Programming (ERIM)

Lecture 8: Version control (git), namespaces, packages

Tommi Tervonen

Econometric Institute, Erasmus School of Economics

Why version control?

- To allow multiple developers to work on the same program
- To do version control (!)
- Allow branching and merging
- Store moments in development when a new feature (e.g. a function) has been implemented completely, and enable changing the program files back to that moment easily

Version control systems (all open source)

- CVS (Concurrent Versioning System): old, client-server architecture, no renaming of files or directories
- SVN (Apache Subversion): client-server architecture, designed as a successor to CVS. No tagging (tags are just full copies of the repository. Limited file renaming.
- Git: made initially by Linux Torvalds for Linux kernel development. Distributed system (no need for a server), used by companies such as google, facebook, m$, twitter, and in various open source projects (also by me! see `http://github.com/tommite`)

Version control concepts

- Commit
- Branch
- Tag
- Merging
- Push and pull
- Fork (see `http://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg`)

http://try.github.com

Namespaces in R and Matlab

- Namespaces allow to group together symbols (e.g. functions)
- Different namespaces can have same identifiers
- R packages define namespaces, members accessed with '::', e.g. 'stats::var'. 'library(package)' imports the package to the current namespace (no need to use namespace specifier anymore)
- Matlab packages define namespaces, members accessed with '.', e.g. 'mypkg.func'. Matlab packages are constructed by having files in directory starting with '+'. Packages imported with 'import pkg'

CRAN is the main reason for the awesomeness of R

- Comprehensive R Archive Network
- 5064 packages
- Easy install with `install.packages`
- Don't implement standard algorithms yourself, use existing ones (and extend them, if required)