

Reproducible science: Module 4

Data visualization in Tidyverse: the power of ggplot2

Gbadamassi G.O. Dossa

Xishuangbanna Tropical Botanical Garden, XTBG-CAS

2021/09/28 (updated: 2021-11-02)

Acknowledgements

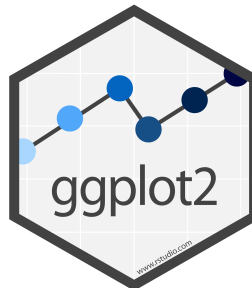
The content of this module are based on materials from:



olivier gimenez's materials

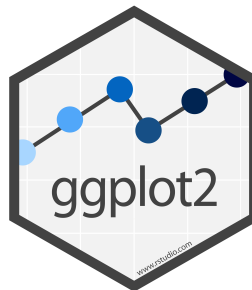
ggplot2: Introduction

- This package was created by [Hadley Wickham](#) check out its [book](#);
- A powerful package for visualizing data;
- The package ggplot2 implements a grammar of graphics;
- Operates on data.frames or tibbles, not vectors like base R;
- Explicitly differentiates between the data and its representation;
- Consists on stacking different layers together, if you have ever worked with GIS, then this notion of layer would be familiar to you.



The ggplot2 grammar

Grammar element	What it is
Data	The data frame being plotted
Geometrics	The geometric shape that will represent the data (e.g., point, boxplot, histogram)
Aesthetics	The aesthetics of the geometric object (e.g., color, size, shape)



ggplot basics

- 1) The ggplot function and the data argument specify a data frame in the main ggplot function

```
#ggplot(data = df) where df= dataframe or tibble
```

- 2) The mapping aesthetics, or aes; most importantly, the variable(s) that we want to plot. aes() specify as an embedded argument in the ggplot() function

```
# ggplot(data = df, mapping = aes(x = h5_median, y = h5_index, color
```

- 3) The geometric objects, or geom; the visual representations specify, after a plus sign +, as an additional function

```
# ggplot(data = df, mapping = aes(x = h5_median, y = h5_index, color
```

Examples of plots

Scatter plots: Import data

We will continue using the precedent data on how twitting can predict citations.

```
# Set the url from where to download the data  
url<-"https://doi.org/10.1371/journal.pone.0166570.s001"  
# name the file to be downloaded and save as destfile object  
destfile <- "twitter_cit_data.csv"  
# Apply download.file function in R to download from url  
download.file(url, destfile)  
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.1
```

```
## Warning: package 'readr' was built under R version 4.1.1
```

```
# Read the data file with read_csv() and save with name "citations_raw"  
citations_raw<-read_csv(file="twitter_cit_data.csv")  
citations <- rename(citations_raw,  
  journal = 'Journal identity',  
  impactfactor = '5-year journal impact factor',  
  pubyear = 'Year published',  
  colldate = 'Collection date',
```

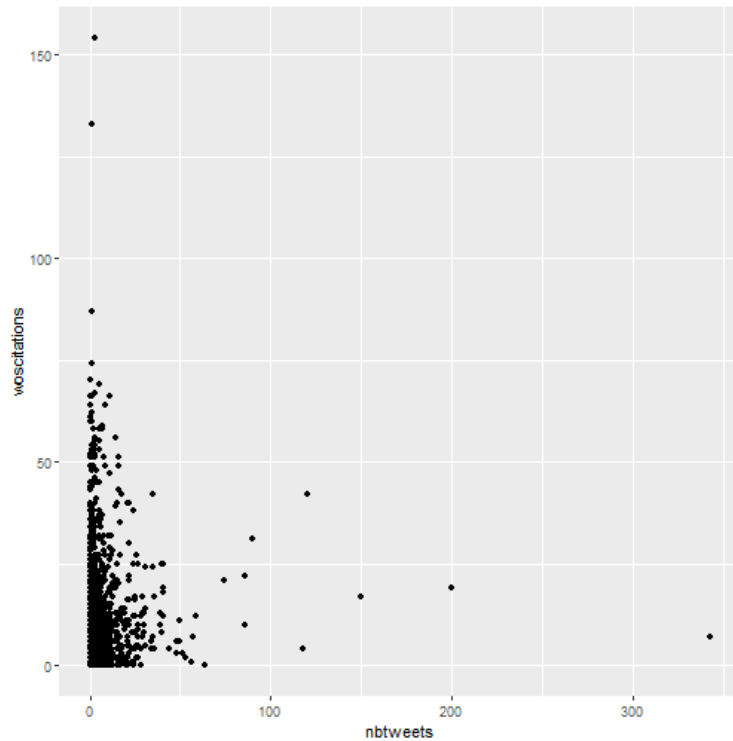
Scatter plot: Plotting

```
scatterplot<-citations %>%  
  ggplot() +  
  aes(x = nbtweets, y = woscitations) +  
  geom_point()
```

- Pass in the data frame as your first argument;
- Aesthetics maps the data onto plot characteristics, here x and y axes
- Display the data geometrically as points

Scatter plot

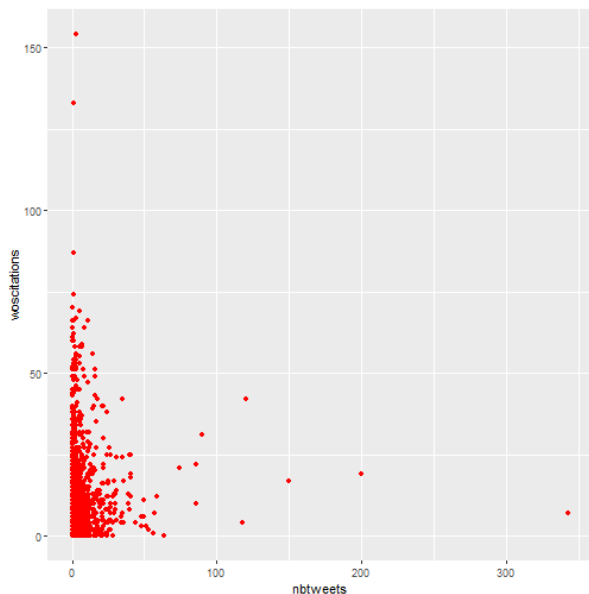
scatterplot



Scatterplots with colors

Puts all points in same color.

