R and Statistical Analysis

Norms

- Raise your hand for questions
  - If I don’t see you, just call it out
- Don’t let the deck fool you; we can go off script and down rabbit holes the class finds interesting
- I’m going to try to make jokes.


NO
made on imgur

Statistics, Data Science, and Machine Learning

Oh my!

Definitions

- Machine Learning: Focuses on making many, small, low-stakes decisions.
  - E.g. Decide if the picture is of a cat
  - Doesn't matter too much if we get some cats/non-cats wrong, as long as we're pretty good on average
- Statistics: Focuses one, important decision.
  - E.g. Decide whether to replace product A with product B
  - We want to make the right call as often as possible, and mitigate risks
- Data Science: ??????????
  - Covers both of the above, but also biology, maybe?
  - Social Science? Economics uses data, right?

[1]I owe the Statistics / ML distinction to Cassie Kozyrkov of Google. See her talks if you ever have a chance.

Definitions

- Statistics: Focuses one, important decision.
  - E.g. Decide whether to replace product A with product B
  - We want to make the right call as often as possible, and mitigate risks

- We're talking about this one.

- But we're staying so high-level that what we say will apply to the others, too.

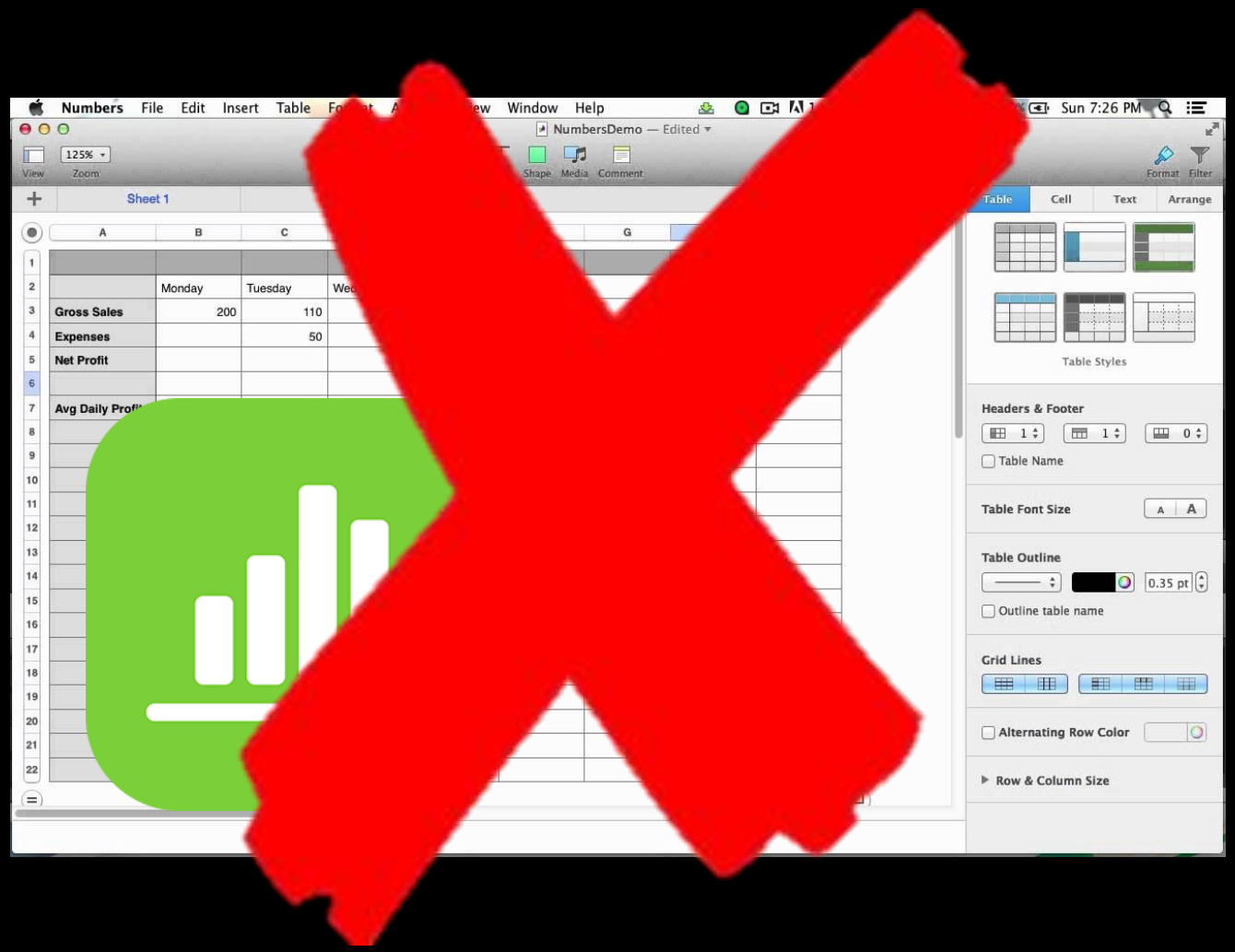
The toolkit

Excel is not an analysis tool!

