Compulsory Exercise 2 – Solution

TMA4268 Statistical Learning V2022

Daesoo Lee, Emma Skarstein, Stefanie Muff, Department of Mathematical Sciences, NTNU

Hand out date: March 21, 2022

R packages

You need to install the following packages in R to run the code in this file. It is of course also possible to use more or different packages.

```
install.packages("ggplot2")
install.packages("tidyverse")
install.packages("palmerpenguins")
install.packages("GGally")
install.packages("MASS")
install.packages("caret")
install.packages("leaps")
install.packages("glmnet")
install.packages("pls")
install.packages("gam")
install.packages("e1071")
install.packages("tree")
install.packages("randomForest")
install.packages("ggfortify")
```

Problem 1 (10P)

In the following questions in this problem, we use the *Boston Housing Price* dataset. A detailed description for the dataset can be found here. Note that a response variable is `medv` (= housing price) and predictor variables are the rest of the covariates. The dataset is preprocessed and split into 80% and 20% for a training set and test set, respectively. In this problem, several feature selection methods are addressed.

```
library(MASS)
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
```

```
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```r
set.seed(1)

# pre-processing by scaling NB! Strictly speaking, pre-processing should be done
# on a training set only and it should be done on a test set with statistics of
# the pre-processing from the training set. But, we're preprocessing the entire
# dataset here for convenience.
boston <- scale(Boston, center = T, scale = T)

# split into training and test sets
train.ind = sample(1:nrow(boston), 0.8 * nrow(boston))
boston.train = data.frame(boston[train.ind, ])
boston.test = data.frame(boston[-train.ind, ])
```

a) (2P)

Perform *Forward Stepwise Selection* and *Backward Stepwise Selection* on `boston.train` method, and plot a graph of adjusted R^2 on the y-axis and a number of predictors on the x-axis.

R-hints:

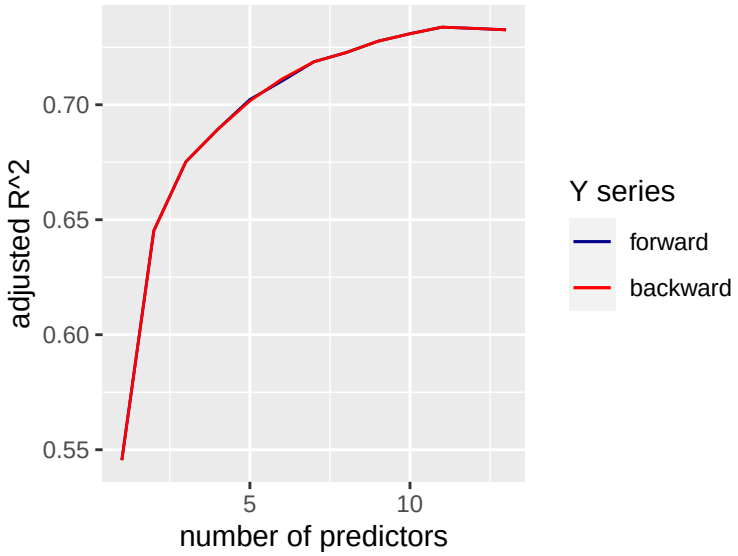
- use `regsubsets(..)` from `library(leaps)` for the selection methods
- use `set.seed(1)`

Solution

```r
set.seed(1)
library(leaps)

run_stepwise_selection_metrics <- function(method) {
    # Perform the stepwise selection using all the predictors in `boston.train`
    n_predictors = ncol(boston.train) - 1
    best_subset_method = regsubsets(medv ~ ., boston.train, nvmax = n_predictors,
        method = method)

    # Save summary obj
    best_subset_method_summary = summary(best_subset_method)
}
```

```r
best_subset_method_summary_forward <- run_stepwise_selection_metrics("forward")

best_subset_method_summary_backward <- run_stepwise_selection_metrics("backward")

# plot
library(ggplot2)
n_predictors = ncol(boston.train) - 1
x_rng = seq(n_predictors)
df <- data.frame(x_rng = x_rng, forward = best_subset_method_summary_forward$adjr2,
```

```
    backward = best_subset_method_summary_backward$adjr2)

ggplot() + geom_line(data = df, aes(y = forward, x = x_rng, colour = "forward")) +
    geom_line(data = df, aes(y = backward, x = x_rng, colour = "backward")) + ylab("adjusted R^2") +
    xlab("number of predictors") + scale_color_manual(name = "Y series", values = c(forward = "darkblue"
    backward = "red"))
```



b) (2P)

Choose the *four* "best" (selected) predictors from the forward stepwise selection. You can use the results obtained in a).

Solution

```
n_selected_predictors <- 4
indices <- best_subset_method_summary_forward$which[n_selected_predictors, ]
selected_predictors <- colnames(best_subset_method_summary_forward$which)[indices]

# remove the `(Intercept)` term
(selected_predictors <- selected_predictors[-c(1)])
```

```
## [1] "rm"      "dis"     "ptratio" "lstat"
```

Therefore, the selected predictors are "rm, dis, ptratio, lstat".

c) (4P)

(i) Run K-fold cross-validation (K=5) on `boston.train` with Lasso and show a plot where the x-axis shows the lambda (or log(lambda)) of Lasso and the y-axis shows MSE (mean squared error). (2P)

(ii) Report the best lambda that you find (i.e., λ with minimum MSE). (1P)

(iii) Report the fitted coefficients at the best λ. (1P)

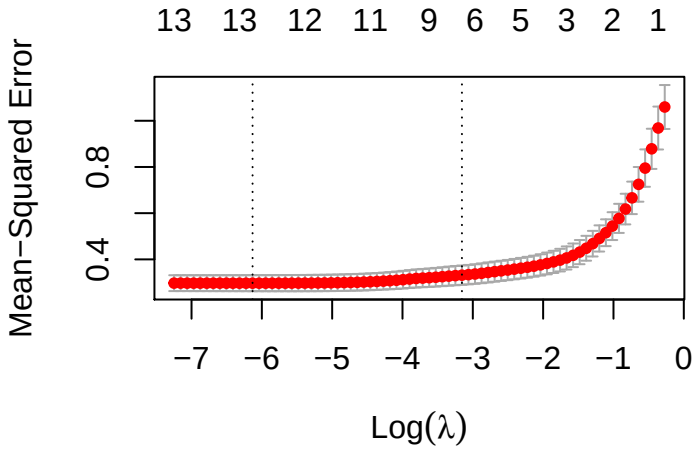
R-hints:

- Use `set.seed(1)`

- you can use `coef()` to print the coefficients (see here)

Solution

```
library(glmnet)
set.seed(1)

y_col_idx <- 14  # `medv`
cvob1 = cv.glmnet(as.matrix(boston.train[, -c(y_col_idx)]), as.matrix(boston.train[,
    c(y_col_idx)]), nfolds = 5, alpha = 1, type.measure = "mse")
plot(cvob1)
```



The best lambda is:

```
print(cvob1$lambda.min)
```

```
## [1] 0.002172032
```

To print the fitted coefficients:

```
coef(cvob1, s = cvob1$lambda.min)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  0.023625259
## crim        -0.081492676
## zn           0.094199322
## indus        0.003673933
## chas         0.087338926
## nox         -0.174725136
## rm           0.312944488
## age         -0.010989639
## dis         -0.315732328
## rad          0.269191647
## tax         -0.207059619
## ptratio     -0.204123213
## black        0.102992828
## lstat       -0.428585719
```

d) Multiple choice (2P)

Say for *each* of them if it is true or false.

(i) When comparing computational speed between step-wise feature selection methods and Lasso for features selection, Lasso is much faster.
(ii) It is easier for ridge regression than Lasso to result in coefficients equal zero, namely due to the quadratic penalization term in ridge.
(iii) For the purpose of feature selection, both Ridge and Lasso are equally appropriate.
(iv) Elastic Net is a combination of Lasso and Ridge.

