Compulsory Exercise 2 – Solutions

TMA4268 Statistical Learning V2021

Emma Skarstein, Michail Spitieris, Stefanie Muff, Department of Mathematical Sciences, NTNU

Problem 1 (10P)

a) (2P) - Multiple choice

Which of the following statements are true, which false?

(i) In general, regularization will reduce test error, but not training error.

(ii) In best subset selection, we cannot use RSS as a criterion to choose between models with different numbers of predictors, only to select between models with the same number of predictors.

(iii) The tuning parameter λ in ridge and lasso regression should be chosen through cross-validation.

(iv) Principal component regression can be used for variable selection.

Solution TRUE - TRUE - TRUE - FALSE

(i) Yes. (ii) RSS will simply decrease the mode variables we include. (iii) Yes this is a good idea. (iv) No, it reduces the dimensions by constructing linear combinations of the variables in a clever way. But it doesn't select variables.

For the following tasks we will use a data set showing the number of birds killed by cats during a year. The number of killed birds is recorded along with 16 other variables. There were 452 cats participating in the study. The variables are

- `birds`: number of birds killed by the cat throughout a year
- `sex`: the sex of the cat
- `weight`: the cat's weight
- `dryfood`: daily amount of dry food (g)
- `wetfood`: daily amount of wet food (g)
- `age`: the cat's age
- `owner.income`: household yearly income (NOK)
- `daily.playtime`: daily time of owners spent playing with cat (minutes)
- `fellow.cats`: number of additional cats in household
- `owner.age`: average age of humans in household
- `house.area`: house area (sq. meters)
- `children.13`: number of children under the age of 13 in the household
- `urban`: whether the cat's home is in a urban location or not
- `bell`: does the cat wear a bell?
- `dogs`: number of household dogs
- `daily.outdoortime`: amount of time spent outside daily (minutes)
- `daily.catnip`: self-reported daily amount of catnip (g)
- `neutered`: whether or not the cat has been neutered/spayed

```
id <- "1iI6YaqgG0QJW5onZ_GTBsCvpKPExF30G"  # google file ID
catdat <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id),
    header = T)
```

We let `birds` be our response, and begin by splitting into training and testing sets (50% of the data in each) using the following code:

```
set.seed(4268)
train.ind = sample(1:nrow(catdat), 0.5 * nrow(catdat))
catdat.train = catdat[train.ind, ]
catdat.test = catdat[-train.ind, ]
```

b) (2P) - Best subset selection

Use best subset selection to identify a satisfactory model that uses a subset of the variables. Justify any choices you make. Report the selected variables and the test MSE.

Solution

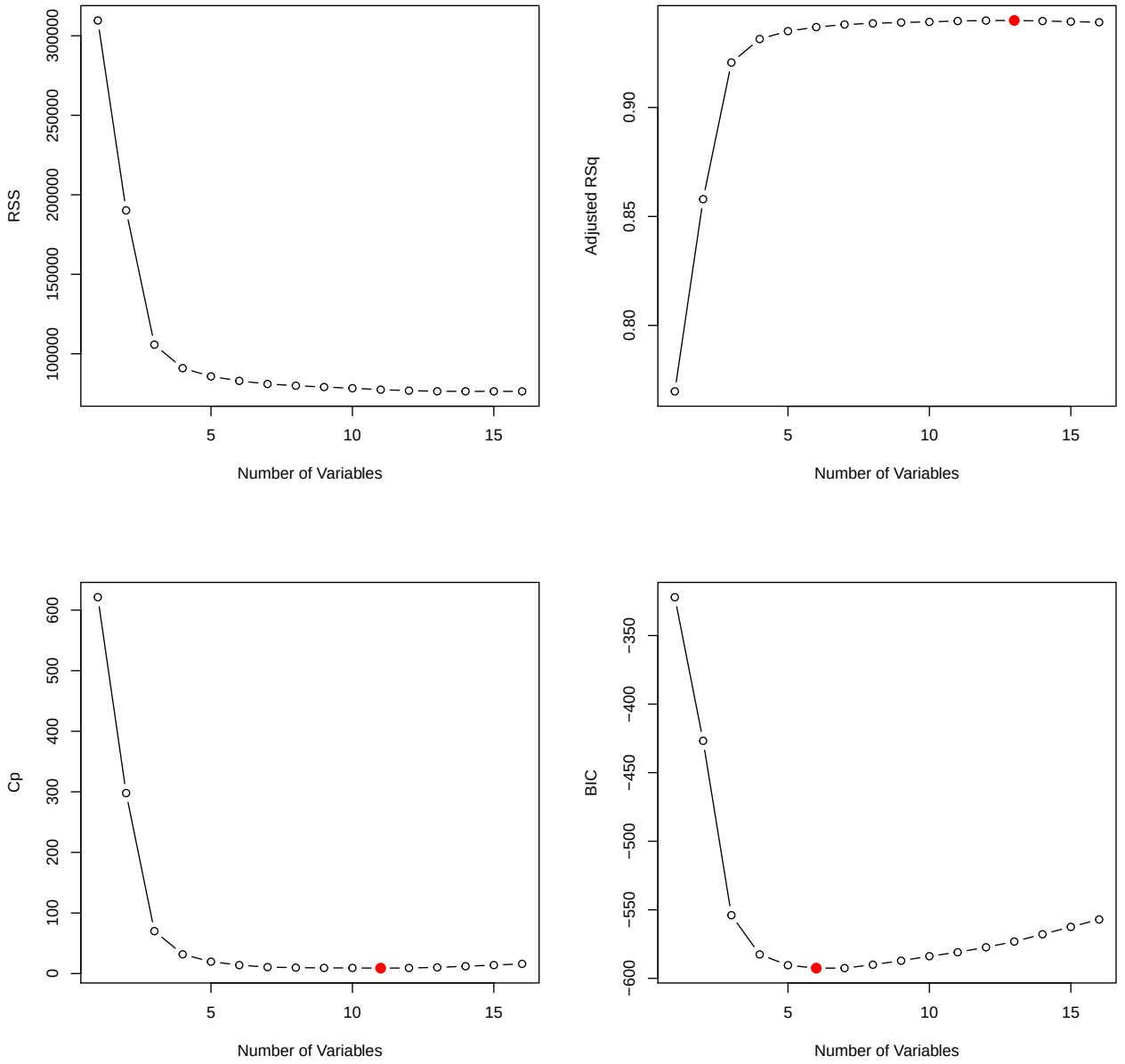
```
library(leaps)
catdat.bestsub <- regsubsets(birds ~ ., data = catdat.train, nvmax = 16)
(sum.bestsub <- summary(catdat.bestsub))
```

```
## Subset selection object
## Call: regsubsets.formula(birds ~ ., data = catdat.train, nvmax = 16)
## 17 Variables  (and intercept)
##                   Forced in Forced out
## sex                   FALSE      FALSE
## weight                FALSE      FALSE
## dryfood               FALSE      FALSE
## wetfood               FALSE      FALSE
## age                   FALSE      FALSE
## owner.income          FALSE      FALSE
## daily.playtime        FALSE      FALSE
## fellow.cats           FALSE      FALSE
## owner.age             FALSE      FALSE
## house.area            FALSE      FALSE
## children.13           FALSE      FALSE
## urban                 FALSE      FALSE
## bell                  FALSE      FALSE
## dogs                  FALSE      FALSE
## daily.outdoortime     FALSE      FALSE
## daily.catnip          FALSE      FALSE
## neutered              FALSE      FALSE
## 1 subsets of each size up to 16
## Selection Algorithm: exhaustive
##           sex weight dryfood wetfood age owner.income daily.playtime
## 1  ( 1 )  " " " "    " "     " "     " " " "          " "
## 2  ( 1 )  " " " "    " "     " "     " " " "          "*"
## 3  ( 1 )  " " " "    " "     " "     " " " "          "*"
## 4  ( 1 )  " " " "    " "     "*"     " " " "          "*"
## 5  ( 1 )  " " " "    " "     "*"     " " " "          "*"
## 6  ( 1 )  " " " "    " "     "*"     " " " "          "*"
## 7  ( 1 )  " " "*"    " "     "*"     " " " "          "*"
## 8  ( 1 )  " " "*"    " "     "*"     " " " "          "*"
```

```
## 9  ( 1 )  " " "*"    "*"     "*"     " " " " "          "*"
## 10  ( 1 ) " " "*"    "*"     "*"     " " " " "          "*"
## 11  ( 1 ) " " "*"    "*"     "*"     " " " " "          "*"
## 12  ( 1 ) " " "*"    "*"     "*"     " " " " "          "*"
## 13  ( 1 ) " " "*"    "*"     "*"     " " "*" "          "*"
## 14  ( 1 ) "*" "*"    "*"     "*"     " " "*" "          "*"
## 15  ( 1 ) "*" "*"    "*"     "*"     "*" "*" "          "*"
## 16  ( 1 ) "*" "*"    "*"     "*"     "*" "*" "          "*"
##           fellow.cats owner.age house.area children.13 urban bell dogs
## 1  ( 1 )  " "         " "       " "        " "          " "   " "  " "
## 2  ( 1 )  " "         " "       " "        " "          " "   " "  " "
## 3  ( 1 )  " "         " "       " "        " "          " "   "*"  " "
## 4  ( 1 )  " "         " "       " "        " "          " "   "*"  " "
## 5  ( 1 )  " "         " "       " "        " "          "*"   "*"  " "
## 6  ( 1 )  " "         " "       " "        "*"          "*"   "*"  " "
## 7  ( 1 )  " "         " "       " "        "*"          "*"   "*"  " "
## 8  ( 1 )  " "         " "       " "        "*"          "*"   "*"  " "
## 9  ( 1 )  " "         " "       " "        "*"          "*"   "*"  " "
## 10  ( 1 ) " "         "*"       " "        "*"          "*"   "*"  " "
## 11  ( 1 ) "*"         " "       " "        "*"          "*"   "*"  "*"
## 12  ( 1 ) "*"         "*"       " "        "*"          "*"   "*"  "*"
## 13  ( 1 ) "*"         "*"       " "        "*"          "*"   "*"  "*"
## 14  ( 1 ) "*"         "*"       " "        "*"          "*"   "*"  "*"
## 15  ( 1 ) "*"         "*"       " "        "*"          "*"   "*"  "*"
## 16  ( 1 ) "*"         "*"       "*"        "*"          "*"   "*"  "*"
##           daily.outdoortime daily.catnip neutered
## 1  ( 1 )  "*"               " "          " "
## 2  ( 1 )  "*"               " "          " "
## 3  ( 1 )  "*"               " "          " "
## 4  ( 1 )  "*"               " "          " "
## 5  ( 1 )  "*"               " "          " "
## 6  ( 1 )  "*"               " "          " "
## 7  ( 1 )  "*"               " "          " "
## 8  ( 1 )  "*"               " "          "*"
## 9  ( 1 )  "*"               " "          "*"
## 10  ( 1 ) "*"               " "          "*"
## 11  ( 1 ) "*"               " "          "*"
## 12  ( 1 ) "*"               " "          "*"
## 13  ( 1 ) "*"               " "          "*"
## 14  ( 1 ) "*"               " "          "*"
## 15  ( 1 ) "*"               " "          "*"
## 16  ( 1 ) "*"               " "          "*"
```

```r
# Plot RSS, Adjusted R^2, C_p and BIC
par(mfrow = c(2, 2))
plot(sum.bestsub$rss, xlab = "Number of Variables", ylab = "RSS", type = "b")
plot(sum.bestsub$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "b")
bsm_best_adjr2 = which.max(sum.bestsub$adjr2)
points(bsm_best_adjr2, sum.bestsub$adjr2[bsm_best_adjr2], col = "red", cex = 2, pch = 20)
plot(sum.bestsub$cp, xlab = "Number of Variables", ylab = "Cp", type = "b")
bsm_best_cp = which.min(sum.bestsub$cp)
points(bsm_best_cp, sum.bestsub$cp[bsm_best_cp], col = "red", cex = 2, pch = 20)
bsm_best_bic = which.min(sum.bestsub$bic)
plot(sum.bestsub$bic, xlab = "Number of Variables", ylab = "BIC", type = "b")
```

```
points(bsm_best_bic, sum.bestsub$bic[bsm_best_bic], col = "red", cex = 2, pch = 20)
```



