

# Data visualization

Tidyverse

Juan R Gonzalez  
juanr.gonzalez@isglobal.org

BRGE - Bioinformatics Research Group in Epidemiology  
ISGlobal - Barcelona Institute for Global Health  
<http://brge.isglobal.org>

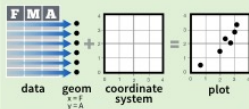
# ggplot2

Every visualization in ggplot2 is composed of the following:

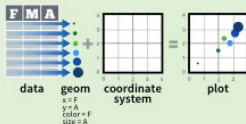
- ▶ Data: the raw material of your visualization
- ▶ Layers: what you see on the plot (e.g., points, lines, ...)
- ▶ Scales: Maps the data to graphical output
- ▶ Coordinates: The visualization's perspective (e.g., grids)
- ▶ Faceting: Provides “visual drill-down” into the data
- ▶ Themes: Controls the details of the display (e.g. fonts)

**plot = data + Aesthetics + geometry**

**ggplot2** is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. `a + geom_bar(stat = "bin")`

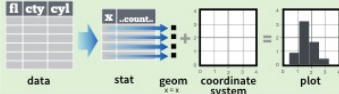


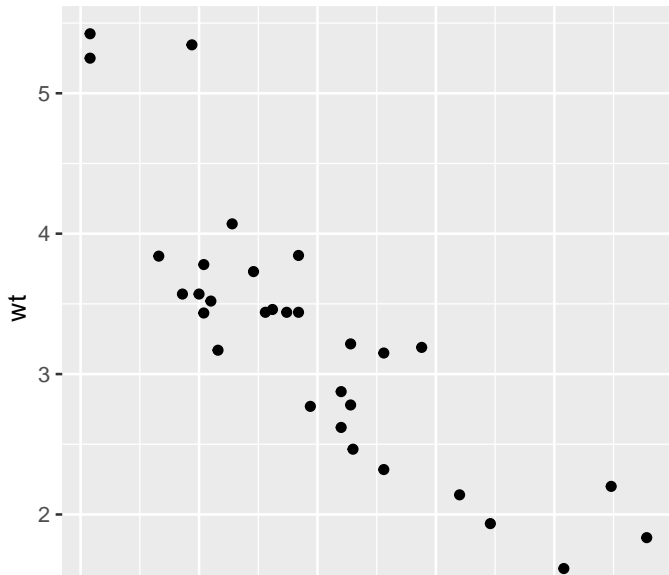
Figure 1: ggplot cheat seet:

<https://www.rstudio.com/resources/cheatsheets/>

```
mtcars[1:5,1:8]
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0

```
library(ggplot2)
ggplot(mtcars) +           # data
  aes(x = mpg, y=wt) +     # Aesthetics
  geom_point()             # geometry (layer)
```

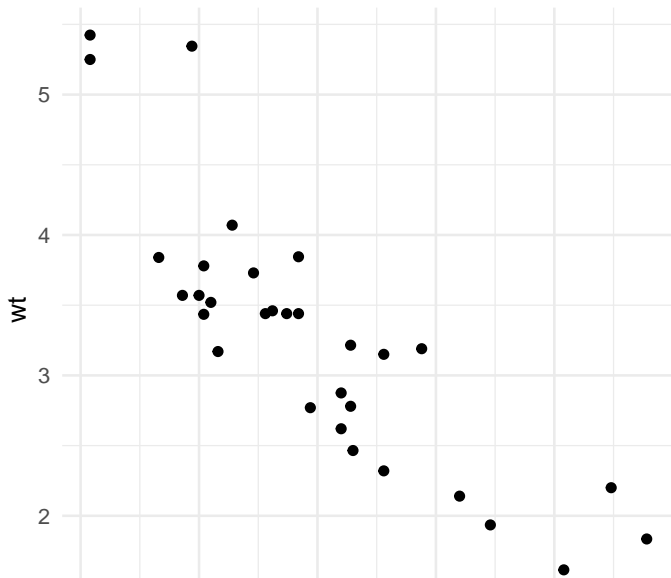


# Possible aesthetics

These are some:

- ▶ `theme_dark()`
- ▶ `theme_minimal()`
- ▶ `theme_classic()`
- ▶ `theme_void()`
- ▶ `theme_test()`

```
ggplot(mtcars) +           # data
  aes(x = mpg, y=wt) +     # Aesthetics
  geom_point() +           # geometry (layer)
  theme_minimal()          # theme
```



## Possible geometry (layers)

- ▶ For one continuous variable:
  - ▶ `geom_area()` for area plot
  - ▶ `geom_density()` for density plot
  - ▶ `geom_dotplot()` for dot plot
  - ▶ `geom_freqpoly()` for frequency polygon
  - ▶ `geom_histogram()` for histogram plot
  - ▶ `stat_ecdf()` for empirical cumulative density function
  - ▶ `stat_qq()` for quantile - quantile plot
- ▶ For one discrete variable:
  - ▶ `geom_bar()` for bar plot



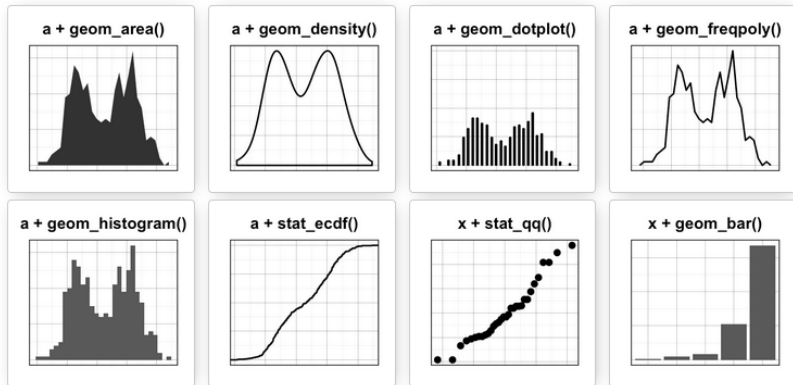
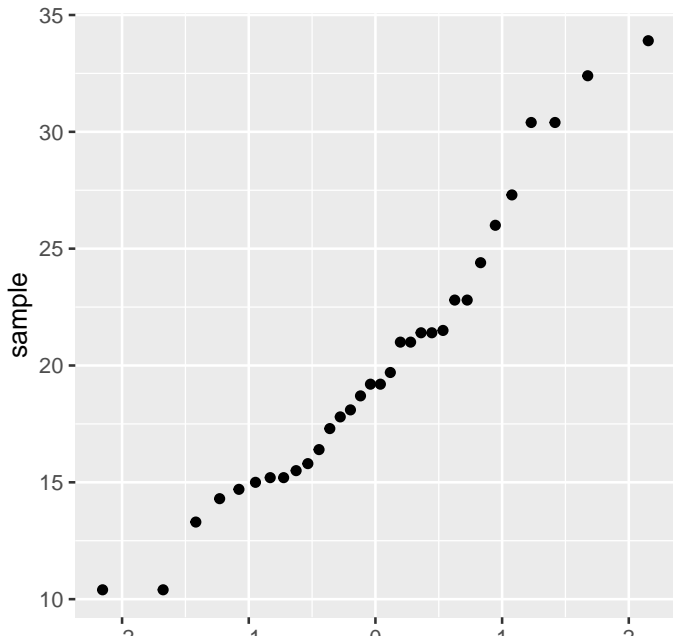


Figure 2: Possible layers

