

Fastbook 09

Visual Analytics

Visualizaciones para series
temporales



09. Visualizaciones para series temporales

En este fastbook vamos a realizar una inmersión en los gráficos utilizados cuando nuestro conjunto de datos dispone de un eje temporal. Este tipo de visualizaciones nos ayudarán a entender tendencias y cómo los datos pueden cambiar a lo largo del tiempo. Posiblemente sean los gráficos más utilizados en los medios de comunicación hoy en día, entenderlos y dominarlos será algo muy útil en vuestro día a día.

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 líneas: descripción](#)

[Gráficos de líneas en R](#)

[Gráficos de área: descripción](#)

[Gráficos de área en R](#)

[Conclusiones](#)

[Bibliografía](#)

Gráficos de líneas: descripción

X Edix Educación

Los gráficos de líneas nos ayudan a entender **cómo los datos cambian a lo largo del tiempo y exponen las tendencias** de estos.

Un **gráfico de líneas** es una representación visual que muestra la evolución de una o múltiples variables numéricas. La información es representada por unos puntos, denominados **marcadores**, que se conectan mediante líneas rectas.

Esta visualización hace uso de un **sistema de coordenadas cartesiano** y, tradicionalmente, el eje X ha servido para mostrar la escala temporal y, el eje Y, los valores cuantitativos.

Para profundizar en este concepto veamos un ejemplo de este gráfico.



Fuente: [DataVizCatalogue](#).

Habitualmente, también se les suele llamar **series temporales**. La lectura es muy sencilla: el eje X se desplaza de izquierda a derecha, donde las pendientes inclinadas hacia arriba nos indican incrementos en los valores y, las pendientes que decaen, decrementos.

El ‘viaje’ que forma cada línea a lo largo de todo el eje puede revelar los patrones y las tendencias estacionales de aprendizajes interesantes.

Algunos consejos sobre estas visualizaciones:

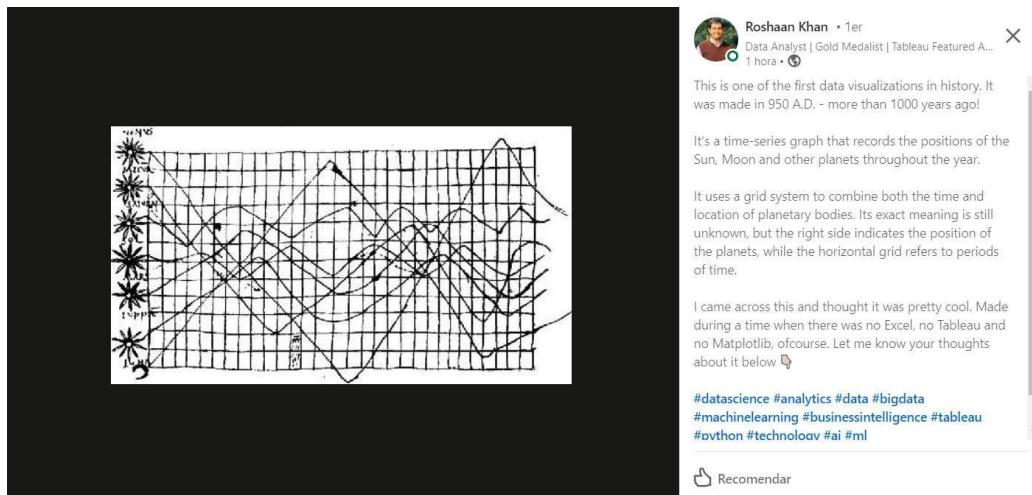
- Recuerda de los primeros fastbooks el dilema con el eje Y y su truncamiento. Es un tema polémico en Data Science, **¿debe empezar el eje siempre en 0?** Cuando hablamos de diagramas de barras es un sí absoluto, para evitar realzar las diferencias no significativas. Con series temporales no hay consenso, puesto que en ocasiones recurrir a incluir el 0 puede arruinar la escala visual de tu gráfico...

- Utilizando **múltiples líneas** se puede obtener visualizaciones muy potentes al comparar sobre el mismo eje. Sin embargo, evita utilizar cinco líneas o más en el mismo gráfico, esto generará ruido y hará que la información sea difícil de leer. Existen alternativas como los famosos facets, que separan cada línea en su propio gráfico y otras posibilidades que vamos a explorar en este fastbook.

Veamos ahora algunas nociones de historia y curiosidades.

El origen de esta visualización es muy antiguo. El primer gráfico de líneas se atribuye a [William Playfair](#), ingeniero escocés que lo popularizó en 1786. Como curiosidad adicional, a este hombre se le atribuye también el diagrama de barras y el vilipendiado gráfico de tartas.

No obstante, a continuación, te muestro una de las que se pueden considerar la **primera visualización de la historia**, que demuestra que esta visualización tan intuitiva ya lleva tiempo entre nosotros.



Si hace más de mil años ya conseguían estos resultados sin contar con R, no tienes excusa para crear una. ¡Vamos a ello!

Gráficos de líneas en R

X Edix Educación

Como ya es tradición con R, cargamos algunas opciones para trabajar más ágilmente. En esta ocasión, dada la particularidad de estos gráficos, deberás instalar las librerías que ves al final de la imagen:

```
# Desactivamos la notación científica. ¿A quién le gusta ver en sus gráficos números como
# 1e25?
options(scipen = 999)

# Cargamos las librerías necesarias para pintar
library(ggplot2) # Nuestra biblia a partir de ahora
library(scales) # Nos ayudará a mejorar el aspecto de nuestros gráficos

library(tidyverse) # Necesario si queremos realizar algún tratamiento en los datos

# Establecemos un tema por defecto para nuestros gráficos
# Personalmente soy fanático de theme_bw(), es el tema clásico 'dark-on-light'
# ggplot ofrece un listado de temas completos que puedes aprovechar. Echa un vistazo:
# https://ggplot2.tidyverse.org/reference/ggtheme.html

# Hay gente que realiza sus propios temas, generando auténticas obras de arte, ¿te atreves?
theme_set(theme_bw())

# Cargamos los datos que vamos a utilizar

data("diamonds", package = "ggplot2")
data("txhousing", package = "ggplot2")
data("midwest", package = "ggplot2")

## Librerías con datasets para series temporales
library(ukbabynames)
library(babynames)|
```

Descarga aquí las librerías [ukbabynames](#) y [babynames](#).

Ambas librerías, proporcionan datos oficiales del censo de nacimientos de UK y USA, respectivamente. Te dejo algunos detalles adicionales:

- **Ukbabynames:** cuenta con registros de los nombres de bebés recogidos en UK de forma anual desde 1974. Tienes tablas separadas para Inglaterra, Gales, Irlanda del Norte y Escocia. Además, cuenta con un ranking de popularidad de nombres desde 1.904 a 1.994. ¡Perfecto para estudiar evolutivos!
- **Babynames:** listado de nombres desde 1880 a 2017 provistos por el organismo de la SSA. La tabla recoge nombres que hayan sido asignados al menos cinco veces. Esto da como resultado un tablón de casi dos millones de filas. Evidentemente, filtraremos algunos nombres interesantes entre los más de 97 mil nombres registrados. Proporcionaremos un dataset filtrado con un conjunto de nombres para aquellos que tengáis un PC más limitado en recursos.

