

Fastbook 10

Visual Analytics

Gráficos big data



10. Gráficos big data



En este fastbook vamos a descubrir una gama de gráficos muy útiles cuando la cantidad de datos o variables se vuelve inmanejable con las opciones que ya hemos aprendido. El objetivo de estas visualizaciones es resumir una masa ingente de información y preservar la capacidad de exhibir patrones en los datos que nos ayuden a indagar y obtener aprendizajes.

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 ridgeline o joyplot: descripción](#)

[Heatmaps en R](#)

[Gráficos ridgeline en R](#)

[Gráficos de densidad 2D: descripción](#)

[Correlogramas: descripción](#)

[Gráficos de densidad 2D en R](#)

[Correlogramas en R](#)

[Conclusiones](#)

[Heatmaps: descripción](#)

[Bibliografía](#)

Gráficos ridgeline o joyplot: descripción

 Edix Educación



Un **gráfico ridgeline** es un conjunto de diagramas de densidad apilados sobre el mismo eje con una ligera superposición.

Permite visualizar un número elevado de grupos o categorías de una forma limpia y condensada que no dificulta la legibilidad del resultado. Cuando los **facets** no te permitan obtener un resultado útil, esta será tu mejor opción.

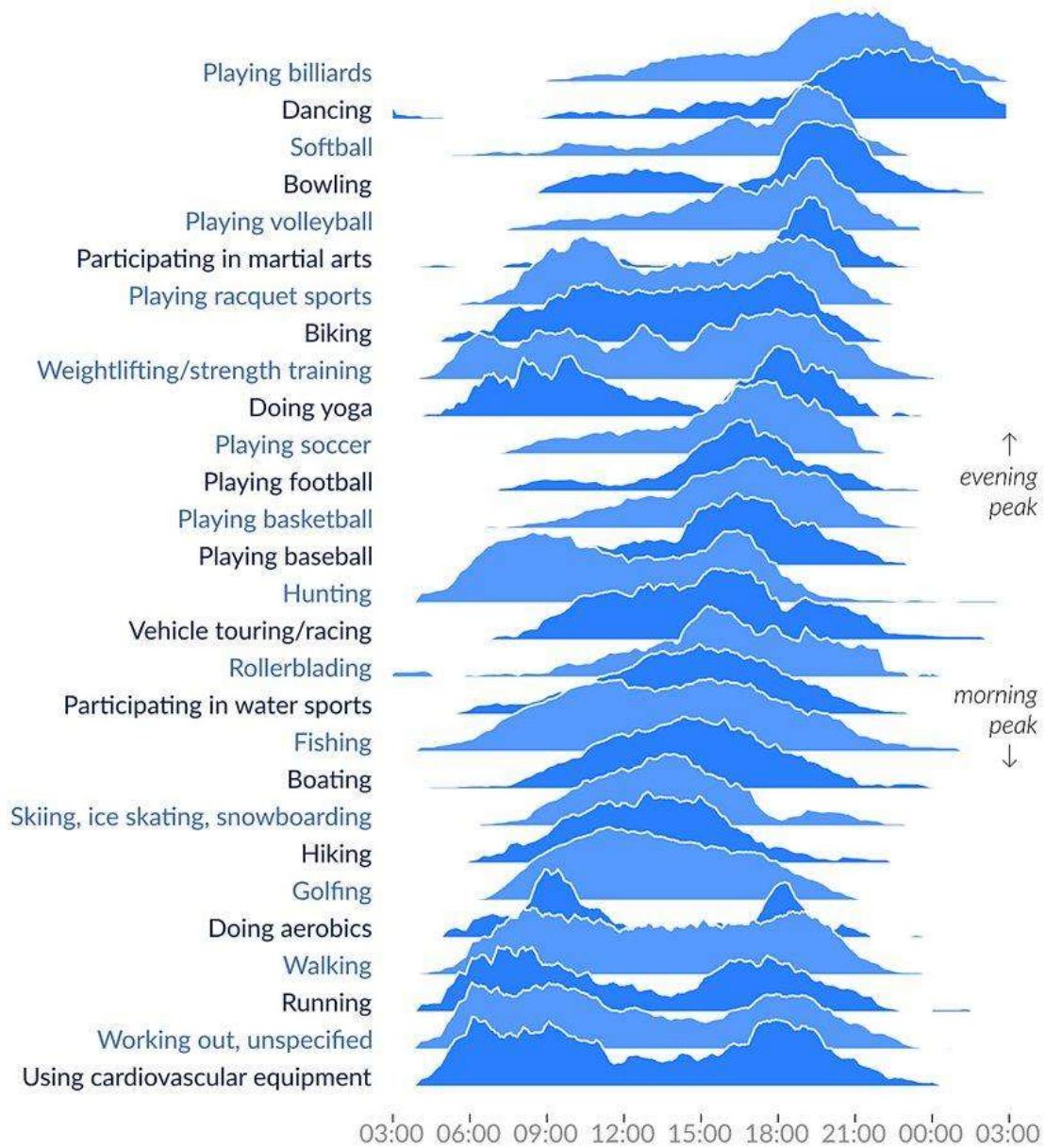
Para profundizar en este concepto veamos un ejemplo de este gráfico:

En el gráfico, podemos ver una visualización de **Henrik Lindberg** (un consejo: síguelo en Twitter), responsable del nombre **joyplot**.

¿Por qué ese nombre? Pues, porque su visualización no solo describe multitud de opciones de ocio y deporte para una muestra de personas con las que ‘disfrutan’; si no que es una referencia a [una de las portadas más míticas de la música de Joy Division](#).

Peak time of day for sports and leisure

Number of participants throughout the day compared to peak popularity.
Note the morning-and-evening everyday workouts, the midday hobbies,
and the evenings/late nights out.



@hnrikLndbrg | Source: American Time Use Survey

Fuente: [Henrik Lindberg](#).

Volviendo a la imagen, salta a la vista que el gráfico maneja perfectamente una gran cantidad de información. En la parte superior encontramos las actividades que tienen su pico hacia la noche, mientras que, en la parte inferior, tienes las actividades centradas en las primeras horas del día (aunque hay algunas que exhiben comportamientos bimodales).

Algunos consejos sobre estas visualizaciones:

- Puede haber problemas con el grado de solapamiento en algunos grupos, ocasionando que se oculte información, por ejemplo: no estamos seguros de lo que ocurre entre la mañana y tarde con la gente que hace aeróbic.
- Al igual que como veíamos con los diagramas de densidad o histogramas (también se puede realizar la visualización utilizando estos últimos) existen parámetros específicos que deben ser revisados con cuidado para obtener el mejor resultado.
- Como casi siempre, existen variaciones que apuntan a reducir estos problemas, pero vamos poco a poco, ¿no?

Veamos cómo podemos conseguir esta maravilla de visualización con R, ¡vamos a ello!

Gráficos ridgeline en R

X Edix Educación

Esto ya te lo sabes de memoria, cargamos los elementos clave para poder trabajar. En esta ocasión, añadimos una librería extra (*ggridges*) y el conjunto de datos sobre la emisión de CO₂ que producen los distintos tipos de alimentos.

Puedes descargar los datos [aquí](#).

```
# 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 la librería específica (Necesitarás instalarla primero!)
library(ggridges)

# Cargamos los datos para este ejercicio

food <- read_csv("data/food_consumption.csv")
```

Algunos apuntes antes de entrar en faena:

- No existen geoms específicos para este tipo de visualizaciones en ggplot pero, que no cunda el pánico, [Claus Wilke](#) (biólogo especialista y autor de *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling*) creó un paquete específico para solucionar este problema y tener acceso de una forma muy fácil a este tipo de gráficos.
- Los [datos](#) que utilizaremos provienen del proyecto [TidyTuesday](#), al que te recomiendo encarecidamente que le eches un vistazo. Es un proyecto semanal, que comparte un set de datos y hace énfasis en el entendimiento de estos, mediante transformaciones y visualizaciones utilizando la **suite tidyverse**. ¡Ideal para los que se introducen en este mundillo!

