Lecture 6: Functions and Storytelling

March 18, 2019

Overview

Course Administration

Good, Bad and Ugly

Telling a Story

Maps in R

Course Administration

1. Sign up for consultations!
   - sign up for slots April 8, 10 or 11
   - no class meeting April 15
2. In-class workshop April 8: handout today
3. Anything else?

Class 7, March 25: Good Bad and Ugly

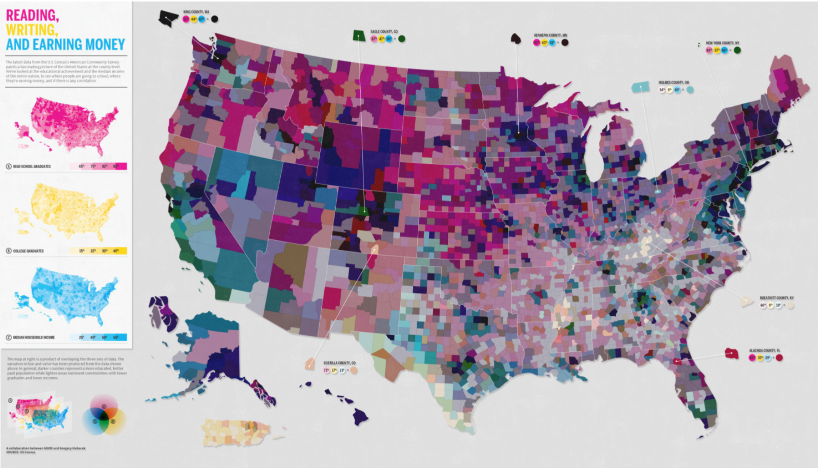
Send by 9 am next Monday. See if you can find a story-telling graphic.

- MF
- IT

This Week's Good Bad and Ugly

- EW
- MP

McCall's Example



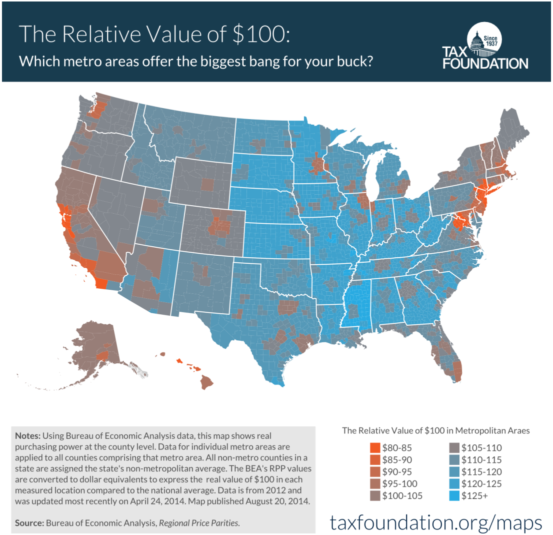
READING, WRITING, AND EARNING MONEY

The latest data from the U.S. Census's American Community Survey paints a fascinating picture of the United States at the county level. We've looked at the educational achievement and the median income of the entire nation, to see where people are going to school, where they're earning money, and if there is any correlation.

The map at right is a product of overlaying the three sets of data. The variation in hue and value has been produced from the data shown above. In general, darker counties represent a more educated, better paid population while lighter areas represent communities with fewer graduates and lower incomes.

A collaboration between GOOD and Gregory Hubacek
SOURCE: US Census

Ellen’s Example



The Relative Value of $100:

Which metro areas offer the biggest bang for your buck?

TAX FOUNDATION

Since 1937

Notes: Using Bureau of Economic Analysis data, this map shows real purchasing power at the county level. Data for individual metro areas are applied to all counties comprising that metro area. All non-metro counties in a state are assigned the state's non-metropolitan average. The BEA's RPP values are converted to dollar equivalents to express the real value of $100 in each measured location compared to the national average. Data is from 2012 and was updated most recently on April 24, 2014. Map published August 20, 2014.

