

# Version control

(using git)

---

**Danny Awty-Carroll**

May 15, 2018

This will focus on using git in windows with a UI

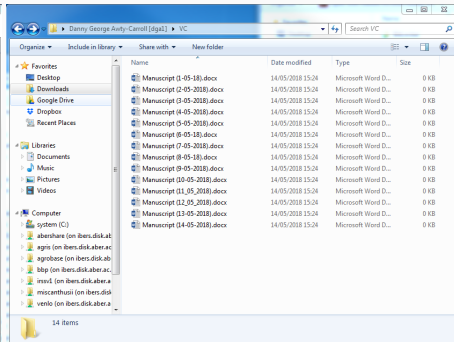
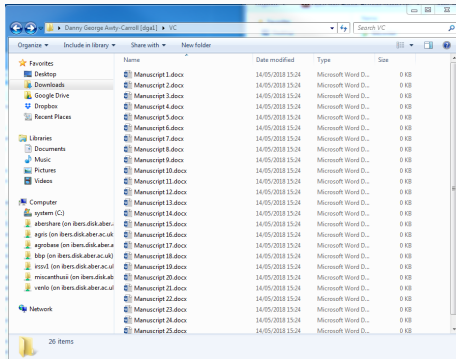
1. Why version control?
2. Version control system
3. Platform
4. UI
5. Conclusions

# Why version control?

---

# What is version control

This is just the management of versions of a document.  
One document through time.



All of us use some version control

# Where things get complicated

Numbering or dating documents works OK but can fall down when

- There are multiple documents that need to work together (i.e. a script and data)
- There are multiple people working on the documents (are they on the latest version and merging changes)
- There are updates to the project that may break things
- Line changes need to be reviewed

Some of this is solved in programs like word with track changes (so you see who altered what when)

# What version control systems can do

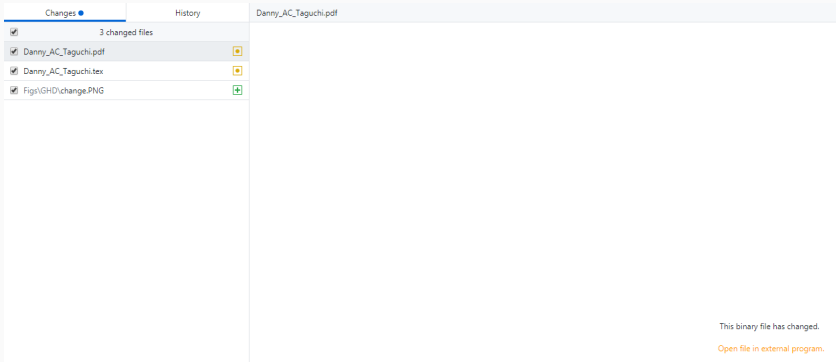
- Give line by line user by user line changes
- Make a stucher to vershons so there can be side branches
- Minimise problems of multiple users working on the same document at the same time

Changes ●	History	Danny_AC_Taguchi.tex	
☑ 2 changed files		72	72
☑ Danny_AC_Taguchi.pdf	📄	73	73
☑ Danny_AC_Taguchi.tex	📄	74	74
		75	-
		76	-
		77	-
		78	-
		75	+
		76	+
		77	+
		78	+
		79	78
		80	79
		81	80
		82	-
		81	+
		82	+
		83	83
		84	84
		85	85

```
@@ -72,14 +72,14 @@ Some of this is solved in programs like word with track changes (so you see who
\begin{frame}[fragile]{what version control systems can do}
\begin{itemize}
\item There are multiple documents that need to work together (i.e. a script and data)
\item There are multiple people working on the documents (are they on the latest version and merging changes)
\item There are updates to the project that may break things
\item Line changes need to be reviewed
\item Give line by line user by user line changes
\item Make a stucher to vershons so there can be side branches
\item Minimise problems of multiple users working on the same document at the same time
\end{itemize}
\end{frame}
\end{frame}
\begin{frame}[fragile]{what version control systems can do}
\begin{itemize}
\item There are multiple documents that need to work together (i.e. a script and data)
\item There are multiple people working on the documents (are they on the latest version and merging changes)
```

# What version control systems can't do

- Add much extra control to binary files (not plan text)
- Back up in real time



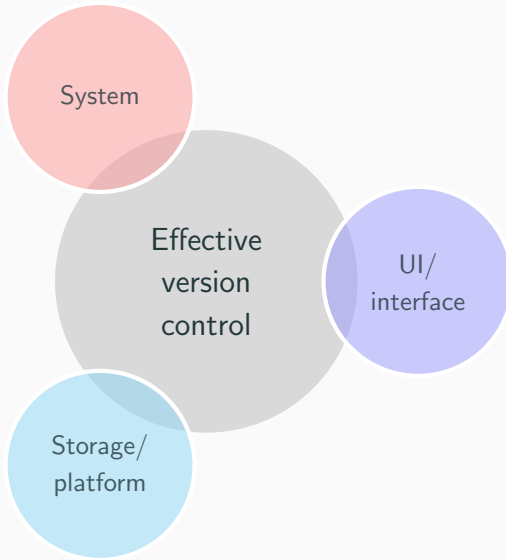
# Version control system

---



# What version control system?

The first thing about version control systems is which system (we will cover git)



# What version control system?

- There are two popular version control systems (git SVN)
- We will cover git as it is the most popular, the one I know and easiest to use
- Git was developed in 2005 by Linus Torvalds

