Biometría II - 2021

TP Nº 1

Objetivos

1. Repasar conceptos de biometría 1: estadística descriptiva, modelo lineal simple y sus supuestos.
2. Utilizar R para análisis descriptivos: Estadísticos, Tablas y gráficos 3.
Poder construir gráficos con R e interpretarlos.
4. Analizar relaciones entre variables
5. Implementar modelos lineales en R para variables respuesta normal y variable explicatoria categorica o cuantitativa. Interpretar salidas y resultados en contexto.

Parte A: Estadística descriptiva e Inferencial

· Ejercicios guiados en Script de R (Ejercicio '*Iris' y Ejercicio 'Rendimiento'*) · Shiny para obtención de la muestra ejercicio 'Rendimiento' (1 muestra 'personalizada' por grupo)

· Planilla compartida ('Resultados TP 1 – Rendimiento')
· Cuestionario para resolver en Aula Virtual

Parte B: Análisis exploratorio a partir del manejo de salidas gráficas · Ejercicios en Guía de TP (a continuación) + Script de R (*Con base de datos 'Iris'*)

Análisis exploratorio a partir del manejo de salidas gráficas

Uno de los principales objetivos que tenían los programas antecesores de R era mejorar el análisis exploratorio de los datos a partir de la visualización de los mismos, con la creación y el manejo de salidas gráficas que simplifiquen su acceso e interpretación. Los comandos gráficos en R pueden ser, al principio, engorrosos de utilizar, pero la versatilidad que ofrece para realizar gráficos supera la de la gran mayoría de los *software* estadísticos de uso frecuente. A continuación, vamos a aprender a realizar, manipular y almacenar salidas gráficas de R usando las funciones existentes en el paquete base (*plot*, *barplot*, *hist*, etc), y utilizando el popular paquete para gráficos *ggplot2*.

Hay diferentes formas de crear "dispositivos" donde realizar gráficos (dispositivo llamamos a "algo" donde uno puede hacer que aparezca un gráfico), por ejemplo un archivo pdf, jpg, o bien la pantalla de nuestra computadora. Si queremos que el gráfico aparezca en la pantalla, al ejecutar la función *windows()* (*) nos devolverá una nueva ventana donde se empezará a generar el gráfico que queramos realizar. Por *default*, en *RStudio* los gráficos se visualizan en una ventana propia de la consola.

(*) x11() para Linux y quartz() para Mac

1era Parte: Gráficos básicos

Vamos a seguir trabajando con la base de datos *Iris*. Comience un nuevo *script* cargando la base de datos *Iris.txt* en un objeto *data.frame* denominada *Datos*. Atachee la base de datos.

- Realice los siguientes gráficos, analice qué información le brinda cada uno de ellos. Investigue que significan los parámetros "main", "xlab", "ylab", "pch", "col". ¿Qué función cumplía "as.numeric"?

  plot(LongSepalo, AnchoSepalo, col=especie)

  pairs(Datos[,2:5], pch=as.numeric(especie))

  hist(LongSepalo, ylab="Frecuencia", xlab="Longitud del Sépalo")

  plot(LongSepalo~especie, main="Longitud del sepalo por especie")

barplot(tapply(Datos$LongSepalo,Datos$especie,mean), *main="Longitud del sepalo por especie"*)

- Intente modificar el número de divisiones en el histograma utilizando el parámetro "breaks". Una vez hecho esto, pruebe ingresar: *plot(density(LongSepalo), main="Densidad de LongSepalo")*. ¿A qué tipo de distribución se asemeja la variable LongSepalo?

- Confeccione, en una ventana aparte, un gráfico subdividido en cuatro donde se muestre el histograma de cada una de las variables medidas. Revise el Anexo 1 para poder realizarlo. ¿Qué tipo de simetría presentan las variables? ¿Es razonable suponer que provienen de distribuciones normales?

- Sabiendo que el comando para realizar un gráfico de cajas y bigotes es *boxplot()*, confeccione un gráfico de este tipo para las 4 variables registradas en las 150 flores. Analice qué se está representando e investigue la presencia de datos atípicos en los datos.

- Por último, confeccione un *boxplot* para cada variable pero discriminando por especie. Compare a las especies respecto de las variables medidas en cuanto a tendencia central y variabilidad.

¿Cómo almacenar las salidas gráficas en distintos formatos?

R es capaz de trabajar con diferentes dispositivos gráficos. Uno de ellos ya lo hemos usado, y es el comando windows(). Este comando abre un dispositivo gráfico que permite visualizar en pantalla las salidas que vayamos realizando. Pero windows() no es el único... existen los commandos pdf(), png(), tiff(), bmp(), jpeg(), que permiten almacenar las salidas gráficas en diferentes formatos de archivos, del tamaño y resolución que necesitemos. Por ejemplo, almacenemos en formato *png* y *pdf* la relación entre la longitud del sépalo y el pétalo en Iris:

```
 # png
png("myplot2.png", width= 15, height= 10, units= "cm", res
= 90) plot(LongPetalo,LongSepalo,
```

```
      col=especie,
      ylab="Longitud del Sépalo (cm)",
      xlab="Longitud del Pétalo (cm)")
dev.off()
```

Primero se crea el dispositivo png() con las características deseadas de tamaño, resolución, nombre del archivo, y luego se escribe sobre él el gráfico que se desea almacenar. Como último paso, el dispositivo abierto se cierra con el comando dev.off(), generándose así el archivo correspondiente.