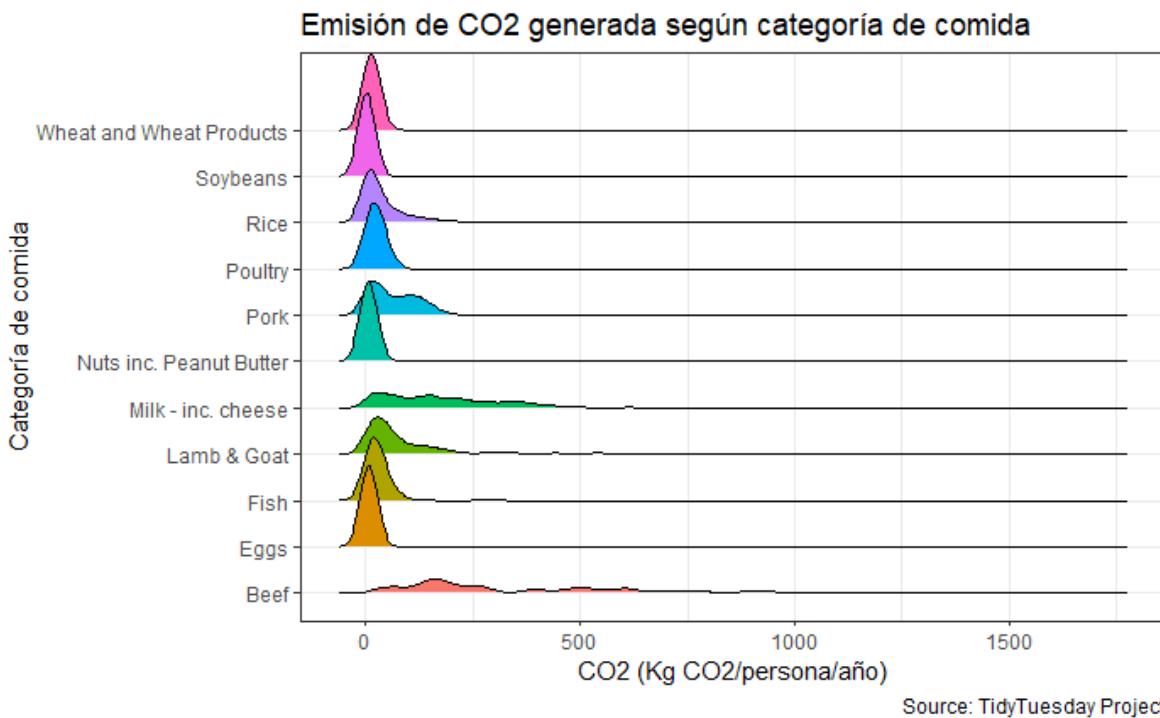
Vamos a aprender cómo manejar esta excelente visualización.

```
# Tu primer Ridgeline plot

food %>%
  filter(co2_emmission >= 1) %>%
  ggplot(aes(x=co2_emmission, y=food_category, fill=food_category))+
  geom_density_ridges()+
  labs(title = "Emisión de CO2 generada según categoría de comida",
       caption = "Source: TidyTuesday Project",
       y = "Categoría de comida",
       x = "CO2 (Kg CO2/persona/año)") +
  theme(legend.position = "none")
```

Esta librería te aporta el `geom_density_ridges()` que utilizamos para esta visualización. Previo a eso, hemos filtrado datos con poco interés y que solo cargan la altura de la distribución en torno a 0.

Solo necesitas especificar una **variable numérica** (de la que nos interesa su distribución), una **variable categórica** (que romperá los datos en diferentes grupos) y otra que **rellene los diagramas de densidad** (normalmente, la misma categórica).



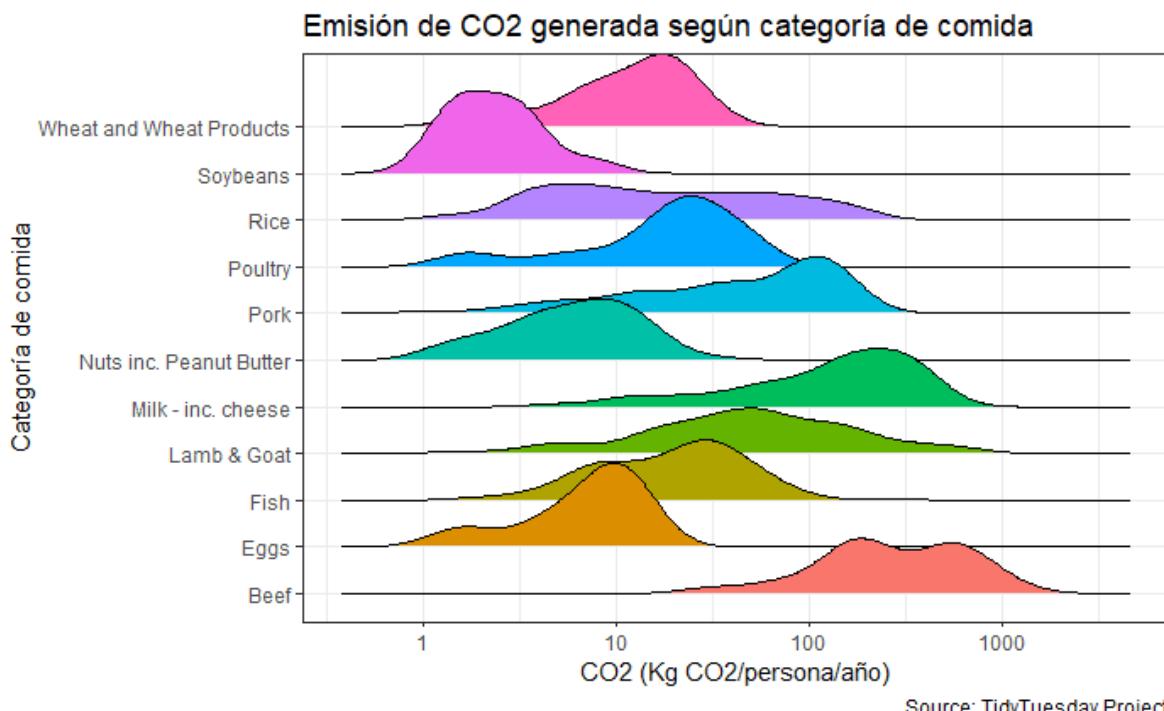
Para empezar, observamos que obtenemos un gráfico muy acotado a la izquierda, esto es debido a la presencia de valores realmente grandes en alguna de las categorías.

Solucionemos este problema:

```
# Tu primer Ridgeline plot

food %>%
  filter(co2_emmission >= 1) %>%
  ggplot(aes(x=co2_emmission, y=food_category, fill=food_category))+
  geom_density_ridges()+
  labs(title = "Emisión de CO2 generada según categoría de comida",
       caption = "Source: TidyTuesday Project",
       y = "Categoria de comida",
       x = "CO2 (Kg CO2/persona/año)") +
  theme(legend.position = "none")
```

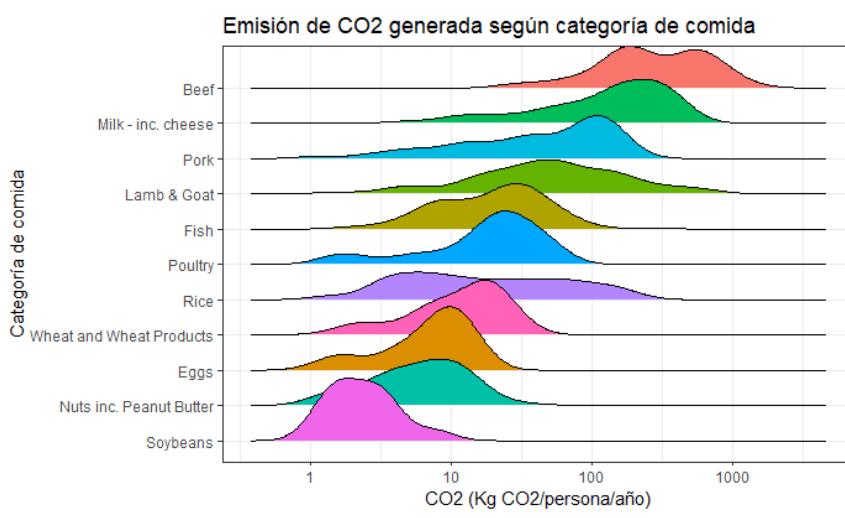
Con el uso de `scale_x_continuous(trans = "log10")` aplicamos una transformación logarítmica en el eje numérico que mejora la visualización.