```
ggplot(mtcars) +  
  aes(sample = mpg) + stat_qq()
```



# Data distribution (categorical)

```
library(tidyverse)
diamonds
```

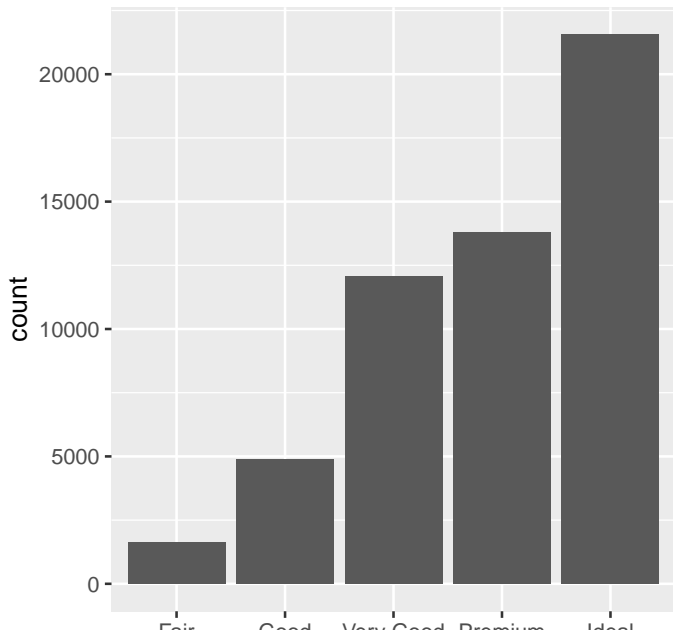
```
# A tibble: 53,940 x 10
  carat cut      color clarity depth table price     x     y     z
  <dbl> <ord>    <ord> <ord>    <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23 Ideal    E      SI2     61.5   55   326   3.95   3.98   2.43
2 0.21 Premium E      SI1     59.8   61   326   3.89   3.84   2.31
3 0.23 Good     E      VS1     56.9   65   327   4.05   4.07   2.31
4 0.290 Premium I      VS2     62.4   58   334   4.2    4.23   2.63
5 0.31 Good     J      SI2     63.3   58   335   4.34   4.35   2.75
6 0.24 Very Good J      VVS2     62.8   57   336   3.94   3.96   2.48
7 0.24 Very Good I      VVS1     62.3   57   336   3.95   3.98   2.47
8 0.26 Very Good H      SI1     61.9   55   337   4.07   4.11   2.53
9 0.22 Fair     E      VS2     65.1   61   337   3.87   3.78   2.49
10 0.23 Very Good H      VS1     59.4   61   338   4      4.05   2.39
# ... with 53,930 more rows
```

```
count(diamonds, cut)
```

```
# A tibble: 5 x 2
  cut      n
  <ord> <int>
1 Fair   1610
2 Good   4906
3 Very Good 12082
4 Premium 13791
5 Ideal  21551
```

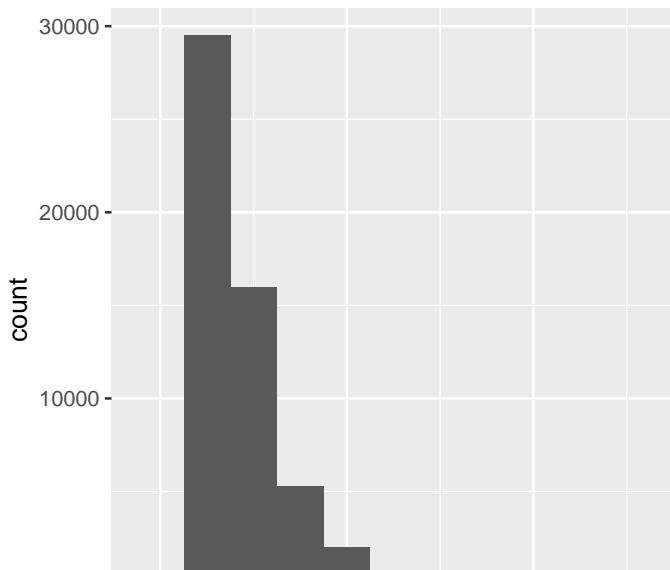
```
ggplot(data = diamonds)
```