Este mismo formato es válido para genera tiff(), jpeg(), y bmp(). Para generar archivos pdf() los argumentos son ligeramente diferentes:

```
# pdf
pdf("myplot2.pdf", width= 7, height= 8,
paper="special")  # ancho y alto en pulgadas de la
region para graficar
# paper "special", setea el tamaño de la figura al del ancho y
largo del  grafico
# alternativas paper "a4", "letter" etc
plot(LongPetalo,LongSepalo,
 col=especie,
 ylab="Longitud del Sépalo (cm)",
 xlab="Longitud del Pétalo (cm)")
dev.off()
```

¡Desafío! ¿Cómo hacer un gráfico “presentable”?:

En esta parte trabajaremos con algunas de las visualizaciones de datos que hicimos anteriormente, pero intentando emprolijarlas como si tuviésemos que presentar la salida en alguna publicación, informe o TP final de la materia… El desafío que les proponemos es que intenten llegar al gráfico que se presenta a continuación (Figura 1), compuesto de 3 gráficos en una misma figura, a partir de la información que les brindamos en los anexos 1 y 2 de esta guía. ¡Vamos, a trabajar! Luego de generarlo, intente almacenarlo en formato *jpg*.

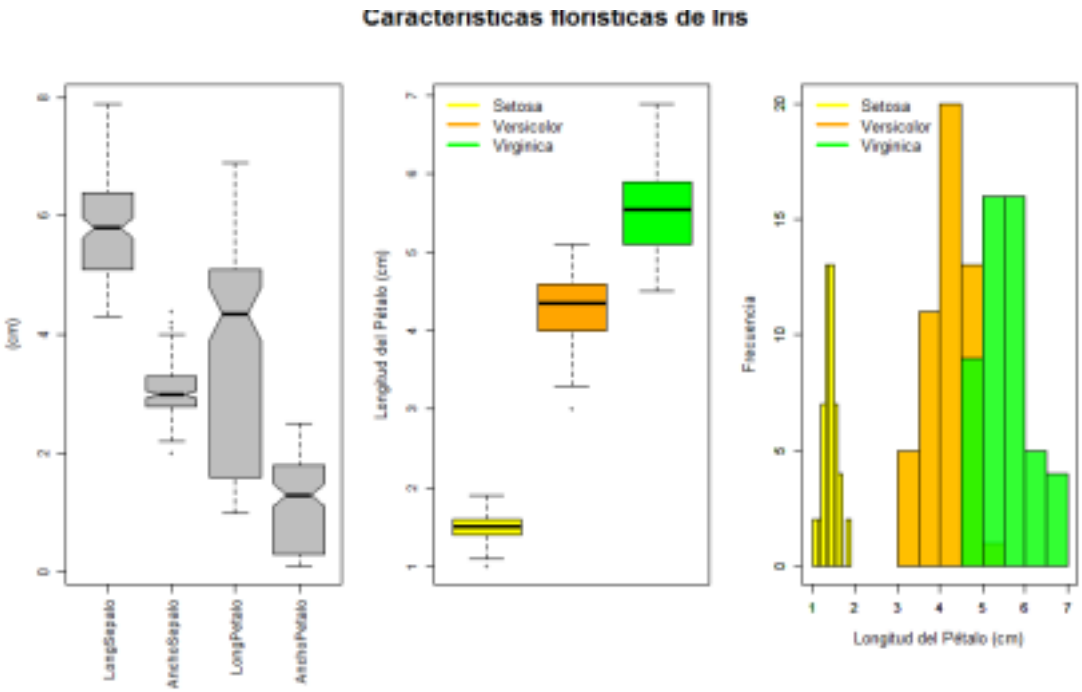


Figura 1. Características florísticas de Iris. Izq: Gráfico de cajas para las variables analizadas. Centro: Gráfico de cajas para la variable Longitud del Pétalo, discriminado por especie. Derecha: Histograma de la variable Longitud del Pétalo discriminado por especie.

2da Parte: Generando salidas gráficas con ggplot2

Con R es posible obtener el mismo resultado yendo por diferentes caminos. En ese sentido, *ggplot2* es un entorno gráfico de R que permite hacer gráficos similares a los anteriores y otros más complejos y "profesionales", trabajando a partir de capas o *layers* que se van sumando para obtener el resultado final (para una mejor explicación de la estructura o sintaxis básica de *ggplot,* ver *Anexo 1*). La idea es introducir la confección de algunos gráficos utilizando este entorno, para que se vayan familiarizando con su uso y puedan extenderlo a otros contextos si así lo desean. El paquete se encuentra disponible en el repositorio CRAN y pueden acceder vía el comando *install.packages()*. Instálenlo, cárguenlo, y comencemos a trabajar!

