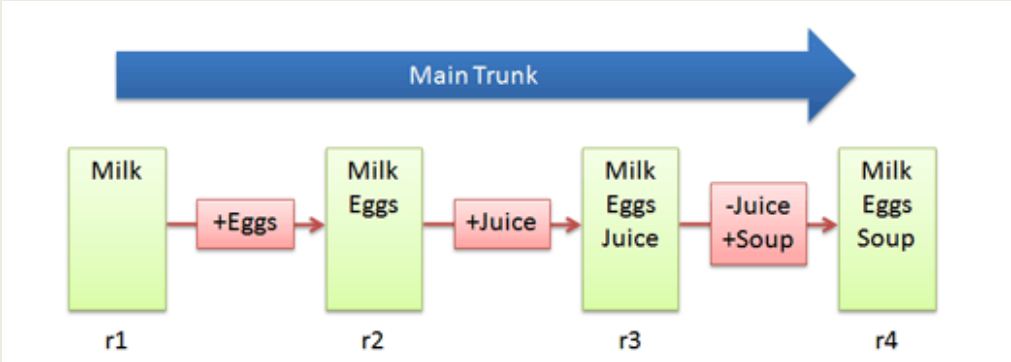
Version control and documentation

Lecture 8, CPSC 499 Fall 2018

What is version control?

- A computer program that tracks changes you have made to a set of files
- Lets you easily revert to old versions
- Lets you document why you made the changes
- Allows multiple users working on the same project



Git

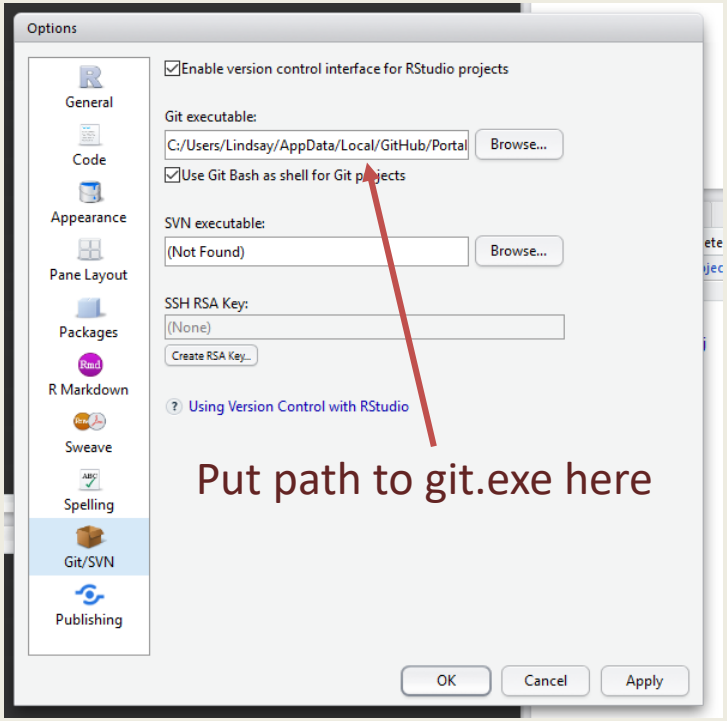
- Popular software for version control
- Integrates with RStudio to track changes to R scripts and source code on your computer
- Allows branching so you can try new things out before permanently incorporating changes into software

Useful info for integrating Git with RStudio

In RStudio Tools -> Global Options

At git-scm.com you can get a "portable" version of git.exe that doesn't require admin access. Use this to put Git on the lab computers.

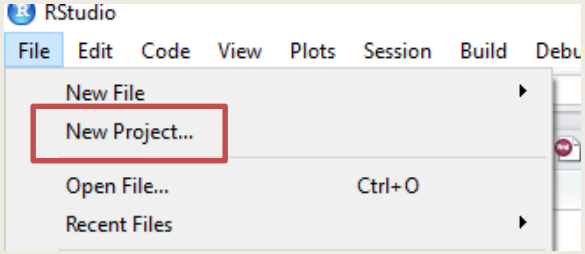
For your own computer, I recommend installing the regular version of Git.

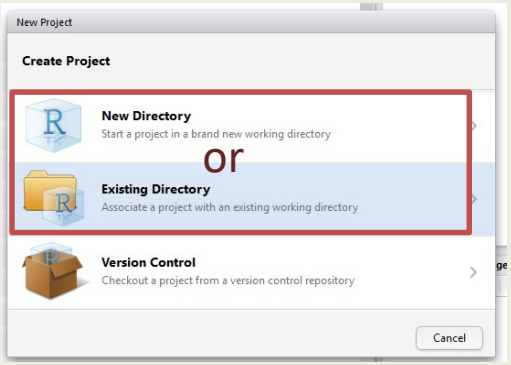


Restart RStudio after adding git.exe path. Edit .gitconfig in your user directory with your name and email.

