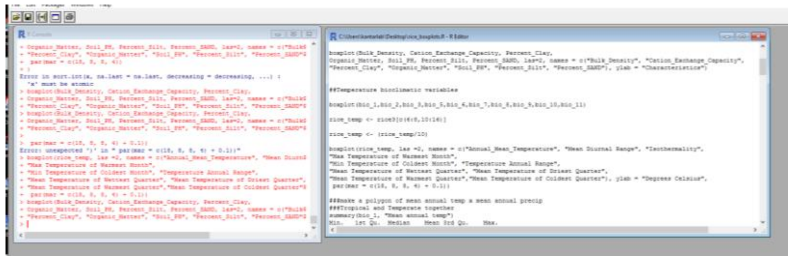
The R ecosystem

Download 'R'

- R is open source software developed from proprietary software developed by AT&T

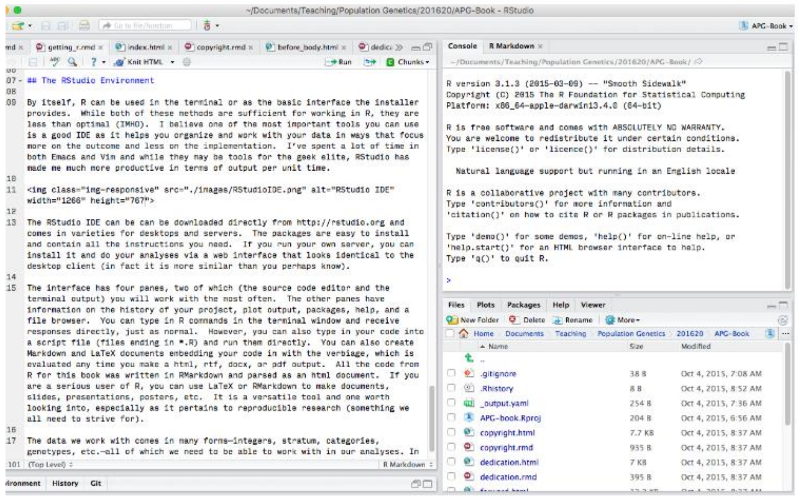
- The software works on all platforms and can be downloaded at:
  - https://cran.r-project.org/

- A useful way to use this software is RSTUDIO
  - https://www.rstudio.com/home/



```
+ Organic_Matter, Soil_PH, Percent_Silt, Percent_SAND, las=2, names = c("Bulk
+ "Percent_Clay", "Organic_Matter", "Soil_PH", "Percent_Silt", "Percent_SAND"
+  par(mar = c(18, 8, 8, 4))
+ )
Error in sort.int(x, na.last = na.last, decreasing = decreasing, ...) :
  'x' must be atomic
> boxplot(Bulk_Density, Cation_Exchange_Capacity, Percent_Clay,
+ Organic_Matter, Soil_PH, Percent_Silt, Percent_SAND, las=2, names = c("Bulk
+ "Percent_Clay", "Organic_Matter", "Soil_PH", "Percent_Silt", "Percent_SAND"
> boxplot(Bulk_Density, Cation_Exchange_Capacity, Percent_Clay,
+ Organic_Matter, Soil_PH, Percent_Silt, Percent_SAND, las=2, names = c("Bulk
+ "Percent_Clay", "Organic_Matter", "Soil_PH", "Percent_Silt", "Percent_SAND"
>
>  par(mar = c(18, 8, 8, 4) + 0.1))
Error: unexpected ')' in " par(mar = c(18, 8, 8, 4) + 0.1))"
> boxplot(rice_temp, las =2, names = c("Annual_Mean_Temperature", "Mean Diurn
+ "Max Temperature of Warmest Month",
+ "Min Temperature of Coldest Month", "Temperature Annual Range",
+ "Mean Temperature of Wettest Quarter", "Mean Temperature of Driest Quarter",
+ "Mean Temperature of Warmest Quarter","Mean Temperature of Coldest Quarter"
+  par(mar = c(18, 8, 8, 4) + 0.1))
> boxplot(Bulk_Density, Cation_Exchange_Capacity, Percent_Clay,
+ Organic_Matter, Soil_PH, Percent_Silt, Percent_SAND, las=2, names = c("Bulk
+ "Percent_Clay", "Organic_Matter", "Soil_PH", "Percent_Silt", "Percent_SAND"
> 
```

```
boxplot(Bulk_Density, Cation_Exchange_Capacity, Percent_Clay,
Organic_Matter, Soil_PH, Percent_Silt, Percent_SAND, las=2, names = c("Bulk_Density", "Cation_Exchange_Capacity",
"Percent_Clay", "Organic_Matter", "Soil_PH", "Percent_Silt", "Percent_SAND"), ylab = "Characteristics")


##Temperature bioclimatic variables

boxplot(bio_1,bio_2,bio_3,bio_5,bio_6,bio_7,bio_8,bio_9,bio_10,bio_11)

rice_temp <- rice3[c(6:8,10:16)]

rice_temp <- (rice_temp/10)

boxplot(rice_temp, las =2, names = c("Annual_Mean_Temperature", "Mean Diurnal Range", "Isothermality",
"Max Temperature of Warmest Month",
"Min Temperature of Coldest Month", "Temperature Annual Range",
"Mean Temperature of Wettest Quarter", "Mean Temperature of Driest Quarter",
"Mean Temperature of Warmest Quarter","Mean Temperature of Coldest Quarter"), ylab = "Degrees Celsius",
 par(mar = c(18, 8, 8, 4) + 0.1))

###make a polygon of mean annual temp x mean annual precip
###Tropical and Temperate together
summary(bio_1, "Mean annual temp")
Min.   1st Qu.  Median    Mean 3rd Qu.    Max.
```



```
## The RStudio Environment

By itself, R can be used in the terminal or as the basic interface the installer
provides.  While both of these methods are sufficient for working in R, they are
less than optimal (IMHO).  I believe one of the most important tools you can use
is a good IDE as it helps you organize and work with your data in ways that focus
more on the outcome and less on the implementation.  I've spent a lot of time in
both Emacs and Vim and while they may be tools for the geek elite, RStudio has
made me much more productive in terms of output per unit time.

<img class="img-responsive" src="./images/RStudioIDE.png" alt="RStudio IDE"
width="1266" height="767">

The RStudio IDE can be can be downloaded directly from http://rstudio.org and
comes in varieties for desktops and servers.  The packages are easy to install
and contain all the instructions you need.  If you run your own server, you can
install it and do your analyses via a web interface that looks identical to the
desktop client (in fact it is more similar than you perhaps know).

The interface has four panes, two of which (the source code editor and the
terminal output) you will work with the most often.  The other panes have
information on the history of your project, plot output, packages, help, and a
file browser.  You can type in R commands in the terminal window and receive
responses directly, just as normal.  However, you can also type in your code into
a script file (files ending in *.R) and run them directly.  You can also create
Markdown and LaTeX documents embedding your code in with the verbiage, which is
evaluated any time you make a html, rtf, docx, or pdf output.  All the code from
R for this book was written in RMarkdown and parsed as an html document.  If you
are a serious user of R, you can use LaTeX or RMarkdown to make documents,
slides, presentations, posters, etc.  It is a versatile tool and one worth
looking into, especially as it pertains to reproducible research (something we
all need to strive for).

The data we work with comes in many forms—integers, stratum, categories,
genotypes, etc.—all of which we need to be able to work with in our analyses. In
```

```
~/Documents/Teaching/Population Genetics/201620/APG-Book/

R version 3.1.3 (2015-03-09) -- "Smooth Sidewalk"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Size	Modified
..		
.gitignore	38 B	Oct 4, 2015, 7:08 AM
.Rhistory	8 B	Oct 4, 2015, 8:52 AM
_output.yaml	254 B	Oct 4, 2015, 7:36 AM
APG-book.Rproj	204 B	Oct 4, 2015, 6:56 AM
copyright.html	7.7 KB	Oct 4, 2015, 8:37 AM
copyright.rmd	935 B	Oct 4, 2015, 8:37 AM
dedication.html	7 KB	Oct 4, 2015, 8:37 AM
dedication.rmd	395 B	Oct 4, 2015, 8:37 AM

Getting Started

- There is only a limited amount of functionality in R
    - It will only do what you tell it to do
- People write ‘scripts’ (software) known as packages that you load to add specific functionality
- It is good to use specific packages to answer your specific research questions, and if there isn’t a function you can write one yourself
- You tell R to use a package by using the command
    - library()

Using R to do basic Math

- Basic operations can be done using
  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - Division: /
  - Exponentiation: ^
  - Modulo: %%
  - sqrt()
  - = assigns value, is not an operation

Assign a value to a name

