

ICO Workshop R & RStudio

Part 4

Powerful visualisations with `ggplot2`

Sven De Maeyer & Tine van Daal

2nd - 4th July, 2024

1 / 71

Overview

1. Simple plots in R --- ([click here](#))
2. Grammar of graphics --- ([click here](#))
3. How `ggplot2` works (in a nutshell) --- ([click here](#))
4. Visualising a categorical variable --- ([click here](#))
5. Visualising a quantitative variable --- ([click here](#))
6. Visualising more than one variable --- ([click here](#))
7. More about visualisation? --- ([click here](#))

2 / 71

1. Simple plots in R

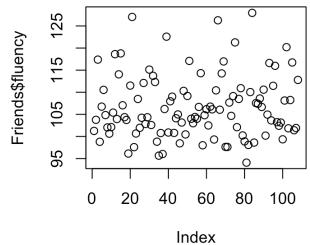
3 / 71

Plots in base

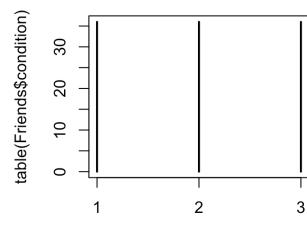
plot() boxplot() hist()

The generic function `plot()` "knows" what to do (plot) with the input it receives.

```
# one quantitative variable  
plot(Friends$fluency)
```



```
# one qualitative variables  
plot(table(Friends$condition))
```

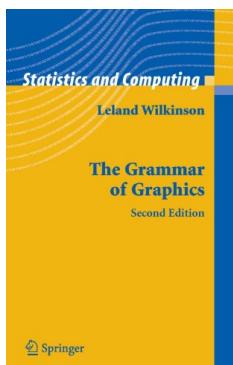


4 / 71

2. Grammar of Graphics

5 / 71

A more theoretical approach to visualisation



- Theoretical 'breakdown' of a visualisation into components (called layers)
- One system to create different visualisations
- At the heart of several modern graphical applications:
 - ggplot2
 - Tableau (Polaris)
 - Vega-Lite

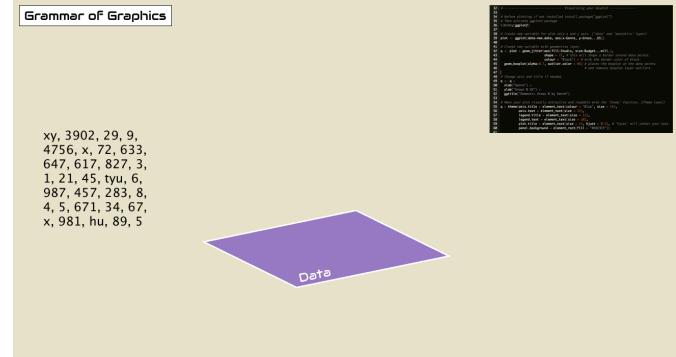
Slide taken from slide show by Thomas Lin Pedersen

6 / 71

Key idea behind the Grammar of Graphics

Layers of a visualisation:

data
aesthetics
geometries
facets
statistics
coordinates
themes



Animation by [Thomas de Beus](#)

7 / 71

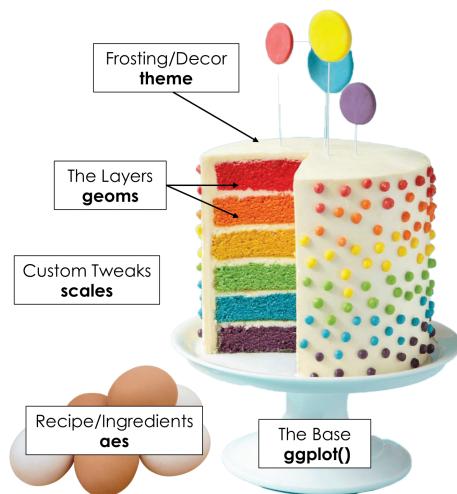
Grammar of Graphics is a bit like cake

Start by setting up the *foundation* with **ggplot()**

Specify *ingredients* (variables) with **aes()** and a *flavour* with **scales**

Create *layers* to plot with **geoms**

Style the cake with **theme**



Slide by [Tanya Shapiro](#)

8 / 71



Grammar of Graphics: data

- Data is not only raw data, but can also be the results from an analysis

group	country	gender	mean_age	SD_age	CI_lower	CI_upper
BE Male	BE	Male	39	11.0	17.44	60.56
BE Female	BE	Female	41	13.2	15.13	66.87
BE Other	BE	Other	36	8.2	19.93	52.07
NL Male	NL	Male	37	12.0	13.48	60.52
NL Female	NL	Female	36	14.0	8.56	63.44
NL Other	NL	Other	31	7.2	16.89	45.11

- Data has to be 'tidy'

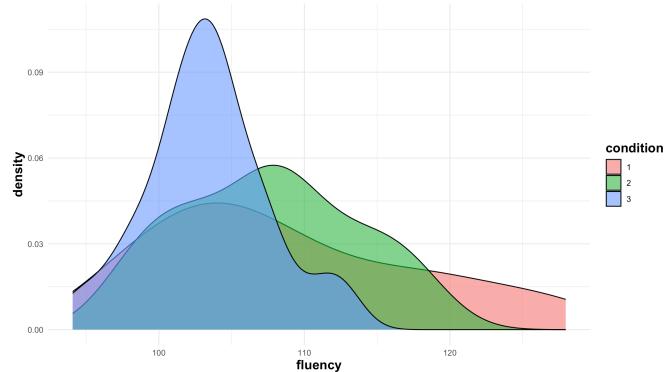
9 / 71



Grammar of Graphics: aesthetics

- Describe how variables in data are *mapped* to visual properties. For example:
 - variables mapped on x- and y-axis
 - variable that defines color or size of points
 - ...
- Change appearances of aesthetics using scales

Three aesthetics mapped: x-axis, y-axis and fill

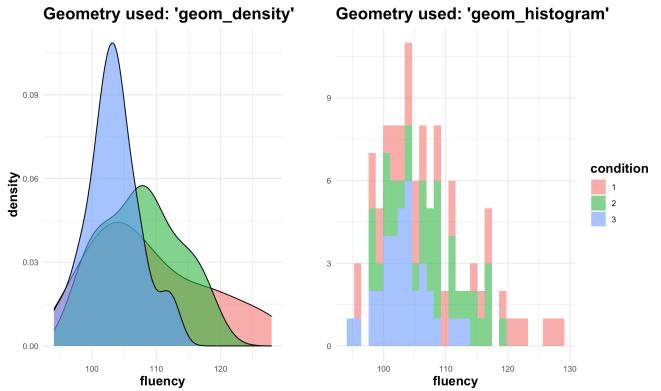


10 / 71



Grammar of Graphics: geometries

- Geometrical shapes at the heart of visualisation. For example:
 - boxplot
 - line
 - ...

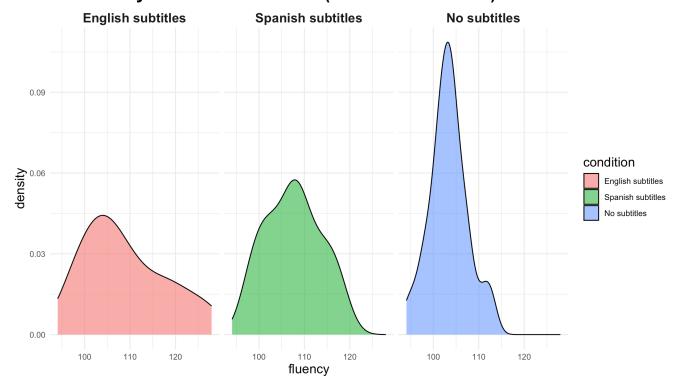


11 / 71

Grammar of Graphics: facets

- Also called 'small multiples'
- Define how much panels are shown and how they are arranged

Faceted by variable 'condition' (all facets on 1 row)



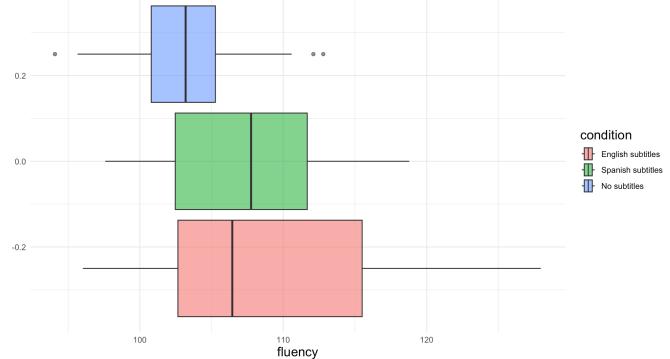
12 / 71



Grammar of Graphics: statistics

- Data might be `tidy`, but still in need of some statistical calculations. For example:
 - Calculate descriptive statistics to create a boxplot
 - Estimate a linear (or other) model to draw a regression line in a scatter plot
- Often implicit done by `ggplot2`

`'geom_boxplot()'` computes the necessary statistics in the background



13 / 71

3. How `ggplot2` works (in a nutshell)

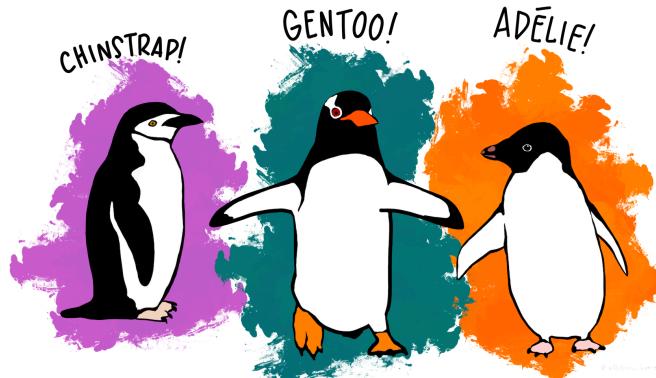
14 / 71



Time to get the penguins in...