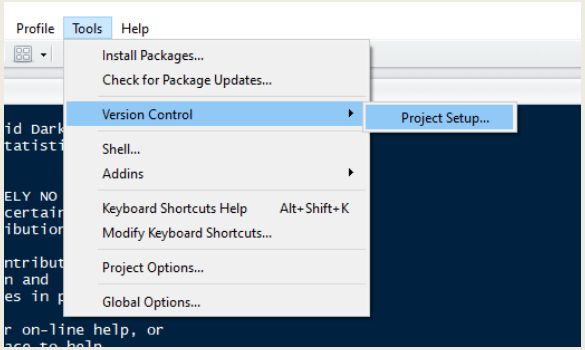
Setting up version control for an R project stored locally on your computer

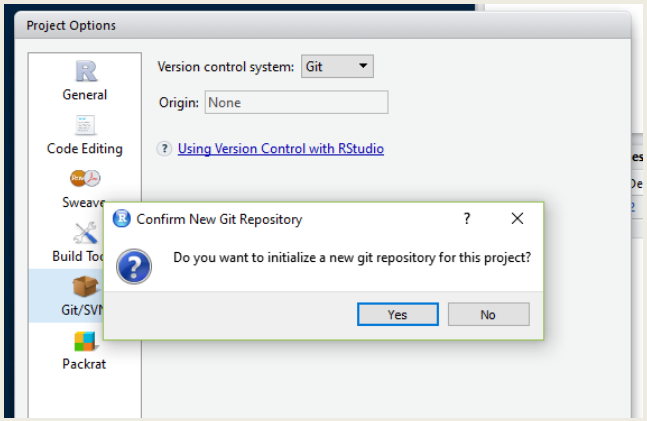
- For analysis etc. that you don't want to share with the world (yet)





Remaining dialogs to set up the project



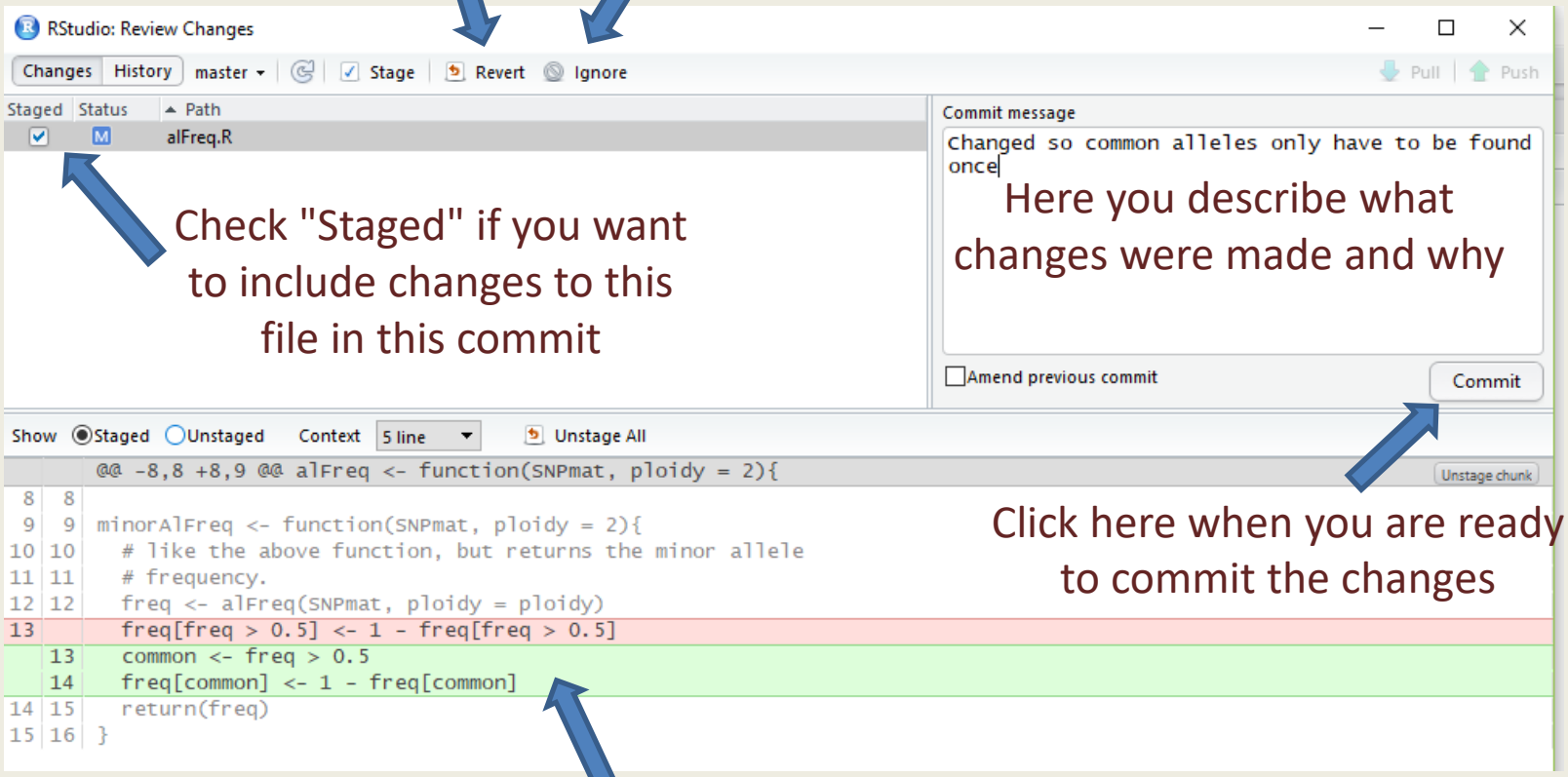


Reviewing changes and making commits

This button

To "undo" changes

To stop tracking changes to a file



Check "Staged" if you want to include changes to this file in this commit

Here you describe what changes were made and why

Click here when you are ready to commit the changes

Shows what changes have been made to each file

"History" to browse through all commits

Changes | History | master | (all commits) | Search | Pull

Subject	Author	Date	SHA
HEAD -> refs/heads/master Changed so common alleles only have to be found once	Lindsay Clark <dragonzuela04@gmail.com>	2017-11-11	fc1d6bf0
Add minorAlFreq function	Lindsay Clark <dragonzuela04@gmail.com>	2017-11-11	2b1d3cf3
First commit	Lindsay Clark <dragonzuela04@gmail.com>	2017-11-11	9fb4c238

