Algunos ejemplos en R para Exploración de Datos

Para más detalles, ver *Data Mining and Predictive Analytics – Daniel T. Larose y Chantal D. Larose – Wiley 2015*

 # Leer los datasets Cars y Cars2 datasets:
Reemplazar "C:/ … /" con la localización exacta del archivo que se desea abrir

```
cars <- read.csv("C:/ …/cars.txt", stringsAsFactors = FALSE)
cars2 <- read.csv("C:/ …/cars2.txt", stringsAsFactors = FALSE)
```

(cars cuenta con 261 observaciones y 8 variables, mientras que cars2 cuenta con 263 observaciones y 8 variables)

 # Datos Faltantes
 # Ver cuatro variables del conjunto *cars*

```
cars.4var <-cars[,c(1,3,4,8)]
head(cars.4var)
```

261 observaciones y 4 variables

```
mpg cubicinches  hp    brand
1 14.0          350 165      US.
2 31.9           89  71  Europe.
3 17.0          302 140      US.
4 15.0          400 150      US.
5 30.5           98  63      US.
6 23.0          350 125      US.
```

 # Convertir a *Missing* algunos valores

```
cars.4var[2,2] <- cars.4var[4,4] <-NA
head(cars.4var)
```

```
   mpg cubicinches  hp    brand
1 14.0          350 165      US.
2 31.9           NA  71  Europe.
3 17.0          302 140      US.
4 15.0          400 150     <NA>
5 30.5           98  63      US.
6 23.0          350 125      US.
```

 # Reemplazar valores faltantes con constantes

```
cars.4var[2,2] <-0
cars.4var[4,4] <- "Missing"
head(cars.4var)
```

```
   mpg cubicinches  hp    brand
1 14.0         350 165      US.
2 31.9           0  71  Europe.
3 17.0         302 140      US.
4 15.0         400 150  Missing
5 30.5          98  63      US.
6 23.0         350 125      US.
```

Reemplazar valores con la media y la moda

```
cars.4var[2,2] <- mean(na.omit(cars.4var$cubicinches))
our_table <- table(cars.4var$brand)
our_mode <- names(our_table) [our_table == max(our_table)]
cars.4var[4,4] <- our_mode
head(cars.4var)
```

```
   mpg cubicinches  hp    brand
1 14.0    350.0000 165      US.
2 31.9    200.7625  71  Europe.
3 17.0    302.0000 140      US.
4 15.0    400.0000 150      US.
5 30.5     98.0000  63      US.
6 23.0    350.0000 125      US.
```

Generar observaciones aleatorias

```
obs_brand <- sample(na.omit(cars.4var$brand), 1)
obs_cubicinches <-sample(na.omit(cars.4var$cubicinches), 1)
cars.4var[2,2] <- obs_cubicinches
cars.4var[4,4] <- obs_brand
head(cars.4var)
```

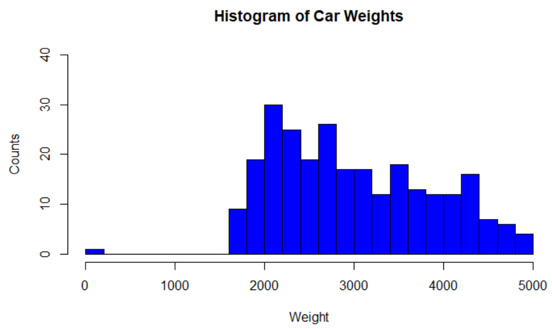
```
   mpg cubicinches  hp    brand
1 14.0         350 165      US.
2 31.9          97  71  Europe.
3 17.0         302 140      US.
4 15.0         400 150      US.
5 30.5          98  63      US.
6 23.0         350 125      US.
```

Crear un Histograma
Definir el área de dibujo

```
par(mfrow = c(1,1))
```

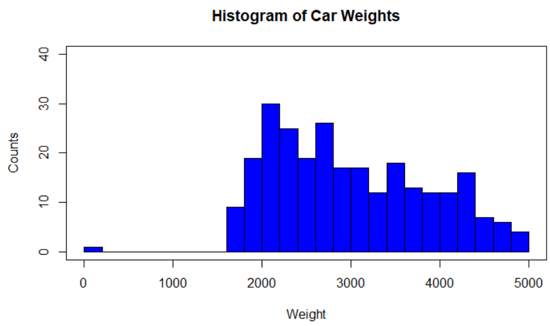
```
# Crear el histograma
```

```
hist(cars2$weight, breaks = 30, xlim= c(0, 5000), col = "blue", border = "black", ylim= c(0, 40), xlab = "Weight", ylab = "Counts", main = "Histogram of Car Weights")
```



