OLET1601 Notes - Analysing and Plotting Data: R

Basic Operations and Functions

```
Numerical Operators
+, -, *, /, ^, %%, %/%
Relational Operators
<, >, <=, >=, ==, !=
Logical Operators
!, &&, ||
Other Functions
- c() # vectors
- paste(x, sep=" ") or paste0()
- length(x)
- vector[-2] # removes object at 2nd index
- vector[n] <- x # replaces nth element with x, or creates index with that object
- append(vector, x, after=position)
- help(fnc)
```

- Indexing starts at 1.

Module 1

Major Functions

```
require(tidyverse)
read_delim("data", delim = " ", skip=32) # last parameter if needed
summarize(x, by)
group_by(x)
mutate(x = new_column_calc)
filter(x, expression)
```

Looking at Data

```
head(data) # first five rows
tail(data) # last five rows
print(data) # prints data and shows data type
print(data["col"]) # prints data from specified column
print(data[c(1,2,3), c("x", "y", "z")]) # prints specified columns from rows 1, 2, 3
summary(data) # provides summary information of each column e.g. mean etc.
data # outputs nice tabulation of data, but seems Jupyter Notebook specific
as.character(data)
```

```
require(tidyverse)
data <- read_delim("data.dat", delim = " ")
```

```
# Finding Max/Min
idx <- summarize(data, which.max(column))
data[as.numeric(idx), "column"] # prints all columns from row idx

# OR

data[which.max(data$column), "column"]

# Finding Mean and Standard Deviation Grouped By Column
data %>%
      group_by(column) %>%
      summarize(
                average_c = mean(column) # naming columns
                st_dev_c = sd(column)
      )

# Creating a New Column
data <- data %>%
              mutate(new_column_perc = column/column2*100)

# Filtering by Column
data <- data %>%
              filter(data, column == "x")
```

- Growth rate questions found in Quiz 1

```
gr <- ((filter(df_nsw,Year == 2015)["Total"]-filter(df_nsw,Year == 2011)
["Total"])/filter(df_nsw,Year == 2011)["Total"])
#annual growth rate
N_year <- 5
agr <- gr/N_year
print(agr)

p2020 <- agr * N_year * filter(df_nsw,Year == 2020)["Total"] + filter(df_nsw,Year ==
2020)["Total"]
print("Estimated NSW population 2016 is:")
print(p2020)
```

Module 2

Major Functions

```
require(rvest)
read_html(url) # creates list
html_nodes(html) # selects some part of html list e.g. table
html_table(node) # converts node into table
arrange(func(table)) # ordering columns
top_n(number, column) # showing n rows based off highest values in column
```

```
# Compact Table Scraping Code With the Works
data <- html_nodes(read_html("url"), "table")[[1]] %>%
              html_table(, fill=TRUE) %>%
              mutate(`Column X` = parse_number(`Column X`)) %>% # if numbers have
special characters e.g. $20
              mutate(`Column X` = as.numeric(`Column X`)) %>% # any other time
              na.omit()

# Renaming Column Name
names(table)[1] <- "column name" # renames the first column
colnames(table)[1] <- "column name"

# Creating a Frequency Table
table <- data $>$
              group_by(column) %>%
              summarise(column_name = n()) # counts frequency

# Sorting by Descending Order
table <- data $>$
              arrange(desc(first_column), desc(second_column))

# Creating a (Vertical) Bar Graph
p <- ggplot(table, aes(x = reorder(column, by, FUN = desc), y=n)) +
    geom_bar(stat='identity',fill='blue') +
    # coord_flip() for horizontal +
    labs(x = "x_label", y = "y_label") +
    theme(axis.text.x=element_text(angle=45,hjust=1,vjust=1)) # nicer formatting
print(p)
```

Module 3

Major Functions

```
ggplot2() # size parameter within aes scales size of point in relation to y
          # scale_size_continuous(guide="none") removes legend
geom_line(colour='x', linetype=1) # default linetype 1, linetype 2 for dashed (--)
geom_point(shape=x, col='x', fill='x', size = x) # where 'x' is a colour
geom_hist()
xlab('label')
ylab('label')
ggtitle('label')
scale_x_log10() # and scale_y_log10
options(repr.plot.width=x, repr.plt.height=y) # stand alone method, controls size
scale_XXX_discrete/continuous(name="MyTitle") # XXX referring to legend attribute
```

Cookbook Links

- Shape and Line Types: http://www.cookbook-r.com/Graphs/Shapes_and_line_types/ for more info
- Overall Link: http://www.cookbook-r.com/Graphs/ including legends

Finding a Subset of Code with Different Components

```
# given that kdf is a tibble dataframe

plot.df <- kdf %>%
                filter(pl_discmethod == "Transit" | pl_discmethod == "Radial Velocity")

# OR

T_group <- kdf %>%
    filter(pl_discmethod == "Transit")
RV_group <- kdf %>%
    filter(pl_discmethod == "Radial Velocity")

plot.df <- rbind(T_group, RV_group)

# Graphing

p <- ggplot(plot.df, aes(pl_orbper,pl_orbeccen, colour = pl_discmethod)) + geom_point()
# colour = pl_discmethod adjusts colour for each discovery method
```