Solution True-False-False-True

Problem 2 (6P)

In this problem, a synthetic dataset is used. This dataset is purposefully created to show the difference between PCR and PLSR (PLS regression). We recommend you to explore the dataset before moving on.

```
library(MASS)
set.seed(1)

# load a synthetic dataset
id <- "1CWZYfrLOrFdrIZ6Hv73e3xxt0SFgU4Ph"  # google file ID
synthetic <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id))

# split into training and test sets
train.ind = sample(1:nrow(synthetic), 0.8 * nrow(synthetic))
synthetic.train = data.frame(synthetic[train.ind, ])
synthetic.test = data.frame(synthetic[-train.ind, ])

# show head(..)  Y: response variable; X: predictor variable
head(synthetic)
```

```
##             Y          X1          X2         X3         X4         X5
## 1 -1.43753239 -0.75905055 -0.69720326 -0.3016852 -0.7434697  0.8807558
## 2 -1.70972989 -0.28635632  0.04809182  0.5791725 -0.7446170  0.9935311
## 3  1.33931240  0.09574117 -0.89605758 -0.9636347  0.5554647 -0.5341800
## 4  0.20354906 -0.28702695  1.72952687  1.4289705 -0.1596993 -0.7161976
## 5 -0.09261896  0.02345825  0.51201583  0.1544345  0.4318039 -0.8674060
## 6  1.69952325  1.19231791 -0.98179754 -0.9567773 -0.6933918  0.4656891
##           X6         X7         X8          X9        X10
## 1 -0.8705750 -0.7448252 -0.4639697  0.62502272 -0.8149674
## 2  0.3532248 -0.5860332 -0.7964403  0.84868110 -0.1065119
## 3  0.4707434 -0.6588069 -0.7327518 -0.29429307  0.6588927
## 4 -0.7774007  0.2502145  0.5987052 -0.04428773  0.6247479
## 5 -0.9066908  0.8946086 -0.9700185  0.09082626  0.6102134
## 6 -0.7381794  0.8650175  0.4108119  0.75677429 -0.2281439
```

a) (2P)

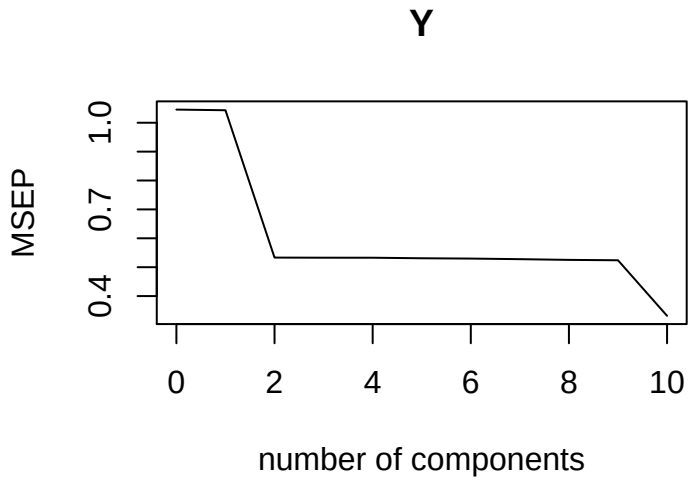
Fit PCR and PLSR on `synthetic.train` and show a graph of MSEP (mean squared error of prediction) with respect to the number of principal components for PCR and PLS, respectively.

Hint:

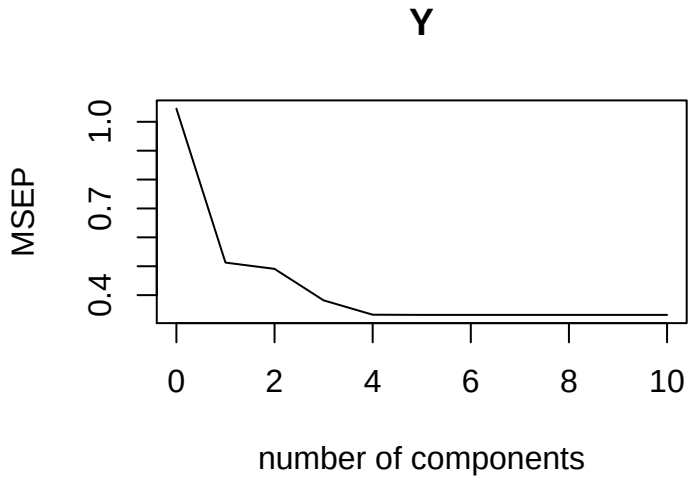
- use `validationplot(.., val.type="MSEP")` for the graph.

Solution

```
library(pls)

# fit PCR
pcr_model <- pcr(Y ~ ., data = synthetic.train)
validationplot(pcr_model, val.type = "MSEP")
```




```
# fit PLSR
plsr_model <- plsr(Y ~ ., data = synthetic.train)
validationplot(plsr_model, val.type = "MSEP")
```



b) (4P)

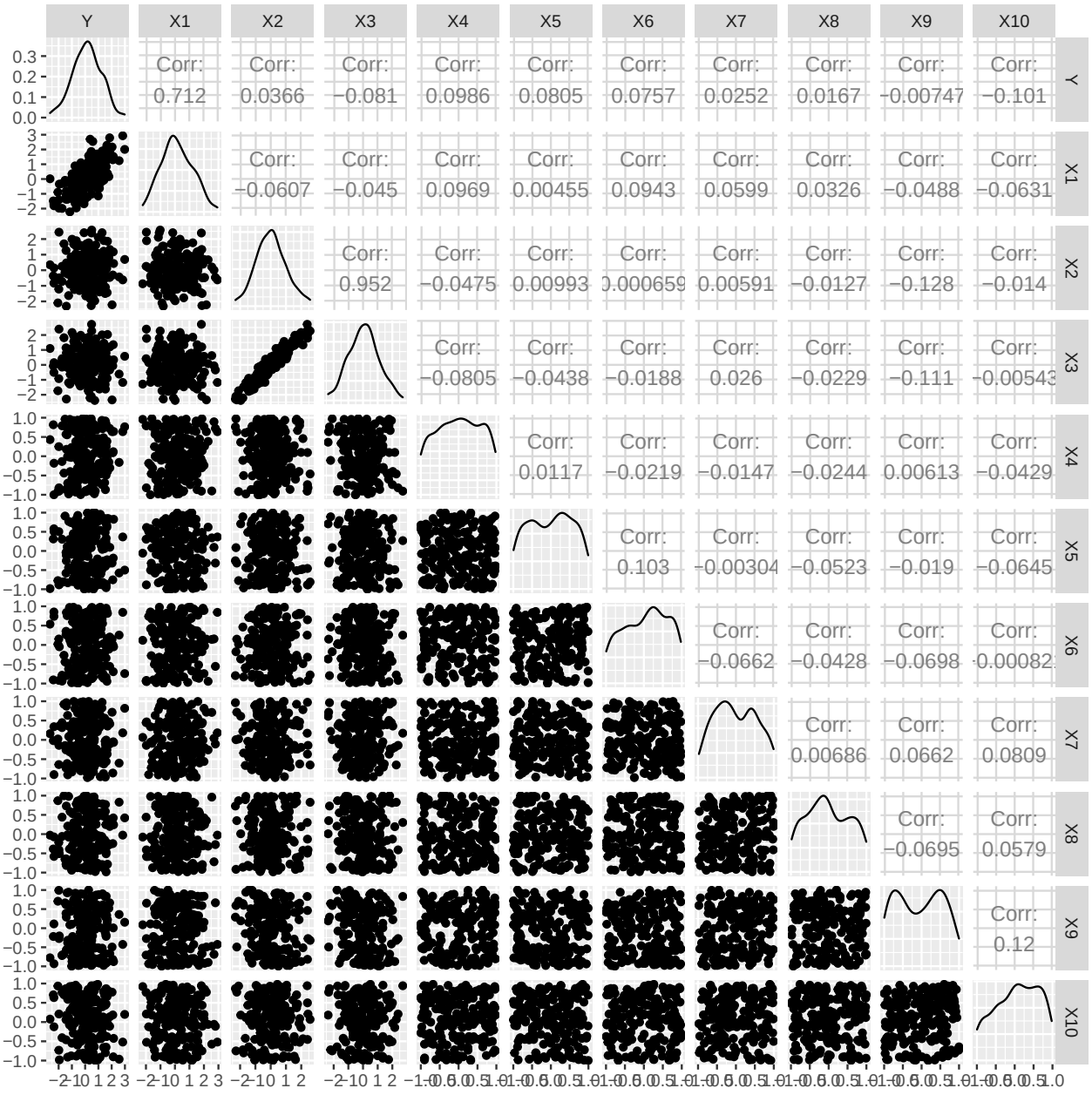
Given the two plots from a), explain what causes the difference between the results from the two methods.