Para este tutorial nos decantamos por la opción estadounidense al tener un histórico de 137 años por nombre, lo que proporcionará evolutivos más claros. Dispones de ambas opciones para que puedas practicar con diferentes horizontes temporales, ¡elige la que quieras!

```
## Tu primer gráfico de líneas

# Filtramos un nombre para nuestro primer gráfico
df <- babynames %>% filter(name = "Linda" & sex = "F")

gg <- ggplot(df, aes(x = year, y = n)) +
  geom_line(color = "lightblue", size = 1.2) +
  scale_y_continuous(labels = comma) +
  scale_x_continuous(breaks = pretty_breaks(n = 10)) +
  labs(title = "Serie evolutiva de bebés con el nombre Linda en los Estados Unidos",
       subtitle = "De 1880 a 2017",
       y = "Frecuencia",
       x = "Año")
```

gg

Et voilá! Aquí tu primer gráfico de líneas o serie temporal. Como ves, he seleccionado un nombre en especial por los patrones que exhibe. Es la primera línea que verás.

Del dataset total nos quedamos con los bebés llamados ‘Linda’ y que sean **femeninos** (en algunos años se registran bebés masculinos con ese nombre y el gráfico con ese código te generaría algo un poco raro).



¿Qué aprendizajes sacamos de este gráfico? El nombre de Linda apenas tenía registros hasta mediados de la década de los años 30, cuando comenzó a escalar en popularidad para alcanzar su pico a finales de la década de los años 40, con casi cien mil bebés registrados. A partir de ahí, ha ido perdiendo fuerza siendo hoy en día apenas un residuo entre los nuevos nacimientos.

Sobre el código también te comarto algún detalle más:

- Podemos **modificar los ejes X e Y para que luzcan una mejor apariencia**. En el caso del eje X, hemos asignado un mayor número de cortes en los años que mostraba el gráfico por defecto. En el eje Y hemos formateado los números para que añadan la coma y así, tener una mejor lectura de la frecuencia. Tanto ‘comma’ como ‘pretty_breaks’ son funciones pertenecientes al paquete **scales**.

- Con ‘`labs`’ puedes modificar el texto que aparece en los diferentes elementos clave de tu visualización. En nuestro caso hemos añadido título y subtítulo informativo y, además, hemos cambiado el nombre original de las variables X e Y para un mejor entendimiento.
- Dentro del geom utilizado, lógicamente `geom_line()`, podemos utilizar ciertos parámetros para mejorar la apariencia de nuestra serie temporal. En nuestro caso, hemos utilizado un color azulado. Y recuerda que puedes usar nombres (te dejo en el siguiente [enlace](#) un listado con todos los **nombres** que acepta R base) o **códigos HEX**, para lo cuál te dejo otro [enlace](#) a un generador que te va a encantar. También hemos modificado el tamaño de la línea para que sea más clara.

Vamos ahora con un ejemplo utilizando múltiples nombres.

Comenzamos con el código, en este caso hemos realizado un trabajo previo de filtrado. Hemos buscado aquellos nombres que eran los más populares entre 1880 y 1890 para consultar su evolución. Nos hemos quedado con el top 5, ya que pintar más líneas de ese límite en el mismo gráfico puede dificultar la lectura.

```
## Pasamos a algo más elaborado
# Filtramos un nombre para nuestro primer gráfico
nombres ← babynames %>%
  filter(between(year, 1880, 1890)) %>%
  group_by(name) %>%
  summarise(frec = sum(n), .groups = "drop") %>% arrange(desc(frec)) %>%
  top_n(5, wt = frec)

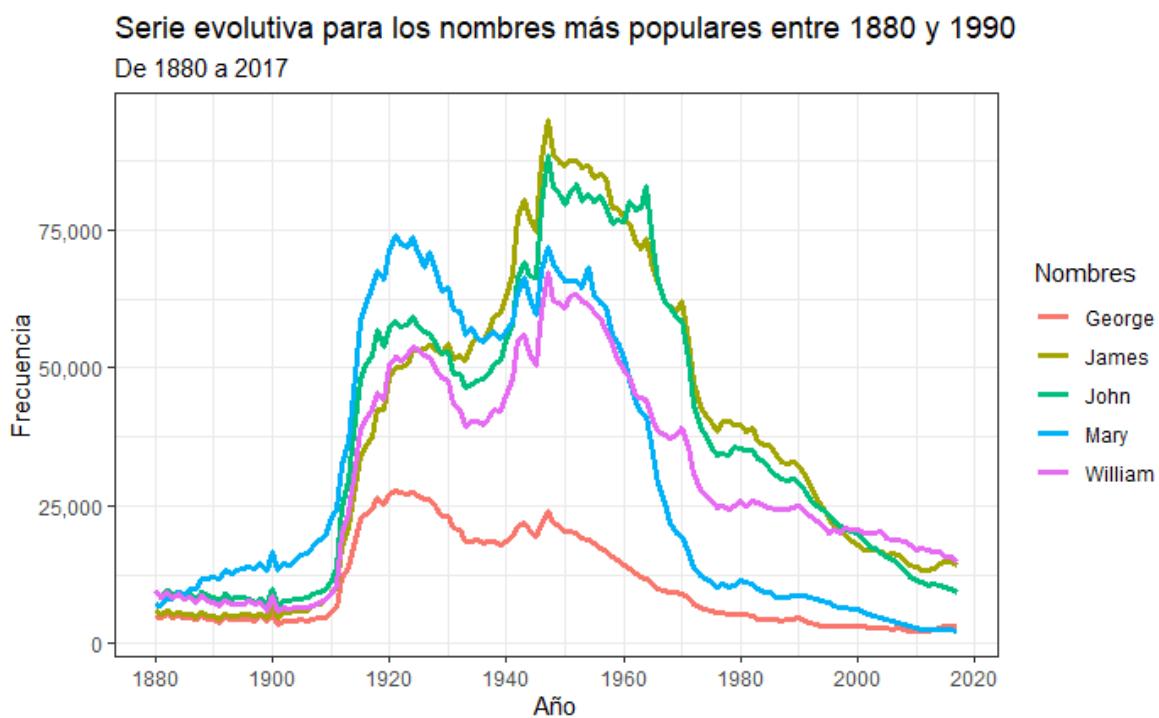
df ← babynames %>%
  filter(name %in% nombres$name) %>%
  group_by(year, name) %>%
  summarise(n = sum(n), .groups = "drop")

gg ← ggplot(df, aes(x = year, y = n, colour = name)) +
  geom_line(size = 1.2) +
  scale_y_continuous(labels = comma) +
  scale_x_continuous(breaks = pretty_breaks(n = 10)) +
  labs(title = "Serie evolutiva para los nombres más populares entre 1880 y 1990",
       subtitle = "De 1880 a 2017",
       y = "Frecuencia",
       x = "Año",
       colour = "Nombres")

gg
```

Tras encontrar esos nombres, hemos filtrado del dataset original los mismos y hemos agregado por año, unificando de esta forma el sexo (nos da lo mismo que alguien llame ‘William’ a su niño o niña, solo queremos el número de casos por año).

Volvemos a pintar con una salvedad: ahora el color es especificado en ggplot con el argumento **colour** y utilizando la variable ‘name’ para dotar de colorido a cada línea (recuerda que puedes cambiar esto con la orden `scale_color_brewer()` y la paleta deseada).



