Técnicas de visualización y *reporting*

PID_00264733

Xavi Font

Tiempo mínimo de dedicación recomendado: 4 horas

Universitat Oberta de Catalunya

Xavi Font

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Jordi Ayza Graells (2019)

Primera edición: marzo 2019

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Oberta UOC Publishing, SL

Índice

Introducción

Las técnicas de visualización, y en general de procesos de comunicación visual, tienen en la actualidad una relevancia muy importante. La interpretación de estudios a través de valores numéricos o de tablas no es siempre la mejor opción. En ocasiones, estas herramientas no sirven para incrementar el conocimiento o pueden generar una interpretación errónea de la que los investigadores buscaban.

Imaginemos que disponemos de los salarios de una muestra de trabajadores agrupados según el grupo empresarial donde trabajan.

Grupo 1		Grupo 2		Grupo 3		Grupo 4	
YearG1	WageG1	YearG2	WageG2	YearG3	WageG3	YearG4	WageG4
5.72	2320	6.37	2320	5.38	2320	4.86	1946
5.08	1946	5.78	1946	4.97	1946	4.38	1946
5.45	2881	6.13	2881	8.48	2881	5.53	1946
6.17	2133	6.15	2133	5.17	2133	6.19	1946
5.89	2507	6.44	2507	5.58	2507	5.97	1946
6.85	3068	5.75	3068	6.19	3068	5.13	1946
5.25	1572	4.60	1572	4.57	1572	4.08	1946
3.50	1198	2.82	1198	4.16	1198	8.34	4003
7.36	2694	6.36	2694	5.78	2694	4.26	1946
3.83	1759	5.26	1759	4.77	1759	5.64	1946
4.33	1385	3.78	1385	4.36	1385	5.04	1946

¿Qué podemos intentar resolver al visualizar esta tabla de salarios en euros mensuales brutos y años de estudios no obligatorios? Podemos intentar obtener una idea de lo que los datos ofrecen e incluso intentar resolver algunas de las siguientes cuestiones:

- ¿Qué grupo paga más a sus trabajadores?
- ¿Qué grupos tienen trabajadores mejor formados?
- ¿Qué correlación existe entre la formación educativa y los salarios?

Si realizamos unos sencillos cálculos sobre la variable años de estudios, podemos ver que para cualquiera de los grupos en consideración, el valor de la media aritmética, así como el de la desviación, es el mismo. Es decir, aunque los valores representativos de cada grupo parecen claramente diferentes, el promedio es exactamente el mismo. ¿Qué diferencia hay, pues, en las cuatro situaciones presentadas en lo que respecta a los años de educación no obligatoria?

	YearG1	YearG2	YearG3	YearG4
Mean	5.403	5.403	5.402	5.403
Sd	1.193	1.193	1.192	1.192

Podemos ver qué sucede con la variable de salarios mensuales. De nuevo, se aprecia (no con la misma sorpresa anterior) que los estadísticos de medida central y de dispersión correspondientes a la media y a la desviación dan el mismo valor para cada grupo. Parece que en promedio voy a ganar exactamente lo mismo, es decir, 2.133 euros independientemente del grupo elegido.

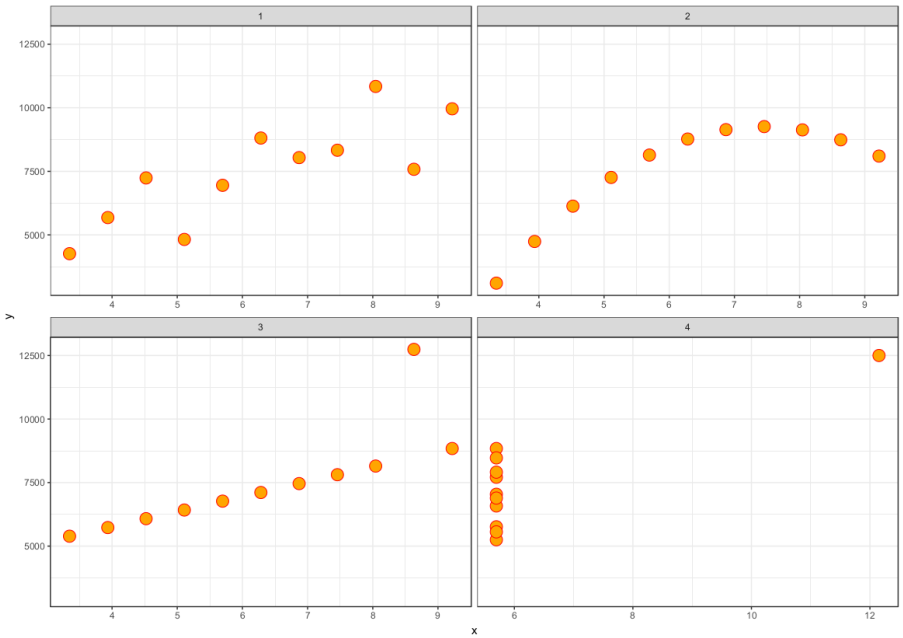
	WageG1	WageG2	WageG3	WageG4
Mean	2133.0	2133.0	2133.0	2133.0
Sd	620.2	620.2	620.2	620.2

Si de nuevo observamos otras medidas de carácter numérico como las correlaciones entre ambas variables, ¿qué diferencias observaremos? Seguramente que los valores resumen respecto al conjunto de grupos es exactamente el mismo. El cálculo de las correlaciones para cada grupo sorprendentemente es el mismo para los cuatro grupos.

Group 1	Group 2	Group 3	Group 4
0.816	0.816	0.816	0.816

Todas estas sorpresas que hemos mostrado pueden desvanecerse si aplicamos otra aproximación. ¿Qué cambio se produce en la interpretación de este conjunto de datos si realizamos una visualización gráfica de los datos?

Figura 1. Representación de las observaciones



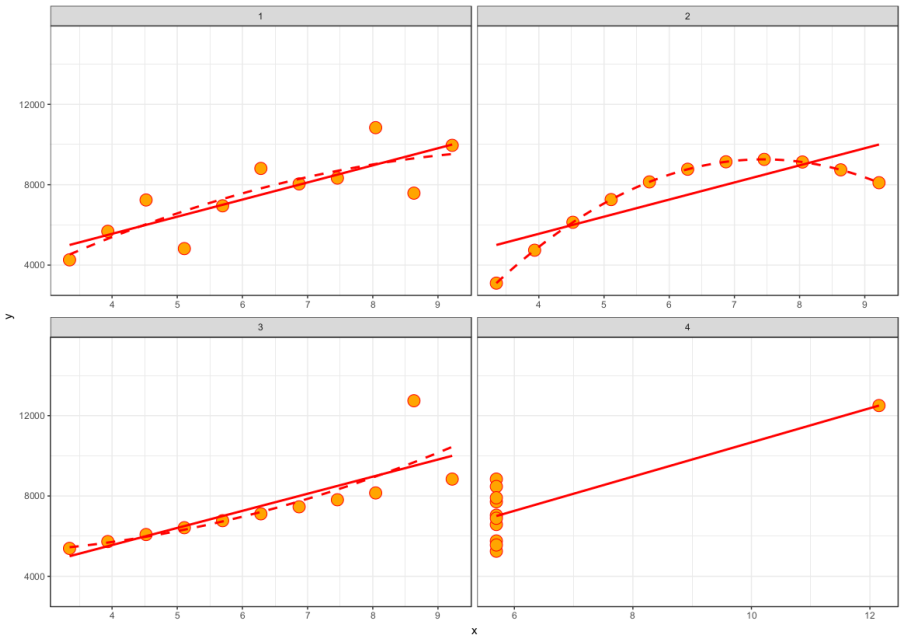
La correcta interpretación de la naturaleza de los datos salta a la vista con la representación gráfica de la figura 1. De un vistazo podemos obtener mucha más información de la que anteriormente obtuvimos a través de las tablas. De hecho, una tabla obliga a una mayor concentración y análisis para generar parte de la historia contada por los datos. En este módulo es relevante la historia que se quiere que los datos reflejen y transmitan a través de diferentes estrategias de comunicación.

Las estrategias de comunicación que se pueden utilizar son:

- La utilización del color.
- La posición de los objetos en el gráfico o de los datos.
- La forma de los datos que representar.
- Los arcos como indicación de importancia.
- La medida de los puntos.
- La distribución o agrupación de los elementos que representar.

¿Existen diferencias entre los cuatro grupos? Obviamente, si consideramos la modelización de los datos con una regresión polinómica, se pueden apreciar sustanciales diferencias entre los grupos. Y quizá la interpretación que de estos modelos se deriva (por su capacidad predictiva) es claramente diferente.

Figura 2. Representación de las observaciones con modelos de regresión lineal y polinómico ajustados sobre los datos de cada grupo



Objetivos

Los objetivos que se ofrecen en este módulo son principalmente tres. En primer lugar, la capacidad para generar un gran conjunto de gráficos diversos y poderlos interpretar correctamente. Se entiende que la aplicación del gráfico se ajusta a los objetivos de este. En segundo lugar, describir la importancia de los cuadros de mando (*dashboard*) para el correcto seguimiento de las KPI de nuestra empresa o unidad de negocio. Se consideraran diferentes opciones y aproximaciones a los *dashboards*. En tercer y último lugar, la utilización de herramientas de *reporting* automatizadas.

Esquemáticamente se pretende:

1. Representación gráfica de datos

- Uso de gráficos estáticos (librería ggplot2)
- Uso de gráficos dinámicos (librería plotly)
- Uso de librerías adicionales para representar otras situaciones: árboles, redes, etc.

2. Descripción de cuadros de mando (*dashboards*): diferentes enfoques para el uso de estas herramientas

3. Utilización de herramientas de *reporting* vía ficheros *.Rmd*

- Sintaxis asociada a los documentos R markdown
- Documentos HTML
- Documentos Word y documentos PDF
- Generación de diapositivas

1. Estrategias de visualización

1.1. Introducción

Vamos a introducir con un poco más de detalle una de las librerías que ya fueron introducidas en el módulo de *data cleaning* y *data wrangling*. La librería ggplot2 es una de las librerías más utilizadas como herramienta eficiente para realizar buenas representaciones gráficas. Explicaremos cómo se construye un gráfico desde cero para que se adquiera suficiente habilidad como para poder generar nuestros gráficos de una manera autónoma y sin necesidad de intermediarios.

Se explicarán las representaciones más utilizadas (no todas) y se ofrecerán alternativas para convertir los gráficos en dinámicos. Esta opción es interesante si la salida es un documento HTML o se quiere generar un informe para subirlo a internet. La librería plotly es otra opción de interés para este tipo de salidas o la librería highcharter.

Este apartado terminará con algunas representaciones específicas y que tienen habitualmente una o varias librerías especializadas. Nos referimos a utilizar representaciones de geolocalización, a la utilización de árboles de decisión o a representaciones de red.

1.2. Gráficos con ggplot2

La librería ggplot2 desarrollada por Hadley Wickham que implementa la gramática gráfica (significado del uso de las dos ges) se ha convertido en la librería de visualización de datos principal en el entorno R. Permite describir de una manera efectiva los diferentes elementos que intervienen al generar gráficos para nuestros datos.

La filosofía detrás de esta gramática es coherente y efectiva. Permite crear el gráfico utilizando los elementos que intervienen en la creación de este. Cómo creamos un gráfico:

- Especificamos el objeto (*data frame*) que contiene los datos que representar.
- Mapeamos entre los elementos de la visualización (*aesthetic*) y las variables de los datos.
- Indicamos los objetos que hay que visualizar, que pueden ser: líneas, puntos, etc.

- Especificamos si hay alguna transformación.
- Indicamos el sistema de coordenadas, posición y escalas.
- Agrupamos por vía de los *facets*.

La forma de organizar estos elementos es por capas. De esta manera, la creación de un gráfico pasa por añadir capas según nuestras necesidades y prioridades.

Para poder practicar de modo conveniente, debemos tener instalada la librería ggplot2 o bien la librería tidyverse. Si ya tenemos alguna de estas dos librerías instaladas, no es necesario realizar este paso.

```
install.packages("tidyverse")

install.packages("ggplot2")
```

Con la librería instalada ya podemos cargarla para poner en práctica diferentes ejemplos. Todos los ejemplos parten del siguiente inicio:

```
library("ggplot2")
data(mtcars)
```

Seguiremos tratando el conjunto de datos ya visto repetitivamente mtcars.

