Computational Methods in Astrophysics

Dr Rob Thacker (AT319E)

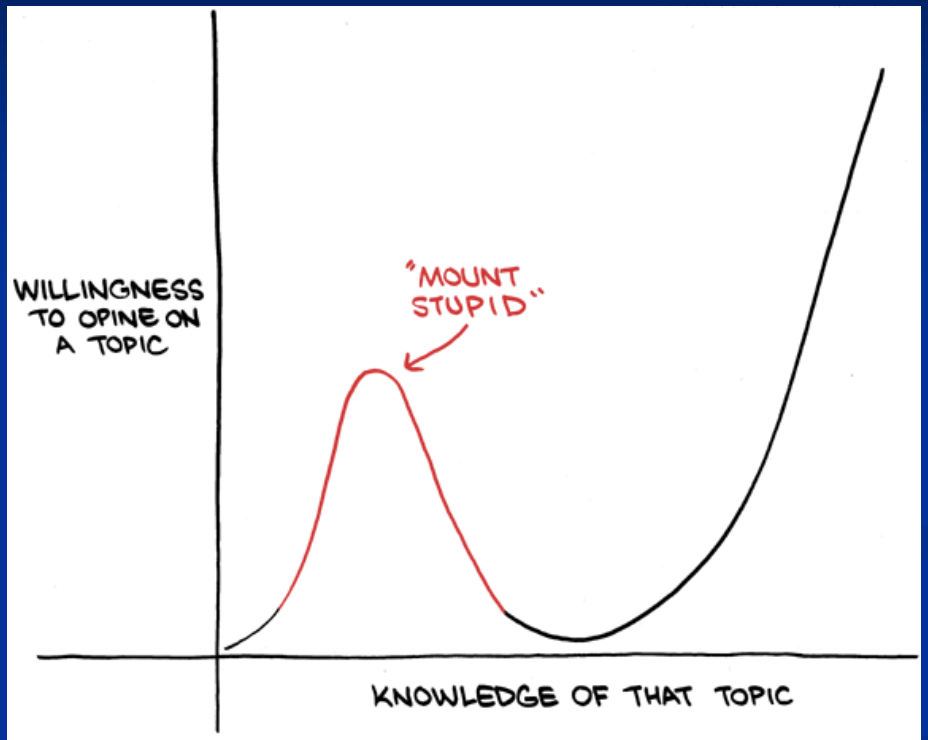
thacker@ap.smu.ca

Languages for data analysis

- Caveat: I am not a data scientist!
  - I am computational scientist
- Purpose of these next few lectures is to give you *exposure*
  - Get you to a place of familiarity but not expertise
    - Build specialty from there, if you need it
  - Think of it as a little like a series of intro workshops

Only half joking

From smbc-comics.com



Phrases uttered atop Mount Stupid:

- *"Historically, the Amazons would cut off their right breast so they could shoot a bow and arrow."*

- *"The American Civil War really had nothing to do with slavery."*

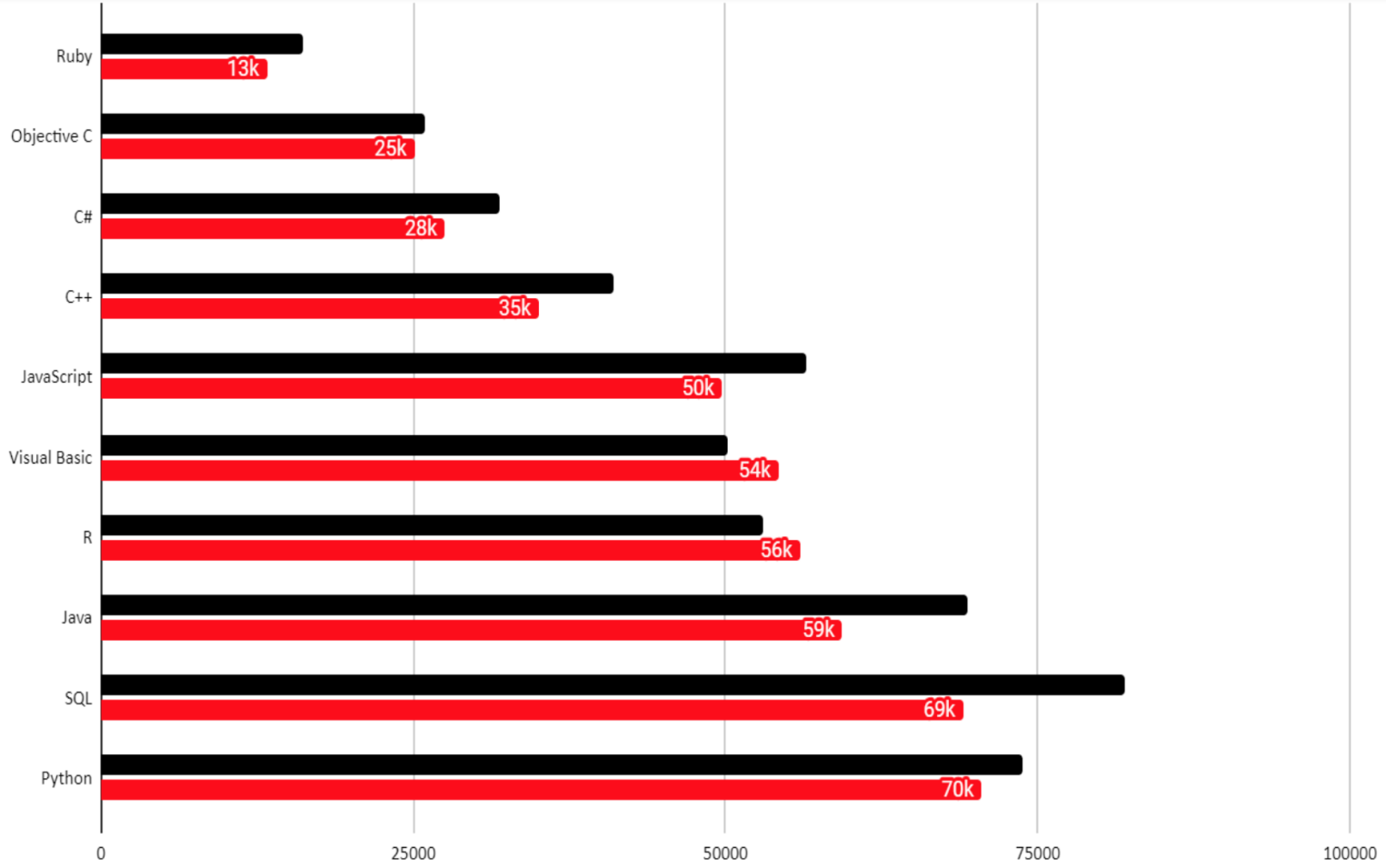
- *"Biologically, tomato is a fruit, not a vegetable."*

