Data Visualisation with R
Workshop Part 1

Multiple layers, facetting and tidying your data

Presented by Emi Tanaka

Department of Econometrics and Business Statistics

MONASH University

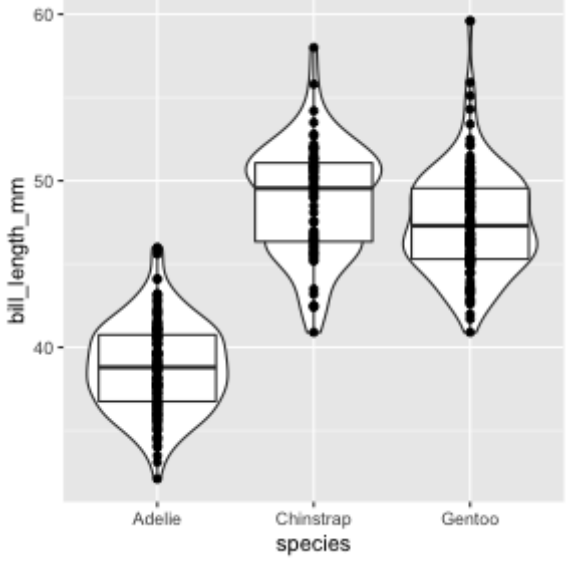
6th Dec 2021 @ Statistical Society of Australia NSW Branch | Online

Add multiple layers

```
ggplot(<data>, aes(<mappings>)) +
  <layer>() +
  <layer>() +
  <layer>()
```

Each layer inherits mapping and data from `ggplot` by default.

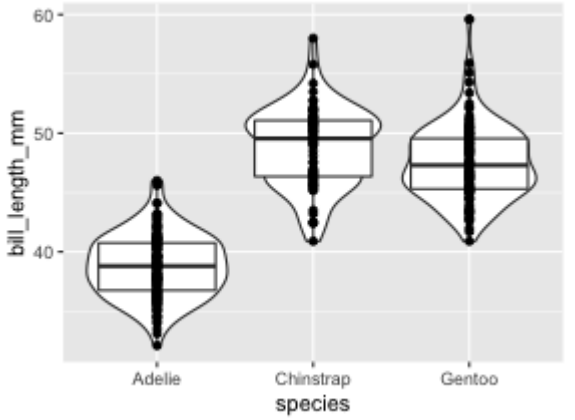
```
ggplot(penguins, aes(x = species, y = bill_length_mm)) +
  geom_violin() +
  geom_boxplot() +
  geom_point()
```



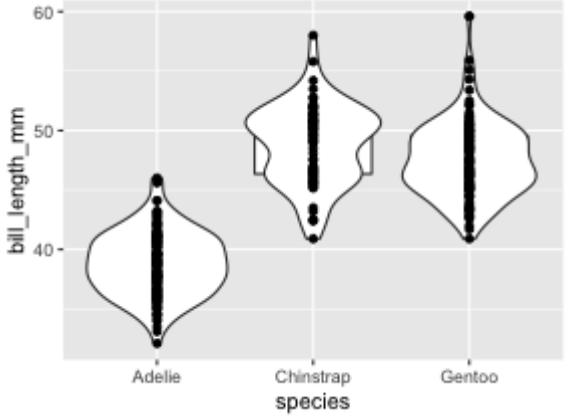
Order of the layers matters!

Boxplot and violin plot order are switched around.

```
ggplot(penguins, aes(species, bill_length_mm))  +
  geom_violin() +
  geom_boxplot() +
  geom_point()
```




```
ggplot(penguins, aes(species, bill_length_mm)) +
  geom_boxplot() +
  geom_violin() +
  geom_point()
```

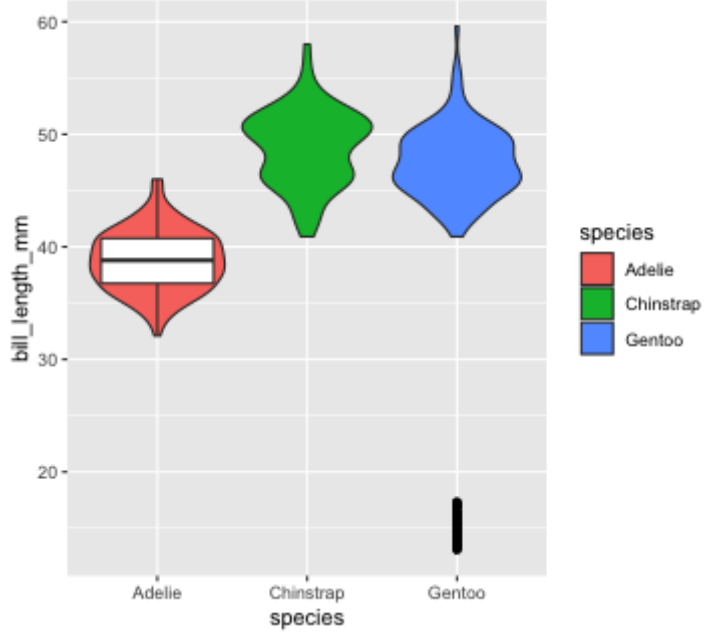


Layer-specific data and aesthetic mappings

```
ggplot(<data>, aes(<mappings>)) +
  <layer>(aes(<mappings>)) +
  <layer>(data = <data>) +
  <layer>(aes(<mappings>), <data>)
```