Tambien vamos a utilizar otros paquetes además del ggplot2 que se indican en el Script.

Vamos a comenzar con una figura donde visualizaremos la relación entre las variables Longitud y Ancho del Sépalo, para las 3 especies por separado (si no lo hicieron antes, revisen el anexo 1 porque lo van a necesitar):

```
AyL_Sepalo <- ggplot(Datos,aes(LongSepalo,AnchoSepalo))
summary(AyL_Sepalo)
AyL_Sepalo
```

En primer lugar creamos un objeto que va a contener toda la información para crear el gráfico. Con

el comando **summary** verificamos la información que contiene el objeto, y luego lo visualizamos. A continuación vamos a ir agregando diferentes aspectos al gráfico para ir construyéndolo paso a paso:

```
AyL_Sepalo<- AyL_Sepalo + theme_classic() + labs(x="Longitud del
 Sepalo  (cm)",y="Ancho del Sepalo (cm)")
AyL_Sepalo
```

El comando **theme_classic** nos permite cambiar el estilo del gráfico, y **labs** editar los nombres de los ejes (compárenlo con el anterior). Existen varios estilos (*themes*) pre-definidos (para ver la lista, ?theme_bw)

Una vez que le indicamos las características generales del gráfico, ejecutemos el siguiente código y veamos las diferencias entre los gráficos y cómo se generan:

1)

```
AyL_Sepalo_n <- AyL_Sepalo + geom_point(size=2)
```

¿Qué hace el commando **geom_point**? ¿Y el argumento **size**? ¿Cómo haría para discriminar los datos por especie?

2)

```
AyL_Sepalo_col <- AyL_Sepalo + geom_point(size=2,
aes(color=especie)) +  theme(legend.title = element_blank())

AyL_Sepalo_col + scale_color_manual(values=c("red","green","orange"))
```

Acá les damos una opción para discriminar los datos por especie (¿cuál?). ¿Cómo haría para modificar el tipo de símbolo por especie? Inténtelo.

3)

```
AyL_Sepalo_grid <- AyL_Sepalo +
 facet_grid(~ especie, scales="free_x") +
 geom_point()

AyL_Sepalo_grid <- AyL_Sepalo_grid +
theme(strip.background =
element_rect(fill="lightgreen"))

AyL_Sepalo_grid <- AyL_Sepalo_grid + geom_smooth(method =
"lm",se=F,  color="red")
```

¿Cuál es la utilidad del comando **facet_grid**? ¿Y de **geom_smooth**?

Por último, guardemos el último gráfico en un archivo png

```
# generar archivo
```

```
ggsave("sepalo.png", AyL_Sepalo_grid, width=10, height=5)
```

- Continúe explorando los distintos gráficos que se muestran en el Script (boxplot, barplot, hist)

- Interprete cada uno de los comandos en las siguientes líneas de código

```
ggplot(Datos, aes(x=LongPetalo, y=..density..,
fill=especie)) +  labs(x="Longitud del Petalo
(cm)",y="densidad (u.a.)")+
 geom_histogram(colour="grey60", size=0.2, binwidth = 0.15,
alpha=.65) +  geom_density(alpha=.6,colour=NA)+
 theme_classic() + theme(legend.title = element_blank()) +
 scale_fill_manual(values=c("red","green","orange"))
```

  - ¿Qué función cumplen *geom_histogram()*, *geom_density()*, *alpha*, y *fill=especie*? ○ ¿Cómo haría para modificar el ancho de las barras del histograma? ¿Y para cambiar de posición la leyenda? Ubíquela en el costado superior derecho (ayuda: lea el *help* del comando *theme*).

- *Gráficos de barras o líneas con errores.* Hacer un gráfico de barras o líneas utilizando el paquete básico de R que incluya algún tipo estimación de la variabilidad no es sencillo. En *ggplot2* existe una geometría que permite incluir las barras de error de manera muy sencilla, habiendo previamente obtenido, por ej., el desvío estándar de cada variable. Explore el libro incluido en las referencias de Winston Chang (2012, *Cookbook for R* – Receta 7.7) y realice un gráfico de barras incluyendo el desvío estándar para la variable Longitud del Pétalo discriminando por la especie de Iris.

