Análisis Estadístico con R

Estadística Descriptiva

Víctor Morales-Oñate

22 de julio de 2018

Contents

Manipulate

El paquete `manipulate` es una forma de hacer que los gráficos estándar en R se vuelvan interactivos. Veamos unos ejemplos básicos.

Empecemos con un gráfico donde se desea manipular un parámetro con un `slider`:

```
# install.packages("manipulate")
library(manipulate)
manipulate(plot(1:x), x = slider(1, 100))
```

Otro ejemplo:

```
manipulate(
  plot(cars, xlim=c(0,x.max)),
  x.max=slider(15,25))
```

Control de selección

```
manipulate(
  barplot(as.matrix(longley[,factor]),
          beside = TRUE, main = factor),
  factor = picker("GNP", "Unemployed", "Employed"))
```

Control con caja

```
manipulate(
  boxplot(Freq ~ Class, data = Titanic, outline = outline),
  outline = checkbox(FALSE, "Mostrar outliers"))
```

Combinando controles

```
manipulate(
  plot(cars, xlim = c(0, x.max), type = type, ann = label),
  x.max = slider(10, 25, step=5, initial = 25),
  type = picker("Puntos" = "p", "Lineas" = "l", "Pasos" = "s"),
  label = checkbox(TRUE, "Etiquetas"))
```

Estadística Descriptiva

Antes de empezar con la elaboración de modelos (modelos lineales por ejemplo), es útil hacer análisis de los datos de modo descriptivo y gráfico.

Resúmen estadístico

- Es fácil calcular estadísticos de resumen en `R`. Aquí tienen cómo calcular la media, desviación estándar, varianza y mediana.

```
x <- rnorm(50)
mean(x)
```

```
## [1] -0.1232708
```

```
sd(x)
```

```
## [1] 1.172418
```

```
var(x)
```

```
## [1] 1.374564
```

```
median(x)
```

```
## [1] -0.04978008
```

Recuerda que estamos usando la generación de números aleatorios, de modo que nuestros resultados no van a coincidir.

- Los *quantiles* pueden ser obtenidos con la función `quantile`.

```
quantile(x)
```

```
##          0%         25%         50%         75%        100%
## -2.59786682 -0.99370900 -0.04978008  0.53029047  2.78269141
```

Nota que por defecto obtienes el mínimo, máximo y tres *cuartiles* (esto es, los *cuantiles* 0.25, 0.50 y 0.75), se llaman así porque se divide en 4 partes. Se puede tener *deciles*.

A la diferencia entre el primer y tercer cuartil se llama *rango intercuartil* (IQR - siglas en inglés) y a veces es usando como una alterniativa *robusta* a la desviación estándar.

- Es posible obtener otros cuantiles añadiento el argumento que contiene los puntos deseados.

```
pvec <- seq(0,1,0.1)
pvec
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
quantile(x,pvec)
```

```
##          0%         10%         20%         30%         40%         50%
## -2.59786682 -1.58885335 -1.05978240 -0.82313002 -0.47718434 -0.04978008
##         60%         70%         80%         90%        100%
##  0.20410006  0.49226071  0.79538624  1.47878350  2.78269141
```

Veamos un ejemplo cuando hay datos perdidos.

```
# install.packages("ISwR")
library(ISwR)
data(juul)
attach(juul)
mean(igf1)
```

```
## [1] NA
```

- `R` usa los datos perdidos si no se le pide lo contrario. Sin embargo, se puede usar el argumento `na.rm` para solucionar el problema

```
mean(igf1,na.rm=T)
```

```
## [1] 340.168
```

- Ahora, contemos el número de valores perdidos en la variable

```
sum(!is.na(igf1))
```

```
## [1] 1018
```

- Una forma agradable y directa de obtener lo anterior:

```
summary(igf1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    25.0   202.2   313.5   340.2   462.8   915.0     321
```

- De hecho, se puede hacer un resumen de todo el data frame:

```
summary(juul)
```

```
##       age            menarche          sex             igf1
##  Min.   : 0.170   Min.   :1.000   Min.   :1.000   Min.   : 25.0
##  1st Qu.: 9.053   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:202.2
##  Median :12.560   Median :1.000   Median :2.000   Median :313.5
##  Mean   :15.095   Mean   :1.476   Mean   :1.534   Mean   :340.2
##  3rd Qu.:16.855   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:462.8
##  Max.   :83.000   Max.   :2.000   Max.   :2.000   Max.   :915.0
##  NA's   :5        NA's   :635     NA's   :5       NA's   :321
##      tanner        testvol
##  Min.   :1.00   Min.   : 1.000
##  1st Qu.:1.00   1st Qu.: 1.000
##  Median :2.00   Median : 3.000
##  Mean   :2.64   Mean   : 7.896
##  3rd Qu.:5.00   3rd Qu.:15.000
##  Max.   :5.00   Max.   :30.000
##  NA's   :240    NA's   :859
```

Noten que las variables `menarche`, `sex` y `tanner` se reportan como numéricas pero son categóricas. Esto se arregla así:

```
detach(juul)
juul$sex <- factor(juul$sex,labels=c("M","F"))
```

```
juul$menarche <- factor(juul$menarche,labels=c("No","Yes"))
juul$tanner <- factor(juul$tanner, labels=c("I","II","III","IV","V"))
attach(juul)
summary(juul)
```

```
##       age         menarche     sex           igf1        tanner
##  Min.   : 0.170   No  :369   M   :621   Min.   : 25.0   I   :515
##  1st Qu.: 9.053   Yes :335   F   :713   1st Qu.:202.2   II  :103
##  Median :12.560   NA's:635   NA's:  5   Median :313.5   III : 72
##  Mean   :15.095                         Mean   :340.2   IV  : 81
##  3rd Qu.:16.855                         3rd Qu.:462.8   V   :328
##  Max.   :83.000                         Max.   :915.0   NA's:240
##  NA's   :5                              NA's   :321
##     testvol
##  Min.   : 1.000
##  1st Qu.: 1.000
##  Median : 3.000
##  Mean   : 7.896
##  3rd Qu.:15.000
##  Max.   :30.000
##  NA's   :859
```

Noten que la forma en que se despliegan las variables categóricas ha cambiado. Cabe mencionarse que la función `summary` trabaja directamente en el conjunto de datos pese a que exista o no, una versión *attached*.

En lo anterior se cambió las variables numéricas a factores. Otra forma de hacer lo mismo es:

```
juul <- transform(juul,
 sex = factor(sex,labels=c("M","F")),
 menarche = factor(menarche,labels=c("No","Yes")),
 tanner = factor(tanner,labels=c("I","II","III","IV","V")))
```

