DSA 8020 R Session 0: A Quick Introduction to R

Whitney Huang

December 27, 2021

Contents

This `R` session is modified from a R tutorial from Doug Nychka [link].

Basic

What is in the workspace?

```
ls()
```

```
## character(0)
```

Where is my R working directory?

```
getwd()
```

```
## [1] "/Users/whitneyhuang/Desktop/Desktop - mass-mini19-huang/Teaching/DSA/DSA8020/22 Spring"
```

What is in my working directory?

```
dir()
```

```
## [1] "BoulderTemperature.RData"   "BT.RData"
## [3] "DSA8020_22sp_RCode0.pdf"    "DSA8020_22sp_RCode0.Rmd"
## [5] "DSA8020_22sp_Syllabus.pdf" "DSA8020_RCode1.Rmd"
## [7] "DSA8020_Slides0.pdf"        "notes0.pdf"
```

"*c*" combines sets of numbers (or datasets)

```
x <- c(2, 3, 20)
```

Note: R is case sensitive. Type "X" in R console and then click Enter to see what happen

Now recheck workspace

```
ls()
```

```
## [1] "x"
```

```
# print out x
x
```

```
## [1]  2  3 20
```

Reassign "x" to another name

```
x2 <- x
ls()
```

```
## [1] "x"  "x2"
```

Remove "x" and create another object "x3"

```
rm(x)
x3 <- c(3, 4, 5)
```

Question: how would you combine “x2” and “x3” to make a new data set ?

```
#give a try here
```

Arithemtic

Add numbers in `R`

```
A <- 2
B <- 10
Y <- A + B

A <- c(2, 3, 4)
B <- c(10, 100, 1000)
Y <- A + B
Y # note that the numbers have been added row by row like a spread sheet.
```

```
## [1]   12  103 1004
```

Some other operations

```
2^4
```

```
## [1] 16
```

```
2 * (1 + 4)
```

```
## [1] 10
```

```
sqrt(81)
```

```
## [1] 9
```

```
exp(2)
```

```
## [1] 7.389056
```

Generating a sequence

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
-5:5
```

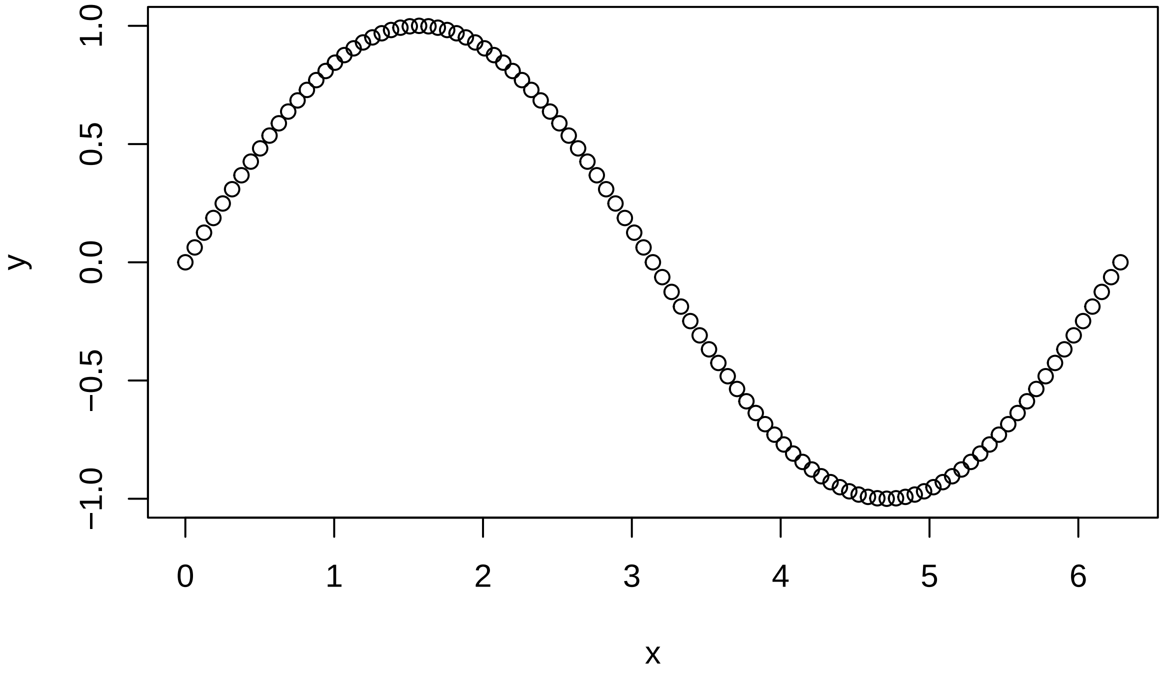
```
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

