Laporan Praktikum 6 AMP

Antonius Aditya Rizky Wijaya

G5402221003

2025-02-27

Linear Models and Regularization Methods

Kode halaman 268:

```
library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.3.3

names(Hitters)

##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"

dim(Hitters)

## [1] 322  20

sum(is.na(Hitters$Salary))

## [1] 59

Hitters <- na.omit(Hitters)
dim(Hitters)

## [1] 263  20

sum(is.na(Hitters))

## [1] 0
```

Kode halaman 275

```
x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary
```

Kode halaman 276

```
set.seed(1)
train <- sample(1:nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]
```

PCR and PLS Regression

Principal Components Regression

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.3.3
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

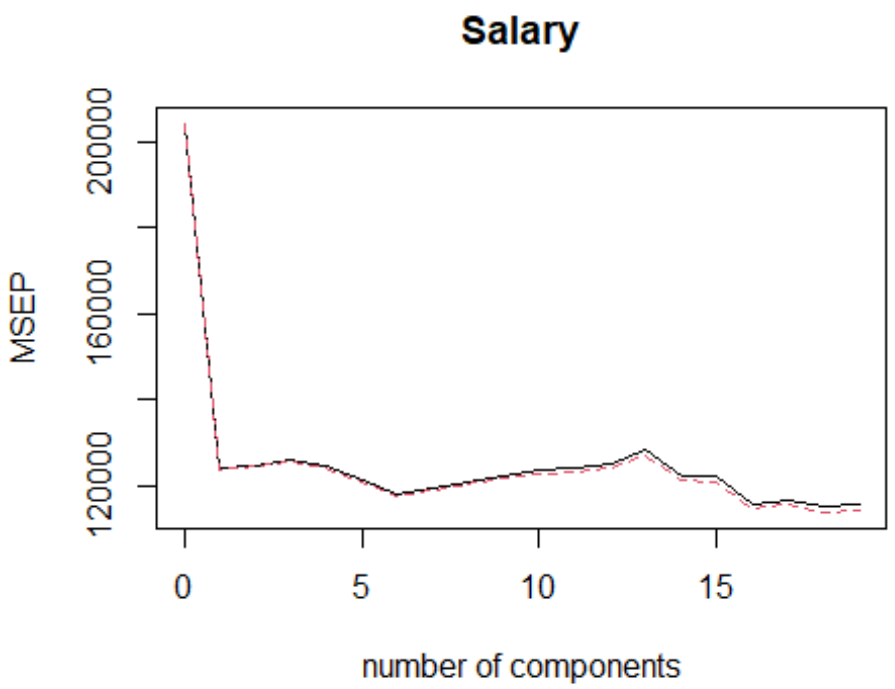
```
set.seed(2)
pcr.fit <- pcr(Salary ~ ., data = Hitters, scale = TRUE,
    validation = "CV")
```

```
summary(pcr.fit)
```

