FIT3152 Data analytics. Tutorial 03: Manipulating data

Pre-tutorial Activity

1 Read in the data from the file "Ped_Count_December_Long.csv" that we also used for the Week 2 pre-tutorial activities. Answer the following questions performing required data manipulations in R.

- Find the five sensor locations having the highest average pedestrian activity, and the five locations having the lowest average pedestrian activity over the month.

```
>library(dplyr)
>DEC_long = read.csv("Ped_Count_December_Long.csv)
#group by sensor location, get average counts and extract top 5 instances
>DEC_long  %>%  group_by(Sensor_Location)  %>%  summarise(Average_Count  =
mean(Count, na.rm = TRUE)) %>% slice_max(Average_Count, n =5)
```

```
# A tibble: 5 x 2
  Sensor_Location                Average_Count
  <chr>                                  <dbl>
1 Flinders.La.Swanston.St..West.         1528.
2 Southbank                              1466.
3 Town.Hall..West.                       1224.
4 Bourke.Street.Mall..North.             1040.
5 Princes.Bridge                          974.
```

```
#group by sensor location, get average counts and extract bottom 5 instances
>DEC_long  %>%  group_by(Sensor_Location)  %>%  summarise(Average_Count  =
mean(Count, na.rm = TRUE)) %>% slice_min(Average_Count, n =5)
```

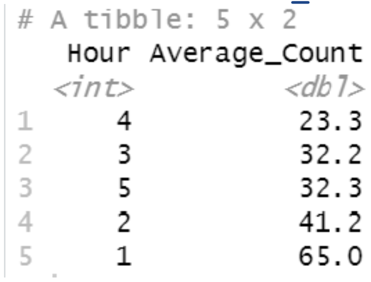
```
# A tibble: 5 x 2
  Sensor_Location                  Average_Count
  <chr>                                    <dbl>
1 Victoria.Point                            9.10
2 Pelham.St..South.                        29.1
3 Tin.Alley.Swanston.St..West.             29.7
4 Westwood.Place                           46.2
5 Harbour.Esplanade...Bike.Path            54.0
```

- Find the five one-hour periods having the highest average pedestrian activity and the five one-hour periods where pedestrian activity is at its lowest during the month.

```
#group by Hour, get average counts and extract top 5 instances
>DEC_long %>% group_by(Hour) %>% summarise(Average_Count = mean(Count, na.rm
= TRUE)) %>% slice_max(Average_Count, n =5)
# A tibble: 5 x 2
   Hour Average_Count
  <int>         <dbl>
1    17          646.
2    13          631.
3    12          614.
4    15          607.
5    14          605.
```

```
#group by Hour, get average counts and extract bottom 5 instances
```

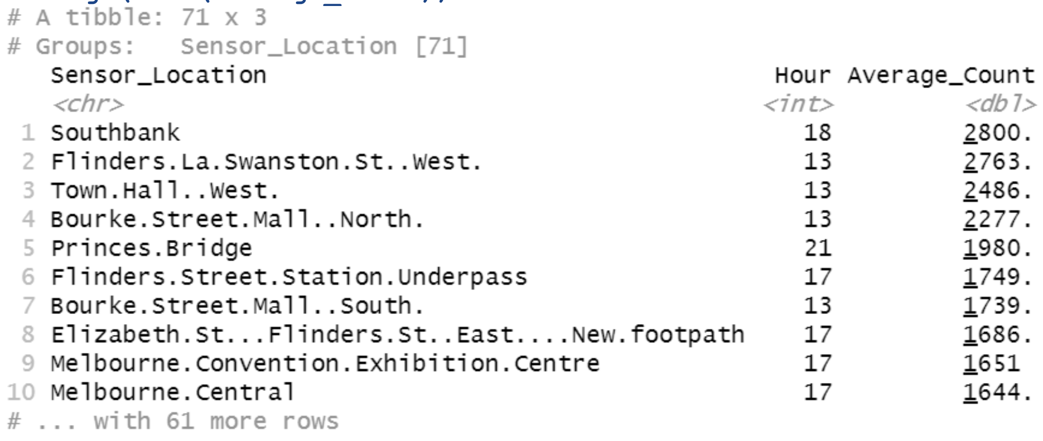
```
>DEC_long %>% group_by(Hour) %>% summarise(Average_Count = mean(Count, na.rm
= TRUE)) %>% slice_min(Average_Count, n =5)
```



```
# A tibble: 5 x 2
   Hour Average_Count
  <int>         <dbl>
1     4          23.3
2     3          32.2
3     5          32.3
4     2          41.2
5     1          65.0
```

