MLStats: Resampling Methods

Darshan Patel

2/2/2019

In this assignment, mimic the lab exercises from ISLR Chapter 5: Resampling Methods.

Libraries

Load the following libraries.

```
rm(list = ls())
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 3.4.4
```

```
library(boot)
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.4

## Warning: package 'purrr' was built under R version 3.4.4

## Warning: package 'dplyr' was built under R version 3.4.4
```

```
library(scales)
```

```
## Warning: package 'scales' was built under R version 3.4.4
```

```
library(gridExtra)
library(RColorBrewer)
```

Dataset

In this assignment, the dataset that will be used is `BodyFat.csv` (Source: https://www2.stetson.edu/~jrasp/data.htm).

This dataset contains bodyfat measurements from a sample of men of different body size. The target variable in this study is body fat. It is measurable by weighing a person underwater, which is an inconvenient method. Try to predict body fat percentage using other easily attainable health measurements.

```
df = read_excel("BodyFat.xls")
```

The number of men whose body measurements were taken from is

```
nrow(df)
```

```
## [1] 252
```

The columns in this dataset are:

```
colnames(df)
```

```
##  [1] "IDNO"       "BODYFAT"    "DENSITY"    "AGE"        "WEIGHT"
##  [6] "HEIGHT"     "ADIPOSITY"  "NECK"       "CHEST"      "ABDOMEN"
## [11] "HIP"        "THIGH"      "KNEE"       "ANKLE"      "BICEPS"
## [16] "FOREARM"   "WRIST"
```

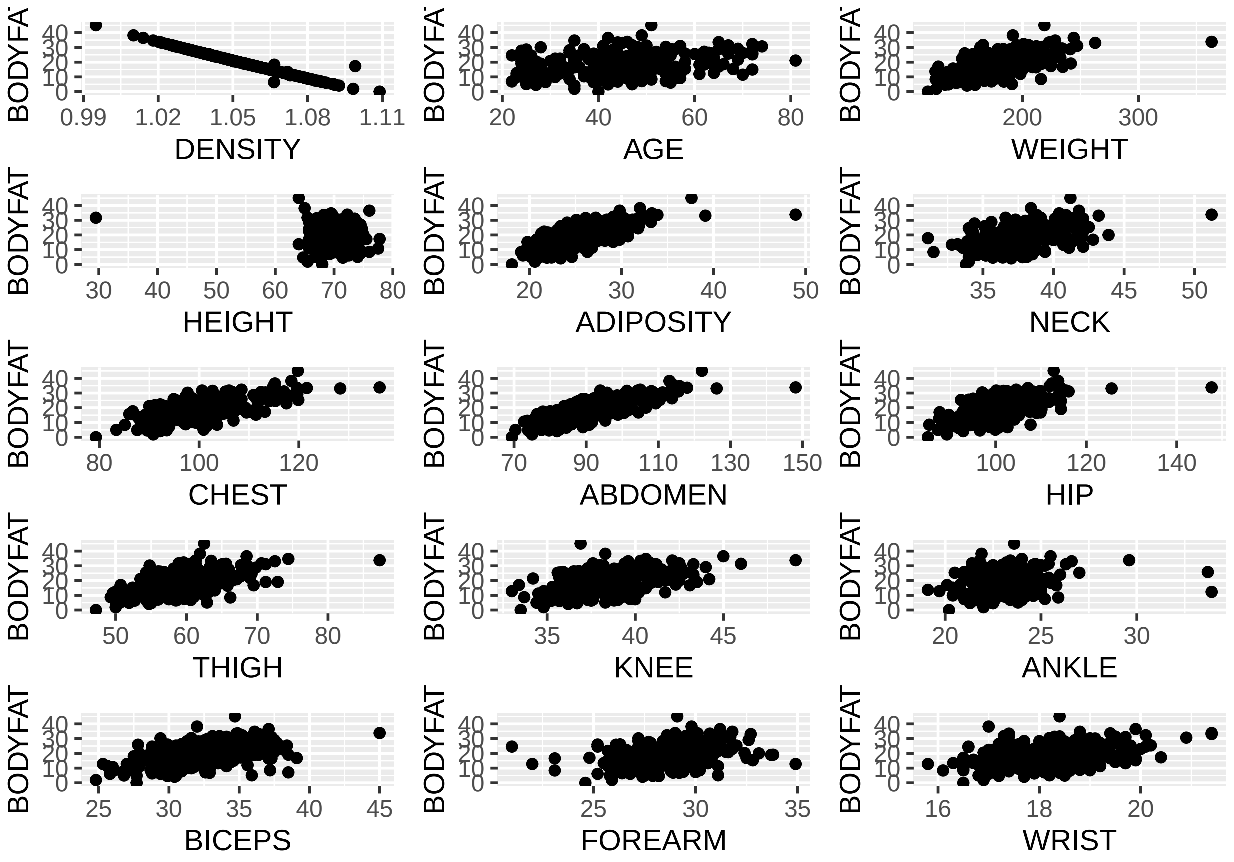
Ignore the `IDNO` column, which is simply an identifier for each man. Various pieces of information about the body are given here such as age, weight, neck measurement, chest measurement, forearm measurement, etc.

