Show Me the Data!

Types of Variables
Data Visualization
Smoothers

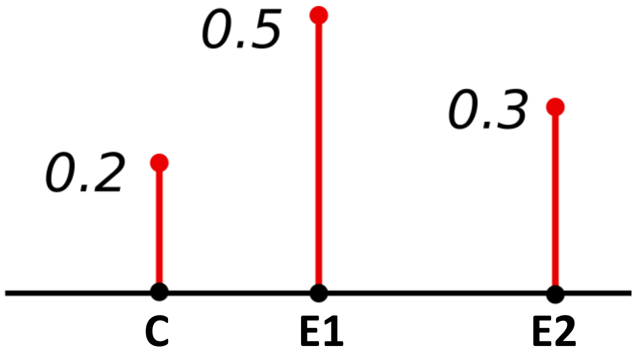
Building a model

- Regression is about understanding the **conditional distribution** of a response variable, Y

Two types of variables

- Categorical (i.e., discrete)
  - Variables take one of several specific values (countable number)

A discrete distribution

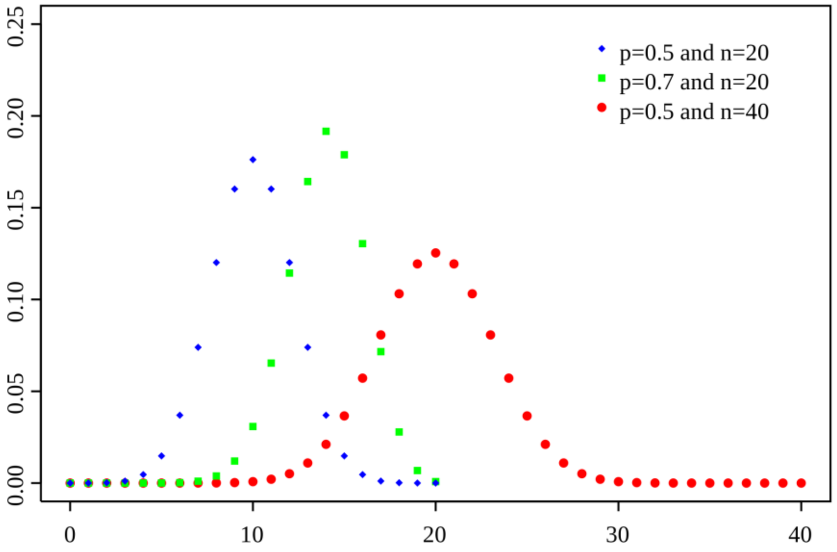


Bernoulli distribution

$$y \sim Bernoulli(p)$$

$$p_Y(y) = \begin{cases} p & \text{if } y \text{ is } 1 \\ 1 - p & \text{if } y \text{ is } 0 \end{cases}$$

Binomial distribution

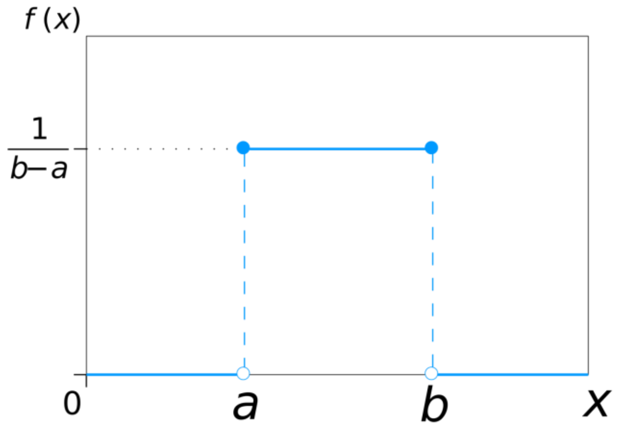


UNIVERSITY OF SOUTH FLORIDA

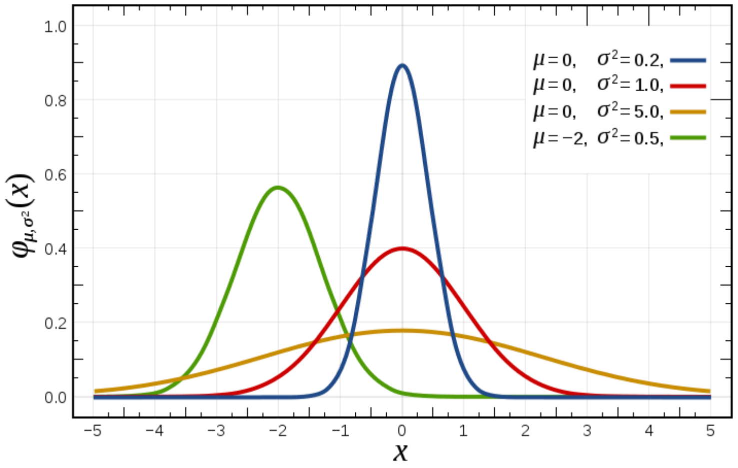
Two types of variables

- Continuous (i.e., numeric, metric)
  - Variables take any real value (potentially bounded)

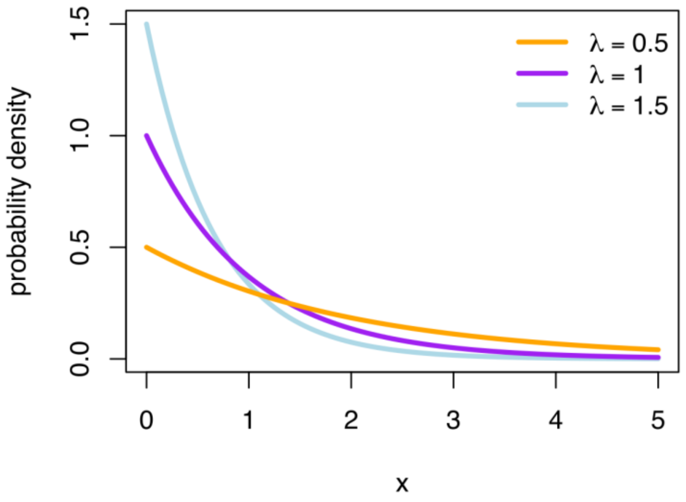
A uniform distribution



Uniform distribution



Exponential distribution



UNIVERSITY OF SOUTH FLORIDA

Special case: Ordinal variables

- Ordinal (or "ordered") variables are a bit special
- Categorical, but with an order
  - More information than unordered categorical, but less than continuous
- Various ways to handle ordinal variables
  - Need to make some assumption about relationship to underlying continuous variable
  - Handling depends on whether the ordinal variable is the response (Y), a predictor (X), or both
  - In some cases, it can be okay to treat an ordinal scale like it's continuous, but this can also cause problems

~~Stevens' levels of measurement~~

- Notice that we **aren't** talking about Stevens' levels of measurement!
  - Nominal, Ordinal, Interval, Ratio
- Stevens suggested that choice of statistical model should be determined by **what data transformations are allowed**
  - That doesn't make much sense
- What matters is the sampling distribution, not how you can transform the data
  - e.g., some θ scores estimated in IRT can be nonlinearly transformed without changing the meaning ("ordinal" in Stevens' classification) but can still be meaningfully modeled using a normal OLS regression
  - See (Zumbo & Kroc, 2019, https://doi.org/10.1177/0013164419844305)

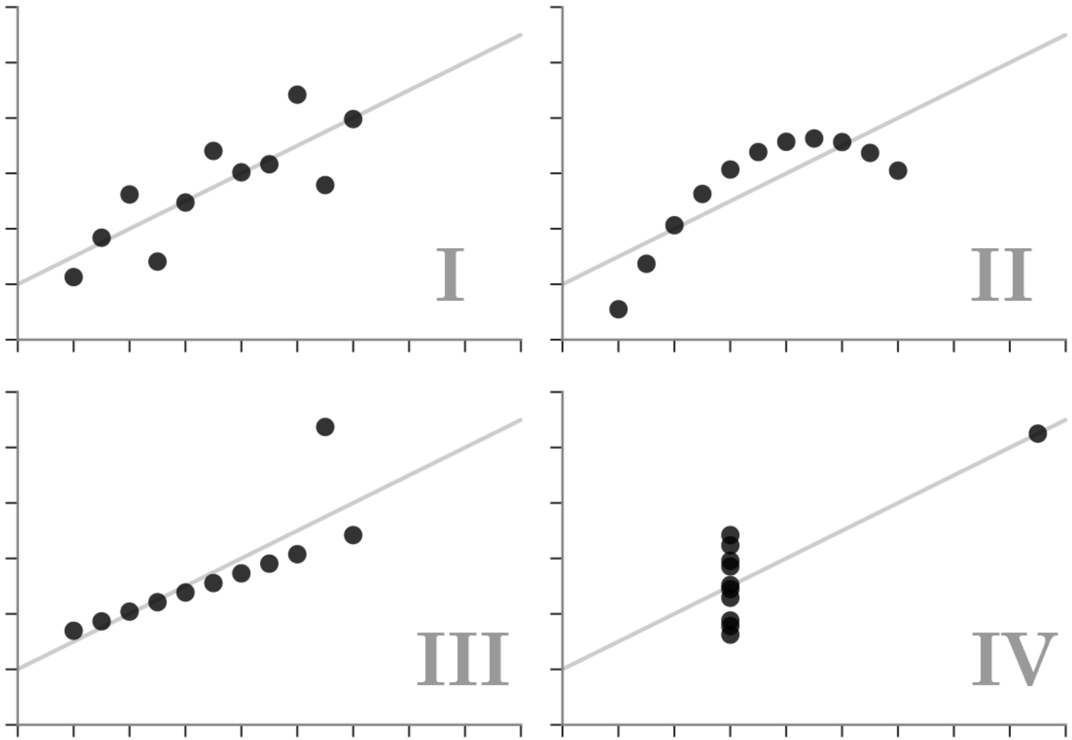
Modeling rule no. 1: Always plot your data!

Always plot your data!

- `datasets::anscombe`

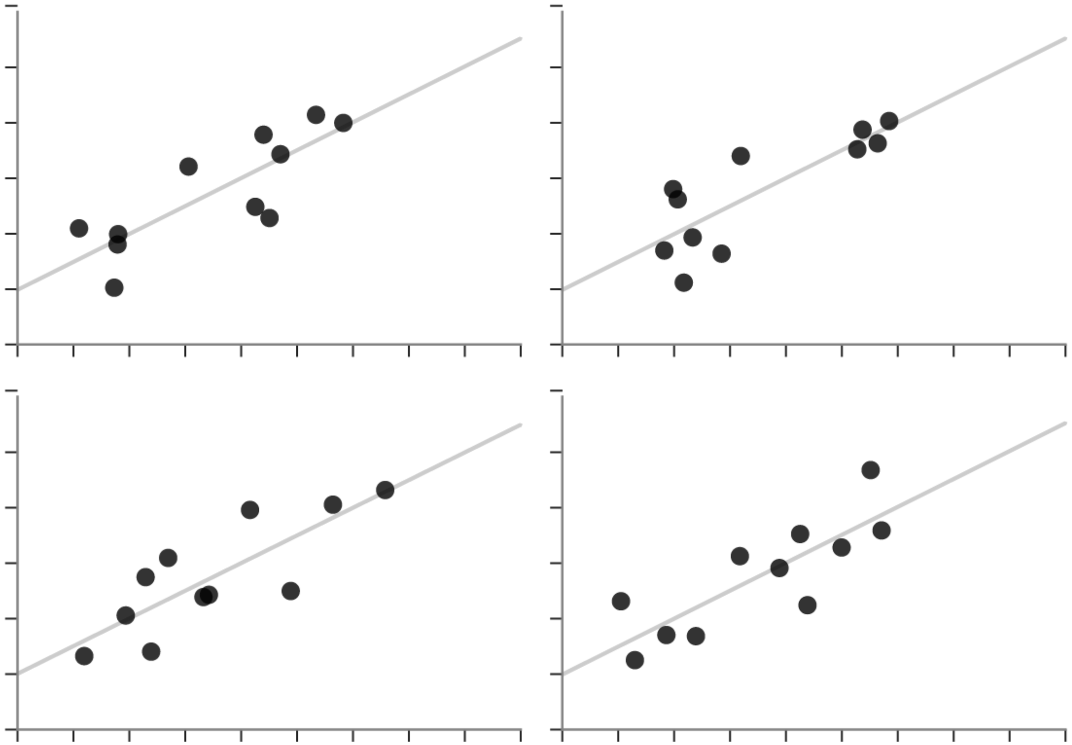
Anscombe's Quartet

Each dataset has the same summary statistics (mean, standard deviation, correlation), and the datasets are *clearly different*, and *visually distinct*.

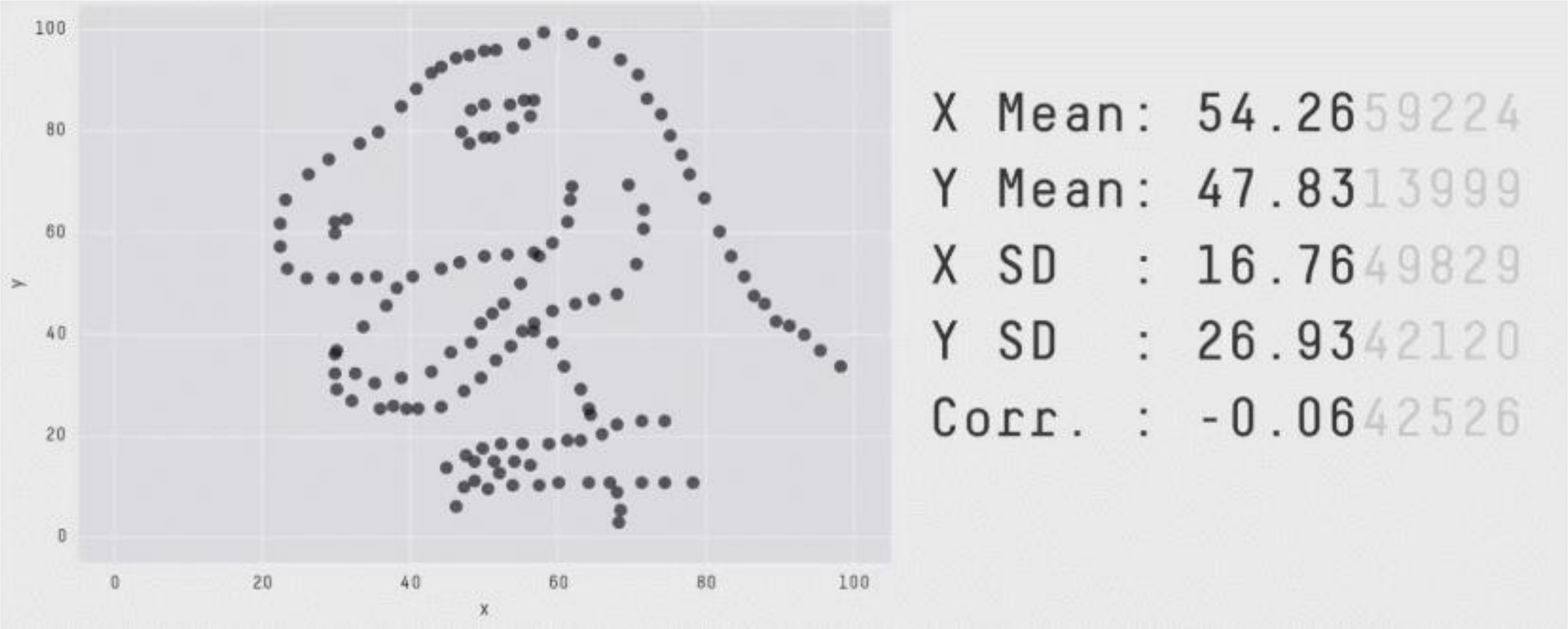


Unstructured Quartet

Each dataset here also has the same summary statistics. However, they are not *clearly different* or *visually distinct*.



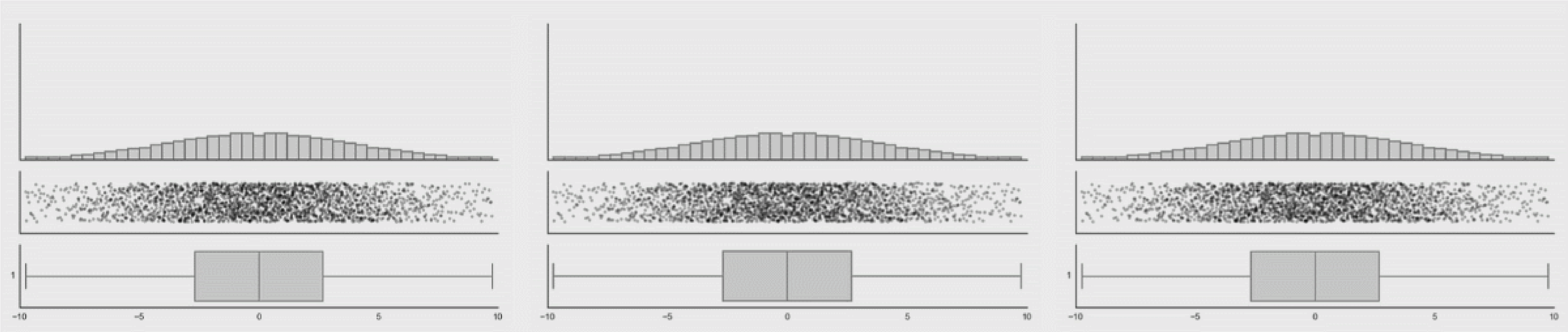
Always plot your data!