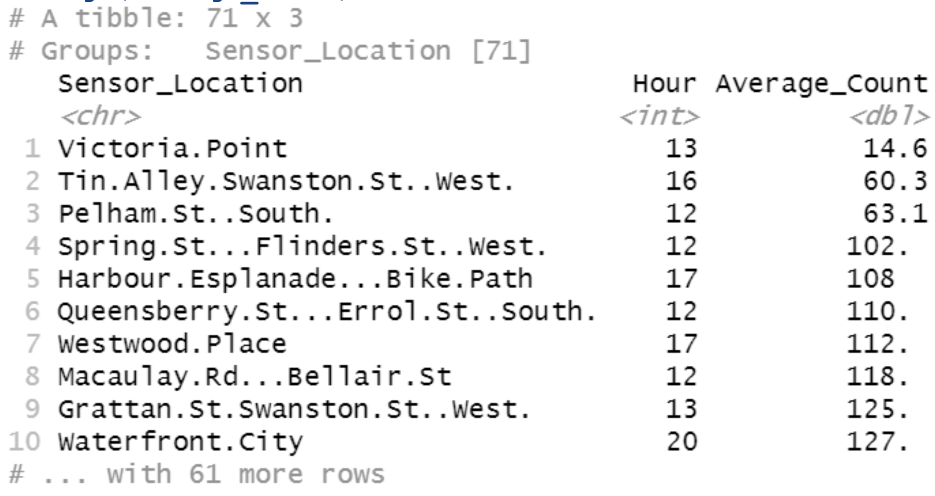
- Find the hour and location at which the average pedestrian activity is at its lowest and the hour and location at which it is at its highest over the month.

```
#group by both sensor location and hour, get average counts, extract top
instances in each -group and arrange output in descending order
>DEC_long %>% group_by(Sensor_Location, Hour) %>% summarise(Average_Count =
mean(Count, na.rm = TRUE)) %>% slice_max(Average_Count, n =1) %>%
arrange(desc(Average_Count))
```



```
# A tibble: 71 x 3
# Groups:   Sensor_Location [71]
   Sensor_Location                                    Hour Average_Count
   <chr>                                             <int>         <dbl>
 1 Southbank                                            18         2800.
 2 Flinders.La.Swanston.St..West.                       13         2763.
 3 Town.Hall..West.                                     13         2486.
 4 Bourke.Street.Mall..North.                           13         2277.
 5 Princes.Bridge                                       21         1980.
 6 Flinders.Street.Station.Underpass                    17         1749.
 7 Bourke.Street.Mall..South.                           13         1739.
 8 Elizabeth.St...Flinders.St..East....New.footpath     17         1686.
 9 Melbourne.Convention.Exhibition.Centre               17         1651
10 Melbourne.Central                                    17         1644.
# ... with 61 more rows
```

```
#group by both sensor location and hour, get average counts, extract minimum
instances in -each -group and arrange output in ascending order
>DEC_long %>% group_by(Sensor_Location, Hour) %>% summarise(Average_Count =
mean(Count, na.rm = TRUE)) %>% slice_max(Average_Count, n =1) %>%
arrange(Average_Count)
```



```
# A tibble: 71 x 3
# Groups:   Sensor_Location [71]
   Sensor_Location                    Hour Average_Count
   <chr>                             <int>         <dbl>
 1 Victoria.Point                       13          14.6
 2 Tin.Alley.Swanston.St..West.         16          60.3
 3 Pelham.St..South.                    12          63.1
 4 Spring.St...Flinders.St..West.       12         102.
 5 Harbour.Esplanade...Bike.Path        17         108
 6 Queensberry.St...Errol.St..South.    12         110.
 7 Westwood.Place                       17         112.
 8 Macaulay.Rd...Bellair.St             12         118.
 9 Grattan.St.Swanston.St..West.        13         125.
10 Waterfront.City                      20         127.
# ... with 61 more rows
```