Commits 1-3 of 3

SHA 2b1d3cf3
Author Lindsay Clark <dragonzuela04@gmail.com>
Date 2017-11-11 18:30
Subject **Add minorAlFreq function**
Parent 9fb4c238

alFreq.R

alFreq.R View file @ 2b1d3cf3

```
@@ -4,4 +4,12 @@ alFreq <- function(SNPmat, ploidy = 2){
4  4   # vector of allele frequencies in the population.
5  5   freq <- colMeans(SNPmat, na.rm = TRUE)/ploidy
6  6   return(freq)
7     }
       No newline at end of file
   7 }
   8
   9 minorAlFreq <- function(SNPmat, ploidy = 2){
  10   # like the above function, but returns the minor allele
  11   # frequency.
  12   freq <- alFreq(SNPmat, ploidy = ploidy)
  13   freq[freq > 0.5] <- 1 - freq[freq > 0.5]
  14   return(freq)
  15 }
```

Mini exercise

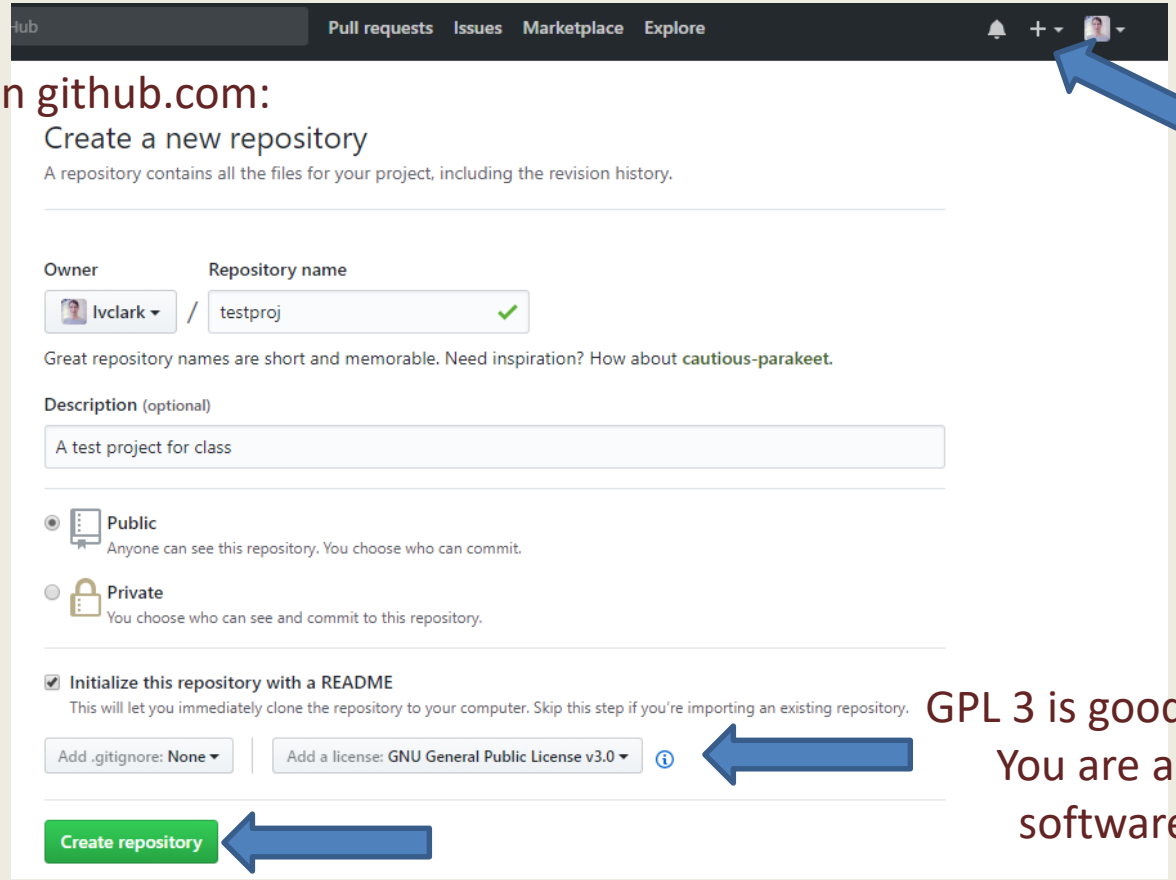
- Open a project from a previous lab or lecture
- Add version control using Git
- Make a couple commits and then view them in history

GitHub

- Free website where you can host your Git repositories
- Good for sharing code, collaborating
- Social networking features, wikis, web hosting
- Can give other people feedback on their software
- Integrates with RStudio