For each layer, aesthetic and/or data can be overwritten.

```
ggplot(penguins, aes(species, bill_length_mm)) +
  geom_violin(aes(fill = species)) +
  geom_boxplot(data = filter(penguins, species=="Adelie")) +
  geom_point(data = filter(penguins, species=="Gentoo"),
             aes(y = bill_depth_mm))
```



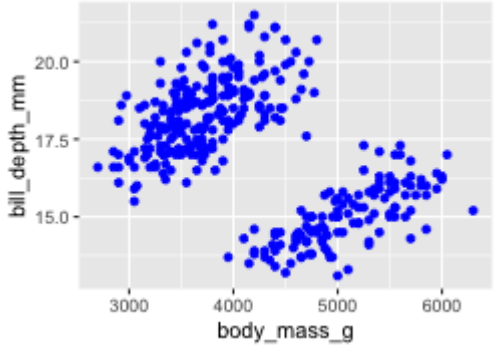
Aesthetic or Attribute?

Not what you want

```
ggplot(penguins) +
  geom_point(aes(body_mass_g,
                 bill_depth_mm,
                 color = "blue"))
```

What you want

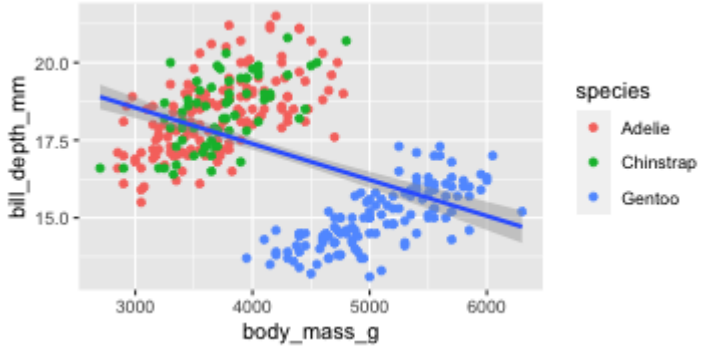
```
ggplot(penguins) +
  geom_point(aes(body_mass_g,
                 bill_depth_mm),
             color = "blue")
```




```
ggplot(penguins) +
  geom_point(aes(body_mass_g,
                 bill_depth_mm,
                 color = I("blue")))
```

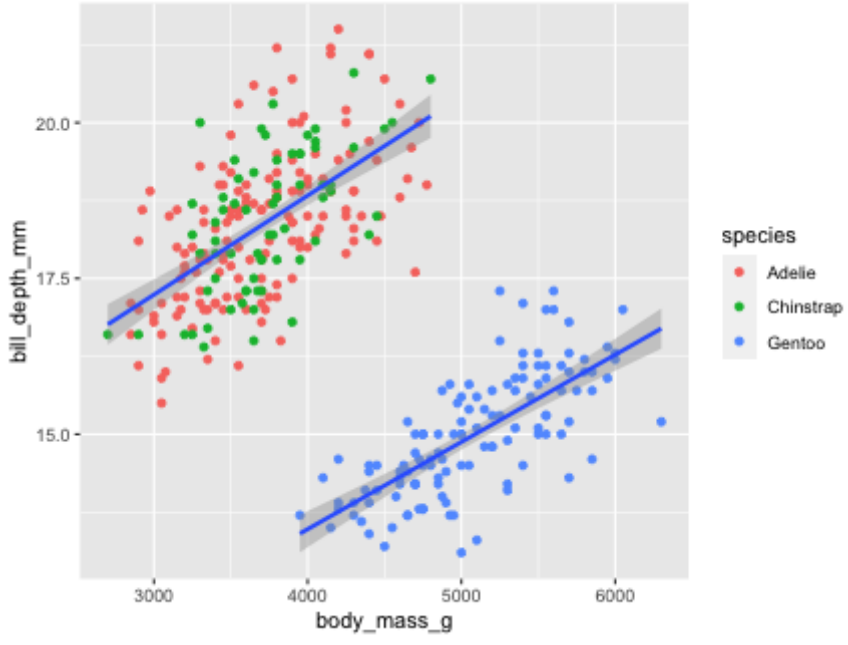
group in ggplot

```
ggplot(penguins,
       aes(body_mass_g,
           bill_depth_mm)) +
  geom_point(aes(color = species)) +
  geom_smooth(method = "lm")
```



