R Tutorial

CS 361

Installation

- Please follow the instructions on the following website according to your OS:

  https://www.rstudio.com/products/rstudio/download/#download

- Make sure to have requisite version of R installed before installing R-Studio.
- For Linux/Ubuntu users, the following might be helpful as well:

  https://linuxconfig.org/rstudio-on-ubuntu-18-04-bionic-beaver-linux

Installing and Loading packages in R

- To install libraries and packages in R, type the commands in the “console” window. For example, in order to install **ggplot2:**
  - `install.packages("ggplot2")`
- Similarly to load the same package, use:
  - `library("ggplot2")`

Naming Conventions in R

- R is case sensitive.
- Object names (data or functions) should only contain alpha-numeric characters (A-Z, a-z, 0-9) or a period. Such names cannot start with a digit.
- You should avoid using object names that are already used by R, such as t, c, q, T, F, ls, pt, mean, var, pi, etc.. Use descriptive names.
- **Useful Link:** https://www.johndcook.com/blog/r_language

Assignment

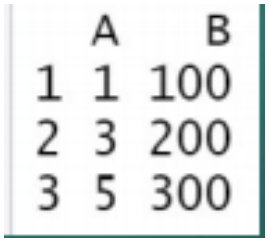
- “<-” and “->” for assignment:

  Ex: “x <- 5”, “5 -> x” are equivalent assignments.

- “=“ also serves as assignment, but mainly used as a specifier of named parameter.

Data Frame

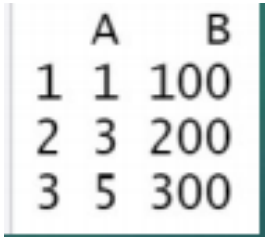
- A Data Frame is a table or a two-dimensional array-like structure in which:
  - Each column contains values of one variable (vector)
  - Each column should contain same number of data items.
  - The column names should be non-empty.
- Each row contains one set of values from each column.
  - The row names if exists should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Example
  - A <- c(1, 3, 5)
  - B <- c(100, 500, 900)
  - C<- data.frame(A, B)



	A	B
1	1	100
2	3	200
3	5	300

Data Frame (continued..)

- Data Frame column vectors can be accessed as:
  - CA, CB
- Other possible access methods are:
  - C[,”A”], C[,”B”]
  - C[[1]], C[[2]]



	A	B
1	1	100
2	3	200
3	5	300

Reading Data in R

R supports a lot of different data types, including but not limited to .txt, .csv, XML and json files. Please consult the following in order to read data in R with different delimiters:

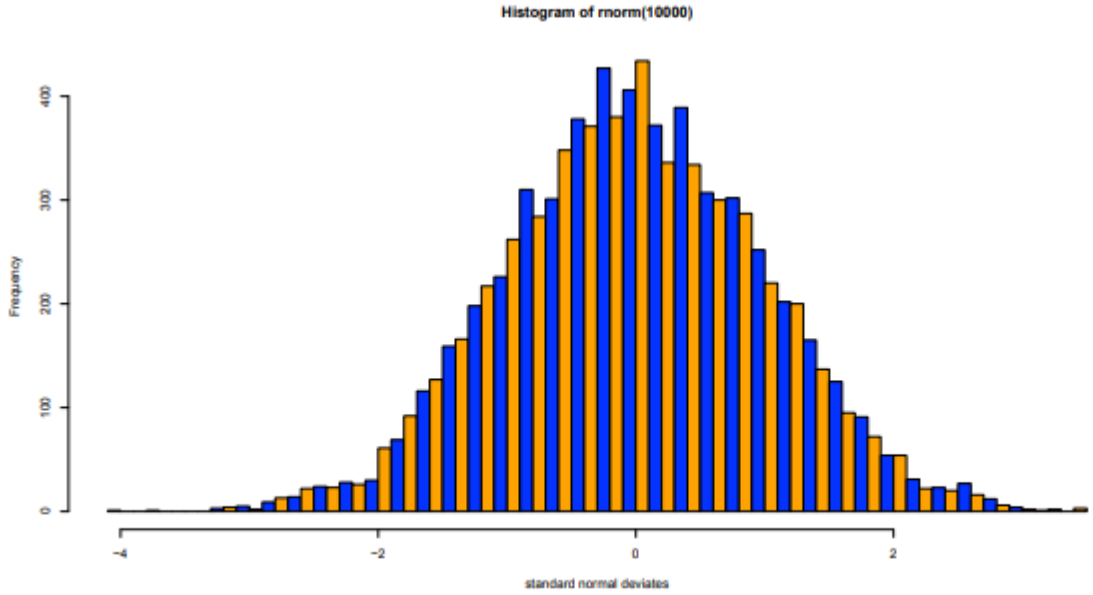
https://www.datacamp.com/community/tutorials/r-data-import-tutorial

