```
r[[rg_mm$max]] <- terra::extend(r[[rg_mm$max]], ext.combi)

# shift the smaller one over to the other side
ext.min <- terra::ext(r[[rg_mm$min]])
ext.min <- terra::ext(as.numeric(cc.xmin - min(rg)), cc.xmin, ext.min[3], ext.min[4])

terra::ext(r[[rg_mm$min]]) <- ext.min

# extent the smaller one too, resample to larger one
r[[rg_mm$min]] <- terra::extend(r[[rg_mm$min]], ext.combi)
r[[rg_mm$min]] <- terra::resample(r[[rg_mm$min]], r[[rg_mm$max]])

# fuse rasters over grid end
r <- terra::merge(r[[1]], r[[2]])
```

EAGLE M.Sc., MB2 2023/24

Introduction into programming using R

Part II: Concepts, code & practical exercises

Objectives

A training session for

- follow-up content, background inputs and exercise on the week's topics (part I of MB2)

Objectives

A training session for

- follow-up content, background inputs and exercise on the week's topics (part I of MB2)
- clarifying open questions of the week's topics before we move on next week to the next

Objectives

A training session for

- follow-up content, background inputs and exercise on the week's topics (part I of MB2)
- clarifying open questions of the week's topics before we move on next week to the next
- helping with your takes on tasks

Objectives

A training session for

- follow-up content, background inputs and exercise on the week's topics (part I of MB2)
- clarifying open questions of the week's topics before we move on next week to the next
- helping with your takes on tasks
- room for questions: better clarifying something right away as not and then missing a basic building stone for everything that comes next

How to contact me

Jakob Schwalb-Willmann
jakob.schwalb-willmann@uni-wuerzburg.de
+49 (0)931 31-88115

Department of Remote Sensing
Room 01.B.09, 1st floor
John-Skilton-Straße 4a, 97074 Würzburg

Introduction round

Please introduce yourselves!

What's your name? Where do you come from? What is your educational background and scientific interest?

Poll: Open a markdown file

Save it somewhere for later, named *poll.md*. Then, note the answers to the following 3 questions.

1. How would you rate your coding/programming experience before starting EAGLE? (0-5, 0: none, 5: advanced)

1. How would you rate your coding/programming experience before starting EAGLE? (0-5, 0: none, 5: advanced)
2. Which programming languages have you already used/worked with?

1. How would you rate your coding/programming experience before starting EAGLE? (0-5, 0: none, 5: advanced)
2. Which programming languages have you already used/worked with?
3. Which operating system is running on the computer that you use for studying?

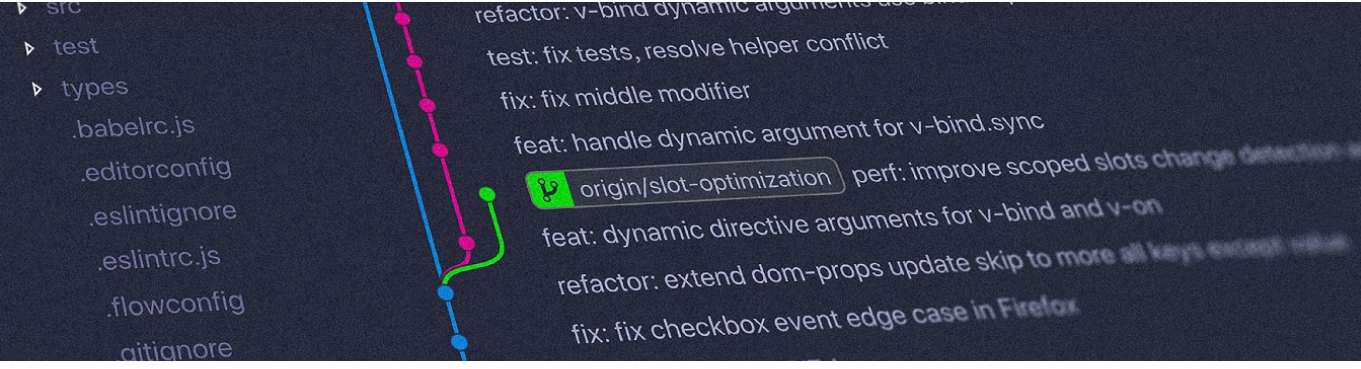
1. How would you rate your coding/programming experience before starting EAGLE? (0-5, 0: none, 5: advanced)
2. Which programming languages have you already used/worked with?
3. Which operating system is running on the computer that you use for studying?