NB!

- The answer must be associated with characteristics of PLS and PCR and be associated with the training set the model is fitted on.

Solution

```
library(GGally)
ggpairs(synthetic.test)
```



The dataset has the following characteristics (refer to the ggpairs above):

- `X1` and `Y` are correlated (corr: 0.7).
- `X2` and `X3` are strongly correlated (corr: 0.95). But they are not correlated with `Y`.

PCA in PCR finds its principal component axes that explains a dataset (not including a response variable) the most by finding axes with the highest variance. Yet, PCA does not guarantee that the axes found are necessarily useful for predicting the response. On the other hand, PLS finds axes that help explain both the response and the predictors.

In this example, PCR results in higher coefficient values for `X2` and `X3` and PSL results in a higher coefficient value for `X1` (refer to the code below).

In short, the answer should contain the followings:

- some sentence to do with PCR having its main axes direction on `X2` and `X3` which are useless for

predicting `Y` because PCA does not guarantee that the axes found are necessarily useful for predicting the response (i.e., unsupervised). (1P for explaining the characteristics of PCR, 1P for describing the problem with `X2` and `X3`.)
- some sentence to do with PSL having its main direction on `X1` which is useful for predicting `Y` because PLS finds the directions that help explain both the response and the predictors (i.e., supervised). (1P for explaining both characteristics of PLS, 1P for understanding that `X1` is useful while `X2` and `X3` are useless.)

```
print("PCR coefficients (n_comp=1):")
```

```
## [1] "PCR coefficients (n_comp=1):"
```

```
print(pcr_model$coefficients[, , 1])
```

```
##            X1            X2            X3            X4            X5
## -2.511968e-03  2.273185e-02  2.238810e-02  3.436408e-04 -7.185494e-06
##            X6            X7            X8            X9           X10
## -1.160326e-03  7.421283e-04  1.264102e-04 -7.171610e-04  3.961301e-04
```

```
print("PSLR coefficients (n_comp=1):")
```

```
## [1] "PSLR coefficients (n_comp=1):"
```

```
print(plsr_model$coefficients[, , 1])
```

```
##           X1           X2           X3           X4           X5           X6
##  0.693556576  0.131248709  0.036719392 -0.001101287 -0.011393744 -0.018971121
##           X7           X8           X9          X10
## -0.017875037 -0.009839922  0.017988765  0.001978983
```

Problem 3 (5P)

a) (2P) - Multiple choice

Say for *each* of them if it is true or false.

(i) For the polynomial regression (where polynomial functions of features are used as predictors), variance increases when including predictor with a high order of the power.
(ii) If the polynomial functions from (i) are replaced with step functions, then the regression model is too simple to be overfitted on a dataset even with multiple cutpoints.
(iii) The smoothing spline ensures smoothness of its function, g, by having a penalty term $\int g'(t)^2 dt$ in its loss.
(iv) The K-nearest neighbors regression (local regression) has a high bias when its parameter, k, is high.

Solution

(i) True
(ii) False: If there are a lot of cutpoints, it can be overfitted.
(iii) False: The penalty term is $\int g''(t)^2 dt$.
(iv) True: because the local regression is based on k-nearest neighbor algorithm.

b) (3P)

Fit an additive model on `boston.train` using the function `gam()` from package `gam` with the following conditions, and plot the resulting curves.

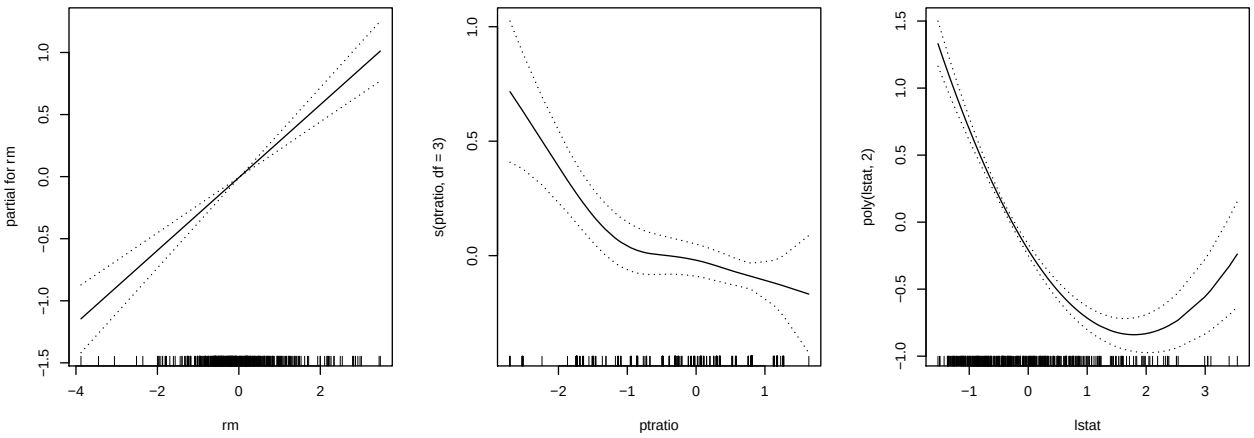
- response: `medv`; predictors: `rm`, `ptratio`, `lstat` (use these three predictors only).
- `rm` is a linear function
- `ptratio` is a smoothing spline with `df=3`.
- `lstat` is a polynomial of degree 2.

Solution

```
library(gam)

boston.train.subset <- subset(boston.train, select = c("medv", "rm", "ptratio", "lstat"))

# fit
fitgam <- gam(medv ~ rm + s(ptratio, df = 3) + poly(lstat, 2), data = boston.train.subset)

# plot
print(fitgam)
```

```
## Call:
## gam(formula = medv ~ rm + s(ptratio, df = 3) + poly(lstat, 2),
##     data = boston.train.subset)
##
## Degrees of Freedom: 403 total; 397.0002 Residual
## Residual Deviance: 111.9776
```

```
par(mfrow = c(2, 3))
plot(fitgam, se = T)
```



Problem 4 (11P)

a) (2P) - Multiple choice

Which of the following statements are true, which false?

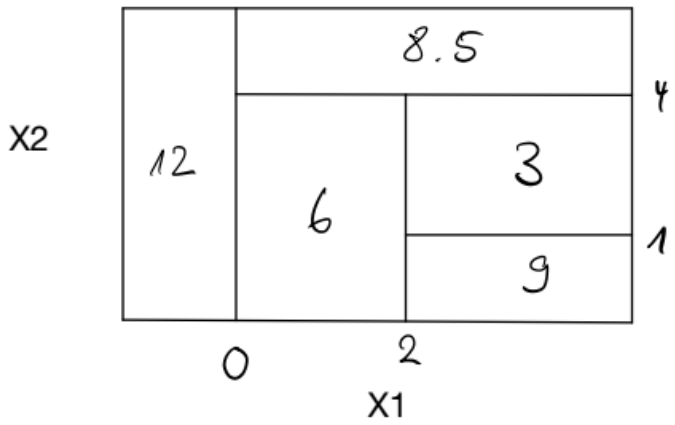
(i) A downside of simple regression trees is that they cannot handle interaction terms.
(ii) In boosting, the parameter d controls the number of splits allowed in each try. When $d = 2$, we allow for models with 2-way interactions.
(iii) The random forest approach improves bagging, because it reduces the variance of the predictor function by decorrelating the trees.
(iv) The number of trees B in boosting is a tuning parameter.

