R-Ladies Taipei

維度縮減
Reducing Data Dimensionality

Why

- Large data (p >>n)
- Computation time
- We may want simpler models
- Etc.

Some strategies (I)

- Reduce the number of variables (feature selection)
- Reduce the number of cases
  - resampling
- Reduce the number of values on the variables
  - Grouping values (k-means method, equal-size groups, equal-frequency groups, etc)

Some strategies (II)

- Reduce the number of variables (feature selection)
  - PCA (numerical variables)
  - PCAmix (categorical variables & numerical variables)
  - Package: Caret
  - Package: Boruta

PCA method

- prcomp()：主成份分析的基本函式
- plot()：繪製陡坡圖(screet plot)，選擇多少個主成份
- dotchart()：繪製主成份負荷圖(PCA loadings plot)

https://rpubs.com/skydome20/R-Note7-PCA

Package: PCAmixdata

- 語法:

PCAmix(X.quanti = df[,1:20], #a numeric matrix of data
X.quali = df[,21-30], #a categorical matrix of data
ndim = 5, #number of dimensions kept in the results (default = 5)
rename.level = FALSE,
weight.col.quanti = NULL,
weight.col.quali = NULL,
graph = TRUE)

Package: PCAmixdata

```r
library(PCAmixdata)
data(wine)
str(wine)
X.quanti <- splitmix(wine)$X.quanti
X.quali <- splitmix(wine)$X.quali
pca<-PCAmix(X.quanti[,1:27],X.quali,ndim=4)
pca<-PCAmix(X.quanti[,1:27],X.quali,ndim=4,graph=FALSE)
pca$eig
pca$ind$coord
```

https://rdrr.io/cran/PCAmixdata/man/PCAmix.html

Package: Caret

- **C**lassification **A**nd **RE**gression **T**raining (caret)

- caret provides you with essential tools for:
  - Data preparation, including: imputation, centering/scaling data, removing correlated predictors, reducing skewness
  - Data splitting
  - Model evaluation
  - Variable selection

https://topepo.github.io/caret/

Data description

- Example: HR Employee Attrition and Performance
  - Download: https://www.ibm.com/communities/analytics/watson-analytics-blog/hr-employee-attrition/
- Sample size: 1470(row) x 35(column)
- Target variable: attrition
- Excluding variable: EmployeeCount, EmployeeNumber, JobRole,over18, StandardHours

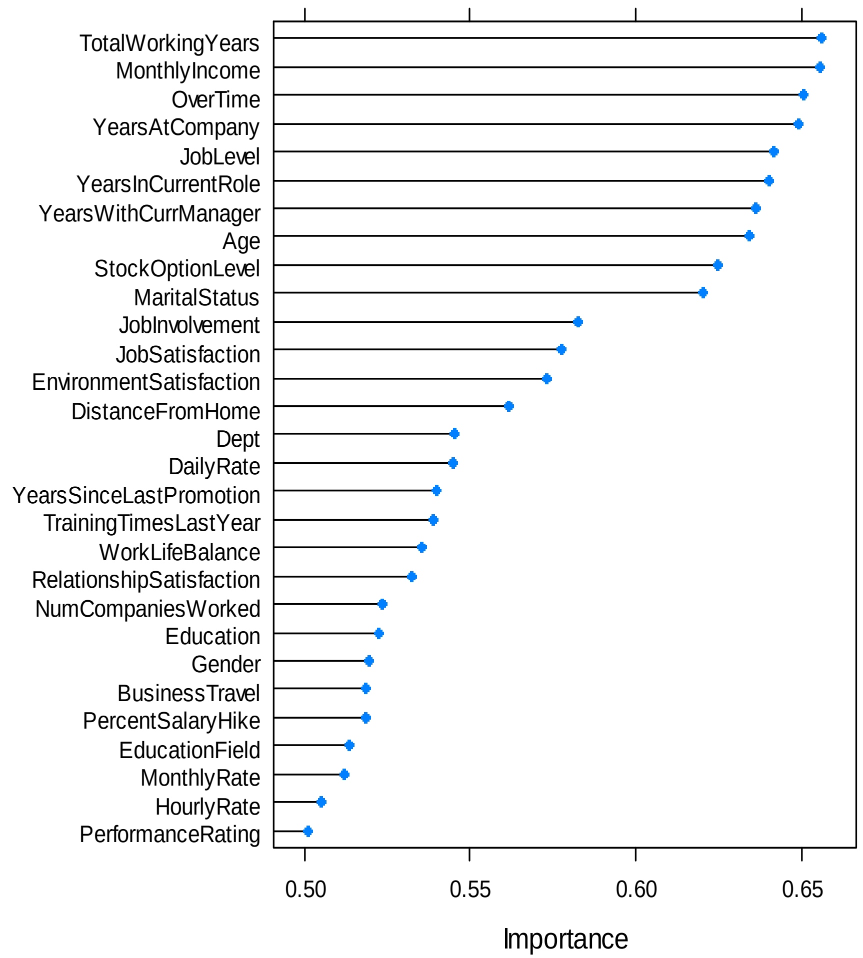
Caret package-feature selection

```
library(mlbench)
library(caret)
library(e1071)

# Load the data
setwd("I://data preprocess")
dat0 <- read.csv("HR_InputData.csv",header=TRUE)

dat1 <- as.numeric(dat0[,-c(2:7)])

#Method: Rank Features By Importance
# prepare training scheme
crtl <- trainControl(method="repeatedcv", number=10, repeats=3)

# train the model
model <- train(Attrition~., data=dat0, method="lvq", preProcess="scale", trControl=crtl)
# estimate variable importance
importance <- varImp(model, scale=FALSE)
# summarize importance
print(importance)
# plot importance
plot(importance)
```

