

Instructions to Download R, RStudio, and Packages

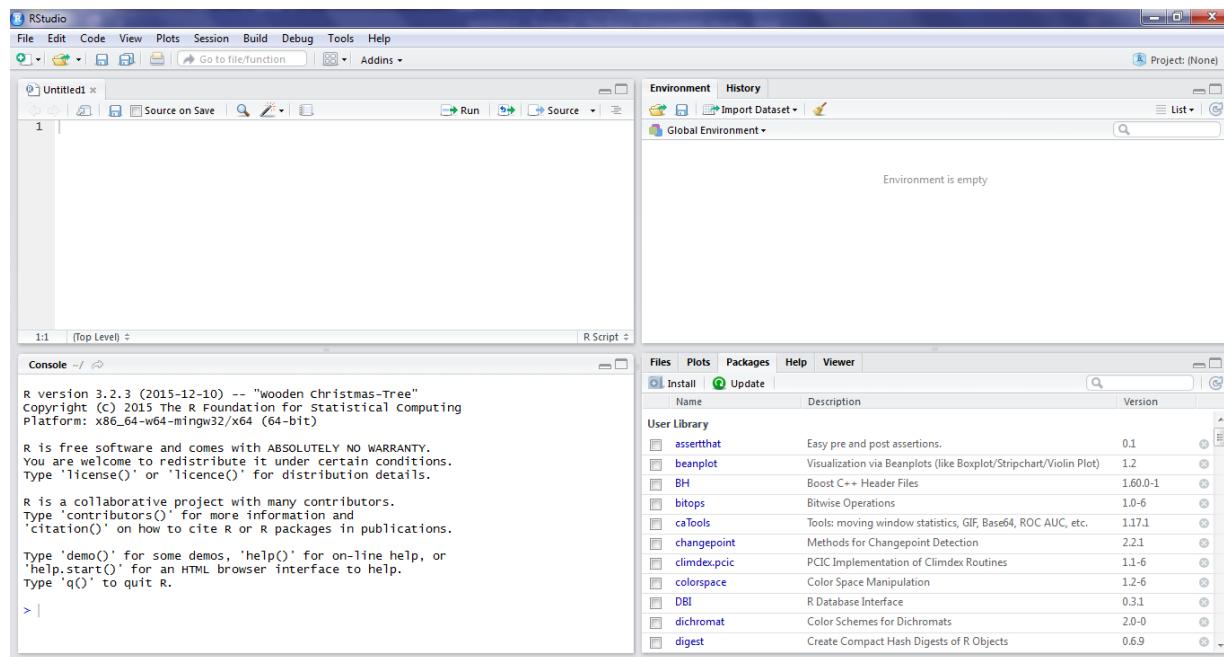
A) You must download base R first, since it must be installed on your computer for RStudio to work.

- 1) Go to this webpage: <https://cloud.r-project.org/>
- 2) Under “Download and Install R,” click the link that corresponds to your computer’s operating system.
- 3) FOR WINDOWS:
 - a) In the first line (after “base”), click the link “install R for the first time”.
 - b) Click “Download R 4 ...” at the top of the page.
- 4) FOR MAC:
 - a) Under “Latest Release”, click on the first file name that ends in “.pkg”.
- 5) After downloading, proceed as you normally would to complete installation of a program.

B) Next, install RStudio – this requires base R to already be installed on your computer.

- 1) Go to this webpage: <https://www.rstudio.com/products/rstudio/download/#download>
- 2) Under “All Installers,” click the link that corresponds to your computer’s operating system.
- 3) After downloading, proceed as you normally would to complete installation of a program.

C) I recommend opening RStudio after installation is complete to ensure it runs correctly. You should see a screen that looks something like this:



D) Install four packages in RStudio that we will use during the workshop.

- 1) Open RStudio.
- 2) Click the “Tools” toolbar at the top of the screen.

- 3) Click “Install Packages...”.
- 4) Make sure “Repository (CRAN)” is selected in the “Install from.” box.
- 5) Type the names of packages to install in the “Packages” box:
`ggplot2, ggpublisher`
- 6) Make sure “Install dependencies” is checked.
- 7) Click “Install”.
- 8) If successfully installed, you should see something like this in the “Console” panel:



The screenshot shows the RStudio Console window with the title "Console G:/My Drive/rachel-PC/Miami-OH/". The console output displays the installation of several R packages from CRAN. The packages listed are `apiyr_1.0.0`, `tidyverse_1.1.0`, `ggplot2_3.3.2`, `lubridate_1.7.9`, and `rlang`, `tidyselect`, `vctrs`, `dplyr`, `tidyverse`, `ggplot2`, and `lubridate` from the `base` repository. The output shows the URL tried, content type, length, and download time for each package. After the packages are unpacked, the MD5 sums are checked. Finally, the path where the downloaded binary packages are stored is printed: `C:\Users\rache\AppData\Local\Temp\Rtmp63z4Ek\downloaded_packages`.

```
Console G:/My Drive/rachel-PC/Miami-OH/
trying URL https://cran.rstudio.com/bin/windows/contrib/4.0/apiyr_1.0.0.zip
Content type 'application/zip' length 1304095 bytes (1.2 MB)
downloaded 1.2 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/tidyverse_1.1.0.zip'
Content type 'application/zip' length 1514564 bytes (1.4 MB)
downloaded 1.4 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/ggplot2_3.3.2.zip'
Content type 'application/zip' length 4066720 bytes (3.9 MB)
downloaded 3.9 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/lubridate_1.7.9.zip'
Content type 'application/zip' length 1748550 bytes (1.7 MB)
downloaded 1.7 MB

package 'rlang' successfully unpacked and MD5 sums checked
package 'tidyselect' successfully unpacked and MD5 sums checked
package 'vctrs' successfully unpacked and MD5 sums checked
package 'dplyr' successfully unpacked and MD5 sums checked
package 'tidyverse' successfully unpacked and MD5 sums checked
package 'ggplot2' successfully unpacked and MD5 sums checked
package 'lubridate' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\rache\AppData\Local\Temp\Rtmp63z4Ek\downloaded_packages
> |
```

*If you have any problems during the installation,
please email me (pillarm@miamioh.edu) and I'll do my best to help you out!*

Computer Science in Modern Biology

Data Visualization in R

Instructor:

Rachel Pilla

Mahbobe Lesani

TAs:

Mason Murphy



1

Workshop Schedule

Day 1 (Wednesday):

- Structure and design of graphs
- Creating graphs in base R
- Overview of ggplot2

Day 2 (Thursday):

- Introduction to ggplot2
- Building graphs and customizing in ggplot2
- Additional graphing resources
- Q&A and practice

2

Workshop Logistics

Zoom

- Feel free to unmute yourself to ask questions at any point

Chat

- Post any questions or comments here
- The TAs will be checking the chat to help answer questions

Break-Out Rooms

- If you are really stuck, you can move to a break-out room with a TA to share your screen and get you up to speed
- Break time is a great opportunity for this if needed!

3

Packages in R

- **Packages** are bundles of tools and functions that others have developed to be used in R
- They are often grouped to specific types of functions, analyses, or datasets
- `rLakeAnalyzer`, for example, has lots of functions developed by limnologists to help you analyze common types of data collected from lakes

4

Packages in R

- Currently, there are over 16,000 packages!
- We are going to install 4 packages to use today:
 - 1) ggplot2
 - 2) ggpublisher

5

Installing a Package

- Tools → Install Packages...

