Gold
Microsoft Partner

Microsoft

# Git 101

Introduction to Version Control &
Collaboration with Git & GitHub

SEAMS
AN ARCADIS COMPANY

# This Session

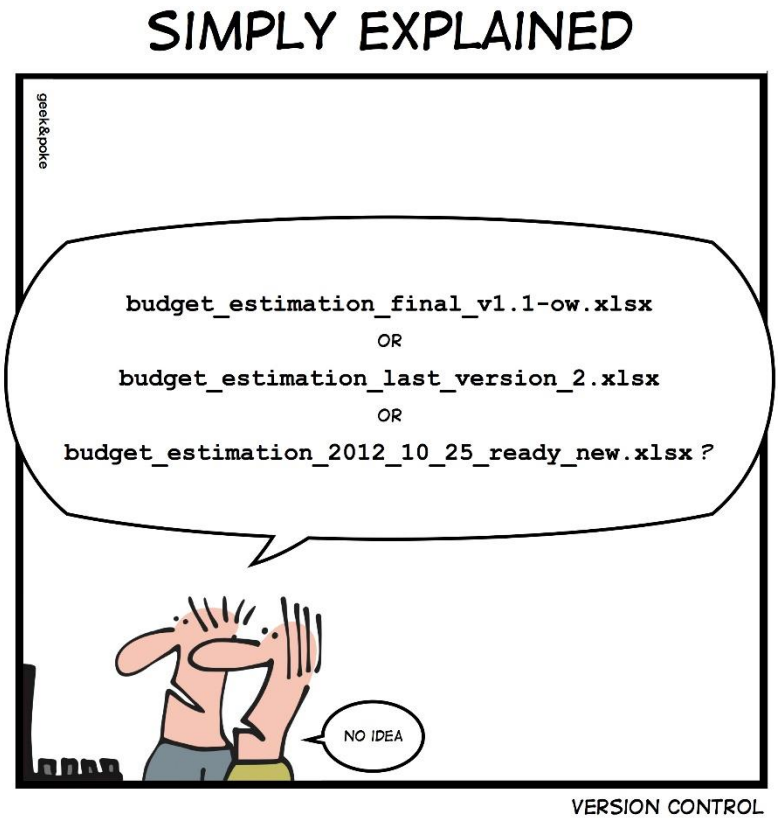
SEAMS
AN ARCADIS COMPANY

- Introduction to Version Control Systems
- Introduction to Git and GitHub
- A step-by-step example of collaborating on a data science project hosted on GitHub
  - Basic Git shell commands for managing your project
  - Pull requests for collaboration

# Version Control

SEAMS
AN ARCADIS COMPANY

- Keep track of changes in your project
- No more mile-long filenames
- Makes collaboration easier and **safer**
- Used as standard in software industry
- Git is not the only one
  - There's Mercurial, SVN and more



# Git Version Control

SEAMS
AN ARCADIS COMPANY

- Git exists as a hidden folder you put in your project that tracks all the changes and manages versions

- A folder that is set up for Git is known as a **repository**, usually called a **repo**

- Git != GitHub
  - GitHub is a place for hosting repos (as **remotes**) so that you can share them with your collaborators
    - There are alternatives, like BitBucket, Beanstalk, etc.
  - Think of GitHub as cloud storage
  - At SEAMS, we probably won't use GitHub as the free version is not private
    - IT will set something up for us
    - You don't need to use remotes to still get a lot of benefit from Git

# To Git or not to Git?

SEAMS
AN ARCADIS COMPANY

- Git is for controlling versions of things you and your team will change as you work
- Don't use it for data!
  - Data should be immutable
- Git can track changes in any file, but it makes most sense with code and text documents
- You can tell Git what you want it to track and what you would like it to ignore, using a text file called a **.gitignore**

# Install Git on Windows

SEAMS
AN ARCADIS COMPANY

- Go to: https://git-scm.com/
- Hit *Download X.XX.X for Windows*
- Open the executable, follow the installer
  - On *Select Components*, make sure *Git Bash Here* and *Git GUI Here* are selected
- Allow Git to be run by 3rd-party software
  - This allows integration with, e.g., Rstudio
- On *Configuring the line ending conversions,* select *Checkout Windows-style,…*

# Questions

SEAMS
AN ARCADIS COMPANY

- We have Git installed now, any questions before we start using it?

# Clone a Public Repo

SEAMS
AN ARCADIS COMPANY

- Go to: http://www.github.com/tomdakin/git_101
  - Click *Fork* to make a copy on your personal GitHub account.
  - Click *Clone or Download* and copy the link. This is the URL of your personal **remote**

- Find a handy place to work on your hard drive in windows explorer and right-click there
  - Select *Git Bash Here*: A handy Git shell we'll use for our git commands

- Do: `git clone shift+ins`
  - Now you have a copy of the repo on your hard drive. It still has a connection to the remote, but it's your copy and you can do what you like with it without affecting the remote

- Head back to the explorer window and have a look at the folder you just cloned.
  - Tick *View->Hidden items* : This is where the .git folder is lurking

# Change the Code and Commit

SEAMS
AN ARCADIS COMPANY

- Add the movies.csv data file to the data folder
- Open the .Rproj file and run the src/movies.R script.
  - The script loads the data, runs a little analysis and then prints some stuff
- Add your own print statement to show your own favourite film (or anything you like)
- In the Git shell, run
  - `git status`
  - `git add src/movies.R` – This adds the change to a **staging area**. It just tells Git that you want to include the updates to this file in the next **commit**.
  - `git commit –m "adding favourite movie"`
  - `git status`
- Now your local copy has your changes but also has a complete history of what the code used to look like. We can **reset** or **revert** to this if we want to use an older version of the project
- Do `git push` to publish your changes to your personal remote

# Collaboration

SEAMS
AN ARCADIS COMPANY

- Let's send your changes back to the project lead for inclusion in the production version
  - This can be done using a **pull request**

- Head back to github
  - Find your remote: www.github.com/<username>/git_101
  - Click *Pull Requests -> New Pull Request -> Create Pull Request -> Create Pull Request*

- The project lead then can merge all the changes into the project
  - It's their job to resolve any conflicts in the code
  - The final version will have everyone's changes combined
  - Afterwards, you can refresh your local repo with
    - `git remote add upstream https://www.github.com/tomdakin/git_101.git`
    - `git fetch upstream`
    - `git pull upstream master`

# Summary

SEAMS
AN ARCADIS COMPANY

- Version Control – To manage versions and aid collaboration
- Git & GitHub – Tools for VC (Git) and collaboration (GitHub)
- `git fork` & `git clone` – Copying repos server-side (fork) and locally (clone)
- `git add` & `git commit` – Tell Git which updates to look at (add) and save them when you're done (commit)
- `git push` & `git request-pull` – Save your changes to your own server (push) and request that they be added to the production project (request-pull)

# Questions

SEAMS
AN ARCADIS COMPANY

- Git can be confusing at first, but it becomes much easier with use
  - Data scientists and software developers around the world use it every day so there is lots of support on Google/Stack Overflow/GitHub Help
  - Any Git problem you may have will not be new!

- Ask questions!