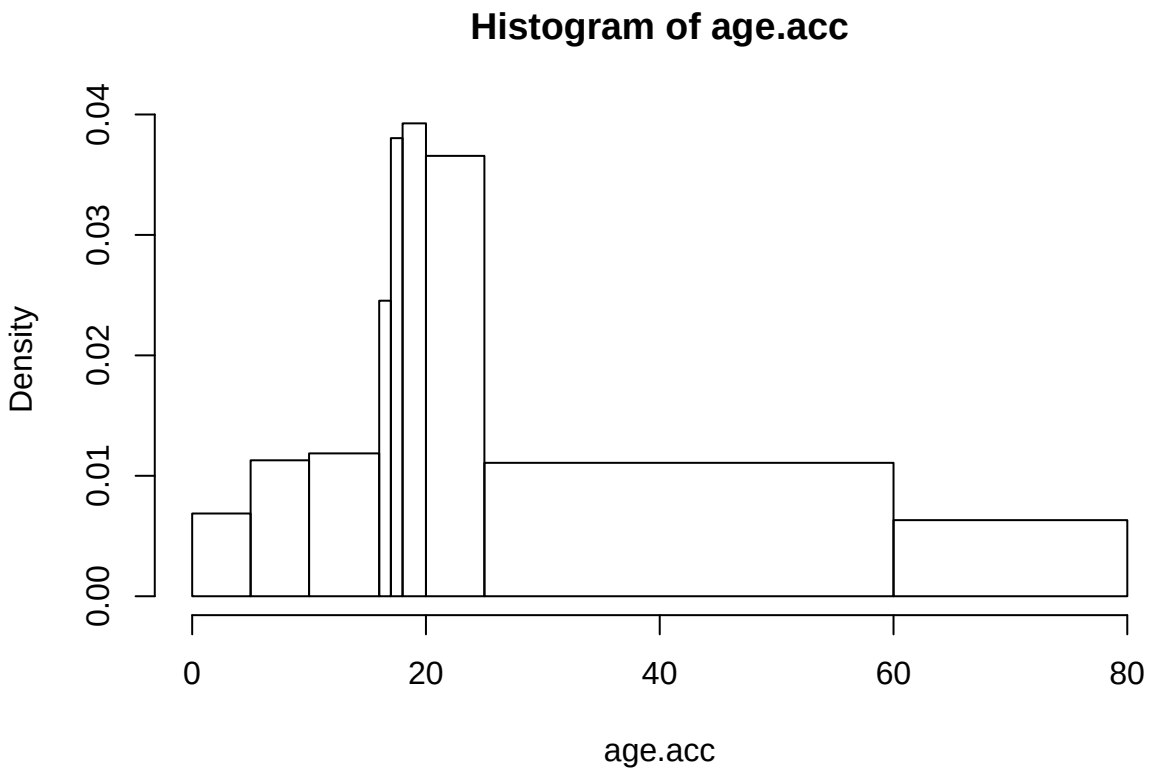
Salidas gráficas de distribuciones

Histogramas

Ya hemos usado la función `hist`, mirémosla con más detalle.

- El histograma sirve para tener una vista razonable de la forma de la distribución de la variable aleatoria. Esto es, el número de observaciones que se tiene en los *bins*.
- Especificando la opción `breaks = n` arroja *aproximadamente* n barras en el histograma.

```
mid.age <- c(2.5,7.5,13,16.5,17.5,19,22.5,44.5,70.5)
acc.count <- c(28,46,58,20,31,64,149,316,103)
age.acc <- rep(mid.age, acc.count)
brk <- c(0,5,10,16,17,18,20,25,60,80)
hist(age.acc, breaks = brk)
```

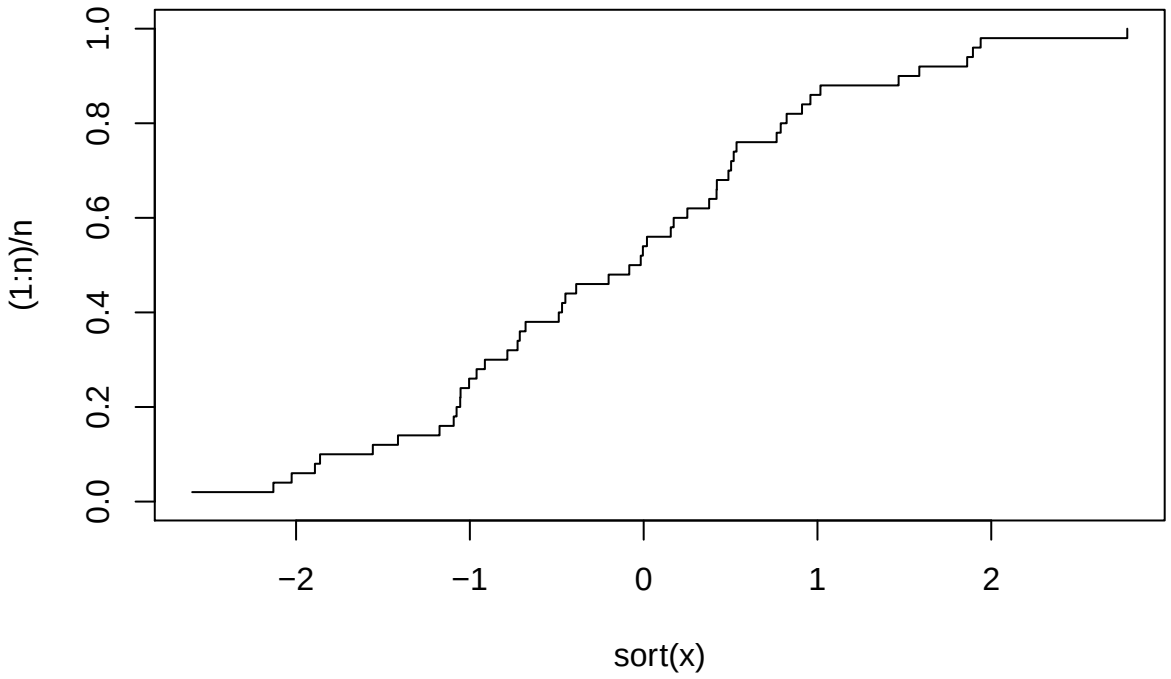


Nota que el eje y está en unidades de densidad, es decir, la proporción de los datos por unidad x tal que el total del área del histograma es 1. Si se desea que el histograma que la altura de las barras se mida en números absolutos, se puede especificar la opción `freq = TRUE`.

Disfribución empírica acumulativa

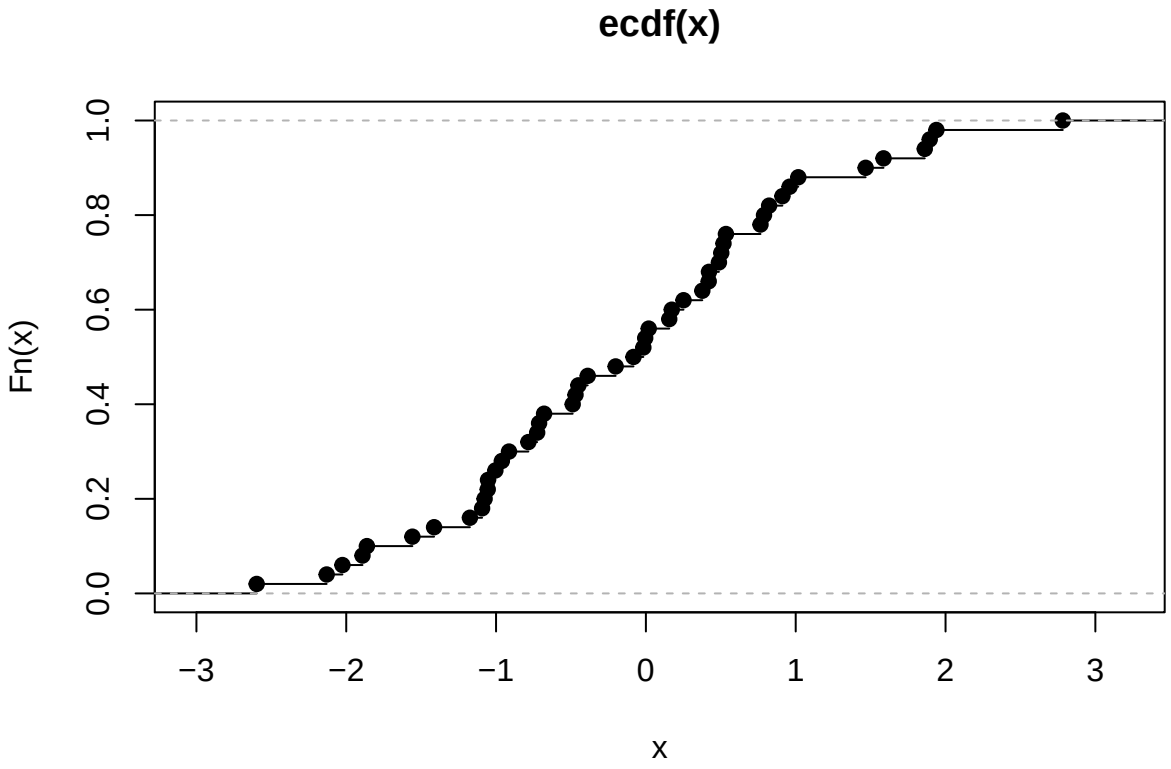
La distribución acumulativa empírica está definida como la fracción de los datos menor o igual que x. Se la puede graficar así:

```
n <- length(x)
plot(sort(x), (1:n)/n, type="s", ylim=c(0,1))
```