Based on BIC, we would choose the model with 6 predictors, although we see that the other criterion choose models with more predictors. The predictors that minimize BIC are:

```
coef(catdat.bestsub, bsm_best_bic)
```

```
##       (Intercept)           wetfood    daily.playtime       children.13
##         106.097330        -10.209847        -10.311803          4.224025
##             urban              bell daily.outdoortime
##         -10.573511        -49.364863          1.142209
```

```
predict.regsubsets = function(object, newdata, id, ...) {
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object, id = id)
    xvars = names(coefi)
```

```
    mat[, xvars] %*% coefi
}
```

```
mse.bestsub <- mean((catdat.test$birds - predict.regsubsets(catdat.bestsub, catdat.test,
    id = bsm_best_bic))^2)
mse.bestsub
```

```
## [1] 299.8835
```

c) (2P) - Lasso regression

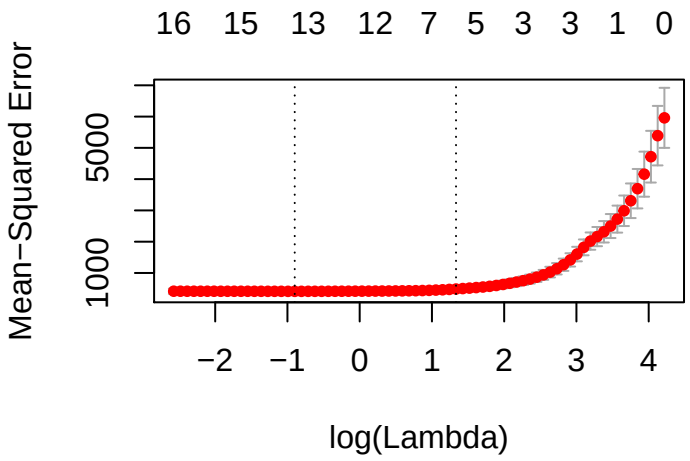
Now use lasso regression on the same data set. Explain how you choose λ. Report the non-zero coefficients, and the test MSE.

R-hints:

```
x.train <- model.matrix(birds ~ ., data = catdat.train)[, -1]
y.train <- catdat.train$birds
x.test = model.matrix(birds ~ ., data = catdat.test)[, -1]
y.test = catdat.test$birds
```

Solution

```
library(glmnet)
cv.lasso <- cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.lasso)
```




```
cv.lasso$lambda.min
```

```
## [1] 0.4068483
```

```
catdat.lasso <- glmnet(x.train, y.train, alpha = 1, lambda = cv.lasso$lambda.min)

coef(catdat.lasso)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)         1.231262e+02
## sex                 .
## weight             -2.769850e+00
## dryfood            -2.749701e+00
## wetfood            -7.763341e+00
```

```
## age                 .
## owner.income       -1.189554e-05
## daily.playtime     -1.021128e+01
## fellow.cats         .
## owner.age          -1.709233e-01
## house.area          7.622237e-02
## children.13         3.814341e+00
## urban              -8.461236e+00
## bell               -5.014600e+01
## dogs                3.811795e+00
## daily.outdoortime   1.139921e+00
## daily.catnip        .
## neutered           -4.649205e+00
```

```
mse.lasso = mean((y.test - predict(catdat.lasso, newx = x.test))^2)
mse.lasso
```

```
## [1] 298.5064
```

d) (1P)

For the lasso regression, what happens when $\lambda \to \infty$? What happens when $\lambda = 0$?

Solution $\lambda \to \infty$: intercept only. $\lambda = 0$: ordinary linear regression.

e) (2P)

Now check whether the test MSE is actually better for the models in b) and c), compared to

i) a model with only intercept, and
ii) a multiple linear regression using all covariates.

