Contenido

Week 1

Tomás

5 de diciembre de 2019

Principles of Analytic Graphics

Principios:

- 1: Muestra comparaciones para poder entender mejor los datos
- 2: Demuestra causalidad, explicaciones, estructuras sistematicas, que nos permite identificar el por que de estas diferencias
- 3: Busca muchas variables o datos, esto permite asemejar las comparaciones al mundo real
- 4: Integra evidencia, ocupa muchas formas de mostrar la evidencia para que tu analisis sea lo mas rico posible. Numeros, imagenes, diagramas, graficos.
- 5: Describe y documenta tu evidencia
- 6: El contenido es lo importante, un analisis que no tiene un contenido solido e interesante solamente va a ser una perdida de tiempo

Exploratory Graphs (part 1)

Por que usamos graficos?

- Entender las propiedades de la data
- Encontrar patrones
- Sugerir modelos
- Realizar analsiis
- Comunicar resultados

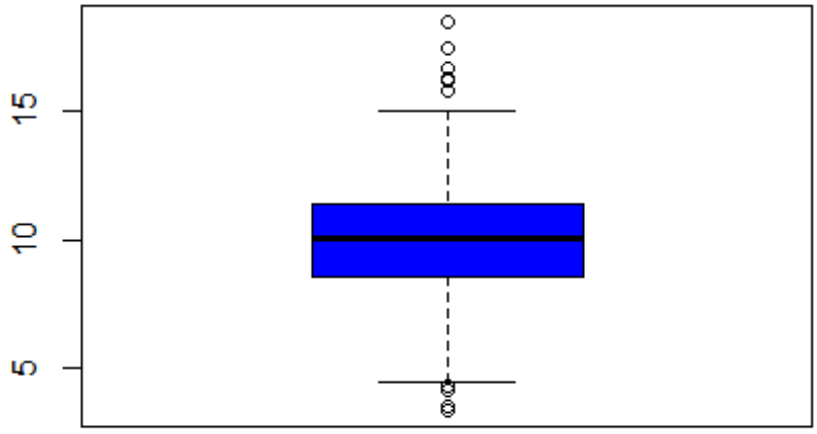
```
pollution <- read.csv("avgpm25.csv", colClasses =
c("numeric","character","factor","numeric","numeric"))
head(pollution)
```

```
##        pm25  fips region longitude latitude
## 1  9.771185 01003   east -87.74826 30.59278
## 2  9.993817 01027   east -85.84286 33.26581
## 3 10.688618 01033   east -87.72596 34.73148
## 4 11.337424 01049   east -85.79892 34.45913
## 5 12.119764 01055   east -86.03212 34.01860
## 6 10.827805 01069   east -85.35039 31.18973
```

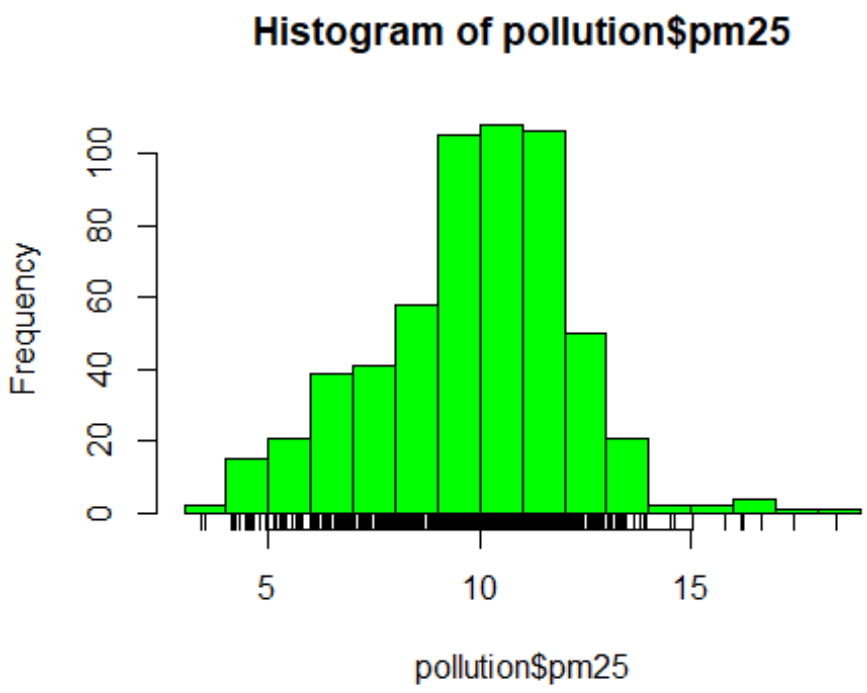
```r
summary(pollution$pm25)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.383   8.549  10.047   9.836  11.356  18.441
```

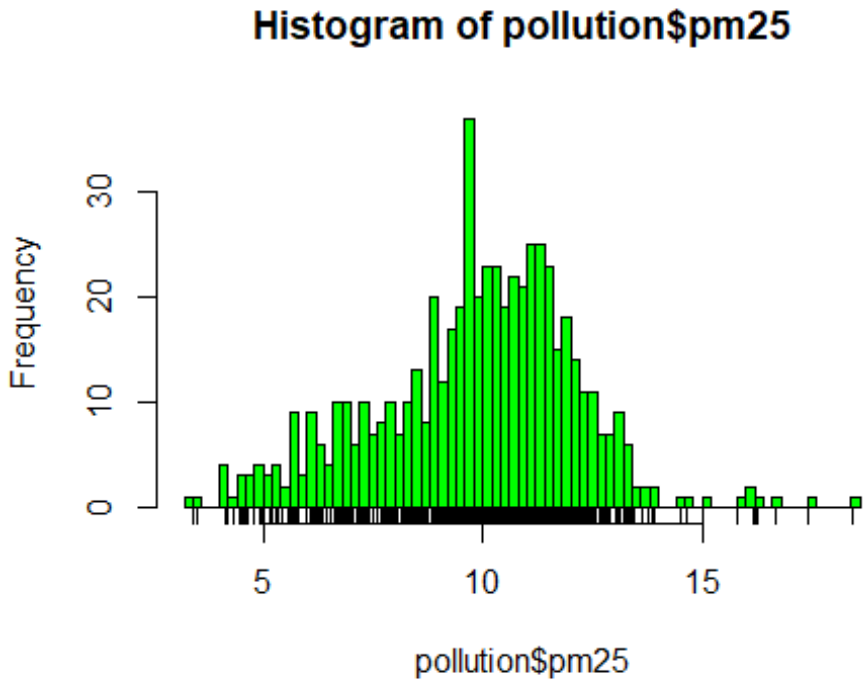
```r
boxplot(pollution$pm25, col = "blue") # Boxplot
```



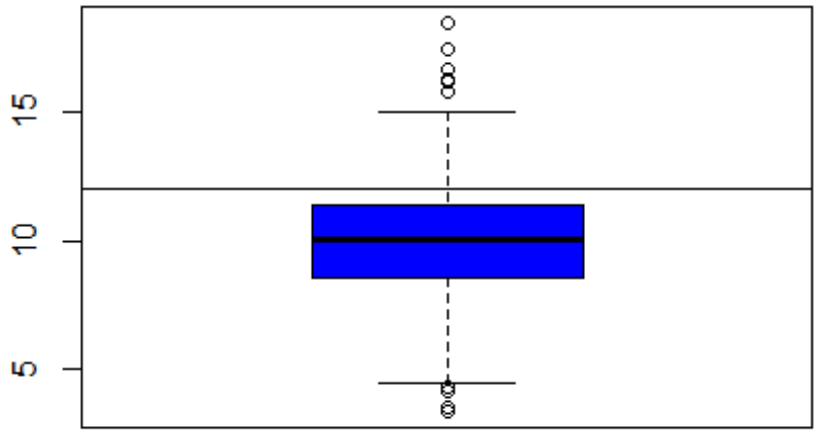

```r
hist(pollution$pm25, col = "green") # Histograma
rug(pollution$pm25) #Entrega una "Alfombra"" que situa las observaciones en el grafico
```



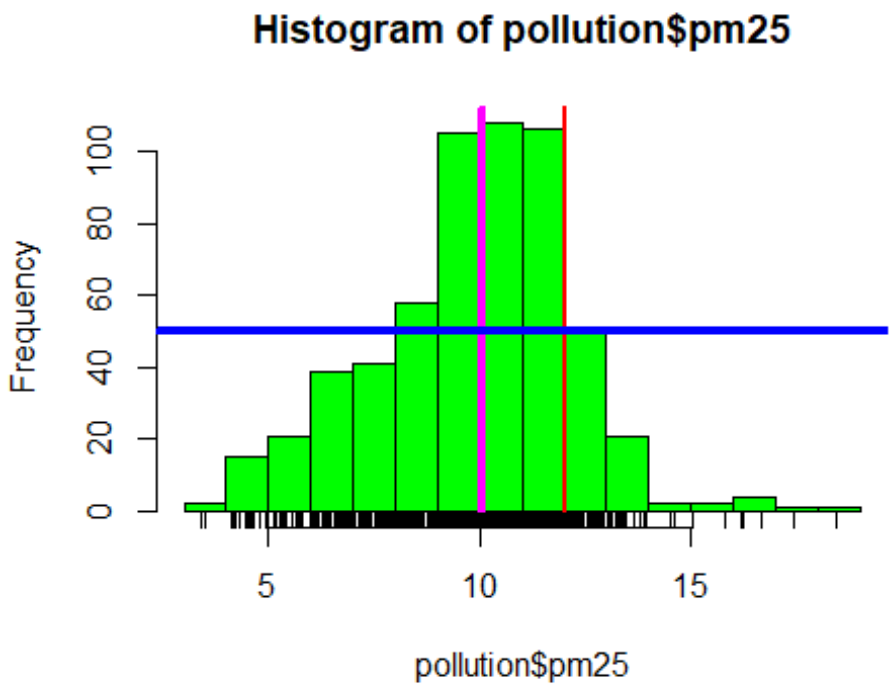

```r
hist(pollution$pm25, col = "green", breaks = 100) # Breaks permite entregar la cantidad de columnas que quiero
rug(pollution$pm25)
```



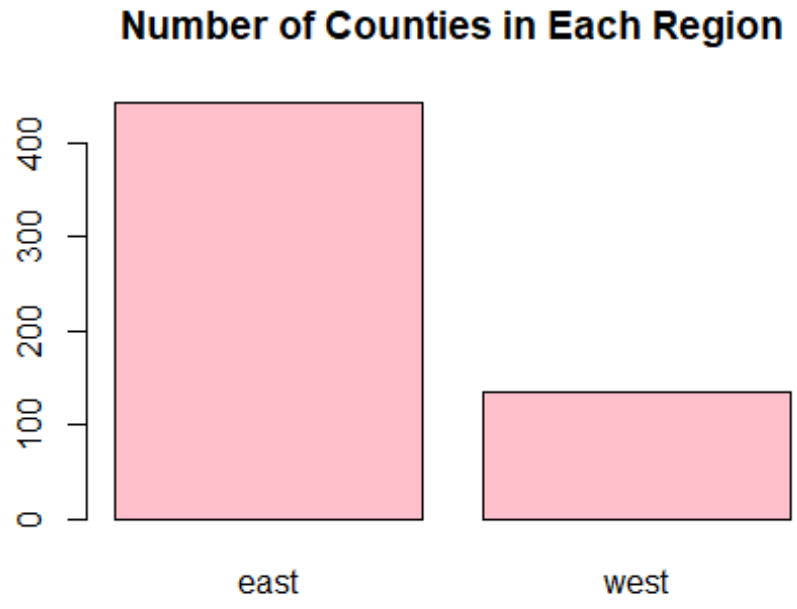
```r
boxplot(pollution$pm25, col = "blue") # Boxplot
abline(h = 12) # Crea una linea horizontal en el numero que le indico (En el cual me interesa)
```




```r
hist(pollution$pm25, col = "green")
abline(v = 12, lwd = 2, col = "red") #Crea una linea pero ahora vertical, v es la posicion, lwd el ancho o grosor
abline(v = median(pollution$pm25), col = "magenta", lwd = 4)
abline(h = 50, lwd = 4, col = "blue") # h= horizontal, v = vertical
rug(pollution$pm25)
```

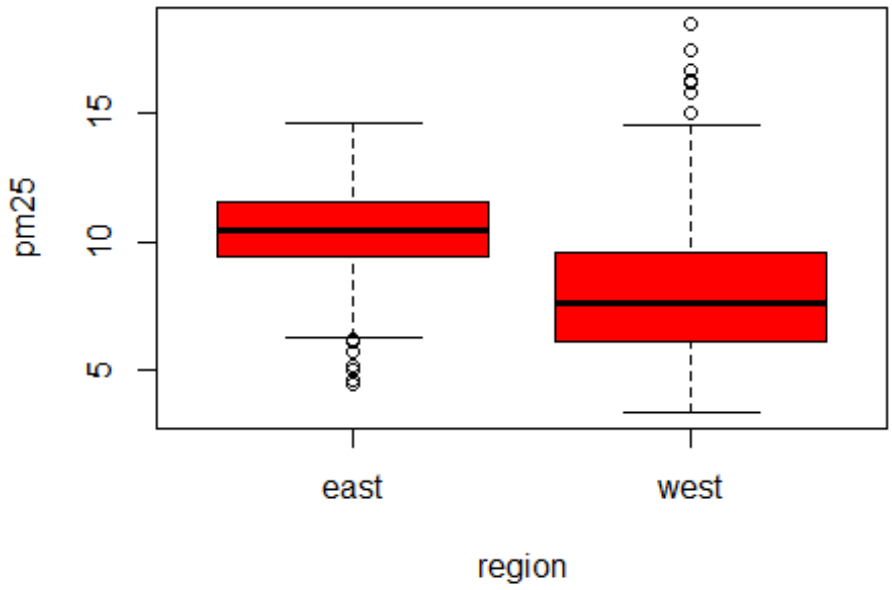



```r
barplot(table(pollution$region), col = "pink", main = "Number of Counties in Each Region") # main permite darle titulo al grafico (En general se utiliza para graficar variables categoricas "factors")
```

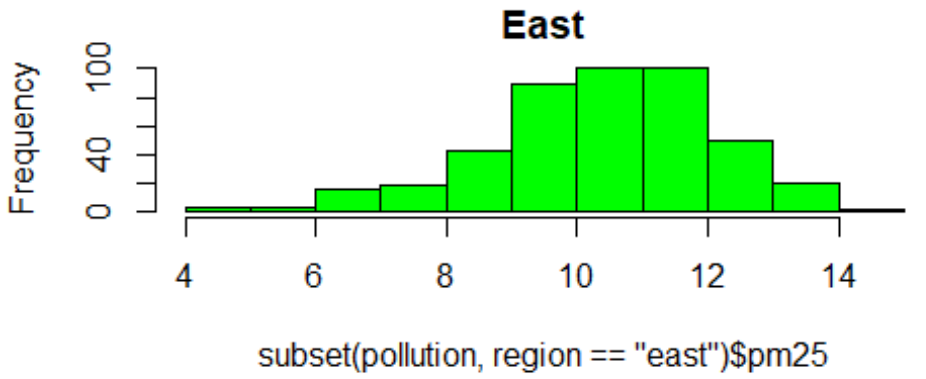


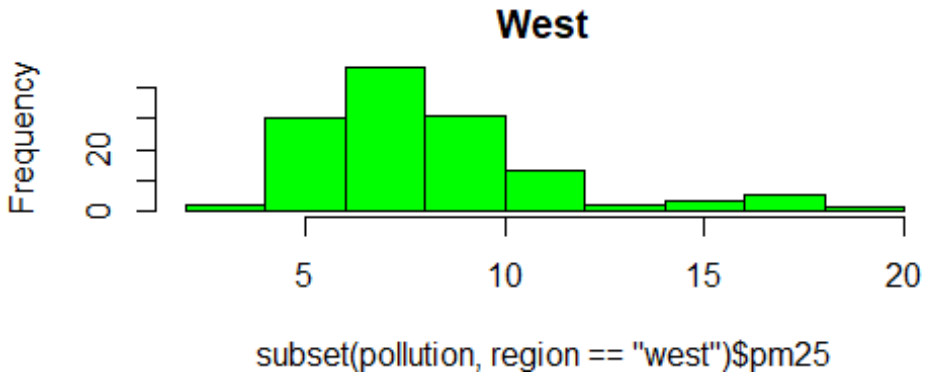
Exploratory Graphs (part2)

```r
boxplot(pm25 ~ region, data = pollution, col = "red") # data permite utilizar las variables por su nombre y poder utilizar la cola de chancho para graficar mas de una variable a la vez
```

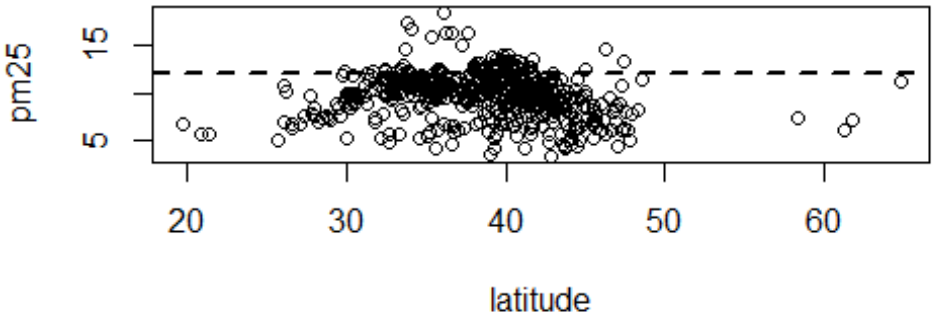


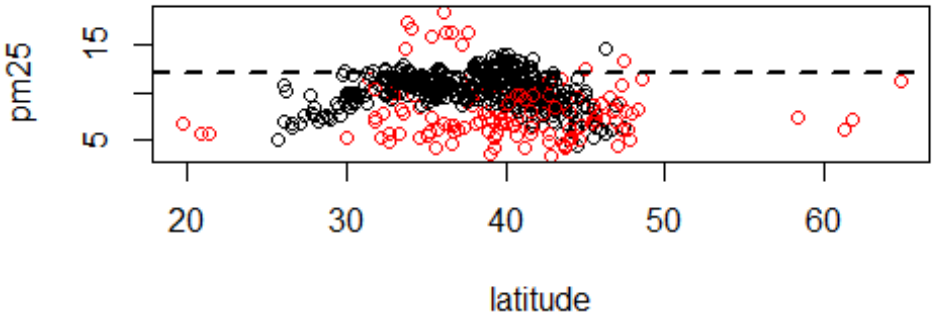

```r
par(mfrow = c(2,1), mar = c(4,4,2,1)) # Permite graficar en una foto dos graficos, mfrow = Las dimensiones de los graficos en este caso 2 linea con 1 grafico, mar = Me da los margenes de los histogramas para que queden mejor posicionados
hist(subset(pollution, region == "east")$pm25, col = "green", main = "East") #Creo un histograma solo con las observaciones east
hist(subset(pollution, region == "west")$pm25, col = "green", main = "West") #Creo un histograma solo con las observaciones west
```





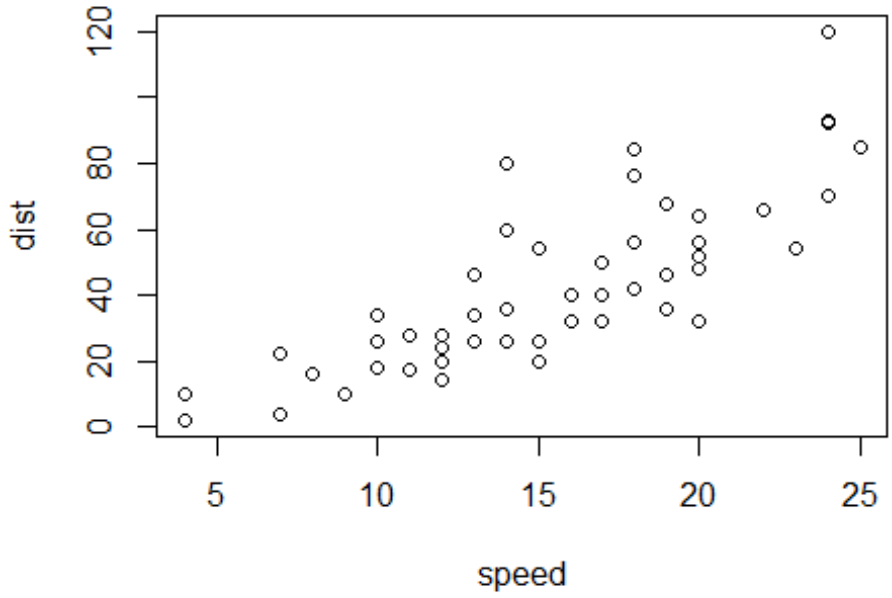

```r
with(pollution, plot(latitude,pm25)) #Plot es un grafico de puntos, (with permite utilizar los nombres de las variables de pollution y no tener que llamar a pollution$pm25)
abline(h = 12, lwd = 2, lty = 2) # lty nos permite puntear la linea
with(pollution, plot(latitude,pm25, col = region)) # col permite diferencia por region
abline(h = 12, lwd = 2, lty = 2)
```



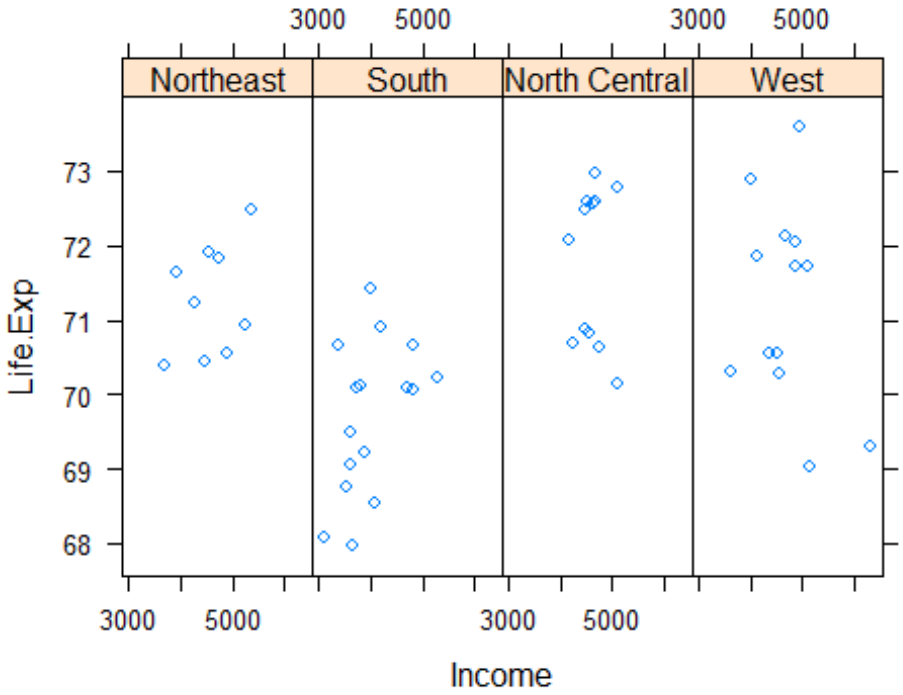


Plotting Systems in R

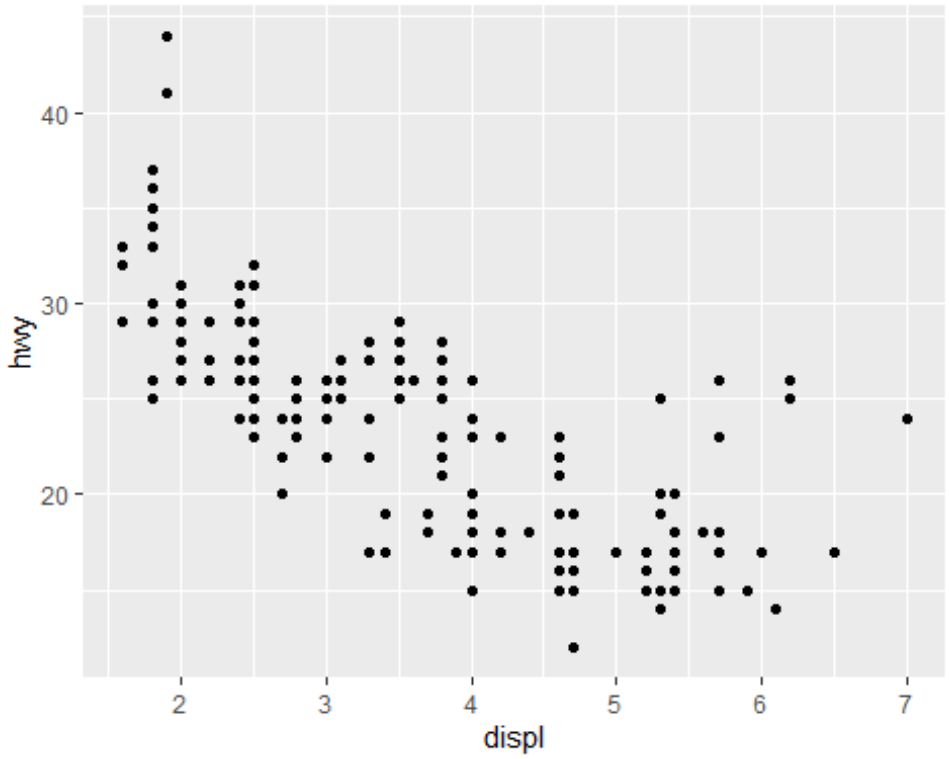
```
#Basic Plot
library(datasets)
data(cars)
with(cars, plot(speed,dist))
#The Lattice System
library(lattice)
```




```
state <- data.frame(state.x77, region = state.region)
xyplot(Life.Exp ~ Income | region, data = state, layout = c(4,1))
```

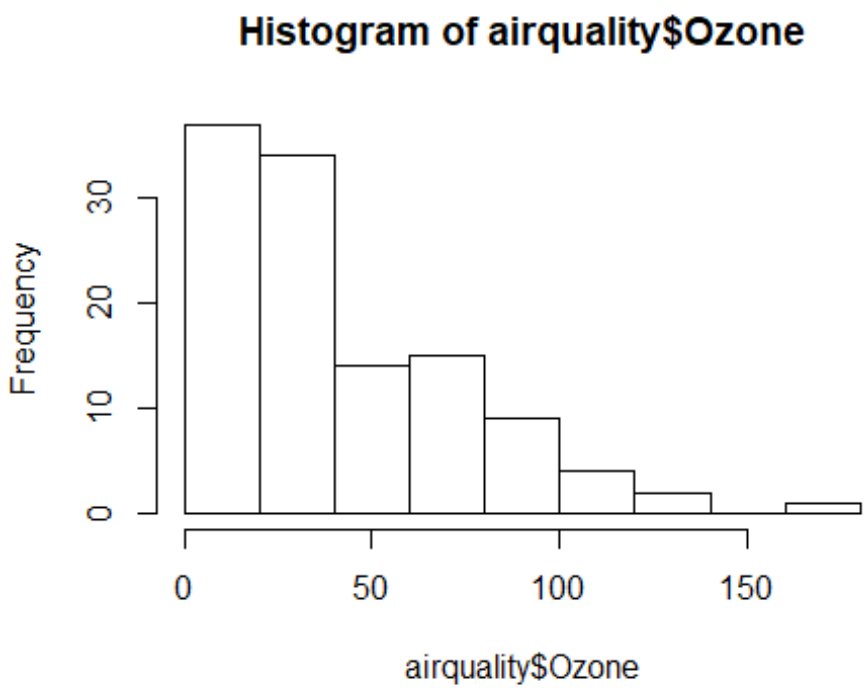


```
## ggplot2 System
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.6.1

data(mpg)
qplot(displ, hwy, data = mpg)
```

