

GIT WALK-THROUGH

IMFORFUTURE MEETING 2018, ZAGREB

L.C. Karssen

25–28 March 2018

PolyOmega

's-Hertogenbosch, NL

l.c.karssen@polyomica.com



1. About version control systems in general
2. Git concepts
3. Git commands
4. Final notes

ABOUT VERSION CONTROL SYSTEMS IN GENERAL

WHAT IS IT?

- "track changes" on steroids
- for complete project, not a single file
- "Lab notebook" → reproducible research
- save files to a 'repository'
 - local (on your PC)
 - on a central server
 - on other computers
- record snapshots/changes + explanatory log message
- record relationships between snapshots
- → allows 'time travel'
- works for any file, but best for text files (scripts, csv, ...)
- Nothing gets lost (!!)

WHAT IS IT?

- "track changes" on steroids
 - for complete project, not a single file
 - "Lab notebook" → reproducible research
 - save files to a 'repository'
 - local (on your PC)
 - on a central server
 - on other computers
 - record snapshots/changes + explanatory log message
 - record relationships between snapshots
 - → allows 'time travel'
 - works for any file, but best for text files (scripts, csv, ...)
 - Nothing gets lost (!!)
- (unless you really try)

WHY VERSION CONTROL?

- history: what changed when (and by whom)
- clean working directory (no `script_v1`, `script_old`, `script_20170605`, `script1`, etc. lying around)
- allows experimentation (change, test, accept/revert)
- collaborate with others (sharing, integrating changes, conflict resolution) → branches
- sort of backup (when pushing to other computers)

EXAMPLE: CHANGES

```
► git diff b7f7bf6566981c62ac4706a594a6abf17960cece f30050a7807b4c947bc45fbed8b611c78dc87391
diff --git a/src/invsigma.cpp b/src/invsigma.cpp
index a2ac618..32bbf23 100644
--- a/src/invsigma.cpp
+++ b/src/invsigma.cpp
@@ -52,7 +52,7 @@ InvSigma::InvSigma(const char * filename_, const phedata& phe) : filename(filena
 {
     npeople = phe.nids;
     std::ifstream myfile(filename_);
-    std::string line;
+    char * line = new char[MAXIMUM_PEOPLE_AMOUNT];
     std::string id;

     matrix.reinit(npeople, npeople);
@@ -63,7 +63,7 @@ InvSigma::InvSigma(const char * filename_, const phedata& phe) : filename(filena
 {
     double val;
     unsigned row = 0;
-    while (std::getline(myfile, line))
+    while (myfile.getline(line, MAXIMUM_PEOPLE_AMOUNT))
     {
         std::stringstream line_stream(line);
         line_stream >> id;
@@ -99,6 +99,8 @@ InvSigma::InvSigma(const char * filename_, const phedata& phe) : filename(filena
         std::cerr << "error: inv file: cannot open file '"
             << filename_ << "'\n";
     }
+    delete[] line;
 }
```


SOME VERSION CONTROL SYSTEM HISTORY

- Source Code Control System (SCCS)
 - since 1972
- Revision Control System
 - since 1985
- Concurrent Versions System (CVS)
 - since 1990
 - very popular
- Subversion (SVN)
 - since 2000
 - very popular
- Git, Mercurial (Hg), Bazaar (Bzr)
 - since 2005
 - distributed
 - GitLab, GitHub, BitBucket
 - Git currently most popular

