Lab 2 2020-01-31 V1.03 - Exercise answers

Biomedical Data Science

Question 1

Note, if you attempt to load `data.x77` you will receive an error. The `data.x77` dataset includes several datasets, so you will need to load the complete `state` set to view the x77 subset. Alternatively, you can directly assign the subset into your global environment using the left assignment operator `data.x77 <- data.x77`.

Union and intersection of column names:

```
> union(colnames(state.x77), colnames(USArrests))
 [1] "Population" "Income"     "Illiteracy" "Life Exp"   "Murder"
 [6] "HS Grad"    "Frost"      "Area"       "Assault"    "UrbanPop"
[11] "Rape"
> intersect(colnames(state.x77), colnames(USArrests))
[1] "Murder"
```

Merge of the two dataframes:

```
> library(data.table)
> USdata.dt <- merge(data.table(USArrests, keep.rownames="State"),
+                  data.table(state.x77, keep.rownames="State"), by="State", all.xy=T)
> dim(USdata.dt)
[1] 50 13
> sum(is.na(USdata.dt))
[1] 0
```

Check the alphabetical ordering of state names:

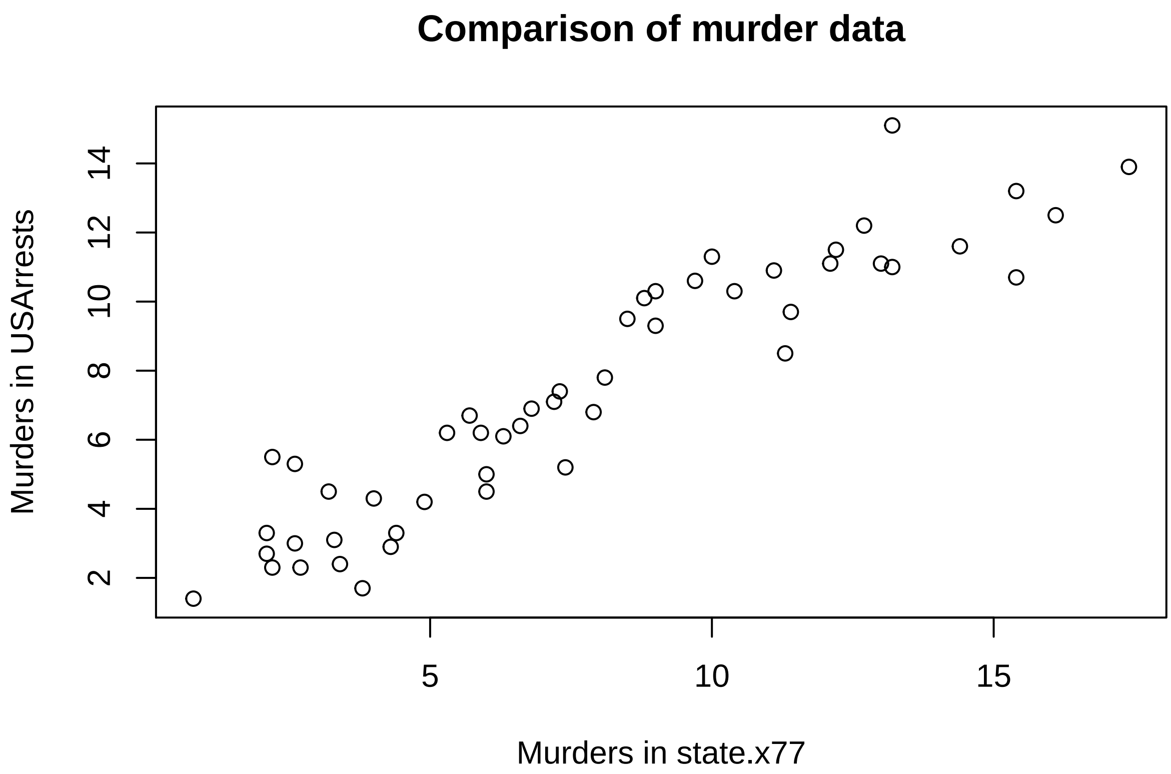
```
> all(USdata.dt$State == sort(USdata.dt$State))
[1] TRUE
```

Reorder the columns based on number of assaults. Note that the use of the efficient set* comnmands results in an updated data.table.

```
> setorder(USdata.dt, Assault)  # Order by Assault
> setorder(USdata.dt, State) # Return order back to by State.
```

Scatter plot and correlation of the two "Murder" variables:

```
> with(USdata.dt, plot(Murder.x, Murder.y, main="Comparison of murder data",
+                    xlab="Murders in state.x77", ylab="Murders in USArrests"))
```



```
> with(USdata.dt, signif(cor(Murder.x, Murder.y), 3))
[1] 0.934
```

Add two new variables derived from the “Murder” variables:

```
> USdata.dt[, MeanMurder := mean(c(Murder.x,Murder.y)), by=State]
> USdata.dt[, MaxMurder := max(c(Murder.x,Murder.y)), by=State]
> USdata.dt[, c("Murder.x","Murder.y") := NULL] # Remove columns
```

