**CSE 578: Data Visualization (2020 Spring)**

**LAB 1 - (R) 1/23**

1. **R or Python?**

* We know that R and Python both are open source programming languages.
* The major purpose of using R is for statistical analysis, on the other hand Python provide the more general approach to data science.
* R is having the most powerful communication libraries that are quite helpful in data science.
* In addition, R is equipped with many packages that are used to perform the data mining and time series analysis.
* If you use R and you want to perform some object-oriented function than you can’t use it on R.
* On the other hand, Python is not suitable for statistical distributions.
* R is more functional, Python is more object-oriented.
* R has more data analysis built-in, Python relies on packages.
* R has more statistical support in general
* It’s usually more straightforward to do non-statistical tasks in Python.

**2. R as a calculator:**

> 1+2

[1] 3

> 3^2

[1] 9

#Try built-in functions

> exp(2)-log(100)

[1] 2.783886

# Define a compound function

> sqrt(abs(-2))

[1] 1.414214

> a<-1

> b=2

> (a+b)^2

[1] 9

#Define a function z=f(x,y)

> f<-function(x, y) z<-(y^2-x^2)\*pi

> print(f(1,2))

[1] 9.424778

#See what variables you have

> ls()

[1] "a" "A" "b" "B" "f"

#Remove a and b from working space

> rm(a,b) # Remove all with “rm(list=ls())”

> ls()

[1] "A" "B" "f"

**3. Create Vectors in R:**

> A<-c(2,3,5,7,11)

> B<-seq(100,108, by=2)

> B

[1] 100 102 104 106 108

> c(A,B)

[1] 2 3 5 7 11 100 102 104 106 108

> A+B

[1] 102 105 109 113 119

> airports<-c("JFK","LGA","EWR","SFO")

> length(airports)

[1] 4

> airports[4] #How about airports[-4] ?

[1] "SFO"

> airports[1:3]

[1] "JFK" "LGA" "EWR"

> airports[c(2,4)]

[1] "LGA" "SFO"

**Q1: What are the differences among vector, matrix, data frame, and factor?**

**Your answer should provide a concrete example in codes and annotations.**

Solution:

1. **Vector:**  
   A **vector** is what is called an array in all other programming languages except R — a collection of cells with a fixed size where all cells hold the same type (integers or characters or reals or whatever). The elements of a vector must all have the same mode or data type. You can have a vector consisting of three character strings (of mode character) or four integer elements (of mode integer), but not a vector with a mix of integer elements and character string elements.  
   For e.g. x and y are vectors:
   1. x <- c(5,7,9,7)
   2. y <- c("a", "b", "c")
2. **Matrix:**A **matrix** is a two-dimensional vector (fixed size, all cell types the same). All columns in a matrix must have the same mode(numeric, character, etc.) and the same length. The general format is:

mymatrix <- matrix(vector, nrow=r, ncol=c, byrow=FALSE, dimnames=list(char\_vector\_rownames, char\_vector\_colnames))

For e.g.   
# generates 5 x 4 numeric matrix   
y<-matrix(1:20, nrow=5,ncol=4)

1. **Data Frame:**A **data frame** is a matrix-like data structure in R, with two-dimensional rows and columns where each column may have a different mode or data type. For instance, one column may consist of numbers, and another column might have character strings.For e.g.
2. kids <- c("John", "Mary")
3. ages <- c(5, 7)
4. d <- data.frame(kids, ages, stringsAsFactors = F)
5. d
6. kids ages
7. 1 John 5
8. 2 Mary 7

Here, d is a simple data frame consisting of a character vector "kids" and a numeric vector "ages".