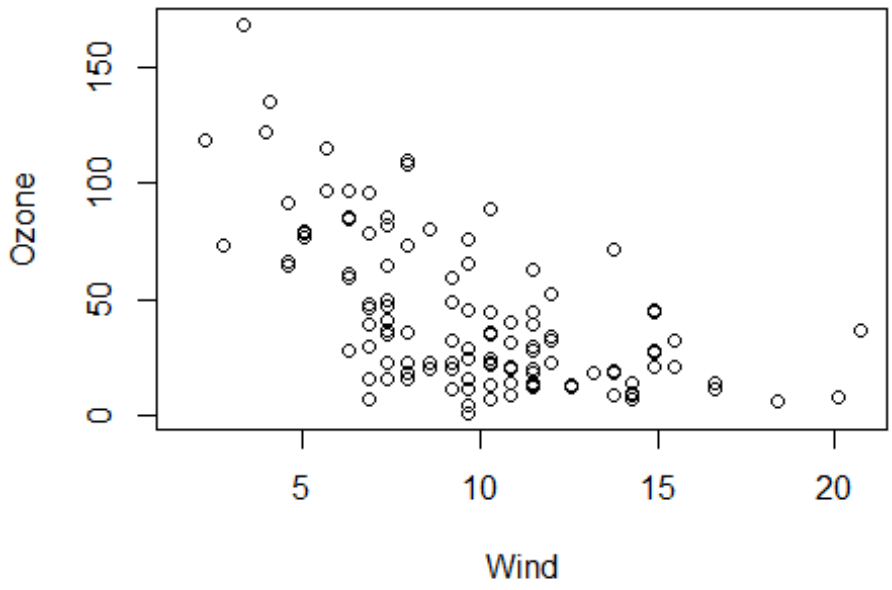


Base Plotting System (part 1)

```
library(datasets)
hist(airquality$Ozone)
```

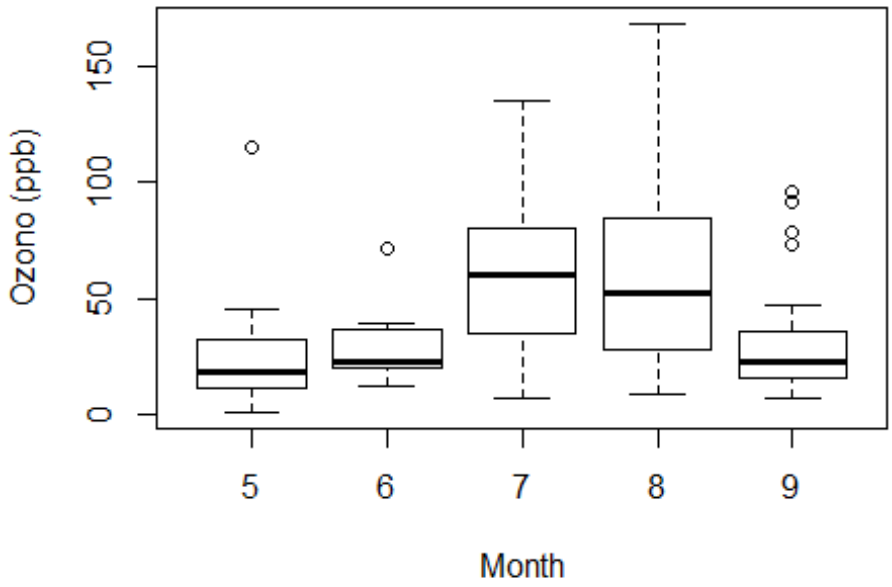



```r
with(airquality, plot(Wind,Ozone))
```




```r
airquality <- transform(airquality, Month = (factor(Month))) # Convierte los meses en factores
```

```r
boxplot(Ozone ~ Month, airquality, xlab = "Month", ylab = "Ozono (ppb)")
# Grafica el Ozono en cada mes un boxplot separados
```



Los graficos pueden tener una infinidad de parametros pero es importante recordar estos:

- pch: Cambia el simbolo que utiliza, default son circulos. Puedo entregarle numeros o caracteres (“H”, “A”, 1)
- lty: El tipo de linea que se utiliza, puede ser punteada etc…
- lwd: Permite controlar el ancho de las lineas que se utilizan
- col: Es el color
- xlab: Nombre de la variable x
- ylan: Nombre de la variable y
- main: Nombre del grafico

La funcion par() permite setear los parametros globales de cualquier grafico, una vez usado sirve para todos los graficos y se puede sobreescribir:

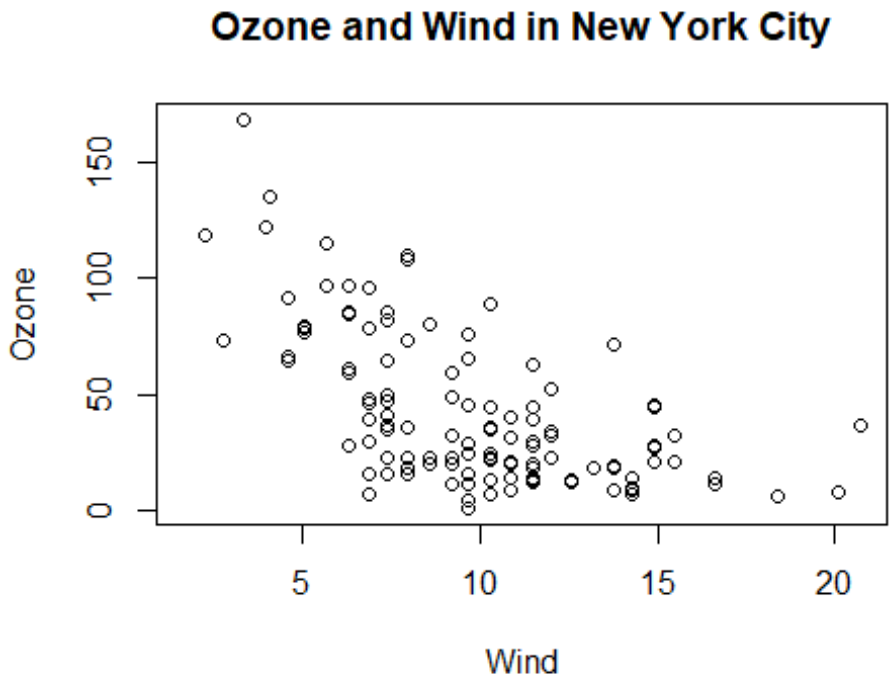
- las: La orientacion de la etiqueta en los ejes
- bg: El color de fondo
- mar: Los margenes
- oma: El margen exterior
- mfrow: Numero de graficos por (fila,columna) (Los graficos se llenan por filas)
- mfcol: Numero de graficos por (fila,columna) (Los graficos se llenan por columnas)

Base Plotting System (part 2)

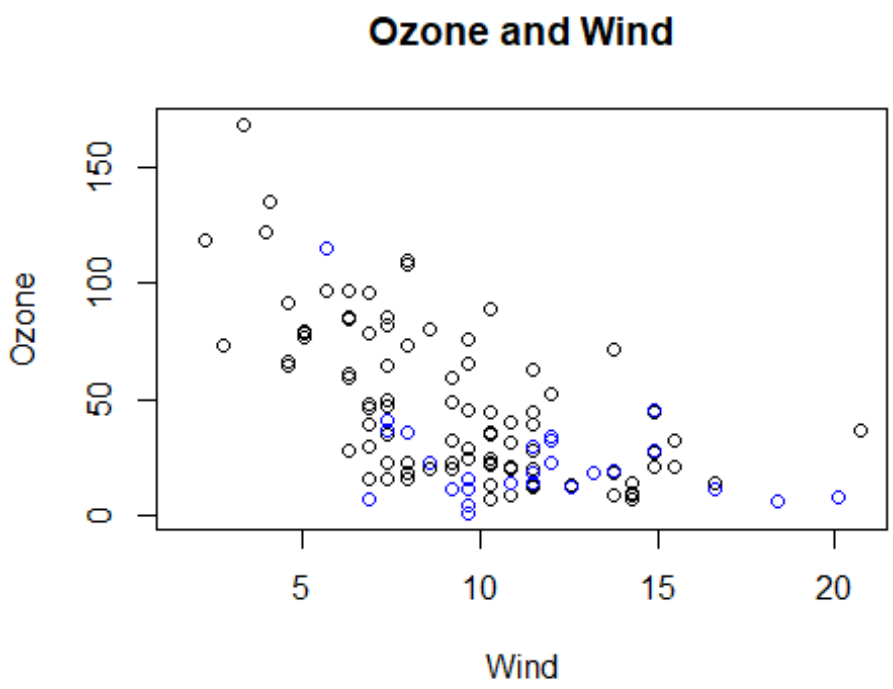
Funciones importantes del Base Plotting:

- plot: Permite realizar un grafico de puntos, sin embargo, dependiendo del tipo de variables puede ser otra cosa
- lines: Conecta los puntos de un grafico de puntos
- points: Agrega puntos a un graficos
- text: Agrega textos de los graficos
- title: Cambia los titulos de los ejes, del grafico, subtitulos
- mtext: Agrega texto a los margenes
- axis: Agrega marcas de ejes o etiquetas

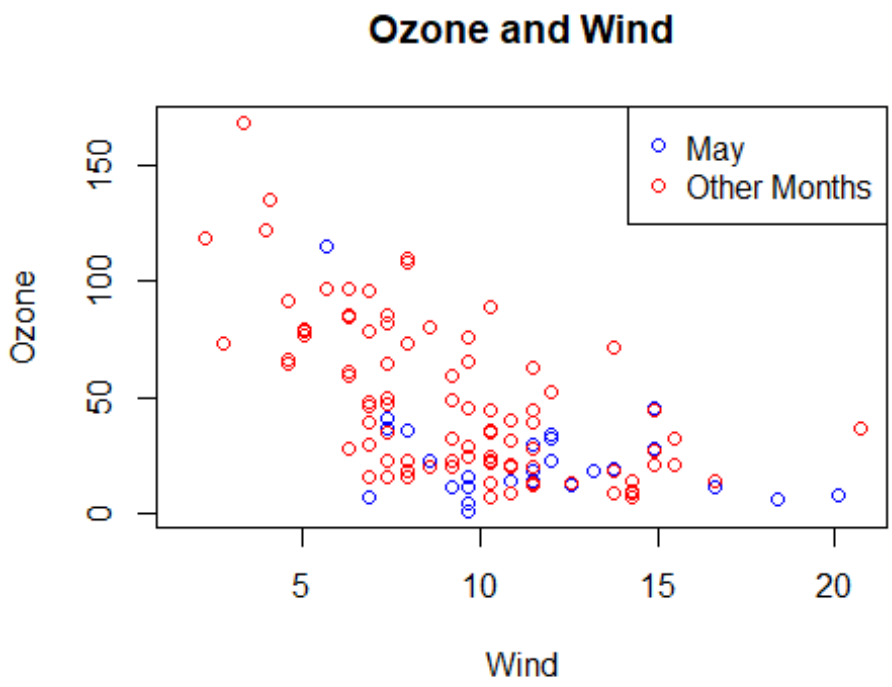
```
library(datasets)
with(airquality, plot(Wind,Ozone))
title(main = "Ozone and Wind in New York City") #Agrega titulo
```



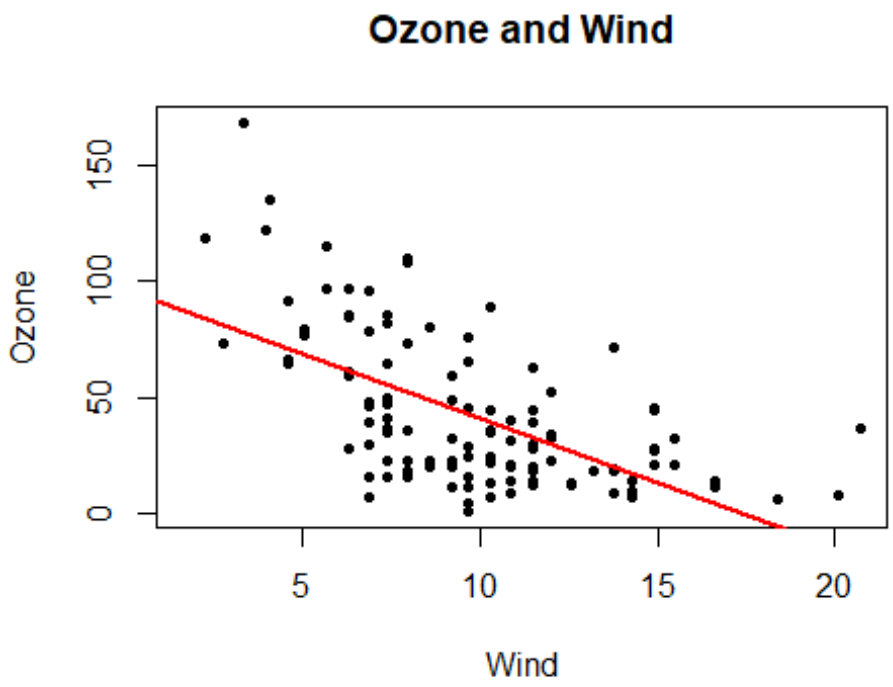

```
with(airquality, plot(Wind,Ozone, main = "Ozone and Wind")) #Incluye
titulo
with(subset(airquality, Month == 5), points(Wind,Ozone, col = "blue"))
#Incluyo los puntos del mes 5 en azul
```



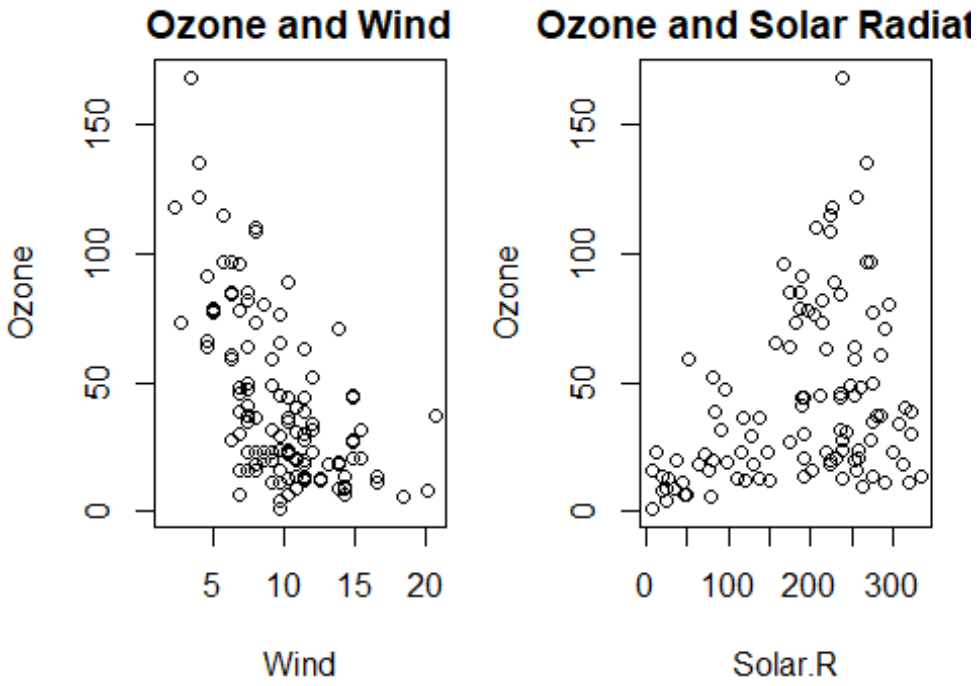

```r
with(airquality, plot(Wind,Ozone, main = "Ozone and Wind", type = "n"))
#Ploteo el grafico, type = "n"  significa que no agrega nada dentro del grafico para despues crear los puntos por separado
with(subset(airquality, Month == 5), points(Wind,Ozone,col="blue")) # Agrego los puntos azules al mes 5
with(subset(airquality, Month != 5), points(Wind,Ozone,col="red"))
#Agrego los puntos rojos a los otro meses
legend("topright", pch = 1, col = c("blue","red"), legend = c("May", "Other Months")) #Creo la leyenda, pch = tipo de simbolo (circulo, tringualo), col = color. legend = titulos de las leyendas
```



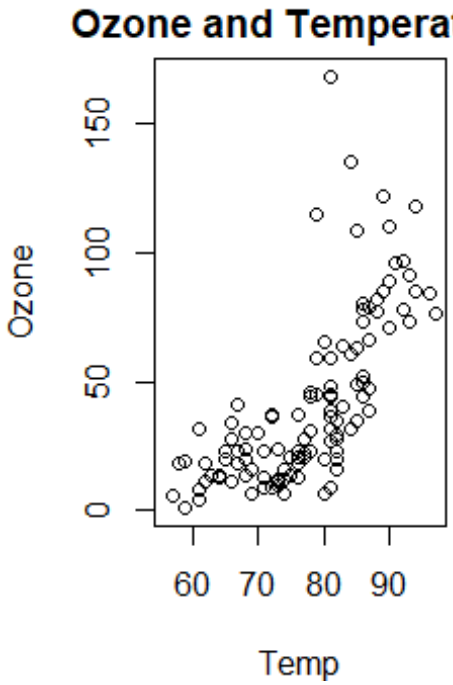

```r
with(airquality, plot(Wind,Ozone, main = "Ozone and Wind", pch =20))
model <- lm(Ozone ~ Wind, airquality) #Creo un modelo de regresion lineal que se ajuste a los datos
abline(model, lwd = 2, col = "red") #Agrego una linea que es igual al modelo de color rojo
```




```r
par(mfrow = c(1,2))
with(airquality, {
        plot(Wind,Ozone, main = "Ozone and Wind")
        plot(Solar.R, Ozone, main = "Ozone and Solar Radiation") #Con la
funcion par puedo poner dos o mas graficos
})
par(mfrow = c(1,2), mar = c(4,4,2,1), oma = c(0,0,2,0))
with(airquality, {
        plot(Wind,Ozone, main = "Ozone and Wind")
        plot(Solar.R, Ozone, main = "Ozone and Solar Radiation") #Con la
funcion par puedo poner dos o mas graficos
        plot(Temp, Ozone, main = "Ozone and Temperature")
        mtext("Ozone and Weather in New York City", outer = TRUE) #mtext
permite poner un texto general al ploteo, si uso la funcion main es para
cada uno de los graficos.
})
```



Ozone and Weather in New York City



Graphics Devices in R (part 1)

```
library(datasets)
pdf(file = "myplot.pdf") #Crea un archivo pdf
with(faithful, plot(eruptions,waiting))# Plotea
title(main = "Old Faithgul Geyser data")# Agrega titulo
dev.off()# Cierra el pdf y lo guarda, grafic devices

## png
##   2
```

Graphics Devices in R (part 2)

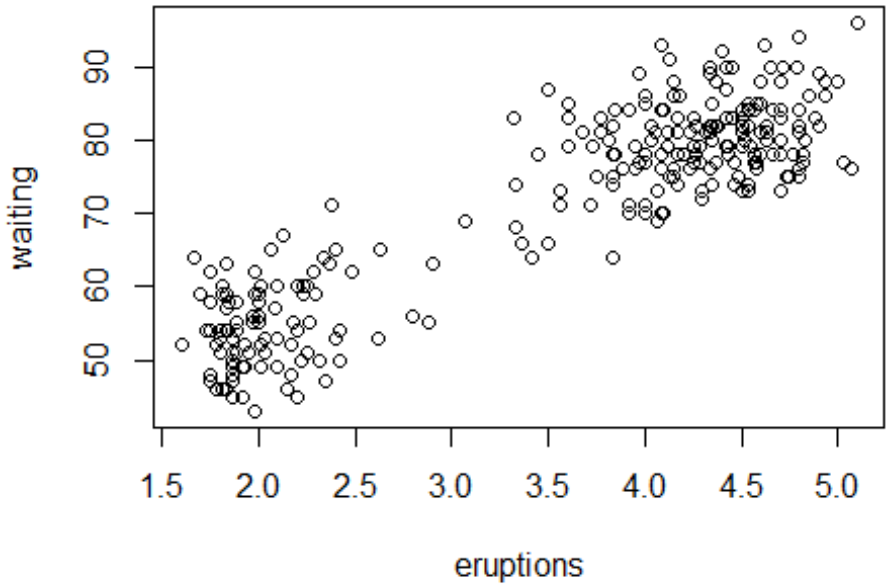
Existen dos tipos basic para guardar archivos: Vector devices and Bitmap Devices

- Vector: Grafico de lineas, sin imagenes o fotos. No sufre la calidad de la imagen (Si los graficos tienen demasiados puntos no es conveniente usar este tipo)

```
    + pdf: Clasico
    + svg: Graficos para web (Aguanta animaciones)
    + win.metafile: Predecesor del pdf
    + postscript
```

- Bitmap:

```
    + png: Bueno para guardar graficos que tienen demasiados puntos
    + jpeg: Para fotos
    + tiff: Formato antiguo
    + bmp: un formato nativo de windows usado para los iconos

with(faithful, plot(eruptions,waiting))
```




```
dev.copy(png, file = "eruptions.png") #Esto es mas usado, copia el
grafico que se abre en windows y lo convierte en pdf, asi se puede ver
que se esta incluyendo en el pdf
```

```
## png
##   3
```

```
dev.off()
```

```
## png
##   2
```

Week 2

Tomás

6 de diciembre de 2019

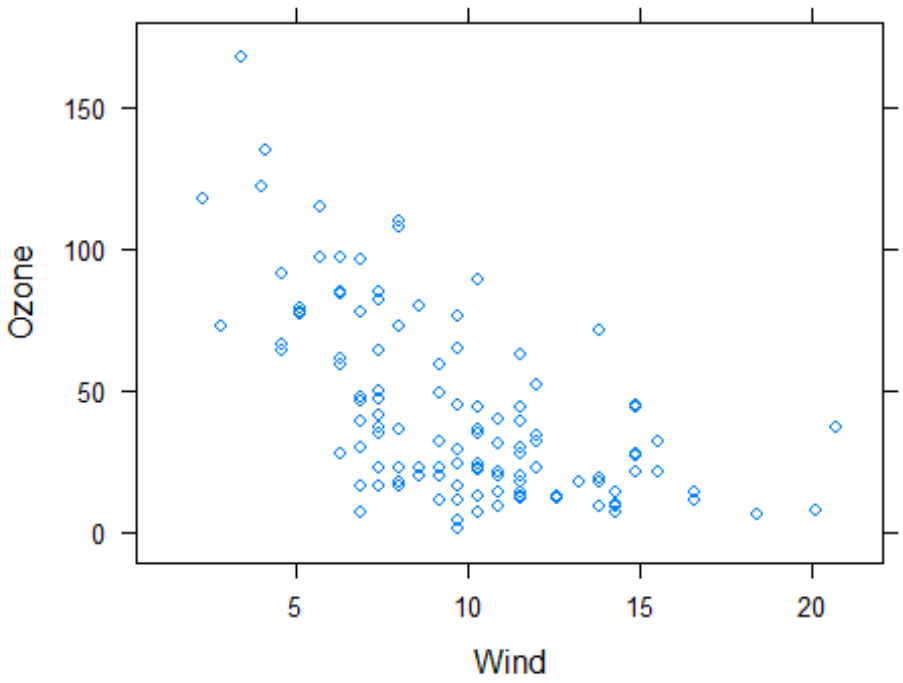
Lattice Plotting System (part 1)

Las funciones mas imporatnte de Lattices son:

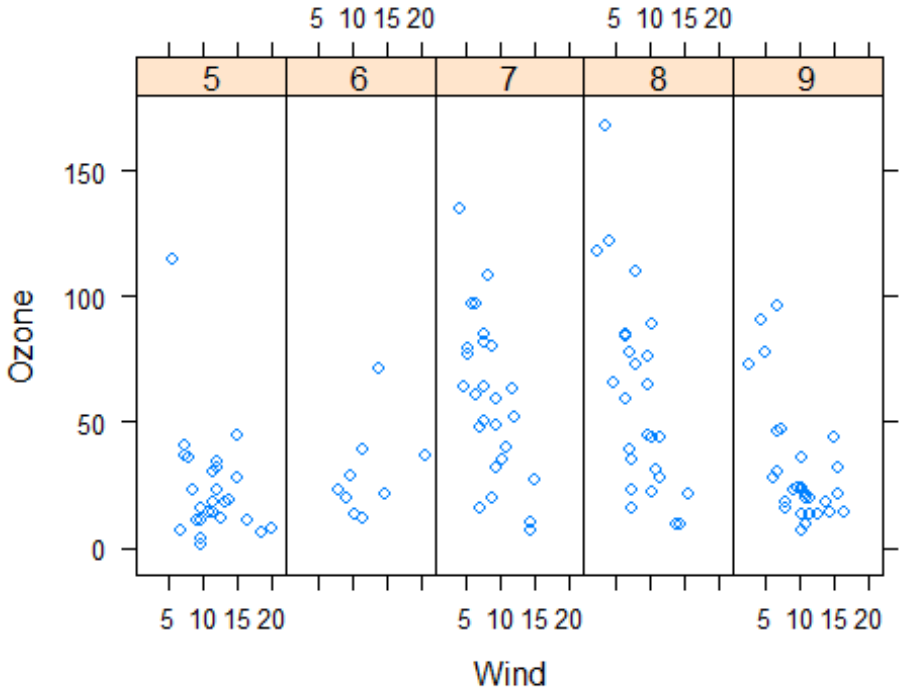
- xyplot: Genera graficos de dispersion o puntos
- bwplot: boxplot

- histogram: Histogramas
- stripplot: Diagrama de cajas pero usa puntos
- dotplot: Traza puntos “lineas”
- splom: Matriz de graficas de dispersion (Es como par en el base plot)
- levelplot, contourplot: Graficar datos de imagenes

```
library(datasets)
library(lattice)
xyplot(Ozone ~ Wind, data = airquality) # xyplot(y~x, data)
```

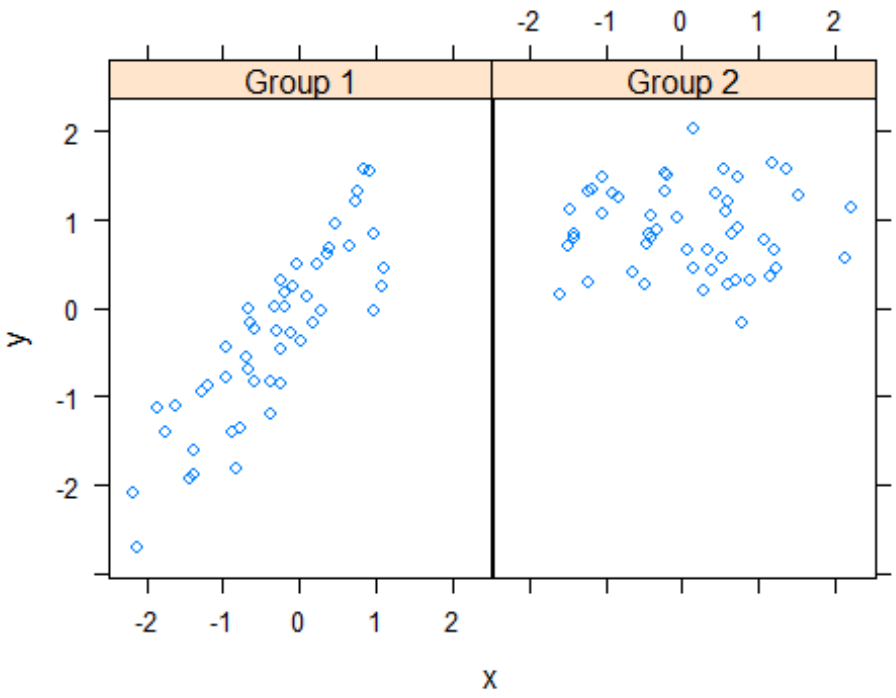



```
airquality <- transform(airquality, Month = factor(Month))
xyplot(Ozone ~ Wind | Month, data = airquality, layout = c(5,1)) # La
barra permite utilizar categorias para que me las grafique por separado,
layout = en que tipo de matriz quiero mostrar los graficos (5 columnas
por 1 fila)
```

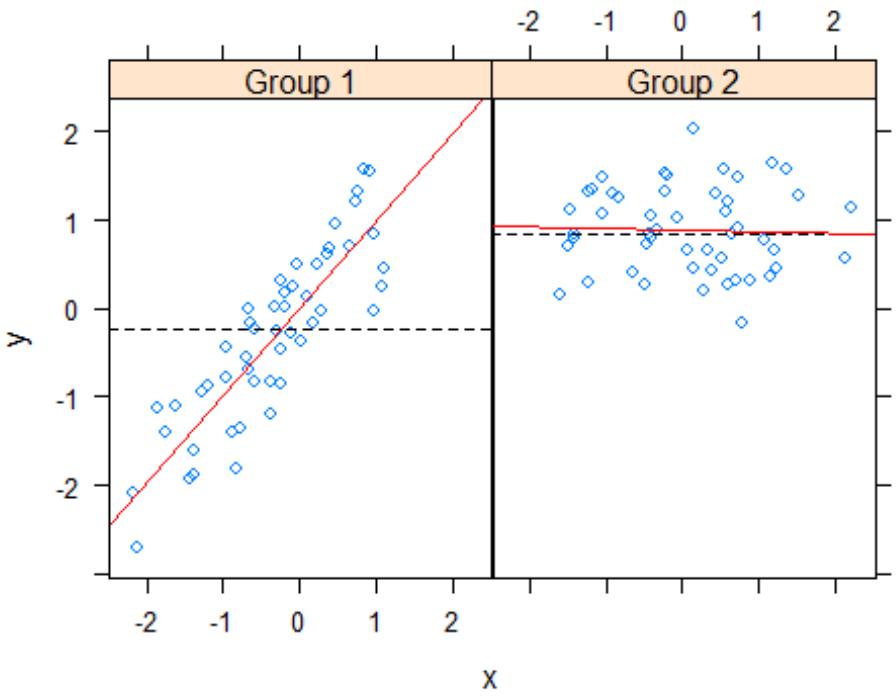


Lattice Plotting System (part 2)

```
set.seed(10)
x <- rnorm(100)
f <- rep(0:1, each = 50)
y <- x+f-f*x+rnorm(100, sd = 0.5)
f <- factor(f,labels = c("Group 1", "Group 2"))
xyplot(y ~ x | f, layout = c(2,1)) #Plot normal
```