Ahora es más sencilla la interpretación, pero todavía se puede mejorar algo más (si tienes en mente los **lollipop charts** ya deberías saber por dónde voy...):

```
# Ordenando para un mejor resultado visual
food %>%
  filter(co2_emmission >= 1) %>%
  ggplot(aes(x=co2_emmission,
             y=fct_reorder(food_category,co2_emmission),
             fill=food_category))+ 
  geom_density_ridges()+
  scale_x_continuous(trans="log10") +
  labs(title = "Emisión de CO2 generada según categoría de comida",
       caption = "Source: TidyTuesday Project",
       y = "Categoría de comida",
       x = "CO2 (Kg CO2/persona/año)") +
  theme(legend.position = "none")
```

Ahora es muy sencillo para cualquier persona **extraer conclusiones fácilmente**: la ternera y la leche (y derivados), ambos procedentes del mismo animal, son los alimentos que más CO2 generan en su producción, seguidos muy de cerca del cerdo. En la cola opuesta, la soja es el alimento más respetuoso con la capa de ozono (los veganos obtienen un punto en esta ocasión...).



Source: TidyTuesday Project

Un **consejo extra**, uno de los problemas de esta visualización es que la superposición de los diagramas de densidad puede ocultar información clave (lo veíamos en la imagen de

presentación). Para evitar este problema, R y la librería ggridges tienen una magnífica solución:

```
# Reduciendo/eliminando el overlapping
food %>%
  filter(co2_emmission >= 1) %>%
  ggplot(aes(x=co2_emmission,
             y=fct_reorder(food_category,co2_emmission),
             fill=food_category))+

  geom_density_ridges(scale = 1)+

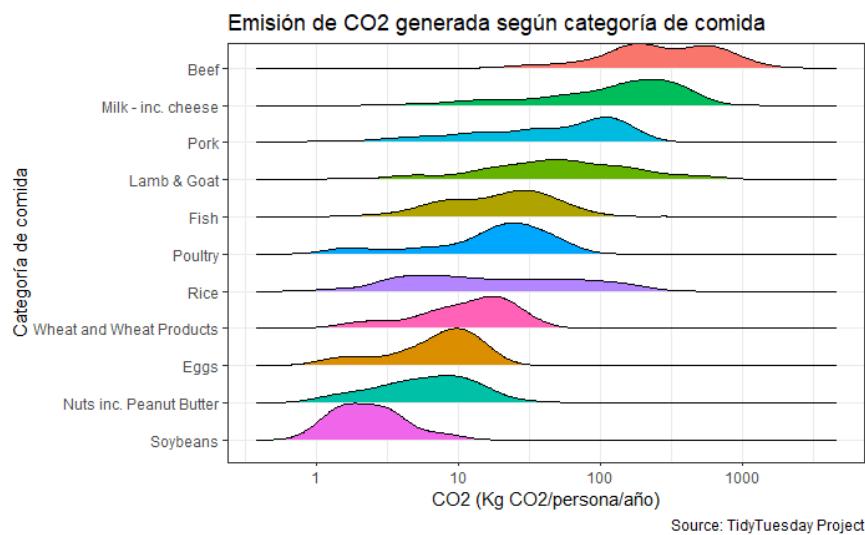
  scale_x_continuous(trans="log10") +

  labs(title = "Emisión de CO2 generada según categoría de comida",
       caption = "Source: TidyTuesday Project",
       y = "Categoria de comida",
       x = "CO2 (Kg CO2/persona/año)") +

  theme(legend.position = "none")
```

Con el parámetro `scale = 1` en el geom haces que el punto más alto de una distribución esté tocando, como mucho, el siguiente (en este caso lo puedes ver en los huevos).

Si utilizas un valor inferior a 1 separarás los diagramas aún más, valores por encima de 1 generará superposición. Estarás de acuerdo que esto soluciona el problema, pero se ha perdido la ‘alegría’ que transmiten estos gráficos, ¿no? **Úsalo solo cuando se oculta información.**

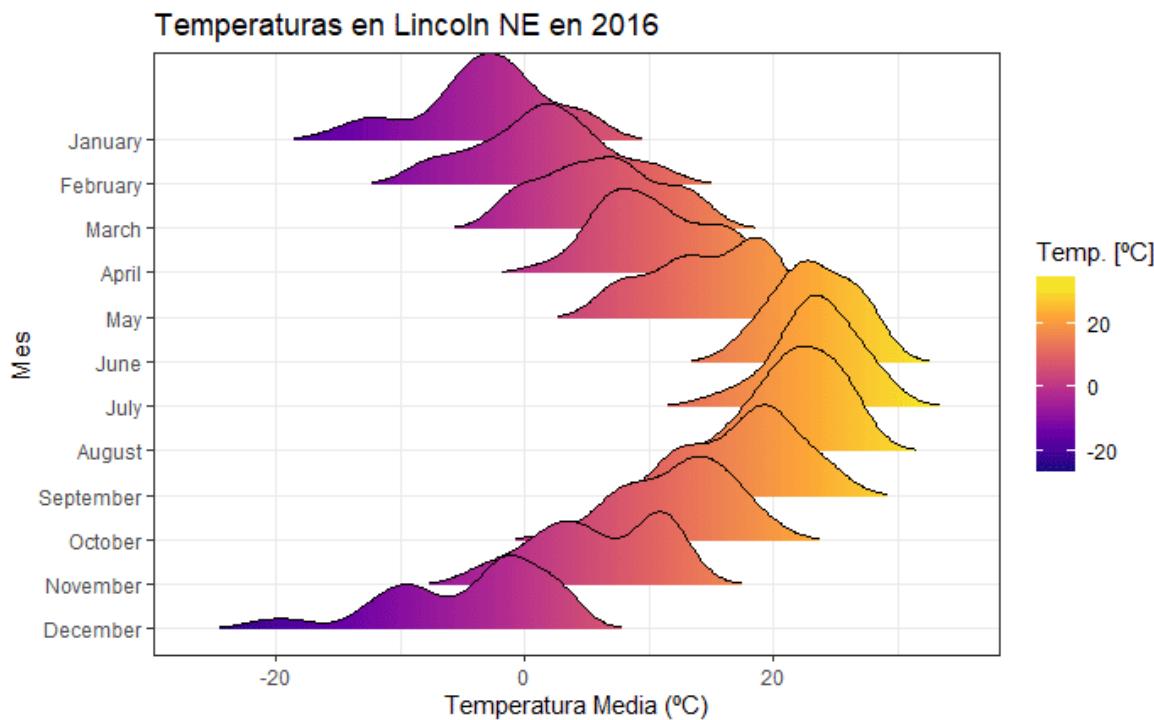


Finalmente, esta visualización es muy útil para analizar el desarrollo de una variable numérica en un eje temporal. Veamos un ejemplo:

```
# Ridgeplots con escala temporal

lincoln_weather %>%
  mutate(grados = (`Mean Temperature [F]` - 32) / 1.8) %>%
  # ggplot(aes(x = grados, y = Month, fill = stat(x))) +
  ggplot(aes(x = grados, y = Month, fill = stat(x))) +
  geom_density_ridges_gradient(scale = 2.5, rel_min_height = 0.01) +
  scale_fill_viridis_c(name = "Temp. [°C]", option = "C") +
  labs(title = 'Temperaturas en Lincoln NE en 2016',
       x = "Temperatura Media (°C)",
       y = "Mes")
```

Utilizando el **dataset que incorpora ggridges** ([Lincoln weather](#)), que ya tendrás disponible en memoria una vez cargada la librería, podemos ilustrar este caso. En ese fichero, encontrarás las temperaturas registradas (entre otras muchas variables) durante 2016 en Lincoln, Nebraska. Lo único que hemos aplicado es una transformación a grados centígrados y hemos coloreado por la temperatura.



