

# Example workflow: Getting organized using GitHub and RStudio

Olivier Binette

# Agenda

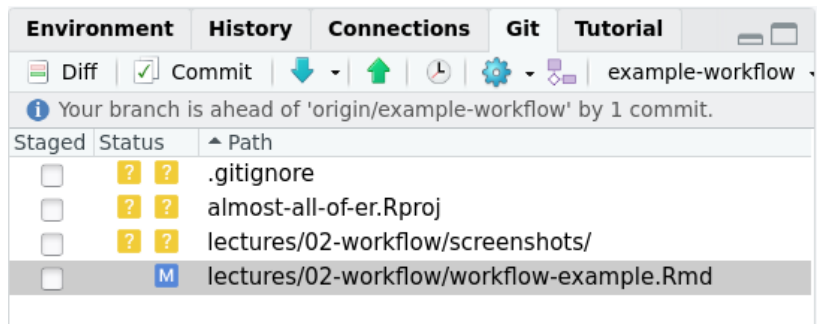
- ▶ Getting set-up in RStudio
- ▶ Organizing your projects
- ▶ Using GitHub for project management

## Getting set-up in RStudio

# Getting set-up in RStudio

## Why use RStudio?

- ▶ Convenient **git pane** to manage a git project (commit, push, pull, etc)



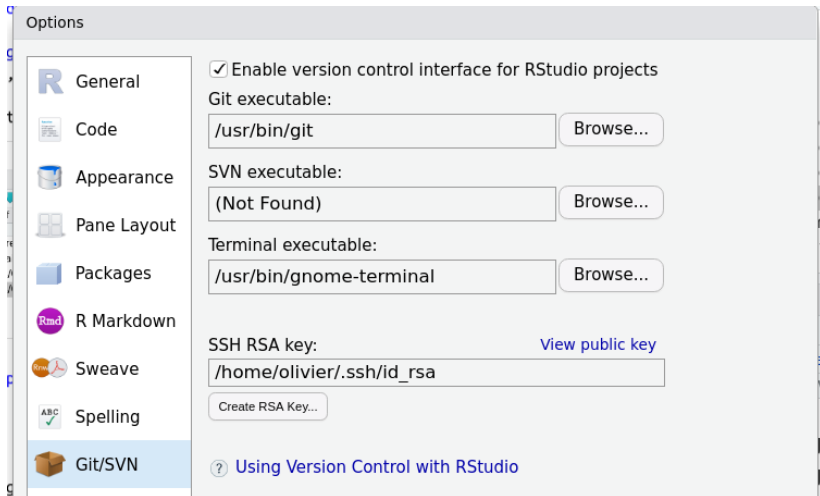
The screenshot shows the RStudio interface with the Git pane active. The pane has tabs for Environment, History, Connections, Git, and Tutorial. The Git pane displays a toolbar with icons for Diff, Commit, Push, Pull, and other Git operations. Below the toolbar, a message indicates that the current branch is ahead of 'origin/example-workflow' by 1 commit. A table lists the files in the repository, showing their status (Staged) and path.

Staged	Status	Path
<input type="checkbox"/>	??	.gitignore
<input type="checkbox"/>	??	almost-all-of-er.Rproj
<input type="checkbox"/>	??	lectures/02-workflow/screenshots/
<input type="checkbox"/>	M	lectures/02-workflow/workflow-example.Rmd

# Getting set-up in RStudio

**First step:** identify yourself using a SSH key

1. In RStudio, go to Tools -> Global Options. . . -> Git/SVN
2. Click the “Enable version control” box if it's not there already.



