

with R

, , ,

2021-07-18



# Contents

		<b>7</b>
<b>1</b>		<b>9</b>
<b>I 1 -</b>		<b>11</b>
<b>2</b>		<b>13</b>
2.1	R	13
2.2		13
2.3		14
2.4		18
<b>3</b>		<b>21</b>
3.1	R	21
3.2		21
3.3		36
<b>4</b>		<b>41</b>
4.1	R	41
4.2		41
4.3		49
<b>II 2 -</b>		<b>59</b>
<b>5</b>		<b>61</b>
5.1	R	61
5.2		61
5.3		62
<b>6</b>		<b>77</b>
6.1	R	77
6.2		77

6.3	.....	85
6.4	.....	91
<b>7</b>		<b>99</b>
7.1	.....	99
7.2 R	.....	99
7.3	.....	99
7.4	.....	105
7.5	.....	111
7.6	.....	113
7.7	.....	117
<b>8</b>		<b>121</b>
8.1 CART	.....	121
8.2 R package	.....	121
8.3 CART	.....	121
8.4	.....	127
8.5 R	.....	134
<b>9</b>		<b>137</b>
9.1	.....	137
9.2 R package	.....	137
9.3 SVM -	.....	137
9.4 SVM -	.....	142
9.5 SVM	.....	146
9.6 R SVM	.....	150
<b>10</b>		<b>155</b>
10.1 R	.....	155
10.2	.....	155
10.3 ,	.....	156
10.4 ROC	.....	158
10.5	.....	160
<b>III 3 -</b>		<b>165</b>
<b>11</b>		<b>167</b>
11.1 R	.....	167
11.2	.....	167
11.3	.....	168
11.4	.....	173
<b>12</b>		<b>179</b>
12.1 R	.....	179
12.2	.....	179
12.3	.....	179

<b>CONTENTS</b>	<b>5</b>
12.4 . . . . .	188
12.5 - . . . . .	197
12.6 . . . . .	202
<b>13</b>	<b>207</b>
13.1 R package . . . . .	207
13.2 K-means . . . . .	207
13.3 K-medoids . . . . .	213
13.4 K-means . . . . .	230
13.5 . . . . .	236
<b>14</b>	<b>253</b>
14.1 R package . . . . .	253
14.2 . . . . .	253
14.3 . . . . .	266
<b>IV 4 -</b>	<b>267</b>
<b>15</b>	<b>269</b>
15.1 R package . . . . .	269
15.2 . . . . .	269
15.3 . . . . .	272
15.4 . . . . .	277
<b>16</b>	<b>287</b>
16.1 R package . . . . .	287
16.2 . . . . .	287
16.3 . . . . .	293
16.4 . . . . .	297



- R CRAN . R 4.1.0 version ,  
R , R . R
- R for Data Science (by Hadley Wickham & Garrett Grolemund): <https://r4ds.had.co.nz>
  - Advanced R (by Hadley Wickham): <https://adv-r.hadley.nz>



# Chapter 1

, (prediction), (classification), (clustering),  
(association rule)



# **Part I**

**1 -**



# Chapter 2

## 2.1 R

package	version
tidyverse	1.3.1
stats	4.1.0
broom	0.7.8

## 2.2

$$n \quad k \quad (\mathbf{x}) \quad (y)$$

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + \epsilon_i \quad (2.1)$$

$$, \quad \epsilon_i \sim N(0, \sigma^2)$$

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \quad (2.2)$$

,

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$$

$$\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

,

$$E[\epsilon] = \mathbf{0}, \text{Var}[\epsilon] = \sigma^2 \mathbf{I}$$

## 2.3

### 2.3.1

#### 2.3.1.1 R

```
10 (age), (height), (weight)

train_df <- tribble(
  ~age, ~height, ~weight,
  21, 170, 60,
  47, 167, 65,
  36, 173, 67,
  15, 165, 54,
  54, 168, 73,
  25, 177, 71,
  32, 169, 68,
  18, 172, 62,
  43, 171, 66,
  28, 175, 68
)
```

Table 2.1: , ,

(age)	(height)	(weight)
21	170	60
47	167	65
36	173	67
15	165	54
54	168	73
25	177	71
32	169	68
18	172	62
43	171	66
28	175	68

```

knitr::kable(
  train_df, booktabs = TRUE,
  align = c('r', 'r', 'r'),
  col.names = c(' (age)', ' (height)', ' (weight)'),
  caption = ' ', ,
)

```

```

lm_fit <- lm(weight ~ age + height, data = train_df)

coef(lm_fit)

## (Intercept)           age         height
## -108.1671993     0.3291212     0.9552913

vcov(lm_fit)

##             (Intercept)           age         height
## (Intercept) 1774.3280624 -0.671107283 -10.264885141
## age          -0.6711073   0.004794717   0.003035476
## height       -10.2648851   0.003035476   0.059566804

```

```

40, 170           95%
predict(lm_fit, newdata = tibble(age = 40, height = 170),
        interval = "confidence", level = 0.95)

```

```

##      fit      lwr      upr

```

```
## 1 67.39718 65.01701 69.77735
```

### 2.3.1.2

$$\mathbf{x}_0 \quad , \quad \hat{y}_0 = \mathbf{x}_0^\top \hat{\beta} \quad (2.3)$$

(2.3) .

$$Var(\hat{y}_0) = \mathbf{x}_0^\top Var(\hat{\beta}) \mathbf{x}_0 \quad (2.4)$$

$$= \sigma^2 \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_0 \quad (2.5)$$

(2.5)  $\sigma^2$  *MSE* (mean squared error)

$$\hat{Var}(\hat{y}_0) = MSE \times \left( \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_0 \right) \quad (2.6)$$

Table 2.1  $\hat{\beta}$  *MSE*

```
n <- nrow(train_df)
X <- cbind(
  intercept = rep(1, n),
  train_df[, c("age", "height")] %>% as.matrix()
)
y <- train_df[, c("weight")] %>% as.matrix()
k <- ncol(X) - 1

# regression coefficient
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y

# response estimate
y_hat <- X %*% beta_hat

# MSE
MSE <- sum((y - y_hat) ^ 2) / (n - k - 1)
```

, 40, 170  $\mathbf{x}_0$   $\hat{y}_0$  95%

```
# variance of y_hat on new_x
new_x <- matrix(c(
  intercept = 1,
  age = 40,
  height = 170
), ncol = 1)
```

```

new_y_hat <- t(new_x) %*% beta_hat

var_new_y_hat <- MSE * t(new_x) %*% solve(t(X) %*% X) %*% new_x
se_new_y_hat <- sqrt(var_new_y_hat)

lci <- new_y_hat + qt(0.025, n - k - 1) * se_new_y_hat
uci <- new_y_hat + qt(0.975, n - k - 1) * se_new_y_hat

str_glue("({format(lci, nsmall = 3L)}, {format(uci, nsmall = 3L)})")

## (65.01701, 69.77735)

```

## 2.3.2

### 2.3.2.1 R

2.3.1 ,  $(\hat{y}_0 = 40, \hat{y}_{170} = 170)$  .  
 2.3.1

```

predict(lm_fit, newdata = tibble(age = 40, height = 170),
        interval = "prediction", level = 0.95)

##      fit      lwr      upr
## 1 67.39718 60.68745 74.1069

```

### 2.3.2.2

$$y_0 \quad , \quad \quad \quad , \quad \quad \quad (2.5) \quad \sigma^2$$

$$Var(y_0 - \hat{y}_0) = Var(y_0) + Var(\hat{y}_0) \quad (2.7)$$

$$= \sigma^2 + \sigma^2 \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_0 \quad (2.8)$$

$$= \sigma^2 \left( 1 + \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_0 \right) \quad (2.9)$$

$$(2.9) \quad \sigma^2 \quad MSE \quad .$$

$$\hat{Var}(y_0 - \hat{y}_0) = MSE \times \left( 1 + \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_0 \right) \quad (2.10)$$

$$, \quad 40, \quad 170 \quad \mathbf{x}_0 \quad \quad \quad y_0 \quad 95\% \quad .$$

```

# variance of y_hat on new_x
new_x <- matrix(c(
  intercept = 1,

```

```

age = 40,
height = 170
), ncol = 1)

new_y_hat <- t(new_x) %*% beta_hat

var_new_y_pred <- MSE * (1 + t(new_x)) %*% solve(t(X) %*% X) %*% new_x
se_new_y_pred <- sqrt(var_new_y_pred)

lci <- new_y_hat + qt(0.025, n - k - 1) * se_new_y_pred
uci <- new_y_hat + qt(0.975, n - k - 1) * se_new_y_pred

str_glue("({format(lci, nsmall = 3L)}, {format(uci, nsmall = 3L)})")

## (60.68745, 74.1069)

```

## 2.4

### 2.4.1 R

$(TS)$ $(2mm, 6mm)$	$(CT)$ $.$	$TS$ $,$ $CT$	$.$
------------------------	---------------	---------------------	-----

```

train_df <- tribble(
  ~ct, ~thickness, ~ts,
  540, 2, 52.5,
  660, 2, 50.2,
  610, 2, 51.3,
  710, 2, 49.1,
  570, 6, 50.8,
  700, 6, 48.7,
  560, 6, 51.2,
  600, 6, 50.8,
  680, 6, 49.3,
  530, 6, 51.5
) %>%
  mutate(thickness = factor(thickness, levels = c(6, 2)))

str(train_df)

## tibble [10 x 3] (S3: tbl_df/tbl/data.frame)
## $ ct      : num [1:10] 540 660 610 710 570 700 560 600 680 530
## $ thickness: Factor w/ 2 levels "6","2": 2 2 2 2 1 1 1 1 1 1
## $ ts      : num [1:10] 52.5 50.2 51.3 49.1 50.8 48.7 51.2 50.8 49.3 51.5

```

factor .

```

lm_fit <- lm(ts ~ ct + thickness, data = train_df)

coef(lm_fit)

## (Intercept)          ct  thickness2
## 61.10796610 -0.01767797  0.80415254

CT   TS           CT      (interaction)   ,
lm_interaction_fit <- lm(
  ts ~ ct + thickness + ct:thickness,
  data = train_df
)

broom::tidy(lm_interaction_fit)

## # A tibble: 4 x 5
##   term       estimate std.error statistic p.value
##   <chr>     <dbl>    <dbl>     <dbl>    <dbl>
## 1 (Intercept) 60.1     0.750     80.2  2.53e-10
## 2 ct        -0.0161    0.00123    -13.1  1.23e- 5
## 3 thickness2  3.28     1.21      2.71  3.52e- 2
## 4 ct:thickness2 -0.00399  0.00194    -2.05 8.57e- 2

CT   TS           .

new_df <- crossing(
  ct = seq(500, 750, by = 10),
  thickness = factor(c(6, 2), levels = c(6, 2))
)

new_df %>%
  mutate(ts_hat = predict(lm_interaction_fit, .)) %>%
  ggplot(aes(x = ct, y = ts_hat)) +
  geom_line(aes(color = thickness)) +
  geom_point(aes(x = ct, y = ts, color = thickness), data = train_df) +
  labs(color = "thickness", x = "CT", y = "TS")

```

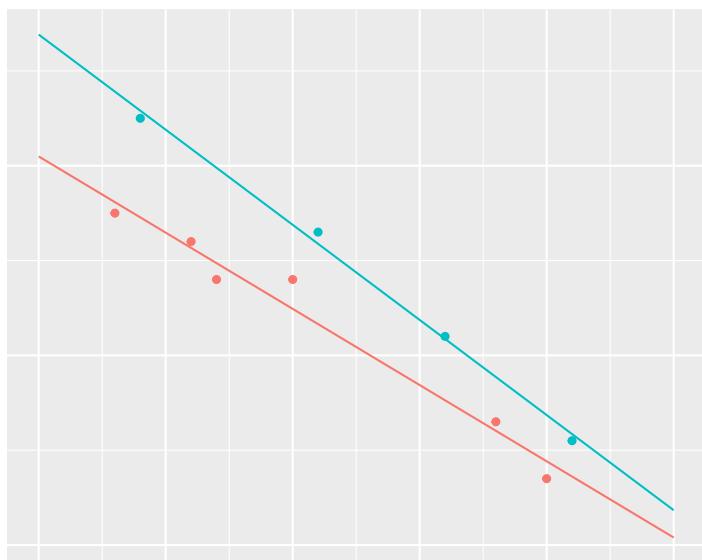


Figure 2.1: CT TS

# Chapter 3

## 3.1 R

package	version
tidyverse	1.3.1
stats	4.1.0
broom	0.7.8
Matrix	1.3-4
nipals	0.7

## 3.2

### 3.2.1 R

Table 3.1 18

```
train_df <- tribble(
  ~company, ~roa, ~roe, ~bis, ~de_ratio, ~turnover,
  "SK ", 2.43, 11.10, 18.46, 441.67, 0.90,
  " ", 3.09, 9.95, 29.46, 239.43, 0.90,
  " ", 2.22, 6.86, 28.62, 249.36, 0.69,
  " ", 5.76, 23.19, 23.47, 326.09, 1.43,
  " ", 1.60, 5.64, 25.64, 289.98, 1.42,
  " ", 3.53, 10.64, 32.25, 210.10, 1.17,
  " ", 4.26, 15.56, 24.40, 309.78, 0.81,
  " ", 3.86, 5.50, 70.74, 41.36, 0.81,
  " ", 4.09, 6.44, 64.38, 55.32, 0.32,
  " ", 2.73, 10.68, 24.41, 309.59, 0.64,
  " ", 2.03, 4.50, 42.53, 135.12, 0.59,
```

```

"  ", 1.96, 8.92, 18.48, 441.19, 1.07,
"  ", 3.25, 7.96, 40.42, 147.41, 1.19,
"  ", 2.01, 10.28, 17.46, 472.78, 1.25,
"  ", 2.28, 3.65, 63.71, 56.96, 0.12,
"  ", 4.51, 7.50, 63.52, 57.44, 0.80,
"  ", 3.29, 12.37, 24.47, 308.63, 0.57,
"  ", 1.73, 7.57, 19.59, 410.45, 1.19
)

knitr::kable(
  train_df, booktabs = TRUE,
  align = rep("r", ncol(train_df)),
  col.names = c(
    " ",
    " ($x_1$)",
    " ($x_2$)",
    " ($x_3$)",
    " ($x_4$)",
    " ($x_5$)"
  ),
  caption = " "
)

```

R stats prcomp .

```

pca_fit <- prcomp(~ roa + roe + bis + de_ratio + turnover,
                   data = train_df, scale = TRUE)

pca_fit
```

```

## Standard deviations (1, ..., p=5):
## [1] 1.6617648 1.2671437 0.7419994 0.2531070 0.1351235
##
## Rotation (n x k) = (5 x 5):
##          PC1         PC2         PC3         PC4         PC5
## roa      0.07608427 -0.77966993  0.0008915975 -0.140755404  0.60540325
## roe     -0.39463007 -0.56541218 -0.2953216494  0.117644166 -0.65078503
## bis      0.56970191 -0.16228156  0.2412221065 -0.637721889 -0.42921686
## de_ratio -0.55982770  0.19654293 -0.2565972887 -0.748094314  0.14992183
## turnover -0.44778451 -0.08636803  0.8881182665 -0.003668418 -0.05711464
summary(pca_fit)
```

```

## Importance of components:
##          PC1    PC2    PC3    PC4    PC5
## Standard deviation 1.6618 1.2671 0.7420 0.25311 0.13512
## Proportion of Variance 0.5523 0.3211 0.1101 0.01281 0.00365
```

Table 3.1:

	$(\$x\_1\$)$	$(\$x\_2\$)$	$(\$x\_3\$)$	$(\$x\_4\$)$	$(\$x\_5\$)$
SK	2.43	11.10	18.46	441.67	0.90
	3.09	9.95	29.46	239.43	0.90
	2.22	6.86	28.62	249.36	0.69
	5.76	23.19	23.47	326.09	1.43
	1.60	5.64	25.64	289.98	1.42
	3.53	10.64	32.25	210.10	1.17
	4.26	15.56	24.40	309.78	0.81
	3.86	5.50	70.74	41.36	0.81
	4.09	6.44	64.38	55.32	0.32
	2.73	10.68	24.41	309.59	0.64
	2.03	4.50	42.53	135.12	0.59
	1.96	8.92	18.48	441.19	1.07
	3.25	7.96	40.42	147.41	1.19
	2.01	10.28	17.46	472.78	1.25
	2.28	3.65	63.71	56.96	0.12
	4.51	7.50	63.52	57.44	0.80
	3.29	12.37	24.47	308.63	0.57
	1.73	7.57	19.59	410.45	1.19

```
## Cumulative Proportion  0.5523 0.8734 0.9835 0.99635 1.00000
```

Figure 3.1

```
screeplot(pca_fit, main = NULL)
```

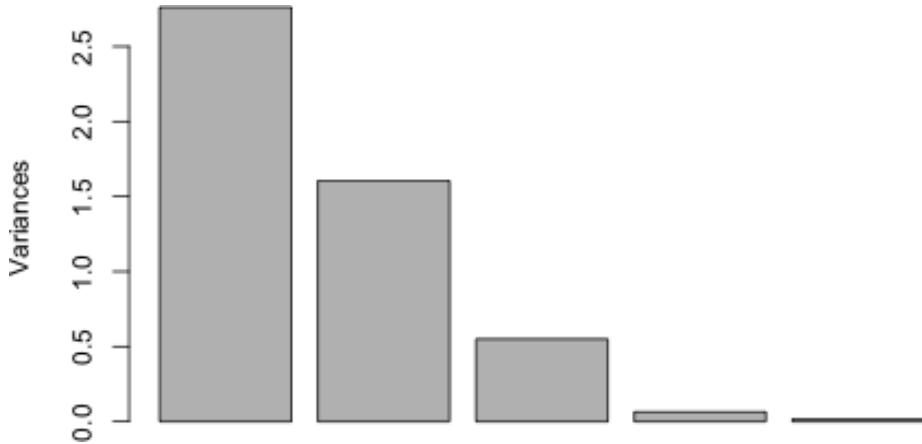


Figure 3.1:

### 3.2.2

$k$        $n$       . ,  $x_{ij}$     $j$        $i$       . ,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

,      (mean-centered)      .  
0      .

$$x_{ij} \leftarrow x_{ij} - \frac{1}{n} \sum_{l=1}^n x_{lj} \quad (3.1)$$

,       $\mathbf{X}$        $x_{ij}$       (3.1)  
0      .      1

$$z_{ij} \leftarrow \frac{x_{ij}}{\sqrt{\frac{1}{n-1} \sum_{l=1}^n x_{lj}^2}}$$

,      (3.2.2)       $s_j$

$$s_j = \sqrt{\frac{1}{n-1} \sum_{l=1}^n x_{lj}^2}$$

$$\mathbf{Z} = z_{ij} \quad .$$

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1k} \\ z_{21} & z_{22} & \cdots & z_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nk} \end{bmatrix}$$

$$\mathbf{x}_j = [x_{1j} \ x_{2j} \ \cdots \ x_{nj}]^\top \quad .$$

$$SS(\mathbf{x}_j) = \mathbf{x}_j^\top \mathbf{x}_j = \sum_{i=1}^n x_{ij}^2 \quad (3.2)$$

,

$$SS(\mathbf{x}_j) = (n-1)s_j^2$$

$$\mathbf{X} = (\mathbf{x}_j) \quad .$$

$$SS(\mathbf{X}) = \sum_{j=1}^k SS(\mathbf{x}_j) = \sum_{j=1}^k \sum_{i=1}^n x_{ij}^2 \quad (3.3)$$

Table 3.1

```
train_df %>%
  mutate_if(is.numeric, function(x) x - mean(x)) %>% # mean-centering
  summarize_if(is.numeric, function(x) sum(x ^ 2)) # sum of squares by variable

## # A tibble: 1 x 5
##       roa     roe    bis de_ratio turnover
##   <dbl> <dbl> <dbl>    <dbl>     <dbl>
## 1  21.9  355.  5591.  347817.     2.23

train_df %>%
  mutate_if(is.numeric, function(x) x - mean(x)) %>% # mean-centering
  summarize_if(is.numeric, function(x) sum(x ^ 2)) %>% # sum of squares by variable
  {sum(.)}

## [1] 353786.6
```

```

(de_ratio) , (turnover) ,
( ) . .
standardized_df <- train_df %>%
  mutate_if(is.numeric, function(x) x - mean(x)) %>% # mean-centering
  mutate_if(is.numeric, function(x) x / sd(x)) # scaling

standardized_df

## # A tibble: 18 x 6
##   company      roa     roe     bis de_ratio turnover
##   <chr>       <dbl>   <dbl>   <dbl>    <dbl>    <dbl>
## 1 SK        -0.533   0.383  -0.918   1.34     0.0507
## 2           0.0484  0.131  -0.312  -0.0749    0.0507
## 3        -0.718  -0.545  -0.358  -0.00551   -0.530
## 4         2.40   3.03   -0.642   0.531     1.52
## 5        -1.26  -0.812  -0.522   0.278     1.49
## 6         0.436   0.282  -0.158  -0.280     0.797
## 7        1.08   1.36   -0.591   0.417    -0.198
## 8         0.726  -0.843   1.96   -1.46    -0.198
## 9         0.929  -0.637   1.61   -1.36    -1.55
## 10        -0.269   0.291  -0.590   0.416    -0.668
## 11        -0.885  -1.06   0.409  -0.804    -0.806
## 12        -0.947  -0.0942 -0.917   1.34     0.521
## 13        0.189  -0.304   0.293  -0.718     0.852
## 14       -0.903   0.203  -0.973   1.56     1.02
## 15        -0.665  -1.25   1.58   -1.35    -2.11
## 16         1.30  -0.405   1.57   -1.35    -0.226
## 17        0.225   0.661  -0.587   0.409    -0.861
## 18        -1.15  -0.390  -0.856   1.12     0.852

n - 1 , 1 .
standardized_df %>%
  summarize_if(is.numeric, function(x) sum(x ^ 2)) # sum of squares by variable

## # A tibble: 1 x 5
##   roa     roe     bis de_ratio turnover
##   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>
## 1    17     17     17      17      17

,
total_ss <- standardized_df %>%
  summarize_if(is.numeric, function(x) sum(x ^ 2)) %>% # sum of squares by variable
  {sum(.)}

total_ss

```

```
## [1] 85
```

### 3.2.3

$$\mathbf{t}_1, \dots, \mathbf{t}_A \quad . \quad p_{aj} \quad . \quad \mathbf{x}_j \quad . \quad k \quad A \quad \mathbf{X}$$

$$\mathbf{t}_a = \sum_{j=1}^k p_{aj} \mathbf{x}_j, \quad a = 1, \dots, A \quad (3.4)$$

$$A \quad . \quad p_{aj} \quad . \quad \mathbf{x}_j \quad . \quad \mathbf{t}_1 \quad \mathbf{X} \quad . \quad , \quad \mathbf{t}_2 \quad \mathbf{t}_1$$

Table 3.1

$$t_1 = 0.07608427 \times roa - 0.39463007 \times roe + 0.56970191 \times bis - 0.55982770 \times de\_ratio - 0.44778451 \times turnover$$

```
new_df <- standardized_df %>%
  mutate(t_1 = 0.07608427 * roa - 0.39463007 * roe
    + 0.56970191 * bis - 0.55982770 * de_ratio
    - 0.44778451 * turnover) %>%
  select(company, t_1) # new variable

print(new_df)

## # A tibble: 18 x 2
##   company      t_1
##   <chr>       <dbl>
## 1 SK        -1.49
## 2          -0.206
## 3           0.197
## 4           -2.35
## 5           -0.895
## 6           -0.368
## 7           -0.935
## 8            2.41
## 9            2.70
## 10           -0.405
## 11            1.40
## 12           -1.54
## 13            0.322
## 14           -2.03
## 15            3.04
```

```

## 16      2.01
## 17     -0.421
## 18     -1.43

,   t1   Z   55% .
t1_ss <- new_df %>%
  summarize_if(is.numeric, function(x) sum(x ^ 2))

t1_ss / total_ss

##          t_1
## 1 0.5522924

t1   Z   ,   Z
Z( X) ,

```

### 3.2.4 (Singular Value Decomposition)

$$\mathbf{Z} = \mathbf{U} \mathbf{D} \mathbf{V}^\top$$

$\mathbf{Z}$    .   .  
 $(n \times k)$     $\mathbf{Z}$    .

$$\mathbf{Z} = \mathbf{U} \mathbf{D} \mathbf{V}^\top \quad (3.5)$$

$$, r = \min\{n, k\} ,$$

- $\mathbf{U}$ :  $(n \times r)$  (orthogonal)
- $\mathbf{D}$ :  $(r \times r)$  (diagonal) . rank ,  $\mathbf{Z}$  (singular value)
- $\mathbf{V}$ :  $(k \times r)$  (orthogonal)

, R svd   .  
 $\mathbf{Z} \leftarrow \text{as.matrix}(\text{standardized_df[, -1]})$   
 $\text{svd}_Z \leftarrow \text{svd}(Z)$   
 $Z_{\text{rec}} \leftarrow \text{svd}_Z\$u \%*\% \text{diag}(\text{svd}_Z\$d) \%*\% \text{t}(\text{svd}_Z\$v)$   
 $\text{all}(\text{near}(Z, Z_{\text{rec}}))$

```

## [1] TRUE
,   V   .   ,   V   t1   .
svd_Z$v[, 1]

## [1] 0.07608427 -0.39463007  0.56970191 -0.55982770 -0.44778451

svd_Z$d

```

```
## [1] 6.8516318 5.2245674 3.0593417 1.0435870 0.5571285
```

### 3.2.5 (Spectral Decomposition)

**A**

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

$\mathbf{v} \neq \mathbf{0}$ ,  $\lambda$  (eigenvalue),  $\mathbf{v}$  (eigenvector)

$\mathbf{v}^\top \mathbf{v} = 1$ ,  $(r \times r)$ ,  $\mathbf{A}$   $r$   $\lambda_1, \dots, \lambda_r$ ,  $\mathbf{v}_1, \dots, \mathbf{v}_r$ ,

**A**

$$\mathbf{A}[\mathbf{v}_1 \dots \mathbf{v}_r] = [\mathbf{v}_1 \dots \mathbf{v}_r] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_r \end{bmatrix} \mathbf{A} = \mathbf{A}[\mathbf{v}_1 \dots \mathbf{v}_r][\mathbf{v}_1 \dots \mathbf{v}_r]^{-1} = [\mathbf{v}_1 \dots \mathbf{v}_r] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_r \end{bmatrix} [\mathbf{v}_1 \dots \mathbf{v}_r]$$

$\mathbf{A}$  (symmetric),  $(\mathbf{V}\mathbf{V}^\top = \mathbf{I})$ ,

$$\mathbf{A} = \mathbf{V}\mathbf{V}^\top$$

$\mathbf{Z}^\top \mathbf{Z}$  (3.5),

$$\mathbf{Z}^\top \mathbf{Z} = \mathbf{V}\mathbf{D}^\top \mathbf{U}^\top \mathbf{U}\mathbf{D}\mathbf{V}^\top = \mathbf{V}\mathbf{D}^2\mathbf{V}^\top = \mathbf{V}\mathbf{V}^\top$$

,  $\mathbf{V}$ ,

R eigen

```
eig_Z <- eigen(t(Z) %*% Z, symmetric = TRUE)
eig_Z
```

```
## eigen() decomposition
## $values
## [1] 46.9448582 27.2961041  9.3595718  1.0890737  0.3103922
##
## $vectors
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -0.07608427 -0.77966993  0.0008915975 -0.140755404  0.60540325
## [2,]  0.39463007 -0.56541218 -0.2953216494  0.117644166 -0.65078503
## [3,] -0.56970191 -0.16228156  0.2412221065 -0.637721889 -0.42921686
```

```

## [4,]  0.55982770  0.19654293 -0.2565972887 -0.748094314  0.14992183
## [5,]  0.44778451 -0.08636803  0.8881182665 -0.003668418 -0.05711464
      values   ZTZ   (eigenvalue) .
all(near(eig_Z$values, svd_Z$d ^ 2))

## [1] TRUE

      vectors   ZTZ   (eigenvector) .
all(near(eig_Z$vectors, svd_Z$v))

## [1] FALSE

,
.

sign_adjust <- 1 - 2 * ((eig_Z$vectors * svd_Z$v) < 0)
all(near(eig_Z$vectors * sign_adjust, svd_Z$v))

## [1] TRUE

,
.

eig_Z$values / sum(eig_Z$values)

## [1] 0.552292449 0.321130636 0.110112610 0.012812632 0.003651673
Z
      ,
.

      
$$\frac{1}{n-1} \mathbf{Z}^T \mathbf{Z}$$

all(near(cov(Z), t(Z) %*% Z / (nrow(Z) - 1)))

## [1] TRUE

,
.

      
$$\frac{1}{n-1} \mathbf{Z}^T \mathbf{Z} = \frac{1}{n-1} \mathbf{V} \mathbf{V}^T = \mathbf{V} \left( \frac{1}{n-1} \right) \mathbf{V}^T$$

, Z - , V ZTZ , ZTZ (n - 1)

eig_cov_Z <- eigen(cov(Z))
eig_cov_Z

## eigen() decomposition
## $values
## [1] 2.76146225 1.60565318 0.55056305 0.06406316 0.01825836

```

```

## 
## $vectors
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -0.07608427 0.77966993 0.0008915975 -0.140755404 0.60540325
## [2,] 0.39463007 0.56541218 -0.2953216494 0.117644166 -0.65078503
## [3,] -0.56970191 0.16228156 0.2412221065 -0.637721889 -0.42921686
## [4,] 0.55982770 -0.19654293 -0.2565972887 -0.748094314 0.14992183
## [5,] 0.44778451 0.08636803 0.8881182665 -0.003668418 -0.05711464
all(near(eig_cov_Z$values, eig_Z$values / (nrow(Z) - 1)))

## [1] TRUE

(correlation matrix)

eig_cor_raw <- eigen(cor(train_df[, -1]))
eig_cor_raw

## eigen() decomposition
## $values
## [1] 2.76146225 1.60565318 0.55056305 0.06406316 0.01825836
##
## $vectors
## [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -0.07608427 0.77966993 0.0008915975 -0.140755404 0.60540325
## [2,] 0.39463007 0.56541218 -0.2953216494 0.117644166 -0.65078503
## [3,] -0.56970191 0.16228156 0.2412221065 -0.637721889 -0.42921686
## [4,] 0.55982770 -0.19654293 -0.2565972887 -0.748094314 0.14992183
## [5,] 0.44778451 0.08636803 0.8881182665 -0.003668418 -0.05711464
all(near(eig_cov_Z$values, eig_cor_raw$values))

## [1] TRUE

all(near(eig_cov_Z$vectors, eig_cor_raw$vectors))

## [1] TRUE

```

### 3.2.6 NIPALS

NIPALS(Nonlinear Iterative Parital Least Squares) (iterative)

,

,

$Z$

$T$

$V$

.

(

$Z$

$X$

)

$$Z = UDV^\top = TV^\top$$

,

$T$

.

$$\mathbf{T} = \mathbf{Z}\mathbf{V}$$

```
T_mat <- Z %*% svd_Z$v
T_mat

##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -1.4870243  0.6066594 -0.63361774 -0.29625002  0.020293731
## [2,] -0.2063797 -0.0804627 -0.04965017  0.26323513  0.063581473
## [3,]  0.1968538  0.9704605 -0.39507856  0.27123746  0.103351746
## [4,] -2.3542884 -3.5056480  0.16252734  0.02524924 -0.249920974
## [5,] -0.8953707  1.4552899  1.36265905  0.20161775 -0.055517167
## [6,] -0.3682082 -0.5976313  0.65857722  0.27901317  0.060458248
## [7,] -0.9354306 -1.4144519 -0.82574638  0.07358977  0.095960908
## [8,]  2.4129728 -0.6785064  0.92207607 -0.36161577 -0.062593521
## [9,]  2.6991862 -0.7596591 -0.45091077 -0.21030378  0.168645128
## [10,] -0.4050098  0.2800099 -0.92835441  0.13993488  0.001811118
## [11,]  1.3958199  1.1353513 -0.09819177  0.34335126 -0.094986796
## [12,] -1.5381192  1.1576616 -0.07467334 -0.29404424  0.052430946
## [13,]  0.3217681 -0.2378023  1.10180230  0.28507243  0.030666763
## [14,] -2.0306806  0.9646122  0.20906175 -0.39639758 -0.085778570
## [15,]  3.0389460  0.8841645 -0.77478769 -0.04079854 -0.349688462
## [16,]  2.0064063 -1.2831337  0.64388897 -0.22077705  0.188366871
## [17,] -0.4211779 -0.2987099 -1.20644766  0.11766274  0.068250991
## [18,] -1.4302634  1.4017959  0.37686579 -0.17977686  0.044667568

NIPALS           T           V
• [ 0]           .  $\mathbf{h} \leftarrow 1, \mathbf{Z}_h \leftarrow \mathbf{Z}$ .
• [ 1]            $\mathbf{Z}_h$             $\mathbf{t}_h$            .
• [ 2]           .  $\mathbf{v}_h \leftarrow \mathbf{Z}_h \mathbf{t}_h / \sqrt{\mathbf{t}_h^\top \mathbf{t}_h}$ 
• [ 3]           1           .  $\mathbf{v}_h \leftarrow \mathbf{v}_h / \sqrt{\mathbf{v}_h^\top \mathbf{v}_h}$ 
• [ 4]           .  $\mathbf{t}_h \leftarrow \mathbf{Z}_h \mathbf{v}_h$ 
• [ 5]            $\mathbf{t}_h$            [ 6]           ,           [ 2]           .
• [ 6]            $\mathbf{Z}_h$             $\mathbf{t}_h$             $\mathbf{v}_h$            .            $\mathbf{Z}_{h+1}$            .
                                          $\mathbf{Z}_{h+1} \leftarrow \mathbf{Z}_h - \mathbf{t}_h \mathbf{v}_h^\top$ 
• [ 7]  $\mathbf{h} \leftarrow h + 1$      , [ 1]     . [ 1] - [ 7]      $\mathbf{Z}$  rank
                                         .           X           ,           X           Z           .           r
                                         .           .           .           .           .           .           .

nipals_pca <- function(X, r = NULL) {
  if (is_empty(r) || (r > min(dim(X)))) {
    r <- min(dim(X))
  }
}
```

```

Th <- matrix(NA, nrow = nrow(X), ncol = r)
Vh <- matrix(NA, nrow = ncol(X), ncol = r)

for (h in seq_len(r)) {
  # 1
  j <- sample(ncol(X), 1)
  Th[, h] <- X[, j]

  while (TRUE) {
    # 2
    Vh[, h] <- t(t(Th[, h]) %*% X / (norm(Th[, h], "2") ^ 2))

    # 3
    Vh[, h] <- Vh[, h] / norm(Vh[, h], "2")

    # 4
    th <- X %*% Vh[, h]

    # 5
    if (all(near(Th[, h], th))) break
    Th[, h] <- th
  }

  # 6
  X <- X - Th[, h] %*% t(Vh[, h])
}

return(list(T = Th, V = Vh))
}

nipals_Z <- nipals_pca(Z)
nipals_Z

## $T
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -1.4870243 -0.6066594  0.63361775  0.29625002 -0.020293733
## [2,] -0.2063797  0.0804627  0.04965017 -0.26323513 -0.063581471
## [3,]  0.1968538 -0.9704605  0.39507856 -0.27123747 -0.103351743
## [4,] -2.3542884  3.5056480 -0.16252734 -0.02524923  0.249920974
## [5,] -0.8953708 -1.4552900 -1.36265905 -0.20161775  0.055517169
## [6,] -0.3682082  0.5976313 -0.65857723 -0.27901317 -0.060458246
## [7,] -0.9354306  1.4144520  0.82574637 -0.07358977 -0.095960907
## [8,]  2.4129728  0.6785064 -0.92207607  0.36161577  0.062593518
## [9,]  2.6991862  0.7596591  0.45091077  0.21030377 -0.168645129
## [10,] -0.4050098 -0.2800099  0.92835441 -0.13993489 -0.001811117

```

```

## [11,]  1.3958199 -1.1353513  0.09819177 -0.34335126  0.094986799
## [12,] -1.5381192 -1.1576616  0.07467335  0.29404424 -0.052430948
## [13,]  0.3217681  0.2378023 -1.10180230 -0.28507243 -0.030666760
## [14,] -2.0306806 -0.9646122 -0.20906175  0.39639758  0.085778567
## [15,]  3.0389460 -0.8841645  0.77478770  0.04079855  0.349688462
## [16,]  2.0064063  1.2831337 -0.64388897  0.22077705 -0.188366873
## [17,] -0.4211779  0.2987099  1.20644766 -0.11766275 -0.068250990
## [18,] -1.4302634 -1.4017959 -0.37686579  0.17977686 -0.044667569
##
## $V
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  0.07608428  0.77966993 -0.0008916024  0.140755398 -0.60540326
## [2,] -0.39463007  0.56541218  0.2953216455 -0.117644164  0.65078503
## [3,]  0.56970191  0.16228156 -0.2412221067  0.637721895  0.42921684
## [4,] -0.55982770 -0.19654293  0.2565972909  0.748094309 -0.14992185
## [5,] -0.44778451  0.08636803 -0.8881182671  0.003668428  0.05711464

Z
.
all(near(Z, nipals_Z$T %*% t(nipals_Z$V)))

## [1] TRUE

R   nipals    nipals      NIPALS
library(nipals)
nipals(Z, center = FALSE, scale = FALSE)

## $eig
## [1] 6.8516317 5.2245674 3.0593417 1.0435869 0.5571285
##
## $scores
##          PC1       PC2       PC3       PC4       PC5
## [1,] -0.21705404 -0.11608067 -0.20710543  0.28388475 -0.036368396
## [2,] -0.03011834  0.01540551 -0.01623006 -0.25221776 -0.114174269
## [3,]  0.02869590 -0.18575892 -0.12913406 -0.25987079 -0.185560165
## [4,] -0.34348333  0.67105909  0.05310696 -0.02428657  0.448582844
## [5,] -0.13073246 -0.27851123  0.44541637 -0.19321728  0.099609669
## [6,] -0.05371865  0.11440401  0.21526389 -0.26733867 -0.108571464
## [7,] -0.13647563  0.27074924 -0.26991736 -0.07048155 -0.172255999
## [8,]  0.35219939  0.12981093  0.30139420  0.34648877  0.112419837
## [9,]  0.39397535  0.14532356 -0.14739177  0.20158102 -0.302663550
## [10,] -0.05912155 -0.05359220 -0.30344792 -0.13408884 -0.003277662
## [11,]  0.20367981 -0.21735017 -0.03209051 -0.32904441  0.170427304
## [12,] -0.22453126 -0.22153804 -0.02440174  0.28178254 -0.094052577
## [13,]  0.04697085  0.04551641  0.36014173 -0.27315578 -0.055099430
## [14,] -0.29641393 -0.18457147  0.06834143  0.37981073  0.154041866
## [15,]  0.44350417 -0.16932257 -0.25324815  0.03896920  0.627670066

```

```

## [16,]  0.29288258  0.24554714  0.21046020  0.21162296 -0.338060584
## [17,] -0.06146040  0.05717479 -0.39435062 -0.11272299 -0.122527440
## [18,] -0.20879845 -0.26826541  0.12319285  0.17228468 -0.080140048
##
## $loadings
##          PC1       PC2       PC3       PC4       PC5
## roa      0.07627711  0.7796534  0.0008551484  0.140974596 -0.60534928
## roe     -0.39449021  0.5654941 -0.2953469599 -0.117893972  0.65074198
## bis      0.56974203  0.1621586  0.2412197864  0.637556663  0.42945678
## de_ratio -0.55987629 -0.1964075 -0.2565837179  0.748154680 -0.14963952
## turnover -0.44776314  0.0865197  0.8881144365  0.003640124  0.05711403
##
## $fitted
## NULL
##
## $ncomp
## [1] 5
##
## $R2
## [1] 0.552292435 0.321130644 0.110112610 0.012812631 0.003651673
##
## $iter
## [1] 14 4 4 6 3
##
## $center
## [1] NA
##
## $scale
## [1] NA
,
            center( )  scale( )    TRUE
.

library(nipals)
nipals(train_df[, -1], center = TRUE, scale = TRUE)

##
## $eig
## [1] 6.8516317 5.2245674 3.0593417 1.0435869 0.5571285
##
## $scores
##          PC1       PC2       PC3       PC4       PC5
## [1,] -0.21705404 -0.11608067 -0.20710543  0.28388475 -0.036368396
## [2,] -0.03011834  0.01540551 -0.01623006 -0.25221776 -0.114174269
## [3,]  0.02869590 -0.18575892 -0.12913406 -0.25987079 -0.185560165
## [4,] -0.34348333  0.67105909  0.05310696 -0.02428657  0.448582844
## [5,] -0.13073246 -0.27851123  0.44541637 -0.19321728  0.099609669
## [6,] -0.05371865  0.11440401  0.21526389 -0.26733867 -0.108571464
## [7,] -0.13647563  0.27074924 -0.26991736 -0.07048155 -0.172255999

```

```

## [8,] 0.35219939 0.12981093 0.30139420 0.34648877 0.112419837
## [9,] 0.39397535 0.14532356 -0.14739177 0.20158102 -0.302663550
## [10,] -0.05912155 -0.05359220 -0.30344792 -0.13408884 -0.003277662
## [11,] 0.20367981 -0.21735017 -0.03209051 -0.32904441 0.170427304
## [12,] -0.22453126 -0.22153804 -0.02440174 0.28178254 -0.094052577
## [13,] 0.04697085 0.04551641 0.36014173 -0.27315578 -0.055099430
## [14,] -0.29641393 -0.18457147 0.06834143 0.37981073 0.154041866
## [15,] 0.44350417 -0.16932257 -0.25324815 0.03896920 0.627670066
## [16,] 0.29288258 0.24554714 0.21046020 0.21162296 -0.338060584
## [17,] -0.06146040 0.05717479 -0.39435062 -0.11272299 -0.122527440
## [18,] -0.20879845 -0.26826541 0.12319285 0.17228468 -0.080140048
##
## $loadings
##          PC1      PC2      PC3      PC4      PC5
## roa      0.07627711 0.7796534 0.0008551484 0.140974596 -0.60534928
## roe     -0.39449021 0.5654941 -0.2953469599 -0.117893972 0.65074198
## bis      0.56974203 0.1621586 0.2412197864 0.637556663 0.42945678
## de_ratio -0.55987629 -0.1964075 -0.2565837179 0.748154680 -0.14963952
## turnover -0.44776314 0.0865197 0.8881144365 0.003640124 0.05711403
##
## $fitted
## NULL
##
## $ncomp
## [1] 5
##
## $R2
## [1] 0.552292435 0.321130644 0.110112610 0.012812631 0.003651673
##
## $iter
## [1] 14 4 4 6 3
##
## $center
##      roa      roe      bis      de_ratio      turnover
## 3.0350000 9.3505556 35.1116667 250.1477778 0.8816667
##
## $scale
##      roa      roe      bis      de_ratio      turnover
## 1.1356949 4.5692430 18.1343662 143.0378269 0.3618132

```

### 3.3

$$2.2 \quad (2.2) \quad .$$

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \quad (3.6)$$

,  $\beta$  ,  $\epsilon$  ,  $\mathbf{X}$  ,  $\mathbf{y}$  .  $\mathbf{X}$   
 (multicollinearity)  $\beta$  ,  $\mathbf{X}$  ,  $\mathbf{y}$  .  $\mathbf{X}$   
 (principal component regression; PCR)  $\mathbf{X}$   $\beta$  .  $\mathbf{X}$   
 $A$  ( $A \leq \text{rank}(\mathbf{X})$ )

$$\mathbf{y} = q_1 \mathbf{t}_1 + q_2 \mathbf{t}_2 + \cdots + q_A \mathbf{t}_A + \mathbf{f} \quad (3.7)$$

$$\mathbf{f} \quad , q_1, \dots, q_A \quad , A \quad (n \times A) \quad \mathbf{T}_A = [\mathbf{t}_1 \dots \mathbf{t}_A] , \\ \mathbf{q} = [q_1 \dots q_A]^T \quad , \quad (3.7) \quad .$$

$$\mathbf{y} = \mathbf{T}_A \mathbf{q}_A + \mathbf{f} \quad (3.8)$$

$\mathbf{t}_1, \dots, \mathbf{t}_A$  (linearly  
 independent) , .

### 3.3.1 R

3 1 (y) .

```
train_df <- tribble(
  ~x1, ~x2, ~x3, ~y,
  -3, -3, 5, -30,
  -2, -3, 7, -20,
  0, 0, 4, 0,
  1, 2, 0, 5,
  2, 2, -5, 10,
  2, 2, -11, 35
)

knitr::kable(
  train_df, booktabs = TRUE,
  align = rep("r", ncol(train_df)),
  caption = ""
)
```

3 2 .

```
pcr_fit <- pls::pcr(y ~ x1 + x2 + x3, data = train_df, ncomp = 2)
coef(pcr_fit, intercept = TRUE)
```

```
## , , 2 comps
```

```
##
```

```
## y
```

Table 3.2:

x1	x2	x3	y
-3	-3	5	-30
-2	-3	7	-20
0	0	4	0
1	2	0	5
2	2	-5	10
2	2	-11	35

```

## (Intercept) 0.000000
## x1          2.130440
## x2          2.721789
## x3         -1.737825
(      ) .
summary(pcr_fit)

## Data:   X dimension: 6 3
## Y dimension: 6 1
## Fit method: svdpc
## Number of components considered: 2
## TRAINING: % variance explained
##    1 comps  2 comps
## X     94.98    99.79
## y     87.94    91.31
87.9415591% 91.3101613%
.
.
```

### 3.3.2

Table 3.2

```

pca_fit <- prcomp(train_df[, c("x1", "x2", "x3")], rank. = 2,
                     center = TRUE, scale. = FALSE)
pca_fit$x

##           PC1       PC2
## [1,] -6.2346992  1.9880169
## [2,] -7.8320036  0.6817026
## [3,] -3.6996775 -1.5151642
## [4,]  0.8208672 -2.0392493
## [5,]  5.6979984 -0.6940262
## [6,] 11.2475146  1.5787202
.
```

```

y_centered <- train_df$y - mean(train_df$y)
y_centered

## [1] -30 -20   0   5  10  35
           , intercept
pc_lm_fit <- lm(y_centered ~ - 1 + pca_fit$x)
coef(pc_lm_fit)

## pca_fit$xPC1 pca_fit$xPC2
##      2.918798    -2.539206
(3.8)      q_A      (A = 2).

