Welcome!

Virtual tutorial starts at 15:00 GMT

Please leave feedback afterwards at:
www.archer.ac.uk/training/feedback/online-course-feedback.php

Introduction to Version Control (part 1)

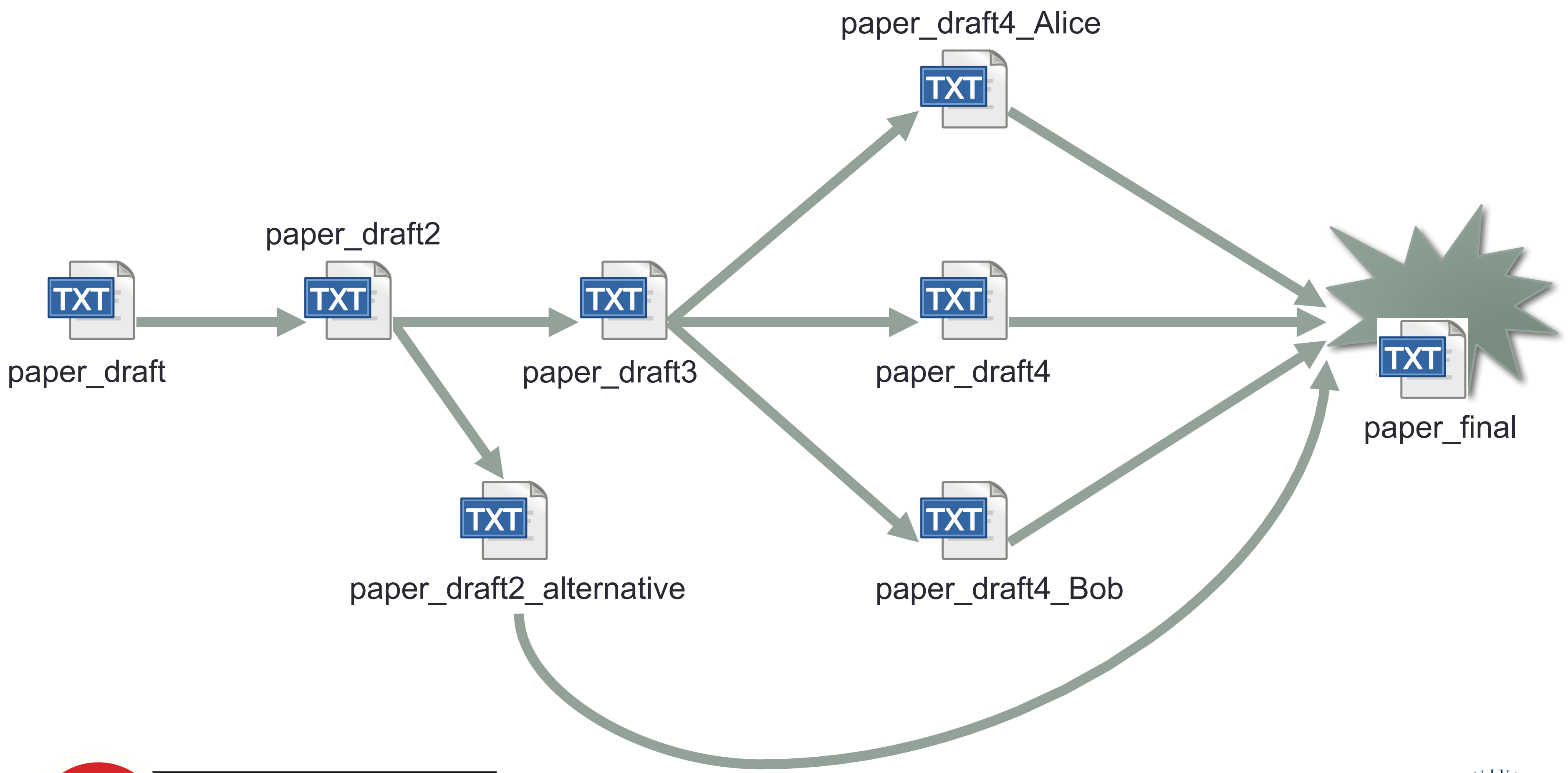
ARCHER Virtual Tutorial

Reusing this material

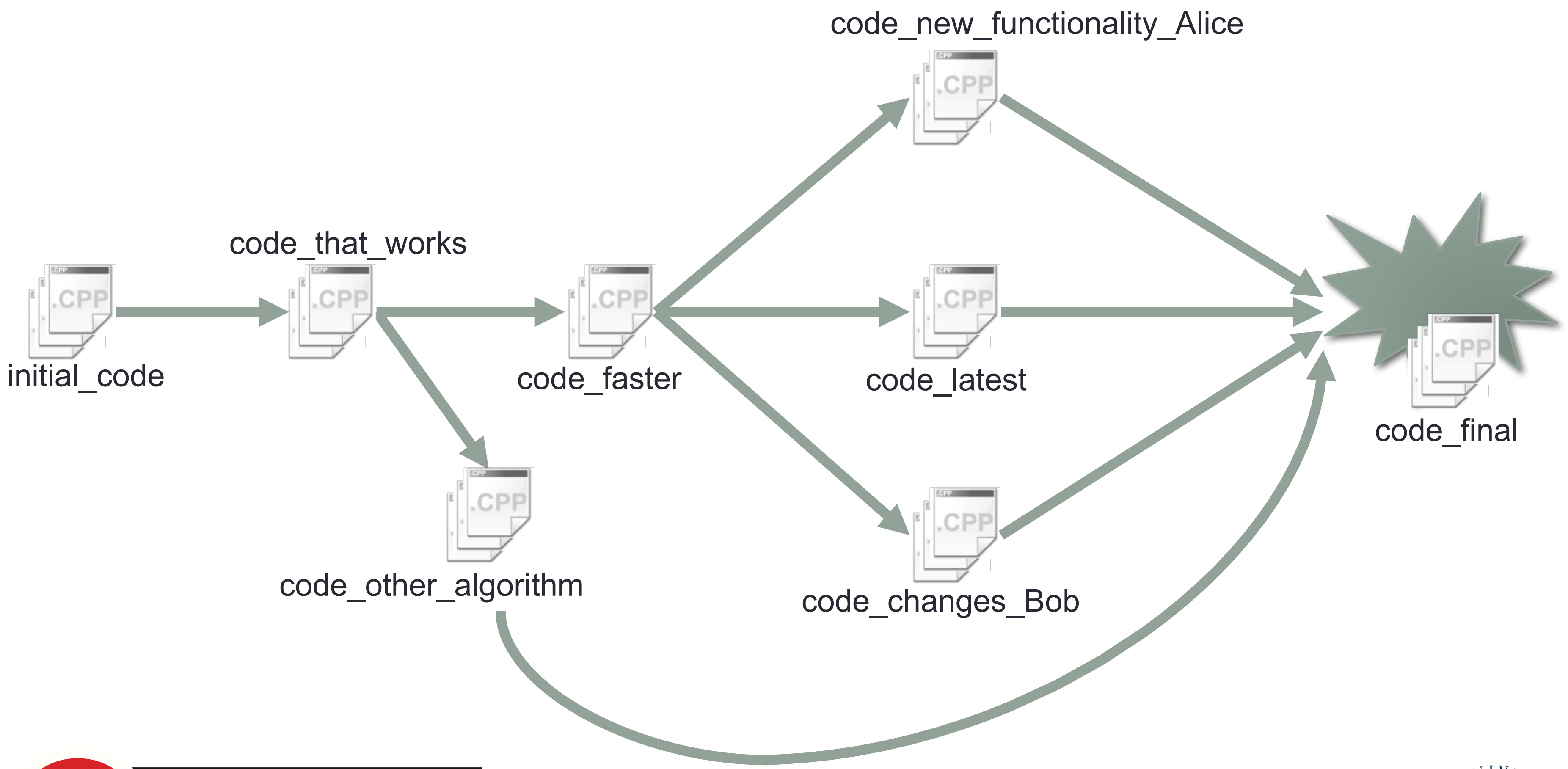
Outline

- Version Control – do it yourself?
  - What’s the problem?
- Version Control Systems
  - Benefits
  - Common version control systems
  - Common core concepts and terminology
- Simple demonstration using SVN
- Where do repositories live?
- Word of warning
- Practical demonstration