También se puede hacer una proximación más exacta con la función `ecdf`

```
plot(ecdf(x))
```



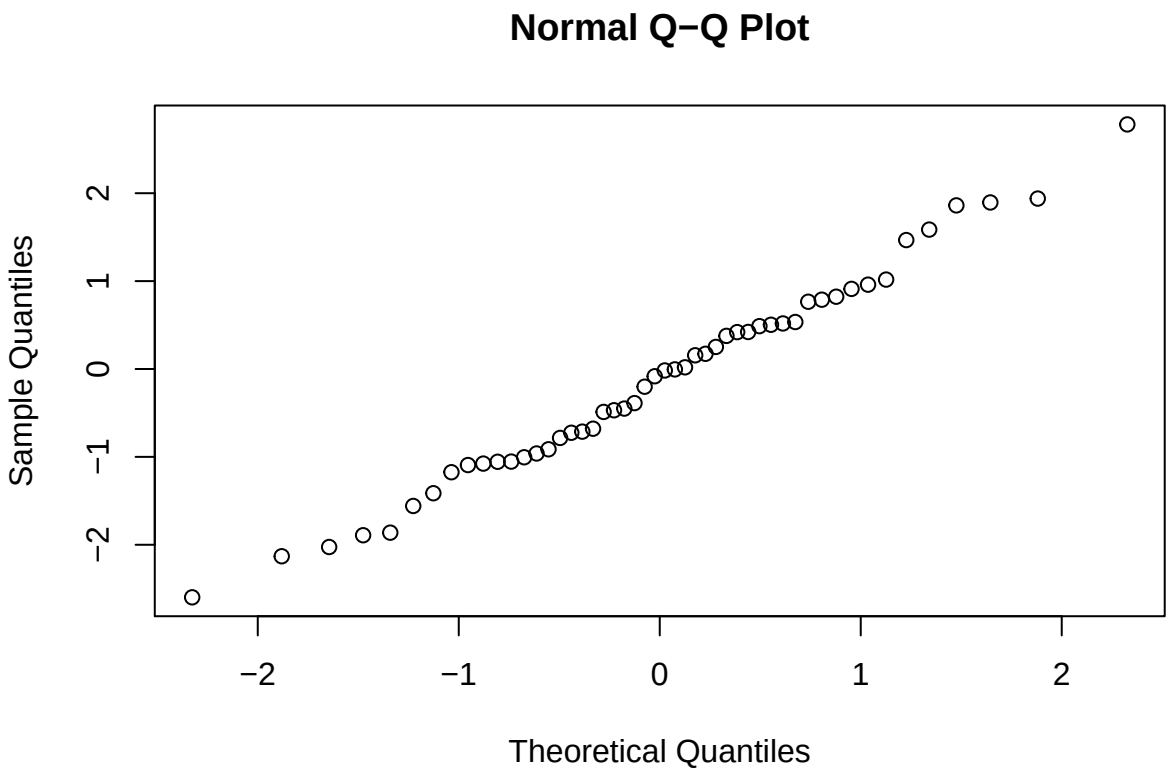
Gráficos QQ

Otra forma de evaluar la procedencia de una distribución de probabilidad es a través de los gráficos Quantil-Quantil. Es decir que se grafican los cuantiles teóricos VS los cuantiles empíricos, se espera entonces que se

produzca una línea con pendiente igual a 1.

En la distribución normal sería:

```
qqnorm(x)
```



Tablas

Los datos categóricos usualmente se presentan en forma de tablas. En esta parte se revisa el cómo crear tablas de los datos y se calcula frecuncias relativas.

Generación de tablas

Veamos unos datos de nivel de consumo de cafeina por estado civil entre mujeres:

```
caff.marital <- matrix(c(652,1537,598,242,36,46,38,21,218
                         ,327,106,67),nrow=3,byrow=T)
colnames(caff.marital) <- c("0","1-150","151-300",">300")
rownames(caff.marital) <- c("Casada","Prev. casada","Soltera")
caff.marital
```

```
##                0 1-150 151-300 >300
## Casada       652  1537     598  242
## Prev. casada  36    46      38   21
## Soltera      218   327     106   67
```

Un objeto `tabla` no es *exactamente* lo mismo que una matriz. Las tablas tienen atributos especiales, pero podemos convertir una matriz en tabla usando `as.table(caff.marital)`. Esto crea un objeto `tabla`.

Para casos elementales, en principio, no existe distinción entre tablas y matrices de 2×2 por ejemplo. Un caso importante en el que se necesita `as.tables` es:

```r
as.data.frame(as.table(caff.marital))
```

```
##            Var1    Var2 Freq
## 1        Casada       0  652
## 2  Prev. casada       0   36
## 3       Soltera       0  218
## 4        Casada   1-150 1537
## 5  Prev. casada   1-150   46
## 6       Soltera   1-150  327
## 7        Casada 151-300  598
## 8  Prev. casada 151-300   38
## 9       Soltera 151-300  106
## 10       Casada    >300  242
## 11 Prev. casada    >300   21
## 12      Soltera    >300   67
```

Como cualquier matriz, una tabla puede ser transpuesta:

```r
t(caff.marital)
```

```
##         Casada Prev. casada Soltera
## 0          652           36     218
## 1-150     1537           46     327
## 151-300    598           38     106
## >300       242           21      67
```

Veamos los tados de `juul` y generamos tablas:

```r
data(juul)
attach(juul)
table(sex)
```

```
## sex
##   1   2
## 621 713
```

```r
table(sex,menarche) # Sexo y menarquia
```

```
##    menarche
## sex   1   2
##   1   0   0
##   2 369 335
```

```r
table(menarche,tanner) # Menarquía y etapa de la pubertad
```

```
##         tanner
## menarche   1   2   3   4   5
##        1 221  43  32  14   2
##        2   1   1   5  26 202
```

La función `xtable` es muy similar a `table`, pero su argumento es al estilo `formula`:

```r
xtabs(~ tanner + sex , data=juul)
```

```
##       sex
## tanner   1   2
##      1 291 224
##      2  55  48
##      3  34  38
```

```
##      4  41  40
##      5 124 204
```

Una tabla múltiple sería:

```
xtabs(~ dgn + diab + coma, data=stroke)
```

```
## , , coma = No
##
##      diab
## dgn    No Yes
##   ICH  53   6
##   ID  143  21
##   INF 411  64
##   SAH  38   0
##
## , , coma = Yes
##
##      diab
## dgn    No Yes
##   ICH  19   1
##   ID   23   3
##   INF  23   2
##   SAH   9   0
```

Si se desea una salida más legible, se puede usar:

```
ftable(coma + diab ~ dgn, data=stroke)
```

```
##     coma  No     Yes
##     diab  No Yes  No Yes
## dgn
## ICH       53   6  19   1
## ID       143  21  23   3
## INF      411  64  23   2
## SAH       38   0   9   0
```

Tablas marginales y frecuencia relativa.