- This is an obvious case of Simpson's paradox.
- What if we wanted to draw the fit of a simple linear model for each

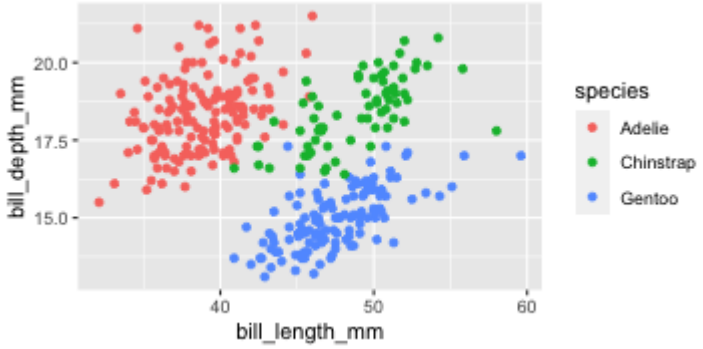
```
ggplot(penguins,
       aes(body_mass_g,
           bill_depth_mm)) +
  geom_point(aes(color = species)) +
  geom_smooth(method = "lm",
              aes(group = species=="Gentoo"))
```



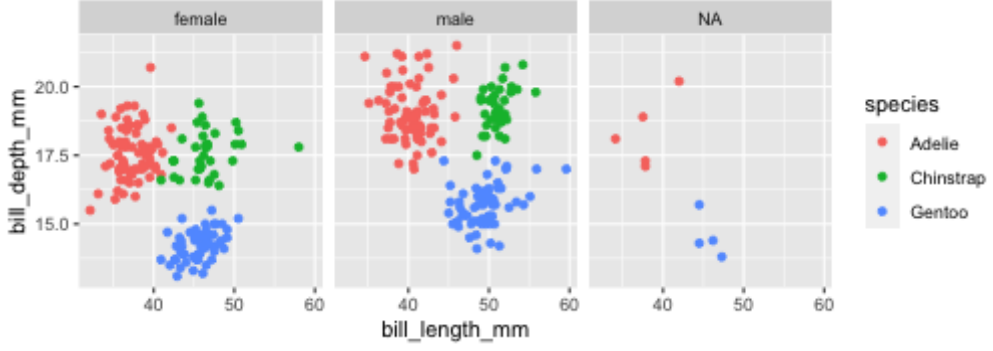
Facetting

```
g <- ggplot(penguins, aes(bill_length_mm, bill_depth_mm, color = species)) + geom_point()
```

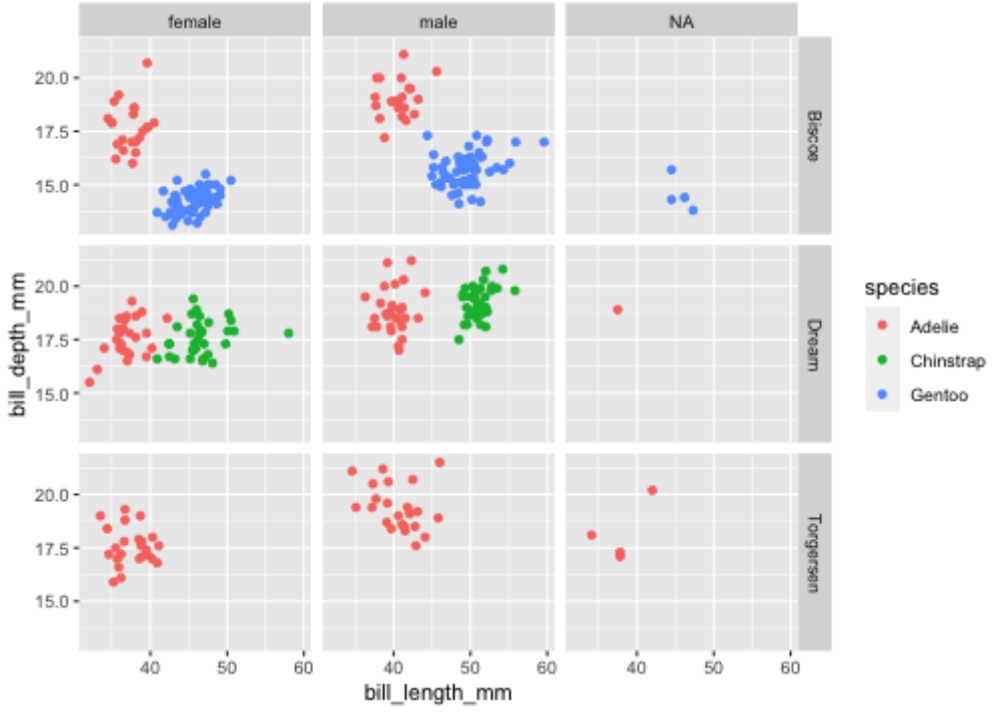
```
g
```




```
g + facet_wrap(~sex)
```