Tutorial Activities

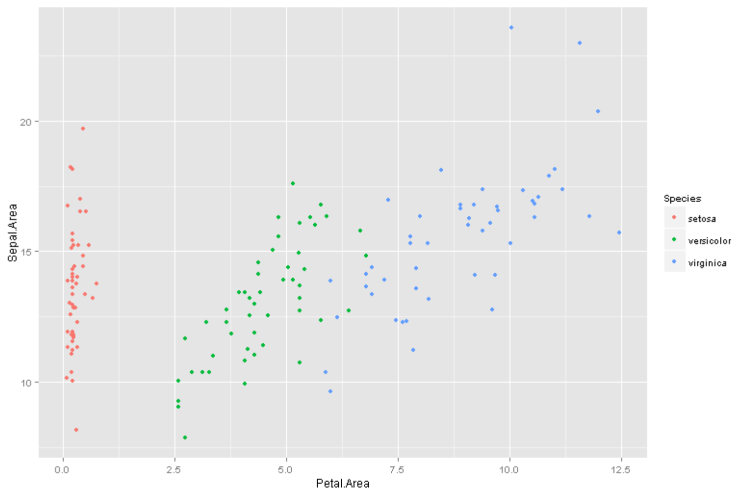
1. Try and reproduce the summary tables from Lecture 3.

2. (a) Using the 'iris' data set calculate the area of each sepal and petal using the following approximation: Area ≈ `Length * Width *` π/4. Create a new column for each area measurement.

   (b) Draw a scatterplot showing petal area vs sepal area, identifying each species.

   (c) Report the measurements of the plant in each species having (i) the largest petal and (ii) the largest sepal as a data frame. Your data frame should contain the original variables data along with the new columns showing sepal and petal area.

```
options(width = 100)
options(digits = 3)
library(ggplot2)
iris2 <- iris
area <- function(length, width) length * width * pi/4
iris2 <- cbind(iris2, area(iris2[1], iris2[2]))
colnames(iris2)[6] <- "Sepal.Area"
iris2 <- cbind(iris2, area(iris2[3], iris2[4]))
colnames(iris2)[7] <- "Petal.Area"
# print(iris2)
qplot(Petal.Area, Sepal.Area, color=Species, data = iris2)
max.sepal <- by(iris2, iris2[5], function(df) df[which.max(df[,6]),])
biggest.sepal <-  do.call(rbind, max.sepal)
max.petal <- by(iris2, iris2[5], function(df) df[which.max(df[,7]),])
biggest.petal <-  do.call(rbind, max.petal)
Biggest <- rbind(biggest.sepal, biggest.petal)
Biggest
```

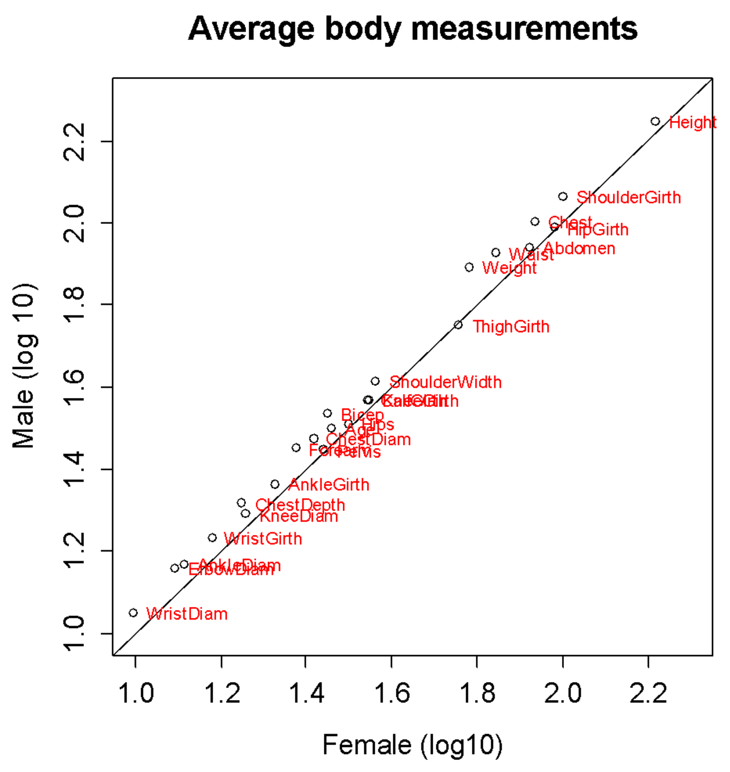


```
            Sepal.Length Sepal.Width Petal.Length Petal.Width    Species Sepal.Area Petal.Area
setosa               5.7         4.4          1.5         0.4     setosa       19.7      0.471
versicolor           7.0         3.2          4.7         1.4 versicolor       17.6      5.168
virginica            7.9         3.8          6.4         2.0  virginica       23.6     10.053
setosa1              5.0         3.5          1.6         0.6     setosa       13.7      0.754
versicolor1          5.9         3.2          4.8         1.8 versicolor       14.8      6.786
virginica1           7.7         2.6          6.9         2.3  virginica       15.7     12.464
```