1.2.1. Gráficos básicos

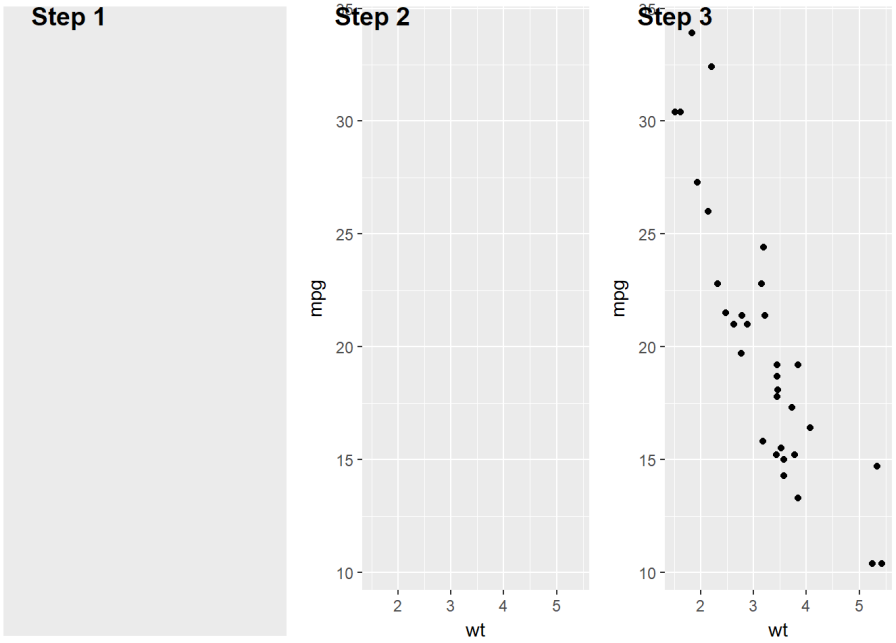
La realización de un gráfico parte de los siguientes elementos:

- la creación de la ventana de visualización del gráfico (canvas),
- enlace (o mapeo) de las variables a la ventana de visualización y
- especificación del elemento que se ha de visualizar.

Los elementos especificados en el mapeo tienen relación directa con el objeto gráfico que vamos a visualizar. Si es un diagrama x *vs*. y, la especificación en el mapeo tiene relación con el elemento gráfico que se va a utilizar (por ejemplo, un punto). En la figura 3 se aprecia esta situación.

```
ggplot()
ggplot(mtcars,mapping = aes(x = wt, y = mpg))
ggplot(mtcars,mapping = aes(x = wt, y = mpg)) + geom_point()
```

Figura 3. Representación de las observaciones



Los *maping* que realizamos enlazan las características de los datos con las características de la representación gráfica. Las más utilizadas son:

- **Posición**: coordenadas x e y.
- **Color**: que permite identificar una variable categórica fácilmente.
- **Tamaño**: que permite añadir otra característica numérica de los datos.

1.2.2. Formas disponibles

La librería ggplot2 dispone de un gran repertorio de formas, que denomina *geom*. Podemos disponer de los siguientes *geom*:

- **geom_point**: para el uso de puntos.
- **geom_line**: para la representación de líneas.
- **geom_histogram**: para visualizar histogramas.
- **geom_boxplot**, **geom_violin**, **geom_dotplot**: representación robusta de la distribución de los datos en formato *box-plot*, violín o gráfico de puntos.

En la representación de los histogramas vamos a usar la fórmula de Freedman-Diaconis para determinar el número de grupos. La longitud de cada grupo se calcula con la fórmula:

$$h = \frac{2 \times IQR}{n^3}$$

Y de esta fórmula se deduce el número de grupos.

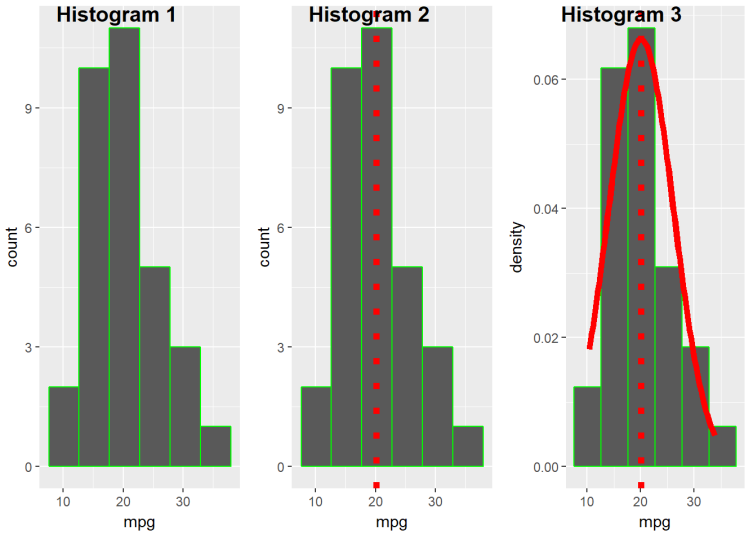
$$bw = \frac{max - min}{h}$$

De esta manera evitamos que el valor por defecto sea excesivamente grande o demasiado pequeño. La figura mostrada representa tres histogramas. Se puede observar que los tres histogramas son iguales, excepto que en el segundo se ha añadido una nueva capa que corresponde a una línea vertical para indicar la posición de la media. En el tercer histograma, se añade adicionalmente la distribución normal asociada (ver figura 4).

El código asociado a la representación es el siguiente:

```
x <- mtcars$mpg
bw <- diff(range(x)) / (2 * IQR(x) / length(x)^(1/3))

p0 <- ggplot(mtcars,mapping = aes(x=mpg))
p0 + geom_histogram(binwidth = bw,colour ="green")
p0 + geom_histogram(binwidth = bw,colour ="green")+
  geom_vline(xintercept = mean(x),linetype="dotted",colour="red",size=2)
p0 + geom_histogram(aes(y = ..density..),binwidth = bw,colour ="green")+
  geom_vline(xintercept = mean(x),linetype="dotted",colour="red",size=2)+
  stat_function(fun=dnorm,color="red",
args=list(mean=mean(mtcars$mpg),sd=sd(mtcars$mpg)),colour="red",size=2)
```

Figura 4. Representación de histogramas con diferentes objetos añadidos, línea de puntos y distribución normal

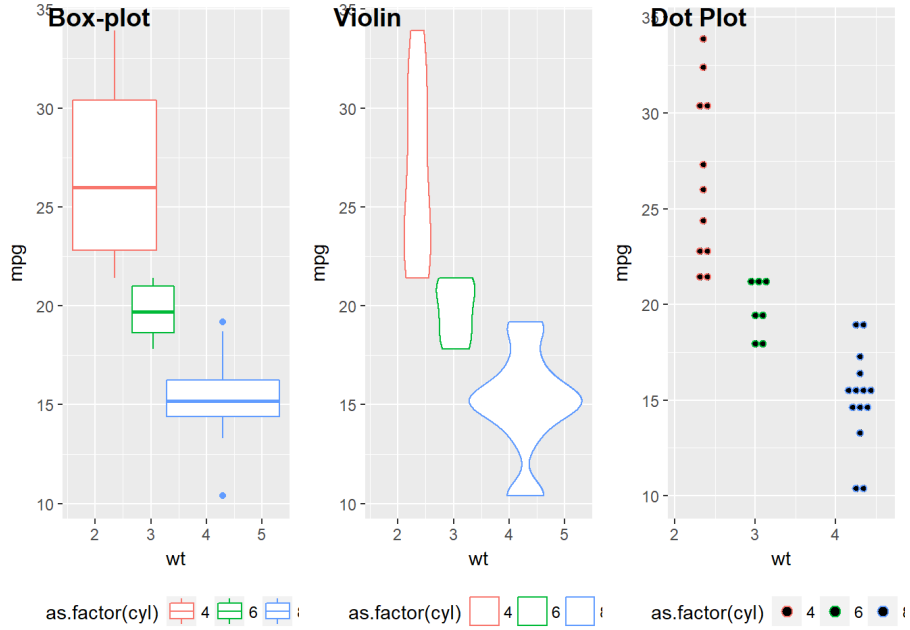


En la siguiente representación (ver figura 5) tenemos el conjunto de diagramas similares al conocido *box-plot*.

El código asociado a la representación es el siguiente:

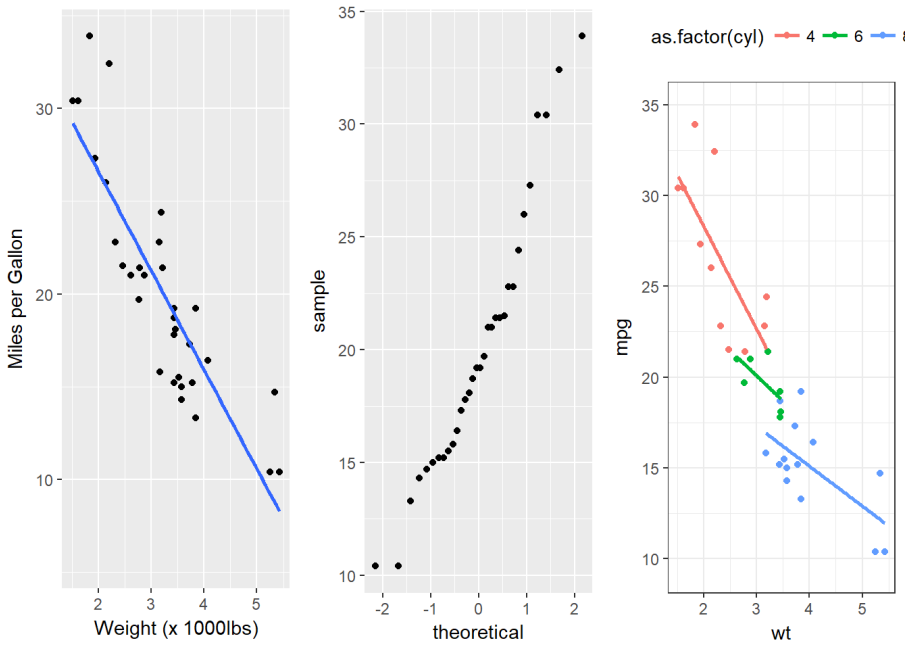
```
p0 <- ggplot(mtcars,mapping = aes(x = wt, y = mpg,group=as.factor(cyl)))
p0 + geom_boxplot(aes(color=as.factor(cyl)))+theme(legend.position="bottom")
p0 + geom_violin(aes(color=as.factor(cyl)))+theme(legend.position="bottom")
p0 + geom_dotplot(aes(color=as.factor(cyl)),stackdir = "center",
 binaxis = "y", dotsize = 0.5)+theme(legend.position="bottom")
```

Figura 5. Representación de los *geom box-plot*, violín y diagrama de puntos



Representación de tres diagramas relativos a la variable *mpg*. En el primer caso como variable respuesta, en el segundo caso como diagrama QQ y en el último caso en función de la variable categórica número de cilindros (ver figura 6).

Figura 6. Representación de *scatter plot* junto a recta de regresión, *quantile plot* de la variable mpg y regresión lineal para cada grupo según número de cilindros



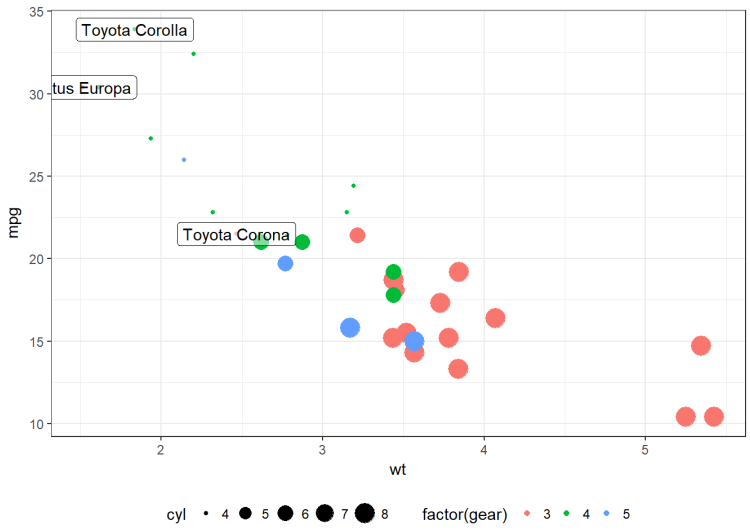
Gráficos generados por:

```
p0 <- ggplot(mtcars,aes(x=wt,y=mpg)) + geom_point() + xlab('Weight (x 1000lbs)')
+ ylab('Miles per Gallon')
p0  + geom_smooth(method = 'lm',alpha=0)

ggplot(mtcars,aes(sample=mpg)) + stat_qq()

ggplot(data = mtcars, mapping = aes(x = wt, y = mpg, color = as.factor(cyl)))
+ geom_point() +
  stat_smooth(method = 'lm',alpha=0)+theme_bw()+
  theme(legend.position="top")
```

La principal ventaja de la sintaxis que utiliza ggplot2 es que la construcción del gráfico se puede descomponer en aquellos elementos que, desde nuestra posición, son importantes y deben aparecer en la visualización. La incorporación de etiquetas en la representación puede ser un buen ejemplo (figura 7).

```
labCars <- row.names(mtcars)
df <- cbind(mtcars,labCars)
bestCarByG <- df %>%
  group_by(gear) %>%
  filter(row_number(desc(mpg)) == 1)

ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(aes(color = factor(gear),size=cyl)) +
  geom_label(data = bestCarByG, aes(label = labCars), alpha = 0.5)
```

Figura 7. Representación de *scatter plot* junto a etiquetas para indicar modelo de vehículo con mejor consumo según número de marchas



Otros de los aspectos de interés son las coordenadas, la orientación y las escalas.

