Lecture 6:
Storytelling and Functions

March 1, 2021

Overview

Course Administration

Good, Bad and Ugly

Telling a Story

Functions in R

Course Administration

1. Reading quiz

Course Administration

1. Reading quiz
2. Sign up for consultations!
   - sign up for slots April 8, link is on Lecture 11
   - let me know if you cannot make any slot

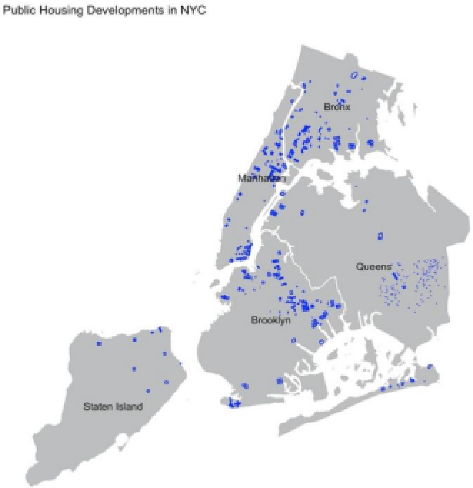
Course Administration

1. Reading quiz
2. Sign up for consultations!
   - sign up for slots April 8, link is on Lecture 11
   - let me know if you cannot make any slot
3. In-class workshop: March 29, instructions posted

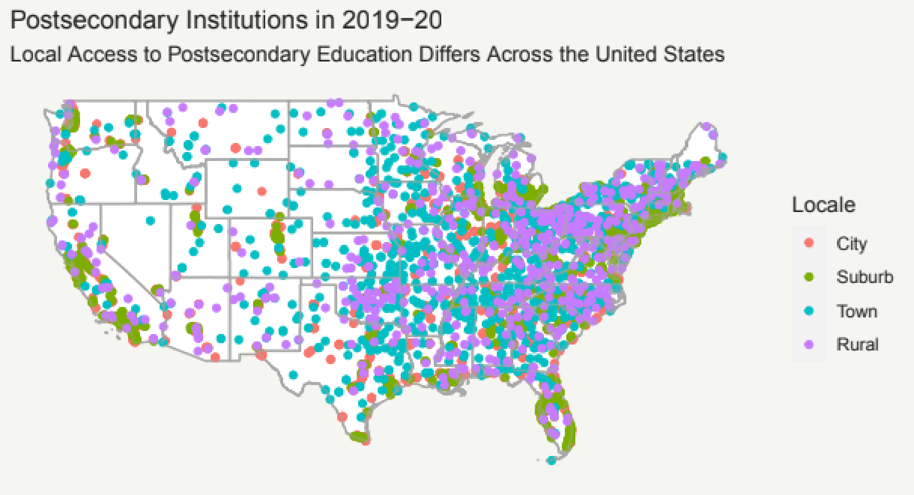
Course Administration

1. Reading quiz
2. Sign up for consultations!
   - sign up for slots April 8, link is on Lecture 11
   - let me know if you cannot make any slot
3. In-class workshop: March 29, instructions posted
4. Meet visualizers
   - March 29: Former students and my old boss
   - April 5: Kate Rabinowitz, Washington Post
5. Next week: comments on charts
6. Grading worksheet up by end of day
7. Anything else?

Anthony's Map of Public Housing in New York

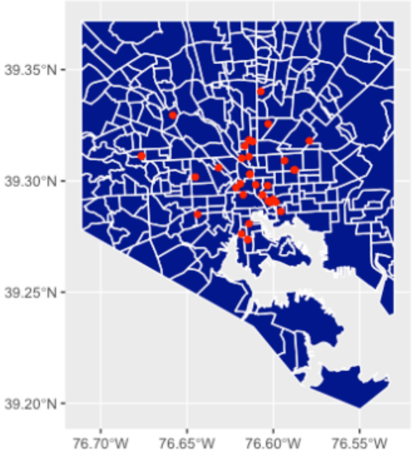


Sarah’s Map


Postsecondary Institutions in 2019–20
Local Access to Postsecondary Education Differs Across the United States
Locale
City
Suburb
Town
Rural