1. How would you rate your coding/programming experience before starting EAGLE? (0-5, 0: none, 5: advanced)
2. Which programming languages have you already used/worked with?
3. Which operating system is running on the computer that you use for studying?

Use the following template for answering the above questions:

```
# Your name
* Answer to question 1
* Answer to question 2
* Answer to question 3
```

Everyone has poll.md locally saved now. Imagine we wanted to collaborate on this file, while also tracking changes. How could we achieve that?



Follow-up:

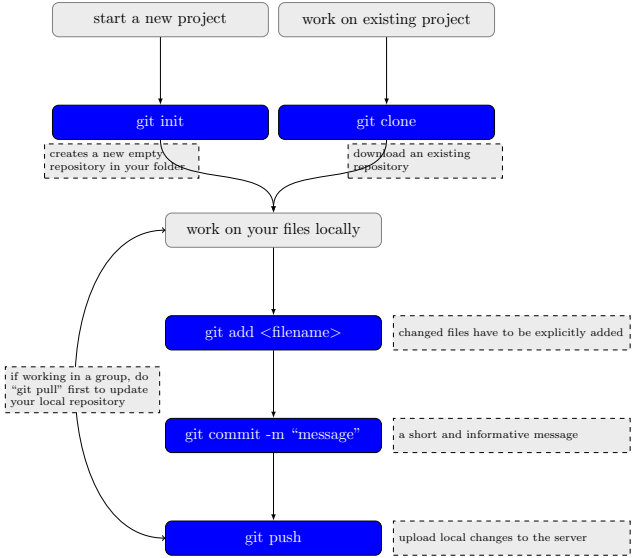
Version control and collaboration using git

Recap: Why should I use git?

Why should I use git?

- **version control**: being able to identify mistakes and revert changes made to files
- **collaboration**: remove barriers of well documented collaborations
- **accountability**: transparent history of your development
- **early start**: mastery takes time, good to learn early

Recap: Git flow-chart



git $\neq$ GitHub

- **git** is a version control software invented by Linus Torvalds that can be installed locally on any system

git $\neq$ GitHub

- **git** is a version control software invented by Linus Torvalds that can be installed locally on any system
- **git** can be *initialized* for any local directory that you want to track (e.g. to revert unintentional or unwanted changes)

git $\neq$ GitHub

- **git** is a version control software invented by Linus Torvalds that can be installed locally on any system
- **git** can be *initialized* for any local directory that you want to track (e.g. to revert unintentional or unwanted changes)
- *optionally*, you can connect your local **git** repository with another **git** repository somewhere else (*upstream*)

git $\neq$ GitHub

- **git** is a version control software invented by Linus Torvalds that can be installed locally on any system
- **git** can be *initialized* for any local directory that you want to track (e.g. to revert unintentional or unwanted changes)
- *optionally*, you can connect your local **git** repository with another **git** repository somewhere else (*upstream*)
- this allows you to push changes of your local repository to the *upstream* repository

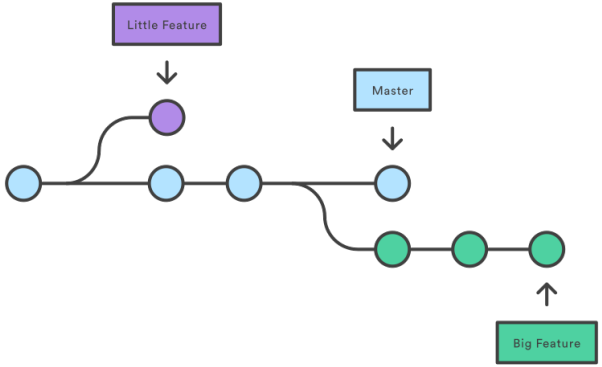
git $\neq$ GitHub

- **git** is a version control software invented by Linus Torvalds that can be installed locally on any system
- **git** can be *initialized* for any local directory that you want to track (e.g. to revert unintentional or unwanted changes)
- *optionally*, you can connect your local **git** repository with another **git** repository somewhere else (*upstream*)
- this allows you to push changes of your local repository to the *upstream* repository
- the upstream repository can be hosted on a **git** server, e.g. **GitHub**