Setting up a GitHub repository

If you know you will want to host a repository on GitHub, it is easiest to set it up on GitHub then import it to RStudio, rather than the other way around. (see happygitwithr.com)

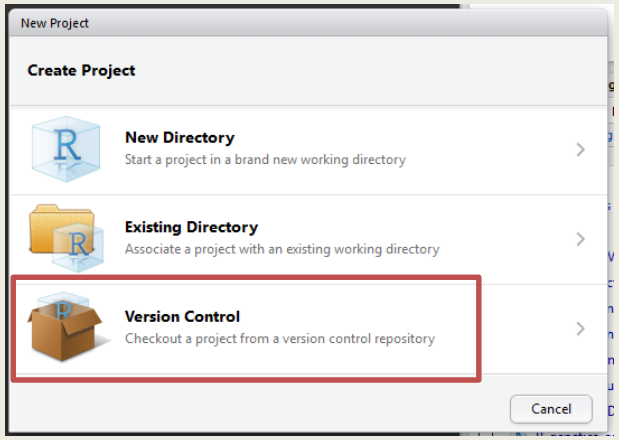


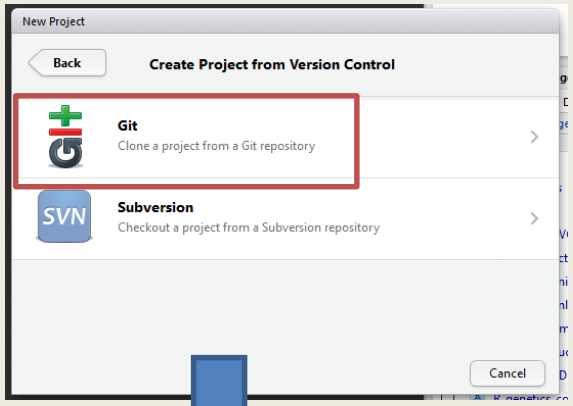
On github.com:

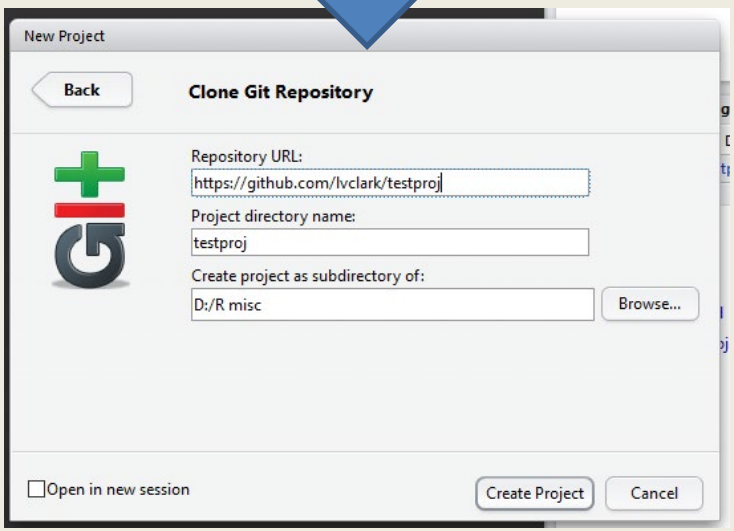
+ button to bring up this dialog

GPL 3 is good for scientific software. You are also not liable if your software causes any harm.