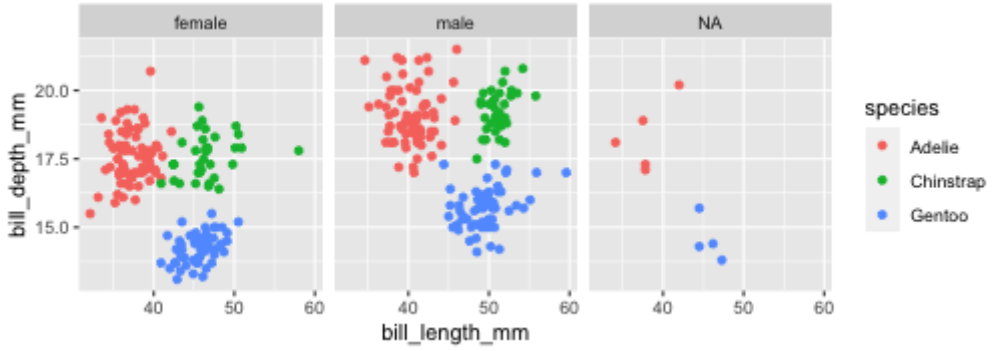



```
g + facet_grid(island ~ sex)
```

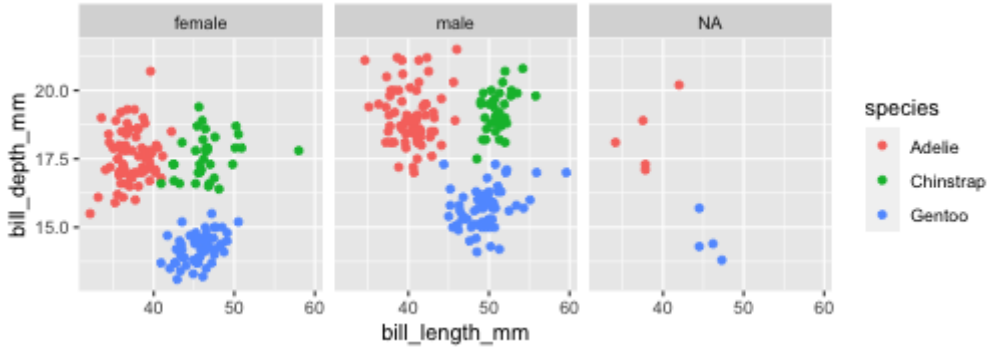


facet_wrap and facet_grid

```
g + facet_wrap( ~ sex)
```

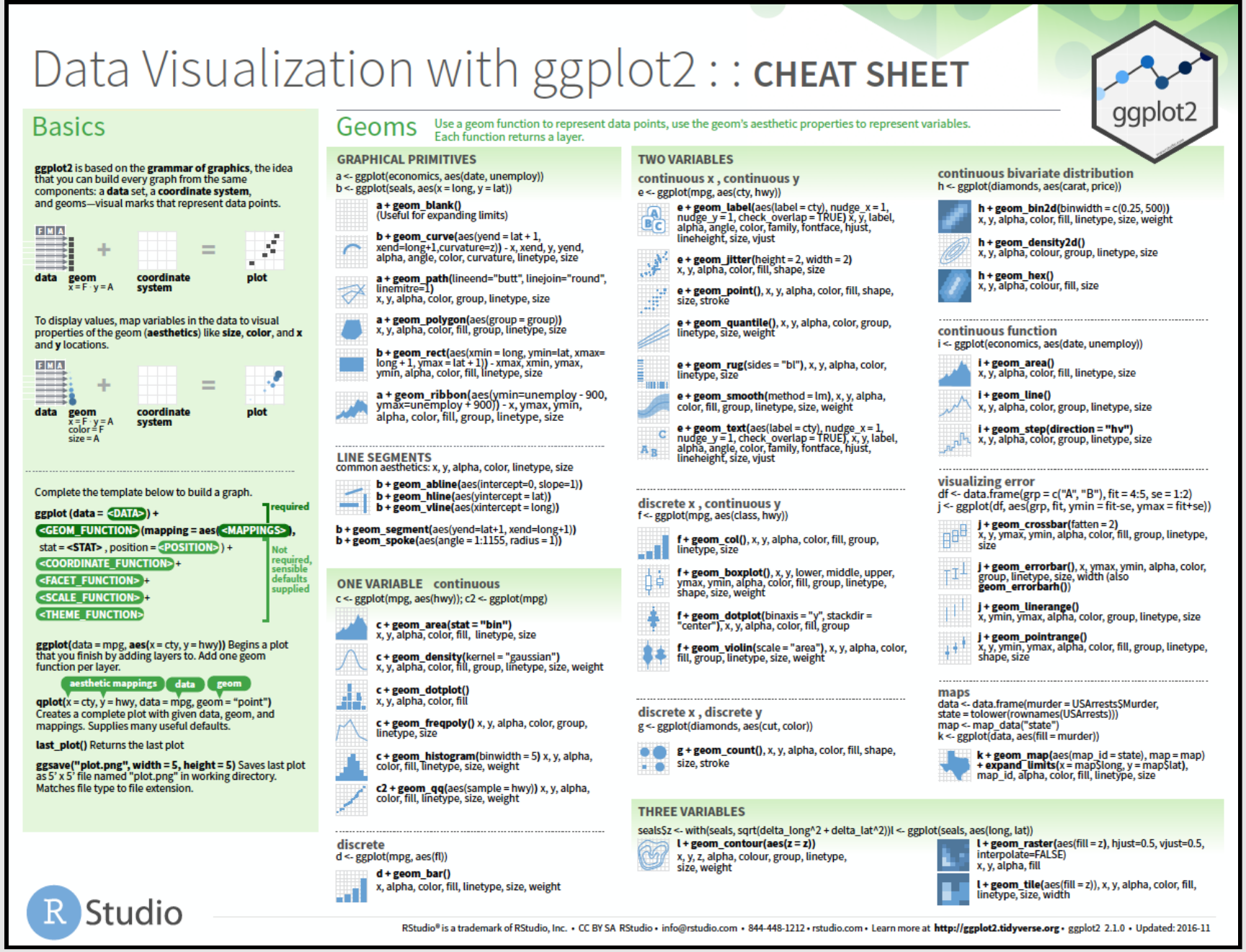



```
g + facet_grid(. ~ sex)
```




```
g + facet_wrap( ~ sex, ncol = 1)
```

```
g + facet_grid(sex ~ .)
```



Data Visualization with ggplot2 : : CHEAT SHEET

ggplot2

Basics

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

data + geom (x = F, y = A) = coordinate system → plot

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data + geom (x = F, y = A, color = F, size = A) = coordinate system → plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),
 stat = <STAT> , position = <POSITION> ) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

required

Not required, sensible defaults supplied

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings | data | geom

qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

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.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

```
d <- ggplot(mpg, aes(fl))
```

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

TWO VARIABLES

continuous x , continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

```
g <- ggplot(diamonds, aes(cut, color))
```

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

j + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

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

maps