En cuanto al gráfico, queda claro que todos los nombres experimentaron un comportamiento muy similar: una crecida de popularidad desde 1.910 con ‘Mary’ a la cabeza para llegar a su máximo potencial en los años 50 con ‘James’. A partir de ahí, caída a mínimos, sobre todo del caso de ‘Mary’ y ‘George’.

Un apunte estético: **muchos expertos en visualización odian la leyenda en este tipo de gráficos**. Argumentan que no es efectivo el tener que estar volteando la mirada continuamente a la misma para ir iterando sobre los colores asignados a cada línea.

Te dejo un truco para solucionar esto y de paso te presento la maravillosa librería [ggrepel](#).

```
library(ggrepel)
## Pasamos a algo más elaborado
# Filtramos un nombre para nuestro primer gráfico
nombres ← babynames %>%
  filter(between(year, 1880, 1890)) %>%
  group_by(name) %>%
  summarise(frec = sum(n), .groups = "drop") %>% arrange(desc(frec)) %>%
  top_n(5, wt = frec)

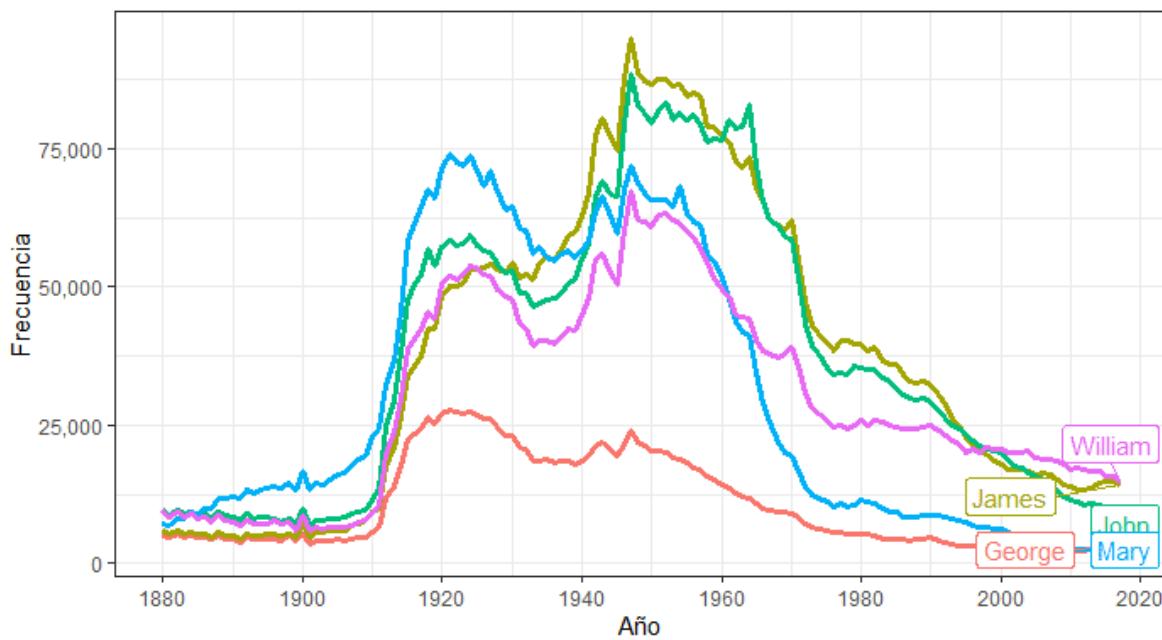
df ← babynames %>%
  filter(name %in% nombres$name) %>%
  group_by(year, name) %>%
  summarise(n = sum(n), .groups = "drop")

df ← df %>%
  mutate(label = if_else(year == max(year), name, NA_character_))

gg ← ggplot(df, aes(x = year, y = n, colour = name)) +
  geom_line(size = 1.2) +
  geom_label_repel(aes(label = label), nudge_x = 1, na.rm = T) +
  scale_y_continuous(labels = comma) +
  scale_x_continuous(breaks = pretty_breaks(n = 10)) +
  labs(title = "Serie evolutiva para los nombres más populares entre 1880 y 1990",
       subtitle = "De 1880 a 2017",
       y = "Frecuencia",
       x = "Año",
       colour = "Nombres") +
  theme(legend.position = "none")
gg
```

Verás que requiere un poco de trabajo extra. Generamos una nueva columna, que será la etiqueta mostrada en el último valor, y para ello creamos dicha etiqueta (que será el nombre de cada línea) en función de si el año es el máximo o no de todo el dataset.

Serie evolutiva para los nombres más populares entre 1880 y 1990
De 1880 a 2017



A partir de aquí, solo necesitamos añadir la función `geom_label_repel` (la otra función que contiene el paquete es `geom_text_repel`, ¡échale un vistazo!).

En ella especificamos el **argumento aes** con la variable que acabamos de crear y algunos parámetros, como la posición de inicio de la etiqueta y que omita los NAs (para no inundar el gráfico con etiquetas inservibles).

La especialidad de esta librería es repeler los textos o etiquetas, de tal forma que no se produzca superposición entre ellas que impida su lectura. Los argumentos y funciones que

trae por defecto ggplot no cuentan con estas características y suelen generar gráficos ilegibles cuando se utilizan etiquetas. Piensa en `ggrepel` siempre que te topes con estos problemas.

En ocasiones, los elementos a visualizar exceden los cinco casos. En esas situaciones, plotear todo en la misma visualización hace muy complicado leer y extraer todos los detalles importantes. Vamos a ver qué podemos hacer:

```
## Y si tenemos más de 5 líneas?
# Filtramos un nombre para nuestro primer gráfico
set.seed(1)

nombres ← babynames %>%
  group_by(name) %>% summarise(frec = sum(n)) %>%
  arrange(desc(frec)) %>%
  top_n(100) %>%
  sample_n(10)

df ← babynames %>%
  filter(name %in% nombres$name) %>%
  group_by(year, name) %>%
  summarise(n = sum(n), .groups = "drop")

df ← df %>%
  mutate(label = if_else(year == max(year), name, NA_character_))

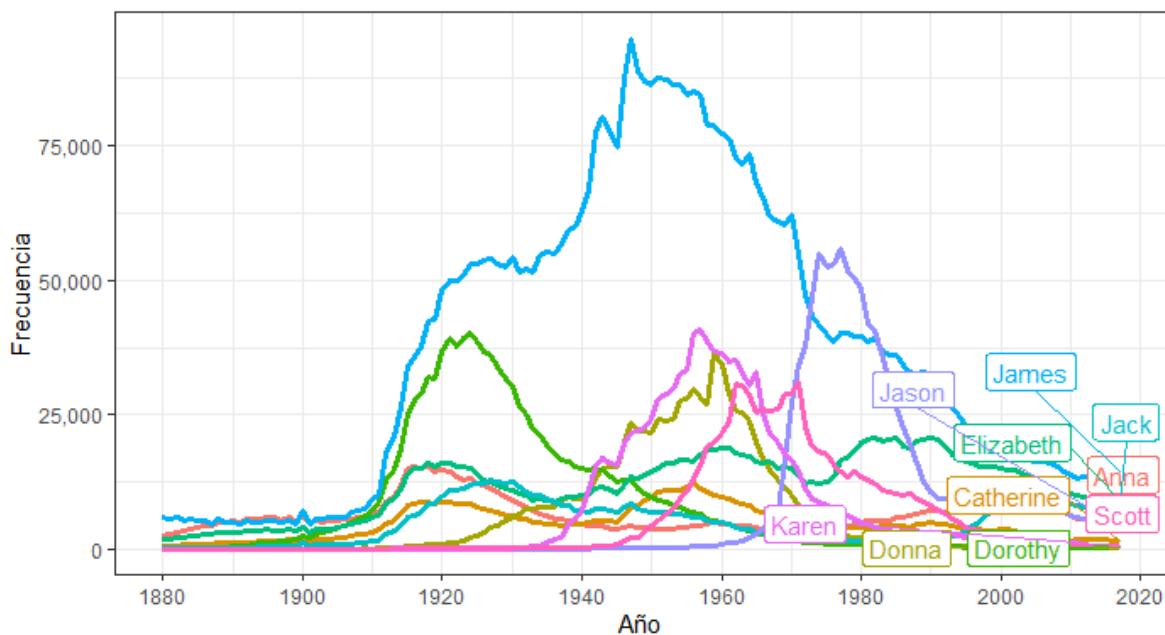
gg ← ggplot(df, aes(x = year, y = n, colour = name)) +
  geom_line(size = 1.2) +
  geom_label_repel(aes(label = label), nudge_x = 1, na.rm = T) +
  scale_y_continuous(labels = comma) +
  scale_x_continuous(breaks = pretty_breaks(n = 10)) +
  labs(title = "Serie evolutiva para una muestra de 10 de los nombres más populares históricamente",
       subtitle = "De 1880 a 2017",
       y = "Frecuencia",
       x = "Año") +
  theme(legend.position = "none")

gg
```