Bringing your GitHub repository into RStudio





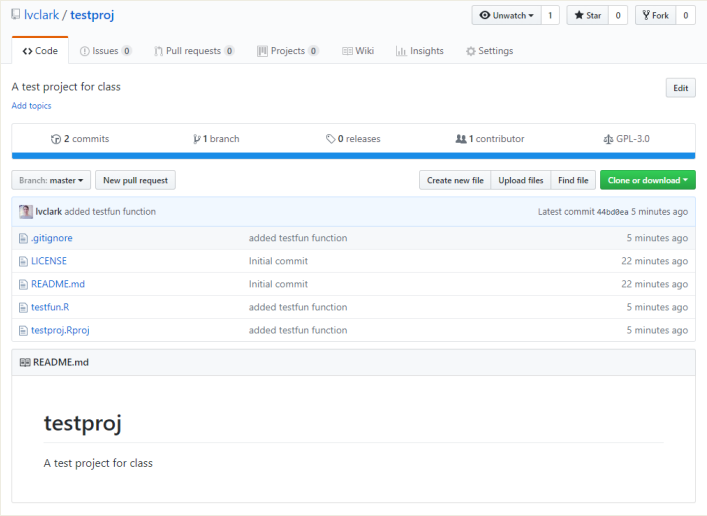


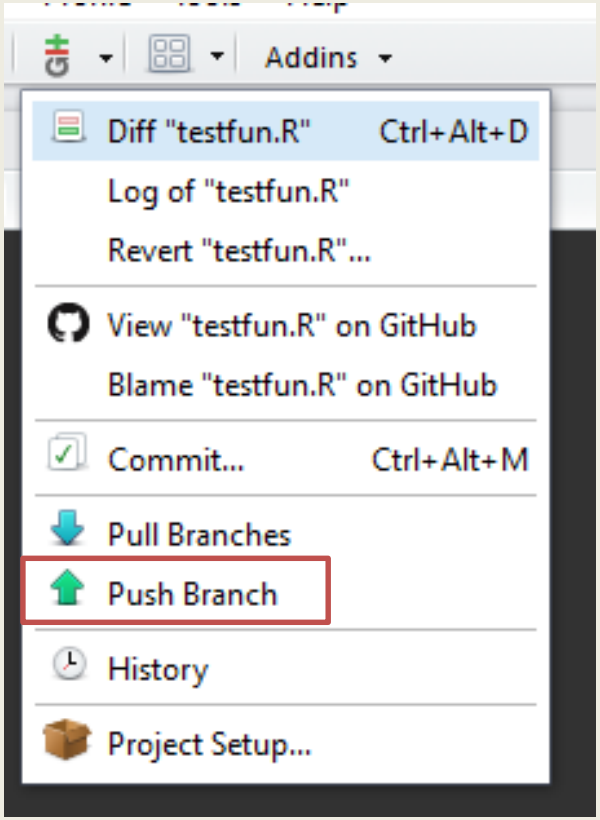
Now you can add and edit files and directories and make commits as demonstrated earlier

To sync your changes back to GitHub

- Make a commit in RStudio as demonstrated earlier
- "Push" the changes

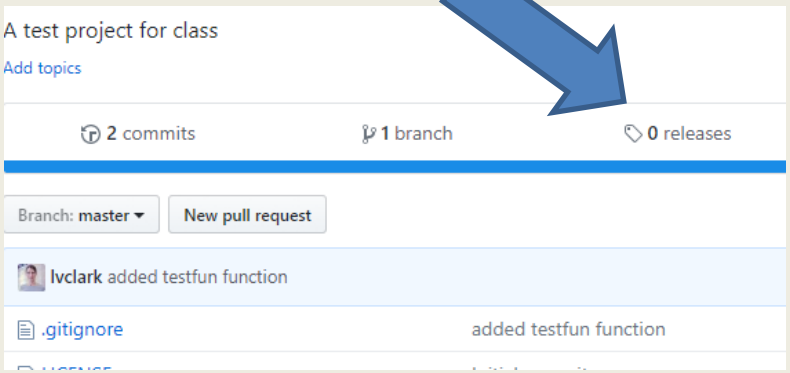
Now you should see your changes on the website

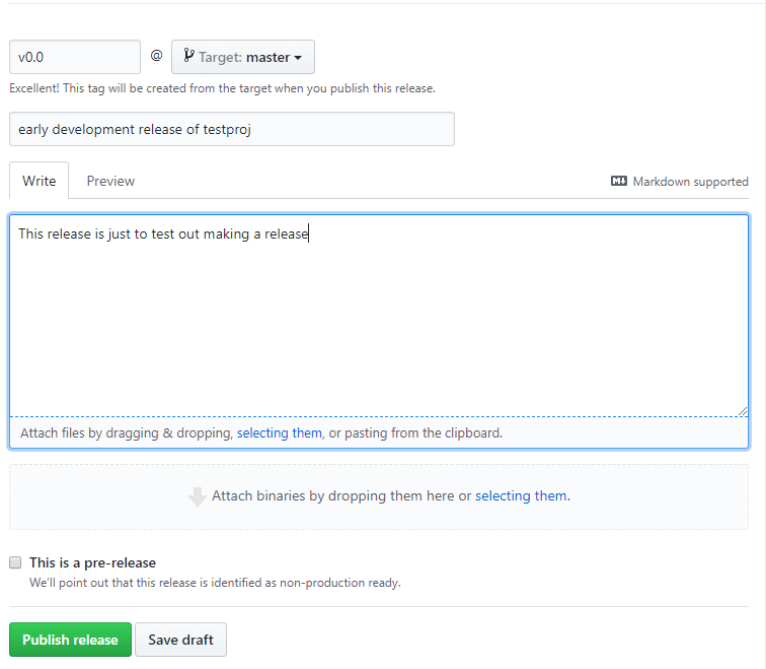




Making a release

- A snapshot of the repository at a given point in time
- Generally a fully-working version of the software
- If the current version isn't working, users can always download last release instead
- To turn in final project, make a release and send me the link