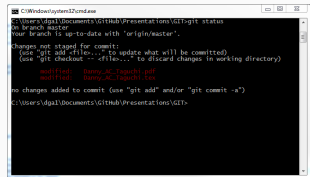


(Principal developer of Linux)

# What is git?



- Git is just system of version control

A screenshot of a Windows Command Prompt window titled "GitWindows\system32\cmd.exe". The window shows the output of the "git status" command. The text is as follows:

```
c:\Users\dgal\Documents\Github\Presentations\GIT>git status
On branch master
Your branch is up-to-date with 'origin/master'.

changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Dumpy-X-12epoch.pdf
        modified:   Dumpy-X-12epoch.tex

no changes added to commit (use "git add" and/or "git commit -a")
c:\Users\dgal\Documents\Github\Presentations\GIT>
```

- By default it is used through a terminal
- Git has projects could reposetrys (or repos)
- It then has a tree struchure to manige the versions

# Tree structure?



[master] 6c6faa5 My first commit - John Doe

[develop] 3e89ec8 Develop a feature - part 1 - John Doe

[develop] e188fa9 Develop a feature - part 2 - John Doe

[master] 665003d Fast bugfix - John Fixer

[myfeature] eaf618c New cool feature - John Feature

[master] 8f1e0e7 Merge branch 'develop' into 'master' - John Doe

[master] 6a3dacc Merge branch 'myfeature' into 'master' - John Doe

0.1

[master] abcdef0 Release of version 0.1 - John Releaser



The master branch is in gray with colour branches coming of and then being merged back

# Key comards?

In the terminal interface there are some useful commards:



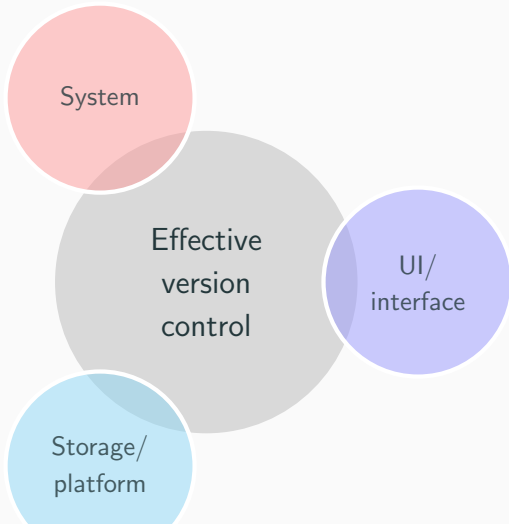
- *init* - inisholies a repo
- *status* - tells you if you have flies out of sync with the current verntion
- *commit* - adds commits to the current verntion - a change with a comment and an ID
- *add* - adds new files and current commit
- *remove* - removes files and current commit
- *push* - pushes comits to an online repo (only if using a git platform)
- *pull* - pulls commits from an online platform (only if using a git platform)

# Platform

---

# What platform?

You can just use git with the terminal on your PC or you can store work on an online repository managing site, the most popular of these are GitHub and Bitbucket



# Why not just store the repostry on dropbox?

You can and it would be baked up and sherable, but there are problems:

- You can run into Dropbox syncing and git version control clashes
- If you shere the files with a colaborator they are't identified separately to you
- You or a colaborator can completely mess up the git system (mostly by deleating the records in the '.git' folder)
- If you do you can't just re download





# What platform?

## Bitbucket

- Allows unlimited public or private repositories
- Charges per collaborator over 5 on the repositories
- 1GB storage for large files all repositories, 5GB if using a .ac email



## GitHub

- Has unlimited public repositories
- With unlimited collaborators
- Pay for private repositories
- However as a student or academic you can sign up for free private repositories
- Github is about  $\times 4$  more popular than Bitbucket

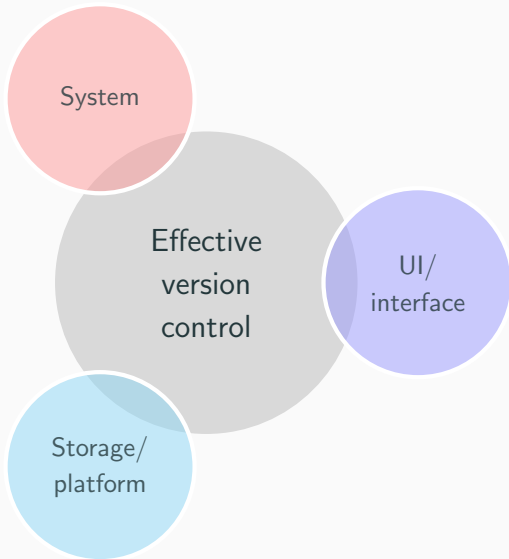


UI



# What UI?

You can just use git with the terminal but there is a friendlier interface



 Sourcetree

# What UI?

## The Terminal

You can just use the terminal with the raw git commands as seen before... but:

- If you are not using it anyway or are happy using it a graphical UI is nice
- You don't need to remember commands or repository names



```
C:\Windows\system32\cmd.exe

C:\Users\dgal\My Documents\GitHub\Presentations\GIT>git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   Figs/git/Dropbox.png

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   Danny_AC_Taguchi.pdf
    modified:   Danny_AC_Taguchi.tex

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    Figs/git/gitdesktop.png
    Figs/git/terminal.png

C:\Users\dgal\My Documents\GitHub\Presentations\GIT>git commit -m "added image f
or dropbox"
```

## Conclusions

---

THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.



