen(solve(SW)%*%SB)
w
```

```
eigen() decomposition
$values
[1] 10.06786  0.00000

$vectors
           [,1]       [,2]
[1,] -0.2855048 -0.4763708
[2,] -0.9583773  0.8792445
```

この結果を lda 関数の結果と比較してみましょう。

Hide

```
result$scaling / w$vectors[ , 1]
```

```
                  LD1
Petal.Length -3.051725
Petal.Width  -3.051725
```

このことから lda 関数は eigen 関数の固有ベクトルを -3.052 倍した結果を出力していることがわかります。判別関数は係数が定数倍されても同じ意味なので、この結果はFisherの線形判別分析（特に lda 関数）の動作原理を数式を用いて確認できたことを意味しています。

第13回：非線形判別分析

Code

1. 概要

Fisherの線形判別分析は、一次式で表される判別関数でラベルを予測する手法でした。そこで今回は、Fisherの線形判別分析の拡張にあたる非線形判別分析を紹介します。この判別関数は二次式になっていて、Fisherの線形判別分析より複雑な決定境界を作ることができます。

Remark : 決定境界とは、ラベルの予測結果が変わる条件のことです。つまり、判別関数=0をみたす説明変数の値全体のことです。判別関数が一次式のとき、決定境界は散布図上で超平面（説明変数が2つなら直線・3つなら平面...）になります。

2. 非線形判別分析

2.1 データセットの準備

Ｒ言語にもともと備え付けられている iris データセットをデモに用いましょう。話を簡単にするため、事前に以下の処理を行っておいてください。

- 説明変数は Petal.Length と Petal.Width の2変数のみとする。
- 目的変数は Species とし、ラベルは versicolor と virginica の2種類のみを対象とする。

Hide

```
# デモデータセットの作成
data(iris)    # irisデータの読み込み
dat <- iris[iris$Species %in% c("versicolor", "virginica"),
            c("Petal.Length", "Petal.Width", "Species")]
dat$Species <- factor(dat$Species, levels = c("versicolor", "virginica"))
head(dat, n = 5)
```

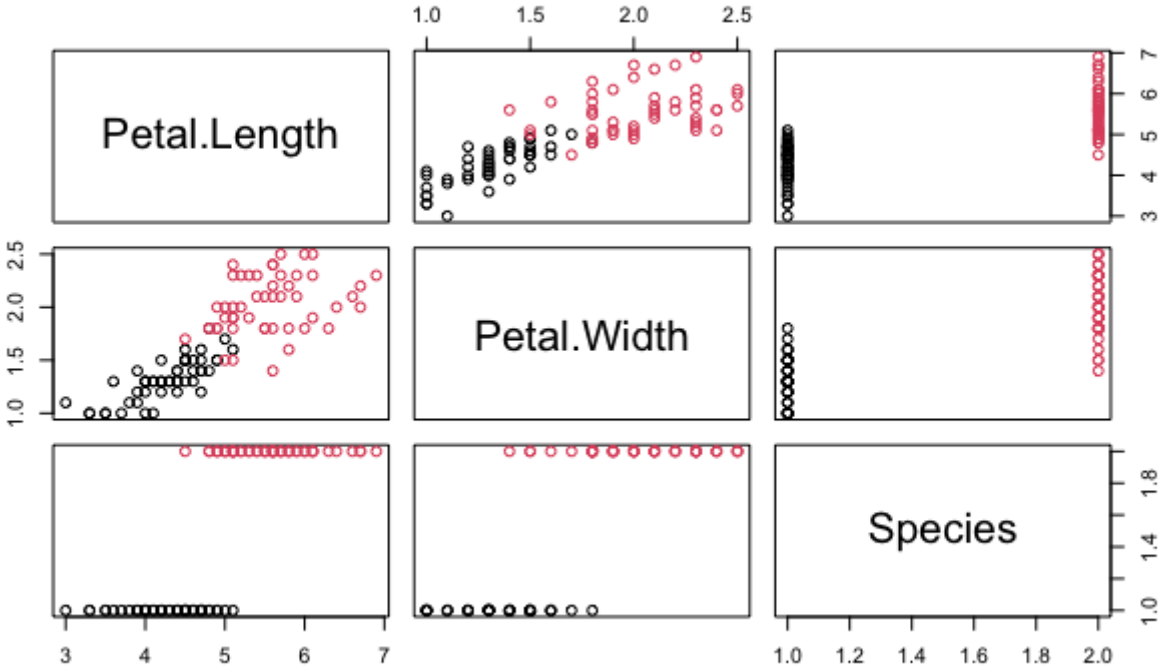
	Petal.Length <dbl>	Petal.Width <dbl>	Species <fctr>
51	4.7	1.4	versicolor
52	4.5	1.5	versicolor
53	4.9	1.5	versicolor
54	4.0	1.3	versicolor
55	4.6	1.5	versicolor

5 rows

散布図をかいてデータの様子を確認しておきましょう。

Hide

```
plot(dat, col = dat$Species)
```



2.2 課題設定

次の課題を考えます。

課題：Petal.Length と Petal.Width の値から Species が versicolor と virginica のどちらであるかを予測する式を作りましょう。

2.3 非線形判別分析

非線形判別分析によって、そのデータが versicolor と virginica のどちらなのかを予測するための判別関数を求めることができます。具体的にどう求めるのか、その考え方は第3章で説明します。ひとまず R 言語を用いて判別関数を求めてみましょう。非線形判別分析は MASS パッケージの qda 関数によって計算することができます。

Hide

```
library(MASS)
result <- qda(Species ~ Petal.Length + Petal.Width, data = dat)
result
```

```
Call:
qda(Species ~ Petal.Length + Petal.Width, data = dat)

Prior probabilities of groups:
versicolor  virginica
       0.5        0.5

Group means:
           Petal.Length Petal.Width
versicolor        4.260       1.326
virginica         5.552       2.026
```

qda 関数からは残念ながら判別関数の情報を取得することができませんが、次のように予測を行うことはできます。 predict 関数を用いて判別関数を計算できるので、確認してみましょう。

Hide

```
# R言語のpredict関数で計算した判別得点
pred <- predict(result, dat[1, c("Petal.Length", "Petal.Width")])
pred$class
```

```
[1] versicolor
Levels: versicolor virginica
```

3. 非線形判別分析の仕組み

3.1 基本的な考え方

非線形判別分析のアイディアは次の通りです。

アイディア : ラベルを予測したいデータ点が、ラベル $y = 0$ の群に近いか？ラベル $y = 1$ の群に近いか？

もし、ラベルを予測したいデータ点がラベル $y = 0$ より $y = 1$ の群に近いなら、ラベルを $y = 1$ と予測しようというわけです。これは、もしデータ点 x とラベル y が与えられているデータ群 G_y との間の距離 $d(x, G)$ が定義できるなら、判別関数を

$$f(x) = -d(x, G_{y=1}) + d(x, G_{y=0})$$

と定義することと同じです。そして判別関数 $f(x)$ の値が正ならラベルを $y = 1$、負ならラベルを $y = 0$ と予測しようというわけです。

では、データ点 x とラベル y が与えられているデータ群 G_y との間の距離 $d(x, G_y)$ をどのように定義すればよいでしょうか。非線形判別分析では**Mahalanobis距離**とよばれる距離が用いられています。

3.2 Mahalanobis距離

Mahalanobis距離は、1936年にインドの数理統計学者であるプラサンタ・チャンドラ・マハラノビスによって提案された距離です。次のような定義になっています。

定義 : x を興味のあるデータ点、G をデータ群、$\bar{x}_G$ をデータ群 G の標本平均、Σ_G をデータ群 G の分散共分散行列とします。このとき、

$$d(x, G) = (x - \bar{x}_G)^T \Sigma_G^{-1} (x - \bar{x}_G)$$

をMahalanobis距離といいます。

Mahalanobis距離のイメージを掴むために次の問題を解いてみましょう。

問題 : 2変数のデータ(x_1, x_2)を考えます。群 G の標本平均を $\bar{x} = (\bar{x_1}, \bar{x_2}) = (0, 0)$、分散共分散行列を

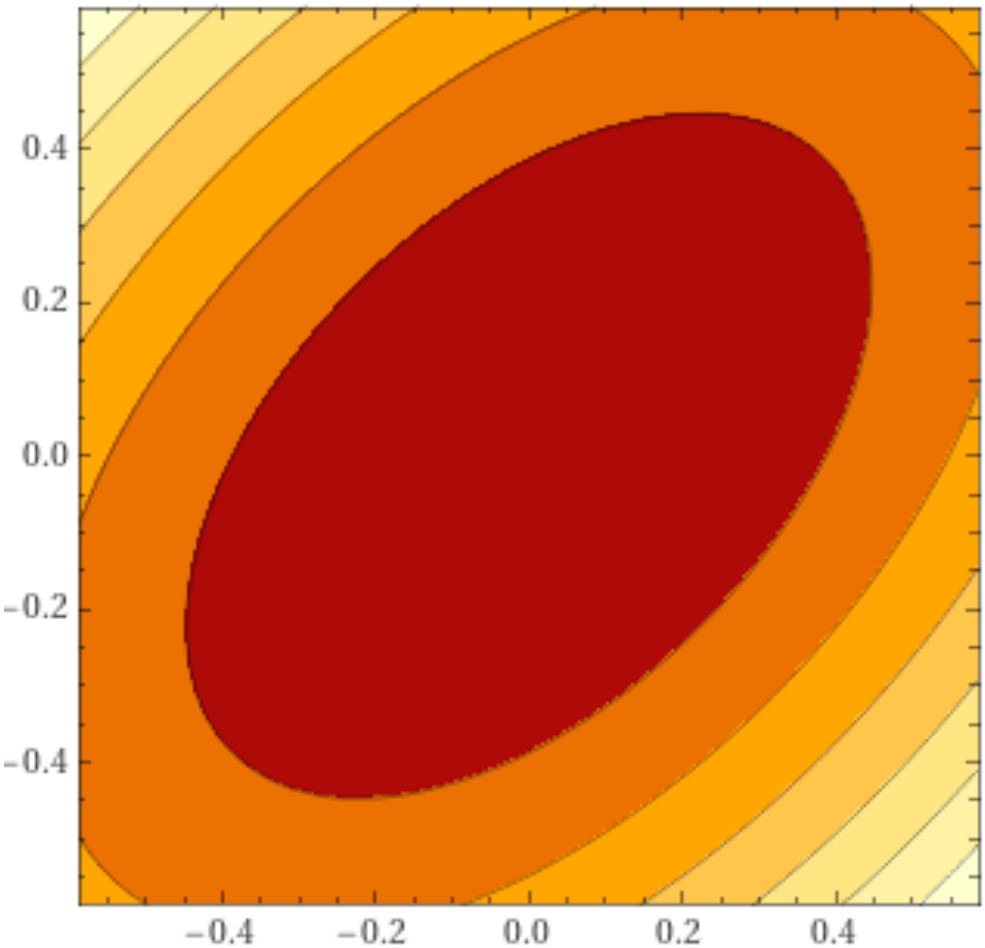
$$\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

とします。このとき、(x_1, x_2) の散布図上にMahalanobis距離の等高線を描いでください。

解答 : Mahalanobis距離を式で表すと、次のようになります。

$$
\begin{aligned}
d(x, G) &= \begin{pmatrix} x_1 - 0 & x_2 - 0 \end{pmatrix} \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x_1 - 0 \\ x_2 - 0 \end{pmatrix} \\
&= \frac{1}{0.75} \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
&= \frac{1}{0.75} (x_1^2 - x_1 x_2 + x_2^2)
\end{aligned}
$$

これより等高線は次のようになります。（以下の図はwolfram alphaでかいたものです。）



以上が問題の解答になります。■

つまりMahalanobis距離は、データ点 x が群 G の母集団からどれだけ発生しづらいデータかを見積もっているものだと解釈することができるでしょう。また、Mahalanobis距離は説明変数の2次式なので、その差をとる非線形判別分析の判別関数もまた一般には2次式になります。このため、非線形判別分析を二次判別分析ということもあります。

3.3 qda 関数の結果の再現

qda 関数による次の計算結果を再現してみましょう。

Hide

```
# R言語のpredict関数で計算した判別得点
pred <- predict(result, dat[1, c("Petal.Length", "Petal.Width")])
pred$class
```

```
[1] versicolor
Levels: versicolor virginica
```

versicolor 群と viginica 群の標本平均、分散共分散行列を求めておきます。

Hide

```
# versicolor群の標本平均と分散共分散行列
dat_versi <- dat[dat$Species == "versicolor",
                 c("Petal.Length", "Petal.Width")]
xbar_versi <- apply(dat_versi, MARGIN = 2, FUN = mean)
cov_versi <- cov(dat_versi)
```

Hide

```
# virginica群の標本平均と分散共分散行列
dat_virgi <- dat[dat$Species == "virginica",
                 c("Petal.Length", "Petal.Width")]
xbar_virgi <- apply(dat_virgi, MARGIN = 2, FUN = mean)
cov_virgi <- cov(dat_virgi)
```

あとは、興味のあるデータ点について、 versicolor 群および virginica 群とのMahalanobis距離を計算し、判別関数の値を計算します。

Hide

```
x <- dat[1, c("Petal.Length", "Petal.Width")]
d_versi <- mahalanobis(x = x, center = xbar_versi, cov = cov_versi)
d_virgi <- mahalanobis(x = x, center = xbar_virgi, cov = cov_virgi)
- d_versi + d_virgi
```

```
     51
4.705084
```

今回、判別関数は正なので、ラベルを versicolor だと予測できます。このようにして qda 関数の予測を再現することができました。

3.4 Fisherの線形判別分析との関連

やや難しい計算になるので詳細は省きますが、Fisherの線形判別分析もこのMahalanobis距離を用いた手法だと解釈できることが知られています。Fisherの線形判別分析では、群 $G_{y=1}, G_{y=0}$ の分散共分散行列が母集団では等しいという仮定を入れます。つまり、分散共分散行列の推定量にはpoolされた分散 S_W を用いるわけです。これは、Fisherの線形判別分析でいうところの群内分散そのものです。

さてこの場合、Mahalanobis距離を用いた判別関数は

$$
\begin{aligned}
f(x) &= -d(x, G_{y=1}) + d(x, G_{y=0}) \\
&= -(x - \bar{x}_{y=1})^T S_W^{-1} (x - \bar{x}_{y=1}) + (x - \bar{x}_{y=0})^T S_W^{-1} (x - \bar{x}_{y=0}) \\
&= 2(\bar{x}_{y=1} - \bar{x}_{y=0})^T S_W^{-1} x + (-\bar{x}_{y=1}^T S_W^{-1} \bar{x}_{y=1} + \bar{x}_{y=0}^T S_W^{-1} \bar{x}_{y=0})
\end{aligned}
$$

です。実は、この判別関数はFisherの線形判別分析から得られる判別関数に等しくなることが証明できます。実際、デモデータで一次式の係数 $(\bar{x}_{y=1} - \bar{x}_{y=0})^T S_W^{-1}$ を計算すると、次のようになります。

Hide

```
Sm <- cov(dat[dat$Species == "versicolor", c("Petal.Length", "Petal.Width")])
Sp <- cov(dat[dat$Species == "virginica", c("Petal.Length", "Petal.Width")])
SW <- 1/(100-2) * ((50-1)*Sm + (50-1)*Sp)
coef <- t(xbar_versi - xbar_virgi) %*% solve(SW)
coef
```

