

Fastbook 08

Visual Analytics

Gráficos comparativos



08. Gráficos comparativos



En este fastbook vamos a introducir una serie de gráficos muy llamativos visualmente y que nos ayudarán a realizar múltiples comparaciones entre observaciones y numerosas variables numéricas. Como veremos, algunas de estas visualizaciones sufren de una larga lista de quejas: veremos alternativas que solventan algunos de estos problemas. En cualquier caso, como en todos los que hemos visto ya, está en poder del analista la elección del gráfico y estos ejemplos son cartas que debes tener en tu mano obligatoriamente.

De nuevo, utilizaremos R. Te recuerdo que lo idóneo es que piques el código y pruebes todo lo posible. ¡Equivocarse y experimentar es parte del aprendizaje también!

Autor: Daniel Pegalajar Luque

[Gráficos de radar o araña: descripción](#)

[Gráficos de radar en R](#)

[Rosa de Nightingale: descripción](#)

[Rosa de Nightingale en R](#)

[Conclusiones](#)

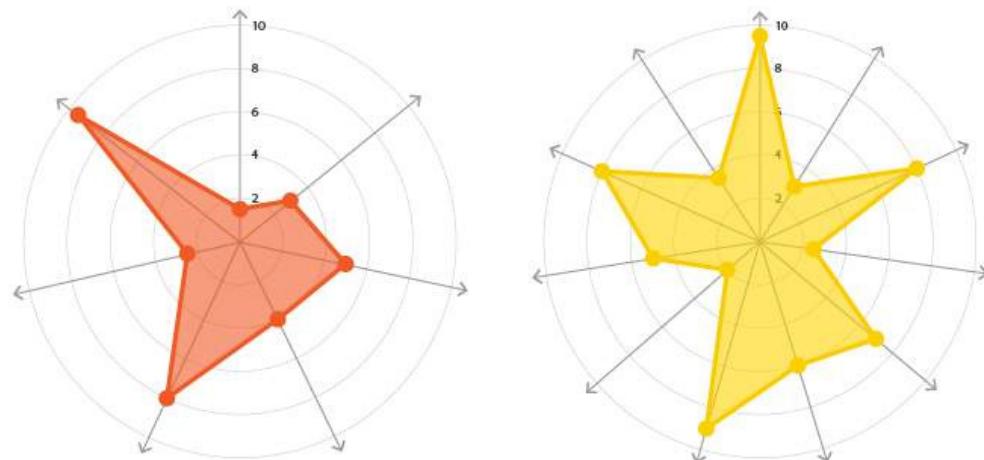
[Bibliografía](#)

Gráficos de radar o araña: descripción

X Edix Educación

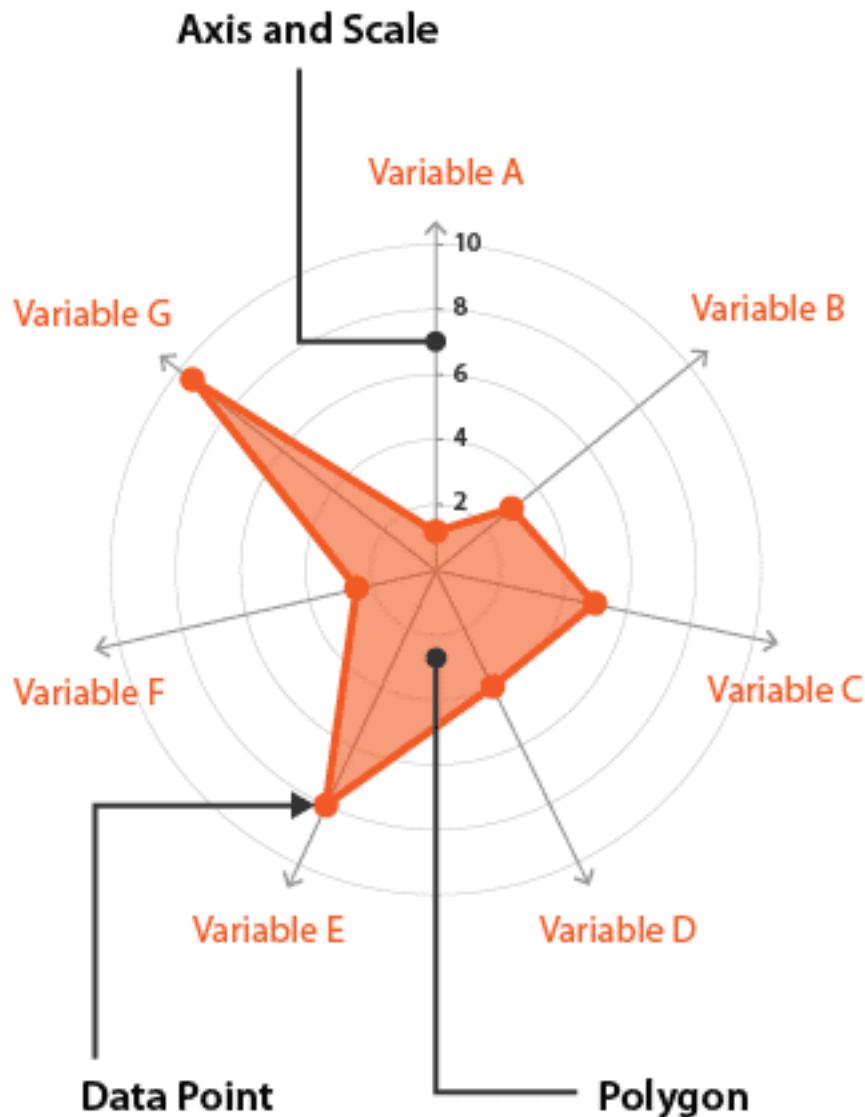
Un gráfico de radar es aquel que permite visualizar una serie de valores numéricos sobre un set común de múltiples variables cuantitativas. Cada una de esas variables cuenta con su propio eje y se unen en un punto común, el centro de la figura.