- And before you ask,

Excel is not an analysis tool!

- Numbers isn't, either



Sheets?


NO
made on imgur

A Parable

- Reinhart and Rogoff, of Harvard, published “Growth in a Time of Debt”
- It found that countries experiencing high government debt had flat or negative growth rates
- Influenced Paul Ryan’s budget proposals, and the debate on government spending generally

- Three years later, a grad student at Umass realizes the Excel formula left off 5 rows of data

No Excel for Analysis

A CS perspective

What's wrong with Excel?

- Excel serves two masters
  - It wants to make data entry and storage easy
  - It wants to make analyses robust

- It's hard to do both.
- You have to come up with an encoding [pattern of 1s and 0s] that covers both the data and the manipulations on the data

- Moral: Use specific tools for specific jobs

What's right with Excel?

- Excel & co are still great tools for entering data
  - See: Every business's finance sheets
- Sheets is particularly neat because it easily allows multiple collaborators
  - See also: Sharepoint
  - Many products are headed towards these Cloud solutions – more on that, and its risks, later in the course

Take Home #1

- Excel is fine for collating or entering data, but when you want to analyze the data pick a different tool

- Don’t build labyrinthine Excel sheets, and don’t rest the business on them
  - It’s impossible to maintain, update, or check for bugs

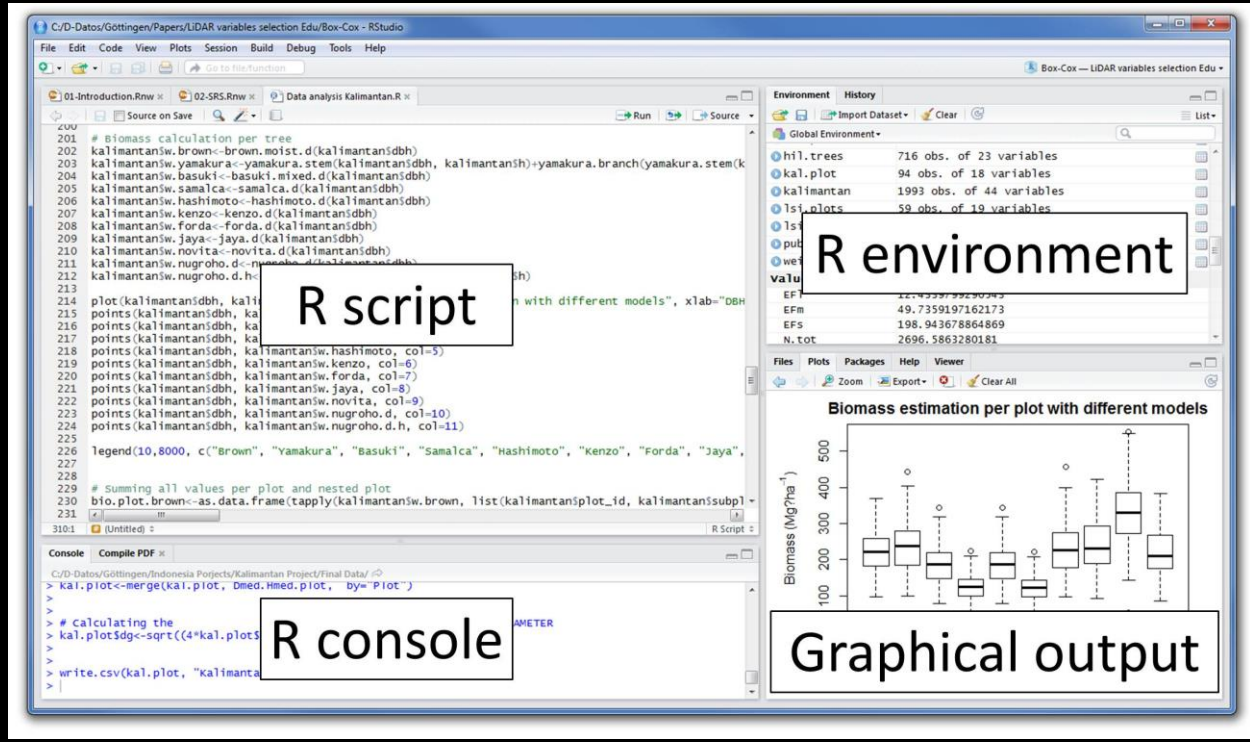
- Use good software and good software design

Use R for Analysis

A highly biased perspective

What is R?

