

# **Lecture 2.2 – Basic Plotting with R Markdown**

## **Specific Learning Objectives:**

**2.2.1 – Create reproducible scripts in R.**

**2.2.2 – Include effective documentation in scripts and projects.**

**2.2.4 – Create and use Notebooks and documents using RMarkdown.**

**3.2 – Learn how to plot quickly using R's base graphics.**

# Literate Programming and Reproducible Science

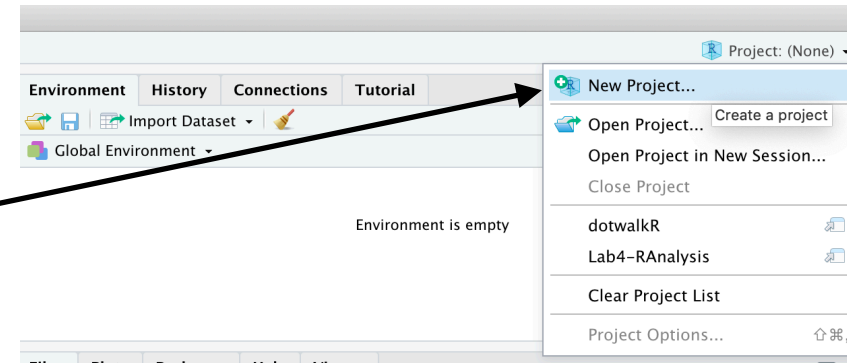
- One of the basic tenets of science is that work is reproducible. This applies to code as well!
  - We will talk more about how to accomplish this in the next unit, but for now, we'll focus on using **documentation** and **literate programming** to build reproducible analyses.
- What is **documentation**?
  - Code documentation refers to all of the notes, instructions, and comments that provide context and help a user run the code successfully
- What is **literate programming**?
  - Literate programming is the method of directly embedding chunks of code in the documentation that helps the user understand the code and its outputs.

# R Markdown – a laboratory notebook for code!

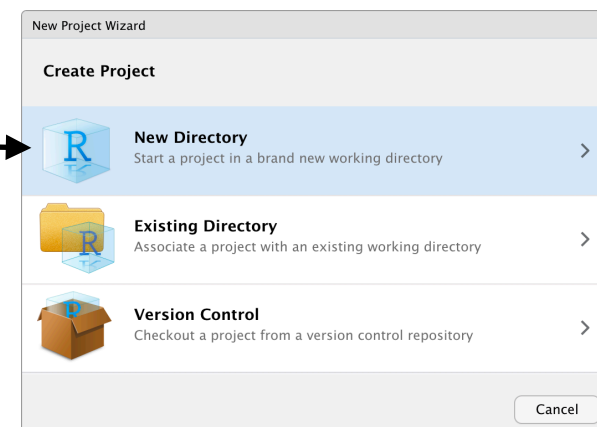
- R Markdown documents help by giving you the ability to embed your code into a regular document.
  - This gives you the space to explain your code, plots, figures, and analyses in context!
  - It helps you by providing a reproducible environment for your code and plots.
- We'll use R Markdown documents with R Projects
  - R projects help you organize your data and code into a neat package.
  - R projects will automatically set the working directory for you, so you don't have to worry about it!
  - R projects will keep track of the work you do in that specific project so it doesn't get mixed up with other projects!

# Creating an R Project

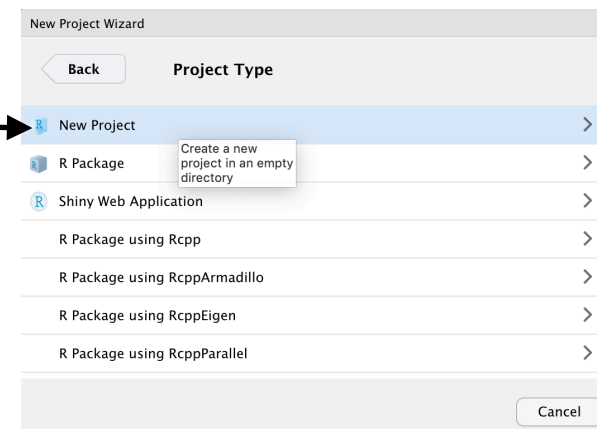
1. Either under “File...” or at the top right corner, select “New Project...”



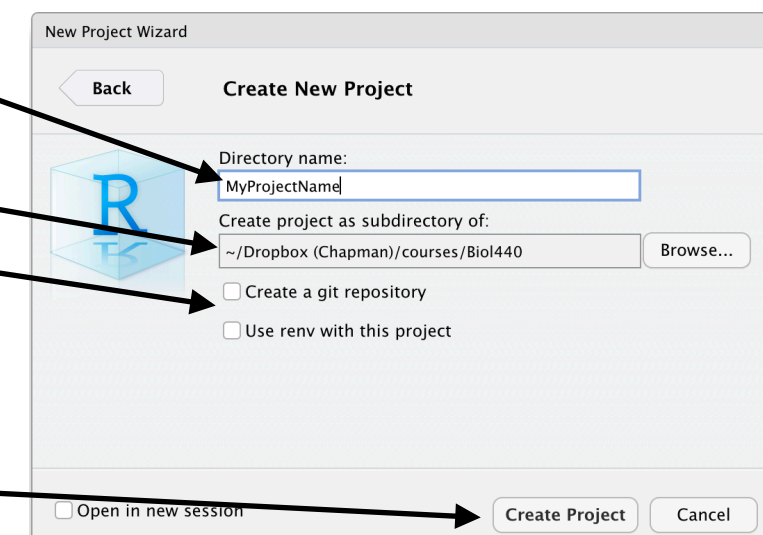
2. Select “New Directory” in the Wizard.



3. Select “New Project” again.



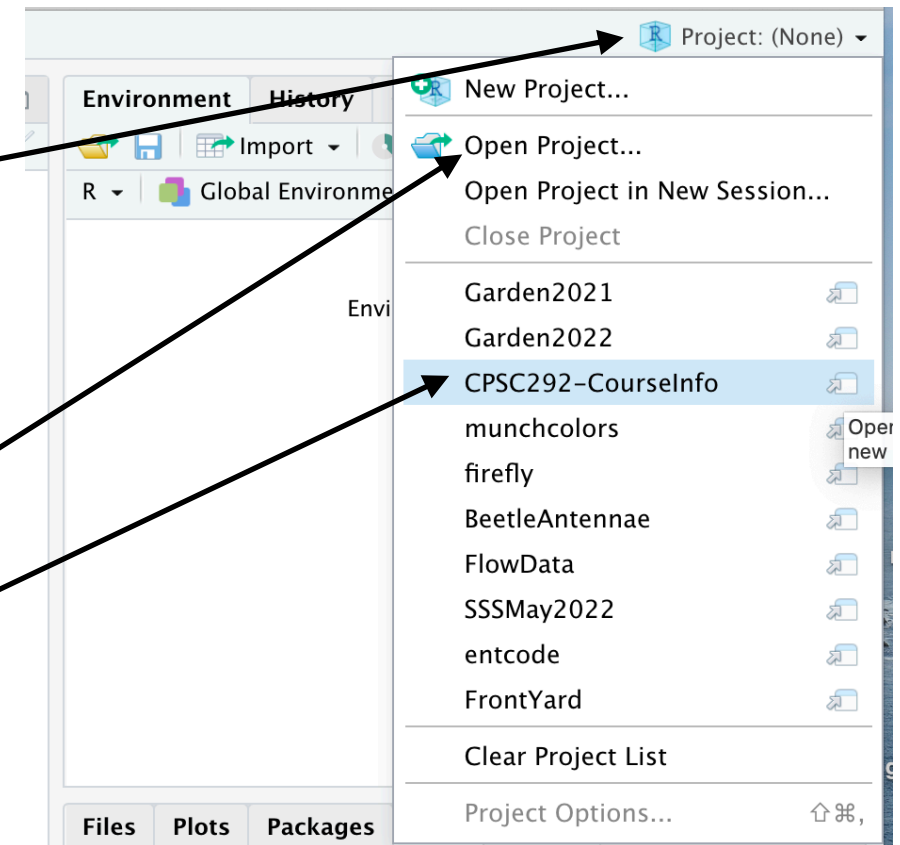
4. Create a directory name for your project. Choose where you want the project directory to be (using Browse...). Uncheck both boxes.



