Introduction to R

Lecture 3: Data Visualization

Swapnil Singh
Lietuvos Bankas | Course Link

Basic Visualization

Why we need visualization?

- Imagine how different painters have drawn trees:
  - Picasso
  - Monet
  - Mondrian
  - Kasiulis
- Paintings convey
  - information
  - perspective of a painter
- Same is with data visualization
  - convey information in a concise manner
  - provide a perspective to look at the data

What we will do in this class?

- Basics, nitty-gritty details of visualization
  - understand which type of plot to use for different questions
  - understand different concepts within each plot
- Learn how `ggplot` helps us to visualize and convey information
  - create **presentable** graphics

Distributions

- When we want to understand only one variable, we look at the distribution
- Variables can be of two types:
  - discrete or categorical (take only few values)
    - Ex: gender, number of children in the household, ...
  - numerical or continuous (take many values)
    - Ex: age, population of different cities, GDP growth rate,...

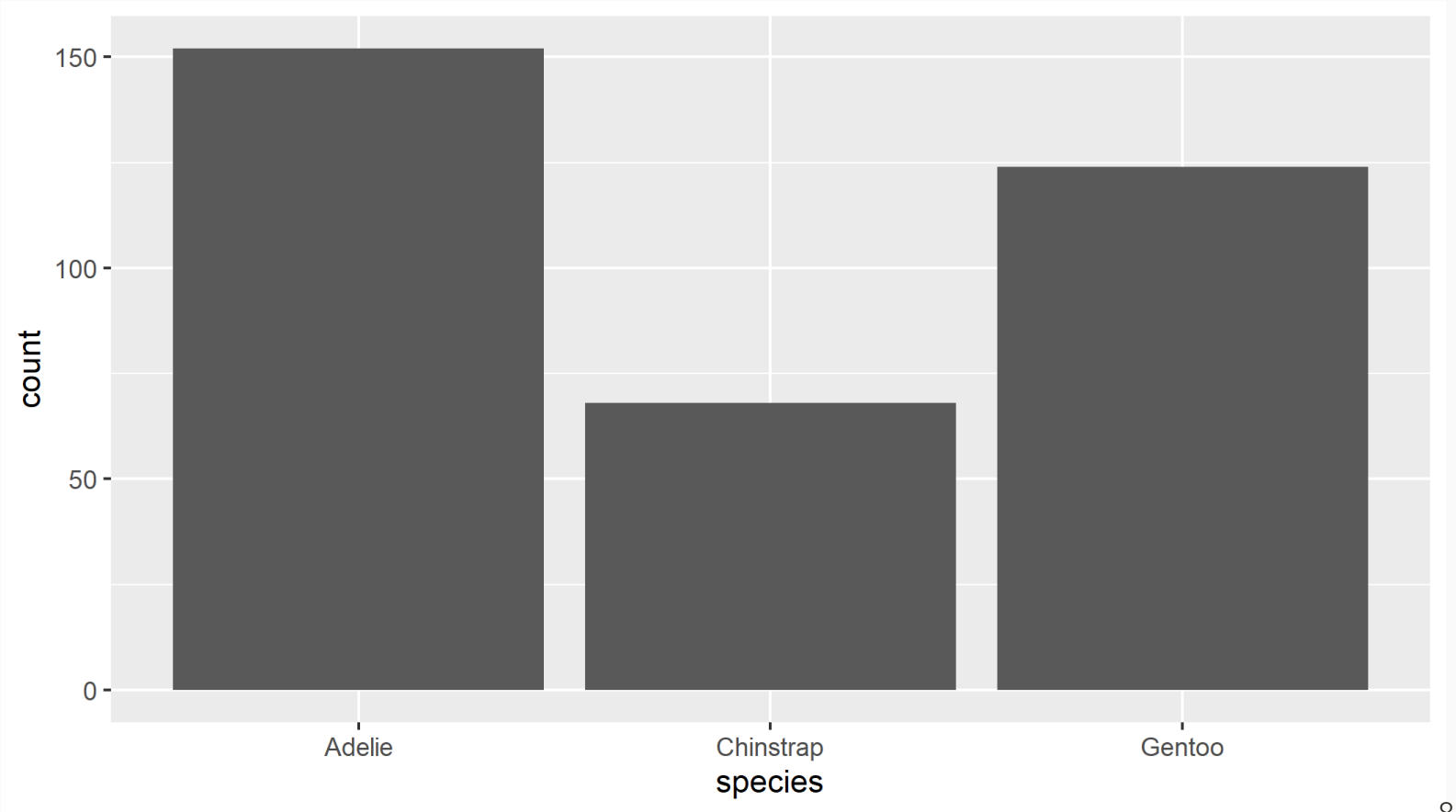
Distribution: Discrete Variable

- To visualize the distribution of a discrete variable, use `bar` chart
- For instance, in `penguins` data, we might be interested in the distribution of penguin species

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

Distribution: Discrete Variable

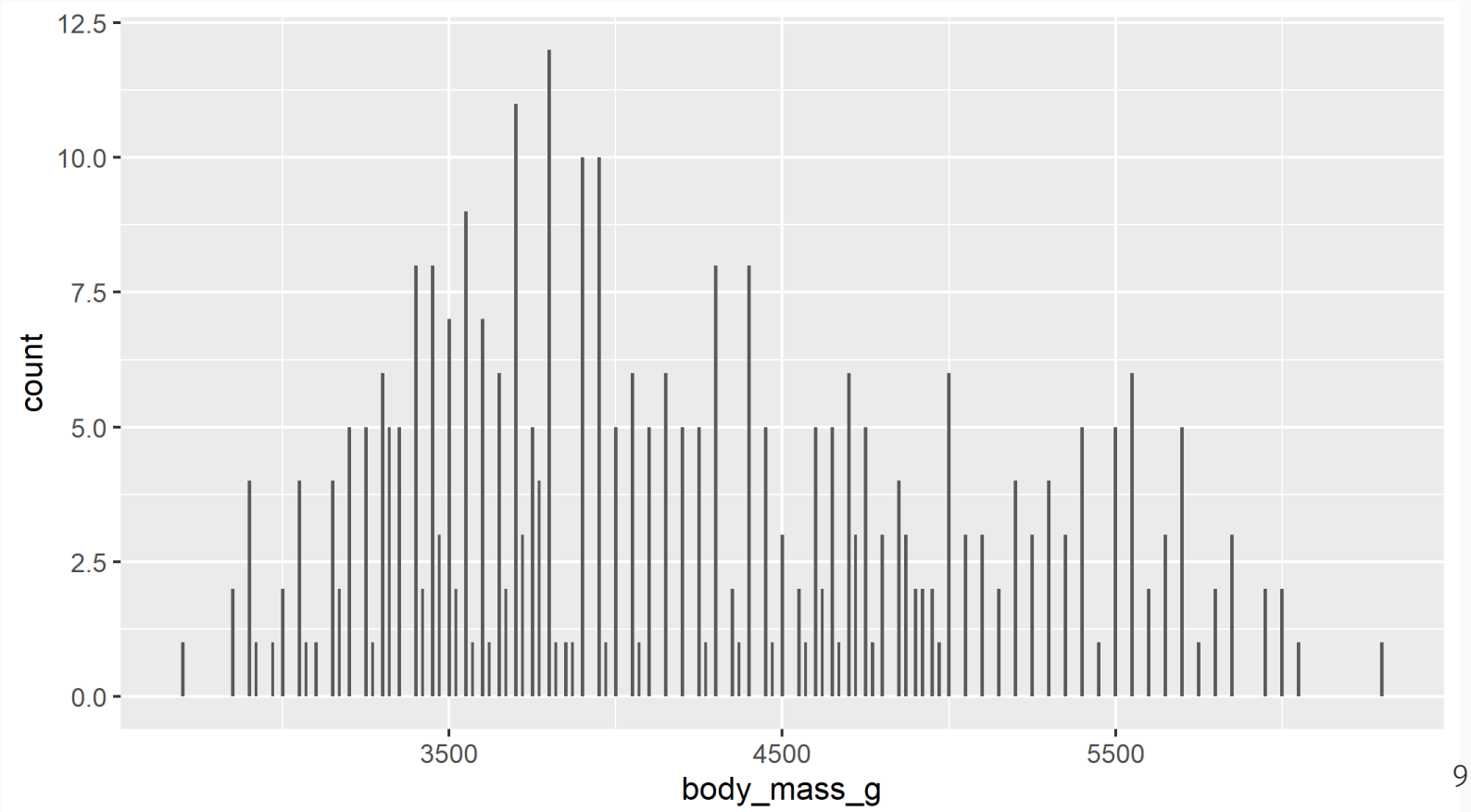
- To visualize the distribution of a discrete variable, use `bar` chart
- For instance, in `penguins` data, we might be interested in the distribution of penguin species