In this assignment, use linear regression with various forms of sampling methods to best predict the body fat percentage of a person.

EDA

Before regressing on `BODYFAT`, find the predictors that best correlate with body fat.

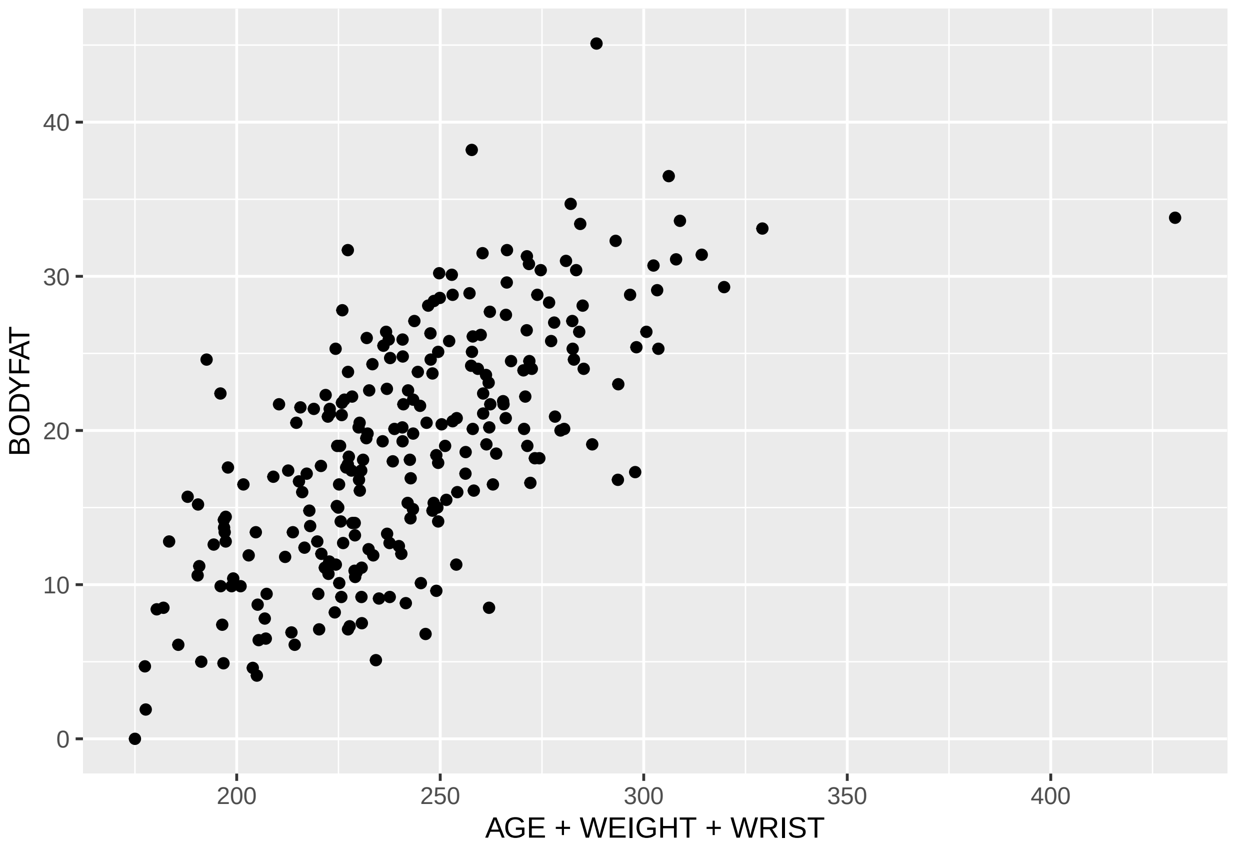
```r
g1 = ggplot(df, aes(x = DENSITY, y = BODYFAT)) + geom_point()
g2 = ggplot(df, aes(x = AGE, y = BODYFAT)) + geom_point()
g3 = ggplot(df, aes(x = WEIGHT, y = BODYFAT)) + geom_point()
g4 = ggplot(df, aes(x = HEIGHT, y = BODYFAT)) + geom_point()
g5 = ggplot(df, aes(x = ADIPOSITY, y = BODYFAT)) + geom_point()
g6 = ggplot(df, aes(x = NECK, y = BODYFAT)) + geom_point()
g7 = ggplot(df, aes(x = CHEST, y = BODYFAT)) + geom_point()
g8 = ggplot(df, aes(x = ABDOMEN, y = BODYFAT)) + geom_point()
g9 = ggplot(df, aes(x = HIP, y = BODYFAT)) + geom_point()
g10 = ggplot(df, aes(x = THIGH, y = BODYFAT)) + geom_point()
g11 = ggplot(df, aes(x = KNEE, y = BODYFAT)) + geom_point()
g12 = ggplot(df, aes(x = ANKLE, y = BODYFAT)) + geom_point()
g13 = ggplot(df, aes(x = BICEPS, y = BODYFAT)) + geom_point()
g14 = ggplot(df, aes(x = FOREARM, y = BODYFAT)) + geom_point()
g15 = ggplot(df, aes(x = WRIST, y = BODYFAT)) + geom_point()

grid.arrange(g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,ncol=3)
```