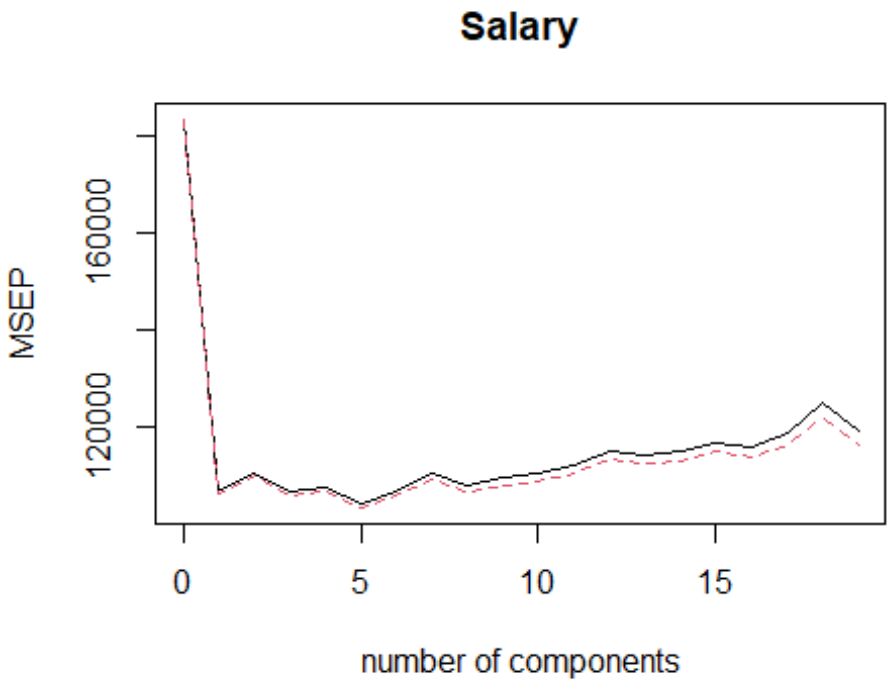
```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    351.9    353.2    355.0    352.8    348.4    343.6
## adjCV          452    351.6    352.7    354.4    352.1    347.6    342.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.5    347.7    349.6     351.4     352.1     353.5     358.2
## adjCV    344.7    346.7    348.5     350.1     350.7     352.0     356.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        349.7     349.4     339.9     341.6     339.2     339.6
## adjCV     348.0     347.7     338.2     339.7     337.2     337.6
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8
comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26
94.96
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69
46.75
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15
comps
## X         96.28     97.26     97.98     98.65     99.15     99.47
99.75
## Salary    46.86     47.76     47.82     47.85     48.10     50.40
50.55
##         16 comps  17 comps  18 comps  19 comps
```

```
## X          99.89     99.97     99.99    100.00
## Salary     53.01     53.85     54.61     54.61
```

```
validationplot(pcr.fit, val.type = "MSEP")
```




```
set.seed(1)
pcr.fit <- pcr(Salary ~ ., data = Hitters, subset = train,
    scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```



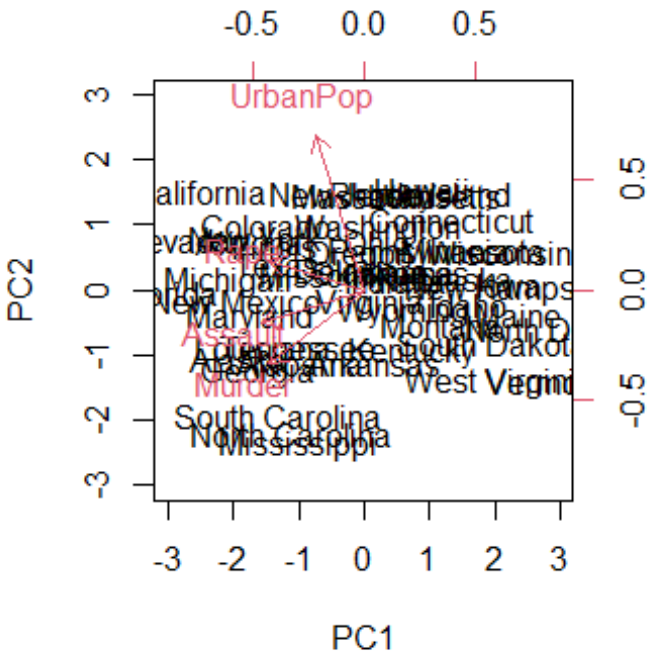

```
pcr.pred <- predict(pcr.fit, x[test, ], ncomp = 5)
mean((pcr.pred - y.test)^2)

## [1] 142811.8

pcr.fit <- pcr(y ~ x, scale = TRUE, ncomp = 5)
summary(pcr.fit)

## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps
## X    38.31    60.16    70.84    79.03    84.29
## y    40.63    41.58    42.17    43.22    44.90
```

Unsupervised Learning

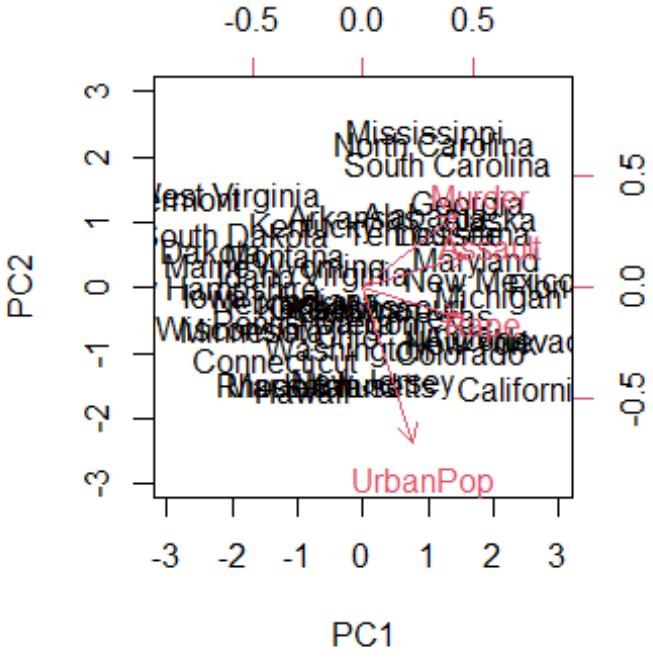
Principal Components Analysis

```
states <- row.names(USArrests)
states

##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
```

```
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"

names(USArrests)

## [1] "Murder"   "Assault"  "UrbanPop" "Rape"

apply(USArrests, 2, mean)

##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232

apply(USArrests, 2, var)

##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916

pr.out <- prcomp(USArrests, scale = TRUE)

names(pr.out)

## [1] "sdev"     "rotation" "center"   "scale"    "x"

pr.out$center

##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232

pr.out$scale

##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385

pr.out$rotation

##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995 -0.4181809  0.3412327  0.64922780
## Assault  -0.5831836 -0.1879856  0.2681484 -0.74340748
## UrbanPop -0.2781909  0.8728062  0.3780158  0.13387773
## Rape     -0.5434321  0.1673186 -0.8177779  0.08902432

dim(pr.out$x)

## [1] 50  4

biplot(pr.out, scale = 0)
```




```
pr.out$rotation = -pr.out$rotation
pr.out$x = -pr.out$x
biplot(pr.out, scale = 0)
```



```
pr.out$sdev
```

```
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
```

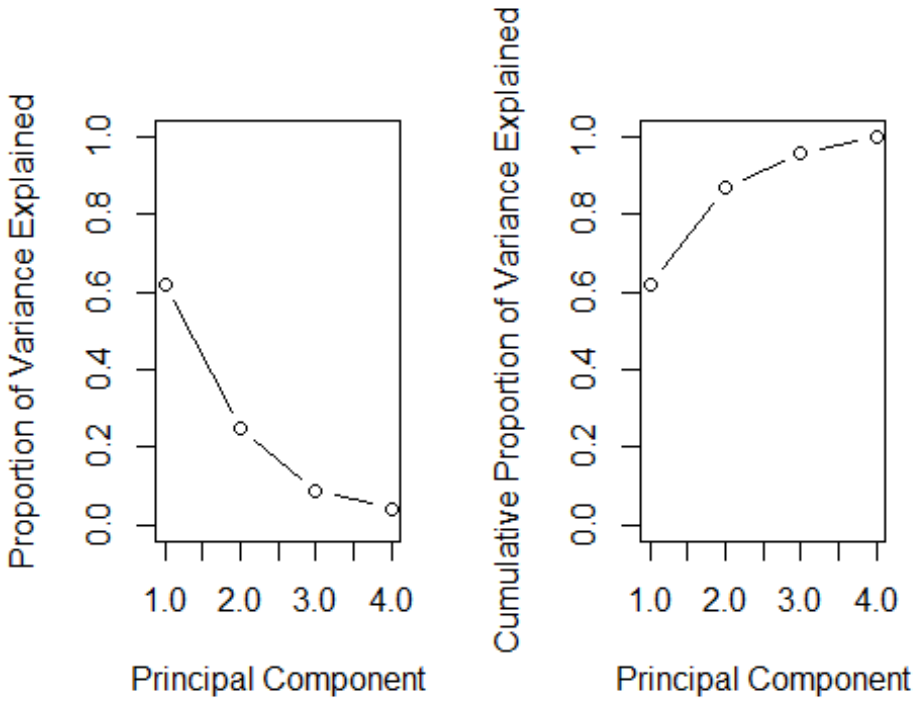
```
pr.var <- pr.out$sdev^2
pr.var
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

```
pve <- pr.var / sum(pr.var)
pve
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