Son una **herramienta comparativa muy potente** y permiten detectar atípicos en alguna de las variables con facilidad. El uso más frecuente que se le da a este gráfico es para mostrar el **rendimiento** de una persona en un área (podríamos usarlo para comparar las notas obtenidas en las asignaturas ya superadas...).



Fuente: [DataVizCatalogue](#).

La lectura es sencilla: podemos apoyarnos en la imagen que detalla la ‘anatomía’ del gráfico. Cada una de las **flechas** simboliza una **variable** (asignatura, por ejemplo) y partiendo desde el centro se van generando **radios igualmente distribuidos** que contienen los valores que tomarán las variables. El corte entre radio y eje de variable sirve de guía para leer la información. En cada eje, se marca con un **punto** el valor que tiene un sujeto en dicha variable para, finalmente, conectar todos los puntos generando un **polígono**.



Fuente: [DataVizCatalogue](#).

Es necesario que sepas que estos gráficos **son habitualmente criticados** por el mal uso que se suele hacer de ellos. A continuación, detallamos algunos **consejos y errores a evitar**:

- Ten mucho cuidado con la **escala de tus variables**. Si son diferentes, sería recomendable que aplicases una normalización 0-1 para evitar gráficos inservibles.
- Hay que tener **precaución si se pintan múltiples polígonos**, sobre todo si están rellenos, ya que el superior puede tapar parcialmente al resto. Además, usar demasiadas variables dará lugar a numerosos ejes, haciendo que la legibilidad del gráfico se degrade.
- Su **base circular es normalmente difícil de leer**, la simpleza de un solo eje ofrece mayor facilidad para comparar y leer rápidamente los datos. En este sentido, un gráfico de **barras** o un **lollipop** sobresalen más; este último, además, ofrece la ventaja del ranking lo que hace que sea instantáneo detectar la variable clave.
- El **orden de las variables puede generar confusión**. El lector habitualmente se centra en la forma que toma el polígono, pero esta puede cambiar muchísimo solo alterando el orden de las variables, lo ilustraremos en un ejemplo.

Gráficos de radar en R

X Edix Educación

Para comenzar con estos gráficos en R, necesitamos instalar [ggradar](#), una librería complementaria que trabaja sobre ggplot.

Descarga [aquí](#) los datos con los que vamos a trabajar.

```
## Instalamos la versión de desarrollo usando la librería devtools
# devtools::install_github("ricardo-bion/ggradar")

library(ggradar)

# Cargamos los datos
poke ← read_csv("data/pokemon.csv")

poke ← poke %>%
  select(name, hp, attack, defense, sp_attack, sp_defense, speed, base_total, type1, type2, generation,
is_legendary)

poke %>% head()
```

Para esta ocasión, vamos a utilizar un conjunto de datos que puede traer nostalgia a muchos de vosotros. Se trata de un dataset completo de [pokémons](#) con información completa sobre sus estadísticas y que nos va a permitir utilizar perfectamente este tipo de visualizaciones.

Si has cargado todo correctamente, asegúrate de que tienes la columna ‘name’ y las columnas de estadísticas:

name	hp	attack	defense	sp_attack	sp_defense	speed	base_total	type1	type2
Bulbasaur	45	49	49	65	65	45	318	grass	poison
Ivysaur	60	62	63	80	80	60	405	grass	poison
Venusaur	80	100	123	122	120	80	625	grass	poison
Charmander	39	52	43	60	50	65	309	fire	NA
Charmeleon	58	64	58	80	65	80	405	fire	NA
Charizard	78	104	78	159	115	100	634	fire	flying

Una vez tenemos esto, podemos comenzar a dibujar:

```
poke %>%
  ## Nos quedamos con solo la primera generación
  filter(generation == 1) %>%

  ## Reescalamos las variables para que la escala sea comparable
  mutate_at(vars(hp:base_total), list(rescale)) %>%

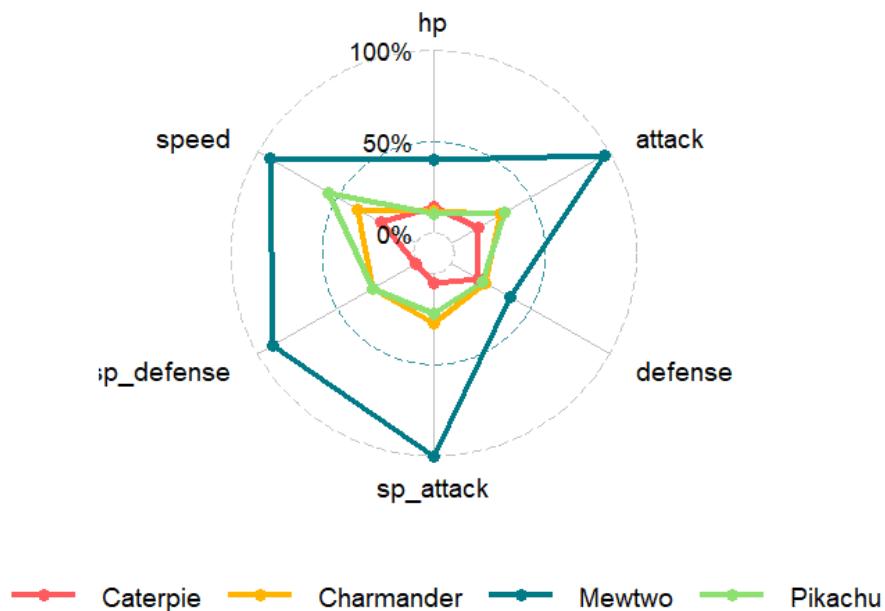
  ## Filtramos aquellos Pokémons que nos interesen
  filter(name %in% c("Pikachu", "Caterpie", "Mewtwo", "Charmander")) %>%

  ## Elegimos las columnas adecuadas
  select(1:7) %>% rename(group = name) → first_radar

  ## Pintamos los resultados
  ggradar(first_radar, group.point.size = 3,
          background.circle.colour = "white",
          legend.position = "bottom")
```

