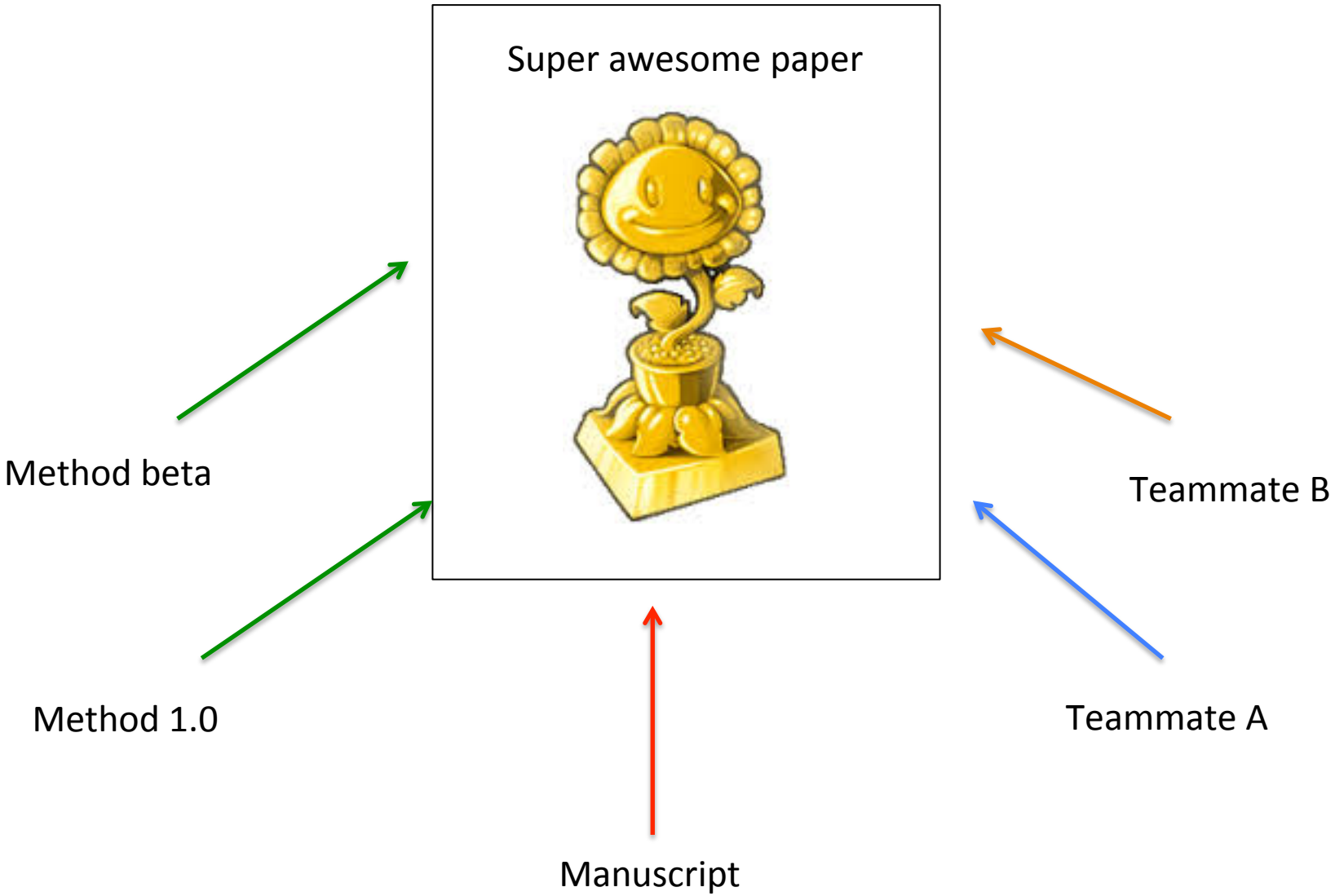
Collaborating on GitHub

Stephens lab meeting