Bianca’s Map

Location of Homeless Shelters in Balitmore City, 2019



Next Week's Assignment

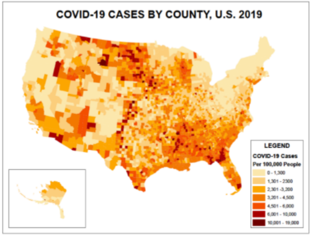
Find a descriptive or choropleth map, or any type of chart that we have already covered in class. Post link to google sheet by Wednesday noon.

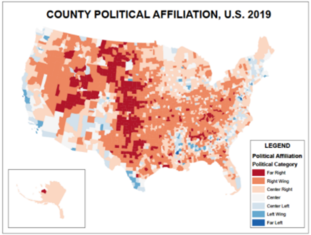
Finder	Commenter
Maureen M.	Winnie
Preston B.	Or Winnie

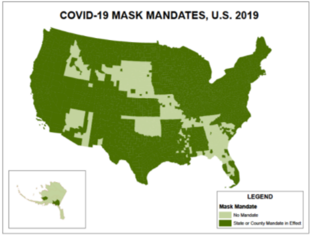
This Week's Good Bad and Ugly

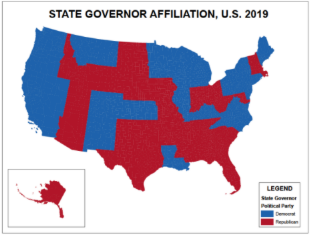
Finder	Commenter
Stephanie P.	Detrick C.
Detrick C.	Bianca O.

Stephanie's Example from Last Semester

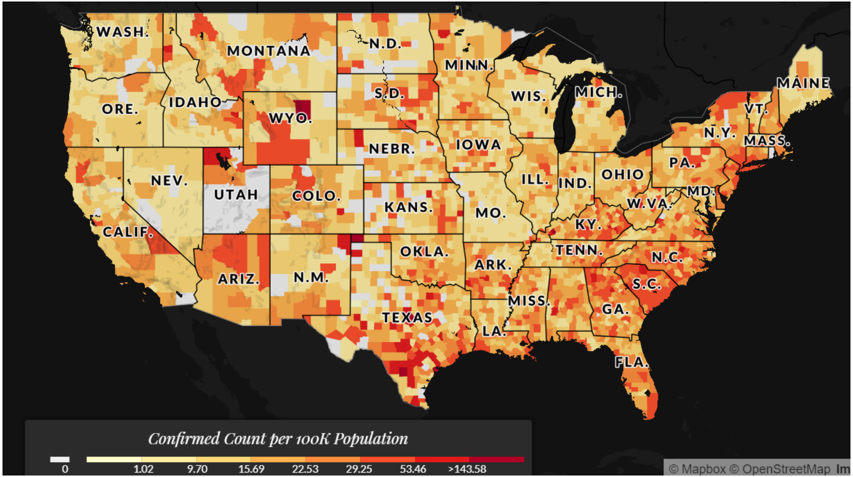





COVID-19 MASK MANDATES, U.S. 2019
LEGEND
Mask Mandate
No Mandate
State or County Mandate in Effect



Detrick's Example from the US Covid Atlas, U. of Chicago



Go yourself at https://theuscovidatlas.org/.

Stories

Today

1. Components of a story
2. Pulling apart a graph

1. Components of a Story

- Act 1: introduce characters, set up problem
- Act 2: working on the problem, main character changes as a result of problem
- Act 3: climax and resolution of the problem

What Does this Mean for a Policy Brief?

What Does this Mean for a Policy Brief?

1. Pose the problem, showing its importance
2. Give evidence for the problem or magnitude
3. Propose resolutions

Which of Knaflic's Advice is Most Relevant for this Communication?

- Storyboard

Which of Knaflic's Advice is Most Relevant for this Communication?

