Enter the Tidyverse

BIO5312 FALL2017

STEPHANIE J. SPIELMAN, PHD

What is the “tidyverse”?

A collection of R packages largely developed by Hadley Wickham and others at Rstudio

Have emerged as staples of modern-day data science in the past 5—10 years

We will focus on:

- Visualization/plotting with `ggplot2`
- Data management and ”wrangling” with `dplyr` and `tidyr`
- Document presentation with `RMarkdown`

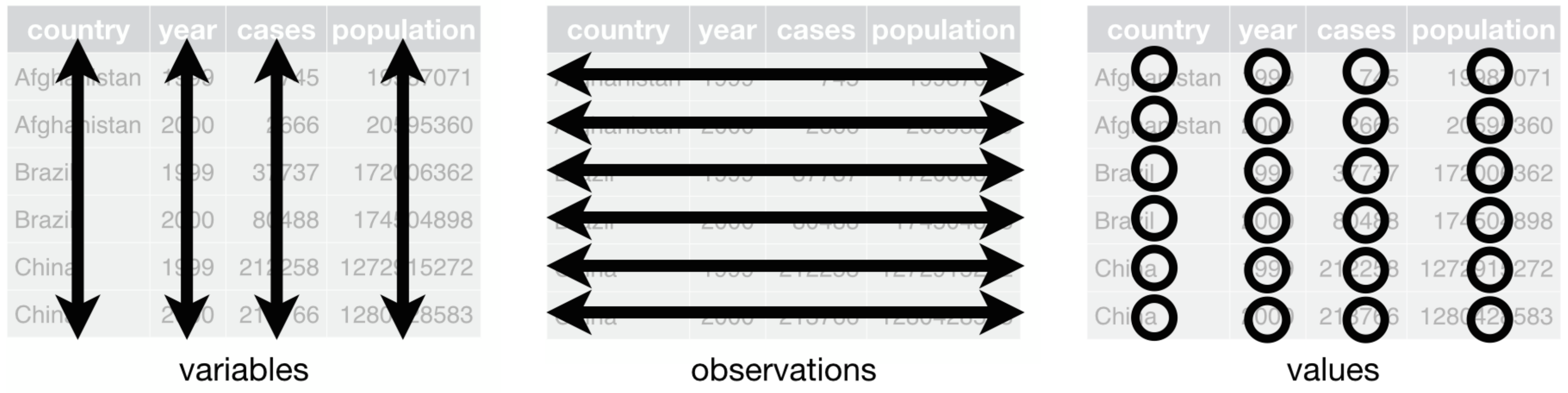
Focus is on **tidy dataframes**

Each variable forms a column.

Each observation forms a row.

Each type of observational unit forms a table.

Tidy data provides a consistent approach to data management that greatly facilitates downstream analysis and viz

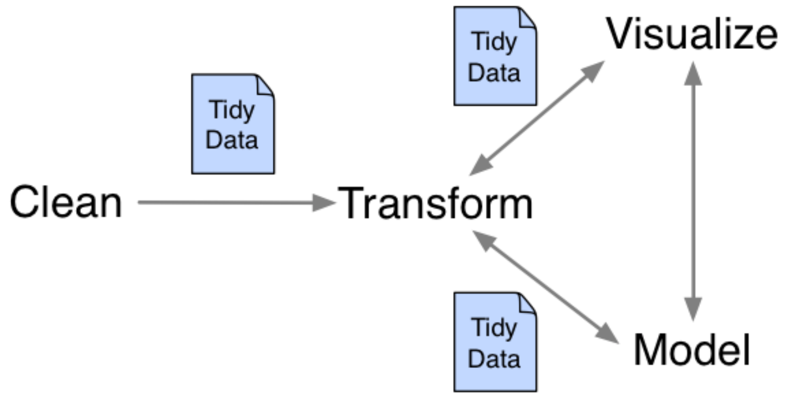


Working with tidy data

The package `dplyr` can manipulate and manage tidy data

The package `tidyr` can rearrange data to convert to/from tidy data

The package `ggplot2` is used for visualization/plotting



The fundamental verbs of `dplyr`

`filter()`	select rows
`select()`	select columns
`mutate()`	create new columns
`group_by()`	establish a data grouping
`tally()`	count observations in a grouping
`summarize()`	calculate summary statistic
`arrange()`	arrange rows

There are more functions but these ones are key!

The pipe operator %>%

"Pipes" output from one function/operation as input to the next

```
## Start simple: display data
head(iris)

## Using %>%
iris %>% head()
```

```
## Find the mean of iris sepal lengths
mean.sepal <- mean(iris$Sepal.Length)

## Using %>%
mean.sepal <- iris$Sepal.Length %>% mean()

iris$Sepal.Length %>% mean() -> mean.sepal

iris %>% mean(Sepal.Length)  -> mean.sepal
```

"forward assignment" operator follows the logical flow of piping

`dplyr` demo

Commands in demo are on sjspielman.org/bio5312_fall2017/day2_tidyverse1

Visualizing with `ggplot2`

The package `ggplot2` is a graphics package that implements a **g**rammar of **g**raphics

- Operates on *data frames*, not vectors like Base R
- Explicitly differentiates between the data and the representation of the data

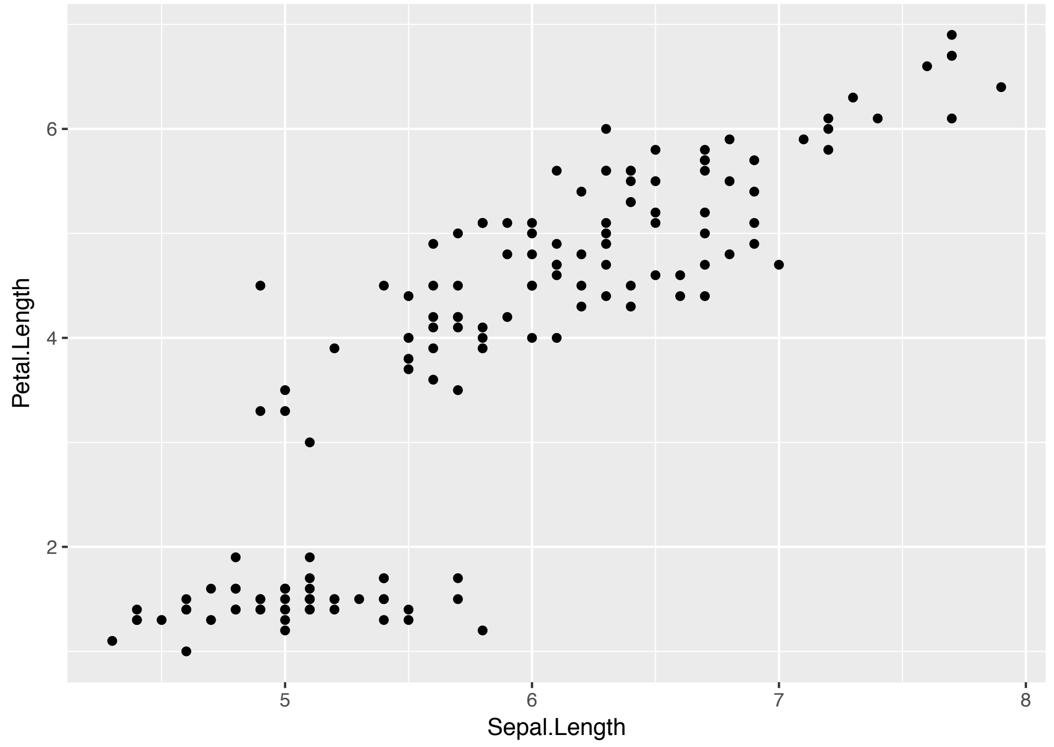
The `ggplot2` grammar

Grammar element*	What is it
Data	The data frame being plotted
Geometrics	The geometric shape that will represent the data • Point, boxplot, histogram, violin, bar, etc.
Aesthetics	The aesthetics of the geometric object • Color, size, shape, etc.

*Table is tiny subset of what ggplot2 has to offer

Example: scatterplot

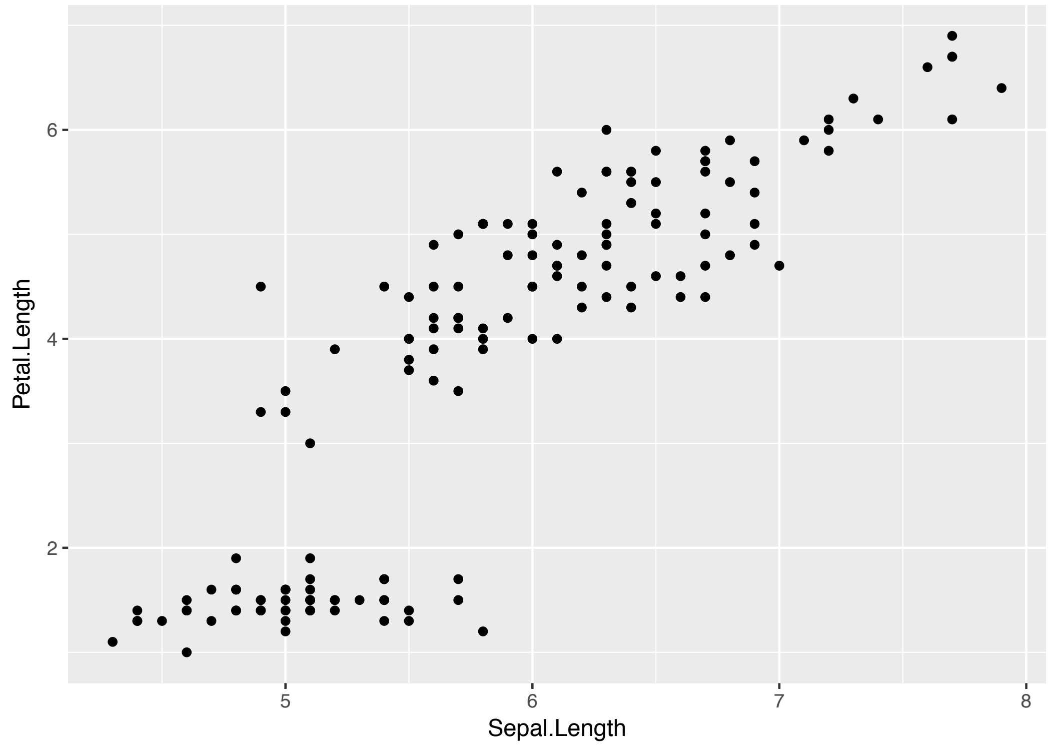
```
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) + geom_point()
```



