

Git & github

Getting started

- Login in <https://github.ugent.be/> with your ugent name and password
- Create a new repository “test_*your_name*”
 - Intialize with readme file.
- Go to R-studio and create a new project
 - Click version control
 - Click Git
 - Copy paste the URL from github
 - Put the repository where you want and create project

Click here

Create repository

The screenshot displays the GitHub Enterprise web interface. At the top, a dark navigation bar contains the GitHub logo, the word "Enterprise", a search bar labeled "Search GitHub", and links for "Pull requests", "Issues", and "Gist". On the right side of the bar are a plus icon and a user profile icon. A red arrow points from the text "Click here" to the GitHub logo.

Below the navigation bar, the main content area is divided into two columns. The left column shows a list of repository activity for the user "bksercu". Each entry includes a commit icon, a timestamp, the user's name, the branch pushed to, and the repository name. For example, "lhertzog pushed to lionel-dev at bksercu/heterogeneity-paper" 2 days ago. The right column shows "Repositories you contribute to" and "Your repositories". A red arrow points from the text "Create repository" to a green "New repository" button in the "Your repositories" section.

The "Your repositories" section lists 25 repositories, including "heterogeneity-paper", "lhertzog/coordination_files", "Geum_paper", "light_availability_paper", "Frass_paper", "dodkeuke/GontrodeExperiment", "dodkeuke/Breedingperformance_paper", "Stage_2017_terec", "analys_thesis_C-D", "R_simulations", "ivschoj/Fragmentation", "tree_phenology_herbivory", and "light_calculations".

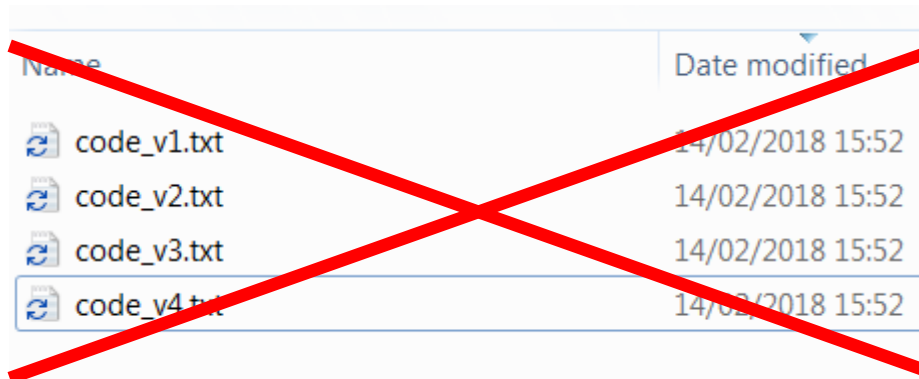
At the bottom left, the URL "https://github.ugent.be/new" is visible in the browser's address bar.





Part 1:

Git as a version control system

Git as version control system (VCS)

- Tracks changes to files.
- Enables to go back to previous versions
- Registers time and responsible of the changes
- Started in programming but can be used for several (simple) file types



Name	Date modified
 code_v1.txt	14/02/2018 15:52
 code_v2.txt	14/02/2018 15:52
 code_v3.txt	14/02/2018 15:52
 code_v4.txt	14/02/2018 15:52

Git as version control system (VCS)

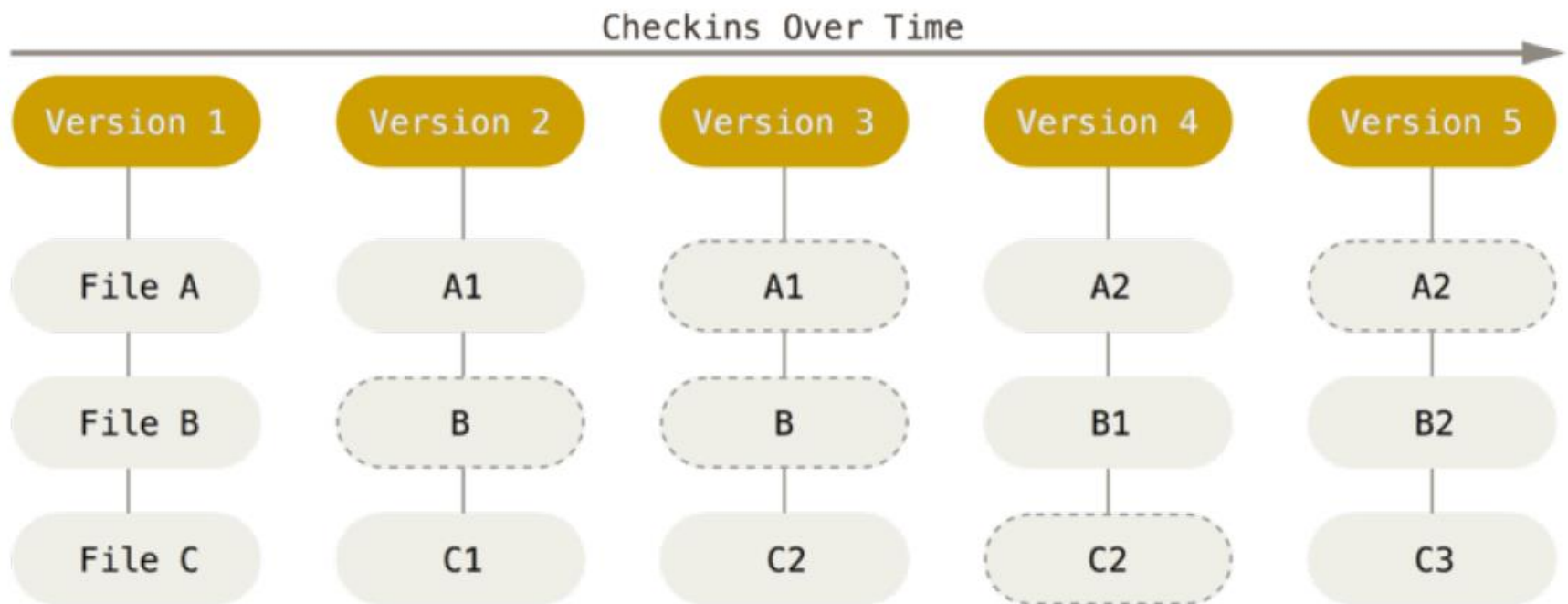


Figure 5. Storing data as snapshots of the project over time.

Git as version control

The screenshot shows the Git GUI interface for a repository named 'presentatie_github'. The top panel displays the commit history for the 'master' branch, listing 10 commits by Bram Sercu with timestamps ranging from 2016-05-20 16:35:48 to 2016-05-20 17:40:17. The left sidebar shows a list of commits with a yellow dot next to the current 'master' branch. The bottom panel shows a diff view for the file 'preparation_doc.docx', highlighting changes in green and red. The diff shows a new line added: '+ Switch to prepared folder and show masters history'.

presentatie_github: master - gitk

File Edit View Help

master extra explanation step

- change layout of title
- divide general - R scripts
- bring point forward
- new title
- first slide
- opbouw stap 1
- start presentatie

SHA1 ID: f7378a422dbda946b8625862d2bcbe99485fbb47 Row 1 / 8

Find commit containing:

Search

Diff Old version New version Lines of context: 3 Ignore space changes

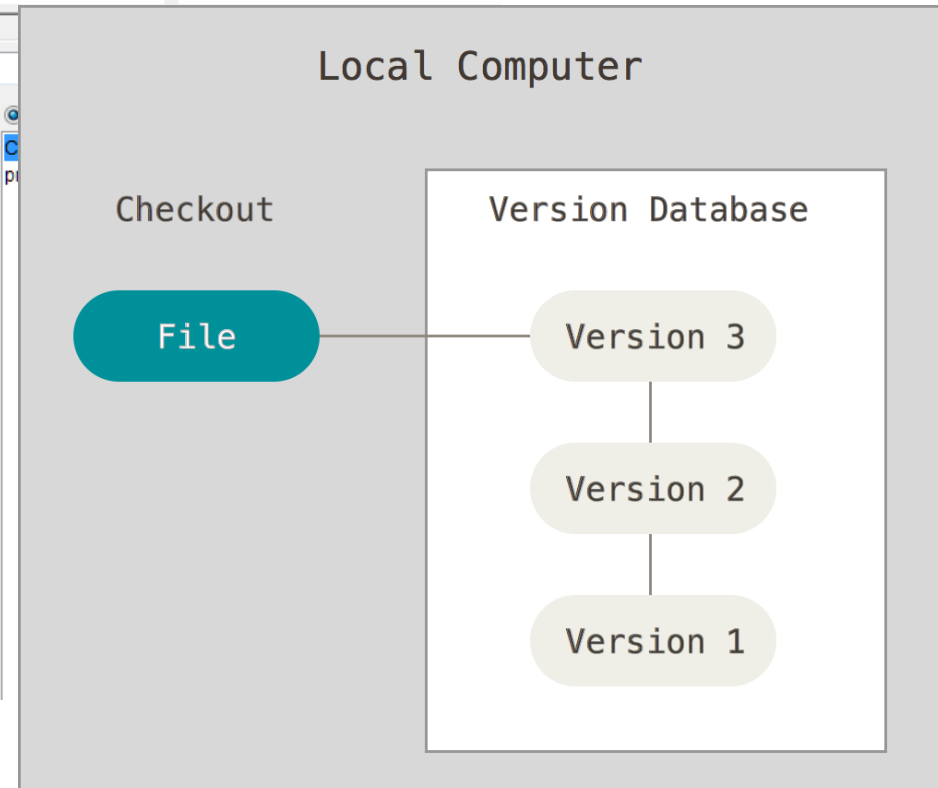
Author: Bram Sercu <bram.sercu@gmail.com> 2016-05-20 17:40:17
Committer: Bram Sercu <bram.sercu@gmail.com> 2016-05-20 17:40:17
Parent: 973a4869dd0782d348fdffb190a4dcd2a387c571 (change layout of title)
Branch: master
Follows:
Precedes:

extra explanation step

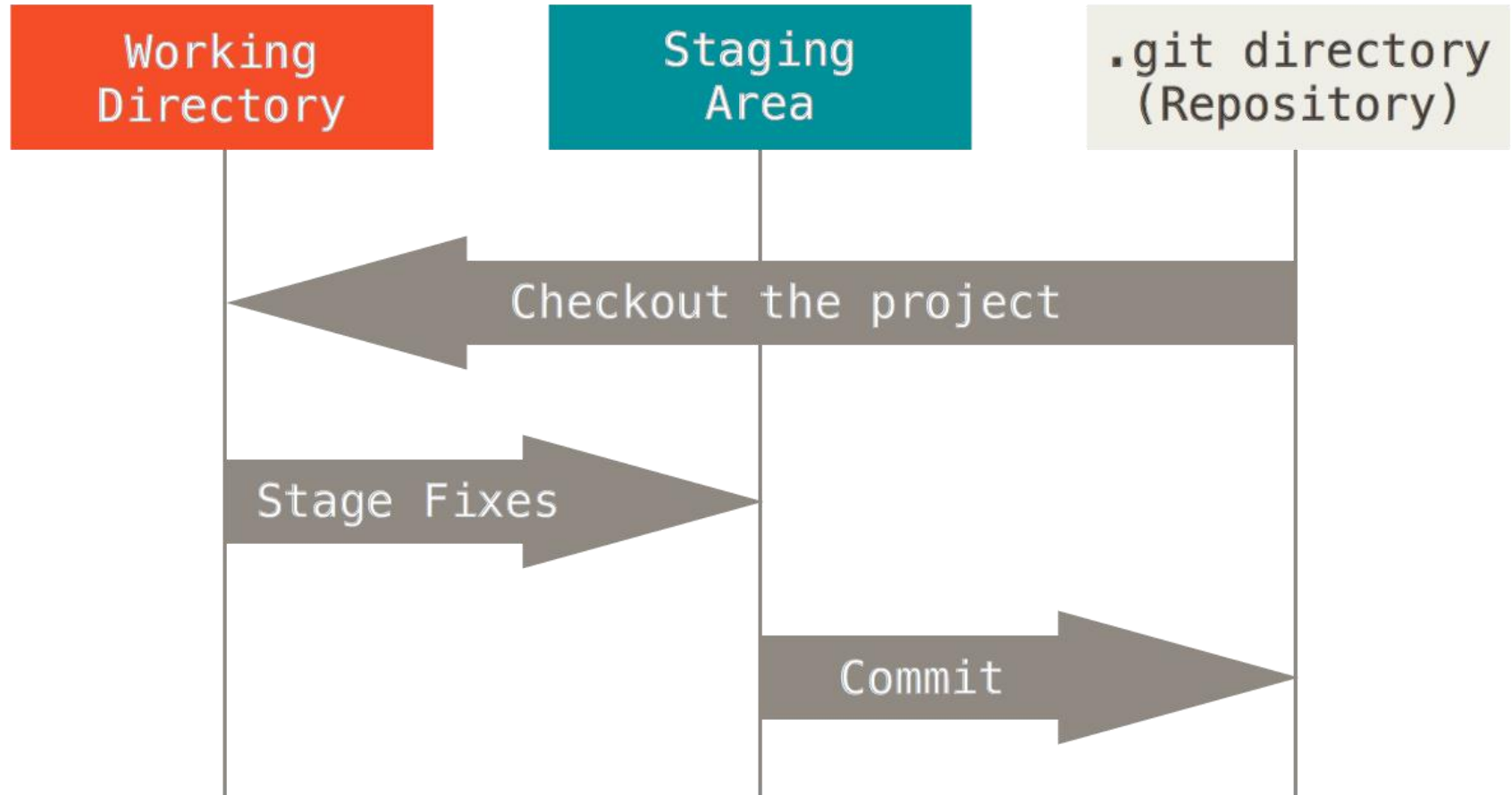
preparation_doc.docx

index 87caee7..c241034 100644

@@ -3,6 +3,7 @@ Git on local computer
Concept of version control
Bring normal folder under git control
Show example how to commit a change
+ Switch to prepared folder and show masters history
Version control most useful for R coding
Show example with R-project



Git basics: workflow



Git basics: staging area

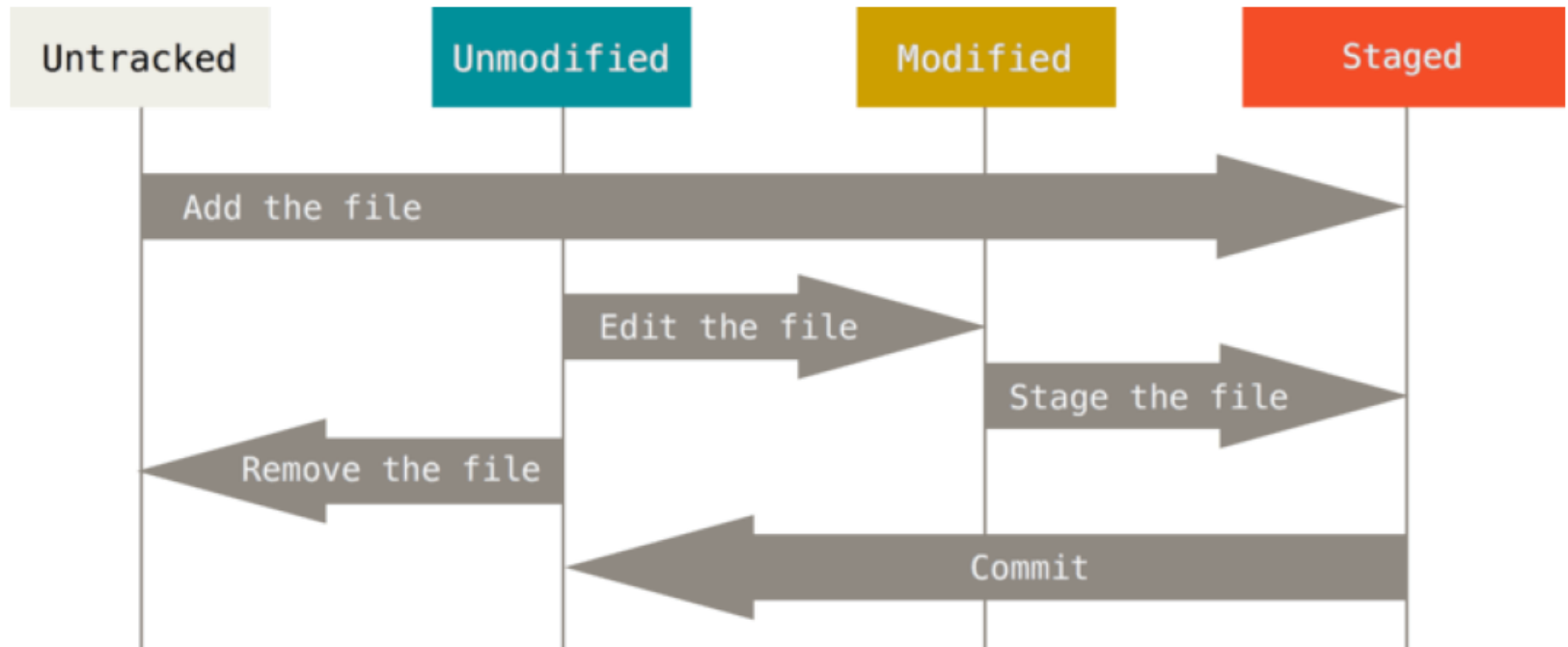
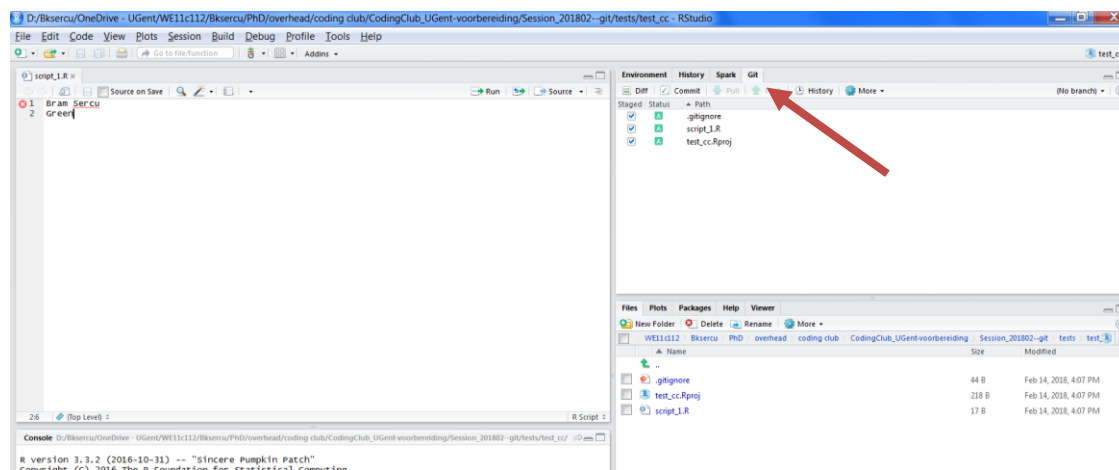


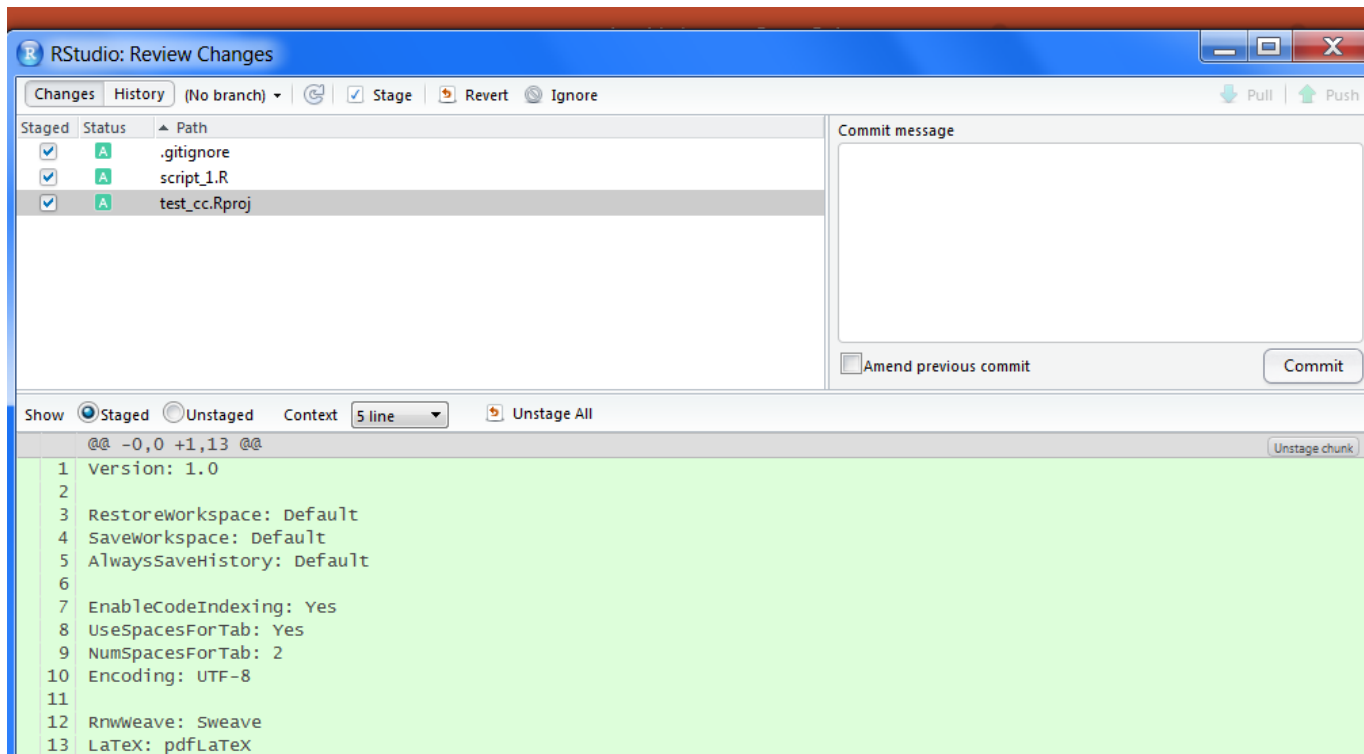
Figure 8. The lifecycle of the status of your files.