- Storyboard
- Motivate: identify a problem/question/tension

Which of Knaflic's Advice is Most Relevant for this Communication?

- Storyboard
- Motivate: identify a problem/question/tension
- The evidence

Which of Knaflic's Advice is Most Relevant for this Communication?

- Storyboard
- Motivate: identify a problem/question/tension
- The evidence
  - In Knaflic's book this is the lead-up to a policy
  - In this work, it can be the lead-up to a conclusion
  - Or an establishment of fact

Which of Knaflic's Advice is Most Relevant for this Communication?

- Storyboard
- Motivate: identify a problem/question/tension
- The evidence
  - In Knaflic's book this is the lead-up to a policy
  - In this work, it can be the lead-up to a conclusion
  - Or an establishment of fact
- Call to action

Which of Knaflic's Advice is Most Relevant for this Communication?

- Storyboard
- Motivate: identify a problem/question/tension
- The evidence
  - In Knaflic's book this is the lead-up to a policy
  - In this work, it can be the lead-up to a conclusion
  - Or an establishment of fact
- Call to action
  - people want a resolution
  - make sure these relate to evidence

Which of Knaflic's Advice is Most Relevant for this Communication?

- Storyboard
- Motivate: identify a problem/question/tension
- The evidence
  - In Knaflic's book this is the lead-up to a policy
  - In this work, it can be the lead-up to a conclusion
  - Or an establishment of fact
- Call to action
  - people want a resolution
  - make sure these relate to evidence
- All parts should be linked

Helpful Tips You Can Apply

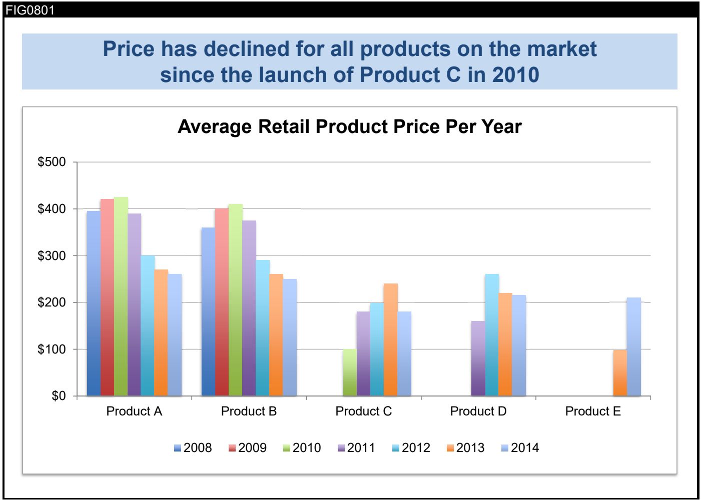
- Do slide headers read as a story? aka horizonal alignment
- Vertical alignment – within slide agreement
- Use headers to work out your story, then build inside
- Be wary that things that work for a presentation don't always work for a static paper product

Common Pitfalls

- Failure to motivate problem or issue
- Too little definition
- Too much information
- Conclusion without evidence

Telling a Story with Graphics

FIG0801



Price has declined for all products on the market since the launch of Product C in 2010

Telling a Story with Graphics

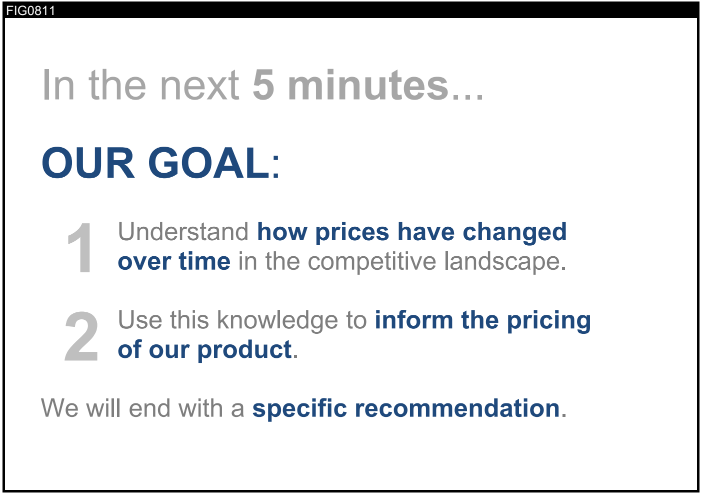


FIG0811

In the next **5 minutes**...