- R is a programming language built from the ground up to handle statistical analyses
- You can get started with a few simple commands, but it scales to doing fully custom analyses
- R Studio makes interacting with R extremely friendly



Why R?

- R is open source and Free
  - That second one really matters
- Extremely easy installation of cutting-edge analysis code
- Good Documentation
- Great Plotting and Graphics
- Basic data structure is just like Excel

Why Not R?

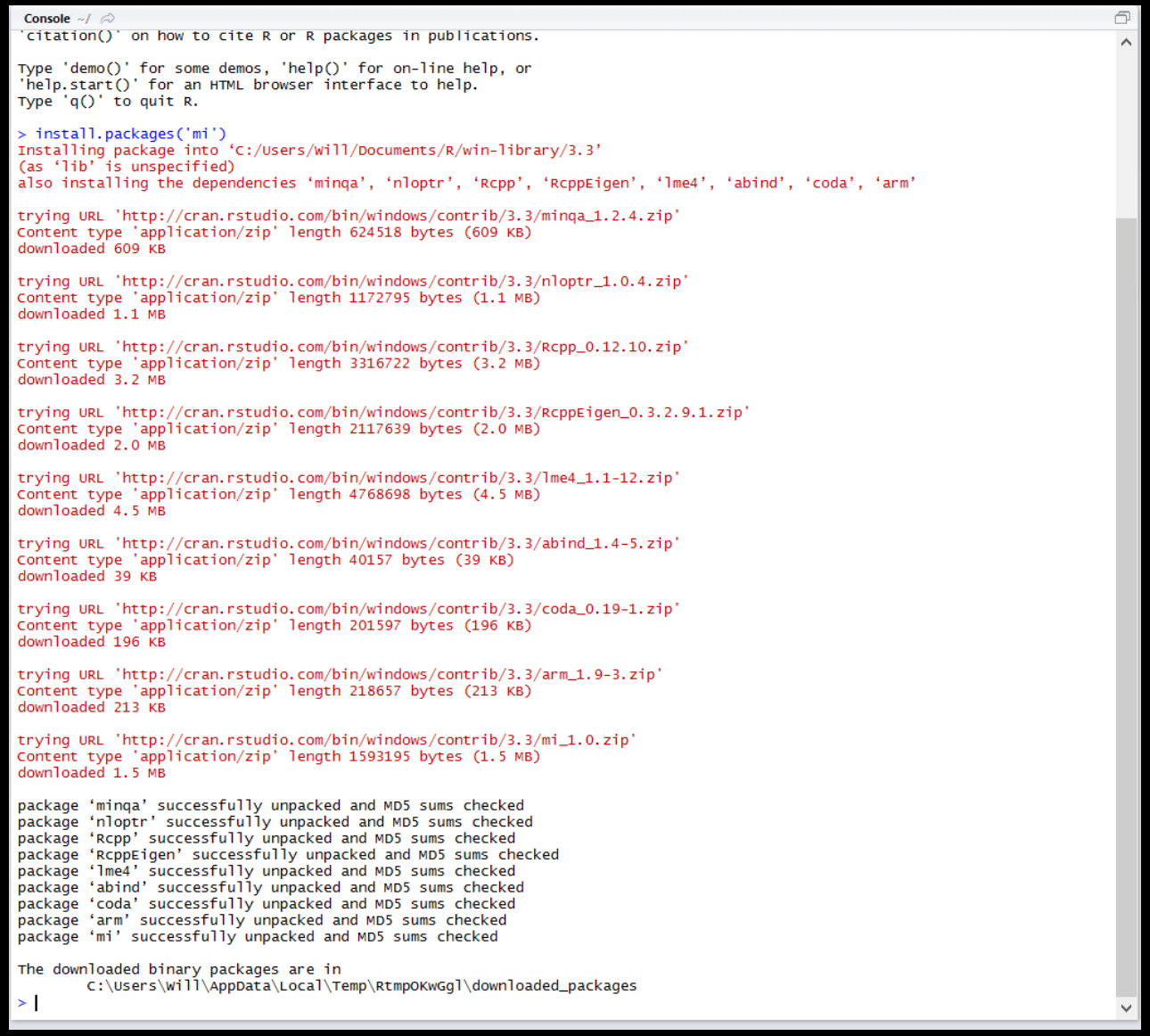
- R is slower than other languages like C and Java
- R isn’t good at massive-scale datasets
  - For those, see Hadoop, et al
- Inconsistent naming and syntax
- In general: R is amazing for exploring and prototyping, but bad for deployment

Let me prove it

A highly risky demo

Easy Installation of Cutting-Edge Code

- Consult the internet to find the name of the package you want
- I want the ‘mi’ package (for Multiple Imputation, a neat, nerdy statistical technique)
- `install.packages('mi')`
- Bam. Cutting edge code on my machine for free.