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

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

THREE VARIABLES

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))
```

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

l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

RStudio

HELP!

- RStudio > Help > Cheatsheets
- R4DS Community Slack
- Twitter with hastag #rstats
- RStudio Community
- Stackoverflow

Tidying your data

Wickham (2014) Tidy Data. *Journal of Statistical Software* 59 (10): 1–23.

Weight gain in pigs for different treatments

The `crampton.pig` is from the `agridat` 📦

```
library(agridat)
glimpse(crampton.pig)

## Rows: 50
## Columns: 5
## $ treatment <fct> T1, T1, T1, T1, T1, T1, T1, T1, T1, T1, T2, T2, T2, T2, T2,
## $ rep       <fct> R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R1, R2, R3, R4, R5
## $ weight1   <int> 30, 21, 21, 33, 27, 24, 20, 29, 28, 26, 26, 24, 20, 35, 25,
## $ feed      <int> 674, 628, 661, 694, 713, 585, 575, 638, 632, 637, 699, 626,
## $ weight2   <int> 195, 177, 180, 200, 197, 170, 150, 180, 192, 184, 194, 204,
```

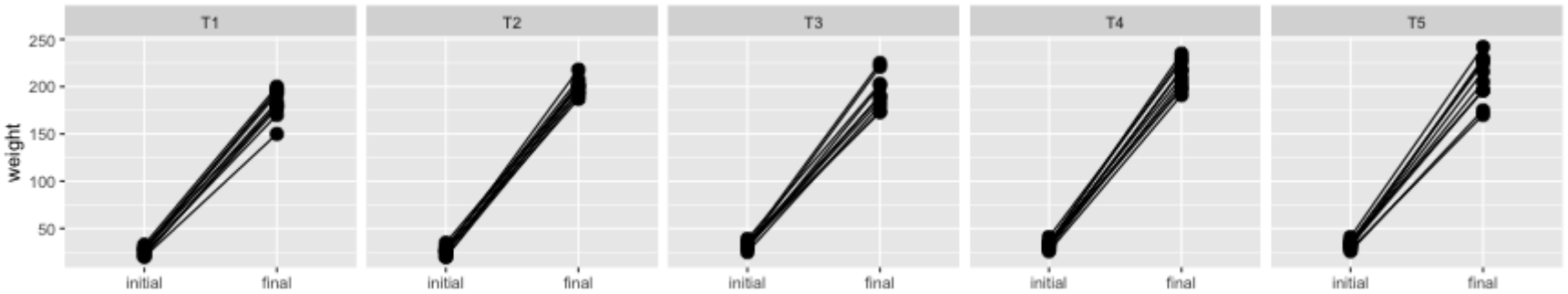
`weight1` is initial weight and `weight2` is final weight

Wright (2018). agridat: Agricultural Datasets. R package version 1.16. https://CRAN.R-project.org/package=agridat

Crampton and Hopkins (1934). The Use of the Method of Partial Regression in the Analysis of Comparative Feeding Trial Data, Part II. *The Journal of Nutrition* 8 113-123.