```
     Petal.Length Petal.Width
[1,]    -2.764568   -9.280052
```

これは、 lda 関数によって計算される判別関数の一次式の係数に比例（-3.17倍）していることが次のようにして確認できます。

Hide

```
result <- lda(Species ~ Petal.Length + Petal.Width, data = dat)
coef / as.vector(result$scaling)
```

```
     Petal.Length Petal.Width
[1,]    -3.172989   -3.172989
```

第14回 決定木分析

Code

1. 決定木の概要

決定木分析（decision tree analysis）は、ある変数を他の変数から予測するためのif-thenルールを求める方法です。予測対象の変数を目的変数、予測に用いる変数を説明変数、得られるif-thenルールを**決定木**といいます。また目的変数には量的変数・質的変数のどちらもとることができ、

- 目的変数が量的変数の場合 : 決定木回帰
- 目的変数が質的変数の場合 : 決定木分類

と異なる名前でよばれます。今回は、決定木分類のデモを行い、決定木を求める仕組みを解説します。

2. 決定木分類のデモ

2.1 デモデータ

`data` ディレクトリの `split_demo.csv` をデモデータに用います。このデータは、標本サイズが100で3変数のデータです。今回は、 `x` , `y` を説明変数、 `label` を目的変数とした決定木分類を行ってみます。

Hide

```
dat <- read.csv("./data/split_demo.csv", fileEncoding = "utf-8")
dat$label <- as.factor(dat$label)
head(x = dat, n = 5)
```

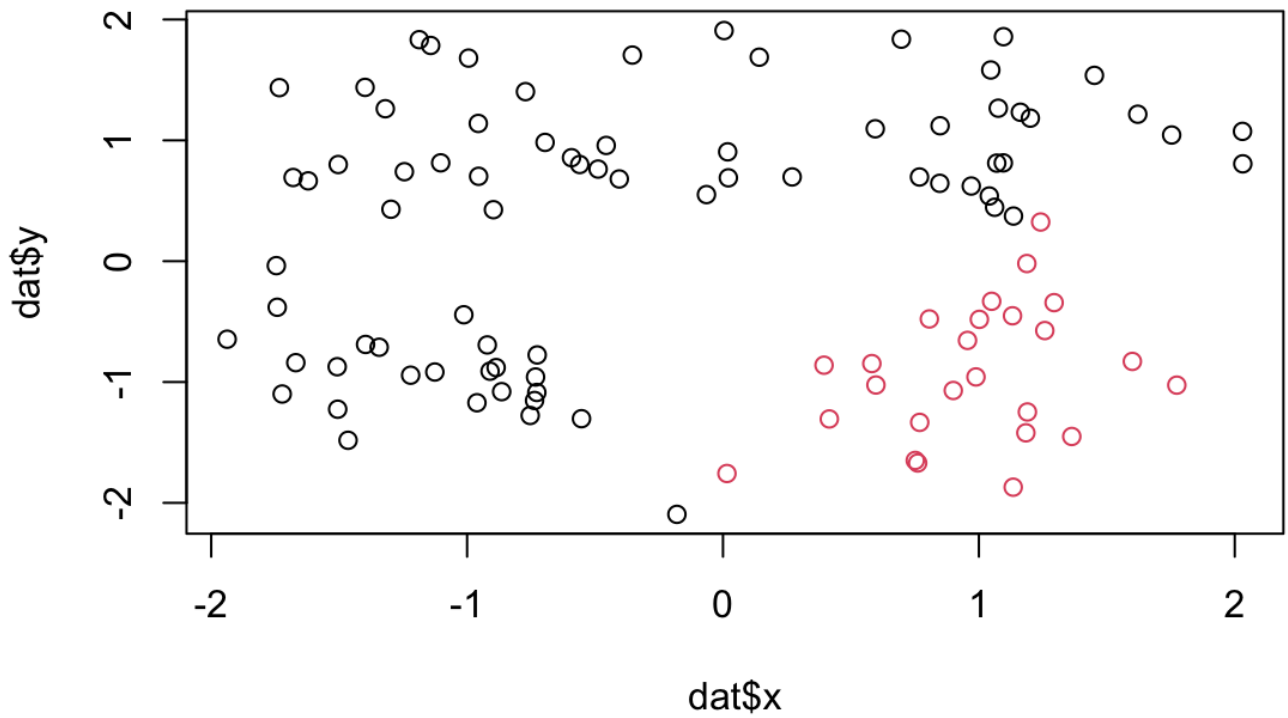
	x	y	label
	<dbl>	<dbl>	<fctr>
1	1.6208878	1.2160687	1
2	0.1418463	1.6881527	1
3	1.1352569	0.3735209	1
4	1.4515459	1.5378424	1
5	1.0407771	0.5394458	1

5 rows

`label` 別に色分けした散布図をかいて、標本点の分布を確認しておきましょう。

Hide

```
plot(dat$x, dat$y, col = dat$label)
```



問題：この散布図から、決定木分類によって得られそうなif-thenルールを予め想定してみましょう。

2.2 決定木分類の計算

決定木分類は `rpart` パッケージの `rpart` 関数で計算することができます。指定する引数は以下のとおりです。

- `formula` ：目的変数と説明変数を指定する。
- `data` ：決定木を作るために用いるデータ。
- `method` ：回帰か分類か。回帰の場合は `anova` 、分類の場合は `class` を指定する。
- `control` ：得られる決定木の複雑さをコントロールするための引数（**ハイパーパラメータ**）
  - `maxdepth` ：ここに渡した数より深い決定木は作らない。
  - `minsplit` ：ここに渡した数より所属する標本点の個数が少ないノードでは、これ以上分割を試みない。
  - `minbucket` ：終端ノードに所属する標本点の個数は、すくなくともここに渡した数より大きくなるようにする。

`control` の説明にはわからない単語もあると思いますが、3.2で詳しく説明します。

Hide

```
library(rpart)
result <- rpart(formula = label ~ x + y,
                data = dat,
                method = "class",
                control = rpart.control(maxdepth = 2,
                                        minsplit = 20,
                                        minbucket = 5))
result
```

```
n= 100

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 100 25 1 (0.75000000 0.25000000)
  2) x< 0.3325615 55  1 1 (0.98181818 0.01818182) *
  3) x>=0.3325615 45 21 2 (0.46666667 0.53333333)
    6) y>=0.3488362 21  0 1 (1.00000000 0.00000000) *
    7) y< 0.3488362 24  0 2 (0.00000000 1.00000000) *
```

`result` には得られた決定木が出力されていますが、分かりづらいと思います。 `partykit` パッケージの `as.party` 関数でわかりやすい形に書き換えておきましょう。

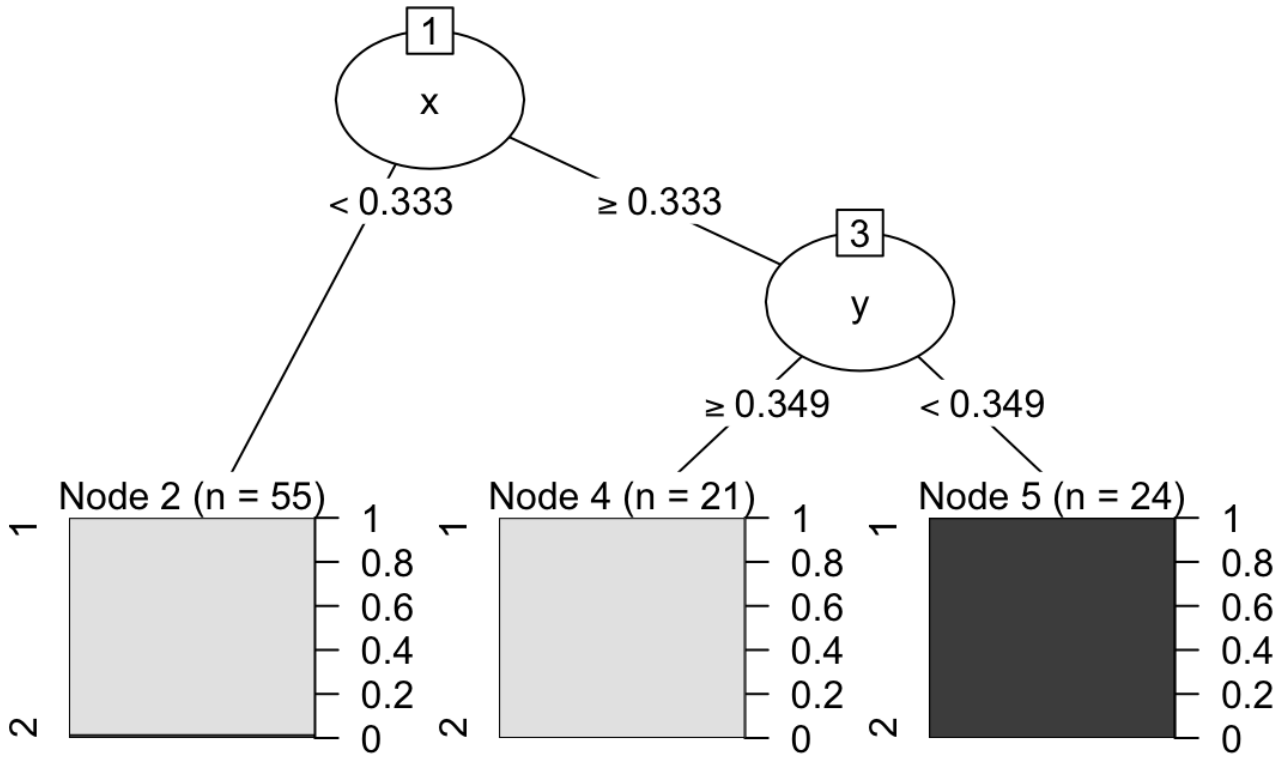
Hide

```
library(partykit)
```

```
要求されたパッケージ grid をロード中です
要求されたパッケージ libcoin をロード中です
要求されたパッケージ mvtnorm をロード中です
```

Hide

```
plot(as.party(result))
```



問題：得られた決定境界を散布図上に図示してください。

2.3 変数重要度

各説明変数が目的変数の予測にどれだけ寄与したのかを数値を用いて表現する方法に**変数重要度**（variable importance）があります。詳しくは3.4節で説明しますが、ひとまず値を参照する方法を紹介します。

Hide

```
result$variable.importance
```

```
       y        x
23.56768 16.33636
```

3. 決定木分類の仕組み

3.1 Giniエントロピー

決定木分類の仕組みを理解するうえで大切になる**Giniエントロピー**を解説します。Giniエントロピーは、データに含まれるラベルの個数の均等さをあらわす数値のことで、すべてのラベルが同じ数含まれているとき最も大きな値をとるように定義されています。

Giniエントロピーの定義を述べます。データに L 種類のラベルがあり、それぞれ $p_1, \cdots, p_K$ の割合で含まれているとします。このとき、Giniエントロピーは

$$I = 1 - \sum_{l=1}^{K} p_l^2$$

と定義されます。

問題 : 2種類のラベル $l = 1, 2$ を考えます。データに各ラベルが含まれる割合 p_1, p_2 が以下のようになっているとき、Giniエントロピーがいくらになっているかを計算してください。

1. $p_1 = 0.5, p_2 = 0.5$
2. $p_1 = 0.8, p_2 = 0.2$
3. $p_1 = 1.0, p_2 = 0.0$

解答 : 1. $I = 1 - (0.5^2 + 0.5^2) = 0.5$ 2. $I = 1 - (0.8^2 + 0.2^2) = 0.32$ 3. $I = 1 - (1^2 + 0^2) = 0$

3.2 決定木を求める仕組み

決定木分類で決定木を求める仕組みを説明します。決定木を求めるときに大切なのは、Giniエントロピーが最も変化するようなルールを見つけるというアイディアです。

データ D を説明変数 x が t 以上の値をもつデータ D_1 と t 未満の値をもつデータ D_2 に分割したとします。このとき、Giniエントロピーの変化量を次のように計算することができます。

$$\Delta I(x, t) = I(D) - \left(\frac{n(D_1)}{n(D)} I(D_1) + \frac{n(D_2)}{n(D)} I(D_2) \right)$$

ここで、$n(D)$ はデータ D に含まれる標本サイズです。決定木分類では、この変化量が最も大きくなるような説明変数 x と閾値 t のペアを逐次的に探していきます。

なお、得られる決定木が複雑になりすぎないように、以下のようなハイパーパラメータを予め設定します。

- `maxdepth` ：ここに渡した数より深い決定木は作らない。
- `minsplit` ：ここに渡した数より所属する標本点の個数が少ないノードでは、これ以上分割を試みない。
- `minbucket` ：終端ノードに所属する標本点の個数は、すくなくともここに渡した数より大きくなるようにする。

ここでノードとは、決定木のうち説明変数と閾値のペア (x, t) が書かれている部分のことです。また、終端ノードとはすべての条件をみたしたデータのうちの何割が各ラベルに所属しているかを記載する部分のことです。

例えば `maxdepth=2` , `minsplit=20` , `minbucket=5` のとき、以下のようなことに注意して説明変数と閾値のペアを探します。

- 深さが2以上になるような決定木を作らない。
- 標本サイズが20を下回るようなデータでは説明変数と閾値のペアを探さない。
- 分割するときは、分割後のデータの標本サイズがいずれも 5 以上になるようにする。

3.3 決定木についての重要な注意点

決定木分析によって得られる決定木が、必ずしも人間の直感と一致するとは限らない点には注意を払ってください。決定木分析では、**得られた決定木を信頼しすぎず、どれだけ得られた決定木に納得感があるかを分析者が反省する**ことが大切になってきます。

単純かつ有名な例としてXOR問題とよばれている分類問題があります。 `XOR_demo.csv` に対して、事前にif-thenルールを想定し、決定木分類によって得られる決定木と比較をおこなってみてください。

Hide

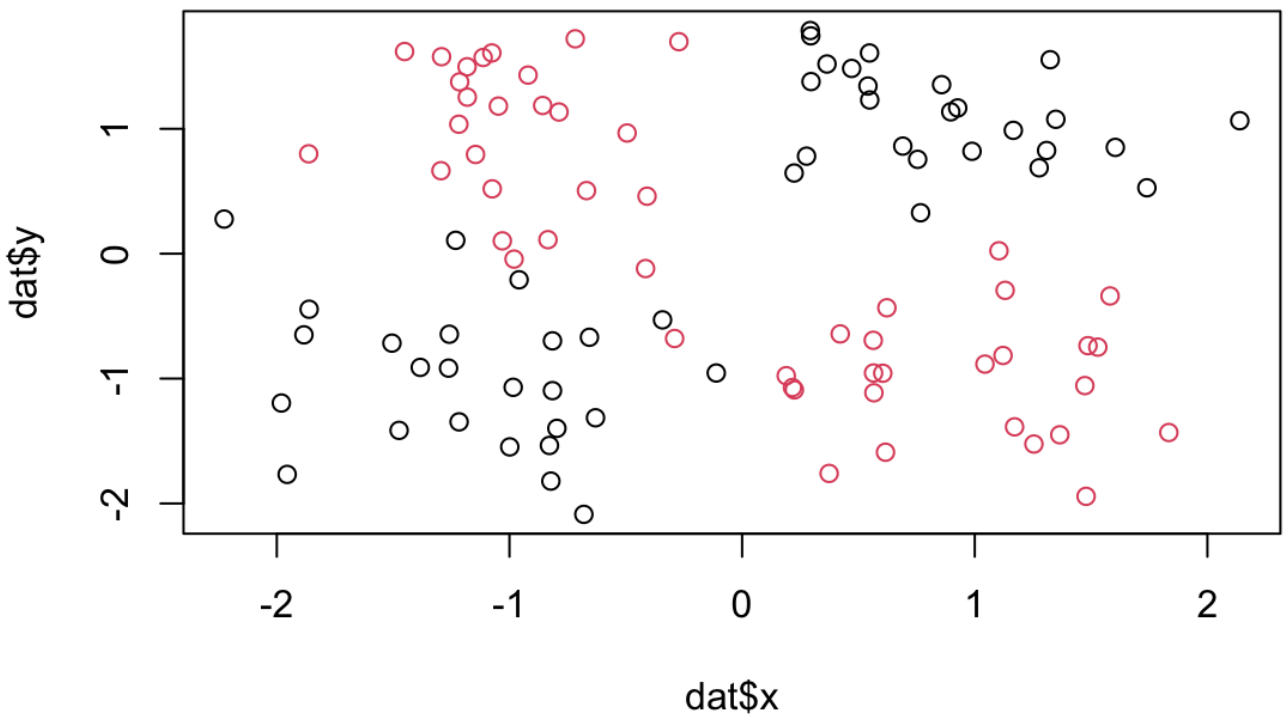
```
dat <- read.csv("./data/XOR_demo.csv", fileEncoding = "utf-8")
dat$label <- as.factor(dat$label)
head(x = dat, n = 5)
```

	x	y	label
	<dbl>	<dbl>	<fctr>
1	0.5477087	1.6073376	1
2	0.2236389	0.6460584	1
3	0.5484982	1.2307716	1
4	0.9882086	0.8199968	1
5	1.1657290	0.9876346	1

5 rows

Hide

