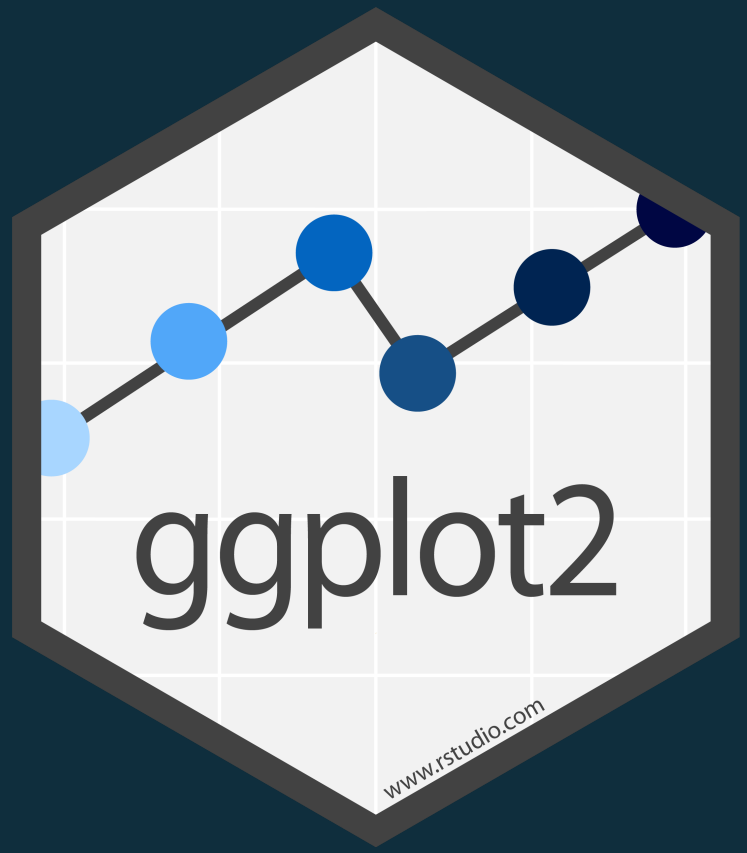
Data visualization with ggplot2

Introduction to R - Day 2

Instructor: Selina Baldauf

Freie Universität Berlin - Theoretical Ecology



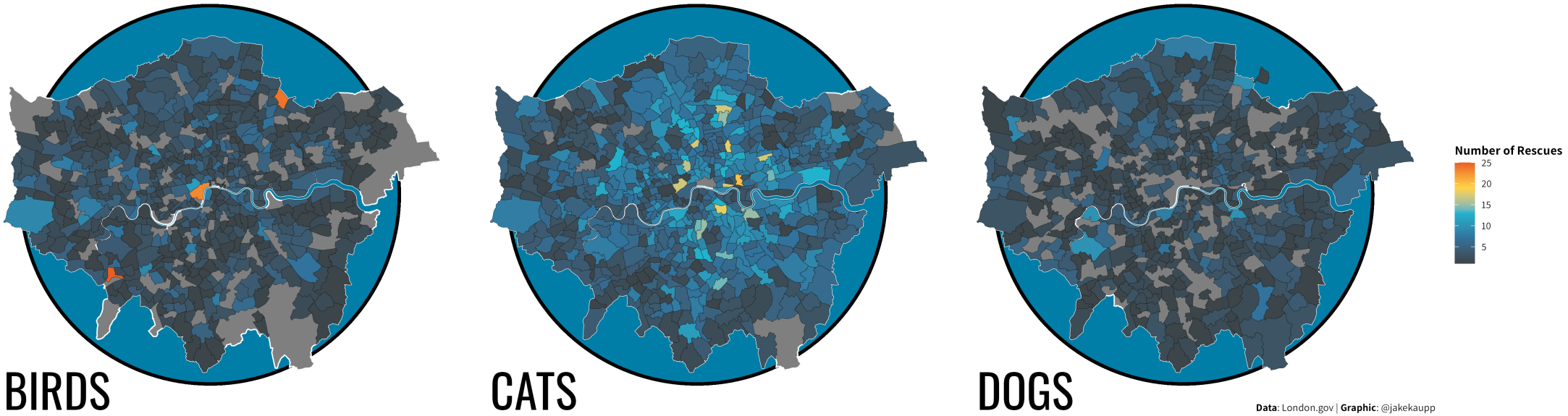
2021-08-01 (updated: 2023-02-25)

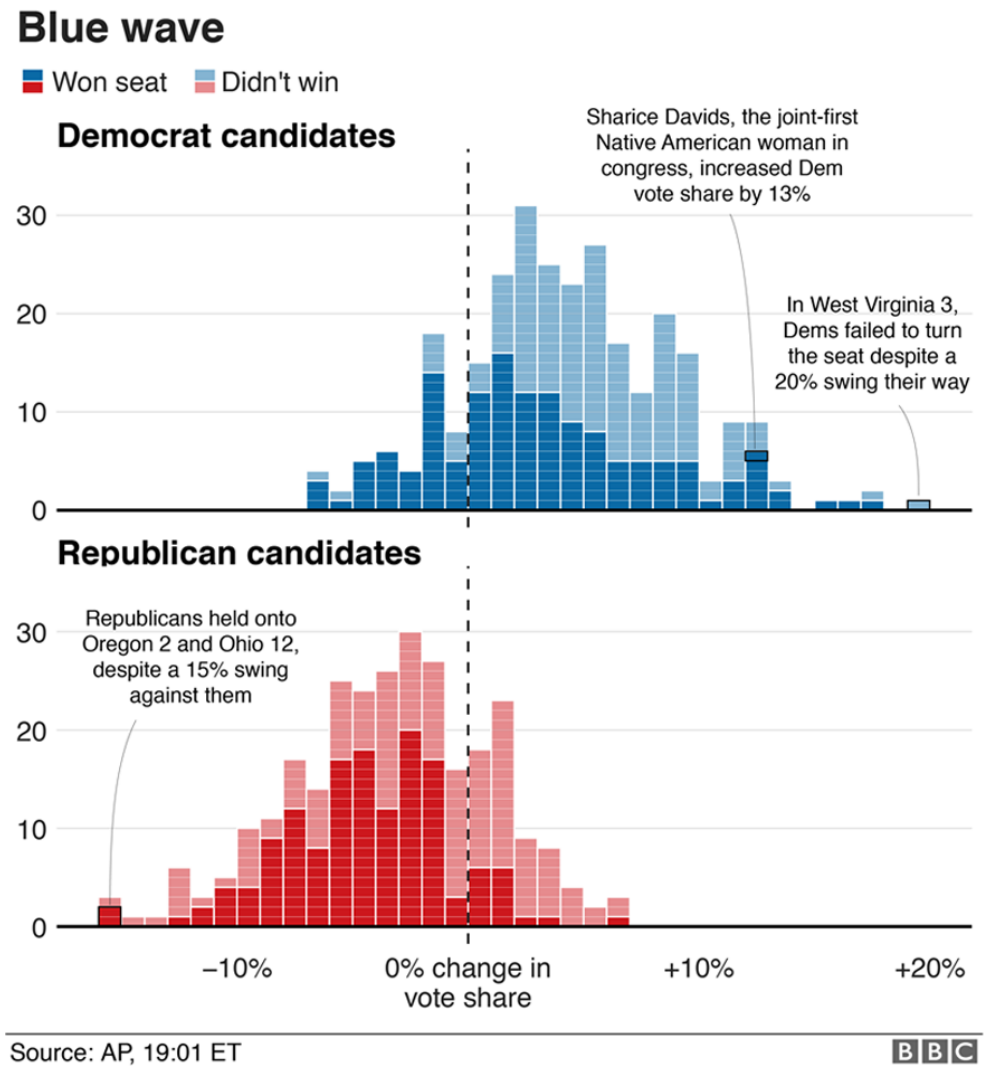
A ggplot showcase

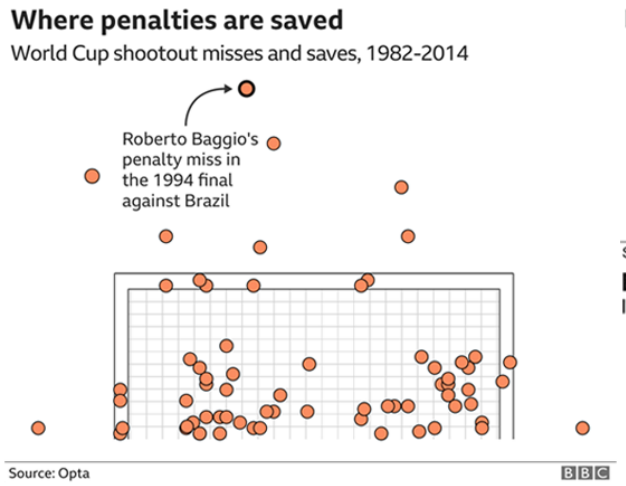
Some examples of plots you can create with ggplot

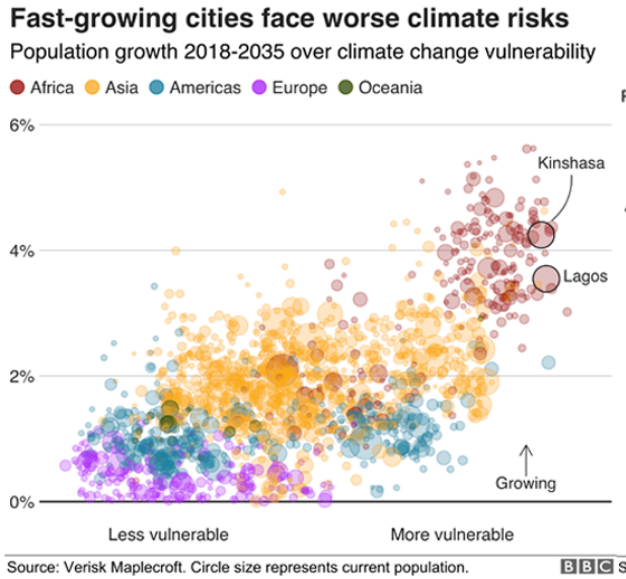
Frequency of Rescues of Birds, Cats and Dogs in London from 2009-2021

Illustrated below in three choropleth maps are rescues of birds, cats and dogs in London wards. Darker colors indicate lower rescue numbers while brighter colors indicate a greater number of rescues in that ward.

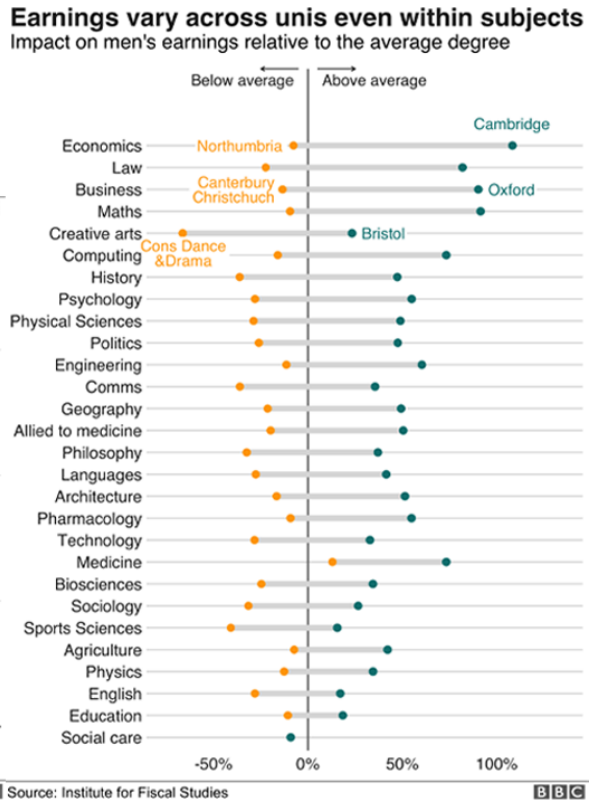


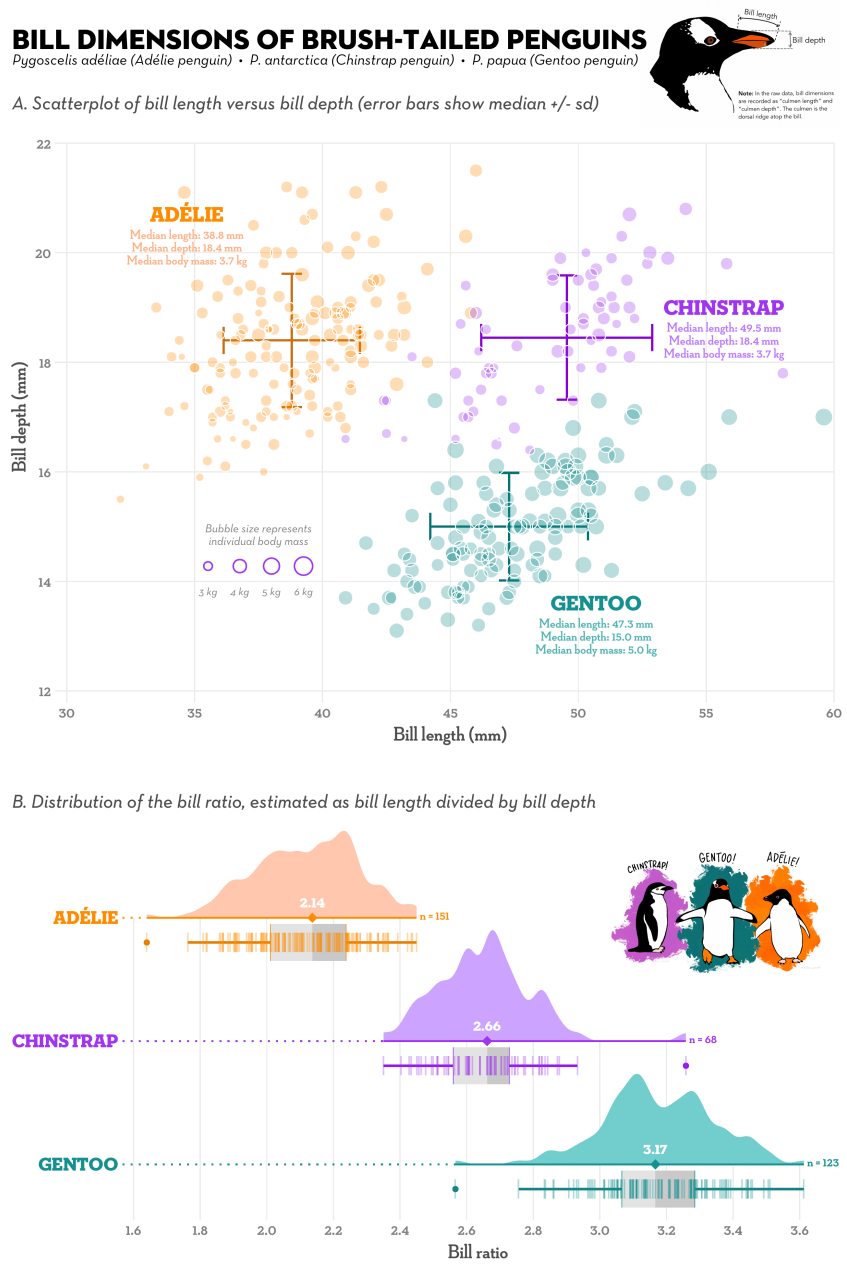

Blue wave
Won seat
Didn't win
Democrat candidates
Sharice Davids, the joint-first Native American woman in congress, increased Dem vote share by 13%
In West Virginia 3, Dems failed to turn the seat despite a 20% swing their way
30
20
10
0
Republican candidates
Republicans held onto Oregon 2 and Ohio 12, despite a 15% swing against them
30
20
10
0
–10%
0% change in vote share
+10%
+20%
Source: AP, 19:01 ET
BBC