Distribution: Continuous Variable

- For continuous variables we can use `histogram`

```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth=10)
```



Distribution: Continuous Variable

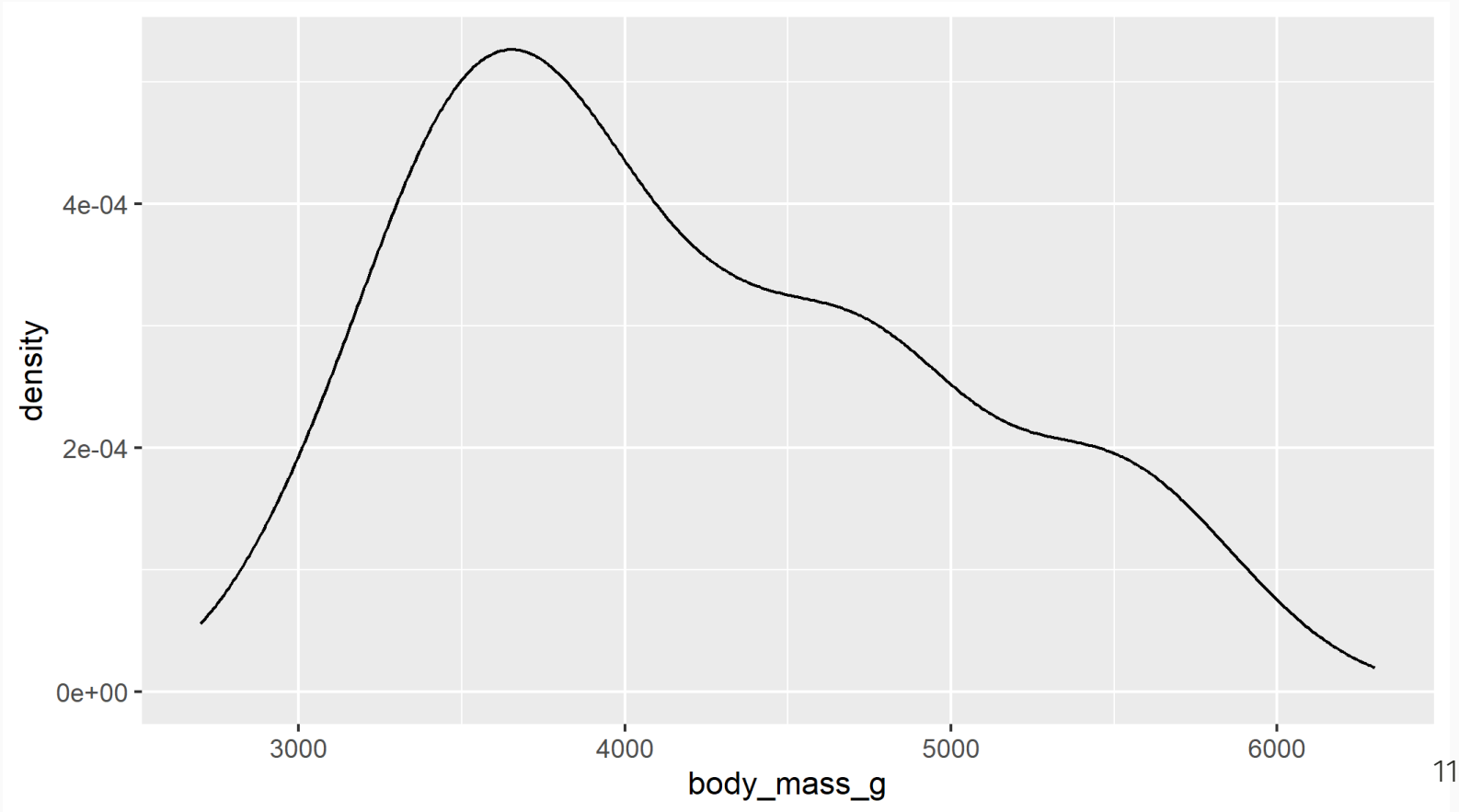
- For continuous variables we can use `histogram`

```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth=100)
```

Distribution: Continuous Variable

- You can also use `density` which is just a smoothed histogram

```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_density()
```