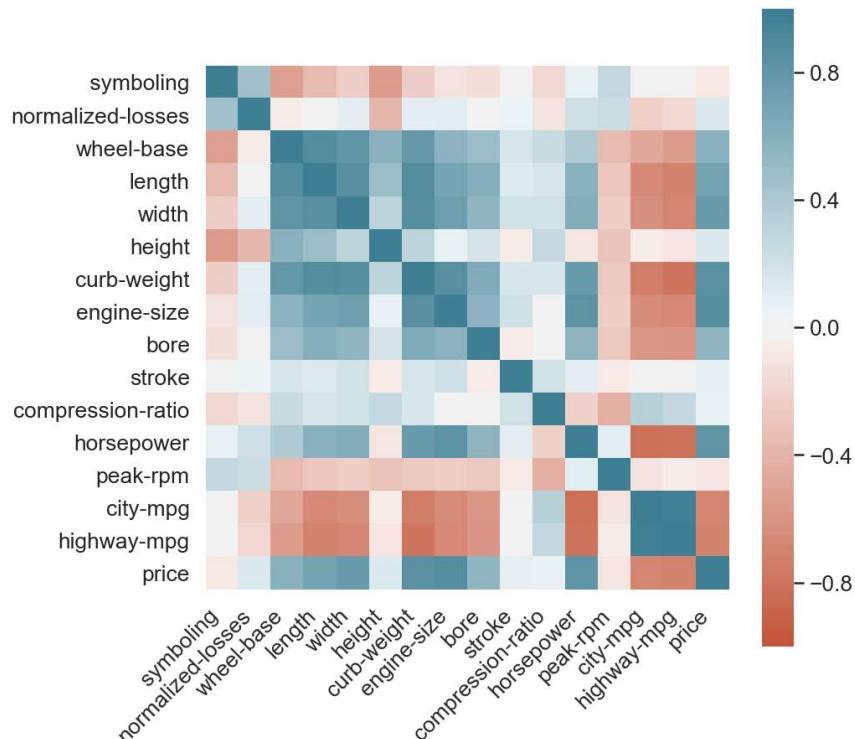
En este caso, ilustramos también el uso del parámetro `rel_min_height` que suprime las colas residuales de cada distribución. Un valor de **0.01** suele ser adecuado, pero puedes experimentar para entender cómo recorta la distribución y qué valor puede encajar mejor en tu caso.

Correlogramas: descripción

X Edix Educación

Un **correlograma** es un tipo de visualización que resume la relación entre pares de variables numéricas. Básicamente, es una matriz de correlaciones que **sustituye los coeficientes numéricos por una forma** (cuadrados, círculos, elipses...) **coloreada**, en función de la dirección de la relación; de este modo, se obtienen aprendizajes con un simple vistazo.

Observa la siguiente imagen para entender el porqué:



Fuente: [TowardsDataScience](#).

Los correlogramas son herramientas muy útiles en cualquier **análisis exploratorio**, ya que permiten analizar todo el dataset en un segundo (esto es discutible). Apúntalo como un **gráfico obligatorio**, siempre que analices un nuevo conjunto de datos.

En el caso anterior, los cuadrados con colores azules fuertes indican una **fuerte relación directa** entre esas variables (*length – width*); los colores rojos fuertes indican una relación indirecta (*horsepower – city-mpg*) y, por último, aquellos que apenas tienen color destacan variables que **no poseen relación lineal** entre ellas (*normalized-losses* con la mayoría de las variables).

Algunos consejos sobre estas visualizaciones:

- Superar las **20 variables** puede hacer que el gráfico sea **ilegible**. Detallaremos algunos consejos para mejorar estos casos (aunque ya te advierto que no hacen milagros...).
- Este gráfico puede sufrir a la hora de calcular las correlaciones pareadas, si la **muestra de datos es gigantesca** (>10M de filas). Puede necesitar gran capacidad de computación cuando la muestra es muy grande.
- Existen **variaciones que combinan scatterplots con histogramas** para ofrecer más detalle y resaltar ciertos patrones.

Vamos directamente a practicar esto, una buena manera de reforzar los conocimientos.

Correlogramas en R

X Edix Educación

Para construir esta visualización en R nos apoyaremos en los datos de [mtcars](#), un dataset incluido de base en R.

Su mejor propiedad es que está poblado de variables numéricas y, por tanto, será perfecto para mostrar el potencial de esta visualización:

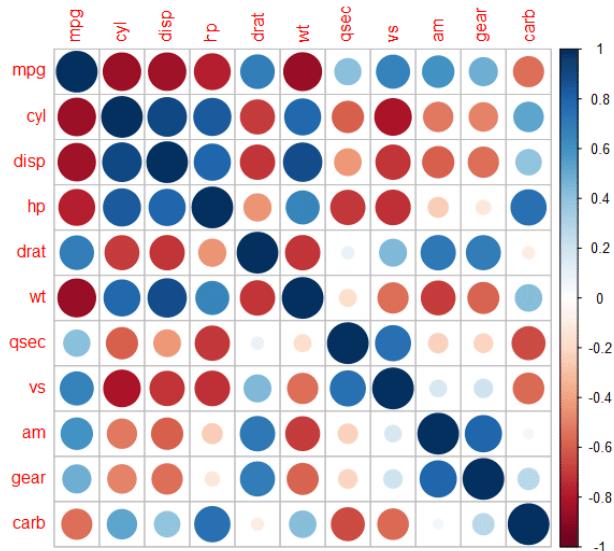
```
# Nuestro primer correograma
library(corrplot)

m_cor ← cor(mtcars)

# Generamos la visualización
corrplot(m_cor, method = "circle")
```

Muy sencillo, ¿no? Solo necesitamos instalar una librería y tendremos acceso a un poderoso gráfico con una sola línea. El único requisito que necesita la función `corrplot()` es recibir una matriz de correlaciones.

¡Por cierto! **Asegúrate de que únicamente la construyes con variables numéricas para no provocar errores.** En nuestro caso, todas las variables son numéricas y no hay que realizar operaciones extra.



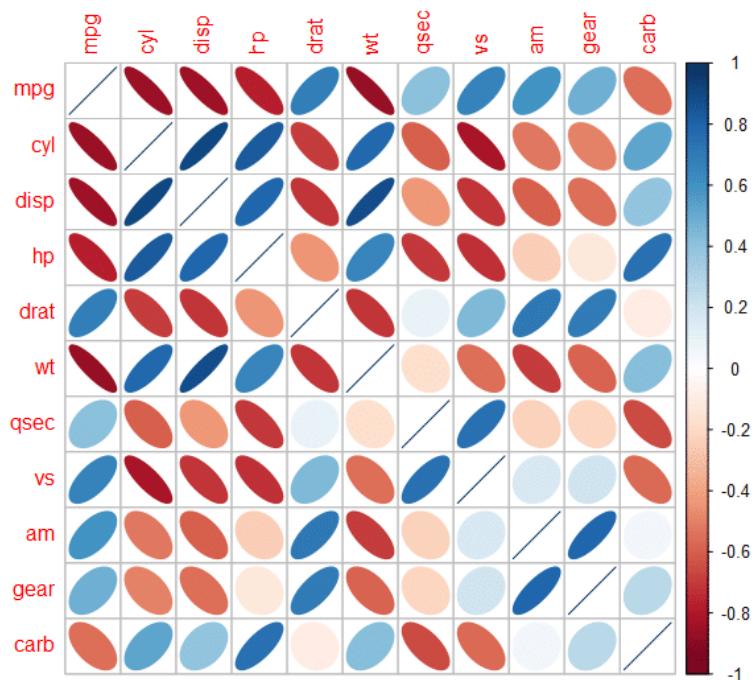
El resultado es una matriz completa de círculos coloreados en función de la intensidad de la correlación entre cada par de variables. En este caso, por ejemplo, existe alta correlación entre la cilindrada del vehículo (cyl) y la potencia en caballos de este (hp); en cambio, la cilindrada tiene correlación indirecta con el mpg (Millas / (US) galón).