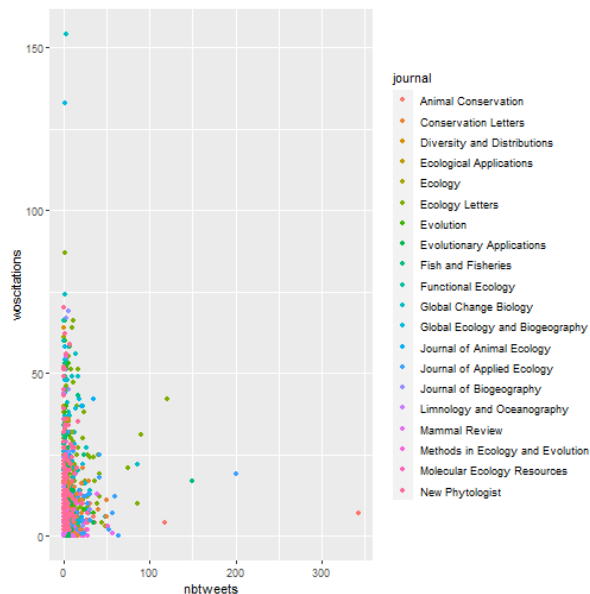
```
scatter_col<-citations %>%  
  ggplot() +  
  aes(x = nbtweets, y = woscitations) +  
  geom_point(color = "red")  
scatter_col
```



Scatterplots with color per species

Gives different color per species.

```
scatter_spcol<-citations %>%  
  ggplot() +  
  aes(x = nbtweets, y = woscitations, color = journal) +  
  geom_point()  
scatter_spcol
```



Scatterplots with shape per journal

Gives different shape per journal. First need to pick few journals. Let's do journal on ecology. Filter these journals to three: JAE, JAppE, Ecol.

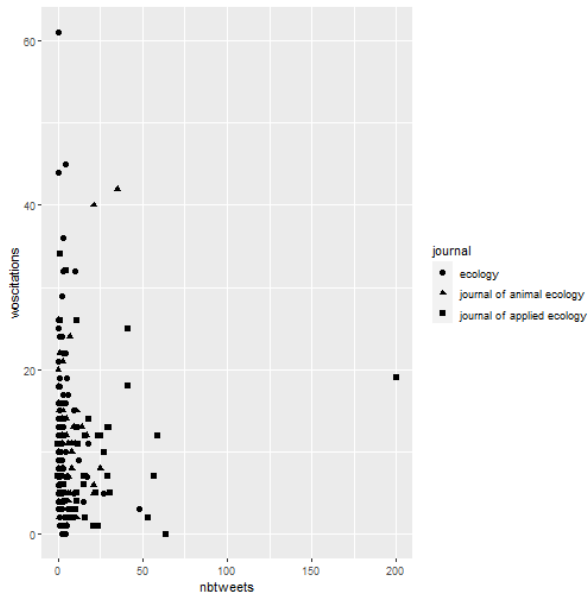
```
citations_ecology <- citations %>%  
  mutate(journal = str_to_lower(journal)) %>% # all journals names to lower  
  filter(journal %in%  
    c('journal of animal ecology', 'journal of applied ecology',  
      'ecology'))  
head(citations_ecology)
```

```
## # A tibble: 6 x 12  
##   journal impactfactor pubyear Volume Issue Authors colldate pubdate nb  
##   <chr>          <dbl>   <dbl>   <dbl> <chr> <chr>      <chr>      <chr>  
## 1 ecology          6.16    2014     95  12   Maglianes~ 3/19/20~ 12/1/2~  
## 2 ecology          6.16    2014     95  12   Soinen     3/19/20~ 12/1/2~  
## 3 ecology          6.16    2014     95  12   Graham an~ 3/19/20~ 12/1/2~  
## 4 ecology          6.16    2014     95  11   White et ~ 3/19/20~ 11/1/2~  
## 5 ecology          6.16    2014     95  11   Einarson ~ 3/19/20~ 11/1/2~  
## 6 ecology          6.16    2014     95  11   Haav and ~ 3/19/20~ 11/1/2~  
## # ... with 3 more variables: Number of users <dbl>, Twitter reach <dbl>,  
## #   woscitations <dbl>
```

Scatterplots with shape per journal

Gives different shape per journal.

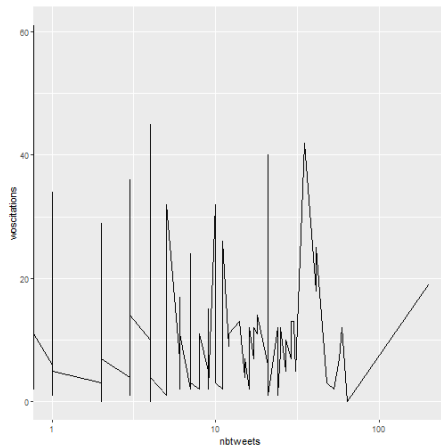
```
scatter_ecol<-citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets, y = woscitations, shape = journal) +  
  geom_point(size=2)  
scatter_ecol
```



Scatterplots with lines not points

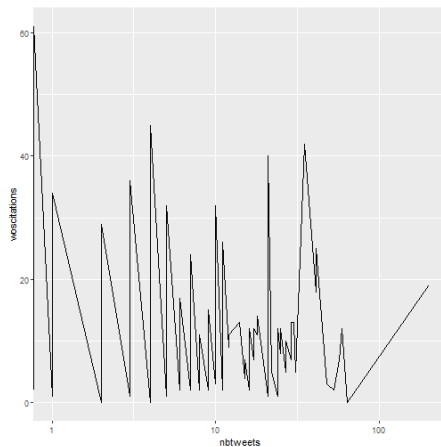
By now, you would guess this requires change in geom, so this should intuitively geom_line.

```
scatter_line<-citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets, y = woscitations) +  
  geom_line() +  
  scale_x_log10()  
scatter_line
```



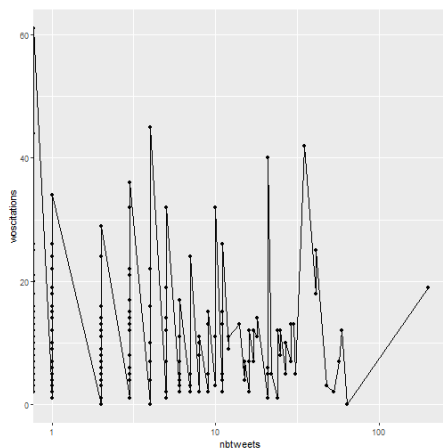
Scatterplots with sorting then add line

```
scatter_line2<-citations_ecology %>%  
  arrange(woscitations) %>%  
  ggplot() +  
    aes(x = nbtweets, y = woscitations) +  
    geom_line() +  
    scale_x_log10()  
scatter_line2
```



