Graphics in R for Research and Publication

Presented @ SEPA 2019

Matthew D. Turner
Georgia State University
mturner46@gsu.edu

Setup for Workshop

- If you do not have R:
  - Copy the files from the USB stick
  - Install R, then RStudio, in that order, using the standard installer packages
  - Then, from RStudio, switch to the relevant directory and run the program `local_installer.R` we provide in the materials for today
- If you already have R and RStudio installed
  - If you are using a very old version of R, you may need to remove it and reinstall (see above)
  - Copy the files from the USB stick
  - Run the program `local_installer.R` from within RStudio
- Ask for help if you need it!

Aside:

- Setup at SEPA is always hard because we have to assume that we do not have access to the internet
- However, everywhere else installing packages is easy:

```
install.packages("tidyverse")
```

- When you need more functionality, just run the line above with the name of the library replacing the example (in the quotes!)

Presenters

- Matthew D. Turner
  - Research Scientist, GSU
  - My background is primarily in statistics, mathematics, and computing in science
    - MA, Applied Math
    - MS, Applied and Theoretical Statistics
    - MA, Social Science
    - PhD, Psychology
  - Psychological work: hearing (psycho-acoustics), meditation and anxiety, EEG recording for neurofeedback, moral injury, psychedelics, methodology
  - Computing work: scientific literature curation, machine learning, artificial intelligence, scientific applications
- Jessica A. Turner
  - Associate Professor of Psychology, GSU
  - My background is psychology, math, and neuroimaging
  - Research program in cognitive neuroscience in clinical populations
    - Schizophrenia
    - Huntington's Disease
  - Big data: neuroimaging/neuroinformatics
  - Combining imaging and genetics
  - Scientific literature curation

Goals of Today's Workshop

- Sketch out the graphical capabilities of R
- Summarize the process of loading and preprocessing data for making figures (introduction only!)
- Show how to make "quick" figures in R for day to day use (rough figures)
- Show how to improve figures for publication, including making annotations, changing formats and resolution, etc.
- Show how to export these improved figures in formats required by journals

Goals of Today's Workshop

- Sketch out the graphical capabilities of R
- Sum
- Sho
- Sho
- Show how to export these improved figures in formats required by journals

aking

Really, I just want to get you over the initial hurdles of using ggplot so that you can continue on your own!

Outline of Workshop

If you do NOT have a computer, it would be a good idea to pair up with someone who has one

- Introduction to R and ggplot
  - Slides
  - Demonstration
- Exercises with ggplot
  - Work on your own and with other attendees
  - Instructors will move around to help
- Publication graphics
- Additional plot examples (as time allows) including adding details like annotations, etc.

RStudio

- RStudio is the main way most people use R
  - It is an IDE – Integrated Development Environment
  - It has “panes” in its window that show different things you need to use to work with R
  - It is vastly better than the default interface with R (**NO ONE** uses the default interface for regular work!)
- The main parts of the window are shown on the following slide

~/GitHub/SEPA-2019-Workshops/Graphics_Workshop - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Graphics_Workshop

overview_of_ggplot.R | example_figures_for_psychological_rese... | scratchpad.R

Source on Save | Run | Source

```
26 #
27 # and this can give you a quick start on what you need, as well as
28 # giving you some new ideas.
29
30
31
32 ## Set up to use ggplot
33 #
34 # Remember that to use functions that are not part of "base R" you
35 # must first load the relevant libraries. You can do this at any point
36 # before using functions from the libraries, but it is considered good
37 # practice by many people to put them at the top of the code file.
38
39 set.seed(1001)     # Fixes the random numbers to be the same every time
40 library(ggplot2)   # Required to load the ggplot plotting functions
41
42
```

62:1 (Top Level) R Script

Console | Terminal

~/GitHub/SEPA-2019-Workshops/Graphics_Workshop/

```
4  2.22      SI2      J Very Good 10662
5  1.02     VVS1      E   Premium 11062
6  0.56      VS1      G Very Good  1912
> head(small_diamonds)  # Look at the first few rows of the data
  carat clarity color       cut price
1  0.71      SI1     E   Premium  2629
2  1.40      VS2     G     Ideal 10311
3  1.20     VVS1     F      Good 11182
4  2.22      SI2     J Very Good 10662
5  1.02     VVS1     E   Premium 11062
6  0.56      VS1     G Very Good  1912
> dim(small_diamonds)   # What is the size? (53940 rows and 10 variables)
[1] 5000    5
> plot(small_diamonds$carat, small_diamonds$price, pch = 16)
>
```

