HDS Exercise set 1.

YOUR NAME (STUDENT NUMBER)

Return by **10.15** o'clock on Tue **5.11.2019** to the Moodle area of the course. You can use 'HDS_ex1.Rmd' file as a template for your answers, in which case each question will also be shown there before your own solution. Return the final file in pdf format with name "HDS1_yourname.pdf" where "yourname" is, unsurprisingly, your name.

Problem 1.

Read from IRSL p.110-117 about basic linear model analysis of `Boston` dataset that is part of the `MASS` package. You can get `Boston` available by command `library(MASS)` in R.

(a) It is always good to start with looking carefully at your data before running into analyses. Use `?Boston` to read what each variable means (no need to output this). Apply command `str()` to see n and p and which variables there are and which data types they are. Check also for any missing values (NA) using command `anyNA`. What are your n and p and is there any missing values? Do you have any non numerical data fields (that could be problematic in linear regression)?

(b) Show how numerical variables are correlated by using package `corrplot` as in notes "0.2. Linear regression in practice" to visualize a pairwise correlation matrix of the variables. (Note that you may need to install `corrplot` first to your system using `install.packages()`.) Are there any highly (say $|r| > 0.9$) correlated pairs of variables? If you want to predict median value of houses (`medv`), which other variables would expect to be most predictive of `medv` based on the correlations?

(c) We want to predict median value of houses (`medv`). Make 12 separate plots where in each y-axis is `medv` and x-axis goes through all other variables except `chas` (that is binary variable). For each plot, add also linear regression line of the model `medv~x`, where x is the variable on x-axis. In the title of each plot add the value of adjusted R2. (Title is set by `main=` parameter in plot command, and adjusted R2 can be found from output object of `summary(lm.fit)`, where `lm.fit` is output of linear model `lm(medv~x)`.) For example, you can arrange these 12 panels in a plotting area by calling `par(mfrow=c(2,3))` at the beginning of code and do a 12-step for-loop over the chosen variables. This would fill two 2x3 areas with your 12 plots. You can also use a 4x3 area but then you need to adjust `fig.height` parameter at the beginning of your R block to make the plotting area larger than it is by default. Don't copy paste the command 12 times with different variable name! Don't output the linear model outputs, but only add the lines and adjusted R2 values to the plots. Don't hard code any R2 values in the code, but extract them directly to the plots from the `summary` command output. Which predictors are most predictive of `medv`? Is there any predictors that do not seem to be predictive at all (based on the plots and R2, no need to do formal test here)? How do the R2 values reported here relate to the pairwise correlation values with `medv` from part (b)?

(d) Fit a linear model `medv ~ lstat + rm` and output its `summary()`. Explain in everyday language how each of the predictors is predicting the median house value and what is uncertainty of the estimates (in terms of 95% CIs). Use `predict` function to make a prediction of median value of a house in an area where avg number of rooms is 5 and where percentage of the lower status population is 7, and compare it to the median house value when `lstat` is 17 (and `rm` is still 5).

Answer 1.

(Your solution goes here.)

Problem 2.

Let's continue with the linear model `medv ~ lstat + rm` in `Boston` dataset.

(a) Make diagnostic plots by calling `plot()` on the lm-object of model `medv ~ lstat + rm`. Arrange the plots in 2x2 area and adjust size of the plotting area appropriately at the R block definition (e.g. `{r, fig.height = 6, fig.width = 6}` seems to work ok). Do you see possible indications of problems with model fit in the two residuals vs. fitted plots show (2 plots at the left column), or in the QQ-plot (upper right), or in the residuals vs. leverage plot (lower right)? Some hints for interpretation these plots: https://data.library.virginia.edu/diagnostic-plots/.

(b) Add quadratic terms for both `lstat` and `rm` in the model and look at `summary()`. Are the added quadratic terms useful for these variables? Plot again the 4 diagonstic plots. Do they look better than in (a)? We see 3 outlier observations with large |residuals| named in the diagnostic plots. Which one of these is having a concerning effect on the total model fit as determined by residuals vs. leverage plot?

(c) Try to figure out whether this one observation is outlier in `medv` because of its `lstat` value or because of its `rm` value, as follows. Do separate linear regressions (with quadratic terms included) of `medv` on `rm` and `medv` on `lstat` and plot only the 4th diagnostic plot of both models. (For a single predictor model, the 4th plot shows only the Cook's distance, while for multiple predictors it shows the residuals vs. leverage plot where Cook's distance bounds 0.5 and 1 are shown in red.) Does the outlier observation from (b) have a large Cook's distance for either/both/none of these two predictors? (You can read more about Cook's distance from Wikipedia, for example.)