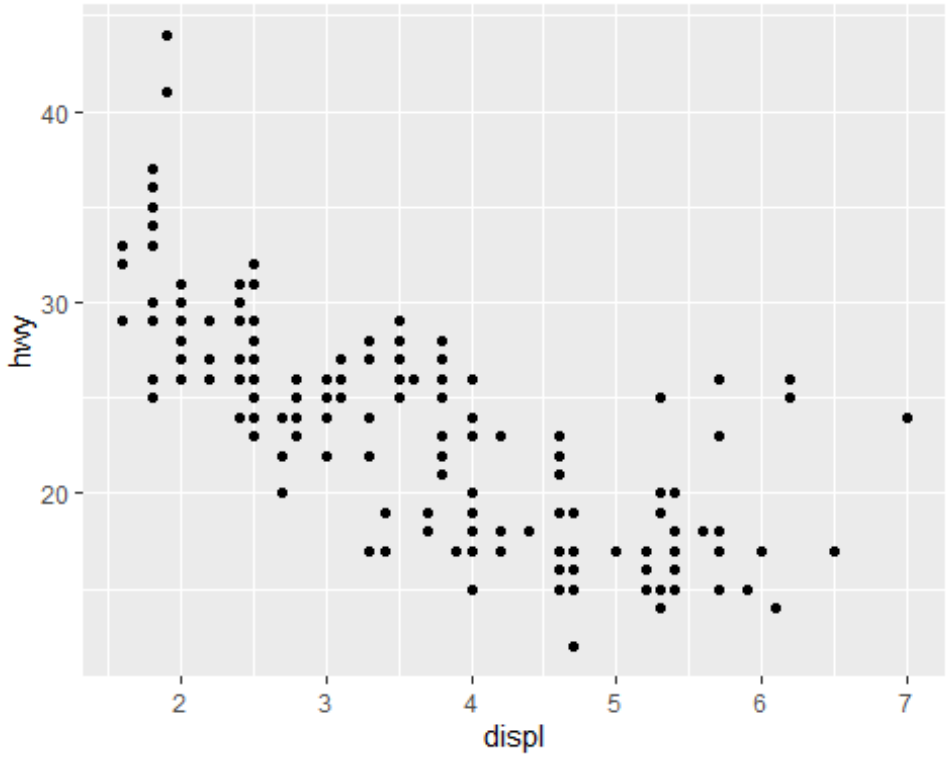



```
xyplot( y ~ x | f, panel = function(x,y,...){ # La funcion panel me
permite editar el plot
        panel.xyplot(x,y,...) # Llamo a la funcion xyplot para crear los
puntos
        panel.abline(h = median(y), lty = 2) #Creo una linea horizontal
igual a la media (lty me da las lineas punteadas)
        panel.lmline(x , y, col = 2) #Una simple regresion lineal para
los datos
})
```

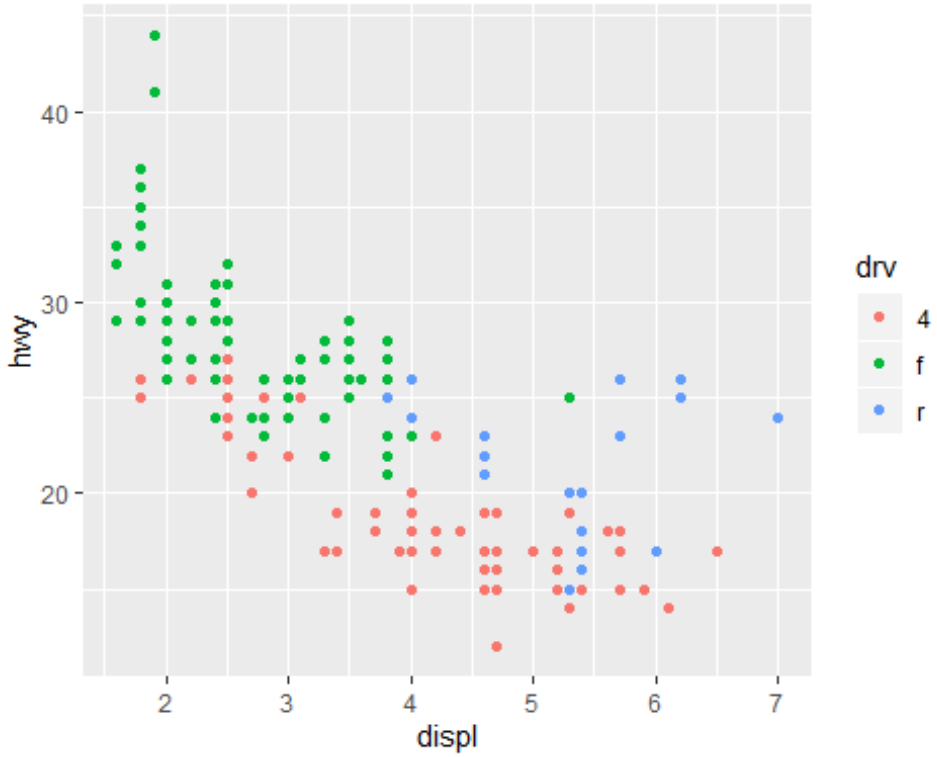


ggplot2 (part 2)

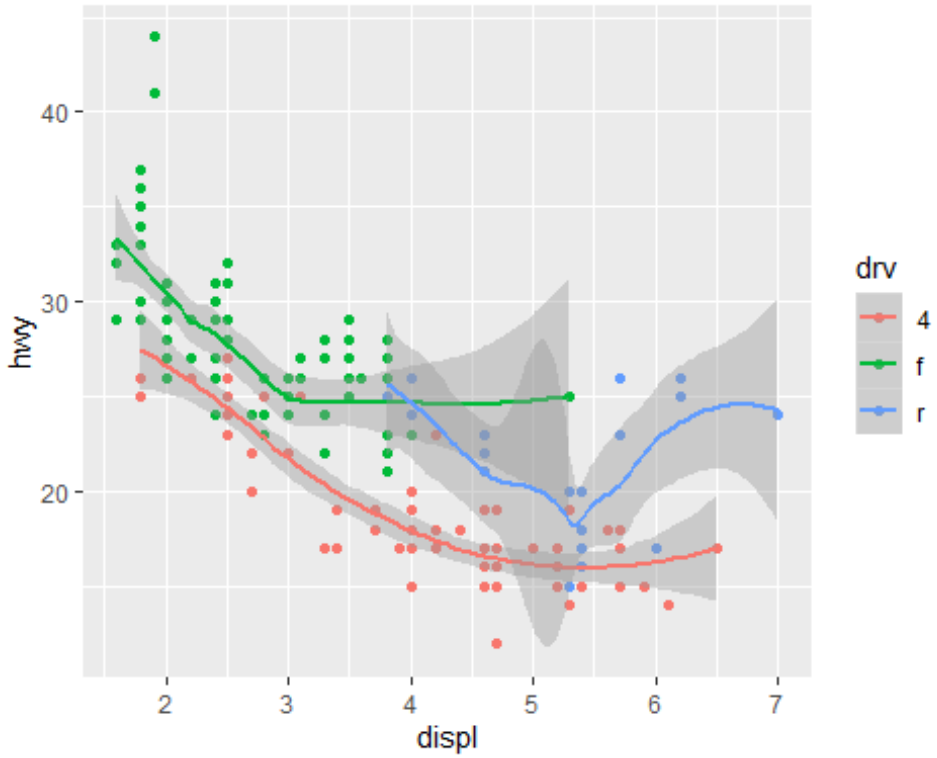
```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.6.1

qplot(displ,hwy, data = mpg) # qplot(x,y,data)
```



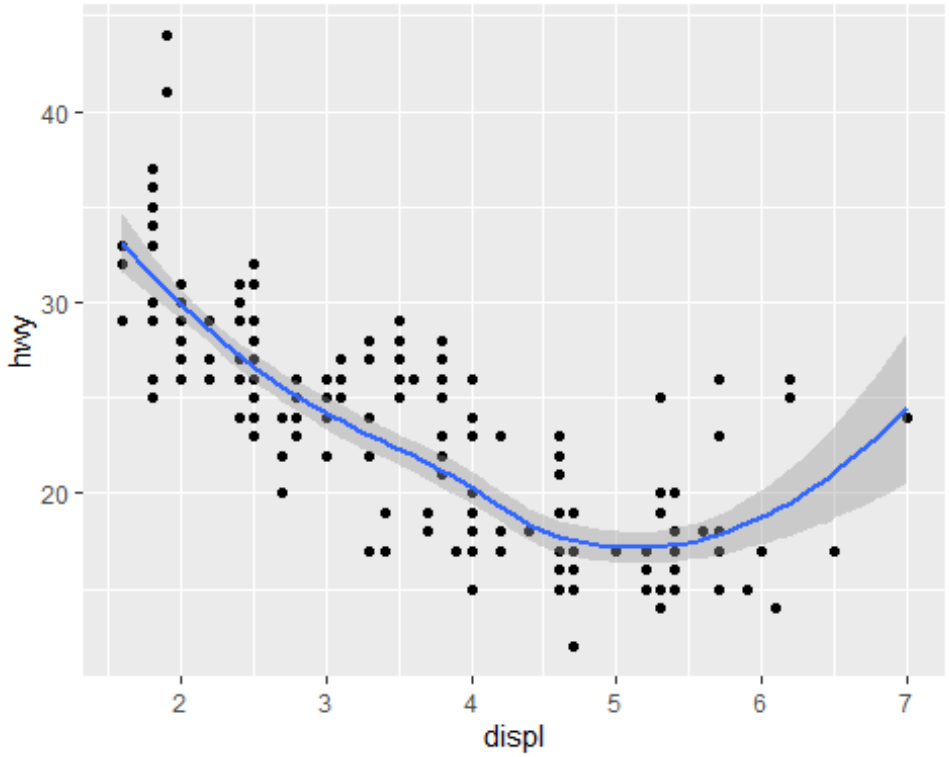

```r
qplot(displ,hwy, data = mpg, color = drv) # Puedo destacar subgrupos (factores) con color = factor
```



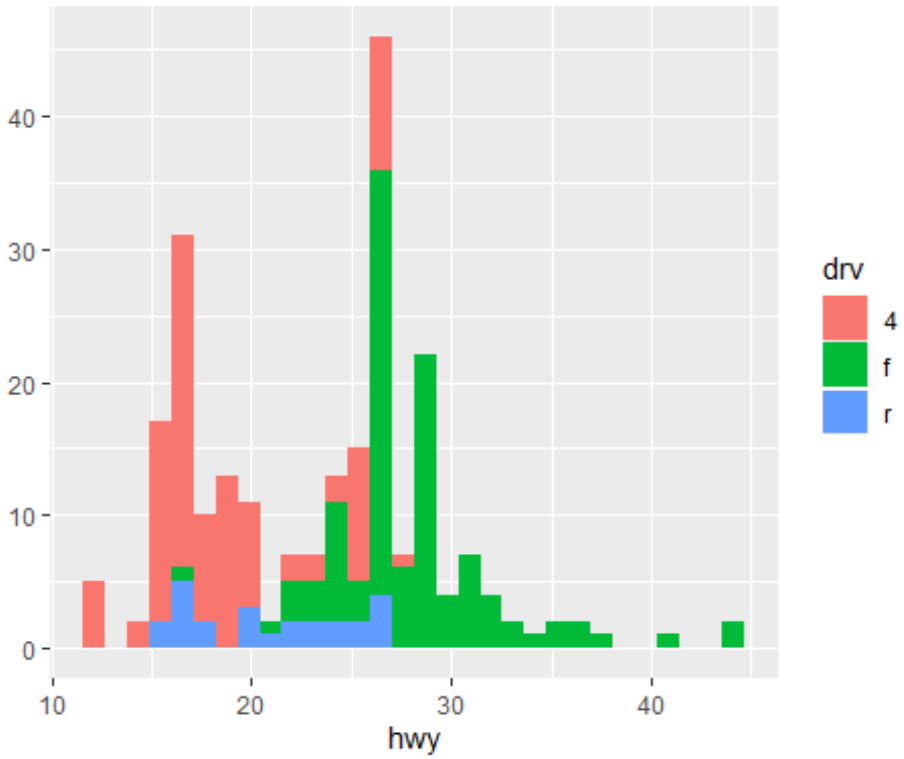
```r
qplot(displ,hwy, data = mpg, color = drv, geom = c("point", "smooth"))
#Puedo crear un suavizamiento de los datos por tipo

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



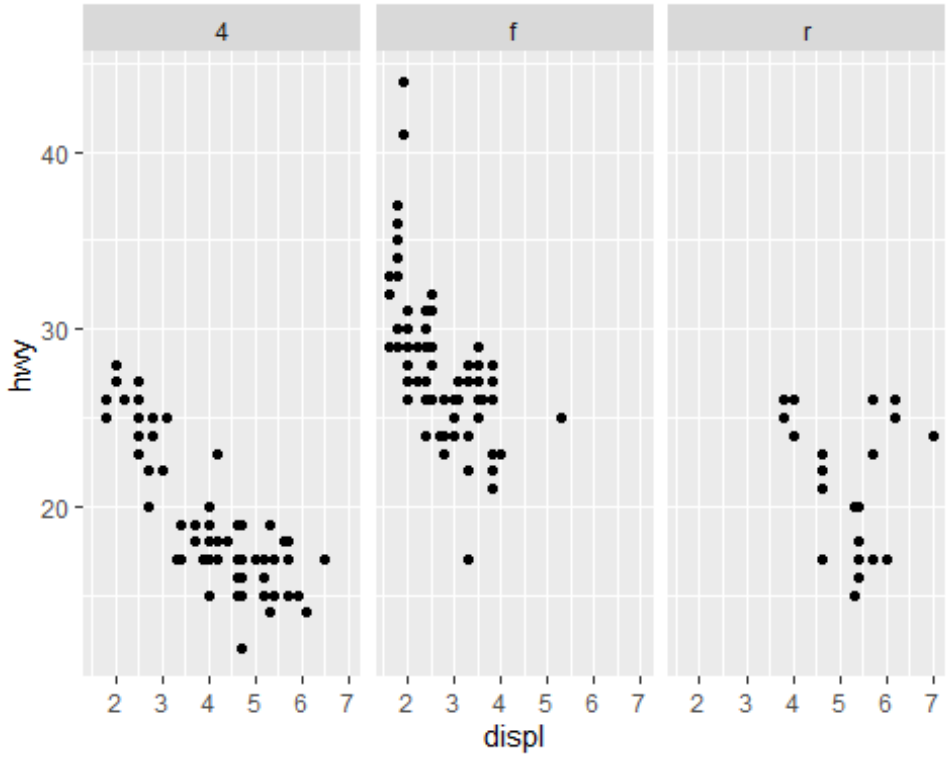

```r
qplot(displ,hwy, data = mpg, geom = c("point", "smooth")) #Aqui lo puedo tener de manera general, la zona gris me indica el 95% de confiabilidad

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



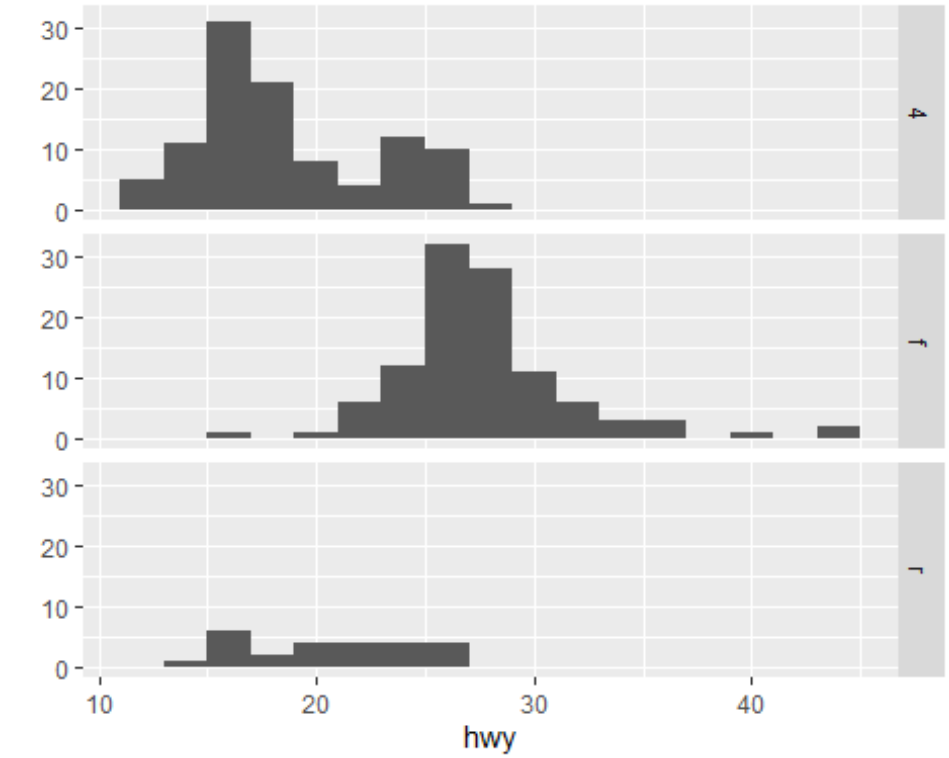

```
qplot(hwy, data = mpg, fill = drv) #Puedo crear un histograma
especificando solamente 1 variables qplot(x,data), aqui los separo por
tipos como antes

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



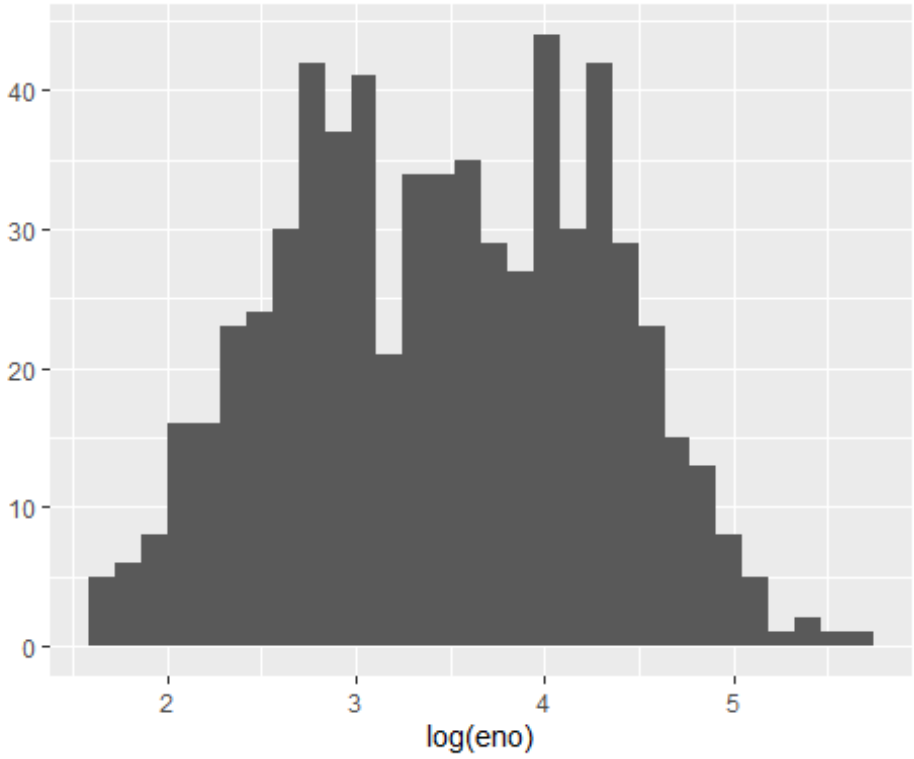

```r
qplot(displ, hwy, data = mpg, facets = .~drv) # Facets permite generar ventanas con varios graficos (El argumento se separa por ~, si pongo la variable categorica al lado derecho obtengo solo filas, al derecho solo columnas)
```



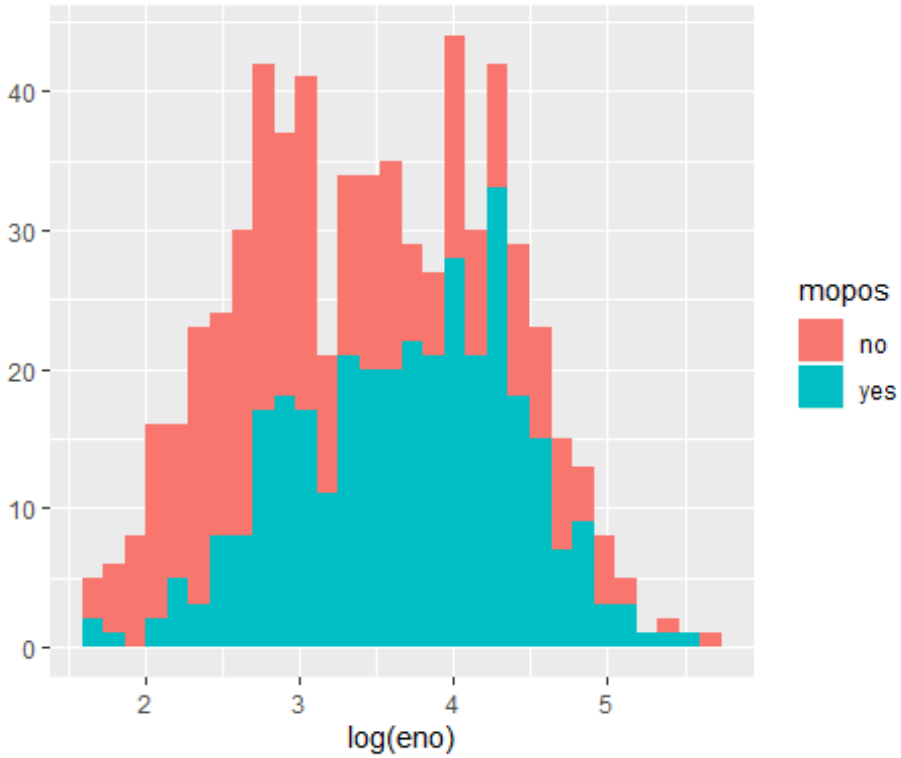

```
qplot(hwy, data = mpg, facets = drv~., binwidth = 2)
```



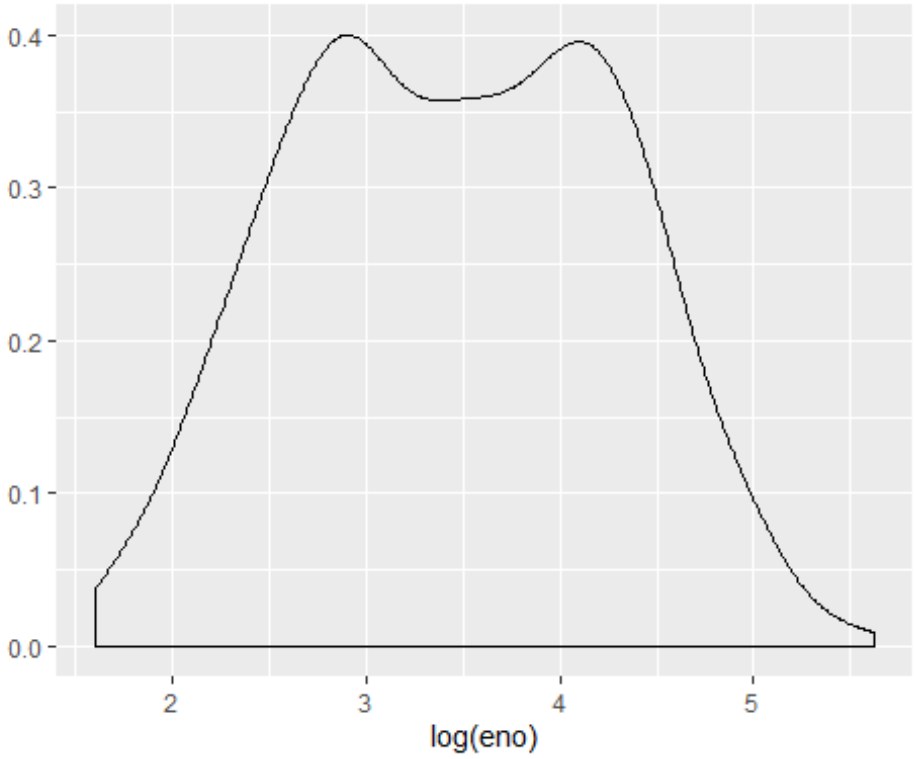

```
maacs <- read.csv("maacs.csv")
qplot(log(eno), data = maacs, na.rm =TRUE, bins = 30)
```



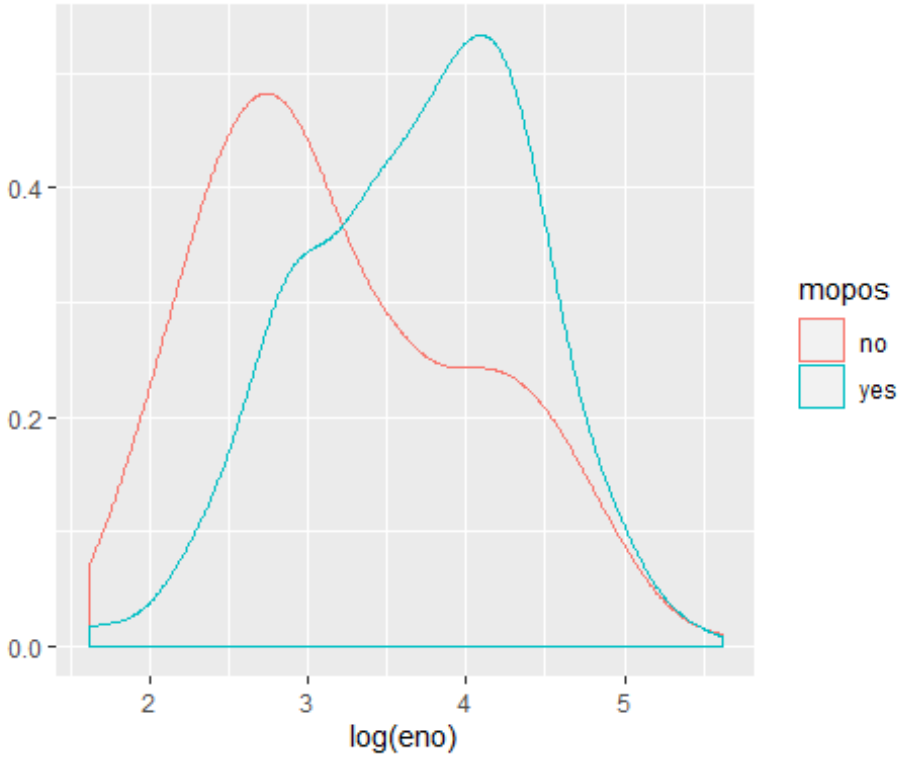