Example: scatterplot

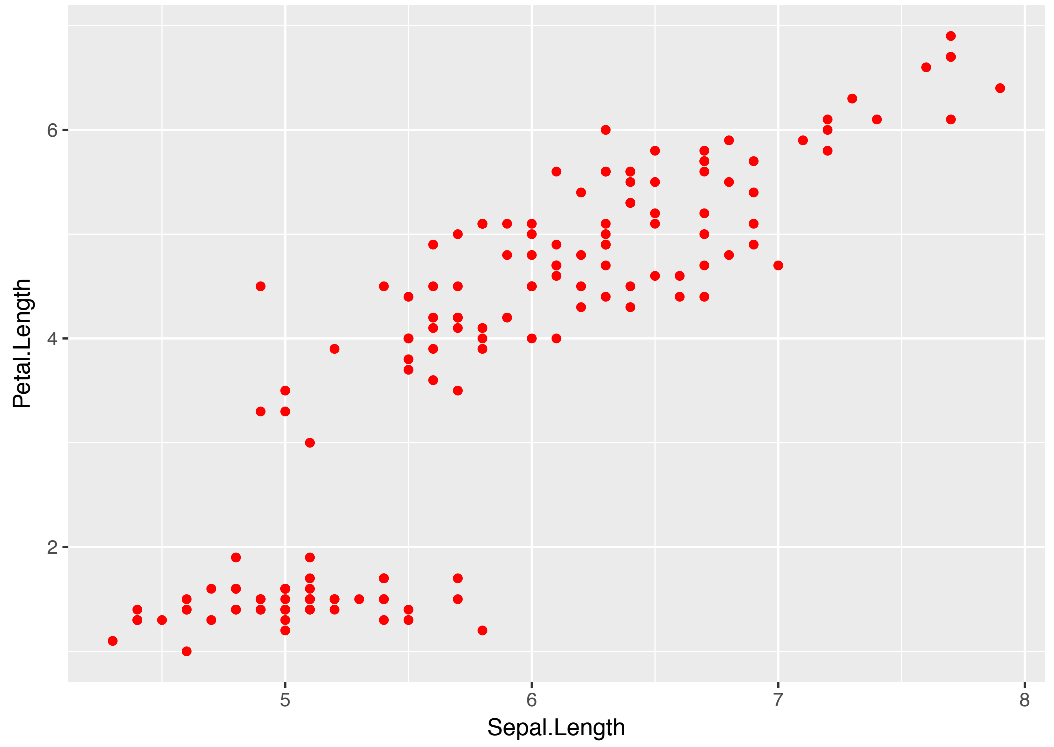
```
> ggplot( iris, aes(x = Sepal.Length, y = Petal.Length) ) + geom_point()
```

- Pass in the data frame as your first argument
- Aesthetics map the data onto plot characteristics, here x and y axes
- Display the data geometrically as points



Example: scatterplot with color

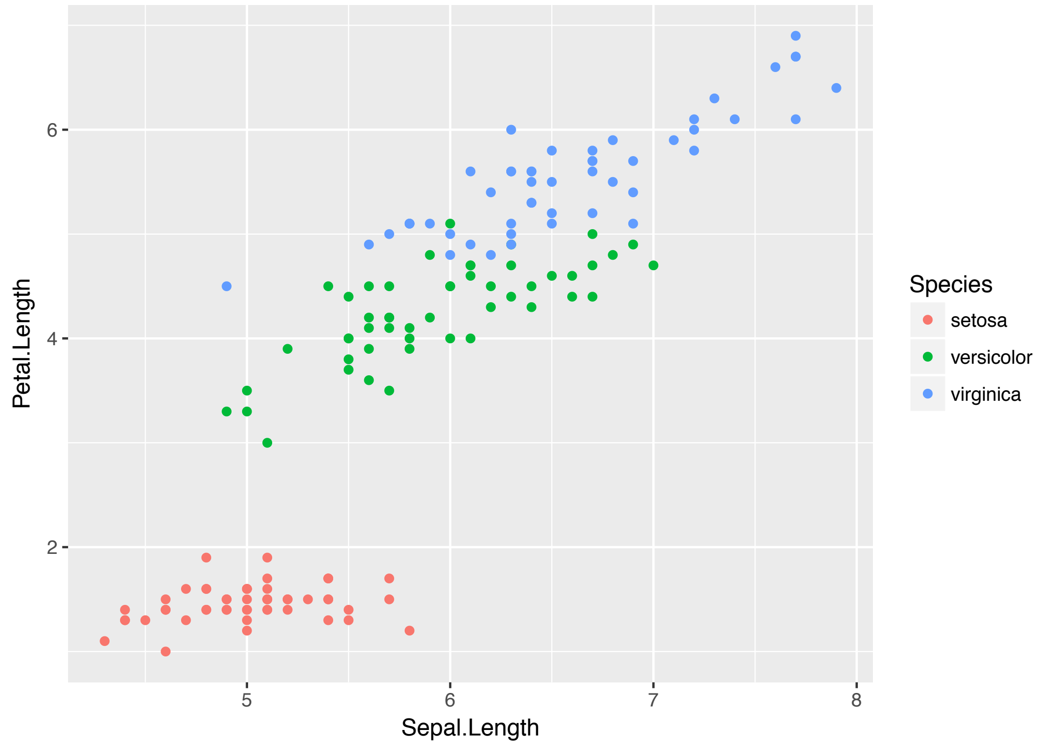
```
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) + geom_point(color = "red" )
```



Example: scatterplot with **aes** color

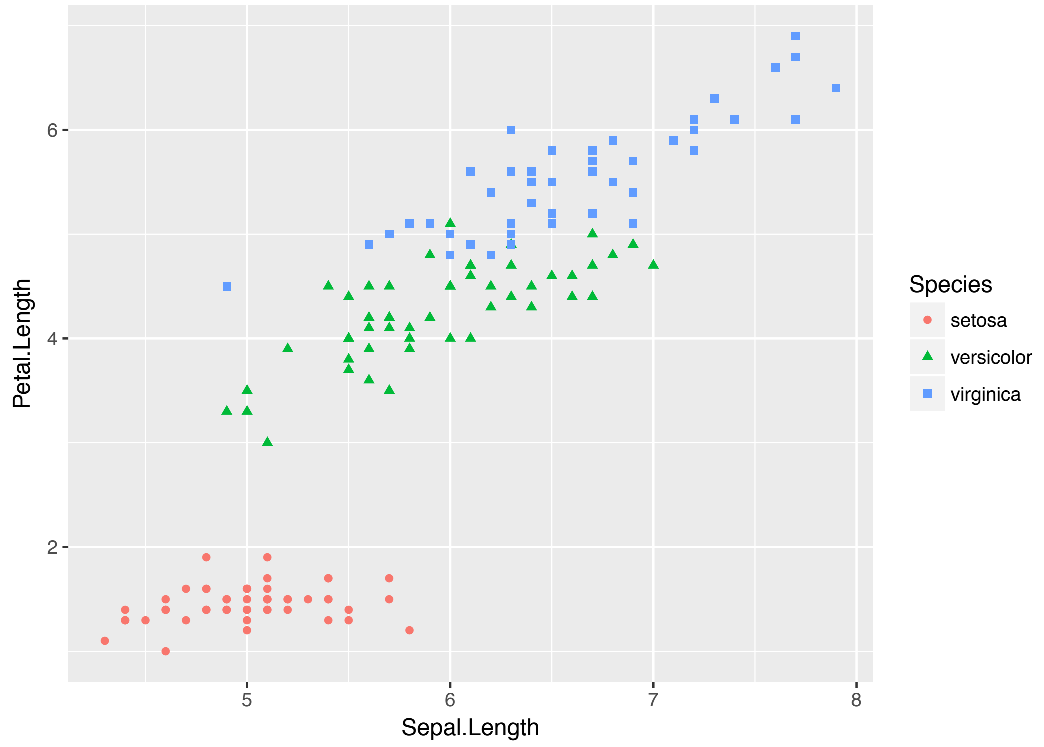
```
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) + geom_point()
```

- **Placing color inside aesethetic maps it to the data.**



Example: scatterplot with **aes** color, shape

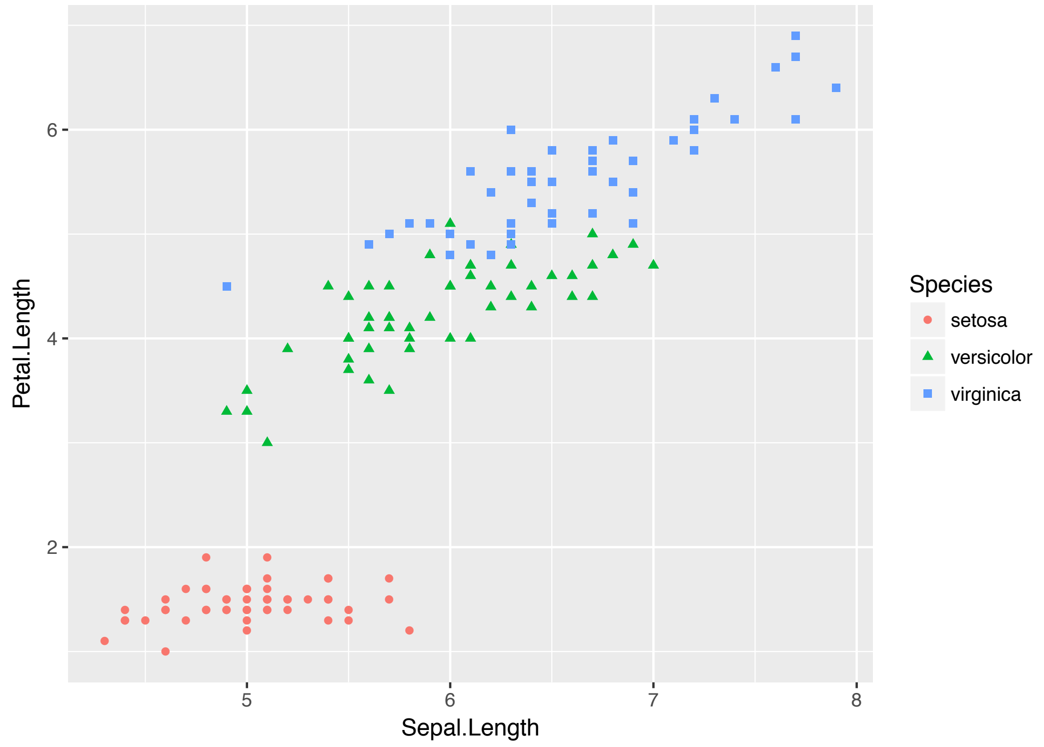
```
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species, shape = Species))
+ geom_point()
```