Esta visualización es una poderosa herramienta para detectar relaciones lineales interesantes entre las variables que conforman nuestro conjunto de datos. Veamos algunos retoques adicionales para mejorar este ejemplo.

Empezamos con el estilo para mostrar los índices de correlación: por defecto son círculos como en el caso de arriba, pero tienes a tu disposición **7 modalidades** ('circle', 'square', 'ellipse', 'number', 'shade', 'color', 'pie'). En todos ellos, las correlaciones se muestran en azul y rojo según su intensidad (indirecta y directa, respectivamente).

```
# Generamos la visualización  
corrplot(m_cor, method = "ellipse")
```

Personalmente, mi favorito son las **elipses** por su rápida interpretación (mientras más delgada sea la línea más fuerte es la relación). Te animo a probar el resto y decidir cuál te resulta más claro de entender.

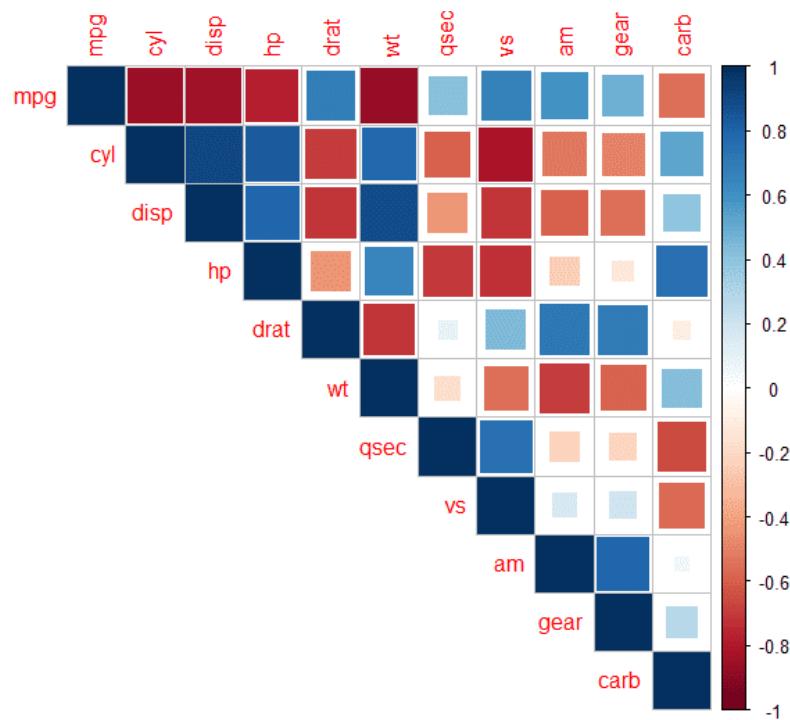


Otro punto importante, es la **forma del correograma**; al ser simétrico, se suele descartar una de las dos partes, superior o inferior.

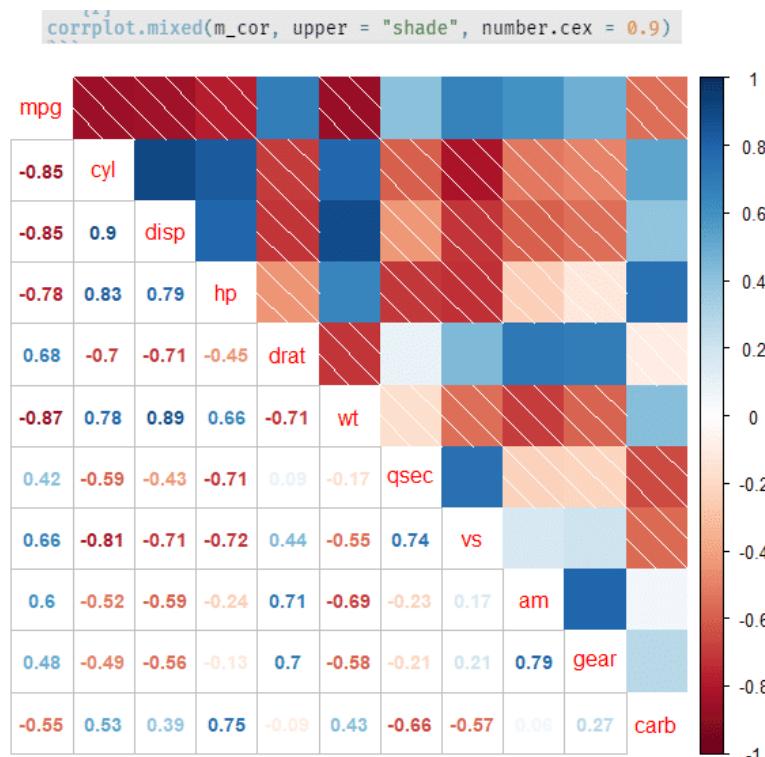
Veamos un ejemplo:

```
corrplot(m_cor, type = "upper", method = "square")
```

Mucho más sencilla de leer, ¿no?



Tienes la opción ‘full’ por defecto, pero puedes escoger entre ‘lower’ o ‘upper’ para mejorar el resultado visual.



Esta librería ofrece también la función `corrplot.mixed()` que combina dos visiones en la misma matriz. Por defecto, arriba utiliza alguno de los métodos que hemos visto anteriormente y abajo refleja los índices de correlación para una lectura total. Practica con ella hasta que encuentres la combinación perfecta.

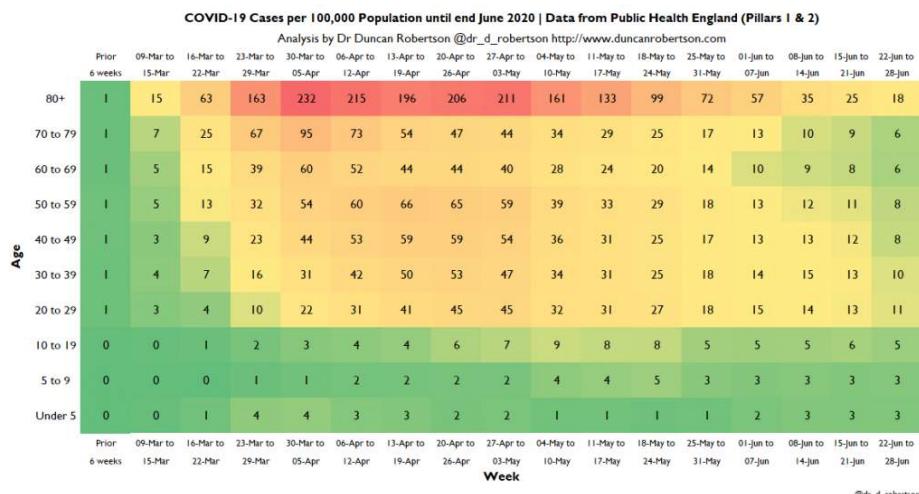
Heatmaps: descripción

Edix Educación

Continuamos con un gráfico que tiene un alto parecido con los correlogramas, hablamos de los **heatmaps o mapas de calor**.

Esta visualización es una representación visual de los datos mediante un **mapa de colores** que permite abstraerse de los datos específicos y ver la información de manera global. Son buenos **revelando variaciones** a lo largo de múltiples variables, **revelando patrones, identificando variables** similares e, incluso, **detectando correlaciones**.