```r
qplot(log(eno), data = maacs, na.rm =TRUE, bins = 30, fill = mopos)
```



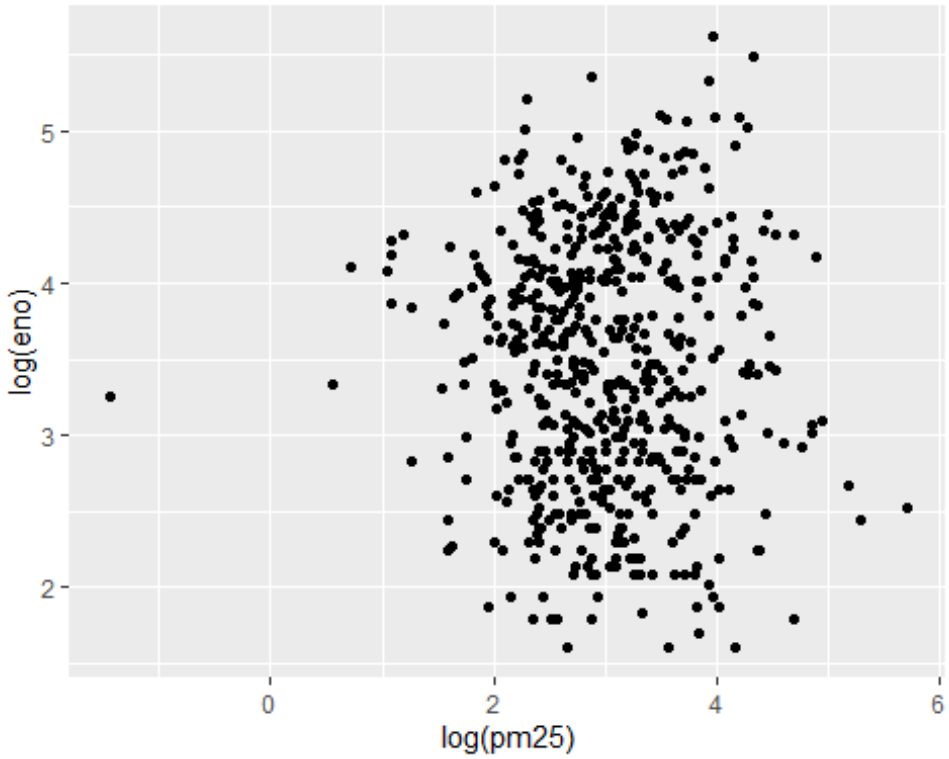

```r
qplot(log(eno), data = maacs , geom = "density", na.rm = TRUE) # Crea un grafico de densidad
```



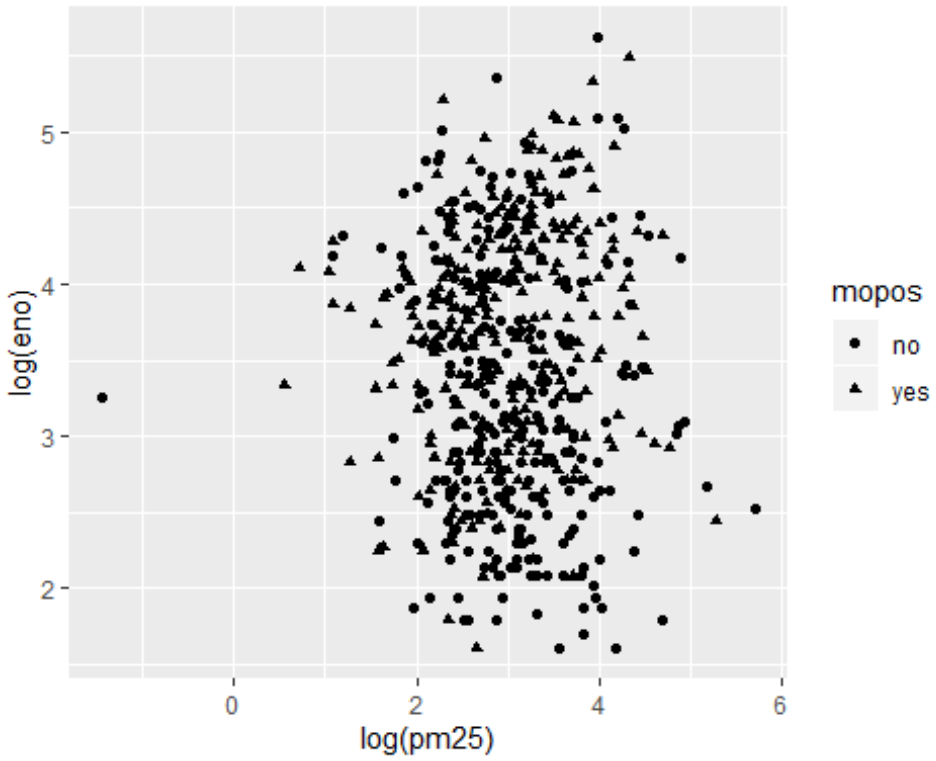

```
qplot(log(eno), data = maacs , geom = "density", na.rm = TRUE, color =
mopos) #Crea el grafico de densidad por tipo
```



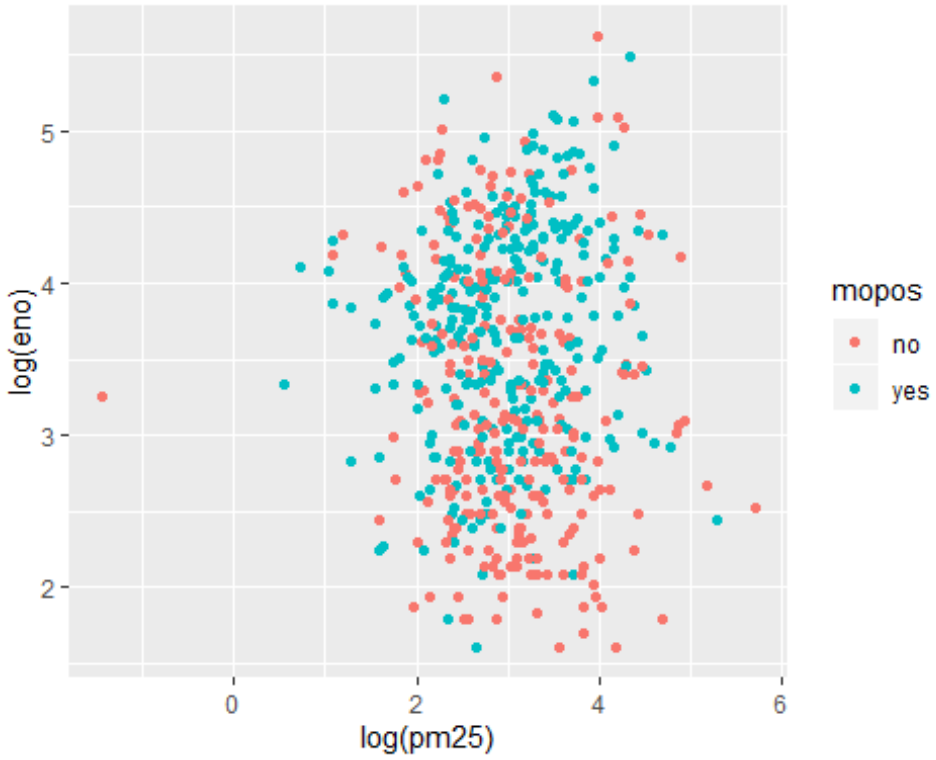
```
qplot(log(pm25), log(eno), data = maacs, na.rm = TRUE) #Grafico de
dispersion entre pm25 y eno
```




```
qplot(log(pm25), log(eno), data = maacs, na.rm = TRUE, shape = mopos)
#Shapes permite cambiar las figuras
```

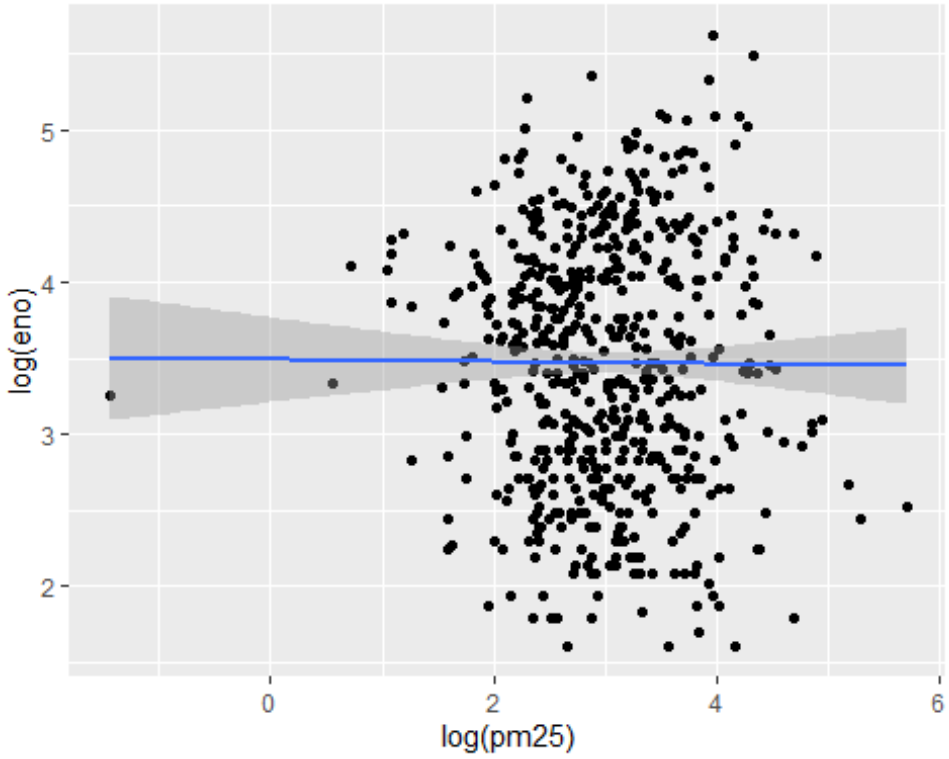



```
qplot(log(pm25), log(eno), data = maacs, na.rm = TRUE, color = mopos)
```

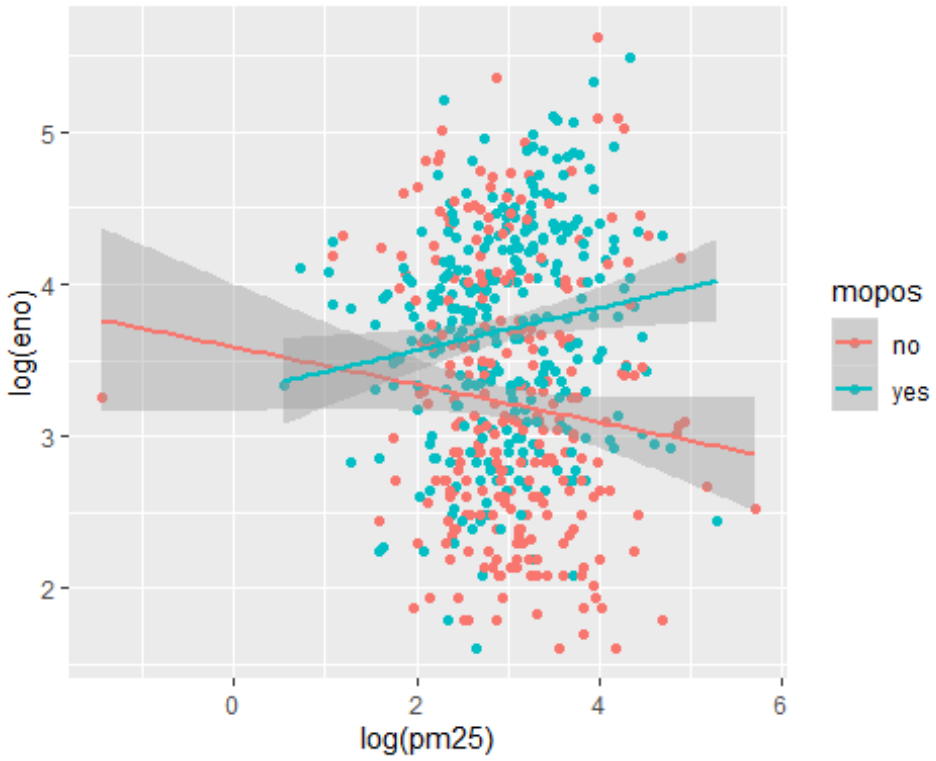



```
qplot(log(pm25),log(eno), data = maacs, na.rm = TRUE) +
geom_smooth(method = "lm") #Agrego una regresion lineal
```

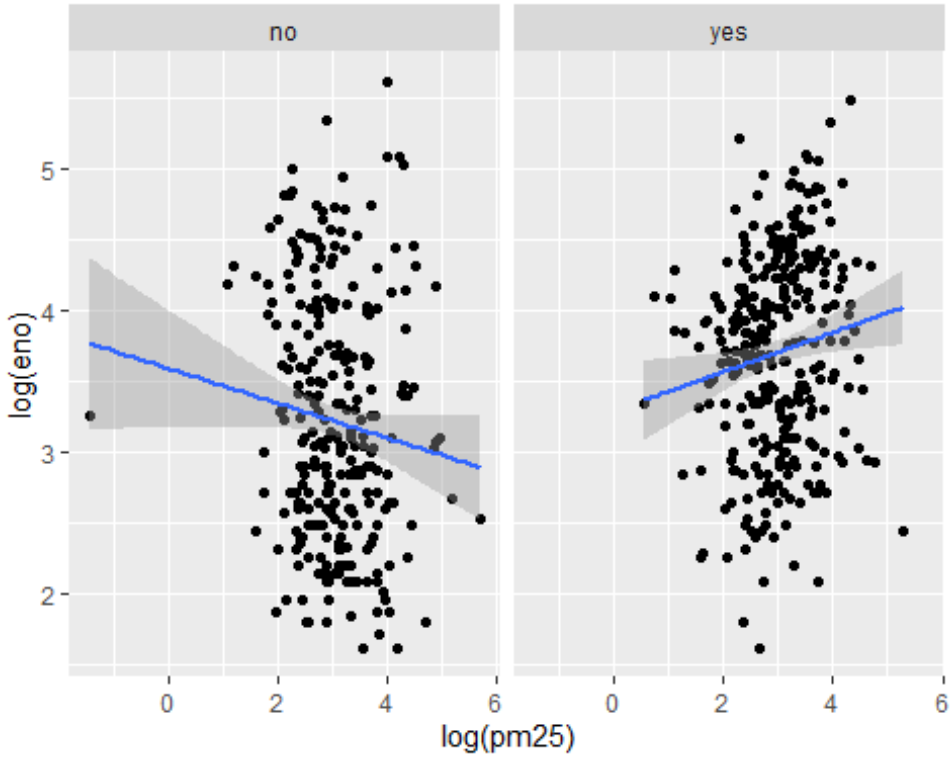
```
## Warning: Removed 184 rows containing non-finite values (stat_smooth).
```




```
qplot(log(pm25),log(eno), data = maacs, color = mopos, na.rm = TRUE) +
geom_smooth(method = "lm", na.rm = TRUE) #La misma regresion pero por
grupos
```




```
qplot(log(pm25), log(eno), data = maacs, facets = .~mopos, na.rm = TRUE)
+ geom_smooth(method = "lm", na.rm = TRUE)
```

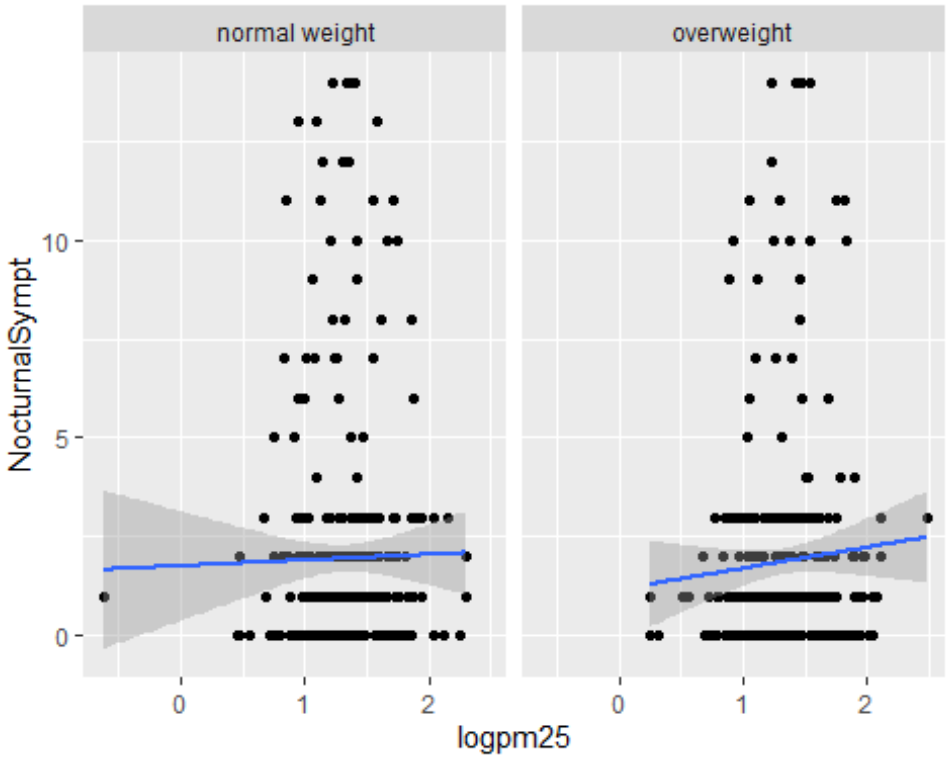


ggplot2 (part 3)

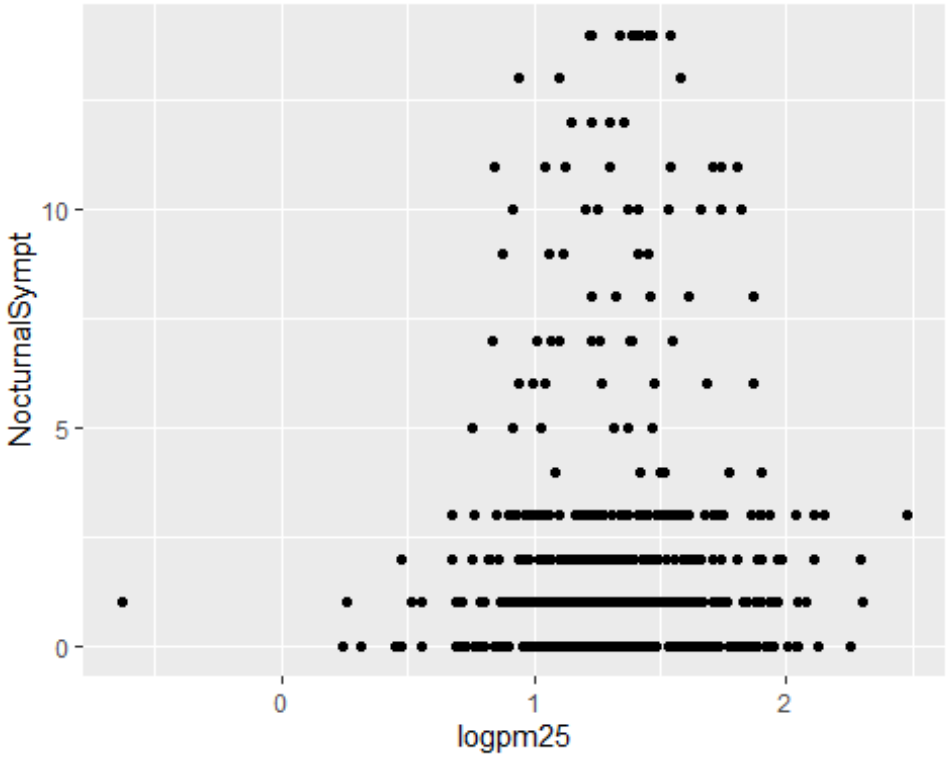
Los componentes basicos de ggplot2 son:

- dataframe: Datos
- aesthetic mappings: Como los datos son mapeados, color, tamaño, posicion
- geoms: figuras geometricas a usar
- facets: Para crear mas de un grafico
- stats: Para realizar transformaciones como suavizamientos, quantiles, regresiones
- scales: Que escala usa aesthetic maps (male = red, female = blue)
- coordinate system

```
library(ggplot2)
qplot(logpm25, NocturnalSympt, data = maacs, facets = .~bmicat, method =
"lm", na.rm = TRUE, geom = c("point", "smooth"))

## Warning: Ignoring unknown parameters: method
```




```
# Manera mas lenta paso a paso
g <- ggplot(maacs[,c(6,7,8)], aes(logpm25, NocturnalSympt), na.rm = TRUE)
#Creo el grafico
g + geom_point(na.rm = TRUE) #Le agrego los puntos
```

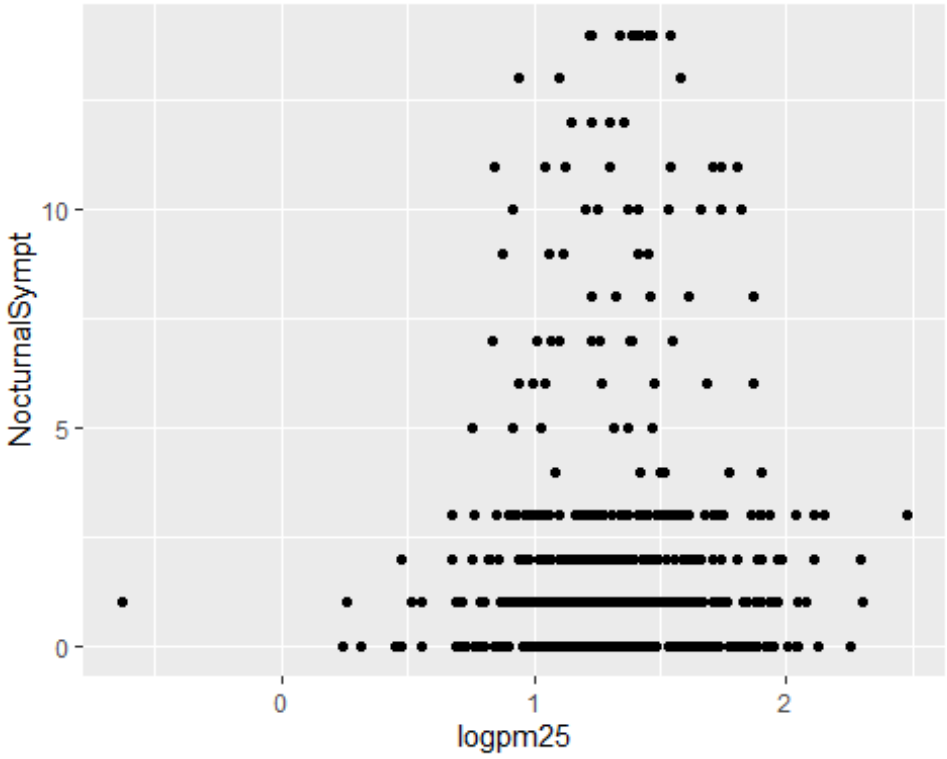


ggplot2 (part 4)

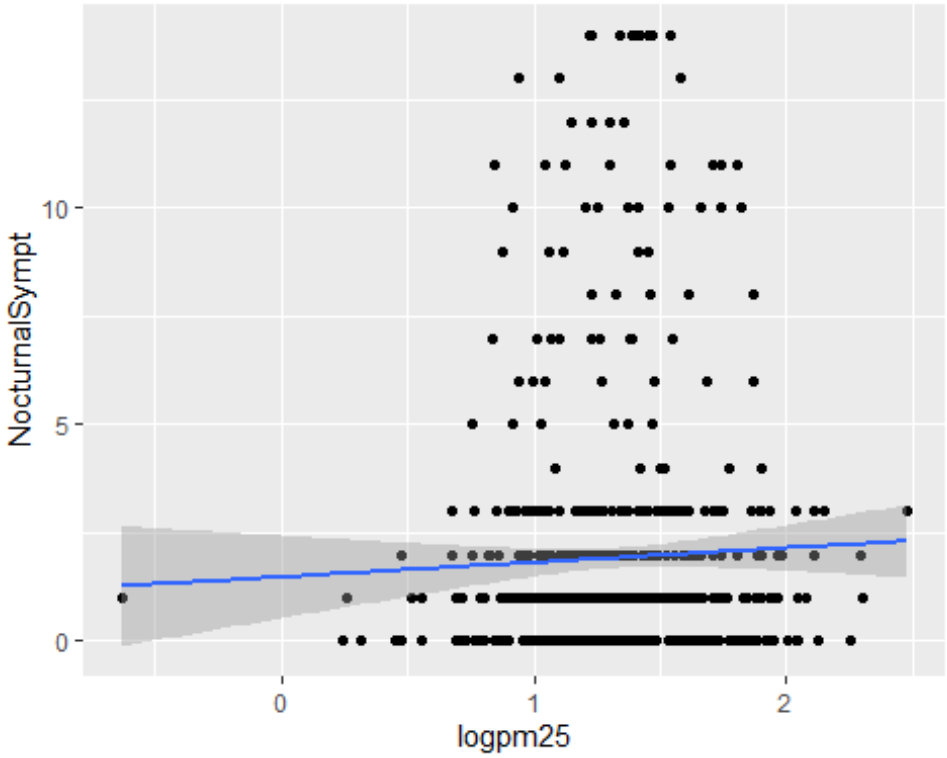
Algunas cosas extras que siempre son utiles:

- xlab()
- ylab()
- labs()
- ggtitle()
- geom(): Existen muchas variaciones de este no solo “points”, “smooth”
- themes(): Permite editar colores, position de la legenda entre otras cosas (El “tema”)
- theme_gray(): El tema default (fondo gris)
- theme_bw(): Fondo blanco y negro

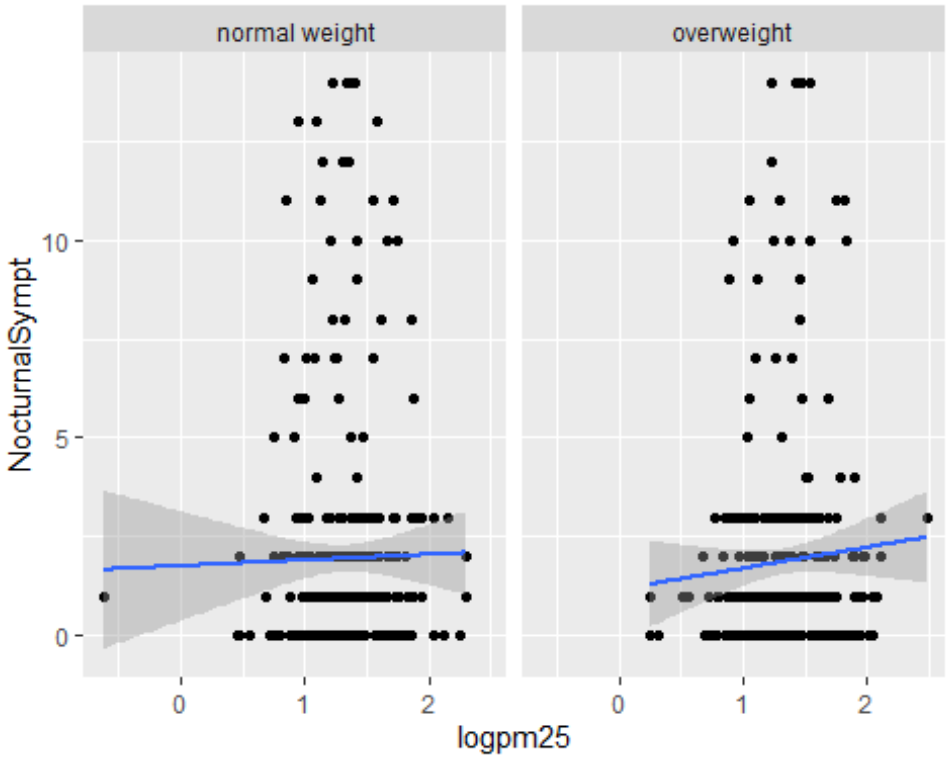
```
g <- ggplot(maacs[,c(6,7,8)], aes(logpm25, NocturnalSympt), na.rm = TRUE)
#Creo el grafico
g + geom_point(na.rm = TRUE) #Le agrego los puntos
```



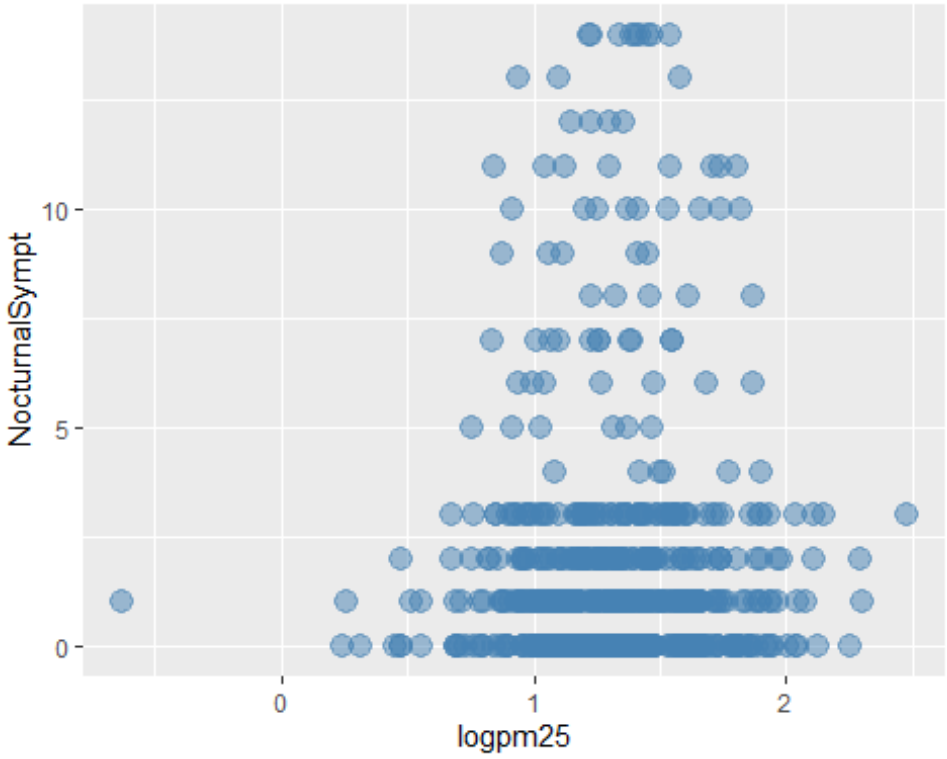