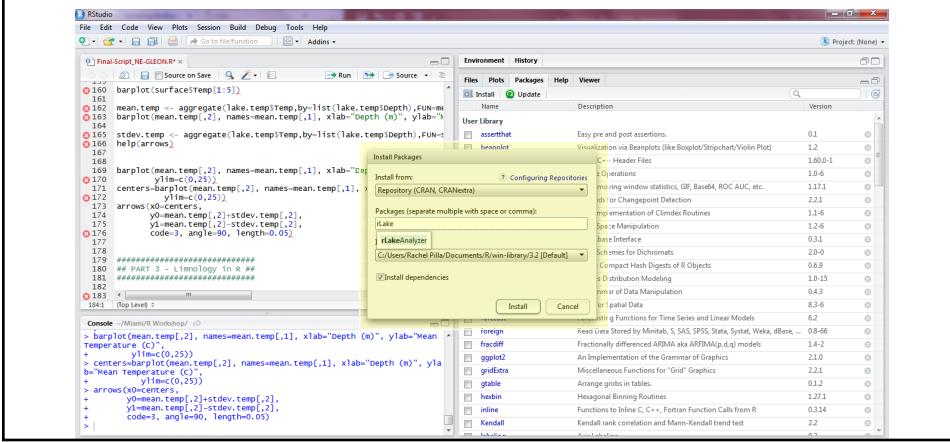
The screenshot shows the RStudio interface with the 'Tools' menu open, specifically the 'Install Packages...' option under the 'Tools' menu. The 'Packages' tab is selected in the sidebar, displaying a list of packages. The list includes:

Name	Description	Version
assertthat	Easy pre and post assertions.	0.1
beanplot	Visualization via Beanplots (like Boxplot/Stripchart/Violin Plot)	1.2
BH	Boost C++ Header Files	1.60.0-1
bitops	Bitwise Operations	1.0-6
caTools	Tools moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1
changepoint	Methods for Changepoint Detection	2.21
cldmde.pcic	PCIC Implementation of Cldmde Routines	1.1-6
colospace	Color Space Manipulation	1.2-2
DBI	Database Interface	0.3.1
dchexmet	Color Scheme for Dchexmet	2.0-0
digest	Create Compact Hash Digests of R Objects	0.6.9
dmo	Species Distribution Modeling	1.0-15
dplyr	A Grammar of Data Manipulation	0.4.3
fields	Tools for Spatial Data	8.3-6
forecast	Forecasting Functions for Time Series and Linear Models	6.2
foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, JSS, ...	0.8-66
frcdiff	Fractionally differenced ARIMA aka ARIMA(p,d,q) models	1.4-2
ggplot2	An Implementation of the Grammar of Graphics	2.1.0
gridExtra	Miscellaneous Functions for "Grid" Graphics	2.2.1
gtable	Arrange grobs in tables.	0.1.2
hexbin	Hexagonal Binning Routines	1.27.1
inline	Functions to Inline C, C++, Fortran Function Calls from R	0.3.14
Kendall	Kendall rank correlation and Mann-Kendall trend test	2.2

6

Installing a Package

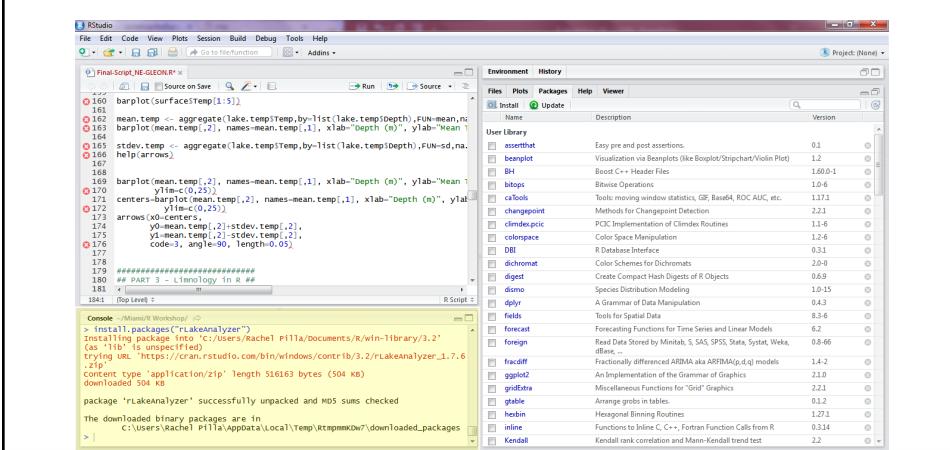
- Type the packages you want to install:
ggplot2, ggpublisher



7

Installing a Package

- Console will tell you when they have been installed



8

Installing a Package

- Packages only need to be installed ONCE
- But, each time you re-open R, you need to load the package(s) you want to use so R can make all the functions available:

```
library(package name)
```

 Hadley Wickham ✅
@hadleywickham

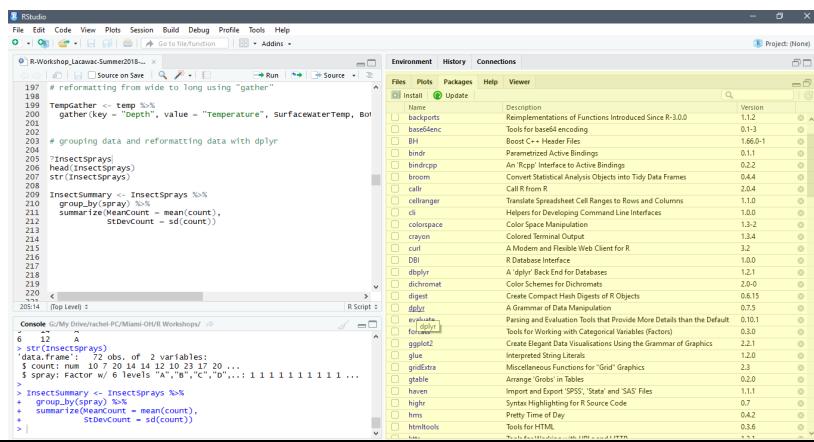
Replying to @ijlytle
@ijlytle a package is a like a book, a library is like a library; you use library() to check a package out of the library #rsats

8:34 AM · Dec 8, 2014 · Echofon

9

Package Information

- Under the “Packages” tab, click on **dplyr**



The screenshot shows the RStudio interface with the 'Packages' tab selected in the top navigation bar. The search bar at the top has 'dplyr' typed into it. Below the search bar, a list of packages is displayed, with 'dplyr' highlighted. The list includes various packages such as backports, base64enc, bindrcpp, broom, callr, cellranger, DBI, dplyr, dichromat, digest, evaluate, gridExtra, ggplot2, grid, gtable, gridSVG, gtable, haven, highr, hms, and htmltools.

10

Package Information

- This will list all the available functions
 - Clicking on a function takes you to the help file

The screenshot shows the RStudio interface with the following details:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Code Editor:** A script named "R-Workshop.Latwacv.Summer2018.R" containing R code for data manipulation.
- Environment Tab:** Shows objects like TempGather, InsectSprays, InsectSummary, and dplyr.
- Help Tab:** Documentation for the 'dplyr' package version 0.7.5, including sections for DESCRIPTION, User guides, vignettes, and other documentation.
- Help Pages:** A list of functions including add_count, add_tally, all_equal, and all_true.
- Project Tab:** Project: (None).

11

Structure & Design of Graphs

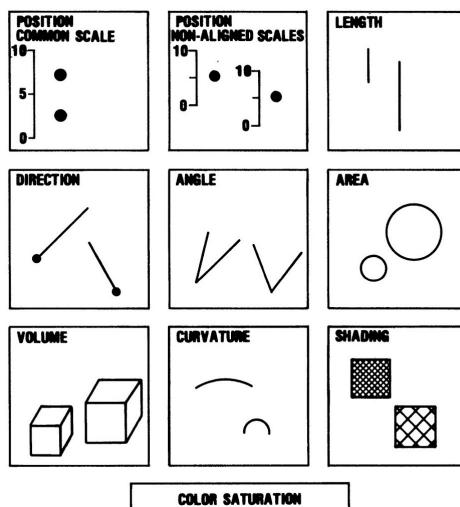
12

Components of a Graph

- 1) Axes
 - Horizontal x-axis and vertical y-axis
- 2) Labels
 - What are your axes measuring?
- 3) Title and/or caption
 - Description and information about the graph
- 4) Legend (sometimes)
 - If multiple lines, colors, shapes, etc., what does each represent?
- 5) DATA
 - The actual information!