Es importante reescalar los datos en este tipo de visualizaciones. He seleccionado alguno de los pokémons más conocidos para comparar su potencial. Es clave que la columna que indica el nombre sea renombrada a ‘group’, ya que es un requisito de la función ggradar.

En la función, solo es necesario incluir el subdataset que hemos generado y algunos argumentos estéticos, como el tamaño del punto o la posición de la leyenda, para facilitar la lectura de la visualización.



Se puede apreciar que, por defecto, esta aplicación no rellena los polígonos para evitar problemas en la comparativa. En la foto tenemos tres pokémons iniciales o de la fase inicial del juego vs. Mewtwo, que es uno de los definitivos y más poderosos (queda claro observando su polígono). Entre los que no son legendarios, podemos destacar a Charmander y Pikachu, muy por encima de Caterpie. Ambos son muy parecidos, pero gana Pikachu por velocidad y Charmander lo supera en 'Ataque especial'.

Cuando quieres comparar múltiples ítems como antes, una solución si estás teniendo problemas para leer los resultados es utilizar los **facets**:

```
## Pintamos los resultados
ggradar(first_radar, group.point.size = 3,
        background.circle.colour = "white",
        legend.position = "none") +
  facet_wrap(~group)
```

Ahora la comparativa es uno a uno y ‘facilita’ la lectura. En cualquier caso, este tipo de visualizaciones suele recibir muchas críticas como hemos comentado antes.

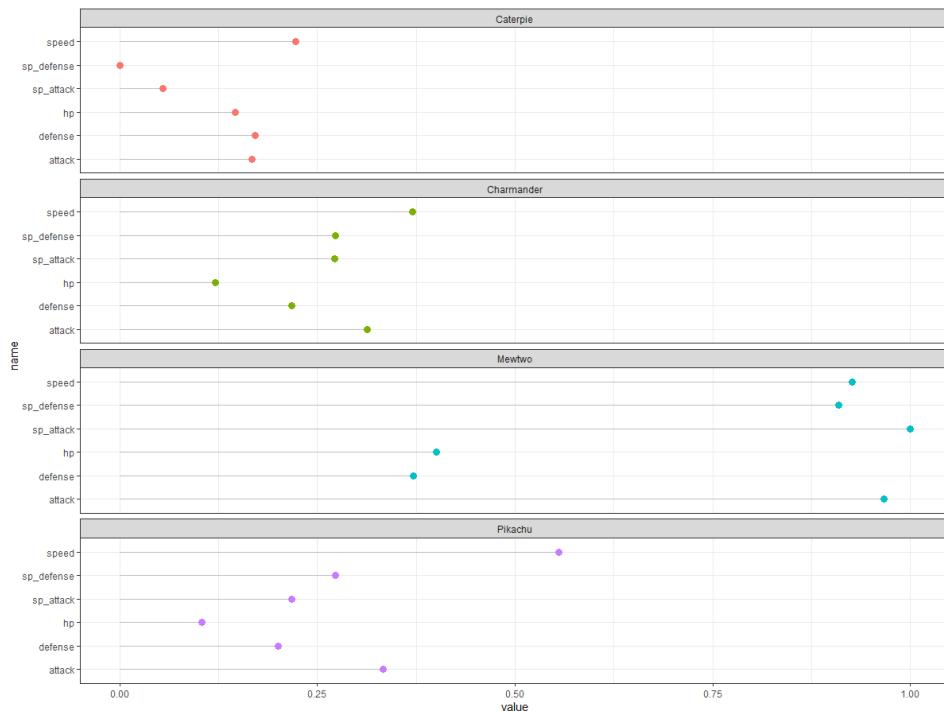


Ilustremos algunos casos para demostrar esto.

La lectura de un gráfico radar es relativamente difícil, sobre todo cuando tienes que comparar casos similares sobre la base radial que utiliza. La alternativa a esto ya la conoces, *lollipop charts*.

```
first_radar %>%
  pivot_longer(cols = hp:speed) %>%
  ggplot( aes(x=name, y=value, col = group)) +
  geom_segment( aes(x=name ,xend=name, y=0, yend=value), color="grey") +
  geom_point(size=3) +
  coord_flip() +
  facet_wrap(~group, ncol = 1) +
  theme(legend.position = "none")
```

A la visión humana le resulta más sencillo ubicar diferencias en un eje lineal de este tipo, ¿qué opinas?