BonusTrack

ggplot permite hacer cosas *lindas*, fíjense si no este gráfico de correlación para las variables incluidas en el set de datos de Iris. Ejecuten el código y analicen el resultado visualizado en el gráfico. library(ggplot2)

```
library(ggcorrplot)
# Correlation matrix
corr_datos <- round(cor(Datos[,2:5]), 1)
# Plot
ggcorrplot(corr_datos, type = "lower", method="circle",
colors = c("tomato2", "white", "springgreen3"),
outline.color="black", ggtheme=theme_bw)
```

Explore e interprete más opciones de gráficos de correlación que se ofrecen en el Script

Y esto recién empieza… hay todo un mundo por explorar

Top-50 de visualizaciones con ggplot2

http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R Code.html

ANEXO 1: ¿Cómo se estructura una salida gráfica en R?

Área de dibujo, márgenes internos (mar), márgenes externos

(adaptado de E. F. Glynn, http://research.stowers.org/mcm/efg/R/Graphics/Basics/mar-oma/index.htm)

En términos generales, la ventana de dibujo en R puede dividirse en tres partes: un área de dibujo, un margen interno y un margen externo.

La función *par* permite ajustar el tamaño de los márgenes (abajo, izquierda, arriba, derecho), a partir del ajuste de los parámetros *mar* (*margin size*) y *oma (outer margin área)*. Por default, los valores son mar=(c(5.1, 4.1, 4.1, 2.1) y oma=c(0,0,0,0). ¿Qué significan estos valores? El margen interno inferior es de aprox. 5 líneas (unidades arbitrarias de R, 1 línea = 0.5 cm aprox.), el margen izquierdo y superior aprox. 4 líneas cada uno, y el margen derecho de aprox. 2 líneas. Estos tamaños reflejan las necesidades de las etiquetas (*labels)* de la mayoría de los gráficos. Los gráficos en R por *default* no tienen margen externo, a menos que uno explícitamente modifique sus valores. En el ejemplo de la figura, estos márgenes fueron ajustados de la siguiente manera:

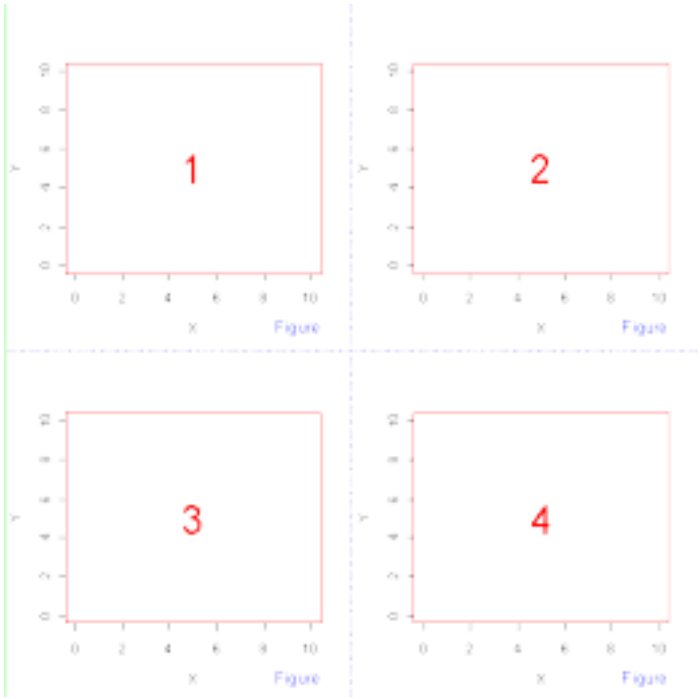
```
par(mar=c(5.1, 4.1, 4.1, 2.1))
par(oma=c(2, 2, 2, 2))
```

Figuras múltiples

Para generar distintos sub-gráficos en una misma Figura pueden usarse los parámetros *mfrow* o *mcol* de la función *par*. La ejecución del comando:

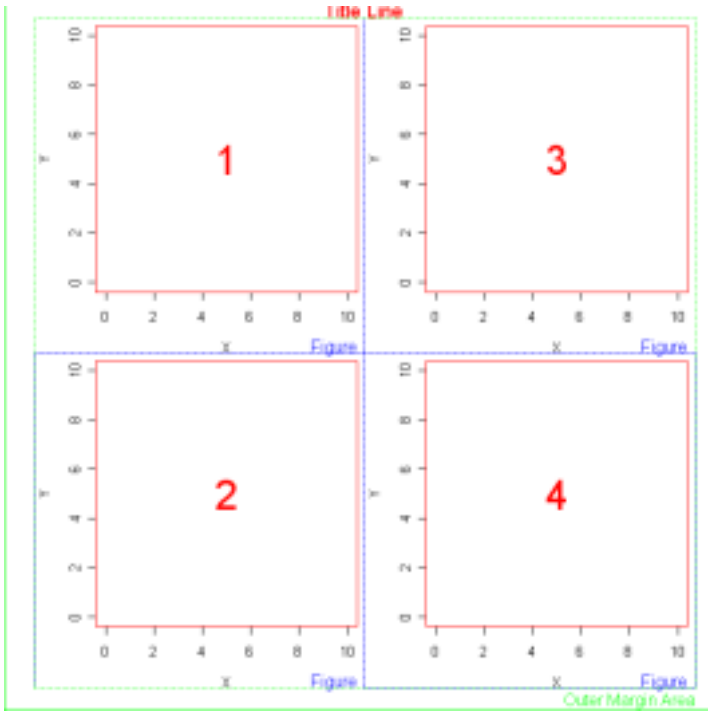
```
par(mfrow=c(2,2))
```

divide a la figura en 4 sub-gráficos (2 filas y 2 columnas), comenzando a graficar desde arriba a la izquierda y avanzando a través de las filas como se muestra en la siguiente figura:



Alternativamente,

par(mcol=c(2,2), mar=c(4, 4, 0.5, 0.5), oma=c(1.5, 2, 1, 1) hace lo mismo pero avanzando por columna, y además en este caso se está modificando el ajuste de los márgenes internos y externos tal y como vimos anteriormente, para optimizar los espacios en blanco presentes entre cada gráfico. Aquí vemos además una utilidad del margen externo: la inclusión de un título general para la Figura diseñada.



Algunos otros *tips* útiles respecto a múltiples gráficos en una misma figura:

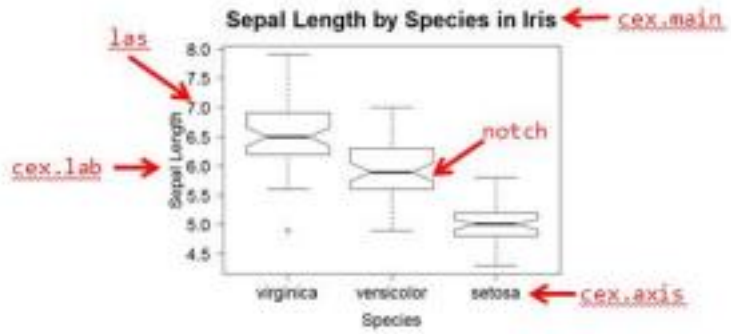
· Una mayor flexibilidad para hacer múltiples gráficos en una misma figura se consigue utilizando el argumento *fig*, a partir del cual podemos controlar más efectivamente en qué lugar de la figura se va a ubicar el gráfico. Para ello hay que proveerle el área del gráfico a partir de sus coordenadas x e y normalizadas entre (0,1) mediante el formato c(x1, x2, y1, y2). Por ej: con el comando par(fig=c(0.1,0.4,0.1,0.4)) estaríamos localizando al gráfico arriba a la izquierda en el área total de la figura.

· Para combinar *plots* en un mismo gráfico se utiliza el comando par(new=TRUE); de esta manera, el siguiente *plot* que hagamos se va a ubicar sobre el área gráfica del *plot* anterior (obviamente hay que tener cuidado que no se superpongan; para ello se pueden usar transparencias, etc).

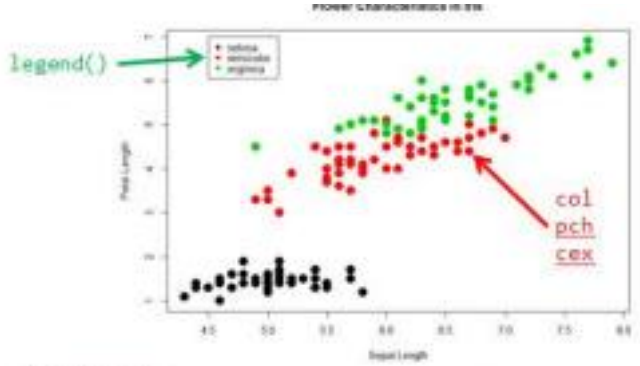
Principales *argumentos* para modificar las salidas gráficas de R con el paquete base

En las siguientes figuras podrán ver los principales argumentos que se pueden modificar para editar gráficos según nuestras necesidades, y el código en R para realizarlo:

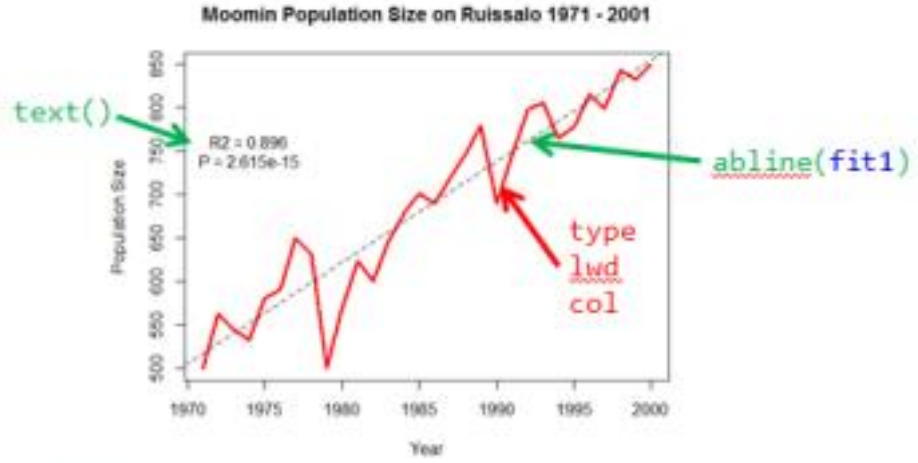

```
#-- FINAL PLOT:

hist(unicornsBirthweight,                          # x value
     breaks = 40,                                  # number of cells
     xlab = "Birth weight",                        # x-axis label
     main = "Histogram of Unicorn Birth Weight",   # plot title
     ylim = c(0,80))                               # limits of the y axis (min,max)
```




```
#-- FINAL PLOT

irisSpecies<-factor(irisSpecies, levels = c("virginica","versicolor","setosa"))

boxplot(irisSepal.Length ~ irisSpecies,            # x variable, y variable
        notch = T,                                 # Draw notch
        las = 1,                                   # Orientate the axis tick labels
        xlab = "Species",                          # x-axis label
        ylab = "Sepal Length",                     # y-axis label
        main = "Sepal Length by Species in Iris",  # Plot title
        cex.lab = 1.5,                             # Size of axis labels
        cex.axis = 1.5,                            # Size of the tick mark labels
        cex.main = 2)                              # Size of the plot title
```




```
# FINAL PLOT

plot(Irissepal.Length, Irispetal.Length,        # x variable, y variable
     col = Irisspecies,                         # colour by species
     pch = 16,                                  # type of point to use
     cex = 2,                                   # size of point to use
     xlab = "Sepal Length",                     # x axis label
     ylab = "Petal Length",                     # y axis label
     main = "Flower Characteristics in Iris")   # plot title

legend (x = 6.5, y = 7, legend = levels(Irisspecies), col = c(1:3), pch = 16)
# legend with titles of Irisspecies and colours 1 to 3, point type pch at coords (x,y)
```

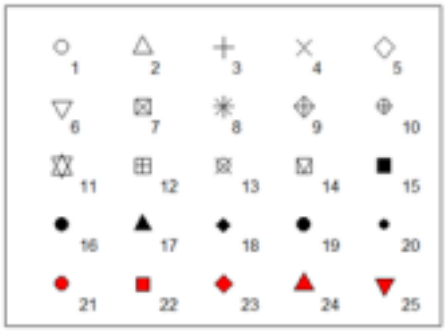



```
#-- FINAL PLOT Script

plot(moominsYear, moominsPopSize,                          # x variable, y variable
     type = "l",                                           # draw a line graphs
     col = "red",                                          # red line colour
     lwd = 3,                                              # line width of 3
     xlab = "Year",                                        # x axis label
     ylab = "Population Size",                             # y axis label
     main = "Moomin Population Size on Ruissalo 1971 - 2001")   # plot title
fit1 <- lm (PopSize ~ Year, data = moomins)                # carry out a linear regression
abline(fit1, lty = "dashed")                               # add the regression line to the plot
text(x=1974,y=750,labels="R2 = 0.896\nP = 2.615e-15")      # add a label to the plot at coordinates (x,y)

```

(extraído de: http://rstudio-pubs-static.s3.amazonaws.com/7953_4e3efd5b9415444ca065b1167862c349.html)

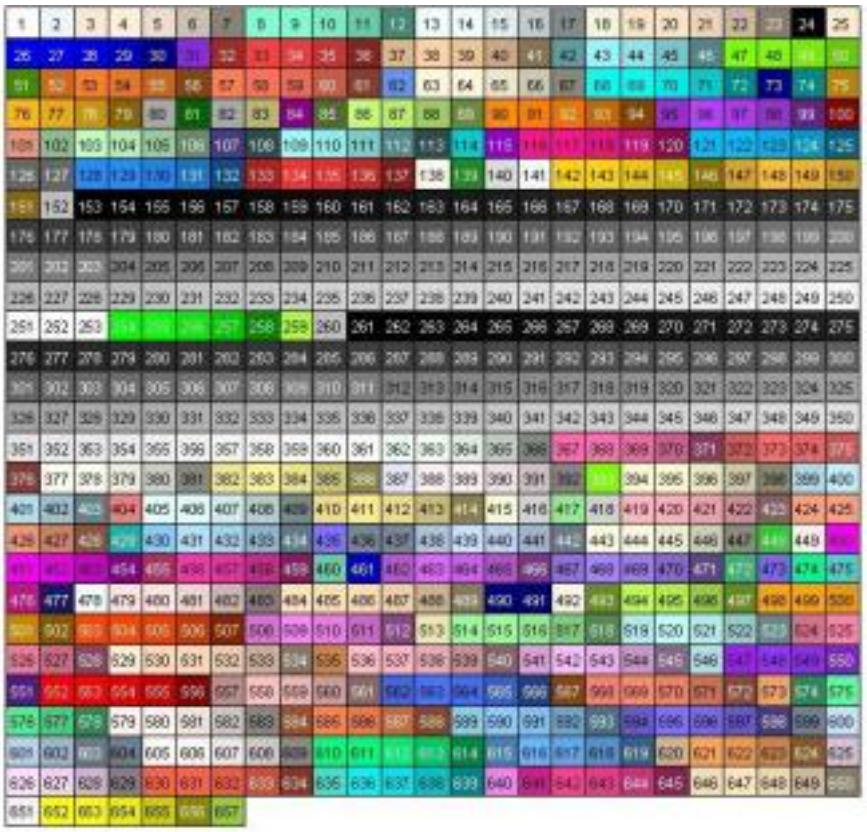
Argumento	**Descripción**
ylim, xlim	Ajustan los límites (min, max) de los ejes y e x respectivamente
main, xlab, ylab	Introducen un título y un rótulo para cada eje
cex.main, cex.lab, cex.axis	Ajustan el tamaño de letra del título principal, de los rótulos, y de los valores de cada eje
col, pch, cex	Ajustan el color, el tipo de símbolo y el tamaño de los puntos de cada dato.
lty	Ajusta el tipo de línea

