Solution to Series 6

1. Help function giving regression coefficients, as described on the exercise sheet:

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

a) First we complete the function that generates a set of bootstrapped parameter estimators, that is, generates a matrix of 3 $\times$ B where the (i, j)-th element corresponds to $\hat{\beta}_i^j$, where j means that we used the j-th bootstrap sample to make the estimation. The code should look as follows:

```
> obst.est <- function(data, B)
   {
     ind <- replicate(B, sample.int(nrow(data), replace = TRUE))
     apply(ind, 2, lmcoefs, data = data)
   }
```

Now we use this function to calculate bootstrapped confidence intervals:

```
> obst.ci <- function(bst.pars, data, alpha)
   {
     ## Estimate regression parameters
     reg.pars <- lmcoefs(data, 1:nrow(data))

     ## Calculate empirical quantiles of the bootstrap distribution
     qt <- apply(bst.pars, 1, quantile,
                 probs = c(1 - alpha/2, alpha/2), names = FALSE)

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

b) In order to be able to use the same bootstrap samples for all methods, it is better to use the same R-code for assignment **b)** and **c)**. Here, we define *one* function for all these simulations:

```
> ##' Simulates the datasets and calculates the confidence intervals
> ##' for the different methods
> ##'
> ##' @param err.type: error distribution to be used. 1 = Gaussian,
> ##'   2 = t_2, 3 = exponential
> ##' @param nruns: number of datasets to be simulated
> ##' @param alpha: 1 - confidence level
> ##' @param B: number of bootstrap samples
> ##' @param true.coef: true coefficients of generated data
> ##' @return list with  lower and upper bounds of confidence
> ##'   intervals, L1-errors and L1-generalization errors
> ##'   (cf. code for details...)
> Simu <- function(err.type, nruns, alpha, B, true.coef, sqrt.n = 5, verbose=TRUE)
 {
     nErrT <- length(errTypes <- 1:3) #
     stopifnot(length(true.coef) == 3, err.type %in% errTypes, B >= 1, nruns >= 1)

     ## boot pkg -> boot.ci() 'type' and resulting object --> need this:
     ciTypes <- c(normal = "norm", basic="basic", percent="perc", bca="bca")
     ##In Exercises sheet you are only asked to use
```

```
##ciTypes = "basic". However, as an extra exercise,
## we will also use ciTypes="norm", "perc", and "bca"
##(see below  sub-Exercise d)).


## Generate data for predictors and for response without noise
## (we will add it at the begining of each "for" loop)
x2 <- rep(1:sqrt.n, sqrt.n)
x3 <- rep(1:sqrt.n, each = sqrt.n)
X <- cbind("(I)" = 1, x2, x3)
n <- nrow(X)
Xb <- X %*% true.coef

## Initialize matrices to store lower and upper bounds
## of confidence intervals calculated by classical "normal
## theory" ("cl"), bootstrapping ("bst") and by the own
## bootstrap routine ("obst") of assignment c).

## Now, we compute (but not yet show)
## all 4  boot() intervals, notably the BCa:
boo.r <- array(NA, dim  = c   (    ncol(X),         2,         length(ciTypes), nruns),
               dimnames = list(colnames(X), c("lower","upper"), names(ciTypes), NULL))
## classical and own bootstrap need just one slice:
class.r <- ownB.r <- boo.r[,,1,]
err <- generr <- numeric(nruns)

for(i in 1: nruns) {
  if(verbose) { ## using stderr(), so it stays in console, not in TeX -> PDF output:
      cat(".", file=stderr()); if(i %% 10 == 0) cat(i, "\n", file=stderr()) }
  ## Choose response according to error type
  eps <- switch(err.type,
                rnorm(n),
                rt (n, 2),
                rexp(n, rate=3) - 1/3)

  dat <- data.frame(y = Xb + eps, x2, x3)