Nice data set that can be used within R (Source: <https://allisonhorst.github.io/palmerpenguins/articles/intro.html>)

```
install.packages("palmerpenguins")
library(palmerpenguins)
data("penguins")
```



Artwork by @allison_horst

15 / 71

Time to get the penguins in...



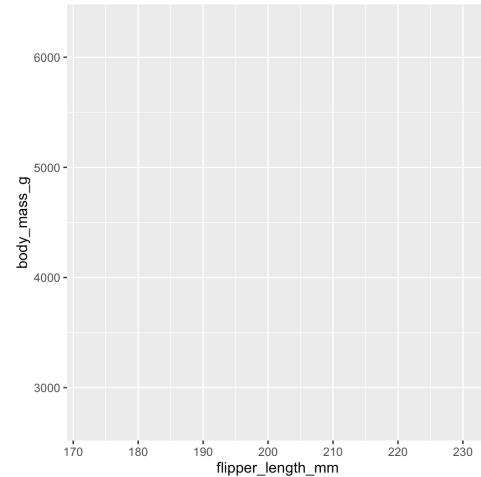
Table 1. Random sample of 10 observations from the Palmer Pinguins dataset

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Dream	41.1	17.5	190	3900	male	2009
Gentoo	Biscoe	44.0	13.6	208	4350	female	2008
Adelie	Dream	40.6	17.2	187	3475	male	2009
Gentoo	Biscoe	45.5	13.9	210	4200	female	2008
Chinstrap	Dream	52.7	19.8	197	3725	male	2007
Adelie	Torgersen	38.8	17.6	191	3275	female	2009
Chinstrap	Dream	45.2	17.8	198	3950	female	2007
Adelie	Dream	37.6	19.3	181	3300	female	2007
Adelie	Biscoe	36.5	16.6	181	2850	female	2008
Adelie	Torgersen	39.0	17.1	191	3050	female	2009

16 / 71

The basics of `ggplot2`: *data & aesthetics*

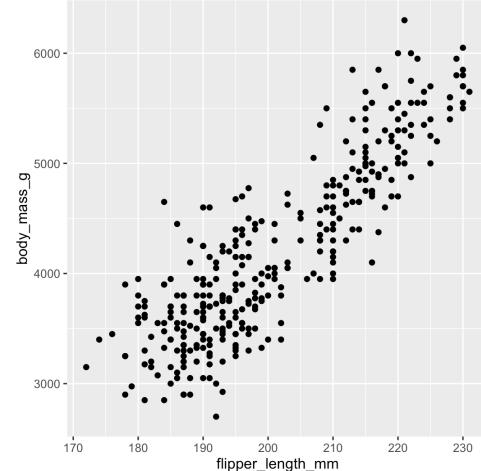
```
Plot <- ggplot(  
  ## Step 1: data  
  data = penguins,  
  ## Step 2: specify aesthetics (mapping)  
  aes(  
    x = flipper_length_mm,  
    y = body_mass_g  
)  
  
Plot
```



17 / 71

The basics of `ggplot2`: *geometry*

```
Plot <- ggplot(  
  ## Step 1: data  
  data = penguins,  
  ## Step 2: specify aesthetics (mapping)  
  aes(  
    x = flipper_length_mm,  
    y = body_mass_g  
) +  
  ## Step 3: add geometry  
  geom_point()  
  
Plot
```



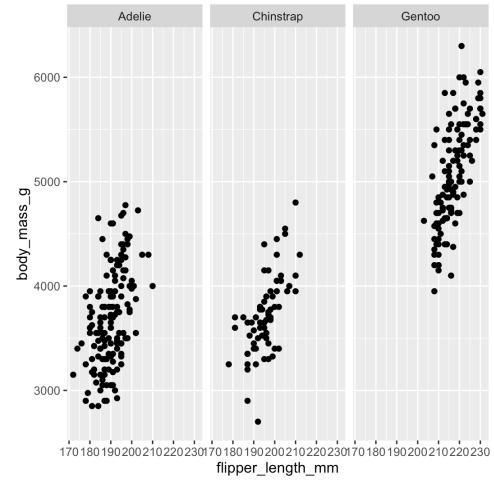
Not every component of the *Grammar of Graphics* needs to be defined. The other components have *default* values that are automatically applied.

18 / 71

The basics of ggplot2: facets

```
Plot <- ggplot(  
  ## Step 1: data  
  data = penguins,  
  
  ## Step 2: specify aesthetics (mapping)  
  aes(  
    x = flipper_length_mm,  
    y = body_mass_g  
) +  
  ## Step 3: add geometry  
  geom_point() +  
  
  ## Step 4: define facets  
  facet_wrap(~species)
```

Plot

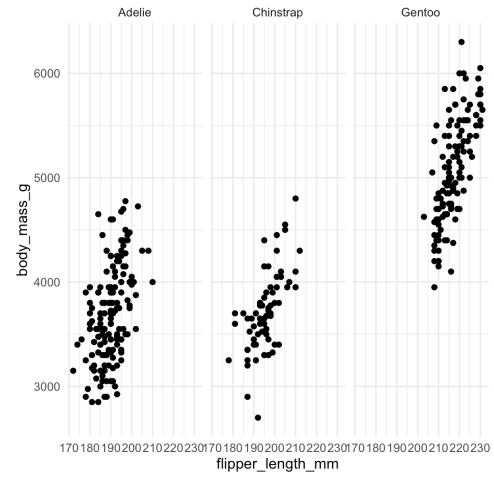


19 / 71

The basics of ggplot2: theme

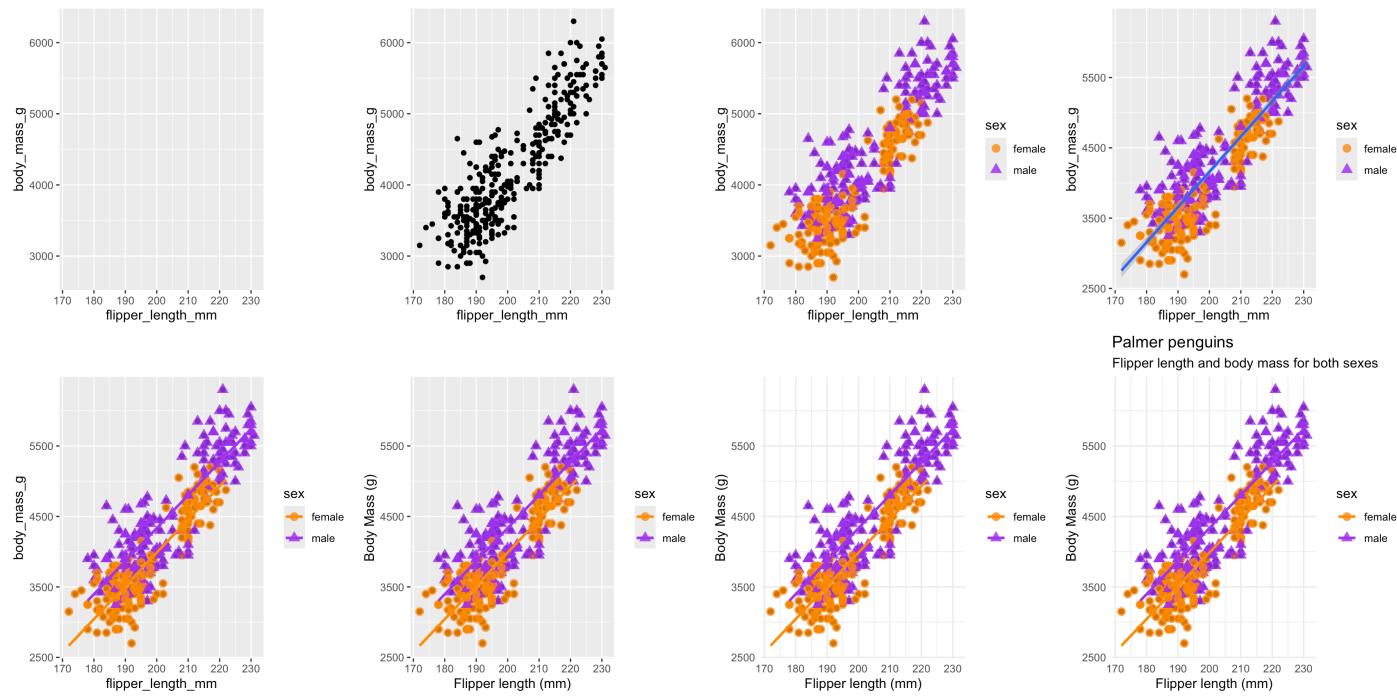
```
Plot <- ggplot(  
  ## Step 1: data  
  data = penguins,  
  
  ## Step 2: specify aesthetics (mapping)  
  aes(  
    x = flipper_length_mm,  
    y = body_mass_g  
) +  
  ## Step 3: add geometry  
  geom_point() +  
  
  ## Step 4: define facets  
  facet_wrap(~species) +  
  
  ## Step 5: set theme  
  theme_minimal()
```

Plot



20 / 71

Building a visualisation by adding layers ...



21 / 71

Several geom_* options...