Generamos la tabla de la que queremos ver los valores marginales:

```
tanner.sex <- table(tanner,sex)
```

Se calcula ahora los valores:

```
margin.table(tanner.sex,1)
```

```
## tanner
##   1   2   3   4   5
## 515 103  72  81 328
```

```
margin.table(tanner.sex,2)
```

```
## sex
##   1   2
## 545 554
```

Las freciencias relativas se calculan:

```
prop.table(tanner.sex,1)
```

```
##       sex
## tanner         1         2
##      1 0.5650485 0.4349515
##      2 0.5339806 0.4660194
##      3 0.4722222 0.5277778
##      4 0.5061728 0.4938272
##      5 0.3780488 0.6219512
```

`prop.table` no se puede utilizar para expresar los número en relación a un gran total, pero se puede usar:

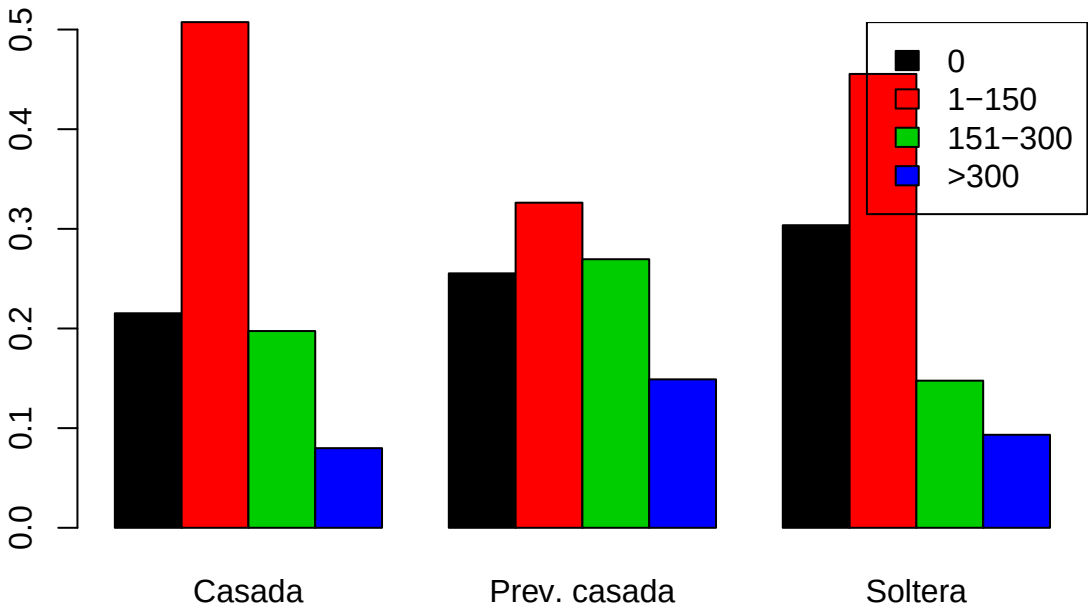
```
tanner.sex/sum(tanner.sex)
```

```
##       sex
## tanner          1          2
##      1 0.26478617 0.20382166
##      2 0.05004550 0.04367607
##      3 0.03093722 0.03457689
##      4 0.03730664 0.03639672
##      5 0.11282985 0.18562329
```

Las funciones `margin.table` y `prop.table` también funcionan sobre tablas múltiples.

Una forma gráfica de expresar estos resultados marginales sería:

```
barplot(prop.table(t(caff.marital),2),beside=T,col = 1:4)
legend("topright",legend = colnames(caff.marital),fill = 1:4)
```



Distribuciones de probabilidad

Cuatro ítems fundamentales pueden ser calculados para una distribución de probabilidad:

- Densidad o probabilidad puntual.
- Probabilidad acumulada, función de distribución.
- Quantiles.

- Pseudo número aleatorios.

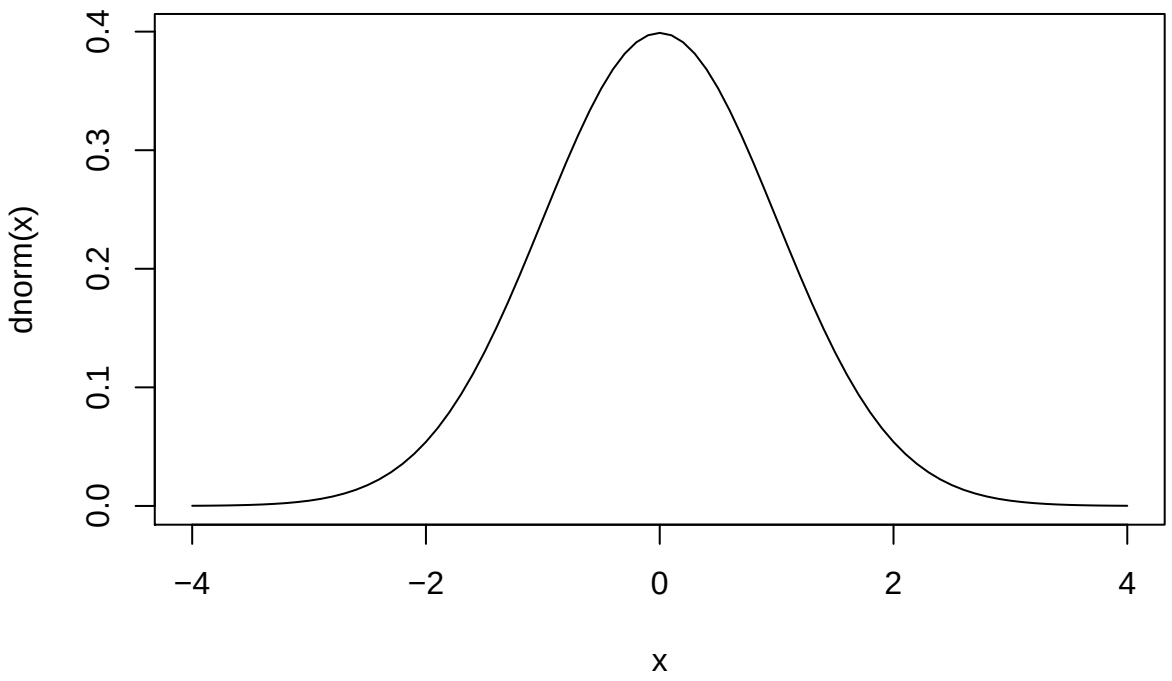
Densidades

La densidad de una distribución continua es una medida de la probabilidad relativa de *obtener un valor cercano a* x. La probabilidad de obtener un valor en un intervalo particular es el área correspondiente debajo de la curva.

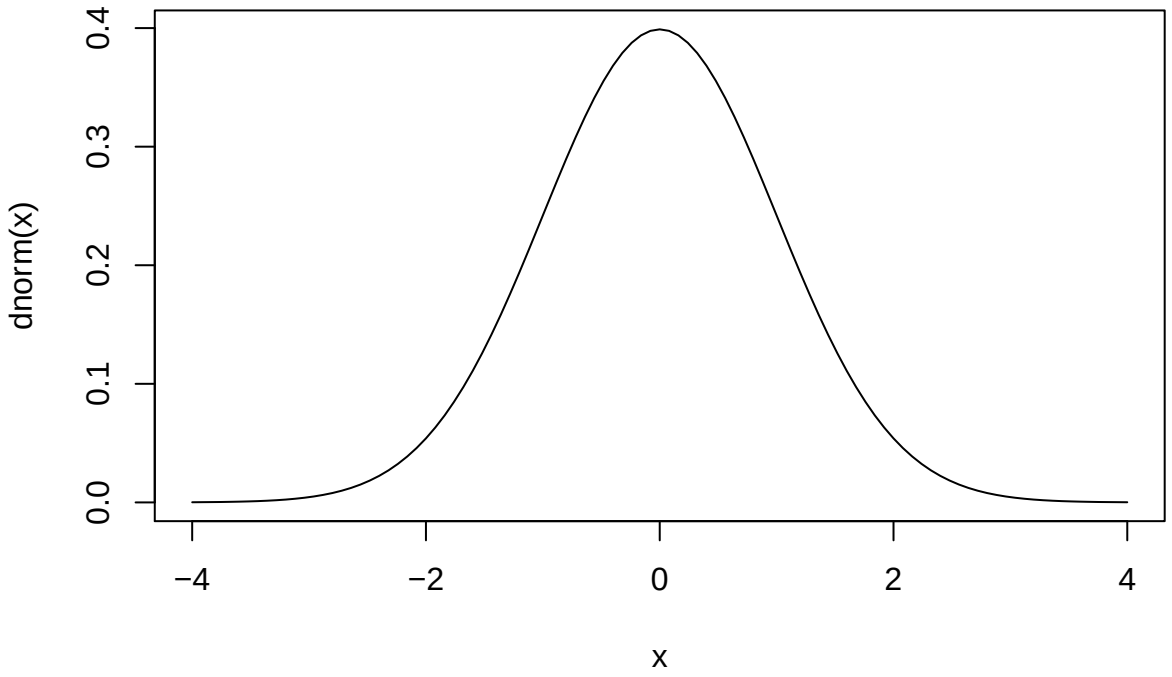
Para distribuciones discretas, el término *densidad* se usa para probabilidad puntual, la probabilidad de obtener exactamente el valor x.