```
names(crampton.pig)

## [1] "treatment" "rep"       "weight1"   "feed"      "weight2"
```

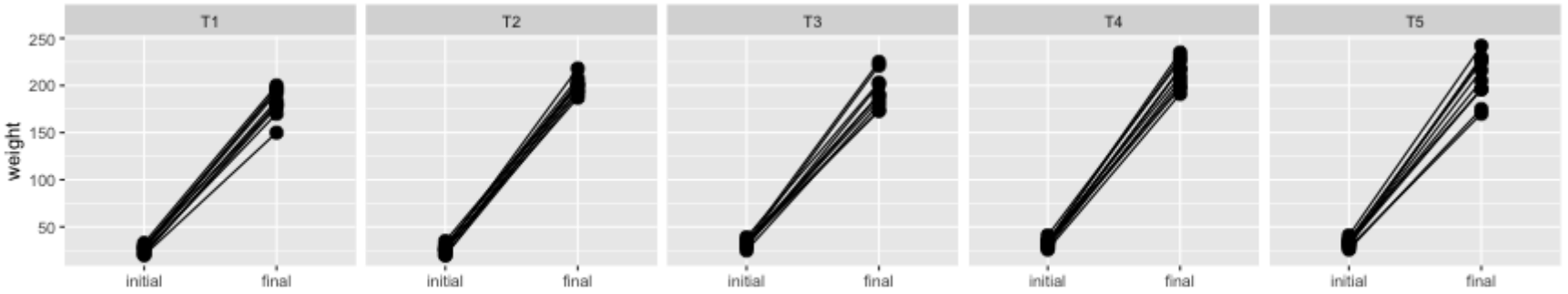


What are the mappings to get the above graph? 🤔

```
ggplot(crampton.pig, aes(x = ???, y = ???)) +
  geom_point() +
  geom_line() +
  facet_grid(. ~ treatment)
```

😐

Getting the data in the right form



- The x-axis is the time when pig was weighed
- The y-axis is the weight
- The facetting is by treatment

```
## # A tibble: 100 × 5
##    treatment  feed id    when    weight
##    <fct>     <int> <chr> <fct>    <int>
##  1 T1          674 pig1  initial     30
##  2 T1          674 pig1  final      195
##  3 T1          628 pig2  initial     21
```

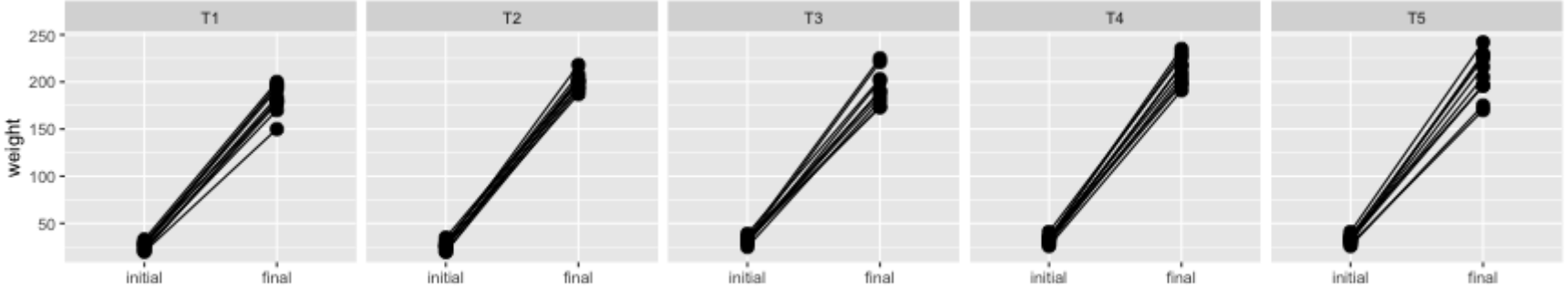
How I wrangled this data

