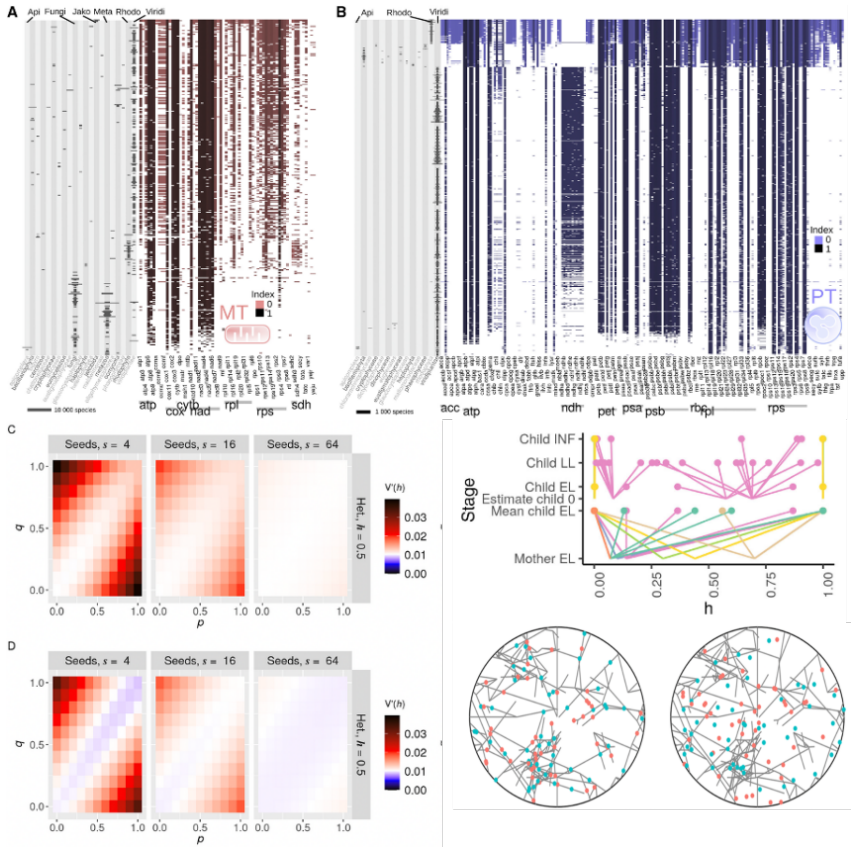
Visualisation with `ggplot2`: A primer

Iain Johnston

Department of Mathematics
University of Bergen

Hello!

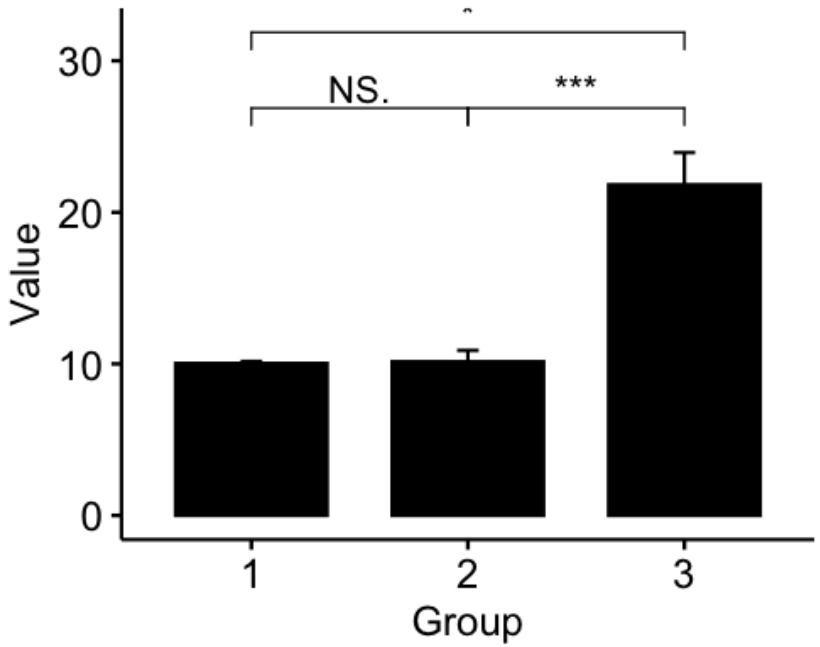
Disclaimer: I have no formal training in data visualization and everything here will be principles I work to and approaches I use, rather than a rigorous and complete syllabus on data visualization theory and practice. For that (and ECTS credits) look to `https://vis.uib.no/teaching/`.



Why do we produce plots?

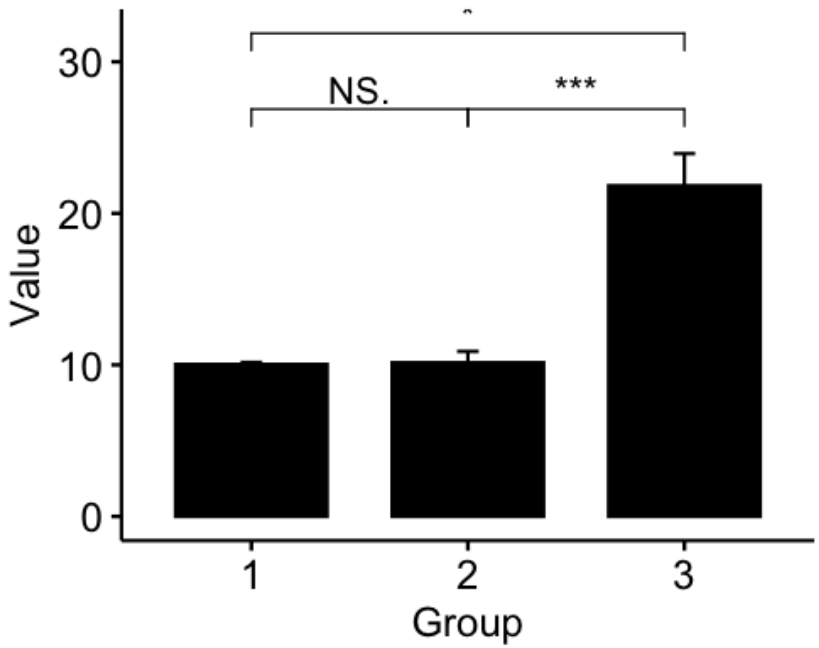
Why do we produce plots?

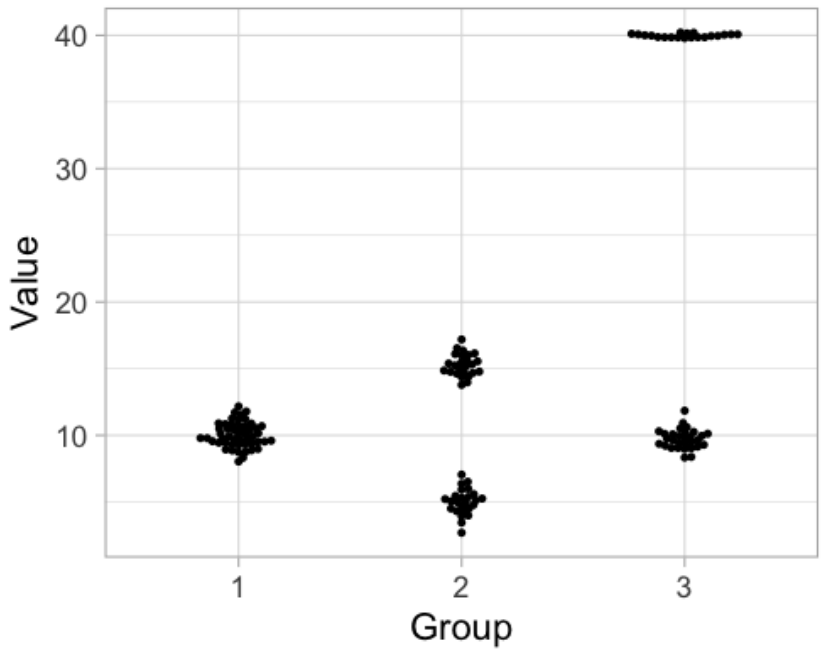
- To summarise and convey data
- To turn data into information
- To tell a story



Why do we produce plots?

- To summarise and convey data
- To turn data into information
- To tell a story
- ... clearly, transparently, honestly, reproducibly, beautifully, pre-empting questions





Why do we produce plots?

- 'Friends don't let friends' (Chenxin Li) `https://github.com/cxli233/FriendsDontLetFriends`
- Complete ggplot tutorial `http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html`

Theory of mind!

- **Motivation:** A reader will, at first, only spend a couple of seconds glancing at your figure.

Theory of mind!

- **Motivation:** A reader will, at first, only spend a couple of seconds glancing at your figure.
- That's whether they're skimming an article, looking around at posters, or looking at your slide while they're watching you talk

Theory of mind!

- **Motivation:** A reader will, at first, only spend a couple of seconds glancing at your figure.
- That's whether they're skimming an article, looking around at posters, or looking at your slide while they're watching you talk
- Even if they spend longer, the easier the messages come through, the better

Theory of mind!

- **Motivation:** A reader will, at first, only spend a couple of seconds glancing at your figure.
- That's whether they're skimming an article, looking around at posters, or looking at your slide while they're watching you talk
- Even if they spend longer, the easier the messages come through, the better
- **Therefore:** The big picture should be clear and get the most important message, and as much transparent support as you can provide, across at a glance

Theory of mind!

- **Motivation:** A reader will, at first, only spend a couple of seconds glancing at your figure.
- That's whether they're skimming an article, looking around at posters, or looking at your slide while they're watching you talk
- Even if they spend longer, the easier the messages come through, the better
- **Therefore:** The big picture should be clear and get the most important message, and as much transparent support as you can provide, across at a glance
- It should also look cool! Especially if your reader's attention is tempted elsewhere (it is)

Theory of mind!

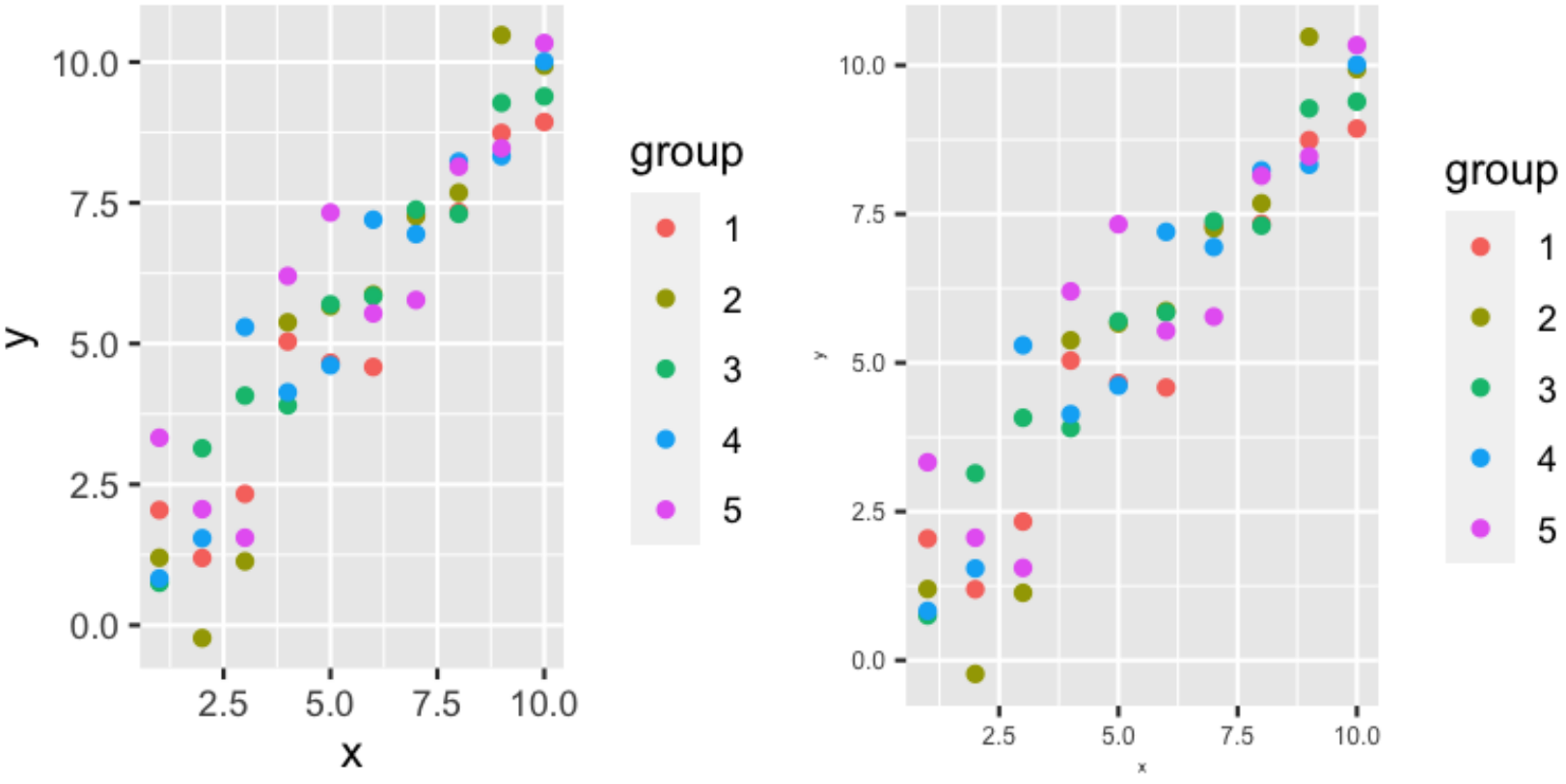
- **Motivation:** A reader will, at first, only spend a couple of seconds glancing at your figure.
- That's whether they're skimming an article, looking around at posters, or looking at your slide while they're watching you talk
- Even if they spend longer, the easier the messages come through, the better
- **Therefore:** The big picture should be clear and get the most important message, and as much transparent support as you can provide, across at a glance
- It should also look cool! Especially if your reader's attention is tempted elsewhere (it is)
- A figure is not just an image! In written work, the caption is essential too (art museums). In oral work, your description plays this role.

