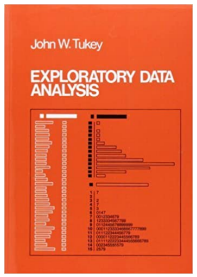
STATS 220

Data visualisation

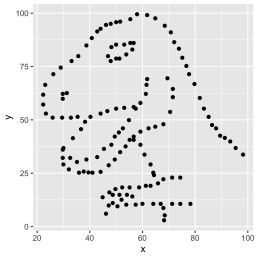
> *The greatest value of a picture is when it forces us to notice what we never expected to see.*
> -- *John W. Tukey*



numbers vs plots

```
dino
```

```
#> # A tibble: 142 x 2
#>       x     y
#>   <dbl> <dbl>
#> 1  55.4  97.2
#> 2  51.5  96.0
#> 3  46.2  94.5
#> 4  42.8  91.4
#> 5  40.8  88.3
#> 6  38.7  84.9
#> # … with 136 more rows
```



numbers vs plots

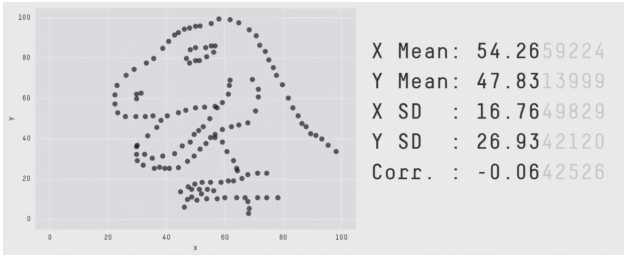


image credit: Steph Locke

Why data visualisation?

> *A picture is worth a thousand words. -- Henrik Ibsen*

1. Data visualisation communicates information much quicker than numerical tables.
2. Data visualisation can reveal unexpected structures in data; it is not surprising that data visualisation is one of the key tools in exploratory data analysis.
3. Data plot is usually more eye-catching even if you lose accuracy of the information.

Charts Graphics

A toy example

```
sci_tbl
```

```
#> # A tibble: 4 x 2
#>   dept             count
#>   <chr>            <int>
#> 1 Physics             12
#> 2 Mathematics          8
#> 3 Statistics          20
#> 4 Computer Science    23
```

- `dept`: discrete/categorical
- `count`: quantitative/numeric

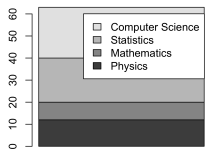
What types of plots can we make?

1. bar plot for counts
2. pie chart for proportions

Named charts

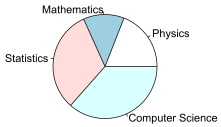
- Bar plot

```
barplot(as.matrix(sci_tbl$count),
  legend = sci_tbl$dept)
```



- Pie chart

```
pie(sci_tbl$count,
  labels = sci_tbl$dept)
```

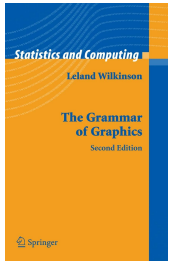


Seems convenient, but ...

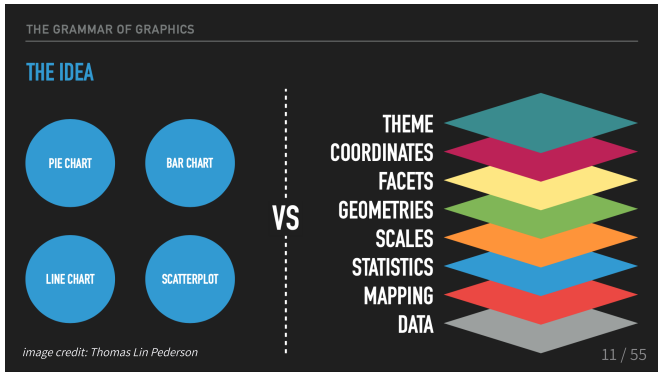
- ✗ a limited set of named charts
- ✗ single purpose functions
- ✗ inconsistent inputs

```
barplot(as.matrix(sci_tbl$count),
  legend = sci_tbl$dept)
```

```
pie(sci_tbl$count,
  labels = sci_tbl$dept)
```