lvq: Learning vector quantization

output

	Importance
TotalWorkingYears	0.6559
MonthlyIncome	0.6557
OverTime	0.6507
YearsAtCompany	0.6490
JobLevel	0.6416
YearsInCurrentRole	0.6400
YearsWithCurrManager	0.6360
Age	0.6343
StockOptionLevel	0.6249
MaritalStatus	0.6202
JobInvolvement	0.5827
JobSatisfaction	0.5778
EnvironmentSatisfaction	0.5730
DistanceFromHome	0.5620
Dept	0.5451
DailyRate	0.5447
YearsSinceLastPromotion	0.5402
TrainingTimesLastYear	0.5388
WorkLifeBalance	0.5356
RelationshipSatisfaction	0.5323



Caret package-feature selection (II)

```
#Method: Feature Selection : recursive feature elimination (RFE)

# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)

# run the RFE algorithm
results <- rfe (dat1[,2:24], dat1[,1], sizes=c(1:23), rfeControl=control)

# summarize the results
print(results)

# list the chosen features
predictors(results)

# plot the results
plot(results, type=c("g", "o"))
```

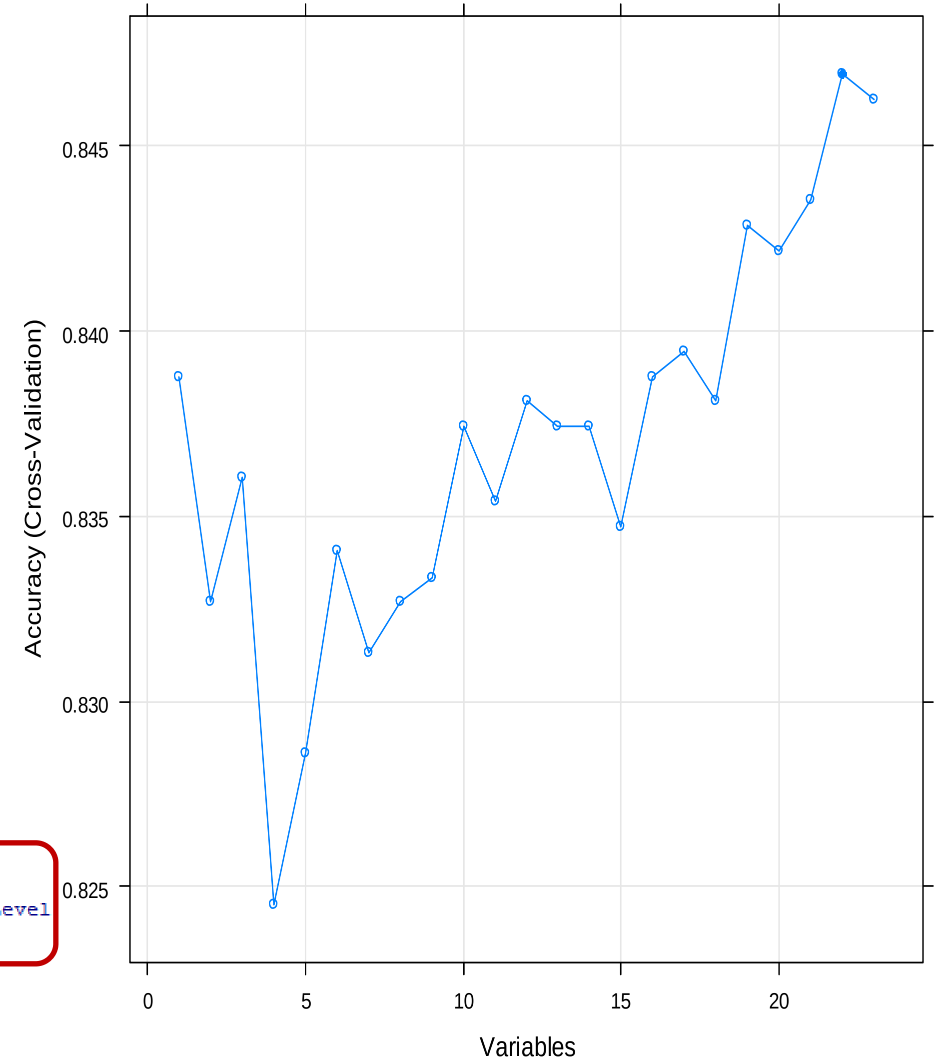
dependent variable

Independent variables

num of ind. variable

Output-(1)

```
Recursive feature selection

Outer resampling method: Cross-Validated (10 fold)

Resampling performance over subset size:

 Variables Accuracy   Kappa AccuracySD KappaSD Selected
         1   0.8388 0.03708   0.009665 0.06772
         2   0.8327 0.02982   0.018354 0.08189
         3   0.8361 0.07856   0.011629 0.07977
         4   0.8245 0.15311   0.025015 0.12087
         5   0.8286 0.15781   0.021858 0.10729
         6   0.8341 0.14338   0.020716 0.09409
         7   0.8313 0.14116   0.020716 0.09629