1.2.3. Sistemas de coordenadas y escalas

Todo el sistema de coordenadas disponibles en ggplot2 comienza con el nombre *coord_*. Las operaciones más habituales son las siguientes:

- **coord_cartesian**: es el sistema por defecto. No es necesario especificar.
- **coord_polar**: sistema de coordenadas polares.
- **coord_flip**: esta operación cambia los ejes x por y.
- **coord_fixed**: permite fijar el aspecto entre x e y (por ejemplo, la proporción áurea 1.61 (1 + sqrt(5))/ 2 # golden ratio portrait o bien 2 / (1 + sqrt(5)) # golden ratio landscape

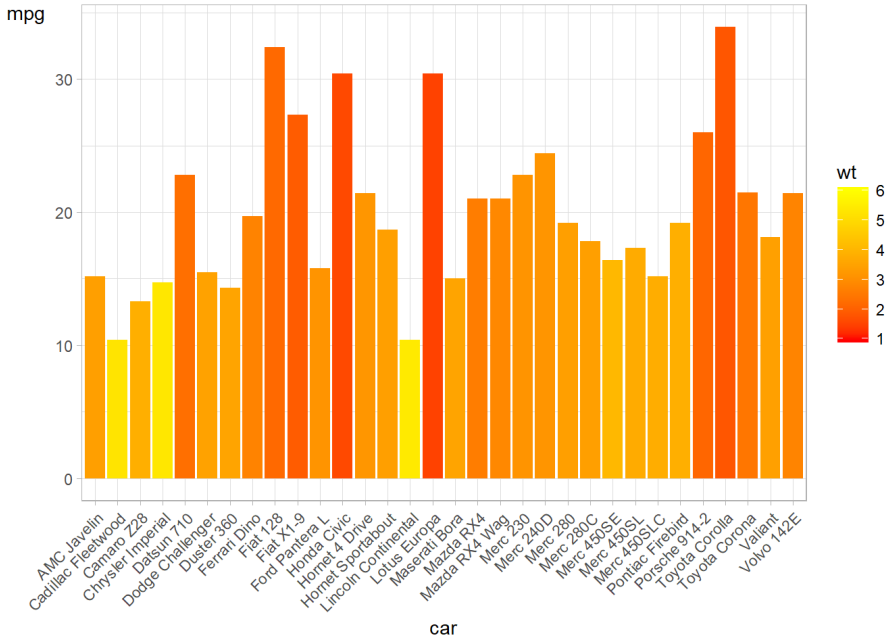
En la visualización de las coordenadas tendrá efecto la identificación de cómo son las escalas para las abscisas y las ordenadas.

- **scale_x_log10()**: permite aplicar sobre el eje x una transformación logarítmica (base 10).
- **scale_x_sqrt()**: permite transformar el eje x con una transformación tipo raíz cuadrada.
- **scale_x_reverse()**: cambia la dirección de la escala x.
- **xlim(minx, maxx)**: fija los límites del eje x.
- **expand_limits(x=0, y=0)**: permite que el sistema de coordenadas empiece por (0,0).
- **scale_x_continuous(name, breaks, labels, limits, trans)**: con varias opciones.

Para realizar el ejemplo con coordenadas polares, vamos a representar inicialmente un diagrama de barras. La forma de realizar un diagrama de barras es con una variable categórica en el eje de las abscisas y una variable numérica para las ordenadas (o bien representar una medida de número de observaciones por grupo).

```
mtcars$car = row.names(mtcars)
p <- ggplot(mtcars, aes(x=car, y=mpg, fill=wt)) +
    geom_bar( stat="identity") +theme_light() +
    scale_fill_gradient(low="red", high="yellow", limits=c(1,6))
    + # for mpg limits=c(5,40)) +
    theme(axis.title.y=element_text(angle=0))
p + theme(axis.text.x = element_text(angle=45, vjust = 1, hjust=1))
```

Figura 8. Representación de *scatter plot* junto a etiquetas para indicar modelo vehículo con mejor consumo según número de marchas

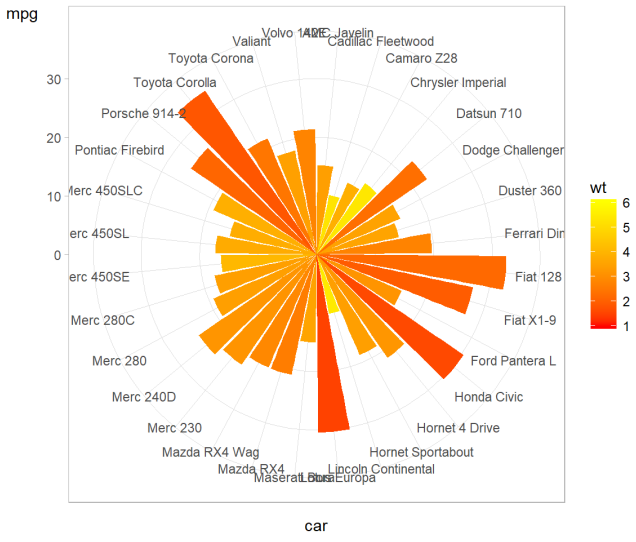


El gráfico muestra en color amarillo las vehículos más pesados, aquellos con valores de la variable wt mayores. El color rojo es para los vehículos que tienen menor peso. Entre ambos extremos hay una gradación.

Convertir el gráfico anterior en uno polar es muy fácil y directo. La primera línea detalla el primer diagrama polar, y la segunda hace referencia a la ordenación de estos según mpg.

```
p + coord_polar()
p + coord_polar() + aes(x=reorder(car, mpg)) +
    theme(axis.text.x = element_text(angle=-20))
```

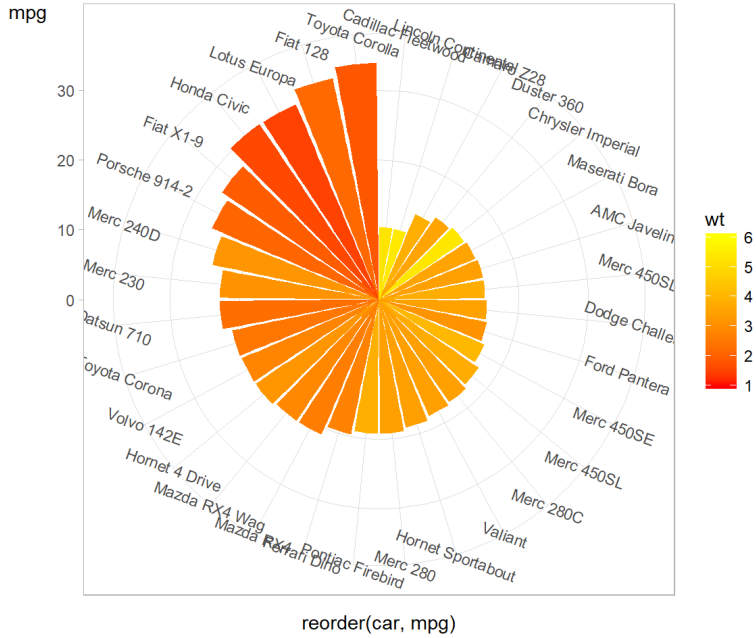
Figura 9. Diagrama polar de la representación anterior con un diagrama de barras



Observaciones

La utilización de representaciones en formato polar no son muy aconsejables por los expertos en los casos de comparación de valores entre diferentes objetos (usuarios, clientes, trabajadores, sistemas) sobre la retícula de características. Si queremos comparar, los gráficos tipo *lollipop* son una alternativa que ofrece muy buenos resultados. La razón científica detrás de este *warning* es que el lector del gráfico se centrará en la forma generada.

Figura 10. Diagrama polar con una ordenación y un cambio en la orientación de las etiquetas del modelo del vehículo



1.2.4. *Faceting*

La idea del *faceting* es dividir el gráfico en una matriz de subgráficos. Cada subgráfico visualiza una parte de la totalidad del conjunto de datos en función de las variables categóricas especificadas.

Por ejemplo, si se quieren visualizar los datos entre mpg y wt en función del número de cilindros, podemos realizar lo siguiente:

```
p <- ggplot(data = mtcars, mapping = aes(x = wt, y = mpg)) + geom_point()
p + facet_grid(~cyl) # Filas
p + facet_grid(cyl~.) # Columnas
p + facet_grid(cyl ~ am) # Filas y columnas
```

Por ejemplo, el uso de un ajuste sobre cada uno de los datos en función de la variable cilindros y cómo podemos ver el ajuste sobre la variable respuesta en función del tipo de modelo y el número de cilindro se visualiza en los siguientes gráficos:

Figura 11. *Facets* según número de cilindros y ajuste lineal

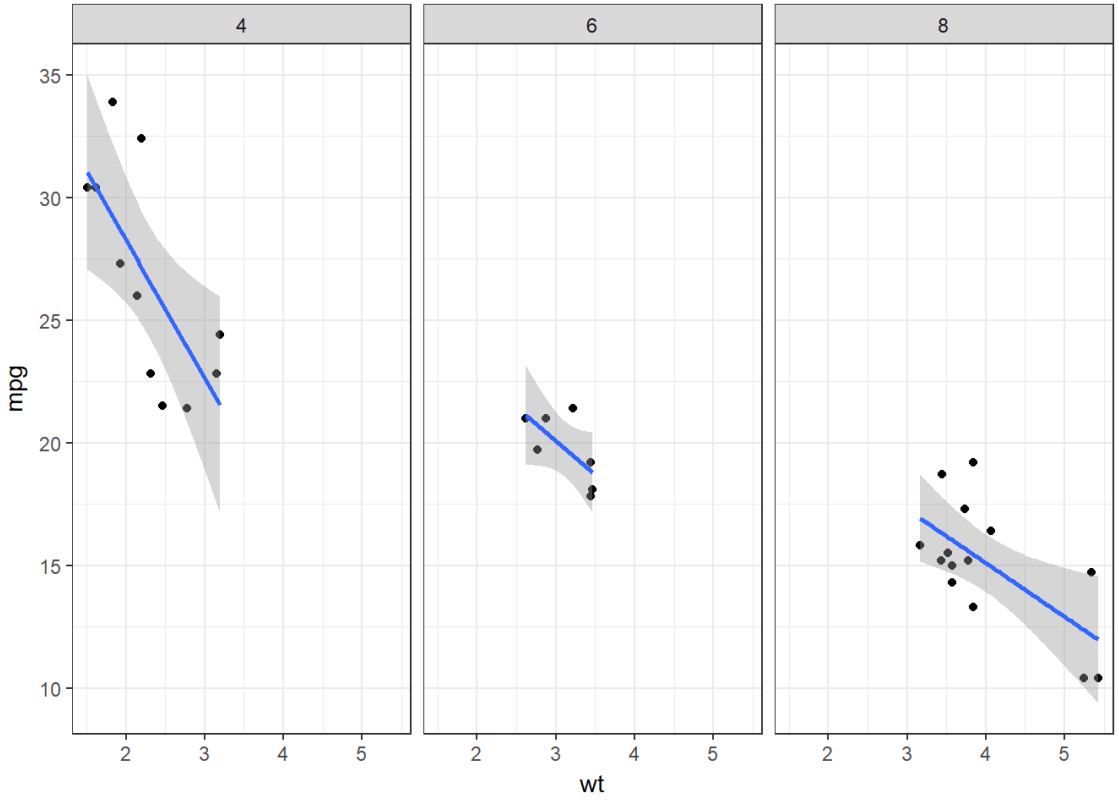


Figura 12. *Facets* según número de cilindros y ajuste lineal con escala eje x libre

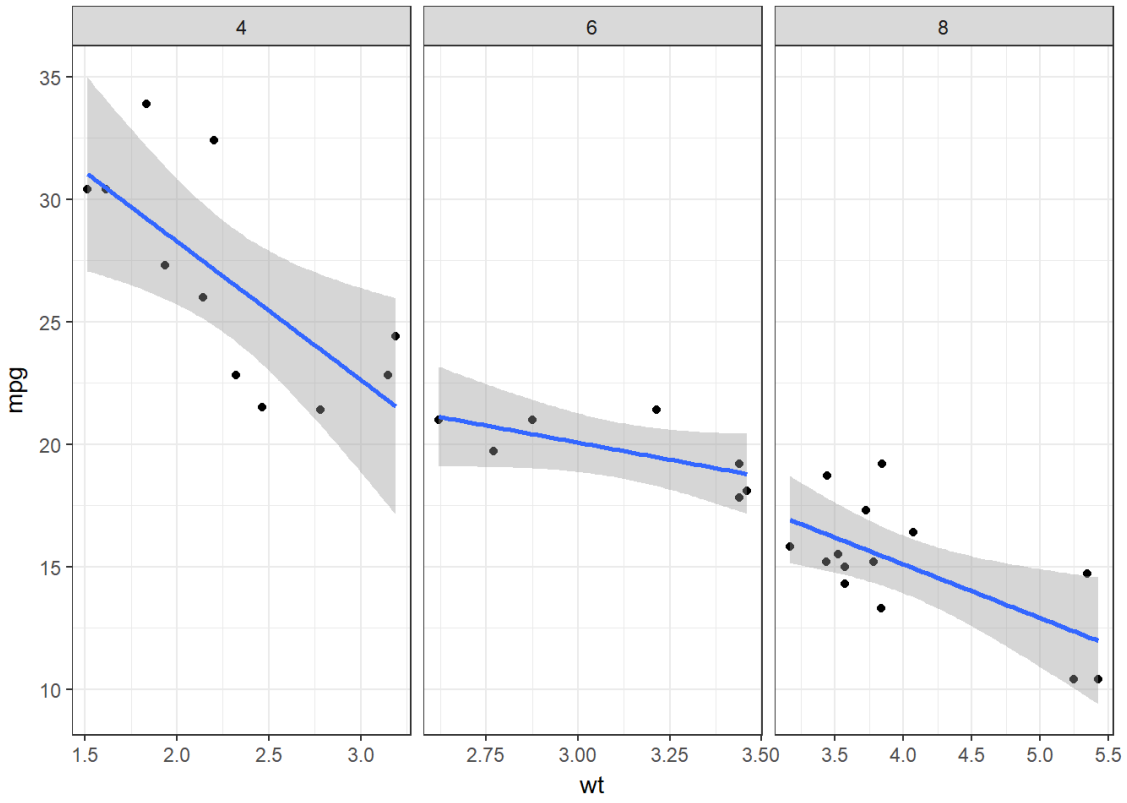


Figura 13. *Facets* según número de cilindros y ajuste cuadrático con escala x libre