Un ejemplo:

```
x <- seq(-4,4,0.1)
plot(x,dnorm(x),type="l")
```

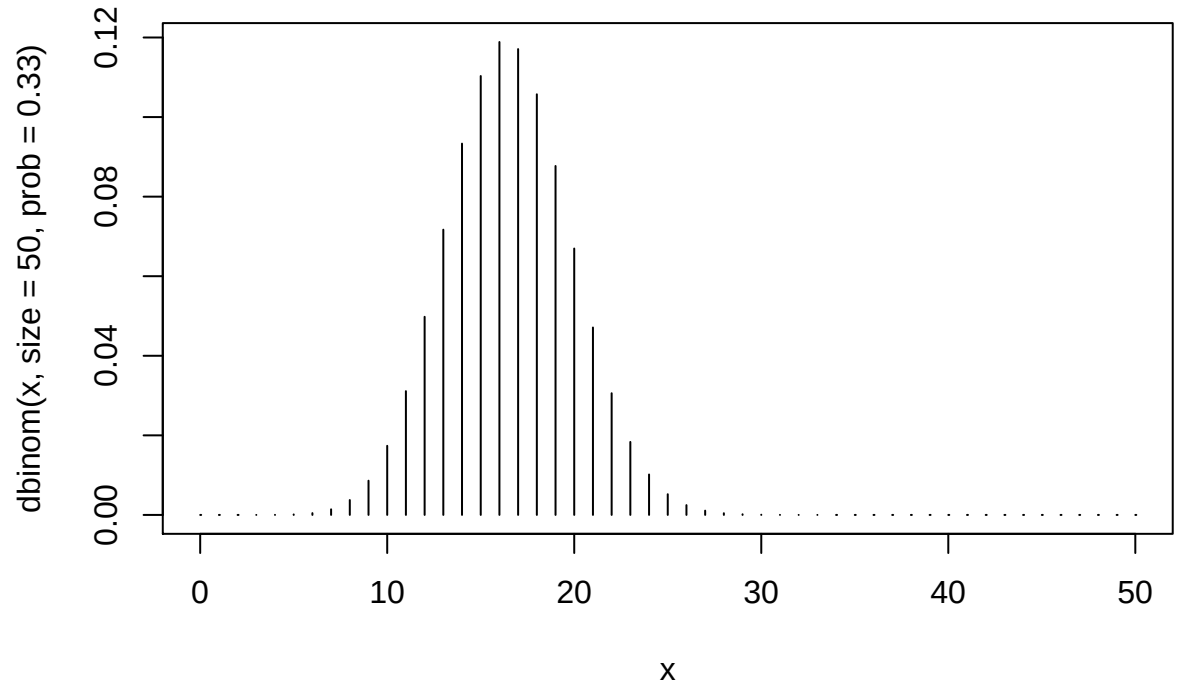



```
curve(dnorm(x), from=-4, to=4)
```



Para distribuciones discretas es mejor este tipo de gráficos:

```
x <- 0:50
plot(x,dbinom(x,size=50,prob=.33),type="h")
```



Distribuciones acumuladas

Esta función describe la probabilidad de `acertarle` a x o menor a x en una distribución dada.

Digamos por ejemplo que la salud de los individuos se describe adecuadamente por una distribución normal con media 132 y desviación estándar de 13. Entonces si un individuo tiene 160, hay

```r
1-pnorm(160,mean=132,sd=13)
```

```
## [1] 0.01562612
```

un 1.5% de la población que tiene ese valor o más.

Quantiles

Esta función es la inversa de la distribución acumulada. El cuantil p es el valor con la propiedad de que hay una probabilidad p de obtener un valor menor o igual que él.

Los cuantiles teóricos se usan comúnmente para el cálculo de intervalos de confianza. Veamos un ejemplo:

Si tenemos n datos distribuidos normalmente con la misma media μ y desviación estándar σ, entonces el promedio $\bar{x}$ se distribuye de forma normal centrado en μ con desviación estándar $\sigma\sqrt{n}$. Un intervalo del 95% del confianza para μ se puede obtener así:

$$\bar{x} + \frac{\sigma}{\sqrt{n}} \times N_{0.025} \leq \bar{x} + \frac{\sigma}{\sqrt{n}} \times N_{0.975}$$

Donde $N_{0.025}$ es el cuantil 2.5% en la distribución normal.

Si $\sigma = 12$, $n = 5$ y encontramos un promedio de $\bar{x} = 83$, entonces:

```r
xbar <- 83
sigma <- 12
n <- 5
sem <- sigma/sqrt(n)
sem
```

```
## [1] 5.366563
```

```r
xbar + sem * qnorm(0.025)
```

```
## [1] 72.48173
```

```r
xbar + sem * qnorm(0.975)
```

```
## [1] 93.51827
```

entonces el intervalo de confianza al 95% para μ está entre 72.48 a 93.52.

Test sobre una y dos muestras

Se introducen dos funciones: `t.test` y `wilcox.test` para el test t y el test de *Wilcoxon* respectivamente. Ambos pueden ser usados para una muestra o dos muestras así como para datos pareados. Note que el test de Wilcoxon para dos muestras es lo mismo que el test de *Mann–Whitney*.

El test t

Este test se basa en el supuesto de normalidad de los datos. Es decir que los datos $x_1 \ldots, x_n$ se asumen como realizaciones independientes de variables aleatorias con media μ y media σ^2, $N(\mu, \sigma^2)$. Se tiene que la hipótesis nula es que $\mu = \mu_0$.

Se puede estimar los parámetros μ y σ por la media $\bar{x}$ y la desviación estándar σ, aunque recuerde que solo son estimaciones del valor real.

Veamos un ejemplo del consuo diario de calorías de 11 mujeres:

```
daily.intake <- c(5260,5470,5640,6180,6390,6515,
                  6805,7515,7515,8230,8770)
```

Veamos algunas estadísticas de resumen:

```
mean(daily.intake)
```

```
## [1] 6753.636
```

```
sd(daily.intake)
```

```
## [1] 1142.123
```

```
quantile(daily.intake)
```

```
##   0%  25%  50%  75% 100%
## 5260 5910 6515 7515 8770
```

Se podría querer saber si el consumo de energía de las mujeres se desvía de una valor recomendado de 7725. Asumiendo que los datos vienen de una distribución normal, el objetivo es hacer una prueba para saber si la media de la distribución es $\mu = 7725$.

```
t.test(daily.intake,mu=7725)
```

```
##
##  One Sample t-test
##
## data:  daily.intake
## t = -2.8208, df = 10, p-value = 0.01814
## alternative hypothesis: true mean is not equal to 7725
## 95 percent confidence interval:
##  5986.348 7520.925
## sample estimates:
## mean of x
##  6753.636
```

Analicemos el resultado como Jack el Destripador (por partes).

- `One Sample t-test`: Muestra el tipo de test.
- `data: daily.intake`: Indica los datos usados para el test
- `t = -2.8208, df = 10, p-value = 0.01814`: Aquí se empieza a poner interesante. Arroja el valor del estadístico t, los grados de libertad y el valor p. Como tenemos el valor p, no es necesario ir a la tabla de valores de la distribución t. Con un nivel de significancia de 5%, en este caso se rechaza la hipótesis nula.
- `alternative hypothesis: true mean is not equal to 7725`. Esto contiene dos pedazos de información importante:
  - El valor puntual sobre el que realizamos el test.
  - Que el test es de dos colas (`not equal to`).
- `95 percent confidence interval`: es el intervalo de confianza de la media *verdadera*.
- `sample estimates`: es la estimación puntual de la media *verdadera*.