Basic principles: aesthetic

- Match scales, colours, fonts (basically, consistent style)
- Have information occupy most of the plot
- Plot as close to individual datapoints as possible
- Every approximation (especially smoothing) needs explicit mention
- Define all degrees of freedom (only) in figure caption / oral commentary
- Consider accessibility

Basic principles: aesthetic

- Match scales, colours, fonts (basically, consistent style)



Basic principles: aesthetic

- Match scales, colours, fonts (basically, consistent style)

(Peppa goes to London)

Basic principles: aesthetic

- Match scales, colours, fonts (basically, consistent style)

(Peppa goes to London)

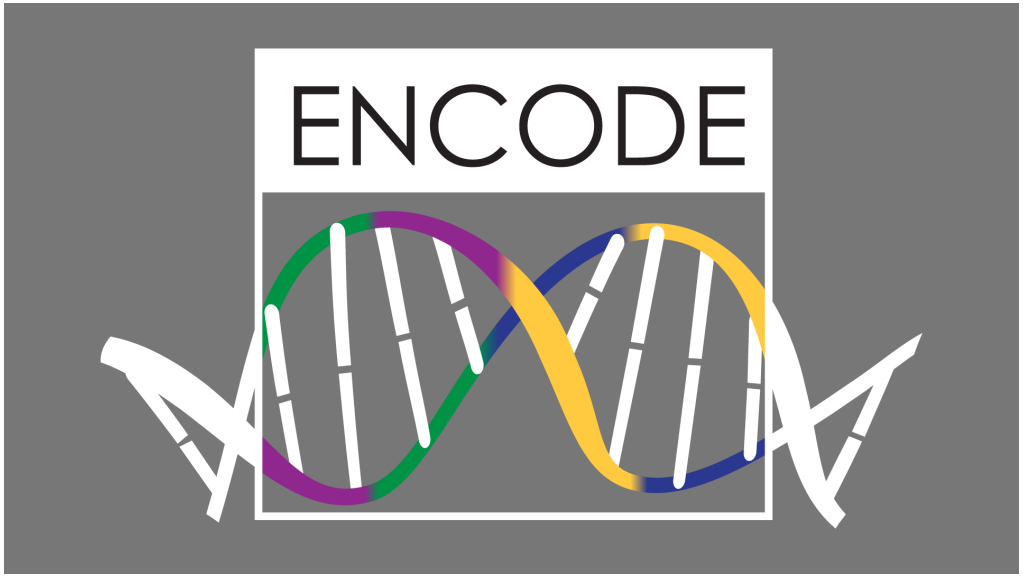
Basic principles: aesthetic

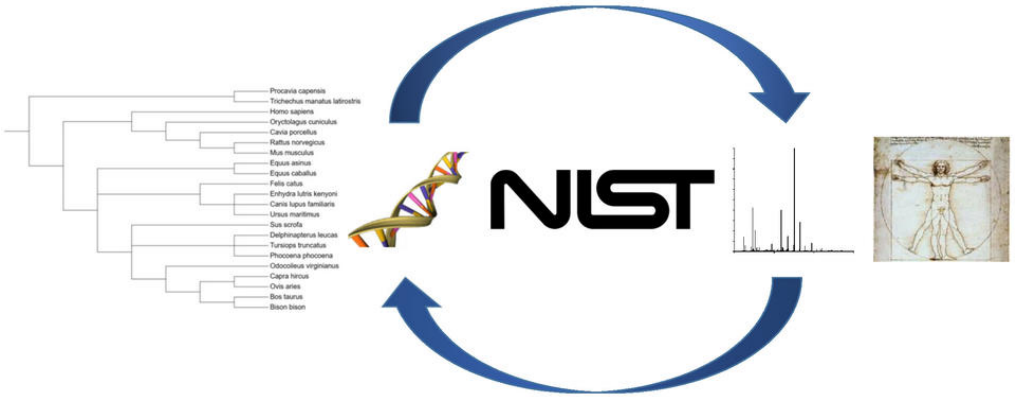
- Match scales, colours, fonts (basically, consistent style)

(Peppa goes to London)

Basic principles: aesthetic

- Match scales, colours, fonts (basically, consistent style)





(https://www.cshl.edu/encode3-interpreting-the-human-and-mouse-genomes/, https://www.nist.gov/image/serum-project-logo)

Arial
Helvetica
Myriad Pro
DejaVu Sans

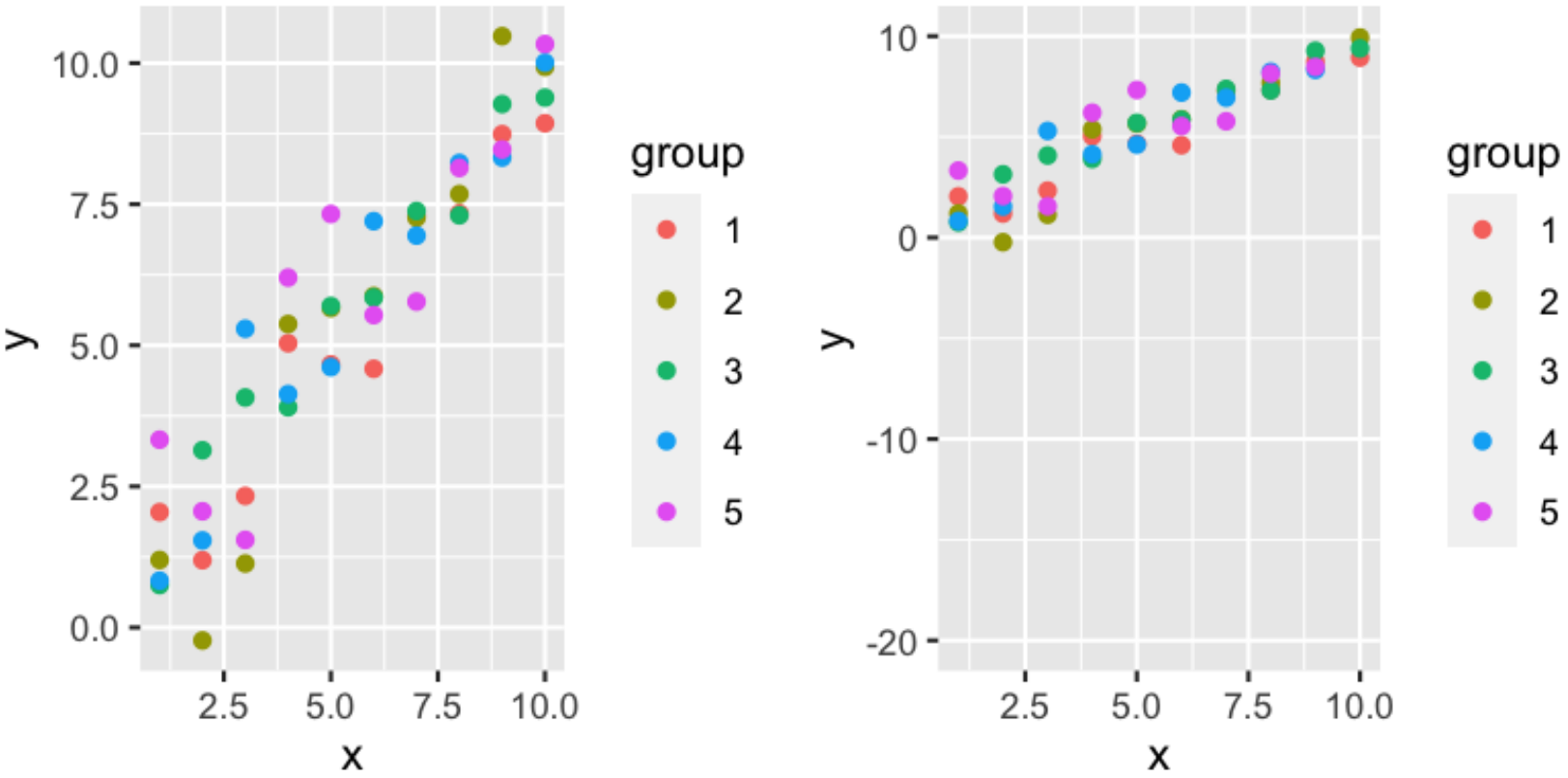
Basic principles: aesthetic

Arial
Helvetica
Myriad Pro
DejaVu Sans

- Fonts: just use Arial (or Helvetica, though it is slightly more embellished)
- Serifs are for guiding the eye through large blocks of text; they complicate text in small sections
- Myriad Pro (Illustrator default) looks like we're learning letters in primary school
- DejaVu Sans (matplotlib default) looks like the font designer is still learning letters (and kerning) in primary school