Question: How would you generate the values in order 5 to 1?

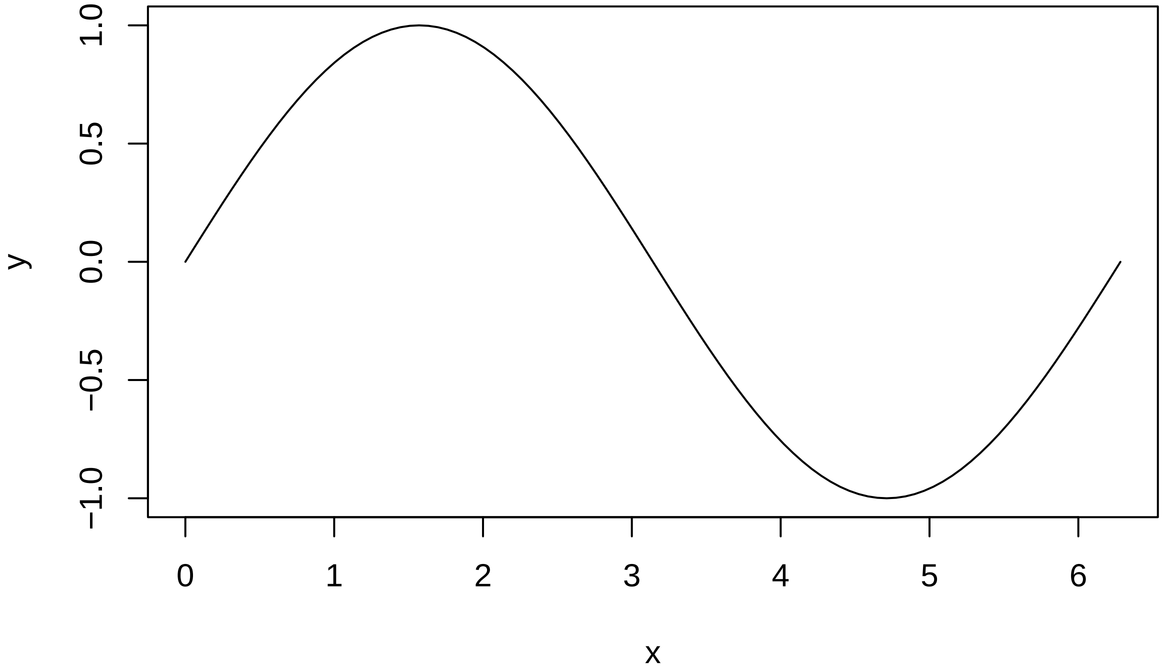
Generating sin wave

```
x <- 0:100
# hundred values between 0 and 2*pi
```

```
x <-  2 * pi * (x / 100)
y <- sin(x)
# plot the sin wave
plot(x, y)
```



```
# change the plot to connect points with a line instead of points
plot(x, y, type = "l")
```