Where penalties are saved
World Cup shootout misses and saves, 1982-2014
Roberto Baggio's penalty miss in the 1994 final against Brazil
Source: Opta
BBC


Fast-growing cities face worse climate risks
Population growth 2018-2035 over climate change vulnerability
Africa
Asia
Americas
Europe
Oceania
6%
4%
2%
0%
Kinshasa
Lagos
Growing
Less vulnerable
More vulnerable
Source: Verisk Maplecroft. Circle size represents current population.
BBC


MPs rejected Theresa May's deal by 230 votes
For
Con (196)
202
Lab 3, Ind 3
Against
Con rebels (118)
Labour (248)
432
SNP 35, Lib Dem 11, DUP 10, Plaid 4, Green 1, Ind 5
Source: Commons Votes Services. Excludes 'tellers', the Speaker and deputies
BBC


Earnings vary across unis even within subjects
Impact on men's earnings relative to the average degree
Below average
Above average
Economics
Law
Business
Maths
Creative arts
Computing
History
Psychology
Physical Sciences
Politics
Engineering
Comms
Geography
Allied to medicine
Philosophy
Languages
Architecture
Pharmacology
Technology
Medicine
Biosciences
Sociology
Sports Sciences
Agriculture
Physics
English
Education
Social care
Northumbria
Canterbury Christchuch
Cons Dance &Drama
Cambridge
Oxford
Bristol
-50%
0%
50%
100%
Source: Institute for Fiscal Studies
BBC



Visualization by Cédric Scherer,
code available on Github

Advantages of ggplot

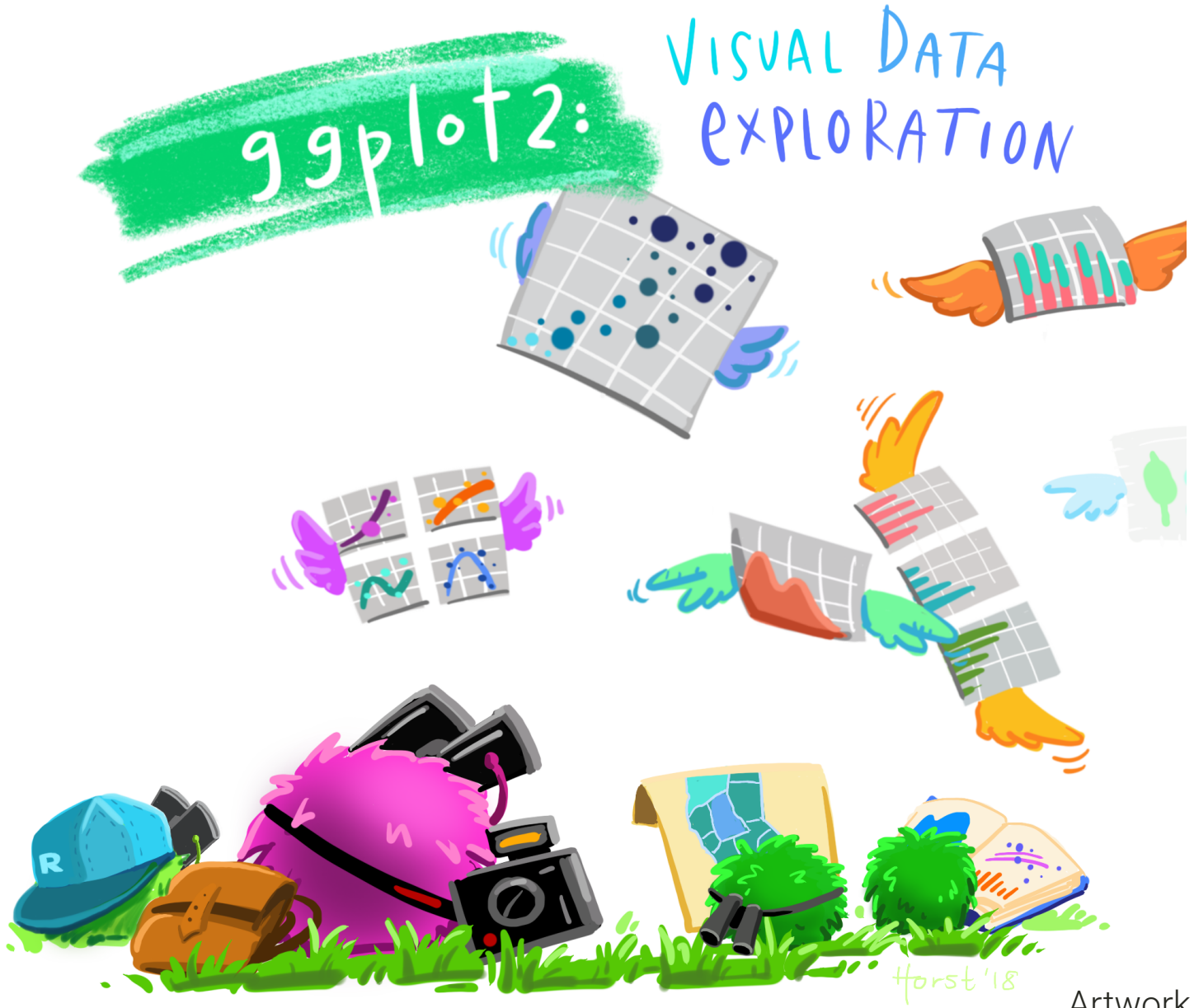
- **Consistent** grammar/structure
- **Flexible** structure allows you to produce any type of plots
- Highly **customizable appearance** (themes)
- Many **extension packages** that provide
  - Additional plot types
  - Additional themes
  - Color palettes
  - Animation
  - Composition of multiple plots
  - ...
- Active community that provides help and inspiration
- Perfect package for **exploratory data analysis** and **beautiful plots**

The data

The `lterdatasampler` package contains a data set called `and_vertebrates` about the length and weight of a fish and a salamander species in both clear cut and old growth coniferous forest sections.

Data variables (among others):

- `year`: observation year
- `section`: CC (clear cut forest) or OG (old growht forest)
- `unittype`: channel unit classification (C = Cascade, P = Pool, ...)
- `species`: Species measured
- `length_1_mm`: body length [mm]
- `weight_g`: body weight [g]



Artwork by Allison Horst

ggplot(data)

The `ggplot()` function initializes a ggplot object. Every ggplot needs this function.

```
library(ggplot2) # or library(tidyverse)

ggplot(data = and_vertebrates)
```

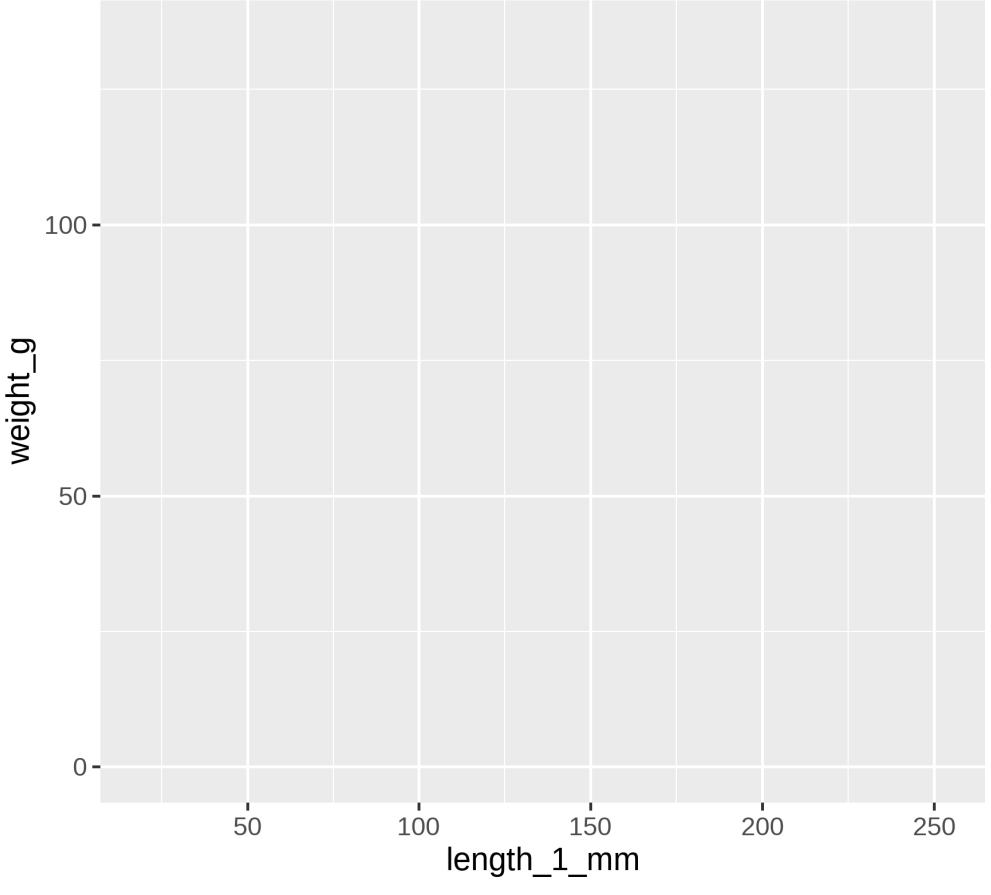
- Empty plot because we did not specify the mapping of data variables

aes(x, y)

The aesthetic mapping defines how variables are mapped to aesthetic properties of the plot.

```
ggplot(data = and_vertebrates,
       mapping = aes(
         x = length_1_mm,
         y = weight_g))
```

- Map variable `length_1_mm` to x-axis and `weight_g` to y-axis
- Default scales are automatically adapted to range of data

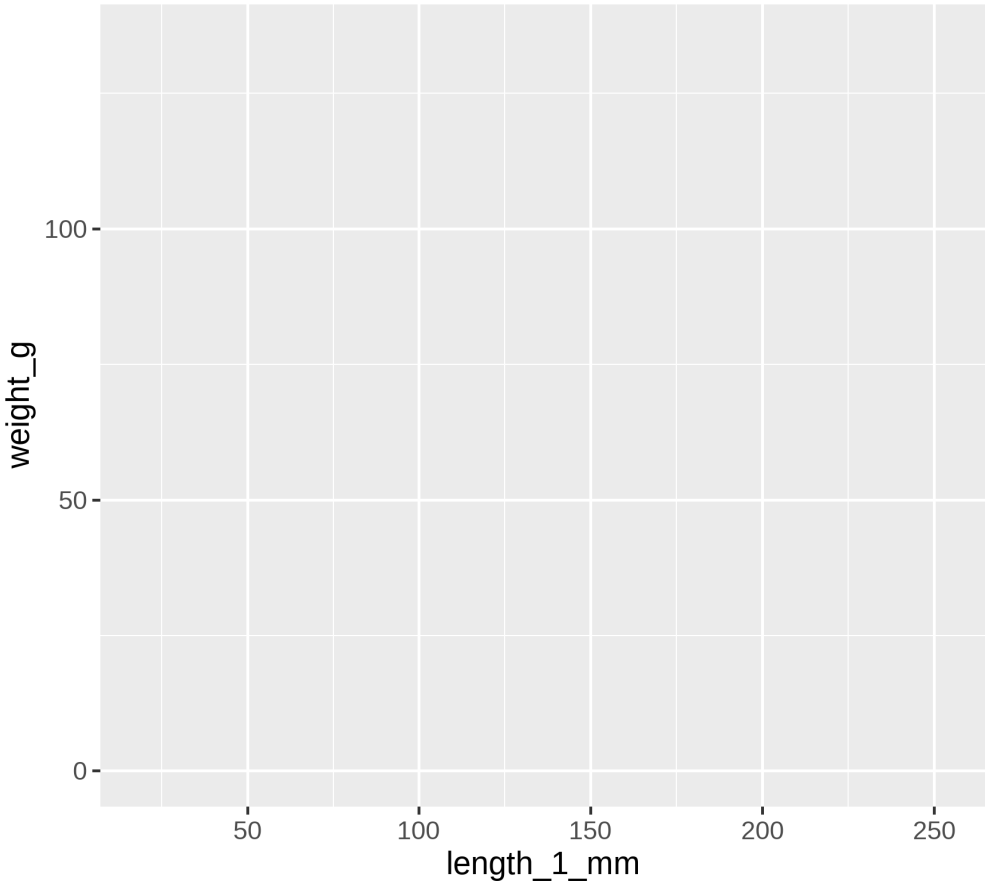


aes(x, y)

The **aes**thetic mapping defines how variables are mapped to aesthetic properties of the plot.

```
ggplot(data = and_vertebrates,
       mapping = aes(
         x = length_1_mm,
         y = weight_g))
```

- Map variable `length_1_mm` to x-axis and `weight_g` to y-axis
- Default scales are automatically adapted to range of data



This is the same but shorter:

```
ggplot(and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g))
```

Remember argument matching by position?

