Sample Answer Lab 01:

Handout date: Wednesday, August 26, 2020
Due date: Wednesday, September 2, 2020 submitted as Word document to eLearning's

Task 1: Equation Editor Exercise (1 pt)

Typeset the derivation of the definitional equation of the variance equation in line (1) from its computational counterpart in line (7). That is, your sequence of equations needs to be in the opposite order starting with equation (7) $s_x^2 = \frac{1}{n-1} \cdot \left(\sum_{i=1}^{n} x_i^2 - n \cdot \bar{x}^2\right)$ with $\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$ and ending at equation (1)

Make sure that your equation formatting follows exactly the one given in the screen shot below. In particular, focus on [a] the change from the *italics* equation mode to the upright text mode now in your first equation, [b] proper nesting of the employed mathematical templates, [c] the alignment of all equation at the "="-sign, and [d] the use of underbraces $\underbrace{top}_{bottom}$ in equations (3) and (4). You do not need to replicate the *explanations* and *line numbers* in red. Deviations from the formatting in the screen shot below will lead to a loss of partial points.

$$
\begin{aligned}
s_x^2 &= \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^2 \text{ with } \bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^{n} \left(x_i^2 - 2 \cdot x_i \cdot \bar{x} + \bar{x}^2\right) \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^{n} x_i^2 - \frac{1}{n-1} \cdot \sum_{i=1}^{n} 2 \cdot x_i \cdot \bar{x} + \frac{1}{n-1} \cdot \underbrace{\sum_{i=1}^{n} \bar{x}^2}_{=n \cdot \bar{x}^2} \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^{n} x_i^2 - \frac{1}{n-1} \cdot 2 \cdot \bar{x} \cdot \underbrace{\sum_{i=1}^{n} x_i}_{=n \cdot \bar{x}^2} + \frac{n}{n-1} \cdot \bar{x}^2 \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^{n} x_i^2 - \frac{n}{n-1} \cdot 2 \cdot \bar{x}^2 + \frac{n}{n-1} \cdot \bar{x}^2 \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^{n} x_i^2 - \frac{n}{n-1} \cdot \bar{x}^2
\end{aligned}
$$

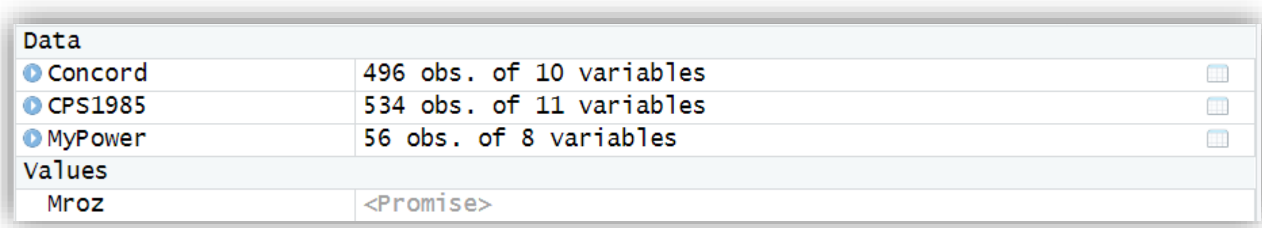
$$= \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n \cdot \bar{x}^2\right)$$

Task 2: Importing Data (0.5 pts)

Setup an R working directory and save the files **MyPower.RData** , **Concord1.sav** and **CPS1985.dbf**. into this directory.

a. Explore the **`load( )`** function and import **MyPower.RData** .
```
load('MyPower.RData')
```
b. Use a function from the library **`foreign`** to import **Concord1.sav** and save it under the name **`Concord`**.
```
library(foreign)
Concord <- read.spss('Concord1.sav', to.data.frame=TRUE)
```
c. Use a function from the library **`foreign`** to import **CPS1985.dbf** and save it under the name **`CPS1985`**.
```
CPS1985 <- read.dbf('CPS1985.dbf')
```
d. Explore the documentation of the data-frame **`Mroz`** in the library **`carData`** and link the data-frame to your R session with the **`data( )`** function.
```
data(Mroz, package="carData")
```
e. To demonstrate that everything worked as intended show a screenshot of **Global Environment**, which displays all 4 data-frames.

