

Fastbook 05

Visual Analytics

Box plots



05. Box plots

En este fastbook vamos a dominar uno de los gráficos más utilizados para **analizar la distribución de un conjunto de datos**. Para ello, comenzaremos analizando su uso y la lógica que hay detrás de este tipo de visualizaciones, entendiendo que **en su simpleza reside la fortaleza**. Aun así, también profundizaremos en sus puntos débiles y las alternativas que han ido surgiendo a lo largo de los años para reforzar estas carencias.

De nuevo, utilizaremos R para desarrollar toda la práctica. 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

[Box plots o diagramas de cajas y bigotes: descripción](#)

[Box plots en R](#)

[Conclusiones](#)

[Bibliografía](#)

Box plots o diagramas de cajas y bigotes: descripción

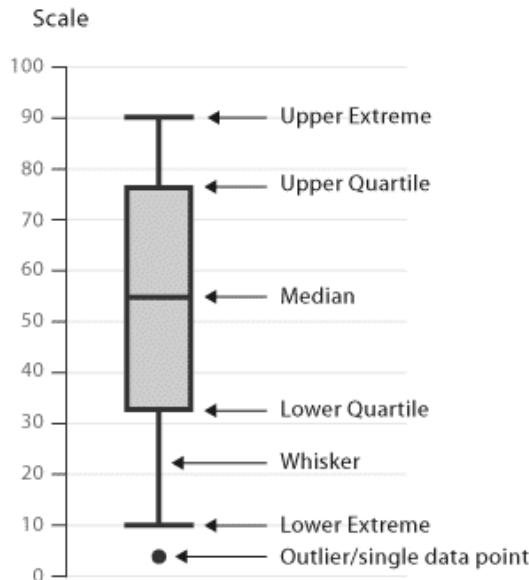
X Edix Educación

Un **box plot** (conocido como **diagrama de cajas y bigotes** —pronto entenderás esto— o, también, **de caja**) es un tipo de visualización utilizada para representar, de manera efectiva, la distribución de un conjunto de datos a través de sus cuartiles.

Es un gráfico que puede resultar raro al primer contacto, mucho más simple que otras visualizaciones centradas en la distribución de los datos como son los **histogramas** o los **gráficos de densidad**, que ya estudiaremos. Sin embargo, en esa simpleza reside su fortaleza a la hora de comparar múltiples grupos o conjuntos de datos. Vamos a analizar su funcionamiento.

El gráfico se compone de un **cuerpo central**, la **caja** y unas **líneas** que parten de la misma, cada una hacia un extremo, los **bigotes** (por su forma).

Y si no estás de acuerdo con el símil, las quejas a [Mary Eleanor Spear](#), ¡una de las primeras expertas de la historia en visualización de datos! O una reclamación al que impulsó su uso y forma, tal y como la conocemos hoy en día, a [John W. Tukey](#).

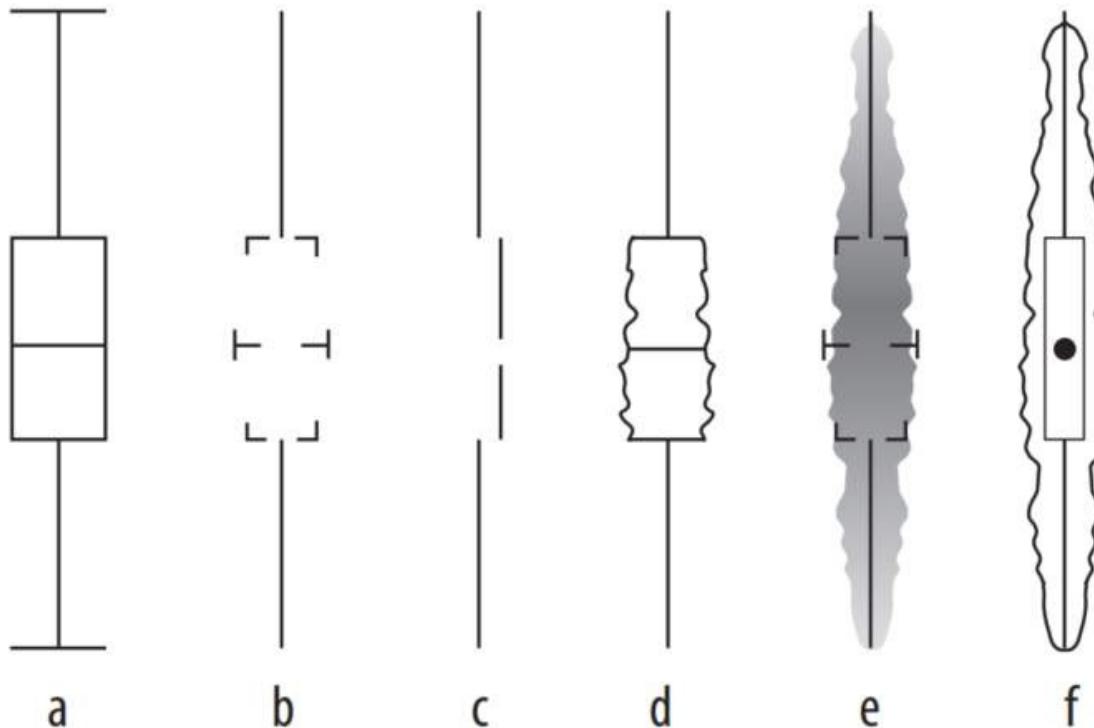


Fuente: [DataVizCatalogue](#).

Detallemos ahora qué significa cada parte:

- La **caja** representa el recorrido intercuartílico (**IQR**) de la variable que deseamos representar. Esto es la **diferencia entre el primer y tercer cuartil** (25%-75% de la distribución de los datos). Vendría a representar el ‘cuerpo’ de los datos contenidos en esa variable, la mayoría.
En el ejemplo superior, los datos de la variable estarían contenidos entre 32 y 76.
- La **línea central** de la caja marca la **mediana** de los datos, es decir, en qué valor se sitúa el corte que deja a cada lado un 50% de los datos.
En el ejemplo, la mediana estaría en 55.
- Los **bigotes** representan el ‘mínimo’ y ‘máximo’ de la distribución de los datos. Las comillas no son casualidad, y es que los bigotes se calculan de la siguiente forma:
 - Bigote inferior:** $Q_1 - 1,5 * IQR$
 - Bigote superior:** $Q_3 + 1,5 * IQR$
- Finalmente, los **outliers** o datos atípicos son todos aquellos puntos que exceden estos límites teóricos que establecen los bigotes anteriores.

En cuanto a su **estilo**, existen muchas variaciones que han ido surgiendo a lo largo de la historia: algunas para simplificar la visualización aún más, otras versiones buscando complementar carencias del diseño original. Echa un vistazo a la siguiente imagen.



Fuente: [Data Visualization de Kieran Healy](#).

Son seis tipos diferentes de box plots. El más habitual y el que utilizaremos en nuestro caso es el A, denominado de **tukey** por ser [John W. Tukey](#), su creador. Existen otras versiones como la C, originalmente diseñada por **Tufte** (¿recuerdas a Edward Tufte de los primeros fastbooks y su obsesión por la simplificación?) y que buscaba la simpleza absoluta, tanto es así que puede resultar hasta complejo de leer... ¿Qué opinas?