```
# この散布図から、事前に適切なif-thenルールを想定してください。
plot(dat$x, dat$y, col = dat$label)
```



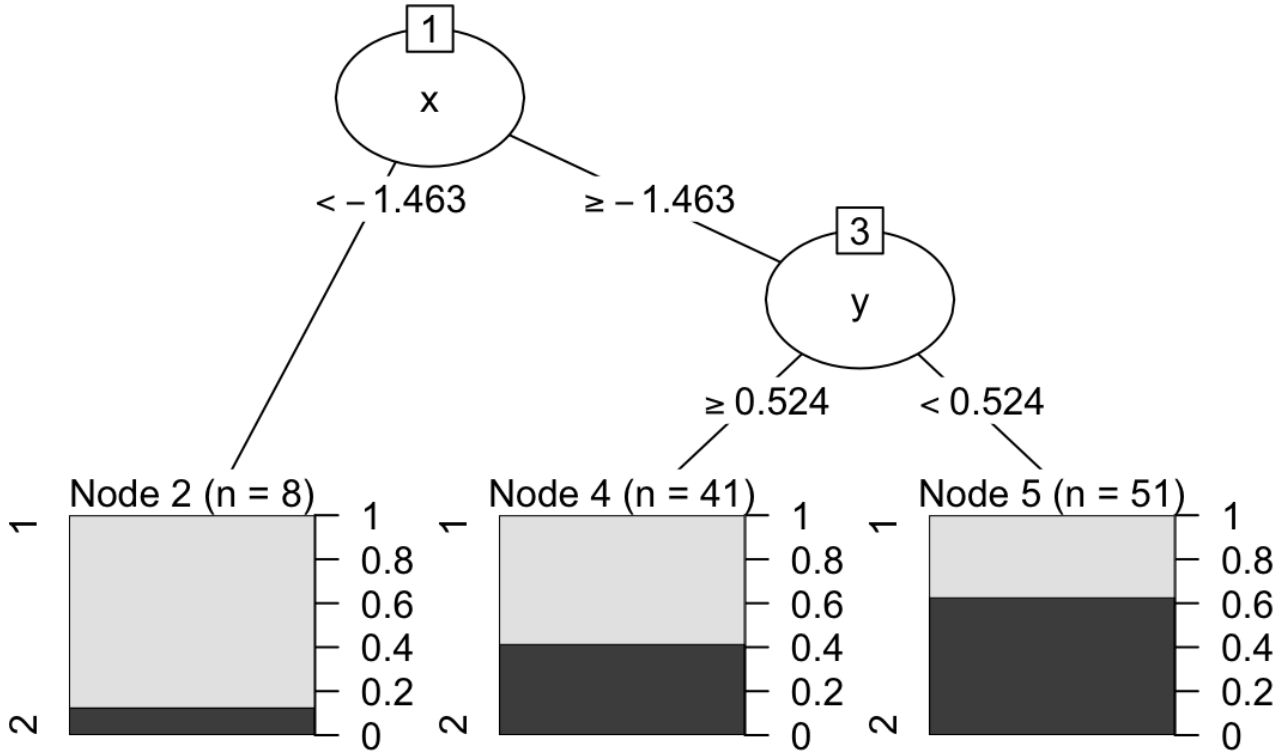
Hide

```
result <- rpart(formula = label ~ x + y,
                data = dat,
                method = "class",
                control = rpart.control(maxdepth = 2,
                                        minsplit = 20,
                                        minbucket = 5))
result
```

```
n= 100

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 100 50 1 (0.5000000 0.5000000)
  2) x< -1.462537 8  1 1 (0.8750000 0.1250000) *
  3) x>=-1.462537 92 43 2 (0.4673913 0.5326087)
    6) y>=0.5235568 41 17 1 (0.5853659 0.4146341) *
    7) y< 0.5235568 51 19 2 (0.3725490 0.6274510) *
```

Hide

```
# 得られた決定木と想定したif-thenルールを比較してみましょう。
plot(as.party(result))
```



3.4 変数重要度

あるノードにおいて、説明変数 x を用いてGiniエントロピーを改善した値は $I(x,t)$ をそのノードの標本サイズとの積、つまり $n(D)I(x,t)$ で表すことができます。これを**total improvement**といいます。説明変数 x の変数重要度は、x が使われたノードにおけるtotal improvementの合計のことです。つまり、変数重要度が大きい説明変数はそれだけ目的変数の予測に寄与していると解釈することができます。

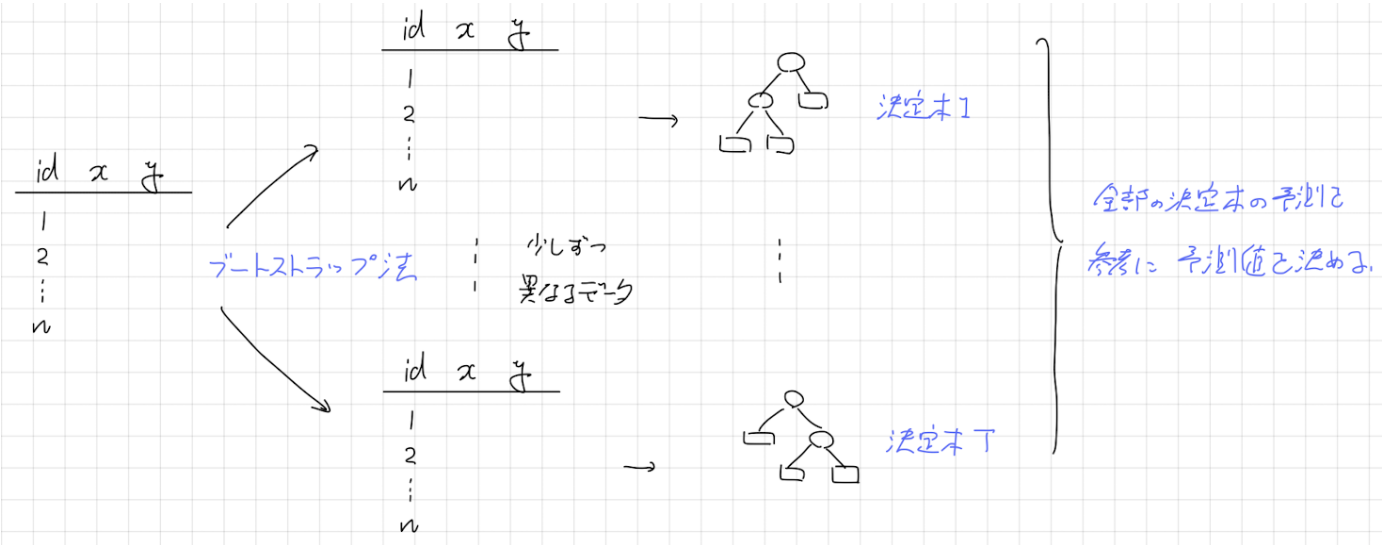
Remark : 実は `rpart` ではもう少し修正を加えた変数重要度の計算を行っていますが、今回は最もシンプルな変数重要度の定義を紹介しました。

第15回 バギング・ランダムフォレスト

Code

1. バギング・ランダムフォレストの概要

バギング（bagging）とは、ブートストラップサンプリングとよばれる方法でデータを複数回複製し、複製したデータたちで作った決定木たちによって予測を決定する方法です。次の図が参考になるでしょう。



今回はバギング、またバギングに修正を加えた**ランダムフォレスト**（random forest）を紹介します。

Remark : 実際にはバギングは決定木以外の手法を使うこともありますが、決定木を用いたほうがバギングという手法が効果的であるため、一般に決定木が用いられています。■

2. ランダムフォレストのデモ

2.1 デモデータ

`data` ディレクトリの `split_demo.csv` をデモデータに用います。このデータは、標本サイズが100で3変数のデータです。今回は、 `x` , `y` を説明変数、 `label` を目的変数としたランダムフォレスト分類を行ってみます。

Hide

```
dat <- read.csv("./data/split_demo.csv", fileEncoding = "utf-8")
dat$label <- as.factor(dat$label)
head(x = dat, n = 5)
```

	x <dbl>	y <dbl>	label <fctr>
1	1.6208878	1.2160687	1
2	0.1418463	1.6881527	1
3	1.1352569	0.3735209	1
4	1.4515459	1.5378424	1

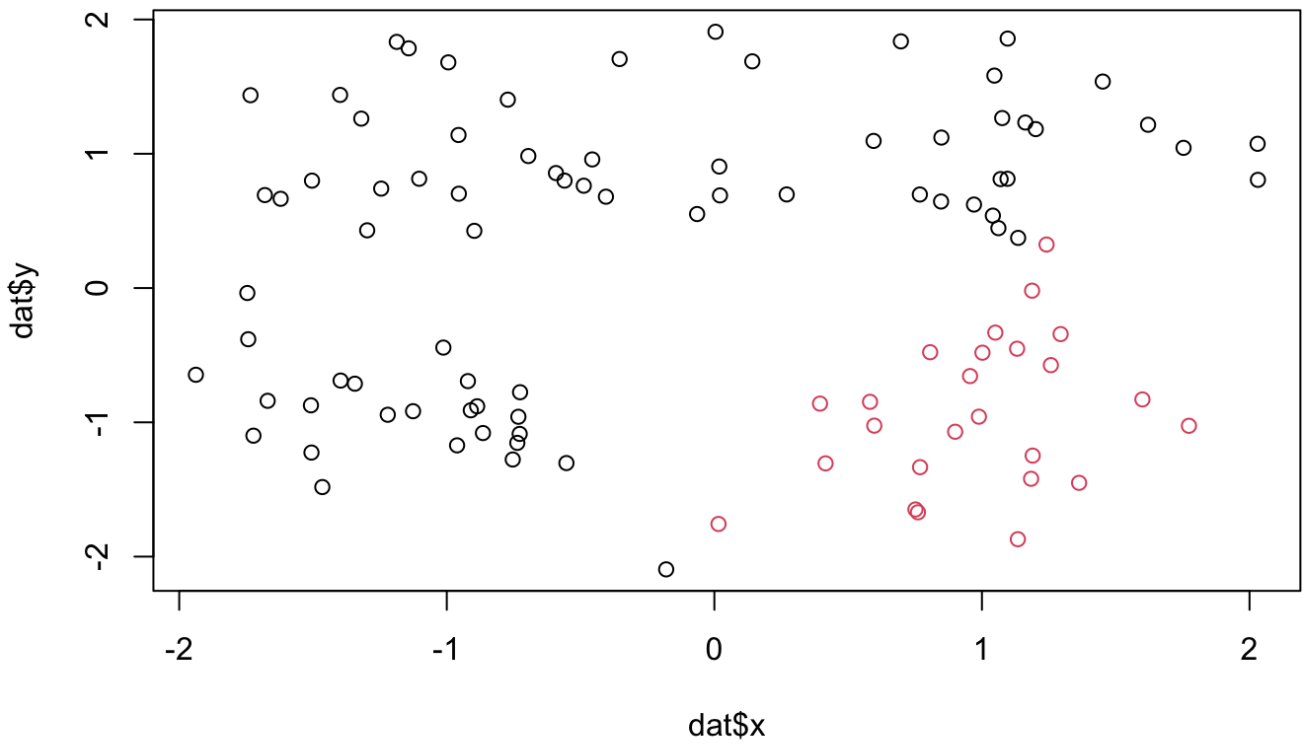
	x	y	label
	<dbl>	<dbl>	<fctr>
5	1.0407771	0.5394458	1

5 rows

`label` 別に色分けした散布図をかいて、標本点の分布を確認しておきましょう。

Hide

```
plot(dat$x, dat$y, col = dat$label)
```



問題：この散布図から、変数 x と変数 y のどちらがよりラベルの予測に貢献するか、また特にその差はないかを想定してください。

2.2 ランダムフォレスト分類の計算

ランダムフォレスト分類は `ranger` パッケージの `ranger` 関数で計算することができます。指定する引数は以下のとおりです。

- `formula` ：目的変数と説明変数を指定する。
- `data` ：決定木を作るために用いるデータ。
- `classification` ：回帰か分類か。目的変数を `factor` 型にcastしているときは、指定しなくても分類になる。
- `importance` ：変数重要度の計算方法
- `write.forest` ：あとで予測を行う場合は `TRUE` にする。
- `num.trees` ：木の数
- `mtry` ：各ノードで分割条件を探す変数の個数
- `max.depth` ：ここに渡した数より深い決定木は作らない。
- `min.node.size` ：ここに渡した数より所属する標本点の個数が少ないノードでは、これ以上分割を試みない。

このうち、本質的に重要なのは `num.trees` 、 `mtry` 、 `max.depth` 、 `min.node.size` です。 `max.depth` と `min.node.size` が作る決定木の複雑度を決めるハイパーパラメータです。 `num.trees` は3.2節で、 `mtry` は3.4節で解説します。

Hide

```
library(ranger)
result <- ranger(formula = label ~ x + y,
                 data = dat,
                 mtry = 1,
                 max.depth = 2,
                 min.node.size = 20,
                 write.forest = TRUE,
                 importance = "impurity",
                 classification = TRUE)    # 今回はなくてよい。
result
```

```
Ranger result

Call:
 ranger(formula = label ~ x + y, data = dat, mtry = 1, max.depth = 2,      min.node.size = 20, write.f
orest = TRUE, importance = "impurity",      classification = TRUE)

Type:                             Classification
Number of trees:                  500
Sample size:                      100
Number of independent variables:  2
Mtry:                             1
Target node size:                 20
Variable importance mode:         impurity
Splitrule:                        gini
OOB prediction error:             3.00 %
```

2.3 変数重要度

各説明変数が目的変数の予測にどれだけ寄与したのかを数値を用いて表現する方法に**変数重要度**（variable importance）があります。詳しくは3.4節で説明しますが、ひとまず値を参照する方法を紹介します。

Hide