```
rainbow<-5

rainbow<-c(1,2,3,4,5,6)

rainbow/2

mean(rainbow)
```

Create a matrix

```
mat <- matrix(c(6,5,4,3,2,1),ncol=3,byrow=TRUE)

diag(mat)

mat%*% t(mat)
```

Inputting Data

- Most common formats are ‘flat’ files
  - Comma separated files (.csv)
  - Tab delimited files (.txt)
- Regular Files can also be imported
  - Excel file (.xls)

- How does this work?

```
data<-read.csv(“c:/mycsv.csv”)
data<-read.table(“c:/mytext.txt”, header=T)
library(xlsx)
data <- read.xlsx("c:/myexcel.xlsx", 1)
```

Name of R dataframe

Command to recognize the file

Path to where the file lives

Inputting Data

- What if I don’t want t write the whole file path

- Print working directory

getwd()

Print the current directory

- Set working directory

setwd(“dir where your files are”)

Set directory where your files are and where you can call your files, now you just type the file name rather than the entire file path

Exploring Data: What does the data look like?

```
#Check if data is correct
names(data)
```

Gives the headers of all the columns in the data

```
#look at the data
head(data)
```

Looks at the first six lines of the dataframe

Exploring Data: Importance of type of variables

```
#Check if data is correct
str(data)
```

Look at what the internal structure of an R object

```
#look at the data
summary(data)
```

Generate summary statistics on every variable in your dataset

Manipulating your data set

- Sometimes you will need to manipulate your dataset

- Remove a column

```
Subdata<-data[,-1]
```

When exploring a dataframe R recognizes data as [row, column], this command says remove the first column and leave the number of rows the same

- Add a column

```
data$newfactor<-as.factor(data2$factor)
```

Add a new factor from a second dataset

Manipulating your data set

- Add rownames

```
rownames(data)<-data[,1]
```

Take the first column and make these data rownames for each row

- Transpose your dataset