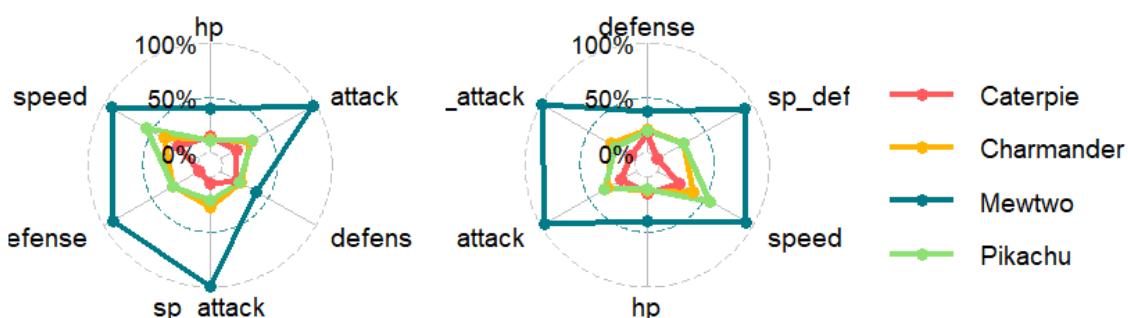


El orden de las variables elegido es determinante para la forma del polígono final, esto no puede parecer un gran hándicap, pero los lectores de la visualización se centran únicamente en esto.

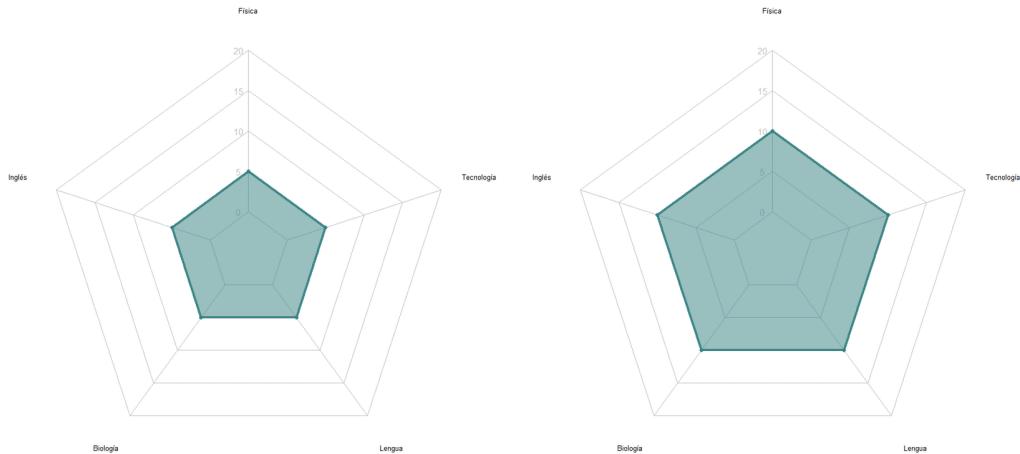
Veamos un ejemplo:

```
library(patchwork)  
p1 <- ggradar(first_radar, group.point.size = 3,  
                background.circle.colour = "white",  
                legend.position = "right")  
p2 <- ggradar(first_radar %>% select(group, defense, sp_defense, speed, everything()),  
                group.point.size = 3,  
                background.circle.colour = "white",  
                legend.position = "none")  
p1 + p2 + plot_layout(guides = 'collect')
```

En esta ocasión nos apoyamos en la magnífica librería **patchwork**, para combinar dos visualizaciones fácilmente. ¿Puedes apreciar cómo cambia la foto alterando solo la posición de las variables?



Finalmente, las diferencias entre polígonos pueden parecer más grandes de lo que son. El área de los polígonos se incrementa de forma cuadrática en lugar de linealmente, esto lleva a que se sobreestimen. Echemos un vistazo al siguiente ejemplo:



Tenemos las notas de dos estudiantes en diferentes asignaturas. El de la izquierda ha sacado 5 en todo, mientras que el de la derecha ha sacado el doble, un 10 en todo. Sin embargo, las diferencias en el área de una figura a otra son mucho más que el doble.