```r
g + geom_point(na.rm = TRUE) + geom_smooth(method = "lm", na.rm = TRUE)
#Creo una regresion lineal
```



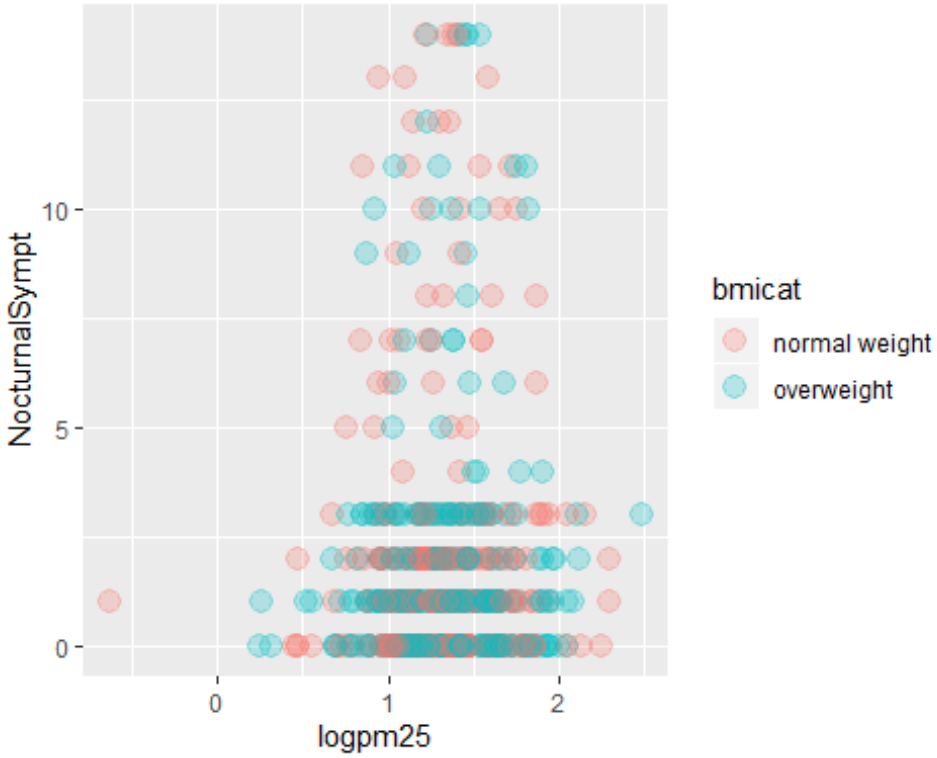
```
g + geom_point(na.rm = TRUE) + geom_smooth(method = "lm", na.rm = TRUE) +
facet_grid(.~bmicat) #Agrego mas de un plot
```



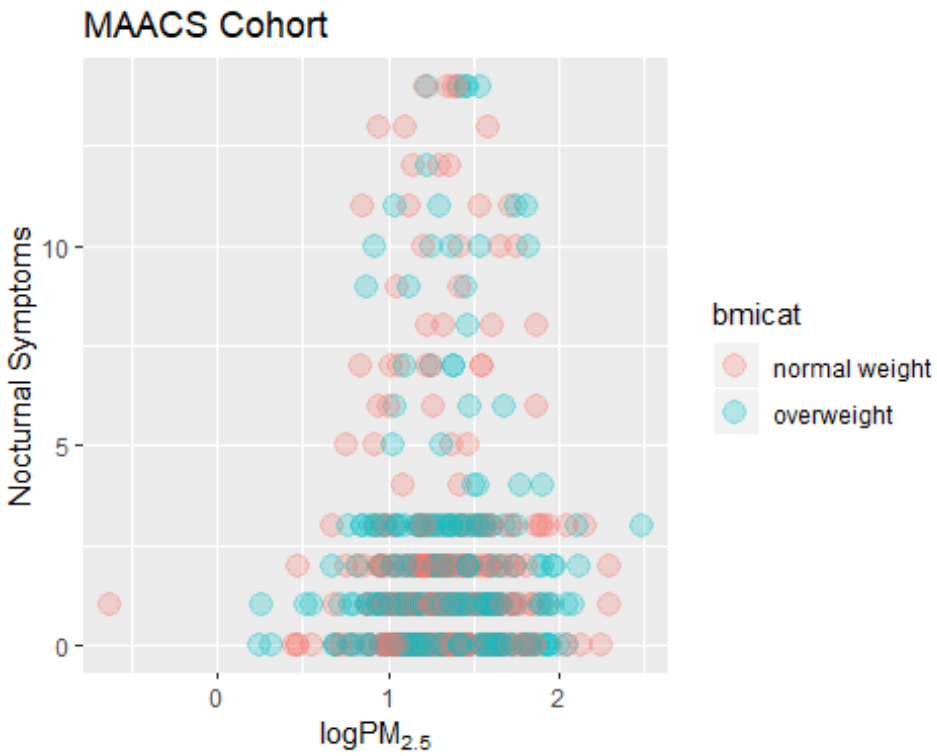

```
g + geom_point(color = "steelblue", size = 4, alpha = 1/2, na.rm = TRUE)
# Cambio de color, tamaño, transparencia
```



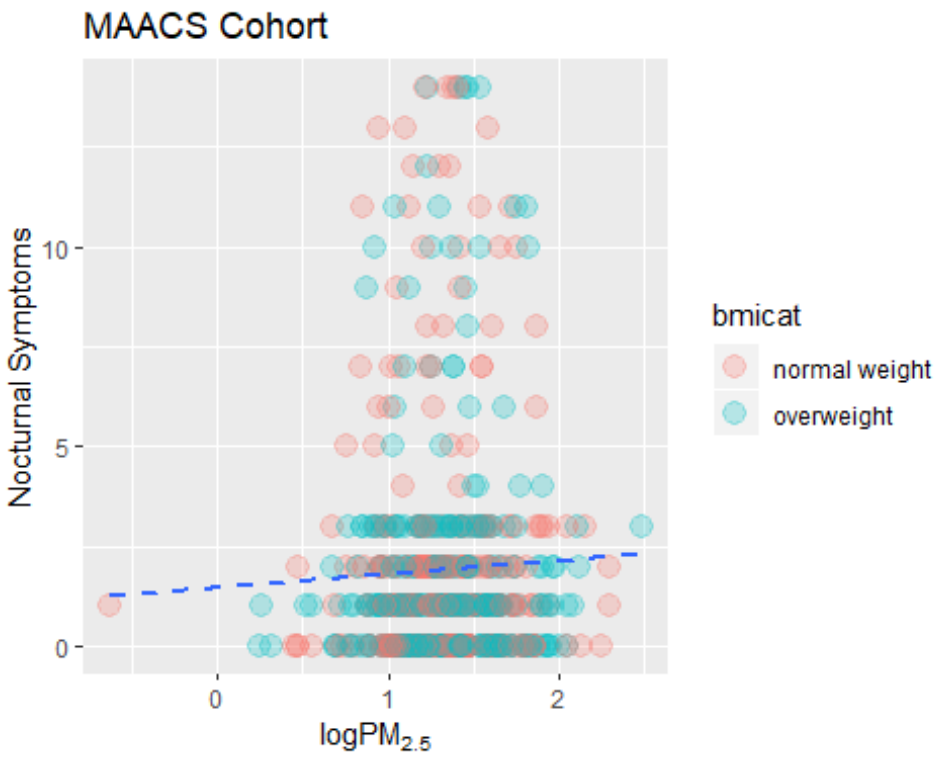

```r
g + geom_point(aes(color = bmicat), size = 4, alpha = 1/4, na.rm = TRUE)
# Cambio de color por categoria
```



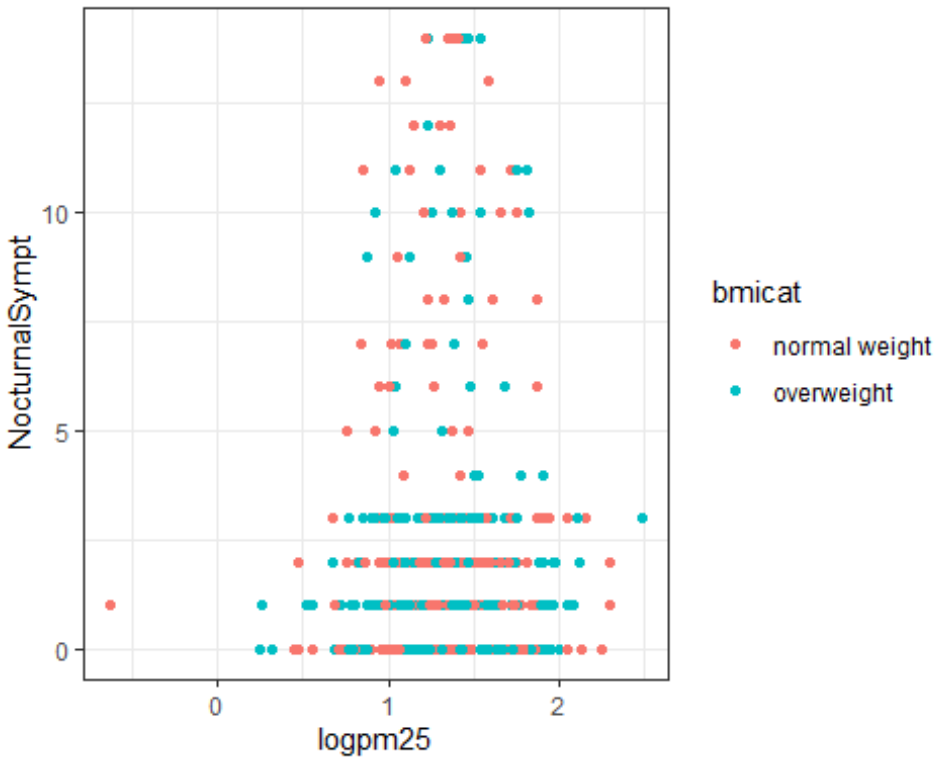
```
g + geom_point(aes(color = bmicat), size = 4, alpha = 1/4, na.rm = TRUE)
+  labs(title = "MAACS Cohort") + labs(x = expression("log"*PM[2.5]), y =
"Nocturnal Symptoms") #Le entrego titulo y nombre a los ejes
```




```
g + geom_point(aes(color = bmicat), size = 4, alpha = 1/4, na.rm = TRUE)
+  labs(title = "MAACS Cohort") + labs(x = expression("log"*PM[2.5]), y =
"Nocturnal Symptoms") + geom_smooth(size = 1, linetype = 2, method =
"lm", se = FALSE, na.rm = TRUE) # Agrego una regresion lineal, (se =
FALSE, desactiva el intervalo de confiaza)
```

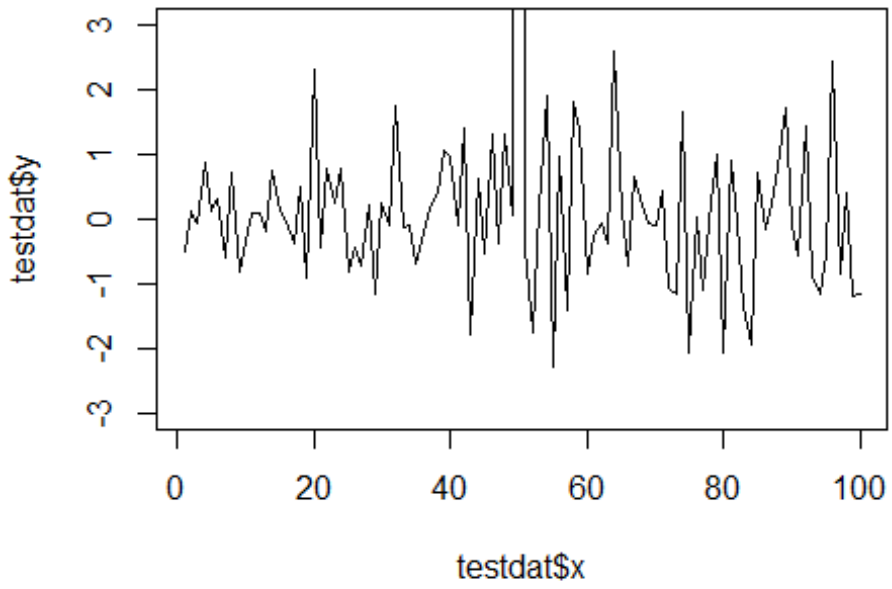



```
g + geom_point(aes(color = bmicat), na.rm = TRUE) + theme_bw(base_family
= "") #Cambio el aspecto del grafico
```

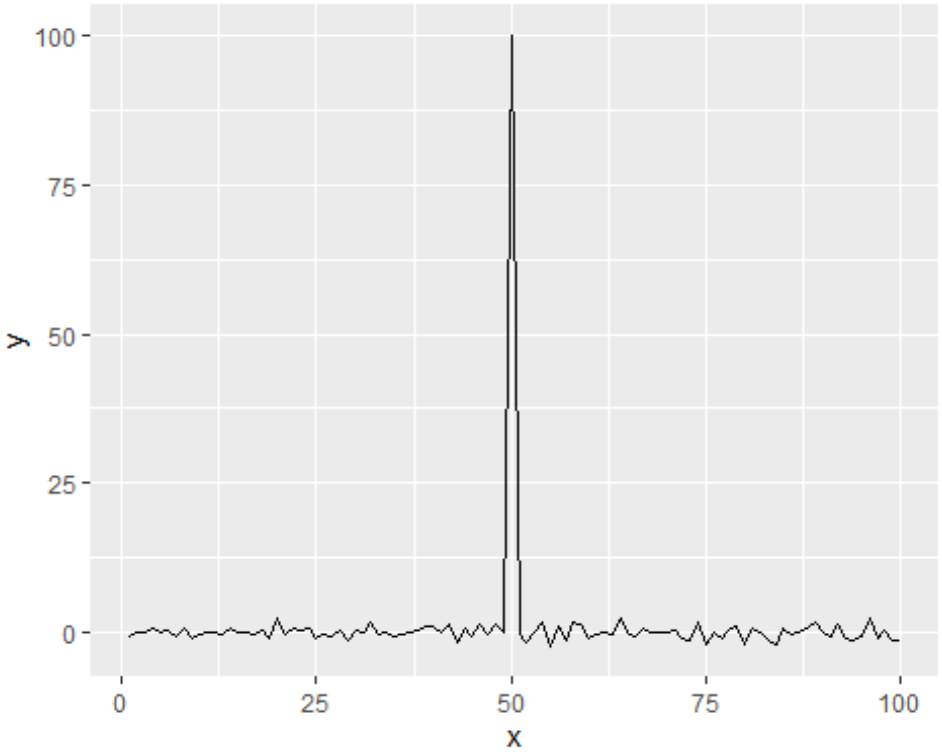


ggplot2 (part 5)

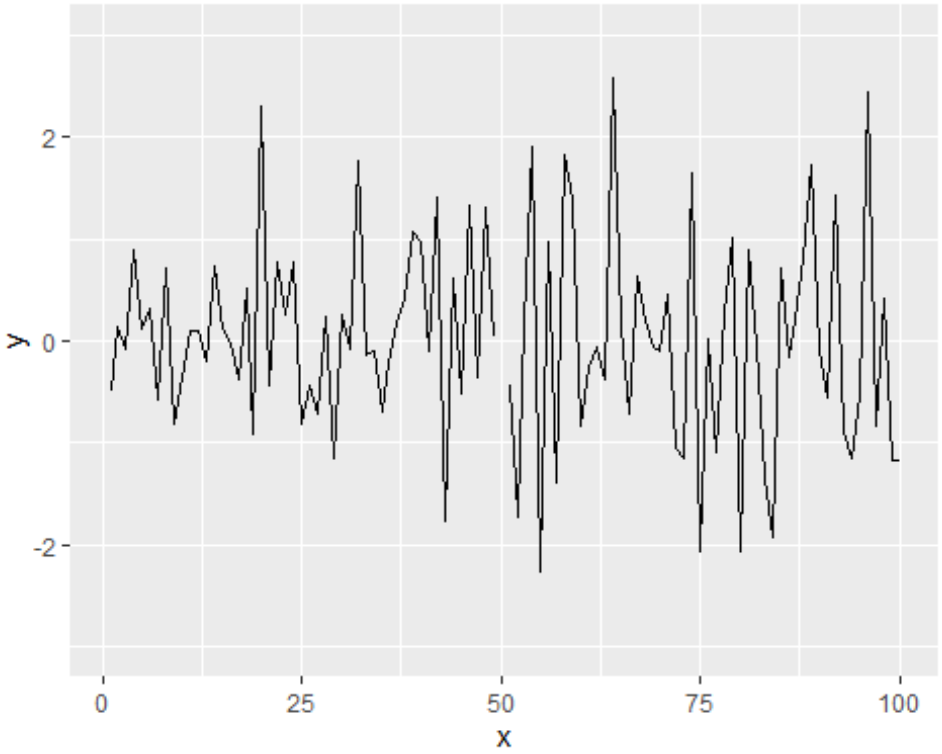
```r
set.seed(100)
testdat <- data.frame(x = 1:100, y = rnorm(100))
testdat[50,2] <- 100
plot(testdat$x,testdat$y, type = "l", ylim = c(-3,3)) #Existe un dato que se escapa de el grafico
```



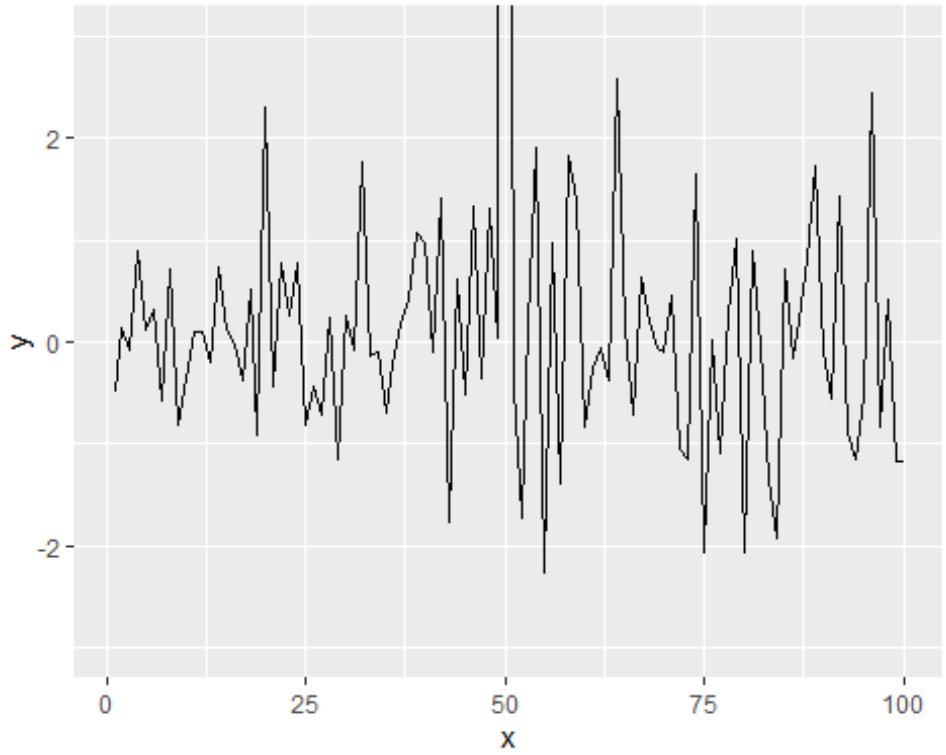

```r
g <- ggplot(testdat, aes(x = x, y = y))
g + geom_line() #El valor extremo hace que todo mi grafico se vea demasiado grande
```




```
g + geom_line() + ylim(-3,3) #solo me muestra los datos que estan entre -3 y 3, oculta el resto
```



```
g + geom_line() + coord_cartesian(ylim = c(-3,3)) #coord_cartesian
permite limitar el plano y no esconder datos
```

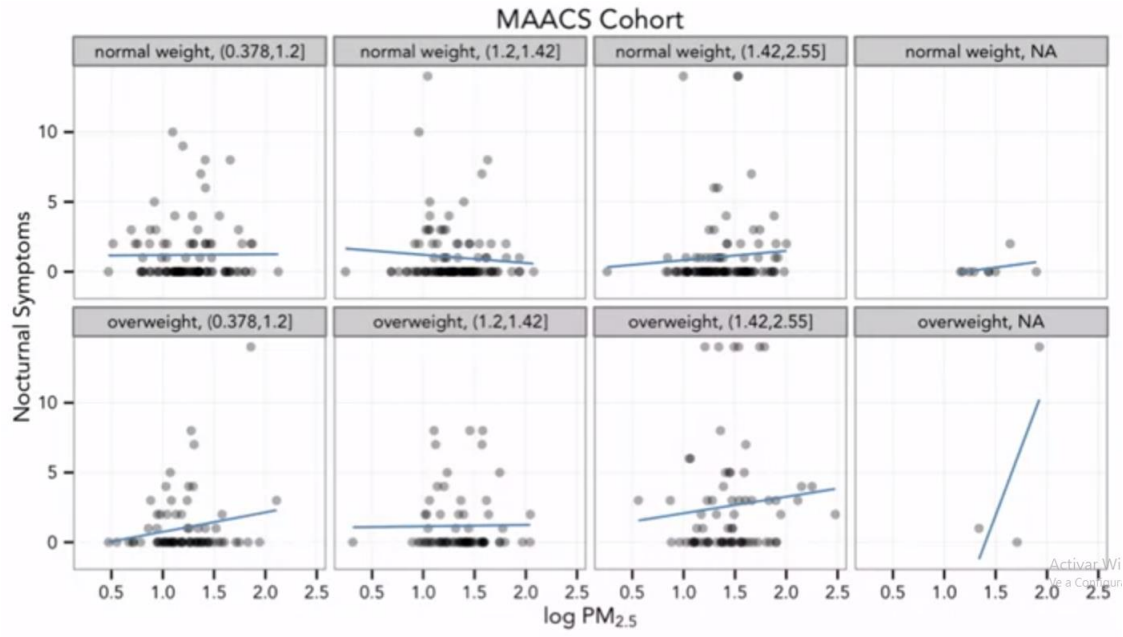



```
cutpoints <- quantile(maacs$logno2_new, seq(0,1,length = 4), na.rm =
TRUE) #Me crea 4 intervalos para una variable continua poder hacerla
categorica
maacs$logno2_new <- cut(maacs$logno2_new, cutpoints) #Corta la variable y
la transforma en factores con los puntos dados
levels(maacs$logno2_new) #Niveles creados

## [1] "(-0.629,1.18]" "(1.18,1.44]"   "(1.44,2.48]"
```

Grafico Final

Matriz de plots

g <- ggplot(maacs, aes(logpm25, NocturnalSymptoms), na.rm = TRUE) g + geom_point(alpha = 1/3) + facet_wrap(bmicat ~ logno2_new, nrow = 2, ncol = 3) + geom_smooth(methods = “lm”, se = FALSE , col = “steelblue”) + theme_bw(base_family = “Avenir”, base_size = 10) + labs(x = expression(“log” * PM[2.5])) + labs (y = “Nocturnal Symptoms”) + labs(title = “MAACS Cohort”)



Week 3

Tomás

6 de diciembre de 2019

Hierarchical Clustering (part 1)

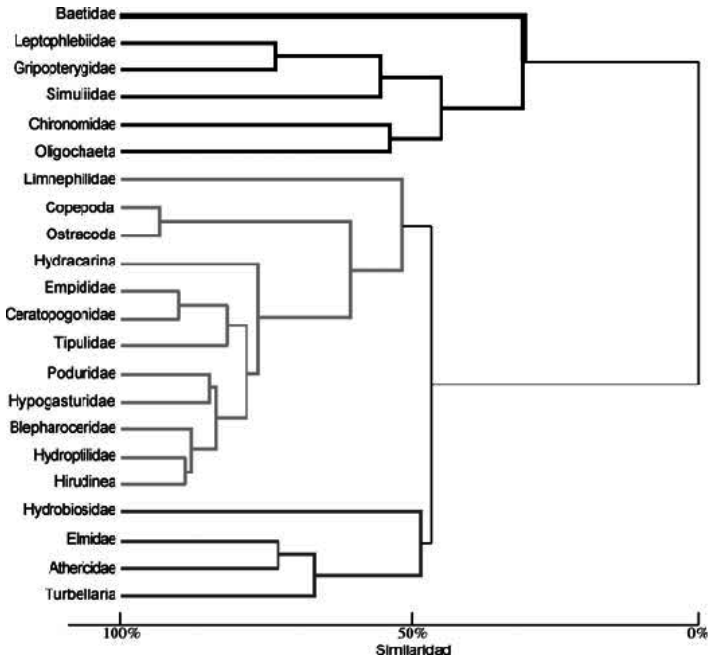
Para realizar metodos de clusters es importante preguntarse lo siguiente:

- Como definimos “cercania”
- Como agrupamos cosas
- Como vizualizamos los grupos
- Como interpretamos estos grupos

Es una tecnica de crear grupos por jerarquizacion, con un approach de aglomeracion, se comienza con un dato y se van agregando al cluster. El metodo de clusters requiere dos cosas importantes:

- La metrica de la distancia
- Una vez que encuentro dos puntos como los uno

El metodo de Hierarcal clustering produce una especie de arbol llamada “dendrogram””.



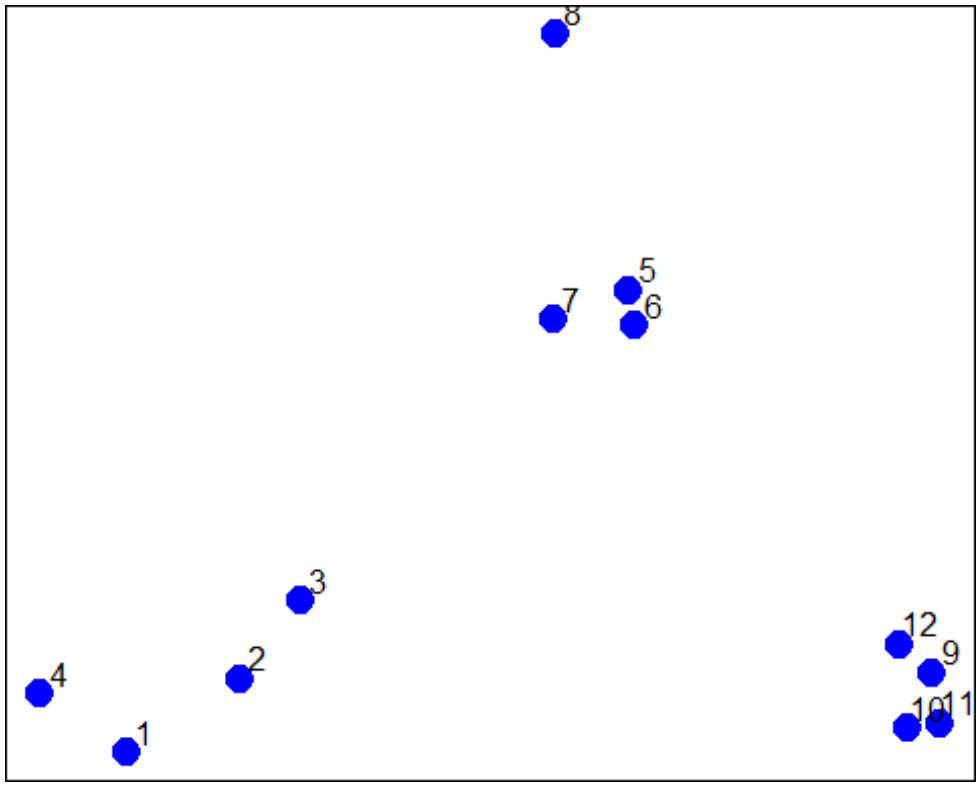
El punto importante de la distancia tiene 3 posibles interpretaciones:

- Continua: Distancia Euclidiana (linea recta)
- Continua: Correlacion similar
- Binaria: Distancia de manhattan (Por los catetos)

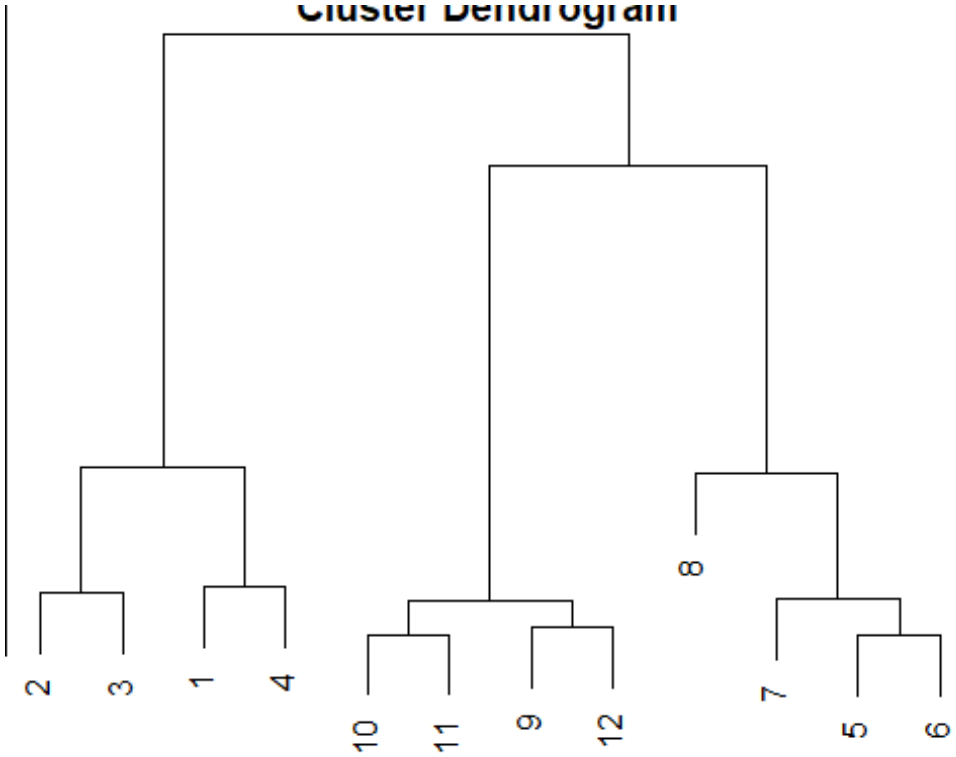
Hay que elegir una metrica que tenga sentido en cada uno de los problemas.

Hierarchical Clustering (part 2)