Environment | History | Connections

Import Dataset | List

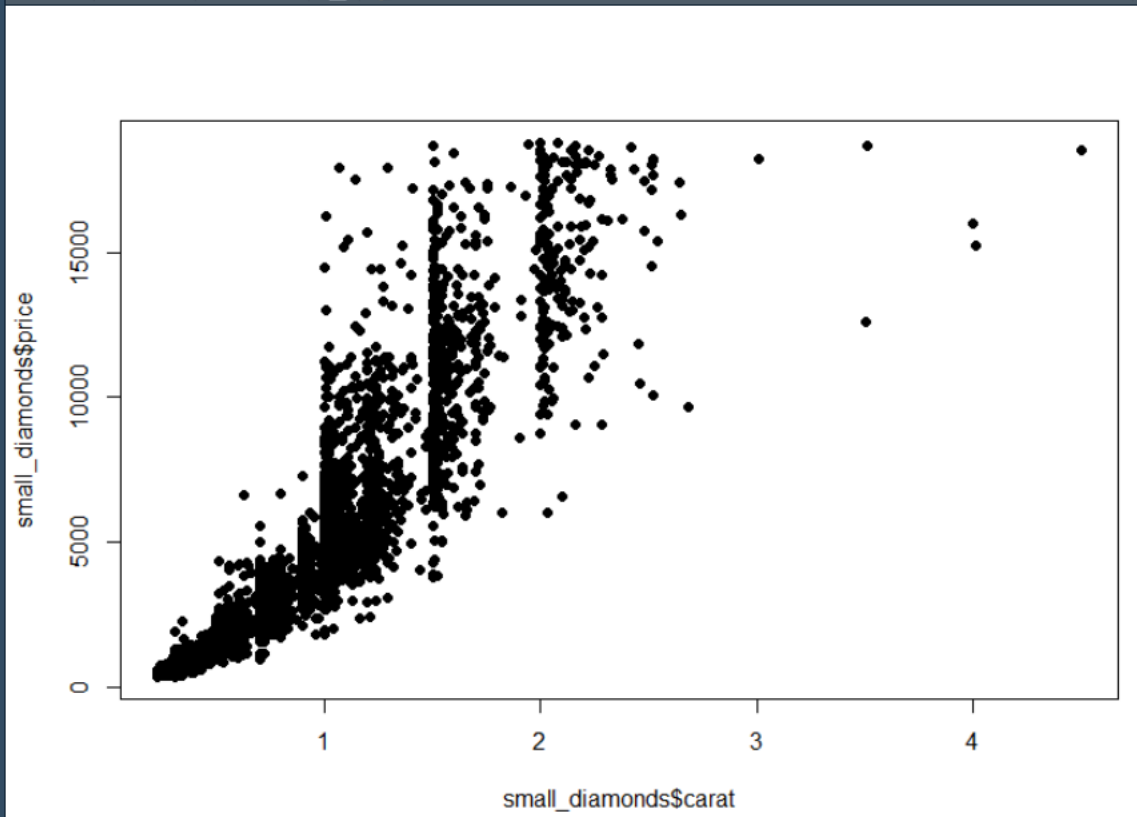
Global Environment

Data

d	86 obs. of 5 variables
pvc	List of 9
sm2	5000 obs. of 5 variables
small_diamonds	5000 obs. of 5 variables

Files | Plots | Packages | Help | Viewer

Zoom | Export | Publish



Type here to search

1:27 PM
3/7/2019

~/GitHub/SEPA-2019-Workshops/Graphics_Workshop - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Graphics_Workshop

overview_of_ggplot.R | example_figures_for_psychological_rese... | scratchpad.R

Source on Save | Run | Source

```
26 #
27 # and this can give you a quick start on what you need, as well as
28 # giving you some new ideas.
29
30
31
32 ## Set up to
33 #
34 # Remember t                    t of "base R" you
35 # must first                    do this at any point
36 # before usi                    it is considered good
37 # practice b                    of the code file.
38
39 set.seed(1001)     # Fixes the random numbers to be the same every time
40 library(ggplot2)   # Required to load the ggplot plotting functions
41
42
```

62:1 (Top Level) R Script

Editor: Your code usually goes into a file that you can edit here

Console | Terminal

~/GitHub/SEPA-2019-Workshops/Graphics_Workshop/

```
4  2.22      SI2      J Very Good 10662
5  1.02     VVS1      E   Premium 11062
6  0.56      VS1      G Very Good  1912
> head(small_diamonds)  # Look at the first few rows of the data
  carat clarity co
1  0.71      SI1
2  1.40      VS2
3  1.20     VVS1
4  2.22      SI2
5  1.02     VVS1
6  0.56      VS1
> dim(small_diamor                    and 10 variables)
[1] 5000    5
> plot(small_diamonds$carat, small_diamonds$price, pch = 16)
>
```