Source

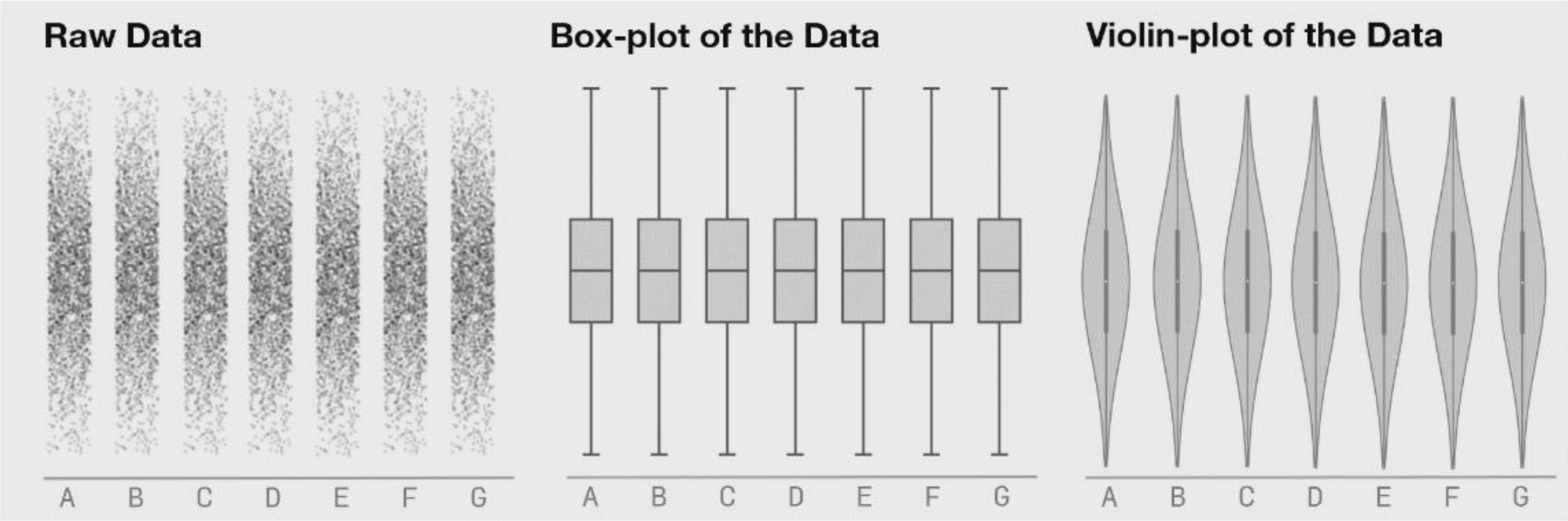
Modeling rule no. 2: Always plot your ***DATA***!

Always plot your data!



Source

Always plot your data!



Source

Always plot your data

- At the beginning of analyses:
  - Plot marginal distributions
  - Plot bivariate distributions
  - Show the actual data points!
  - **Exploratory data analysis**

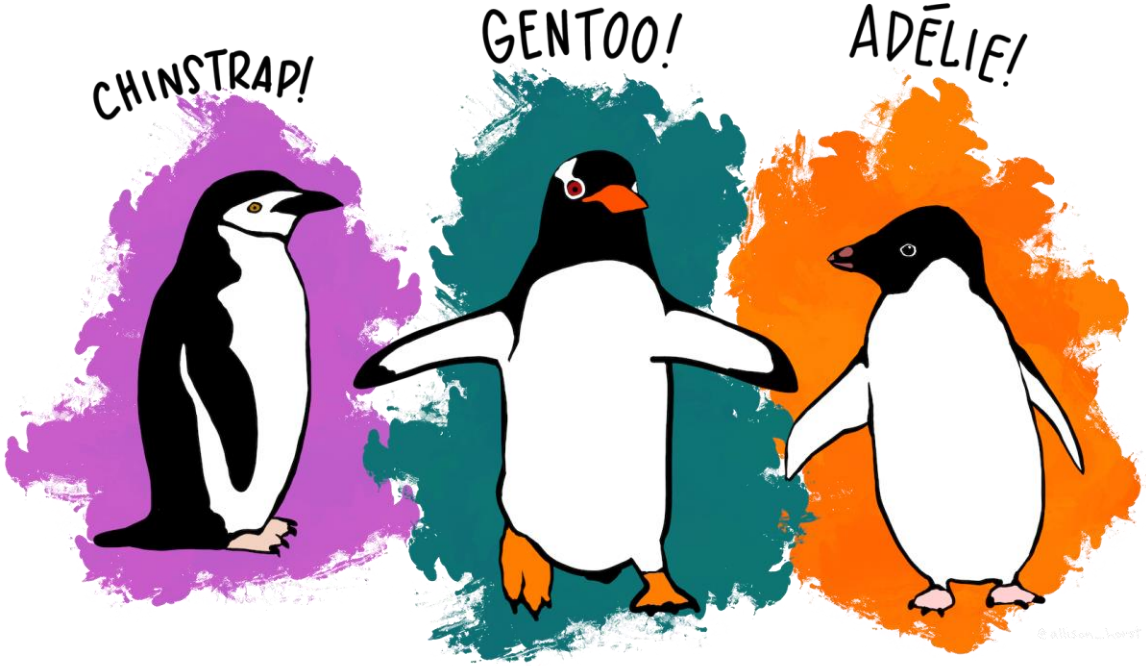
- At the end of analyses:
  - Visualize your results
  - A plot is worth 1000 tables!
  - Show **model estimates** and **uncertainty**
  - Show **raw data** and **model predictions**

Plotting one variable

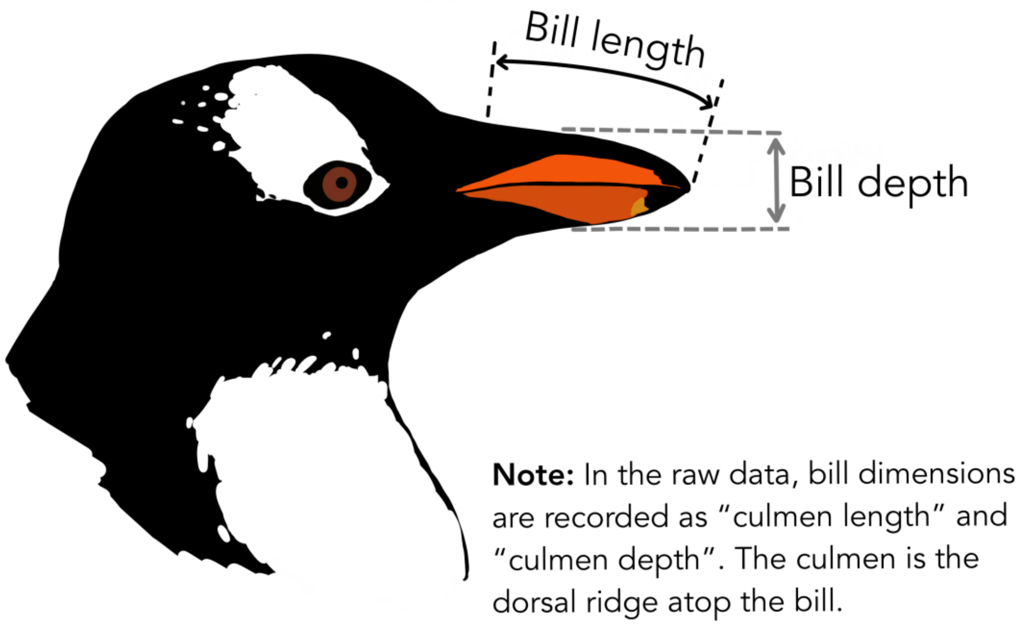
- Show the **shape of the distribution**
  - What are the **possible values**?
  - Which values are **more versus less common**?
- Goal:
  - Make it easier to compare sizes of groups
  - Enable good choices of distributions in model building

Datasets: penguins

- `?palmerpenguins::penguins`

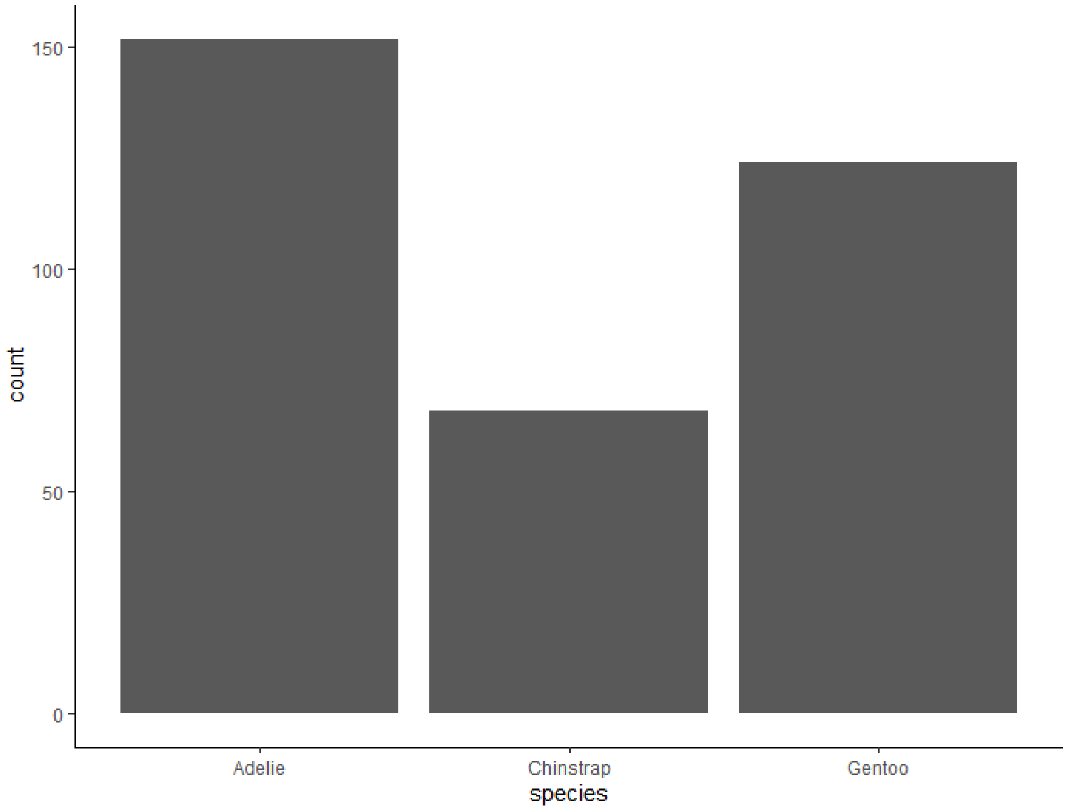


Source



One variable: Discrete

Bar chart

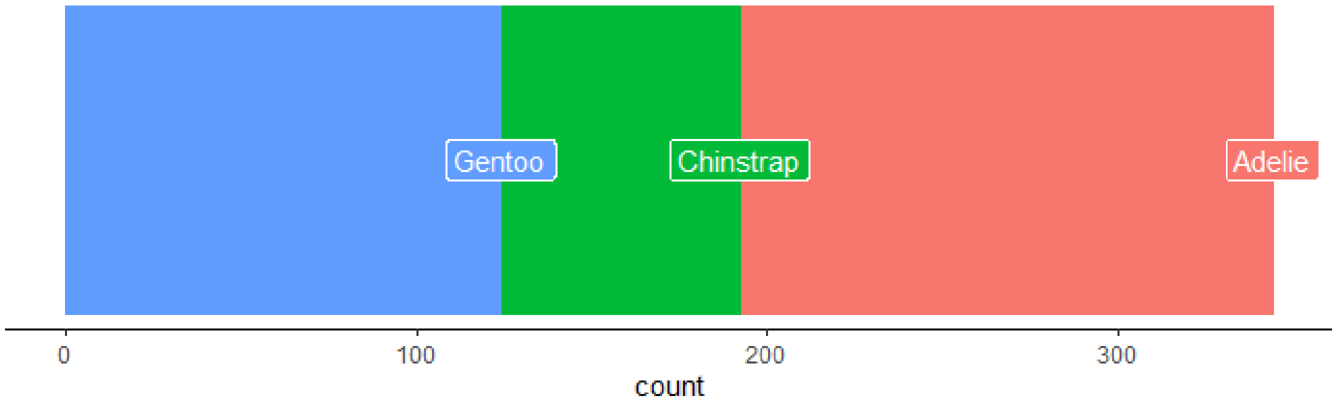



```
library(palmerpenguins)
library(ggplot2)
theme_set(theme_classic())

ggplot(penguins) +
  aes(x = species) +
  geom_bar()
```

One variable: Discrete

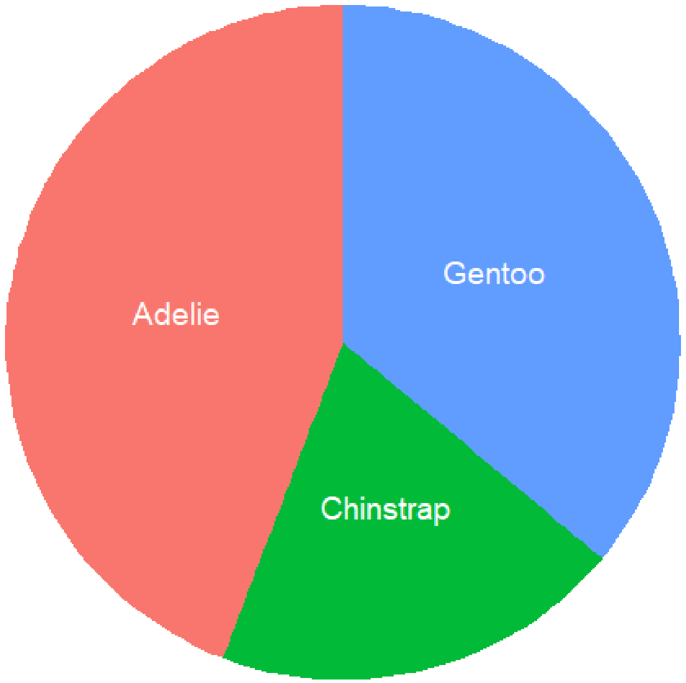
“Candybar” chart




```
ggplot(penguins) +
  aes(y = 1,
      color = species,
      fill = species,
      label = species) +
  stat_count(orientation = "y") +
  guides(y = guide_none(),
         color = guide_none(),
         fill = guide_none()) +
  ylab(NULL) +
  stat_count(geom = "label",
             color = "white")
```

Avoid pie charts!