# Getting set-up in RStudio

## 3. Create a SSH RSA key

Create RSA Key

The RSA key will be created at: [SSH/RSA key management](#)

Passphrase (optional):

Confirm:

Create

Cancel

# Getting set-up in RStudio

## 4. Copy the key to your clipboard

### Public Key

Press Ctrl+c to copy the key to the clipboard

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGDnffhQiP+cW+CRLVLGFG5n580rTEkH
ZOBI4SWnh+wnSubAWJSEIUx09vlsIdhUYFQe9rmS0+0iB8ob+1riH4kTy+xm
7kQNoCcNBkSzd7AZ9rKca4KZdLzWxP/8T93cBYCEfBs9WSBuxRX2HV6e4F1E
ZZ1NoTq6IHNVLqu0iZFfRr054wTeKrIDc5dHv44yN/PZAawNQVrKgjt/HNS2
3RqMEsypehnyLA23AaxSyJejpi920MBsf8z2EY9mIVzAQ6HrZ/2s4nFSV5H
vJfB/LEBvBoXo9u7iaT1J5SJcdFbTtd1DduRZiLQW8Go+LST6My/98RL09bi
RCAuN/iWe3B59MnK3z8oB24i+y6poHuNmWT1jEFk9J2U6fNvRcOd4rK1Cpwo
JrL6db2uA1n0KJsDpRFD16t3AwowzDpNodhVYgAvb/BJpYHKmbZ7UMN3sSiA
d9gNZLDrWMUXc86DXX5ea5/TikIoTLmf9qosqkFdPXMT6FvHrKAHmET6A7Rc
vwc= olivier@home
```

# Getting set-up in RStudio

## 4. Register the key on GitHub:

- ▶ Under your account tab, got to Settings -> SSH and GPG keys -> New SSH key



**Olivier Binette**

Your personal account

[Switch to another account](#)

[Go to your personal profile](#)

Account  
settings

Profile

Account

Appearance New

Account security

Billing & plans

Security log

Security &  
analysis

Emails

Notifications

## SSH keys / Add new

Title

Olivier's Computer

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGCsByvLkdWnP17wGSceVV4IW
jTH3rq9cO6DSeugzfoqkaOIAIOihpdDKixkPcZx5Qmzsc8winINVeNEmXcg5
p57PBb31E/N1wuYVbINij1YXVRLjtmYWzlk5wAfAkqkmPF6LghpayHv8ieZ3
sEAeZ14jhGP3kk3Jd8OLPFX
/3xt/9bfeRtKcn3qeMpkOYRK9jhJBZV8gxD9XSnY5ke2f
/ejKxuWJ3nI4ZI4blyUfekUacSxjx1zj9jAr74qpjW6rWEORN7E68cn9BGW5L
X9Fek+r6pBqBwaHIXLG23tYKY383cl1yvB5LLJBHuN
/beQhov2ZuXYtB6ea6cAL7AKPV2rowU7ulfecGkJTXABoiLUQhzNHH+Kau
```

Add SSH key



# Getting set-up in RStudio

5. GitHub can now recognize RStudio as being associated to your account!



Using RStudio's git pane, you can now:

- ▶ pull from the repo, commit your changes, and push your changes,
- ▶ create new branches.


# Getting set-up in RStudio

**Note: make sure to clone repos using SSH:**

## 1. Download ssh address


 **Clone** 

HTTPS SSH GitHub CLI



Use a password-protected SSH key.

---


 **Download ZIP**

# Getting set-up in RStudio

2. Create a new project in RStudio using this ssh address
  - ▶ File -> New Project... -> Version Control -> Git
  - ▶ Add the ssh address as the repository URL.

New Project Wizard

**Back** **Clone Git Repository**



Repository URL:

Project directory name:

Create project as subdirectory of:  
 **Browse...**

☐ Open in new session

**Create Project** **Cancel**

## Organizing your project

# Organizing your project

**Tip 1:** Create a **R/** folder where you put re-usable functions.

- ▶ Document these functions using Roxygen2 and `devtools::document()`
- ▶ Load all of these functions using `devtools::load_all()`
- ▶ Show examples of the use of these functions in Rmd files under a **vignettes** folder.

# Organizing your project

**Tip 2:** Have a single place where you put your reproducible analyses.

- ▶ Create an “analyses” folder where all reproducible analyses are contained.
- ▶ Each analysis is in its own subfolder.
- ▶ Each analysis contains the folders **input**, **src**, and **output**.
  - ▶ **input** is never changed
  - ▶ running the code in **src** together with what you need in **input** always creates the same result in **output**.