`geom_*()`

`geoms` define how data points are represented. There are many different geoms to chose from

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1),curvature=1) - 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

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

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))

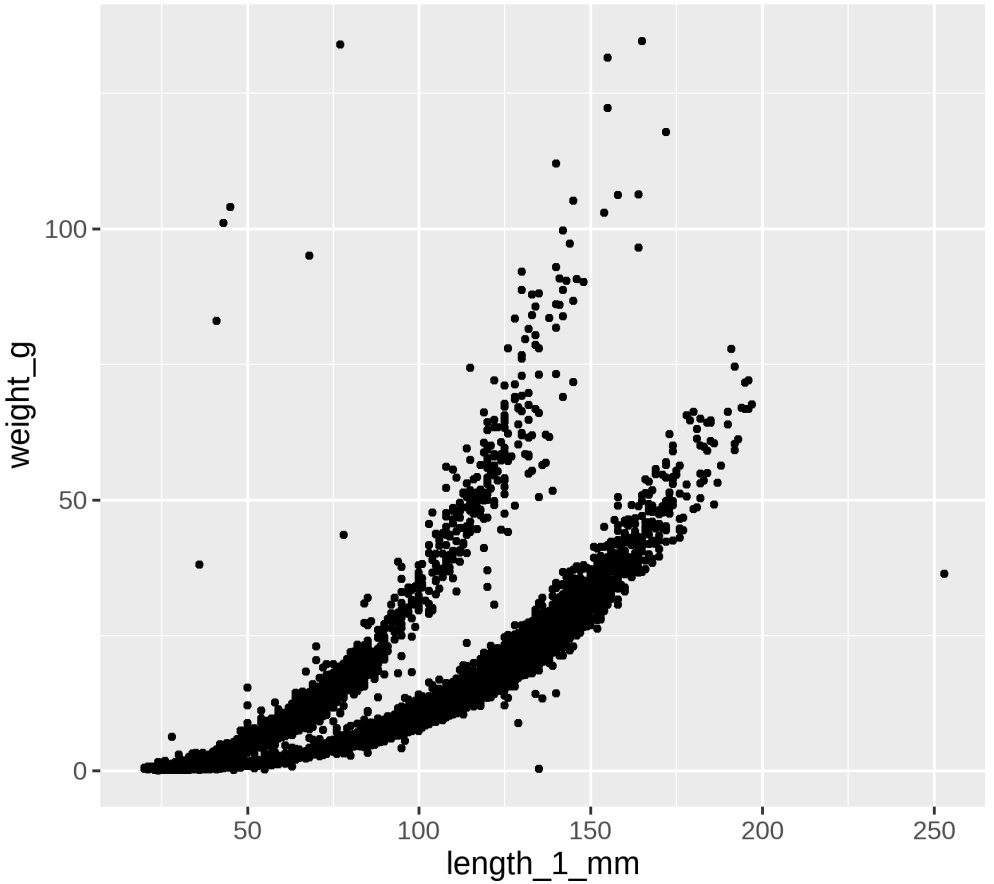
k + geom_map(aes(map_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

geom_point()

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g)) +
  geom_point()
```

- New plot layers added with `+`
- Warning that points could not be plotted due to missing values
- `data` and `aes` defined in `ggplot` call are inherited to all plot layers

```
## Warning: Removed 13270 rows containing missing
values (`geom_point()`).
```



geom_point()

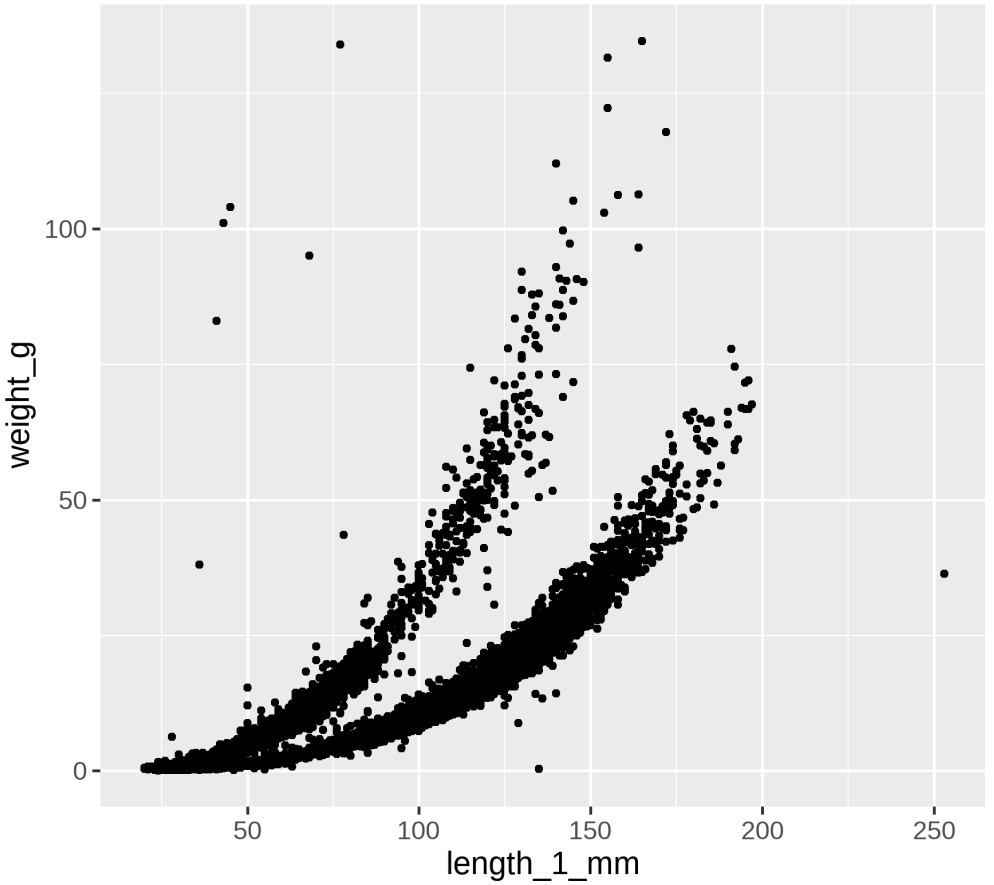
```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g)) +
  geom_point()
```

- New plot layers added with `+`
- Warning that points could not be plotted due to missing values
- `data` and `aes` defined in `ggplot` call are inherited to all plot layers
- `data` and `aes` can be local to a layer:

```
ggplot(data = and_vertebrates,) +
  geom_point(aes(x = length_1_mm,
                 y = weight_g))
```

Here, it does not make a difference.

```
## Warning: Removed 13270 rows containing missing
values (`geom_point()`).
```

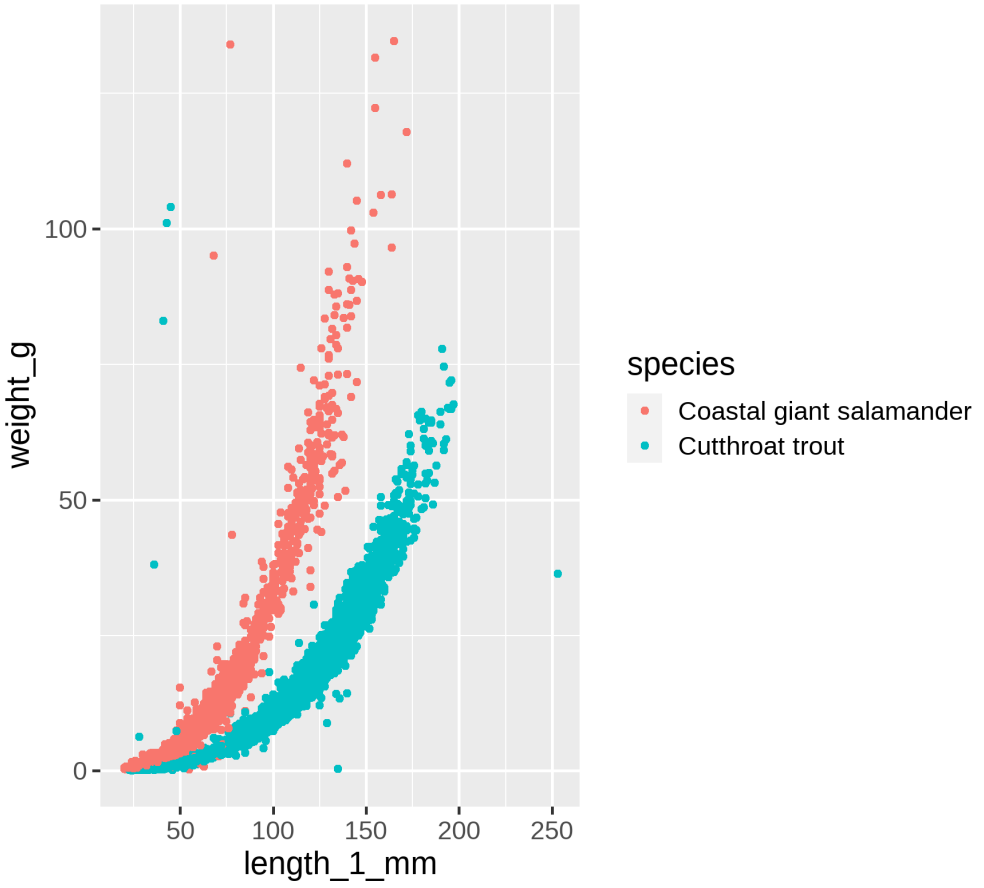


`aes(color)`: mapping color to a variable

Looks like there are two groups of data: **Map color of points to a variable** by adding it to aesthetics:

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           color = species)) +
  geom_point()
```

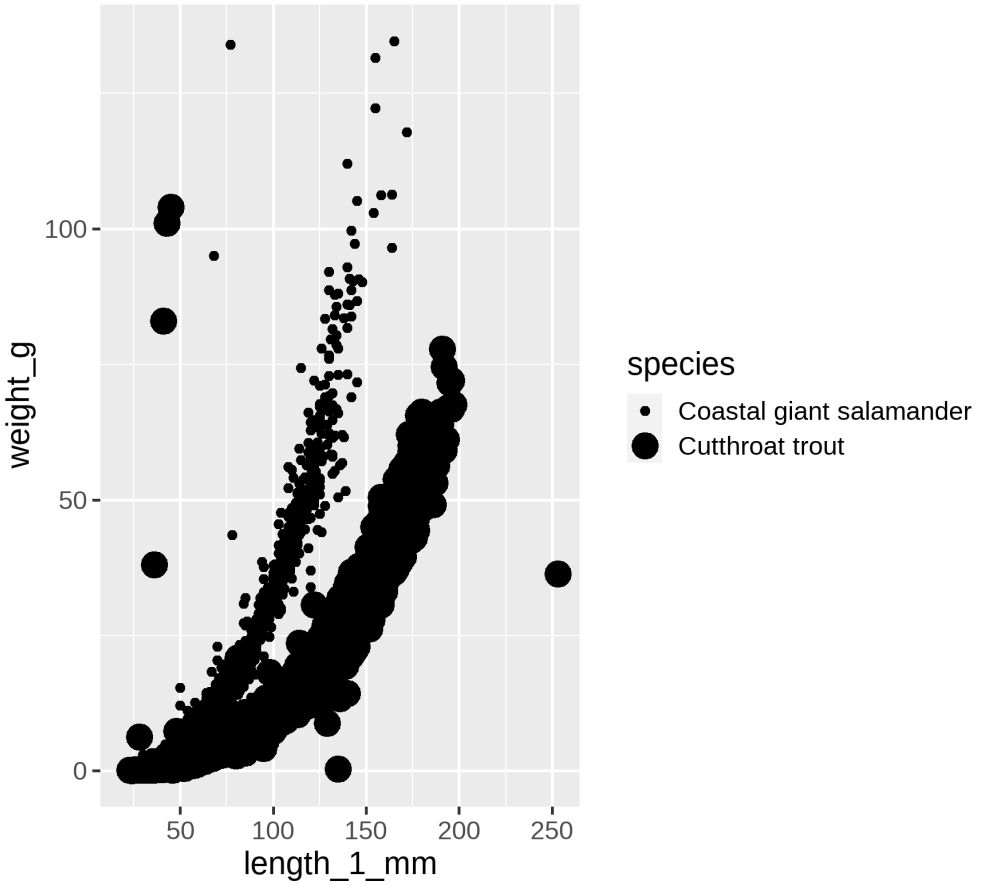
- Map the `species` variable to the color aesthetic of the plot



`aes(size)`: mapping size to a variable

We can do the same with size:

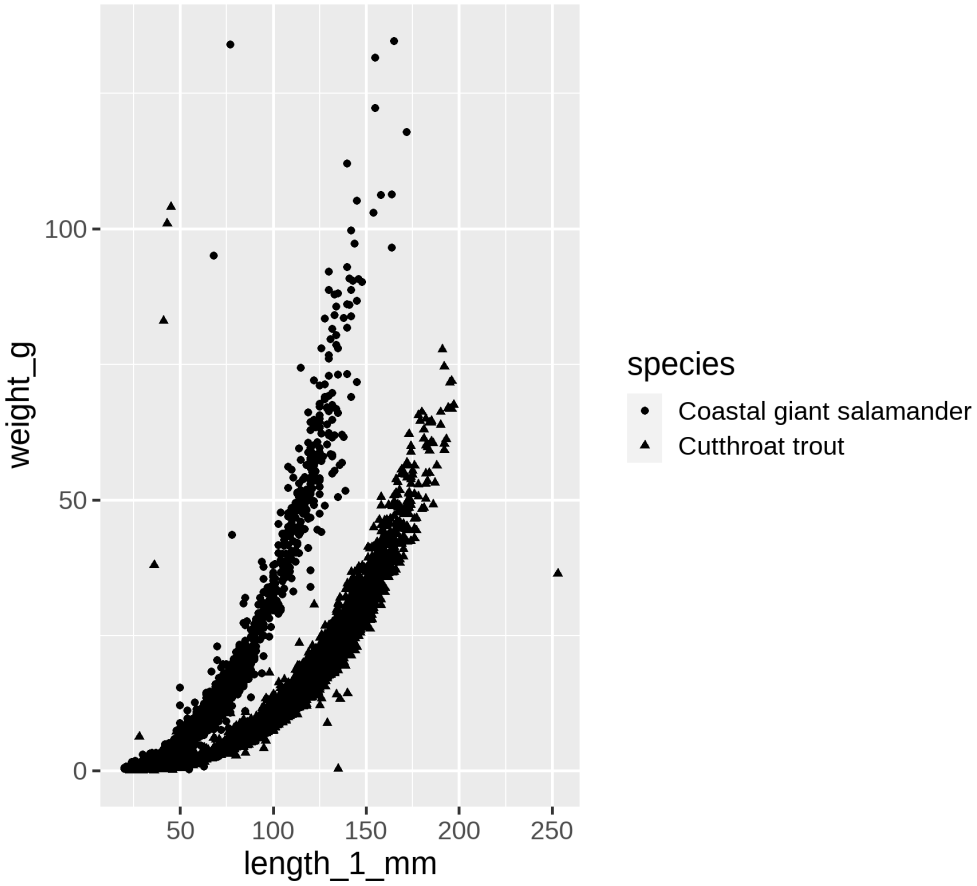
```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           size = species)) +
  geom_point()
```



`aes(shape)`: mapping shape to a variable

We can do the same with shape:

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           shape = species)) +
  geom_point()
```

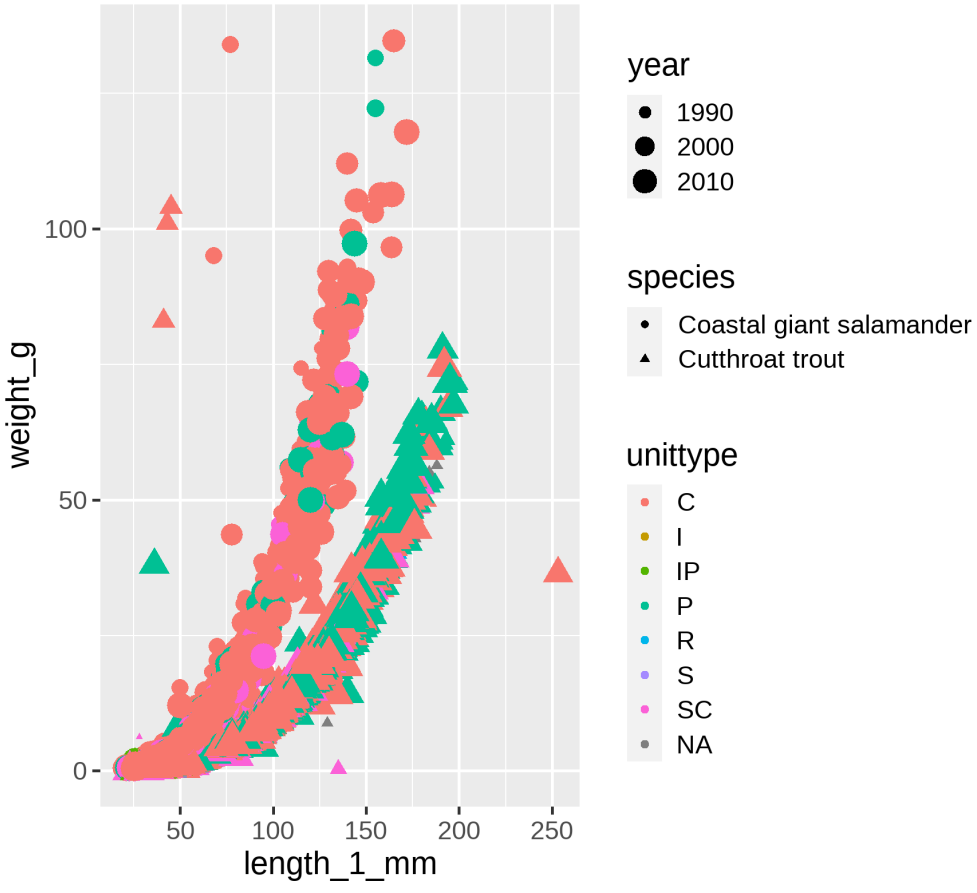


Combine color, size and shape

We can also combine these aesthetics and map different variables

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           color = unittype,
           shape = species,
           size = year)) +
  geom_point()
```

