

Fundamentos de la programación estadística y Data Mining en R

Unidad 1. - Introducción (brevísima) a ggplot2

Dr. Germán Rosati (Digital House - UNSAM - UNTREF)

04 marzo, 2018

¿Qué es ggplot?

- Es un paquete de R que permite crear gráficos bastante más bonitos que los que se pueden generar con la librería base de R.
- Se basa en la teoría de la “Gramática de Gráficos” -Grammar of Graphics -de Wilkinson, Leland (2005)-
- Provee un enfoque unificado y estadísticamente orientado para la generación de visualizaciones
- Logra esto gracias al uso de objetos más “abstractos” que los objetos gráficos “elementales” de R.

Usando (rápidamente) ggplot2

- Al menos tres formas
 - ▶ `qplot`
 - ▶ `ggplot` + `geom_xxx`
 - ▶ `ggplot` + `layer`
- Nos enfocamos en la segunda.

Ejemplo rápido

- Usamos el dataset diamonds
- Subseteamos una parte del dataset

```
library(ggplot2)
data(diamonds)
set.seed(42)
small<-diamonds[sample(nrow(diamonds),1000),]
head(small)
## # A tibble: 6 x 10
##   carat cut          color clarity depth table price      x
##   <dbl> <ord>        <ord> <ord>    <dbl> <dbl> <int> <dbl> <dbl>
## 1 0.710 Very Good  H      SI1     62.5  60.0  2096  5.68  5.
## 2 0.790 Premium   H      SI1     61.8  59.0  2275  5.97  5.
## 3 1.03  Ideal     F      SI1     62.4  57.0  6178  6.48  6.
## 4 0.500 Ideal     E      VS2     62.2  54.0  1624  5.08  5.
## 5 0.270 Ideal     E      VS1     61.6  56.0   470  4.14  4.
## 6 0.300 Premium   E      VS2     61.7  58.0   658  4.32  4.
```

Ejemplo rápido

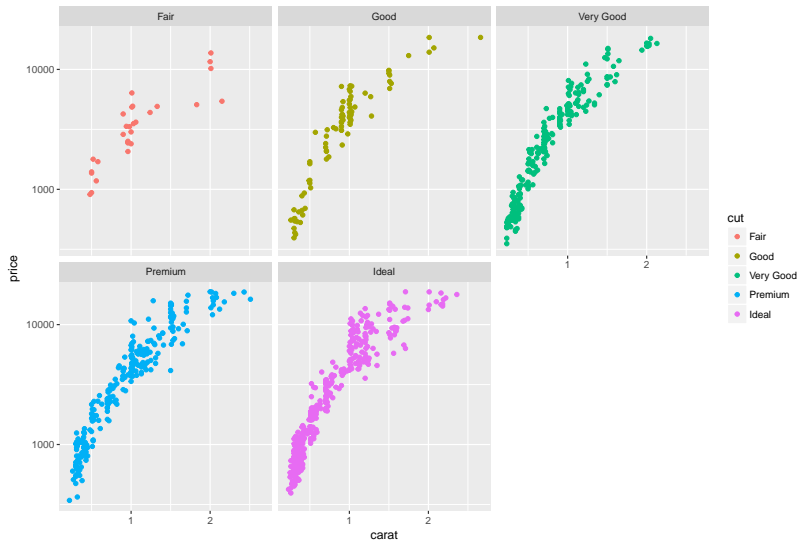
- Veamos un ejemplo para la sintaxis de ggplot2:

```
ggplot(small) +  
  geom_point(aes(x=carat,y=price,colour=cut)) +  
  scale_y_log10() +  
  facet_wrap(~cut) +  
  ggtitle("First example")
```

- La instrucción anterior nos devuelve...