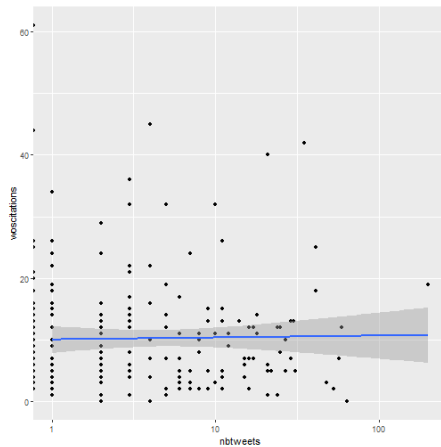
Scatterplots with line and points

```
scatter_line3<-citations_ecology %>%  
  arrange(woscitations) %>%  
  ggplot() +  
  aes(x = nbtweets, y = woscitations) +  
  geom_line() +  
  geom_point() +  
  scale_x_log10()  
scatter_line3
```



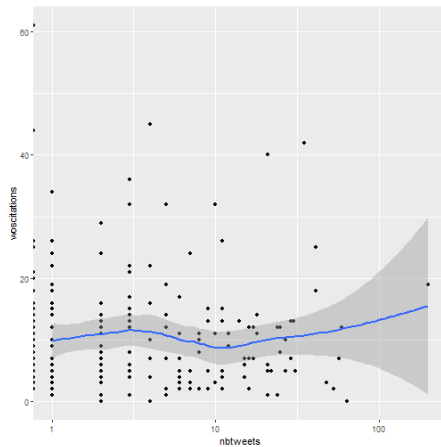
Scatterplots with trend line

```
scatter_line4<-citations_ecology %>%  
  arrange(woscitations) %>%  
  ggplot() +  
  aes(x = nbtweets, y = woscitations) +  
  geom_point() +  
  geom_smooth(method = "lm") +  
  scale_x_log10()  
scatter_line4
```



Scatterplots with smoother

```
scatter_line5<-citations_ecology %>%  
  arrange(woscitations) %>%  
  ggplot() +  
  aes(x = nbtweets, y = woscitations) +  
  geom_point() +  
  geom_smooth() +  
  scale_x_log10()  
scatter_line5
```



aes or not aes?

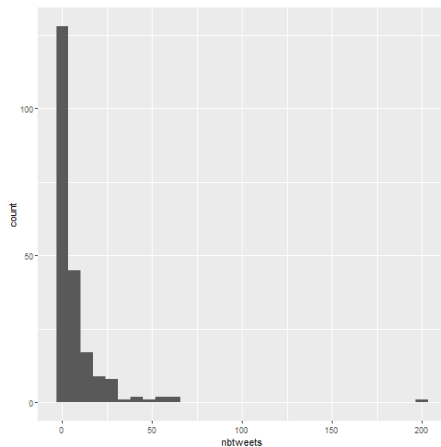
Before continuing to other type of plots, let break to see what we mean by `aes()`.

- If we are to establish a link between the values of a variable and a graphical feature, ie a mapping, then we need an `aes()`.
- Otherwise, the graphical feature is modified irrespective of the data, then we do not need an `aes()`.

Histograms

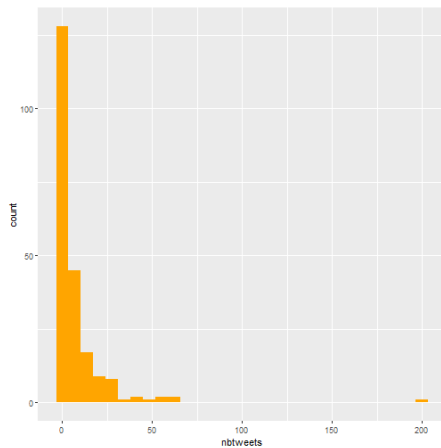
When you only provide x in the aes(), then ggplot will render a histogram.

```
histo<-citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets) +  
  geom_histogram()  
histo
```



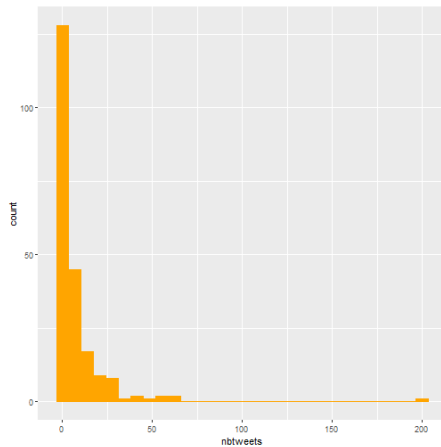
Histograms with bars in colors

```
histo2<-citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets) +  
  geom_histogram(fill = "orange")  
histo2
```



Histograms with bars filled and contour colors

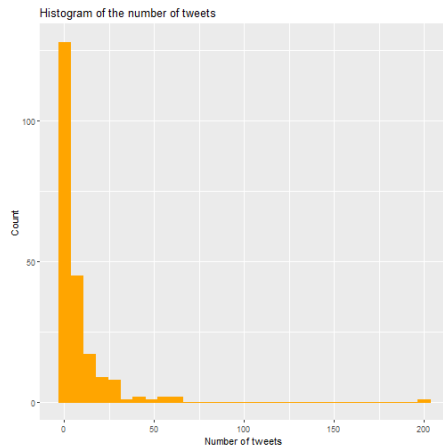
```
histo3<-citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets) +  
  geom_histogram(fill = "orange", color="orange")  
histo3
```



Histograms with labels and title

```
histo4<-citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets) +  
  geom_histogram(fill = "orange", color="orange")+  
  labs(x = "Number of tweets",  
       y = "Count",  
       title = "Histogram of the number of tweets")
```

histo4



Histograms but group this by specific variable

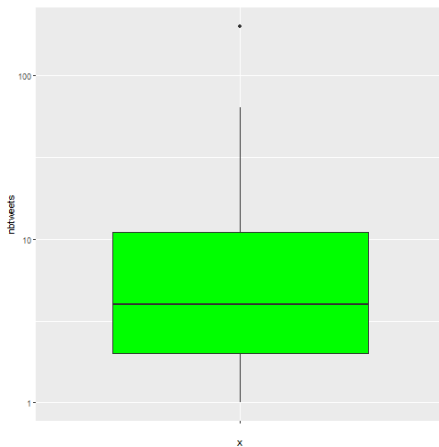
Here we want to have the histogram by journal.

```
histo5<-citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets) +  
  geom_histogram(fill = "orange", color = "brown") +  
  labs(x = "Number of tweets",  
        y = "Count",  
        title = "Histogram of the number of tweets") +  
  facet_wrap(vars(journal))  
histo5
```


Boxplots

Intuitively by now, you would guess this would have something like `geom_boxplot()`. Also, please keep in mind that we would not give x values for the `aes()`, but only y values.