13

Understanding a Graph

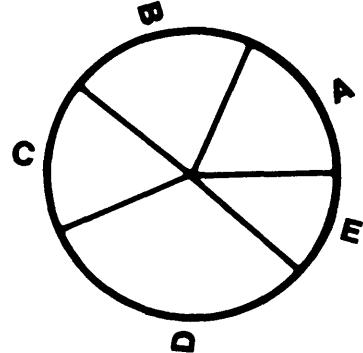


- Easy to perceive and understand
- ↓
- May require more work to understand
 - Can be very valuable when used appropriately in graphs

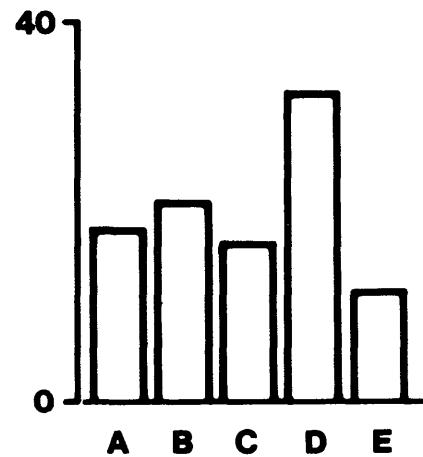
Cleveland & McGill, 1984 (*Journal of the American Statistical Association*)

14

Understanding a Graph



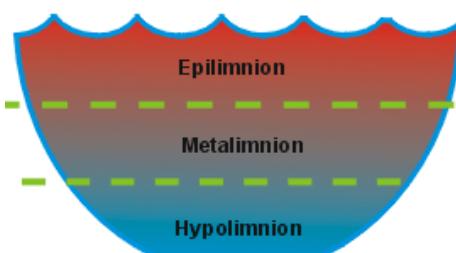
Cleveland & McGill, 1984 (*Journal of the American Statistical Association*)



15

Common Types of Graphs

- Lake Lacawac (Pennsylvania)
- Water temperature distribution

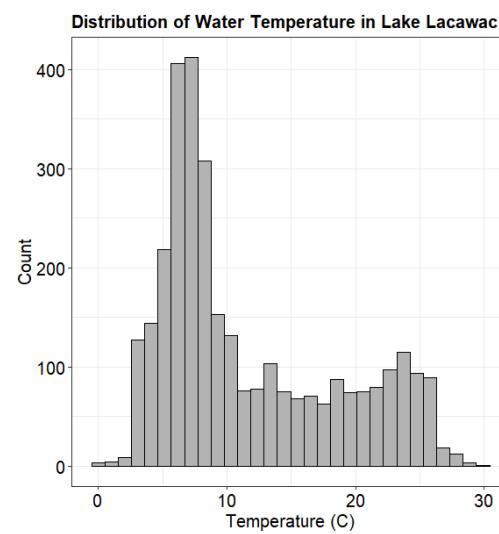


16

Common Types of Graphs

One variable:

- Temperature of Lake Lacawac
- Histogram

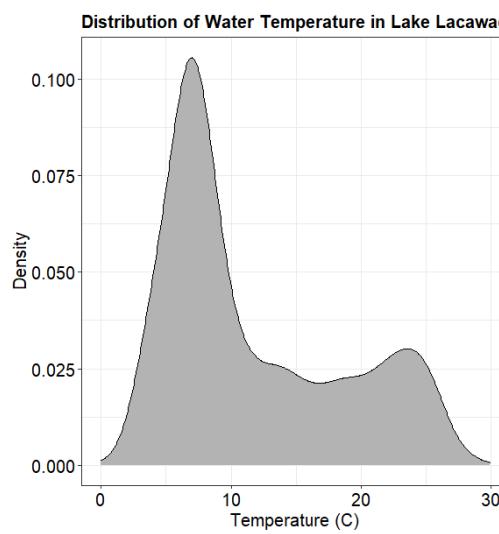


17

Common Types of Graphs

One variable:

- Temperature of Lake Lacawac
- Density plot

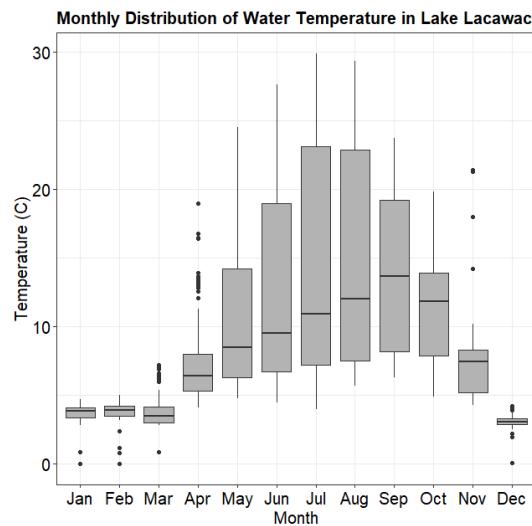


18

Common Types of Graphs

Two variables:

- Temperature of Lake Lacawac for each month of the year
- Box plot

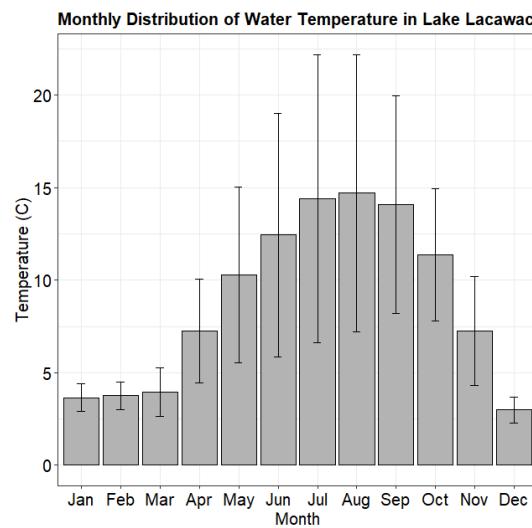


19

Common Types of Graphs

Two variables:

- Temperature of Lake Lacawac for each month of the year
- Bar plot



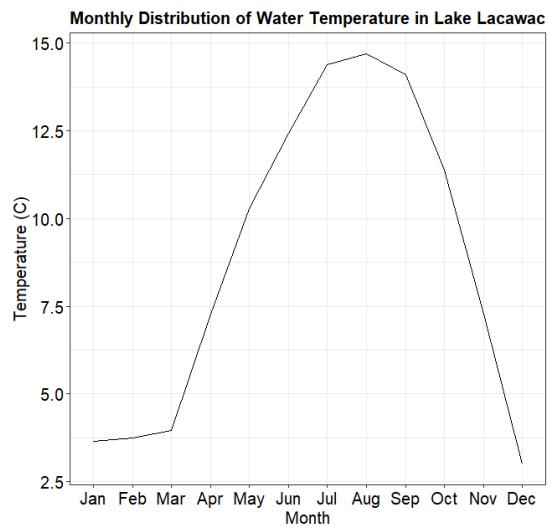
20

Common Types of Graphs

Two variables:

- Temperature of Lake Lacawac for each month of the year

- Line plot



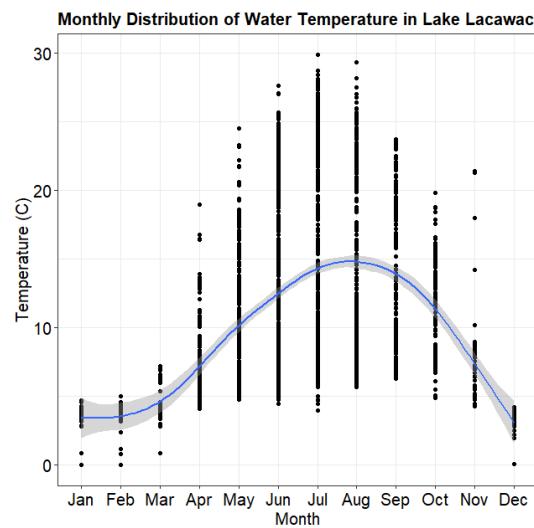
21

Common Types of Graphs

Two variables:

- Temperature of Lake Lacawac for each month of the year

- X-Y scatterplot

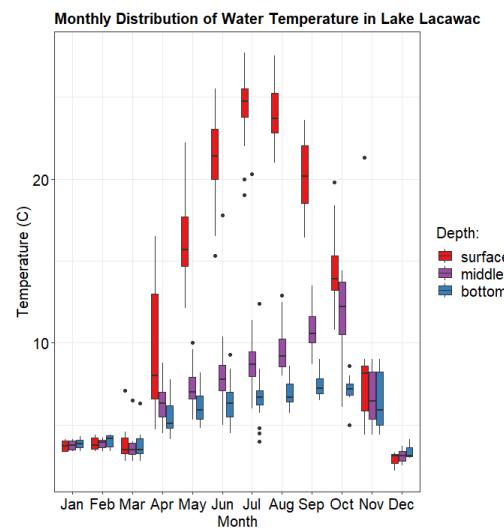


22

Common Types of Graphs

Three⁺ variables:

- Temperature of Lake Lacawac at multiple depths for each month of the year
- Box plot

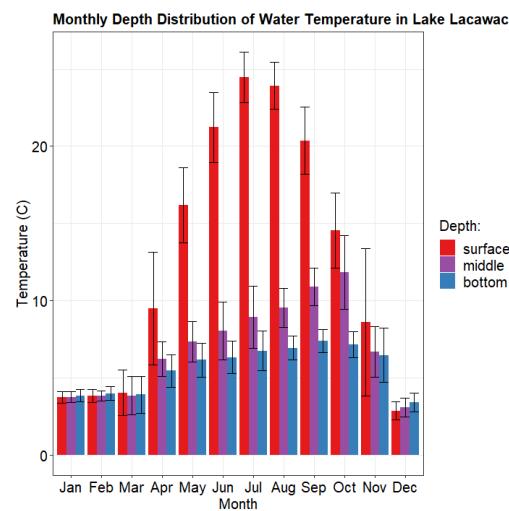


23

Common Types of Graphs

Three⁺ variables:

- Temperature of Lake Lacawac at multiple depths for each month of the year
- Bar plot



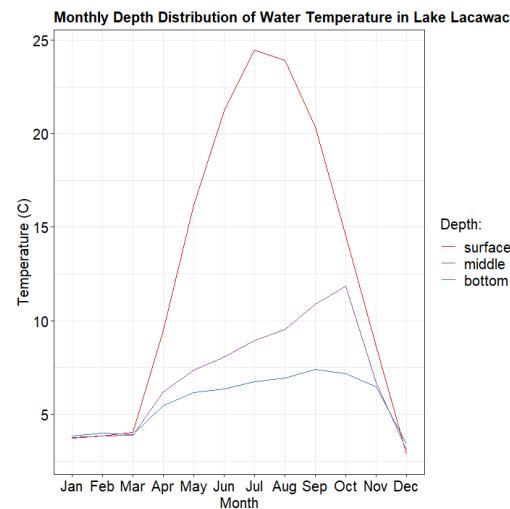
24

Common Types of Graphs

Three⁺ variables:

- Temperature of Lake Lacawac at multiple depths for each month of the year

- Line plot



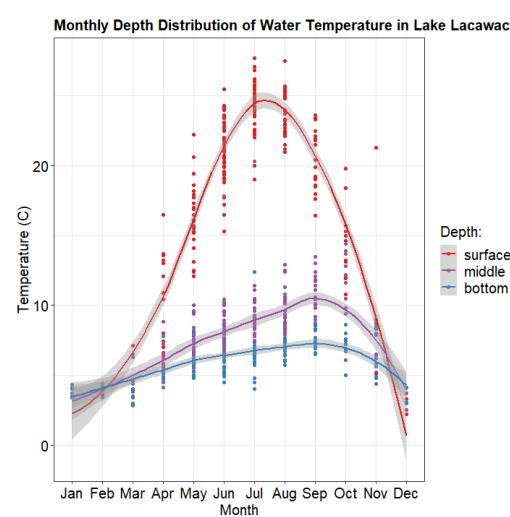
25

Common Types of Graphs

Three⁺ variables:

- Temperature of Lake Lacawac at multiple depths for each month of the year

- X-Y scatterplot



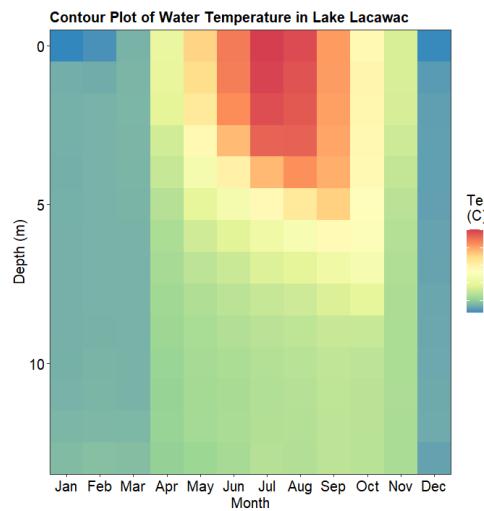
26

Common Types of Graphs

Three⁺ variables:

- Temperature of Lake Lacawac at multiple depths for each month of the year

- Contour plot



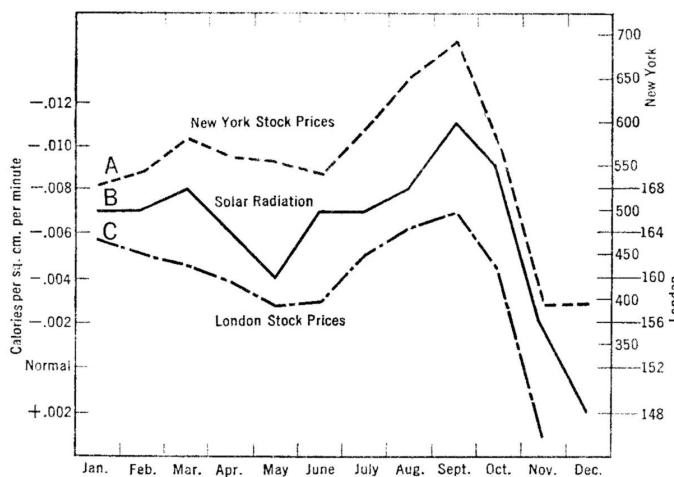
27

Keys to a Good Graph

- Graphs are only as useful as the data they display
- If the data is simple, ***keep the graph simple***
- Don't accidentally or purposefully distort or mislead
- If the data is complex, ***make the graph simple to understand***

28

Examples of Graphs



OVERLY COMPLEX GRAPH!

- Three axes with 3 different variables
- How is solar radiation directly related to stock prices?

from Ross Ihaka, University of Auckland, New Zealand

29

Examples of Graphs



UNNECESSARY GRAPH!