Question 2

Number of observed values per column:

```
> ## option 1: using a for loop
> num.cols <- ncol(airquality)
> num.obs <- integer(length=num.cols)     # initialize a vector of zeros
> for (i in 1:num.cols)
+   num.obs[i] <- sum(!is.na(airquality[, i]))
> names(num.obs) <- colnames(airquality)  # assign names to the vector elements
> num.obs
  Ozone Solar.R    Wind    Temp   Month     Day
    116     146     153     153     153     153
>
> ## option 2: using sapply() and an anonymous function
> num.obs <- sapply(airquality, function(z) sum(!is.na(z)))
> num.obs
  Ozone Solar.R    Wind    Temp   Month     Day
    116     146     153     153     153     153
```

Percentage of missing values per column:

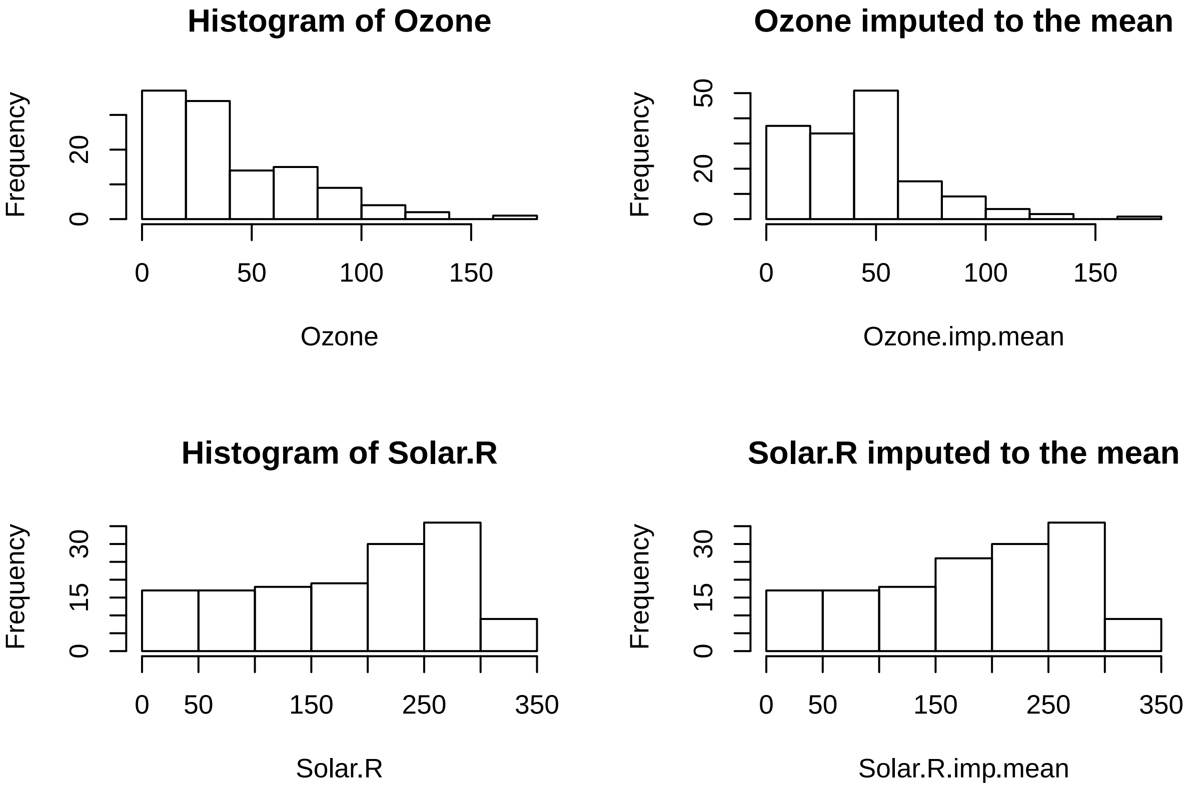
```
> sapply(airquality, function(z) sum(is.na(z)) / length(z) * 100)
    Ozone   Solar.R      Wind      Temp     Month       Day 
24.183007  4.575163  0.000000  0.000000  0.000000  0.000000 
```

Copy to data table and imputation to the mean:

```
> airquality.dt <- copy(data.table(airquality))
> airquality.dt[, Ozone.imp.mean := ifelse(is.na(Ozone),
+                           mean(Ozone, na.rm=TRUE), Ozone) ]
> airquality.dt[, Solar.R.imp.mean := ifelse(is.na(Solar.R),
+                           mean(Solar.R, na.rm=TRUE), Solar.R) ]
```

Plots of imputed and unimputed values:

```
> par(mfrow=c(2,2))
> with(airquality.dt, hist(Ozone))
> with(airquality.dt, hist(Ozone.imp.mean,
+                     main="Ozone imputed to the mean"))
> with(airquality.dt, hist(Solar.R))
> with(airquality.dt, hist(Solar.R.imp.mean,
+                     main="Solar.R imputed to the mean"))
```