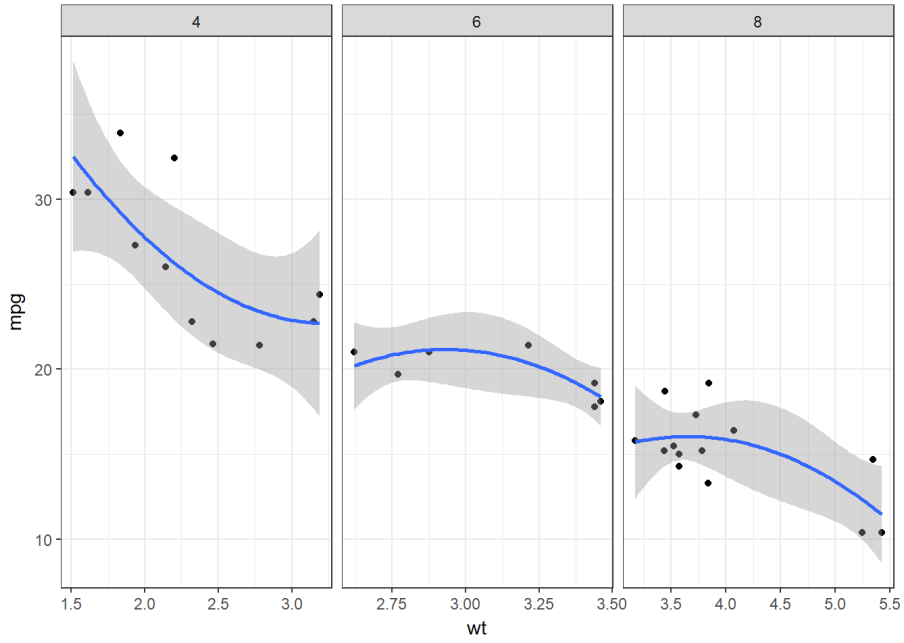
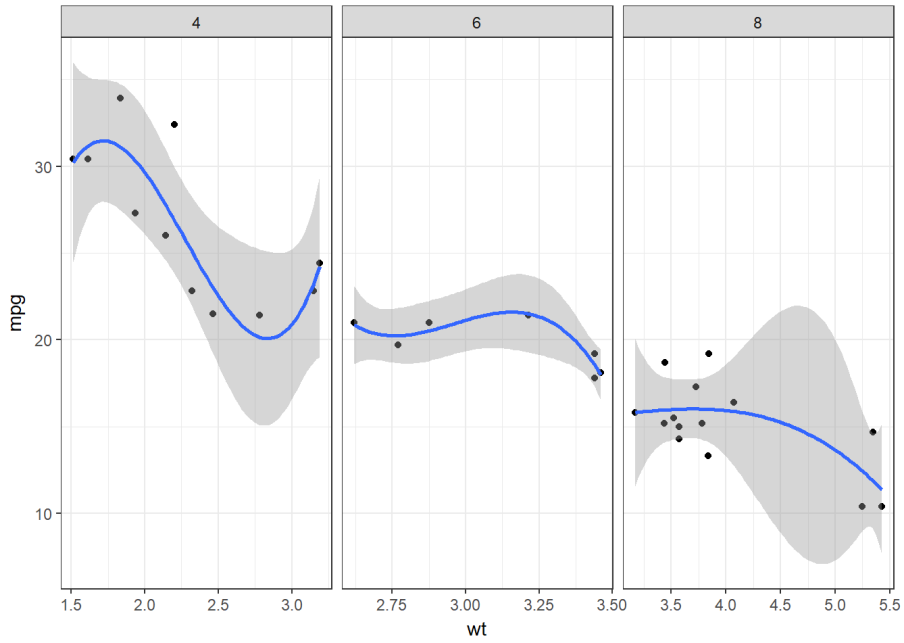


Figura 14. *Facets* según número de cilindros y ajuste cúbico con escala x libre



Enlaces de interés

Podemos encontrar en detalle todas las alternativas en estos enlaces:
https://bit.ly/2tyxc70
https://bit.ly/2XfTok4

1.3. Gráficos interactivos

Los gráficos interactivos son una alternativa muy interesante cuando se persigue generar salidas en formato HTML o que deben residir en páginas web y queremos dar al usuario un plus de flexibilidad e información. Otro aspecto muy diferente es que la salida interactiva sea realmente aprovechada por el conjunto de los usuarios que se supone que deben hacer uso de estas características.

R dispone de una gran cantidad de librerías que permiten la interacción. En las siguientes subsecciones vamos a comentar algunas de las más utilizadas.

Lectura de interés

https://bit.ly/2VdpxXz

1.3.1. La librería plotly

Plotly es una librería que tiene diferentes aspectos positivos. Entre ellos podemos destacar:

- Tiene como objetivo ofrecer un conjunto de herramientas de visualización y representación gráfica para favorecer el análisis de los datos.
- Es multiplataforma. Podemos utilizar los mismos gráficos en entornos como R, Python, Java, etc.
- Se basa en la librería D3.js.
- Fue desarrollado en Python bajo el *framework* Django.
- Permite pasar de la sintaxis de ggplot a la de plotly de forma directa.

Qué necesitamos para utilizar plotly:

1) instalar la librería:

install.packages("plotly")

2) cargar la librería:

library(plotly)

3) Realizar las representaciones gráficas que deseemos. Tenemos una gran cantidad de ejemplos para aprender:

Utilizando la notación propia de plotly:

```
plot_ly(mtcars, y = ~mpg, color = ~as.factor(cyl), type = "box") %>%
  layout(boxmode = "group")

plot_ly(z = ~volcano, type = "contour")
```

O bien la notación ggplot2 y convertir el gráfico en formato plotly con la función *ggplotly*.

1.3.2. La librería Dygraphs

Esta librería permite utilizar series temporales de una manera más eficiente e interactiva que con la librería ggplot2.

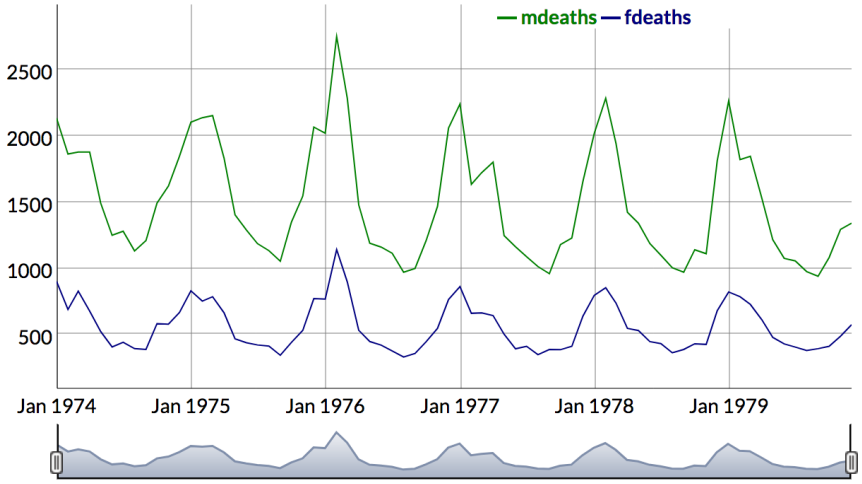
Entre las principales ventajas que ofrece, podemos encontrar:

- Permite la representación directa de objetos tipo series temporales.
- Facilita la configuración.
- Interactividad.
- Integración con generación de informes R Markdown o aplicaciones web vía Shiny.

Enlace de interés

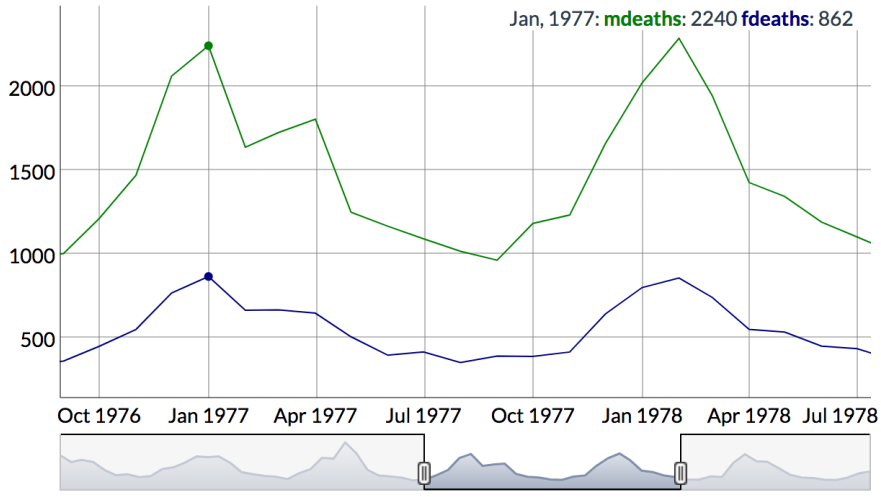
Los gráficos visualizados son uno de los ejemplos que están a nuestra disposición en el siguiente enlace: https://bit.ly/2Xi0zIc

Figura 15. Vista de dos series temporales representadas con la librería Dygraphs



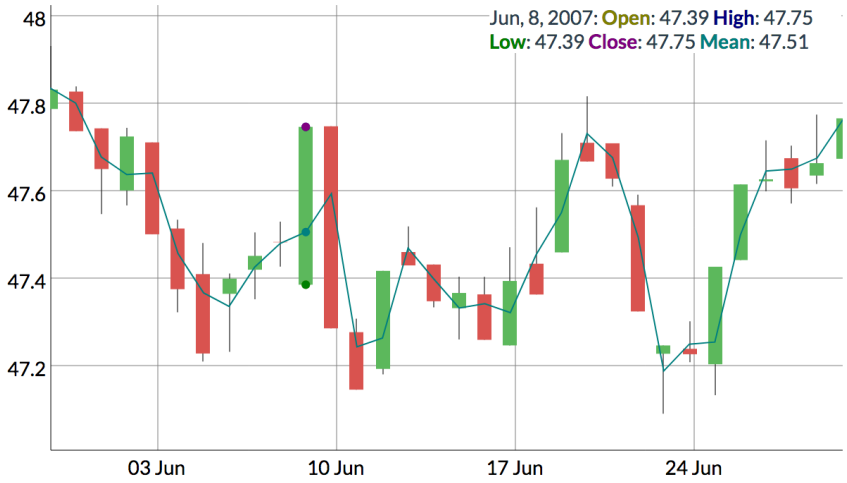
Con la manipulación del panel inferior, se realiza un zoom automático sobre las series que se están mostrando.

Figura 16. Zoom de las series con el panel interactivo inferior



La librería permite otras representaciones que pueden ser apreciadas en el caso de análisis de series vinculadas a cotizaciones en mercados de valores. En la figura 17 podemos apreciar por cada medida vinculada a un día el precio de apertura, el precio de cierre, el máximo, el mínimo y un promedio.

Figura 17. Representación típica de valores cotizados con la librería Dygraphs

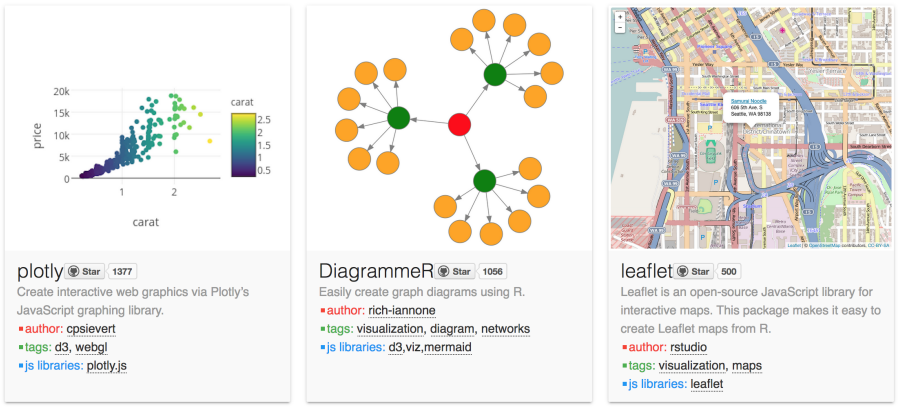


1.4. Otras representaciones gráficas

La ventaja de un entorno como R es que permite utilizar un gran conjunto de soluciones desarrolladas para un fin específico. Por ejemplo, la utilización de una librería para representaciones GIS, para la visualización de información genómica, para la visualización de redes, entre otras.