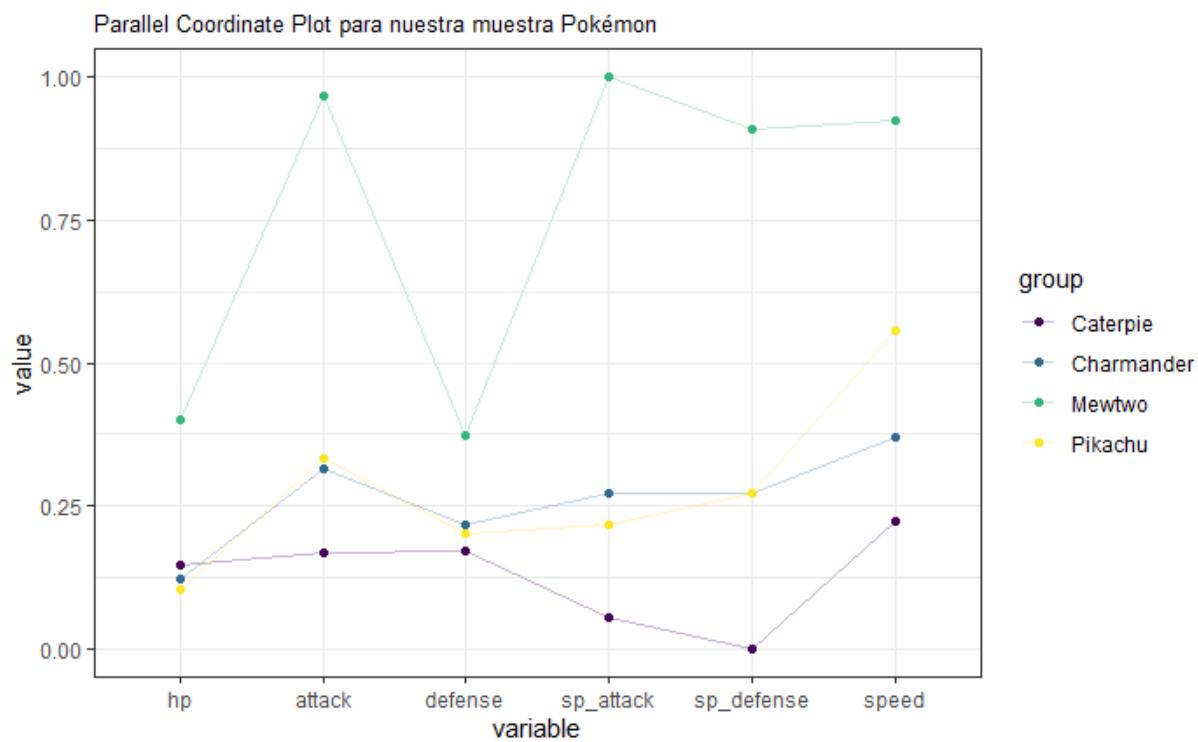
Alternativa a los gráficos de radar

Existe una visualización alternativa que también es útil para comparar numerosas observaciones en una gran cantidad de variables numéricas y que sufre menos problemas que la anterior, se trata de las coordenadas paralelas o *parallel plot*.

Este gráfico es equivalente al gráfico de radar, pero utiliza coordenadas cartesianas, por lo que suele ser preferido sobre el anterior. Veamos un ejemplo:

```
library(GGally)
library(viridis)

ggparcoord(first_radar %>%
  mutate(group = factor(group)),
  columns = 2:7, groupColumn = 1, order = "anyClass",
  showPoints = TRUE,
  scale="globalminmax",
  title = "Parallel Coordinate Plot para nuestra muestra Pokémon",
  alphaLines = 0.7
) +
  scale_color_viridis(discrete=TRUE) +
  theme(
    plot.title = element_text(size=10)
)
```



Necesitarás tener instalada la librería **GGally** que implementa este geom en particular (la librería **viridis** es opcional y se encarga de proporcionar la escala de colores). Además, **es necesario convertir la variable de agrupación (cada pokémon en este caso) en factor**.

En este gráfico, se aplana la visión y se suceden consecutivamente las variables. Cada punto es la estadística del sujeto en dicha variable (previamente escalada para evitar problemas de lectura). Aquí la comparación resulta mucho más sencilla a simple vista que en el caso anterior.

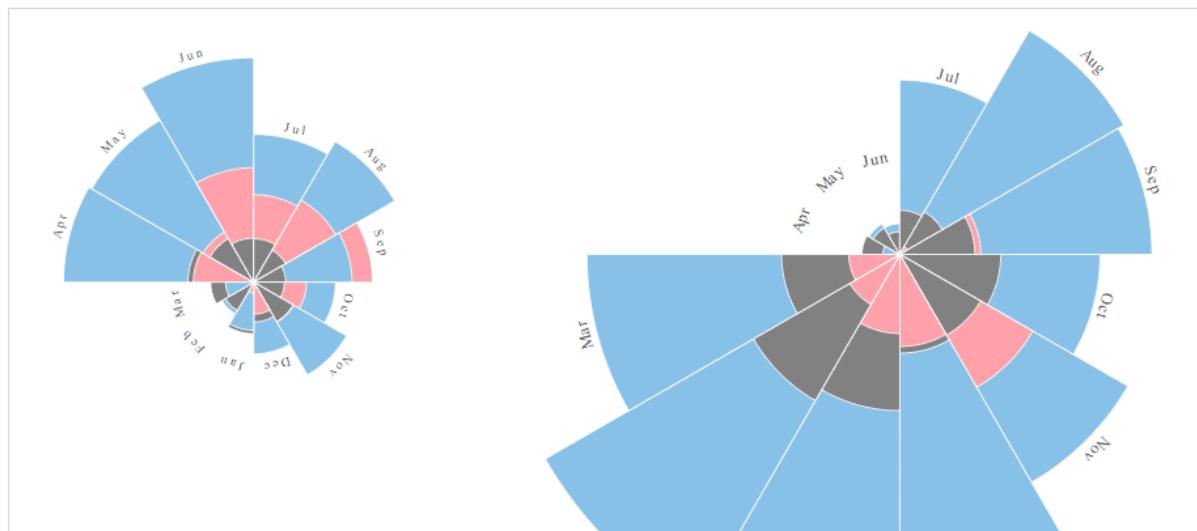
Recuerda que, al ser un gráfico basado en líneas, condensar muchísimas puede dar lugar a problemas de legibilidad.

Rosa de Nightingale: descripción

X Edix Educación

También conocido como *sunburst*, esta visualización debe su nombre a la gran **Florence Nightingale**.

Se dibujan sobre coordenadas radiales al igual que los gráficos radar y la mayor diferencia es que el interior de cada eje se rellena para representar las **proporciones** de una variable categórica mediante el área que ocupa.