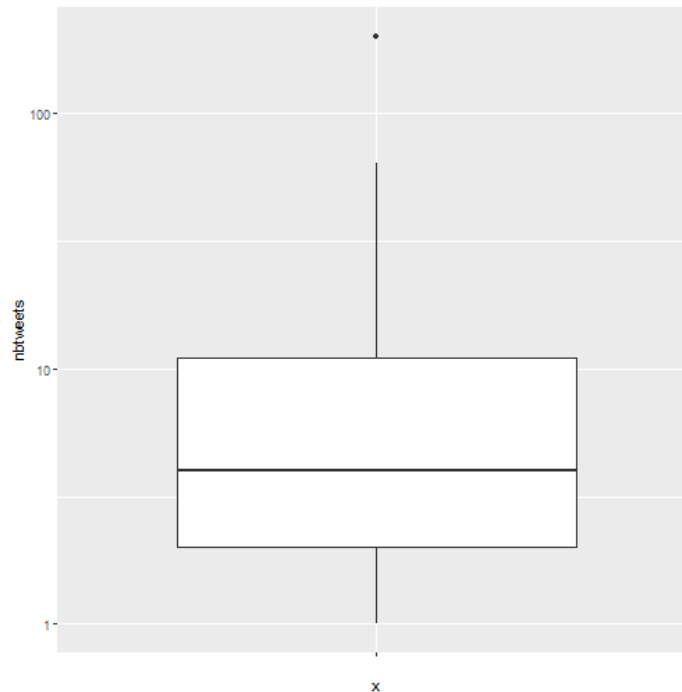
```
boxpl<-citations_ecology %>%  
  ggplot() +  
  aes(x = "", y = nbtweets) +  
  geom_boxplot(fill="green") +  
  scale_y_log10()  
boxpl
```



Some other manipulations

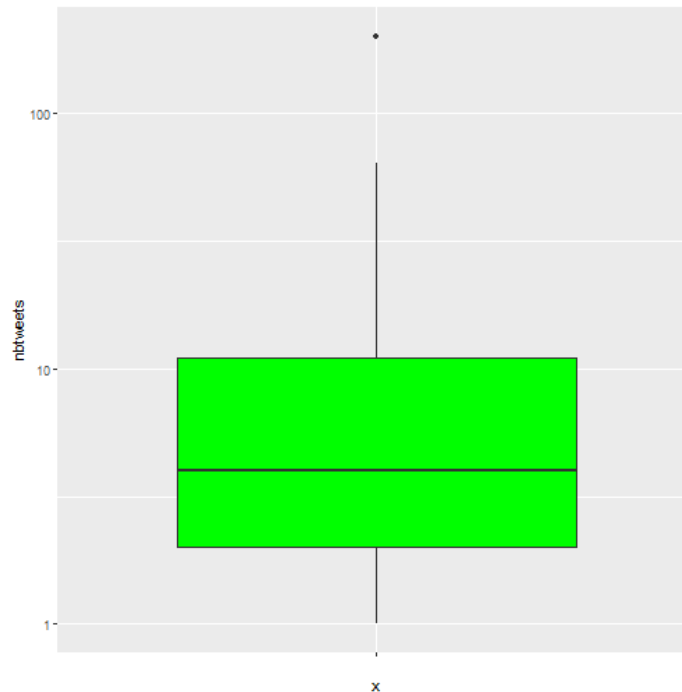
Boxplots

```
citations_ecology %>%  
  ggplot() +  
  aes(x = "", y = nbtweets) +  
  geom_boxplot() +  
  scale_y_log10()
```



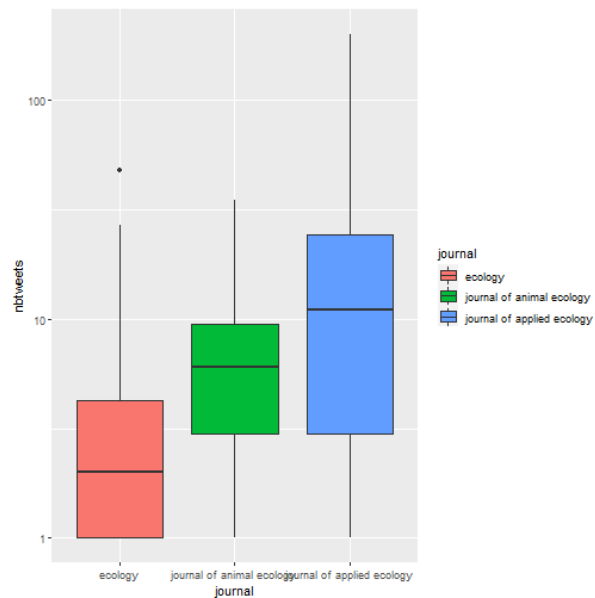
Boxplots with colors

```
citations_ecology %>%  
  ggplot() +  
  aes(x = "", y = nbtweets) +  
  geom_boxplot(fill = "green") +  
  scale_y_log10()
```