According to these plots, it appears that `DENSITY`, `WEIGHT`, `ADIPOSITY`, `WRIST`, `BICEPS`, among others variables, correlate with `BODYFAT`.

Some of the correlated variables are plotted here against body fat percentage.

```r
ggplot(df, aes(x = AGE+WEIGHT+WRIST, y = BODYFAT)) + geom_point()
```



Age, weight and wrist measurements are easy to measure. Use these explanatory variables for regression.

Base Regression

Before starting to use different resampling methods, first create a model using the entire dataset to predict body fat percentage and record its accuracy.

```
model0 = lm(BODYFAT~AGE + WEIGHT + WRIST, data = df)
mse0 = mean((df$BODYFAT - predict(model0))^2)
mse0
```

```
## [1] 28.11141
```

The MSE of this base model is 28.11141.

Validation Set Approach

Split the data ino a training set and a validation set. Train the data on the training set and test on the validation set and report the test set MSE.

```
set.seed(2019)
indices = sample(1:nrow(df), size = round(0.7*nrow(df)))
train = df[indices,]
test = df[-indices,]

model1 = lm(BODYFAT~AGE + WEIGHT + WRIST, data = train)
```

```r
mse1 = mean((test$BODYFAT - predict(model1, test))^2)
mse1
```

```
## [1] 26.42922
```

By splitting up the data, the test set MSE slightly went down and the model performed slightly better than when using the entire dataset.

If different indices are sampled, will the model perform similary?

```r
set.seed(2020)
indices = sample(1:nrow(df), size = round(0.7*nrow(df)))
train = df[indices,]
test = df[-indices,]

model1b = lm(BODYFAT~AGE + WEIGHT + WRIST, data = train)
mse1b = mean((test$BODYFAT - predict(model1b, test))^2)
mse1b
```

```
## [1] 28.63627
```

By splitting up the data differently, a different test set MSE was calculated. The model performed slightly worse than the previous train/test split. In fact, it did slightly worse than the base model as well. In actuality, this is due to randomness of the choice of split. By tinkering with the split manually, a best model can be found. But this is not feasable.

Can body fat percentage be predicted better by LOOCV?

Leave-One-Out Cross Validation

Predict body fat percentage using the LOOCV approach.

```r
model2 = glm(BODYFAT~AGE + WEIGHT + WRIST, data = df)
mse2 = cv.glm(df, model2)$delta[1]
mse2
```

```
## [1] 29.60999
```

By using LOOCV to do 252 splits, the model performed the worse compared to the base model and both of the train/test splits.

How about k-fold CV?