> *Grammar makes language expressive. A language consisting of words and no grammar (statement = word) expresses only as many ideas as there are words. By specifying how words are combined in statements, a grammar expands a language's scope.*

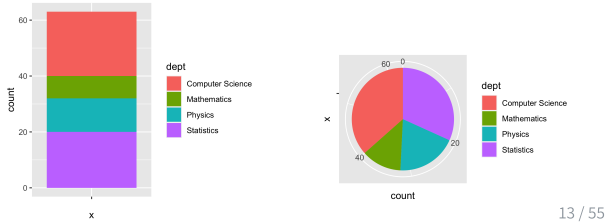


The *grammar of graphics* takes us beyond a limited set of charts (words) to an almost unlimited world of graphical forms (statements).

{ggplot2} provides a cohesive system for declaratively creating elegant graphics, based on The Grammar of Graphics.

```
library(ggplot2)
ggplot(data = sci_tbl) +
  geom_bar(
    aes(x = "", y = count, fill = dept),
    stat = "identity"
  )
```

```
ggplot(data = sci_tbl) +
  geom_bar(
    aes(x = "", y = count, fill = dept),
    stat = "identity"
  ) +
  coord_polar(theta = "y")
```



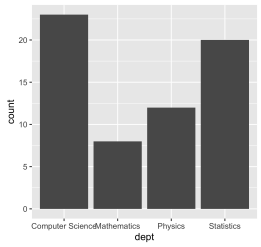
A graphing template

```
ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +
  layer(geom = <GEOM>, stat = <STAT>, position = <POSITION>) +
  layer(geom = <GEOM>, stat = <STAT>, position = <POSITION>)
```

1. `data`: tibble/data.frame.
2. `mapping`: aesthetic mappings between data variables and visual elements, via `aes()`.
3. `layer()`: a graphical layer is a combination of data, stat and geom with a potential position adjustment.
   - `geom`: geometric elements to render each data observation.
   - `stat`: statistical transformations applied to the data prior to plotting.
   - `position`: position adjustment, such as "identity", "stack", "dodge" etc.

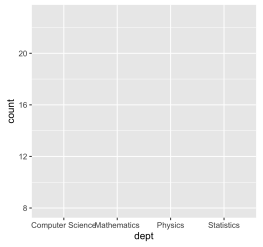
Layers: a bar chart

```
ggplot(data = sci_tbl, mapping = aes(x = dept, y = count)) +
  layer(geom = "bar", stat = "identity", position = "identity")
```



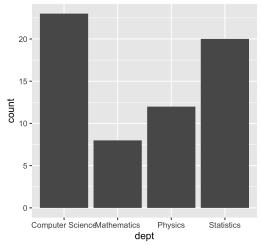
Aesthetic mapping: positional

```
p <- ggplot(sci_tbl, aes(x = dept, y = count))
p
```



Geoms (a shorthand to `layer()`)

```
p +
  geom_bar(stat = "identity")
```



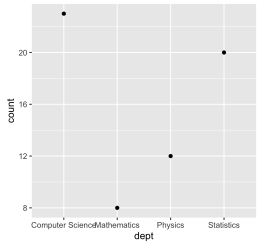

```
p +
  geom_col()
```

- `stat = "identity"` leaves data as is.
- `geom_col()` is a shortcut to `geom_bar(stat = "identity")`.

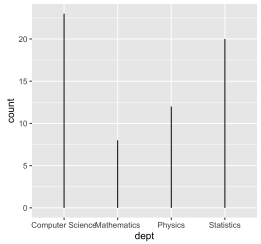
Generally, we use `geom_*()` instead of `layer()` in practice.

Geoms

```
p +
  geom_point()
```

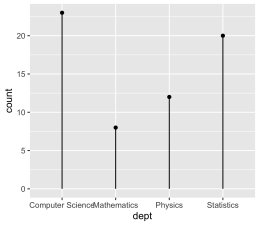



```
p +
  geom_segment(aes(xend = dept, y = 0, yend = count))
```



Composite geoms: lollipop = points + segments

```
p +
  geom_point() +
  geom_segment(aes(xend = dept, y = 0, yend = count))
```



Geom catalogue

geom	Description
geom_abline, geom_hline, geom_vline	Reference lines: horizontal, vertical, and diagonal
geom_bar, geom_col	Bar charts
geom_bin2d	Heatmap of 2d bin counts
geom_blank	Draw nothing
geom_boxplot	A box and whiskers plot (in the style of Tukey)

Previous 1 2 3 4 5 6 7 Next

source code: Emi Tanaka

Stats

- Aggregated (pre-computed)

```
sci_tbl
```

```
#> # A tibble: 4 x 2
#>   dept             count
#>   <chr>            <int>
#> 1 Physics             12
#> 2 Mathematics          8
#> 3 Statistics          20
#> 4 Computer Science    23
```

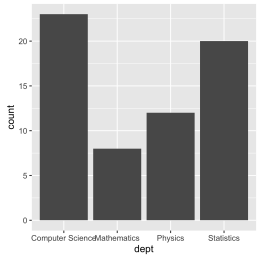
- Disaggregated

```
sci_tbl0
```

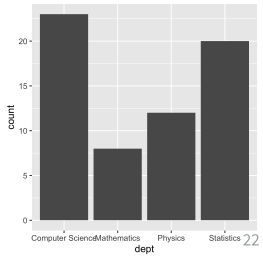
```
#> # A tibble: 63 x 1
#>   dept
#>   <chr>
#> 1 Physics
#> 2 Physics
#> 3 Physics
#> 4 Physics
#> 5 Physics
#> 6 Physics
#> # … with 57 more rows
```


Stats

```
ggplot(sci_tbl, aes(x = dept, y = count)) +
  geom_bar(stat = "identity")
```

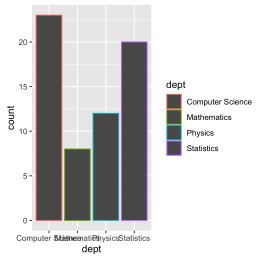



```
ggplot(sci_tbl0, aes(x = dept)) +
  geom_bar(stat = "count")
```

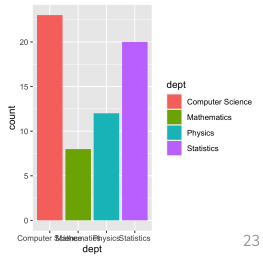


Aesthetic mapping: visual

```
p +
  geom_col(aes(colour = dept))
```

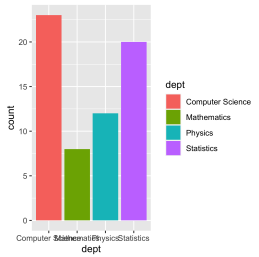



```
p +
  geom_col(aes(fill = dept))
```

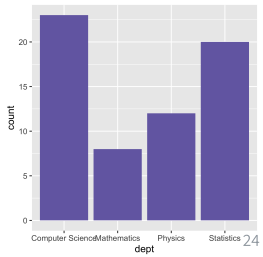


Mapping variables / Setting constants

```
p +
  geom_col(aes(fill = dept))
```

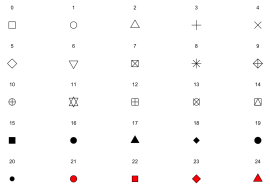



```
p +
  geom_col(fill = "#756bb1")
```



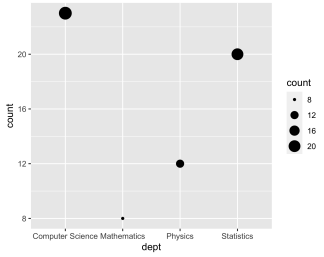
Visual aesthetics

- `colour`/`color`, `fill`:
  - named colours, e.g. `"red"`
  - RGB specification, e.g. `"#756bb1"`
- `alpha`: opacity between 0 and 1
- `shape`:
  - an integer between 0 and 25
  - a single string, e.g. `"triangle open"`
- `linetype`:
  - an integer between 0 and 6
  - a single string, e.g. `"dashed"`
- `size`, `radius`: a numerical value (in millimetres)



Your turn

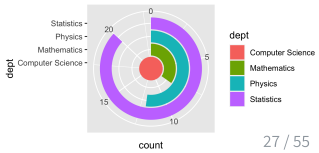
Describe a bubble chart in terms of grammar of graphics.



Coords

- Coordinate systems
  - `coord_cartesian()` (default)
  - ~~`coord_flip()`~~ (deprecated; now you can simply swap `x` and `y`)
  - `coord_map()`
  - `coord_polar()`

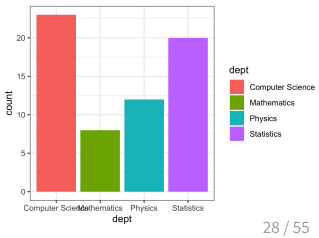
```
p +
  geom_col(aes(fill = dept)) +
  coord_polar(theta = "y")
```



Themes: modify the look

- Built-in ggplot themes
  - `theme_grey()`/`theme_gray()`
  - `theme_bw()`, `theme_linedraw()`
  - `theme_light()`, `theme_dark()`
  - `theme_minimal()`, `theme_classic()`
  - `theme_void()`

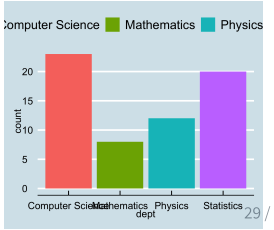
```
p +
  geom_col(aes(fill = dept)) +
  theme_bw()
```



Themes: modify the look

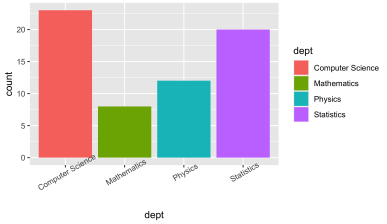
- Many R packages provide themes
  - {ggthemes}
  - {ggthemr}
  - {hrbrthemes}
  - {ggtech}

```
library(ggthemes)
p +
  geom_col(aes(fill = dept)) +
  theme_economist()
```



Modify the look of *texts* with `element_text()`

```
p +
  geom_col(aes(fill = dept)) +
  theme(axis.text.x = element_text(angle = 30))
```



Modify the look of *texts* with `element_text()`

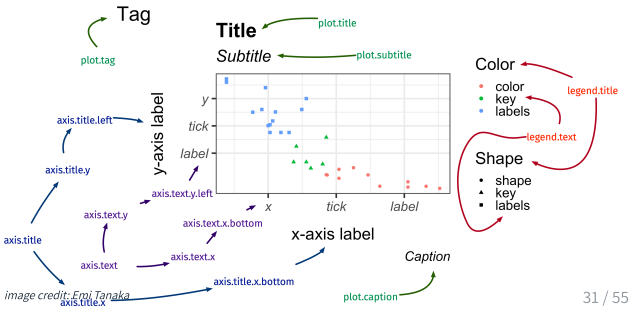
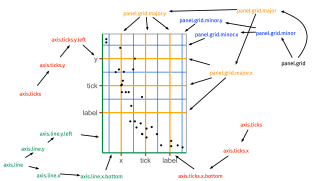


image credit: Emi Tanaka

Modify the look of

lines with `element_line()`



regions with `element_rect()`

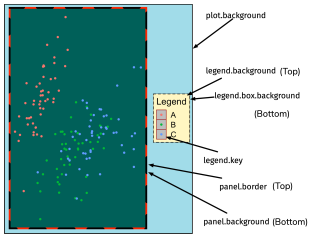


image credit: Emi Tanaka

Small multiples (or trellis/faceting plots)

🌟 the idea of conditioning on the values taken on by one or more of the variables in a data set

Facets

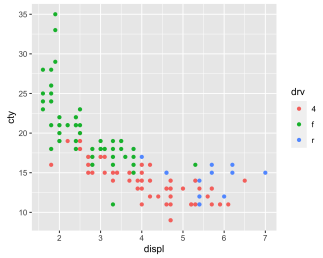
`mpg` data available from {ggplot2}

```
mpg
```

```
#> # A tibble: 234 x 11
#>   manufacturer model displ  year   cyl trans     drv     cty
#>   <chr>        <chr> <dbl> <int> <int> <chr>     <chr> <int>
#> 1 audi         a4      1.8  1999     4 auto(l5)  f        18
#> 2 audi         a4      1.8  1999     4 manual(m… f        21
#> 3 audi         a4      2    2008     4 manual(m… f        20
#> 4 audi         a4      2    2008     4 auto(av)  f        21
#> 5 audi         a4      2.8  1999     6 auto(l5)  f        16
#> 6 audi         a4      2.8  1999     6 manual(m… f        18
#> # … with 228 more rows, and 3 more variables: hwy <int>,
#> #   fl <chr>, class <chr>
```


Facets

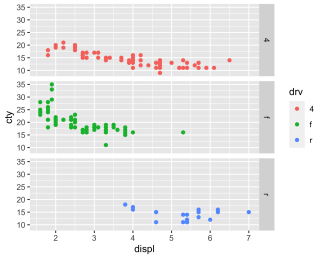
```
p_mpg <- ggplot(mpg, aes(displ, cty)) +
  geom_point(aes(colour = drv))
p_mpg
```



Facets

- `facet_grid()`

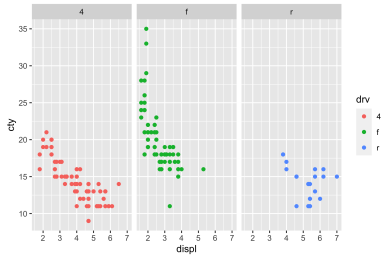
```
p_mpg +
  facet_grid(rows = vars(drv))
  # facet_grid(~ drv)
```



Facets

- `facet_grid()`

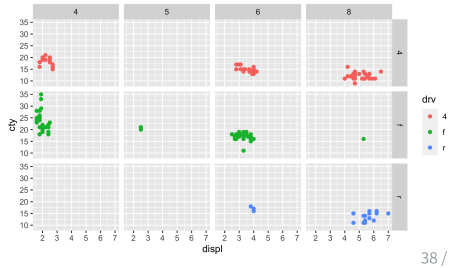
```
p_mpg +
  facet_grid(cols = vars(drv))
  # facet_grid(drv ~ .)
```



Facets

- `facet_grid()`

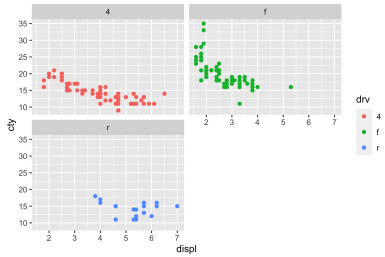
```
p_mpg +
  facet_grid(rows = vars(drv), cols = vars(cyl))
  # facet_grid(cyl ~ drv)
```



Facets

- `facet_grid()`
- `facet_wrap()`

```
p_mpg +
  facet_wrap(vars(drv), ncol = 2)
  # facet_wrap(~ drv, ncol = 2)
```



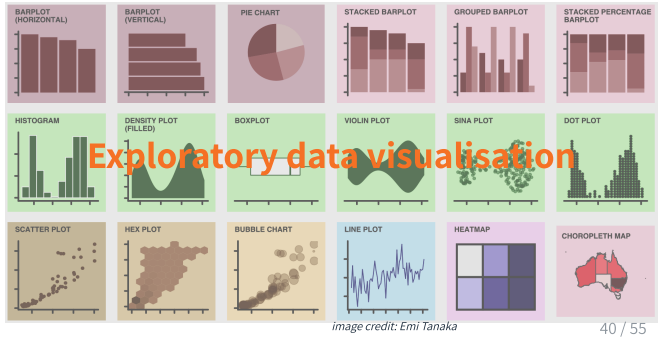


image credit: Emi Tanaka

case study

- **import**

```
movies <- as_tibble(jsonlite::read_json(
  "https://vega.github.io/vega-editor/app/data/movies.json",
  simplifyVector = TRUE))
movies
```

```
#> # A tibble: 3,201 x 16
#>   Title                US_Gross Worldwide_Gross US_DVD_Sales
#>   <chr>                   <int>           <dbl>        <int>
#> 1 The Land Girls         146083          146083           NA
#> 2 First Love, Last Ri…    10876           10876           NA
#> 3 I Married a Strange…   203134          203134           NA
#> 4 Let's Talk About Sex   373615          373615           NA
#> 5 Slam                  1009819         1087521           NA
#> 6 Mississippi Mermaid     24551         2624551           NA
#> # … with 3,195 more rows, and 12 more variables:
#> #   Production_Budget <int>, Release_Date <chr>,
#> #   MPAA_Rating <chr>, Running_Time_min <int>,
#> #   Distributor <chr>, Source <chr>, Major_Genre <chr>,
#> #   Creative_Type <chr>, Director <chr>,
#> #   Rotten_Tomatoes_Rating <int>, IMDB_Rating <dbl>,
#> #   IMDB_Votes <int>
```


case study

- **import**
- **skim**

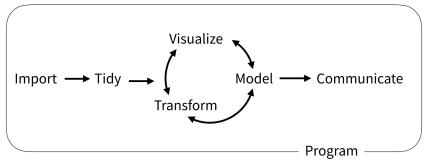
```
skimr::skim(movies)
```

```
#> ── Data Summary ────────────────────
#>                            Values
#> Name                       movies
#> Number of rows             3201
#> Number of columns          16
#> ___________________________
#> Column type frequency:
#>   character                8
#>   numeric                  8
#> ___________________________
#> Group variables            None
#>
#> ── Variable type: character ──────────────────────────────────────────────────────────────────────
#>   skim_variable n_missing complete_rate   min   max empty n_unique whitespace
#> 1 Title                 1         1.00     1    66     0     3176          0
#> 2 Release_Date          7         0.998    8    11     0     1603          0
#> 3 MPAA_Rating         605         0.811    1     9     0        7          0
#> 4 Distributor         232         0.928    3    33     0      174          0
#> 5 Source              365         0.886    6    29     0       18          0
#> 6 Major_Genre         275         0.914    5    19     0       12          0
#> 7 Creative_Type       446         0.861    7    23     0        9          0
#> 8 Director           1331         0.584    7    27     0      550          0
#>
#> ── Variable type: numeric ────────────────────────────────────────────────────────────────────────
#>   skim_variable     n_missing complete_rate         mean           sd
#> 1 US_Gross                  7         0.998 44002085.    62555311.
#> 2 Worldwide_Gross           7         0.998 85343400.   149947343.
```


case study

- import
- skim
- **vis**

› Data analysis starts with questions.

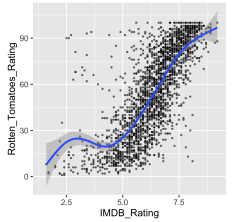


case study

- import
- skim
- **vis**

Are movies ratings consistent b/t IMDB & Rotten Tomatoes

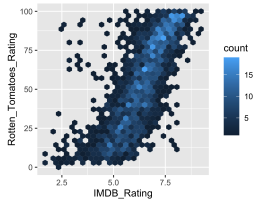
```
ggplot(movies, aes(x = IMDB_Rating, y = Rotten_Tomatoes_Rating)) +
  geom_point(size = 0.5, alpha = 0.5) +
  geom_smooth(method = "gam") +
  theme(aspect.ratio = 1)
```



case study
- import
- skim
- vis

Are movies ratings consistent b/t IMDB & Rotten Tomatoes

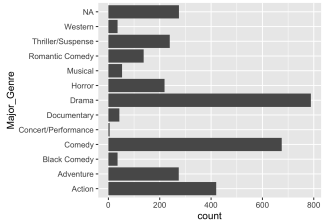
```
ggplot(movies, aes(x = IMDB_Rating, y = Rotten_Tomatoes_Rating)) +
  geom_hex() +
  theme(aspect.ratio = 1)
```