Aesthetics may be placed inside the relevant geom

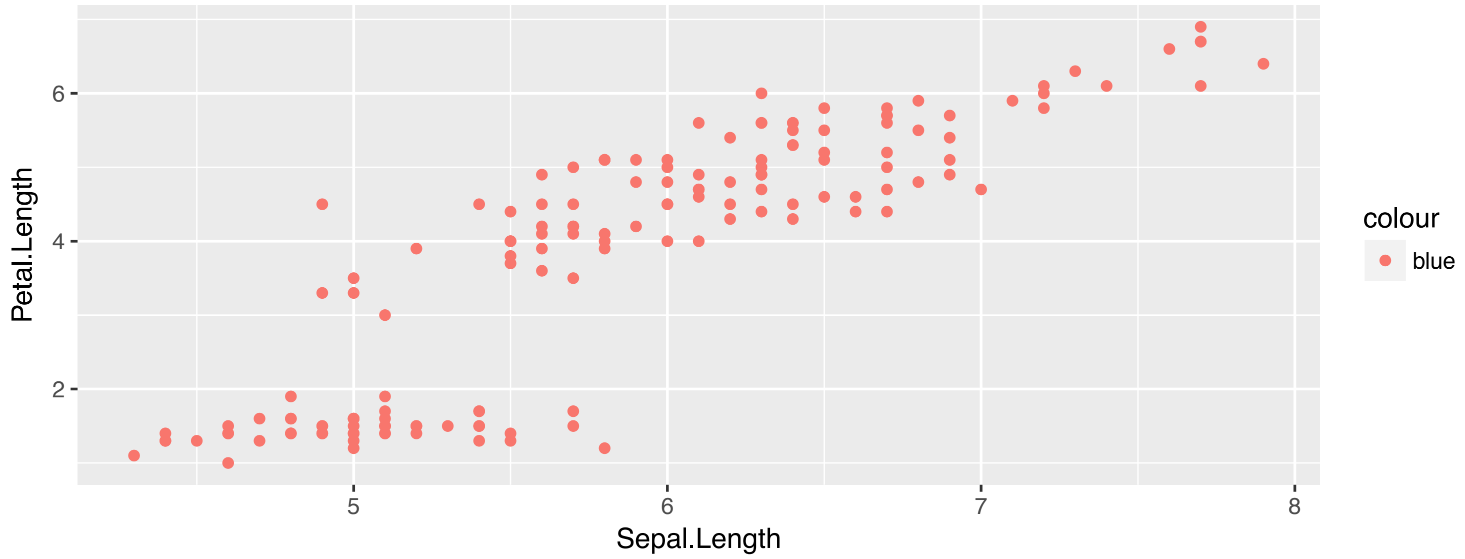
```
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) + geom_point(aes(color =
  Species, shape = Species))
```

```
> ## Remember dplyr!
> iris %>% ggplot(aes(x = Sepal.Length, y =
  Petal.Length)) + geom_point(aes(color =
  Species, shape = Species))
```



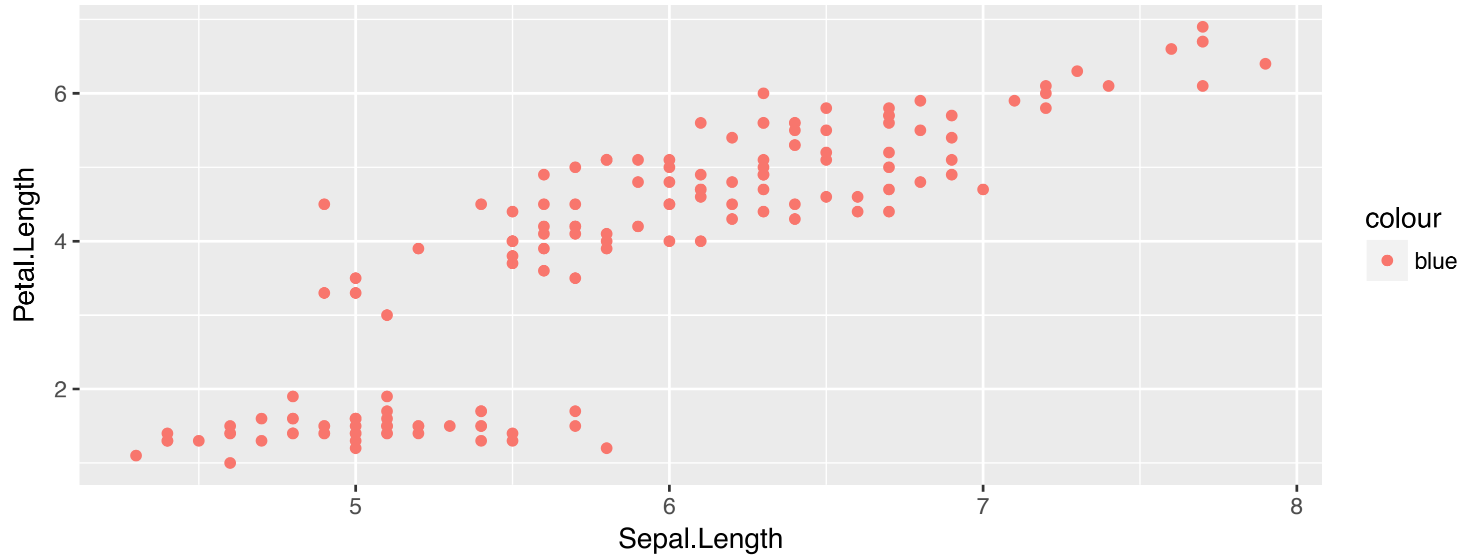
Aesthetics are for mapping only

```
> ### Color all points blue?
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = "blue")) + geom_point()
```



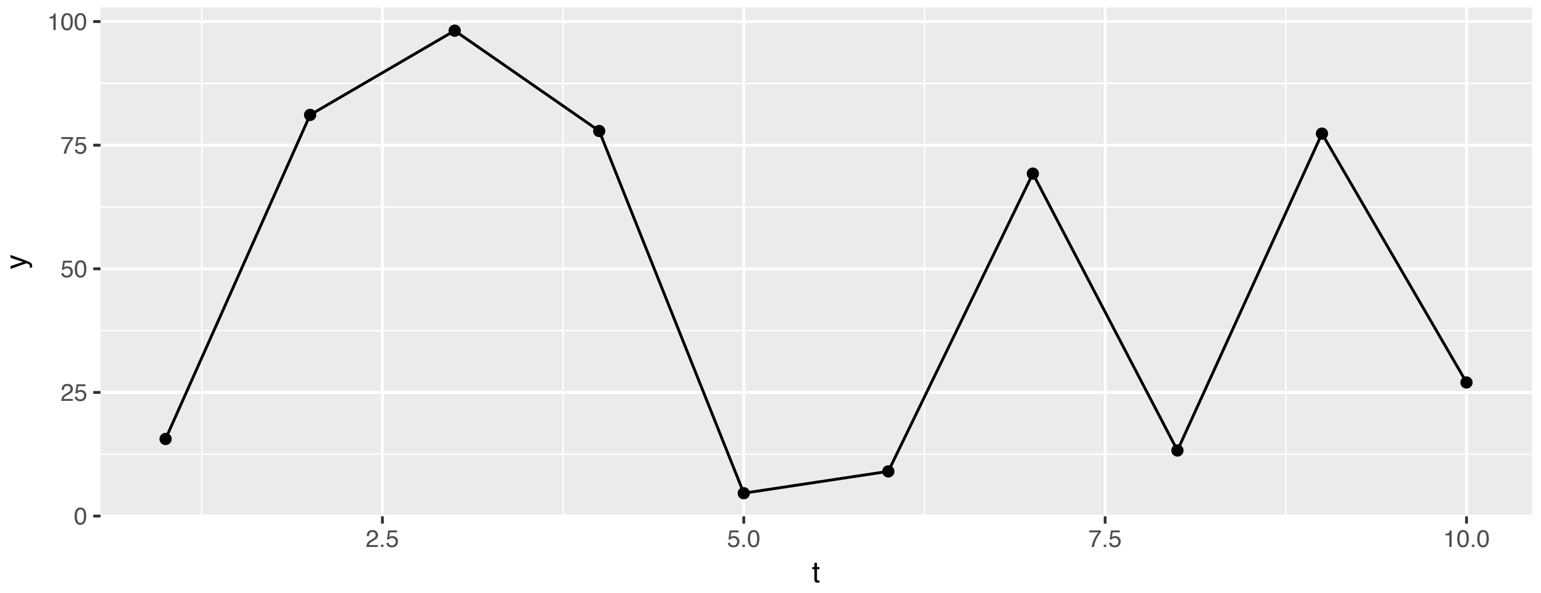
Aesthetics are for mapping only

```
~~> ### Color all points blue?~~
~~> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = "blue")) + geom_point()~~

> ### Correctly color all points blue
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) + geom_point(color = "blue")
```



