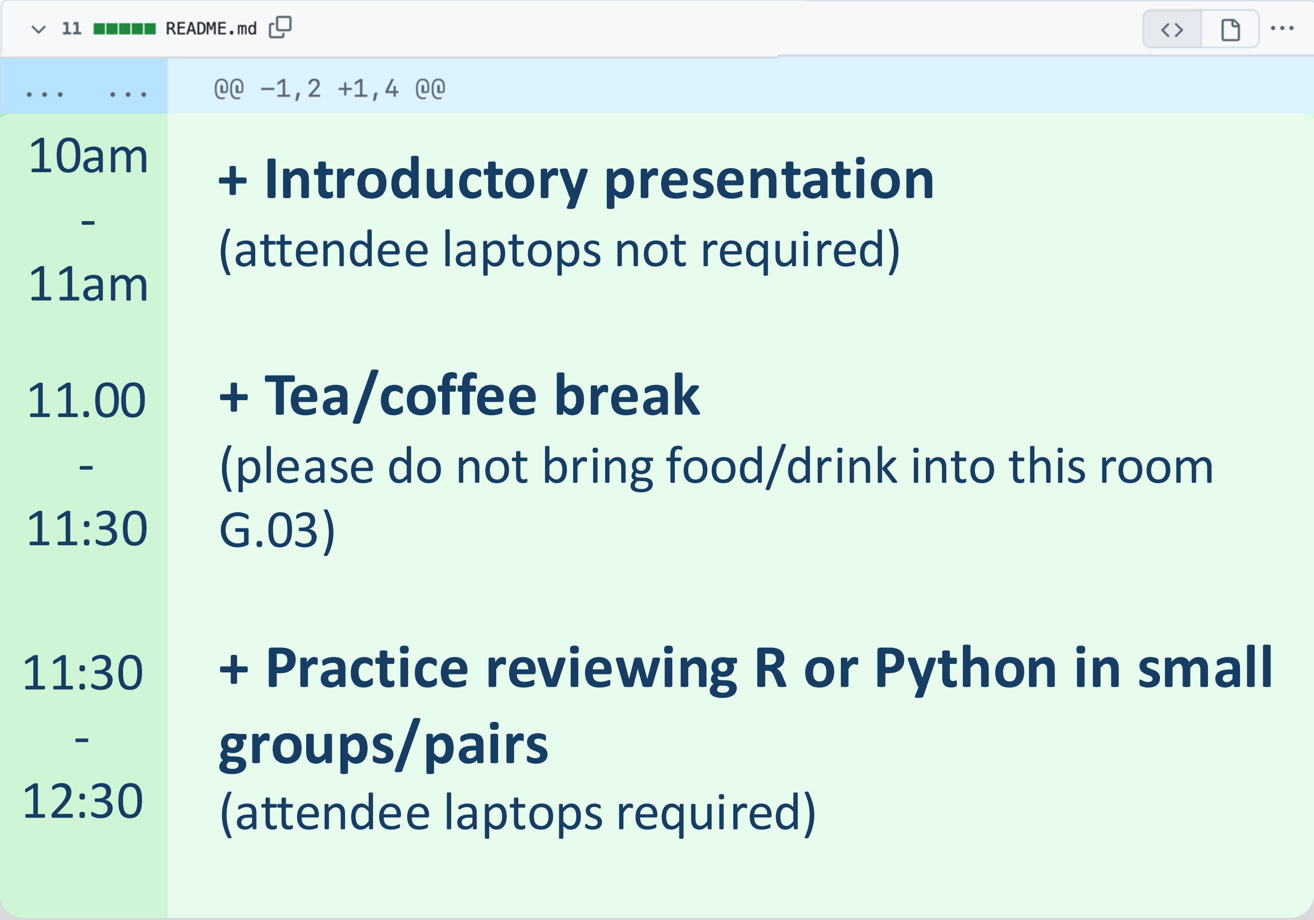
Code Review in Academia Workshop



... ... @@ -1,2 +1,4 @@

10am - 11am	+ **Introductory presentation** (attendee laptops not required)
11.00 - 11:30	+ **Tea/coffee break** (please do not bring food/drink into this room G.03)
11:30 - 12:30	+ **Practice reviewing R or Python in small groups/pairs** (attendee laptops required)