Console: This is where commands run and output appears

Environment | History | Connections

Import Dataset | List

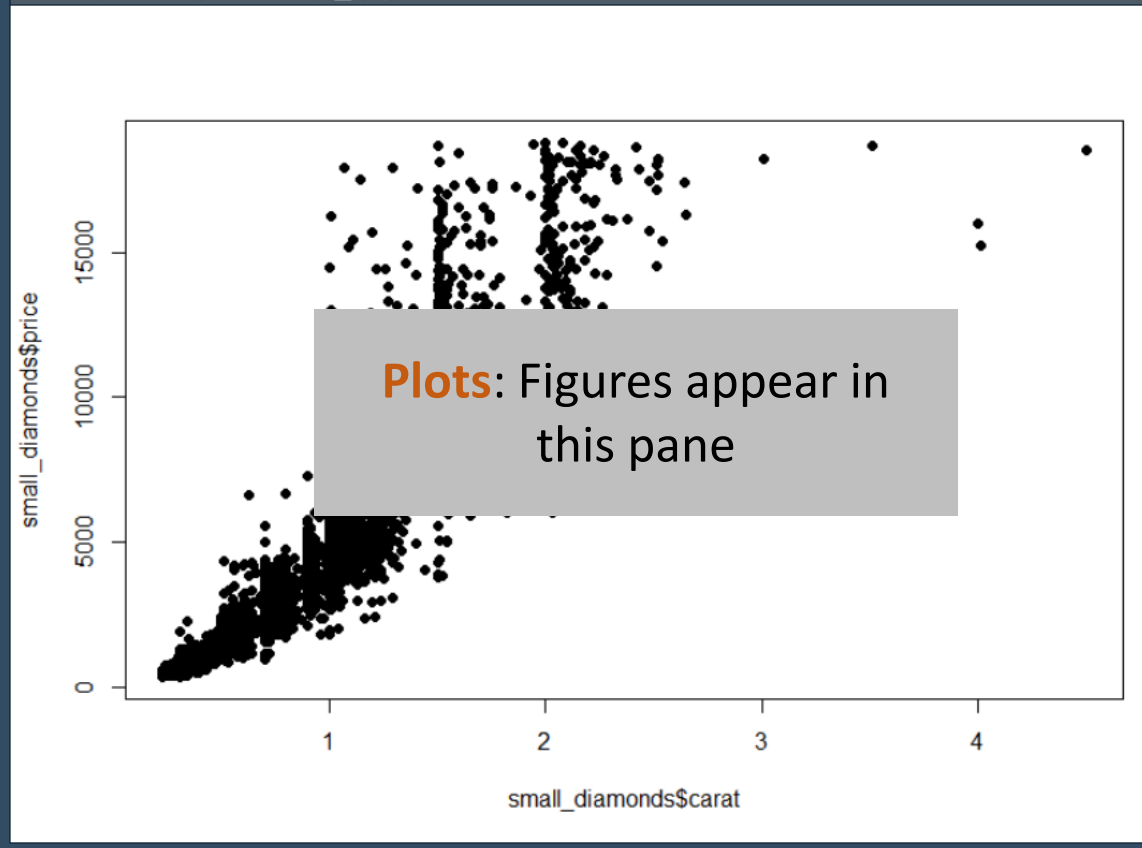
Global Environment

Data

d	86 obs. of 5 variables
pvc	List of 9
sm2	5000 obs. of 5 variables
small_diamonds	5000 obs. of 5 variables