Ya tienes los conceptos básicos de tu lado. Ahora prepara tus herramientas, ¡vamos a practicar!

Box plots en R

X Edix Educación

Al igual que con el fastbook anterior, comenzamos cargando algunas opciones básicas para trabajar de una manera efectiva.

```
# 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")
```

Comencemos con el código más básico.

```
```{r}
Empezamos con la versión simple

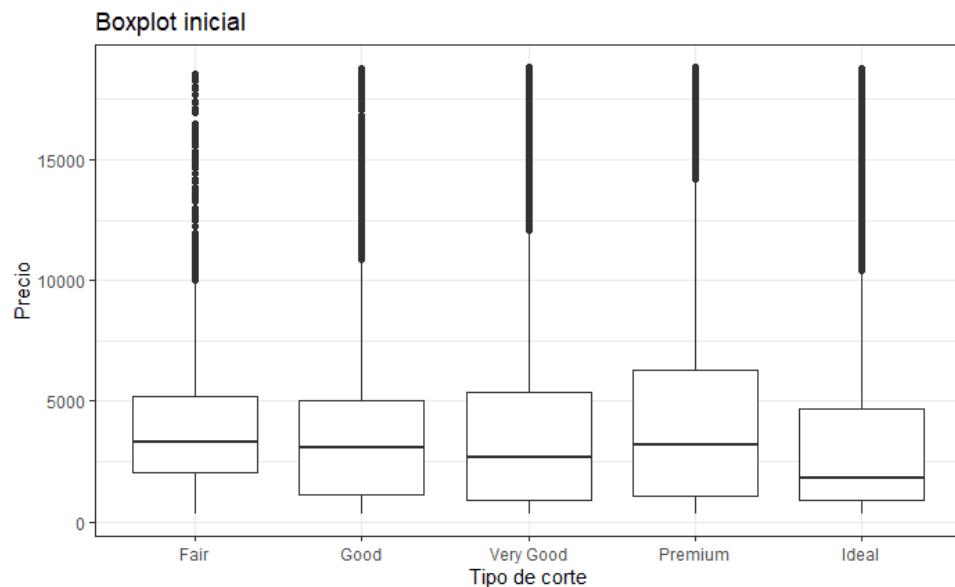
gg ← ggplot(diamonds, aes(x = cut, y = price)) +
 geom_boxplot() +
 labs(title = "Boxplot inicial",
 y = "Precio",
 x = "Tipo de corte")

gg
```

A la función principal debemos pasarle dos variables en el mapeo de `aes()` para conseguir un resultado satisfactorio:

- Una variable categórica, ‘cut’ en este caso.
- Una variable numérica, ‘price’ en este ejemplo.

Tras esto, invocaremos un boxplot utilizando el `geom_boxplot()`, que transformará nuestros datos en la visualización deseada.

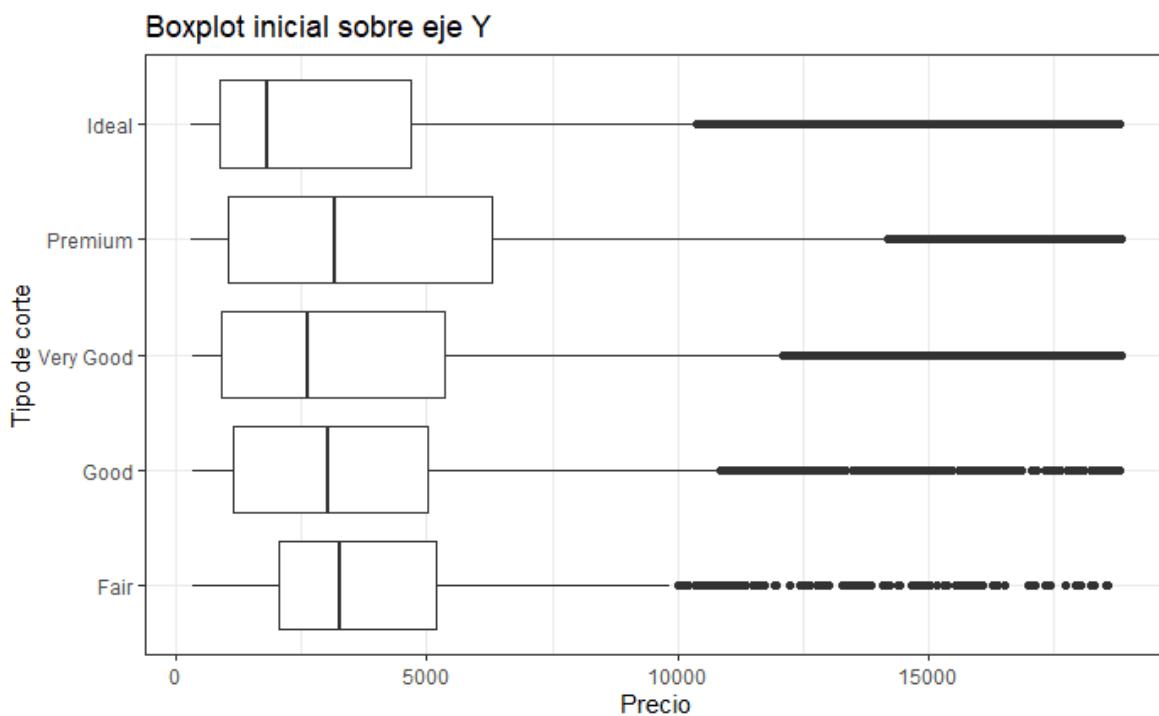


**Nota:** la orientación de las cajas se marca por la posición de la variable categórica seleccionada en los ejes. Lo habitual es ver dicha variable en el eje X, como en el ejemplo anterior. Si cambiamos de posición ambas variables, se rotará el gráfico, tenlo en cuenta.