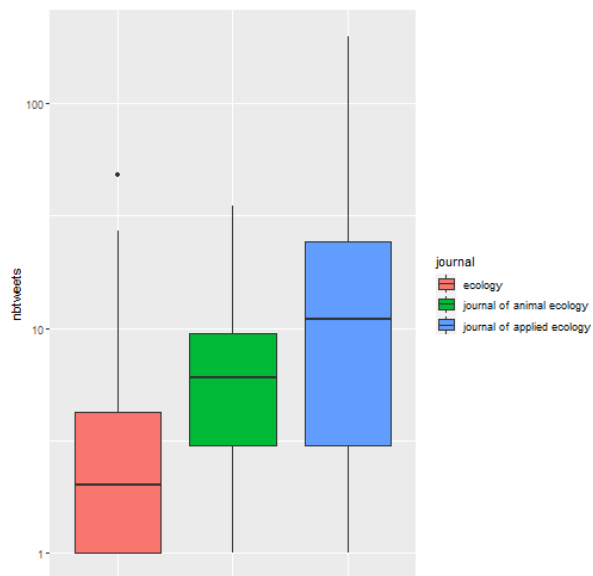
Boxplots with colors by species

```
citations_ecology %>%  
  ggplot() +  
  aes(x = journal, y = nbtweets, fill = journal) +  
  geom_boxplot() +  
  scale_y_log10()
```



Get rid of the ticks on x axis

```
citations_ecology %>%  
  ggplot() +  
  aes(x = journal, y = nbtweets, fill = journal) +  
  geom_boxplot() +  
  scale_y_log10() +  
  theme(axis.text.x = element_blank()) +  
  labs(x = "")
```



Boxplots, user-specified colors by species

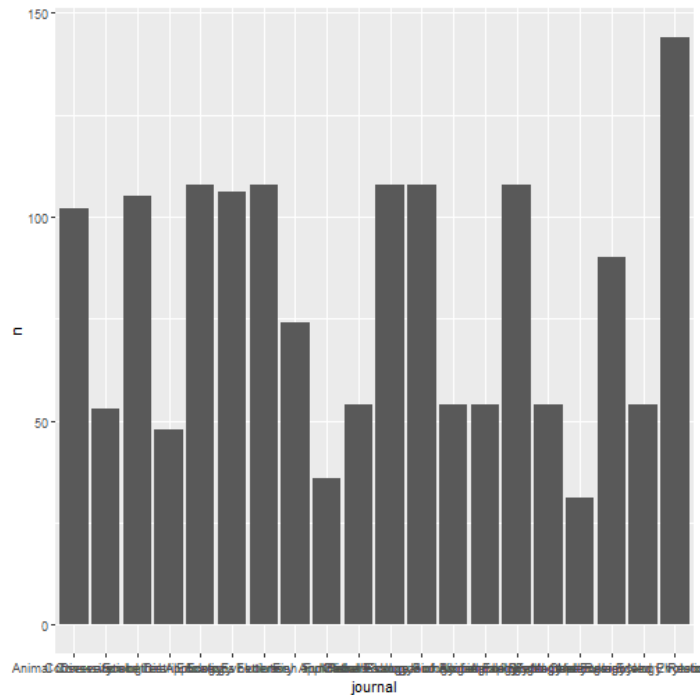
```
citations_ecology %>%  
  ggplot() +  
  aes(x = journal, y = nbtweets, fill = journal) +  
  geom_boxplot() +  
  scale_y_log10() +  
  scale_fill_manual(  
    values = c("red", "blue", "purple")) +  
  theme(axis.text.x = element_blank()) +  
  labs(x = "")
```

Boxplots, change legend settings

```
citations_ecology %>%  
  ggplot() +  
  aes(x = journal, y = nbtweets, fill = journal) +  
  geom_boxplot() +  
  scale_y_log10() +  
  scale_fill_manual(  
    values = c("red", "blue", "purple"),  
    name = "Journal name",  
    labels = c("Ecology", "J Animal Ecology", "J Applied Ecology")) +  
  theme(axis.text.x = element_blank()) +  
  labs(x = "")
```

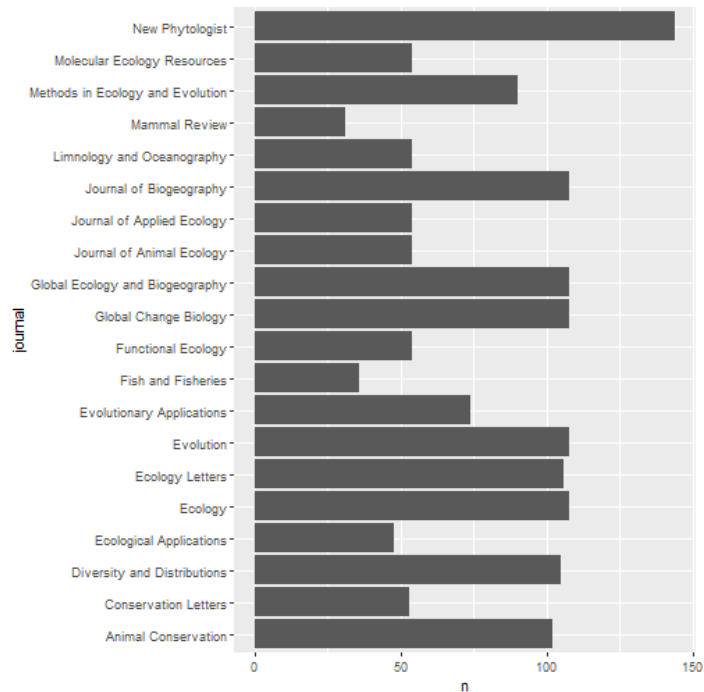

Ugly bar plots

```
citations %>%  
  count(journal) %>%  
  ggplot() +  
  aes(x = journal, y = n) +  
  geom_col()
```



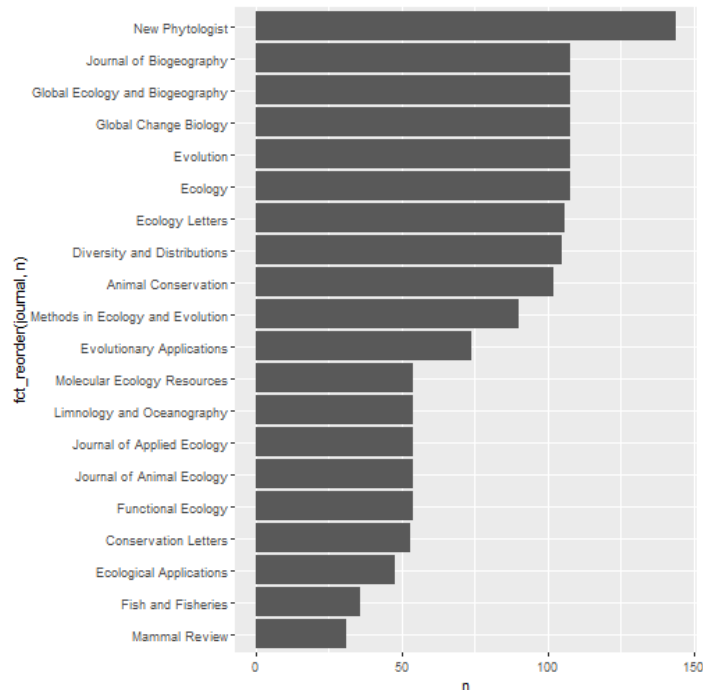
Idem, with flipping

```
citations %>%  
  count(journal) %>%  
  ggplot() +  
  aes(x = n, y = journal) +  
  geom_col()
```