Source: Bureau of Economic Analysis, *Regional Price Parities.*

The Relative Value of $100 in Metropolitan Araes

- ■ $80-85
- ■ $85-90
- ■ $90-95
- ■ $95-100
- ■ $100-105
- ■ $105-110
- ■ $110-115
- ■ $115-120
- ■ $120-125
- ■ $125+

taxfoundation.org/maps

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?

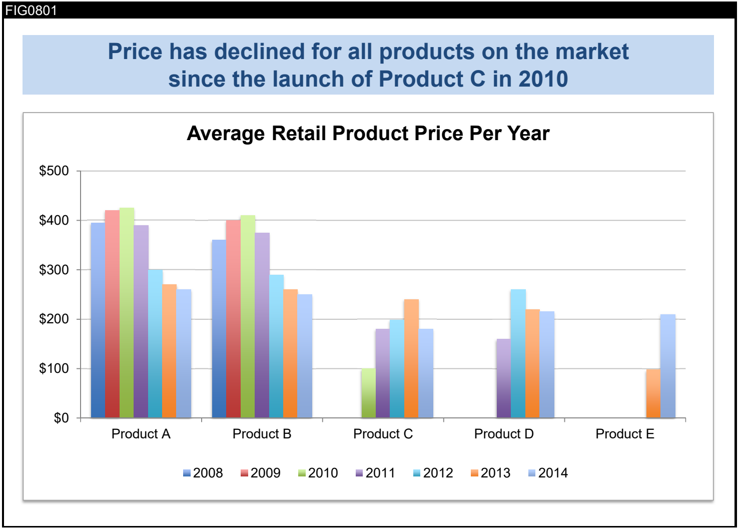
- Storyboard
- Motivate: identify a problem/question/tension
- The evidence
  - In Knaflic's book this is the lead-up to a policy
  - In this work, it can be the lead-up to a conclusion
  - Or an establishment of fact
- Call to action
  - people want a resolution
  - make sure these relate to evidence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

Common Pitfalls

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

Telling a Story with Graphics

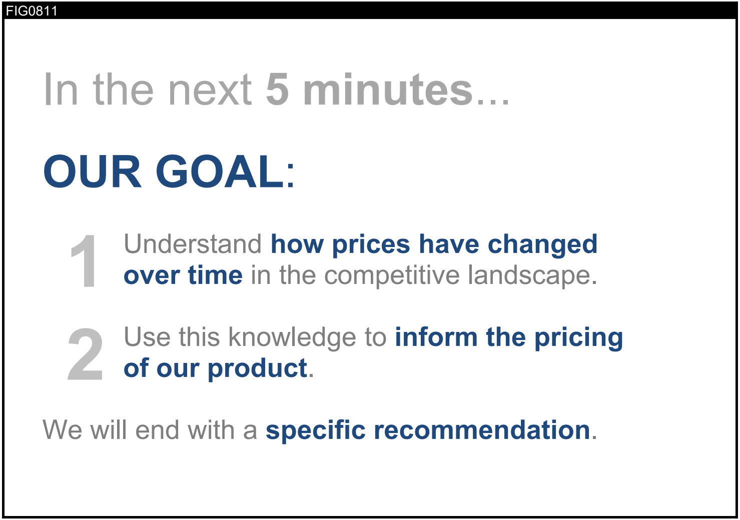
FIG0801



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:

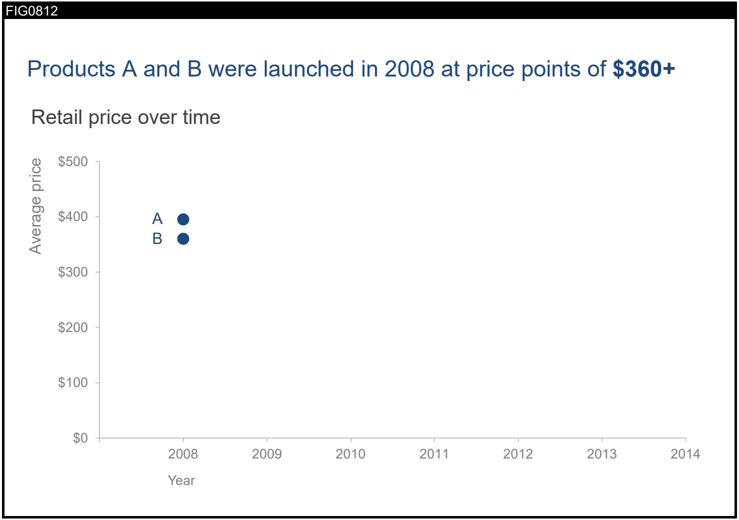
1 Understand **how prices have changed over time** in the competitive landscape.

