Foundations of Data Science for Biologists

Visualizing your data

BIO 724D

25-SEP-2023

Instructors: Greg Wray, Paul Magwene, (Jesse Granger)

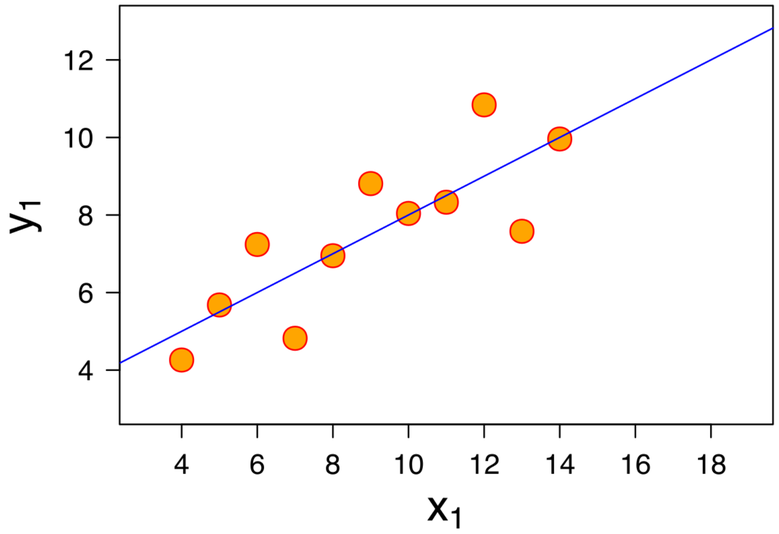
The power of visualization and an introduction to ggplot2

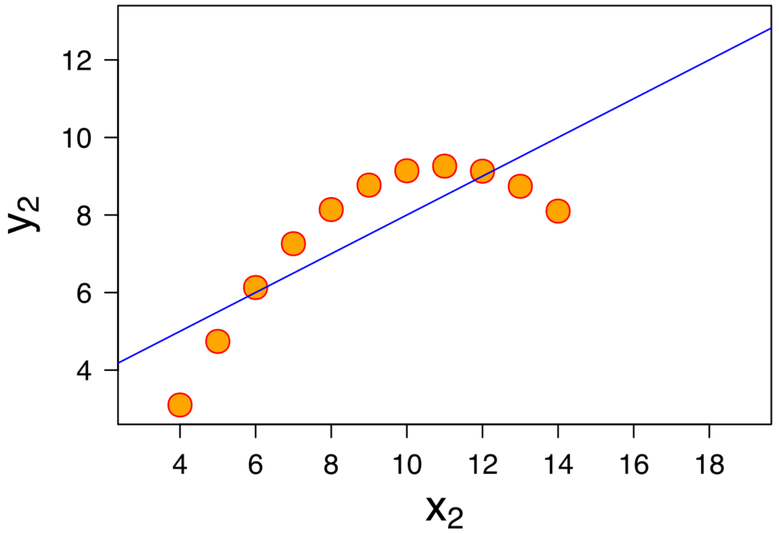
Anscombe’s Quartet

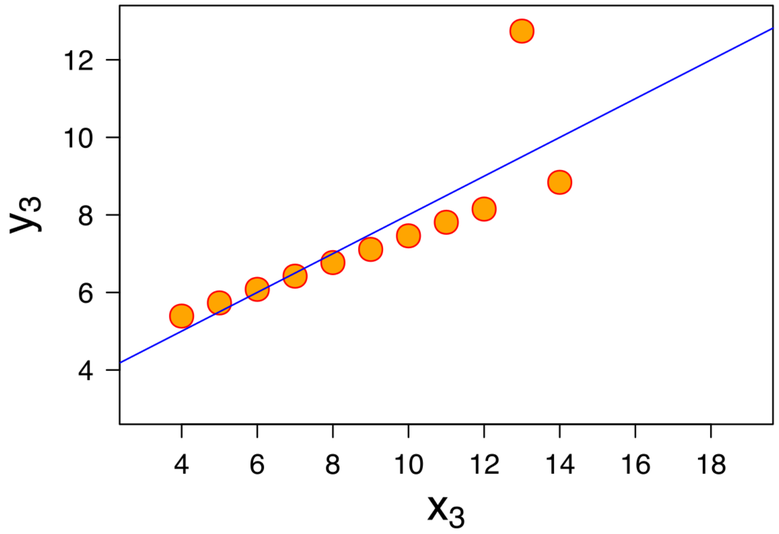
	data set I		data set II		data set III		data set IV	
	x	y	x	y	x	y	x	y
	10.00	8.04	10.00	9.14	10.00	7.46	8.00	6.58
	8.00	6.95	8.00	8.14	8.00	6.77	8.00	5.76
	13.00	7.58	13.00	8.74	13.00	12.74	8.00	7.71
	9.00	8.81	9.00	8.77	9.00	7.11	8.00	8.84
	11.00	8.33	11.00	9.26	11.00	7.81	8.00	8.47
	14.00	9.96	14.00	8.10	14.00	8.84	8.00	7.04
	6.00	7.24	6.00	6.13	6.00	6.08	8.00	5.25
	4.00	4.26	4.00	3.10	4.00	5.39	19.00	12.50
	12.00	10.84	12.00	9.13	12.00	8.15	8.00	5.56
	7.00	4.82	7.00	7.26	7.00	6.42	8.00	7.91
	5.00	5.68	5.00	4.74	5.00	5.73	8.00	6.89

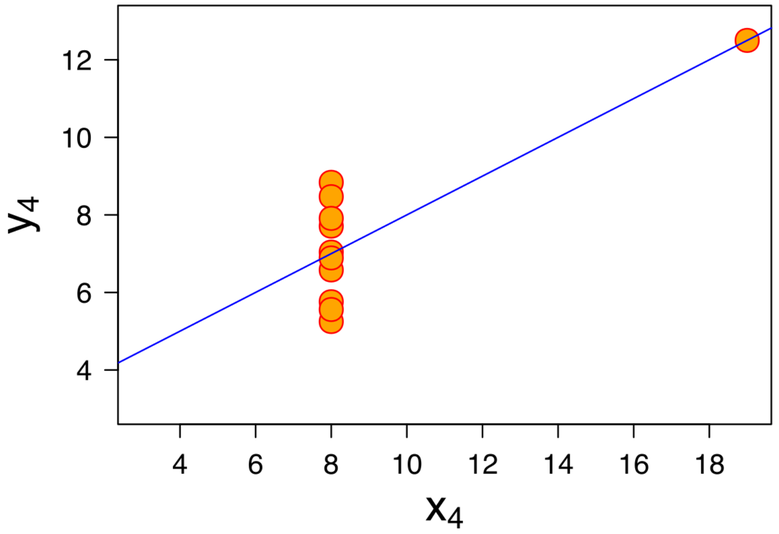
Anscombe, FJ (1973) *American Statistician* 27:17-21

Anscombe’s Quartet visualized









Exploratory visualization

"The greatest value of a picture is when it forces us to notice
what we never expected to see"

John Tukey, *Exploratory Data Analysis*

"Discovering the unexpected is more important than confirming the known."

George E. P. Box

"You can observe a lot by looking."

Yogi Berra

The concept behind ggplot2

Traditional graphics treats each type of plot separately

- Many different functions, each with distinct syntax and arguments
- Hard to learn and remember; hard to read / update code later
- Hard to experiment with different plot types

ggplot2 is built around the idea of a “grammar of graphics” (Wilkinson 1999)

- Core concept: define plots based on semantic components

Important advantages:

- Uses consistent syntax and arguments for all plot types: easier to learn / code
- Allows combination of plot types through layering, multiple plots through faceting
- Provides customizable themes: consistency is easier for audiences / readers to follow

Wilkinson, L (1999) “The Grammar of Graphics” Springer Press.

ggplot2 basic syntax

building “up” metaphor

```
ggplot(data = frame, mapping = aes(columns)) + geom_X()
```

Image credit: RPubs by RStudio

ggplot2 basic syntax

building “up” metaphor

```
ggplot(data = penguins,
    mapping = aes(x = bill_depth_mm, y = bill_length_mm)) +
    geom_point()
```