En la figura 18 se presentan las librerías para la interacción gráfica plotly (ya vista anteriormente), la librería DiagrammeR para la visualización de diferentes diagramas, como árboles, y la librería de mapas interactivos leaflet.

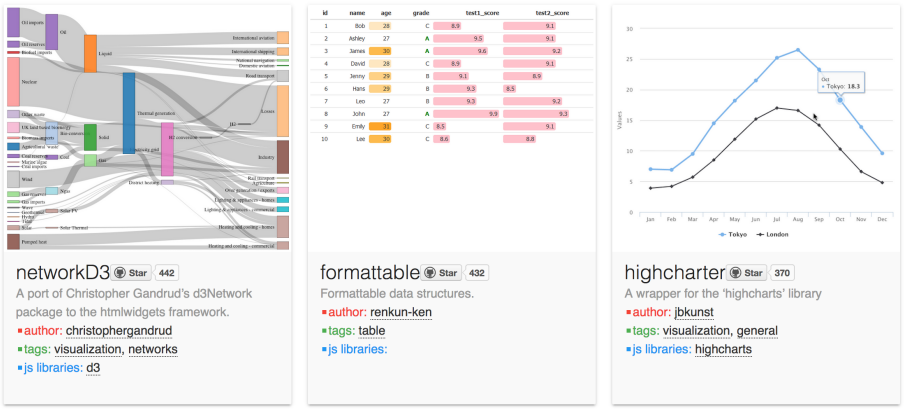
Figura 18. Vista de las librerías plotly, DiagrammeR y leaflet



En la figura 19 se presentan las librerías para la creación de salidas gráficas de redes de datos (representados en un grafo) networkD3, la librería formattable para la mejora de la visualización de salidas en formato de tabla (es utilizada para la generación de informes y/o para uso en *dashboards*) y la librería highcharter de mapas interactivos generales. Esta última librería, que puede asemejarse a plotly, tiene como inconveniente que no utiliza la sintaxis de

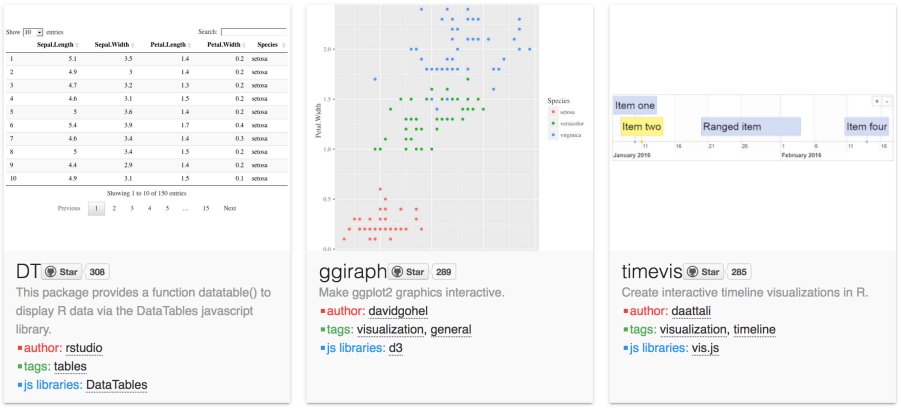
ggplot2 ni permite una traducción directa como sí podíamos realizar con la de plotly.

Figura 19. Vista de las librerías networkD3, formattable y highcharter



En la figura 20 se presentan las librerías para la visualización de tablas interactivas DT y que facilitan la generación de informes dinámicos (formato HTML), la librería ggiraph para la interacción de gráficos vía *grammar of graphics* (ggplot2) y la librería timevis para la visualización de líneas de tiempo interactiva.

Figura 20. Vista de las librerías DT, ggiraph y timevis

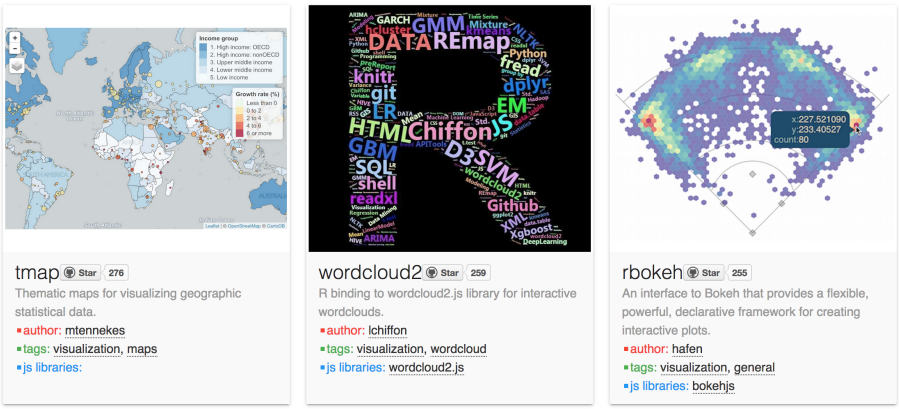


En la figura 21 se presentan las librerías tmap, que facilita la representación de información estadística sobre mapas, la librería wordcloud2 para la salida de gráficos en formato *word cloud*, muy utilizados para las salidas de análisis de *textmining*, y la librería rbokeh para la visualización interactiva de gráficos complejos.

Enlace de interés

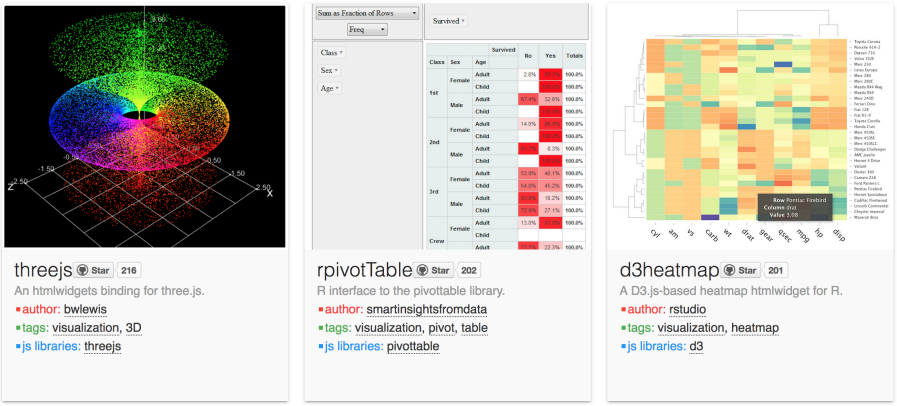
Se puede obtener información detallada en este enlace: https://bit.ly/2E42yXS

Figura 21. Vista de las librerías tmap, wordcloud2 y rbokeh



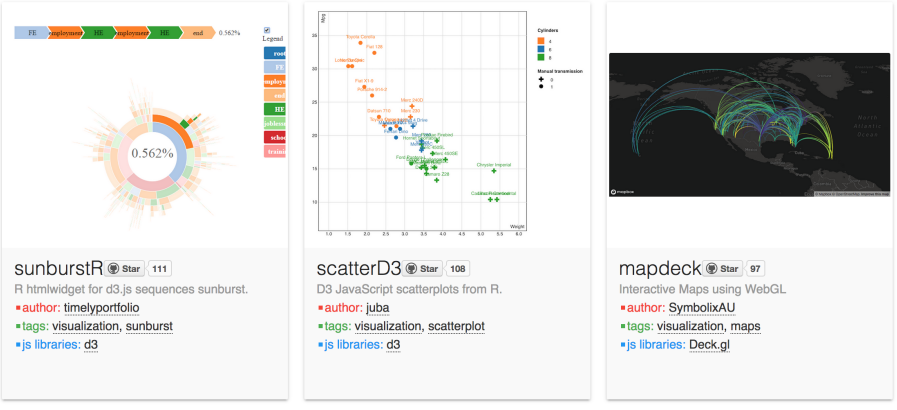
En la figura 22 se presentan las librerías threejs para las representaciones en 3D, la librería rpivotTable para la salida de tablas que pueden pivotar sobre alguna característica categórica y la librería d3heatmap para la visualización de gráficos tipo *heatmap* (mapas de calor).

Figura 22. Vista de las librerías threejs, rpivotTable y d3heatmap



Por último, en la figura 23 se presentan las librerías sunburstR para las representaciones en formato explosiones de sol (complejidad en los datos), la librería scatterD3 para la visualización de diagramas entre dos variables y la librería mapdeck para la visualización de mapas interactivos.

Figura 23. Vista de las librerías sunburstR, scatterD3 y mapdeck



1.4.1. Comentarios finales

Algunas de las librerías tienen la ventaja de ser absolutamente libres, otras únicamente en casos no comerciales o de organismos públicos. Deberemos asegurar qué tipo de licencia vinculada al uso de la librería nos ofrece y, en función de nuestras necesidades, hacer uso de ella o descartarla.

Nota

Todas las imágenes de las librerías mostradas desde la figura 18 hasta la figura 23 provienen de la web: http://gallery.htmlwidgets.org

2. Cuadros de mando. *Dashboard*

2.1. Introducción

Los cuadros de mando, o lo que habitualmente conocemos en terminología anglosajona como *dashboard*, no es otra cosa que una herramienta de gestión habitualmente dirigida a la dirección para visualizar de forma clara una serie de elementos importantes, como KPI, información de algunos datos relevantes para la toma de decisiones o estado de diferentes procesos.

A través de la correcta visualización, la idea detrás del *dashboard* es simplificar la presentación de una realidad compleja del negocio para que, de una simple mirada, tengamos conciencia del estado de elementos importantes de nuestra unidad de negocio.

Podemos entender el *dashboard* como una herramienta imprescindible para la gestión eficiente de la información de una organización.

2.1.1. Características de un cuadro de mando

Podemos encontrar una gran variedad de tipologías de *dashboards* en función del negocio y del propósito de este, si la información es de tipo táctica o estratégica, si el usuario final será un ejecutivo o bien si será un usuario que está en una línea de producción.

Estos elementos y aspectos relativos a la influencia del tiempo (nivel de detalle) permitirán diseñar el *dashboard* de forma efectiva de acuerdo con estas necesidades. Las principales características que debe poseer cualquier *dashboard* son las siguientes:

- **Estructuración de la información**: la información que facilita el *dashboard* proviene en ocasiones de diferentes fuentes o procedencias y puede ser compleja. Esta información debe ser puesta a disposición de los usuarios del cuadro de mando de una forma sencilla.

- **Los elementos facilitan una historia**: cómo publicitan varias empresas y dan importancia a los elementos que configuraran el *dashboard* para que permitan al usuario percibir y entender la realidad del negocio sobre la que tienen responsabilidad.

- **Relevancia de la información**: el cuadro de mando debe trasladar visualmente la relevancia de la información mostrada con el uso de las diferentes técnicas disponibles: color, forma, etc., y esta debe ser fiel a la realidad que representa.

- **Ofrecer la información justa**: no podemos dar más información de la necesaria ni tampoco menos de la que se necesita. En estas situaciones la filosofía del menos es más tiene su razón de ser.

En general, los cuadros de mando quieren ofrecer una visión de gran utilidad a la toma de decisiones. Veamos qué dice la empresa Tableau sobre sus *dashboards*:

> «Tableau allows us to create dashboards that provide actionable insights and drive the business forward. Donald Lay, Senior Manager, Business Intelligence, Charles Schwab.»

Obviamente, no es la única empresa que facilita *dashboard* para sus clientes. La solución que Microsoft ofrece nos cuenta algo similar:

> «A Power BI dashboard is a single page, often called a canvas, that uses visualizations to tell a story. Because it is limited to one page, a well-designed dashboard contains only the most-important elements of that story.»

O la de IBM:

> «With the analytics dashboard, you can build sophisticated visualizations of your analytics results, communicate the insights that you've discovered in your data on the dashboard and then share the dashboard with others.»

2.1.2. Consideraciones de diseño

Al ser una aplicación software, el procedimiento para el diseño de estas soluciones es equivalente al que podamos encontrar en aproximaciones de ingeniería de software. De una forma simplificada:

- Definir los roles y usuarios de la solución.
- Identificar las métricas clave KPI.
- Aplicar para la solución un buen enfoque GUI que permita potenciar el mensaje y la información que este deba transmitir.
- Utilizar sistemas de visualización que no sean ambiguos o que den pie a interpretaciones erróneas.

Como toda solución software, se debería garantizar que la solución ha sido testeada, evaluada correctamente y puesta en las condiciones en que operará.

2.2. Soluciones cerradas

Entendemos por solución cerrada aquella que permite al usuario prescindir por completo de elementos de programación y habitualmente ofrece herramientas de diseño del *dashboard* dirigidas para facilitar su creación.

Se pretende evitar que el mánager tenga que interactuar con el gestor de la base de datos o con un programador para realizar este tipo de configuración. Por regla general, estas soluciones suponen unos costes.