```
pig_df <- crampton.pig %>%
 mutate(id = paste0("pig", 1:n())) %>%
 pivot_longer(c(weight1, weight2),
              names_to = "when",
              values_to = "weight") %>%
 mutate(when = factor(when,
         levels = c("weight1", "weight2"),
         labels = c("initial", "final")))
```

(note: teaching wrangling is not part of this workshop, please see here if you want to learn more)

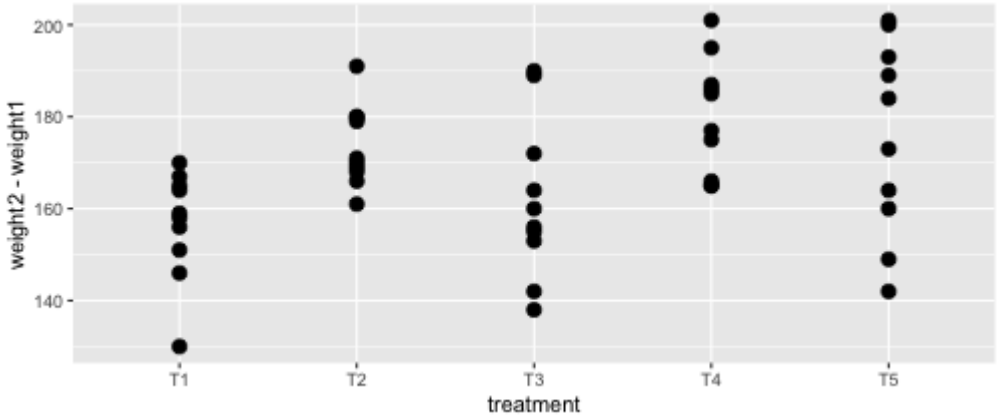
Putting it all together

```
ggplot(pig_df, aes(when, weight)) + # tidying your data for plotting
  geom_point(size = 3) + # attribute not aesthestic
  geom_line(aes(group = id)) + # grouping
  facet_grid(. ~ treatment) + # facetting
  labs(x = "") # we'll learn this in the last session
```



Meaningfully order categorical variables

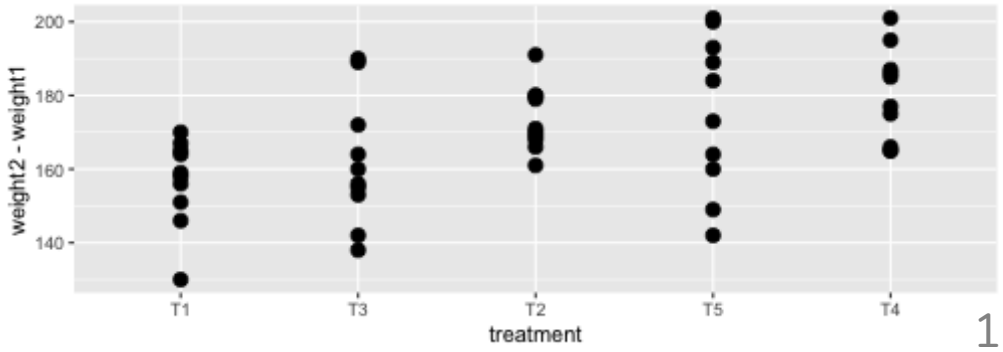
```
ggplot(crampton.pig,
  aes(treatment, weight2 - weight1)) +
  geom_point(size = 3)
```



- Treatments are ordered alphabetically by default
- It's better to order categorical variables meaningfully

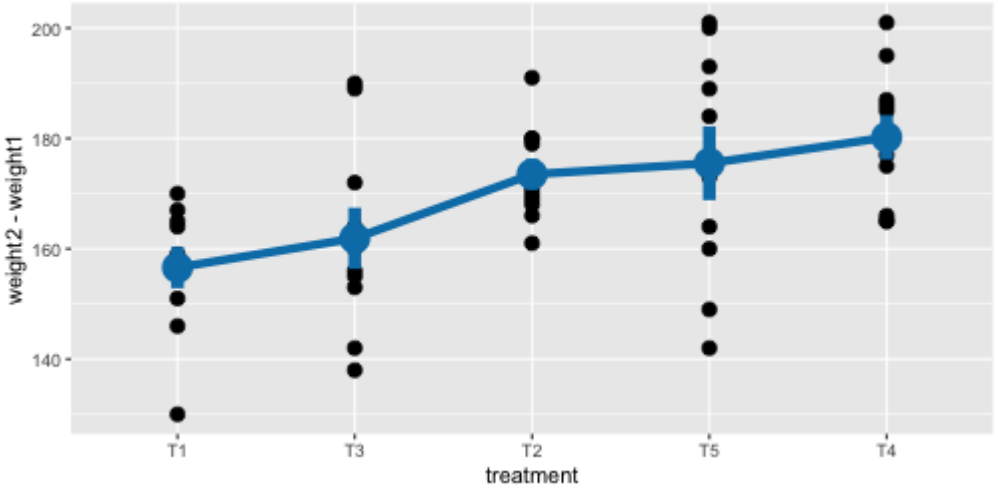
Order factor levels by the mean of the weight difference.

