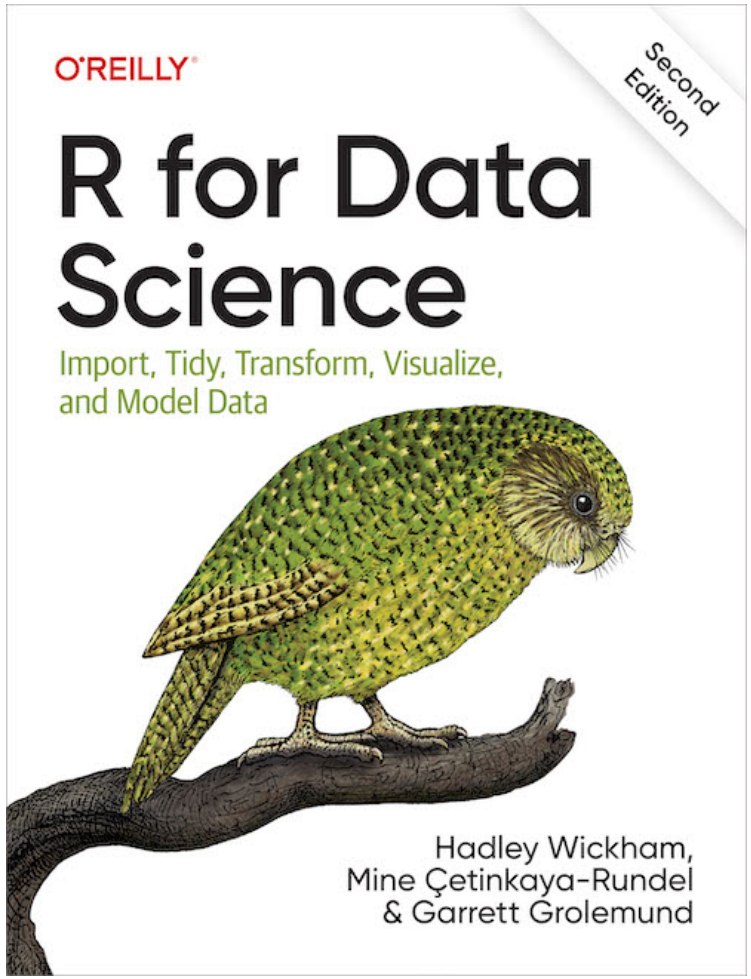
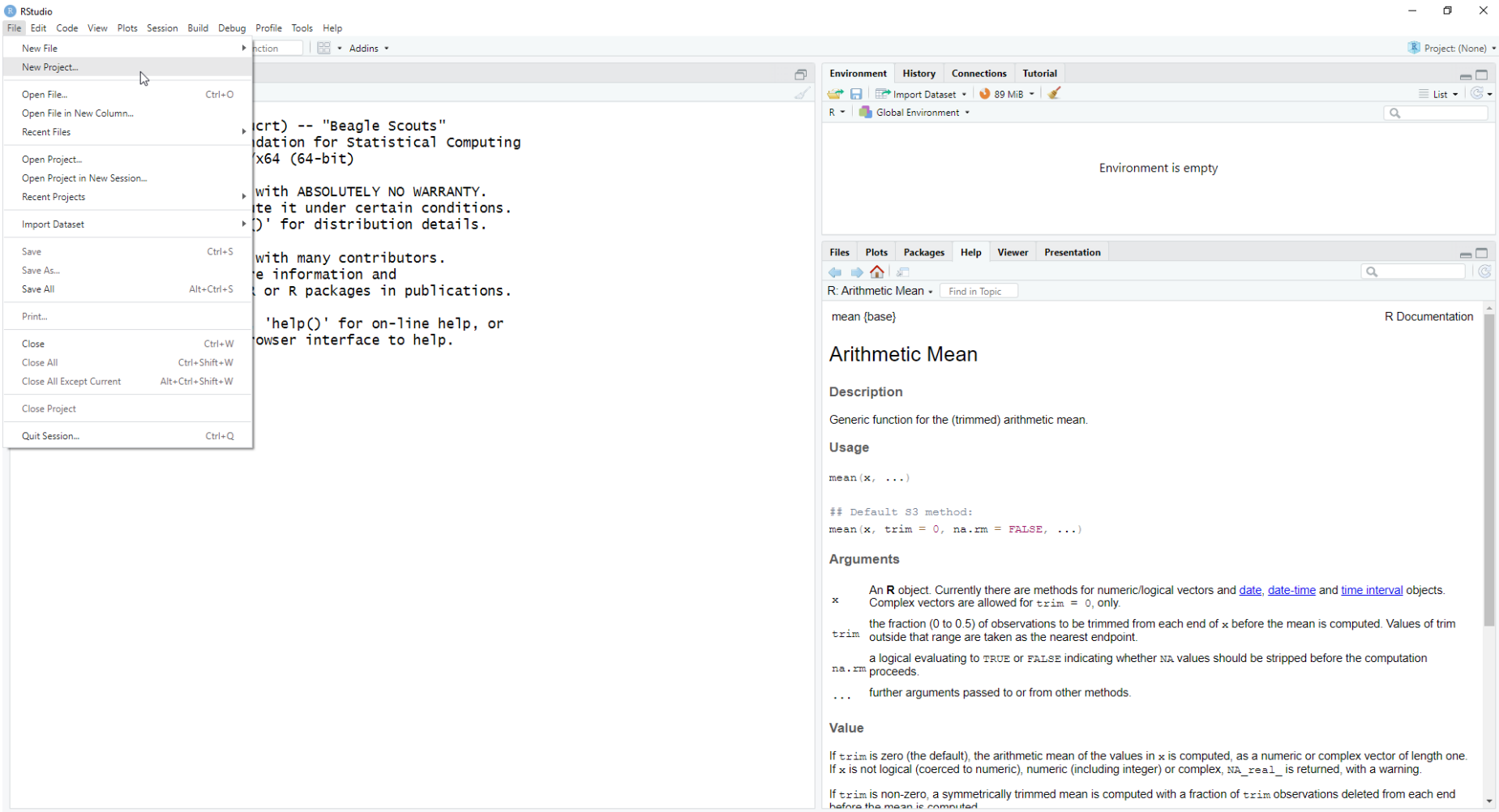
Data Visualization