Files | Plots | Packages | Help | Viewer

Zoom | Export | Publish



Plots: Figures appear in this pane

Type here to search

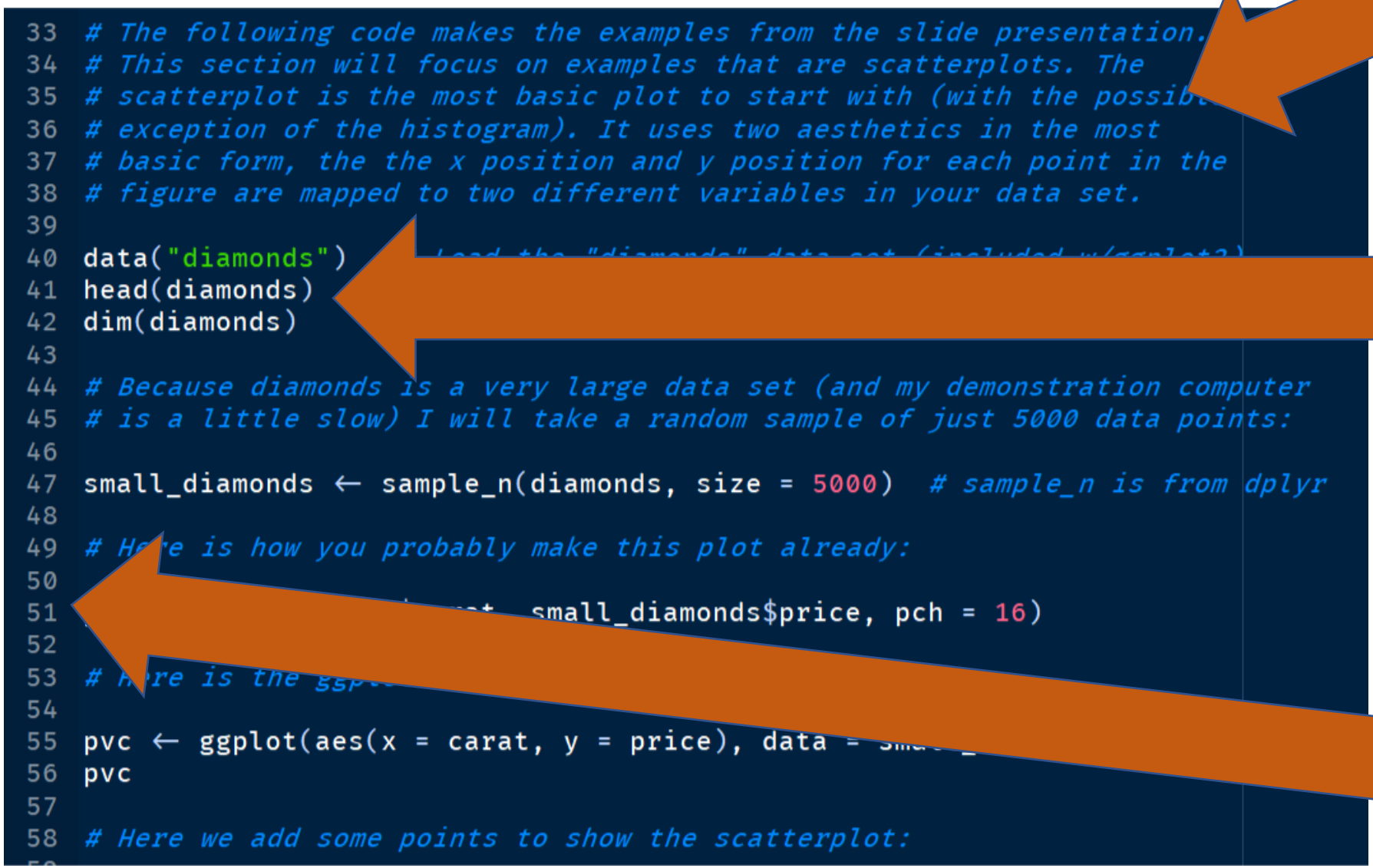
1:27 PM
3/7/2019

RStudio

- RStudio is the main way most people use R
  - It is an IDE – Integrated Development Environment
  - It has “panes” in its window that show different things you need to use to work with R
  - It is vastly better than the default interface with R (**NO ONE** uses the default interface for regular work!
- The main parts of the window are shown on the following slide
- To send a command from the editor to the console where it runs, put the cursor on the command and press **ctrl-enter** (or **command-enter** on Apple computers)

Format

- Today's workshop will be a series of demonstrations and exercises
  - The code files are **heavily annotated with comments** so that you can study them in detail on your own after the workshop (we will ignore a lot of this today)



Comments

Code/
Commands

Line
Numbers

Format

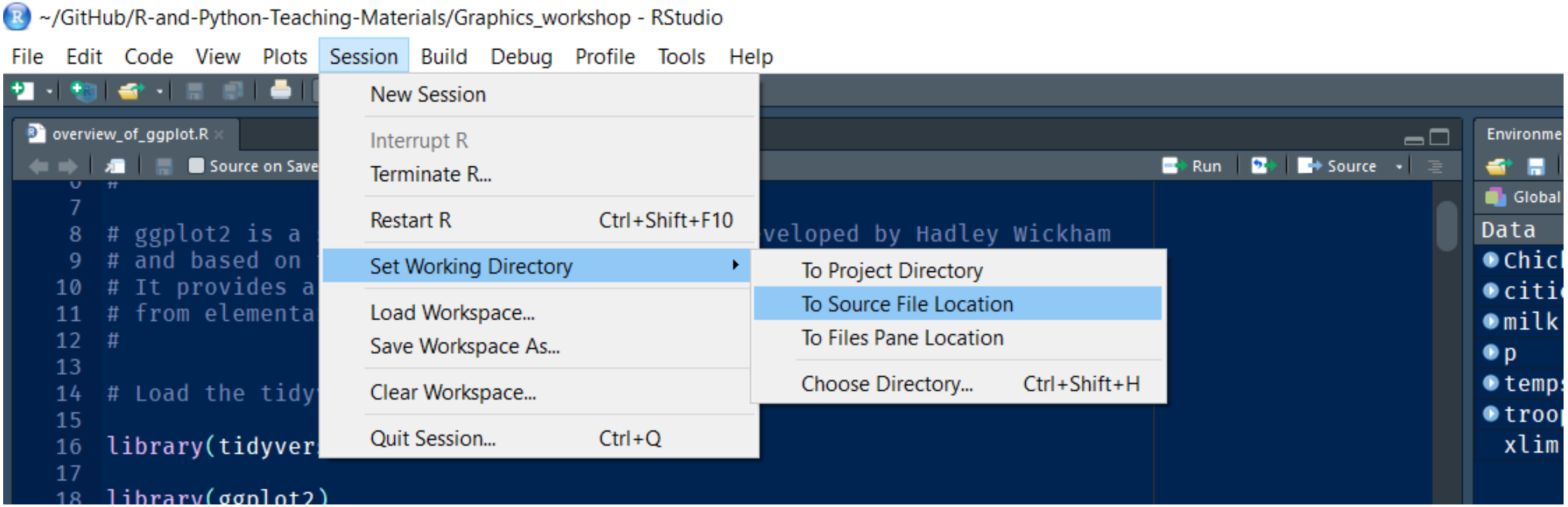
- Today's workshop will be a series of demonstrations and exercises
  - The code files are **heavily annotated with comments** so that you can study them in detail on your own after the workshop (we will ignore a lot of this today)
  - You can modify the files as you wish to (you should!!), but the line numbers will change if you add new lines of comments or code
  - Also the materials will be available online, so you can experiment with them and get new copies if you mess anything up while doing this

Getting things working: two important points

- Rstudio allows you to change its **working directory** to the location of the file you are using from the RStudio menu (at top)
  - You should do this for each of the files from the workshop today when you load a new one
  - Failure to do this may break the file, if that happens, just go to the menu change the working directory and try again