         8   0.8327 0.14859   0.018886 0.08640
         9   0.8333 0.17591   0.019967 0.09480
        10   0.8374 0.18487   0.018373 0.09390
        11   0.8354 0.16867   0.019241 0.10173
        12   0.8381 0.16087   0.014885 0.06428
        13   0.8374 0.15012   0.014460 0.07212
        14   0.8374 0.14904   0.013931 0.07106
        15   0.8347 0.10976   0.012453 0.05100
        16   0.8388 0.14883   0.012987 0.04522
        17   0.8394 0.14292   0.012770 0.04951
        18   0.8381 0.13931   0.013022 0.05571
        19   0.8428 0.14943   0.014897 0.07600
        20   0.8422 0.13522   0.014729 0.07185
        21   0.8435 0.15207   0.011249 0.04883
        22   0.8469 0.15907   0.010834 0.05234        *
        23   0.8462 0.14449   0.012270 0.06219

The top 5 variables (out of 22):
   Age, StockOptionLevel, TotalWorkingYears, MonthlyIncome, JobLevel
```



Output-(2)

> predictors(results)

```
predictors(results)
 [1] "Age"                          "StockOptionLevel"
 [3] "TotalWorkingYears"            "MonthlyIncome"
 [5] "JobLevel"                     "YearsWithCurrManager"
 [7] "YearsAtCompany"               "JobInvolvement"
 [9] "YearsInCurrentRole"           "JobSatisfaction"
11] "EnvironmentSatisfaction"       "NumCompaniesWorked"
13] "WorkLifeBalance"               "DistanceFromHome"
15] "YearsSinceLastPromotion"       "PercentSalaryHike"
17] "DailyRate"                     "RelationshipSatisfaction"
19] "HourlyRate"                    "PerformanceRating"
21] "TrainingTimesLastYear"         "MonthlyRate"
```

