Technical Computing for the Earth Sciences, Lecture 2:

Intro to software engineering

(incl: your first Julia program, vcs/git, commenting)

EARS 80.03

Software engineering?

How to design and build good software

- Tools, like:
    - Version control (git/mercurial/etc.)
    - Debuggers
    - Profilers
    - Continuous integrationn
- Techniques, like
    - Commenting your code
    - Writing good (unit &/or integration) tests for your code
    - etc.

First, we need a program though

“Hello world”

```
julia> println("Hello, world!")
```

Other ways we could write this?

“Hello world”

```
julia> println("Hello, world!")
```

Other ways we could write this?

```
julia> print("Hello, world!\n") # '\n' = newline
```

also: string interpolation

```
julia> person = "world"
julia> println("Hello, $(person)!") # this is string interpolation
```

Commenting

Comment character: `#`

- Anything after the '#' doesn't evaluate.
- Use comments to explain what your program does
- Not too many though, or it gets redundant

Multi-line comments?

```
# You could comment
# multiple lines manually

#=
or you can make a special block like this
that comments-out anything between the bounds
=#
```

Why Version Control?

Have you ever . . .

1. Tried to add a feature to a program and broken it so badly you wished you could abandon all your changes?
2. Left a program in an unrunnable, or uncompilable state. . . and your advisor needed new results **now**?
3. Tried to work on a program or write a document with one or more other people?

There are solutions. You could:

1. Backup frequently, keeping all backups.
2. Only ever develop on a copy of your code
3. Appoint someone whose job it is to merge all contributions

Version Control Systems (VCSs) offer a far better solution to all these problems. *And* offer many other advantages too.

Version control: git

```
shell$ git --help
```

First time you run git:

```
shell$ git config --global user.name = "Your Name"
shell$ git config --global user.email = you@dartmouth.edu
```

Some common commands

```
git init                                  create a new repository
git clone url                copy an existing repository from a git server

git add filename                              prepare to save changes
git commit -m "some notes"    save (commit) those changes locally
git push                        send your changes to a git server
git pull                            get changes from a git server
git rm filename                     delete & stop tracking a file

git status                                              what's up?
git log                                      list of recent commits
git diff                                  difference between commits
```

First program continued

Write a Julia function to calculate the first N prime numbers

```
function findprimes(N)
    primelist = Array{Int64}(undef, N)

    # Do something

    return primelist
end
```

See how quickly you can calculate 10^4 primes. 10^5? 10^6?

```
N = 10^4
@time primes = findprimes(N)
```