Ejemplo rápido

First example



Ejemplo rápido: descomponiendo el código

- 1 Creamos el objeto ggplot y lo “llenamos” con los datos (siempre deberá ser un dataframe)

```
ggplot(small) +
```

- 2 Agregamos una capa estética (en este caso, de puntos)

```
geom_point(aes(x=carat,y=price,colour=cut)) +
```

- 3 “Facets”: se usa para condicionar sobre variables

```
facet_wrap(~cut) +
```

- 4 “Scales”: se usa para definir escalas de las dimensiones, paletas de color, etc.

```
scale_y_log10()
```

Mapeo de estética

- Todos los `geom_xxx()` requieren alguna capa referida a estética. Cada variable se “mapea” con alguna estética.
- Para ello se usa la función `aes`: `geom_point(aes(x = var))`
- Se pueden usar las siguientes:
 - ▶ `x`: x position (required)
 - ▶ `y`: y position (required)
 - ▶ `shape`: shape of point
 - ▶ `colour`: border colour
 - ▶ `size`: size
 - ▶ `fill`: internal colour
 - ▶ `alpha`: transparency

Mapeo de estéticas

- Por eso usamos

```
ggplot(small) +  
  geom_point(aes(x=carat,y=price,colour=cut))
```

- También serían posibles otras formas:

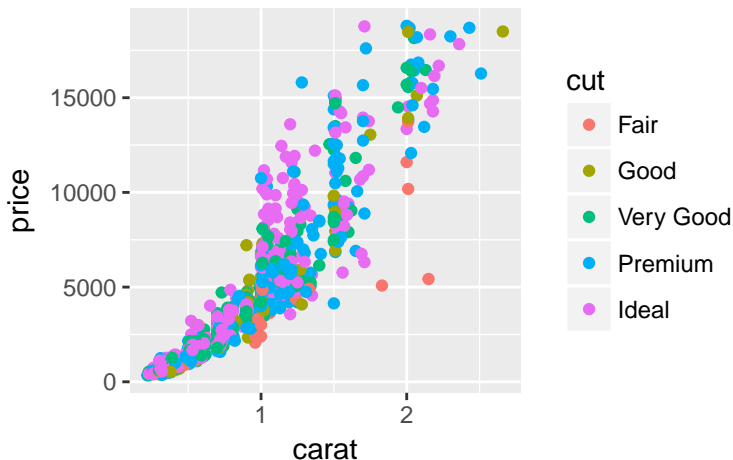
```
ggplot(small,aes(x=carat,y=price,colour=cut)) +  
  geom_point()
```

```
ggplot(small,aes(x=carat,y=price)) +  
  geom_point(aes(colour=cut))
```

Mapecto de estéticas: una sutileza

- Hay una diferencia entre setear y asignar estéticas.
- La asignación (o “mapping”) se hace a través de aes

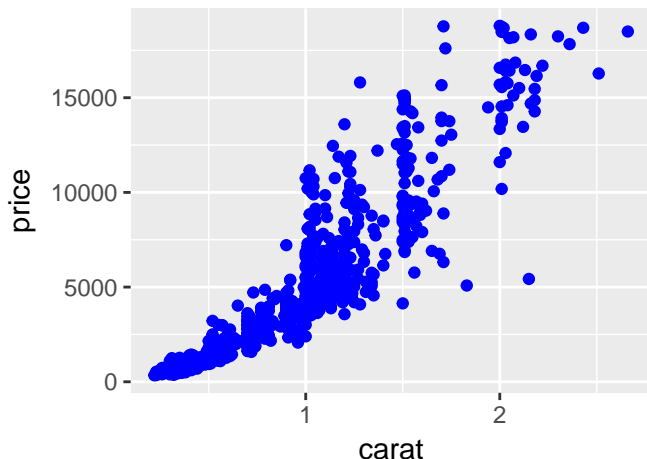
```
p + geom_point(aes(x=carat,y=price,colour=cut))
```



Mapecto de estéticas: una sutileza

- Pero el “setting” es decir, el hecho de fijar un valor a un determinado parámetro fijo se hace *fuera* de aes