```
par(mfrow = c(1, 2))
plot(pve, xlab = "Principal Component",
    ylab = "Proportion of Variance Explained", ylim = c(0, 1),
    type = "b")
plot(cumsum(pve), xlab = "Principal Component",
    ylab = "Cumulative Proportion of Variance Explained",
    ylim = c(0, 1), type = "b")
```




```
a <- c(1, 2, 8, -3)
cumsum(a)
```

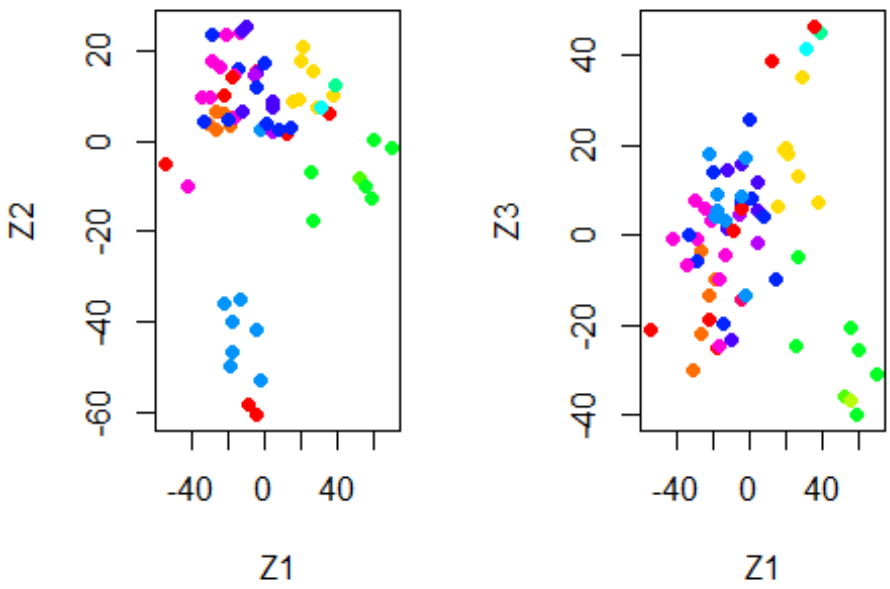
```
## [1]  1  3 11  8
```

NCI60 Data Example

```
library(ISLR2)
nci.labs <- NCI60$labs
nci.data <- NCI60$data

dim(nci.data)

## [1]   64 6830

nci.labs[1:4]

## [1] "CNS"   "CNS"   "CNS"   "RENAL"

table(nci.labs)

## nci.labs
##      BREAST         CNS       COLON K562A-repro K562B-repro    LEUKEMIA
##           7           5           7           1           1           6
## MCF7A-repro MCF7D-repro    MELANOMA       NSCLC     OVARIAN    PROSTATE
##           1           1           8           9           6           2
##       RENAL     UNKNOWN
##           9           1
```

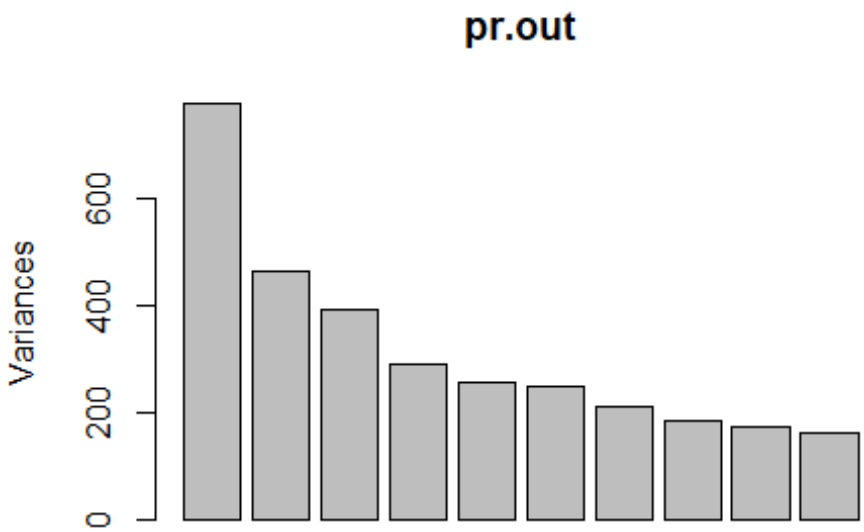
PCA on the NCI60 Data

