




TRIA-FoR

Transformative Risk Assessment and Forest Resilience Using Genomic Tools for the Mountain Pine Beetle Outbreak

TRiA-FoR R Workshop

Overview

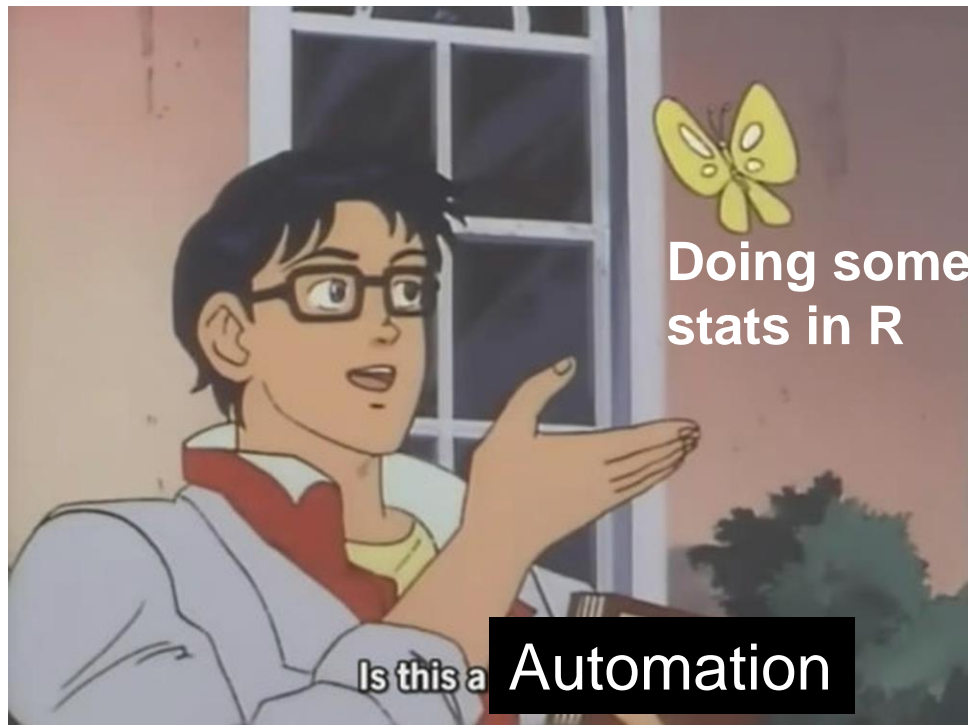
- PowerPoint on R and programming
 - R basics tutorial
 - A Mountain Pine Beetle problem set
 - Q&A / “office hours”
- 
- <1 hour

Why programming?

Automation

Computers are good at ...

- Being precise
- Repeating things



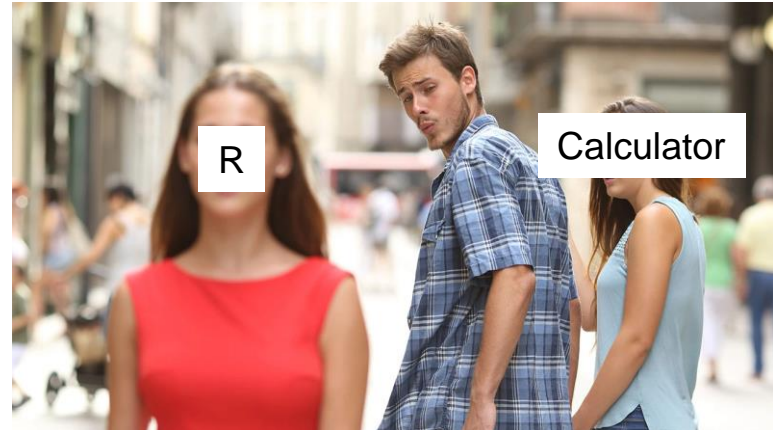
Example: calculating relative abundance of lodgepole pine

R code

```
df <- read.csv("file_name.csv")  
relative_abund <- lodge_abund / pine_abund  
print(relative_abund)
```