For example to read a delimitted .txt file in a Data Frame, the command looks like follows:

```
1  # Read a delimited file
2  df <- read.delim("https://s3.amazonaws.com/assets.datacamp.com/blog_assets/test_delim.txt", sep="$")
3  df <- read.delim2("https://s3.amazonaws.com/assets.datacamp.com/blog_assets/test_delim.txt", sep="$")
4
```

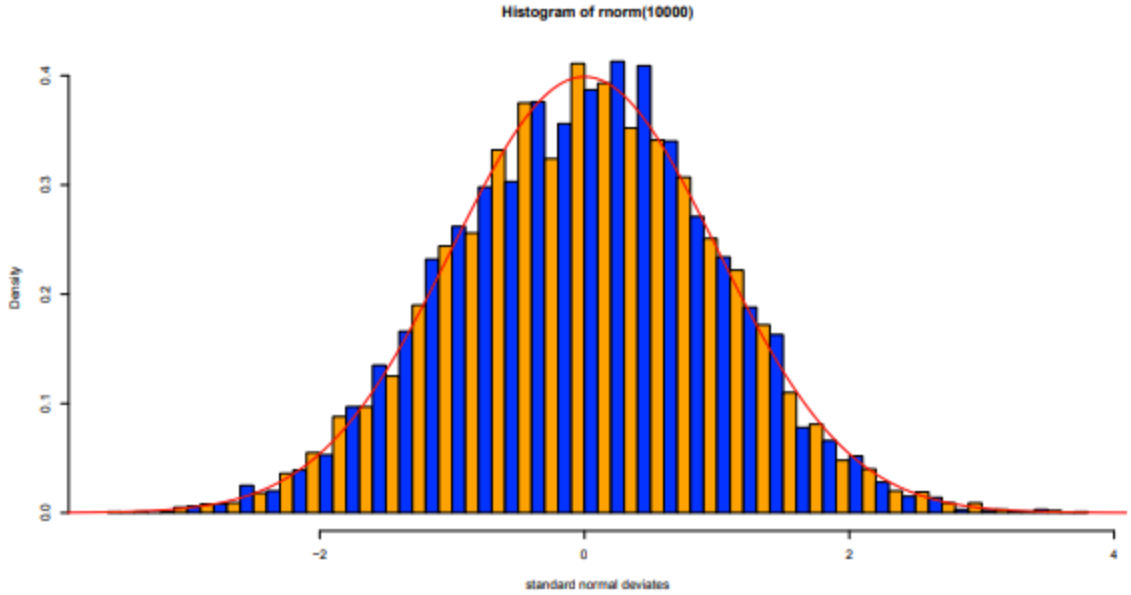
Simple Command Line Graphics

```
> hist(rnorm(10000),nclass=101,col=c("blue","orange"),
+ xlab="standard normal deviates") # rnorm(10000) produces a vector
>                                  # of 10000 standard normal deviates
```