```
pr.out <- prcomp(nci.data, scale = TRUE)

Cols <- function(vec) {
   cols <- rainbow(length(unique(vec)))
   return(cols[as.numeric(as.factor(vec))])
 }

par(mfrow = c(1, 2))
plot(pr.out$x[, 1:2], col = Cols(nci.labs), pch = 19,
    xlab = "Z1", ylab = "Z2")
plot(pr.out$x[, c(1, 3)], col = Cols(nci.labs), pch = 19,
    xlab = "Z1", ylab = "Z3")
```

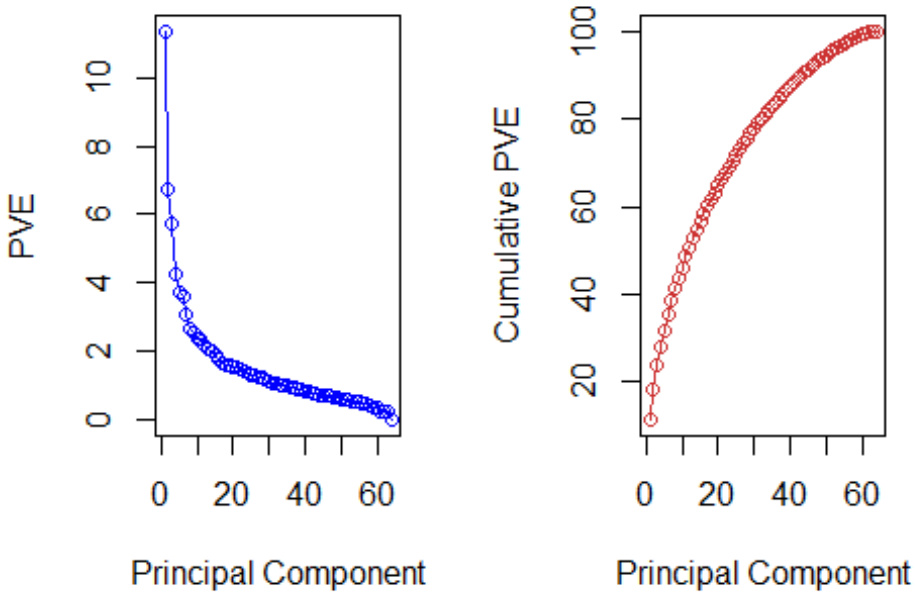



```
summary(pr.out)

## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5
PC6
## Standard deviation     27.8535 21.48136 19.82046 17.03256 15.97181
15.72108
## Proportion of Variance  0.1136  0.06756  0.05752  0.04248  0.03735
0.03619
## Cumulative Proportion   0.1136  0.18115  0.23867  0.28115  0.31850
0.35468
##                             PC7      PC8      PC9     PC10     PC11
PC12
## Standard deviation     14.47145 13.54427 13.14400 12.73860 12.68672
12.15769
## Proportion of Variance  0.03066  0.02686  0.02529  0.02376  0.02357
0.02164
## Cumulative Proportion   0.38534  0.41220  0.43750  0.46126  0.48482
0.50646
##                            PC13     PC14     PC15     PC16     PC17
PC18
## Standard deviation     11.83019 11.62554 11.43779 11.00051 10.65666
10.48880
## Proportion of Variance  0.02049  0.01979  0.01915  0.01772  0.01663
0.01611
## Cumulative Proportion   0.52695  0.54674  0.56590  0.58361  0.60024
0.61635
```

