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

Week 01.02: A quick and limited introduction to R

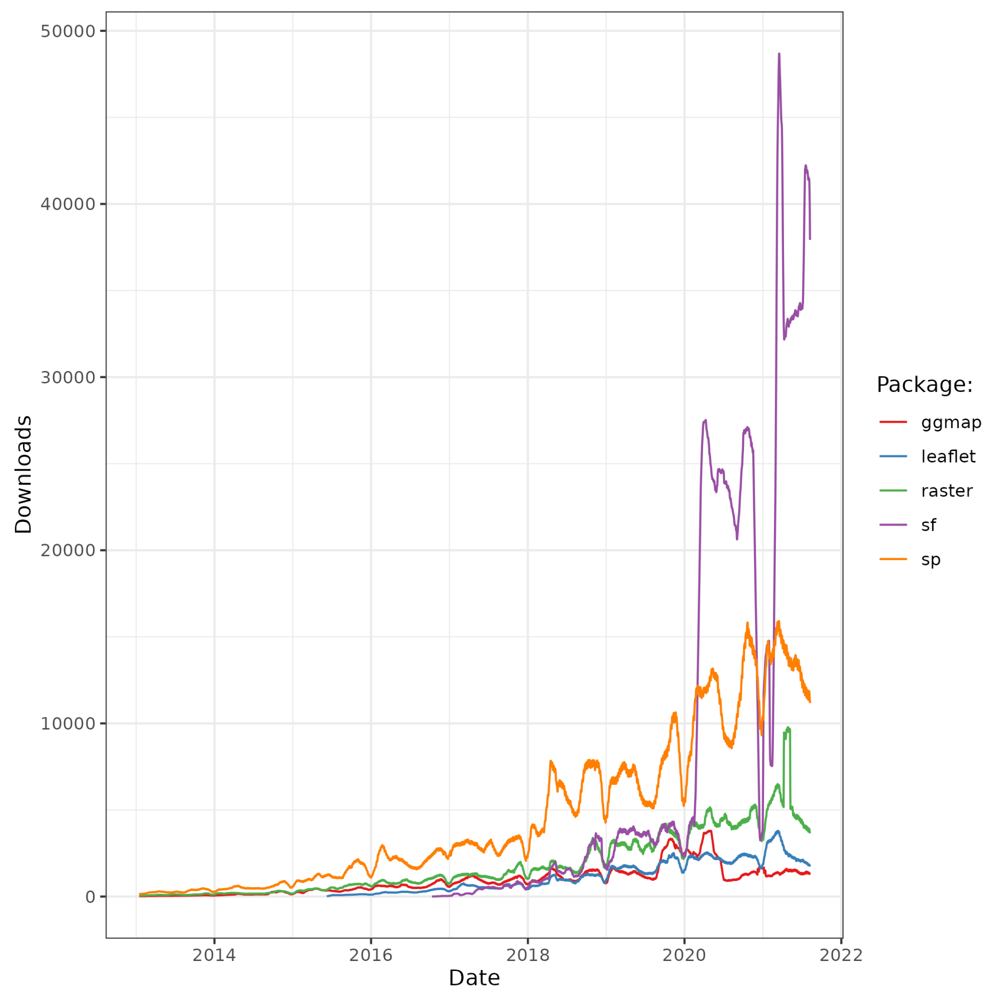
Dr. Bitterman

Today's schedule

- Open discussion
- R basics and practice

Anything to discuss? Questions?

A quick bit of background



https://geocompr.robinlovelace.net/intro.html#rs-spatial-ecosystem (for more)

Let's review some GIScience

- Wait, what's GIScience?
- And how's it different than GIS?

A quick review

- Components of a GIS?
- Data types?
- Spatial functions?

Let's get started with some R

1. Open RStudio
2. Create a new project in a temporary working directory (it can be anywhere)
    a. File -> new Project
    b. What do you see?
    c. what's the `>`?
3. Writing in the console vs. writing code in a script

Some simple work

type the following in the console

```
x <- 7
```

- the `<-` is the "assignment operator"
- `=` also works, but can be incorrect in rare occasions
- so use `<-`

Did anything else change?

So we have a numeric value stored as a variable

Let's do stuff with it

Try the following

```
x + 2
x * 8
x / 1
x ** 2 # what does this do? <- and what is this ancillary text?
x/2 == 0 # what kind of test is this?
```

Working with vectors

A vector is a 1-D ordered collection of values, you designate a vector in R with `c()`

For example, `y <- c(1, 2, 3, 4, 5)`

try it!

What does your environment viewer tell you?

R operations are "vectorized"

this means you can do things like this:

STOP: What do you think is going to happen?

```
y + 2
```

or:

```
c(1,2,3) + c(4,5,6)
```

But a vector can't mix data types

What do you think will happen if we try:

```r
z <- c(1, 2, "3")
```

what DID happen?

Much of R is built on vectors, a lot is also built on lists

Lists can mix and match data types

```
mylist <- list(1, 2, "banana")
```

Vectorized operations don't work on lists...

getting elements from a vector or a list

```
mylist[2]
```

what does this return?

and what does it tell you about R data structures?

data frames

- A 2-dimensional data structure that has a lot in common with a common "table"
- Functionally, it's a list of lists

Let's break this down first. What do we expect to happen? How does the syntax work?

```
mydf <- data.frame(names = c("Huey", "Dewey", "Louis"),
  height = c(45, 43, 44))

#then...

mydf
```

What do you see in the console?

Packages

- Collections of code, function, and data written by others
- The foundation of the R ecosystem
- Need to be "installed" once
- Then need to read into memory for each session
- Packages of packages are a thing

```r
#install it
install.packges("tidyverse") # Quotes here

#load it into memory
library(tidyverse) # no quotes here
```

Calling a function

- once it's in memory, you can call a function directly

```
filter(mydf, height > 43)
```

- but namespace conflicts happen, so you can be explicit too

```
dplyr::filter(mydf, height > 43)
```

Getting help on a function

```
?dplyr::filter
```

or on a package

```
?dplyr
```

Try it, what happens?

Writing your own function

syntax is a bit weird, so let's break it down

```r
myfirstfunction <- function(x, y){
  x + y
}
```

then call the function (make sure it's in memory first)

```r
myfirstfunction(4, 8)
```

If there's time...

- In small groups, figure out how you'd do the following:
- Write a function that takes two integers. If **both are even** or **both are odd**, the function returns **TRUE**. Otherwise, it returns **FALSE**

- Start with the algorithm, NOT the code
- Then try to write the function

Review and next class

- Any questions?
- Next week’s readings/tasks:
    - Chapter 2 in textbook
    - Review Hadley's book/site
    - Practice on your own
- Next week's topics: data structures, data munging, plots 101