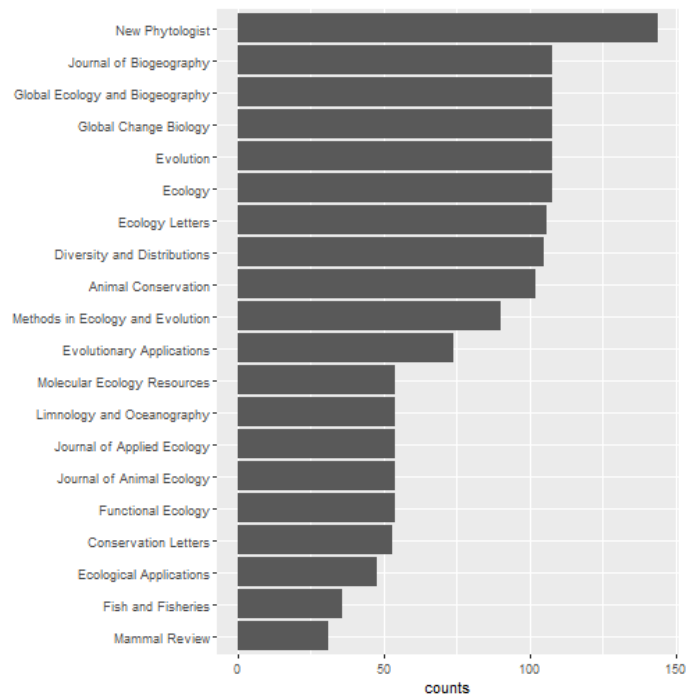
Idem, with factors reordering and flipping

```
citations %>%  
  count(journal) %>%  
  ggplot() +  
  aes(x = n, y = fct_reorder(journal, n)) +  
  geom_col()
```



Further cleaning

```
citations %>%  
  count(journal) %>%  
  ggplot() +  
  aes(x = n, y = fct_reorder(journal, n)) +  
  geom_col() +  
  labs(x = "counts", y = "")
```

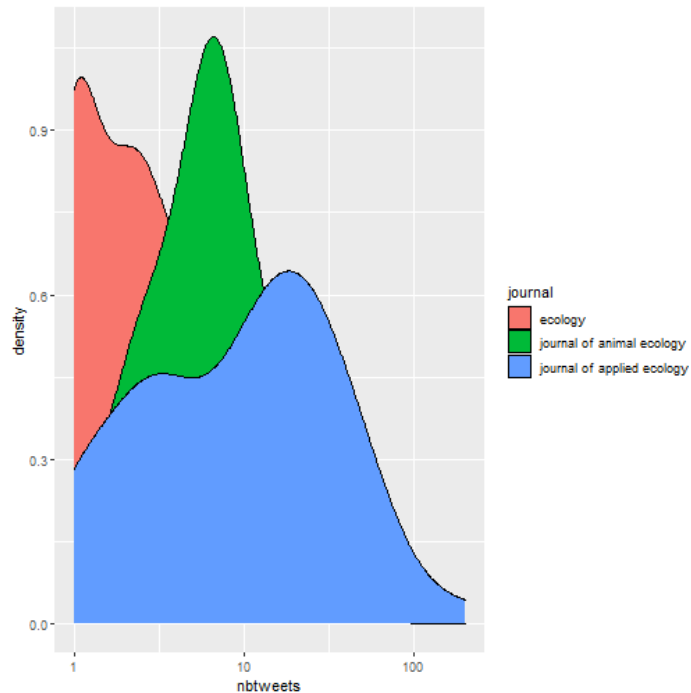


More about how to (tidy) work with factors

- Be the boss of your factors and
- forcats, forcats, vous avez dit forcats ?.

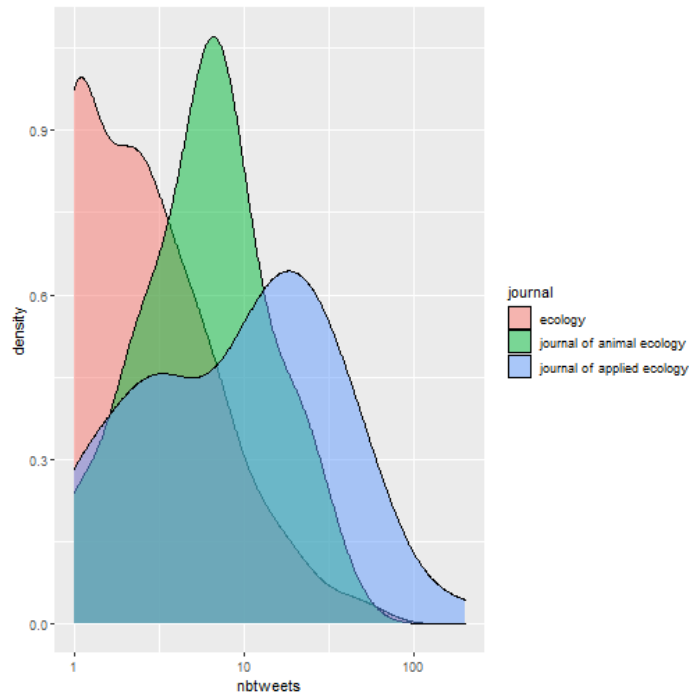
Density plots

```
citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets, fill = journal) +  
  geom_density() +  
  scale_x_log10()
```



