

# Introducción a R

**Genomeeting 2016**

# Outline

## Módulo 1. Fundamentos de R

- ▶ Introducción a R
- ▶ R Studio
- ▶ ¿Cómo funciona R?
- ▶ Manejo de objetos
- ▶ Factores
- ▶ Creación de gráficos
- ▶ Exportación e importación de datos
- ▶ Manipulación de matrices, data frames y listas

# Outline

## Módulo 1. Fundamentos de R

### - Introducción a R

- ▶ R Studio
- ▶ ¿Cómo funciona R?
- ▶ Manejo de objetos
- ▶ Factores
- ▶ Creación de gráficos
- ▶ Exportación e importación de datos
- ▶ Manipulación de matrices, data frames y listas

# Introducción a R

R es un software totalmente gratuito y disponible para distintos sistemas operativos (Unix, Windows y MACOS).

<http://cran.r-project.org/> linked phrase



```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
```

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

```
Natural language support but running in an English locale
```

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

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

```
During startup - Warning messages:
1: Setting LC_CTYPE failed, using "C"
2: Setting LC_COLLATE failed, using "C"
3: Setting LC_TIME failed, using "C"
4: Setting LC_MESSAGES failed, using "C"
5: Setting LC_MONETARY failed, using "C"
[R.app GUI 1.66 (6996) x86_64-apple-darwin13.4.0]
```

```
WARNING: You're using a non-UTF8 locale, therefore only ASCII characters will work.
Please read R for Mac OS X FAQ (see Help) section 9 and adjust your system preferences accordingly.
[History restored from /Users/lilyj/.Rapp.history]
```

```
>
```

```
lilyj@lilyj-SX
```

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
```

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

```
Natural language support but running in an English locale
```

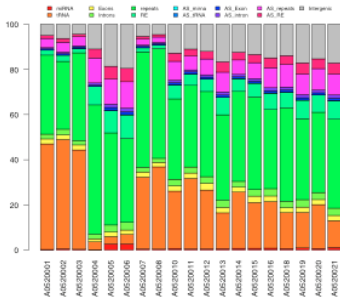
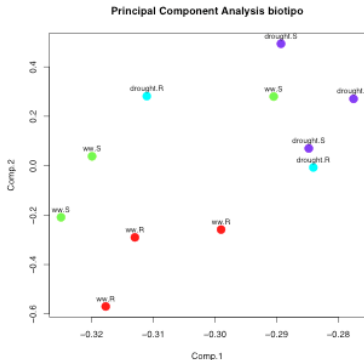
```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

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



# Introducción a R

- ▶ R es un lenguaje el cual tiene un ambiente estadístico y gráfico
- ▶ R implementa un lenguaje que fue desarrollado en los laboratorios Bell por John Chambers et al., 1998.
- ▶ Para algunos usuarios es simplemente un programa estadístico de fácil acceso para generar una variedad de operaciones y gráficas.



# Introducción a R

- ▶ Para usuarios más avanzados ofrece un lenguaje de programación que añade funcionalidad definiendo funciones para generar resolver un sin fin de problemas.
- ▶ La comunidad de R es muy dinámica (ej., crecimiento en número de paquetes), integrada por estadísticos de gran renombre (ej., J. Chambers, L. Terney, B. Ripley, D. Bates, etc).
- ▶ Extensiones específicas a nuevas áreas (bioinformática, geoestadística, modelos gráficos).

# Outline

## Módulo 1. Fundamentos de R

- ▶ Introducción a R

### - R Studio

- ▶ ¿Cómo funciona R?
- ▶ Manejo de objetos
- ▶ Factores
- ▶ Creación de gráficos
- ▶ Exportación e importación de datos
- ▶ Manipulación de matrices, data frames y listas

# ¿Cómo funciona R?

¡Manos a la obra, abre R Studio!