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

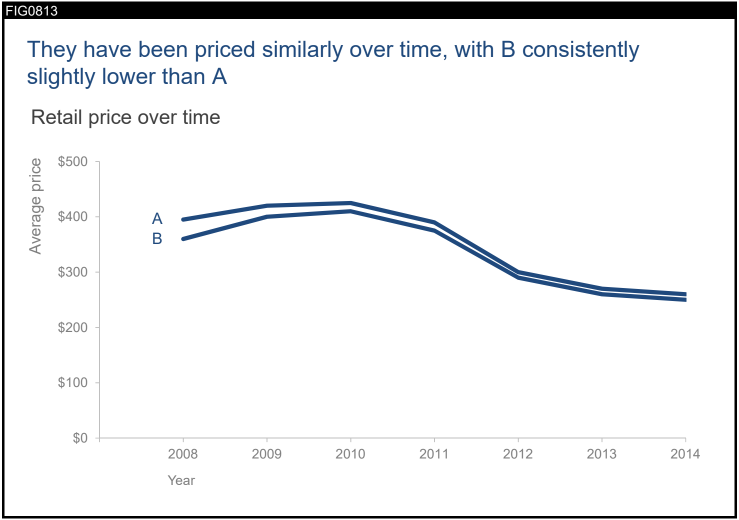


FIG0813

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

Telling a Story with Graphics

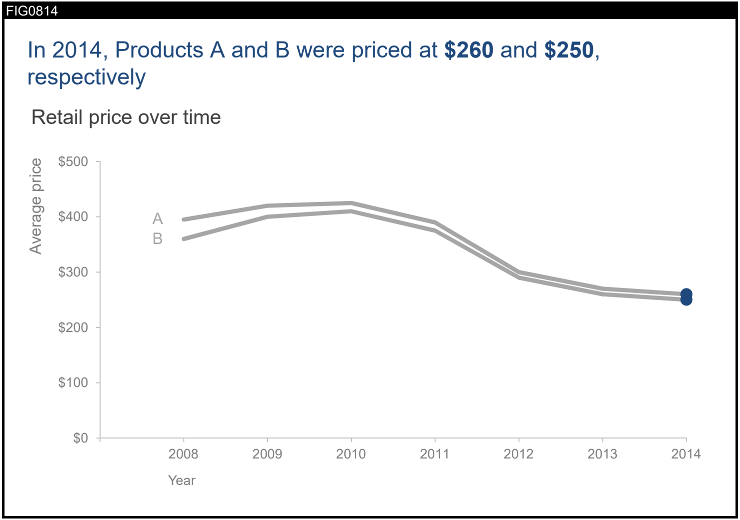


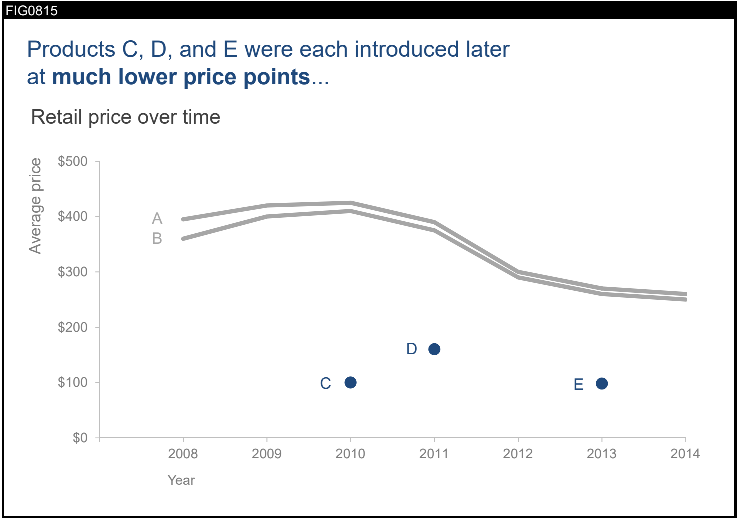
FIG0814

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

Retail price over time

Telling a Story with Graphics

FIG0815

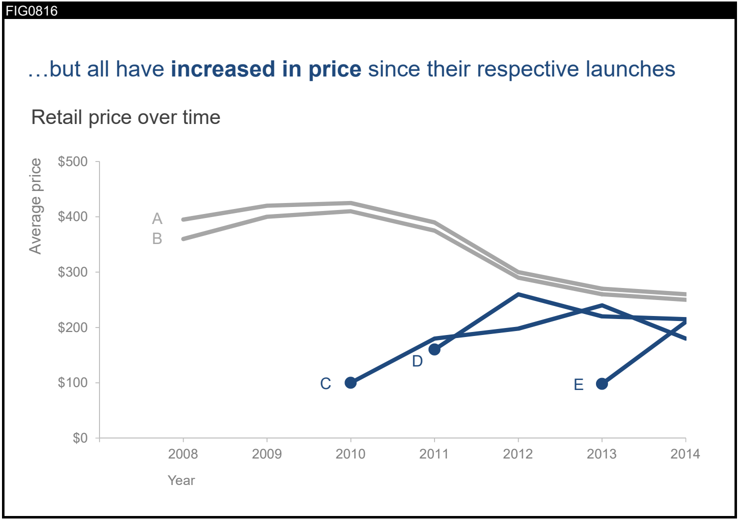


Products C, D, and E were each introduced later at **much lower price points**...

