Series 6

1. Consider the following linear regression model

$$Y_i = \beta_1 + \beta_2 x_{i2} + \beta_3 x_{i3} + \epsilon_i,\ i = 1, \ldots, 25,\ \beta_1 = 1, \beta_2 = -2, \beta_3 = 3, \tag{1}$$

where the pairs x_{i2}, x_{i3} lie on a $\{1, \ldots, 5\} \times \{1, \ldots, 5\}$-grid, i.e.,

```
x2 <- rep(1:5, 5)
x3 <- rep(1:5, each = 5)
```

In this exercise, we will simulate datasets from this model using different error distributions and perform linear regression. Confidence intervals for the regression parameters will then be estimated using classical theory and bootstrapping.

For convenience, we will use an auxiliary function that gives the regression coefficients when a given permutation of the data is used (such permutation is defined by `ind`, which is itself a vector of indexes):

```
> lmcoefs1 <- function(data, ind) coef(lm(y ~ x2 + x3, data = data[ind, ]))
```

This is easy to understand but somewhat slow when called 1000's of times. Hence, we use the equivalent (but 10× faster) version, `lmcoefs`.

```
> lmcoefs <- function(data, ind) {
    d <- as.matrix(data)[ind,,drop=FALSE]
    coef(lm.fit(cbind(1, d[,c("x2","x3")]), d[,"y"]))
 }
```

a) Implement a bootstrap routine in R that takes a data frame with columns `y`, `x2`, and `x3` as input and that returns three confidence intervals, one for each regression parameter β_j, $j = 1, 2, 3$. **R-hints:** Complete the following skeleton:

```
obst.est <- function(data, B)
  {
    len <- nrow(???)

    #Create matrix of indexes of dimension (B x len) where
     each colum corresponds to a bootstrap sample
    ind <- replicate(???, sample(???, ???, replace = TRUE))

    #Bootstrap estimations of regression coefficients using
     the B bootstrap samples
    apply(???, ???, lmcoefs, data = data)
  }
obst.ci <- function(bst.pars, data, alpha)
  {
    ## Estimate regression parameters without using bootstrap
    reg.pars <- lmcoefs(???, ???)

    ## Calculate empirical quantiles of the bootstrap distribution
    qt <- apply(bst.pars, ???, quantile,
                probs = c(???,???), names = FALSE)
```

```
    ## Return vector of bootstrap confidence intervals
    ## (cf. Formula (5.5) in the lecture notes)
    ??? - t(qt)
}
```

`obst.est` gives a matrix of 3 × `B` with the bootstrapped estimated coefficients. Its argument `data` contains the original data and `B` is the number of bootstrap samples.
`obst.ci` gives a matrix of 3 × 2 with the bootstrap confidence intervals for β_1, β_2, and β_3 (first column corresponds to lower bounds and second column to the upper bounds). Its argument `bst.pars` should be a matrix like the one given by the function `obst.est`, `data` contains the original data, and `alpha` is the significance level (e.g. 10 %).

b) Simulate 100 datasets[1] from model (1) computing each time classical theory 0.90-confidence intervals and bootstrap[2] 0.90-confidence intervals for the three regression parameters.
For the simulations, use three different types of error distributions (resulting in 300 simulated datasets, all in all):

1. $\epsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$,
2. $\epsilon_i \overset{\text{i.i.d.}}{\sim} t_3$ (R function `rt`) (heavier tails than Gaussian, symmetric, centered),
3. $\epsilon_i = e_i - 1/3$, where $e_i \overset{\text{i.i.d.}}{\sim} \text{Exp}(3)$ (R function `rexp`) (asymmetric, centered).

How often do the confidence intervals include the true values (estimated coverage rate)?

R-hints: To make your results reproducible, use `set.seed(84)` at the beginning of your simulation experiment.
Classical confidence intervals for output objects of `lm` can be computed using `confint`.
Use $B = 1000$ bootstrap samples. Beware that bootstrapping is computationally quite expensive. In order to avoid long waiting times, first develop and test your code with few simulations and bootstrap samples (say, 10 each), and augment these numbers only when your code works.

c) Repeat the bootstrap calculation of the confidence intervals by using the function `boot.ci` from package `boot`.
R-hints: The function `boot` from package `boot` allows automatic bootstrapping of statistics on given data. To apply this function, you have to write your own R-function which returns the regression coefficients and has arguments `dat` and `ind`. `dat` is a data frame containing the variables `y`, `x2` and `x3`, and `ind` is a vector of indices (see help page, parameter `statistic` and see provided function `lmcoefs`).
Then use the `boot` function:

```
bst.sample <- boot(data=dat, statistic=lmcoefs, R=B)
```

Bootstrap confidence intervals are then computed by `boot.ci`, which may look as follows:

```
bst.ci <- boot.ci(bst.sample, conf=1-alpha, type="basic", index=k)
```

`bst.sample` is the output of `boot`, `index` should be 1 for the intercept parameter, 2 and 3 for the regression parameters (if computed as in `lmcoefs` above). The interval bounds come as values `bst.ci$basic[4]` and `bst.ci$basic[5]`.

d) Compare the usual L_1-loss $\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{m}(x_i)|$ with the L_1-generalization error $\mathbf{E}\left[|Y_{\text{new}} - \hat{m}(X_{\text{new}})|\right]$. This time the L_1-generalization-error is estimated by bootstrapping instead of cross-validation as described in the manuscript. Do 100 simulations for each of the given error distributions. In each simulation calculate the two quantities of interest and compare their averages over the whole range of simulations. A histogram of the two quantities may be informative, too. You might want to recycle the bootstrap-samples you generated above.

Preliminary discussion: Friday, April 22.

Deadline: Friday, April 29.

[1] It depends on the computer time you can spend whether you try 25, 50, 100 or 200 simulations. It may need lots of time, because each time a complete bootstrap simulation has to be carried out. You can always downsize your simulations by simulating fewer datasets and/or varying the number of bootstrap replicates.

[2] The bootstrap replicates should be generated by sampling from the set $\{(x_1, Y_1), \ldots, (x_{100}, Y_{100})\}$, and **not** by resampling the residuals as in the model-based bootstrap.