Observa la siguiente imagen para entender el porqué:



Fuente: [DuncanRobertson](#)

Como en la visualización anterior, los heatmaps son **realmente útiles en los análisis exploratorios**, debido a la visión general de la información que ofrecen de un solo plumazo. Normalmente, en uno de los ejes se sitúa una **serie de grupos** y, en el otro, las diferentes **variables numéricas**. Este eje también puede adoptar la forma de un eje temporal (para reflejar evolución) y un punto de vista diferente.

En la imagen anterior, se puede apreciar el efecto del covid-19 en la población de Reino Unido, dividido por rango de edad para un periodo específico. Podemos localizar rápidamente que la ola afectó severamente a los mayores de 80 años, siendo la semana del 30 de marzo al 5 de abril la más crítica.

Algunos consejos sobre estas visualizaciones:

- El uso de una **paleta de colores adecuada** es clave en esta visualización para que captar los patrones sea fácil e intuitivo.
- Es una **alternativa** muy potente para **visualizar series temporales** que se repiten, por ejemplo: las series diarias fragmentadas a nivel mensual mediante facets.
- En algunos casos, puede resultar clave **normalizar los datos** previamente al gráfico para evitar que ciertas observaciones absorban todo el color de la columna debido a las escalas utilizadas (imagina comparar población de países y que aparezca China, apenas se verán variaciones en el resto de los países).

Veamos cómo se construyen estas visualizaciones.

Heatmaps en R

X Edix Educación

A la hora de ilustrar el heatmap en R, utilizaremos el mismo dataset de **mtcars** usado en el apartado de correlogramas.

Para realizar la magia es necesario un ligero tratamiento de datos: básicamente convertir todas las **columnas numéricas a formato long** (repasa estos conceptos en los primeros fastbooks, si no te acuerdas).

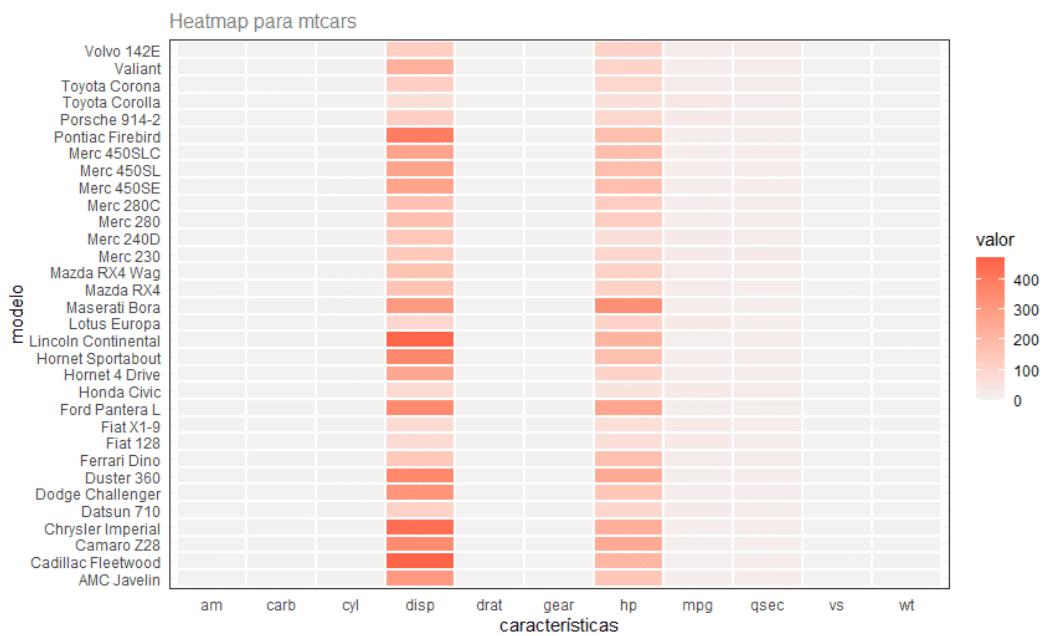
```
## Tu primer Heatmap

## Preparamos los datos para el gráfico
# Pasamos los nombres de los modelos a una nueva columna
carsdf ← mtcars
carsdf$modelo = rownames(mtcars)

# Redefinimos el formato de los datos a long
carsdf ← carsdf %>% pivot_longer(cols = (mpg:carb), names_to = "variable",
                                      values_to = "valor")

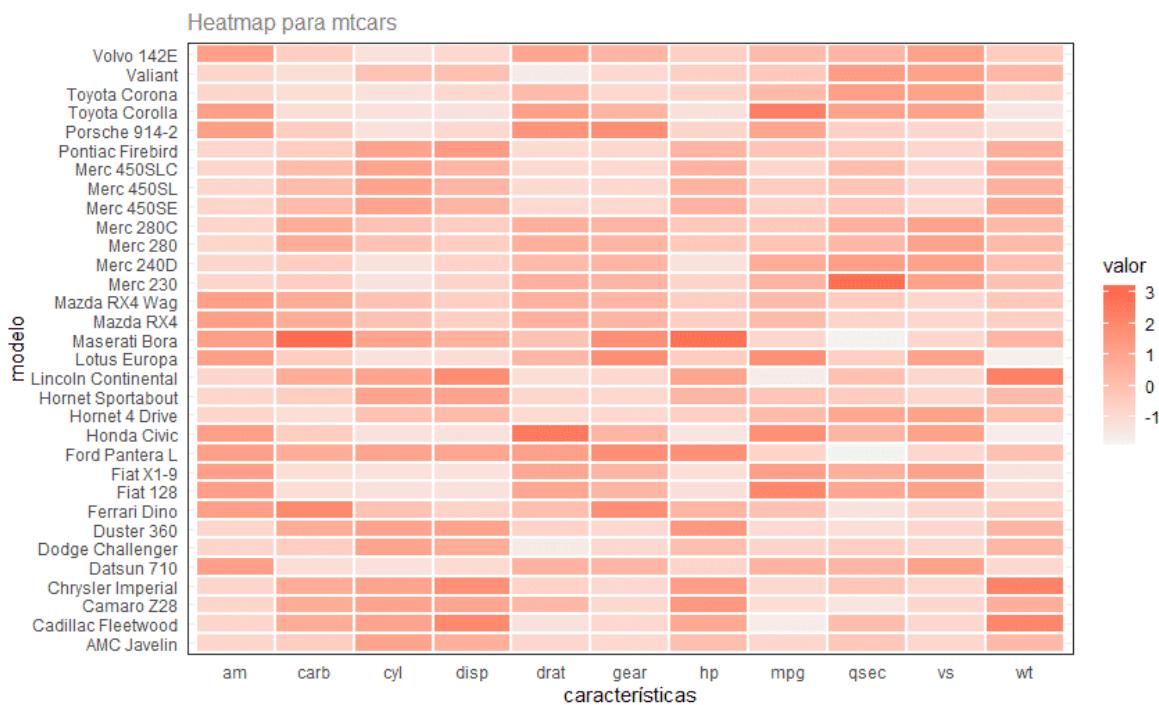
## Pintamos el resultado
ggplot(data = carsdf, aes(x = variable, y = modelo)) +
  geom_tile(aes(fill = valor), color = "white", size = 1) +
  scale_fill_gradient(low = "gray95", high = "tomato") +
  xlab("características") +
  ggtitle("Heatmap para mtcars") +
  theme(axis.ticks = element_blank(),
        panel.background = element_blank(),
        plot.title = element_text(size = 12, colour = "gray50"))
```

El geom que nos permite generar el mapa de calor es `geom_tile()`, que necesita las dos variables que van a generar la malla a colorear. Los valores de rellenos son los propios valores de cada variable. El resto del código es estético.



El único problema que observamos es que, al parecer, dos columnas absorben todo el color del cuadro (*disp* y *hp*). Para evitar estos problemas, basta con que **escalemos el conjunto de datos con esta función:**

Ahora resulta muy fácil detectar patrones e identificar vehículos que destacan en numerosos atributos como, por ejemplo, el Maserati Bora.