Thursday
13th March 2025

Room G.03
Bayes Centre

The University of
Edinburgh,
EH8 9BT

Who we are

Amelia Edmondson-Stait
(post-doc)

Emily Ball
(post-doc)

Hannah Casey
(PhD student)

Ella Davyson
(PhD student)

Poppy Grimes
(PhD student)

- Psychiatry Department, in Prof Andrew McIntosh & Prof Heather Whalley's groups
- Apply epidemiological and statistical methods to large population cohorts and electronic health records.
- Advocators of good coding practices in academia.
- This workshop aims to initiate discussion and teach some concepts of code review.
- Funding for this workshop comes from the **Improving Research Community Builder Award**

Workshop Expectations

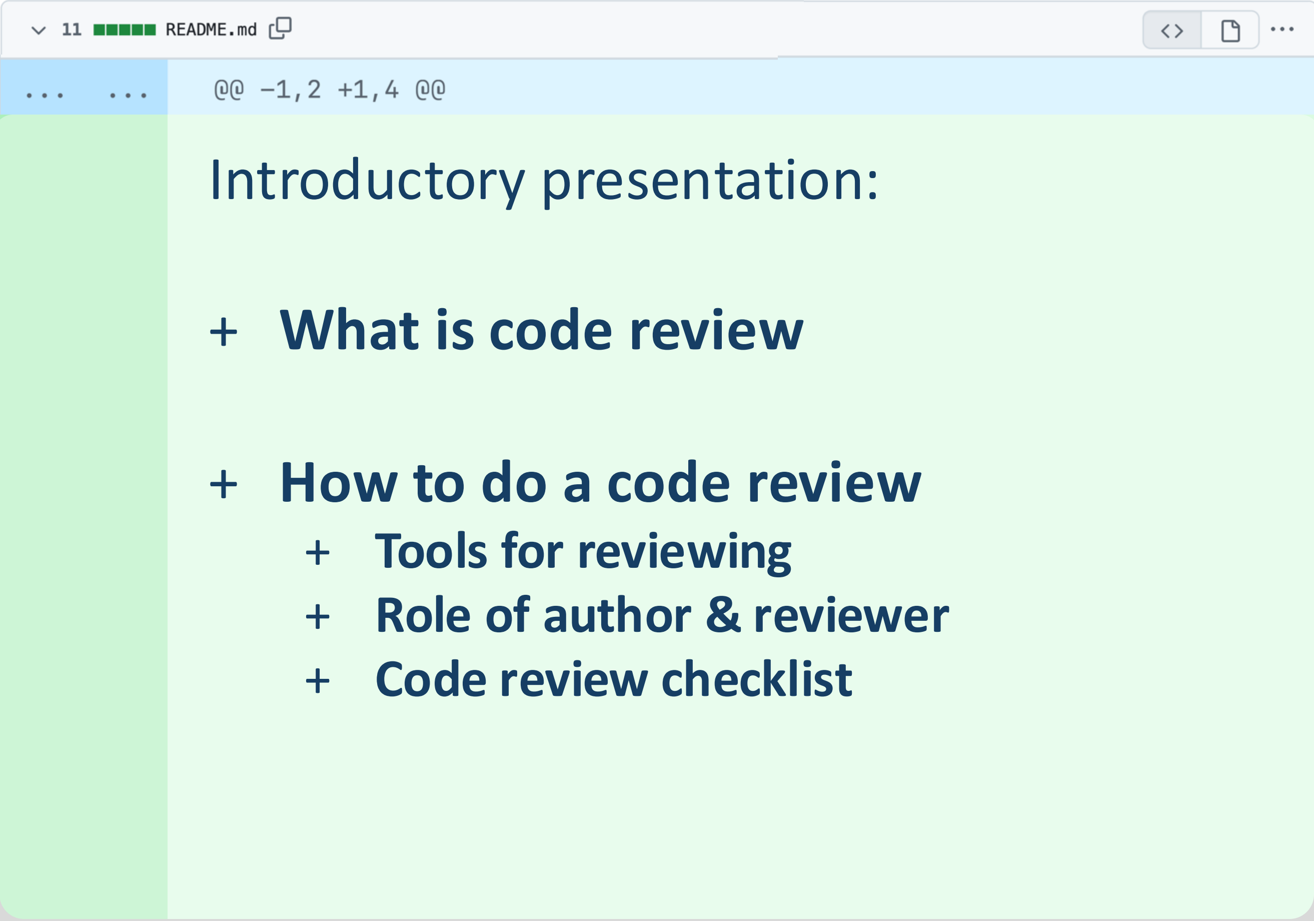
This is a safe, supportive, respectful space