En este subapartado presentamos algunas de las posibles soluciones cerradas disponibles para la industria:

- IBM: https://ibm.co/2Elekyn
- Microsoft: https://www.edureka.co/blog/power-bi-dashboard/
- SAP: https://www.tutorialspoint.com/sap_dashboards/
- Tableau: https://www.tableau.com/

En la figura 24 podemos ver la publicidad de la empresa Tableau. Y en las posteriores representaciones un ejemplo de *dashboard* con dos vistas; en la figura 25 para la visualización de KPI relativos a las empresas que generan más beneficios para el grupo y una representación gráfica. Y en la segunda vista (ver figura 26) del mismo *dashboard*, información adicional sobre la tipología de los envios según sean buenos, correctos o malos.

Figura 24. Publicidad de la empresa Tableau con referencia a los cuadros de mando

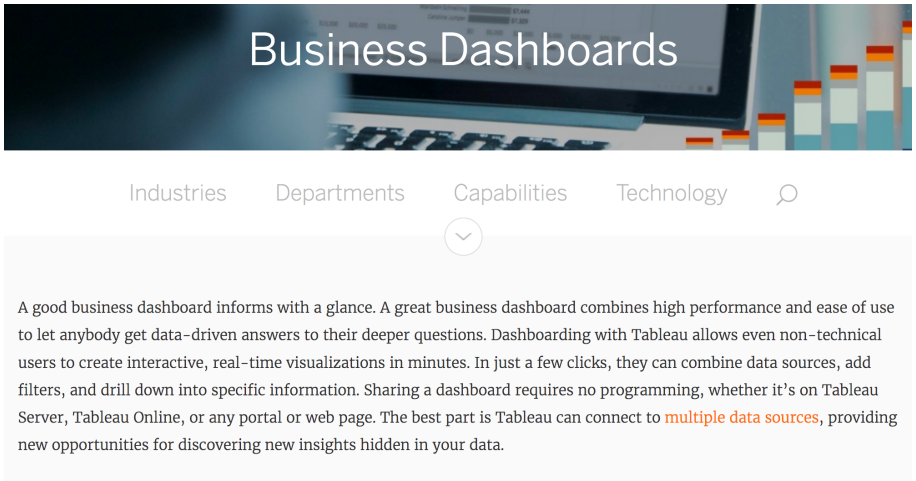


Figura 25. Ejemplo de *dashboard* de RStudio llamado pokemon

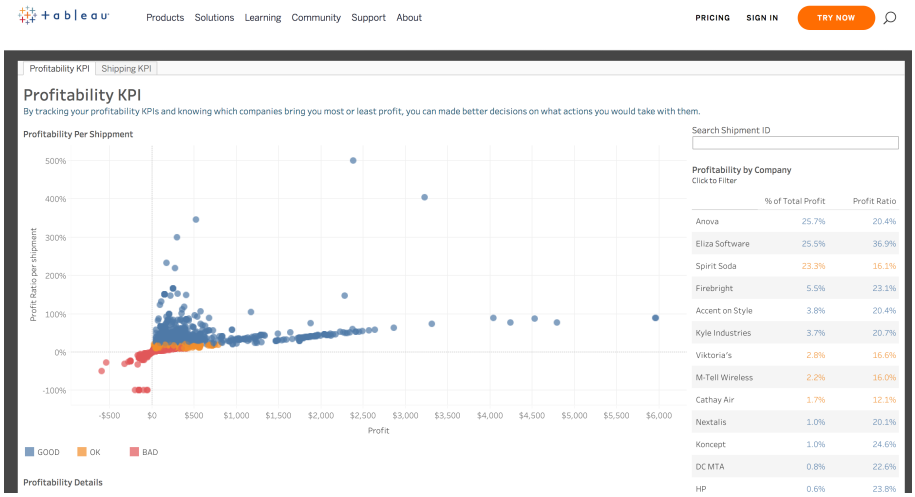
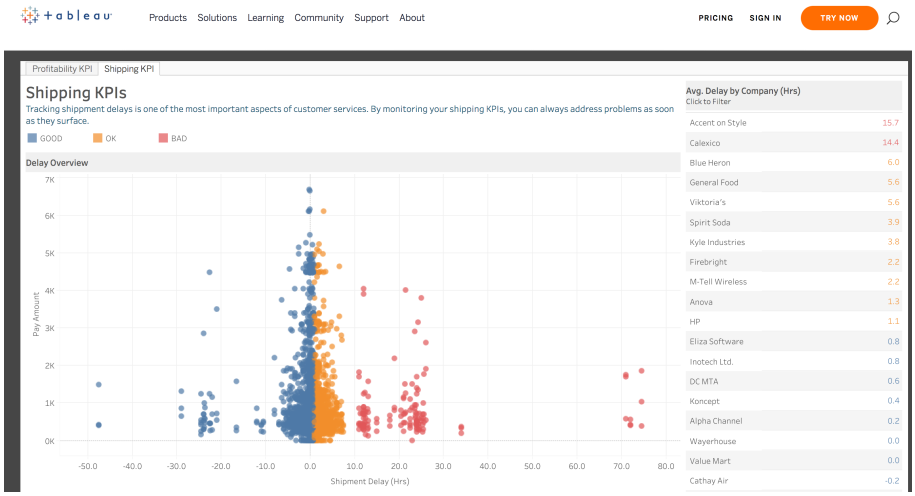


Figura 26. Ejemplo de *dashboard* de RStudio llamado pokemon



2.3. Shiny App

Una de las opciones que ofrece R para desarrollar *dashboards* es a través de su *framework* de aplicaciones web con el nombre de Shiny. Este tipo de soluciones permite una gran flexibilidad y entre una de sus utilidades está la de ofrecer *dashboards*.

Desde una perspectiva objetiva, Shiny es un paquete R de código abierto que proporciona un marco web elegante y poderoso para crear aplicaciones web utilizando R. La idea es crear entornos analíticos como aplicaciones web interactivas sin necesidad de conocimientos de HTML, CSS o JavaScript.

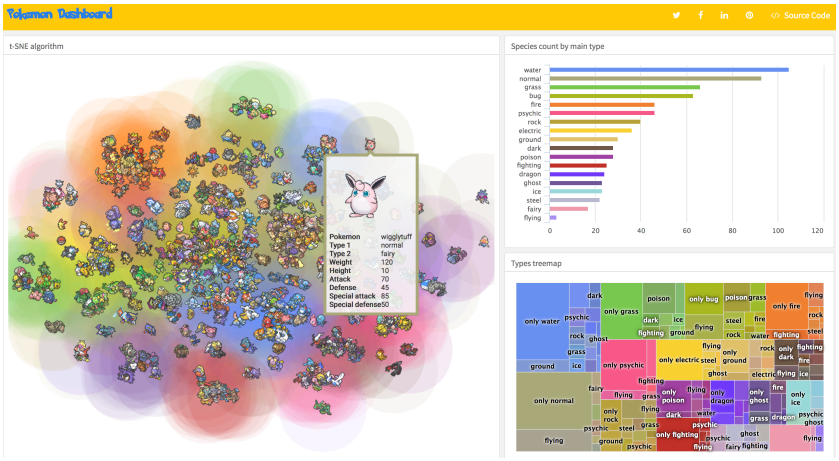
Enlace de interés

Podéis ver algunos ejemplos en el siguiente enlace: https://bit.ly/2E2obb4

2.3.1. Ejemplos de *dashboards* con Shiny

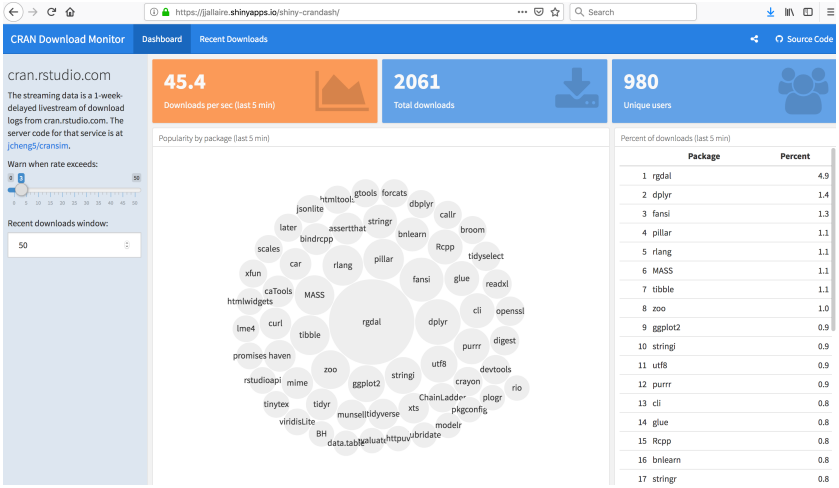
Si bien la codificación de algunos de estos ejemplos puede ser compleja, la idea es mostrar algunos ejemplos:

Figura 27. Ejemplo de *dashboard* de RStudio llamado pokemon



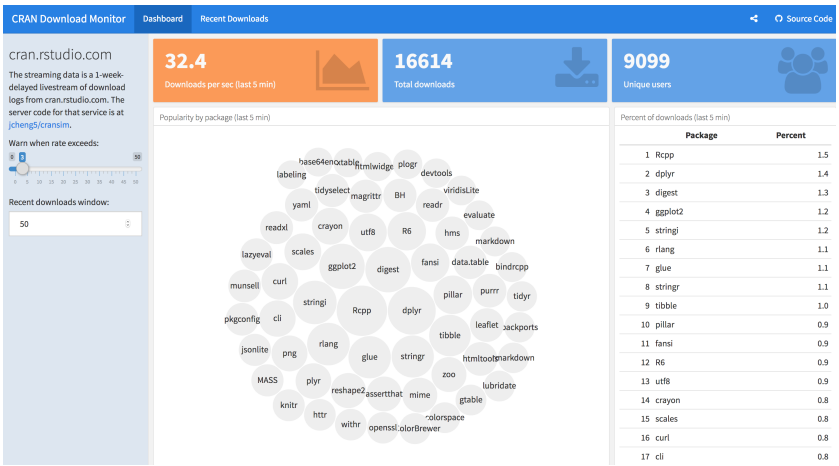
Fuente: https://rmarkdown.rstudio.com/flexdashboard/examples.html

Figura 28. Ejemplo de *dashboard* de RStudio Estado de descargas de librerías R (instante 1)



Fuente: https://rmarkdown.rstudio.com/flexdashboard/examples.html

Figura 29. Ejemplo de *dashboard* de RStudio Estado de descargas de librerías R (instante 2)



Fuente: https://rmarkdown.rstudio.com/flexdashboard/examples.html

Desde R hay dos aproximaciones posibles:

- *Shinydashboard*, que es la opción habitual de código Shiny, no del todo fácil, dinámico y que utiliza *bootstrap*.

- *Flexdashboard*, que es una opción que utiliza la sintaxis de R markdown, muy fácil de utilizar, con opciones de salida estática o dinámica.

2.4. Conclusiones

Enlaces de interés

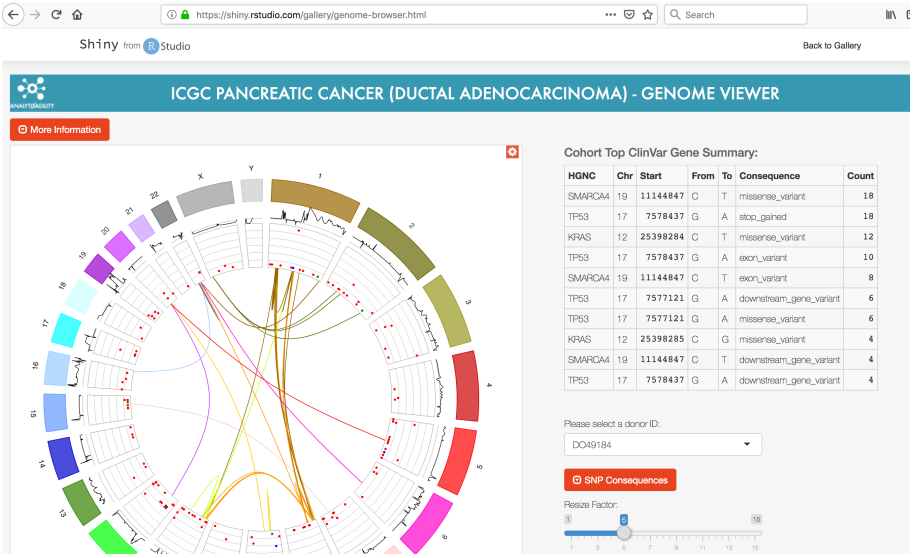
Para más información podéis consultar los siguientes enlaces:
https://bit.ly/2Ni1Drf
https://bit.ly/2jhiADJ
https://bit.ly/2GCmQf0

La mejor opción para desarrollar un cuadro de mando para nuestro negocio dependerá de nuestros requerimientos, los costes asociados a la solución y las garantías que se deban cumplir (seguridad, tiempo de respuesta, etc.).

Entendiendo que incluso la opción de RStudio o Python tienen un coste asociado, ya que se debe utilizar código, muchas soluciones de empresa no van a ofrecer nada que estas dos soluciones no puedan ofrecer.

Obviamente, cada organización puede ponderar aquello que considere más relevante para la decisión. En ocasiones, el elemento determinante puede ser que la solución esté en la nube o como app accesible para móvil. En otros casos, que la solución se ajusta a la especialidad que el departamento o la unidad de negocio requiera (como la representada en la figura 30 para datos de ámbito genómico.