Imputation to the monthly mean:

```
> impute.to.monthly.mean <- function(x, month) {
+   x.imp <- x
+   for (m in unique(month)) {
+     month.idx <- which(month == m)
+     mean.month <- mean(x[month.idx], na.rm=TRUE)
+     x.imp[is.na(x.imp) & month == m] <- mean.month
+   }
+   return(x.imp)
+ }
>
> airquality.dt$Ozone.imp.month <- with(airquality.dt,
+                                   impute.to.monthly.mean(Ozone, Month))
> airquality.dt$Solar.R.imp.month <- with(airquality.dt,
+                                   impute.to.monthly.mean(Solar.R, Month))
```

Here, with data tables the a function is overkill. It's much simpler to use the data table group by. Note that the mean results in a numeric type. We could cast this to an integer, but may lose precision, so we cast the original `non-NA` data to a numeric type instead.

```
> airquality.dt[, Ozone.imp.month.simple := ifelse(is.na(Ozone),
+                          mean(Ozone, na.rm=TRUE), as.numeric(Ozone)), by="Month"]
> airquality.dt[, Solar.R.imp.month.simple := ifelse(is.na(Solar.R),
+                          mean(Solar.R, na.rm=TRUE), as.numeric(Solar.R)), by="Month"]
>
> ## Assert that both our methods yield the same result
> stopifnot(nrow(airquality.dt[Ozone.imp.month != Ozone.imp.month.simple &
+                          Solar.R.imp.month != Solar.R.imp.month.simple]) == 0)
```

Maximum and mean absolute difference added as a columnn across the set.

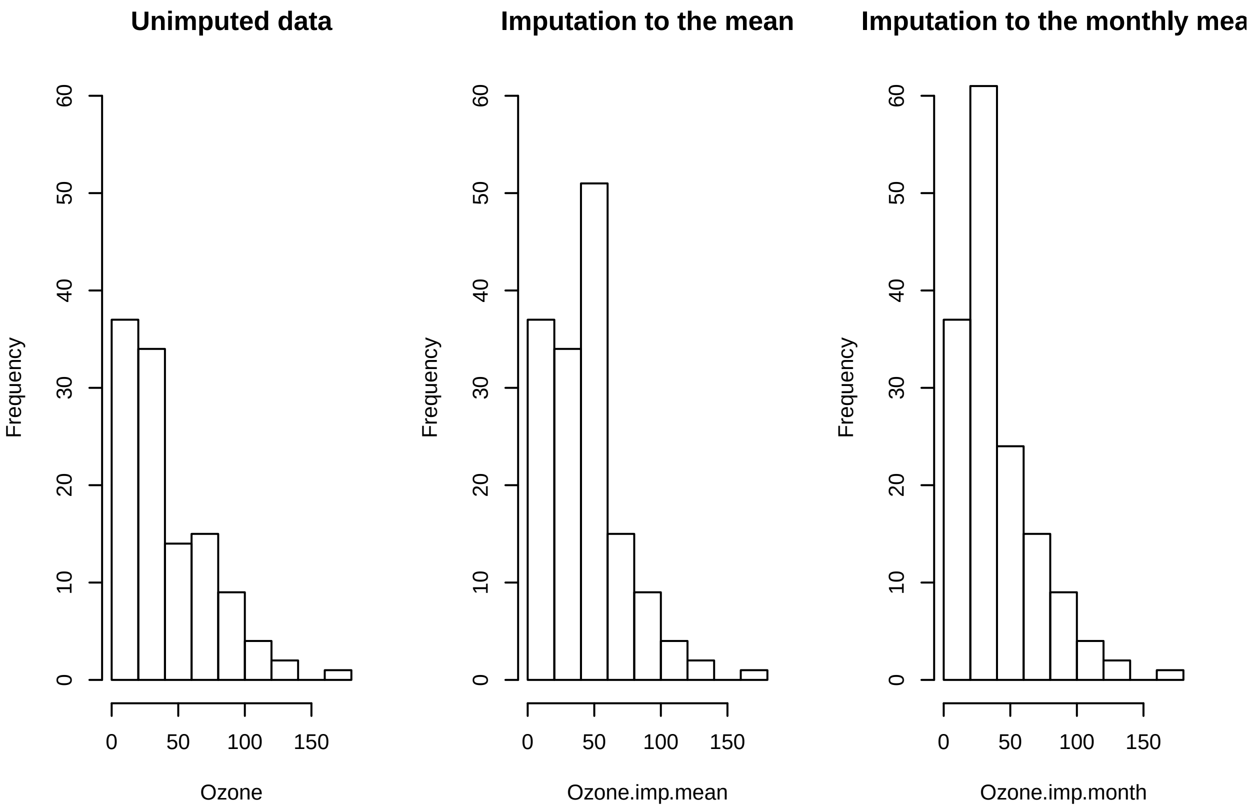
```
> airquality.dt[, Ozone.max.abs.diff :=
+                  round(max(abs(Ozone.imp.month - Ozone.imp.mean), na.rm=T),2)]
> airquality.dt[, Ozone.mean.abs.diff :=
+                  round(mean(abs(Ozone.imp.month - Ozone.imp.mean), na.rm=T),2)]
> airquality.dt[, Solar.R.max.abs.diff :=
+                  round(max(abs(Solar.R.imp.month - Solar.R.imp.mean), na.rm=T),2)]
> airquality.dt[, Solar.R.mean.abs.diff :=
+                  round(mean(abs(Solar.R.imp.month - Solar.R.imp.mean), na.rm=T),2)]
```