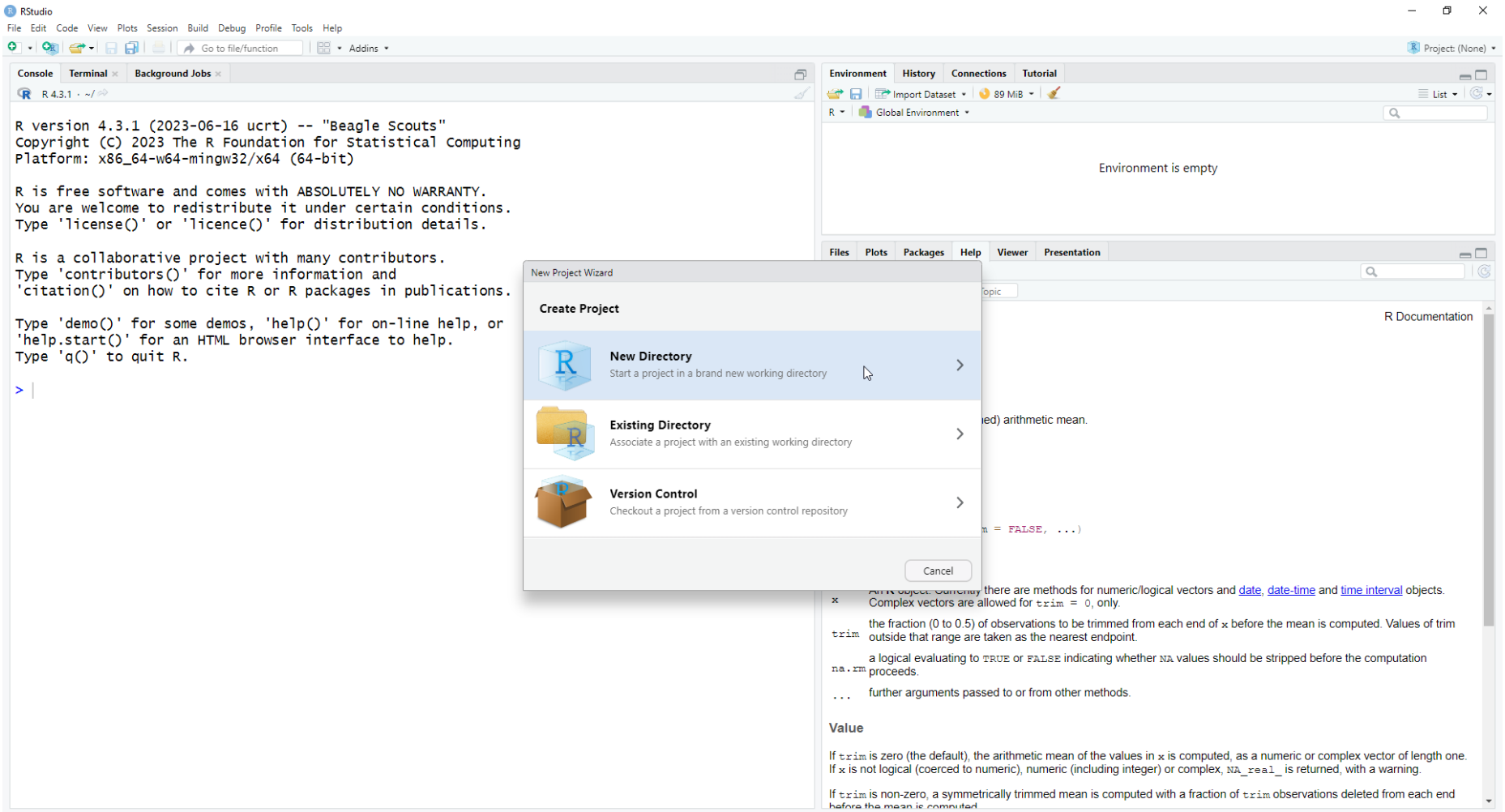
R Software Workshop

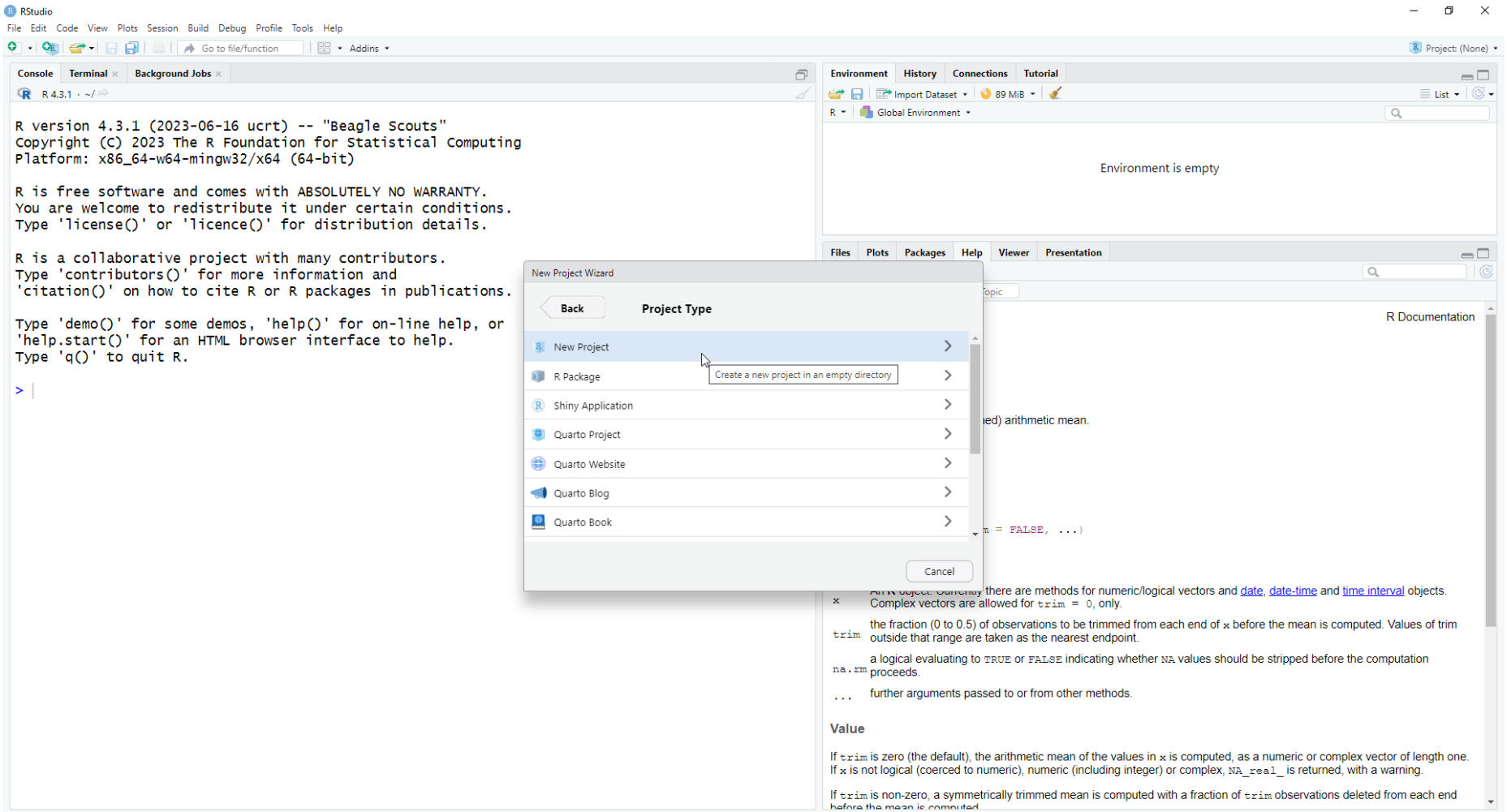
Department of Mathematics and Statistics
MSU - Iligan Institute of Technology