# Organizing your project

**Tip 3:** It's ok to mess up!

- ▶ Create an **experiments** folder where you can put... well, your failable experiments.

# Organizing your project

**Tip 4:** You'll need to write that up.

- ▶ Create a **writeup** folder where you put your TeX writeups.



# Organizing your project

**Tip 5:** Avoid file path issues using the “here” package.

- ▶ Place an empty file named “.here” at the project root.
- ▶ Using there “here” package, you can obtain the path to the project root by calling the function `here()`.
- ▶ Refer to project files using e.g. `here("R/my_code.R")`

This avoids common issues related to file paths using R/RStudio.

# Organizing your project

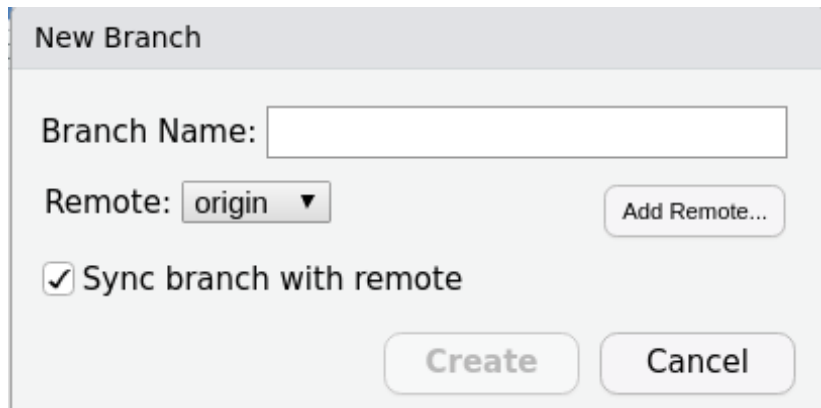
Overall, your project could be structured like this:

- ▶ `.here`
- ▶ `R/`
- ▶ `vignettes/`
- ▶ `experiments/`
  - ▶ `olivier's-buggy-code.Rmd`
- ▶ `analyses/`
  - ▶ `PCA/`
    - ▶ `input/`: `data.rds`
    - ▶ `src/`: `1-parse-data.R`, `2-PCA.R`, `3-make-plots.R`
    - ▶ `output/`: `pca-results.rds`, `plot.pdf`
- ▶ `writeup/`
  - ▶ `main.tex`
  - ▶ `biblio.bib`

## Organizing your project

**Tip 6:** Want to avoid merge conflicts?

- ▶ Branch off!
- ▶ Tweak that analysis in a new branch. Once you and your team is happy with the changes, you can make a pull request to merge back the changes.



The image shows a 'New Branch' dialog box with a light gray header. Below the header, there is a text input field for 'Branch Name:'. Below that is a 'Remote:' label followed by a dropdown menu showing 'origin' with a downward arrow. To the right of the dropdown is a button labeled 'Add Remote...'. Below these is a checked checkbox followed by the text 'Sync branch with remote'. At the bottom right are two buttons: 'Create' and 'Cancel'.

New Branch

Branch Name:

Remote: origin ▼ Add Remote...