Augmented Command Line Graphics

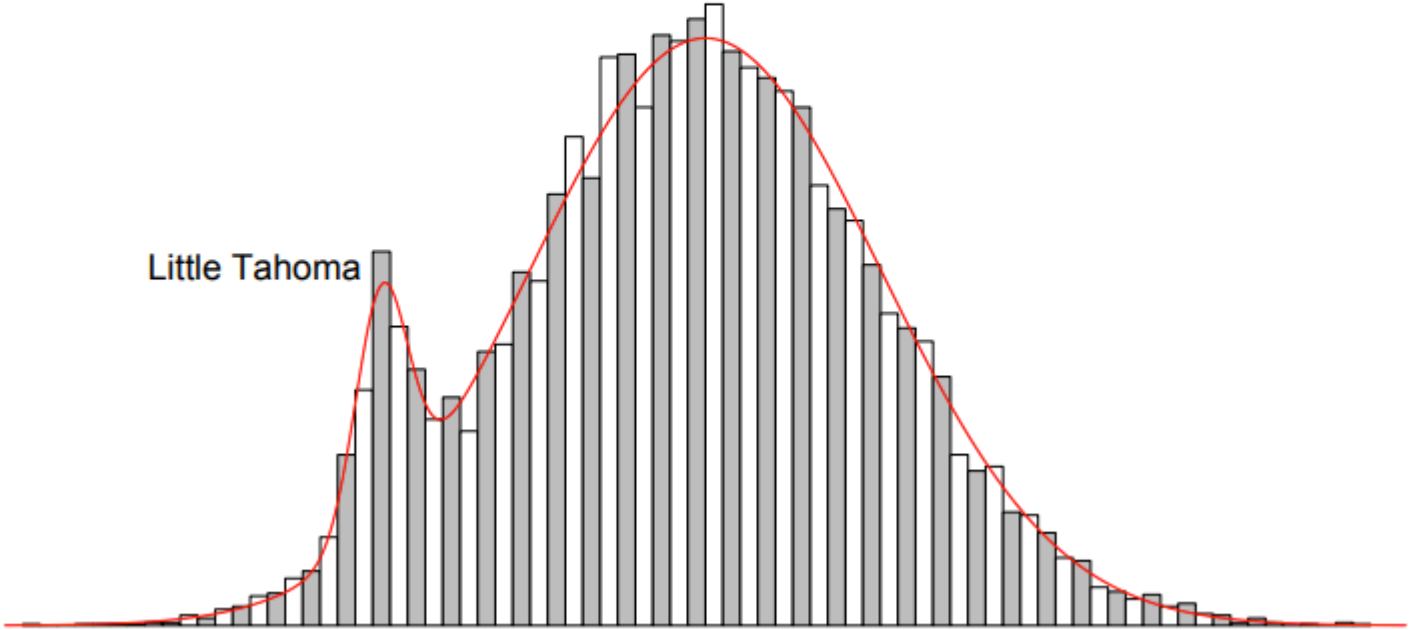
```
> hist(rnorm(10000),nclass=101,col=c("blue","orange"),probability=T,
+ xlab="standard normal deviates")
> lines(seq(-5,5,.01),dnorm(seq(-5,5,.01)),col="red")
# the probability = T option renders the histogram with area 1.
```



Text Annotated Graphics

```
hist.rainier <- function ()
{
x <- rnorm(10000)
y <- rnorm(600,-1.85,.15)
hist(c(x,y),nclass=91,probability=T,main="",xlab="",ylab="",axes=F,
     col=c("grey","white"))
xx <- seq(-4,4,.01)
yy <- 600/10600*dnorm(xx,-1.85,.15)+10000/10600*dnorm(xx)
lines(xx,yy,lwd=2,col="red")
text(-2,.23,"Little Tahoma",adj=1)
}
```