Data visualization with ggplot2 :: CHEAT SHEET

This cheat sheet provides a quick reference for ggplot2's geom functions, organized into categories:

- Basics:** Explains how ggplot2 uses the grammar of graphics and provides a template for building plots.
- Geoms:** A large section detailing over 20 geom functions, each with a diagram and a brief description of its purpose.
- Graphical Primitives:** Includes geom_abline, geom_annotate, geom_bar, geom_boxplot, geom_hex, geom_label, geom_line, geom_rect, geom_rug, geom_ribbon, geom_smooth, geom_text, and geom_vline.
- Coordinate Functions:** Includes geom_map, geom_sf, and geom_sf_hex.
- Statistical Functions:** Includes geom_area, geom_bar, geom_boxplot, geom_dotplot, geom_hex, geom_linerange, geom_pointrange, and geom_violin.
- Theming Functions:** Includes theme_* functions.
- Other:** Includes a section on maps and a legend for common aesthetic values.

The cheatsheet: <https://github.com/rstudio/cheatsheets/blob/main/data-visualization-2.1.pdf>

22 / 71

4. Visualising a categorical variable

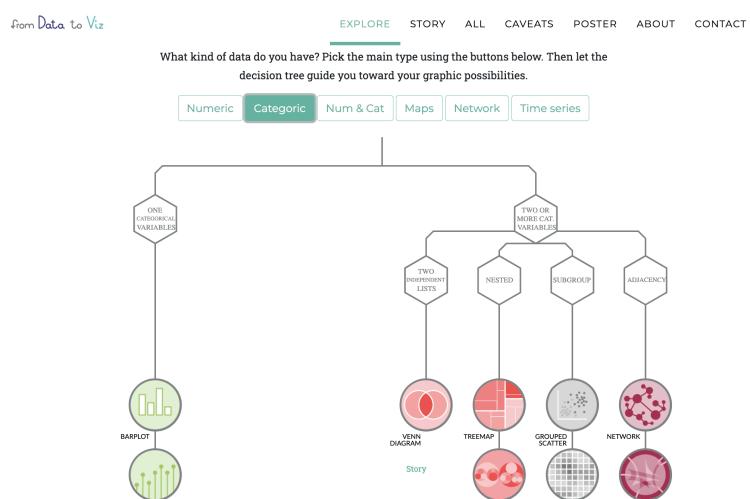
"The bar is open... Let's have a lollipop?"

23 / 71

Visualising a categorical variable?

There are many ways
to visualise a
categorical variable...

Can you think of
some?

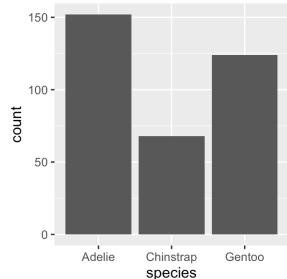


Have a look at [data to viz.com](http://data-to-viz.com)

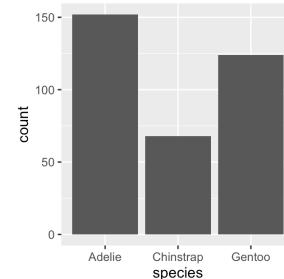
24 / 71

Creating a barplot with `geom_bar()` or `geom_col()`

`geom_bar()`



`geom_col()`



```
ggplot(  
  penguins,  
  aes(  
    x = species  
  )  
) +  
  geom_bar()  
  
# stat_count() does the counting automatically
```

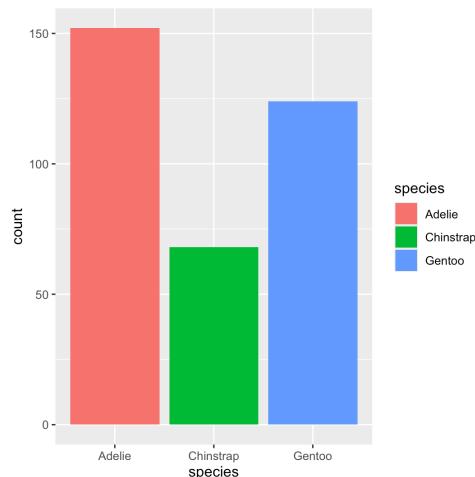
```
count_data <-  
  penguins %>%  
  count(species, name = 'count')  
  
ggplot(  
  count_data,  
  aes(  
    x = species,  
    y = count  
  )  
) +  
  geom_col()
```

25 / 71

Creating a barplot with `geom_bar()`

- Add color to barplot by defining an additional aesthetic: `fill`

```
ggplot(  
  penguins,  
  aes(  
    x = species  
  )  
) +  
  geom_bar(  
    ## Additional aesthetic: "fill"-scale  
    aes(fill = species)  
  )
```

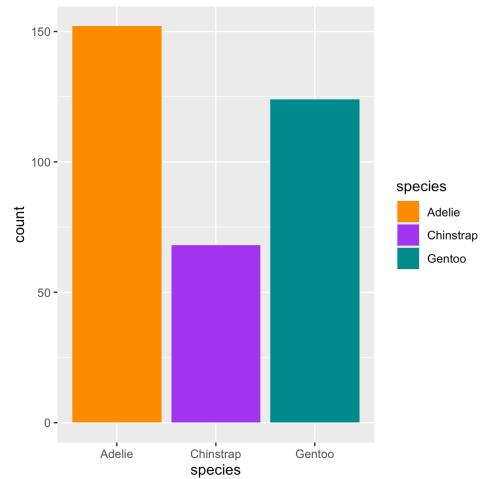


26 / 71

Creating a barplot with `geom_bar()`

- Determine fill-colors by adding `scale_fill_manual()`

```
ggplot(  
  penguins,  
  aes(  
    x = species  
  )  
) +  
  geom_bar(  
    aes(fill = species)  
  ) +  
  ## Specify fill-colors  
  scale_fill_manual(  
    values = c("darkorange", "purple", "cyan4")  
  )
```

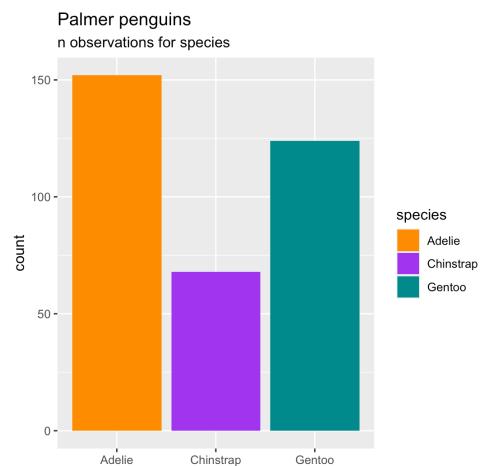


27 / 71

Creating a barplot with `geom_bar()`

- Add title, subtitle and change label of x-axis using `labs()`

```
ggplot(  
  penguins,  
  aes(  
    x = species  
  )  
) +  
  geom_bar(  
    aes(fill = species)    ) +  
  scale_fill_manual(  
    values =  
      c("darkorange", "purple", "cyan4")  
  ) +  
  ## Add title, subtitle and label x-axis  
  labs(  
    title = "Palmer penguins",  
    subtitle = "n observations for species",  
    x = ""  
)
```

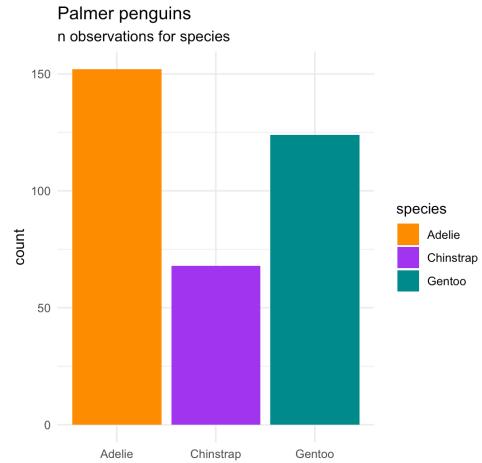


28 / 71

Creating a barplot with `geom_bar()`

- Add another *theme* using `theme_minimal()`

```
ggplot(  
  penguins,  
  aes(  
    x = species  
  )  
) +  
  geom_bar(  
    aes(fill = species)  
  ) +  
  scale_fill_manual(  
    values =  
      c("darkorange", "purple", "cyan4")  
  ) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "n observations for species",  
    x = ""  
) +  
  ## Choose theme  
  theme_minimal()
```

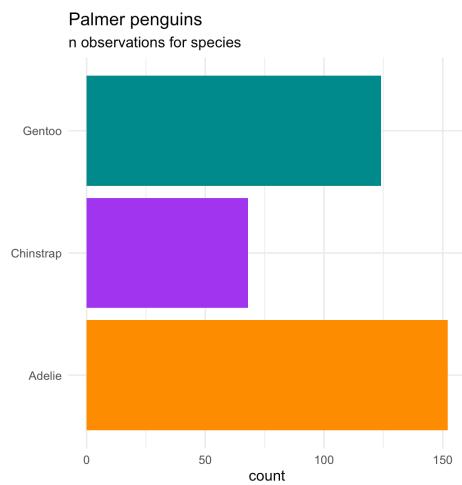


29 / 71

Creating a barplot with `geom_bar()`

- Flip x- and y-axis using `coord_flip()`
- Remove legend using `theme(legend.position = "none")`

```
ggplot(  
  penguins,  
  aes(  
    x = species  
  )  
) +  
  geom_bar(  
    aes(fill = species)  
  ) +  
  scale_fill_manual(  
    values = c("darkorange", "purple", "cyan4")  
  ) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "n observations for species",  
    x = ""  
) +  
  ## Flip x- and y-axis  
  coord_flip() +  
  theme_minimal() +  
  ## Remove legend  
  theme(  
    legend.position = "none"  
)
```