Image credit: RPubs by RStudio

ggplot2 basic syntax

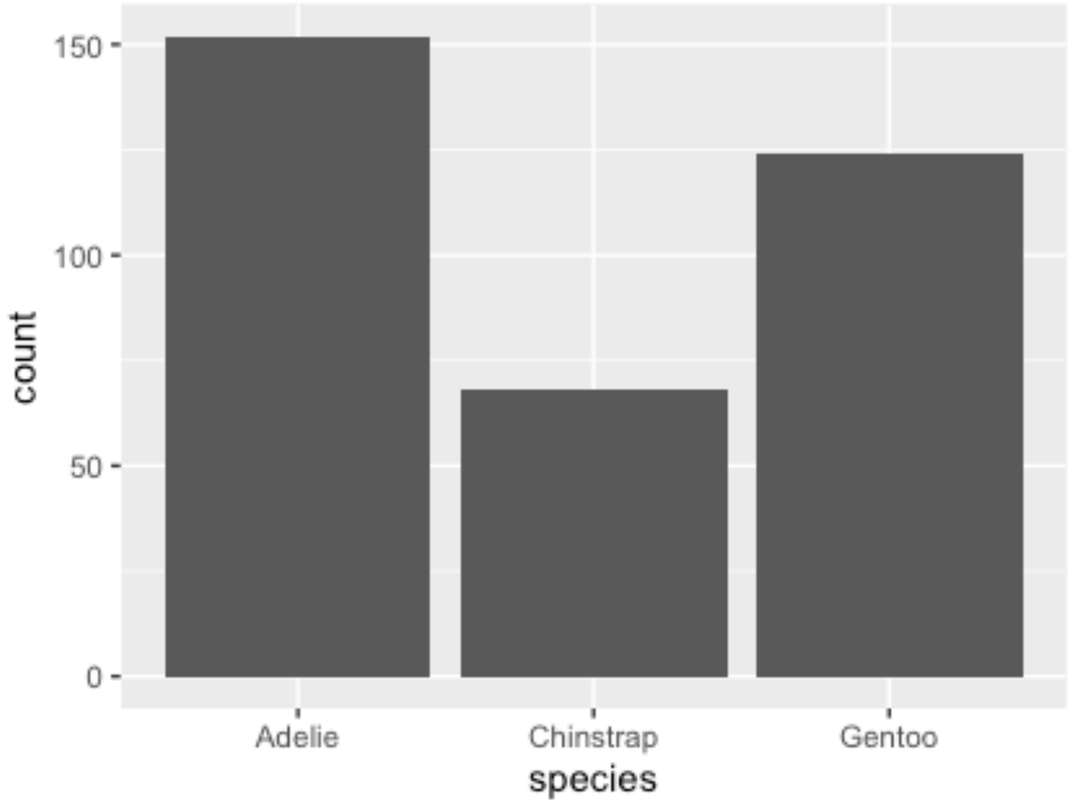
building “up” metaphor

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

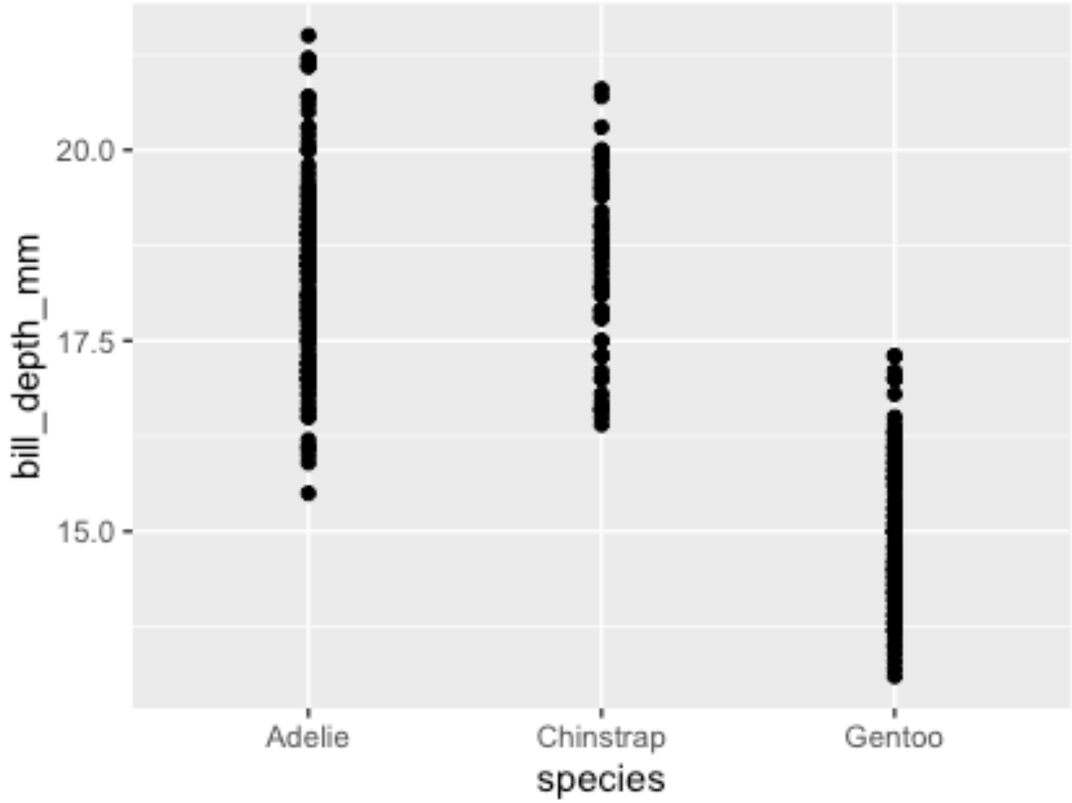
Image credit: RPubs by RStudio

ggplot2 basic syntax

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

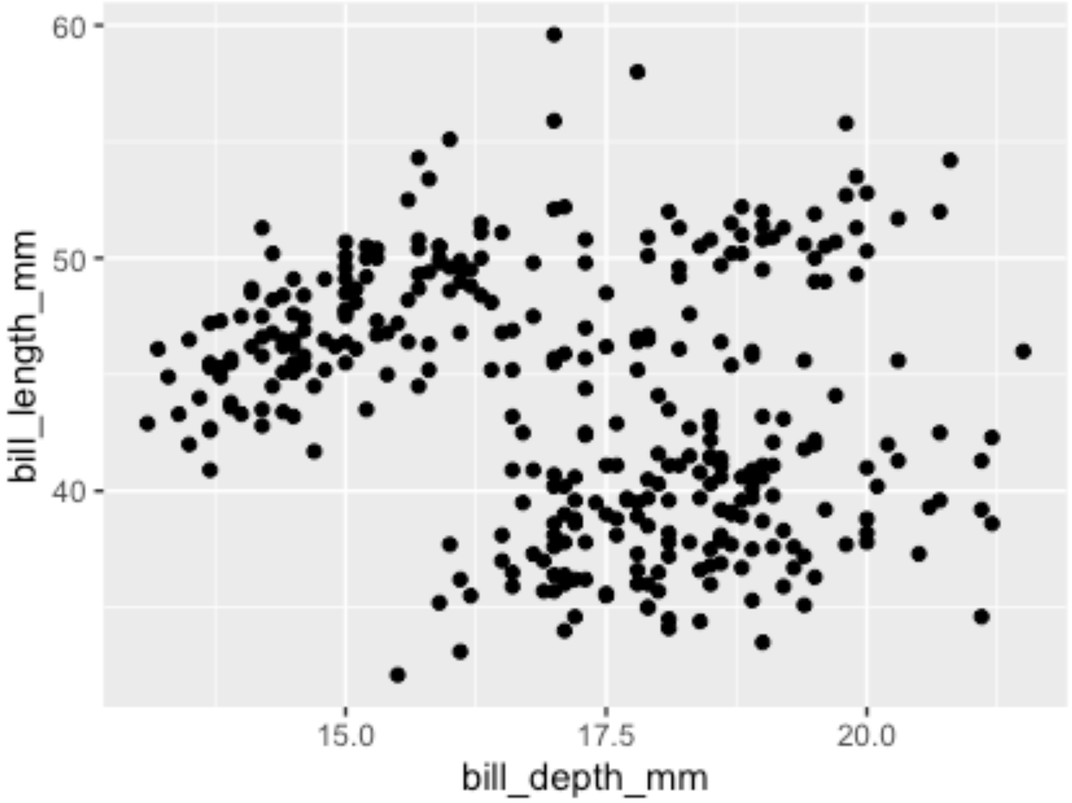


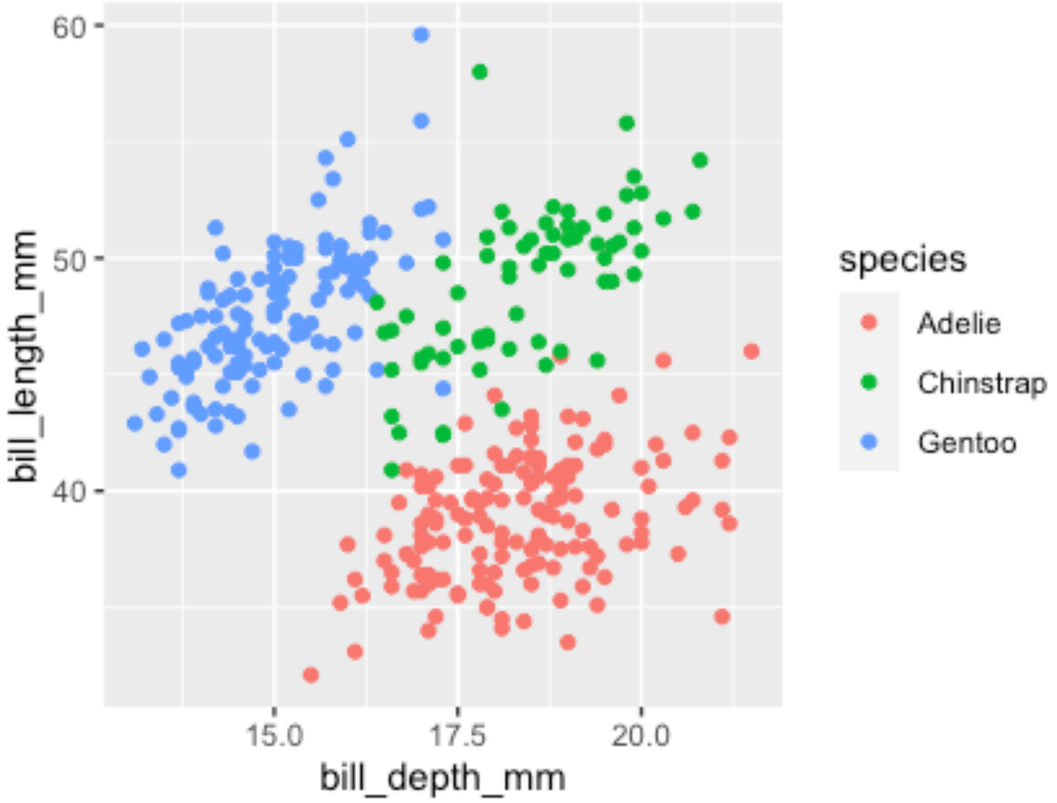

```
ggplot(penguins,
    aes(x=species, y=bill_depth_mm)) +
    geom_point()
```



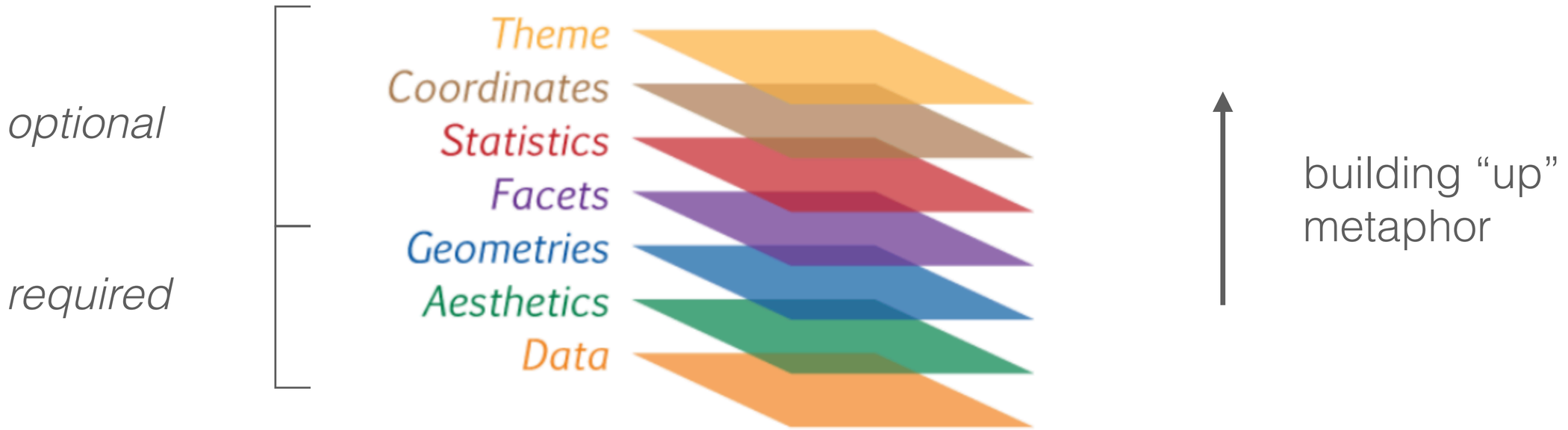
ggplot2 basic syntax, continued

```
ggplot(penguins,
    aes(x=bill_depth_mm, y=bill_length_mm, color = species)) +
    geom_point()
```





ggplot2 required and optional components



Theme: every part of the graph *not* related directly to data; the “look and feel”

Coordinates: how variables are mapped, scaled, and applied to a particular geometry

Statistics: calculations of derived values; linked to specific geoms

Facets: specify multiple panels with equivalent logic, split data among panels

Image credit: RPubs by RStudio

Group exercise

Install and load the `palmerpenguins` package (available from CRAN or link on class wiki)

We will be working with the `penguins` data set that is part of this package

Make a bar plot of the number of individuals from each island using `geom_bar()`

Make a scatterplot of `bill_depth_mm` by `bill_length_mm` and color by `island`

Make a box plot of `bill_depth_mm` using `geom_boxplot()`

- Rotate the plot so that `bill_depth_mm` is on the y-axis and plot by species

Convert the above to a violin plot using `geom_violin()`

Best practices for graphs

A graphic should convey information

Clearly: the focus is on the data

Accurately: no distortions, distractions, or critical omissions in the data presented

Intuitively: with minimal need for explanation

Revealingly: uncovers trends and organization in data; ideally leads to insight

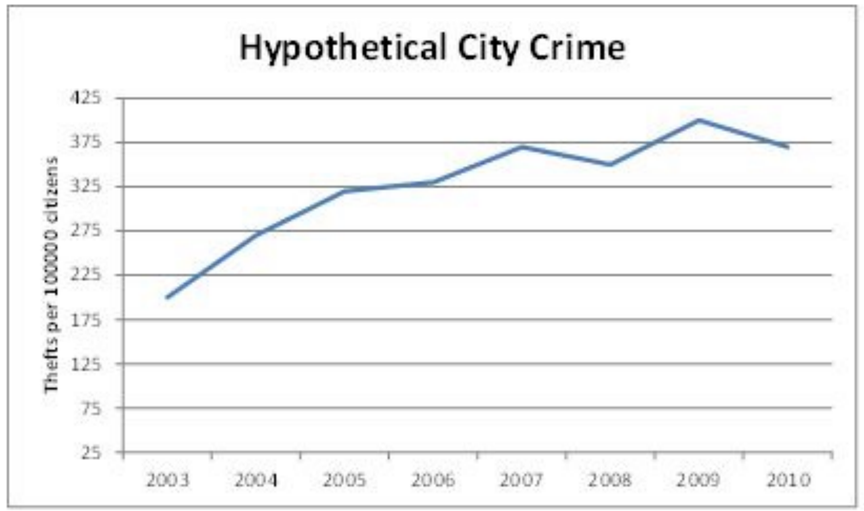
Engagingly: invites the viewer to think about what results mean

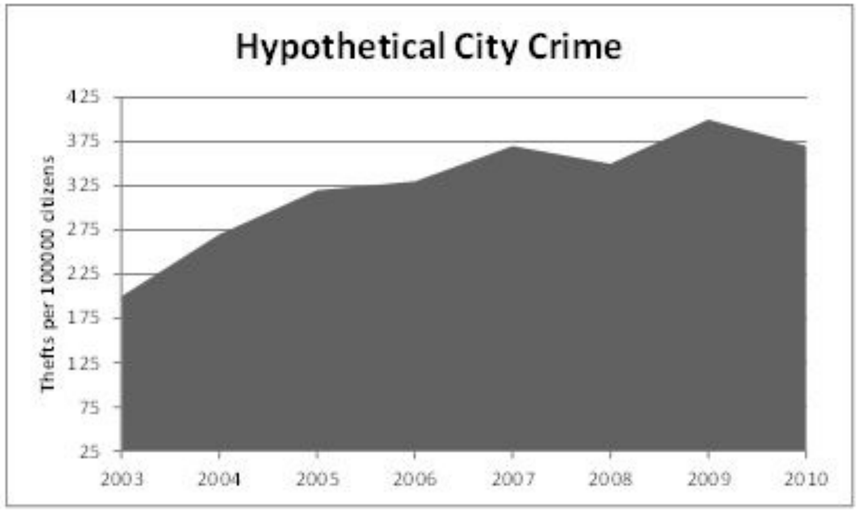
And it should fit into the overall presentation:

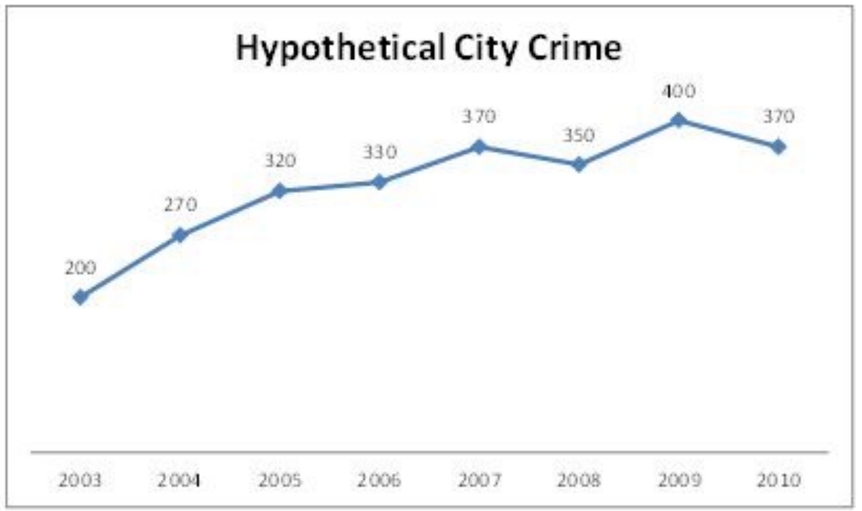
Consistently: orientation, scale, color, font, encodings, expected location, etc.