```
##                            PC19    PC20     PC21    PC22    PC23    PC24
## Standard deviation     10.43518 10.3219 10.14608 10.0544 9.90265 9.64766
## Proportion of Variance  0.01594  0.0156  0.01507  0.0148 0.01436 0.01363
## Cumulative Proportion   0.63229  0.6479  0.66296  0.6778 0.69212 0.70575
##                            PC25    PC26    PC27   PC28    PC29    PC30
PC31
## Standard deviation     9.50764 9.33253 9.27320 9.0900 8.98117 8.75003
8.59962
## Proportion of Variance 0.01324 0.01275 0.01259 0.0121 0.01181 0.01121
0.01083
## Cumulative Proportion  0.71899 0.73174 0.74433 0.7564 0.76824 0.77945
0.79027
##                            PC32    PC33    PC34    PC35    PC36    PC37
PC38
## Standard deviation     8.44738 8.37305 8.21579 8.15731 7.97465 7.90446
7.82127
## Proportion of Variance 0.01045 0.01026 0.00988 0.00974 0.00931 0.00915
0.00896
## Cumulative Proportion  0.80072 0.81099 0.82087 0.83061 0.83992 0.84907
0.85803
##                            PC39    PC40    PC41   PC42    PC43   PC44
PC45
## Standard deviation     7.72156 7.58603 7.45619 7.3444 7.10449 7.0131
6.95839
## Proportion of Variance 0.00873 0.00843 0.00814 0.0079 0.00739 0.0072
0.00709
## Cumulative Proportion  0.86676 0.87518 0.88332 0.8912 0.89861 0.9058
0.91290
##                           PC46    PC47    PC48    PC49    PC50    PC51
PC52
## Standard deviation     6.8663 6.80744 6.64763 6.61607 6.40793 6.21984
6.20326
## Proportion of Variance 0.0069 0.00678 0.00647 0.00641 0.00601 0.00566
0.00563
## Cumulative Proportion  0.9198 0.92659 0.93306 0.93947 0.94548 0.95114
0.95678
##                            PC53    PC54    PC55    PC56    PC57   PC58
PC59
## Standard deviation     6.06706 5.91805 5.91233 5.73539 5.47261 5.2921
5.02117
## Proportion of Variance 0.00539 0.00513 0.00512 0.00482 0.00438 0.0041
0.00369
## Cumulative Proportion  0.96216 0.96729 0.97241 0.97723 0.98161 0.9857
0.98940
##                            PC60    PC61    PC62    PC63      PC64
## Standard deviation     4.68398 4.17567 4.08212 4.04124 1.951e-14
## Proportion of Variance 0.00321 0.00255 0.00244 0.00239 0.000e+00
## Cumulative Proportion  0.99262 0.99517 0.99761 1.00000 1.000e+00

plot(pr.out)
```




```r
pve <- 100 * pr.out$sdev^2 / sum(pr.out$sdev^2)
par(mfrow = c(1, 2))
plot(pve,  type = "o", ylab = "PVE",
    xlab = "Principal Component", col = "blue")
plot(cumsum(pve), type = "o", ylab = "Cumulative PVE",
    xlab = "Principal Component", col = "brown3")
```



(Subbab 6.6) Exercises Linear Models and Regularization Methods

Nomor 9

In this exercise, we will predict the number of applications received using the other variables in the `College` data set.

a. Split the data set into a training set and a test set.

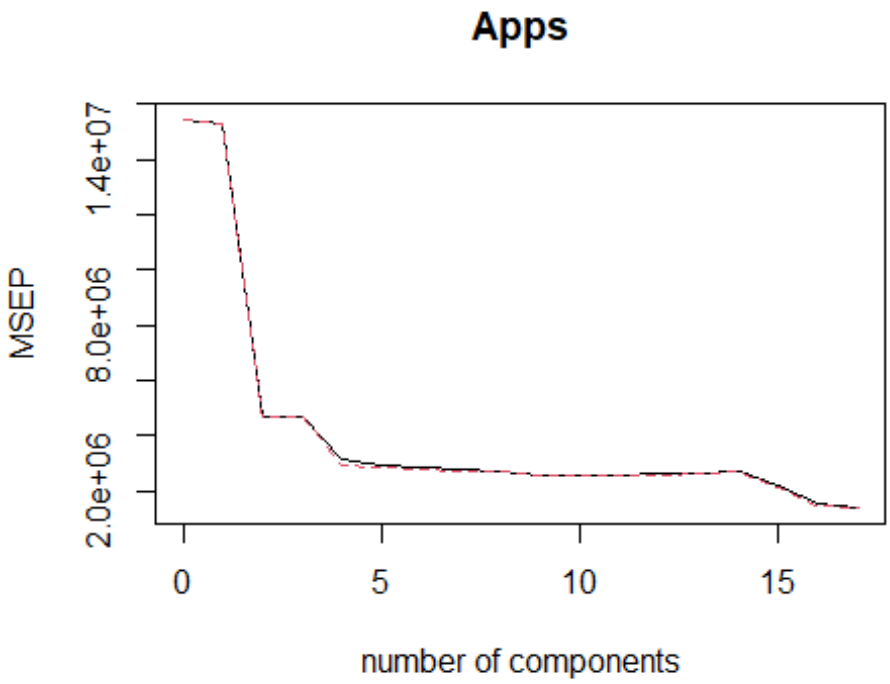
```
data("College")
set.seed(9)
train_index <- sample(1:nrow(College), size = 0.7 * nrow(College))
train_data <- College[train_index, ]
test_data <- College[-train_index, ]
```

b. Fit a linear model using least squares on the training set, and report the test error obtained.