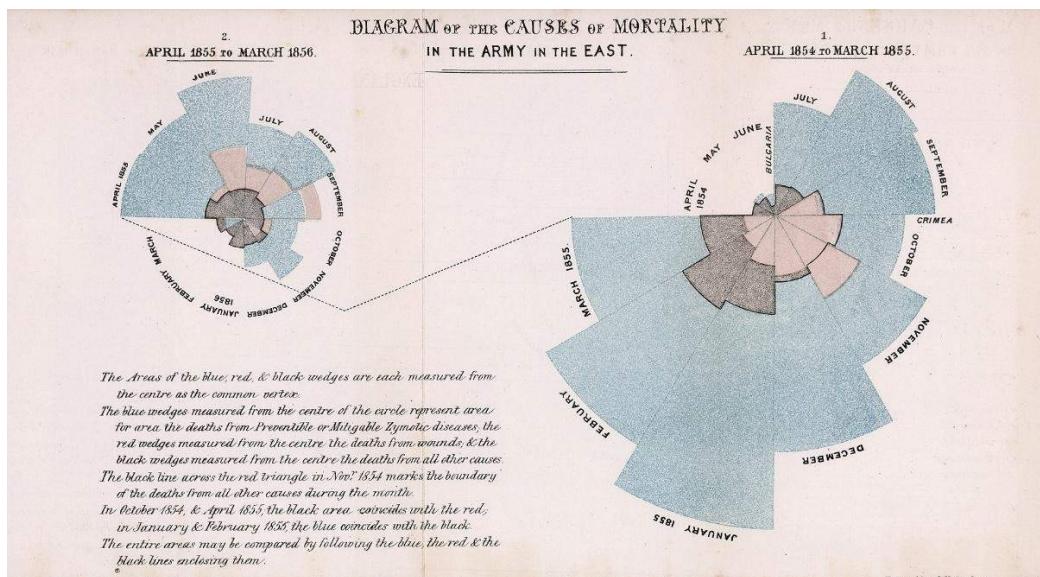


Fuente: [DataVizCatalogue](#).

Su lectura es muy similar a los gráficos de radar por lo que nos ahorraremos ese paso. Este tipo de gráfico hereda los problemas que hemos visto en el caso anterior y, además, añade las quejas que provienen de los gráficos de tarta o *donut*:

- Es difícil comparar el tamaño de los diferentes trozos resultantes debido a las áreas, cuantificar visualmente el área de cada trozo puede ser confusa.
- Utiliza este gráfico solo cuando la información se pueda leer bien, si no fuese el caso, considera otras **alternativas como los treemaps** (los veremos en esta sección).

Por añadir un poco de historia, la visualización original que utilizó Florence es la siguiente:



Fuente: [Wikipedia](#)

El origen de esta visualización es el ejemplo más claro de que la necesidad es la madre de la invención. Florence publicó, en 1858, *Notes on Matters Affecting the Health, Efficiency and Hospital Administration of the British Army*, que contenía un gráfico a todo color titulado *Diagram of the Causes of Mortality in the Army in the East*, el cual ilustraba las causas de mortalidad de los soldados que atendía en el hospital que dirigía durante la guerra de Crimea. Este gráfico demostraba fácilmente que gran parte de las muertes en el lado británico podían evitarse controlando numerosos factores como la nutrición, ventilación o el refugio donde eran atendidos los heridos.

Florence Nightingale fue pionera en utilizar las visualizaciones como vía para comunicar complejos datos estadísticos de una forma clara y persuasiva y os recomiendo encarecidamente leerlos algo más sobre su vida, pues fue el germen de la profesionalización de la enfermería, inspiración para la creación de la Cruz Roja... todo un referente.

Rosa de Nightingale en R

X Edix Educación

Para ilustrar el ejemplo de este tipo de visualizaciones en R, vamos a utilizar un dataset muy interesante: las mujeres más importantes según la BBC en 2020 por sus logros y contribuciones.

Para hacer esto posible, necesitarás instalar la librería **sunburstR** que nos ayudará a generar gráficos interactivos.

```
## Cargamos la libreria que necesitaremos (instálala si no lo has hecho)
library(sunburstR)

## Cargamos los datos desde github
women ← read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-12-08/women.csv')

women %>% head()
```

	name	img	category	country	role	description
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
1	Unsung	~ https://ichef.bbci.co.uk~	All	Worldwi~	Making a d~	"In an extraordinary year, w~
2	Loza Ab~	https://ichef.bbci.co.uk~	Leadersh~	Ethiopia	Footballer	"Loza Abera Geinore was born~
3	Houda A~	https://ichef.bbci.co.uk~	Creativi~	Morocco	Rapper	"Houda Abouz, aka Khtek, is ~
4	Christi~	https://ichef.bbci.co.uk~	Leadersh~	Netherl~	Campaigner	"Christina was behind the UK~
5	Yvonne ~	https://ichef.bbci.co.uk~	Leadersh~	Sierra ~	Mayor	"Mayor Yvonne Aki-Sawyerr OB~
6	Rina Ak~	https://ichef.bbci.co.uk~	Knowledge	Banglad~	Former sex~	"During the pandemic, Rina a~
7	Sarah A~	https://ichef.bbci.co.uk~	Knowledge	UAE	Minister f~	"Her Excellency Sarah Al-Ami~
8	Waad a~	https://ichef.bbci.co.uk~	Creativi~	Syria	Film-maker	"Waad al-Kateab is a Syrian ~
9	Adriana~	https://ichef.bbci.co.uk~	Knowledge	Italy	Pathologist	"Adriana Albini is head of t~
10	Ubah Ali	https://ichef.bbci.co.uk~	Leadersh~	Somalil~	FGM educat~	"Ubah Ali is a co-founder of~