En este caso, hemos ordenado los nombres por popularidad total y **hemos seleccionado diez al azar del top 100**. Para que te salgan los mismos que a mí, si quieres imitar este resultado,

he establecido la semilla aleatoria para obtener **reproducibilidad** (si alteras el número que hay en la función `set.seed` obtendrás otros resultados, ¡prueba!).

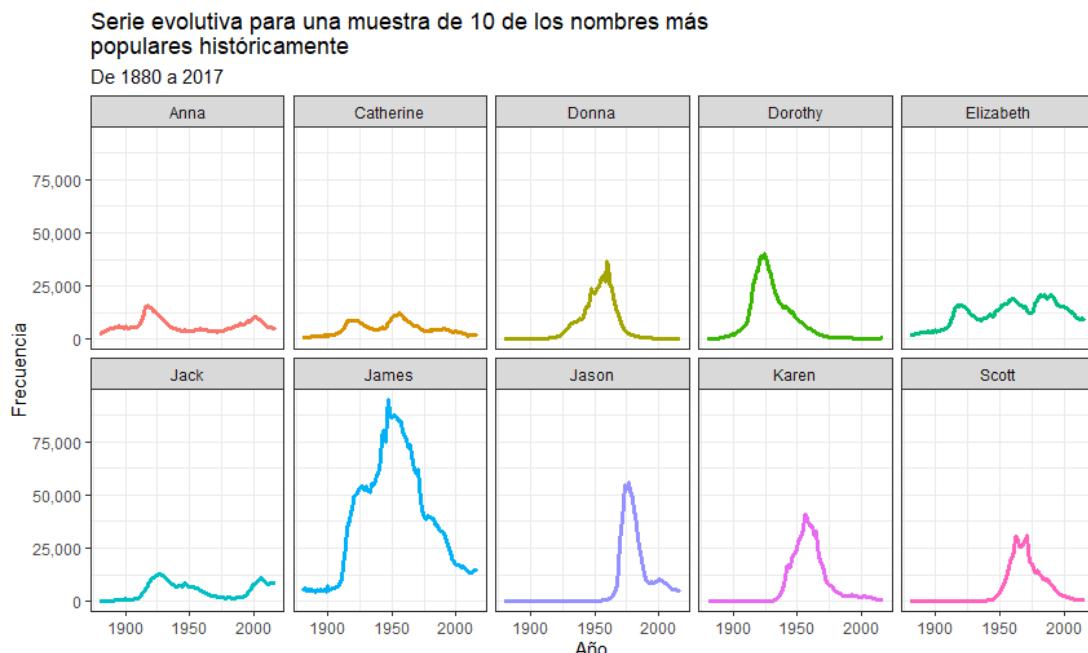
Serie evolutiva para una muestra de 10 de los nombres más populares históricos De 1880 a 2017



Salta a la vista que los nombres más destacados son sencillos de leer, pero en la zona baja se forma una maraña imposible de entender. Para solucionar esto, ya tienes las herramientas perfectas... ¡los facets!

```
gg ← ggplot(df, aes(x = year, y = n, colour = name)) +  
  geom_line(size = 1.2) +  
  # geom_label_repel(aes(label = label), nudge_x = 1, na.rm = T) +  
  scale_y_continuous(labels = comma) +  
  scale_x_continuous(breaks = pretty_breaks(n = 5)) +  
  facet_wrap(~name) +  
  labs(title = "Serie evolutiva para una muestra de 10 de los nombres más \nipopulares históricamente"  
       subtitle = "De 1880 a 2017",  
       y = "Frecuencia",  
       x = "Año") +  
  theme(legend.position = "none")  
  
gg
```

Mucho mejor, ¿no? Solo ha sido necesario el uso de la orden `facet_wrap` y, en este caso, especificar el número de filas a mostrar.