Símbolo elegibles para salidas gráficas (argumento "pch")



Tipos de Líneas (argumento "lty")

Tabla de colores R



En el siguiente link podrán encontrar una tabla con los nombres detallados de cada color:

http://research.stowers.org/mcm/efg/R/Color/Chart/ColorChart.pdf

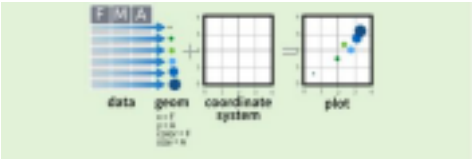
Para hacer gráficos con semi-transparencias, el comando rgb(red, green, blue, alpha) nos permite asignarle a través del código de colores RGB un color y una transparencia con el parámetro alpha (1 -> totalmente opaco; 0 -> totalmente transparente). Todos los parámetros (red, green, blue, alpha) pueden tomar valores entre 0 y 1.

Estructura de salidas gráficas usando ggplot2

Un gráfico en ggplot2 consiste de una serie de componentes que se van disponiendo en el dispositivo gráfico, con una sintaxis y gramática que flexibiliza enormemente el tipo de cosas que se pueden hacer y simplifica la creación de salidas gráficas nuevas (lo que se denomina *Grammar*

of Graphics). Para eso sólo hay que conocer la estructura básica de la sintaxis, buenas recomendaciones y… mucha práctica.

Los componentes básicos de cualquier gráfico creado con ggplot son:



Base de datos (**data frame**) + Geometría de visualización (***geoms***) + Mapeos gráficos de variables (***aes***) + sistema de coordenadas (**coord**)

Esencialmente, uno empieza especificando la base de datos que desea utilizar, un sistema de visualización de los datos (**geoms** – puntos, líneas, polígonos, etc), algunas características gráficas que deseamos agregar -y que mapean alguna variable a los datos con un color determinado, tamaño, ubicación, etc. (***aesthetic mappings; aes***), y un determinado sistema de coordenadas (**coord.**; que por *default* es el Sistema Cartesiano); para luego ir agregando gradualmente distintas piezas o *layers* para crear un gráfico (a través de más **geoms** por ejemplo).

Otros componentes opcionales que pueden incluirse son:

- Múltiples paneles (***facets***): describe la manera en que varios subplots/plots condicionales deben ser construidos en el gráfico

- Transformaciones estadísticas *(**stats***): permite realizar múltiples transformaciones (obtención de cuartiles, agregación de datos, etc)