Solution

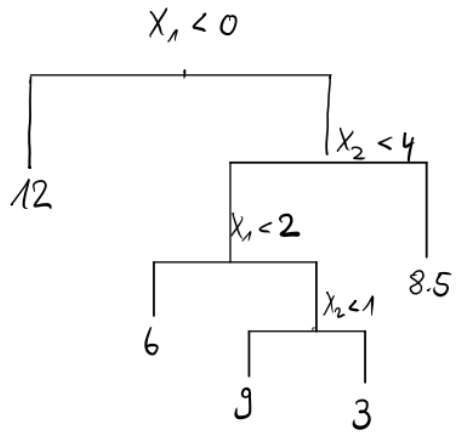
(i) FALSE
(ii) TRUE
(iii) TRUE
(iv) TRUE (in boosting it is, but not in random forests)

b) (2P)

Sketch the tree corresponding to the partition of the predictor space illustrated in the figure. The numbers inside the boxes are the mean of y within the regions (feel free to draw the tree by hand and upload a figure, or plot it with the computer).



Solution -1P for each error



c) (4P)

We are now again looking at the penguin dataset that we used in compulsory exercise 1 (the one that Basil, the cat, did not analyze very well). This time we are interested in a tree-based method to build a model that predicts the three different penguin species. In addition, we want to understand which factors are most relevant in discriminating the species.

We start again by loading and preparing the data set, similarly to exercise 3 in compulsory 1. We are also splitting the data into a training and a test set. You can run the following code without any changes:

```
library(tidyverse)
library(palmerpenguins)  # Contains the data set 'penguins'.
data(penguins)

names(penguins) <- c("species", "island", "billL", "billD", "flipperL", "mass", "sex",
    "year")

Penguins_reduced <- penguins %>% dplyr::mutate(mass = as.numeric(mass), flipperL = as.numeric(flipperL)
    year = as.numeric(year)) %>% drop_na()

# We do not want 'year' in the data (this will not help for future predictions)
Penguins_reduced <- Penguins_reduced[, -c(8)]

set.seed(4268)
# 70% of the sample size for training set
training_set_size <- floor(0.7 * nrow(Penguins_reduced))
train_ind <- sample(seq_len(nrow(Penguins_reduced)), size = training_set_size)
train <- Penguins_reduced[train_ind, ]
test <- Penguins_reduced[-train_ind, ]
```

Tasks:

(i) Start by generating a simple classification tree using the Gini index (using default `control` parameters) to find the splits and plot the resulting tree using the training data (1P).
(ii) Apply cost-complexity pruning using 10-fold CV, still using the training data (1P).
(iii) Find the optimal tree (1P) and report the misclassification error rate on the test set (1P).

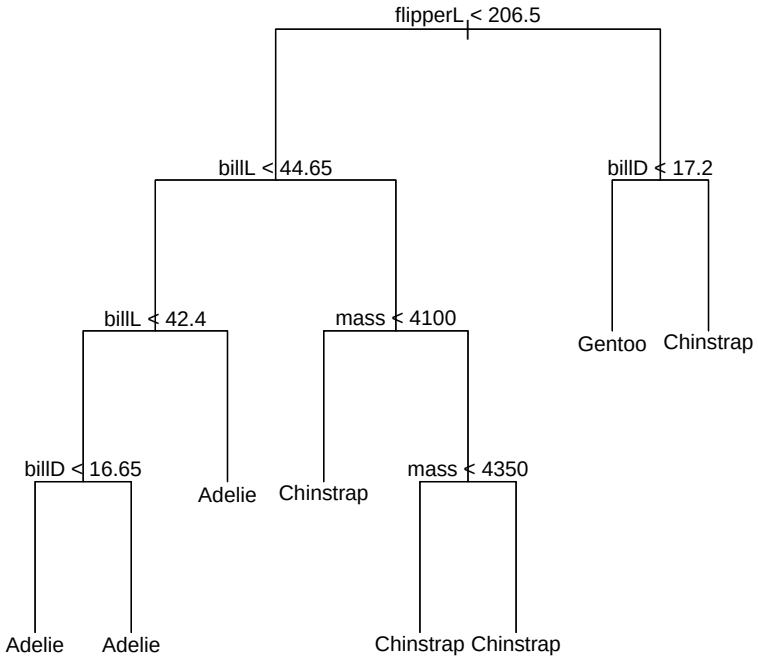
R-hints:
(i) When plotting the tree, use the argument `type="uniform"` in the `plot()` function.
(ii) Please use `set.seed(123)` before your run cross-validation, so that it is easier to reproduce your results.

Solution: (i)

```
library(tree)
tree.peng = tree(species ~ ., data = train, split = "gini")

summary(tree.peng)
```

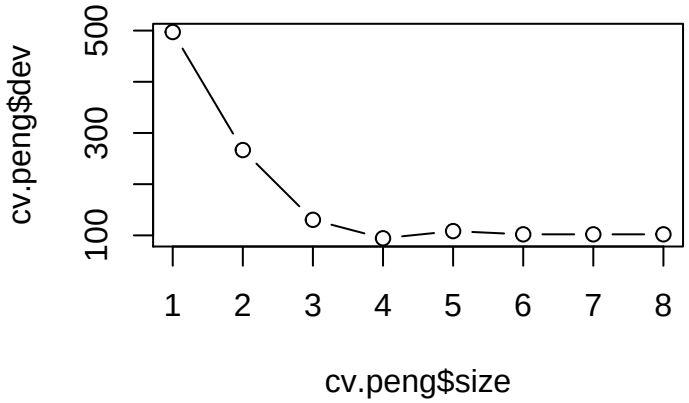
```
##
## Classification tree:
## tree(formula = species ~ ., data = train, split = "gini")
## Variables actually used in tree construction:
## [1] "flipperL" "billL"    "billD"    "mass"
## Number of terminal nodes:  8
## Residual mean deviance:  0.1869 = 42.06 / 225
## Misclassification error rate: 0.04292 = 10 / 233
```

```
plot(tree.peng, type = "uniform")
text(tree.peng)
```



(ii)

```
set.seed(123)
cv.peng = cv.tree(tree.peng, K = 10)
plot(cv.peng$size, cv.peng$dev, type = "b")
```



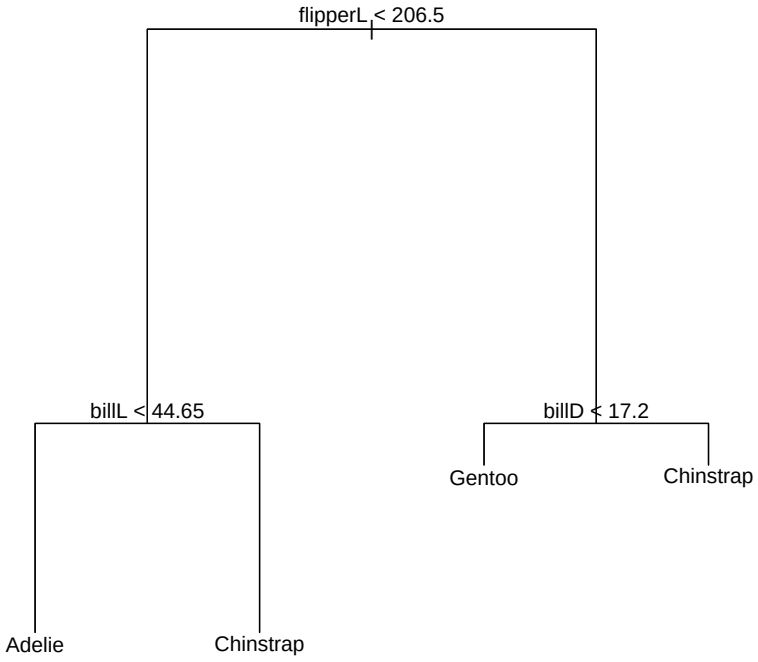
(iii) The size 4 is the tree with smallest error. Therefore:

```
prune.peng = prune.tree(tree.peng, best = 4)
```

It is possible that some students get slightly different results, e.g. when they don't use the same seed. This is ok, as long as the results are consistent.

Plotting the tree is possible, but not required

```
plot(prune.peng)
text(prune.peng, pretty = 0)
```



To get the misclassification error rate. Important: There are three species, so the code must be adapted accordingly, otherwise the point cannot be given.