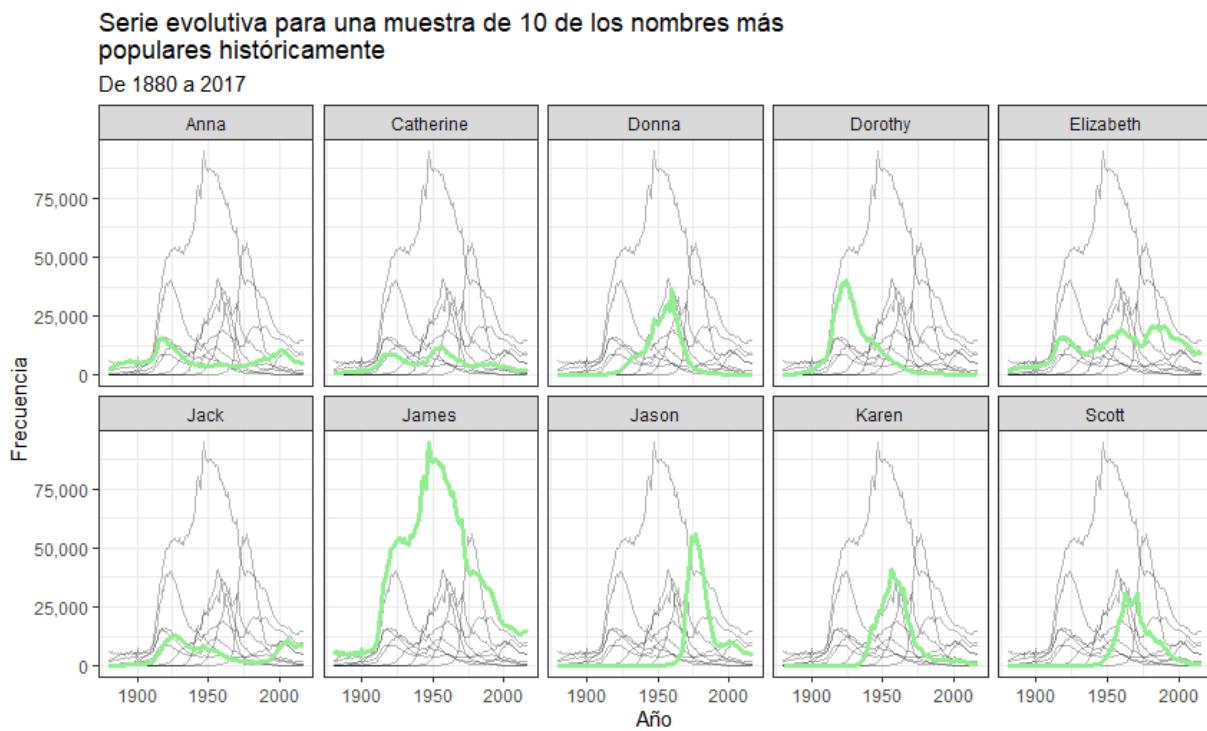
Alternativa al uso de facets

Cuando queremos mostrar múltiples líneas en la misma visualización, la legibilidad del gráfico se pierde y lo que obtenemos es un laberinto imposible de entender para nuestra audiencia. Los facets son una posible solución a este caso, pero no la única. Cuando te enfrentes a esta problemática, considera utilizar como alternativa los **gráficos espagueti** (vaya nombres, ¿no?).

```
df <- df %>%
  mutate(name2 = name)

gg <- ggplot(df, aes(x = year, y = n)) +
  geom_line(data = df %>% select(-name), aes(group = name2), color = "black", size = 0.5, alpha = 0.3) +
  geom_line(aes(color = name), color = "lightgreen", size = 1.2) +
  # geom_label_repel(aes(label = label), nudge_x = 1, na.rm = T) +
  scale_y_continuous(labels = comma) +
  scale_x_continuous(breaks = pretty_breaks(n = 5)) +
  facet_wrap(~name, nrow = 2) +
  labs(title = "Serie evolutiva para una muestra de 10 de los nombres más \n\npopulares históricamente",
       subtitle = "De 1880 a 2017",
       y = "Frecuencia",
       x = "Año") +
  theme(legend.position = "none")

gg.
```



No existe un geom específico para esta función. **Se genera combinando dos geom_line**: el primero será el encargado de pintar todos los nombres superpuestos, pero en un color negro y más pequeño de lo normal (además de una fuerte transparencia). El segundo geom_line es el que conectará con el facet e iluminará cada nombre con un color determinado. En este caso hemos fijado el mismo color para todos, pero puedes aprovechar las paletas como en los ejemplos anteriores.

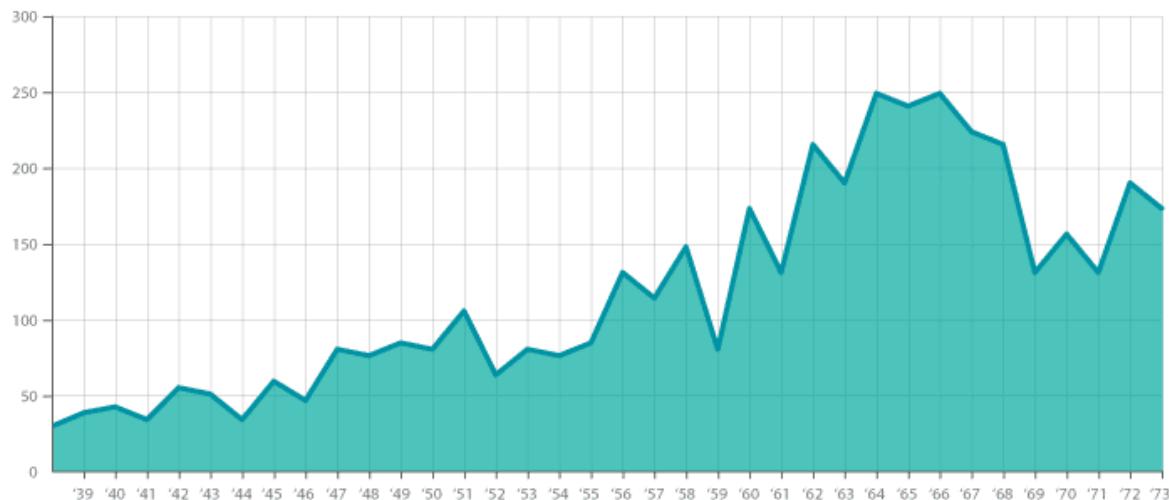
¿No crees que es más sencillo centrar la vista en cada ejemplo particular con esta técnica?

Gráficos de área: descripción

X Edix Educación

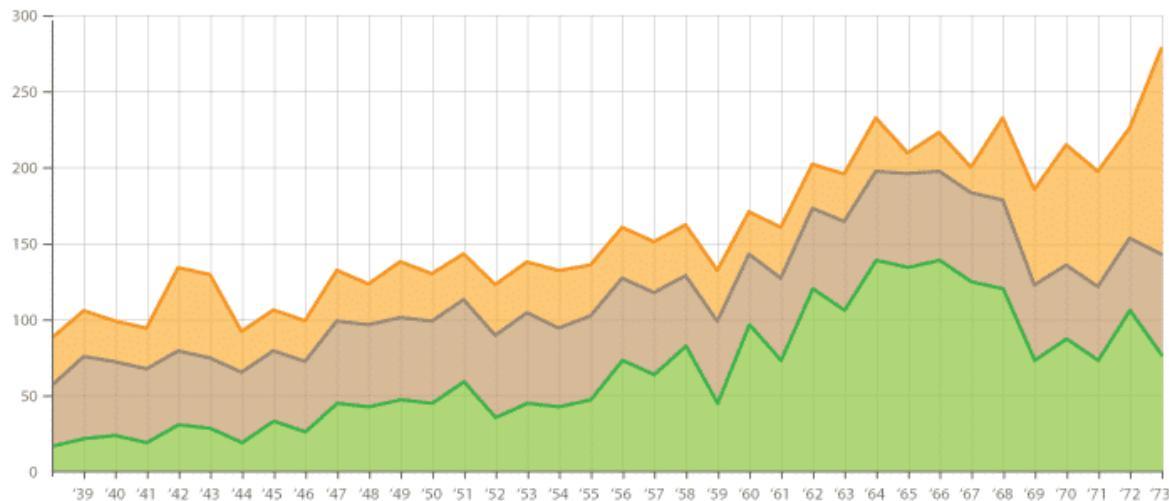
Un **gráfico de áreas** combina las capacidades de un **gráfico de líneas** y un **gráfico de barras** para mostrar cómo uno o más grupos de variables numéricas evolucionan a lo largo de un eje temporal.

Este tipo de gráficos se distinguen de los de líneas porque añaden un **sombreado de color** entre la línea y el eje X, justo como en un gráfico de barras.



Fuente: [DataVizCatalogue](#)

La lectura es exactamente igual que la de los gráficos de líneas. Esta visualización está **especialmente indicada** cuando cuentas con múltiples líneas y quieres realizar comparaciones entre ellas; o como un ‘total’, dividido por la suma de dichas líneas. Estas opciones darán pie a los dos tipos de gráficos que veremos en este punto.