Mroz is promise type when first time load (for saving the memory), once you call it in code, it would be transferred to a data frame.



Data	
Concord	496 obs. of 10 variables
CPS1985	534 obs. of 11 variables
MyPower	56 obs. of 8 variables
Values	
Mroz	<Promise>

Task 3: Data-frame Basics (1.5 pts)

a. For the data-frame **`MyPower`** calculate the average daily power consumption by using the variables **`kWhBill`** and **`DaysBill`** and add the new variable to the data-frame **`MyPower`** with the variable name **`DailykWh`**. Show your R code for this calculation. (0.2 pts)
```
MyPower$DailykWh <- MyPower$kWhBill / MyPower$DaysBill
```

b. Apply the statements

`MyPowerNames <- names(MyPower)` (1)

`length(MyPowerNames)` (2)

MyPowerNames[4:6] (3)

What are these statements doing? (0.2 pts)

(1) get columns' name and save in new variable

(2) get the number of columns'

(3) get the column names for 4^{th} to 6^{th} variables

c. Apply the statement **sapply(*data-frame*, is.factor)** on the data-frame **MyPower** to evaluate the which variables are factors. What is this statement doing? Show a copy of the Console with the output of this investigation. (0.1 pts)

```
sapply(MyPower, is.factor)
SeqID     Year    Month  MinTemp  AveTemp  MaxTemp  kWhBill DaysBill DailykWh
FALSE    FALSE     TRUE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
```

C

d. Apply the **str( )** on the data-frame **MyPower**. What information about the data-frame does the **str( )** provide to you? (0.1 pts)

```
str(MyPower);
```

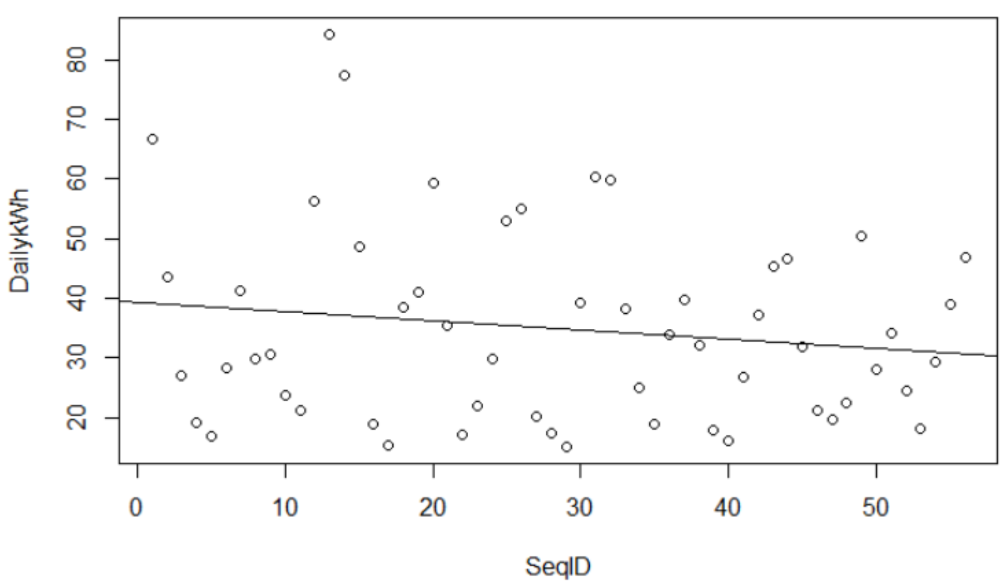
str() provides the internal structure of MyPower (shows the data type of each column)

e. What are the following statements doing? Show the plot and elaborate on the syntax of the statements. Is the power consumption over time decreasing? (0.3 pts)

```
plot(DailykWh~SeqID, data=MyPower)
abline( lm(DailykWh~SeqID, data=MyPower) )
```

Scatterplot the Daily KWH (as y) across seqID(as x), then add the fitted line from the linear regression to it.

