notes_week2

Benjamin Cole

April 4, 2018

Application stack

Set of tools to be competant as data scientists

Applications stack Need a balance of applications for each task; limited no. of good choices. Most (but not all) are open source

Language *R & Python (either/or?)*; complementary Spark, scala sas IDE R: Rstudio only option (older tools are superceded) Python: jupyter notebooks. (Berkeley inst. for data scientists) VC *Git* Only tool being used. Natively involved in Rstudio, jupyter, etc. Invest time and learn git from command line. Not complicated. Some things not possible from Rstudio, e.g.

Code management Github, atlassian project (bitbucket, competitor to github). Bitbucket allows limited no. of closed source project for free *Gitlab* (open source) - run in house.

Scalability Spark, Spark ML, AWS, H2O, google R is not high performance or scalable. Great if data can be kept on one server. (also true of python). Azure C/C++ to speed algorithms nVidia Most can be called directly from R or python.

Developemnt Agile: too heavy for data science workflow Scrums, meetings, sprints more variability & unknown Most agile efforts, you know there's an achievable endstate. DS is empirical, research-based. Can use agile to track workflow and stop there. YMMV. Cambam? *JIRA servicenow Git*/Gitlab Can manage/track projects in github.

Deployment Where endpoint is – who is consuming project Spreadsheet? Database? Called by web app? Build a specific app? ALl have different software.

Reporting Use Rmarkdown, develop set of libraries that spit out reports.

Difference between vc and code management? Git is not github! Github (repositories) does code management, git (version control) does version control.

Git

Source control tool/process to promote (enable) collaborative development. Features Distributed your copy has entire history of project gives ability to return to any point in time. supports branching and merging Work on branch w/o interrupting main branch test changes merge and make part of main project. Can edit on github directly SourceTree (best GUI for git) Rstudio has built in Command line Start project (new): first thing is *clone* Get a copy of the project, bring down to workspace. *Checkout* (get latest commits). *Pull* pull remote commits. Do work. Make changes. Git does not automatically save work. have to explicitly tell what to incorporate. Ask for git status? What changed? git add takes wildcards. have to specify each one (metacharacters). Adding isn't enough, just says file is ready to be added (*staged*). Have to commit. Add, then commit. Tells git that files should be incorporated.

Before pushing, generally does another *pull*, gives change to fix conflicts. Then push.

95% of what we do in git.

Do adds and commits frequently!

Often best to create branches, do work in the branch, for the particular purpose of accomplishing one thing. Branches are ephemeral, when ready, can merge back in.

Fork is a notion on github, not a git command.

Have class repo -> fork -> our repo -> clone -> Laptop repo

Then pull from class repo to get new assignment.

git vs. svn?

Fully distributed (git). SVN uses master repository. Once repo is cloned in git, there master repository relationship is lost. Keeps track only of differences of binaries, uses less storage. When making a change to a file, only tracks the differences. Big thing w/ git: if you want to put binary files on git, and binary changes, git will make an entirely new copy. Includes word docs. Git is mostly for text files and code.

diff bw/ checkout an dpull

```
checkout applies to branches
pull applies to head of working branch, tries to merge them in. Pull = fetch + merge. Almost never use
```

Repositories

Repository is where files are stored Forked repos are if the repo is different than original source. Repote is named storage location of a repo Paritcular to git Doesn't mean in cloud; just a named storage location, regardless of where it is. Remote is name you call it. upstream is a remote. associated with web url. Can have multiple remotes Origin is another remote. Associated with github.com/id/project (a pointer) Allow us to git pull from whatever remote, on whatever branch. Can define remote however you want. Only defined at each local repository.

Branch copy of code than can be independently worked on. Master branch is where stable accepted version of code is stored. Can have development branch, branch for each issue, etc.

Staged Changes ready to be accepted, but not accpeted. Commit – every commit gets an id. Can go back to any point in history.

Git commands have same format *git* [verb] [options] [target]

Verbs (navigation, editing, synchronization, informational) branch (navication) creates branch. checkout (navication) switches into branch. git checkout -b name (creates branch, chekcs out, and moves into branch) add (editing) git add files commit (editing) git commit (can commit individaul files) all commits have comments. forces to tell future self what you were working on. pull [remote] [branch] (synchronization) get changes from remote push [remote] [branch] (synchronization) send committed changes to the branch (*best practice to be explicit*) clone [remote] (synchronization) log (informational) review history of commits status (informational) most useful in conjucntion w/ commit/add, see which files are staged.