Version control via Git

Git (and GitHub) facilitate:

- tracking changes to files
- easy comparison of different versions
- identifying and undoing changes that introduced a bug
- orderly collaboration
- distributed development in which files are stored locally on each person's computer, but all local copies can be synchronized

Each file is in one of these states in the git repository of snapshots (`.git` directory):

- untracked (so not in the repository)
- modified: changed but not committed to snapshot; can be staged
- staged: ready to be included in next snapshot; can be committed
- committed: in snapshot; can be modified

Here are some essential commands:

- `git config` configures git; e.g.

  ```
  git config --global user.name "your name"
  git config --global user.email "your email address"
  git config --global core.editor emacs
  git config --list
  ```

- `git init` creates a `.git` directory in `.` in which to store a version control repository. e.g.

  ```
  mkdir ~/git
  cd ~/git
  git init
  ls # note: files and directories beginning with "." are hidden
  ls -a # list all files, including hidden ones
  ```

- `git add <filepattern>` adds file(s) to be tracked to a *stage* before committing.
- `git commit -m "<message>"` saves a snapshot of files on the stage (at the time they were added) to `.git` along with a message describing changes.

  Git stores a graph of snapshots, with each new snapshot linked to a previous one. (Early on, the graph is just a linked list: `first <- second <- third`.) e.g.

```
# make two files for practice with git
echo 'Anna,6' > kids.csv # write message to stdout; ">" redirects output to file
echo 'Teresa,9' >> kids.csv # ">>" appends to file
echo 'kids = read.csv("kids.csv", header=FALSE)' > kids.R
echo 'colnames(kids) = c("name", "age")' >> kids.R
git status # note untracked files

git add kids.csv kids.R
git status # note staged but not commited files

git commit -m "initial commit"
git status # note clean status
```

- `git branch -M main` renames branch from "master" to "main" (git's default to GitHub's).
- `git log [--pretty=oneline]` shows a log of commits and their *hash* labels.

Branching facilitates multiple versions:

- `git branch` shows the available branches; the current one is marked with an asterisk `*`.
- `git status` describes the current branch and its changes, staged and unstaged
- `git checkout` retreives a snapshot.
  - `git checkout <branch>` retrieves the specified branch; the typical branch is `main`
  - `git checkout -b <branch>` creates a new specified branch and checks it out
  - `git checkout <file>` retrieves `file`, overwriting/undoing any changes to `<file>`
  - `git checkout <commit>` retrieves the specified `<commit>` (from `git log`)

  e.g.

```
git checkout -b fun # create branch to add a new feature, favorite fun
# ... use emacs to add the column c("swing", "trampoline") to kids.csv
# ... and add the column name "fun" to kids.R (along with "print(kids)")
git diff kids.R
git add kids.csv kids.R
git commit -m 'added "fun" column to data and code'

# Now return to main branch:
git checkout main
cat kids.csv # these files do not have the "fun" revisions
cat kids.R
```

- `git diff` shows changes between commits, commits and working tree, branches, etc.

- `git merge ...` makes a commit to reconcile a branch into the current one. e.g.

  ```
  git merge fun -m "bring fun branch changes to main branch"
  git branch -d fun # delete fun branch now that we are done with it
  ```

  There can be conflicts we have to resolve manually. e.g.

  ```
  git checkout -b drink  # create branch to add favorite drink
  # ... use emacs to add column c("milk", "juice") to kids.csv
  # ... and add column name "drink" to kids.R
  git add kids.csv kids.R
  git commit -m 'added "drink" column from drink branch'

  # Return to main branch:
  git checkout main
  # ... use emacs to add column c("milk", "soda") to kids.csv
  # ... and add column name "drink" to kids.R
  git add kids.csv kids.R
  git commit -m 'added "drink" column from main branch'

  # Now merge and see conflict; resolve it manually.
  git merge drink -m "bring drink branch changes to main"
  # ... use emacs to choose "soda" line to keep in kids.csv
  git status
  git add kids.csv
  git commit -m "retained soda as Teresa's drink"
  git branch -d drink # delete branch we no longer need
  ```

GitHub facilitates storing a git repository online, where it is accessible to all collaborators:

- `git clone <repository>` clones `<repository>` from GitHub to local machine
- `git pull origin main` downloads and merges with another repository from GitHub (pull only with no uncommitted local changes)
- `git push origin main` uploads a local repository to GitHub

One typical workflow is `git add`, `git commit`, `git pull`, `git push`.

See gitExercise.pdf for a class example of the use of git and GitHub.

For more information,

- See `https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf`
- Check manual pages via `M-x man git` and `M-x man git-COMMAND`, where `COMMAND` is any of the `git` commands above, e.g. `M-x man git-status`
- See the "Pro Git" book at `https://git-scm.com/book/en/v2`, especially its chapters 1-3
- See `http://swcarpentry.github.io/git-novice`