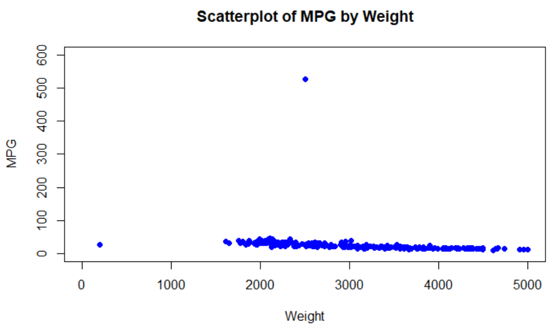
Colocar una línea alrededor del gráfico

```
box(which = "plot", lty = "solid", col = "black")
```



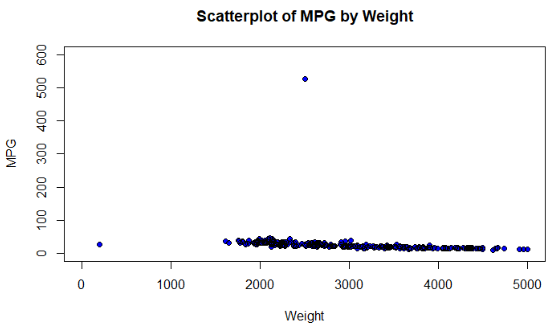
Crear un gráfico de dispersión

```
plot(cars2$weight, cars2$mpg, xlim= c(0, 5000), ylim= c(0, 600), xlab = "Weight", ylab = "MPG", main = "Scatterplot of MPG by Weight", type = "p", pch = 16, col = "blue")
```



#Agregar los puntos como círculos negros sin relleno

```
points(cars2$weight, cars2$mpg, type = "p", col = "black")
```



Estadísticas Descriptivas

```
mean(cars$weight)   # Mean
median(cars$weight)         # Median
length(cars$weight)         # Number of observations
sd(cars$weight)     # Standard deviation
summary(cars$weight)        # Min, Q1, Median, Mean, Q3, Max
```

```
> mean(cars$weight)        # Mean
[1] 3005.49
> median(cars$weight)      # Median
[1] 2835
> length(cars$weight)      # Number of observations
[1] 261
> sd(cars$weight)          # Standard deviation
[1] 852.6456
> summary(cars$weight)     # Min, Q1, Median, Mean, Q3, Max
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1613    2246    2835    3005    3664    4997
```

Transformaciones
Min-max normalization

```
summary(cars$weight)
mi <- min(cars$weight)
ma <- max(cars$weight)
minmax.weight <- (cars$weight - mi)/(ma - mi)
minmax.weight
```

```
[1] 0.76713948 0.09219858 0.54255319 0.63475177 0.12943262 0.67582742 0.81264775 0.79757683
  [9] 0.56648936 0.12913712 0.18676123 0.16991726 0.74704492 0.60608747 0.52245863 0.81176123
 [17] 0.12027187 0.53782506 0.19651300 0.24143026 0.11997636 0.29166667 0.41341608 0.72133570
 [25] 0.10490544 0.05378251 0.04728132 0.89952719 0.57949173 0.22665485 0.15277778 0.49202128
 [33] 0.26388889 0.30585106 0.10992908 0.53250591 0.06560284 0.32712766 0.69208038 0.48817967
 [41] 0.67494090 0.89391253 0.60047281 0.56353428 0.43528369 0.30200946 0.63297872 0.23108747
…
```

Z-score standarization

```
m <- mean(cars$weight); s <- sd(cars$weight)
z.weight <- (cars$weight - m)/s
z.weight
length(cars$weight)
```

```
[1]  1.411500375 -1.267221019  0.520157004  0.886076914 -1.119445671  1.049098925  1.592114690
  [8]  1.532300858  0.615155442 -1.120618491 -0.891918547 -0.958769300  1.331748600  0.772313352
 [15]  0.440405229  1.588596229 -1.155803098  0.501391880 -0.853215480 -0.674946806 -1.156975918
 [22] -0.475567367  0.007634565  1.229713240 -1.216789750 -1.419687649 -1.445489694  1.936923836
 [29]  0.666759532 -0.733587817 -1.026792873  0.319604745 -0.585812469 -0.419271997 -1.196851806
 [36]  0.480281116 -1.372774840 -0.334828940  1.113604038  0.304358082  1.045580465  1.914640252
…
```