Good Documentation

- RStuido will automatically suggest function and variable names as you type
- Entering ?some_function, e.g. `?rowSums` will summon a help page
- These pages tell you about the inputs to the functions, and give simple examples you can run

Form Row and Column Sums and Means

Description

Form row and column sums and means for numeric arrays (or data frames).

Usage

```
colSums (x, na.rm = FALSE, dims = 1)
rowSums (x, na.rm = FALSE, dims = 1)
colMeans(x, na.rm = FALSE, dims = 1)
rowMeans(x, na.rm = FALSE, dims = 1)

.colSums(x, m, n, na.rm = FALSE)
.rowSums(x, m, n, na.rm = FALSE)
.colMeans(x, m, n, na.rm = FALSE)
.rowMeans(x, m, n, na.rm = FALSE)
```

Arguments

`x`	an array of two or more dimensions, containing numeric, complex, integer or logical values, or a numeric data frame. For `.colSums()` etc, a numeric, integer or logical matrix (or vector of length `m * n`).
`na.rm`	logical. Should missing values (including `NaN`) be omitted from the calculations?
`dims`	integer: Which dimensions are regarded as 'rows' or 'columns' to sum over. For `row*`, the sum or mean is over dimensions `dims+1, ...`; for `col*` it is over dimensions `1:dims`.

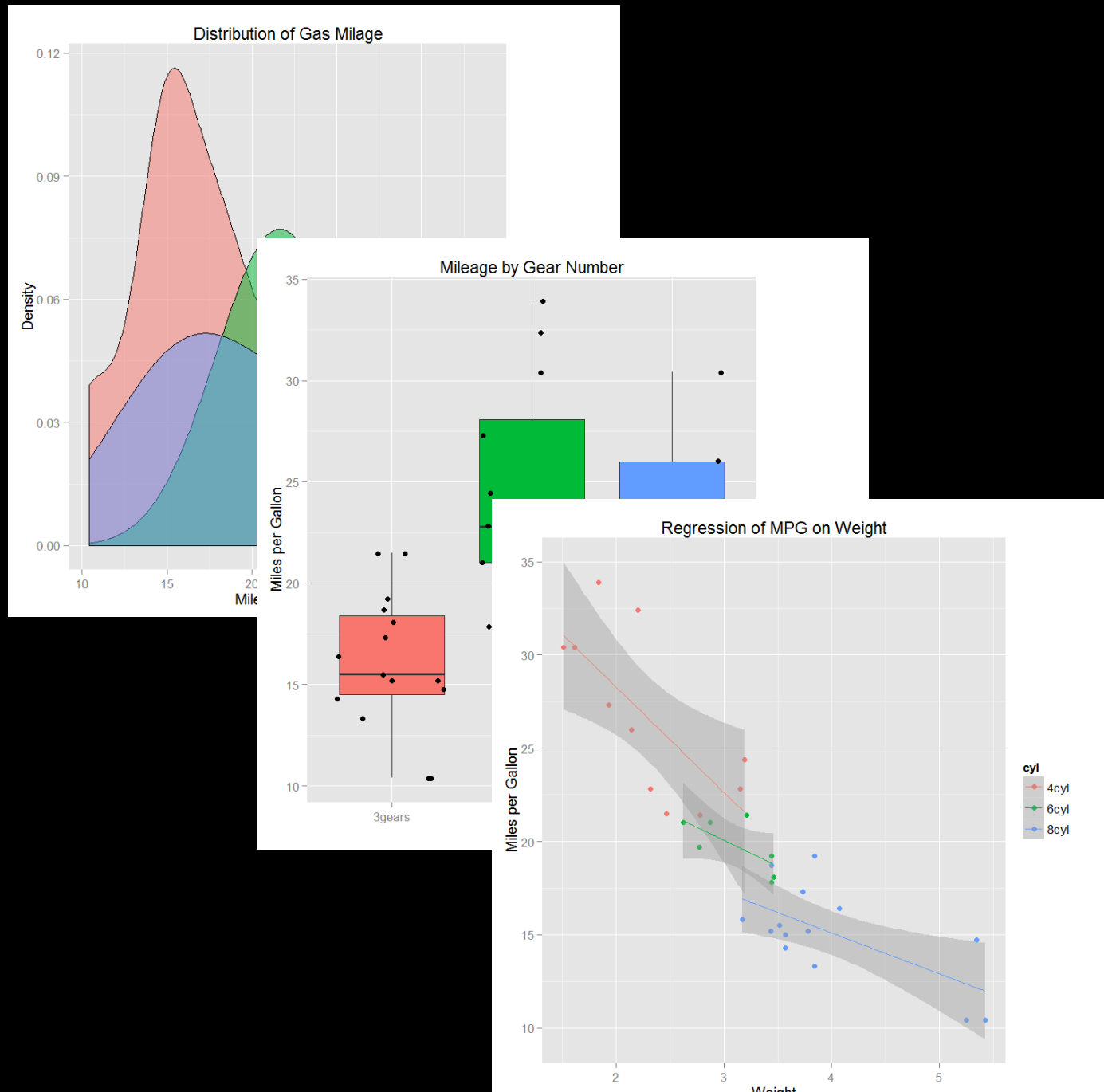
Examples

```
## Compute row and column sums for a matrix:
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
rowSums(x); colSums(x)
dimnames(x)[[1]] <- letters[1:8]
rowSums(x); colSums(x); rowMeans(x); colMeans(x)
x[] <- as.integer(x)
rowSums(x); colSums(x)
x[] <- x < 3
rowSums(x); colSums(x)
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
x[3, ] <- NA; x[4, 2] <- NA
rowSums(x); colSums(x); rowMeans(x); colMeans(x)
rowSums(x, na.rm = TRUE); colSums(x, na.rm = TRUE)
rowMeans(x, na.rm = TRUE); colMeans(x, na.rm = TRUE)
```