- This is a bit too much for this plot, but sometimes can be useful

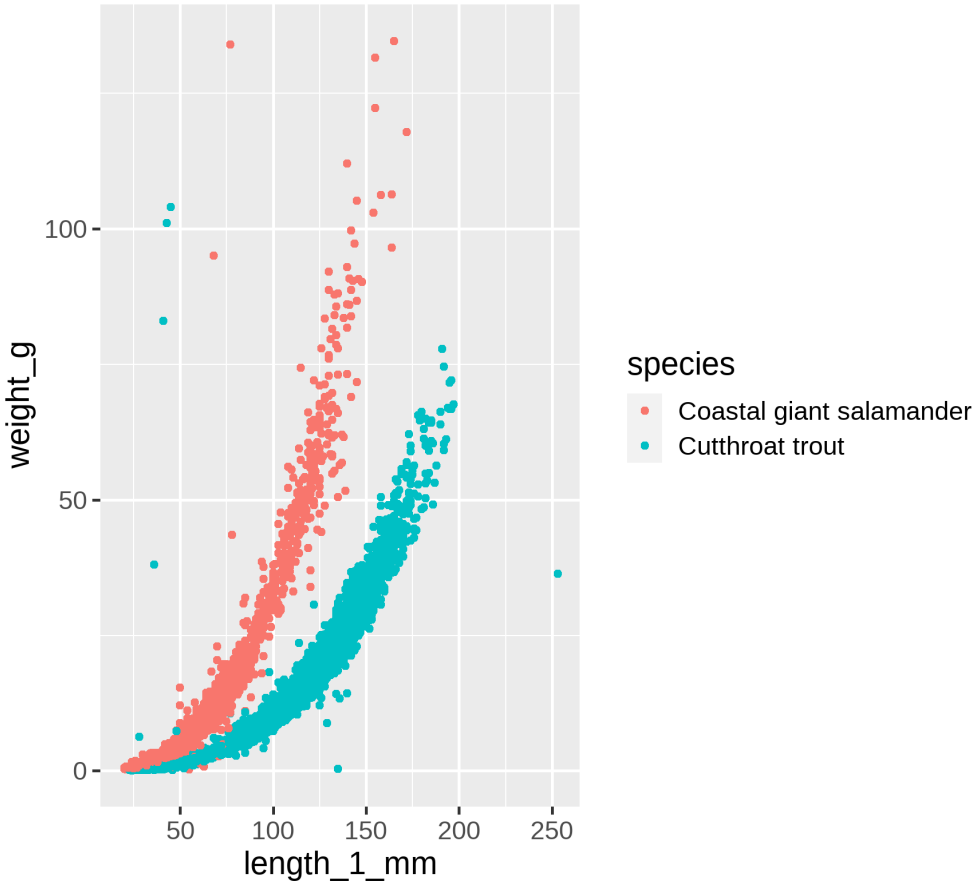


Changing the scales of the aesthetics

The scales onto which the aesthetic elements are mapped can be changed.

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           color = species)) +
  geom_point()
```

- Exponential relationship?
- How does it look like on the log scale?



scale_x_log10()

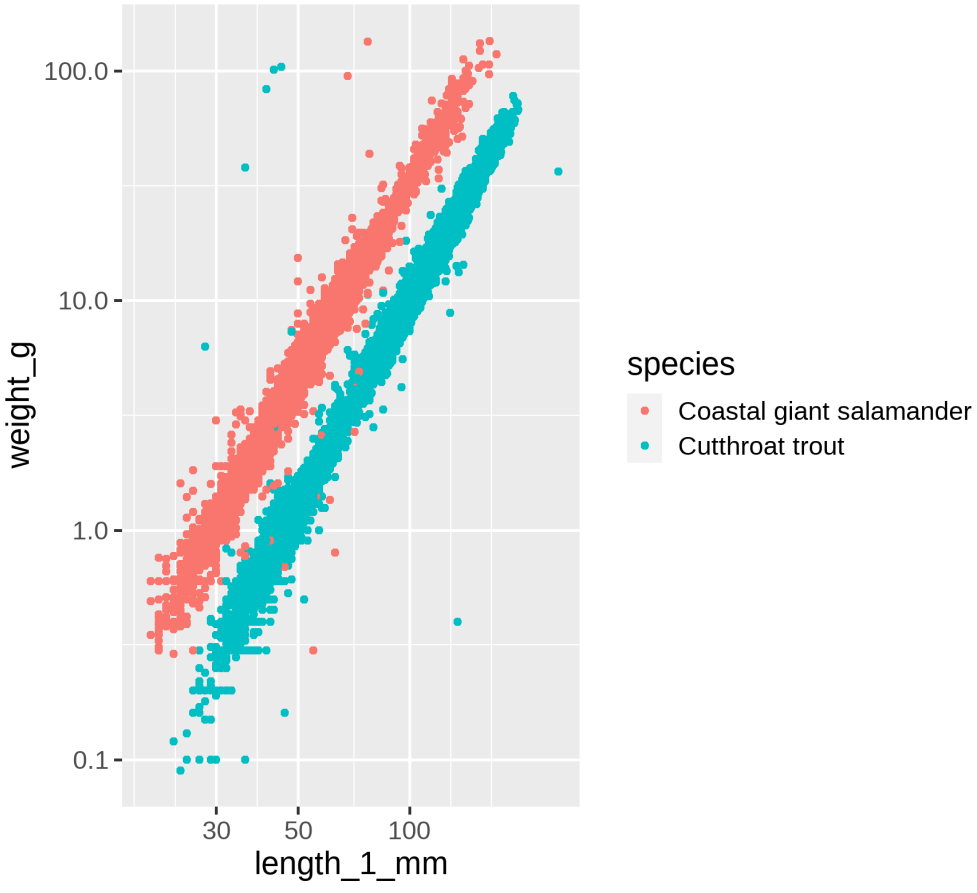
The scales onto which the aesthetic elements are mapped can be changed.

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           color = species)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```

- Scales can be changed for all elements of `aes`
- The general format of scale functions are:

scale_aes-name_scale-type

In this example we scale the x and the y aesthetic to log10.

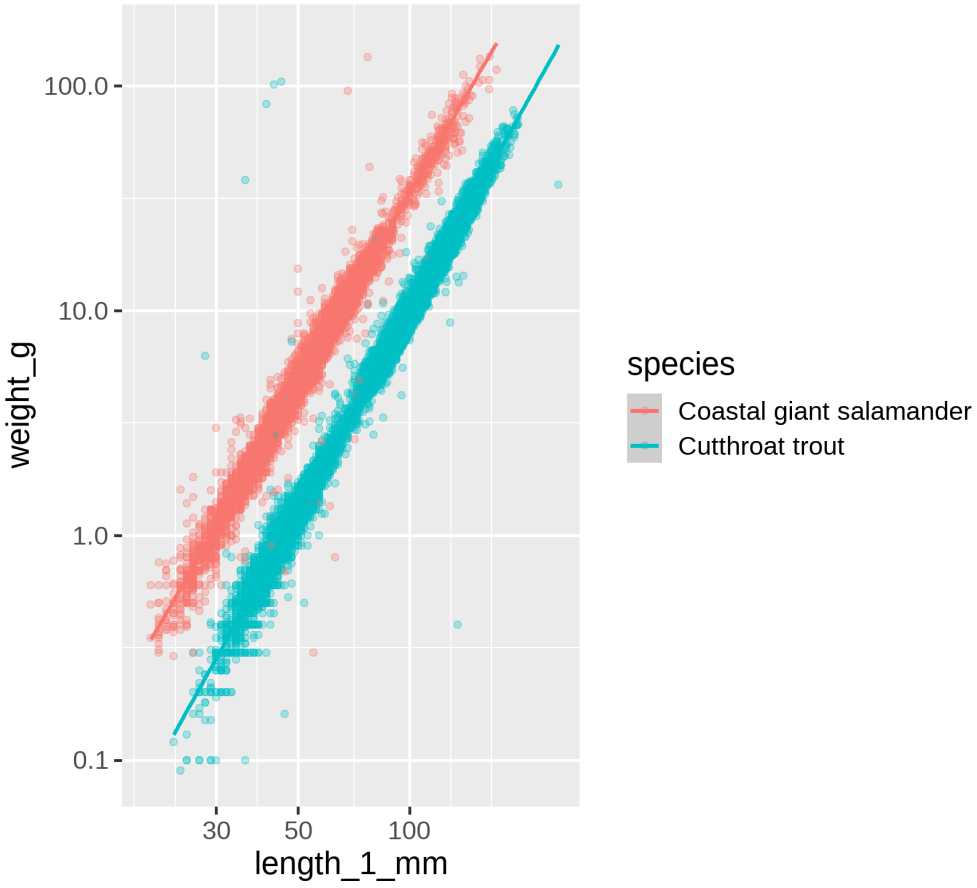


geom_smooth()

Add a smoothing line that helps see patterns in the data

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           color = species)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm") +
  scale_x_log10() +
  scale_y_log10()
```

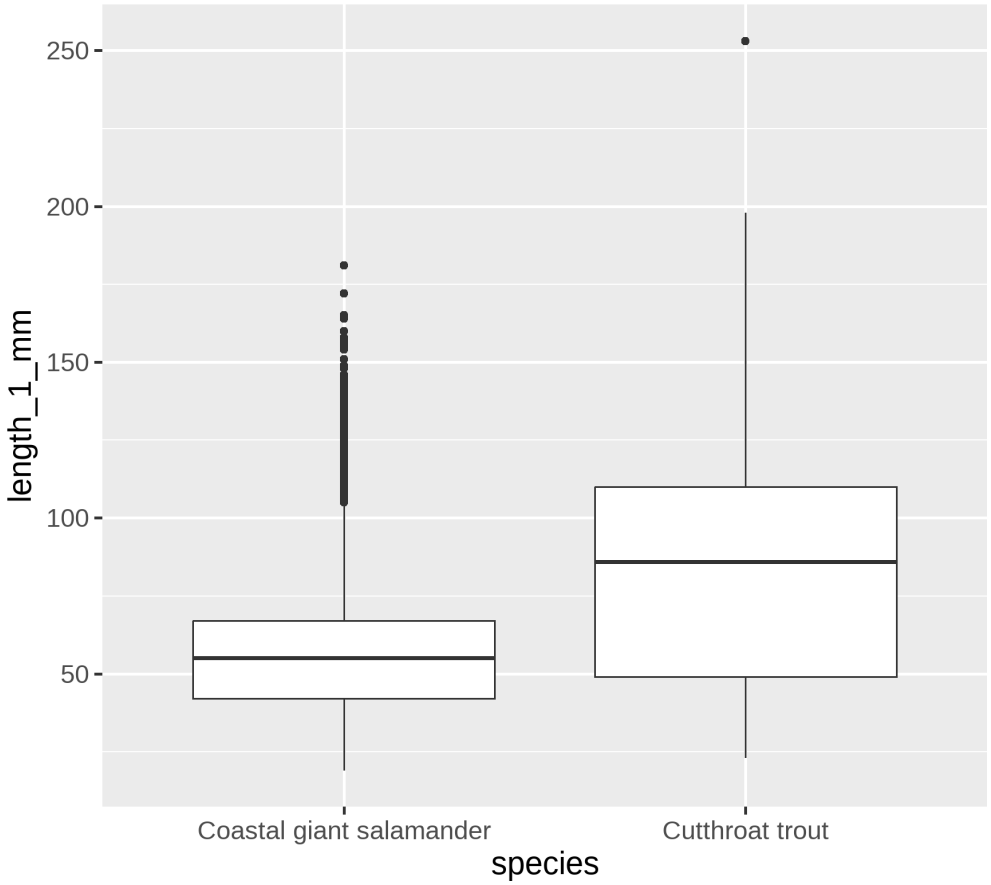
- With `method = "lm"`, a linear regression line is added
- All geoms are done separately for the species
  - Color is defined globally
- alpha makes the points transparent (0-1 with 0 being invisible)



geom_boxplot

Compare groups using a boxplot

```
ggplot(and_vertebrates,
       aes(x = species,
           y = length_1_mm)) +
  geom_boxplot()
```

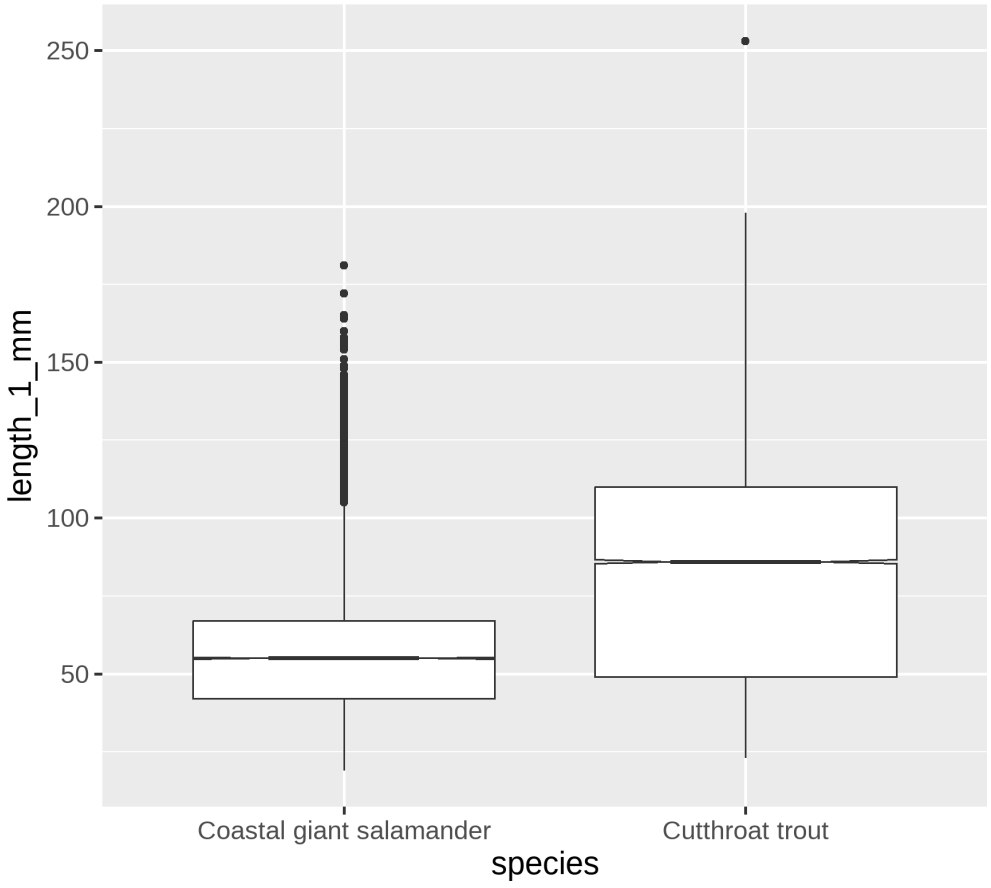


geom_boxplot

Compare groups using a boxplot

```
ggplot(and_vertebrates,
       aes(x = species,
           y = length_1_mm)) +
  geom_boxplot(notch = TRUE)
```