  ## Calculate regression coefficients and classical confidence intervals
  lmc <- lm(y ~ x2 + x3, data = dat)
  cl.ci <- confint(lmc, level = 1 - alpha)
  class.r[,,i] <- cl.ci

  ## Calculate L1-loss (assignment d))
  err[i] <- mean(abs(dat$y - fitted(lmc)))

  ## Calculate bootstrap - confidence intervals using own bootstrap routine
  bst.pars <- obst.est(dat, B)
  boundary <- obst.ci(bst.pars, dat, alpha)
  ownB.r[,,i] <- boundary
  ## Calculate regression fit for every set of bootstrap-estimates
  bst.fit <- X %*% bst.pars
  ## Calculate L1-bootstrapped-generalization error for each simulation run
  generr[i] <- mean(apply(abs(dat$y - bst.fit), 2, mean))

  ## Calculate bootstrap confidence intervals with routines from
  ## package "boot" (assignment c))
  bst <- boot(dat, statistic = lmcoefs, B)
  ## fast code to get all intervals for all three coefficients:
  getB.ci <- function(k) { ## using globals (bst, alpha, ciTypes, cetCI)
```

```
            bci <- boot.ci(bst, conf = 1-alpha, type = ciTypes, index = k)
            getCI <- function(v) v[length(v)+(-1:0)] ## the last two entries
            vapply(bci[names(ciTypes)], getCI, numeric(2))
        }
        BB <- vapply(1:3, getB.ci, matrix(NA_real_, 2, length(ciTypes)))
        boo.r[,,, i] <- aperm(BB, c(3, 1:2))
      }

      list(classic = class.r,
           ownBoot = ownB.r,
           bootAll = boo.r,
           err = err, generr = generr)
    }
```

Last, we provide a function to plot the confidence intervals *and* report the (empirical) coverage rates:

```
> ##' Count hits, i.e. number of simulations where the confidence
> ##' intervals include the true values
> nIncludes <- function(ciArr, theta) {
      stopifnot(is.matrix(low <- ciArr["lower",]),
                is.matrix(upp <- ciArr["upper",]),
                length(theta) == nrow(low))
      low < theta  &  theta <= upp
  }
> ##' @param true.coef: true coefficients of simulated data
> ##' @param ci: return value of function "simulate"
> plot.ci <- function(true.coef, ci)
    {
      ## Labeling
      ptitle <- expression("intercept", theta[2], theta[3])

      cl.hit   <- nIncludes(ci$classic, true.coef)
      obst.hit <- nIncludes(ci$ownBoot, true.coef)
       bst.hit <- nIncludes(ci$bootAll[,,"basic",], true.coef)
      nruns <- ncol(cl.hit)

      op <- mult.fig(3, main = paste("standard (solid), bootstrap (dashed),",
                               "and own-bootstrap (dotted) confidence intervals"),
                     cex.main = 1, marP = c(1,0,0,0), mgp = c(1.75, 0.5, 0))$old.par
      on.exit(par(op))
      for(j in 1:3)
      {
        plot(seq_len(nruns), type = "n",
             ylim = extendrange(ci$classic[j,,]),
             main = substitute(paste("for ", PNAME), list(PNAME = ptitle[[j]])),
             sub = paste(
                 sum(cl.hit[j, ]), "normal-ci.,",
                 sum(bst.hit[j,]), "bootstrap-ci. and",
                 sum(obst.hit[j,]), "own-bootstrap-ci.",
                 "contain the true value", true.coef[j], ".\n",
             "Approximate confidence levels:",
                "standard normal:", signif(sum( cl.hit [j,]/nruns),4),
                "bootstrap:",       signif(sum( bst.hit[j,]/nruns),4),
                "own-bootstrap:",   signif(sum(obst.hit[j,]/nruns),4), "."),
             xlab = NA, ylab = "Confidence intervals", cex.main=1.75, cex.sub= 1.4)
        ii <- 1:nruns
        segments(ii, ci$classic[j,"lower",], ## col = 2 - is.inside() <==> 'red' if outside
                 ii, ci$classic[j,"upper",],                    col= 2 - cl.hit[j,])
        segments(ii+0.25, ci$bootAll[j,"lower","basic",],
                 ii+0.25, ci$bootAll[j,"upper","basic",], lty=2, col= 2- bst.hit[j,])
        segments(ii+0.5,  ci$ownBoot[j,"lower",],
```