Great Plotting

- Basic R plots are highly customizable
- The ggplot2 package (again, just install.packages('ggplot2')) can make them exceptionally pretty

- Lots of useful one-line plots are available! E.g., plot(), boxplot() or hist()
- Most R outputs know how best to respond to plot() !!!



Inconsistent Naming

- Want mean of each column?

  colMeans()

- Want the name for each column?

  colnames()

- Names for each row?

  row.names()

AAAAARRRRRRRRGH

- But you get used to it…

Excel-Like Data Structre

- R runs on Data Frames

- A data frame has the row/column format of an Excel sheet
- Each column is some fact or feature, each row is some entity

- Supports missing data natively via the symbol NA

	state	fips	county	countyfips	dc	pop	pct_mortality
1	ALABAMA	1	AUTAUGA	1001	1	64915	1.540476e-05
2	ALABAMA	1	BALDWIN	1003	15	195253	7.682340e-05
3	ALABAMA	1	BARBOUR	1005	1	33987	2.942301e-05
4	ALABAMA	1	BIBB	1007	1	31175	3.207698e-05
5	ALABAMA	1	BLOUNT	1009	5	91547	5.461675e-05
6	ALABAMA	1	BULLOCK	1011	0	8197	0.000000e+00
7	ALABAMA	1	BUTLER	1013	1	31722	3.152386e-05
8	ALABAMA	1	CALHOUN	1015	12	233021	5.149750e-05
9	ALABAMA	1	CHAMBERS	1017	0	57813	0.000000e+00
10	ALABAMA	1	CHEROKEE	1019	0	43828	0.000000e+00
11	ALABAMA	1	CHILTON	1021	3	68837	4.358121e-05
12	ALABAMA	1	CHOCTAW	1023	3	22441	1.336839e-04
13	ALABAMA	1	CLARKE	1025	3	37772	7.942391e-05
14	ALABAMA	1	CLAY	1027	3	26954	1.113007e-04
15	ALABAMA	1	CLEBURNE	1029	1	29735	3.363040e-05
16	ALABAMA	1	COFFEE	1031	4	81882	4.885079e-05
17	ALABAMA	1	COLBERT	1033	6	104971	5.715864e-05
18	ALABAMA	1	CONECUH	1035	2	20687	9.667907e-05
19	ALABAMA	1	COOSA	1037	1	17828	5.609154e-05

Showing 1 to 20 of 3,113 entries

R compared to other languages

What tradeoffs does R make, why, and what are the impacts?

Anyone can contribute to R

- Some languages [C] have governing bodies that carefully define what syntax they support
- Others [Python] encourage community contributions, but the community sets and enforces strong style conventions
- R will let anyone publish their code.
- **Impact**: Research teams often publish first in R, but the syntax and style are all over the place

R is loosely typed and interpreted

- Typed vs loosely typed: whether the programmer has to state and plan out whether each variable will be an int, float, etc.
- Interpreted vs Compiled: Whether the code runs line-by-line or is massaged into a more optimal set of instructions

- Python is also loosely typed and interpreted
- C is strongly typed and complied

- **Impact**: C is considerably faster to run, but considerably slower to write in.
- C is easier on the computer, harder on the human.