# RStudio

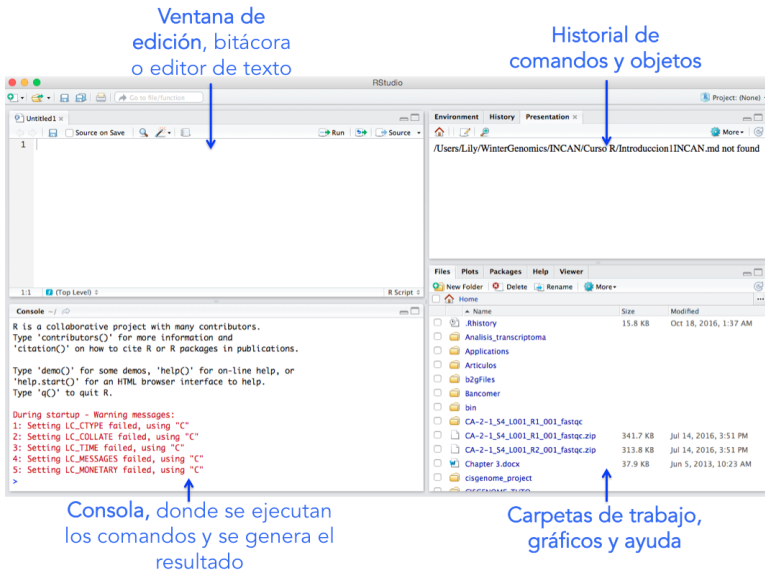


Figure 3: alt text

# RStudio

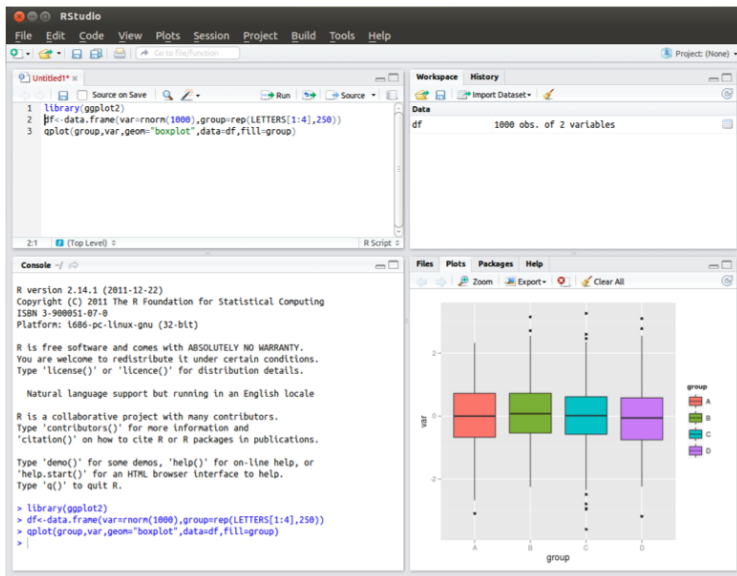


Figure 4: alt text

# Outline

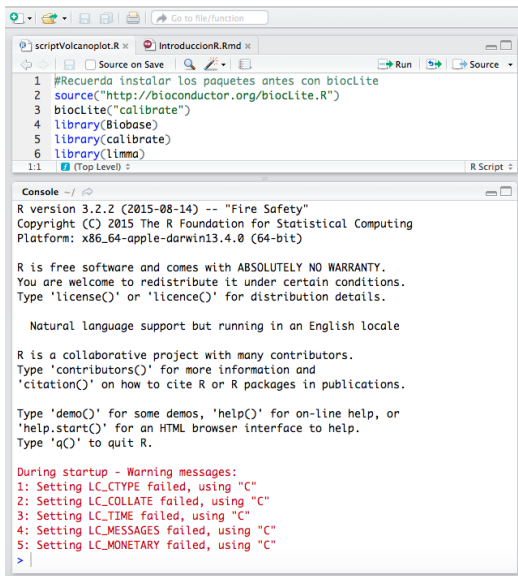
## Módulo 1. Fundamentos de R

- ▶ Introducción a R
- ▶ R Studio

### - **¿Cómo funciona R?**

- ▶ Manejo de objetos
- ▶ Factores
- ▶ Creación de gráficos
- ▶ Exportación e importación de datos
- ▶ Manipulación de matrices, data frames y listas