Fuente: [DataVizCatalogue](#)

Dentro de los gráficos de área también veremos los de **área apilada o stacked**. Estas visualizaciones son simplemente una extensión del original cuya utilidad reside en la comparación de grupos. Esta comparación se realiza mediante el apilado de los valores en cada punto del eje temporal, permitiendo analizar la evolución de los diferentes grupos y calibrando la importancia de cada uno.

Ten en cuenta que estos gráficos **no están exentos de crítica** dada la dificultad que supone leerlos correctamente. **El lector tendrá que calcular mentalmente la importancia de cada grupo para entender la foto global**, algo similar a lo que pasa con los pie charts. Si tu objetivo es ese, opta por utilizar gráficos de líneas vistos en el punto anterior que ofrecerán mejores resultados.

Gráficos de área en R

Edix Educación

Continuamos desde el ejemplo anterior, por lo tanto, se da por hecho la carga inicial de datos y librerías para poder trabajar. Continuamos trabajando con ‘babynames’, los datos de nombres de bebés en USA:

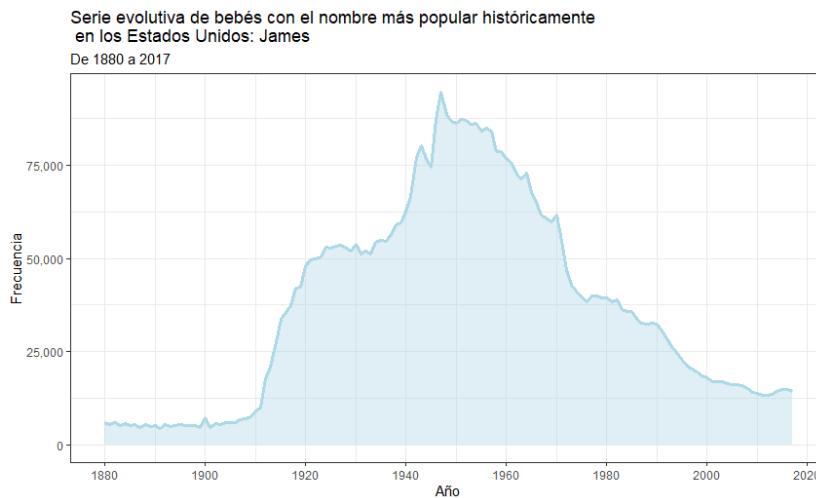
```
## Tu primer gráfico de áreas

# Filtramos un nombre para nuestro primer gráfico
df ← babynames %>% filter(name == "James" & sex == "M")

gg ← ggplot(df, aes(x = year, y = n)) +
  geom_area(fill = "lightblue", alpha = 0.4) +
  geom_line(color = "lightblue", size = 1.2) +
  scale_y_continuous(labels = comma) +
  scale_x_continuous(breaks = pretty_breaks(n = 10)) +
  labs(title = "Serie evolutiva de bebés con el nombre más popular históricamente\nen los Estados Unidos: James",
       subtitle = "De 1880 a 2017",
       y = "Frecuencia",
       x = "Año")

gg
```

Aquí podemos ver la evolución de los niños llamados ‘James’ en todo el histórico. Es el nombre más popular en todo el histórico (te dejo a ti la comprobación).



Como puedes apreciar en el código, se combinan dos geom: **geom_area** y **geom_line**. Se suele dotar de transparencia al área en sí para obtener un resultado visualmente más atractivo.

```
## Gráfico de área superpuesto

# Filtramos un nombre para nuestro primer gráfico
nombres_clasicos ← babynames %>%
  filter(between(year, 1880, 1950)) %>%
  group_by(name) %>%
  summarise(frec = sum(n), .groups = "drop") %>% arrange(desc(frec)) %>%
  top_n(3, wt = frec)

nombres_modernos ← babynames %>%
  filter(between(year, 1980, 2020)) %>%
  group_by(name) %>%
  summarise(frec = sum(n), .groups = "drop") %>% arrange(desc(frec)) %>%
  top_n(3, wt = frec)

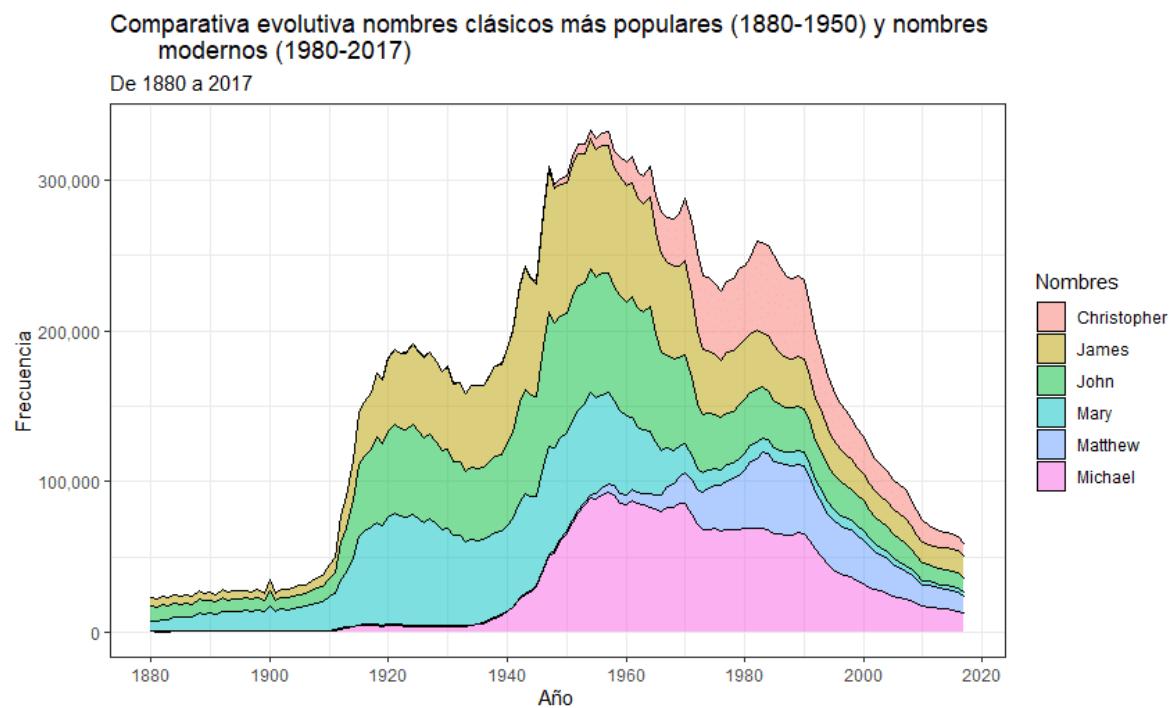
df ← babynames %>%
  filter((name %in% nombres_clasicos$name) | (name %in% nombres_modernos$name)) %>%
  group_by(year, name) %>%
  summarise(n = sum(n), .groups = "drop")

gg ← ggplot(df, aes(x = year, y = n, fill = name)) +
  geom_area(alpha = 0.5, size = 0.5, colour = "black") +
  scale_y_continuous(labels = comma) +
  scale_x_continuous(breaks = pretty_breaks(n = 10)) +
  labs(title = "Comparativa evolutiva nombres clásicos más populares (1880-1950) y nombres modernos (1980-2017)",
       subtitle = "De 1880 a 2017",
       y = "Frecuencia",
       x = "Año",
       fill = "Nombres")

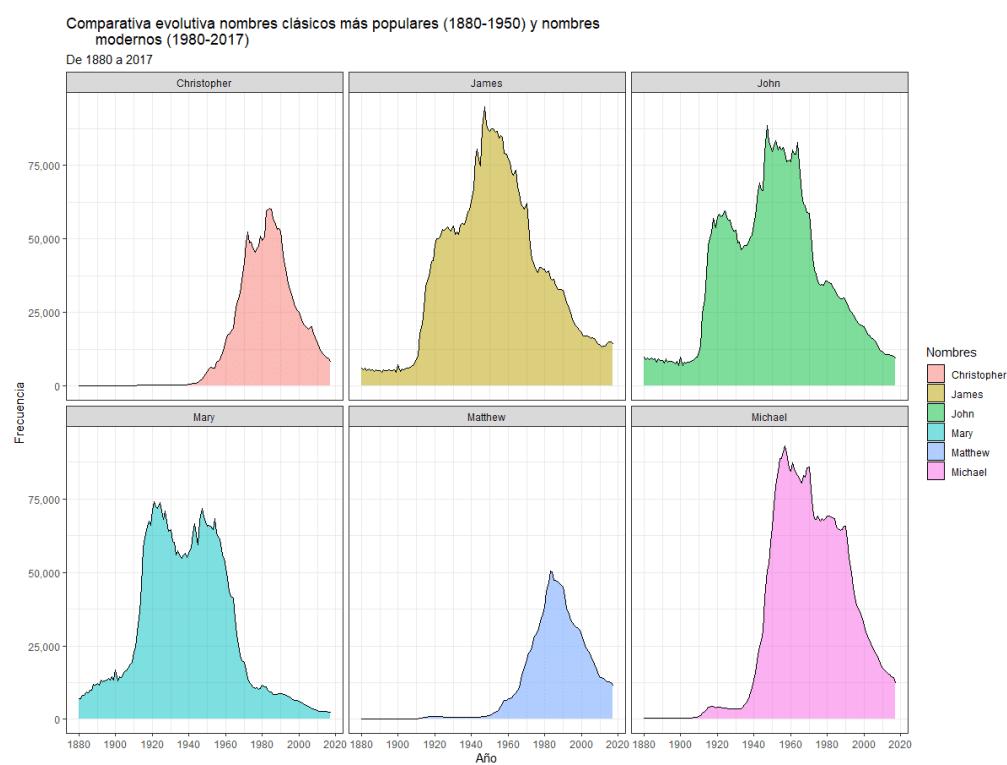
gg
```