```
lm_model <- lm(Apps ~ ., data = train_data)
lm_predictions <- predict(lm_model, newdata = test_data)
lm_mse <- mean((test_data$Apps - lm_predictions)^2)
cat("Test Error (MSE) - Linear Regression:", lm_mse, "\n")

## Test Error (MSE) - Linear Regression: 1156240
```

e. Fit a PCR model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
library(pls)
set.seed(9)
pcr_model <- pcr(Apps ~ ., data = train_data, scale = TRUE, validation =
"CV")
summary(pcr_model)

## Data:    X dimension: 543 17
##  Y dimension: 543 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3931     3914     2166     2162     1776     1709     1678
## adjCV         3931     3914     2162     2162     1706     1697     1672
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1657     1644     1602      1598      1610      1614      1629
## adjCV     1644     1634     1598      1594      1605      1609      1624
##        14 comps  15 comps  16 comps  17 comps
## CV         1640      1485      1226      1189
## adjCV      1639      1465      1215      1179
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8
comps
## X      32.129    57.43    64.27    69.93    75.26    80.25    83.81
87.28
## Apps    1.879    71.08    71.14    82.47    82.49    82.93    83.77
84.15
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.36     92.93     95.05     96.84     97.87     98.70     99.38
## Apps    84.74     85.13     85.13     85.19     85.20     85.21     90.65
##       16 comps  17 comps
## X        99.85    100.00
## Apps     92.89     93.19

M_optimal <- which.min(pcr_model$validation$PRESS)
cat("\nJumlah Komponen Optimal (M):", M_optimal, "\n")
## Jumlah Komponen Optimal (M): 17

pcr_predictions <- predict(pcr_model, newdata = test_data, ncomp = M_optimal)
pcr_mse <- mean((test_data$Apps - pcr_predictions)^2)
cat("Test Error (MSE) - PCR:", pcr_mse, "\n")

## Test Error (MSE) - PCR: 1156240

validationplot(pcr_model, val.type = "MSEP")
```



MSEP adalah error prediksi dalam Cross-Validation Karena MSEP stabil setelah M = 17, maka M optimal = 17 (menghindari overfitting)

(Subbab 12.6) Exercises Unsupervised Learning

Nomor 8

In Section 12.2.3, a formula for calculating PVE was given in Equation 12.10. We also saw that the PVE can be obtained using the `sdev` output of the `prcomp()` function.

On the `USArrests` data, calculate PVE in two ways:

a. Using the `sdev` output of the `prcomp()` function, as was done in Section 12.2.3.

```
data("USArrests")
hasil_pca <- prcomp(USArrests, center = TRUE, scale. = TRUE)
pve_a <- (hasil_pca$sdev^2) / sum(hasil_pca$sdev^2)
cat("PVE menggunakan sdev dari prcomp():\n")

## PVE menggunakan sdev dari prcomp():

print(pve_a)

## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

b. By applying Equation 12.10 directly. That is, use the `prcomp()` function to compute the principal component loadings. Then, use those loadings in Equation 12.10 to obtain the PVE.

These two approaches should give the same results.