```
ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut))
```



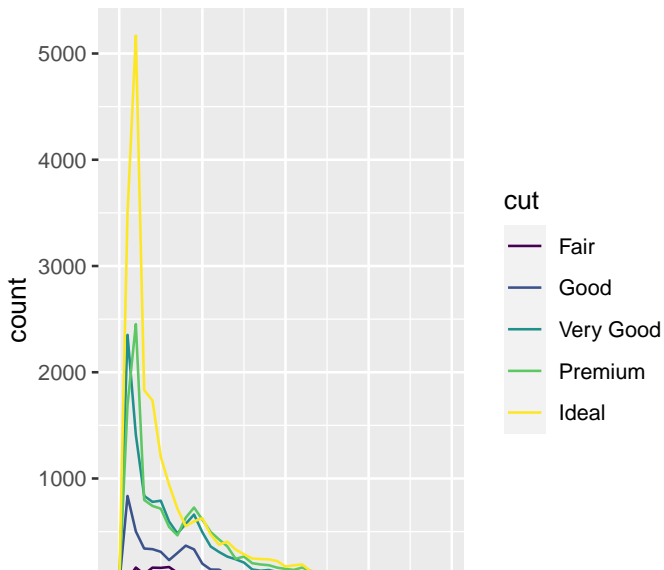
## Data distribution (continous)

```
ggplot(data = diamonds) +  
  geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```



## Data distribution (categorical and continuous)

```
ggplot(data = diamonds, mapping = aes(x = price)) +  
  geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```



# Data distribution (categorical and continuous)

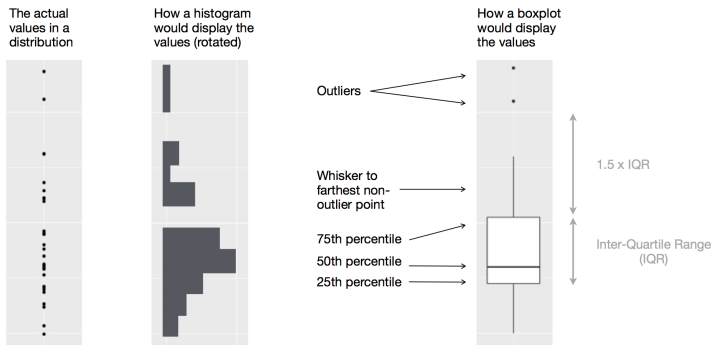
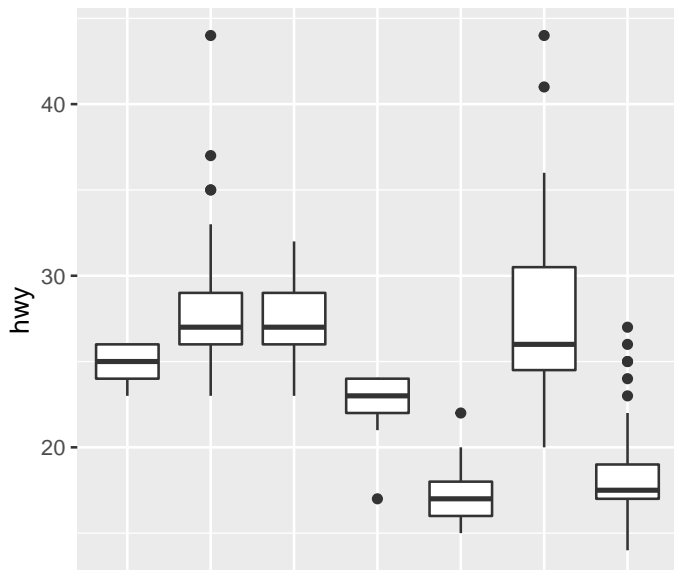


Figure 3: Box-plot

# Boxplot

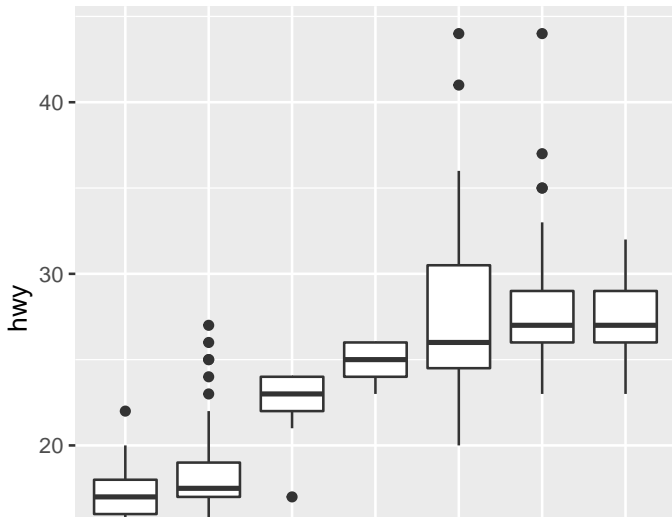
```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +  
  geom_boxplot()
```



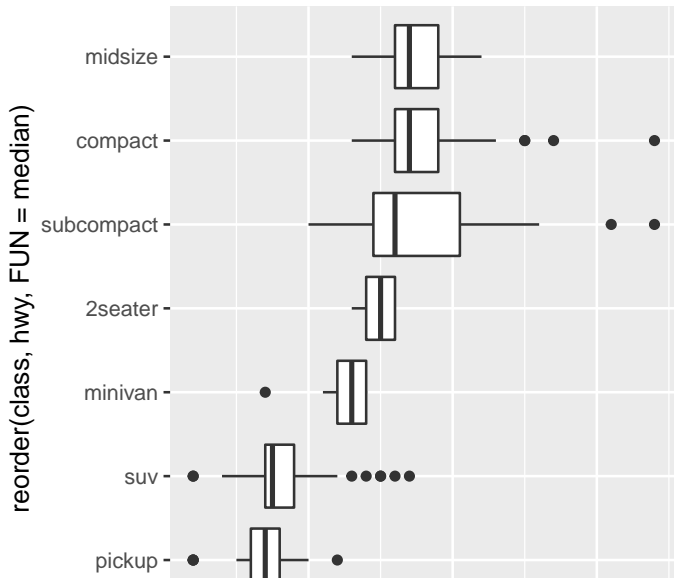


## Reordered boxplot (improves interpretation)