```

### 3.3.3

```

pca_fit$rotation

##          PC1        PC2
## x1  0.2525343 -0.5487321
## x2  0.2841664 -0.7452586
## x3 -0.9249194 -0.3787911
,
(     )
beta_x <- pca_fit$rotation %*% coef(pc_lm_fit)
beta_x

##      [,1]
## x1  2.130440
## x2  2.721789
## x3 -1.737825

Intercept
mean(train_df$y) - colMeans(train_df[, c("x1", "x2", "x3")]) %*% beta_x

##      [,1]
## [1,]    0

```

Table 3.2              Intercept 0



# Chapter 4

squares: PLS       $k$       .      .      .      .      .      .  
(partial least  
(PCR)      ,      .  
(PLS)      ( )      ,      .      .      .      .  
,

## 4.1 R

package	version
tidyverse	1.3.1
pls	2.7-3

## 4.2

### 4.2.1 R

```
train_df <- tribble(  
  ~x1, ~x2, ~x3, ~y,  
  -3, -3, 5, -30,  
  -2, -3, 7, -20,  
  0, 0, 4, 0,  
  1, 2, 0, 5,  
  2, 2, -5, 10,  
  2, 2, -11, 35  
)
```

Table 4.1:

x1	x2	x3	y
-3	-3	5	-30
-2	-3	7	-20
0	0	4	0
1	2	0	5
2	2	-5	10
2	2	-11	35

```

knitr::kable(
  train_df, booktabs = TRUE,
  align = rep("r", ncol(train_df)),
  caption = ""
)

R   pls     plsr()    PLS
plsr_fit <- pls::plsr(y ~ x1 + x2 + x3, data = train_df, ncomp = 2)
coef(plsr_fit)

## , , 2 comps
##
##          y
## x1  2.475395
## x2  2.523238
## x3 -1.704636

object summary()      X      y
summary(plsr_fit)

## Data:   X dimension: 6 3
## Y dimension: 6 1
## Fit method: kernelpls
## Number of components considered: 2
## TRAINING: % variance explained
##    1 comps  2 comps
## X    94.97    99.78
## y    88.28    91.46
88.2814529% 91.4578959%

```

### 4.2.2 PLS

$$\begin{array}{ccccc} \mathbf{X} & & \mathbf{y} & & \\ \mathbf{t}_1, \dots, \mathbf{t}_A & & . & (n \times k) & \mathbf{X} & & \mathbf{y} & & A \end{array}$$

$$\mathbf{X} = \mathbf{t}_1 \mathbf{p}_1^\top + \mathbf{t}_2 \mathbf{p}_2^\top + \dots + \mathbf{t}_A \mathbf{p}_A^\top + \mathbf{E} \quad (4.1)$$

$$\mathbf{y} = \mathbf{t}_1 b_1 + \mathbf{t}_2 b_2 + \dots + \mathbf{t}_A b_A + \mathbf{f} \quad (4.2)$$

$$\begin{array}{ccccccc} \mathbf{p}_a & \mathbf{X} & & (\text{loading}), & b_a & \mathbf{y} & , \mathbf{E} \mathbf{f} & ( ) . \\ (n \times A) & \mathbf{T} = [\mathbf{t}_1 \dots \mathbf{t}_A] & (k \times A) & & \mathbf{P} = [\mathbf{p}_1 \dots \mathbf{p}_A], & & \mathbf{b} = [b_1 \dots b_A]^\top \end{array}$$

$$\mathbf{X} = \mathbf{T} \mathbf{P}^\top + \mathbf{E} \quad (4.3)$$

$$\mathbf{y} = \mathbf{T} \mathbf{b} + \mathbf{f} \quad (4.4)$$

### 4.2.3 NIPALS

- [ 0]  $\mathbf{X}_a$ ,  $a \leftarrow 1$ ,  $\mathbf{X}_a \leftarrow \mathbf{X}$ ,  $\mathbf{y}_a \leftarrow \mathbf{y}$ ,  $\mathbf{w}_a = [w_{a1} \dots w_{ak}]^\top$ .
- [ 1]  $\mathbf{w}_a \leftarrow \mathbf{X}_a^\top \mathbf{y}_a / \mathbf{y}_a^\top \mathbf{y}_a$
- [ 2]  $\mathbf{w}_a \leftarrow \mathbf{w}_a / \sqrt{\mathbf{w}_a^\top \mathbf{w}_a}$
- [ 3]  $\mathbf{t}_a \leftarrow \mathbf{X}_a \mathbf{w}_a$ ,  $\mathbf{w}_a \leftarrow \mathbf{t}_a^\top \mathbf{X}_a$
- [ 4] (4.1)  $\mathbf{X}_a$ ,  $\mathbf{t}_a$ ,  $\mathbf{p}_a$ ,  $\mathbf{p}_a \leftarrow \mathbf{X}_a^\top \mathbf{t}_a / \mathbf{t}_a^\top \mathbf{t}_a$
- [ 5]  $\mathbf{p}_a \leftarrow \mathbf{p}_a / \sqrt{\mathbf{p}_a^\top \mathbf{p}_a}$ ,  $\mathbf{t}_a \leftarrow \mathbf{t}_a d$ ,  $\mathbf{w}_a \leftarrow \mathbf{w}_a d$ ,  $\mathbf{p}_a \leftarrow \frac{1}{d} \mathbf{p}_a$
- [ 6] (4.2)  $\mathbf{t}_a \leftarrow \mathbf{y}_a / b_a$ ,  $b_a \leftarrow \mathbf{y}_a^\top \mathbf{t}_a / \mathbf{t}_a^\top \mathbf{t}_a$
- [ 7]  $\mathbf{X}_a$ ,  $\mathbf{y}_a$ ,  $\mathbf{t}_a$ ,  $\mathbf{X}_{a+1} \leftarrow \mathbf{X}_a - \mathbf{t}_a \mathbf{p}_a^\top$ ,  $\mathbf{y}_{a+1} \leftarrow \mathbf{y}_a - \mathbf{t}_a b_a$ ,  $\mathbf{y}_{a+1}$

```

• [ 8]  $a \leftarrow a + 1$       , [ 1]    . [ 1] - [ 8]     A      .
NIPALS      nipals_plsr      . ,      .
• X:      ( $n \times k$ )
• y:
• A:

nipals_plsr <- function(X, y, A = NULL) {
  if (is_empty(A) || (A > min(dim(X)))) {
    A <- min(dim(X))
  }

  Tmat <- matrix(NA, nrow = nrow(X), ncol = A)
  Wmat <- matrix(NA, nrow = ncol(X), ncol = A)
  Pmat <- matrix(NA, nrow = ncol(X), ncol = A)
  b <- vector("numeric", length = A)

  for (a in seq_len(A)) {
    # 1
    Wmat[, a] <- coef(lm(X ~ -1 + y))

    # 2
    Wmat[, a] <- Wmat[, a] / sqrt(sum(Wmat[, a]^2))

    # 3
    Tmat[, a] <- X %*% Wmat[, a]

    # 4
    Pmat[, a] <- coef(lm(X ~ -1 + Tmat[, a]))

    # 5
    p_size <- sqrt(sum(Pmat[, a]^2))
    Tmat[, a] <- Tmat[, a] * p_size
    Wmat[, a] <- Wmat[, a] * p_size
    Pmat[, a] <- Pmat[, a] / p_size

    # 6
    b[a] <- coef(lm(y ~ -1 + Tmat[, a]))

    # 7
    X <- X - Tmat[, a] %*% t(Pmat[, a])
    y <- y - Tmat[, a] %*% t(b[a])
  }

  return(list(T = Tmat, W = Wmat, P = Pmat, b = b))
}

```

```
X <- as.matrix(train_df[, c("x1", "x2", "x3")])
y <- train_df$y
nipals_fit <- nipals_plsr(X, y, A = 2)
nipals_fit
```

```
## $T
##          [,1]      [,2]
## [1,] -6.3243200 -2.0380766
## [2,] -7.8584372 -0.5965965
## [3,] -3.6317877  1.5429616
## [4,]  0.9079469  1.9745198
## [5,]  5.7294582  0.7158171
## [6,] 11.1771398 -1.5986253
##
## $W
##          [,1]      [,2]
## [1,]  0.2817766  0.6579688
## [2,]  0.3130851  0.6388931
## [3,] -0.9079469  0.4245052
##
## $P
##          [,1]      [,2]
## [1,]  0.2537679  0.5424154
## [2,]  0.2858180  0.7279599
## [3,] -0.9240724  0.4193565
##
## $b
## [1] 2.924570 2.464658
```

$$(4.3) \quad (4.4), \quad \mathbf{T} \quad \mathbf{P} \quad \mathbf{b}$$

$$\hat{\mathbf{P}}^\top = (\mathbf{T}^\top \mathbf{T})^{-1} \mathbf{T}^\top \mathbf{X} \quad (4.5)$$

$$\hat{\mathbf{b}} = (\mathbf{T}^\top \mathbf{T})^{-1} \mathbf{T}^\top \mathbf{y} \quad (4.6)$$

NIPALS

```
P_hat <- t(solve(t(nipals_fit$T) %*% nipals_fit$T) %*% t(nipals_fit$T) %*% X)
all(near(nipals_fit$P, P_hat))

## [1] TRUE

b_hat <- as.vector(t(solve(t(nipals_fit$T) %*% nipals_fit$T) %*%
t(nipals_fit$T) %*% as.matrix(y, ncol = 1)))
all(near(nipals_fit$b, b_hat))
```

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

#### 4.2.4

```
NIPALS      X      y      .
T      X      W,      P      .
```

$$\mathbf{T} = \mathbf{X}\mathbf{W}(\mathbf{P}^\top\mathbf{W})^{-1} \quad (4.7)$$

(4.4) ,

$$\begin{aligned} \mathbf{y} &= \mathbf{X}\mathbf{W}(\mathbf{P}^\top\mathbf{W})^{-1}\mathbf{b} + \mathbf{f} \\ &= \mathbf{X}\beta_{PLS} + \mathbf{f} \end{aligned} \quad (4.8)$$

```
,      X      .

```

$$\beta_{PLS} = \mathbf{W}(\mathbf{P}^\top\mathbf{W})^{-1}\mathbf{b} \quad (4.9)$$

```
beta_pls <- nipals_fit$W %*%
  solve(t(nipals_fit$P) %*% nipals_fit$W) %*%
  as.matrix(nipals_fit$b, ncol = 1L)
beta_pls

##          [,1]
## [1,]  2.475395
## [2,]  2.523238
## [3,] -1.704636
```

#### 4.2.5

```
A      y      ( ) SSR,      SSE      , y      (SST)      .

```

$$SST = SS(\mathbf{y}) = SSR + SSE$$

```
SS()      ,      x      .

```

$$SS(\mathbf{x}) = \mathbf{x}^\top\mathbf{x}$$

```
, SSR      .

```

$$\begin{aligned} SSR &= \sum_{a=1}^A SS(b_a \mathbf{t}_a) \\ &= \sum_{a=1}^A b_a^2 SS(\mathbf{t}_a) \end{aligned} \quad (4.10)$$

$a \quad \mathbf{t}_a \quad \mathbf{y} \quad SSR_a = b_a^2 SS(\mathbf{t}_a) \quad , \quad SSR$

$$SSR = \sum_{a=1}^A SSR_a$$

$2 \quad \mathbf{y} \quad \text{PLS}$

```
SSR_a <- nipals_fit$b ^ 2 * diag(t(nipals_fit$T) %*% nipals_fit$T)
SSR_a
```

```
## [1] 2339.45850 84.17574
```

,  $\mathbf{y}$

$$\Delta R_a^2 = \frac{SSR_a}{SST} \quad (4.11)$$

$\mathbf{y}$

```
SST <- sum(y ^ 2)
delta_Rsq <- SSR_a / SST
delta_Rsq
```

```
## [1] 0.88281453 0.03176443
```

$A \quad \text{PLS} \quad \mathbf{y} \quad (SSR/SST)$

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{a=1}^A SSR_a}{SST} = \sum_{a=1}^A \Delta R_a^2$$

,  $A \quad \text{PLS} \quad \mathbf{y} \quad (SSR/SST)$

$2 \quad \mathbf{y}$

```
sum(delta_Rsq)
```

```
## [1] 0.914579
```

```

4.2.1 R    pls
,      X          .      SS(ta)  X      (SS(X))

X          .
diag(t(nipals_fit$T) %*% nipals_fit$T) / sum(diag(t(X) %*% X))

## [1] 0.94972728 0.04811507
2      PLS   X
sum(diag(t(nipals_fit$T) %*% nipals_fit$T)) / sum(diag(t(X) %*% X))

## [1] 0.9978423
2
,      4.2.1 R    pls
.
```

## 4.2.6

$$(4.7) \quad \mathbf{T} \quad \mathbf{X}$$

$$\begin{aligned} \mathbf{T} &= \mathbf{XW} (\mathbf{P}^\top \mathbf{W})^{-1} \\ &= \mathbf{XW}^* \end{aligned} \quad (4.12)$$

$$, \mathbf{W}^* = [\mathbf{w}_1^* \cdots \mathbf{w}_A^*]$$

$$\mathbf{W}^* = \mathbf{W} (\mathbf{P}^\top \mathbf{W})^{-1} \quad (4.13)$$

, y , j VIP(variable importance  
in projection) .

$$VIP_j = \sqrt{\frac{k}{SSR} \sum_{a=1}^A SSR_a \left( w_{aj}^* / \|\mathbf{w}_a^*\| \right)^2} \quad (4.14)$$

$$\sum_{j=1}^k VIP_j^2 = k$$

$$, \quad 1 \quad . \quad , \quad VIP \quad 1 \quad .$$

```

k <- ncol(X)
Wx <- nipals_fit$W %*% solve(t(nipals_fit$P) %*% nipals_fit$W)
VIP <- sqrt(
  colSums(
    k / sum(SSR_a) * SSR_a *
      (t(Wx ^ 2) / diag(t(Wx) %*% Wx))
  )
)
VIP

## [1] 0.5246196 0.5715532 1.5485804
, x3
.
.
.

```

## 4.3

$m \times n$ ,  $(n \times m)$

$$\mathbf{Y} = [\mathbf{y}_1 \cdots \mathbf{y}_m]$$

, ,  $\mathbf{T}$  . . .

### 4.3.1 R

4 2

```

train_df <- tribble(
  ~x1, ~x2, ~x3, ~x4, ~y1, ~y2,
  -1, -0.5, -1, 1, 5.9, -10,
  1, 1.1, -6, -6, -3.7, -2,
  0, 0.3, -5, -2, 1, 11,
  -3, -3.2, -9, 19, 7.7, -22,
  4, 1.2, 14, -12, -7.5, 4,
  -2, -2.6, -2, 9, 2.8, 1,
  1, 3.7, 9, -9, -6.2, 18
)

knitr::kable(
  train_df, booktabs = TRUE,
  align = rep("r", ncol(train_df)),
  caption = " "
)

```

, R pls plsr() PLS . , formula

. . .

Table 4.2:

x1	x2	x3	x4	y1	y2
-1	-0.5	-1	1	5.9	-10
1	1.1	-6	-6	-3.7	-2
0	0.3	-5	-2	1.0	11
-3	-3.2	-9	19	7.7	-22
4	1.2	14	-12	-7.5	4
-2	-2.6	-2	9	2.8	1
1	3.7	9	-9	-6.2	18

```
X <- as.matrix(train_df[, c("x1", "x2", "x3", "x4")])
Y <- as.matrix(train_df[, c("y1", "y2")])
plsrmulti_fit <- pls::pls(Y ~ X, ncomp = 3)
```

```
PLS           x1,...,x4      y1,y2      coef()
coef(plsrmulti_fit)
```

```
## , , 3 comps
##
##          y1          y2
## x1 -0.26509645 -2.8773570
## x2  0.08864959  2.7249194
## x3 -0.14258488  0.3377420
## x4  0.37330470 -0.7406377
```

```
summary()
```

```
summary(plsrmulti_fit)
```

```
## Data:    X dimension: 7 4
## Y dimension: 7 2
## Fit method: kernelpls
## Number of components considered: 3
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps
## X     86.59    99.00   99.81
## y1    81.71    81.89   82.19
## y2    53.98    56.50   65.99
```

### 4.3.2 PLS

#### 4.2.2

$$\mathbf{X} = \mathbf{T}\mathbf{P}^\top + \mathbf{E} \quad (4.15)$$

$$\mathbf{Y} = \mathbf{U}\mathbf{Q}^\top + \mathbf{F} \quad (4.16)$$

$$\mathbf{U} = \mathbf{T}\mathbf{B} + \mathbf{H} \quad (4.17)$$

$$(4.15) \quad , (m \times A) \quad \mathbf{Q} \quad \mathbf{Y} \quad \mathbf{U} \quad . \quad (4.16) \quad (n \times A) \quad \mathbf{U} \quad \mathbf{Y} \quad \mathbf{A} \\ \mathbf{B} \quad (A \times A) \quad (\text{diagonal matrix}) \quad , \quad \mathbf{U} \quad \mathbf{T} \quad . \quad (4.17) \quad \mathbf{T} \quad \mathbf{U} \quad ,$$

$$\mathbf{u}_a = b_a \mathbf{t}_a + \mathbf{h}_a, \quad a = 1, \dots, A$$

$$, b_a \quad \mathbf{B} \quad a$$

$$\mathbf{B} = \begin{bmatrix} b_1 & 0 & \cdots & 0 \\ 0 & b_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & b_A \end{bmatrix}$$

E, F, H

### 4.3.3 NIPALS

, NIPALS .  $t_a$  4.2.3  
 $t_a \ u_a$  (iterative) .

• [ 0 ] .  $a \leftarrow 1$ ,  $\mathbf{X}_a \leftarrow \mathbf{X}$ ,  $\mathbf{Y}_a \leftarrow \mathbf{Y}$ .

$$\bullet \quad \left[ \begin{array}{c} 1 \\ 2 \end{array} \right] \mathbf{X}_a \quad \mathbf{Y}_a \quad , \quad \mathbf{u}_a \quad \mathbf{u}_a \quad .$$

$$\bullet \quad \mathbf{w}_a = [w_{a1} \cdots w_{ak}]^\top$$

$$\mathbf{w}_a \leftarrow \mathbf{X}_a^\top \mathbf{u}_a / \mathbf{u}_a^\top \mathbf{u}_a$$

$$\bullet \quad [ -3 ] \qquad \mathbf{w}_a \qquad 1$$

$$\mathbf{w}_a \leftarrow \mathbf{w}_a / \sqrt{\mathbf{w}_a^\top \mathbf{w}_a}$$

• [ 4]  $t_a$   $X_a$

$$\mathbf{t}_a \leftarrow \mathbf{X}_a \mathbf{w}_a$$

$$\bullet \quad [ \begin{array}{c} 5 \end{array} ] \mathbf{Y}_a \quad , \quad \mathbf{t}_a \quad (\quad ) \mathbf{q}_a = [q_{a1} \cdots q_{am}]^\top$$

$$\mathbf{q}_a \leftarrow \mathbf{Y}_a^\top \mathbf{t}_a / \mathbf{t}_a^\top \mathbf{t}_a$$

$$\bullet \quad [ -6 ] \qquad q_a \qquad 1 \qquad .$$

$$\mathbf{q}_a \leftarrow \mathbf{q}_a / \sqrt{\mathbf{q}_a^\top \mathbf{q}_a}$$

- [ 7]  $\mathbf{u}_a = \mathbf{Y}_a \cdot \dots, \mathbf{q}_a \cdot \dots$   

$$\mathbf{u}_a \leftarrow \mathbf{Y}_a \mathbf{q}_a$$
- [ 8]  $(\dots) [ 2] \begin{bmatrix} 7 \\ \mathbf{t}_a \end{bmatrix} \mathbf{X}_a \cdot, \mathbf{X}_a \cdot, \mathbf{t}_a \cdot \mathbf{p}_a \cdot [ 9]$   

$$\mathbf{p}_a \leftarrow \mathbf{X}_a^\top \mathbf{t}_a / \mathbf{t}_a^\top \mathbf{t}_a$$
- [ 10]  $\mathbf{p}_a = 1 \cdot, \mathbf{t}_a \cdot \mathbf{w}_a \cdot$   

$$d \leftarrow \sqrt{\mathbf{p}_a^\top \mathbf{p}_a}, \mathbf{t}_a \leftarrow \mathbf{t}_a d, \mathbf{w}_a \leftarrow \mathbf{w}_a d, \mathbf{p}_a \leftarrow \frac{1}{d} \mathbf{p}_a$$
- [ 11]  $\mathbf{u}_a \cdot \mathbf{t}_a \cdot b_a \cdot$   

$$b_a \leftarrow \mathbf{u}_a^\top \mathbf{t}_a / \mathbf{t}_a^\top \mathbf{t}_a$$
- [ 12]  $\mathbf{X}_a \cdot \mathbf{Y}_a \cdot \mathbf{t}_a \cdot \mathbf{X}_{a+1}$   

$$\mathbf{X}_{a+1} \leftarrow \mathbf{X}_a - \mathbf{t}_a \mathbf{p}_a^\top, \mathbf{Y}_{a+1} \leftarrow \mathbf{Y}_a - b_a \mathbf{t}_a \mathbf{q}_a^\top$$
- [ 13]  $a \leftarrow a + 1 \cdot, [ 1] \cdot, [ 1] - [ 13] \cdot A$   

$$\text{nipals_plsr2} \cdot, \cdot, \cdot$$
  - X:  $(n \times k)$
  - Y:  $(n \times m)$
  - A:

```
nipals_plsr2 <- function(X, Y, A = NULL) {
  if (is.vector(Y)) {
    Y <- as.matrix(Y, ncol = 1L)
  }

  if (nrow(X) != nrow(Y)) stop("X and Y must have the same numbers of observations.")

  if (is_empty(A) || (A > min(dim(X)))) {
    A <- min(dim(X))
  }

  Tmat <- matrix(NA, nrow = nrow(X), ncol = A)
  Umat <- matrix(NA, nrow = nrow(X), ncol = A)
  Wmat <- matrix(NA, nrow = ncol(X), ncol = A)
  Pmat <- matrix(NA, nrow = ncol(X), ncol = A)
  Qmat <- matrix(NA, nrow = ncol(Y), ncol = A)
  Bmat <- diag(nrow = A)

  for (a in seq_len(A)) {
    # 1
    ...
```

```

j <- sample.int(ncol(Y), size = 1L)
Umat[, a] <- Y[, j]

while (TRUE) {
  # 2
  Wmat[, a] <- coef(lm(X ~ -1 + Umat[, a]))

  # 3
  Wmat[, a] <- Wmat[, a] / sqrt(sum(Wmat[, a]^ 2))

  # 4
  Tmat[, a] <- X %*% Wmat[, a]

  # 5
  Qmat[, a] <- coef(lm(Y ~ -1 + Tmat[, a]))

  # 6
  Qmat[, a] <- Qmat[, a] / sqrt(sum(Qmat[, a]^ 2))

  # 7
  u_new <- Y %*% Qmat[, a]

  # 8
  if (all(near(u_new, Umat[, a]))) break

  Umat[, a] <- u_new
}

# 9
Pmat[, a] <- coef(lm(X ~ -1 + Tmat[, a]))

# 10
p_size <- sqrt(sum(Pmat[, a]^ 2))
Tmat[, a] <- Tmat[, a] * p_size
Wmat[, a] <- Wmat[, a] * p_size
Pmat[, a] <- Pmat[, a] / p_size

# 11
Bmat[a, a] <- coef(lm(Umat[, a] ~ -1 + Tmat[, a]))

# 12
X <- X - Tmat[, a] %*% t(Pmat[, a])
Y <- Y - Bmat[a, a] * Tmat[, a] %*% t(Qmat[, a])
}

```

```

    return(list(T = Tmat, W = Wmat, P = Pmat,
                U = Umat, Q = Qmat, B = Bmat))
}

nipals_fit2 <- nipals_plsr2(X, Y, A = 3)
nipals_fit2

## $T
##      [,1]      [,2]      [,3]
## [1,] 1.5777938 -0.4117531 0.44707709
## [2,] -2.2993972 -8.0194451 -0.79803681
## [3,] 0.8145895 -5.1398023 -0.30016399
## [4,] 21.3867331  3.2065797 -0.01083136
## [5,] -17.8784038  5.7770058 -1.68928119
## [6,]  9.2701258  3.6198261 -0.09042238
## [7,] -12.8714412  0.9675888  2.44165865
##
## $W
##      [,1]      [,2]      [,3]
## [1,] -0.1476019  0.4020251 -0.6921766
## [2,] -0.1848598 -0.4862964  0.5027723
## [3,] -0.5065102  0.8236460  0.4768770
## [4,]  0.8312518  0.4651155  0.2794808
##
## $P
##      [,1]      [,2]      [,3]
## [1,] -0.1648502 -0.002634502 -0.5484068
## [2,] -0.1588038 -0.130002426  0.6241044
## [3,] -0.5486716  0.859489687  0.4559029
## [4,]  0.8040928  0.494337847  0.3192118
##
## $U
##      [,1]      [,2]      [,3]
## [1,] 11.60599122  9.401433 -8.559602
## [2,] -0.03696152  3.340983 -7.252566
## [3,] -9.14720168 -11.437784  9.474719
## [4,] 22.98172912  6.021677 -4.914480
## [5,] -7.12619591  9.125035 -6.794076
## [6,]  0.47754875 -7.914057  9.259791
## [7,] -18.75490997 -8.537286  8.786214
##
## $Q
##      [,1]      [,2]      [,3]
## [1,] 0.4832295  0.1202142 0.07948906
## [2,] -0.8754937 -0.9927480 0.99683574

```

```
##  
## $B  
## [,1]      [,2]      [,3]  
## [1,] 0.8443757 0.0000000 0.000000  
## [2,] 0.0000000 0.4255916 0.000000  
## [3,] 0.0000000 0.0000000 3.206299
```

#### 4.3.4

NIPALS

$\mathbf{X} \quad \mathbf{Y}$

$\mathbf{T}$

$\mathbf{X} \quad \mathbf{W}, \quad \mathbf{P}$

$$\mathbf{T} = \mathbf{X}\mathbf{W}(\mathbf{P}^\top\mathbf{W})^{-1} \quad (4.18)$$

(4.17) ,

$$\mathbf{U} = \mathbf{X}\mathbf{W}(\mathbf{P}^\top\mathbf{W})^{-1} \mathbf{B} + \mathbf{H} \quad (4.19)$$

(4.16) ,

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\mathbf{W}(\mathbf{P}^\top\mathbf{W})^{-1} \mathbf{B}\mathbf{Q}^\top + \mathbf{H}\mathbf{Q}^\top + \mathbf{F} \\ &= \mathbf{X}\mathbf{B}_{PLS} + \mathbf{G} \end{aligned} \quad (4.20)$$

$\mathbf{G} = \mathbf{H}\mathbf{Q}^\top + \mathbf{F} \quad \mathbf{X} \quad \mathbf{Y} \quad \dots, \text{PLS} \quad \mathbf{X}$

$$\mathbf{B}_{PLS} = \mathbf{W}(\mathbf{P}^\top\mathbf{W})^{-1} \mathbf{B}\mathbf{Q}^\top \quad (4.21)$$

```
beta_pls2 <- nipals_fit2$W %*%  
  solve(t(nipals_fit2$P) %*% nipals_fit2$W) %*%  
  nipals_fit2$B %*% t(nipals_fit2$Q)  
beta_pls2
```

```
## [,1]      [,2]  
## [1,] -0.26509645 -2.8773570  
## [2,]  0.08864959  2.7249194  
## [3,] -0.14258488  0.3377420  
## [4,]  0.37330470 -0.7406377
```

4.3.1 R pls

```
all(near(beta_pls2, coef(plsr_multi_fit)[, , 1]))
```

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

### 4.3.5

$\mathbf{Y}$

$$SST = SSR + SSE$$

$$, \quad SSR$$

$$\begin{aligned} SSR &= \sum_{a=1}^A SSR_a \\ &= \sum_{a=1}^A SS(b_a \mathbf{t}_a \mathbf{q}_a^\top) \\ &= \sum_{a=1}^A b_a^2 SS(\mathbf{t}_a) \end{aligned} \tag{4.22}$$

```
, SSR_a      t_a      Y      .
SSR_a <- diag(
  t(nipals_fit2$T %*% nipals_fit2$B) %*%
  (nipals_fit2$T %*% nipals_fit2$B)
)
SSR_a
```

```
## [1] 739.40663 26.91455 100.23860
```

```
SSR_a      SST      Y      .
SST <- sum(Y ^ 2)
SSR_a / SST
```

```
## [1] 0.58621653 0.02133840 0.07947119
```

,       $\mathbf{t}_a$        $\mathbf{Y}_j$        $SSR_{aj}$  , .

$$SSR_{aj} = q_{ja}^2 SSR_a \tag{4.23}$$

```
SSR_aj <- diag(SSR_a) %*% t(nipals_fit2$Q ^ 2)
SSR_aj
```

```
##          [,1]      [,2]
## [1,] 172.6593685 566.74726
## [2,]  0.3889542 26.52560
## [3,]  0.6333587 99.60524
```

```
SSRaj j      yj      SS(yj) , yj   ta .  
SS_j <- colSums(Y ^ 2)  
SSR_aj %*% diag(1 / SS_j)  
  
## [,1]      [,2]  
## [1,] 0.817051715 0.53975930  
## [2,] 0.001840593 0.02526248  
## [3,] 0.002997154 0.09486213  
  
y1      t1      , y2      t2  t3 .
```



## **Part II**

**2 -**



# Chapter 5

(classification analysis) . . . (attribute) . . . (object) . . . (class,  
category) . . . , 3 3 2 3 .  
( ) . . . (classification rule) . . . .  
sample) . . . (learning

## 5.1 R

package	version
tidyverse	1.3.1
stats	4.1.0
class	7.3-19

## 5.2

- $N$  . . .  $\{(\mathbf{x}_i, y_i)\}_{i=1,\dots,N}$  . . .  
•  $\mathbf{x}_i$ :  $i$  . . .  $(\mathbf{x}_i = [x_{i1} x_{i2} \cdots x_{ip}]^\top)$   
•  $J$ : . . .  
•  $y_i$ :  $i$  ;  $y_i \in \{1, 2, \dots, J\}$

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$$

1,  $\dots$ ,  $J$  . . . , . . . (classifier) . . .  $r(\mathbf{x})$  . . . ,  $r(\mathbf{x})$  . . . ,

$$\frac{r(\mathbf{x}_i) - y_i}{J} \quad \dots, \quad r(\mathbf{x}_i) = y_i \quad \dots, \quad \frac{10}{\dots}$$

- 1. : ,
- 2. : CART, C4.5, CHAID
- 3. : (support vector machine; SVM)
- 4. : (neural network)

6 , 7 , 8 , 9 .

## 5.3

### 5.3.1

(nearest neighbor classification)  
 $\dots, k, \dots, k-$  (k-nearest neighbor method)  
 $\dots, 11 \dots$

#### 5.3.1.1 R

```
7
train_df <- tribble(
  ~id, ~x1, ~x2, ~y,
  1, 5, 7, 1,
  2, 4, 3, 2,
  3, 7, 8, 2,
  4, 8, 6, 2,
  5, 3, 6, 1,
  6, 2, 5, 1,
  7, 9, 6, 2
) %>%
  mutate(y = factor(y, levels = c(1, 2)))

knitr::kable(
  train_df, booktabs = TRUE,
  align = c('r', 'r', 'r', 'r'),
  col.names = c(' ', '$x_1$', '$x_2$', ' '),
  caption = ' '
)
```

class knn.cv ( $\dots, k$ )  
 leave-one-out cross validation .

leave-one-out cross validation .

Table 5.1

Table 5.1:

	\$x\_1\$	\$x\_2\$		
1	5	7	1	
2	4	3	2	
3	7	8	2	
4	8	6	2	
5	3	6	1	
6	2	5	1	
7	9	6	2	

Table 5.2: -

	\$x\_1\$	\$x\_2\$			
1	5	7	1	2	
2	4	3	2	1	
3	7	8	2	2	
4	8	6	2	2	
5	3	6	1	1	
6	2	5	1	1	
7	9	6	2	2	

```

y_hat <- class::knn.cv(
  train = train_df[, c("x1", "x2")],
  cl = train_df$y,
  k = 3
)

train_df %>%
  mutate(y_hat = y_hat) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r', 'r'),
    col.names = c(' ', '$x\_1$', '$x\_2$', ' ', ' '),
    caption = ' '
  )

class      knn
3-          .
test_df <- tribble(
  ~id, ~x1, ~x2,

```

$(6, 7)^T \quad (4, 2)^T$

Table 5.3:

	\$x\_1\$	\$x\_2\$	-
8	6	7	2
9	4	2	1

```

  8, 6, 7,
  9, 4, 2
)

y_hat <- class::knn(
  train = train_df %>% select(x1, x2),
  test = test_df %>% select(x1, x2),
  cl = train_df$y,
  k = 3
)

test_df %>%
  mutate(y_hat = y_hat) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r'),
    col.names = c(' ', '$x_1$', '$x_2$', ' '),
    caption = ' '
)

```

### 5.3.1.2

*k-*

- [ 1]  $k$
- [ 2]  $\mathbf{z}$
- 2-1.  $\mathbf{x}_i$
- 2-2.  $k$
- 2-3.  $k$

Table 5.1  $(6, 7)^\top \quad (4, 2)^\top$

[ 1]  $k$

```

train_pairwise_dist <- dist(train_df[, c("x1", "x2")], upper = TRUE) %>%
  broom::tidy()

train_pairwise_dist

```

```
## # A tibble: 42 x 3
##   item1 item2 distance
##   <fct> <fct>     <dbl>
## 1 1     2     4.12
## 2 1     3     2.24
## 3 1     4     3.16
## 4 1     5     2.24
## 5 1     6     3.61
## 6 1     7     4.12
## 7 2     1     4.12
## 8 2     3     5.83
## 9 2     4     5
## 10 2    5     3.16
## # ... with 32 more rows
## (rank) .
train_nn_rank <- train_pairwise_dist %>%
  group_by(item1) %>%
  mutate(nn_rank = rank(distance, ties.method = "random")) %>%
  ungroup() %>%
  arrange(item1, nn_rank)

train_nn_rank

## # A tibble: 42 x 4
##   item1 item2 distance nn_rank
##   <fct> <fct>     <dbl>   <int>
## 1 1     3     2.24     1
## 2 1     5     2.24     2
## 3 1     4     3.16     3
## 4 1     6     3.61     4
## 5 1     7     4.12     5
## 6 1     2     4.12     6
## 7 2     6     2.83     1
## 8 2     5     3.16     2
## 9 2     1     4.12     3
## 10 2    4     5         4
## # ... with 32 more rows
## k      k-       .
loo_cv <- bind_cols(
  train_nn_rank %>% select(item1, nn_rank),
  map2_dfr(
    train_nn_rank$item1,
```

```

train_nn_rank$nn_rank,
function(.x, .y, df, y) {
  df %>%
    filter(
      item1 == .x,
      nn_rank <= .y
    ) %>%
    mutate(y = y[item2]) %>%
    count(y) %>%
    slice(which.max(n))
},
df = train_nn_rank,
y = train_df$y
)
) %>%
rename(k = nn_rank, y_hat = y) %>%
mutate(y = train_df$y[item1])

loo_cv

## # A tibble: 42 x 5
##   item1     k y_hat     n y
##   <fct> <int> <fct> <int> <fct>
## 1 1         1 2     1 1
## 2 1         2 1     1 1
## 3 1         3 2     2 1
## 4 1         4 1     2 1
## 5 1         5 2     3 1
## 6 1         6 2     4 1
## 7 2         1 1     1 2
## 8 2         2 1     2 2
## 9 2         3 1     3 2
## 10 2        4 1     3 2
## # ... with 32 more rows
      , k
loo_cv %>%
  mutate(is_correct = (y == y_hat)) %>%
  group_by(k) %>%
  summarize(accuracy = mean(is_correct)) %>%
  arrange(desc(accuracy))

## # A tibble: 6 x 2
##       k accuracy
##   <int>     <dbl>
## 1     1     0.714

```

```

## 2      2      0.714
## 3      3      0.714
## 4      4      0.571
## 5      5      0.286
## 6      6      0

,           k           k = 3     k
[ 2]       3-       .

test_df <- tribble(
  ~id, ~x1, ~x2,
  8, 6, 7,
  9, 4, 2
)

test_train_dist <- flexclust::dist2(
  test_df %>% select(x1, x2),
  train_df %>% select(x1, x2)
) %>%
  as_tibble() %>%
  `names<-`(seq_len(nrow(train_df))) %>%
  mutate(item1 = seq_len(nrow(test_df))) %>%
  gather(key = "item2", value = "distance", -item1) %>%
  mutate(item2 = as.numeric(item2))

## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.`.name_repair
## Using compatibility `.`.name_repair`.

## Warning: The `value` argument of `names<-` must be a character vector as of
## tibble 3.0.0.

test_train_dist

## # A tibble: 14 x 3
##   item1 item2 distance
##   <int> <dbl>    <dbl>
## 1     1     1      1
## 2     2     2      5.10
## 3     3     1      2      4.47
## 4     4     2      2      1
## 5     5     1      3      1.41
## 6     6     2      3      6.71
## 7     7     1      4      2.24
## 8     8     2      4      5.66
## 9     9     1      5      3.16
## 10   10    2      5      4.12

```

```

## 11      1      6      4.47
## 12      2      6      3.61
## 13      1      7      3.16
## 14      2      7      6.40
            .
test_nn <- test_train_dist %>%
  group_by(item1) %>%
  arrange(distance) %>%
  mutate(nn_rank = row_number()) %>%
  filter(nn_rank <= 3) %>%
  ungroup()

test_nn

## # A tibble: 6 x 4
##   item1 item2 distance nn_rank
##   <int> <dbl>     <dbl>    <int>
## 1     1     1       1           1
## 2     2     2       1           1
## 3     1     3      1.41        2
## 4     1     4      2.24        3
## 5     2     6      3.61        2
## 6     2     5      4.12        3
            ,
            .

test_yhat <- test_nn %>%
  mutate(
    id = test_df$id[item1],
    y = train_df$y[item2]
  ) %>%
  group_by(id, y) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  slice(1) %>%
  ungroup() %>%
  rename(y_hat = y)

## `summarise()` has grouped output by 'id'. You can override using the `.`groups` argument

test_yhat

## # A tibble: 2 x 3
##       id y_hat     n
##   <dbl> <fct> <int>
## 1     8 2          2
## 2     9 1          2

```

Table 5.4:

	$(\$x\_1\$)$	$(\$x\_2\$)$	$(\$y\$)$
1		20	1
2		20	2
3		30	1
4		40	1
5		10	1
6		20	2
7		20	1
8		30	2
9		40	2

,  $(6, 7)^\top$  2 ,  $(4, 2)^\top$  1 .

### 5.3.2

(Naive Bayesian)

#### 5.3.2.1 R

```

9
.
train_df <- tribble(
  ~id, ~x1, ~x2, ~y,
  1, " ", "20 ", 1,
  2, " ", "20 ", 2,
  3, " ", "30 ", 1,
  4, " ", "40 ", 1,
  5, " ", "10 ", 1,
  6, " ", "20 ", 2,
  7, " ", "20 ", 1,
  8, " ", "30 ", 2,
  9, " ", "40 ", 2
) %>%
  mutate(y = factor(y, levels = c(1, 2)))

knitr::kable(
  train_df, booktabs = TRUE,
  align = c('r', 'r', 'r', 'r'),
  col.names = c(' ', ' ($x_1$)', ' ($x_2$)', ' ($y$)'),
  caption = ''
)

```

```

e1071    naiveBayes      ,
nb_fit <- e1071::naiveBayes(formula = y ~ x1 + x2, data = train_df)
print(nb_fit)

##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##      1      2
## 0.5555556 0.4444444
##
## Conditional probabilities:
##   x1
## Y
##   1 0.60 0.40
##   2 0.25 0.75
##
##   x2
## Y 10 20 30 40
##   1 0.20 0.40 0.20 0.20
##   2 0.00 0.50 0.25 0.25

#
y_hat <- predict(nb_fit, train_df)

#
nb_posterior <- predict(nb_fit, train_df, type = "raw") %>%
  as_tibble() %>%
  `colnames<-`(str_c("p", levels(train_df$y)))

train_df %>%
  mutate(y_hat = y_hat) %>%
  bind_cols(nb_posterior) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r', 'r'),
    col.names = c(' ', ' ($x_1$)', ' ($x_2$)',
                 ' ($y$)', ' ($\hat{y}$)'),
    str_c(' ($y$ = ', levels(train_df$y), ')'))

```

Table 5.5:

$(\$x\_1\$)$	$(\$x\_2\$)$	$(\$y\$)$	$(\$\\hat{y}\$)$	$(\$y\$ = 1)$	$(\$y\$ = 2)$
1	20	1	1	0.7058824	0.2941176
2	20	2	1	0.7058824	0.2941176
3	30	1	1	0.7058824	0.2941176
4	40	1	1	0.7058824	0.2941176
5	10	1	1	0.9925558	0.0074442
6	20	2	2	0.3478261	0.6521739
7	20	1	2	0.3478261	0.6521739
8	30	2	2	0.3478261	0.6521739
9	40	2	2	0.3478261	0.6521739

```

    caption = ' '
)
,
      10
predict(nb_fit, tibble(x1 = " ", x2 = "10"))

## [1] 1
## Levels: 1 2

```

### 5.3.2.2

$$\begin{array}{cccccc}
 \mathbf{x} & & y & & . & \\
 P(y) & & y & , P(y|\mathbf{x}) & & P(\mathbf{x}|y) \\
 y & . & & & & \\
 & & P(\mathbf{x}|y) & , p & & \mathbf{x} = (x_1, x_2, \dots, x_p) \\
 & & . & & &
 \end{array}$$

$$\begin{array}{c}
 P(x_a | x_{a+1}, x_{a+2}, \dots, x_p, y) = P(x_a | y) \\
 , \quad (5.1) \\
 .
 \end{array}$$

$$P(y|\mathbf{x}) \propto P(y) \prod_{a=1}^p P(x_a|y), y = 1, \dots, J \quad (5.2)$$

$$, \quad 5.4 \quad P(y) \quad .$$

```

prior_prob <- train_df %>%
  group_by(y) %>%
  summarise(n = n()) %>%
  mutate(prior = n / sum(n)) %>%
  select(-n)

prior_prob

## # A tibble: 2 x 2
##   y     prior
##   <fct> <dbl>
## 1 1     0.556
## 2 2     0.444

,      5.4           $P(x_a | y)$     .

condition_prob <- train_df %>%
  gather(key = "variable", value = "value", x1, x2) %>%
  group_by(y, variable, value) %>%
  summarise(n = n()) %>%
  mutate(cond_prob = n / sum(n)) %>%
  select(-n) %>%
  ungroup() %>%
  complete(y, nesting(variable, value), fill = list(cond_prob = 0))

## `summarise()` has grouped output by 'y', 'variable'. You can override using the `.`g
condition_prob

## # A tibble: 12 x 4
##   y     variable value cond_prob
##   <fct> <chr>    <chr>    <dbl>
## 1 1     x1        10       0.6
## 2 1     x1        20       0.4
## 3 1     x2        10       0.2
## 4 1     x2        20       0.4
## 5 1     x2        30       0.2
## 6 1     x2        40       0.2
## 7 2     x1        10       0.25
## 8 2     x1        20       0.75
## 9 2     x2        10       0
## 10 2    x2        20       0.5
## 11 2    x2        30       0.25
## 12 2    x2        40       0.25

(5.2) , .

```

```

posterior_prob <- train_df %>%
  select(-y) %>%
  gather(key = "variable", value = "value", x1, x2) %>%
  inner_join(condition_prob, by = c("variable", "value")) %>%
  group_by(id, y) %>%
  summarize(cond_prob = reduce(cond_prob, `*`)) %>%
  inner_join(prior_prob, by = "y") %>%
  mutate(posterior_unadjust = prior * cond_prob) %>%
  mutate(posterior = posterior_unadjust / sum(posterior_unadjust)) %>%
  select(id, y, posterior) %>%
  ungroup()

## `summarise()` has grouped output by 'id'. You can override using the ` `.groups` argument.
posterior_prob %>%
  spread(key = y, value = posterior)

## # A tibble: 9 x 3
##       id     `1`     `2`
##   <dbl> <dbl> <dbl>
## 1     1  0.706  0.294
## 2     2  0.706  0.294
## 3     3  0.706  0.294
## 4     4  0.706  0.294
## 5     5     1     0
## 6     6  0.348  0.652
## 7     7  0.348  0.652
## 8     8  0.348  0.652
## 9     9  0.348  0.652

posterior_prob %>%
  group_by(id) %>%
  top_n(1, posterior) %>%
  slice(1)

## # A tibble: 9 x 3
## # Groups:   id [9]
##       id y     posterior
##   <dbl> <fct>    <dbl>
## 1     1 1         0.706
## 2     2 1         0.706
## 3     3 1         0.706
## 4     4 1         0.706
## 5     5 1         1
## 6     6 2         0.652

```

```
## 7      7 2      0.652
## 8      8 2      0.652
## 9      9 2      0.652
```

### 5.3.2.3 R

```
5.3.2.1          e1071      naiveBayes
nb_fit <- e1071::naiveBayes(formula = y ~ x1 + x2, data = train_df)

naiveBayes      component  apriori           table   ,
.

str(nb_fit$apriori)

##  'table' int [1:2(1d)] 5 4
##  - attr(*, "dimnames")=List of 1
##    ..$ Y: chr [1:2] "1" "2"
,
                               (prior distribution)
.

nb_fit$apriori %>%
  broom::tidy() %>%
  mutate(p = n / sum(n))

## Warning: 'tidy.table' is deprecated.
## See help("Deprecated")

## # A tibble: 2 x 3
##       Y     n     p
##   <chr> <int> <dbl>
## 1 1      5  0.556
## 2 2      4  0.444

tables
.

nb_fit$tables

## $x1
##   x1
##   Y
##   1 0.60 0.40
##   2 0.25 0.75
##
## $x2
##   x2
##   Y   10 20 30 40
##   1 0.20 0.40 0.20 0.20
##   2 0.00 0.50 0.25 0.25
```

```
predict           , threshold           .   0.001 ,      0.1%
.
predict(nb_fit, newdata = train_df[5, ], type = "raw")

##          1          2
## [1,] 0.9925558 0.007444169
0.01      ,      2
.
predict(nb_fit, newdata = train_df[5, ], type = "raw", threshold = 0.01)

##          1          2
## [1,] 0.9302326 0.06976744
```



# Chapter 6

(logistic regression) 2 ( / , / , / ) , 3  
(ordinal) (nominal)

## 6.1 R

package	version
tidyverse	1.3.1
stats	4.1.0
nnet	7.3-16
MASS	7.3-54
VGAM	1.1-5

## 6.2

### 6.2.1 R

```
train_df <- tribble(  
  ~id, ~x1, ~x2, ~x3, ~y,  
  1, 0, 8, 2, " ",  
  2, 1, 7, 1, " ",  
  3, 0, 9, 0, " ",  
  4, 1, 6, 4, " ",  
  5, 1, 8, 2, " ",  
  6, 0, 7, 3, " ",  
  7, 0, 7, 0, " ",  
  8, 1, 6, 1, " ",  
  9, 0, 7, 2, " ",
```

Table 6.1: /

	(\$x_1\$)	(\$x_2\$)	(\$x_3\$)	(y)
1	0	8	2	
2	1	7	1	
3	0	9	0	
4	1	6	4	
5	1	8	2	
6	0	7	3	
7	0	7	0	
8	1	6	1	
9	0	7	2	
10	0	8	1	
11	0	5	2	
12	1	8	0	
13	0	6	3	
14	1	7	2	
15	0	6	1	

```

10, 0, 8, 1, " ",
11, 0, 5, 2, " ",
12, 1, 8, 0, " ",
13, 0, 6, 3, " ",
14, 1, 7, 2, " ",
15, 0, 6, 1, " "
) %>%
  mutate(y = factor(y, levels = c(" ", " ")))

knitr::kable(train_df, booktabs = TRUE,
             align = c('r', 'r', 'r', 'r', 'r'),
             col.names = c(' ', ' ($x_1$)', ' ($x_2$)', ' ($x_3$)', ' (y)'), 
             caption = ' / ')

```

Table 6.1       $x_1, x_2, x_3$        $y$     ( $= 0, = 1$ )      15      train\_df  
data frame .

```

glm .
glm_fit <- glm(y ~ x1 + x2 + x3, family = binomial(link = "logit"), data = train_df)

knitr::kable(
  glm_fit %>% broom::tidy(),
  booktabs = TRUE,

```

Table 6.2: / Logistic Regression

term	estimate	std.error	statistic	p.value
(Intercept)	-30.510836	18.018256	-1.693329	0.0903929
x1	2.031278	1.983692	1.023989	0.3058406
x2	3.470671	2.074978	1.672631	0.0944000
x3	2.414387	1.396372	1.729043	0.0838015

```

caption = " / Logistic Regression "
# caption = "Table \\\@ref(tab:binary-logistic-reg-train-data) Logistic Regression "
)

```

Table 6.2 estmate std.error, (standardized) statistic  
 $(= \text{estmate} / \text{std.error})$ ,  $H_0: \text{statistic} = 0$  p.value .

### 6.2.2

- 2 .
- N  $\{(\mathbf{x}_i, y_i)\}_{i=1,\dots,N}$  .
- $\mathbf{x}_i \in \mathbb{R}^p$ :  $p$  ( $\mathbf{x}_i = [x_{i1} x_{i2} \dots x_{ip}]^\top$ )
- $y_i$ : 0 1 (indicator variable)
- $\mathbf{x}_i$   $y_i$   $P_i$  .

$$P_i = P(y_i = 1 | \mathbf{x}_i) \quad (6.1)$$

$$= E[y_i | \mathbf{x}_i] \quad (6.2)$$

$$= \frac{\exp(\beta_0 + \beta^\top \mathbf{x}_i)}{1 + \exp(\beta_0 + \beta^\top \mathbf{x}_i)} \quad (6.3)$$

$$\beta \in \mathbb{R}^p \quad \mathbf{x}_i \quad (\beta = [\beta_1 \beta_2 \dots \beta_p]^\top).$$

$$(6.3) \quad \mathbf{x}_i \quad 0 \quad 1 \quad , \quad \mathbf{x}$$

(logit)

$$\text{logit}(P_i) = \ln \left[ \frac{P_i}{1 - P_i} \right] \quad (6.4)$$

$$= \ln(\exp(\beta_0 + \beta^\top \mathbf{x}_i)) \quad (6.5)$$

$$= \beta_0 + \beta^\top \mathbf{x}_i \quad (6.6)$$

$$(6.6) \quad P_i \quad 0 \quad 1 \quad y_i \quad , \quad P_i \quad (\text{latent variable}) \quad .$$

$$y_i = \begin{cases} 1 & \text{if } \beta_0 + \beta^\top \mathbf{x}_i + \varepsilon_i > 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.7)$$

$$(6.7) \quad \varepsilon_i \quad (\text{standard logistic distribution}) \quad .$$

### 6.2.3

(maximum likelihood estimation)  $\cdot N$

$$L = \prod_{i=1}^N P_i^{y_i} (1 - P_i)^{1-y_i}$$

$$\log L = \sum_{i=1}^N y_i \log P_i + \sum_{i=1}^N (1 - y_i) \log(1 - P_i) \quad (6.8)$$

$$= \sum_{i=1}^N y_i \log \frac{P_i}{1 - P_i} + \sum_{i=1}^N \log(1 - P_i) \quad (6.9)$$

$$= \sum_{i=1}^N y_i (\beta_0 + \beta^\top \mathbf{x}_i) - \sum_{i=1}^N \log(1 + \exp(\beta_0 + \beta^\top \mathbf{x}_i)) \quad (6.10)$$

$$= \sum_{i=1}^N y_i \left( \beta_0 + \sum_{j=1}^p \beta_j x_{ij} \right) - \sum_{i=1}^N \log \left( 1 + \exp \left( \beta_0 + \sum_{j=1}^p \beta_j x_{ij} \right) \right) \quad (6.11)$$

$$(6.11) \quad \beta_0, \beta_1, \dots, \beta_p \quad . \quad \text{Newton-Raphson algorithm} \\ \text{quasi-Newton} \quad (, 2012), \quad \text{gradient descent}$$

#### 6.2.3.1

$$(6.3) \quad P(y_i = 0 \mid \mathbf{x}_i) = 1 - P_i,$$

$$\frac{\exp(z)}{1 + \exp(z)} = \frac{1}{1 + \exp(-z)}$$

$$P(y = y_i \mid \mathbf{x}_i, \beta_0, \beta) = \frac{1}{1 + \exp((1 - 2y_i)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))}$$

(6.11)

$$\log \prod_{i=1}^N P(y = y_i | \mathbf{x}_i, \beta_0, \beta) = - \sum_{i=1}^N \log \left( 1 + \exp \left( (1 - 2y_i)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}) \right) \right)$$

$$f(\beta_0, \beta) = \sum_{i=1}^N \log \left( 1 + \exp \left( (1 - 2y_i)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}) \right) \right) \quad (6.12)$$

- 1.  $\beta_0, \beta_1, \dots, \beta_j$
- 2. (6.12)
- 3. 2 (step size)
- 4. 2-3

(6.12)

$$\begin{aligned} \frac{\partial f}{\partial \beta_0} &= \sum_{i=1}^N (1 - 2y_i) \frac{\exp((1 - 2y_i)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))}{1 + \exp((1 - 2y_i)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))} \\ &= \sum_{i=1}^N \frac{1 - 2y_i}{1 + \exp((2y_i - 1)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))} \end{aligned}$$

$$\begin{aligned} \frac{\partial f}{\partial \beta_j} &= \sum_{i=1}^N (1 - 2y_i) x_{ij} \frac{\exp((1 - 2y_i)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))}{1 + \exp((1 - 2y_i)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))} \\ &= \sum_{i=1}^N \frac{(1 - 2y_i) x_{ij}}{1 + \exp((2y_i - 1)(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))} \end{aligned}$$

```

        update_beta
update_beta <- function(x, y, beta0 = 0, beta = rep(0, dim(x)[2]), alpha = 0.01) {
  #
  denominator <- 1 + exp((2 * y - 1) * (beta0 + (x %*% beta)))

  # intercept
  beta0_numerator <- 1 - 2 * y
  beta0_update = sum(beta0_numerator / denominator)

  # intercept

```

```

beta_numerator <- sweep(x, MARGIN = 1, STATS = 1 - 2 * y, FUN = "*")
beta_update = apply(beta_numerator, MARGIN = 2,
                     function(x) sum(x / denominator))

#
beta0 <- beta0 - alpha * beta0_update
beta <- beta - alpha * beta_update

return(list(beta0 = beta0, beta = beta))
}

estimate_beta . .
alpha . .
alpha
caltculate_loglik <- function(x, y,
                                beta0 = 0,
                                beta = rep(0, dim(x)[2])) {
  sum(y * (beta0 + (x %*% beta))) -
  sum(log(1 + exp(beta0 + (x %*% beta))))
}

estimate_beta <- function(x, y,
                           beta0 = 0,
                           beta = rep(0, dim(x)[2]),
                           alpha = 0.01,
                           conv_threshold = 1e-5,
                           max_iter = 1e+5) {
  new_beta0 <- beta0
  new_beta <- beta
  conv <- FALSE

  i_iter <- 0
  while(i_iter < max_iter) {
    res <- update_beta(x, y, new_beta0, new_beta, alpha)

    if(abs(caltculate_loglik(x, y, beta0, beta)
           - caltculate_loglik(x, y, res$beta0, res$beta))
       < conv_threshold) {
      conv <- TRUE
      break
    }

    new_beta0 <- res$beta0
    new_beta <- res$beta

    i_iter <- i_iter + 1
  }
}

```

```

    }

  return(list(conv = conv, beta0 = new_beta0, beta = new_beta))
}

```

Table 6.1

```

res <- estimate_beta(train_df[, c("x1", "x2", "x3")] %>% as.matrix(),
                      train_df$y %>% as.numeric() - 1,
                      alpha = 0.015,
                      conv_threshold = 1e-6)

print(res)

## $conv
## [1] FALSE
##
## $beta0
## [1] -30.39189
##
## $beta
##      x1        x2        x3
## 2.022808 3.457019 2.405969

```

R glm (Table 6.2)

### 6.2.3.2

R glm (iteratively reweighted least squares; IRLS IWLS)

$$\text{logit}(y_i) = \beta_0 + \beta^\top \mathbf{x}_i + \varepsilon_i$$

,  $y_i \in [0, 1]$ ,  $\text{logit}(y_i) \in (-\infty, \infty)$ , (6.3)

$$P = \frac{\exp(\beta_0 + \beta^\top \mathbf{x})}{1 + \exp(\beta_0 + \beta^\top \mathbf{x})}$$

(Taylor series)  $\text{logit}(y)$ 

$$\begin{aligned} g(y) &= \text{logit}(P) + (y - P) \frac{\partial \text{logit}(P)}{\partial P} \\ &= \log \frac{P}{1 - P} + (y - P) \left( \frac{1}{P} + \frac{1}{1 - P} \right) \end{aligned}$$

$g(y_i) = \beta_0 + \beta^\top \mathbf{x}_i + \varepsilon_i$

$\varepsilon_i \sim P_i$ , (ordinary least squares; OLS)  
 (weighted least squares; WLS).

$$w_i = P_i(1 - P_i)$$

,

```

P_i      w_i      calculate_weight
calculate_weight <- function(x, beta0 = 0,
                                beta = rep(0, dim(x)[2])) {
  #   y  1
  P <- (1 + exp(- beta0 - (x %*% beta)))^(-1) %>% drop()

  #
  w <- P * (1 - P)

  return(list(P = P, w = w))
}

calculate_beta
R      lm
calculate_beta <- function(x, y, P, w) {
  #   0  1
  #   logit
  logit_derivative <- 1/P + 1/(1 - P)
  is_good <- !is.nan(logit_derivative)

  #
  #   0  1
  if(all(!is_good)) return(NULL)

  #
  g_y <- log(P[is_good]) -
    log(1 - P[is_good]) +
    (y[is_good] - P[is_good]) * logit_derivative

  #
  df <- bind_cols(as_tibble(x) %>%
                    `colnames<-`(`colnames(x)),
                    tibble(g_y = g_y))
  lm(g_y ~ ., data = df, subset = is_good, weights = w)
}

```

calculate_weight	calculate_beta	Table 6.1
.	.	.
1/10000		

```

X <- train_df[, c("x1", "x2", "x3")] %>% as.matrix()
y <- train_df$y %>% as.numeric() - 1

weight <- calculate_weight(X)
for(i in 1:10) {
  wls_fit <- calculate_beta(X, y, weight$P, weight$w)

  if(is.null(wls_fit)) {break}

  new_weight <- calculate_weight(x = X,
                                   beta0 = coef(wls_fit)[1],
                                   beta = coef(wls_fit)[-1])

  if(max(abs(new_weight$w - weight$w)) < 1e-4) {break}

  weight <- new_weight
}

coef(wls_fit)

## (Intercept)          x1          x2          x3
## -30.510837    2.031278   3.470671   2.414387

```

7 , glm (Table 6.2) .