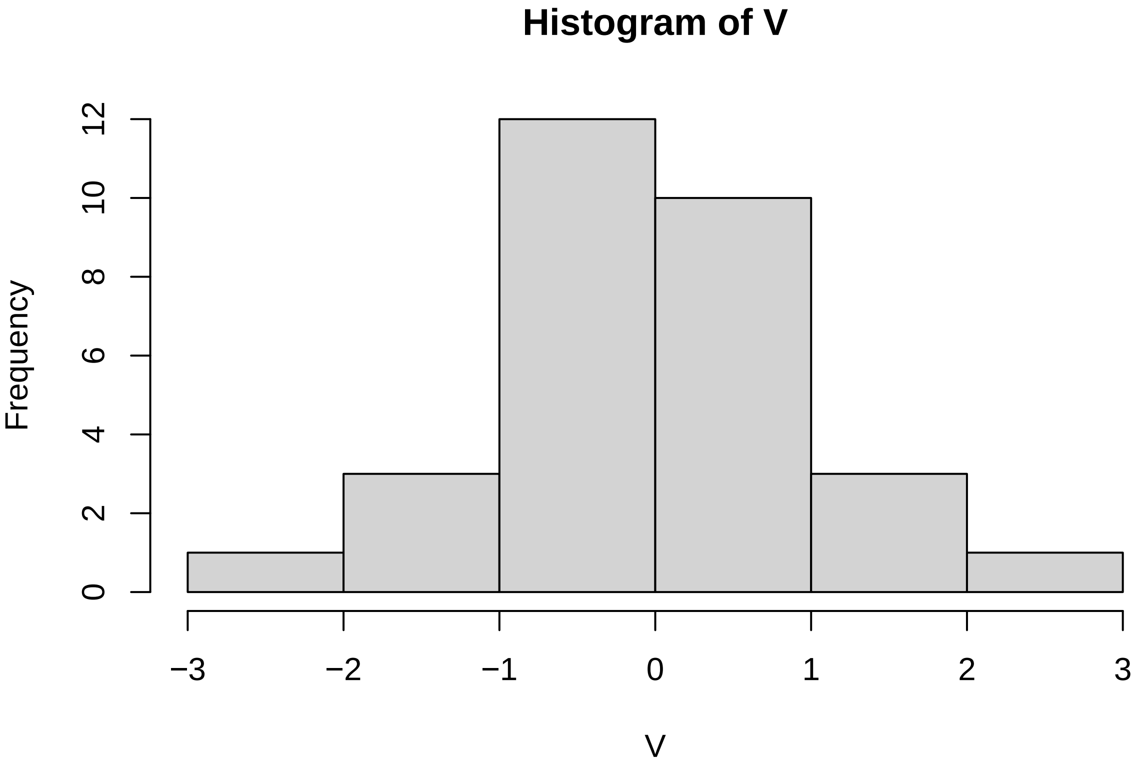
```
# another way of creating the x
x <- seq(0, 2 * pi, length.out = 101)
```