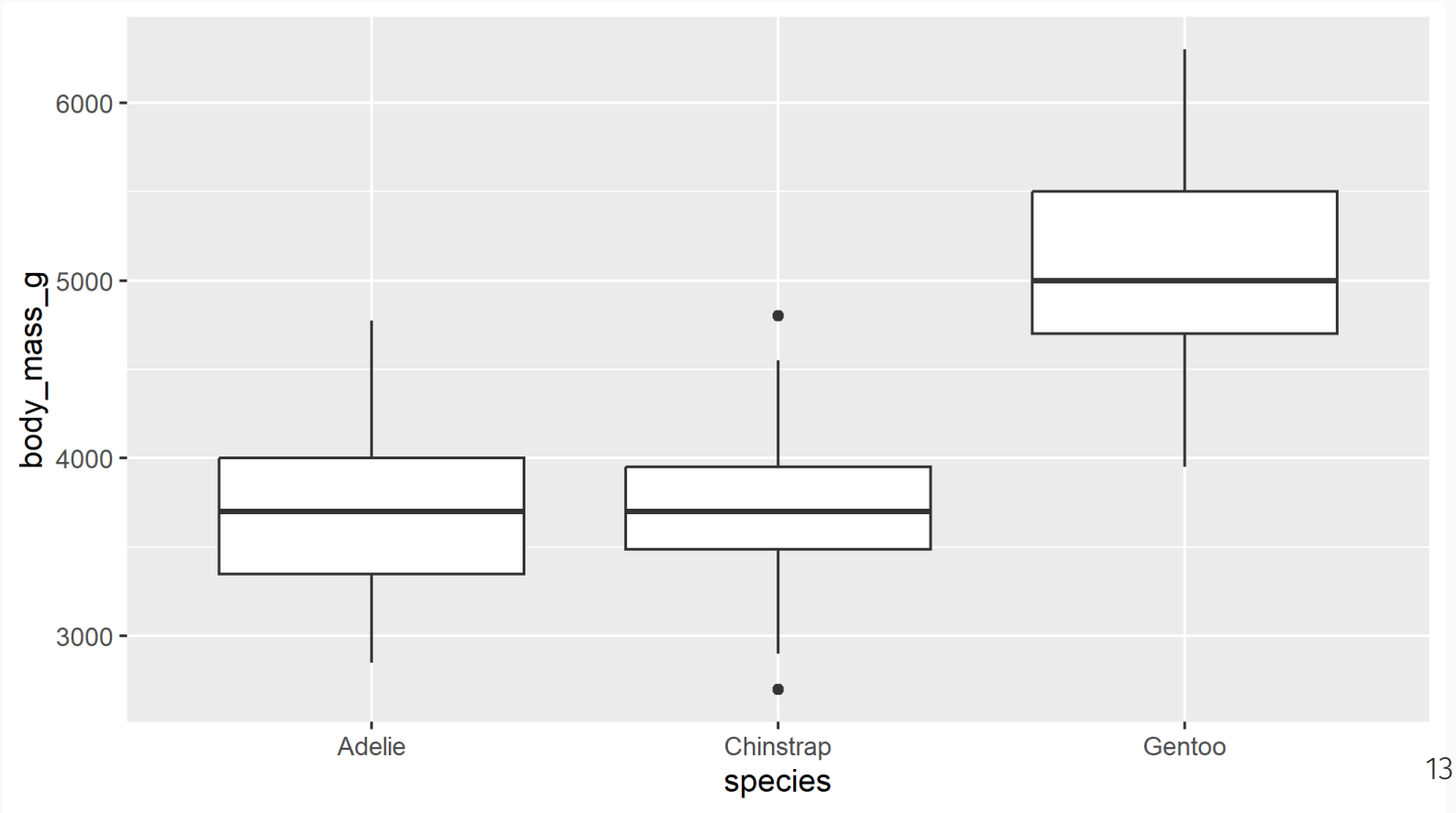
Relationship between two variables

- Two variables can be
  - continuous and discrete
  - discrete and discrete
  - continuous and continuous

Continuous and Discrete

- Distribution of body mass by species of penguins

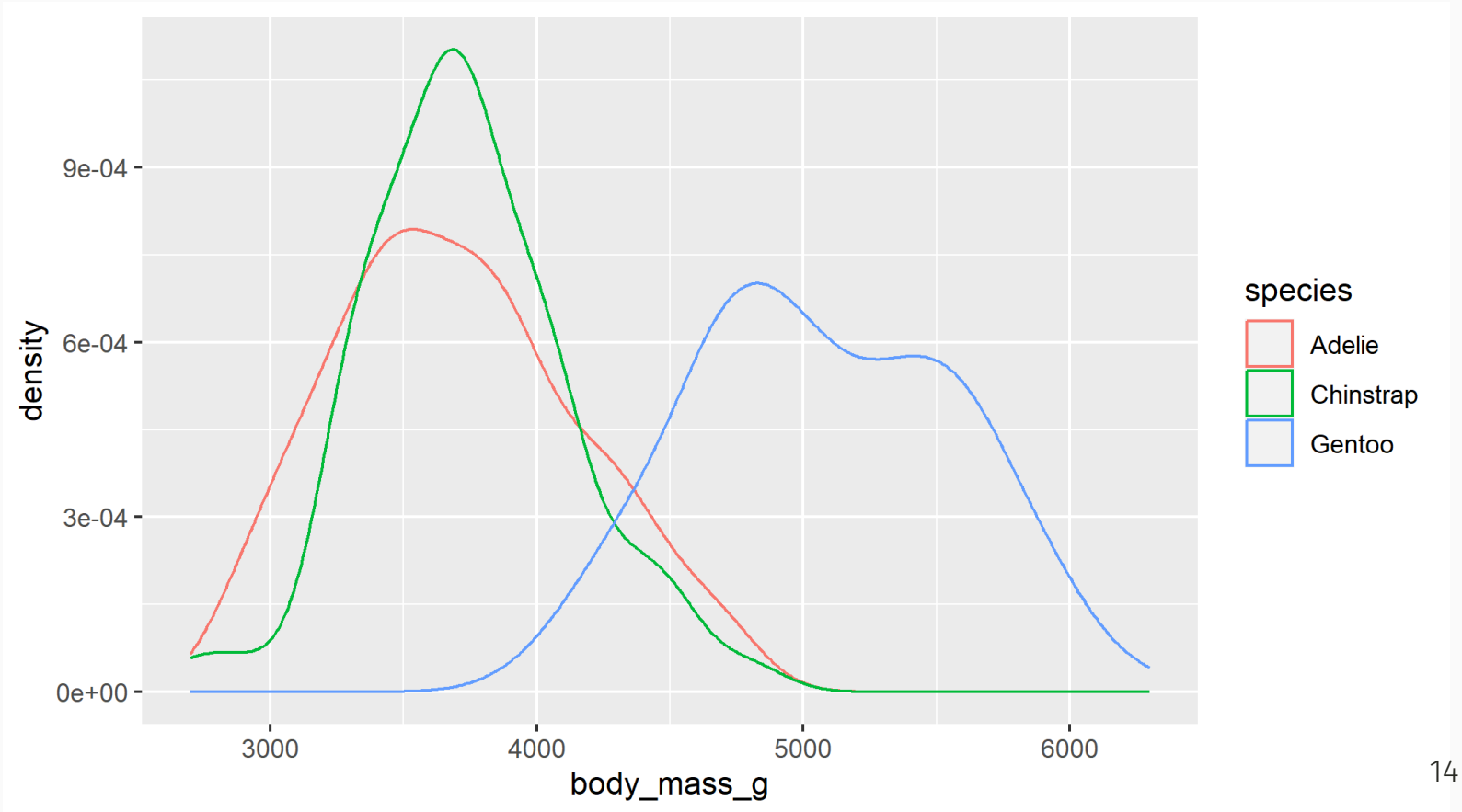
```
ggplot(data = penguins, aes(x = species, y = body_mass_g)) +
  geom_boxplot()
```