The above may be desirable, but may waste memory with repetitive data. The following shows the creation of a function to perform the task separately.

```
> max.abs.diff <- function(x, y) round(max(abs(x - y), na.rm=TRUE), 2)
> mean.abs.diff <- function(x, y) round(mean(abs(x - y), na.rm=TRUE), 2)
> max.abs.diff(airquality.dt$Ozone.imp.month, airquality.dt$Ozone.imp.mean)
[1] 18.51
> mean.abs.diff(airquality.dt$Ozone.imp.month, airquality.dt$Ozone.imp.mean)
[1] 3.55
> max.abs.diff(airquality.dt$Solar.R.imp.month, airquality.dt$Solar.R.imp.mean)
[1] 14.07
> mean.abs.diff(airquality.dt$Solar.R.imp.month, airquality.dt$Solar.R.imp.mean)
[1] 0.4
```

Graphical comparison of unimputed and imputed data:

```
> par(mfrow=c(1,3))
> ylim <- c(0, 60)      # set range of y axis to keep all plots consistent
> with(airquality.dt, hist(Ozone, main="Unimputed data", ylim=ylim))
> with(airquality.dt, hist(Ozone.imp.mean, main="Imputation to the mean", ylim=ylim))
> with(airquality.dt, hist(Ozone.imp.month, main="Imputation to the monthly mean", ylim=ylim))
```



Imputing to the mean seems to be affected by the presence of extreme values, which increases the mass towards the centre of the distribution. Imputing to the monthly mean reduces this effect because the extreme values affect only the imputation of missing value in the month when they occur, rather than biasing the overall mean.

Question 3

Fit a linear regression model for total cholesterol:

```
> diab01.dt <- fread("data/diab01.txt", stringsAsFactors = TRUE)
> regr.tc <- lm(Y ~ AGE + SEX + TC, data=diab01.dt)
```

Create the `results.table` vector:

```
> summ.regr <- summary(regr.tc)
> results.table <- c(summ.regr$coefficients[4, c(1, 4)],
+                    r.squared=summ.regr$r.squared,
+                    adj.r.squared=summ.regr$adj.r.squared)
> round(results.table, 4)
     Estimate      Pr(>|t|)     r.squared adj.r.squared
       0.0925        0.6999        0.0319       -0.0007
```

Build all other linear regression models:

```
> predictors <- c("BMI", "BP", "LDL", "HDL", "GLU")
> for (i in predictors) {
+   regr.i <- lm(paste0("Y ~ AGE + SEX + ", i), data=diab01.dt)
+   summ.regr <- summary(regr.i)
+
+   # append to the results vector
+   results.table <- rbind(results.table,
+                          c(summ.regr$coefficients[4, c(1, 4)],
+                            summ.regr$r.squared, summ.regr$adj.r.squared))
+ }
> rownames(results.table) <- c("TC", predictors)
> colnames(results.table) <- c("Coeff", "p.value", "r.squared", "adj.r.squared")
> results.table
          Coeff      p.value  r.squared adj.r.squared
TC   0.09251708 6.999078e-01 0.03192178 -0.0007100752
BMI  8.61276092 1.425186e-06 0.24675644  0.2227167504
BP   1.44490274 1.022339e-02 0.09168931  0.0627006722
LDL  0.01873207 9.351416e-01 0.01812697 -0.0135463562
HDL -2.20007522 7.562832e-05 0.16931725  0.1428060944
GLU  1.44411071 3.332959e-02 0.06955029  0.0381868168
```

Select the predictor that produces the best performing model:

```
> rownames(results.table)[which.max(results.table[, 4])]
[1] "BMI"
```

Starting from the best model, create models with one additional predictor:

```
> predictors <- c("TC", "BP", "LDL", "HDL", "GLU")
> r2.table <- NULL
> for (i in predictors) {
+   regr.i <- lm(paste0("Y ~ AGE + SEX + BMI + ", i), data=diab01.dt)
+   summ.regr <- summary(regr.i)
+   r2.table <- rbind(r2.table,
+                     c(summ.regr$r.squared, summ.regr$adj.r.squared))
+ }
> rownames(r2.table) <- predictors
> colnames(r2.table) <- c("r.squared", "adj.r.squared")
> r2.table
    r.squared adj.r.squared
TC  0.2634702     0.2292130
BP  0.2701687     0.2380882
LDL 0.2404929     0.2067370
HDL 0.2839504     0.2524757
GLU 0.2273689     0.1914326
```