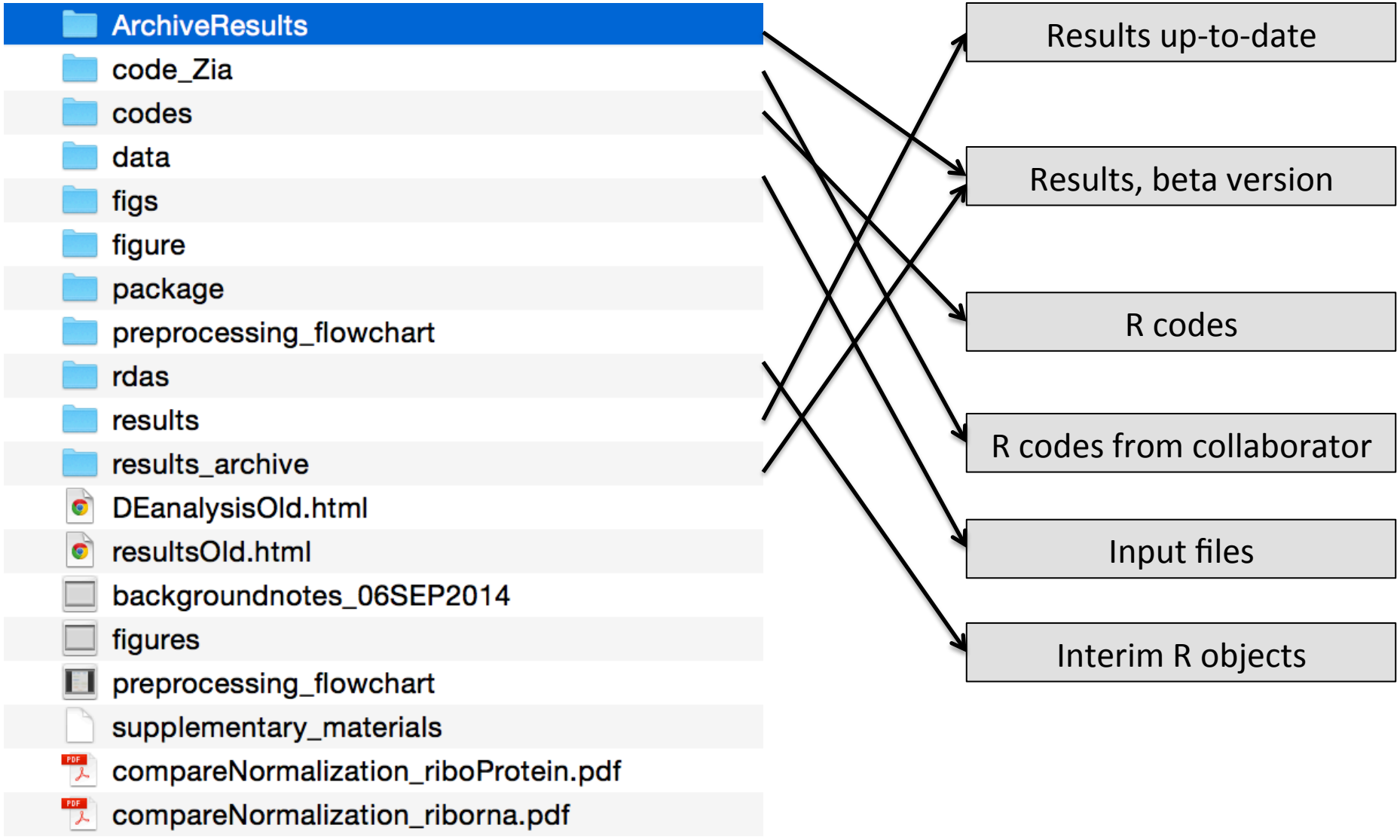
Joyce Hsiao

2015-12-03

We work as a team



Before GitHub…



A workflow that works!