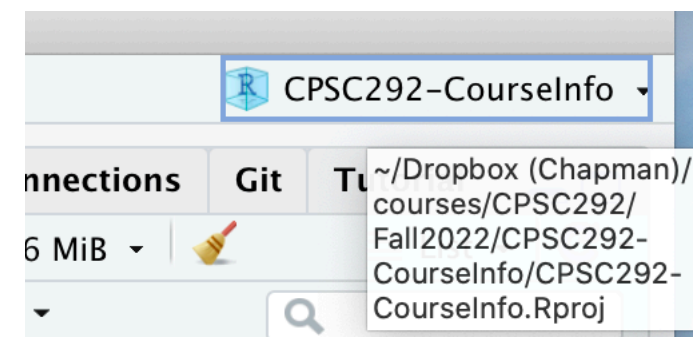
5. Click “Create Project”!

# Using R Projects

1. In the upper right corner, you'll see a project status. Right now, no project is open.
2. To open an existing project, you can either select "Open Project..." or select a project from the list.



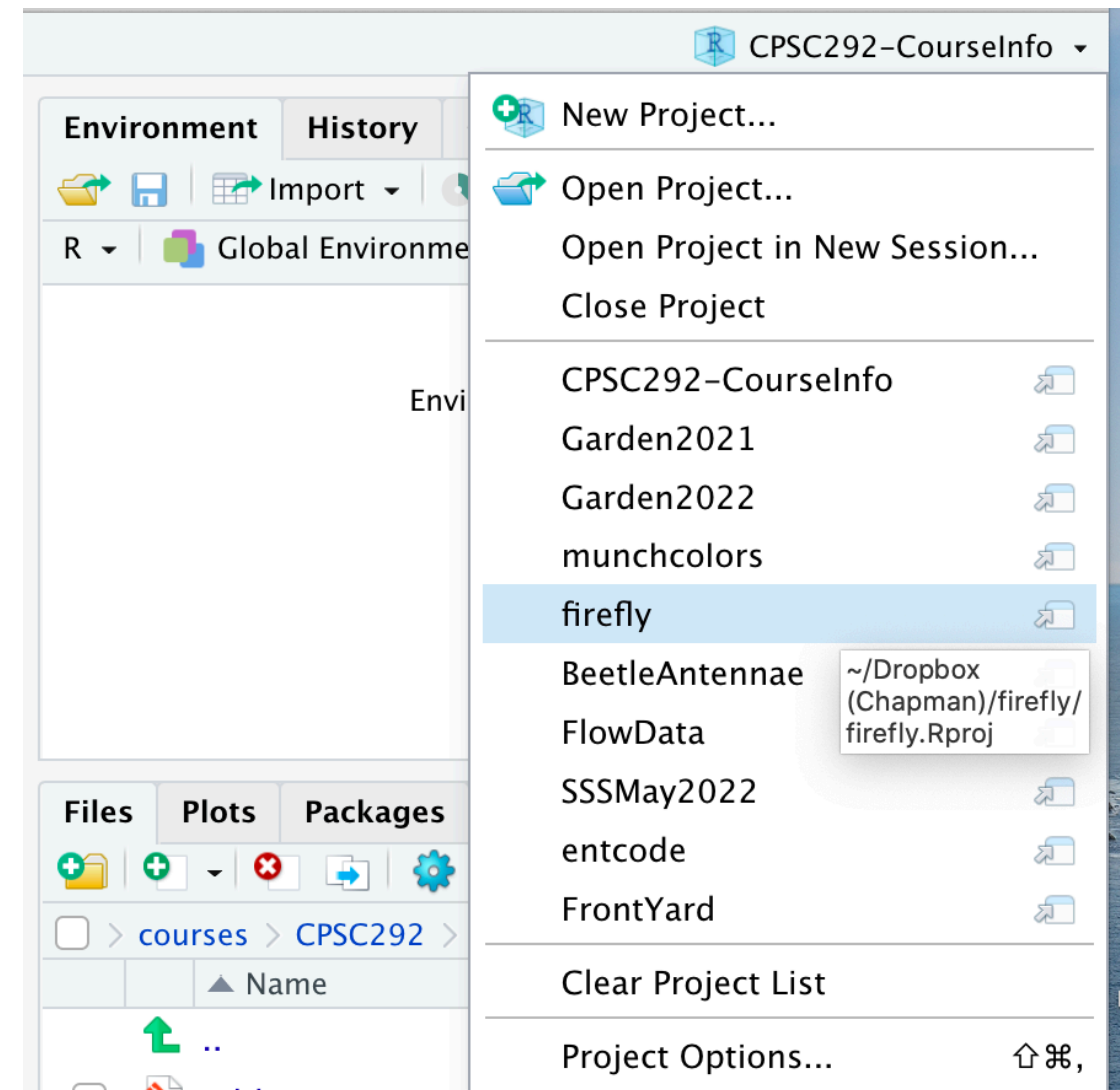
**Project is now open, status is updated!**



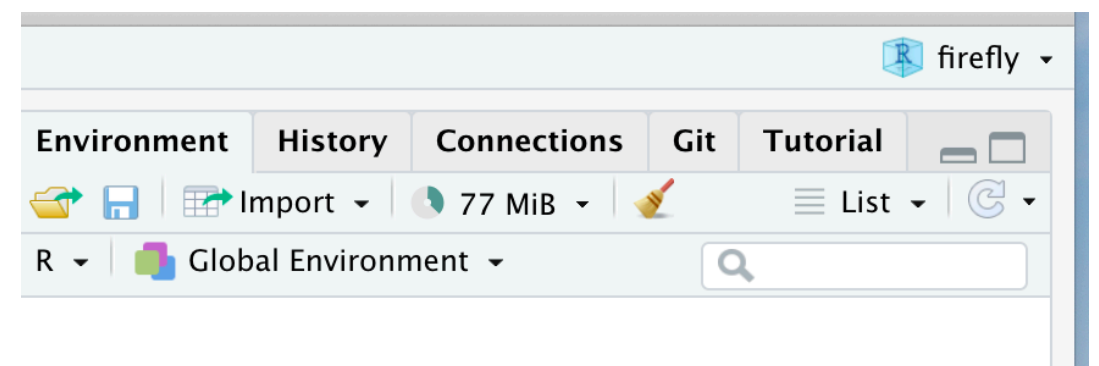
# Using R Projects

Remember, projects are used to separate workflows for different uses of code. When you switch uses (different homework, different project), remember to switch out your R Project!!

I'm switching from working on classwork (CPSC-292-CourseInfo RProject) to research work (firefly RProject).



**Project is now open, status is updated!**



# Your First R Markdown Notebook

Click on “New File” tab the in upper left and select “R Notebook”.



You’ll get an untitled notebook template to use. Change the title and save with a name ending in .Rmd

# R Markdown Notebook Features

- R Markdown documents consist of **Markdown text** and **R code chunks**.