OUR GOAL:

1 Understand **how prices have changed over time** in the competitive landscape.

2 Use this knowledge to **inform the pricing of our product**.

We will end with a **specific recommendation**.

Telling a Story with Graphics

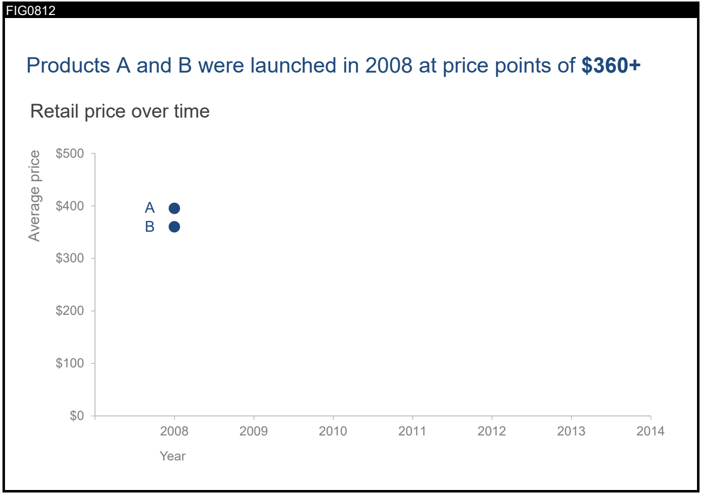


FIG0812

Products A and B were launched in 2008 at price points of **$360+**

Retail price over time

Telling a Story with Graphics

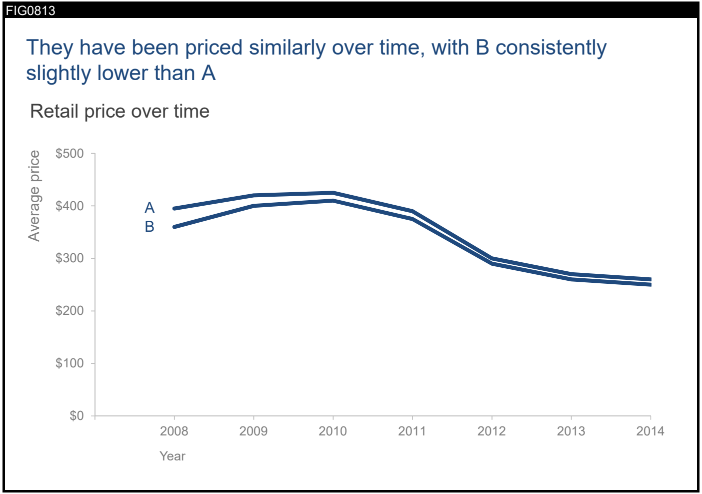


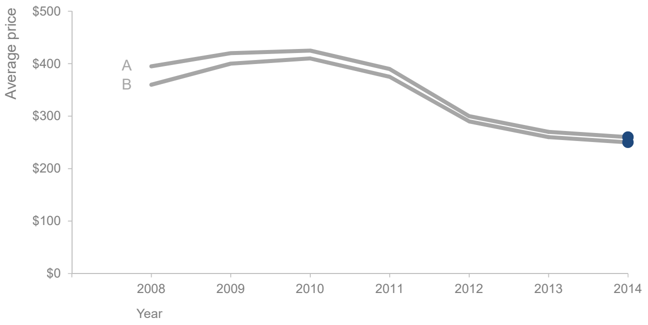
FIG0813

They have been priced similarly over time, with B consistently slightly lower than A