```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = reorder(class, hwy,  
                                FUN = median),  
                              y = hwy))
```

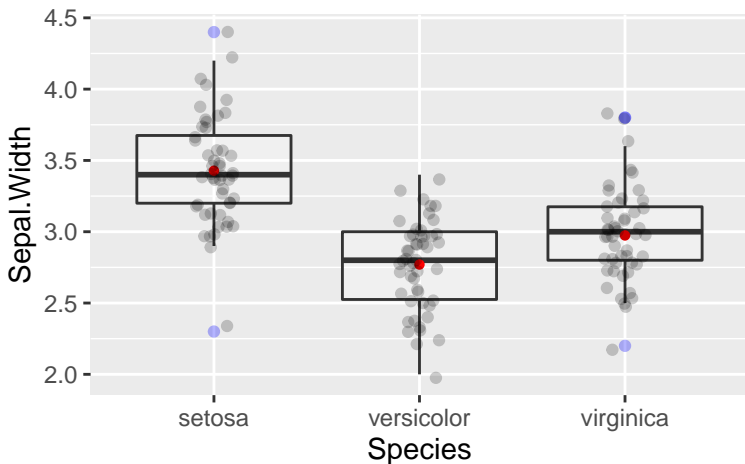


```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = reorder(class, hwy,  
                                         FUN = median),  
                             y = hwy)) + coord_flip()
```



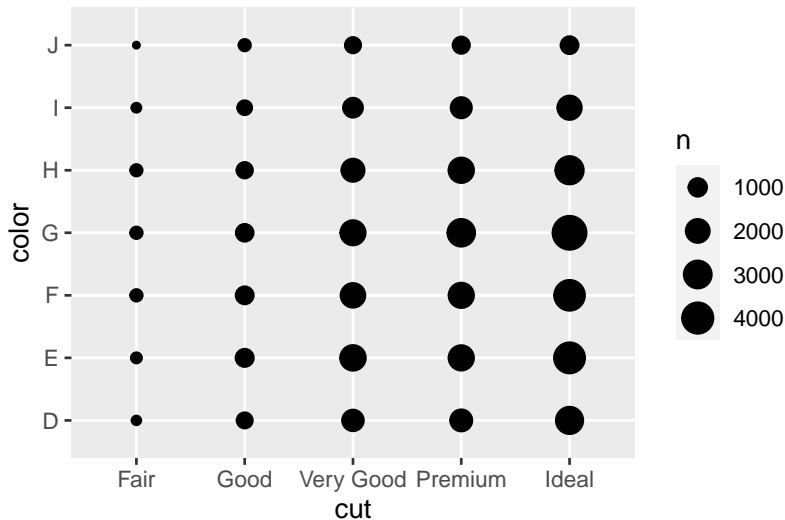
## Improved boxplot

```
ggplot(iris, aes(x=Species, y=Sepal.Width) ) +  
  geom_boxplot(alpha=0.3, outlier.colour = "blue") +  
  geom_point(stat= "summary", fun.y=mean,  
            shape=16, size=1.5, color="red") +  
  geom_jitter(width = 0.1, alpha = 0.2)
```



## Two categorical variables

```
ggplot(data = diamonds) +  
  geom_count(mapping = aes(x = cut, y = color))
```



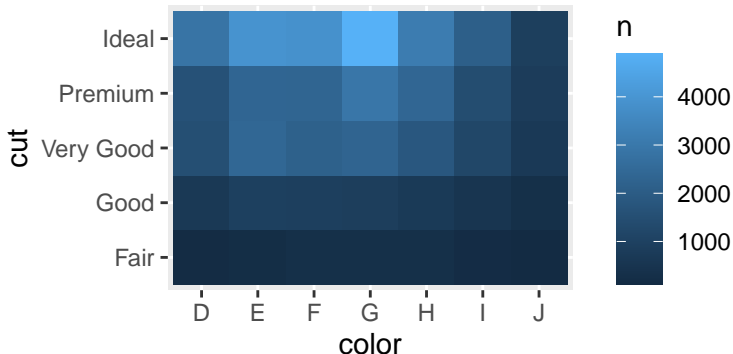
Another approach is to compute the count with dplyr:

```
diamonds %>%  
  count(color, cut)
```

```
# A tibble: 35 x 3  
  color cut      n  
  <ord> <ord>   <int>  
1 D     Fair    163  
2 D     Good    662  
3 D     Very Good 1513  
4 D     Premium 1603  
5 D     Ideal   2834  
6 E     Fair    224  
7 E     Good    933  
8 E     Very Good 2400  
9 E     Premium 2337  
10 E    Ideal   3903  
# ... with 25 more rows
```

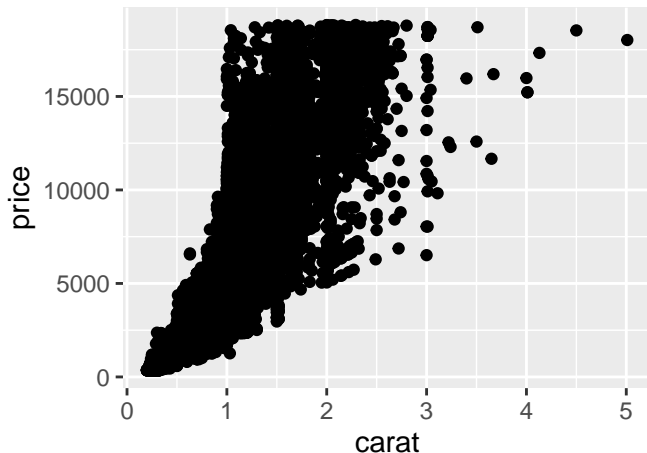
and then visualize with `geom_tile()` and the fill aesthetic:

```
diamonds %>%  
  count(color, cut) %>%  
  ggplot(mapping = aes(x = color, y = cut)) +  
    geom_tile(mapping = aes(fill = n))
```



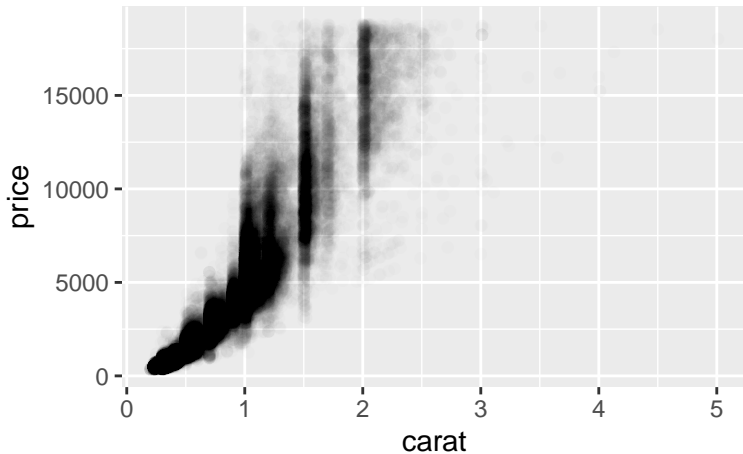
## Two continuos variables

```
ggplot(data = diamonds) +  
  geom_point(mapping = aes(x = carat, y = price))
```



Scatterplots become less useful as the size of your dataset grows because points begin to overplot: use alpha aesthetic:

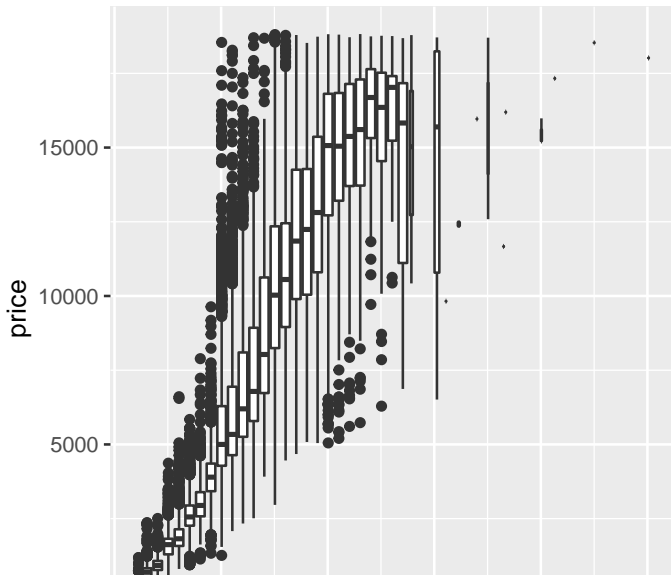
```
ggplot(data = diamonds) +  
  geom_point(mapping = aes(x = carat, y = price),  
              alpha = 1 / 100)
```





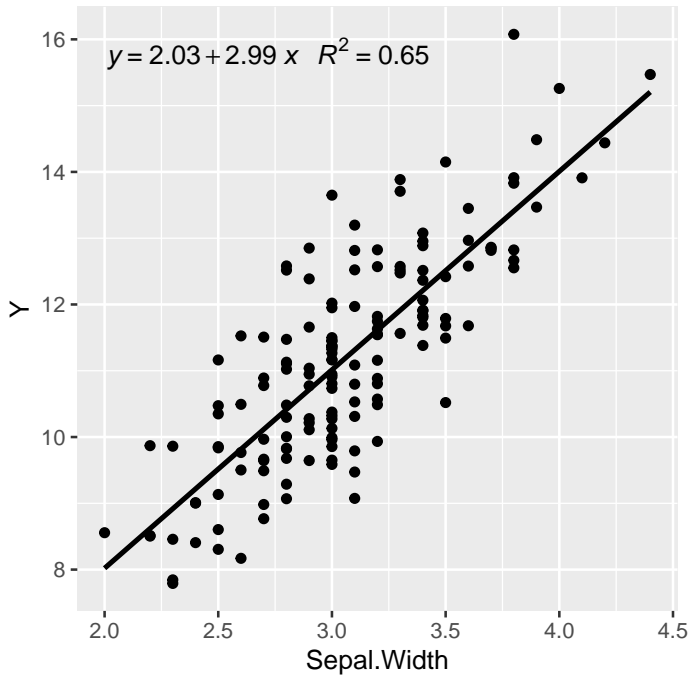
Another option is to bin one continuous variable and use boxplot

```
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +  
  geom_boxplot(mapping = aes(group = cut_width(carat, 0.1)))
```



## Fitting regression line

```
library(ggpmisc)
set.seed(1234)
iris <- mutate(iris,
               Y = 1.5 + 3.2*Sepal.Width +
                 rnorm(nrow(iris)))
ggplot(iris, aes(x = Sepal.Width, y = Y)) +
  geom_smooth(method = "lm", se=FALSE, color="black",
             formula = y ~ x) +
  stat_poly_eq(formula = y ~ x,
              aes(label = paste(..eq.label.., ..rr.label..,
                                sep = "~~~")),
              parse = TRUE) +
  geom_point()
```



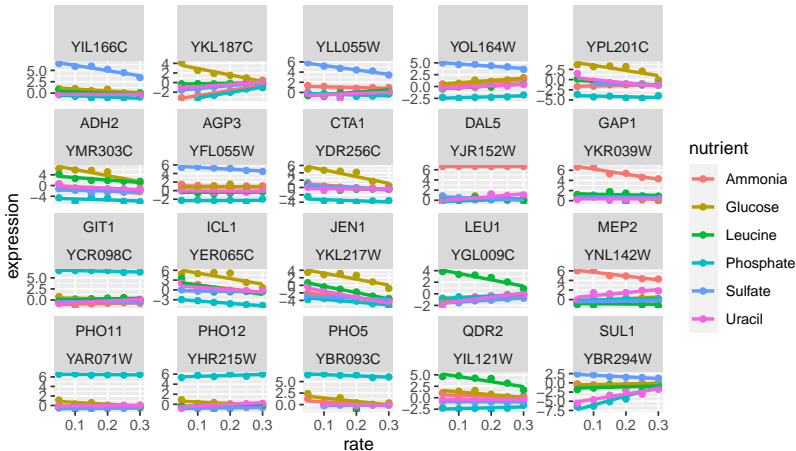
## Why using ggplot2?

Let's imagine we want to compare gene expression by growth rate in twenty genes in six conditions\footnote(example obtained from <http://varianceexplained.org/r/why-i-use-ggplot2/>)