- 99% of phrases that start with *"Now, I don't know much about quantum physics, BUT-"*

- Also, check out Dunning-Kruger effect

Common Languages 2021/20

From
codingdojo.com



Drivers for (big) data analysis in astro

- Optical astronomy: LSST
  - 3.2 gigapixel camera, i.e. 1500 HDTV screens
  - Survey of the sky every three days
    - Each day = 2 Sloan surveys!!!
  - 0.4 Exabytes of data in final catalogue
- Radio: Square Kilometer Array (SKA)
  - Storing all spectral line cubes would require 27 exabytes a year
  - Technology and cost problem!

But there are problems in small data analysis too

- Claims of detections of extrasolar planets have been shown to be inaccurate
  - Bayesian/Frequentist discussions
- Recognition that we need to move beyond simple statistical approaches
- But this requires new analysis paradigms
  - Pressure to publish means that senior researchers often don't spend time learning new tools
    - People naturally want to continue doing what they know
  - Common issue in other fields/businesses

Python vs R

- Not a remotely fair comparison
- Python is general purpose, R specialist
  - They are very similar in age (20+ yrs), R slightly younger
- Let's restrict to the data science discussion
- Python has more flexibility, R has more built-in packages
- Many feel R's visualization tools are more powerful
- R has a steeper learning curve if you don't have a computing background

Parallel execution in R

- Doesn't interface nicely with OpenMP
- Rcpp = interface between C++ and R
  - In theory you could then use OpenMP
  - But need to check that R code is thread safe!
- Packages multicore (unix forks), snow (mpi) and now 'parallel' offer some parallel support
- See `https://ljdursi.github.io/beyond-single-core-R/#/`
- Still a bit specialist right now, but more packages coming out all the time

R – Some History & Facts

- R is ironically an implementation of the "S" programming language + extra semantics
  - "S" Developed at Bell Labs in 1976
- Original implementation by Ross Ihaka & Robert Gentleman in 1993 (New Zealand)
- GNU GPL licence
- 50% of R is written in C, 30% FORTRAN (yes! I'm not kidding) and 20% in R itself
- Estimated user base is now several million and growing
- Very popular in biology, ecology, political sci… anywhere doing lots of stats

The R "ecosystem"

- There are also almost 8000 additional packages!
  - https://cran.r-project.org/web/packages/
- It has it's own journal!
  - https://journal.r-project.org/
- User mailing lists (including R-help)
  - https://www.r-project.org/mail.html
- Plenty of blogs and books
- And commercial companies providing support that you can purchase

Downloading R

- http://cran.r-project.org
- CRAN = Comprehensive R Archive Network
  - Just a distribution network
- Binaries for windows, mac, linux
  - Note I've found the linux versions sometimes have path issues on NFS systems
- Install the "base" system
- Can also download Rstudio (https://www.**rstudio**.com/)
  - Complete integrated development environment (IDE)
    - Editor for scripts etc
  - Need R installed first

R environment

- R provides a suite of tools for data manipulation, analysis and graphical display
  - Operators for arrays (and matrices)
  - Large collection of tools & functions for data analysis
  - On-screen or hardcopy graphical facilities
  - A simple and effective programming language
- The overall environment has been planned, not cobbled together
- "Think of it as an environment for implementing statistical techniques"