Getting things working: two important points

- Rstudio allows you to change its **working directory** to the location of the file you are using from the RStudio menu (at top)



Getting things working: two important points

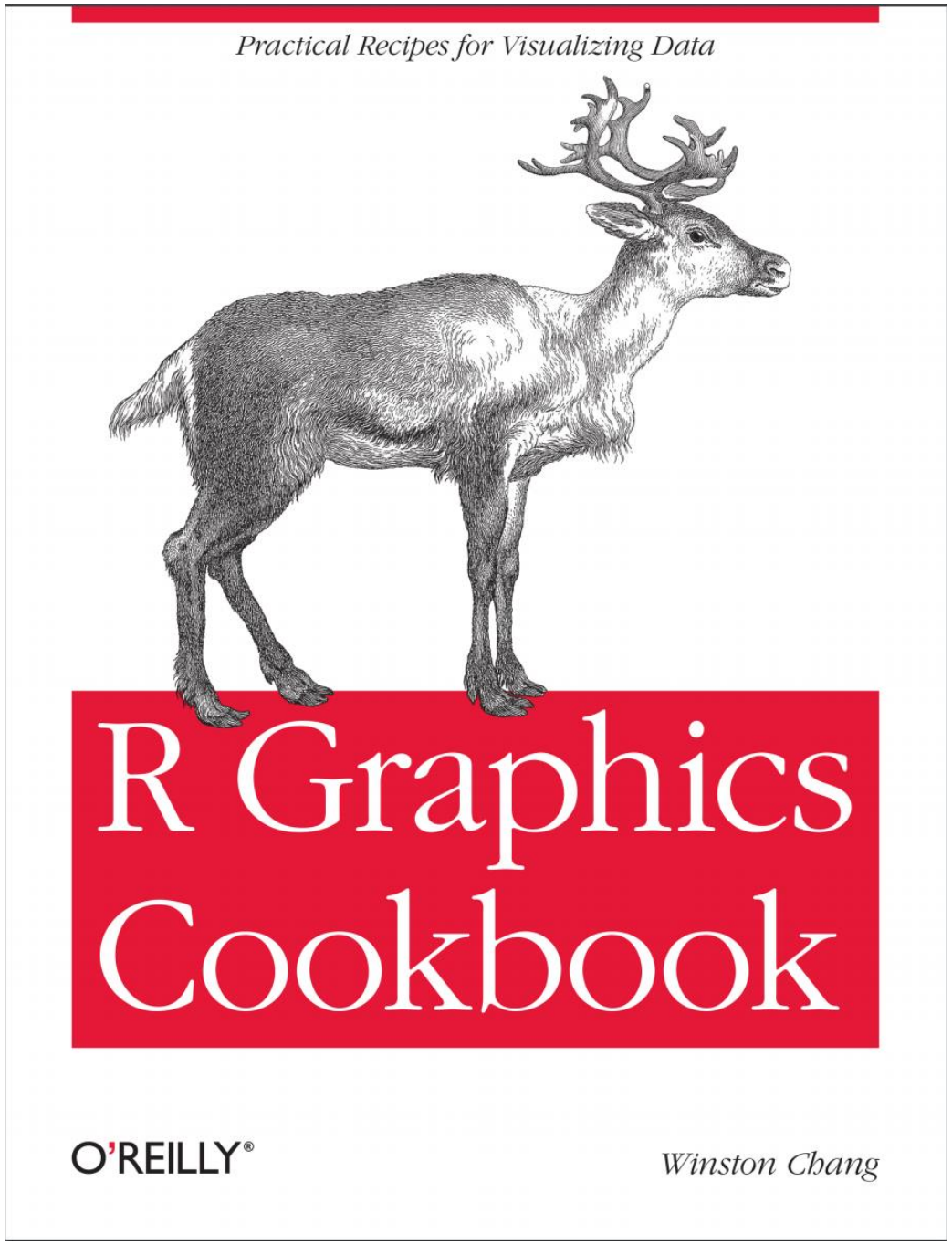
- Rstudio allows you to change its **working directory** to the location of the file you are using from the RStudio menu (at top)
  - You should do this for each of the files from the workshop today when you load a new one
  - Failure to do this may break the file, if that happens, just go to the menu change the working directory and try again
- You need to load the ggplot library before using it
  - This can be done in two ways. Use one of the following commands:
    - `library(ggplot2)`
- This will be done in the example files, but you have to **run the files in order** if you skip around you may need to go back and do this!

Additional Materials

- The materials provided today are more than we will have time to go through
- Notice especially:
  - The folder `Microsoft_Office_and_R` contains a PowerPoint and PDF showing some issues with MS Office products and R graphics
  - The folder `Extras` contains some PDF “cheat sheets” from Rstudio, Inc., who develop many of the tools we use (including some good ones we don’t have time to cover today!)
    - The file `rstudio-ide.pdf` is an overview of RStudio & its shortcut keys
    - Some of these cheat sheets require libraries we did not install due to limitations at SEPA

Sources and Additional Readings

- Some of the examples I will be using today come from a set of online notes from a course at IDRE at UCLA
  - The course has slides at: stats.idre.ucla.edu/stat/data/intro_ggplot2/ggplot2_intro_slidy.html
  - The full set of materials for the UCLA workshop is at: stats.idre.ucla.edu/r/seminars/ggplot2_intro/
- The presentation here will be somewhat simpler and assume less experience with R
- The UCLA workshop is a good follow-up for more information



Another good resource for ggplot and R graphics generally is Winston Chang's **R Graphics Cookbook** now available in a **2nd** edition.