- Interpret 3 total components (2 axes and 1 color scheme) for only 2 pieces of data

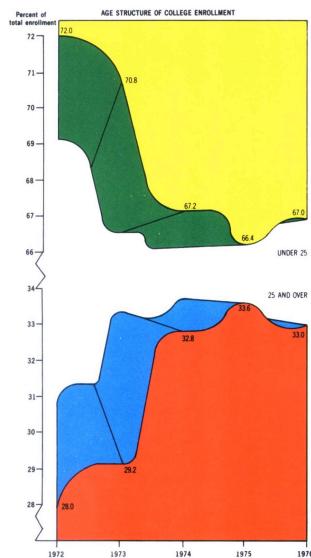
IMPROVEMENT:

- Just report “60% male and 40% female”

from Ross Ihaka,
University of Auckland,
New Zealand

30

Examples of Graphs



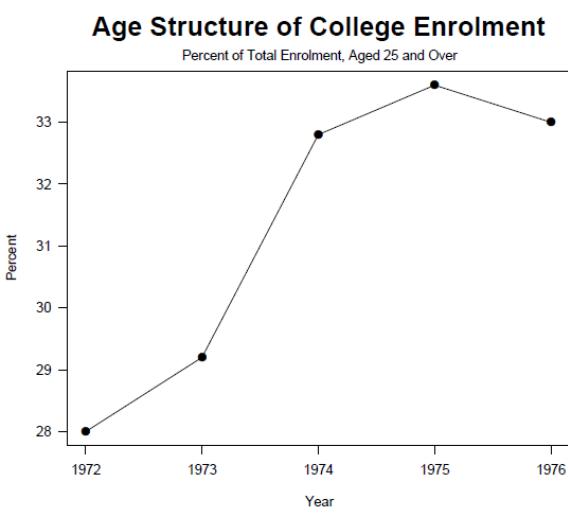
OVERLY COMPLEX GRAPH!

- Extra dimension is unnecessary – beware of 3-D graphs in Excel!
- Four colors misleads how many pieces of data are being presented
- Unnecessary splitting of y-axis

from Ross Ihaka,
University of Auckland,
New Zealand

31

Examples of Graphs



IMPROVEMENTS:

- Remove 3-D
- Focus on simple presentation of 5 key data points
- Clear axes

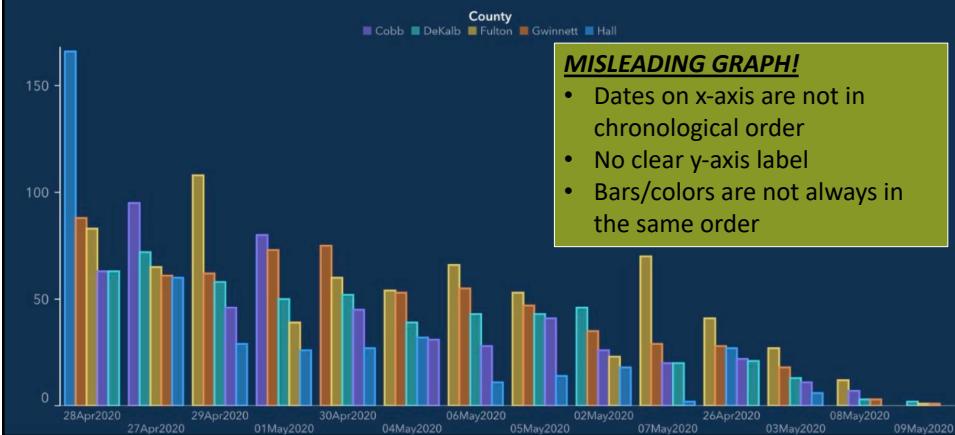
from Ross Ihaka,
University of Auckland,
New Zealand

32

Examples of Graphs

Top 5 Counties with the Greatest Number of Confirmed COVID-19 Cases

The chart below represents the most impacted counties over the past 15 days and the number of cases over time. The table below also represents the number of deaths and hospitalizations in each of those impacted counties.

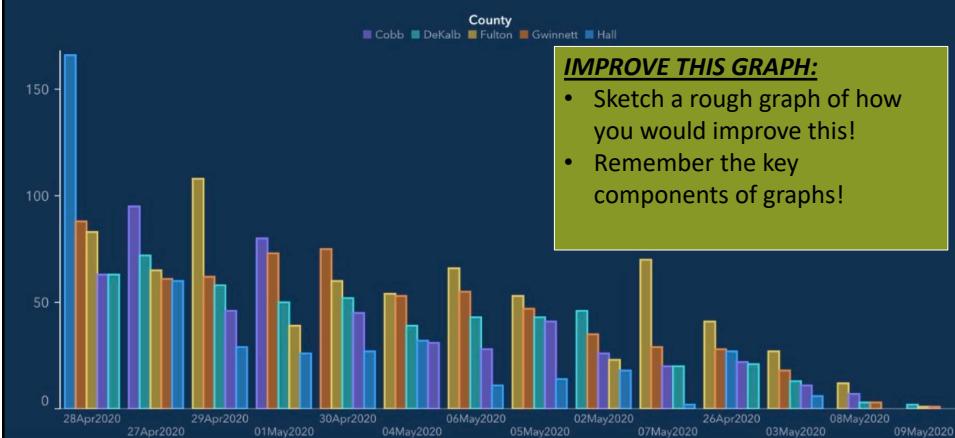


33

Examples of Graphs

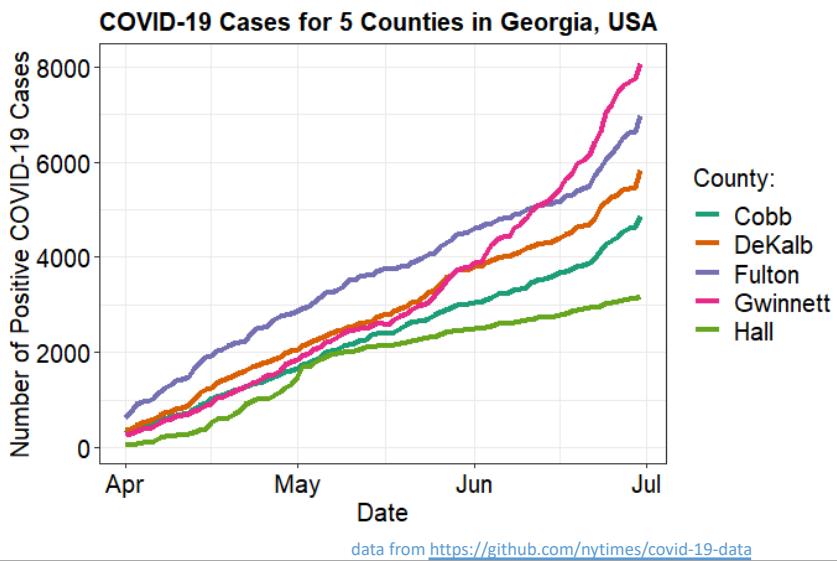
Top 5 Counties with the Greatest Number of Confirmed COVID-19 Cases

The chart below represents the most impacted counties over the past 15 days and the number of cases over time. The table below also represents the number of deaths and hospitalizations in each of those impacted counties.



34

Examples of Graphs



35

Live Coding

open RStudio and code along!