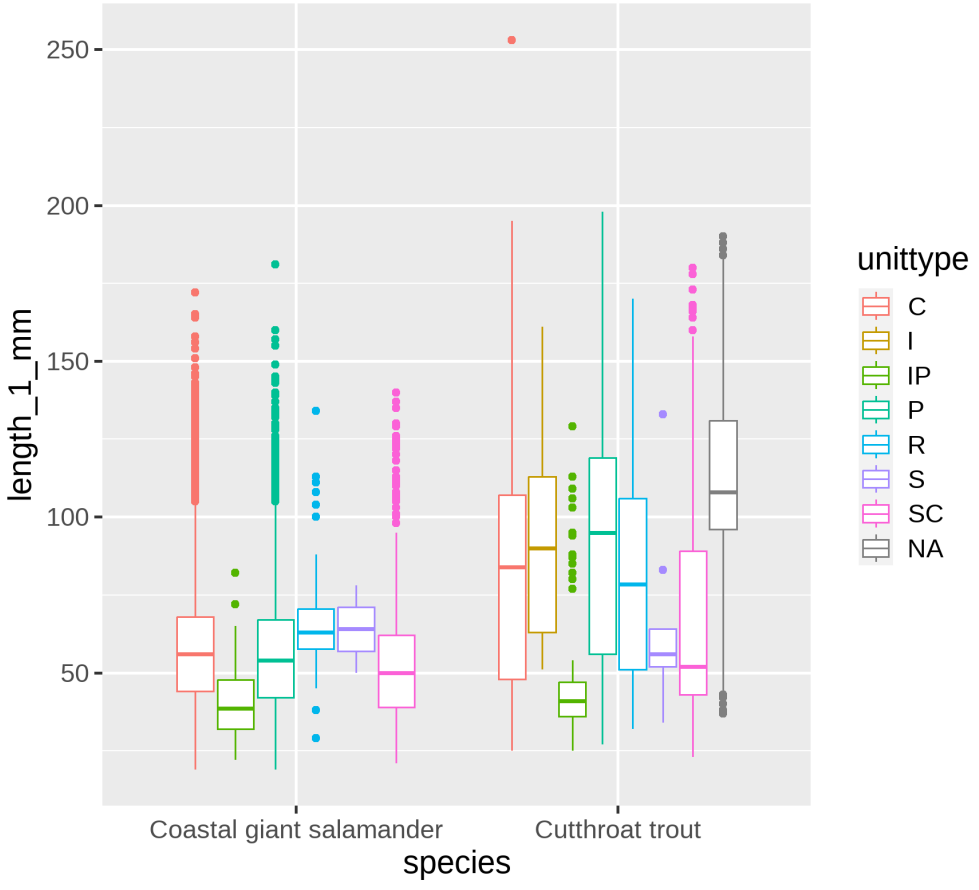
- If notches don't overlap, the medians of the groups are likely different



geom_boxplot

Map the `unittype` to the **color** aesthetic of the box

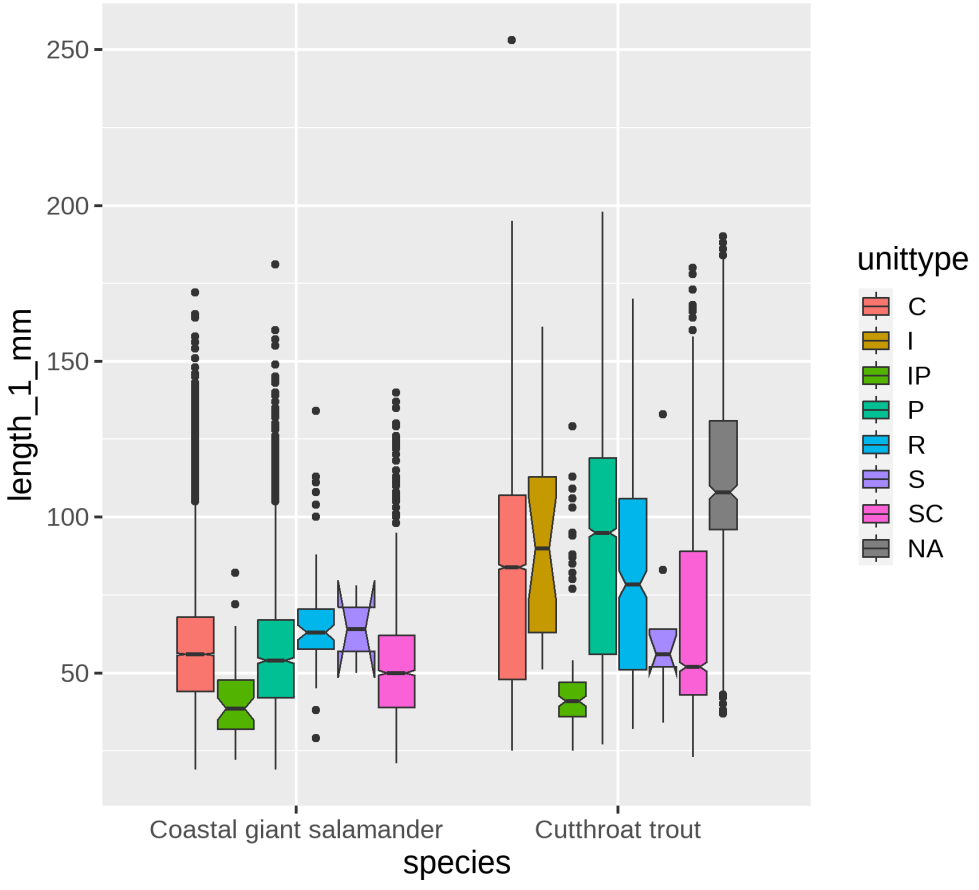
```
ggplot(and_vertebrates,
       aes(x = species,
           y = length_1_mm,
           color = unittype)) +
  geom_boxplot()
```



geom_boxplot

Map the `unittype` to the **fill** aesthetic of the box

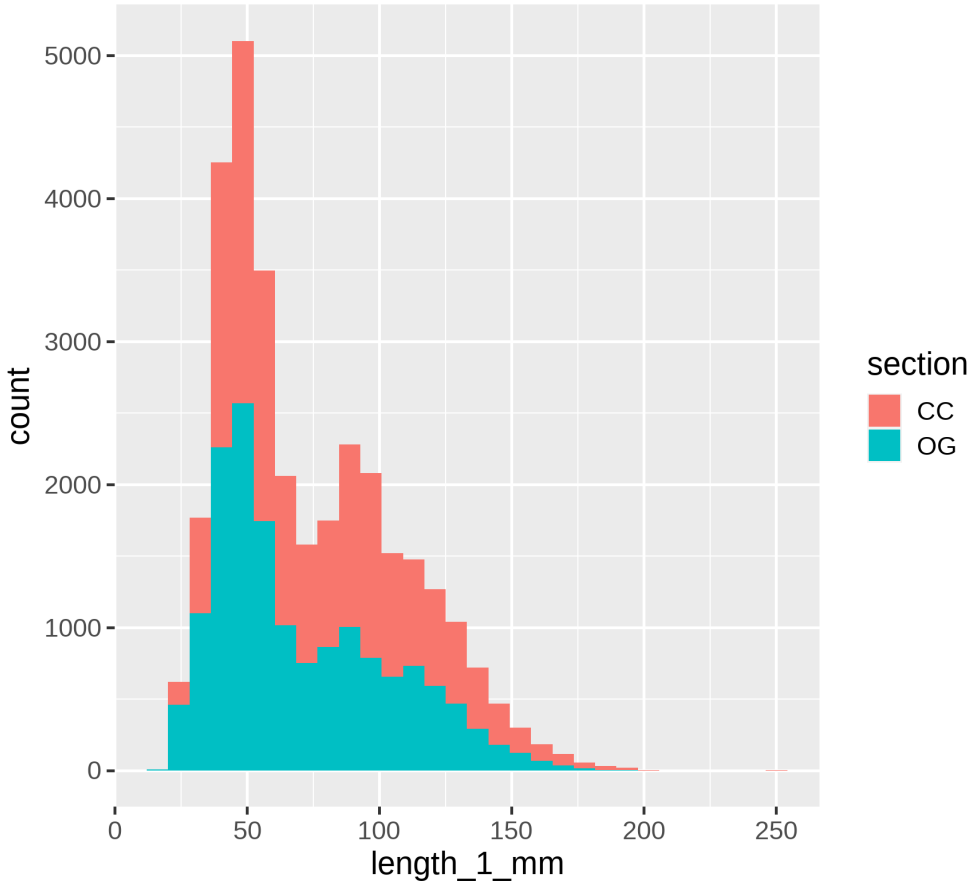
```
ggplot(and_vertebrates,
       aes(x = species,
           y = length_1_mm,
           fill = unittype)) +
  geom_boxplot(notch = TRUE)
```



geom_histogram

```
ggplot(and_vertebrates,
       aes(x = length_1_mm,
           fill = section)) +
  geom_histogram()
```

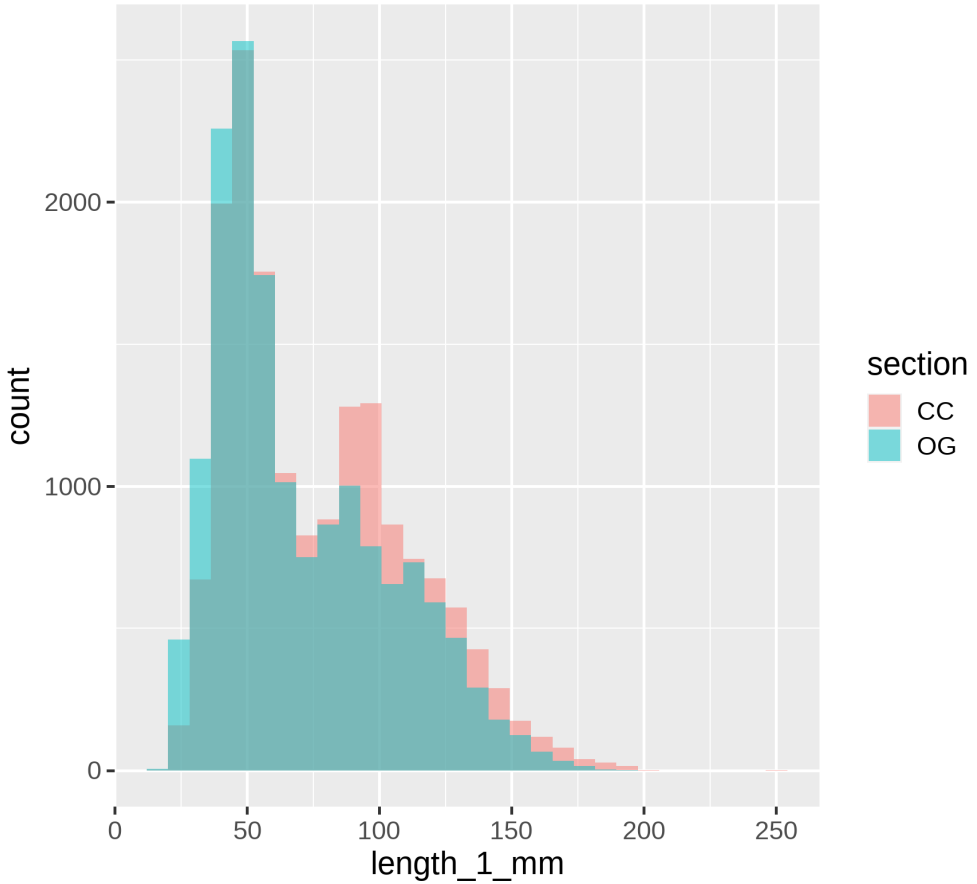
- Careful: By default the histogram is stacked for the different groups!



geom_histogram

```
ggplot(and_vertebrates,
       aes(x = length_1_mm,
           fill = section)) +
  geom_histogram(
    position = "identity",
    alpha = 0.5)
```

- Change the position of the histogram to `"identity"`, if you don't want it stacked
- `alpha` makes sure that you see overlapping areas

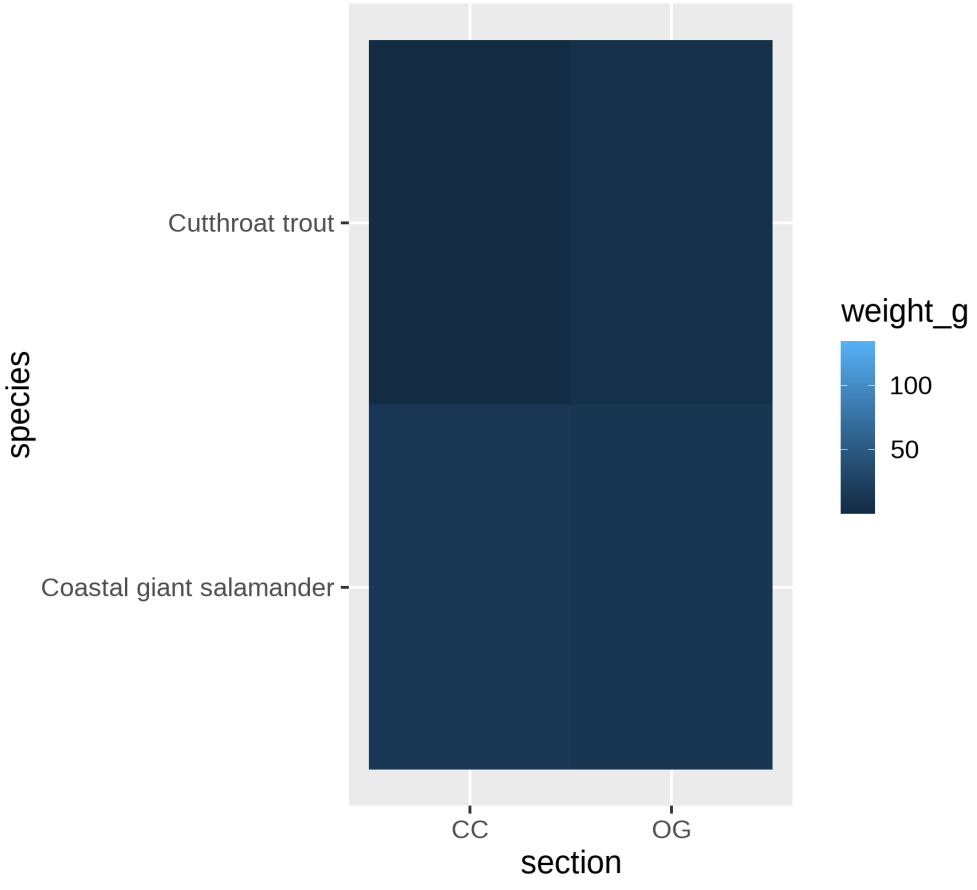


geom_tile

You can create a simple heatmap with `geom_tile`

```
ggplot(and_vertebrates,
       aes(x = section,
           y = species,
           fill = weight_g)) +
  geom_tile()
```