Relationally: connects to the overall narrative and to other visuals

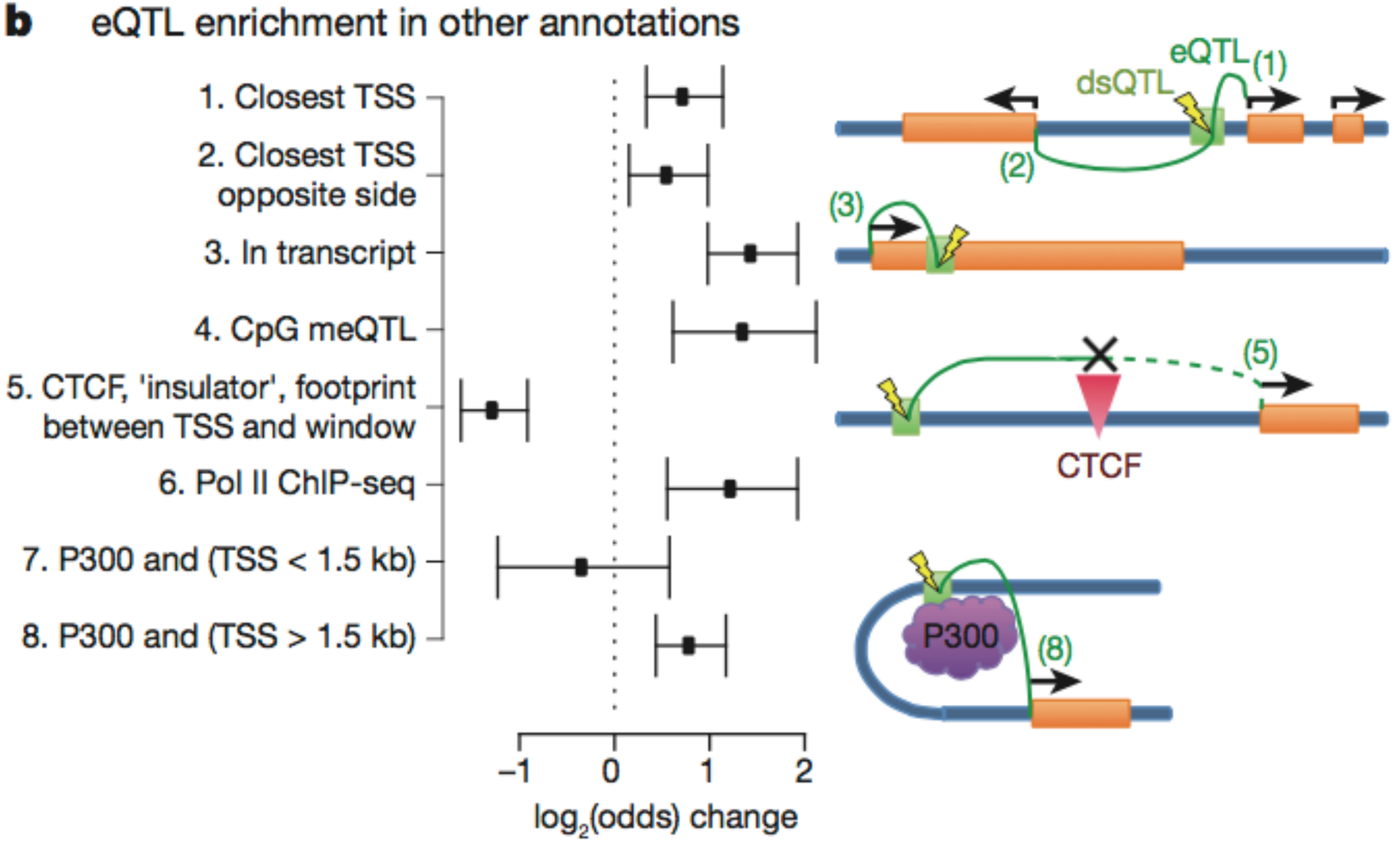
Make the data the focus





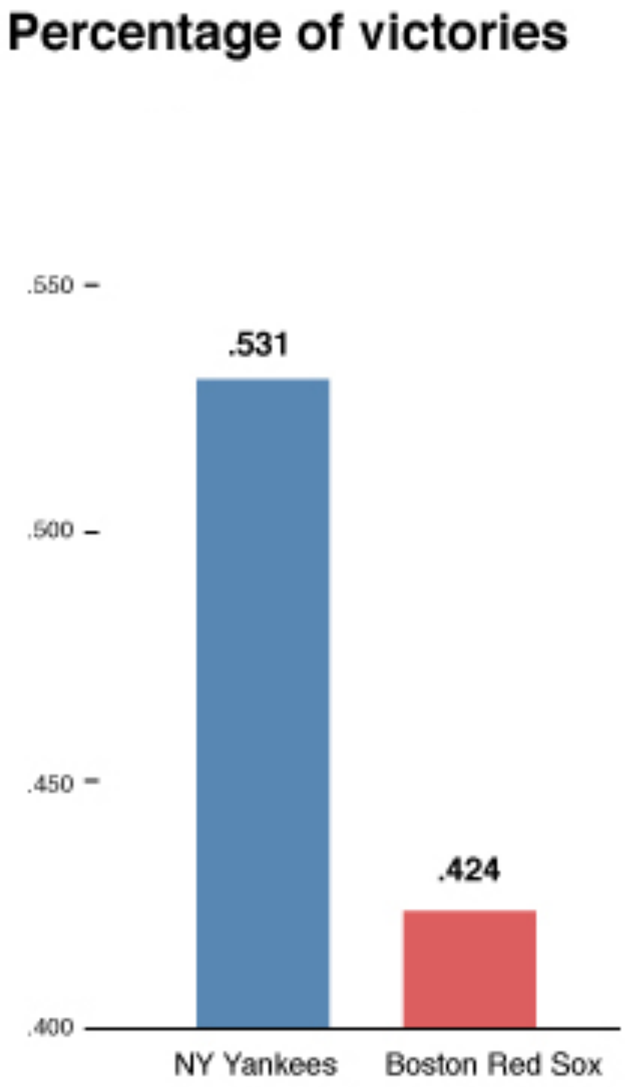


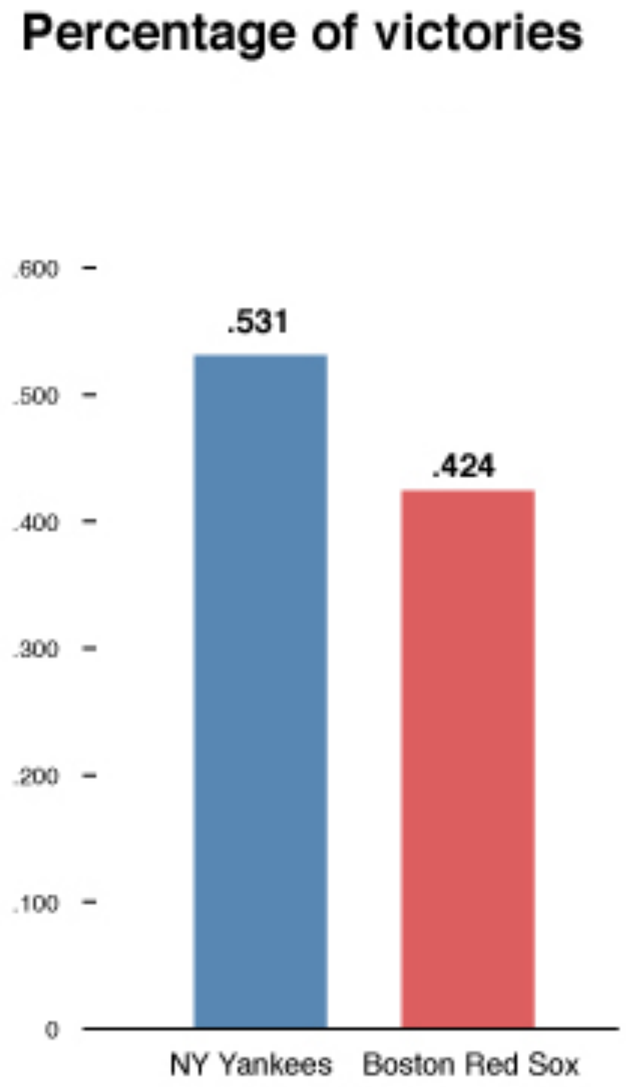
Make the data the focus



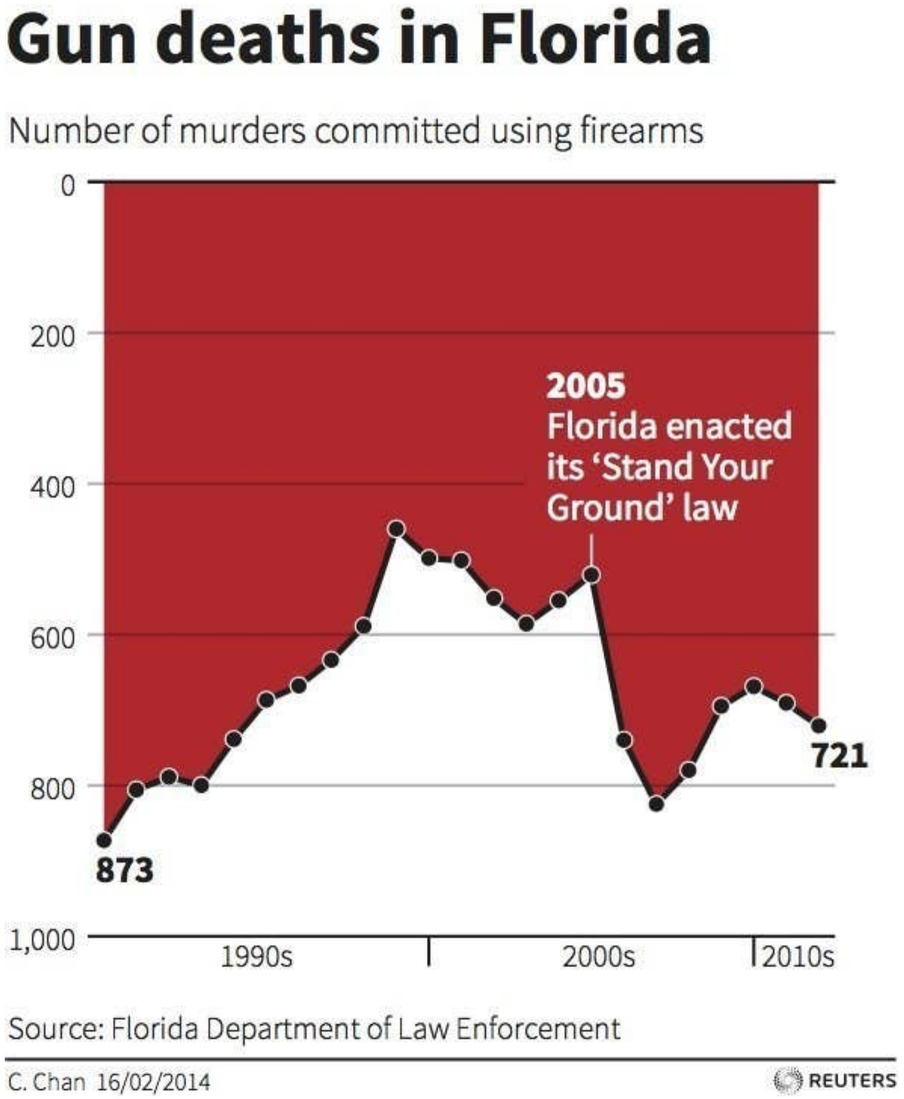
Degner et al. (2012) *Nature* 482:390

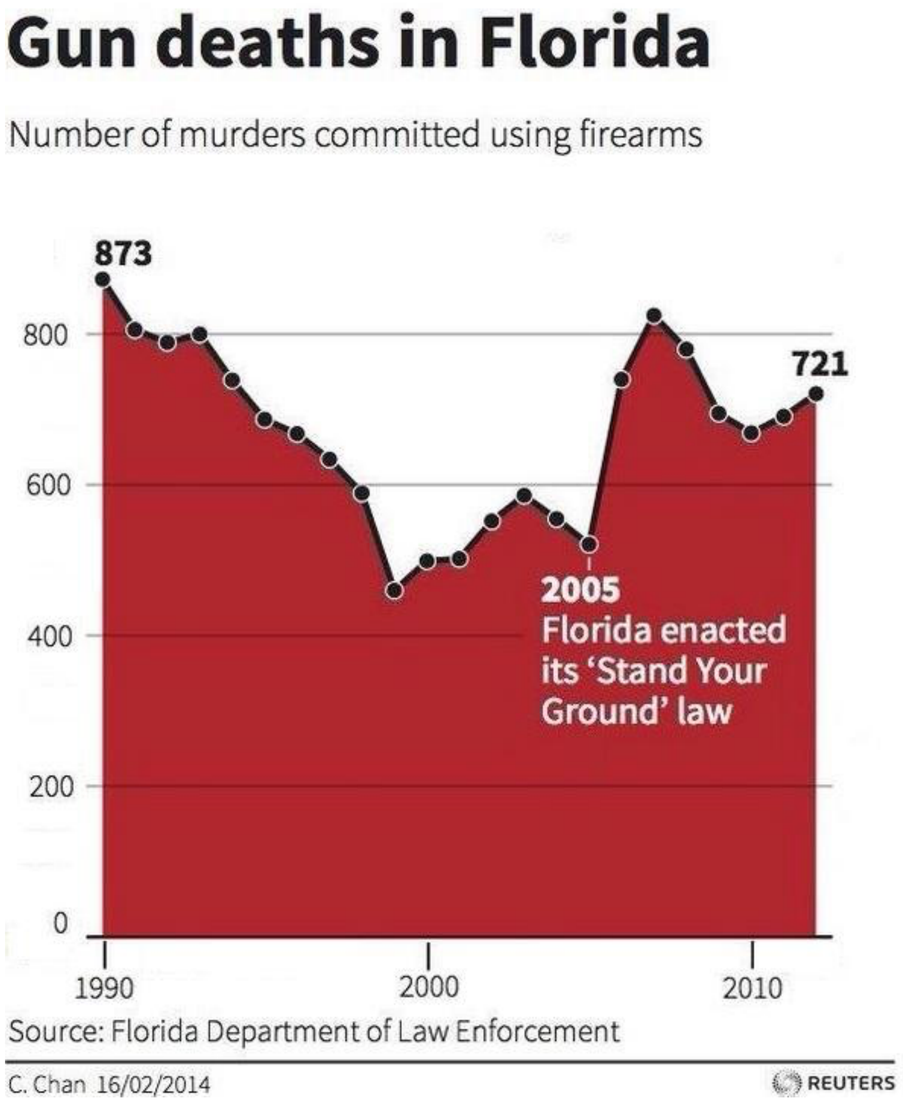
Present the data honestly



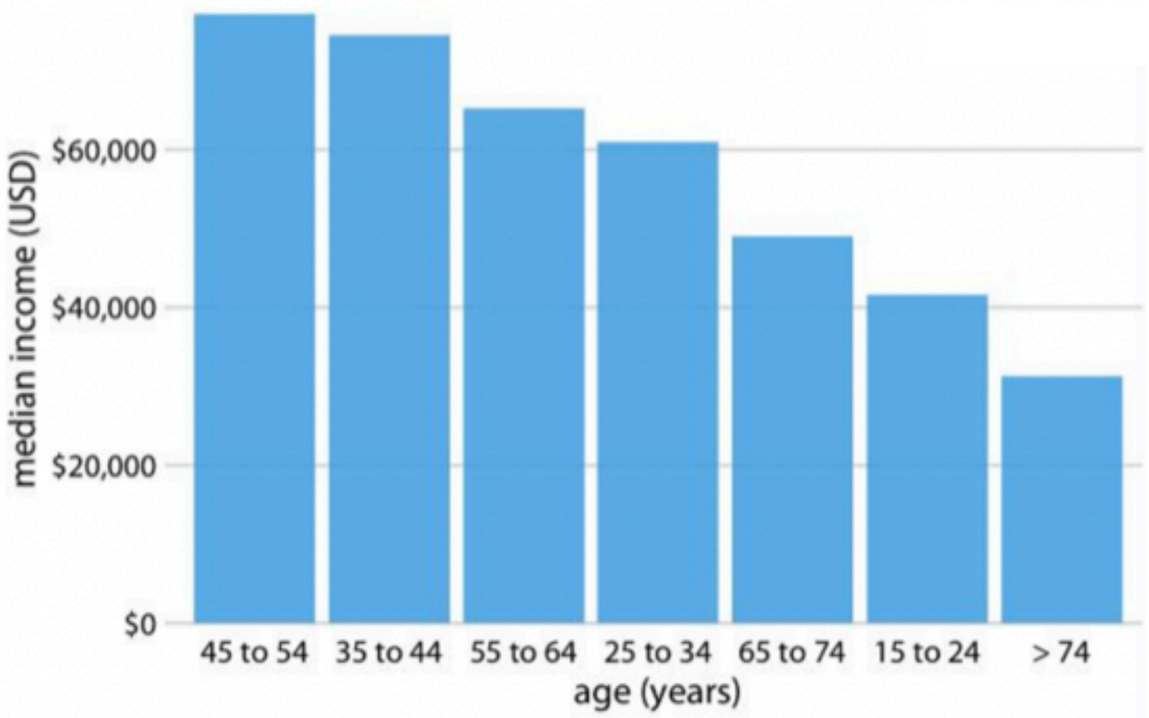


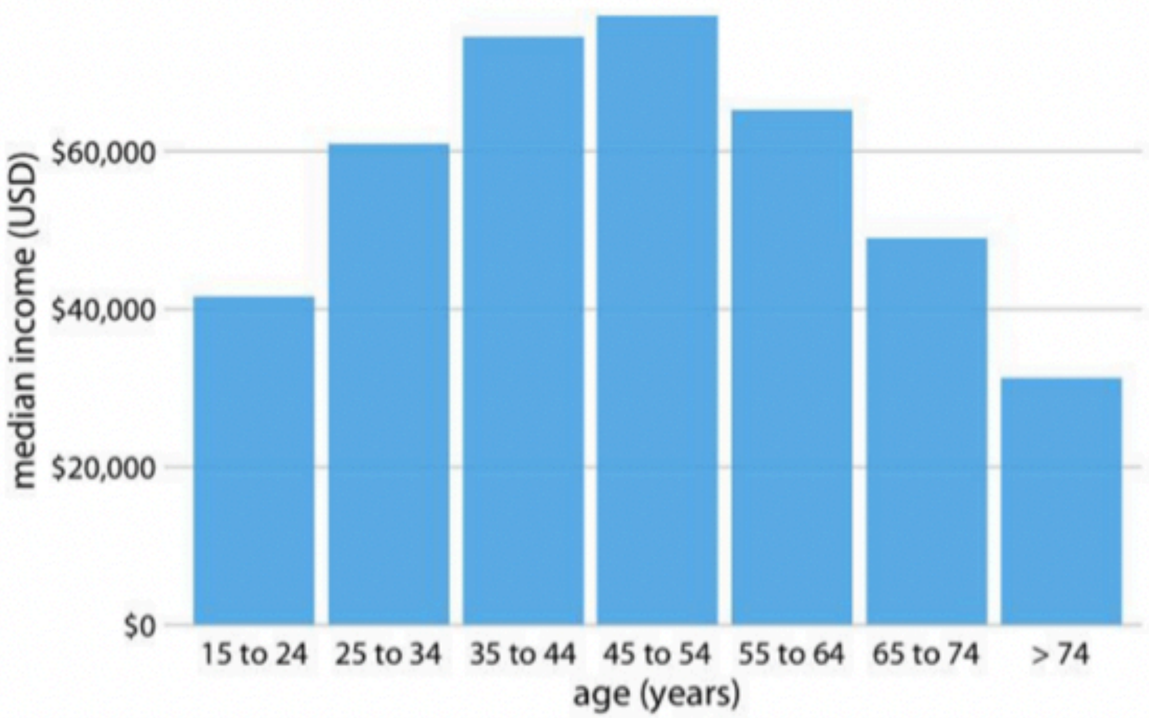
Present the data honestly





Organize the graphic in an intuitive way



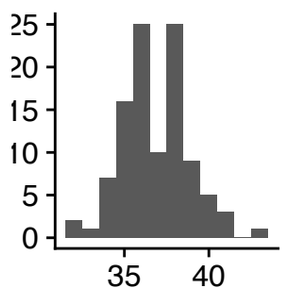


Data visualizations can be generalized as consisting of three parts

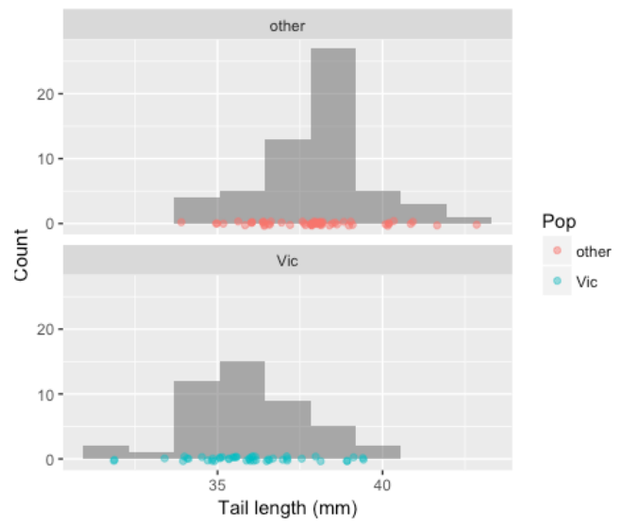
Data + **Geometric Mapping** + **Aesthetic Properties of the Geometric Mapping**

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

+



+

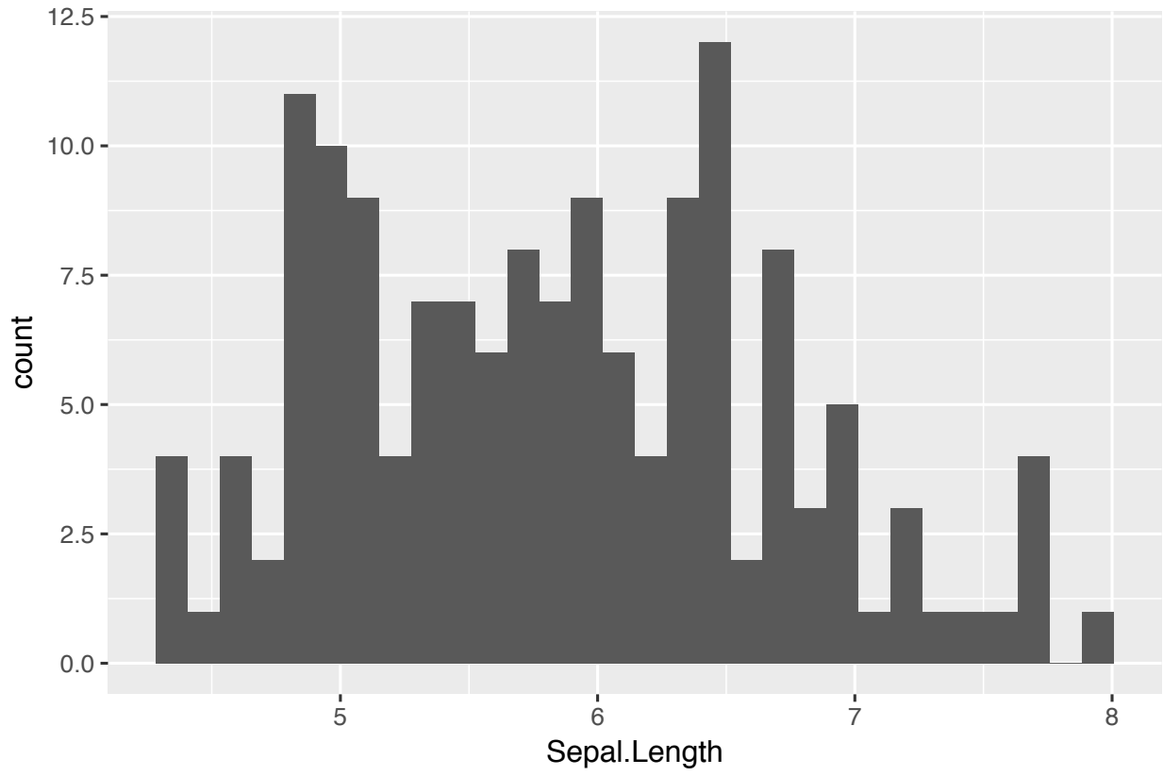



```
ggplot(data = <DATA>) + <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Example 1: Creating a histogram

Data + **Geometric Mapping** + **Aesthetic Properties of the Geometric Mapping**

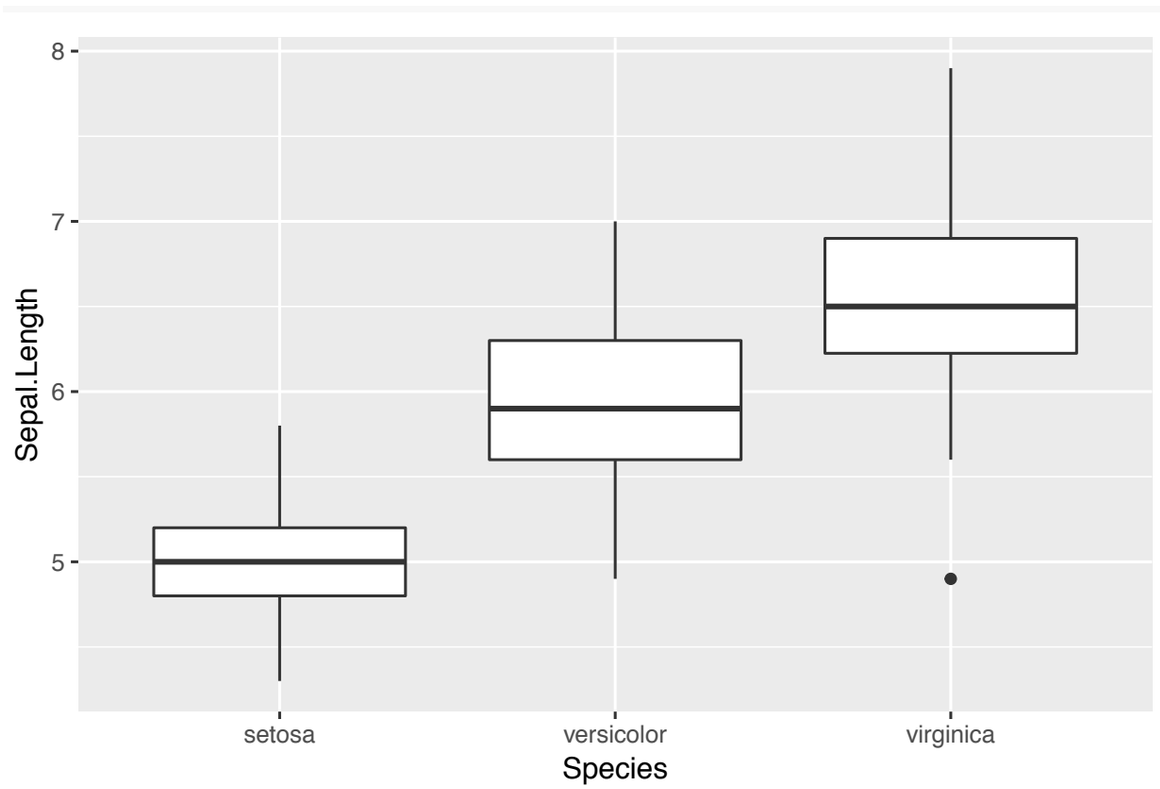
```