Continuous and Discrete

- Distribution of body mass by species of penguins

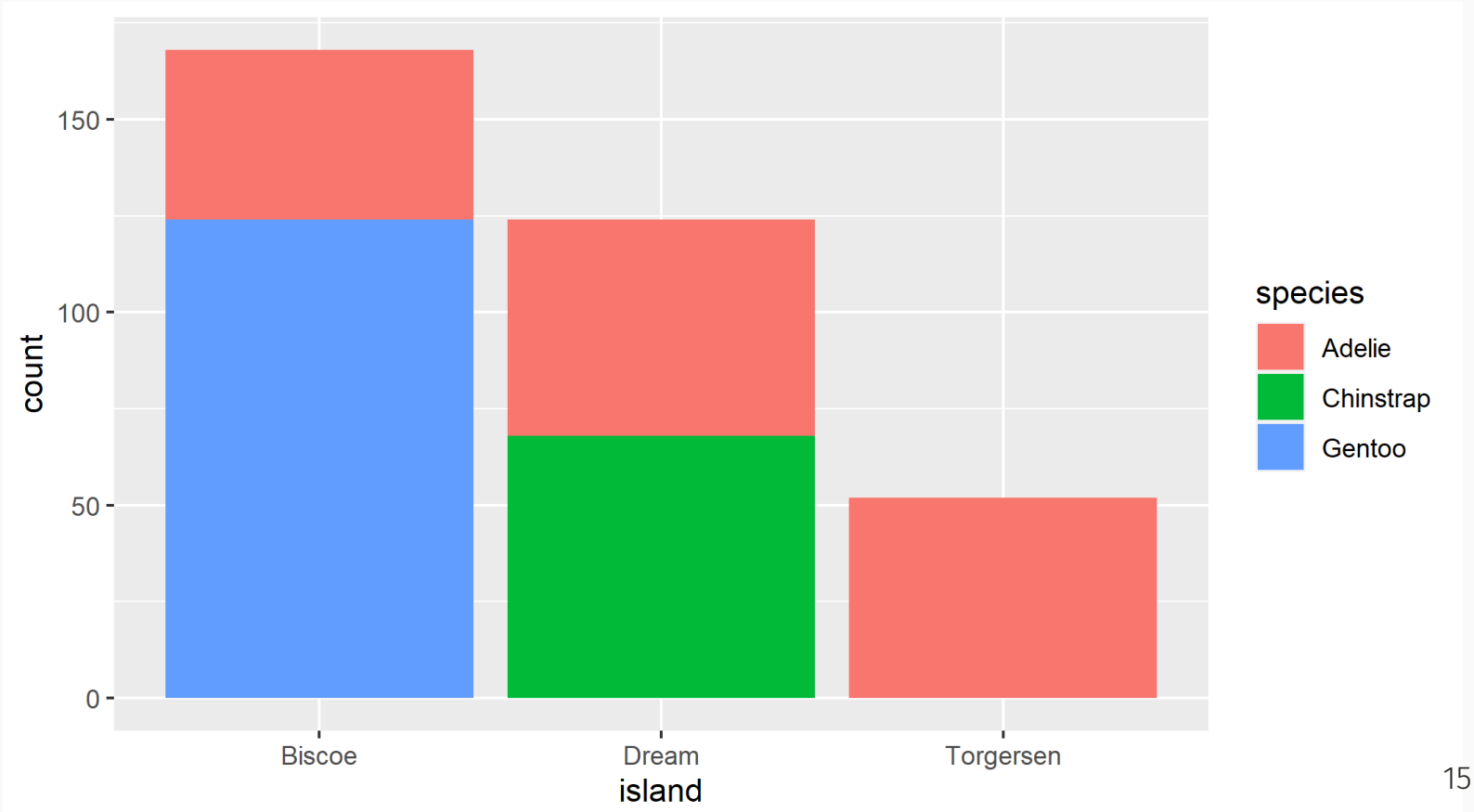
```
ggplot(data = penguins, aes( x = body_mass_g, color=species)) +
  geom_density()
```



Two Discrete Variables

- Distribution of species across island: stacked bar plot

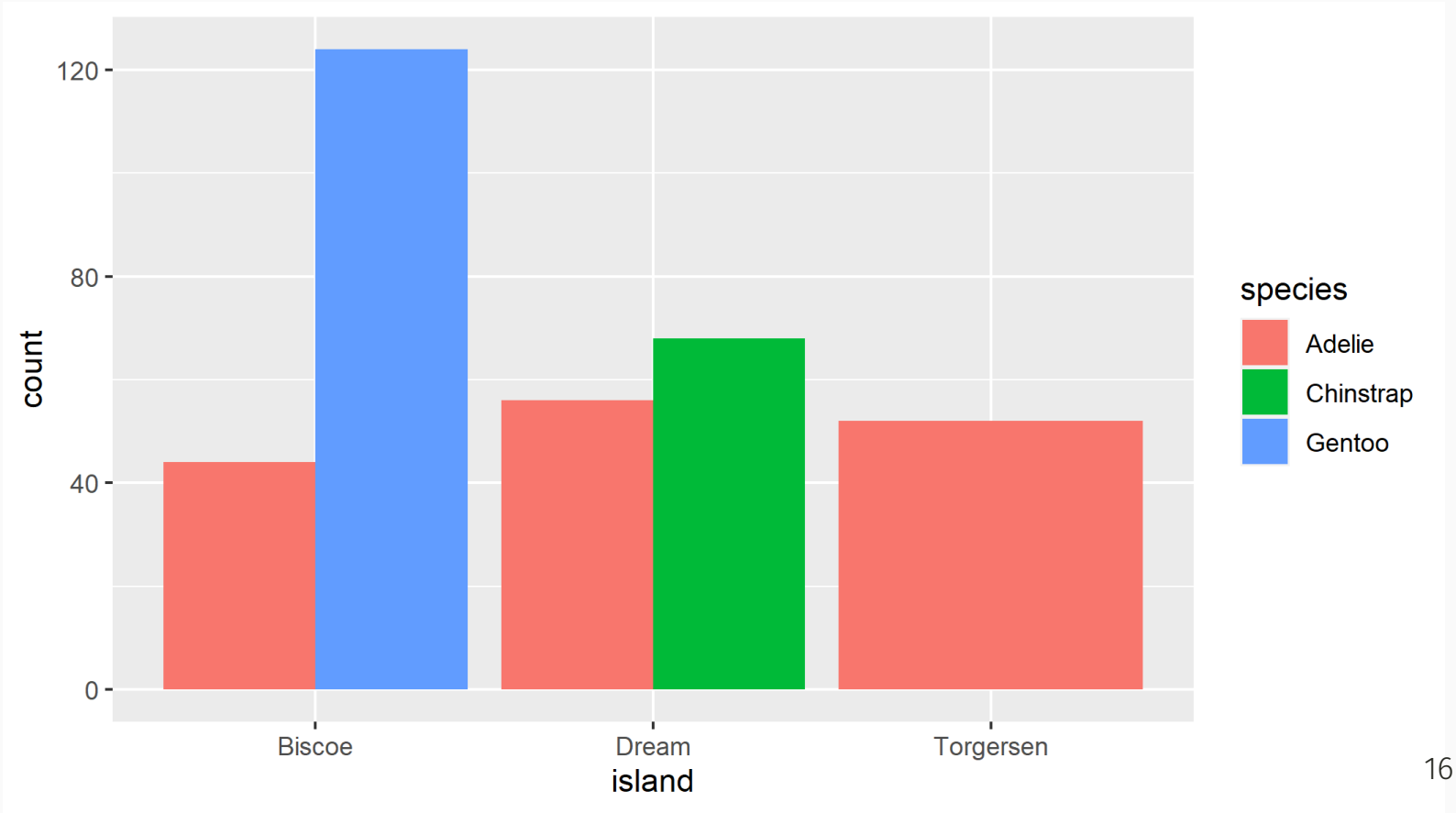
```
ggplot(data = penguins, aes(x = island, fill=species)) +
  geom_bar()
```