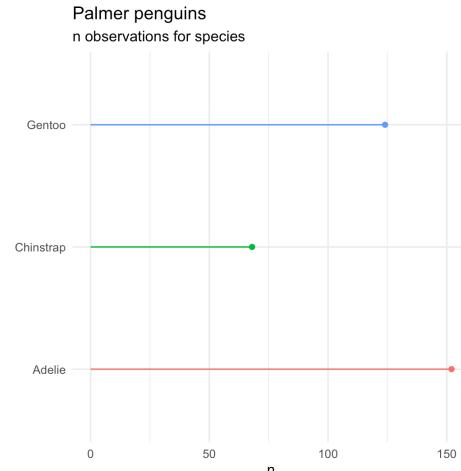


30 / 71

Creating a lollipop plot

Use two *geoms*: `geom_point()` and `geom_segment()`

```
penguins %>%
  count(species) %>%
  ggplot(
    aes(
      x = species,
      y = n)
  ) +
  geom_point(
    aes(col = species)
  ) +
  geom_segment(
    aes(
      x = species,
      xend = species,
      y = 0,
      yend = n,
      col = species
    )
  ) +
```



Re-use remainder of code!

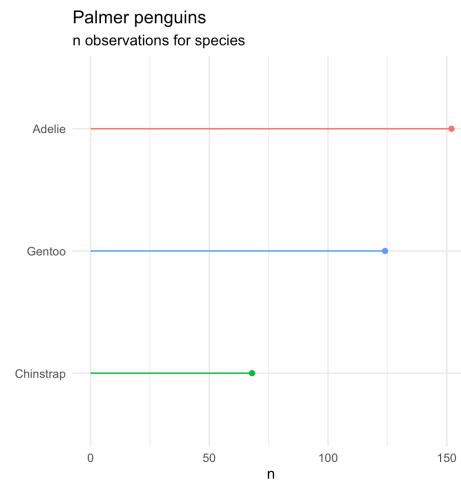
```
scale_fill_manual(
  values = c("darkorange", "purple", "cyan4")
) +
  labs(
    title = "Palmer penguins",
    subtitle = "n observations for species",
    x = ""
  ) +
  coord_flip() +
  theme_minimal() +
  theme(
    legend.position = "none"
)
```

31 / 71

Creating a lollipop plot

- Reorder 'bars' by creating a new variable using `fct_reorder()`

```
penguins %>%
  count(species) %>%
  # Create an ordered factor
  mutate(
    species_ord = fct_reorder(species, n)
  ) %>%
  ggplot(
    aes(x = species_ord)) +
  geom_point(
    aes(col = species)
  ) +
  geom_segment(
    aes(
      x = species,
      xend = species,
      y = 0,
      yend = n,
      col = species
    )
  ) +
  scale_fill_manual(
    values = c("darkorange", "purple", "cyan4")
  ) +
  labs(
    title = "Palmer penguins",
    subtitle = "n observations for species",
    x = ""
  ) +
  coord_flip() +
  theme_minimal() +
  theme(
    legend.position = "none"
)
```



32 / 71

Chosing colors ...

A colorblind-friendly palette

These are color-blind-friendly palettes, one with gray, and one with black.



To use with ggplot2, it is possible to store the palette in a variable, then use it later.

```
# The palette with grey:  
cbPalette <- c("#999999", "#E69F00", "#5684E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")  
  
# The palette with black:  
cbPalette <- c("#000000", "#E69F00", "#5684E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")  
  
# To use for fills, add  
scale_fill_manual(values=cbPalette)  
  
# To use for Line and point colors, add  
scale_color_manual(values=cbPalette)
```

This palette is from <http://jfly.iam.u-tokyo.ac.jp/color/>:

Original	Simulation	for Photoshop, Illustrator, Freehand, etc.			for Word, Power Point, Canvas, etc.			
		Protan	Deutan	Tritan	Hue	C.M.Y.K (%)	R.G.B (0-255)	R.G.B (%)
1	Black	Black	Black	Black	-°	(0,0,0,100)	(0,0,0)	(0,0,0)
2	Orange	Orange	Yellow	Pink	41°	(0,50,100,0)	(230,159,0)	(90,60,0)
3	Sky Blue	Blue	Blue	Blue	202°	(80,0,0,0)	(86,180,233)	(35,70,90)
4	bluish Green	Green	Green	Green	164°	(97,0,75,0)	(0,158,115)	(0,60,50)
5	Yellow	Yellow	Yellow	Yellow	56°	(10,50,90,0)	(240,228,66)	(95,90,25)
6	Blue	Blue	Blue	Blue	202°	(100,50,0,0)	(0,114,178)	(0,45,70)
7	Vermilion	Red	Red	Red	27°	(0,80,100,0)	(213,94,0)	(80,40,0)
8	reddish Purple	Purple	Purple	Purple	326°	(10,70,0,0)	(204,121,167)	(80,60,70)

[http://www.cookbook-r.com/Graphs/Colors_\(ggplot2\)/](http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/)

33 / 71

Exercises [ggplot2] : part 1



- You can find the qmd-file `Exercises_ggplot2.qmd` at the course website.
- Download the qmd-file `Exercises_ggplot2.qmd` to your laptop
- Open the file in RStudio
- The file contains a set of coding assignments with empty code blocks
- Now, we focus on part 1 of the exercises
- Write the code (and test it by running it)
- Stuck? No Worries!
 - We are there
 - Help each other
 - There is a solution key (at the website) (`Exercises_ggplot2_solutions.qmd`)

34 / 71

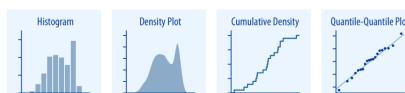
5. Visualising a quantitative variable

Visualising a quantitative variable?

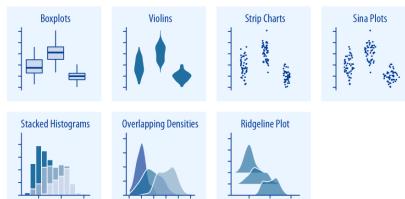
There are many ways to visualise a quantitative variable...

Can you think of some?

5.2 Distributions



Histograms and density plots (Chapter 7) provide the most intuitive visualizations of a distribution, but both require arbitrary parameter choices and can be misleading. Cumulative densities and quantile-quantile (q-q) plots (Chapter 8) always represent the data faithfully but can be more difficult to interpret.

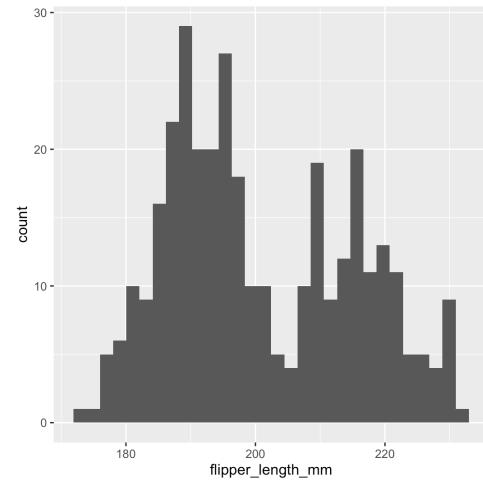


Boxplots, violins, strip charts, and sina plots are useful when we want to visualize many distributions at once and/or if we are primarily interested in overall shifts among the distributions (Chapter 9.1). Stacked histograms and overlapping densities allow a more in-depth comparison of a smaller number of distributions, though stacked histograms can be difficult to interpret and are best avoided (Chapter 7.2). Ridgeline plots can be a useful alternative to violin plots and are often useful when visualizing very large numbers of distributions or changes in distributions over time (Chapter 9.2).

Taken from *Fundamentals of Data Visualization* by Claus Wilke

Creating a histogram with `geom_histogram()`

```
ggplot(  
  penguins,  
  aes(  
    x = flipper_length_mm  
  )  
) +  
  geom_histogram()
```

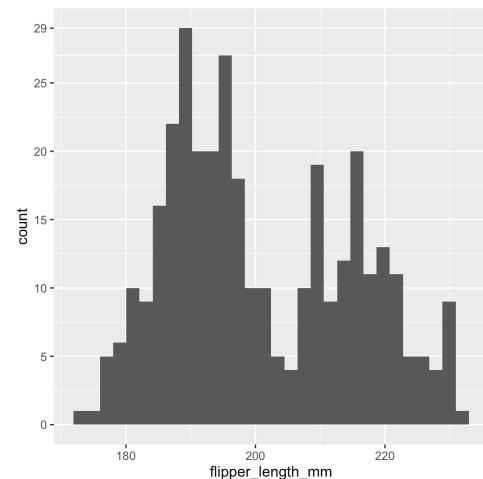


37 / 71

Creating a histogram with `geom_histogram()`

- Change breaks on y-axis using `scale_y_continuous()`

```
ggplot(  
  penguins,  
  aes(  
    x = flipper_length_mm  
  )  
) +  
  geom_histogram() +  
  ## Specify breaks  
  scale_y_continuous(  
    breaks = c(seq(0, 25, 5), 29)  
  )
```