```
Data_transpose<-t(data)
```

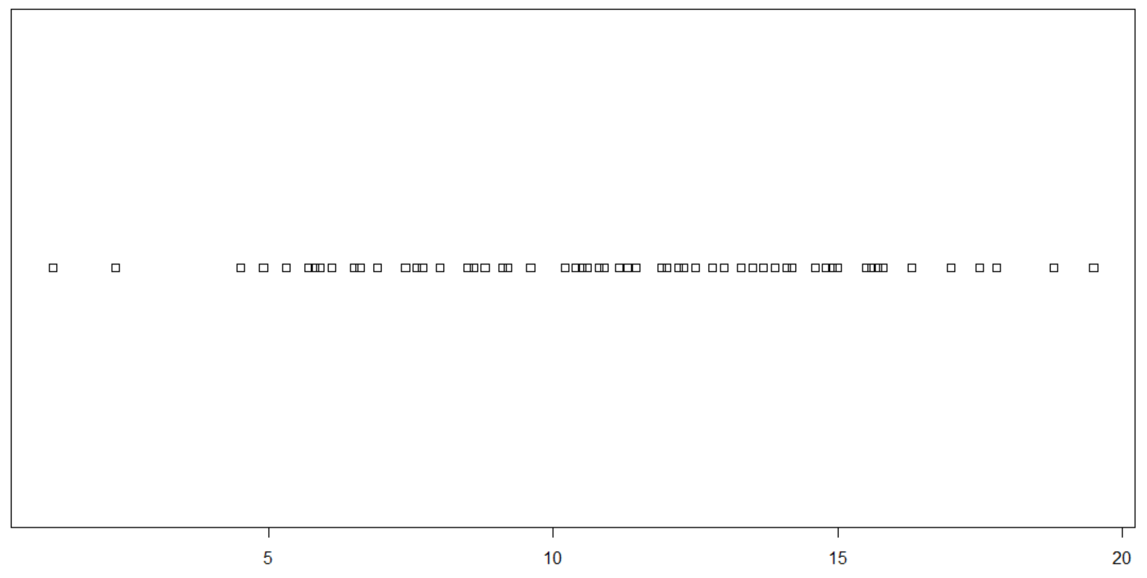
Transpose your matrix

Visualizing Data: Common Graphics

- Strip Charts
- Histograms
- Boxplots
- Scatter Plots
- Normal QQ Plots

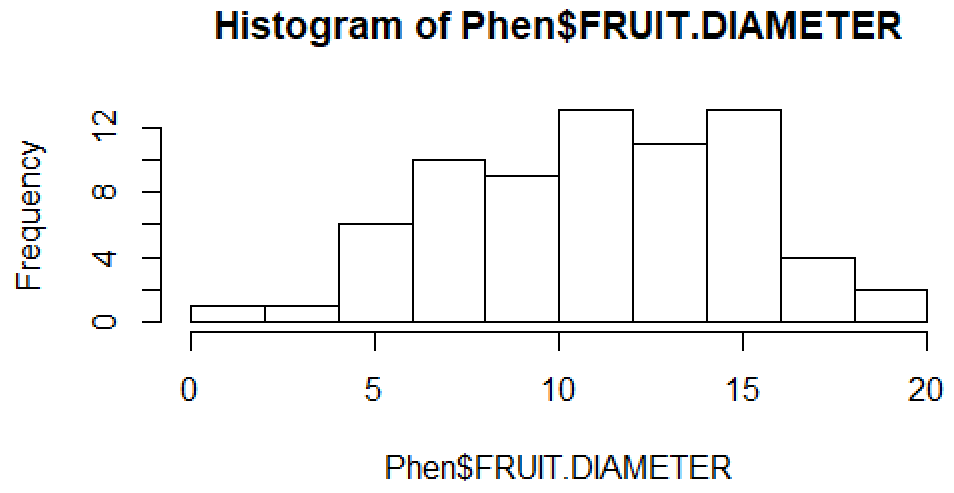
Stripchar

- stripchart(data$factor1)



Histograms

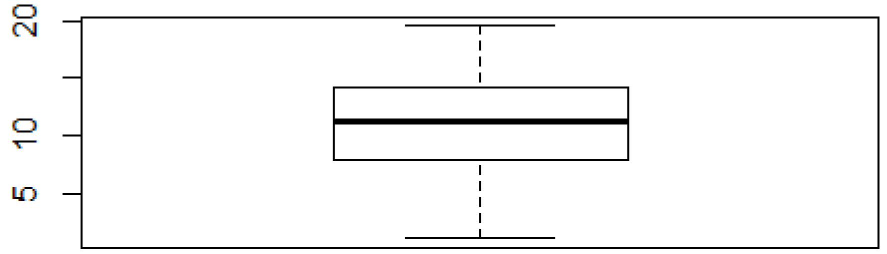
```
hist(data$factor1)
```

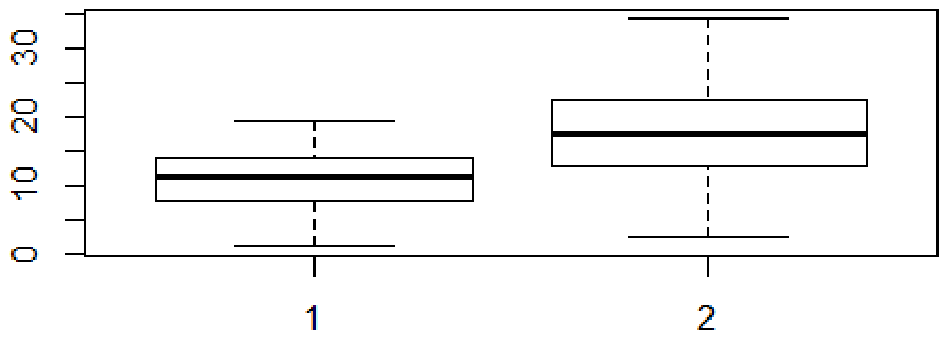


Box Plots

Boxplot(data$factor1)

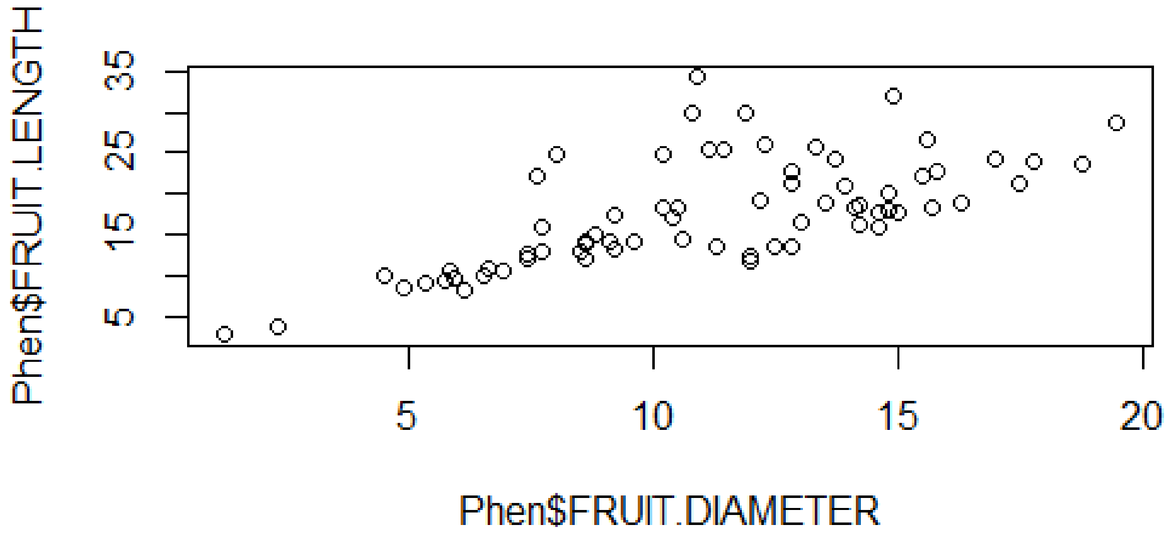
boxplot(data$factor1, data$factor2)





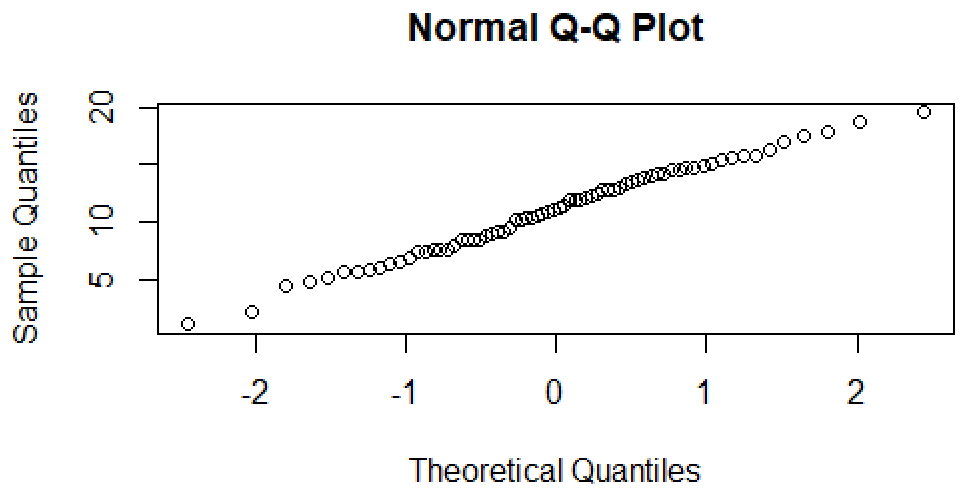
Scatter Plots