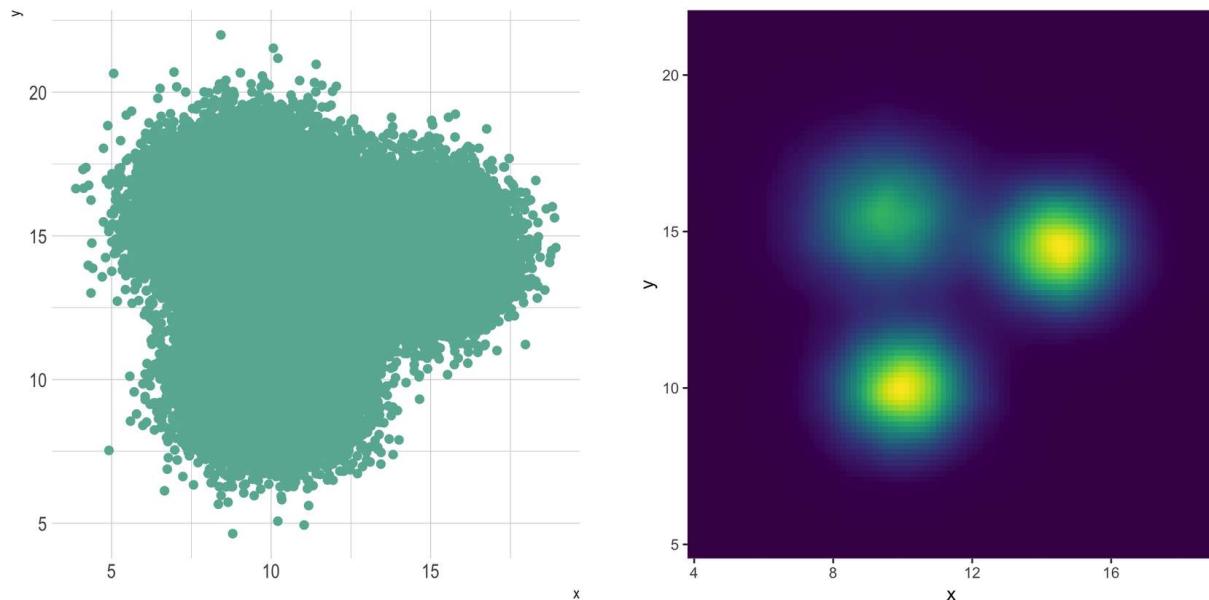
Gráficos de densidad 2D: descripción

X Edix Educación

Un **gráfico de densidad 2D** exhibe la relación entre dos variables numéricas a través de un eje cartesiano.

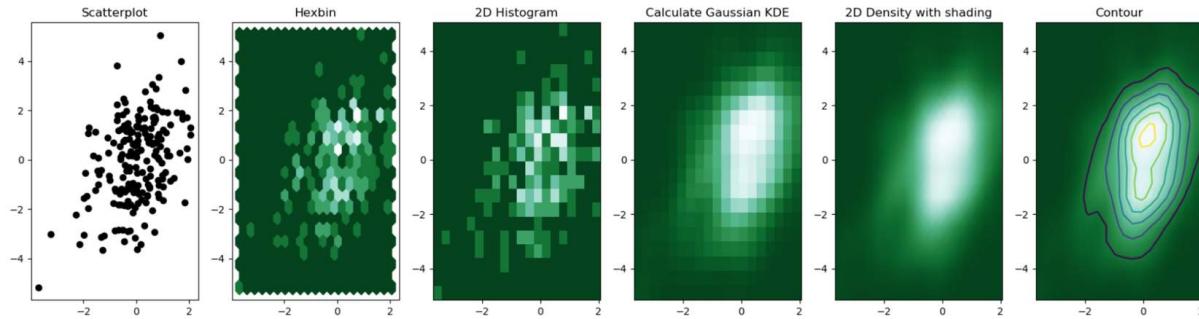
Con esta definición, si llevas la asignatura bien estudiada, debería sonarte a que es la misma que utilizábamos para el **scatterplot**. No es un error, la única diferencia es que este gráfico añade una capa de agregación por área, cuya función es **contar el número de observaciones en el interior y aplicar un color** en base al mismo.

Observa la siguiente imagen para entender el porqué:



Fuente: [Data-to-viz](#).

En función del **tipo de área** utilizada para encerrar las observaciones, obtendremos diferentes tipos de visualizaciones:



Fuente: [Data-to-viz](#).

Esta tipología de gráfico es adecuada de utilizar para evitar lo que se conoce como *overplotting*.

Este fenómeno ocurre cuando el conjunto de datos utilizado es muy grande. Esto consigue que nuestros scatterplots generen zonas muy solapadas de puntos y sea imposible leer el gráfico.

Considera su uso siempre que te enfrentes a una situación como la de la imagen izquierda inicial que veíamos.

Algunos consejos sobre estas visualizaciones:

- Como en los **heatmaps**, el uso de una **paleta de colores adecuada** es clave en esta visualización para que captar patrones sea fácil e intuitivo.
- Solo es útil si la cantidad de datos es gigantesca.** Utilizarlo en conjuntos de datos pequeños es un error, pues existen mejores alternativas.
- Jugar con los parámetros que definen o suavizan el área** es clave para obtener resultados adecuados.

Intentemos reproducir ejemplos con nuestras herramientas.

Gráficos de densidad 2D en R

X Edix Educación

Vamos a ilustrar esta visualización en R mediante la generación masiva de datos artificiales (siempre viene bien conocer esta metodología):