ggplot(data = iris) + geom_histogram(mapping = aes(x = Sepal.Length))
```



Example 2: Creating a Boxplot

Data + **Geometric Mapping** + **Aesthetic Properties of the Geometric Mapping**

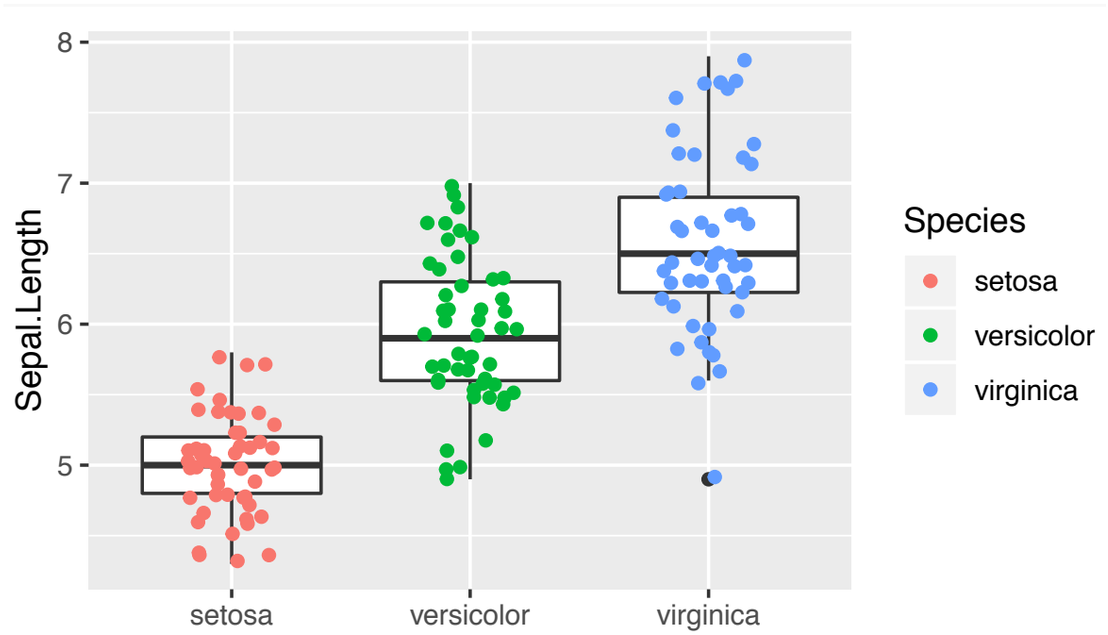
```
ggplot(data = iris) + geom_boxplot(mapping = aes(x = Species,
                                                 y = Sepal.Length))
```



Example 3: Combining Geometric Representations

Data + **Geometric Mapping** + **Aesthetic Properties of the Geometric Mapping**

```
ggplot(data = iris) + geom_boxplot(mapping = aes(x = Species,
                                                 y = Sepal.Length)) +
                      geom_jitter(mapping = aes(x = Species,
                                                y = Sepal.Length,
                                                color = Species),
                                  width = 0.2)
```



Shared aesthetics properties can be specified in `ggplot()` or in an independent `aes()` call

```
ggplot(data = iris, aes(x = Species, y = Sepal.Length)) +
  geom_boxplot() +
  geom_jitter(mapping = aes(color = Species),
              width = 0.2)
```

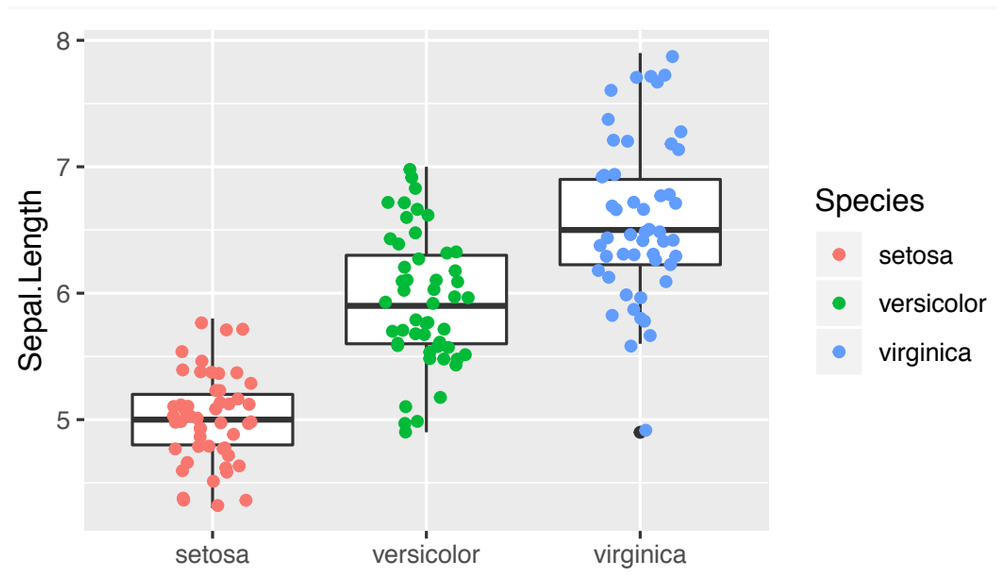
or