1. **Factor:**An R **factor** might be viewed simply as a vector with a bit of extra information that consists of a record of the distinct values in that vector called levels. The motivation for factors comes from the notion of nominal / categorical variables in statistics. These values are non-numerical in nature, corresponding to categories such as male / female or high / medium / low, although they may be coded using numbers. For e.g.
2. x <- c(5,7,9,7)
3. xf <- factor(x)
4. xf
5. [1] 5 7 9 7
6. Levels: 5 7 9

The distinct values in xf: 5, 7 and 9 are the levels here. Let's take a look inside:

1. str(xf)
2. Factor w/ 3 levels "5","7","9": 1 2 3 2
3. unclass(xf)
4. [1] 1 2 3 2
5. attr(,"levels")
6. [1] "5" "7" "9"

The core of xf here is not (5, 7, 9, 7) but rather (1, 2, 3, 2) i.e. (level-1, level-2, level-3, level-2). So, the data has been recoded by level.

R will treat factors as nominal variables and ordered factors as ordinal variables in statistical proceedures and graphical analyses. You can use options in the **factor( )** and **ordered( )** functions to control the mapping of integers to strings (overiding the alphabetical ordering).

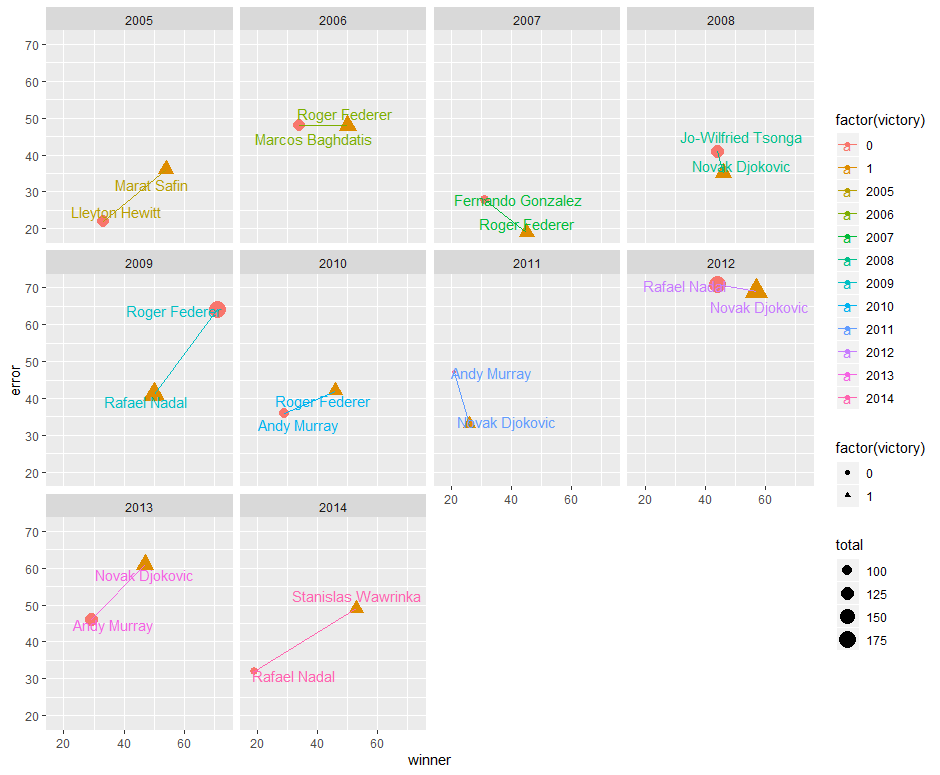
Source: <https://www.statmethods.net/input/datatypes.html> , <https://jamesmccaffrey.wordpress.com/2016/05/02/r-language-vectors-vs-arrays-vs-lists-vs-matrices-vs-data-frames/> , <https://www.quora.com/What-is-the-difference-between-Vector-Factor-and-DataFrames-in-R> , <http://adv-r.had.co.nz/Data-structures.html> ,

**4. Exploratory Analysis:**

p <- ggplot(sample, aes(winner, error))

p + geom\_point(aes(shape=factor(victory),size=total,colour=factor(victory)))+ geom\_text\_repel(aes(colour=factor(year),label=player), position = position\_jitter(width=5, height=1.5) ) + facet\_wrap(~year) + geom\_line(aes(colour=factor(year)))