Figura 30. Ejemplo de *dashboard* de RStudio para datos genómicos



Fuente: https://rmarkdown.rstudio.com/flexdashboard/examples.html

3. Herramientas de *reporting*

3.1. Introducción

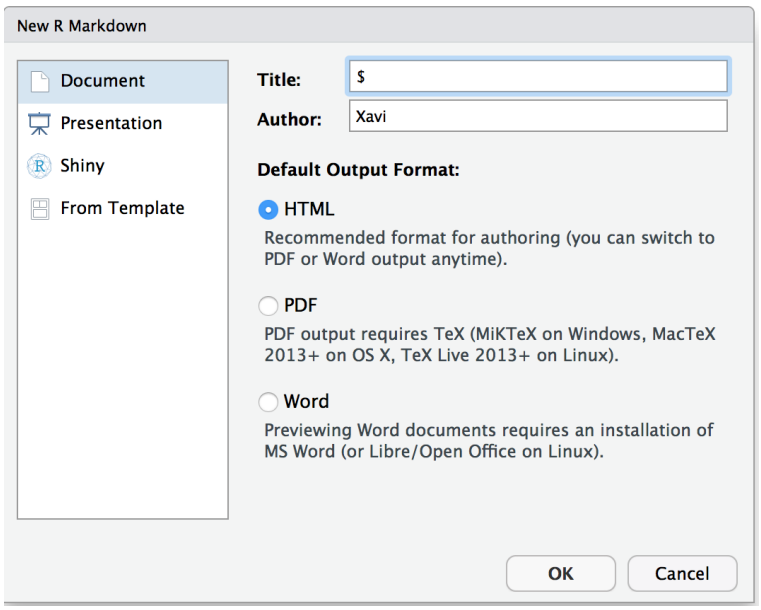
En este apartado vamos a tratar de las diferentes herramientas disponibles para sacar provecho de la visualización, así como las que permiten analizar los datos y se integran de una forma interesante con los documentos R markdown.

Los documentos R markdown permiten editar un documento donde podremos incluir tanto el código de R como las partes de texto que deseemos que figuren en el documento que debemos desarrollar. Esta combinación permite una gran versatilidad, flexibilidad y potencia.

Para poder ver los ejemplos de una forma sencilla, se recomienda abrir un nuevo proyecto. Una vez dentro del proyecto, podemos crear un primer documento R markdown (.Rmd) (ver figura 31). Al crear un nuevo documento de tipo Rmd, RStudio no indica cuál va a ser el tipo de documento de salida. Nos ofrece tres opciones:

- Documento HTML
- Documento PDF (necesita el compilador de LaTeX)
- Documento Word

Figura 31. Opciones disponibles para el formato de salida: HTML, PDF o Word



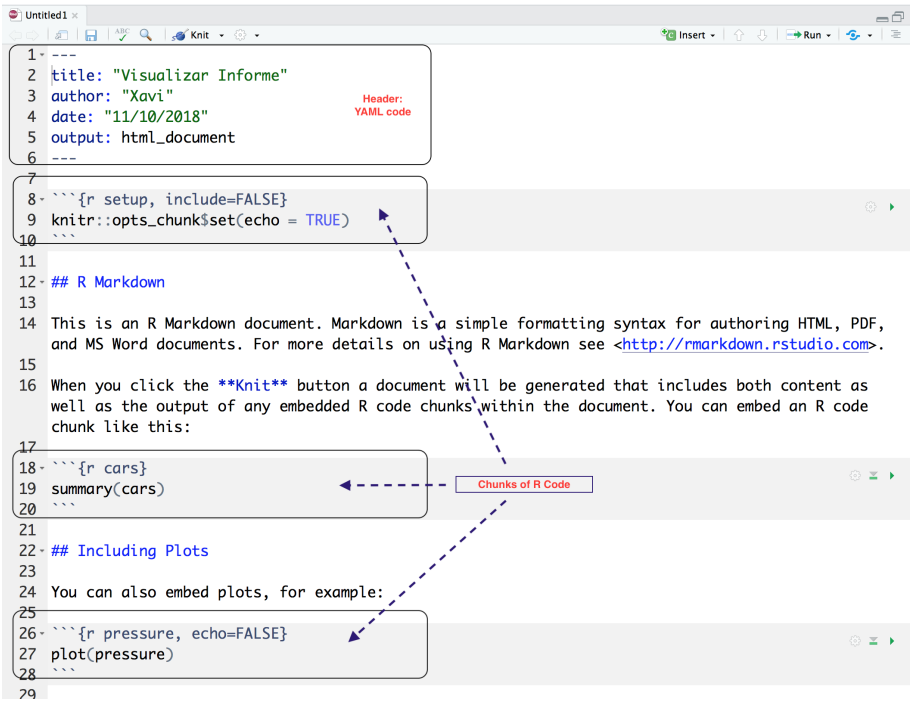
3.2. Estructura de un fichero R markdown

Una vez que tenemos definido el tipo de fichero de salida, podemos describir los detalles y características de este.

El conjunto de elementos que podemos visualizar son los siguientes. Primero, la cabecera del fichero que está en formato YAML. En este apartado se definen los aspectos generales del documento, como nombre del autor, título del documento y fecha, entre otros. A continuación tenemos un conjunto de elementos que podemos clasificar en texto con sintaxis R markdown y trozos de código que insertamos en las posiciones del documento donde se espera obtener algún resultado para visualizar.

La ventaja de este tipo de documentos es que facilitan el seguimiento de los pasos realizados en el análisis y la generación del informe final.

Figura 32. Estructura de un fichero .Rmd



La descripción de la sintaxis que puede contener los trozos de código depende de nuestro interés. Es decir, el código puede ser en R, en Python, en Rcpp, en SQL o en Stan. Las partes de código van identificadas entre tres acentos abiertos seguidos de unas llaves que especifica características de cómo ha de ser ejecutado y visualizado el código. La parte más simple y con mayor potencial es la sintaxis asociada a R MArkdown.

3.2.1. Trozos de código

Las opciones más usadas y comunes son las siguientes:

- **eval**: si es cierto, el código se ejecuta y visualiza.
- **echo**: si es cierto, el código se visualiza.
- **warning**, **error**, **message**: si es cierto, se muestran las advertencias, los errores o los mensajes que genere el código ejecutado.
- **results**: cómo se muestran los resultados.
- **fig.width**, **fig.height**: dimensiones de los gráficos que se generen.

Por defecto, el código se introduce (*echo*), a continuación se evalua (*eval*) y por último se añade la salida (*include*).

Partes de los cálculos realizados en estos trozos de código pueden añadirse al documento principal utilizando la siguiente sintaxis:

```
`r <expression>`
```

Incorporación de mapas interactivos

Las salidas de código pueden generar gráficos interactivos, si por ejemplo se utiliza la librería leaflet. Se puede observar en las figuras 33 y 34.

Figura 33. Visualización de mapa interactivo 1 .Rmd

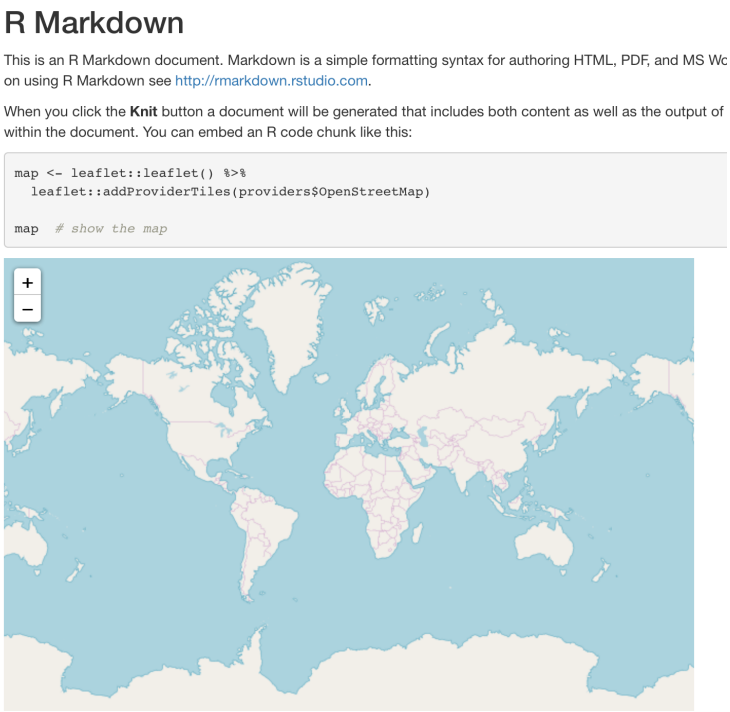
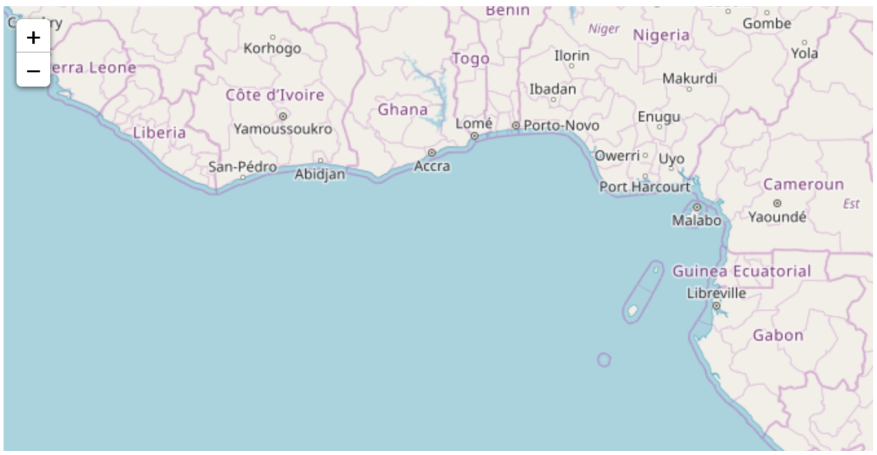


Figura 34. Visualización de mapa interactivo 2 .Rmd

```
map <- leaflet::leaflet() %>%
  leaflet::addProviderTiles(providers$OpenStreetMap)

map  # show the map
```



Incorporación de tablas interactivas

El uso de la librería adecuada, como la DT, posibilita generar salidas en formato HTML que permiten la interacción sobre la tabla generada.

```
library(DT)
datatable(iris, options = list(
  searching = FALSE,
  pageLength = 5,
  lengthMenu = c(5, 10, 15, 20)
))

# global search
datatable(iris, options = list(searchHighlight = TRUE,
 search = list(search = 'setosa')))
```

Incorporación de tablas fijas

Si el documento que estamos diseñando tiene que ser flexible en cuanto a garantizar su paso por un documento Word o PDF, debemos prescindir de los elementos interactivos. Hay opciones de visualización interesantes que permiten pasar a los diferentes formatos.

Figura 35. Visualización de tablas interactivas .Rmd

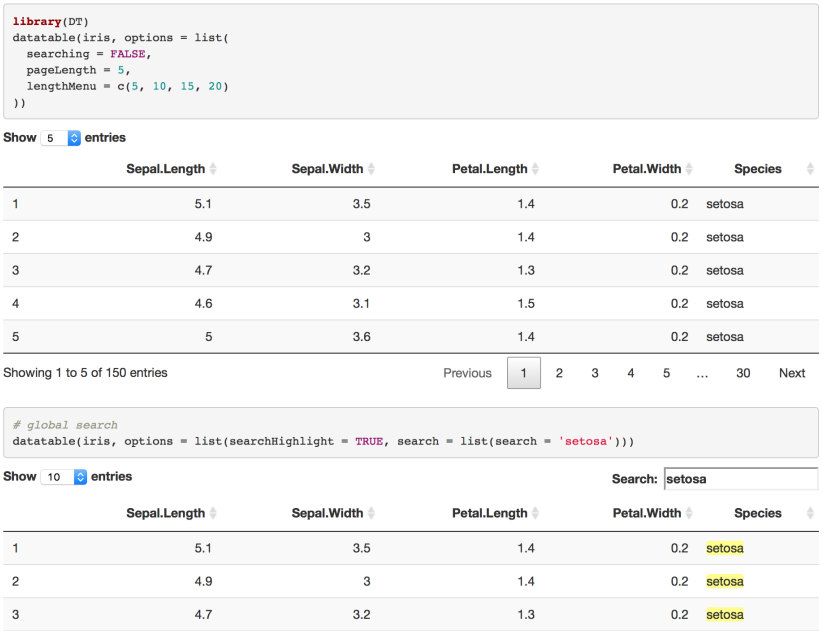
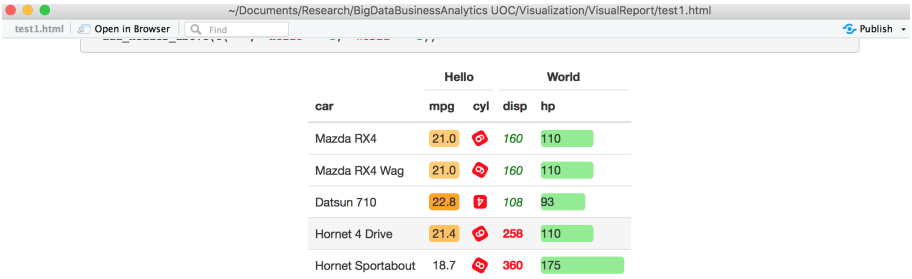


Figura 36. Visualización de tabla estática con elementos para resaltar el aspecto



3.2.2. R markdown

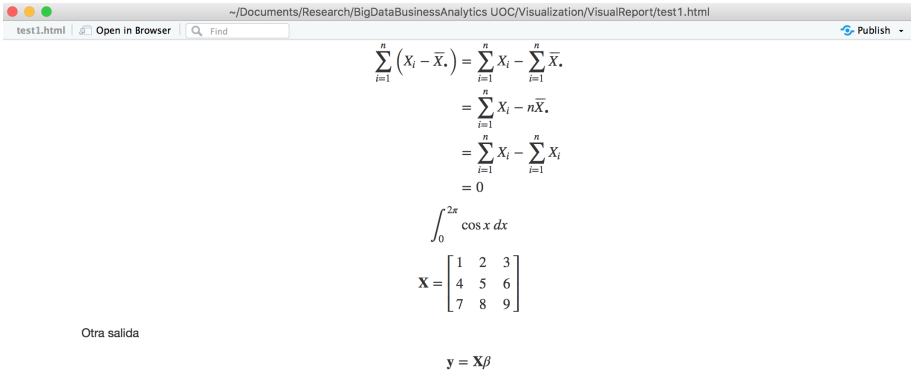
Sin entrar en un detalle exhaustivo de las posibilidades que nos brinda esta sintaxis, sí que conviene destacar su simplicidad. Se describen aquellos elementos que ayudan a dar formato a nuestro documento.