Estos datos pertenecen a la **semana 50 del TidyTuesday** (te dejo el [enlace](#) para que lo uses en tu sesión de R). Las cien mujeres están separadas en países y, para no generar un gráfico ilegible, vamos a agregarlas por continente. A partir de ahí, podremos filtrar fácilmente y representar la información. Te dejo este trozo de código para ahorrarte el escribir tanto país.

```
asia <- c('Afghanistan', 'Bangladesh', 'China', 'Exiled Uighur from Ghulja (in Chinese, Yining)',  
'Hong Kong', 'India', 'Indonesia', 'Iran', 'Iraq/UK', 'Japan', 'Kyrgyzstan', 'Lebanon', 'Malaysia',  
'Myanmar', 'Nepal', 'Pakistan', 'Singapore', 'South Korea', 'Syria', 'Thailand', 'UAE', 'Vietnam',  
'Yemen')
```

```
south_america <- c('Argentina', 'Brazil', 'Colombia', 'Ecuador', 'Peru', 'Venezuela')
```

```
oceania <- c('Australia')
```

```
europe <- c('Belarus', 'Finland', 'France', 'Germany', 'Italy', 'Netherlands', 'Northern Ireland',  
'Norway', 'Republic of Ireland', 'Russia', 'Turkey', 'UK', 'Ukraine', 'Wales, UK')
```

```
africa <- c('Benin', 'DR Congo', 'Egypt', 'Ethiopia', 'Kenya', 'Morocco', 'Mozambique', 'Nigeria',  
'Sierra Leone', 'Somalia', 'Somaliland', 'South Africa', 'Tanzania', 'Uganda', 'Zambia',  
'Zimbabwe')
```

```
north_america <- c('El Salvador', 'Jamaica', 'Mexico', 'US')
```

Una vez realizado esto, solo necesitas crear una columna y llenarla con el continente según el país que tenga la observación.

```
women <- women %>%  
| mutate(continent = NA)  
  
# Rellenamos la variable continente con ifelse y auxiliar %in%  
women$continent <- ifelse(women$country %in% asia, 'Asia', women$continent)  
women$continent <- ifelse(women$country %in% south_america, 'South America', women$continent)  
women$continent <- ifelse(women$country %in% oceania, 'Oceania', women$continent)  
women$continent <- ifelse(women$country %in% europe, 'Europe', women$continent)  
women$continent <- ifelse(women$country %in% africa, 'Africa', women$continent)  
women$continent <- ifelse(women$country %in% north_america, 'North America', women$continent)
```

Por último, filtramos un continente en particular y realizamos algunas adecuaciones a la información. Vital es el paso de eliminar ‘-’ en las variables ‘role’ y ‘name’ para evitar problemas en la visualización.

En este ejemplo, la variable **path** enlaza toda la jerarquía que se encargará de renderizar nuestro gráfico y vendrá separada por guiones (¿entiendes ahora el paso anterior?).

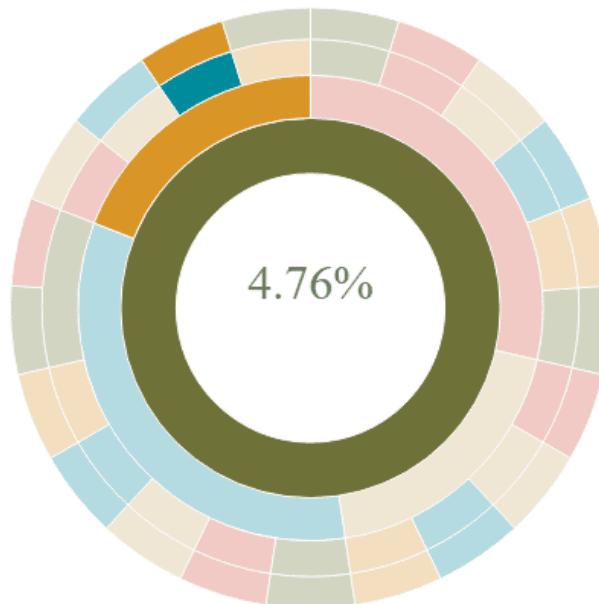
Finalmente, la variable **V2** es la que da peso a cada mujer, en este caso todas valen 1 y así lo referenciamos.

```
# Filtro para Europa
europa_name ← women %>%
  select(continent, category, role, name) %>%
  # Eliminamos los guiones para evitar problemas
  mutate_at(vars(3, 4), list(~gsub("-", "", .))) %>%
  filter(continent=='Europe') %>%
  # Creamos el path (vital para la visualización)
  mutate(
    path = paste(continent, category, role, name, sep = "-")
  ) %>%
  slice(2:100) %>%
  mutate(
    V2 = 1
  )

# Lanzamos el gráfico
sunburst(data = data.frame(xtabs(V2~path, europa_name)), legend = FALSE,
          colors = c("D99527", "6F7239", "CE4B3C", "C8AC70", "018A9D"))
```

En este ejemplo con **Sanna Marin** puedes visualizar rápidamente el uso de la variable **path** y cómo se estructura la jerarquía de todo el continente europeo. Puedes probar a generar por tu parte otros continentes de interés.

Europe > Leadership > Prime Minister of Finland > Sanna Marin > 4.76%



Alternativa a la rosa de Nightingale

Existe una alternativa muy potente a este tipo de gráficos denominados *treemaps*.