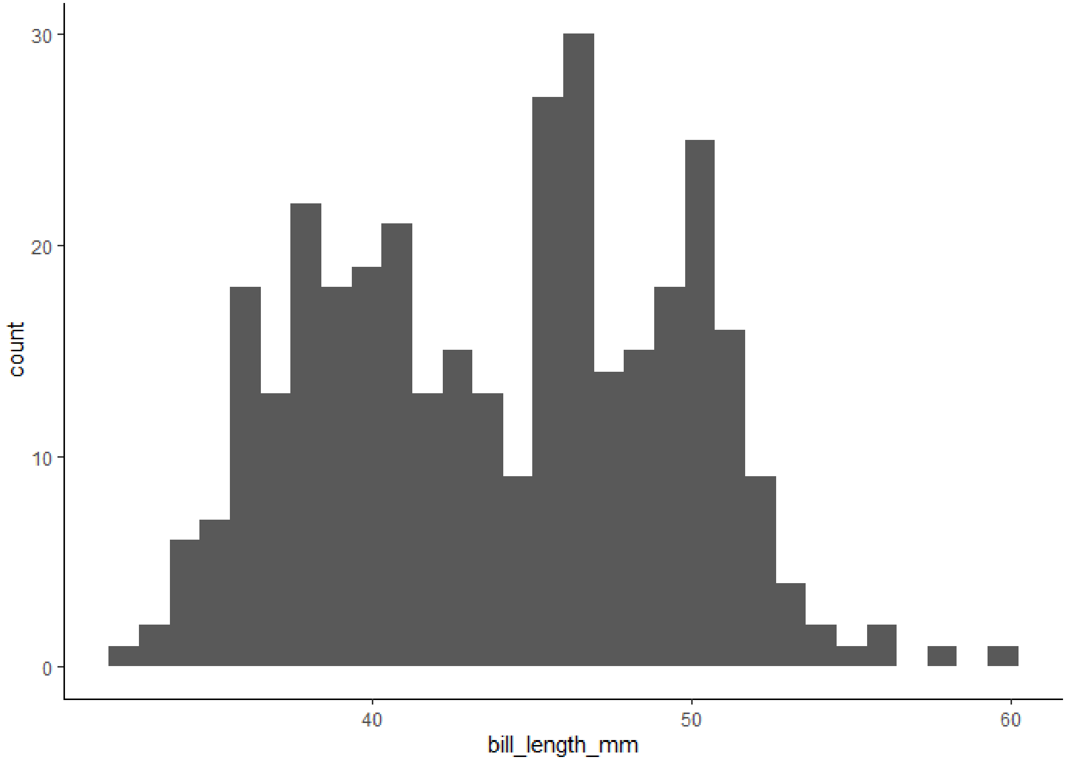
Angles are hard




```
ggplot(penguins) +
  aes(x = factor(1),
      fill = species,
      label = species) +
  geom_bar(width = 1) +
  stat_count(geom = "text",
             size = 5,
             color = "white",
             label = paste
             position = position_stack(
               vjust = .5)
             ) +
  guides(y = guide_none(),
         x = guide_none(),
         fill = guide_none()) +
  xlab(NULL) +
  ylab(NULL) +
  coord_polar(theta = "y") +
  theme(axis.text = element_blank(),
        axis.line = element_blank(),
        axis.ticks = element_blank())
```

One variable: Continuous

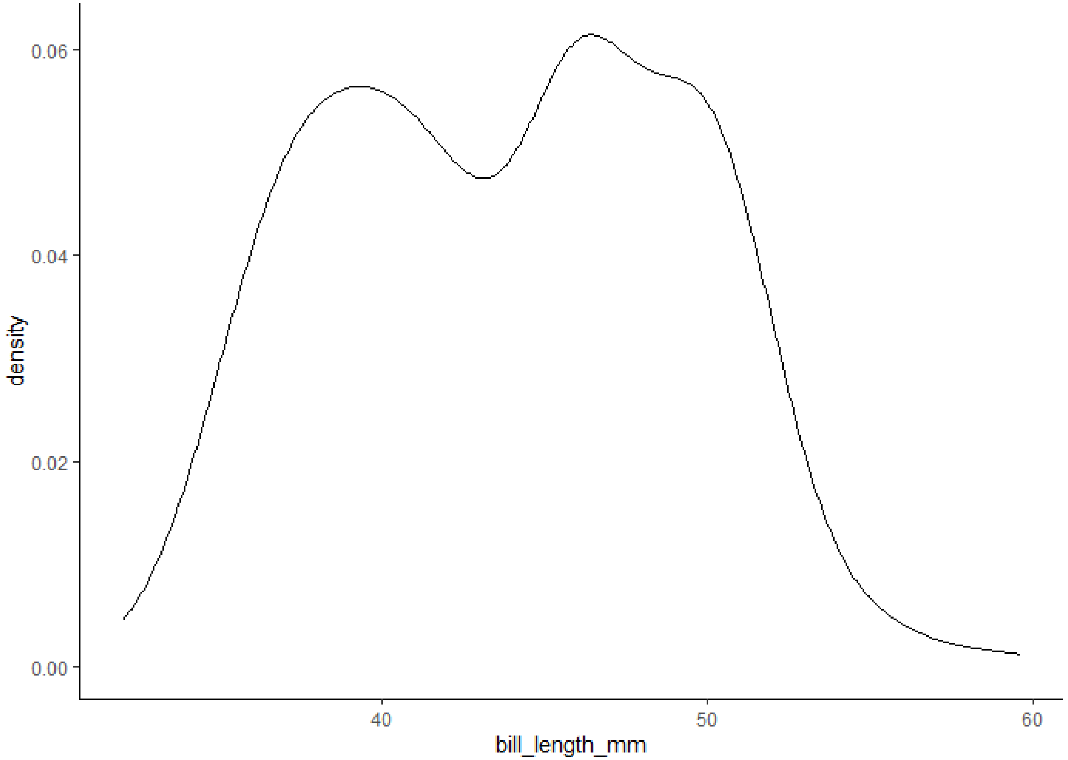
Hisotogram




```
ggplot(penguins) +
  aes(x = species) +
  geom_histogram(
    binwidth = 1
  )
```

One variable: Continuous

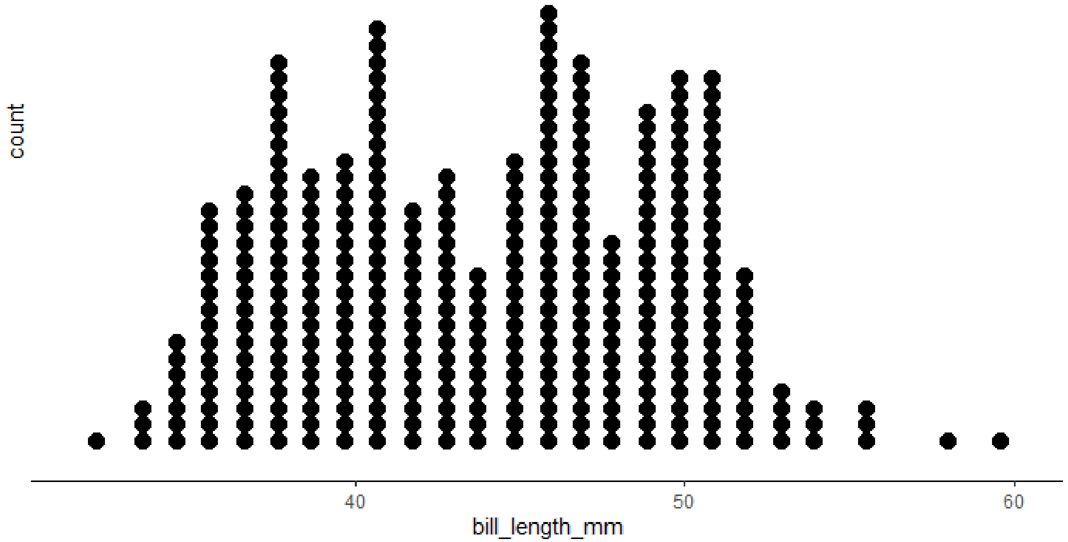
Density




```
ggplot(penguins) +
  aes(x = species) +
  geom_density()
```

One variable: Continuous

Dotplot

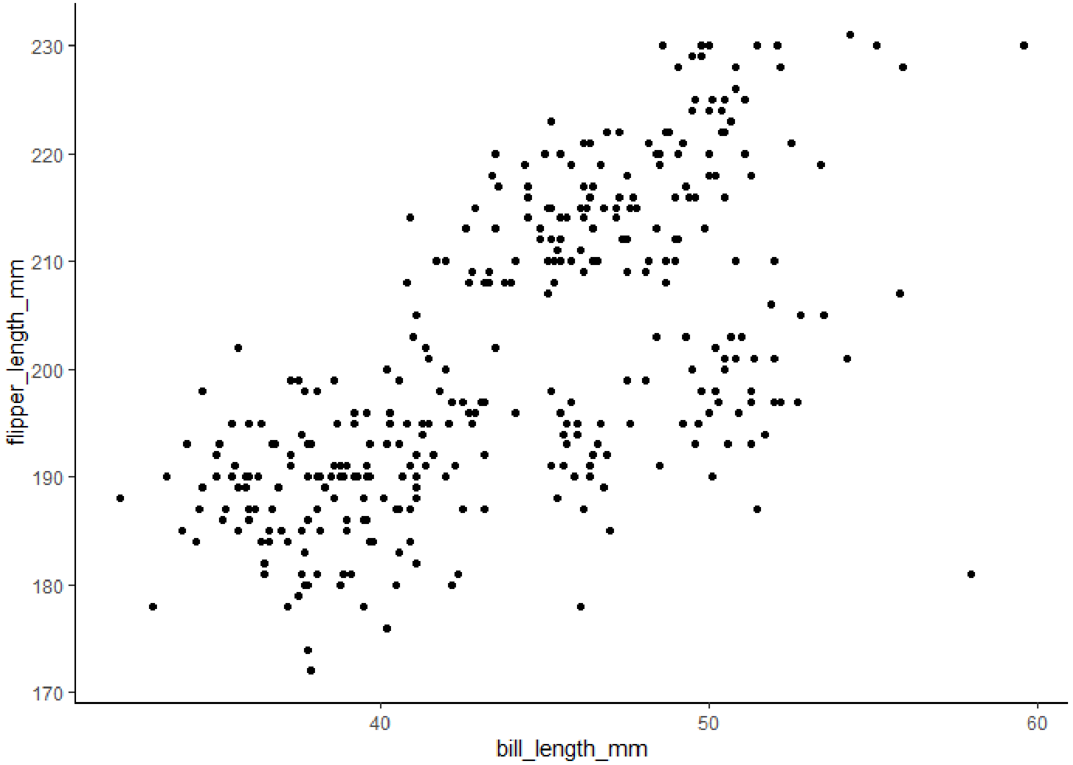



```
ggplot(penguins) +
  aes(x = bill_length_mm) +
  geom_dotplot(binwidth = 1,
               dotsize = .5) +
  guides(y = guide_none())
```

More than one variable

- Show **relationships between variables**
  - **Trends** between continuous variables
  - **Differences** across groups for discrete variables
- Goal:
  - Show **strength** of relationship
  - Show **form** of relationship
  - Identify **anomalies** (e.g., outliers/leverage points)

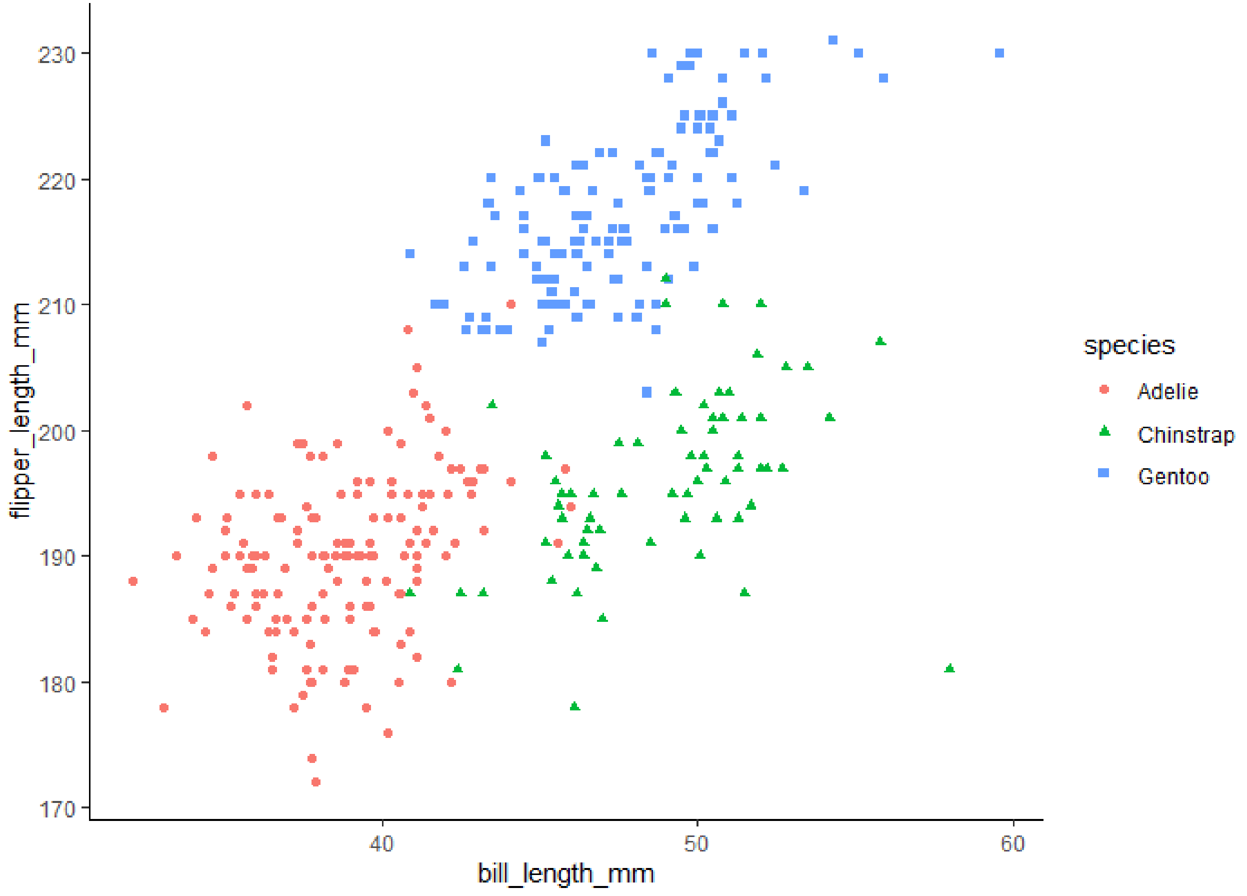
Scatterplots




```
ggplot(penguins) +
  aes(x = bill_length_mm,
      y = flipper_length_mm) +
  geom_point()
```

You can put more than two variables on a plot!

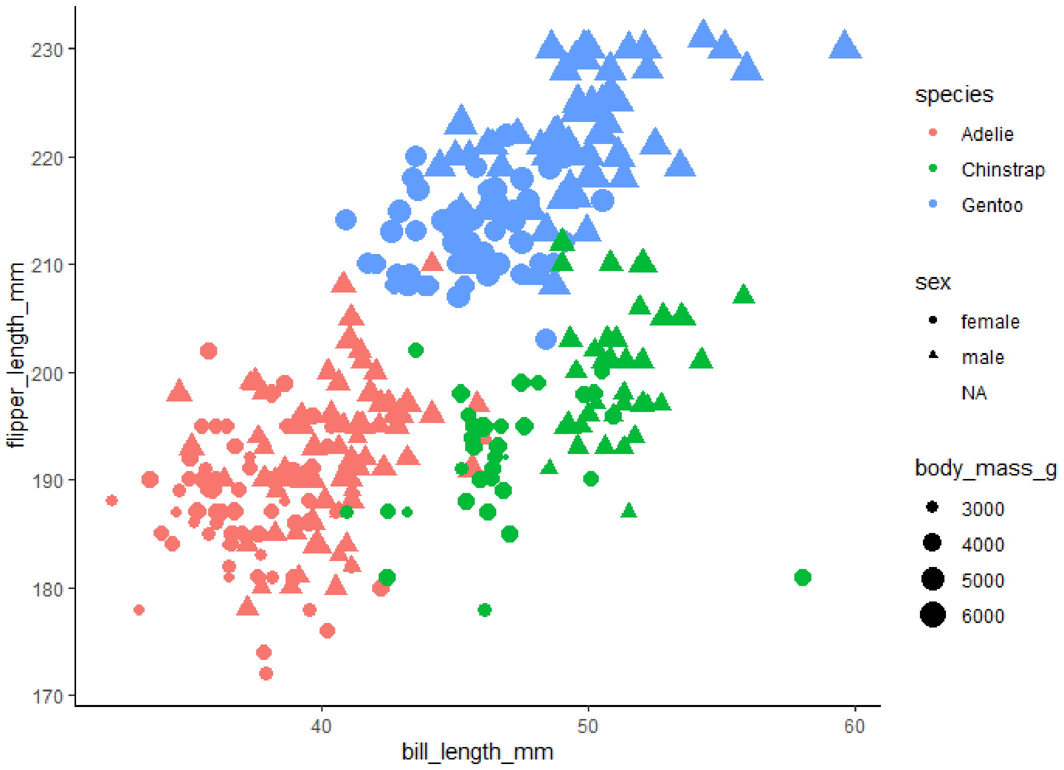
Scatterplots: Use your aesthetics!




```
ggplot(penguins) +
  aes(x = bill_length_mm,
      y = flipper_length_mm,
      fill = species,
      color = species,
      shape = species) +
  geom_point()
```

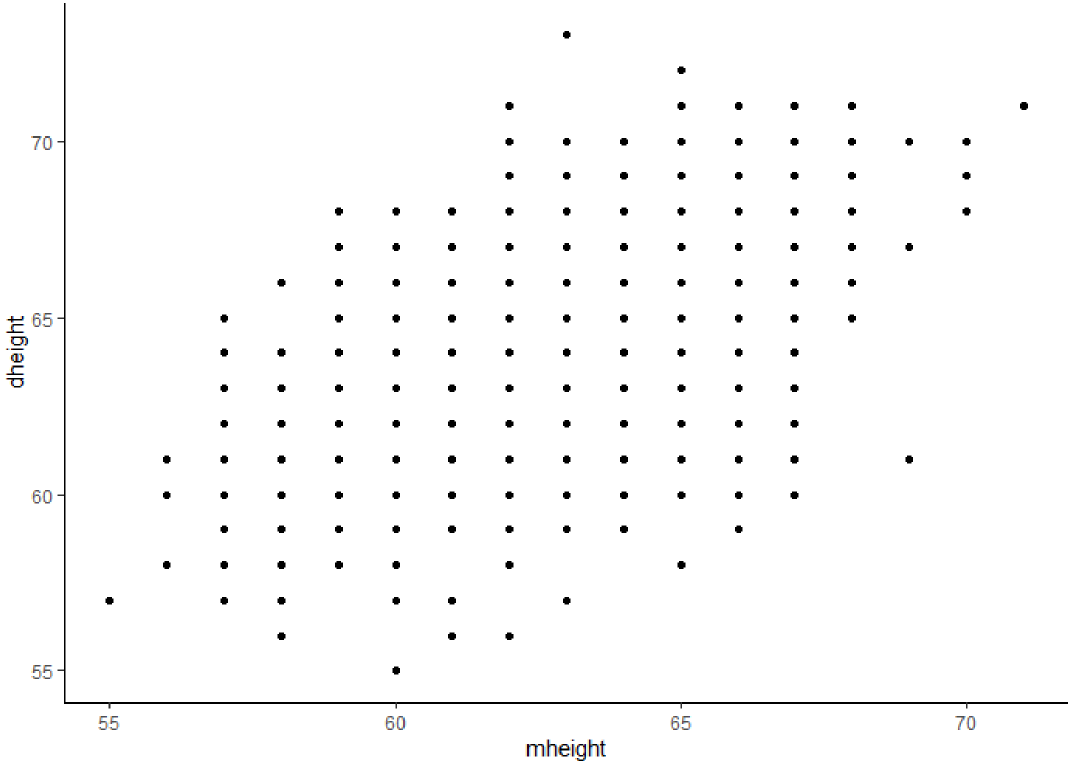

UNIVERSITY OF SOUTH FLORIDA.