Participation is encouraged

It's ok to make mistakes

We'll use a "Parking Sheet" for items that are off topic or taking up too much time

Code Review in Academia Workshop



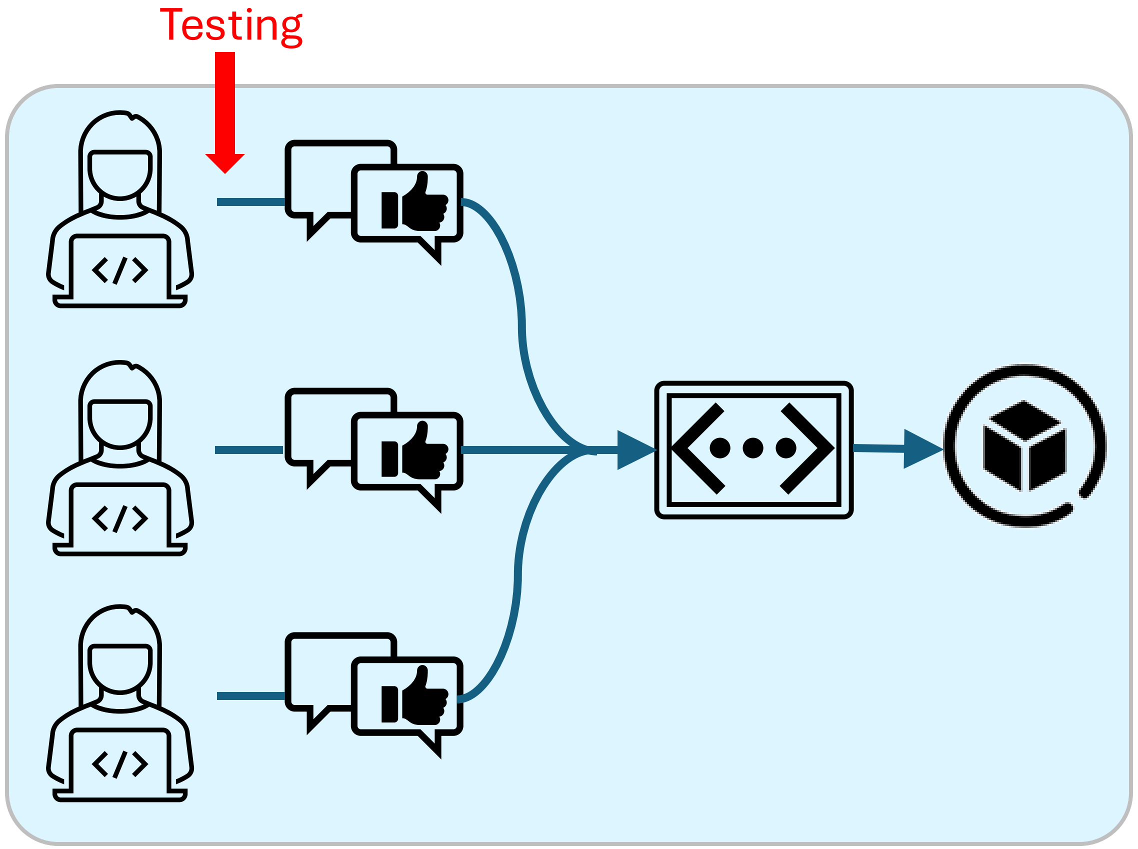
```
...   ...   @@ -1,2 +1,4 @@
```

Introductory presentation:

+ **What is code review**

+ **How to do a code review**
  + **Tools for reviewing**
  + **Role of author & reviewer**
  + **Code review checklist**

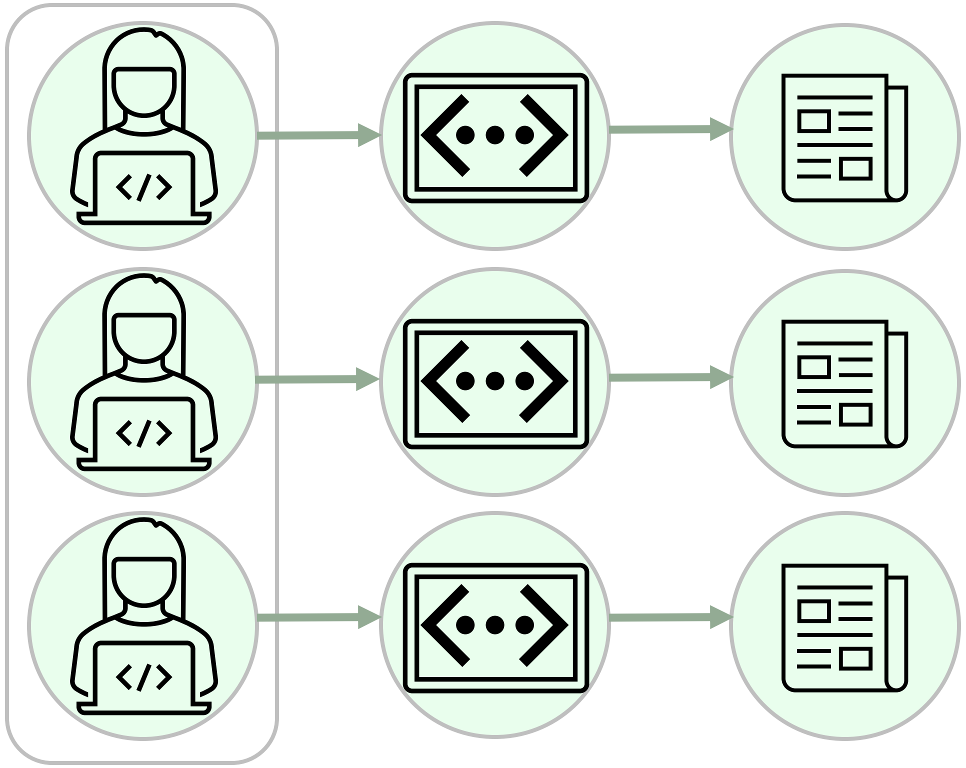
Coding practices in industry



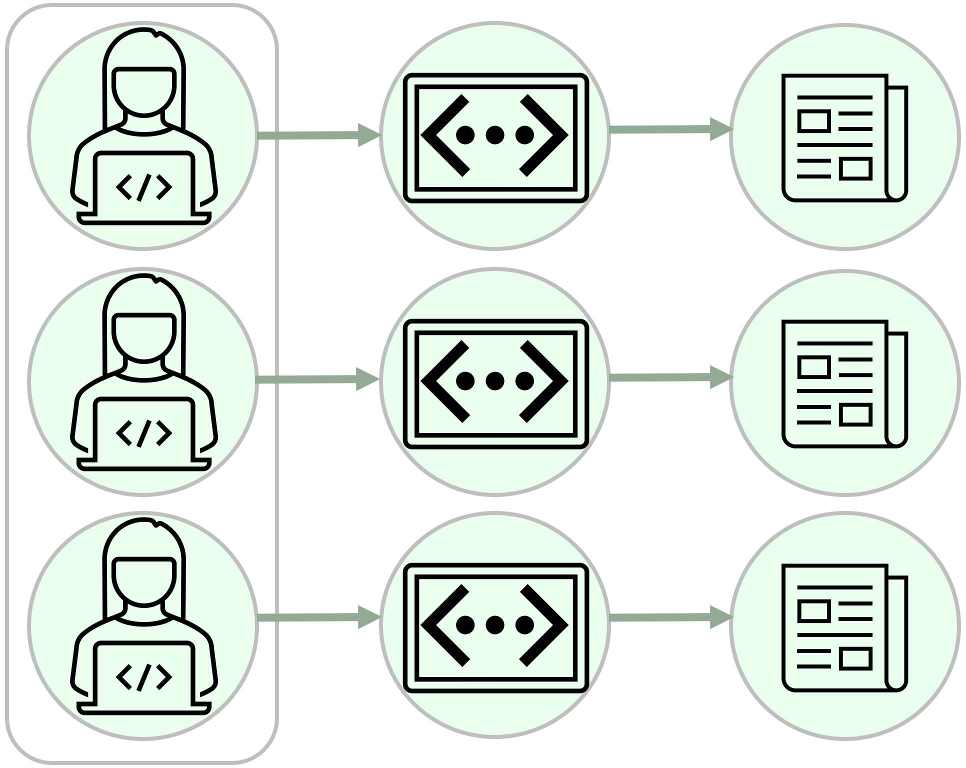
Why code review is important in industry