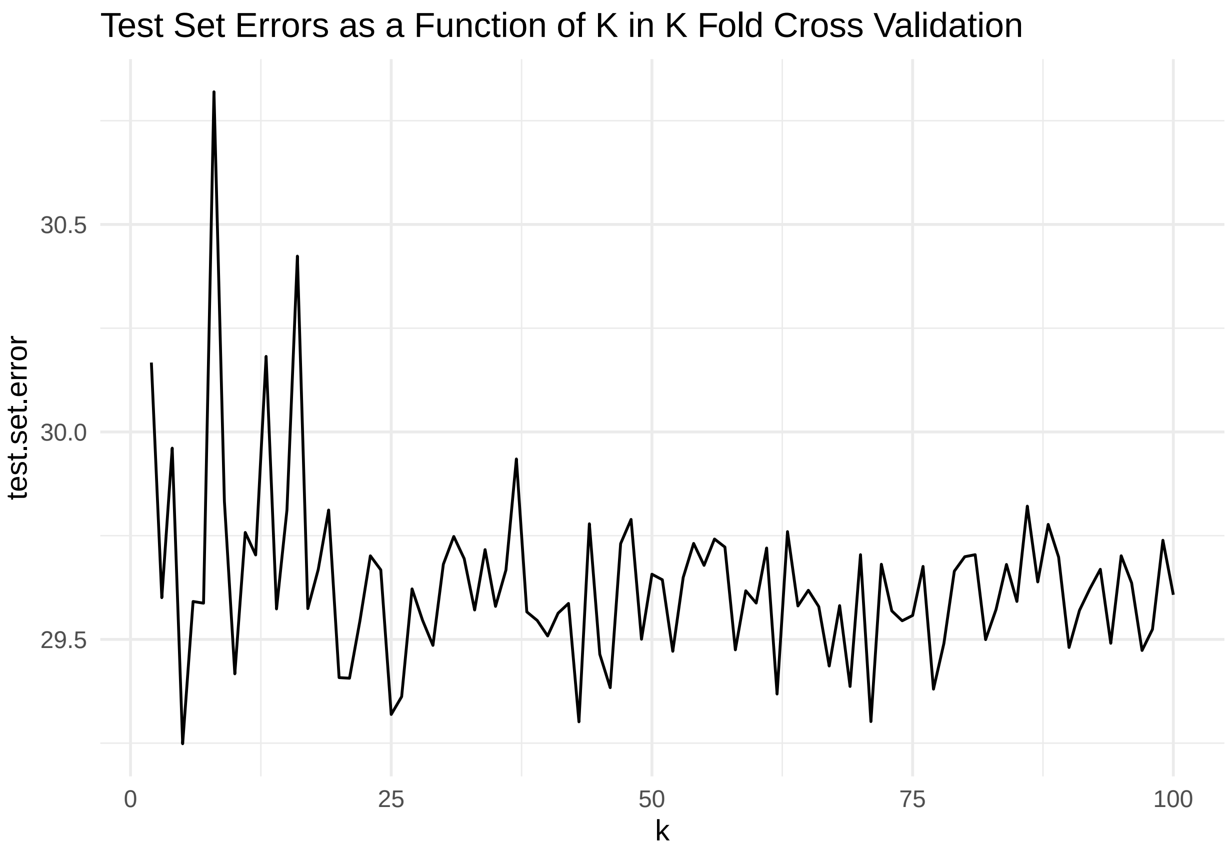
k-Fold Cross Validation

```r
set.seed(1)
model3 = glm(BODYFAT~AGE + WEIGHT + WRIST, data = df)
mse3 = cv.glm(df, model3, K = 10)$delta[1]
mse3
```

```
## [1] 29.60261
```

The model created using k-fold cross validation performed slightly better than by the LOOCV method but relatively the same. Here a k value was manually selected. Is there a best k value that results in the lowest test set error?