Adjusted R^2 for the best fitting model:

```
> round(r2.table[which.max(r2.table[, 2]), 2], 3)
[1] 0.252
```

Question 4

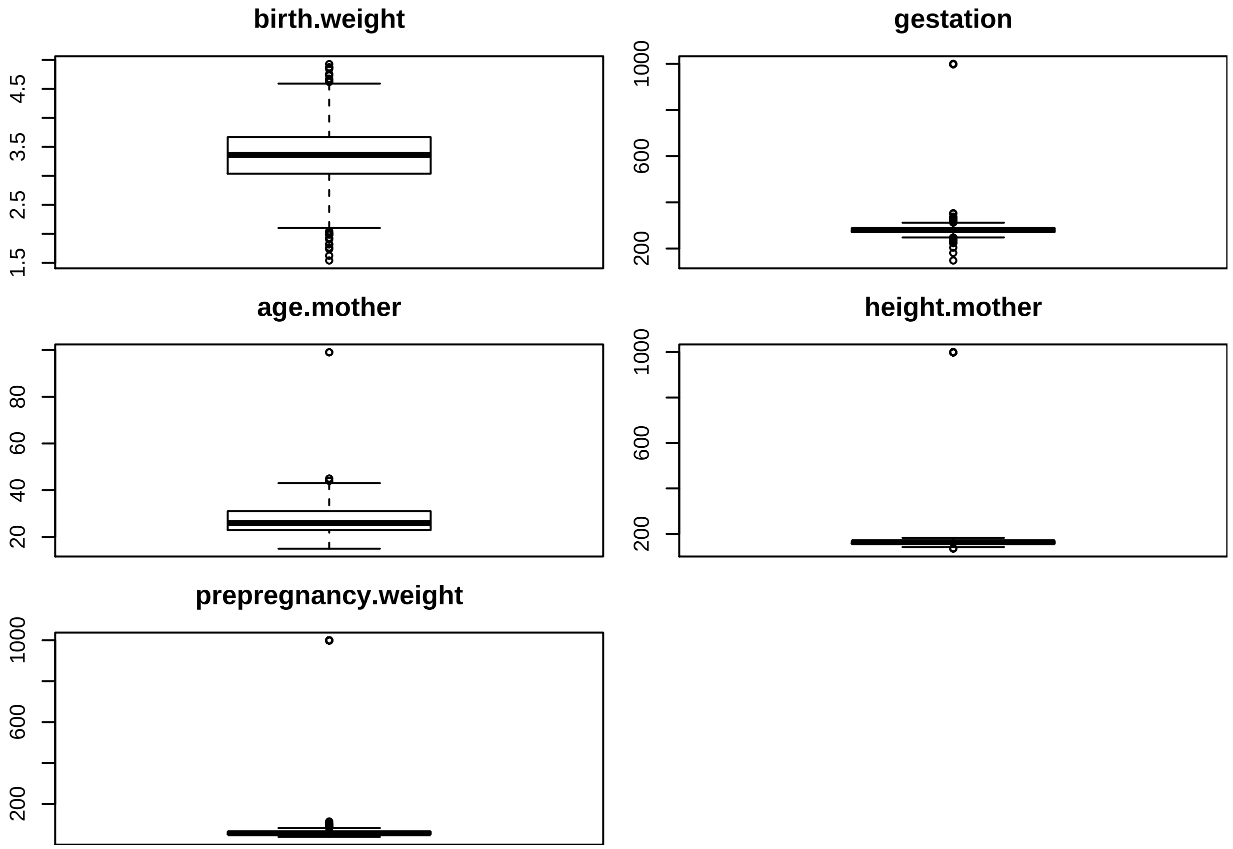
Here we will deal with dataframes. You should now find it easy to switch between dataframe and data table format to generate efficient code with minimal syntax and complexity.

Explore the dataset:

```
> bw.dt <- fread("data/birthweight.txt", stringsAsFactors = TRUE)
>
```

```
> # this is only one of possible explorations!
> summary(bw.dt)
  birth.weight     gestation       first.born       age.mother   
 Min.   :1.540   Min.   :148.0   Min.   :0.0000   Min.   :15.00  
 1st Qu.:3.045   1st Qu.:272.0   1st Qu.:0.0000   1st Qu.:23.00  
 Median :3.360   Median :280.0   Median :0.0000   Median :26.00  
 Mean   :3.348   Mean   :286.9   Mean   :0.2549   Mean   :27.37  
 3rd Qu.:3.668   3rd Qu.:288.0   3rd Qu.:1.0000   3rd Qu.:31.00  
 Max.   :4.928   Max.   :999.0   Max.   :1.0000   Max.   :99.00  
 height.mother   prepregnancy.weight     smoker       
 Min.   :135.0   Min.   : 39.00      Min.   :  0.000  
 1st Qu.:157.0   1st Qu.: 52.00      1st Qu.:  0.000  
 Median :163.0   Median : 57.00      Median :  0.000  
 Mean   :177.6   Mean   : 85.79      Mean   :  8.474  
 3rd Qu.:168.0   3rd Qu.: 64.00      3rd Qu.:  1.000  
 Max.   :999.0   Max.   :999.00      Max.   :999.000  
> 
> # if smoker is supposed to be a binary variable, something strange is happening
> table(bw.dt$smoker)

  0   1 999 
742 484  10 
> par(mfrow=c(3,2), mar=c(0,3,3,0))
> for (col in c("birth.weight", "gestation", "age.mother", "height.mother",
+               "prepregnancy.weight"))
+   boxplot(bw.dt[, ..col], main=col)
```