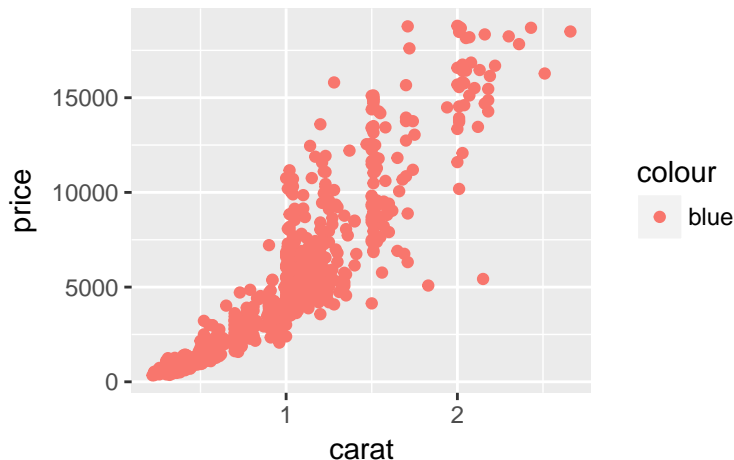
```
p + geom_point(aes(x=carat,y=price), colour="blue")
```



Mapecto de estéticas: una sutileza

- Tratar de fijar una estética dentro de aes ambas formas puede traer resultados inesperados:

```
p + geom_point(aes(x=carat,y=price,colour="blue"))
```

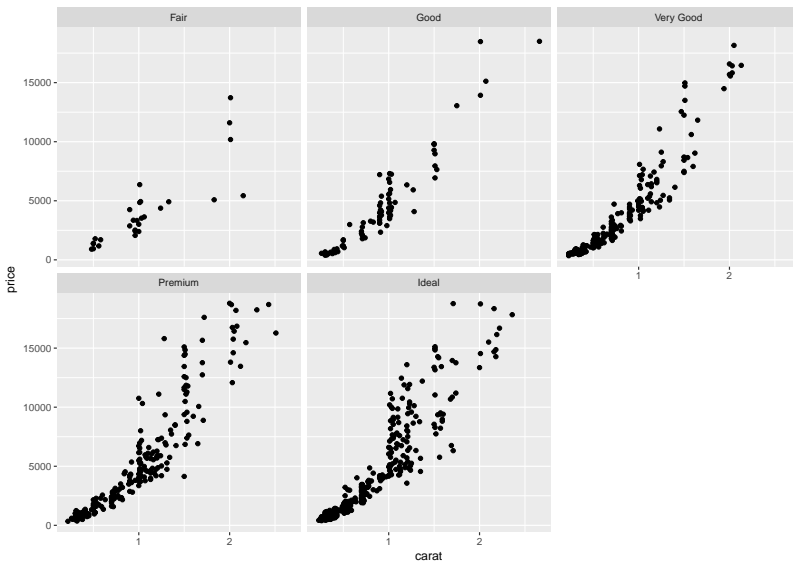


Faceting: condicionando sobre variables

- `facet_xxx` se usa para condicionar sobre 1 o 2 variables. Es decir, para producir un gráfico diferente para cada valor de la variable en `facet_xxx`. Veamos sobre la variable `cut`:

```
p + geom_point(aes(x=carat,y=price)) +  
  facet_wrap(~cut)
```