Telling a Story with Graphics

FIG0814

In 2014, Products A and B were priced at **$260** and **$250**, respectively



Retail price over time

Telling a Story with Graphics

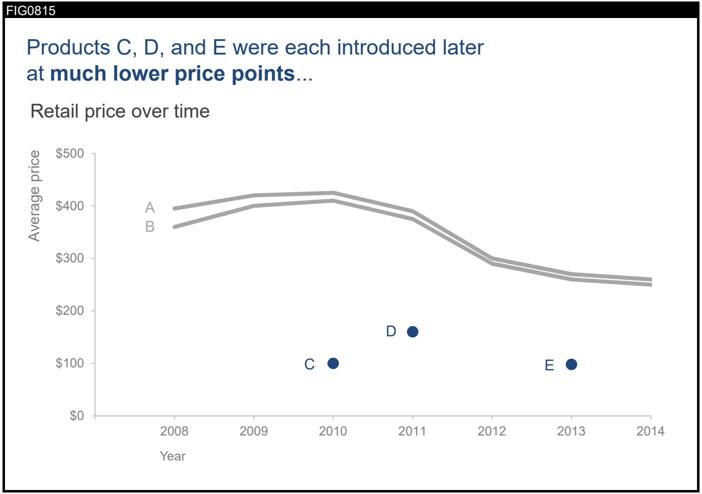


FIG0815

Products C, D, and E were each introduced later at **much lower price points**...

Retail price over time

Telling a Story with Graphics

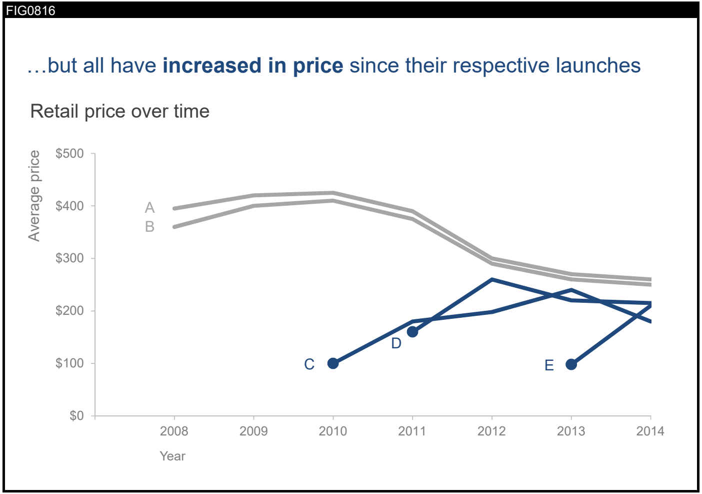


FIG0816

…but all have **increased in price** since their respective launches

Retail price over time

Telling a Story with Graphics

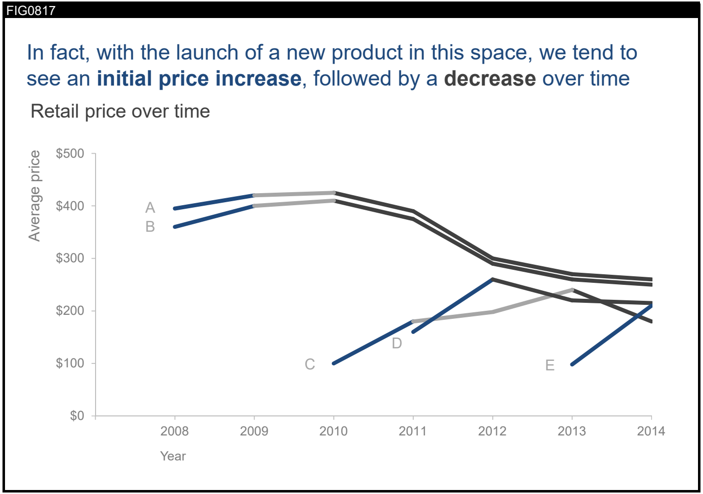


FIG0817

In fact, with the launch of a new product in this space, we tend to see an **initial price increase**, followed by a **decrease** over time