```
ggplot(data = iris) +
  aes(x = Species, y = Sepal.Length) +
  geom_boxplot() +
  geom_jitter(mapping = aes(color = Species),
              width = 0.2)
```

Default aesthetic properties

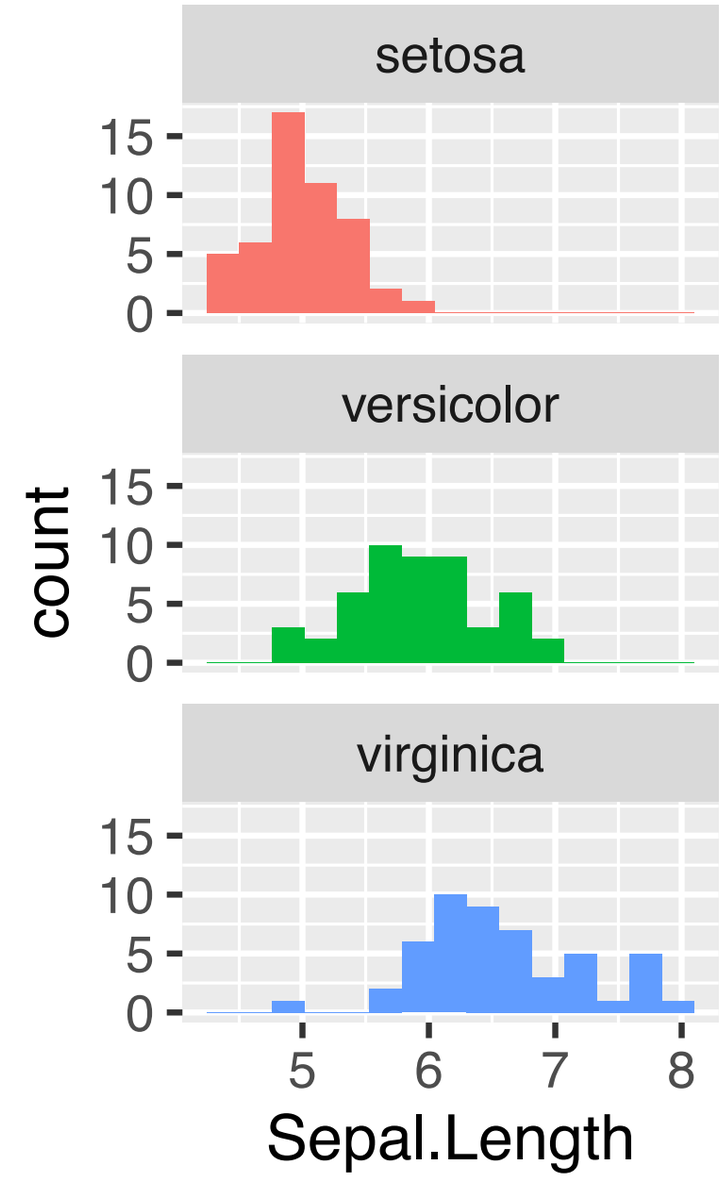
Subsequent geoms inherit the default aesthetics but can specify additional ones

If you don't want to inherit the default aesthetics can specify with argument `inherit.aes = FALSE`



Faceting creates subplots based on conditioning of one or more variables

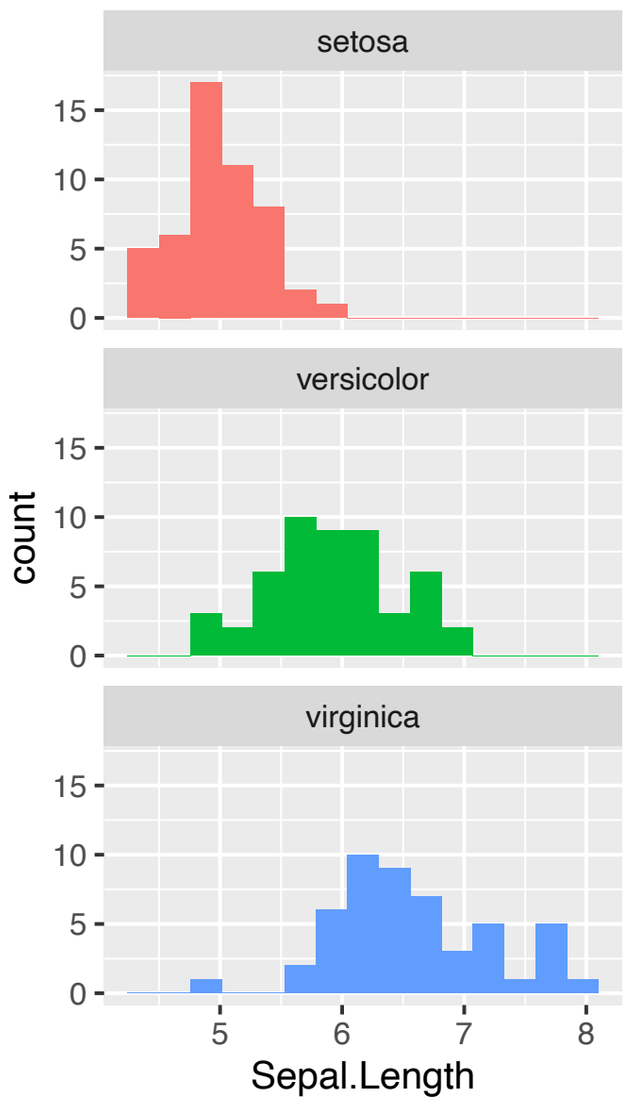
```
ggplot(iris) +
  aes(x = Sepal.Length, fill = Species) +
  geom_histogram(bins = 15) +
  facet_wrap(~Species, ncol = 1)
```



Themes change look-and-feel across the plot

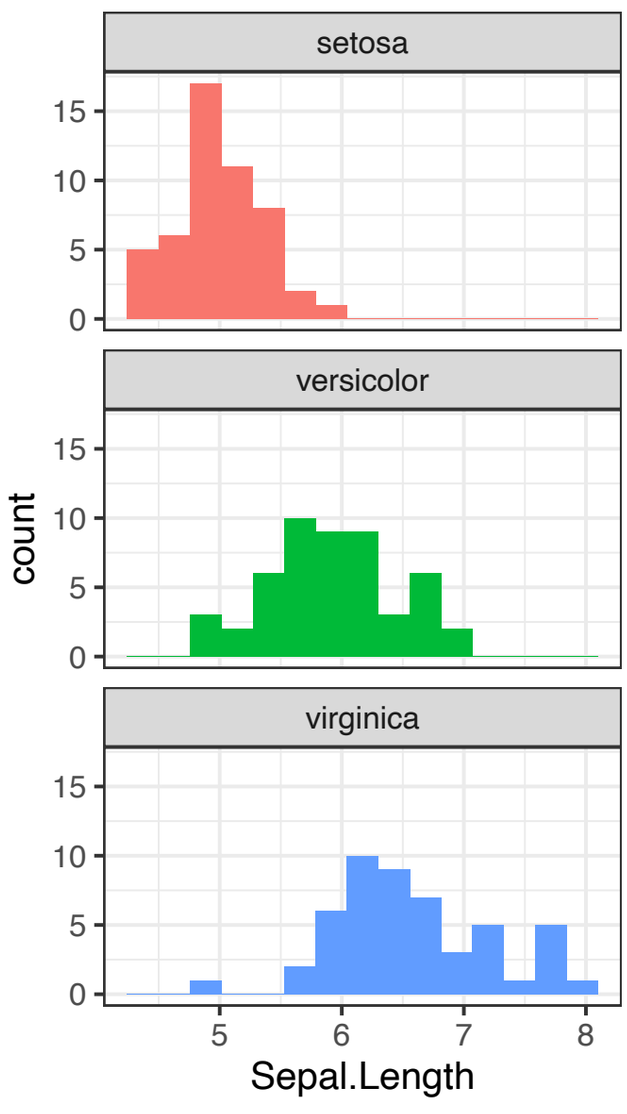
default theme

```
ggplot(data = iris, aes(x = Sepal.Length, fill=Species)) +
  geom_histogram(bins=15) + facet_wrap(~Species,ncol=1) +
  theme(aspect.ratio = 0.5, legend.position = "none")
```



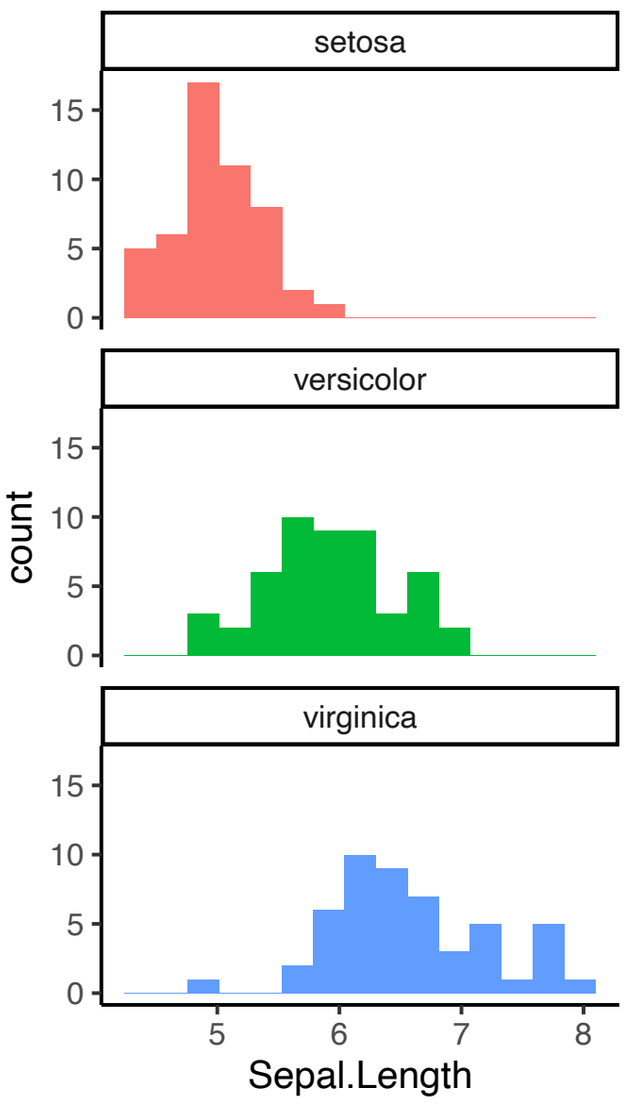
+theme_bw()

```
ggplot(data = iris, aes(x = Sepal.Length, fill=Species)) +
  geom_histogram(bins=15) + facet_wrap(~Species,ncol=1) +
  theme_bw() + theme(aspect.ratio = 0.5, legend.position = "none")
```



+theme_classic()

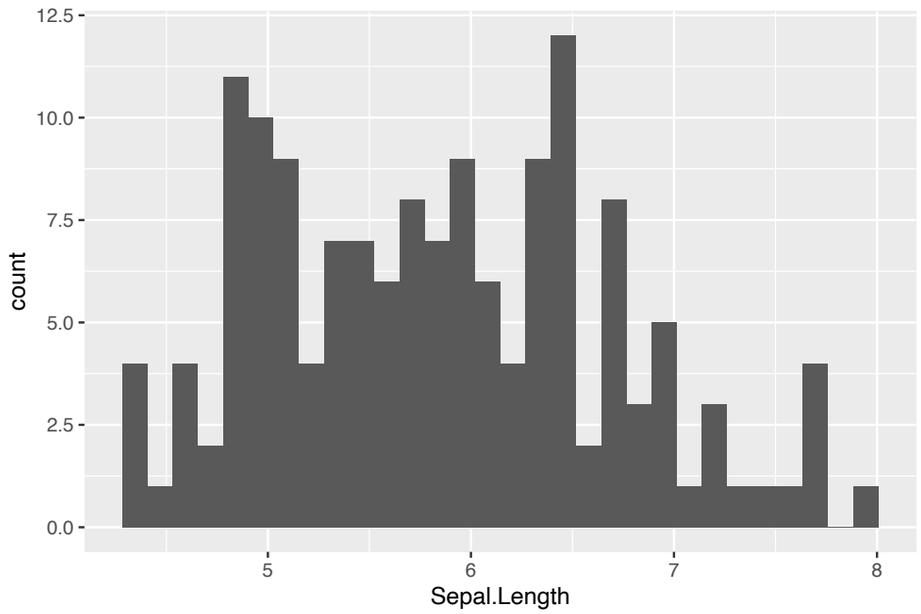
```
ggplot(data = iris, aes(x = Sepal.Length, fill=Species)) +
  geom_histogram(bins=15) + facet_wrap(~Species,ncol=1) +
  theme_classic() + theme(aspect.ratio = 0.5, legend.position = "none")
```



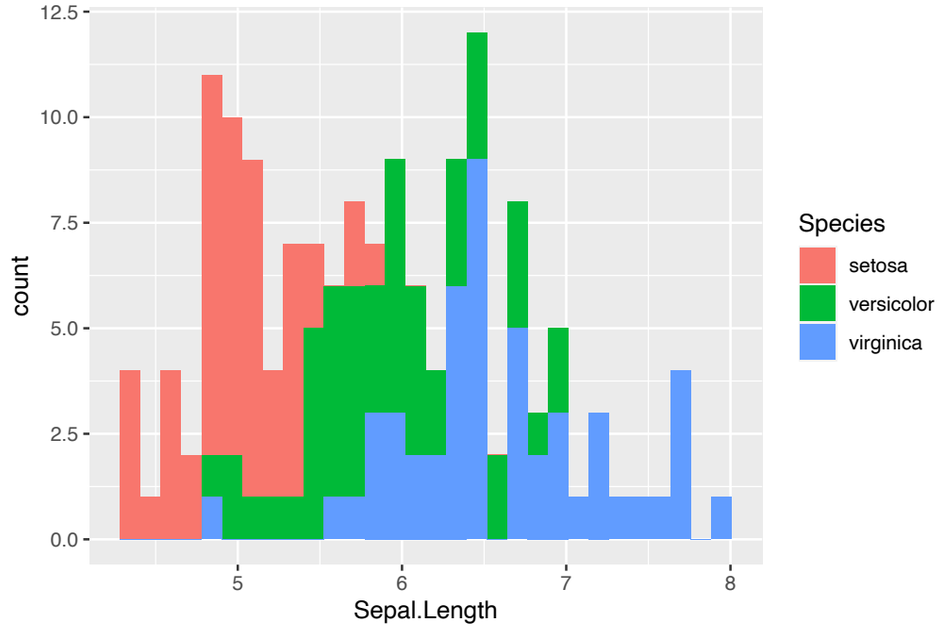
"Geom" tour: Histograms

geom_histogram()

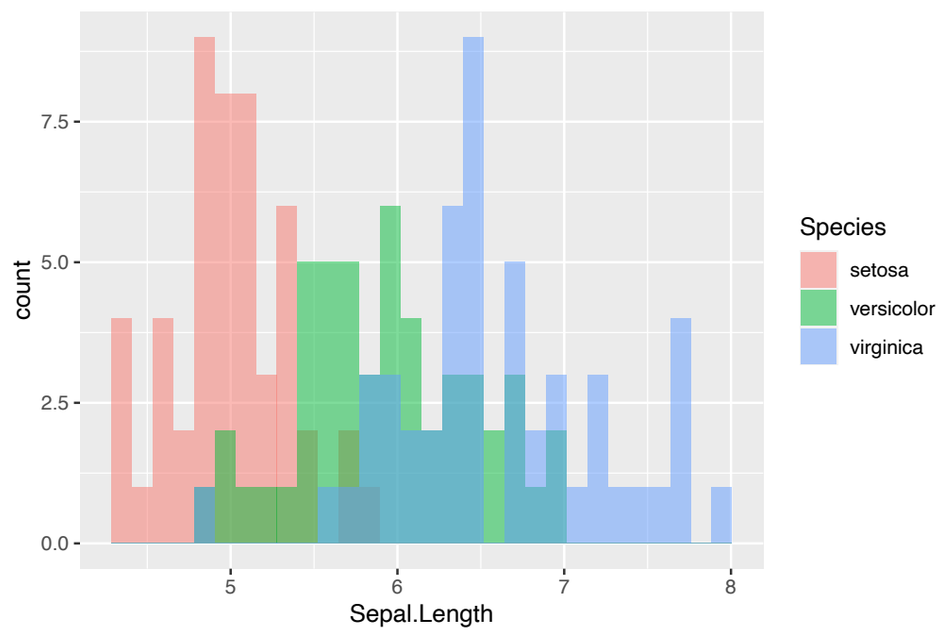
```
ggplot(iris) +
  aes(x = Sepal.Length) +
  geom_histogram()
```