# ¿Cómo funciona R?



The screenshot shows the RStudio interface. The top pane displays a script named 'IntroduccionR.Rmd' with the following R code:

```
1 #Recuerda instalar los paquetes antes con biocLite
2 source("http://bioconductor.org/biocLite.R")
3 bioclite("calibrate")
4 library(Biobase)
5 library(calibrate)
6 library(limma)
```

The bottom pane shows the R console output:

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

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

Natural language support but running in an English locale

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

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

During startup - Warning messages:
1: Setting LC_CTYPE failed, using "C"
2: Setting LC_COLLATE failed, using "C"
3: Setting LC_TIME failed, using "C"
4: Setting LC_MESSAGES failed, using "C"
5: Setting LC_MONETARY failed, using "C"
> |
```

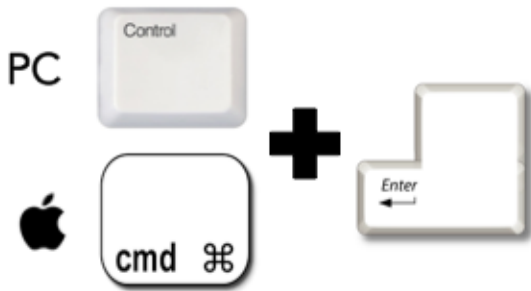
Figure 5: alt text

# ¿Cómo funciona R?

- ▶ La consola de R es el área en la que se ejecuta código
- ▶ Indica con `>` que está listo para aceptar comandos.
- ▶ Permite recuperar comandos antiguos con flechas arriba y abajo.
- ▶ El área de código es donde se edita y almacena código. Esto puede servir para un trabajo más sistematizado y ordenado, se pueden definir instrucciones en un fichero (script) y ejecutarlos secuencialmente.
- ▶ Además los scripts tienen la ventaja de que funciona como una bitácora.

# ¿Cómo funciona R?

- Escribir (y grabar) en área de código y enviar a consola



## ¿Cómo funciona R?

- ▶ Permite completar comandos con TAB



Ejemplo, escribe esto en el área de código y ejecutalo para que lo mande a la consola:

```
sum(78+1246)
```

# ¿Cómo funciona R?

## Función

En el mundo de R las **instrucciones** se dan mediante funciones, las cuales se aplican a lo que se encuentra dentro del parentesis.

## Funcion()

Ejemplo de funciones para movernos en el escritorio

- ▶ Sirve para saber en que directorio estás **getwd()**
- ▶ Lista los archivos que están en el directorio donde te encuentras **list.files()**
- ▶ Sirve para cambiarte de directorio **setwd("Entre comillas pega o usa el tabulador para moverte en tu dir")**



# ¿Cómo funciona R?

También podemos usar una manera más amigable

**Para moverse entre carpetas**

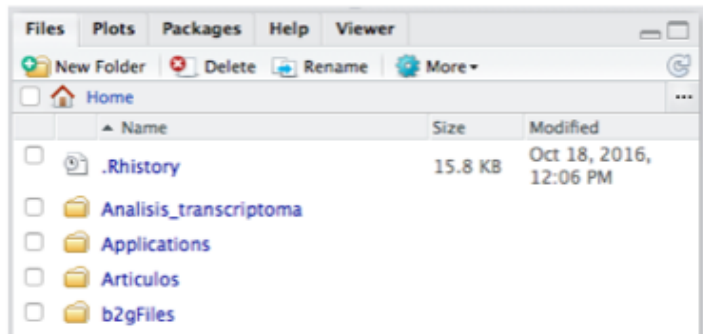


Figure 6: alt text

# ¿Cómo funciona R?

## Pidiendo ayuda a R

Si no saben el nombre de una función, la pueden buscar con “?”

Ejemplo

**?mean**

**help(mean)**

# ¿Cómo funciona R?

## Editando nuestros scripts

R ejecuta las funciones línea por línea interactivamente, un ejemplo es lo siguiente:

Escriban la siguiente instrucción:

```
sum(18+25)
```

```
## [1] 43
```

# ¿Cómo funciona R?

## Comentarios

En R podemos poner comentarios que nos funcionen a manera de descripción de funciones o bitácora, esto se hace usando el signo “#” que indica que después de este signo R no ejecuta esa línea.

Ejemplo:

```
#En este script vamos a hacer un comentario
```

Pueden ver que pasa si ejecutan una funcion enseguida del signo “#”

Escriban esto en el editor

```
sum(3+3) # Esto es una simple suma
```

# ¿Cómo funciona R?

## Espacios

```
1 + 2 # + 20
```

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

Ten cuidado, valores faltantes pueden romper el código

```
1 + # 2
```

# ¿Cómo funciona R?

## Espacios

Por lo general a R no le importan los espacios

```
1 + 2
```

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

sigue produciendo el mismo resultado

```
1      +      2
```

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

# ¿Cómo funciona R?

R puede **imprimir** textos a pantalla

La acción de imprimir se puede realizar usando la función **print()**. Esta función pide a R que imprima resultados en la consola.

Esto puede ser útil cuando queremos conocer el valor de un objeto.

Ejemplo:

```
print (1)
```

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

# ¿Cómo funciona R?

R puede **imprimir** textos a pantalla

```
print ("Necesitamos café por favor!" )
```

```
## [1] "Necesitamos caf<U+00E9> por favor!"
```

NOTA: Hay que evitar acentos



## ¿Cómo funciona R?

```
print("Tienes X genes diferencialmente expresados")  
print("Tambien imprimo signos (&%?$)")
```

# ¿Cómo funciona R?

## Cuestiones básicas de R

¿Cómo pedir ayuda en R? Con `help.start` las paginas de ayuda se abren en su navegador

**`help.start()`**

**`?(q)`**

Buscando funciones `help` es para cuando conocen el nombre exacto de la función. Si intuyen el nombre de la función, podemos buscar funciones usando `?`:

**`?quit`**

También puedes usar la función **`apropos()`** la cual regresará una lista de funciones relacionadas con la función que buscas.

O simplemente googlear la función, hay un sin fin de blogs para aclarar tus dudas.

# Outline

## Módulo 1. Fundamentos de R

- ▶ Introducción a R
- ▶ R Studio
- ▶ ¿Cómo funciona R??

## Manejo de objetos

- ▶ Factores
- ▶ Creación de gráficos
- ▶ Exportación e importación de datos
- ▶ Manipulación de matrices, data frames y listas

# Manejo de objetos

## Objetos

**Todo lo que manipula R es un objeto.**

Las entidades que crea y manipula R se llaman objetos. Hay varias clases de objetos como veremos a continuación:

**Vectores:** Son secuencias unidimensionales de elementos del mismo tipo.

```
edades <- c(12,11,15,13,16,12,11,8)
```

**Factores:** Son vectores de tipo categoricos, pueden contener datos numéricos, integrales o caracteres.

# Manejo de objetos

## Objetos

**Funciones:** Son instrucciones que vienen determinadas en R, las cuales suelen formar un nuevo objeto mediante el resultado de estas instrucciones. Todas las funciones están compuestas por parentesis después de la instrucción.

- `print()` – prints objects
- `log()` – computes logarithms
- `exp()` – computes the exponential function
- `sqrt()` – takes the square root
- `abs()` – returns the absolute value
- `sin()` – returns the sine
- `cos()` – returns the cosine
- `tan()` – returns the tangent
- `asin()` – returns the arc-sine

# Manejo de objetos

## Objetos

**Matrices:** Son objetos de dos dimensiones. Pero el contenido de este objeto debe de ser de un solo tipo de datos.

**Listas:** Las listas son parecidas a los vectores, pero a diferencia de estas, no tienen que contener el mismo tipo de datos.

Ejemplo: el primer vector puede contener 26 letras del alfabeto y el segundo elemento puede contener un vector numerico de los primeros numeros del 1 al 1000. Incluso el tercer elemento puede contener una matriz.

**Data frame:** Es una matriz pero puede contener datos de diferentes tipos, ya sea numericos o tipo caracter.

# Manejo de objetos

## Asignacion de valores

En R podemos asignar valores a objetos con el siguiente signo:

Para la asignación `<-` usar