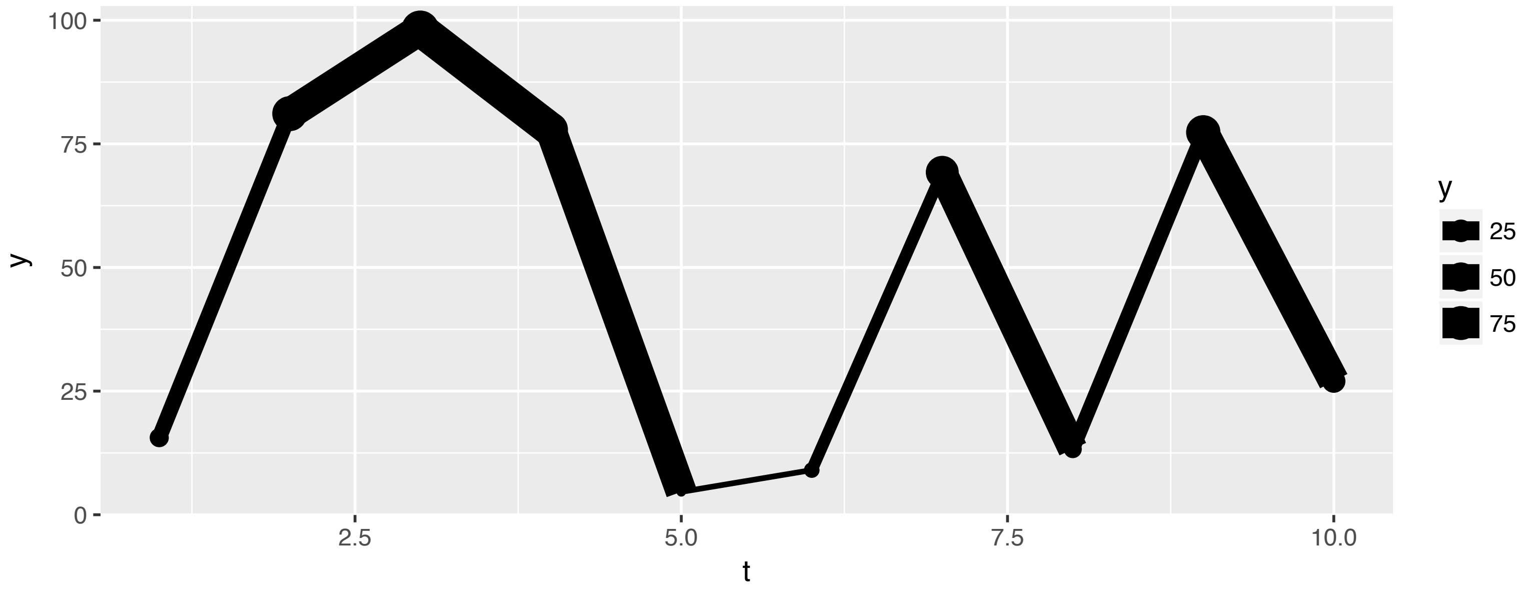
Example: multiple geoms

```
> ### Use some fake data:
> fake.data <- data.frame(t = 1:10, y = runif(10, 1, 100))

> ggplot(fake.data, aes(x = t, y = y)) + geom_point() + geom_line()
```



Make sure aesthetic mappings are properly applied

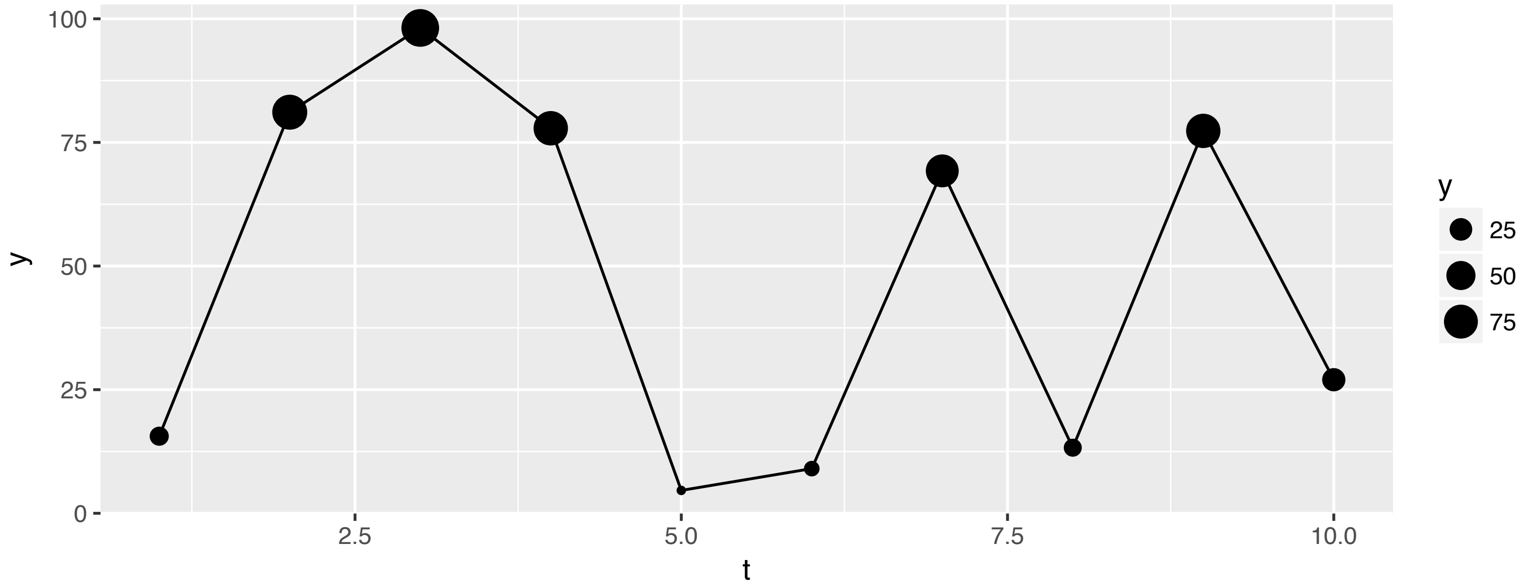
```
> ggplot(fake.data, aes(x = t, y = y, size = y)) + geom_point() + geom_line()
```



Make sure aesthetic mappings are properly applied

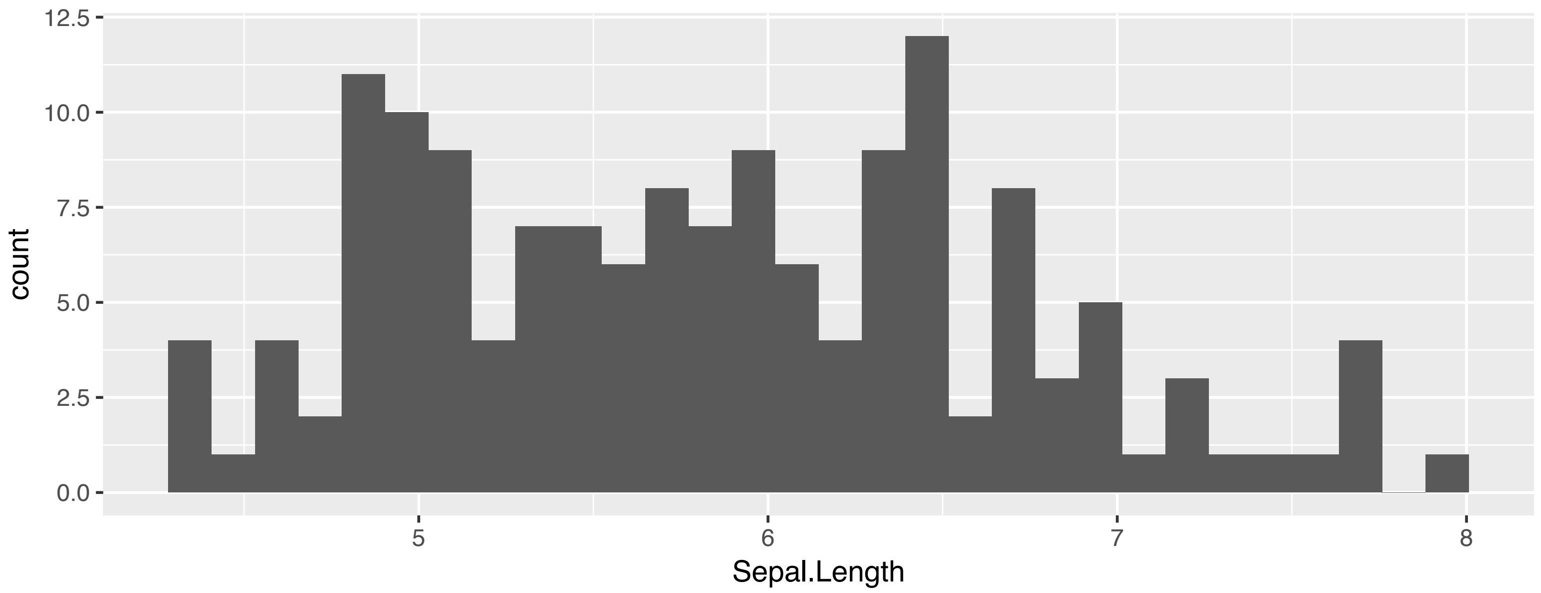
```
~~> ggplot(fake.data, aes(x = t, y = y, size = y)) + geom_point() + geom_line()~~
```

```
> ggplot(fake.data, aes(x = t, y = y)) + geom_point( aes(size=y) ) + geom_line()
```