36

Overview of ggplot2

37

Grammar of Graphics

- Originally developed by Leland Wilkinson, adapted for R by Hadley Wickham
- Descriptive language to talk about components and visual elements of a graph or plot
 - Data being plotted
 - Geometric objects (lines, points, etc.)
 - Aesthetic appearance
 - Statistical information and transformations
 - Position, scales, coordinates
 - Facets or panels

38

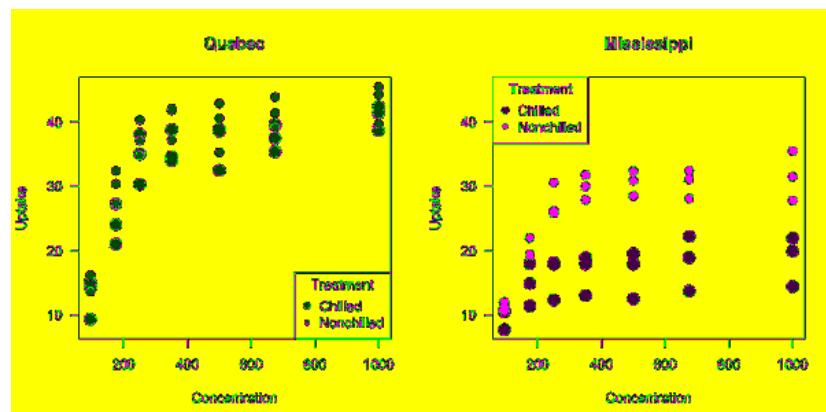
Grammar of Graphics

- Building a plot in ggplot2 is essentially constructing layers or pieces of a plot based on specific aspects of the data
- Each new layer is one new piece of code

39

Base R vs. ggplot2

CO_2 uptake in chilled vs. non-chilled treatments in 2 locations:



<https://statsinthewild.com/2019/08/22/ggplot2-vs-base-r-graphics-an-example/>

40

Base R vs. ggplot2

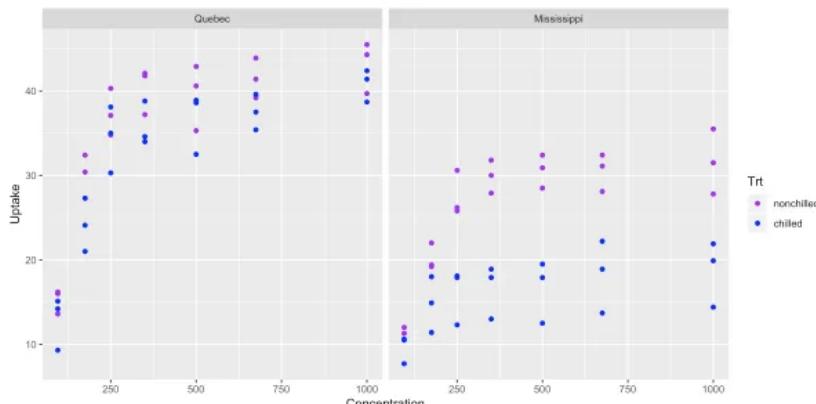
Base R code (30 lines):

```
summary(co2)
str(co2)
head(co2, 20)
CO2.QC <- subset(co2, Type == "Quebec" & Treatment == "chilled")
CO2.QN <- subset(co2, Type == "Quebec" & Treatment == "nonchilled")
CO2.MC <- subset(co2, Type == "Mississippi" & Treatment == "chilled")
CO2.MN <- subset(co2, Type == "Mississippi" & Treatment == "nonchilled")
xrange <- range(CO2$conc)
yrange<-range(CO2$uptake)
par(mfrow = c(1, 2))
plot(CO2.QC$uptake ~ CO2.QC$conc, pch = 19, lty = 2, lwd = 3, cex = 1.5, las = 1,
ylim = yrange, xlim = xrange, col = "purple", xlab = "concentration",
ylab = "uptake")
par(new=T)
plot(CO2.QN$uptake ~ CO2.QN$conc, pch = 20, lty = 2, lwd = 3, cex = 1.5, las = 1,
ylim = yrange, xlim = xrange, col = "lightblue", main = "Quebec", xlab = "",
ylab = "")
legend("bottomright", title = "Treatment", c("chilled", "Nonchilled"),
pch = c(19, 20), col = c("purple", "lightblue"))
plot(CO2.MC$uptake ~ CO2.MC$conc, pch = 19, lty = 2, lwd = 3, cex = 1.5, las = 1,
ylim = yrange, xlim = xrange, col = "orange", xlab = "Concentration",
ylab = "uptake")
par(new=T)
plot(CO2.MN$uptake ~ CO2.MN$conc, pch = 20, lty = 2, lwd = 3, cex = 1.5,
ylim = yrange, las = 1,
xlim = xrange, col = "red", main = "Mississippi", xlab = "", ylab = "")
legend("topleft", title = "Treatment", c("chilled", "Nonchilled"), pch = c(19,20),
col = c("orange", "red"))
par(mfrow = c(1, 1), las = 1)
```

41

Base R vs. ggplot2

CO_2 uptake in chilled vs. non-chilled treatments in 2 locations:



<https://statsinthewild.com/2019/08/22/ggplot2-vs-base-r-graphics-an-example/>

42

Base R vs. ggplot2

ggplot2 code (8 lines):

```
library(ggplot2)
ggplot(aes(x = conc, y = uptake, color = Treatment), data = CO2) +
  geom_point() +
  facet_grid(~ Type) +
  xlab("Concentration") +
  ylab("Uptake") +
  labs(color = "Trt") +
  scale_color_manual(values = c("purple", "blue"))
```

43

Base R vs. ggplot2

Plotting in base R can be very useful for...

- 1) Beginners learning R
- 2) Exploratory plots (don't have to be perfect)
- 3) Heat or contour maps with missing data

ggplot2 is often much quicker and more intuitive when creating...

- 1) Legends
- 2) Groupings
- 3) Faceting or panels
- 4) Easy customizing for presentations or publications

44

Computer Science in Modern Biology

Data Visualization in R

Instructor:

Rachel Pilla

Mahbobe Lesani

TAs:
Mason Murphy



45

Review of Day 1

46

23

Workshop Logistics

Zoom

- Feel free to unmute yourself to ask questions at any point

Chat

- Post any questions or comments here
- The TAs will be checking the chat to help answer questions

Break-Out Rooms

- If you are really stuck, you can move to a break-out room with a TA to share your screen and get you up to speed
- Break time is a great opportunity for this if needed!

47

Packages

- Packages only need to be installed ONCE
- But, each time you re-open R (today!), you need to load the packages you want to use so R can make all the functions available:

```
library(package name)
```



Hadley Wickham @hadleywickham

Replies to [@ijlytle](#)

@ijlytle a package is like a book, a library is like a library; you use library() to check a package out of the library #rsats

8:34 AM · Dec 8, 2014 · [Echofon](#)

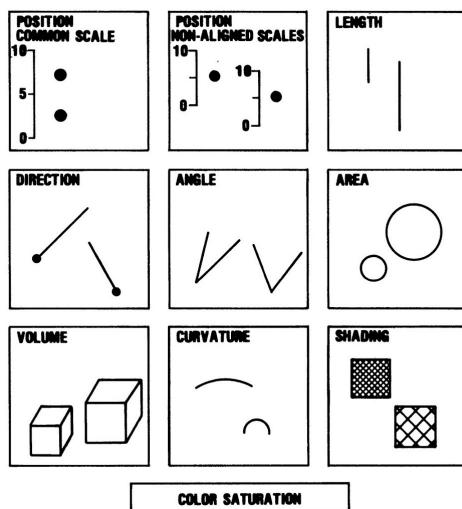
48

Components of a Graph

- 1) Axes
 - Horizontal x-axis and vertical y-axis
- 2) Labels
 - What are your axes measuring?
- 3) Title and/or caption
 - Description and information about the graph
- 4) Legend (sometimes)
 - If multiple lines, colors, shapes, etc., what does each represent?
- 5) DATA
 - The actual information!