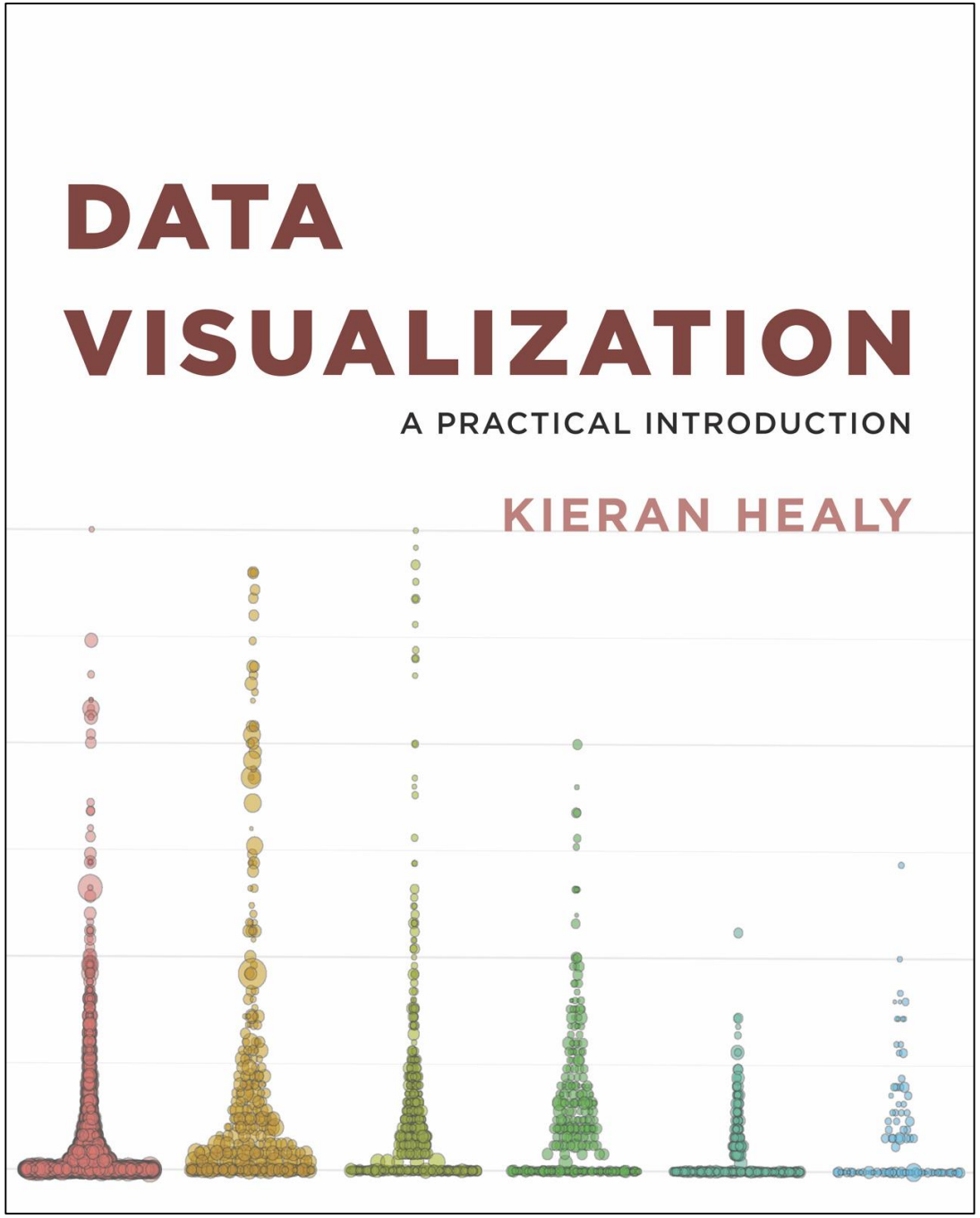
This book provides hundreds of examples of a wide variety of plots as well as some introduction to general use of ggplot, and a chapter on **data munging** (data cleaning/re-arrangement). All in the context of examples you can try.

If you buy just one book, make it this one!

Another book that is good is Healy's **Data Visualization: A Practical Introduction**.

In addition to teaching how to make sophisticated plots in R, it digs into the principles of good visualizations and graphic design at a level that is appropriate for people without any design background.

The examples in the book are presented in R using **`ggplot`**. The book also discusses other packages from the **Tidyverse** – a set of tools for quickly organizing, transforming and processing data.