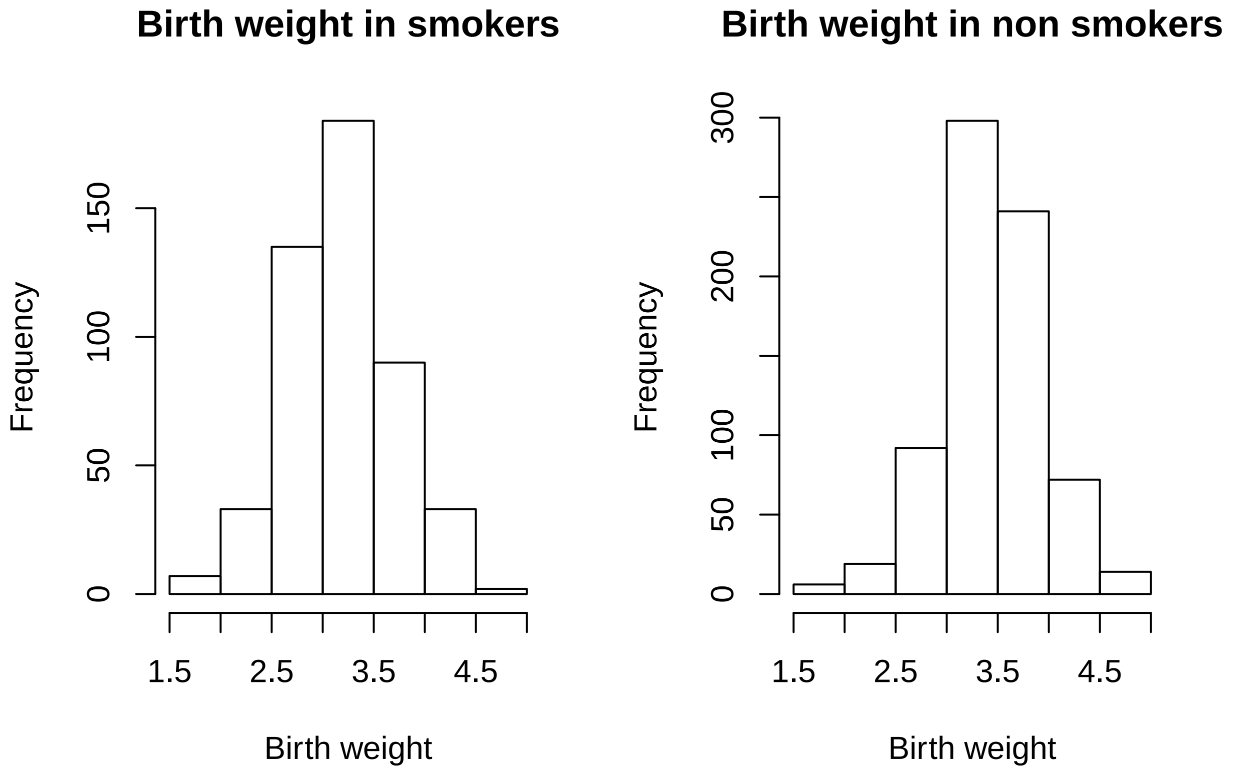


Whatever approach you use, you should notice that most variables have “outlier” values: these are actually codes for missing values that other statistical packages (such as SPSS) use instead of `NA`, and should be removed before the analysis.

```
> bw.dt <- within(bw.dt,
+               { age.mother[age.mother == 99] <- NA
+                 height.mother[height.mother == 999] <- NA
+                 prepregnancy.weight[prepregnancy.weight == 999] <- NA
+                 gestation[gestation == 999] <- NA
+                 smoker[smoker == 999] <- NA
+                })
```

Distribution of birth weights stratified by smoking status:

```
> bw.dt.smoker <- subset(bw.dt, smoker == 1)
> bw.dt.nonsmoker <- subset(bw.dt, smoker == 0)
> with(bw.dt, summary(birth.weight[smoker == 1]))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  1.624   2.856   3.220   3.195   3.528   4.564      10 
> with(bw.dt, summary(birth.weight[smoker == 0]))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  1.540   3.164   3.444   3.445   3.752   4.928      10 
> par(mfrow=c(1,2))
> with(bw.dt, hist(bw.dt.smoker$birth.weight, main="Birth weight in smokers",
+                xlab="Birth weight"))
> with(bw.dt, hist(bw.dt.nonsmoker$birth.weight, main="Birth weight in non smokers",
+                xlab="Birth weight"))
```