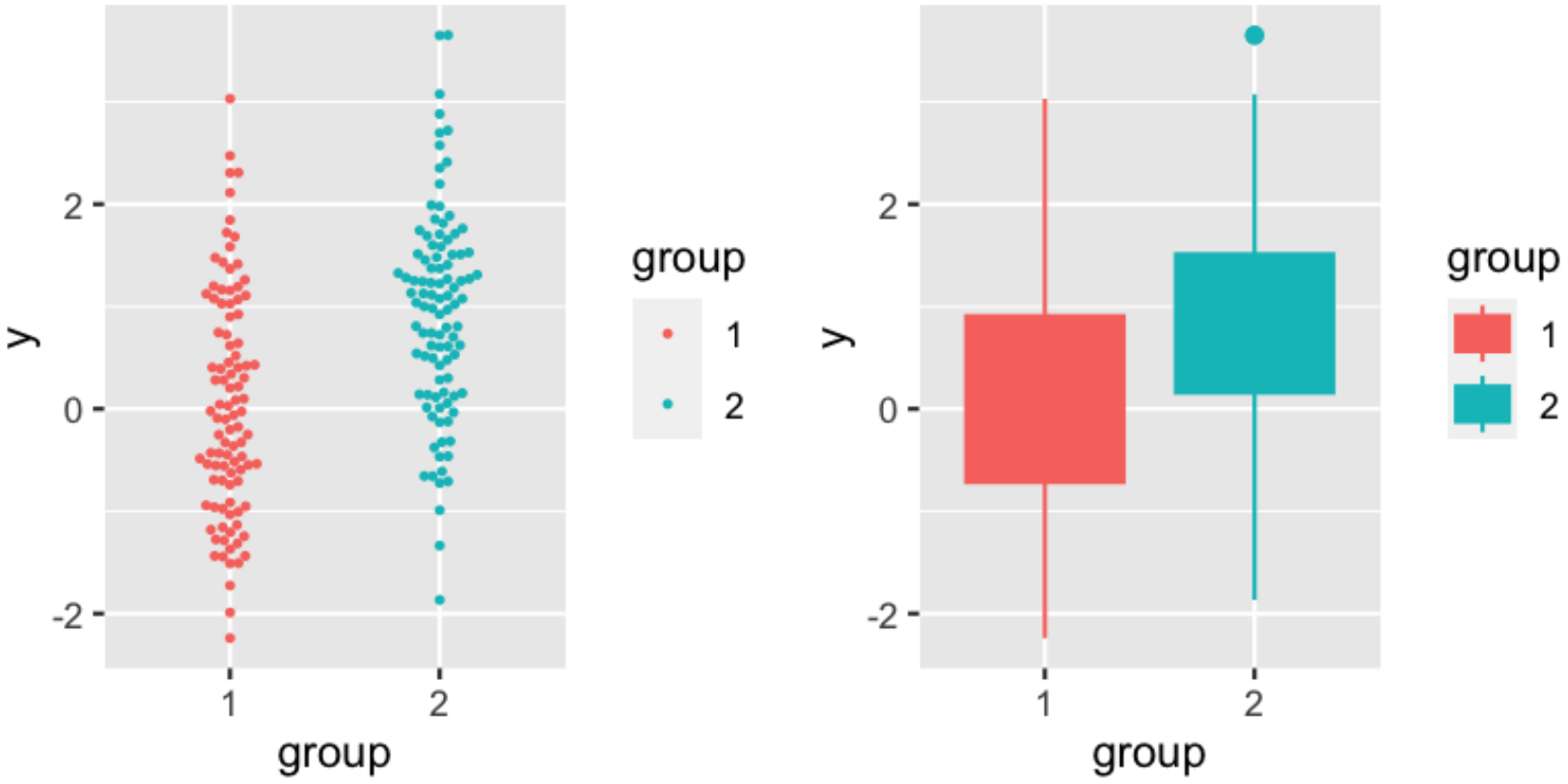
Basic principles: aesthetic

- Have information occupy most of the plot



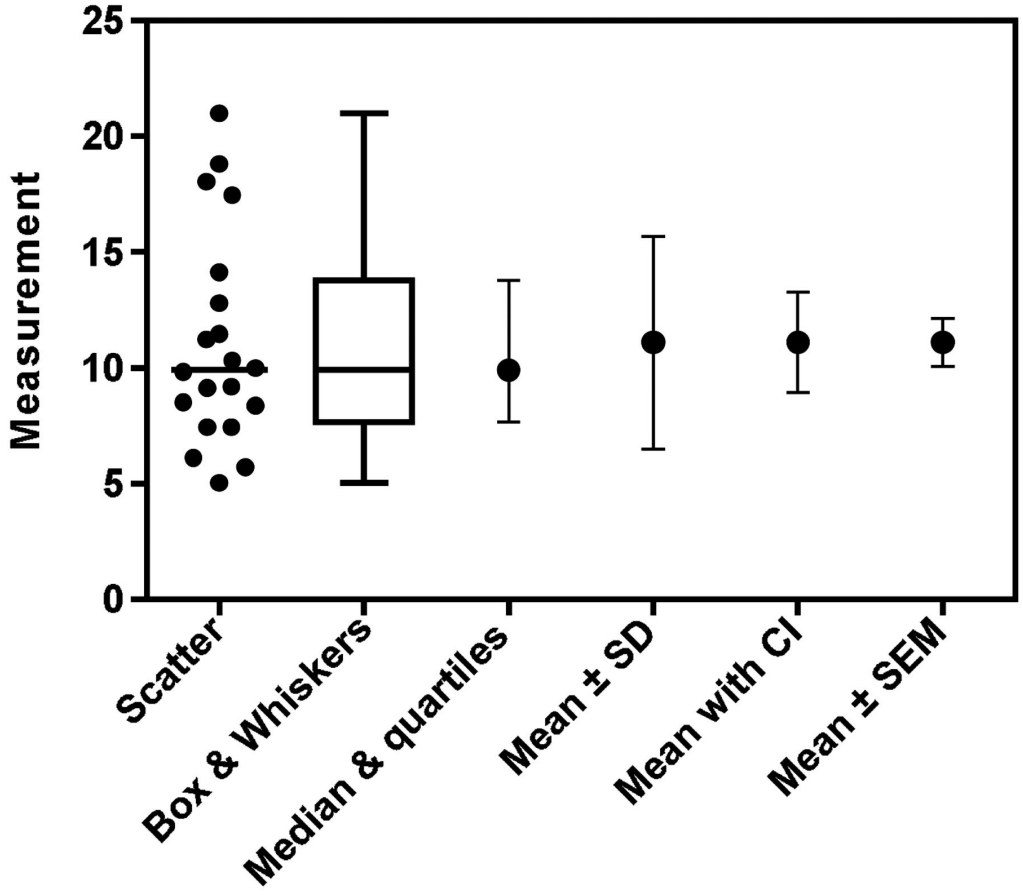
Basic principles: aesthetic

- Plot as close to individual datapoints as possible



Basic principles: aesthetic

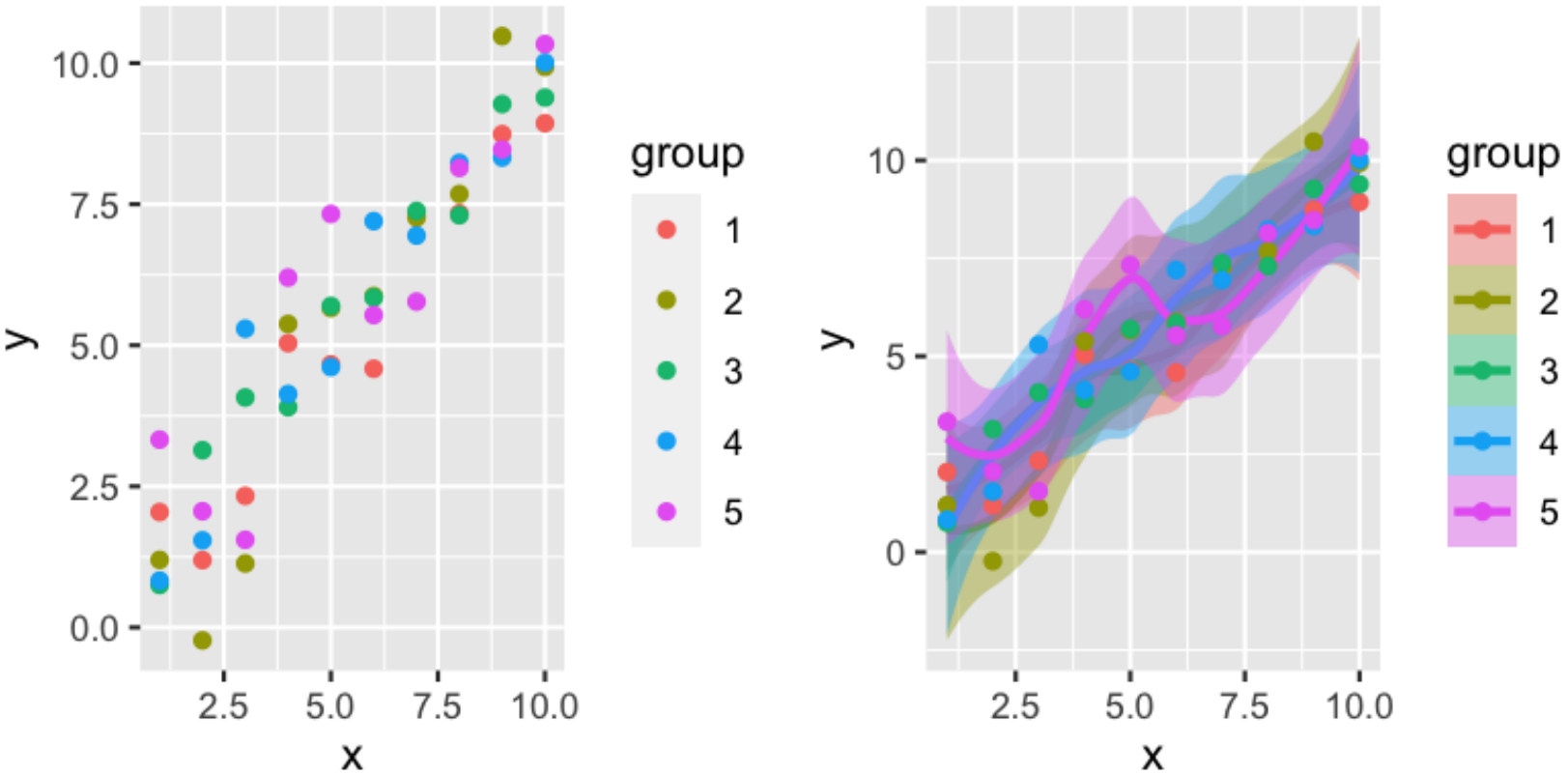
- Plot as close to individual datapoints as possible
- Showing more data is better; showing less data is worse
- If you want to summarise, why not do both?



[*Common Misconceptions about Data Analysis and Statistics*, Motulsky, J Pharm Expt Therap 351 200 (2014)]

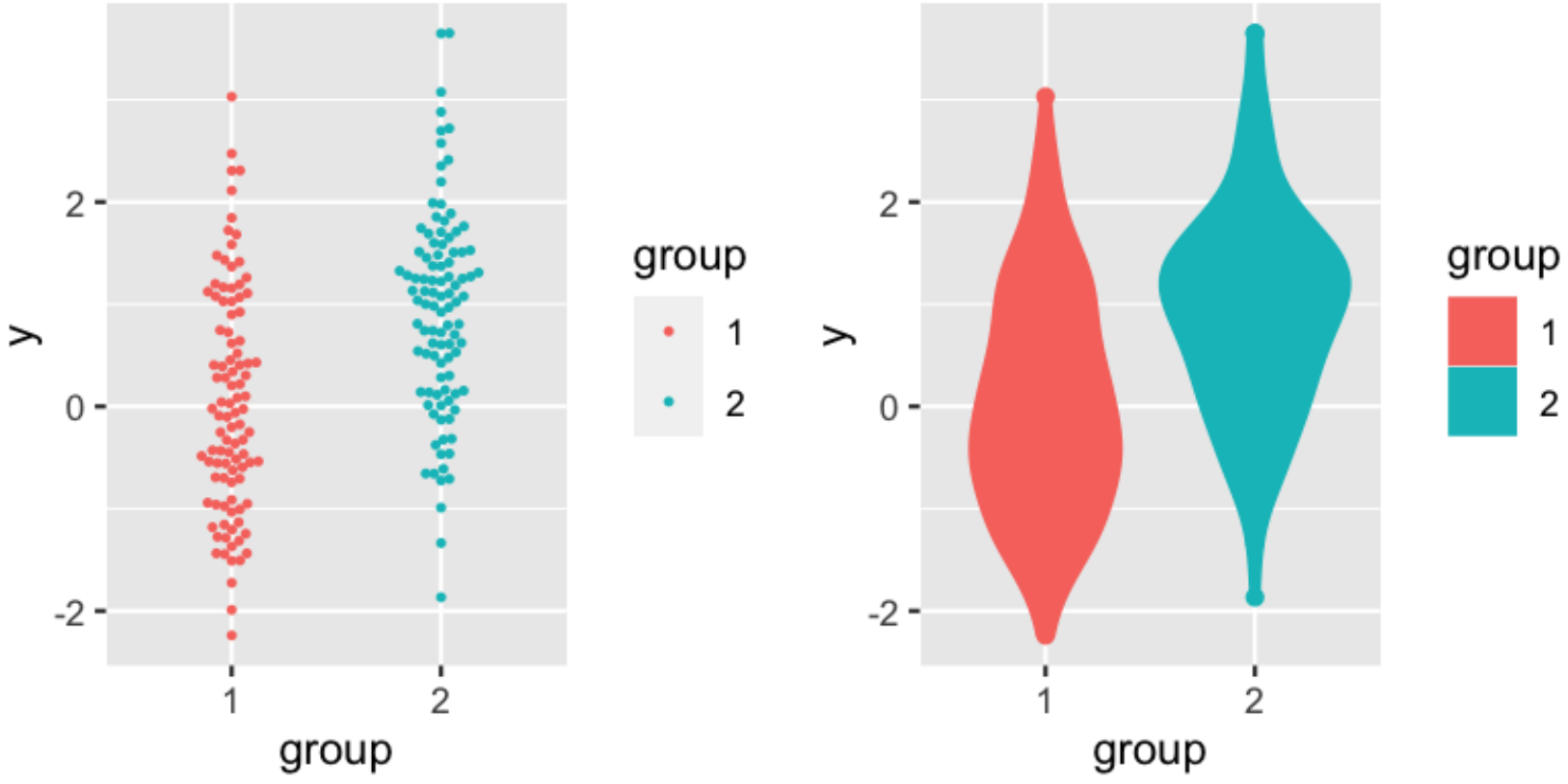
Basic principles: aesthetic

- Every approximation (especially smoothing) needs explicit mention. What parameters did you use? (Without knowing?)



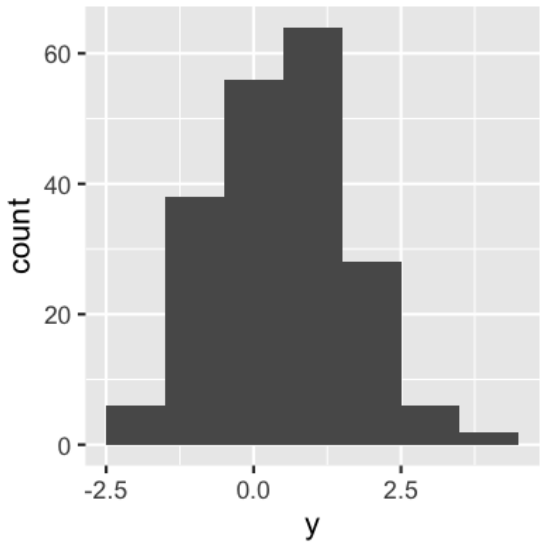
Basic principles: aesthetic

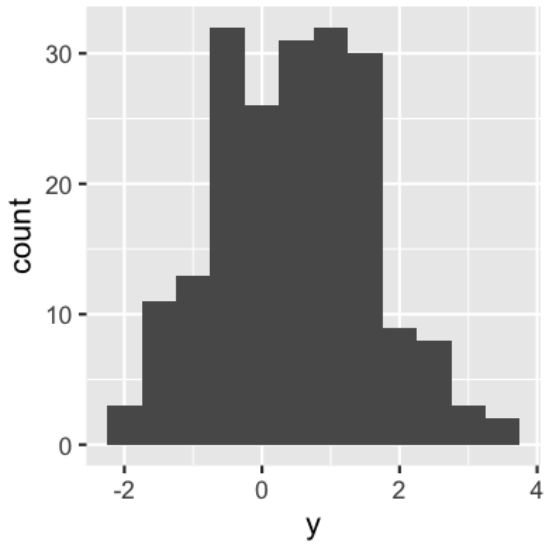
- Every approximation (especially smoothing) needs explicit mention. What parameters did you use? (Without knowing?)