Decimal scaling

```
max(abs(cars$weight)) # 4 digits
d.weight <- cars$weight/(10^4); d.weight
```

(el Maximo es 4997)

```
[1] 0.4209 0.1925 0.3449 0.3761 0.2051 0.3900 0.4363 0.4312 0.3530 0.2050 0.2245 0.2188 0.4141
 [14] 0.3664 0.3381 0.4360 0.2020 0.3433 0.2278 0.2430 0.2019 0.2600 0.3012 0.4054 0.1968 0.1795
 [27] 0.1773 0.4657 0.3574 0.2380 0.2130 0.3278 0.2506 0.2648 0.1985 0.3415 0.1835 0.2720 0.3955
 [40] 0.3265 0.3897 0.4638 0.3645 0.3520 0.3086 0.2635 0.3755 0.2395 0.1940 0.3060 0.4464 0.3190
…
```

Histogramas lado a lado

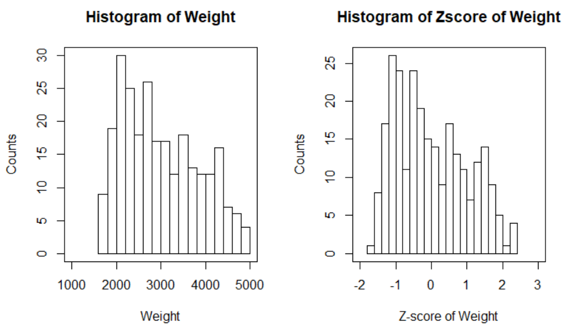
```
par(mfrow = c(1,2))
```

Crear dos histogramas

```
hist(cars$weight, breaks = 20,
xlim = c(1000, 5000),
main = "Histogram of Weight",
xlab = "Weight",
```

```
ylab = "Counts")
box(which = "plot",
lty = "solid",
col = "black")

hist(z.weight,
breaks = 20, xlim = c(-2, 3),
main = "Histogram of Zscore of Weight",
xlab = "Z-score of Weight",
ylab = "Counts")
box(which = "plot",
lty = "solid",
col = "black")
```



Sesgo

```
(3*(mean(cars$weight) - median(cars$weight)))/sd(cars$weight)
(3*(mean(z.weight) - median(z.weight)))/sd(z.weight)
```

(es el mismo valor en ambos casos…)

Transformations for Normality

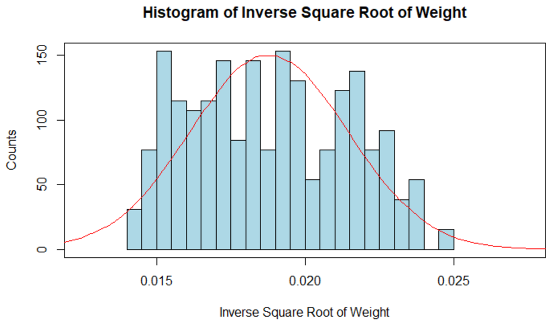
```
sqrt.weight <- sqrt(cars$weight)                # Square root
sqrt.weight_skew <- (3*(mean(sqrt.weight) - median(sqrt.weight))) / sd(sqrt.weight)
ln.weight <- log(cars$weight)                   # Natural log
ln.weight_skew <- (3*(mean(ln.weight) - median(ln.weight))) / sd(ln.weight)
invsqrt.weight <- 1 / sqrt(cars$weight)         # Inverse square root
invsqrt.weight_skew <- (3*(mean(invsqrt.weight) - median(invsqrt.weight))) /
sd(invsqrt.weight)
```

Sesgo para transformación raíz cuadrada = 0.4028
Sesgo para Logaritmo Natural = 0.1956
Sesgo para Raíz Cuadrada Inversa = 0.0154

Histograma con curva de distribución normal

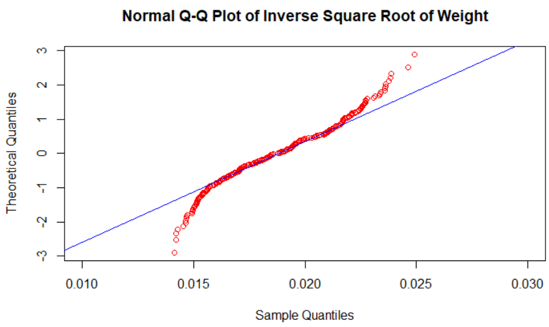
```
par(mfrow=c(1,1))
x <- rnorm(1000000, mean = mean (invsqrt.weight), sd = sd(invsqrt.weight))
hist(invsqrt.weight, breaks = 30, xlim =c(0.0125, 0.0275), col = "lightblue", prob =
TRUE, border = "black", xlab="Inverse Square Root of Weight", ylab = "Counts", main =
"Histogram of Inverse Square Root of Weight")
box(which = "plot", lty = "solid", col="black")
```

```
# Densidad Normal
lines(density(x), col="red")
```