John Blischak says:

1. "Everything" goes into GitHub repo
2. Create a project website hosted on GitHub

JB's Contributing guidelines

https://github.com/jdblischak/singleCellSeq/blob/master/CONTRIBUTING.md

- Running RStudio Server
- Creating a new analysis
- Style guide
- Adding figures
- Building the site
- Building the paper
- Adding citations

Highly recommended! Helped me to keep my codes and repo Kosher!

Step-by-step guide for first-timers

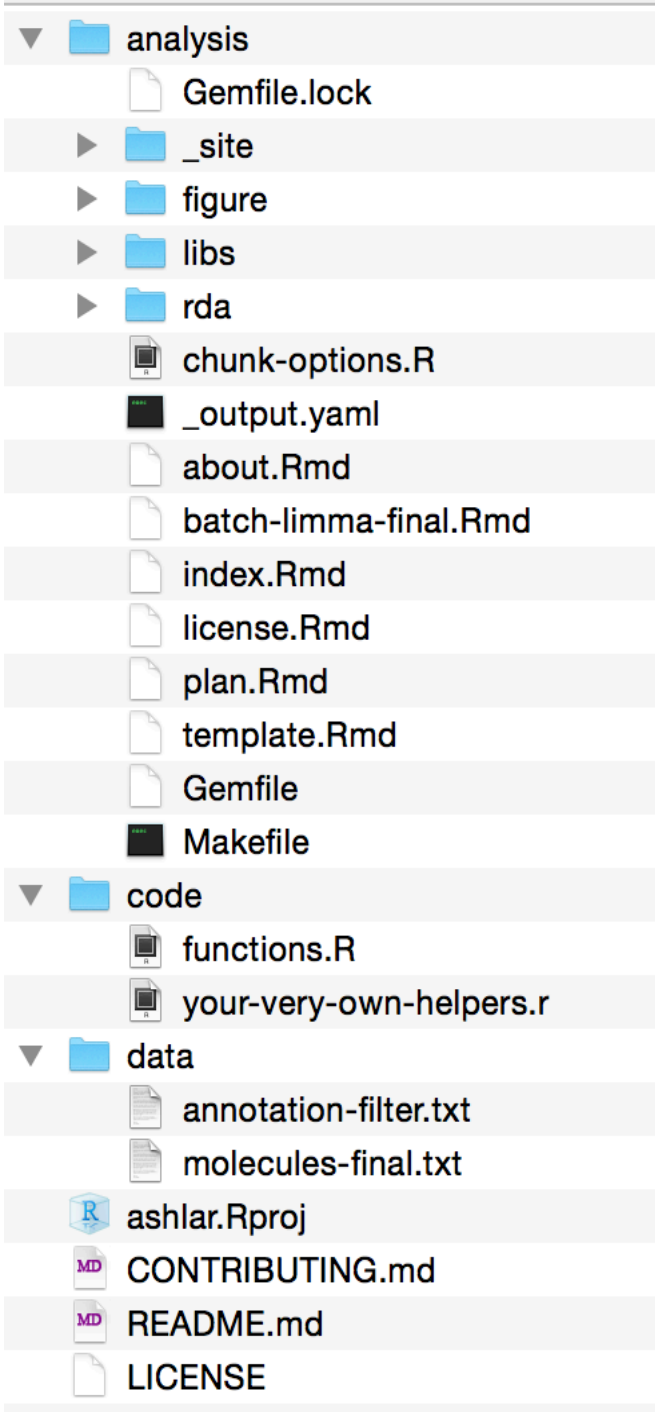
A. Making a project directory
B. Setting up GitHub repo
C. Setting up Jekyll pipeline
D. Prepare for publishing

Where I learned about this:
http://ialsa.github.io/tutorials/gh-pages-setup.html

GitHub repo folders

- Analysis (ALL Rmarkdown files and htmls)
  - Figures (exported directly by user or as a by-product of Rmarkdown files)
  - Output (R objects, work-in-progress analysis results)
- Code (bash scripts, fresh R functions, or R functions that may not contribute to method development)
- Data (data that do not change frequently over the course the analysis, such as read count or phenotype information)
- Docs (method drafts)

https://github.com/jhsiao999/ashlar



A. Making a project directory

1. Clone ashlar and rename the repository

```
git clone https://github.com/jhsiao999/ashlar.git ashlar-trial
```

2. Open ashlar.Rproj (R project object) in the analysis directory. Once you do so, working directory of the current R session becomes *ashlar-trial/analysis*. No more specifying user-specific home directory. This is especially when working on collaborative projects.