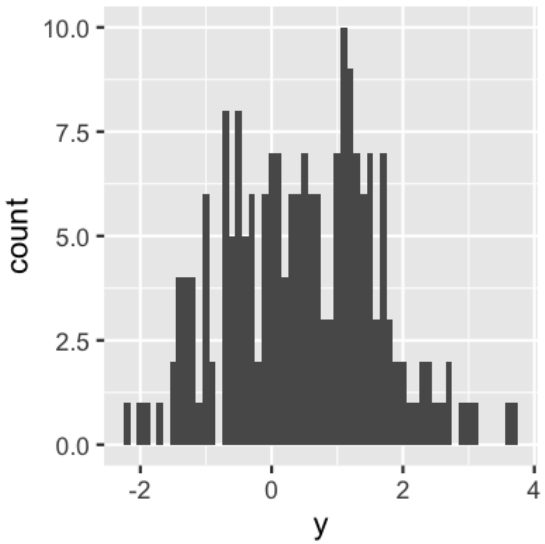


Basic principles: aesthetic

- Every approximation (especially smoothing) needs explicit mention. What parameters did you use? (Without knowing?)



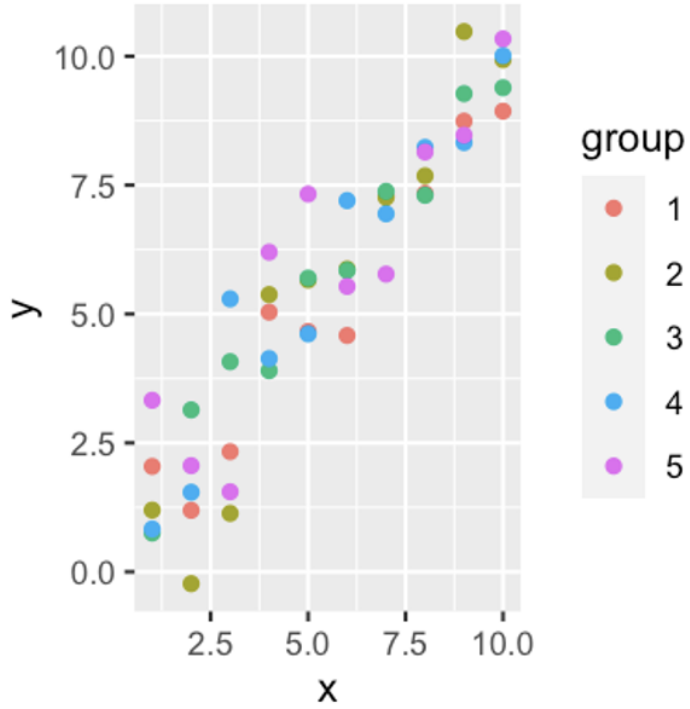




Basic principles: aesthetic

- Define all degrees of freedom (only) in figure caption / oral commentary

The red points in this figure are evenly spaced around the green and blue points, although they squash up around x=8.



Therefore there isn't much difference in the spreads of points.

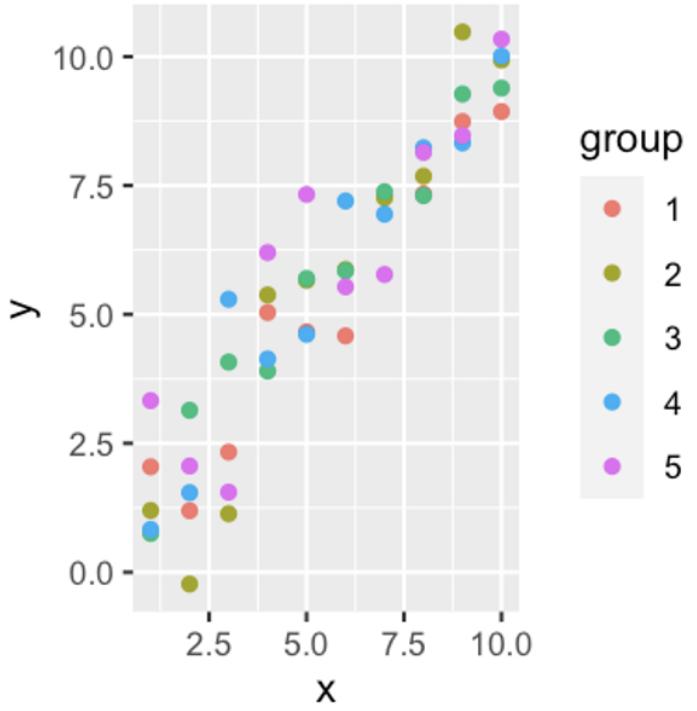


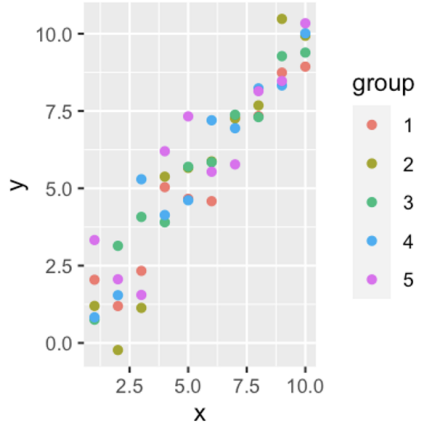
Figure 1. **x-y behaviour across different groups**. Groups are given by different colours. The red points are evenly spaced around the green and blue points, although they squash up around x=8.

Fig. 1 shows that there isn't much difference in the spreads of points.

Basic principles: aesthetic

- Define all degrees of freedom (only) in figure caption / oral commentary
- Information flow: the figure (image + caption) must give the message on its own – it's all that some people will read. That message can then be 'cited' in the main text

The red points in this figure are evenly spaced around the green and blue points, although they squash up around x=8.



Therefore there isn't much difference in the spreads of points.

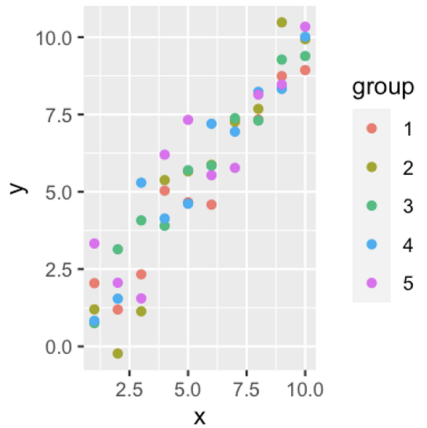
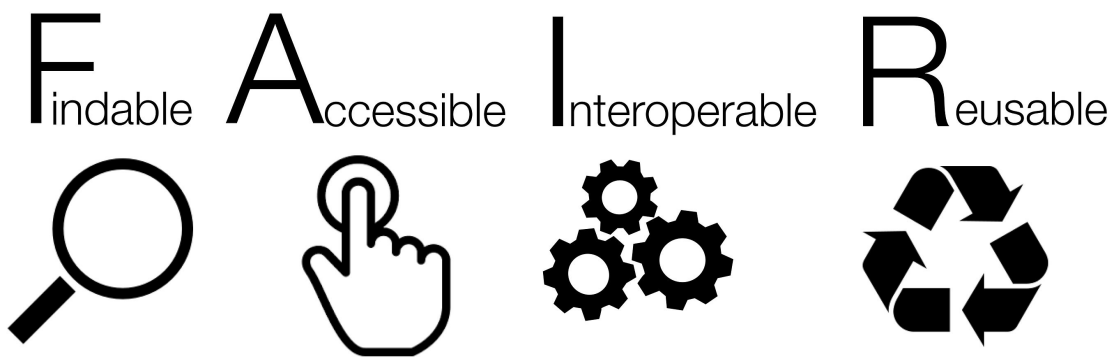


Figure 1. **x-y behaviour across different groups**. Groups are given by different colours. The red points are evenly spaced around the green and blue points, although they squash up around x=8.

Fig. 1 shows that there isn't much difference in the spreads of points.

Basic principles: production

- We'll use R and `ggplot2` (and additions)
- Part of the `tidyverse`
- Strongest for 2D plots; for 3D, consider looking elsewhere
- Aim to make all code and data for a project public, and to have a single script that produces all the display items for the paper
- FAIR reporting; but this also helps investigation!



Basic principles: production

- We need a file output for basically all purposes except live production

[1] JPG loses info; TIFF compression hard to implement in R

Basic principles: production

- We need a file output for basically all purposes except live production
- Bitmap/raster (JPG, TIFF, PNG, GIF, PSD) vs vector (PS, EPS, PDF, AI, SVG) formats

[1] JPG loses info; TIFF compression hard to implement in R

Basic principles: production

- We need a file output for basically all purposes except live production
- Bitmap/raster (JPG, TIFF, PNG, GIF, PSD) vs vector (PS, EPS, PDF, AI, SVG) formats
- I recommend PNG[1] and SVG(lite)

[1] JPG loses info; TIFF compression hard to implement in R

Basic principles: production

- We need a file output for basically all purposes except live production
- Bitmap/raster (JPG, TIFF, PNG, GIF, PSD) vs vector (PS, EPS, PDF, AI, SVG) formats
- I recommend PNG[1] and SVG(lite)
- Have your script get as close to published figure as possible (`ggpubr` for panels, annotation etc)

[1] JPG loses info; TIFF compression hard to implement in R

Basic principles: production

- We need a file output for basically all purposes except live production
- Bitmap/raster (JPG, TIFF, PNG, GIF, PSD) vs vector (PS, EPS, PDF, AI, SVG) formats
- I recommend PNG[1] and SVG(lite)
- Have your script get as close to published figure as possible (`ggpubr` for panels, annotation etc)
- One image file per figure

[1] JPG loses info; TIFF compression hard to implement in R

Basic principles: production

- We need a file output for basically all purposes except live production
- Bitmap/raster (JPG, TIFF, PNG, GIF, PSD) vs vector (PS, EPS, PDF, AI, SVG) formats
- I recommend PNG[1] and SVG(lite)
- Have your script get as close to published figure as possible (`ggpubr` for panels, annotation etc)
- One image file per figure
- For bitmaps, fix image size (in pixels), although journals will require 300dpi. 2490 pixels is A4 width at 300dpi

[1] JPG loses info; TIFF compression hard to implement in R

Basic principles: production

- My approach (for compatibility):

```
png("filename.png", width=400*sf,
height=300*sf, res=72*sf)
print( (plot command) )
dev.off()
```

- `png` sets output 'terminal' to PNG file
- 400×300 gives aspect ratio and scale of graphical elements; adjust to style output. Don't rescale in post-processing!
- `sf` scales resolution, *keeping those features intact*
- Also consider `ggsave`

Basic principles: input data for `ggplot2`

- What are the types of variables to plot? Continuous, discrete, factor, other? This determines plot types
- Long (not wide) data frames
- Every row is a datapoint; (possibly many) columns describe predictors and labels
- `reshape2` and `dplyr` offer functions to convert (pivot, melt, reshape)

Tree species	Site A	Site B	Site C	Site D
Acer rubrum	15	8	30	27
Quercus alba	29	17	14	42
Pinus taeda	10	19	25	23

Tree species	Site	DBH (cm)
Acer rubrum	A	15
Acer rubrum	B	8
Acer rubrum	C	30
Acer rubrum	D	27
Quercus alba	A	29
Quercus alba	B	17
Quercus alba	C	14
Quercus alba	D	42
Pinus taeda	A	10
Pinus taeda	B	19
Pinus taeda	C	25
Pinus taeda	D	23

(https://www.rforecology.com/post/reshaping-data-in-r/)

Basic principles: `ggplot2`

- 'Grammar of graphics' (Leland Wilkinson, 2005)
- Idea: build up graphics using system of elements (data, geometric objects, styling features, ...) combined using given rules
- Effectively: the final plot is an object which is built by adding the output of different functions
- Each function controls some elements of the grammar, specifying the details of our plot
- We'll look at a subset. Online searches are your (and my!) friend for completeness (remember to cite sources though!)