Code Explaination:



**Q2: Come up with another assumption and vision the outcome may be in a similar**

**comparative small multiples chart?**

Ans:

Assumption: High return, high breaks, high total = wins the match

p <- ggplot(sample, aes(return,break.))

p + geom\_point(aes(shape=factor(victory),size=total,color=ifelse((victory==0),"lose","win")))+ geom\_text\_repel(aes(label=player, color=ifelse((victory==0),"lose","win")), position = position\_jitter(width=5, height=1.5),vjust=1.8) + facet\_wrap(~year) + scale\_shape\_manual(labels = c("Lose", "Win"),values=c(13,21)) + labs(title = "Tennis Finals\n", color = "Color\n", shape="Shape\n") + geom\_line()

**A close up of a map

Description automatically generated**

**5. Deeper Analysis (Modeling):**

#only look at the big 3 players

p <- ggplot(big3, aes(factor(year), winner1))

p + geom\_boxplot() + facet\_grid(~player1)+ geom\_jitter(height = 0)

A close up of a white background

Description automatically generated

# distribution and density

p + geom\_violin() +facet\_grid(~player1)+ geom\_jitter(height = 0)

A group of people

Description automatically generated

# regression line

ggplot(big3, aes(x=total1, y=winner1, size=total1, color=player1)) + geom\_point()+geom\_smooth(method=lm, se=F)

A close up of a map

Description automatically generated

# regression + prediction

ggplot(big3, aes(x=total1, y=winner1, size=total1, color=player1)) + geom\_point()+ geom\_smooth(method=lm)

A close up of a map

Description automatically generated

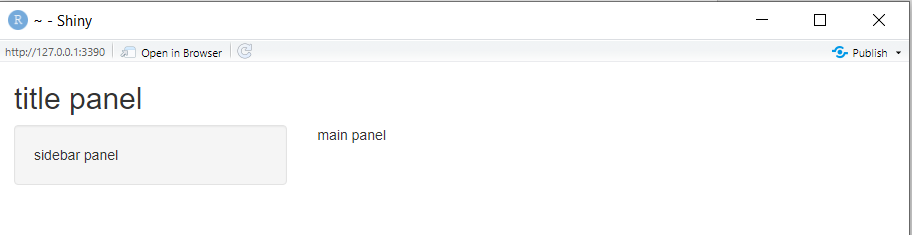
**6. Interactive Visualization with R: Shiny**

Example 1:

ui <- fluidPage(titlePanel("title panel"),sidebarLayout(sidebarPanel("sidebar panel"),mainPanel("main panel")))

server <- function(input,output) {}

shinyApp(ui=ui,server=server)



Example 2: Create two files ui.R and server.R and save them in a new folder “newdir”

#ui.R file code

library(shiny)

shinyUI (

pageWithSidebar

(

#Specify Application title

headerPanel ("Differences Between Champions and Runnerups"),

#Sidebar with controls to select the variable to plot against match result

sidebarPanel

( selectInput ("variable", "Variable:", list("Winner" = "winner",

"Error" = "error",

"Total" = "total")

),

# Add an optional input: to specify whether outliers should be displayed

checkboxInput ("outliers", "Show outliers", FALSE)

),

#Show the caption and plot of the requested variable against match result as outputs

mainPanel (h3(textOutput("caption")),

plotOutput("tennisPlot")

) #

) # pageWithSidebar end

) #UI end

#server.R file code

library(shiny)

shinyServer(function(input, output)

{

# Construct the formula for the title of the plot

formulaText <- reactive(

{ paste(input$variable, "against match results") }

)

# Return the formula text for printing as a caption

output$caption <- renderText (

{ formulaText() }

)

