Final Exam

Dawid Dąbkowski

2 February 2018

PRE-PROCESSING

We start by setting the random seed and loading useful libraries.

```
set.seed(02022018)
library(RCurl)
library(gdata)
library(dplyr)
library(magrittr)
library(ggplot2)
library(MASS)
library(caret)
library(lars)
library(pls)
library(glmnet)
library(ade4)
library(CCA)
library(yacca)
library(cluster)
```

Now we load our data frames for the exercises.

```
urls <- c("https://www.mimuw.edu.pl/~noble/courses/SDA/data/ushighways.txt",
          "https://www.mimuw.edu.pl/~noble/courses/SDA/data/PET.txt",
          "https://www.mimuw.edu.pl/~noble/courses/SDA/data/ozone.csv",
          "https://www.mimuw.edu.pl/~noble/courses/SDA/data/pendigits.txt",
          "https://www.mimuw.edu.pl/~noble/courses/SDA/data/carmarks.txt",
          "https://www.mimuw.edu.pl/~noble/courses/SDA/data/primate.scapulae.txt")
ushighways <- read.table(urls[1], header=T)
bodyfat <- read.xls("bodyfat2.xlsx")
yarn <- read.table(urls[2], header=T)   #data(yarn, package="pls")
ozone <- read.csv(urls[3])
pendigits <- read.table(urls[4])
carmarks <- read.table(urls[5], sep=";", header=T)
scapular <- read.table(urls[6], sep=" ", header=T)
```

EXERCISE 1

Let us take a look at data.

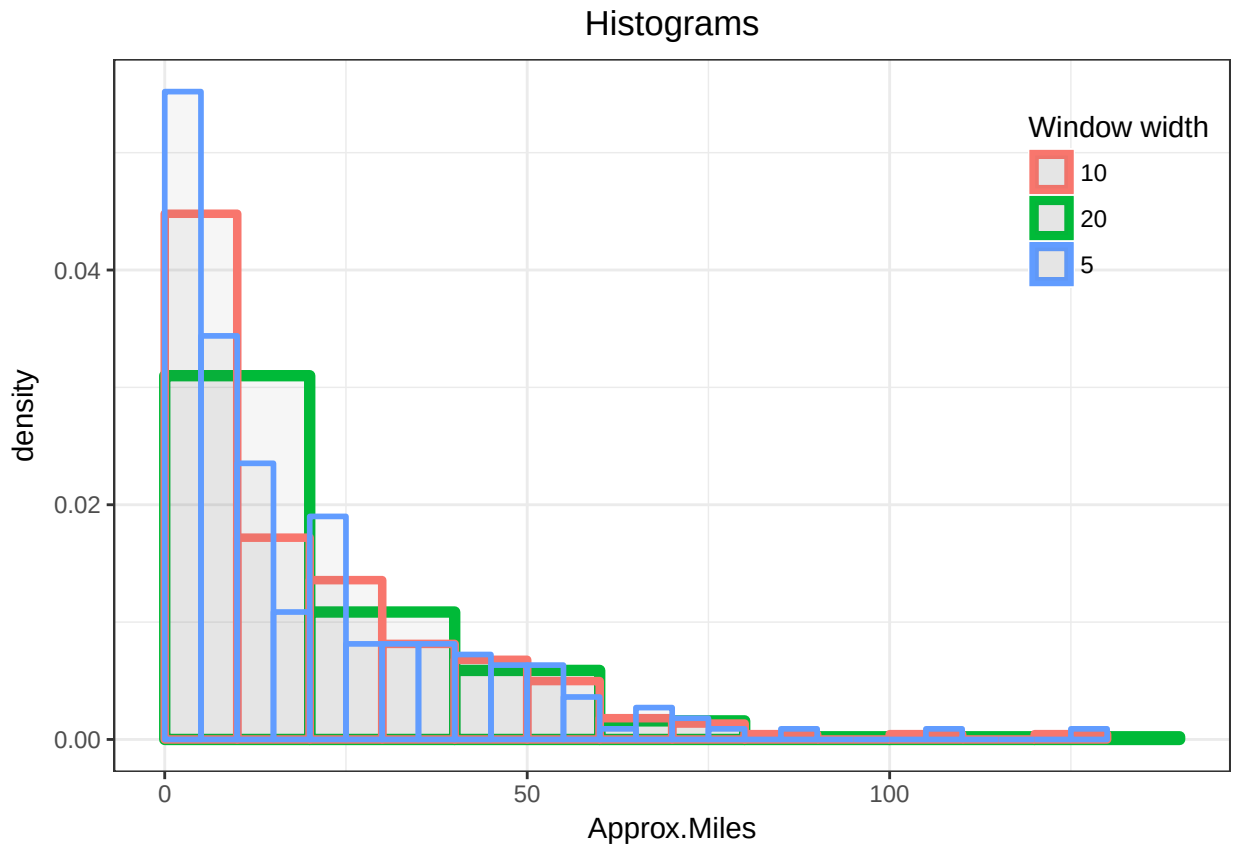
```
head(ushighways)
```

```
##   Interstate.. State Approx.Miles    Location
## 1          165    AL            4      MOBILE
## 2          359    AL            4  TUSCALOOSA
## 3          459    AL           32  BIRMINGHAM
## 4          759    AL            4     GADSDEN
## 5          565    AL           21  HUNTSVILLE
```

```
## 6          430    AR           13 LITTLE ROCK
```

We first draw some histograms with different window widths.

```
ggplot(ushighways, aes(x=Approx.Miles)) +
    geom_histogram(size=2, alpha=0.05, binwidth=20, boundary=0,
                   aes(color="20", y=..density..)) +
    geom_histogram(size=1.5, alpha=0.05, binwidth=10, boundary=0,
                   aes(color="10", y=..density..)) +
    geom_histogram(size=1, alpha=0.05, binwidth=5, boundary=0,
                   aes(color="5", y=..density..)) +
    theme_bw() + theme(plot.title = element_text(hjust = 0.5)) + ggtitle("Histograms") +
    guides(color=guide_legend(title="Window width")) +
    theme(legend.position=c(0.9,0.8), legend.background=element_rect(fill=alpha('blue',0)))
```



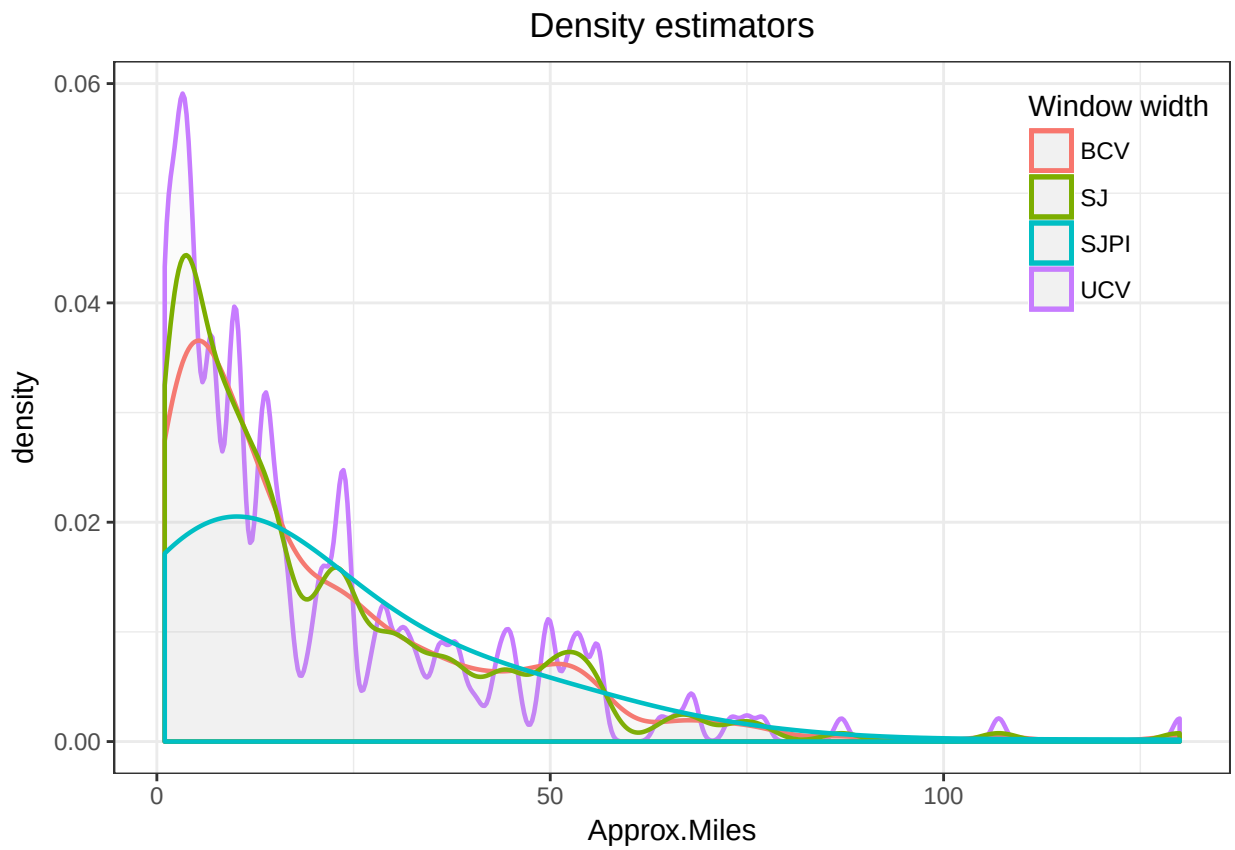
The second picture with window width of 10 miles looks quite good. It is well-splitted and shows the monotonicity of the distribution.

Now we will use UCV, BCV and SJPI estimators for window width to plot densities with gaussian kernel.

```
sjpi <- width.SJ(ushighways$Approx.Miles, method="dpi")
ggplot(ushighways, aes(x=Approx.Miles)) +
    geom_density(fill="grey", size=0.8, alpha=0.05, bw="ucv", aes(color="UCV")) +
    geom_density(fill="grey", size=0.8, alpha=0.05, bw="bcv", aes(color="BCV")) +
    geom_density(fill="grey", size=0.8, alpha=0.05, bw="sj", aes(color="SJ")) +
    geom_density(fill="grey", size=0.8, alpha=0.05, bw=sjpi, aes(color="SJPI")) +
    theme_bw() + theme(plot.title = element_text(hjust = 0.5)) +
    ggtitle("Density estimators") + guides(color=guide_legend(title="Window width")) +
```

```
    theme(legend.position=c(0.9,0.8), legend.background=element_rect(fill=alpha('blue',0)))
```

```
## Warning in bw.ucv(x): minimum occurred at one end of the range
```



The BCV estimator (red one) looks quite well-fitted and monotonic. The other kernel choices (triangular, cosine etc.) gave very similar results.

EXERCISE 2

Let us take a look at data.

```
head(bodyfat[,1:6])
```

```
##   density bodyfat age weight height neck
## 1  1.0708    12.3  23 154.25  67.75 36.2
## 2  1.0853     6.1  22 173.25  72.25 38.5
## 3  1.0414    25.3  22 154.00  66.25 34.0
## 4  1.0751    10.4  26 184.75  72.25 37.4
## 5  1.0340    28.7  24 184.25  71.25 34.4
## 6  1.0502    20.9  24 210.25  74.75 39.0
```

We now calculate correlation matrix for 13 explanatory variables.

```
round(cor(bodyfat[,-(1:2)]), 2)
```

```
##           age weight height neck chest abdomen   hip thigh knee ankle
## age      1.00  -0.01  -0.25 0.11  0.18    0.23 -0.05 -0.20 0.02 -0.11
```

```
## weight  -0.01   1.00   0.49 0.83  0.89    0.89  0.94  0.87 0.85  0.61
## height  -0.25   0.49   1.00 0.32  0.23    0.19  0.37  0.34 0.50  0.39
## neck     0.11   0.83   0.32 1.00  0.78    0.75  0.73  0.70 0.67  0.48
## chest    0.18   0.89   0.23 0.78  1.00    0.92  0.83  0.73 0.72  0.48
## abdomen  0.23   0.89   0.19 0.75  0.92    1.00  0.87  0.77 0.74  0.45
## hip     -0.05   0.94   0.37 0.73  0.83    0.87  1.00  0.90 0.82  0.56
## thigh   -0.20   0.87   0.34 0.70  0.73    0.77  0.90  1.00 0.80  0.54
## knee     0.02   0.85   0.50 0.67  0.72    0.74  0.82  0.80 1.00  0.61
## ankle   -0.11   0.61   0.39 0.48  0.48    0.45  0.56  0.54 0.61  1.00
## biceps  -0.04   0.80   0.32 0.73  0.73    0.68  0.74  0.76 0.68  0.48
## forearm -0.09   0.63   0.32 0.62  0.58    0.50  0.55  0.57 0.56  0.42
## wrist    0.21   0.73   0.40 0.74  0.66    0.62  0.63  0.56 0.66  0.57
##         biceps forearm wrist
## age      -0.04   -0.09  0.21
## weight    0.80    0.63  0.73
## height    0.32    0.32  0.40
## neck      0.73    0.62  0.74
## chest     0.73    0.58  0.66
## abdomen   0.68    0.50  0.62
## hip       0.74    0.55  0.63
## thigh     0.76    0.57  0.56
## knee      0.68    0.56  0.66
## ankle     0.48    0.42  0.57
## biceps    1.00    0.68  0.63
## forearm   0.68    1.00  0.59
## wrist     0.63    0.59  1.00
```

Some of the variables are highly correlated (around 80-90%), which may lead to ill-conditioning for regression. Let us fit a regression model.

```
bodyfat_fit <- lm(bodyfat~., bodyfat[,-1])
summary(bodyfat_fit)
```

```
##
## Call:
## lm(formula = bodyfat ~ ., data = bodyfat[, -1])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1966  -2.8824  -0.1111   3.1901   9.9979
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -21.35323   22.18616  -0.962  0.33680
## age           0.06457    0.03219   2.006  0.04601 *
## weight       -0.09638    0.06185  -1.558  0.12047
## height       -0.04394    0.17870  -0.246  0.80599
## neck         -0.47547    0.23557  -2.018  0.04467 *
## chest        -0.01718    0.10322  -0.166  0.86792
## abdomen       0.95500    0.09016  10.592  < 2e-16 ***
## hip          -0.18859    0.14479  -1.302  0.19401
## thigh         0.24835    0.14617   1.699  0.09061 .
## knee          0.01395    0.24775   0.056  0.95516
## ankle         0.17788    0.22262   0.799  0.42505
## biceps        0.18230    0.17250   1.057  0.29166
```

```
## forearm       0.45574    0.19930   2.287  0.02309 *
## wrist        -1.65450    0.53316  -3.103  0.00215 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.309 on 238 degrees of freedom
## Multiple R-squared:  0.7486, Adjusted R-squared:  0.7348
## F-statistic:  54.5 on 13 and 238 DF,  p-value: < 2.2e-16
```

The signifficant variables are: age, neck, abdomen, forearm and wrist. Let us now fit regression by stepwise elimination.

```
bodyfat_fit_forward <- step(lm(bodyfat~1, bodyfat[,-1]), direction="forward",
                            scope=formula(bodyfat_fit), trace=0)
summary(bodyfat_fit_forward)$call
```

```
## lm(formula = bodyfat ~ abdomen + weight + wrist + forearm + neck +
##     age + thigh + hip, data = bodyfat[, -1])
```

```
bodyfat_fit_backward <- step(lm(bodyfat~., bodyfat[,-1]), direction="backward", trace=0)
summary(bodyfat_fit_backward)$call
```

```
## lm(formula = bodyfat ~ age + weight + neck + abdomen + hip +
##     thigh + forearm + wrist, data = bodyfat[, -1])
```

Both on the forward and backward stepwise elimination yield the same model based on 8 variables. Now we wan't to find the best model using leave-one-out cross-validation. It is computationally hard to check all possible models of 13 predictors (approx time 4h). We will then implement and use backward stepwise elimination heuristics instead.

```
stepwise_loocv <- function(variables, trace=F){
    bodyfat_ctrl <- trainControl(method="LOOCV")
    len <- length(variables)
    errors <- data.frame(var=1:len, err=rep(NA,len))
    init_form <- as.formula(paste("bodyfat~", paste(variables, collapse="+")))
    init_fit <- train(init_form, bodyfat[,-1], method="lm", trControl=bodyfat_ctrl)
    init_err <- init_fit$results$RMSE
    for (i in 1:len){
        form <- as.formula(paste("bodyfat~", paste(variables[-i], collapse="+")))
        fit <- train(form, bodyfat[,-1], method="lm", trControl=bodyfat_ctrl)
        errors$err[i] <- fit$results$RMSE
    }
    worst_var <- which.min(errors$err)
    if (trace) print(paste0("bodyfat~", paste(substr(variables, 1, 4), collapse="+"),
                            ": ", round(init_err,2)))
    if (errors$err[worst_var]<init_err) stepwise_loocv(variables[-worst_var], trace=T)
    else return(init_fit$finalModel)
}

bodyfat_fit_loocv <- stepwise_loocv(colnames(bodyfat[,-(1:2)]), trace=T)
```

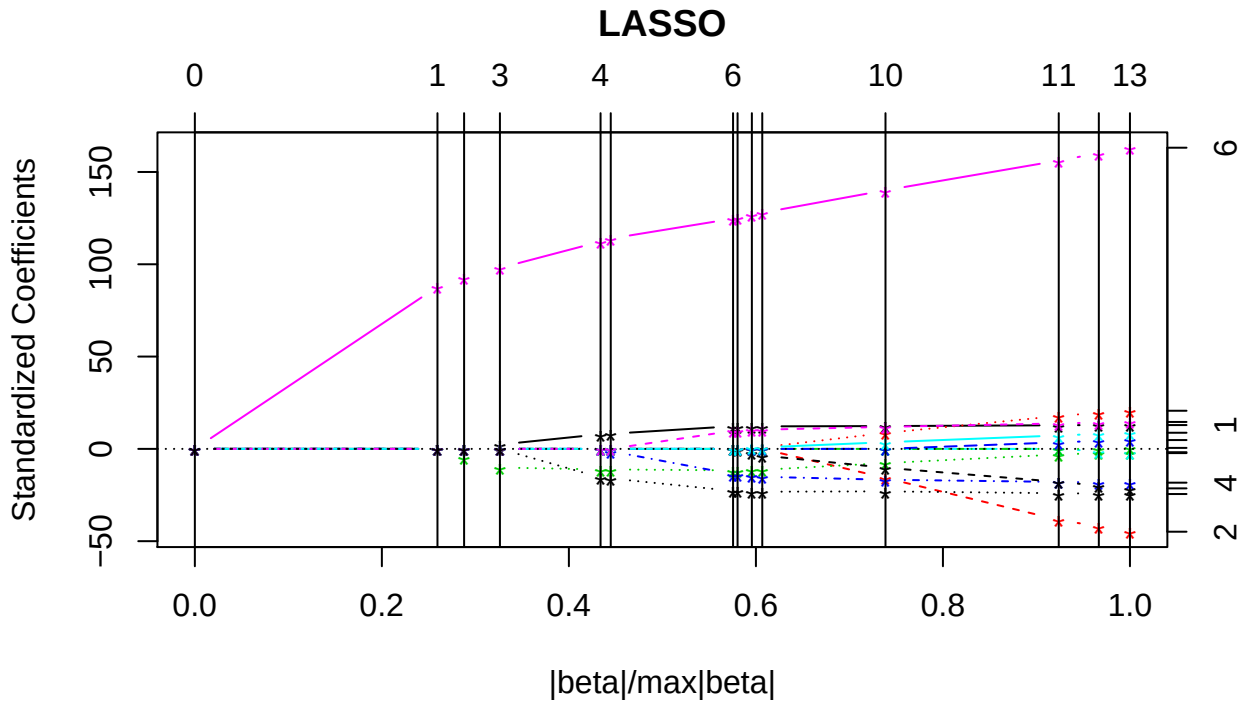
```
## [1] "bodyfat~age+weig+heig+neck+ches+abdo+hip+thig+knee+ankl+bice+fore+wris: 4.5"
## [1] "bodyfat~age+weig+neck+ches+abdo+hip+thig+knee+ankl+bice+fore+wris: 4.45"
## [1] "bodyfat~age+weig+neck+ches+abdo+hip+thig+knee+bice+fore+wris: 4.43"
## [1] "bodyfat~age+weig+neck+abdo+hip+thig+knee+bice+fore+wris: 4.4"
## [1] "bodyfat~age+weig+neck+abdo+hip+thig+bice+fore+wris: 4.38"
## [1] "bodyfat~age+weig+neck+abdo+hip+thig+fore+wris: 4.38"
```

```
formula(bodyfat_fit_loocv)
```

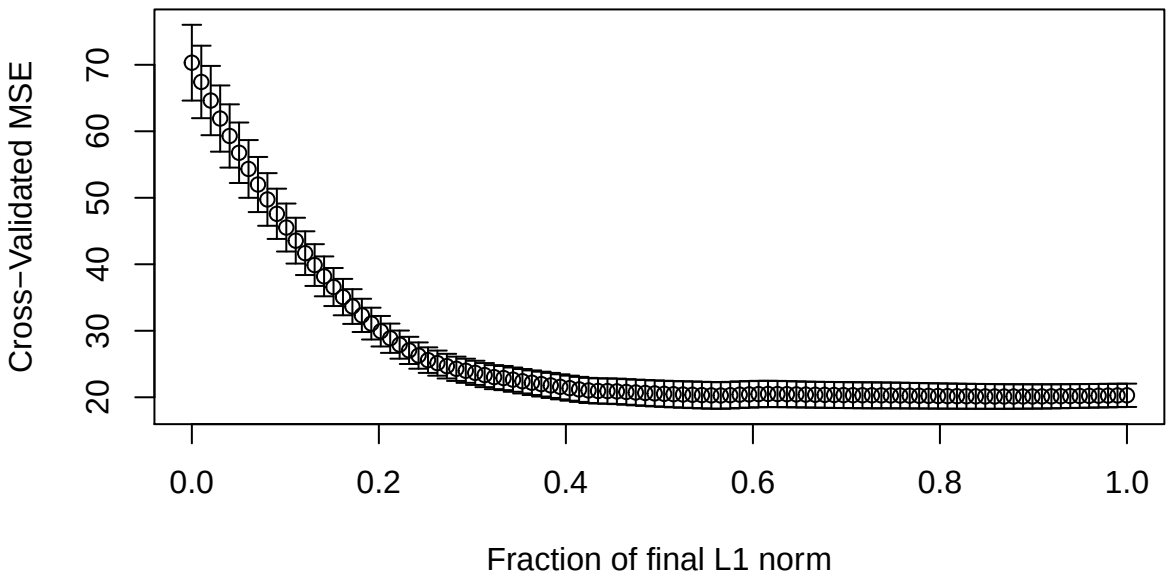
```
## .outcome ~ age + weight + neck + abdomen + hip + thigh + forearm +
##     wrist
## <environment: 0x000000003ac80a00>
```

This method yields the same model as the ones from the elimination based on AIC. Now we fit a LASSO model and look for a best fit by a leave-one-out cross-validation.

```
bodyfat_fit_lasso <- lars(as.matrix(bodyfat[,-(1:2)]), bodyfat[,2], type="lasso")
plot(bodyfat_fit_lasso)
```

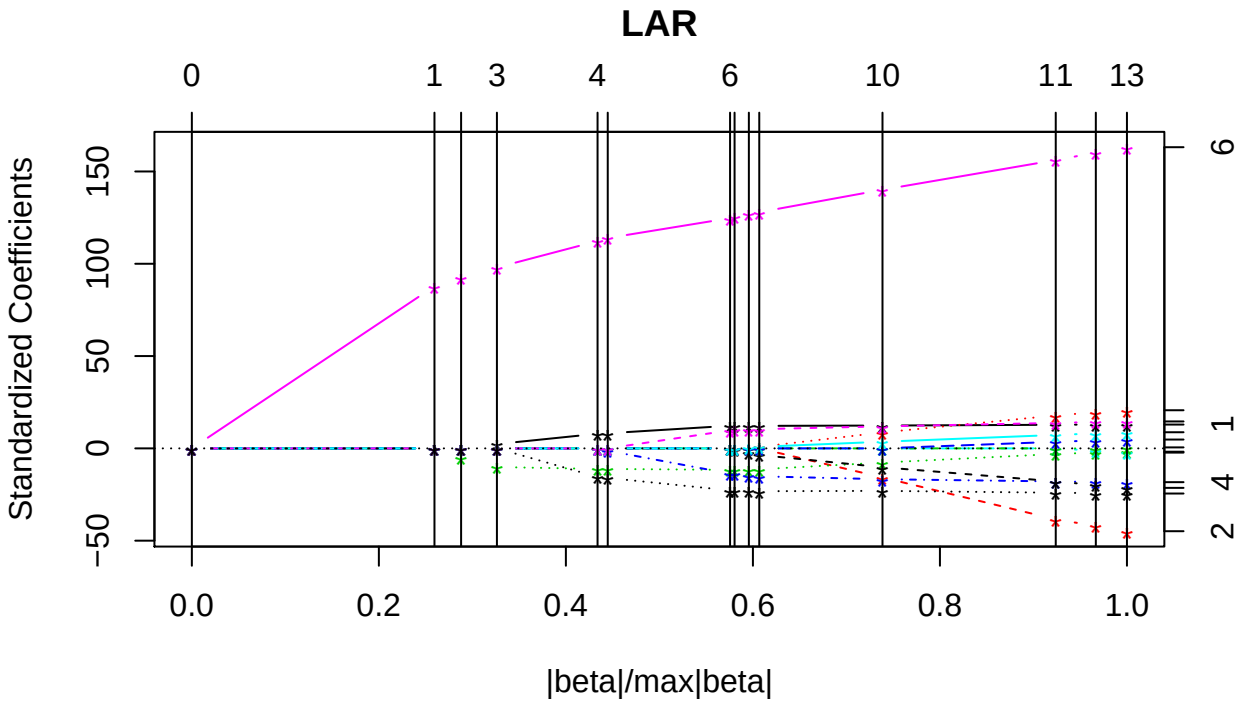



```
bodyfat_lasso <- cv.lars(as.matrix(bodyfat[,-(1:2)]), bodyfat[,2], K=252, type="lasso")
```

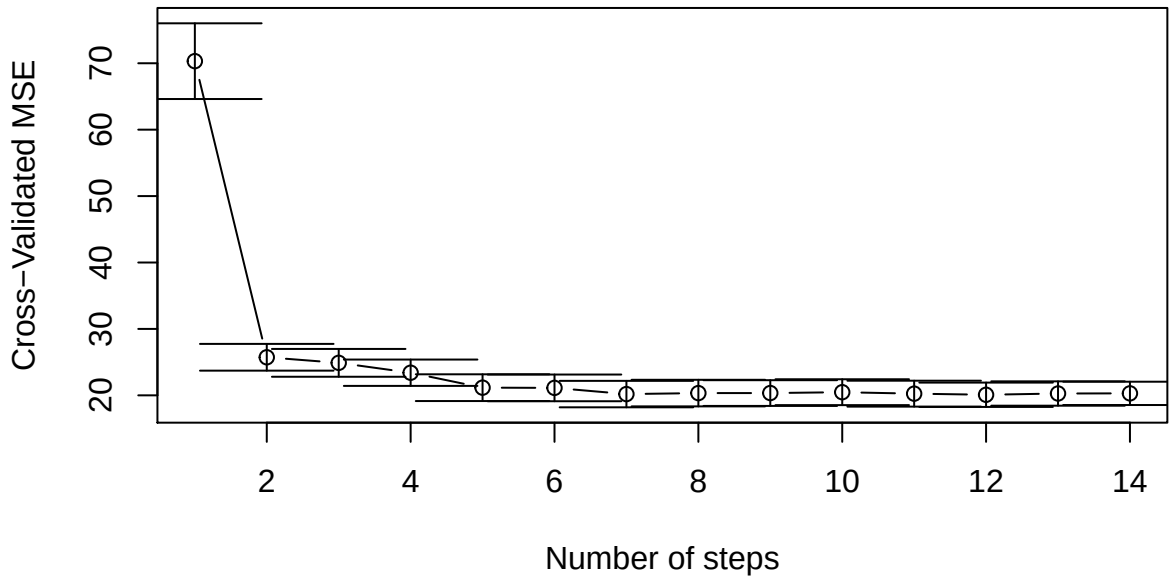


From this we estimate that the lowest MSE of 88 is obtained for LASSO model with parameter 0.8787879. Now we do the same analysis for LARS model.

```
bodyfat_fit_lars <- lars(as.matrix(bodyfat[,-(1:2)]), bodyfat[,2], type="lar")
plot(bodyfat_fit_lars)
```



```
bodyfat_lars <- cv.lars(as.matrix(bodyfat[,-(1:2)]), bodyfat[,2], K=252, type="lar")
```



From this we estimate that the lowest MSE of 88 is obtained for LARS model with parameter 0.8787879. This is a much better result comparing to LASSO model.