```
            ii+0.5,  ci$ownBoot[j,"upper",],         lty=3, col= 2-obst.hit[j,])
    abline(h = true.coef[j])
  }
}
```

Now, we will set the seed and define the parameters to run the simulations and plots. We will also create a matrix `errors` to store the L1-error and estimation of the Generalization Error of each configuration of the error (Exercise d)) and display all of them in a table for comparison.

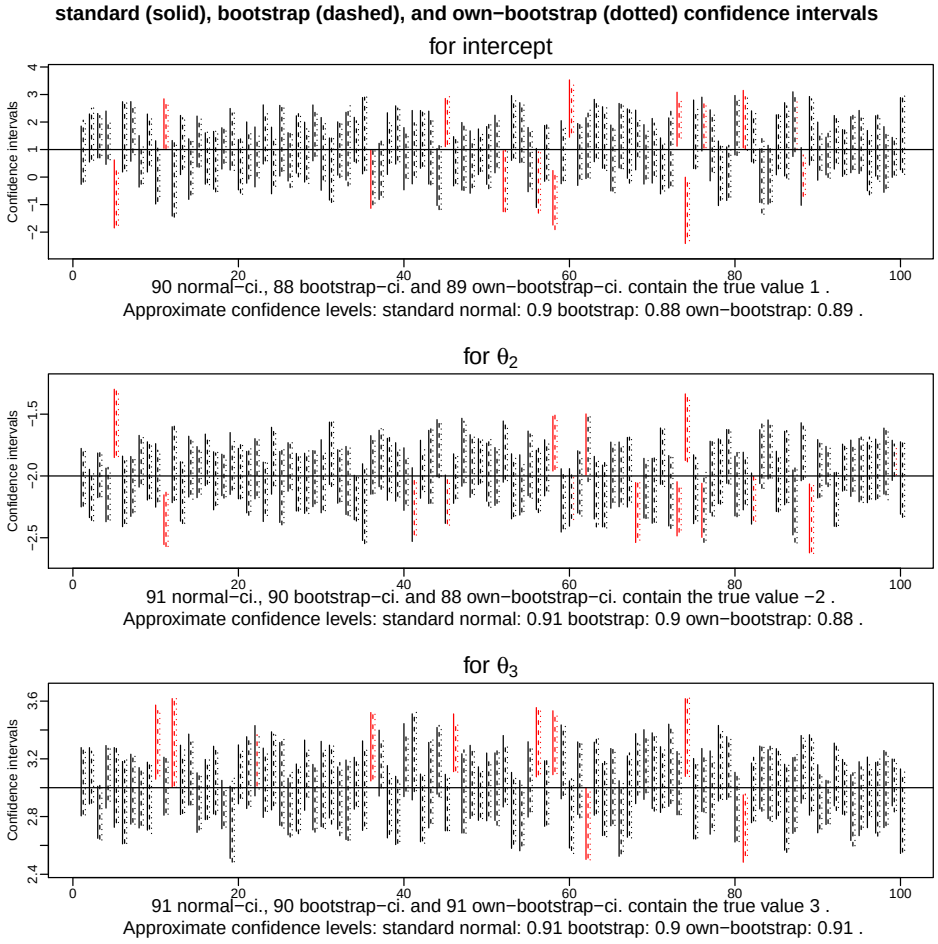
```
> library(boot)
> set.seed(84)
> B <- 1000
> nruns <- 100
> alpha <- 0.1
> true.coef <- c(1, -2, 3)
> errors <- data.frame(l1loss = rep(NA, 3),
                       generr = rep(NA, 3))
```