Answer 2.

(Your solution goes here.)

Problem 3.

Generate a set of $p = 100$ P-values using command

`pval = c(runif(80), rbeta(20, 1, 100))`.

Compute manually in R (i.e. wihout any ready-made function) Benjamini-Hochberg adjusted versions of these P-values. You should get the same values as given by `p.adjust(pval, method = "BH")`. Compare the two explicitly.

Problem 4.

Let's consider a multiple testing problem, where we do $p = 10^4$ tests, each time using a separate linear regression model on $n = 1000$ individuals. (E.g. a genome-wide association study of lipid levels on 10^4 genetic variants using a sample of 1000 individuals.)

Part (a).

Repeat the following procedure $R = 1000$ times:

Generate a realistic set of P-values when all p tested null hypotheses are true (i.e. there are no non-zero effects). (Do not do actual linear regression, but simply generate the P-values. See the beginning of Lecture 2 "False discovery rate" for a way to generate P-values from linear regression.) For that set of P-values compute the proportion of P-values that are labelled significant ("discoveries") by (unadjusted) significance level $\alpha = 0.05$, by Bonferroni-adjusted significance level $\alpha_B = \alpha/p$ and by Benjamini-Hochberg method at FDR level α.

For each repetition collect the values of those three proportions. At the end, draw three histograms over R repetitions, one for each method.

Are the results what you expected? BH method should be much better able to label true positives as discoveries than Bonferroni method. Do the two differ much here, when there are no true positives?

Part (b).

Do the same as in part (a) except that change first $m = 100$ P-values out of all $p = 10^4$, at each of the $R = 1000$ replications, to come from the model where the true effect size parameter $\beta = 0.1$ and residual variance $\sigma^2 = 1$ and predictor's variance $v_x = 1$. (Again, see beginning of Lecture 2 for P-value generation.) Collect from each replication, for each method, the total number of discoveries and the number of false discoveries.

Make 6 histograms in a 2x3 area. First row has one histogram per method showing the proportion of discoveries across the replications. Second row has one histogram per method showing false discovery proportion FDP (the proportion of false discoveries out of all discoveries).

Do you see bigger differences between Bonferroni and BH now than in part (a)? What does the differences tell about the methods?

Problem 5.

When we have a large number p of predictors x_j collected to $n \times p$ matrix $\boldsymbol{X}$ that we want to use to predict outcome y, the first step is often to fit p simple linear models of type $y \sim \mu_j + x_j\beta_j$. In lecture 1 we did this by applying `lm()` on each column of $\boldsymbol{X}$ separately:

```
#by mean-centering y and each x, we can ignore intercept terms (since they are 0, see Lecture 0)
X = as.matrix( scale(X, scale = F) ) #mean-centers columns of X to have mean 0
y = as.vector( scale(y, scale = F) )
#apply lm to each column of X separately and without intercept (see Lecture 0.)
lm.res = apply(X, 2 , function(x) summary(lm(y ~ -1 + x))$coeff[1,])
```

In this exercise, we do the same much more efficiently by using direct matrix-vector operations.

Part (a).

We know that the formula for the regression coefficient for mean-centered model is

$$\widehat{\beta}_j = \frac{\boldsymbol{x}_j^T \boldsymbol{y}}{\boldsymbol{x}_j^T \boldsymbol{x}_j}.$$

How can you efficiently compute the vector $\widehat{\boldsymbol{\beta}} = (\widehat{\beta_1}, \ldots, \widehat{\beta_p})^T$ using only matrix-matrix and matrix-vector operations in R? (No for-loops, no apply-functions.)

Demonstrate your method on a data set generated as

```
n = 100
p = 1000
X = matrix(rnorm(n*p, 0, 1), nrow = n)
y = rnorm(n)
```

Compare your β-estimates to the ones given by `apply()` as above to see that they agree (up to precision 1e-16).

Part (b).

Standard errors from the mean-centered simple linear model are of the form

$$s_j = \sqrt{\frac{\sigma_j^2}{\boldsymbol{x}_j^T \boldsymbol{x}_j}},$$

where σ_j^2 is the error variance of the simple linear regression with x_j as the predictor. How can you estimate these efficiently for all j using $\boldsymbol{X}$, $\boldsymbol{y}$ and $\widehat{\boldsymbol{\beta}}$? Again, verify your results by comparing to the standard errors given by `apply()` function.

(Extra: Once everything works, you can test by increasing p from 1,000 to 30,000 that the matrix operations still produce results instantaneously but with `apply()` function it starts to take annoyingly long.)