49

Understanding a Graph



- Easy to perceive and understand
- ↓
- May require more work to understand
 - Can be very valuable when used appropriately in graphs

Cleveland & McGill, 1984 (*Journal of the American Statistical Association*)

50

Remember: Keys to a Good Graph

- Graphs are only as useful as the data they display
- If the data is simple, ***keep the graph simple***
- Don't accidentally or purposefully distort or mislead
- If the data is complex, ***make the graph simple to understand***

51

Introduction to ggplot2

52

Grammar of Graphics

- Originally developed by Leland Wilkinson, adapted for R by Hadley Wickham
- Descriptive language to talk about components and visual elements of a graph or plot
 - Data being plotted
 - Geometric objects (lines, points, etc.)
 - Aesthetic appearance
 - Statistical information and transformations
 - Position, scales, coordinates
 - Facets or panels

53

Grammar of Graphics

- Building a plot in ggplot2 is essentially constructing layers or pieces of a plot based on specific aspects of the data

Layer 1: Plot points of water temperature (y-axis) over time (x-axis)

Layer 2: Scale axes to preference

Layer 3: Add labels for axes and plot title

Layer 4: Make adjustments for font size, etc.

- Each new layer is one new piece of code

54

Grammar of Graphics

ggplot () +

Base Layer:
Initiate ggplot2 (always
start a new plot with this
code)

55

Grammar of Graphics

ggplot () +

“+” indicates another
layer will follow

56

Grammar of Graphics

```
ggplot() +  
  geom_point( ... ) +
```

Layer 1:
Plot points of water
temperature (y-axis) over
time (x-axis)

57

Grammar of Graphics

```
ggplot() +  
  geom_point( ... ) +  
  scale_x_continuous( ... ) +
```

Layer 2:
Scale axes to preference

58

Grammar of Graphics

```
ggplot() +  
  geom_point( ... ) +  
  scale_x_continuous( ... ) +  
  labs( ... ) +
```

Layer 3:
Add labels for axes and
plot title

59

Grammar of Graphics

```
ggplot() +  
  geom_point( ... ) +  
  scale_x_continuous( ... ) +  
  labs( ... ) +  
  theme( ... )
```

Layer 4:
Make adjustments for
font size, etc.

60

Geometries

- Describe the different types or ‘shapes’ of graphs that will be produced, and are very self-explanatory
- Examples:
 - geom_point
 - geom_line
 - geom_histogram
 - geom_smooth
 - geom_col
 - geom_bar

61

Aesthetic

- Indicates which variables will be plotted and on which axis or scale
- Inside the geometry function, always use the following coding syntax:

```
geom_point( mapping = aes( ... ), ... )
```

- Use column names to map variables to x, y, color, fill, etc.

62

Options & Customizations

- Additional layers to make the graph look how you want it to
- Examples:
 - Labels
 - Color or fill scales
 - Axis scales
 - Themes
 - Annotations
 - Facets or panels
 - Many other options!

63

Live Coding

open RStudio and code along!

64

Q&A Session

65

Additional Tips & Useful Resources

66

RStudio Cheat Sheets

- Short, handy guides to help with many packages and programming needs in RStudio
- Available free for download in multiple language options
- <https://rstudio.com/resources/cheatsheets/>

67

ggplot2 Cheat Sheet

Data Visualization with ggplot2 :: CHEAT SHEET

The diagram illustrates the grammar of graphics flow:

```

graph TD
    A[ggplot(data = mpg, aes(x = mpg, y = hwy)) + coord_fixed()] --> B[aes()]
    B --> C[geom_point()]
    C --> D[+ scale_color_hue(180)]
    D --> E[+ theme_minimal()]
    E --> F[print(ggplot)]
  
```

Basics

ggplot is based on the grammar of graphics; the idea is that you can build every plot from the same basic components—data, aesthetic mappings, and geoms—visual marks that represent data points, continuous x or y values, map variables in the data to visual properties, and define the plot's structure like size, color, and y location.

To display values, map variables in the data to visual properties using the geom functions: line, size, color, and shape.

Complete the template below to build a graph.

```

ggplot(data = DATA) +
  AES(MAP, MAP) +
  COORDINATE FUNCTIONS +
  SCALE +
  CREATE FUNCTION +
  THEME FUNCTIONS
  
```

ggplot(data = mpg, aes(x = mpg, y = hwy)) + coord_fixed() begins a plot that you finish by adding layers. Add one geom layer at a time, and then add them together to create a complete plot with glue data, geom, and mapping layers in the right order.

last.plot() returns the last plot.
 ggname("myplot", width = 5, height = 5) saves last plot as myplot.png in working directory. Machines the type for the extension.

Geoms Use the geom function to represent data points, use the geom's aesthetic properties to represent variables.

Use the grammar of graphics to structure a layer:

```

GRAPHICAL PRIMITIVES
  a = ggplot(mtcars, aes(mpg, wt))
  a + geom_point()
  a + geom_line()
  a + geom_rect()
  a + geom_bar()
  a + geom_text()
  a + geom_abline()
  a + geom_smooth()
  a + geom_hex()
  a + geom_vline()
  a + geom_qdensity()
  a + geom_jitter()
  a + geom_pointr()
  a + geom_quadrat()
  a + geom_rect()
  a + geom_rug()
  a + geom_smooth(method = "loess")
  a + geom_smooth(method = "lm", formula = y ~ x)
  a + geom_smooth(method = "gam", formula = y ~ s(x))
  a + geom_smooth(method = "kern", bandwidth = 0.05)
  a + geom_smooth(method = "wgam", bandwidth = 0.05)
  a + geom_smooth(method = "wgam", bandwidth = 0.05, fullrange = TRUE)
  a + geom_hex(stat = "hex")
  a + geom_hex(stat = "hex", binwidth = 0.05)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5, fill = "white")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5, fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5, fill = "white", alpha = 0.5, fill = "white")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5, fill = "white", alpha = 0.5, fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white")
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5)
  a + geom_hex(stat = "hex", binwidth = 0.05, binrange = TRUE, fullrange = TRUE, fill = "white", alpha = 0.5, color = "black", size = 1, stroke = "black", strokeWidth = 2, shape = 2, fill = "white", alpha = 0.5, stroke = "black", fill = "white", alpha = 0.5, fill = "white", alpha = 0.5]
  
```

TWO VARIABLES continuous x, continuous y

ONE VARIABLE continuous

THREE VARIABLES

continuous x, continuous y

maps

continuous bivariate distribution

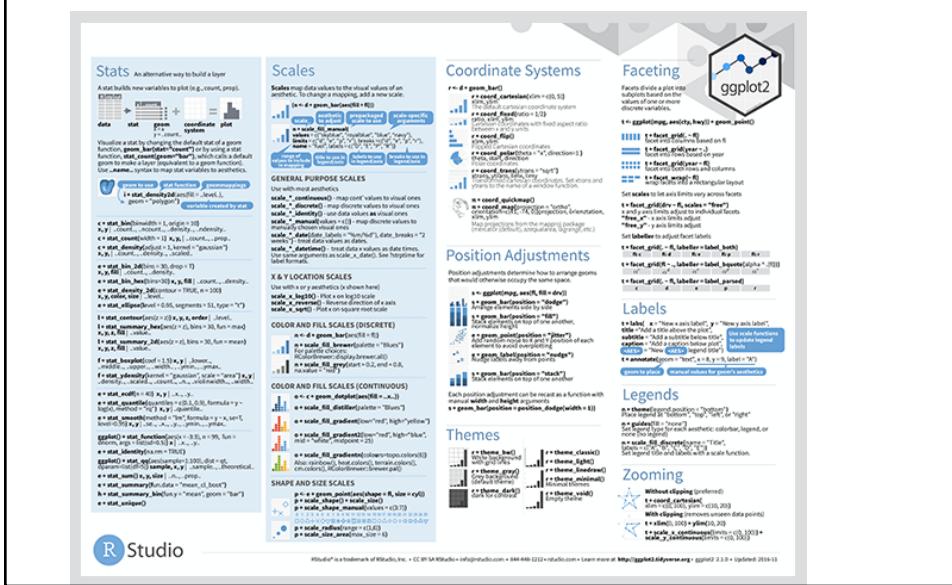
continuous function

visualizing error

maps

68

ggplot2 Cheat Sheet



69

Tidyverse

- Packages to help organize data and streamline R coding, specifically for data science
- Functions in the packages share similar syntax and coding structure
- “dplyr”, “tidyverse”, and “ggplot2” are part of tidyverse
- <https://www.tidyverse.org/>

70

Lubridate

- Package for more effectively working with dates between Excel and R
 - Can recognize date/times with minimal formatting issues
 - Also allows for date/time differencing, among other useful functions
 - Included in tidyverse
 - <https://lubridate.tidyverse.org/>

71

Lubridate

72

Lubridate

Math with Date-times — Lubridate provides three classes of timespans to facilitate math with dates and datetimes.