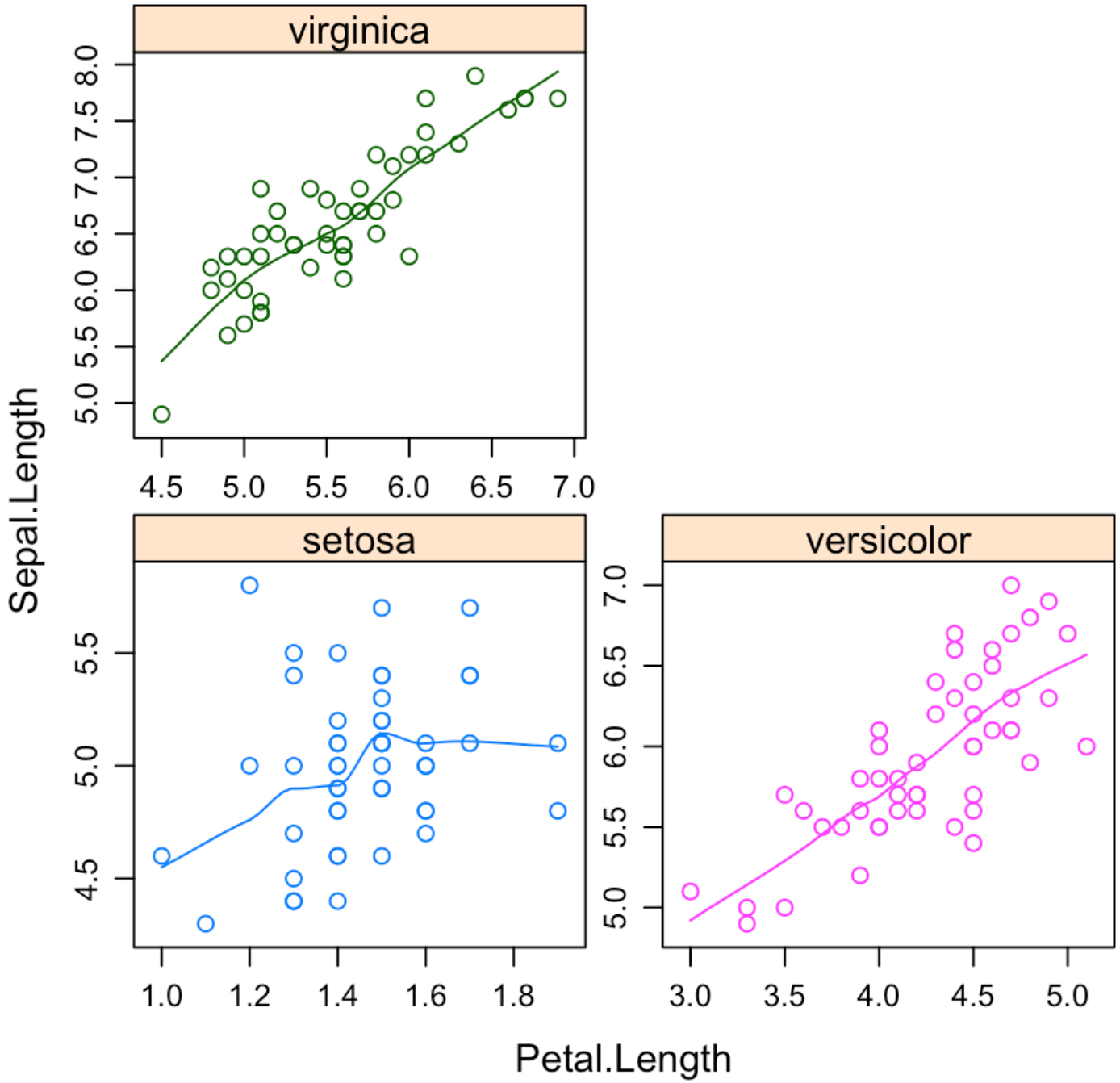
R Plotting Systems

R Plotting Systems

- R has three plotting systems:
  1. Base R Graphics
  2. Lattice Graphics
  3. **ggplot2**
- Most people start with the base R graphics system
  - I will assume you have used this a little bit
- The ggplot2 system is built on top of it
- We'll focus on the ggplot2 system today because it is the easiest way to get good quality figures easily for a wide variety of types of data

R Plotting Systems

- Base R graphics examples will be mixed in with the ggplot2 examples as we go
  - They mostly use the **`plot()`** function and a lot of specialized functions like **`par()`**, **`pairs()`**, and many of the default graphics that appear with other functions
- Lattice graphics:
  - Based on an older system called trellis graphics
  - Good for multivariate work: it can show figures that relate variables and these can be conditioned on other variables
  - See: www.statmethods.net/advgraphs/trellis.html for more!



Example of a **lattice** graph using conditioning.

It shows the relationship between petal length and sepal length conditioned on species of plant.

This was a big deal when it was introduced, but ggplot can do this sort of thing, too.

From:
sthda.com/english/wiki/lattice-graphs

Theory of ggplot

Some Brief History

- Leland Wilkinson developed a system for building graphic displays called the “grammar of graphics”
  - The idea was to develop a general language for describing plots of data
  - It allows more arbitrary mappings of data to the aesthetic aspects of pictures like size, color, position, etc.