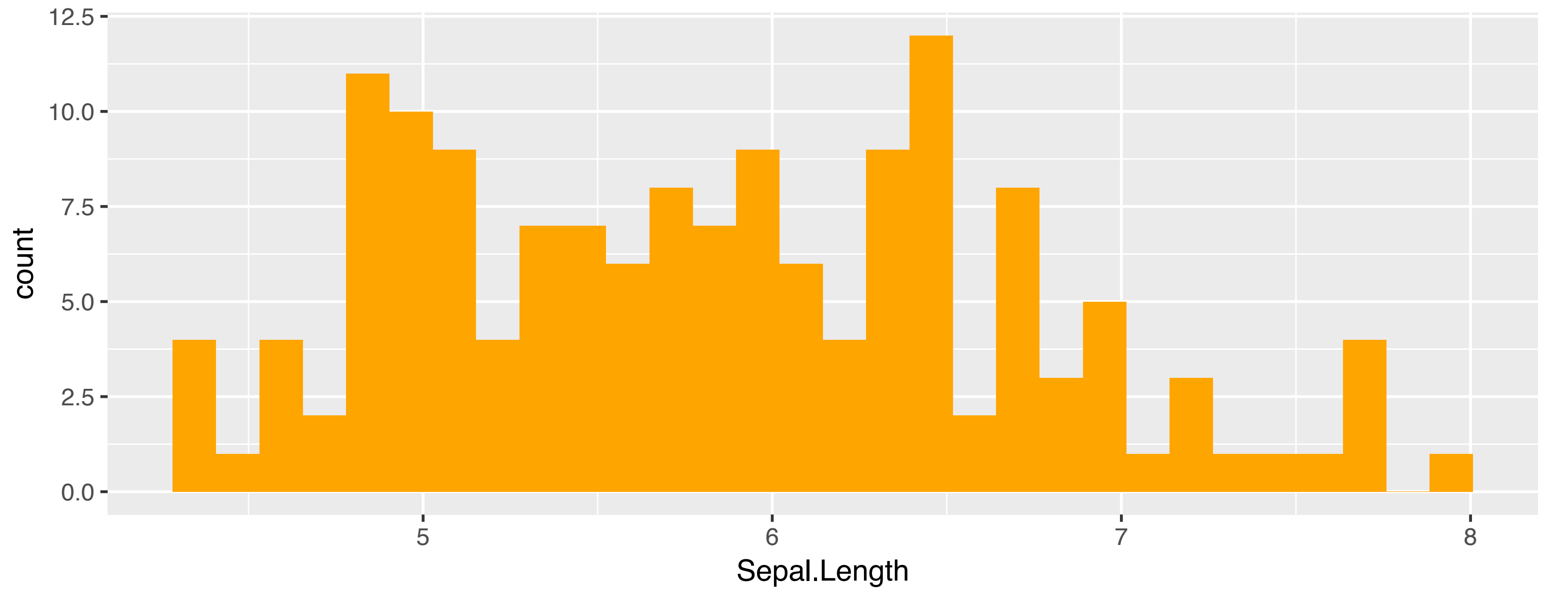
Histograms

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



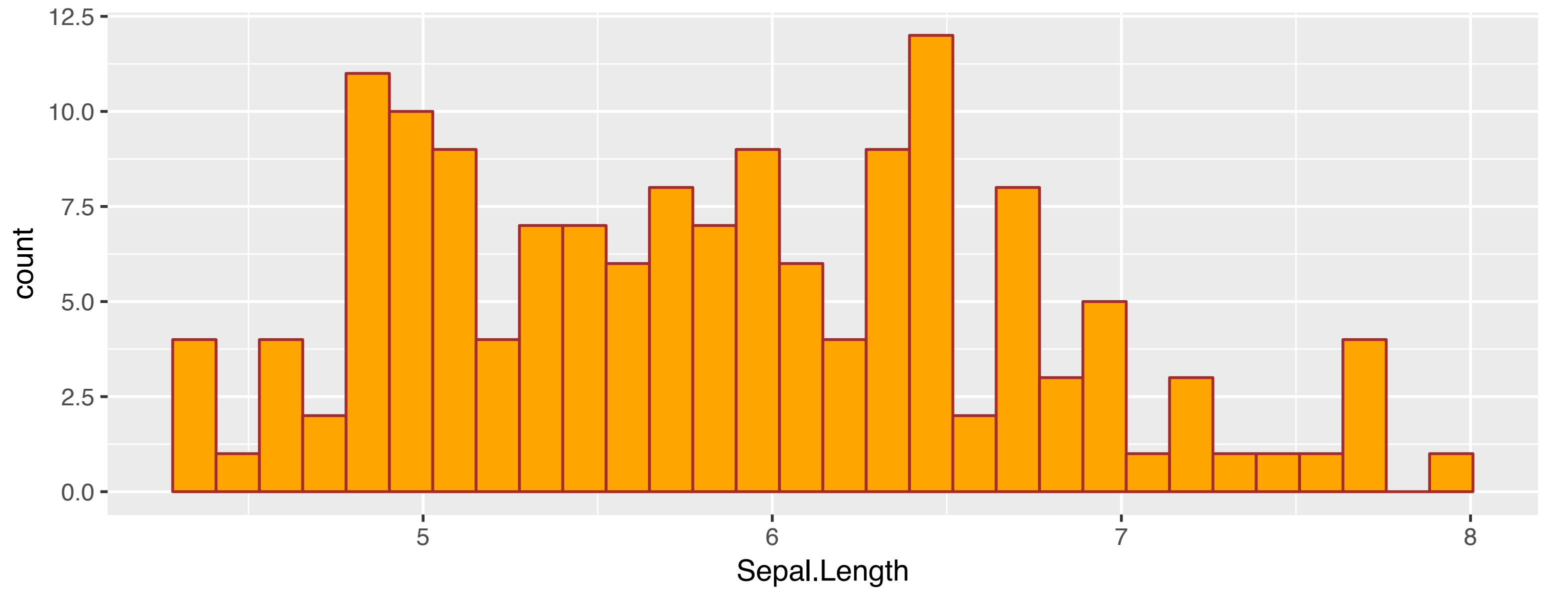
Histograms

```
> ggplot(iris, aes(x = Sepal.Length)) + geom_histogram( fill = "orange" )
```



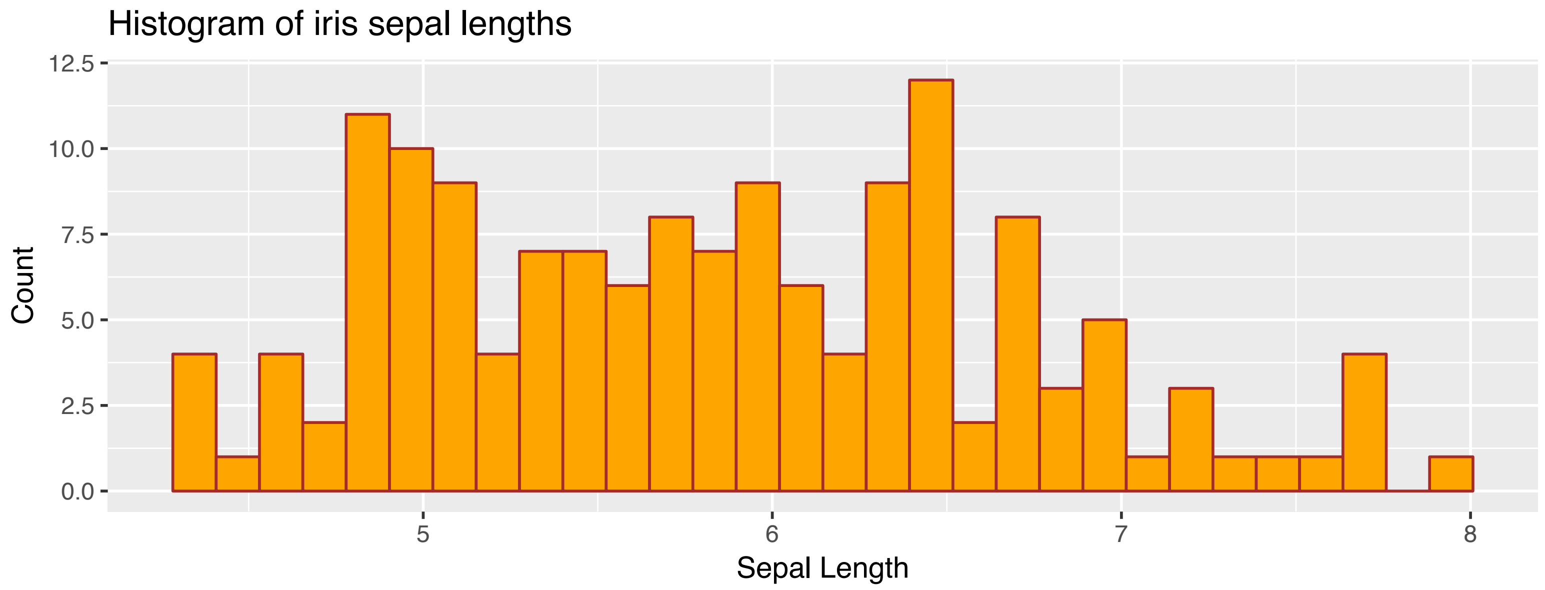
Histograms

```
> ggplot(iris, aes(x = Sepal.Length)) + geom_histogram( fill = "orange", line = "brown" )
```



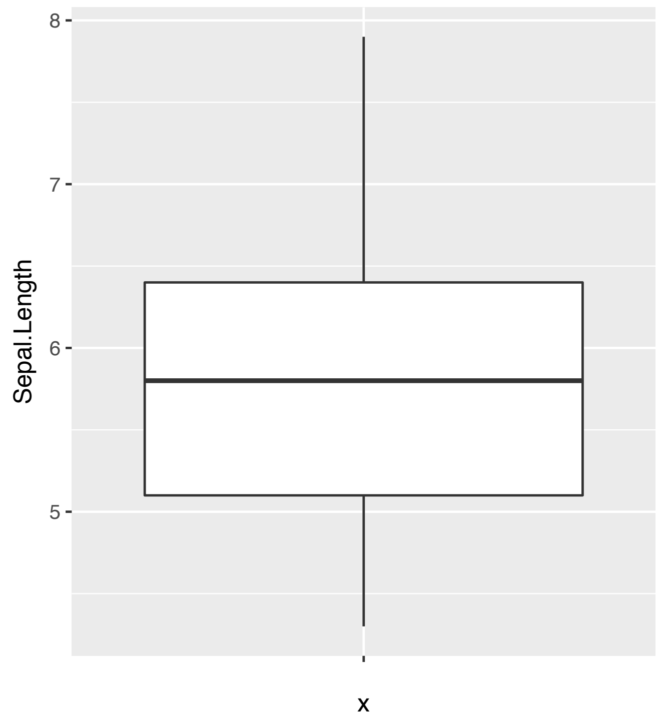
Histograms

```
> ggplot(iris, aes(x = Sepal.Length)) + geom_histogram( fill = "orange", color = "brown" )
  + xlab("Sepal Length") + ylab("Count") + ggtitle("Histogram of iris sepal lengths")
```