Boruta package

- Program:

```
library(Boruta)
set.seed(123)
boruta.train <- Boruta(Attrition~.-Attrition, data = dat1, doTrace = 2)
print(boruta.train)
```

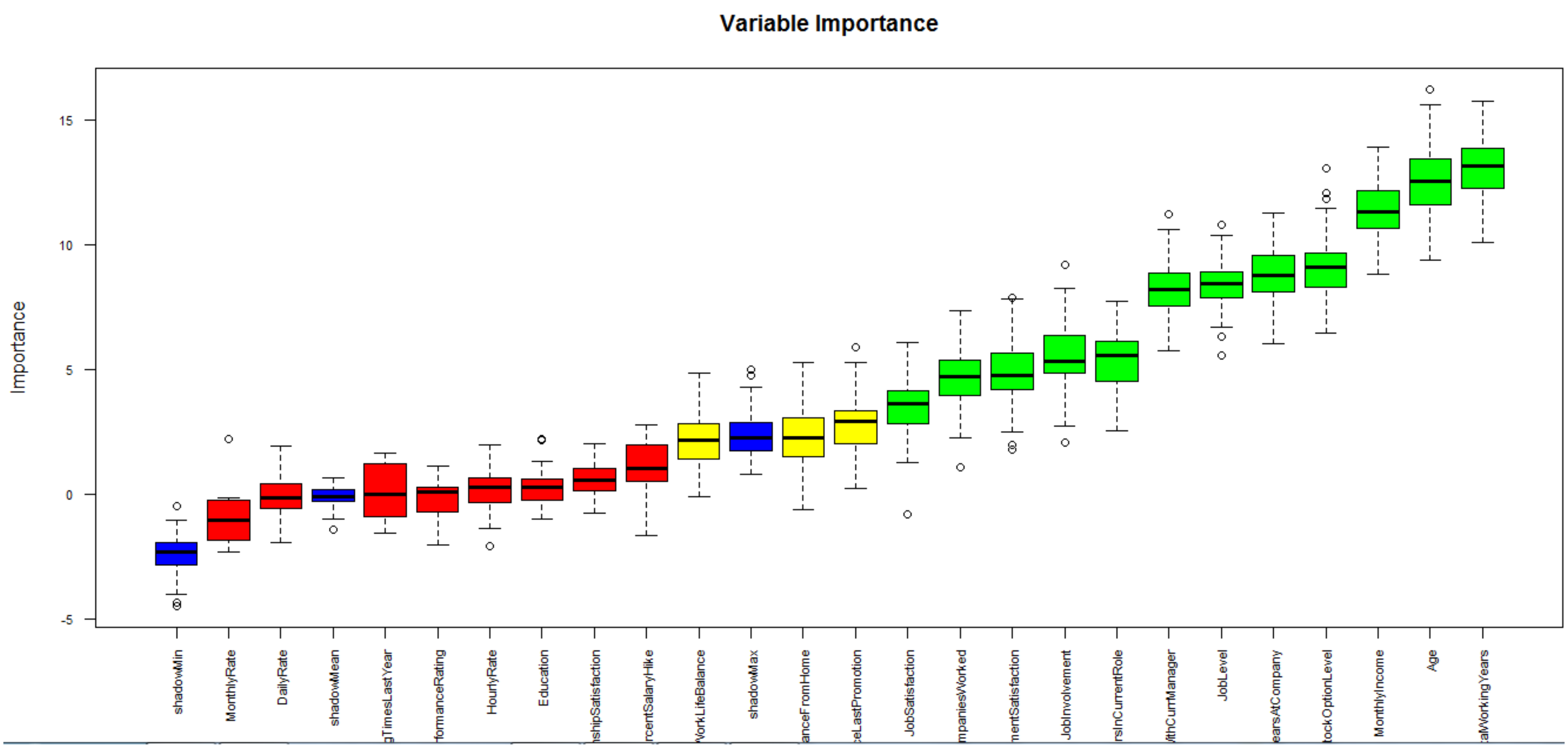
Output

```
> print(boruta.train)
Boruta performed 99 iterations in 34.87741 secs.
 12 attributes confirmed important: Age, EnvironmentSatisfaction,
JobInvolvement, JobLevel, JobSatisfaction and 7 more;
 8 attributes confirmed unimportant: DailyRate, Education,
HourlyRate, MonthlyRate, PercentSalaryHike and 3 more;
 3 tentative attributes left: DistanceFromHome, WorkLifeBalance,
YearsSinceLastPromotion;
```