```
plot(data$factor1, data$factor2)
```



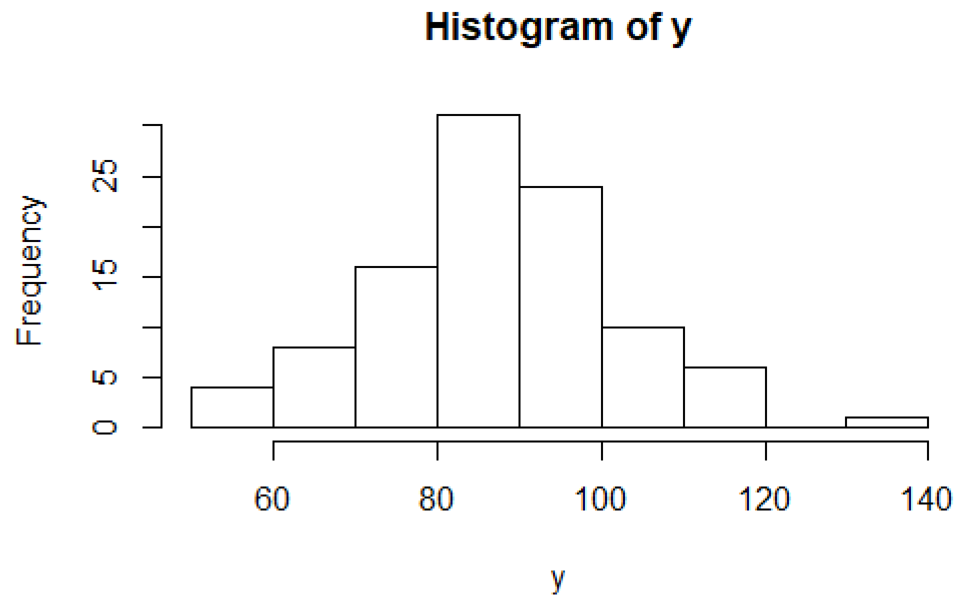
QQplot

```
qqnorm(data$factor1,
        main="Normal Q-Q Plot",
        xlab="Theoretical Quantiles",
        ylab="Sample Quantiles")
```



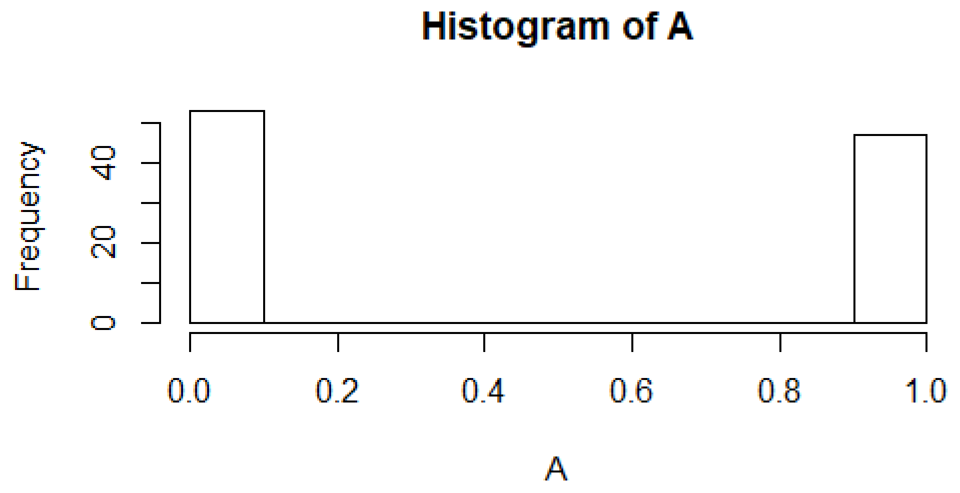
Lets Create Distributions