#Generate a boxplot of requested variable against result and include outliers if requested

output$tennisPlot <- renderPlot(

{ #Construct a formula for the plot

boxplot(as.formula(paste(input$variable, "~victory" )),

data = sample,

outline = input$outliers,

col="orange")

}

)

}

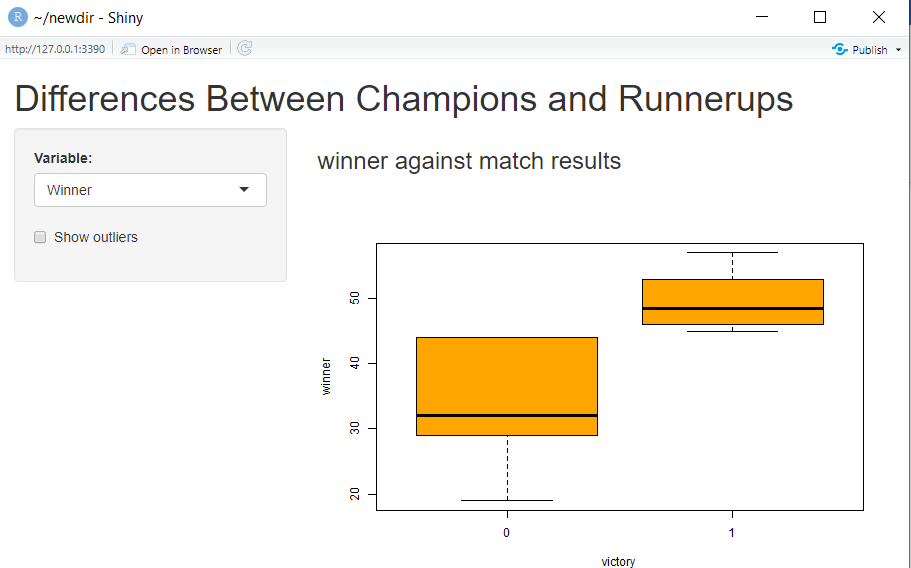
)

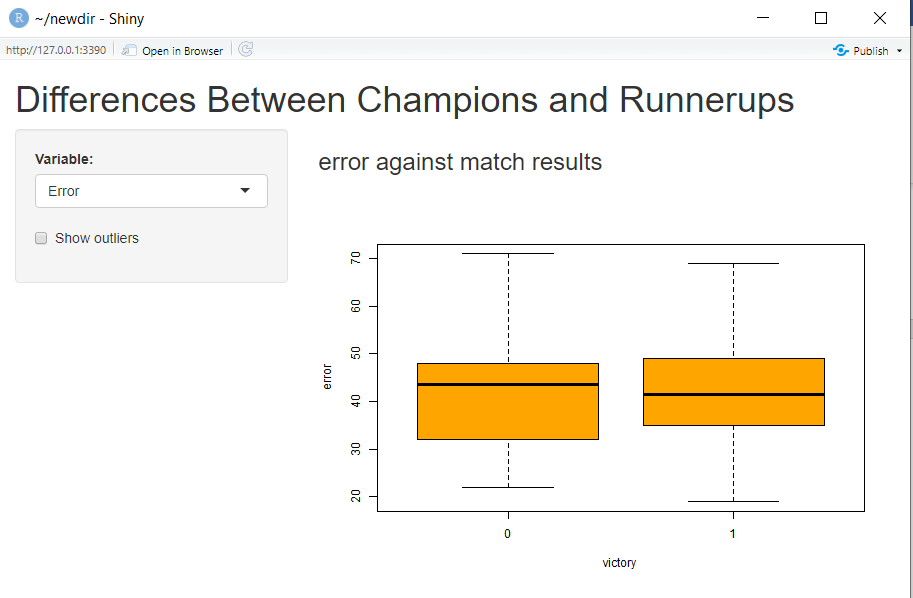
# To run the application

runApp(“newdir”) //here newdir is the name of the folder where ui.R and server.R are stored

Output:

Case 1:

  
Case2:



Case 3:

