Data Visualisation

Data Visualisation

> Du skal tegne før du må regne / “You must illustrate before you calculate”
> — George Rasch or Thorvald Nicolai Thiele

> A picture is worth a thousand words.
> — Henrik Ibsen

- Data visualisation communicates information much quicker than numerical tables
- Data visualisation can reveal unexpected structures in data (a key tools in exploratory data analysis)

  Making the correct visual summary of a dataset or the results of an analysis is the most powerful and conveying method to present your work to others.

Hidden structures

```
# A tibble: 142 × 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
 7  35.6  79.9
 8  33.1  77.6
 9  29.0  74.5
10  26.2  71.4
# ℹ 132 more rows
```

Hidden structures

Data from **Alberto Cairo**.

The Datasaurus Dozen

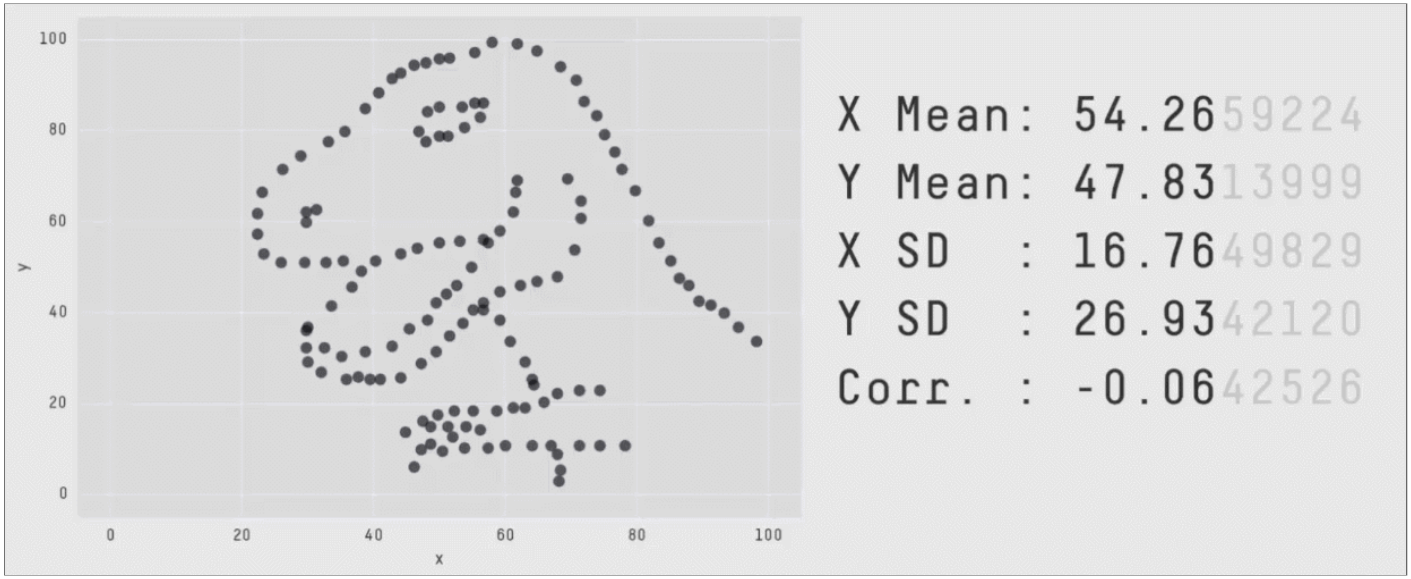

Image credit: Steph Locke, based on **datasauRus** / DOI:**10.1145/3025453.3025912**.

Programming graphics

An iterative process

The `ggplot2` package

- R has several ways to plot and visualise data and models. We typically group these in two overall categories - **base R** and **ggplot**.
- The former is the built-in feature in R that historically made R a favorite tool for visualisation compared to other contemporary statistic software (e.g. Excel, SAS, SPSS, etc.).
- However, with the introduction of the `ggplot2` package in 2007 the features and functionalities in R's visual toolbox has increased dramatically.
- `ggplot2` implements what is referred to as "The Grammar of Graphics". Here each feature in a plot is controlled by a variable in the data.

A graphing template for layered plots

```
library(ggplot2)
```

`ggplot2` works by adding layers to a canvas, where the canvas is defined by the columns in the data. By using **aesthetics** we may control the appearance of the data.

```
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` (more later!)
2. `mapping`: **aes**thetic 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 (points, bars, lines, …).
   - `stat`: statistical transformations applied to the data prior to plotting.
   - `position`: position adjustment, such as “identity”, “stack”, “dodge” etc.

A graphing template for layered plots

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

Fuel economy data (**m**iles **p**er **g**allon) from `ggplot2`:

```
mpg
```

```
# A tibble: 234 × 11
   manufacturer model  displ  year   cyl
   <chr>        <chr>  <dbl> <int> <int>
 1 audi         a4       1.8  1999     4
 2 audi         a4       1.8  1999     4
 3 audi         a4       2    2008     4
 4 audi         a4       2    2008     4
 5 audi         a4       2.8  1999     6
 6 audi         a4       2.8  1999     6
 7 audi         a4       3.1  2008     6
 8 audi         a4 qu…   1.8  1999     4
 9 audi         a4 qu…   1.8  1999     4
10 audi         a4 qu…   2    2008     4
# ℹ 224 more rows
# ℹ 6 more variables: trans <chr>,
#   drv <chr>, cty <int>, hwy <int>,
#   fl <chr>, class <chr>
```

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  layer(geom = "point", stat = "identity", position = "identity")
```

`layer()` short-hands:

- `geom_point()` shorthand for
- `layer(geom = "point", stat = "identity", position = "identity")`

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  layer(geom = "point", stat = "identity", position = "identity")
```

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point()
```

- Often we will use the pipe to start with data and send it to ggplot-function:

```
mpg |> ggplot(aes(displ, hwy)) +
  geom_point()
```

Examples

```
mpg |> ggplot(aes(x = displ, y = hwy))
mpg |> ggplot(aes(x = displ, y = hwy)) +
  geom_point()
mpg |> ggplot(aes(x = displ, y = hwy)) +
  geom_point(aes(colour = class))
```

Inheriting mappings and data

- We have a structure where the data is defined and features of the plot is controlled by adding *layers* to the plot

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))

ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +
  <GEOM_FUNCTION>()
```

- Note, if an aesthetic is set by the `mapping` at the `ggplot`-level it will be inherited by *all* the subsequent layers (globally). Similarly for `data`.

```
mpg |> ggplot(aes(x = displ, y = hwy, colour = class)) +
  geom_point() +
  geom_smooth()
```

- However, if specified at a `GEOM_FUNCTION` it will only affect that `geom` (locally).

```
mpg |> ggplot(aes(x = displ, y = hwy)) +
  geom_point(aes(colour = class)) +
  geom_smooth()
```

Controllable aesthetics

Which features/aesthetics that can be controlled depends on the type of plot, e.g.:

- scatterplot
  - colour
  - size
  - shape
  - alpha
  - *fill* (only for specific plot characters)
  - *stroke* (only for specific plot characters)
- boxplot
  - fill (of the boxes)
  - color (outline colour)
  - …
- barplot
  - fill (of the boxes)
  - color (outline colour)
  - …
- histogram
  - fill (of the boxes)
  - color (outline colour)
  - …
- maps

- heatmaps
- densities
- …

  The help packages for each `geom`, e.g. `?geom_point` tell which elements that can be controlled.

Facets

Another powerful feature of `ggplot2` is the ease of making facet plots (subplots where each panel/plot is conditioned on other variables)

```
mpg_plot <- mpg |> ggplot(aes(x = displ, y = hwy)) +
  geom_point(aes(colour = class)) +
  geom_smooth(method = "lm", formula = y ~ x)

mpg_plot + facet_wrap(~manufacturer)
```

```
mpg_plot + facet_wrap(~manufacturer, scales = "free_x")
```

```
mpg_plot + facet_wrap(~manufacturer, scales = "free")
```

One can facet on several variables at the same time

```
## Construct a new variable from `trans` to determine Auto or Manual transmisson:
## NB! When making changes to the data we need to "revoke" ggplot once again for it to work!
mpg |>
  mutate(transmission = ifelse(str_detect(trans, "auto"), "Auto", "Manual")) |>
  ggplot(aes(x = displ, y = hwy)) +
  geom_point(aes(colour = class)) +
  geom_smooth(method = "lm", formula = y ~ x) +
  facet_grid(transmission ~ cyl, labeller = label_both)
```

Position adjustment

When plotting *histograms*, *barplots* and *boxplots* it is important to be able to control the position of the graphical elements. There is a "family" of `position`-functions that can be used to make finer control and adjustment of the position:

- `position_dodge`
- `position_dodge2`
- `position_fill`
- `position_identity`
- `position_jitter`
- `position_jitterdodge`
- `position_nudge`
- `position_stack`

The online help pages has many examples: https://ggplot2.tidyverse.org/reference/position_dodge.html

Example: Barplot

```
mpg2 <- mpg |> mutate(transmission = ifelse(str_detect(trans, "auto"), "Auto", "Manual"))
```

- What is wrong with the fill color here?

```
mpg2 |>
  ggplot(aes(x = transmission, fill = cyl)) +
  geom_bar()
```

```
mpg2
```

```
# A tibble: 234 × 12
   manufacturer model    displ  year   cyl
   <chr>        <chr>    <dbl> <int> <int>
 1 audi         a4         1.8  1999     4
 2 audi         a4         1.8  1999     4
 3 audi         a4         2    2008     4
 4 audi         a4         2    2008     4
 5 audi         a4         2.8  1999     6
 6 audi         a4         2.8  1999     6
 7 audi         a4         3.1  2008     6
 8 audi         a4 quat…   1.8  1999     4
 9 audi         a4 quat…   1.8  1999     4
10 audi         a4 quat…   2    2008     4
# ℹ 224 more rows
# ℹ 7 more variables: trans <chr>,
#   drv <chr>, cty <int>, hwy <int>,
#   fl <chr>, class <chr>,
#   transmission <chr>
```

Example: Barplot

`cyl` was numeric

```
mpg2 |> ggplot(aes(x = transmission, fill = factor(cyl))) + geom_bar()
```

```
mpg2 |> ggplot(aes(x = transmission, fill = factor(cyl))) +
  geom_bar() + labs(fill = "Cylinders")
```

Example: Barplot

Not stacked - side-by-side

```
mpg2 |> ggplot(aes(x = transmission, fill = factor(cyl))) +
  geom_bar(position = position_dodge()) + labs(fill = "Cylinders")
```

Example: Barplot

Relative distribution within *each* group

```
mpg2 |> ggplot(aes(x = transmission, fill = factor(cyl))) +
  geom_bar(position = position_fill()) +
  labs(y = "fraction", fill = "Cylinders")
```

Difference between `geom_bar` and `geom_col`

Barcharts/plots are often used to present summarised data by different combinations of qualitative variables.
In `ggplot2` there are two geoms that do almost the same: `geom_bar` and `geom_col`. The former performs a statistical calculation on the data before plotting - the latter doesn't.
Hence, if we already have aggregated data we use `geom_col` and otherwise `geom_bar`.

Themes

In `ggplot2` there are a number of *themes* that controls the overall appearance of the plot. There are a number of themes that comes with `ggplot2`

- `theme_bw`
- `theme_classic`
- `theme_dark`
- `theme_gray` /`theme_grey`
- `theme_light`
- `theme_linedraw`
- `theme_minimal`
- `theme_void`

The `ggthemes` provides even more:

- `theme_base`
- `theme_calc`
- `theme_clean`
- `theme_economist`
- `theme_economist_white`
- `theme_excel`
- `theme_excel_new`
- `theme_few`
- `theme_fivethirtyeight`
- `theme_foundation`

- `theme_gdocs`
- `theme_hc`
- `theme_igray`
- `theme_map`
- `theme_pander`
- `theme_par`
- `theme_solarized`
- `theme_solarized_2`
- `theme_solid`
- `theme_stata`
- `theme_tufte`
- `theme_wsj`

  The function `theme_set` can be used to set the theme globally in a session:

```
theme_set(theme_bw(base_size = 20))
```

Use `theme` for finer control

The `theme`-function enables an even finer control of the look.
We typically use the `theme` to control the position of e.g. the legend.

```
mpg2 |> ggplot(aes(x = displ, y = hwy, colour = transmission)) +
  geom_point() + geom_smooth(method = "lm", formula = y ~ x) +
  theme(legend.position = "top")
```

Typical error

NB! Remember to use `theme` **before** adding a change to the theme

```
mpg2 |> ggplot(aes(x = displ, y = hwy, colour = transmission)) +
  geom_point() + geom_smooth(method = "lm", formula = y ~ x) +
  theme(legend.position = "top") + theme_excel()
```

```
mpg2 |> ggplot(aes(x = displ, y = hwy, colour = transmission)) +
  geom_point() + geom_smooth(method = "lm", formula = y ~ x) +
  theme_excel() + theme(legend.position = "top")
```

Vast list of resources

Stack overflow: https://stackoverflow.com/questions/tagged/ggplot2 (Over 50k questions)
Types of geoms: https://ggplot2.tidyverse.org/reference/index.html
Gallery: https://exts.ggplot2.tidyverse.org/gallery/
Cheatsheet: https://github.com/rstudio/cheatsheets/blob/main/data-visualization-2.1.pdf

Exercises

Try to have a go with some of the many exercises in Chapter 1 (also available in qmd file on Moodle):

- Exercises 1.2.5: https://r4ds.hadley.nz/data-visualize#exercises
- Exercises 1.4.3: https://r4ds.hadley.nz/data-visualize#exercises-1
- Exercises 1.5.5: https://r4ds.hadley.nz/data-visualize#exercises-2