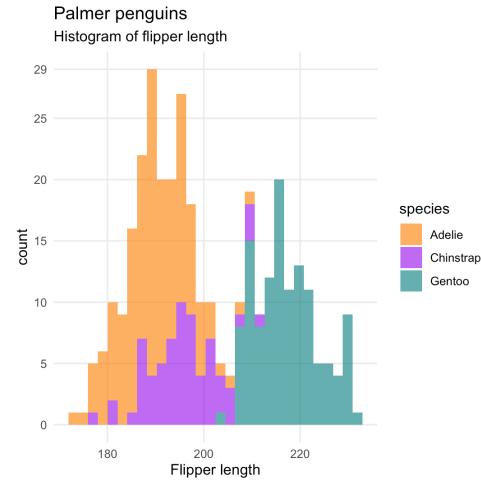


38 / 71

Creating a histogram with `geom_histogram()`

- Change transparency using argument `alpha`
- Remove minor grid lines using argument `panel.grid.minor = element_blank()`

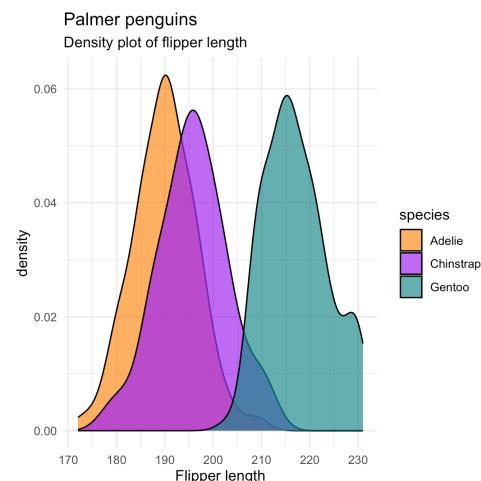
```
ggplot(  
  penguins,  
  aes(  
    x = flipper_length_mm,  
    fill = species  
  )  
) +  
  geom_histogram(  
    ## Change transparency of points  
    alpha = .7  
  ) +  
  ## Specify breaks  
  scale_y_continuous(  
    breaks = c(seq(0, 25, 5), 29)  
  ) +  
  scale_fill_manual(  
    values =  
      c("darkorange", "purple", "cyan4")  
  ) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Histogram of flipper length",  
    x = "Flipper length"  
  ) +  
  theme_minimal() +  
  theme(  
    ## Remove minor grid lines  
    panel.grid.minor = element_blank()  
)
```



Creating a density plot with `geom_density()`

- Replace `geom_histogram()` with `geom_density()` plot

```
ggplot(  
  penguins,  
  aes(  
    x = flipper_length_mm,  
    fill = species  
  )  
) +  
  ## Use geom_density  
  geom_density(  
    alpha = .7  
  ) +  
  scale_fill_manual(  
    values = c("darkorange", "purple", "cyan4")  
  ) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Density plot of flipper length",  
    x = "Flipper length"  
  ) +  
  theme_minimal()
```

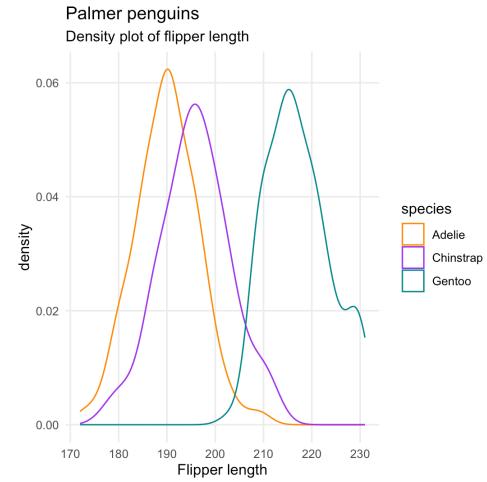


*surface below curve = 100%

Creating a density plot with `geom_density()`

- Only colored lines by replacing argument `fill` with `color`

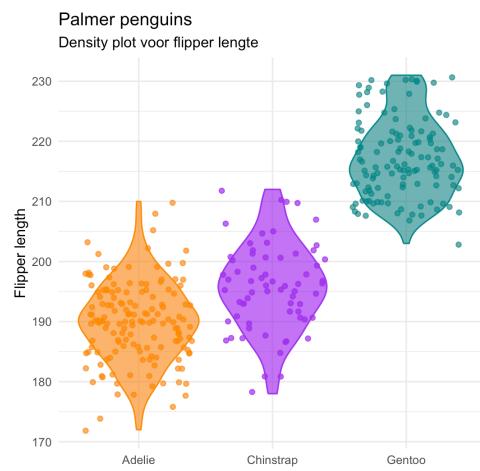
```
ggplot(  
  penguins,  
  aes(  
    x = flipper_length_mm,  
    color = species  
  )  
) +  
  geom_density() +  
  ## Replace scale_color with scale_fill  
  scale_color_manual(  
    values = c("darkorange", "purple", "cyan4")  
  ) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Density plot of flipper length",  
    x = "Flipper length"  
  ) +  
  theme_minimal() +  
  theme(  
    panel.grid.minor = element_blank()  
  )
```



41 / 71

Or even better ... `geom_violin()` + `geom_jitter()`

```
ggplot(  
  penguins,  
  aes(  
    x = species,  
    y = flipper_length_mm,  
    ## Aesthetics fill and color applied to EVERY geom  
    fill = species,  
    color = species  
  )  
) +  
  ## Use geom_violin and geom_jitter  
  geom_violin(  
    alpha = .65  
  ) +  
  geom_jitter(  
    alpha = .7  
  ) +  
  scale_fill_manual(  
    values = c("darkorange", "purple", "cyan4")  
  ) +  
  scale_color_manual(  
    values = c("darkorange", "purple", "cyan4")  
  ) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Density plot voor flipper lengte",  
    y = "Flipper length",  
    x = "",  
  ) +  
  theme_minimal() +  
  theme(  
    legend.position = "none"  
  )
```

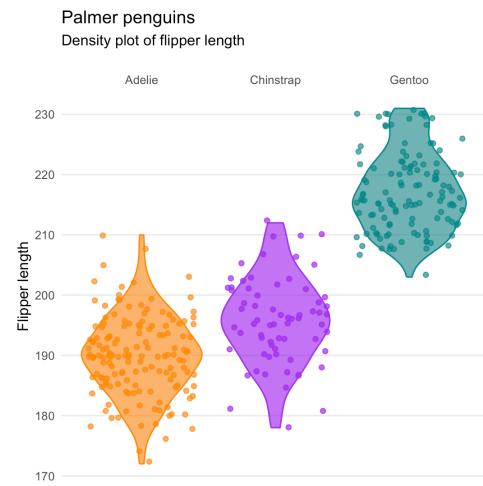


42 / 71

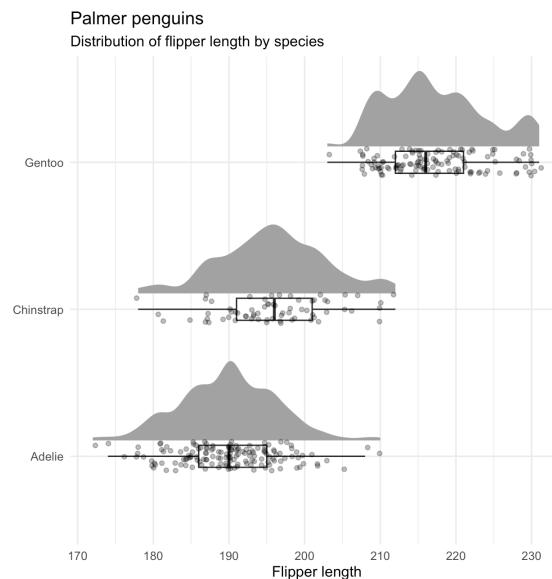
Or even better `geom_violin()` + `geom_jitter()`

- Put names of species on top using argument `position="top"`
- Remove minor grid lines using argument `panel.grid.major = element_blank()`

```
ggplot(  
  penguins,  
  aes(  
    x = species,  
    y = flipper_length_mm,  
    fill = species,  
    color = species  
  ) +  
  geom_violin(  
    alpha = .65  
  ) +  
  geom_jitter(  
    alpha = .7  
  ) +  
  scale_x_discrete(  
    position = "top"  
  ) +  
  scale_fill_manual(  
    values = c("darkorange","purple","cyan4")  
  ) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Density plot of flipper length",  
    y = "Flipper length",  
    x = "",  
    ) +  
  theme_minimal() +  
  theme(  
    legend.position = "none",  
    panel.grid.minor = element_blank(),  
    ## Remove major grid lines  
    panel.grid.major.x = element_blank()  
  )
```



Or a *rain cloud plot*



Based on the tutorial of Cédric Scherer: <https://www.cedricscherer.com/>

Or a *rain cloud plot* (Appendix 1: How to grow a rain cloud plot?)