Scatterplots: Don't go overboard!



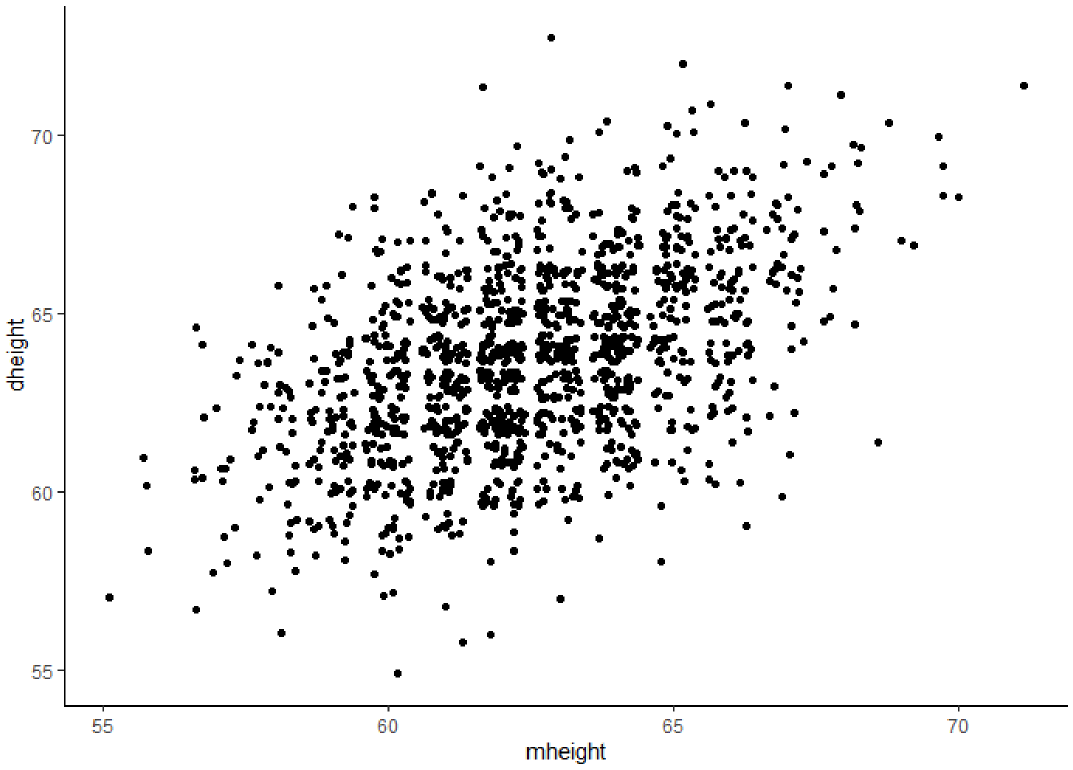

```
ggplot(penguins) +
  aes(x = bill_length_mm,
      y = flipper_length_mm,
      fill = species,
      color = species,
      shape = sex,
      size = body_mass_g) +
  geom_point()
```

Scatterplots: Overplotting



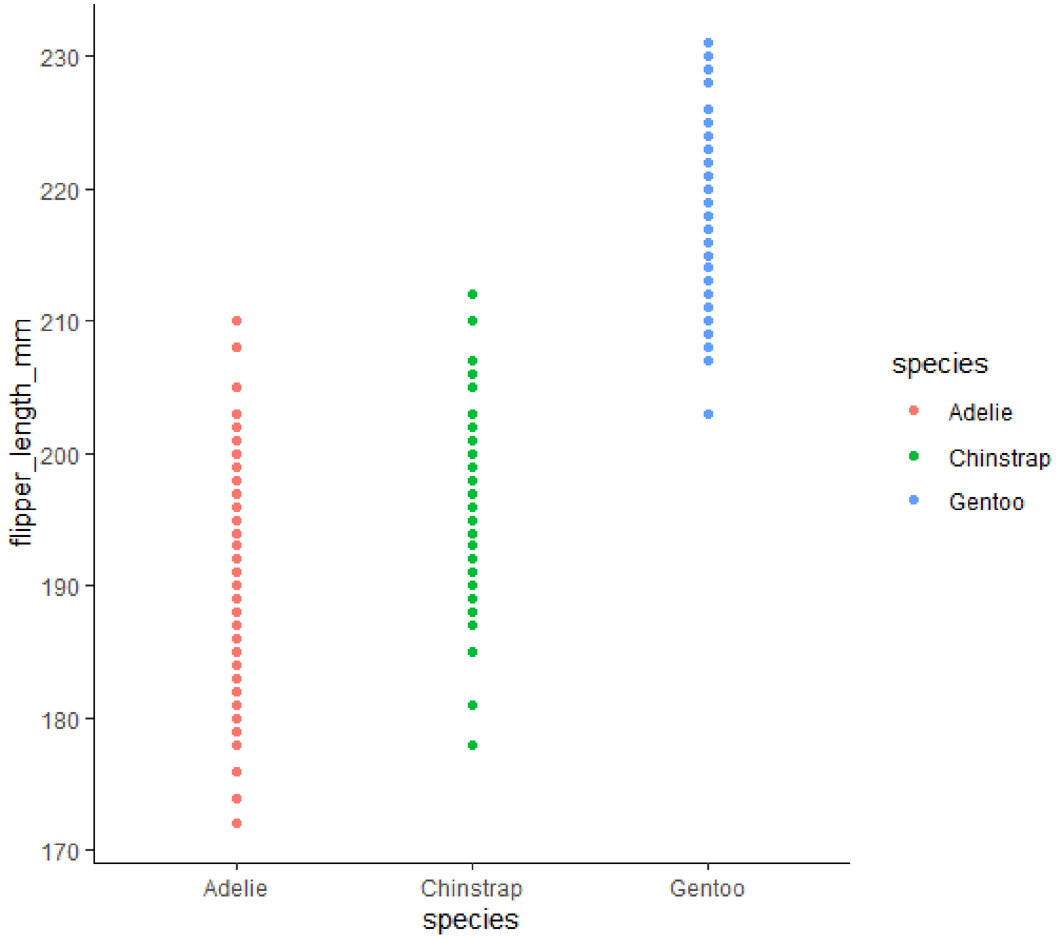

```
ggplot(round(alr4::Heights)) +
  aes(x = mheight,
      y = dheight) +
  geom_point()
```

Scatterplots: Overplotting




```
ggplot(round(alr4::Heights)) +
  aes(x = mheight,
      y = dheight) +
  geom_jitter()
```

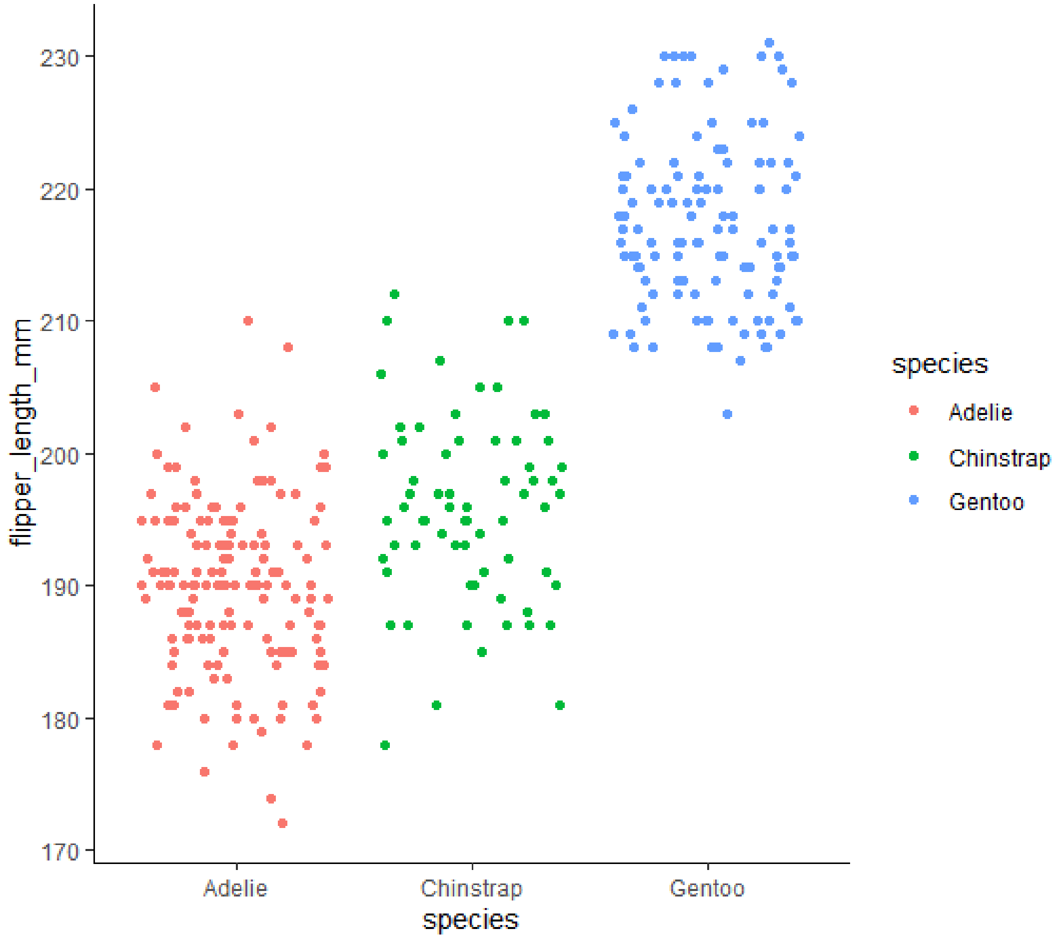
Scatterplots: Discrete variables




```
ggplot(penguins) +
  aes(x = species,
      y = flipper_length_mm,
      fill = species,
      color = species) +
  geom_point()
```

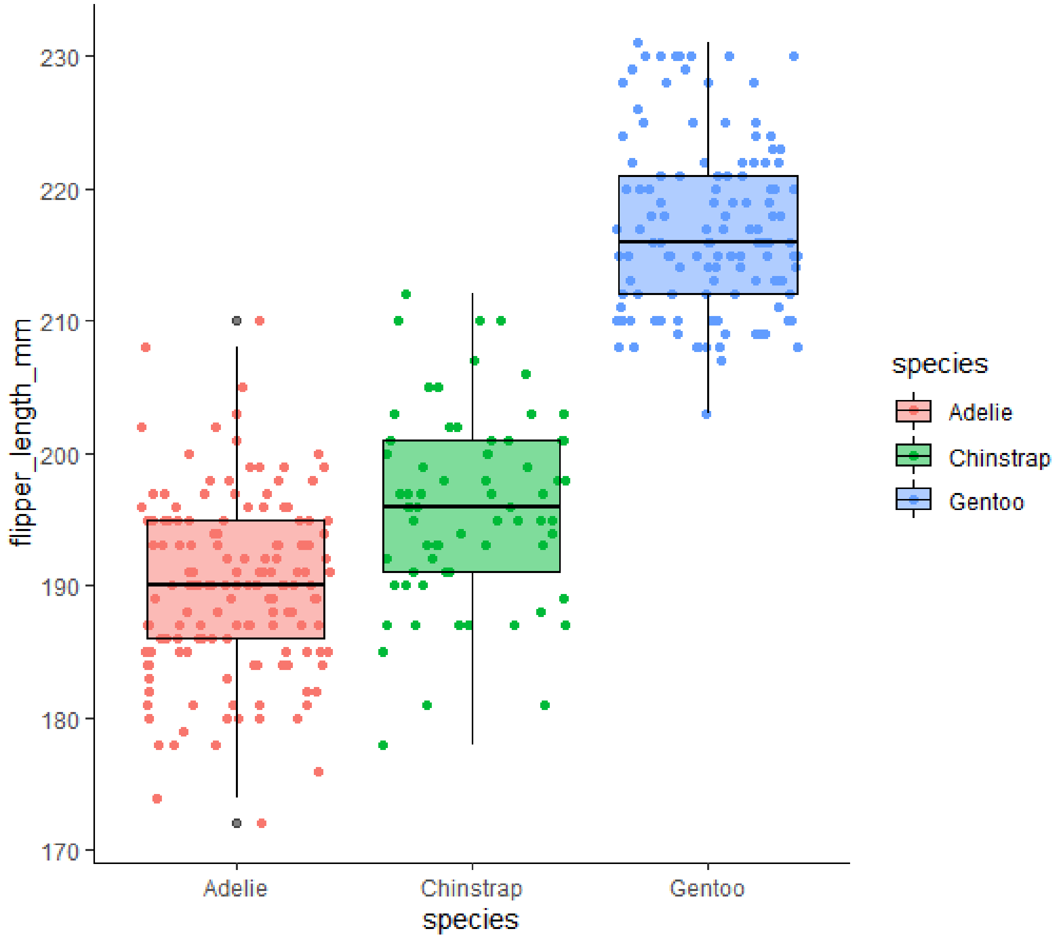

UNIVERSITY OF SOUTH FLORIDA.

Scatterplots: Discrete variables



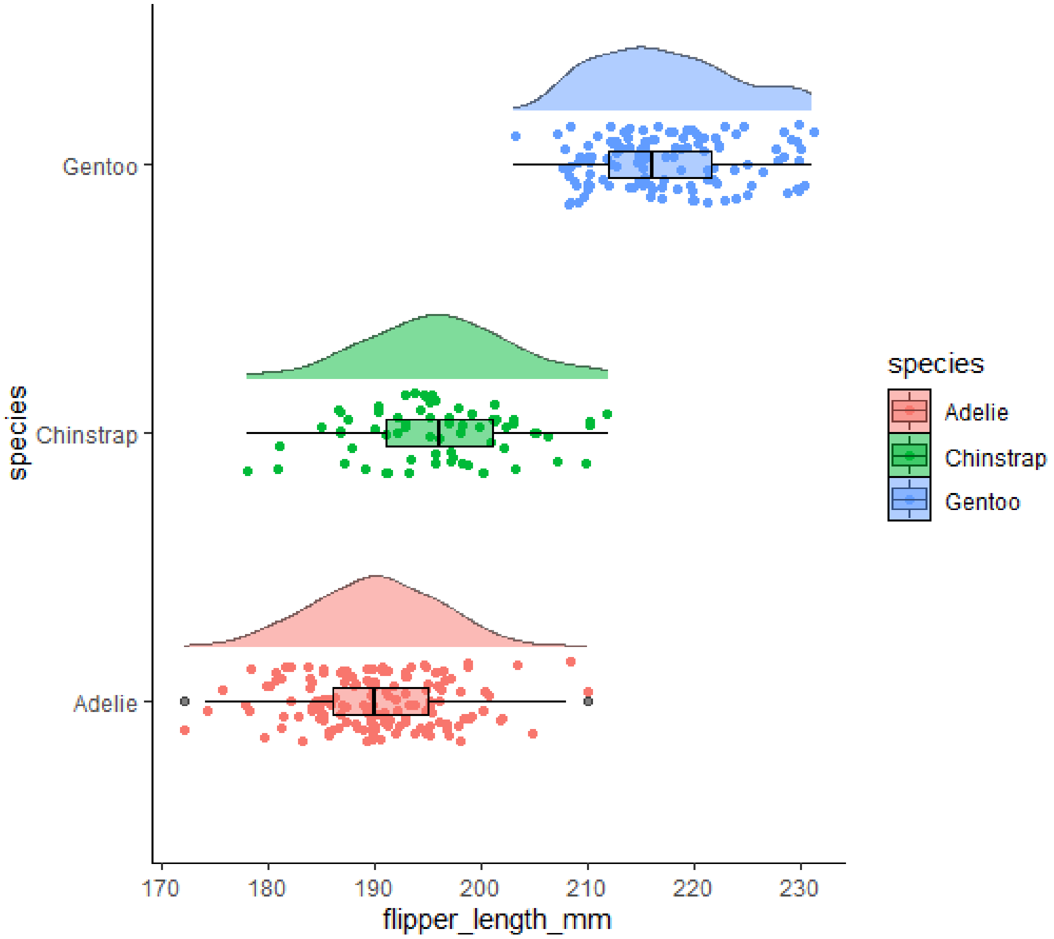

```
ggplot(penguins) +
  aes(x = species,
      y = flipper_length_mm,
      fill = species,
      color = species) +
  geom_jitter(height = 0,
              width = .4)
```