Basic principles: `ggplot2`

- Plot command + data + mapping/aesthetic + geoms + styling options
- `ggplot(data = my.df, mapping = aes(x=x, y=y)) + geom_point() + theme_light()`
- Implicit: `ggplot(my.df, aes(x=x, y=y)) + geom_point()+ theme_light()`
- Plot command + data hopefully intuitive
- mapping/aesthetic defines which columns in the data correspond to which aspects of the plot
- geoms define which types of plot we're producing and specify overall presentational aspects for the data
- styling options set presentational aspects additional to the data (axes, frame, fonts, etc)

Basic principles: ggplot2

- Let's try it!

```
library(ggplot2)
my.df = data.frame(x=1:10, y=1:10+rnorm(10))
ggplot(my.df, aes(x=x, y=y)) + geom_point()+
theme_light()
```

Data styling

- Geoms have a set of 'attributes' which may be connected to data (via `aes`) or specified otherwise

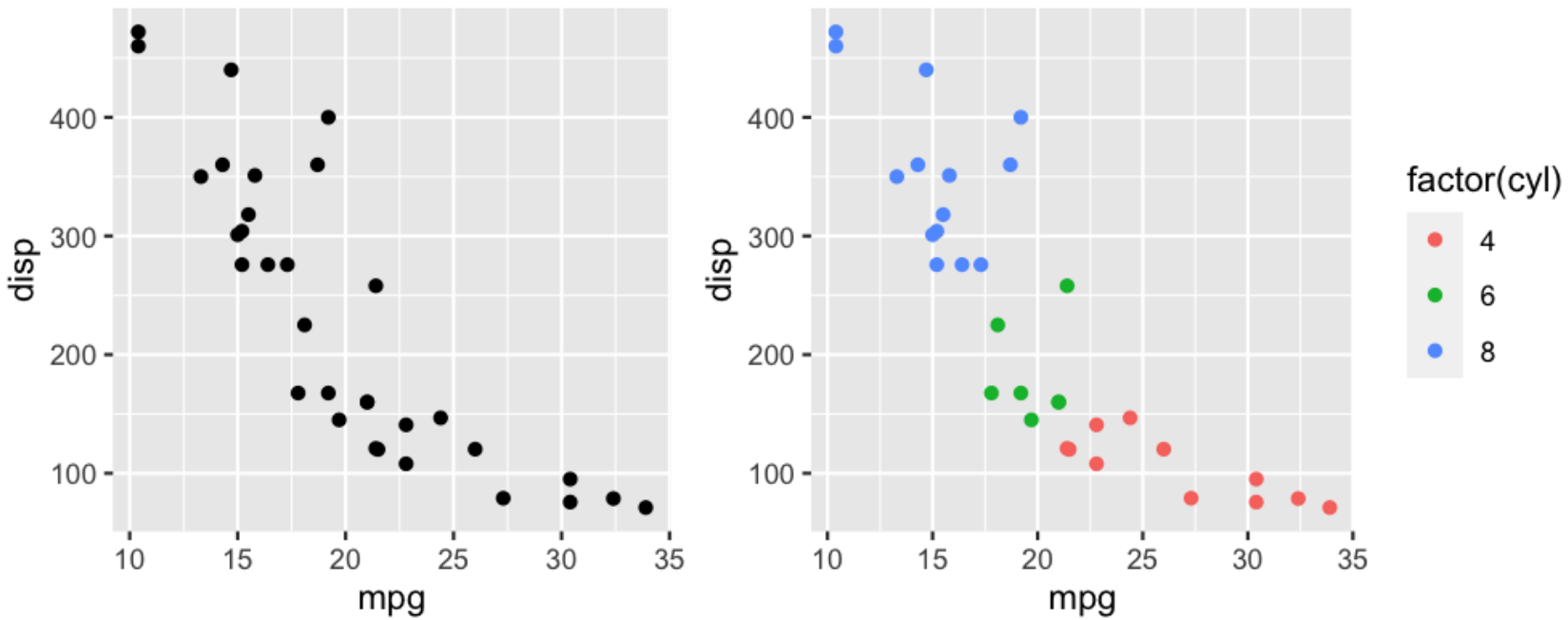
  e.g. `geom_point(aes(x=x,y=y,colour=group), size=2, alpha=0.5)`
- Common ones:
  - `size`
  - `fill`
  - `colour`
  - `alpha`
  - `shape`
- Size is relative (to dimensions of plot) – this will change for different output size/resolution combinations!
- Colour can be specified in several ways: some names (`red`, `green` etc), `#RRGGBB`, `#RRGGBBAA`

Exercise

- Continuous-continuous data

  `data(mtcars)`

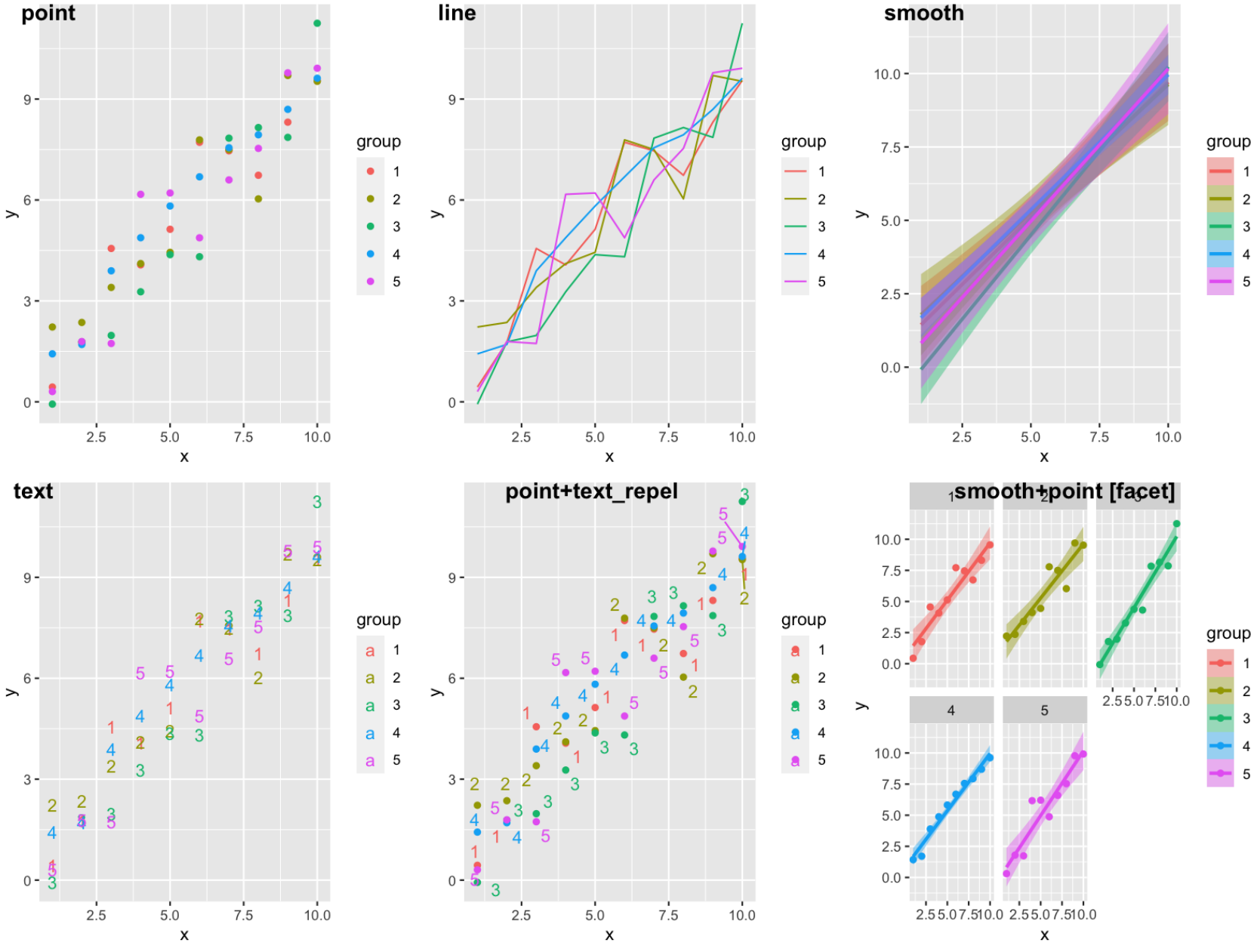
- `mtcars` has data on several car models. `mpg` (miles per gallon), `disp` displacement, `cyl` number of cylinders
- Graphically explore the relationship between these. Can you reproduce these plots?



`geom_...` for continuous-continuous

- `point`: points
- `line`: lines (connecting points by *x*)
- `path`: path (connecting point by order in data)
- `segment`: line segments (specifying start and end points `xend`, `yend`)
- `smooth`: smoothed lines, including regression (`method=''lm''`)
- `tile`: filled tiles (specifying fill colour `fill`); also `rect`
- `text`, `label`, `text_repel`*: text (specifying text `label`) with and without box, and avoiding overlaps

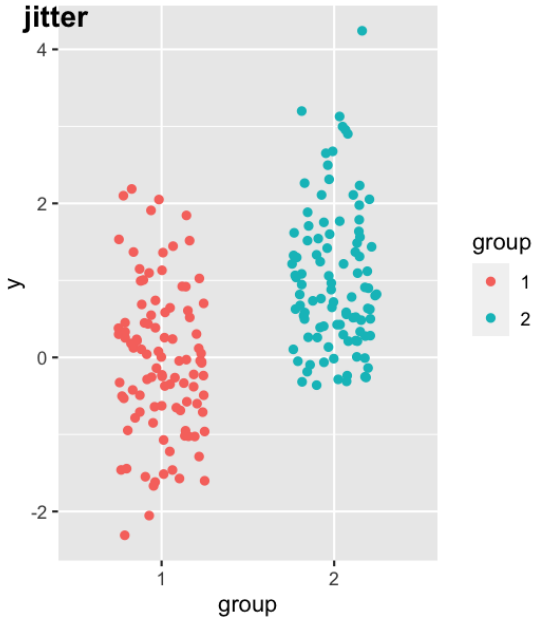
`geom_...` for continuous-continuous

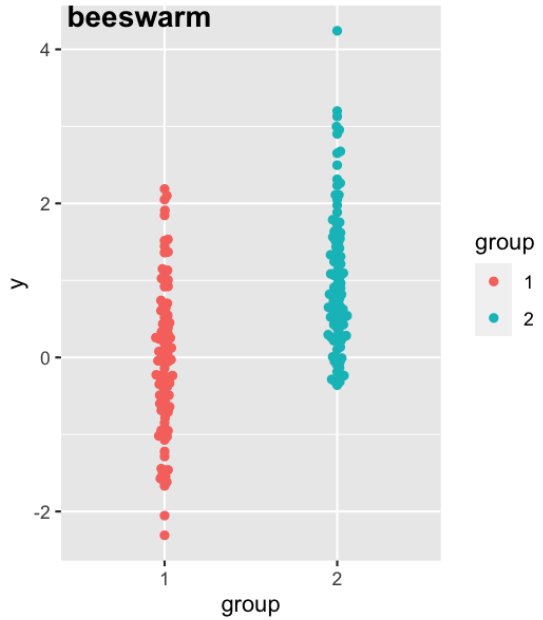


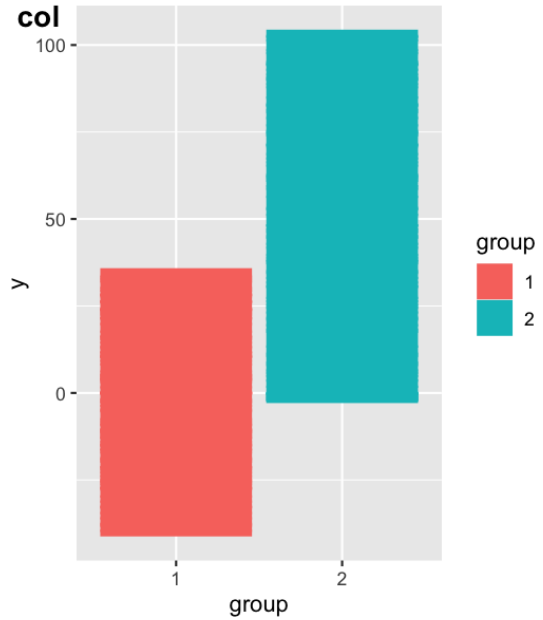
`geom_...` for factor-continuous

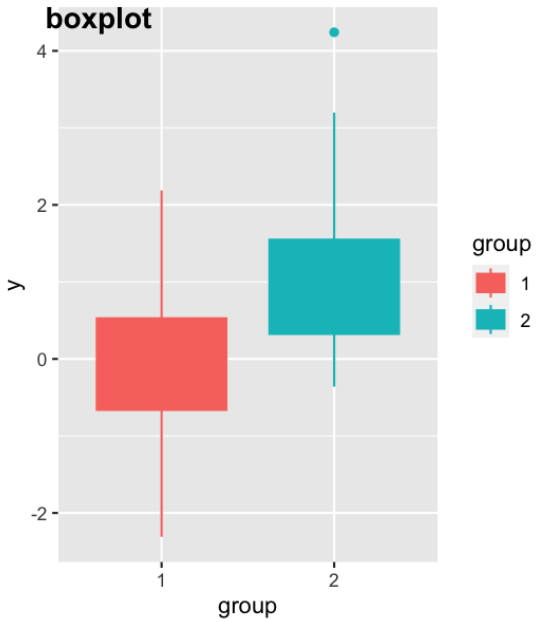
- `point`: points
- `col`: columns
- `jitter` and `beeswarm`*: points arranged to avoid overlaps
- `boxplot`: box plots
- `violin`: violin plots
- `histogram`: histogram (consider `bins`)

