**Lesson 1: Introduction to R**

Link to data you will be using in this workshop: <https://raw.githubusercontent.com/AnnaWilliford/2017-11-11-UTA/gh-pages/workshop/SWC_fall2017/Data.zip>

Introduction and R basic commands: <https://www.dropbox.com/s/hifr9hn3y2v7sy6/R_commands.R?dl=0>

Subsetting and Simple R scripts commands: <https://www.dropbox.com/s/diz4an2o3c5tzob/MeanGdpPlot.R?dl=0>

Challenge:

Try to create a list named 'myOrder' that contains the

following data structures as list elements:

* -- Element 1 is a character vector of length 4 that
* lists the menu items you ordered from the restauranchicken, soup, salad, tea.
* -- Element 2 is a factor describes menu items
* as "liquid" or "solid".
* -- Element 3 is a vector that records the cost of each menu item:
* 4.99, 2.99, 3.29, 1.89.
* \*Hint: Define your elements first, then create a list with them.

Challenge 1 answer:

menuItems <- (c("chicken", "soup", "salad", "tea"))

menuType <- factor(c("solid", "liquid", "solid", "liquid"))

menuCost <- (c(4.99, 2.99, 3.29, 1.89))

myOrder <- list(menuItems, menuType, menuCost)

#some more basic R commands:

    getwd()

    #- this will let you know which directory are you in right now- current wokring directory

    in a path format e.g.

    "/Users/farahshamma/Desktop/SWC\_fall2017/R\_intro"

    setwd()

    #- let's you set your current working directory (wd)

    Provide the path inside the parentheses as the argument to the setwd function e.g.

    setwd("/Users/farahshamma/Desktop/SWC\_fall2017/R\_intro")

    # you can also add vectors: e.g.

        v1<-c(1:3, 45)

        v2<-c(1:4)

* v3 <- v1+v2

         \* note that you must have same number of elements in both the vectors

         That is your object length must be equal to perform additions

    Thanks

    Farah

**##################################     Notes on the factor() function in R: Farah     ##########################**

#In R, factors are special vectors used to represent categorical data.You can create a categorical variable using the factor function.

#Suppose, there is a data for 4 observations of eyecolor of males and females, means there are  2 categories here, male and female.

#Let's create a categorical variable 's' with this data that has 4 observations, the first one for male, other 3 for females. #Let's designate Male and Female by M and F respectively.

s<-factor(c("M", "F", "F", "F"))

s

 #output:

     [1] M F F F

     Levels: F M

#This means your categorical variable s has 4 observations for levels ( or categories) F and M.

str(s)

output:

#Factor w/ 2 levels "F","M": 2 1 1 1

#This means it is a categorical variable that has 2 levels or categories- F and M. #Now alphabetically F comes first and M comes later. So, this output shows you the levels in alphabetical order. So, F is level 1 and M is level 2 here. But You entered the category F in the second, third and fourth position after M in your data, right? So here, it shows you that your eyecolor observations for 'level 1 F' is in position 2, 3 and 4 and the observation of eyecolor for 'level 2 M' is in the first position (index). Thus 2 1 1 1.

#Try this?

fac <- factor(c("Cat", "Dog", "Dog"))

fac

str(fac)

#You can try this with the gapminder.txt data

myData<-read.table("gapminder.txt", header = TRUE)

View(myData)

 #what do you see in the 1st and 2nd column? Many countries and continents? Try to find out how many categories are there in your data.

 str(myData)

 #Do you see the country variable has 142 levels/categories? Alphabetically the 1st observation is Afghanistan which is also the 1st observation position wise in the data.

 #Do you see the continent has 5 levels? Now here Africa comes first of the 5 alphabetically in the output but all these 3 3 3... mean that the observations for Africa are after the observations for 2 other continents in the data.

str(myData$continent)

levels(myData$continent)

cont <- factor(myData$continent)

str(cont)

#you can also create and/or view individual categories in a big data this way

####################################################################################

Challenge:

Anser the following questions about the 'myData' object:

* 1. What is the overall object structure? What function will you use? str(myData)
* 2. Can you tell the data type of the elements in each column?typeof(myData)
* 3. Can you extract the 3rd and 5th column of the dataset?myData [,c(3,5)]   myData [,3]
* 4. Can you extract the list of countries in this dataset? ### Hint: use unique(). ### unique(myData$country)     unique(myData[,1])
* 5. Can you get a part of this dataset that inclues information about Sweden? myData[myData$country="Sweeden",]
* 6. Can you extract all countries for which life expectancy is below 70? myData[myData$lifeExp<70,]
* 7. Can you make a new column that contains population size in units of millions of people?         myData$PopM <- myData$pop/10^6

####### CHALLENGE #######

Write a script to calculate mean gdpPerCapita for African and European

countries.