Two Discrete Variables

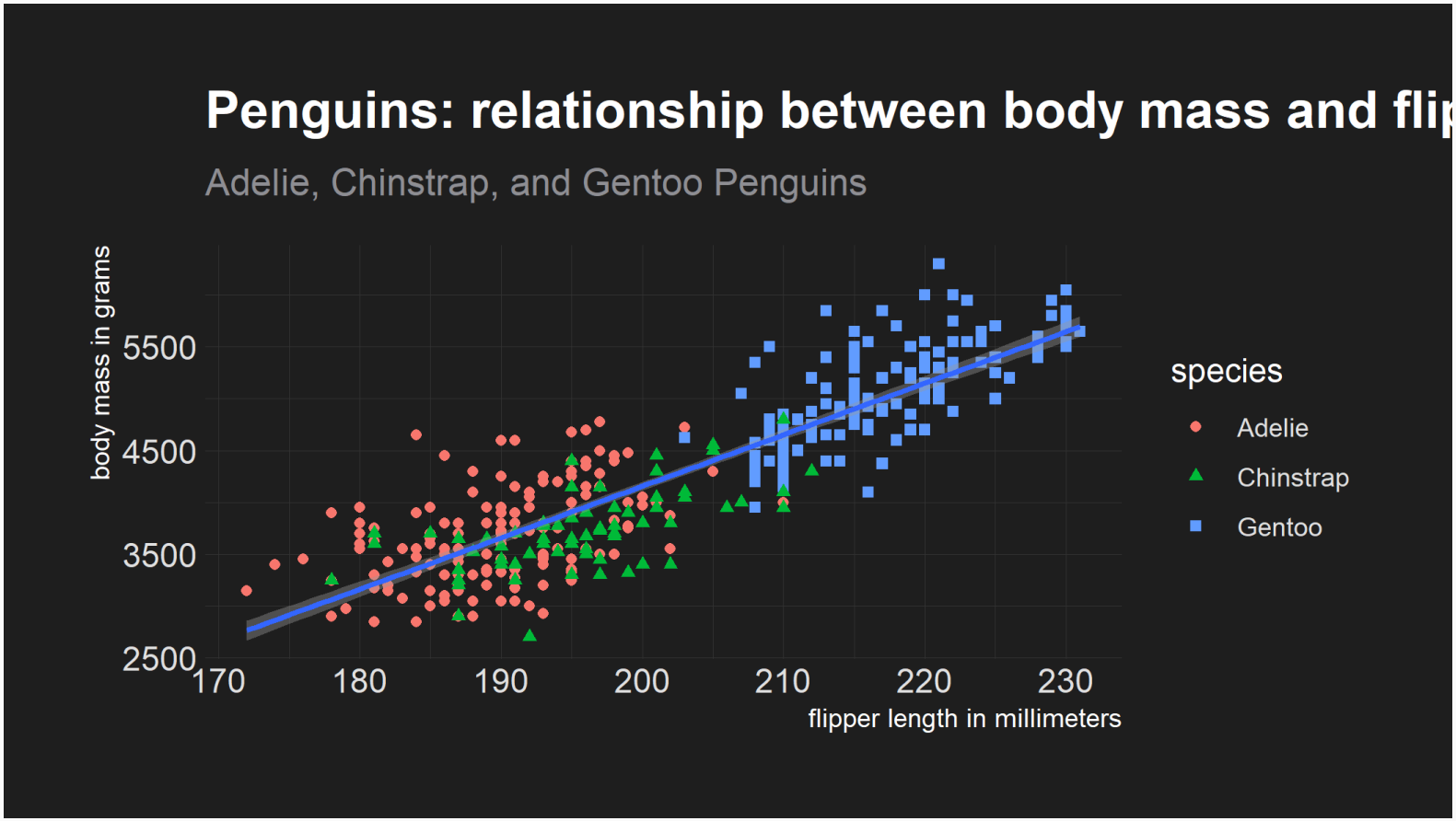
- Distribution of species across island: side by side bar plot

```
ggplot(data = penguins, aes(x = island, fill=species)) +
  geom_bar(position = 'dodge')
```