Use `R` to generate random values

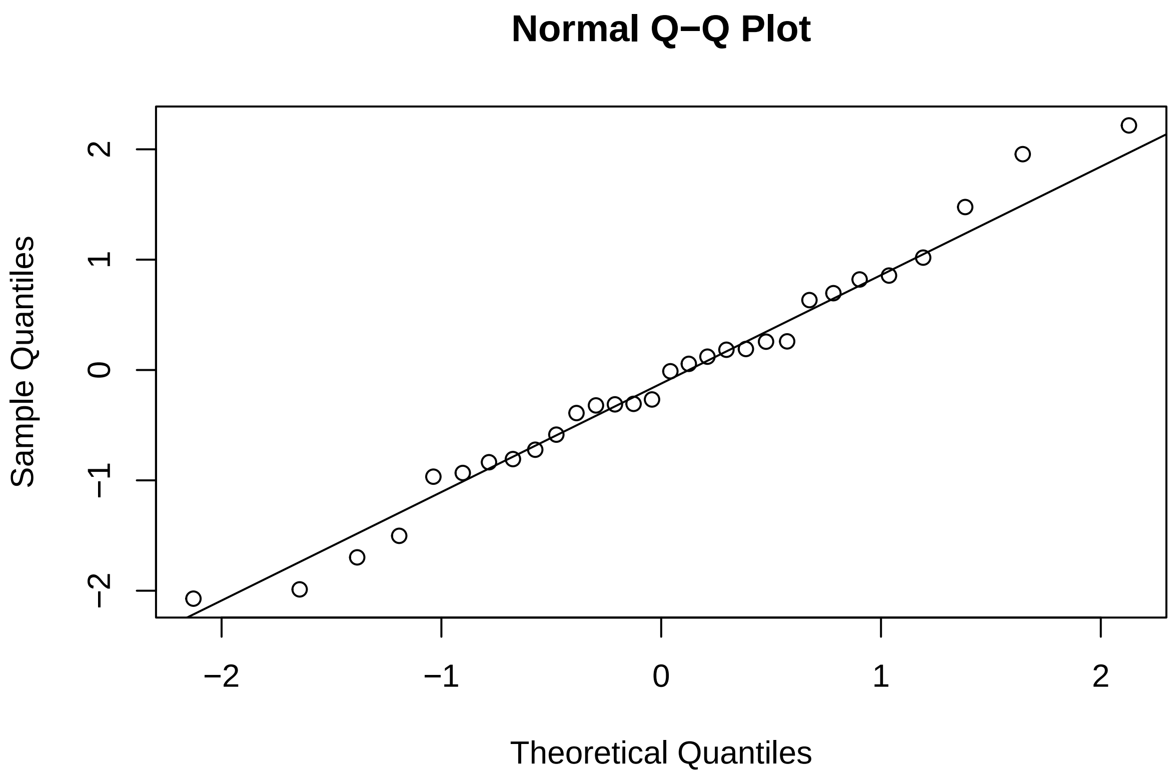
```
# generating 10 random numbers between 0 and 1
U <- runif(10)
U
```

```
##  [1] 0.614253061 0.687874304 0.009322416 0.736307974 0.878694482 0.611269986
##  [7] 0.216010526 0.920248689 0.457006954 0.534641163
```

```
# generating 30 random numbers from a standard normal distriburion
V <- rnorm(n = 30, mean = 0, sd = 1)
hist(V)
```



```
qqnorm(V); qqline(V)
```



Subsetting

Load a data set

```
load("BT.RData")
# copy for easy typing
BT <- BoulderJuneTemperature$Temp
BAll <- BoulderJuneTemperature
head(BT); head(BAll, 10)
```

```
## [1] 65.51667 68.58333 69.21667 68.58333 70.91667 64.25000
```

```
##    Year     Temp
## 1  1984 65.51667
## 2  1985 68.58333
## 3  1986 69.21667
## 4  1987 68.58333
## 5  1988 70.91667
## 6  1989 64.25000
## 7  1990 69.95000
## 8  1991 66.56667
## 9  1992 62.90000
## 10 1993 64.66667
```

Print the first 10 values

```
BT[1:10]
```

```
##  [1] 65.51667 68.58333 69.21667 68.58333 70.91667 64.25000 69.95000 66.56667
##  [9] 62.90000 64.66667
```

An indicator for all values over 70

```
ind70 <- BT > 70
ind70
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
## [25] FALSE FALSE FALSE FALSE  TRUE FALSE
```

```
# temperatures with values over 70
BT[ind70]
```

```
## [1] 70.91667 70.05000 70.36667 71.56667 74.13333
```

```
# the years with values over 70
BAll$Year[ind70]
```

```
## [1] 1988 1994 2002 2006 2012
```

Question: How many years exceed 70 degrees?

Working with these data as a matrix

```
dim(BAll)
```

```
## [1] 30  2
```

```
# This is the first row and first column
BAll[1, 1]
```

```
## [1] 1984
```

```
# first row
BAll[1,]
```

```
##   Year     Temp
## 1 1984 65.51667
```

```
# first column
BAll[, 1]
```

```
##  [1] 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998
## [16] 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
```

```
# column with Year (this is also column 1)
BAll[, "Year"]
```

```
##  [1] 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998
## [16] 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
```

```
# second column (could also use "Temp" to refer to this)
BAll[, 2]
```

```
##  [1] 65.51667 68.58333 69.21667 68.58333 70.91667 64.25000 69.95000 66.56667
##  [9] 62.90000 64.66667 70.05000 62.33333 66.93333 66.40000 62.68333 64.85000
## [17] 67.40000 68.85000 70.36667 62.83333 62.70000 65.41667 71.56667 67.65000
## [25] 66.01667 63.16667 66.35000 67.56667 74.13333 69.82000
```

```
# rows 10 through 20
BAll[10:20,]
```

```
##    Year     Temp
## 10 1993 64.66667
## 11 1994 70.05000
## 12 1995 62.33333
## 13 1996 66.93333
## 14 1997 66.40000
## 15 1998 62.68333
## 16 1999 64.85000
## 17 2000 67.40000
## 18 2001 68.85000
## 19 2002 70.36667
## 20 2003 62.83333
```