R is a functional language

- ‘Functional’ means that whole functions can be saved as variables. In particular, some function might take a helper function as one of its inputs

- Python likewise supports this style, but to a more limited degree

- **Impact**: R has functions like `apply(some_data,some_function)` which does the function on e.g. each row of the data
- This helps do more in a single line and mitigate the penalty for interpreted vs compiled

R is vectorized

- ‘vecotrized’ means that the language’s data structures and functions work on big chunks of data (like rows or columns) at a time

- In C, we tend to think on the level of individual numbers and write lots of loops
- In R loops are slower, and we get functions like colmeans() that do a lot of work at once

- Result: Again, R bites off more in a single line- this helps keep the CPU fed with numbers to crunch, and thus mitigate dead time

Take Home #2

- All software needs to make tradeoffs in what it's good at
- Nothing can be fastest, easiest, and prettiest
- But, ultimately, all languages can do the same set things – just with varying levels of pain and speed
- So when choosing a language, research what it does well and what it does poorly

Alternatives to R

Python

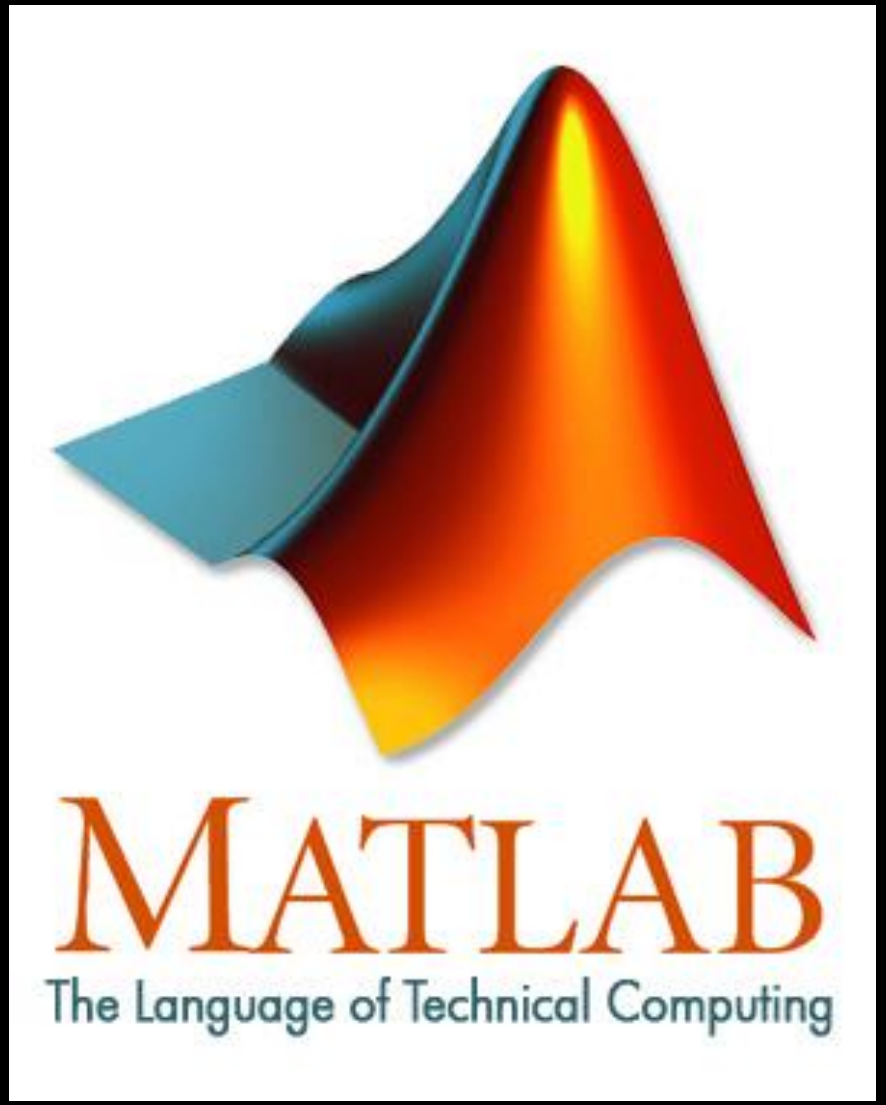
- Easier and more consistent to write in than R, especially for typical programming tasks
- More general-purpose: statistics and data analyses features may have complex syntax and require special packages
- Still gathering everyone in python 3 (vice python 2)

Stata

- Proprietary software, requires buying a license
- Purpose-built for statistics, and very well curated
- Not as cutting-edge as R
- Struggles with large datasets

MATLAB

- Expensive, especially if purchasing the Statistics Toolbox on top of the base product
- Great development environment- graphical tools to debug and measure code
- Slow to pick up new features and techniques



Take Home #3

- All leading analysis software favors the user over the computer. They're not lightning quick, but they're fast enough

- Compared to the alternatives, R elected to be fast-moving and make new developments available quickly
  - And its free, so you get way more than you paid for