Prueba de los rangos con signo de Wilcoxon

Si se desea evitar el supuesto de normalidad de los datos, los *métodos de distribución libre* son una alternativa. Estos generalmente se obtienen reemplazando los datos con estadísticos de orden.

```
wilcox.test(daily.intake, mu=7725)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  daily.intake
## V = 8, p-value = 0.0293
## alternative hypothesis: true location is not equal to 7725
```

Se ve que no se tiene tantos resultados como en el test t. Esto es porque no se tiene la estimación de un parámetro por lo que no hay intervalos de confianza, etc. Para efectos prácticos, cuando se trata de una muestra, el test t y el de Wilcoxon suelen arrijar resultados muy similares.

Test t para dos muestras

Se usa esta prueba con la hipótesis nula de que dos muestras provengan de distribuciones normales con la misma media.

Se puede tener dos enfoques, que las muestras tengan la misma varianza (enfoque clásico) o difieran en varianza.

```
data("energy")
attach(energy)
head(energy)
```

```
##   expend stature
## 1   9.21   obese
## 2   7.53    lean
## 3   7.48    lean
## 4   8.08    lean
## 5   8.09    lean
## 6  10.15    lean
```

```
t.test(expend~stature)
```

```
##
##  Welch Two Sample t-test
##
## data:  expend by stature
## t = -3.8555, df = 15.919, p-value = 0.001411
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.459167 -1.004081
## sample estimates:
##  mean in group lean mean in group obese
##            8.066154           10.297778
```

El intervalo de confianza es para las diferencias entre las medias (note que no contiene a 0). Por defecto se calcula el test asumiendo que se tiene varianzas diferentes en las muestras. Si se desea especificar que las varianzas con iguales se tiene:

```r
t.test(expend~stature, var.equal=T)
```

```
##
##  Two Sample t-test
##
## data:  expend by stature
## t = -3.9456, df = 20, p-value = 0.000799
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.411451 -1.051796
## sample estimates:
##  mean in group lean mean in group obese
##            8.066154           10.297778
```

Note que ahora los grados de libertad ahora son $13 + 9 - 2 = 20$

Comparación de varianzas

Aún cuando en R se puede hacer la prueba sobre dos muestras sin el supuesto de igualdad en las varianzas, podrías estar interesado en hacer una prueba exclusiva de este supuesto.

```r
var.test(expend~stature)
```

```
##
##  F test to compare two variances
##
## data:  expend by stature
## F = 0.78445, num df = 12, denom df = 8, p-value = 0.6797
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.1867876 2.7547991
## sample estimates:
## ratio of variances
##           0.784446
```

Note que este test asume que los grupos son independientes, no se debe aplicar el test cuando los datos son dependientes.

Test de Wilcoxon para dos muestras.

```r
wilcox.test(expend~stature)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  expend by stature
## W = 12, p-value = 0.002122
## alternative hypothesis: true location shift is not equal to 0
```

El test t para muestras pareadas

Este test se usa cuando dos medicioes han sido tomadas en la misma unidad experimental. La teoría escencialmente se basa en tomar las diferencias y el problema se reduce a la prueba t en una muestra.

```r
data(intake)
attach(intake)
t.test(pre, post, paired=T)
```

```
##
##  Paired t-test
##
## data:  pre and post
## t = 11.941, df = 10, p-value = 3.059e-07
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1074.072 1566.838
## sample estimates:
## mean of the differences
##                1320.455
```

Recueda que la clave está en especificar `paired=TRUE`. Un ejemplo de lo que NO debes hacer es:

```r
t.test(pre, post) # MAL!!!
```

```
##
##  Welch Two Sample t-test
##
## data:  pre and post
## t = 2.6242, df = 19.92, p-value = 0.01629
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   270.5633 2370.3458
## sample estimates:
## mean of x mean of y
##  6753.636  5433.182
```

Test de Wilcoxon para muestras pareadas

```r
wilcox.test(pre, post, paired=T)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  pre and post
## V = 66, p-value = 0.00384
## alternative hypothesis: true location shift is not equal to 0
```

Correlación

Se aborda a continuación medidas de correlación paramétricas y no paramétricas. El coeficiente de correlación es una medida de asociación que varía entre -1 y 1.

Correlación de Pearson

El coeficiente de correlación empírico es:

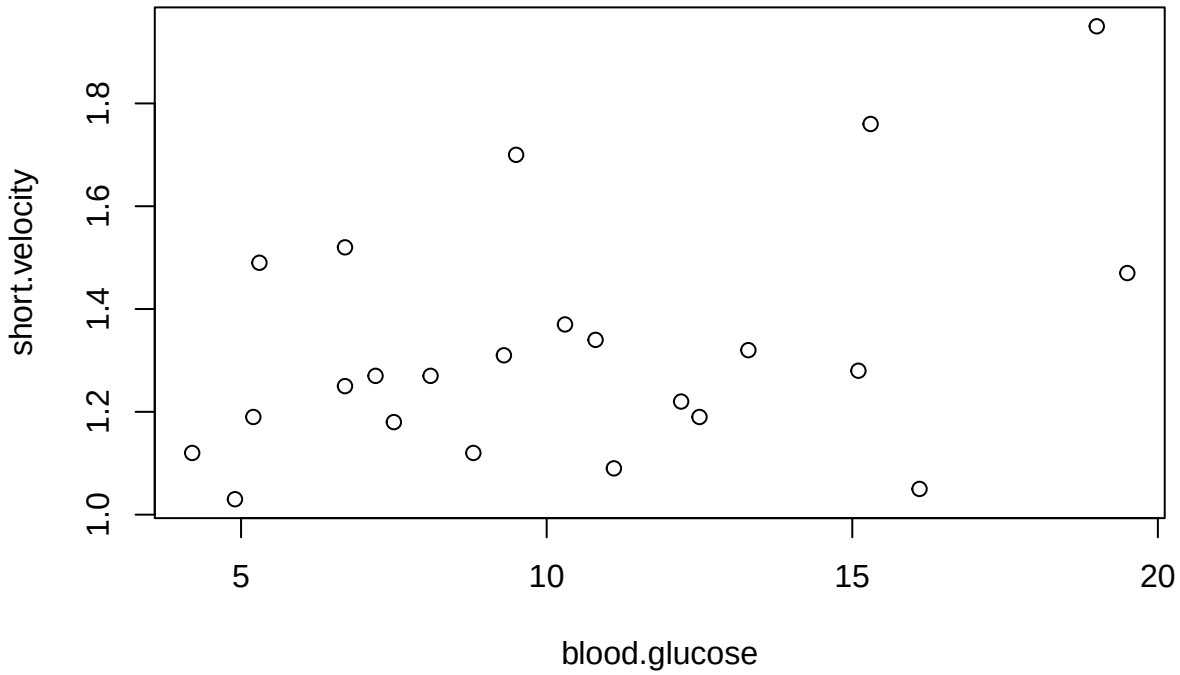
$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

La función `cor` en `R` calcula la correlación entre dos o más vectores.

```
data(thuesen)
attach(thuesen)
cor(blood.glucose,short.velocity)
```

```
## [1] NA
```

```
plot(blood.glucose,short.velocity)
```



Las funciones elementales en `R` requieren que el usuario especifique la acción a tomar con los `NA`. En el caso de la correlación:

```
cor(blood.glucose,short.velocity,use="complete.obs")
```

```
## [1] 0.4167546
```

Se pude obtener todas las correlaciones de un data frame:

```
cor(thuesen,use="complete.obs")
```

```
##                blood.glucose short.velocity
## blood.glucose      1.0000000      0.4167546
## short.velocity     0.4167546      1.0000000
```

Sin embargo, los cálculos que hemos hecho, no nos indican si la correlación es significativamente diferente de cero. Para ello hacemos:

```
cor.test(blood.glucose,short.velocity)
```

```
##
##  Pearson's product-moment correlation
##
```

```
## data:  blood.glucose and short.velocity
## t = 2.101, df = 21, p-value = 0.0479
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.005496682 0.707429479
## sample estimates:
##       cor
## 0.4167546
```

Correlación de Spearman ρ

Esta se obtiene al reemplazar las observaciones por su rango y luego se calcula la correlación. La hipótesis nula es la independencia entre las variables.