Scatter Plot

```
# We have to define jet colors, as they are not a standard set in R
jet.colors <-
  colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan",
                     "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000"))

# see ?colours for other options e.g. topo.colors(), heath.colors(), rainbow()

#set NA to 1
```

```
kdf[is.na(kdf['pl_eqt']),'pl_eqt'] <- 1
#Colour-Coded data using planet equilibrium temperature
p <- ggplot(kdf, aes(pl_orbper,st_mass,colour=pl_eqt, size = pl_eqt/10)) + geom_point()
#adding a colorbar and its title
p <- p + scale_colour_gradientn(name='Planet Equilibrium Temperature (K)',
                                colours=jet.colors(10))
# 10 represents number of breaks in the colour ramp

p <- p + xlab('Planet Orbital Period (days)') +
ylab('Solar Mass (Solar mass)') +
ggtitle('Figure 3: Colour-Coded scatter plot')
p <- p + scale_x_log10() + scale_y_log10()
options(repr.plot.width=7, repr.plot.height=5)
print(p)
# NOTE: set the limit of the vertical (or y) axis to extend from 10^(-3)=0.001 to
10^2=100

+ scale_size_continuous(guide="none") # if wanted
```

Histograms

```
p2 <- ggplot(plot.df, aes(pl_bmassj, fill=pl_discmethod)) +
            geom_histogram(binwidth=2, position="dodge") +
            scale_fill_discrete(name="Discovery Method") +
            ylab("Count") +
            xlab("Mass (Mj)")
options(repr.plot.width = 8, repr.plot.height = 9)

# fill=x within aes for multiple variables, position="dodge" for side by side
histograms
```

Module 4

Major Functions

```
library(jsonlite)
library(lubridate)
cat("text", "text", sep="\n") # concatenating strings over two lines
str_detect(data.frame$column, regex("x", ignore_case = TRUE/FALSE)) # returns boolean
based off each row, and whether they match regex
str_subset(data.frame$column, regex("x")) # extract column
str_extract(data.frame$column, regex("x|y|z")) # extracts words
str_starts(data.frame$column, regex("x")) # returns boolean
str_ends(data.frame$column, regex("x")) # length()/nrow() is whole vector, sum() is how
many matches
```

Dealing with Twitter Data

```
tweets <- fromJSON("file") %>% as.data.frame

# Finding length of each tweet

l <- tweets %>%
        mutate(length = nchar(text)) %>% # select(length) if you just want to see data
frame
        filter(length > 100) %>% # counts how many tweets over 100 characters
        count()

# mean(l$length) also useful

# Finding tweets of a certain type

ind <- str_detect(tweets$text, regex("x|y", ignore_case=TRUE)) # if summed, only TRUE
is counted
tweets$text[ind]

tweety <- str_subset(tweets$text, "x") # only returns a vector - you lose other columns

tweety <- tweets %>%
              filter(str_detect(text, regex("x", ignore_case=T))) # all columns

tweety <- str_extract(tweets$text, regex = "x") # finds all words matching regex
              %>% na.omit()
```

Dealing with Dates

```
grouped_tweets <- Trumptweets %>%
                    group_by(days=date(Date))

tweets <- Trumptweets %>% mutate(Date = ymd_hms(created_at))

# Others to be wary of
ymd_hms()
ymd()
dmy()
mdy()
date()
year()
month()
mday()
```

Graphing with Multiple Lines

```
pl <- ggplot(grouped_tweets, aes(x=days,y=Likes)) +
          geom_line(aes(colour="Likes")) +
          geom_line(aes(x=days,y=retweets, colour = "retweets")) +
# Manually change the colour of the legend.
          scale_colour_manual(name='', values=c('retweets'='blue', 'Likes'='red')) +
          xlab("Date and time") + ylab("Daily retweets and likes")
print(pl)

# OR

grouped_tweets %>%
    gather(key="likes_faves", value = "count", retweets, Likes) %>%
    ggplot(aes(days, count, colour = likes_faves)) + geom_line() +
scale_colour_discrete(name="Likes/Faves")
```

Making New Columns and Graphing

```
number_border <- sum(str_detect(Trumptweets$text,regex("border|immigration|wall",
ignore_case=T)))
number_jobs <- sum(str_detect(Trumptweets$text,regex("job|work|unemployment",
ignore_case=T)))
number_other <-
sum(!str_detect(Trumptweets$text,regex("job|work|unemployment|border|immigration|wall",
ignore_case=T)))

#Percentage
tot_tweets <- nrow(Trumptweets)
perc_other=(number_other/tot_tweets)*100
perc_jobs=(number_jobs/tot_tweets)*100
perc_border=(number_border/tot_tweets)*100

#create again a data.frame for the plot
perc_all = data.frame(labels=c('Border Security','Others', 'Jobs'),
                      percentage=c(perc_border,perc_other,perc_jobs))


#  make a barchart
bp<- ggplot(perc_all, aes(x="",y=percentage, fill=labels))+
  geom_bar(width = 1, stat = "identity") + xlab("")
# transform to a piechart
pie <- bp + coord_polar("y", start=0)
# control the size of the plot (you don't have to do this)
options(repr.plot.width=5, repr.plot.height=5)
pie
```

Final Example

```
plot_df <- Trumptweets %>%
                mutate(iPhone = str_detect(source, "iPhone"),
                       Android = str_detect(source, "Android"),
                       Other = !str_detect(source, "iPhone|Android")) %>%
                group_by(days = date(Date)) %>%
                summarise(iPhone = sum(iPhone),
                          Android = sum(Android),
                          Other = sum(Other))

p1 <- ggplot(plot_df, aes(days, iPhone)) + geom_line(colour = "red") +
geom_line(aes(days, Android), colour = "blue") + geom_line(aes(days, Other), colour =
"green") + labs(x="Date", y=
"number of times device used")
p1
```