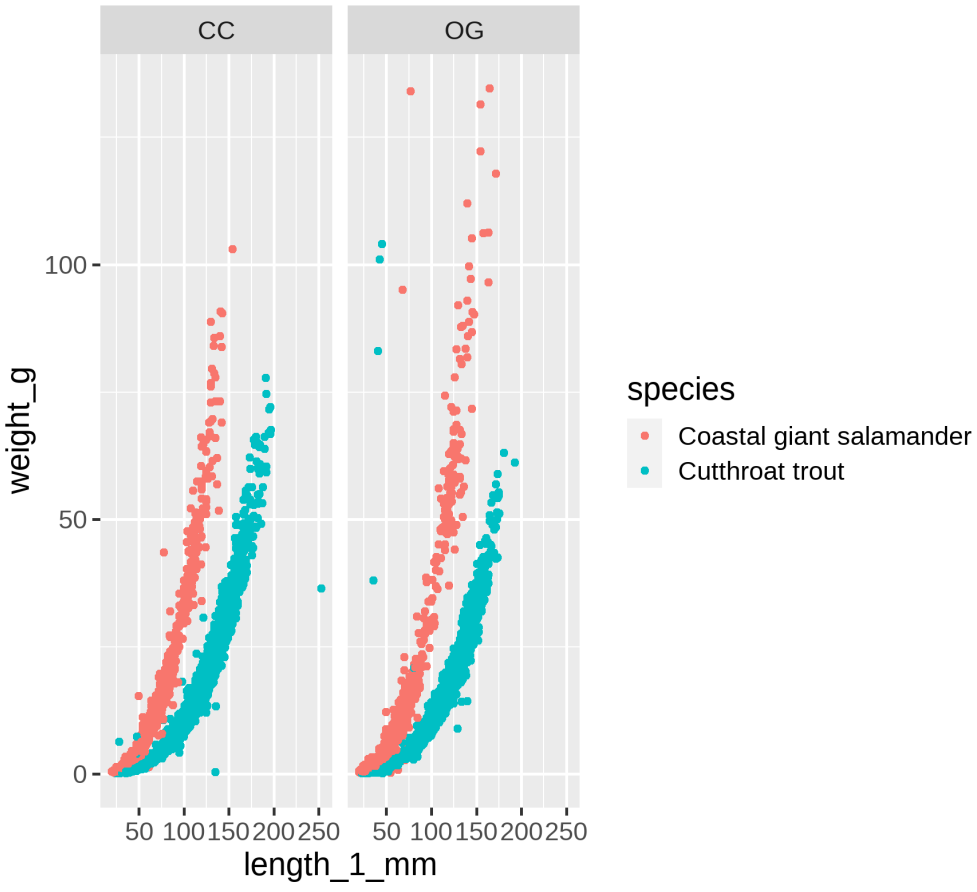
- Here we would have to chose a different color scheme to see differences



Small multiples with `facet_wrap`

Split your plots along one variable with `facet_wrap`

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           color = species)) +
  geom_point() +
  facet_wrap(~section)
```

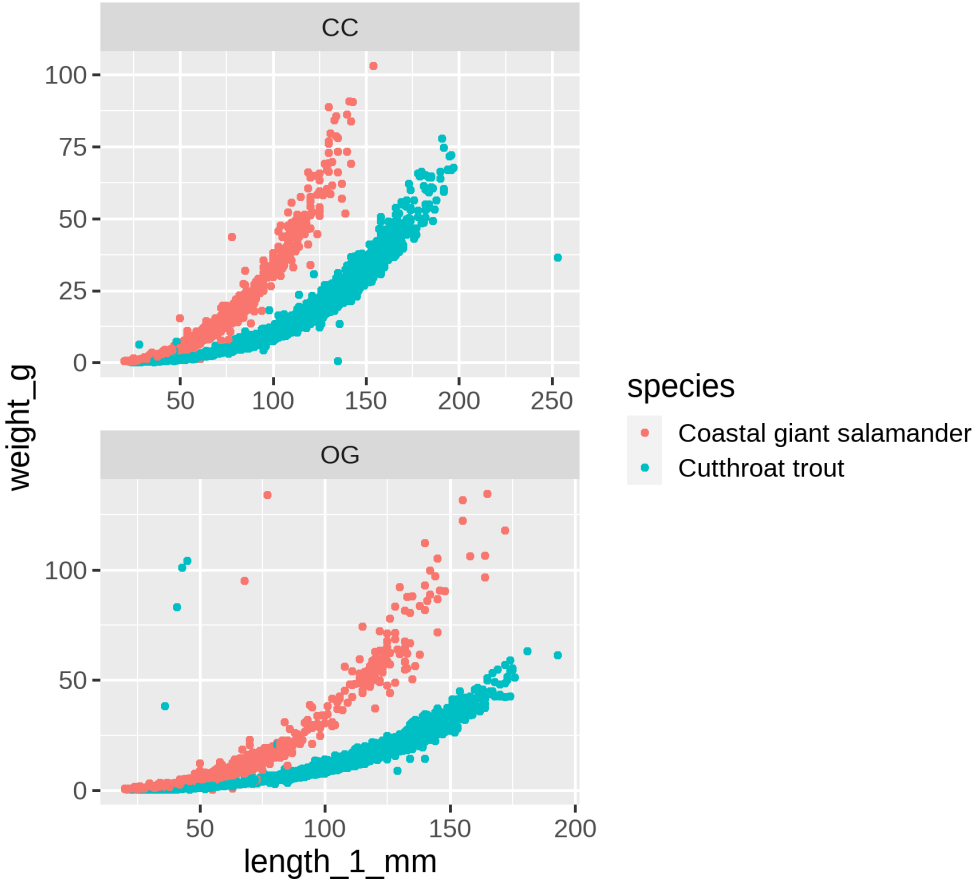


Small multiples with `facet_wrap`

Split your plots along one variable with `facet_wrap`

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           color = species)) +
  geom_point() +
  facet_wrap(~section,
             nrow = 2,
             scales = "free")
```

- Specify number of rows and columns with `nrow`/`ncol`
- Set separate scales for x and y with `scales`
  - `"free"`: x and y axis are free
  - `"free_y"`: only y-axis is free
  - `"free_x"`: only x-axis is free

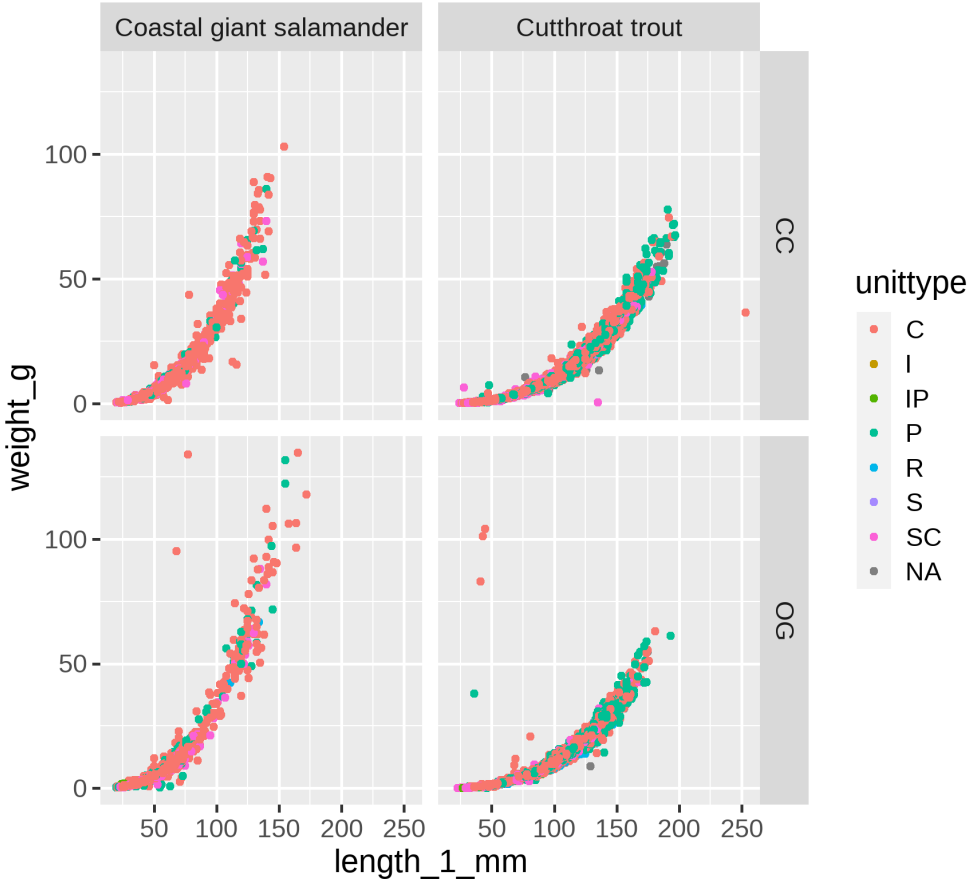


Small multiples with `facet_grid`

Split your plots along two variables with `facet_grid`

```
ggplot(data = and_vertebrates,
       aes(x = length_1_mm,
           y = weight_g,
           color = unittype)) +
  geom_point() +
  facet_grid(section ~ species)
```

- `facet_grid(rows ~ columns)`



Now you

Task 1: Exploratory data analysis with the penguin data set (45 min)

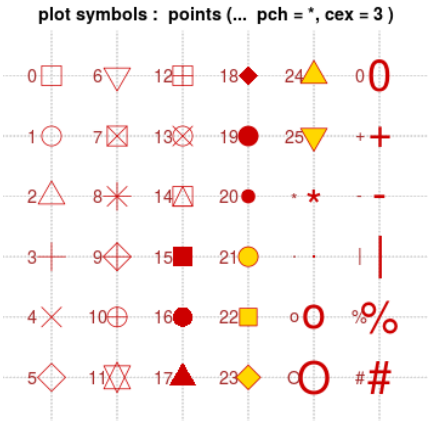
Find the task description (task 1.1-1.3) here

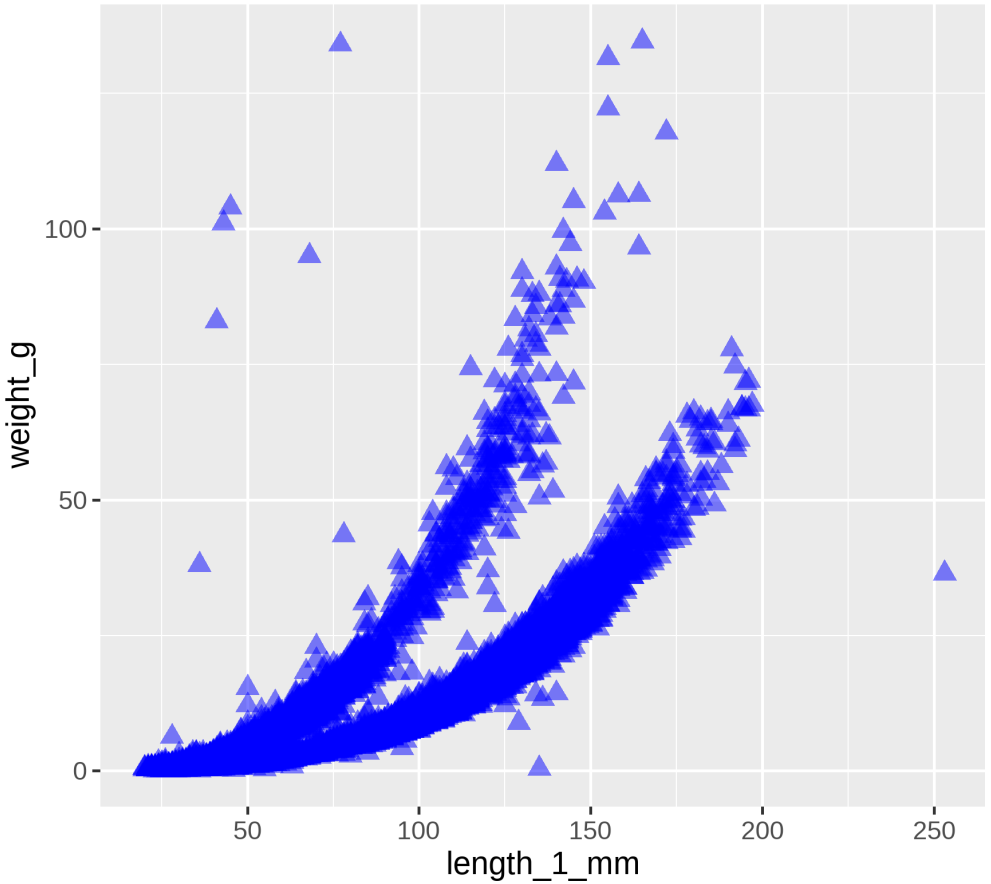

ggplot2:
Build a data
MASTERPIECE
THEMES
GEOMS
aes
fill
color
ggANIMATE
plotly
HORST '18

Change appearance of points

```
ggplot(and_vertebrates, aes(
  x = length_1_mm,
  y = weight_g
)) +
  geom_point(
    size = 4,
    shape = 17,
    color = "blue",
    alpha = 0.5
  )
```

- Have a look at shape and color codes:



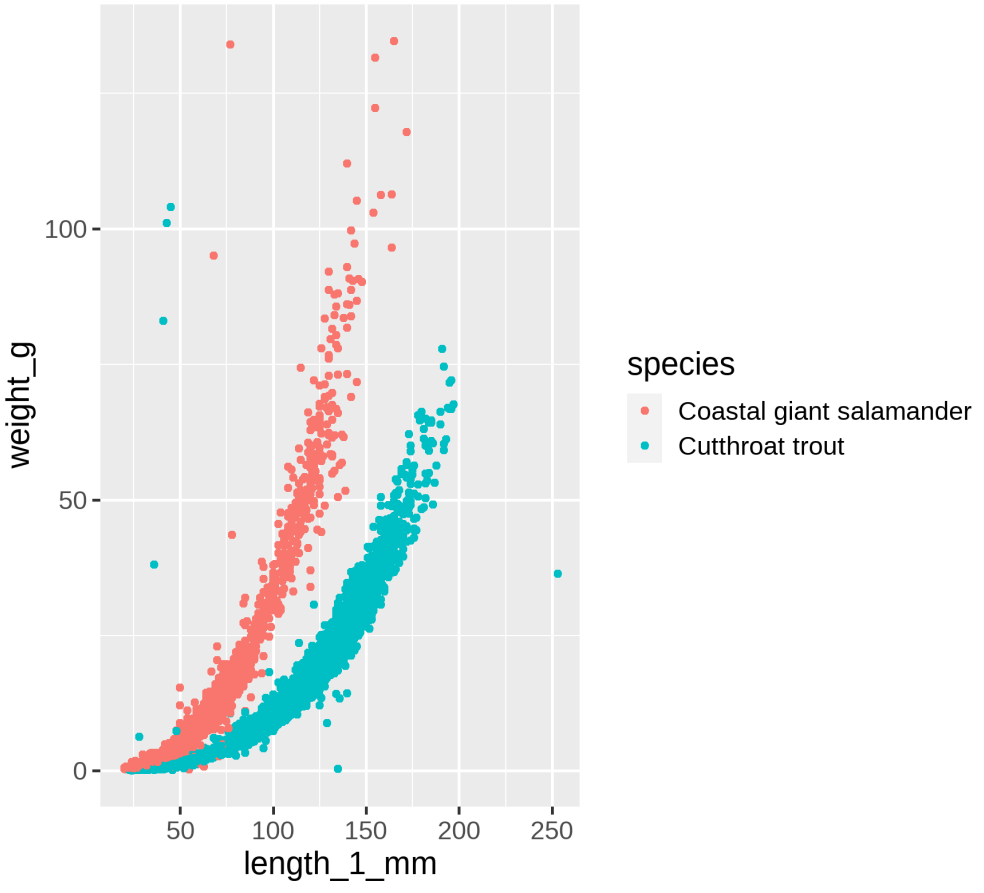


Change color scale

We can also save a plot in a variable

```
g <- ggplot(and_vertebrates, aes(
  x = length_1_mm,
  y = weight_g,
  color = species
)) +
  geom_point()
g
```