## 6.3

### 6.3.1 R

```

train_df <- tribble(
  ~id, ~x1, ~x2, ~y,
  1, 0.09, 5.02, 1,
  2, 0.1, 5.01, 1,
  3, 0.12, 4.94, 1,
  4, 0.12, 5.12, 1,
  5, 0.12, 5.03, 1,
  6, 0.12, 4.94, 2,
  7, 0.1, 5.13, 2,
  8, 0.1, 4.87, 1,
  9, 0.1, 5.13, 2,
  10, 0.11, 4.94, 3,
  11, 0.11, 4.93, 3,
  12, 0.09, 5.02, 3,
  13, 0.1, 5.01, 3,
  14, 0.09, 4.94, 3,

```

Table 6.3: -

	\$x\_1\$	\$x\_2\$	(\$y\$)
1	0.09	5.02	1
2	0.10	5.01	1
3	0.12	4.94	1
4	0.12	5.12	1
5	0.12	5.03	1
6	0.12	4.94	2
7	0.10	5.13	2
8	0.10	4.87	1
9	0.10	5.13	2
10	0.11	4.94	3
11	0.11	4.93	3
12	0.09	5.02	3
13	0.10	5.01	3
14	0.09	4.94	3
15	0.10	5.12	2
16	0.12	4.93	2
17	0.10	5.00	1
18	0.09	5.01	3

```

15, 0.1, 5.12, 2,
16, 0.12, 4.93, 2,
17, 0.1, 5, 1,
18, 0.09, 5.01, 3
) %>%
  mutate(y = factor(y, levels = c(1, 2, 3)))

knitr::kable(train_df, booktabs = TRUE,
             align = c('r', 'r', 'r', 'r'),
             col.names = c(' ', '$x_1$', '$x_2$', '($y$)'),
             caption = ' - ')

```

Table 6.3       $x_1, x_2$        $(y = 1, 2, 3)$  ,      nnet      multinom

```

multinom_fit <- nnet::multinom(
  y ~ x1 + x2,
  data = train_df,
  maxit = 1000)

## # weights: 12 (6 variable)

```

Table 6.4: - Logistic Regression

y.level	term	estimate	std.error	statistic	p.value
2	(Intercept)	-53.924661	45.402992	-1.1876896	0.2349557
2	x1	31.545749	62.162158	0.5074751	0.6118215
2	x2	9.992311	8.479305	1.1784352	0.2386231
3	(Intercept)	64.191145	57.282533	1.1206059	0.2624556
3	x1	-101.817613	65.516143	-1.5540844	0.1201643
3	x2	-10.810865	10.915881	-0.9903795	0.3219887

```

## initial value 19.775021
## iter 10 value 17.825831
## iter 20 value 16.489924
## iter 30 value 16.116751
## iter 40 value 16.044223
## iter 50 value 16.025259
## iter 60 value 16.024629
## iter 70 value 16.016395
## final value 16.015185
## converged

knitr:::kable(
  multinom_fit %>%
    broom::tidy(exponentiate = FALSE),
  booktabs = TRUE,
  caption = ' - Logistic Regression '
#  caption = 'Table \\\@ref(tab:nominal-logistic-reg-train-data) Logistic Regression '
)

multinom      y   1, 2, 3      1  (reference category) .

```

### 6.3.2

- ,
- $N$   $\{(\mathbf{x}_i, y_i)\}_{i=1,\dots,N}$ .
- $\mathbf{x}_i \in \mathbb{R}^p$ :  $p$  ( $\mathbf{x}_i = [x_{i1} x_{i2} \cdots x_{ip}]^\top$ )
  - $J$ :
  - $y_i$ :  $i \in \{1, 2, \dots, J\}$
- $i$   $\pi_{ij}$ .

$$\pi_{ij} = P(y_i = j | \mathbf{x}_i), j = 1, \dots, J$$

,  $i$

$$\sum_{j=1}^J \pi_{ij} = 1$$

(reference category    baseline category)

( , 2012)     $J$  ).

$$\log \left( \frac{\pi_{ij}}{\pi_{i1}} \right) = \beta_{0,j} + \beta_j^\top \mathbf{x}_i, j = 2, \dots, J$$

$\pi_{ij}$  , (Czepiel, 2002).

$$\pi_{ij} = \frac{\exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i)}{1 + \sum_{j=2}^J \exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i)}, j = 2, \dots, J$$

$$\pi_{i1} = \frac{1}{1 + \sum_{j=2}^J \exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i)}$$

, ,

$$v_{ij} = \begin{cases} 1 & \text{if } y_i = j \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} L &= \prod_{i=1}^n \prod_{j=1}^J (\pi_{ij})^{v_{ij}} \\ &= \prod_{i=1}^n \pi_{i1}^{1 - \sum_{j=2}^J v_{ij}} \prod_{j=2}^J (\pi_{ij})^{v_{ij}} \\ &= \prod_{i=1}^n \frac{\pi_{i1}}{\pi_{i1}^{\sum_{j=2}^J v_{ij}}} \prod_{j=2}^J (\pi_{ij})^{v_{ij}} \\ &= \prod_{i=1}^n \frac{\pi_{i1}}{\prod_{j=2}^J \pi_{i1}^{v_{ij}}} \prod_{j=2}^J (\pi_{ij})^{v_{ij}} \\ &= \prod_{i=1}^n \pi_{i1} \prod_{j=2}^J \left( \frac{\pi_{ij}}{\pi_{i1}} \right)^{v_{ij}} \end{aligned}$$

, (6.13)

$$\begin{aligned}
L &= \prod_{i=1}^n \frac{1}{1 + \sum_{j=2}^J \exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i)} \prod_{j=2}^J (\exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i))^{v_{ij}} \\
&= \prod_{i=1}^n \left( 1 + \sum_{j=2}^J \exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i) \right)^{-1} \prod_{j=2}^J (\exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i))^{v_{ij}} \\
\log L &= \sum_{i=1}^n \left( -\log \left( 1 + \sum_{j=2}^J \exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i) \right) + \sum_{j=2}^J v_{ij} (\beta_{0,j} + \beta_j^\top \mathbf{x}_i) \right) \tag{6.14}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \log L}{\partial \beta_{0,j}} &= \sum_{i=1}^n v_{ij} - \frac{\exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i)}{1 + \sum_{j=2}^J \exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i)} \\
\frac{\partial \log L}{\partial \beta_{k,j}} &= \sum_{i=1}^n v_{ij} x_{ik} - \frac{x_{ik} \exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i)}{1 + \sum_{j=2}^J \exp(\beta_{0,j} + \beta_j^\top \mathbf{x}_i)}, \quad k = 1, \dots, p \tag{6.15}
\end{aligned}$$

, (6.15)  $(J-1) \times (p+1)$  0 . closed form  
solution , . Newton-Raphson method Czepiel  
(2002) .

Table 6.3

```

multinom_fit <- nnet::multinom(
  y ~ x1 + x2,
  data = train_df,
  maxit = 1000)

## # weights: 12 (6 variable)
## initial value 19.775021
## iter 10 value 17.825831
## iter 20 value 16.489924
## iter 30 value 16.116751
## iter 40 value 16.044223
## iter 50 value 16.025259
## iter 60 value 16.024629
## iter 70 value 16.016395
## final value 16.015185
## converged

```

Table 6.5:

	$y_i$	$\pi_{i1}$	$\pi_{i2}$	$\pi_{i3}$	$\hat{y}_i$
1	1	0.283	0.112	0.604	3
2	1	0.425	0.209	0.365	1
3	1	0.589	0.271	0.141	1
4	1	0.262	0.729	0.009	2
5	1	0.450	0.509	0.041	2
6	2	0.589	0.271	0.141	1
7	2	0.349	0.570	0.082	2
8	1	0.199	0.024	0.777	3
9	2	0.349	0.570	0.082	2
10	3	0.501	0.168	0.331	1
11	3	0.490	0.149	0.361	1
12	3	0.283	0.112	0.604	3
13	3	0.425	0.209	0.365	1
14	3	0.160	0.029	0.811	3
15	2	0.365	0.540	0.095	2
16	2	0.595	0.247	0.158	1
17	1	0.416	0.186	0.398	1
18	3	0.268	0.096	0.636	3

```

predict_df <- predict(multinom_fit, train_df, type = "probs") %>%
  as_data_frame() %>%
  `colnames<-`((c("p1", "p2", "p3")) %>%
    mutate(pred_class = predict(multinom_fit, train_df, type = "class"))

## Warning: `as_data_frame()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.

bind_cols(train_df, predict_df) %>%
  select(-x1, -x2) %>%
  knitr:::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r', 'r', 'r'),
    col.names = c(' ', ' '),
    '$\\pi_{i1}$',
    '$\\pi_{i2}$',
    '$\\pi_{i3}$',
    '$\\hat{y}_i$'),
    caption = '',
    digits = 3
  )
}

nnet      glmnet, mlogit, VGAM   R

```

```

VGAM::vglm(y ~ x1 + x2,
            data = train_df,
            family = VGAM::multinomial)

## 
## Call:
## VGAM::vglm(formula = y ~ x1 + x2, family = VGAM::multinomial,
##           data = train_df)
## 
## 
## Coefficients:
## (Intercept):1 (Intercept):2          x1:1          x1:2          x2:1
##      -64.56378     -118.15274     102.65063    133.29235     10.86829
##          x2:2
##      20.81307
## 
## Degrees of Freedom: 36 Total; 30 Residual
## Residual deviance: 32.03007
## Log-likelihood: -16.01503
## 
## This is a multinomial logit model with 3 levels

```

## 6.4

3

,

.

### 6.4.1 R

```

train_df <- tribble(
  ~N, ~L, ~y,
  25, 5, 3,
  25, 10, 3,
  25, 20, 2,
  25, 30, 1,
  32, 5, 3,
  32, 10, 3,
  32, 20, 2,
  32, 30, 1,
  42, 5, 1,
  42, 10, 3,
  42, 20, 1,
  42, 30, 1
) %>%
  mutate(y = as.ordered(y))

```

Table 6.6:

(N)	(L)	(\$y\$)
25	5	3
25	10	3
25	20	2
25	30	1
32	5	3
32	10	3
32	20	2
32	30	1
42	5	1
42	10	3
42	20	1
42	30	1

```
knitr::kable(train_df, booktabs = TRUE,
             align = c('r', 'r', 'r'),
             col.names = c(' (N)', ' (L)', ' ($y$)'), 
             caption = '')
```

Table 6.6 . . . (circuit noise: N) (loudness  
loss: L) . . . (Cavanaugh et al., 1976) 3 . .

$$y = \begin{cases} 1 & \text{good} \\ 2 & \text{fair} \\ 3 & \text{poor} \end{cases}$$

. . . (cumulative logit model) MASS polr . .

```
MASS::polr(y ~ N + L, data = train_df) %>%
  broom::tidy()
```

```
##  
## Re-fitting to get Hessian  
  
## # A tibble: 4 x 5  
##   term estimate std.error statistic coef.type  
##   <chr>    <dbl>     <dbl>     <dbl> <chr>  
## 1 N        -0.224     0.146     -1.53 coefficient  
## 2 L        -0.300     0.137     -2.19 coefficient  
## 3 1|2      -13.0      6.46     -2.02 scale  
## 4 2|3      -11.4      6.17     -1.85 scale
```

```
(adjacent-categories logit model)      VGAM      vglm      .
VGAM::vglm(y ~ N + L,
            data = train_df,
            family = VGAM::acat(reverse = TRUE))

##
## Call:
## VGAM::vglm(formula = y ~ N + L, family = VGAM::acat(reverse = TRUE),
##           data = train_df)
##
##
## Coefficients:
## (Intercept):1 (Intercept):2          N:1          N:2          L:1
## -12.94227208   -6.10597713    0.31431599    0.04643039   0.17500408
##           L:2
##           0.29578067
##
## Degrees of Freedom: 24 Total; 18 Residual
## Residual deviance: 11.67769
## Log-likelihood: -5.838847
##
## This is an adjacent categories model with 3 levels
```

### 6.4.2

$$i \quad j \quad \kappa_{ij} \quad .$$

$$\kappa_{ij} = P(y_i \leq j | \mathbf{x}_i), j = 1, \dots, J$$

$$\log \left( \frac{\kappa_{ij}}{1 - \kappa_{ij}} \right) = \beta_{0,j} + \beta^\top \mathbf{x}_i, j = 1, \dots, J-1 \quad (6.16)$$

$$(6.16) \quad \beta \quad (\text{intercept}) \quad \beta_{0,j} \quad (\text{proportional odds model}) \quad .$$

$$6.3.2 \quad . \quad i \quad j \quad .$$

$$\begin{aligned}\pi_{ij} &= \kappa_{ij} - \kappa_{i,j-1} \\ &= \frac{\exp(\beta_{0,j} + \beta^\top \mathbf{x}_i)}{1 + \exp(\beta_{0,j} + \beta^\top \mathbf{x}_i)} - \frac{\exp(\beta_{0,j-1} + \beta^\top \mathbf{x}_i)}{1 + \exp(\beta_{0,j-1} + \beta^\top \mathbf{x}_i)}, j = 2, \dots, J-1\end{aligned}$$

$$\begin{aligned}\pi_{i1} &= \kappa_{i1} \\ &= \frac{\exp(\beta_{0,1} + \beta^\top \mathbf{x}_i)}{1 + \exp(\beta_{0,1} + \beta^\top \mathbf{x}_i)}\end{aligned}$$

$$\begin{aligned}\pi_{iJ} &= 1 - \kappa_{i,J-1} \\ &= 1 - \frac{\exp(\beta_{0,J-1} + \beta^\top \mathbf{x}_i)}{1 + \exp(\beta_{0,J-1} + \beta^\top \mathbf{x}_i)}\end{aligned}$$

$$\sum_{i=1}^n \sum_{j=1}^J y_i \log \pi_{ij}, \quad \text{concave} \quad (\text{Pratt, 1981}), \quad 0$$

```
polr_fit <- MASS::polr(y ~ N + L, data = train_df)
print(polr_fit)
```

```
## Call:
## MASS::polr(formula = y ~ N + L, data = train_df)
##
## Coefficients:
##             N          L
## -0.2236292 -0.2998833
##
## Intercepts:
##      1|2      2|3
## -13.03527 -11.39902
##
## Residual Deviance: 12.8825
## AIC: 20.8825
```

polr ( , 2012) , polr .

$$\log \left( \frac{\kappa_{ij}}{1 - \kappa_{ij}} \right) = \beta_{0,j} - \beta^\top \mathbf{x}_i, j = 1, \dots, J-1$$

Table 6.7:

(N)	(L)	$\$y\_i\$$	$\$\backslash pi\_{i1}\$$	$\$\backslash pi\_{i2}\$$	$\$\backslash pi\_{i3}\$$	$\$\backslash hat{y}\_i\$$
25	5	3	0.0026	0.0107	0.9867	3
25	10	3	0.0116	0.0452	0.9432	3
25	20	2	0.1905	0.3567	0.4528	3
25	30	1	0.8252	0.1352	0.0396	1
32	5	3	0.0124	0.0481	0.9395	3
32	10	3	0.0531	0.1706	0.7763	3
32	20	2	0.5296	0.3230	0.1474	1
32	30	1	0.9576	0.0338	0.0085	1
42	5	1	0.1049	0.2709	0.6241	3
42	10	3	0.3443	0.3852	0.2705	2
42	20	1	0.9133	0.0685	0.0181	1
42	30	1	0.9953	0.0038	0.0009	1

polr                     $\beta_{0,1} = -13.0352721, \beta_{0,2} = -11.3990207$  ,     N, L  
-0.2236292, -0.2998833 .

(6.4.2)            i        j        Table 6.7        . R        predict

```

predict_df <- predict(polr_fit, train_df, type = "probs") %>%
  as_data_frame() %>%
  `colnames<-`c("p1", "p2", "p3")) %>%
  mutate(pred_class = predict(polr_fit, train_df, type = "class"))

bind_cols(train_df, predict_df) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r', 'r', 'r', 'r'),
    col.names = c(' (N)', ' (L)', ' $y_i$',
                 '$\\pi_{i1}$', '$\\pi_{i2}$', '$\\pi_{i3}$', '$\\hat{y}_i$'),
    caption = '',
    #   caption = 'Table \\@ref(tab:ordinal-logistic-reg-train-data)', digits = 4
  )

```

### 6.4.3

$$\log \left( \frac{\pi_{ij}}{\pi_{i,j+1}} \right) = \beta_{0,j} + \beta^\top \mathbf{x}_i, j = 1, \dots, J-1 \quad (6.17)$$

,  $\pi_{ij}$

$$\begin{aligned} \pi_{ij} &= \exp(\beta_{0,j} + \beta^\top \mathbf{x}_i) \pi_{i,j+1} \\ &= \pi_{iJ} \exp \left( \sum_{k=j}^{J-1} \beta_{0,k} + (J-j)\beta^\top \mathbf{x}_i \right), j = 1, \dots, J-1 \\ \sum_{j=1}^J \pi_{ij} &= 1 \end{aligned}$$

$$\begin{aligned} \pi_{ij} &= \frac{\exp \left( \sum_{l=j}^{J-1} \beta_{0,l} + (J-j)\beta^\top \mathbf{x}_i \right)}{1 + \sum_{k=1}^{J-1} \exp \left( \sum_{l=k}^{J-1} \beta_{0,l} + (J-k)\beta^\top \mathbf{x}_i \right)}, j = 1, \dots, J-1 \\ \pi_{iJ} &= \frac{1}{1 + \sum_{k=1}^{J-1} \exp \left( \sum_{l=k}^{J-1} \beta_{0,l} + (J-k)\beta^\top \mathbf{x}_i \right)} \end{aligned} \quad (6.18)$$

R VGAM vglm family VGAM acat  
acat parallel TRUE (6.17) (proportional odds model)

```
vglm_fit <- VGAM::vglm(  

  y ~ N + L,  

  data = train_df,  

  family = VGAM::acat(reverse = TRUE, parallel = TRUE)  

)  
  
print(coef(vglm_fit))  
  
## (Intercept):1 (Intercept):2 N L  
## -9.0658976 -8.9018134 0.1725867 0.2082167  
(6.18) i j . VGAM predictvglm  
predict_df <- VGAM::predictvglm(vglm_fit, train_df, "response") %>%  
  as_data_frame() %>%  
  `colnames<-`(`c("p1", "p2", "p3")`) %>%  
  mutate(pred_class = ordered(apply(., 1, function(x) which.max(x))))
```

Table 6.8:

(N)	(L)	\$y_i\$	\$\pi_{i1}\$	\$\pi_{i2}\$	\$\pi_{i3}\$	\$\hat{y}_i\$
25	5	3	0.0007	0.0280	0.9713	3
25	10	3	0.0052	0.0751	0.9197	3
25	20	2	0.1804	0.3244	0.4952	3
25	30	1	0.7894	0.1770	0.0337	1
32	5	3	0.0072	0.0874	0.9054	3
32	10	3	0.0474	0.2045	0.7480	3
32	20	2	0.5611	0.3015	0.1375	1
32	30	1	0.9339	0.0626	0.0036	1
42	5	1	0.1393	0.3026	0.5581	3
42	10	3	0.4411	0.3385	0.2204	1
42	20	1	0.9063	0.0867	0.0070	1
42	30	1	0.9881	0.0118	0.0001	1

```
bind_cols(train_df, predict_df) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r', 'r', 'r', 'r'),
    col.names = c(' (N)', ' (L)', ' $y_i$', '$\pi_{i1}$', '$\pi_{i2}$', '$\pi_{i3}$', '$\hat{y}_i$'),
    caption = '',
    #   caption = 'Table \@ref(tab:ordinal-logistic-reg-train-data)',
    digits = 4
  )
```

(proportional odds model)

$$\log \left( \frac{\pi_{ij}}{\pi_{i,j+1}} \right) = \beta_{0,j} + \beta_j^\top \mathbf{x}_i, \quad j = 1, \dots, J-1 \quad (6.19)$$

```
acat parallel FALSE .
vglm_fit <- VGAM::vglm(
  y ~ N + L,
  data = train_df,
  family = VGAM::acat(reverse = TRUE, parallel = FALSE)
)
print(coef(vglm_fit))
## (Intercept):1 (Intercept):2           N:1           N:2           L:1
```

Table 6.9: ( )

(N)	(L)	$\$y_i\$$	$\$\\pi_{i1}\$$	$\$\\pi_{i2}\$$	$\$\\pi_{i3}\$$	$\$\\hat{y}_i\$$
25	5	3	0.0004	0.0303	0.9693	3
25	10	3	0.0043	0.1200	0.8757	3
25	20	2	0.1295	0.6313	0.2392	2
25	30	1	0.5365	0.4546	0.0089	1
32	5	3	0.0055	0.0412	0.9533	3
32	10	3	0.0488	0.1517	0.7995	3
32	20	2	0.5924	0.3200	0.0876	1
32	30	1	0.9131	0.0857	0.0012	1
42	5	1	0.1667	0.0536	0.7797	3
42	10	3	0.6335	0.0850	0.2815	1
42	20	1	0.9734	0.0227	0.0039	1
42	30	1	0.9959	0.0040	0.0000	1

```

## -12.94227208 -6.10597713 0.31431599 0.04643039 0.17500408
## L:2
## 0.29578067
predict_df <- VGAM::predictvglm(vglm_fit, train_df, "response") %>%
  as_data_frame() %>%
  `colnames<-`c("p1", "p2", "p3")) %>%
  mutate(pred_class = ordered(apply(., 1, function(x) which.max(x))))


bind_cols(train_df, predict_df) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r', 'r', 'r', 'r'),
    col.names = c(' (N)', ' (L)', ' $y_i$', '$\\pi_{i1}\$', '$\\pi_{i2}\$', '$\\pi_{i3}\$', '$\\hat{y}_i$'),
    caption = ' ( )',
    #   caption = 'Table \\@ref(tab:ordinal-logistic-reg-train-data)' ( )
    digits = 4
  )

```

# Chapter 7

## 7.1

(discriminant analysis)  
(Fisher)

## 7.2 R

R

package	version
tidyverse	1.3.1
MASS	7.3-54
mvtnorm	1.1-2

## 7.3

### 7.3.1 R

```
train_df <- tibble(  
  id = c(1:9),  
  x1 = c(5, 4, 7, 8, 3, 2, 6, 9, 5),  
  x2 = c(7, 3, 8, 6, 6, 5, 6, 6, 4),  
  class = factor(c(1, 2, 2, 2, 1, 1, 1, 2, 2), levels = c(1, 2))  
)  
  
knitr::kable(train_df, booktabs = TRUE,  
             align = c('r', 'r', 'r', 'r'),  
             col.names = c(' ', '$x_1$', '$x_2$', ' ')),
```

Table 7.1:

	\$x\_1\$	\$x\_2\$	
1	5	7	1
2	4	3	2
3	7	8	2
4	8	6	2
5	3	6	1
6	2	5	1
7	6	6	1
8	9	6	2
9	5	4	2

```

caption = '          ')
Table 7.1      x1, x2      class      9      train_df  data frame  .
fisher_da <- MASS::lda(class ~ x1 + x2, train_df)
print(fisher_da)

## Call:
## lda(class ~ x1 + x2, data = train_df)
##
## Prior probabilities of groups:
##           1           2
## 0.4444444 0.5555556
##
## Group means:
##      x1  x2
## 1 4.0 6.0
## 2 6.6 5.4
##
## Coefficients of linear discriminants:
##            LD1
## x1  0.6850490
## x2 -0.7003859

```

### 7.3.2

$\mathbf{x} \in \mathbb{R}^p$      $y \in \{1, 2\}$     .     $\mathbf{x}$     -    (variance-covariance matrix) .

$$\begin{aligned}\mu_1 &= E(\mathbf{x}|y=1) \\ \mu_2 &= E(\mathbf{x}|y=2) \\ \Sigma &= Var(\mathbf{x}|y=1) = Var(\mathbf{x}|y=2)\end{aligned}$$

$z$  (Fisher's discriminant function) .

$$z = \mathbf{w}^\top \mathbf{x} \quad (7.1)$$

$$\mathbf{w} \in \mathbb{R}^p$$

$$z \quad .$$

$$\operatorname{argmin}_{\mathbf{w}} \frac{\mathbf{w}^\top (\mu_1 - \mu_2)}{\mathbf{w}^\top \Sigma \mathbf{w}} \quad (7.2)$$

(7.2)

$$\mathbf{w} \propto \Sigma^{-1}(\mu_1 - \mu_2)$$

$$, \quad 1 \quad .$$

$$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2) \quad (7.3)$$

$$, \quad \mu_1, \mu_2, \Sigma \quad (7.3) \quad . \quad ( , 2012)$$

Table 7.1

$$\hat{\mu}_1, \hat{\mu}_2$$

```
mu_hat <- train_df %>%
  group_by(class) %>%
  summarize(x1 = mean(x1),
            x2 = mean(x2)) %>%
  arrange(class)

print(mu_hat)

## # A tibble: 2 x 3
##   class     x1     x2
##   <fct> <dbl> <dbl>
## 1 1       4      6
## 2 2      6.6    5.4
-       S1, S2       S_within_group       1       S1,       2
-       S2       .
```

```

S_within_group <- lapply(
  unique(train_df$class) %>% sort(), function(x) {
    train_df %>% filter(class == x) %>% select(x1, x2) %>% var()
  }
)

print(S_within_group)

## [[1]]
##          x1          x2
## x1 3.333333 1.0000000
## x2 1.000000 0.6666667
##
## [[2]]
##          x1      x2
## x1 4.30 2.95
## x2 2.95 3.80
-
-
-

```

$$\hat{\Sigma} = \mathbf{S}_p = \frac{(n_1 - 1)\mathbf{S}_1 + (n_2 - 1)\mathbf{S}_2}{n_1 + n_2 - 2}$$

```

n1, n2      1, 2      .      R      n      -      S
-      pooled_variance ,      -      Sigma_hat .
pooled_variance <- function(n, S) {
  lapply(1:length(n), function(i) (n[i] - 1)*S[[i]]) %>%
    Reduce(`+`, .) %>%
    `/~(sum(n) - length(n))
}

n_obs <- train_df %>%
  group_by(class) %>%
  count() %>%
  ungroup() %>%
  mutate(pi = n / sum(n)) %>%
  arrange(class)

Sigma_hat <- pooled_variance(n_obs$n, S_within_group)

print(Sigma_hat)

##          x1          x2
## x1 3.885714 2.114286
## x2 2.114286 2.457143

```

$\hat{\mathbf{w}}$ 

$$\hat{\mathbf{w}} = \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_2)$$

```
w_hat <- solve(Sigma_hat) %*% t(mu_hat[1, c('x1', 'x2)]) - mu_hat[2, c('x1', 'x2')])

print(w_hat)

##          [,1]
## x1 -1.508039
## x2  1.541801
```

### 7.3.3

$$\bar{z} = \frac{1}{N} \sum_i^N \hat{\mathbf{w}}^\top \mathbf{x}_i$$

```
z_mean <- t(w_hat) %*% (train_df %>% select(x1, x2) %>% colMeans()) %>% drop()

print(z_mean)

## [1] 0.526438

,
•  $\hat{\mathbf{w}}^\top \mathbf{x} \geq \bar{z}$ ,  $\mathbf{x} = 1$ 
•  $\hat{\mathbf{w}}^\top \mathbf{x} < \bar{z}$ ,  $\mathbf{x} = 2$ 

train_prediction_df <- train_df %>%
  mutate(
    z = w_hat[1]*x1 + w_hat[2]*x2,
    predicted_class = factor(if_else(z >= z_mean, 1, 2), levels = c(1, 2))
  )

knitr::kable(train_prediction_df, booktabs = TRUE,
             align = c('r', 'r', 'r', 'r', 'r', 'r'),
             col.names = c(' ', '$x_1$', '$x_2$', ' ', '$z$', ' '),
             caption = ' ')
```

3, 7

### 7.3.4 R

```
MASS     lda      w      , lda      1)
, 2)
```

Table 7.2:

	\$x\_1\$	\$x\_2\$		\$z\$	
1	5	7	1	3.2524116	1
2	4	3	2	-1.4067524	2
3	7	8	2	1.7781350	1
4	8	6	2	-2.8135048	2
5	3	6	1	4.7266881	1
6	2	5	1	4.6929260	1
7	6	6	1	0.2025723	2
8	9	6	2	-4.3215434	2
9	5	4	2	-1.3729904	2

$$\begin{aligned} \max \quad & \mathbf{w}^\top (\mu_1 - \mu_2) \\ \text{s.t. } & \mathbf{w}^\top \Sigma \mathbf{w} = 1 \end{aligned}$$

```

lda
fisher_da <- MASS::lda(class ~ x1 + x2, train_df)

w_hat_lda <- fisher_da$scaling
print(w_hat_lda)

##          LD1
## x1  0.6850490
## x2 -0.7003859

z_mean_lda <- t(fisher_da$scaling) %*% (train_df %>% select(x1, x2) %>% colMeans()) %>%
print(z_mean_lda)

##          LD1
## -0.2391423

scale_adjust <- t(w_hat) %*% Sigma_hat %*% w_hat %>% drop() %>% sqrt()
sign_adjust <- -1

w_hat <- w_hat_lda * scale_adjust * sign_adjust
print(w_hat)

##          LD1
## x1 -1.508039
## x2  1.541801

```

```

z_mean <- z_mean_lda * scale_adjust * sign_adjust
print(z_mean)

##      LD1
## 0.526438

lda

 $\hat{\mathbf{w}}^\top \mathbf{x} - \bar{z}$ 

predict(fisher_da, train_df)$x

##      LD1
## 1 -1.2383140
## 2  0.8781805
## 3 -0.5686020
## 4  1.5172187
## 5 -1.9080261
## 6 -1.8926892
## 7  0.1471208
## 8  2.2022677
## 9  0.8628436

0      1, 0      2 . 

train_df %>%
  mutate(
    centered_z = predict(fisher_da, .)$x,
    predicted_class = factor(if_else(centered_z <= 0, 1, 2), levels = c(1, 2))
  ) %>%
  knitr::kable(booktabs = TRUE,
               align = c('r', 'r', 'r', 'r', 'r', 'r'),
               col.names = c(' ', '$x_1$', '$x_2$',
                            ' ', '$z - \bar{z}$', ' `MASS::lda` '),
               caption = ' ')

```

Table 7.3 Table 7.2

**7.4**

- $\pi_k$ :  $k$
- $f_k(\mathbf{x})$ :  $k$

$$\mathbf{x} = \mu_k + \Sigma \cdot \epsilon$$

Table 7.3: - ‘MASS::lda’

	\$x\_1\$	\$x\_2\$		\$z - \bar{z}\$	
1	5	7	1	-1.2383140	1
2	4	3	2	0.8781805	2
3	7	8	2	-0.5686020	1
4	8	6	2	1.5172187	2
5	3	6	1	-1.9080261	1
6	2	5	1	-1.8926892	1
7	6	6	1	0.1471208	2
8	9	6	2	2.2022677	2
9	5	4	2	0.8628436	2

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left\{-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right\} \quad (7.4)$$

$$(k = 1, 2), \dots$$

#### 7.4.1 R

```
Table 7.1           R           , prior      π₁   π₂
1, 2               .           .           .           .
lda_fit <- MASS::lda(class ~ x1 + x2, train_df)
print(lda_fit)

## Call:
## lda(class ~ x1 + x2, data = train_df)
##
## Prior probabilities of groups:
##          1          2
## 0.44444444 0.55555556
##
## Group means:
##    x1  x2
## 1 4.0 6.0
## 2 6.6 5.4
##
## Coefficients of linear discriminants:
##          LD1
## x1  0.6850490
## x2 -0.7003859
```

### 7.4.2

,

 $\mathbf{x}$ 

.

$$f(\mathbf{x}) = \pi_1 f_1(\mathbf{x}) + \pi_2 f_2(\mathbf{x})$$

$$\text{(Bayes's theorem)} \quad \mathbf{x} \quad k \quad \text{(posterior)}$$

$$P(y = k | \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{f(\mathbf{x})} \quad (7.5)$$

,

$$\hat{y} = \begin{cases} 1, & \text{if } P(y = 1 | \mathbf{x}) \geq P(y = 2 | \mathbf{x}) \\ 2, & \text{otherwise} \end{cases} \quad (7.6)$$

(7.4)

( , 2012) .

$$\hat{y} = \begin{cases} 1, & \text{if } \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \geq \frac{\pi_2}{\pi_1} \\ 2, & \text{otherwise} \end{cases}$$

$$\hat{y} = \begin{cases} 1, & \text{if } \mu_1^\top \Sigma^{-1} \mathbf{x} - \frac{1}{2} \mu_1^\top \Sigma^{-1} \mu_1 + \ln \pi_1 \geq \mu_2^\top \Sigma^{-1} \mathbf{x} - \frac{1}{2} \mu_2^\top \Sigma^{-1} \mu_2 + \ln \pi_2 \\ 2, & \text{otherwise} \end{cases}$$

,

$$u_k(\mathbf{x}) = \mu_k^\top \Sigma^{-1} \mathbf{x} - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k + \ln \pi_k \quad (7.7)$$

,

$$\hat{y} = \begin{cases} 1, & \text{if } u_1(\mathbf{x}) \geq u_2(\mathbf{x}) \\ 2, & \text{otherwise} \end{cases} \quad (7.8)$$

Table 7.1

```
discriminant_func <- function(X, mu, Sigma, pi) {
  Sigma_inv <- solve(Sigma)
  (t(mu) %*% Sigma_inv %*% X %>% drop()) -
    0.5 * (t(mu) %*% Sigma_inv %*% mu %>% drop()) +
    log(pi)
}
```

Table 7.4: LDA :

	\$x\_1\$	\$x\_2\$		\$u\_1(\mathbf{x})\$	\$u\_2(\mathbf{x})\$	
1	5	7	1	9.2051470	6.971538	1
2	4	3	2	-1.9363321	0.489223	2
3	7	8	2	11.0057900	10.246458	1
4	8	6	2	4.5909990	8.423307	2
5	3	6	1	7.4045039	3.696619	1
6	2	5	1	5.0411598	1.367036	1
7	6	6	1	5.7164010	6.532631	2
8	9	6	2	4.0282981	9.368644	2
9	5	4	2	0.4270119	2.818805	2

```

lda_discriminant_result_df <- train_df %>%
  mutate(
    u1 = discriminant_func(
      .[c("x1", "x2")] %>% t(),
      mu_hat[1, c("x1", "x2")] %>% unlist(),
      Sigma_hat,
      n_obs$pi[1]
    ),
    u2 = discriminant_func(
      .[c("x1", "x2")] %>% t(),
      mu_hat[2, c("x1", "x2")] %>% unlist(),
      Sigma_hat,
      n_obs$pi[2]
    )
  ) %>%
  mutate(
    predicted_class = factor(if_else(u1 >= u2, 1, 2), levels = c(1, 2))
  )

knitr::kable(
  lda_discriminant_result_df,
  booktabs = TRUE,
  align = rep('r', dim(lda_discriminant_result_df)[2]),
  col.names = c(' ', '$x\_1$', '$x\_2$',
               ' ', '$u\_1(\mathbf{x})$', '$u\_2(\mathbf{x})$',
               ' '),
  caption = 'LDA : ')

```

(7.5)

(7.6)

.

```

lda_posterior_result_df <- train_df %>%
  mutate(
    f1 = mvtnorm::dmvnorm(
      .[c("x1", "x2")],
      mu_hat[1, c("x1", "x2")] %>% unlist(),
      Sigma_hat),
    f2 = mvtnorm::dmvnorm(
      .[c("x1", "x2")],
      mu_hat[2, c("x1", "x2")] %>% unlist(),
      Sigma_hat),
    f = n_obs$pi[1] * f1 + n_obs$pi[2] * f2
  ) %>%
  mutate(
    p1 = n_obs$pi[1] * f1 / f,
    p2 = n_obs$pi[2] * f2 / f
  ) %>%
  mutate(
    predicted_class = factor(if_else(p1 >= p2, 1, 2), levels = c(1, 2))
  ) %>%
  select(
    id, x1, x2, class, p1, p2, predicted_class
  )

knitr::kable(
  lda_posterior_result_df,
  booktabs = TRUE,
  align = rep('r', dim(lda_posterior_result_df)[2]),
  col.names = c(' ', '$x_1$', '$x_2$',
               ' ',
               '$P(y = 1 | \mathbf{x})$',
               '$P(y = 2 | \mathbf{x})$',
               ' '),
  caption = '      LDA      :      ')

```

```

MASS     lda     Table 7.5
lda_fit <- MASS::lda(class ~ x1 + x2, train_df)

train_df %>% bind_cols(
  predict(lda_fit, train_df)$posterior %>%
    `colnames<-` (paste0("p", colnames(.))) %>% as_data_frame()
) %>%
  mutate(
    predicted_class = predict(lda_fit, .)$class
  )

```

Table 7.5: LDA :

	\$x\_1\$	\$x\_2\$		\$P(y = 1   \mathbf{x})\$	\$P(y = 2   \mathbf{x})\$	
1	5	7	1	0.9032273	0.0967727	1
2	4	3	2	0.0812446	0.9187554	2
3	7	8	2	0.6812088	0.3187912	1
4	8	6	2	0.0212004	0.9787996	2
5	3	6	1	0.9760579	0.0239421	1
6	2	5	1	0.9752562	0.0247438	1
7	6	6	1	0.3065644	0.6934356	2
8	9	6	2	0.0047713	0.9952287	2
9	5	4	2	0.0838007	0.9161993	2

```

## # A tibble: 9 x 7
##       id     x1     x2 class      p1      p2 predicted_class
##   <int> <dbl> <dbl> <fct>    <dbl>    <dbl> <fct>
## 1     1     5     7 1        0.903    0.0968 1
## 2     2     4     3 2        0.0812   0.919   2
## 3     3     7     8 2        0.681    0.319   1
## 4     4     8     6 2        0.0212   0.979   2
## 5     5     3     6 1        0.976    0.0239 1
## 6     6     2     5 1        0.975    0.0247 1
## 7     7     6     6 1        0.307    0.693   2
## 8     8     9     6 2        0.00477  0.995   2
## 9     9     5     4 2        0.0838   0.916   2

( , 2012)           ,
lda          prior .           .
pi_1 = pi_2 = 0.5

lda_fit_equal_prior <- MASS::lda(class ~ x1 + x2, train_df, prior = c(1/2, 1/2))

train_df %>% bind_cols(
  predict(lda_fit_equal_prior, train_df)$posterior %>%
    `colnames<-`(paste0("p", colnames(.))) %>% as_data_frame()
) %>%
  mutate(
    predicted_class = predict(lda_fit_equal_prior, .)$class
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = rep('r', dim(lda_posterior_result_df)[2]),
    col.names = c(' ', '$x\_1$', '$x\_2$', 
                 ' ', 
                 '$P(y = 1 | \mathbf{x})$',

```

Table 7.6: LDA : ( $\pi = 0.5$ )

\$x\_1\$	\$x\_2\$	\$P(y = 1   \mathbf{x})\$	\$P(y = 2   \mathbf{x})\$	
1	5	0.9210538	0.0789462	1
2	4	0.0995341	0.9004659	2
3	7	0.7275992	0.2724008	1
4	8	0.0263608	0.9736392	2
5	3	0.9807542	0.0192458	1
6	2	0.9801065	0.0198935	1
7	6	0.3559269	0.6440731	2
8	9	0.0059571	0.9940429	2
9	5	0.1026013	0.8973987	2

```
'$P(y = 2 | \mathbf{x})$',
''),
caption = 'LDA : (\pi = 0.5)')
```

## 7.5

Table 7.5 3, 7

- $C(1|2): 2 \ 1$
- $C(2|1): 1 \ 2$

$$C(1|2)\pi_2 \int_{\mathbf{x} \in R_1} f_2(\mathbf{x}) d\mathbf{x} + C(2|1)\pi_1 \int_{\mathbf{x} \in R_2} f_1(\mathbf{x}) d\mathbf{x} \quad (7.9)$$

$$R_1 \subset \mathbb{R}^p, R_2 = \mathbb{R}^p - R_1 \quad 1, 2$$

$$\hat{y} = \begin{cases} 1, & \text{if } \mathbf{x} \in R_1 \\ 2, & \text{otherwise} \end{cases}$$

$$(7.9) \quad R_1, R_2$$

$$\begin{aligned} R_1 &= \left\{ \mathbf{x} \in \mathbb{R}^p : \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \geq \frac{\pi_2}{\pi_1} \left( \frac{C(1|2)}{C(2|1)} \right) \right\} \\ R_2 &= \left\{ \mathbf{x} \in \mathbb{R}^p : \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} < \frac{\pi_2}{\pi_1} \left( \frac{C(1|2)}{C(2|1)} \right) \right\} \end{aligned}$$

$R_1$ 

$$\begin{aligned}
R_1 &= \left\{ \mathbf{x} \in \mathbb{R}^p : \frac{\pi_1 f_1(\mathbf{x})}{\pi_2 f_2(\mathbf{x})} \geq \frac{C(1|2)}{C(2|1)} \right\} \\
&= \left\{ \mathbf{x} \in \mathbb{R}^p : \frac{\frac{\pi_1 f_1(\mathbf{x})}{\pi_1 f_1(\mathbf{x}) + \pi_2 f_2(\mathbf{x})}}{\frac{\pi_2 f_2(\mathbf{x})}{\pi_1 f_1(\mathbf{x}) + \pi_2 f_2(\mathbf{x})}} \geq \frac{C(1|2)}{C(2|1)} \right\} \\
&= \left\{ \mathbf{x} \in \mathbb{R}^p : \frac{P(y=1|\mathbf{x})}{P(y=2|\mathbf{x})} \geq \frac{C(1|2)}{C(2|1)} \right\} \\
&= \{ \mathbf{x} \in \mathbb{R}^p : C(2|1)P(y=1|\mathbf{x}) \geq C(1|2)P(y=2|\mathbf{x}) \}
\end{aligned}$$

1), 2)

Table 7.5  $C(1|2) = 1, C(2|1) = 5$ 

```

lda_fit <- MASS::lda(class ~ x1 + x2, train_df)

misclassification_cost <- c(5, 1)

lda Unequal_cost_result_df <- train_df %>% bind_cols(
  predict(lda_fit, train_df)$posterior %*% diag(misclassification_cost) %>%
    as_data_frame() %>%
    `names<-` (paste0("s", lda_fit$lev))
) %>%
  mutate(
  predicted_class = factor(if_else(s1 >= s2, 1, 2), levels = c(1, 2))
)

knitr::kable(
  lda Unequal_cost_result_df,
  booktabs = TRUE,
  align = rep('r', dim(lda Unequal_cost_result_df)[2]),
  col.names = c(' ', '$x_1$', '$x_2$',
               ' ', '$C(2 | |, 1) P(y = 1 | \\mathbf{x})$', '$C(1 | |, 2) P(y = 2 | \\mathbf{x})$',
               ' ', '),
  caption = 'LDA')

```

Table 7.7

2	1
, 1	1

Table 7.5

3 ,	1
, .	2

1

2	2
, 1	2

1 2

Table 7.7: LDA

\$x\_1\$	\$x\_2\$	\$C(2 \setminus,   \setminus, 1) P(y = 1   \mathbf{x})\$	\$C(1 \setminus,   \setminus, 2) P(y = 2   \mathbf{x})\$		
1	5	7	1	4.5161363	0.0967727
2	4	3	2	0.4062232	0.9187554
3	7	8	2	3.4060438	0.3187912
4	8	6	2	0.1060019	0.9787996
5	3	6	1	4.8802897	0.0239421
6	2	5	1	4.8762808	0.0247438
7	6	6	1	1.5328222	0.6934356
8	9	6	2	0.0238567	0.9952287
9	5	4	2	0.4190033	0.9161993

## 7.6

,                    **x**                    -

### 7.6.1 R

Table 7.1                    R                    MASS            qda            .

```
qda_fit <- MASS::qda(class ~ x1 + x2, train_df)
print(qda_fit)
```

```
## Call:
## qda(class ~ x1 + x2, data = train_df)
##
## Prior probabilities of groups:
##      1      2
## 0.4444444 0.5555556
##
## Group means:
##   x1 x2
## 1 4.0 6.0
## 2 6.6 5.4
```

### 7.6.2

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} |\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right\} \quad (7.10)$$

$$(7.10) \quad (7.4) \quad - \quad \Sigma_k \quad k$$

$$u_k(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_k| + \ln \pi_k \quad (7.11)$$

```
X = (x_1, x_2, ..., x_N)
qda_discriminant_func <- function(X, mu, Sigma, pi) {
  Sigma_inv_sqrt <- chol(solve(Sigma))
  - 0.5 * rowSums((t(X - mu) %*% t(Sigma_inv_sqrt))^2) - 0.5 * log(det(Sigma)) + log(pi)
}
```

### 7.6.3

$$\hat{y} = \begin{cases} 1, & \text{if } u_1(\mathbf{x}) \geq u_2(\mathbf{x}) \\ 2, & \text{otherwise} \end{cases}$$

Table 7.1

```
qda_discriminant_result_df <- train_df %>% mutate(
  u1 = qda_discriminant_func(
    .[c("x1", "x2")] %>% t(),
    mu_hat[1, c("x1", "x2")] %>% unlist(),
    S_within_group[[1]],
    n_obs$pi[1]
  ),
  u2 = qda_discriminant_func(
    .[c("x1", "x2")] %>% t(),
    mu_hat[2, c("x1", "x2")] %>% unlist(),
    S_within_group[[2]],
    n_obs$pi[2]
  )
) %>%
  mutate(
    predicted_class = factor(if_else(u1 >= u2, 1, 2), levels = c(1, 2))
  )

knitr::kable(
  qda_discriminant_result_df,
  booktabs = TRUE,
  align = rep('r', dim(qda_discriminant_result_df)[2]),
  col.names = c(' ', '$x_1$', '$x_2$',
  