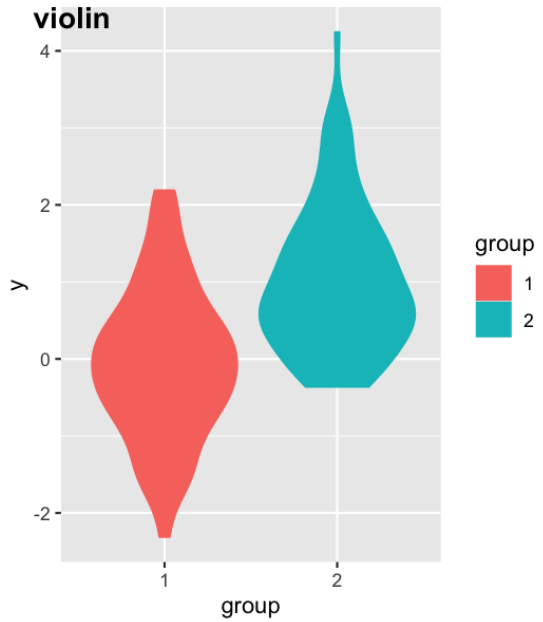
geom_... for factor-continuous









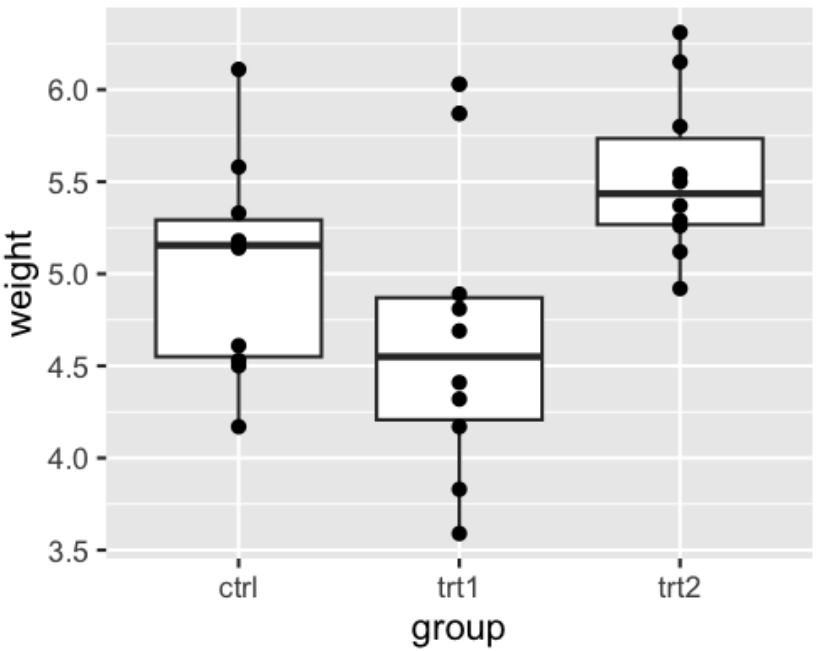


Exercise

- Factor-continuous data

  `data(PlantGrowth)`

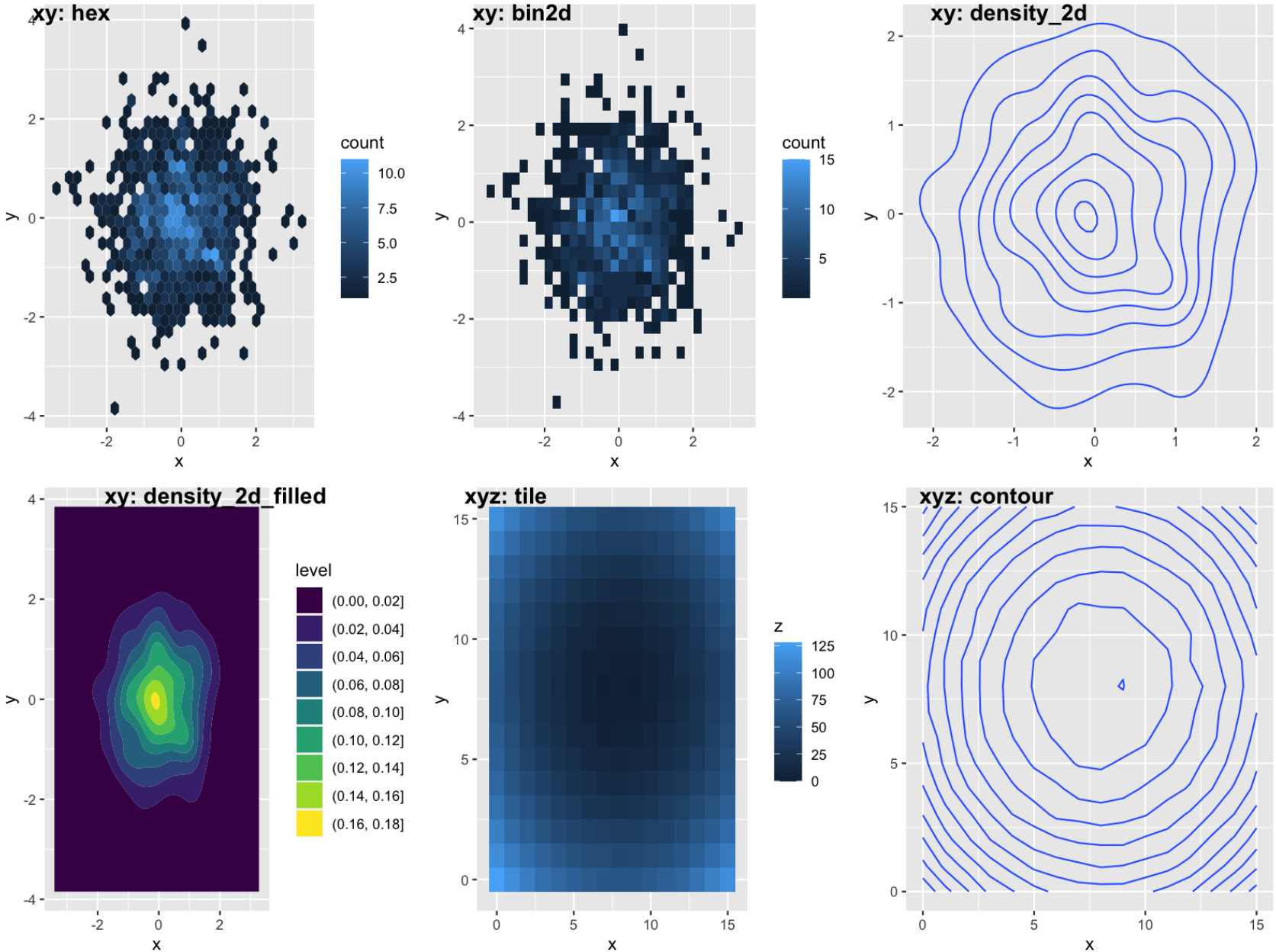
- `PlantGrowth` has data on plant yield under different treatments (and control). `weight` is yield (dry weight), `group` is treatment
- Graphically explore the relationship between these. Can you reproduce this plot?



`geom_...` for continuous-continuous-continuous

- Takes x, y scatter and works with point density:
  - `bin_2d`: filled rectangles
  - `hex`: filled hexagons
  - `geom_density_2d`, `geom_density_2d_filled`: contours (after smoothing)
- Takes x, y, z data (can be produced from above by smoothing):
  - `tile` and `rect` as before
  - `contour`, `label_contour`*: contour plots

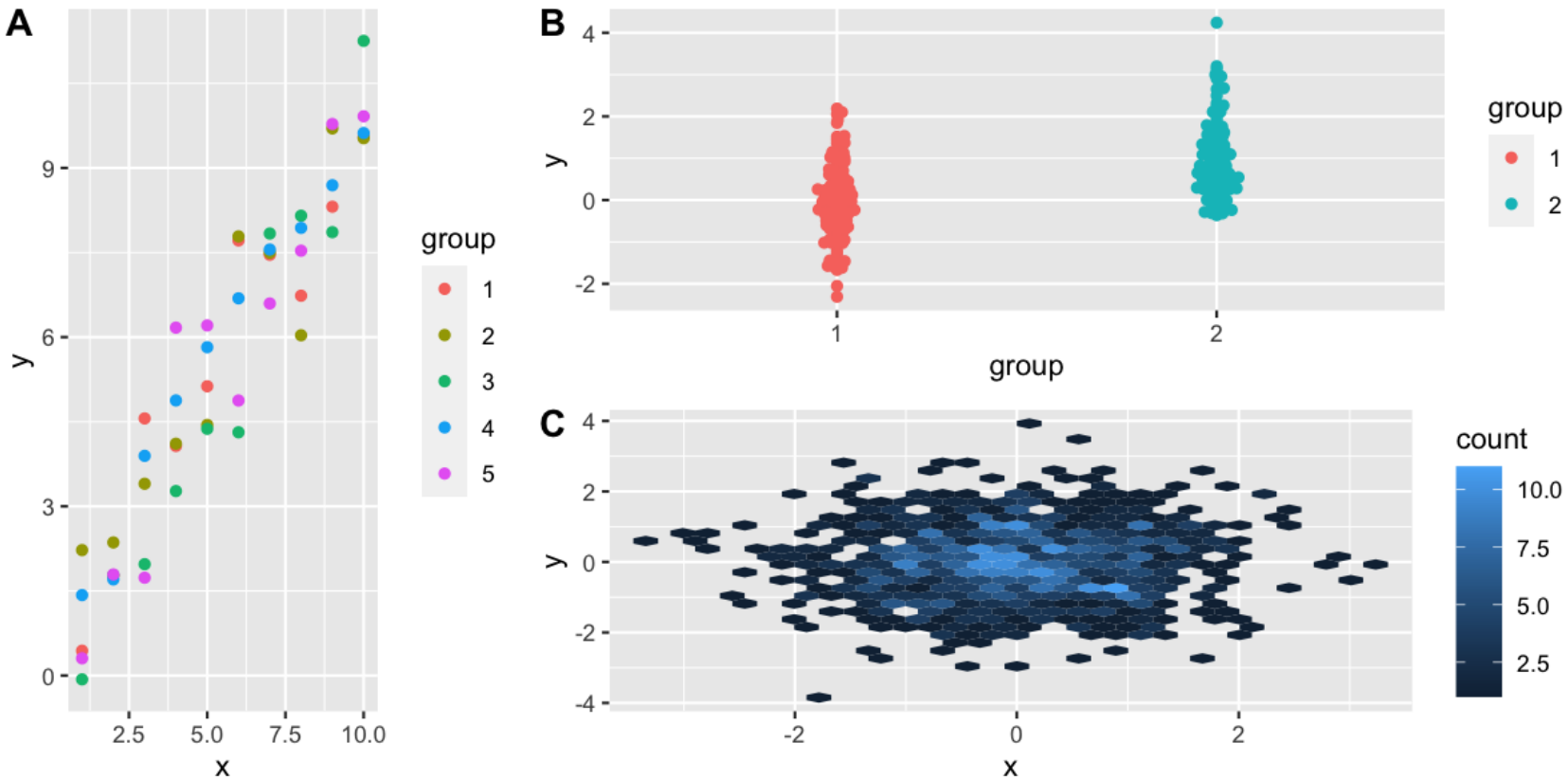
`geom_...` for continuous-continuous-continuous



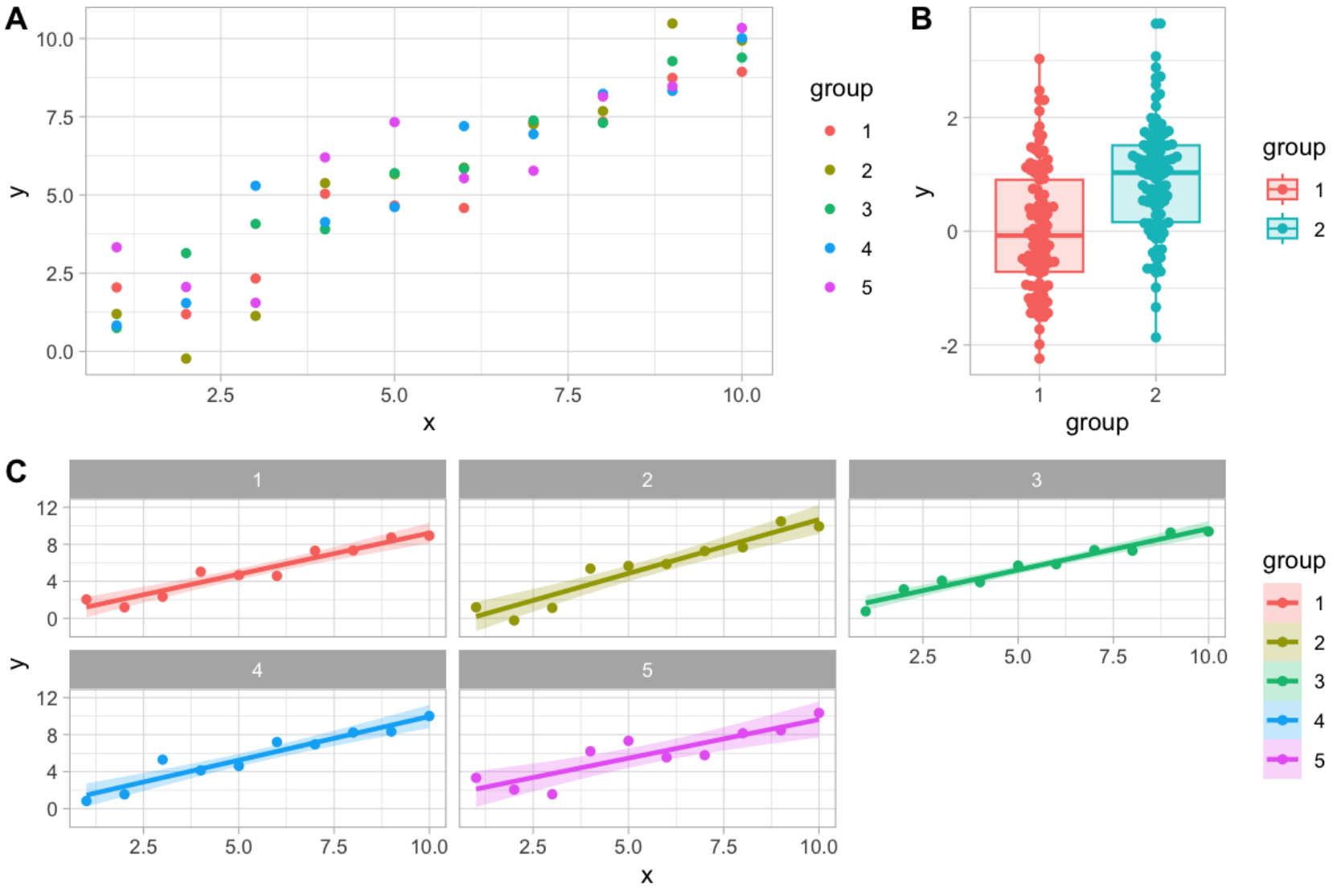
Plot theme styling: arrangements

- Organise a set of plots in 1D: `facet_wrap(~ variable)`
- Organise a set of plots in 2D: `facet_grid(var1 ~ var2)`
- Organise generally: `ggarrange(g.1, g.2, ...)`
- For example,