3. The file "body.dat.csv" contains data from a study on the relationship between body dimensions. The study measured 500+ active individuals.

The data was obtained from http://www.amstat.org/publications/jse/jse_data_archive.htm
A related article is http://www.amstat.org/publications/jse/v11n2/datasets.heinz.html

Using the data draw the following plot using only R. The plot shows (on a log10 scale) the average measurements for males and females. Body parts have been identified and an x = y line has been overplotted.



To construct the plot, I first created a data frame using a variety of functions to: aggregate and calculate the mean; transpose the data (from rows to columns); assign column names; remove rows and convert class of the data to numeric. You will also need to investigate plotting with the base graphics, (? plot) and parameter (? par) settings.

Record the commands required to produce the plot. Make any improvements you can.

```
rm(list=ls())
body.dat = read.csv("body.dat.csv")
# By function inputs all the columns at once to the 3rd argument of by
function i.e colMeans
# We can't use mean function here as it doesn't work on a dataframe.
# It will be only usefull to take means if the first argument is a vector
instead of a dataframe
gender_means = by(body.dat[,-25], body.dat$Gender, colMeans)
```

```
# OR
# Alternatively if you want to use a function that accepts a column of a
dataframe instead of the
# entire dataframe, you can use by function together with sapply and the
funcion you want to apply
# on each column. For example;
# 4th argument i.e mean is used as argument of sapply function and
# sapply takes the dataframe and applies the mean function on each column
one-by-one
gender_means = by(body.dat[,-25], body.dat$Gender, sapply, mean)

# Turn the output of by function into a dataframe
gender_means = do.call(rbind,gender_means)
gender_means = as.data.frame(gender_means)

# Take transpose to represent every point in the graph as a row
gender_means = t(gender_means)

# Transpose is a matrix operation so it returns a matrix.
# We turn it back into a data frame here.
gender_means = as.data.frame(gender_means)

# Create scatter plot with log10 of values
plot( log10(gender_means$Female), log10(gender_means$Male), main="Average
body measurements",
     xlab="Female (log10)", ylab="Male (log 10)", xlim=c(1, 2.3),
ylim=c(1, 2.3))
# Add y=x line
abline(0,1)
# Annotate points
text(log10(gender_means$Female), log10(gender_means$Male),
row.names(gender_means), cex=0.6, pos=4, col="red")
```

4. The data file “Dunnhumby1-20.csv” is a cut down and modified set of test data from the Kaggle competition to predict when consumers would next visit a Dunnhumby supermarket and how much they would spend. See: http://www.kaggle.com/c/dunnhumbychallenge for more information. The current modified data set contains the customer ID, Date of visit, Date since last visit, and Spend for 20 customers from the test set.