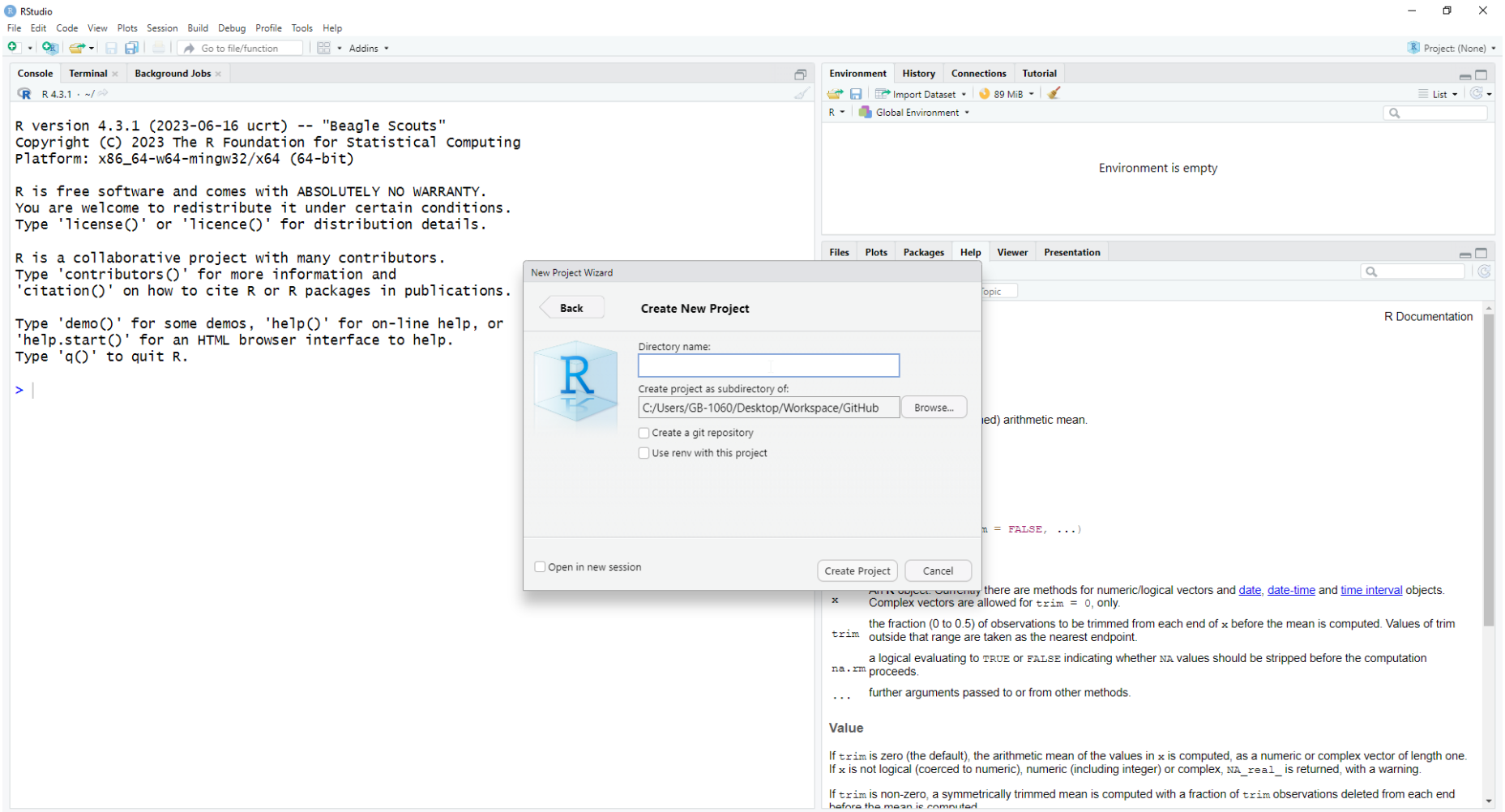

O'REILLY
Second Edition
R for Data Science
Import, Tidy, Transform, Visualize, and Model Data
Hadley Wickham,
Mine Çetinkaya-Rundel
& Garrett Grolemund

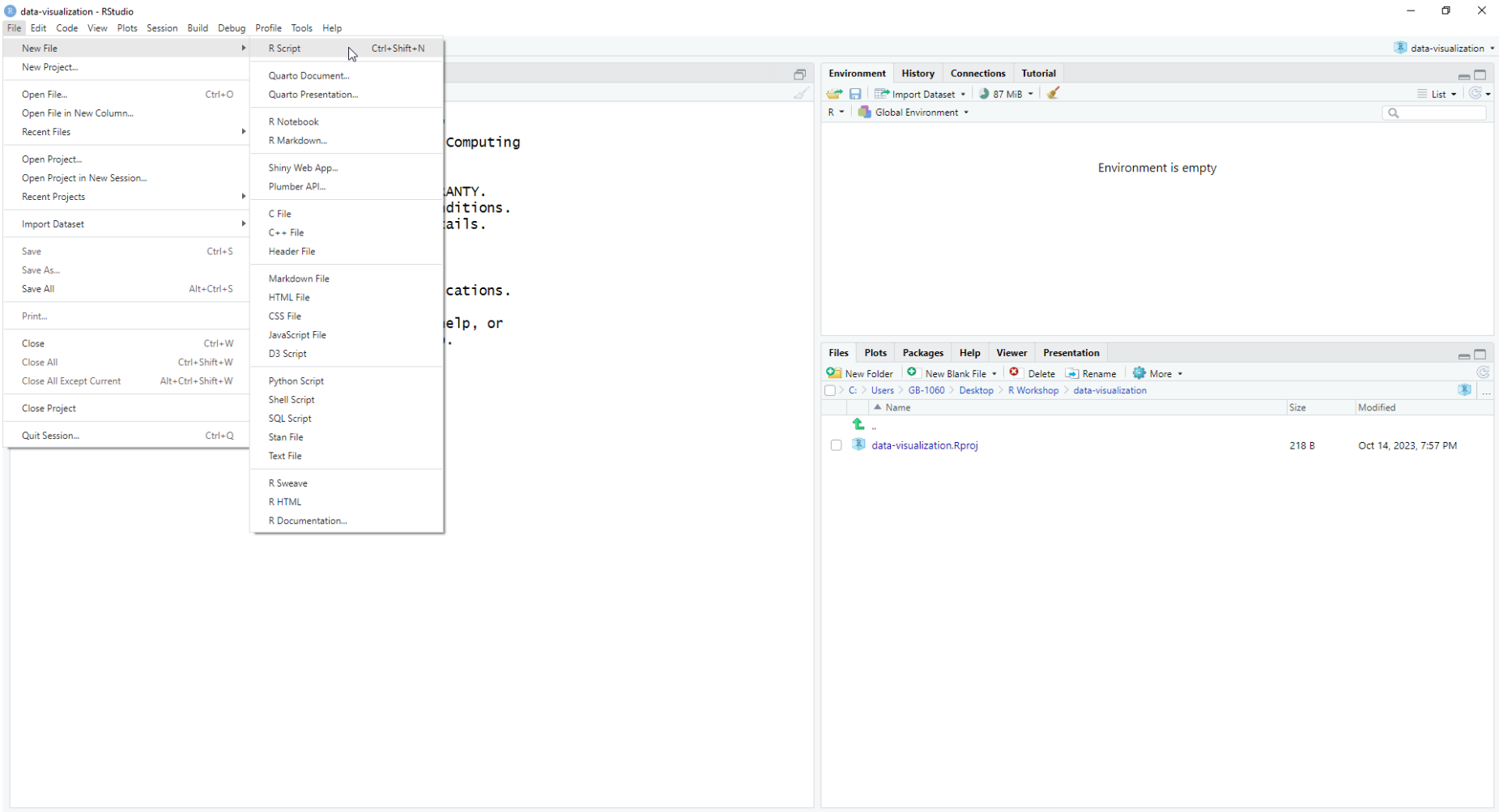
Penguin chicks rescued by unlikely hero | Spy in the Snow - BBC