```
y <- rnorm(100,mean=0,sd=1)
hist(y)
```



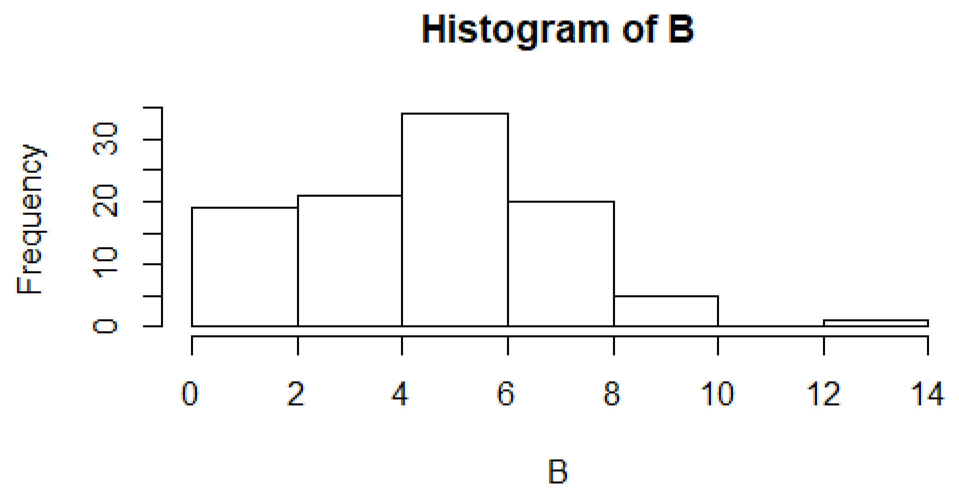
Bernoulli Distribution

```
A<-rbinom(100,1,.5)
hist(A)
```



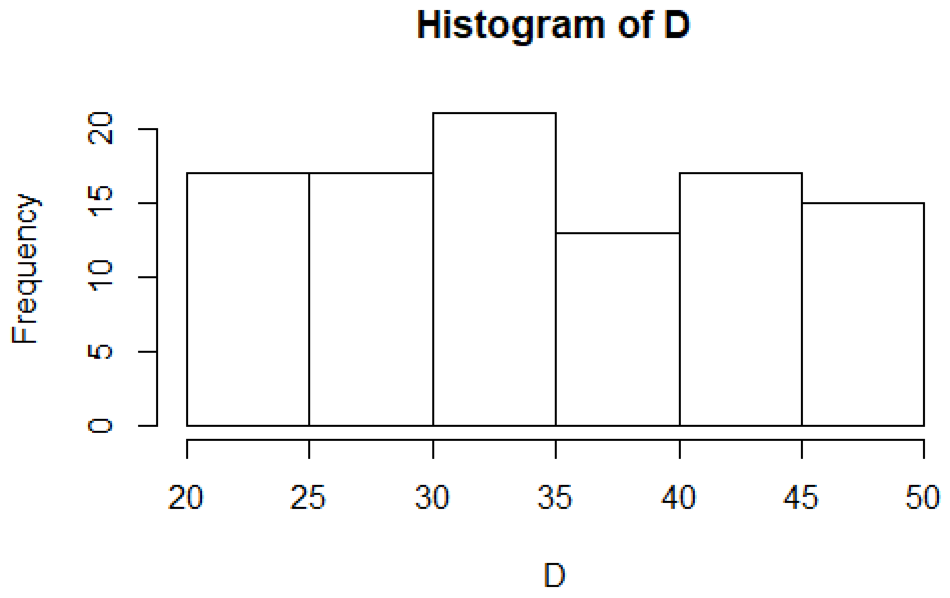
The Poisson Distribution

```
B<-rpois(100, 5)
hist(B)
```



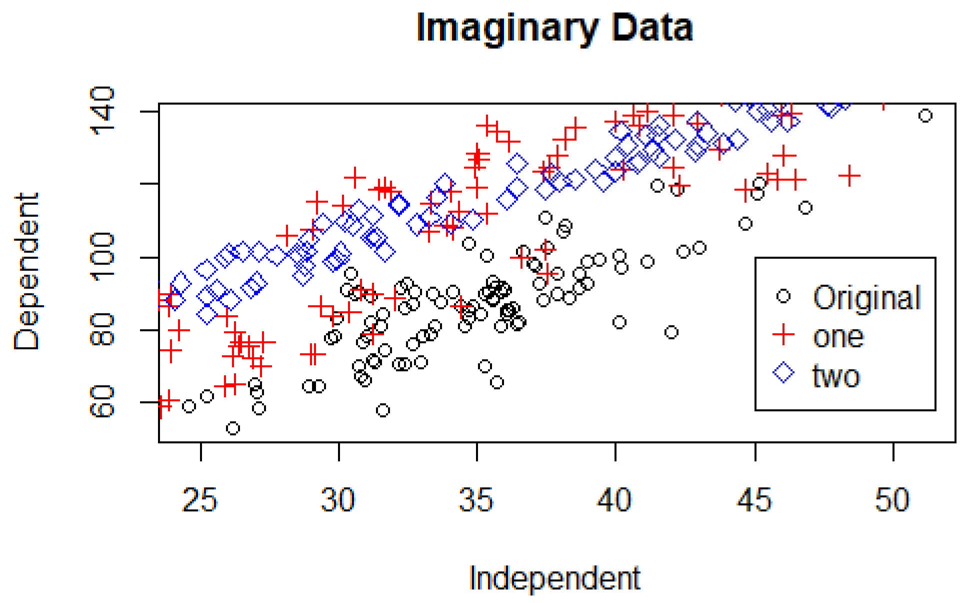
Uniform distribution