The image shows a screenshot of an R Markdown notebook editor window titled 'Untitled1'. The editor displays a sequence of alternating Markdown text and R code chunks. On the left side, three vertical brackets with labels identify these sections: 'Markdown text' for the first and third blocks, and 'R code "chunk"' for the second block. The first Markdown block (lines 1-9) contains a title, output format, and introductory text. The second R code chunk (lines 10-12) contains a single R command. The third Markdown block (lines 14-19) provides instructions on how to use the editor's features. The interface includes a toolbar at the top with icons for navigation, saving, previewing, and running code. The code chunk is highlighted with a light gray background.

**Markdown text**

```
1 ---
2 title: "R Notebook"
3 output: html_notebook
4 ---
5
6 This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the
7 notebook, the results appear beneath the code.
8
9 Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor
10 inside it and pressing *Cmd+Shift+Enter*.
```

**Writing goes here.**

**R code "chunk"**

```
10 ```{r}
11 plot(cars)
12 ```
```

**R code goes here!**

**Markdown text**

```
14 Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing
15 *Cmd+Option+I*.
16
17 When you save the notebook, an HTML file containing the code and output will be saved alongside it
18 (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).
19
20 The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike
*Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was
last run in the editor is displayed.
```

**Writing goes here.**

- R code will not work in Markdown text!!!



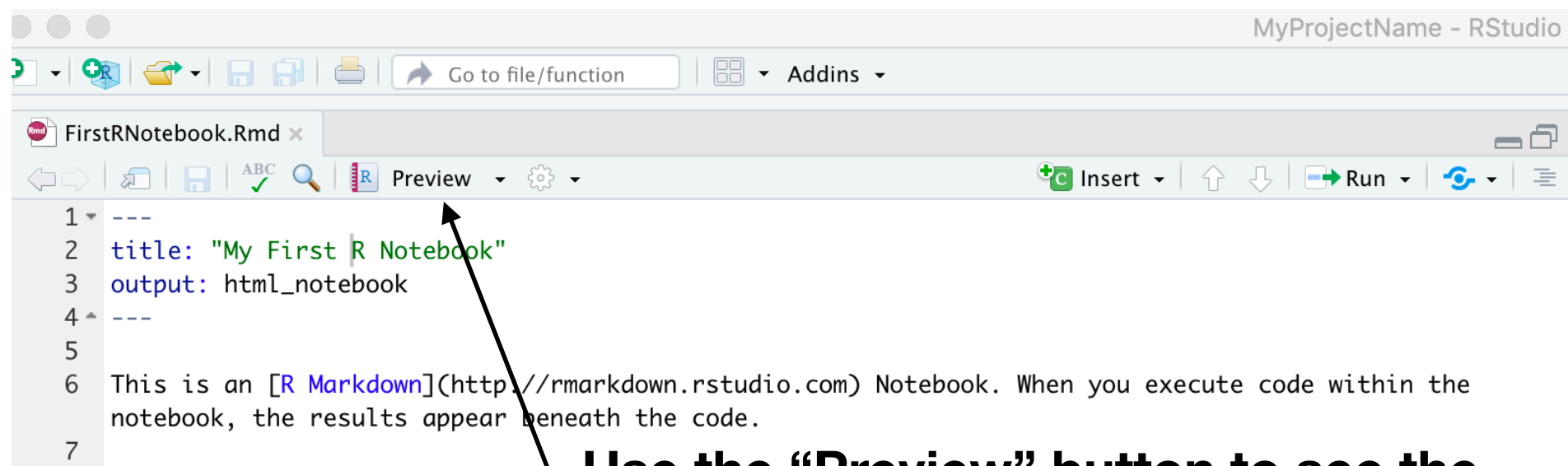
# R Markdown Notebook Features

- Running R code chunks will produce output for you embedded in the Markdown document.

```
10 {r}  
11 plot(cars)  
12 {r}  
13
```

**Use the run button to execute code in chunks within the document.**

- To view the full document, hit “Preview”.

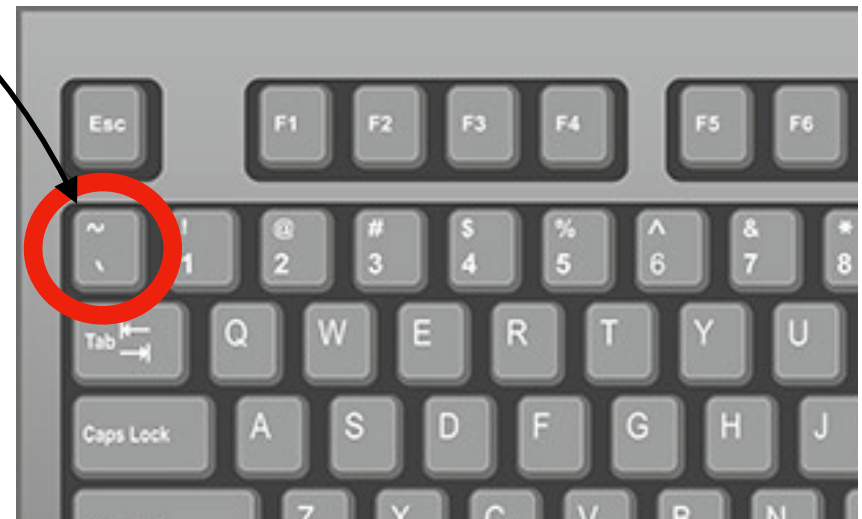


**Use the “Preview” button to see the html version of your markdown.**

# R Markdown Notebook Features

- Markdown text is very simple to manipulate.
  - Link to website: [R Markdown] (<https://rmarkdown.rstudio.com>)
  - Italics text: *text*
  - Bold text: **text**
  - Strikethrough text: ~~text~~
  - Superscript: <sup>text</sup>
  - Subscript: <sub>text</sub>
  - Headers: # header text

tildé

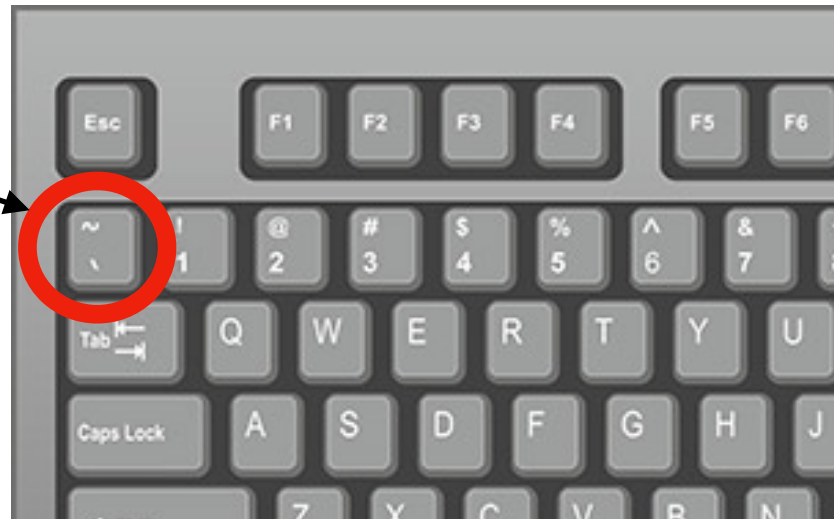


# R Markdown Notebook Features

## - R Code Chunks:

- begin with: ````\{r\}`
- end with: `````

open apostrophe



Click play button to run the chunk

```
10 ▾ ```{r}
11   plot(cars)
12 ▴ ```
```

click cog to get different options for running code chunk

If this field isn't gray, then the document doesn't recognize it as a code chunk!

From now on, work in an R Markdown document!