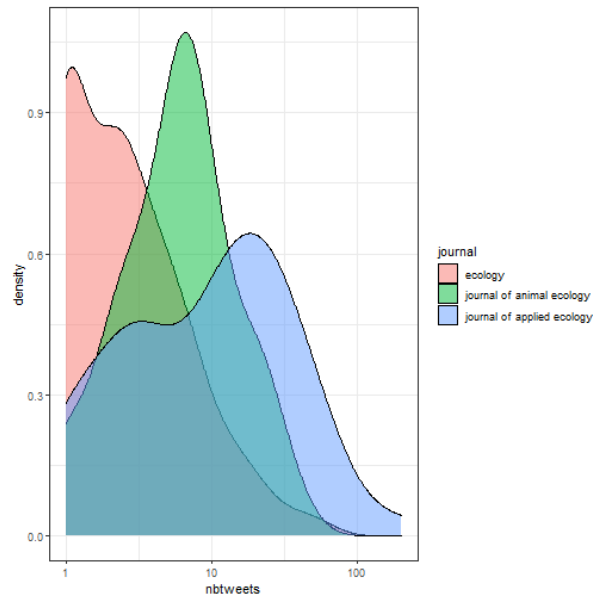
Density plots, control transparency

```
citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets, fill = journal) +  
  geom_density(alpha = 0.5) +  
  scale_x_log10()
```



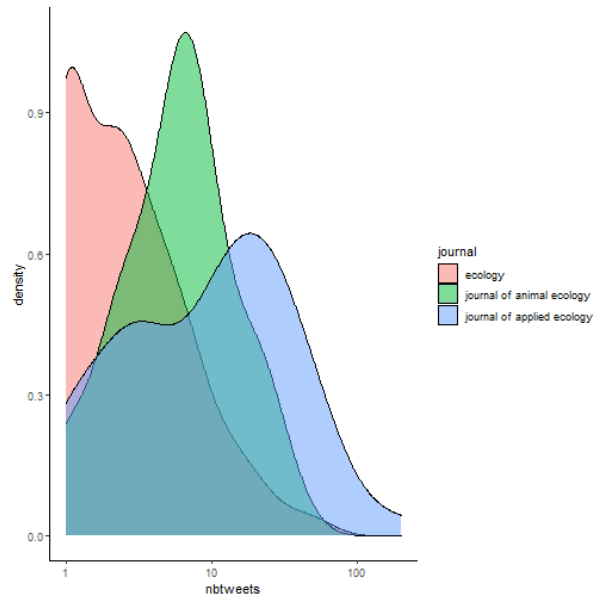
Change default background

```
# `B & W theme`  
citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets, fill = journal) +  
  geom_density(alpha = 0.5) +  
  scale_x_log10() +  
  theme_bw()
```



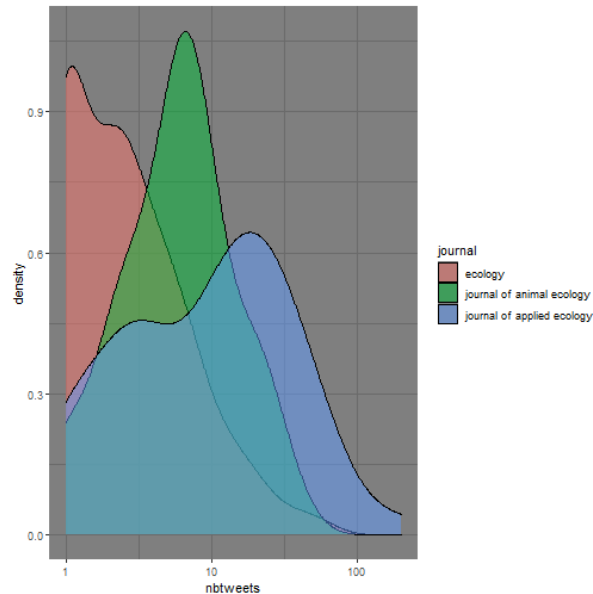
Change default background theme

```
# `classic theme`  
citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets, fill = journal) +  
  geom_density(alpha = 0.5) +  
  scale_x_log10() +  
  theme_classic()
```



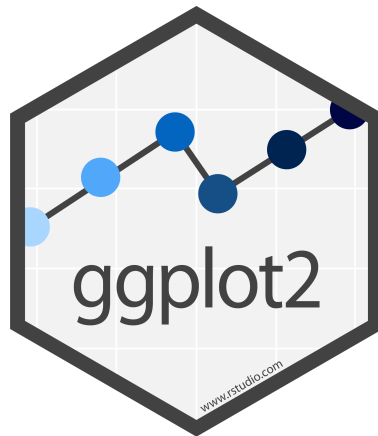
Change default background theme

```
# `dark theme`  
citations_ecology %>%  
  ggplot() +  
  aes(x = nbtweets, fill = journal) +  
  geom_density(alpha = 0.5) +  
  scale_x_log10() +  
  theme_dark()
```



More on data visualisation with ggplot2

- **Portfolio** of ggplot2 plots
- **Cedric Scherer's portfolio** of data visualizations
- **Top** ggplot2 visualizations
- **Interactive** ggplot2 visualizations



To dive deeper in data visualisation with the tidyverse

- **Learn the tidyverse**: books, workshops and online courses
- **R for Data Science** and **Advanced R**
- **Fundamentals of Data visualization**
- **Data Visualization: A practical introduction**
- **Tidy Tuesdays videos** by D. Robinson
- Material of the **2-day workshop Data Science in the tidyverse** held at the RStudio 2019 conference
- Material of the stat545 course on **Data wrangling, exploration, and analysis with R** at the University of British Columbia


Data Visualization with ggplot2 : CHEAT SHEET



The diagram shows the equation: **data** + **geom** = **plot**. The **data** component is represented by a vertical stack of horizontal bars with colored dots (blue, orange, green) at their ends. The **geom** component is a small gray plus sign. The **coordinate** component is a small gray equals sign. The **plot** component is a scatter plot with blue dots on a grid.