Exercise 1

- Create an R script with two lines of code and save as “my_favorite_things.R”
 - Line 1: your name
 - Line 2: your favorite color
- Go to the git tab in the upper right panel
 - Tick all boxes
 - Click commit
 - Press commit



What happens?



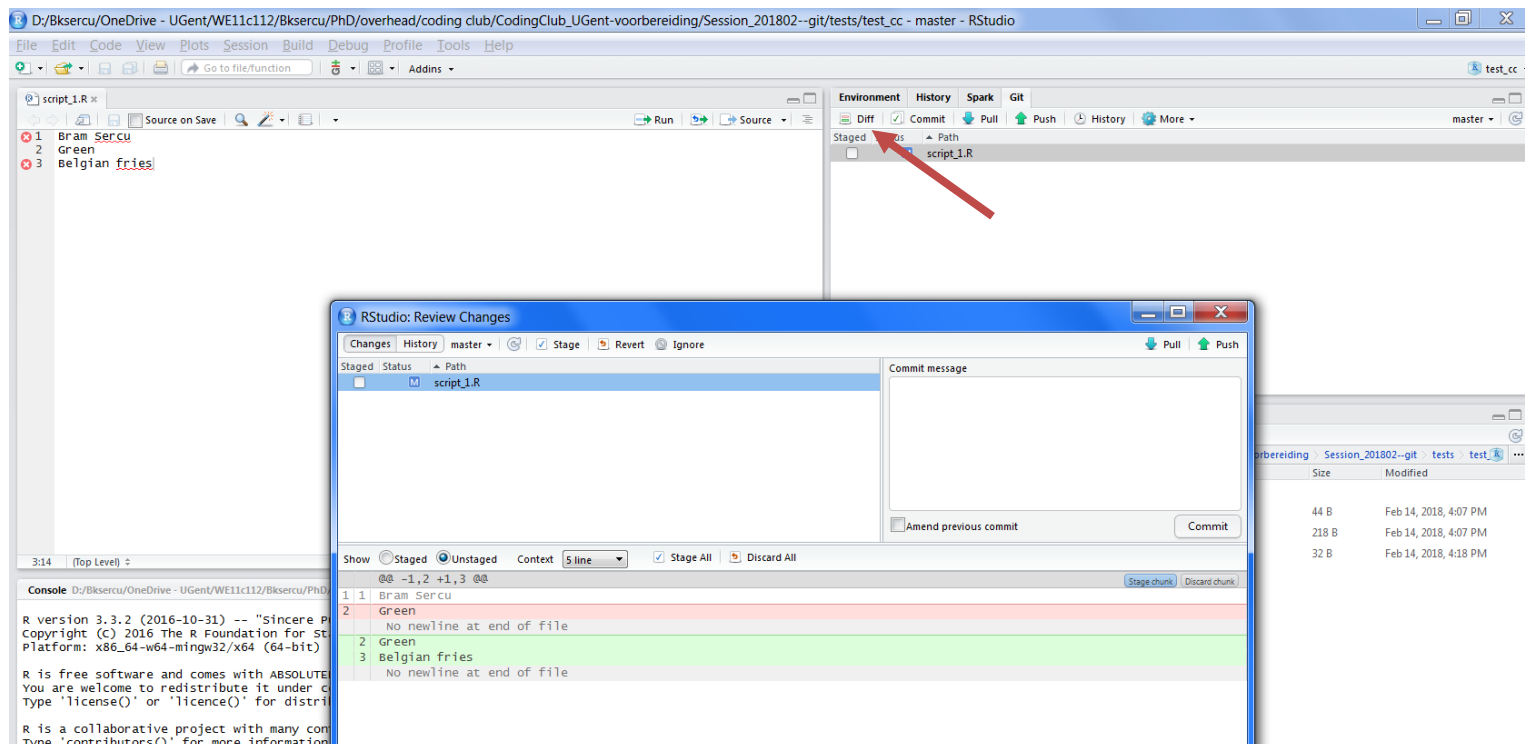
Exercise 1 continued

- It is not possible to commit a file without message
 - Write a commit message eg. “first commit”
 - Press again on commit

Hurray your first commit is a fact!

Exercise 1 continued

- Go back to the R-script, add your favorite meal in line 3 and save
- Select the file in the git tab and click 'diff'



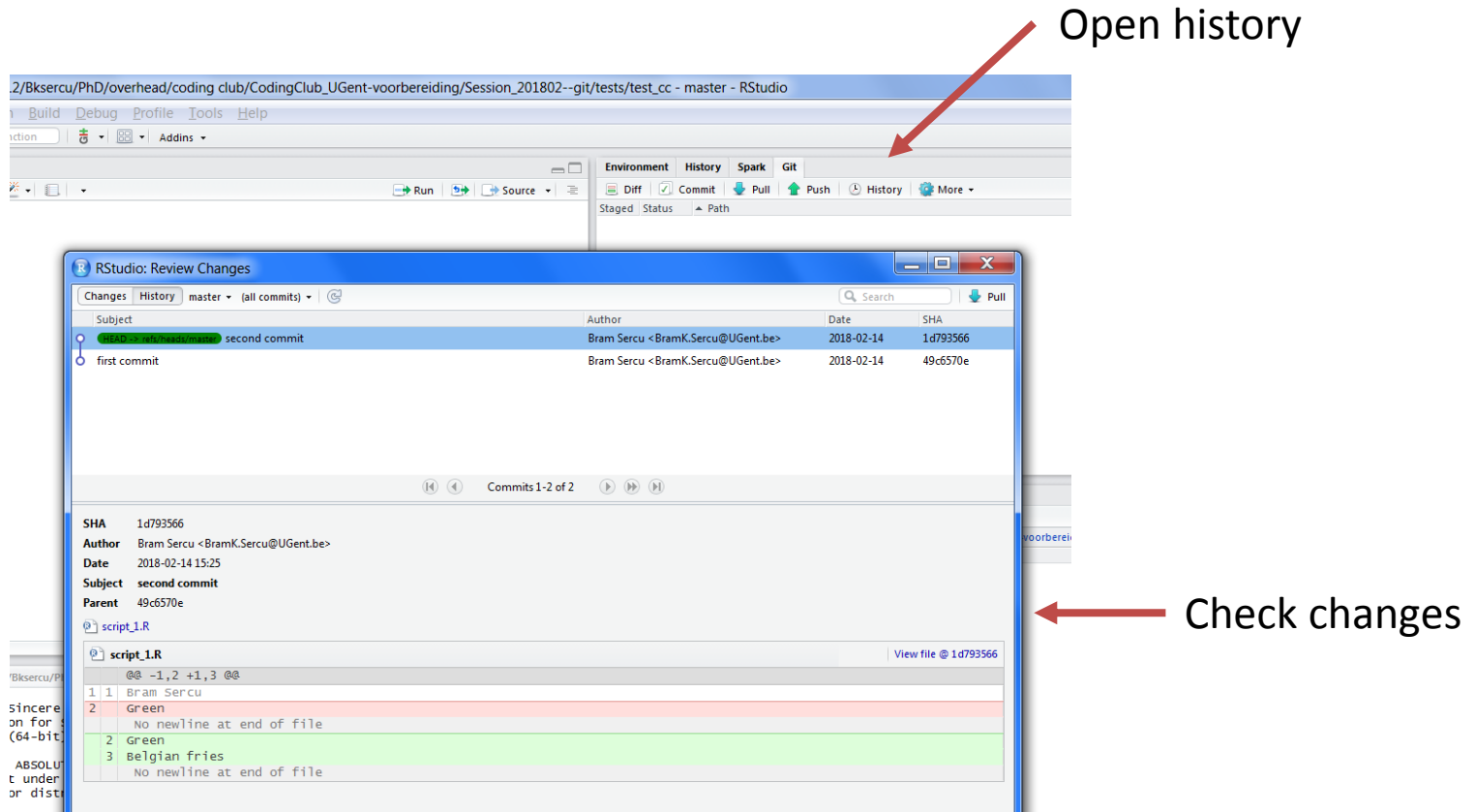
Exercise 1 continued

- Here you can see the differences between your new script and the last commit
- Commit the new changes
- Add your favorite species group in line 4 and commit
- Change your favorite colour to black and commit

Exercise 1 continued

- Click History in the commit tab and inspect the changes in every commit

Open history



The screenshot shows the RStudio Git interface. The top panel displays the 'History' tab, which lists two commits: 'second commit' (SHA: 1d793566) and 'first commit' (SHA: 49c6570e). A red arrow points to the 'History' tab in the top panel. The bottom panel shows the 'Review Changes' window for the 'second commit'. It displays the commit details (SHA, Author, Date, Subject, Parent) and a diff view of the file 'script_1.R'. The diff shows changes to the file content, with a red arrow pointing to the 'Check changes' button in the bottom right corner.

Subject	Author	Date	SHA
second commit	Bram Sercu <BramK.Sercu@UGent.be>	2018-02-14	1d793566
first commit	Bram Sercu <BramK.Sercu@UGent.be>	2018-02-14	49c6570e

Commits 1-2 of 2

SHA 1d793566
Author Bram Sercu <BramK.Sercu@UGent.be>
Date 2018-02-14 15:25
Subject second commit
Parent 49c6570e

script_1.R

@@ -1,2 +1,3 @@
1 | Bram Sercu
2 | Green
 | No newline at end of file
2 | Green
3 | Belgian fries
 | No newline at end of file

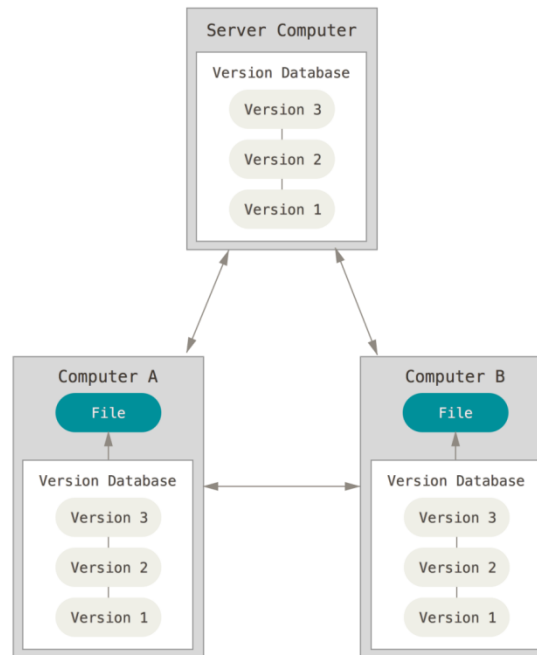
View file @ 1d793566

Check changes

Part 2: cooperating

Sharing and cooperating

- Distributed VCS
- Full mirror of the repository local.