3. Add your project information:
   - *analysis/*About.Rmd (project description)
   - *analysis/*Index.Rmd (homepage for the website)
   - *analysis/*License.Rmd (my default is Creative Common)
   - *analysis/*Template.Rmd (an example Rmarkdown template)
   - *ashlar/*README.md (github repo README)
   - *ashlar/*analysis/include/before_body.html (webpage header)

B. Making htmls

1. Run make command. This makes htmls for the Rmds that do not have an html.

```
cd ashlar-trial/anaysis
make
```

To make htmls for all of the Rmds, doesn't matter if they already have a html in the directory or not:

```
cd ashlar-trial/anaysis
make -B
```

2. Use knitr to compile Rmds into html files. If you work with RStudio, simply click on "knit html" in the tool bar.

B. Setting up GitHub repo

1. Reset git remote directory.

```
git remote rm origin
git remote add origin https:/github.com/jhsiao999/ashlar-trial.git
```

2. Go to to github.com. Create a repo called ashlar-trial. Then, commit all files

```
git add --all
git commit –m “first commit”
git push origin master
```

C. Producing the website (not publishing)

This step assembles the htmls and generates a webpage, based on index.Rmd.

1. Setting up Jekyll pipeline. Here I assume you already have Ruby.

```
cd analysis
sudo gem install bundler
bundle install
```

2. Producing the website

```
cd analysis
make
bundle exec jekyll serve
Localhost:4000
```

To learn more about this step:
http://ialsa.github.io/tutorials/gh-pages-setup.html

C. Publishing website

Deploy to gh-pages.

```
git checkout gh-pages
git push origin gh-pages
```

Give it a minute, then wala!

https://jhsiao999.github.io/ashlar-trial

D. Add new analysis

Once you have finished working on an Rmd file,

```
cd ashlar
git checkout master
git add new-analysis.Rmd
git commit –m “add new analysis”
git push origin master
```

Push the master branch to the gh-branches, run Make file and then push the updates to the gh-branches:

```
git checkout gh-pages
git merge master
cd analysis
make
git add --all
git commit –m “build site”
git push origin gh-pages
```

Workflow in one slide

1. Start from the master branch. Add the new analysis Rmd and also the updated homepage (index.Rmd). Htmls are not committed by default (set in .gitignore).

```
cd ashlar-trial
git checkout master
git add new-analysis.Rmd index.Rmd
git commit –m “add new analysis”
git push origin master
```

2. Switch to gh-pages. Make and add htmls. “git add –f” overrides .gitignore default and forces add htmls. Push the website.

```
git checkout gh-pages
git merge master
cd analysis
make
git add –f new-analysis.html index.html
git commit –m “build site”
git push origin gh-pages
```

A useful Git tip

If > 1 person contributes to the repo on a regular basis, do the following to avoid disasters!

```
git checkout work-branch  ## move the pointer to local work branch
git add new_edits
git commit -m "new_edits"
git push origin work-branch  ## push edits to remove work branch

### At this point, you can make a merge and a pull request to review the edits

git checkout master  ## move the pointer to local master
git pull origin master  ## fetch and merge remote master to local master
git merge work-branch  ## merge local work-branch into master and update master
git checkout work-branch  ## move to local work-branch
git merge master  ## merge remote master to local work-branch
git push origin work-branch  ## move the pointer back to local work-branch

### At this point, the commit numbers of your work-branch and master should be the same!!!!!!!!!!!!!!!!!!!
```

https://github.com/jhsiao999/singleCell-method#collaborating-on-the-project

Other tips

You don't have write permissions for the /Library/Ruby/Gems/2.0.0 directory.

sudo gem update --system