GIT CONCEPTS

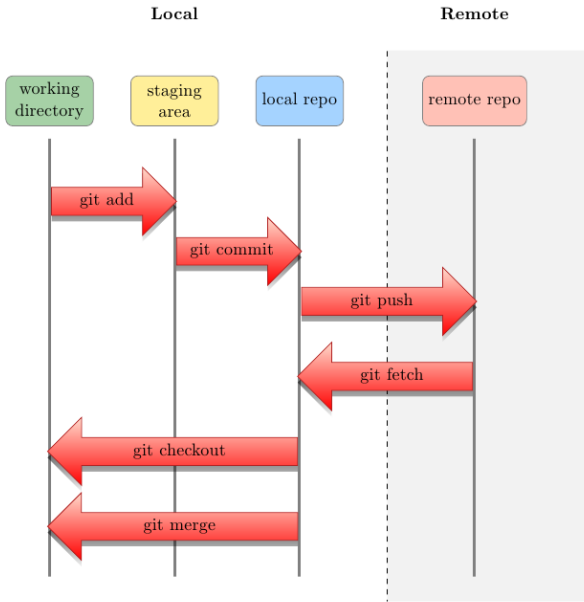
- Working directory:
 - the directory in which you are working on your files. This is where you actually edit and save files.

- Working directory:
 - the directory in which you are working on your files. This is where you actually edit and save files.
- Repository:
 - the place where all versions, log messages, etc. are stored.
 - generally the `.git` subdirectory in your working directory
 - files are *committed* to the repository
 - files are *checked out* from the repository

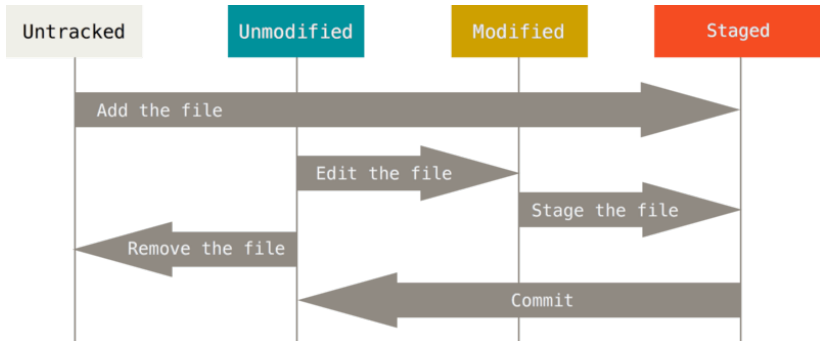
- Working directory:
 - the directory in which you are working on your files. This is where you actually edit and save files.
- Repository:
 - the place where all versions, log messages, etc. are stored.
 - generally the `.git` subdirectory in your working directory
 - files are *committed* to the repository
 - files are *checked out* from the repository
- Staging area:
 - A 'virtual' place. This contains all changes that you want to add to the next commit (so not yet in the repository) told Git to keep an eye on

- Working directory:
 - the directory in which you are working on your files. This is where you actually edit and save files.
- Repository:
 - the place where all versions, log messages, etc. are stored.
 - generally the `.git` subdirectory in your working directory
 - files are *committed* to the repository
 - files are *checked out* from the repository
- Staging area:
 - A 'virtual' place. This contains all changes that you want to add to the next commit (so not yet in the repository) told Git to keep an eye on
- Remote (optional):
 - Other location with a copy of the repository (e.g. in the cloud)
 - *clone, push, pull*
 - histories can diverge (!)

REPOSITORY, WORKING DIRECTORY, STAGING AREA



FILE LIFECYCLE



Source: <https://git-scm.com/book/en/v2/>

- Cloud services like GitLab, GitHub and BitBucket
- A central company/university Git server
- No single repo is the 'truth',
- unless all collaborators agree otherwise

- allows separate development, experimenting, etc.
 - branch with stable/tested code
 - branch for development of feature X
 - branch for bug fixes in stable version `x.y.z`
 - etc.
- create new branch using `git branch`
- merge one branch into another using `git merge`
- default: `master`
- although `master` is present by default, it is just a regular branch