Solution Intercept-only MSE can for example be found by

```
mean((mean(y.train) - y.test)^2)
```

```
## [1] 4914.069
```

or

```
intercept.pred <- predict(glmnet(x.train, y.train, alpha = 1, lambda = 1e+10), newx = x.test)
(mse.intercept <- mean((intercept.pred - y.test)^2))
```

```
## [1] 4914.069
```

For the ordinary linear regression we could of course use 'lm', or we can use the same lasso regression command with λ set to zero.

```
ols.pred <- predict(glmnet(x.train, y.train, alpha = 1, lambda = 0), newx = x.test)
lm.pred <- predict(lm(birds ~ ., data = catdat.train), newdata = as.data.frame(x.test))
(mse.ols <- mean((ols.pred - y.test)^2))
```

```
## [1] 302.4566
```

```
(mse.lm <- mean((lm.pred - y.test)^2))
```

```
## [1] 301.9186
```

f) (1P)

Present all the MSE values from the best subset selection, lasso regression, intercept-only and ordinary linear regression in a table. Explain what you see. Does it fit with what you would have expected?

Solution

```
mse.df <- data.frame(bestsubset = mse.bestsub, lasso = mse.lasso, intercept = mse.intercept,
    ols = mse.ols)
mse.df
```

```
##   bestsubset    lasso intercept      ols
## 1   299.8835 298.5064  4914.069 302.4566
```

We see that lasso and best subset regression seem to be the best here. However, ordinary linear regression is not that much worse in this case. Not exactly sure why this is. There are many variabes in the data set that don't really contribute that much to the response, but for some reason it doesn't overfit when they are included.

Problem 2 (6P)

a) (2P) - Multiple choice

Which of the following statements are true, which false?

(i) A *natural cubic spline* has fewer degrees of freedom than a *cubic spline*.
(ii) Smoothing splines attempt to reduce the variance of a fit by penalizing the second-order derivative of the fit.
(iii) A regression spline with polynomials of degree $M-1$ has continous derivatives up to order $M-2$, but not at the knots.
(iv) A regression spline of order 3 with 4 knots has 8 basis functions (not counting the intercept)

Solution

TRUE - TRUE - FALSE - FALSE

(i) Yes, since the fit is linear beyond the boundary knots we free up some degrees of freedom. (ii) Yes, a large penalty on the second-order derivative will lead to a linear fit. (iii) The derivative at the knots is also continuous. (iv) We need 7 basis functions, but estimate 8 parameters (including the intercept, which is not a basis function).

b) (2P)

Write down the basis functions for a cubic spline with knots at the quartiles q_1, q_2 of variable X.

Solution

$$h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = X^2,$$
$$h_4(X) = X^3, \quad h_5(X) = (X - q_1)_+^3, \quad h_6(X) = (X - q_2)_+^3.$$

Here we have to be a bit strict with the correction. If student foget the "+" symbol, for example, you should derive 0.5 for each case.

c) (2P)

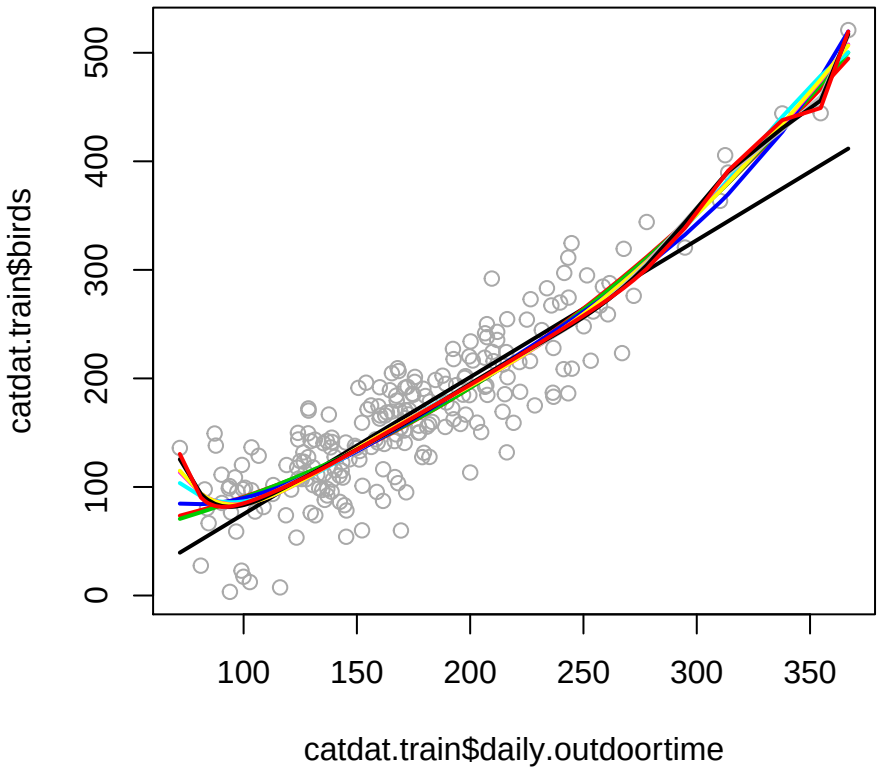
When you look at the plot of `birds` against `daily.outdoortime` in the cat-dataset we used in problem 1, it may look like there is a slight non-linearity in the relationship.

(i) Fit polynomial regression model for `birds` with `daily.outdoortime` as the only covariate for a range of polynomial degrees ($d = 1, \ldots, 10$) and plot the results. Use the training data for this task.

(ii) Report the *training* and *test* MSE for (i), and explain what is happening.

Solution:

```
mse_train = c()
mse_test = c()
plot(catdat.train$daily.outdoortime, catdat.train$birds, type = "p", col = "darkgrey")
for (i in 1:10) {
    poly.mod = lm(birds ~ poly(daily.outdoortime, degree = i), data = catdat.train)
    lines(catdat.train$daily.outdoortime[order(catdat.train$daily.outdoortime)],
        poly.mod$fitted.values[order(catdat.train$daily.outdoortime)], col = i, lwd = 2)
    mse_train[i] = mean(poly.mod$residuals^2)
    mse_test[i] <- mean((predict(poly.mod, newdata = catdat.test) - catdat.test$birds)^2)
}
```




```
mse_train
```

```
##  [1] 1369.997 1199.202 1198.499 1188.227 1175.981 1173.485 1173.437 1166.437
##  [9] 1166.019 1164.368
```

```
mse_test
```

```
##  [1] 1283.779 1102.879 1105.324 1130.092 1129.765 1121.386 1122.211 1131.431
##  [9] 1131.832 1137.369
```

The training error is becoming smaller for higher flexibility (i.e., polynomial degree). This is expected and is kown as overfitting. On the other hand, the test error has a minimum for polynomial degree 2. Such a minimum is expected because of the bias-variance trade-off.

Problem 3 (12P)

a) (2P) - Multiple choice

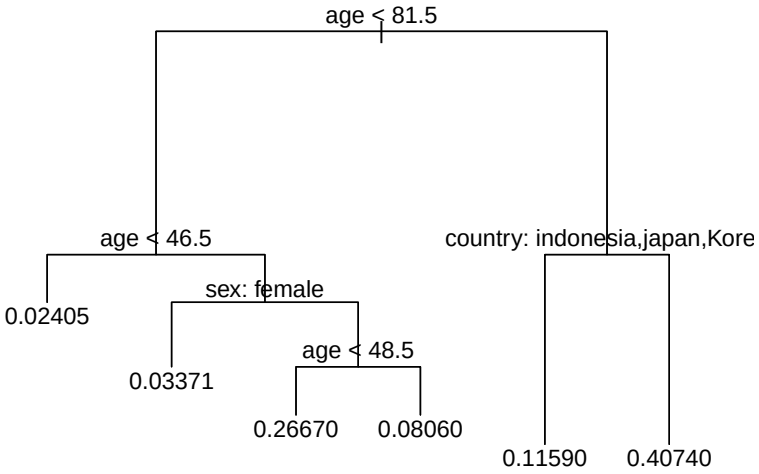
We are using again the Covid-19 dataset that we analyzed in compulsory exercise 1. Remember that the outcome was a binary variable that indicated if a patient died (`deceased=1`) or not (`deceased=0`). This time we are building a regression tree where we use the response as a numeric value (why?), and then apply cost-complexity pruning (figures and code below). Which of the following statements are true, which false?