# Check Your Understanding

**In your basic R Markdown notebook, add a header, some text, and a new chunk with a bit of code (doesn't matter what it is). Try previewing the document!**

# Basic Plotting

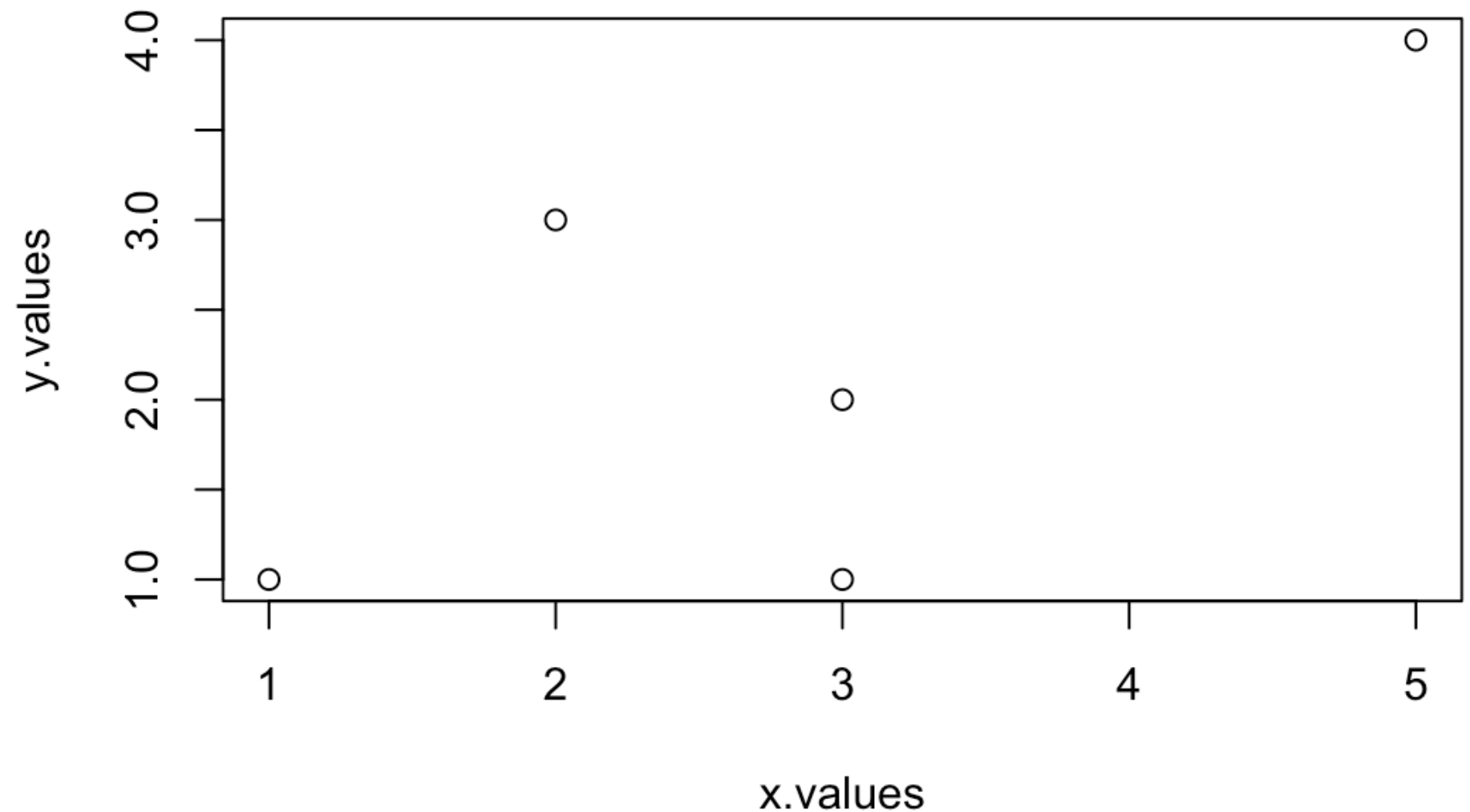
- R is a popular language because of how easy it is to produce beautiful plots!

Basic plot using  
the `plot()`  
command

```
```{r}
x.values <- c(3, 2, 5, 1, 3)
y.values <- c(1, 3, 4, 1, 2)