BRANCHES: EXAMPLE HISTORY

	read-gzipped-genotypes	origin/read-gzipped-genotypes	Mention issue number in the CI	Lennart C. Karssen	dec 29 2016, 15:42
			Merge pull request #42 from lckarssen/read-gzipped-genotypes	Lennart C. Karssen	dec 29 2016, 15:37
			Update manual/Changelog with info on gzipped files	L.C. Karssen	dec 29 2016, 15:22
			Merge pull request #41 from lckarssen/read-gzipped-genotypes	Lennart C. Karssen	dec 29 2016, 14:53
			Add Boost.IOstreams library to travis requirements	L.C. Karssen	dec 29 2016, 14:40
			Allow reading from gzipped map files	L.C. Karssen	dec 28 2016, 23:34
			Allow compressed invsigma files when running with mmscore	L.C. Karssen	dec 28 2016, 17:41
			Add tests for gzipped info and geno files for MMS	L.C. Karssen	dec 28 2016, 16:49
			Minor code improvements in the gendata class	L.C. Karssen	dec 28 2016, 16:47
			Add missing .gz files and missing file to Makefile.am	L.C. Karssen	dec 28 2016, 22:43
			Implement reading from gzipped info files	L.C. Karssen	dec 28 2016, 16:46
			Add tests for gzipped geno input	L.C. Karssen	dec 18 2016, 16:41
			Minor changes in gendata.cpp, mostly code layout	L.C. Karssen	dec 16 2016, 16:50
			Merge branch 'master' into read-gzipped-genotypes	L.C. Karssen	dec 29 2016, 13:45
	master	origin/master	Merge pull request #39 from lckarssen/master	Lennart C. Karssen	dec 28 2016, 17:04
			Improve status message when reading pheno data	L.C. Karssen	dec 28 2016, 16:24
			Convert the allmeasured int array to a boolean vector	L.C. Karssen	dec 28 2016, 16:20
			In invsigma: use strings instead of char[] with fixed buffer size	L.C. Karssen	dec 28 2016, 16:06
			In phedata: use strings instead of char[] with fixed buffer size	L.C. Karssen	dec 28 2016, 14:15
			Merge pull request #38 from lckarssen/master	Lennart C. Karssen	dec 27 2016, 17:06
			Read/open the info file only once	L.C. Karssen	dec 24 2016, 18:04
			In mlinfo: use strings instead of char[] with fixed buffer size	L.C. Karssen	dec 23 2016, 00:09
			Fix incorrect line number in error message	L.C. Karssen	dec 16 2016, 18:06
			Add doxygen explanation of skipd param	Lennart C. Karssen	dec 16 2016, 17:19
			Minor code layout changes in cli settings	L.C. Karssen	dec 14 2016, 17:50
			Move progress message to correct location	Lennart C. Karssen	dec 14 2016, 17:23
			Fix doxygen description of text file read function	Lennart C. Karssen	dec 14 2016, 16:41
			Merge pull request #36 from lckarssen/fix_issue31	Lennart C. Karssen	dec 6 2016, 18:12
			Fix a memory leak in pacoxph	L.C. Karssen	dec 6 2016, 18:05
			Merge pull request #34 from lckarssen/fix_issue31	Lennart C. Karssen	dec 6 2016, 17:10
			Merge pull request #35 from lckarssen/fix_issue32	Lennart C. Karssen	dec 6 2016, 16:59

GIT COMMANDS

A Git command typically looks like this:

```
git verb
```

Options can be added too:

```
git verb --option_A --option_B
```

Two ways to get more information on a git command:

For a quick list of options for **verb**

```
git verb -h
```

Show the manual page for the given **verb**

```
git verb --help  
man git verb
```

FINAL NOTES

- Store only scripts, not their output
- Works best for plain text files, not binary files
 - e.g. for MS Word documents you are better off with its internal versioning.
- Not well suited for large (data) files (→ Git LFS)
- Make clear, concise commit messages:
 - 1st line: describe what is done
 - 2nd line: empty
 - 3rd line and further: explain the changes, reasoning, etc.
- Use explanatory names for branches

- Start small
 - one project
 - only for yourself
 - add, commit, show log
- Continue with branching or remote repositories

Cheat sheets

- GitLab cheat sheet; has nice figure with work spaces etc.
- GitHub cheat sheet

Git courses/tutorials

- <http://swcarpentry.github.io/git-novice/>

Other links

- *A Quick Introduction to Version Control with Git and GitHub* in PLoS Comp Biol