Exercise 2

- Check your online repository
- Go to R-studio and click push in the git tab
- Check your online repository again

You succesfully created a copy of your files online!

Exercise 2

Form pairs of people for the next exercise

- Create a new file online: “my_partners_favorite_things.R”
- Type the name of your partner (line 1) and guess his favorite
 - Line2: colour
 - Line3: dish
 - Line4: species group

The screenshot shows a GitHub repository interface. At the top, it displays repository statistics: 8 commits, 1 branch, 0 releases, and 1 contributor. Below this is a navigation bar with buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A red arrow points to the 'Create new file' button. The main content area shows a list of files and their commit history:

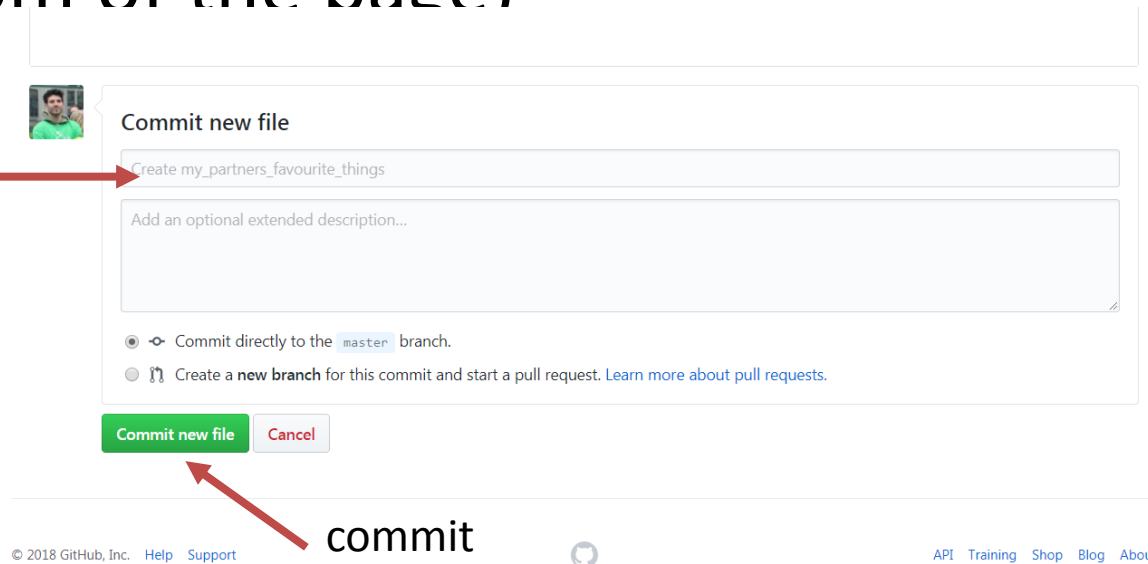
File	Commit	Time
.gitignore	first commit	5 days ago
my_partners_favorite_things.R	create new file	5 days ago
script_1.R	conflict resolved	3 days ago
test_cc.Rproj	first commit	5 days ago

At the bottom, there is a prompt to 'Add a README' to help people understand the project.

Exercise 2

- Commit the file (see commit section at the bottom of the page)

Write
message



The screenshot shows the GitHub 'Commit new file' interface. A red arrow points from the text 'Write message' to the commit message input field, which contains the text 'Create my_partners_favourite_things'. Another red arrow points from the word 'commit' at the bottom of the page to the 'Commit new file' button. The interface includes a profile picture, a title 'Commit new file', a commit message input field, an optional extended description field, and two radio button options: 'Commit directly to the master branch.' and 'Create a new branch for this commit and start a pull request.' The footer contains copyright information, links for Help and Support, the GitHub logo, and links for API, Training, Shop, Blog, and About.

Commit new file


Create my_partners_favourite_things

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit new file Cancel

© 2018 GitHub, Inc. [Help](#) [Support](#)  [API](#) [Training](#) [Shop](#) [Blog](#) [About](#)

- To get the changes locally: Go to R-studio and click 'pull' in the git tab

Summary: exercise 2

- The push and pull button enable you to
 - Push local changes to the online repository and make them available online
 - Pull changes that are online but not on your local computer to update your local folder.

Exercise 3

collaborating

Go to github (online)

- Go to settings – collaborators
- Search the username of your partner and add as collaborator (and vice-versa)
- Now go to your main github page and copy the link of the repository of your partner
- Create a new R-project with your partner's link (see slide 2 & 3)

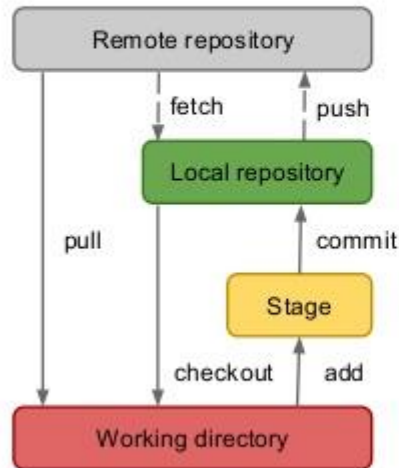
Exercise 3

collaborating

- In the project of your partner:
 - Open `my_partners_favorite_things.R`
 - Change the preferences to your correct ones, save, commit and push
- Do you both see the changes online?
- Pull the changes made to your own project

Summary exercise 3: workflow local - remote

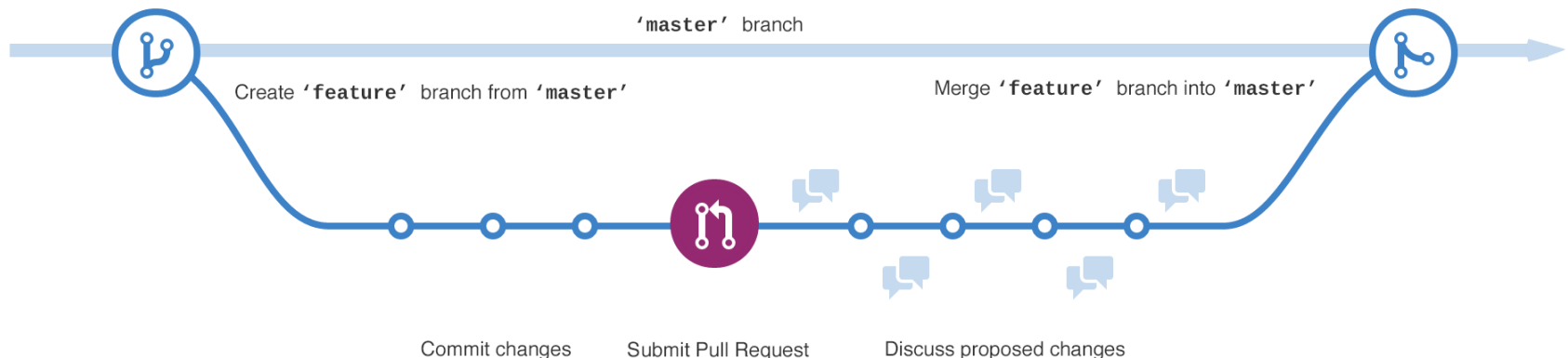
Understanding of workflow



- Obtain a repository
 - `git init` or `git clone`
- Make some changes
- Stage your changes
 - `git add`
- Commit changes to the local repository
 - `git commit -m "My message"`
- Push changes to remote
 - `git push remotename remotebranch`

Branching

- A nicer way of cooperating is to use branches instead of working directly in the files of the other person.
 - You can work in a branch without affecting the workflow of your partner (or yourself)
- A pull request proposes to the owner of the master to accept the changes you made in the branch



Exercise 4

branching

- Create a branch in your partners repository
- pull changes to the local version of you partners repo in R-studio
- Ensure that you are working on the newly created branch
 - The branch name is on the upper right of the git panel.
- Create a new file and adapt something in an old file in this branch in R-studio
- Commit and push
- Go online and create a pull request to merge your branch with the master
- Accept the pull request of your partner in your own repository

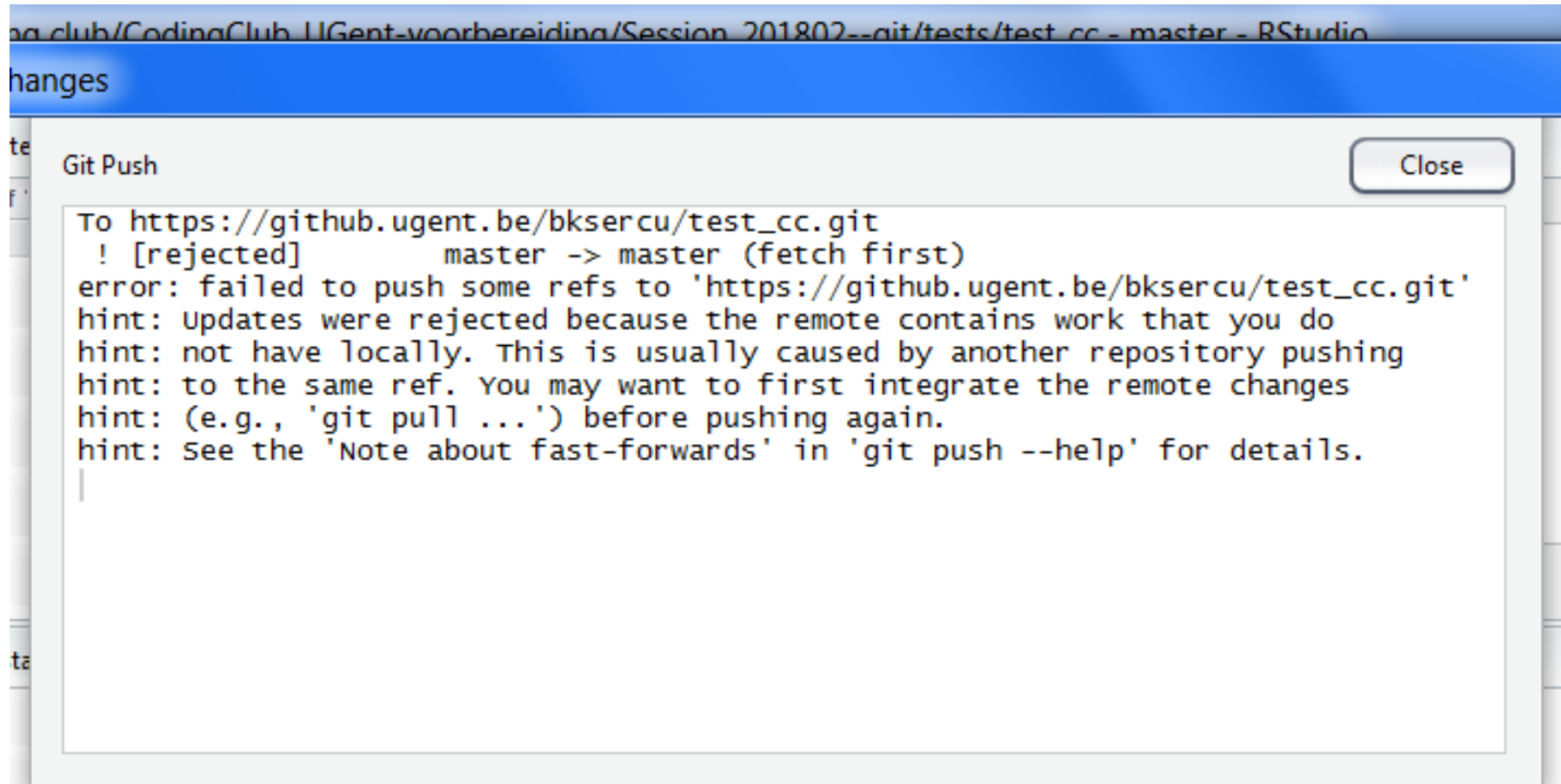
Exercise 5:

Solving merge conflicts

- We will work in the master branches so ensure you work in the correct branch
- In your own project
 - Add your hobby's online 5
 - Commit and push
- In your partners project (without pulling first)
 - Add "I love working in R" online 5
 - Commit and push

Exercise 5:

Solving merge conflicts



The screenshot shows a dialog box titled "Git Push" with a "Close" button in the top right corner. The dialog contains the following text:

```
To https://github.ugent.be/bksercu/test_cc.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.ugent.be/bksercu/test_cc.git'
hint: updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

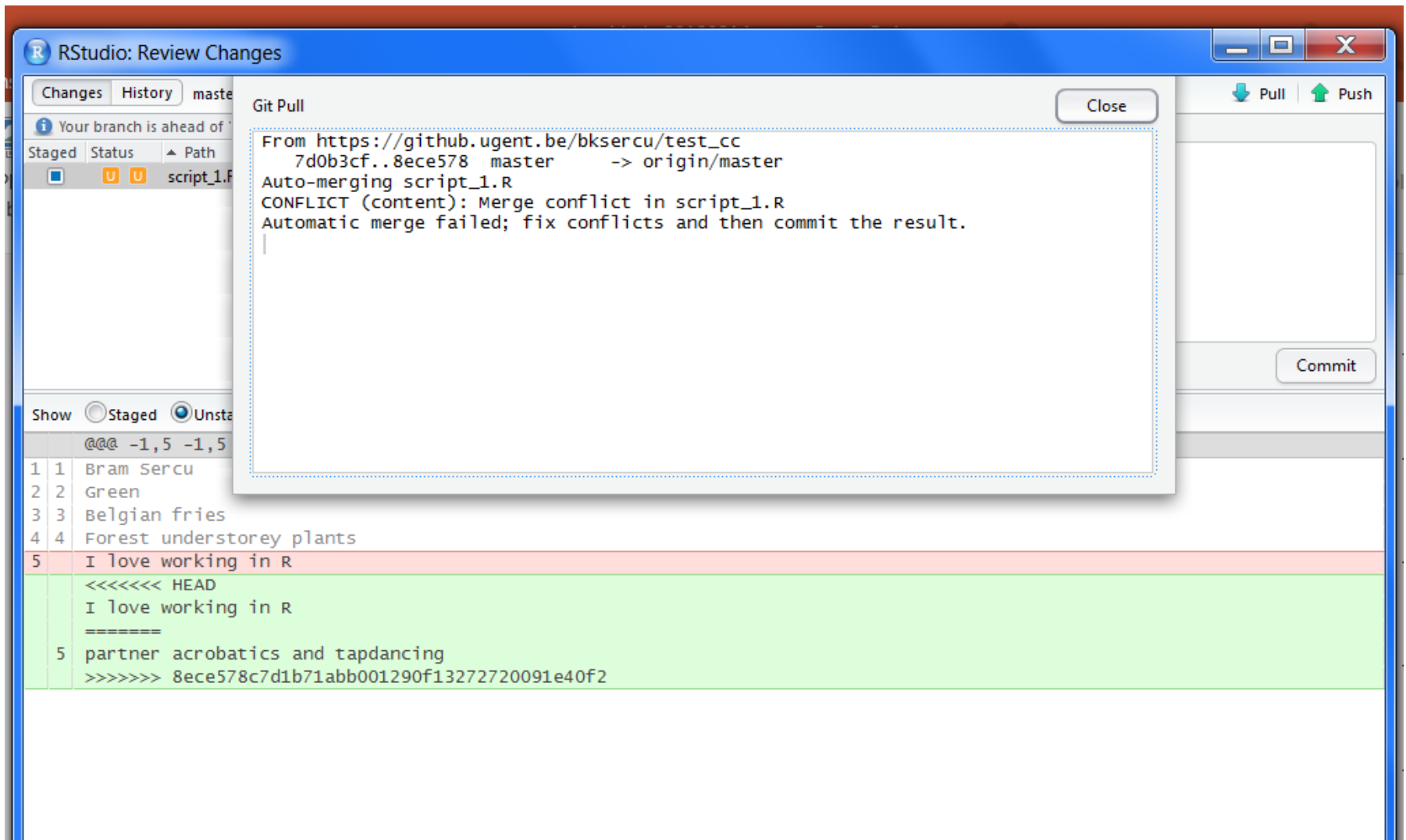
Exercise 5:

Solving merge conflicts

- Auwch you can't push because your partner already changed something in this file
- We will resolve this
 - Click 'pull'
 - You will get a warning that there is a conflict
 - Git tells you to 'fix conflicts and then commit the result'
 - Open the file with the conflict in R-studio (or with a text editor)

Exercise 5:

Solving merge conflicts



The screenshot shows the RStudio interface with a 'Git Pull' dialog box open. The dialog box contains the following text:

```
From https://github.ugent.be/bksercu/test_cc
7d0b3cf..8ece578 master -> origin/master
Auto-merging script_1.R
CONFLICT (content): Merge conflict in script_1.R
Automatic merge failed; fix conflicts and then commit the result.
```

The background shows the RStudio 'Review Changes' window. The 'Changes' tab is active, showing a list of files with their status. The file 'script_1.R' is highlighted in red, indicating a conflict. Below the file list, the diff for 'script_1.R' is shown, with the conflict area highlighted in green:

```
@@@ -1,5 -1,5
1 1 Bram Sercu
2 2 Green
3 3 Belgian fries
4 4 Forest understorey plants
5 I love working in R
<<<<<< HEAD
I love working in R
=====
5 partner acrobatics and tapdancing
>>>>>> 8ece578c7d1b71abb001290f13272720091e40f2
```

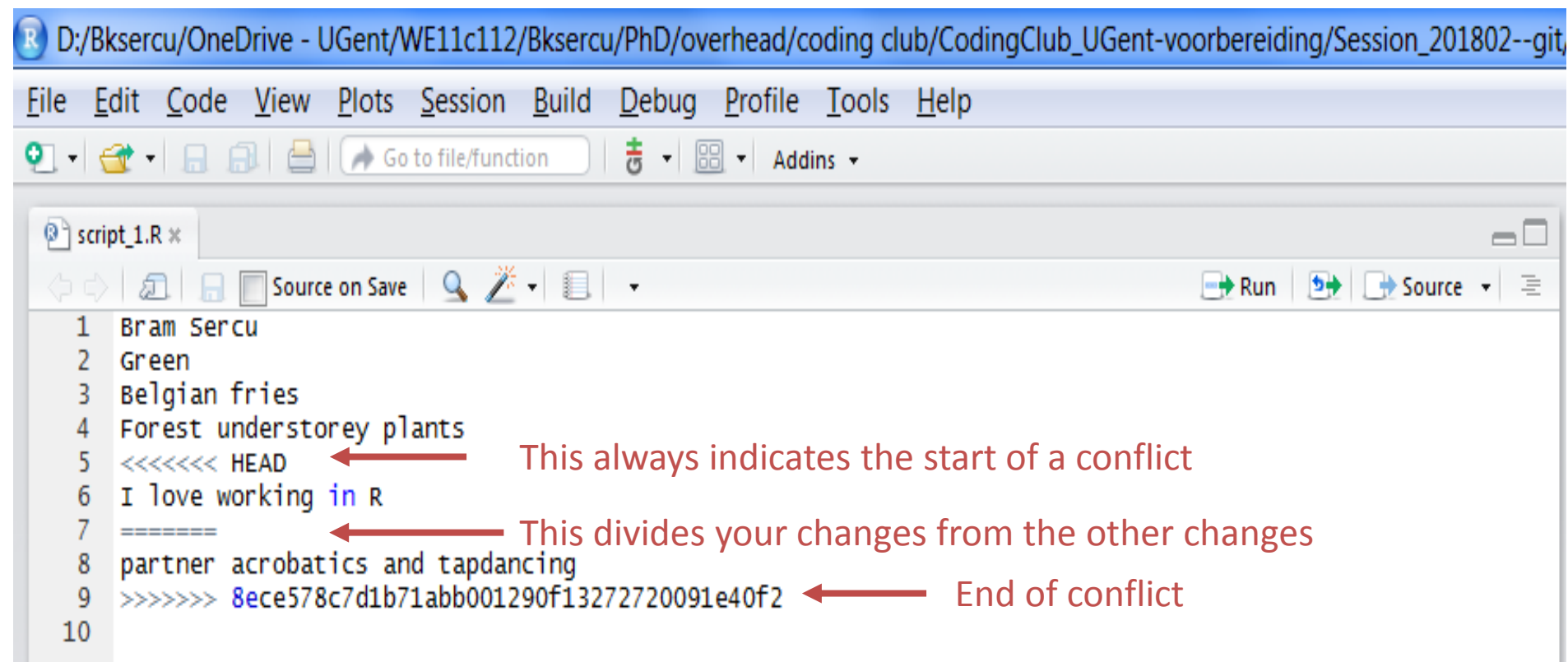
Exercise 5:

Solving merge conflicts

- Find the conflict
 - ‘<<<<<<< ‘ indicates the start of the conflict
 - ‘=====’ divides the original text from your adaptation
 - ‘>>>>>>>’ indicates the end of the conflict
- Manually resolve the conflict
 - Delete what you don’t want or adapt if you want to keep both
 - Make sure you delete the conflict markers
 - Save the file
- Go to R-studio commit and push

Exercise 5:

Solving merge conflicts



The screenshot shows the RStudio interface with a file named 'script_1.R' open. The editor displays the following text:

```
1 Bram Sercu
2 Green
3 Belgian fries
4 Forest understorey plants
5 <<<<<< HEAD
6 I love working in R
7 =====
8 partner acrobatics and tapdancing
9 >>>>>> 8ece578c7d1b71abb001290f13272720091e40f2
10
```

Red arrows point to the conflict markers with the following annotations:

- An arrow points to the line `<<<<<< HEAD` with the text: "This always indicates the start of a conflict".
- An arrow points to the line `=====` with the text: "This divides your changes from the other changes".
- An arrow points to the line `>>>>>> 8ece578c7d1b71abb001290f13272720091e40f2` with the text: "End of conflict".

Exercise 5:

Solving merge conflicts

Resolved conflict:

```
D:/Bksercu/OneDrive - UGent/WE11c112/Bksercu/PhD/overhead/coding club/CodingC  
File Edit Code View Plots Session Build Debug Profile Tools Help  
Go to file/function  
script_1.R *  
Source on Save  
1 Bram Sercu  
2 Green  
3 Belgian fries  
4 Forest understorey plants  
5  
6 I love partner acrobatics, tapdancing and working in R  
7
```

Summary: branching and conflicts

- Branches are used to work without affecting the main workflow
 - You can do this in your own repo's
 - Or you can do it when cooperating
- It enables you to work without creating conflicts because people are working in the same repository.
- When conflicts arise they should be solved manually in the files

Ignoring unsuited files

What should be put on github

- Simple file formats
 - Txt
 - Csv
 - R files
 - Markdown
- Not suitable
 - Docx
 - Excel
 - Images
 - Large files

Ignore unsuited files with .gitignore

- Sometimes you want git to ignore changes in certain files
- You can specify this by adding files, folders or filetypes to the .gitignore file.
- More info and complex specifications
 - <https://git-scm.com/docs/gitignore>

Ignore unsuited files with .gitignore

```
# ignore all .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
!lib.a

# only ignore the TODO file in the current directory, not subdir/TODD
/TODD

# ignore all files in the build/ directory
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

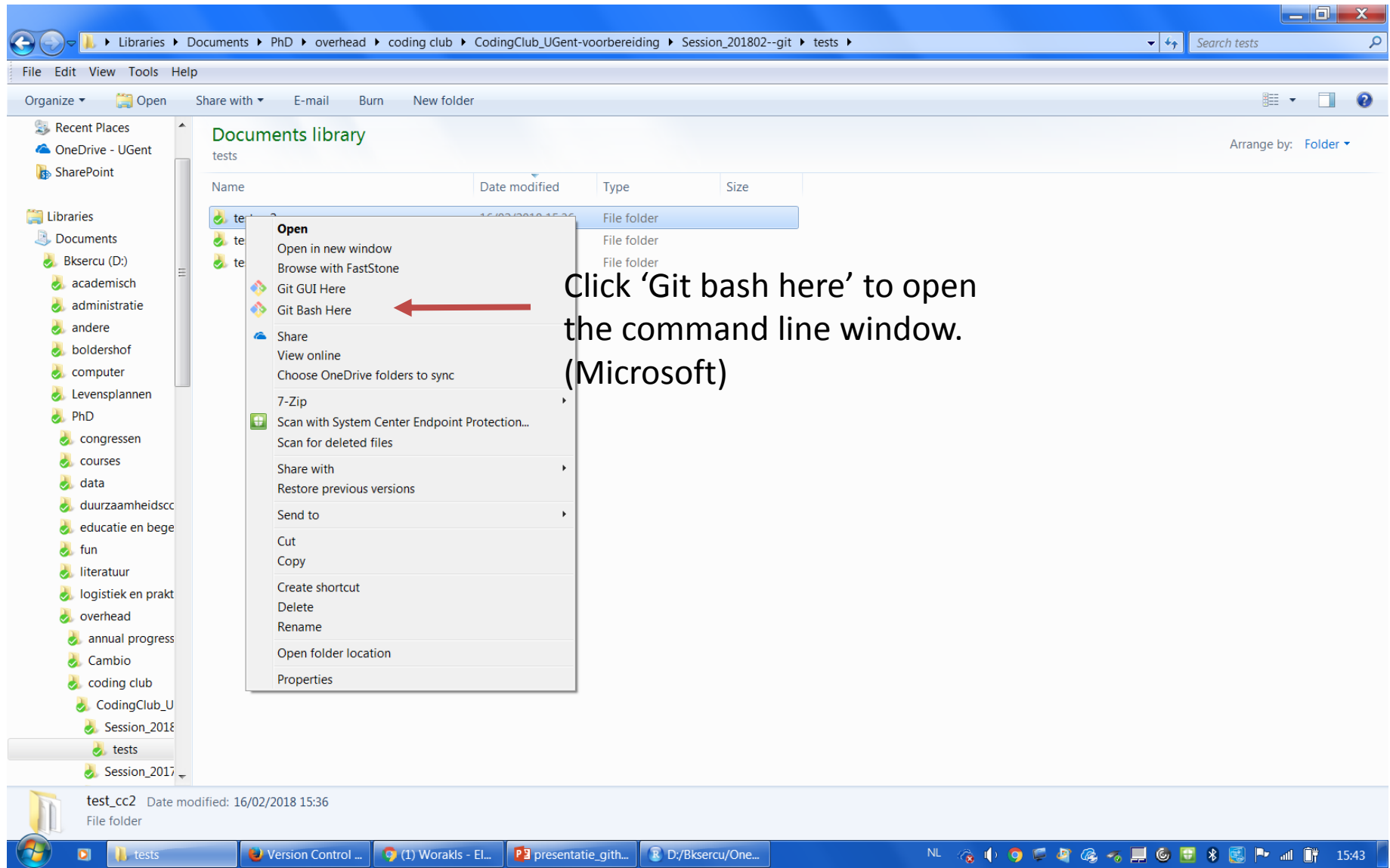
# ignore all .pdf files in the doc/ directory and any of its subdirectories
doc/**/*.*pdf
```

Using the command line

Command line

- R studio has a nice user interface for the most common actions in Git.
- If you are not working in R studio you could use another git interface (see downloads) or work from the command line
- Complex errors might force you to use the command line
- The 'progit book' gives a clear overview of the different commands

Opening the command line window



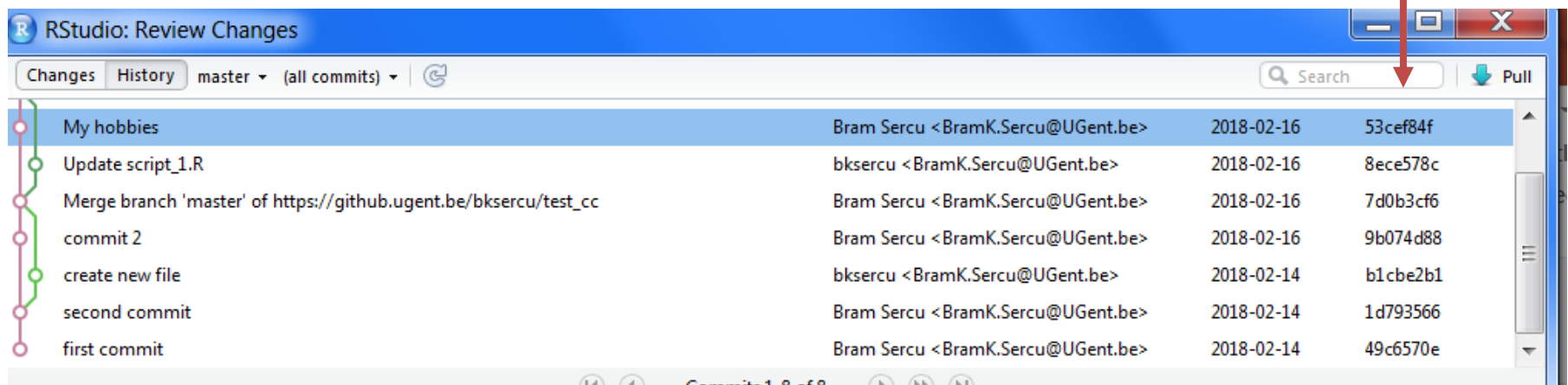
One example for the command line: going back to a previous state

- Open the command line window in your project
- Check out your working directory type:
 - \$ Git status
- Check whether you are on the correct branch and whether everything is staged and committed you want to keep

going back to a previous state

- Every commit has a code which can be used to return to that specific commit

Commit codes



going back to a previous state

- If you want to temporarily go back to it, fool around, then come back to where you are, all you have to do is check out the desired commit:
 - `$ git checkout 0ad5a7a6`
- To go back to where you were, just check out the branch you were on again. (If you've made changes, as always when switching branches, you'll have to deal with them as appropriate. You could reset to throw them away; you could stash, checkout, stash pop to take them with you; you could commit them to a branch there if you want a branch there.)

going back to a previous state

- Since "branches" are so cheap and easy in Git, we can easily create a new branch which starts at that old revision:
 - `$ git checkout -b old-project-state 0ad5a7a6`
- Normally, the *checkout* command is used to just *switch* branches. However, providing the `-b` parameter, you can also let it **create** a new branch (named "old-project-state" in this example). If you *don't* want it to start at the current HEAD revision, you also need to provide a commit hash - the old project revision we want to restore.