case study
- import
- skim
- vis

The popularity of major genre

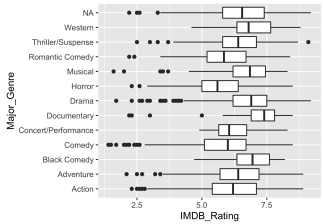
```
ggplot(movies, aes(y = Major_Genre)) +
  geom_bar()
```



case study
- import
- skim
- vis

The likeness of major genre

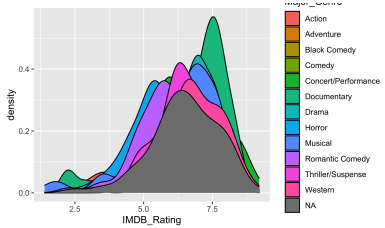
```
ggplot(movies) +
  geom_boxplot(aes(x = IMDB_Rating, y = Major_Genre))
```



case study
- import
- skim
- vis

The likeness of major genre

```
ggplot(movies) +
  geom_density(aes(x = IMDB_Rating, fill = Major_Genre))
```

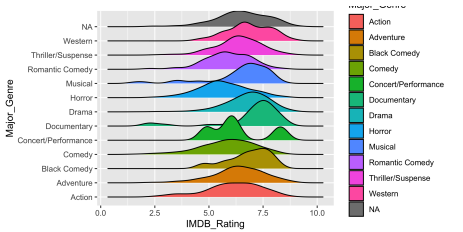


case study

- import
- skim
- **vis**

The likeness of major genre

```
library(ggridges)
ggplot(movies, aes(x = IMDB_Rating, y = Major_Genre)) +
  geom_density_ridges(aes(fill = Major_Genre))
```

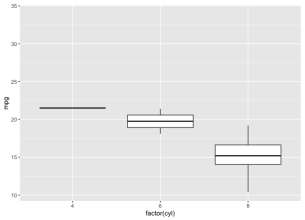


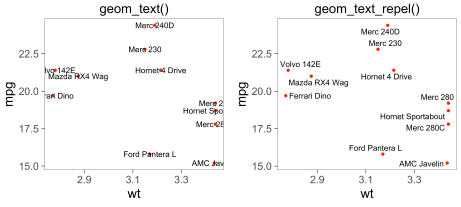
{ggplot2}-ext 📦

> *{ggplot2} now has an official extension mechanism. This means that others can now easily create their own stats, geoms and positions, and provide them in other packages. This should allow the ggplot2 community to flourish, even as less development work happens in ggplot2 itself.*

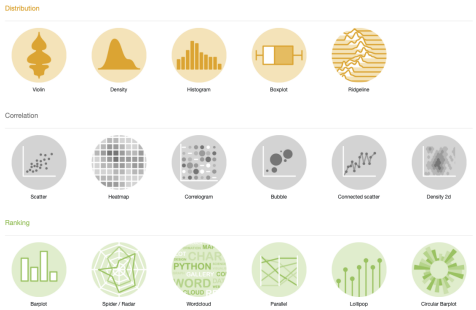
➡️ https://exts.ggplot2.tidyverse.org/gallery/


```
library(gganimate)
ggplot(mtcars, aes(factor(cyl), mpg)) +
  geom_boxplot() +
  # Here comes the gganimate code
  transition_states(
    gear,
    transition_length = 2,
    state_length = 1
  ) +
  enter_fade() +
  exit_shrink() +
  ease_aes('sine-in-out')
```

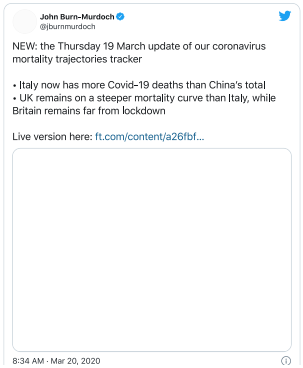




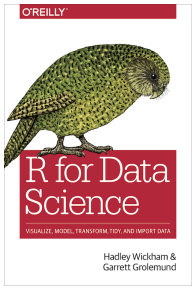
The R Graph Gallery



To be continued ...



Reading



- Data visualisation
- {ggplot2} cheatsheet