Faceting: condicionando sobre variables

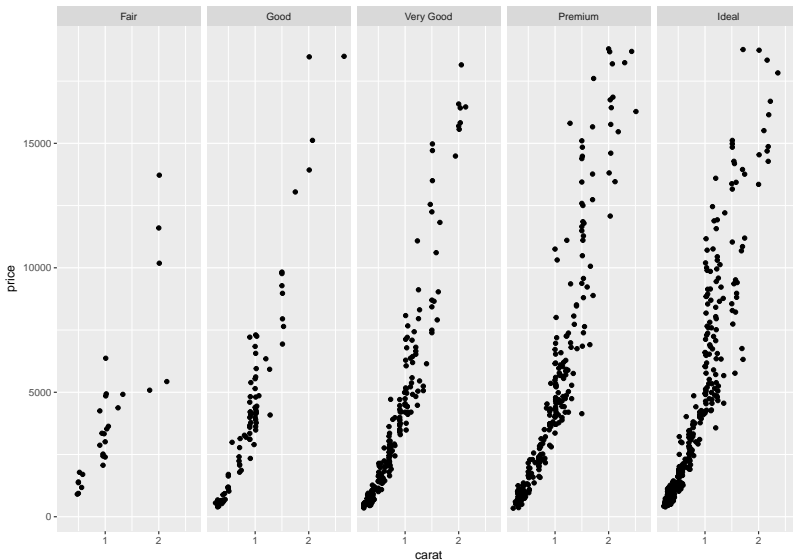


Faceting: condicionando sobre variables

- Con el argumento `nrow=1` generamos todos los gráficos sobre una sola fila.

```
p + geom_point(aes(x=carat,y=price)) +  
  facet_wrap(~cut, nrow=1)
```

Faceting: condicionando sobre variables

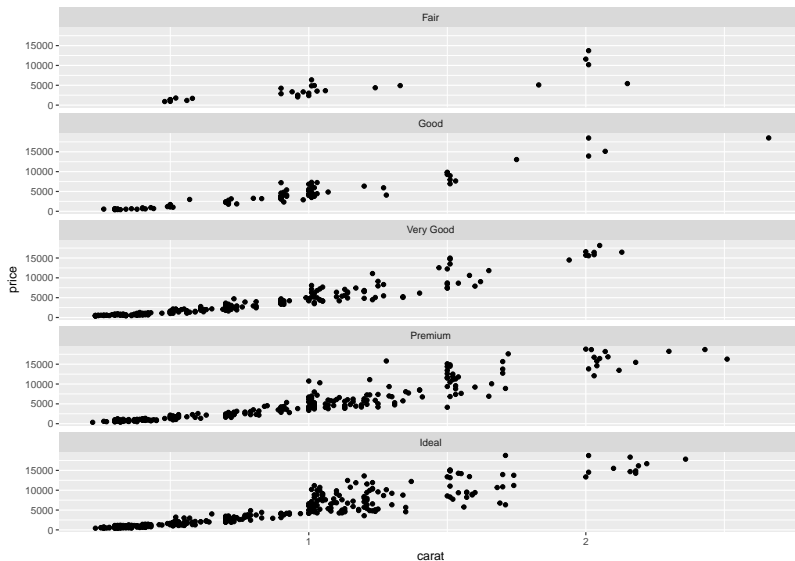


Faceting: condicionando sobre variables

- Con el argumento `ncol=1` generamos todos los gráficos sobre una sola columna.

```
p + geom_point(aes(x=carat, y=price)) +  
  facet_wrap(~cut, ncol=1)
```