```
Empezamos con la versión simple
gg <- ggplot(diamonds, aes(y = cut, x = price)) +
 geom_boxplot() +
 labs(title = "Boxplot inicial sobre eje Y",
 x = "Precio",
 y = "Tipo de corte")
gg
```

Elige siempre la que sea más apropiada para contar tu historia o la que te resulte más fácil de leer. Esto suele ser subjetivo.

Respecto a la imagen obtenida, podrás observar que, dependiendo del tipo de corte del diamante, la variabilidad en el precio (tamaño de la caja) es diferente. Salta a la vista también la gran cantidad de atípicos en todos los tipos de corte. Las rayas centrales que resaltan la mediana de cada corte nos dan, a su vez, información acerca de la distribución de los datos en cada caso.



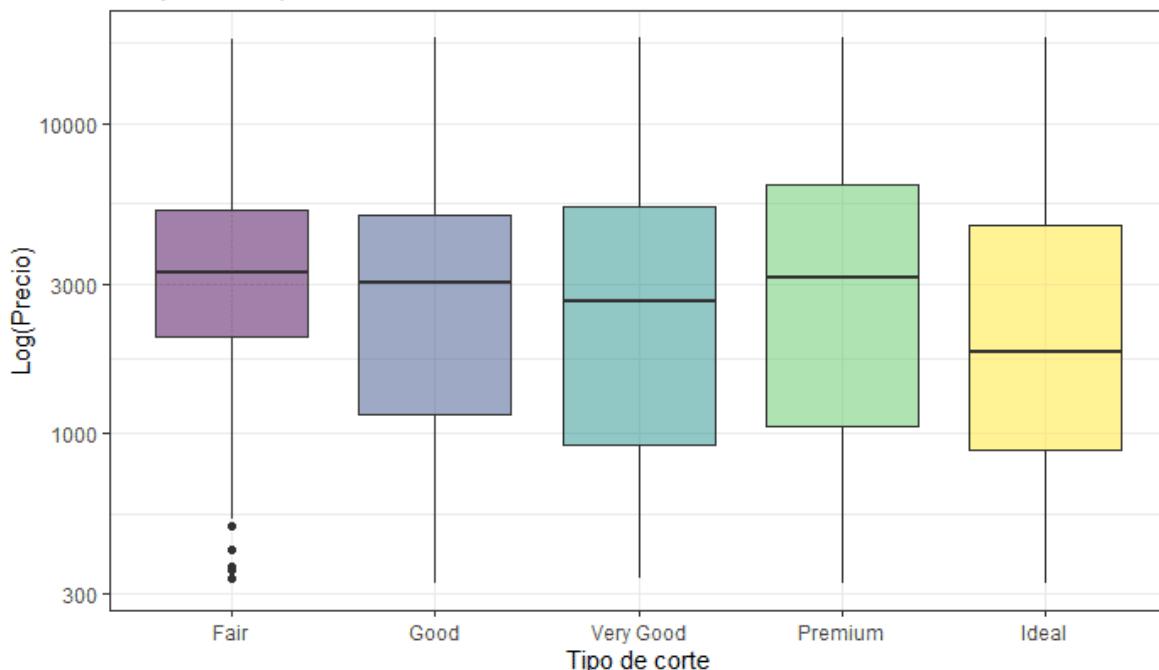
En este caso, podemos **embellecer y mejorar la visualización** aplicando algunos colores y utilizando una escala logarítmica en el precio para ‘achatar’ los extremos de la distribución de precios. Además, un cambio de escala facilita la lectura de los datos.

```
Enriquecemos nuestros resultados
gg ← ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
 geom_boxplot() +
 scale_y_log10() +
 scale_fill_viridis_d(alpha = 0.5) +
 # scale_fill_brewer(palette = "Dark2") +
 labs(title = "Boxplot enriquecido",
 y = "Log(Precio)",
 x = "Tipo de corte") +
 theme(legend.position = "none")

gg
```

En esta ocasión hemos sofisticado un poco más el código.

**Boxplot enriquecido**



Vayamos por partes:

- Hemos añadido el argumento *fill* dentro del `aes()` para colorear las cajas. En este caso, hemos usado la misma variable que se representa en el eje X para dotar de color las mismas.
- Hemos añadido la función `scale_y_log10()` que aplica una transformación en la variable que se representa sobre el eje Y, precio en este caso, y que permite aplanar la distribución de los datos para ‘limpiar’ y facilitar la visualización. Ahora los grupos se pueden comparar más fácilmente.
- Hemos utilizado la función `scale_fill_viridis_d()` para seleccionar la paleta de color a aplicar. Recuerda que también puedes utilizar la función `scale_fill_brewer()`, seleccionando la paleta deseada. Además, hemos usado el argumento `alpha` para añadir una ligera transparencia.
- Finalmente, **desactivamos la leyenda** de color pues no aporta nada nuevo respecto a la información mostrada en el eje X.

Ahora podemos comparar mejor los grupos, podemos observar que, los diamantes con corte ‘premium’ son los que, en general, poseen mayor precio de mercado, superando incluso a los de corte ‘ideal’.

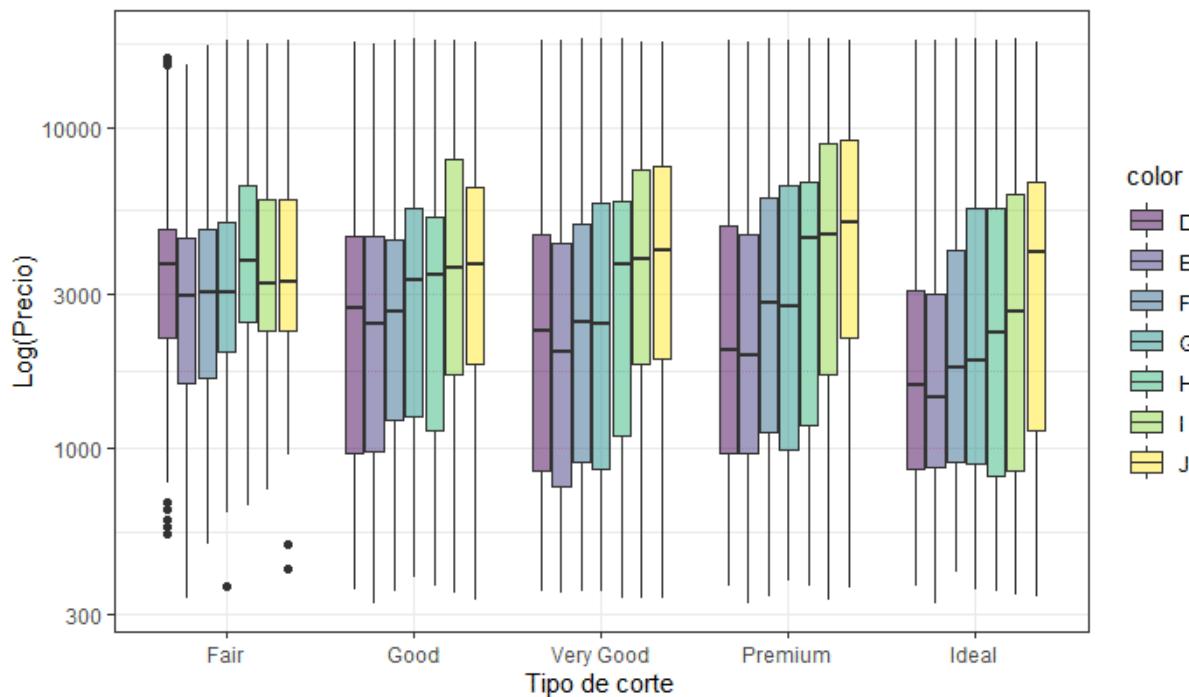
Como ocurría en el fastbook anterior, nuestras visualizaciones pueden ser dotadas de múltiples dimensiones, una de ellas es el color. En el caso anterior utilizábamos la misma variable que ya representábamos en el eje X, **¿pero qué ocurriría si la variable fuese otra diferente**, como por ejemplo, el color del diamante? Vamos a comprobarlo:

```
Añadimos nuevas dimensiones

gg ← ggplot(diamonds, aes(x = cut, y = price, fill = color)) +
 geom_boxplot() +
 scale_y_log10() +
 scale_fill_viridis_d(alpha = 0.5) +
 # scale_fill_brewer(palette = "Dark2") +
 labs(title = "Boxplot de Precios vs Tipo de corte teniendo en cuenta el Color",
 y = "Log(Precio)",
 x = "Tipo de corté")
 # theme(legend.position = "none")

gg
```

### Boxplot de Precios vs Tipo de corte teniendo en cuenta el Color



Pocos cambios en el código respecto al paso anterior: **solo hemos sustituido la variable mapeada en fill y desactivado la anulación de la leyenda.** Y fíjate: ahora sí que aporta la información necesaria.

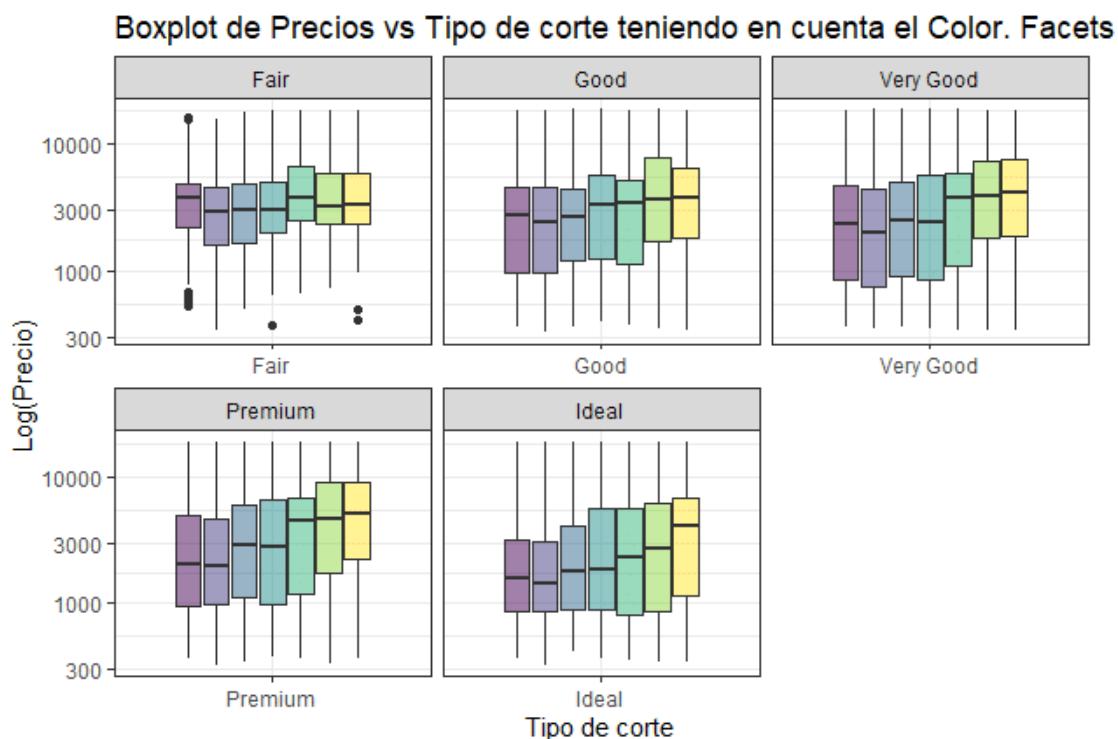
De hecho, con tan pocos cambios, hemos añadido una cantidad de información brutal. Ya puedes comparar la distribución de precio de cada diamante según su tipología de corte y color. El primer aprendizaje que obtenemos es que **conforme aumenta el grado de color, tiende a aumentar el precio de los diamantes** (esto lo podemos ver al comprobar que las cajas J de color amarillo tienden a estar más arriba en comparación con el resto; además, las medianas nos dan la pista para aseverar dicho aprendizaje).

Aun así, seguro que te empieza a costar leer el gráfico: demasiada información condensada en poco espacio. No te preocupes, está hecho adrede y nos da pie a introducir una función maravillosa en ggplot, los **facets**. Vamos a demostrarlo primero en código y resultado y comentamos *a posteriori*.

```
Descubriendo los facets

gg <- ggplot(diamonds, aes(x = cut, y = price, fill = color)) +
 geom_boxplot() +
 scale_y_log10() +
 scale_fill_viridis_d(alpha = 0.5) +
 facet_wrap(~ cut, scales = "free_x") +
 # scale_fill_brewer(palette = "Dark2") +
 labs(title = "Boxplot de Precios vs Tipo de corte teniendo en cuenta el Color. Facets",
 y = "Log(Precio)",
 x = "Tipo de corte")
theme(legend.position = "none")

gg
```



La función `facet_wrap()` permite 'romper' un gráfico de ggplot y aislarlo por la variable deseada.

---

Esto nos permite un **análisis más limpio**, en algunos casos, y una **comparación centrada dentro de cada categoría de nuestra variable de referencia**. Son una herramienta muy útil y que permite generar visualizaciones muy limpias.

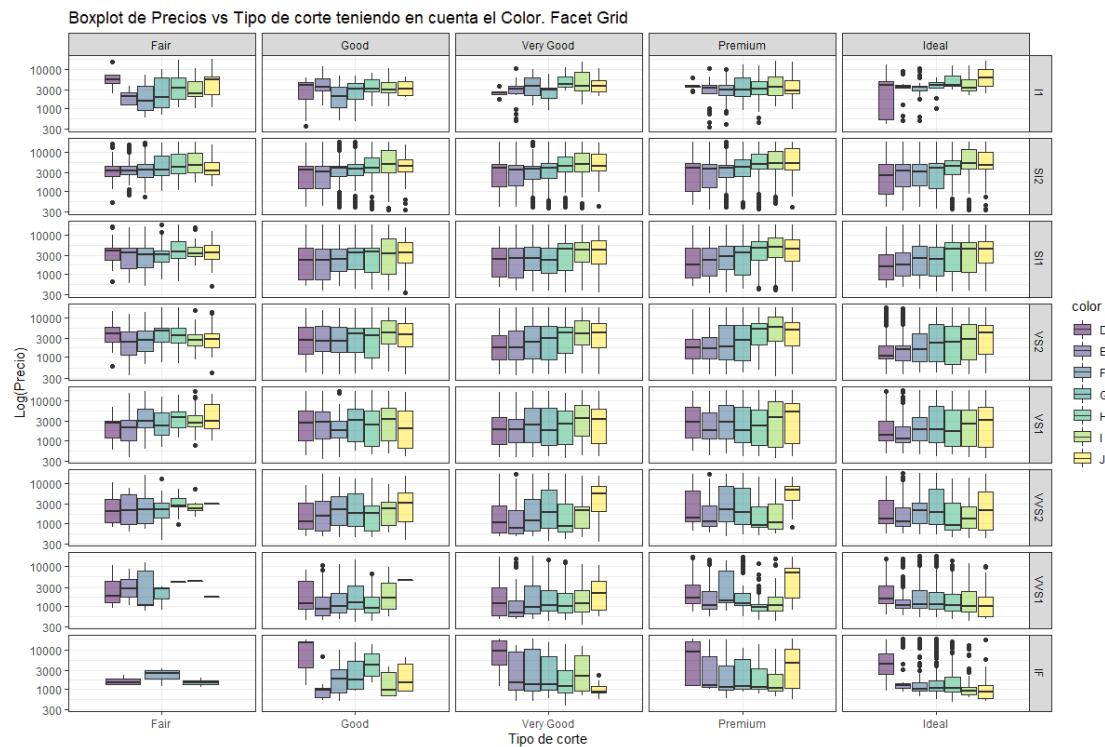
Es importante, en este caso, el añadido del argumento de `scales = "free_x"` para no repetir todos los valores de corte dentro de cada subcaja.

## Prueba a eliminar ese argumento y chequea los resultados por ti mismo... Horrible, ¿verdad?

Existe una variante de esta función que acepta hasta dos variables, lo que permite complicar la comparativa aún más. En este caso, por ejemplo, podríamos añadir la claridad para terminar de rizar el rizo.