Plot importance chart

- Program:

```
plot(boruta.train, cex.axis=.7, las=2, xlab="", main="Variable Importance")
```



Get the selected attributes

- Program:

getSelectedAttributes(boruta.train, withTentative = F)

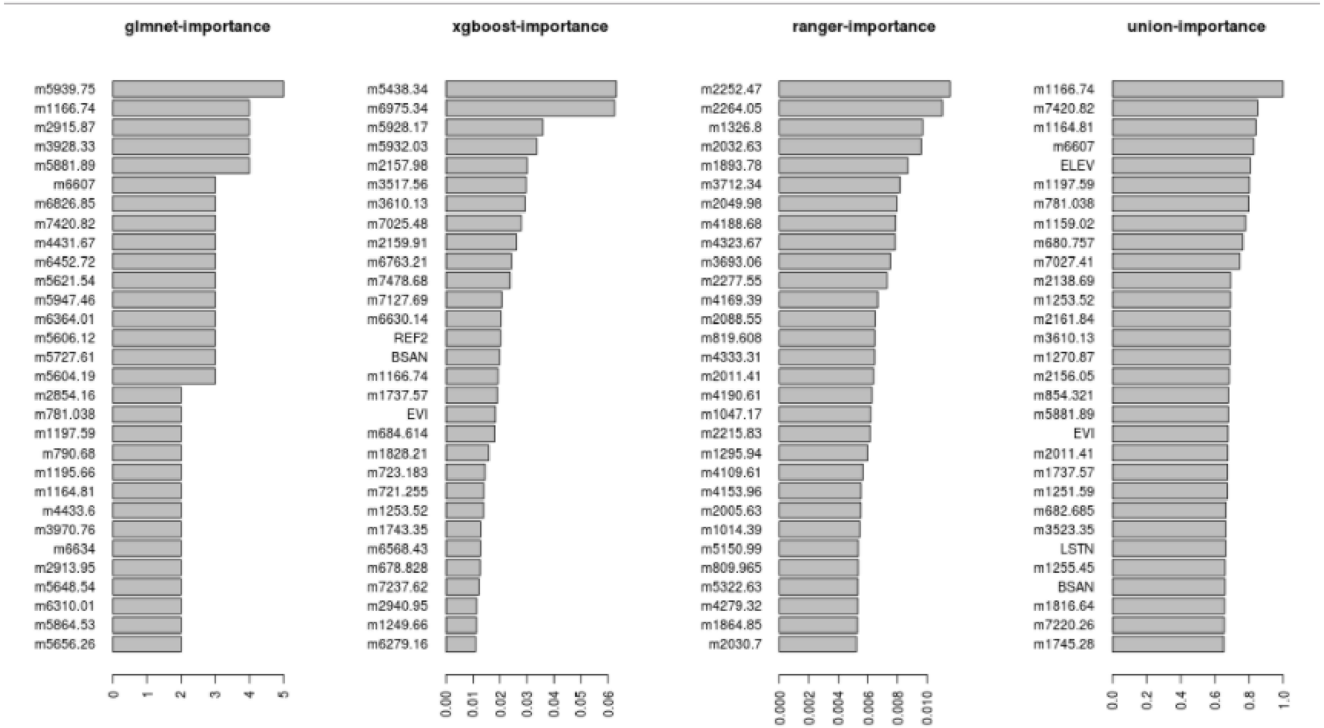
Output

```
> getSelectedAttributes(boruta.train, withTentative = F)
 [1] "Age"                       "EnvironmentSatisfaction"
 [3] "JobInvolvement"            "JobLevel"
 [5] "JobSatisfaction"           "MonthlyIncome"
 [7] "NumCompaniesWorked"        "StockOptionLevel"
 [9] "TotalWorkingYears"         "YearsAtCompany"
[11] "YearsInCurrentRole"        "YearsWithCurrManager"
> 
```