(i) The probability of dying (`deceased = 1`) is about 40.7% for a French person with age above 81.5.

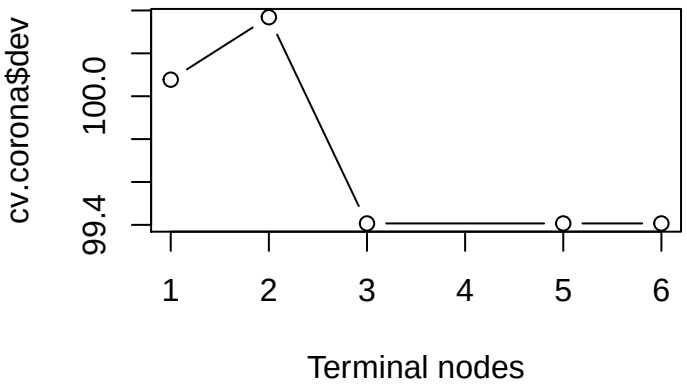
(ii) Age seems to be a more important predictor for mortality than sex.

(iii) Cost-complexity pruning was done using 10-fold CV.

(iv) It looks like the tree with 6 terminal nodes was the best choice, so there is no need to prune the original tree.




```
set.seed(1)
cv.corona = cv.tree(t.corona, K = 5)
plot(cv.corona$size, cv.corona$dev, type = "b", xlab = "Terminal nodes")
```



Solution: TRUE - TRUE - FALSE - FALSE

(iii) we used $k = 5$ not 10; (iv) The tree with 6 nodes is as good as the one with 3 nodes, so we should choose the simpler one.

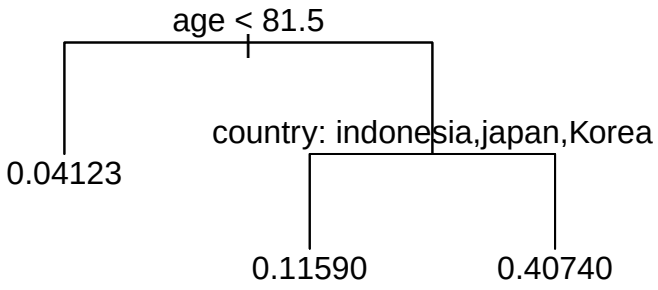
b) (2P)

Imagine that you decide to prune the tree from a) down to three leaves. From looking at the tree above, which three leaves are they (1P) and why (1P)? You do not need to carry out an analysis, only look at the tree above and argue. You may choose to draw a figure, or to only describe your tree with words.

Solution: The first cut at the root is age < 81.5, and that already leads to two nodes. Now the question is wheter we will keep a second split on the left or the right branch. The split that is chosen is the one based on country, because this one leads to a much larger reduction in the deviance (visible from the branch lengths).

We are here anyway doing the analysis, but the students should argue without doing this:

```
prune.corona = prune.tree(t.corona, best = 3)
plot(prune.corona)
text(prune.corona, pretty = 0)
```



c) (6P)

We will use the classical data set of *diabetes* from a population of women of Pima Indian heritage in the US, available in the R `MASS` package. The following information is available for each woman:

- diabetes: `0`= not present, `1`= present
- npreg: number of pregnancies
- glu: plasma glucose concentration in an oral glucose tolerance test
- bp: diastolic blood pressure (mmHg)
- skin: triceps skin fold thickness (mm)
- bmi: body mass index (weight in kg/(height in m)^2)
- ped: diabetes pedigree function.
- age: age in years

We will use a training set (called `d.train`) with 300 observations (200 non-diabetes and 100 diabetes cases) and a test set (called `d.test`) with 232 observations (155 non-diabetes and 77 diabetes cases). Our aim is to make a classification rule for the presence of diabetes (yes/no) based on the available data. You can load the data as follows:

```
id <- "1Fv6xwKLSZHldRAC1MrcK2mzdOYnbgv0E"  # google file ID
d.diabetes <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
d.train = d.diabetes$ctrain
d.test = d.diabetes$ctest
```

We are interested in a tree-based method in order to build a model to predict the presence for `diabetes`. In addition, we want to understand which factors are most relevant in predicting diabetes.

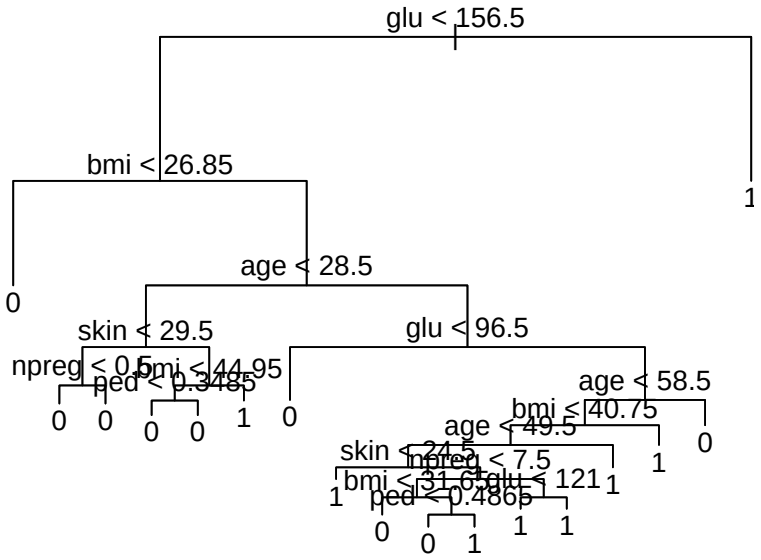
(i) (3P) Start by using a simple classification tree. Apply cost-complexity pruning using 10-fold CV on the training data set. Report the misclassification error on the test set.

(ii) (3P) Now construct a classification tree based on a more advanced method. Train the model using the training data and report the misclassification error for the test data. Explain your choice of the (tuning) parameters. Which two variables are the most influential ones in the prediction of diabetes?

R-hint: Please use `set.seed(1)` before your run cross-validation in task (i), so that it is easier to reproduce your results.

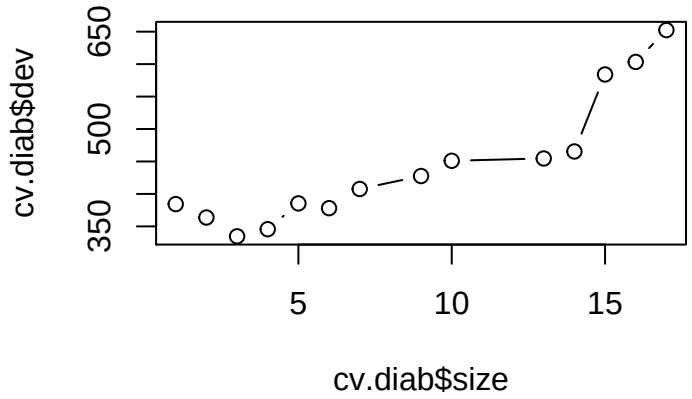
Solution

(i) Students might choose the *deviance* (cross entropy) or the *Gini index*. Results might therefore differ from what we give here (using the deviance):

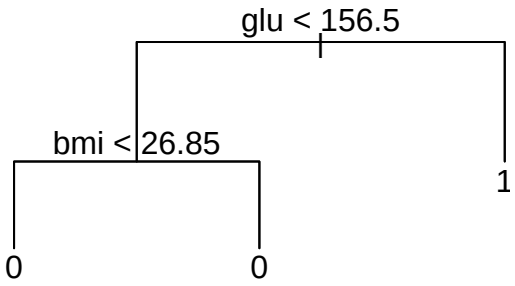


We do the pruning first for the default parameters in the `cv.tree` function

```
set.seed(1)
cv.diab = cv.tree(t.diab, K = 10)
plot(cv.diab$size, cv.diab$dev, type = "b")
```




```
prune.diab = prune.tree(t.diab, best = 3)
plot(prune.diab)
text(prune.diab, pretty = 0)
```




```
library(caret)
t.pred.prune <- predict(prune.diab, d.test, type = "class")
(confMat <- confusionMatrix(t.pred.prune, d.test$diabetes)$table)
```