```
Descubriendo los facets
gg <- ggplot(diamonds, aes(x = cut, y = price, fill = color)) +
 geom_boxplot() +
 scale_y_log10() +
 scale_fill_viridis_d(alpha = 0.5) +
 facet_grid(clarity ~ cut, scales = "free_x") +
 # scale_fill_brewer(palette = "Dark2") +
 labs(title = "Boxplot de Precios vs Tipo de corte teniendo en cuenta el Color. Facet Grid",
 y = "Log(Precio)",
 x = "Tipo de corte")
 # theme(legend.position = "none")

gg
```



Variando la función `a facet_grid()` y enfrentando las dos variables deseadas podemos conseguir visualizaciones que son prácticamente dashboards en sí. Esto tiene un **potencial tremendo y permite explorar grandes masas de información de golpe**. Aun así, la lectura inicial puede sorprender e intimidar precisamente por la gran cantidad de información mostrada, pero no deja de ser una malla por la que desplazarse libremente y efectuar las comparaciones deseadas.

## Variaciones o tipos de box plots

Vamos a detallar variaciones de este tipo de visualización que pueden resultar útiles en algunas circunstancias.

### El notch

---

El **notch** es una muesca alrededor de la **mediana de la caja que permite representar los intervalos de confianza al 95%** sobre dicho estadístico.

La utilidad de esto reside en determinar con fiabilidad cuando dos grupos provienen de poblaciones de datos diferentes, es decir, cuando sus distribuciones difieren.

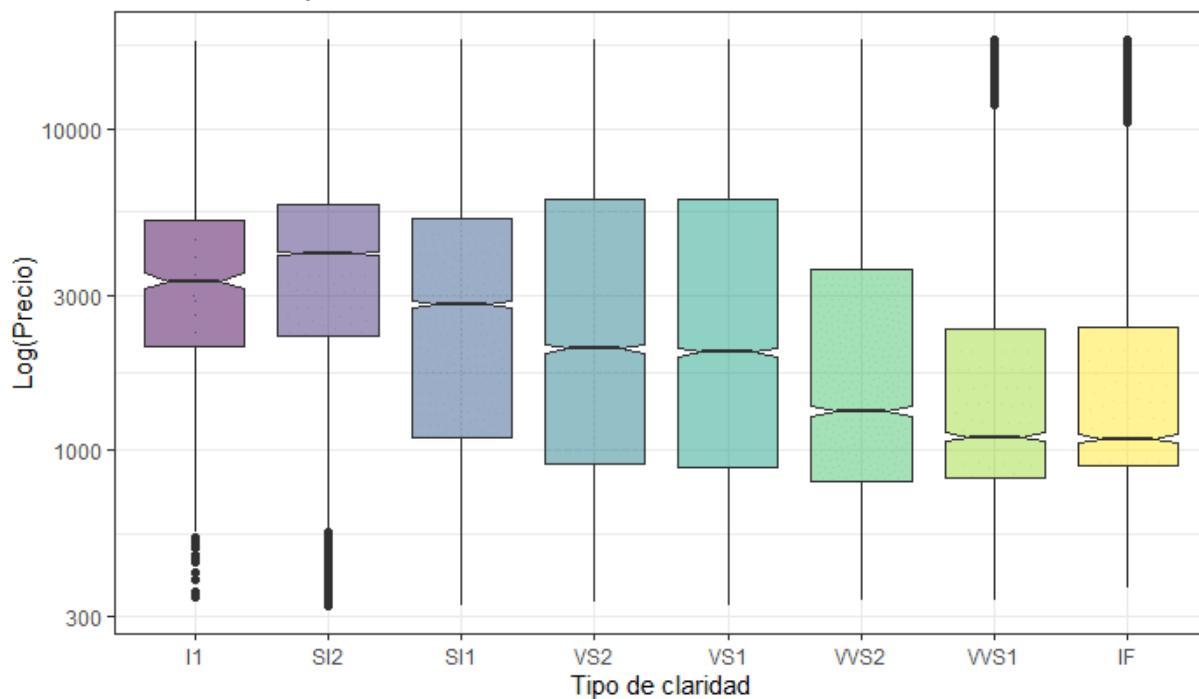
La forma de obtenerlo no puede ser más sencilla, solo requiere el uso de un nuevo argumento dentro del **geom**.

```
```{r}
## Variaciones del Boxplot. The Notch

gg ← ggplot(diamonds, aes(x = clarity, y = price, fill = clarity)) +
  geom_boxplot(notch = T) +
  scale_y_log10() +
  scale_fill_viridis_d(alpha = 0.5) +
  # scale_fill_brewer(palette = "Dark2") +
  labs(title = "Mostrando el potencial del Notch",
       y = "Log(Precio)",
       x = "Tipo de claridad") +
  theme(legend.position = "none")

gg
```

Mostrando el potencial del Notch



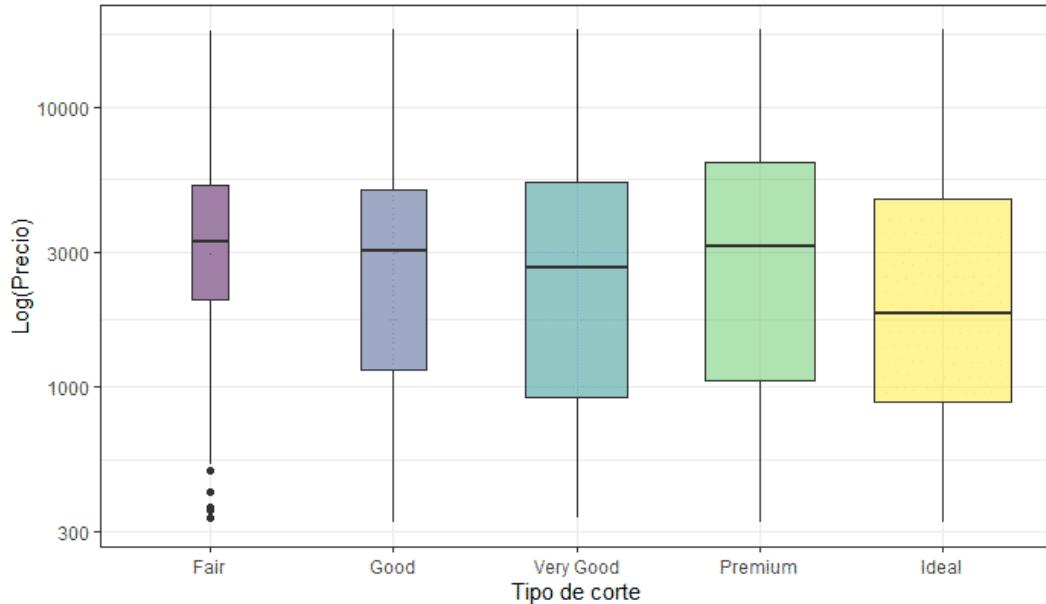
Si los intervalos de confianza de dos o más diagramas no se superponen significa que hay fuerte evidencia estadística de que las medianas son diferentes y, por tanto, la distribución de los datos.

En el ejemplo anterior, esto no se cumpliría para las claridades VS1 y VS2 que tienen una superposición perfecta. Sin embargo, sí podemos afirmar que existen diferencias entre VS1 y la claridad IF o SI2.