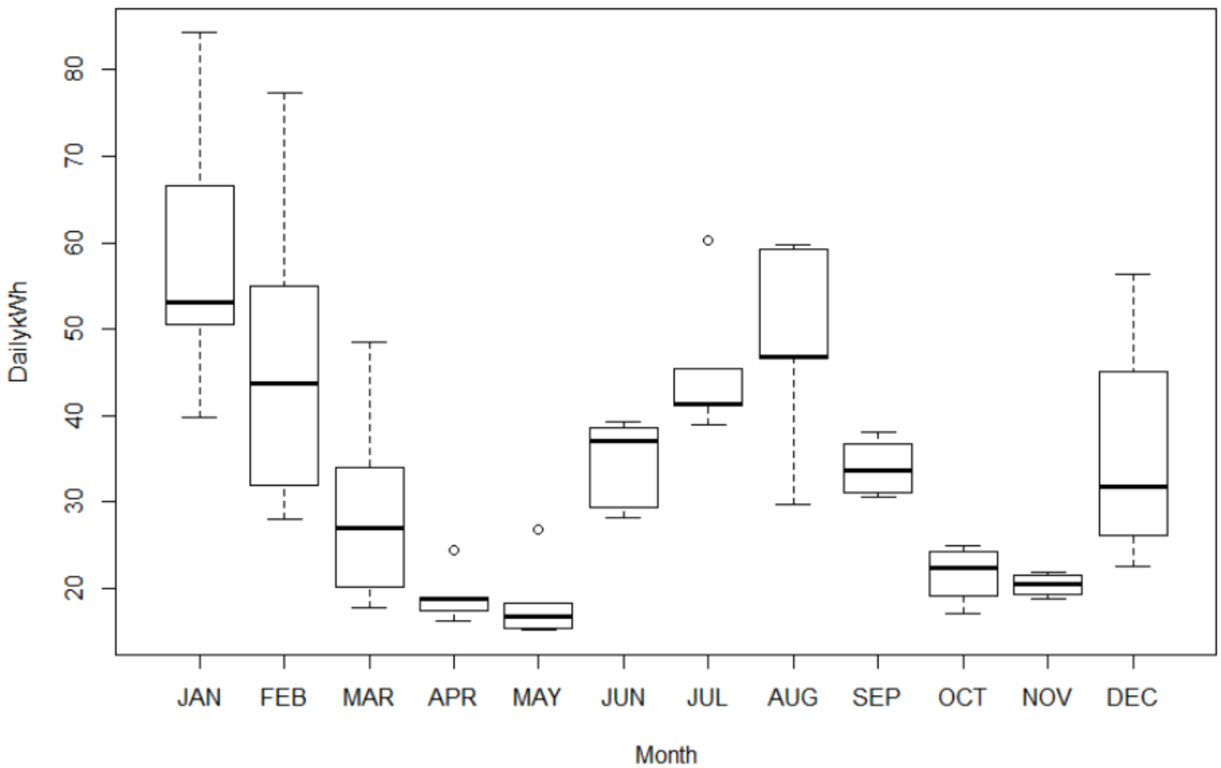
Since the SeqID follows the time order of the observations the power consumption over the years is decreasing.



f. What is the following statement doing? Show the plot and elaborate on the syntax of the statements. Why does the power consumption fluctuate over the seasons? (0.2 pts)

```
plot(DailykWh~Month, data=MyPower)
```

Since the electric heater runs in the winter and air conditioner in the summer,. in both seasons the power consumption increases.



g. Use the syntax **MyPower[*rows*, *cols*]** to select all records with the three variables **c("MinTemp","AveTemp","MaxTemp")** (alternatively you could use **MyPowerNames[4:6]**) in the month of **MyPower$Month=="JAN"**. Show the code and its output. (0.2 pts)

```
subset(MyPower[,4:6],MyPower$Month=="JAN")
MyPower[MyPower$Month=="JAN",c("MinTemp","AveTemp","MaxTemp")]
add output
```

h. Look and show at the header and the tail of the data-frame **MyPower** with the functions **head( )** and **tail( )**. (0.1 pts)

```
head(MyPower)
tail(MyPower)
what are these functions doing?
```

i. What class is the output **MetricPower** of the operation below? (0.1 pts)

```
MetricPower <- MyPower[ , c("MinTemp","AveTemp","MaxTemp", "DailykWh")
```