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
New File
New Project...
Open File... Ctrl+O
Open File in New Column...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save Ctrl+S
Save As...
Save All Alt+Ctrl+S
Print...
Close Ctrl+W
Close All Ctrl+Shift+W
Close All Except Current Alt+Ctrl+Shift+W
Close Project
Quit Session... Ctrl+Q
Addins
Project: (None)
Environment History Connections Tutorial
Import Dataset
89 MiB
List
R
Global Environment
Environment is empty
Files Plots Packages Help Viewer Presentation
R: Arithmetic Mean
Find in Topic
mean {base}
R Documentation
Arithmetic Mean
Description
Generic function for the (trimmed) arithmetic mean.
Usage
mean(x, ...)
Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
Arguments
x
trim
na.rm
...
Value


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
Background Jobs
R 4.3.1 · ~/
R version 4.3.1 (2023-06-16 ucrt) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
Tutorial
Import Dataset
89 MiB
List
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
R Documentation
New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel
ed) arithmetic mean.
= FALSE, ...)
there are methods for numeric/logical vectors and date, date-time and time interval objects.
Complex vectors are allowed for trim = 0, only.
x
trim
the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm
a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds.
...
further arguments passed to or from other methods.
Value
If trim is zero (the default), the arithmetic mean of the values in x is computed, as a numeric or complex vector of length one. If x is not logical (coerced to numeric), numeric (including integer) or complex, NA_real_ is returned, with a warning.
If trim is non-zero, a symmetrically trimmed mean is computed with a fraction of trim observations deleted from each end


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
Background Jobs
R 4.3.1 · ~/
R version 4.3.1 (2023-06-16 ucrt) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
Tutorial
Import Dataset
89 MiB
List
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
R Documentation
ed) arithmetic mean.
n = FALSE, ...)
x
trim
the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm
a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds.
...
further arguments passed to or from other methods.
Value
If trim is zero (the default), the arithmetic mean of the values in x is computed, as a numeric or complex vector of length one. If x is not logical (coerced to numeric), numeric (including integer) or complex, NA_real_ is returned, with a warning.
If trim is non-zero, a symmetrically trimmed mean is computed with a fraction of trim observations deleted from each end
New Project Wizard
Back
Project Type
New Project
R Package
Shiny Application
Quarto Project
Quarto Website
Quarto Blog
Quarto Book
Create a new project in an empty directory
Cancel


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
Background Jobs
R 4.3.1 · ~/
R version 4.3.1 (2023-06-16 ucrt) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
Tutorial
Import Dataset
89 MiB
List
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
R Documentation
New Project Wizard
Back
Create New Project
Directory name:
Create project as subdirectory of:
C:/Users/GB-1060/Desktop/Workspace/GitHub
Browse...
Create a git repository
Use renv with this project
Open in new session
Create Project
Cancel
ed) arithmetic mean.
m = FALSE, ...)
there are methods for numeric/logical vectors and date, date-time and time interval objects.
Complex vectors are allowed for trim = 0, only.
x
trim
the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm
a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds.
...
further arguments passed to or from other methods.
Value
If trim is zero (the default), the arithmetic mean of the values in x is computed, as a numeric or complex vector of length one.
If x is not logical (coerced to numeric), numeric (including integer) or complex, NA_real_ is returned, with a warning.
If trim is non-zero, a symmetrically trimmed mean is computed with a fraction of trim observations deleted from each end


data-visualization - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
New File
New Project...
Open File... Ctrl+O
Open File in New Column...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save Ctrl+S
Save As...
Save All Alt+Ctrl+S
Print...
Close Ctrl+W
Close All Ctrl+Shift+W
Close All Except Current Alt+Ctrl+Shift+W
Close Project
Quit Session... Ctrl+Q
R Script Ctrl+Shift+N
Quarto Document...
Quarto Presentation...
R Notebook
R Markdown...
Shiny Web App...
Plumber API...
C File
C++ File
Header File
Markdown File
HTML File
CSS File
JavaScript File
D3 Script
Python Script
Shell Script
SQL Script
Stan File
Text File
R Sweave
R HTML
R Documentation...
data-visualization
Environment History Connections Tutorial
Import Dataset 87 MiB
List
R Global Environment
Environment is empty
Files Plots Packages Help Viewer Presentation
New Folder New Blank File Delete Rename More
C: Users GB-1060 Desktop R Workshop data-visualization
Name Size Modified
data-visualization.Rproj 218 B Oct 14, 2023, 7:57 PM

Prerequisites

This course focuses on **ggplot2**, one of the core packages in the tidyverse.

```
library(tidyverse)
```

```
library(palmerpenguins)
library(ggthemes)
```

```
install.packages("palmerpenguins")
install.packages("ggthemes")
```

The **palmerpenguins** package includes the `penguins` dataset containing body measurements for penguins on three islands in the Palmer Archipelago.

The **ggthemes** package offers a colorblind safe color palette.

The penguins data frame

- Do penguins with longer flippers weigh more or less than penguins with shorter flippers?

- What does the relationship between flipper length and body mass look like?

- What does the relationship between flipper length and body mass look like?
- Does the relationship vary by the species of the penguin?

- What does the relationship between flipper length and body mass look like?
- Does the relationship vary by the species of the penguin?
- Does the relationship vary by the island where the penguin lives?

`penguins` contains 344 observations collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER.[1]

[1] Horst AM, Hill AP, Gorman KB (2020). palmerpenguins: Palmer Archipelago (Antarctica) penguin data. R package version 0.1.0. https://allisonhorst.github.io/palmerpenguins/. doi: 10.5281/zenodo.3960218.

```
penguins
```

```
glimpse(penguins)
```

Among the variables in penguins are:

- `species`: the penguin's species

Among the variables in penguins are:

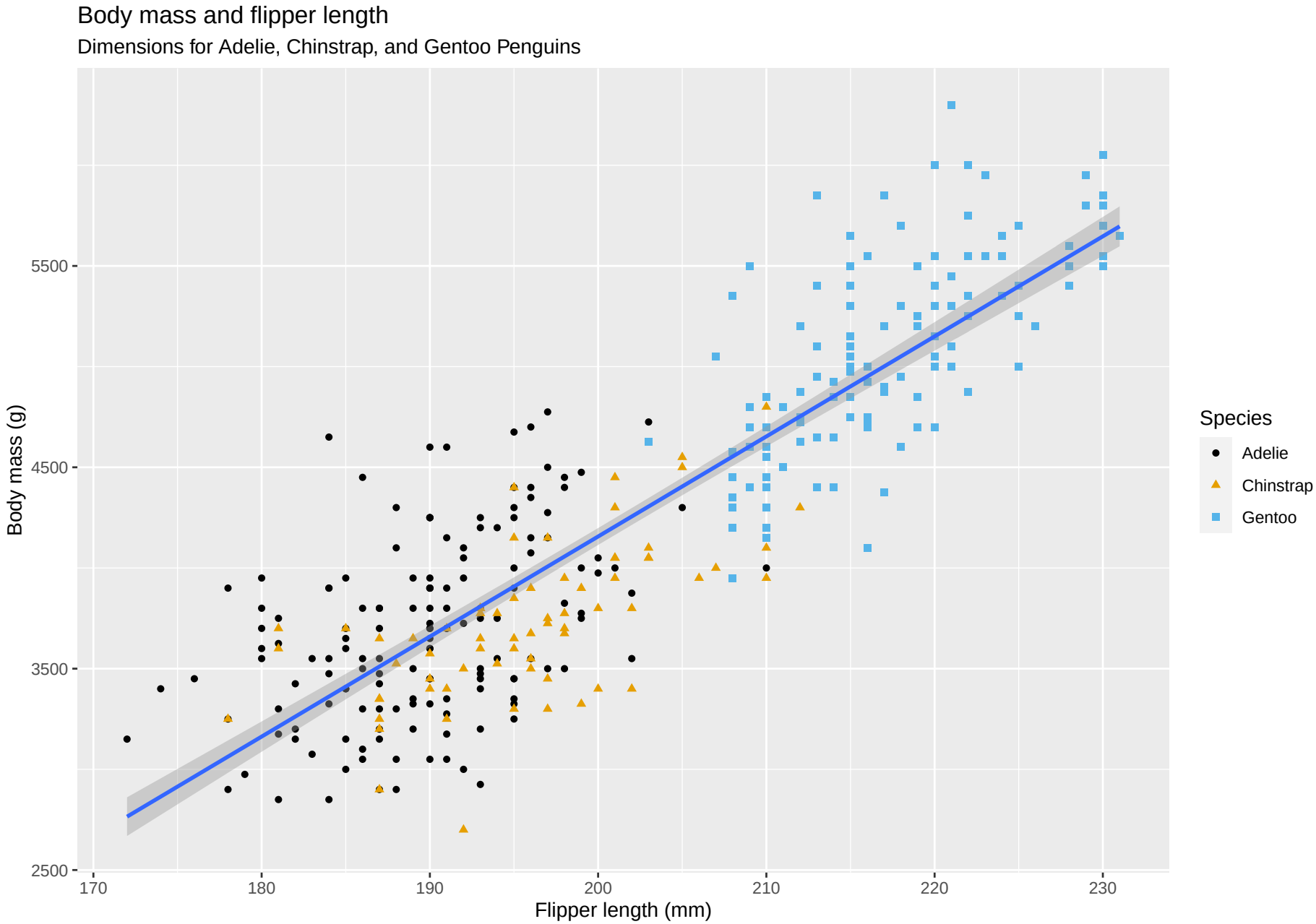
- `species`: the penguin's species
- `flipper_length_mm`: length of a penguin's flipper

Among the variables in penguins are:

- `species`: the penguin's species
- `flipper_length_mm`: length of a penguin's flipper
- `body_mass_g`: body mass of a penguin

Ultimate goal

Our ultimate goal in this section is to create a visualization displaying *the relationship between flipper lengths and body masses of these penguins*, taking into consideration the species of the penguin.


Body mass and flipper length
Dimensions for Adelie, Chinstrap, and Gentoo Penguins
Body mass (g)
5500
4500
3500
2500
170
180
190
200
210
220
230
Flipper length (mm)
Species
Adelie
Chinstrap
Gentoo

```
ggplot(data = penguins)
```

```
ggplot(data = penguins)
```

- The first argument of `ggplot()` is the dataset to use in the graph.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
)
```

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
)
```

- The mapping argument of the `ggplot()` function defines how variables in your dataset are mapped to visual properties (**aesthetics**) of your plot.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point()
```

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point()
```

- **geom**: the geometrical object that a plot uses to represent data

- Bar charts use bar geoms: `geom_bar()`

- Bar charts use bar geoms: `geom_bar()`
- Line charts use line geoms: `geom_line()`

- Bar charts use bar geoms: `geom_bar()`
- Line charts use line geoms: `geom_line()`
- Boxplots use boxplot geoms: `geom_boxplot()`

- Bar charts use bar geoms: `geom_bar()`
- Line charts use line geoms: `geom_line()`
- Boxplots use boxplot geoms: `geom_boxplot()`
- Scatterplots use point geoms: `geom_point()`

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point()
```

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point()
```

- The function `geom_point()` adds a layer of points to your plot, which creates a scatterplot.

What does the relationship between flipper length and body mass look like?

The relationship appears to be

- positive,

The relationship appears to be

- positive,
- fairly linear, and

The relationship appears to be

- positive,
- fairly linear, and
- moderately strong.

Penguins with longer flippers are generally larger in terms of their body mass.

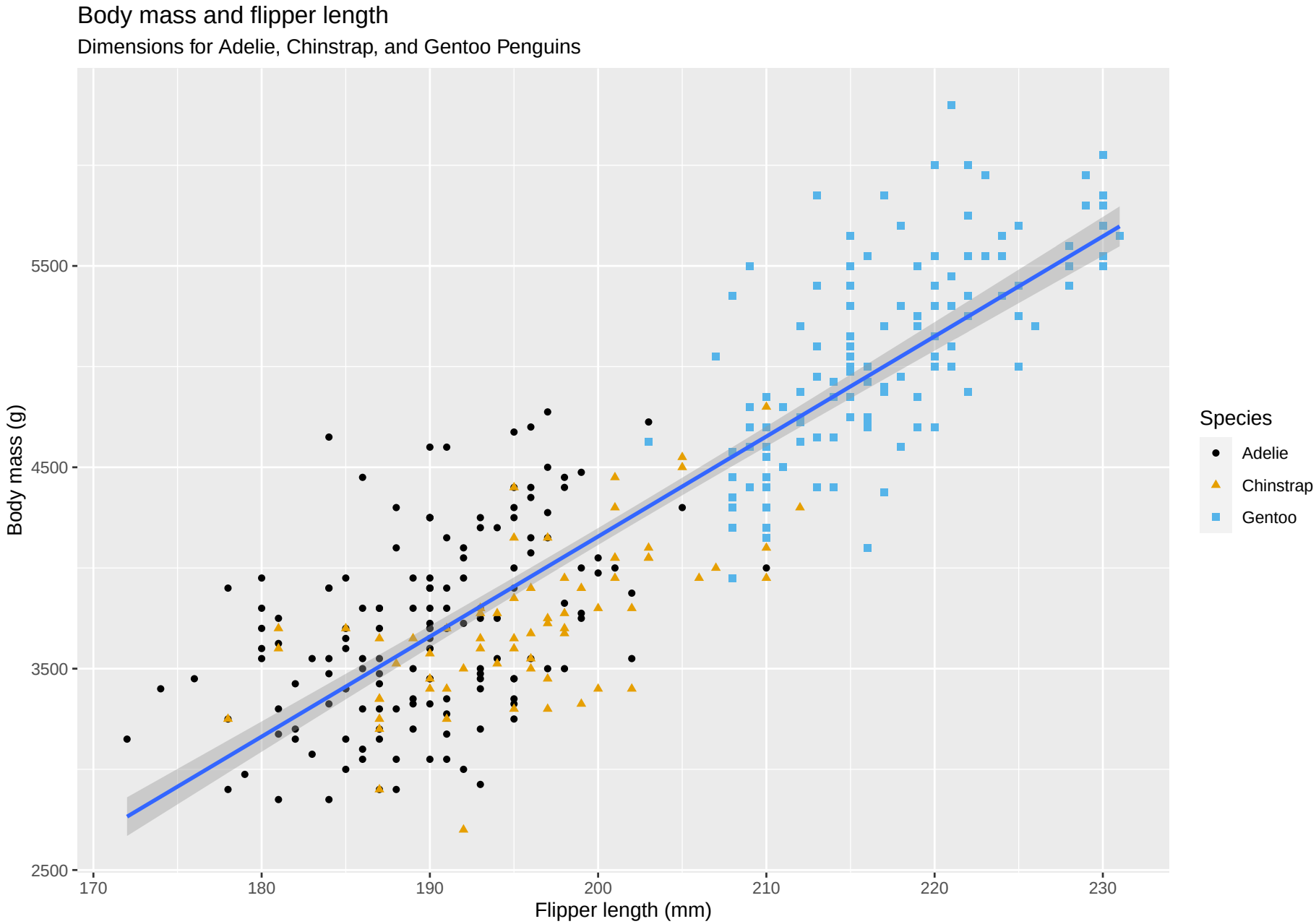
```
#> Warning: Removed 2 rows containing
#> missing values (`geom_point()`).
```