Normal Q-Q Plot

```
qqnorm(invsqrt.weight, datax = TRUE, col = "red", ylim = c(0.01, 0.03), main = "Normal Q-Q Plot of Inverse Square Root of Weight")
qqline(invsqrt.weight, col = "blue", datax = TRUE)
```



Detransformar datos

```
# Transformar x empleando y = 1 / sqrt(x)
x <- cars$weight[1]; y <- 1 / sqrt(x)

# Detransformar x empleando x = 1 / (y)^2
detransformedx <- 1 / y^2
x; y; detransformedx
```

```
> x; y; detransformedx
[1] 4209
[1] 0.01541383
[1] 4209
```

Crear variables indicadoras

north_flag <- east_flag <- south_flag <- c(rep(NA, 10))

region <- c(rep(c("north", "south", "east", "west"),2), "north", "south")
# Change the region variable to indicators
for (i in 1:length(region)) {
  if(region[i] == "north") north_flag[i] = 1 else north_flag[i] = 0
  if(region[i] == "east") east_flag[i] = 1 else east_flag[i] = 0
  if(region[i] == "south") south_flag[i] = 1 else south_flag[i] = 0 }
north_flag; east_flag; south_flag

```
north_flag; east_flag; south_flag
 [1] 1 0 0 0 1 0 0 0 1 0
 [1] 0 0 1 0 0 0 1 0 0 0
 [1] 0 1 0 0 0 1 0 0 0 1
```

 # Index fields

```
# Data frames have an index field;
# the left-most column cars
cars[order(cars$mpg),]

# For vectors or matrices,
# add a column to act as an index field
x <- c(1,1,3:1,1:4,3);
y <- c(9,9:1)
z <- c(2,1:9)
matrix <- t(rbind(x,y,z)); matrix
indexed_m <- cbind(c(1:length(x)), matrix); indexed_m
indexed_m[order(z),]
```

```
> matrix <- t(rbind(x,y,z)); matrix
      x y z
 [1,] 1 9 2
 [2,] 1 9 1
 [3,] 3 8 2
 [4,] 2 7 3
 [5,] 1 6 4
 [6,] 1 5 5
 [7,] 2 4 6
 [8,] 3 3 7
 [9,] 4 2 8
[10,] 3 1 9
> indexed_m <- cbind(c(1:length(x)), matrix); indexed_m
         x y z
 [1,]  1 1 9 2
 [2,]  2 1 9 1
 [3,]  3 3 8 2
 [4,]  4 2 7 3
 [5,]  5 1 6 4
 [6,]  6 1 5 5
 [7,]  7 2 4 6
 [8,]  8 3 3 7
 [9,]  9 4 2 8
[10,] 10 3 1 9
> indexed_m[order(z),]
         x y z
```

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

 # Duplicate records

```
# For number of duplicate records, use anyDuplicated
anyDuplicated(cars)
# To examine each record, use Duplicated
duplicated(cars)
# 'True': record is a duplicate,
# 'False': record is not a duplicate
# Let's duplicate the first record
new.cars <- rbind(cars, cars[1,])
# Check for duplicates
anyDuplicated(new.cars)
# The 262nd record is a duplicate
duplicated(new.cars)
```