Data frame use class(MetricPower)

Task 4: R Basics (1 pt)

a. Depending on the input object class type functions behave differently. To see the different class-specific implementations of the generic **`summary( )`** function try the command **`methods(summary)`**.
Discuss the difference in the behavior of the **`summary( )`** function when applied to a **`data.frame`** or a **`lm`** object. (0.4 pts) (0.2 pts)
method(summary) shows all available summary methods from different packages
(1) Data-frame: get the statistical information(mean, quantile, factor level counts, NA…) about each column of a data frame
(2) Linear model: get the statistical information (residuals, coefficients, R-squared….) about this linear model

b. Explore the online help for
`?summary.lm`
`?summary.data.frame`
and discuss the optional parameters (0.2 pts)
`?summary.lm`

Arguments

`object`	an object of class `"lm"`, usually, a result of a call to `lm`.
`x`	an object of class `"summary.lm"`, usually, a result of a call to `summary.lm`.
`correlation`	logical; if `TRUE`, the correlation matrix of the estimated parameters is returned and printed.
`digits`	the number of significant digits to use when printing.
`symbolic.cor`	logical. If `TRUE`, print the correlations in a symbolic form (see `symnum`) rather than as numbers.
`signif.stars`	logical. If `TRUE`, 'significance stars' are printed for each coefficient.
`...`	further arguments passed to or from other methods.

`?summary.data.frame`

Arguments

`object`	an object for which a summary is desired.
`x`	a result of the *default* method of `summary()`.
`maxsum`	integer, indicating how many levels should be shown for `factor`s.
`digits`	integer, used for number formatting with `signif()` (for `summary.default`) or `format()` (for `summary.data.frame`). In `summary.default`, if not specified (i.e., `missing(.)`), `signif()` will *not* be called anymore (since R >= 3.4.0, where the default has been changed to only round in the `print` and `format` methods).
`quantile.type`	integer code used in `quantile(*, type=quantile.type)` for the default method.
`...`	additional arguments affecting the summary produced.

c. What are the following statements below doing when processing the vector **x** of length 6? (0.3 pts) show output of operations

```
x <- c(1,3,5,7,9,NA)
x * 2; x + 2 (1)
y <- seq(0,2, by=1); x * y (2)
z <- rep(c(1,2),3); x * z (3)
```

(1) Scalar multiplication: x[i] * 2 or x[i] + 2 (i from 1 to 6)

(2) extended multiplication extends y to the same length with x by repeating, then y[i] * x[i] (i from 1 to 6)
(3) element-wise multiplication: z[i] * x[i] (i from 1 to 6)

Note, the semicolon **;** allows to place several independent commands into one line. Look the online help up for the functions **seq( )** and **rep( )**.

```
?seq()
```

seq {base} R Documentation

Sequence Generation

Description

Generate regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

Usage

```
seq(...)

## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)

seq.int(from, to, by, length.out, along.with, ...)

seq_along(along.with)
seq_len(length.out)
```

```
?rep()
```

rep {base} R Documentation

Replicate Elements of Vectors and Lists

Description

`rep` replicates the values in `x`. It is a generic function, and the (internal) default method is described here.

`rep.int` and `rep_len` are faster simplified versions for two common cases. Internally, they are generic, so methods can be defined for them (see InternalMethods).

Usage

```
rep(x, ...)

rep.int(x, times)

rep_len(x, length.out)
```

d. Define the function **myMean( )** and apply it on **x**

```
myMean <- function(x){
  x <- na.omit(x)
  sum(x)/length(x)
}
myMean(x)
```

Compare its output to that of the standard function **base::mean( )**. Look up the help for the functions **na.omit( )** and **mean( )**. (0.2 pts)

myMean(x)	mean(x)	mean(na.omit(x))
5	NA	5

e. How does the statement **`x[c(T,T,T,T,T,F)]`** work? (0.1 pts)

```
x[c(T,T,T,T,T,F)]
[1] 1 3 5 7 9
```

Gets rid of the last element (T stands for True and show the vector value at that position. F stands for False and suppresses the vector value at that position)