Semantic versioning

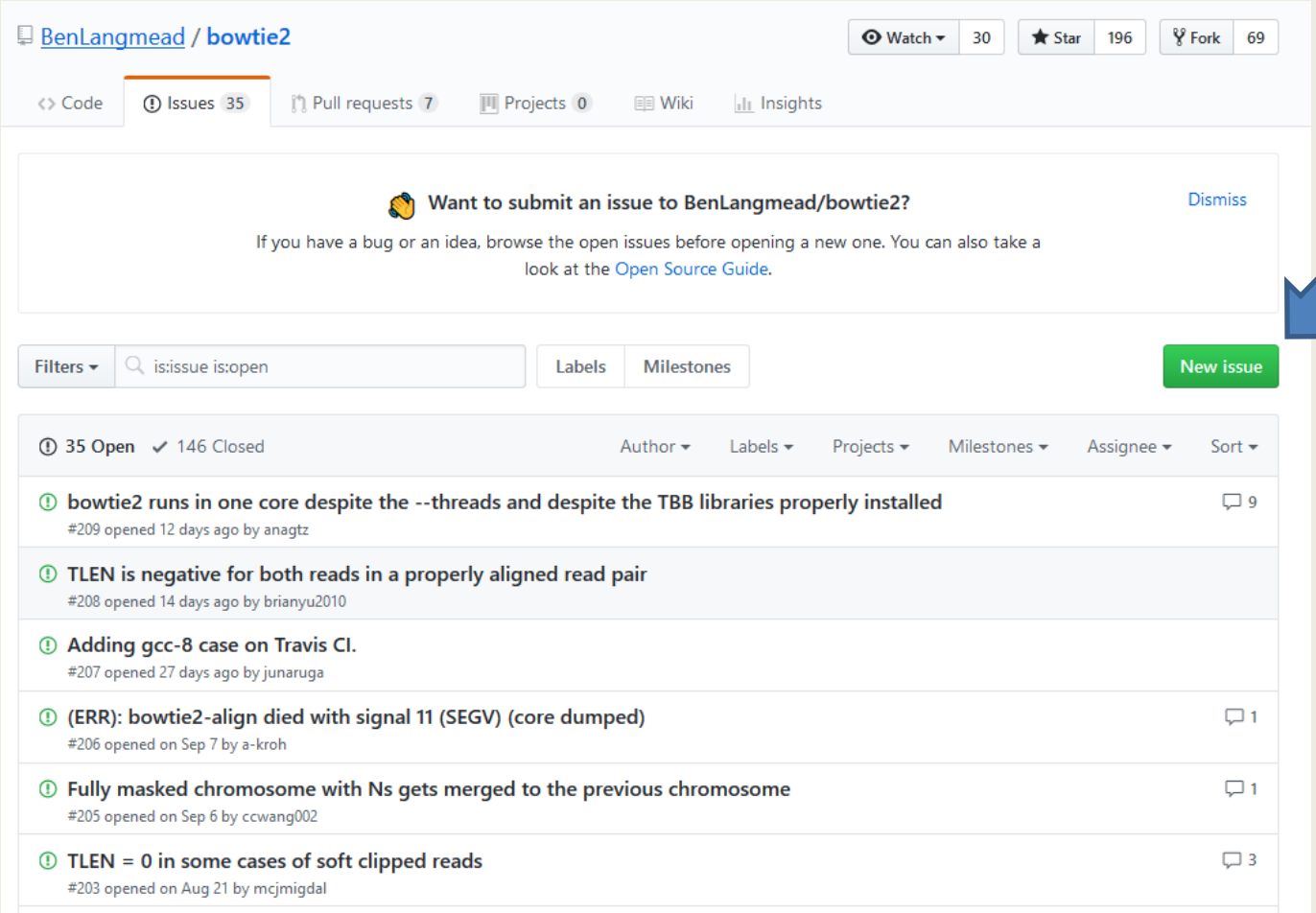
- E.g. 2.14.1, 1.0, 0.2-1
- First number is MAJOR version
  - 0 when software still in development
  - 1 for first stable release to public
  - Increase further when there are backwards-incompatible changes
- Second number is MINOR version
  - Starts at 0, increase when new functionality added
- Third number is PATCH version
  - Starts at 0, increase for bug fixes or code optimization

Getting a DOI for a release

- See https://guides.github.com/activities/citable-code/
- Link to Zenodo account
- Then every time you make a release of your repository in GitHub, it gets permanently archived on Zenodo (hosted at CERN)

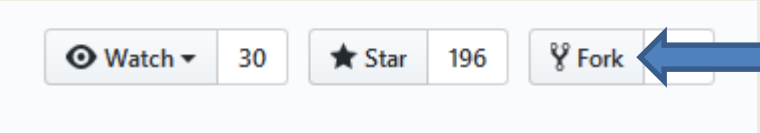
Using GitHub for collaboration

Reporting bugs and problems

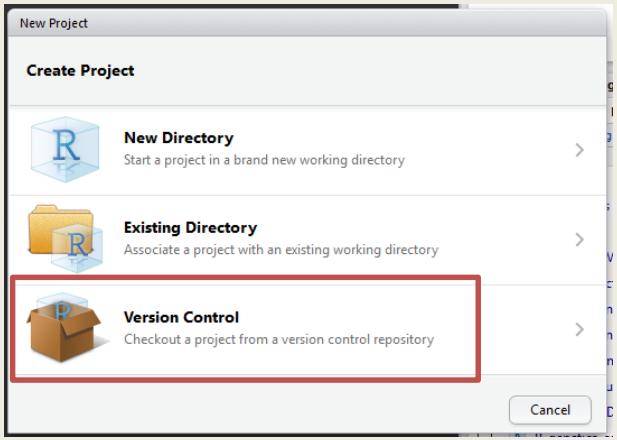


Better option than emailing software creator. Can see if your issue was already reported.

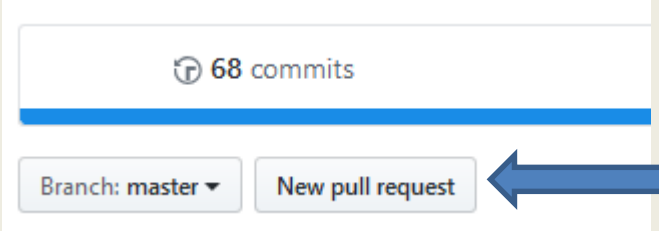
Suggesting changes to someone else's software



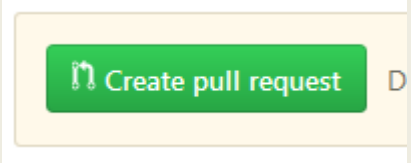
First click "**fork**" to make your own copy



Then **clone** your fork to your computer using RStudio (or Git clone). Make any changes you want, make **commits**, and **push** back to GitHub.