Retail price over time

Telling a Story with Graphics

FIG0816

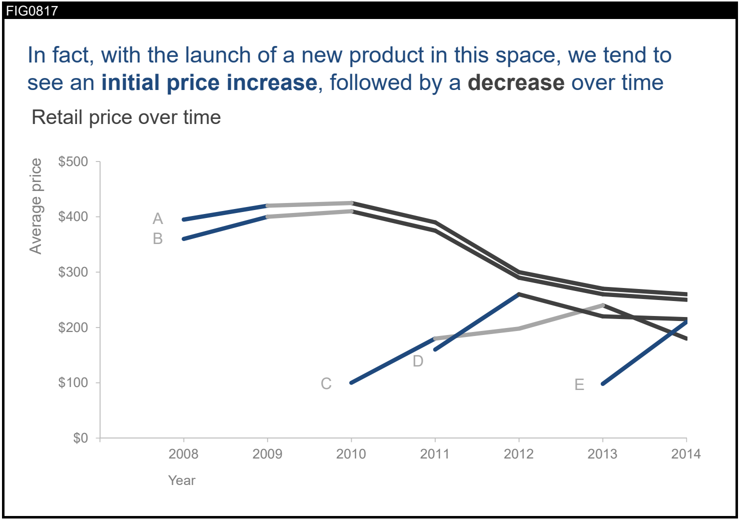


…but all have **increased in price** since their respective launches

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

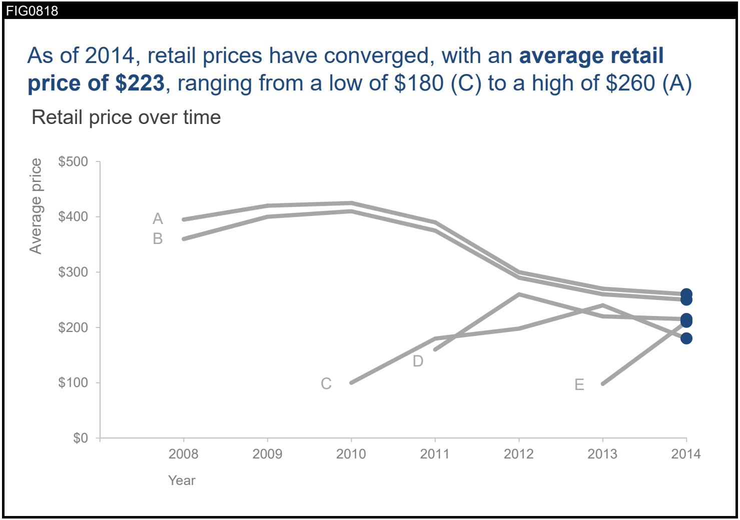


FIG0818

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

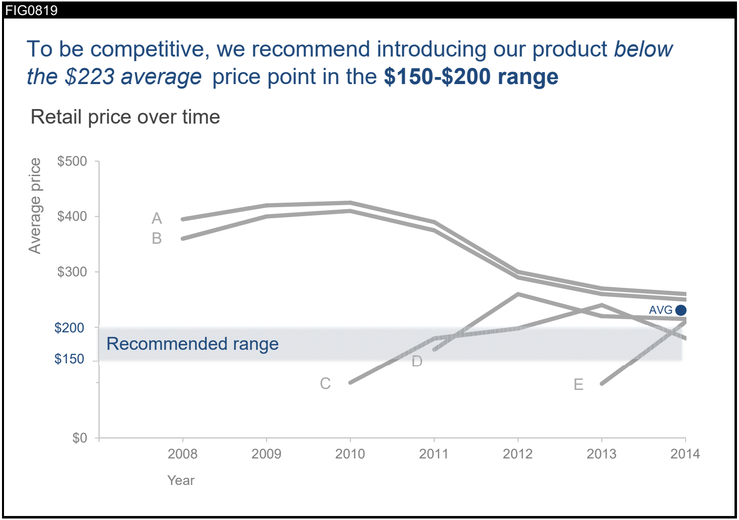


FIG0819

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

Retail price over time

R

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 with 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

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
- inside the curly braces: what you want the function to do

Simple Function Example

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

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

Calling a Function

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

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

Calling a Function

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

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

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

Calling a Function

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

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

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

```
summer(3,2)
```

Calling a Function

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

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

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

```
summer(3,2)
```

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

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

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

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

```
summer2(1,2)
```

```
## [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(1,2)
```

```
## [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(1,2)
```

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

What if I write o2?

```
o2
```

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

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(1,2)
```

```
## [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(1,2)
```

```
## [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(1,2)
```

```
## [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

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

```r
o3 <- summer2(1,2)
```

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

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(1,2)
```