Additional package required: `ggdist`

```
library(ggdist)
ggplot(penguins, aes(x = species, y = flipper_length_mm)) +
  stat_halfeye(
    adjust = .5,
    width = .6,
    .width = 0,
    justification = -.2,
    point_color = NA
  ) +
  geom_boxplot(
    width = .15,
    outlier.shape = NA
  ) +
  geom_point(
    size = 1.3,
    alpha = .3,
    position = position_jitter(
      seed = 1, width = .1
    )
  ) +
  labs(
    title = "Palmer penguins",
    subtitle = "Distribution of flipper length by species",
    x = "",
    y = "Flipper length"
  ) +
  coord_cartesian(xlim = c(1.2, NA), clip = "off") +
  coord_flip() +
  theme_minimal()
```

45 / 71

💻 Exercises [ggplot2] : part 2



- You can find the qmd-file `Exercises_ggplot2.qmd` at the course website.
- Download this file to your laptop
- Open the file in RStudio
- The file contains a set of coding assignments with empty code blocks
- Now, we focus on part 2 of the exercises
- Write the code (and test it by running it)
- Stuck? No Worries!
 - We are there
 - Help each other
 - There is a solution key (`Exercises_ggplot2_solutions.qmd`)

46 / 71

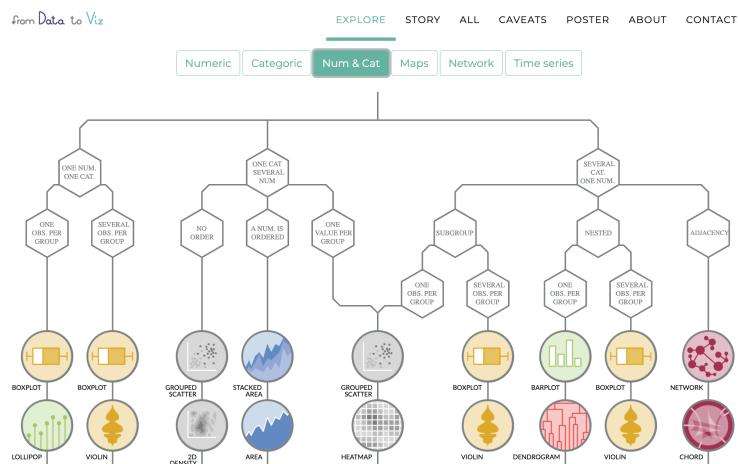
6. Visualising more than one variable

Visualising more than one variable?

There are many ways to visualise more than one variable...
The choice depends (among other things) on the type of variables you want to plot:

- only quantitative variables;
- only qualitative variables;
- or a combination of both

Can you think of an example of each?



Have a look at [data to viz.com](http://data-to-viz.com)

Visualising more than one variable

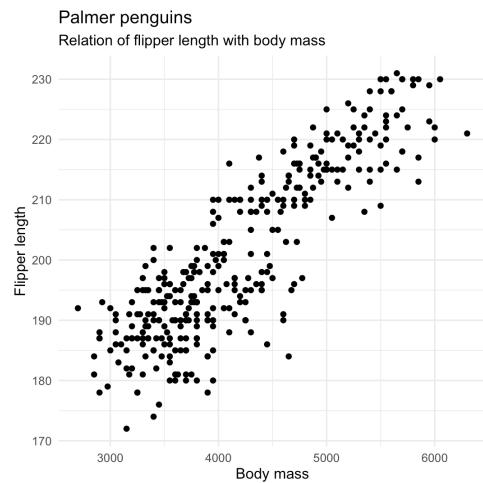
We focus today upon:

- **scatterplots** (two numeric variables, two numeric variables and one categorical variable)
- **grouped barplots** (two categorical variables)

49 / 71

Creating a scatterplot with `geom_point()`

```
ggplot(  
  penguins,  
  aes(  
    x = body_mass_g,  
    y = flipper_length_mm  
  )  
) +  
  geom_point() +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Relation of flipper length with body mass",  
    y = "Flipper length",  
    x = "Body mass",  
  ) +  
  theme_minimal() +  
  theme(  
    legend.position = "none"  
  )
```

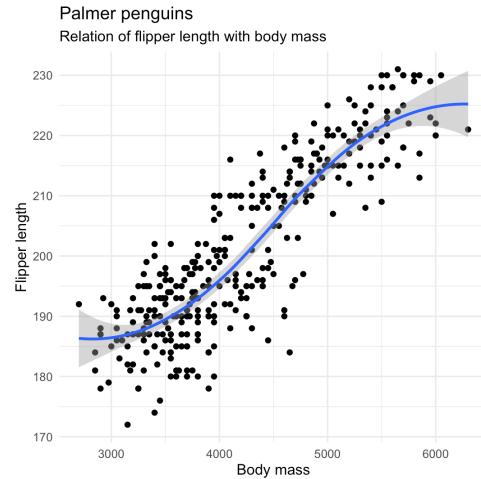


50 / 71

Creating a scatterplot with `geom_point()` and `geom_smooth()`

- Add trend line using `geom_smooth()`

```
ggplot(  
  penguins,  
  aes(  
    x = body_mass_g,  
    y = flipper_length_mm  
  )  
) +  
  geom_point() +  
  ## Add a trendline  
  geom_smooth() +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Relation of flipper length with body mass",  
    y = "Flipper length",  
    x = "Body mass"  
) +  
  theme_minimal() +  
  theme(  
    legend.position = "none"  
)
```

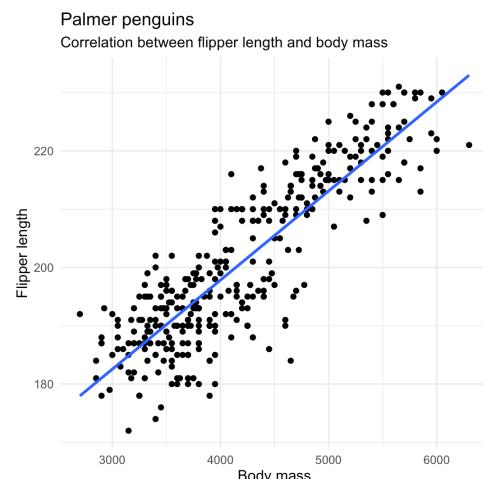


51 / 71

Creating a scatterplot with `geom_point()` and `geom_smooth()`

- Linear trend line using argument `method = "lm"`
- Remove confidence intervals using argument `se = FALSE`

```
ggplot(  
  penguins,  
  aes(  
    x = body_mass_g,  
    y = flipper_length_mm  
  )  
) +  
  geom_point() +  
  geom_smooth(  
    ## Add linear trend line  
    method = "lm",  
    ## Remove confidence band  
    se = FALSE  
) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Correlation between flipper length and body mass",  
    y = "Flipper length",  
    x = "Body mass",  
) +  
  theme_minimal() +  
  theme(  
    legend.position = "none"  
)
```



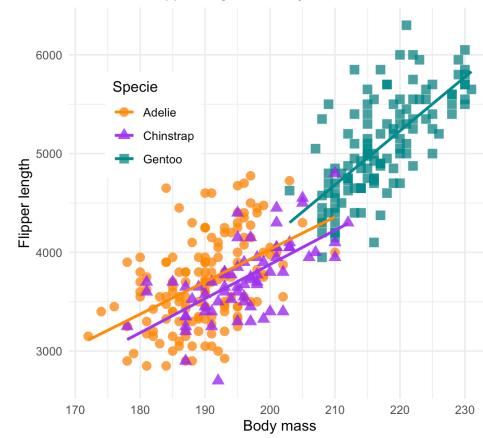
52 / 71

Creating a scatterplot with `geom_point()` and `geom_smooth()`

- Add shape and color by specifying *aesthetics*: `shape` and `color`
- Change legend position and background, position of plot title, and layout of (sub)title

```
ggplot(  
  penguins,  
  aes(  
    x = flipper_length_mm,  
    y = body_mass_g  
  )  
)+  
  geom_point(  
    aes(  
      color = species,  
      shape = species  
    ),  
    size = 3,  
    alpha = 0.8  
  )  
  geom_smooth(  
    aes(color = species),  
    se = F,  
    method = "lm"  
  )  
  theme_minimal()  
  scale_color_manual(  
    values = c("darkorange", "purple", "cyan4")) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Correlation between flipper length and body mass",  
    y = "Flipper length",  
    x = "Body mass",  
    color = "Species",  
    shape = "Species") +  
  theme(  
    legend.position = c(0.2, 0.7),  
    legend.background = element_rect(fill = "white", color = NA),  
    plot.title.position = "plot",  
    plot.title = element_text(hjust = 0, face = "bold"),  
    plot.subtitle = element_text(hjust = 0, face = "italic")  
)
```

Palmer penguins
Correlation between flipper length and body mass



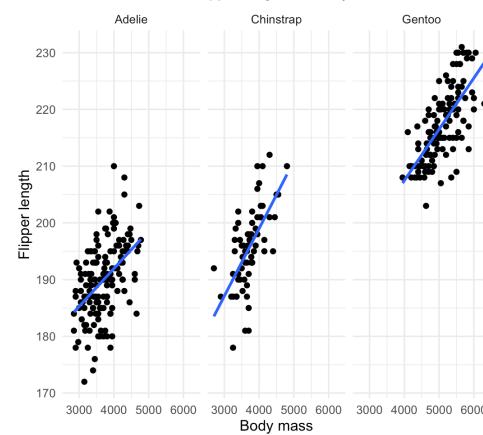
53 / 71

Creating a scatterplot with `geom_point()` and `geom_smooth()`

- Adds facets using `facet_wrap()`

```
ggplot(  
  penguins,  
  aes(  
    x = body_mass_g,  
    y = flipper_length_mm  
  )  
)+  
  geom_point() +  
  geom_smooth(  
    method = "lm",  
    se = FALSE  
  ) +  
  facet_wrap(~species) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Correlation between flipper length and body mass",  
    y = "Flipper length",  
    x = "Body mass",  
  ) +  
  theme_minimal() +  
  theme(  
    legend.position = "none",  
    plot.title = element_text(hjust = 0, face = "bold"),  
    plot.subtitle = element_text(hjust = 0, face = "italic")  
)
```