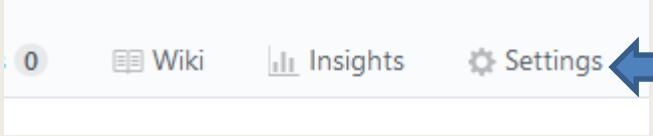


Go back to your fork and start a **pull request**

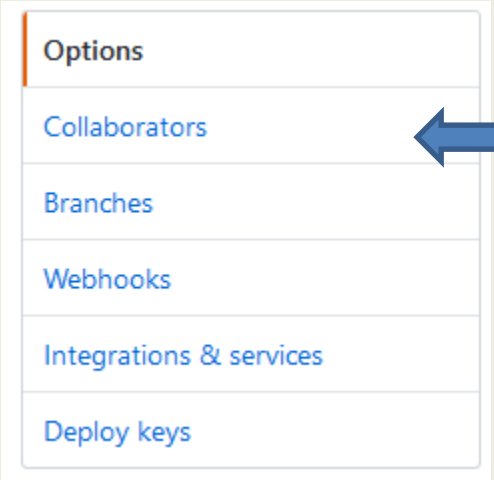


Check that you are using the right fork, and create the pull request. Include a description.

Collaborating on a repository

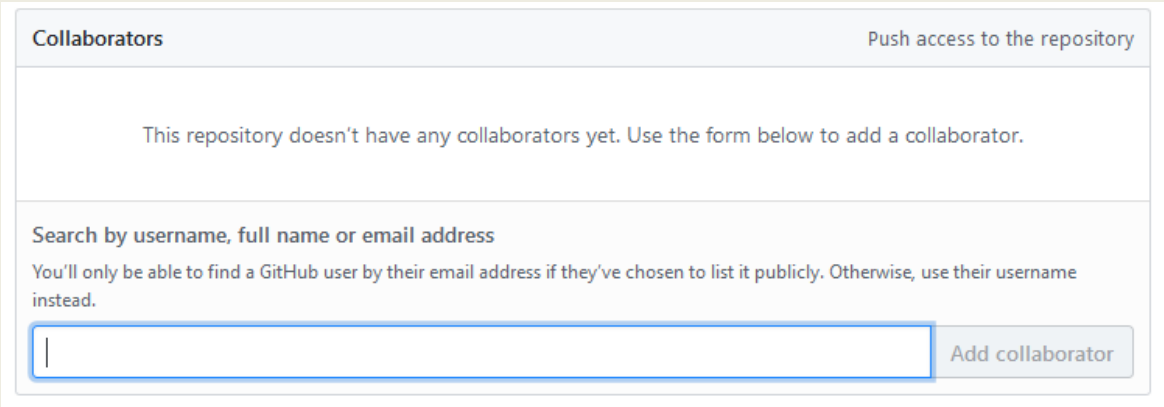


Go into Settings



Go to Collaborators

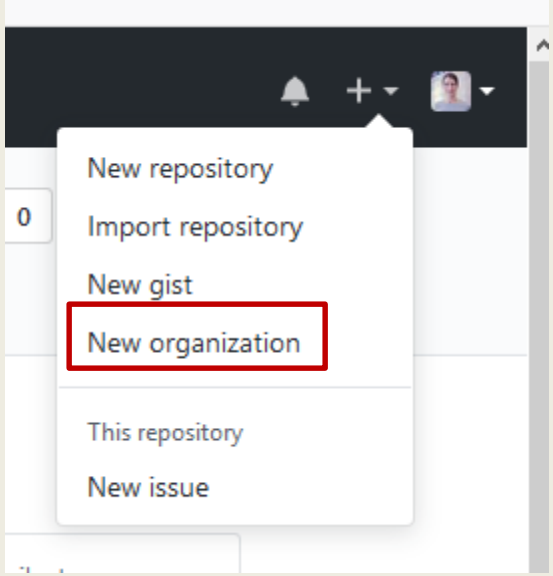
I recommend doing this for group work on the final projects.



Add usernames.
These people now can **push** to the repository the same as you can.
Remember to **pull** before you edit to avoid conflicts!

GitHub Organizations

- For if you have multiple repositories that you want to work on with the same group of people
- Repositories will belong to the organization rather than any one individual
- Can organize people into teams with different privileges



Might make an organization for your lab, etc.

Tools for incorporating R code and output into documents

R Markdown format

- Available with "knitr" package
- Based on Markdown
- Quick and easy to write
- Renders nicely to HTML
- .Rmd extension

Graphics in R

One of the major draws of the R programming language is its graphical capabilities. As you prepare manuscripts for publication, you may often find that R is the best choice of (free) software for producing publication-quality figures, even if you did the analysis using some other software.

In this lab we'll use graphical functions available with the standard installation of R, which will give you some opportunities to practice indexing and other skills that we learned earlier in class. There is also a popular package called ggplot2 for making plots in R, which I recommend looking into on your own time.