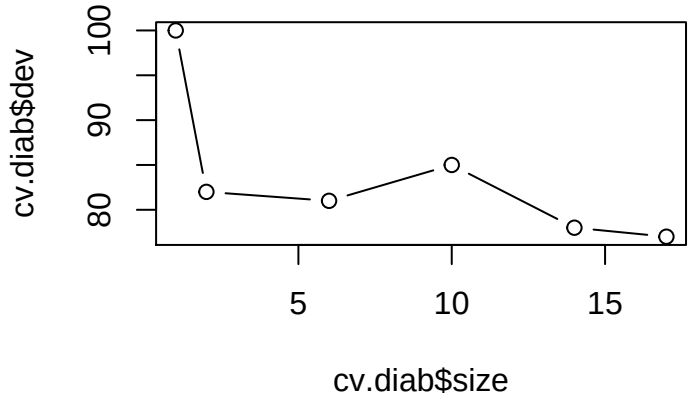
```
##           Reference
## Prediction   0   1
##          0 148  49
##          1   7  28
```

```
1 - (sum(diag(confMat))/sum(confMat[1:2, 1:2]))
```

```
## [1] 0.2413793
```

Some students might want to use the misclassification error to do pruning, thus this is also correct:

```
set.seed(1)
cv.diab = cv.tree(t.diab, K = 10, FUN = prune.misclass)
plot(cv.diab$size, cv.diab$dev, type = "b")
```



In that case, no pruning is required!

```
library(caret)
t.pred.prune <- predict(prune.diab, d.test, type = "class")
(confMat <- confusionMatrix(t.pred.prune, d.test$diabetes)$table)
```

```
##           Reference
## Prediction   0   1
##          0 148  49
##          1   7  28
```

```
1 - (sum(diag(confMat))/sum(confMat[1:2, 1:2]))
```

```
## [1] 0.2413793
```

Interestingly, that tree that we did not need to prune (according to the misclassification error pruning) gives a sligthly higher error than the tree we pruned with the deviance! (But this is not something we expect the

students to see – they only have to apply one of the two pruning methods.)

(ii) We use the random forest approach. An alternative would be boosting, but we did not discuss it for classification trees.

In a random forest, we have two parameters to decide: `mtry` and `ntree`.

```
library(randomForest)
set.seed(1)
rf.diab = randomForest(diabetes ~ ., data = d.train, ntree = 1000, importance = TRUE)
rf.diab
```

```
##
## Call:
##  randomForest(formula = diabetes ~ ., data = d.train, ntree = 1000,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 21%
## Confusion matrix:
##     0  1 class.error
## 0 176 24        0.12
## 1  39 61        0.39
```

```
1 - sum(diag(rf.diab$confusion))/sum(rf.diab$confusion[1:2, 1:2])
```

```
## [1] 0.21
```

Here the students should mention that $\sqrt{7} = 2.65$, thus `mtry` should be 2 (used by default) or 3. The number of trees is not a tuning parameter, so it should be chosen large enough.

```
t.pred.rf <- predict(rf.diab, d.test, type = "class")
(confMat <- confusionMatrix(t.pred.rf, d.test$diabetes)$table)
```

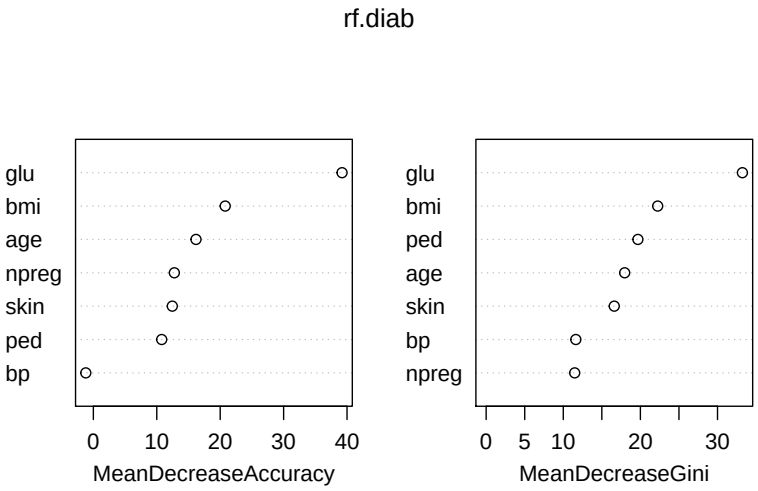
```
##           Reference
## Prediction   0   1
##          0 134  33
##          1  21  44
```

```
1 - (sum(diag(confMat))/sum(confMat[1:2, 1:2]))
```

```
## [1] 0.2327586
```

The misclassification error is now slightly smaller than for the simple tree-based method, though the difference is not very impressive. To assess variable importance we can create a variable importance plot, or print the importance. According to those, plasma glucose concentration (glu) and BMI seem the most predictive variables.

```
##               0         1 MeanDecreaseAccuracy MeanDecreaseGini
## npreg 13.837504  2.255417            12.771036         11.48233
## glu   29.312304 30.634655            39.216469         33.23059
## bp     2.327397 -4.405279            -1.192003         11.62648
## skin   9.947283  6.642430            12.446610         16.61112
## bmi   13.990208 15.809096            20.791471         22.24202
## ped    8.012437  7.575117            10.774631         19.67667
## age   14.483748  6.847745            16.183666         17.96268
```



Problem 4 (12P)

a) (2P) - Multiple choice

Imagine you have gene expression data for leukemia patients, with $p = 4387$ genes measured on blood samples for a total of $n = 92$ patients, of wich 42 have leukemia and 50 patients are healthy. Which statements are true?

(i) In this dataset we are guaranteed to find a separating hyperplane, unless there are exact feature ties (two patients with the exact same gene data, but different outcome).
(ii) In the analysis of this dataset we should prefer a soft-margin classifier over a separating hyperplane to omit overfitting.
(iii) Logistic regression is the preferred method for this data set, because it gives nice interpretable parameter estimates.
(iv) By choosing a large budget parameter C we are making the model more robust, but introduce more bias.

Solution: TRUE - TRUE - FALSE - TRUE

(iii) Log. reg is not possible, because $p > n$. (iv) bias-variance trade-off: C large means we are allowing more violations of the boundaries, thus we increase the bias, reduce the variance and make the model more robust.

b) (7P)

We are looking at a (subset of) a dataset that contains gene expression data for 60 patients and more than 22'000 genes per patient. It is generated from the U133A platform and collected from The Children's Hospital at Westmead and was extracted and modified using the original publication by Anaissi et al (2016; https://doi.org/10.1371/journal.pone.0157330). Here we only use a subset of 10'000 genes for computational efficiency:

```
id <- "1x_E8xnmz9CMHh_tMwIsWP94czPa1Fpsj"  # google file ID
d.leukemia <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id), header = T)
```

We are splitting the dataset into a training set and a test set of size 45 and 15, respectively:

```
set.seed(2399)
t.samples <- sample(1:60, 15, replace = F)
d.leukemia$Category <- as.factor(d.leukemia$Category)
d.leukemia.test <- d.leukemia[t.samples, ]
d.leukemia.train <- d.leukemia[-t.samples, ]
```

(i) (1P) Why is a support vector machine (SVM) more suitable here than a logistic regression approach? What other approach could be used instead of a SVM?

(ii) (1P) Say in 1-2 sentences what the paper where we have taken the data from (https://doi.org/10.1371/journal.pone.0157330) intends to do, that is, what kind of method do they suggest and what is the purpose?

(iii) (3P) Fit a support vector classifier (linear boundary) to find a function that predicts whether a child that has been successfully treated against leukemia later relapsed or not. Fix the tuning parameter to $C = 1$ (1P). Report the confusion tables and misclassification error rates for both the training and test sets (0.5P each).

- (0.5P) Is it surprising to see the training error rate? Why?
- (0.5P) Which is the most common type of error that you see in the test set? Do you think the classification method is successful?

(iv) (2P) Repeat the analysis with a radial kernel using $C = 1$. Do the analysis twice, one with $\gamma = 10^{-2}$ and once with $\gamma = 10^{-5}$. Interpret the training and test error rates that you find now, and compare to the results in (iii).