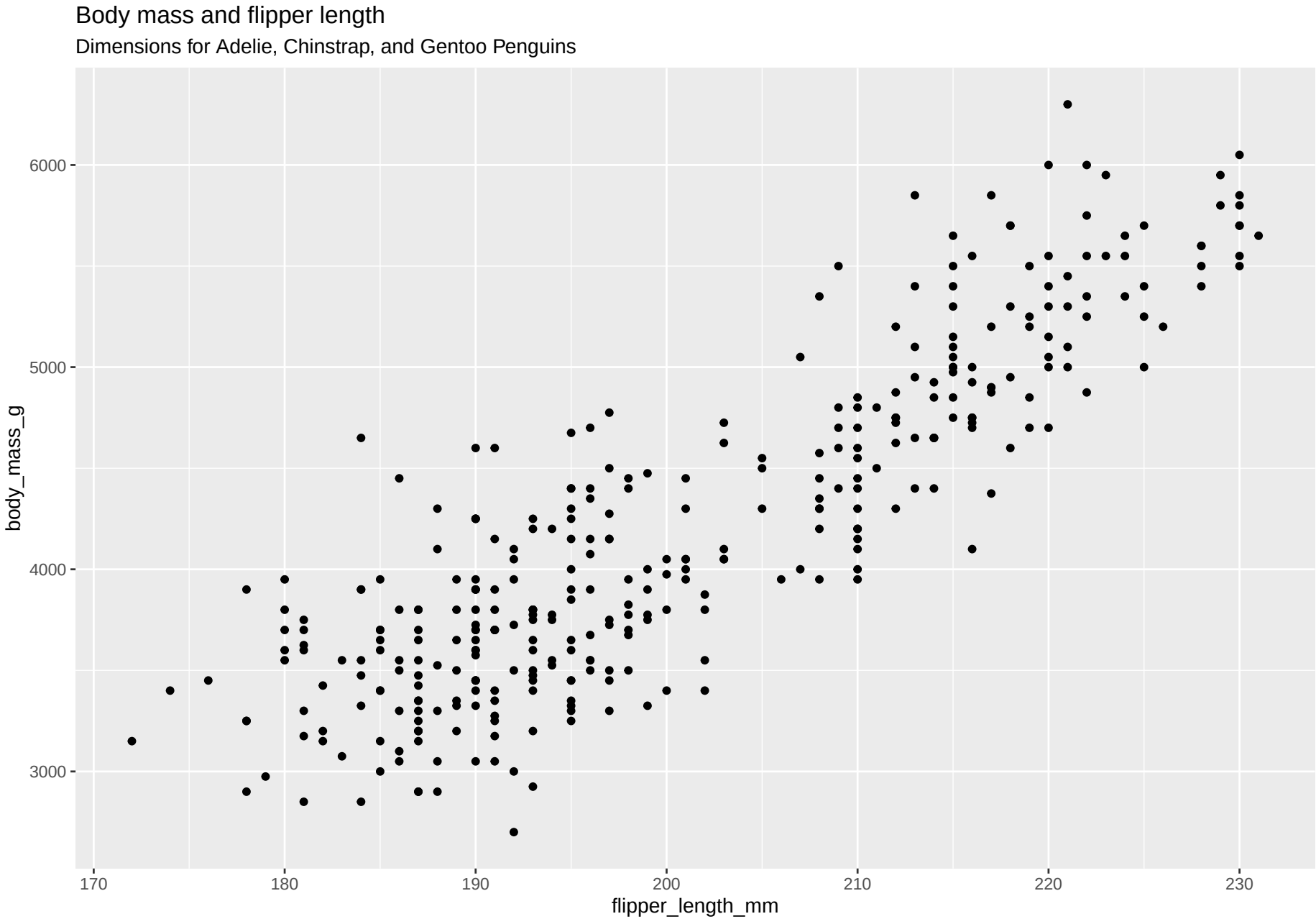
```
#> Warning: Removed 2 rows containing
#> missing values (`geom_point()`).
```

- There are two penguins in our dataset with missing body mass and/or flipper length values.

```
#> Warning: Removed 2 rows containing
#> missing values (`geom_point()`).
```

- There are two penguins in our dataset with missing body mass and/or flipper length values.
- This type of warning is probably one of the most common types of warnings when working with real data.


Body mass and flipper length
Dimensions for Adelie, Chinstrap, and Gentoo Penguins
Body mass (g)
5500
4500
3500
2500
170
180
190
200
210
220
230
Flipper length (mm)
Species
Adelie
Chinstrap
Gentoo


Body mass and flipper length
Dimensions for Adelie, Chinstrap, and Gentoo Penguins
body_mass_g
6000
5000
4000
3000
170
180
190
200
210
220
230
flipper_length_mm

Does the relationship between flipper length and body mass differ by species?

```r
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g,
                color = species)
) +
  geom_point()
```

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g,
                color = species)
) +
  geom_point()
```

- When a categorical variable is mapped to an aesthetic, ggplot2 will automatically assign a unique value of the aesthetic to each unique level of the variable.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g,
                color = species)
) +
  geom_point()
```

- ggplot2 will also add a legend that explains which values correspond to which levels.

```r
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g,
                color = species)
) +
  geom_point() +
  geom_smooth(method = "lm")
```

```r
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g,
                color = species)
) +
  # geom_point() +
  geom_smooth(method = "lm")
```

```r
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g,
                color = species)
) +
  geom_point() +
  geom_smooth(method = "lm")
```

```r
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point(mapping = aes(color = species)) +
  geom_smooth(method = "lm")
```

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point(mapping = aes(color = species)) +
  geom_smooth(method = "lm")
```

Each geom function in ggplot2 can also take a mapping argument, which allows for aesthetic mappings at the **local** level that are added to those inherited from the global level.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g,
                color = species)
) +
  geom_point() +
  geom_smooth(method = "lm")
```

When aesthetic mappings are defined in `ggplot()`, at the **global** level, they're passed down to each of the subsequent geom layers of the plot.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point(mapping = aes(color = species,
                           shape = species)) +
  geom_smooth(method = "lm")
```

```r
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point(aes(color = species,
                 shape = species)) +
  geom_smooth(method = "lm") +
  labs(
    title = "Body mass and flipper length",
    subtitle = "Dimensions for Adelie, Chinstrap,
                and Gentoo Penguins",
    x = "Flipper length (mm)",
    y = "Body mass (g)",
    color = "Species",
    shape = "Species"
  ) +
  scale_color_colorblind()
```

```r
ggsave(filename = "penguin-scatterplot.png")
```

ggplot2 calls

```r
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm,
                y = body_mass_g)
) +
  geom_point()
```

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

Visualizing distributions

How you visualize the distribution of a variable depends on the type of variable: **categorical** or **numerical**.

A categorical variable

```
glimpse(penguins)
```

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

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

A numerical variable