Retail price over time

Telling a Story with Graphics

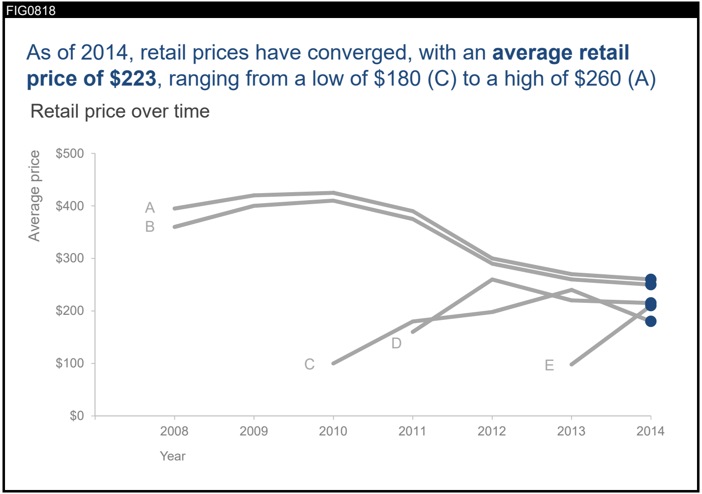


FIG0818

As of 2014, retail prices have converged, with an **average retail price of $223**, ranging from a low of $180 (C) to a high of $260 (A)

Retail price over time

Telling a Story with Graphics

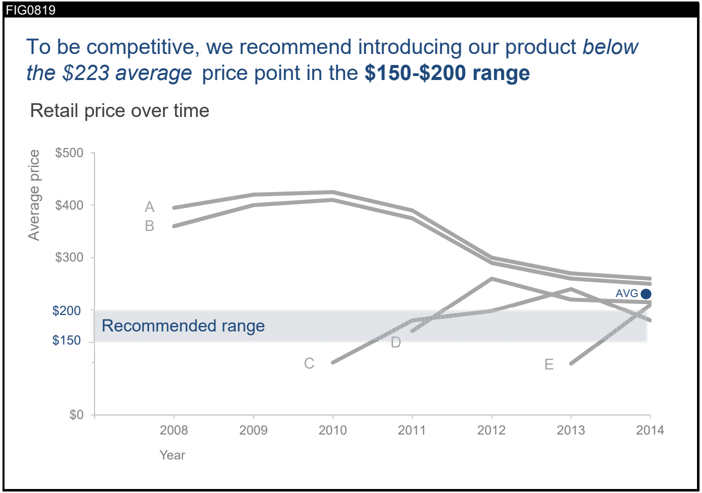


FIG0819

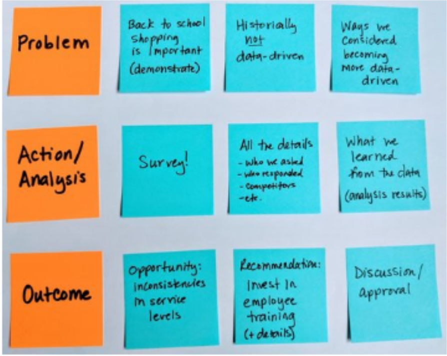
To be competitive, we recommend introducing our product *below the $223 average* price point in the **$150-$200 range**

Be Aware that the Presentation Version is Not the Print Version

- The final graph of the sequence just before is not a good explanation
- You may need to limit the points along the way
- And make sure you highlight the finding

Telling a Story with Post-its

- Goal today is brainstorming
- Write down your key points
- One per post-it note
- Re-organize and delete as needed
- Tell your story to your breakout group
- I'll wait five minutes and then pop in to each group



Problem	Back to school shopping is important (demonstrate)	Historically not data-driven	Ways we considered becoming more data-driven
Action/ Analysis	Survey!	All the details - who we asked - who responded - competitors - etc.	What we learned from the data (analysis results)
Outcome	Opportunity: inconsistencies in service levels	Recommendation: Invest in employee training (+ details)	Discussion/ approval

From Knaflic's webpage

R

To last slide

Today in R

1. Why Functions?
2. Defining a function
3. Getting things out of a function
4. Modifying a dataframe
5. Functions and ggplot

Why Functions?