Exercise: Plot the temperatures by year

Apply Functions in R

apply functions

1. a family of functions in R which allow you to repetitively perform an action on multiple chunks of data
2. run faster than loops and often require less code.

Let's take a look at some examples

Load the Boulder temperature data set into R

```
load("BoulderTemperature.RData") # monthly mean temperatures
dim(BoulderTemperature)
```

```
## [1] 118  12
```

```
# check out first row
BoulderTemperature[1,]
```

```
##      jan feb mar apr      may   jun      jul      aug      sep      oct
## 1897 NaN NaN NaN NaN 60.25806 64.65 70.56452 69.06452 66.81667 52.41935
##           nov      dec
## 1897 41.86667 30.40323
```

```
#extract 1991 - 2010
yr <- rownames(BoulderTemperature)
index <- which(yr %in% 1991:2010)
tempData <- BoulderTemperature[index,]
# check this out
tempData
```

```
##           jan      feb      mar      apr      may      jun      jul      aug
## 1991 29.61290 40.96429 42.69355 47.73333 58.22581 66.56667 70.46774 69.11290
## 1992 35.28571 40.56897 43.25806 54.21667 59.06452 62.90000 68.14516 66.29032
## 1993 28.33871 30.58929 42.30645 47.56667 57.91935 64.66667 69.46774 67.37097
## 1994 35.50000 32.10714 43.82258 47.61667 60.80645 70.05000 71.14516 71.01613
## 1995 34.67742 38.28571 42.11290 44.51667 50.85484 62.33333 70.48387 73.96774
## 1996 29.70968 37.67241 37.62903 50.41667 58.87097 66.93333 71.45161 69.48387
## 1997 31.37097 32.96429 45.53226 42.81667 57.17742 66.40000 71.40323 68.85484
## 1998 36.50000 36.39286 38.54839 46.50000 58.61290 62.68333 72.75806 70.37097
## 1999 36.20968 42.10714 45.98387 44.55000 55.58065 64.85000 73.33871 69.30645
## 2000 36.41935 41.06897 42.85484 51.23333 60.98387 67.40000 74.66129 73.03226
## 2001 32.93548 32.32143 40.75806 50.63333 58.40323 68.85000 75.14516 71.85484
## 2002 33.11290 35.98214 37.27419 53.01667 56.16129 70.36667 76.90323 71.30645
## 2003 40.20968 32.08929 43.66129 50.60000 57.35484 62.83333 75.66129 72.79032
## 2004 35.40323 33.67241 48.17742 49.18333 59.96774 62.70000 69.16129 66.40323
## 2005 35.43548 37.87500 41.96774 48.40000 57.72581 65.41667 75.04839 69.70968
## 2006 40.66129 33.67857 39.38710 53.88333 60.95161 71.56667 74.37097 71.56452
## 2007 27.22581 34.58929 47.56452 47.81667 58.00000 67.65000 74.75806 73.56452
## 2008 31.62903 36.10345 40.75806 47.80000 57.03226 66.01667 75.01613 69.62903
## 2009 38.19355 39.33929 44.20968 47.30000 59.30645 63.16667 69.53226 69.48387
## 2010 33.01613 30.05357 42.37097 48.75000 53.90323 66.35000 72.45161 72.41935
##           sep      oct      nov      dec
## 1991 61.56667 52.14516 37.05000 35.48387
## 1992 64.41667 53.87097 34.00000 29.77419
## 1993 59.01667 48.61290 35.61667 35.41935
## 1994 64.83333 50.69355 36.53333 36.08065
## 1995 60.38333 51.33871 44.95000 36.24194
## 1996 60.76667 53.01613 40.58333 36.46774
## 1997 64.01667 52.66129 37.86667 33.83871
## 1998 67.20000 50.32258 44.01667 32.16129
## 1999 58.48333 51.90323 47.98333 36.91935
## 2000 63.10000 49.59677 31.31667 31.20968
## 2001 65.00000 53.83871 43.85000 34.98387
## 2002 64.06667 45.75806 40.26667 36.58065
## 2003 60.50000 57.38710 38.91667 36.35484
## 2004 62.85000 51.85484 39.66667 36.45161
## 2005 66.35000 53.09677 44.93333 33.30645
## 2006 58.40000 50.98387 43.36667 35.29032
## 2007 64.43333 55.17742 44.86667 30.06452
## 2008 60.90000 51.80645 46.20000 31.09677
## 2009 63.10000 44.46774 43.76667 26.66129
## 2010 66.55000 54.77419 39.76667 37.19355
```