Scatterplots: Discrete variables



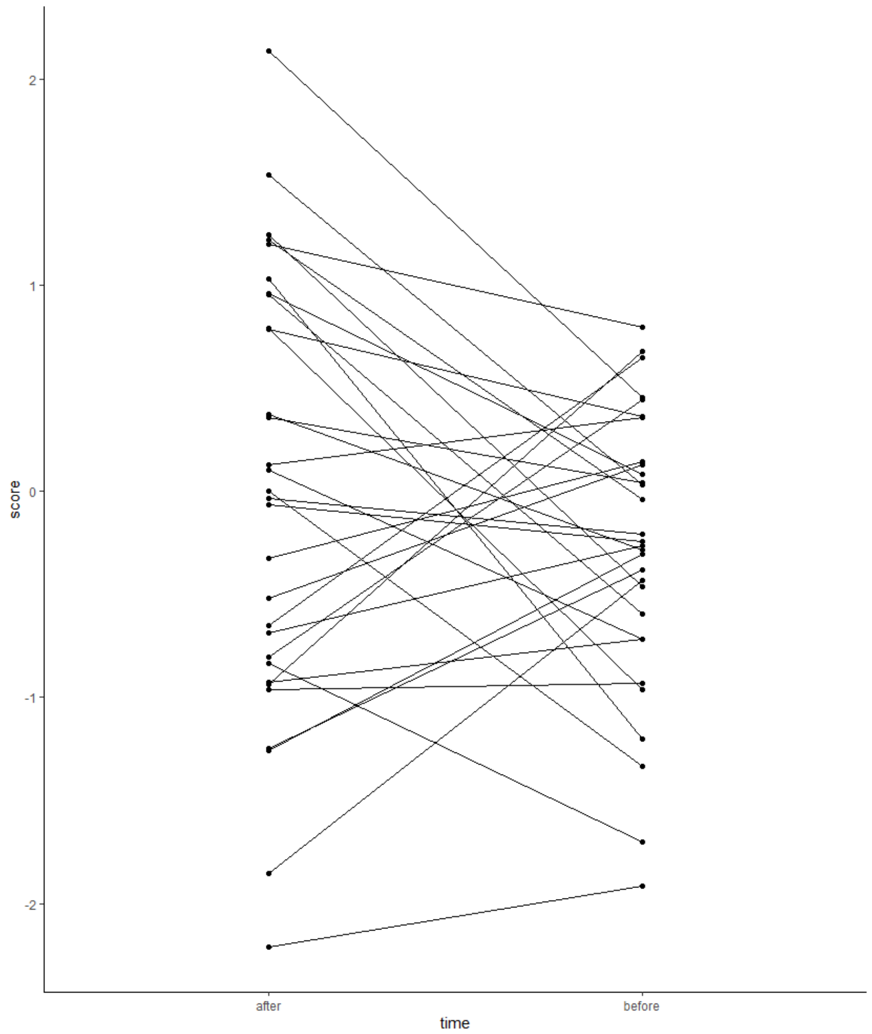

```
ggplot(penguins) +
  aes(x = species,
      y = flipper_length_mm,
      fill = species,
      color = species) +
  geom_jitter(height = 0,
              width = .4) +
  geom_boxplot(color = "black",
               alpha = .5)
```

Raincloud plots!




```
library(ggdist)
ggplot(na.omit(penguins)) +
  aes(y = species,
      x = flipper_length_mm,
      fill = species,
      color = species) +
  geom_jitter(height = .15) +
  geom_boxplot(color = "black",
               alpha = .5,
               width = .1,
               size = .5) +
  stat_slab(height = .3,
            color = "black",
            size = .2,
            alpha = .5,
            position = position_nudge(y = .2))
```

Scatterplots for change

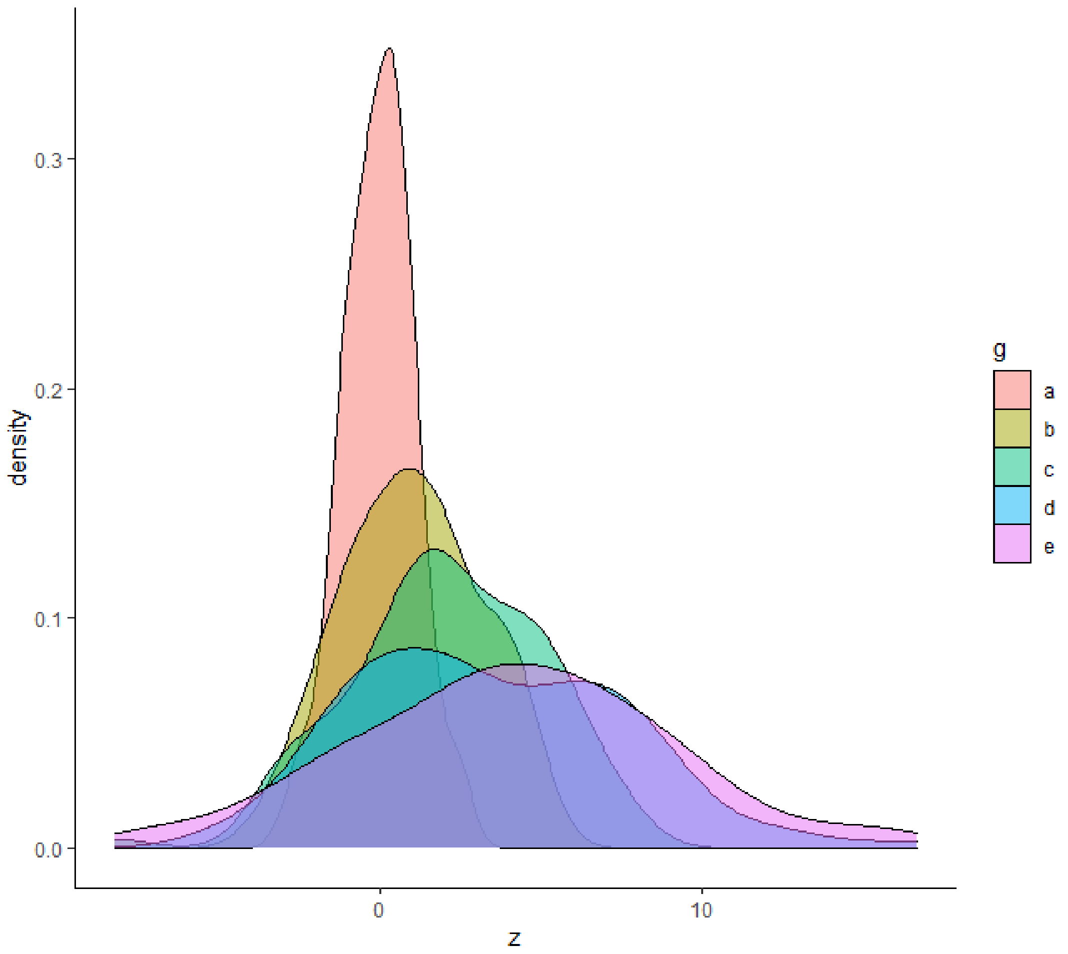



```
df <- data.frame(
  id = 1:30,
  before = rnorm(30),
  after = rnorm(30))
df <- tidyr::pivot_longer(
  df,
  -id,
  names_to = "time",
  values_to = "score")
ggplot(df) +
  aes(x = time,
      y = score,
      group = id) +
  geom_point() +
  geom_line()
```

Comparing densities

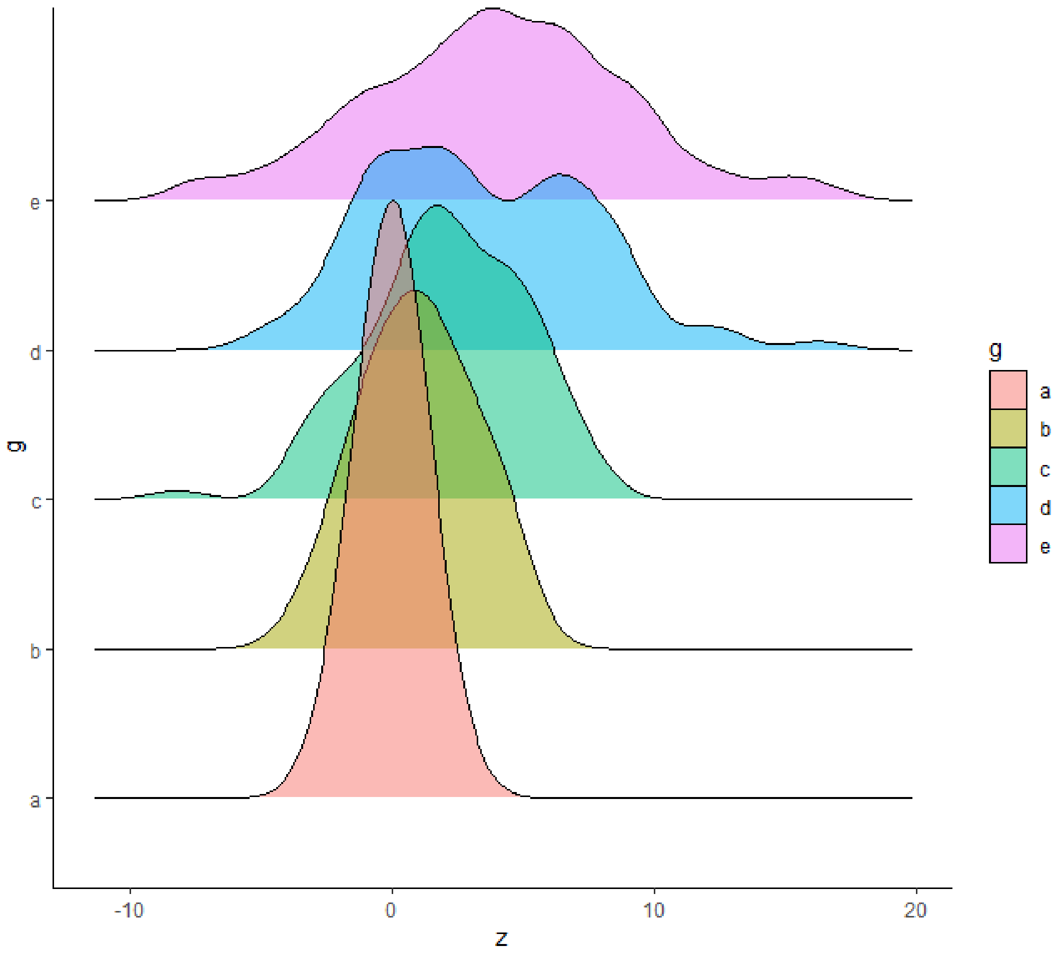
```
df <- data.frame(
  g = c(rep("a", times = 100),
        rep("b", times = 100),
        rep("c", times = 100),
        rep("d", times = 100),
        rep("e", times = 100)),
  z = c(rnorm(100, mean = 0, sd = 1),
        rnorm(100, mean = 1, sd = 2),
        rnorm(100, mean = 2, sd = 3),
        rnorm(100, mean = 3, sd = 4),
        rnorm(100, mean = 4, sd = 5))
  )
```

Comparing densities




```
ggplot(df) +
  aes(x = z,
      group = g,
      fill = g) +
  geom_density(size = .2,
               alpha = .5)
```

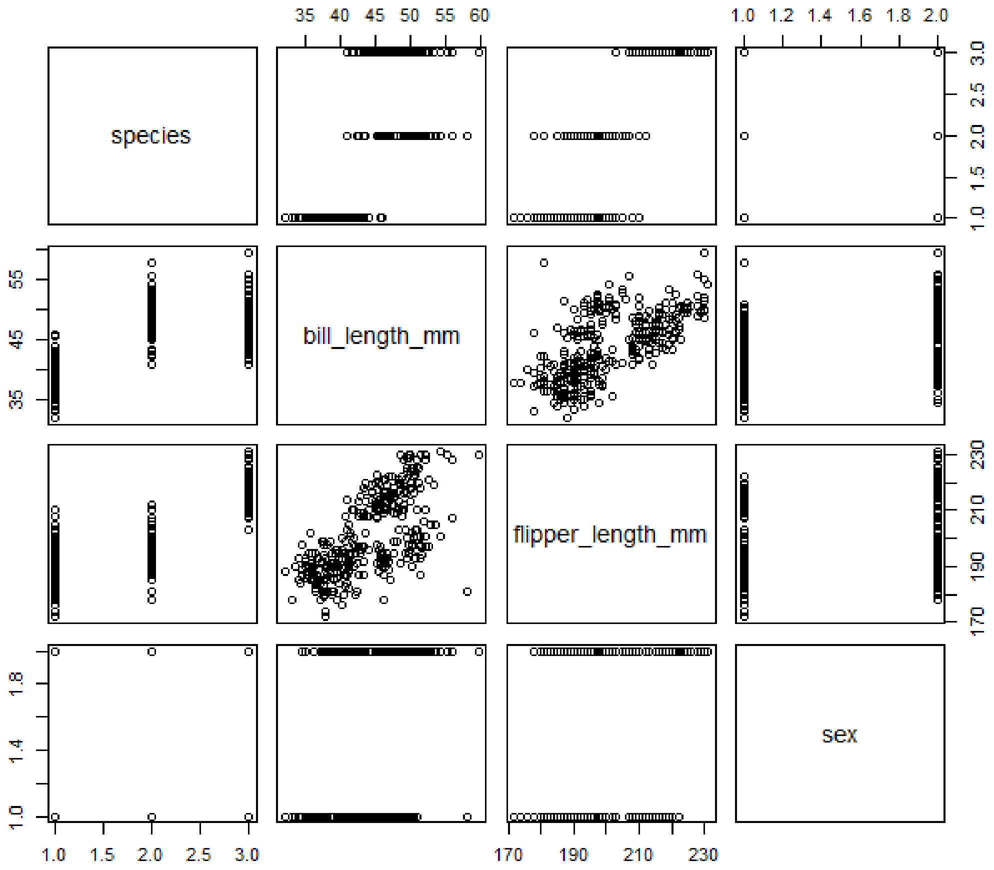
Comparing densities




```
library(ggridges)
ggplot(df) +
  aes(x = z,
      y = g,
      fill = g) +
geom_density_ridges(
    size = .2,
    alpha = .5,
    scale = 4
    )
```


USF UNIVERSITY OF SOUTH FLORIDA.