   Summarize the following information for each customer in a similar format to table below. Create a data frame showing: Customer ID; average time between visits (Delta); average spent each time; correlation between delta and spend; the number of observations. *As a challenge also try and report the slope and intercept of the least squares regression of spend vs delta.*

CustomerID	AveDelta	AveSpend	CorDeltaSpend	N	RegSlope	RegInt
40	4.86	36.71	-0.003	73	-0.030	36.745
…						
…						

```
rm(list=ls())
DH = read.csv("Dunnhumby1-20.csv")
# Calculate the AveDelta and AveSpend
# again you can use df1 = by(DH[,c(3,4)], DH$customer_id, colMeans,
na.rm=TRUE) instead
df1 = by(DH[,c(3,4)], DH$customer_id, sapply, mean, na.rm=TRUE)
df1 = do.call(rbind, df1)
df1 = as.data.frame(df1)

# Calculate the CorDeltaSpend
# Up to this point, we combined the output of by function into a dataframe
using do.call(rbind,..)
# This time, we will be using as.table() function because every group in
the output of by function
# only have one  value -- basically number representing the correlation.
# We can't use rbind because rbind only works with dataframes or row of a
dataframe. -- with 2 dimensional structures --
# But we can turn individual numbers into a table structure using as.table
funciton as in this example.
df2 = by(DH, DH$customer_id, function(df) cor(df[,3], df[,4],
use="complete.obs"))
df2 = as.data.frame(as.table(df2))

# Calculate N
df3 = by(DH, DH$customer_id, nrow)
df3 = as.data.frame(as.table(df3))

# Calculate RegSlope and RegInt
df4 = by(DH, DH$customer_id, function(df) lm(df$visit_spend ~
df$visit_delta)$coefficients)
df4 = do.call(rbind, df4)
df4 = as.data.frame(df4)

# Combine all the columns into one dataframe
# We can do this using cbind as rows are alligned in all of the
df1,df2,df3 and df4. We can see that they are all
# ordered based on customer_id. If that was not the case, we might needed
to use merge() or order the dataframes
# then cbind to make sure matching rows are representing the same customer
in each dataframe.
# Also, cbind the columns in the order that it is presented in the
question.
final = cbind(df2[,1], df1[,1], df1[,2], df2[,2], df3[,2], df4[,2],
df4[,1] )

# Rename the columns
colnames(final) = c("CustomerID", "AveDelta", "AveSpend", "CorDeltaSpend",
"N", "RegSlope", "RegInt")
```

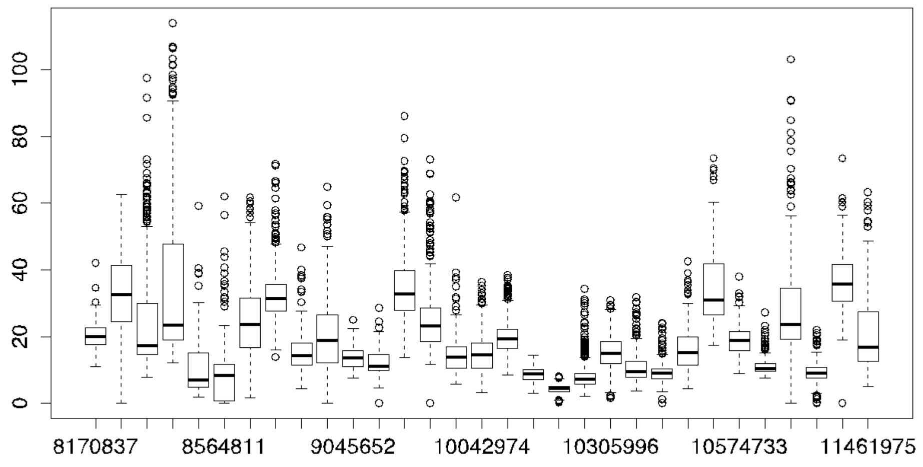
5. The data file “govhackelectricitytimeofusedataset.csv” has been created from the .txt file originally available as part of the Australian Government’s data resources. See link at: https://data.gov.au/dataset/sample-household-electricity-time-of-use-data. The file contains the smart meter records for a number of households recorded at 30 minute intervals over varying periods of time. The first few rows of the csv file are below.