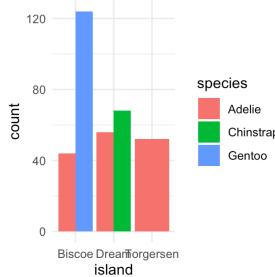
Palmer penguins
Correlation between flipper length and body mass



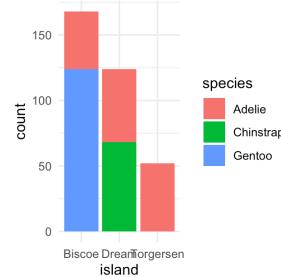
54 / 71

Creating a barplot of two variables (counts)

Grouped barplot of counts using
`position_dodge()`



Stacked barplot of counts using argument
`position_stack()`



```
penguins %>%
  ggplot(
    aes(
      x = island,
      fill = species
    )
  ) +
  geom_bar(
    position = position_dodge()
  ) +
  theme_minimal()
```

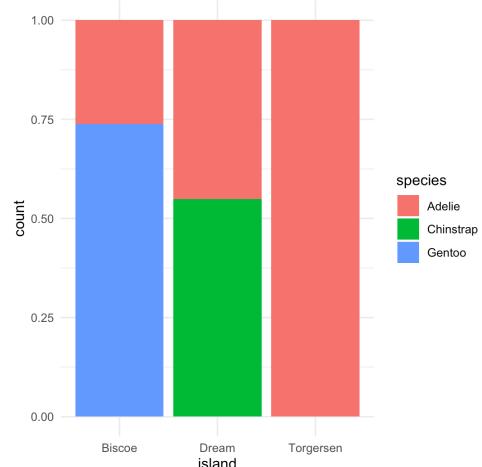
```
penguins %>%
  ggplot(
    aes(
      x = island,
      fill = species
    )
  ) +
  geom_bar(
    position = position_stack()
  ) +
  theme_minimal()
```

55 / 71

Creating a barplot of two variables (percentage)

- Switch to relative frequencies using
`position_fill()`

```
penguins %>%
  ggplot(
    aes(
      x = island,
      fill = species
    )
  ) +
  geom_bar(
    position = position_fill()
  ) +
  theme_minimal()
```

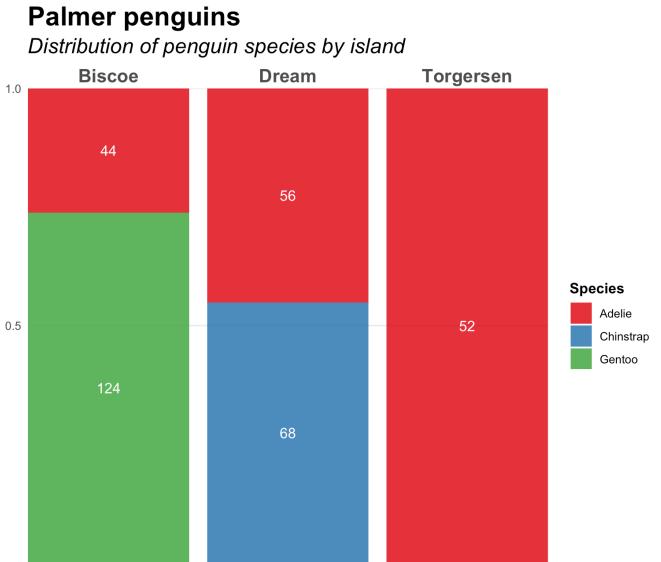


56 / 71

Creating a barplot of two variables (percentage)

- Add texts to bars using `geom_text()`
- Add "fill" colors using `scale_fill_brewer()`
- Remove additional space using
`coord_cartesian(expand=FALSE)`

```
penguins %>%  
  ggplot(  
    aes(  
      x = island,  
      fill = species  
    )  
  ) +  
  geom_bar(  
    position = position_fill(),  
    alpha = .9  
  ) +  
  geom_text(  
    aes(label = ..count..),  
    stat = "count",  
    colour = "white",  
    position = position_fill(vjust = 0.5)  
  ) +  
  scale_fill_brewer(  
    type = "qual",  
    palette = 6  
  ) +  
  scale_x_discrete(position = "top") +  
  scale_y_continuous(breaks = c(0.5, 1)) +  
  labs(  
    title = "Palmer penguins",  
    subtitle = "Distribution of penguin species by island",  
    fill = "Species"  
  ) +  
  coord_cartesian(expand = FALSE) +  
  theme_minimal() +  
  theme(  
    plot.title = element_text(size = 20, face = "bold"),  
    plot.subtitle = element_text(size = 16, face = "italic"),  
    axis.title = element_blank(),  
    axis.text = element_text(size = 14, face = "bold"),  
    legend.title = element_text(size = 14, face = "bold"),  
    panel.grid.minor = element_blank(),  
    panel.grid.major.x = element_blank()  
  )
```



57 / 71

Choosing colors ...

- Colors are not meaningless and ... they grab attention
- R has several built-in color palettes. The functions `scale_fill_brewer` and `scale_color_brewer` use palettes of [RColorBrewer](#). These palettes come in three 'flavours':
 - **sequential**: for ordered data ((for example, scoring low/mediocre/high on a likert scale)
 - **qualitative**: for nominal or categorical information (for example, different islands)
 - **diverging**: for ordered data with meaningful mid values ((for example, temperature or change in score)



58 / 71



Exercises [ggplot2] : part 3



- You can find the qmd-file [Exercises_ggplot2.qmd](#) at the course website.
- Download this file to your laptop
- Open the file in RStudio
- The file contains a set of coding assignments with empty code blocks
- Now, we focus on part 3 of the exercises
- Write the code (and test it by running it)
- Stuck? No Worries!
 - We are there
 - Help each other
 - There is a solution key ([Exercises_ggplot2_solutions.qmd](#))

7. More about visualisation?

The tip of the icebeRg...

Gallery of different types of visualisations
accompained by ggplot-code

Q CHART TYPES QUICK TOOLS ALL RELATED ABOUT

The R Graph Gallery



Welcome to The R graph gallery, a collection of charts made with the R programming language. Hundreds of charts are displayed in several sections, always with their reproducible code available. The gallery makes a focus on the tidyverse and `ggplot2`. Feel free to suggest a chart or report a bug: any feedback is highly welcome. Stay in touch with the gallery by following it on [Twitter](#) or [GitHub](#). If you're new to R, consider following [this course](#).

library of graphs with ggplot-code

R CHARTS HOME BASE R + GGPLOT2 COLORS LEARN R ABOUT CONTACT

ggplot2 package

ggplot2 is the most popular alternative to base R graphics. It is based on the Grammar of Graphics and its main advantage is its flexibility, as you can create and customize the graphics adding more layers to it. This library allows creating ready-to-publish charts easily.

Top50 ggplot visualisations...

Top 50 ggplot2 Visualizations - The Master List (With Full R Code)

What type of visualization to use for what sort of problem? This tutorial helps you choose the right type of chart for your specific objectives and how to implement it in R using `ggplot2`.

- This is part 3 of a three part tutorial on `ggplot2`, an aesthetically pleasing (and very popular) graphics framework in R. This tutorial is primarily geared towards those having some basic knowledge of the R programming language and want to make complex and nice looking charts with R `ggplot2`.
- Part 1: [Introduction to ggplot2](#), covers the basic knowledge about constructing simple ggplots and modifying the components and aesthetics.
 - Part 2: [Customizing the Look and Feel](#), is about more advanced customization like manipulating legend, annotations, multiplots with faceting and custom layouts
 - Part 3: [Top 50 ggplot2 Visualizations - The Master List](#), applies what was learnt in part 1 and 2 to construct other types of ggplots such as bar charts, boxplots etc.

ggplot-extentions



61 / 71

Which visualisation to use?

Visual vocabulary

Visual Vocabulary

Designing with data

There are so many ways to visualise data – how do we know which one to pick? Click on the coloured categories below to decide which data relationship is most important in your story, then look at the different types of chart within the category to form some initial ideas about what might work best. This list is not meant to be exhaustive, nor a wizard, but is a useful starting point for making informative and meaningful data visualizations

Inspired by the Graphic Continuum by Jon Scherzer and Sevenero Ribeiro

The Data Visualisation Catalogue

What do you want to show?

Here you can find a list of charts categorized by their data visualization functions or by what you want a chart to communicate. This list is not exhaustive, nor a wizard, but is a useful guide for selecting chart based on your analysis or communication needs.



Research visuals: all the resources ...

BARYON SERVICES PORTFOLIO BLOG JOBS BOOK ABOUT US CONTACT ← ANALYSTS

Research visuals: all the resources you'll ever need!

Website of Cedric Scherer

Blog Gallery Portfolio About Me Links

CÉDRIC SCHERER Data Visualization & Information Design

62 / 71

In need of help?

Don't forget RStudio's help-function!

Google is your best fRiend...

Just google your question and you'll find code, examples, ...

Generative AI can help as well!