```
glimpse(penguins)
```

A **histogram** divides the x-axis into equally spaced bins and then uses the height of a bar to display the number of observations that fall in each bin.

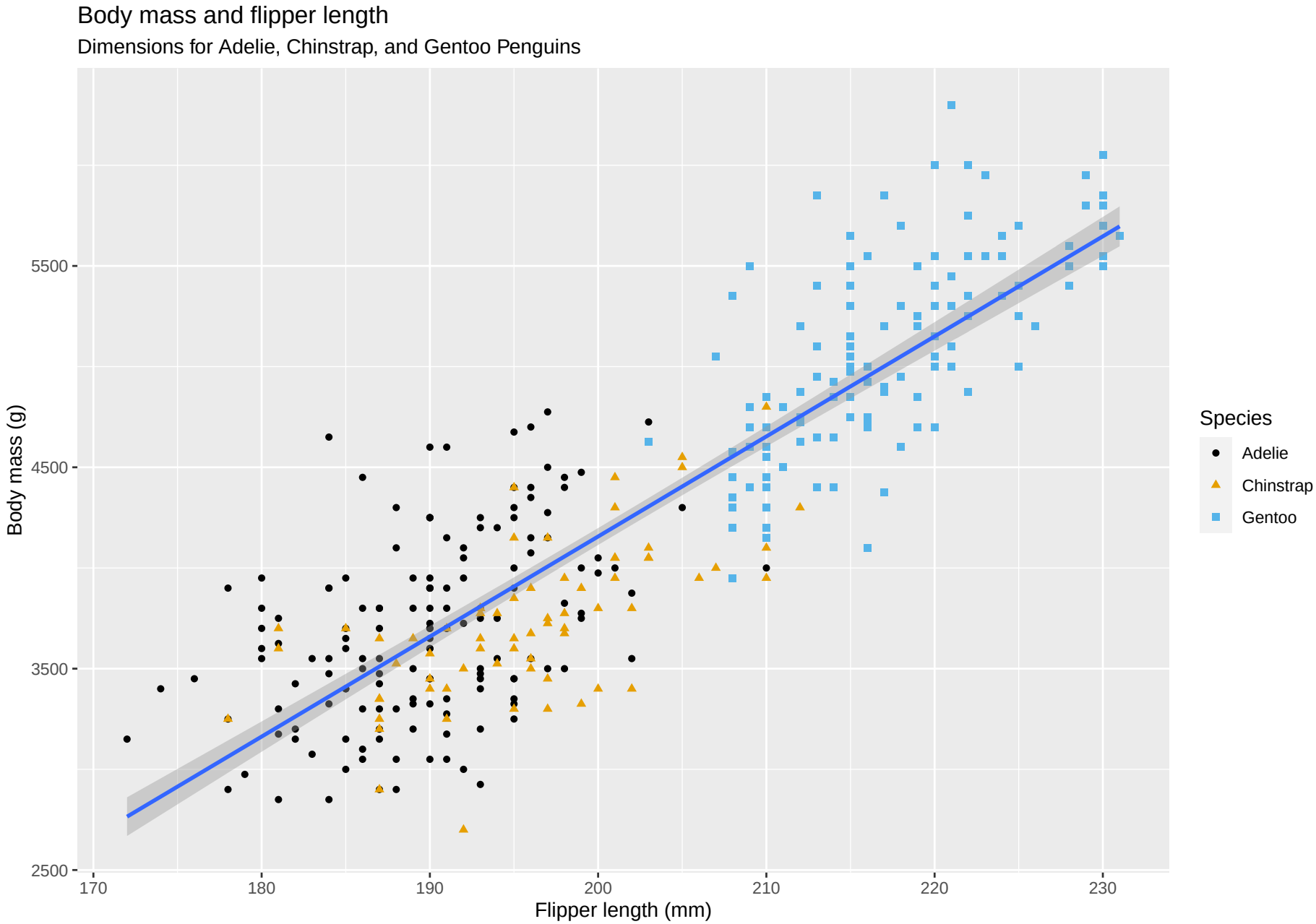
```r
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 200)
```

Different binwidths can reveal different patterns.

```r
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 20)

ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 2000)
```

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


Body mass and flipper length
Dimensions for Adelie, Chinstrap, and Gentoo Penguins
Body mass (g)
5500
4500
3500
2500
170
180
190
200
210
220
230
Flipper length (mm)
Species
Adelie
Chinstrap
Gentoo

Checkpoint

- Create a bar plot of `species` of `penguins`, where you assign species to the `y` aesthetic.

Visualizing relationships

A numerical and a categorical variable

```
glimpse(penguins)
```

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

```r
ggsave(filename = "penguin-boxplot.png")
```

```
ggplot(penguins, aes(x = body_mass_g,
                     color = species)) +
  geom_density(linewidth = 0.75)
```

```
ggplot(penguins, aes(x = body_mass_g,
                     color = species,
                     fill = species)) +
  geom_density(alpha = 0.5)
```

```
ggplot(penguins, aes(x = body_mass_g,
                     color = species,
                     fill = species)) +
  geom_density(alpha = 0.5)
```

We **map** variables to aesthetics if we want the visual attribute represented by that aesthetic to vary based on the values of that variable.

```r
ggsave(filename = "penguin-density-plot.png")
```

Two categorical variables

```
glimpse(penguins)
```

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

```
ggsave(filename = "penguin-barplot.png")
```

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

```
ggsave(filename =
         "penguin-relative-frequency.png")
```

Two numerical variables

```
glimpse(penguins)
```

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

```
ggsave(filename =
         "penguin-scatterplot-simple.png")
```

Three or more variables

```
glimpse(penguins)
```

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

```
ggsave(filename =
         "penguin-scatterplot-islands.png")
```

```
ggplot(penguins, aes(x = flipper_length_mm,
                     y = body_mass_g)) +
  geom_point(aes(color = species,
                 shape = species)) +
  facet_wrap(~island)
```

```
ggplot(penguins, aes(x = flipper_length_mm,
                     y = body_mass_g)) +
  geom_point(aes(color = species,
                 shape = species)) +
  facet_wrap(~island)
```

- **facets**: subplots that each display one subset of the data.