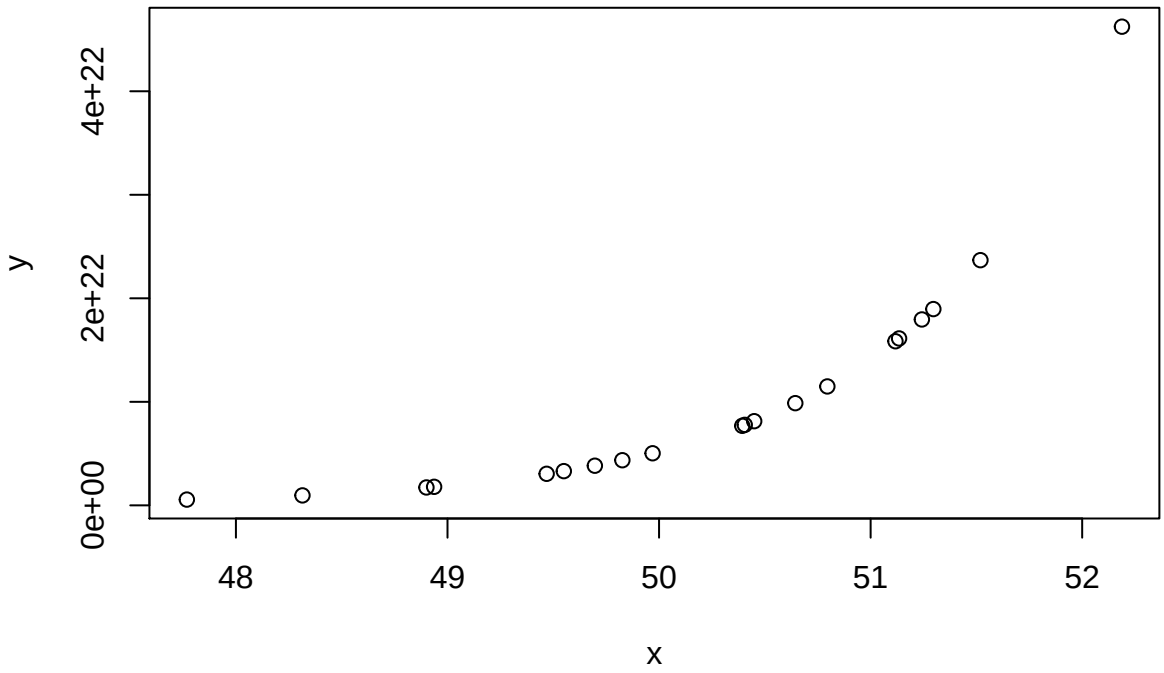
```
cor.test(blood.glucose,short.velocity,method="spearman")
```

```
##
##  Spearman's rank correlation rho
##
## data:  blood.glucose and short.velocity
## S = 1380.4, p-value = 0.1392
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##      rho
## 0.318002
```

Ejemplo: r vs ρ

Veamos un ejemplo donde $y = exp(x)$:

```
x <- rnorm(20,50)
x <- sort(x)
y <- exp(x)
plot(x,y)
```




```
cor(x,y,method="pearson")
```

```
## [1] 0.8385442
```

```
cor(x,y,method="spearman")
```

```
## [1] 1
```

Ahora $y = x$

```
x <- rnorm(20)
x <- sort(x)
y <- x
cor(x,y,method="pearson")
```

```
## [1] 1
```

```
cor(x,y,method="spearman")
```

```
## [1] 1
```

Interpretación de la correlación:

- La correlación esta siempre entre -1 y 1. Lo primero que se interpreta es el signo
- Directamente proporcional si es positivo, si es negativo pasa lo contrario
- En segundo lugar se interpreta es la fuerza de la relación. Si esta más cerca de 1, significa que si aumenta una variable, la otra también.
- Números intermedios, reducen la fuerza de la relación.

Regresión lineal, primeras ideas

Funciones genéricas

Las funciones que usen posteriormente para extraer los resultados de un análisis actuarán específicamente con respecto a la clase del objeto. Estas funciones se denominan genéricas.

Función	Descripción
`print`	devuelve un corto resumen
`summary`	devuelve un resumen detallado
`df.residual`	devuelve el número de grados de libertad
`coef`	devuelve los coeficientes estimados (algunas veces con sus errores estándar)
`residuals`	devuelve los residuales (residuos)
`fitted`	devuelve los valores ajustados (estimados)

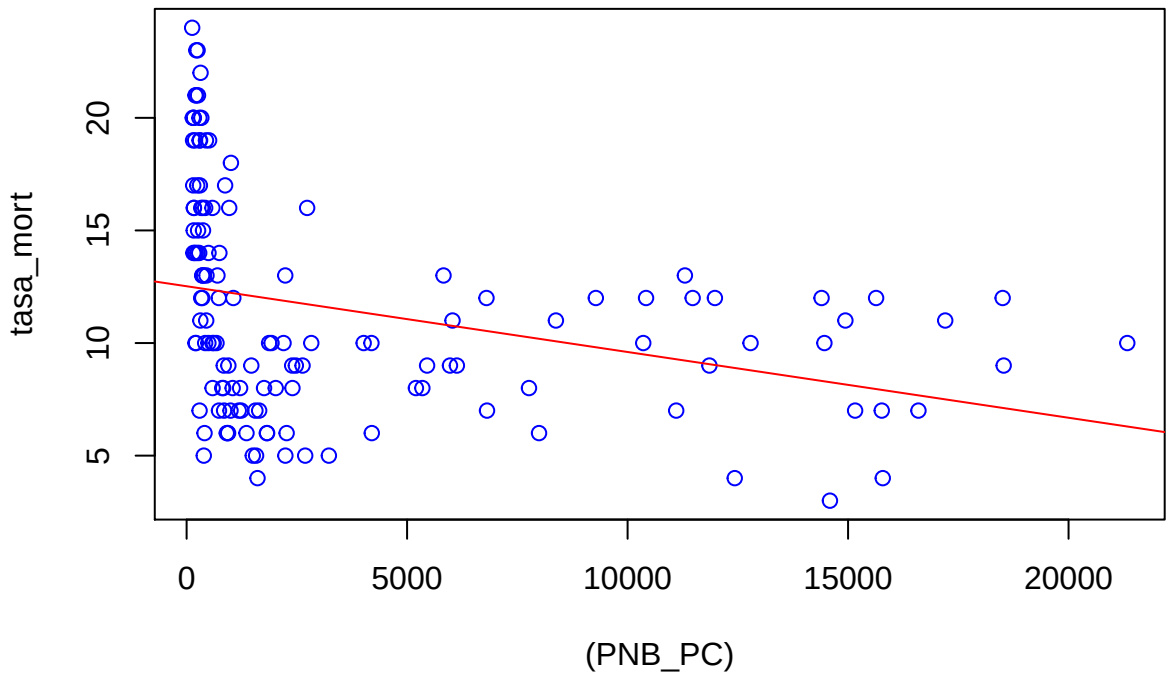
Abrir la base de datos `Mundo.csv` (recuerden usar attach)

```
mundo <- read.csv("Mundo.csv", sep=";", header=TRUE)
attach(mundo)
reg1 <- lm(tasa_mort~PNB_PC)
df.residual(reg1)
```

```
## [1] 137
```