```

Table 7.8: QDA :

	\$x\_1\$	\$x\_2\$		\$u\_1(\mathbf{x})\$	\$u\_2(\mathbf{x})\$	
1	5	7	1	-1.729447	-3.950639	1
2	4	3	2	-13.183993	-2.497284	2
3	7	8	2	-3.911266	-3.145402	2
4	8	6	2	-5.274902	-1.868806	2
5	3	6	1	-1.183993	-5.764060	1
6	2	5	1	-1.729447	-6.202685	1
7	6	6	1	-2.002175	-1.934273	2
8	9	6	2	-7.729447	-2.582391	2
9	5	4	2	-8.274902	-1.927726	2

```

    ', '$u_1(\mathbf{x})$', '$u_2(\mathbf{x})$',
),
caption = 'QDA : ')

```

Table 7.8

```

qda_posterior_result_df <- train_df %>%
  mutate(
    f1 = mvtnorm::dmvnorm(
      .[c("x1", "x2")],
      mu_hat[1, c("x1", "x2")] %>% unlist(),
      S_within_group[[1]]),
    f2 = mvtnorm::dmvnorm(
      .[c("x1", "x2")],
      mu_hat[2, c("x1", "x2")] %>% unlist(),
      S_within_group[[2]]),
    f = n_obs$pi[1] * f1 + n_obs$pi[2] * f2
  ) %>%
  mutate(
    p1 = n_obs$pi[1] * f1 / f,
    p2 = n_obs$pi[2] * f2 / f
  ) %>%
  mutate(
    predicted_class = factor(if_else(p1 >= p2, 1, 2), levels = c(1, 2))
  ) %>%
  select(
    id, x1, x2, class, p1, p2, predicted_class
  )

```

Table 7.9: QDA :

	\$x\_1\$	\$x\_2\$		\$P(y = 1   \mathbf{x})\$	\$P(y = 2   \mathbf{x})\$	
1	5	7	1	0.9021365	0.0978635	1
2	4	3	2	0.0000228	0.9999772	2
3	7	8	2	0.3173746	0.6826254	2
4	8	6	2	0.0321055	0.9678945	2
5	3	6	1	0.9898499	0.0101501	1
6	2	5	1	0.9887184	0.0112816	1
7	6	6	1	0.4830310	0.5169690	2
8	9	6	2	0.0057829	0.9942171	2
9	5	4	2	0.0017486	0.9982514	2

```

knitr::kable(
  qda_posterior_result_df,
  booktabs = TRUE,
  align = rep('r', dim(qda_posterior_result_df)[2]),
  col.names = c(' ', '$x_1$', '$x_2$',
               ' ',
               '$P(y = 1 | \mathbf{x})$',
               '$P(y = 2 | \mathbf{x})$',
               ' '),
  caption = 'QDA : ')

```

```

MASS      predict.qda
train_df %>% bind_cols(
  predict(qda_fit, train_df)$posterior %>%
    `colnames<-`(paste0("p", colnames(.))) %>% as_data_frame()
  ) %>%
  mutate(
    predicted_class = predict(qda_fit, .)$class
  )

## # A tibble: 9 x 7
##       id     x1     x2 class      p1      p2 predicted_class
##   <int> <dbl> <dbl> <fct>    <dbl>    <dbl> <fct>
## 1     1      5     7 1        0.902    0.0979 1
## 2     2      4     3 2        0.0000228 1.00   2
## 3     3      7     8 2        0.317    0.683  2
## 4     4      8     6 2        0.0321   0.968  2
## 5     5      3     6 1        0.990    0.0102 1
## 6     6      2     5 1        0.989    0.0113 1
## 7     7      6     6 1        0.483    0.517  2

```

```
## 8     8     9     6 2     0.00578   0.994 2
## 9     9     5     4 2     0.00175   0.998 2
```

## 7.7

### 7.7.1 R

```
3      (iris)          R      .      50      30      .
iris_train_df <- datasets::iris %>%
  rename(x1 = Sepal.Length,
         x2 = Sepal.Width,
         x3 = Petal.Length,
         x4 = Petal.Width,
         class = Species) %>%
  group_by(class) %>%
  slice(1:30) %>%
  ungroup() %>%
  mutate(id = row_number())

iris_lda_fit <- MASS::lda(class ~ . -id, iris_train_df)

print(iris_lda_fit)

## Call:
## lda(class ~ . - id, data = iris_train_df)
##
## Prior probabilities of groups:
##       setosa versicolor  virginica
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##           x1        x2        x3        x4
## setosa    5.026667 3.450000 1.473333 0.2466667
## versicolor 6.070000 2.790000 4.333333 1.3533333
## virginica  6.583333 2.933333 5.603333 2.0066667
##
## Coefficients of linear discriminants:
##           LD1        LD2
## x1  0.5711419 -1.2397647
## x2  1.8752911  3.0223980
## x3 -1.7361767  0.3159667
## x4 -3.4672646  1.3954748
##
## Proportion of trace:
##      LD1      LD2
```

```
## 0.9929 0.0071
```

### 7.7.2

$K(> 2)$

- $\pi_k: k, k = 1, 2, \dots, K$
- $C(k' | k) \geq 0: k \quad k'$  ( $C(k' | k) = 0$  if  $k' = k$ )
- $f_k(\mathbf{x}): k \quad \mathbf{x}$
- $R_k \subset \mathbb{R}^p: k \quad \mathbf{x}$
- $P(k' | k) = \int_{\mathbf{x} \in R_{k'}} f_k(\mathbf{x}) d\mathbf{x}: k \quad k'$

,

$$\sum_{k=1}^K \pi_k \sum_{k' \neq k} C(k' | k) \int_{\mathbf{x} \in R_{k'}} f_k(\mathbf{x}) d\mathbf{x}$$

$R_1, \dots, R_K$

,

$\mathbf{x}$

$$f(\mathbf{x}) = \sum_{k=1}^K \pi_k f_k(\mathbf{x})$$

,  $\mathbf{x} \quad k$

$$P(y = k | \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{f(\mathbf{x})}$$

$$P(y = k | \mathbf{x}) \propto \pi_k f_k(\mathbf{x})$$

,

$$\hat{y} = \arg \max_k \pi_k f_k(\mathbf{x})$$

,  $f_k(\mathbf{x})$

MASS lda

```
iris_lda_fit <- MASS::lda(class ~ x1 + x2 + x3 + x4, iris_train_df)

iris_lda_result <- iris_train_df %>%
  bind_cols(
    predict(iris_lda_fit, .)$posterior %>%
```

Table 7.10: LDA -

			setosa	versicolor	virginica
51	versicolor	virginica	0	0.3088912	0.6911088

```

    as_data_frame()
) %>%
mutate(
  predicted_class = predict(iris_lda_fit, .)$class
)

print(iris_lda_result)

## # A tibble: 90 x 10
##       x1     x2     x3     x4 class      id setosa versicolor virginica
##   <dbl> <dbl> <dbl> <dbl> <fct> <int> <dbl>      <dbl>      <dbl>
## 1  5.1   3.5   1.4   0.2 setosa     1   1   5.57e-22  6.34e-42
## 2  4.9   3.0   1.4   0.2 setosa     2   1   3.32e-17  5.67e-36
## 3  4.7   3.2   1.3   0.2 setosa     3   1   2.75e-19  2.14e-38
## 4  4.6   3.1   1.5   0.2 setosa     4   1   8.12e-17  5.62e-35
## 5  5.0   3.6   1.4   0.2 setosa     5   1   1.14e-22  1.25e-42
## 6  5.4   3.9   1.7   0.4 setosa     6   1   3.20e-21  4.38e-40
## 7  4.6   3.4   1.4   0.3 setosa     7   1   8.74e-19  4.07e-37
## 8  5.0   3.4   1.5   0.2 setosa     8   1   3.27e-20  1.62e-39
## 9  4.4   2.9   1.4   0.2 setosa     9   1   1.00e+00  2.22e-15 3.42e-33
## 10 4.9   3.1   1.5   0.1 setosa    10   1   9.36e-19  4.85e-38
## # ... with 80 more rows, and 1 more variable: predicted_class <fct>

knitr:::kable(
  iris_lda_result %>%
    select(id, class, predicted_class,
           setosa, versicolor, virginica) %>%
    filter(class != predicted_class),
  booktabs = TRUE,
  align = rep('r', 6),
  col.names = c(' ', ' ', ' ', ' ',
               'setosa', 'versicolor', 'virginica'),
  caption = 'LDA - ')

```

MASS qda .

```

iris_qda_fit <- MASS::qda(class ~ x1 + x2 + x3 + x4, iris_train_df)

iris_qda_result <- iris_train_df %>%

```

Table 7.11: QDA -

		setosa	versicolor	virginica	
51	versicolor	virginica	0	0.4274712	0.5725288

```

bind_cols(
  predict(iris_qda_fit, .)$posterior %>%
    as_data_frame()
  ) %>%
mutate(
  predicted_class = predict(iris_qda_fit, .)$class
)

knitr::kable(
  iris_qda_result %>%
    select(id, class, predicted_class,
           setosa, versicolor, virginica) %>%
    filter(class != predicted_class),
  booktabs = TRUE,
  align = rep('r', 6),
  col.names = c(' ', ' ', ' ', ' ',
               'setosa', 'versicolor', 'virginica'),
  caption = 'QDA - ')

```

# Chapter 8

## 8.1 CART

CART(Classification and Regression Trees) Breiman et al. (1984)  
( ) (binary split) ( ) ( ) .

## 8.2 R package

package	version
tidyverse	1.3.1
rpart	4.1-15
rpart.plot	3.0.9

## 8.3 CART

### 8.3.1 R

```
train_df <- tibble(  
  x1 = c(1,2,2,2,2,3,4,4,4,5),  
  x2 = c(4,6,5,4,3,6,6,5,4,3),  
  class = as.factor(c(1,1,1,2,2,1,1,2,2,2))  
)
```

Table 8.1  $x_1, x_2$   $class$  10  $train\_df$  data frame .

```
library(rpart)  
library(rpart.plot)
```

Table 8.1:

x1	x2	class
1	4	1
2	6	1
2	5	1
2	4	2
2	3	2
3	6	1
4	6	1
4	5	2
4	4	2
5	3	2

```

cart.est <- rpart(
  class ~ x1 + x2
  , data = train_df
  , method = "class"
  , parms = list(split = "gini")
  , control = rpart.control(minsplit = 2
    , minbucket = 1
    , cp = 0
    , xval = 0
    , maxcompete = 0)
)
rpart.plot(cart.est)

```

rpart package , x1 x2 class CART , rpart.plot  
 package Figure 8.1 .

### 8.3.2

- $T$ :
- $A(T)$ :  $T$
- $J$ :
- $N$ :
- $N_j$ :  $j$
- $N(t)$ :  $t$
- $N_j(t)$ :  $t$        $j$
- $p(j, t)$ :       $j$        $t$

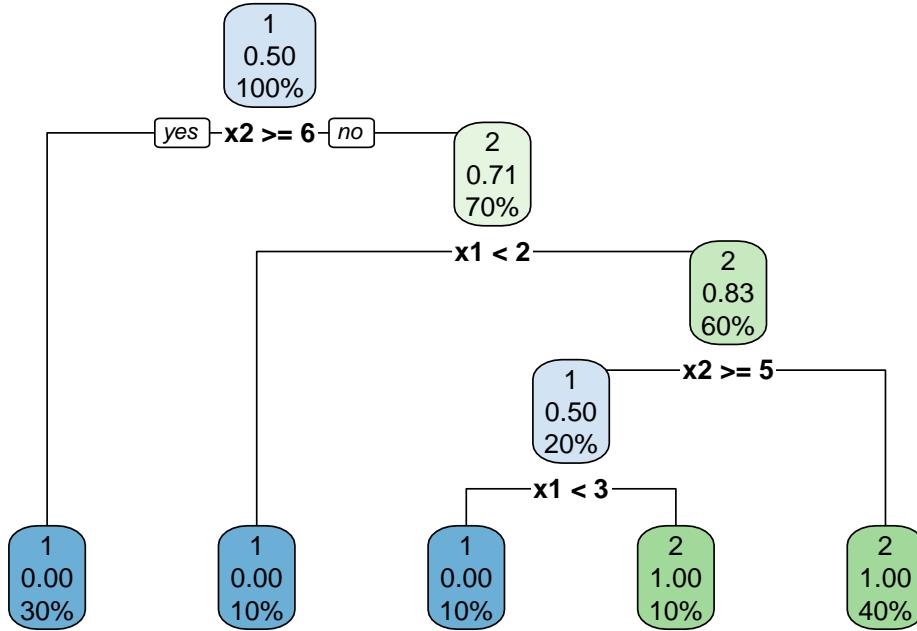


Figure 8.1: CART

- $p(t)$ :  $t$

$$p(t) = \sum_{j=1}^J p(j, t)$$

- $p(j|t)$ :  $t$   $j$

$$p(j|t) = \frac{p(j, t)}{p(t)}, \quad \sum_{j=1}^J p(j|t) = 1$$

,

$$p(j, t) \approx \frac{N_j(t)}{N} \quad (8.1)$$

$$p(t) \approx \frac{N(t)}{N} \quad (8.2)$$

$$p(j|t) \approx \frac{N_j(t)}{N(t)} \quad (8.3)$$

### 8.3.3

#### 8.3.3.1

$$\text{CART} \quad \text{(Gini index)} \quad J \quad p_1, \dots, p_J \quad (\sum_{j=1}^J p_j = 1), \quad (8.4)$$

$$G(p_1, \dots, p_J) = \sum_{j=1}^J p_j(1-p_j) = 1 - \sum_{j=1}^J p_j^2 \quad (8.4)$$

$$t \quad p(1|t), \dots, p(J|t) \quad , \quad t \quad (8.5) \quad .$$

$$\begin{aligned} i(t) &= 1 - \sum_{j=1}^J p(j|t)^2 \\ &\approx 1 - \sum_{j=1}^J \left[ \frac{N_j(t)}{N(t)} \right]^2 \end{aligned} \quad (8.5)$$

### 8.3.3.2

$$T \quad (8.6) \quad .$$

$$I(T) = \sum_{t \in A(T)} i(t)p(t) \quad (8.6)$$

$$I(t) = i(t)p(t)$$

$$I(T) = \sum_{t \in A(T)} I(t)$$

### 8.3.4

control parameter *maxdepth* = 1 . , CART

```
cart.firstsplit <- rpart(class ~ x1 + x2
  , data = train_df
  , method = "class"
  , parms = list(split = "gini")
  , control = rpart.control(minsplit = 2
    , minbucket = 1
    , maxdepth = 1
    , cp = 0
    , xval = 0
    , maxcompete = 0
  )
)
rpart.plot(cart.firstsplit)
```

Table 8.2 *frame* data frame *t* ,

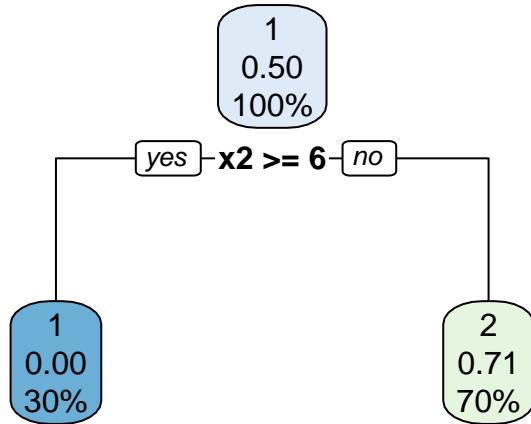


Figure 8.2:

Table 8.2: (frame)

	var	n	wt	dev	yval	complexity	ncompete	nsurrogate
1	x2	10	10	5	1	0.6	0	0
2	\<leaf\>	3	3	0	1	0.0	0	0
3	\<leaf\>	7	7	2	2	0.0	0	0

- var:  $t$  .  $<\text{leaf}>$   $t$
- n:  $N(t)$
- wt: ( appendix )
- dev:
- yval:  $t$
- complexity:  $t$  relative error ; error , relative error 1 .

frame  $yval2$  Table 8.3 .  $yval2$ ,  
( = 2) .  $t$ , .

- 1:  $t$   $j^*$
- 2:  $t$  class=1  $N_1(t)$
- 3:  $t$  class=2  $N_2(t)$
- 4:  $t$  class=1  $p(1|t) \approx \frac{N_1(t)}{N(t)}$
- 5:  $t$  class=2  $p(2|t) \approx \frac{N_2(t)}{N(t)}$
- nodeprob:  $t$   $p(t) \approx \frac{N(t)}{N}$

## Warning: Setting row names on a tibble is deprecated.

CART

Table 8.3: (yval2)

	1	2	3	4	5	nodeprob
1	1	5	5	0.50	0.50	1.0
2	1	3	0	1.00	0.00	0.3
3	2	2	5	0.29	0.71	0.7

```

yval2 x      (i(t))      rpartNodeImpurity .
rpartNodeImpurity <- function(x, yval2) {
  node_vec <- yval2[x, ]
  n.columns <- length(node_vec)
  class.prob <- node_vec[((n.columns/2)+1):(n.columns-1)]
  return(1 - sum(class.prob^2))
}

CART tree    leaf node   rpartNodeImpurity      i(t)      ,      p(t)
I(T)          rpartImpurity .

rpartImpurity <- function(rpart.obj) {
  leaf.nodes <- which(rpart.obj$frame$var=="<leaf>")
  node.impurity <- sapply(leaf.nodes,
                         rpartNodeImpurity,
                         yval2 = rpart.obj$frame$yval2)
  node.prob <- rpart.obj$frame$yval2[leaf.nodes, 'nodeprob']
  return(sum(node.prob * node.impurity))
}

```

Figure 8.1 0.29 .

```

rpartImpurity(cart.firstsplit)

## [1] 0.2857143

maxdepth control parameter
• maxdepth: (default=30)

maxdepth     1 4

library(ggplot2)

tree.impurity <- sapply(c(1:4), function(depth) {
  rpart(class ~ x1 + x2
        , data = train_df
        , method = "class"
        , parms = list(split = "gini")
        , control = rpart.control(minsplit = 2

```

```

        , minbucket = 1
        , maxdepth = depth
        , cp = 0
        , xval = 0
        , maxcompete = 0)) %>%
rpartImpurity()
})

tibble(maxdepth=c(1:4), impurity=tree.impurity) %>%
  ggplot(aes(x=maxdepth, y=impurity)) +
  geom_line()

```

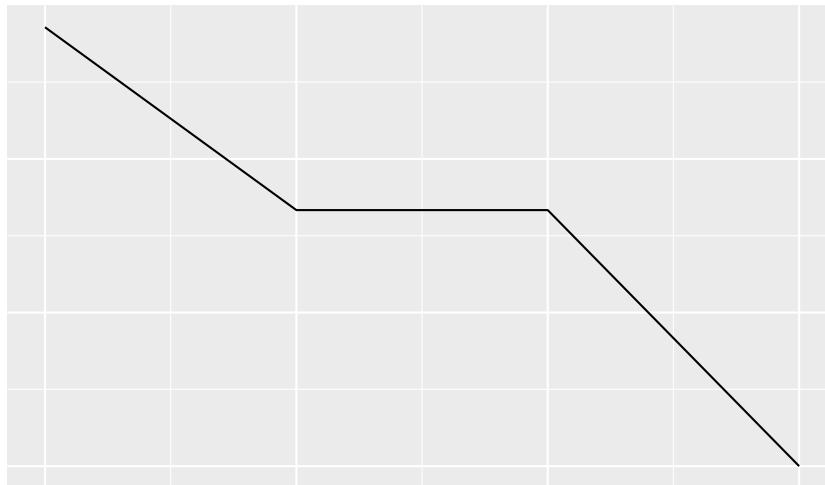


Figure 8.3: maxdepth

```

,          0.29, 0.17, 0.17, 0    . maxdepth 3
      rpart           .           cp control parameter .
• cp:          relative error   (default = 0.01). 0
  cp  0      ,

```

## 8.4

### 8.4.1

8.1  
 bias-variance tradeoff

Table 8.4: (frame)

	var	n	wt	dev	yval	complexity	ncompetet	nsurrogate
1	x2	10	10	5	1	0.6	0	0
2	\<leaf\>	3	3	0	1	0.0	0	0
3	x1	7	7	2	2	0.2	0	0
6	\<leaf\>	1	1	0	1	0.0	0	0
7	x2	6	6	1	2	0.1	0	0
14	x1	2	2	1	1	0.1	0	0
28	\<leaf\>	1	1	0	1	0.0	0	0
29	\<leaf\>	1	1	0	2	0.0	0	0
15	\<leaf\>	4	4	0	2	0.0	0	0

Table 8.4 (*var <leaf>*) (1, 3, 7, 14), snip.rpart

```

internal.node.index <- rownames(cart.est$frame)[which(cart.est$frame$var != '<leaf>')]
as.numeric()
snipped <- lapply(internal.node.index, function(x){snip.rpart(cart.est, x)})
n.trees <- length(snipped)
par(mfrow=c(2,2))
invisible(lapply(c(1:n.trees), function(x) {
  rpart.plot(snipped[[x]])))
)

nodeCost
nodeCost <- function(node, tree) {
  node_vec <- tree$frame$yval2[as.character(node) == row.names(tree$frame), ]
  n.columns <- length(node_vec)
  class.prob.max <- max(node_vec[(n.columns/2)+1):(n.columns-1)])
  node.prob <- node_vec[n.columns]
  node.misclassification.cost <- (1-class.prob.max)*node.prob
  return(node.misclassification.cost)
}

tibble(
  pruning_node = internal.node.index,
  node_cost = sapply(internal.node.index, nodeCost, tree=cart.est)
) %>%
  knitr::kable()

```

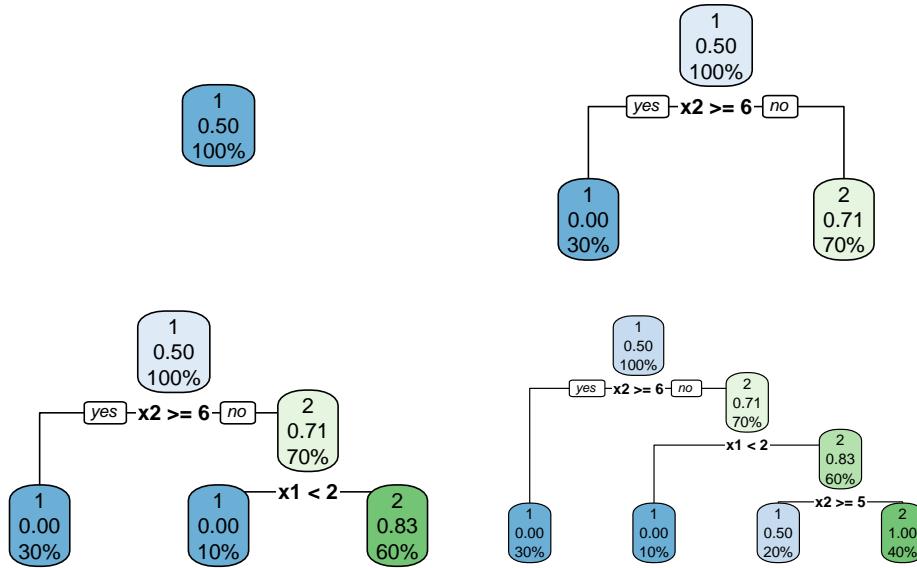


Figure 8.4:

pruning_node	node_cost
1	0.5
3	0.2
7	0.1
14	0.1

```

subtreeEval

subtreeEval <- function(node, tree) {
  snipped <- snip.rpart(tree, node)$frame
  leaf.nodes <- setdiff(rownames(tree$frame[tree$frame$var=="<leaf>",]),
                        rownames(snipped)) %>%
    as.numeric()

  tibble(
    pruning_node = node,
    node.cost = nodeCost(node, tree),
    subtree.cost = sapply(leaf.nodes, nodeCost, tree=tree) %>% sum(),
    subtree.size = length(leaf.nodes)
  ) %>%
  mutate(alpha = (node.cost - subtree.cost) / (subtree.size - 1))
}
  
```

Table 8.5: (df.cost)

pruning_node	node.cost	subtree.cost	subtree.size	alpha
1	0.5	0	5	0.12
3	0.2	0	4	0.07
7	0.1	0	3	0.05
14	0.1	0	2	0.10

```
df.cost <- lapply(internal.node.index, subtreeEval, tree=cart.est) %>%
  bind_rows()
```

Table 8.5

7

```
pruned.tree.1 <- snip.rpart(cart.est,
  df.cost$pruning_node[which.min(df.cost$alpha)])
rpart.plot(pruned.tree.1)
```

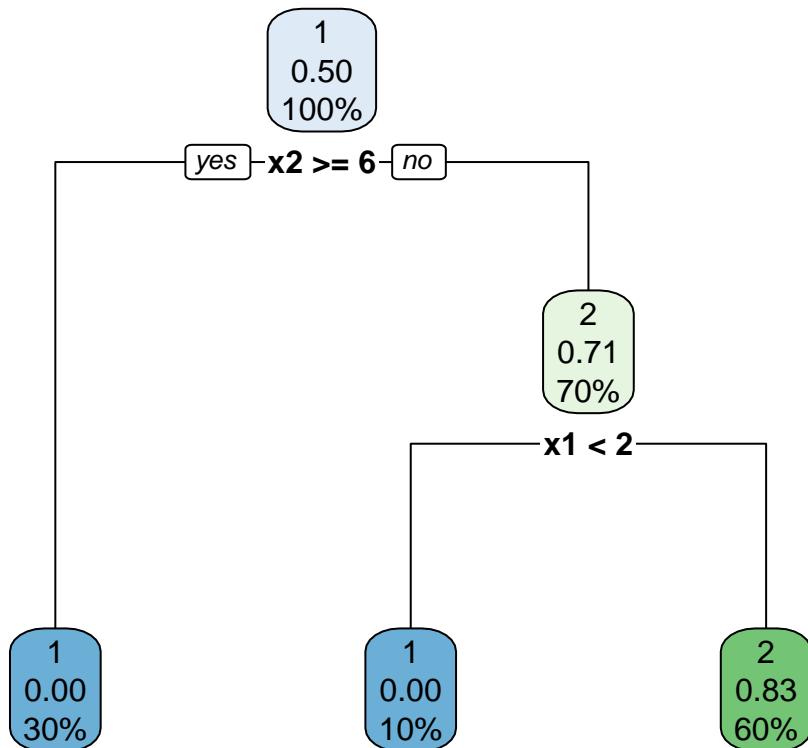


Figure 8.5: 1

```

df.cost <- rownames(pruned.tree.1$frame)[pruned.tree.1$frame$var!="<leaf>"] %>%
  as.numeric() %>%
  lapply(subtreeEval, tree=pruned.tree.1) %>%
  bind_rows()

knitr::kable(df.cost)

```

pruning_node	node.cost	subtree.cost	subtree.size	alpha
1	0.5	0.1	3	0.2
3	0.2	0.1	2	0.1

```

3
.
pruned.tree.2 <- snip.rpart(pruned.tree.1,
                             df.cost$pruning_node[which.min(df.cost$alpha)])
rpart.plot(pruned.tree.2)

```

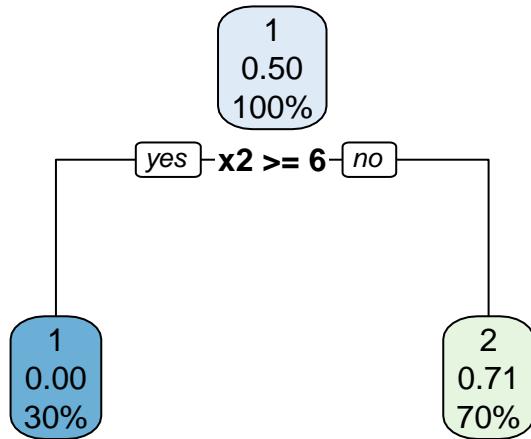


Figure 8.6: 2

### 8.4.2

Table 8.1 , Table 8.6 6

```

test_df <- tibble(
  x1 = c(1,0,3,4,2,1),
  x2 = c(5,5,4,3,7,4),
  class = factor(c(1,1,2,2,1,2), levels=c(1,2))
)

```

Table 8.6:

x1	x2	class
1	5	1
0	5	1
3	4	2
4	3	2
2	7	1
1	4	2

Table 8.7:

x1	x2	class	pred_maxtree	pred_prune1	pred_prune2
1	5	1	1	1	2
0	5	1	1	1	2
3	4	2	2	2	2
4	3	2	2	2	2
2	7	1	1	1	1
1	4	2	1	1	2

, *cart.est* *pruned.tree.1* & *pruned.tree.2*

```

test_pred <- test_df %>%
  bind_cols(
    pred_maxtree = predict(cart.est, test_df, type="class"),
    pred_prune1 = predict(pruned.tree.1, test_df, type="class"),
    pred_prune2 = predict(pruned.tree.2, test_df, type="class")
)

```

Table 8.7

$$1, \quad \quad \quad 1.$$

2 .

4

$$1. \quad \quad \quad R^{ts} \\ 2. \quad n_{test} \quad , \quad .$$

$$SE = \sqrt{\frac{R^{ts}(1 - R^{ts})}{n_{test}}}$$

$$3.~1 \qquad\qquad 2 \qquad\qquad R^{ts} + SE$$

Table 8.8:

	$(\$R^{\wedge}\{ts\})\$$	$(\$SE\$)$	$(\$R^{\wedge}\{ts\} + SE\$)$
cart.est	0.17	0.15	0.32
pruned.tree.1	0.17	0.15	0.32
pruned.tree.2	0.33	0.19	0.53

```

test.summary <- test_pred %>%
  summarize(n.test = n(),
            cart.est = sum(pred_maxtree != class) / n.test,
            pruned.tree.1 = sum(pred_prune1 != class) / n.test,
            pruned.tree.2 = sum(pred_prune2 != class) / n.test) %>%
  gather("tree", "R.ts", -n.test) %>%
  mutate(SE = sqrt((R.ts*(1 - R.ts))/n.test),
        score = R.ts + SE) %>%
  select(-n.test)

  ,
  .

rpart_learn <- function(formula, train_df, test_df) {
  #
  max_tree <- rpart(formula
    , data = train_df
    , method = "class"
    , parms = list(split = "gini")
    , control = rpart.control(minsplit = 2
      , minbucket = 1
      , cp = 0
      , xval = 0
      , maxcompete = 0
    )
  )

  #
  curr_tree <- list()
  k <- 1
  curr_tree[[k]] <- max_tree
  while(dim(curr_tree[[k]]$frame)[1] > 1) {
    internal.node.index <- rownames(curr_tree[[k]]$frame)[which(curr_tree[[k]]$frame$var != '<leading
      as.numeric())
    df.cost <- lapply(internal.node.index, subtreeEval, tree=curr_tree[[k]]) %>% bind_rows()
    curr_tree[[k + 1]] <- snip.rpart(curr_tree[[k]],
  
```

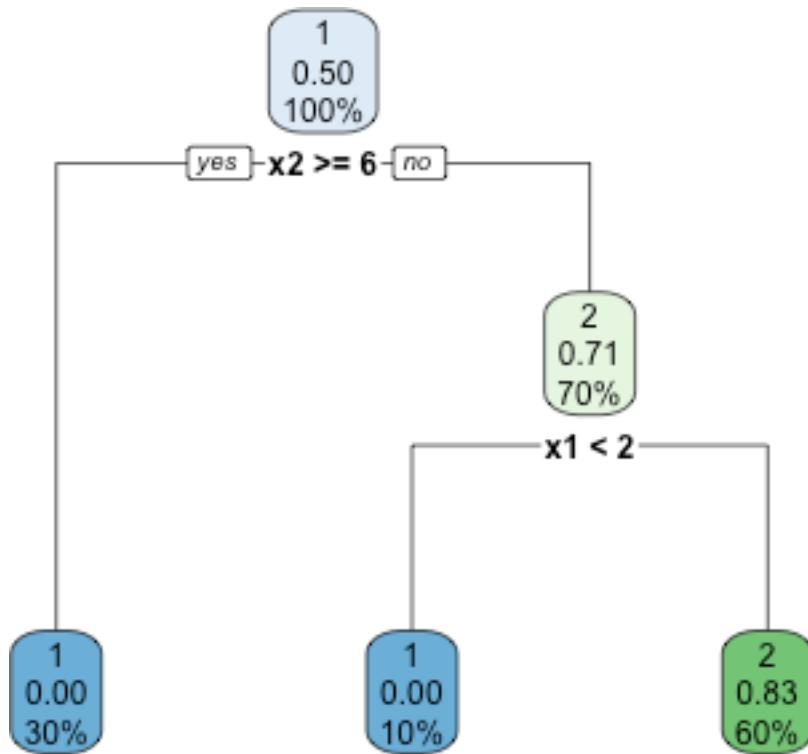
```

        df.cost$pruning_node[which.min(df.cost$alpha)])
k <- k + 1
}

#
n.test <- dim(test_df)[1]
R.ts <- lapply(curr_tree, function(x) {
  sum(predict(x, test_df, type="class") != test_df$class) / n.test
}) %>% unlist()
score <- R.ts + sqrt((R.ts*(1 - R.ts))/n.test)
return(curr_tree[[max(which(score == min(score)))]])
}

optimal_tree <- rpart_learn(class ~ x1 + x2, train_df, test_df)
rpart.plot(optimal_tree)

```



## 8.5 R

*rpart*            8.2 - 8.3            .     *rpart*  
*rpart*            .            .     .

### 8.5.1

$$\begin{aligned}
 i(t) &= \sum_{j=1}^J f(p(j|t)) \\
 p(j|t) &\quad t \quad N(t) \quad j \quad N_j(t) \\
 p(j|t) &\approx \frac{N_j(t)}{N(t)} \\
 f \text{ concave} \quad , f(0) = f(1) = 0 &\quad rpart \quad f \\
 t &\quad s \quad t_L \quad t_R \quad , \\
 \end{aligned}$$

$$\Delta I(s, t) = I(t) - I(t_L) - I(t_R) \quad (8.7)$$

$$= p(t)i(t) - p(t_L)i(t_L) - p(t_R)i(t_R) \quad (8.8)$$

rpart  $\Delta I(s, t)$   $s^*$   $t$   $f$

#### 8.5.1.1

rpart parms split

1. Gini index (parms=list(split='gini'))  $f(p) = p(1-p)$

$$f(p) = p(1-p)$$

2. information index (parms=list(split='information')) (Entropy index)  $f(p) = -p \log(p)$

3. user-defined function

### 8.5.2

$$\begin{aligned}
 t &\quad ( ) \\
 r(t) &= \sum_{j \neq \tau(t)} p(j|t) C(\tau(t)|j) \\
 C(i|j) &\quad j \quad i \quad \tau(t) \quad t \\
 rpart &\quad C(i|j) \\
 C(i|j) &= \begin{cases} 1, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}
 \end{aligned}$$

, *parms*      *loss*       $C(i|j)$       .  
 $A(T) \quad T$       ,       $|T|$       ,       $T \quad R(T)$       .  
 $R(T) = \sum_{t \in A(T)} p(t)r(t)$   
(complexity parameter)  $\alpha \in [0, \infty)$       ,      .  
 $R_\alpha(T) = R(T) + \alpha|T|$   
,       $\alpha$        $R_\alpha(T)$        $T_\alpha$       ,      .  
•  $T_0$ :  
•  $T_\infty$ :      ( )  
•  $\alpha > \beta$       ,  $T_\alpha \quad T_\beta$        $T_\beta$       .

### 8.5.3

*rpart*      ,  
• *minsplit*:      (default=20)  
• *cp*:      relative error      (default = 0.01).      0  
• *maxdepth*:      (default=30)

# Chapter 9

## 9.1

(suuport vector machine; SVM)

## 9.2 R package

R

package	version
tidyverse	1.3.1
e1071	1.7-7
Matrix	1.3-4
quadprog	1.5-8

## 9.3 SVM -

### 9.3.1 R

```
train_df <- tibble(
  x1 = c(5, 4, 7, 8, 3, 2, 6, 9, 5),
  x2 = c(7, 3, 8, 6, 6, 5, 6, 6, 4),
  class = c(1, -1, 1, 1, -1, -1, 1, 1, -1)
)

knitr::kable(train_df, booktabs = TRUE,
             align = c('r', 'r', 'r'),
             caption = '')
```

Table 9.1       $x_1, x_2$        $class$       9       $train\_df$  data frame

Table 9.1:

x1	x2	class
5	7	1
4	3	-1
7	8	1
8	6	1
3	6	-1
2	5	-1
6	6	1
9	6	1
5	4	-1

```
library(e1071)
svm_model <- svm(as.factor(class) ~ x1 + x2, data = train_df, kernel = "linear", scale = TRUE)
plot(svm_model, data = train_df, formula = x2 ~ x1, grid = 200)
```

$$9.1 \quad (\text{X} \quad ), \quad ( = -1, \quad = 1) \quad .$$

$$0.6666667x_1 + 0.6666667x_2 = 7$$

### 9.3.2

- $\mathbf{x} \in \mathbb{R}^p$ : p
- $y \in \{-1, 1\}$ :
- $N$ :
- $(\mathbf{x}_i, y_i)$ : i

#### 9.3.3

SVM

$$\mathbf{w}^\top \mathbf{x} + b = 0 \tag{9.1}$$

$$\mathbf{w} \in \mathbb{R}^p \quad b \in \mathbb{R}$$

1

$$H_1 : \mathbf{w}^\top \mathbf{x} + b = 1$$

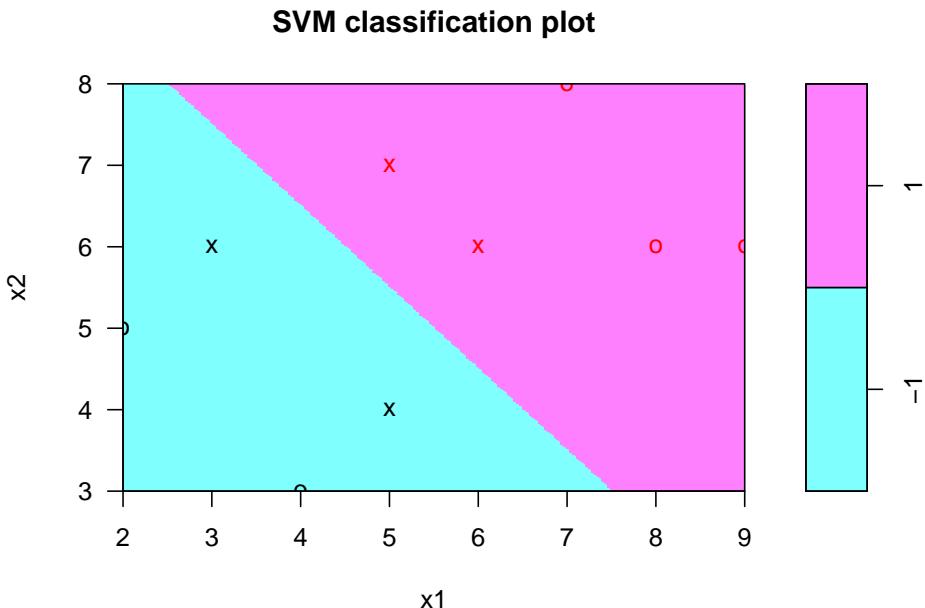


Figure 9.1: SVM

-1

$$H_2 : \mathbf{w}^\top \mathbf{x} + b = -1$$

$$H_1 \quad H_2 \quad (\text{margin}) \quad 2/\|\mathbf{w}\| \quad . \quad \text{SVM} \quad H_1 \quad H_2 \quad .$$

$$\begin{aligned} & \max \frac{2}{\mathbf{w}^\top \mathbf{w}} \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{x}_i + b \geq 1 \text{ for } y_i = 1 \\ & \mathbf{w}^\top \mathbf{x}_i + b \leq -1 \text{ for } y_i = -1 \end{aligned}$$

$$\begin{aligned} & \min \frac{\mathbf{w}^\top \mathbf{w}}{2} \\ \text{s.t.} \quad & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \end{aligned}$$

$$, \quad i \quad (\text{Lagrange multiplier}) \quad \alpha_i \geq 0 \quad (9.2)$$

(primal problem) .

$$\begin{aligned} \min \quad L_P &= \frac{1}{2} \mathbf{w}^\top \mathbf{w} + \sum_{i=1}^N \alpha_i [y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1] \\ \text{s.t. } \alpha_i &\geq 0, \quad i = 1, \dots, N \end{aligned} \quad (9.2)$$

(9.2)      (Wolfe dual problem)      (9.3)      .      ( , 2012) .

$$\begin{aligned} \max \quad L_D &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t. } \sum_{i=1}^N \alpha_i y_i &= 0 \\ \alpha_i &\geq 0, \quad i = 1, \dots, N \end{aligned} \quad (9.3)$$

(9.3)      (quadratic programming) ,      .      quadprog  
 .      e1071 svm ,      .      .  
 quadprog solve.QP      formulation      (Goldfarb and Idnani, 1983)

$$\begin{aligned} \min \quad & -\mathbf{d}^\top \alpha + \frac{1}{2} \alpha^\top \mathbf{D} \alpha \\ \text{s.t. } \mathbf{A}^\top \alpha &\geq \mathbf{b}_0 \end{aligned} \quad (9.4)$$

(9.4)      (9.3) .

$$\begin{aligned} \mathbf{d} &= \mathbf{1}_{N \times 1} \\ \mathbf{D} &= \mathbf{y} \mathbf{y}^\top \mathbf{X} \mathbf{X}^\top \end{aligned}$$

where

$$\begin{aligned} \mathbf{y} &= [y_1 \quad y_2 \quad \cdots \quad y_N]^\top \\ \mathbf{X} &= [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_N]^\top \end{aligned}$$

```
N <- dim(train_df)[1]
X <- train_df[c('x1', 'x2')] %>% as.matrix()
y <- train_df[['class']] %>% as.numeric()

d <- rep(1, N)
D <- (y %*% t(y)) * (X %*% t(X))
```

Table 9.2:

variable	solution
alpha_1	0.2234
alpha_2	0.0000
alpha_3	0.0000
alpha_4	0.0000
alpha_5	0.2228
alpha_6	0.0000
alpha_7	0.2210
alpha_8	0.0000
alpha_9	0.2216

**D** determinant 0 , Goldfarb and Idnani (1983) symmetric positive definite matrix      **solve.QP** .      **Matrix**      **nearPD**  
**D** symmetric positive definite matrix .

```
D_pd <- Matrix::nearPD(D, doSym = T)$mat %>% as.matrix()
```

$$(9.4) \quad \text{inequality} \quad , \quad (9.3) \quad \text{equality constraint} \quad \sum_{i=1}^N \alpha_i y_i = 0 \\ \sum_{i=1}^N \alpha_i y_i \geq 0 \quad \sum_{i=1}^N -\alpha_i y_i \geq 0 \quad .$$

$$\mathbf{A}^\top = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \\ -y_1 & -y_2 & \cdots & -y_N \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \mathbf{b}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$

```
A <- cbind(
  y,
  -y,
  diag(N)
)
b_zero <- rep(0, 2 + N)

solve.QP
res <- quadprog::solve.QP(D_pd, d, A, b_zero)
alpha_sol <- res$solution
obj_val <- -res$value
```

9.2 ( , 2012) , 0.4444 .  
 $\alpha_i^*$  ,

$$\mathbf{w} = \sum_{i=1}^N \alpha_i^* y_i \mathbf{x}_i$$

$$b = \sum_{i:\alpha_i^*>0} \frac{1 - y_i \mathbf{w}^\top \mathbf{x}_i}{y_i} \left/ \sum_{i:\alpha_i^*>0} 1 \right.$$

```
w <- colSums(alpha_sol * y * X)
print(w)

##          x1          x2
## 0.6666658 0.6666657

sv_ind <- which(round(alpha_sol, digits = 4) > 0)
b <- mean((1 - y[sv_ind] * (X[sv_ind, ] %*% w)) / y[sv_ind])
print(b)

## [1] -6.99999
```

## 9.4 SVM -

(9.2)

### 9.4.1 R

```
inseparable_train_df <- bind_rows(train_df,
                                      tibble(x1 = 7, x2 = 6, class = -1))

knitr::kable(inseparable_train_df, booktabs = TRUE,
             align = c('r', 'r', 'r'),
             caption = '')

library(e1071)
svm_model <- svm(as.factor(class) ~ x1 + x2, data = inseparable_train_df,
                  kernel = "linear", cost = 1, scale = FALSE)
plot(svm_model, data = inseparable_train_df, formula = x2 ~ x1, grid = 200)
```

Figure 9.2 , ( = -1) 1 , ,

$$0.6x_1 + 0.8x_2 = 7.6$$

Table 9.3:

x1	x2	class
5	7	1
4	3	-1
7	8	1
8	6	1
3	6	-1
2	5	-1
6	6	1
9	6	1
5	4	-1
7	6	-1

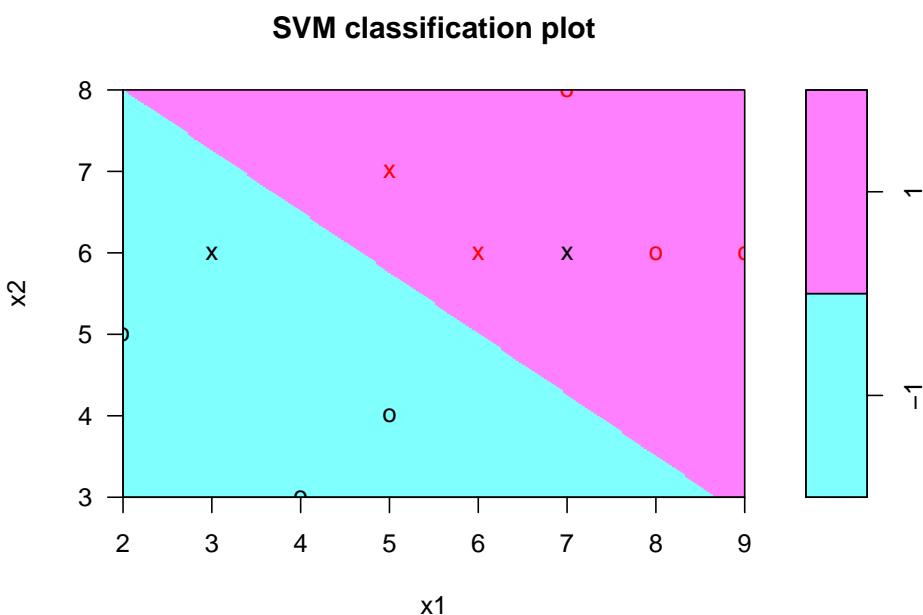


Figure 9.2: SVM

### 9.4.2

$$(\text{slack variable}) \quad \xi_i \quad i = 1, \dots, N \quad .$$

$$\xi_i = \max \{0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}$$

$$C = \begin{pmatrix} -1 & H_1, & -1 & H_2 \\ & \cdot \end{pmatrix} \quad . \quad \xi_i$$

$$\min \frac{\mathbf{w}^\top \mathbf{w}}{2} + C \sum_{i=1}^N \xi_i$$

s.t.

$$y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, N$$

$$\xi \geq 0, \quad i = 1, \dots, N$$

### 9.3.3 (9.5)

$$\max L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

s.t.