Try to make a barplot to display your results

#### Hint : ?mean ?barplot

#### Answer:

##########  This is  MeanGdpPlot.R script #############

* #read data into R
* myDataFull<-read.table("gapminder.txt", header = TRUE)
* #What about logical operators?
* #summary(myDataFull)
* #select information about Africa
* Afri<-myDataFull[myDataFull$continent=="Africa",]
* #calculate the mean
* Afri.Mean <- mean(Afri$gdpPercap)
* #Do the same for Europe
* Europe<-myDataFull[myDataFull$continent=="Europe",]
* Euro.Mean <- mean(Europe$gdpPercap)
* #Store the mean values in a vector
* Afri.Euro.Mean <- c(Afri.Mean, Euro.Mean)
* #plot Meangdp
* png("Afri.Euro.Mean.png")  #open png device to write your plot to
* continents.names <- c("Africa","Europe")
* barplot(Afri.Euro.Mean, names.arg = continents.names, col = c("#a8ddb5","#43a2ca"), xlab = "Continents", ylab = "Mean GDP per Capita")
* dev.off()

###############

================================

Linux Shell

Make sure that GitBash is installed. You should be able to open Gitbash window

Instructions on the Software Carpentry website:

<https://annawilliford.github.io/2017-11-11-UTA/>

Bash History link:

<https://www.dropbox.com/s/zkhdirkymawlp4u/History.txt?dl=0>

$ ls Desktop/SWC\_fall2017

Challenge 1

Remember, we made a SWC\_fall2017 folder in the beginning of workshop and moved our Datafolder there. Can you now list files in SWC\_fall2017 directory from your current directory?

ls Desktop/SWC\_fall2017

(From Desktop): ls SWC\_fall2017

$ ls Desktop/SWC\_fall2017

$ ls Desktop/SWC\_fall2017

Challenge 3

Next we will work with files from the Data folder that you moved to SWC\_fall2017 folder in the beginning of this workshop. Please move Data folder to unix\_shell folder. We will keep all the files for Linux lesson in the unix\_shell folder from now on. Make sure you understand the directory structure of SWC\_fall2017 before we continue.

Here is my link to the commands coming in this Linus Lession 2

<https://1drv.ms/t/s!AiorwA8Oz7lMlakFObCO90z1HvcjWw>

-r flag:

    -you can use it with rm command and cp command

    e.g. cp -r or rm -r

    -it is used to delete all files or copy all files under a folder altogether,

    which means you can delete or copy a folder if you use -r after rm or cp command

    -It refers to recursive, meaning probably it goes to that folder, recursively copies or deletes files under the folder, and thus deletes or copies the entire folder

    Please let me know if that is clear. :)

    James's History

  143  clear

  147  pwd

  148  cd ./

  149  wc Data/gapminder.txt

  150  wc -help

  151  head -n 10 Data/gapminder.txt

  152  tail -n 10 Data/gapminder.txt

  153  cut -f1 Data/gapminder.txt

  154  head -n 10 Data/gapminder.txt

  155  cut -f1 Data/gapminder.txt > CountryList.txt

  156  ls

  157  edit CountryList.txt

  158  cat CountryList.txt

  159  sort CountryList.txt > CountryListSorted.txt

  160  ls

  161  cat

  162  cat

  163  cat CountryListSorted.txt

  164  uniq CountryListSorted.txt > CountryList-uniq.txt

  165  ls

  166  rm CountryList-uniq.txt

  167  uniq CountryListSorted.txt > CountryList\_uniq.txt

  168  ls

  169  sort -u CountryList.txt > CountryList\_uniq.txt

  170  wc -l CountryList\_uniq.txt > CountryCount\_gapminder.txt

  171  edit CountryCount\_gapminder.txt

  172  cat CountryCount\_gapminder.txt

  173  cut -f1,3 Data/gapminder.txt > CountryYear.txt

  174  head -n 10 CountryYear.txt

  CHALLANGE 4

What country had the highest “LifeExp” in 2002?

Use gapminder.txt as an input file and generate Country\_HighestLifeExp.txt as your ONLY output file.