Appendix 1

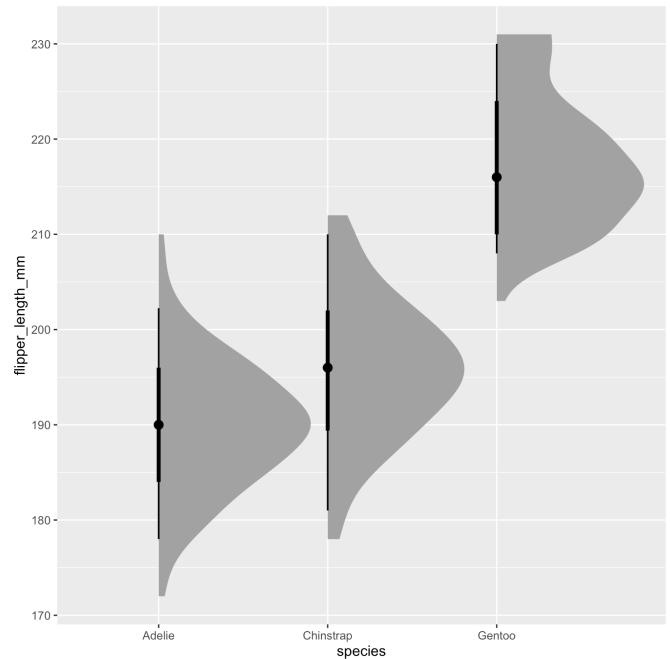
How to grow a rain cloud plot?

[Back to slide show...](#)

Step 1

```
library(ggdist)
P1 <- ggplot(
  penguins,
  aes(
    x = species,
    y = flipper_length_mm
  )
) +
  stat_halfeye()
```

P1

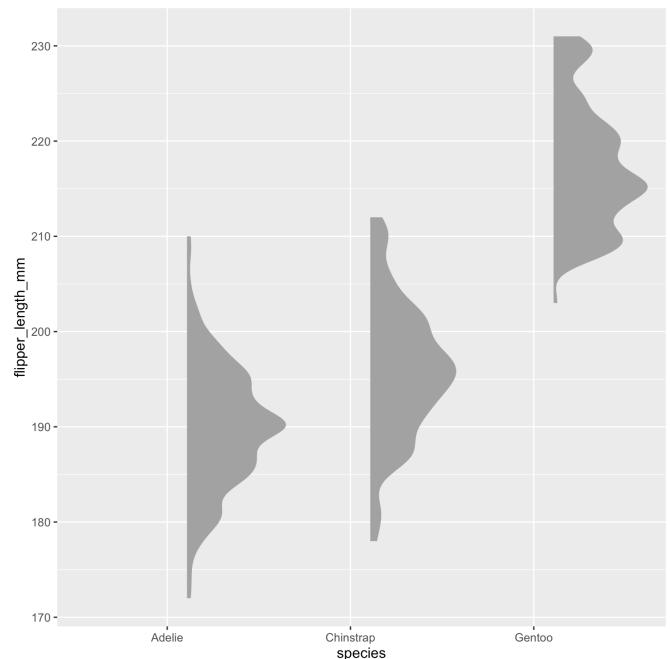


65 / 71

Step 2

```
P1 <- ggplot(
  penguins,
  aes(
    x = species,
    y = flipper_length_mm
  )
) +
  stat_halfeye(
    adjust = .5,
    width = .6,
    .width = 0,
    justification = -.2,
    point_color = NA
  )
```

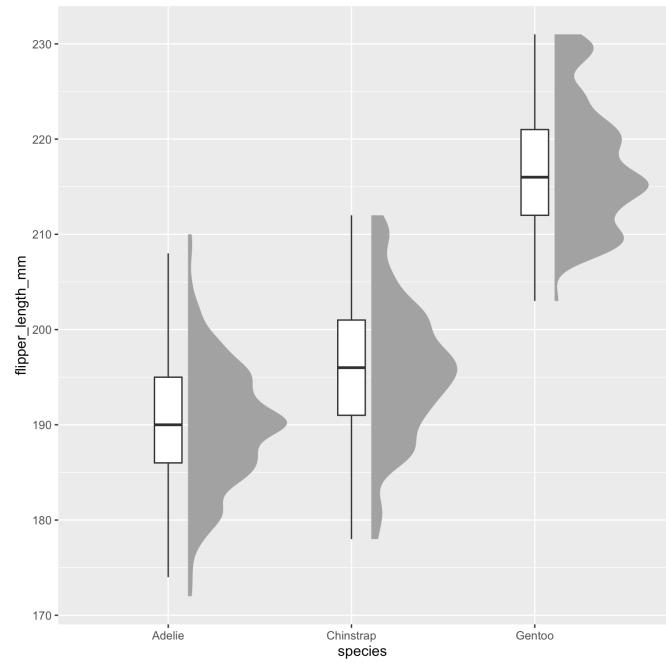
P1



66 / 71

Step 3

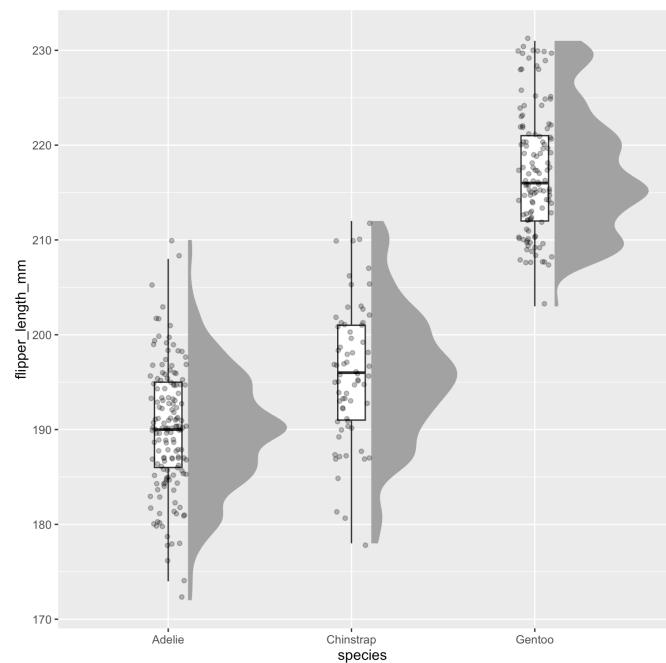
```
P2 <- P1 +
  geom_boxplot(
    width = .15,
    outlier.shape = NA
  )
P2
```



67 / 71

Step 4

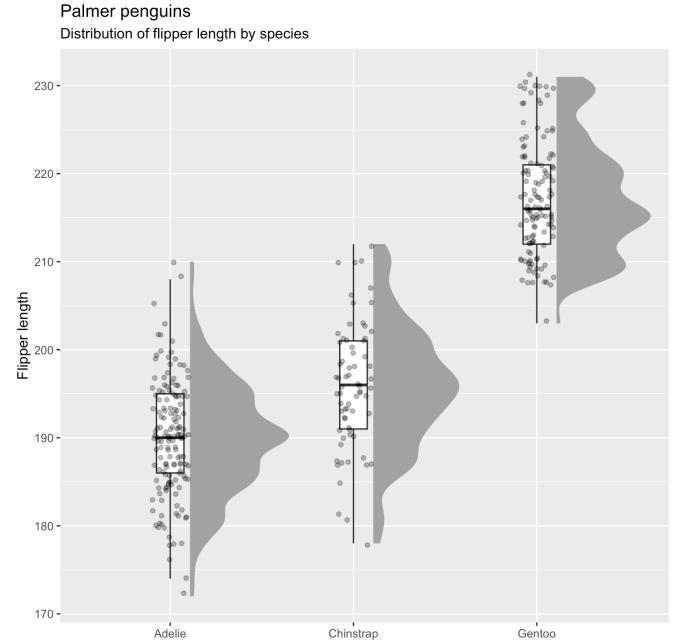
```
P3 <- P2 +
  geom_point(
    size = 1.3,
    alpha = .3,
    position = position_jitter(
      seed = 1,
      width = .1
    )
  )
P3
```



68 / 71

Step 5

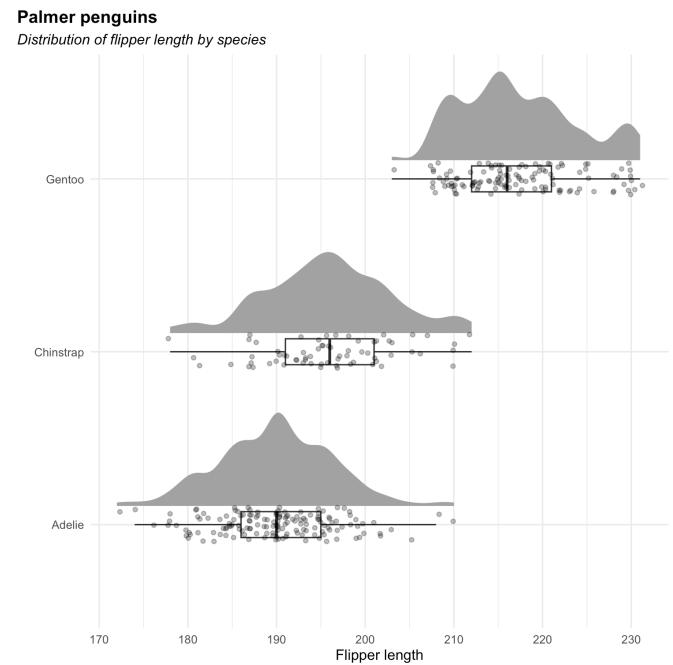
```
P4 <- P3 +
  labs(
    title = "Palmer penguins",
    subtitle = "Distribution of flipper length by species",
    x = "",
    y = "Flipper length"
  )
P4
```



69 / 71

Step 6

```
P5 <- P4 +
  coord_cartesian(xlim = c(1.2, NA), clip = "off") +
  coord_flip() +
  theme_minimal() +
  theme(
    plot.title.position = "plot",
    plot.title = element_text(face = "bold"),
    plot.subtitle = element_text(face = "italic")
  )
P5
```



70 / 71

THE RAIN CLOUD PLOTS / Back to slide show...