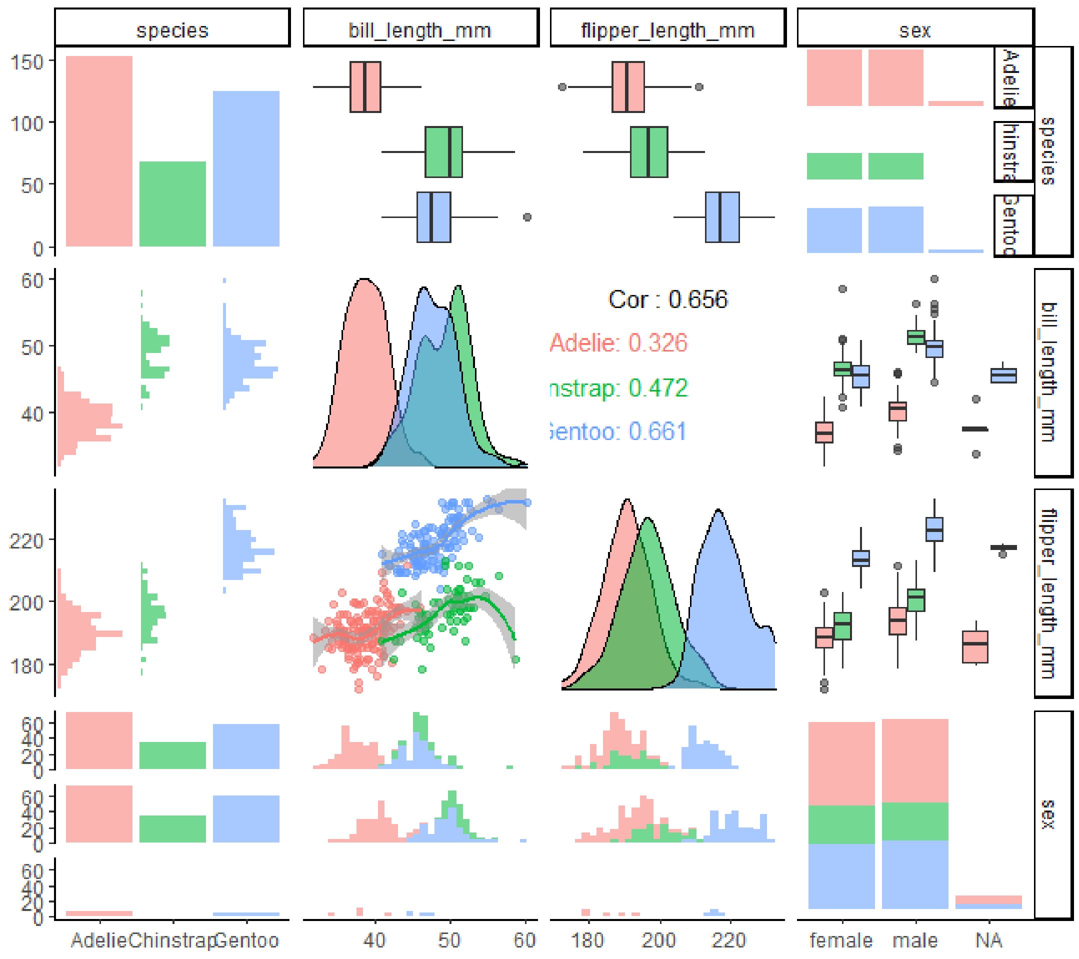
Scatterplot matrix




```r
penguins_focal <-
  penguins[, c("species",
               "bill_length_mm",
               "flipper_length_mm",
               "sex")]

pairs(penguins_focal)
```

Scatterplot matrix




```
GGally::ggpairs(
  penguins_focal,
  aes(color = species,
      alpha = .5),
  lower = list(
    continuous = "smooth_loess",
    combo = "facethist",
    discrete = "facetbar",
    na = "na")
)
```

Exploratory data analysis with smoothers

What is “Regression”

- Make the best prediction about cases given available data

- Conditional distribution
  - What is the distribution of Y given other information we know?
  - Mean + Variance

Mean Function

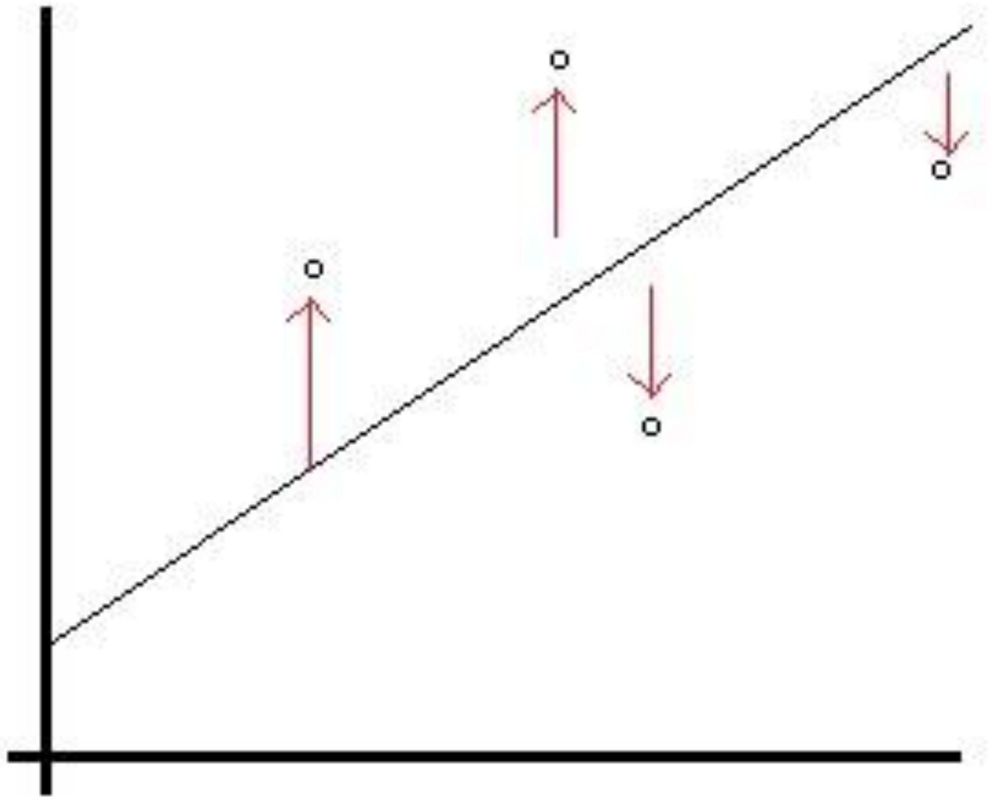
- E(Y_i | X_i)
  - E() is the "expectation" or "expected value" function
  - Read: "What is the expected value of Y, given X?"
  - If I know your score on X, what's my best guess for your level on Y?
  - We want to choose the mean function for *Y* that minimizes how wrong our predictions are

- Why the mean?
  - We will focus most of the semester on predicting the **mean** of *Y*
  - Other estimators of the center of a distribution exist (e.g., median), but the mean has many nice properties
  - The mean is the value that minimizes the squared prediction errors

Regression in a nutshell

Conceptually,

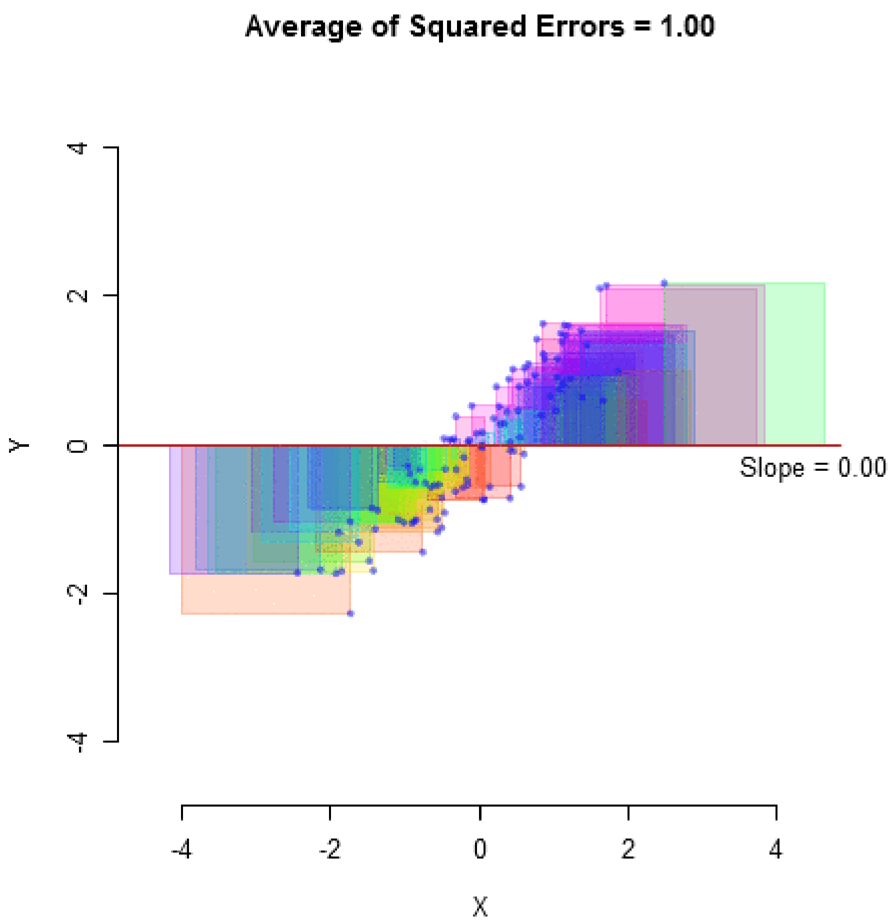
1. Make a scatterplot with response (Y) on y-axis, predictor (X) on x-axis
2. Draw a line
3. Measure the **vertical distance** from each point to that line
4. Move the line until you make those distances as small as possible
   - Technically, until the square of those distances are minimized



Least Squares Mean Function

- Classic Null model:
  - No relationship
  - Best prediction is always $\hat{\mu}$ = mean(Y_i)
  - Huge amounts of error

- If we rotate the line, we can find an angle that will make our errors much smaller
  - How much do we need to rotate this line?
  - How complex does the formula for the line need to be?



Variance Function

- Var(Y_i | X_i)
  - Read: "What is the variance of Y, given X?"
  - If I know your score on X, how **uncertain** am I about your score on Y?
  - If I predict your Y score, **how wrong** will I be on average?

- This is called "residual variance" or "residual error"
  - $y_i = y_i + \varepsilon_i$
  - Observed y_i = predicted y_i + error
  - $\mathrm{Var}(Y_i \mid X_i) = \mathrm{Var}(\varepsilon_i)$

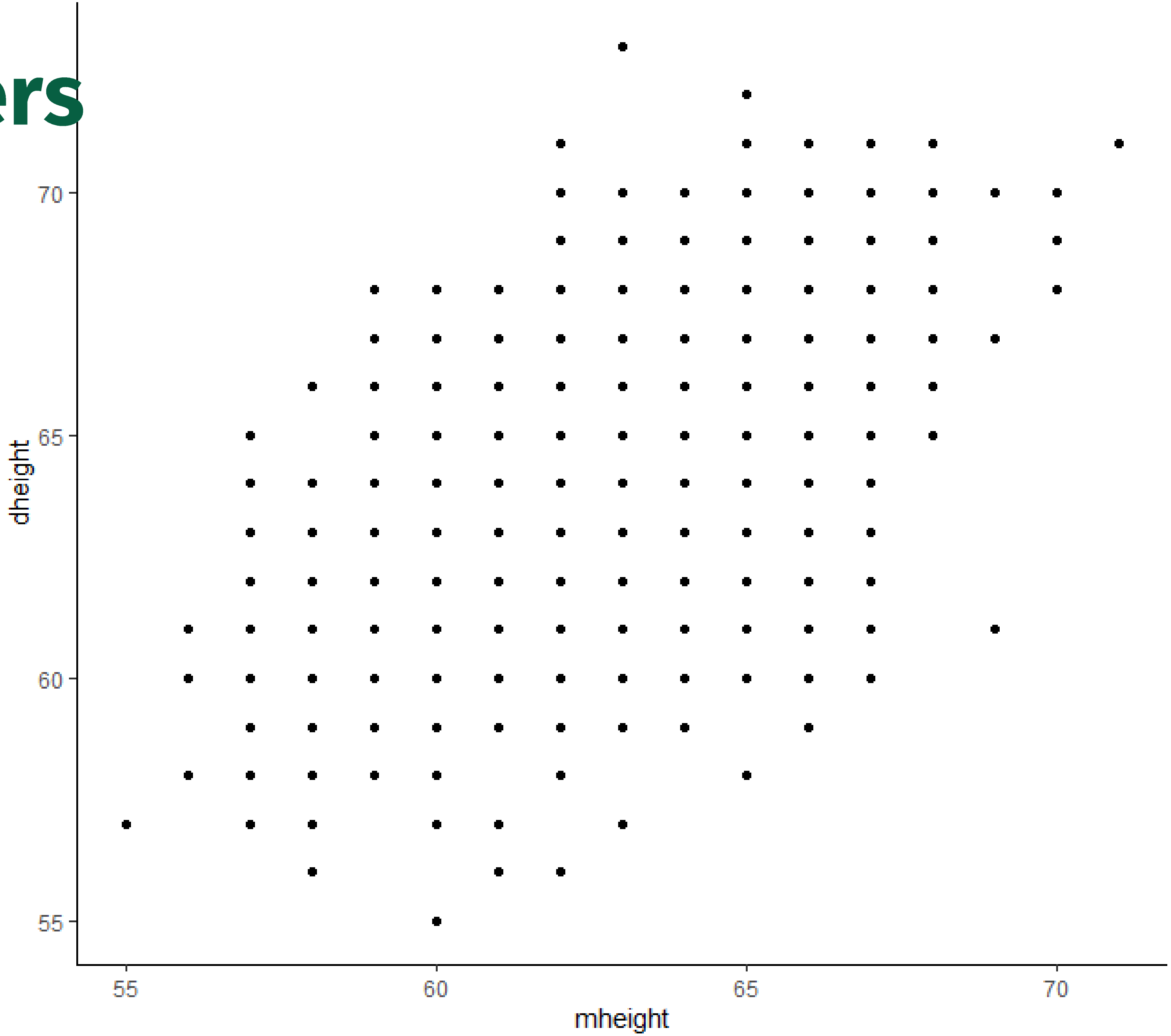
Exploratory data analysis with smoothers

- Drawing straight lines requires some strong assumptions
  - e.g., linearity 😜
- It's best to start off our exploratory analyses with something more flexible

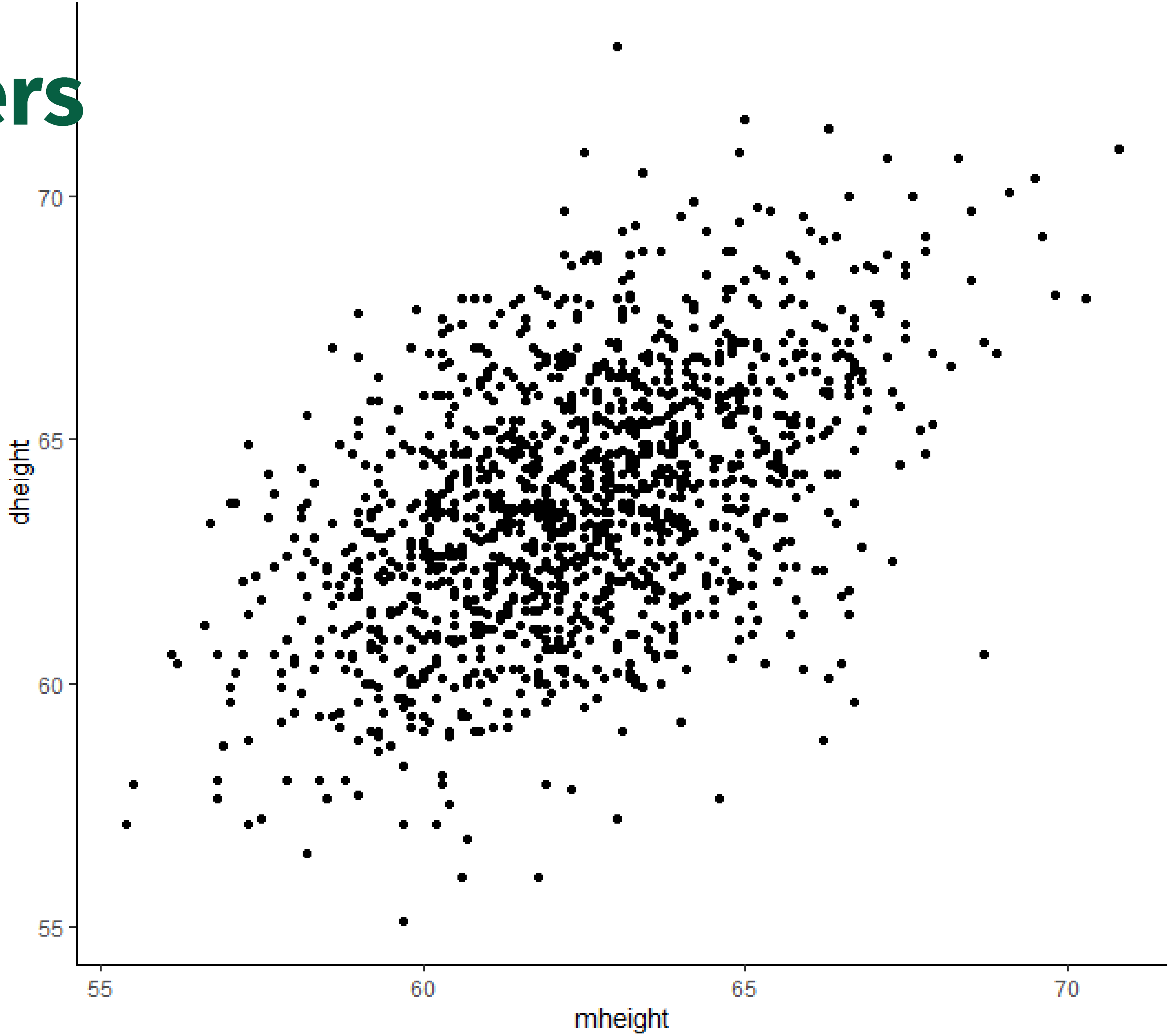
- A **smoother** is a non-parametric method for fitting a line
  - No easy to formula can be written down
  - Think of it as letting the data fit itself