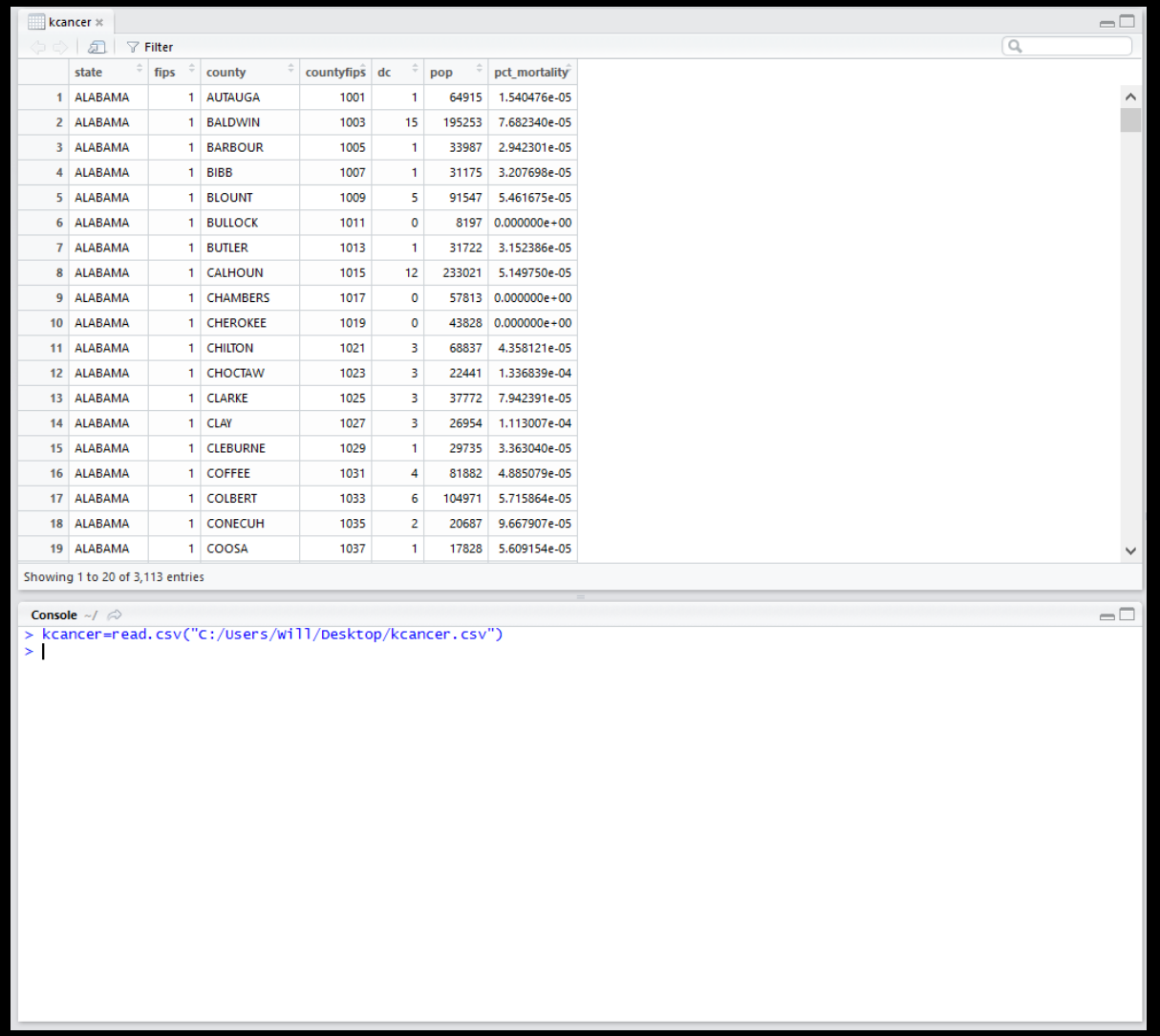
- R is a great choice that will grow with you and reward investment

Using R

This is for future you. Catch what you can now, come back to review later.

Getting data in

- Excel can read relatively few formats
- R can read almost anything, especially if you have a dev team (or can install packages!)
- To get started[1], read.csv() is the easiest



[1]Any excel workbook can be saved directly as .csv (though only the numbers in each cell survive)

Excel-Like Data Structure

- We can easily refer to individual columns

  `kcancer$pop` #refers to the population of each county

- Or we can refer to specific rows and columns:

  In general: kcancer[which rows?, which columns?]

  Example: `kcancer[1:10,c("state","county","pop")]`

  Gets the first 10 rows, and displays the three stated columns

- We can combine to great effect:

```
kcancer[kcancer$pop<100000,c("state","county","pct_mortality")]
```

#get rows (i.e. counties) where population is less than 100,000; display state, county name, #and percent mortality

#Net: dig in and investigate the low-population counties!