Hint: you can accomplish this by using grep, cut, sort and tail but you might want to look up help pages for some of these commands…

 ##This is  PlotLifeExp.R script

#read data into R

myDataFull<-read.table("gapminder.txt", header = TRUE)

#myDataFull is a dataframe - check

#select information about Canada

Canada<-myDataFull[myDataFull$country=="Canada",]

#Canada is a new dataframe

png("Canda.png")

#plot lifeExp

plot(Canada$year, Canada$lifeExp, col="blue", type="l")

dev.off()

export PATH=$PATH: "path to R"

$ export PATH=$PATH:"C:\Program Files\R\R-3.4.2\bin"

**### Day 2 ###**

\*\*\* Keep notes as a class here. \*\*\*

*Session 1: R Programming*

<https://1drv.ms/f/s!AiorwA8Oz7lMlakbj3Mr5XH1qQaNtQ>

Challenge 1

Goal : Wrapped function calls.

As we’ve seen in our print statements, we can use paste or paste0 to concatenate strings.

* Write a function called fence that takes two parameters called original and wrapper and returns a new string that has the wrapper character at the beginning and end of the original.

Example function call and output:

fence('name', '---')

---name---

fence <- function(orginal, wrapper) {

    result <- paste0(wrapper, original, wrapper)

    result

}

GetGDPCountry.R script

# This script computes the GDP for a country using the gapminder data

# clear all variables

rm(list=ls())

# location of the data

filename <- "../Data/gapminder.txt"

# read the data file

gapminder <- read.table(filename, header = TRUE)

# extract gdpPercap from the gapminder data, using a specific country

getAverageGdpPerCapita <- function(country) {

  selectedCountryData <- gapminder[gapminder$country == country, "gdpPercap"]

  mean(selectedCountryData)

}

gdpUSA <- getAverageGdpPerCapita("United\_States")

gdpCanada <- getAverageGdpPerCapita("Canada")

gdpMexico <- getAverageGdpPerCapita("Mexico")

print(paste('GDP of USA is', gdpUSA))

print(paste('GDP of Canada is', gdpCanada))

print(paste('GDP of Mexico is', gdpMexico))

-----------------------------------------------------------------------------

southAmericanCountries <- c('Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador', 'Paraguay', 'Peru', 'Uruguay', 'Venezuela')

-----------------------------------------------------------------------------

Calling R scripts from the command line

Just like we can call other scripts and functions from our R programs, we can call an R script from the command line. We use the Rscript program to do this.

Rscript filename.R

If Rscript is not on your system’s path, you can invoke Rscript using it’s full path name.

Try it out:

script.R:

print('This is a simple R script')

On the command line:

Rscript script.R

Even if you do your data analysis in a different programming language, you can still use R’s plotting capabilities (which we will learn more about in the afternoon session). So if the majority of your work is in another programming language, you can still make use of R to do parts of the analysis, or only to make publication quality plots.

We will write a script that will produce a barplot from a given datafile. In the previous example, script.R prints a fixed message and has no way of getting any information from the shell. Rscript allows you to send arguments to the R script by specifying them after the file name:

Rscript filename.R argument\_1 argument\_2 ...

These arguments can be accesed from within the R script by using the commandArgs() function. By default the list will include the name of the R executable, the script file name, and several switches. Using commandArgs(FALSE) only presents us with the arguments starting from argument\_1 onwards.

Updated script.R

countryList <- commandArgs(TRUE)

# location of the data

fileName <- '../Data/gapminder.txt'

# read gapminder data

gapminder <- read.table(fileName, header=TRUE)

getAverageGdpPerCapita <- function(country) {

  # extract gdpPercap from the gapminder data for the specified country.

  selectedCountryData <- gapminder[gapminder$country == country, 'gdpPercap']

  mean(selectedCountryData)

}

averagedGdp <- sapply(countryList, getAverageGdpPerCapita)

barplot(averagedGdp)

print(averagedGdp)

We now write a shell script that is going to call this method on a list of countries.

Create a file makePlots.sh

countries="Canada Belgium"

Rscript cmdScript.R $countries

Run it from bash:

bash makePlots.sh

The generated plots are saved in PDF format in the file Rplots.pdf.

\*\*\* Keep notes as a class here. \*\*\*

*Session 2: Data Visualization w/ R using ggplot2*

Commands can be viewed at following link:

<https://www.dropbox.com/s/dr2qw8zhzkulnu3/r_viz_ggplot.R?dl=0>

ggplot2 cheatsheet

* <https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>