(9.5)

$$0 \leq \alpha_i \leq C, i = 1, \dots, N$$

(9.3)  $\alpha_i$        $C$       ,    e1071    svm      LIBSVM  
 (Chang and Lin, 2011)  $C$ -support vector classification( $C$ -SVC)      , LIBSVM  
 (Fan et al., 2005)      .

```
svm           type = "C-classification"  (9.3)           , cost = 1  
C -> 1
```

```
svm_model <- svm(as.factor(class) ~ x1 + x2, data = inseparable_train_df,  
                  kernel = "linear", scale = FALSE,  
                  type = "C-classification", cost = 1)
```

`, coefs`       $y_i$       (9.5)      .

```
N <- dim(inseparable_train_df)[1]
```

```
X <- inseparable_train_df[c('x1', 'x2')] %>% as.matrix()
y <- inseparable_train_df[['class']] %>% as.numeric()
```

Table 9.4: :

variable	solution
alpha_1	0.8
alpha_2	0.0
alpha_3	0.0
alpha_4	0.0
alpha_5	0.8
alpha_6	0.0
alpha_7	1.0
alpha_8	0.0
alpha_9	0.0
alpha_10	1.0

```
sv_ind <- svm_model$index
alpha_sol <- vector("numeric", N)
alpha_sol[sv_ind] <- drop(svm_model$coefs[, 1]) / y[sv_ind]
```

**w**

$$\mathbf{w} = \sum_{i=1}^N \alpha_i^* y_i \mathbf{x}_i$$

```
w <- colSums(alpha_sol * y * X)
print(w)
```

```
## x1 x2
## 0.6 0.8
```

**b**

$$b = \sum_{i:0<\alpha_i^*<C} \frac{1 - y_i \mathbf{w}^\top \mathbf{x}_i}{y_i} \left/ \sum_{i:0<\alpha_i^*<C} 1 \right.$$

```
ind <- sv_ind[alpha_sol[sv_ind] < svm_model$cost]
b <- mean((1 - y[ind] * (X[ind, ] %*% w)) / y[ind])
print(b)
```

```
## [1] -7.6
```

**w b svm**

Table 9.5: C

\$C\$	$\$(w_1, w_2)$$	\$b\$			
1	0.6, 0.8	-7.6	1, 7, 5, 10	10	
5	0.4, 1.2	-9.4	1, 4, 7, 5, 10	10	
100	0.4, 1.2	-9.4	1, 4, 7, 5, 10	10	

```
w <- t(svm_model$coefs) %*% svm_model$SV
print(w)

##      x1   x2
## [1,] 0.6 0.8

b <- -svm_model$rho
print(b)

## [1] -7.6

,      C           . C    1, 5, 100

svm_models <- lapply(c(1, 5, 100), function(C)
  svm(as.factor(class) ~ x1 + x2, data = inseparable_train_df,
    kernel = "linear", scale = FALSE,
    type = "C-classification", cost = C))

getHyperplane <- function(model) {
  list(C = model$cost,
       w = paste(round(t(model$coefs) %*% model$SV, digits = 2), collapse = ", "),
       b = -round(model$rho, digits = 2),
       sv = paste(model$index, collapse = ", "),
       misclassified = paste(which(model$fitted != as.factor(inseparable_train_df$class)),
                             collapse = ", "))
}

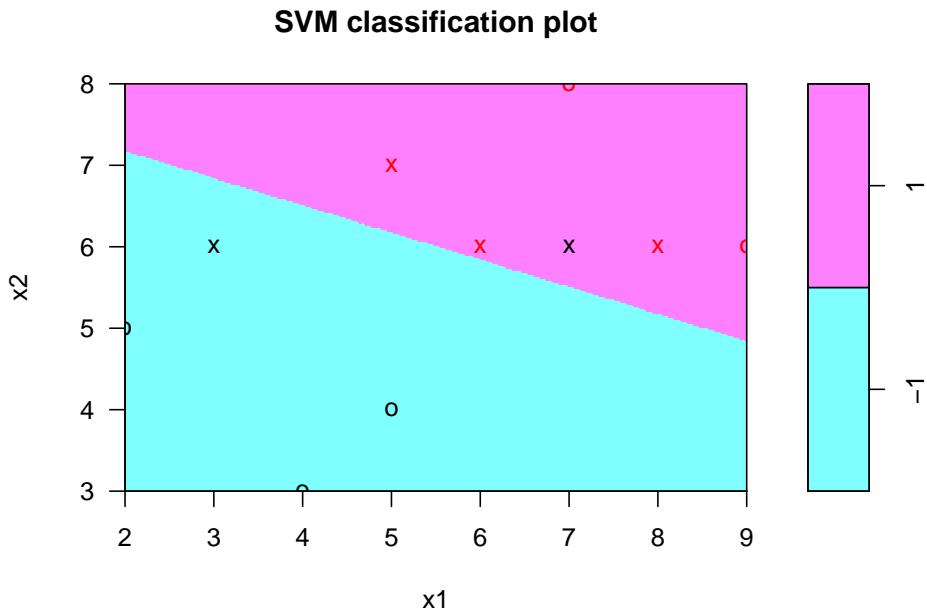
svm_summary <- lapply(svm_models, getHyperplane) %>% bind_rows()

Table 9.5      ,      C    1 5           . C    5 100
plot(svm_models[[2]], data = inseparable_train_df, formula = x2 ~ x1, grid = 200)

Figure 9.3      (C = 5    )  Figure 9.2      (C = 1    )

```

## 9.5 SVM

Figure 9.3: SVM ( $C = 5$ )

### 9.5.1 R

```
nonlinear_train_df <- tibble(
  x1 = c(5, 4, 7, 8, 3, 2, 6, 9, 5),
  x2 = c(7, 3, 8, 6, 6, 5, 6, 6, 4),
  class = c(1, -1, -1, -1, 1, 1, 1, -1, -1)
)

knitr::kable(nonlinear_train_df, booktabs = TRUE,
            align = c('r', 'r', 'r'),
            caption = 'SVM')

library(e1071)
svm_model <- svm(as.factor(class) ~ x1 + x2, data = nonlinear_train_df,
                  kernel = "polynomial", coef0 = 1, gamma = 1, degree = 2,
                  cost = 5, scale = FALSE)
plot(svm_model, data = nonlinear_train_df, formula = x2 ~ x1, grid = 200)
```

### 9.5.2

(9.1)

$$f(\mathbf{x}) = \Phi(\mathbf{x})^\top \mathbf{w} + b \quad (9.6)$$

Table 9.6: SVM

x1	x2	class
5	7	1
4	3	-1
7	8	-1
8	6	-1
3	6	1
2	5	1
6	6	1
9	6	-1
5	4	-1

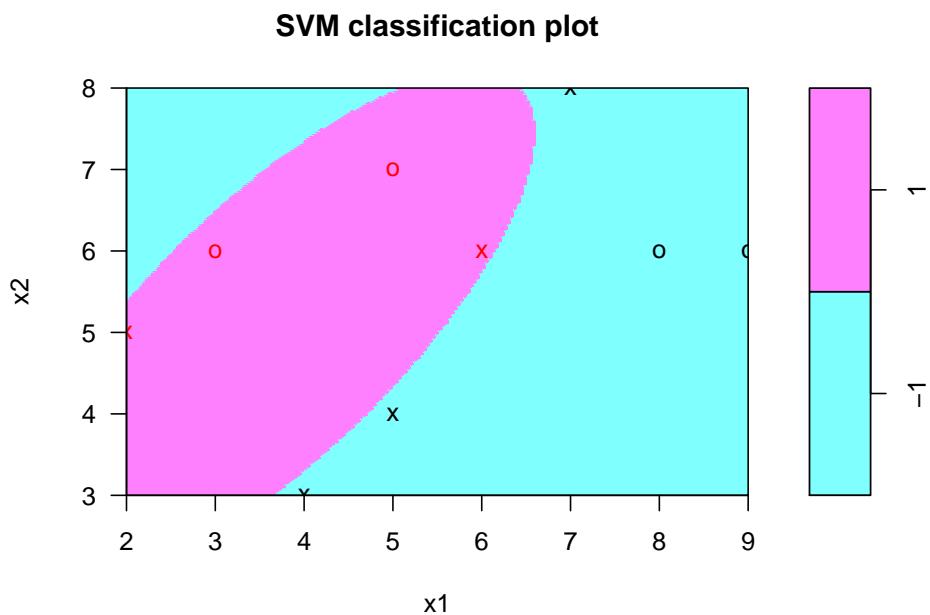


Figure 9.4: SVM

$$\begin{array}{lll} \Phi : \mathbb{R}^p \rightarrow \mathbb{R}^m & \mathbf{x} & (\text{feature}) \\ \cdot \quad \mathbf{x} & (\text{basis function}) & , \quad m \\ \text{SVM} & (9.5) & (\mathbf{w} \in \mathbb{R}^m). \quad , \quad m \quad \mathbf{x} \quad p \end{array}$$

$$\begin{aligned} \max \quad L_D = & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j) \\ \text{s.t.} \quad & \sum_{i=1}^N \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, N \end{aligned} \quad (9.7)$$

$$(9.7) \quad \begin{array}{lll} \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j) & (\text{kernel function}) & , \quad \mathbf{x}_i, \mathbf{x}_j \\ (\text{similarity measure}) & . & \end{array}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$$

$$\begin{array}{ll} \text{Gaussian RBF:} & K(\mathbf{x}_i, \mathbf{x}_j) = \exp \left( \frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right) \\ \text{$r$-th order polynomial:} & K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^r \\ \text{Sigmoid:} & K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i^\top \mathbf{x}_j - \delta) \end{array}$$

$$\begin{aligned} \max \quad L_D = & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j k_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^N \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, N \end{aligned} \quad (9.8)$$

$$(9.8) \quad \begin{array}{lll} k_{ij} & K(\mathbf{x}_i, \mathbf{x}_j) & . \quad (9.8) \quad \alpha^* \quad \text{SVM} \quad (\text{quadratic programming}) \\ / & . & . \end{array}$$

Table 9.6      e1071      svm      . svm      kernel =  
 "polynomial"

Table 9.7: C SVM

	\$C\$	
1	1, 7, 2, 3, 9	7
5	6, 7, 2, 3, 9	
100	6, 7, 2, 3, 9	

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^\top \mathbf{x}_j + \beta_0)^r$$

$\gamma, \beta_0, r$  svm gamma, coef0, degree .

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^2$$

SVM svm .

```
svm_model <- svm(as.factor(class) ~ x1 + x2, data = nonlinear_train_df,
                  kernel = "polynomial", coef0 = 1, gamma = 1, degree = 2,
                  scale = FALSE)
```

1, 7, 2, 3, 9 .

SVM C . SVM C = 1,5,100 SVM .

```
svm_models <- lapply(
  c(1, 5, 100),
  function(C)
    svm(as.factor(class) ~ x1 + x2, data = nonlinear_train_df,
        kernel = "polynomial", coef0 = 1, gamma = 1, degree = 2,
        cost = C, scale = FALSE)
  )

getSummary <- function(model) {
  list(C = model$cost,
       sv = paste(model$index, collapse = ", "),
       misclassified = paste(which(model$fitted != as.factor(nonlinear_train_df$class)))
  }

svm_summary <- lapply(svm_models, getSummary) %>% bind_rows()
```

## 9.6 R SVM

### 9.6.1

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

```

 $\gamma$            , svm   gamma
svm_model <- svm(as.factor(class) ~ x1 + x2, data = nonlinear_train_df,
                  kernel = "radial", gamma = 1,
                  cost = 5, scale = FALSE)
plot(svm_model, data = nonlinear_train_df, formula = x2 ~ x1, grid = 200)

```

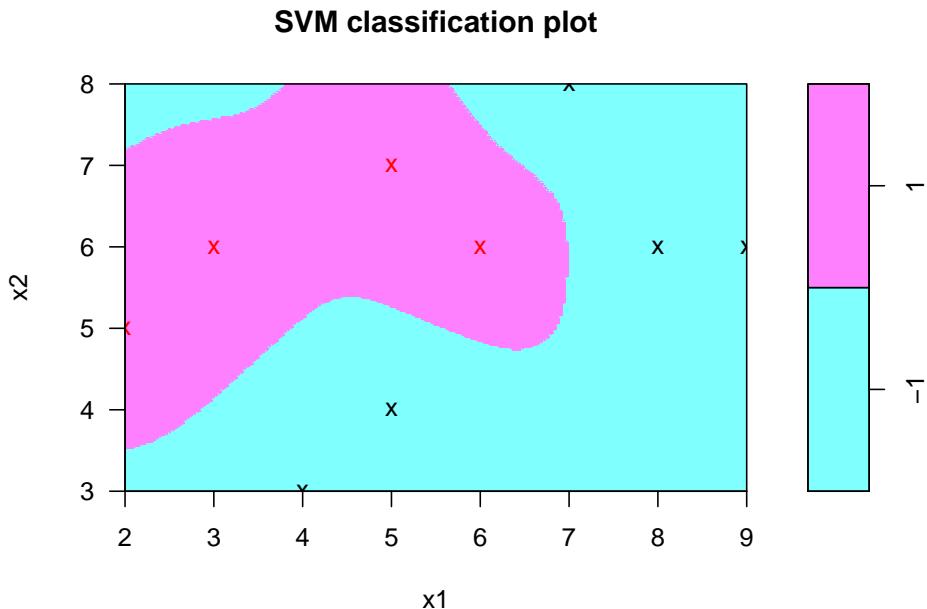


Figure 9.5: SVM

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\gamma \mathbf{x}_i^\top \mathbf{x}_j + \beta_0\right)$$

```

 $\gamma, \beta_0$       svm   gamma, coef0
svm_model <- svm(as.factor(class) ~ x1 + x2, data = nonlinear_train_df,
                  kernel = "sigmoid", gamma = 0.01, coef0 = -1,
                  cost = 5, scale = FALSE)
plot(svm_model, data = nonlinear_train_df, formula = x2 ~ x1, grid = 200)

kernel,      gamma, coef0, degree,      cost  svm

```

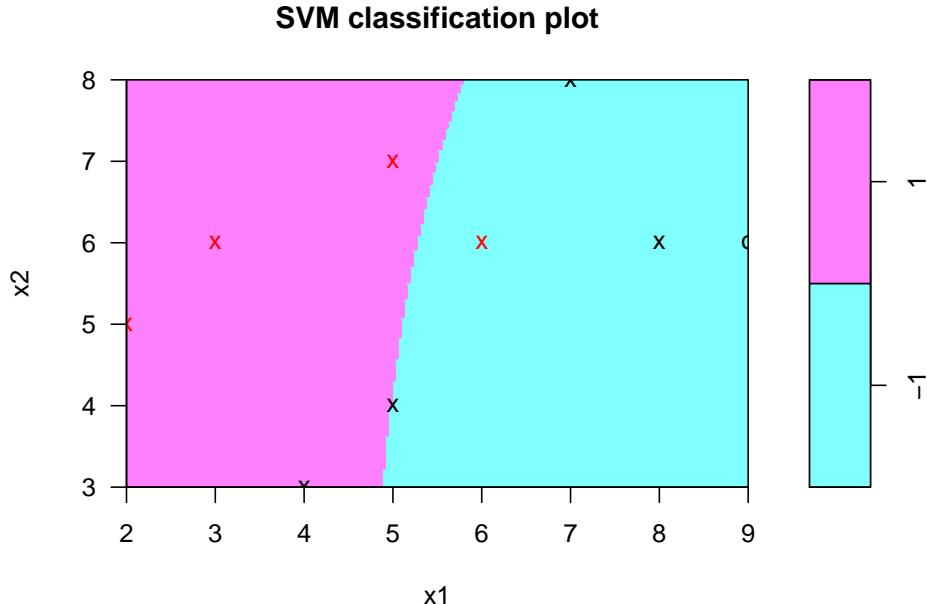


Figure 9.6: SVM

### 9.6.2 $\nu$ -SVC

$\nu$ -support vector classification( $\nu$ -SVC) (Schölkopf et al., 2000, Chang and Lin (2001))  $C$ -SVC      (9.8)      ,       $C = \nu$  .

$$\min L_D = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j k_{ij}$$

s.t.

$$\sum_{i=1}^N \alpha_i y_i = 0 \quad (9.9)$$

$$0 \leq \alpha_i \leq \frac{1}{N}, \quad i = 1, \dots, N$$

$$\sum_{i=1}^N \alpha_i = \nu$$

$$, \quad \alpha_i = 1/N \quad , \nu = \frac{\alpha_i}{\nu} \quad , \quad \nu \in (0, 1] \quad (9.9)$$

$$0 < \nu \leq \frac{2}{N} \min \left( \sum_i I(y_i = 1), \sum_i I(y_i = -1) \right)$$

(Chang and Lin, 2001),  $1 - 10\% \approx 0.2$ .  
**svm** LIBSVM (9.9)  $N$  ( ) .

$$\min L_D = \sum_{i=1}^N \sum_{j=1}^N \bar{\alpha}_i \bar{\alpha}_j y_i y_j k_{ij}$$

s.t.

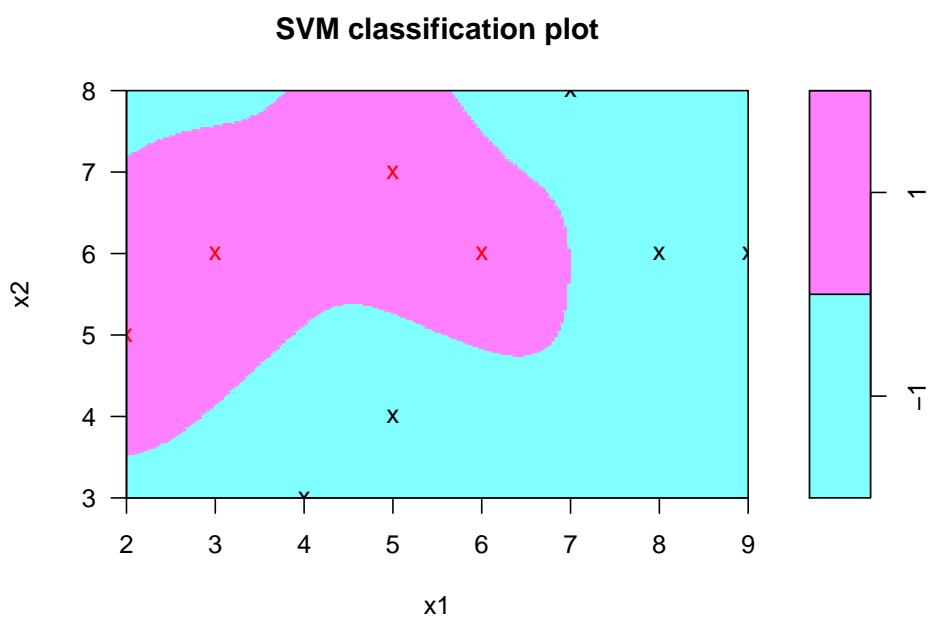
$$\sum_{i=1}^N \bar{\alpha}_i y_i = 0 \quad (9.10)$$

$$0 \leq \bar{\alpha}_i \leq 1, \quad i = 1, \dots, N$$

$$\sum_{i=1}^N \bar{\alpha}_i = \nu N$$

$$\bar{\alpha}_i = \alpha_i N .$$

```
<-- SVC svm type = "nu-classification" nu .
svm_model <- svm(as.factor(class) ~ x1 + x2, data = nonlinear_train_df,
  type = "nu-classification",
  kernel = "radial", gamma = 1,
  nu = 0.5, scale = FALSE)
plot(svm_model, data = nonlinear_train_df, formula = x2 ~ x1, grid = 200)
```

Figure 9.7:  $\nu$ -SVC

# Chapter 10

## 10.1 R

package	version
tidyverse	1.3.1
caret	6.0-88

## 10.2

$$\{(\mathbf{x}_i, y_i)\}_{i=1,\dots,N}$$

- $\mathbf{x}_i$ :  $i$  ( $\mathbf{x}_i = [x_{i1} x_{i2} \cdots x_{ip}]^\top$ )
  - $J$ :
  - $y_i$ :  $i$ ;  $y_i \in \{1, 2, \dots, J\}$
- $d(\mathbf{x})$  (misclassification rate)

$$R(d) = \frac{1}{N} \sum_{i=1}^N I(d(\mathbf{x}_i) \neq y_i) \quad (10.1)$$

$$I(x) \quad x \quad (\text{true}) \quad 1, \quad (\text{false}) \quad 0$$

(10.1) , (overfitting) ,

$$d(\mathbf{x}) \quad L \quad \{(\mathbf{x}_i, y_i)\}_{i=N+1,\dots,N+L}$$

$$R^{ts}(d) = \frac{1}{L} \sum_{i=N+1}^{N+L} I(d(\mathbf{x}_i) \neq y_i) \quad (10.2)$$

cross validation .

### 10.3 ,

- ) , (1 +) (0 -) . (1 + ) (0  
 (confusion matrix) .

```
cm <- matrix(c('a', 'b', 'c', 'd'), nrow = 2, byrow = TRUE)
attr(cm, "dimnames") <- list(Prediction = c("1", "0"), Reference = c("1", "0"))
class(cm) <- "table"
print(cm)

##             Reference
## Prediction 1 0
##           1 a b
##           0 c d
```

- *a*: number of true positive prediction
- *b*: number of false positive prediction
- *c*: number of false negative prediction
- *d*: number of true negative prediction

“positive” “negative” “ ” “ ” , “true” “false” .

$$\text{misclassification rate} = \frac{b+c}{a+b+c+d} \quad (10.3)$$

(accuracy) , .

$$\text{accuracy} = \frac{a+d}{a+b+c+d} = 1 - \text{misclassification rate} \quad (10.4)$$

, (sensitivity) , .

$$\text{sensitivity} = \frac{a}{a+c} \quad (10.5)$$

(specificity) .

$$\text{specificity} = \frac{d}{b+d} \quad (10.6)$$

$$\text{accuracy} = \frac{a+c}{a+b+c+d} \text{sensitivity} + \frac{b+d}{a+b+c+d} \text{specificity}$$

,

### 10.3.1 R

100

$$y_i = \begin{cases} 1 & i = 1, \dots, 20 \\ 0 & i = 21, \dots, 100 \end{cases},$$

$$\hat{y}_i = \begin{cases} 1 & i = 1, \dots, 15, 91, \dots, 100 \\ 0 & i = 16, \dots, 90 \end{cases}$$

```
y <- factor(c(rep(1, 20), rep(0, 80)), levels = c(1, 0))
y_hat <- factor(c(rep(1, 15), rep(0, 75), rep(1, 10)), levels = c(1, 0))

caret    confusionMatrix

cm <- caret::confusionMatrix(data = y_hat, reference = y)

table component

cm$table

##             Reference
## Prediction  1  0
##           1 15 10
##           0  5 70

overall   component

cm$overall

##      Accuracy      Kappa AccuracyLower AccuracyUpper AccuracyNull
##      0.8500000  0.5714286  0.7646925  0.9135456  0.8000000
## AccuracyPValue  McnemarPValue
##      0.1285055  0.3016996

, , byClass component
```

```
cm$byClass
```

	Sensitivity	Specificity	Pos Pred Value
##	0.7500000	0.8750000	0.6000000
##	Neg Pred Value	Precision	Recall
##	0.9333333	0.6000000	0.7500000
##	F1	Prevalence	Detection Rate
##	0.6666667	0.2000000	0.1500000
##	Detection Prevalence	Balanced Accuracy	
##	0.2500000	0.8125000	

## 10.4 ROC

```
    . , . , .  
10  
train_df <- tribble(  
  ~x, ~y,  
  24, 0,  
  35, 0,  
  37, 1,  
  42, 0,  
  49, 1,  
  54, 1,  
  56, 0,  
  68, 1,  
  72, 1,  
  73, 1  
) %>%  
  mutate(y = factor(y, levels = c(1, 0)))  
  
x < 40    0, x ≥ 40    1    , .  
cm40 <- caret::confusionMatrix(  
  factor(as.integer(train_df$x >= 40)), levels = c(1, 0)),  
  train_df$y  
)  
  
cm40$table  
  
##      Reference  
## Prediction 1 0  
##             1 5 2  
##             0 1 2
```

```

cm40$byClass[c("Sensitivity", "Specificity")]

## Sensitivity Specificity
## 0.8333333 0.5000000

,      x < 50    0, x ≥ 50    1      ,
cm50 <- caret::confusionMatrix(
  factor(as.integer(train_df$x >= 50), levels = c(1, 0)),
  train_df$y
)

cm50$table

##             Reference
## Prediction 1 0
##               1 4 1
##               0 2 3

,
cm50$byClass[c("Sensitivity", "Specificity")]

## Sensitivity Specificity
## 0.6666667 0.7500000

x 40
x  (threshold)
.
univariate_binary_rule <- function(x, y, th) {
  cm <- caret::confusionMatrix(
    factor(as.integer(x >= th), levels = c(1, 0)),
    y
  )

  tibble(threshold = th,
         sensitivity = cm$byClass["Sensitivity"],
         specificity = cm$byClass["Specificity"])
}

th <- c(sort(train_df$x), Inf)

roc_df <- map_dfr(th, univariate_binary_rule, x = train_df$x, y = train_df$y)

knitr::kable(
  roc_df, booktabs = TRUE,
  align = c('r', 'r', 'r', 'r'),
  col.names = c('  (x$)', ' (sensitivity)', ' (specificity)'),
```

Table 10.1:

(\$x\$)	(sensitivity)	(specificity)
24	1.0000000	0.00
35	1.0000000	0.25
37	1.0000000	0.50
42	0.8333333	0.50
49	0.8333333	0.75
54	0.6666667	0.75
56	0.5000000	0.75
68	0.5000000	1.00
72	0.3333333	1.00
73	0.1666667	1.00
Inf	0.0000000	1.00

ROC(receiver operating characteristic)  
 $(1 - \alpha)(\text{false positive rate})^x \cdot y$ .

Table 10.1 ROC

```
roc_df %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path()
```

10.5

$$k \quad j \quad n_{kj} \quad , \quad j \quad k \quad . \quad ( \quad K \quad . \quad ,$$

$$k \quad \sum_{i=1}^J n_{kj} = \frac{N}{K} \quad . )$$

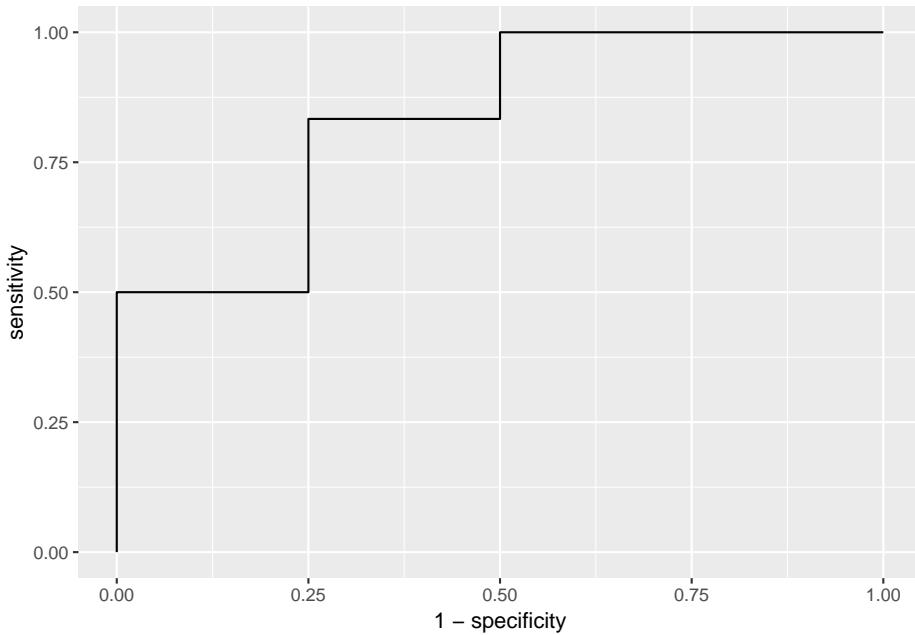


Figure 10.1: ROC

$$\begin{aligned}
 \% \text{ captured response} &= \frac{n_{kj}}{\sum_{k=1}^K n_{kj}} \times 100 \\
 \text{cumulative \% captured response} &= \frac{\sum_{l=1}^k n_{lj}}{\sum_{k=1}^K n_{kj}} \times 100 \\
 \% \text{ response} &= \frac{n_{kj}}{\sum_{j=1}^J n_{kj}} \times 100 \\
 \text{lift} &= \frac{n_{kj}}{\frac{1}{K} \sum_{k=1}^K n_{kj}}
 \end{aligned}$$

```

1,000
.
y_freq <- tribble(
  ~y, ~n,
  1, 437,
  2, 348,
  3, 215
) %>%
  mutate(y = factor(y, levels = c(1, 2, 3)))

```

```
y_freq

## # A tibble: 3 x 2
##   y      n
##   <fct> <dbl>
## 1 1     437
## 2 2     348
## 3 3     215

,
1(    )      ,
1          .      100

freq_within_group <- tribble(
  ~k, ~n,
  1, 92,
  2, 78,
  3, 64,
  4, 57,
  5, 43,
  6, 35,
  7, 29,
  8, 22,
  9, 7,
  10, 10
)

freq_within_group

## # A tibble: 10 x 2
##       k     n
##   <dbl> <dbl>
## 1     1     92
## 2     2     78
## 3     3     64
## 4     4     57
## 5     5     43
## 6     6     35
## 7     7     29
## 8     8     22
## 9     9      7
## 10    10    10

1          .

stat_df <- freq_within_group %>%
  mutate(cum_n = cumsum(n)) %>%
  mutate(
```

Table 10.2:

	1	% captured response	cum. % captured response	% response	lift
1	92	21.05	21.05	92	2.11
2	78	17.85	38.90	78	1.78
3	64	14.65	53.55	64	1.46
4	57	13.04	66.59	57	1.30
5	43	9.84	76.43	43	0.98
6	35	8.01	84.44	35	0.80
7	29	6.64	91.08	29	0.66
8	22	5.03	96.11	22	0.50
9	7	1.60	97.71	7	0.16
10	10	2.29	100.00	10	0.23

```

captured_response_pct = n / sum(n) * 100,
cum_captured_response_pct = cum_n / sum(n) * 100,
response_pct = n / 100 * 100,
lift = n / mean(n)
) %>%
select(-cum_n)

knitr::kable(
  stat_df,
  booktabs = TRUE,
  align = rep('r', 6),
  col.names = c(' ', ' 1  ', '% captured response',
               'cum. % captured response', '% response', 'lift'),
  caption = '      ',
  digits = 2
)

```

Table 10.2

```

stat_df %>%
gather(key = "stat", value = "value",
       captured_response_pct:lift) %>%
ggplot(aes(x = k, y = value)) +
geom_point() +
geom_line() +
facet_wrap(vars(stat), nrow = 2, ncol = 2, scales = "free_y",
           labeller = as_labeller(
             c("captured_response_pct" = "% captured response",
               "cum_captured_response_pct" = "cum. % captured response",

```

```
    "response_pct" = "% response",
    "lift" = "lift")
)) +
xlab("group") +
ylab("statistics")
```

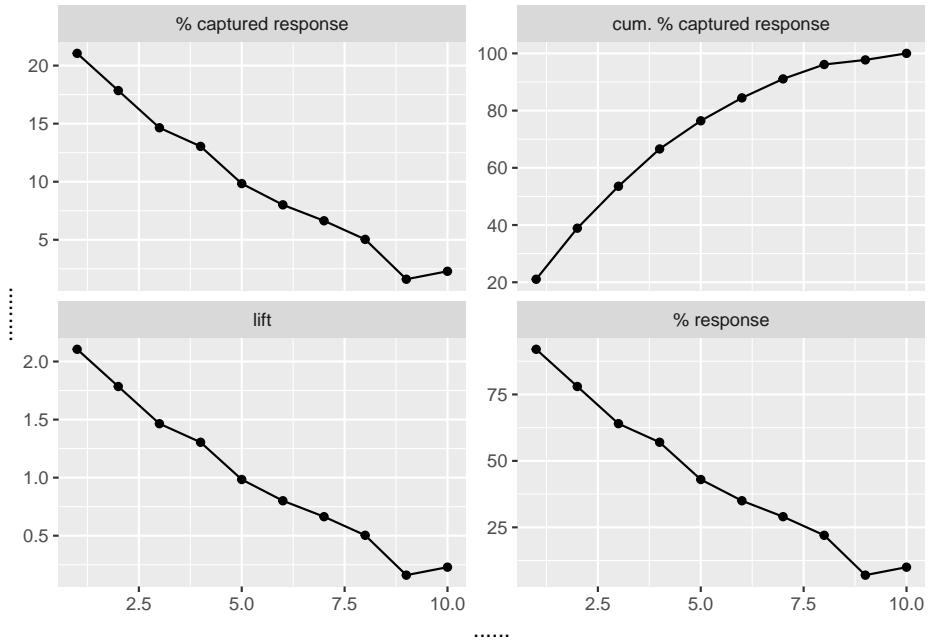


Figure 10.2:

## **Part III**

**3 -**



# Chapter 11

(object) (attribute) ,  
(cluster) .

## 11.1 R

R	
package	version
tidyverse	1.3.1
stats	4.1.0
corr	0.4.3
cluster	2.1.2

## 11.2

$n$ ,  $i$   $O_i$ ,  $S$ .

$$S = \{O_1, \dots, O_n\}$$

$S$ ,  $K$ ,  $C_1, \dots, C_K$ .

$$\begin{aligned}C_i \cap C_j &= \emptyset, 1 \leq i \neq j \leq K \\ \cup_{i=1}^K C_i &= S\end{aligned}$$

,  $C_j$   $j$  (  $j$ ) .  
(clustering result) (clustering solution) .

$$C = \{C_1, \dots, C_K\}$$

- (clustering method) .
- (hierarchical method)
  - (agglomerative method)
  - (divisive method)
- (non-hierarchical method)

## 11.3

### 11.3.1

$$p \quad (\text{variable}) \quad , j \quad i \quad x_{ji} \quad , \quad i \quad p \quad .$$

$$\mathbf{x}_i = [x_{1i} \ x_{2i} \ \dots \ x_{pi}]^\top$$

,     $i$      $j$     .

- (Euclidean distance)

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_j) &= \sqrt{\sum_{a=1}^p (x_{ai} - x_{aj})^2} \\ &= \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)} \end{aligned}$$

- (Manhattan distance)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{a=1}^p |x_{ai} - x_{aj}|$$

- (Minkowski distance)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left( \sum_{a=1}^p |x_{ai} - x_{aj}|^m \right)^{\frac{1}{m}}$$

- (standardized Euclidean distance)

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_j) &= \sqrt{\sum_{a=1}^p \left( \frac{x_{ai} - x_{aj}}{s_a} \right)^2} \\ &= \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{S}_d^{-1} (\mathbf{x}_i - \mathbf{x}_j)} \end{aligned}$$

$$\begin{aligned}\mathbf{S}_d &= \begin{bmatrix} s_1^2 & 0 & \dots & 0 \\ 0 & s_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & s_p^2 \end{bmatrix} \\ s_a &= \sqrt{\frac{\sum_{i=1}^n (x_{ai} - \bar{x}_a)^2}{n-1}} \\ \bar{x}_a &= \frac{1}{n} \sum_{i=1}^n x_{ai}\end{aligned}$$

- (Mahalanobis distance)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j)}$$

$$\begin{aligned}\mathbf{S} &= \begin{bmatrix} s_1^2 & s_{12} & \dots & s_{1p} \\ s_{21} & s_2^2 & \dots & s_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ s_{p1} & s_{p2} & \dots & s_p^2 \end{bmatrix} \\ s_{ab} &= \frac{\sum_{i=1}^n (x_{ai} - \bar{x}_a)(x_{bi} - \bar{x}_b)}{n-1} \\ \bar{x}_a &= \frac{1}{n} \sum_{i=1}^n x_{ai}\end{aligned}$$

$$(n \times n) \quad \mathbf{D} \quad .$$

$$\mathbf{D} = \begin{bmatrix} 0 & d(\mathbf{x}_1, \mathbf{x}_2) & \dots & d(\mathbf{x}_1, \mathbf{x}_n) \\ d(\mathbf{x}_2, \mathbf{x}_1) & 0 & \dots & d(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ d(\mathbf{x}_n, \mathbf{x}_1) & d(\mathbf{x}_n, \mathbf{x}_2) & \dots & 0 \end{bmatrix}$$

$$\text{PC} \quad 10 \quad (x_1), \text{PC} \quad (x_2), \quad (x_3) \quad .$$

```
df <- tribble(
  ~id, ~x1, ~x2, ~x3,
  1, 20, 6, 14,
  2, 28, 8, 13,
  3, 42, 14, 6,
```

Table 11.1: PC

	(\$x_1\$)	PC	(\$x_2\$)	(\$x_3\$)
1	20		6	14
2	28		8	13
3	42		14	6
4	35		12	7
5	30		15	7
6	30		7	15
7	45		13	6
8	46		4	2
9	51		3	3
10	41		3	2

```

4, 35, 12, 7,
5, 30, 15, 7,
6, 30, 7, 15,
7, 45, 13, 6,
8, 46, 4, 2,
9, 51, 3, 3,
10, 41, 3, 2
)

df %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r'),
    col.names = c(' ', ' ($x_1$)', 'PC ($x_2$)', ' ($x_3$)'),
    caption = 'PC '
  )

R dist
      .
      2   4, 5
dist(df[, c("x1", "x2", "x3")], upper = TRUE) %>%
  broom::tidy() %>%
  filter(
    item1 == 2,
    item2 %in% c(4, 5)
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c('item1', 'item2', 'value')
  )

```

Table 11.2:

(from)	(to)	
2	4	10.049876
2	5	9.433981

Table 11.3:

(from)	(to)	
2	4	1.663576
2	5	1.954486

```

col.names = c(' (from)', ' (to)', ' ')
caption = ' '
)

,   2      ,   4   5
               scale      dist
dist(scale(df[, c("x1", "x2", "x3")]), upper = TRUE) %>%
  broom::tidy() %>%
  filter(
    item1 == 2,
    item2 %in% c(4, 5)
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(' (from)', ' (to)', ' '),
    caption = ' '
  )
5   4   2
,       dist      method  p

```

### 11.3.2

$$sim(\mathbf{x}_i, \mathbf{x}_j) = r_{ij} = \frac{\sum_{a=1}^p (x_{ai} - m_i)(x_{aj} - m_j)}{\sqrt{\sum_{a=1}^p (x_{ai} - m_i)^2} \sqrt{\sum_{a=1}^p (x_{aj} - m_j)^2}} \quad (11.1)$$

Table 11.4:

(from)	(to)	
1	6	0.9718362
1	8	-0.8348917

$$m_i = \frac{1}{p} \sum_{a=1}^p x_{ai}$$

(11.1) -1 1 ,

Table 11.1 1 6, 8

```
t(scale(df[, c("x1", "x2", "x3")])) %>%
  corrr::correlate() %>%
  corrr::stretch(na.rm = TRUE) %>%
  mutate(
    x = as.integer(gsub("V", "", x)),
    y = as.integer(gsub("V", "", y)))
  ) %>%
  filter(
    x == 1,
    y %in% c(6, 8)
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(' (from)', ' (to)', ''),
    caption = ''
  )

## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

$$d(\mathbf{x}_i, \mathbf{x}_j) = 1 - r_{ij}$$

```
t(scale(df[, c("x1", "x2", "x3")])) %>%
  corrr::correlate() %>%
  corrr::stretch(na.rm = TRUE) %>%
```

Table 11.5:

(from)	(to)	
1	6	0.0281638
1	8	1.8348917

```

mutate(
  x = as.integer(gsub("V", "", x)),
  y = as.integer(gsub("V", "", y)),
  d = 1 - r
) %>%
select(-r) %>%
filter(
  x == 1,
  y %in% c(6, 8)
) %>%
knitr::kable(
  booktabs = TRUE,
  align = c('r', 'r', 'r'),
  col.names = c(' (from)', ' (to)', ''),
  caption = ''
)

## 
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'

```

## 11.4

( ) . . . (binary), (ordinal),  
 (nominal) . , .

### 11.4.1

, 0 1 . . , (simple matching)  
 (Jaccard) . . .

$\mathbf{x}_i \ \mathbf{x}_j \quad k \quad , \quad .$

$$sim(x_{ki}, x_{kj}) = \begin{cases} 1 & \text{if } x_{ki} = x_{kj} \\ 0 & \text{if } x_{ki} \neq x_{kj} \end{cases}$$

Table 11.6:

	$(\$x\_1\$)$	$(\$x\_2\$)$	$(\$x\_3\$)$	$(\$x\_4\$)$	$(\$x\_5\$)$
1	1	1	1	0	1
2	1	0	1	0	0
3	0	1	0	1	0

$p$ , , .

$$sim(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{p} \sum_{k=1}^p sim(x_{ki}, x_{kj})$$

- (Jaccard)

, (presence) 1, (absence) 0, .

$$sim(x_{ki}, x_{kj}) = \begin{cases} 1 & \text{if } x_{ki} = x_{kj} = 1 \\ \text{ignored} & \text{if } x_{ki} = x_{kj} = 0 \\ 0 & \text{if } x_{ki} \neq x_{kj} \end{cases}$$

,  $p$ , , .

$$sim(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k:x_{ki}+x_{kj}>0} sim(x_{ki}, x_{kj})}{\sum_{k:x_{ki}+x_{kj}>0} 1}$$

3, .

```
df <- tribble(
  ~id, ~x1, ~x2, ~x3, ~x4, ~x5,
  1, 1, 1, 1, 0, 1,
  2, 1, 0, 1, 0, 0,
  3, 0, 1, 0, 1, 0
)

df %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r', 'r', 'r'),
    col.names = c(' ', ' $(\$x\_1\$)', ' $(\$x\_2\$)', ' $(\$x\_3\$)', ' $(\$x\_4\$)', ' $(\$x\_5\$)' ),
    caption = ' '
  )
```

Table 11.7:

(from)	(to)	
1	2	0.6

```

1 2
.
similarity_simplematching <- function(vec_1, vec_2) {
  sum(1 - abs(vec_1 - vec_2)) / length(vec_1)
}

df_pairs <- df %>%
  select(id) %>%
  expand(id_1 = id, id_2 = id) %>%
  filter(id_1 != id_2)

df_pairs$similarity <- df_pairs %>%
  inner_join(df, by=c("id_1" = "id")) %>%
  inner_join(df, by=c("id_2" = "id")) %>%
  rowwise() %>%
  do(similarity = similarity_simplematching(
    .[c("x1.x", "x2.x", "x3.x", "x4.x", "x5.x")] %>% unlist(),
    .[c("x1.y", "x2.y", "x3.y", "x4.y", "x5.y")] %>% unlist())) %>%
  unlist()

df_pairs %>%
  filter(
    id_1 == 1,
    id_2 == 2
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(' (from)', ' (to)', ''),
    caption =
  )
}

dist      . dist      ,  $d(\mathbf{x}_i, \mathbf{x}_j) = 1 - sim(\mathbf{x}_i, \mathbf{x}_j)$ 
. ,  $, sim(\mathbf{x}_i, \mathbf{x}_j) = 1 - d(\mathbf{x}_i, \mathbf{x}_j)$  .

dist(df[, -1], method = "binary", upper = TRUE) %>%
  broom::tidy() %>%
  mutate(similarity = 1 - distance) %>%
  select(-distance) %>%
  filter(

```

Table 11.8:

(from)	(to)	
1	2	0.5

```

item1 == 1,
item2 == 2
) %>%
knitr::kable(
  booktabs = TRUE,
  align = c('r', 'r', 'r'),
  col.names = c(' (from)', ' (to)', ''),
  caption =
)

```

**11.4.2**

$$k \quad 1, 2, \dots, M_k \quad , \quad .$$

$$d(x_{ki}, x_{kj}) = \frac{|x_{ki} - x_{kj}|}{M_k - 1}$$

$$(\text{range}) \quad , \quad 0 \quad 1 \quad . \quad ,$$

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^p d(x_{ki}, x_{kj}) = \sum_{k=1}^p \frac{|x_{ki} - x_{kj}|}{M_k - 1}$$

$$0 \quad 1 \quad , \quad . \quad i \quad k \quad .$$

$$x'_{ki} = \frac{x_{ki} - 1}{M_k - 1}$$

**11.4.3**

$$k \quad , \quad 1, \quad 0 \quad . \quad ,$$

$$sim(x_{ki}, x_{kj}) = \begin{cases} 1 & \text{if } x_{ki} = x_{kj} \\ 0 & \text{if } x_{ki} \neq x_{kj} \end{cases}$$

$$p \quad , \quad .$$

$$sim(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{p} \sum_{k: x_{ki} = x_{kj}} 1$$

#### 11.4.4

$$, , , 0 \quad 1 , . , , 0 \quad 1$$

$$d(x_{ki}, x_{kj}) = \frac{|x_{ki} - x_{kj}|}{R_k}$$

$$R_k \quad k \quad (= - \quad ) \quad .$$

$$sim(x_{ki}, x_{kj}) = 1 - d(x_{ki}, x_{kj})$$

$$sim(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{p} \sum_{k=1}^p sim(x_{ki}, x_{kj})$$

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{p} \sum_{k=1}^p d(x_{ki}, x_{kj})$$

Gower (1971) , R cluster daisy . daisy  
range , , , range

```
df <- tribble(
  ~id, ~x1, ~x2, ~x3, ~x4, ~x5,
  1, "", 46, "", 35000, 2,
  2, "", 28, "", 51000, 3,
  3, "", 32, "", 46000, 4
) %>%
  mutate(
    x1 = factor(x1, levels = c(" ", " ")),
    x3 = factor(x3),
    x5 = factor(x5, levels = c(1:5), ordered = TRUE)
)
```

Table 11.9: Gower

(from)	(to)
--------	------

```

n_obs <- nrow(df)

range_df <- tibble(
  x2 = c(25, 70),
  x4 = c(0, 150000),
  x5 = factor(c(1, 5), levels = c(1:5), ordered = TRUE)
)

df %>%
  bind_rows(range_df) %>%
  select(-id) %>%
  cluster::daisy() %>%
  as.dist() %>%
  broom::tidy() %>%
  filter(
    item1 <= n_obs,
    item2 <= n_obs
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(' (from)', ' (to)', ''),
    caption = ' Gower '
  )

## Warning in Ops.factor(item1, n_obs): '<=' not meaningful for factors
## Warning in Ops.factor(item2, n_obs): '<=' not meaningful for factors

```

# Chapter 12

(DIANA)

## 12.1 R

package	version
tidyverse	1.3.1
stats	4.1.0
cluster	2.1.2

## 12.2

- $C_i$ :  $i \in C_i$
- $|C_i|$ :  $i \in C_i$
- $\mathbf{c}_i = (\bar{x}_1^{(i)}, \bar{x}_2^{(i)}, \dots, \bar{x}_p^{(i)})$ :  $i \in C_i$  (centroid)  $(\bar{x}_a^{(i)} = \frac{1}{|C_i|} \sum_{j \in C_i} x_{aj})$
- $d(u, v) = d(\mathbf{x}_u, \mathbf{x}_v)$ :  $u \in C_i, v \in C_j$  (distance)
- $D(C_i, C_j)$ :  $i \in C_i, j \in C_j$  (linkage method)

## 12.3

### 12.3.1 R

```
train_df <- tibble(  
  id = c(1:10),
```

Table 12.1:

	$\$D(C_i, C_j) \$$
(single linkage method)	$\$\\min_{\{u \in C_i, v \in C_j\}} d(u, v) \$$
(complete linkage method)	$\$\\max_{\{u \in C_i, v \in C_j\}} d(u, v) \$$
(average linkage method)	$\$\\frac{1}{ C_i  C_j } \\sum_{\{u \in C_i, v \in C_j\}} d(u, v) \$$
(centroid linkage method)	$\$d(\\mathbf{c}_i, \\mathbf{c}_j) \$$

Table 12.2: PC

PC	$(, \$x_1\$)$	$(, \$x_2\$)$
1	6	14
2	8	13
3	14	6
4	11	8
5	15	7
6	7	15
7	13	6
8	5	4
9	3	3
10	3	2

```

x1 = c(6, 8, 14, 11, 15, 7, 13, 5, 3, 3,
x2 = c(14,13, 6, 8, 7, 15, 6, 4, 3, 2)
)

knitr::kable(train_df, booktabs = TRUE,
            align = c('r', 'r', 'r'),
            col.names = c(' ', 'PC (, $x_1$)', ' (, $x_2$)'),
            caption = 'PC ')

```

```

theme_set(theme_gray(base_family='NanumGothic'))
ggplot(train_df, aes(x = x1, y = x2)) +
  geom_text(aes(label = id)) +
  xlab("PC ") +
  ylab(" ")

```

Table 12.2 10 ( ) PC PC . . . , Figure 12.1  
 $\{(1, 2, 6), (3, 4, 5, 7), (8, 9, 10)\}$

R  
 1. stats dist

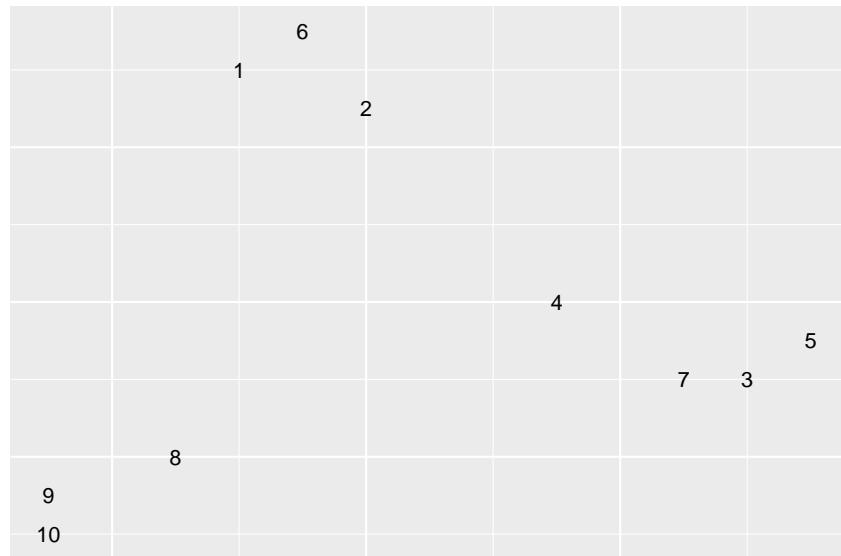


Figure 12.1: PC

```

2. 1           stats   hclust           . ,   method   "average"
.
dist(train_df[, -1]) %>%
  hclust(method = "average") %>%
  plot(
    main = NULL,
    ylab = "distance",
    xlab = "observation"
  )

```

### 12.3.2

```

,
0. 0:
  1.
  2.
  3.  $k \leftarrow n$ 
1. 1:
  1.            $D(C_i, C_j)$    ,    $i \neq j$ 
  2.  $k \leftarrow k - 1$ 
2. 2:  $k = 1$  Stop,   1 .

```

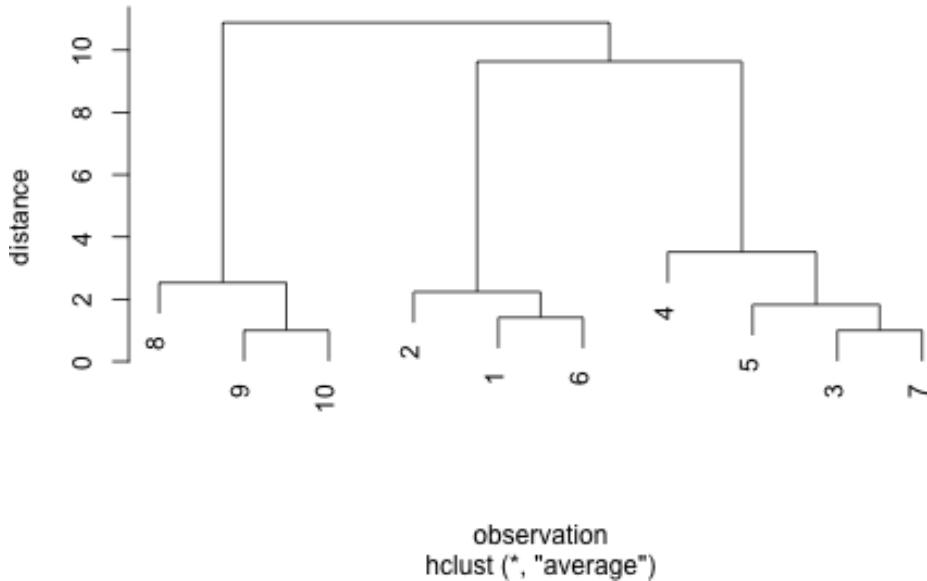


Figure 12.2: PC

```

1      n
iteration <- vector("list", length = nrow(train_df))

      , 1           merge_cluster
• df: . . . id   , . . .
• cluster_label: . . . id   , cluster
      ,
      ,
• cluster_dist: . . .
• closest_clusters: . . . item1 item2   ,
distance
• new_cluster_label: closest_clusters

merge_cluster <- function(df, cluster_label) {
# . . .
cluster_dist <- dist(subset(df, select = -id), upper = TRUE) %>%
  broom::tidy() %>%
  mutate_if(is.factor, ~ as.integer(as.character(.))) %>%
  inner_join(
    cluster_label %>% rename(
      item1 = id, cluster1 = cluster
    ),
}

```