Código en R para la exploración de datos del caso 2

Leer el conjunto de datos

```
churn <- read.csv(file = "C:/ … /churn.txt", stringsAsFactors=TRUE)
# Mostrar los primeros diez registros
churn[1:10,]
```

```
State Account.Length Area.Code    Phone Int.l.Plan VMail.Plan VMail.Message Day.Mins Day.Calls
1     KS            128       415 382-4657         no        yes            25    265.1       110
2     OH            107       415 371-7191         no        yes            26    161.6       123
3     NJ            137       415 358-1921         no         no             0    243.4       114
4     OH             84       408 375-9999        yes         no             0    299.4        71
5     OK             75       415 330-6626        yes         no             0    166.7       113
6     AL            118       510 391-8027        yes         no             0    223.4        98
7     MA            121       510 355-9993         no        yes            24    218.2        88
8     MO            147       415 329-9001        yes         no             0    157.0        79
9     LA            117       408 335-4719         no         no             0    184.5        97
10    WV            141       415 330-8173        yes        yes            37    258.6        84
   Day.Charge Eve.Mins Eve.Calls Eve.Charge Night.Mins Night.Calls Night.Charge Intl.Mins Intl.Cal
ls
1       45.07    197.4        99      16.78      244.7          91        11.01      10.0
 3
2       27.47    195.5       103      16.62      254.4         103        11.45      13.7
 3
3       41.38    121.2       110      10.30      162.6         104         7.32      12.2
 5
4       50.90     61.9        88       5.26      196.9          89         8.86       6.6
 7
5       28.34    148.3       122      12.61      186.9         121         8.41      10.1
 3
6       37.98    220.6       101      18.75      203.9         118         9.18       6.3
 6
7       37.09    348.5       108      29.62      212.6         118         9.57       7.5
 7
8       26.69    103.1        94       8.76      211.8          96         9.53       7.1
 6
9       31.37    351.6        80      29.89      215.8          90         9.71       8.7
 4
10      43.96    222.0       111      18.87      326.4          97        14.69      11.2
 5
   Intl.Charge CustServ.Calls Churn.
1         2.70              1 False.
2         3.70              1 False.
3         3.29              0 False.
4         1.78              2 False.
5         2.73              3 False.
6         1.70              0 False.
7         2.03              3 False.
8         1.92              0 False.
9         2.35              1 False.
10        3.02              0 False.
```

Sumarizar la variable CHURN

```
sum.churn <- summary(churn$Churn)
sum.churn
```

```
> sum.churn
False.  True.
  2850    483
```

Calcular la proporción de CHURNERS en el total de registros

```
prop.churn <- sum(churn$Churn == "True.") / length(churn$Churn)
prop.churn
```

```
> prop.churn
[1] 0.1449145
```

Grafico de Barras de la variable Churn

```
barplot(sum.churn, ylim = c(0, 3000), main = "Bar Graph of Churners and Non-Churners",
col = "lightblue")
box(which = "plot", lty = "solid", col="black")
```

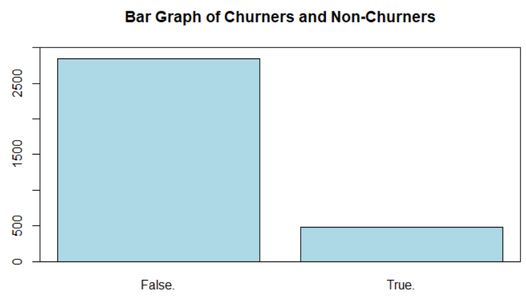


Tabla doble entrada para churners e International Plan

```
counts <- table(churn$Churn, churn$Int.l.Plan, dnn=c("Churn", "International Plan"))
counts
```

```
> counts
        International Plan
Churn      no  yes
  False. 2664  186
  True.   346  137
```

Grafico de Barras Superpuesto Proporciones de churners por International Plan

```
barplot(counts, legend = rownames(counts), col = c("blue", "red"), ylim = c(0, 3300),
ylab = "Count", xlab = "International Plan", main = "Comparison Bar Chart: Churn
Proportions by International Plan")
box(which = "plot", lty = "solid", col="black")
```

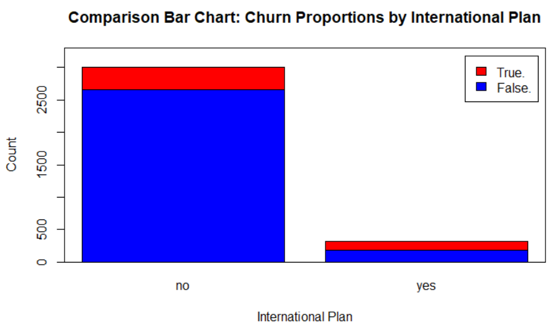


Tabla suma para ambas variables

```
sumtable <- addmargins(counts, FUN = sum)
sumtable
```

```
> sumtable
        International Plan
Churn      no  yes  sum
  False. 2664  186 2850
  True.   346  137  483
  sum    3010  323 3333
```

Crear una tabla de proporciones sobre las filas

```
row.margin <- round(prop.table(counts, margin = 1), 4)*100
row.margin
```

```
> row.margin
        International Plan
Churn        no   yes
  False. 93.47  6.53
  True.  71.64 28.36
```

Crear una table de proporciones sobre las columnas

```
col.margin <- round(prop.table(counts, margin = 2), 4)*100
col.margin
```

```
> col.margin
        International Plan
Churn        no   yes
  False. 88.50 57.59
  True.  11.50 42.41
```