git $\neq$ GitHub

- **git** is a version control software invented by Linus Torvalds that can be installed locally on any system
- **git** can be *initialized* for any local directory that you want to track (e.g. to revert unintentional or unwanted changes)
- *optionally*, you can connect your local **git** repository with another **git** repository somewhere else (*upstream*)
- this allows you to push changes of your local repository to the *upstream* repository
- the upstream repository can be hosted on a **git** server, e.g. **GitHub**
- **GitHub** is a service offered by Microsoft that offers public and private git repository servers for collaboration and sharing

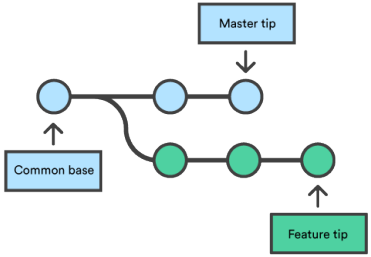
Terminology



1 Branching in git – *atlassian.com*

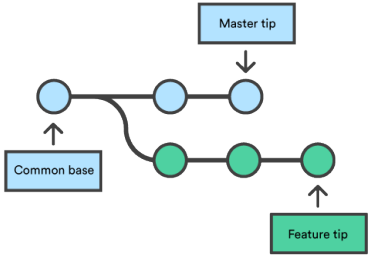
```
1       git branch mynewbranch
2       git checkout mynewbranch
```

Terminology

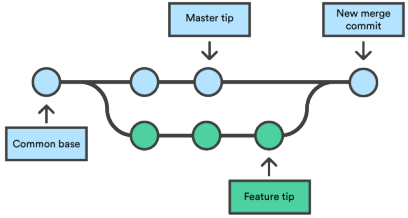


2 Before merging two branches in git...

Terminology



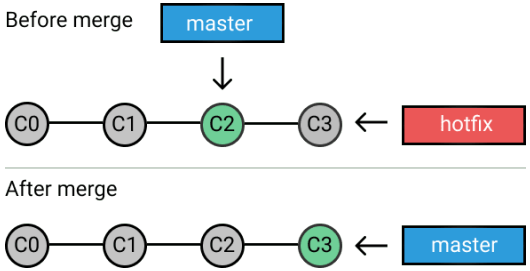
4 Before merging two branches in git...



5 ...and after.

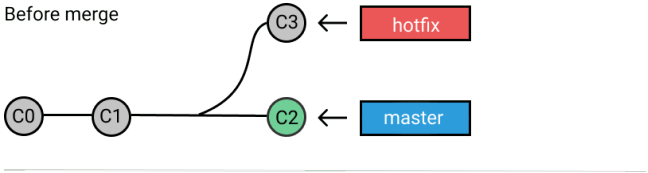
```
1       git chekout master
2       git merge mynewbranch
3       git branch -d mynewbranch
```

Terminology



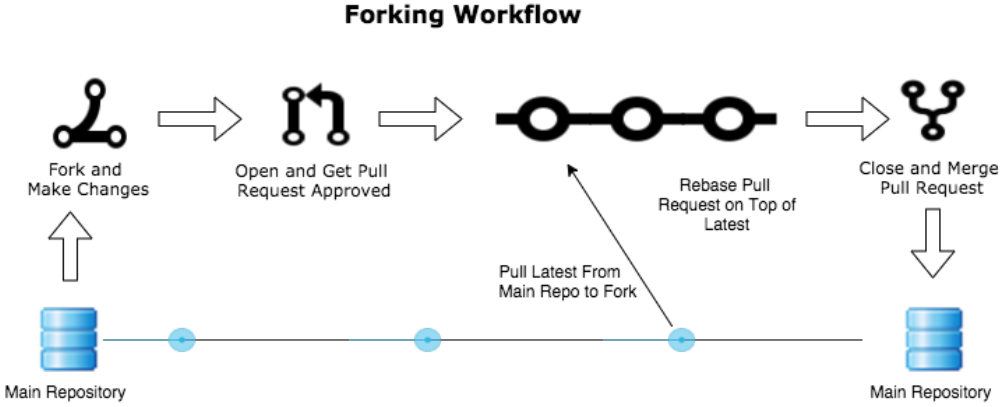
6 Fast-forward merge

Terminology



7 Three-way merge

Terminology



8 Git collaboration using fork and pull request

Joint task: Collaboration using git and GitHub

Imagine the following scenario:

- You all want to collaborate on a **joint project** (e.g. an analysis, documentation of it etc.)

Joint task: Collaboration using git and GitHub

Imagine the following scenario:

- You all want to collaborate on a **joint project** (e.g. an analysis, documentation of it etc.)
- There is no person who "oversees" the project and decides on how to merge individual work (e.g. due to a flat hierarchy, equal partners, the amount of resources needed to organize the collaboration in this fashion etc.)

Joint task: Collaboration using git and GitHub

Imagine the following scenario:

- You all want to collaborate on a **joint project** (e.g. an analysis, documentation of it etc.)
- There is no person who "oversees" the project and decides on how to merge individual work (e.g. due to a flat hierarchy, equal partners, the amount of resources needed to organize the collaboration in this fashion etc.)
- Instead, you decide to use version control for your collaboration (git) and a git server service for collaboration (GitHub)

Joint task: Collaboration using git and GitHub

Imagine the following scenario:

- You all want to collaborate on a **joint project** (e.g. an analysis, documentation of it etc.)
- There is no person who "oversees" the project and decides on how to merge individual work (e.g. due to a flat hierarchy, equal partners, the amount of resources needed to organize the collaboration in this fashion etc.)
- Instead, you decide to use version control for your collaboration (git) and a git server service for collaboration (GitHub)
- Your version control setup will manage your collaboration by tracking all versions of the work being done by each individual collaborator and its changes

Joint task: Collaboration using git and GitHub

Imagine the following scenario:

- You all want to collaborate on a **joint project** (e.g. an analysis, documentation of it etc.)
- There is no person who "oversees" the project and decides on how to merge individual work (e.g. due to a flat hierarchy, equal partners, the amount of resources needed to organize the collaboration in this fashion etc.)
- Instead, you decide to use version control for your collaboration (git) and a git server service for collaboration (GitHub)
- Your version control setup will manage your collaboration by tracking all versions of the work being done by each individual collaborator and its changes
- Your git server service allows you to view and publish your work

Make sure to have git installed

If git is *not installed* on your OS, install it from

`https://git-scm.com/downloads`

Initialization

Open a git shell and initialize your local git installation (if not already done):

```
git config --global user.email "your-email@xyz.de"
git config --global user.name "yourGitHubUser"
```

This makes sure that all your commits and other changes are signed with your user name and email.

Authentification with GitHub

Creating and using an GitHub authentifiaction token

Joint task: Collaboration using git and GitHub

Task:

1. Fork the prepared GitHub repository at https://github.com/16EAGLE/MB12 via GitHub (create an account if you do not have one yet)
2. Create a local git repository that is connected to your fork (e.g. using RStudio's "New Project" feature)
3. Pull everything from your fork on GitHub
4. Start editing the contained *README.md*, e.g. by adding a section with your name

Joint task: Collaboration using git and GitHub

Task:

1. Fork the prepared GitHub repository at https://github.com/16EAGLE/MB12 via GitHub (create an account if you do not have one yet)
2. Create a local git repository that is connected to your fork (e.g. using RStudio's "New Project" feature)
3. Pull everything from your fork on GitHub
4. Start editing the contained *README.md*, e.g. by adding a section with your name
5. Copy the previously created *poll.md* file into the same directory as *README.md*.
6. Commit your changes and push them
7. Create a pull request in the original repository
8. We will merge the changes together and see if we find conflicts to resolve using GitHub

Joint task: Collaboration using git and GitHub

Task:

1. Fork the prepared GitHub repository at https://github.com/16EAGLE/MB12 via GitHub (create an account if you do not have one yet)
2. Create a local git repository that is connected to your fork (e.g. using RStudio's "New Project" feature)
3. Pull everything from your fork on GitHub
4. Start editing the contained *README.md*, e.g. by adding a section with your name
5. Copy the previously created *poll.md* file into the same directory as *README.md*.
6. Commit your changes and push them
7. Create a pull request in the original repository
8. We will merge the changes together and see if we find conflicts to resolve using GitHub

Extra:

- What happens if we create merge conflicts? How could we resolve them?
- How can we do all of this soley on the command line?