```
load("../..data/genes.Rdata")
genes
```

```
# A tibble: 711 x 7
  name BP MF systematic_name nutrient rate expression
  <chr> <chr> <chr> <chr> <chr> <dbl> <dbl>
1 "SUL1" sulfate tra~ sulfate transp~ YBR294W Glucose 0.05 -0.32
2 "" biological ~ molecular func~ YKL187C Glucose 0.05 4.13
3 "QDR2" multidrug t~ multidrug effl~ YIL121W Glucose 0.05 1.07
4 "LEU1" leucine bio~ 3-isopropylmal~ YGL009C Glucose 0.05 -1.12
5 "PHO5" phosphate m~ acid phosphata~ YBR093C Glucose 0.05 2.39
6 "PHO1~ biological ~ acid phosphata~ YHR215W Glucose 0.05 0.9
7 "PHO1~ phosphate m~ acid phosphata~ YAR071W Glucose 0.05 1.14
8 "GIT1" glycerophos~ glycerophospho~ YCR098C Glucose 0.05 0.77
9 "AGP3" amino acid ~ amino acid tra~ YFL055W Glucose 0.05 0.570
10 "" biological ~ molecular func~ YOL164W Glucose 0.05 0.53
# ... with 701 more rows
```

```
ggplot(genes, aes(rate, expression, color = nutrient)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~name + systematic_name, scales = "free_y")
```



```

par(mar = c(1.5, 1.5, 1.5, 1.5))

colors <- 1:6
names(colors) <- unique(genes$nutrient)

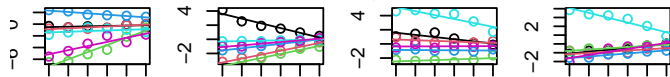
m <- matrix(c(1:20, 21, 21, 21, 21), nrow = 6,
             ncol = 4, byrow = TRUE)
layout(mat = m, heights = c(.18, .18, .18, .18, .18, .1))

genes$combined <- paste(genes$name, genes$systematic_name)
for (gene in unique(genes$combined)) {
  sub_data <- filter(genes, combined == gene)
  plot(expression ~ rate, sub_data,
        col = colors[sub_data$nutrient], main = gene)
  for (n in unique(sub_data$nutrient)) {
    m <- lm(expression ~ rate,
             filter(sub_data, nutrient == n))
    if (!is.na(m$coefficients[2])) {
      abline(m, col = colors[n])
    }
  }
}

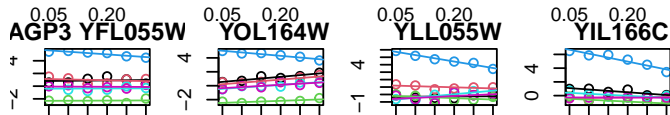
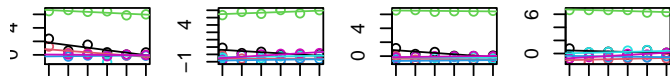
# create a new plot for legend
plot(1, type = "n", axes = FALSE, xlab = "", ylab = "")
legend("top", names(colors), col = colors, horiz = TRUE, lwd = 4

```

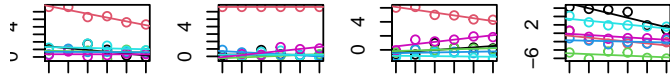
**SUL1 YBR294W YKL187C QDR2 YIL121WLEU1 YGL009C**



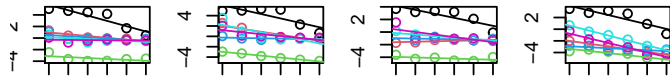
**PHO5 YBR093CCHO12 YHR215VHO11 YAR071VGIT1 YCR098C**



**GAP1 YKR039WDAL5 YJR152WMEP2 YNL142WADH2 YMR303C**



**CTA1 YDR256CICL1 YER065C YPL201C JEN1 YKL217W**



0.05 0.20 0.05 0.20 0.05 0.20 0.05 0.20

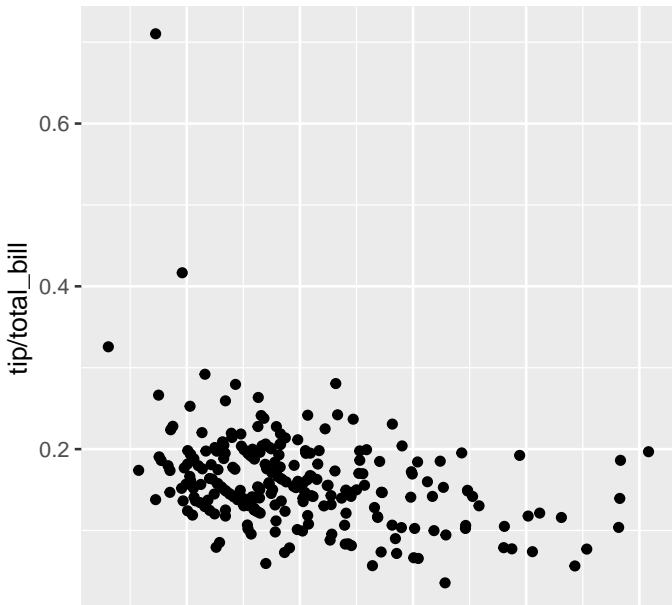
# Facets