Anchura según observaciones

Otra variación que suele resultar muy útil es establecer la anchura de la caja en función del número de observaciones presente en dicho grupo o corte de la variable. ¿Qué significa esto? Pues en todas las visualizaciones realizadas hasta ahora, el tamaño de cada caja es el mismo, por lo tanto, podríamos estar extrayendo aprendizajes de grupos con muy pocas observaciones, con el peligro que conlleva esto. Para evitar estas problemáticas surge esta variación.

```
## Variaciones del Boxplot. Anchura según observaciones
gg <- ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
  geom_boxplot(varwidth = T) +
  scale_y_log10() +
  scale_fill_viridis_d(alpha = 0.5) +
  # scale_fill_brewer(palette = "Dark2") +
  labs(title = "El tamaño de la caja se fija según el número de observaciones que caigan dentro",
       y = "Log(Precio)",
       x = "Tipo de corte") +
  theme(legend.position = "none")
gg.
```



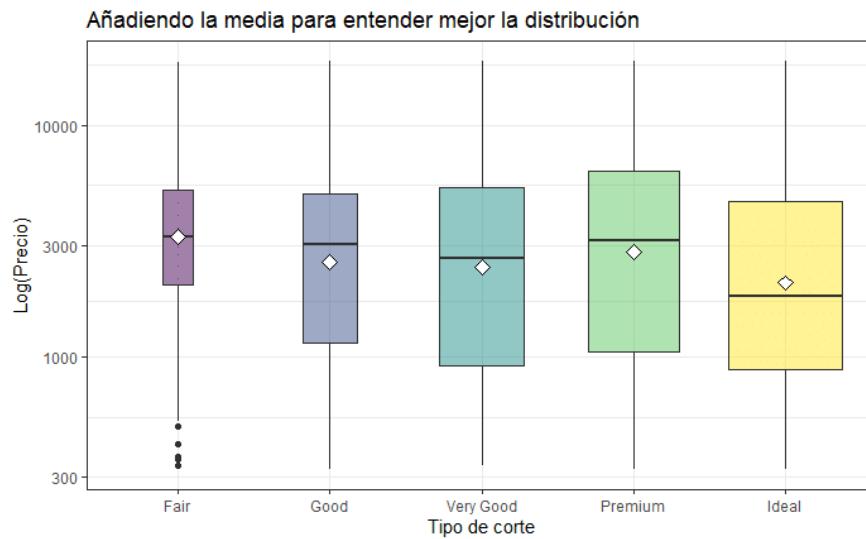
Esta visualización ya la conocemos, pero ahora tiene una capa extra de información. Es sencillo identificar visualmente que los diamantes con un corte ‘fair’ son los minoritarios, mientras que los de corte ‘ideal’ son los más numerosos. Si la caja fuese extremadamente delgada, nos alertaría de posibles problemas en dicha categoría y tendríamos más información a priori para tomar las decisiones.

Añadiendo la media a la ecuación

Como ya hemos detallado, la raya central de la caja de un box plot permite conocer el valor de la mediana de la distribución de los datos. Ahora bien, añadiendo la media a esa foto, podemos extraer conclusiones acerca del tipo de distribución de los datos y de la asimetría de estos.

Vamos a verlo.

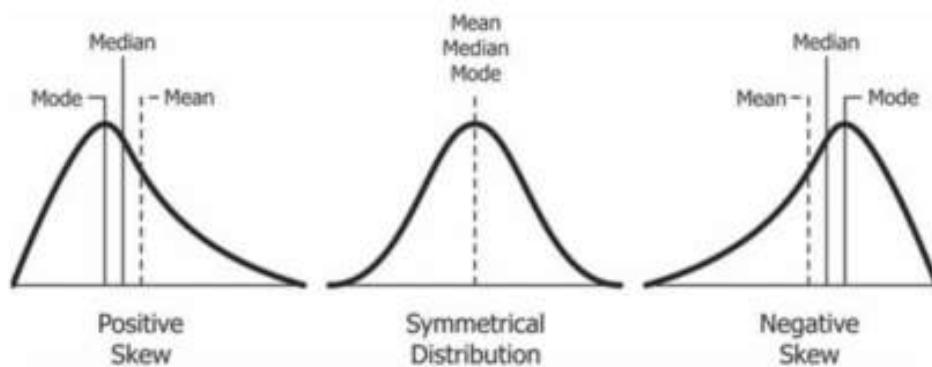
```
## Variaciones del Boxplot. Añadiendo la media al boxplot
gg ← ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
  geom_boxplot(varwidth = T) +
  scale_y_log10() +
  scale_fill_viridis_d(alpha = 0.5) +
  # scale_fill_brewer(palette = "Dark2") +
  stat_summary(fun = "mean", geom = "point", shape = 23, size = 3, fill = "white") +
  labs(title = "Añadiendo la media para entender mejor la distribución",
       y = "Log(Precio)",
       x = "Tipo de corte") +
  theme(legend.position = "none")
gg
```



Con la función `stat_summary()` podemos especificar estadísticos básicos sobre las visualizaciones. En este caso, hemos añadido la media como un punto blanco con un tamaño determinado.

La sintaxis del código es sencilla, prueba a cambiar algunos argumentos como el tamaño o la forma del punto.

Según la posición de la media respecto a la mediana podemos entender cómo es la asimetría de la distribución. Por ejemplo: en este caso, los diamantes con corte fair tienen una distribución simétrica al coincidir ambos estadísticos; ‘good’, ‘very good’ y ‘premium’ poseen una asimetría negativa o a la izquierda, al estar la media por debajo del valor de la mediana; y finalmente, el corte ‘ideal’ es el único caso con asimetría positiva o a la derecha. Te dejo esta imagen para refrescar los conocimientos.



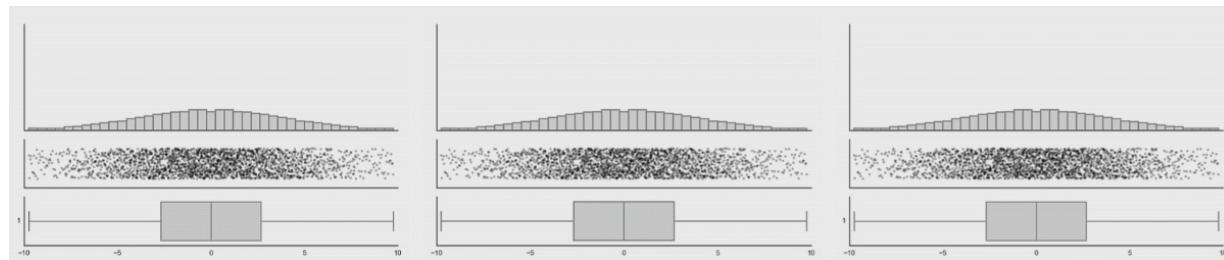
Fuente: [Wikipedia](#).

Jitter y los violin plots

Como hemos visto en los apartados anteriores, los **diagramas de cajas** son herramientas que nos permiten visualizar los datos de una forma rápida y nos aportan muchísima información. Podemos resumir sus ventajas en los siguientes puntos:

- Resume la distribución de grandes masas de información con una gran simpleza visual.
- Detecta y muestra la presencia de datos atípicos.
- Permite comparar la distribución de múltiples grupos.
- Permite conocer la asimetría de la distribución, además de su curtosis.
- Fáciles de pintar y de interpretar (una vez que conoces todo su potencial).
- Los bigotes son graciosos.