plot(x = x.values, y = y.values)
```
```

**Note: the vectors must be  
the same length!**



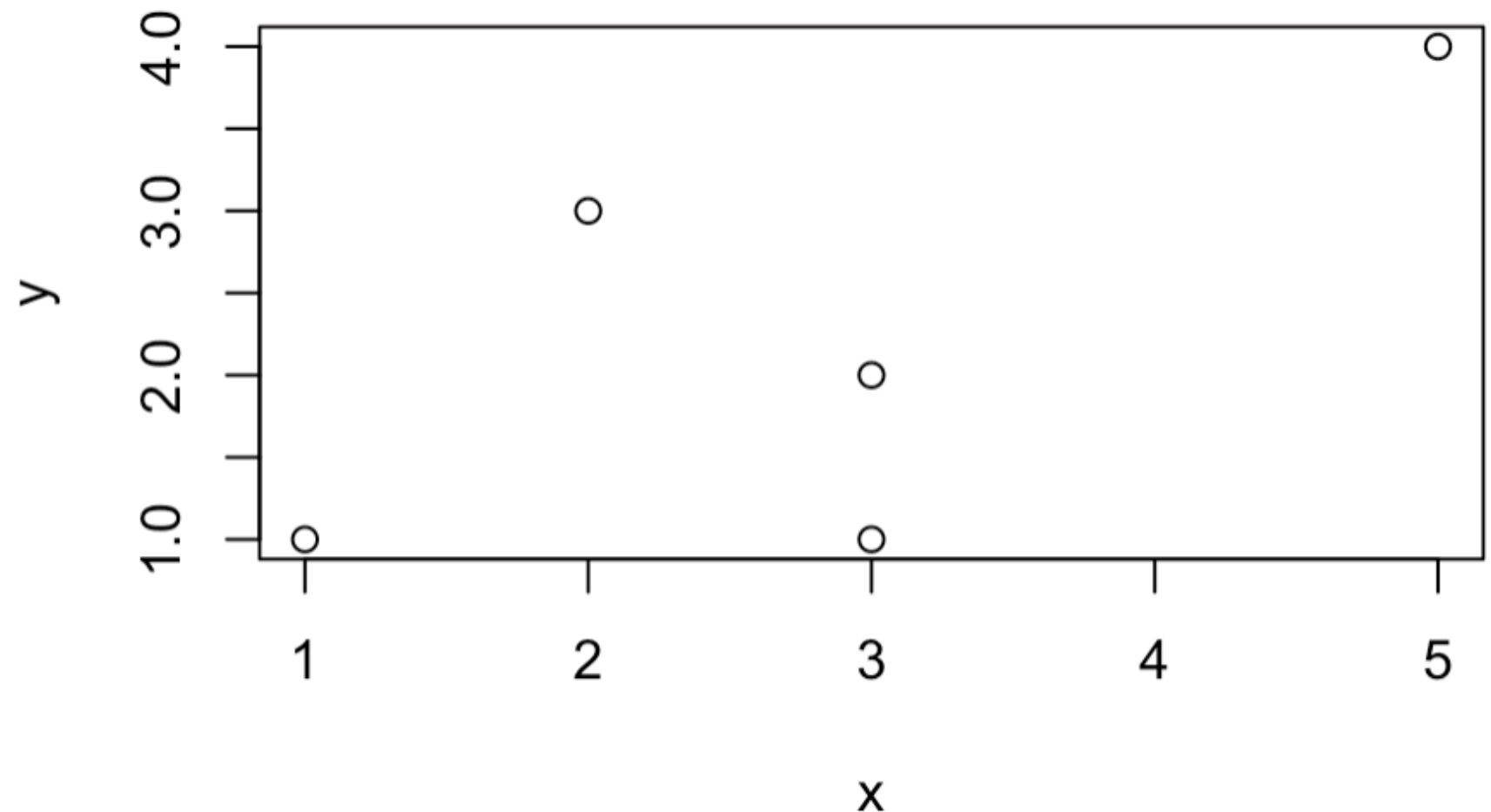
# Basic Plotting

**Basic plot using  
the `plot()`  
command and a  
data frame**

**Using a formula  
(`y~x`) and the  
argument  
`data=df` will  
give you the  
same result.**

**Remember: there's  
often multiple ways  
to do the same  
thing!**

```
```\r}\n x.values <- c(3, 2, 5, 1, 3)\ny.values <- c(1, 3, 4, 1, 2)\n\ndf <- data.frame(x = x.values, y = y.values)\n\nplot(y~x, data = df)\n```\n
```



# Basic Plotting: Titles and Axis Labels

- Additional arguments in `plot()` can edit different features of the plot.

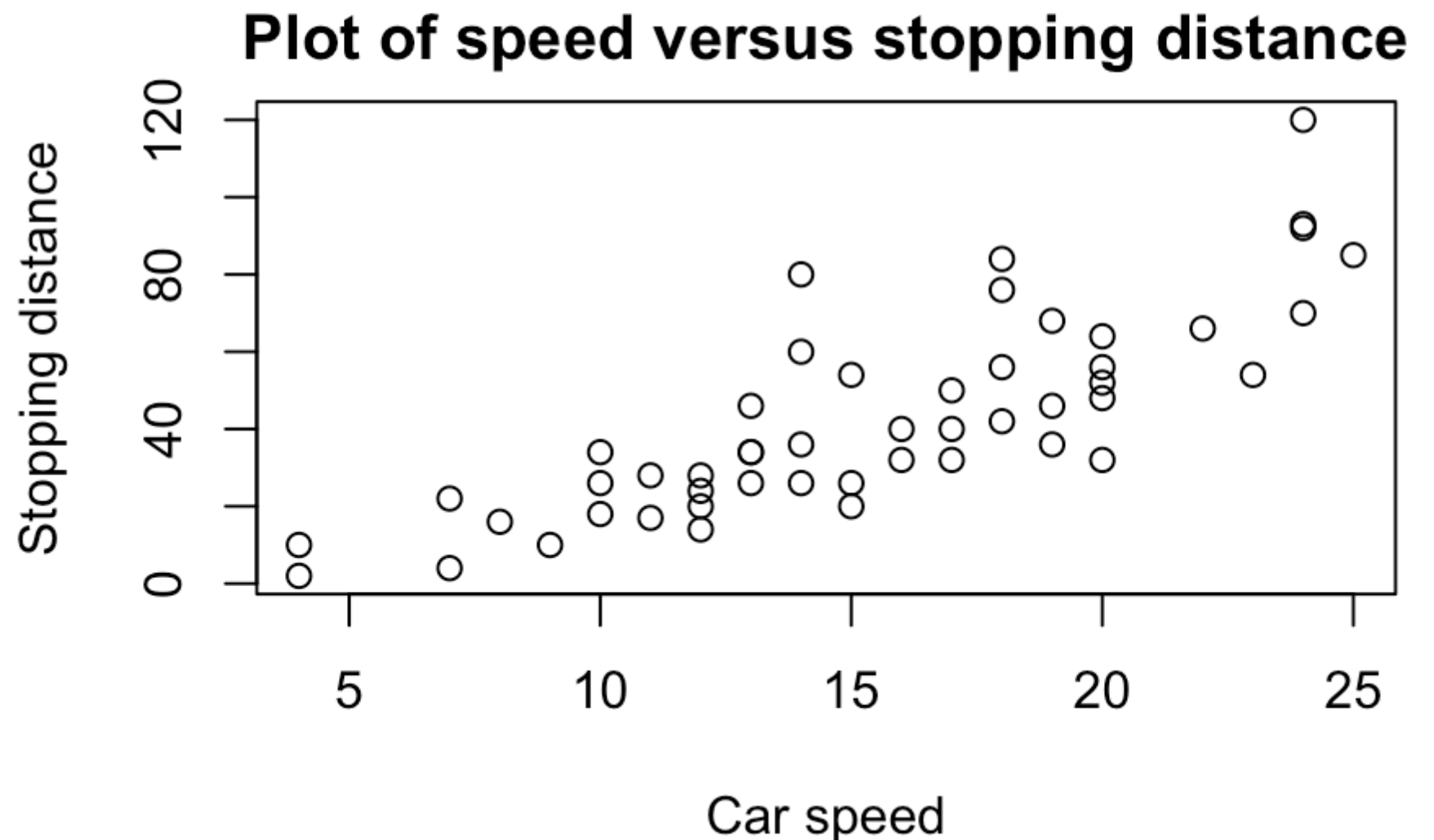
Hitting enter after a comma will help indent the function, making it easier to read!

**xlab:** x-axis label

**ylab:** y-axis label

**main:** main plot title

```
```{r, include=TRUE}
plot(dist ~ speed, data = cars,
      xlab = "Car speed",
      ylab = "Stopping distance",
      main = "Plot of speed versus stopping distance")
```
```

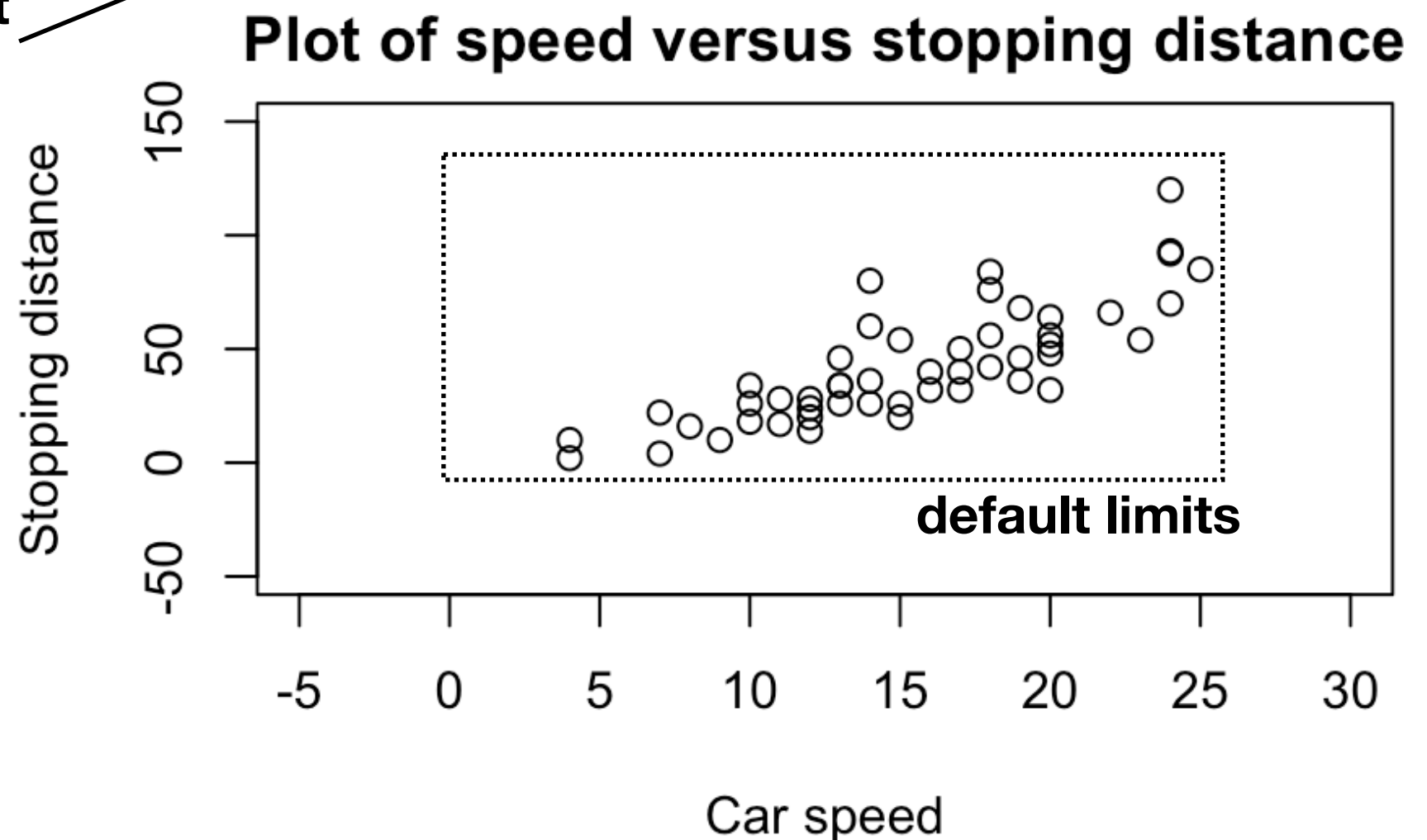


# Basic Plotting: Axis Limits

**xlim:** vector length 2 that sets x-axis limits

**ylim:** vector length 2 that sets y-axis limits