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

```
##            Reference
## Prediction  Adelie Chinstrap Gentoo
##   Adelie        42         5      1
##   Chinstrap      0        15      0
##   Gentoo         0         0     37
```

```
1 - (sum(diag(confMat))/sum(confMat[1:3, 1:3]))
```

```
## [1] 0.06
```

d) (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 (1P). Explain your choice of the (tuning) parameters (1P). Which two variables are the most influential ones in the prediction of the penguin species (1P)?

Solution:

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.peng = randomForest(species ~ ., data = train, mtry = 2, ntree = 1000, importance = TRUE)
rf.peng
```

```
##
```

```
## Call:
##  randomForest(formula = species ~ ., data = train, mtry = 2, 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: 0.86%
## Confusion matrix:
##           Adelie Chinstrap Gentoo class.error
## Adelie       104         0      0  0.00000000
## Chinstrap      2        46      0  0.04166667
## Gentoo         0         0     81  0.00000000
```

```
round(1 - sum(diag(rf.peng$confusion))/sum(rf.peng$confusion[1:3, 1:3]), 4)
```

```
## [1] 0.0086
```

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.peng, test, type = "class")
(confMat <- confusionMatrix(t.pred.rf, test$species)$table)
```

```
##            Reference
## Prediction  Adelie Chinstrap Gentoo
##   Adelie        42         2      0
##   Chinstrap      0        18      0
##   Gentoo         0         0     38
```

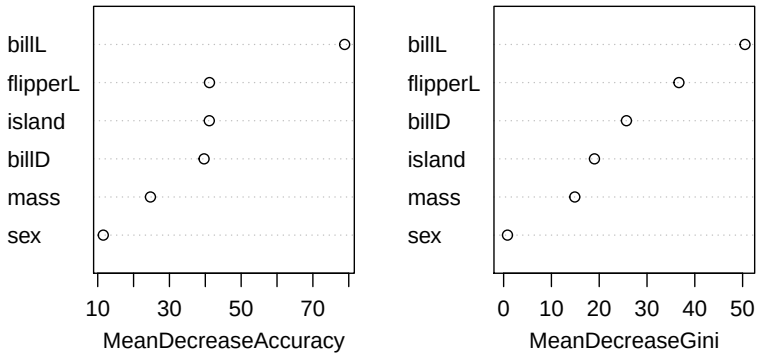
```
1 - (sum(diag(confMat))/sum(confMat[1:3, 1:3]))
```

```
## [1] 0.02
```

The misclassification error is now much smaller than for the simple tree-based method, which is quite impressive. To assess variable importance we can create a variable importance plot, or print the importance. According to those, bill length is clearly an important predictor, followed by flipper length. Also bill depth and maybe island seem relevant.

```
##              Adelie Chinstrap    Gentoo MeanDecreaseAccuracy MeanDecreaseGini
## island    0.4516805 44.895752 25.037473             41.11502       19.0053855
## billL    61.2900723 84.896978 18.870557             78.86027       50.5052448
## billD    26.4469061 20.241067 35.820223             39.65884       25.6964911
## flipperL 24.8535495 24.965057 36.864438             41.16388       36.6750243
## mass     11.6089050 19.117442 19.091525             24.70025       14.8890450
## sex       7.5838407  7.503686  6.934029             11.54579        0.8085654
```

rf.peng



Problem 5 (6P)

a) (2P) - Multiple choice

Imagine you have gene expression data for leukemia patients, with $p = 4387$ gene expressions 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) Logistic regression is the preferred method for this data set, because it gives nice interpretable parameter estimates.
(ii) 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).
(iii) When fitting a support vector classifier, we usually have to standardize the variables first.
(iv) By choosing a smaller budget parameter C we are making the model less biased, but introduce more variance.

Solution: FALSE - TRUE - TRUE - TRUE

(i) 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) (4P)

We are a last time looking at the penguin dataset, using again the same training and test sets as in Problem 4.

(i) (2P) Fit a support vector classifier (linear boundary) and a support vector machine (radial boundary) to find good functions that predict the three dolphin species. Use cross-validation to find a good cost parameter (for the linear boundary) and a good combination of cost *and* γ parameters (for the radial boundary), and report the error rates in the training set for both cases.
(ii) (1P) Report the confusion tables and misclassification error rates for the test set in both cases, using the best parameters you found in (i).
(iii) (1P) Which classifier do you prefer and why?

R-hints:
To run cross-validation over a grid of two tuning parameters, you can use the `tune()` function where `ranges` defines the grid points as follows:

```
tune(svm, formula, kernel = ..., ranges = list(cost = c(...), gamma = c(...)))
```

Solution:

(i) 1P each for corret linear and radial kernel optimization. The grid points can be chosen somewhat differently, and not everybody will find the best models. Note: it is possible to get down to 0 error on the training set for both linear and radial kernel. If students do not find such a solution, we deduct -0.5P (only once).

```
library(e1071)

CV_svm_linear <- tune(svm, species ~ ., data = train, kernel = "linear", ranges = list(cost = c(0.001,
    0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50)))
```

The following command gives a long list with an overview over the performance for the different values of cost (output not printed).

```
summary(CV_svm_linear)
```

A way to check the error rate on the training set is to extract the best model and then to count the number of wrong predictions:

```
bestmod_linear <- CV_svm_linear$best.model
sum(predict(bestmod_linear, train) != train$species)
```

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

For the radial boundary we have to use `kernel="radial"`, and cross-validate over a grid of cost and γ parameters.

```
set.seed(123)
CV_svm_radial <- tune(svm, species ~ ., data = train, kernel = "radial", ranges = list(cost = c(0.001,
    0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50), gamma = c(0.001, 0.01, 0.1, 1, 10)))
```

The following command gives a long list with an overview over the performance for the different sets of parameters (output not printed).

```
summary(CV_svm_radial)
```

Again, counting the number of wrong predictions for the best radial model:

```
bestmod_radial <- CV_svm_radial$best.model
sum(predict(bestmod_radial, train) != train$species)
```

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

(ii) Prediction error for linear and radial kernels:

```
ypred_linear = predict(bestmod_linear, test)
(tt1 <- table(predict = ypred_linear, truth = test$species))
```

```
##            truth
## predict     Adelie Chinstrap Gentoo
##   Adelie        42         0      0
##   Chinstrap      0        20      0
##   Gentoo         0         0     38
```

```
1 - sum(diag(tt1))/sum(tt1)
```

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

```
ypred_radial = predict(bestmod_radial, test)
(tt2 <- table(predict = ypred_radial, truth = test$species))
```

```
##            truth
## predict     Adelie Chinstrap Gentoo
##   Adelie        41         1      0
##   Chinstrap      1        19      0
##   Gentoo         0         0     38
```

```
1 - sum(diag(tt2))/sum(tt2)
```

```
## [1] 0.02
```

Interestingly, the error on the test set is zero for the linear kernel.

(iii) There are two reasons why we would use he SVM with linear kernel (0.5P for choosing linear, 0.5P when at least one of the arguments is there):

- it seems to work best (no error on the test set).
- it is simpler than the radial kernel (fewer parameters, thus more robust).

Problem 6 (12P)

In the following code, we're importing a word-happiness-report-2021 dataset. This dataset has a response of happiness score and several predictors such as GDP (gross domestic product), social support, life expectancy, freedom, generosity, and corruption level of 149 countries. One of the typical uses of this dataset is analysis of important variables that largely contribute to the happiness level by using a method such as PCA.