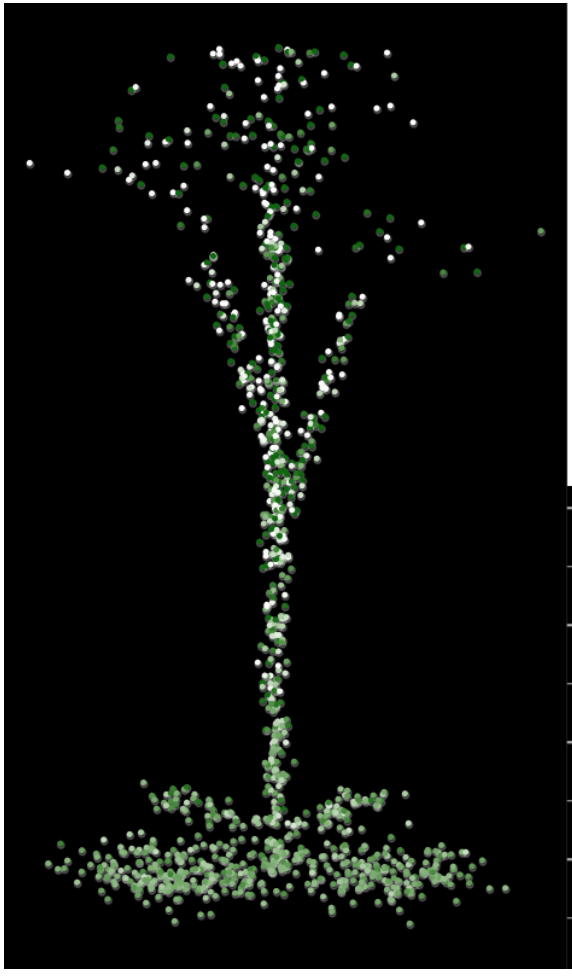
```
ggarrange(g.1, ggarrange(g.2, g.3,
labels=c(``B'', ``C''), nrow=2),
labels=c(``A'', ``''), widths=c(1,2), nrow=1)
```

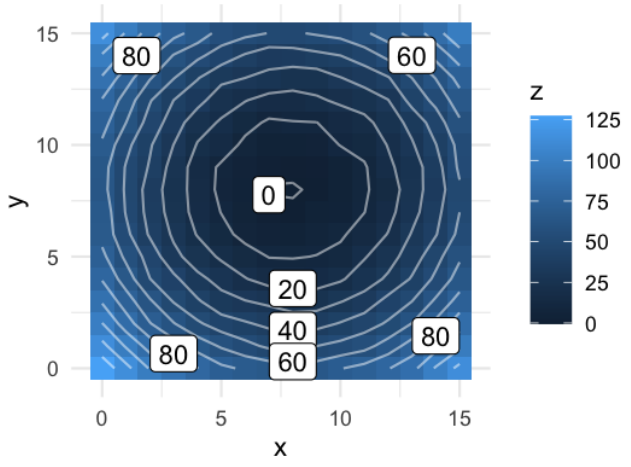


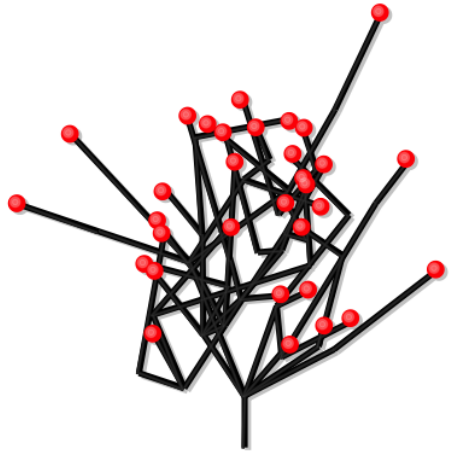
Which geoms?

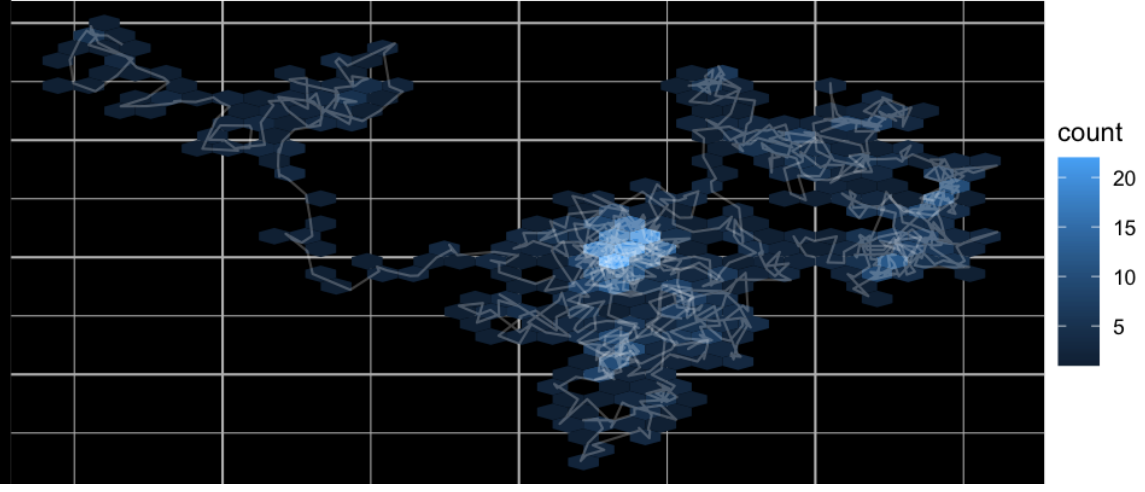

A
10.0
7.5
5.0
2.5
0.0
y
2.5
5.0
7.5
10.0
x
group
1
2
3
4
5
B
2
0
-2
y
1
2
group
group
1
2
C
1
2
3
4
5
12
8
4
0
y
2.5
5.0
7.5
10.0
x
group
1
2
3
4
5

Which geoms?




15
10
5
0
y
0
5
10
15
x
80
60
0
20
40
60
80
80
z
125
100
75
50
25
0

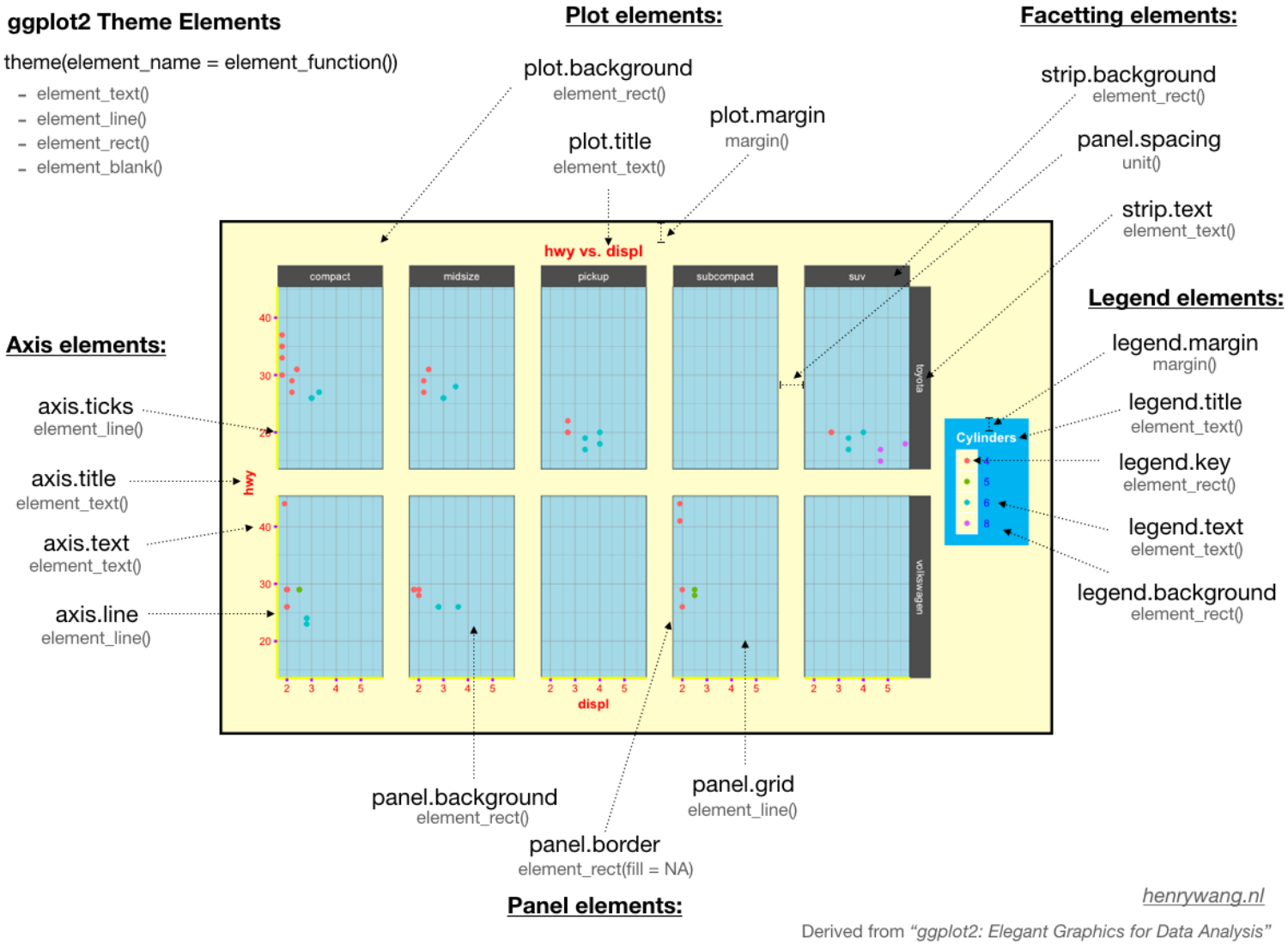



count
20
15
10
5

Plot theme styling

- Every part of the plot is addressable as an element
- `element_...` functions take styling arguments: `text`, `line`, `rect`, ...
- Overall `theme_...()`: `light`, `minimal`, `bw`, `classic`
- `theme(...)`:
  - `legend.position = ``none''`
  - `axis.text.x = element_text(size=12, angle=45, hjust=1)`
  - `axis.line.x = element_line(color = ``#AAAAAA'')`
  - `axis.title.x = element_text(size=24,color = ``#AAAAAA'')`

Plot theme styling



(Henry Wang, henrywang.nl)

Plot theme styling: axes

- Some useful tweaks:
- `labs(x=``x label'', y=``y label'', color=``colour scale label'', fill=``fill scale label'',...)`
- `xlim(0, 100), ylim(0, NA)` (also `lims`)
- `scale_x_continuous(trans = ``log'')` – log (or other) transformation
- `scale_x_continuous(expand = c(0.1, 0.1))` – expand axis (e.g. to fit labels)
- `scale_x_continuous(breaks = c(0, 1, 2, 3), labels=c("Reference", "Early", "Late", "Complete"))` – custom ticks
- `theme(axis.text.x = element_text(angle=45, hjust=1))` – diagonal tick labels
- For discrete scales, factor levels will be plotted in their internal order. (Re)set with