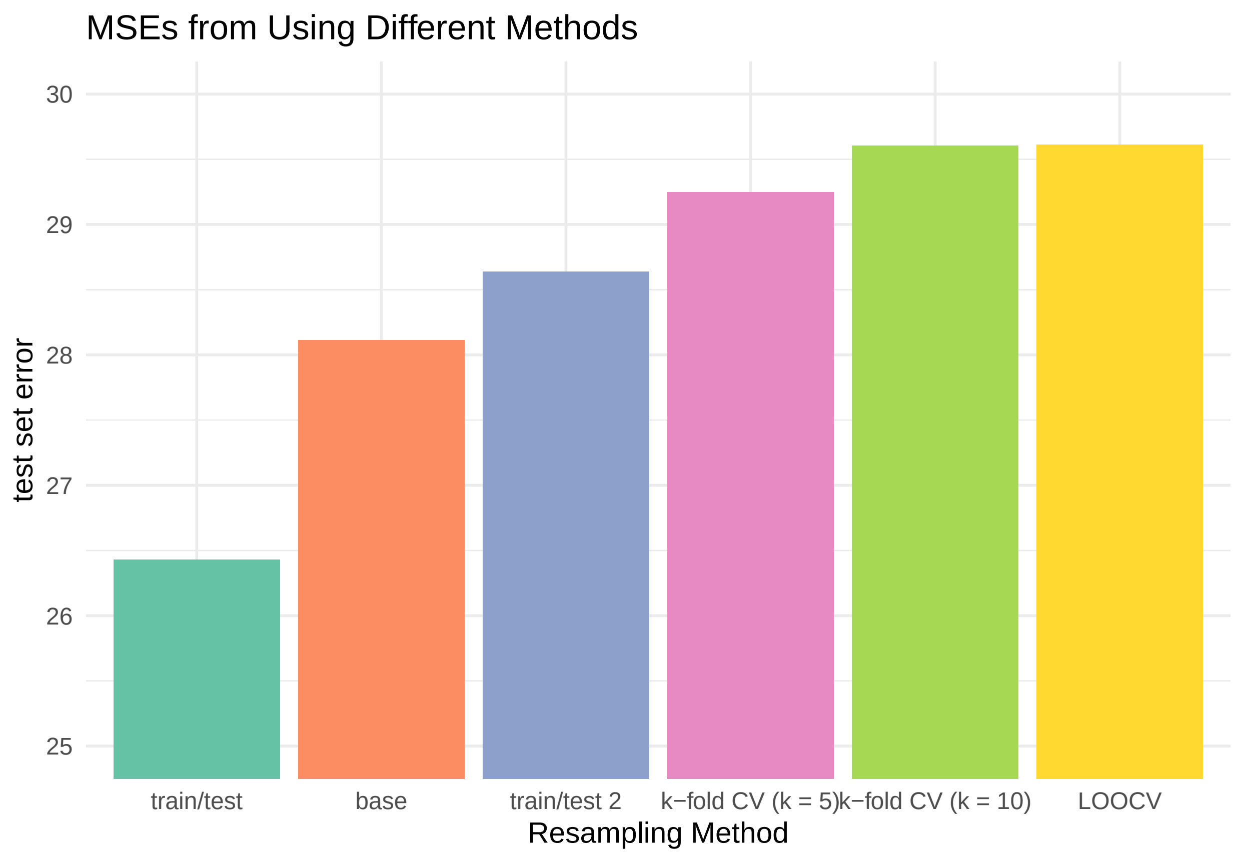
```r
set.seed(1)
error_list = c()
for(i in 2:100){
  error_list = c(error_list, cv.glm(df, model3, K = i)$delta[1])
}

min_k_kfold = which.min(error_list)+1
min_k_kfold_error = error_list[which.min(error_list)]

kfold_cv_errors = data.frame("k" = seq(2,100,by=1), "test set error" = error_list)
ggplot(kfold_cv_errors, aes(x = k, y = test.set.error)) +
  geom_path() +
  ggtitle("Test Set Errors as a Function of K in K Fold Cross Validation") +
  theme_minimal()
```



The lowest test set error is found for when 5 and its associated error is 29.2484814. The highest test set error is found for when 8 and its associated error is 30.8197089.

This plot shows how the accuracies of the models based on the type of resampling methods.

```r
mses = data.frame("method" = c("base", "train/test", "train/test 2",
                               "LOOCV", "k-fold CV (k = 10)",
                               paste("k-fold CV (k = ", min_k_kfold, ")", sep = "")),
                  "MSE" = c(mse0, mse1, mse1b, mse2, mse3, min_k_kfold_error))
ggplot(mses, aes(x = reorder(method, MSE), y = MSE)) +
  geom_bar(stat = "identity", fill = brewer.pal(n = 6, name = "Set2")) +
  scale_y_continuous(limits=c(25,30),oob = rescale_none) +
  ggtitle("MSEs from Using Different Methods") +
  xlab("Resampling Method") +
  ylab("test set error") +
  theme_minimal()
```



As seen from this plot, the model that performed the best was the first validation set model. The base performed next best, with ≈ 2% jump in test set error. The LOOCV method performed the worst, along with k-fold cross validation with $k = 10$.