```
data(tips, package="reshape2")  
head(tips)
```

	total_bill	tip	sex	smoker	day	time	size
1	16.99	1.01	Female	No	Sun	Dinner	2
2	10.34	1.66	Male	No	Sun	Dinner	3
3	21.01	3.50	Male	No	Sun	Dinner	3
4	23.68	3.31	Male	No	Sun	Dinner	2
5	24.59	3.61	Female	No	Sun	Dinner	4
6	25.29	4.71	Male	No	Sun	Dinner	4

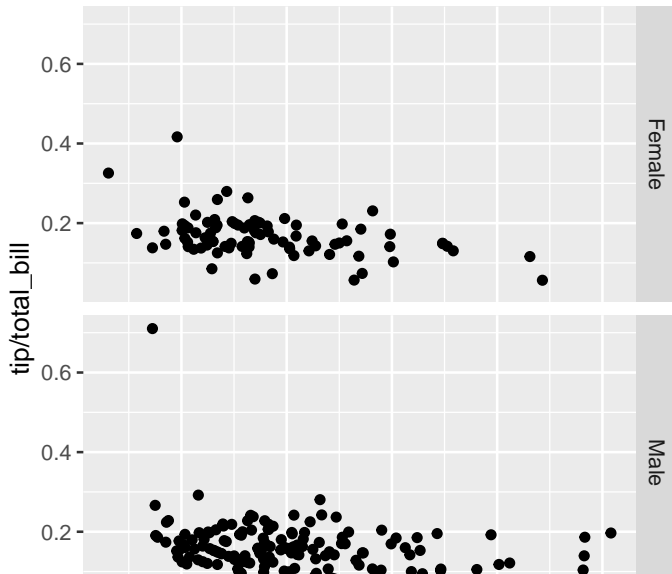


```
sp <- ggplot(tips, aes(x=total_bill, y=tip/total_bill)) +  
  geom_point()  
sp
```

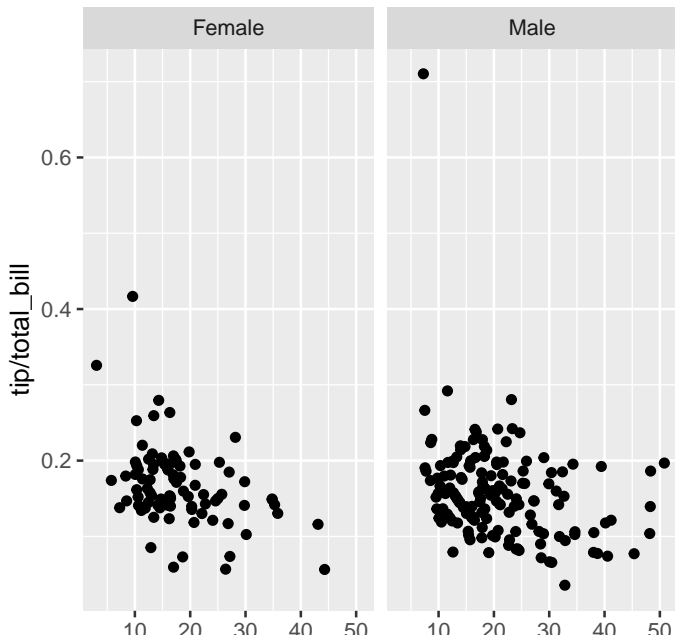


## Facet grid

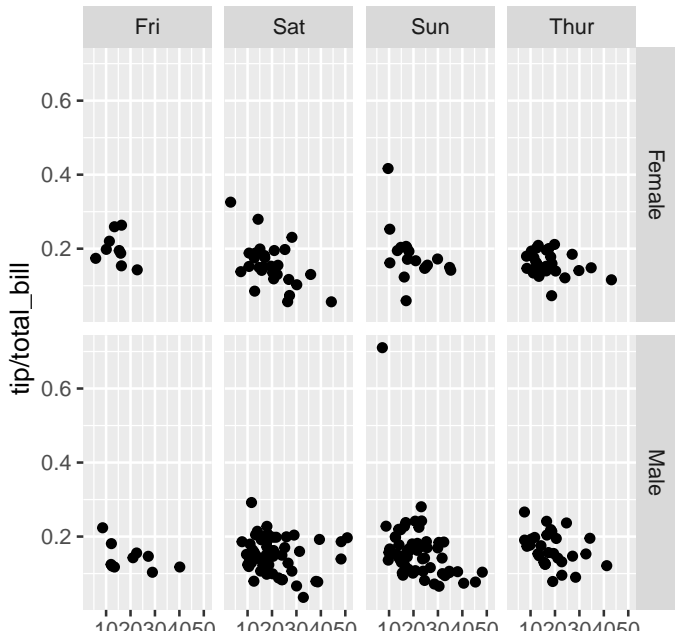
```
# vertical direction  
sp + facet_grid(sex ~ .)
```