going back to a previous state

- To delete commits and restore an old version you can use the "reset" command:
 - `$ git reset --hard 0ad5a7a6`
- This removes all commits after the specified commit

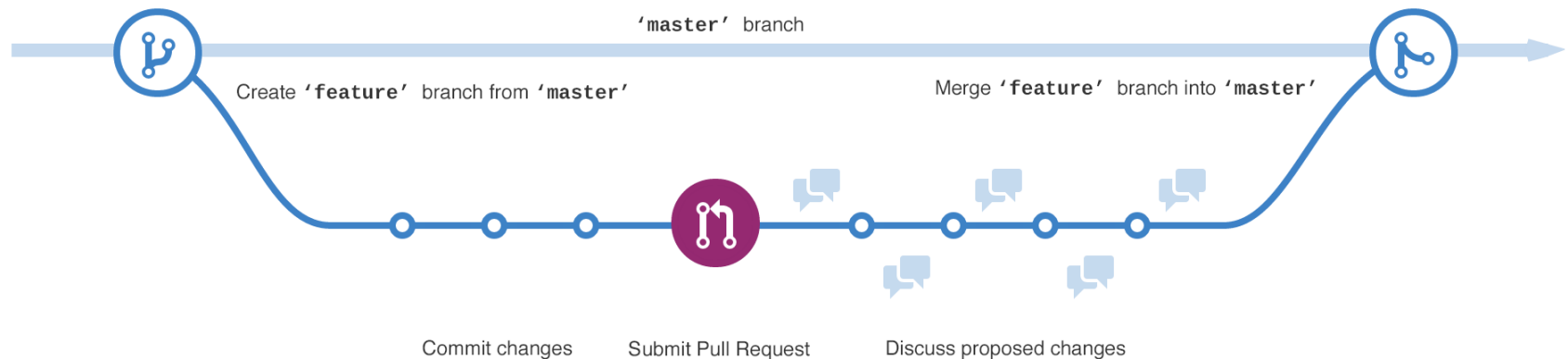
Forking

Forking

- Git is an open source community. There is a lot of code online and Github enables and promotes free access and collaboration.
- Forking allows you to
 - Collaborate with several people in the same way as branching does
 - copy repositories and mess around

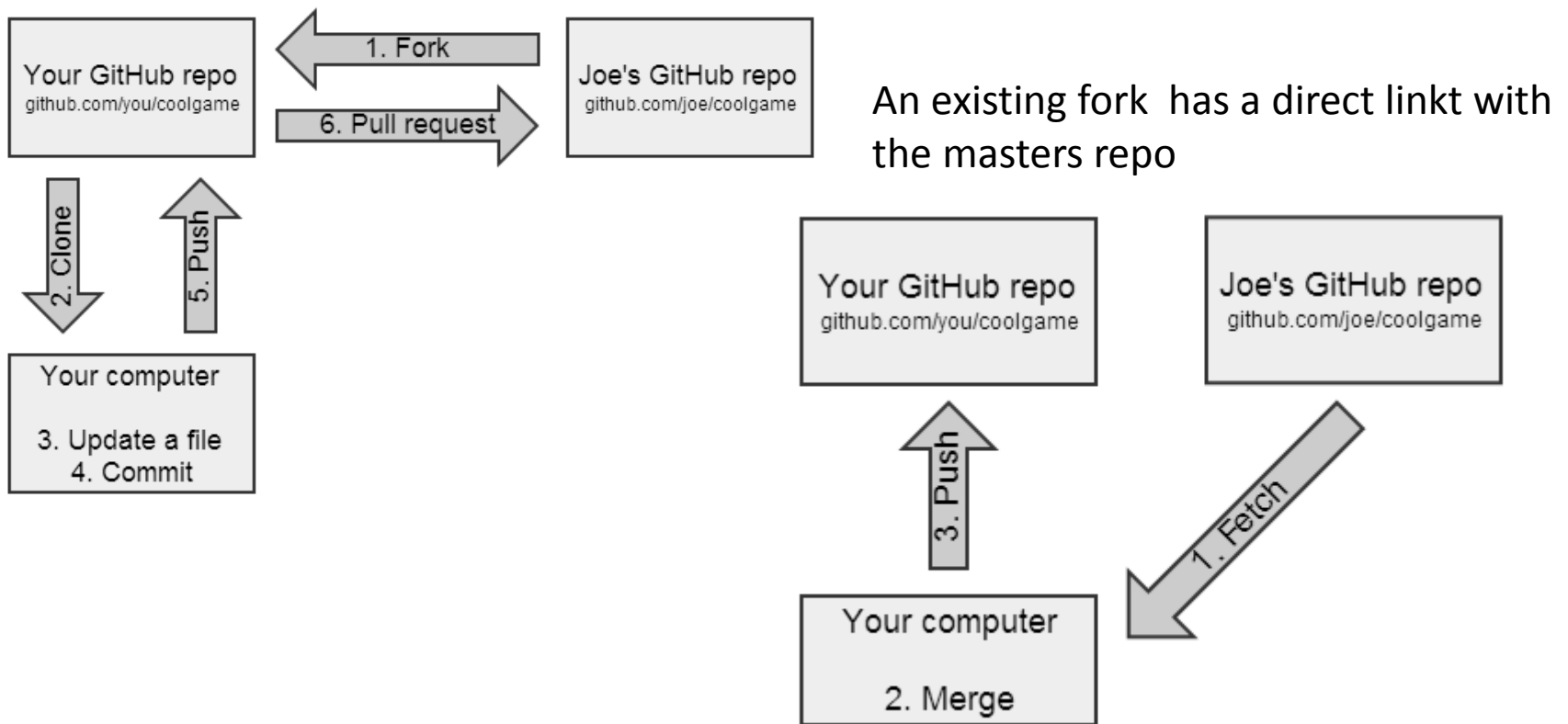
Active use collaborating

- Forking a repository creates a copy of the repository.
 - You can work in this forked branch without affecting the original code
- A pull request proposes to the owner of the master to accept the changes you made in the fork



Collaborating Forking

Create an online fork and clone the forked repo to your computer

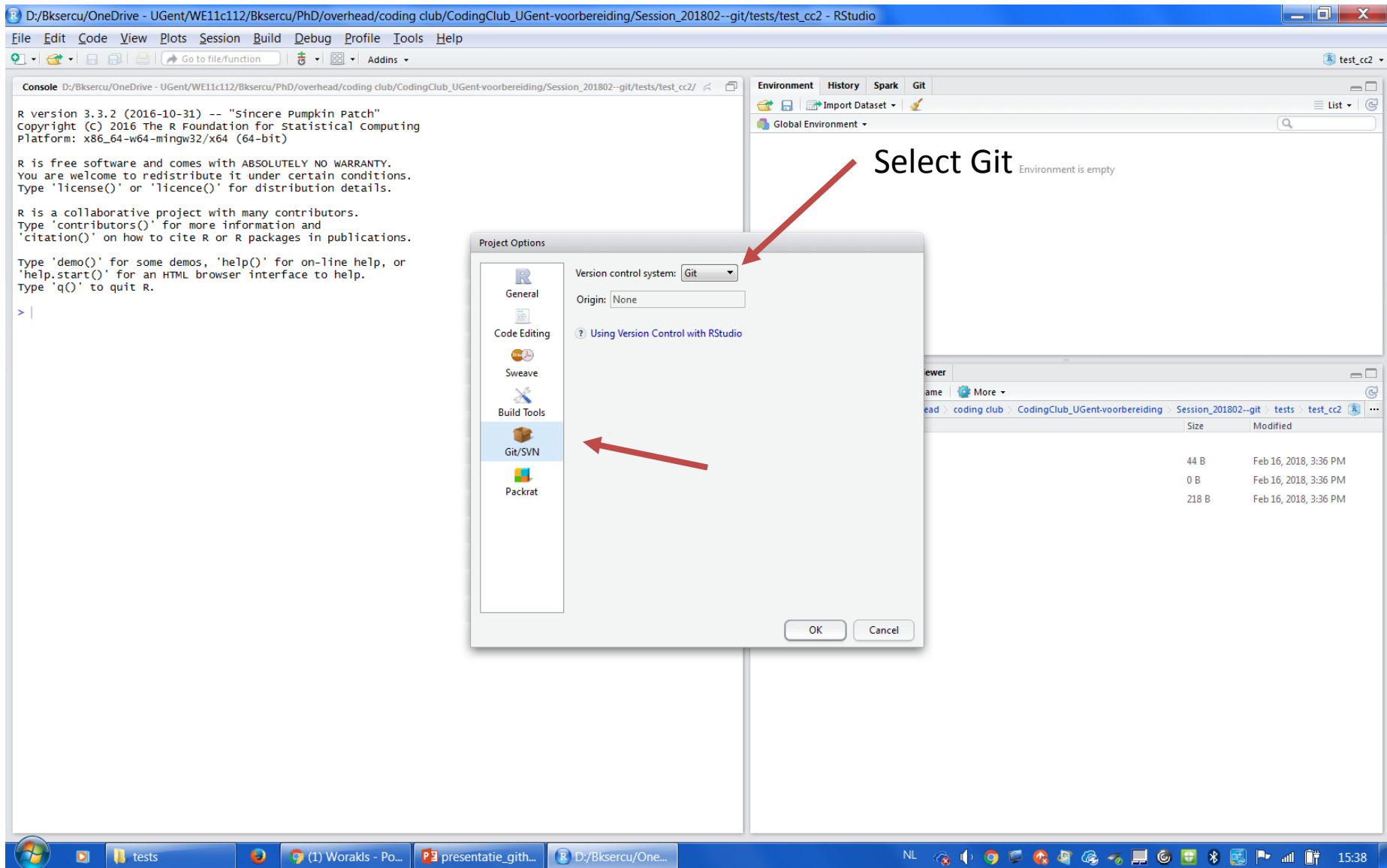


source: <http://www.dataschool.io/simple-guide-to-forks-in-github-and-git/>

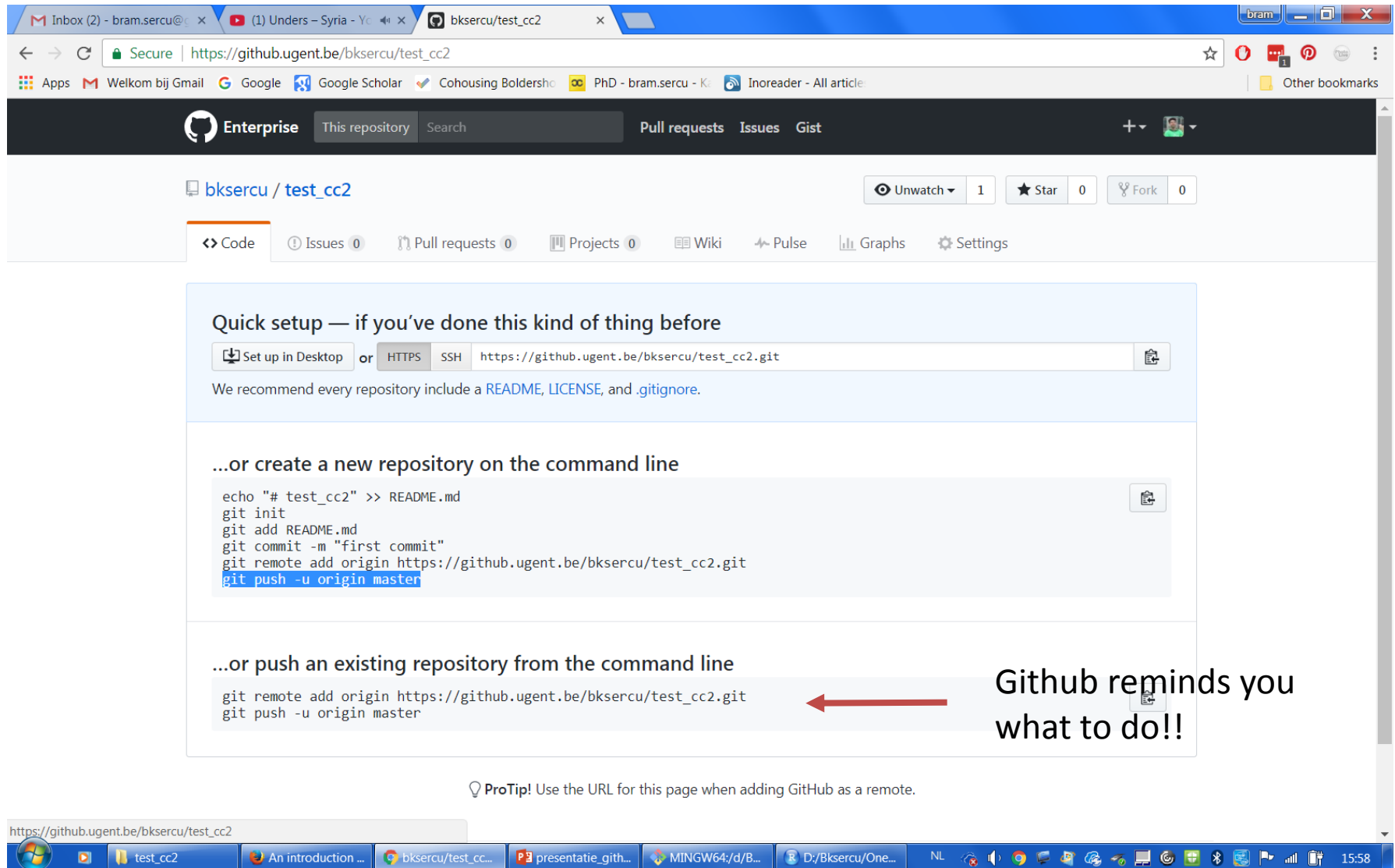
Bringing an existing R-project
under git control

Step1: Bringing an existing R-project under git control

- Open the project in R-studio
- Go to tools – Version control
- Click “Git/SVN” in the left menu
- Select ‘Git’ in the drop down menu



Step2: create an online repository



The screenshot shows a web browser window with the GitHub repository page for 'bksercu/test_cc2'. The browser's address bar shows the URL 'https://github.ugent.be/bksercu/test_cc2'. The page header includes the GitHub logo, the repository name, and navigation links like 'Pull requests', 'Issues', and 'Gist'. Below the header, there are statistics for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). The main content area has tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The 'Code' tab is selected, displaying a 'Quick setup' section. This section offers two options: 'Set up in Desktop' or 'HTTPS'. The 'HTTPS' option shows the repository URL 'https://github.ugent.be/bksercu/test_cc2.git'. Below this, it recommends including a README, LICENSE, and .gitignore. The next section, '...or create a new repository on the command line', provides a series of terminal commands: 'echo "# test_cc2" >> README.md', 'git init', 'git add README.md', 'git commit -m "first commit"', 'git remote add origin https://github.ugent.be/bksercu/test_cc2.git', and 'git push -u origin master'. The final section, '...or push an existing repository from the command line', shows the commands 'git remote add origin https://github.ugent.be/bksercu/test_cc2.git' and 'git push -u origin master'. A red arrow points from the text 'Github reminds you what to do!!' to the 'git push' command in the final section. At the bottom, a 'ProTip!' suggests using the URL for adding GitHub as a remote. The browser's taskbar at the very bottom shows several open applications, including a terminal window.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.ugent.be/bksercu/test_cc2.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# test_cc2" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.ugent.be/bksercu/test_cc2.git
git push -u origin master
```