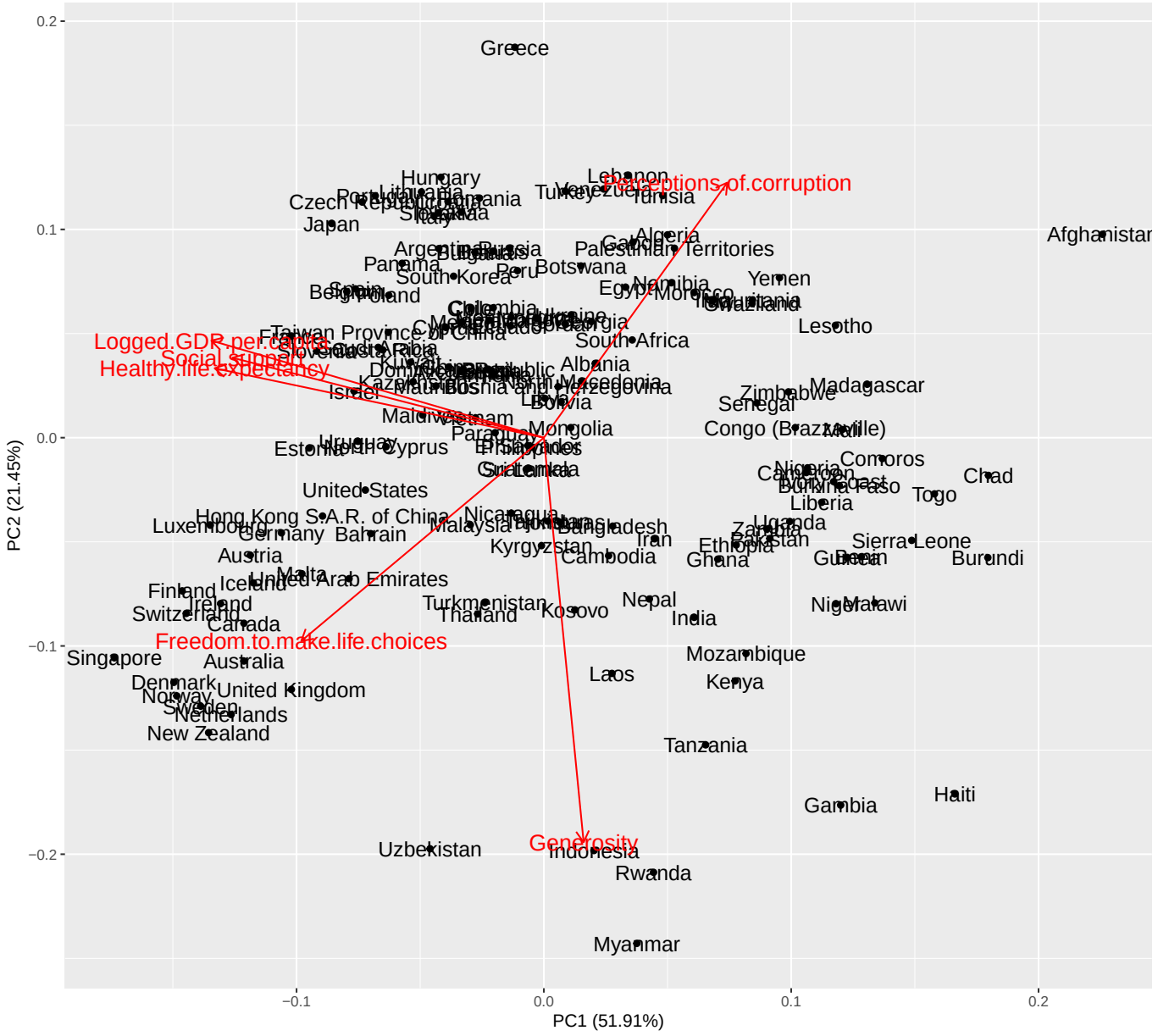
```
# load a synthetic dataset
id <- "1NJ1SuUBebl5P8rMSIwm_n3S8a7K43yP4" # google file ID
happiness <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id), fileEncoding="UTF

colnames(happiness)
```

```
##  [1] "Country.name"
##  [2] "Regional.indicator"
##  [3] "Ladder.score"
##  [4] "Standard.error.of.ladder.score"
##  [5] "upperwhisker"
##  [6] "lowerwhisker"
##  [7] "Logged.GDP.per.capita"
##  [8] "Social.support"
##  [9] "Healthy.life.expectancy"
## [10] "Freedom.to.make.life.choices"
## [11] "Generosity"
## [12] "Perceptions.of.corruption"
## [13] "Ladder.score.in.Dystopia"
## [14] "Explained.by..Log.GDP.per.capita"
## [15] "Explained.by..Social.support"
## [16] "Explained.by..Healthy.life.expectancy"
## [17] "Explained.by..Freedom.to.make.life.choices"
## [18] "Explained.by..Generosity"
## [19] "Explained.by..Perceptions.of.corruption"
## [20] "Dystopia...residual"
```

```r
cols = c('Country.name',
         'Ladder.score',  # happiness score
         'Logged.GDP.per.capita',
         'Social.support',
         'Healthy.life.expectancy',
         'Freedom.to.make.life.choices',
         'Generosity',  # how generous people are
         'Perceptions.of.corruption')

# We continue with a subset of 8 columns:
happiness = subset(happiness, select = cols)
rownames(happiness) <- happiness[, c(1)]

# And we creat an X and a Y matrix
happiness.X = happiness[, -c(1, 2)]
happiness.Y = happiness[, c(1, 2)]
happiness.XY = happiness[, -c(1)]

# scale
happiness.X = data.frame(scale(happiness.X))


str(happiness)
```

```
## 'data.frame':    149 obs. of  8 variables:
##  $ Country.name                : Factor w/ 149 levels "Afghanistan",..: 41 34 129 55 97 104 128 79 98
##  $ Ladder.score                : num  7.84 7.62 7.57 7.55 7.46 ...
##  $ Logged.GDP.per.capita       : num  10.8 10.9 11.1 10.9 10.9 ...
##  $ Social.support              : num  0.954 0.954 0.942 0.983 0.942 0.954 0.934 0.908 0.948 0.934 ..
##  $ Healthy.life.expectancy     : num  72 72.7 74.4 73 72.4 73.3 72.7 72.6 73.4 73.3 ...
##  $ Freedom.to.make.life.choices: num  0.949 0.946 0.919 0.955 0.913 0.96 0.945 0.907 0.929 0.908 ...
##  $ Generosity                  : num  -0.098 0.03 0.025 0.16 0.175 0.093 0.086 -0.034 0.134 0.042 ..
##  $ Perceptions.of.corruption   : num  0.186 0.179 0.292 0.673 0.338 0.27 0.237 0.386 0.242 0.481 ...
```

```r
library(ggfortify)
pca_mat = prcomp(happiness.X, center = T, scale = T)

# Score and loadings plot:
autoplot(pca_mat, data = happiness.X, colour = "Black", loadings = TRUE, loadings.colour = "red",
    loadings.label = TRUE, loadings.label.size = 5, label = T, label.size = 4.5)
```



a) (3P)

(i) Look at the loading directions in the plot above. Describe two characteristics that you observe in the relations between the variables. (2P) (NB: Describe the variable relations that are obvious. If you don't see much relation between certain variables, you don't have to describe that relation.)

(ii) The scores in the above plot can also be used for doing outlier/anomaly detection. Which country can be considered to an outlier among the followings: {Norway, Vietnam, South Korea, Afghanistan, India}? (1P)

Solution

(i) The answer must contain:

- `Perception of corruption` has an opposite characteristics to that of `Freedom to make life choices` given that the vector directions are opposite to each other. (1P)

- `Healthy life expectancy`, `Logged GDP per capita`, and `Social support` have similar characteristics given that they are gathered in the same neighborhood. (1P)

(ii) Afghanistan

b) (4P)

Here, we're going to find out which variables are important by principal component analysis (PCA) and partial least squares regression (PLSR).

Note that we can naturally assume the followings:

- PCA will find out important variables w.r.t explainability of the dataset of the predictors.
- PLSR can find out important variables w.r.t the response in the model, that is, happiness (= `Ladder.score`).

(i) Make a graphical description of the absolute values of the first principal component (= `PC1`) by PCA. You can use a bar plot, or any other graphical description of your choice (see R-hints below). (1P)

(ii) Fit PLSR on `happiness.XY` with a response of `Ladder.score` (= happiness score) and all the remaining variables in that dataset as predictors. (1P)

(iii) Plot a bar graph of the absolute values of the first principal component for `X` (= predictors of `happiness.XY`) by PLSR. Use the same type of plot as in (i) in order to compare. (1P)

(iv) What are the three most important predictor to predict the happiness score based on the PLSR bar graph from (iii)? (1P)

R-hints:

(i)

- Use `data.frame(pca_mat$rotation)$PC1`.
- The x-axis should show the variable names.
- The y-axis should show the `abs(PC1)`. (Note that `abs(PC)` denotes the feature/variable importance.)