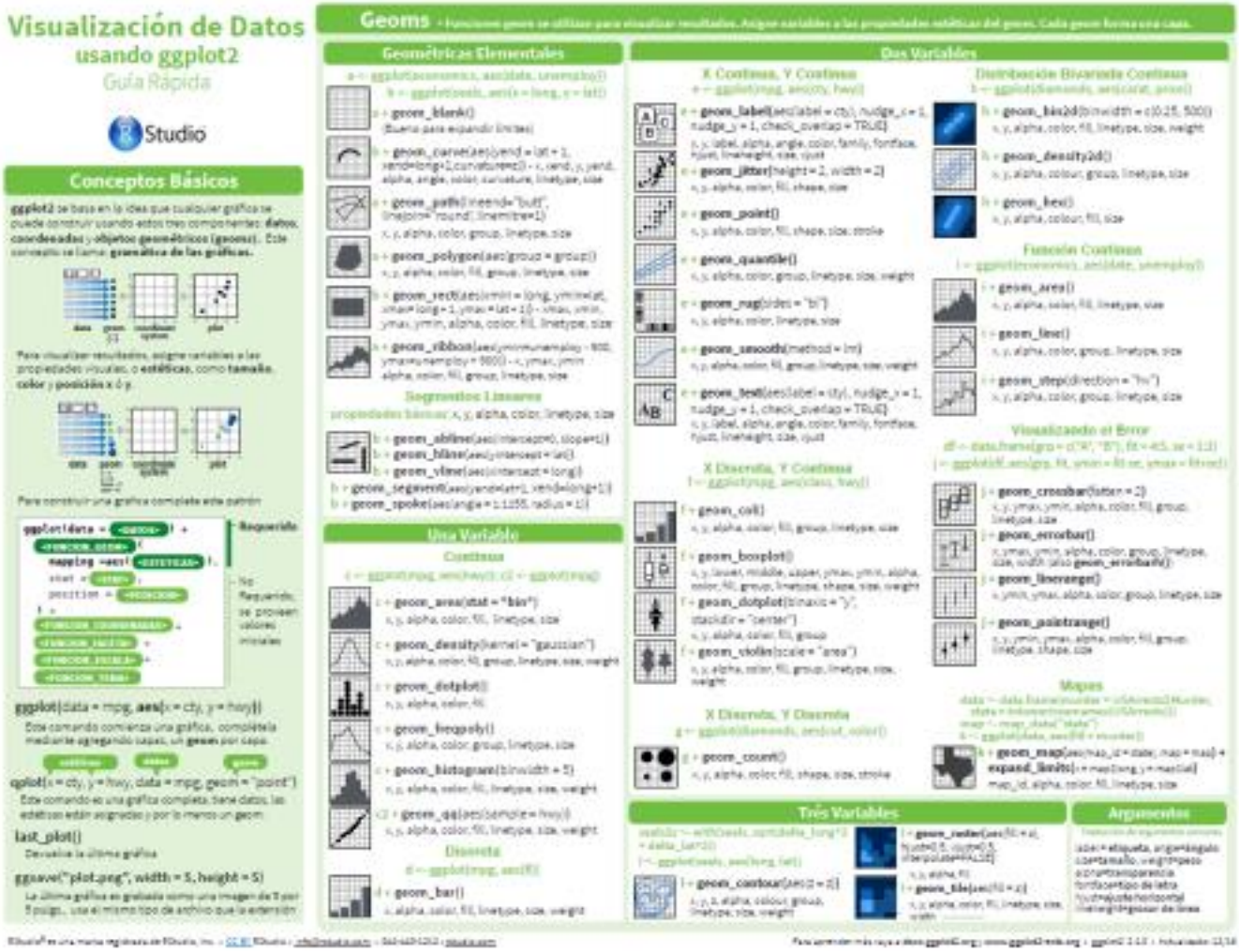
- Escala (***scales***): permite seleccionar la escala utilizada en el mapeo de alguna variable (por ej. Una escala discreta control: rojo, tratados: azul)

Un ejemplo sencillo de la sintaxis:

```
ggplot(Datos, aes(LongSepalo, AnchoPetalo)) +
 geom_point(colour="black") +
 geom_smooth(colour="red")
```

El commando ggplot() es uno de los comandos que nos permite comenzar una nueva salida gráfica en el entorno ggplot2. En el ejemplo, *Datos* es nuestro *dataframe* donde están contenidos los datos crudos, y los primeros dos argumentos de ***aes*** le indica cuáles son las variables x e y (siempre, por *default*). Luego, le indicamos que la geometría de visualización de una primera capa es una geometría de puntos de color negro (geom_point), y por último agregamos un estimador suavizado en color rojo para ver el patrón dominante de los datos.

A continuación les dejamos una referencia rápida a los componentes más usados en *ggplot* y los principales argumentos posibles (extraído de https://www.rstudio.com/resources/cheatsheets/):



Bibliografía (por si quieren profundizar más en estos temas)

Chang, W. (2012). R Graphics Cookbook: Practical Recipes for Visualizing Data. " O'Reilly Media, Inc.". Un MUY buen libro con ejemplos hechos tanto con el paquete base de R como con ggplot2.

Peng R (2004) Exploratory Data Analysis. Libro online. https://bookdown.org/rdpeng/exdata/ Un resumen básico y sintético del análisis exploratorio en R bastante ameno de leer y seguir.

Quinn GP, Keough MJ (2002) Experimental design and data analysis for biologists. Cambridge Univ Press. Capitulo 4: Graphical exploration of data.

Wickham, H. (2016). *ggplot2: elegant graphics for data analysis*. Springer.

Wickham, H., & Grolemund, G. (2016). R for data science. O'Reilly. Excelente libro para iniciarse con ggplot2 (entre otras cosas en R). Versión on-line: http://r4ds.had.co.nz/

Zuur, A. F., Ieno, E. N., & Elphick, C. S. (2010). A protocol for data exploration to avoid common

statistical problems. Methods in Ecology and Evolution, 1(1), 3-14.

Zuur, A. F. & Ieno, E. N. (2016). A protocol for conducting and presenting results of regression-type analyses. Methods in Ecology and Evolution, 7, 636–645.