```
valor1 <- 10
```

```
valor2 <- 2 * 3
```

¿Cómo le podemos hacer para ver el contenido de estos dos últimos objetos?

¿Cuales podrían ser la ventajas?

# Manejo de objetos

## Asignacion de valores

Ahora realiza el siguiente ejercicio:

Asigna el resultado de multiplicar 2 por 8 a la variable oreos



# Manejo de objetos

## Asignacion de valores

Resultado

```
oreos <- 2*8
```

```
oreos
```

```
## [1] 16
```

Ahora asigna el valor 5 a la variable suavicremas

Suma las variables y guardalas en la variable mis\_galletas

# Manejo de objetos

## Resultado

```
suavicremas <- 5
```

```
mis_galletas <- oreos + suavicremas
```

```
mis_galletas <- sum(oreos, suavicremas)
```

# Manejo de objetos

## Tipos de datos en R

En R así como hay diferentes tipos de objetos, tenemos varios tipos de datos, veamoslo con un ejercicio:

Ejercicio:

```
#Agrega un valor numerico a la variable mi_numero
```

```
mi_numero <- 30
```

```
#Agrega la siguiente palabra a mi_caracter: Mexico
```

```
mi_caracter <- "Mexico"
```

NOTA: Cuando usamos texto debemos escribirlo con " "

```
#Agrega el valor logico
```

```
mi_logico <- TRUE
```

# Manejo de objetos

## Tipos de datos en R

### Tipos de datos

```
mi_numero <- 30
```

```
mi_caracter <- "Mexico"
```

```
mi_logico <- TRUE
```

# Podemos checar la clase a la que corresponde cada uno de nuestros datos con la función **class()**

# Manejo de objetos

## Tipos de datos en R

```
class(mi_numero)
```

```
## [1] "numeric"
```

```
class(mi_caracter)
```

```
## [1] "character"
```

```
class(mi_logico)
```

```
## [1] "logical"
```

# Manejo de objetos

## Vectores en R

Los vectores son cadenas unidimensionales (es decir una sola columna o fila) de un tipo de valores (numericos, caracteres, etc.)

```
longitud <- c(12,11,15,13,16,12,11)
colores <- c("Negro", "Rosa", "Amarillo", "Blanco",
             "Azul", "Marron", "Guinda")
```

# Manejo de objetos

Hacer una operación para varios números a la vez usando la función combine “c()”

```
PesosDePacientes <- c(55, 45, 85, 55, 63, 78, 57)
```

Para ver los valores, tecleamos **PesosDePacientes** en la consola

```
PesosDePacientes + 1
```

```
## [1] 56 46 86 56 64 79 58
```

# Manejo de objetos

Usemos diferentes funciones para aplicarlas al valor de peso

```
sum(PesosDePacientes)
```

```
## [1] 438
```

```
mean(PesosDePacientes)
```

```
## [1] 62.57143
```

```
max(PesosDePacientes)
```

```
## [1] 85
```

```
min(PesosDePacientes)
```

```
## [1] 45
```



# Manejo de objetos

```
sort(PesosDePacientes)
```

```
## [1] 45 55 55 57 63 78 85
```

```
unique(PesosDePacientes)
```

```
## [1] 55 45 85 63 78 57
```

# Manejo de objetos

En R también podemos tener valores de texto, los cuales siempre van entre comillas. Así como guardamos una serie de números en el objeto **PesosDePacientes** podemos guardar valores de texto en un nuevo objeto de la siguiente manera:

```
nombres <- c("Susana", "Angela", "Oscar", "Joel",  
             "Blanca", "Karla", "Manuel")
```

# Manejo de objetos

El objeto **nombres** nos puede servir para etiquetar los valores numéricos de los pesos de los pacientes, esto con la función `names()` para asignar nombres de texto.

```
names(PesosDePacientes) <- nombres
```

Veamos el contenido de **PesosDePacientes**

# Manejo de objetos

## Vectores en R

Un ejemplo de un vector es nuestro objeto **PesosDePacientes**, ya que tiene una serie de valores. En dado caso que solo queramos acceder o trabajar con una sola parte del vector, debemos usar corchetes cuadrados:

```
PesosDePacientes[1]
```

```
## Susana
```

```
##      55
```

¿Y Para obtener el peso de “Karla”?