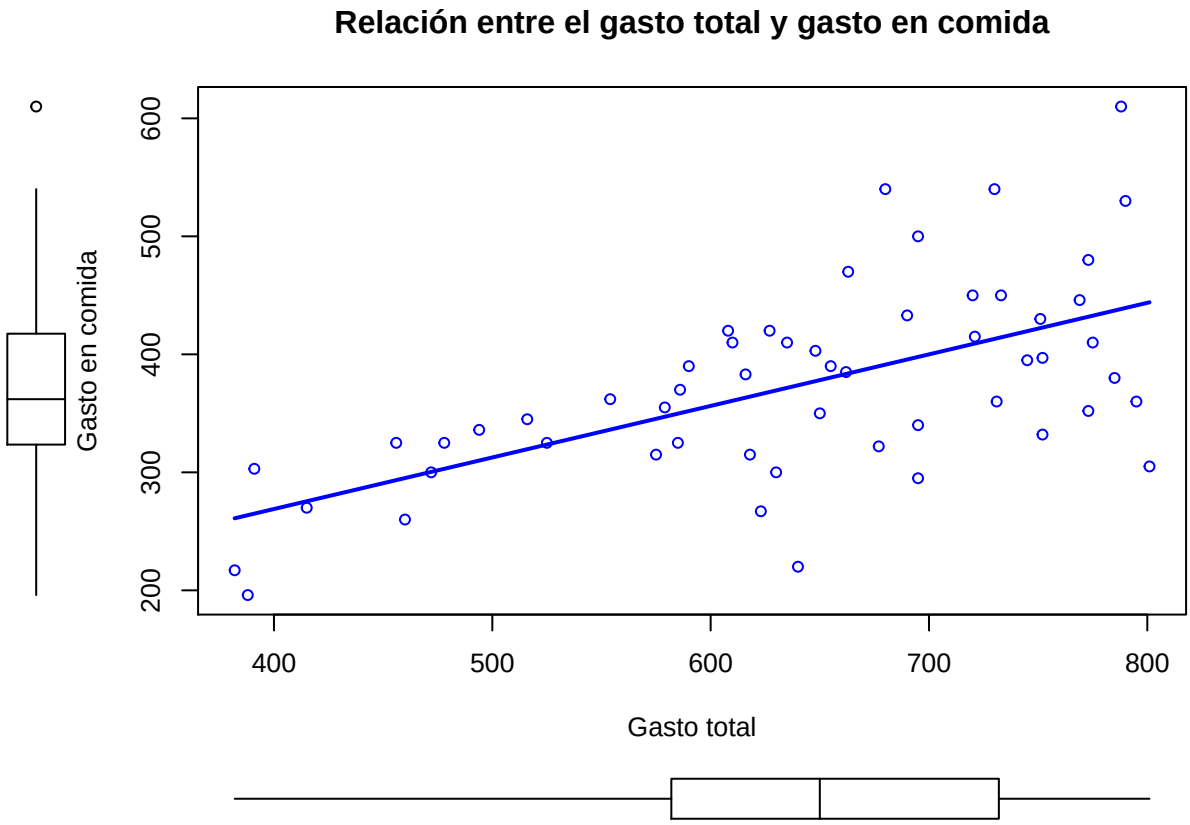
Una exploración gráfica:

```
plot((PNB_PC),tasa_mort,col="blue")
abline(coef(reg1),col="red")
```



Veamos un `scatterplot` más informativo a partir de la tabla 2.8:

```
library(car)
datos <- read.csv("table2_8.csv", sep=";", header=TRUE)
attach(datos)
scatterplot(TOTALEXP, FOODEXP, smooth=F, main="Relación entre el gasto total y gasto en comida",xlab="Ga
```



La normalización de datos se lo puede realizar de varias formas, podemos entrar a la ayuda de esta función para más detalle. La estandarización más usada es n1, es decir: (observación-media)/desviación estándar

```
# install.packages("clusterSim")
library(clusterSim)
normfood <- data.Normalization (FOODEXP,type="n1")
normtot <- data.Normalization (TOTALEXP,type="n1")
```

¿Qué ganamos normalizando los datos?

```
mean(normtot)
```

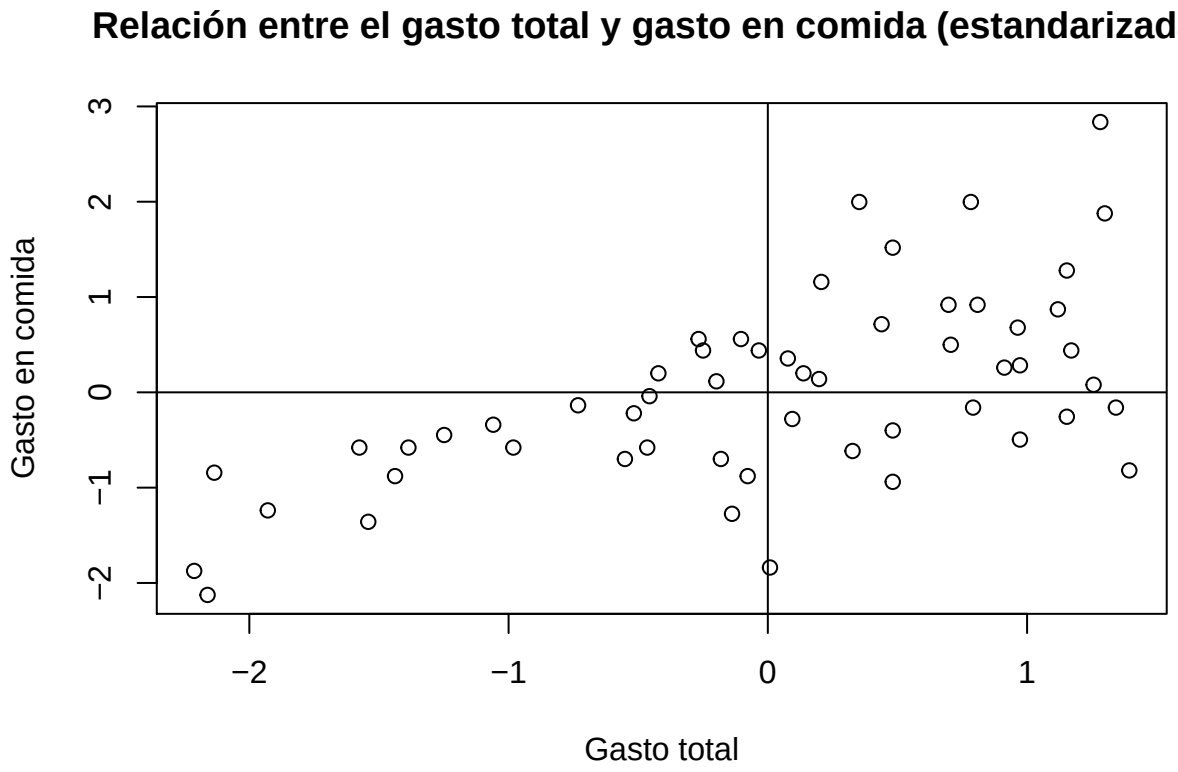
```
## [1] 2.814234e-16
```

```
sd(normtot)
```

```
## [1] 1
```

Veamos los resultados gráficamente:

```
plot(normtot, normfood, main="Relación entre el gasto total y gasto en comida (estandarizadas)",xlab="Ga
abline(v=0)
abline(h=0)
```



Interpretación

- Cuadrante 1: gasto total y en comida en comida mayor que el promedio
- Cuadrante 2: gasto total inferior al promedio, pero gastan más que el promedio en comida
- Cuadrante 3: tienen menos gasto total que el promedio y el gasto en comida es menor al promedio
- Cuadrante 4: tiene un gasto total mayor el promedio pero su gasto en comida es menor al promedio

Veamos algunas particularidades de las funciones estadísticas básicas:

```
# Variables originales
var(TOTALEXP,FOODEXP) #varianza
```

```
## [1] 5893.876
```

```
cor(TOTALEXP,FOODEXP) #correlacion
```

```
## [1] 0.6081313
```

```
# Variables normalizadas
var(normtot,normfood) #varianza
```

```
## [1] 0.6081313
```

```
cor(normtot,normfood) #correlacion
```

```
## [1] 0.6081313
```

Ahora veamos:

```
totgasto=TOTALEXP/40
foodgasto=FOODEXP/40
cor(totgasto,foodgasto)
```

```
## [1] 0.6081313
```

Conclusión:La correlación no depende de las unidades en las que estámos midiendo, ¿La varianza?

Una última conclusión:

```
gastocom=totgasto+4
mean(gastocom)
```

```
## [1] 19.97591
```

```
mean(totgasto)
```

```
## [1] 15.97591
```

```
cor(gastocom,foodgasto)
```

```
## [1] 0.6081313
```

Conclusión: La correlación no depende ni de la escala ni del origen