Con esa lista parece que esta visualización no tiene puntos débiles, desgraciadamente, esto no es así. Vamos a estudiar la problemática principal de este recurso. Echa un vistazo a la siguiente imagen.



Fuente: [Autodesk](#).

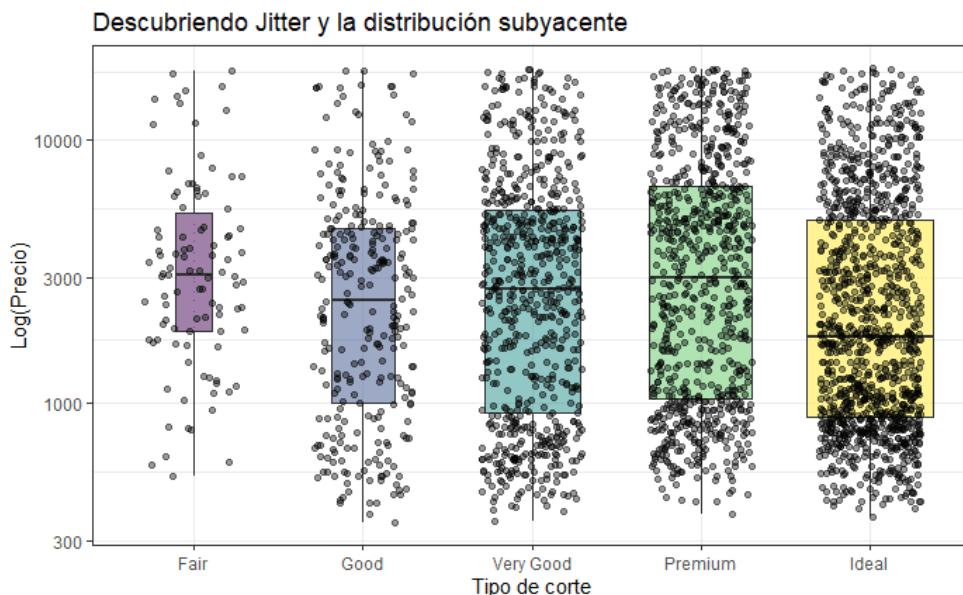
Un boxplot puede resumir bien la información, pero a su vez, puede esconder clusters de puntos y múltiples modas detrás de esa representación de caja que permanecerán invisibles a nuestros análisis.

Dedica unos minutos a observar de nuevo el GIF y cómo varía la distribución subyacente de cada visualización mientras esta permanece impasible. Estamos de acuerdo en que eso es un auténtico problema, ¿no? Algo similar al número de observaciones que veíamos hace un rato...

A lo largo de los años, han surgido variaciones que vienen a reparar y fortalecer estos puntos débiles de tan maravillosa visualización. El primer ‘ parche’ fue pintar los puntos que representan los diagramas de caja superpuestos.

Para poder visualizarlos de manera correcta, se añadía una especie de ‘ruido’ a cada punto, una perturbación que deshiciese la superposición de los puntos. Esto es lo que se denomina **jitter** y permite analizar la estructura subyacente de los datos que conforman el diagrama.

```
## Variaciones del Boxplot. Reforzando la debilidad
gg <- ggplot(diamonds %>% sample_n(3000), aes(x = cut, y = price, fill = cut)) +
  geom_boxplot(varwidth = T) +
  geom_jitter(width = 0.3, alpha = 0.4) +
  scale_y_log10() +
  scale_fill_viridis_d(alpha = 0.5) +
  # scale_fill_brewer(palette = "Dark2") +
  labs(title = "Descubriendo Jitter y la distribución subyacente",
       y = "Log(Precio)",
       x = "Tipo de corte") +
  theme(legend.position = "none")
gg.
```

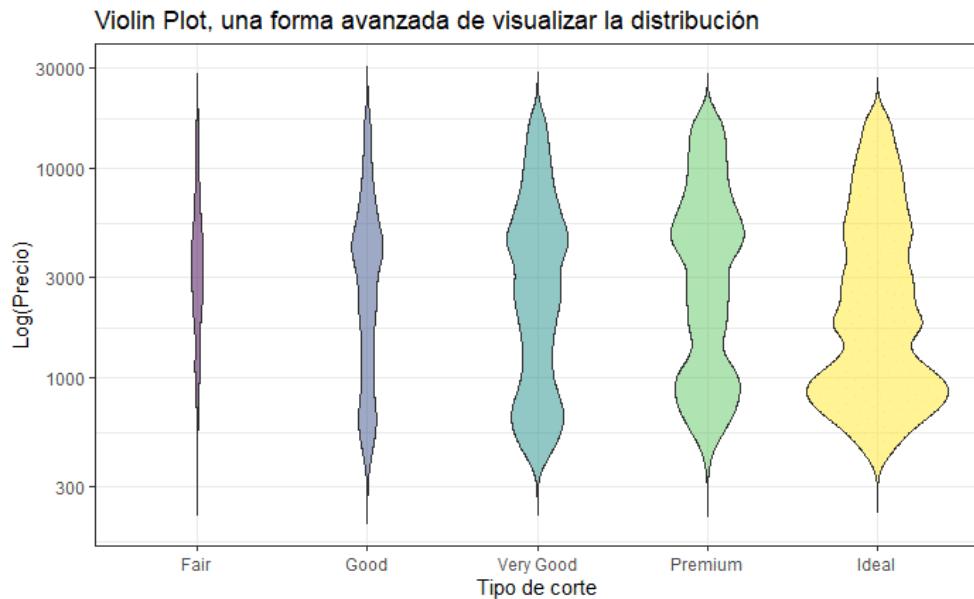


Solo necesitas utilizar la función *geom_jitter()* que tiene como argumento principal *width* (o *height*) donde se especifica la cantidad de ruido horizontal (o vertical) que se introduce en ambas direcciones. Además, en presencia de un dataset con muchísimos datos, es conveniente añadir un toque de transparencia con *alpha* para mejorar la visualización. Por último, para este punto, hemos seleccionado una **muestra** de los 54 mil diamantes.

Ten en cuenta este detalle, pues plotear 54 mil puntos en una sola visualización puede congelar tu ordenador y el resultado será una serie de columnas negras que no servirán de nada.

Finalmente, una alternativa surgida a los diagramas de caja y bigotes son los **gráficos de violín**, una opción reciente que **combina la funcionalidad del box plot y la estimación de la densidad de la distribución original**. Gracias a estas propiedades, nos permite un conocimiento más profundo de la distribución; luego, **son idóneos cuando se necesita dibujar una gran cantidad de información**. Vamos a analizarlo.