R-hints: To run cross-validation over a grid of two tuning parameters, we would usually use the `tune()` function. However this is not an efficient way of studying different tuning parameters for this large dataset.

Solution:

(i) Plain (i.e., unregularized) logistic regression is not able to handle the case $p > n$. Instead of an SVM we could also use a random forest or (better!) principal component regression. Some might also mention neural networks, that's also correct.

(ii) The paper suggests a way to do feature selection (i.e., finding the most predictive features, here: genes) in a SVM context (0.5P). They do this using an ensemble approach, namely bagging (0.5P).

(iii) Linear classifier using the `svm()` function (1P)

```
library(e1071)
r.svm.linear <- svm(Category ~ ., data = d.leukemia.train, kernel = "linear", scale = TRUE,
    cost = 1)
```

Confusion tables using the prediction error (0.5P each):

```
ypred_linear = predict(r.svm.linear, d.leukemia.train)
(tt1 <- table(predict = ypred_linear, truth = d.leukemia.train[, "Category"]))
```

```
##              truth
## predict       Non-Relapse Relapse
##   Non-Relapse          30       0
##   Relapse               0      15
```

```
ypred_linear = predict(r.svm.linear, d.leukemia.test)
(tt1 <- table(predict = ypred_linear, truth = d.leukemia.test[, "Category"]))
```

```
##              truth
## predict       Non-Relapse Relapse
##   Non-Relapse           8       4
```

```
##   Relapse               1       2
```

- It is not surprising to see zero training error: Due to $p > n$ the two groups are linearly separable (and we do not expect any ties in genome data) (0.5P).
- The most common error are false negatives, that is, the prediction is "non-relapse" when the patient actually did relapse. This is not a very successful method, because we are missing the actual cases (0.5P).

(iv) $\gamma = 0.01$:

```
r.svm.radial2 <- svm(Category ~ ., data = d.leukemia.train, kernel = "radial", scale = TRUE,
    cost = 1, gamma = 0.01)

ypred_radial2 = predict(r.svm.radial2, d.leukemia.train)
(tt2 <- table(predict = ypred_radial2, truth = d.leukemia.train[, "Category"]))
```

```
##              truth
## predict       Non-Relapse Relapse
##   Non-Relapse          30       0
##   Relapse               0      15
```

```
ypred_radial2 = predict(r.svm.radial2, d.leukemia.test)
(tt2 <- table(predict = ypred_radial2, truth = d.leukemia.test[, "Category"]))
```

```
##              truth
## predict       Non-Relapse Relapse
##   Non-Relapse           9       6
##   Relapse               0       0
```

$\gamma = 10^{-5}$:

```
r.svm.radial <- svm(Category ~ ., data = d.leukemia.train, kernel = "radial", scale = TRUE,
    cost = 1, gamma = 1e-05)

ypred_radial = predict(r.svm.radial, d.leukemia.train)
(tt2 <- table(predict = ypred_radial, truth = d.leukemia.train[, "Category"]))
```

```
##              truth
## predict       Non-Relapse Relapse
##   Non-Relapse          30      15
##   Relapse               0       0
```

```
ypred_radial = predict(r.svm.radial, d.leukemia.test)
(tt2 <- table(predict = ypred_radial, truth = d.leukemia.test[, "Category"]))
```

```
##              truth
## predict       Non-Relapse Relapse
##   Non-Relapse           9       6
##   Relapse               0       0
```

When we see the results for $\gamma = 0.01$, our reaction should be that we are overfitting: Zero training error, but the sensitivity of the method is zero on the test data.

On the other hand, for $\gamma = 10^{-5}$ things get even worse: We still have a sensitivity of zero for the test data, but now we in addition have sensitivity equals zero for the training data.

c) (3P)

The SVM is an extension of the support vector classifier, by enlarging the feature space using kernels. The polynomial kernel is a popular choice and has the following form

$$K(\boldsymbol{x}_i, \boldsymbol{x}_i') = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j'})^d$$

Show that for a feature space with inputs X_1 and X_2 and for degree $d = 2$, the above kernel can be represented as the inner product

$$K(X, X') = \langle h(X), h(X') \rangle \ ,$$

where $h(X)$ is a 6-dimensional transformation function in an enlarged space, and $X = (X_1, X_2)^\top$ is the input vector. Explicitly derive $h(X) = (h_1(X), \ldots, h_6(X))$.

Solution:

The input vector (X_1, X_2) is denoted by X. We can then expand the kernel as follows:

$$\begin{aligned} K(X, X') &= (1 + \langle X, X' \rangle) \\ &= (1 + X_1 X_1' + X_2 X_2')^2 \\ &= 1 + 2X_1 X_1' + 2X_2 X_2' + (X_1 X_1')^2 + (X_2 X_2')^2 + 2X_1 X_1' X_2 X_2'. \end{aligned}$$

We have that $h_1(X) = 1$, $h_2(X) = \sqrt{2}X_1$, $h_3(X) = \sqrt{2}X_2$, $h_4(X) = X_1^2$, $h_5(X) = X_2^2$, $h_6(X) = \sqrt{2}X_1 X_2$, that is the the enlarged space created by the Kernel $K(X, X') = \langle h(X), h(X') \rangle$ has dimension $M = 6$.

Problem 5 (12P)

a) (2P) - Multiple choice

Which of the following statements are true, and which are false?

(i) In principal component analysis, the second principal component is the linear combination of the predictors that has the largest variance of all the linear combinations that are uncorrelated with the first principal component.

(ii) It makes no difference for the results of PCA if the variables are standardized beforehand or not.

(iii) K-means clustering is robust against differing initial choices of cluster assignments.

(iv) PCA is most helpful when all the variables are uncorrelated.

Solution TRUE - FALSE - FALSE - FALSE

(i) Yes. (ii) No, it does make a difference and they should be standardized. (iii) No, the final clustering can vary dramatically based on the initialization - it finds a local optimum. (iv) No, then we will just end up with as many principal components as we have variables.

b) (3P)

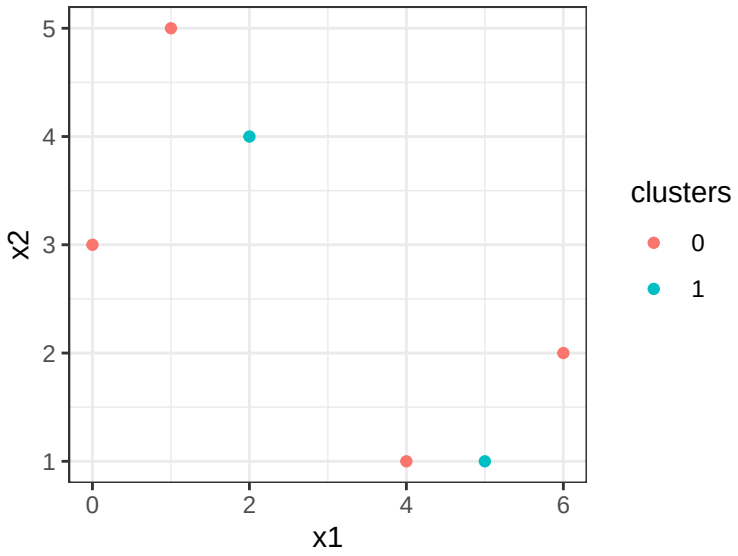
In this problem, we use a simple data set of six observations to preform K-means clustering manually, as specified in algorithm 10.1 on page 388 of the ISLR book. The observations are given below. In this problem we will use $K = 2$.

```r
x1 <- c(1, 2, 0, 4, 5, 6)
x2 <- c(5, 4, 3, 1, 1, 2)
```

(i) (1P) Randomly assign a cluster to each of the observations, and visualize the resulting observations with the color indicating the assigned cluster.