```r
set.seed(1234)
par(mar = c(0,0,0,0))
x <- rnorm(12, mean = rep(1:3, each = 4), sd = 0.2)
y <- rnorm(12, mean = rep(c(1,2,1), each = 4), sd = 0.2)
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12)) #Creamos 8 puntos
donde se pueden ver 3 clusters facilmente
```



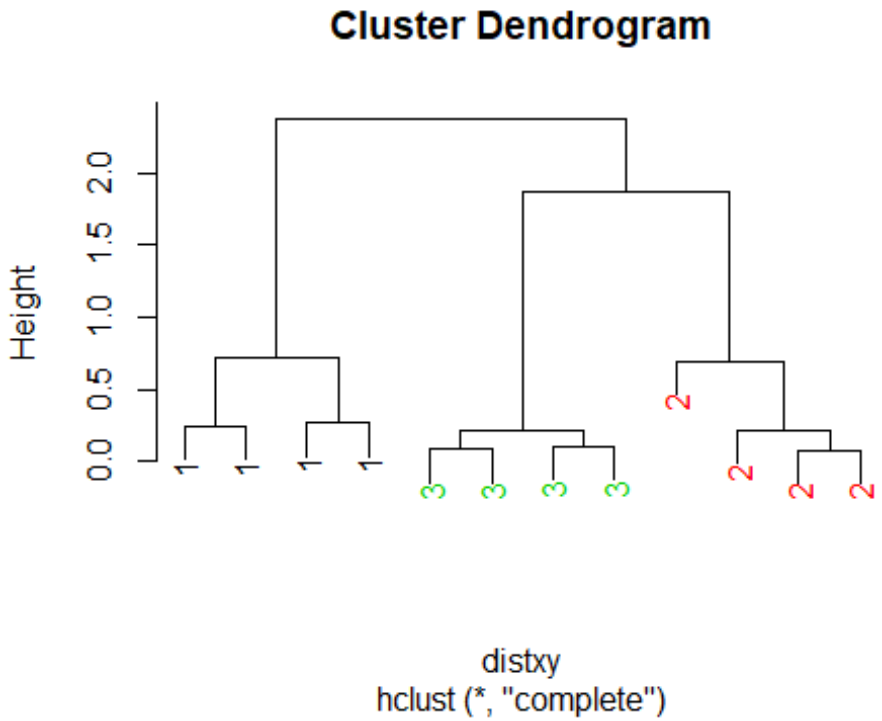

```r
data <- data.frame(x = x, y = y)
distxy <- dist(data) #Sin ningun parametro calcula la distancia euclidiana
hClustering <- hclust(distxy)
plot(hClustering) #El ploteo no me entrega la cantidad de clusters, uno debe cortar en el lugar que le parezca apropiado para elegir la cantidad (2.0 = 2 clusters, 1.0 = 3 clusters)
```



Hierarchical Clustering (part 3)

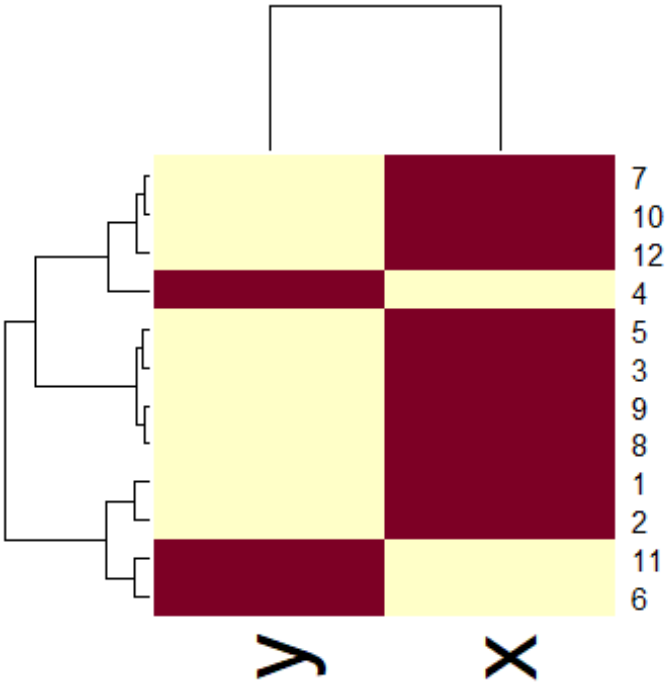
```
myplclust <- function( hclust, lab=hclust$labels,
lab.col=rep(1,length(hclust$labels)), hang=0.1,...){
 ## modifiction of plclust for plotting hclust objects *in colour*!
 ## Copyright Eva KF Chan 2009
 ## Arguments:
 ##    hclust:    hclust object
 ##    lab:        a character vector of labels of the leaves of the tree
 ##    lab.col:    colour for the labels; NA=default device foreground
colour
 ##    hang:     as in hclust & plclust
 ## Side effect:
 ##    A display of hierarchical cluster with coloured leaf labels.
 y <- rep(hclust$height,2)
 x <- as.numeric(hclust$merge)
 y <- y[which(x<0)]
 x <- x[which(x<0)]
 x <- abs(x)
 y <- y[order(x)]
 x <- x[order(x)]
 plot( hclust, labels=FALSE, hang=hang, ... )
 text( x=x, y=y[hclust$order]-(max(hclust$height)*hang),
labels=lab[hclust$order], col=lab.col[hclust$order], srt=90,
adj=c(1,0.5), xpd=NA, ... )}
```

```
myplclust(hClustering, lab = rep(1:3, each = 4), lab.col = rep (1:3, each
= 4)) #Me entrega los clusters por grupo y color
```



Una de las formas de unir dos puntos en uno solo es tomando el promedio de sus distancias como el nuevo punto del cluster, o se puede tomar la distancia de los puntos mas lejanos de cada cluster. Cada una de las dos metricas dan resultados distintos y dependen del tipo de problema.

```
set.seed(153)
dataMatrix <- as.matrix(data)[sample(1:12),]
heatmap(dataMatrix)
```



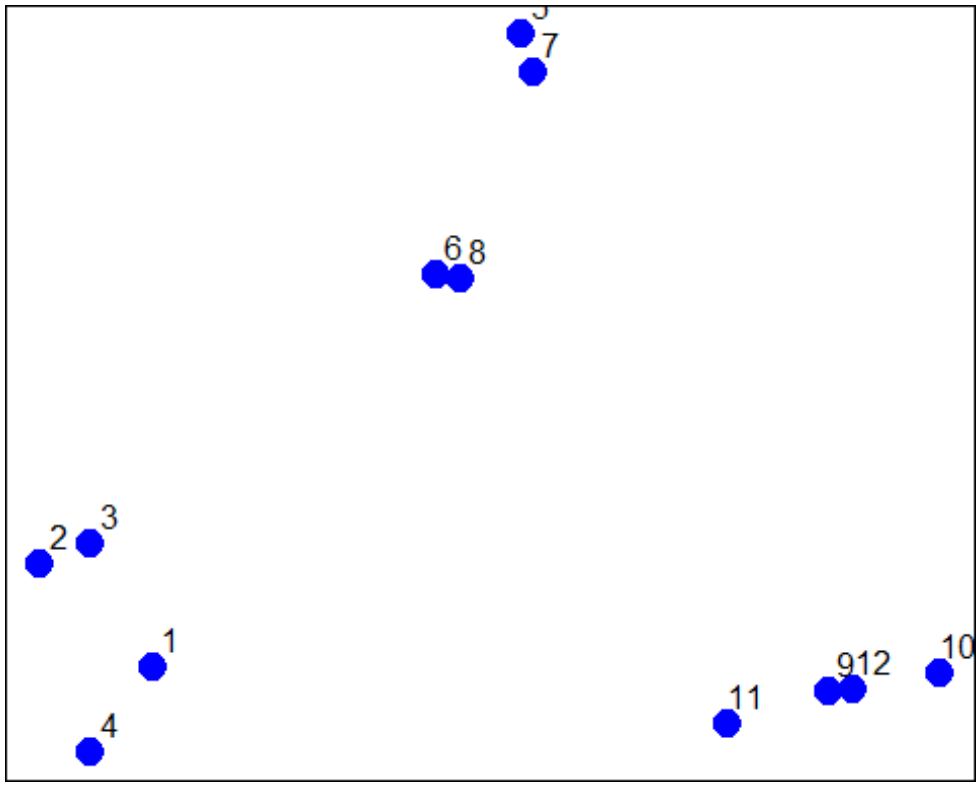
K-Means Clustering (part 1)

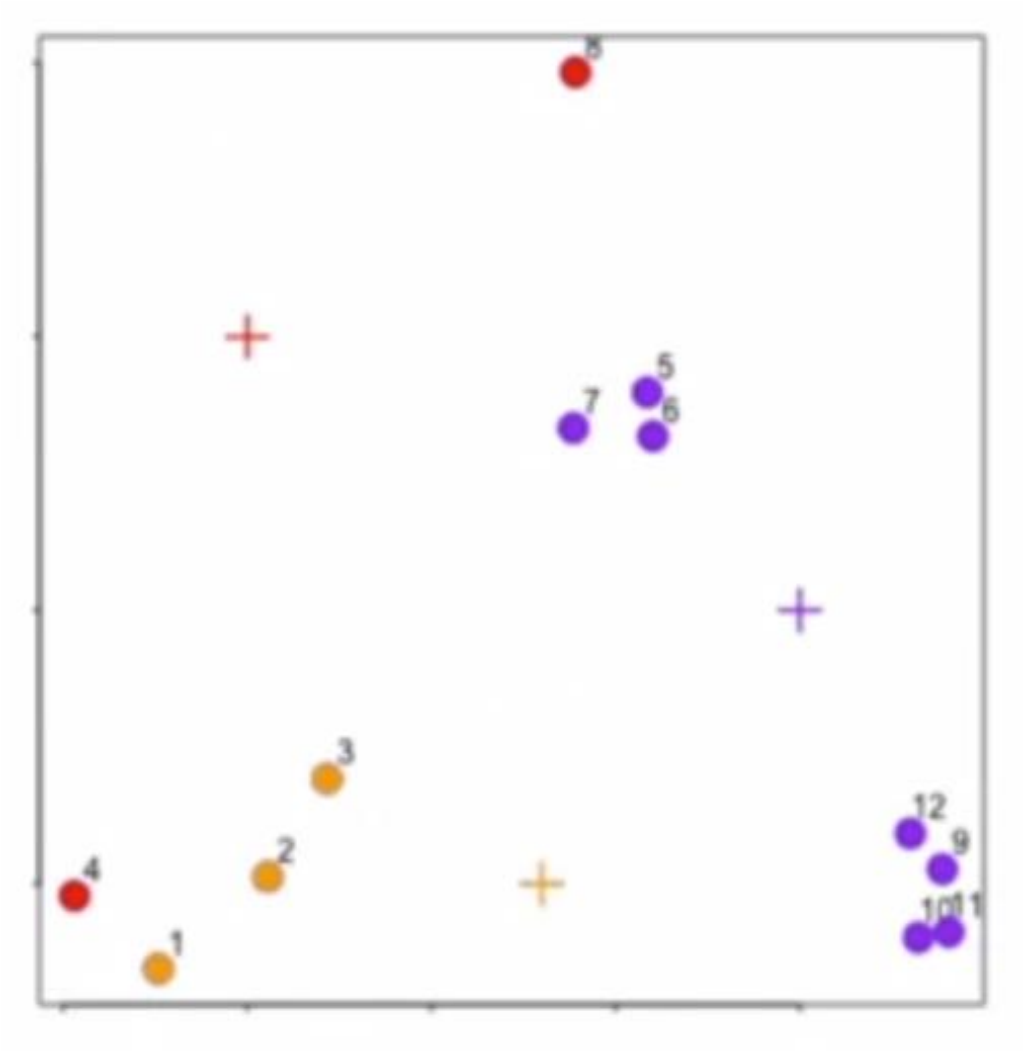
Para realizar k-means cluster se deben decidir los siguientes conceptos:

- Cuantos clusters quiero
- Encontrar el centroide de cada cluster
- Asignar los puntos a cada centroide
- Recalcular los centroides

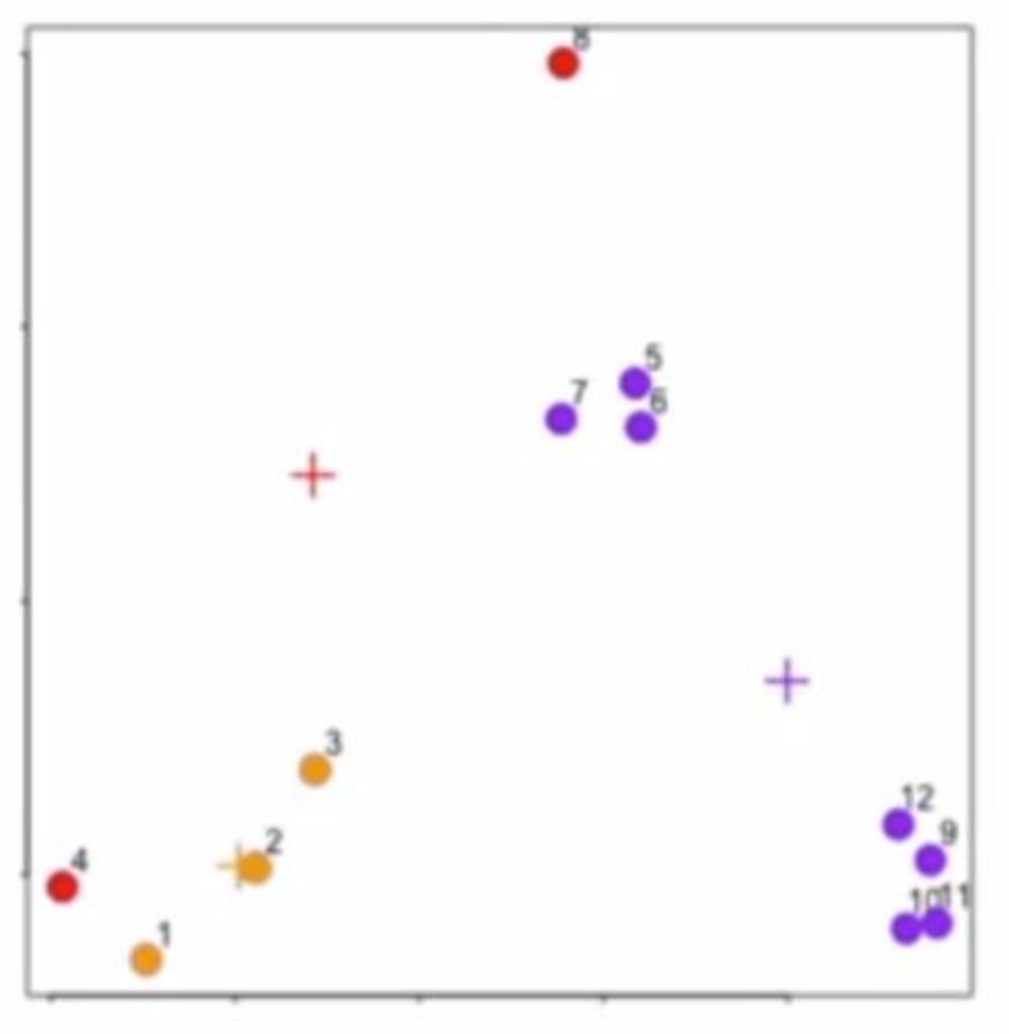
El algoritmo de k-means entrega donde estan los centroides de los clusters y a que cluster pertenece cada punto

```
set.seed(1241212)
par(mar = c(0,0,0,0))
x <- rnorm(12, mean = rep(1:3, each = 4), sd = 0.2)
y <- rnorm(12, mean = rep(c(1,2,1), each = 4), sd = 0.2)
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
```

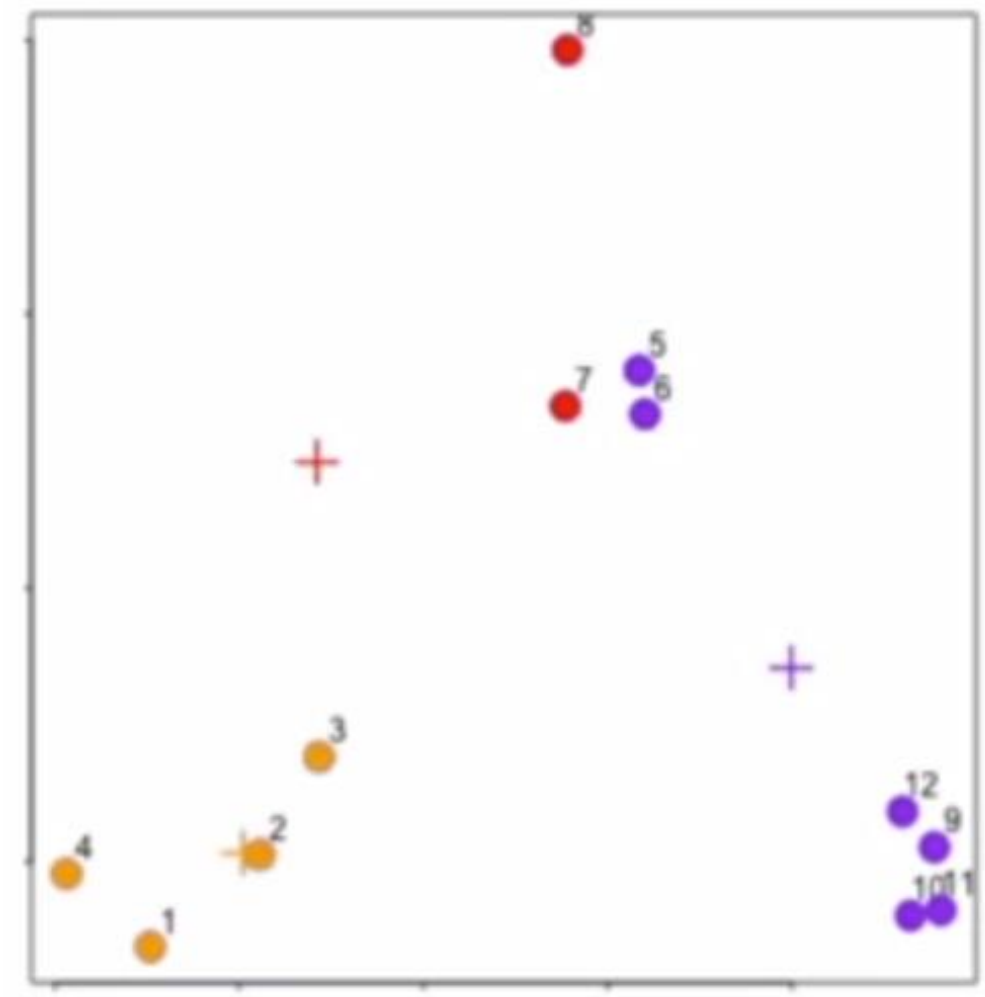

5
7
6
8
2
3
1
4
11
9
12
10



Se inicia con dos centroides y se asignan los puntos al mas cercano



Se calculan los nuevos centroides

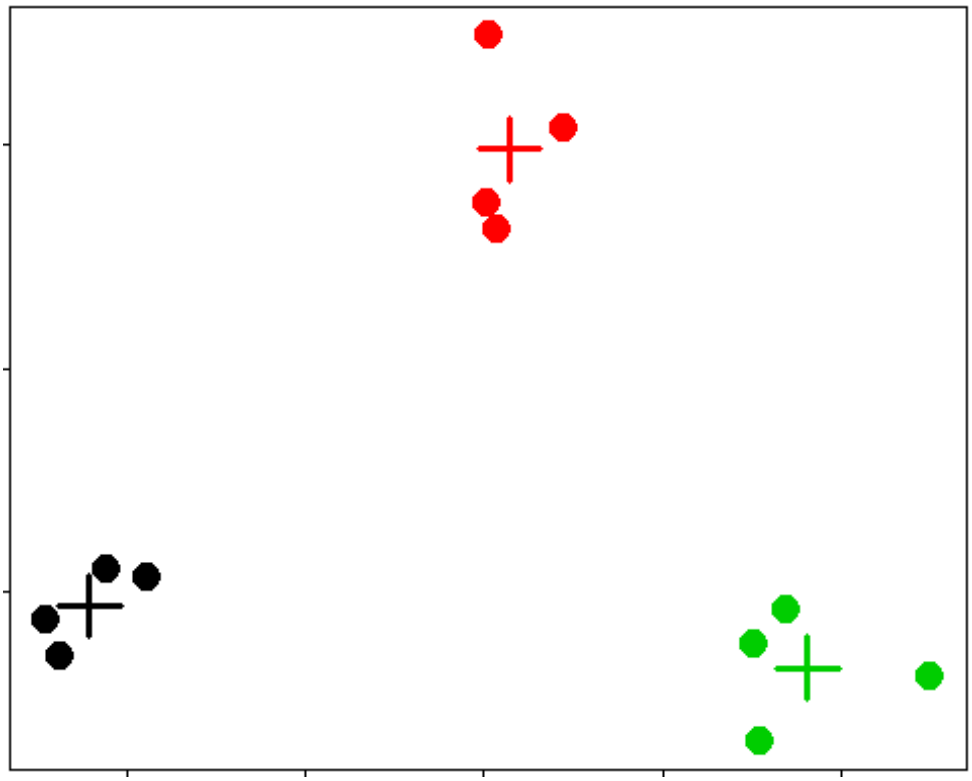


Se recalcula hasta el equilibrio

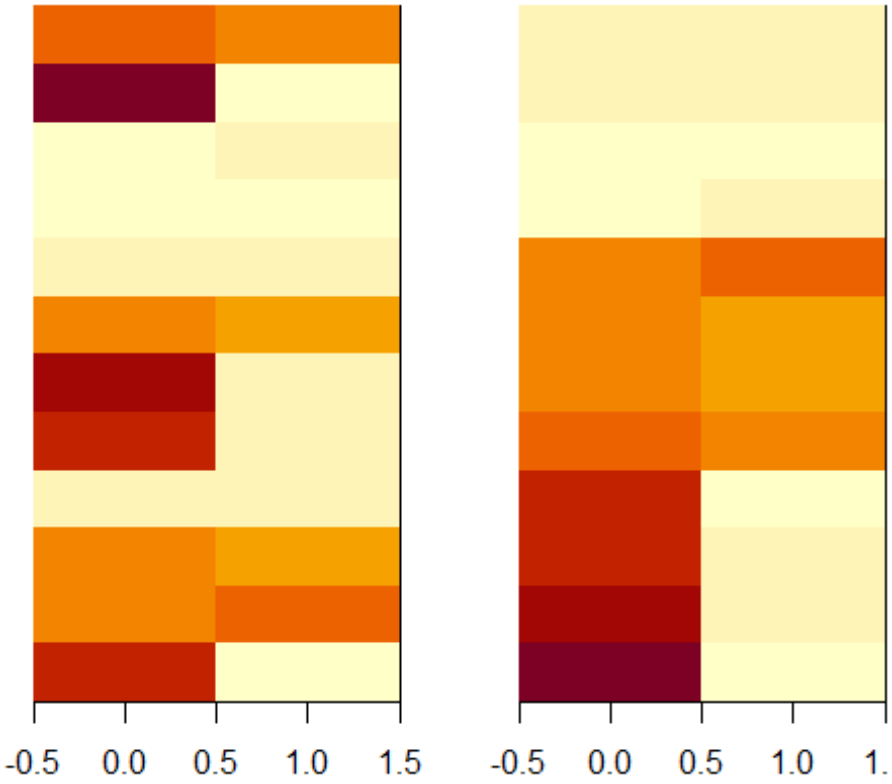
K-Means Clustering (part 2)

```
set.seed(3)
x <- rnorm(12, mean = rep(1:3, each = 4), sd = 0.2)
y <- rnorm(12, mean = rep(c(1,2,1), each = 4), sd = 0.2)
dataFrame <- data.frame(x,y)
kmeansObj <- kmeans(dataFrame, centers = 3) #Centers indica la cantidad
de centroides
names(kmeansObj)

## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

kmeansObj$cluster #Nos indica en que cluster esta cada uno de los puntos

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

```r
par(mar = rep(0.2,4))
plot(x,y,col = kmeansObj$cluster, pch = 19, cex = 2) ;
points(kmeansObj$centers, col = 1:3, pch = 3, cex = 3, lwd = 3)  #Ploteo los puntos y agrego sus centroides
```