Percentage of babies born weighing under 2.5kg stratified by smoking status:

```
> round(with(bw.dt.smoker,
+            sum(birth.weight < 2.5) / nrow(bw.dt.smoker) * 100), 2)
[1] 8.26
> round(with(bw.dt.nonsmoker,
+            sum(birth.weight < 2.5) / nrow(bw.dt.nonsmoker) * 100), 2)
[1] 3.37
```

Association between birth weight and smoking:

```
> regr.smoking <- lm(birth.weight ~ smoker, data=bw.dt)
> coef(summary(regr.smoking))
              Estimate Std. Error    t value    Pr(>|t|)
(Intercept)  3.4453208 0.01817177 189.597395 0.00000e+00
smoker      -0.2502546 0.02892144  -8.652912 1.55385e-17
> coef(summary(regr.smoking))[2, 4] < 0.05  # the association is significant
[1] TRUE
```

Change in birth weight according to smoking: babies born from smoking mothers are on average 0.25kg lighter:

```
> coef(regr.smoking)[2]
    smoker
-0.2502546
```

Gestation length for first born children is 2.6 days longer:

```
> regr.gest <- lm(gestation ~ first.born, data=bw.dt)
> coef(regr.gest)[2]
first.born
  2.585058
> coef(summary(regr.gest))[2, 4]           # the association is significant
[1] 0.01378078
```

Association of pre-pregnancy weight with length of gestation:

```
> regr.gestweight <- lm(gestation ~ prepregnancy.weight, data=bw.dt)
> coef(summary(regr.gestweight))           # no association
                       Estimate Std. Error    t value Pr(>|t|)
(Intercept)         276.8559484 2.89784999 95.5383990 0.000000
prepregnancy.weight   0.0386393 0.04901314  0.7883458 0.430652
```

Association of birth weight with pre-pregnancy weight:

```
> regr.bw.dt <- lm(birth.weight ~ prepregnancy.weight, data=bw.dt)
> coef(summary(regr.bw.dt))
                       Estimate Std. Error   t value      Pr(>|t|)
(Intercept)         2.861157973 0.09096505 31.453375 8.379939e-159
prepregnancy.weight 0.008301632 0.00153752  5.399364  8.058039e-08
```

Association of birth weight with pre-pregnancy weight adjusted for height of the mother:

```
> regr.bw.dth <- lm(birth.weight ~ prepregnancy.weight + height.mother, data=bw.dt)
> coef(summary(regr.bw.dth))
                       Estimate  Std. Error  t value     Pr(>|t|)
(Intercept)         1.005048886 0.371120332 2.708148 6.862634e-03
prepregnancy.weight 0.004562081 0.001689754 2.699849 7.035096e-03
height.mother       0.012743199 0.002477123 5.144354 3.135476e-07
```

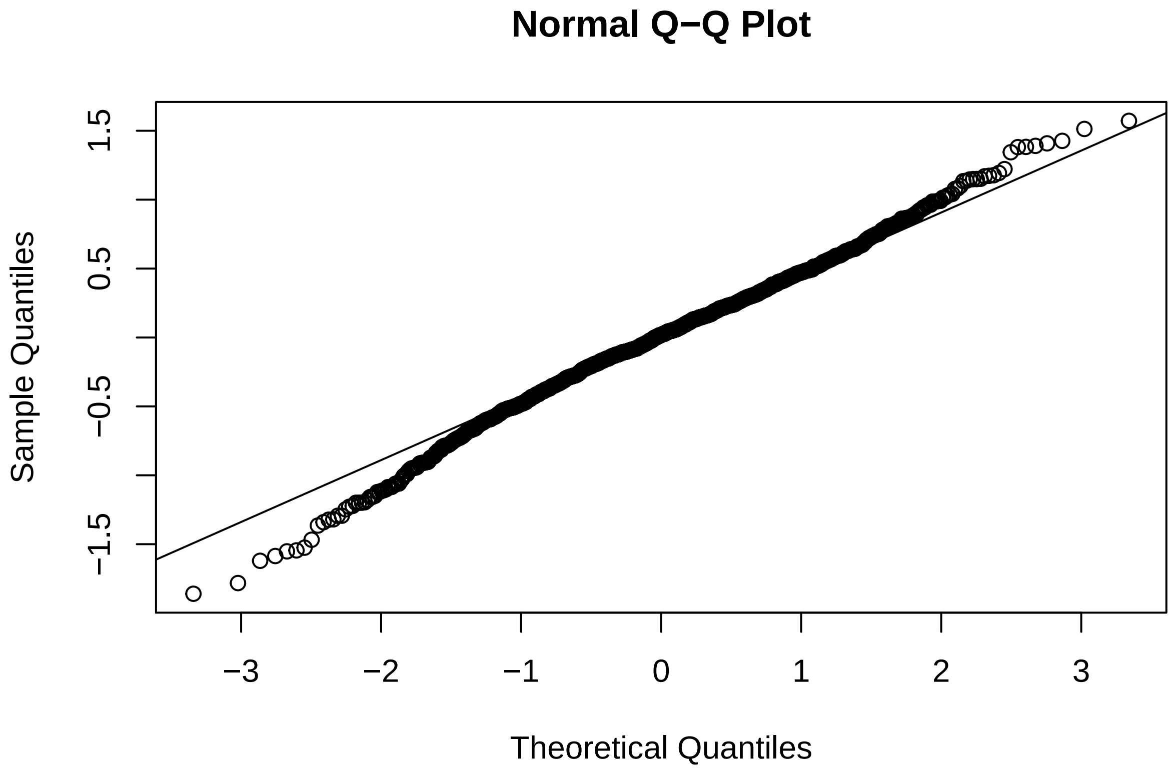
Pre-pregnancy weight is associated with birth weight, and this association is independed of height of the mother. However its effect size is roughly halved (from 0.0083 to 0.0046) after adjusting for height. Presumably this occurs because pre-pregnancy weight is explained in part by height of the mother.