Lo que obtenemos ahora es una superposición de áreas a lo largo del eje temporal (en inglés se conoce como **stacked area chart**).

Nos permite analizar y comparar la evolución de los tres nombres clásicos ('James', 'John' y 'Mary') frente a los tres modernos ('Christopher', 'Matthew' y 'Michael'). Es interesante ver el crecimiento tan fuerte de los clásicos y su reducción a lo largo de los últimos cuarenta años, sobre todo en el caso de 'Mary'. El nombre de 'Michael' nace con fuerza desde los años 40 y 'Matthew' apenas existía previo a los años 50.



Como curiosidad, si utilizas la función de `facet_wrap` para este ejemplo, separarás cada área en una visualización individual:

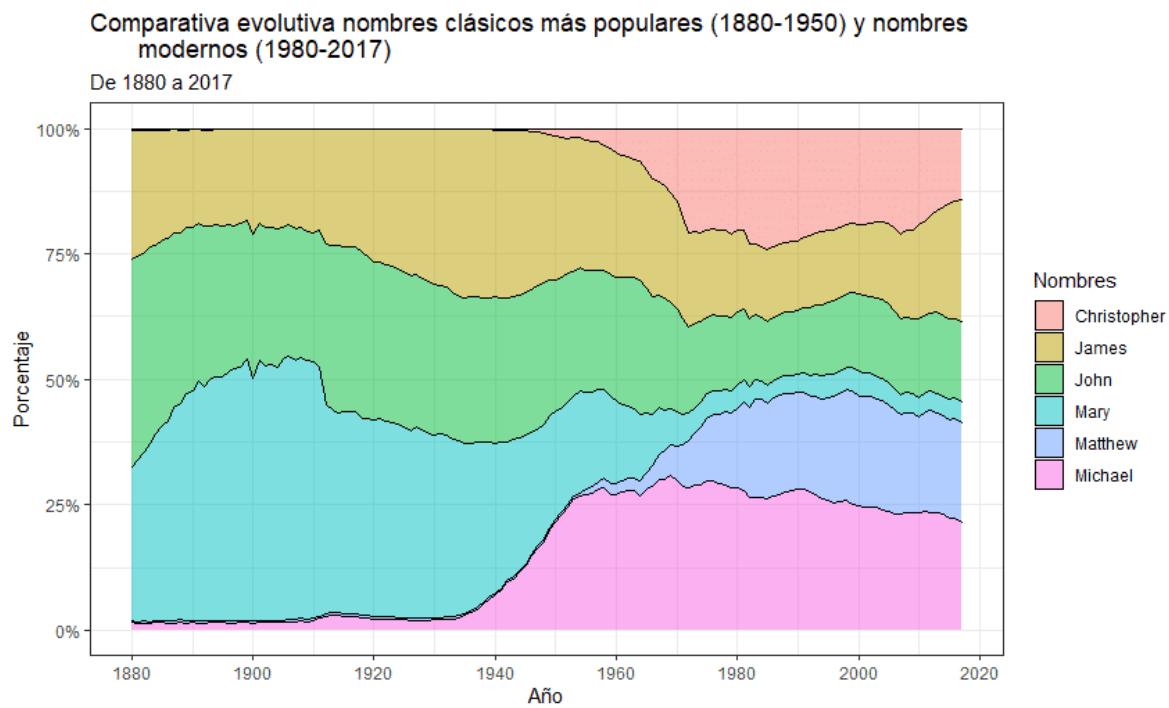


Este tipo de gráficos es muy útil para entender cómo un total se fragmenta en las diferentes series que lo componen y permite observar tendencias de una forma más obvia que los gráficos de líneas.

Por último, la otra opción que permite este tipo de gráficos es la **comparativa a nivel porcentual sobre el total del área**. En lugar de utilizar los valores absolutos, se calculan las proporciones de todos los elementos para que sumen 100%. Este tipo de gráficos pierde información sobre la tendencia que proporcionan los valores absolutos, pero permite comparar de forma relativa los distintos grupos.

```
gg <- ggplot(df, aes(x = year, y = n, fill = name)) +  
  geom_area(position = "fill", stat = "identity", alpha = 0.5, size = 0.6, colour = "black") +  
  scale_y_continuous(labels = percent) +  
  scale_x_continuous(breaks = pretty_breaks(n = 10)) +  
  labs(title = "Comparativa evolutiva nombres clásicos más populares (1880-1950) y nombres  
  modernos (1980-2017)",  
    subtitle = "De 1880 a 2017",  
    y = "Porcentaje",  
    x = "Año",  
    fill = "Nombres")  
  
gg
```

Basta con añadir los **argumentos position = “fill”** y **stat = “identity”** para obtener esta visión. De esta forma es fácil observar que los nombres clásicos no se han perdido del todo, aunque les han surgido duros competidores que en sus tiempos no existían.