Two continuous variables

- From the last class



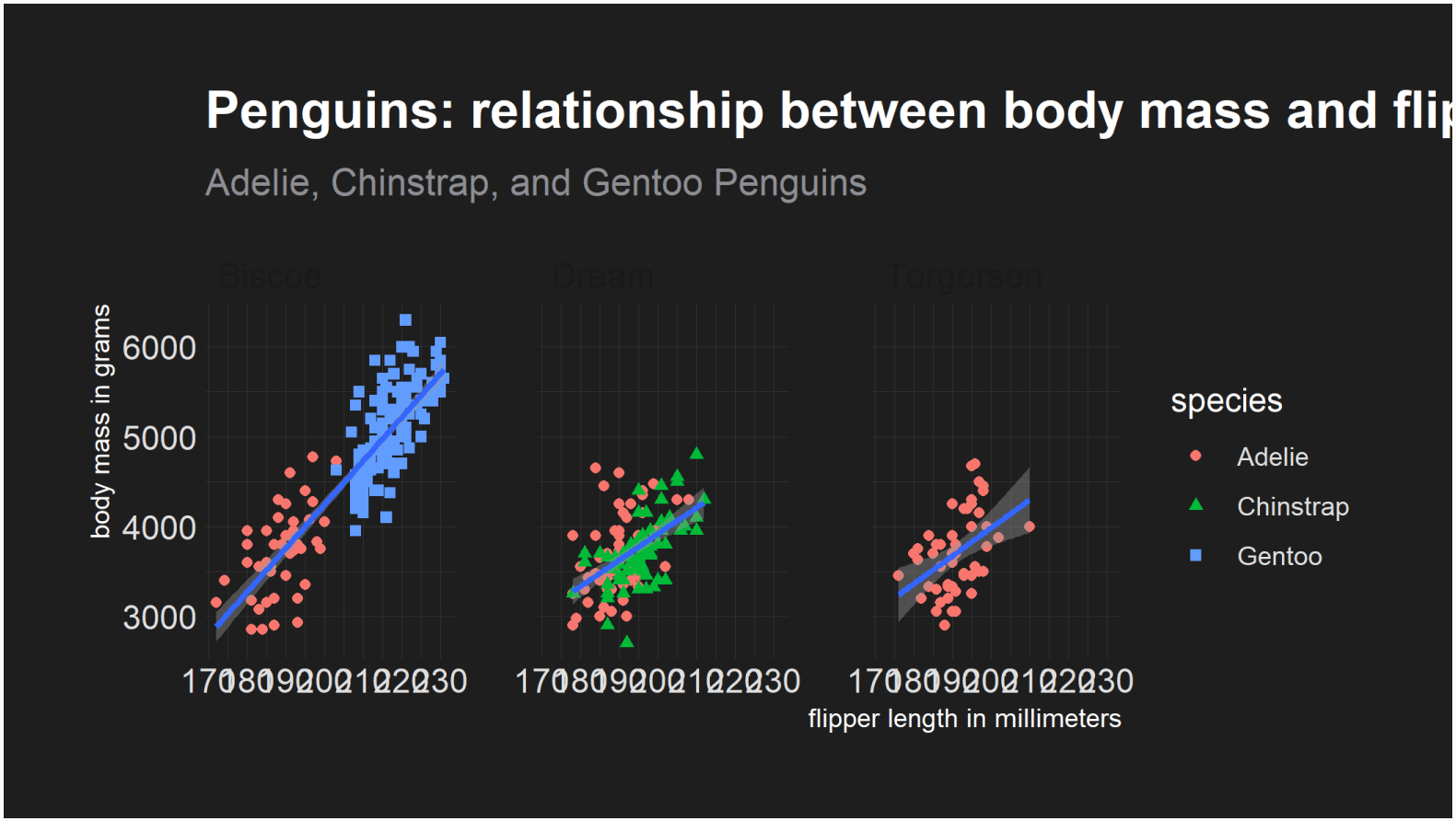
Two continuous variables

- From the last class (conditional on island)

```
p ←  ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +

  # notice how `aes` for geom_point shifted from above to here
  geom_point(mapping = aes(color=species, shape=species)) +

  #add linear fit (one line across all three groups)
  geom_smooth(method='lm')  +

  #conditional on island
  facet_wrap(~island) +

  #modern theme
  theme_modern_rc() +

  labs(
    title = "Penguins: relationship between body mass and flipper length",
    subtitle = "Adelie, Chinstrap, and Gentoo Penguins",
    x = "flipper length in millimeters",
    y = "body mass in grams",
    color = "species",
    shape = "species"
  )
```

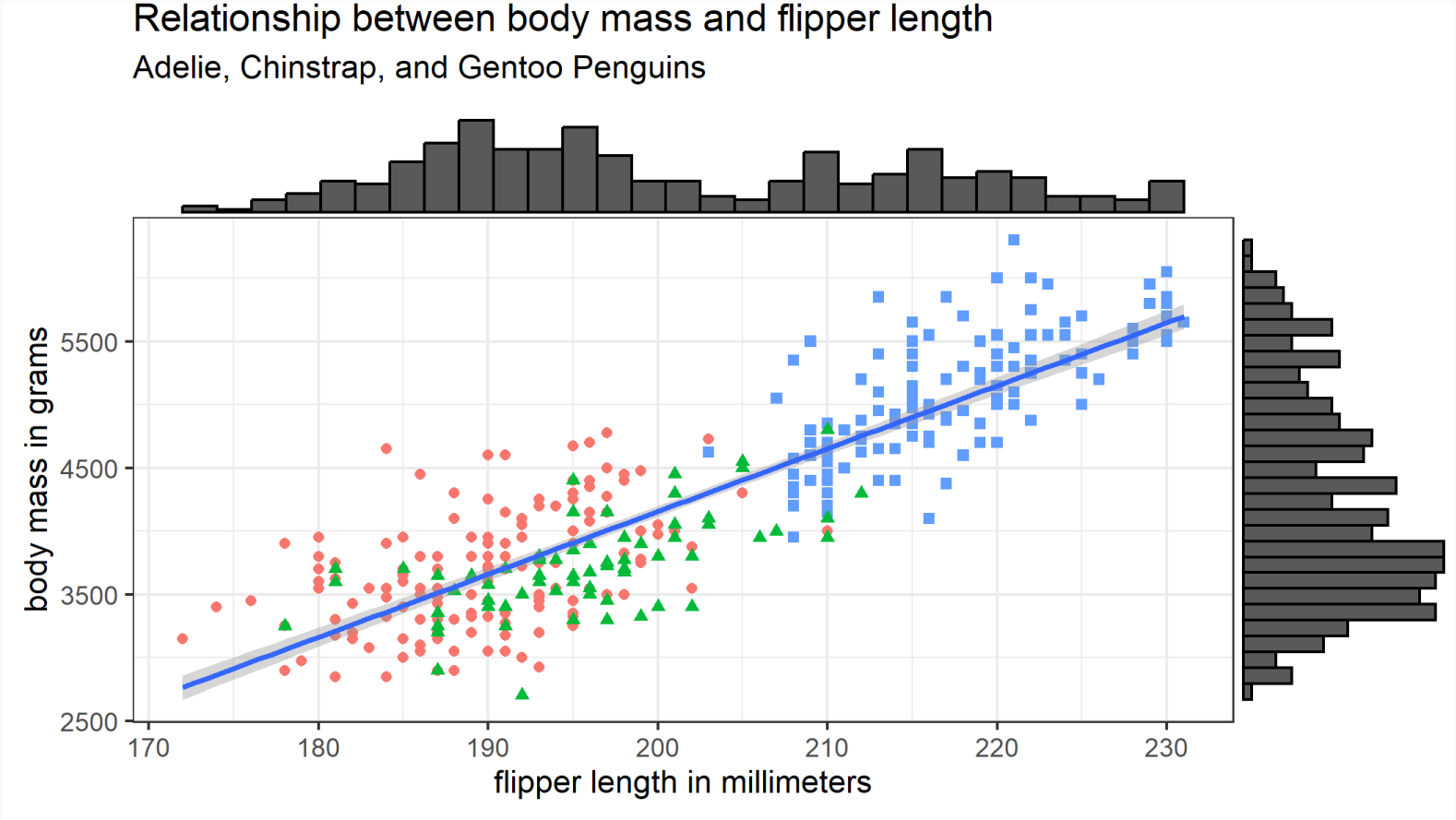
Two continuous variables

- From the last class (conditional on island)