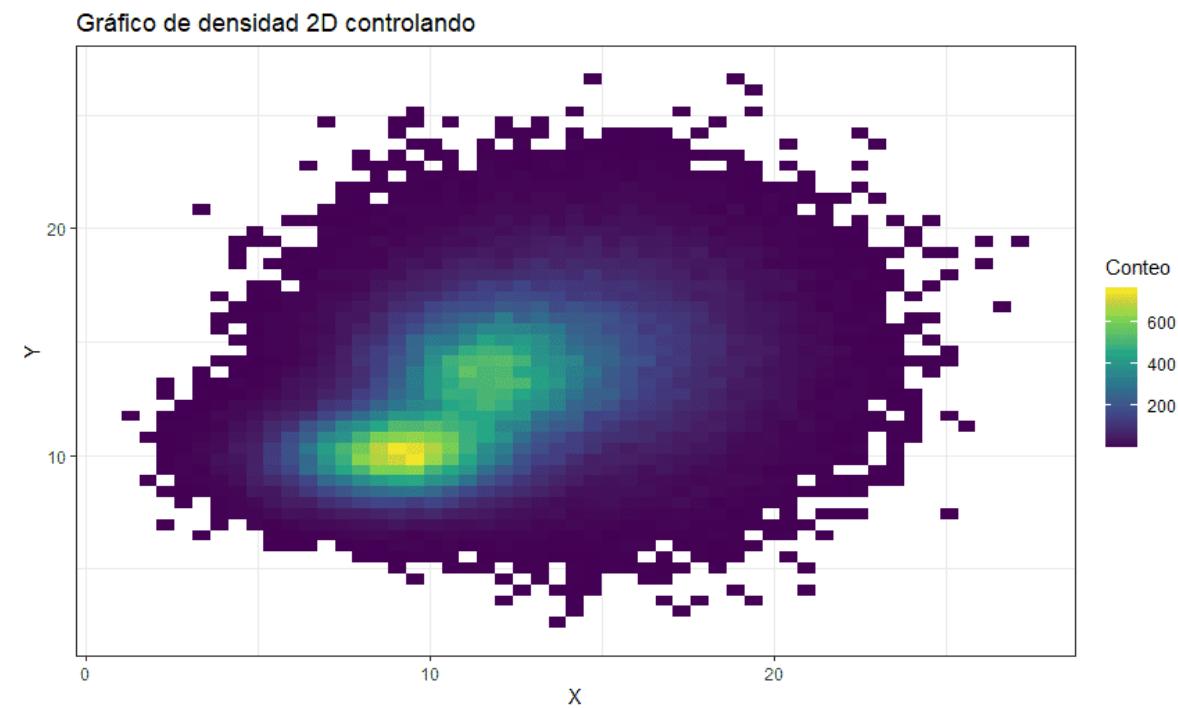
```
# Datos artificiales
set.seed(42)
a <- data.frame(x=rnorm(40000, 9, 1.9), y=rnorm(40000, 10, 1.2) )
b <- data.frame(x=rnorm(40000, 14.5, 3), y=rnorm(40000, 14.5, 3) )
c <- data.frame(x=rnorm(40000, 11.5, 1.9), y=rnorm(40000, 13.5, 1.9) )

# Unificamos los datos
data <- rbind(a,b,c)

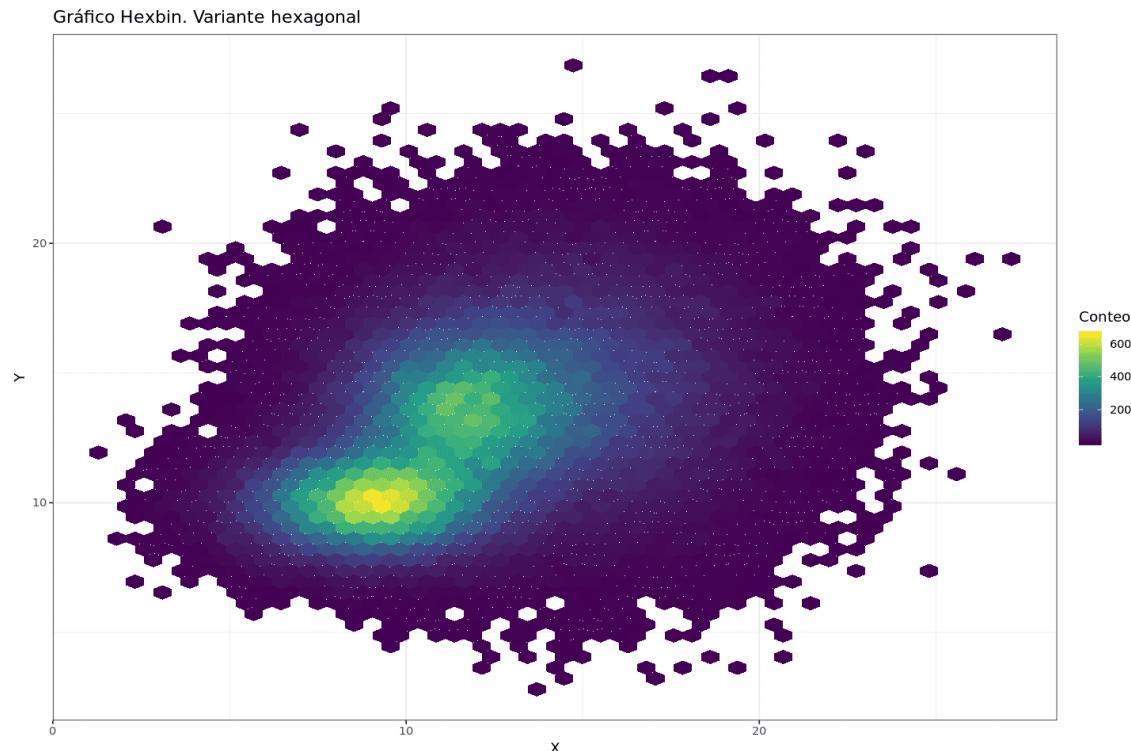
# Tu primer 2d density chart
ggplot(data, aes(x=x, y=y) ) +
  geom_bin2d(bins = 50) +
  scale_fill_continuous(type = "viridis") +
  theme_bw() +
  labs(title = "Gráfico de densidad 2D controlando",
       x = "X",
       y = "Y",
       fill = "Conteo")
```

Adapta el número de observaciones (40.000) a los requisitos de tu ordenador para evitar problemas en la generación.

Si buscas pintar esto, necesitarás el `geom_bin2d()`, que tiene un parámetro de `bins` para controlar el grosor de las áreas: mientras más alto, más granular serán las parcelas para contar datos.



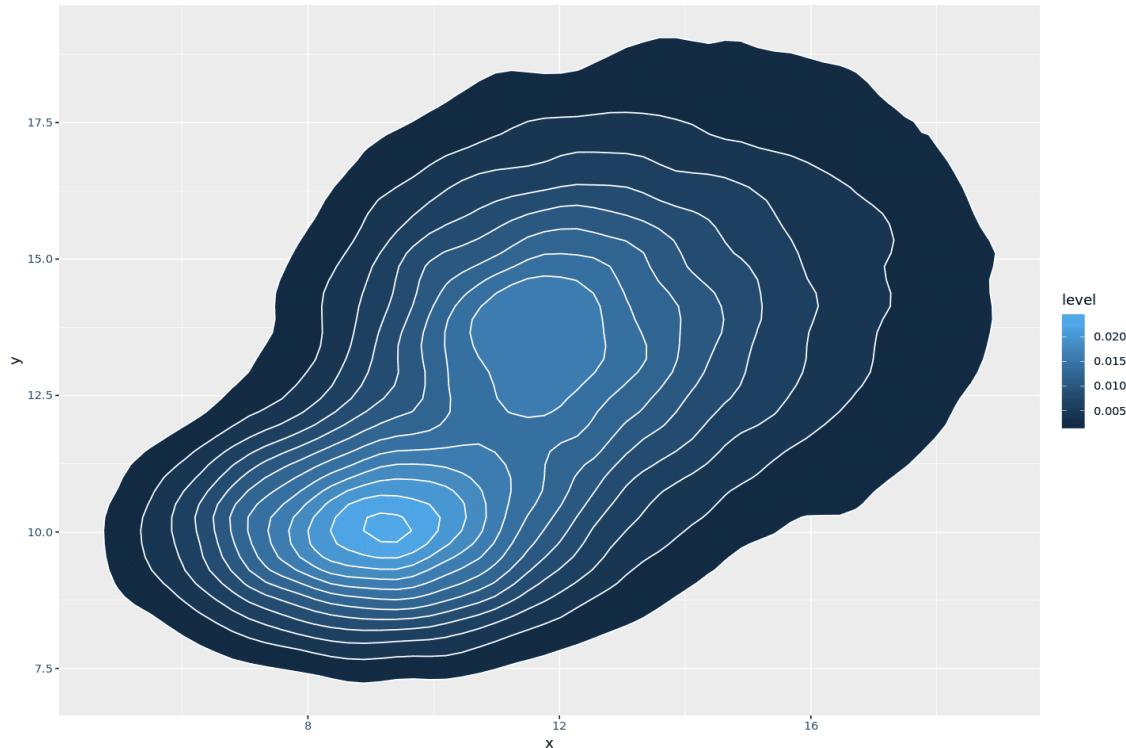
Existe una variación cuando utilizamos hexágonos en lugar de cuadrados como parcelas.
Solo necesitas utilizar `geom_hex()`:



Por último, también podemos utilizar contornos y áreas para representar estas densidades:

```
# Area + contour
ggplot(data, aes(x=x, y=y) ) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon", colour="white")
```

Basta con utilizar la función de [stat_density_2d\(\)](#).



Conclusiones

X Edix Educación

En este tema, hemos descubierto las **visualizaciones adecuadas para grandes conjuntos de datos**, tanto en número de observaciones como de variables. Algunas de ellas son **esenciales en cualquier análisis descriptivo** que realicemos, como los **correlogramas o heatmaps** (que permiten analizar patrones en grandes masas de información con un solo vistazo).

Otras visualizaciones, como los **joyplots o los gráficos 2D de densidad**, nos permiten analizar múltiples variables o enormes masas de puntos **preservando la legibilidad de las visualizaciones** de una forma muy estética.

Como siempre, te recomiendo encarecidamente que pruebes los resultados en cada opción y no te quedes solo con la lectura del **fastbook**. 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