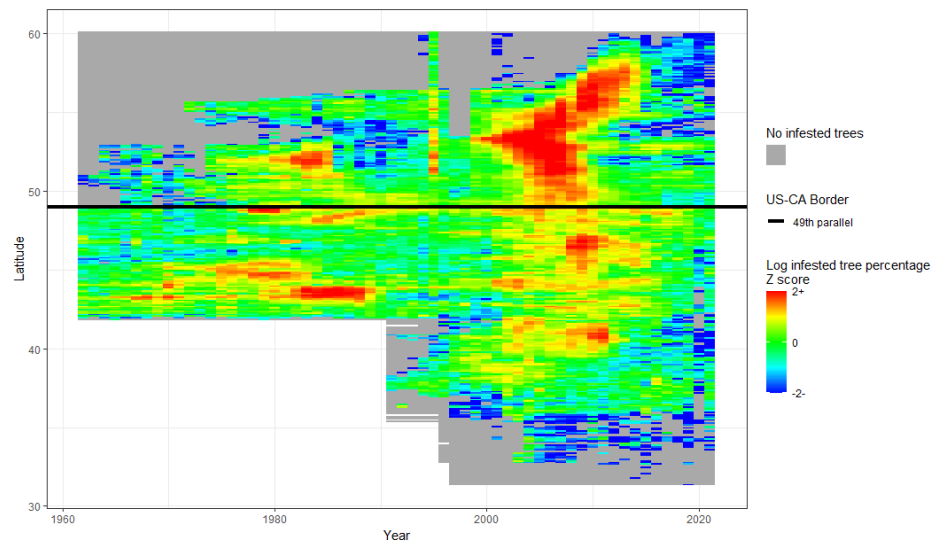
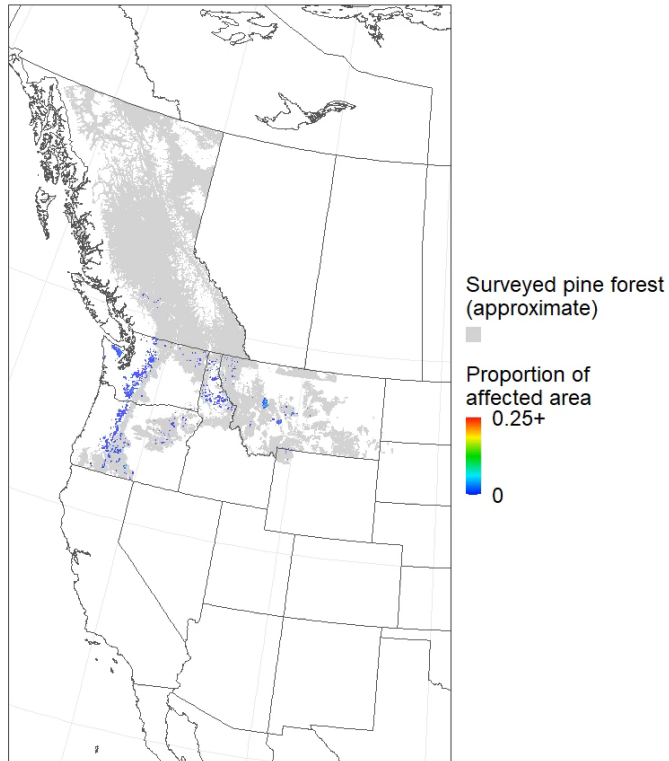
Data, in CSV format

	A	B
1	lodge_abund	pine_abund
2	10	25
3	12	75
4	4000	10113
5	325	430
6	2	3

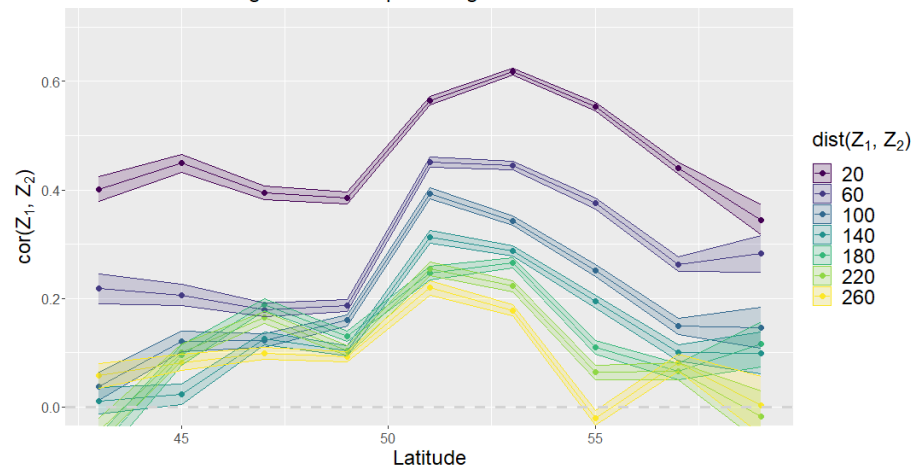


You can make cool things

Year: 1962



MPB spatial correlations across latitude, by distance
 Z := standardized log infested tree percentage