```
```{r, include=TRUE}
plot(dist ~ speed, data = cars,
      xlab = "Car speed",
      ylab = "Stopping distance",
      main = "Plot of speed versus stopping distance",
      xlim = c(-5,30), ylim = c(-50,150))
```
```





# Check Your Understanding

**Make a plot of vapor pressure in mmHg versus temperature in C of mercury using the pressure data set. Set the axis labels and title.**

**Make the same plot using a different method (either with or without a formula).**

# Basic Plotting: Formatting Plots

- Many attributes can be included as argument in plot:

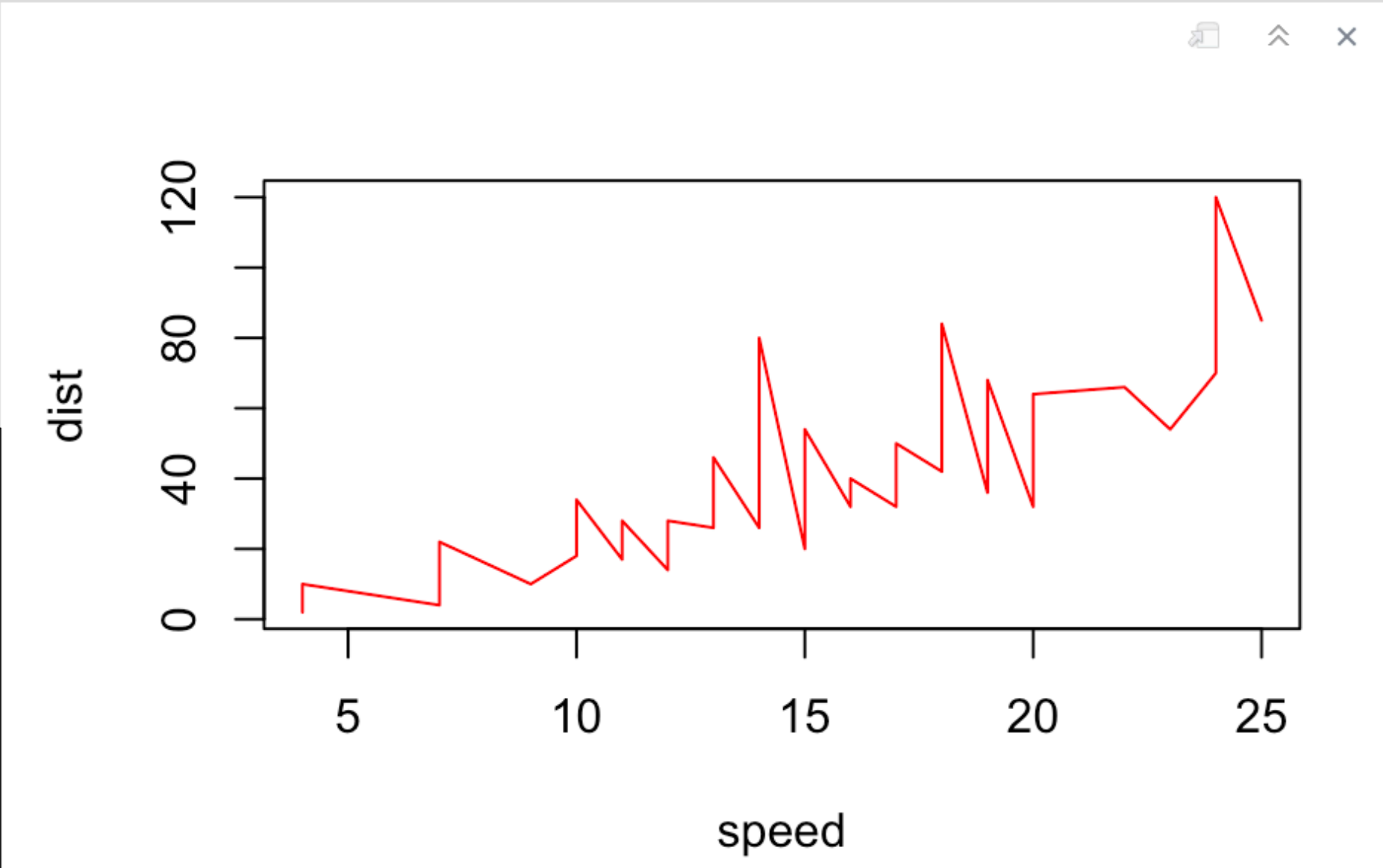
**type:** plot style (l, p, or b)

**col:** color of plot, must be character (most basic colors are valid)

**pch:** shape of points (integer/numeric)

|    |   |    |   |    |   |    |   |    |   |
|----|---|----|---|----|---|----|---|----|---|
| 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 | ▽ |    |   |    |   |    |   |    |   |