```
ggsave(filename =
         "penguin-scatterplot-facets.png")
```

Checkpoint

- Create a scatterplot of `bill_depth_mm` vs. `bill_length_mm` and color the points by `species`.

Checkpoint

- Create a scatterplot of `bill_depth_mm` vs. `bill_length_mm` and color the points by `species`.
- What does adding coloring by species reveal about the relationship between these two variables?

Checkpoint

- Create a scatterplot of `bill_depth_mm` vs. `bill_length_mm` and color the points by `species`.
- What does adding coloring by species reveal about the relationship between these two variables?
- What about faceting by `species`?

Data visualization with ggplot2 : : CHEATSHEET

Basics

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

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

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                  required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT>, position = <POSITION>) +                  Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

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

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5**)** Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Aes

Common aesthetic values.

color and **fill** - string ("red", "#RRGGBB")

linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

size - integer (line width in mm)

shape - integer/shape name or a single character ("a")

Geoms

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

GRAPHICAL PRIMITIVES

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

a + geom_blank() and **a + expand_limits()** Ensure limits include values across all plots.

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(alpha = 50)**)** - x, y, alpha, color, fill, group, subgroup, linetype, size

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

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

LINE SEGMENTS

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

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

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

ONE VARIABLE continuous

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

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

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

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

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

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

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

discrete

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

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

TWO VARIABLES

both continuous

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

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

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**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

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

both discrete

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

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

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

continuous bivariate distribution

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

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

h + geom_density_2d() x, y, alpha, color, group, linetype, size

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

continuous function

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

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

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

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

visualizing error

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

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

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

j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

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

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

THREE VARIABLES

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

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

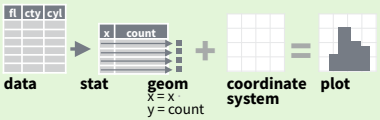
l + geom_contour_filled(aes(fill = z)**)** x, y, alpha, color, fill, group, linetype, size, subgroup

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

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

Stats An alternative way to build a layer.

A stat builds new variables to plot (e.g., count, prop).



Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function).
Use **after_stat(name)** syntax to map the stat variable **name** to an aesthetic.

geom to use | stat function | geommappings

i + stat_density_2d(aes(fill = after_stat(level)), geom = "polygon"**)**

variable created by stat

c + stat_bin(binwidth = 1, boundary = 10**)**
x, y | count, ncount, density, ndensity

c + stat_count(width = 1**) x, y** | count, prop

c + stat_density(adjust = 1, kernel = "gaussian"**)**
x, y | count, density, scaled

e + stat_bin_2d(bins = 30, drop = T**)**
x, y, fill | count, density

e + stat_bin_hex(bins = 30**) x, y, fill** | count, density

e + stat_density_2d(contour = TRUE, n = 100**)**
x, y, color, size | level

e + stat_ellipse(level = 0.95, segments = 51, type = "t"**)**

l + stat_contour(aes(z = z)**) x, y, z, order** | level

l + stat_summary_hex(aes(z = z), bins = 30, fun = max**)**
x, y, z, fill | value

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | value

f + stat_boxplot(coef = 1.5**)**
x, y | lower, middle, upper, width , ymin, ymax

f + stat_ydensity(kernel = "gaussian", scale = "area"**) x, y** | density, scaled, count, n, violinwidth, width

e + stat_ecdf(n = 40**) x, y** | x, y

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq"**) x, y** | quantile

e + stat_smooth(method = "lm", formula = y ~ x, se = T, level = 0.95**) x, y** | se, x, y, ymin, ymax

ggplot() + xlim(-5, 5**) + stat_function(**fun = dnorm, n = 20, geom = "point"**) x** | x, y

ggplot() + stat_qq(aes(sample = 1:100)**)**
x, y, sample | sample, theoretical

e + stat_sum() x, y, size | n, prop

e + stat_summary(fun.data = "mean_cl_boot"**)**

h + stat_summary_bin(fun = "mean", geom = "bar"**)**

e + stat_identity()

e + stat_unique()

Scales Override defaults with **scales** package.

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

n <- d + geom_bar(aes(fill = fl))

scale_ | aesthetic to adjust | prepackaged scale to use | scale-specific arguments

n + scale_fill_manual(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

range of values to include in | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

GENERAL PURPOSE SCALES

Use with most aesthetics

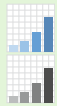
scale_*_continuous() - Map cont' values to visual ones.
scale_*_discrete() - Map discrete values to visual ones.
scale_*_binned() - Map continuous values to discrete bins.
scale_*_identity() - Use data values as visual ones.
scale_*_manual(values = c()**)** - Map discrete values to manually chosen visual ones.
scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks"**)** - Treat data values as dates.
scale_*_datetime() - Treat data values as date times. Same as scale_*_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale.
scale_x_reverse() - Reverse the direction of the x axis.
scale_x_sqrt() - Plot x on square root scale.

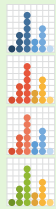
COLOR AND FILL SCALES (DISCRETE)



n + scale_fill_brewer(palette = "Blues"**)**
For palette choices:
RColorBrewer::display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red"**)**

COLOR AND FILL SCALES (CONTINUOUS)



o <- c + geom_dotplot(aes(fill = x))

o + scale_fill_distiller(palette = "Blues"**)**

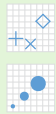
o + scale_fill_gradient(low="red", high="yellow"**)**

o + scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25**)**

o + scale_fill_gradientn(colors = topo.colors(6)**)**
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES

p <- e + geom_point(aes(shape = fl, size = cyl))



p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7)**)**

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

p + scale_radius(range = c(1,6)**)**
p + scale_size_area(max_size = 6**)**

Coordinate Systems

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5)**)** - xlim, ylim
The default cartesian coordinate system.

r + coord_fixed(ratio = 1/2)
ratio, xlim, ylim - Cartesian coordinates with fixed aspect ratio between x and y units.

r + coord_flip()
Flip cartesian coordinates by switching x and y aesthetic mappings.

r + coord_polar(theta = "x", direction=1**)**
theta, start, direction - Polar coordinates.

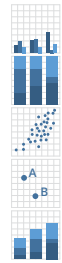
r + coord_trans(y = "sqrt"**)** - x, y, xlim, ylim
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation = c(41, -74, 0)**)** - projection, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.).

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))



s + geom_bar(position = "dodge")
Arrange elements side by side.

s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height.

e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting.

e + geom_label(position = "nudge")
Nudge labels away from points.

s + geom_bar(position = "stack")
Stack elements on top of one another.

Each position adjustment can be recast as a function with manual **width** and **height** arguments:
s + geom_bar(position = position_dodge(width = 1))

Themes

r + theme_bw()
White background with grid lines.

r + theme_gray()
Grey background (default theme).

r + theme_dark()
Dark for contrast.

r + theme_classic()

r + theme_light()

r + theme_linedraw()

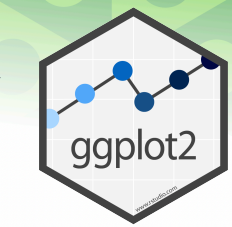
r + theme_minimal()
Minimal theme.

r + theme_void()
Empty theme.

r + theme() Customize aspects of the theme such as axis, legend, panel, and facet properties.
r + ggtitle("Title") + theme(plot.title.postion = "plot")
r + theme(panel.background = element_rect(fill = "blue"))

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.



t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(. ~ fl)
Facet into columns based on fl.

t + facet_grid(year ~ .)
Facet into rows based on year.

t + facet_grid(year ~ fl)
Facet into both rows and columns.

t + facet_wrap(~ fl)
Wrap facets into a rectangular layout.

Set **scales** to let axis limits vary across facets.

t + facet_grid(drv ~ fl, scales = "free")
x and y axis limits adjust to individual facets:
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet label:

t + facet_grid(. ~ fl, labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(fl ~ ., labeller = label_bquote(alpha ^ .(fl)**))**

α^c	α^d	α^e	α^p	α^r

Labels and Legends

Use **labs()** to label the elements of your plot.

t + labs(x = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
alt = "Add alt text to the plot",
<AES> = "New <AES> legend title"**)**

t + annotate(geom = "text", x = 8, y = 9, label = "A"**)**
Places a geom with manually selected aesthetics.

p + guides(x = guide_axis(n.dodge = 2)**)** Avoid crowded or overlapping labels with guide_axis(n.dodge or angle).

n + guides(fill = "none"**)** Set legend type for each aesthetic: colorbar, legend, or none (no legend).

n + theme(legend.position = "bottom"**)**
Place legend at "bottom", "top", "left", or "right".

n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E"**)**
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred):
t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20)**)**

With clipping (removes unseen data points):
t + xlim(0, 100**) + ylim(**10, 20**)**
t + scale_x_continuous(limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**