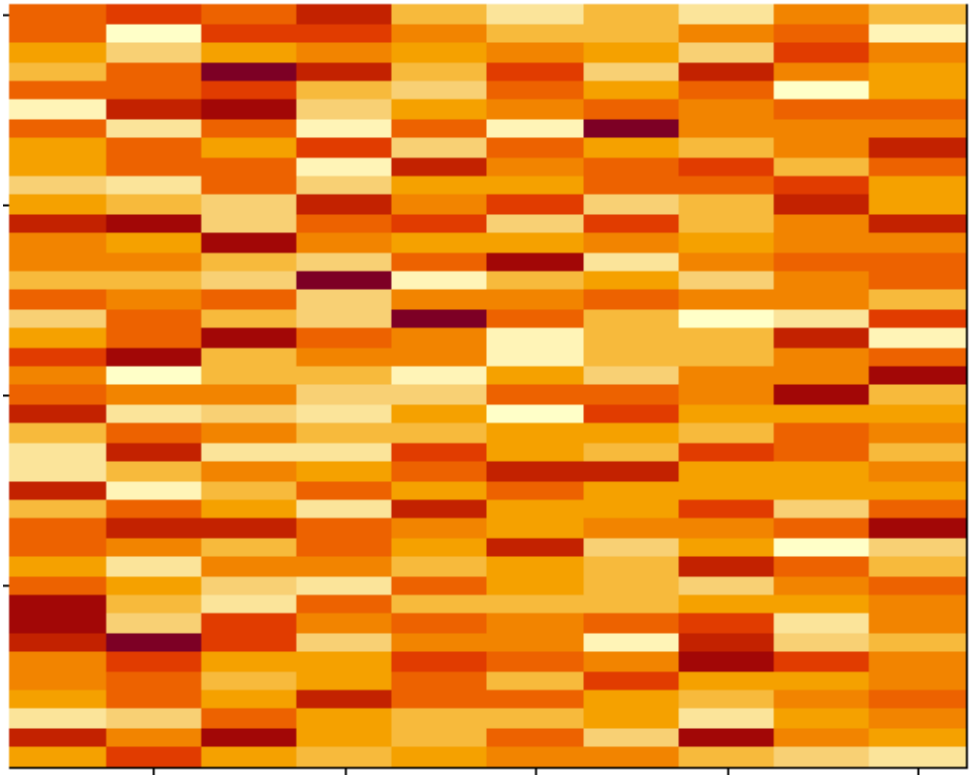


```r
dataMatrix <- as.matrix(dataFrame)[sample(1:12), ]
kmeansObj2 <- kmeans(dataMatrix, centers = 3)
par(mfrow = c(1,2), mar = c(2,3,0.1,0.1))
image(t(dataMatrix)[, nrow(dataMatrix):1], yaxt = "n")
image(t(dataMatrix)[, order(kmeansObj2$cluster)], yaxt = "n") #Mapa de calor
```

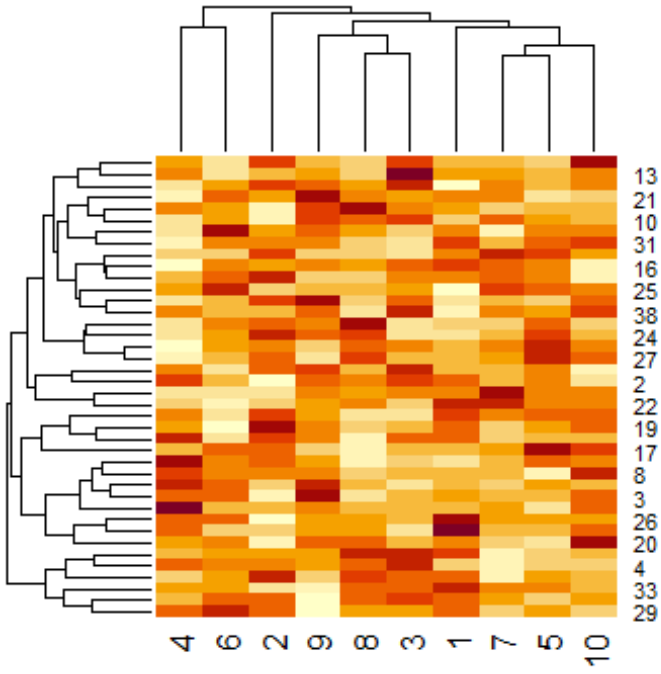


Dimension Reduction (part 1)

```
set.seed(12345)
par(mar = rep(0.2,4))
dataMatrix <- matrix(rnorm(400), nrow = 40)
image(1:10,1:40,t(dataMatrix)[, nrow(dataMatrix):1])
```

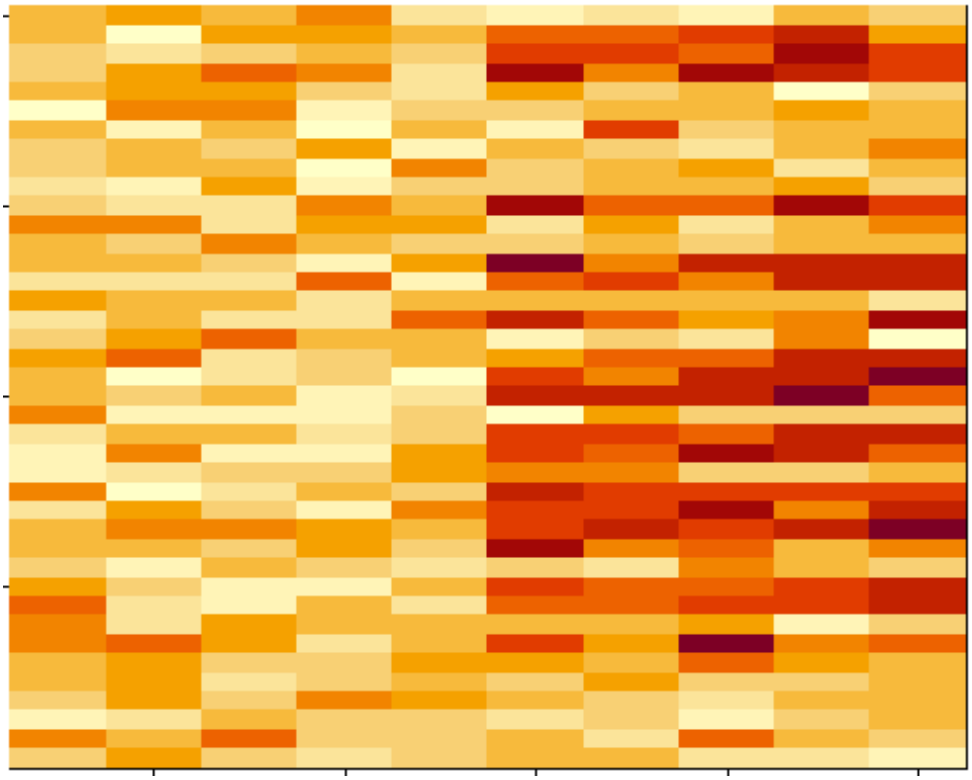


```r
heatmap(dataMatrix) #Mapa de calor
```

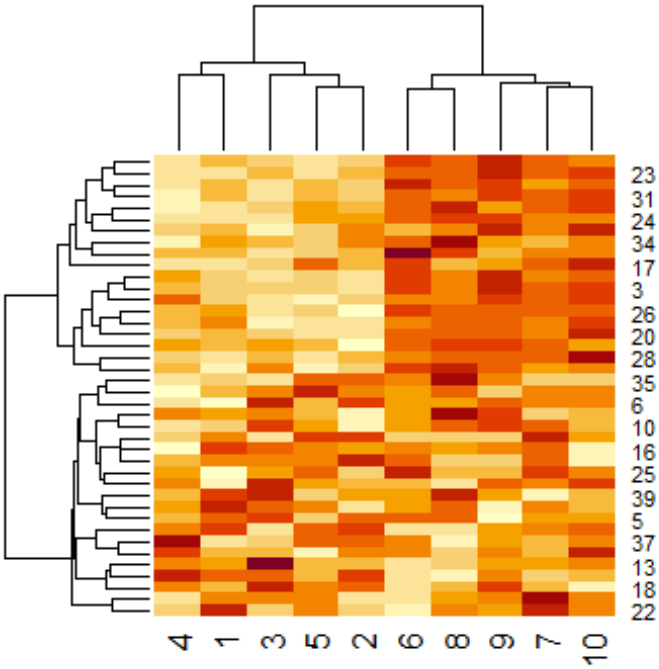



```r
set.seed(54754)
for (i in 1:40){
```

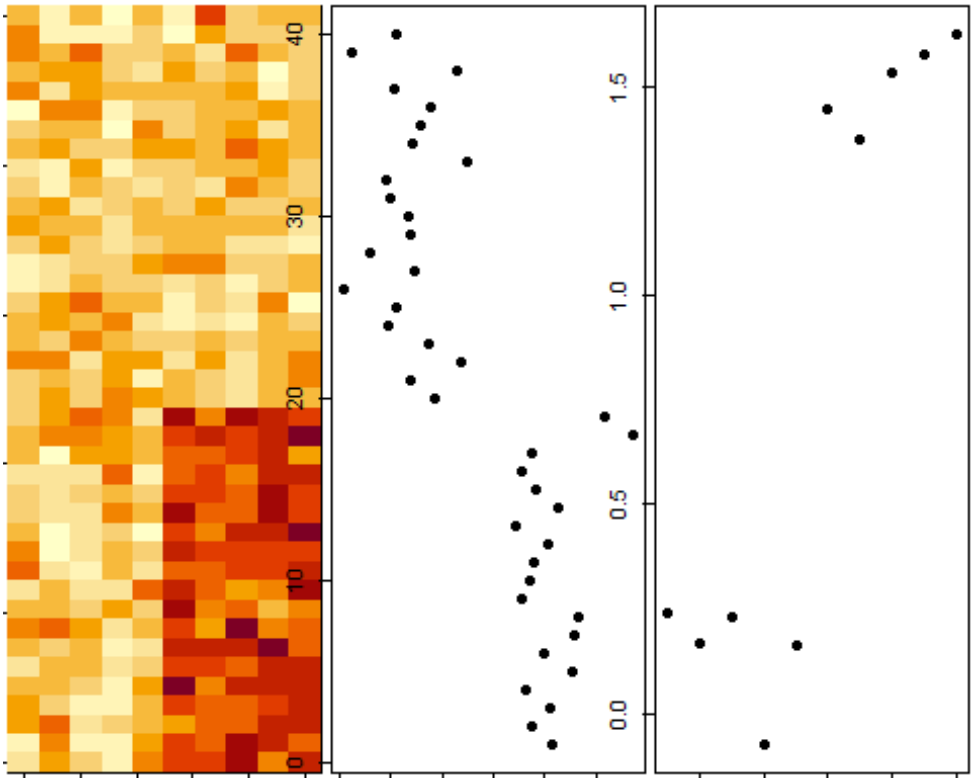
```
        #flip a coin
        coinFlip <- rbinom(1, size =1, prob = 0.5)
        # if coin is heads add a common pattern to that row
        if (coinFlip){
                dataMatrix[i, ] <- dataMatrix[i, ] + rep(c(0,3), each =
5)
        }
} #Creo un àtron para entender mejor el mapa de calor
image(1:10,1:40,t(dataMatrix)[, nrow(dataMatrix):1])
```



```
heatmap(dataMatrix)
```




```r
hh <- hclust(dist(dataMatrix))
dataMatrixOrdered <- dataMatrix[hh$order, ]
par(mfrow = c(1,3))
image(t(dataMatrixOrdered)[, nrow(dataMatrixOrdered):1])
plot(rowMeans(dataMatrixOrdered), 40:1, xlab = "Row mean", ylab = "Row",
pch = 19)
plot(colMeans(dataMatrixOrdered), xlab = "Columns", ylab = "Column Mean",
pch = 19) #Mapa de calor y como interpretar este
```



Para problemas con muchas variables que probablemente esten correlacionadas es importante intentar realizar estos dos pasos:

- Encontrar un nuevo set de variables que no este correlacionadas y que expliquen bien los datos
- Ponerlas todas en una matrix de menor rango

Para realizar lo anterior existen dos tecnicas:

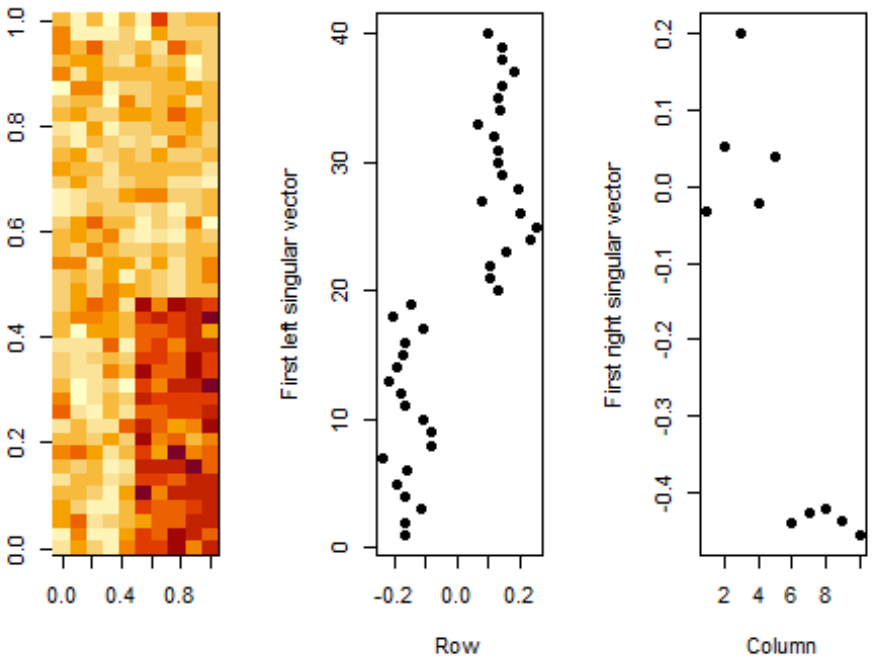
- SVD (Singular Value Decomposition):

```
    + Crea 3 matrices de la forma UDV^T^, donde U es es una matriz
    ortogonal (Matriz izquierda singular), V matriz ortogonal (Matrisz
    derecha singular) y D es la matriz de valores singulares (*Una matriz
    ortogonal es la que su matriz traspuesta es igual a la original)
    + PCA (Principal Component Analisis): Esta tecnica dice que si a cada
    columna le resto el promedio de esta y la divido por su desviacion
    estandar puedo encontrar la matrix V^T^
```

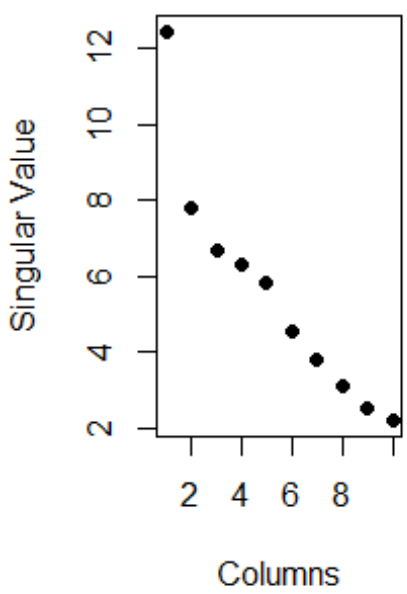
Dimension Reduction (part 2)

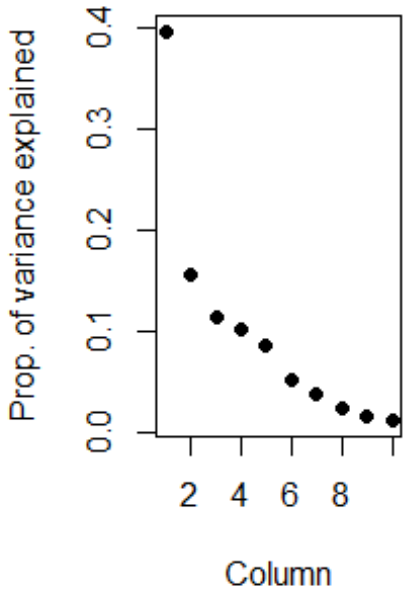
```
svd1 <- svd(scale(dataMatrixOrdered)) #Singular value decomposition
par(mfrow = c(1,3))
image(t(dataMatrixOrdered)[, nrow(dataMatrixOrdered):1])
plot(svd1$u[, 1], 40:1, xlab = "Row", ylab = "First left singular
vector", pch = 19) #La mtriz U
```

```r
plot(svd1$v[, 1], xlab = "Column", ylab = "First right singular vector",
pch = 19) #La mtriz V
```

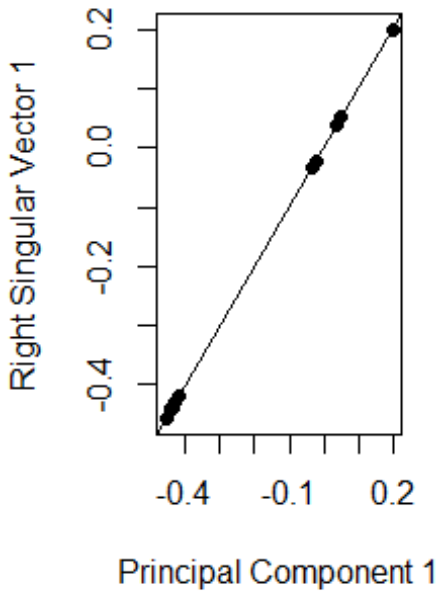



```r
par(mfrow = c(1,2)) #Variance explain
plot(svd1$d, xlab = "Columns", ylab = "Singular Value", pch = 19) #Matriz
D
plot(svd1$d^2/sum(svd1$d^2), xlab = "Column", ylab = "Prop. of variance
explained", pch = 19)# La proporcion de como explica la variacion de la
data
```

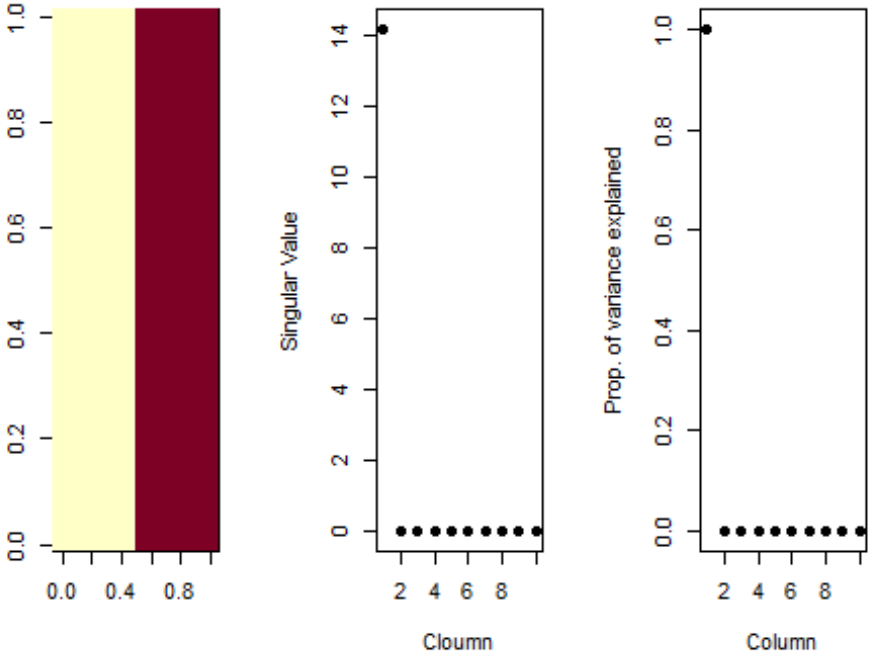




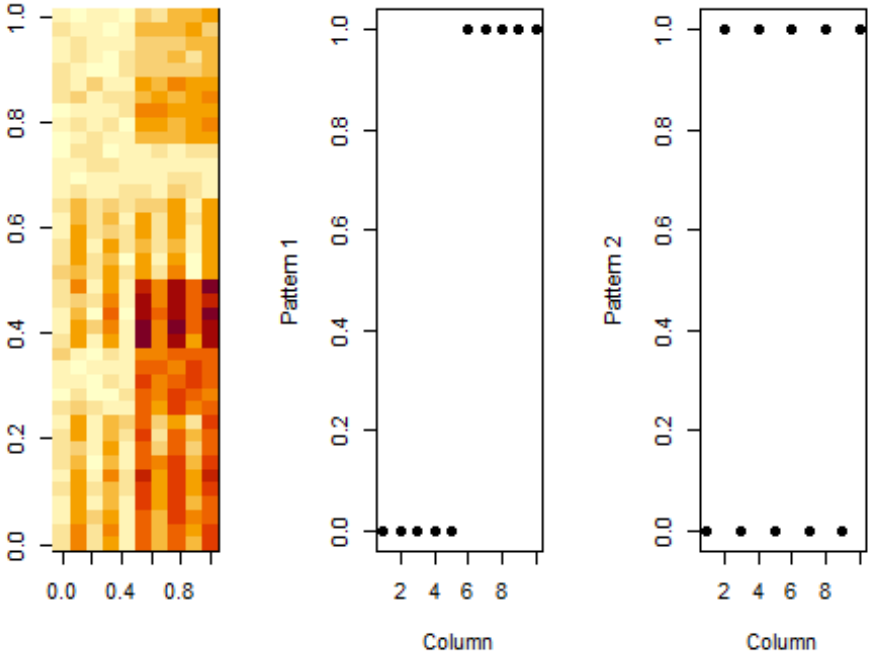

```r
svd1 <- svd(scale(dataMatrixOrdered))
pca1 <- prcomp(dataMatrixOrdered, scale = TRUE)
plot(pca1$rotation[, 1], svd1$v[, 1], pch = 19, xlab = "Principal
Component 1", ylab = "Right Singular Vector 1")# Se plotea el primer
vector de la matriz v de la svd, y el primer vector del pca y se nota que
son exactamente iguales como se habia propuesto
abline(c(0,1))

constantMatrix <- dataMatrixOrdered*0 #Variance explained
for(i in 1:dim(dataMatrixOrdered)[1]){constantMatrix[i,] <- rep(c(0,1),
each = 5)}
svd1 <- svd(constantMatrix)
par(mfrow = c(1,3))
```



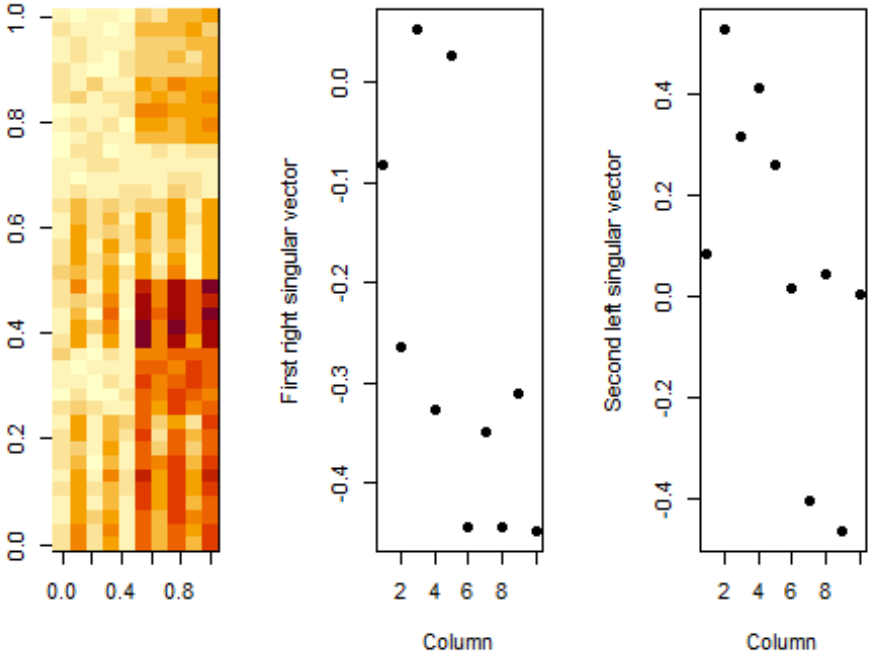

```r
image(t(constantMatrix)[, nrow(constantMatrix):1])
plot(svd1$d, xlab = "Cloumn", ylab ="Singular Value", pch = 19)
plot(svd1$d^2/sum(svd1$d^2), xlab = "Column", ylab = "Prop. of variance explained", pch = 19)#Esto explica que si estas en las primeras 5 columnas tienes un valor 0 y si estas en las siguientes 5 tienes un valor de 5. Estoy nos ayuda a entender realciones o que algunas columnas o filas no aportan mucho a los datos
```



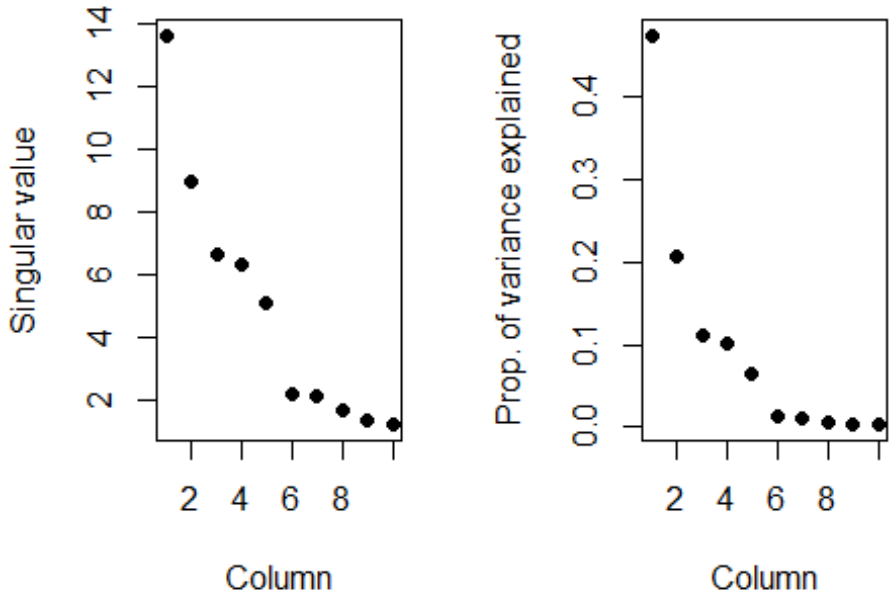

```
set.seed((31252352))
for(i in 1:40){
        coinFlip1 <- rbinom(1, size = 1, prob = 0.5)
        coinFlip2 <- rbinom(1, size = 1 , prob = 0.5)
        if(coinFlip1){
                dataMatrix[i, ] <- dataMatrix[i, ] + rep(c(0,5), each =
5)
        }
        if(coinFlip2){
                dataMatrix[i, ] <- dataMatrix[i, ] + rep(c(0,5), 5)
        }
}
hh <- hclust(dist(dataMatrix))
dataMatrixOrdered <- dataMatrix[hh$order, ]
svd2 <- svd(scale(dataMatrixOrdered))
par(mfrow = c(1,3))
image(t(dataMatrixOrdered)[, nrow(dataMatrixOrdered):1])
plot(rep(c(0,1), each = 5), pch = 19, xlab = "Column", ylab = "Pattern
1")
plot(rep(c(0,1), 5), pch = 19, xlab = "Column", ylab = "Pattern 2")
```




```
svd2 <- svd(scale(dataMatrixOrdered))
par(mfrow = c(1,3))
image(t(dataMatrixOrdered)[, nrow(dataMatrixOrdered):1])
plot(svd2$v[, 1], xlab = "Column", ylab = "First right singular vector",
pch = 19) #La mtriz V
plot(svd2$v[, 2], xlab = "Column", ylab = "Second left singular vector",
pch = 19) #La mtriz U
```

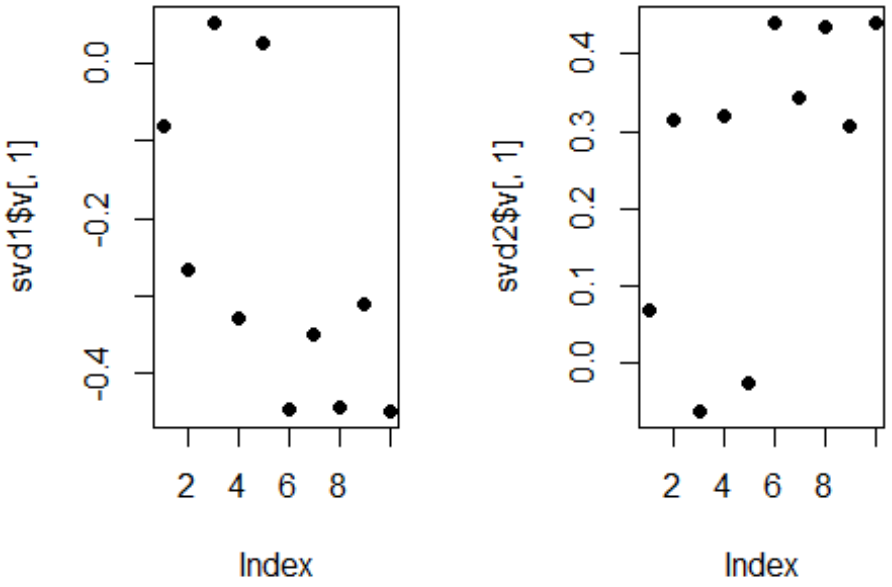



```
svd1 <-svd(scale(dataMatrixOrdered))
par(mfrow = c(1,2))
plot(svd1$d, xlab = "Column", ylab = "Singular value", pch = 19)
plot(svd1$d^2/sum(svd1$d^2), xlab = "Column", ylab = "Prop. of variance
explained", pch = 19) #La primera columna explica mas del 50% de la
variacion total de la data
```