```
```{r, include=TRUE}  
plot(dist ~ speed, data = cars,  
      type="l", col = "red")  
```
```



# Basic Plotting: Formatting Plots

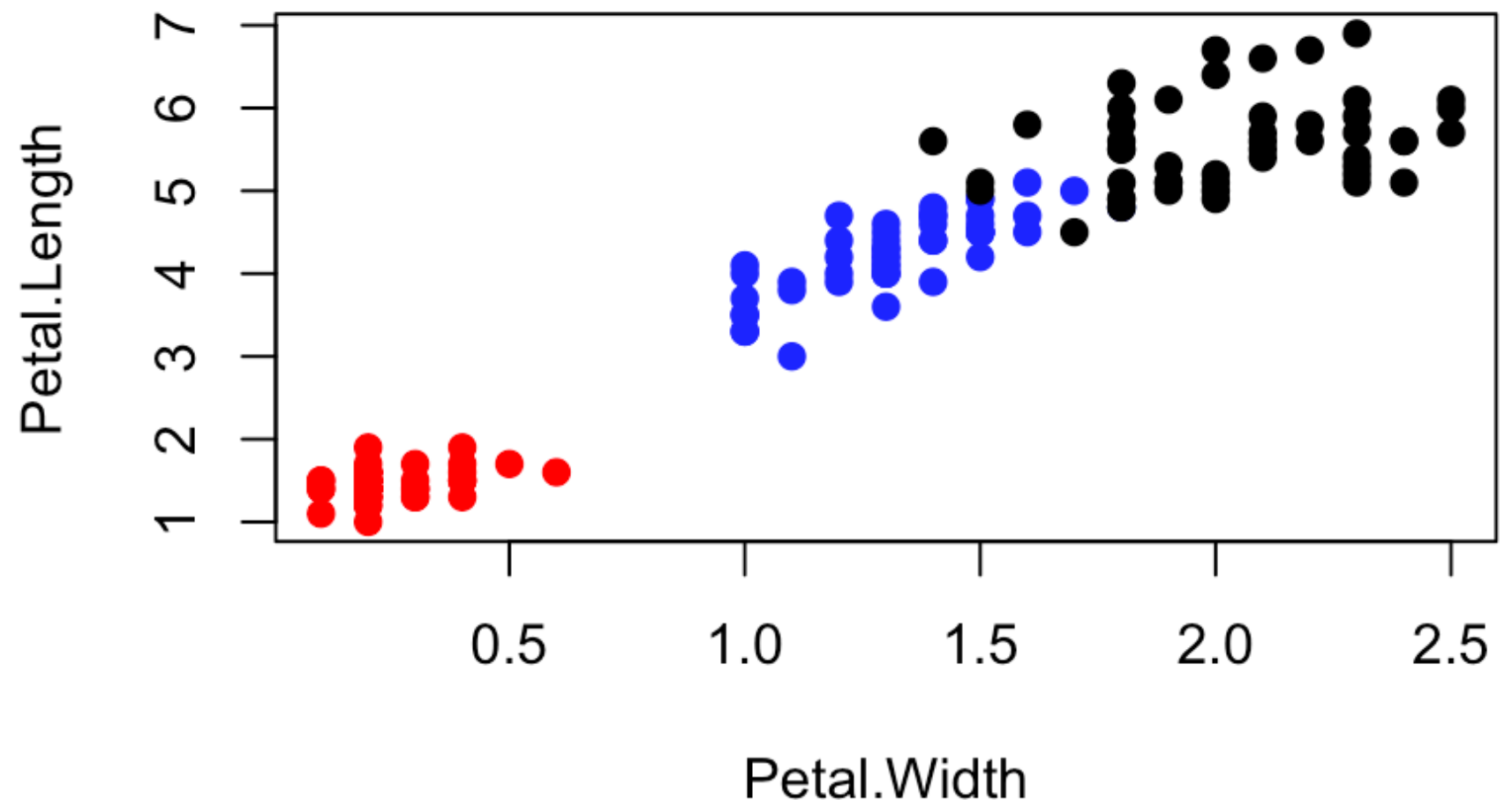
- Color can be assigned based on factors!

vector with colors,  
length = number of  
factors

```
``{r, include=TRUE}
iriscolors <- c("red", "blue", "black")
plot(Petal.Length ~ Petal.Width, data = iris,
     type="p", pch = 19, col = iriscolors[iris$Species])
``
```

Find number of levels:  
`nlevels(iris$Species)`

references color  
value based on  
factor number



# Check Your Understanding

**Make a plot of circumference versus age of the orange trees in the data set Orange. Give each tree a unique color *and* point shape.**

# Basic Plotting: Adding Things to Existing Plots

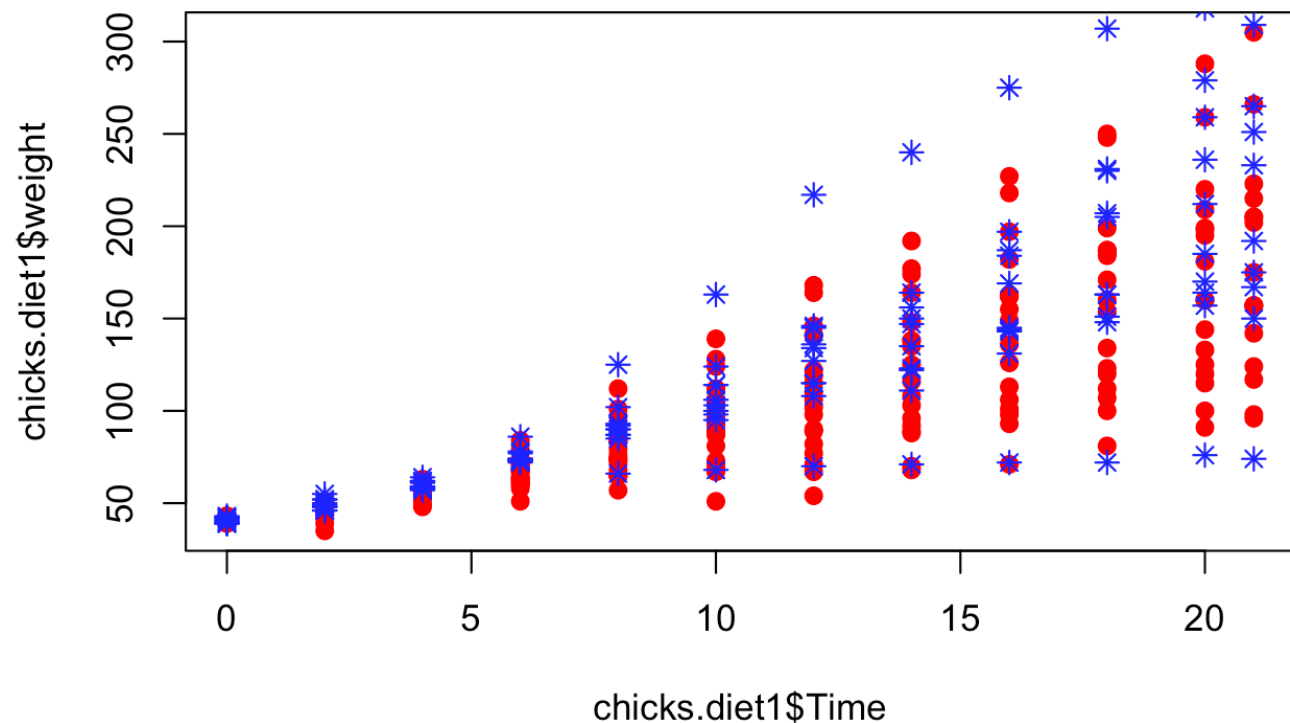
- You can add separate data sources to plots using `points()`, `lines()`, or `arrows()`.

**Breaking ChickWeight into two subsets, plotting one and then the other.**