Faceting: condicionando sobre variables

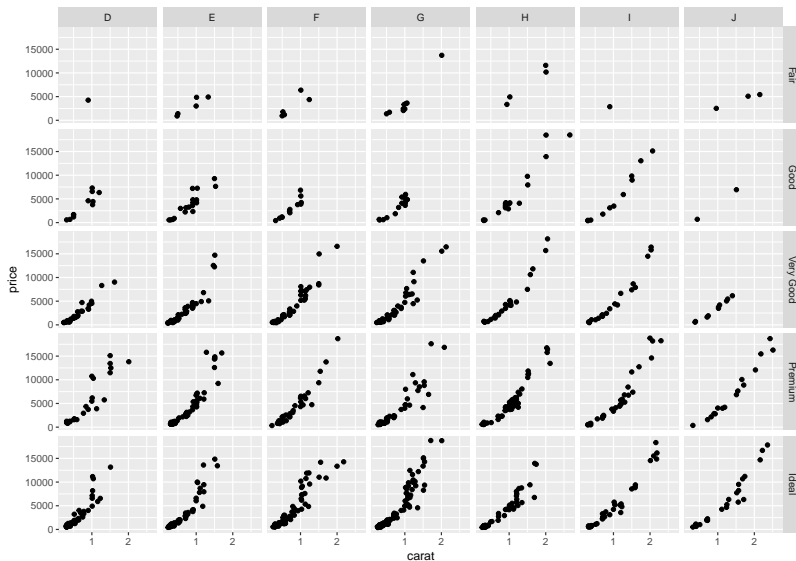


Faceting: condicionando sobre variables

- Usando la estética `facet_grid(cut~color)` generamos una grilla con los valores de `cut` y `color` y ploteamos un gráfico scatter para cada elemento de la grilla.

```
p + geom_point(aes(x=carat,y=price)) +  
  facet_grid(cut~color)
```

Faceting: condicionando sobre variables



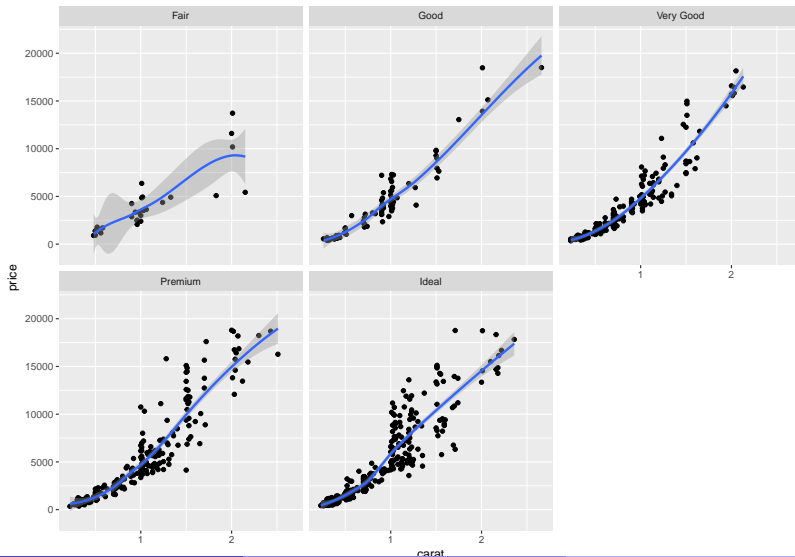
Otras geometrías: geom_smooth

- geom_smooth se usa para desplegar alguna línea de tendencia en los datos.
- Por defecto, usa un método llamado splines.

```
p <- ggplot(small, aes(x=carat, y=price))  
p + geom_point() +  
  geom_smooth() +  
  facet_wrap(~cut)
```