(ii)

- Use `plsr_model <- plsr(..., scale=T)`

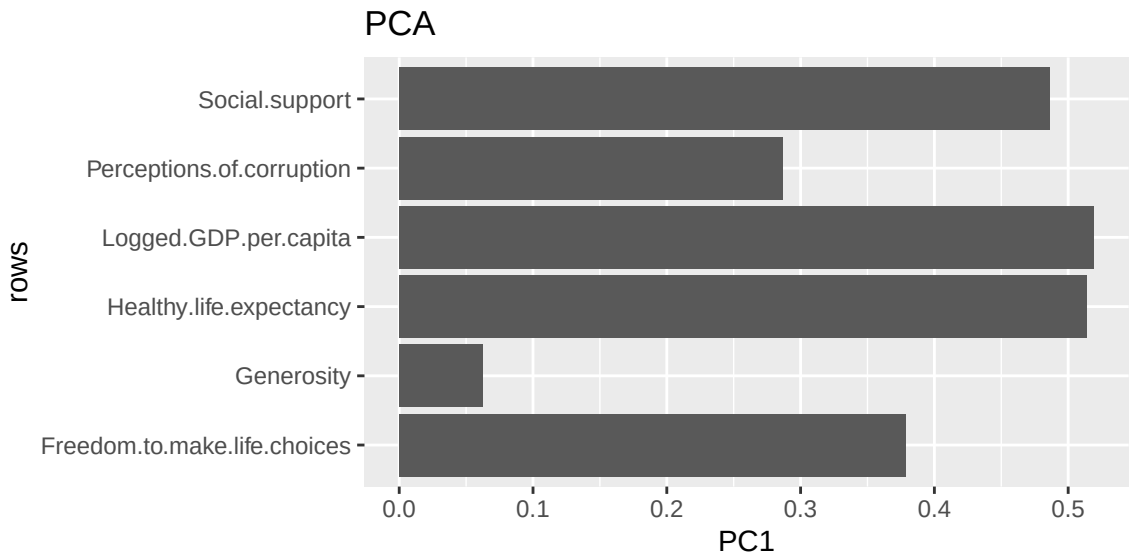
(iii)

- Use `plsr_model$loadings[,c('Comp 1')]`

Solution

(i)

```
library(ggplot2)

df <- abs(data.frame(pca_mat$rotation))
df$rows <- rownames(df)

p <- ggplot(data = df, aes(x = rows, y = PC1)) + geom_bar(stat = "identity") + ggtitle("PCA")
p + coord_flip()
```

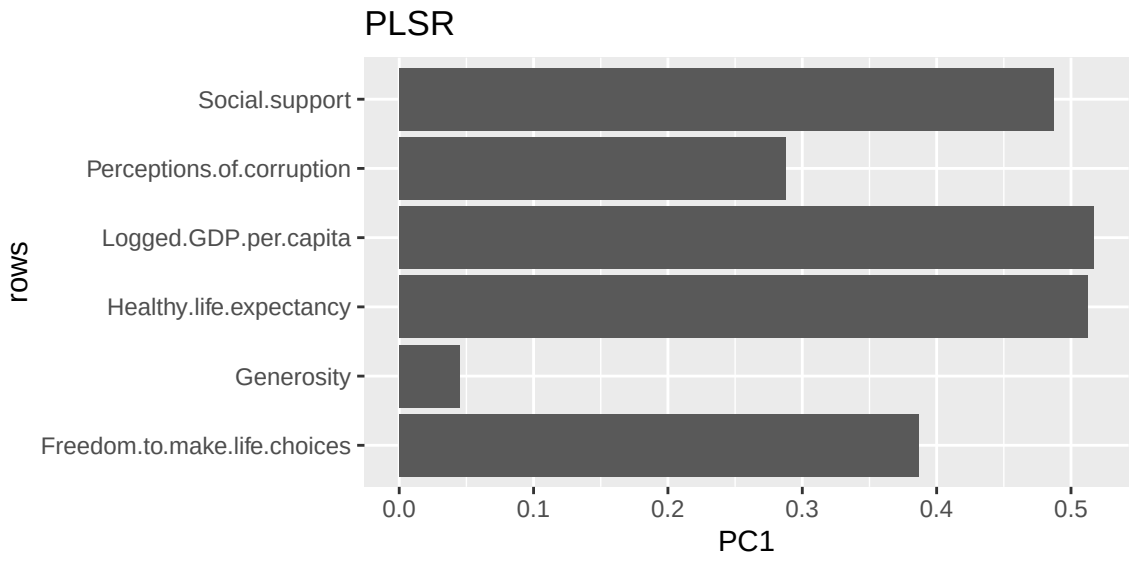


(ii)

```
library(pls)

# fit PLSR
plsr_model <- plsr(Ladder.score ~ ., data = happiness.XY, scale = T)
# validationplot(plsr_model, val.type='MSEP')
```

(iii)

```
df <- plsr_model$loadings[, c("Comp 1", "Comp 2")]
df <- data.frame(df)
colnames(df) <- c("PC1", "PC2")
df <- abs(df)
df$rows <- rownames(df)

# plot
p <- ggplot(data = df, aes(x = rows, y = PC1)) + geom_bar(stat = "identity") + ggtitle("PLSR")
p + coord_flip()
```



(iv) GDP, social support and health

c) (2P) - Multiple choice

Say for *each* of them if it is true or false.

(i) K-means is optimizing clusters such that the within-cluster variance becomes large.
(ii) No matter how many times you run K-means clustering, its cluster centroids will always end up in the same locations.
(iii) Strong correlation between predictors allows PCR to be more effective for predicting a response when prediction is made based on the first two principal components.
(iv) We can do outlier/anomaly detection with PCA.

Solution

(i) False; K-means makes the within-cluster variance small.
(ii) False; not always due to the randomness in assigning the initial cluster centroids.
(iii) False; strong correlation between the predictors makes it worse for predicting the response for PCR.
(iv) True

d) (3P)

(i) We are now doing a K-means clusterization. Run the k-means clustering on `happiness.X` given the following condition and visualize the clusters using the code below. (This question is given to let you explore how countries are clustered together based on `happiness.X`. There are multiple answers for `K`. You can use whatever `K` value that satisfies the condition.) (1P)

Condition:

- Norway, Denmark, Sweden, Finland should be in the same cluster, while United States is in a different cluster.