R-Ladies Taipei

Other approach

- feature selection using lasso, boosting and random forest
- http://mlampros.github.io/2016/02/14/feature-selection/



Appendix

PCA 指令 example

- pca <- prcomp(
    formula = ~ H1B+H2B+H3B+HR+RBI+SB+BB, *#選擇七個變數*
    data = data, *# 資料*
    scale = TRUE)

###output

```
## Standard deviations (1, .., p=7):
## [1] 1.4222856 1.3785035 1.0108522 0.9578441 0.7700729 0.7131148 0.1897347
##
## Rotation (n x k) = (7 x 7):
##             PC1        PC2         PC3          PC4         PC5
## H1B -0.40991503 -0.4681242  0.07174689  0.056704066 -0.07882016
## H2B -0.51441491 -0.2004156  0.01669591  0.255448162 -0.46809834
## H3B  0.01853759 -0.5595940 -0.19427151 -0.004051477  0.71490431
## HR  -0.34336124  0.5417488 -0.03416307 -0.394140194  0.26396281
## RBI -0.64629912  0.1016251 -0.25396353 -0.156840299  0.20084751
## SB   0.16722000 -0.2741655 -0.52853255 -0.679207860 -0.39181790
## BB   0.05866272  0.2203673 -0.78218985  0.538744635 -0.00676713
##             PC6         PC7
## H1B -0.66701643 -0.39159028
## H2B  0.62315846 -0.14911690
## H3B  0.34921449 -0.12535585
## HR   0.10991843 -0.59189466
## RBI -0.12239863  0.65387482
## SB   0.04037564 -0.03239357
## BB  -0.12695740 -0.17252886
```

https://rpubs.com/skydome20/R-Note7-PCA

陡坡圖(Scree plot)

- *# 使用plot()函式*

```
plot(pca,                  # 放pca
    type="line", # 用直線連結每個點
    main= "scree plot") # 主標題
#用藍線標示出特徵值=1的地方
    abline(h=1, col="blue")
```

https://rpubs.com/skydome20/R-Note7-PCA

- 求出每個主成份的特徵值(也就是variance = std^2)

  vars <- (pca$sdev)^2

  vars

2.02289644 1.90027181 1.02182222 0.91746533 0.59301228 0.50853268 0.03599925

- 計算每個主成分的解釋比例 = 各個主成份的特徵值/總特徵值

props <- vars / sum(vars) props

0.288985205 0.271467401 0.145974603 0.131066475 0.084716040 0.072647526 0.005142749

https://rpubs.com/skydome20/R-Note7-PCA

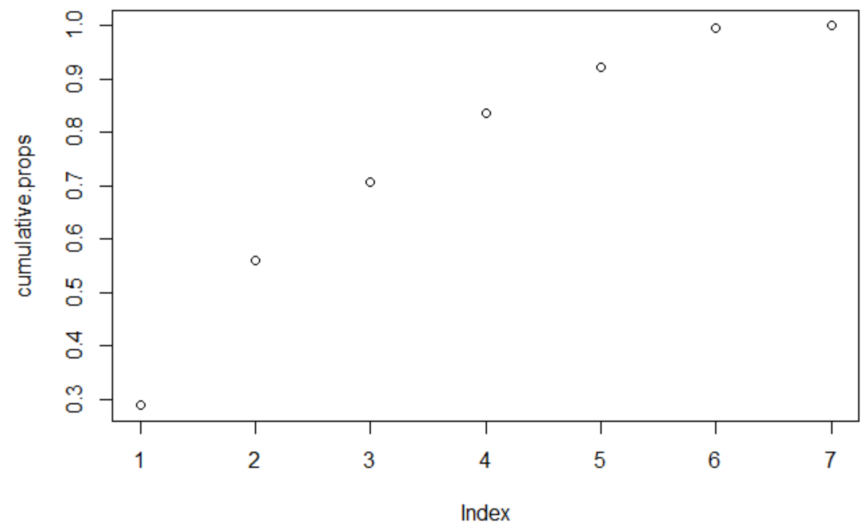
- 累加每個主成份的解釋比例

```
cumulative.props <- cumsum(props) # 累加前n個元素的值
cumulative.props
```

```
## [1] 0.2889852 0.5604526 0.7064272 0.8374937 0.9222097 0.9948573 1.0000000
```