Grafico de Barras Agrupado, con referencias

```
barplot(counts, col = c("blue", "red"), ylim = c(0, 3300), ylab = "Count", xlab =
"International Plan", main = "Churn Count by International Plan", beside = TRUE)
legend("topright", c(rownames(counts)), col = c("blue", "red"), pch = 15, title =
"Churn")
box(which = "plot", lty = "solid", col="black")
```

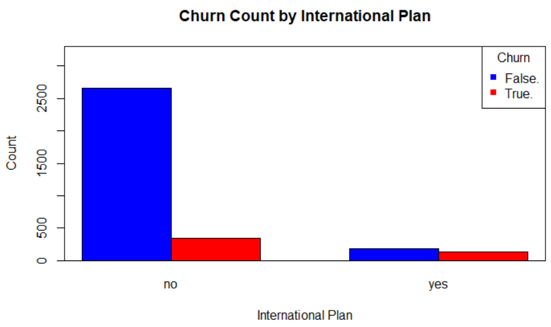
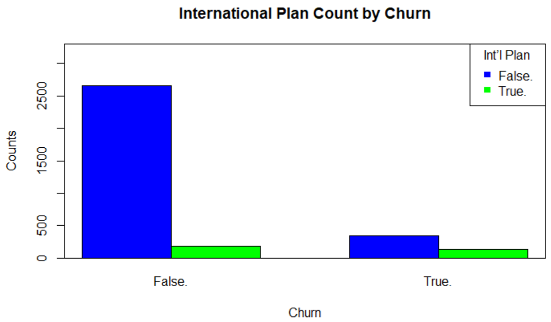


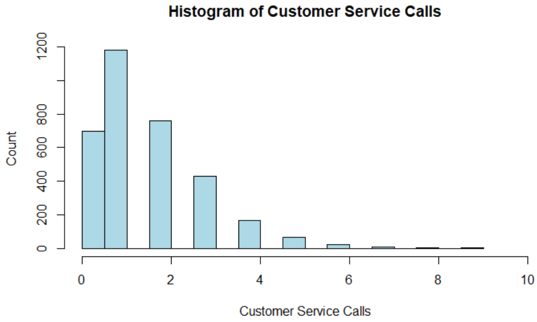
Grafico de Barras Agrupado de Churn e International Plan con referencias

```
barplot(t(counts), col = c("blue", "green"), ylim = c(0, 3300), ylab = "Counts", xlab =
"Churn", main = "International Plan Count by Churn", beside = TRUE)
legend("topright", c(rownames(counts)), col = c("blue", "green"), pch = 15, title =
"Int'l Plan")
box(which = "plot", lty = "solid", col="black")
```



Histograma de las llamadas a atención al cliente (Customer Service Calls)

```
hist(churn$CustServ.Calls, xlim = c(0,10), col = "lightblue", ylab = "Count", xlab =
"Customer Service Calls", main = "Histogram of Customer Service Calls")
```

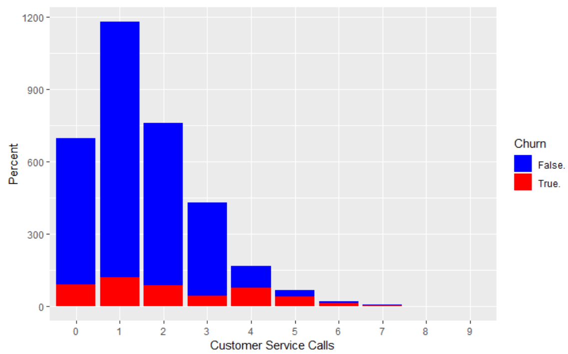


Asegurarse tener instalado el paquete **ggplot2**, sino:

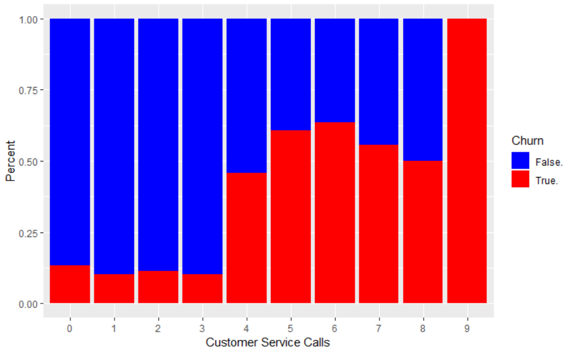
```
install.packages("ggplot2")
# cargarlo
library(ggplot2)
```