Math with date-times relies on the timeline, which behaves inconsistently. Consider how the timeline behaves during:

- A normal day: `now = ymd("2018-01-01 10:30:00") %>% "US/Eastern"`
- The start of daylight savings (spring forward): `gap = ymd("2018-03-22 03:30:00") %>% "US/Eastern"`
- The end of daylight savings (fall back): `gap = ymd("2018-11-04 00:30:00") %>% "US/Eastern"`
- Long years with leap seconds: `gap = ymd("2018-03-07")`

PERIODS
Add or subtract periods to model events that happen at specific times, like the NYSE opening time.

```
p = months(1) + days(12)
# "2018-01-01" + 12 months, weeks(1) + 13 weeks,
# days(365) + 12 days, hours(12) + 12 hours,
# minutes(720) + 12 minutes,
# milliseconds(12) + 12 milliseconds,
# microseconds(12) + 12 microseconds,
# nanoseconds(12) + 12 nanoseconds,
# picoseconds(12) + 12 picoseconds.

period_(num = NULL, units = "second", ...)
```

DURATIONS
Add or subtract durations to model physical processes, like battery life. Units are stored as seconds, the only time unit with a consistent length. DURATIONS are a class of durations found in base R.

```
d = days(24)
# 1 * 24 * 60 * 60 seconds,
# 1 * 24 * 60 * 60 * 1000 milliseconds,
# 1 * 24 * 60 * 60 * 1000000 microseconds,
# 1 * 24 * 60 * 60 * 1000000000 nanoseconds,
```

INTERVALS
Divide a duration by a duration to determine its physical length, divide an interval by a period to determine its implied length in clock time.

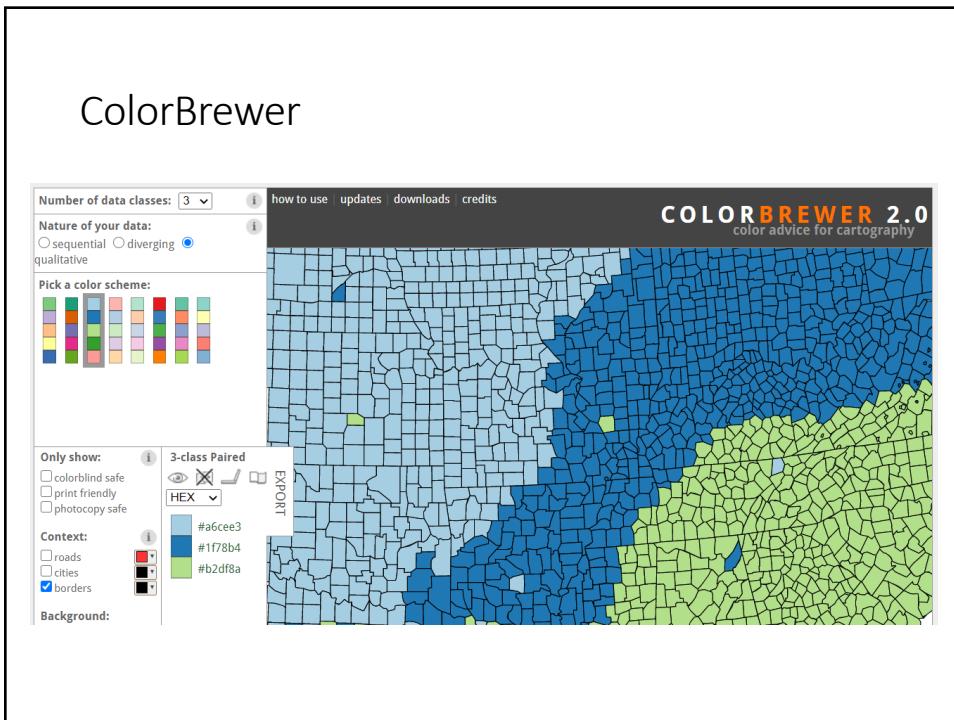
```
interv = intervals("2017-12-31 00:00:00 UTC", "2017-12-31 10:00:00 UTC")
# An interval from 2017-12-31 00:00:00 UTC to 2017-12-31 10:00:00 UTC
# 10 hours = 10 * 60 * 60 seconds = 36000 seconds = 3.6e+06 milliseconds = 3.6e+12 microseconds = 3.6e+15 nanoseconds = 3.6e+18 picoseconds.
```

73

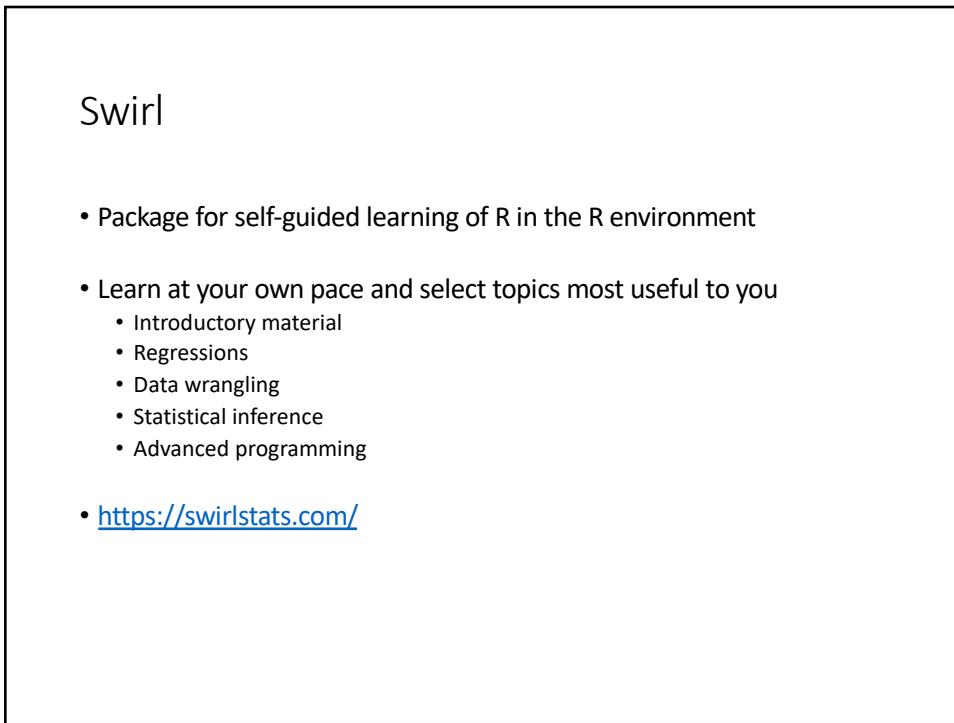
ColorBrewer

- Pre-made color schemes and palettes
- Effective graphic color selections for maximum perception
 - Qualitative, sequential, and diverging
- Includes options for photo-copy friendly, color-blind friendly, etc.
- <http://colorbrewer2.org/>

74



75



76

Thank you for joining!

to share feedback or ask questions,
please feel free to contact me:

 pillarm@miamioh.edu
 @rmpilla



Notes