- *# 累積解釋比例圖*

```
plot(cumulative.props)
```



- **取前三個主成份**，作為新的資料集：
- *# pca$rotation*

```
top3_pca.data <- pca$x[, 1:3]
top3_pca.data
```

 # 特徵向量(原變數的線性組合)

```
pca$rotation
```

```
##             PC1        PC2         PC3          PC4         PC5
## H1B -0.40991503 -0.4681242  0.07174689  0.056704066 -0.07882016
## H2B -0.51441491 -0.2004156  0.01669591  0.255448162 -0.46809834
## H3B  0.01853759 -0.5595940 -0.19427151 -0.004051477  0.71490431
## HR  -0.34336124  0.5417488 -0.03416307 -0.394140194  0.26396281
## RBI -0.64629912  0.1016251 -0.25396353 -0.156840299  0.20084751
## SB   0.16722000 -0.2741655 -0.52853255 -0.679207860 -0.39181790
## BB   0.05866272  0.2203673 -0.78218985  0.538744635 -0.00676713
##             PC6         PC7
## H1B -0.66701643 -0.39159028
## H2B  0.62315846 -0.14911690
## H3B  0.34921449 -0.12535585
## HR   0.10991843 -0.59189466
## RBI -0.12239863  0.65387482
## SB   0.04037564 -0.03239357
## BB  -0.12695740 -0.17252886
```

```
##            PC1         PC2         PC3
## 1  -2.65536140  0.04641055  0.05124254
## 2  -1.37712847  0.01360254  0.24495540
## 3  -1.57754875 -1.72554295  0.14807629
## 4  -1.76751032 -0.84074064 -0.75258974
## 5  -0.44097214 -3.66454431 -0.36109275
## 6  -1.08166263 -0.10861369  0.27429500
## 7  -0.45474791 -2.65730709  1.13595923
## 8  -2.46449798  0.03093137  1.32250201
## 9  -0.11216989 -1.29948488 -0.83784413
## 10 -1.01441489  0.55336109  0.51565842
## 11 -1.53519975  3.14421085 -1.02043498
## 12 -1.20736887 -0.28285945 -1.24157398
## 13 -1.06901074  0.22513361 -0.72630319
## 14  0.01063384 -0.25128338  0.55678238
## 15 -0.25896225  1.04428266  0.46364492
## 16 -0.32397454  0.64019528  0.36177440
## 18  0.41732553  0.71111234 -1.21591003
## 19  1.04120451  0.11986429 -0.63306531
## 20  1.39709979 -0.53060810  0.67003668
## 21  1.64046682 -1.54053971 -1.66879107
## 22 -0.51860317  2.45956726  1.36247833
## 23  1.10286061  0.60454851  1.70393647
## 24  1.91924719 -1.14973539 -0.64690007
## 25  0.84451316  1.55315562  0.06346943
## 26  1.74614534 -0.60978067  1.36484924
## 27  1.23431051  0.91240050 -2.16643033
## 28  2.60386762  0.08178357  0.87401532
## 29  1.00595356  1.50672401 -1.44179251
## 30  2.43463278  0.39714752  0.92229467
```