- Hadley Wickham substantially extended and modified this system and implemented it in the R language
  - **`ggplot1`** was a prototype version with very different syntax
  - **`ggplot2`** is essentially the first (and only) version ever available

- This system is being continuously developed by many people

Syntax: Overview

- **`ggplot2`** builds graphics in **layers** that are specified in *ascending* order
  - You build the plots from items in the **background** to the **foreground**

- Each layer specifies **aesthetic mappings**
  - More in a moment!
  - If these are not given, then they are inherited from the previous layer

- Finally, the physical parts of the plot are specified by "**geoms**"

Syntax: Aesthetics

- Aesthetics are aspects of the plot that can be used to convey information
- Basically it is a list of physical aspects of the plot that **could** carry information
- These are then mapped to variables of interest

Some example aesthetics:

- `x`: positioning along x-axis
- `y`: positioning along y-axis
- `color`: color of objects; for 2-d objects, the color of the object's outline (compare to fill below)
- `fill`: fill color of objects
- `alpha`: transparency of objects (value between 0, transparent, and 1, opaque – inverse of how many stacked objects it will take to be opaque)
- `linetype`: how lines should be drawn (solid, dashed, dotted, etc.)
- `shape`: shape of markers in scatter plots
- `size`: how large objects appear

From: https://stats.idre.ucla.edu/r/seminars/ggplot2_intro/

Syntax: Geoms

- Geoms are standard plot elements that we are used to using for displaying data
- Each geom function accepts a subset of the aesthetics available

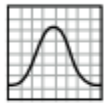
Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

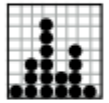
Continuous

a <- ggplot(mpg, aes(hwy))

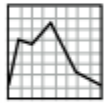
a + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")



a + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))



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



a + geom_freqpoly()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + geom_histogram(binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

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

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

Graphical Primitives

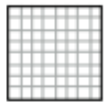
c <- ggplot(map, aes(long, lat))

c + geom_polygon(aes(group = group)**)**

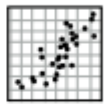
Two Variables

Continuous X, Continuous Y

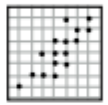
f <- ggplot(mpg, aes(cty, hwy))



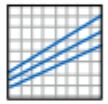
f + geom_blank()



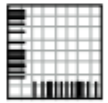
f + geom_jitter()
x, y, alpha, color, fill, shape, size



f + geom_point()
x, y, alpha, color, fill, shape, size

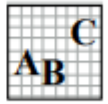


f + geom_quantile()
x, y, alpha, color, linetype, size, weight



f + geom_rug(sides = "bl"**)**
alpha, color, linetype, size

f + geom_smooth(model = lm**)**
x, y, alpha, color, fill, linetype, size, weight



f + geom_text(aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

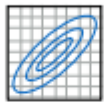
g <- ggplot(mpg, aes(class, hwy))

g + geom_bar(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + geom_bin2d(binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight



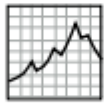
i + geom_density2d()
x, y, alpha, colour, linetype, size

i + geom_hex()
x, y, alpha, colour, fill size

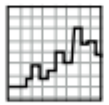
Continuous Function

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

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



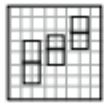
j + geom_line()
x, y, alpha, color, linetype, size



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

Visualizing error

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



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

k + geom_errorbar()

From the ggplot2 cheat sheet (included with workshop materials). From: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Notice what physical features of a plot geom_point() makes available for data…

m's aesthetic properties to represent variables. Each function returns a layer.

Two Variables

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d(**binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Function

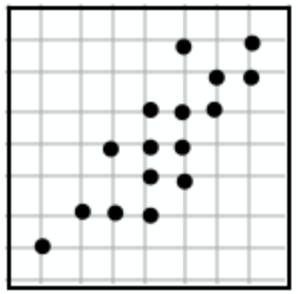
j <- ggplot(economics, aes(date, unemploy))

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight



f + geom_point()

x, y, alpha, color, fill, shape, size

a + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + **geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + **geom_dotplot()**
x, y, alpha, color, fill

a + **geom_freqpoly()**
x, y, alpha, color, linetype, si
b + geom_freqpoly(aes(y = ..

a + **geom_histogram(**binw
x, y, alpha, color, fill, linetype
b + geom_histogram(aes(y =

Discrete

b <- ggplot(mpg, aes(

b + **geom_bar()**
x, alpha, color, fill, linetype,

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar(stat = "identity")**
x, y, alpha, color, fill, linetype, size, weight

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

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

k + **geom_errorbar()**

Graphical Primitives

c <- ggplot(map, aes(long, lat))

c + **geom_polygon(**aes(group = group)**)**

From the ggplot2 cheat sheet (included with workshop materials). From: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Syntax

- All of your data needs to be in a single data frame for simple plots

- You may need to restructure your data a bit to make things work
  - Much of the **`tidyverse`** collection of R packages were made to make this easier
  - We don't have time to touch on the tidyverse today, but if you are interested there is a lot of information available online (it is **very** powerful!)

Moving on to example files

- From here on out we will be working in R files
  - You can follow along or watch me for the demonstrations
  - I recommend following along!
  - After some demonstrations we will move on to exercises for you to try
  - Then we will discuss more advanced figures and publication quality output