```
ggplot(iris) +
  aes(x = Sepal.Length, fill = Species) +
  geom_histogram()
```



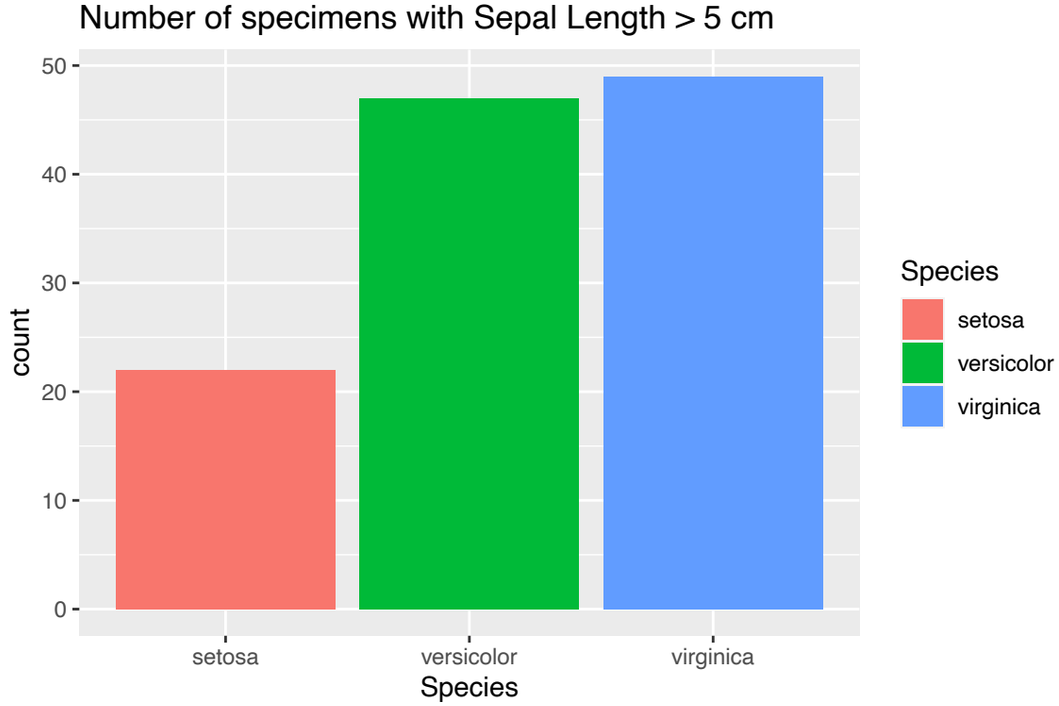

```
ggplot(iris) +
  aes(x = Sepal.Length, fill = Species) +
  geom_histogram(position = "identity", alpha=0.5)
```



"Geom" tour: Bar charts

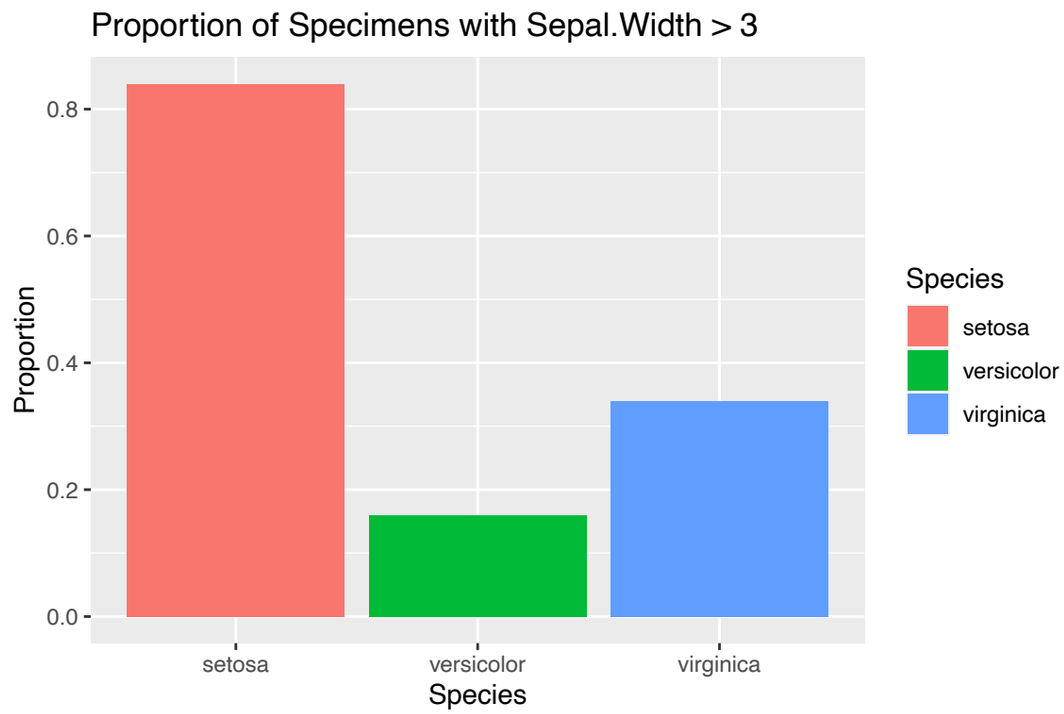
geom_bar()

```
iris |>
  filter(Sepal.Length > 5) |>
  ggplot(aes(x = Species, fill = Species)) +
  geom_bar() +
  labs(title = "Number of specimens with Sepal Length > 5 cm")
```



geom_col()

```
iris |>
  group_by(Species) |>
  summarize(prop_Sepal.Width_gt3 = sum(Sepal.Width > 3)/n()) |>
  ggplot(aes(x = Species, fill = Species,
             y = prop_Sepal.Width_gt3)) +
  geom_col() +
  labs(y = "Proportion",
       title = "Proportion of Specimens with Sepal.Width > 3")
```

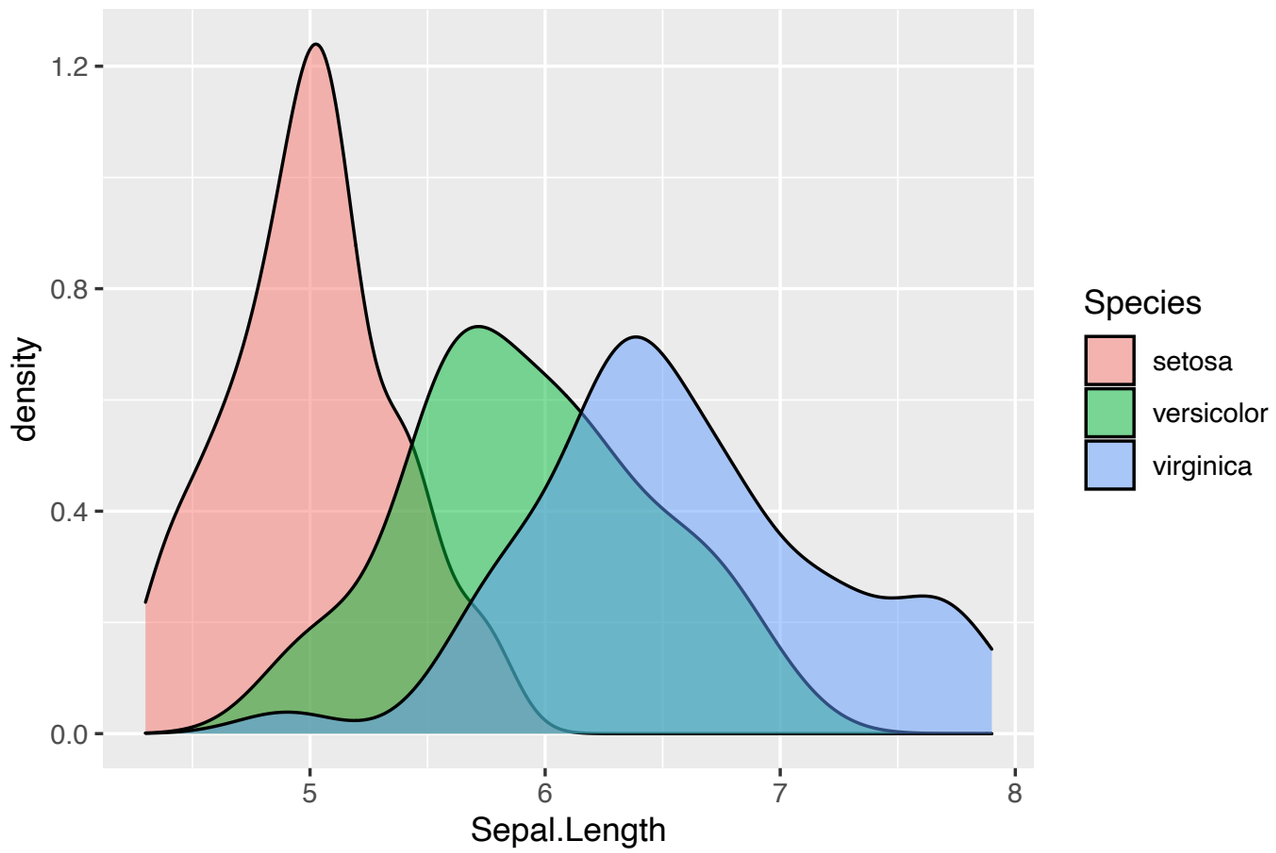


Use `geom_col()` when you've pre-computed the values you want to depict

"Geom" tour: Density and violin plots

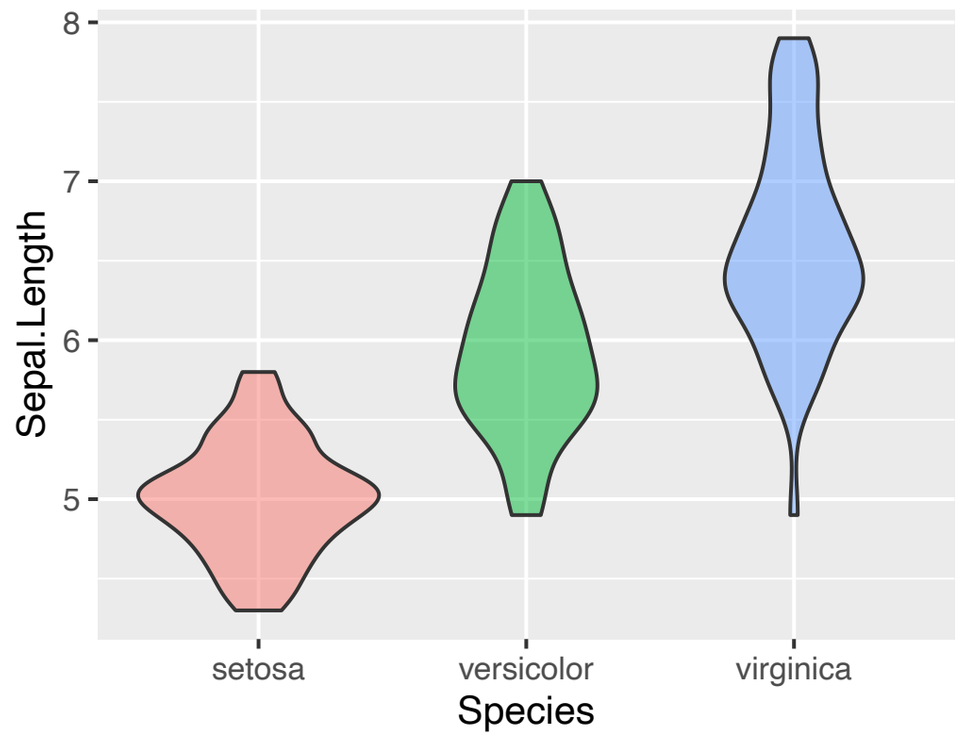
geom_density()

