**Getting Started With Rgui**

**Things to Note**

* Console Box is where we run code and where results are shown
* R editor/Script is where we can store and save syntax (can also run syntax from here)
  + Useful in that we can edit, save, and store code for specific analyses
  + To open file > “new script” or “open script”
* Graphics window – graphs and other images appear here
* Many functions can be split onto multiple lines
  + However for some functions this causes issues – thus it is sometimes better to leave one function on one line
* To comment start a line with “#”
  + In Rgui you can only comment one line at a time
  + In R studio to mass comment – select/highlight all lines that you want to turn to comment and press ctrl shift c
* Typically R works by functions reading arguments giving output
  + Function typically goes before parentheses and argument is inside parentheses
  + Ex – setwd("C:\\Users\\John Schumacher\\Documents\\stats consulting\\R example")
    - Function is setwd – this tells the program that we are about to set the location to retrieve and store files on the computer
    - "C:\Users\John Schumacher\Documents\stats consulting\R example" is the file path or argument that we are using for are function
  + Think of the function as a recipe and the arguments as the ingredients
* Often we use functions and arguments to create objects
  + Objects = anything created in R (variable, collection of variables, output, etc)
  + Objects come before the function and both are separated by “<-“
    - object <- function(argument)
  + to view a object just type the object name and run it as code
* Capalization does matter
* Missing values are represented by NA
* Impossible values are represented by NaN
* You will often

**R Working Directory (WD)**

* Location (usually a folder) on computer where R automatically stores and retrieves files to and from
* If you don’t set this then you must use full path names in R
  + If we set it, then any file in that location doesn’t need the full path

**Non-Syntax Steps**

1. Make sure the console is selected
2. Click file
3. Click “Change dir…”
4. Find the proper folder

**Syntax**

setwd("C:\\Users\\John Schumacher\\stats consulting\\R example")

# notice how I use 2 \\ instead of 1 \ this is because R is weird

# could also use / instead of \

# “getwd()” tells you the location of your WD if you are unsure

**Packages R**

* R has only so many built in commands and functions, thus we often need packages to do more than the defaults allow us to do
* Packages in short are programs that we install that use basic r language to create more sophisticated functions that we can then use in our analyses to speed things up

**Installing Package**

install.packages("openxlsx")

#openxlsx is the name of the package we are installing

#openxlsx allows us to load and write excel files

#once you have installed a package you don’t need to do it again on that comp

**Loading Packages**

library("openxlsx")

#library is a function that loads a package onto R workspace for us to use it

**Creating Dataset from Scratch**

* Easiest way is to create individual variables first then combine them into a dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| name | birth\_date | job | friends | alcohol | income | neuroticism |
| Leo | 1-Jan-60 |  | 5 | 10 | 20000 | 10 |
| Martin | 24-May-69 | 1 | 2 |  | 40000 | 17 |
| Andy | 21-Jun-73 | 1 | 0 | 20 | 35000 | 14 |
| Paul | 16-Jul-70 | 1 | 4 | 5 | 22000 | 13 |
| Graham | 10-Oct-49 | 1 | 1 | 30 | 50000 | 21 |
| Carina | 5-Nov-83 | 2 | 10 | 25 | 5000 | 7 |
|  | 8-Oct-87 | 2 | 12 | 20 | 100 | 13 |
| Doug | 16-Sep-89 | 2 | 15 | 16 | 3000 | 9 |
| Mark | 20-May-73 | 2 | 12 | 17 | 10000 | 14 |
| zoe | 28-Mar-90 | 2 | 17 | 18 | 10 | 13 |

**Creating String Variables**

name<-c("Ben", "Martin", "Andy", "Paul", "Graham", "Carina", NA, "Doug", "Mark", "Zoe")

#notice how as with SAS we place string variables in “” parentheses, this is often very important

#”name” is what we labeled our variable or object

#notice how are missing name is represented as NA

**Creating Date Variable**

birth\_date<-as.Date(c("1960-01-01", "1969-05-24", "1973-06-21", "1970-07-16", "1949-10-10", "1983-11-05", "1987-10-08", "1989-09-16", "1973-05-20", "1990-03-28"))

# as.Date is a function that allows us to enter in a date format, and then r reads it as a date

**Creating Numeric Variables**

job <- c(NA, 1, 1, 1, 1, 2, 2, 2, 2, 2)

friends<-c(5,2,0,4,1,10,12,15,12,17)

alcohol<-c(10, NA,20,5,30,25,20,16,17,18)

income<-c(20000,40000,35000,22000,50000,5000,100,3000,10000,10)

neuroticism<-c(10,17,14,13,21,7,13,9,14,13)

**Combining Variables and Creating Dataset**

lecturerData<-data.frame(name, birth\_date, job, friends, alcohol, income, neuroticism)

#lecturerData is new datafram/dataset

**Making a Numeric Variable a Factor Variable (i.e. categorical variable)**

* lets change job variable
  + 1 = lecturer
  + 2 = student

**Simply changing numeric variable to a factor variable without changing labels**

lecturerData$job <- factor(lecturerData$job)