Getting started

- Once installed, start with R on unix, or click icon for windows
- In unix you'll go straight to the "console"
- In windows you'll see Rgui and drop into the console
  - Pretty much like an interactive calculator
- Try `1+1`
- Up arrow gives last instruction like unix terminal
- To exit type `q()`
- To see last value type `.Last.value`
- Note!! R is case-sensitive!
- Can stop evaluation with escape key

Getting help

- Of course there is always google… but!
- `help.start()` will start your browser and take you to R help page
- `help(`*`function`*`)` will take you to the on-line manual page
  - Try `help(mean)` or `?mean` (short hand)
- You can search for functions containing a keyword
  - Try `apropos("mean")`
- Really helpful: you can also get examples of usage
  - Try `example(mean)`
- If you learn by example "`example`" is ++helpful

Basic steps: R as a calculator

- The console behaves just like a calculator
  - R stores variables in double precision!
  - Try `cos(1)`
- For powers use ^ symbol, usual ops `+`, `-`, `/`, `*` etc
  - `pi` is a defined constant
- Modulo arithmetic can be done with `%%`
  - Try `7%%5`
- Use `log10` for base 10 logs, log2, log also available
- arcsine is abbreviated `asin`
- When in doubt, try `apropos("name")` to find the function name

"Entering" variables

- Technically, forming objects rather than variables
  - I may be looser in terminology
- R uses vectors/arrays extensively so try (c=combine)
  - `x = c(1,2,3,4,5)`
  - `x = c(1:4,5)` or `x = seq(1,5,by=1)`
  - `x = ("hello","world")`
  - `x` alone prints value of object, `x[2]` the second element
- `typeof(`*`varname`*`)` will tell you the kind of data
- `rm(`*`varname`*`)` will remove variable, `ls()` lists defined

Why two options?

- You can give variables values using either = or <-
- This is a hangover from the A Programming Language (APL)
- Keyboards for it originally had a <- key
- Different from the operation "is a=b?"
  - Note == is a logical question giving true/false answer in R
  - Try `1==2`
- R also requires = to be used to set values in functions
- "Traditionalists" prefer <- be used for assignment
- There are some drawbacks to either method

End Intro

- Please download and set-up the environment for Thursday
  - Will help to follow along

Simple yet useful functions

- `x = c(3,2,1:5)`
- `sort(x)` or decreasing `sort(x,decreasing=TRUE)`
- Lots of simple functions, try

```
mean(x),median(x)
sd(x),var(x),max(x)
min(x),sum(x)
range(x),length(x)
```

- Also try outputting into a variable e.g.

```
z=var(x)
z
```

Data types in R

- **Numbers** are stored in double precision
  - Matrices also supported. Address using `a[2,3] first number=row, second number=col`
    - Enter values using rows first in combine function
  - Multiple arrays can be linked in to a "data frame"
- **Strings** can be stored as well (we already gave an example)
- **Logical** datatypes are also allowed (TRUE/FALSE)
- Also, **dates**, **missing data** and **factor** types
- # symbol allows you to add comments
- Note need one on each line – no blocks of comments

Logical operations

Operation	Function
`x > y`	Greater than (num+logic)
`x < y`	Less than (num+logic)
`x >= y`	Greater than or equal
`x <= y`	Less than or equal
`x == y`	Equal to
`!x`	NOT x (x logical)
`x != y`	Not equal to
`x & y`	x_i AND y_i elements
`x | y`	x_i OR y_i elements

There are more, including single output AND, OR – check manual

Manipulating 1-d arrays/vectors

- Already mentioned can use `x[2]` for third element
- To delete the third element use `x[-2]`
- `max(x)` will give max value, but `which.max(x)` specifies the index
- Can also use which(x<3) to find matching indices
- Match will find positions of elements e.g.

```
x=c(2,3,3,4,5)
match(3,x)
```

- V. important – math ops work on individual elements!
  - Try a simple operation: `b=2*x+3`
  - Same is true for `b*x`

Lists

- Lists go beyond arrays by allow elements to have arbitrary data types e.g.

```
f = list(c(1,2,3),"whoop", FALSE)
f
[[1]]
[1] 1 2 3
[[2]]
[1] "whoop"
[[3]]
[1] FALSE
```

Scripting

- Assuming you are on windows/mac File>New Script
- On unix you can use any text editor
  - Advice: use one with R syntax extensions e.g. Rgedit
  - Rstudio may well be best
- In editor window type a quick script e.g.

```
x = c(2,3,4,5)
mean(x)
```

- Highlight with cursor and hit Ctrl+R
- Avoid using c,q,s,t,C,D,F,I,T for variables – not reserved, but you can break expected meanings

Script Syntax

- No need for semicolons at end of each line
- But two statements on one line must be separated by a semicolon
- You can overflow onto another line but must ensure it is not a complete expression e.g. compare

```
y = c(2,3,4)
+ 2
y = c(2,3,4) +
2
```

- Both will execute – but only one gives expected behaviour!!!! Which one ☺? And why?

Summary

- R provides and entire ecosystem that has been carefully planned
- Addressing of arrays similar to FORTRAN
- It is not, in any sense, new
- It's primary strength is much like python's – huge amounts of user contributed packages and online help via mailing lists