```
library(forcats) #  for easy factor manipulation
crampton.pig2 <- crampton.pig %>%
 mutate(
  treatment = fct_reorder(treatment,
                          weight2 - weight1,
                          mean))
ggplot(crampton.pig2,
       aes(treatment, weight2 - weight1)) +
  geom_point(size = 3)
```



Plotting auxilliary data

Plot you may want:

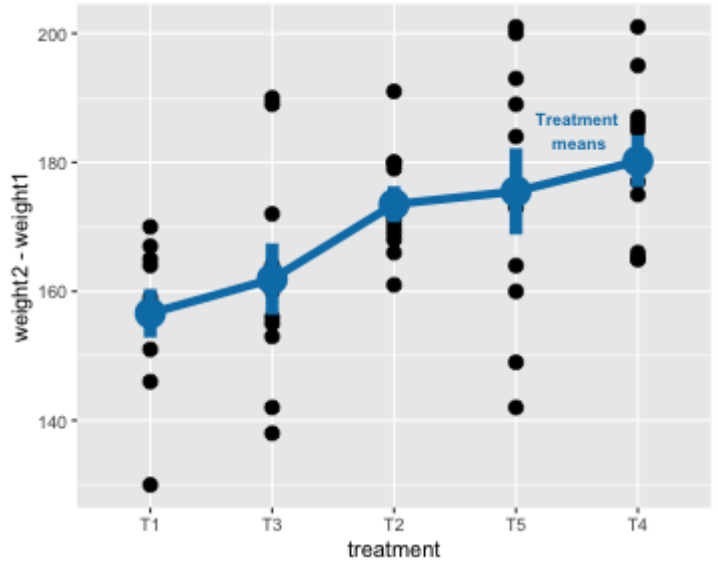


One way to do this:

```
fig <- ggplot(crampton.pig2,
              aes(treatment,
                  weight2 - weight1)) +
  geom_point(size = 3) +
  stat_summary(fun.data = mean_se,
               geom = "pointrange", fatten = 2,
               color = "#027EB6", size = 3) +
  stat_summary(fun = mean, geom = "line",
               group = 1, color = "#027EB6",
               size = 2)
```

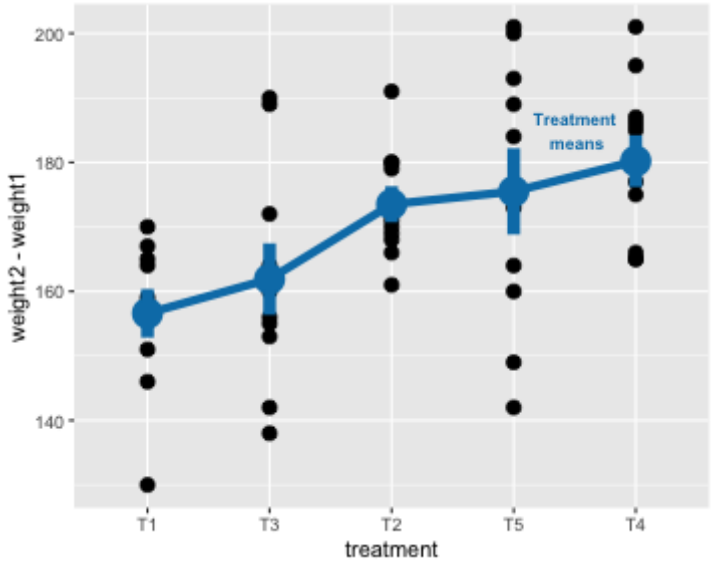
Plotting annotations

```
fig +
  geom_text(data = data.frame(treatment = 4.5,
                              weight2 = 185,
                              weight1 = 0),
            label = "Treatment\n means",
            size = 3,
            color = "#027EB6",
            fontface = "bold")
```



But it might be just easier to:

```
fig +
  annotate("text",
           x = 4.5, y = 185,
           label = "Treatment\n means",
           size = 3,
           color = "#027EB6",
           fontface = "bold")
```



</> Open part1-exercise-02.Rmd

15:00

Session Information

```
devtools::session_info()
```

```
## ─ Session info  🍼  🏖  🦸🏽  ──────────────────────────────────────────────
##  hash: baby bottle, beach with umbrella, superhero: medium skin tone
##
##  setting  value
##  version  R version 4.1.2 (2021-11-01)
##  os       macOS Big Sur 10.16
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Melbourne
##  date     2021-12-06
```