Otras geometrías: geom_smooth

```
## `geom_smooth()` using method = 'loess'
```

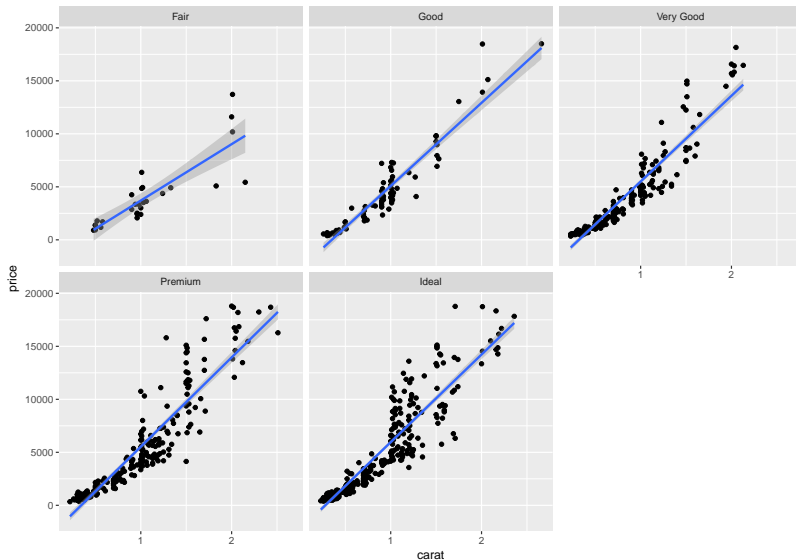


Otras geometrías: geom_smooth

- También se puede especificar alguna función para usar: en este caso una regresión lineal con `method='lm'`

```
p + geom_point() +  
  geom_smooth(method="lm") +  
  facet_wrap(~cut)
```

Otras geometrías: geom_smooth

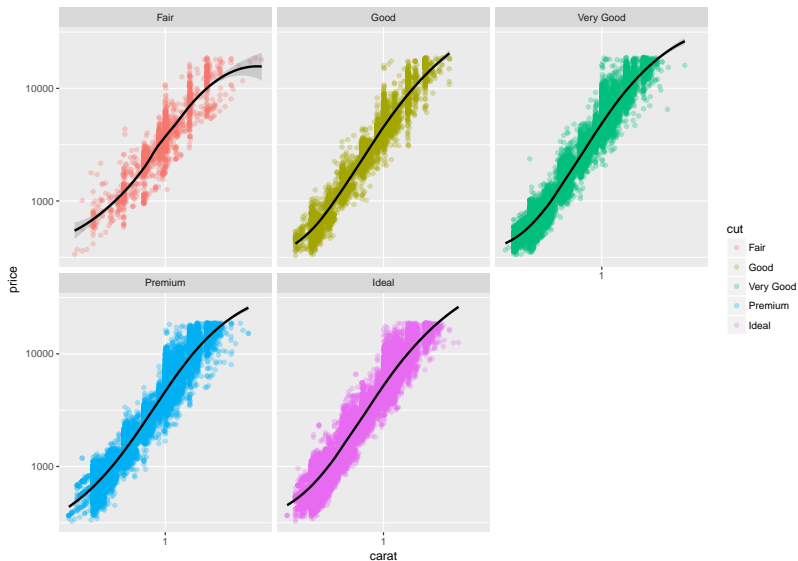


Otras geometrías: geom_smooth

- Pasemos en limpio...

```
p <- ggplot(diamonds, aes(x=carat, y=price, colour=cut))
p <- p + scale_x_log10() + scale_y_log10()
p <- p + geom_point(alpha=0.3) +
  geom_smooth(method='loess', colour='black')
p <- p + facet_wrap(~cut)
print(p)
```