- A smoother is still regression: $E(Y_i \mid X_i)$ and $Var(Y_i \mid X_i)$

Slice Smoothers



Slice Smoothers



Datasets: mtcars

- `?datasets::mtcars`

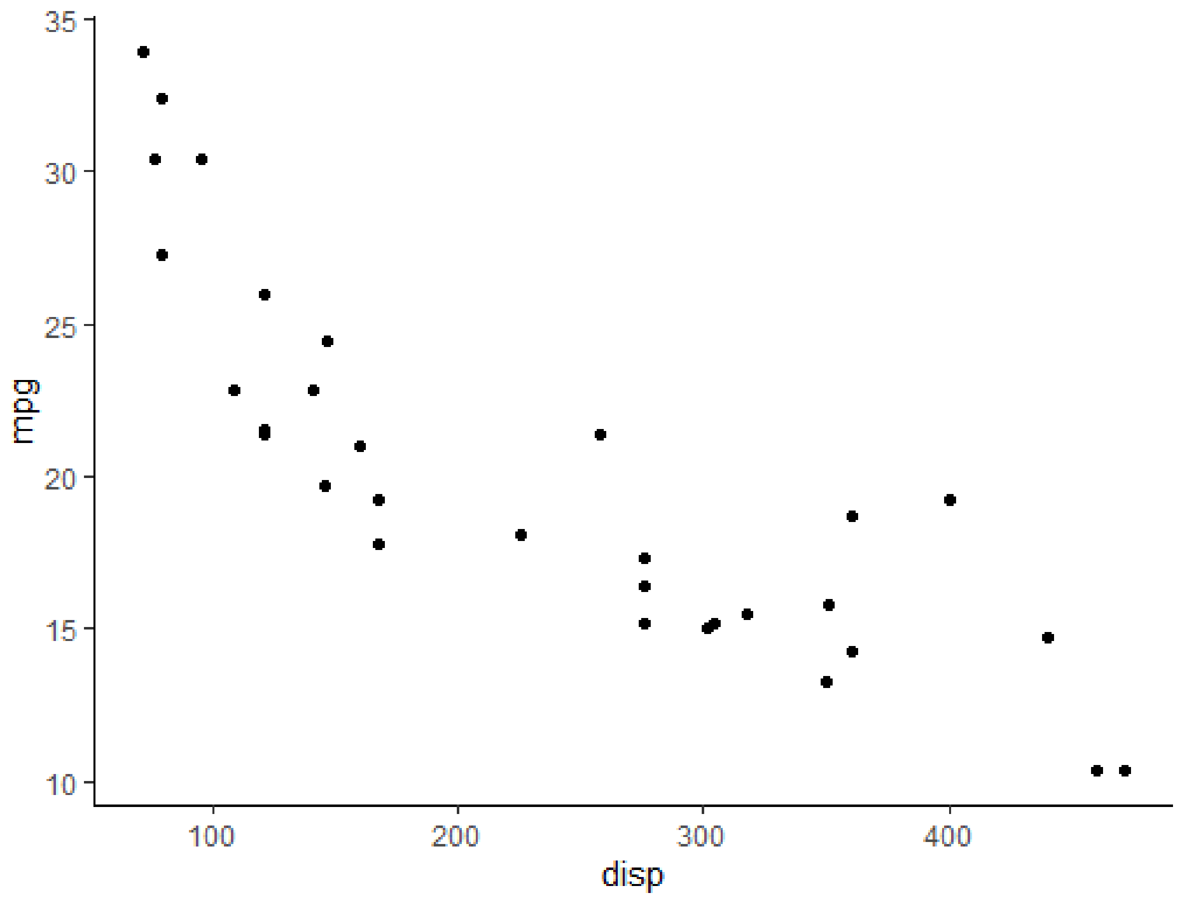
	var	Description
[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (1000 lbs)
[, 7]	qsec	1/4 mile time
[, 8]	vs	Engine (0 = V-shaped, 1 = straight)
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears

If you, like me, are not a car person, click here for an explainer.

UNIVERSITY OF SOUTH FLORIDA

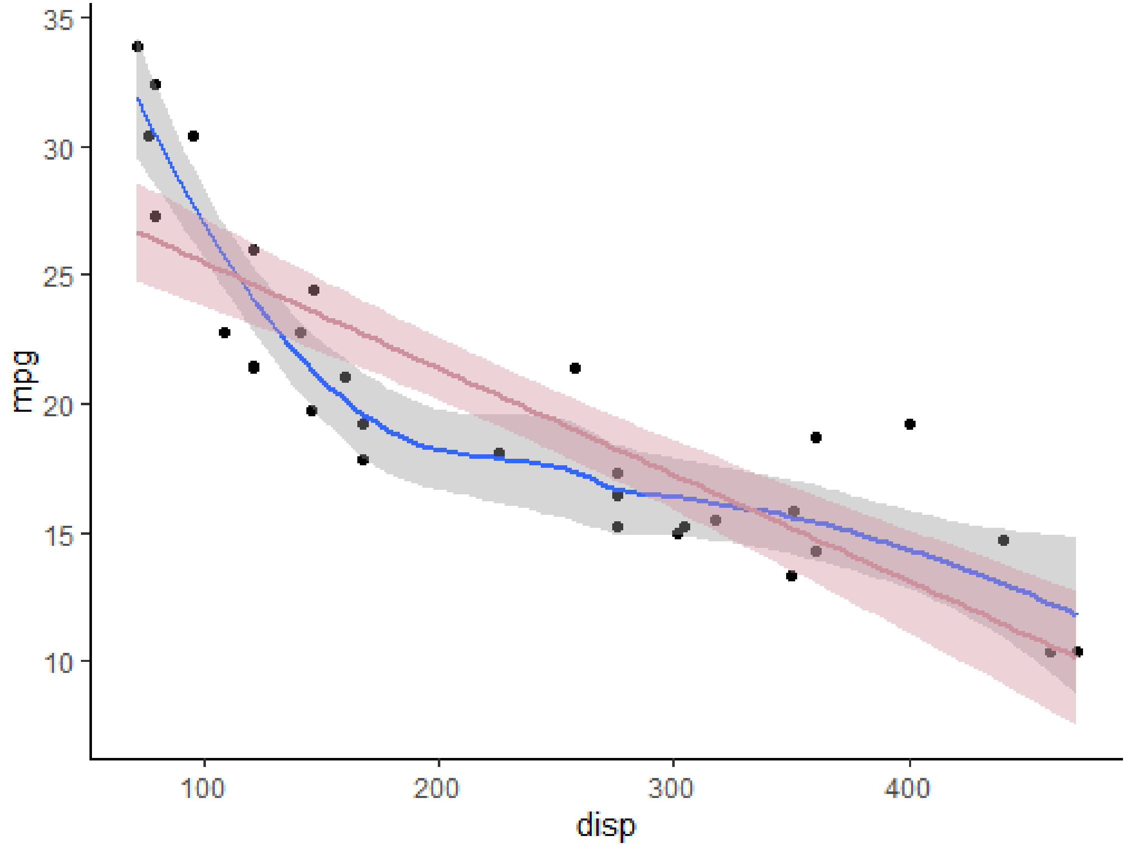
loess smoother

- Estimate E(Y_i | X_i) by:
  - For each value of x_i ,
  - select the points with that x_i value and some fraction of other points closest to it
    - (usually 66%–75%)
    - (higher % makes a “smoother” line)
  - fit a straight line model to the selected points
  - move to the next x_i and repeat



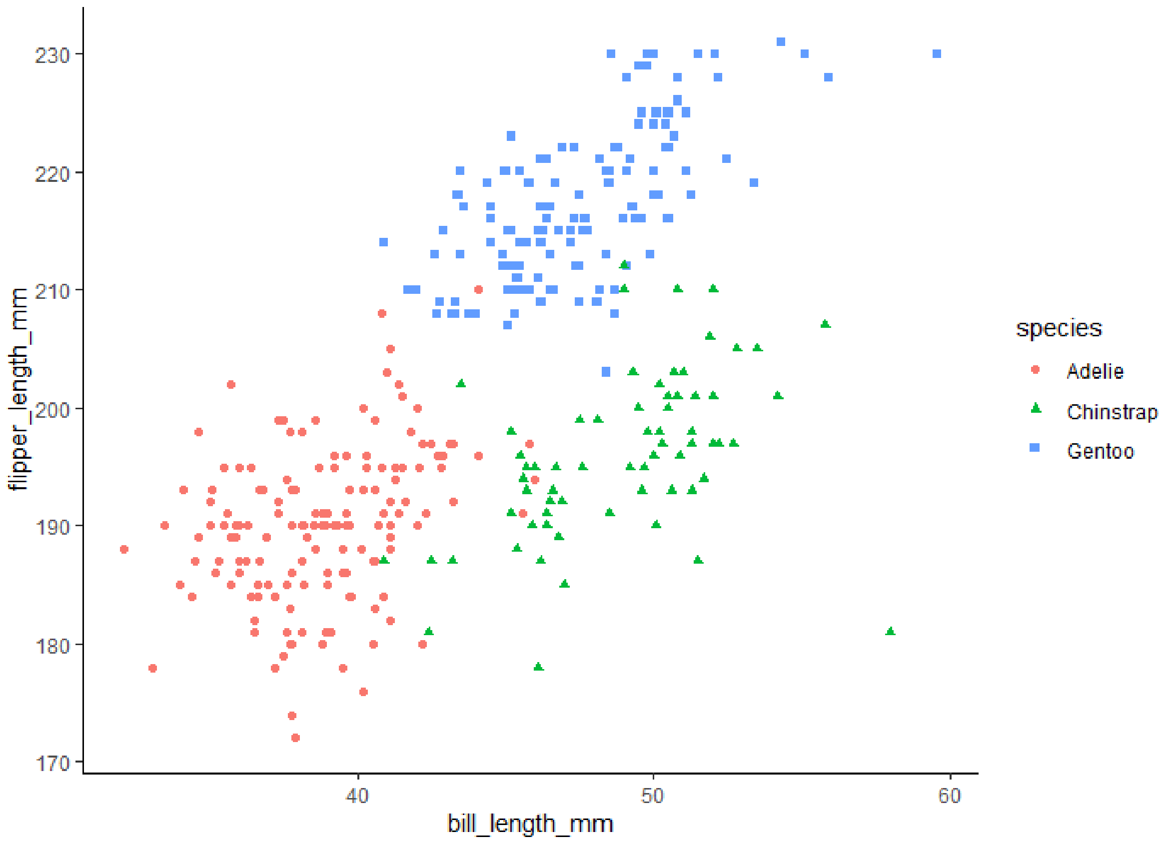
USF UNIVERSITY OF SOUTH FLORIDA.

loess smoother




```
ggplot(mtcars) +
  aes(x = disp,
      y = mpg) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method = "lm",
              color = "pink3",
              fill = "pink3")
```

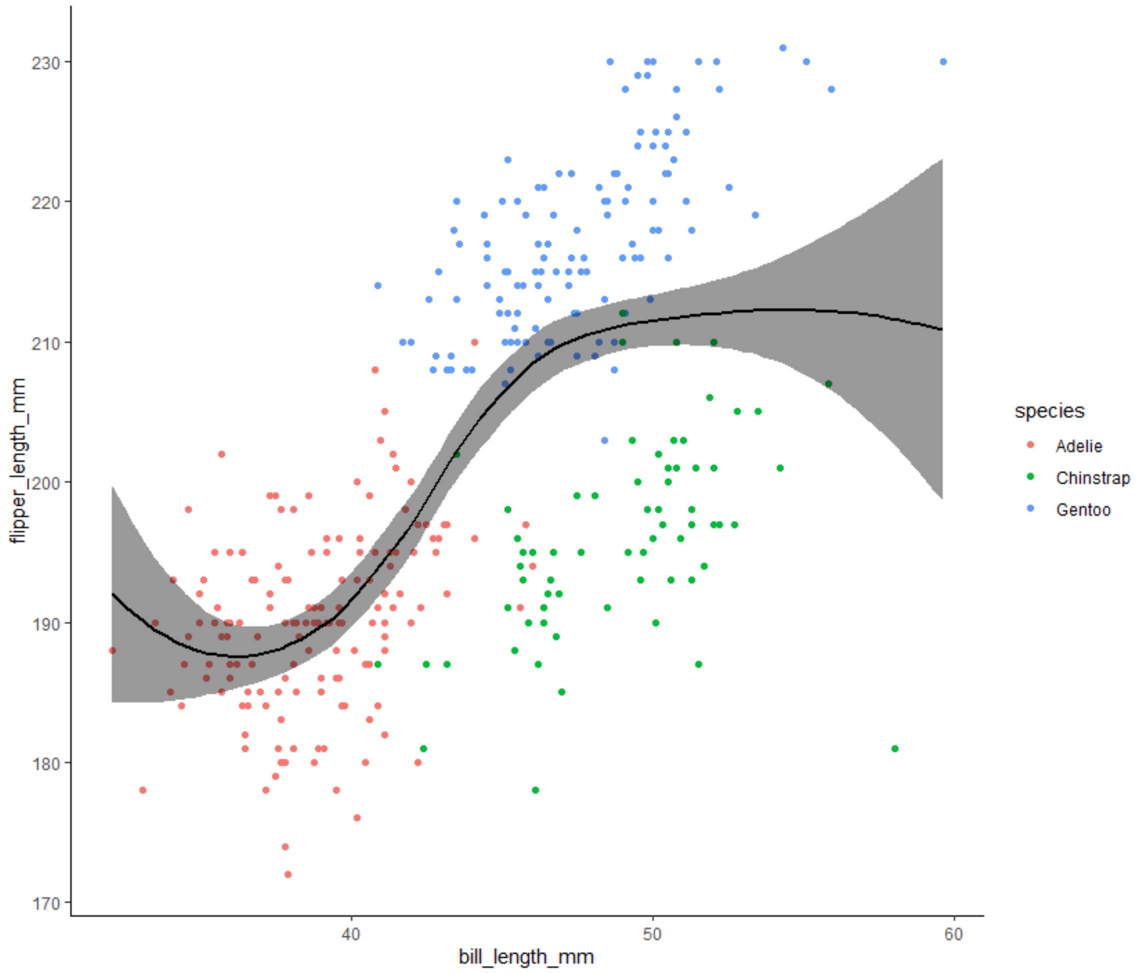
loess smoother



- loess can also help to look for differences in trends across groups
- Plot the `penguins` data

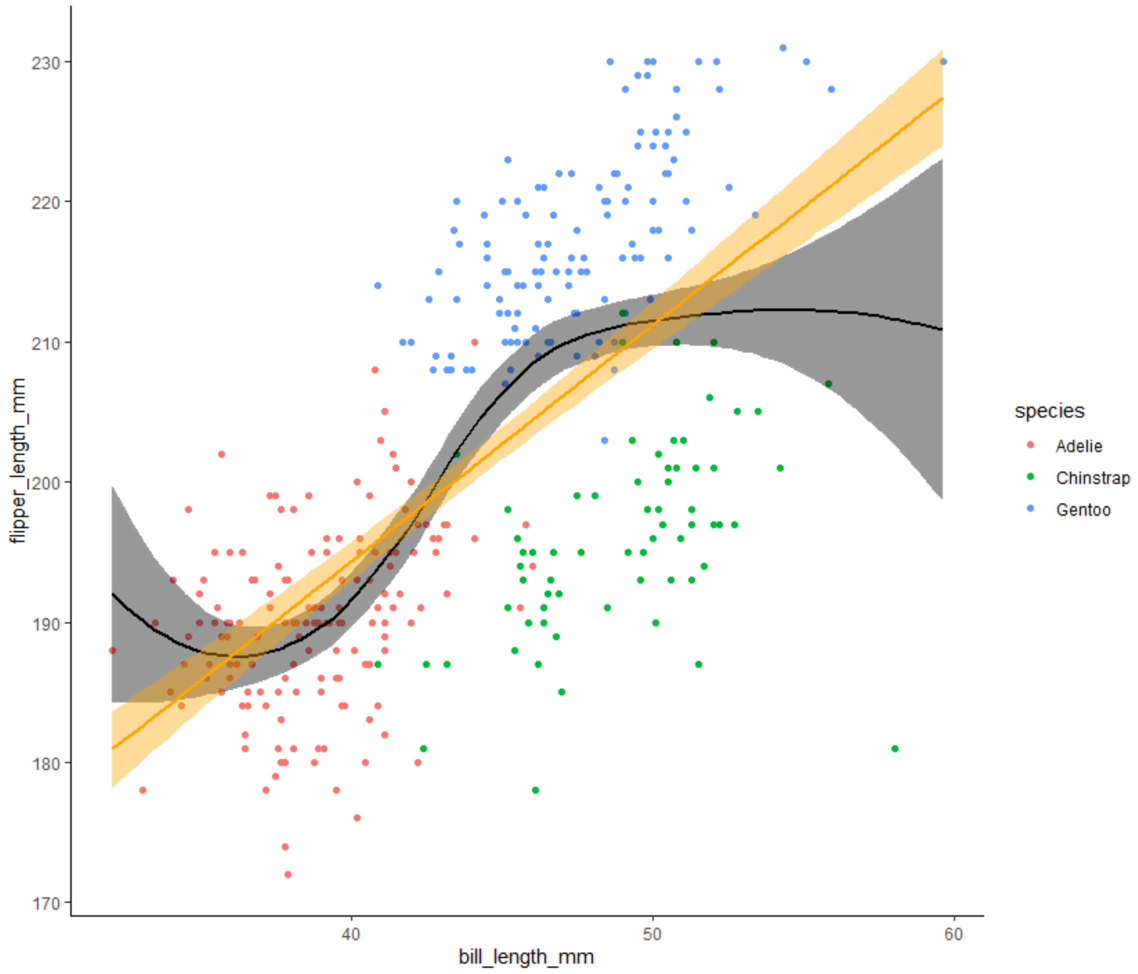
UNIVERSITY OF SOUTH FLORIDA.