Joint task:

Collaboration using git and GitHub

Joint task: Pulling your fork using the git cli

In your local directory that you want to use for the project (use *cd* to get there):

```
git init
git remote add origin https://github.com/yourGitHubUser/MB12.git
git pull origin master
```

Make your changes (edit the markdown file, copy new files etc.), then:

```
git add myfile.R
git commit -m "my commit message"
git push -u origin master
```

Now, you may create a *Pull Request* at `https://github.com/16EAGLE/MB12`. Next time, you will se how to merge a PR on the command line instead of using github's web GUI.

https://github.githistory.xyz/

Visualize the history of a file using githistory.xyz

Recap: **Tasks of this week's MB2 session**

Recap: Task 1/2 – git

1. setup your own git account e.g. with github or bitbucket
2. structure your account (folder) to suit your needs
3. commit a first project from your file system
4. commit a first project from RStudio
   - a normal R script with comments
   - a knitr script (report and presentation)
5. share/clone a project with others e.g. little helper scripts
6. recommendation: project work only via git and as knitr

Recap: Task 1/2 – git

1. setup your own git account e.g. with github or bitbucket
2. structure your account (folder) to suit your needs
3. commit a first project from your file system
4. commit a first project from RStudio
   - a normal R script with comments
   - a knitr script (report and presentation)
5. share/clone a project with others e.g. little helper scripts
6. recommendation: project work only via git and as knitr

Recap: Task 2/2 – Markdown

1. create a document (html, pdf and doc)
2. create a presentation (html and pdf)
3. insert different commands and plots
4. make code visible and execute it; visible but not executable; invisible and executable
5. change transitions (presentations) or style (see rstudio webpage for markdown)

Extra, for the fast among you:

- officeR: http://blog.revolutionanalytics.com/2017/10/officer-powerpoint.html
- rticles, flexdashboards, slidify or shiny

Are there issues with the tasks? Are there open questions regarding the contents of this week?

Tasks

Do the two exercises from Tuesday to gain experience with *markdown* and *git*.

Any questions?

Feedback

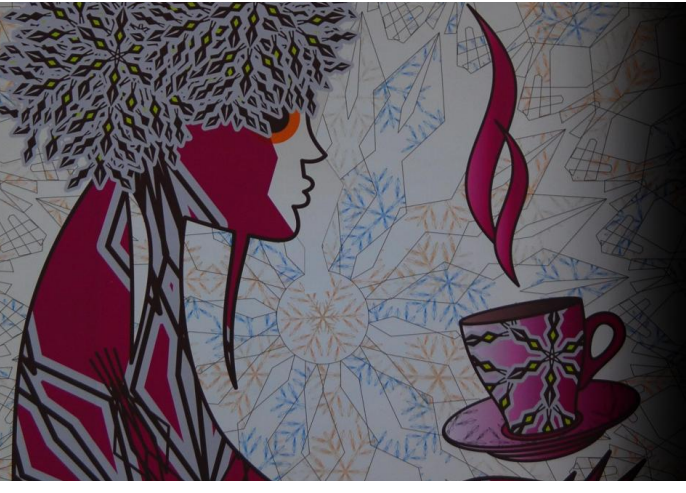
Anonymous feedback via slido.com, event code **#mb2training**.

What we would like to know from you:

- Is the speed ok?
- Is the session structure ok?
- Is there any content you'd like to be covered?
- Is it boring or not?
- Is it helpful or not?
- Is the presentation style ok?
- Anything else?

Go to slido.com & join #mb2training

Please give feedback using the feedback box.



Coffee time – see you next week