```
PesosDePacientes[ ]
```

## Manejo de objetos

Para extraer de nuestro objeto más de un elemento se usa la función para combinar o concatenar `c()`

```
PesosDePacientes[c(3,6)]
```

```
## Oscar Karla  
##      85      78
```

También podemos usar índices negativos para quitar temporalmente ciertos elementos de un vector:

```
PesosDePacientes[-c(3,6)]
```

```
## Susana Angela      Joel Blanca Manuel  
##      55      45      55      63      57
```

# Manejo de objetos

## Condiciones

En alguna ocasión vamos a querer obtener solo los elementos de un vector que pasen un cierto valor mayor o menor. Por ejemplo si queremos ciertos genes que tengan cierto nivel de expresión. Esto se contesta con operaciones condicionales (" $>$ ", " $<$ ", " $=$ ").

Sigamos usando nuestro objeto `PesosDePacientes`.

¿Que pacientes tienen un peso mayor a 50?

# Manejo de objetos

## Condiciones

```
PesosDePacientes > 50
```

##	Susana	Angela	Oscar	Joel	Blanca	Karla	Manuel
##	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE

# Manejo de objetos

## Condiciones

¿Cómo preguntamos cuales son menores o iguales que?

```
PesosDePacientes == 55
```

```
## Susana Angela Oscar Joel Blanca Karla Manuel  
## TRUE FALSE FALSE TRUE FALSE FALSE FALSE
```

```
PesosDePacientes < 55
```

```
## Susana Angela Oscar Joel Blanca Karla Manuel  
## FALSE TRUE FALSE FALSE FALSE FALSE FALSE
```



# Manejo de objetos

## Otras operaciones

Valores lógicos TRUE/FALSE genera respuesta como si hiciéramos una pregunta a R.

Muy útil cuando queremos saber:

```
sum(PesosDePacientes > 50)
```

```
## [1] 6
```

```
PesosDePacientes[PesosDePacientes > 50]
```

```
## Susana  Oscar    Joel  Blanca  Karla  Manuel  
##      55      85      55      63      78      57
```

# Manejo de objetos

## Ejercicio:

1. ¿Que obtenemos y que hacen las últimas instrucciones?
2. ¿Cómo obtienen aquellos pesos que son mayores al promedio de todas los pesos?

# Outline

## Módulo 1. Fundamentos de R

- ▶ Introducción a R
- ▶ R Studio
- ▶ ¿Cómo funciona R??
- ▶ Manejo de objetos
- ▶ Manipulación de Objetos y vectores

### - Factores

- ▶ Creación de gráficos
- ▶ Exportación e importación de datos
- ▶ Manipulación de matrices, data frames y listas

# Factores

Este tipo de objeto es fundamental para el análisis estadístico ya que es la forma como se tratan las variables categóricas

```
as.factor(PesosDePacientes)
```

```
## Susana Angela Oscar Joel Blanca Karla Manuel  
##      55      45      85      55      63      78      57  
## Levels: 45 55 57 63 78 85
```

edades

# Outline

## Módulo 1. Fundamentos de R

- ▶ Introducción a R
- ▶ R Studio
- ▶ ¿Cómo funciona R??
- ▶ Manejo de objetos
- ▶ Manipulación de Objetos y vectores
- ▶ Factores

### - Creación de gráficos

- ▶ Exportación e importación de datos
- ▶ Manipulación de matrices, data frames y listas

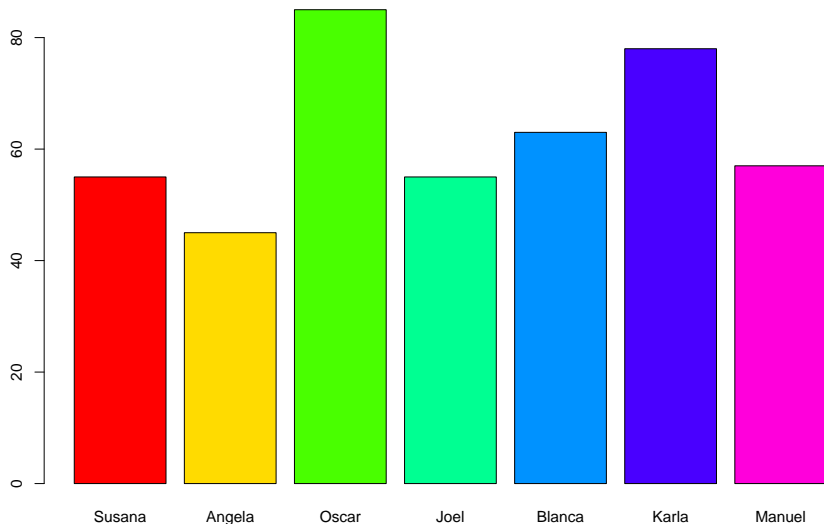
# Creación de gráficos