EXERCISE 3

Let us take a look at data. We will separate the test set to use it for performance test later.

```
head(yarn[,1:6])
```

```
##      X.1    X.2    X.3    X.4    X.5    X.6
## 1 3.0663 3.0861 3.1079 3.0972 2.9979 2.8273
## 2 3.0675 3.0857 3.0958 3.0692 2.9818 2.8408
## 3 3.0750 3.0966 3.0916 3.0288 2.8849 2.6885
## 4 3.0828 3.0973 3.1010 3.0735 2.9913 2.8709
## 5 3.1029 3.1034 3.0848 3.0228 2.8927 2.7159
## 6 3.0815 3.0849 3.0487 2.9305 2.7323 2.5089
```

```
yarn_train <- (yarn %>% filter(train==1))[,-270]
yarn_test <- (yarn %>% filter(train==0))[,-270]
```

Now we perform PCR and PSLR. We will estimate the error for different number of components by the leave-one-out cross-validation.

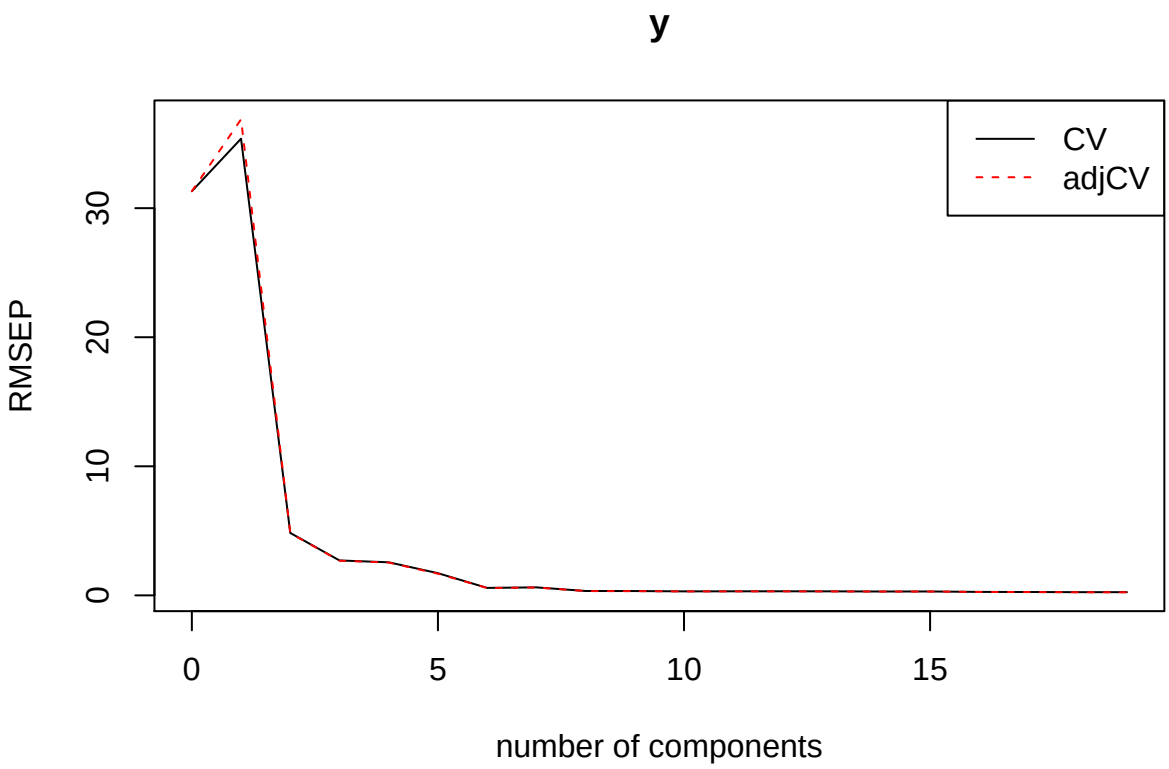
```
yarn_fit_pcr <- pcr(y~., ncomp=20, data=yarn_train, validation="LOO")
```

```
## Warning in pls::mvr(y ~ ., ncomp = 20, data = yarn_train, validation =
## "LOO", : `ncomp' reduced to 19 due to cross-validation
```

```r
summary(yarn_fit_pcr)
```

```
## Data:    X dimension: 21 268
##  Y dimension: 21 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 21 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           31.31    35.38    4.836    2.703    2.562    1.719   0.5782
## adjCV        31.31    36.88    4.816    2.687    2.543    1.680   0.5712
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.6199   0.3394   0.3345    0.3099    0.3143    0.3159    0.3111
## adjCV   0.6140   0.3326   0.3288    0.3042    0.3096    0.3103    0.3043
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.3001    0.3009    0.2677    0.2608    0.2479    0.2474
## adjCV    0.2934    0.2959    0.2616    0.2550    0.2422    0.2421
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X   52.053    98.78    99.51    99.74    99.89    99.98    99.99    99.99
## y    5.173    98.21    99.47    99.77    99.95    99.99    99.99   100.00
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    99.99       100       100       100       100       100       100
## y   100.00       100       100       100       100       100       100
##    16 comps  17 comps  18 comps  19 comps
## X       100       100       100       100
## y       100       100       100       100
```

```r
plot(RMSEP(yarn_fit_pcr), legendpos="topright")
```



We get the lowest cross-validated error for 19 components, so we choose this number. By the look of graph we could take less components for simplicity (not much difference after 6-th component).

```
yarn_fit_plsr <- plsr(y~., ncomp=20, data=yarn_train, validation="LOO")
```

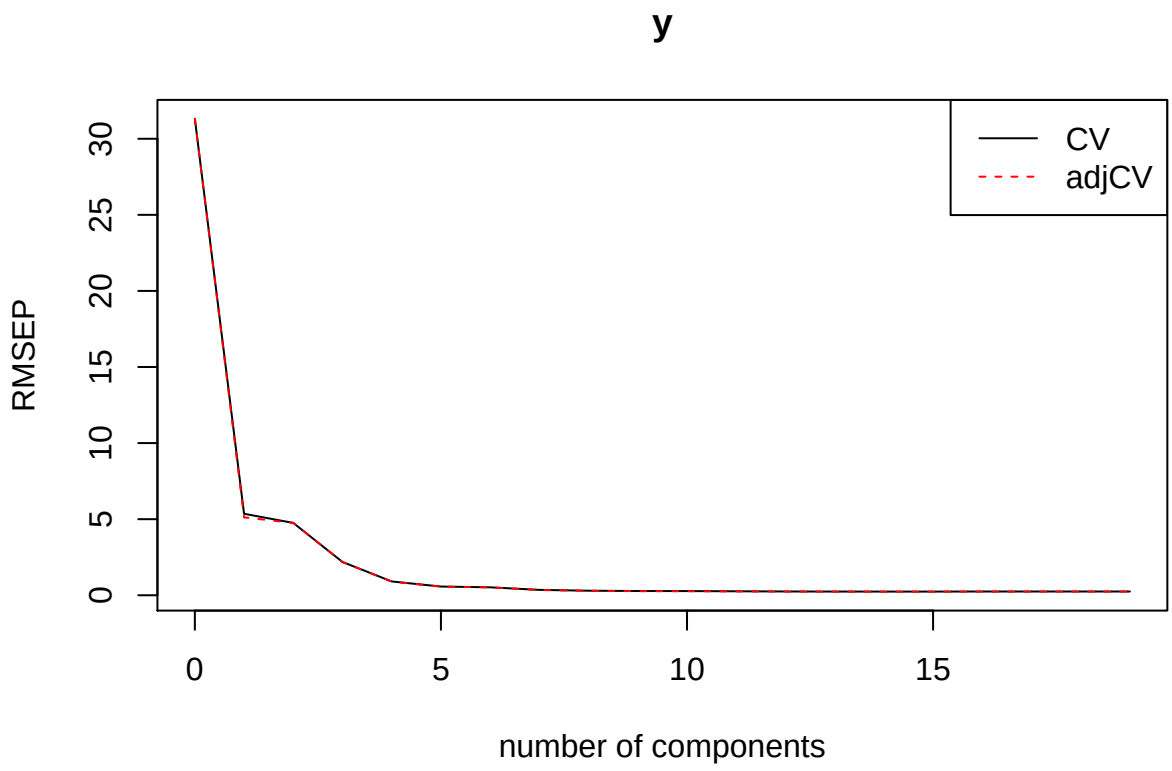
```
## Warning in pls::mvr(y ~ ., ncomp = 20, data = yarn_train, validation =
## "LOO", : `ncomp' reduced to 19 due to cross-validation
```

```
summary(yarn_fit_plsr)
```

```
## Data:    X dimension: 21 268
##  Y dimension: 21 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 21 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           31.31    5.356    4.754    2.187   0.9078   0.5737   0.5228
## adjCV        31.31    5.117    4.755    2.177   0.8933   0.5675   0.5191
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.3544   0.2988   0.2777    0.2745    0.2611    0.2488    0.2431
## adjCV   0.3474   0.2933   0.2718    0.2682    0.2549    0.2430    0.2374
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.2410    0.2425    0.2466    0.2477    0.2472    0.2474
## adjCV    0.2353    0.2367    0.2407    0.2417    0.2412    0.2414
##
## TRAINING: % variance explained
```

```
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    47.07    98.58    99.50    99.72    99.87    99.98    99.98    99.99
## y    98.19    98.29    99.71    99.97    99.99    99.99   100.00   100.00
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    99.99     99.99       100       100       100       100       100
## y   100.00    100.00       100       100       100       100       100
##    16 comps  17 comps  18 comps  19 comps
## X       100       100       100       100
## y       100       100       100       100
```

```
plot(RMSEP(yarn_fit_plsr), legendpos="topright")
```



We get the lowest cross-validated error for 14 components, so we choose this number. By the look of graph we could take less components for simplicity (not much difference after 5-th component).