```
result$variable.importance
```

```
       x        y
13.73685 12.61893
```

問題 : 前回計算した決定木の変数重要度と比較して、どちらのほうが想定に近いかを確認してみましょう。

3. ランダムフォレスト分類の仕組み

3.1 決定木分析の不安定性

決定木分析の不安定性という性質を説明するために、実験をしてみましょう。以下のような確率分布 D からラベルAが 50 個、ラベルBが 50 個含まれる標本をとり、このラベルを予測する決定木を求めます。これを繰り返し行ったとき、得られる決定木にあまり変化がないか、それとも大きく異なるかに注目してみてください。

確率分布 D は次のとおりです。このスクリプトの内容を理解できる必要はありませんが、興味のある方はフォローしてみてください。

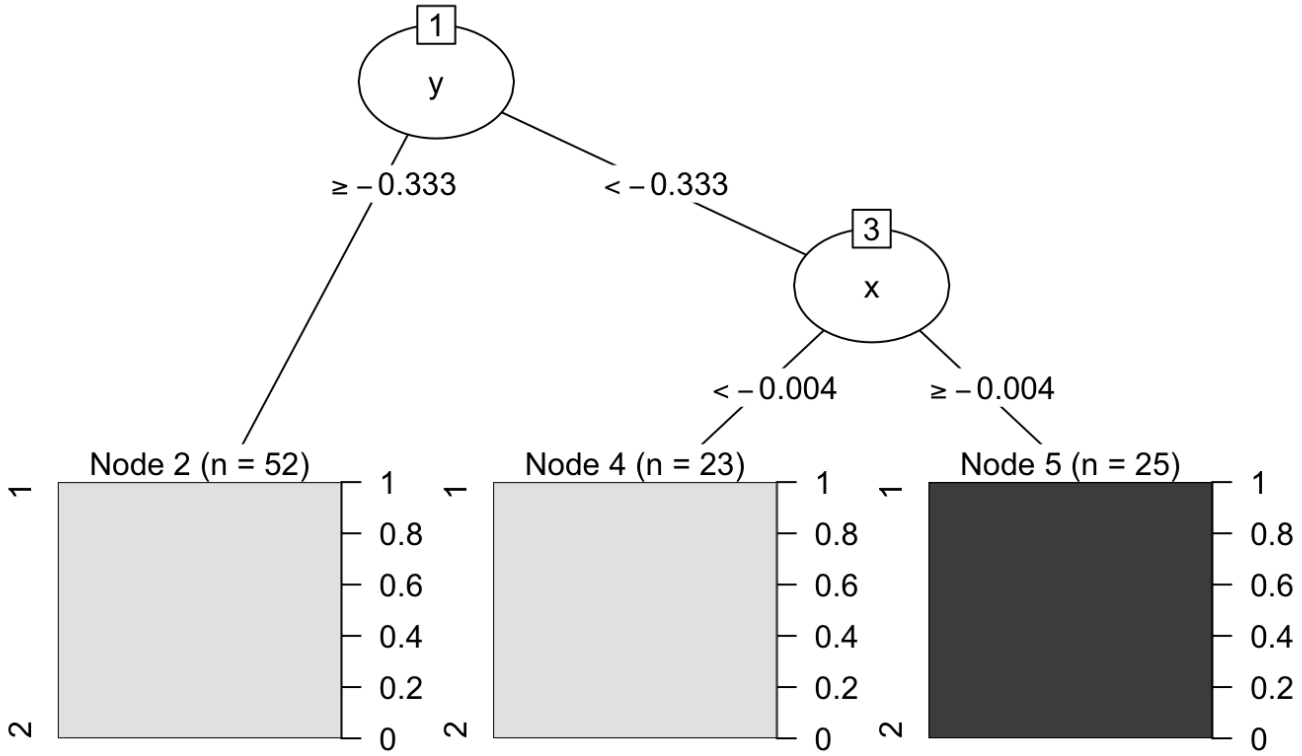
Hide

```
# 確率分布Dからの標本抽出
library(MASS)
library(ggplot2)
sample_demo <- function(){
  A1 <- mvrnorm(n = 25, mu = c(1, 1), Sigma = 0.25*diag(2))
  A2 <- mvrnorm(n = 25, mu = c(-1, -1), Sigma = 0.25*diag(2))
  A3 <- mvrnorm(n = 25, mu = c(-1, 1), Sigma = 0.25*diag(2))
  B1 <- mvrnorm(n = 25, mu = c(1, -1), Sigma = 0.25*diag(2))
  X <- rbind(A1, A2, A3, B1)
  label <- rep(c(1, 2), c(75, 25))

  dat <- data.frame(x = X[, 1], y = X[, 2], label = label)
  return(dat)
}

dat <- sample_demo()
dat$label <- as.factor(dat$label)
qplot(x = x, y = y, color = label, data = dat, geom = "point")
```

それでは、実験してみましょう。以下のスクリプトを繰り返し実行して、得られる決定木が各回ごとに異なるかを確認してみましょう。

Hide

```
# 実験
library(rpart)
library(partykit)
dat <- sample_demo()
dat$label <- as.factor(dat$label)
result <- rpart(formula = label ~ x + y,
                data = dat,
                method = "class",
                control = rpart.control(maxdepth = 3,
                                        minsplit = 10,
                                        minbucket = 3))
plot(as.party(result))
```



問題 : この実験からわかる決定木分析の性質を言葉で説明してみましょう。

解答 : 決定木分析は母集団から得られる標本によって、計算される決定木が大きく変化しうる。この性質を**決定木分析の不安定性**（instability）という。■

3.2 バギングとブートストラップサンプリング

決定木分析には2つの欠点が知られています。 * 十分な予測精度が得られづらい。 * 決定木分析の不安定性により、変数重要度もまた不安定になる。

決定木分析の不安定性を逆手にとり、元のデータとは少し異なるデータを複数回複製し、複製したデータたちで作った決定木たちによって予測を決定することで、 * 決定木分析の予測精度を向上できる * 変数重要度の結果が安定する

のではないかと考えたのがバギングです。ここで、データを複製する方法として**ブートストラップサンプリング**が用いられます。

ブートストラップサンプリングとは、データからデータ点を復元抽出する方法のことです。例えば、次のような操作をやっていることになります。得られる標本をブートストラップ標本といいます。 `ranger` 関数では、 `num.trees` 引数で作るブートストラップ標本の数を指定できます。 `num.trees` は大きい値を設定するほど一般に予測精度が改善します。ただし、

- `num.trees` に十分に大きい値を渡した場合、それ以上大きくしても予測精度の改善は飽和する。
- `num.trees` が大きい値になると、それだけ決定木を作らないといけないので、計算に時間がかかる。

という2点に気をつけてください。この点を踏まえて、大きすぎない `num.trees` を設定すると良いでしょう。

Hide

```
# デモデータ
demo <- data.frame(x = 2*(1:5), y = 3*(1:5))
demo
```

x <dbl>	y <dbl>
2	3
4	6
6	9
8	12
10	15

5 rows

Hide

```
# demoのブートストラップサンプリング
idx_sample <- sample(x = 1:5, size = 5, replace = TRUE)
demo_bootstrap <- demo[idx_sample, ]
demo_bootstrap
```

	x <dbl>	y <dbl>
5	10	15
5.1	10	15
2	4	6
3	6	9
5.2	10	15

5 rows

3.3 バギングの予測と変数重要度の計算

バギングではラベルの予測を作った決定木たちの多数決によって決めます。なお回帰の場合は、多数決のかわりに平均値を用います。また、変数重要度は作った決定木たちの変数重要度の平均値を計算します。

3.4 バギングとランダムフォレストの違い

2001年にL. Breimanは、ブートストラップ標本から得られる決定木の類似性をより小さくする工夫をすれば、予測精度をより向上させられるのではないかと考え、ランダムフォレストを提案しました。このアイディアは、次のような数式でよく説明されます。

$X_1, \cdots, X_B$ を期待値が μ、分散が σ^2 の確率変数とします。またこの標本平均を $\bar{X}$ と表すこととします。確率変数 X_i と X_j の間に正の相関 ρ がある場合、

$$
\begin{aligned}
\mathbb{V}\left[\bar{X}\right] &= \frac{1}{B^2}\mathbb{V}\left[\sum_{b=1}^{B} X_b\right] \\
&= \frac{1}{B^2}\left(\sum_{b=1}^{B}\mathbb{V}[X_i] + \sum_{b\neq b'} Cov(X_b, X_{b'})\right) \\
&= \frac{1}{B^2}\left(B\sigma^2 + B(B-1)\rho\sigma^2\right) \\
&= \frac{1}{B}\sigma^2 + \frac{B-1}{B}\rho\sigma^2 \\
&= \frac{1}{B}(1+(B-1)\rho)\,\sigma^2
\end{aligned}
$$

が成り立ちます。B をブートストラップ標本の数、$\bar{X}$ を予測値、ρ を決定木の予測値の相関係数だと考えてみましょう。すると、

- ブートストラップ標本の数を増やすほど B が大きくなり予測値の分散が小さくなる。
- 作る決定木たちの予測値の相関を小さくするほど予測値の分散が小さくなる。

ことがわかります。B を大きくするというアイディアはバギングで実現できています。一方で、作る決定木たちの予測値の相関を小さくするというアイディアは、バギングでは達成できていないと考えたわけです。

そこでBreimanは、決定木を計算するときに各ノードに現れる説明変数の候補をランダムに選択するという**特徴量のバギング**とよばれるアイディアを導入しました。また、これを取り入れた手法をランダムフォレストとして提案しました。 `ranger` 関数では、 `mtry` 引数がこれに対応します。

Remark : なお、random forestを提案した4年後の2005年にLeo Breimanは亡くなってしまいました。random forestを提案したとき、Breimanはすでに73歳でした。■

第16回 サポートベクトルマシン

Code

1. 概要

サポートベクトルマシン（support vector machine）は、ある変数を他の変数から予測する式を作る手法の一つです。目的変数が量的変数の場合に用いるサポートベクトル回帰と目的変数が質的変数のときに用いるサポートベクトル分類があります。今回は、特に2種類のラベルを目的変数としたサポートベクトル分類に絞って解説します。

2. サポートベクトル分類のデモ

2.1 デモデータ

`data` ディレクトリの `xor_demo_train.csv` をデモデータに用います。このデータは、標本サイズが300で3変数のデータです。 `x1` と `x2` は量的変数、 `y` は `1` と `2` からなる2値の質的変数です。今回は、 `x1` , `x2` を説明変数、 `y` を目的変数としたサポートベクトル分類を行ってみます。

Hide

```
train <- read.csv("./data/xor_demo_train.csv",
                  fileEncoding = "utf-8")
train$y <- as.factor(train$y)
head(x = train, n = 5)
```

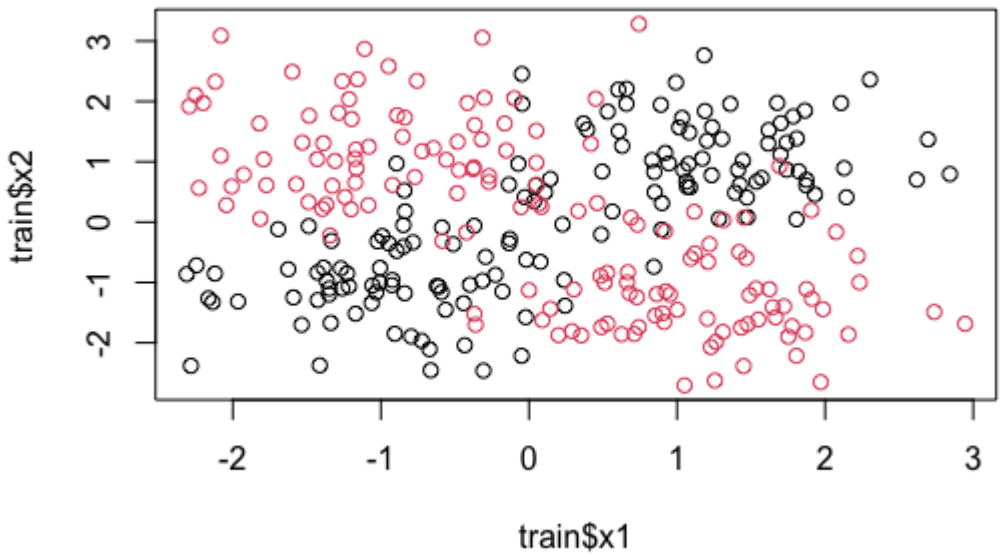
	x1 <dbl>	x2 <dbl>	y <fctr>
1	1.619380	1.5255072	1
2	0.368291	1.6351876	1
3	1.087715	0.5777269	1
4	1.571684	0.7303400	1
5	0.603115	2.1941157	1

5 rows

`y` の値に応じて色分けした散布図をかいて、分布の様子を確認しておきましょう。

Hide

```
plot(train$x1, train$x2, col = train$y)
```



問題：2値の質的変数からなる目的変数 `y` を説明変数 `x` を用いて予測するとき、予測が切り替わる境界線を**決定境界**（decision boundary）といいます。今回のデータに対して適切な決定境界を事前に想定しておきましょう。

2.2 前処理

サポートベクトルマシンは、大きなスケールを持つ変数があると、その変数によって予測を決めてしまう傾向にあります。そのため、予め変数を正規化することが一般的です。今回は例として、min-max正規化を施します。

Hide

```
summary(train)
```

```
       x1                 x2           y
 Min.   :-2.31119   Min.   :-2.70567   1:150
 1st Qu.:-0.93063   1st Qu.:-1.11221   2:150
 Median : 0.05916   Median : 0.17369
 Mean   : 0.10721   Mean   : 0.05168
 3rd Qu.: 1.17379   3rd Qu.: 1.04597
 Max.   : 2.94544   Max.   : 3.28292
```

Hide

```
x1_min <- min(train$x1)
x1_max <- max(train$x1)
x2_min <- min(train$x2)
x2_max <- max(train$x2)
```

Hide

```
x1 <- (train$x1 - x1_min) / (x1_max - x1_min)
x2 <- (train$x2 - x2_min) / (x2_max - x2_min)
train_scaled <- data.frame(x1 = x1, x2 = x2, y = train$y)
head(train_scaled, n = 5)
```

	x1 <dbl>	x2 <dbl>	y <fctr>
1	0.7477357	0.7065401	1
2	0.5097335	0.7248550	1
3	0.6465939	0.5482757	1
4	0.7386622	0.5737597	1
5	0.5544055	0.8181872	1

5 rows

2.3 サポートベクトル分類の計算

サポートベクトル分類は `kernlab` パッケージの `ksvm` 関数で計算することが出来ます。指定する引数は以下のとおりです。

- `x` ：目的変数と説明変数を指定する。
- `data` ：サポートベクトル分類を作るために用いるデータ。
- `type` ：回帰か分類か。分類には二通りあり、 `C_svc` と `nu-svc` があるが、今回は `C-svc` を指定する。
- `C` ：決定境界の複雑度をコントロールするハイパーパラメータの一つです。
- `kernel` ：標本点の類似度を計算する関数（カーネル関数）を指定する。バニラカーネルとよばれる `vanilladot` 、多項式カーネルとよばれる `polydot` 、RBFカーネルとよばれる `rbfdot` などがある。今回は `rbfdot` を指定する。
- `kpar` ： `kernel` で指定したカーネル関数ごとに計算される決定氷塊の複雑度をコントロールするハイパーパラメータがある。RBFカーネル `rbfdot` の場合、 `sigma` を指定する。

わからない単語もあると思いますが、3節で詳しく解説します。

Hide