Marginal Distributions

Marginal Distributions

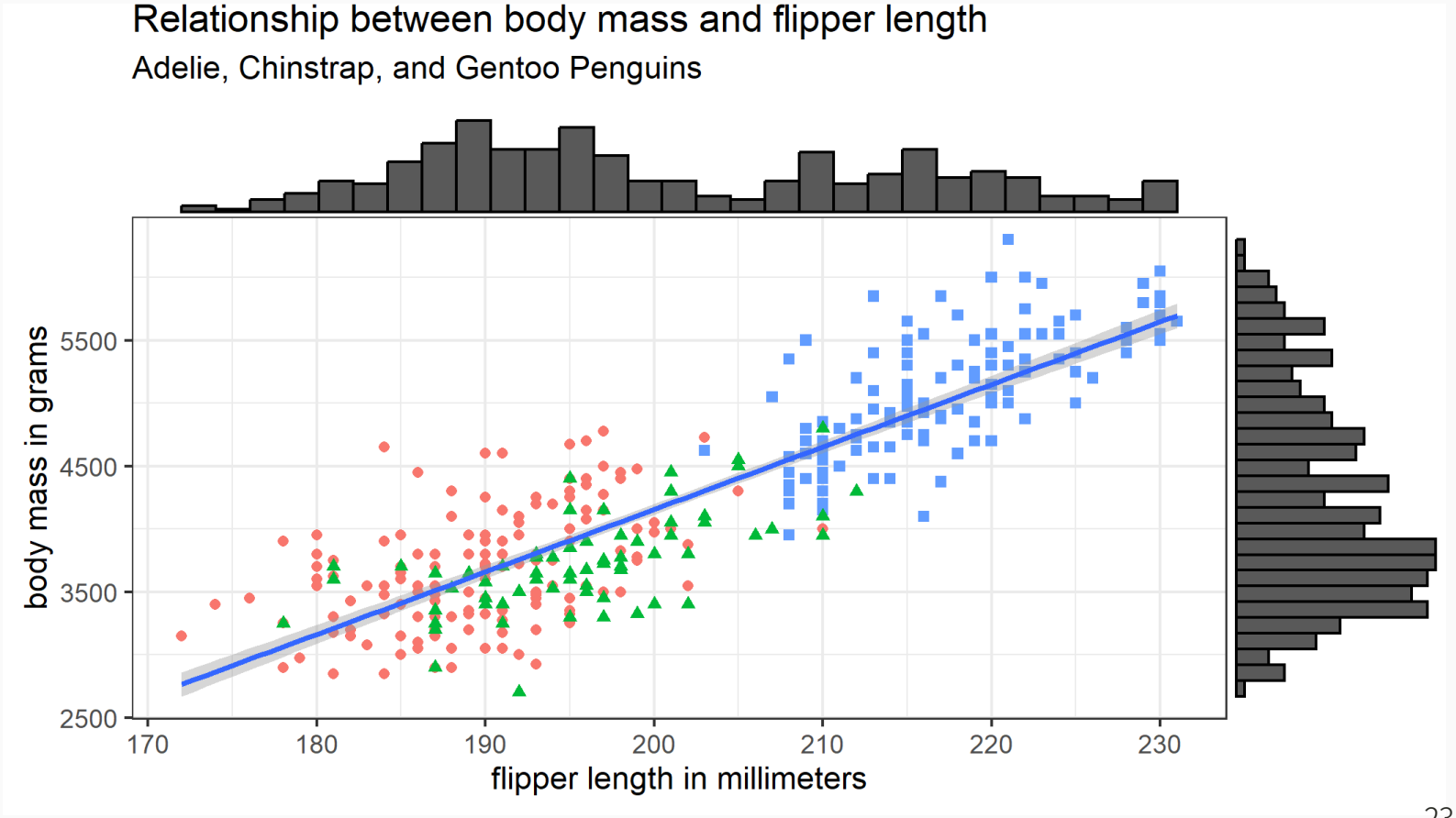


Marginal Distributions

```
p ← ggplot(penguins, aes(x=flipper_length_mm, y=body_mass_g)) +

  # notice how `aes` for geom_point shifted from above to here
  geom_point(mapping = aes(color=species, shape=species)) +


  #add linear fit (one line across all three groups)
  geom_smooth(method='lm')  +

  #modern theme
  theme_bw() +

  labs(
    title = "Relationship between body mass and flipper length",
    subtitle = "Adelie, Chinstrap, and Gentoo Penguins",
    x = "flipper length in millimeters",
    y = "body mass in grams",
    color = "species",
    shape = "species"
  ) +

  theme(legend.position = 'none')
```

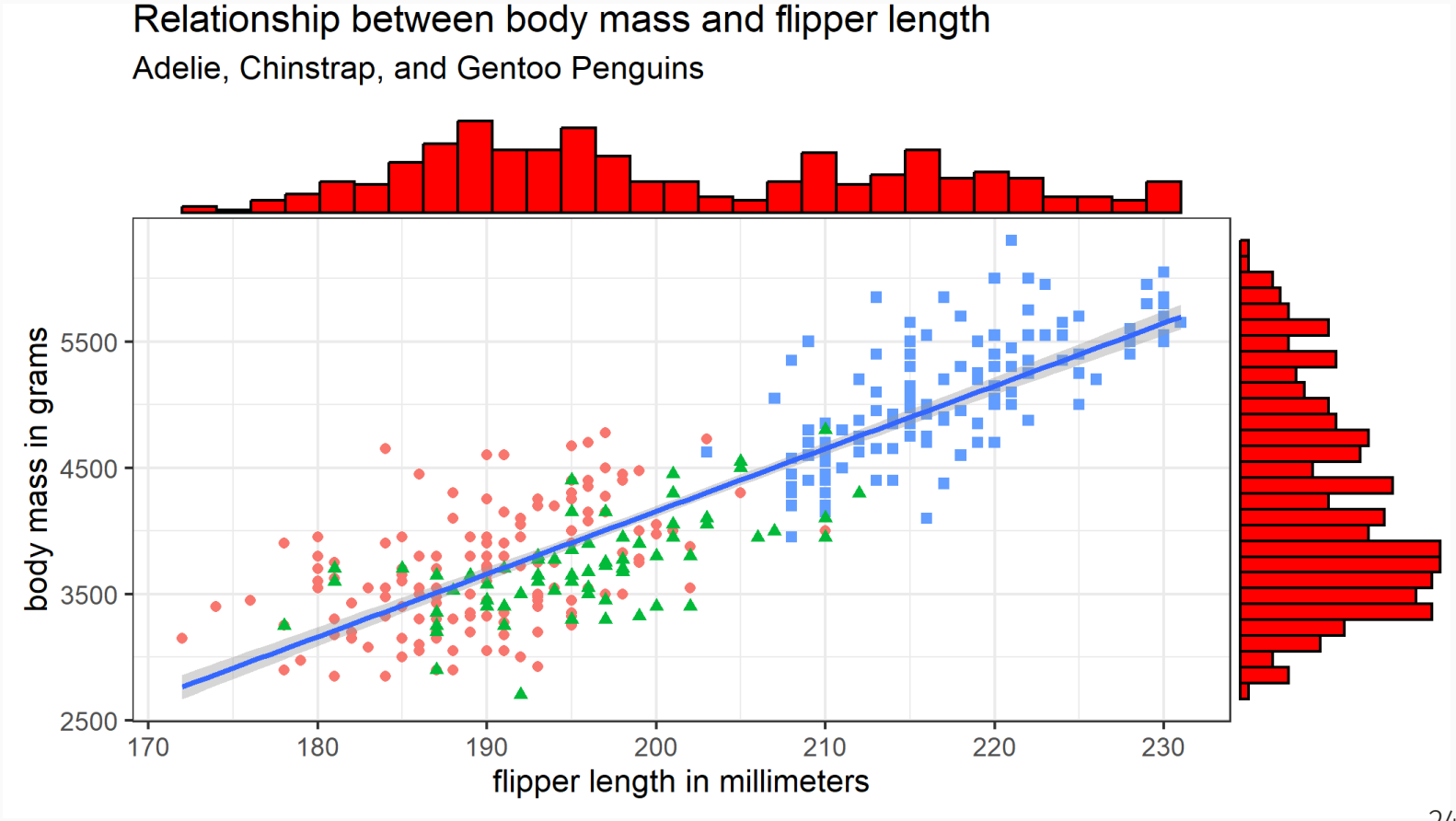
Marginal Distributions

```
# with marginal histogram
p1 ← ggMarginal(p, type="histogram")
p1
```