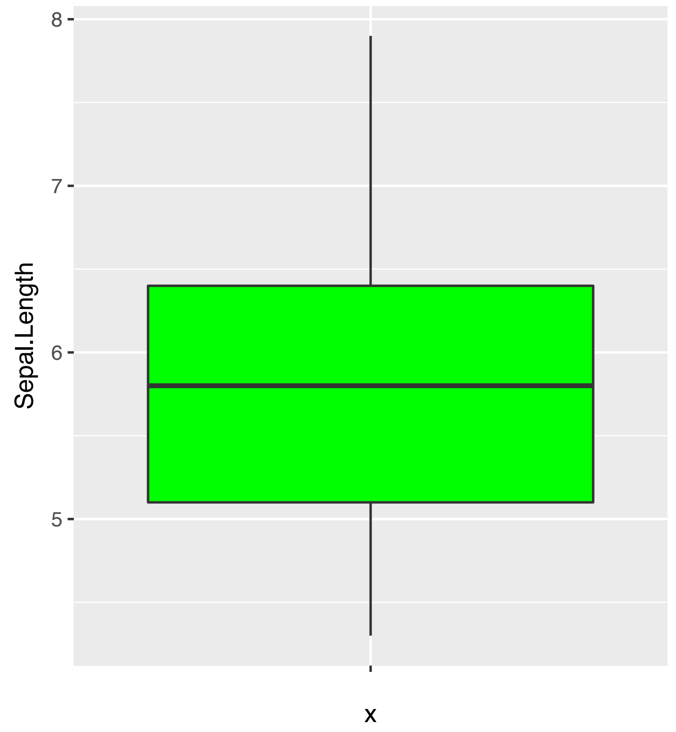
Boxplots

```
> ggplot(iris, aes(x = "", y = Sepal.Length)) + geom_boxplot()
```



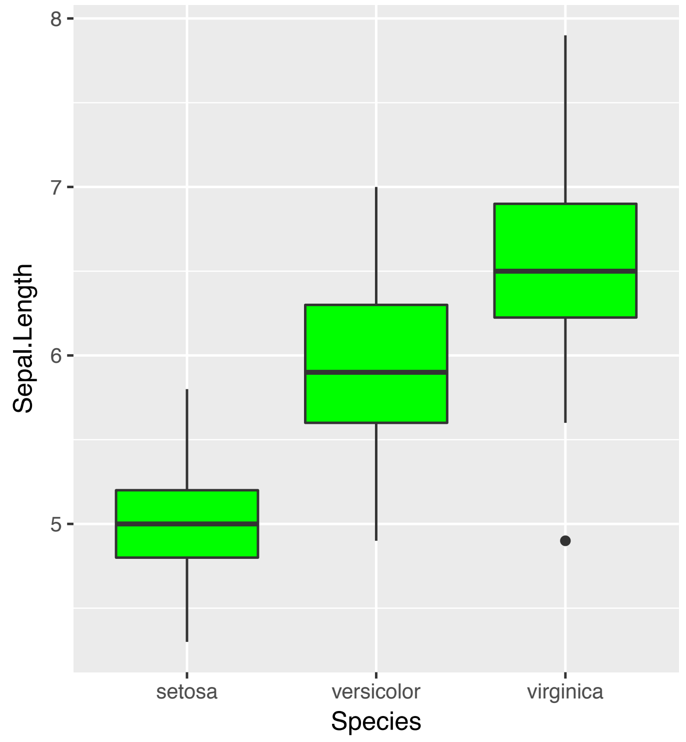
Boxplots

```
> ggplot(iris, aes(x = "", y = Sepal.Length)) + geom_boxplot(color = "green")
```



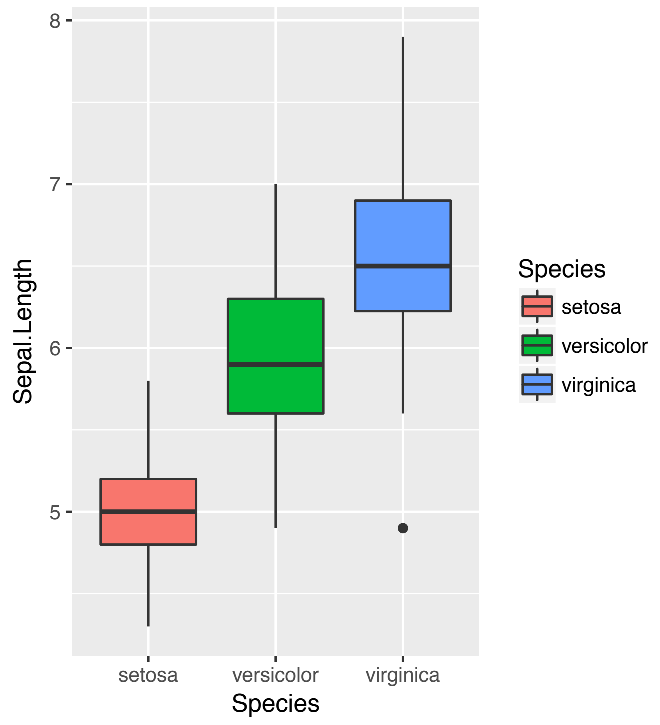
Boxplots

```
> ggplot(iris, aes(x = Species, y = Sepal.Length)) + geom_boxplot(color = "green")
```



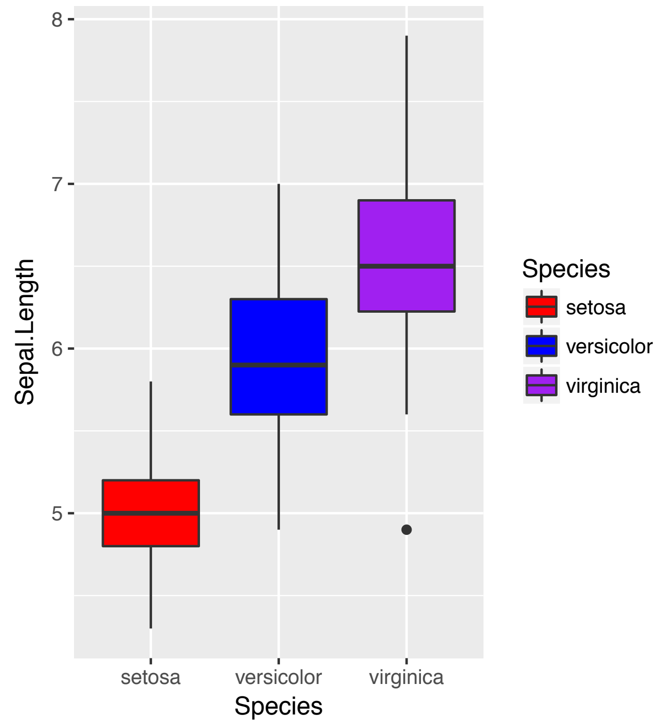
Boxplots

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



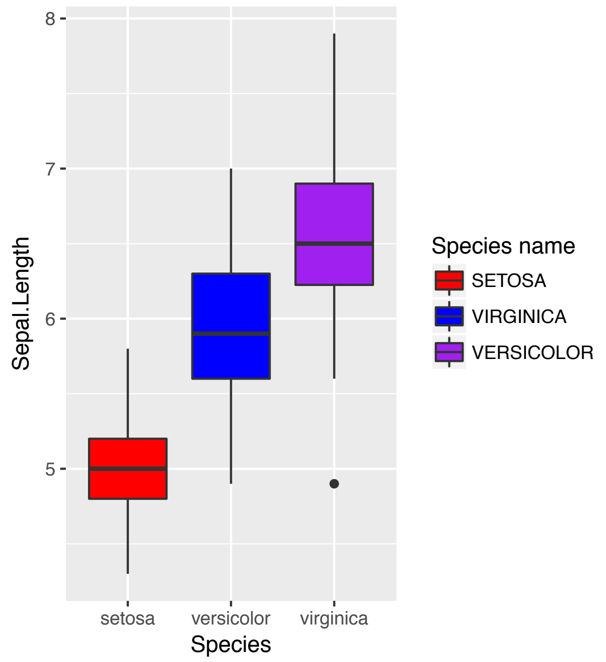
Boxplots: Customizing the fill mappings

```
> ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) + geom_boxplot() +
  scale_fill_manual(values=c("red", "blue", "purple"))
```



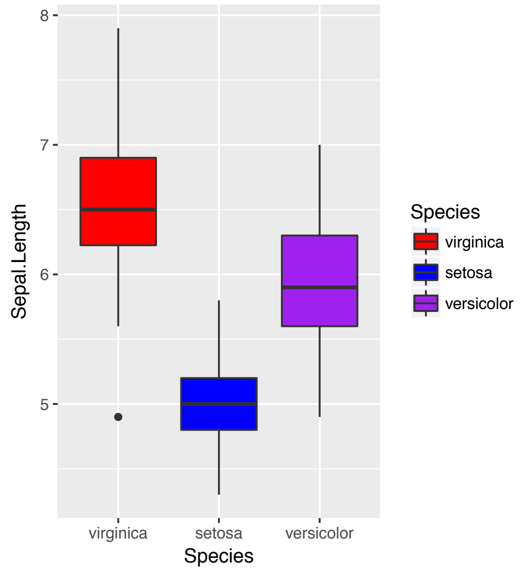
scale_fill_manual() also tweaks legend

```
> ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) + geom_boxplot() +
  scale_fill_manual(values=c("red", "blue", "purple"), name = "Species name",
  labels=c("SETOSA", "VIRGINICA", "VERSICOLOR"))
```



Changing the order

```
> ### Ordering depends on factor levels
> levels(iris$Species)
    [1] "setosa"     "versicolor" "virginica"

> ### Change order of levels
> iris$Species <- factor(iris$Species, levels=c("virginica", "setosa", "versicolor"))
    [1] "virginica"  "setosa"     "versicolor"

> ### Replot
> ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
    geom_boxplot() +
    scale_fill_manual(values=c("red", "blue", "purple"))
```