1. Confidence intervals of the three parameters for the simulations with normally distributed errors, $\varepsilon \sim \mathcal{N}(0, 1)$:

   ```
   >  ci1 <- Simu(1, nruns, alpha, B, true.coef)
   ```

   Plots of the confidence intervals. The number of simulations where the confidence intervals include the true values are specified in the legend of each plot.

   ```
   > plot.ci(true.coef, ci1)
   ```



L1-loss and estimation of the Generalization Error.

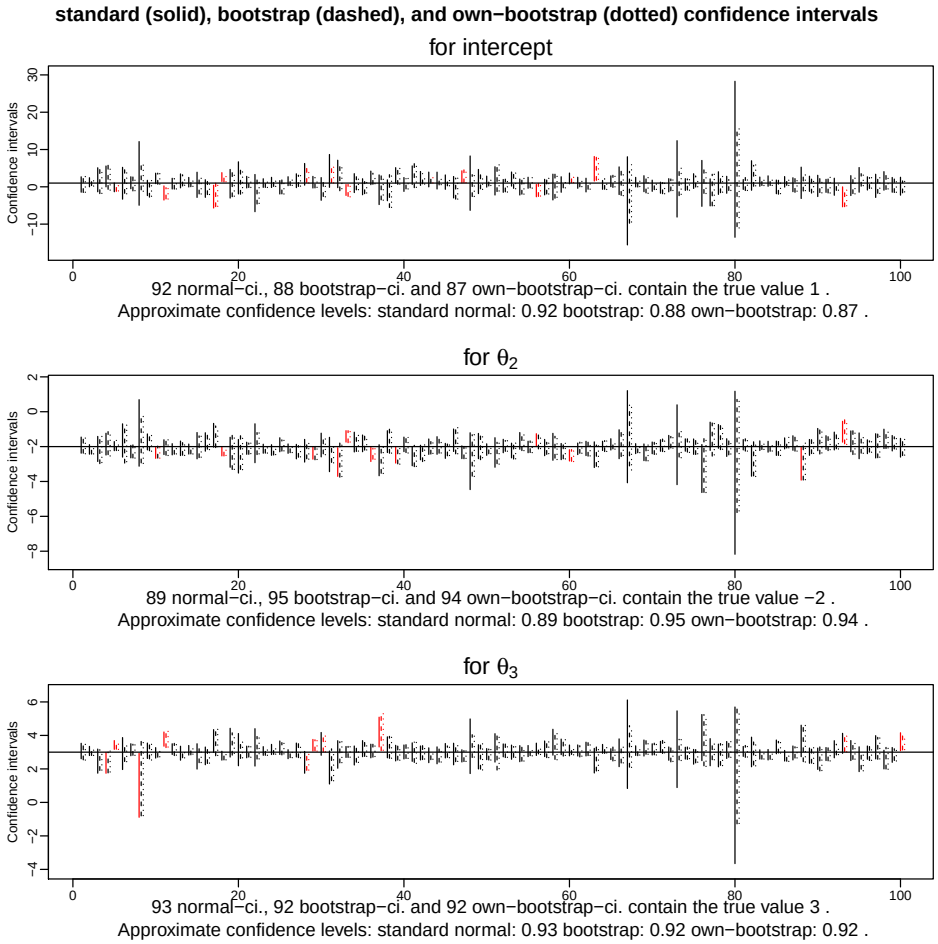
```
> (errors$l1loss[1] <- mean(ci1$err))
[1] 0.7519276
> (errors$generr[1] <- mean(ci1$generr))
[1] 0.7954565
```

2. Confidence intervals of the three parameters for the simulations with t_2 distributed errors, $\varepsilon \sim t_2$:

```
> ci2 <- Simu(2, nruns, alpha, B, true.coef)
```

Plots of the confidence intervals.

```
> plot.ci(true.coef, ci2)
```



L1-loss and estimation of the Generalization Error.

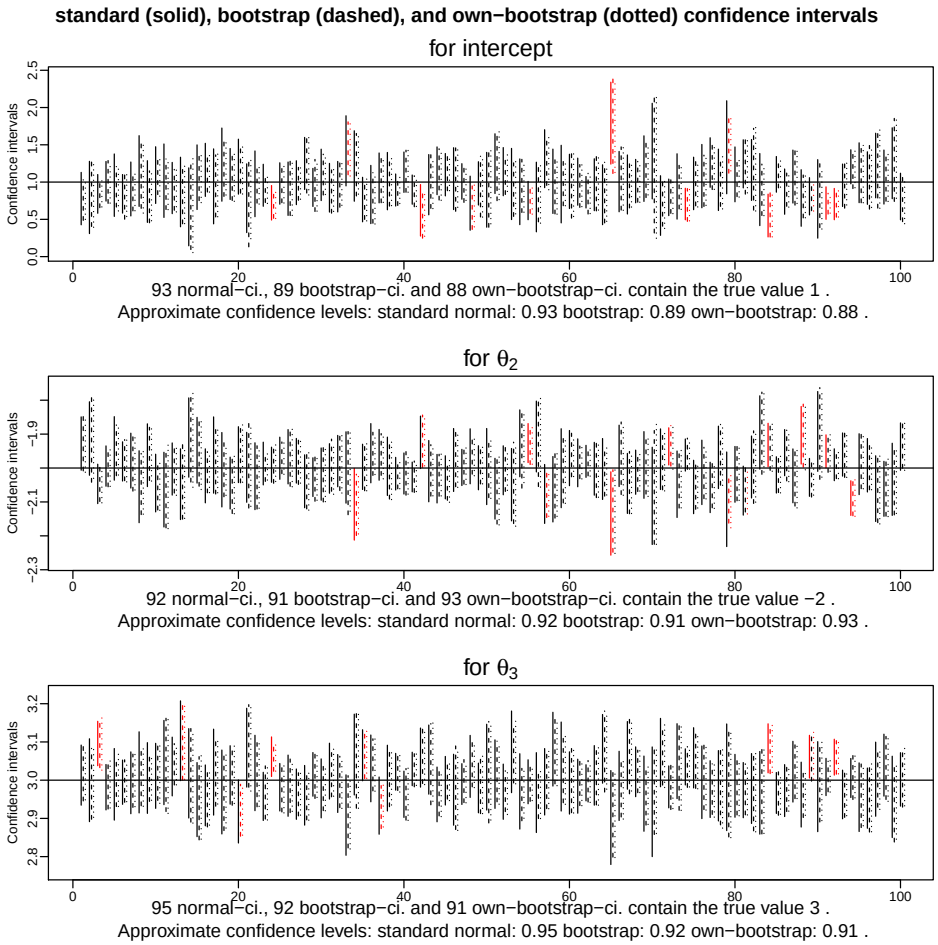
```
> (errors$l1loss[2] <- mean(ci2$err))
[1] 1.524357
> (errors$generr[2] <- mean(ci2$generr))
[1] 1.660723
```

3. Confidence intervals of the three parameters for the simulations with exponentially distributed errors, $\varepsilon \sim \mathrm{Exp}(3) - 1/3$:

   ```
   > ci3 <- Simu(3, nruns, alpha, B , true.coef)
   ```

   Plots of the confidence intervals.

   ```
   > plot.ci(true.coef, ci3)
   ```



   L1-loss and estimation of the Generalization Error.

   ```
   > (errors$l1loss[3] <- mean(ci3$err))
   [1] 0.2411664
   > (errors$generr[3] <- mean(ci3$generr))
   [1] 0.2554702
   ```

   In the t-student and Exponential case, we expect that the Confidence Intervals constructed using classical theory perform worse than those constructed using bootstrap. This is because classical theory relies on the assumption of Gaussian distributed errors (in this case, standard Gaussian) and Bootstrap does not make any assumption on the underlying distribution (it only uses the empirical distribution of the data)

c) cf. **b)**.

d) Finally, we take the values of the L1-loss and the estimation of the Generalization Error saved in the matrix `errors` and display them in a table:

```
> errs <- round(errors, digits = 4) #--> show in LaTeX table (via \Sexpr{*})
```

Error distribution:	Normal	t_2	Exponential
L_1-loss	0.7519	1.5244	0.2412
Generalization error	0.7955	1.6607	0.2555

As expected, the generalization error is larger than the L_1-loss in all three cases.

[Extra Exercise, not in Exercises sheet] In the plots above we used the "basic" bootstrap Confidence Intervals, however, using `boot.ci`, there are *more* possibilities (see help file `boot.ci -> type`). In the tables below, we will display the approximate confidence levels of the Confidence levels already shown above (columns `classic`, `ownBoot`, and `basic`) and we will add those of the bootstrap Confidence Intervals using `type = normal, percent`, and `bca`.

```
> allCI <- function(ci, th.true) {
     cover <- function(mat) rowMeans(nIncludes(mat, th.true))
     cbind(classic = cover(ci$classic),
           ownBoot = cover(ci$ownBoot),
           sapply(dimnames(ci$bootAll)[[3]], function(II) cover(ci$bootAll[,,II,])))
 }
> allCI(ci1, true.coef)# N(0,1)

    classic ownBoot normal basic percent  bca
(I)    0.90    0.89   0.89  0.88    0.87 0.89
x2     0.91    0.88   0.89  0.90    0.88 0.88
x3     0.91    0.91   0.89  0.90    0.89 0.89

> allCI(ci2, true.coef)# t_2()

    classic ownBoot normal basic percent  bca
(I)    0.92    0.87   0.89  0.88    0.86 0.85
x2     0.89    0.94   0.93  0.95    0.88 0.82
x3     0.93    0.92   0.91  0.92    0.91 0.87

> allCI(ci3, true.coef)# Exp(3)

    classic ownBoot normal basic percent  bca
(I)    0.93    0.88   0.89  0.89    0.87 0.88
x2     0.92    0.93   0.92  0.91    0.91 0.88
x3     0.95    0.91   0.92  0.92    0.91 0.87

> ## coverage - true alpha level:
> allCI(ci1, true.coef) - 0.90

    classic ownBoot normal basic percent   bca
(I)    0.00   -0.01  -0.01 -0.02   -0.03 -0.01
x2     0.01   -0.02  -0.01  0.00   -0.02 -0.02
x3     0.01    0.01  -0.01  0.00   -0.01 -0.01

> allCI(ci2, true.coef) - 0.90

    classic ownBoot normal basic percent   bca
(I)    0.02   -0.03  -0.01 -0.02   -0.04 -0.05
x2    -0.01    0.04   0.03  0.05   -0.02 -0.08
x3     0.03    0.02   0.01  0.02    0.01 -0.03

> allCI(ci3, true.coef) - 0.90

    classic ownBoot normal basic percent   bca
(I)    0.03   -0.02  -0.01 -0.01   -0.03 -0.02
x2     0.02    0.03   0.02  0.01    0.01 -0.02
x3     0.05    0.01   0.02  0.02    0.01 -0.03

> ## total coverage "deviation" over all three coefficients:
> ## small is good:
> rbind(
     Norm = colSums(abs(allCI(ci1, true.coef) - 0.90 )),
     t_2  = colSums(abs(allCI(ci2, true.coef) - 0.90 )),
     Exp =  colSums(abs(allCI(ci3, true.coef) - 0.90 )))
```

```
     classic ownBoot normal basic percent  bca
Norm    0.02    0.04   0.03  0.02    0.06 0.04
t_2     0.06    0.09   0.05  0.09    0.07 0.16
Exp     0.10    0.06   0.05  0.04    0.05 0.07

>
```