Now we perform a ridge regression with cross-validation.

```
yarn_fit_ridge <- cv.glmnet(as.matrix(yarn_train[,-269]), yarn_train[,269], nfolds=21)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations
## per fold
```

We choose the model with lambda.1se estimator, which equals to 0.8926168.

Now we compare all models.

```
predict(yarn_fit_pcr, newdata=yarn_test)[,,19] -> a
RMSEP(yarn_fit_pcr, newdata=yarn_test)$val[19]
```

```
## [1] 0.08475492
```

```
RMSEP(yarn_fit_plsr, newdata=yarn_test)$val[14]
```

```
## [1] 0.09504633
```

```
sqrt(mean((predict(yarn_fit_ridge, newx=as.matrix(yarn_test[,-269]),
                   s=yarn_fit_ridge$lambda.1se)-yarn_test[,269])^2))
```

```
## [1] 0.4837982
```

We can see that the PCR and PLSR models perform very well. The ridge model gives slightly worse results for this example. More precise lambda search might be helpful.

EXERCISE 4

Let us take a look at data.

```
head(ozone)
```

```
##   Station   Av8top      Lat       Lon
## 1      60 7.225806 34.13583 -117.9236
## 2      69 5.899194 34.17611 -118.3153
## 3      72 4.052885 33.82361 -118.1875
## 4      74 7.181452 34.19944 -118.5347
## 5      75 6.076613 34.06694 -117.7514
## 6      84 3.157258 33.92917 -118.2097
```

To perfrom Mantel test we need to calculate two matrices - one with measurement differences and one with geographic distances.

```
ozone_m1 <- dist(ozone$Av8top)
ozone_m2 <- dist(cbind(ozone$Lon, ozone$Lat))
set.seed(06022018)
mantel.rtest(ozone_m1, ozone_m2)
```

```
## Warning in is.euclid(m1): Zero distance(s)
```

```
## Warning in is.euclid(distmat): Zero distance(s)
```

```
## Monte-Carlo test
## Call: mantel.rtest(m1 = ozone_m1, m2 = ozone_m2)
##
## Observation: 0.1636308
##
## Based on 99 replicates
## Simulated p-value: 0.03
## Alternative hypothesis: greater
##
##      Std.Obs  Expectation     Variance
##  2.311312808 -0.013827128  0.005894858
```

The simulated p-value is lower than 0.05, which suggest that the alternative hypothesis of related distances is true. We can say that stations, which are closer, have similar ozone measurements.

EXERCISE 5

Let us take a look at data.

```
head(pendigits[,1:6])
```

```
##    V1  V2 V3  V4  V5  V6
## 1  47 100 27  81  57  37
## 2   0  89 27 100  42  75
## 3   0  57 31  68  72  90
## 4   0 100  7  92   5  68
## 5   0  67 49  83 100 100
## 6 100 100 88  99  49  74
```

Some of the columns are reduntant. We will throw them away.

```
pendigits <- pendigits[,1:17]
```

We start by computing the variances.

```
sapply(pendigits[,-17], var)
```

```
##        V1        V2        V3        V4        V5        V6        V7
## 1173.5957  263.0420  693.9528  367.2453 1162.8451  728.8211  934.9457
##        V8        V9       V10       V11       V12       V13       V14
##  894.3750 1165.2344  742.6469 1390.4013  735.4934  498.8763 1099.2847
##       V15       V16
## 1743.9310 1280.0720
```

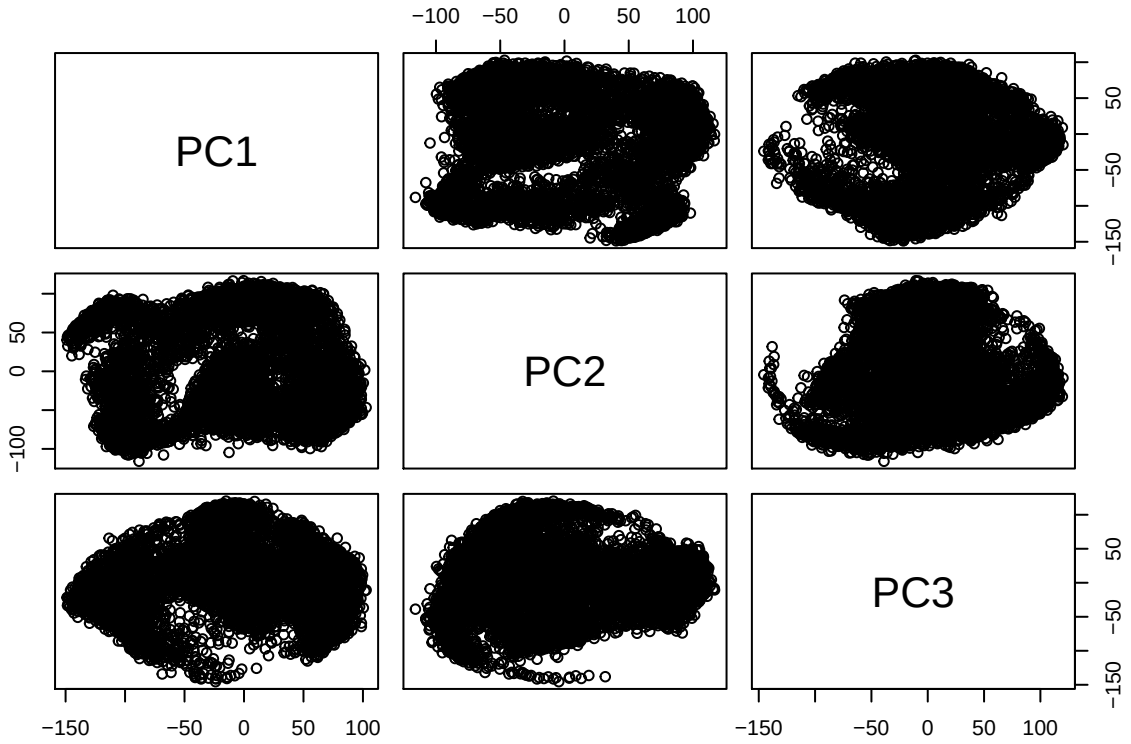
We see that variables differ. Now we perform a PCA on covariance matrix.

```
pendigits_pcacov <- prcomp(pendigits[,-17], scale=F)
summary(pendigits_pcacov)
```

```
## Importance of components:
##                            PC1     PC2     PC3      PC4      PC5      PC6
## Standard deviation     64.9131 60.8446 47.8075 36.62327 29.35851 26.80043
## Proportion of Variance  0.2833  0.2489  0.1537  0.09017  0.05795  0.04829
## Cumulative Proportion   0.2833  0.5322  0.6858  0.77599  0.83393  0.88222
##                            PC7      PC8      PC9     PC10     PC11
## Standard deviation     21.38547 19.93971 16.93488 14.29245 11.36052
## Proportion of Variance  0.03075  0.02673  0.01928  0.01373  0.00868
## Cumulative Proportion   0.91296  0.93969  0.95897  0.97271  0.98138
##                           PC12    PC13    PC14    PC15    PC16
## Standard deviation     10.01589 8.13346 7.66022 5.23502 4.93634
## Proportion of Variance  0.00674 0.00445 0.00394 0.00184 0.00164
## Cumulative Proportion   0.98813 0.99257 0.99652 0.99836 1.00000
```

As we can see, taking just 5 components retain more than 80% of data variance; taking 7 retain more than 90%. Now we visualise three first components on the scatterplots.

```
plot(data.frame(pendigits_pcacov$x[,1:3]))
```

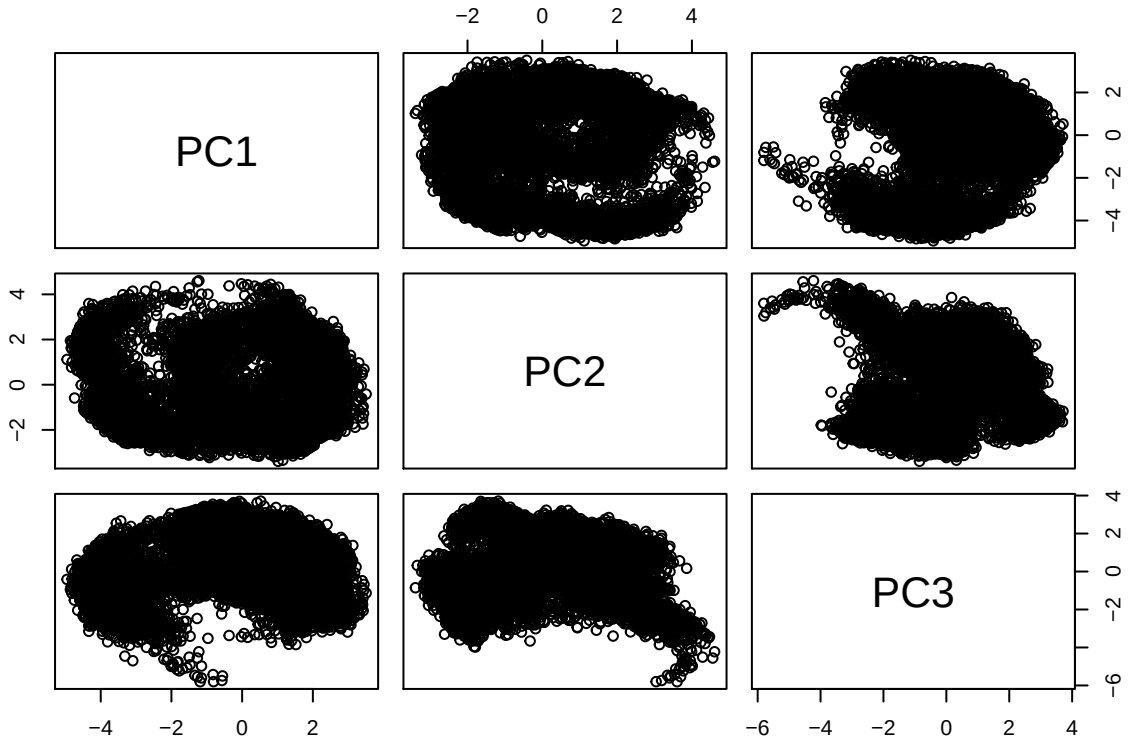


We will now carry a PCA with the correlation matrix.

```
pendigits_pcacor <- prcomp(pendigits[,-17], scale=T)
summary(pendigits_pcacor)
```