CUSTOMER_KEY	End Datetime	General Supply KWH	Off Peak KWH	Gross Generation KW	Net Generation KWH
8170837	4/04/2013 11:59	0.137	0	0	0
8170837	4/04/2013 12:29	0.197	0	0	0
8170837	4/04/2013 12:59	0.296	0	0	0
8170837	4/04/2013 13:29	0.24	0	0	0
8170837	4/04/2013 13:59	0.253	0	0	0
8170837	4/04/2013 14:29	0.24	0	0	0
8170837	4/04/2013 14:59	0.238	0	0	0
8170837	4/04/2013 15:29	0.225	0	0	0
8170837	4/04/2013 15:59	0.246	0	0	0

The columns of interest are “Customer_Key” (meter), “End Datetime”, and “General SupplyKWH” (power used each 30 mins).

Using the 30 minute general supply, calculate the daily supply for each meter for every day there is data available. Because the number of records is unreliable you will also need to count the number of daily observations for each (day, meter). You should then discard any (day, meter) readings that do not have the complete number of observations.

Once you have this list you should be able to draw a box plot showing daily power consumption for each of the households. Your boxplot should look something like this:



Some functions you might try are: read.csv, by, length, sum, cbind, colnames, subset.

Extension, calculate the minimum, average and maximum daily consumption for each customer for each month you have data.

```
rm(list=ls())
df = read.csv("govhackelectricitytimeofusedataset.csv")

# Extract Date and store as a new column
df$date = as.Date(df$End.Datetime, format = "%d/%m/%Y")

# Remove unnecassary columns
df = df[,c(1,7,3)]

# Count daily number of readings for each meter
# To group the data by 2 variables, we set the second argument as list of
variables as below
daily_reading_count = by(df, list(df$CUSTOMER_KEY, df$date), nrow)
daily_reading_count = as.data.frame(as.table(daily_reading_count))

# Some days some of the meters doesn't have any reading and those returns
NA.
# Hence, we can replace them with 0.
# Select the rows with nas then assign a 0 to them as follows
daily_reading_count[is.na(daily_reading_count$Freq),3] = 0

# Calculate daily total amount of daily consumption
daily_consumption = by(df$General.Supply.KWH, list(df$CUSTOMER_KEY,
df$date), sum)
daily_consumption = as.data.frame(as.table(daily_consumption))

# Those 2 dataframes have identical number of rows and they are in the
same order.
# Because we created them using the same dataframe and same grouping
variable i.e list(df$CUSTOMER_KEY, df$date)
```

```
# Therefore, we can directly cbind their columns and rows will be
matching.

final = cbind(daily_reading_count, daily_consumption[,3])

# Now we can select the daily readings that have total 48 readings as
follows.
# P.S You might realise that all the NAs have 0 number of readings and
this is the reason they are NAs
# and will be removed now.

final = final[final$Freq==48,]

# Print number of NAs in the total consumption
sum(is.na(final[,4]))

# Rename the columns
colnames(final) = c("meter","date","reading_count", "usage")

# Plot distribution of daily consumption for each household
boxplot(final$usage ~ final$meter)


############ Q5 extension #############

# final dataset has date column as factor -- it is got converted while
creating
# final from df. Hence, we need to turn it back into date format to be
able to
# extract month information.
final$date = as.Date(final$date)

# Extract month from date object
final$month = format(final$date, "%m")

# Group data monthly instead of daily to calculate min-max-avg daily
consumption for each month and each customer
daily_min = by(final$usage, list(final$meter, final$month), min)
daily_min = as.data.frame(as.table(daily_min))

# Group data monthly instead of daily to calculate min-max-avg daily
consumption for each month and each customer
daily_avg = by(final$usage, list(final$meter, final$month), mean)
daily_avg = as.data.frame(as.table(daily_avg))

# Group data monthly instead of daily to calculate min-max-avg daily
consumption for each month and each customer
daily_max = by(final$usage, list(final$meter, final$month), max)
daily_max = as.data.frame(as.table(daily_max))

# Combine daily min-avg-max

final2 = cbind(daily_min, daily_avg[,3], daily_max[,3])

# Rename columns
colnames(final2) = c("meter", "month", "min_daily_usage",
"avg_daily_usage",  "max_daily_usage")
```