```
ggplot(data = iris, aes(x = Sepal.Length, fill=Species)) +
  geom_density(alpha=0.5)
```



geom_violin()

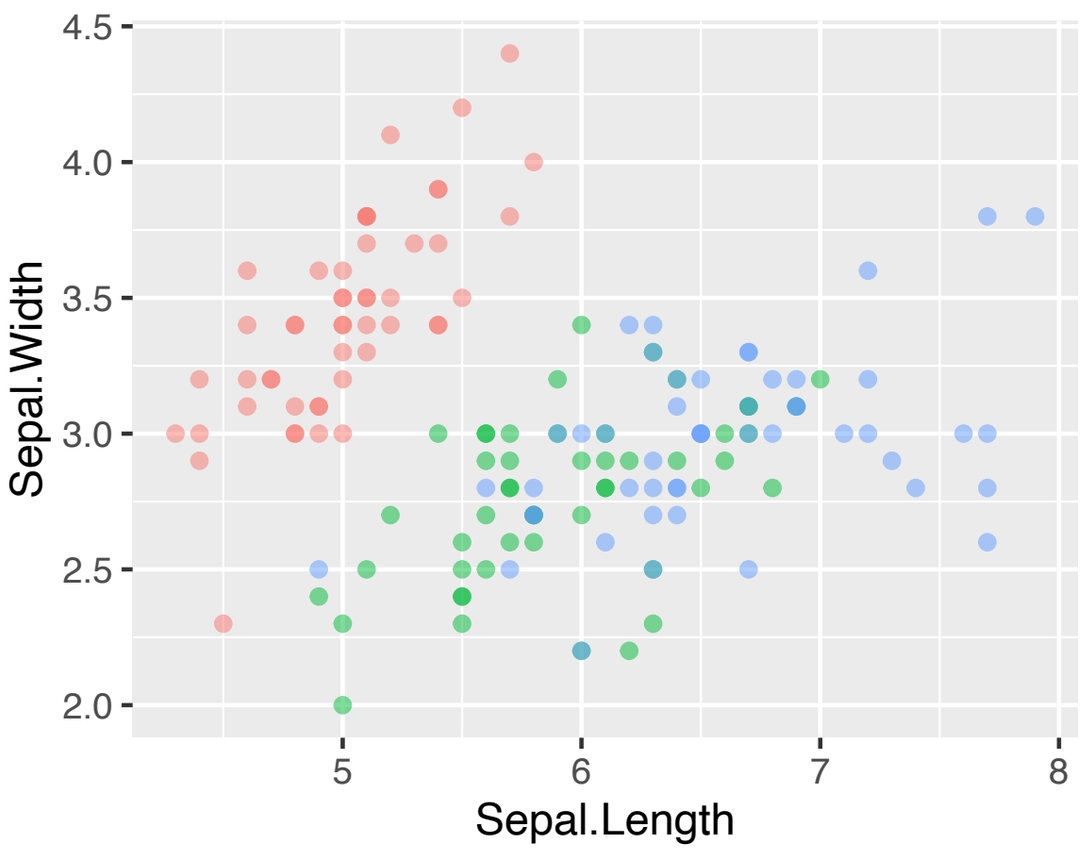
```
ggplot(data = iris, aes(x = Species,
                        y = Sepal.Length,
                        fill=Species)) +
  geom_violin(alpha=0.5)
```



"Geom" tour: Scatter and 2d density

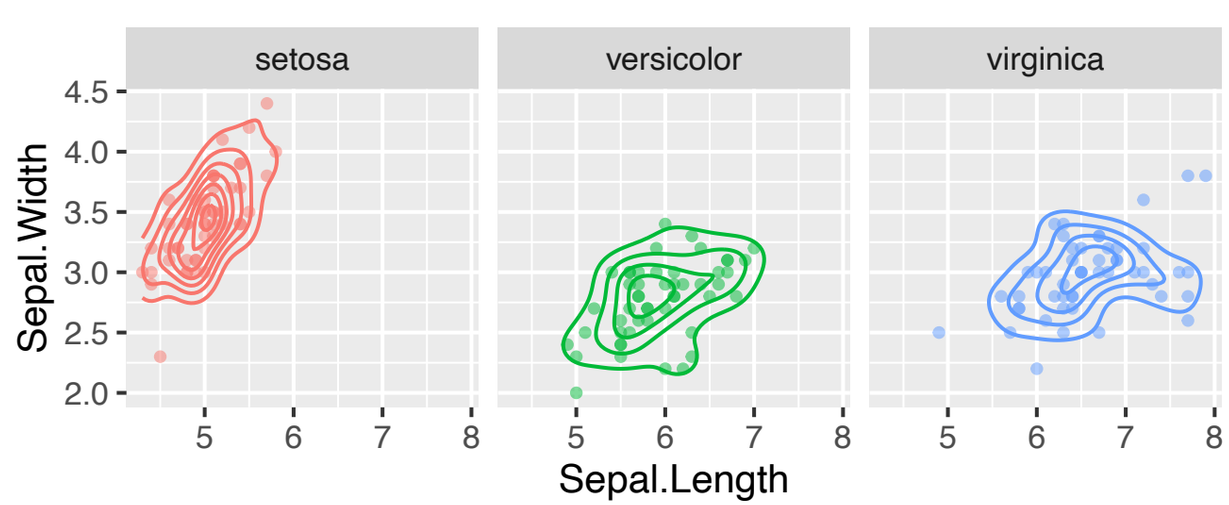
geom_point()

```
ggplot(data = iris, aes(x = Sepal.Length,
                        y = Sepal.Width,
                        color=Species)) +
  geom_point(alpha=0.5)
```



geom_density_2d()

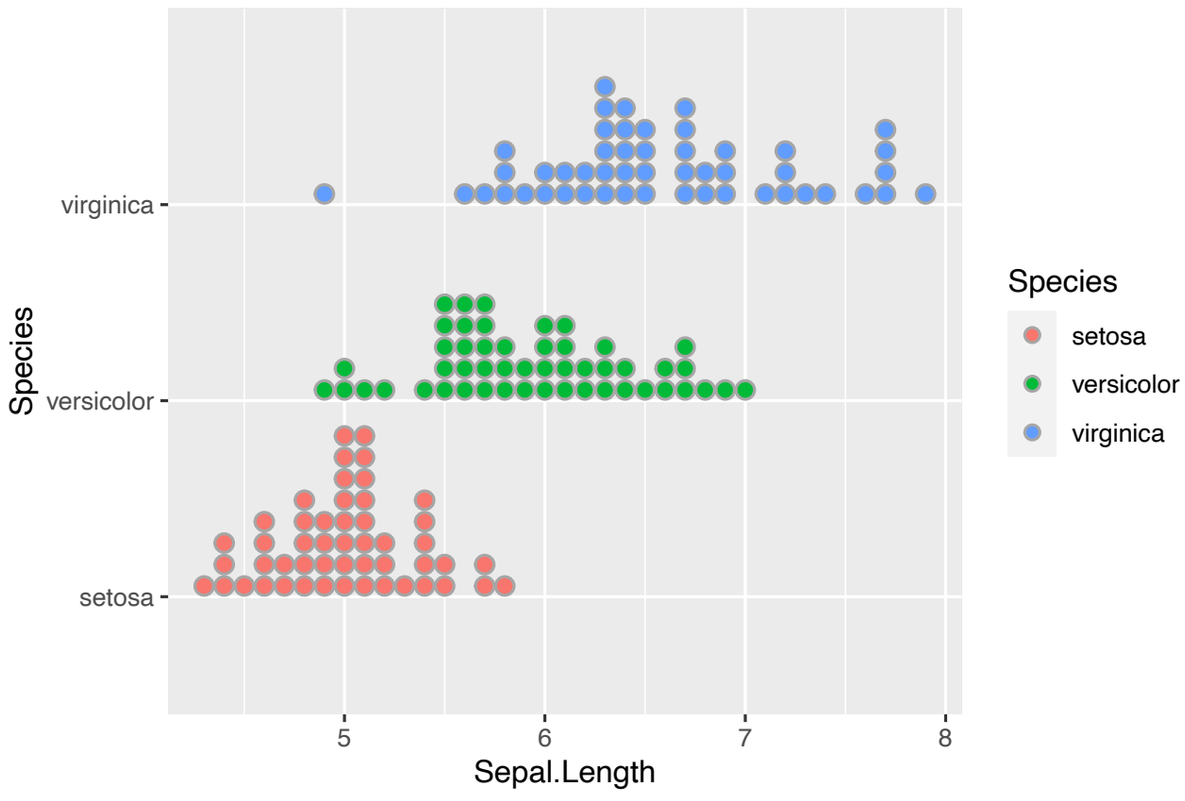
```
ggplot(data = iris, aes(x = Sepal.Length,
                        y = Sepal.Width,
                        color=Species)) +
  geom_density_2d() +
  geom_point(alpha=0.5,size=1) +
  facet_wrap(~Species)
```



ggplot2 extension: `ggdist` for comparing distributions

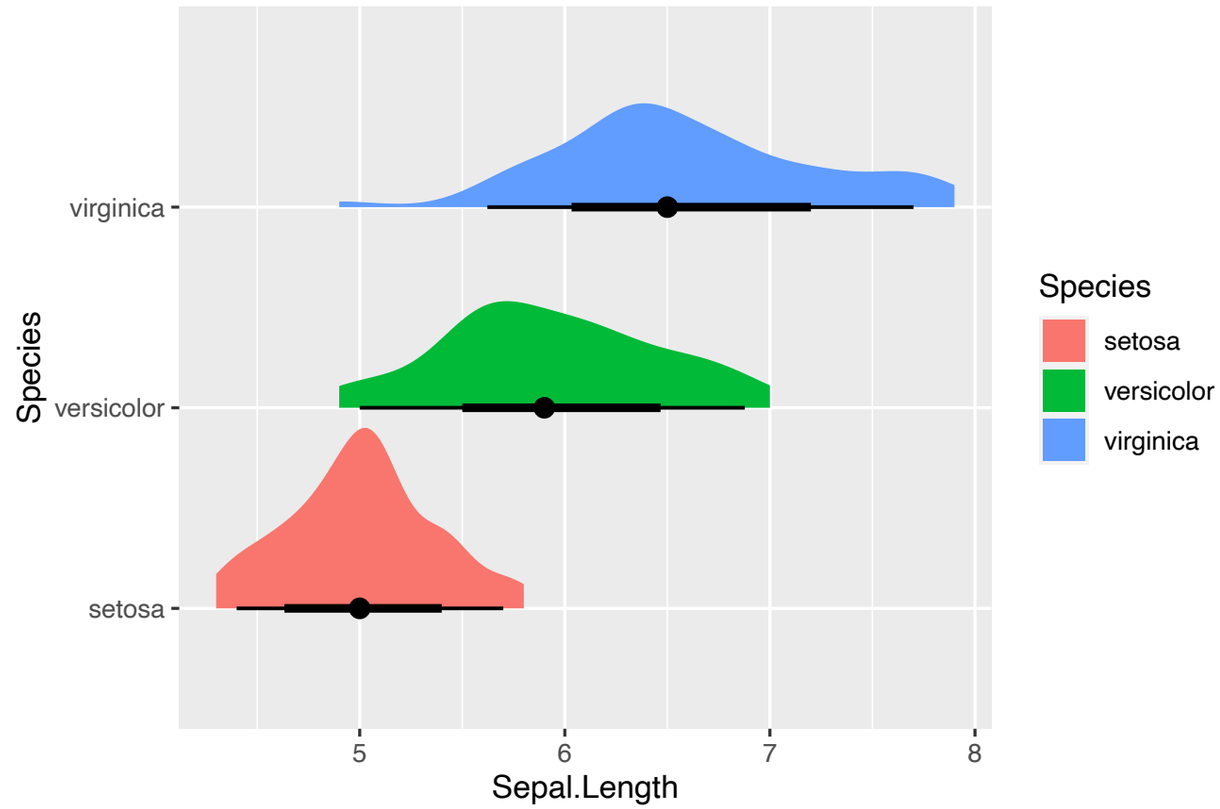
`geom_dots()`

```
library(ggdist)

ggplot(iris, aes(x = Sepal.Length,
                 y = Species,
                 fill=Species)) +
  geom_dots()
```



`stat_slabinterval()`

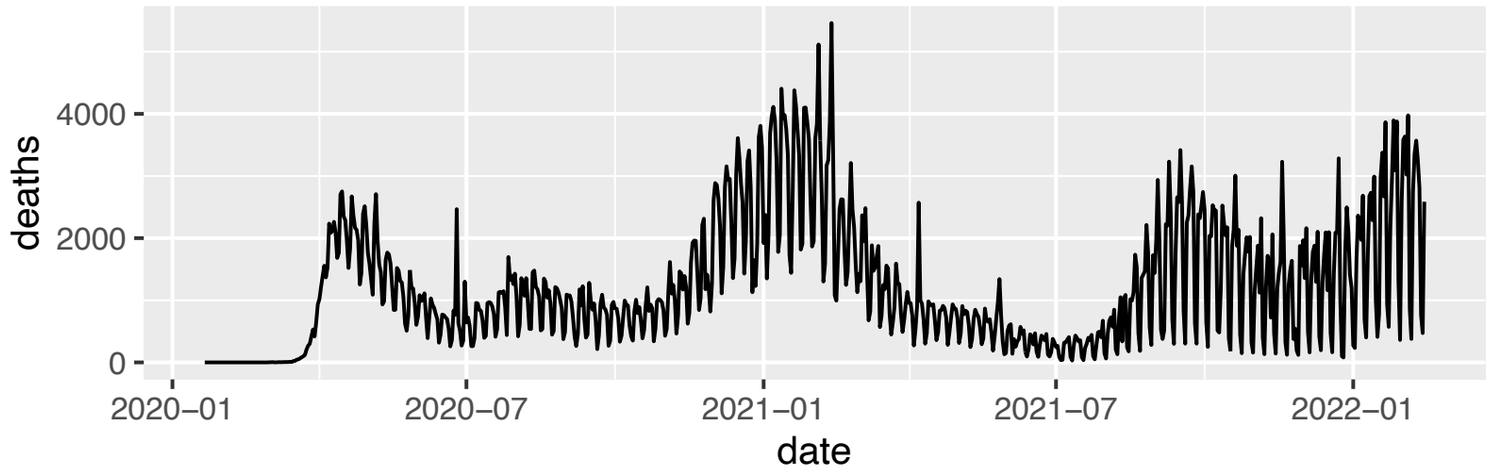
```
ggplot(iris, aes(x = Sepal.Length,
                 y = Species,
                 fill=Species)) +
  stat_slabinterval()
```



"Geom" tour: Line and Area plots

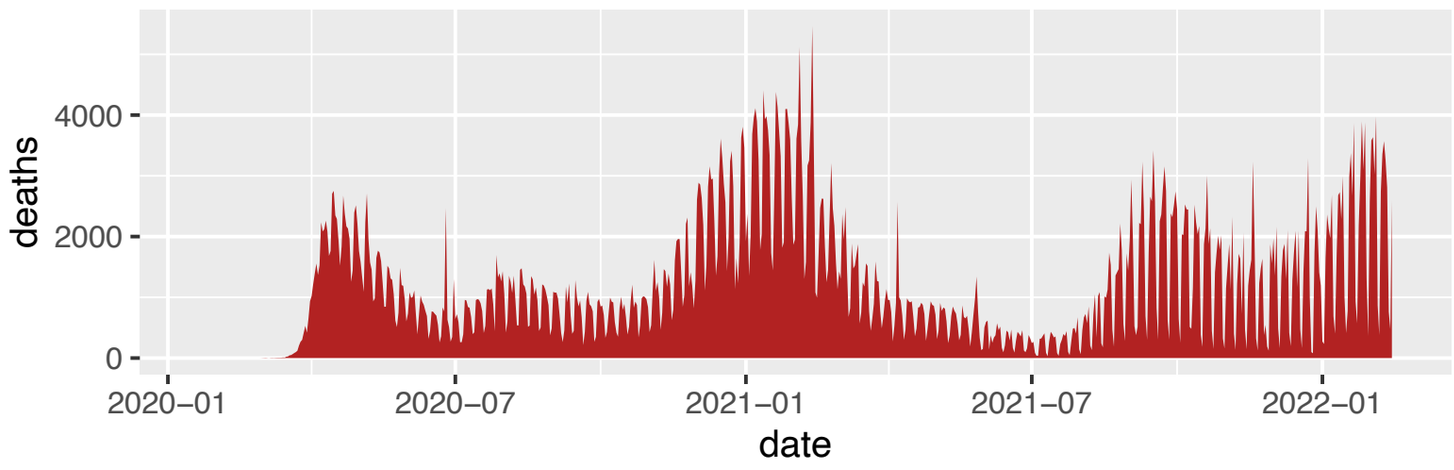
geom_line()