```
library(kernlab)
result <- ksvm(x = y ~ x1 + x2,
               data = train_scaled,    # 正規化したデータ
               type = "C-svc",
               C = 1.0,
               kernel = "rbfdot",
               kpar = list(sigma = 1.0))
result
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  1

Number of Support Vectors : 113

Objective Function Value : -90.1058
Training error : 0.113333
```

2.4 決定境界の確認

サポートベクトル分類で得られる決定境界の式は複雑なので、数式を確認する代わりに散布図上に図示しましょう。

Hide

```
# メッシュグリッドの作成
xx1 <- seq(-3, 3, 0.05)
xx2 <- seq(-3, 3, 0.05)
meshgrid <- expand.grid(xx1, xx2)
names(meshgrid) <- c("x1", "x2")
head(meshgrid, n = 5)
```

	x1 <dbl>	x2 <dbl>
1	-3.00	-3
2	-2.95	-3
3	-2.90	-3
4	-2.85	-3
5	-2.80	-3

5 rows

Hide

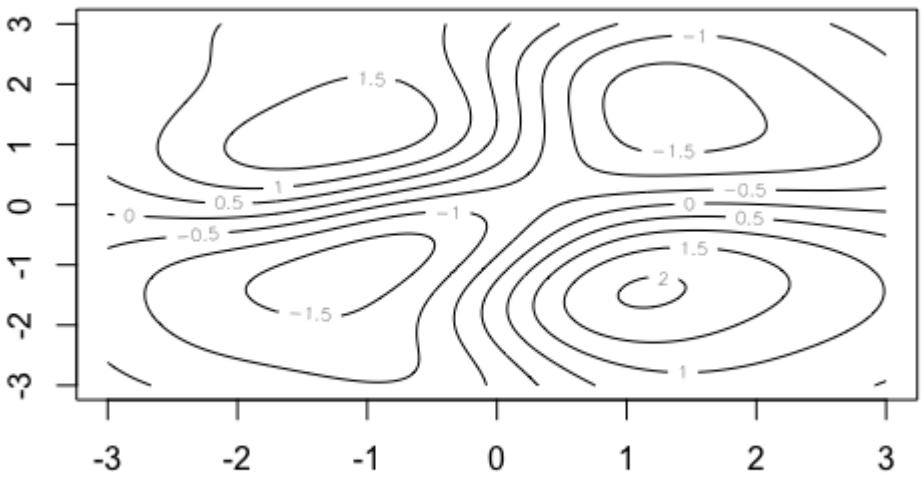
```
# メッシュグリッドデータの正規化
x1 <- (meshgrid$x1 - x1_min) / (x1_max - x1_min)
x2 <- (meshgrid$x2 - x2_min) / (x2_max - x2_min)
meshgrid_scaled <- data.frame(x1 = x1, x2 = x2)
head(meshgrid_scaled, n = 5)
```

	x1 <dbl>	x2 <dbl>
1	-0.13103677	-0.04914855
2	-0.12152497	-0.04914855
3	-0.11201316	-0.04914855
4	-0.10250136	-0.04914855
5	-0.09298956	-0.04914855

5 rows

Hide

```
# 決定関数の計算
pred <- predict(result, meshgrid_scaled, type = "decision")
contour(xx1, xx2,
        array(pred, dim = c(length(xx1), length(xx2))))
```



問題：`C` や `gamma` を動かして、決定関数の等高線をかいてみましょう。

2.5 予測精度の確認と過剰適合・過少適合

RBFカーネルのサポートベクトル分類のように、ハイパーパラメータを調整することで、とても複雑な決定関数を計算できる場合、決定関数を求めるために用いたデータで精度を確認しても、未知のデータに対する予測精度とは大きく結果が乖離してしまうことがあります。このような現象を**過剰適合**（overfit）といいます。逆に単純すぎる決定関数を計算してしまった場合を**過少適合**（underfit）といいます。

問題：線形回帰や決定木分析などでも過剰適合・過少適合が起こります。どのようなケースかを考えてみてください。

このため、一般に予測の精度を確認するためには、モデルの作成には用いないデータを別に準備しておき、このデータで予測精度を確認することが一般的です。これを**ホールドアウト法**（hold-out method）といいます。モデルの作成に用いるデータを**訓練データ**、モデルの予測精度の検討に使うデータを**テストデータ**といいます。

今回はホールドアウト法のために、 `xor_demo_test.csv` を準備しました。

Hide

```
# テストデータの読み込み
test <- read.csv("./data/xor_demo_test.csv",
                 fileEncoding = "utf-8")
test$y <- as.factor(test$y)
head(x = test, n = 5)
```

	x1 <dbl>	x2 <dbl>	y <fctr>
1	-0.03780117	2.0447945	1
2	-0.17821851	2.1147932	1
3	1.33747359	-0.5855833	1
4	1.19782619	2.3928090	1
5	0.88209690	0.8831835	1

5 rows

テストデータにも訓練データと同様の前処理を行っておきます。

Hide

```
# min-max正規化
x1 <- (test$x1 - x1_min) / (x1_max - x1_min)
x2 <- (test$x2 - x2_min) / (x2_max - x2_min)
test_scaled <- data.frame(x1 = x1, x2 = x2, y = test$y)
head(test_scaled, n = 5)
```

	x1 <dbl>	x2 <dbl>	y <fctr>
1	0.4324802	0.7932529	1
2	0.4057677	0.8049416	1
3	0.6941070	0.3540211	1
4	0.6675410	0.8513659	1
5	0.6074779	0.5992821	1

5 rows

では、テストデータに対する予測を行い、精度を評価してみましょう。ラベルの予測の精度を評価するうえでは、正解と予測のクロス集計表をかくのが便利でしょう。これを**混同行列**（confusion matrix）といいます。

Hide

```
pred_test <- predict(result, test_scaled, type = "response")
table(test$y, pred_test)
```

```
   pred_test
     1  2
  1 38 12
  2  7 43
```

参考のため訓練データでの混同行列も計算しておきましょう。

Hide

```
# 訓練データでの混同行列
pred_train <- predict(result, train_scaled, type = "response")
table(train$y, pred_train)
```

```
   pred_train
      1   2
  1 137  13
  2  21 129
```

問題：ハイパーパラメータ `c` や `gamma` を大きくしたり、小さくしたりして、訓練データとテストデータの混同行列がどう変化するかを観察してみてください。

3. サポートベクトル分類の仕組み

3.1 カーネル関数

標本点の間の類似度を測る方法にカーネル関数があります。2つの標本点 x, x' に対してカーネル関数を $k(x, x')$ とおくと、類似度を $\dfrac{k(x, x')}{\sqrt{k(x, x)}\sqrt{k(x', x')}}$ で与えます。なお、どんな標本点 x にも $k(x, x) = 1$ がなりたつようなカーネル関数を正規化カーネル関数といいます。

カーネル関数の厳密な定義を紹介するかわりに、よく用いられるカーネル関数の例を紹介します。

- バニラカーネル : $k(x, x') = x^T x'$
- 多項式カーネル : $k(x, x') = (1 + x^T x')^d$
- RBFカーネル : $k(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$

問題 : RBFカーネルについて以下の問いに答えてください。

1. RBFカーネルが正規化カーネル関数であることを示してください。
2. $x = (1, 1), x' = (1, 1)$ および $x = (1, 1), x' = (1, 2)$の場合にRBFカーネルによる類似度を計算してください。
3. RBFカーネルが0以上1以下の値しか取れないことを示してください。

カーネル関数では、3.2節で述べるようにカーネル関数による標本点の間の類似度を目的変数 y の予測に用います。 `R` 言語の `ksvm` 関数の場合は `type` 引数に `vaniladot` 、 `polydot` 、 `rbfdot` などと指定します。

3.2 カーネルサポートベクトル分類の決定関数

カーネルサポートベクトル分類では、次のような決定関数を考えます。なお、目的変数 y は ± 1 のいずれかの値を取るものと定めます。

$$f(x) = \sum_i \beta_i y_i k(x_i, x) + b$$

そして、$f(x)$ が正だった場合には目的変数の予測値を $+1$、負だった場合には目的変数の予測値を -1 とします。$\beta \geq 0, b$ がそれぞれ決定関数のパラメータで、これを訓練データから求めます。なお、

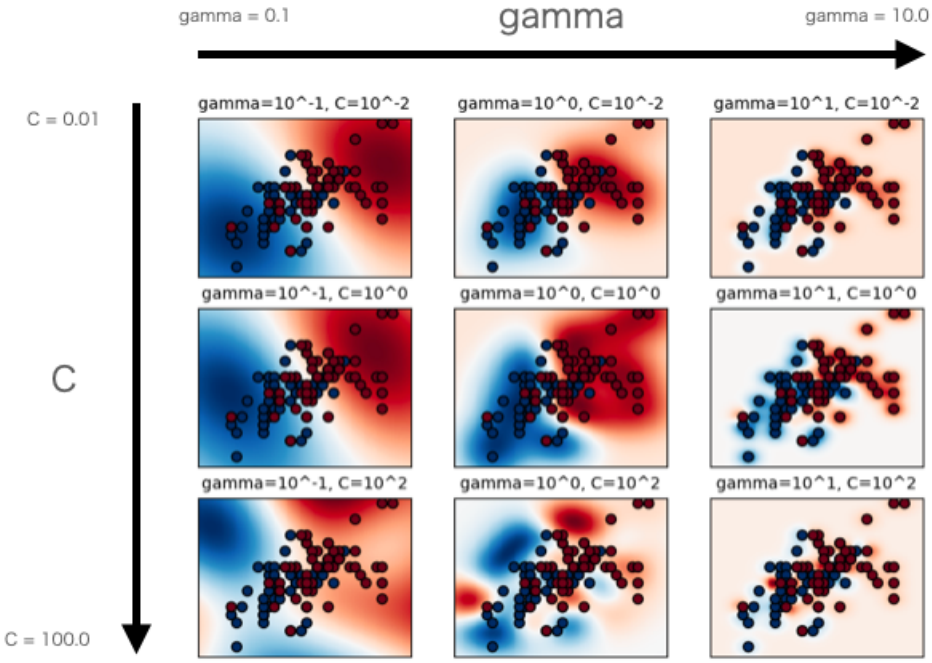
- β_i は必ず 0 以上の値になる。
- カーネル関数 $k(x, x')$ は必ず 0 以上の値をとる。

が成り立ちます。ゆえに、興味のある標本点が β_i の値が大きい標本点と高い類似度をもつとき、目的変数 y の予測はその標本点の目的変数の値になりやすいことが、決定関数の形から確認できます。

Remark : パラメータ β, b の求め方は難しいので、ここでは省略します。興味のある方は講座「カーネル法入門」を参考にしてみてください。■

問題 : 多項式カーネルを用いたサポートベクトル分類器によって、以下のようなデータのラベル y を予測できる決定関数を作りたい。このとき、多項式カーネルの次数ハイパーパラメータ d をいくらにすることが適切であろうか。

なお、RBFカーネルを用いたサポートベクトルマシンには次のような性質があることが知られています。この図から $\gamma = 1/(2\sigma^2)$ の設定によって決定境界がどのように変化するかを考察しておきましょう。



出典 : https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

γ が小さい場合

γ が小さい値のときを考えてみましょう。この場合、Taylor展開を用いることで決定関数を

$$
\begin{aligned}
f(x; w, b) &\sim \sum_{i:\beta_i>0} \beta_i y_i (1 - \gamma \|x_i - x\|^2) + b \\
&= - \sum_{i:\beta_i>0} \beta_i y_i \gamma \|x_i - x\|^2 + b
\end{aligned}
$$

と近似できます。このことから、目的変数の予測は最も近いデータ点 $x_i, \beta_i > 0$ の目的変数の値 y_i によって決まる線形な境界に近くなることがわかります。

γ が大きい場合

γ の値が大きくなると、$\beta_i \neq 0$ をみたすようなデータ点 x_i の周辺に球状の決定境界を作る傾向があることがみてとれます。

3.3 ハイパーパラメータ C の影響

実は、パラメータ β は、

$$0 \leq \beta_i \leq C$$

をみたすように計算されます。このことから、C を調整することで、訓練データの標本点が予測に与える影響度を制御することができます。

第17回 順伝播型ニューラルネットワーク

Code

1. 概要

順伝播型ニューラルネットワーク（feedfoward neural network）は、ある変数を他の変数から予測する式を作る手法の一つです。目的変数が量的変数の場合に用いるニューラルネットワーク回帰と目的変数が質的変数のときに用いるニューラルネットワーク分類があります。今回は、特に2種類のラベルを目的変数としたニューラルネットワーク分類に絞って解説します。

2. ニューラルネットワーク分類のデモ

2.1 パッケージのインストール

今回は、 `neuralnet` パッケージを用います。このパッケージはCRANには公開されていない（つまり `install.packages` 関数ではインストールできない）最新版のパッケージを用いる必要があるため、次のような方法でインストールします。

Hide

```
# パッケージのインストールと読み込み
install.packages("devtools")
devtools::install_github("bips-hb/neuralnet")    # 最新版のneuralnetパッケージ
```

インストールが完了したら、 `neuralnet` パッケージを読み込みましょう。

Hide

```
library(neuralnet)
```

2.2 デモデータ

`data` ディレクトリの `xor_demo_train.csv` をデモデータに用います。このデータは、標本サイズが300で3変数のデータです。 `x1` と `x2` は量的変数、 `y` は `1` と `2` からなる2値の質的変数です。今回は、 `x1` , `x2` を説明変数、 `y` を目的変数としたニューラルネットワーク分類を行ってみます。

Hide