```
D<-runif(100,20,50)
hist(D)
```



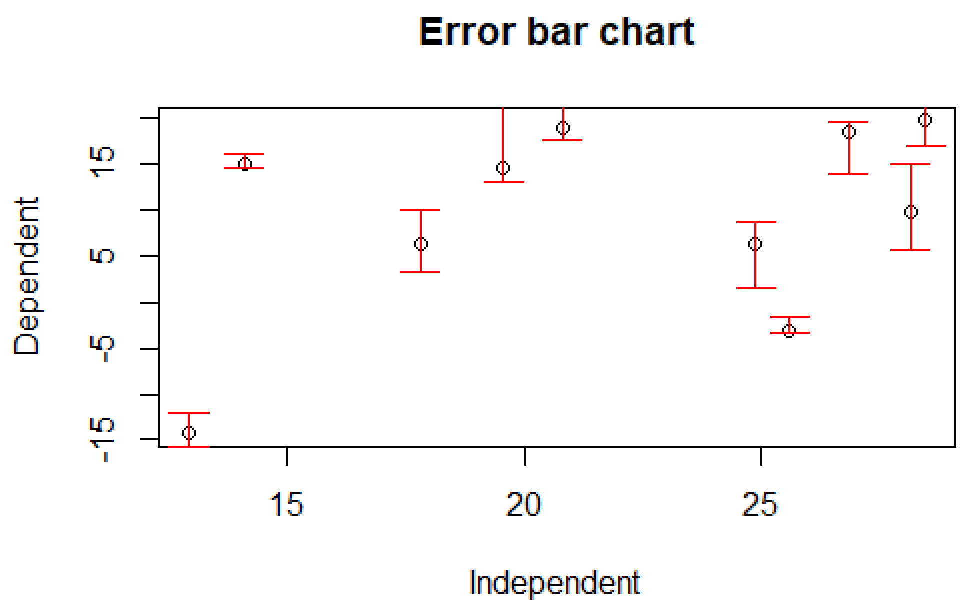
Let plot some random data

```
x <- rnorm(100,sd=5,mean=35)
y <- 2.5*x - 1.0 + rnorm(100,sd=9,mean=0)
cor(x,y)
plot(x,y,xlab="Independent",ylab="Dependent",main="Imaginary Data")
x1 <- runif(100,20,50) #random uniform distribution, number of samples,min,max
y1 <- 2.5*x1 - 1.0 + runif(100,0,50)
points(x1,y1,col=2)

x2 <- runif(100,20,50)
y2 <- 2.5*x2 - 1.0 + runif(100,22,37)
points(x2,y2,col=3,pch=2)

plot(x,y,xlab="Independent",ylab="Dependent",main="Imaginary Data")
points(x1,y1,col=2,pch=3)
points(x2,y2,col=4,pch=5)
legend(45,100,c("Original","one","two"),col=c(1,2,4),pch=c(1,3,5))
```



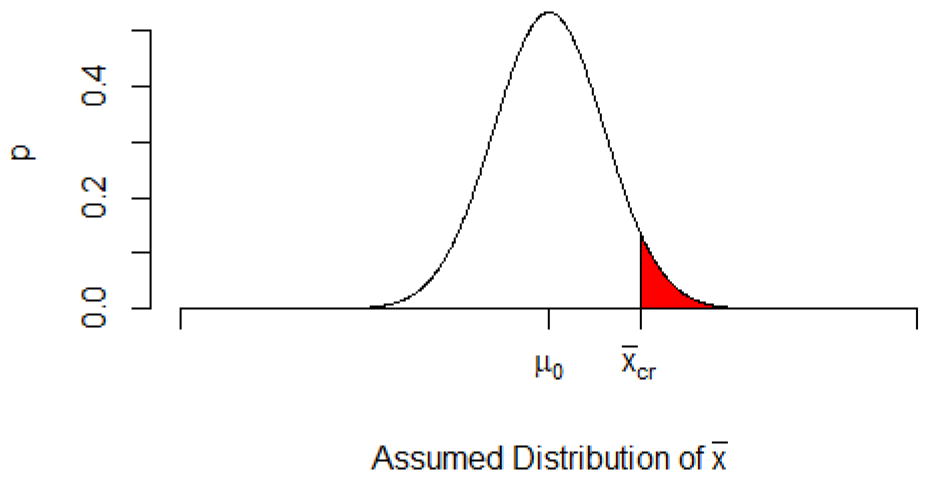
Add error bars

```
x <- rnorm(10,sd=5,mean=20)
y <- rnorm(10,sd=9,mean=10)
plot(x,y,xlab="Independent",ylab="Dependent"
, main="Error bar chart")
xHigh <- x
yHigh <- y + abs(rnorm(10,sd=3.5))
xLow <- x
yLow <- y - abs(rnorm(10,sd=3.1))
arrows(xHigh,yHigh,xLow,yLow,col=2,angle=90,
length=0.1,code=3)
```



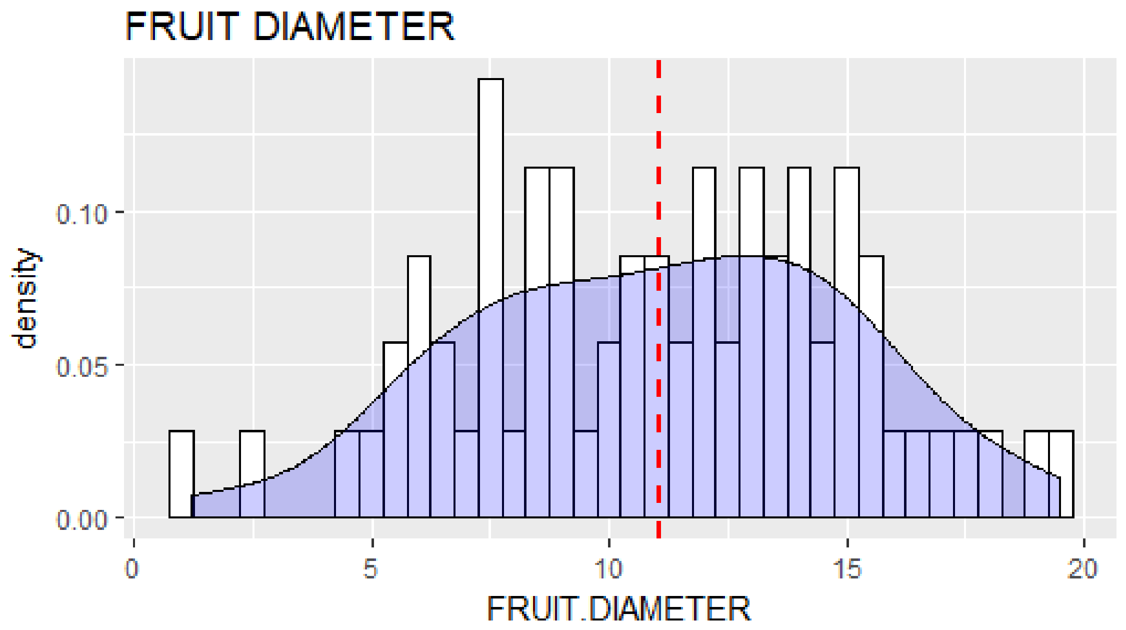
More complicated graphic