R and R Studio



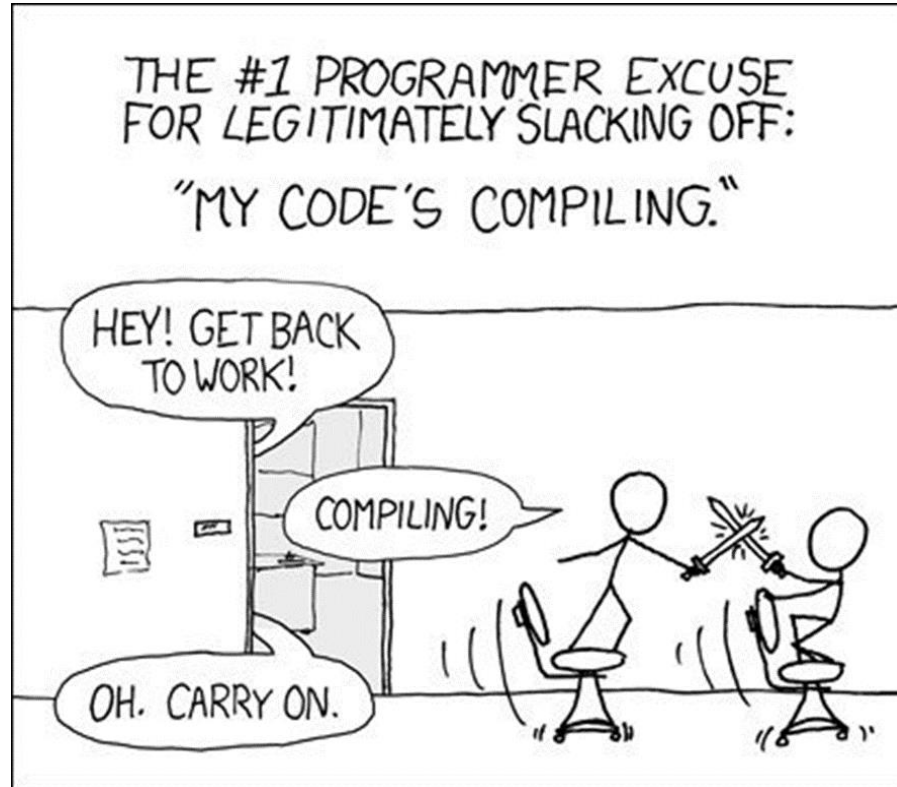
Every other programming language



Why R?

- It's free and open source
- Lots of Packages
- Your collaborators use R
- R Studio
 - A widely-used Interactive Development Environment (IDE)
- Designed for data wrangling and stats
- Interpreted language

Interpreted means you can run one line at a time



Why R?

- It's free and open source
- Lots of Packages
- Your collaborators use R
- R Studio
 - A universally used Interactive Development Environment (IDE)
- Designed for data wrangling and stats
- Interpreted language
- Dynamic typing

Dynamic typing means R tries to figure out what you mean



Static, declared typing

```
int x = 1
```

```
String y = "1"
```

```
x+y = ERROR
```



Dynamic typing

```
x = 1
```

```
y = "1"
```

```
x+y = 2
```

Cons

- Quirky
 - E.g. three ways to extract a column from a dataframe
- Slow
 - But packages can fix that

For loops in R

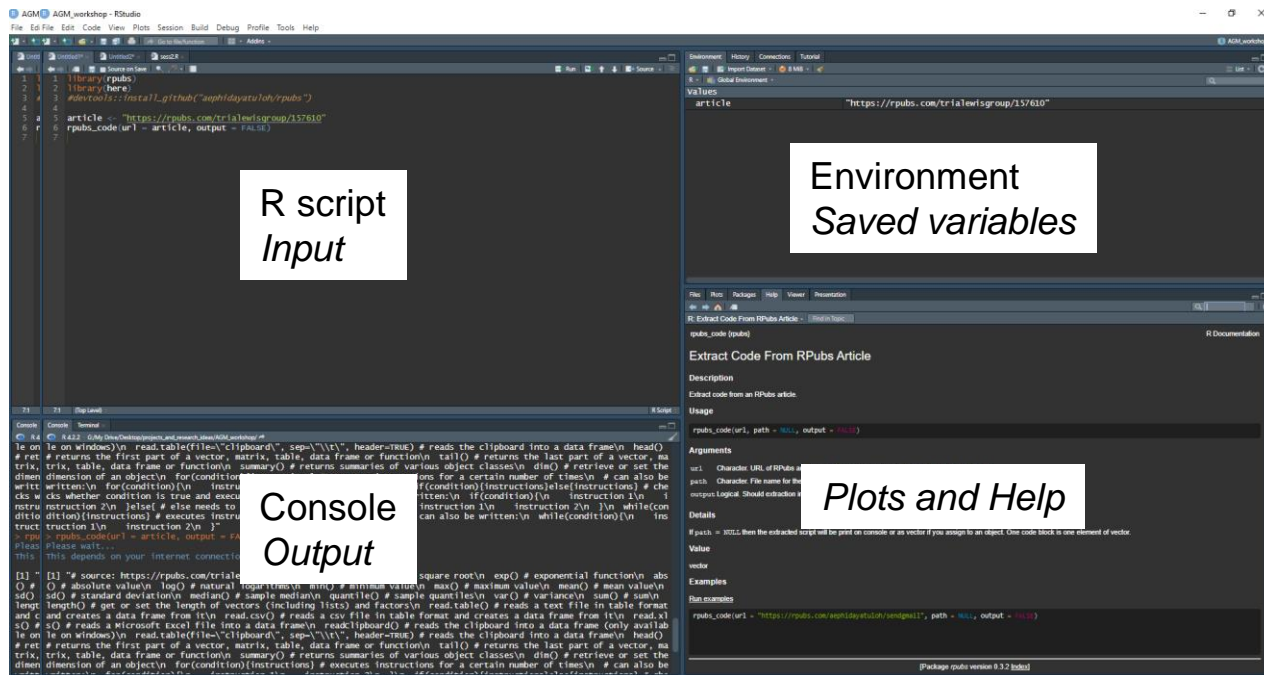
Learning Julia,
just for the fast
for loops