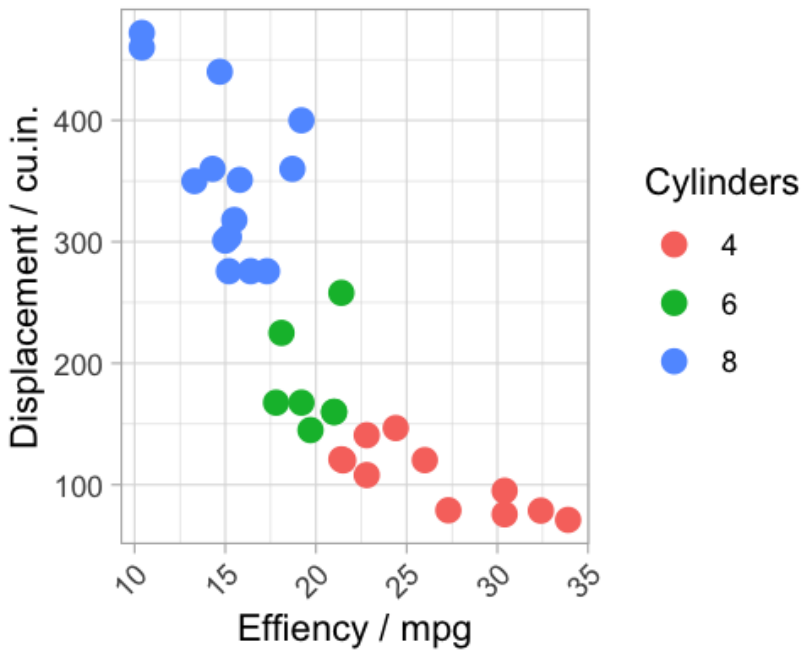
```
f = factor(f, levels=c(``First level'',
``Second level''))
```

Exercise

- Continuous-continuous data

  `data(mtcars)`

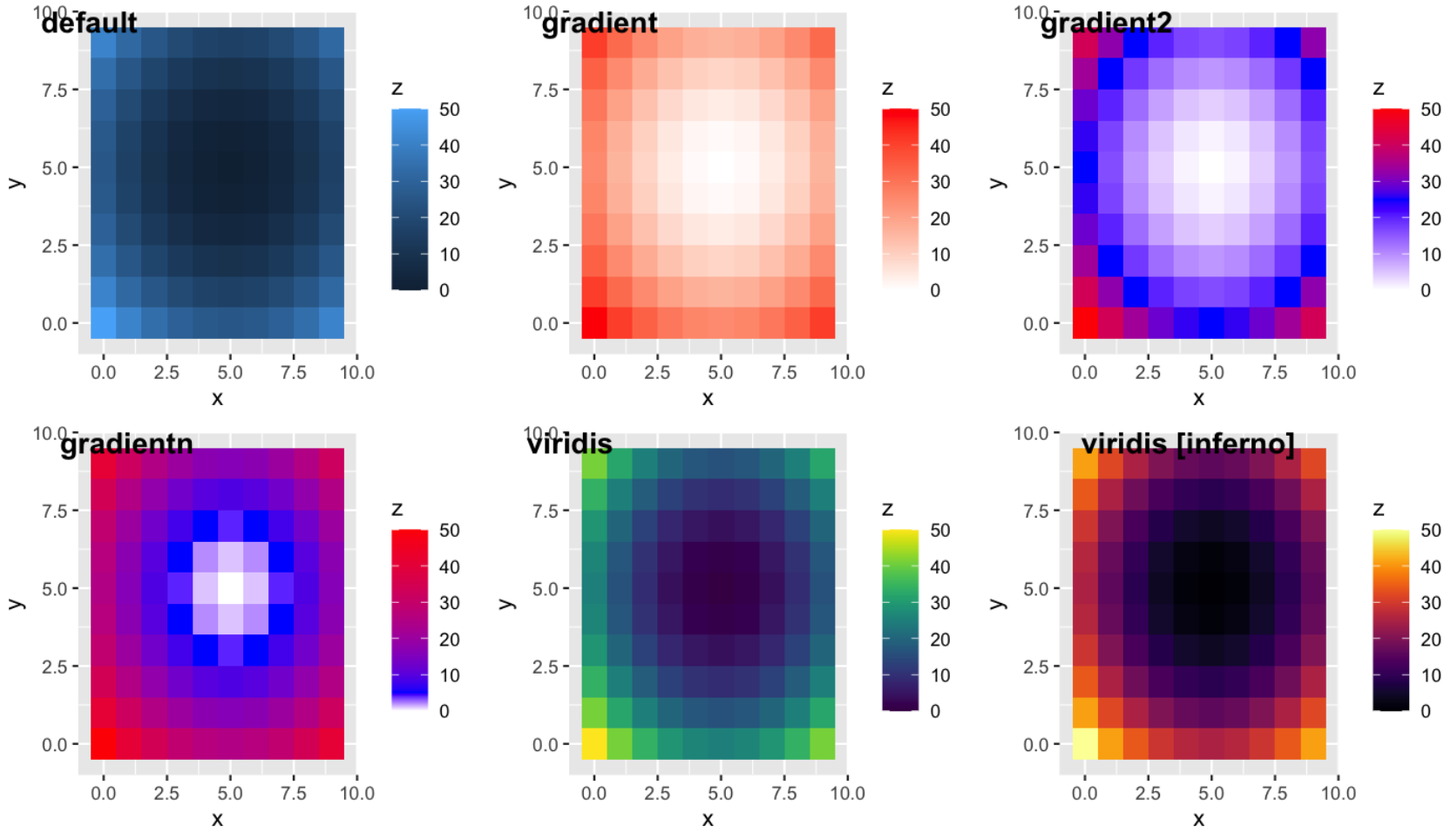
- `mtcars` has data on several car models. `mpg` (miles per gallon), `disp` displacement, `cyl` number of cylinders
- Graphically explore the relationship between these. Can you reproduce this plot?



Colours

- We think differently about continuous and discrete scales
- For both, we can tell ggplot a set of colours, or a scheme, that defines a scale
- `scale_fill_gradient`, `gradient2`, `gradientn`: specify low-high, low-mid-high, custom colours
- `scale_fill_viridis`*: specify option (default, inferno, ...)
- Replace `fill` with `colour` as required
- For factor variables with viridis, use `discrete=TRUE`
- I like viridis; also consider ColorBrewer

Colours


default
gradient
gradient2
gradientn
viridis
viridis [inferno]
x
y
z
0.0
2.5
5.0
7.5
10.0
0
10
20
30
40
50

Colours

- Perception: consider (maximising) distinctions in hue and brightness

(Introduction to the viridis color maps, Bob Rudis, Noam Ross and Simon Garnier)

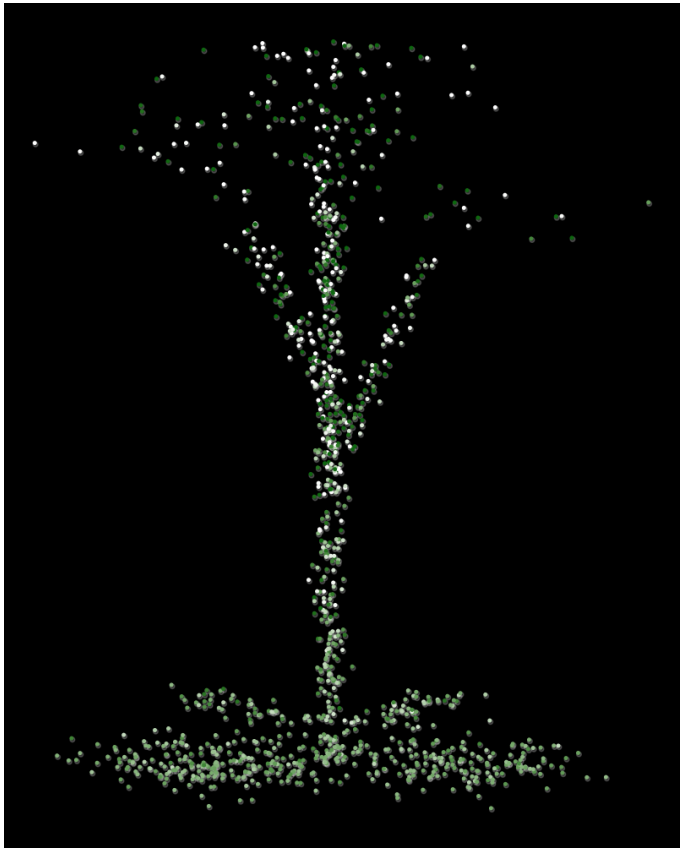
Colours

- Perception: consider (maximising) distinctions in hue and brightness
- Consider colourblind viewers / black-and-white prints
  `https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html`

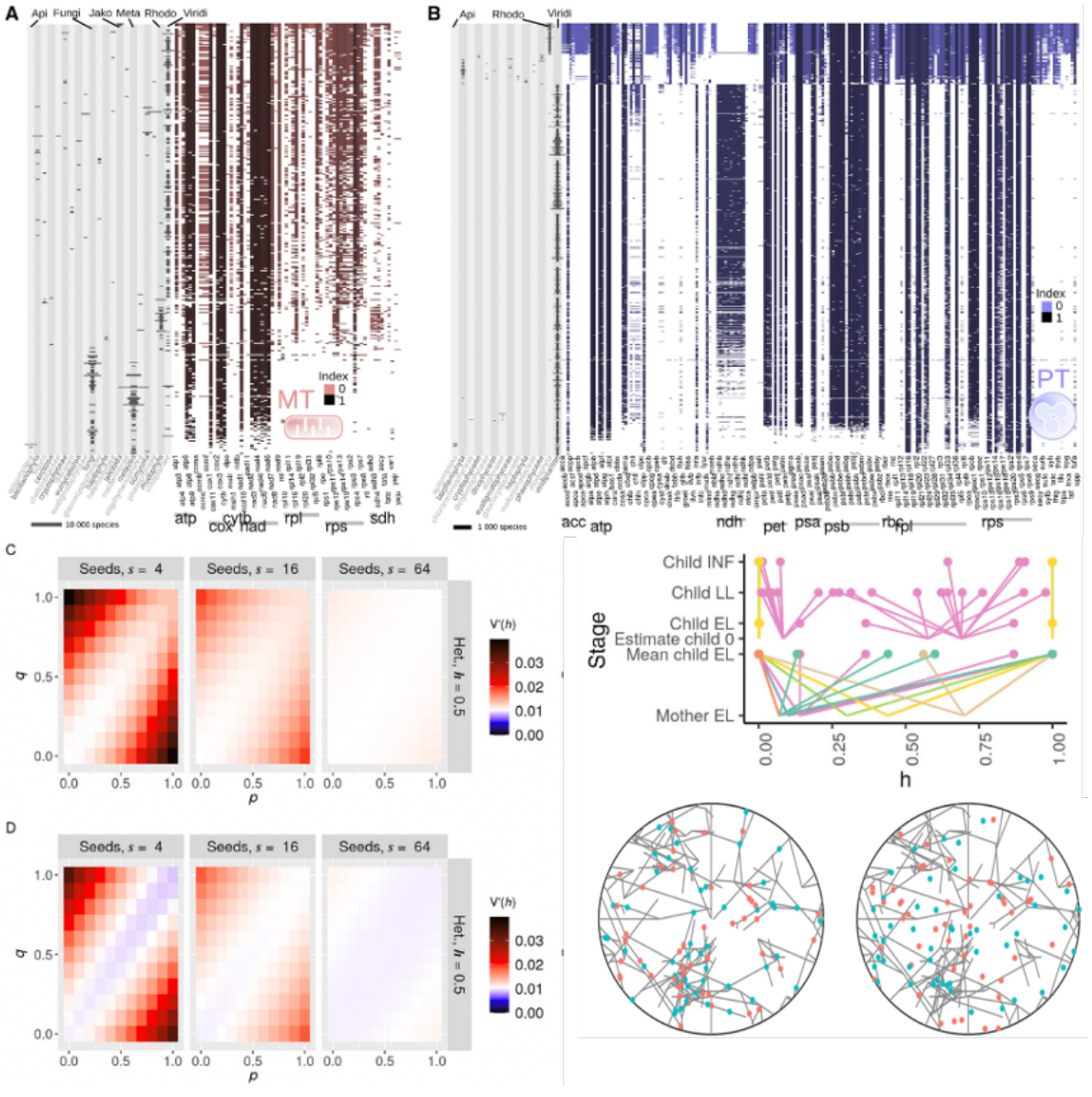
(Introduction to the viridis color maps, Bob Rudis, Noam Ross and Simon Garnier)

Exercise

- What's going on here?



Which geoms?


A
Api
Fungi
Jako
Meta
Rhodo
Viridi
Index
0
1
MT
atp
cox
cytb
nad
rpl
rps
sdh
B
Api
Rhodo
Viridi
Index
0
1
PT
acc
atp
ndh
pet
psa
psb
rbc
rpl
rps
C
Seeds, s = 4
Seeds, s = 16
Seeds, s = 64
Het, h = 0.5
V'(h)
0.03
0.02
0.01
0.00
q
p
D
Seeds, s = 4
Seeds, s = 16
Seeds, s = 64
Het, h = 0.5
V'(h)
0.03
0.02
0.01
0.00
q
p
Stage
Child INF
Child LL
Child EL
Estimate child 0
Mean child EL
Mother EL
h

Summaries

- Match scales, colours, fonts (basically, consistent style)
- Have information occupy most of the plot
- Plot as close to individual datapoints as possible
- Every approximation (especially smoothing) needs explicit mention
- Define all degrees of freedom (only) in figure caption / oral commentary
- Consider accessibility

Summaries

- Plot command + data + mapping/aesthetic + geoms + styling options
- Consider PNG and SVG(lite)
- Use the internet!
- Theory of mind!
- Be creative, transparent, and thoughtful!

```
ggplot(df.rw, aes(x=x,y=y)) +
  geom_hex() +
  geom_path(color="white", alpha=0.5) +
  scale_fill_viridis() +
  theme_dark()
```