```
barplot(PesosDePacientes, col = rainbow(7))
```

```
boxplot(PesosDePacientes)
```

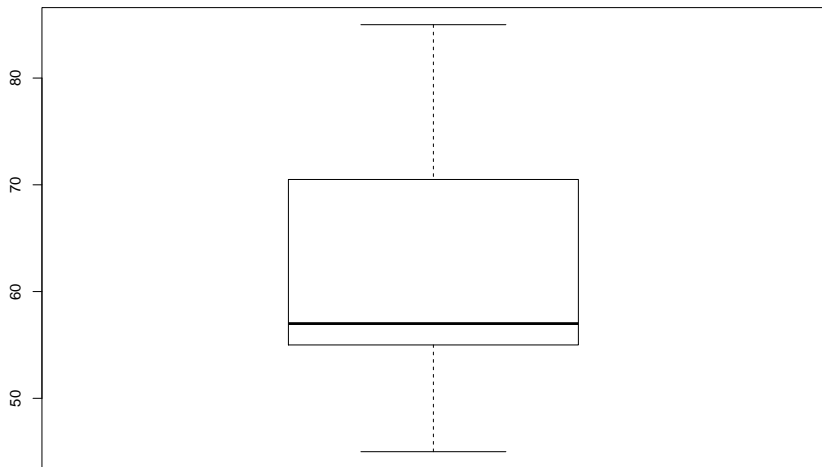
# Creación de gráficos

```
barplot(PesosDePacientes, col = rainbow(7))
```



# Creación de gráficos

```
boxplot(PesosDePacientes)
```





# Creación de gráficos

## Guardar gráficos

### En PDF

```
pdf("Nombre.pdf")  
barplot(PesosDePacientes, col = rainbow(7))  
dev.off()
```

### En PNG

```
png("Nombre.png")  
barplot(PesosDePacientes, col = rainbow(7))  
dev.off()
```

# Outline

## Módulo 1. Fundamentos de R

- ▶ Introducción a R
- ▶ R Studio
- ▶ ¿Cómo funciona R??
- ▶ Manejo de objetos
- ▶ Manipulación de Objetos y vectores
- ▶ Factores
- ▶ Creación de gráficos

### - **Exportación e importación de datos**

- ▶ Manipulación de matrices, data frames y listas

# Exportación e importación de datos

Cargando tablas en R Recuerda las funciones útiles para  **cambiarnos de directorio**.

```
setwd("/home/alumno/genomeeting-2016/count-tables")
```

```
getwd()
```

Leer tablas en R

```
Tabla <- read.table("pdata.txt", header=TRUE,  
as.is=TRUE)
```

Otras funciones

```
read.csv()
```

```
read.delim()
```

# Exportación e importación de datos

Podemos guardar tablas mediante diferentes funciones, estas tablas se van a generar en el directorio donde estemos, recordemos aquí que podemos ejecutar `getwd()` para saber en que carpeta estamos.

```
write.table(Tabla, file="MiTabla.txt", sep="^", row.names=FALSE,  
col.names=TRUE)
```

# Ejercicios

Si quieres seguir realizando ejercicios, puedes acceder al siguiente link. Tutoriales publicos de **O'REILLY**

<http://tryr.codeschool.com/levels/2/challenges/1>