- Many times, you need to repeat very similar code
- You can copy and paste, but ..
  - Subject to error when you make your small changes
  - A real bother when you need to change things
- For example

Why Functions?

- Many times, you need to repeat very similar code
- You can copy and paste, but ..
  - Subject to error when you make your small changes
  - A real bother when you need to change things
- For example
  - Make many similar graphs
  - Load multiple files with similar names
  - Create summary stats with different subsets

Good Functions

1. Make code more readable
2. Avoid coding errors
3. Make you more productive

From “Nice R Code” on github.

However: Never Start Writing a Function by Writing a Function

- Get one version of your code working first
- Then build the function
- When you've been programming for two years, try the function first

What We Cover About Functions

1. Defining a function
2. Calling a function
3. Getting things out of a function
4. Modifying a dataframe
5. Functions and `ggplot`

1. Defining a Function

```r
function.name <- function(arg1, arg2){
  # stuff your function does
}
```

- `function.name`: what you call the function
- `function`: needed to tell R this is a function
- `arg1`: first argument of the function
- `arg2`: second argument of the function
- inside the curly braces: what you want the function to do

Simple Function Example

```r
summer <- function(x,y){
  x^y
}
```

- function name?
- arguments?
- body of the function?

2. Calling a Function

```
summer <- function(x,y){
  x^y
}
```

```
summer(x = 2,y = 3)
```

2. Calling a Function

```
summer <- function(x,y){
  x^y
}
```

```
summer(x = 2,y = 3)
```

```
## [1] 8
```

2. Calling a Function

```
summer <- function(x,y){
  x^y
}
```

```
summer(x = 2,y = 3)
```

```
## [1] 8
```

```
summer(x = 3,y = 2)
```

2. Calling a Function

```r
summer <- function(x,y){
  x^y
}
```

```r
summer(x = 2,y = 3)
```

```
## [1] 8
```

```r
summer(x = 3,y = 2)
```

```
## [1] 9
```

3. Getting things out of a function

- Suppose you want to use the output of `summer` elsewhere in your program
- Functions “return” the last line
- “Return” means makes a value that exists outside of the function
- Best explained via example

3. Getting things out of a function

- Suppose you want to use the output of `summer` elsewhere in your program
- Functions “return” the last line
- “Return” means makes a value that exists outside of the function
- Best explained via example

However, if you save a graph with `ggsave()` in the function, that will exist outside the function.

What Gets Returned, 1 of 4

```r
summer2 <- function(x,y){
  o1 <- x^y
  o1
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
}
```

```r
summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What Gets Returned, 1 of 4

```r
summer2 <- function(x,y){
  o1 <- x^y
  o1
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
}
```

```r
summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What if I write o2?

What Gets Returned, 1 of 4

```
summer2 <- function(x,y){
  o1 <- x^y
  o1
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
}
```

```
summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What if I write o2?

```
o2
```

```
## Error in eval(expr, envir, enclos): object 'o2' not found
```

What Gets Returned, 2 of 4

```r
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
}
```

```r
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What Gets Returned, 2 of 4

```r
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
}
```

```r
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What if I call o3?

What Gets Returned, 2 of 4

```r
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
}
```

```r
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What if I call o3?

```r
o3
```

```
## [1] "o2 is 3"
```

What Gets Returned, 3 of 4

```
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  #print(paste0("o2 is ", o2))
}
```

```
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
```

What Gets Returned, 3 of 4

```r
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  #print(paste0("o2 is ", o2))
}
```

```r
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
```

What if I call o3?