- Other plot layers can still be added to `g`

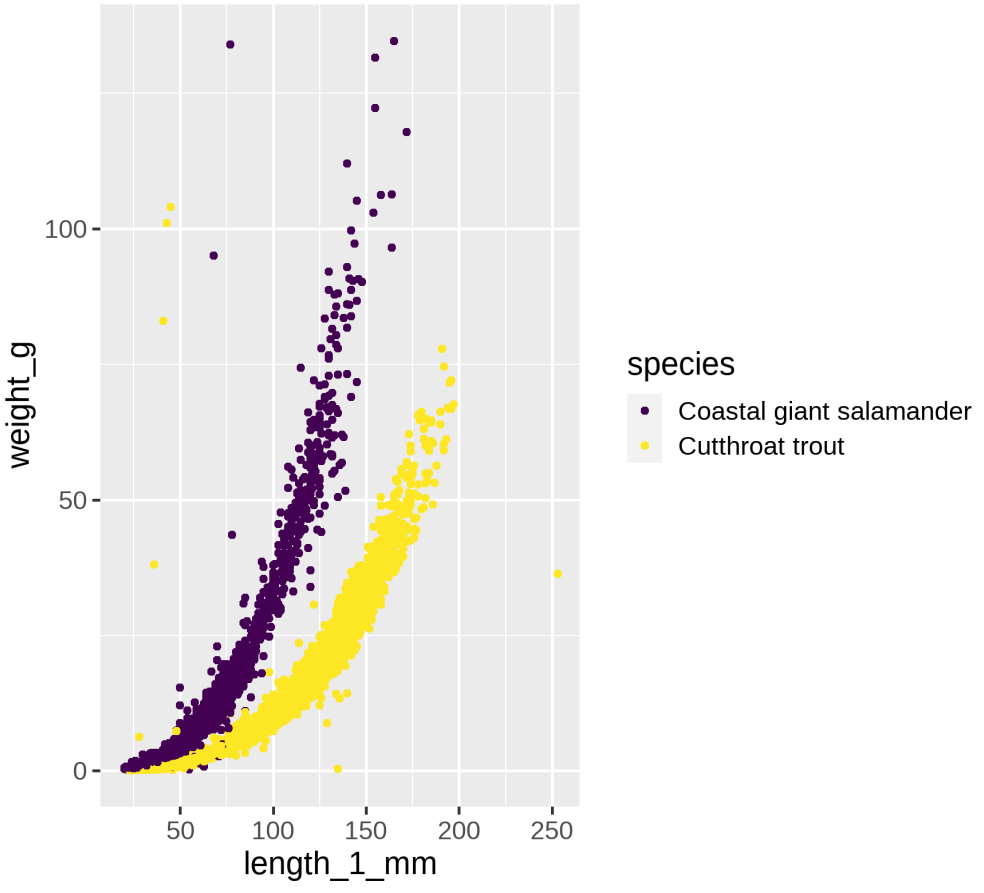


scale_color_viridis_d

Change the colors of the color aesthetic:

```
g +
  scale_color_viridis_d()
```

- The viridis color palette is designed for viewers with common forms of color blindness
- Different options of viridis color palettes are are:
  - `"magma"`, `"inferno"`, `"plasma"`, `"viridis"`, `"cividis"`
  - use the options with `scale_color_viridis_d(option = "cividis")`

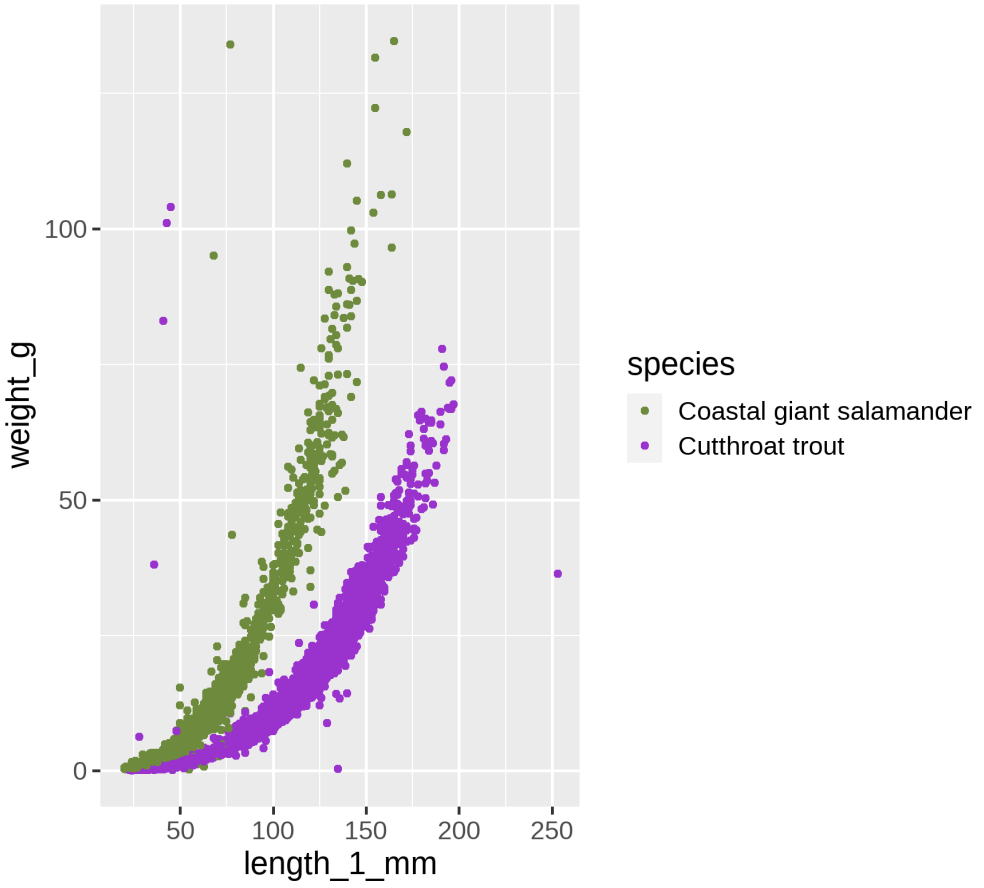


scale_color_manual

We can also manually specify colors:

```
g +
  scale_color_manual(values = c(
    "darkolivegreen4",
    "darkorchid3"
  ))
```

- Length of color vector has to match number of levels in your aesthetic
- Specify colors
  - Via their name (see here for all color names)
  - Via their Hex color codes (use websites to generate your own color palettes, e.g. here)

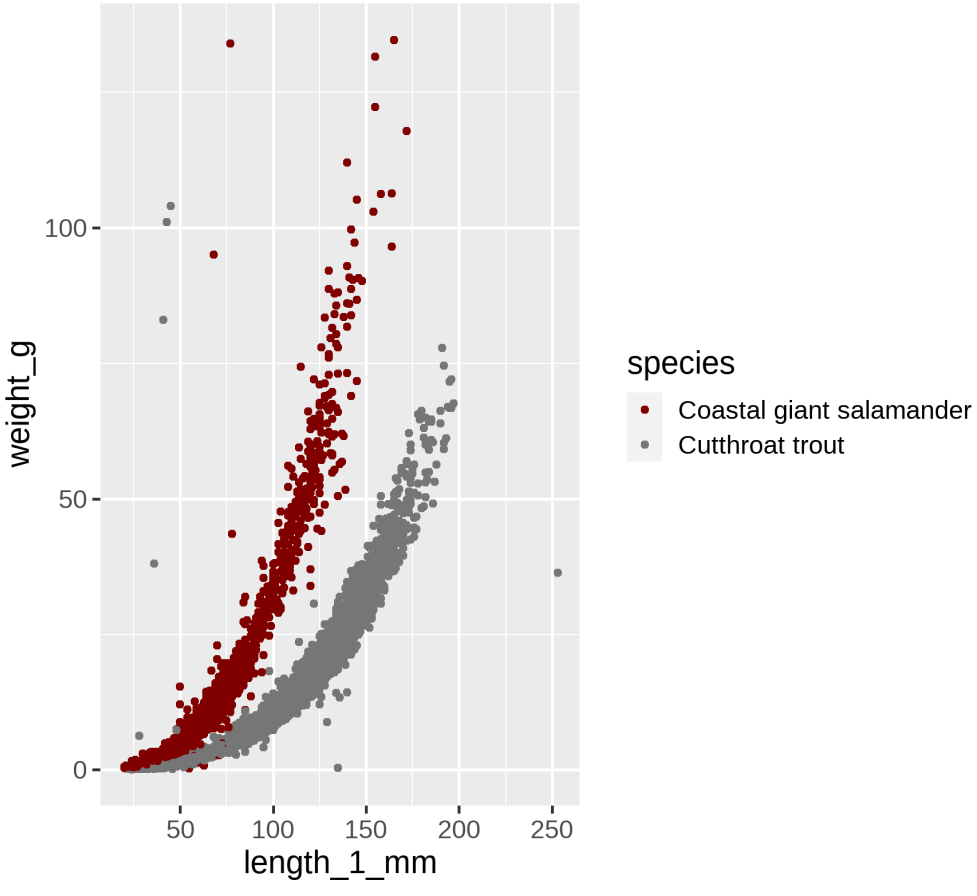


Other color scales

You can use the `paletteer` package to access color scales from many packages.

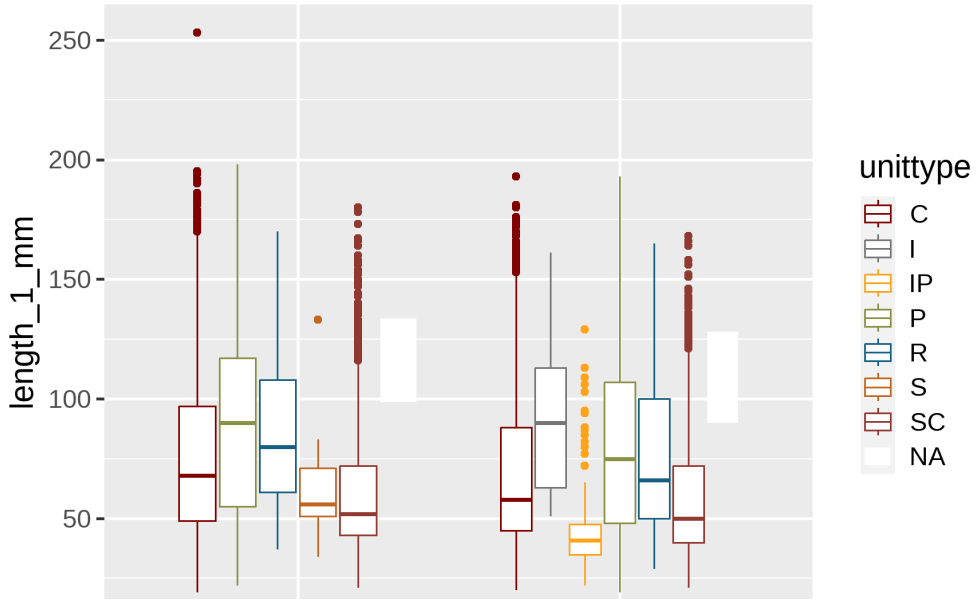
```
# install.packages("paletteer")
library(paletteer)
g <- g +
  scale_color_paletteer_d(
    palette = "ggsci::default_uchicago"
  )
g
```

- Use `scale_color_paletteer_d` for discrete and `scale_color_paletteer_c` for continuous color scales
- Check out all palettes available here

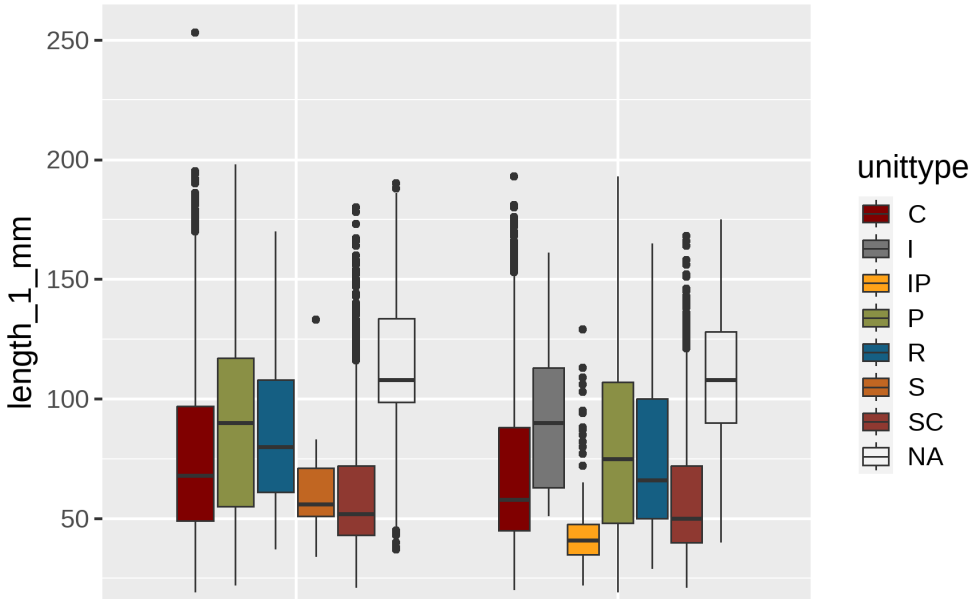


scale_fill_* vs. scale_color_*

```
ggplot(
  and_vertebrates,
  aes(
    x = section,
    y = length_1_mm,
    color = unittype )) +
  geom_boxplot() +
  paletteer::scale_color_paletteer_d(
    palette = "ggsci::default_uchicago"
  )
```