```

    by = "item1") %>%
inner_join(
  cluster_label %>% rename(
    item2 = id, cluster2 = cluster
  ),
  by = "item2") %>%
filter(cluster1 != cluster2) %>%
group_by(cluster1, cluster2) %>%
summarize(distance = mean(distance)) %>%
ungroup()

# .
closest_clusters <- cluster_dist %>%
arrange(distance) %>%
slice(1)

# .
cluster_label[
  cluster_label$cluster %in% (
    closest_clusters[, c("cluster1", "cluster2")] %>% unlist()
  ),
  "cluster"
] <- paste(
  closest_clusters[, c("cluster1", "cluster2")] %>% unlist(),
  collapse = ","
)

list(cluster_dist = cluster_dist,
     closest_clusters = closest_clusters,
     new_cluster_label = cluster_label)
}

0 .
init_cluster <- tibble(
  id = train_df$id,
  cluster = as.character(1:nrow(train_df))
)

print(unique(init_cluster$cluster))

## [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
k <- length(unique(init_cluster$cluster))

print(k)

```

```

## [1] 10
,
          10 .
1
iteration[[1]] <- merge_cluster(train_df, init_cluster)

## `summarise()` has grouped output by 'cluster1'. You can override using the `groups` argument

iteration[[1]]$closest_cluster

## # A tibble: 1 x 3
##   cluster1 cluster2 distance
##   <chr>     <chr>      <dbl>
## 1 10         9           1

iteration[[1]]$new_cluster_label

## # A tibble: 10 x 2
##       id cluster
##   <int> <chr>
## 1 1     1 1
## 2 2     2 2
## 3 3     3 3
## 4 4     4 4
## 5 5     5 5
## 6 6     6 6
## 7 7     7 7
## 8 8     8 8
## 9 9     9 10,9
## 10 10    10 10,9
9 .     1 ,     1 .
iteration[[2]] <- merge_cluster(
  train_df,
  iteration[[1]]$new_cluster_label
)

## `summarise()` has grouped output by 'cluster1'. You can override using the `groups` argument

iteration[[2]]$closest_cluster

## # A tibble: 1 x 3
##   cluster1 cluster2 distance
##   <chr>     <chr>      <dbl>
## 1 10         9           1

```

```
##   <chr>   <chr>      <dbl>
## 1 3       7           1

iteration[[2]]$new_cluster_label

## # A tibble: 10 x 2
##       id cluster
##   <int> <chr>
## 1     1 1
## 2     2 2
## 3     3 3,7
## 4     4 4
## 5     5 5
## 6     6 6
## 7     7 3,7
## 8     8 8
## 9     9 10,9
## 10    10 10,9

1     .

iteration[[3]] <- merge_cluster(
  train_df,
  iteration[[2]]$new_cluster_label
)

## `summarise()` has grouped output by 'cluster1'. You can override using the `.`groups` argument.
print(iteration[[3]]$closest_cluster)

## # A tibble: 1 x 3
##   cluster1 cluster2 distance
##   <chr>     <chr>      <dbl>
## 1 1         6          1.41

print(iteration[[3]]$new_cluster_label)

## # A tibble: 10 x 2
##       id cluster
##   <int> <chr>
## 1     1 1,6
## 2     2 2
## 3     3 3,7
## 4     4 4
## 5     5 5
## 6     6 1,6
## 7     7 3,7
## 8     8 8
```

```

## 9      9 10,9
## 10     10 10,9

# 0
init_cluster <- tibble(
  id = train_df$id,
  cluster = as.character(1:nrow(train_df))
)

i <- 0L
current_clusters <- unique(init_cluster$cluster)
k <- length(current_clusters)

print_clusters <- function(i, k, clusters) {
  cat("Iteration: ", i, ", k = ", k, ", clusters = ", paste0("{", clusters, "}"), "\n")
}

print_clusters(i, k, current_clusters)

## Iteration: 0 , k = 10 , clusters = {1} {2} {3} {4} {5} {6} {7} {8} {9} {10}
# 1
iteration <- vector("list", length = nrow(train_df) - 1)
while(k > 1) {
  i <- i + 1
  if(i == 1) {
    iteration[[i]] <- merge_cluster(
      train_df,
      init_cluster
    )
  } else {
    iteration[[i]] <- merge_cluster(
      train_df,
      iteration[[i-1]]$new_cluster_label
    )
  }
}

current_clusters <- unique(iteration[[i]]$new_cluster_label$cluster)
k <- length(current_clusters)

print_clusters(i, k, current_clusters)
}

## Iteration: 1 , k = 9 , clusters = {1} {2} {3} {4} {5} {6} {7} {8} {10,9}
## Iteration: 2 , k = 8 , clusters = {1} {2} {3,7} {4} {5} {6} {8} {10,9}
## Iteration: 3 , k = 7 , clusters = {1,6} {2} {3,7} {4} {5} {8} {10,9}

```

```
## Iteration: 4 , k = 6 , clusters = {1,6} {2} {3,7,5} {4} {8} {10,9}
## Iteration: 5 , k = 5 , clusters = {1,6,2} {3,7,5} {4} {8} {10,9}
## Iteration: 6 , k = 4 , clusters = {1,6,2} {3,7,5} {4} {10,9,8}
## Iteration: 7 , k = 3 , clusters = {1,6,2} {3,7,5,4} {10,9,8}
## Iteration: 8 , k = 2 , clusters = {1,6,2,3,7,5,4} {10,9,8}
## Iteration: 9 , k = 1 , clusters = {1,6,2,3,7,5,4,10,9,8}
```

### 12.3.3 R

```
R stats hclust
, dist .
distance_matrix <- dist(train_df[, -1])

, hclust . , method , .
cluster_solution <- hclust(distance_matrix, method = "average")

cluster_solution (components) (list) .
names(cluster_solution)

## [1] "merge"      "height"       "order"        "labels"       "method"
## [6] "call"        "dist.method"

, merge 2 n - 1 , 1 iteration .
cluster_solution$merge

##      [,1] [,2]
## [1,]    -3   -7
## [2,]   -9  -10
## [3,]   -1   -6
## [4,]   -5    1
## [5,]   -2    3
## [6,]   -8    2
## [7,]   -4    4
## [8,]    5    7
## [9,]    6    8

iteration , 0 . 0 iteration
. , 6 (-8, 2) 8 iteration ( 9 10 )
( 8, 9, 10) .

height iteration , Figure 12.2 . Iteration
iteration

cluster_solution$height

## [1] 1.000000 1.000000 1.414214 1.825141 2.236068 2.532248 3.519028
```

Table 12.3:

	$(\$x\_1\$)$	$(\$x\_2\$)$
1	4	15
2	20	13
3	3	13
4	19	4
5	17	17
6	8	11
7	19	12
8	18	6

```
## [8] 9.635217 10.881878
iteration 8 . Figure 12.2      3      2
. , iteration 7   3 .
```

## 12.4

(Ward's method)

### 12.4.1 R

Table 12.3 8

```
train_df <- tibble(
  id = c(1:8),
  x1 = c(4, 20, 3, 19, 17, 8, 19, 18),
  x2 = c(15, 13, 13, 4, 17, 11, 12, 6)
)

knitr::kable(train_df, booktabs = TRUE,
             align = c('r', 'r', 'r'),
             col.names = c(' ', '($x_1$)', '($x_2$)'),
             caption = ' ')
```

```
hclust , method "ward.D2"
dist(train_df[, -1]) %>%
  hclust(method = "ward.D2") %>%
  plot(
    main = NULL,
    xlab = "observation"
  )
```

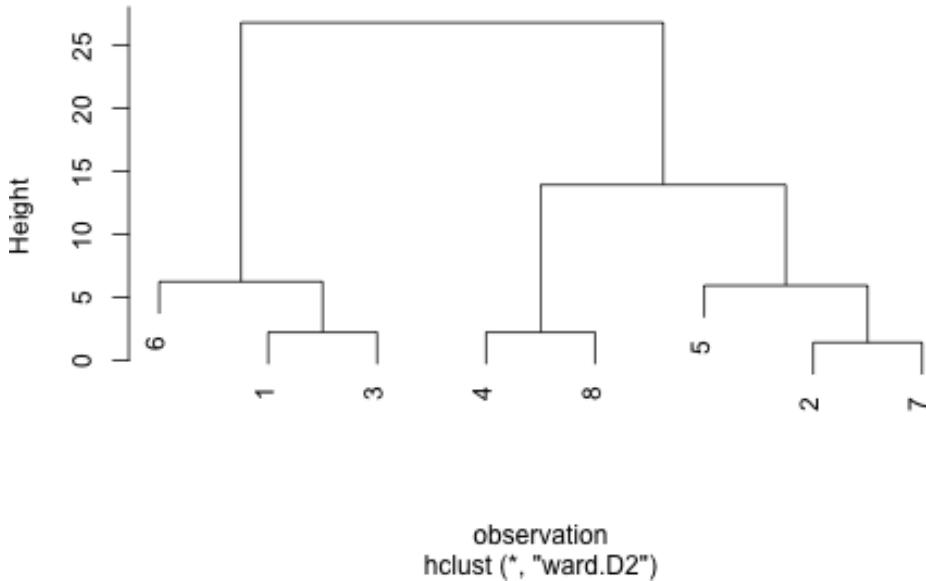


Figure 12.3:

**12.4.2**

$$\mathbf{C} = \{C_1, C_2, \dots, C_k\} \quad , \quad C_i \quad (\text{within sum of squares})$$

$$SS(C_i) = \sum_{u \in C_i} (\mathbf{x}_u - \mathbf{c}_i)^\top (\mathbf{x}_u - \mathbf{c}_i)$$

,       $SSW$       ,      .

$$SSW = \sum_{i=1}^k SS(C_i)$$

,       $SSW$       ,      .

1. 0

1.      .

2.  $k \leftarrow n$

2. 1

1.       $(SSW)$       ,       $i \quad j$       ,

2.  $k \leftarrow k - 1$

3. 2:  $k = 1$  Stop,

R script      .      ,       $SSW$       calculate\_ssw      df  
cluster\_label      .

```

• df: . . . id . . . , . . .
• cluster_label: . . . id . . . , cluster . . .

,
# SSW
calculate_ssw <- function(df, cluster_label) {
  df %>%
    inner_join(cluster_label, by = "id") %>%
    group_by(cluster) %>%
    select(-id) %>%
    summarize_all(function(x) sum((x - mean(x))^2)) %>%
    ungroup() %>%
    mutate(ss = rowSums(subset(., select = -cluster))) %>%
    `[[`("ss") %>%
    sum()
}

SSW . . ,
generate_clusters . . .

generate_clusters cluster_label . . .
calculate_ssw . . . , (list) . . .

#
generate_clusters <- function(cluster_label) {
  unique_clusters <- unique(cluster_label$cluster)

  potential_pairs <- crossing(cluster1 = unique_clusters,
                               cluster2 = unique_clusters) %>%
    filter(cluster1 < cluster2) %>%
    mutate(cluster = paste(cluster1, cluster2, sep = ","))

  candidate_solutions <- potential_pairs %>%
    rowwise() %>%
    do(candidate_solution = merge_cluster(cluster_label, .)) %>%
    `[[`("candidate_solution")

  candidate_solutions
}

candidate_solutions
}

, generate_clusters merge_cluster . . .
cluster_label cluster_merge , cluster_label . . .
cluster_merge . . .

• cluster_merge: 3 character . element cluster_label
, element . .
, cluster_label cluster_merge[1] cluster_merge[2] . .
cluster_merge[3] . .

```

```

#           cluster_merge
merge_cluster <- function(cluster_label, cluster_merge) {
  idx <- cluster_label$cluster %in% cluster_merge[1:2]
  cluster_label[idx, "cluster"] <- cluster_merge[3]
  cluster_label
}

,                      best_merge_cluster

1. generate_clusters
2. 1      calculate_ssw   SSW
3. SSW

#
best_merge_cluster <- function(df, cluster_label) {
  candidate_solutions <- generate_clusters(cluster_label)
  ssw <- sapply(candidate_solutions, function(x) calculate_ssw(df, x))
  list(
    new_cluster_label = candidate_solutions[[which.min(ssw)]],
    new_ssw = min(ssw)
  )
}

```

Table 12.3

```

# 0
init_cluster <- tibble(
  id = train_df$id,
  cluster = as.character(1:nrow(train_df))
)
i <- 0L
current_clusters <- unique(init_cluster$cluster)
k <- length(current_clusters)
ssw <- calculate_ssw(train_df, init_cluster)

print_clusters <- function(i, k, clusters, ssw) {
  cat("Iteration: ", i, ", k = ", k, ", clusters = ", paste0("{", clusters, "}"), ", SSW = ", ssw,
}

print_clusters(i, k, current_clusters, ssw)

## Iteration:  0 , k =  8 , clusters = {1} {2} {3} {4} {5} {6} {7} {8} , SSW = 0
# 1
iteration <- vector("list", length = nrow(train_df) - 1)
while(k > 1) {
  i <- i + 1
  if(i == 1) {

```

```

iteration[[i]] <- best_merge_cluster(
  train_df,
  init_cluster
)
} else {
  iteration[[i]] <- best_merge_cluster(
    train_df,
    iteration[[i-1]]$new_cluster_label
  )
}

current_clusters <- unique(iteration[[i]]$new_cluster_label$cluster)
k <- length(current_clusters)
ssw <- iteration[[i]]$new_ssw

print_clusters(i, k, current_clusters, ssw)
}

## Iteration: 1 , k = 7 , clusters = {1} {2,7} {3} {4} {5} {6} {8} , SSW = 1
## Iteration: 2 , k = 6 , clusters = {1,3} {2,7} {4} {5} {6} {8} , SSW = 3.5
## Iteration: 3 , k = 5 , clusters = {1,3} {2,7} {4,8} {5} {6} , SSW = 6
## Iteration: 4 , k = 4 , clusters = {1,3} {2,7,5} {4,8} {6} , SSW = 23.66667
## Iteration: 5 , k = 3 , clusters = {1,3,6} {2,7,5} {4,8} , SSW = 43.16667
## Iteration: 6 , k = 2 , clusters = {1,3,6} {2,7,5,4,8} , SSW = 140.4
## Iteration: 7 , k = 1 , clusters = {1,3,6,2,7,5,4,8} , SSW = 499.875

```

### 12.4.3 R

R                   $SSW$                   metric                  .                  .  
 Ward Jr (1963)     $ESS$ (error sum of squares)                  ,                   $SSW$                   .

$$\begin{aligned}
 ESS(\{C_1, \dots, C_k\}) &= \sum_{i=1}^k ESS(C_i) \\
 &= \sum_{i=1}^k \sum_{u \in C_i} \mathbf{x}_u^\top \mathbf{x}_u - |C_i| \mathbf{c}_i^\top \mathbf{c}_i \\
 &= SSW
 \end{aligned}$$

$$\frac{C_i, C_j}{C_i \ C_j} \cdot SSW, \quad \frac{C_i \ C_j}{|C_i| \ |C_j|} \cdot \frac{SSW}{|\mathbf{c}_i| \ |\mathbf{c}_j|}, \quad SSW$$

$$\begin{aligned}
\Delta SSW &= ESS(C_i \cup C_j) - ESS(C_i) - ESS(C_j) \\
&= \sum_{u \in C_i \cup C_j} \mathbf{x}_u^\top \mathbf{x}_u - (|C_i| + |C_j|) \left[ \frac{|C_i|\mathbf{c}_i + |C_j|\mathbf{c}_j}{|C_i| + |C_j|} \right]^\top \left[ \frac{|C_i|\mathbf{c}_i + |C_j|\mathbf{c}_j}{|C_i| + |C_j|} \right] \\
&\quad - \left( \sum_{u \in C_i} \mathbf{x}_u^\top \mathbf{x}_u - |C_i|\mathbf{c}_i^\top \mathbf{c}_i \right) - \left( \sum_{u \in C_j} \mathbf{x}_u^\top \mathbf{x}_u - |C_j|\mathbf{c}_j^\top \mathbf{c}_j \right) \\
&= -\frac{1}{|C_i| + |C_j|} (|C_i|\mathbf{c}_i + |C_j|\mathbf{c}_j)^\top (|C_i|\mathbf{c}_i + |C_j|\mathbf{c}_j) + |C_i|\mathbf{c}_i^\top \mathbf{c}_i + |C_j|\mathbf{c}_j^\top \mathbf{c}_j \\
&= \frac{|C_i||C_j|}{|C_i| + |C_j|} (\mathbf{c}_i - \mathbf{c}_j)^\top (\mathbf{c}_i - \mathbf{c}_j)
\end{aligned} \tag{12.1}$$

iteration (12.1)  $C_i, C_j$   
 $, SS(C_i)$   $C_i$

$$\begin{aligned}
D^2(C_i) &= \sum_{u,v \in C_i} (\mathbf{x}_u - \mathbf{x}_v)^\top (\mathbf{x}_u - \mathbf{x}_v) \\
&= \sum_{u,v \in C_i} ((\mathbf{x}_u - \mathbf{c}_i) - (\mathbf{x}_v - \mathbf{c}_i))^\top ((\mathbf{x}_u - \mathbf{c}_i) - (\mathbf{x}_v - \mathbf{c}_i)) \\
&= 2 \sum_{u \in C_i} (\mathbf{x}_u - \mathbf{c}_i)^\top (\mathbf{x}_u - \mathbf{c}_i) - 2 \sum_{u,v \in C_i} (\mathbf{x}_u - \mathbf{c}_i)^\top (\mathbf{x}_v - \mathbf{c}_i) \\
&= 2 \sum_{u \in C_i} (\mathbf{x}_u - \mathbf{c}_i)^\top (\mathbf{x}_u - \mathbf{c}_i) \\
&= 2SS(C_i)
\end{aligned} \tag{12.2}$$

(12.2) ,  $i$  (submatrix)  
 $2SS(C_i)$  . iteration SSW

```

R   stats  hclust      method      "ward.D"  "ward.D2"
,   R
method "ward.D2" , dist
res_ward.D2 <- dist(train_df[, -1]) %>%
  hclust(method = "ward.D2")
, res_ward.D2 criterion height (component)
SSW .

```

Table 12.4: hclust ward.D2 height SSW

iteration	\$height\$	\$\Delta SSW = \frac{1}{2} height^2\$	\$SSW = \sum \Delta SSW\$
1	1.414214	1.00000	1.00000
2	2.236068	2.50000	3.50000
3	2.236068	2.50000	6.00000
4	5.944185	17.66667	23.66667
5	6.244998	19.50000	43.16667
6	13.945131	97.23333	140.40000
7	26.813243	359.47500	499.87500

```
res_ward.D2$height
```

```
## [1] 1.414214 2.236068 2.236068 5.944185 6.244998 13.945131 26.813243
height      SSW , i j      SSW ,
```

$$height = \sqrt{D^2(C_i \cup C_j) - (D^2(C_i) + D^2(C_j))} \quad (12.3)$$

```
, i j      SSW      ΔSSW .
```

$$\Delta SSW = \frac{1}{2} height^2 \quad (12.4)$$

```
iteration      ΔSSW      12.4.2      SSW .
tibble(
  iteration = c(1:(nrow(train_df) - 1)),
  height = res_ward.D2$height
) %>%
  mutate(
    delta_ssw = height ^ 2 / 2
  ) %>%
  mutate(
    ssw = cumsum(delta_ssw)
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r'),
    col.names = c('iteration', '$height$', '$\Delta SSW = \frac{1}{2} height^2$', 'caption = 'hclust ward.D2 height SSW '),
    caption = 'hclust ward.D2 height SSW '
  )
```

```

method "ward.D"      , dist           "ward.D2"    height
,          criterion .
res_ward.D <- dist(train_df[, -1]) %>%
  hclust(method = "ward.D")

res_ward.D$height

## [1] 1.414214 2.236068 2.236068 6.452039 6.615990 17.358484 39.311447
"ward.D2"           , "ward.D" .
Lance and Williams (1967)   i j   ,
.   Lance-Williams update .

```

$$D(C_i \cup C_j, C_{h \notin \{i,j\}}) = \alpha_i D(C_i, C_h) + \alpha_j D(C_j, C_h) + \beta D(C_i, C_j) + \gamma |D(C_i, C_h) - D(C_j, C_h)| \quad (12.5)$$

Wishart (1969)      Lance-Williams update .

$$\begin{aligned} \alpha_i &= \frac{|C_i| + |C_h|}{|C_i| + |C_j| + |C_h|} \\ \alpha_j &= \frac{|C_j| + |C_h|}{|C_i| + |C_j| + |C_h|} \\ \beta &= -\frac{|C_h|}{|C_i| + |C_j| + |C_h|} \\ \gamma &= 0 \end{aligned} \quad (12.6)$$

```

, (12.6) (12.5) D .
"ward.D" , .
Lance-Williams update . , height , criterion .
"ward.D" , height
res_ward.D <- dist(train_df[, -1])^2 %>%
  hclust(method = "ward.D")

res_ward.D$height

## [1] 2.00000 5.00000 5.00000 35.33333 39.00000 194.46667 718.95000
height "ward.D2" . (square root) "ward.D2" height
.
```

Table 12.5: hclust ward.D height SSW

iteration	\$height\$	$\$\\Delta SSW = \\frac{1}{2} height$$	\$SSW = \\sum \\Delta SSW\$
1	2.00000	1.00000	1.00000
2	5.00000	2.50000	3.50000
3	5.00000	2.50000	6.00000
4	35.33333	17.66667	23.66667
5	39.00000	19.50000	43.16667
6	194.46667	97.23333	140.40000
7	718.95000	359.47500	499.87500

```

sqrt(res_ward.D$height)

## [1] 1.414214 2.236068 2.236068 5.944185 6.244998 13.945131 26.813243

      "ward.D"           criterion height 2ΔSSW , iteration
 $\sum_i D(C_i)$  . ( (12.2) )
tibble(
  iteration = c(1:(nrow(train_df) - 1)),
  height = res_ward.D$height
) %>%
  mutate(
    delta_ssw = height / 2
  ) %>%
  mutate(
    ssw = cumsum(delta_ssw)
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r', 'r'),
    col.names = c('iteration', '$height$', '$\\Delta SSW = \\frac{1}{2} height$', '$SSW$'),
    caption = 'hclust ward.D height SSW '
  )

, "ward.D2" "ward.D" (ward.D2) (ward.D) .
, cluster agnes , method "ward" hclust
"ward.D2" .

res_agnes_ward <- cluster::agnes(train_df[, -1], method = "ward")

sort(res_agnes_ward$height)

## [1] 1.414214 2.236068 2.236068 5.944185 6.244998 13.945131 26.813243

```

Table 12.6: DIANA

	$\$x\_1\$$	$\$x\_2\$$
1	30	15
2	45	22
3	25	12
4	40	24
5	50	25
6	20	10
7	42	9

## 12.5

, Kaufman and Rousseeuw (1990)

### 12.5.1 R

```
train_df <- tibble(
  id = c(1:7),
  x1 = c(30, 45, 25, 40, 50, 20, 42),
  x2 = c(15, 22, 12, 24, 25, 10, 9)
)

knitr::kable(train_df, booktabs = TRUE,
             align = c('r', 'r', 'r'),
             col.names = c(' ', '$x_1$', '$x_2$'),
             caption = 'DIANA')
```

Table 12.6       $x_1, x_2$       7      DIANA      cluster      diana

```
res_diana <- cluster::diana(train_df[, -1])
cluster::pltree(res_diana,
                main = NULL,
                xlab = "observation")
```

### 12.5.2

,

,

$i \in C$ .

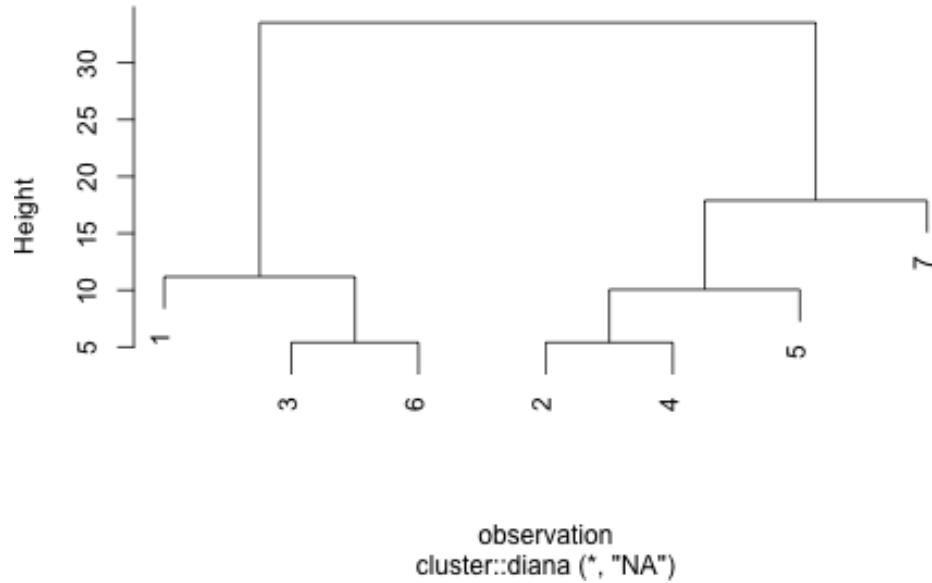


Figure 12.4: DIANA

$$\bar{d}(i, C) = \begin{cases} \frac{1}{|C|-1} \sum_{j \in C} d(i, j) & \text{if } i \in C \\ \frac{1}{|C|} \sum_{j \in C} d(i, j) & \text{if } i \notin C \end{cases}$$

1. 0:  $n$  . ( $k = 1$ )
2. 1: . (  $A$  ,  $B \leftarrow \emptyset$  . )
3. 2:  $A$  .
  1. 2-1:  $i \leftarrow \arg \max_{i'} \bar{d}(i', A)$
  2. 2-2:  $A \leftarrow A - \{i\}$ ,  $B \leftarrow B \cup \{i\}$
  3. 2-3:  $i \leftarrow \arg \max_{i' \in A} e(i') = \bar{d}(i', A) - \bar{d}(i', B)$
  4. 2-4:  $e(i) > 0$  2-2 ,  $e(i) \leq 0$  3
4. 3
  1.  $k \leftarrow k + 1$
  2.  $k < n - 1$  ,  $k = n$  Stop.

DIANA R script .

```
, 1      max_distance_cluster .
.
.
- df: .
* id: .
* :
- cluster_label:
```

```

* id:
* cluster:
•
- cluster:
- distance:

max_distance_cluster <- function(df, cluster_label) {
  unique_cluster <- unique(cluster_label$cluster)

  cluster_df <- lapply(unique_cluster, function(x) {
    cluster_label %>%
      filter(cluster == x) %>%
      inner_join(df, by = "id") %>%
      select(-cluster, -id)
  })

  max_distance <- sapply(cluster_df,
    function(x) {
      if(nrow(x) == 1) return(0)
      max(dist(x))
    }
  )

  list(
    cluster = unique_cluster[which.max(max_distance)],
    distance = max(max_distance)
  )
}

2-1          max_within_distance .       cluster_df
,           id . .

max_within_distance <- function(cluster_df) {
  idx <- dist(subset(cluster_df, select = -id), upper = TRUE) %>%
    broom::tidy() %>%
    group_by(item1) %>%
    summarize(mean_distance = mean(distance)) %>%
    ungroup() %>%
    arrange(-mean_distance) %>%
    .[["item1"]] %>%
    .[1]

  cluster_df$id[idx]
}

2-3      e(i') = d̄(i', A) - d̄(i', B)      e_score .
• object: (id)

```

```

• A(B):   A(B) . . . , id . . .
e_score <- function(object, A, B) {
  d_from_A <- proxy::dist(subset(A, id == object, -id),
                           subset(A, id != object, -id)) %>%
    mean()
  d_from_B <- proxy::dist(subset(A, id == object, -id),
                           B %>% select(-id)) %>%
    mean()
  return(d_from_A - d_from_B)
}

max_within_distance e_score , split_cluster

• : cluster_df . . . id
• : component .
  - idx_A: A
  - idx_B: B

split_cluster <- function(cluster_df) {
  n <- nrow(cluster_df)

  idx_A <- cluster_df$id
  idx_B <- NULL

  # 2-1
  max_object <- max_within_distance(cluster_df)
  e_i <- Inf

  while(e_i > 0) {
    # 2-2
    idx_B <- c(idx_B, max_object)
    idx_A <- setdiff(idx_A, max_object)

    A <- cluster_df %>% filter(id %in% idx_A)
    B <- cluster_df %>% filter(id %in% idx_B)

    # 2-3
    if(nrow(A) > 1) {
      e_is <- sapply(A$id, function(x) e_score(x, A, B))
      max_object <- A$id[which.max(e_is)]
      e_i <- max(e_is)
    } else {
      e_i <- -Inf
    }
  }
}

```

```

    return(list(idx_A = idx_A, idx_B = idx_B))
}

  1 max_distance_cluster  2 split_cluster
# 0
current_cluster <- tibble(
  id = train_df$id
)
current_cluster$cluster <- paste(1:nrow(current_cluster), collapse = ",")
i <- 0L
k <- 1L

while(k < nrow(train_df)) {
  i <- i + 1L

  # 1
  max_cluster <- max_distance_cluster(train_df, current_cluster)

  # 2
  new_split <- current_cluster %>%
    filter(cluster == max_cluster$cluster) %>%
    inner_join(train_df, by = "id") %>%
    select(-cluster) %>%
    split_cluster()

  #
  current_cluster[
    current_cluster$id %in% new_split$idx_A,
    "cluster"] <- paste(new_split$idx_A, collapse = ",")
  current_cluster[
    current_cluster$id %in% new_split$idx_B,
    "cluster"] <- paste(new_split$idx_B, collapse = ",")

  #
  k <- length(unique(current_cluster$cluster))
  cat("Iteration: ", i, ", k = ", k, ", clusters = ",
      paste0("{", unique(current_cluster$cluster), "}" ),
      ", height = ", max_cluster$distance, "\n")
}

## Iteration: 1 , k = 2 , clusters = {6,3,1} {2,4,5,7} , height = 33.54102
## Iteration: 2 , k = 3 , clusters = {6,3,1} {2,4,5} {7} , height = 17.88854
## Iteration: 3 , k = 4 , clusters = {1} {2,4,5} {3,6} {7} , height = 11.18034
## Iteration: 4 , k = 5 , clusters = {1} {2,4} {3,6} {5} {7} , height = 10.04988
## Iteration: 5 , k = 6 , clusters = {1} {2} {3,6} {4} {5} {7} , height = 5.385165

```

```
## Iteration:  6 , k =  7 , clusters = {1} {2} {3} {4} {5} {6} {7} , height =  5.385100
      height   iteration      (diameter) , max_distance_cluster
      , R    cluster diana      height . Iteration
height .
```

## 12.6

1. RMS (root-mean-square standard deviation of the new cluster; RMSSTD)

$$RMSSTD(C_i, C_j) = \sqrt{\frac{SS(C_i \cup C_j)}{p(|C_i| + |C_j| - 1)}}$$

2. Semipartial R-squared(SPR)

$$SPR(C_i, C_j) = \frac{SS(C_i \cup C_j) - (SS(C_i) + SS(C_j))}{SST}$$

where

$$SST = \sum_{i=1}^n \sum_{j=1}^p \left( x_{ji} - \frac{1}{n} \sum_{a=1}^n x_{ja} \right)^2$$

3. R-squared( $R^2$ )

$$1 - \frac{\sum_{i=1}^k SS(C_i)}{SST}$$

### 12.4.2

```
train_df <- tibble(
  id = c(1:8),
  x1 = c(4, 20, 3, 19, 17, 8, 19, 18),
  x2 = c(15, 13, 13, 4, 17, 11, 12, 6)
)

sst <- train_df %>%
  select(-id) %>%
  sapply(function(x) sum((x - mean(x))^2)) %>%
  sum()
```

```

# 0
init_cluster <- tibble(
  id = train_df$id,
  cluster = as.character(1:nrow(train_df))
)
i <- 0L
current_clusters <- unique(init_cluster$cluster)
k <- length(current_clusters)
ssw <- calculate_ssw(train_df, init_cluster)
old_ssw <- NA_real_

# 1
iteration <- vector("list", length = nrow(train_df) - 1)
while(k > 1) {
  i <- i + 1
  old_ssw <- ssw

  if(i == 1) {
    old_cluster <- init_cluster
  } else {
    old_cluster <- iteration[[i-1]]$new_cluster_label
  }

  iteration[[i]] <- best_merge_cluster(
    train_df,
    old_cluster
  )

  merged <- old_cluster %>%
    anti_join(iteration[[i]]$new_cluster_label, by = "cluster")

  current_clusters <- unique(iteration[[i]]$new_cluster_label$cluster)
  k <- length(current_clusters)
  ssw <- iteration[[i]]$new_ssw

  iteration[[i]]$rmsstd <- sqrt(
    merged %>%
      inner_join(train_df, by = "id") %>%
      select(-id, -cluster) %>%
      sapply(function(x) sum((x - mean(x))^2)) %>%
      sum() / (2 * (nrow(merged) - 1))
  )

  iteration[[i]]$iter <- i
  iteration[[i]]$merge <- paste0("{", unique(merged$cluster), "}", collapse = ", ")
}

```

Table 12.7:

Iteration		\$RMSSTD\$	\$SPR\$
0	NA	{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}	NA
1	{2}, {7}	{1}, {2,7}, {3}, {4}, {5}, {6}, {8}	0.7071068
2	{1}, {3}	{1,3}, {2,7}, {4}, {5}, {6}, {8}	1.1180340
3	{4}, {8}	{1,3}, {2,7}, {4,8}, {5}, {6}	1.1180340
4	{2,7}, {5}	{1,3}, {2,7,5}, {4,8}, {6}	2.1602469
5	{1,3}, {6}	{1,3,6}, {2,7,5}, {4,8}	2.3452079
6	{2,7,5}, {4,8}	{1,3,6}, {2,7,5,4,8}	3.8470768
7	{1,3,6}, {2,7,5,4,8}	{1,3,6,2,7,5,4,8}	5.9753960

```

iteration[[i]]$sol <- paste0("{", unique(current_clusters), "}", collapse = ", ")
iteration[[i]]$spr <- (ssw - old_ssw) / sst
iteration[[i]]$r_sq <- 1 - ssw / sst
}

cluster_statistic <- lapply(iteration, function(x) x[
  c("iter", "merge", "sol", "rmsstd", "spr", "r_sq")]) %>%
  bind_rows(
    tibble(
      iter = 0,
      sol = paste0("{", unique(init_cluster$cluster), "}", collapse = ", "),
      r_sq = 1
    )
  ) %>%
  arrange(iter)

cluster_statistic %>%
  knitr::kable(
    booktabs = TRUE,
    col.names = c('Iteration', ' ', ' ', '$RMSSTD$', '$SPR$', '$R^2$'),
    caption = ''
  )

cluster_statistic %>%
  mutate(
    rmsstd = if_else(is.na(rmsstd), 0, rmsstd),
    spr = if_else(is.na(spr), 0, spr)
  ) %>%
  ggplot(aes(x = iter)) +
  geom_line(aes(y = rmsstd, color = "RMSSTD")) +

```

```
geom_line(aes(y = spr * 6, color = "SPR")) +  
  geom_line(aes(y = r_sq * 6, color = "R2")) +  
  scale_y_continuous(sec.axis = sec_axis(~ . / 6, name = "SPR, R2")) +  
  ylab("RMSSSTD") +  
  xlab("Iteration")
```

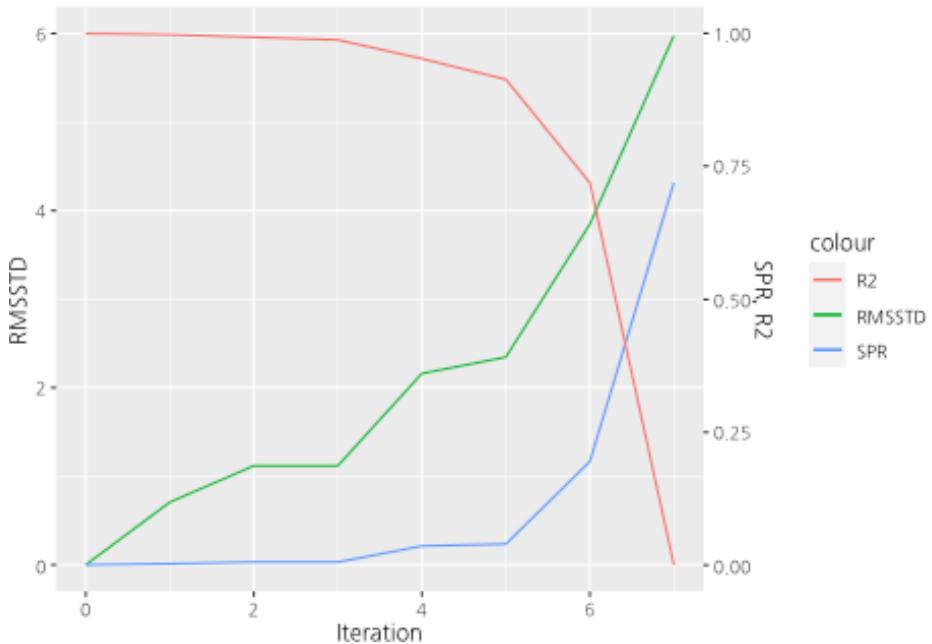


Figure 12.5:

12.5 Iteration 6 3

Iteration 5 3



# Chapter 13

(Nonhierarchical clustering)      (Partitioning method)      .  
K-medoids      ,      K-means      ,      .

## 13.1 R package

package	version
tidyverse	1.3.1
stats	4.1.0
cluster	2.1.2
flexclust	1.4-0
mclust	5.4.7
mvtnorm	1.1-2

## 13.2 K-means

K-means       $K$

### 13.2.1 R

```
10    PC    ( $x_1$ )    ( $x_2$ )    .
df <- tribble(
  ~id, ~x1, ~x2,
  1, 6, 14,
  2, 8, 13,
  3, 14, 6,
```

Table 13.1: PC

	$(\$x\_1\$)$	$(\$x\_2\$)$
1	6	14
2	8	13
3	14	6
4	11	8
5	15	7
6	7	15
7	13	6
8	5	4
9	3	3
10	3	2

```

4, 11, 8,
5, 15, 7,
6, 7, 15,
7, 13, 6,
8, 5, 4,
9, 3, 3,
10, 3, 2
)

df %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(
      '',
      ' $(\$x\_1\$)$ ', ' $(\$x\_2\$)$ '
    ),
    caption = 'PC'
  )

      stats      kmeans      K-means      .
      .          .          .          .
K = 3

set.seed(123)
kmeans_solution <- kmeans(x = df[, -1], centers = 3)

      kmeans      centers      .
kmeans_solution$centers      .

##           x1      x2
## 1 13.250000 6.75

```

```
## 2 3.666667 3.00
## 3 7.000000 14.00

      cluster .
kmeans_solution$cluster

## [1] 3 3 1 1 3 1 2 2 2

df %>%
  mutate(cluster = as.factor(kmeans_solution$cluster)) %>%
  ggplot(aes(x = x1, y = x2)) +
  geom_text(aes(label = id, color = cluster))
```

Figure 13.1: K-means

### 13.2.2

K-means . Table 13.1

$$[\begin{array}{c|ccccc} \mathbf{0} & (\cdot & \cdot & \cdot & \cdot & \cdot) & K \\ \cdot & \mathbf{c}_1, \dots, \mathbf{c}_K & & & & & \\ \hline \cdot & \vdots & & & & & \\ \bullet & \vdots & & & & & K \\ \bullet & \vdots & & & & & K \end{array}] \quad \text{(centroid)} \quad . \quad j \quad \mathbf{c}_j = \left( \bar{x}_1^{(j)}, \dots, \bar{x}_p^{(j)} \right)^T$$

```
[ 0]      Table 13.1   3
set.seed(123)

init_cluster <- function(df, k = 1) {
  k <- min(k, nrow(df))

  df %>%
    sample_n(k)
}

cluster_df <- init_cluster(df[, -1], 3)

cluster_df

## # A tibble: 3 x 2
##       x1     x2
##   <dbl> <dbl>
## 1     14     6
## 2      3     2
## 3      8    13
[ 1] (      )      K      (centroid)  (      )

```

$$a_{ij} = \begin{cases} 1 & \text{if } j = \arg \max_k d(\mathbf{x}_i, \mathbf{c}_k) \\ 0 & \text{otherwise} \end{cases}, i = 1, \dots, n, j = 1, \dots, K$$

```
3
flexclust::dist2(df[, -1], cluster_df)

##      [,1]      [,2]      [,3]
## [1,] 11.313708 12.369317 2.236068
## [2,] 9.219544 12.083046 0.000000
## [3,] 0.000000 11.704700 9.219544
## [4,] 3.605551 10.000000 5.830952
## [5,] 1.414214 13.000000 9.219544
## [6,] 11.401754 13.601471 2.236068
## [7,] 1.000000 10.770330 8.602325
## [8,] 9.219544 2.828427 9.486833
## [9,] 11.401754 1.000000 11.180340
## [10,] 11.704700 0.000000 12.083046
```

```
assign_cluster <- function(df, cluster_df) {
  cluster_ind <- flexclust::dist2(df, cluster_df) %>%
    apply(1, which.min)
```

```

    map(unique(cluster_ind), ~which(cluster_ind == .))
}

cluster_objects <- assign_cluster(df[, -1], cluster_df)

cluster_objects

## [[1]]
## [1] 1 2 6
##
## [[2]]
## [1] 3 4 5 7
##
## [[3]]
## [1] 8 9 10

[ 2] (      ) . . .

```

$$\bar{x}_l^{(j)} = \frac{\sum_i a_{ij} x_{li}}{\sum_i a_{ij}}, l = 1, \dots, p, j = 1, \dots, K$$

```

find_center <- function(df, cluster) {
  map_dfr(cluster, ~df[., ] %>% summarize_all(mean))
}

new_cluster_df <- find_center(df[, -1], cluster_objects)

new_cluster_df

## # A tibble: 3 x 2
##       x1     x2
##   <dbl> <dbl>
## 1    7    14
## 2 13.2   6.75
## 3   3.67    3

[ 3] (      ) , 1 . .

```

```

identical(cluster_df, new_cluster_df)

## [1] FALSE
, iteration . .

```

### 13.2.3 R

R

```

init_cluster <- function(df, k = 1) {
  k <- min(k, nrow(df))

  df %>% sample_n(k)
}

assign_cluster <- function(df, cluster_df) {
  cluster_ind <- flexclust::dist2(df, cluster_df) %>%
    apply(1, which.min)

  map(unique(cluster_ind), ~which(cluster_ind == .))
}

find_center <- function(df, cluster) {
  map_dfr(cluster, ~df[., ] %>% summarize_all(mean))
}

kmeans_cluster <- function(df, k = 1, verbose = FALSE) {
  k <- min(k, nrow(df))

  i <- 0L

  ## 0
  cluster_df <- init_cluster(df, k)

  while(TRUE) {
    i <- i + 1L

    ## 1
    cluster_objects <- assign_cluster(df, cluster_df)
    if (verbose) { #
      cat("Iteration", i, ":", 
          map(cluster_objects, ~ str_c("{", str_c(., collapse = ", "), "}") %>%
            str_c(collapse = ", ", "\n"))
    }
  }

  ## 2
  new_cluster_df <- find_center(df, cluster_objects)

  ## 3
  if(identical(cluster_df, new_cluster_df)) break

  cluster_df <- new_cluster_df
}

```

```
}  
  
res <- list(  
  cluster_centers = cluster_df,  
  assgined_objects = cluster_objects,  
  n_iteration = i  
)  
  
return (res)  
}  
  
set.seed(123)  
  
kmeans_solution <- kmeans_cluster(df[, -1], k = 3, verbose = TRUE)  
  
## Iteration 1 : {1, 2, 6}, {3, 4, 5, 7}, {8, 9, 10}  
## Iteration 2 : {1, 2, 6}, {3, 4, 5, 7}, {8, 9, 10}  
  
2 Iteration , .  
  
kmeans_solution$assgined_objects  
  
## [[1]]  
## [1] 1 2 6  
##  
## [[2]]  
## [1] 3 4 5 7  
##  
## [[3]]  
## [1] 8 9 10
```

### 13.3 K-medoids

K-means (centroid), K-medoids

## K-medoids

1. PAM(Partitioning Around Medoids)
  2. CLARA(Clustering LARge Applications)
  3. CLARANS(Clustering Large Applications based on RANdomized Search)
  4. K-means-like

### 13.3.1 PAM

PAM      Kaufman and Rousseeuw (1990) ,      **BUILD**  
**SWAP** .

Table 13.2: PC

	$(\$x\_1\$)$	$(\$x\_2\$)$
1	3	3
2	5	4
3	11	8
4	13	6
5	14	6
6	15	7

### 13.3.1.1 R

```
df <- tribble(
  ~id, ~x1, ~x2,
  1, 3, 3,
  2, 5, 4,
  3, 11, 8,
  4, 13, 6,
  5, 14, 6,
  6, 15, 7
)

df %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(
      '',
      ' $(\$x\_1\$)$ ', ' $(\$x\_2\$)$ '
    ),
    caption = 'PC'
  )

PAM   cluster      pam
pam_solution <- cluster::pam(df[, -1], k = 2)

pam   id.med
pam_solution$id.med

## [1] 2 5

pam   clustering
```

```
pam_solution$clustering
```

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

### 13.3.1.2 PAM

PAM      Table 13.2

#### 13.3.1.2.1 BUILD BUILD $K$

```
[ 0] , . . M .
pairwise_distance_df <- dist(df[, -1], upper = TRUE) %>%
  broom::tidy() %>%
  mutate_if(is.factor, ~ as.integer(as.character(.)))
```

```
M_idx <- pairwise_distance_df %>%
  group_by(item1) %>%
  summarize(sum_distance = sum(distance)) %>%
  top_n(-1, sum_distance) %>%
  .$item1
```

```
M <- df[M_idx, ]
```

```
M
```

```
## # A tibble: 1 x 3
##       id     x1     x2
##   <dbl> <dbl> <dbl>
## 1     4     13      6
```

```
[ 1] j , j D_j . ,
```

$$D_j = \min_{k \in M} d(j, k), j \notin M$$

```
D_j <- pairwise_distance_df %>%
  filter(
    !item1 %in% M$id,
    item2 %in% M$id
  ) %>%
  group_by(item1) %>%
  top_n(-1, distance) %>%
  ungroup() %>%
  rename(D_j = distance) %>%
  select(-item2)
```

```
D_j
```

```

## # A tibble: 5 x 2
##   item1    D_j
##   <int> <dbl>
## 1     1 10.4
## 2     2  8.25
## 3     3  2.83
## 4     5  1
## 5     6  2.24

```

$i, j$  .

$$C_{ji} = \max(D_j - d(j, i), 0)$$

```

d_ji <- pairwise_distance_df %>%
  filter(
    !item1 %in% M$id,
    !item2 %in% M$id
  )

C_ji <- D_j %>%
  inner_join(d_ji, by = "item1") %>%
  mutate(C_ji = pmax(D_j - distance, 0))

C_ji

```

```

## # A tibble: 20 x 5
##   item1    D_j item2 distance   C_ji
##   <int> <dbl> <int>   <dbl> <dbl>
## 1     1 10.4      2     2.24 8.20
## 2     1 10.4      3     9.43 1.01
## 3     1 10.4      5    11.4  0
## 4     1 10.4      6    12.6  0
## 5     2  8.25     1     2.24 6.01
## 6     2  8.25     3     7.21 1.04
## 7     2  8.25     5     9.22 0
## 8     2  8.25     6    10.4  0
## 9     3  2.83     1     9.43 0
## 10    3  2.83     2     7.21 0
## 11    3  2.83     5     3.61 0
## 12    3  2.83     6     4.12 0
## 13    5  1         1    11.4  0
## 14    5  1         2     9.22 0
## 15    5  1         3     3.61 0
## 16    5  1         6     1.41 0
## 17    6  2.24     1    12.6  0
## 18    6  2.24     2    10.4  0

```

```
## 19      6  2.24      3      4.12 0
## 20      6  2.24      5      1.41 0.822
```

```
i      , j      .
[ 2]      m      ,
```

$$m = \arg \max_{i \notin M} \sum_{j \notin M} C_{ji}$$

```
m <- C_ji %>%
  group_by(item2) %>%
  summarize(sum_C_ji = sum(C_ji)) %>%
  top_n(1, sum_C_ji) %>%
  .$item2
```

```
m
```

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

$$M \leftarrow M \cup \{m\}$$

```
M <- M %>%
  bind_rows(df[m, ])
```

```
[ 3] K      Stop,      [ 1]      .
```

**13.3.1.2.2 SWAP** SWAP       $i$        $h$

```
[ 1] i      h      ,      j      ≠ h      .
```

$$\begin{aligned} C_{jih} &= (i \ h \ j \ ) \\ &\quad - ( \ j \ ), \\ &\quad (j \notin M, i \in M, h \notin M) \end{aligned}$$

```
#  
D_j <- pairwise_distance_df %>%
  filter(
    !item1 %in% M$id,
    item2 %in% M$id
  ) %>%
  group_by(item1) %>%
```

```

top_n(-1, distance) %>%
ungroup() %>%
rename(j = item1) %>%
select(-item2)

D_j

## # A tibble: 4 x 2
##       j   distance
##   <int>     <dbl>
## 1     1     2.24
## 2     3     2.83
## 3     5     1
## 4     6     2.24

#
swap_ids <- tibble(
  i = rep(M$id, each = nrow(df) - nrow(M)),
  h = rep(setdiff(df$id, M$id), nrow(M))
)

#
update_distance <- function(i, h, distance_df, center_ids) {
  new_center_ids <- c(setdiff(center_ids, i), h)

  res <- distance_df %>%
    filter(
      !item1 %in% c(center_ids, h),
      item2 %in% new_center_ids
    ) %>%
    group_by(item1) %>%
    top_n(-1, distance) %>%
    ungroup() %>%
    rename(j = item1) %>%
    select(-item2) %>%
    mutate(
      i = i,
      h = h
    )

  return(res)
}

D_jih <- pmap_dfr(
  swap_ids,
  update_distance,
)

```

```

distance_df = pairwise_distance_df,
center_ids = M$id
)

D_jih

## # A tibble: 24 x 4
##      j distance     i     h
##   <int>    <dbl> <dbl> <dbl>
## 1     3     7.21     4     1
## 2     5     9.22     4     1
## 3     6    10.4      4     1
## 4     1     2.24     4     3
## 5     5     3.61     4     3
## 6     6     4.12     4     3
## 7     1     2.24     4     5
## 8     3     3.61     4     5
## 9     6     1.41     4     5
## 10    1     2.24     4     6
## # ... with 14 more rows
#
C_jih <- D_j %>%
  inner_join(D_jih, by = "j", suffix = c("", "_new")) %>%
  mutate(diff_distance = distance_new - distance)

C_jih

## # A tibble: 24 x 6
##      j distance distance_new     i     h diff_distance
##   <int>    <dbl>       <dbl> <dbl> <dbl>       <dbl>
## 1     1     2.24      2.24     4     3         0
## 2     1     2.24      2.24     4     5         0
## 3     1     2.24      2.24     4     6         0
## 4     1     2.24      9.43     2     3        7.20
## 5     1     2.24     10.4      2     5        8.20
## 6     1     2.24     10.4      2     6        8.20
## 7     3     2.83      7.21     4     1        4.38
## 8     3     2.83      3.61     4     5        0.777
## 9     3     2.83      4.12     4     6        1.29
## 10    3     2.83      2.83     2     1         0
## # ... with 14 more rows

[ 2]   i  h

```

$$T_{ih} = \sum_j C_{jih}$$

```

T_ih <- C_jih %>%
  group_by(i, h) %>%
  summarise(total_diff_distance = sum(diff_distance))

## `summarise()` has grouped output by 'i'. You can override using the `~.groups` argument
T_ih

## # A tibble: 8 x 3
## # Groups:   i [2]
##       i     h total_diff_distance
##   <dbl> <dbl>             <dbl>
## 1     2     1                 0
## 2     2     3                7.20
## 3     2     5                7.38
## 4     2     6                8.20
## 5     4     1               20.8
## 6     4     3                4.49
## 7     4     5              -0.0447
## 8     4     6                1.71

min_{i,h} T_{ih}      i^*   h^*   T_{i^*h^*} < 0      [ 1] , T_{i^*h^*} \geq 0      stop.

swap_ih <- T_ih %>%
  filter(total_diff_distance < 0) %>%
  top_n(-1, total_diff_distance)

swap_ih

## # A tibble: 1 x 3
## # Groups:   i [1]
##       i     h total_diff_distance
##   <dbl> <dbl>             <dbl>
## 1     4     5              -0.0447
## 4     5      , [ 1]  .