```
train <- read.csv("./data/xor_demo_train.csv",
                  fileEncoding = "utf-8")
train$y <- as.factor(train$y)
head(x = train, n = 5)
```

	x1 <dbl>	x2 <dbl>	y <fctr>
1	1.619380	1.5255072	1
2	0.368291	1.6351876	1

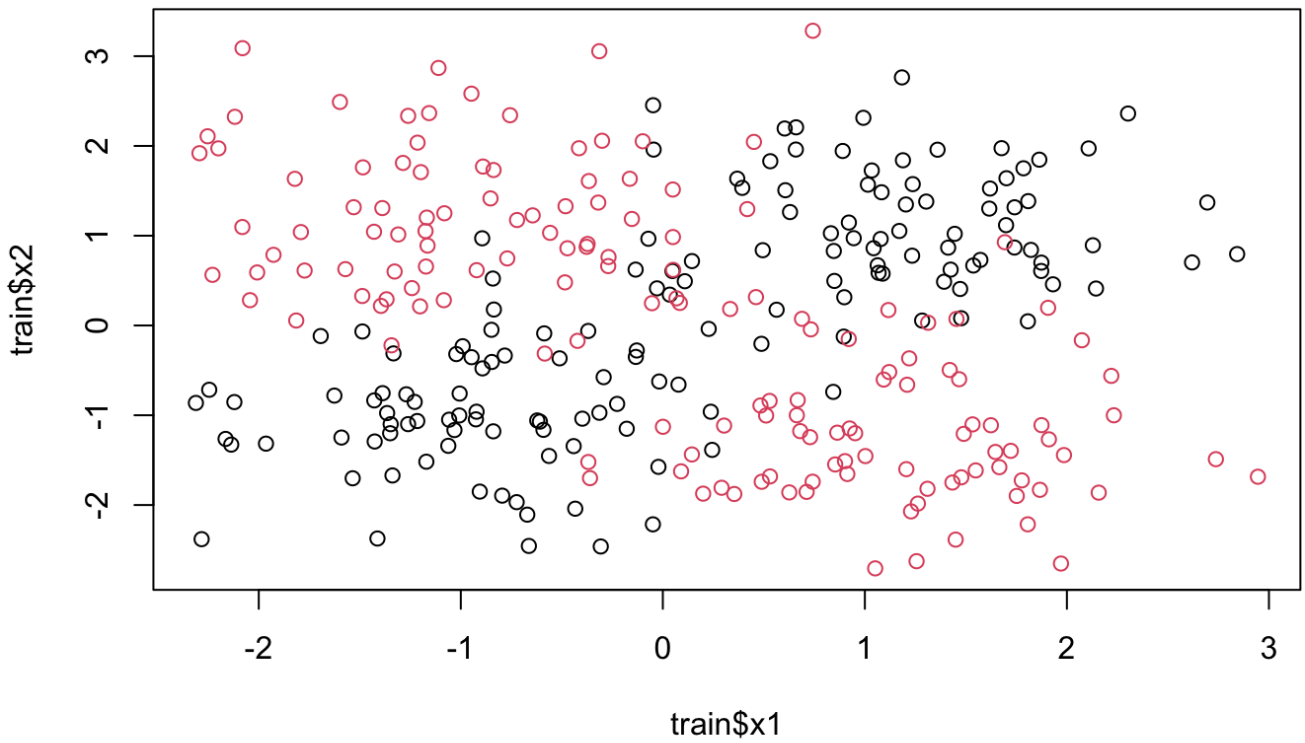
	x1 <dbl>	x2 <dbl>	y <fctr>
3	1.087715	0.5777269	1
4	1.571684	0.7303400	1
5	0.603115	2.1941157	1

5 rows

`y` の値に応じて色分けした散布図をかいて、分布の様子を確認しておきましょう。

Hide

```
plot(train$x1, train$x2, col = train$y)
```



問題：2値の質的変数からなる目的変数 `y` を説明変数 `x` を用いて予測するとき、予測が切り替わる境界線を**決定境界**（decision boundary）といいます。今回のデータに対して適切な決定境界を事前に想定しておきましょう。

2.3 ニューラルネットワーク分類の計算

ニューラルネットワーク分類は `neuralnet` パッケージの `neuralnet` 関数で計算することが出来ます。指定する引数は以下のとおりです。

- `formula` ：目的変数と説明変数を指定する。
- `data` ：サポートベクトル分類を作るために用いるデータ。
- `hidden` ：隠れ層とそのユニットの数
- `linear.output` ：特殊な引数のため、解説を省略します。今回は `FALSE` に指定しておいてください。
- `act.fct` ：隠れ層の活性化関数
- `output.act.fct` ：出力層の活性化関数
- `algorithm` ：パラメータを求めるためのアルゴリズムの指定

- `err.fct` ：損失関数
- `stepmax` ：パラメータの更新回数の最大値
- `threshold` ：パラメータを更新したとき、損失関数の値の差がこの値を下回った場合、更新をやめる。
- `learningrate` ：更新の大きさ（学習率という）

わからない単語もあると思いますが、3節で詳しく解説します。

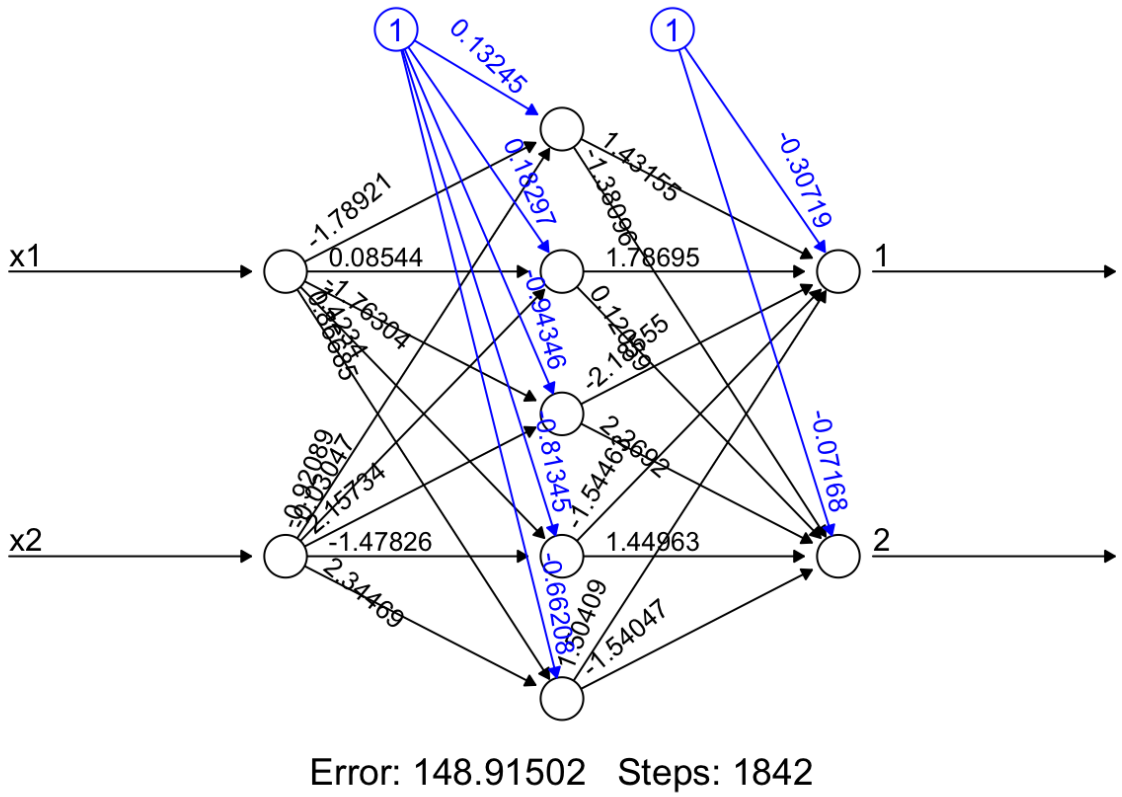
Hide

```
# 隠れ層1層のニューラルネットワークの学習
nnet <- neuralnet(formula = y ~ x1 + x2,
                  data = train,
                  # アーキテクチャの設計
                  hidden = 5,
                  linear.output = FALSE,    # act.fct, output.act.fctを使用するため
                  act.fct = "relu",
                  output.act.fct = "logistic",
                  # 学習アルゴリズムの設計
                  algorithm = "backprop",
                  err.fct = "ce",    # 2乗誤差ならsse
                  stepmax = 1e+04,
                  threshold = 0.5,
                  learningrate = 0.0005)
```

`plot` 関数を用いて得られたモデルの式を確認できます。

Hide

```
# 結果の可視化
plot(nnet)
```



2.4 決定境界の確認

ニューラルネットワーク分類で得られる決定境界の式は複雑なので、数式を確認する代わりに散布図上に図示しましょう。

Hide

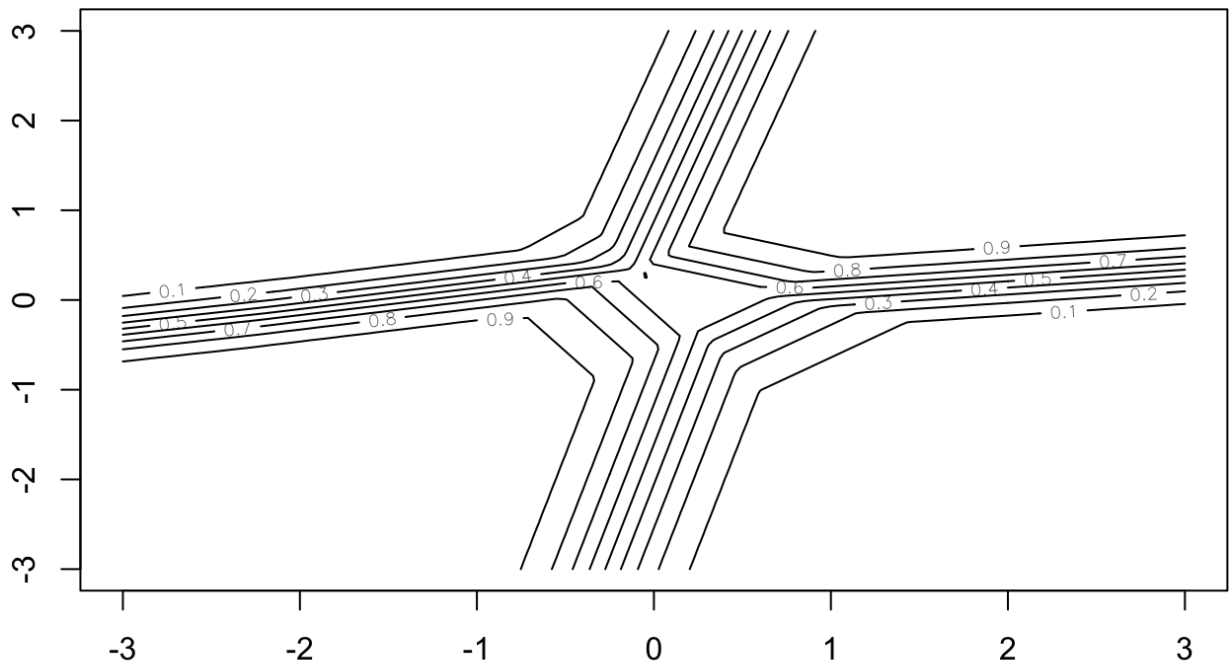
```
# メッシュグリッドの作成
xx1 <- seq(-3, 3, 0.05)
xx2 <- seq(-3, 3, 0.05)
meshgrid <- expand.grid(xx1, xx2)
names(meshgrid) <- c("x1", "x2")
head(meshgrid, n = 5)
```

	x1 <dbl>	x2 <dbl>
1	-3.00	-3
2	-2.95	-3
3	-2.90	-3
4	-2.85	-3
5	-2.80	-3

5 rows

Hide

```
# 決定関数の計算
pred <- predict(nnet, meshgrid)
contour(xx1, xx2,
        array(pred, dim = c(length(xx1), length(xx2))))
```



2.5 予測精度の確認と過剰適合・過少適合

ニューラルネットワーク分類は、隠れ層とそのユニットの数 `hidden` を調整することで、とても複雑な決定関数を作ることができます。このような場合、決定関数を求めるために用いたデータに対しては精度を確認しても、未知のデータに対する予測精度とは大きく結果が乖離してしまうことがあります。このような現象を**過剰適合**（overfit）といいます。逆に単純すぎる決定関数を計算してしまった場合を**過少適合**（underfit）といいます。

このため、一般に予測の精度を確認するためには、モデルの作成には用いないデータを別に準備しておき、このデータで予測精度を確認することが一般的です。これを**ホールドアウト法**（hold-out method）といいます。モデルの作成に用いるデータを**訓練データ**、モデルの予測精度の検討に使うデータを**テストデータ**といいます。

今回はホールドアウト法のために、 `xor_demo_test.csv` を準備しました。

Hide

```
# テストデータの読み込み
test <- read.csv("./data/xor_demo_test.csv",
                 fileEncoding = "utf-8")
test$y <- as.factor(test$y)
head(x = test, n = 5)
```

	x1 <dbl>	x2 <dbl>	y <fctr>
1	-0.03780117	2.0447945	1
2	-0.17821851	2.1147932	1
3	1.33747359	-0.5855833	1
4	1.19782619	2.3928090	1
5	0.88209690	0.8831835	1

5 rows

では、テストデータに対する予測を行い、精度を評価してみましょう。ラベルの予測の精度を評価するうえでは、正解と予測のクロス集計表をかくのが便利でしょう。これを**混同行列**（confusion matrix）といいます。

Hide

```
# テストデータでの混同行列
pred_test_proba <- predict(nnet, test)
pred_test <- apply(pred_test_proba, 1, which.max)
table(test$y, pred_test)
```

```
   pred_test
     1  2
  1 36 14
  2  7 43
```

参考のため訓練データでの混同行列も計算しておきましょう。

Hide

```
# 訓練データでの混同行列
pred_train_proba <- predict(nnet, train)
pred_train <- apply(pred_train_proba, 1, which.max)
table(train$y, pred_train)
```