The "*apply*" function

```
# means  by rows of this table
byYear <- apply(tempData, 1, FUN = mean) # by rows, 1 = first index
byYear
```

```
##     1991     1992     1993     1994     1995     1996     1997     1998 
## 50.96857 50.98260 48.90762 51.68375 50.84554 51.08345 50.40858 51.33892 
##     1999     2000     2001     2002     2003     2004     2005     2006 
## 52.26798 51.90642 52.38118 51.73297 52.36322 51.29098 52.43878 52.84208 
##     2007     2008     2009     2010 
## 52.14257 51.16565 50.71062 51.46661
```

```
# means by columns
byMonth <- apply(tempData, 2, FUN = mean) # by cols, 2 = second index
byMonth
```

```
##      jan      feb      mar      apr      may      jun      jul      aug
## 34.07235 35.92127 42.54355 48.72750 57.84516 65.98500 72.56855 70.37661
##      sep      oct      nov      dec
## 62.79667 51.66532 40.77583 34.07903
```

Writing Functions in R

Finding the inter quartile range (IQR)

```
# 75% quantile
BT75 <- quantile(BT, .75)
#Question: Find the interquartile range 75% - 25% quantiles
# and check this against the built in function
IQR(BT)
```

```
## [1] 4.4125
```

Building your own function

Here is a function that adds the squares of two numbers. It has three parts, the *calling arguments*, the *body* where you do the work and then *returning any results*.

```
myFun <- function(a, b){
    result <- a^2 + b^2
    return(result)
}

test1 <- myFun(2, 3)
test1
```

```
## [1] 13
```

```
test2 <- myFun(1:5, 11:15)
test2
```

```
## [1] 122 148 178 212 250
```

Note that the “a”, “b” and result are only used inside the function and do not appear in your workspace. Also since the body is normal R code, this works for vectors automatically.

Building your own IQR function

```
myIQR <- function(y){
  IQR <- quantile(y, .75, names = FALSE) - quantile(y, .25, names = FALSE)
  return(IQR)
}
myIQR(BT)
```

```
## [1] 4.4125
```

Modify this function to work with NAs

```
test <- c(BT, NA)
myIQR <- function(y, na.rm = FALSE){
            IQR <- quantile(y, .75, names = FALSE, na.rm = na.rm) -
                   quantile(y, .25, names = FALSE, na.rm = na.rm)
            return(IQR)
}
myIQR(test, na.rm = T)
```

```
## [1] 4.4125
```

Adding warning message

```
myIQR <- function(y, na.rm = FALSE){
# an example of adding a warning
    if(na.rm){
        warning("NAs removed from the data")
    }
            IQR <- quantile(y, .75, names = FALSE, na.rm = na.rm) -
                   quantile(y, .25, names = FALSE, na.rm = na.rm)
            return(IQR)
}
myIQR(test, na.rm = T)
```

```
## Warning in myIQR(test, na.rm = T): NAs removed from the data
```

```
## [1] 4.4125
```