M <- M %>%
  anti_join(df[swap_ih$i, ]) %>%
  bind_rows(df[swap_ih$h, ])

## Joining, by = c("id", "x1", "x2")
M

## # A tibble: 2 x 3
##       id     x1     x2

```

```
##   <dbl> <dbl> <dbl>
## 1     2     5     4
## 2     5    14     6
SWAP ,
```

### 13.3.1.3 R

```
#  
distance_from_medoids <- function(distance_df, medoids_ids) {  
  distance_df %>%  
    filter(  
      !item1 %in% medoids_ids,  
      item2 %in% medoids_ids  
    ) %>%  
    group_by(item1) %>%  
    arrange(distance) %>%  
    slice(1) %>%  
    ungroup() %>%  
    rename(j = item1) %>%  
    select(-item2)  
}  
  
# k  
build_medoids <- function(distance_df, k = 1L) {  
  k <- min(k, length(unique(distance_df$item1)))  
  
  #  
  medoids_ids <- distance_df %>%  
    group_by(item1) %>%  
    summarize(sum_distance = sum(distance)) %>%  
    arrange(sum_distance) %>%  
    slice(1) %>%  
    .$item1  
  
  while (length(medoids_ids) < k) {  
    D_j <- distance_from_medoids(distance_df, medoids_ids)  
  
    d_ji <- distance_df %>%  
      filter(  
        !item1 %in% medoids_ids,  
        !item2 %in% medoids_ids  
      ) %>%  
      rename(j = item1, i = item2)
```

```

# C_ji <- D_j %>%
#   inner_join(d_ji, by = "j",
#             suffix = c("", "_new")) %>%
#   mutate(diff_distance = pmax(distance - distance_new, 0))

# m <- C_ji %>%
#   group_by(i) %>%
#   summarize(total_diff_distance = sum(diff_distance)) %>%
#   arrange(desc(total_diff_distance)) %>%
#   slice(1) %>%
#   .$i

# medoids_ids <- c(medoids_ids, m)
}

return(medoids_ids)
}

# update_distance <- function(i, h, distance_df, medoids_ids) {
#   new_medoids_ids <- c(setdiff(medoids_ids, i), h)

#   res <- distance_from_medoids(distance_df, new_medoids_ids) %>%
#     filter(j != h) %>%
#     mutate(
#       i = i,
#       h = h
#     )

#   return(res)
}

#   medoid
swap_medoids <- function(distance_df, medoids_ids) {
  observation_ids <- unique(distance_df$item1)

  # D_j <- distance_from_medoids(distance_df, medoids_ids)

  #
  swap_ids <- tibble(
    i = rep(medoids_ids,

```

```

        each = length(observation_ids) - length(medoids_ids)),
        h = rep(setdiff(observation_ids, medoids_ids),
                length(medoids_ids))
    )

D_jih <- pmap_dfr(
  swap_ids,
  update_distance,
  distance_df = distance_df,
  medoids_ids = medoids_ids
)

#  

C_jih <- D_j %>%
  inner_join(D_jih, by = "j", suffix = c("", "_new")) %>%
  mutate(diff_distance = distance_new - distance)

T_ih <- C_jih %>%
  group_by(i, h) %>%
  summarize(total_diff_distance = sum(diff_distance))

swap_ih <- T_ih %>%
  filter(total_diff_distance < 0) %>%
  arrange(total_diff_distance) %>%
  slice(1)

return(list(remove = swap_ih$i, add = swap_ih$h))
}

#  PAM
pam_medoids <- function(distance_df, k = 1L) {
  # BUILD
  medoids_ids <- build_medoids(distance_df, k)
  k <- length(medoids_ids)

  # SWAP
  while (TRUE) {
    swap_medoids <- swap_medoids(distance_df, medoids_ids)
    if (is_empty(swap_medoids$remove)) {
      break
    } else {
      medoids_ids <- c(
        setdiff(medoids_ids, swap_medoids$remove),
        swap_medoids$add
      )
    }
  }
}

```

```

        }
    }

    return (medoids_ids)
}

PAM . .

pairwise_distance_df <- dist(df[, -1], upper = TRUE) %>%
  broom::tidy() %>%
  mutate_if(is.factor, ~ as.integer(as.character(.)))

medoids_ids <- pam_medoids(pairwise_distance_df, k = 2)

## `summarise()` has grouped output by 'i'. You can override using the `groups` argument
## `summarise()` has grouped output by 'i'. You can override using the `groups` argument
df[medoids_ids, ]

## # A tibble: 2 x 3
##       id     x1     x2
##   <dbl> <dbl> <dbl>
## 1     2      5      4
## 2     5     14      6

2     2, 5   .
.

assign_cluster <- function(distance_df, medoids_ids) {
  distance_df %>%
    filter(
      !item1 %in% medoids_ids,
      item2 %in% medoids_ids
    ) %>%
    group_by(item1) %>%
    arrange(distance) %>%
    slice(1) %>%
    ungroup() %>%
    bind_rows(
      tibble(
        item1 = medoids_ids,
        item2 = medoids_ids,
        distance = 0
      )
    ) %>%
    rename(object = item1) %>%
    arrange(object) %>%

```

```

group_by(item2) %>%
  mutate(cluster = str_c("{", str_c(object, collapse = ", "), "}") ) %>%
  ungroup() %>%
  select(-item2)
}

assign_cluster(pairwise_distance_df, medoids_ids)

## # A tibble: 6 x 3
##   object distance cluster
##     <int>    <dbl> <chr>
## 1       1      2.24 {1, 2}
## 2       2       0    {1, 2}
## 3       3      3.61 {3, 4, 5, 6}
## 4       4       1    {3, 4, 5, 6}
## 5       5       0    {3, 4, 5, 6}
## 6       6      1.41 {3, 4, 5, 6}

```

### 13.3.2 CLARA

PAM	SWAP	,	.	.	CLARA
	PAM	.	.	,	5
.					
( , 2012)					

#### 13.3.2.1 R

```
clara_solution <- cluster::clara(df[, -1], k = 2)
```

```

clara      i.med
clara_solution$i.med
```

```
## [1] 2 5
```

```

clara      clustering
clara_solution$clustering
```

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

### 13.3.3 CLARANS

( , 2012)

### 13.3.4 K-means-like

PAM                    Park and Jun (2009)                    ,                    K-means  
 K-medoids            .                    .                    .                    Table  
 13.2                    .                    .                    .

```
[ 1] (      ) K      ,
d(i, j), i, j = 1, ..., n
pairwise_distance_df <- dist(df[, -1], upper = TRUE) %>%
  broom::tidy() %>%
  mutate_if(is.factor, ~ as.integer(as.character(.)))
```

$j = 1, \dots, n$

$$v_j = \sum_{i=1}^n \frac{d(i, j)}{\sum_{k=1}^n d(i, k)}$$

```
v_j <- pairwise_distance_df %>%
  group_by(item2) %>%
  mutate(prop = distance / sum(distance)) %>%
  ungroup() %>%
  group_by(item1) %>%
  summarize(sum_prop = sum(prop)) %>%
  rename(j = item1)
```

`v_j`

```
## # A tibble: 6 x 2
##       j   sum_prop
##   <int>    <dbl>
## 1     1     1.67
## 2     2     1.33
## 3     3     0.781
## 4     4     0.661
## 5     5     0.713
## 6     6     0.849
```

$v_j$                      $K$                     .                     $K = 2$                     .

```
medoids_ids <- v_j %>%
  arrange(sum_prop) %>%
  slice(1:2) %>%
  .$j
```

`medoids_ids`

```
## [1] 4 5
```

```

            assign_cluster
cluster_solution <- assign_cluster(
  pairwise_distance_df,
  medoids_ids
)

cluster_solution

## # A tibble: 6 x 3
##   object distance cluster
##   <int>     <dbl> <chr>
## 1      1     10.4 {1, 2, 3, 4}
## 2      2      8.25 {1, 2, 3, 4}
## 3      3      2.83 {1, 2, 3, 4}
## 4      4      0    {1, 2, 3, 4}
## 5      5      0    {5, 6}
## 6      6      1.41 {5, 6}
[ 2] ( )

find_medoids <- function(distance_df, ids) {
  distance_df %>%
    filter(
      item1 %in% ids,
      item2 %in% ids
    ) %>%
    group_by(item1) %>%
    summarize(total_distance = sum(distance)) %>%
    arrange(total_distance) %>%
    slice(1) %>%
    .$item1
}

cluster_objects <- cluster_solution %>%
  split(.\$cluster) %>%
  map(~.\$object)

medoids_ids <- map_int(
  cluster_objects,
  find_medoids,
  distance_df = pairwise_distance_df
)

medoids_ids

## {1, 2, 3, 4}      {5, 6}
##                 2          5

```

```
[ 3] (   ) . Stop , [ 2] .

new_cluster_solution <- assign_cluster(
  pairwise_distance_df,
  medoids_ids
)

is_converge <- near(
  1,
  # clusteval::cluster_similarity(
  #   as.factor(cluster_solution$cluster),
  #   as.factor(new_cluster_solution$cluster),
  #   similarity = "rand"
  # )
  flexclust::randIndex(
    as.factor(cluster_solution$cluster),
    as.factor(new_cluster_solution$cluster)
  )
)

print(is_converge)

## ARI
## FALSE
```

iteration                    iteration .

#### 13.3.4.1 R

```
R . .

init_medoids <- function(distance_df, k = 1) {
  object_ids <- unique(distance_df$item1)
  k <- min(k, length(object_ids))

  v_j <- distance_df %>%
    group_by(item2) %>%
    mutate(prop = distance / sum(distance)) %>%
    ungroup() %>%
    group_by(item1) %>%
    summarize(sum_prop = sum(prop)) %>%
    rename(j = item1)

  medoids_ids <- v_j %>%
    arrange(sum_prop) %>%
    slice(1:k) %>%
    .$j
```

```
    return(medoids_ids)
}

assign_cluster <- function(distance_df, medoids_ids) {
  distance_df %>%
    filter(
      !item1 %in% medoids_ids,
      item2 %in% medoids_ids
    ) %>%
    group_by(item1) %>%
    arrange(distance) %>%
    slice(1) %>%
    ungroup() %>%
    bind_rows(
      tibble(
        item1 = medoids_ids,
        item2 = medoids_ids,
        distance = 0
      )
    ) %>%
    rename(object = item1) %>%
    arrange(object) %>%
    group_by(item2) %>%
    mutate(cluster = str_c("{", str_c(object, collapse = ", "), "}")) %>%
    ungroup() %>%
    select(-item2)
}

find_medoids <- function(distance_df, ids) {
  distance_df %>%
    filter(
      item1 %in% ids,
      item2 %in% ids
    ) %>%
    group_by(item1) %>%
    summarize(total_distance = sum(distance)) %>%
    arrange(total_distance) %>%
    slice(1) %>%
    .$item1
}

kmeans_like_kmedoids <- function(distance_df, k = 1) {
  medoids_ids <- init_medoids(distance_df, k)

  while (TRUE) {
```

```

cluster_solution <- assign_cluster(distance_df, medoids_ids)

cluster_objects <- cluster_solution %>%
  split(.\$cluster) %>%
  map(~.\$object)

medoids_ids <- map_int(
  cluster_objects,
  find_medoids,
  distance_df = distance_df
)

new_cluster_solution <- assign_cluster(distance_df, medoids_ids)

is_converge <- near(
  1,
  flexclust::randIndex(
    as.factor(cluster_solution\$cluster),
    as.factor(new_cluster_solution\$cluster)
  )
)

  if (is_converge) break
}

return(medoids_ids)
}

kmeans_like_kmedoids Table 13.2 .
pairwise_distance_df <- dist(df[, -1], upper = TRUE) %>%
  broom::tidy() %>%
  mutate_if(is.factor, ~ as.integer(as.character(.)))

medoids_ids <- kmeans_like_kmedoids(pairwise_distance_df, k = 2)

medoids_ids

##      {1, 2} {3, 4, 5, 6}
##            1           5

```

## 13.4 K-means

$$\begin{array}{c}
 \text{K-means} \quad , \quad (\text{fuzzy}) \quad . \quad i \quad j \\
 P_{ij} \quad .
 \end{array}$$

Table 13.3: PC

	( $\$x\_1\$$ )	( $\$x\_2\$$ )
1	6	14
2	8	13
3	14	6
4	11	8
5	15	7
6	7	15
7	13	6
8	5	4
9	3	3
10	3	2

### 13.4.1 R

```

df <- tibble(
  id = c(1:10),
  x1 = c(6, 8, 14, 11, 15, 7, 13, 5, 3, 3),
  x2 = c(14, 13, 6, 8, 7, 15, 6, 4, 3, 2)
)

df %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(
      '',
      ' $\$x\_1\$$ ',
      ' $\$x\_2\$$ '
    ),
    caption = 'PC'
  )

cluster::fanny(df[, -1], k = 3, metric = "SqEuclidean")

## Fuzzy Clustering object of class 'fanny' :
## m.ship.expon.          2
## objective            18.37061
## tolerance           1e-15
## iterations           12
## converged            1
## maxit                 500
## n                      10

```

```

## Membership coefficients (in %, rounded):
##      [,1] [,2] [,3]
## [1,] 98   1    1
## [2,] 96   3    2
## [3,] 1    99   1
## [4,] 12   80   8
## [5,] 2    96   2
## [6,] 98   1    1
## [7,] 1    99   1
## [8,] 3    3    94
## [9,] 0    0    99
## [10,] 1   1    98
## Fuzzyness coefficients:
## dunn_coeff normalized
## 0.9225653 0.8838480
## Closest hard clustering:
## [1] 1 1 2 2 2 1 2 3 3 3
##
## Available components:
## [1] "membership"   "coeff"        "memb.exp"     "clustering"   "k.crisp"
## [6] "objective"     "convergence"   "diss"        "call"         "silinfo"
## [11] "data"

```

### 13.4.2

[ 0]  $K$  .

$$P_{ij} = \begin{cases} 1 & \text{if object } i \text{ belongs to cluster } j \\ 0 & \text{otherwise} \end{cases}$$

```

init_cluster <- function(df, k = 1) {
  k <- min(k, nrow(df))

  while (TRUE) {
    cluster_ind <- sample.int(k, size = nrow(df), replace = TRUE)
    if (length(unique(cluster_ind)) == k) break
  }

  map_dfc(unique(cluster_ind), ~ as.double(cluster_ind == .))
}

set.seed(4000)

cluster_membership <- init_cluster(df[, -1], k = 3)

```

```

## New names:
## * NA -> ...1
## * NA -> ...2
## * NA -> ...3
cluster_membership

## # A tibble: 10 x 3
##       ...1   ...2   ...3
##     <dbl> <dbl> <dbl>
## 1     1     0     0
## 2     1     0     0
## 3     0     1     0
## 4     1     0     0
## 5     0     0     1
## 6     0     1     0
## 7     0     0     1
## 8     0     1     0
## 9     1     0     0
## 10    1     0     0

```

[ 1] .

$$\mathbf{c}_j = \frac{\sum_{i=1}^n P_{ij}^m \mathbf{x}_i}{\sum_{i=1}^n P_{ij}^m}$$

m 1 .

```

find_center <- function(df, p, m) {
  wt <- p ^ m
  df %>%
    summarize_all(weighted.mean, w = wt)
}

cluster_df <- map_dfr(cluster_membership, find_center, df = df[, -1], m = 2)
cluster_df

```

```

## # A tibble: 3 x 2
##       x1     x2
##     <dbl> <dbl>
## 1   6.2    8
## 2  8.67   8.33
## 3 14      6.5

```

[ 2] membership  $P_{ij}$  .

$$P_{ij} = \frac{d(\mathbf{x}_i, \mathbf{c}_j)^{-\frac{1}{m-1}}}{\sum_{a=1}^K d(\mathbf{x}_i, \mathbf{c}_a)^{-\frac{1}{m-1}}}$$

```

d()
.

update_membership <- function(df, cluster_df, m) {
  distance_mat <- flexclust::dist2(df, cluster_df) ^ 2

  p <- distance_mat ^ (-1 / (m - 1)) %>%
    `/~(rowSums(.)) %>%
    as_tibble(.name_repair = "minimal")

  p
}

update_membership(df[, -1], cluster_df, m = 2)

## # A tibble: 10 x 3
##       ` `   ` `   ` `
##     <dbl> <dbl> <dbl>
## 1 0.451  0.414  0.135
## 2 0.380  0.483  0.137
## 3 0.00381 0.00730 0.989
## 4 0.139  0.576  0.285
## 5 0.0152  0.0285 0.956
## 6 0.406  0.427  0.166
## 7 0.0231  0.0479 0.929
## 8 0.574  0.311  0.115
## 9 0.542  0.315  0.143
## 10 0.508  0.325  0.166

```

### 13.4.3 R

```

fuzzy_kmeans <- function(df, k = 1, m = 2, max_iter = 1000L, tol = 1e-9) {
  k <- min(k, nrow(df))
  i <- 0L

  cluster_membership <- init_cluster(df, k)

  while (i < max_iter) {
    i <- i + 1L

    cluster_df <- map_dfr(cluster_membership, find_center, df = df, m = m)

    new_cluster_membership <- update_membership(df, cluster_df, m)
  }
}

```

```
if (max(abs(cluster_membership - new_cluster_membership)) < tol) break

cluster_membership <- new_cluster_membership
}

res <- list(
  n_iteration = i,
  center = cluster_df,
  membership = cluster_membership %>% as.matrix()
)

return (res)
}

set.seed(4000)

fuzzy_kmeans_solution <- fuzzy_kmeans(df[, -1], k = 3, m = 2)

## New names:
## * NA -> ...1
## * NA -> ...2
## * NA -> ...3
fuzzy_kmeans_solution$n_iteration

## [1] 16
fuzzy_kmeans_solution$center

## # A tibble: 3 x 2
##       x1     x2
##   <dbl> <dbl>
## 1  3.64  2.98
## 2  7.00 14.0 
## 3 13.4   6.63
fuzzy_kmeans_solution$membership

##
## [1,] 0.007809655 0.983147737 0.009042608
## [2,] 0.015726915 0.957515486 0.026757600
## [3,] 0.006068358 0.006268614 0.987663029
## [4,] 0.078772454 0.120672309 0.800555237
## [5,] 0.017165082 0.022105992 0.960728926
## [6,] 0.006532987 0.984345339 0.009121674
## [7,] 0.005945712 0.005766360 0.988287928
## [8,] 0.939278603 0.026072757 0.034648640
```

Table 13.4:

	\$x\_1\$	\$x\_2\$
1	4	12
2	6	13
3	6	15
4	10	4
5	11	3
6	12	2
7	12	5

```
## [9,] 0.993661877 0.002989486 0.003348637
## [10,] 0.981125198 0.008481303 0.010393499
fuzzy_kmeans_solution$membership %>%
  apply(1, which.max)

## [1] 2 2 3 3 3 2 3 1 1 1
```

## 13.5

(model-based clustering) (mixture) .

### 13.5.1 R script

```
df <- tibble(
  id = c(1:7),
  x1 = c(4, 6, 6, 10, 11, 12, 12),
  x2 = c(12, 13, 15, 4, 3, 2, 5)
)

df %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(
      '',
      '$x\_1$',
      '$x\_2$'
    ),
    caption = ''
  )
```

```

Table 13.4 7           R          .      mclust::meVII      -      ,
0           .

set.seed(1024)

#
K <- 2

# z_ik
init_z <- matrix(runif(nrow(df) * K), ncol = K) %>%
  `/~(apply(., 1, sum))

# EM    - - : unequal volume (V), spherical (II)
sol <- mclust::meVII(data = df[, -1], z = init_z)

#
sol$z

##          [,1]      [,2]
## [1,] 7.596402e-28 1.000000e+00
## [2,] 8.254183e-27 1.000000e+00
## [3,] 9.463816e-36 1.000000e+00
## [4,] 1.000000e+00 6.832084e-20
## [5,] 1.000000e+00 1.473491e-25
## [6,] 1.000000e+00 4.876251e-31
## [7,] 1.000000e+00 1.480388e-20

#
sol$parameters

## $pro
## [1] 0.5714286 0.4285714
##
## $mean
##      [,1]      [,2]
## x1 11.25  5.333333
## x2  3.50 13.333333
##
## $variance
## $variance$modelName
## [1] "VII"
##
## $variance$d
## [1] 2
##
## $variance$G
## [1] 2

```

```

##  

## $variance$sigma  

## , , 1  

##  

##      x1      x2  

## x1 0.96875 0.00000  

## x2 0.00000 0.96875  

##  

## , , 2  

##  

##      x1      x2  

## x1 1.222222 0.000000  

## x2 0.000000 1.222222  

##  

##  

## $variance$sigmasq  

## [1] 0.968750 1.222222  

##  

## $variance$scale  

## [1] 0.968750 1.222222  

##  

##  

## $Vinv  

## NULL

```

### 13.5.2 EM

- $f_k(\mathbf{x} | \theta_k)$ :  $k$   $\mathbf{x}$   $(\theta_k)$
- $\tau_k$ :  $k$   $(\tau_k \geq 0, \sum_{k=1}^K \tau = 1)$

$$, \quad \mathbf{x}$$

$$f(\mathbf{x} | \theta) = \sum_{k=1}^K \tau_k f_k(\mathbf{x} | \theta_k)$$

$$f_k \quad .$$

$$, \quad z_{ik} \quad .$$

$$z_{ik} = \begin{cases} 1 & \text{if object } i \text{ belongs to cluster } k \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ik} \quad , \quad , \quad i \quad k \quad . \quad (, 2012) \quad .$$

## 13.5.1

```
[ 0]  $\hat{z}_{ik}$  .  

set.seed(1024)  

#  

K <- 2  

# z_ik  

z_hat <- matrix(runif(nrow(df) * K), ncol = K) %>%
  `/~(apply(., 1, sum)) %>%
  as_tibble() %>%
  `names<-~(str_c("C", 1:K))  

z_hat
```

```
## # A tibble: 7 x 2
##       C1     C2
##   <dbl> <dbl>
## 1 0.406  0.594
## 2 0.623  0.377
## 3 0.372  0.628
## 4 0.956  0.0444
## 5 0.0214 0.979
## 6 0.681  0.319
## 7 0.264  0.736
```

```
[ 1] (M-step)  $\hat{z}_{ik}$  .
```

```
 $\tau_k$   $\mu_k$  .
```

$$\hat{\tau}_k = \frac{\sum_{i=1}^n \hat{z}_{ik}}{n}$$

```
tau_hat <- map(z_hat, mean)
tau_hat
```

```
## $C1
## [1] 0.4746762
##
## $C2
## [1] 0.5253238
```

$$\hat{\mu}_k = \frac{\sum_{i=1}^n \hat{z}_{ik} \mathbf{x}_i}{\sum_{i=1}^n \hat{z}_{ik}}$$

```

mu_hat <- map(z_hat, ~colSums(. * df[, -1]) / sum(.))
mu_hat

## $C1
##      x1      x2
## 8.644042 7.559792
##
## $C2
##      x1      x2
## 8.777757 7.853884

-  $\Sigma_k$  - . , -  $\Sigma_k$  decompose (Banfield and Raftery, 1993).

```

$$\Sigma_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^\top$$

- $\lambda_k = \det \Sigma_k^{1/2}$ ;  $k$
- $\mathbf{D}_k$ : matrix of eigenvectors of  $\Sigma_k$ ;  $k$  (orientation)
- $\mathbf{A}_k$ : diagonal matrix s.t.  $\det \mathbf{A}_k = 1$ ;  $k$

.  $\lambda_k, \mathbf{D}_k, \mathbf{A}_k$  . . . .

```

tribble(
  ~model, ~sigma,
  "VII", "$\\lambda_k \\mathbf{I}$",
  "VEI", "$\\lambda_k \\mathbf{A}$",
  "VEE", "$\\lambda_k \\mathbf{D} \\mathbf{A} \\mathbf{D}^\\top$"
) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('c', 'c'),
    col.names = c(
      ' - ',
      '$\\boldsymbol\\Sigma_k$'
    ),
    caption = 'Within-group - '
  )
,
```

- “VII”
  - for all  $k, \mathbf{D}_k = \mathbf{I}$
  - for all  $k, \mathbf{A}_k = \mathbf{I}$
- “VEI”
  - for all  $k, \mathbf{D}_k = \mathbf{I}$
  - for all  $k, \mathbf{A}_k = \mathbf{A}$

Table 13.5: Within-group -

-	$\$\\boldsymbol\\Sigma_k$$
VII	$\$\\lambda_k \\mathbf{I}$
VEI	$\$\\lambda_k \\mathbf{A}$
VEE	$\$\\lambda_k \\mathbf{D} \\mathbf{A} \\mathbf{D}^\\top$$

- “VEE”
  - for all  $k$ ,  $\mathbf{D}_k = \mathbf{D}$
  - for all  $k$ ,  $\mathbf{A}_k = \mathbf{A}$

Celeux and Govaert (1995) - , ( , 2012)

,  $k$  scatter matrix

$$\mathbf{W}_k = \sum_{i=1}^n \hat{z}_{ik} (\mathbf{x}_i - \hat{\mu}_k)(\mathbf{x}_i - \hat{\mu}_k)^\top$$

```
p <- ncol(df[, -1])
W <- map(mu_hat,
  ~pmap_dfc(list(x = df[, -1], mu = .),
    function(x, mu) x - mu)) %>%
  map(as.matrix, nrow = p, ncol = p) %>%
  map2(z_hat, ~ t(.x) %*% (.y * .x))
```

W

```
## $C1
##           x1          x2
## x1  28.22794 -44.46516
## x2 -44.46516  82.36848
##
## $C2
##           x1          x2
## x1  37.16942 -53.17491
## x2 -53.17491  92.90912
```

[VII ] closed-form

$$\lambda_k = \frac{Tr(\mathbf{W}_k)}{p \sum_{i=1}^n \hat{z}_{ik}}$$

$$\Sigma_k = \lambda_k \mathbf{I}$$

$$p \quad (\mathbf{x} \in \mathbb{R}^p).$$

[VEI ]

VEI-0.  $\mathbf{B} = \mathbf{I}$

VEI-1.  $\lambda_k$

$$\lambda_k = \frac{\text{Tr}(\mathbf{W}_k \mathbf{B}^{-1})}{p \sum_{i=1}^n \hat{z}_{ik}}$$

VEI-2.  $\mathbf{B}$

$$\mathbf{B} = \frac{\text{diag}\left(\sum_{k=1}^K \frac{\mathbf{W}_k}{\lambda_k}\right)}{\left(\det \text{diag}\left(\sum_{k=1}^K \frac{\mathbf{W}_k}{\lambda_k}\right)\right)^{1/p}}$$

VEI-3.

, VEI-1

$$\Sigma_k = \lambda_k \mathbf{B}$$

[VEE ]

VEE-0.  $\mathbf{C} = \mathbf{I}$

VEE-1.  $\lambda_k$

$$\lambda_k = \frac{\text{Tr}(\mathbf{W}_k \mathbf{C}^{-1})}{p \sum_{i=1}^n \hat{z}_{ik}}$$

VEE-2.  $\mathbf{C}$

$$\mathbf{C} = \frac{\sum_{k=1}^K \frac{\mathbf{W}_k}{\lambda_k}}{\left(\det \sum_{k=1}^K \frac{\mathbf{W}_k}{\lambda_k}\right)^{1/p}}$$

VEE-3.

, VEE-1

$$\Sigma_k = \lambda_k \mathbf{C}$$

$$\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^\top$$

Celeux and Govaert (1995)

- estimate\_mixture\_cov

3

- modelName: “VII”, “VEI”, “VEE”

```

• W: K scatter matrix ( $\mathbf{W}_1, \dots, \mathbf{W}_K$ ) (list)
• z:  $z_{ik}$  (data.frame). n K . , , .
estimate_mixture_cov <- function(modelName, W, z) {
  K <- ncol(z)
  p <- ncol(W[[1]])

  #
  D <- map(1:K, ~ diag(1, p))
  A <- map(1:K, ~ diag(1, p))

  get_lambda <- function(W, z, D, A, p) {
    pmap(
      list(W, z, D, A),
      function(W, z, D, A, p)
        sum(diag(W %*% D %*% solve(A) %*% t(D))) / (sum(z) * p),
      p = p
    )
  }

  objective_value <- function(W, z, lambda, D, A, p) {
    pmap_dbl(
      list(W, z, lambda, D, A),
      function(W, z, lambda, D, A, p)
        sum(diag(W %*% D %*% solve(A) %*% t(D))) / lambda +
        p * sum(z) * log(lambda),
      p = p
    ) %>%
      sum()
  }

  i <- 0L
  obj <- Inf

  while(TRUE) {
    i <- i + 1L

    lambda <- get_lambda(W, z, D, A, p)
    new_obj <- objective_value(W, z, lambda, D, A, p)

    if (obj - new_obj < 1e-9) break

    if (modelName == "VII") {

    } else if (modelName == "VEI") {
      B <- map2(W, lambda, ~ diag(.x) / .y) %>%
    }
  }
}

```

```

    reduce(`+`) %>%
    diag() %>%
    `/~(det(.) ^ (1 / p))
  A <- map(1:K, ~ B)
} else if (modelName == "VEE") {
  C <- map2(W, lambda, ~ .x / .y) %>%
    reduce(`+`) %>%
    `/~(det(.) ^ (1 / p))

  s <- svd(C)
  A <- map(1:K, ~ diag(s$d))
  D <- map(1:K, ~ s$u)
} else {
  stop("Model ", modelName, " is not supported yet.")
}

obj <- new_obj
}

Sigma <- pmap(
  list(lambda, D, A),
  function(lambda, D, A) lambda * (D %*% A %*% t(D))
)

return (list(
  volume = lambda,
  shape = A,
  orientation = D,
  Sigma = Sigma
))
}

```

$\hat{z}_{ik}$  “VII” - .  
`estimate_mixture_cov("VII", W, z_hat)`

```

## $volume
## $volume$C1
## [1] 16.64239
##
## $volume$C2
## [1] 17.68685
##
## 
## $shape
## $shape[[1]]

```

```

##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
##
## $shape[[2]]
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
##
## $orientation
## $orientation[[1]]
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
##
## $orientation[[2]]
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
##
## $Sigma
## $Sigma$C1
##      [,1]      [,2]
## [1,] 16.64239  0.00000
## [2,]  0.00000 16.64239
##
## $Sigma$C2
##      [,1]      [,2]
## [1,] 17.68685  0.00000
## [2,]  0.00000 17.68685

estimate_mixture_cov    "VII"     "VEI"  "VEE"  -
# estimate_mixture_cov("VEI", W, z_hat)
# estimate_mixture_cov("VEE", W, z_hat)

GMM_Mstep    .
GMM_Mstep <- function(df, z, modelName = "VII") {
  tau <- map(z, mean)
  mu <- map(z, ~colSums(. * df) / sum(.))

  p <- ncol(df)
  W <- map(mu, ~pmap_dfc(
    list(x = df, mu = .), function(x, mu) x - mu)) %>%

```

```

map(as.matrix, nrow = p, ncol = p) %>%
map2(z, ~ t(.x) %*% (.y * .x))

var_cov <- estimate_mixture_cov(modelName, W, z)

res <- list(
  tau = tau,
  mu = mu,
  Sigma = var_cov$Sigma
)

return (res)
}

Mstep_res <- GMM_Mstep(df[, -1], z_hat)

Mstep_res

## $tau
## $tau$C1
## [1] 0.4746762
##
## $tau$C2
## [1] 0.5253238
##
## 
## $mu
## $mu$C1
##      x1      x2
## 8.644042 7.559792
##
## $mu$C2
##      x1      x2
## 8.777757 7.853884
##
## 
## $Sigma
## $Sigma$C1
##      [,1]      [,2]
## [1,] 16.64239  0.00000
## [2,]  0.00000 16.64239
##
## $Sigma$C2
##      [,1]      [,2]
## [1,] 17.68685  0.00000

```

```

## [2,] 0.00000 17.68685
[ 2] (E-step) M-step       $\hat{z}_{ik}$  .
```

$$\hat{z}_{ik} = \frac{\hat{\tau}_k f_k(\mathbf{x}_i | \hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{l=1}^K \hat{\tau}_l f_l(\mathbf{x}_i | \hat{\mu}_l, \hat{\Sigma}_l)}$$

```

GMM_Estep <- function(df, tau, mu, Sigma) {
  pmap_dfc(list(tau = tau, mu = mu, Sigma = Sigma),
    function(df, tau, mu, Sigma)
      tau * mvtnorm::dmvnorm(df, mean = mu, sigma = Sigma),
      df = df) %>%
    `/.`(rowSums(.))
}

new_z_hat <- GMM_Estep(df[, -1], Mstep_res$tau, Mstep_res$mu, Mstep_res$Sigma)

new_z_hat
```

```

##          C1          C2
## 1 0.4626878 0.5373122
## 2 0.4568716 0.5431284
## 3 0.4373551 0.5626449
## 4 0.4964082 0.5035918
## 5 0.4934276 0.5065724
## 6 0.4886741 0.5113259
## 7 0.4870085 0.5129915
```

```

[ 3] stop,      [ 1] . ,  $z_{ik}$ 
```

```

max(abs(z_hat - new_z_hat)) < 1e-9
```

```

## [1] FALSE
TRUE , FALSE [ 1] .
```

### 13.5.3 R

```

GMM_EM      .      GMM_Mstep  GMM_Estep      .
GMM_EM <- function(df, K, modelName = "VII", tol = 1e-9) {
  K <- min(K, nrow(df))

  i <- 0L

  # [ 0] z_ik
  z_hat <- matrix(runif(nrow(df) * K), ncol = K) %>%
```

```

`/`(apply(., 1, sum)) %>%
as_tibble() %>%
`names<-`(str_c("C", 1:K))

while (TRUE) {
  i <- i + 1L

  # [ 1] M-step
  Mstep_res <- GMM_Mstep(df, z_hat, modelName)

  # [ 2] E-step
  new_z_hat <- GMM_Estep(df, Mstep_res$tau, Mstep_res$mu, Mstep_res$Sigma)

  # [ 3]
  if (max(abs(z_hat - new_z_hat)) < tol) break

  z_hat <- new_z_hat
}

return (list(z = z_hat,
             parameters = Mstep_res,
             n_iteration = i))
}

```

Table 13.4 .

```
VII_res <- GMM_EM(df[, -1], 2, "VII")
```

```
#  
VII_res$z
```

	C1	C2
## 1	1.000000e+00	6.786307e-28
## 2	1.000000e+00	8.771316e-27
## 3	1.000000e+00	8.895007e-36
## 4	5.251505e-17	1.000000e+00
## 5	7.046079e-22	1.000000e+00
## 6	1.843399e-26	1.000000e+00
## 7	1.544788e-17	1.000000e+00

```
#  
VII_res$parameters
```

```
## $tau
## $tau$C1
## [1] 0.4285714
##
```

```

## $tau$C2
## [1] 0.5714286
##
## 
## $mu
## $mu$C1
##      x1      x2
## 5.333333 13.333333
##
## $mu$C2
##      x1      x2
## 11.25   3.50
##
## 
## $Sigma
## $Sigma$C1
##      [,1]      [,2]
## [1,] 1.222222 0.000000
## [2,] 0.000000 1.222222
##
## $Sigma$C2
##      [,1]      [,2]
## [1,] 0.96875 0.00000
## [2,] 0.00000 0.96875

```

### 13.5.4 R

R mclust , VII, VEI, VEE - (Scrucca et al., 2016). “EEI” .

```
res_mclust_EEI <- mclust::meEEI(df[, -1], z_hat)
```

, z  $\hat{z}_{ik}$  , parameters  $(\hat{\tau}_k, \hat{\mu}_k, \hat{\Sigma}_k)$  .

```
res_mclust_EEI$z
```

```

##      [,1]      [,2]
## [1,] 6.198742e-26 1.000000e+00
## [2,] 2.193775e-22 1.000000e+00
## [3,] 1.432979e-28 1.000000e+00
## [4,] 1.000000e+00 3.497777e-20
## [5,] 1.000000e+00 1.350932e-26
## [6,] 1.000000e+00 5.217649e-33
## [7,] 1.000000e+00 9.883335e-24

```

```
res_mclust_EEI$parameters
```

```
## $pro
```

```
## [1] 0.5714286 0.4285714
##
## $mean
## [,1]      [,2]
## x1 11.25  5.333333
## x2  3.50 13.333333
##
## $variance
## $variance$modelName
## [1] "EEI"
##
## $variance$d
## [1] 2
##
## $variance$G
## [1] 2
##
## $variance$sigma
## , , 1
##
##          x1      x2
## x1 0.7738095 0.000000
## x2 0.0000000 1.380952
##
## , , 2
##
##          x1      x2
## x1 0.7738095 0.000000
## x2 0.0000000 1.380952
##
## $variance$Sigma
##          x1      x2
## x1 0.7738095 0.000000
## x2 0.0000000 1.380952
##
## $variance$scale
## [1] 1.033728
##
## $variance$shape
## [1] 0.7485618 1.3358950
##
## $Vinv
## NULL
```

```

mclust      densityMclust Bayesian information criterion (BIC)
             , - , , . .
res_mclust_opt <- mclust::densityMclust(df[, -1], verbose = FALSE)

BIC      .
res_mclust_opt$BIC

## Bayesian Information Criterion (BIC):
##          EII      VII     EEI      VEI      EVI      VVI      EEE
## 1 -85.40006 -85.40006 -85.70851 -85.70851 -85.70851 -85.70851 -75.27433
## 2 -62.00992 -63.86240 -63.37677 -65.22485 -65.32204 -67.17013 -64.66516
## 3 -65.47796      NA -66.01911      NA      NA      NA -67.87781
## 4 -69.04346      NA -70.17240      NA      NA      NA -67.31282
## 5 -72.02226      NA -73.96817      NA      NA      NA      NA
## 6 -62.27998      NA -64.22589      NA      NA      NA      NA
## 7      NA      NA      NA      NA      NA      NA      NA
##          VEE      EVE      VVE      EEV      VEV      EVV      VVV
## 1 -75.27433 -75.27433 -75.27433 -75.27433 -75.27433 -75.27433 -75.27433
## 2 -66.60332 -65.06811 -66.92481 -65.42506 -67.34154 -66.55375 -68.44666
## 3      NA      NA      NA -71.48895      NA      NA      NA
## 4      NA      NA      NA      NA      NA      NA      NA
## 5      NA      NA      NA      NA      NA      NA      NA
## 6      NA      NA      NA      NA      NA      NA      NA
## 7      NA      NA      NA      NA      NA      NA      NA
##
## Top 3 models based on the BIC criterion:
##          EII,2      EII,6      EEI,2
## -62.00992 -62.27998 -63.37677

BIC      EII -      2      .
densityMclust      plotting .
plot(res_mclust_opt, what = "density",
      data = df[, -1], points.cex = 0.5)

```

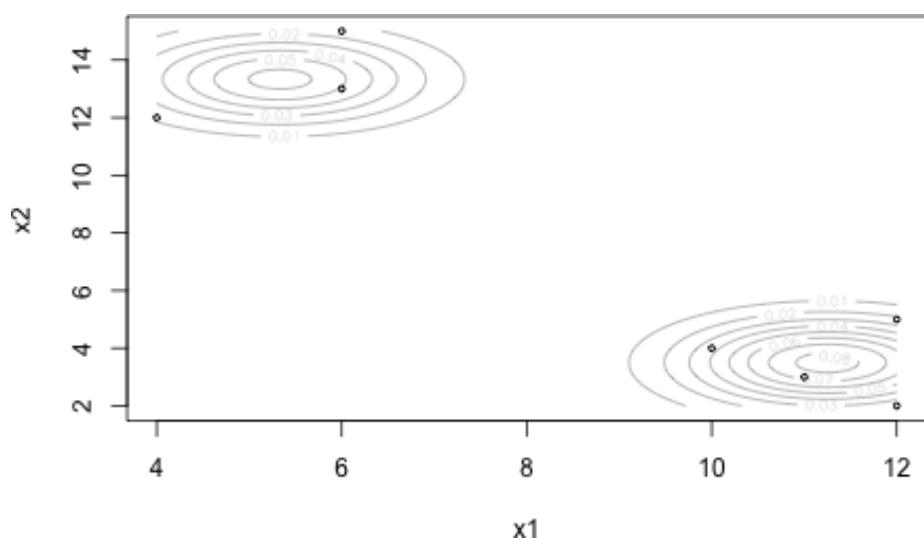


Figure 13.2: mclust::densityMclust

# Chapter 14

## 14.1 R package

package	version
tidyverse	1.3.1
cluster	2.1.2
flexclust	1.4-0
clValid	0.7

## 14.2

, (external index) (internal index)

### 14.2.1

$n$

$$\mathbf{U} = \{U_1, U_2, \dots, U_r\}$$

,  $\mathbf{U}$   $r$  ,  $k$   $U_k$  .  
,  $s$   $\mathbf{V}$  .

$$\mathbf{V} = \{V_1, V_2, \dots, V_s\}$$

### 14.2.1.1 R

$$\begin{aligned} U &= \{\{1, 2, 3, 4\}, \{5, 6, 7\}, \{8, 9, 10\}\} \\ V &= \{\{1, 2, 5, 8\}, \{3, 6, 9\}, \{4, 7, 10\}\} \end{aligned}$$

```

flexclust   randIndex      ,       correct = FALSE,
correct = TRUE

sol_1 <- c(1, 1, 1, 1, 2, 2, 2, 3, 3, 3)
sol_2 <- c(1, 1, 2, 3, 1, 2, 3, 1, 2, 3)
map(c(FALSE, TRUE),
~ flexclust::randIndex(x = sol_1, y = sol_2, correct = .x))

## [[1]]
##          RI
## 0.5111111
##
## [[2]]
##      ARI
## -0.25

```

### 14.2.1.2

$$U, V \quad \text{membership}$$

$$u_{ik} = \begin{cases} 1 & \text{if object } i \text{ belongs to cluster } U_k, \text{ i.e. } i \in U_k, i = 1, \dots, n, k = 1, \dots, r \\ 0 & \text{otherwise} \end{cases}$$

$$v_{ik} = \begin{cases} 1 & \text{if object } i \text{ belongs to cluster } V_k, \text{ i.e. } i \in V_k, i = 1, \dots, n, k = 1, \dots, s \\ 0 & \text{otherwise} \end{cases}$$

,

$$\begin{aligned}
 a &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \sum_{k=1}^r \sum_{l=1}^s u_{ik} u_{jk} v_{il} v_{jl} \\
 b &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \sum_{k=1}^r \sum_{l=1}^s u_{ik} u_{jk} v_{il} (1 - v_{jl}) \\
 c &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \sum_{k=1}^r \sum_{l=1}^s u_{ik} (1 - u_{jk}) v_{il} v_{jl} \\
 d &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \sum_{k=1}^r \sum_{l=1}^s u_{ik} (1 - u_{jk}) v_{il} (1 - v_{jl})
 \end{aligned}$$

, Rand (1971) (Rand index)

$$RI = \frac{a + d}{a + b + c + d} \quad (14.1)$$

,  $a + b + c + d$

$$a + b + c + d = \binom{n}{2}$$

$$(14.1) \quad \begin{matrix} 0 & 1 \\ 0 & . \end{matrix} \quad , \quad \begin{matrix} 0 & . \\ . & . \end{matrix} \quad , \quad \begin{matrix} 1 & . \\ . & . \end{matrix}$$

Hubert and Arabie (1985) (adjusted Rand index)

$$RI_{adj} = \frac{2(ad - bc)}{(a + b)(b + d) + (a + c)(c + d)} \quad (14.2)$$

$$(14.2) \quad \begin{matrix} 0 & . \\ . & . \end{matrix}$$

- `u, v`:  $\cdot$
- component list  $\cdot$ 
  - `ri`:
  - `adj_ri`:

```
rand_index <- function(u, v) {
  if (!is_vector(u) || !is_vector(v)) {
    stop("Input needs to be vector")
  } else if (length(u) != length(v)) {
    stop("Vectors u and v must have the same length.")
  }
```

```

U <- tibble(
  i = seq_along(u),
  cluster = u
) %>%
  inner_join(
    rename(., j = i),
    by = "cluster"
) %>%
  select(-cluster) %>%
  filter(i < j)

V <- tibble(
  i = seq_along(v),
  cluster = v
) %>%
  inner_join(
    rename(., j = i),
    by = "cluster"
) %>%
  select(-cluster) %>%
  filter(i < j)

a <- nrow(inner_join(U, V, by = c("i", "j")))
b <- nrow(anti_join(U, V, by = c("i", "j")))
c <- nrow(anti_join(V, U, by = c("i", "j")))
d <- choose(length(u), 2) - (a + b + c)

ri <- (a + d) / (a + b + c + d)
adj_ri <- 2 * (a * d - b * c) /
  ((a + b) * (b + d) + (a + c) * (c + d))

return(list(ri = ri, adj_ri = adj_ri))
}

```

```

50          (K = 3)           .

random_rand_index <- function(n, r, s) {
  u <- sample.int(r, size = n, replace = TRUE)
  v <- sample.int(s, size = n, replace = TRUE)
  rand_index(u, v)
}

set.seed(500)

rerun(200, random_rand_index(50, 3, 3)) %>%

```

```

bind_rows() %>%
gather(key = "metric", value = "value") %>%
ggplot(aes(x = value, fill = metric)) +
geom_histogram(binwidth = 0.05) +
scale_x_continuous(limits = c(-1, 1)) +
scale_fill_discrete(
  name = "index",
  breaks = c("ri", "adj_ri"),
  labels = c("Rand Index", "adjusted Rand Index")
) +
labs(
  title = "Distribution of index values from 200 random assignments",
  x = "index value",
  y = "frequency"
)

```

## Warning: Removed 4 rows containing missing values (geom\_bar).



Figure 14.1:

Figure 14.1 , , , .

$$\begin{aligned} U &= \{(1, 2, 3, 4), (5, 6, 7), (8, 9, 10)\} \\ V &= \{(1, 2, 5, 8), (3, 6, 9), (4, 7, 10)\} \end{aligned}$$

```
rand_index(
  c(1, 1, 1, 1, 2, 2, 2, 3, 3, 3),
  c(1, 1, 2, 3, 1, 2, 3, 1, 2, 3)
)

## $ri
## [1] 0.5111111
##
## $adj_ri
## [1] -0.25
```

### 14.2.2

(spatial separation) . , (compactness), (connectedness),

$K$  .  $C$  .

$$C = \{C_1, C_2, \dots, C_K\}$$

#### 14.2.2.1 R

```
x1 x2 .
df <- tribble(
  ~id, ~x1, ~x2,
  1, 4, 15,
  2, 20, 13,
  3, 3, 13,
  4, 19, 4,
  5, 17, 17,
  6, 8, 11,
  7, 19, 12,
  8, 18, 6
)

df %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('r', 'r', 'r'),
    col.names = c(
      ' ',
```

Table 14.1:

	\$x\_1\$	\$x\_2\$
1	4	15
2	20	13
3	3	13
4	19	4
5	17	17
6	8	11
7	19	12
8	18	6

```
'$x_1$', '$x_2$'
),
caption = ' '
)
```

$$\begin{aligned} U &= \{\{1, 3, 6\}, \{2, 5, 7\}, \{4, 8\}\} \\ V &= \{\{1, 3, 6\}, \{2, 4, 5, 7, 8\}\} \end{aligned} \quad (14.3)$$

```
sol_1 <- c(1, 2, 1, 3, 2, 1, 2, 3)
sol_2 <- c(1, 2, 1, 2, 2, 1, 2, 2)
```

4 . R cluster\_eval  
R .

- Dunn index (Dunn, 1973): `clValid::dunn`
- CH index (Caliński and Harabasz, 1974): `fpc::calinhara`
- Connectivity (Handl and Knowles, 2005): `clValid::connectivity`
- Silhouettes (Rousseeuw, 1987): `cluster::silhouette`

- `cluster`:  $n, K$
- `df`:
- `dist_method`: ; Dunn, Connectivity, Silhouettes
- `nn`: ( $>= 2$ ); Connectivity

```
cluster_eval <- function(cluster, df, dist_method = "euclidean", nn = 2) {
  dunn_index <- clValid::dunn(
```

```

Data = df,
clusters = cluster,
method = dist_method
)

ch_index <- fpc::calinhara(
  x = df,
  clustering = cluster
)

connectivity <- clValid::connectivity(
  Data = df,
  clusters = cluster,
  neighborSize = nn
)

asw <- cluster::silhouette(
  x = cluster,
  dist = dist(df, method = dist_method)
)[, "sil_width"] %>% mean()

return(list(
  dunn_index = dunn_index,
  ch_index = ch_index,
  connectivity = connectivity,
  asw = asw
))
}

map_dfr(list(sol_1, sol_2), cluster_eval, df = df[, -1], .id = "solution")

## # A tibble: 2 x 5
##   solution dunn_index ch_index connectivity   asw
##   <chr>        <dbl>     <dbl>        <dbl> <dbl>
## 1 1            1.08      26.5         1  0.651
## 2 2            0.822     15.4         0  0.586

```

#### 14.2.2.2

Dunn (1973) .

$$DI = \frac{\min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j, 1 \leq i \neq j \leq K} d(\mathbf{x}, \mathbf{y})}{\max_{\mathbf{x} \in C_i, \mathbf{y} \in C_i, 1 \leq i \leq K} d(\mathbf{x}, \mathbf{y})} \quad (14.4)$$

$$( ), ( ) . \quad (14.4)$$

```
(14.4)      dunn_index

• cluster:          n
• df:              n
• dist_method: base::dist   method

dunn_index <- function(cluster, df, dist_method = "euclidean") {
  #
  n <- nrow(df)

  #
  cluster_df <- tibble(
    id = 1:n,
    cluster = cluster
  )

  #
  #           `inner_join`
  #
  dist_df <- dist(df, method = dist_method, upper = TRUE) %>%
    broom::tidy() %>%
    mutate_if(is.factor, ~ as.integer(as.character(.))) %>%
    inner_join(
      cluster_df %>% rename(item1 = id, item1_cluster = cluster),
      by = "item1"
    ) %>%
    inner_join(
      cluster_df %>% rename(item2 = id, item2_cluster = cluster),
      by = "item2"
    )

  #
  #
  numerator <- dist_df %>%
    filter(item1_cluster != item2_cluster) %>%
    top_n(-1, distance) %>%
    slice(1) %>%
    .$distance

  #
  #
  denominator <- dist_df %>%
    filter(item1_cluster == item2_cluster) %>%
    top_n(1, distance) %>%
    slice(1) %>%
```

```

    .\$distance

    res <- numerator / denominator

    return(res)
}

map_dbl(list(sol_1, sol_2), dunn_index, df = df[, -1])

## [1] 1.075291 0.822375
Caliński and Harabasz (1974)

```

$$CH = \frac{\frac{1}{K-1} \sum_{k=1}^K n_k (\mathbf{c}_k - \mathbf{c})^\top (\mathbf{c}_k - \mathbf{c})}{\frac{1}{n-K} \sum_{k=1}^K \sum_{i \in C_k} (\mathbf{x}_i - \mathbf{c}_k)^\top (\mathbf{x}_i - \mathbf{c}_k)} \quad (14.5)$$

$$\mathbf{c}_k \quad k \quad \text{(centroid)}, \mathbf{c} \quad , \quad n_k \quad C_k \quad , \quad n \quad . \quad (14.5)$$

```

, , .
ch_index <- function(cluster, df) {
  #
  centroid <- df %>% summarize_all(mean)

  cluster_df <- df %>%
    mutate(cluster = cluster) %>%
    group_by(cluster) %>%
    nest()

  #
  cluster_centroid_df <- map_dfr(cluster_df$data, ~ summarize_all(., mean))

  #
  centroid_dist <- flexclust::dist2(
    cluster_centroid_df,
    centroid
  )^2

  #
  cluster_size <- map_dbl(cluster_df$data, nrow)

  numerator <- sum(centroid_dist * cluster_size) / (nrow(cluster_df) - 1)

  denominator <- map_dbl(
    cluster_df$data,
    #
    ~ flexclust::dist2(., summarize_all(., mean))^2 %>% sum()
  )
}
```

```

) %>%
sum() %>%
`/`(nrow(df) - nrow(cluster_df))

res <- numerator / denominator

return(res)
}

map_dbl(list(sol_1, sol_2), ch_index, df = df[, -1])
## [1] 26.45029 15.36218

```

Handl and Knowles (2005)

$$Conn = \sum_{i=1}^n \sum_{j=1}^L v_{i,nn_i(j)} \quad (14.6)$$

,  $nn_i(j)$      $i$     $j$     (nearest neighbor)    ,  $L$     .     $v_{i,nn_i(j)}$

.

$$v_{i,nn_i(j)} = \begin{cases} 0 & \text{if } \exists C_k : i, nn_i(j) \in C_k \\ 1/j & \text{otherwise} \end{cases}$$

, (14.6)     $j(\leq L)$      $1/j$     ,    .