```
stdDev <- 0.75;
x <- seq(-5,5,by=0.01)
y <- dnorm(x,sd=stdDev)
right <- qnorm(0.95,sd=stdDev)
plot(x,y,type="l",xaxt="n",ylab="p", # l indicates lines, xaxt removes the x-axis
    xlab=expression(paste('Assumed Distribution of ',bar(x))),
    axes=FALSE,ylim=c(0,max(y)*1.05),xlim=c(min(x),max(x)),
    frame.plot=FALSE)
axis(1,at=c(-5,right,0,5),
    pos = c(0,0),
    labels=c(expression(' '),expression(bar(x)[cr]),expression(mu[0]),expression(' ')))
axis(2)
xReject <- seq(right,5,by=0.01)
yReject <- dnorm(xReject,sd=stdDev)
polygon(c(xReject,xReject[length(xReject)],xReject[1]),
     c(yReject,0, 0), col='red')
```

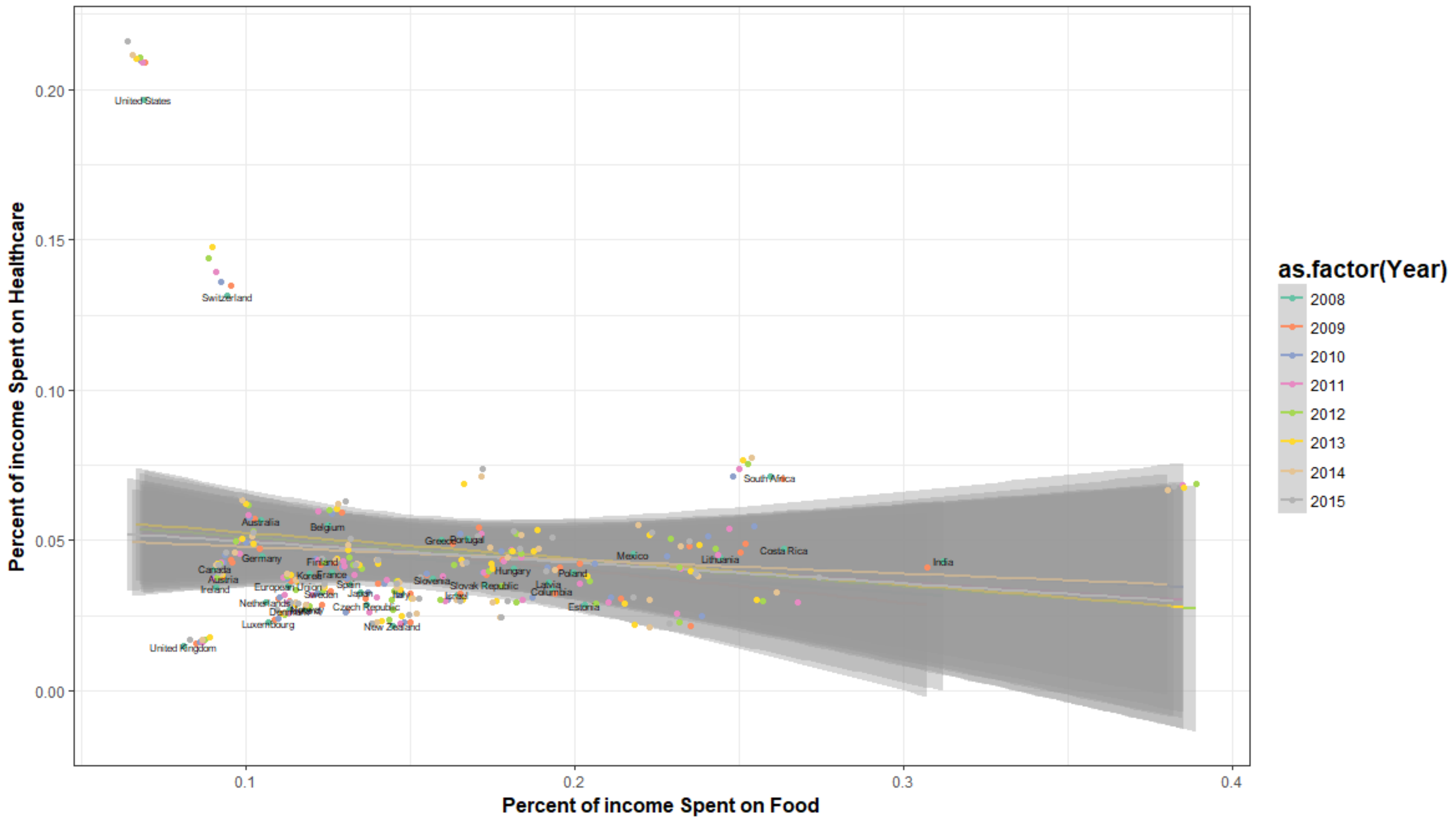


More fun plots with ggplot2