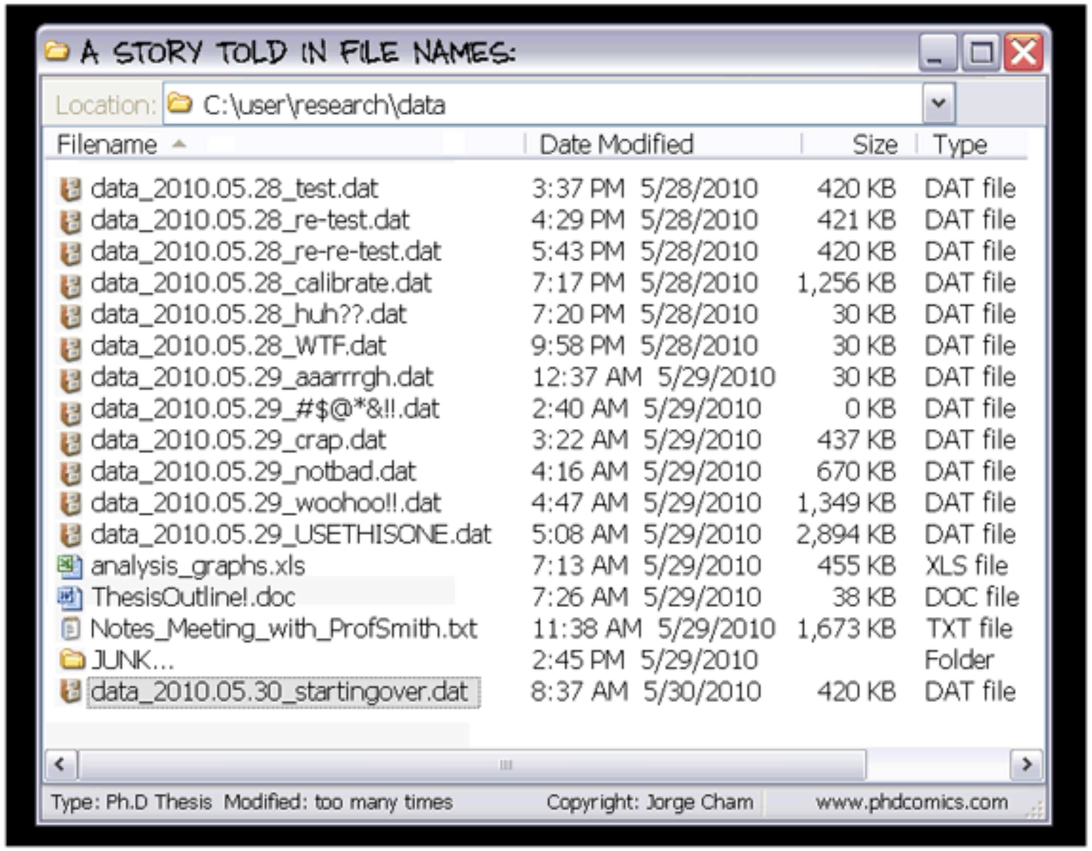
Version Control - Do It Yourself?



Version Control - Do It Yourself?



What's the problem?


A STORY TOLD IN FILE NAMES:

Location: C:\user\research\data

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

Type: Ph.D Thesis Modified: too many times Copyright: Jorge Cham www.phdcomics.com

What's the problem?

- Forced to manually keep track of
  - The distinctive content of each version
  - How versions of the same file are related
  - How versions of different files are related (e.g. code dependencies)
  - Which versions of which files should be used (together) as a basis for further work, for compiling an executable

Do we record this information in filenames and directory structure? Inside the files themselves? Elsewhere?

What’s the problem?