What Gets Returned, 3 of 4

```r
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  #print(paste0("o2 is ", o2))
}
```

```r
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
```

What if I call o3?

```r
o3
```

```
## [1] 3
```

What Gets Returned, 4 of 4

```
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
  return(o2)
}
```

```
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What Gets Returned, 4 of 4

```
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
  return(o2)
}
```

```
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What if I call o3?

What Gets Returned, 4 of 4

```
summer2 <- function(x,y){
  o1 <- x^y
  print(paste0("o1 is ", o1))
  o2 <- x + y
  print(paste0("o2 is ", o2))
  return(o2)
}
```

```
o3 <- summer2(x = 1,y = 2)
```

```
## [1] "o1 is 1"
## [1] "o2 is 3"
```

What if I call o3?

```
o3
```

```
## [1] 3
```

4. What About Modifying a Dataframe?

```
# load north korean data
nkd <- data.frame(year = seq(2000,2017,1),
                  defectors = c(0,0,1,0,0,
                                0,0,0,2,0,
                                1,0,3,0,0,
                                1,1,4))
```

What About Modifying a Dataframe?

```
nkd
```

```
##    year defectors
## 1  2000         0
## 2  2001         0
## 3  2002         1
## 4  2003         0
## 5  2004         0
## 6  2005         0
## 7  2006         0
## 8  2007         0
## 9  2008         2
## 10 2009         0
## 11 2010         1
## 12 2011         0
## 13 2012         3
## 14 2013         0
## 15 2014         0
```

First Try

```
addone <- function(fixyear){
  nkd$defectors <- ifelse(test = nkd$year == fixyear,
                          yes = 100,
                          no = nkd$defectors)
}
```

First Try

```r
addone <- function(fixyear){
  nkd$defectors <- ifelse(test = nkd$year == fixyear,
                          yes = 100,
                          no = nkd$defectors)
}
```

How do you call this?

First Try

```
addone <- function(fixyear){
  nkd$defectors <- ifelse(test = nkd$year == fixyear,
                          yes = 100,
                          no = nkd$defectors)
}
```

How do you call this?

```
addone(fixyear = 2002)
addone(fixyear = 2005)
nkd
```

```
##   year defectors
## 1 2000         0
## 2 2001         0
## 3 2002         1
## 4 2003         0
## 5 2004         0
```

Second Try: Successful

```
addone <- function(fixyear){
  nkd$defectors <- ifelse(test = nkd$year == fixyear,
                          yes = 100,
                          no = nkd$defectors)
  return(nkd)
}
nkd <- addone(fixyear = 2002)
nkd <- addone(fixyear = 2005)
nkd
```

```
##    year defectors
## 1  2000         0
## 2  2001         0
## 3  2002       100
## 4  2003         0
## 5  2004         0
## 6  2005       100
## 7  2006         0
```

Create a new variable as a function of an old variable

This doesn't work:

```r
multiplypls <- function(varo){
  nkd$new_varo <- nkd$varo * 5
  return(nkd)
}
nkd2 <- multiplypls(var = defectors)
```

R doesn't know to plug in defectors for `varo` in the `dataframe$variable` construction

Create a new variable as a function of an old variable

Instead

```
multiplypls <- function(varo){
  nkd[[paste0("new_",varo)]] <- nkd[[varo]] * 5
  return(nkd)
}
nkd2 <- multiplypls(var = "defectors")
head(nkd2)
```

```
##   year defectors new_defectors
## 1 2000         0             0
## 2 2001         0             0
## 3 2002       100           500
## 4 2003         0             0
## 5 2004         0             0
## 6 2005       100           500
```

5. And a Word of Warning About `ggplot()`

- many `tidyverse` commands, including `ggplot()` use non-standard evaluation
- for your purposes, that means that these command don't always work in expected ways in functions
- BUT there are work-arounds – see tutorial

Bottom Line

- Use functions!
- Write a non-function example first
- Test
- Write the function
- Check output

Next Lecture

- Next week: Maps 2 of 2
- Read
  - Monmonier, Chapter 6
  - Goats from the *Post*
  - *NYT* on elections maps
- Heads-up: In-class workshop March 29 – lecture 9