```
```{r, include=TRUE}
data("ChickWeight")

chicks.diet1 <- subset(ChickWeight, ChickWeight$Diet == 1)
plot(x = chicks.diet1$Time, y = chicks.diet1$weight,
     col = "red", pch = 19)

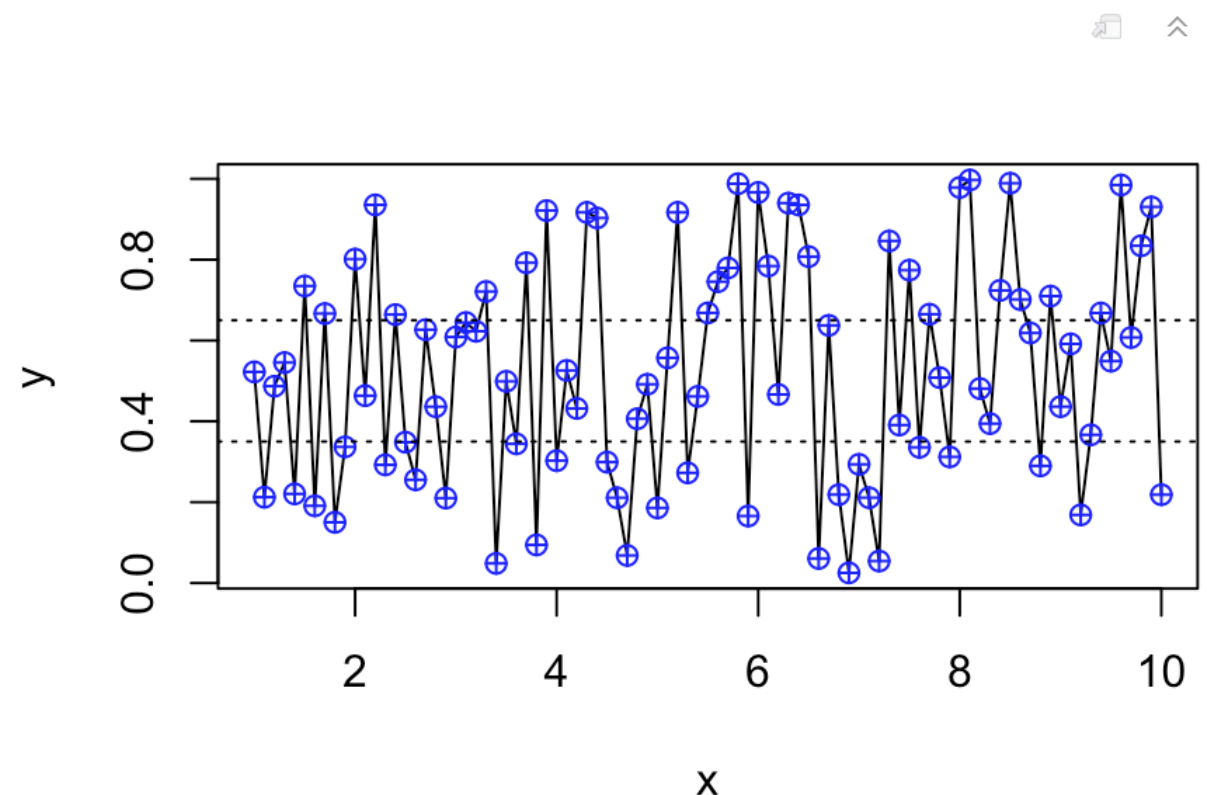
chicks.diet2 <- subset(ChickWeight, ChickWeight$Diet == 2)
points(x = chicks.diet2$Time, y = chicks.diet2$weight,
       col = "blue", pch = 8)
```



**Plotting y values with lines and points, adding bounding lines on the y axis.**

```
```{r, include=TRUE}
x.values <- seq(1,10, by=0.1)
y.values <- runif(length(x.values))
df2 <- data.frame(x = x.values, y = y.values)

plot(y ~ x, data = df2, type = "l")
points(x = df2$x, y = df2$y, pch = 10, col = "blue")
lines(x = c(-1,11), y = c(0.65, 0.65), lty = 3)
lines(x = c(-1,11), y = c(0.35, 0.35), lty = 3)
```
```



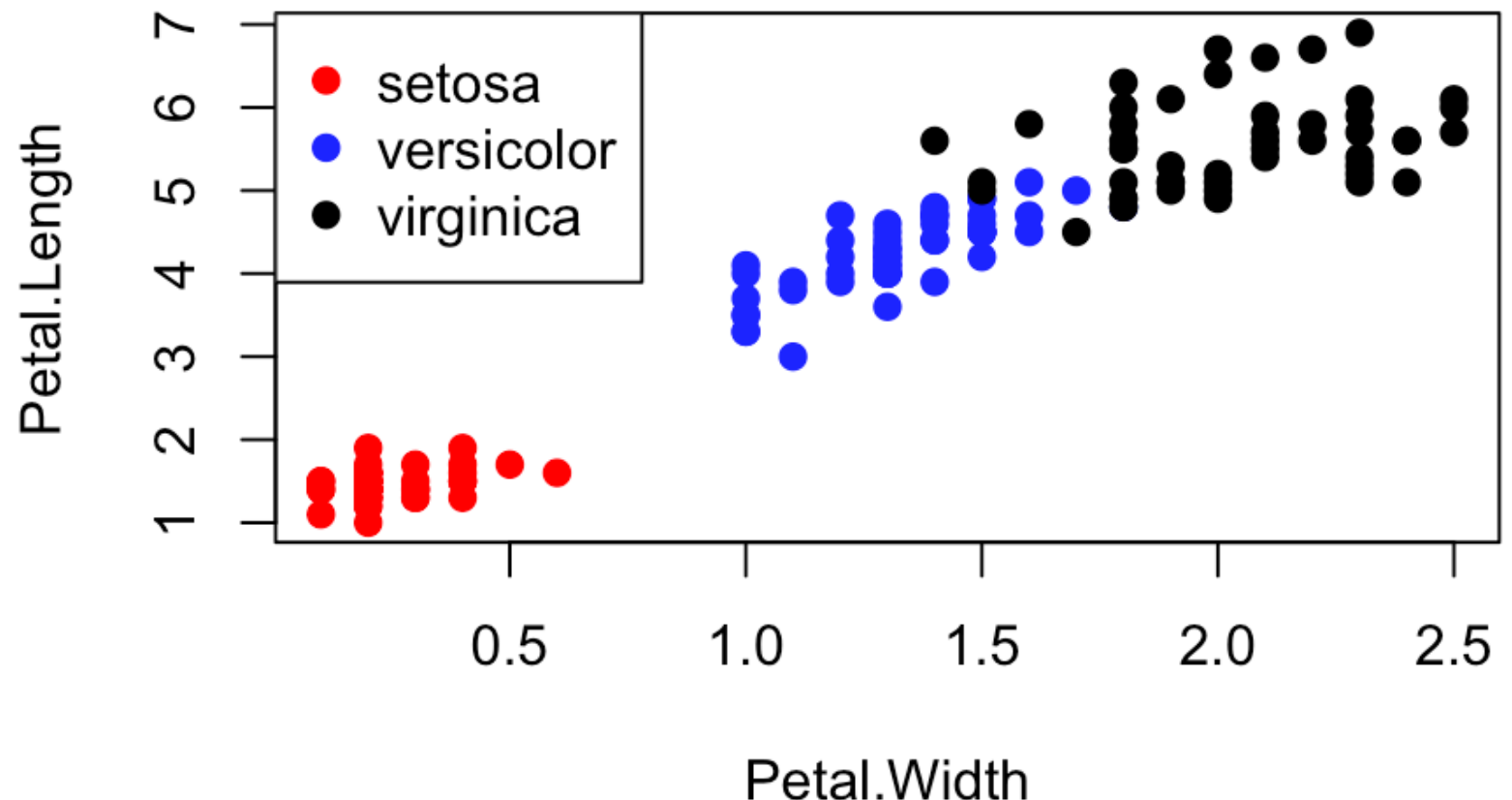
# Basic Plotting: Adding a Legend

- You can add a plot legend with `legend()`.

sets placement, options  
are topright, topleft,  
bottomright,  
bottomleft, or center

```
```{r, include=TRUE}  
iriscolors <- c("red", "blue", "black")  
plot(Petal.Length ~ Petal.Width, data = iris,  
     type="p", pch = 19, col = iriscolors[iris$Species])  
  
legend("topleft", legend=levels(iris$Species), col=iriscolors,  
      pch=rep(19,3))  
```
```

legend: sets text  
pch: shapes  
col: colors



# Check Your Understanding

**Add a legend to your graph of Orange trees.**

# Action Items

- 1. Complete Assignment 2.1 using R Markdown.**
- 2. Read Davies Chapter 8 and Chang Chapters 1-2 for next time.**