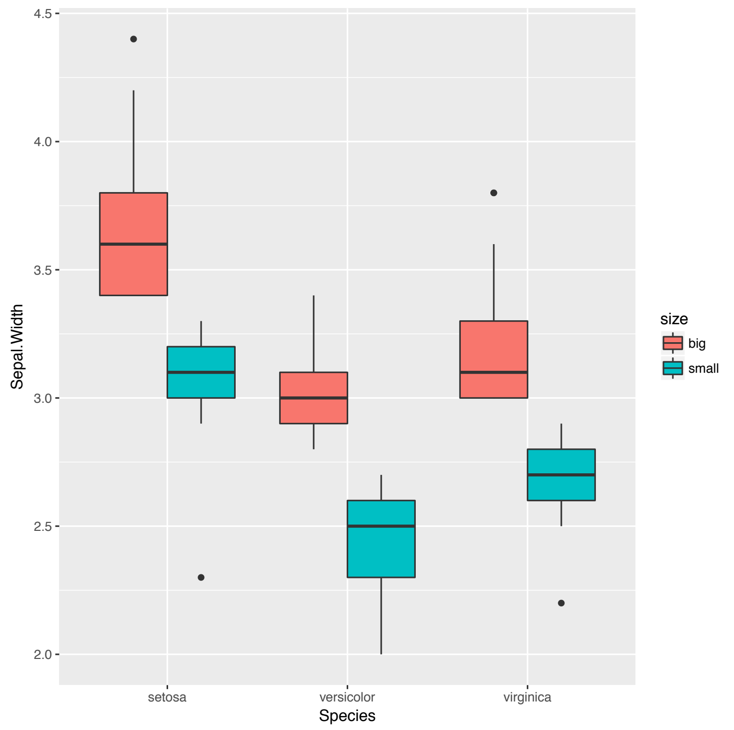
Grouped boxplots

This will apply to violin plots as well.

```
> ## Create another categorical variable for grouping purpopses
> iris %>%
   group_by(Species) %>%
   mutate(size = ifelse( Sepal.Width > median(Sepal.Width) , "big" , "small" )) -> iris2
                          Condition                         Value if  Value if
                                                            TRUE      FALSE
> head(iris2)
        Source: local data frame [150 x 6]
        Groups: Species [3]


          Sepal.Length Sepal.Width Petal.Length Petal.Width Species  size
                 <dbl>       <dbl>        <dbl>       <dbl>  <fctr> <chr>
        1          5.1         3.5          1.4         0.2  setosa   big
        2          4.9         3.0          1.4         0.2  setosa small
        3          4.7         3.2          1.3         0.2  setosa small
        4          4.6         3.1          1.5         0.2  setosa small
        5          5.0         3.6          1.4         0.2  setosa   big
        6          5.4         3.9          1.7         0.4  setosa   big
```

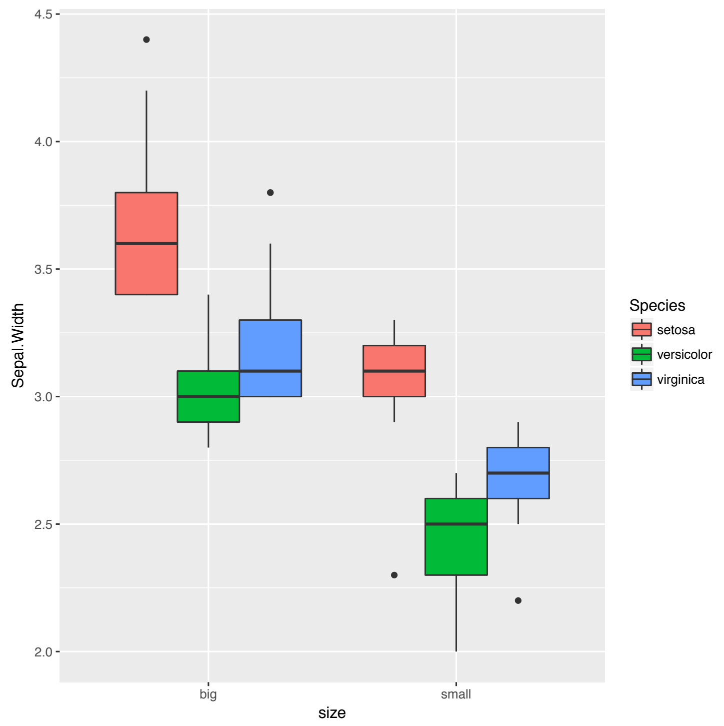
Grouped boxplots

```
> ggplot(iris2, aes( x = Species, fill=size, y=Sepal.Width)) + geom_boxplot()
```



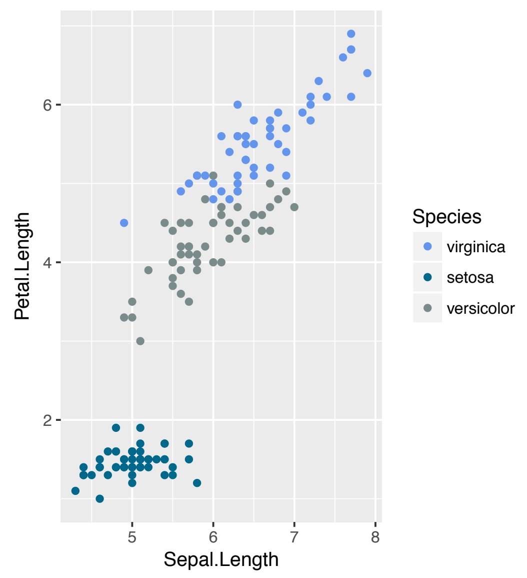
Grouped boxplots

```
> ggplot(iris2, aes( x = size, fill = Species, y=Sepal.Width)) + geom_boxplot()
```



Detour: scale_color_manual() customizes color

```
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) +
    geom_point(aes(color = Species)) +
    scale_color_manual(values=c("cornflowerblue", "deepskyblue4", "lightcyan4"))
```



Detour round 2: scale_<fill/color>_??

There are **many** scales to use besides default and custom.

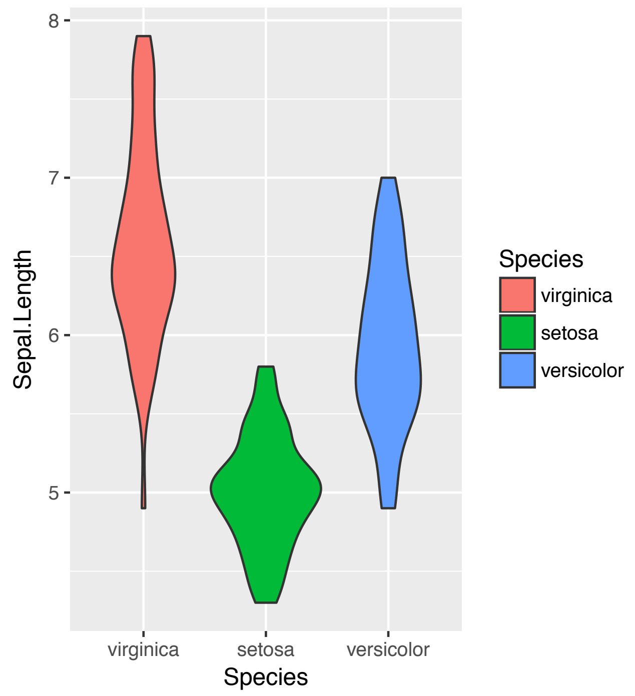
- scale_<fil/color>_brewer() uses pre-made color schemes from colorbrewer.org