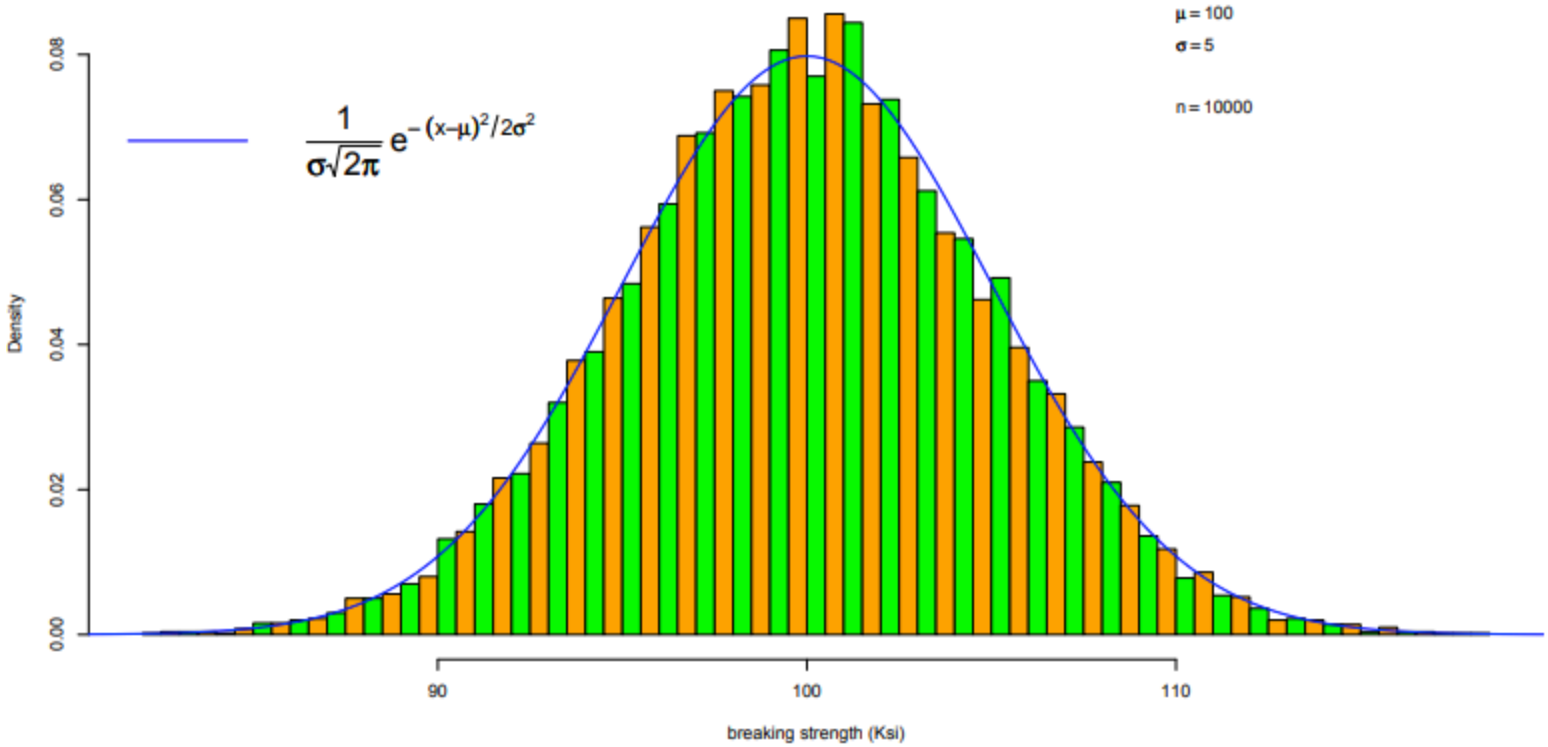
The Resulting Plot



Math Annotated Graphics

```
plot.fun=function (mu=100,sig=5,n=10000)
{
x <- rnorm(n,mu,sig)
out <- hist(x,nclass=101,probability=T,xlab="breaking strength (Ksi)",
       main="",col=c("green","orange"))
M <- max(out$density) # M is height of highest bar in histogram
text(mu+2*sig,.95*M,substitute(sigma==sigx,list(sigx=sig)),adj=0)
text(mu+2*sig,M,substitute(mu==mux,list(mux=mu)),adj=0)
text(mu+2*sig,.85*M, substitute(n==nx,list(nx=n)),adj=0)
xx <- seq(mu-4*sig,mu+4*sig,length.out=201)
lines(xx,dnorm(xx,mu,sig),lwd=2,col="blue")
legend(80,.08,expression(over(1,sigma*sqrt(2*pi))*~
       e^{-~(x-mu)^2/2*sigma^2}),
       bty="n",lty=1,col="blue",lwd=2,cex=1.8)
}
```