Marginal Distributions

```
# with marginal histogram
p2 ← ggMarginal(p, type="histogram", fill='red')
p2
```



Importing Data in R

Reading csv files

- Comma separated value (CSV) files are most common
- Command is: `read_csv`

```
cleanFuelData ← read_csv(file = 'raw_data/clean-fuel-data.csv')
glimpse(cleanFuelData)
```

```
## Rows: 271
## Columns: 67
## $ `Country Name`  <chr> "Afghanistan", "Albania", "Algeria", "American Samoa",…
## $ `Country Code`  <chr> "AFG", "ALB", "DZA", "ASM", "AND", "AGO", "ATG", "ARG"…
## $ `Series Name`   <chr> "Access to clean fuels and technologies for cooking (%…
## $ `Series Code`   <chr> "EG.CFT.ACCS.ZS", "EG.CFT.ACCS.ZS", "EG.CFT.ACCS.ZS", …
## $ `1960 [YR1960]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1961 [YR1961]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1962 [YR1962]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1963 [YR1963]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1964 [YR1964]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1965 [YR1965]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1966 [YR1966]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1967 [YR1967]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1968 [YR1968]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1969 [YR1969]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
## $ `1970 [YR1970]` <chr> "..", "..", "..", "..", "..", "..", "..", "..", "..", …
```

Reading csv files

- How to make sure that `..` is recognized as missing

```
cleanFuelData ← read_csv(file = 'raw_data/clean-fuel-data.csv',
                         na = c('..'))
glimpse(cleanFuelData)
```

```
## Rows: 271
## Columns: 67
## $ `Country Name`  <chr> "Afghanistan", "Albania", "Algeria", "American Samoa",…
## $ `Country Code`  <chr> "AFG", "ALB", "DZA", "ASM", "AND", "AGO", "ATG", "ARG"…
## $ `Series Name`   <chr> "Access to clean fuels and technologies for cooking (%…
## $ `Series Code`   <chr> "EG.CFT.ACCS.ZS", "EG.CFT.ACCS.ZS", "EG.CFT.ACCS.ZS", …
## $ `1960 [YR1960]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1961 [YR1961]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1962 [YR1962]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1963 [YR1963]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1964 [YR1964]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1965 [YR1965]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1966 [YR1966]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1967 [YR1967]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1968 [YR1968]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1969 [YR1969]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1970 [YR1970]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ `1971 [YR1971]` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
```

Reading csv files

- Now look at variable names like country name or country code
  - surrounded by backticks. Why?
    - space between names
- we need to clean up in a cumbersome way

```
cleanFuelData ← read_csv(file = 'raw_data/clean-fuel-data.csv',
                         na = c('..')) ▷
  rename(country_name = `Country Name`,
         country_code = `Country Code`)
```

Reading csv files

- Alternative way to clean variable names: use `janitor` package

```
cleanFuelData ← read_csv(file = 'raw_data/clean-fuel-data.csv',
                         na = c('..')) ▷
  janitor::clean_names()
glimpse(cleanFuelData)
```

```
## Rows: 271
## Columns: 67
## $ country_name <chr> "Afghanistan", "Albania", "Algeria", "American Samoa", "A…
## $ country_code <chr> "AFG", "ALB", "DZA", "ASM", "AND", "AGO", "ATG", "ARG", "…
## $ series_name  <chr> "Access to clean fuels and technologies for cooking (% of…
## $ series_code  <chr> "EG.CFT.ACCS.ZS", "EG.CFT.ACCS.ZS", "EG.CFT.ACCS.ZS", "EG…
## $ x1960_yr1960 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1961_yr1961 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1962_yr1962 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1963_yr1963 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1964_yr1964 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1965_yr1965 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1966_yr1966 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1967_yr1967 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1968_yr1968 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1969_yr1969 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ x1970_yr1970 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
```

Importing from Databases

- CSV or Excel are not always available
- Sometimes, you have to access `databases`
- We need to learn two things then
  - `DBI` package
  - `dbplyr` package
- `DBI` package connects you to a database
- `dbplyr` converts `dplyr` code to `SQL`

What is a database?

- Think of database as a bunch of `dataframes`
- These `dataframes` are also called `tables`
  - there are some difference between `dataframe` and `table` but not important for this class
- Databases are run by database management systems (DBMS)
- Three types of DBMS
  1. Client server
  2. Cloud
  3. In-process

How it works? Big Picture

- There are two steps involved:
  1. use `DBI` to connect to the database and perform simple functions
  2. Depending on the DBMS, you will need specific package
     - `RPostgres` for `PostgreSQL`
     - `RMariaDB` for `MySQL`
- For this class we will use in-house `duckdb`
  - difference between `duckdb` and other DBMS is only how you connect to it
  - everything else is essentially the same

Working with duckdb

```
#create empty database
con ← dbConnect(duckdb(), dbdir='chapter3_db')

# add some data to it
dbWriteTable(con, "penguins", palmerpenguins::penguins, overwrite=TRUE)
dbWriteTable(con, "penguins_raw", palmerpenguins::penguins_raw, overwrite=TRUE)
dbWriteTable(con, "diamonds", ggplot2::diamonds, overwrite=TRUE)

#now check which tables are in the database
dbListTables(con)
```

```
## [1] "diamonds"     "penguins"     "penguins_raw"
```

```
# pull one of the tables
con ▷ dbReadTable('penguins')
```

```
##       species    island bill_length_mm bill_depth_mm flipper_length_mm
## 1      Adelie Torgersen           39.1          18.7               181
## 2      Adelie Torgersen           39.5          17.4               186
## 3      Adelie Torgersen           40.3          18.0               195
## 4      Adelie Torgersen             NA            NA                NA
## 5      Adelie Torgersen           36.7          19.3               193
## 6      Adelie Torgersen           39.3          20.6               190
## 7      Adelie Torgersen           38.9          17.8               181
```

Introducing dbplyr

```
con ← dbConnect(duckdb(), dbdir='chapter3_db')

# add some data to it
dbWriteTable(con, "penguins", palmerpenguins::penguins, overwrite=TRUE)
dbWriteTable(con, "penguins_raw", palmerpenguins::penguins_raw, overwrite=TRUE)
dbWriteTable(con, "diamonds", ggplot2::diamonds, overwrite=TRUE)

penguins_db ← tbl(con, 'penguins')
penguins_db
```

```
## # Source:   table<penguins> [?? x 8]
## # Database: DuckDB 0.8.1 [ssingh@Windows 10 x64:R 4.3.1/chapter3_db]
##    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
##  1 Adelie  Torgersen           39.1          18.7               181        3750
##  2 Adelie  Torgersen           39.5          17.4               186        3800
##  3 Adelie  Torgersen           40.3          18                 195        3250
##  4 Adelie  Torgersen           NA            NA                  NA          NA
##  5 Adelie  Torgersen           36.7          19.3               193        3450
##  6 Adelie  Torgersen           39.3          20.6               190        3650
##  7 Adelie  Torgersen           38.9          17.8               181        3625
##  8 Adelie  Torgersen           39.2          19.6               195        4675
##  9 Adelie  Torgersen           34.1          18.1               193        3475
## 10 Adelie  Torgersen           42            20.2               190        4250
```