Gráficos de Barras Superpuestos

```
ggplot() +
   geom_bar(data = churn, aes(x = factor(churn$CustServ.Calls), fill =
factor(churn$Churn)), position = "stack") +
   scale_x_discrete("Customer Service Calls") +
   scale_y_continuous("Percent") +
   guides(fill=guide_legend(title="Churn")) +
   scale_fill_manual(values=c("blue", "red"))
```




```
ggplot() +
   geom_bar(data=churn, aes(x = factor(churn$CustServ.Calls), fill =
factor(churn$Churn)), position = "fill") +
   scale_x_discrete("Customer Service Calls") +
   scale_y_continuous("Percent") +
   guides(fill=guide_legend(title="Churn"))+
   scale_fill_manual(values=c("blue", "red"))
```



Test T para dos muestras con la variable Int'l Calls

```
# Partitir los datos
churn.false <- subset(churn, churn$Churn == "False.")
churn.true <- subset(churn, churn$Churn == "True.")
# Correr el test t
t.test(churn.false$Intl.Calls, churn.true$Intl.Calls)
```

```
> t.test(churn.false$Intl.Calls, churn.true$Intl.Calls)

        Welch Two Sample t-test

data:  churn.false$Intl.Calls and churn.true$Intl.Calls
t = 2.9604, df = 640.64, p-value = 0.003186
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.1243807 0.6144620
sample estimates:
mean of x mean of y
 4.532982  4.163561
```

Grafico de Dispersión de las variables Evening Minutes y Day Minutes, clasificados por Churn

```
 plot(churn$Eve.Mins,
   churn$Day.Mins,
   xlim = c(0, 400),
   ylim = c(0, 400),
   xlab = "Evening Minutes",
   ylab = "Day Minutes",
   main = "Scatterplot of Day and Evening Minutes by Churn",
   col = ifelse(churn$Churn == "True.", "red", "blue"))
legend("topright",
   c("True", "False"),
   col = c("red", "blue"),
   pch = 1,
   title = "Churn")
```

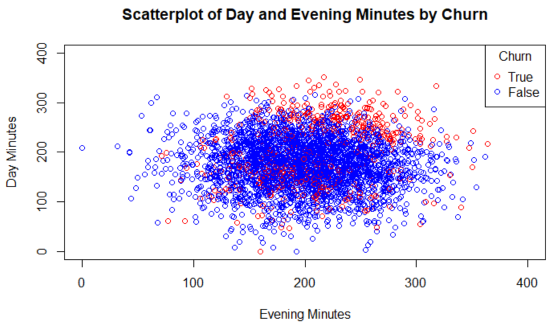
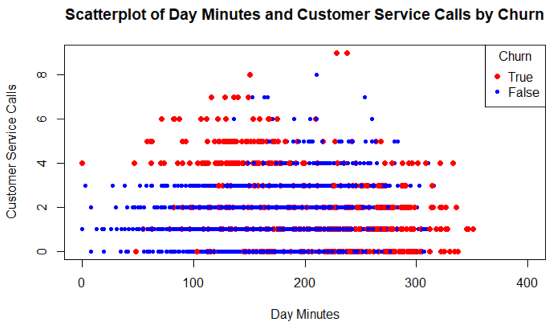


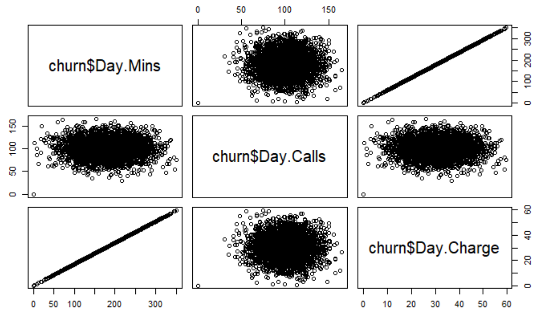
Grafico de dispersión de las variables Day Minutes y Customer Service Calls, coloreado por Churn

```
plot(churn$Day.Mins,
   churn$CustServ.Calls,
   xlim = c(0, 400),
   xlab = "Day Minutes",
   ylab = "Customer Service Calls",
   main = "Scatterplot of Day Minutes and Customer Service Calls by Churn",
   col = ifelse(churn$Churn=="True.", "red", "blue"),
   pch = ifelse(churn$Churn=="True.", 16, 20))
legend("topright",
   c("True", "False"),
   col = c("red", "blue"),
   pch = c(16, 20),
   title = "Churn")
```