R-hint: You can use the function `sample()` for this. Please use `set.seed(1)` before you sample the random clusters, so it is easier to reproduce.

```r
library(ggplot2)
set.seed(1)
clusters <- sample(as.factor(c(0, 1)), replace = TRUE, size = length(x1))
obs <- data.frame(x1 = x1, x2 = x2, clusters = clusters)
ggplot(obs, aes(x = x1, y = x2, color = clusters)) + geom_point() + theme_bw()
```

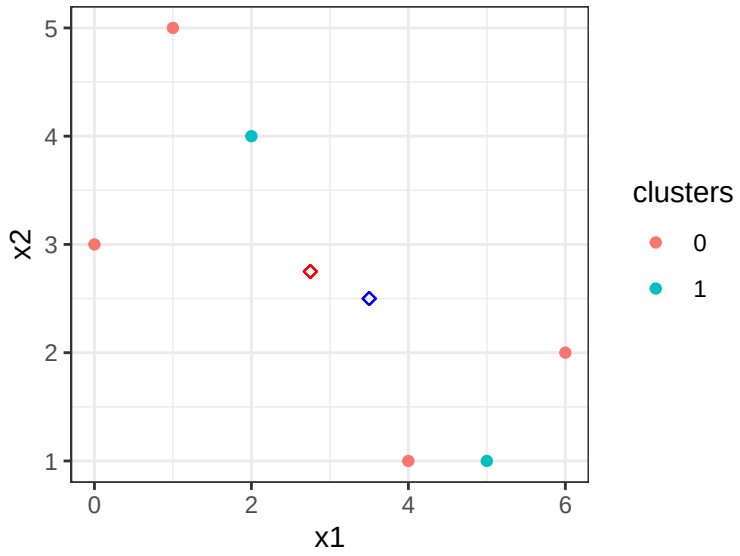


(ii) (1P) Manually calculate the centroids for each cluster (that is, you are encouraged to do the calculations in R, but don't use any non-trivial functions that you haven't written yourself). Plot the centroids on the plot from (i).

```r
library(dplyr)
find_centroid <- function(obs) {
    cluster1 <- obs %>% filter(clusters == 0)
    cluster2 <- obs %>% filter(clusters == 1)
    centroid1 <- c(mean(cluster1$x1), mean(cluster1$x2))
    centroid2 <- c(mean(cluster2$x1), mean(cluster2$x2))
    return(list(centroid1 = centroid1, centroid2 = centroid2))
}

centroids_it1 <- find_centroid(obs)

ggplot(obs, aes(x = x1, y = x2, color = clusters)) + geom_point() + theme_bw() +
    geom_point(aes(centroids_it1$centroid1[1], centroids_it1$centroid1[2]), shape = 23,
        color = "red") + geom_point(aes(centroids_it1$centroid2[1], centroids_it1$centroid2[2]),
    shape = 23, color = "blue")
```



(iii) (1P) Now assign each observation to the centroid which it is closest to, in Euclidean distance. Display the new cluster assignments as in (i).

```
euc.dist <- function(x, centroid) sqrt(sum((x - centroid)^2))

obs
```

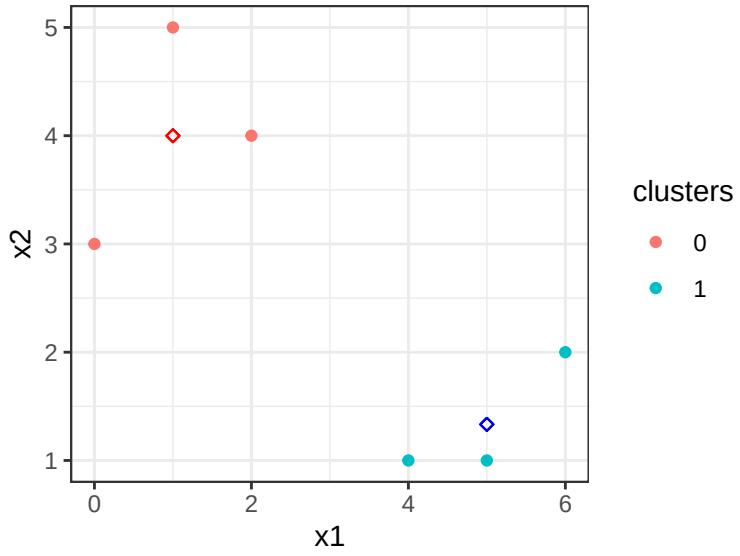
```
##   x1 x2 clusters
## 1  1  5        0
## 2  2  4        1
## 3  0  3        0
## 4  4  1        0
## 5  5  1        1
## 6  6  2        0
```

```
for (i in 1:nrow(obs)) {
    (dist_centroid1 <- euc.dist(obs[i, 1:2], centroids_it1$centroid1))
    (dist_centroid2 <- euc.dist(obs[i, 1:2], centroids_it1$centroid2))
    if (dist_centroid1 < dist_centroid2) {
        obs[i, ]$clusters <- as.factor(0)
    } else {
        obs[i, ]$clusters <- as.factor(1)
    }
}
obs
```

```
##   x1 x2 clusters
## 1  1  5        0
## 2  2  4        0
## 3  0  3        0
## 4  4  1        1
## 5  5  1        1
## 6  6  2        1
```

```
centroids_it2 <- find_centroid(obs)
ggplot(obs, aes(x = x1, y = x2, color = clusters)) + geom_point() + theme_bw() +
    geom_point(aes(centroids_it2$centroid1[1], centroids_it2$centroid1[2]), shape = 23,
        color = "red") + geom_point(aes(centroids_it2$centroid2[1], centroids_it2$centroid2[2]),
```

```
    shape = 23, color = "blue")
```



The following dataset consists of 40 tissue samples with measurements of 1,000 genes. The first 20 tissues come from healthy patients and the remaining 20 come from a diseased patient group. The following code loads the dataset into your session with colum names decribing if the tissue comes from a diseased or healthy person.

```
id <- "1VfVCQvWt121UN39NXZ4aR9Dmsbj-p9OU"  # google file ID
GeneData <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id), header = F)
colnames(GeneData)[1:20] = paste(rep("H", 20), c(1:20), sep = "")
colnames(GeneData)[21:40] = paste(rep("D", 20), c(1:20), sep = "")
row.names(GeneData) = paste(rep("G", 1000), c(1:1000), sep = "")
GeneData = t(GeneData)
GeneData <- scale(GeneData)
```

c) (2P)

Perform hierarchical clustering with complete, single and average linkage using **both** Euclidean distance and correlation-based distance on the dataset. Plot the dendograms. Hint: You can use `par(mfrow=c(1,3))` to plot all three dendograms on one line or `par(mfrow=c(2,3))` to plot all six together.

Solution:

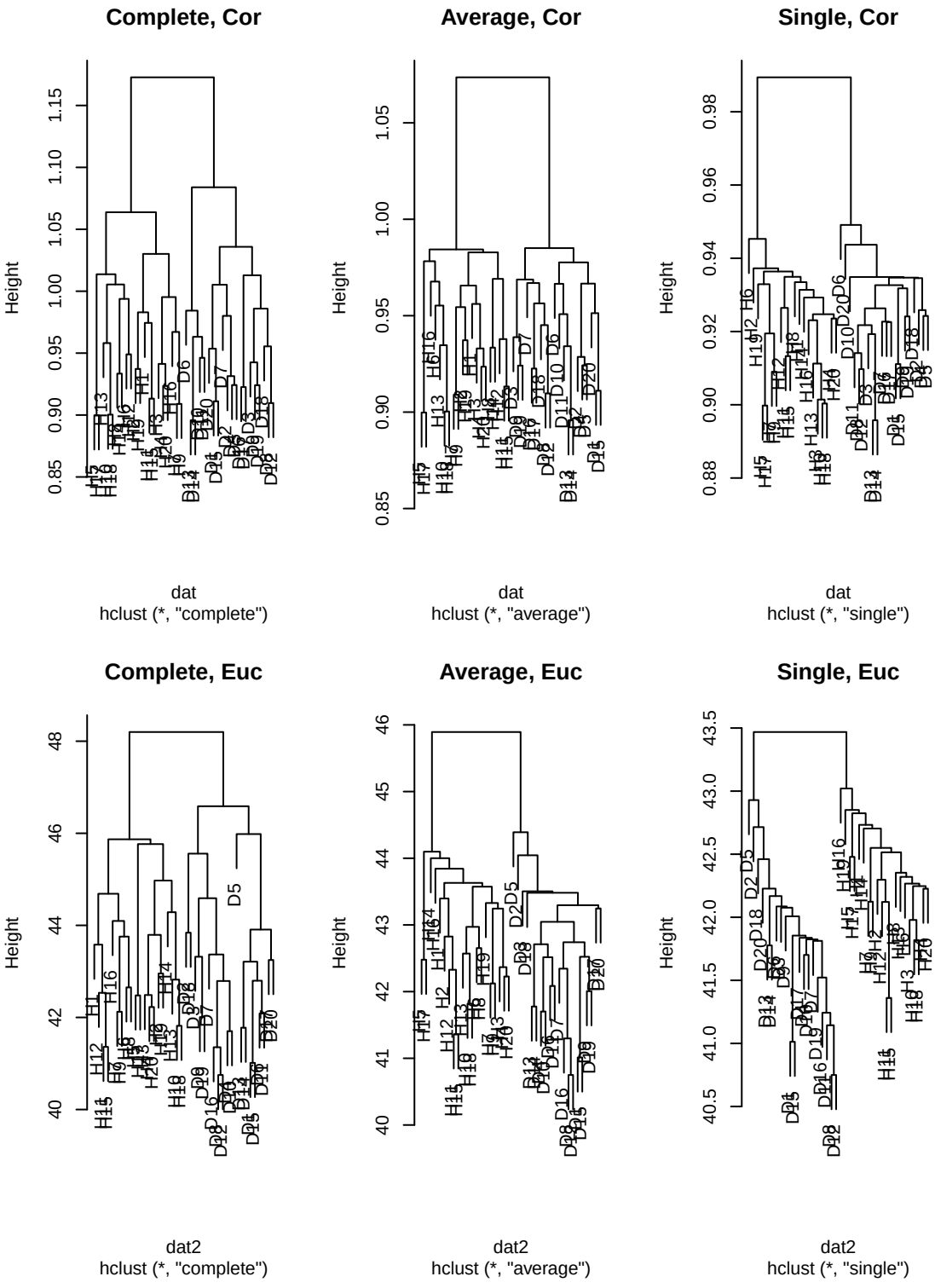
```
# dat = as.dist(t(d)) dat = as.dist(1-cor(d)) correlation based
dat = as.dist(1 - cor(t(GeneData)))
# euclidean distance
dat2 = dist((GeneData), method = "euclidian")

# hier.clust
hc.complete = hclust(dat, method = "complete")
hc.average = hclust(dat, method = "average")
hc.single = hclust(dat, method = "single")

hc.complete2 = hclust(dat2, method = "complete")
```

```r
hc.average2 = hclust(dat2, method = "average")
hc.single2 = hclust(dat2, method = "single")
```

```r
par(mfrow = c(2, 3))
plot(hc.complete, main = "Complete, Cor")
plot(hc.average, main = "Average, Cor")
plot(hc.single, main = "Single, Cor")
plot(hc.complete2, main = "Complete, Euc")
plot(hc.average2, main = "Average, Euc")
plot(hc.single2, main = "Single, Euc")
```



d) (2P)

Use these dendograms to cluster the tissues into two groups. Compare the groups with respect to the patient group the tissue comes from. Which linkage and distance measure performs best when we know the true state of the tissue?

Solution:

```
# cut tree into two braches
clustComp = cutree(hc.complete, k = 2)
clustSing = cutree(hc.single, k = 2)
clustAv = cutree(hc.average, k = 2)

clustComp2 = cutree(hc.complete2, k = 2)
clustSing2 = cutree(hc.single2, k = 2)
clustAv2 = cutree(hc.average2, k = 2)

# actual group:
trueGroup = c(rep(1, 20), rep(2, 20))
Comp = table(trueGroup, clustComp)
Sing = table(trueGroup, clustSing)
Av = table(trueGroup, clustAv)
Comp2 = table(trueGroup, clustComp2)
Sing2 = table(trueGroup, clustSing2)
Av2 = table(trueGroup, clustAv2)

errorRate = function(data) {
    return((sum(data) - sum(diag(data)))/(sum(data)))
}

error = c(errorRate(Comp), errorRate(Sing), errorRate(Av), errorRate(Comp2), errorRate(Sing2),
    errorRate(Av2))
error
```

```
## [1] 0 0 0 0 0 0
```

All three linkages with Euclidean distance have zero errors - so it doesn't matter which one we choose next. (Not just all three with euclidean distance - the correlation based distance as well!)

e) (2P)

(i) (1P) Use PCA to plot the samples in two dimensions. Color the samples based on the tissues group of patients.

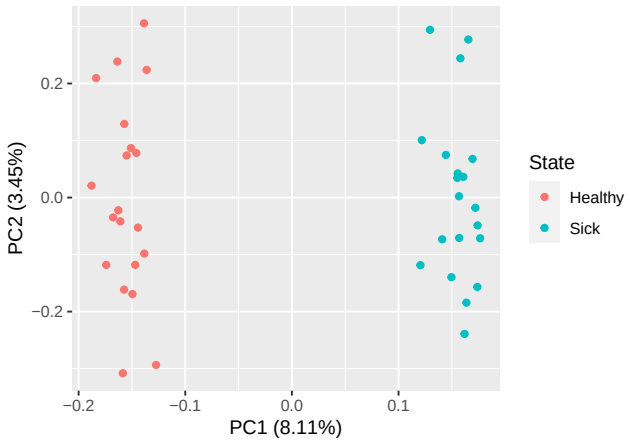
(ii) (1P) How much variance is explained by the first 5 PCs?

Solution:

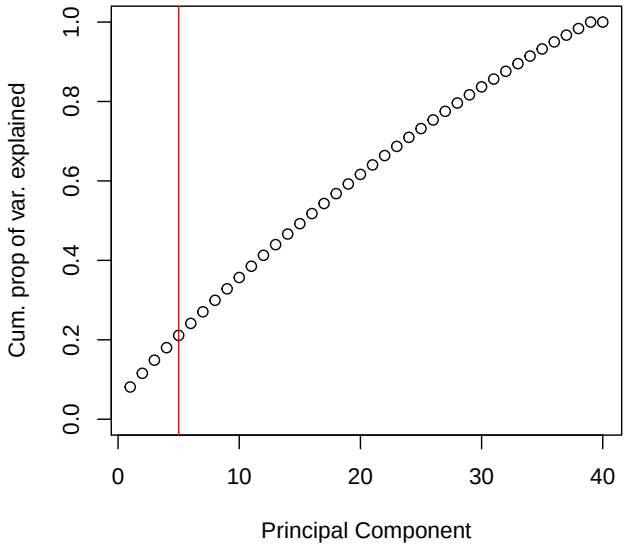
```
# transform and add column for state (diseased/healthy)
PCdata = as.data.frame((GeneData))

PCdata$State = c(rep("Healthy", 20), rep("Sick", 20))

library(ggfortify)
pr.out = prcomp(PCdata[, 1:1000])  # , scale = T)
autoplot(pr.out, data = PCdata, colour = "State")
```




```
# Proportion of variance explained
pr.var = pr.out$sdev^2
pve = pr.var/(sum(pr.var))
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cum. prop of var. explained",
    ylim = c(0, 1), type = "b")
abline(v = 5, col = "red")
```




```
cumsum(pve)[5]
```

```
## [1] 0.2109659
```

The first five prinicipal component explain 21.1% of the variance in the data. However, we see from the plot that the first PC clearly separates the two groups.

f) (1P)

Use your results from PCA to find which genes that vary the most accross the two groups.

Solution:

Look at loadings/rotation vector of PCA and find which genes that contribute the most.

```r
# which genes contribute the most to PC1?
ImpGenes = names(sort(abs(pr.out$rotation[, 1]), decreasing = T)[1:10])
pr.out$rotation[match(ImpGenes, row.names(pr.out$rotation)), 1]
```

```
##       G502       G589       G565       G590       G600       G551       G593
## 0.09485044 0.09449766 0.09183823 0.09173169 0.09167322 0.08768360 0.08758616
##       G538       G584       G509
## 0.08745400 0.08690858 0.08661015
```

All these genes contribute with positive loadings for PCA, meaning that high values of these genes contribute to the disease group and low values point to the healthy group.