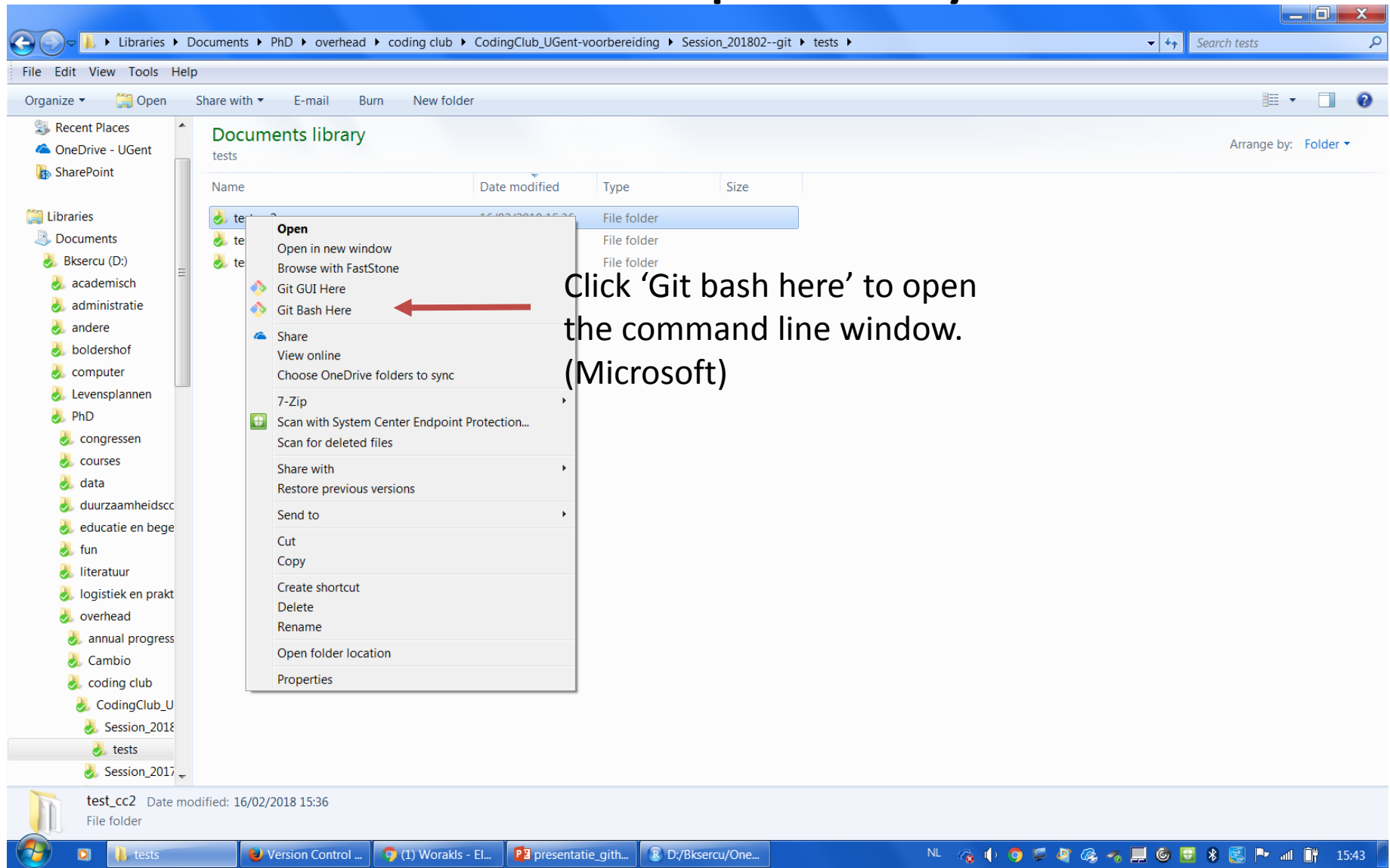
...or push an existing repository from the command line

```
git remote add origin https://github.ugent.be/bksercu/test_cc2.git
git push -u origin master
```

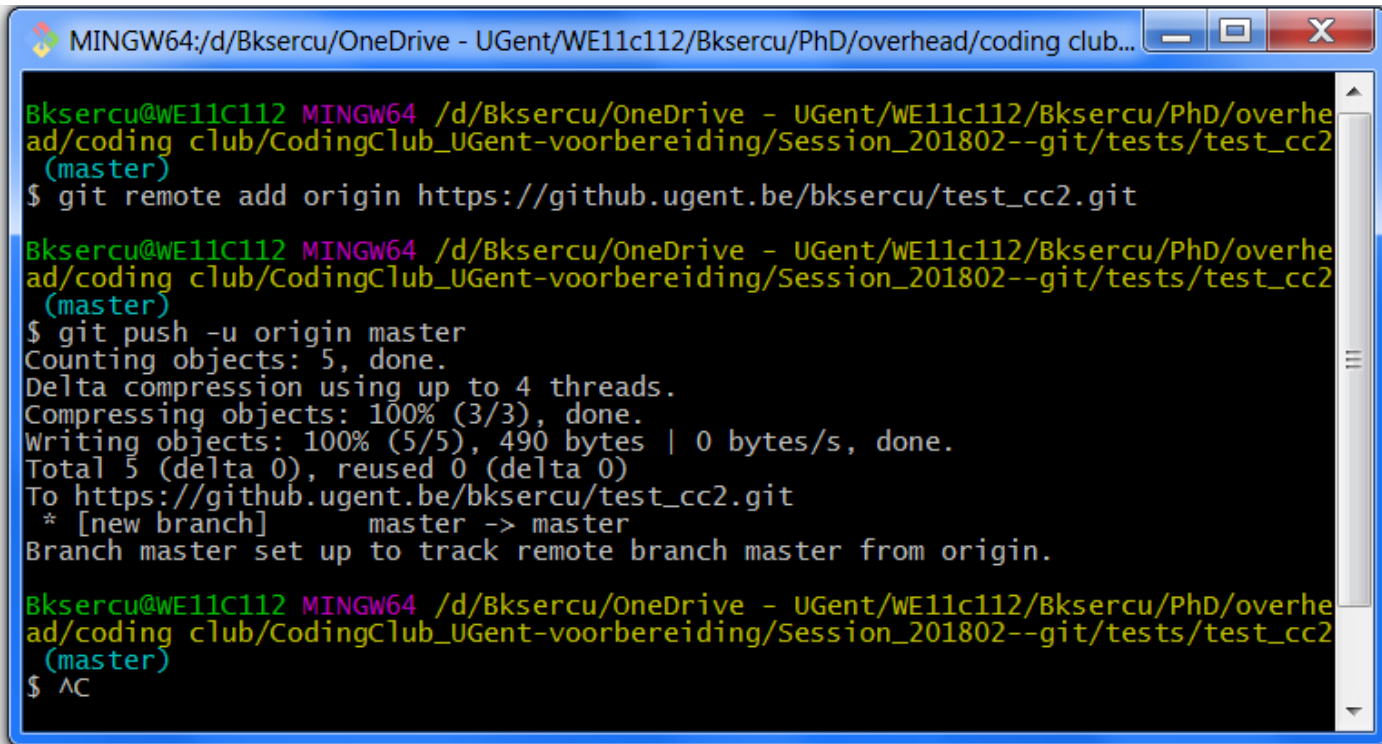
Github reminds you what to do!!

ProTip! Use the URL for this page when adding GitHub as a remote.

Step3: connect the local folder to the online repository



- Type in the console: “git remote add origin *url of the online repository*”
- Click enter and type “git push -u origin master”



```
MINGW64:/d/Bksercu/OneDrive - UGent/WE11c112/Bksercu/PhD/overhead/coding club...
Bksercu@WE11C112 MINGW64 /d/Bksercu/OneDrive - UGent/WE11c112/Bksercu/PhD/overhead/coding club/CodingClub_UGent-voorbereiding/Session_201802--git/tests/test_cc2
(master)
$ git remote add origin https://github.ugent.be/bksercu/test_cc2.git

Bksercu@WE11C112 MINGW64 /d/Bksercu/OneDrive - UGent/WE11c112/Bksercu/PhD/overhead/coding club/CodingClub_UGent-voorbereiding/Session_201802--git/tests/test_cc2
(master)
$ git push -u origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 490 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.ugent.be/bksercu/test_cc2.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

Bksercu@WE11C112 MINGW64 /d/Bksercu/OneDrive - UGent/WE11c112/Bksercu/PhD/overhead/coding club/CodingClub_UGent-voorbereiding/Session_201802--git/tests/test_cc2
(master)
$ ^C
```


Ready!

Install git

- Install github (installs git & github)
 - for mac: <http://mac.github.com>.
 - For windows: <http://windows.github.com>.
 - General installing information:
 - <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Extra information

- Git with R
 - <http://r-bio.github.io/intro-git-rstudio/>
- Git book
 - <https://git-scm.com/book/en/v2>
- Git workflow chart
 - <https://www.git-tower.com/blog/workflow-of-version-control>
- Installing
 - <https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN>

Git GUI's

- Other git GUI's
 - <https://desktop.github.com/>
 - <https://www.slant.co/topics/2089/~git-clients-for-windows>