R + Rcpp package



Some things to know

- The windows in R studio

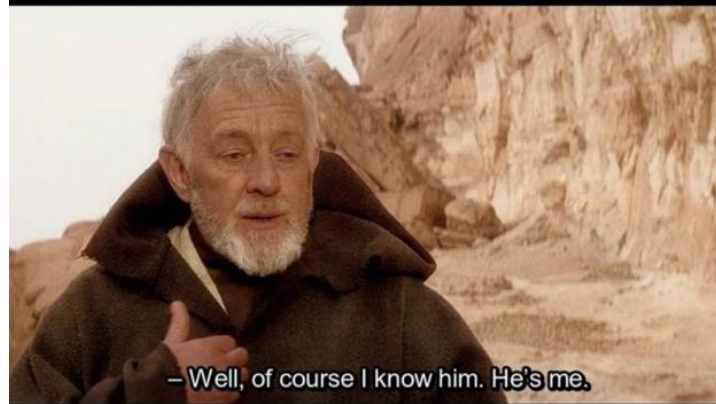


Some things to know

Good style matters. We will cover this as we go, but...

- Consistently use a variable naming convention
 - `my_vector_ab` NOT `my.VectorAb`
- **Comment your code**
- Space between operators
 - `log(3 + 2 * 5)` NOT `log(3+2*5)`

When you read some incredibly bad code, thinking "What moron wrote this...", but halfway through it starts to become familiar.



— Well, of course I know him. He's me.

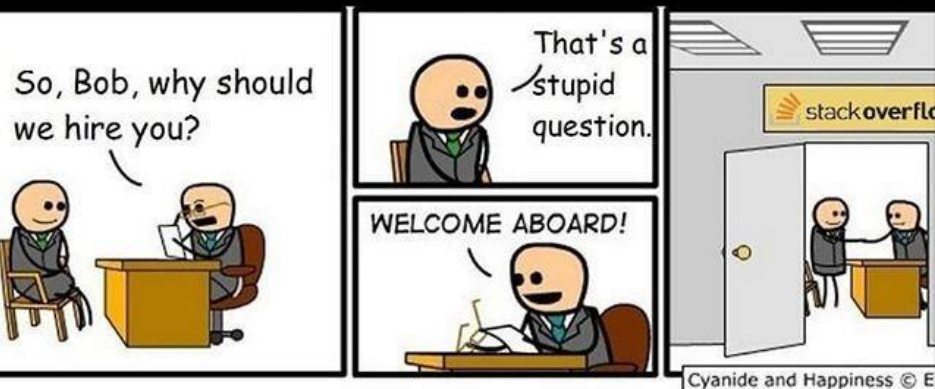
Keys to learning R

#1 Dive in

Resources

- Stack exchange
- Google
 - “site:https://stackoverflow.com/ [query]”
- ChatGPT
- Youtube
- Documentation. In the R console, write `?function_name`

Stack Overflow



ChatGPT

E

I'm afraid I have a stupid question. In R, how do I get the last thing in a vector?



There are no stupid questions! In R, it's common to have questions like this when you're learning or working with the language. To get the last element of a vector in R, you can use the ``length()`` function to find the length of the vector, and then use that to index into the vector. Here's an example:

R

Copy code

```
# create a vector
my_vector <- c(1, 2, 3, 4, 5)

# get the last element of the vector
last_element <- my_vector[length(my_vector)]

# print the last element
print(last_element)
```

In this example, ``length(my_vector)`` returns 5 because there are 5 elements in ``my_vector``. Then ``my_vector[5]`` gets the 5th (last) element of ``my_vector``, which is 5 in this case.

Our hope for you