Estas figuras son útiles para **mostrar datos de manera jerarquizada** y utilizan el área para mostrar diferencias numéricas. Normalmente, cada categoría es asignada a rectángulos que varían en color y tamaño según su información; estos se van anidando, generando subcategorías y formando la jerarquía deseada.

En el ejemplo, tenemos divididos a todos los animales de una tienda según su clase y luego, esta misma se divide proporcionalmente en las diferentes familias que la componen.

Para que quede claro, jerarquizado en este ámbito significa que la suma de los niveles inferiores genera los valores de niveles superiores y así sucesivamente.



Fuente: [DataVizCatalogue](#)

Su creador, [Ben Shneiderman](#), desarrolló originalmente estos gráficos como una forma de visualizar de una eficazmente todos los ficheros almacenados en un ordenador, organizando en categorías como carpetas o tipología de ficheros, dando lugar a una imagen que no fuese muy compleja y no ocupase mucho espacio en la pantalla.

Por tanto, la **ventaja principal** de estas visualizaciones es la **rapidez y eficiencia** para tener una visión general de la composición de nuestros datos; además, la **forma para comparar proporciones** vía el tamaño del área de los cuadrados/rectángulos es otro beneficio muy destacable.

Por otro lado, **estas figuras pierden claridad en su estructura jerárquica** respecto a la rosa de Nightingale y los niveles se pueden difuminar fácilmente. **Elige sabiamente**, según las necesidades que tengas, la visualización más adecuada.

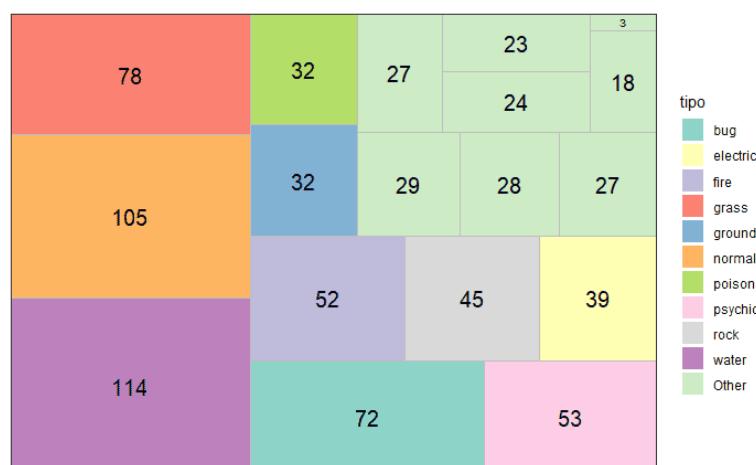
Vamos a reproducir esta visualización en R.

```
# Librería necesaria para usar el geom
library(treemapify)

## Adecantamos los datos de Pokémon
poke %>%
  count(type1) %>% arrange(desc(n)) %>%
  
  ## Convertimos en factor y colapsamos todos aquellos tipos que no exceden los 30
  mutate(tipo = factor(type1),
         tipo = fct_lump_min(tipo, 30, w = n)) %>%
  
  ggplot(aes(area = n, fill = tipo, label = n)) +
  geom_treemap() +
  geom_treemap_text(colour = "black",
                    place = "centre",
                    size = 15) +
  scale_fill_brewer(palette = "Set3")
```

Nos apoyaremos en la librería **treemapify** que nos proporcionará el geom necesario para dibujar un *treemap*.

Para obtener esta visualización, es necesario dar un tratado previo a los datos, en este caso utilizaremos los más de 800 pokémons según su tipo principal para echar un vistazo a las proporciones y entender si el juego actualmente está balanceado. El resultado es el siguiente:



Con el resultado obtenido podemos extraer algunas conclusiones:

1. Los tipos ‘Water’, ‘Normal’ y ‘Grass’ son los tipos más comunes en el juego con 114, 105 y 78 pokémons, respectivamente.
1. El caso menos común es ‘Flying’ puro, con esos tres casos catalogados dentro de ‘Other’.

Como apunte final, sería adecuado cambiar el color a algo más adecuado para evitar contradicciones (quizás el azul para ‘Water’ o el rojo para ‘Fire’, ¿no crees?).

Conclusiones

X Edix Educación

Gracias a este fastbook hemos aprendido las técnicas de visualización **comparativas con base radial**. El resultado de estas visualizaciones puede ser muy útil en determinados escenarios, pero debemos tener cuidado con su lista de defectos, la cual no es pequeña...

Utiliza los gráficos de radar y la rosa de Nightingale siempre que quieras contar una historia que capte la atención del lector y que tus datos te lo permitan. Cuando no sea el caso, puedes optar por opciones más amigables y con menos problemas como los *parallel coordinates* o *treemaps*.

Estos gráficos ofrecen muchos escenarios para su uso: desde comparativa de proporciones, a cambios a lo largo de un eje temporal o hasta una jerarquía establecida. Además, ayudan a entender el ‘todo’ con un simple vistazo. Utilízalos siempre que puedas para obtener un resumen general rápido.

Como siempre, te recomiendo encarecidamente que pruebes los resultados en cada opción y no te quedes solo con la lectura de este documento. Esta ciencia se aprende con práctica, así que, adelante, intenta picar el código para afianzar lo aprendido.

¿Te atreves a utilizar esta visualización con algún conjunto de datos propio? Prueba y extrae conclusiones.

Bibliografía

X Edix Educación

- [Ggplot2](#) – Página principal.
- [Data Visualization](#) de **Kieran Healy**.

¡Enhорabuena! Fastbook superado

edix

Creamos Digital Workers