Create the class as per the lab

```
> library(R6)
>
> # Class definition
> ModelSummary <- R6Class("ModelSummary", public = list(
+   model.store = list(),
+   model.summary.dt = data.table(),
+   model.thresh.set = logical(),
+
+   # The initialize (note spelling) is called by new()
+   initialize = function(model.in) {
+
+     # Input data checks
+     stopifnot(length(model.in$coefficients) >= 2)
+
+     # Initialise member variables
+     self$model.thresh.set = FALSE
+
+     # Create a data table summarising the results
+     self$model.summary.dt <-
+       data.table(summary(model.in)$coefficients, keep.rownames = TRUE)
+     print(self$model.summary.dt)
+     # Add confidence intervals around the coefficients
+     self$model.summary.dt <- cbind(self$model.summary.dt,
+           confint(model.in))
+
+     # Reformat the output table
+     self$model.summary.dt <-
+       self$model.summary.dt[,
+       .(Name = rn, Estimate = signif(Estimate, 3),
+         `95% CI` = paste0("(", signif(`2.5 %`, 3), ", ",
+       signif(`97.5 %`, 3), ")"), `P-value` = signif(`Pr(>|t|)`, 3) )]
+
+     # Keep a copy of the original model
+     self$model.store <- model.in
+   },
+
+   # Reformat significance level description
+   statsignifdt = function(thresh = 0.001) {
+     stopifnot(is.numeric(thresh))
+
+     if(!self$model.thresh.set) {
+       self$model.summary.dt[, `P-value` := ifelse(`P-value` < thresh,
+                      paste0("<", thresh), as.character(round(`P-value`, 3)))]
+       # Only allow setting of the threshold once
+       self$model.thresh.set = TRUE
+     } else {
+       cat("\nError: P-value threshold has been set previously!\n\n")
+     }
+     return(self$model.summary.dt)
+   },
+
+   # Provide a QQ plot of the model residuals
+   qqplot.residuals = function() {
+     with(self$model.store, qqnorm(residuals))
+     with(self$model.store, qqline(residuals))
+     qq.plot <- recordPlot()
+     return(qq.plot)
+   },
+
+   # The print method is called if the object is viewed
+   print = function() {
+     if(nrow(self$model.summary.dt)>1) {
+       print(self$model.summary.dt)
+     } else {
+       cat("No model data.\n")
+     }
+
+     if(!self$model.thresh.set) {
+       cat("P-value threshold not yet set. Run statsignifdt() method.\n")
+     }
+   })
+ )
>
```

Instantiate an object of class ModelSummary, then call the member method The parameter is set to 0.001 as default, so there is no need to pass a value. The default print method is called by viewing the object and finally we return a capture of the QQ plot object which can be used later in a separate output.

```
>
> mod.desc.obj <- ModelSummary$new(regr.bw.dth )
                    rn    Estimate  Std. Error  t value     Pr(>|t|)
1:         (Intercept) 1.005048886 0.371120332 2.708148 6.862634e-03
2: prepregnancy.weight 0.004562081 0.001689754 2.699849 7.035096e-03
3:       height.mother 0.012743199 0.002477123 5.144354 3.135476e-07
> mod.desc.obj$statsignifdt()
> mod.desc.obj
                  Name Estimate             95% CI P-value
1:         (Intercept)  1.01000      (0.277, 1.73)   0.007
2: prepregnancy.weight  0.00456 (0.00125, 0.00788)   0.007
3:       height.mother  0.01270  (0.00788, 0.0176)  <0.001
> plot.mod.resid <- mod.desc.obj$qqplot.residuals()
```


Normal Q−Q Plot
Sample Quantiles
1.5
0.5
−0.5
−1.5
−3
−2
−1
0
1
2
3
Theoretical Quantiles