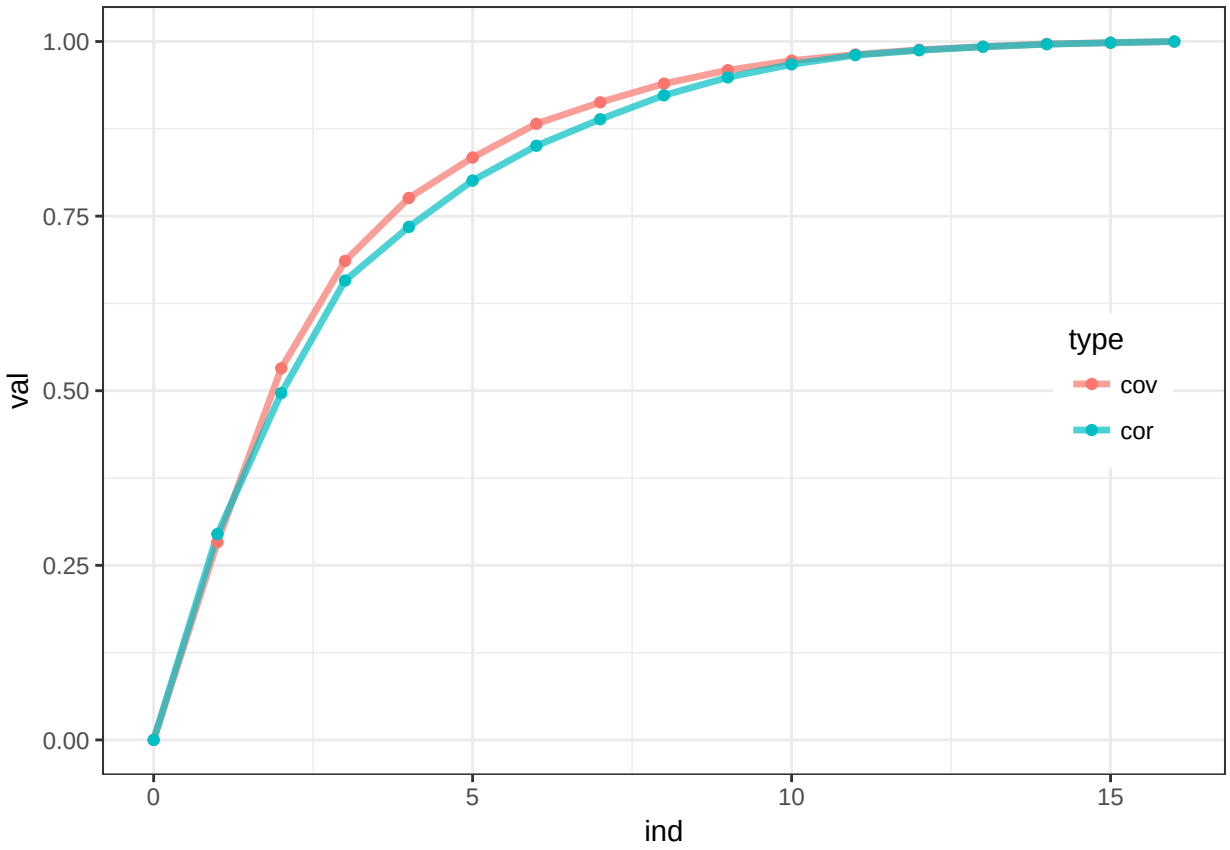
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     2.1718 1.7970 1.6052 1.10894 1.03107 0.89294
## Proportion of Variance 0.2948 0.2018 0.1610 0.07686 0.06644 0.04983
## Cumulative Proportion  0.2948 0.4966 0.6577 0.73452 0.80096 0.85079
##                           PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.77888 0.74044 0.64096 0.54611 0.45882 0.33511
## Proportion of Variance 0.03792 0.03427 0.02568 0.01864 0.01316 0.00702
## Cumulative Proportion  0.88871 0.92297 0.94865 0.96729 0.98045 0.98747
##                           PC13    PC14    PC15    PC16
## Standard deviation     0.28369 0.24084 0.18511 0.16673
## Proportion of Variance 0.00503 0.00363 0.00214 0.00174
## Cumulative Proportion  0.99250 0.99612 0.99826 1.00000
```

```
plot(data.frame(pendigits_pcacor$x[,1:3]))
```



The scaled pca performs slightly worse on this data (we need one more component to explain 80% or 90% of variance). Some of the components are more informative and some are less. Now we make a scree plots for both approaches.

```
cov<-data.frame(ind=1:16, val=summary(pendigits_pcacov)$importance[3,], type=rep("cov",16))
cov<-rbind(list(0,0,"cov"), cov)
cor<-data.frame(ind=1:16, val=summary(pendigits_pcacor)$importance[3,], type=rep("cor",16))
cor<-rbind(list(0,0,"cor"), cor)
ggplot(rbind(cov,cor), aes(x=ind, y=val, color=type)) + geom_line(size=1.2, alpha=0.7) +
     theme_bw() + theme(legend.position=c(0.9,0.5)) + geom_point()
```



We can see that except of first component, the covariance matrix gave better principal components in terms of explained variance. Using elbow method we would take 10 components for further analysis. If we had to reduce dimension even further, we would take 4-5 components.

From the PCA we can see that there is some ill-conditioning of the data matrix - the fractions of variance added from each component are not equal. By taking simply 2 first components, we get more than half information (average of 8 initial variables).

EXERCISE 6

Let us take a look at data.

```
head(carmarks)
```

```
##   CARMARK ECONOMY SERVICE VALUE PRICE DESIGN SPORT SAFETY EASYINESS
## 1    A100     3.9     2.8   2.2   4.2    3.0   3.1    2.4       2.8
## 2    BMW3     4.8     1.6   1.9   5.0    2.0   2.5    1.6       2.8
## 3    CiAX     3.0     3.8   3.8   2.7    4.0   4.4    4.0       2.6
## 4    Ferr     5.3     2.9   2.2   5.9    1.7   1.1    3.3       4.3
## 5    FiUn     2.1     3.9   4.0   2.6    4.5   4.4    4.4       2.2
## 6    FoFi     2.3     3.1   3.4   2.6    3.2   3.3    3.6       2.8
```

We divide the data into X and Y matrices and check the correlations inside these groups. and perform a CCA.

```
carmarks_X <- carmarks[,4:5]
carmarks_Y <- carmarks[,c(2:3,6:9)]
round(cor(carmarks_X),2)
```

```
##       VALUE PRICE
## VALUE  1.00 -0.86
## PRICE -0.86  1.00
```

```
round(cor(carmarks_Y),2)
```

```
##           ECONOMY SERVICE DESIGN SPORT SAFETY EASYINESS
## ECONOMY      1.00   -0.33  -0.62 -0.48  -0.28      0.58
## SERVICE     -0.33    1.00   0.76  0.69   0.94      0.30
## DESIGN      -0.62    0.76   1.00  0.88   0.71     -0.22
## SPORT       -0.48    0.69   0.88  1.00   0.66     -0.25
## SAFETY      -0.28    0.94   0.71  0.66   1.00      0.35
## EASYINESS    0.58    0.30  -0.22 -0.25   0.35      1.00
```

Some of the variables are highly correlated, which may cause problems in CCA. Nevertheless, we will perform CCA and interpret results.

```
carmarks_cca <- cca(carmarks_X, carmarks_Y,
                    xcenter=T, ycenter=T, xscale=T, yscale=T, standardize.score=T)
```

Before we move into conclussions, we will perform a chi-squared test for the signifficance of canonical correlations.

```
pchisq(carmarks_cca$chisq, carmarks_cca$df, ncp=0)
```

```
##      CV 1      CV 2
## 1.0000000 0.9999369
```

There is no basis to reject the null hypothesis, so we can assume that canonical correlations are signifficant. Now let us take a look on the cca object.

```
carmarks_cca
```

```
##
## Canonical Correlation Analysis
##
## Canonical Correlations:
##      CV 1      CV 2
## 0.9791972 0.8851224
##
## X Coefficients:
##             CV 1      CV 2
## VALUE -0.6394099 -1.836135
## PRICE  0.3960227 -1.903524
##
## Y Coefficients:
##                   CV 1        CV 2
## ECONOMY    0.373639712 -0.49066038
## SERVICE   -0.154752889 -0.44050947
## DESIGN    -0.004035987  0.01002372
## SPORT     -0.470514635  0.09819083
## SAFETY    -0.249874034  0.01592503
## EASYINESS -0.212274064 -0.51621224
##
## Structural Correlations (Loadings) - X Vars:
##             CV 1       CV 2
```

```
## VALUE -0.9790363 -0.2036857
## PRICE  0.9443764 -0.3288667
##
## Structural Correlations (Loadings) - Y Vars:
##                   CV 1       CV 2
## ECONOMY    0.599208060 -0.7014229
## SERVICE   -0.903830572 -0.3397643
## DESIGN    -0.899920751  0.1897638
## SPORT     -0.869474987  0.1778602
## SAFETY    -0.887267495 -0.3656884
## EASYINESS -0.008497496 -0.9543117
##
## Aggregate Redundancy Coefficients (Total Variance Explained):
##  X | Y: 0.9457047
##  Y | X: 0.7884929
```

As the smaller group has 2 variables, we have two canonical variables CV1 and CV2. We can see that the highest correlation (0.979) is between PRICE-VALUE and ECONOMY-SERVICE-SPORT-SAFETY-EASYINESS (with some large coefficients). So the higher price and lower value stability, the more economy and less service, sporty, safety and easyiness the car is. The second canonical variable with correlation 0.885 is between -VALUE-PRICE and -ECONOMY-SERVICE-EASINESS (with some large coefficients). So the larger is the value stability and price, the more economy, service and easy to use car is.

If we look at the correlations instead of coefficients, the numbers differ. For example first variable is VALUE-PRICE but correlated with ECONOMY-SERVICE-DESIGN-SPORT-SAFETY (design instead of easyiness) and with different coefficients.

This is the main result and it should be considered that the input correlation matrices were quite ill-conditioned. It gives some intution about the main probable dependecies between X and Y groups of variables.

EXERCISE 7

Let us take a look at data.