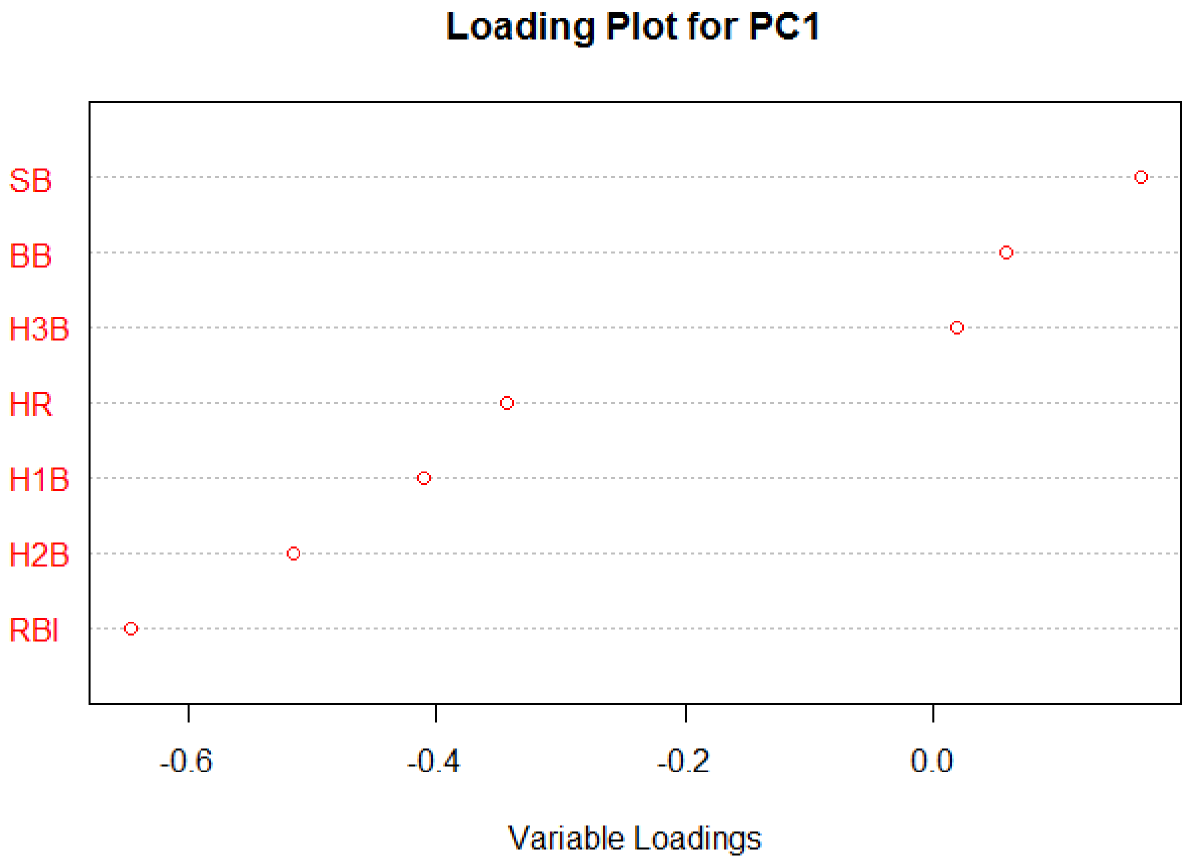
取前三個主成份的特徵向量

```
top3.pca.eigenvector <- pca$rotation[, 1:3]
top3.pca.eigenvector
```

```
##             PC1        PC2         PC3
## H1B -0.40991503 -0.4681242  0.07174689
## H2B -0.51441491 -0.2004156  0.01669591
## H3B  0.01853759 -0.5595940 -0.19427151
## HR  -0.34336124  0.5417488 -0.03416307
## RBI -0.64629912  0.1016251 -0.25396353
## SB   0.16722000 -0.2741655 -0.52853255
## BB   0.05866272  0.2203673 -0.78218985
```

- 繪製主成份負荷圖，觀察原變數和主成份之間的關係：

```
first.pca <- top3.pca.eigenvector[ ,1]
second.pca <- top3.pca.eigenvector[ ,2]
third.pca <- top3.pca.eigenvector[ ,3]
first.pca[order(first.pca, decreasing=FALSE)]

dotchart(first.pca[order(first.pca, decreasing=FALSE)]),
            main = “Loading plot for PCA”,
            xlab = “variable loading”,
            col =“red”)
```


Loading Plot for PC1
SB
BB
H3B
HR
H1B
H2B
RBI
-0.6
-0.4
-0.2
0.0
Variable Loadings