loess smoother




```
ggplot(penguins) +
  aes(x = bill_length_mm,
      y = flipper_length_mm,
      fill = species,
      color = species) +
  geom_point() +
  geom_smooth(color = "black",
              fill = "black")
```

loess smoother

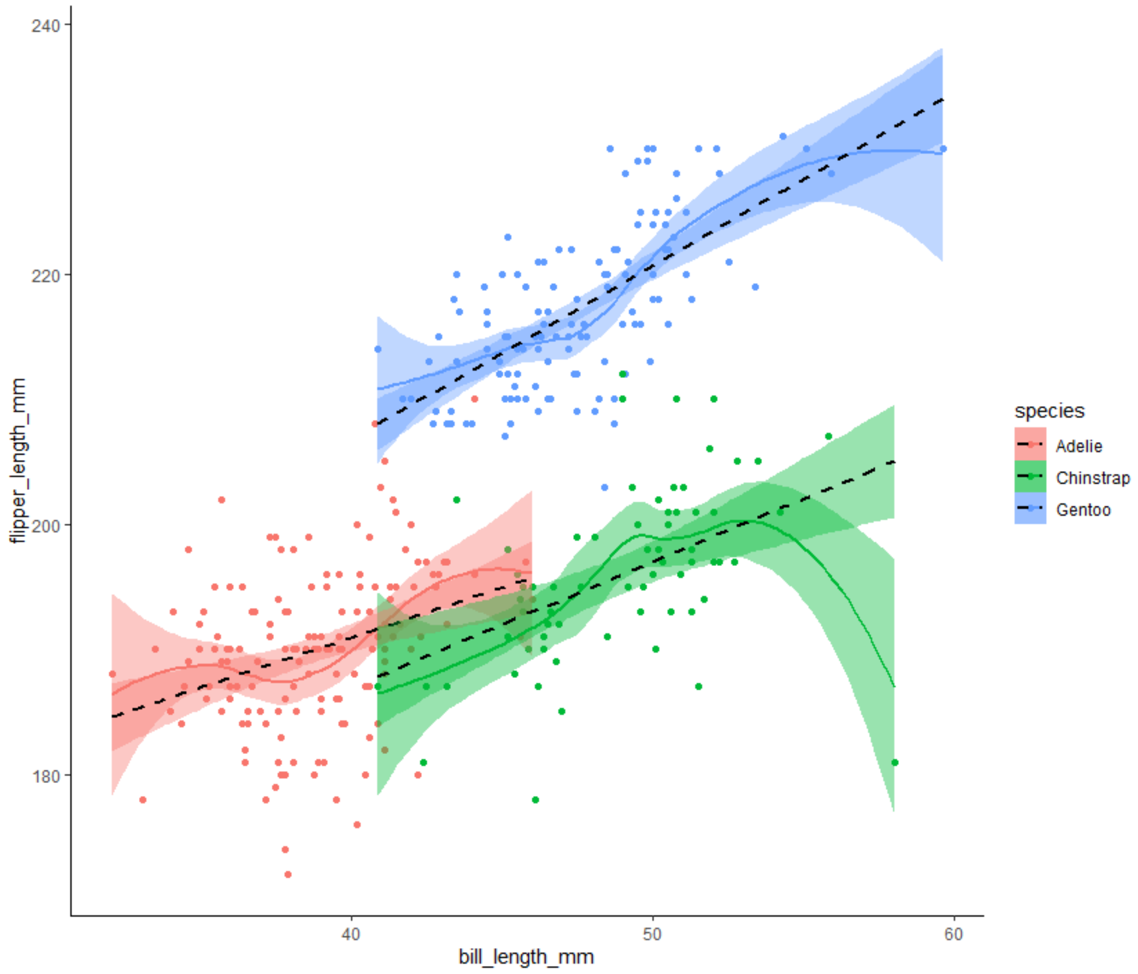



```
ggplot(penguins) +
  aes(x = bill_length_mm,
      y = flipper_length_mm,
      fill = species,
      color = species) +
  geom_point() +
  geom_smooth(color = "black",
              fill = "black") +
  geom_smooth(method = "lm",
              color = "orange",
              fill = "orange")
```

loess smoother

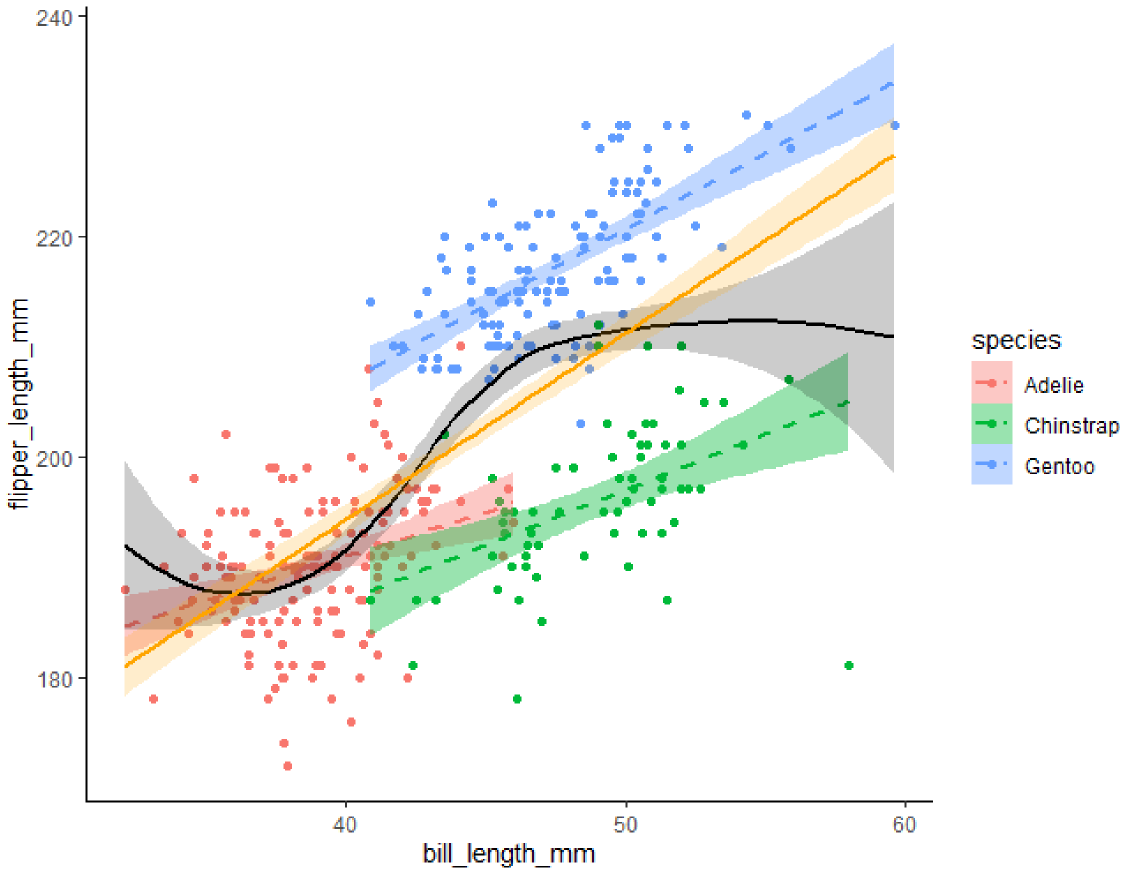

```
ggplot(penguins) +
  aes(x = bill_length_mm,
      y = flipper_length_mm,
      fill = species,
      color = species) +
  geom_point() +
  geom_smooth()
```

loess smoother



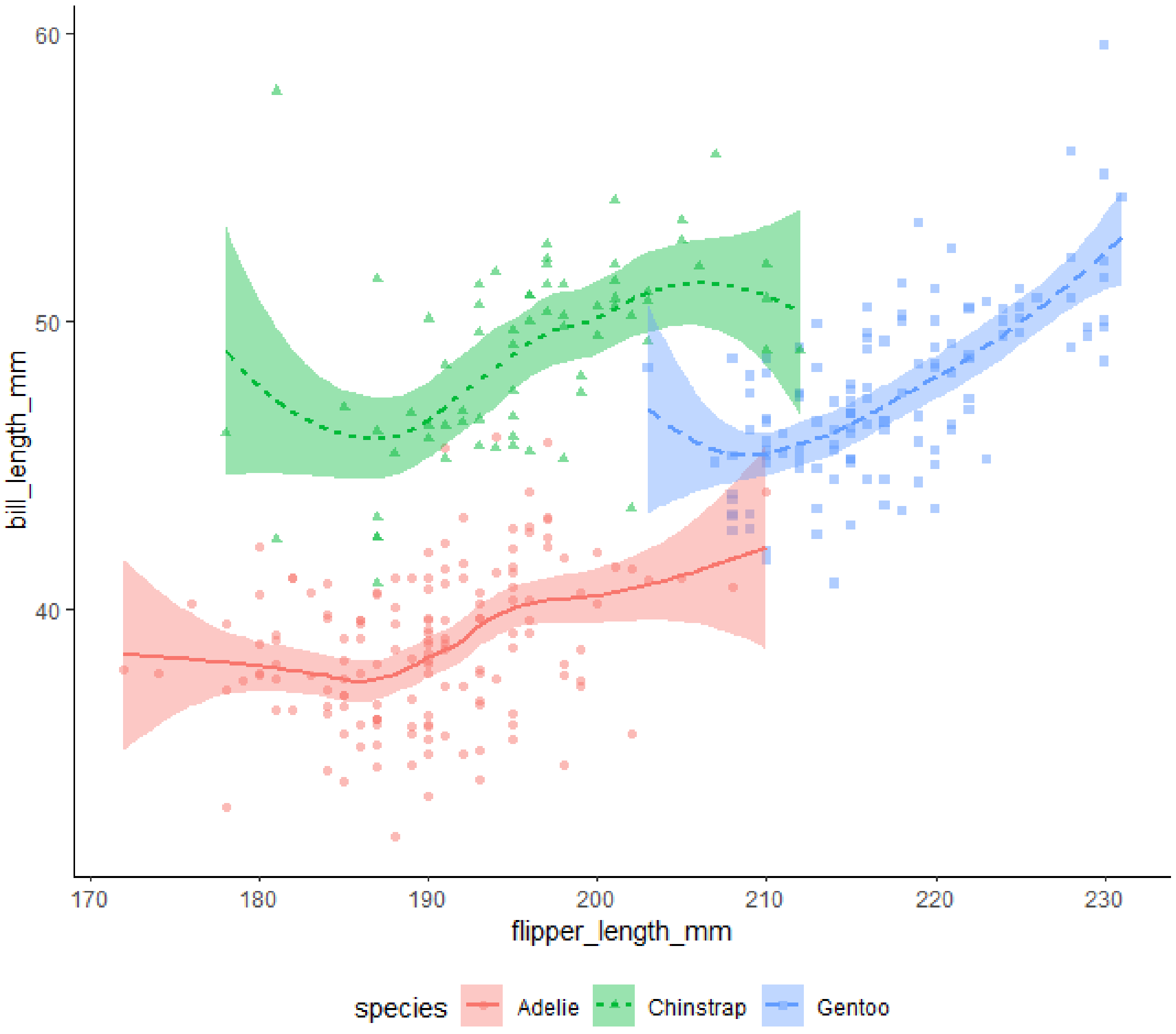

```
ggplot(penguins) +
  aes(x = bill_length_mm,
      y = flipper_length_mm,
      fill = species,
      color = species) +
  geom_point() +
  geom_smooth() +
  geom_smooth(
    method = "lm",
    color = "black",
    linetype = "dashed“
    )
```

loess smoother



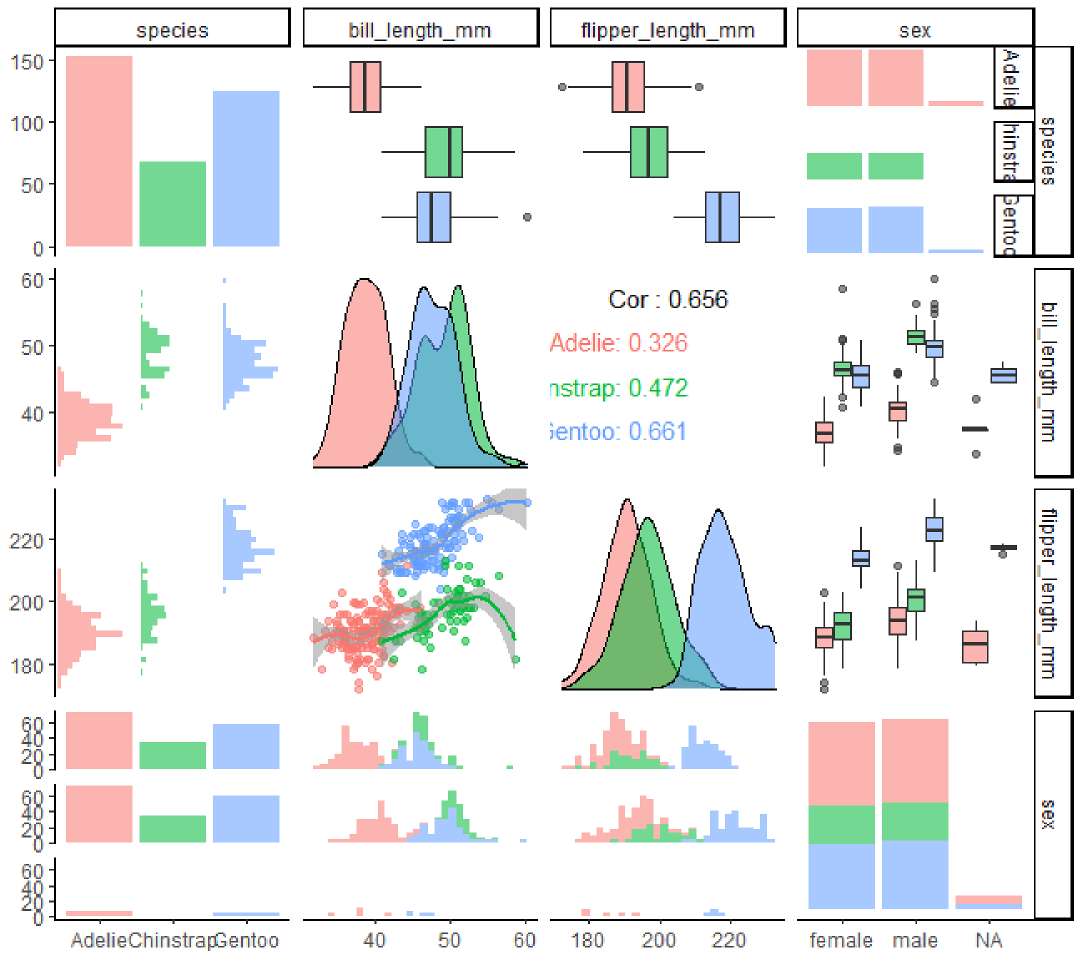

```
ggplot(penguins) +
  aes(x = bill_length_mm,
      y = flipper_length_mm,
      fill = species,
      color = species) +
  geom_point() +
  geom_smooth(method = "lm",
              linetype = "dashed")
+
  geom_smooth(color = "black",
              fill = "black",
              alpha = .2) +
  geom_smooth(method = "lm",
              color = "orange",
              fill = "orange",
              alpha = .2)
```

A full EDA plot




```
ggplot(penguins) +
  aes(x = flipper_length_mm,
      y = bill_length_mm,
      color = species,
      fill = species,
      shape = species,
      linetype = species) +
  geom_point(alpha = .50) +
  geom_smooth() +
  theme(
    legend.position = "bottom“
  )
```

Scatterplot matrix




```
GGally::ggpairs(
  penguins_focal,
  aes(color = species,
      alpha = .5),
  lower = list(
    continuous = "smooth_loess",
    combo = "facethist",
    discrete = "facetbar",
    na = "na")
  )
```

Key ggplot aesthetics

aes	What it does
`x, y`	Which axis to plot the data on; Leave one blank if the geom will compute automatically (e.g., geom_bar(); geom_histogram())
`color`	Color of the lines/borders
`fill`	Color of the fill/area
`shape`	Point shape to use
`linetype`	Type of line to use (solid, dashed, etc.)
`size`	Size of point, width of line
`alpha`	Transparency level (0 [invisible] to 1 [solid])

```
ggplot(penguins) +
  aes(x = flipper_length_mm,
      y = bill_length_mm,
      
      color = species,
      fill = species,
      shape = species,
      linetype = species
  ) +
  geom_point(
    aes(size = body_mass_g),
    alpha = .50
  ) +
  geom_smooth()
```


UNIVERSITY OF SOUTH FLORIDA.

General data viz tips

- Never contrast `"black"` and `"red"` !
- Choose aesthetics that are easy to process
  - Avoid angles, area
- Avoid chart junk
  - Non-communicative color
  - Unnecessary lines (e.g., gridlines)
  - Don’t use a bar when a line will do
- Use direct labeling, rather than legends
- Draw attention to key data points
- See https://www.data-to-viz.com/caveats.html