```
# horizontal direction  
sp + facet_grid(. ~ sex)
```



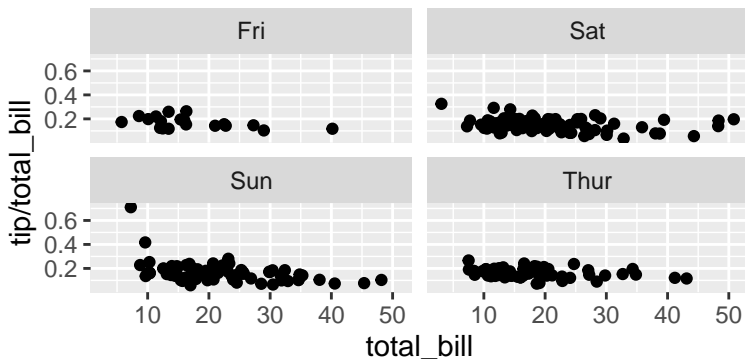
```
# Divide with "sex" vertical, "day" horizontal  
sp + facet_grid(sex ~ day)
```



## Facet wrap

Instead of faceting with a variable in the horizontal or vertical direction, facets can be placed next to each other, wrapping with a certain number of columns or rows. The label for each plot will be at the top of the plot.

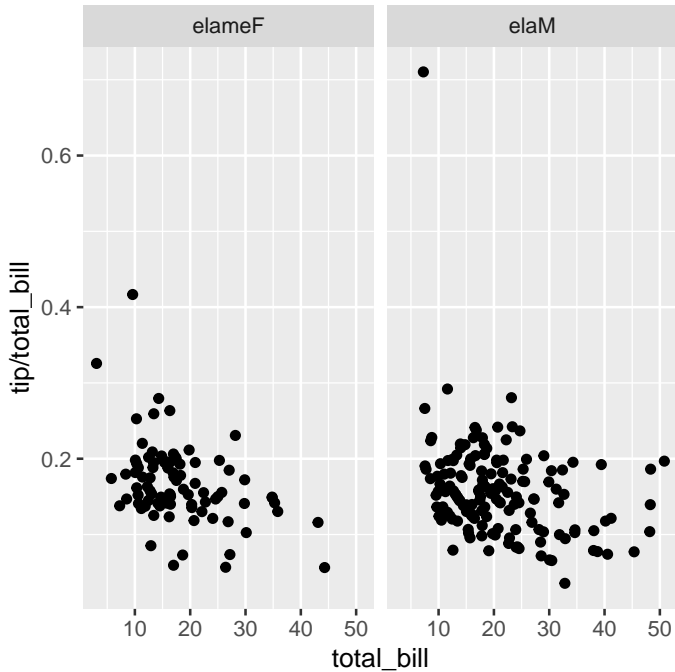
```
# Divide by day, going horizontally and wrapping with 2 columns  
sp + facet_wrap( ~ day, ncol=2)
```



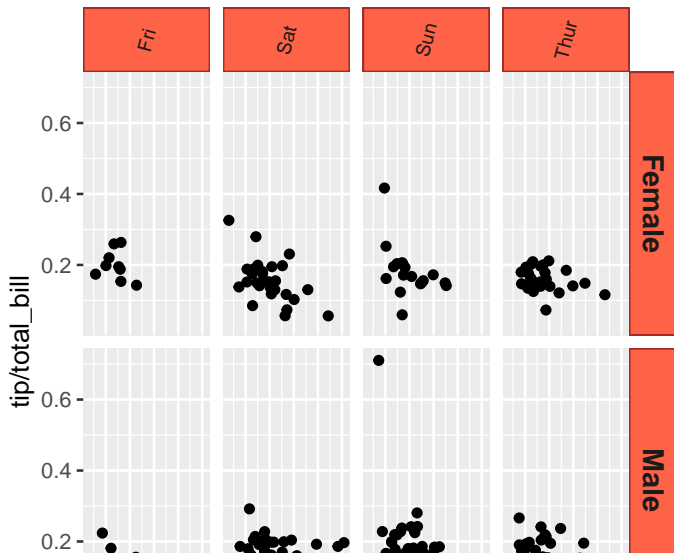
labeller() can use any function that takes a character vector as input and returns a character vector as output (e.g. gsub). We can also define our own custom functions, like this one, which reverses strings:

```
# Reverse each strings in a character vector
reverse <- function(strings) {
  strings <- strsplit(strings, "")
  vapply(strings, function(x) {
    paste(rev(x), collapse = "")
  }, FUN.VALUE = character(1))
}

sp + facet_grid(. ~ sex, labeller=labeller(sex = reverse))
```



```
sp + facet_grid(sex ~ day) +  
  theme(strip.text.x = element_text(size=8, angle=75),  
        strip.text.y = element_text(size=12, face="bold"),  
        strip.background = element_rect(colour="brown",  
                                         fill="tomato"))
```





More about ggplot2 . . . .

<https://ggplot2.tidyverse.org/reference/>

## Ejercicios (Visualización de datos)

1. Visualiza el número de vuelos para cada aerolínea por mes.
2. Visualiza la distribución de la variable carat según el precio en el dataset diamonds.
3. Carga los datos que están en [https://raw.githubusercontent.com/isglobal-brge/TeachingMaterials/master/Master\\_Modelling/data/genome.txt](https://raw.githubusercontent.com/isglobal-brge/TeachingMaterials/master/Master_Modelling/data/genome.txt) en tu sesión de RStudio utilizando la función `read_delim` (NOTA: los datos están delimitados por tabulaciones - no hace falta bajarlos al ordenador se cargan desde la URL).

El archivo contiene información sobre ~730K variantes genéticas en todo el genoma:

- ▶ Name: variante genómica (single nucleotide polymorphism)
- ▶ Chr: cromosoma
- ▶ Position: posición en el cromosoma
- ▶ Log R Ratio: log-ratio de la intensidad de los dos alelos

# Session info

## sessionInfo()

R version 4.0.2 (2020-06-22)  
Platform: x86\_64-w64-mingw32/x64 (64-bit)  
Running under: Windows 10 x64 (build 18362)

Matrix products: default

locale:

[1] LC\_COLLATE=Spanish\_Spain.1252 LC\_CTYPE=Spanish\_Spain.1252  
[3] LC\_MONETARY=Spanish\_Spain.1252 LC\_NUMERIC=C  
[5] LC\_TIME=Spanish\_Spain.1252

attached base packages:

[1] stats graphics grDevices utils datasets methods base

other attached packages:

[1] ggpmisc\_0.3.5 forcats\_0.5.0 stringr\_1.4.0 dplyr\_1.0.2  
[5] purrr\_0.3.4 readr\_1.3.1 tidyr\_1.1.2 tibble\_3.0.3  
[9] tidyverse\_1.3.0 ggplot2\_3.3.2

loaded via a namespace (and not attached):

[1] tidyselect\_1.1.0 xfun\_0.16 lattice\_0.20-41 splines\_4.0.2  
[5] haven\_2.3.1 colorspace\_1.4-1 vctrs\_0.3.3 generics\_0.0.2  
[9] viridisLite\_0.3.0 htmltools\_0.5.0 mgcv\_1.8-33 yaml\_2.2.1  
[13] utf8\_1.1.4 blob\_1.2.1 rlang\_0.4.7 pillar\_1.4.6  
[17] glue\_1.4.2 withr\_2.2.0 DBI\_1.1.0 dbplyr\_1.4.4  
[21] modelr\_0.1.8 readxl\_1.3.1 lifecycle\_0.2.0 munsell\_0.5.0  
[25] gtable\_0.3.0 cellranger\_1.1.0 rvest\_0.3.6 codetools\_0.2-16  
[29] evaluate\_0.14 labeling\_0.3 knitr\_1.29 fansi\_0.4.1  
[33] broom\_0.7.0 Rcpp\_1.0.5 polynom\_1.4-0 backports\_1.1.9  
[37] scales\_1.1.1 jsonlite\_1.7.0 farver\_2.0.3 fs\_1.5.0