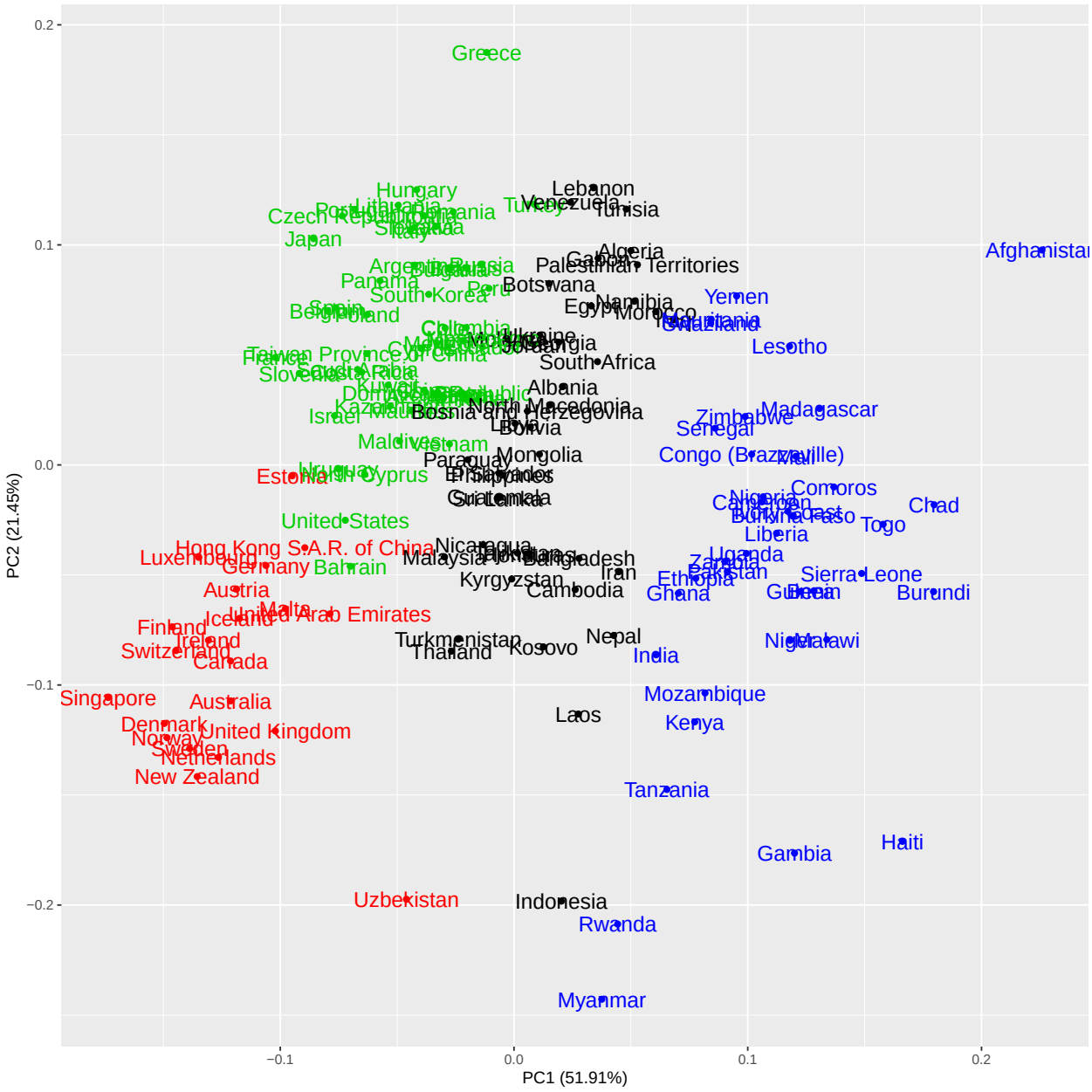
```
K = -1  # your choice
km.out = kmeans(happiness.X, K)

autoplot(pca_mat, data = happiness.X, colour = km.out$cluster, label = T, label.size = 5,
    loadings = F, loadings.colour = "blue", loadings.label = F, loadings.label.size = 3)
```

(ii) Give your interpretation of the clusters w.r.t the happiness score (= `Ladder.score`). One point is given per correct interpretation aspect, -1P for a wrong interpretation. (2P)

Solution

(i)

```
K = 4  # your choice
km.out = kmeans(happiness.X, K)

autoplot(pca_mat, data = happiness.X, colour = km.out$cluster, label = T, label.size = 5,
    loadings = F, loadings.colour = "blue", loadings.label = F, loadings.label.size = 3)
```



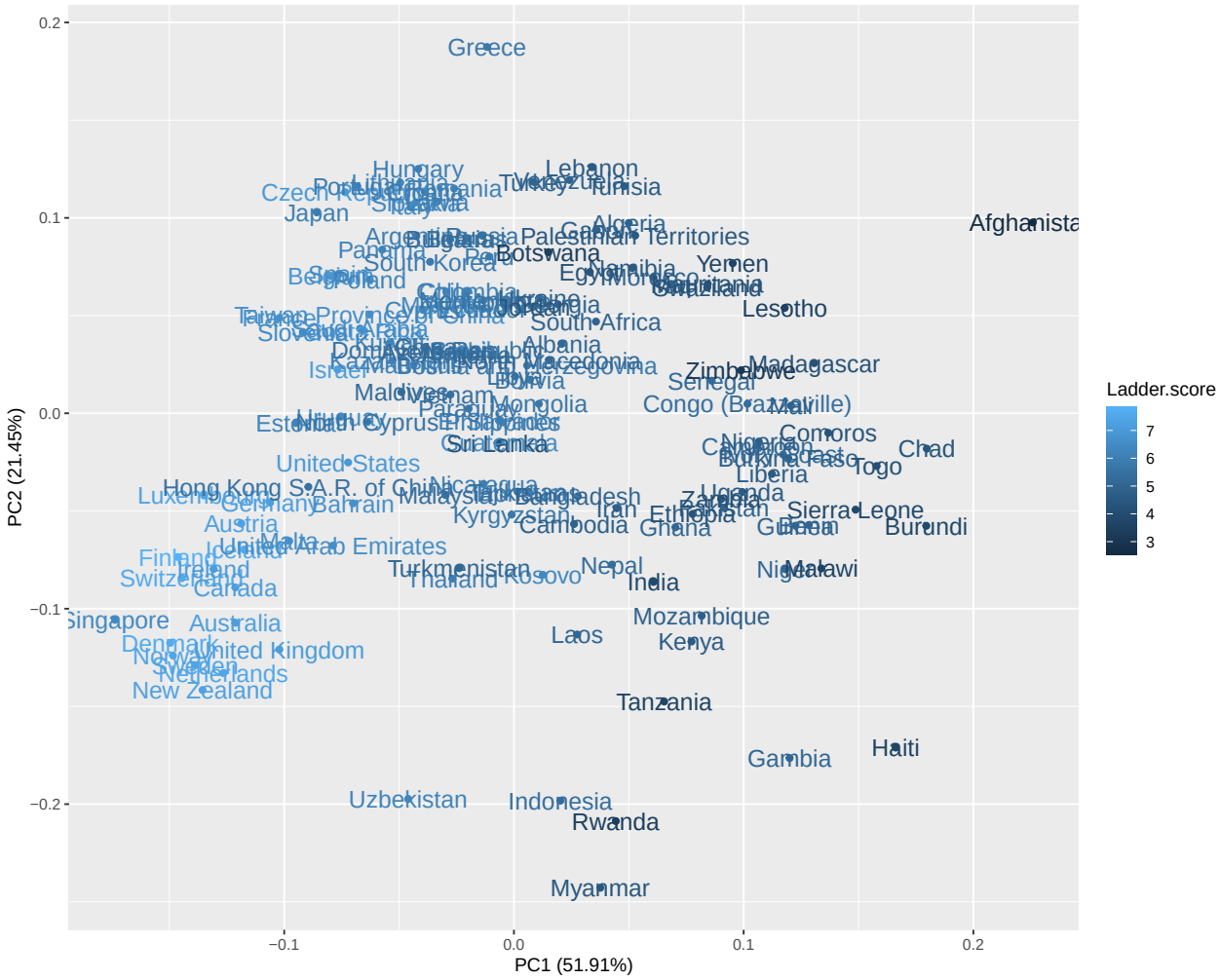
(ii)

the interpretation may include:

- Within each cluster, countries with a similar level of happiness is clustered.
- Norway, Denmark, Sweden, and Finland belong to the happiest countries.
- Other countries such as Australia, New Zeal and, Sigapore are in the same cluster as Norway, Denmark, Sweden, and Finland and they are happy as well.
- Unhappy countries are clustered together on the right side.
- etc.

Note: One point is given per correct interpretation aspect, -1P for a wrong interpretation.

```
# score plot with happiness score
autoplot(pca_mat, data = happiness, colour = "Ladder.score", label = T, label.size = 5,
    loadings = F, loadings.colour = "blue", loadings.label = F, loadings.label.size = 3)
```


Greece
Hungary
Lebanon
Tunisia
Japan
Algeria
Afghanistan
Palestinian Territories
Botswana
South Korea
Yemen
Lesotho
South Africa
Albania
Madagascar
Zimbabwe
Senegal
Mongolia
Comoros
Chad
Togo
Liberia
United States
Hong Kong S.A.R. of China
Bahrain
Kyrgyzstan
Cambodia
Sierra Leone
Burundi
Austria
United Arab Emirates
Finland
Ireland
Switzerland
Canada
Turkmenistan
Nepal
Thailand
India
Malawi
Singapore
Australia
Mozambique
Laos
Kenya
Denmark
United Kingdom
Netherlands
New Zealand
Tanzania
Haiti
Gambia
Uzbekistan
Indonesia
Rwanda
Myanmar
PC2 (21.45%)
PC1 (51.91%)
0.2
0.1
0.0
−0.1
−0.2
−0.1
0.0
0.1
0.2
Ladder.score
7
6
5
4
3