A basic scatter plot

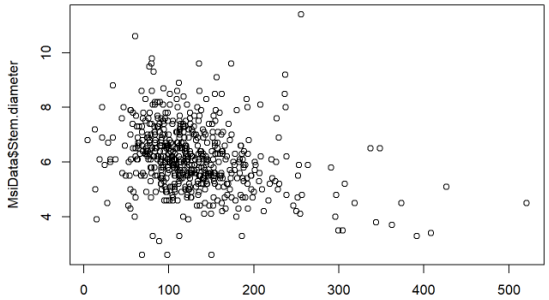
We'll read in some data to practice on. Each row is a genotype (AKA an individual or clone) of the grass *Miscanthus sinensis*, and the columns have some data about those genotypes, including genetic group, latitude and longitude of collection for wild plants, and the least-squared means of some phenotypes measured in a field trial.

```
MsiData <- read.csv("Miscanthus_sinensis_yield.csv", header = TRUE,
                    stringsAsFactors = FALSE)
str(MsiData)
```

```
## 'data.frame':    580 obs. of  7 variables:
##  $ Genetic.group  : chr  "S Japan" "SE China/tropical" "US naturalized" "S Japan" ...
##  $ Latitude       : num  NA 23.7 38.9 NA 44.2 ...
##  $ Longitude      : num  NA 121 -77 NA 142 ...
##  $ Biomass.yield  : num  2436.3 81.9 613.5 2388.9 530.3 ...
##  $ Plant.height   : num  292 165 163 272 184 ...
##  $ Number.of.stems: num  144.4 24.7 97.2 143.4 94.2 ...
##  $ Stem.diameter  : num  8.3 5.9 4.6 7 5.3 6.1 4.8 5.6 6.8 6 ...
```

Let's explore some of this data with a scatter plot. If we call the `plot` function with integer or numeric vectors for both the first and second arguments, we get a scatter plot, with the first argument on the x-axis.

```
plot(MsiData$Number.of.stems, MsiData$Stem.diameter)
```



Sweave format

- Available with base R installation
- Based on LaTeX
- Makes very nicely typeset PDFs
- Renders mathematical equations well
- Lots of LaTeX utilities for citations, tables, figures, links within document, etc.
- .Rnw extension

4 Getting Started: A Tutorial

4.1 Creating a dataset

As with any genetic software, the first thing you want to do is import your data. For this tutorial, go into the "extdata" directory of the polysat package installation, and find a file called "GeneMapperExample.txt". Open this file in a text editor and inspect its contents. This file contains simulated genotypes of 300 diploid and tetraploid individuals at three loci. Move this text file into the R working directory. The working directory can be changed with the `setwd` function, or identified with the `getwd` function:

```
> getwd()

[1] "D:/GitHub/polysat/vignettes"
```

Then read the file using the `read.GeneMapper` function, and assign the dataset a name of your choice (`simgen` in this example) by typing:

```
> simgen <- read.GeneMapper("GeneMapperExample.txt")
```

The dataset now exists as an object in R. The following commands display, respectively, some basic information about the dataset, the sample and locus names, a subset of the genotypes, and a list of which genotypes are missing.

```
> summary(simgen)

Dataset with allele copy number ambiguity.
Insert dataset description here.
Number of missing genotypes: 5
300 samples, 3 loci.
1 populations.
Ploidies: NA
Length(s) of microsatellite repeats: NA

> Samples(simgen)
```

Uses for R markdown and Sweave

- Software tutorials
- Reproducible analysis; can write entire paragraphs about what you are doing
- Dynamically-generated reports; say if you want to make some figures and an explanation of what they mean, and also store the code with it

Documentation used in R packages

All user-level functions in an R package must have an Rd file

- These are used to generate the help pages
- Use the `prompt` function to create a skeleton Rd file for a function in your global environment
- Open the file in RStudio and fill in the sections
- Hit "Preview" to see what it will look like in RStudio after the package is installed

Anatomy of an Rd file

```
\name{testfun}
\alias{testfun}
\title{
Add One to a Number
}
\description{
This function takes a number and adds one.
}
\usage{
testfun(x)
}
% this is a comment
\arguments{
  \item{x}{
A number.
}
}
\details{
This functionis pretty self-explanatory and doesn't need
more detail, but if it did this is where it would go.
}
\value{
The value of \code{x}, plus one.
}
\references{
Podunk, B. B. and Podunk, B. J. (2017) Adding one to stuff.
\emph{Nature} \bold{551}, 13507--13508.
}
\author{
Lindsay Clark
}

\seealso{
\code{\link{Arithmetic}}
}
\examples{
testfun(5)
}
\keyword{ arith }% use one of  RShowDoc("KEYWORDS")
```

- `\xxxx{yyyy}` or `\xxxx{yyyy}{zzzz}` to delineate sections and format text
- `%` to indicate comments
- Examples should be executable without needing anything else in global environment.
- Can put multiple functions into one Rd file: use multiple `\alias` sections
- See "Writing R documentation files" in the "Writing R Extensions" manual

Package vignettes

- For when the package needs a tutorial above and beyond the info in the Rd files
- A package can have any number of vignettes (including zero)
- All code in the vignette must be executable (no errors) for the package to pass its check
- R can extract all the code from a vignette into a script
- Can use R markdown or Sweave

Midterm extra credit

- Contribute to study guide: https://github.com/lvclark/CPSC499-Fall2018-studyguide
- Give me your GitHub username on Thursday (or sooner) and will add you
- One point per commit, up to five points
- (Edit the study guide before the exam, but not during!!)