- Forced to manually merge versions:
  - To produce a “final” version that meets specific requirements by combining content from different versions
  - If you have edited copies of a file in different locations (e.g. laptop & desktop)
  - If other people have added useful contributions to copies of a file and you need to combine these

- Need to keep track of files resulting from these merges

This is (human) error prone and quickly becomes unmanageable for many files / many versions

Version Control Systems

These are software tools that:

- Provide a framework to record meaningful information about versions in a consistent, systematic way
- Help automate the tracking of versions and the changes between them
- Allow you to record the state of a collection of files at a given time as a snapshot and provide easy access to these snapshots
  - This captures and preserves dependencies between particular versions of files, e.g. source code

Version Control Systems

- Provide a safety net (can recover previous versions of snapshotted files)
- Allow for easy duplication and synchronisation of files in multiple locations
  - Avoids error-prone manual transferring of files
  - Can act as a backup of your data
  - Easily work on different machines
- Enable collaborative work on same set of files at the same time, automatically identifying contributions from different authors

Version Control Systems

- Thanks to automatic change tracking, we can
  - Make a copy of a set of files and modify these with a particular goal in mind (“branching”)
  - Merge these modifications (our “branch”) into
    - the original set of files (the “trunk”, or “master branch”) or
    - another set of files also originating from “trunk” but differently modified (another “branch”)

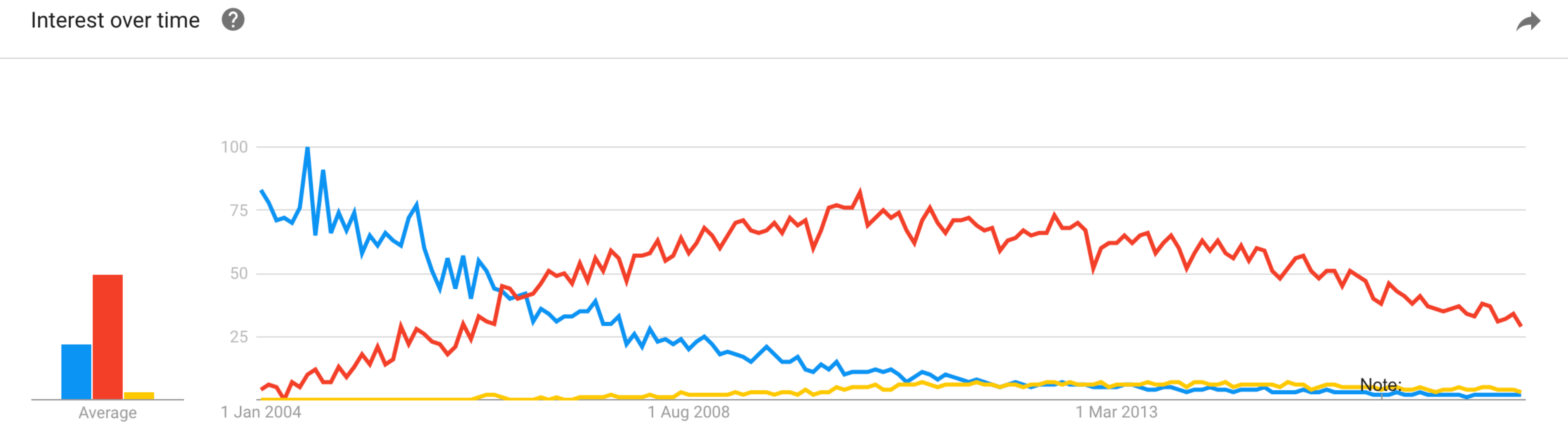
- We can use version information to facilitate
  - Reproducible computational research
  - Testing and development

Common Version Control Systems

Common open-source version control systems:

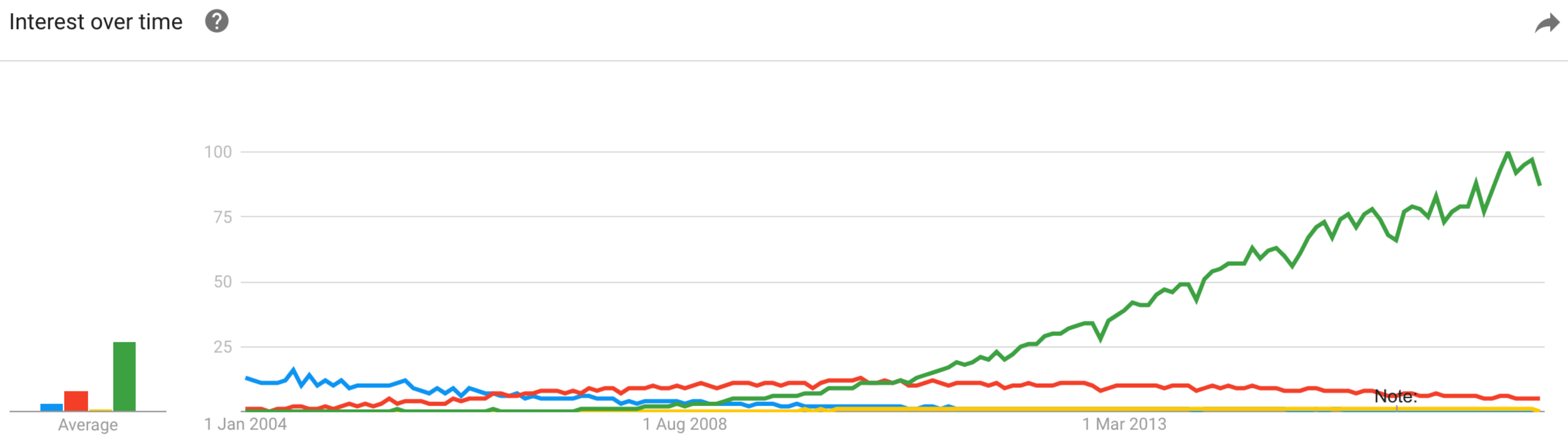
- CVS (Concurrent Versioning System)
  - mature, not as popular any more but still used
- SVN (Subversion)
  - successor to CVS, still somewhat widespread
  - more flexible and efficient than CVS e.g. at handling non-text files
- Git
  - newer, faster, powerful features, popular for many new software projects, able to handle complex workflows
- Mercurial
  - Like Git but simpler in some ways to use

Common Version Control Systems



- Source: Google Trends
- cvs svn mercurial

Common Version Control Systems



- Source: Google Trends
- cvs svn mercurial git

Common core concepts & terminology

Some concepts and terms are common to different version control systems:

- Repository
  - The complete archive / history of all the versions of files that were recorded (i.e. all snapshots), including how they are related and what change(s) led to each version.
- Check out (or “clone” for git)
  - To take a copy of the repository and duplicate it locally as a “working copy”
- Working copy
  - The set of all files contained in your local copy of the repository. Unless you have just checked out or updated your working copy its contents differs from the repository due to changes you have made to files, or due to the repository itself having changed in the mean time.

Common core concepts & terminology

- Commit
  - To record the current state of a file or set of files in your working copy to the repository as a version, or revision (a commit). This transfers data to the repository.
- Log
  - A record of which files in the repository were changed when, including (hopefully) meaningful comments by the author who made the changes.
- Staging area
  - A list of files that you have preselected to commit
- Add
  - To add a file or files to the staging area

Common core concepts & terminology

- A branch
  - A series of successive changes and commits to a copy of a set of files in the repository, typically done in order to explore a particular direction of development such as, in software, a new feature. The original files are left unaffected by any changes on the branch until you choose to do a “merge”.
- Merge
  - The combination of two versions of a file or files into one
  - This can lead to conflicts, which need to be resolved manually (the version control can point out conflicts but you need to think and decide what matters)
- Update
  - To attempt to merge the current state of the repository into your local working copy

Demonstration using SVN

- Check out an existing repository
- Look at the log
- Create a file
- Commit the file
- Make a change
- Commit the change

Where do repositories live?

- Repositories can live on a publicly hosted website (e.g. Bitbucket, Github), on a shared machine e.g. within your research group or institution, or on your own desktop or laptop

- More about this in Part II

Word of warning

- Version control systems are not a magic bullet, but a powerful tool
- You still need to think and decide how to manage your work
- When working collaboratively, need to communicate

- Scripted practical provided on ARCHER website
  - Using SVN to create a repository, add files, create branches, merge changes

- Part II will
  - explore the differences between centralised and distributed models of version control and local and remote repositories
  - Demonstrate the basics of Git and how it compares to SVN
  - Consider which version control system you might want to use