```

connectivity <- function(cluster, df, dist_method = "euclidean", n_neighbor = 1) {
  n <- nrow(df)

  cluster_df <- tibble(
    id = 1:n,
    cluster = cluster
  )

  #
  distance_df <- dist(df, method = dist_method, upper = TRUE) %>%
    broom::tidy() %>%
    mutate_if(is.factor, ~ as.integer(as.character(.)))

  #
  nn_df <- distance_df %>%
    group_by(item1) %>%
    mutate(nearest = rank(distance, ties.method = "random")) %>%
    filter(nearest <= n_neighbor) %>%
    ungroup() %>%

```

```

inner_join(
  cluster_df %>% rename(item1 = id, item1_cluster = cluster),
  by = "item1"
) %>%
inner_join(
  cluster_df %>% rename(item2 = id, item2_cluster = cluster),
  by = "item2"
)

# nn_df %>%
filter(item1_cluster != item2_cluster) %>%
mutate(v = 1 / nearest) %>%
.$v %>%
sum()

}

map_dbl(list(sol_1, sol_2), connectivity, df = df[, -1], n_neighbor = 1)
## [1] 0 0
map_dbl(list(sol_1, sol_2), connectivity, df = df[, -1], n_neighbor = 2)
## [1] 1 0
Rousseeuw (1987) (silhouettes) .

```

- $a(i)$ :  $i$
- $d(i, C_k)$ :  $i \in C_k$ ,  $i \notin C_k$
- $b(i) = \min_{k:i \notin C_k} d(i, C_k)$

$z_{ik}$

$$z_{ik} = \begin{cases} 1 & \text{if } i \in C_k \\ 0 & \text{otherwise} \end{cases}$$

,  $a(i)$   $b(i)$  .

$$\begin{aligned} a(i) &= \sum_{k=1}^K z_{ik} \frac{\sum_{j \neq i} z_{jk} d(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j \neq i} z_{jk}} \\ b(i) &= \max_{k:z_{ik} \neq 1} \frac{\sum_{j \neq i} z_{jk} d(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j \neq i} z_{jk}} \end{aligned}$$

$i$

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (14.7)$$

$$(14.7) \quad \begin{array}{ccccccc} -1 & 1 & , & 1 & i & , & -1 \\ i & & & & (i \in C_k, |C_k| = 1). & a(i) & s(i) \\ . & & s(i) = 0 & . & . & . & (14.7) \end{array}$$

$$s(i) = \begin{cases} \frac{b(i)-a(i)}{\max\{a(i), b(i)\}} & \text{if } a(i) \text{ is defined} \\ 0 & \text{otherwise} \end{cases} \quad (14.8)$$

(overall average silhouette width; ASW)

$$ASW = \frac{1}{n} \sum_{i=1}^n s(i) \quad (14.9)$$

```
asw <- function(cluster, df, dist_method = "euclidean") {
  n <- nrow(df)

  cluster_df <- tibble(
    id = 1:n,
    cluster = cluster
  )

  dist_df <- dist(df, method = dist_method, upper = TRUE) %>%
    broom::tidy() %>%
    mutate_if(is.factor, ~ as.integer(as.character(.))) %>%
    inner_join(
      cluster_df %>% rename(item1 = id, item1_cluster = cluster),
      by = "item1"
    ) %>%
    inner_join(
      cluster_df %>% rename(item2 = id, item2_cluster = cluster),
      by = "item2"
    )

  dist_df <- dist_df %>%
    group_by(item1, item1_cluster, item2_cluster) %>%
    summarize(
      avg_distance = mean(distance)
    )

  a <- dist_df %>%
    filter(item1_cluster == item2_cluster) %>%
```

```
. $avg_distance

b <- dist_df %>%
  filter(item1_cluster != item2_cluster) %>%
  top_n(-1, avg_distance) %>%
  slice(1) %>%
  .$avg_distance

s <- map2_dbl(a, b, ~ (.y - .x) / max(.x, .y))

mean(s)
}

map_dbl(list(sol_1, sol_2), asw, df = df[, -1], dist_method = "euclidean")

## `summarise()` has grouped output by 'item1', 'item1_cluster'. You can override using
## `summarise()` has grouped output by 'item1', 'item1_cluster'. You can override using
## [1] 0.6507364 0.5864226
```

14.3

## **Part IV**

**4 -**



# Chapter 15

```
library(tidyverse)
```

(association rule)

## 15.1 R package

package	version
tidyverse	1.3.1
arules	1.6-8

## 15.2

$n$ ,  $m$ ,  $I$

$$I = \{i_1, \dots, i_m\}$$

,  $Z_j \subseteq I$

$$Z_j \subseteq I, j = 1, \dots, n$$

$R \subseteq X \times Y$  ( $X, Y \subseteq I, X \cap Y = \emptyset$ ) “ $X \Rightarrow Y$ ”

$$R : X \Rightarrow Y \quad (15.1)$$

(15.1)  $R$  (support), (confidence) (lift)

### 15.2.1

$$X \subseteq I$$

$$supp(X) = \frac{1}{n} \sum_{j=1}^n \mathbb{I}(X \subseteq Z_j)$$

,  $\mathbb{I}(a) = a = 1, 0$

(15.1)  $R$

$$supp(R) = supp(X \cup Y)$$

```

5
transaction_df <- tribble(
  ~transaction_id, ~item,
  1, "b",
  1, "c",
  1, "g",
  2, "a",
  2, "b",
  2, "d",
  2, "e",
  2, "f",
  3, "a",
  3, "b",
  3, "c",
  3, "g",
  4, "b",
  4, "c",
  4, "e",
  4, "f",
  5, "b",
  5, "c",
  5, "e",
  5, "f",
  5, "g"
)

transaction_df %>%
  group_by(transaction_id) %>%
  summarize(items = str_c(item, collapse = ", ")) %>%
  knitr::kable()

```

Table 15.1:

1	b, c, g
2	a, b, d, e, f
3	a, b, c, g
4	b, c, e, f
5	b, c, e, f, g

```

booktabs = TRUE,
align = c('c', 'c'),
col.names = c(' ', ' '),
caption = ' '
)

,      I {a,b,c,d,e,f,g} .
R : {b,c} ⇒ {g}

,      X = {b,c} .
support <- function(group_df, item, set) {
  if(is_empty(set)) return(1)

  group_df %>%
    unique() %>%
    summarize(n = sum(!rlang::sym(item) %in% set)) %>%
    mutate(is_support = (n == length(set))) %>%
    {mean(.\$is_support)}
}

X <- c("b", "c")
group_transaction_df <- transaction_df %>% group_by(transaction_id)

support(group_transaction_df, item = "item", set = X)

## [1] 0.8
,      R
Y <- c("g")
support(group_transaction_df, item = "item", set = union(X, Y))

## [1] 0.6

```

### 15.2.2

$R$ , .

$$conf(R) = \frac{supp(R)}{supp(X)} = \frac{supp(X \cup Y)}{supp(X)}$$

```
, X( ) Y( ) .
rule_confidence <- function(group_df, item, x, y) {
  support(group_df, item, union(x, y)) / support(group_df, item, x)
}

rule_confidence(group_transaction_df, item = "item", x = X, y = Y)

## [1] 0.75
```

### 15.2.3

.

$$lift(R) = \frac{conf(R)}{supp(Y)} = \frac{supp(X \cup Y)}{supp(X)supp(Y)}$$

```
rule_lift <- function(group_df, item, x, y) {
  rule_confidence(group_df, item, x, y) / support(group_df, item, y)
}

rule_lift(group_transaction_df, item = "item", x = X, y = Y)
```

```
## [1] 1.25
, b, c g 25% .
```

## 15.3

$R$ , . Apriori (Agrawal et al., 1994) .

### 15.3.1

(large itemsets)  $s_{min}$ , .

1.  $k$   $C_k$   $s_{min}$ , .  
 2.  $L_k$  ( )  $k + 1$   $L_k$   $C_{k+1}$  .

```

k 1 . , apriori_gen .
apriori_gen <- function(L) {
  if(length(L) < 2) return(NULL)

  n_sets <- length(L)
  n_item <- unique(map_dbl(L, length))

  if(length(n_item) > 1) stop("All itemsets must be the same length.")

  C <- combn(L, m = 2, simplify = TRUE) %>%
    t() %>%
    `colnames<-`(`c("set1", "set2")) %>%
    as_tibble() %>%
    pmap(function(set1, set2) {
      if(length(intersect(set1, set2)) != n_item - 1) return(NULL)
      sort(union(set1, set2))
    }) %>%
    compact() %>%
    unique()

  C
}

```

```

, Table 15.1      s_min = 0.4 .
s_min <- 0.4
group_transaction_df <- transaction_df %>% group_by(transaction_id)

candidate_itemsets <- as.list(sort(unique(transaction_df$item)))
large_itemsets <- vector("list", length = length(candidate_itemsets))

for(i in seq_along(large_itemsets)) {
  itemset_support <- map_dbl(candidate_itemsets,
    support,
    group_df = group_transaction_df,
    item = "item")

  large_itemsets[[i]] <- candidate_itemsets[itemset_support >= s_min]

  candidate_itemsets <- apriori_gen(large_itemsets[[i]])

  if(is.null(candidate_itemsets)) break
}

large_itemsets <- compact(large_itemsets)

```

```

map(large_itemsets,
  ~map_chr(.x, ~str_c("{", str_c(.x, collapse = ", "), "}")))
## [[1]]
## [1] "{a}" "{b}" "{c}" "{e}" "{f}" "{g}"
##
## [[2]]
## [1] "{a, b}" "{b, c}" "{b, e}" "{b, f}" "{b, g}" "{c, e}" "{c, f}" "{c, g}"
## [9] "{e, f}"
##
## [[3]]
## [1] "{b, c, e}" "{b, c, f}" "{b, c, g}" "{b, e, f}" "{c, e, f}"
##
## [[4]]
## [1] "{b, c, e, f}"

```

### 15.3.2

$$(L) \quad (X) \quad (Y = L \setminus A) \quad c_{\min} \quad R \quad .$$

$$R : X \Rightarrow L \setminus X$$

```

,      L      generate_rules .
generate_rules <- function(L, n_min_item = 1) {
  n_item <- length(L)
  if(n_item < n_min_item) return(NULL) #

  X <- map(seq_len(n_item), ~combn(L, m = .x - 1, list)) %>% flatten()
  Y <- map(X, ~setdiff(L, .x))

  tibble(X = X, Y = Y)
}
```

```

rule_list <- map_dfr(
  large_itemsets %>% flatten(),
  generate_rules
)

,      c_min      conf_rule_list .
rule_list$confidence <- pmap_dbl(rule_list, function(X, Y)
  rule_confidence(group_transaction_df, item = "item", x = X, y = Y)
)
```

```
c_min <- 0.7
conf_rule_list <- rule_list %>% filter(confidence >= c_min)

conf_rule_list %>%
  rowwise() %>%
  mutate(
    X = str_c("{", str_c(unlist(X), collapse = ", "), "}"),
    Y = str_c("{", str_c(unlist(Y), collapse = ", "), "}")
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('c', 'c', 'c'),
    col.names = c(' $X$', ' $Y$', ' '),
    caption = ''
  )
)
```

, , . . . ( , 2012) .

### 15.3.3 R Apriori

R arules apriori  
, 15.2.1 transaction\_df arules transactions

```
requireNamespace("arules")

## Loading required namespace: arules
transaction_df2 <- as(
  split(transaction_df$item, transaction_df$transaction_id),
  "transactions"
)

, apriori
rule_results <- arules::apriori(
  transaction_df2,
  parameter = list(
    support = 0.4,
    confidence = 0.7,
    target = "rules"
  ),
  control = list(
    verbose = FALSE
  )
)
```

Table 15.2:

\$X\$	\$Y\$	
{}	{b}	1.00
{}	{c}	0.80
{a}	{b}	1.00
{}	{b, c}	0.80
{b}	{c}	0.80
{c}	{b}	1.00
{e}	{b}	1.00
{f}	{b}	1.00
{g}	{b}	1.00
{c}	{g}	0.75
{g}	{c}	1.00
{e}	{f}	1.00
{f}	{e}	1.00
{c, e}	{b}	1.00
{c, f}	{b}	1.00
{c}	{b, g}	0.75
{g}	{b, c}	1.00
{b, c}	{g}	0.75
{b, g}	{c}	1.00
{c, g}	{b}	1.00
{e}	{b, f}	1.00
{f}	{b, e}	1.00
{b, e}	{f}	1.00
{b, f}	{e}	1.00
{e, f}	{b}	1.00
{c, e}	{f}	1.00
{c, f}	{e}	1.00
{c, e}	{b, f}	1.00
{c, f}	{b, e}	1.00
{b, c, e}	{f}	1.00
{b, c, f}	{e}	1.00
{c, e, f}	{b}	1.00

```

)
  rules
  • lhs:
  • rhs:
  • quality: ( , , , )
rule_results_df <- tibble(
  X = as(rule_results@lhs, "list"),
  Y = as(rule_results@rhs, "list")
) %>%
  bind_cols(rule_results@quality)

rule_results_df %>%
  rowwise() %>%
  mutate(
    X = str_c("{", str_c(unlist(X), collapse = ", "), "}"),
    Y = str_c("{", str_c(unlist(Y), collapse = ", "), "}")
) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('c', 'c', 'c', 'c', 'c', 'c', 'c'),
    col.names = c(' $X$', ' $Y$', ' ',
                 ' ', ' ', ' ', ' '),
    caption = ' - arules::apriori'
)

```

Table 15.3 , . Apriori (Agrawal et al.,  
1993) , 15.3.2 Apriori .

## 15.4

(sequential pattern) , “ ” .  
, . (sequence) ,  $A_j$   $j$  , .

$$s = \langle A_1, A_2, \dots, A_n \rangle$$

$$, k \quad k- .$$

$$\text{length}(\langle A_1, A_2, \dots, A_n \rangle) = n$$

$$s_1 = \langle A_1, A_2, \dots, A_n \rangle \quad s_2 = \langle B_1, B_2, \dots, B_m \rangle \quad (n \leq m),$$

Table 15.3: - arules::apriori

\$X\$	\$Y\$					
{}	{c}	0.8	0.80	1.0	1.000000	4
{}	{b}	1.0	1.00	1.0	1.000000	5
{a}	{b}	0.4	1.00	0.4	1.000000	2
{g}	{c}	0.6	1.00	0.6	1.250000	3
{c}	{g}	0.6	0.75	0.8	1.250000	3
{g}	{b}	0.6	1.00	0.6	1.000000	3
{e}	{f}	0.6	1.00	0.6	1.666667	3
{f}	{e}	0.6	1.00	0.6	1.666667	3
{e}	{b}	0.6	1.00	0.6	1.000000	3
{f}	{b}	0.6	1.00	0.6	1.000000	3
{c}	{b}	0.8	1.00	0.8	1.000000	4
{b}	{c}	0.8	0.80	1.0	1.000000	4
{c, g}	{b}	0.6	1.00	0.6	1.000000	3
{b, g}	{c}	0.6	1.00	0.6	1.250000	3
{b, c}	{g}	0.6	0.75	0.8	1.250000	3
{c, e}	{f}	0.4	1.00	0.4	1.666667	2
{c, f}	{e}	0.4	1.00	0.4	1.666667	2
{e, f}	{b}	0.6	1.00	0.6	1.000000	3
{b, e}	{f}	0.6	1.00	0.6	1.666667	3
{b, f}	{e}	0.6	1.00	0.6	1.666667	3
{c, e}	{b}	0.4	1.00	0.4	1.000000	2
{c, f}	{b}	0.4	1.00	0.4	1.000000	2
{c, e, f}	{b}	0.4	1.00	0.4	1.000000	2
{b, c, e}	{f}	0.4	1.00	0.4	1.666667	2
{b, c, f}	{e}	0.4	1.00	0.4	1.666667	2

$$\begin{aligned}
 A_1 &\subseteq B_{i_1}, A_2 \subseteq B_{i_2}, \dots, A_n \subseteq B_{i_n} \\
 i_1 < i_2 < \dots < i_n \quad , \quad s_1 & s_2 \quad , \quad s_1 & s_2 \quad , \quad \dots \\
 s & \qquad \text{(maximal sequence)} \quad . \\
 N & \qquad \text{(customer sequence)} \quad , \quad s_i \quad (i = 1, \dots, N), \\
 s & \qquad \text{.} \\
 supp(s) &= \frac{1}{N} \sum_{i=1}^N I(s \prec s_i) \\
 , \quad & \qquad \text{(large sequence)} \quad . \quad , \quad \qquad \text{(maximal} \\
 \text{sequence)} \quad & \qquad \text{.} \\
 \end{aligned}$$

### 15.4.1 AprioriAll

AprioriAll

```

sequential_transaction_df <- tribble(
  ~customer_id, ~transaction_seq, ~item,
  1, 1, "a",
  1, 2, "b",
  2, 1, "c",
  2, 1, "d",
  2, 2, "a",
  2, 3, "e",
  2, 3, "f",
  2, 3, "g",
  3, 1, "a",
  3, 1, "h",
  3, 1, "g",
  4, 1, "a",
  4, 2, "e",
  4, 2, "g",
  4, 3, "b",
  5, 1, "b"
)

sequential_transaction_df %>%
  group_by(customer_id, transaction_seq) %>%

```

Table 15.4:

ID (\$i\$)	(\$s_i\$)
1	<{a}, {b}>
2	<{c, d}, {a}, {e, f, g}>
3	<{a, h, g}>
4	<{a}, {e, g}, {b}>
5	<{b}>

```

summarize(itemset = str_c("{", str_c(item, collapse = ", "), "})") %>%
summarize(sequence = str_c("<", str_c(itemset, collapse = ", "), ">")) %>%
knitr::kable(
  booktabs = TRUE,
  align = c("c", "c"),
  col.names = c(" ID ($i$)", "      ($s_i$)"),
  caption = "      "
)

## `summarise()` has grouped output by 'customer_id'. You can override using the `group` argument
## or `ungroup()`.

is_contained
support_sequence . .
is_contained <- function(x, pattern) {
  n_x <- length(x)
  n_pattern <- length(pattern)
  if (n_pattern == 0) return (TRUE)

  rtn <- FALSE

  location <- rep(NA_integer_, n_pattern)

  if (n_x >= n_pattern) {
    j <- 1L
    for(i in seq_len(n_x)) {
      if (is_empty(setdiff(pattern[[j]], x[[i]]))) {
        location[j] <- i
        j <- j + 1
        if (j > n_pattern) {
          rtn <- TRUE
          break
        }
      }
    }
  }
}

```

```

        }
    }

    rtn
}

support_sequence <- function(sequence_list, pattern) {
  map_lgl(sequence_list, is_contained, pattern = pattern) %>% mean()
}

15.3.1      apriori_gen      (      )      .
s_min <- 0.4

customer_sequence <- sequential_transaction_df %>%
  group_by(customer_id, transaction_seq) %>%
  summarise(itemset = list(item)) %>%
  summarise(sequence = list(itemset))

## `summarise()` has grouped output by 'customer_id'. You can override using the `.` argument
candidate_itemsets <- map(sort(unique(sequential_transaction_df$item)), ~list(.x))
large_itemsets <- vector("list", length(candidate_itemsets))

for (i in seq_along(large_itemsets)) {
  large_itemsets[[i]] <- candidate_itemsets[
    map_dbl(candidate_itemsets,
           ~support_sequence(customer_sequence$sequence, pattern = .x)) >= s_min
  ] %>% flatten()

  candidate_itemsets <- map(apriori_gen(large_itemsets[[i]]), ~list(.x))
}

large_itemsets <- large_itemsets %>% flatten()

,
large_itemsets

## [[1]]
## [1] "a"
##
## [[2]]
## [1] "b"
##
## [[3]]
## [1] "e"
##

```

Table 15.5:

{a}	1
{b}	2
{e}	3
{g}	4
{e, g}	5

```

## [[4]]
## [1] "g"
##
## [[5]]
## [1] "e" "g"

,
large_itemset_df <- tibble(
  itemset = large_itemsets,
  mapped_to = seq_along(large_itemsets)
)

large_itemset_df %>%
  rowwise() %>%
  mutate(itemset = str_c("{", str_c(itemset, collapse = ", "), "}")) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c("c", "c"),
    col.names = c("    ", "    "),
    caption = "    "
  )

,
customer_sequence$transformed_sequence <- customer_sequence$sequence %>%
  map(~map(.x, function(x) {
    large_itemset_df$mapped_to[
      map_lgl(large_itemset_df$itemset, ~is_contained(x, .x))
    ]
  })) %>% compact()

print_sequence <- function(sequence) {
  str_c("<", str_c(
    map(sequence, function(x)

```

Table 15.6:

ID		
1	$\langle \{a\}, \{b\} \rangle$	$\langle \{1\}, \{2\} \rangle$
2	$\langle \{c, d\}, \{a\}, \{e, f, g\} \rangle$	$\langle \{1\}, \{3, 4, 5\} \rangle$
3	$\langle \{a, h, g\} \rangle$	$\langle \{1, 4\} \rangle$
4	$\langle \{a\}, \{e, g\}, \{b\} \rangle$	$\langle \{1\}, \{3, 4, 5\}, \{2\} \rangle$
5	$\langle \{b\} \rangle$	$\langle \{2\} \rangle$

```

    str_c(map_chr(x, ~str_c("[" , str_c(.x, collapse = " ", ")"),
                  collapse = ", ")),
      ">"))
}

customer_sequence %>%
  group_by(customer_id) %>%
  mutate(
    sequence = print_sequence(sequence),
    transformed_sequence = print_sequence(transformed_sequence)
  ) %>%
  knitr::kable(
    booktabs = TRUE,
    align = c("c", "c", "c"),
    col.names = c(" ID", "      ", "      "),
    caption = "      "
  )
  ,   1      .
      ,   1      (      )      .
      ,
      .
max_sequence_length <- max(map_int(customer_sequence$transformed_sequence, ~length(.x)))

1-
large_sequences <- vector("list", length = max_sequence_length)
large_sequences[[1]] <- map(large_itemset_df$mapped_to, ~list(.x))

  1      generate_sequence      .

generate_sequence <- function(seq1, seq2) {
  if (length(seq1) != length(seq2)) stop("Two sequences must be the same length.")

  # two k-sequences needs to be the same for first k-1 items to generate new sequence
  new_sequence <- NULL

```

```

k <- length(seq1)
if (identical(seq1[seq_len(k - 1)], seq2[seq_len(k - 1)])) {
  new_sequence <- c(seq1, seq2[[k]])
}

new_sequence
}

k- (k + 1)- apriori_seq_gen .
apriori_seq_gen <- function(L) {
  n_seqs <- length(L)
  n_item <- unique(map_dbl(L, length))

  if (length(n_item) > 1) stop("All sequences must be the same length.")

  k <- n_item

  # generate large new sequences with length (k+1)
  C <- vector("list", length = n_seqs * n_seqs)
  for (i in seq_along(L)) {
    for (j in seq_along(L)) {
      candidate_sequence <- generate_sequence(L[[i]], L[[j]])

      # check whether all subsequences with length k are element of L
      if (
        all(map_lgl(seq_len(k), function(x)
          any(map_lgl(L, ~identical(.x, candidate_sequence[-x]))))
        ))
      ) {
        C[[n_seqs * (i - 1) + j]] <- candidate_sequence
      }
    }
  }

  compact(C)
}

apriori_seq_gen (k + 1)-
, . support_sequence , get_large_sequence .
get_large_sequence <- function(sequence_list, C, s_min) {
  is_large <- map_lgl(
    C, ~ support_sequence(sequence_list, .x) >= s_min
  )

  C[is_large]
}

```

```

}

      k-      k   1
s_min <- 0.4

large_sequences <- vector("list", length = max_sequence_length)
large_sequences[[1]] <- map(large_itemset_df$mapped_to, ~list(.x))

for(i in seq_len(max_sequence_length - 1)) {
  large_sequences[[i + 1]] <- get_large_sequence(
    customer_sequence$transformed_sequence,
    apriori_seq_gen(large_sequences[[i]]),
    s_min
  )

  if(is_empty(large_sequences[[i + 1]])) break
}

large_sequences <- flatten(compact(large_sequences))

.

map_chr(large_sequences,
       ~str_c("<", str_c(str_c("{", unlist(.x), "}"), collapse = ", "), ">"))

## [1] "<{1}>"      "<{2}>"      "<{3}>"      "<{4}>"      "<{5}>" 
## [6] "<{1}, {2}>"  "<{1}, {3}>"  "<{1}, {4}>"  "<{1}, {5}>" 

,
maximal_large_sequences <- large_sequences

for (i in seq(from = length(large_sequences), to = 2)) {
  if(!is_empty(maximal_large_sequences[i])) {
    is_subsequence <- map_lgl(maximal_large_sequences[seq_len(i - 1)],
                                ~is_contained(maximal_large_sequences[i], .x))

    walk(which(is_subsequence), function(x) maximal_large_sequences[[x]] <- list())
  }
}

maximal_large_sequences <- compact(maximal_large_sequences)

(      )
map_chr(maximal_large_sequences,
       ~str_c("<", str_c(str_c("{", unlist(.x), "}"), collapse = ", "), ">"))

```

```
## [1] "<{1}, {2}>" "<{1}, {3}>" "<{1}, {4}>" "<{1}, {5}>"  
AprioriSome , DynamicSome . (2012) Agrawal  
et al. (1995) .
```

# Chapter 16

```
library(tidyverse)

(recommender system) , ,
(content-based) , (collaborative filtering), (hybrid) .
```

## 16.1 R package

package	version
tidyverse	1.3.1
tidytext	0.3.1

## 16.2

- $N$ :
- $f_{ij}$ :  $j$   $i$
- $n_i$ :  $i$

```
tidytext

library(tidytext)

janeaustenr      Jane Austen 6
text            book

library(janeaustenr)
tidy_books <- austen_books()
head(tidy_books)
```

```

## # A tibble: 6 x 2
##   text                 book
##   <chr>                <fct>
## 1 "SENSE AND SENSIBILITY" Sense & Sensibility
## 2 ""                   Sense & Sensibility
## 3 "by Jane Austen"     Sense & Sensibility
## 4 ""                   Sense & Sensibility
## 5 "(1811)"            Sense & Sensibility
## 6 ""                   Sense & Sensibility

n_book <- nlevels(tidy_books$book)
print(n_book)

## [1] 6

text
tidy_words <- tidy_books %>% unnest_tokens(word, text)

head(tidy_words)

## # A tibble: 6 x 2
##   book              word
##   <fct>             <chr>
## 1 Sense & Sensibility sense
## 2 Sense & Sensibility and
## 3 Sense & Sensibility sensibility
## 4 Sense & Sensibility by
## 5 Sense & Sensibility jane
## 6 Sense & Sensibility austen

,   i   j      (term frequency)   i   j   .


$$TF_{ij} = \frac{f_{ij}}{\sum_k f_{kj}}$$


tf_results <- tidy_words %>%
  group_by(book, word) %>%
  summarize(n = n()) %>%
  mutate(tf = n / sum(n)) %>%
  select(-n) %>%
  ungroup() %>%
  complete(book, word, fill = list(tf = 0))

## `summarise()` has grouped output by 'book'. You can override using the `groups` arg

```

```
, . . . , “the”, “to”, “and”

tf_results %>% arrange(desc(tf)) %>% head(10)

## # A tibble: 10 x 3
##   book             word      tf
##   <fct>            <chr>    <dbl>
## 1 Northanger Abbey the  0.0409
## 2 Persuasion        the  0.0398
## 3 Mansfield Park   the  0.0387
## 4 Pride & Prejudice the  0.0354
## 5 Sense & Sensibility to  0.0343
## 6 Sense & Sensibility the  0.0342
## 7 Mansfield Park   to   0.0341
## 8 Pride & Prejudice to   0.0341
## 9 Mansfield Park   and  0.0339
## 10 Persuasion       to   0.0336

, i (inverse document frequency) .
```

$$IDF_i = \log \frac{N}{n_i}$$

```
idf_results <- tf_results %>%
  filter(tf > 0) %>%
  group_by(word) %>%
  summarize(n = n()) %>%
  mutate(idf = log(n_book / n)) %>%
  select(-n)

idf_results %>% arrange(desc(idf)) %>% head(10)
```

```
## # A tibble: 10 x 2
##   word      idf
##   <chr>    <dbl>
## 1 _accepted_ 1.79
## 2 _accident_ 1.79
## 3 _adair_    1.79
## 4 _addition_ 1.79
## 5 _advantages_ 1.79
## 6 _affect_   1.79
## 7 _against_  1.79
## 8 _agreeable_ 1.79
## 9 _air_      1.79
## 10 _allow_   1.79
```

, TF-IDF .

$$w_{ij} = TF_{ij} \times IDF_i$$

```
tf_idf_results <- inner_join(tf_results, idf_results, by = "word") %>%
  mutate(tf_idf = tf * idf)
```

## TF-IDF

```
tf_idf_results %>%
  arrange(desc(tf_idf)) %>%
  head(10)
```

```
## # A tibble: 10 x 5
##   book             word      tf     idf    tf_idf
##   <fct>           <chr>    <dbl>  <dbl>   <dbl>
## 1 Sense & Sensibility elinor  0.00519  1.79  0.00931
## 2 Sense & Sensibility marianne 0.00410  1.79  0.00735
## 3 Mansfield Park    crawford 0.00307  1.79  0.00551
## 4 Pride & Prejudice  darcy   0.00305  1.79  0.00547
## 5 Persuasion        elliot   0.00304  1.79  0.00544
## 6 Emma              emma    0.00488  1.10  0.00536
## 7 Northanger Abbey  tilney   0.00252  1.79  0.00452
## 8 Emma              weston   0.00242  1.79  0.00433
## 9 Pride & Prejudice bennet   0.00241  1.79  0.00431
## 10 Persuasion       wentworth 0.00228  1.79  0.00409
```

```
TF-IDF      tidytext    bind_tf_idf
tidy_words %>%
  group_by(book, word) %>%
  summarize(n = n()) %>%
  ungroup() %>%
  bind_tf_idf(word, book, n) %>%
  arrange(desc(tf_idf)) %>%
  head(10)
```

```
## `summarise()` has grouped output by 'book'. You can override using the `.groups` arg
## # A tibble: 10 x 6
##   book             word      n      tf     idf    tf_idf
##   <fct>           <chr>    <int>  <dbl>  <dbl>   <dbl>
## 1 Sense & Sensibility elinor  623  0.00519  1.79  0.00931
## 2 Sense & Sensibility marianne 492  0.00410  1.79  0.00735
## 3 Mansfield Park    crawford 493  0.00307  1.79  0.00551
## 4 Pride & Prejudice  darcy   373  0.00305  1.79  0.00547
## 5 Persuasion        elliot   254  0.00304  1.79  0.00544
```

Table 16.1:

$(\$i\$)$	$(\$w_{iu}\$)$
kitty	0.3
cottage	0.3
judgment	0.1
war	0.1
sea	0.2

```
## 6 Emma           emma      786 0.00488 1.10 0.00536
## 7 Northanger Abbey   tilney    196 0.00252 1.79 0.00452
## 8 Emma           weston     389 0.00242 1.79 0.00433
## 9 Pride & Prejudice bennet     294 0.00241 1.79 0.00431
## 10 Persuasion      wentworth 191 0.00228 1.79 0.00409
```

$u$                      $w_{iu}$                     .

```
words_of_interest <- tibble(
  word = c("kitty", "cottage", "judgment", "war", "sea"),
  weight = c(0.3, 0.3, 0.1, 0.1, 0.2)
)

words_of_interest %>%
  knitr::kable(
    booktabs = TRUE,
    align = c('c', 'c'),
    col.names = c(' $(i$)', ' $(w_{iu})$'),
    caption = ''
  )
,
```

$u$        $j$       (utility)      (cosine similarity measure)      .

$$u(a, j) = \frac{\sum_{i=1}^K w_{iu} w_{ij}}{\sqrt{\sum_{i=1}^K w_{iu}^2} \sqrt{\sum_{i=1}^K w_{ij}^2}}$$

```
utility_results <- tf_idf_results %>%
  inner_join(words_of_interest, by = "word") %>%
  group_by(book) %>%
  summarize(utility = sum(weight * tf_idf) /
            (sqrt(sum(weight ^ 2)) * sqrt(sum(tf_idf ^ 2)))) %>%
  arrange(desc(utility))

print(utility_results)
```

```

## # A tibble: 6 x 2
##   book           utility
##   <fct>        <dbl>
## 1 Persuasion     0.753
## 2 Emma           0.710
## 3 Sense & Sensibility 0.640
## 4 Pride & Prejudice    0.615
## 5 Northanger Abbey 0.529
## 6 Mansfield Park  0.404

Persuasion      .
(2012)          R      .

words_of_interest <- tribble(
  ~word, ~weight,
  "word1", 0.124,
  "word2", 0.275,
  "word3", 0.019,
  "word4", 0.182,
  "word5", 0.223
)

tf_idf_results <- tribble(
  ~document, ~word, ~tf_idf,
  "doc1", "word1", 0.0194,
  "doc1", "word2", 0.0043,
  "doc1", "word3", 0.0054,
  "doc1", "word4", 0.0155,
  "doc1", "word5", 0.0028,
  "doc2", "word1", 0.0082,
  "doc2", "word2", 0.0032,
  "doc2", "word3", 0.0007,
  "doc2", "word4", 0.0104,
  "doc2", "word5", 0.0073,
  "doc3", "word1", 0.0087,
  "doc3", "word2", 0.0174,
  "doc3", "word3", 0.0091,
  "doc3", "word4", 0.0086,
  "doc3", "word5", 0.0268,
  "doc4", "word1", 0.0093,
  "doc4", "word2", 0.0061,
  "doc4", "word3", 0.0172,
  "doc4", "word4", 0.0028,
  "doc4", "word5", 0.0009,
  "doc5", "word1", 0.0185,
  "doc5", "word2", 0.0249,
)

```

16.3

- $v_{ij}$ :  $i$        $j$
- $I_i$ :  $i$
- $|I_i|$ :  $I_i$

,       $i$       .

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{ij}$$

$$w(a, i) = \frac{\sum_{j \in I_a \cap I_i} (v_{aj} - \bar{v}_a)(v_{ij} - \bar{v}_i)}{\sqrt{\sum_{j \in I_a \cap I_i} (v_{aj} - \bar{v}_a)^2} \sqrt{\sum_{j \in I_a \cap I_i} (v_{ij} - \bar{v}_i)^2}}$$

,       $a$                            $j$                           .

$$\hat{v}_{aj} = \bar{v}_a + \frac{1}{\sum_{i=1}^n |w(a, i)|} \sum_{i=1}^n w(a, i)(v_{ij} - \bar{v}_i)$$

@jun2012datamining                                    R                          .

```
rating_df <- tribble(
  ~customer, ~item, ~rating,
  " 1", " 1", 5,
  " 1", " 3", 4,
  " 1", " 5", 1,
  " 1", " 6", 0,
  " 1", " 7", 3,
  " 2", " 1", 4,
  " 2", " 2", 4,
  " 2", " 3", 4,
  " 2", " 7", 1,
  " 3", " 1", 5,
  " 3", " 2", 4,
  " 3", " 4", 1,
  " 3", " 5", 2,
  " 3", " 7", 3,
  " 4", " 1", 1,
  " 4", " 2", 2,
  " 4", " 3", 1,
  " 4", " 4", 4,
  " 4", " 5", 3,
  " 4", " 6", 5,
  " 4", " 7", 2,
  " 5", " 1", 0,
  " 5", " 2", 1,
  " 5", " 4", 3,
  " 5", " 5", 5,
  " 5", " 6", 5,
  " 6", " 2", 2,
  " 6", " 5", 4,
  " 6", " 6", 4,
  " 6", " 7", 2,
  " ", " 1", 5,
  " ", " 4", 1,
```

```

    " ", " 7", 2
)

,            $\bar{v}_i$             $v_{ij}$       mean_centered rating .
centered_rating_df <- rating_df %>%
  group_by(customer) %>%
  mutate(centered_rating = rating - mean(rating)) %>%
  ungroup()

print(centered_rating_df)

## # A tibble: 33 x 4
##   customer item   rating centered_rating
##   <chr>     <chr>   <dbl>          <dbl>
## 1 1         1       5            2.4
## 2 1         3       4            1.4
## 3 1         5       1           -1.6
## 4 1         6       0           -2.6
## 5 1         7       3            0.4
## 6 2         1       4            0.75
## 7 2         2       4            0.75
## 8 2         3       4            0.75
## 9 2         7       1           -2.25
## 10 3        1       5            2
## # ... with 23 more rows

similarity_df <- centered_rating_df %>% filter(customer == " ") %>%
  inner_join(centered_rating_df %>% filter(customer != " "), by = "item") %>%
  group_by(customer.y) %>%
  summarize(similarity = sum(centered_rating.x * centered_rating.y) /
            (sqrt(sum(centered_rating.x ^ 2)) * sqrt(sum(centered_rating.y ^ 2)))) %>%
  rename(customer = customer.y)

print(similarity_df)

## # A tibble: 6 x 2
##   customer similarity
##   <chr>        <dbl>
## 1 1            0.903
## 2 2            0.565
## 3 3            0.961
## 4 4            -0.875
## 5 5            -0.853
## 6 6             1

```

```

    1   normalize .

normalized_similarity_df <- similarity_df %>%
  mutate(normalized_similarity = similarity / sum(abs(similarity)))

print(normalized_similarity_df)

## # A tibble: 6 x 3
##   customer     similarity normalized_similarity
##   <chr>        <dbl>            <dbl>
## 1 1            0.903           0.175
## 2 2            0.565           0.109
## 3 3            0.961           0.186
## 4 4            -0.875          -0.170
## 5 5            -0.853          -0.165
## 6 6            1               0.194

. , j , vij - v̄i = 0 .


$$\hat{v}_{aj} = \bar{v}_a + \frac{1}{\sum_{i=1}^n |w(a, i)|} \sum_{i=1}^n w(a, i)(v_{ij} - \bar{v}_i)$$


items <- sort(setdiff(unique(rating_df$item),
                      rating_df$item[rating_df$customer == " "]))

target_mean <- mean(rating_df$rating[rating_df$customer == " "])

centered_rating_df %>%
  filter(item %in% items) %>%
  inner_join(normalized_similarity_df, by = "customer") %>%
  group_by(item) %>%
  summarize(predicted_rating = target_mean +
            sum(normalized_similarity * centered_rating)) %>%
  arrange(desc(predicted_rating))

## # A tibble: 4 x 2
##   item     predicted_rating
##   <chr>        <dbl>
## 1 3            3.26
## 2 2            3.14
## 3 5            1.96
## 4 6            1.63

, j , j .


$$\hat{v}_{aj} = \bar{v}_a + \frac{1}{\sum_{i:j \in I_i} |w(a, i)|} \sum_{i:j \in I_i} w(a, i)(v_{ij} - \bar{v}_i)$$


```

```

centered_rating_df %>%
  filter(item %in% items) %>%
  inner_join(similarity_df, by = "customer") %>%
  group_by(item) %>%
  summarize(predicted_rating = target_mean +
    sum(similarity * centered_rating) / sum(abs(similarity))) %>%
  arrange(desc(predicted_rating))

## # A tibble: 4 x 2
##   item   predicted_rating
##   <chr>     <dbl>
## 1 3          3.97
## 2 2          3.24
## 3 5          1.87
## 4 6          1.19

```

## 16.4

$$v_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

```

market_basket_df <- tribble(
  ~customer, ~item, ~purchase,
  " 1", " 1", 1,
  " 1", " 3", 1,
  " 1", " 5", 1,
  " 1", " 7", 1,
  " 2", " 1", 1,
  " 2", " 2", 1,
  " 2", " 3", 1,
  " 2", " 7", 1,
  " 3", " 1", 1,
  " 3", " 2", 1,
  " 3", " 4", 1,
  " 3", " 5", 1,
  " 3", " 7", 1,
  " 4", " 1", 1,
  " 4", " 2", 1,
  " 4", " 3", 1,
  " 4", " 4", 1,
  " 4", " 6", 1,
  " 4", " 7", 1,

```

```

    " 5", " 2", 1,
    " 5", " 4", 1,
    " 5", " 5", 1,
    " 5", " 6", 1,
    " 6", " 2", 1,
    " 6", " 5", 1,
    " 6", " 6", 1,
    " 6", " 7", 1,
    " ", " 1", 1,
    " ", " 4", 1,
    " ", " 7", 1
)
,
```

,         $m$      ,      $i$

$$p_i = \frac{|I_i|}{m}$$

· ,  $p_i$                $i$               · ,      $i$     $k$                $p_{ik}$      ·

$$p_{ik} = \frac{|I_i \cap I_k|}{m}$$

$w(a, i)$      .      $a$       $i$      .

$$w(a, i) = \frac{p_{ai} - p_a p_i}{\sqrt{p_a(1-p_a)} \sqrt{p_i(1-p_i)}}$$

```

m <- length(unique(market_basket_df$item))

n_df <- market_basket_df %>%
  group_by(customer) %>%
  summarize(p = n() / m)

common_df <- market_basket_df %>%
  filter(customer == " ") %>%
  inner_join(market_basket_df %>% filter(customer != " "),
             by = "item") %>%
  group_by(customer.y) %>%
  summarize(p = n() / m) %>%
  rename(customer = customer.y)

similarity_df <- crossing(
  target_customer = " ",
  ref_customer = n_df$customer[n_df$customer != " "]
```

```

) %>%
inner_join(n_df %>% rename(target_p = p),
            by = c("target_customer" = "customer")) %>%
inner_join(n_df %>% rename(ref_p = p),
            by = c("ref_customer" = "customer")) %>%
inner_join(common_df %>% rename(common_p = p),
            by = c("ref_customer" = "customer")) %>%
mutate(
  similarity = (common_p - target_p * ref_p) /
    (sqrt(target_p * (1 - target_p)) * sqrt(ref_p * (1 - ref_p)))
)

print(similarity_df)

## # A tibble: 6 x 6
##   target_customer ref_customer target_p ref_p common_p similarity
##   <chr>           <chr>        <dbl>   <dbl>   <dbl>      <dbl>
## 1 1                 1          0.429  0.571   0.286     0.167
## 2 2                 2          0.429  0.571   0.286     0.167
## 3 3                 3          0.429  0.714   0.429     0.548
## 4 4                 4          0.429  0.857   0.429     0.354
## 5 5                 5          0.429  0.571   0.143    -0.417
## 6 6                 6          0.429  0.571   0.143    -0.417

.     a     j     .

```

$$P_{aj} = \frac{\sum_{i=1}^n w(a, i)v_{ij}}{\sum_{i=1}^n |w(a, i)|}$$

```

denom <- sum(abs(similarity_df$similarity))

pred_df <- similarity_df %>%
  inner_join(market_basket_df, by = c("ref_customer" = "customer")) %>%
  anti_join(market_basket_df %>%
              filter(customer == " ") %>%
              select(item),
             by = "item") %>%
  group_by(item) %>%
  summarize(est_score = sum(similarity * purchase) / denom) %>%
  arrange(desc(est_score))

print(pred_df)

## # A tibble: 4 x 2
##   item   est_score
##   <chr>     <dbl>
## 1 1         0.167
## 2 2         0.167
## 3 3         0.548
## 4 4         0.354

```

```
## 1   3   0.332
## 2   2   0.113
## 3   5  -0.0575
## 4   6  -0.232
```

# Bibliography

- Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD '93, pages 207–216, New York, NY, USA. ACM.
- Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.
- Agrawal, R., Srikant, R., et al. (1995). Mining sequential patterns. In *icde*, volume 95, pages 3–14.
- Banfield, J. D. and Raftery, A. E. (1993). Model-based gaussian and non-gaussian clustering. *Biometrics*, pages 803–821.
- Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.
- Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27.
- Cavanaugh, J., Hatch, R., and Sullivan, J. (1976). Models for the subjective effects of loss, noise, and talker echo on telephone connections. *Bell System Technical Journal*, 55(9):1319–1371.
- Celeux, G. and Govaert, G. (1995). Gaussian parsimonious clustering models. *Pattern recognition*, 28(5):781–793.
- Chang, C.-C. and Lin, C.-J. (2001). Training v-support vector classifiers: theory and algorithms. *Neural computation*, 13(9):2119–2147.
- Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.
- Czepiel, S. A. (2002). Maximum likelihood estimation of logistic regression models: theory and implementation. Available at [czep.net/stat/mlelr.pdf](http://czep.net/stat/mlelr.pdf).

- Dunn, J. C. (1973). A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters.
- Fan, R.-E., Chen, P.-H., and Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of machine learning research*, 6(Dec):1889–1918.
- Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical programming*, 27(1):1–33.
- Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, pages 857–871.
- Handl, J. and Knowles, J. (2005). Exploiting the trade-off—the benefits of multiple objectives in data clustering. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 547–560. Springer.
- Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.
- Kaufman, L. and Rousseeuw, P. J. (1990). Finding groups in data: an introduction to cluster analysis.
- Lance, G. N. and Williams, W. T. (1967). A general theory of classificatory sorting strategies: 1. hierarchical systems. *The computer journal*, 9(4):373–380.
- Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341.
- Pratt, J. W. (1981). Concavity of the log likelihood. *Journal of the American Statistical Association*, 76(373):103–106.
- Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.
- Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.
- Schölkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. (2000). New support vector algorithms. *Neural computation*, 12(5):1207–1245.
- Scrucca, L., Fop, M., Murphy, T. B., and Raftery, A. E. (2016). mclust 5: Clustering, classification and density estimation using gaussian finite mixture models. *The R journal*, 8(1):289.
- Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.
- Wishart, D. (1969). 256. note: An algorithm for hierarchical classifications. *Biometrics*, pages 165–170.

(2012).