```
library(ggplot2)

ggplot(Phen, aes(x=FRUIT.DIAMETER)) +
  geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                 binwidth=.5,
                 colour="black", fill="white") +
  geom_density(alpha=.2, fill="Blue") + # Overlay with transparent density plot

geom_vline(aes(xintercept=mean(FRUIT.DIAMETER, na.rm=T)),   # Ignore NA values for mean
             color="red", linetype="dashed", size=1) +ggtitle("FRUIT DIAMETER")
```

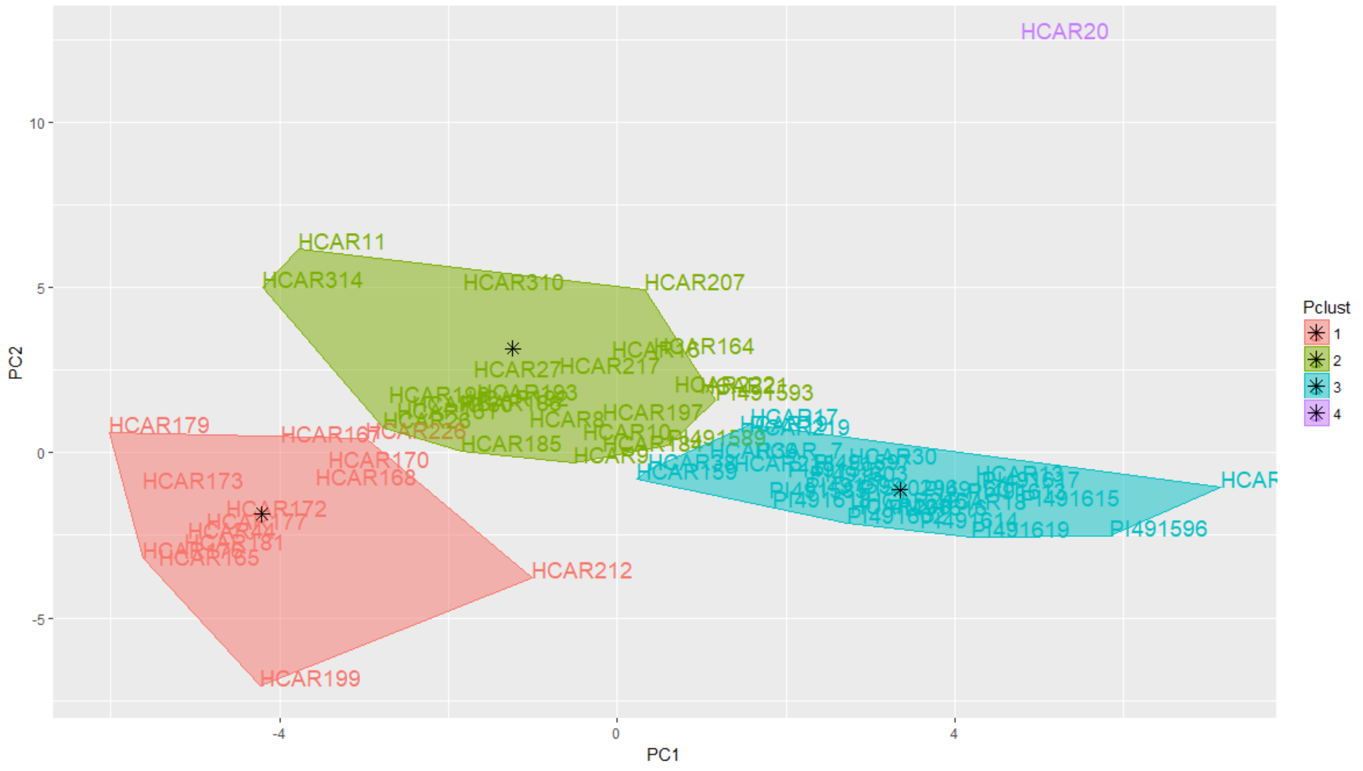


```
ggplot(Food_income_data, aes(x=Percent_food,
y=Percent_health,  colour=as.factor(Year),
group=as.factor(Year), label=Country.Name)) +

geom_smooth(method='lm', formula = y ~ x) +

geom_point() +

geom_text(data=subset(Food_income_data, Year <

2009), size=2, color='black')+

 theme_bw()+

scale_color_brewer(palette = "Set2")+

xlab("Percent of income Spent on Food") +

ylab("Percent of income Spent on Healthcare")+

theme(axis.title.x = element_text(face="bold",
size=12),

axis.title.y = element_text(face="bold", size=12),

legend.title = element_text(size=14, face="bold"))
```



Another fun graphic

```
##Plot PCA with convex Hulls around the groups

library(ggplot2)

library(plyr)

library(dplyr)

#getting the convex hull of each unique point set

df <- data.frame(pap_1_PCA$ind$coord[,1:5])

df$Kmeans3<-as.factor(fit3$cluster)

df$Kmeans4<-as.factor(fit4$cluster)

df$Kmeans5<-as.factor(fit5$cluster)

df$Pclust<-as.factor(res.hcpc$data.clust$clust)

str(df)

find_hull <- function(df) df[chull(df$Dim.1, df$Dim.2), ]

hulls2 <- ddply(df, "Pclust", find_hull)

#Clustering on 5 groups

plot_of_polygons <- ggplot(data = df, aes(x = Dim.1, y = Dim.2, colour=Pclust, fill = Pclust)) +

 geom_text(aes(label=rownames(Phen)),size = 5,hjust=0,vjust=0)+

 geom_polygon(data = hulls2, alpha = 0.5) +

 geom_point( x=-4.21,y=-1.86 ,color='black',size=3,shape=8) +

 geom_point( x=-1.24, y=3.15, color='black',size=3,shape=8) +

 geom_point(x= 3.35,y=-1.13,color='black',size=3, shape=8) +

 labs(x = "PC1", y = "PC2")
```


HCAR20
PC2
PC1
HCAR11
HCAR314
HCAR310
HCAR207
HCAR164
HCAR27
HCAR217
PI491593
HCAR197
HCAR185
HCAR179
HCAR173
HCAR170
HCAR168
HCAR172
HCAR212
HCAR199
HCAR165
HCAR17
PI491615
PI491619
PI491596
Pclust