Bootstrap - Estimating the Accuracy of a Statistic in Interest

Estimate the mean body fat percentage using bootstrap and check the result with the actual mean body fat percentage.

First calculate the actual mean body fat percentage. The mean body fat percentage is

```
mu = mean(df$BODYFAT)
mu
```

```
## [1] 18.93849
```

Now write a function to find the mean body fat percentage using bootstrapping.

```
boot_fn = function(data, index){return(mean(data[index]))}
```

Test the function.

```
set.seed(42)
trial1 = boot_fn(df$BODYFAT, sample(1:nrow(df), size = 0.2*nrow(df)))
set.seed(84)
```

```r
trial2 = boot_fn(df$BODYFAT, sample(1:nrow(df), size = 0.2*nrow(df)))
trial1
```

```
## [1] 17.716
```

```r
trial2
```

```
## [1] 19.264
```

The two body fat percentage means found via bootstrapping comes very close to the actual mean body fat percentage.

Instead of repeatedly doing this for countless number of trials, `boot()` can automate this process and create estimates instantly.

```r
boot(df$BODYFAT,boot_fn,R=1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df$BODYFAT, statistic = boot_fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 18.93849 0.002828175   0.4886926
```

By bootstrapping and creating 1000 estimates for the mean body fat percentage, it was found that the standard error in estimating the statistic is 0.4886926. This is a small error.

Bootstrap - Estimating the Accuracy of a Linear Regression Model

Use the bootstrap approach to assess the coefficient estimates for the linear regression model above.

First write a function to get the coefficients of the linear model

```r
boot_lr_fn = function(data, index){return(coef(lm(BODYFAT~AGE + WEIGHT + WRIST,
                                                  data = data, subset = index)))}
```

Test it.

```r
boot_lr_fn(df, 1:100)
```

```
## (Intercept)         AGE      WEIGHT       WRIST
##  32.5461223   0.2730152   0.2298021  -3.6936240
```

```r
boot_lr_fn(df, sample(1:nrow(df), size = 0.7*nrow(df)))
```

```
## (Intercept)         AGE      WEIGHT       WRIST
##  29.7135973   0.2483076   0.2313550  -3.4724441
```

```r
boot_lr_fn(df, sample(1:nrow(df), size = 0.7*nrow(df), replace = T))
```

```
## (Intercept)         AGE      WEIGHT       WRIST
##  32.6701009   0.2726260   0.2923176  -4.2765842
```

The coefficicent estimates for `AGE`, `WEIGHT` and `WRIST` all appear to come in relative neighborhood of each other with respect to each predictor variable.

Now, use `boot()` to compute the standard errors of 1,000 bootstrap estiamtes for the intercept and slope terms.

```
boot(df, boot_lr_fn, R=1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df, statistic = boot_lr_fn, R = 1000)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 25.6048638  0.1148909221  7.75008289
## t2*  0.2371860  0.0003817652  0.02695274
## t3*  0.2395194  0.0032695258  0.02577125
## t4* -3.3005535 -0.0380713326  0.58591677
```

According to this, the standard error in approximating the coefficient β_{AGE} is 0.02691630, 0.02577562 for β_{WEIGHT} and 0.58624776 for β_{WRIST}. The standard error in estimating the intercept is larger, 7.76207206.

Check with the actual standard errors calculated from the formulas for standard error.

```
summary(lm(BODYFAT~AGE + WEIGHT + WRIST, data = df))$coef
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 25.6048638 7.83140787  3.269510 1.230006e-03
## AGE          0.2371860 0.02831811  8.375771 4.089599e-15
## WEIGHT       0.2395194 0.01735108 13.804289 1.526334e-32
## WRIST       -3.3005535 0.55905989 -5.903757 1.162931e-08
```

Bootstrapping worked well for estimating the accuracy in coefficient estimates for linear regression. The standard errors here are similar to the ones found via bootstrapping.

All of the lab instructions in this document are taken from “An Introduction to Statistical Learning, with applications in R” (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.