GEOG 491/891: Special Topics - Spatial Analysis in R

Week 04.01: Writing functions and reproducible code

Dr. Bitterman

Today's schedule

- Open discussion
- Lab 1 check-in
- Building blocks of reproducible code

Anything to discuss? Questions?

What makes our code "reproducible"?

We've made some big leaps, but we haven't established a formalized vocabulary just yet

Let's start with indexing our vectors

```
library(tidyverse)

### daily rainfall in inches of rain
rainfall <- c(0.0, 2.1, 2.5, .1, 0.0, 0.0, 6.8, 3.1, 2.2)
```

How do we gt the first element?

Indexing

```
# find the first element

rainfall[1]
```

Let's assume a "big" storm is one with 3" or more in a day... how do we check?

A simple logical test

```
# a VERY big storm is one 3" or greater, so let's check

rainfall[1] >= 3
```

what outputs are we limited to?

A simple if-else block

```
# Let's make an if-else block
if(rainfall[1] >= 3){
  print("big storm")
} else{
  print("little storm")
}
```

limitations of this method?

how does it generalize across the dataset?

Let's first wrap the if-else block in a function and replace the variables

```
f.storm.test <- function(rainfallAmount){
  if(rainfallAmount >= 3){
    print("big storm")
  } else{
    print("little storm")
  }
}
```

what does this code do?

Once you write a function, how do you use it?

A function with a loop

```
f.storm.test <- function(rainfallAmount){
  if(rainfallAmount >= 3){
    print("big storm")
  } else{
    print("little storm")
  }
}

for(i in rainfall){
  f.storm.test(i)
}
```

A note on loops

- R loops are inefficient
- Instead of appending the output of a loop to a dataset...
- they make a complete new copy
- So use them in small cases, not large ones

the `purrr::map` function is powerful, but confusing at times

the "tidy way"

```
rainfall %>% purrr::map(., f.storm.test)
```

How did your output differ?

There's also the good 'ol vectorized way of doing things

```
rainfall >= 3
```

if-else blocks vs. ifelse

```
if(sometest){
  do something in here
}else{
  do something else here
}
```

vs

```
ifelse(sometest, do something, do something else)
```

...functionally equivalent, really up to you

Finding the greatest rainfall day

Ideas?

A simple way

```
max(rainfall)
```

But which day is that?

The `which` command can be useful

Let's breakdown a weird syntax first...

```
which(rainfall == max(rainfall))
```

What happened?

A tidy way of working with data.frames

```
mydf <- read_csv("./data/ne_counties.csv")
glimpse(mydf)
```

Find the maximum median housing value

```
max(mydf$MedValHous)
```

Which works the same way, but not super useful

```
which(mydf$MedValHous == max(mydf$MedValHous))
```

what's the output - and what does it mean?

Let's make it a bit more confusing

Break it down step-by-step

```
which(mydf$MedValHous == max(mydf$MedValHous)) %>% mydf[.,]
```

How exactly does this work?

A contrived question

for each county, calculate how much its median housing value is LESS than the max value in the dataset

Thoughts?

Easier than you'd think

Break it down

```
newdf <- mydf %>% mutate(deviation = MedValHous - max(MedValHous))
```

plot it

```
newdf %>% ggplot(., aes(x = deviation)) +
  geom_histogram() +
  theme_minimal()
```

Review and next class

- Any questions?
- This week's readings/tasks:
  - Chapter 4 in textbook
  - Practice, practice, practice
  - Keep working on Lab 1
- IDEAS FOR OUR WILDCARD FRIDAY?