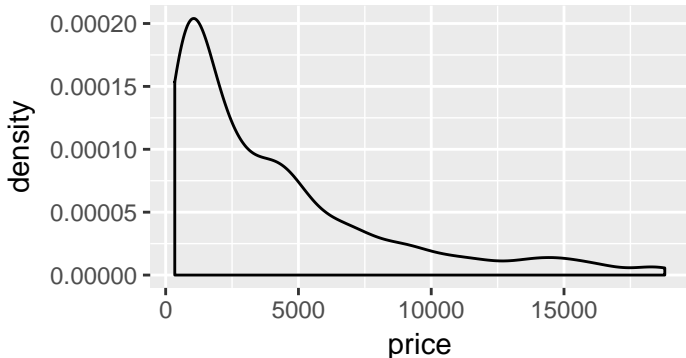
Otras geometrías: geom_smooth



Otras geometrías: density plots

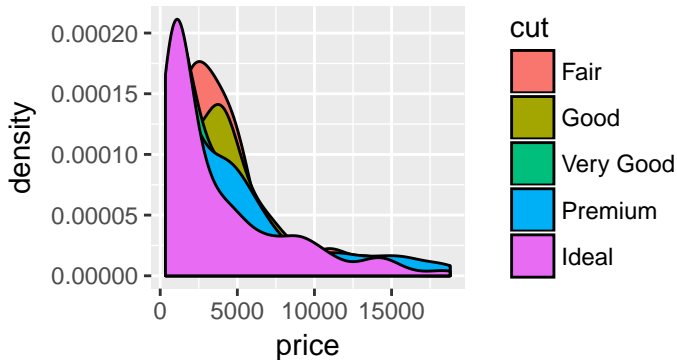
- En muchos casos los gráficos de densidad suelen ser más claros que los histogramas para analizar la distribución de variables cuantitativas.

```
ggplot(small) +  
  geom_density(aes(x=price))
```



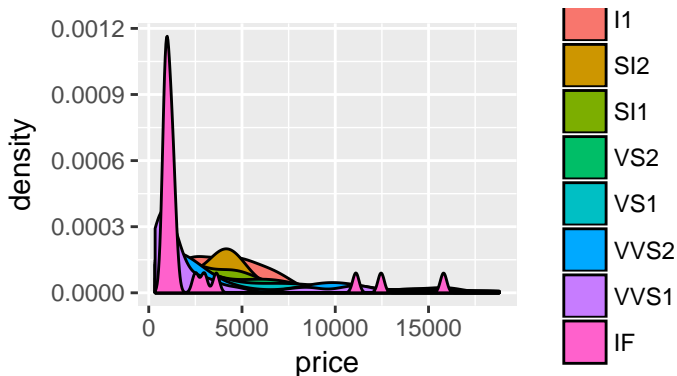
Otras geometrías: density plots

```
ggplot(small) +  
  geom_density(aes(x=price, fill=cut))
```



Otras geometrías: density plots

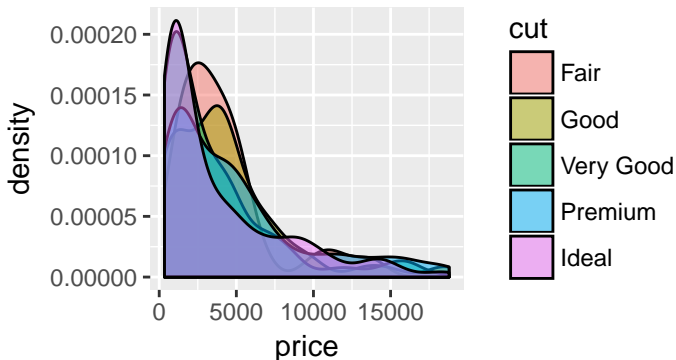
```
ggplot(small) +  
  geom_density(aes(x=price, fill=clarity))
```



Otras geometrías: density plots

- Podemos usar alguna transparencia para distinguir entre las distribuciones

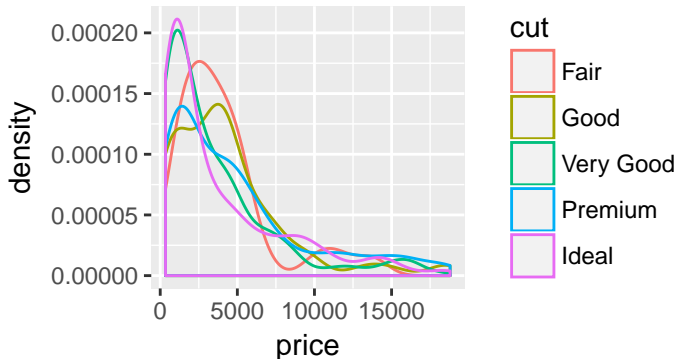
```
ggplot(small) +  
  geom_density(aes(x=price, fill=cut), alpha=0.5)
```



Otras geometrías: density plots

- O usar colour en lugar de fill

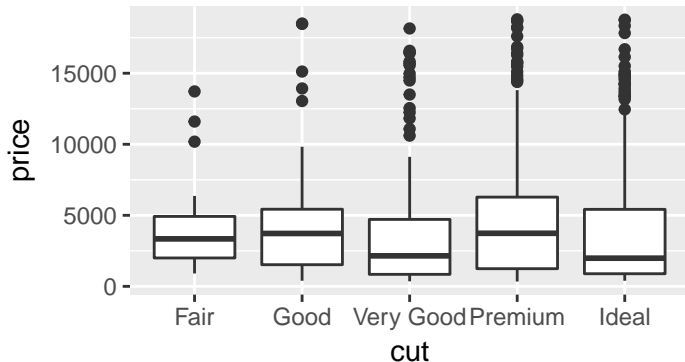
```
ggplot(small) +  
  geom_density(aes(x=price, colour=cut))
```



Otras geometrías: box plots

- Se puede usar `geom_boxplot()`

```
ggplot(small)+  
  geom_boxplot(aes(x=cut, y=price))
```

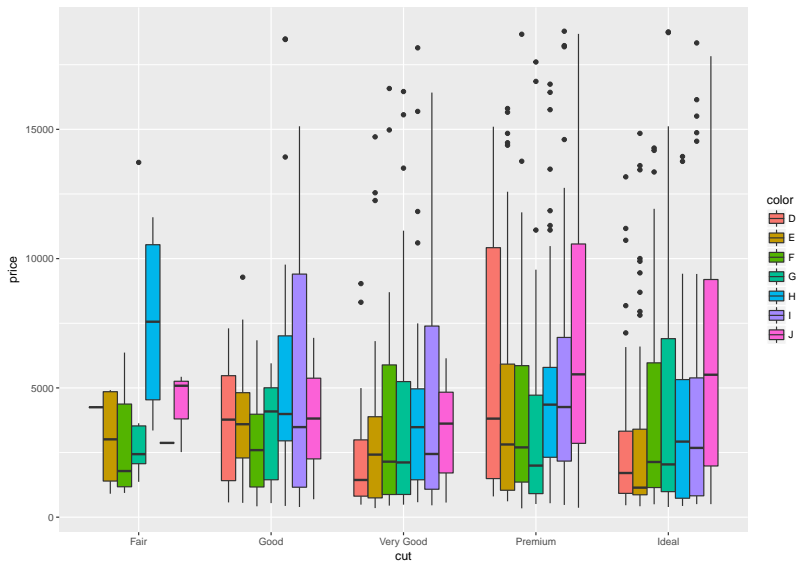


Otras geometrías: box plots

- Al igual que en el módulo base de gráficos de R, podemos confeccionar un boxplot de una variable cuantitativa para diferentes valores de otra variable, simplemente definiendo los parámetros `x=[var1]`, `y=[var2]`.

```
ggplot(small) +  
  geom_boxplot(aes(x=cut, y=price, fill=color))
```

Otras geometrías: box plots



Otras geometrías

- Hay muchísimas otras geometrías para explorar en `ggplot2`
 - ▶ `geom_abline`, `geom_jitter`, `geom_area`, `geom_line`
 - ▶ `geom_bar`, `geom_linerange`, `geom_bin2d`, `geom_path`
 - ▶ `geom_blank`, `geom_point`, `geom_boxplot`, `geom_pointrange`
 - ▶ `geom_contour`, `geom_polygon`, `geom_crossbar`, `geom_quantile` + `geom_density`, `geom_rect`, `geom_density2d`, `geom_ribbon`
 - ▶ `geom_errorbar`, `geom_rug`, `geom_errorbarh`, `geom_segment`
 - ▶ `geom_freqpoly`, `geom_smooth`, `geom_hex`, `geom_step`
 - ▶ `geom_histogram`, `geom_text`, `geom_hline`, `geom_tile`
 - ▶ `geom_vline`, etc.

Otros recursos:

- Libro de Hadley Wickham.
- Web de GGplot2: <http://docs.ggplot2.org>
- Muchos tutoriales online. . .