```
ggplot(data = covid, aes(x = date, y = deaths)) +
  geom_line()
```



geom_area()

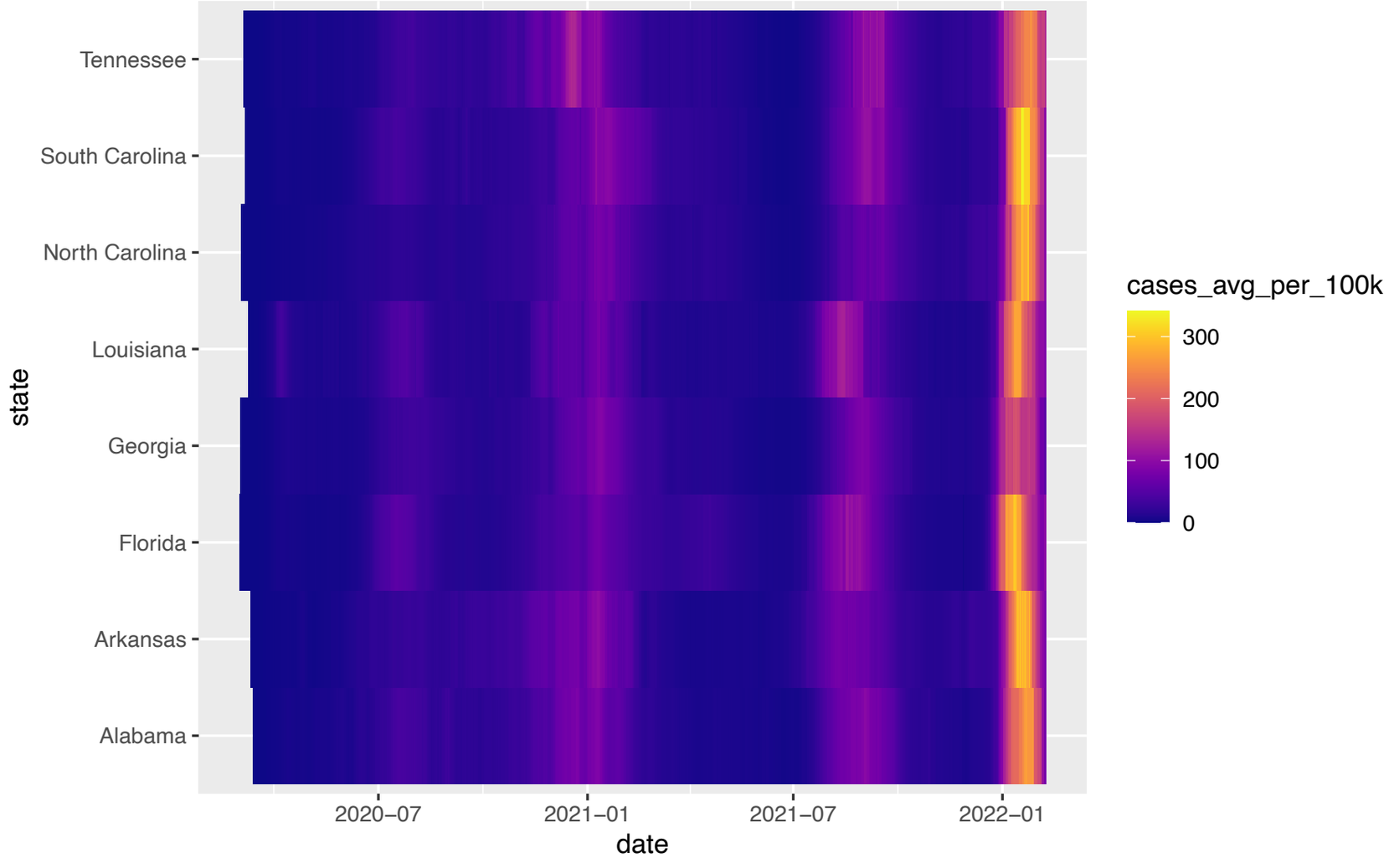
```
ggplot(data = covid, aes(x = date, y = deaths)) +
  geom_area(fill="firebrick")
```



"Geom" tour: Heat maps

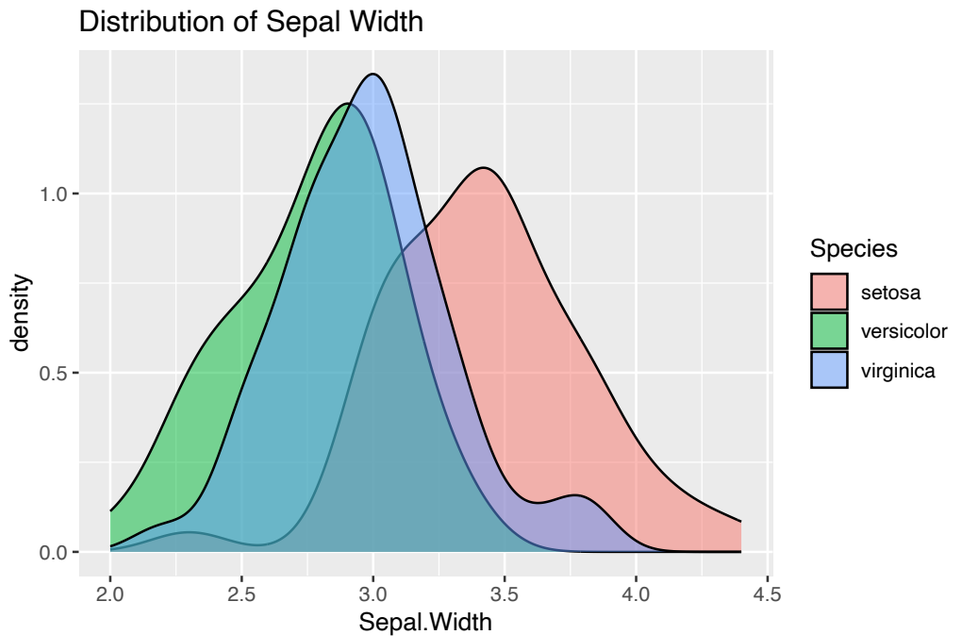
geom_tile() or geom_raster()

```
SE_states <- c("North Carolina","South Carolina",
               "Arkansas","Georgia","Tennessee",
               "Louisiana","Alabama","Florida")

us_states %>%
  filter(state %in% SE_states) %>%
  arrange(state) %>%
  ggplot(aes(x=date, y=state, fill=cases_avg_per_100k)) +
    geom_tile() +
    scale_fill_viridis(option = "C")
```



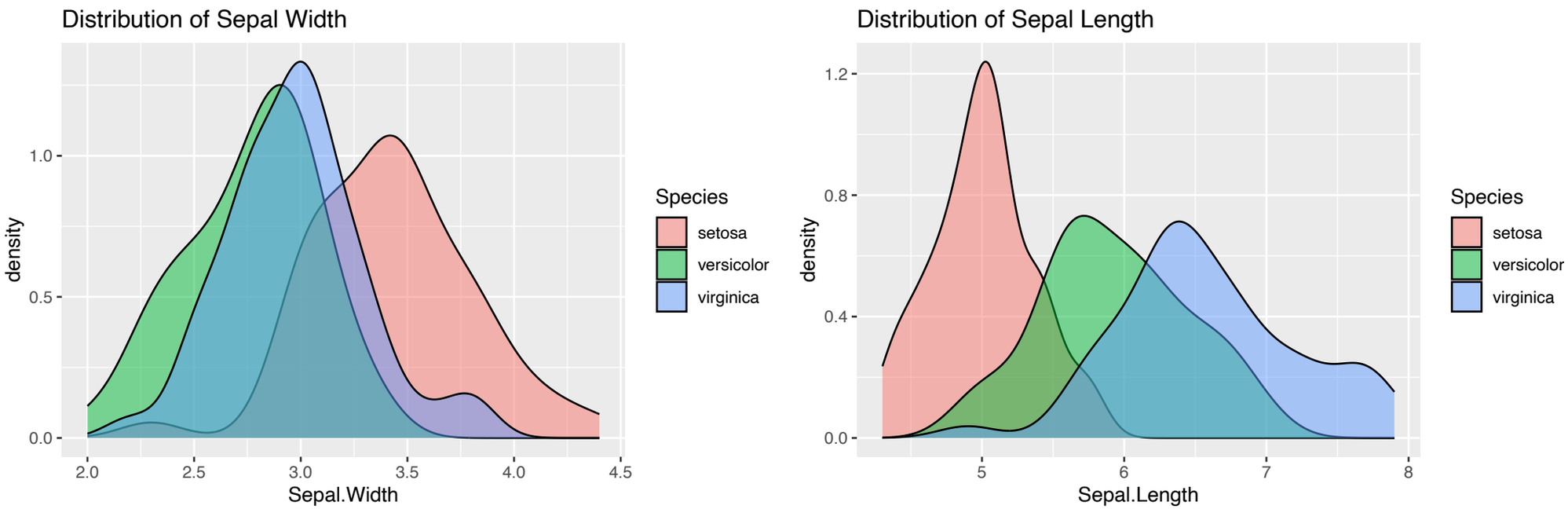
Assigning plots to variables

```
plot_Sepal.Length <-
  ggplot(data = iris, aes(x = Sepal.Length, fill=Species)) +
  geom_density(alpha=0.5) +
  labs(title = "Distribution of Sepal Length")

plot_Sepal.Width <-
  ggplot(data = iris, aes(x = Sepal.Width, fill=Species)) +
  geom_density(alpha=0.5) +
  labs(title = "Distribution of Sepal Width")

plot_Sepal.Ratio <-
  ggplot(data = iris, aes(x = Sepal.Width/Sepal.Length, fill=Species)) +
  geom_density(alpha=0.5) +
  labs(x = "Sepal Ratio (Width / Length)",
       title = "Distribution of Sepal Ratios")

# Note that a plot is not constructed until
# we evaluate one of the variables holding a plot
plot_Sepal.Width
```



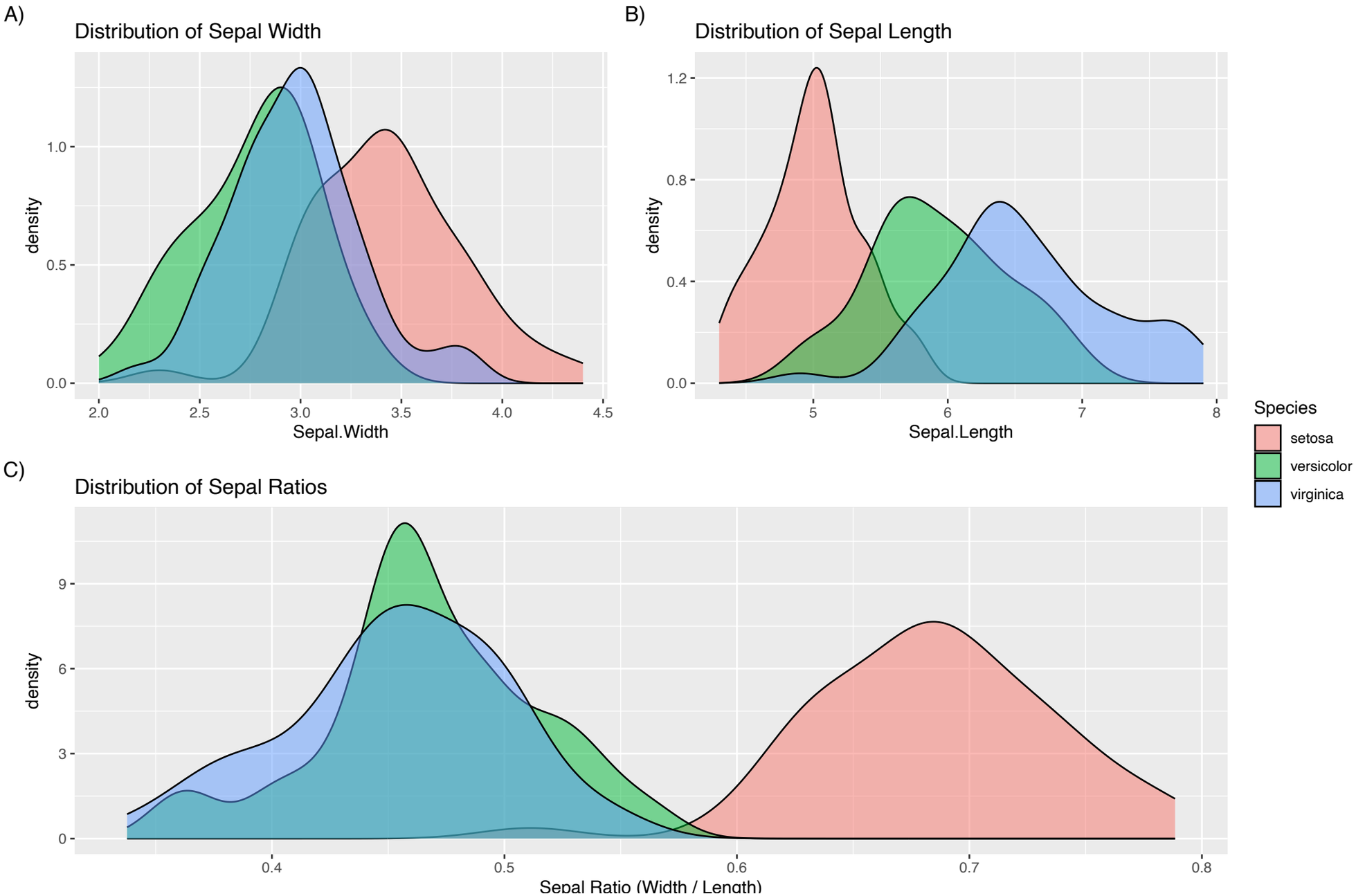
Multi-plot layouts with the `patchwork` library

```
library(patchwork)

# Plots can be laid out side by side using the + operator
plot_Sepal.Width + plot_Sepal.Length
```



Even more complex layouts with patchwork

```
(plot_Sepal.Width | plot_Sepal.Length) / plot_Sepal.Ratio +
  plot_layout(guides = "collect") +
  plot_annotation(tag_levels = 'A', tag_suffix = ")")
```



ggtext allows for simple HTML and markdown formatting of text and labels

```
library(ggtext)

# a named vector provides an explicit map
# between names as given in the data frame
# and the labels we want printed
species_names = c("setosa" = "<i>I. setosa</i>",
                  "versicolor" = "<i>I. versicolor</i>",
                  "virginica" = "<i>I. virginica</i>")

ggplot(iris, aes(x = Sepal.Length, fill=Species)) +
  geom_density(alpha=0.5) +
  labs(title = "Distribution of Sepal Length in three *Iris* species",
       x = "Sepal Length", y = "Density") +
  scale_fill_discrete(labels = species_names) +
  theme(plot.title = element_markdown(),
        legend.text = element_markdown())
```