Useful one-liners

This is for future you, too

Data Exploration

- Get averages
- colmeans(kcancer[,c(“pop”,”pct_mortality”)
- Get spreads (Standard Deviation)
- sd(kcancer$pop)
- See how the data are distributed
- hist(kcancer$pop)
- See how/if two variables are related
- plot(kcancer$pop, kcancer$pct_mortality)

Data Exploration

- View the first few rows of data
- head(data)
- Build a 2d table of the data
- table(kcancer$state,kcancer$fips)
- Make a box and whiskers plot
- boxplot(kcancer$pct_mortality)
- See if data have a linear relationship, and how strong it is
- lm(kcancer$pct_mortality ~ kcancer$pop + kcancer$dc)

Altogether

- Remember, we can combine the above with the subsetting tricks we just saw

- So we can study just the counties where population is low

```
hist(kcancer$pct_mortality[kcancer$pop<100000])
```

- Or just the counties with high mortality rates!

- Or we could program R to comb through all of these possibilities!!!


gifhop.tumblr.com

Hammertime

- That’s not a happy dance.
- That’s because it’s hammer time- you’ve got a hammer, and everything will look like a nail
- Looking at just the counties with high death rates is a great way to discover spurious correlations
- It’s time we had The Stats Talk.

The Joy of Stats

Or: 6 steps to (more) safely exploring and deciding from your data

The Guide to Safer Stats[1]

1. Pick your question carefully. Pre-define what would convince you and let the data speak.
2. Tie your analysis to the decision: measure the things that matter and be honest about the difference between what's measured and what's sought.
3. Get the right data. What's easy often isn't what's needed.
4. Clean and validate your data. This will take the majority of the project.
5. Set aside a validation set, and a test set! You don't get to explore and confirm on the same data set.
6. If measuring an impact, have a control group.

[1]Most of this section is inspired by Cassie Kozyrkov, of Google. She has a 4 hour class on these topics and you'll love every minute

The Guide to Safer Stats

1. Pick your question carefully. Pre-define what would convince you and let the data speak.

- Exploring the data is fine, but if you plan to make a decision based on the data… decide before you look at them

- Figure out what your default is (based on intuition, risk, etc.) and what the data would have to say to move you from that default

- This leads to much better analyses than wandering until you bump in to data that backs up your gut

The Guide to Safer Stats

2. Tie your analysis to the decision: measure the things that matter and be honest about the difference between what's measured and what's sought.

- It's often impossible to measure the real thing of interest (customer satisfaction? Not without a full neuro lab and a massive budget)

- Work with the best approximation you can get (not necessarily the one you have), but be clear with yourself and others about where and how it falls short

The Guide to Safer Stats

3. Get the right data. What's easy often isn't what's needed.

- If you need a random sample, that doesn't mean "whatever you can grab", it means carefully controlling the entire pipeline to make sure each thing is equally likely to make it in

- Likewise, don't trust other people's data. Ask exactly how it came to be.

- Importantly, this may mean that you have to *collect* the data

The Guide to Safer Stats

4. Clean and validate your data. This will take the majority of the project.

- This goes with not trusting other people's data

- Write down every single fact you can about the data. (Units sold should always be positive, dates should fall in a certain range, IDs should be sequential)

- Check each assumption, and figure out how/if discrepancies can be fixed

The Guide to Safer Stats

5. Set aside a validation set, and a test set! You don't get to explore and confirm on the same data set. Ever!

- You can do whatever exploring and theory-building you want in the main dataset, but set aside 10-20% to test your theories

- This protects you against chasing random noise in the data

- If you're passing the analysis off to someone else, lie to them and hide 10-20% of the data under lock and key.
- When they're finally done, see if their predictions hold on the hidden data.

The Guide to Safer Stats

6. If measuring an impact, have a control group.

- The best way we've found to determine cause and effect is to actually change a randomly-selected portion of the world

- If you want to claim that A is better than B, or that A causes X or that A prevents Y, you want two identical groups, one with A and one without

- It's really easy to forget. We think "I want to know about A" so we study A, but we forget to also measure the baseline / alternative

Summary

That’s right, we made it

TL;DR?

- Excel is a data entry tool, not an analysis tool, not a production tool.
- Like all languages, and all programs, R and Excel make tradeoffs that make them good for some tasks and not for others
- R is a great analysis tool, which scales well to many problems
  - Still not a full-scaled production tool
  - But fantastic if the client is you or your team
- Statistics is hard to get right. If you have a decision to make, use the guide below get on the right path

The Guide to Safer Stats[1]

1. Pick your question carefully. Pre-define what would convince you and let the data speak.
2. Tie your analysis to the decision: measure the things that matter and be honest about the difference between what's measured and what's sought.
3. Get the right data. What's easy often isn't what's needed.
4. Clean and validate your data. This will take the majority of the project.
5. Set aside a validation set, and a test set! You don't get to explore and confirm on the same data set. Ever!
6. If measuring an impact, have a control group.

[1]Again, thanks to Cassie Kozyrkov.