- scale_color_gradient() can take a low and high to fill along a spectrum

See here: http://ggplot2.tidyverse.org/reference/#scales

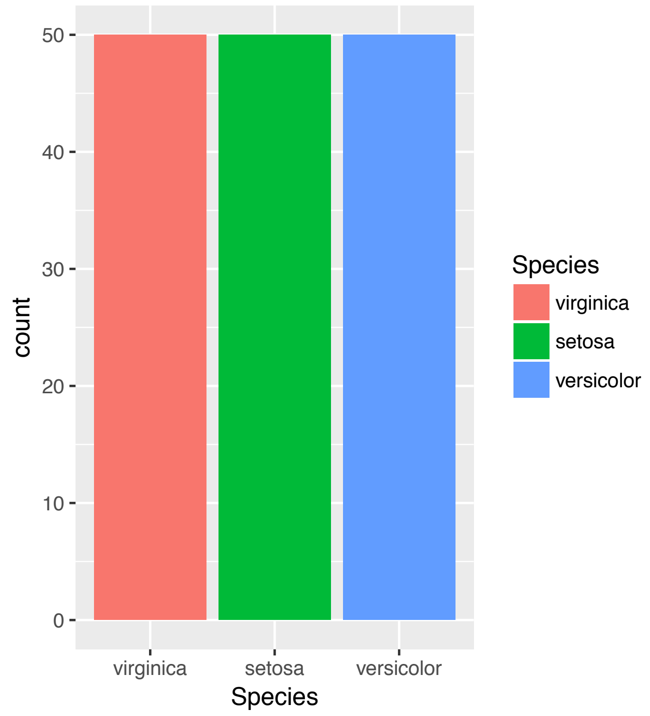
Violin plot

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



Bar plot

```
> ggplot(iris, aes(x = Species, fill = Species)) + geom_bar()
```

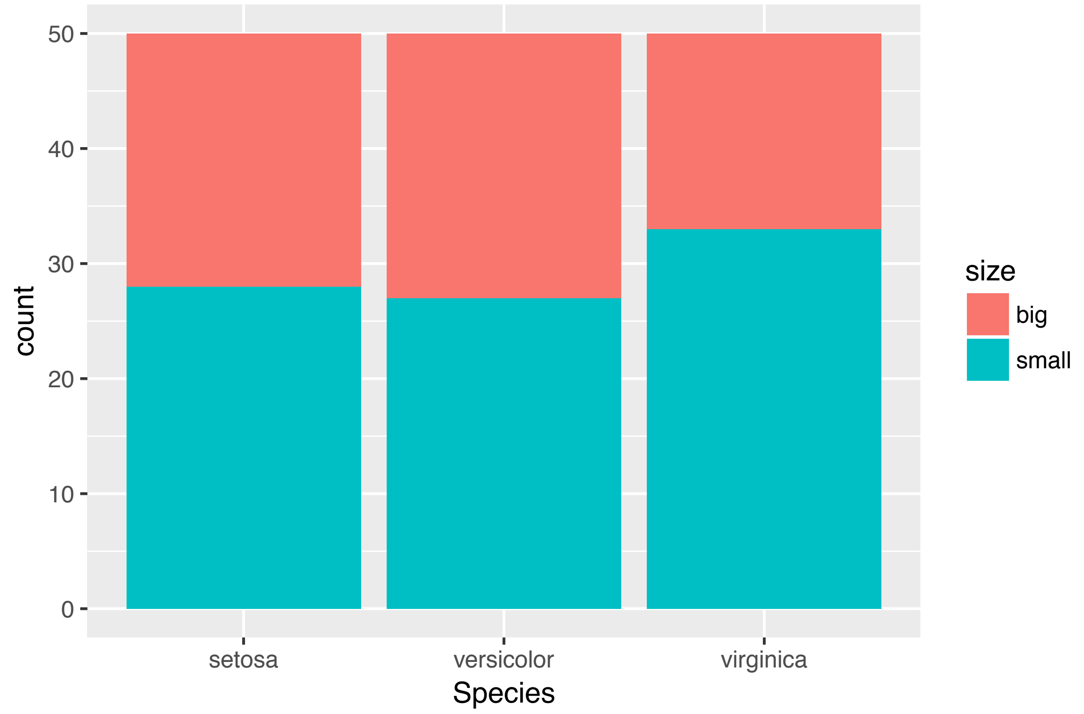


Stacked/grouped bar plot

```
> head(iris2)
        Source: local data frame [150 x 6]
        Groups: Species [3]
           Sepal.Length Sepal.Width Petal.Length Petal.Width Species  size
                  <dbl>       <dbl>        <dbl>       <dbl>  <fctr> <chr>
        1           5.1         3.5          1.4         0.2  setosa   big
        2           4.9         3.0          1.4         0.2  setosa small
        3           4.7         3.2          1.3         0.2  setosa small
        4           4.6         3.1          1.5         0.2  setosa small
        5           5.0         3.6          1.4         0.2  setosa   big
        6           5.4         3.9          1.7         0.4  setosa   big
```

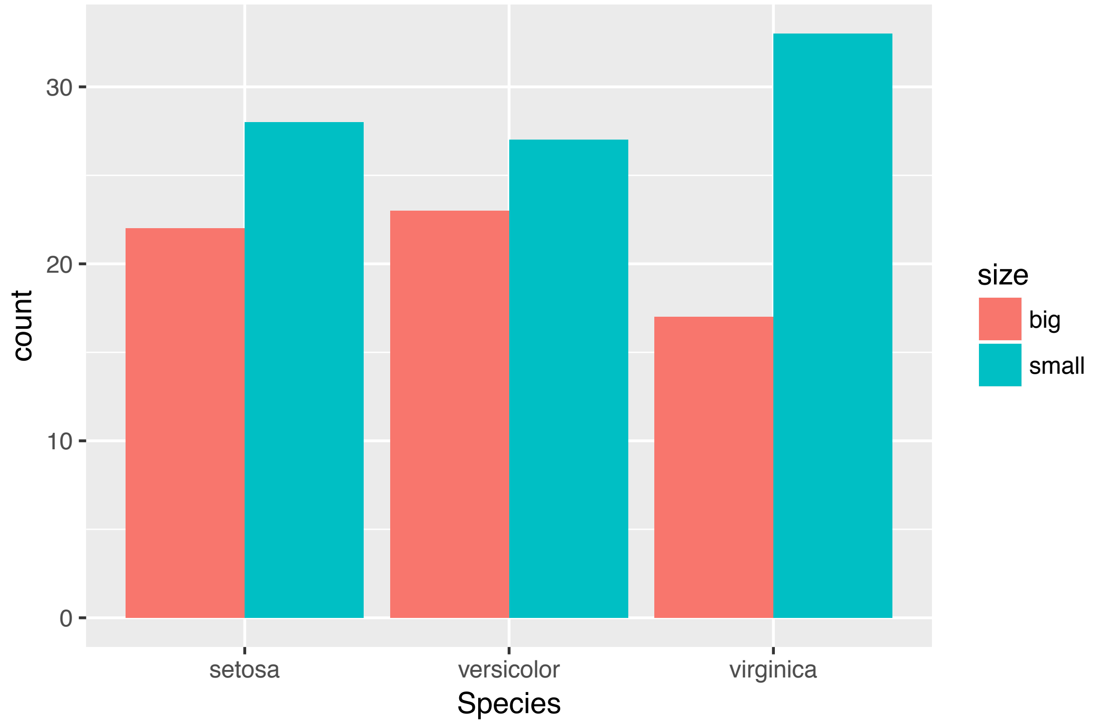
Stacked/grouped bar plot

```
> ggplot(iris, aes(x = Species, fill = size)) + geom_bar()
```



Stacked/grouped bar plot

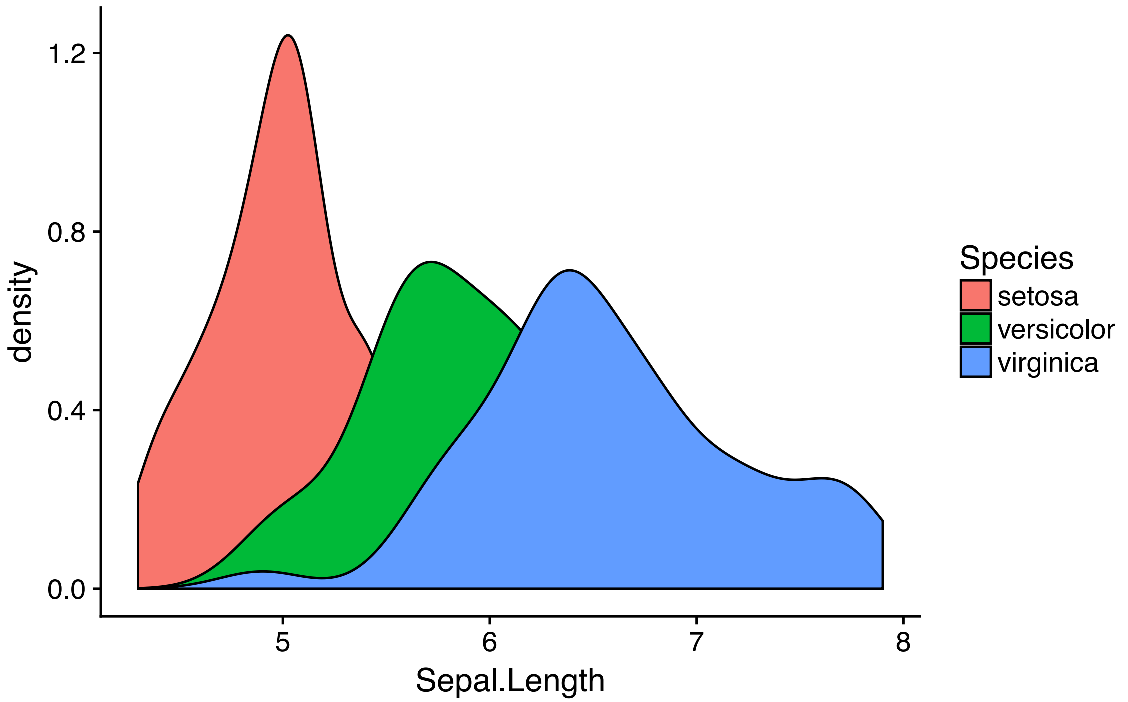
```
> ggplot(iris, aes(x = Species, fill = size)) + geom_bar( position = "dodge" )
```



Density plot

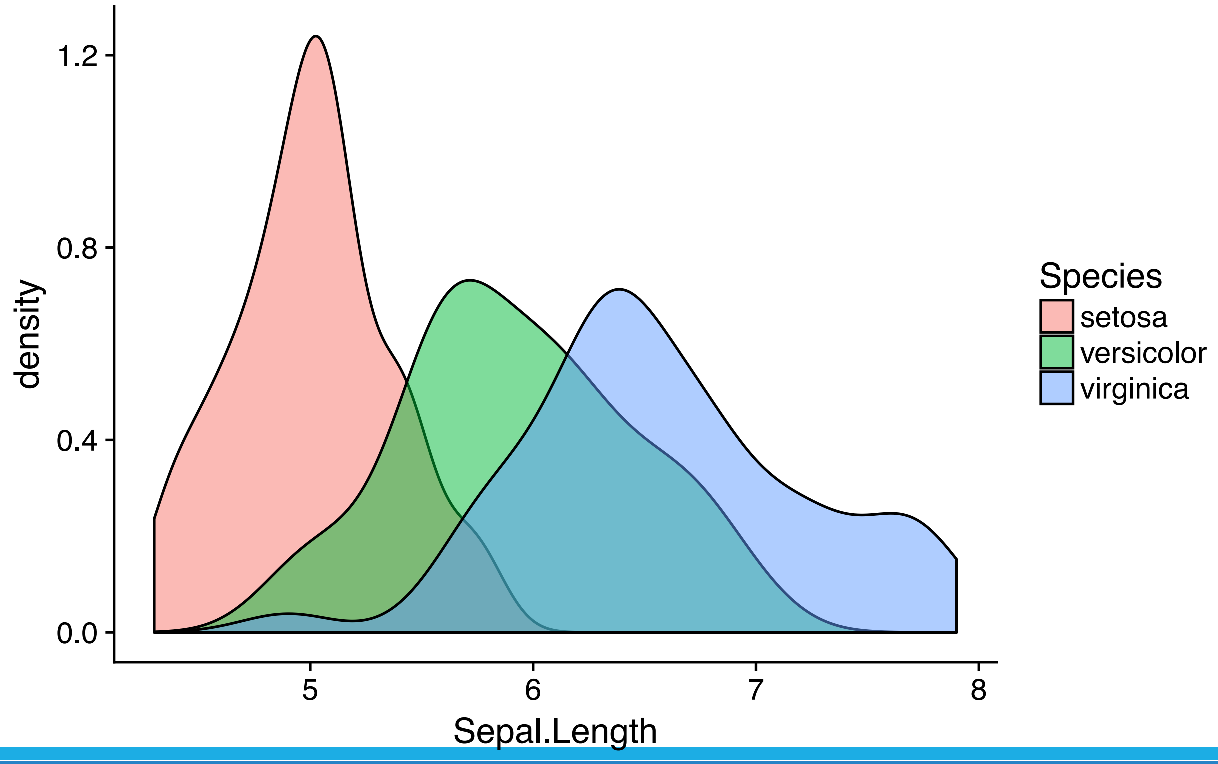
```
> ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) + geom_density()
```

What does the tail of the setosa distribution look like?



Density plot

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


density
Sepal.Length
Species
setosa
versicolor
virginica