```
lambda <- apply(hasil_pca$x, 2, var)
pve_b <- lambda / sum(lambda)
cat("PVE menggunakan Equation 12.10:\n")

## PVE menggunakan Equation 12.10:

print(pve_b)

##         PC1         PC2         PC3         PC4
## 0.62006039 0.24744129 0.08914080 0.04335752

identical(round(unname(pve_a), 8), round(unname(pve_b), 8))

## [1] TRUE
```

Menggunakan `sdev` dari `prcomp()`lebih sederhana daripada menghitung secara manual dengan Equation 12.10. Kedua metode memberikan hasil yang identik, yang membuktikan keakuratan teori PCA dalam menangkap variansi data. PCA berguna untuk reduksi dimensi, karena kita dapat memilih beberapa PC pertama yang menjelaskan sebagian besar variansi, tanpa kehilangan terlalu banyak informasi.

Nomor 10

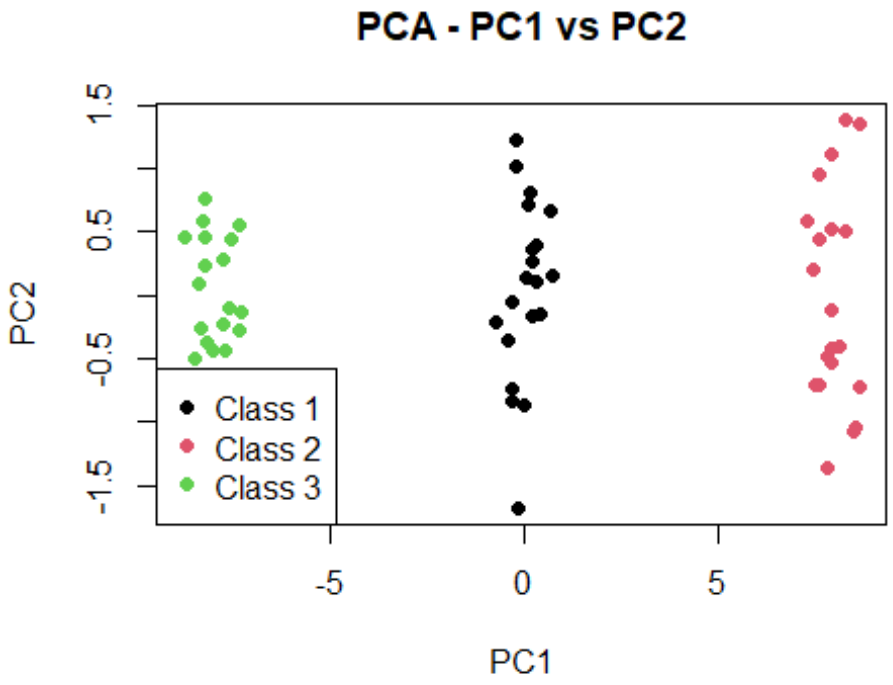
In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data.

a. Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables.

```
set.seed(10)
n <- 20
p <- 50
class1 <- matrix(rnorm(n * p, mean = 0), nrow = n, ncol = p)
class2 <- matrix(rnorm(n * p, mean = 3), nrow = n, ncol = p)
class3 <- matrix(rnorm(n * p, mean = -3), nrow = n, ncol = p)
X <- rbind(class1, class2, class3)
true_labels <- rep(1:3, each = n)
```

b. Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.

```r
pca_result <- prcomp(X, center = TRUE, scale. = TRUE)
plot(pca_result$x[, 1:2], col = true_labels, pch = 19,
     xlab = "PC1", ylab = "PC2", main = "PCA - PC1 vs PC2")
legend("bottomleft", legend = c("Class 1", "Class 2", "Class 3"),
       col = 1:3, pch = 19)
```



Data simulasi dibuat dengan 3 kelas (masing-masing 20 observasi) dan 50 variabel. PCA digunakan untuk melihat apakah kelas dapat dipisahkan dan ternyata berdasarkan plot kelas dapat dipisahkan.