The Resulting Math Annotated Plot



Factors

```
Torque=read.csv("Torque.csv",header=T) # Torque.csv is a data set
> Torque[1:5,]                         # with 3 columns: Screw Plating Torque
  Screw Plating Torque
1  bolt      CW     20
2  bolt      CW     16
3  bolt      CW     17
4  bolt      CW     18
5  bolt      CW     15
> Torque[31:35,]
     Screw Plating Torque
31 mandrel      CW     24
32 mandrel      CW     18
33 mandrel      CW     17
34 mandrel      CW     17
35 mandrel      CW     15
> is.factor(Torque$Plating)
[1] TRUE
```

About Factors

read.table or read.csv reads character data as a factor, coded internally as numbers. Factors are useful for categorical or grouped data.

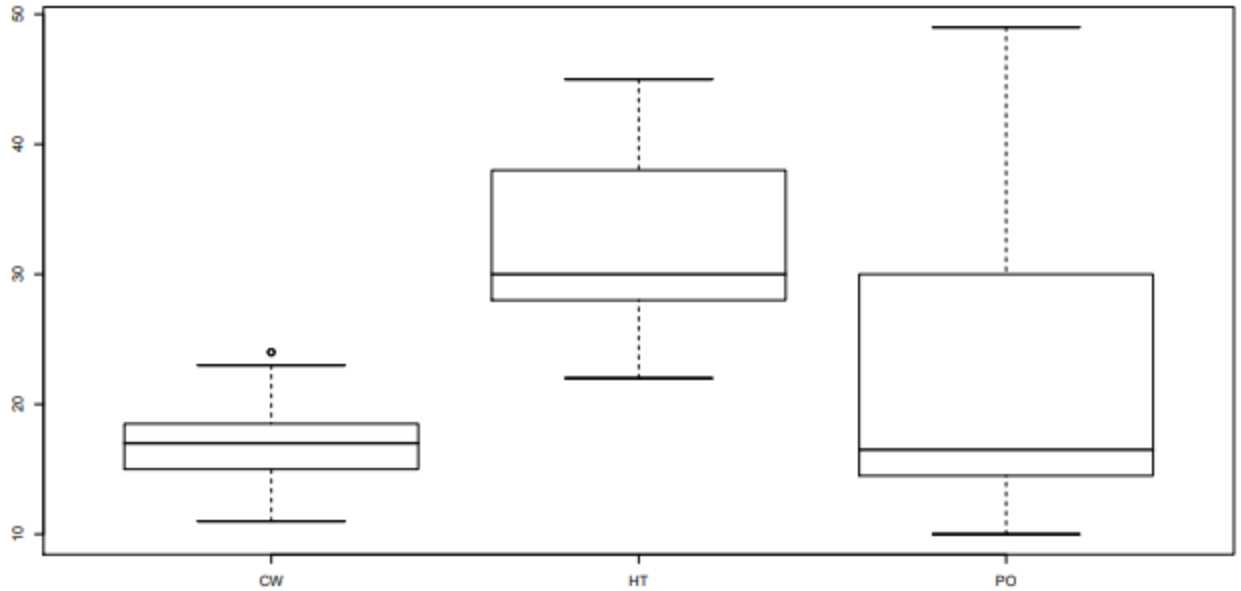
They are treated in special ways in many functions.

Each number or category also has a name, stored in levels.

```
> Torque$Plating[1:20]
 [1] CW CW CW CW CW CW CW CW CW CW HT HT HT HT HT HT HT HT
[19] HT HT
Levels: CW HT PO
>  as.numeric(Torque$Plating)
 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3
[28] 3 3 3 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3
[55] 3 3 3 3 3 3
> levels(Torque$Plating)
[1] "CW" "HT" "PO"
> Torque$Plating[1]=="CW"
[1] TRUE
```

An Example Use of Factors

```
> boxplot(Torque~Plating,data=Torque) #data=Torque identifies the
  # data.frame Torque. Torque~Plating identifies the variable Torque
  # in that data.frame. More on ~ later in the course.
```



References

http://faculty.washington.edu/fscholz/DATAFILES498B2008/Stat498BRintro.pdf

https://www.datacamp.com/community/tutorials/r-data-import-tutorial