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory.
Matches file type to file extension.

d + geom_bar()
x, alpha, color, fill, linetype, size, weight



l + geom_contour(aes(z = z))
 x, y, z, alpha, colour, group, linetype,
 size, weight

- j + geom_linerange()**
x, ymin, ymax, alpha, color, group, linetype, size
- j + geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

```
k + geom_map(aes(map_id = state), map = map)
+ expand_limits(x = map$long, y = map$lat),
map_id, alpha, color, fill, linetype, size
```

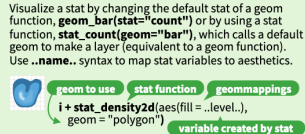
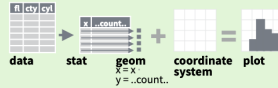
45 / 47

The RStudio Cheat Sheets

Stats

An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).



```
c + stat_bin(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
c + stat_count(width = 1) x, y | ..count.., ..prop..
c + stat_density(adjust = 1, kernel = "gaussian")
x, y | ..count.., ..density.., ..scaled..

e + stat_bin_2d(bins = 30, drop = T)
x, y, fill | ..count.., ..density..
e + stat_bin_hex(bins = 30) x, y, fill | ..count.., ..density..
e + stat_density_2d(contour = TRUE, n = 100)
x, y, color, size | ..level..
e + stat_ellipse(level = 0.95, segments = 51, type = "t")

l + stat_contour(aes(z = z)) x, y, z, order | ..level..
l + stat_summary_hex(aes(z = z), bins = 30, fun = max)
x, y, z, fill | ..value..
l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

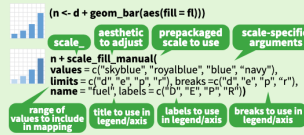
f + stat_boxplot(coef = 1.5) x, y | ..lower.., ..middle.., ..upper.., ..width.., ..ymin.., ..ymax..
f + stat_ydensity(kernel = "gaussian", scale = "area") x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40) x, y | ..x.., ..y..
e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") x, y | ..quantile..
e + stat_smooth(method = "lm", formula = y ~ x, se = T, level = 0.95) x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd = 0.5)) x | ..x.., ..y..
e + stat_identity(na.rm = TRUE)
ggplot() + stat_qq(aes(sample = 1:100), dist = qt, dparam = list(df = 5)) sample, x, y | ..sample.., ..theoretical..
e + stat_sum() x, y, size | ..n.., ..prop..
e + stat_summary(fun.data = "mean_cl_boot")
h + stat_summary_bin(fun.y = "mean", geom = "bar")
e + stat_unique()
```

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.



GENERAL PURPOSE SCALES

Use with most aesthetics
`scale_*` continuous() - map cont' values to visual ones
`scale_*` discrete() - map discrete values to visual ones
`scale_*` manual(values = c()) - map discrete values to manually chosen visual ones
`scale_*` date(date_labels = "%b/m/%d"), date_breaks = "2 weeks" - treat data values as dates.
`scale_*` datetime() - treat data x values as date times. Use same arguments as `scale_x_date()`. See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)
`scale_x_log10()` - Plot x on log10 scale
`scale_x_reverse()` - Reverse direction of x axis
`scale_x_sqrt()` - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)

```
n <- d + geom_bar(aes(fill = fl))
n + scale_fill_brewer(palette = "Blues")
For palette choices:
RColorBrewer::display.brewer.all()
n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")
```

COLOR AND FILL SCALES (CONTINUOUS)

```
o <- c + geom_dotplot(aes(fill = ..x..))
o + scale_fill_distiller(palette = "Blues")
o + scale_fill_gradient(low = "red", high = "yellow")
o + scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25)
o + scale_fill_gradientn(colours = topo.colors(6))
Also: rainbow(), heat.colors(), terrain.colors(),
cm.colors(), RColorBrewer::brewer.pal()
```

SHAPE AND SIZE SCALES

```
p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
Also: shape_manual(values = c(3:7))
p + scale_radius(range = c(1,6))
p + scale_size_area(max_size = 6)
```

Coordinate Systems

```
r <- d + geom_bar()
r + coord_cartesian(xlim = c(0, 5))
xlim, ylim
The default cartesian coordinate system
r + coord_fixed(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units
r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates
r + coord_polar(theta = "x", direction = 1)
theta, start, direction
Polar coordinates
r + coord_trans(ytrans = "sqrt")
xtrans, ytrans, xlim, ylim
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.
r + coord_quickmap()
orientation = c(41, -74, 0) projection, xlim, ylim
Map projections from the maptools package (mercator (default), aequalarea, lagrange, etc.)
```

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```
s <- ggplot(mpg, aes(fl, fill = drv))
s + geom_bar(position = "dodge")
Arrange elements side by side
s + geom_bar(position = "fill")
Stack elements on top of one another,
normalize height
e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting
e + geom_label(position = "nudge")
Nudge labels away from points
s + geom_bar(position = "stack")
Stack elements on top of one another
```

Each position adjustment can be recast as a function with manual width and height arguments

```
s + geom_bar(position = position_dodge(width = 1))
```

Themes

```
r + theme_bw()
White background
with grid lines
r + theme_gray()
Grey background
(default theme)
r + theme_minimal()
Minimal themes
r + theme_void()
Empty theme
```

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

```
t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(cols = vars(fl))
facet into columns based on fl
t + facet_grid(rows = vars(year))
facet into rows based on year
t + facet_grid(rows = vars(year), cols = vars(fl))
facet into both rows and columns
t + facet_wrap(vars(fl))
wrap facets into a rectangular layout
```

Set scales to let axis limits vary across facets

```
t + facet_grid(rows = vars(drv), cols = vars(fl), scales = "free")
```

x and y axis limits adjust to individual facets

```
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust
```

Set labeller to adjust facet labels

```
t + facet_grid(cols = vars(fl), labeller = label_both)
```

ft.c ft.d ft.d ft.p ft.r

```
t + facet_grid(rows = vars(fl), labeller = label_bquote(alpha ^ .(fl)))
```

alpha^ alpha^ alpha^ alpha^ alpha^

Labels

```
t + labs(x = "New x axis label", y = "New y axis label", title = "Add a title above the plot", subtitle = "Add a subtitle below title", caption = "Add a caption below plot", <AES> = "New <AES> legend title")
Use scale functions to update legend labels
```

```
t + annotate(geom = "text", x = 8, y = 9, label = "A")
```

geom to place manual values for geom's aesthetics

Legends

```
n + theme(legend.position = "bottom")
```

Place legend at "bottom", "top", "left", or "right"

```
n + guides(fill = "none")
```

Set legend type for each aesthetic: colorbar, legend, or none (no legend)

```
n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E"))
```

Set legend title and labels with a scale function.

Zooming

```
Without clipping (preferred)
t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20))
With clipping (removes unseen data points)
t + xlim(0, 100) + ylim(10, 20)
t + scale_x_continuous(limits = c(0, 100)) + scale_y_continuous(limits = c(0, 100))
```

Thank you for listening!

Any questions now or email me at dossa@xtbg.org.cn

Slides created via the R package **xaringan**.

The chakra comes from **remark.js**, **knitr**, and **R Markdown**.