Matriz de Dispersion

```
pairs(~churn$Day.Mins+
   churn$Day.Calls+
   churn$Day.Charge)
```



Regresion de Day Charge vs Day Minutes

```
fit <- lm(churn$Day.Charge ~ churn$Day.Mins)
summary(fit)

Call:
lm(formula = churn$Day.Charge ~ churn$Day.Mins)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0045935 -0.0025391  0.0004326  0.0024587  0.0045224

Coefficients:
                Estimate Std. Error   t value Pr(>|t|)
(Intercept)    6.134e-04  1.711e-04 3.585e+00 0.000341 ***
churn$Day.Mins 1.700e-01  9.108e-07 1.866e+05  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.002864 on 3331 degrees of freedom
Multiple R-squared:      1,	Adjusted R-squared:      1
F-statistic: 3.484e+10 on 1 and 3331 DF,  p-value: < 2.2e-16
```

Correlación con valores p

```
days <- cbind(churn$Day.Mins, churn$Day.Calls, churn$Day.Charge)
MinsCallsTest <- cor.test(churn$Day.Mins, churn$Day.Calls)
MinsChargeTest <- cor.test(churn$Day.Mins, churn$Day.Charge)
CallsChargeTest <- cor.test(churn$Day.Calls, churn$Day.Charge)
round(cor(days), 4)
MinsCallsTest$p.value
MinsChargeTest$p.value
CallsChargeTest$p.value
```

```
> round(cor(days), 4)
       [,1]   [,2]   [,3]
[1,] 1.0000 0.0068 1.0000
[2,] 0.0068 1.0000 0.0068
[3,] 1.0000 0.0068 1.0000
> MinsCallsTest$p.value
[1] 0.6968515
> MinsChargeTest$p.value
```

```
[1] 0
> CallsChargeTest$p.value
[1] 0.6967428
```

 # Correlación con valores p en forma matricial

```
# Reunir las variables de interes
corrdata <-
   cbind(churn$Account.Length,
   churn$VMail.Message,
   churn$Day.Mins,
   churn$Day.Calls,
   churn$CustServ.Calls)
# Declarar la matriz
corrpvalues <- matrix(rep(0, 25), ncol = 5)
# Llenar la matriz con las correlaciones
for (i in 1:4) {
   for (j in (i+1):5) {
      corrpvalues[i,j] <-
      corrpvalues[j,i] <-
          round(cor.test(corrdata[,i], corrdata[,j])$p.value, 4) } }
round(cor(corrdata), 4)
corrpvalues
```

```
> round(cor(corrdata), 4)
        [,1]    [,2]    [,3]    [,4]    [,5]
[1,]  1.0000 -0.0046  0.0062  0.0385 -0.0038
[2,] -0.0046  1.0000  0.0008 -0.0095 -0.0133
[3,]  0.0062  0.0008  1.0000  0.0068 -0.0134
[4,]  0.0385 -0.0095  0.0068  1.0000 -0.0189
[5,] -0.0038 -0.0133 -0.0134 -0.0189  1.0000
> corrpvalues
       [,1]   [,2]   [,3]   [,4]   [,5]
[1,] 0.0000 0.7894 0.7198 0.0264 0.8266
[2,] 0.7894 0.0000 0.9642 0.5816 0.4440
[3,] 0.7198 0.9642 0.0000 0.6969 0.4385
[4,] 0.0264 0.5816 0.6969 0.0000 0.2743
[5,] 0.8266 0.4440 0.4385 0.2743 0.0000
```

Algunas notas de Arboles de Decisión con R

En el caso del TP_2 se sugiere:

a) Reducir algunas categorías de las variables, por ejemplo, en *marital_status y workclass*
b) Estandarizar las variables numéricas

Para construir el árbol, es posible emplear el paquete rpart
Para dibujar el árbol, es posible utilizar el paquete rpart.plot
También es posible construir el árbol utilizando C5.0 (paquete C50)

1. Kuhn M, Weston S, Coulter N. 2013. C code for C5.0 by R. Quinlan. C50: C5.0 decision trees and rule-based models. R package version 0.1.0-15. http://CRAN.R-project.org/package=C50.
2. Milborrow S. 2012. rpart.plot: Plot rpart models. An enhanced version of plot.rpart. R package version 1.4-3. http://CRAN.R-project.org/package=rpart.plot.
3. Therneau T, Atkinson B, Ripley B. 2013. rpart: Recursive partitioning. R package version 4.1-3. http://CRAN.R-project.org/package=rpart.
4. R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing; 2012. ISBN: 3-900051-07-0, http://www.R-project.org/.