```
   pred_train
      1   2
  1 134  16
  2  19 131
```

3. 順伝播型ニューラルネットワークの仕組み

3.1 決定関数の表現の仕方

ユニット（unit）

順伝播型ニューラルネットワークの最小構成単位はユニットとよばれるものです。ユニットは、複数の値 $x_1, \cdots, x_d$ を入力されたら、次のような式を計算します。

$$u(b + w_1 x_1 + \cdots + w_d x_d)$$

ここで、u は活性化関数とよばれる非線形な関数で、様々な種類のものから予め使うものを決めておきます。一方、$w_1, \cdots, w_d$ および b は重み・バイアスとよばれ、データから求めるパラメータになります。

活性化関数

活性化関数の代表例には、ReLU（rectified linear unit）やlogistic関数（sigmoid関数ともいう）があります。それぞれのグラフを示します。

層（layer）

順伝播型ニューラルネットワークでは、複数のユニットをまとめた層とよばれるものを複数積むことで決定関数を定義します。数式で表すと複雑になりがちなので、次のようなグラフで表現することが一般的です。

最初の層は説明変数の値を入力する入力層、最後の層は予測値を出力する出力層といいます。入力層と出力層の間にある層を隠れ層といいます。 `neuralnet` 関数では `hidden` に値を渡すことで隠れ層のユニットの数を指定できます。複数の隠れ層を積む場合は、 `c(10, 5)` のようにかくことで、10個の隠れ層と5個の隠れ層を準備することが出来ます。

隠れ層のユニットの活性化関数にはReLUが用いられることが多いです。 `neuralnet` 関数では `act.fct` に relu` を渡すことで隠れ層のユニットの活性化関数をReLUに指定できます。

隠れ層の役割

隠れ層のユニットは、複雑な決定境界をつくるために役に立ちます。実際、ある程度の複雑な決定境界であれば、隠れ層に十分にたくさんのユニットを準備することで、所望の精度で近似できることが知られています。これを**万能近似定理**（universal approximation theorem）といいます。

次の問題を解いて、隠れ層のユニットが決定境界の複雑度に対応していることを確認してみましょう。

問：以下の順伝播型ニューラルネットワークが定める決定関数の決定境界を示してください。

Remark : 同様に万能近似定理が成り立つものに、RBFカーネルのサポートベクトルマシンがあります。ただし、RBFカーネルのサポートベクトルマシンでは近似できない決定境界のなかに、順伝播型ニューラルネットワークでは近似できる関数が存在することが知られています。

3.2 パラメータの求め方

交差エントロピー損失

2値のラベル $0/1$ のどちらかの値をとる変数 y を予測するモデル $p(x; w, b)$ をニューラルネットワークで作りたいとします。このとき、予測モデル $p(x; w, b)$ の出力層の活性化関数にはlogistic関数を用いることで、$y = 1$ の確信度を出力できるようなモデルを作ります。 `neuralnet` 関数では `output.act.fct` に `logistic` を渡すことで出力層の活性化関数をlogistic関数に指定できます。

このとき、モデルの予測と目的変数 y の食い違いを測る方法として、**交差エントロピー損失**（crossentropy loss）があります。

$$l(y, p) = \begin{cases} -\log p & \text{if } y = 1 \\ -\log(1-p) & \text{if } y = 0 \end{cases}$$

予測モデルは重み w とバイアス b によって決まります。これらの値は、与えられたデータに対して交差エントロピー損失の平均値が最小になるようなものを求めます。 `neuralnet` 関数では `err.fct` に `ce` を渡すことで損失関数を交差エントロピーに指定できます。

勾配降下法

重みとバイアスは、最初は初期値をランダムに振っておき、良い予測を返せるような値に少しずつ更新していくことで求めます。 `neuralnet` 関数では `stepmax` がこの更新の最大回数を決める引数、 `theshold` が更新前後の交差エントロピーの値の差をみて、更新を停止するための条件を与える引数になっています。

パラメータ（重みとバイアス）を求める代表的な手法は**勾配降下法**（gradient descent）です。

$$w^{(k+1)} = w^{(k)} - \eta \frac{\partial l}{\partial w}(w^{(k)})$$

`neuralnet` 関数では `algorithm` に `backprop` を渡すことで勾配降下法によるパラメータの計算が実行できます。 勾配降下法の η は**学習率**（learning rate）といい、更新の大きさを決める値です。この値は事前に自分で決めておきます。 `neuralnet` 関数では `learningrate` がこの更新の最大回数を決める引数になっています。

第18回 共分散構造分析（前半）

Code

1. 概要と例題

各変数間の複雑な関係を**パス図**とよばれる図で表しておき、変数間の関係の強さを表す**パス係数**を分散共分散行列をヒントに求める手法を**共分散構造分析**といいます。今回は、

- パス図とはなにか？
- パス係数とはなにか？共分散構造分析では、どうやって求めるのか？
- 得られた結果の読み方

を知るために、次の例題を考えてみましょう。

例題　S大学のデータサイエンス学部では、1年次に「初級統計学」「基礎からのR」、2年次に「続初級統計学」「数理統計学」の講義を開講しています。I教授は、期末試験の結果を用いて、この4科目の理解度の関係を説明できるようなモデルを作ろうと考えました。以下の問に答えてください。

[1]　I教授は以下のような仮説を立てました。この仮説をもとにパス図をかいてください。

1. 「初級統計学」と「基礎からのR」の間には相関関係がある。
2. 2年次の講義の理解度はいずれも、1年次の講義の理解度によって変化する。

[2]　[1]のパス図が与える構造方程式を述べてください。また構造方程式のパラメータ（外生変数の分散、パス係数と誤差の分散）をすべて挙げてください。

[3]　[2]の構造方程式が定める「初級統計学」「基礎からのR」「続初級統計学」「数理統計学」の分散共分散行列を求めてください。

[4]　共分散構造分析には、構造方程式のパラメータ（外生変数の分散、パス係数と誤差の分散）を求める代表的な方法に、最尤法と最小二乗法の2通りがあります。このうち、最小二乗法とはどのような計算を行うものかを説明してください。

[5]　R言語では `lavaan` パッケージの `sem` 関数を用いて[4]の計算を行うことができます。今回、25名の学生について4科目の成績を `data` ディレクトリに `seiseki.csv` という名前のcsvファイルに記録しました。このファイルと `sem` 関数を用いて構造方程式のパラメータを推定してください。

[6]　次の空欄に当てはまる数値を答えてください：「数理統計学」は「初級統計学」と「基礎からのR」から（ア）だけ説明することができる。「初級統計学」が1点高い生徒は、「数理統計学」が（イ）点高くなる傾向にある。

2. パス図

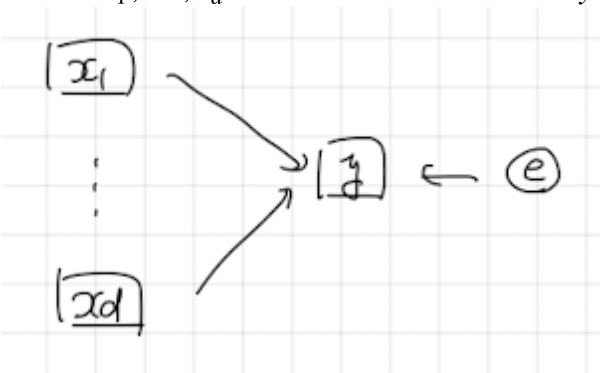
パス図は以下の2つを組み合わせて作る変数間の関係を表す図のことです。

1. 2変数 x と y の相関関係を想定する場合。

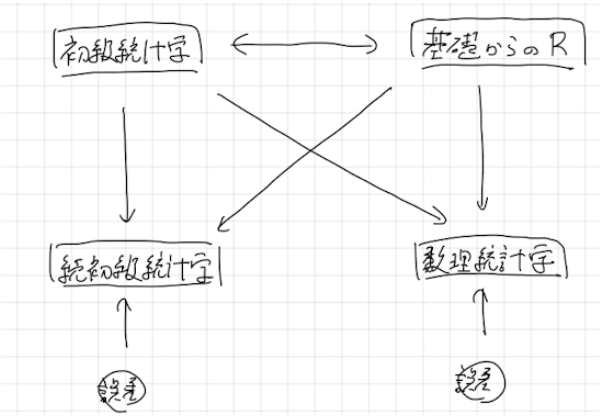
2. 変数 y を変数 $x_1, \cdots, x_d$ によって説明する式

$$y = w_0 + w_1 x_1 + \cdots + w_d x_D + e$$

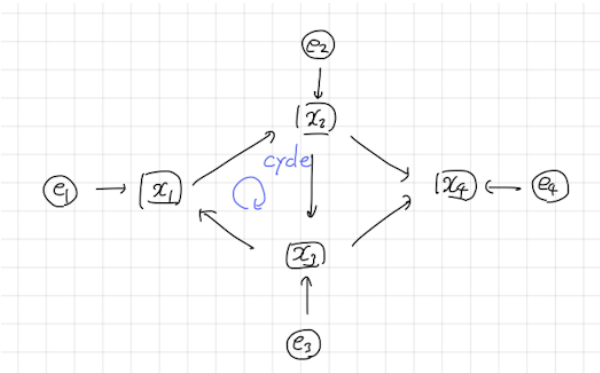
を想定する場合。ここで、e は変数 $x_1, \cdots, x_d$ では説明できなかった変数 y の**誤差**を表すとします。



例題の[1]の解答は次のようになります。



Remark　共分散構造分析では、逐次モデルとよばれるパス図のみを対象にします。逐次モデルとはcycleが含まれないようなパス図のことです。



3. 構造方程式とそのパラメータ

パス図が与える構造方程式とは、パス図で表現した変数間の関係を数式で表したもののことです。「初級統計学」をx_1、「基礎からのR」をx_2、「続初級統計学」をx_3、「数理統計学」をx_4と表すことにして、例題の[2]の解答を与えます。

$$
\begin{aligned}
x_3 &= w_{30} + w_{31}x_1 + w_{32}x_2 + e_3 \\
x_4 &= w_{41} + w_{41}x_1 + w_{42}x_2 + e_4
\end{aligned}
$$

ただしx_1とx_2の分散をそれぞれσ_1^2, σ_2^2、x_1とx_2の共分散を$Cov(x_1, x_2) = \sigma_{12}$、誤差$e_3$と$e_4$をそれぞれ期待値が0かつ分散が$\sigma_3^2, \sigma_4^2$とします。

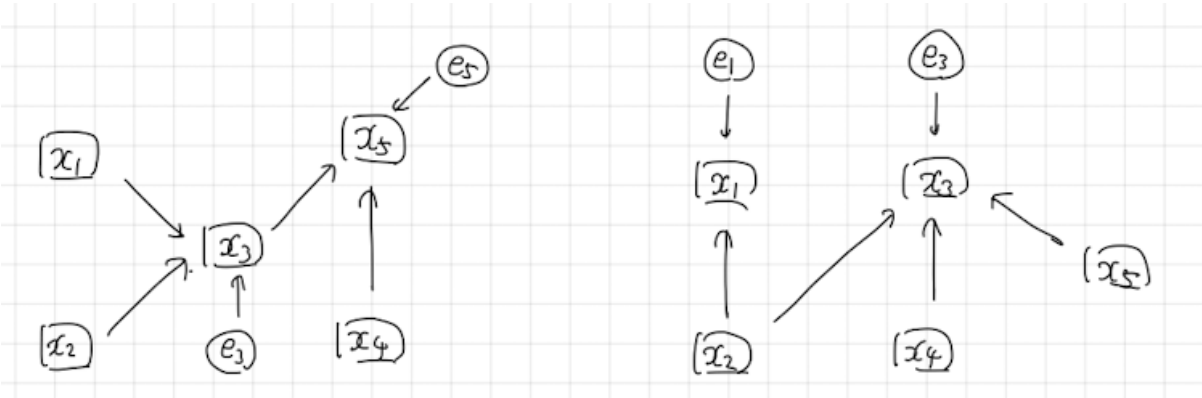
このうち、構造方程式のパラメータは$w_{31}, w_{32}, w_{41}, w_{42}, \sigma_{12}, \sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2$の9個（切片$w_{30}, w_{40}$が含まれないことに注意）で、

- 外生変数の分散：σ_1^2, σ_2^2
- パス係数：$w_{31}, w_{32}, w_{41}, w_{42}, \sigma_{12}$

- 誤差の分散 : σ_3^2, σ_4^2

といいます。

なんらかの変数によって回帰されている変数を**内生変数**、どの変数からも回帰されていない変数のことを**外生変数**といいます。以下のパス図の各変数を内生変数と外生変数に分類してみましょう。



4. 構造方程式が定める分散共分散行列

例題の[2]で解答した構造方程式から、4科目の分散共分散行列は構造方程式のパラメータ（外生変数の分散、パス係数と誤差の分散）を用いて次のように表すことができます。なお、下三角は省略しました。

$$\begin{pmatrix} \sigma_1^2 & c & w_{31}\sigma_1^2 + w_{32}c & w_{41}\sigma_1^2 + w_{42}c \\ & \sigma_2^2 & w_{31}c + w_{32}\sigma_2^2 & w_{41}c + w_{42}\sigma_2^2 \\ & & w_{31}^2\sigma_1^2 + 2w_{31}w_{32}c + w_{32}^2\sigma_2^2 + \sigma_3^2 & w_{31}w_{41}\sigma_1^2 + (w_{31}w_{42} + w_{41}w_{32})c + w_{32}w_{42}\sigma_2^2 \\ & & & w_{41}^2\sigma_1^2 + 2w_{41}w_{42}c + w_{42}^2\sigma_2^2 + \sigma_4^2 \end{pmatrix}$$

5. 共分散構造分析の最小2乗法

共分散構造分析には、構造方程式のパラメータ（外生変数の分散、パス係数と誤差の分散）を推定する方法の代表例として、最尤法と最小二乗法があります。最小二乗法では、

- データ（標本）の分散共分散行列 S
- 構造方程式から求まる分散共分散行列 Σ

の各要素の差の2乗和 $\sum_{i \geq j} (\Sigma_{ij} - S_{ij})^2$ が最小になるように構造方程式のパラメータを求めます。

6. R言語を用いた共分散構造分析の計算

6.1 `lavaan` パッケージのインストール

R言語で共分散構造分析を計算するためのパッケージには、 `lavaan` と `sem` が有名ですが、今回は `lavaan` を用いていきましょう。また、 `lavaan` の結果を確認するために、 `semPlot` パッケージを用います。 `lavaan` パッケージと `semPlot` パッケージをインストールし、読み込んでおいてください。

Hide

```
# パッケージのインストールと読み込み
# install.packages("lavaan")
# install.packages("semPlot")
library(lavaan)
library(semPlot)
```

```
Registered S3 method overwritten by 'htmlwidgets':
  method           from
  print.htmlwidget tools:rstudio
Registered S3 method overwritten by 'data.table':
  method           from
  print.data.table
```

6.2 データの読み込み

今回、学生の4科目の成績を `seiseki.csv` という名前のcsvファイルに記録しました。このファイルを読み込んで、先頭5行を確認してみましょう。

Hide

```
# データの読み込み
dat <- read.csv("./data/seiseki.csv", fileEncoding = "utf-8")
head(dat, n = 5)
```

	初級統計学 <int>	基礎からのR <int>	続初級統計学 <int>	数理統計学 <int>
1	78	83	66	46
2	59	76	73	34
3	71	76	62	42
4	69	84	60	44
5	63	71	58	41

5 rows

データの分散共分散行列も確認しておきましょう。

Hide

```
cov(dat)
```

```
             初級統計学 基礎からのR 続初級統計学 数理統計学
初級統計学     41.91000    5.400000      15.6200  38.698333
基礎からのR     5.40000   16.666667      12.8000   2.558333
続初級統計学   15.62000   12.800000     101.6733   9.930000
数理統計学     38.69833    2.558333       9.9300  51.610000
```

6.3 構造方程式のコーディング

sem 関数には構造方程式を渡す必要があります。相関関係は ~~ 、回帰の関係は ~ を用いて表します。今回の構造方程式は次のようにコーディングできます。

Hide

```
model <- "
  # 変数間の相関関係
  初級統計学 ~~ 基礎からのR
  続初級統計学 ~~ 0*数理統計学      # 相関がないことを明示する。

  # 構造方程式
  続初級統計学 ~ 初級統計学 + 基礎からのR
  数理統計学 ~ 初級統計学 + 基礎からのR
"
```

6.4 構造方程式のパラメータの推定

sem 関数にデータ dat と構造方程式のコード model を渡すと、構造方程式のパラメータが計算できます。なお、 sem 関数では最尤法を用いています。

Hide

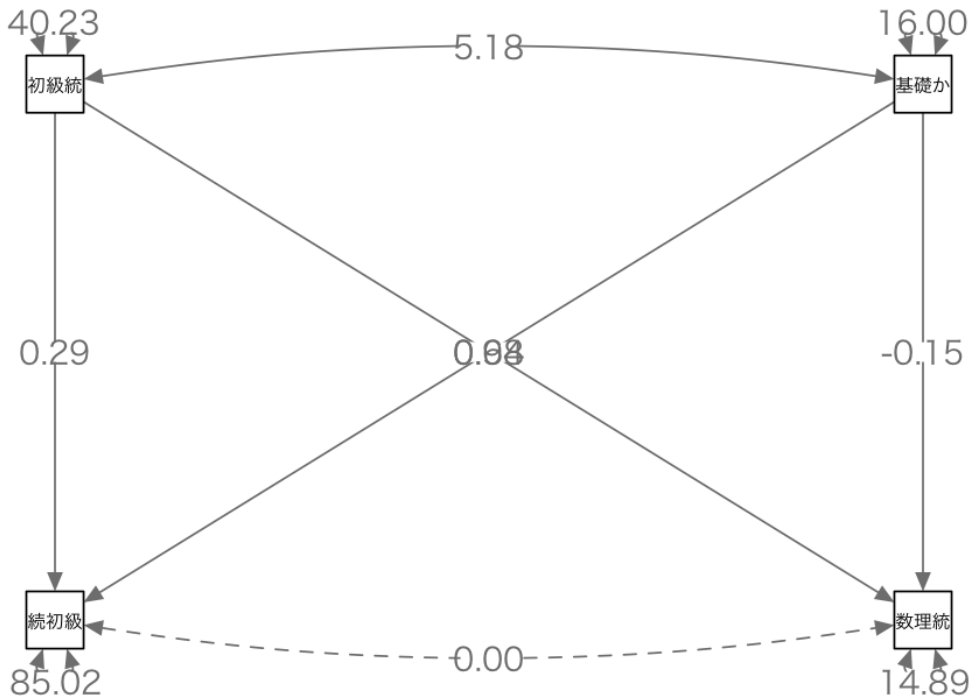
```
result <- sem(model = model, data = dat)
summary(result, rsquare = TRUE)
```