Dimension Reduction (part 3)

```
dataMatrix2 <- dataMatrixOrdered
dataMatrix2[sample(1:100, size = 40, replace = FALSE)] <- NA
#svd1 <- svd(scale(dataMatrix2)) #No funciona cuando existen NAs
library(impute)

## Warning: package 'impute' was built under R version 3.6.1

dataMatrix2 <- dataMatrixOrdered
dataMatrix2[sample(1:100, size = 40, replace = FALSE)] <- NA
dataMatrix2 <- impute.knn(dataMatrix2)$data #toma los k valores mas
cercanos a el (fila) y convierte el NA en la media de esos K valores
svd1 <- svd(scale(dataMatrixOrdered)) ; svd2 <- svd(scale(dataMatrix2))
par(mfrow = c(1,2)); plot(svd1$v[,1],pch = 19); plot(svd2$v[,1], pch =
19) #Son parecidos pero casi iguales
```



Ejemplo de cara y como reducir dimensiones Semana 3 lesson 2 ultimo video

Working with Color in R Plots (part 2)

El paquete "grDevices" tiene dos funciones principales:

- colorRamp: Toma una paleta de colores y returna una funcion que contiene numeros entre 1 y 0
- colorRampPalette: Esta entrega un entero a traves de un vector de colores (se parece a heat o topo color)

Estas dos funciones te permiten interpolar entre colores, la funcion colors() me permite ver todos los colores disponibles

```
library(grDevices)
pal <- colorRamp(c("red","blue")) #Devuelve un vector de largo 3 (Rojo, verde, azul)
pal(0)# (rojo 255 y el resto 0)

##      [,1] [,2] [,3]
## [1,]  255    0    0

pal(1)# azul 255 0 el resto
```

```
##      [,1] [,2] [,3]
## [1,]    0    0  255
```

```
pal(0.5)#(127,5 de rojo y 127,5 de azul)
```

```
##       [,1] [,2]  [,3]
## [1,] 127.5    0 127.5
```

```
pal(seq(0,1,len=10))#Me entrega una secuencia de colores entre rojo y
azul
```

```
##             [,1] [,2]      [,3]
##  [1,] 255.00000    0   0.00000
##  [2,] 226.66667    0  28.33333
##  [3,] 198.33333    0  56.66667
##  [4,] 170.00000    0  85.00000
##  [5,] 141.66667    0 113.33333
##  [6,] 113.33333    0 141.66667
##  [7,]  85.00000    0 170.00000
##  [8,]  56.66667    0 198.33333
##  [9,]  28.33333    0 226.66667
## [10,]   0.00000    0 255.00000
```

```
pal <- colorRampPalette(c("red", "yellow")) #Ramp
pal(2)#Me entrega dos colores interpolando los dados (En este caso me
entrega los originales
```

```
## [1] "#FF0000" "#FFFF00"
```

```
pal(10)#Me entrega un vector de caracteres en hexadecimal(Los primeros
dos numero son rojo, los siguientes verdes y los siguientes azules). La F
representa el maximo numero en hexadecimal.
```

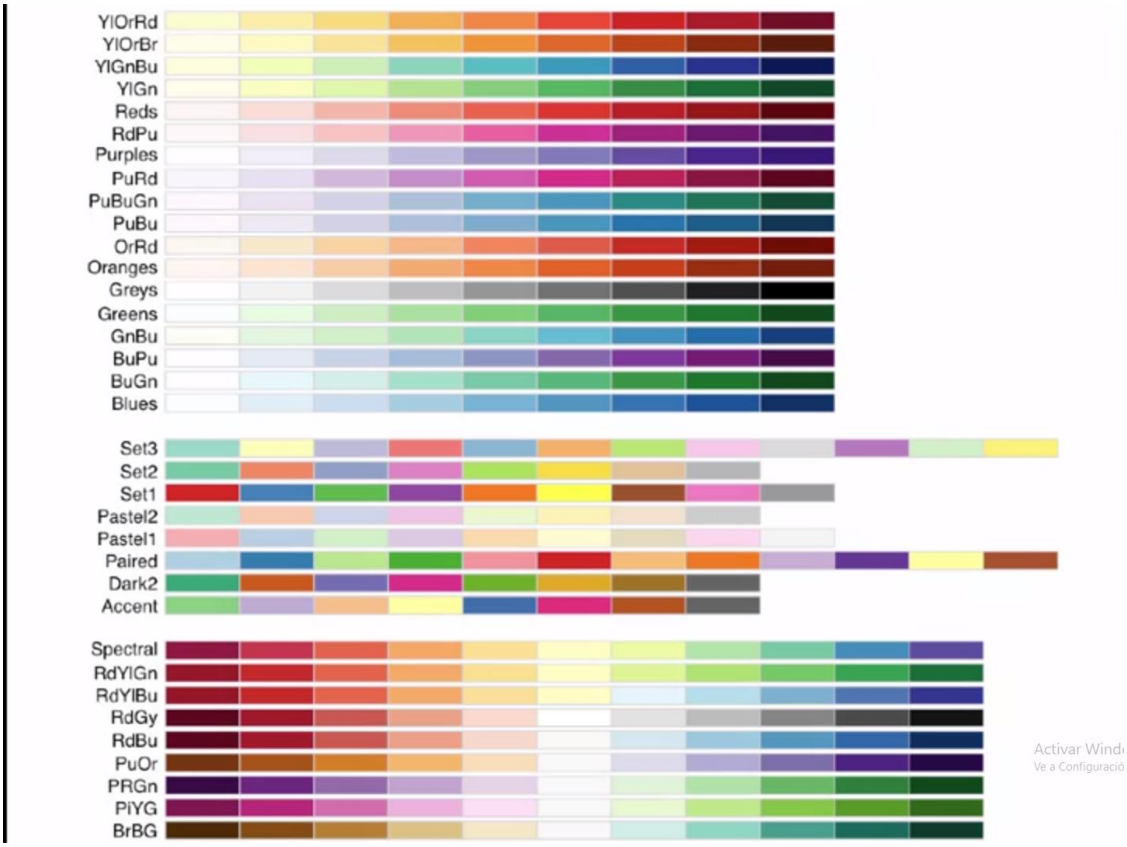
```
##  [1] "#FF0000" "#FF1C00" "#FF3800" "#FF5500" "#FF7100" "#FF8D00"
"#FFAA00"
##  [8] "#FFC600" "#FFE200" "#FFFF00"
```

Working with Color in R Plots (part 3)

Otra opcion es RColorBrewer, que fue creado para hacer mapploting. Existen 3 tipos de paletas:

- Secuenciales: para datos que estan de menor a mayor
- Divergentes: La desviacion de la media por ejemplo
- Cualitativo: Representar data que no este ordenado, factores, variables categoricas

Estas paletas se pueden pasar a las funciones colorRamp y colorRampPalette para usarlas

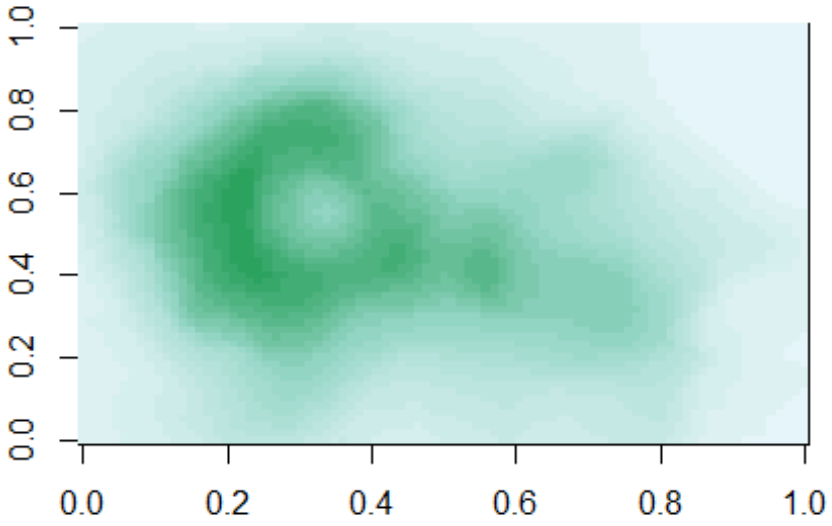


Secuencial, divergente, cualitativo

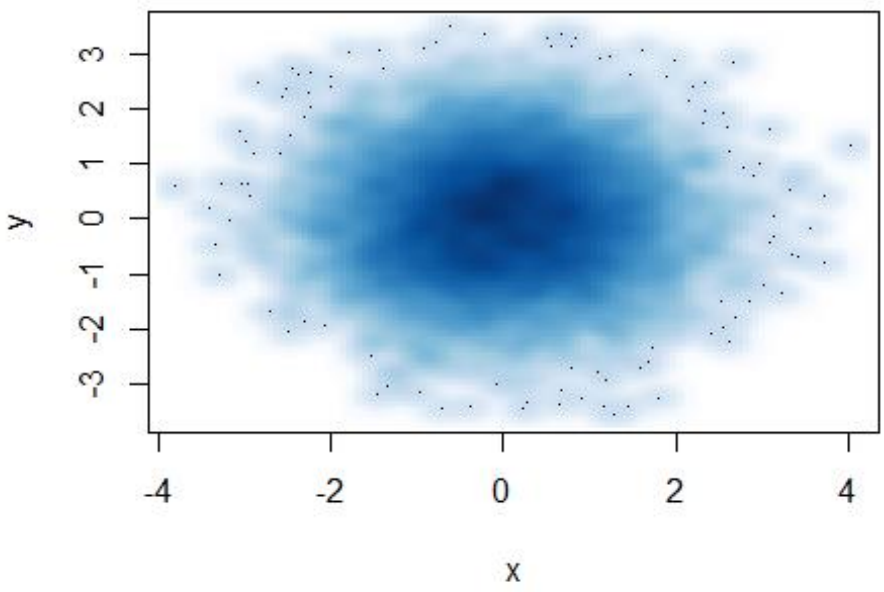
```
library(RColorBrewer)
cols <- brewer.pal(3, "BuGn") #Entrega una paleta de colores
cols
```

```
## [1] "#E5F5F9" "#99D8C9" "#2CA25F"
```

```
pal <- colorRampPalette(cols)
image(volcano, col = pal(20))
```



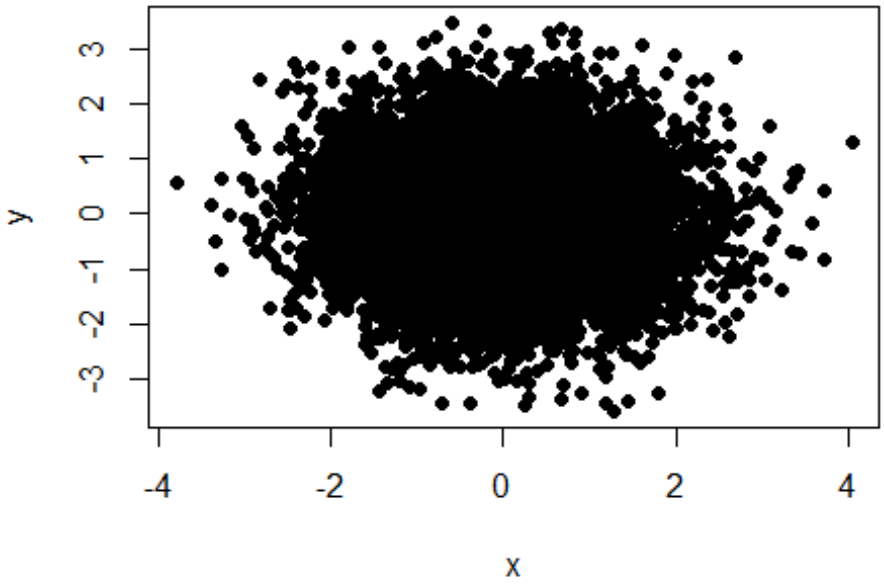

```r
x <- rnorm(10000)
y <- rnorm(10000)
smoothScatter(x, y) #Esta funcion permite plotear muchos puntos cuando no
te interesan los puntos en si, si no que donde estan agrupados
```



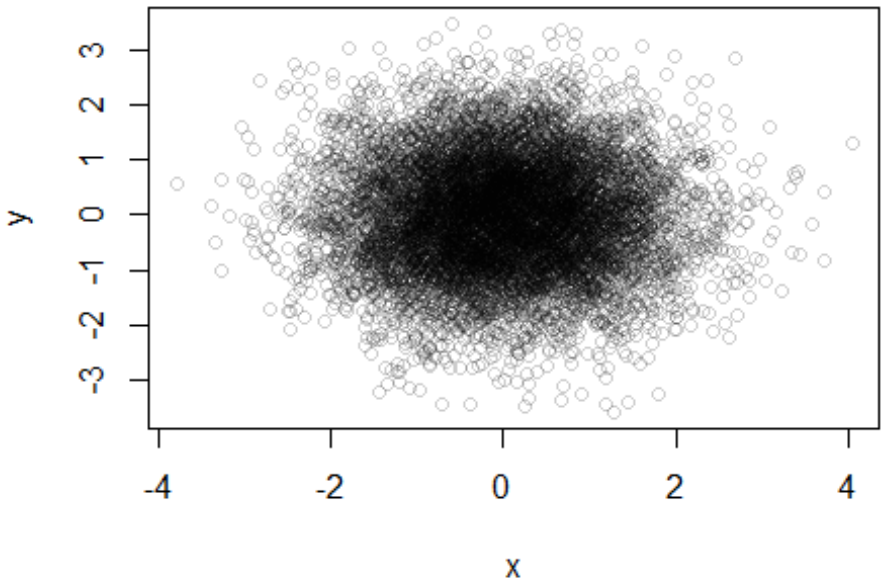
Working with Color in R Plots (part 4)

Funcion rgb, permite producir cualquier color a partir de rojo, verde y azul. Tambien permite la transparencia de colores con alpha. Colorspace tambien permite jugar con los colores.

```
plot(x,y, pch = 19)
```




```r
plot(x,y,col = rgb(0,0,0,0.2))  #Le entrego la transparencia a los numeros (Rojo,Verde,Azul,Alpha)
```



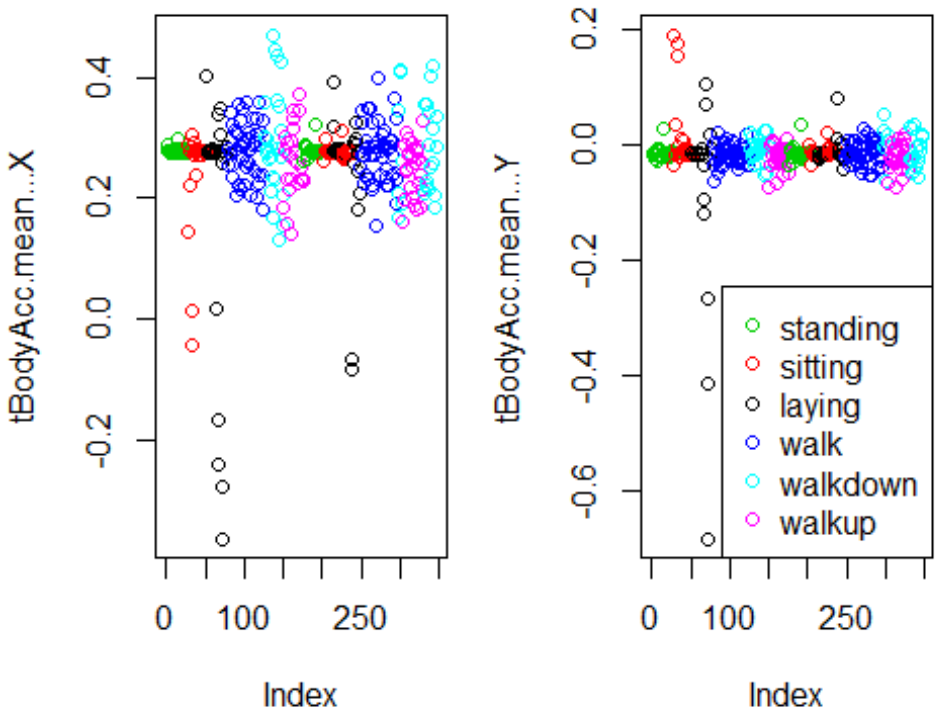
Week 4

Tomás

10 de diciembre de 2019

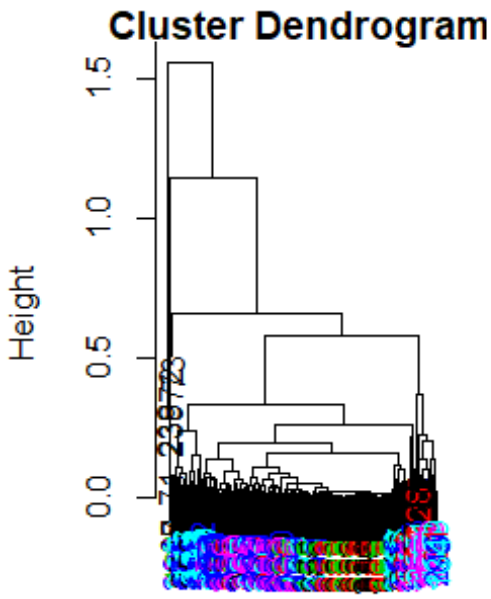
Clustering Case Study

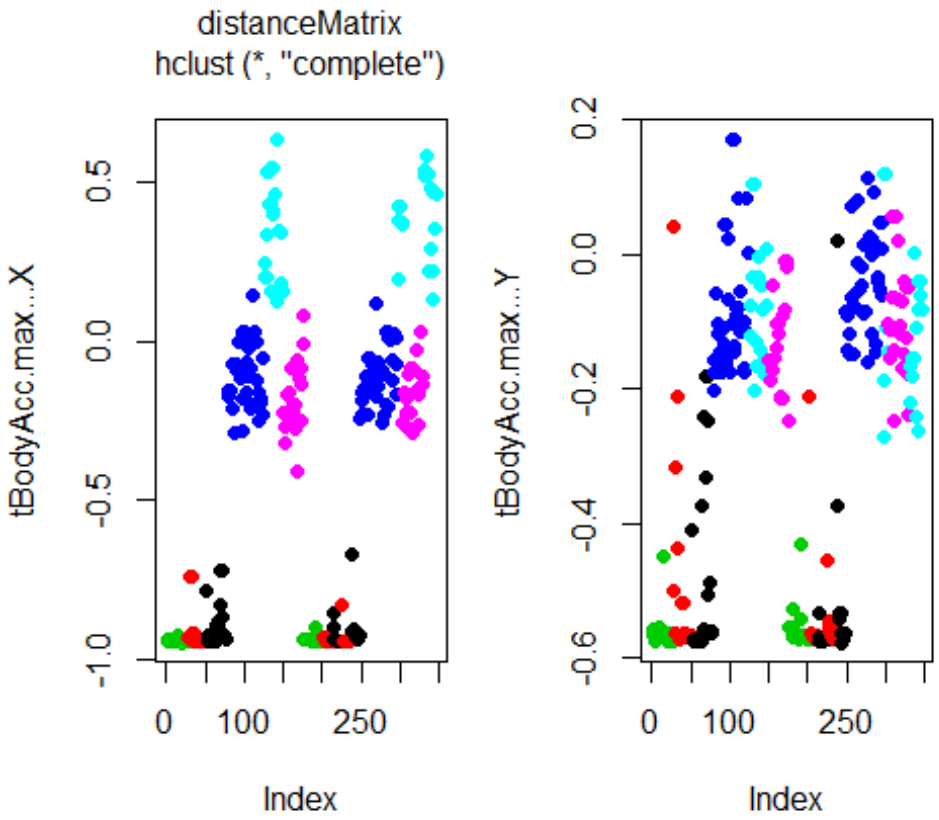
```
data <- read.csv("SamsungData.csv")
table(data$activity)

##
##   laying  sitting standing     walk walkdown   walkup
##     1407     1286     1374     1226      986     1073

par(mfrow = c(1,2), mar = c(5,4,1,1))
data <- transform(data, activity = factor(activity))
sub1 <- subset(data, subject == 1)
plot(sub1[, 2], col = sub1$activity, ylab = names(sub1)[2])#Ploteo las
primeras dos mediciones para el primer sujeto
plot(sub1[, 3], col = sub1$activity, ylab = names(sub1)[3])
legend("bottomright", legend = unique(sub1$activity), col =
unique(sub1$activity), pch = 1)
```




```
source("myplclust.R")
distanceMatrix <- dist(sub1[, 2:4])
```

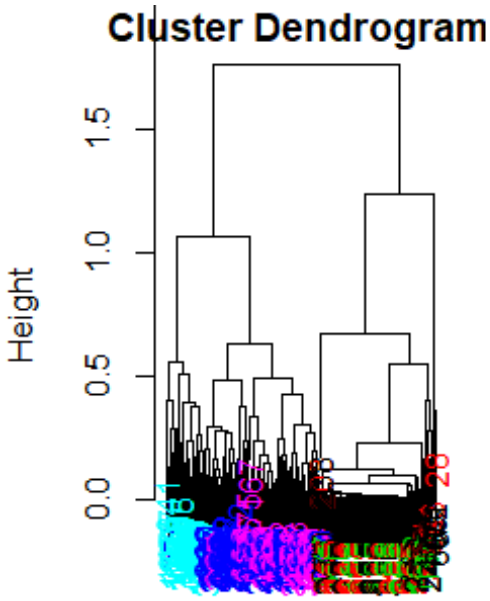
```r
hclustering <- hclust(distanceMatrix)
myplclust(hclustering, lab.col = unclass(sub1$activity))# Creamos un
dendograma que no nos entrega mucha informacion

{par(mfrow = c(1,2), xpd = TRUE)
plot(sub1[, 11], pch = 19, col = sub1$activity, ylab = names(sub1)[11])
plot(sub1[, 12], pch = 19, col = sub1$activity, ylab =
names(sub1)[12])}#Ahora puedo plotear los maximos y encuentro una
diferencia mas notoria sentarse y no hacer nada no entrega informacion.
Pero moverse subir escaleras existe mas variabilidad
```

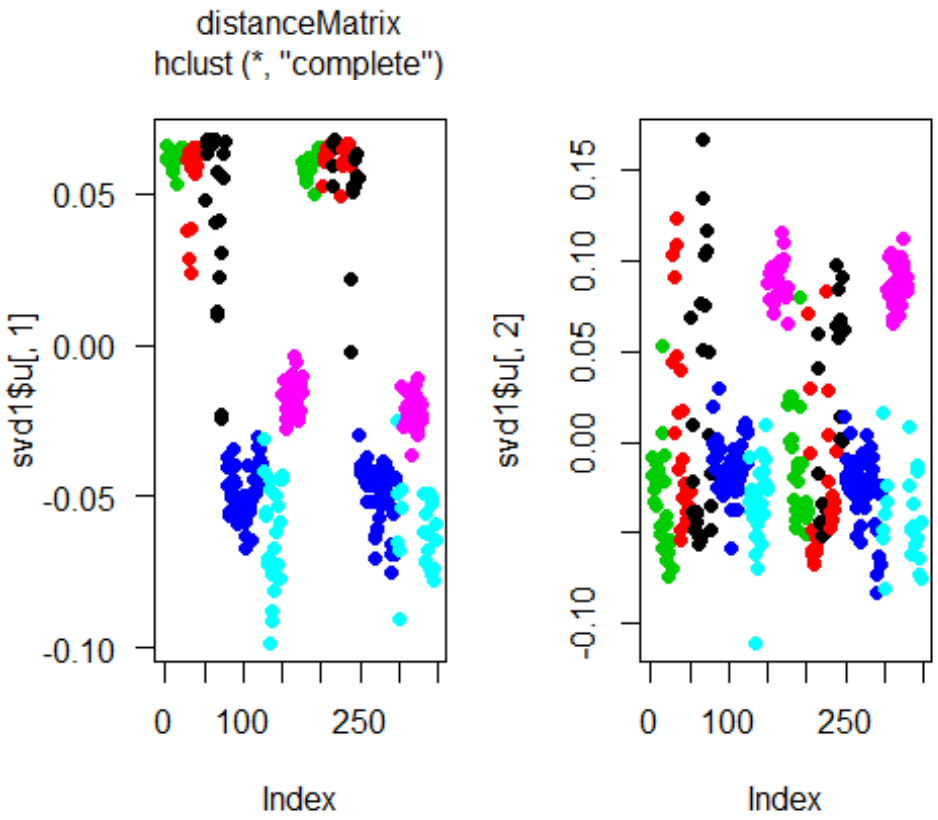





```r
distanceMatrix <- dist(sub1[, 11:13])
hclustering <- hclust(distanceMatrix)
myplclust(hclustering, lab.col = unclass(sub1$activity))# Creamos un
dendrograma que ahora si nos entrega la informacion de que existe dos
```

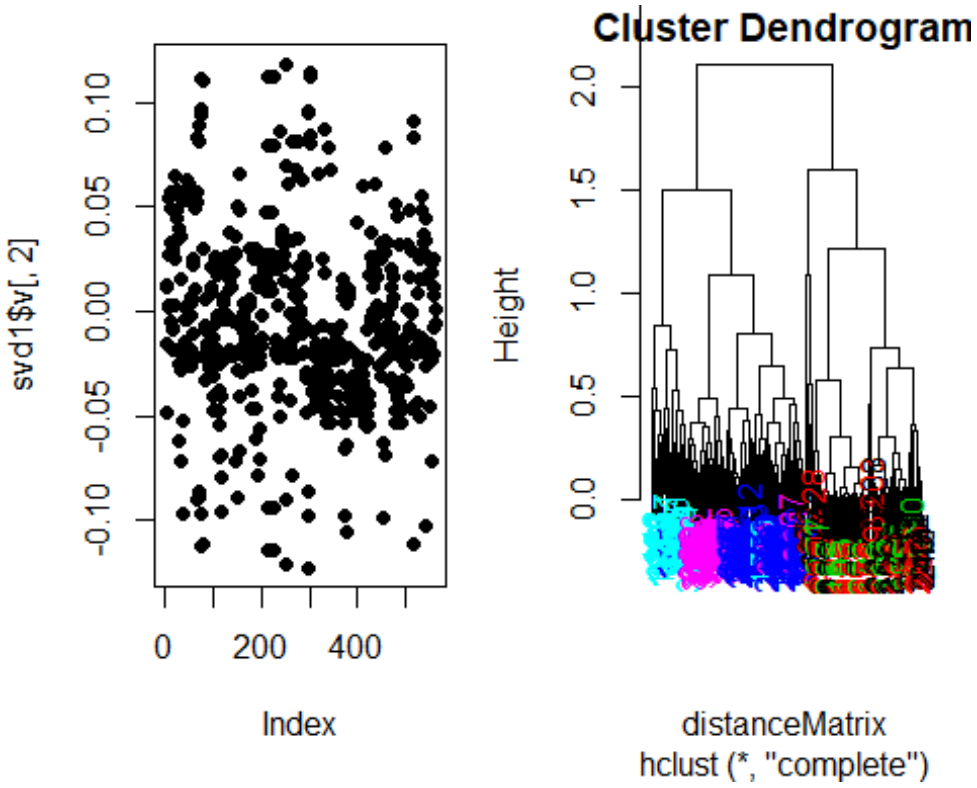
```r
clusters mas marcados

svd1 <- svd(scale(sub1[, -c(563,564)]))
{par(mfrow = c(1,2))
plot(svd1$u[, 1], col = sub1$activity, pch = 19, las = 1)
plot(svd1$u[, 2], col = sub1$activity, pch = 19)}
```






```r
plot(svd1$v[, 2], pch = 19)#Veo cual es la columna que genera mas varianza
maxContrib <- which.max(svd1$v[, 2]) #Entrega cual es la columna que entrega una mayor variacion a los datos
```

```
distanceMatrix <- dist(sub1[, c(11:13, maxContrib)]) #calculamos la
distancia del maximo mas este extra de la maxima contribucion
hclustering <- hclust(distanceMatrix)
myplclust(hclustering, lab.col = unclass(sub1$activity))# Las 3
actividades de mas movimiento estan mejor separadas por clusters, las sin
movimiento no generan mucha diferencia
```




```
names(data)[maxContrib]

## [1] "tBodyGyroMag.arCoeff..2"

kClust <- kmeans(sub1[, -c(563,564)], centers = 6, nstart = 100) #TK
cluster puede encontrar distintos dependiendo de donde parta
table(kClust$cluster, sub1$activity)

##
##     laying sitting standing walk walkdown walkup
##   1      0       0        0    5       24     27
##   2     11       1        0   43        0      0
##   3     10      24       27    0        0      0
##   4     12      22       26    0        0      0
##   5     17       0        0   41        0      0
##   6      0       0        0    6       25     26
```