```
head(scapular)
```

```
##   genus AD.BD AD.CD EA.CD Dx.CD SH.ACR EAD beta gamma     class classdigit
## 1    54 65.56 166.0 50.55 12.80   70.3 115   14    45 Hylobates          1
## 2    54 50.91  93.9 61.82 13.09   75.0 121   20    54 Hylobates          1
## 3    54 46.15  80.8 64.10 11.80   70.0 120   25    61 Hylobates          1
## 4    54 70.29 220.5 50.00 12.75   61.1 113   12    45 Hylobates          1
## 5    54 63.16 144.0 57.89 12.98   64.9 115   14    46 Hylobates          1
## 6    54 50.72 134.6 56.23 11.88   52.6 136   14    46 Hylobates          1
```

Before we start cluster analysis, we scale the columns, so the euclidean distance will have better interpretation.

```
scapular <- scapular[,-9]
scapular[,2:8] %<>% scale
```

Now we cluster our data using kmeans, pam (divisive) and single, average, complete and ward linkage (hierarchical). For devisive metods we use k=5. For hierarchical we plot dendrograms and cut them to get 5 clusters as well.

```
scapular_kmeans <- kmeans(scapular[,2:8], centers=5)
table(scapular_kmeans$cluster)
```

```
##
```

```
##  1  2  3  4  5
## 16 11 36 19 23
```

```
scapular_pam <- pam(scapular[,2:8], k=5)
table(scapular_pam$cluster)
```

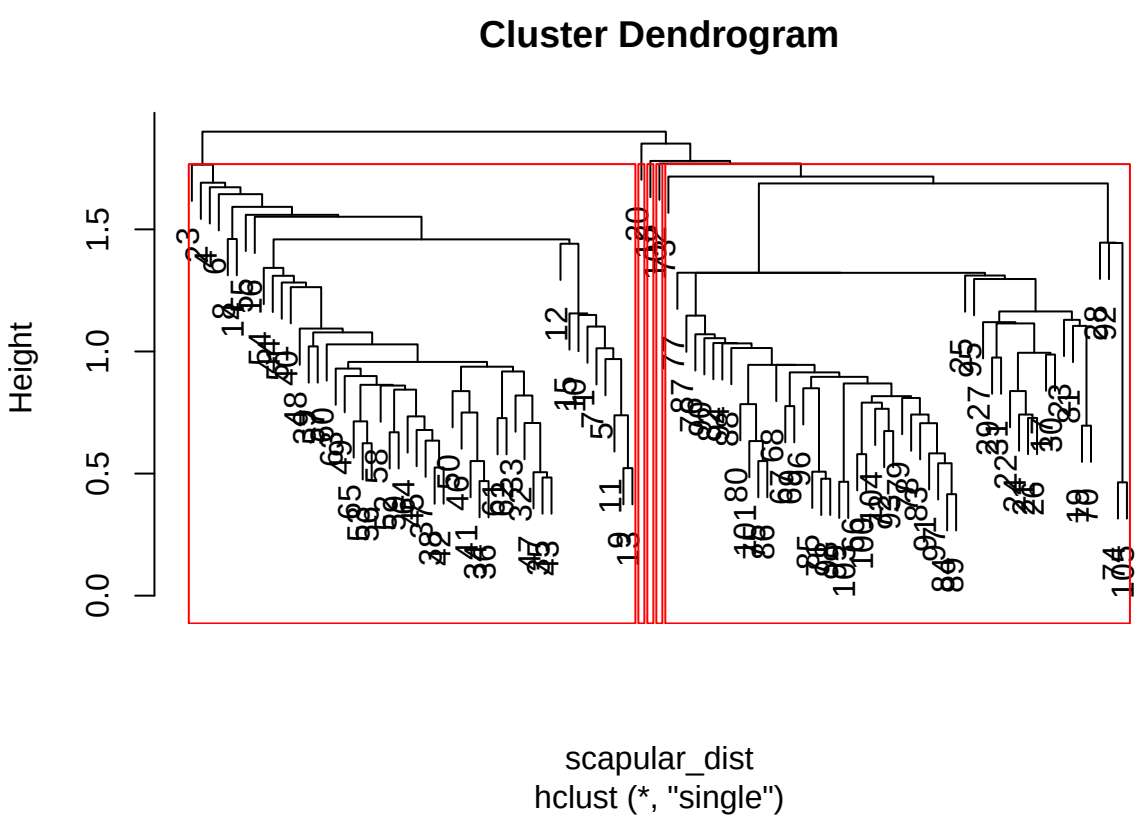
```
##
##  1  2  3  4  5
## 16 16 34 20 19
```

```
scapular_dist <- dist(scapular[,2:8], method="euclidean")

scapular_single <- hclust(scapular_dist, method="single")
scapular_average <- hclust(scapular_dist, method="average")
scapular_complete <- hclust(scapular_dist, method="complete")
scapular_ward <- hclust(scapular_dist, method="ward")
```

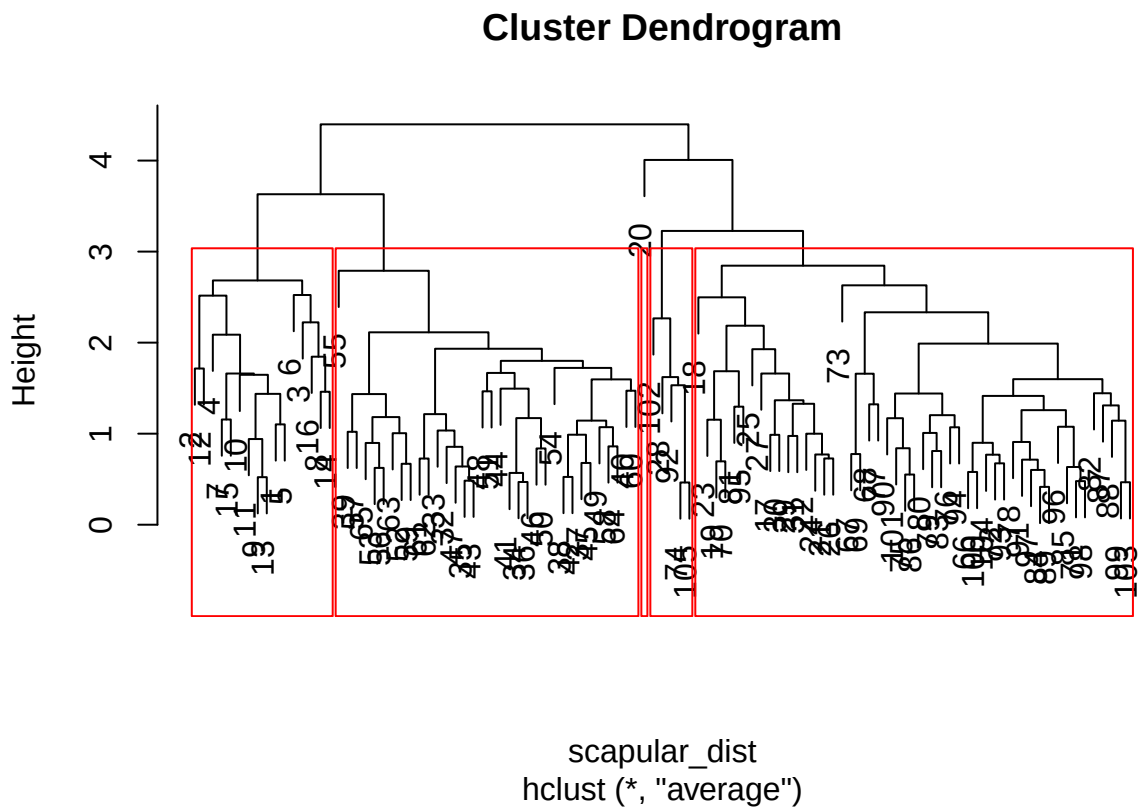
```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
plot(scapular_single)
rect.hclust(scapular_single, k=5, border="red")
```



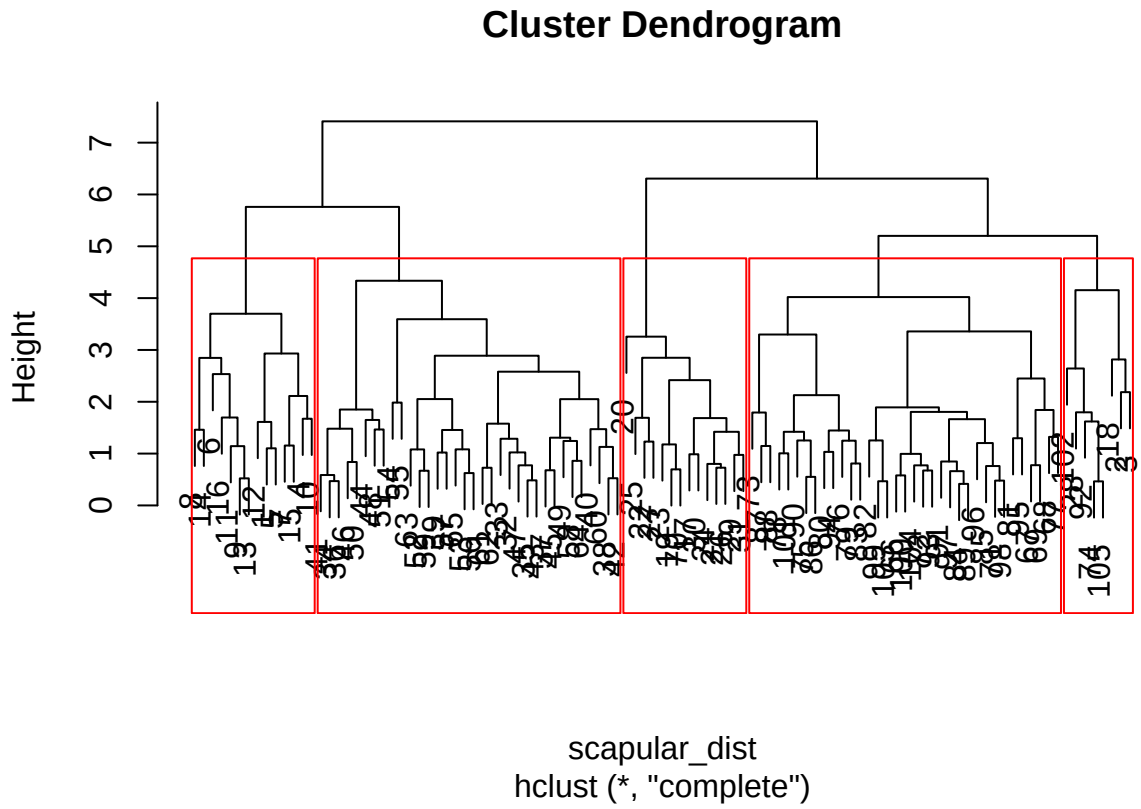