```
lavaan 0.6-9 ended normally after 32 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                         9

  Number of observations                            25

Model Test User Model:

  Test statistic                                 0.149
  Degrees of freedom                                 1
  P-value (Chi-square)                           0.700

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  続初級統計学 ~
    初級統計学        0.286    0.297    0.962    0.336
    基礎からのR       0.675    0.471    1.434    0.152
  数理統計学 ~
    初級統計学        0.943    0.124    7.587    0.000
    基礎からのR      -0.152    0.197   -0.771    0.441

Covariances:
                    Estimate  Std.Err  z-value  P(>|z|)
  初級統計学 ~~
    基礎からのR        5.184    5.179    1.001    0.317
 .続初級統計学 ~~
   .数理統計学         0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .続初級統計学     85.023   24.048    3.536    0.000
   .数理統計学       14.888    4.211    3.536    0.000
    初級統計学       40.234   11.380    3.536    0.000
    基礎からのR      16.000    4.525    3.536    0.000

R-Square:
                   Estimate
    続初級統計学      0.129
    数理統計学        0.700
```

Hide

```
par(family = "ヒラギノ角ゴシック W3")
semPaths(result, "model", "est", sizeMan = 5, edge.label.cex = 1.0)
```

```
abbreviate(Labels, nCharNodes) で警告がありました:
  ASCII 文字でないものが省略名として使われました
```



例題の[5]の解答は次のようになります。

- 外生変数の分散 : $\sigma_1^2 = 40.234, \sigma_2^2 = 16.00$
- パス係数 : $w_{31} = 0.286, w_{32} = 0.675, w_{41} = 0.943, w_{42} = -0.152, \sigma_{12} = 5.184$
- 誤差の分散 : $\sigma_3^2 = 85.023, \sigma_4^2 = 14.888$

7. 共分散構造分析の推定結果の解釈

例題の[6]に答える形で、共分散構造分析の推定結果の解釈を与えます。

7.1 決定係数

空欄（ア）についてです。（ア）は、

$$\text{続初級統計学} = w_{30} + w_{31} \times \text{初級統計学} + w_{32} \times \text{基礎からのR} + e_3$$

の決定係数を聞かれています。第6節の結果から、この回帰式の決定係数は 0.129 だとわかります。この値を、決定係数の定義から直接計算してみましょう。決定係数の定義は

$$\text{決定係数} = 1 - \frac{\text{誤差の分散}}{\text{続初級統計学の分散}}$$

です。回帰式の誤差の分散は $\sigma_3^2 = 85.023$ と推定されています。一方で、続初級統計学の分散は

$$\mathbb{V}[\text{続初級統計学}] = w_{31}^2\sigma_1^2 + w_{32}^2\sigma_2^2 + 2w_{31}w_{32}\sigma_{12} + \sigma_3^2$$

です。Rを用いて値を求めてみましょう。

Hide

```
0.286^2*40.234 + 0.675^2*16.00 + 2*0.286*0.675*5.184 + 85.023
```

あとは、決定係数の定義から次のようにして計算できます。

Hide

```
1 - 85.023 / 97.60552
```

7.2 パス係数の解釈

空欄の（イ）に入るのは、パス係数 w_{31} の推定値、つまり 0.286 です。

第19回 共分散構造分析（後半）

Code

1. 概要と例題

今回は共分散構造分析のワンポイントアドバイスや注意点を紹介します。

1. 共分散構造分析に現れる変数は必ずしもデータに記録されている変数（観測変数）だけとは限りません。探索的因子分析に現れる共通因子のように、観測変数の共通性を意味する変数を作ることができます。これを**潜在変数**といいます。そこで、潜在変数を含む共分散構造分析のやり方を紹介します。

2. どんな構造方程式に対しても共分散構造分析の結果が得られるとは限りません。パラメータが推定可能かを調べる一つの方法である**自由度**を紹介します。

3. 共分散構造分析によって得られたモデルがどれだけデータに当てはまっているかを調べる方法として**適合度指標**を紹介します。

例題　S大学のデータサイエンス学部では、プログラミング言語系の講義として「基礎からのR」「基礎からのPython」、統計学の講義として「初級統計学」「続初級統計学」が開講されています。I教授はプログラミングのスキルが統計学の理解度にどれだけ影響があるのかを調べようと考えています。以下の問に答えてください。

[1]　データに観測されている変数を**観測変数**、観測されていない変数を**潜在変数**といいます。潜在変数「プログラミングのスキル」「統計学の理解度」および観測変数「基礎からのR」「基礎からのPython」「初級統計学」「続初級統計学」の間の関係を表すパス図を示してください。

[2]　潜在変数「プログラミングのスキル」「統計学の理解度」を定義する測定方程式と、これらの関係を示す構造方程式を述べてください。また測定方程式と構造方程式のパラメータ（外生変数の分散、パス係数と誤差の分散）をすべて挙げてください。

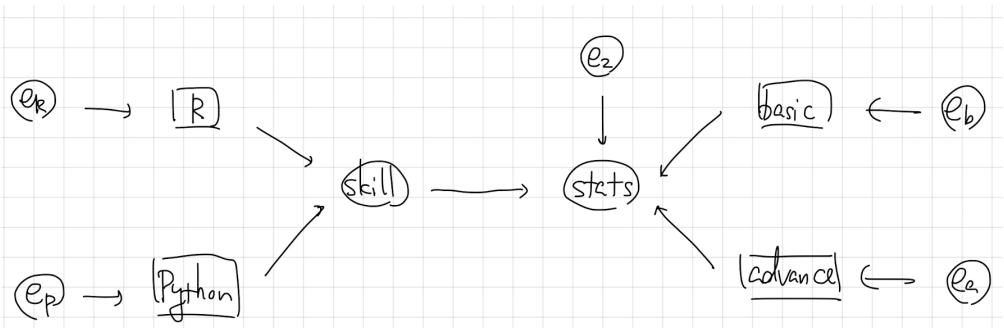
[3]　このモデルの自由度を述べよ。

[4]　今回、25名の学生について4科目の成績を `data` ディレクトリに `skill_stats.csv` という名前のcsvファイルに記録しました。このファイルとR言語の `lavaan` パッケージにある `sem` 関数を用いて測定方程式と構造方程式のパラメータを推定してください。

[5]　モデルがデータにどれだけあてはまっているかを確認する方法に、**適合度指標**があります。適合度指標の例にGFIがあります。GFIの最小二乗法の場合の計算方法を説明してください。

2. 潜在変数を含む場合のパス図

潜在変数はパス図では、丸印で記されます。また潜在変数はデータに記録されていないので、観測変数を説明する関係を定義することで測定します。例題の[1]の場合は、次のような解答になります。



3. 測定方程式

潜在変数が観測変数を説明する式を**測定方程式**といいます。例えば「プログラミングのスキル」（以下、skillと略）という潜在変数は2つの観測変数「基礎からのR」（以下、Rと略）と「基礎からのPython」（以下、Pythonと略）に対して、

$$\mathrm{R} = w_{R0} + w_{R1} \times \mathrm{skill} + e_R$$
$$\mathrm{Python} = w_{P0} + w_{P1} \times \mathrm{skill} + e_P$$

という方程式を測定方程式として準備します。ここで注意してほしいのは、このままだと潜在変数skillの平均と分散が一意に決まらないことです。そこで、一般的には潜在変数の平均を0とします。また分散を決めるために、w_{r1}かw_{p1}のいずれかの係数を1に固定します。

以上の議論から、測定方程式は次のようになります。なお、以下では「統計学の理解度」をstats、「初級統計学」をbasic、「続初級統計学」をadvanceと略すことにします。

$$\mathrm{R} = w_{R0} + \mathrm{skill} + e_R$$
$$\mathrm{Python} = w_{P0} + w_{P1} \times \mathrm{skill} + e_P$$
$$\mathrm{basic} = w_{b0} \times \mathrm{stats} + e_b$$
$$\mathrm{advance} = w_{a0} + w_{a1} \times \mathrm{stats} + e_a$$

なお、誤差e_Rの分散をσ_R^2と表すことにし、その他の誤差の分散も同様の記号を用いることにします。またskillの分散をσ_1^2とします。構造方程式は、

$$\mathrm{stats} = w_{21} \times \mathrm{skill} + e_2$$

です。なお、誤差e_sの分散をσ_2^2と表すことにします。切片項がないのは潜在変数の平均も誤差の平均も0と仮定することから、切片項が必ず0になることが従うためです。

また、測定方程式と構造方程式のパラメータは

- パス係数 : w_{P1}, w_{a1}, w_{21}
- 誤差の分散 : σ_R^2, σ_P^2, σ_b^2, σ_a^2, σ_1^2, σ_2^2

の9個あります。

3. 自由度

分散共分散行列の対角成分と上三角部分の要素の個数に比べて、推定するべきパラメータの個数が多いと、パラメータの値を推定することができなくなります。変数の個数をdと表すとき、分散共分散行列の対角成分と上三角部分の要素の個数は$\frac{1}{2}d(d+1)$です。推定するべきパラメータの個数をpと表すとき、

$$df = \frac{1}{2}d(d+1) - p$$

をモデルの**自由度**といいます。言い換えれば自由度が負ならば、パラメータが推定できません。

4. R言語を用いた共分散構造分析の計算

4.1 `lavaan`パッケージの読み込み

Hide

```
# パッケージの読み込み
library(lavaan)
library(semPlot)
```

4.2 データの読み込み

今回、学生の4科目の成績を `skill_stats.csv` という名前のcsvファイルに記録しました。このファイルを読み込んで、先頭5行を確認してみましょう。

Hide

```
# データの読み込み
dat <- read.csv("./data/skill_stats.csv", fileEncoding = "utf-8")
head(dat, n = 5)
```

	r <dbl>	python <dbl>	basic <dbl>	advance <dbl>
1	3.2	4.8	3.2	4.0
2	3.5	3.7	2.9	2.5
3	6.1	4.9	5.5	6.6
4	3.5	3.5	2.3	3.7
5	6.7	5.5	5.4	5.5

5 rows

データの分散共分散行列も確認しておきましょう。

Hide

```
# データの分散共分散行列
cov(dat)
```

```
               r    python     basic  advance
r       2.109167 1.2765833 1.2381667 1.796667
python  1.276583 1.2922333 0.9079667 1.210917
basic   1.238167 0.9079667 1.2446000 1.361000
advance 1.796667 1.2109167 1.3610000 2.745000
```

4.3 構造方程式のコーディング

`sem` 関数には構造方程式を渡す必要があります。相関関係は `~~` 、回帰の関係は `~` を用いて表します。今回の構造方程式は次のようにコーディングできます。

Hide

```
model <- "
  skill =~ 1*r + python
  stats =~ 1*basic + advance

  stats ~ skill
"
```

4.4 構造方程式のパラメータの推定

`sem` 関数にデータ `dat` と構造方程式のコード `model` を渡すと、構造方程式のパラメータが計算できます。なお、 `sem` 関数では最尤法を用いています。

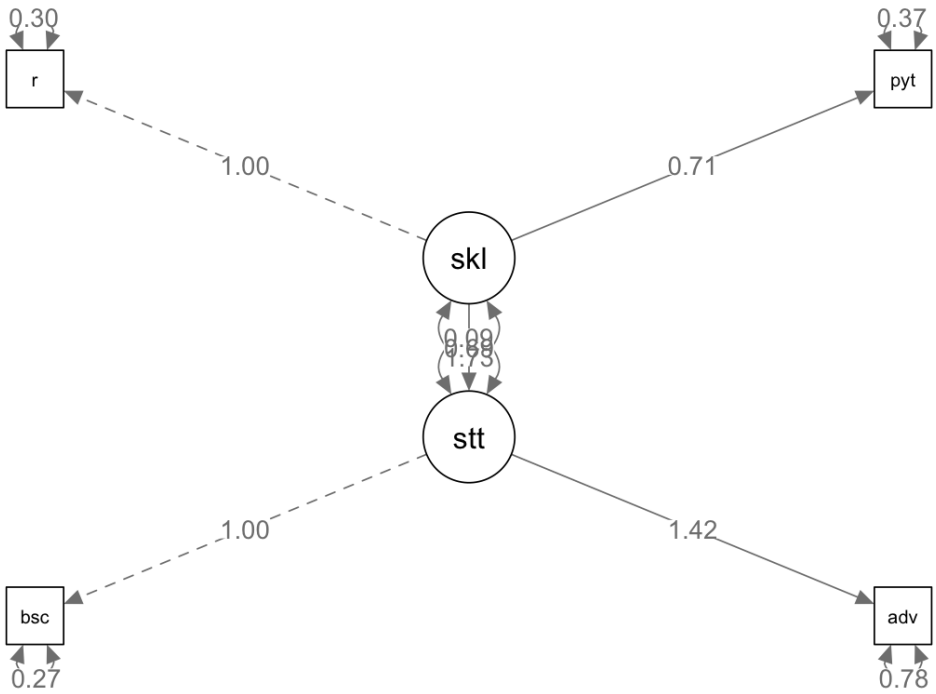
Hide

```
result <- sem(model = model, data = dat)
summary(result)
```

```
lavaan 0.6-9 ended normally after 26 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                         9

  Number of observations                            25

Model Test User Model:

  Test statistic                                 0.373
  Degrees of freedom                                 1
  P-value (Chi-square)                           0.541

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  skill =~
    r                 1.000
    python            0.709    0.125    5.666    0.000
  stats =~
    basic             1.000
    advance           1.417    0.268    5.291    0.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  stats ~
    skill             0.694    0.126    5.513    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .r                 0.295    0.193    1.531    0.126
   .python            0.372    0.137    2.721    0.006
   .basic             0.273    0.127    2.147    0.032
   .advance           0.784    0.300    2.611    0.009
    skill             1.730    0.593    2.918    0.004
   .stats             0.089    0.121    0.733    0.464
```

Hide

```
semPaths(result, "model", "est", sizeMan = 5, edge.label.cex = 1.0)
```



4.5 適合度指標の計算

Hide

```
fitmeasures(result, fit.measures = "gfi")
```

```
  gfi
0.993
```

5. 適合度指標

定義した測定方程式と構造方程式の妥当性を検討するヒントとして、適合度指標とよばれるものを参考にすることがあります。適合度指標の代表例はGFI（Goodness of Fit Index）で、最小二乗法によって得られるモデルの場合、これは

$$\mathrm{GFI} = 1 - \frac{\text{標本分散共分散とモデルの分散共分散の要素の差の2乗和}}{\text{標本分散共分散の要素の2乗和}}$$

で定義されます。決定係数とよく似た定義です。0以上1以下の値をとり、1に近いほど標本によくあてはまっているモデルと考えることができます。他にも、似たような定義のもとにSRMR（Standardized Rood Mean Squared Error）などがあります。