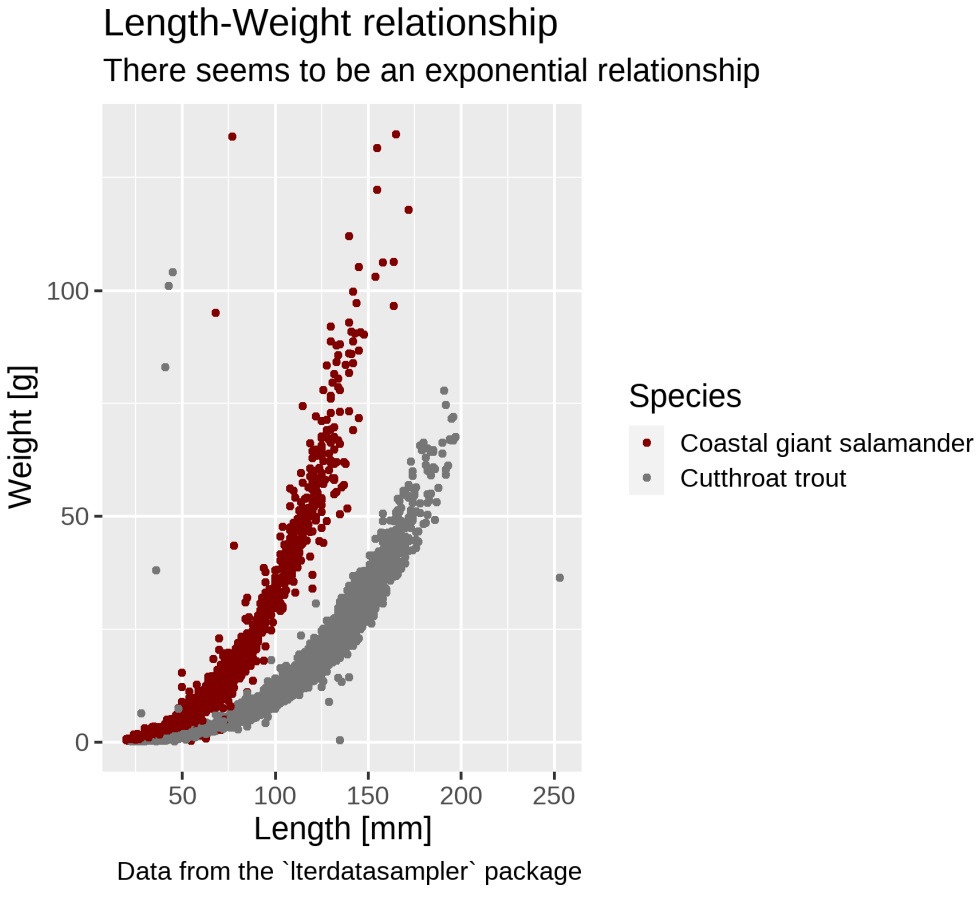



```
ggplot(
  and_vertebrates,
  aes(
    x = section,
    y = length_1_mm,
    fill = unittype )) +
  geom_boxplot() +
  paletteer::scale_fill_paletteer_d(
    palette = "ggsci::default_uchicago"
  )
```



`labs`: Change axis and legend titles and add plot title

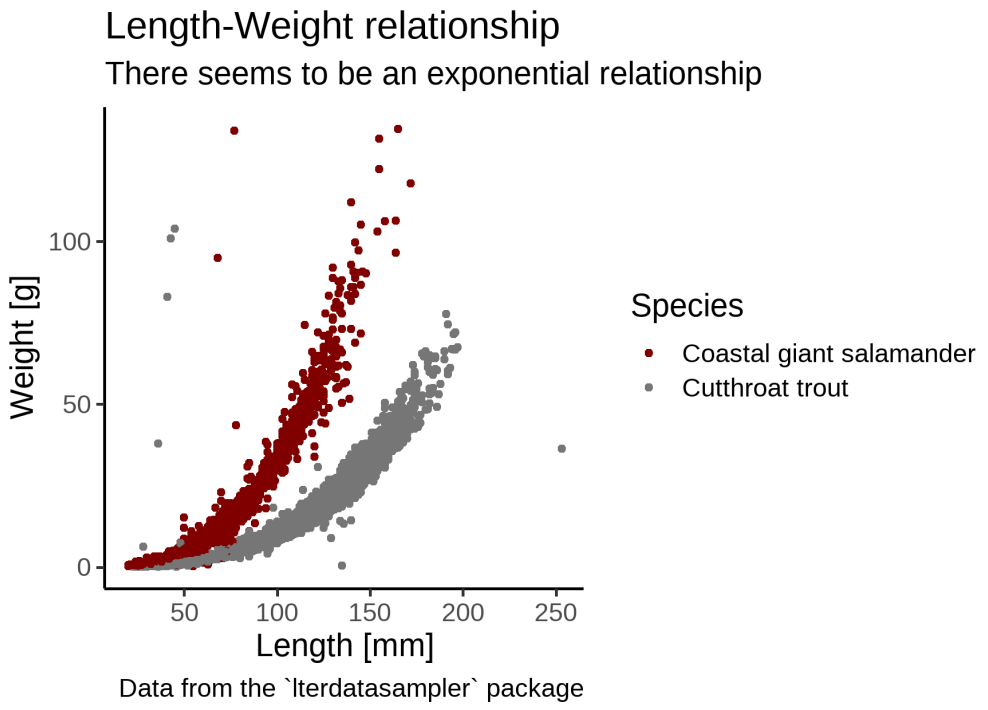
```
g <- g +
  labs(
    x = "Length [mm]",
     y = "Weight [g]",
     color = "Species",
     title = "Length-Weight relationship",
     subtitle = "There seems to be an
exponential relationship",
     caption = "Data from the `lterdatasampler`
package")
g
```



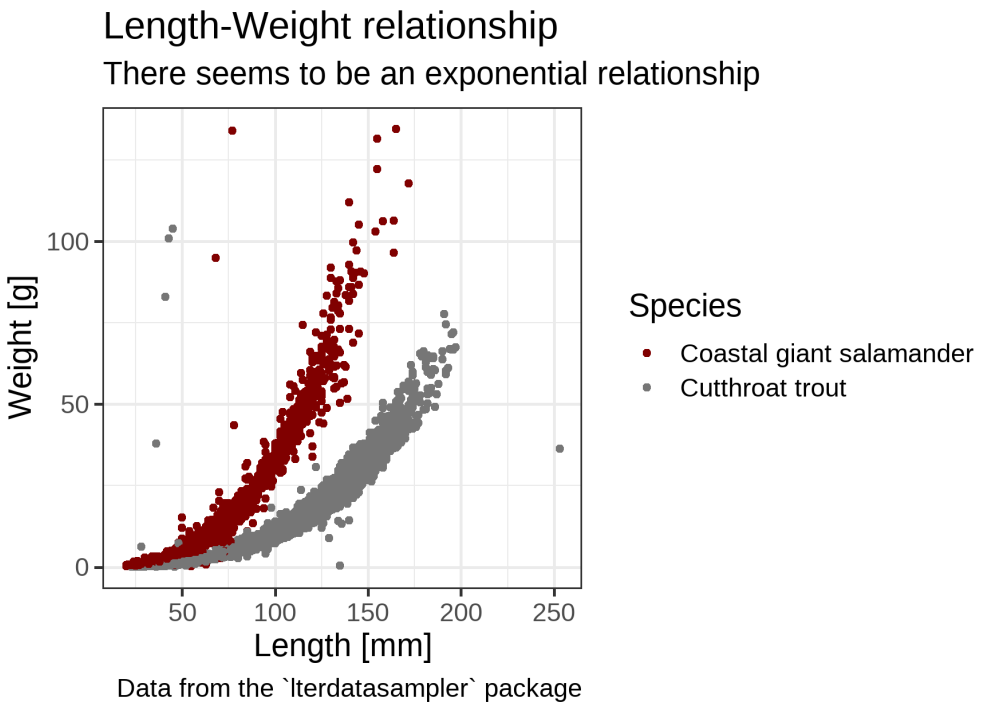
`theme_*`: change appearance

`ggplot2` offers many pre-defined themes that we can apply to change the appearance of a plot.

```
g +
  theme_classic()
```



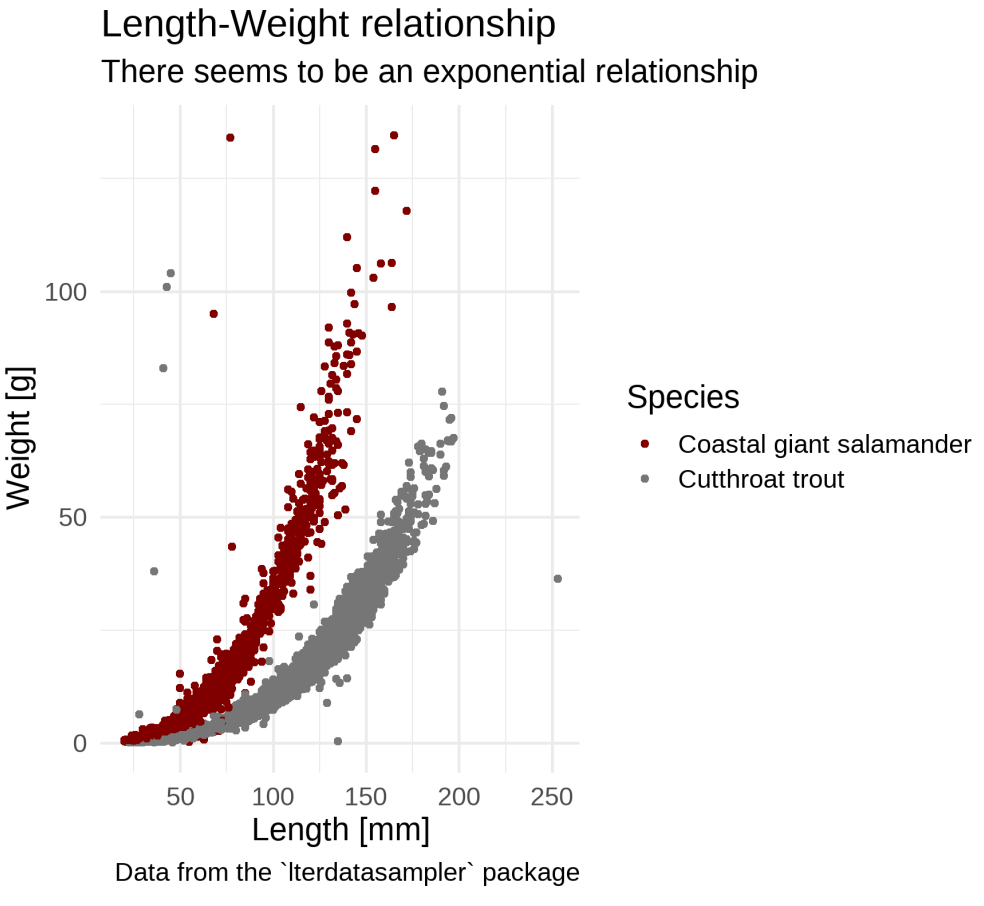

```
g +
  theme_bw()
```



`theme_*`: change appearance

`ggplot2` offers many pre-defined themes that we can apply to change the appearance of a plot.

```
g +
  theme_minimal()
```



`theme()`: customize theme

You can manually change a theme or even create an entire theme yourself. The elements you can control in the theme are:

- titles (plot, axis, legend, ...)
- labels
- background
- borders
- grid lines
- legends

If you want a full list of what you can customize, have a look at

```
?theme
```

- Look here for an overview of the elemtens that you can change and the corresponding functions

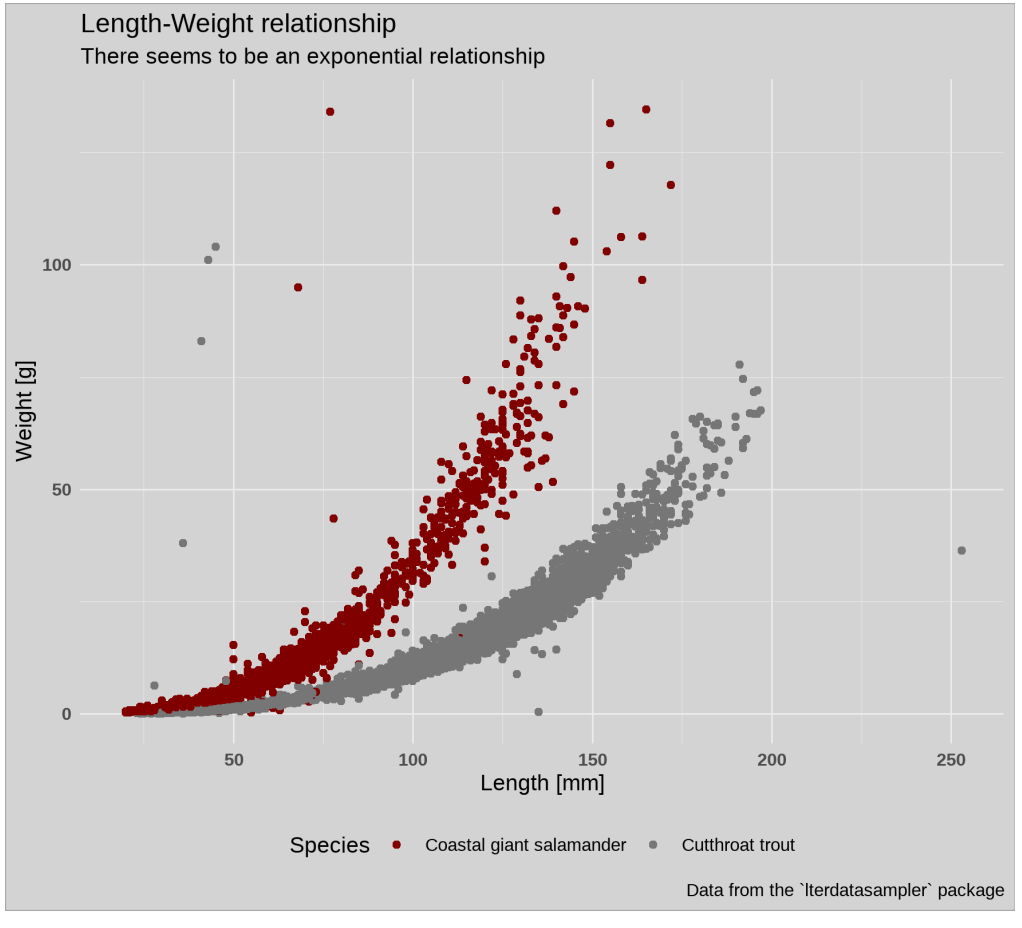
`theme()`: customize theme

To edit a theme, just add another `theme()` layer to your plot.

```
g +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    axis.text = element_text(face = "bold"),
    plot.background = element_rect(
      fill = "lightgrey",
      color = "darkgrey"
    )
  )
```

- The basic functioning is:

```
theme(
  element_name = element_function()
)
```



`theme_set()`: set global theme

You can set a global theme that will be applied to all ggplot objects in the current R session.

```
# Globally set theme_minimal as the default theme
theme_set(theme_minimal())
```

Add this to the beginning of your script.

You can also specify some defaults, e.g. the text size:

```
theme_set(theme_minimal(base_size = 16))
```

This is very practical if you want to achieve a consistent look, e.g. for a scientific journal.

ggsave()

A ggplot object can be saved on disk in different formats.

Without specifications:

```
# save plot g in img as my_plot.pdf
ggsave(filename = "img/my_plot.pdf", plot = g)
# save plot g in img as my_plot.png
ggsave(filename = "img/my_plot.png", plot = g)
```

Or with specifications:

```
# save a plot named g in the img directory under the name my_plot.png with width 16 cm and height
9 cm
ggsave(filename = "img/my_plot.png",
       plot = g,
       width = 16,
       heigth = 9,
       units = "cm")
```

Have a look at `?ggsave` to see all options.

Now you

Task 1: Make your penguin plots more beautiful (30 min)

Find the task description (task 1.4-1.5) here