☒ Sync branch with remote

Create Cancel

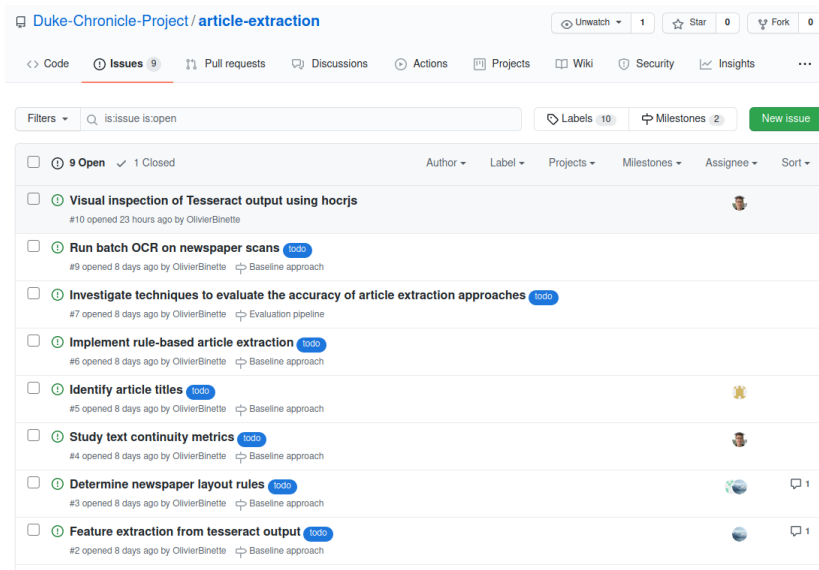
## Project management using GitHub

# Project management using GitHub

1. Break up your project into a set of milestones
2. Break up the milestones into tasks/todos.

# Project management using GitHub

## 3. List each task as an “Issue” on GitHub (the name “Issue” refers to tasks, todos, bugs, etc)



Duke-Chronicle-Project / **article-extraction**

Unwatch 1 Star 0 Fork 0

<> Code **Issues 9** Pull requests Discussions Actions Projects Wiki Security Insights ...

Filters is:issue is:open



Labels 10 Milestones 2 [New issue](#)

☐ **9 Open** ✓ 1 Closed Author Label Projects Milestones Assignee Sort

- ☐ **Visual inspection of Tesseract output using hocrjs**  
#10 opened 23 hours ago by OlivierBinette
- ☐ **Run batch OCR on newspaper scans** **todo**  
#9 opened 8 days ago by OlivierBinette [Baseline approach](#)
- ☐ **Investigate techniques to evaluate the accuracy of article extraction approaches** **todo**  
#7 opened 8 days ago by OlivierBinette [Evaluation pipeline](#)
- ☐ **Implement rule-based article extraction** **todo**  
#6 opened 8 days ago by OlivierBinette [Baseline approach](#)
- ☐ **Identify article titles** **todo**  
#5 opened 8 days ago by OlivierBinette [Baseline approach](#)
- ☐ **Study text continuity metrics** **todo**  
#4 opened 8 days ago by OlivierBinette [Baseline approach](#)
- ☐ **Determine newspaper layout rules** **todo**  
#3 opened 8 days ago by OlivierBinette [Baseline approach](#)
- ☐ **Feature extraction from tesseract output** **todo**  
#2 opened 8 days ago by OlivierBinette [Baseline approach](#)

# Project management using GitHub

4. Assign the issues to team members, document progress, and close the issue when you're done.

  **neel216** self-assigned this 8 days ago



**neel216** commented 2 days ago

Member

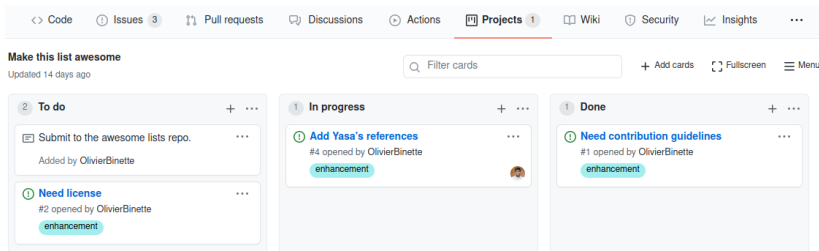


Uploaded an initial version of hOCR data extraction that for each word: parses the word, word confidence, line number, line height, paragraph number, column area number, page number, and bounding box for each feature and saves as a CSV. The CSV is a bit messy because it stores the relationship info of the word and all of its parent elements, so we may want to switch the file format to better represent relationships between elements, although CSV was the easiest to start with.

# Project management using GitHub

## 5. Want more?

- ▶ Use the “Projects” pane on GitHub to create one project Kanban board for each milestone.
- ▶ Track which tasks are being worked on by using the “In Progress” list.





## Summary

- ▶ RStudio's git pane is convenient. Make sure to set up SSH.
- ▶ Agree on a clear folder structure for your repo.
- ▶ Use GitHub Issues to define tasks and to document progress.