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

What if I call o3?

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(1,2)
```

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

What if I call o3?

```
o3
```

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

What Gets Returned, 4 of 4

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

```r
o3 <- summer2(1,2)
```

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

What Gets Returned, 4 of 4

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

```r
o3 <- summer2(1,2)
```

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

What if I call o3?

What Gets Returned, 4 of 4

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

```r
o3 <- summer2(1,2)
```

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

What if I call o3?

```r
o3
```

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

What About Modifying a Dataframe?

```
# load north korean data
nkd <- data.frame(year = c("2000","2001","2002","2003",
                           "2004","2005","2006","2007",
                           "2008","2009","2010","2011",
                           "2012","2013","2014","2015",
                           "2016","2017"),
                  defectors = c("0","0","1","0","0",
                                "0","0","0","2","0",
                                "1","0","3","0","0",
                                "1","1","4"))
nkd
```

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

Modifications don't come out here

```r
addone <- function(fixyear){
  nkd$defectors <- ifelse(nkd$year == fixyear,
                          100,
                          nkd$defectors)
}
```

Modifications don't come out here

```
addone <- function(fixyear){
  nkd$defectors <- ifelse(nkd$year == fixyear,
                          100,
                          nkd$defectors)
}
```

How do you call this?

Modifications don't come out here

```
addone <- function(fixyear){
  nkd$defectors <- ifelse(nkd$year == fixyear,
                          100,
                          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
```

But modifications do come out here

```r
addone <- function(fixyear){
  nkd$defectors <- ifelse(nkd$year == fixyear,
                          100,
                          nkd$defectors)
  return(nkd)
}
nkd <- addone(fixyear = 2002)
nkd <- addone(fixyear = 2005)
nkd
```

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

Bottom Line

- Use functions
- Check output

Next Lecture

- Next week: Maps 2 of 2
- Read Monominier, Chapter 6 and NYT mapping article