```
scapular_single %<>% cutree(k=5)

plot(scapular_average)
rect.hclust(scapular_average, k=5, border="red")
```



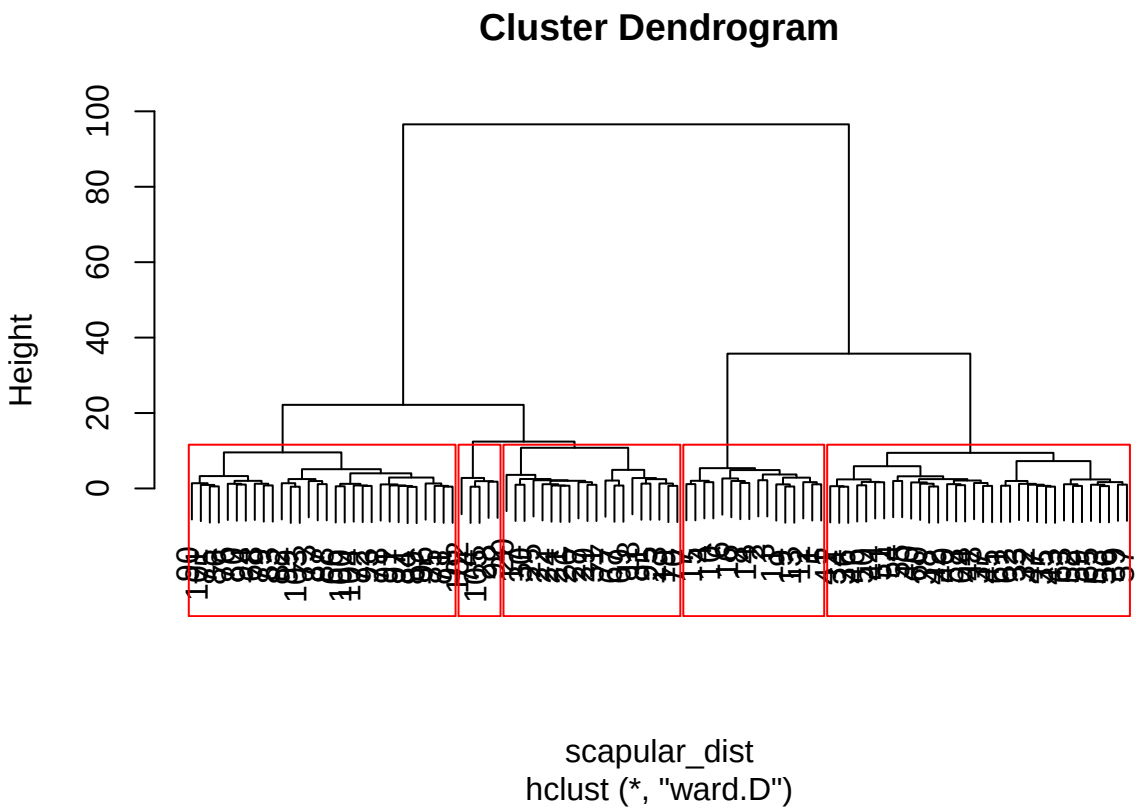

```
scapular_average %<>% cutree(k=5)

plot(scapular_complete)
rect.hclust(scapular_complete, k=5, border="red")
```




```
scapular_complete %<>% cutree(k=5)

plot(scapular_ward)
rect.hclust(scapular_ward, k=5, border="red")
```




```
scapular_ward %<>% cutree(k=5)
```

From the summaries and pictures we can see that devisive algorithms don't give outliers. From the hierarchical methods, single and average linkage produce outliers, while complete and ward linkage does not. Now we compare the results with theoretical groups and write confusion matrices.

```
confusionMatrix(scapular_kmeans$cluster, scapular$classdigit)$table
```

```
##            Reference
## Prediction  1  2  3  4  5
##          1 16  0  0  0  0
##          2  0  0  1 10  0
##          3  0  0  0  0 36
##          4  0 15  0  0  4
##          5  0  0 19  4  0
```

```
confusionMatrix(scapular_kmeans$cluster, scapular$classdigit)$overall[1]
```

```
## Accuracy
## 0.152381
```

```
confusionMatrix(scapular_pam$cluster, scapular$classdigit)$table
```

```
##            Reference
## Prediction  1  2  3  4  5
##          1 16  0  0  0  0
##          2  0 15  0  0  1
##          3  0  0 20 14  0
##          4  0  0  0  0 20
```

```
##          5  0  0  0  0 19
```

```
confusionMatrix(scapular_pam$cluster, scapular$classdigit)$overall[1]
```

```
##  Accuracy
## 0.6666667
```

```
confusionMatrix(scapular_single, scapular$classdigit)$table
```

```
##           Reference
## Prediction  1  2  3  4  5
##          1 16  0 20 14  0
##          2  0 13  0  0 39
##          3  0  1  0  0  0
##          4  0  1  0  0  0
##          5  0  0  0  0  1
```

```
confusionMatrix(scapular_single, scapular$classdigit)$overall[1]
```

```
##  Accuracy
## 0.2857143
```

```
confusionMatrix(scapular_average, scapular$classdigit)$table
```

```
##           Reference
## Prediction  1  2  3  4  5
##          1 16  0  0  0  0
##          2  0 13  0  0 36
##          3  0  1  0  0  0
##          4  0  1  0  0  4
##          5  0  0 20 14  0
```

```
confusionMatrix(scapular_average, scapular$classdigit)$overall[1]
```

```
##  Accuracy
## 0.2761905
```

```
confusionMatrix(scapular_complete, scapular$classdigit)$table
```

```
##           Reference
## Prediction  1  2  3  4  5
##          1 14  0  0  0  0
##          2  2  2  0  0  4
##          3  0 13  0  0  1
##          4  0  0 20 14  0
##          5  0  0  0  0 35
```

```
confusionMatrix(scapular_complete, scapular$classdigit)$overall[1]
```

```
##  Accuracy
## 0.6190476
```

```
confusionMatrix(scapular_ward, scapular$classdigit)$table
```

```
##           Reference
## Prediction  1  2  3  4  5
##          1 16  0  0  0  0
##          2  0 14  0  0  6
##          3  0  1  0  0  4
##          4  0  0 20 14  0
```

```
##          5  0  0  0  0 30
```

```
confusionMatrix(scapular_ward, scapular$classdigit)$overall[1]
```

```
##  Accuracy
## 0.7047619
```

The devisive methods gave quite good results: 0.55 acc for kmeans and 0.67 acc for pam. The hierarchical methods with single and average linkage are worse on this dataset (0.29 and 0.28 acc) but complete and ward linkage works well (0.62 and 0.70 acc).

EXERCISE 8

Let us take a look at data.

```
head(scapular)
```

```
##   genus      AD.BD     AD.CD      EA.CD       Dx.CD     SH.ACR       EAD
## 1    54  0.7559403 2.2773065 -1.7108283 -0.18844057  0.9354432 0.4734640
## 2    54 -0.1469654 0.5632474 -0.6876050  0.26337166  1.4835153 1.0373247
## 3    54 -0.4403327 0.2518165 -0.4805998 -1.74641377  0.9004598 0.9433479
## 4    54  1.0474586 3.5729544 -1.7607637 -0.26633923 -0.1373790 0.2855104
## 5    54  0.6080240 1.7542926 -1.0444167  0.09199461  0.3057432 0.4734640
## 6    54 -0.1586754 1.5308231 -1.1951310 -1.62177591 -1.1285734 2.4469765
##         beta     class classdigit
## 1 -1.6063579 Hylobates          1
## 2 -1.1195828 Hylobates          1
## 3 -0.7139369 Hylobates          1
## 4 -1.7686163 Hylobates          1
## 5 -1.6063579 Hylobates          1
## 6 -1.6063579 Hylobates          1
```