Gráficos de flujo o *streamgraphs*

Este tipo de visualización es una evolución de los gráficos apilados.



Su principal diferencia es que **el eje X** deja de ser un eje fijo y se convierte en una especie de **baseline** alrededor del cual se distribuyen los valores.

Su nombre proviene de la forma que toma la visualización, como si de un río se tratara. ¿Punto a favor? Este tipo de gráficos consigue que los **resultados sean visualmente muy atractivos** y genera un interés inmediato en el lector.

Además **funciona bien con gran cantidad de categorías** y se utiliza para descubrir patrones o picos estacionales en la información.

Su principal problema es el mismo que el de sus predecesores, **la lectura de la información**. Con este tipo de visualizaciones nunca tendrás una visión completa de la información, ya que opaca el aporte de pequeñas categorías. Es un gráfico orientado a un **público de negocio o que no tiene tiempo para descifrar el gráfico** ni explorar hasta el más mínimo detalle de la información.

Vamos a probarlo en R:

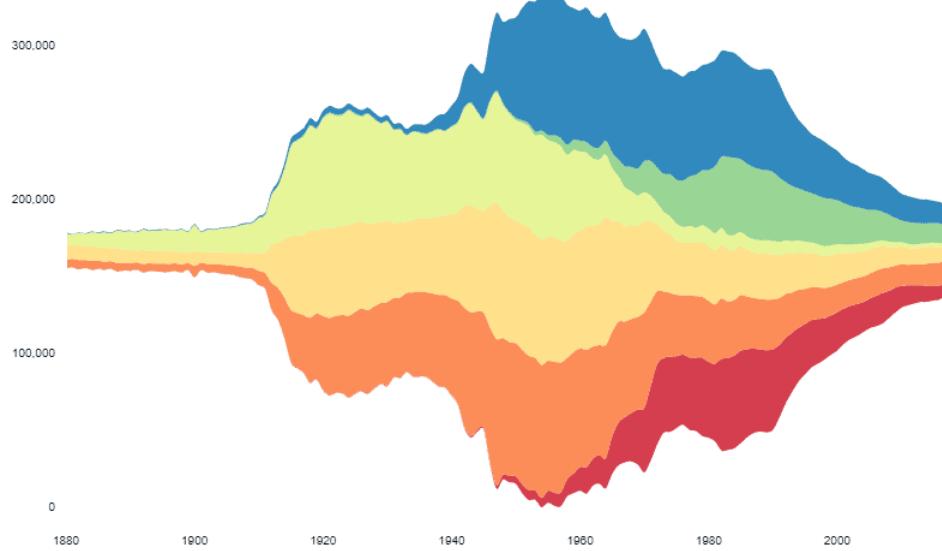
```
devtools::install_github("hrbrmstr/streamgraph")
```

Para poder generar esta visualización, debes instalar un paquete específico haciendo uso de la librería **devtools**, ya que este paquete no está presente en CRAN, el repositorio oficial de paquetes.

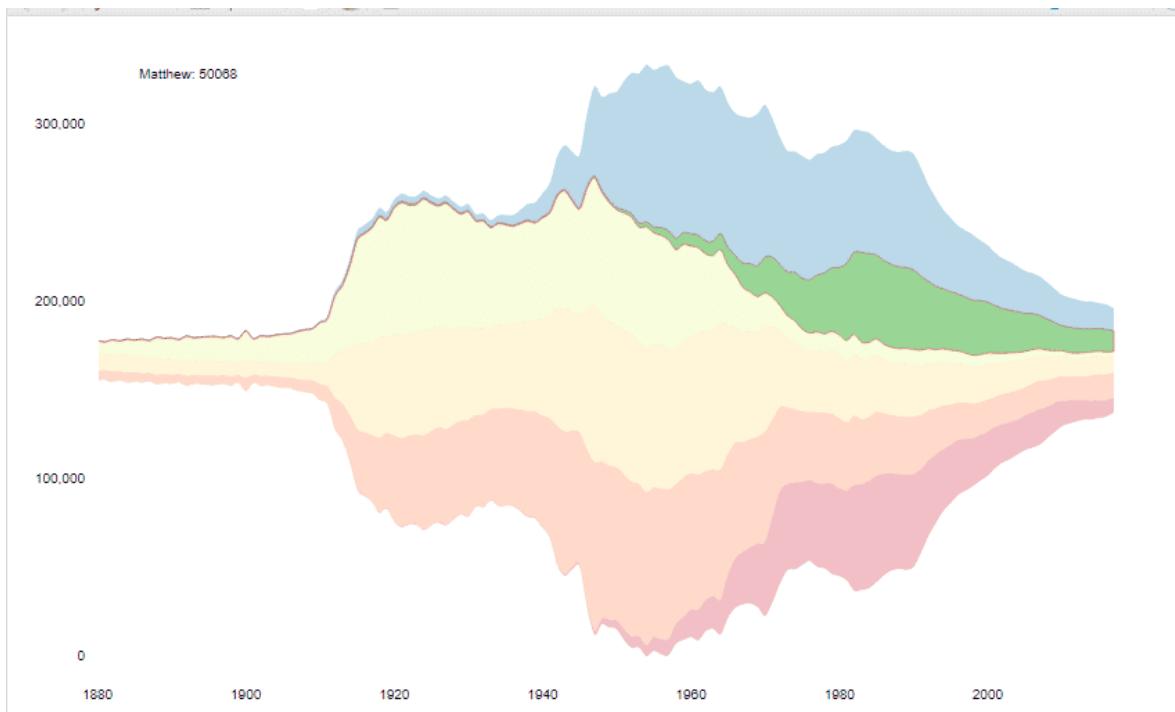
```
library(streamgraph)
streamgraph(df, key = "name", value = "n", date = "year", interactive=TRUE) %>%
  sg_axis_x(20, "year", "%Y") %>%
  sg_fill_brewer("Spectral")
```

Basta con cargar la librería y utilizar el conjunto de datos que ya teníamos anteriormente (nombres clásicos vs. modernos).

El primer argumento es la variable que rompe el total en grupos (los nombres en nuestro caso); luego, necesitas los valores, el número de nacimientos y, finalmente, la variable que indica la evolución temporal (los años del histórico). El resto son modificaciones estéticas para mostrar y... ¡listo!



El gráfico que genera es interactivo, por lo que te animo que lo pruebes y veas la potencia y belleza de los resultados:



Conclusiones

X Edix Educación

En este fastbook hemos dado un repaso global a las principales **visualizaciones enfocadas en series evolutivas**. Algunas de ellas punteras como el denominado *streamgraph* o gráfico de flujo. Estos gráficos nos permiten analizar el desarrollo de una variable numérica a lo largo de un eje temporal, además de obtener información sobre su tendencia o cambio de niveles. De todas formas, algunas visualizaciones como los **gráficos de área apilados** tienen sus luces y sombras, por lo que tienes que vigilar siempre cuando es adecuado o no usarlos.

Cómo 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 algunas conclusiones.

Bibliografía

X Edix Educación

- [Ggplot2 – Página principal](#)

- [Api de Matplotlib – Backend de Seaborn](#)

¡Enhорabuena! Fastbook superado

edix

Creamos Digital Workers