- Cabeceras: utilizamos el carácter de hastag #
- URL: [link](www.uoc.es)
- Incluir ficheros gráficos: ![R Logo](figures/RLogo.png)
- Negrita: texto entre dos asteriscos **
- Cursiva: texto entre un asterisco *

La incorporación de la notación LaTeX sirve para escribir formulación matemática. La renderización será correcta independientemente del fichero de salida que utilicemos. Algunos ejemplos de notación matemática se observan en la figura 37.

```
$$\mathbf{X} = \left[\begin{array}
{rrr}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array}\right]
$$
```

Figura 37. Visualización de notación matemática en el documento .Rmd



3.3. Generación de documentos de salida

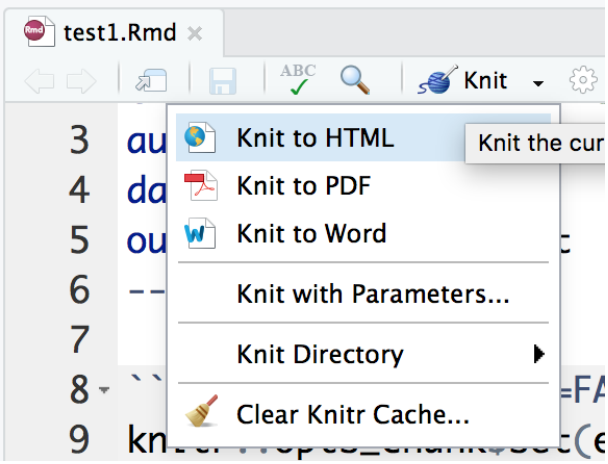
La flexibilidad de R markdown permite que, bien a través de la cabecera del documento YAML en :

```
output:
  html_document
```

o bien a través del botón de Knitr, en la barra de botones del IDE de RStudio, podamos decidir qué formato de salida generamos. Podéis ver la figura 38 y las opciones que tenemos en la figura 39.

Figura 38. Botón de generación de documento de salida knitr

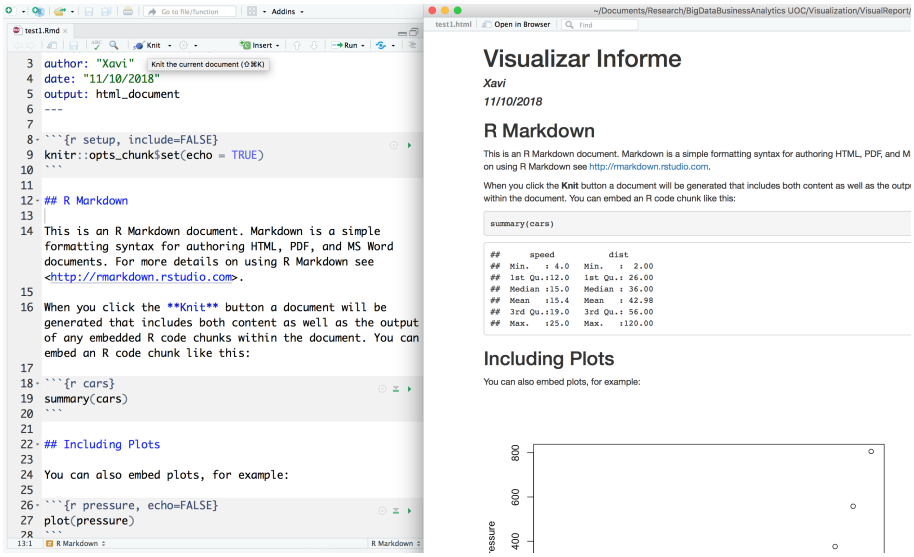
Figura 39. Botón de generación de documento de salida knitr con la lista de posibles opciones



3.3.1. Salida HTML

La salida HTML permite visualizar el documento en cualquier navegador. Ved la figura 40.

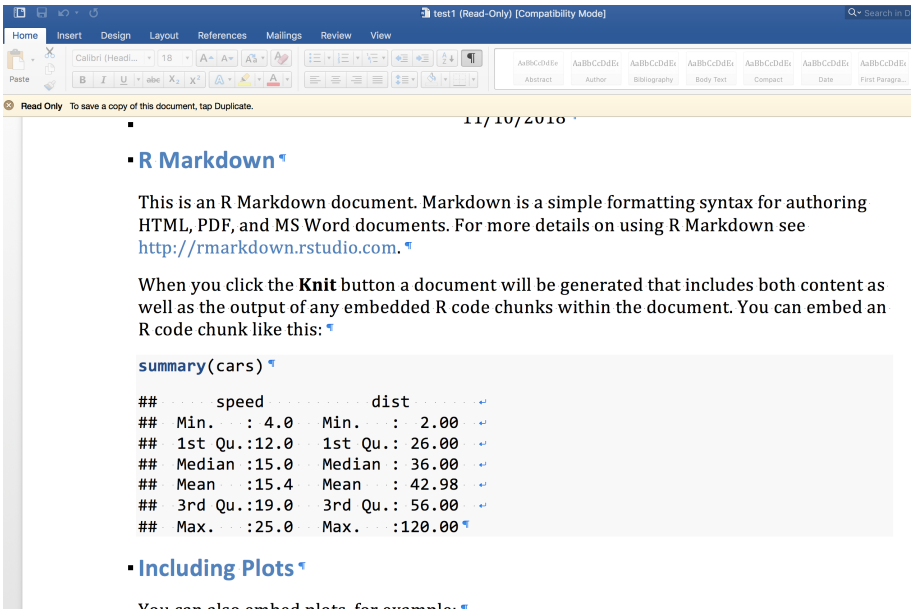
Figura 40. Ejemplo de salida HTML directamente del fichero por defecto



3.3.2. Salida WORD

La salida WORD permite visualizar el documento en el típico formato de Office y que otros aplicativos también pueden leer. Ved la figura 41.

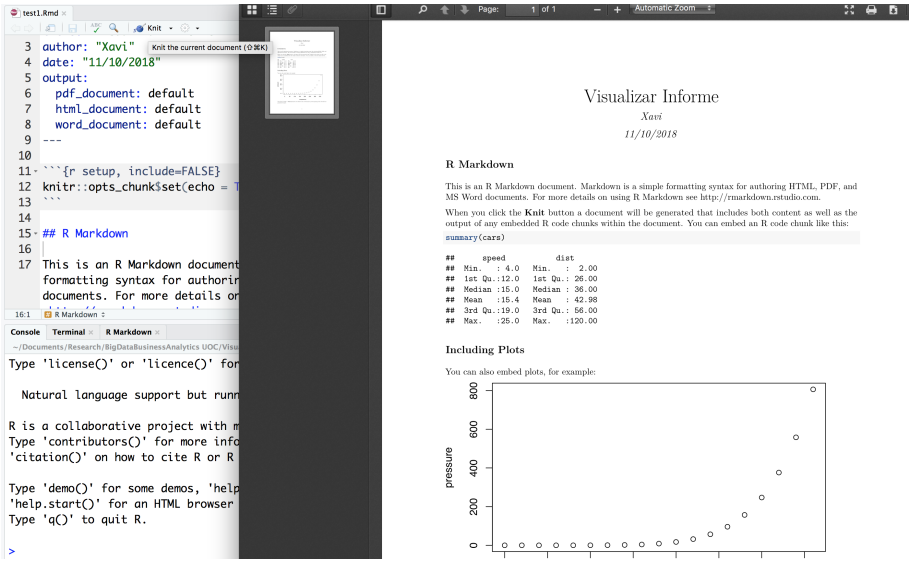
Figura 41. Ejemplo de salida WORD directamente del fichero por defecto .Rmd



3.3.3. Salida PDF

La salida PDF permite visualizar el documento en cualquier visor de documentos PDF. Ved la figura 42.

Figura 42. Ejemplo de salida PDF directamente del fichero por defecto .Rmd



3.4. Información adicional

Una exposición detallada de estas características se puede obtener en los siguientes enlaces:

https://bookdown.org/yihui/rmarkdown/pdf-document.html

https://rmarkdown.rstudio.com/
https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-spanish.pdf

Otra de las ventajas de la utilización de R markdown es que podemos generar un conjunto de diapositivas en diferentes formatos. Con la misma sintaxis generamos una presentación.

Al generar un nuevo documento R markdown la opción por defecto es Document (ved figura 31), pero podemos elegir la opción Presentation. Si seleccionamos esta opción, podemos decidir qué tipo de documento de salida queremos:

- HTML ioslides
- HTML slidy
- PDF beamer

Realizad un ejemplo con alguna de estas opciones.

Resumen

¿Por qué las buenas decisiones se toman con buenos gráficos? Sabemos que los datos no aportan ninguna información a menos que se representen gráficamente y sean capaces de explicar una historia. Explicar las historias de los datos es el objetivo principal del contenido de esta unidad, donde se pone de manifiesto la famosa frase:

> Una imagen vale más que mil palabras.

El uso de la librería ggplot facilita la creación de gráficos utilizando la gramática gráfica que permite añadir capas a los gráficos de una manera intuitiva. La librería ofrece una gran variedad de *geoms*, algunos vistos y otros no. Cada uno de estos *geoms* tiene una serie de usos que debemos conocer para aplicar correctamente. El uso del *geom_bar* es para variables categóricas, mientras que el *geom_histogram* se utiliza para variables continuas. Hay *geoms* para dos variables continuas, como *geom_point* o el *geom_text*, y otros que combinan una variable discreta con una continua, como es el caso de *geom_boxplot*.

Una característica apreciada de ggplot2 es el hecho de disponer de una librería adicional denominada ggthemes, que permite añadir a nuestro gráfico los elementos de estilo definidos por diferentes medios de comunicación. Por ejemplo, podemos dar a nuestro gráfico el mismo look que tienen los gráficos de *The Economist* o el que utiliza *The Wall Street Journal*, entre otros muchos.

El universo de librerías de representación gráfica de R es muy grande y cubre cualquiera de nuestras necesidades. Únicamente necesitamos buscar la solución a nuestra necesidad.

En otro contexto, el desarrollo de soluciones para la utilización o creación de cuadros de mando pasa por el entorno Shiny como *framework web* de R. Obviamente, hay otras aproximaciones que también deben ser valoradas por su gran potencial y facilidad.

Es también un elemento imprescindible el ser capaces de generar documentos donde aparece el texto junto al código incrustado. Esta opción permite facilitar la verificación y/o transmisión de estos informes (estudios) a otros investigadores o profesionales.

Por último, el uso de desarrollo de software a través del IDE RStudio para *dashboards* u otras soluciones debería hacerse bajo alguna herramienta de VCS

(*version control systems*), como Github. Rstudio incorpora esta posibilidad de forma casi transparente en los proyectos R.

Bibliografía

Cairo, A. (2016). *The Truthful Art: Data, Charts, and Maps for Communication*. Ed. New Riders.

Nussbaumer Knafli, C. (2015). *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Ed. Wiley.

RStudio. *R markdown*.

Tufte, E. R. (1989). *The Visual Display of Quantitative Information*.

Wexler, S.; Shaffer, J.; Cotgreave, A. (2017). *The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios*. Ed. Wiley.

Wickham, H. (2016). *ggplot2. Use R!* Springer International Publishing, Cham.

Wickham, H.; Grolemund, G. (2017) *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data Title*. O'Reilly Media.

Wilkinson, L. (2005). *The Grammar of Graphics (Statistics and Computing)* (2 ed.) Springer.

Wong, D. M. (2013). *The Wall Street Journal Guide to Information Graphics: The Dos and Don'ts of Presenting Data, Facts, and Figures*. Ed. W. W. Norton & Company.

Xie, Y.; Allaire, J.; Grolemund, G. *R markdown the definitive guide*.