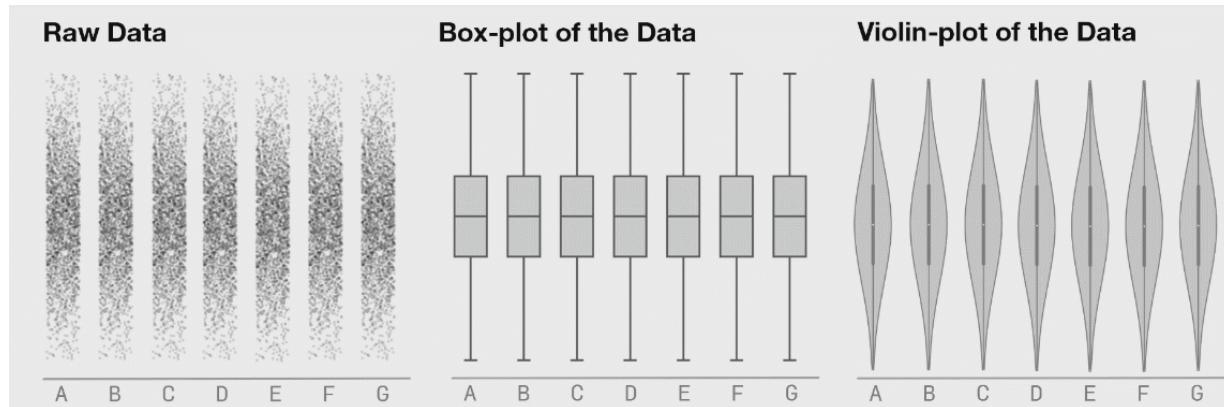
```
## Variaciones del Boxplot. Descubriendo un nuevo enfoque
gg <- ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
  geom_violin(trim = FALSE, scale = "count") +
  scale_y_log10() +
  scale_fill_viridis_d(alpha = 0.5) +
  # scale_fill_brewer(palette = "Dark2") +
  labs(title = "Violin Plot, una forma avanzada de visualizar la distribución",
       y = "Log(Precio)",
       x = "Tipo de corte") +
  theme(legend.position = "none")
gg
```



Hemos añadido la función *geom_violin()* para generar la visualización.

- El argumento *trim* evita que recorte de forma abrupta el final de la distribución superior e inferior de cada gráfico (cuidado con esta opción, porque puede dar la impresión de que existen datos más allá de ciertos límites).
- El argumento *scale* permite modificar la forma de cada violín de acuerdo con un criterio, en este caso, con el número de observaciones dentro de cada subgrupo, tal y como veíamos antes.

Esto nos facilita la tarea de **comparar las formas y posibles concentraciones de datos en cada grupo**. Comparto otro ejemplo donde visualmente podemos ver el potencial de este gráfico sobre el enfoque tradicional de boxplot:

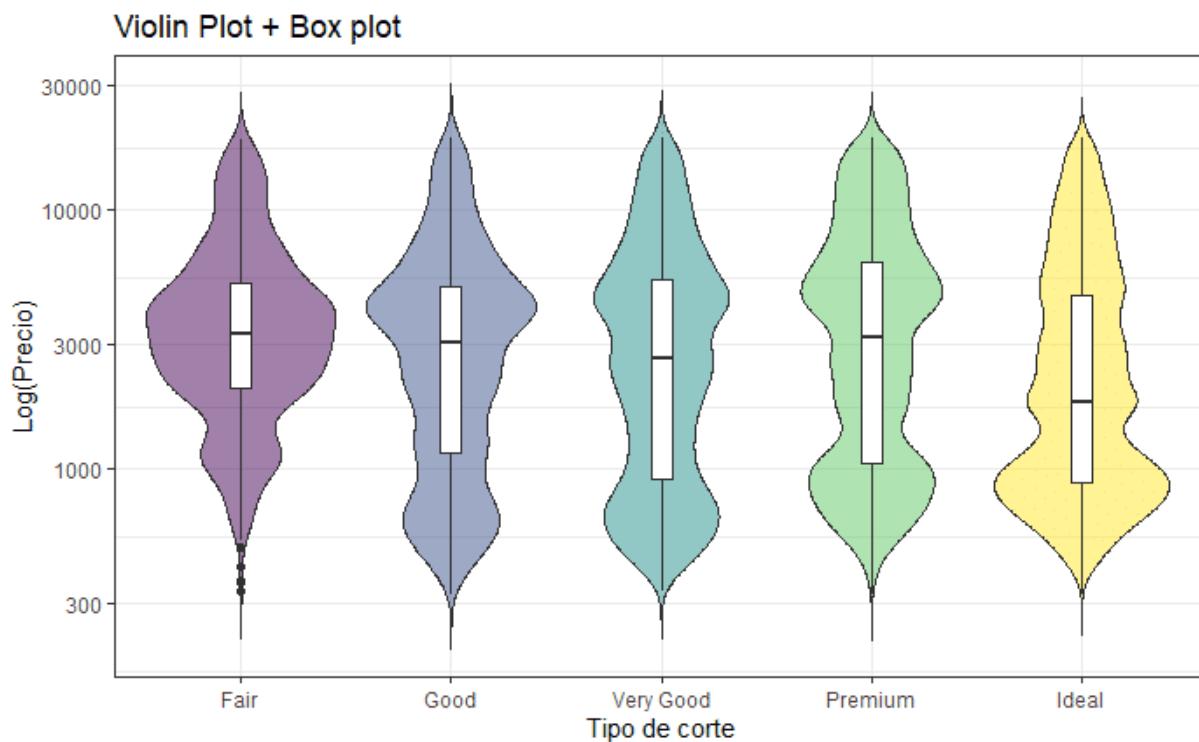


Fuente: [Autodesk](#).

En este GIF podemos comparar la utilidad de este tipo de visualizaciones, la gama de detalles que ofrecen para presentar la distribución de un dataset está muy por encima del enfoque tradicional del diagrama de cajas y bigotes. Ahora bien, **esto no quiere decir que siempre debas utilizar este gráfico sobre un box plot**. Si la distribución de los datos no esconde nada y no merece la pena resaltarla, la simpleza de un diagrama de cajas será siempre superior.

Podemos mejorar la visualización obtenida combinando ambos mundos en una sola imagen.

```
## Variaciones del Boxplot. Combinando los dos mundos
gg <- ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
  geom_violin(trim = FALSE) +
  geom_boxplot(width=0.1, fill = "white") +
  scale_y_log10() +
  scale_fill_viridis_d(alpha = 0.5) +
  # scale_fill_brewer(palette = "Dark2") +
  labs(title = "Violin Plot + Box plot",
       y = "Log(Precio)",
       x = "Tipo de corte") +
  theme(legend.position = "none")
gg
```



Si encadenamos los violines con la función posterior de `geom_boxplot()`, podemos obtener una combinación superpuesta que nos facilita la información sobre la mediana y cuartiles que nos faltaba. También podríamos añadir los puntos subyacentes mediante jitter, pero eso lo dejo a tu elección.

Conclusiones

X Edix Educación

En el quinto fastbook hemos aprendido los entresijos de los **diagramas de cajas y bigotes**, mejor conocidos como box plots. Un gráfico cuyo potencial reside en su simpleza. Son gráficos que nos permiten entender la distribución de los datos y nos alertan de posibles outliers. Además, hemos descubierto que las pocas flaquezas que poseen son cubiertas con nuevas y avanzadas visualizaciones como los gráficos violín.

Hemos introducido otro concepto adicional como los **facets** y la utilidad que tienen para generar múltiples visualizaciones manteniendo la legibilidad y limpieza. Práctica con ellos porque las oportunidades que ofrecen son inmensas.

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 conclusiones.

Lección 4 de 4

Bibliografía

X Edix Educación

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

¡Enhorabuena! Fastbook superado

edix

Creamos Digital Workers