#$ symbol tells us we are changing or working with the variable on the right side of the $ symbol and that the variable comes from the dataframe specified on the left of the $ sign

#factor function simply says we are changing a variable to be a factor variable

**changing numeric variable to a factor variable with changing labels**

* if we hadn’t run the previous code we would run . . .

lecturerData$job <- factor(lecturerData$job, levels = c(1,2), labels=c("lecturer","student"))

#here we are making the factor variable first then changing levels 1 and 2 to I and II

* since we did run the previous code we will run . . .

levels(lecturerData$job) <- list(lecturer="1", student="2")

#note this only works if all levels are present in list, if not they will be replaced with NA

**# to change one specific level we use . . .**

levels(lecturerData$job) [levels(lecturerData$job) =="lecturer"] <- "teacher"

**Creating a factor variable from scratch**

job <- factor(c(NA, "teacher", "teacher", "teacher", "teacher", "student", "student", "student", "student", "student"))

**Permanently Rename a variable**

lecturerData$crazy <- lecturerData$neuroticism

#this line copies neuroticism variable and names new variable crazy

lecturerData$neuroticism <- NULL

#this line deletes old neuroticism variable

**#or we can run the following code in one line . . .**

names(lecturerData)[names(lecturerData)=="neuroticism"] <- "crazy"

**Dropping a Variable**

lecturerData\_delete <- subset(lecturerData, select = -c(crazy) )

#notice how we created a new dataset called “lecturerData\_delete”

#we could have simply deleted the variable from the original dataset by using the original dataset name “lecturerData” ex code lecturerData <- subset(lecturerData, select = -c(crazy) )

**Applying a mathematical transformation to a numeric variable**

**Creating new dataset and simply transforming existing**

lecturerData\_transformed <- lecturerData

#this copies old dataset to new dataset where we will work

attach(lecturerData\_transformed)

lecturerData\_transformed$crazy <- crazy/10

detach(lecturerData\_transformed)

**Using existing dataset and creating new transformed variable from old**

attach(lecturerData)

lecturerData$crazyTENS <- crazy/10

detach(lecturerData)

**Deleting a Row**

**Deleting one specific Row**

lecturerData\_deleteROW <- lecturerData [-8,]

# we are deleting row 8

# it is the – symbol doing the deleting here

**Deleting all rows where job = “student” and income = 10 (aka Zoe)**

lecturerData\_deleteRow2<- lecturerData[!(lecturerData$job=="student" & lecturerData$income==10),]

# ! exclamation mark here means not, thus we are saying use lecturerData but not variables we specified

**Replacing a Value**

lecturerData\_replace <- lecturerData

lecturerData\_replace$friends[lecturerData\_replace$friends==12] <- 24

# here we are changing any friends value of 12 to 24

**Creating new string variable from numeric variable (using**

attach(lecturerData)  
lecturerData$too\_much\_alcohol[alcohol <= 20] <- "Not\_Drunk"  
lecturerData$too\_much\_alcohol[alcohol > 20 & alcohol <= 25] <- "Drunk"  
lecturerData$too\_much\_alcohol[alcohol > 25] <- "Too\_Drunk"  
detach(lecturerData)

**Creating new numeric variable from numeric variable**

attach(lecturerData)

lecturerData$too\_much\_alcohol2[alcohol <= 20] <- 0

lecturerData$too\_much\_alcohol2[alcohol > 20 & alcohol <= 25] <- 1

lecturerData$too\_much\_alcohol2[alcohol > 25] <- 2

detach(lecturerData)

**Listing Variable Names**

names(lecturerData)

**Getting Descriptives**

install.packages("pastecs")

library("pastecs")

lecturerData\_descriptiveOutput <- stat.desc(lecturerData)

**Exporting Output to .txt File**

sink("descritptives\_output.txt" , append = "true" )

# r will create file if it doesn’t exist

# append true = add on to file & false = overwrite file

print(lecturerData\_descriptiveOutput)

# this line tells us what R object to print to .txt file

sink()

#this ends sink

**Exporting Dataset to .CSV File**

write.table(lecturerData, "lecturerData\_CSVdata.csv", sep=",", col.names=NA)

**Exporting Dataset to Excel**

install.packages("openxlsx")

library("openxlsx")

write.xlsx(lecturerData, "lecturerData\_EXCEL.xlsx", sheetName = "Sheet1", col.names = TRUE, row.names = TRUE, append = FALSE)

#r will create file if it doesn’t exist

# sheet1 is name of sheet in excel you are saving to

# col.names = True/False – True if writing col names to excel file; False if not

# row.names = True/False – True if writing row names to excel file; False if not

# append = True/False – True if adding to in progress file; False if adding data to blank excel

# once ran, you can open .xlsx file

**Importing Data from Excel**

library("openxlsx")

lecturerData\_Excel <- read.xlsx("lab1.xlsx", sheet = 1, colNames = TRUE)

#lab1 is name of excel we are importing in

# sheet = 1 just means we are exporting from sheet named 1, if we changed the sheet name we would change 1

#colNames = TRUE just means that the first row are variable names