- Mistakes happen
- ↑ code quality
- Opportunity to learn
- Discover bugs earlier
- ↑ maintainability of code

Coding practices in academia



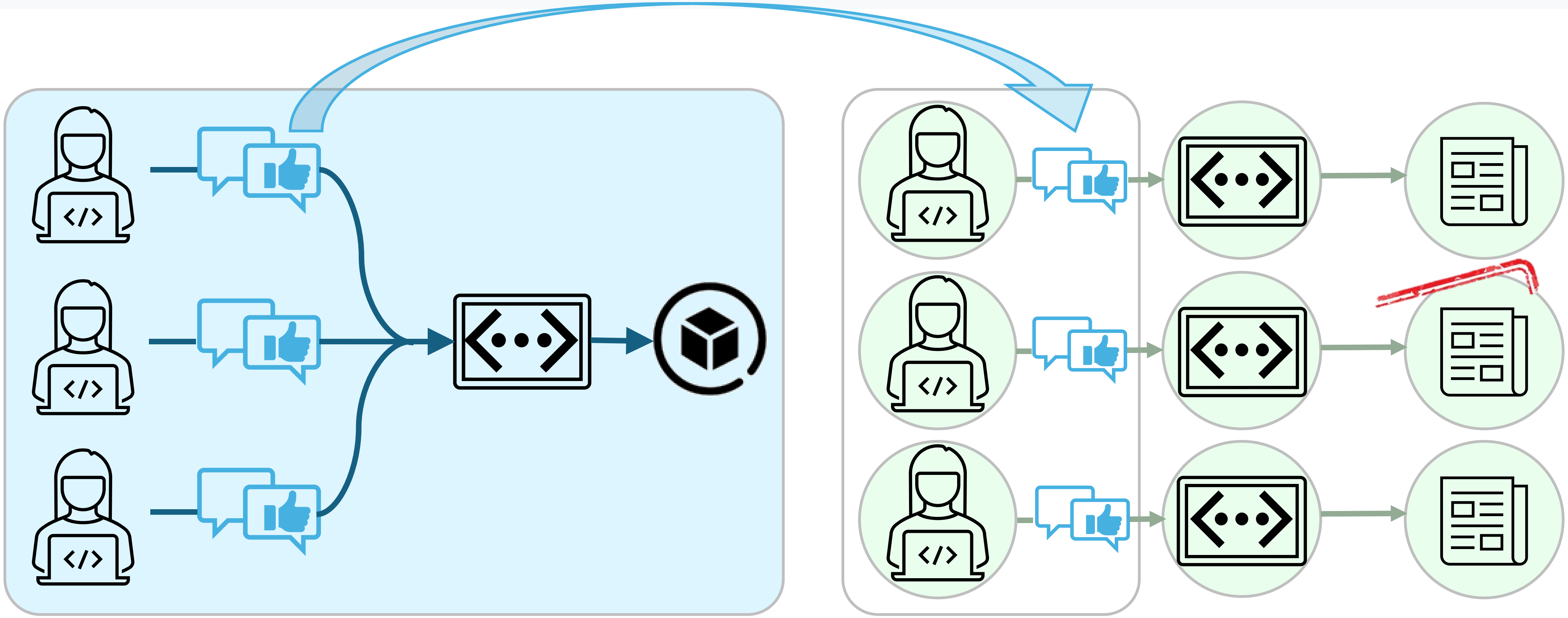
Coding practices in academia



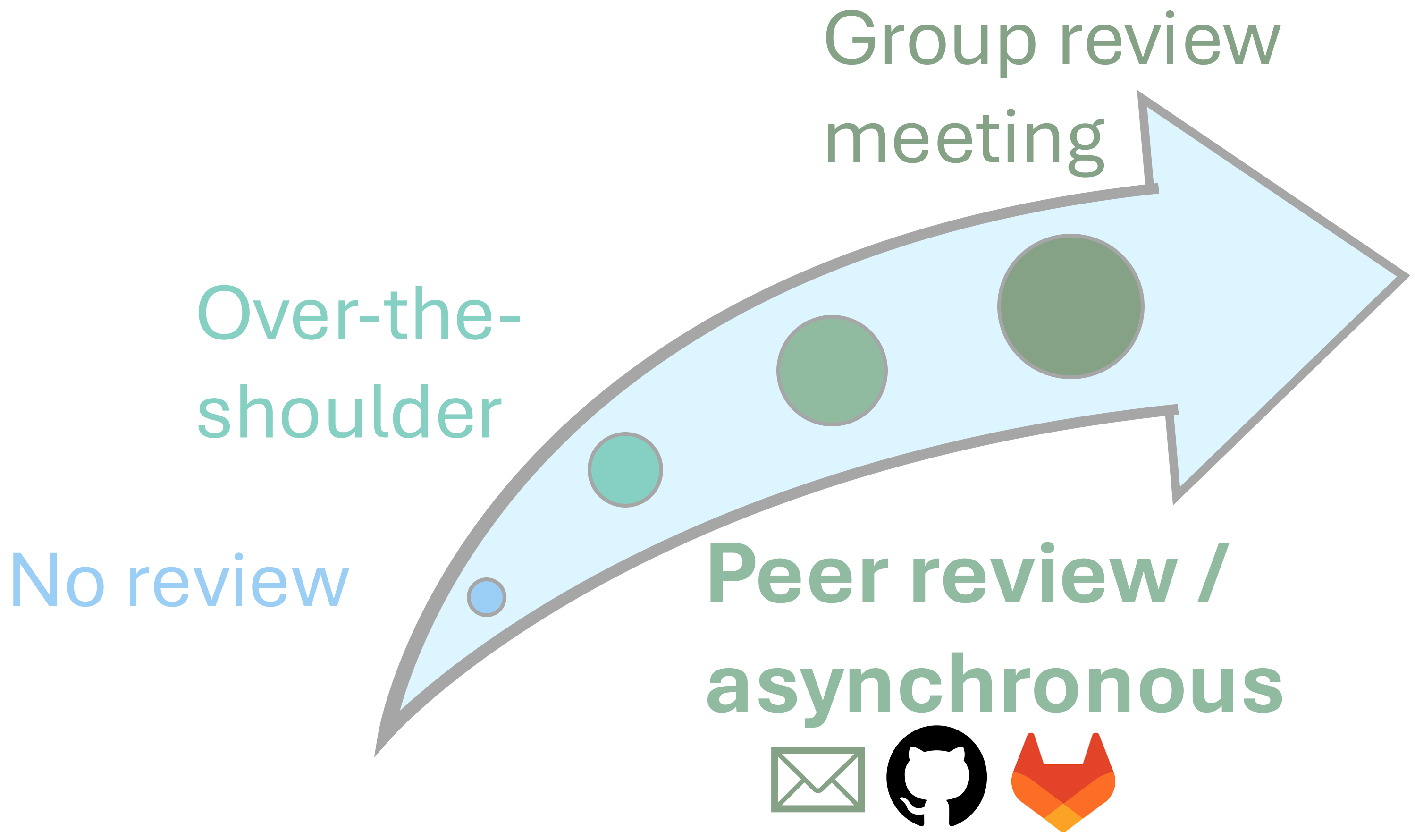
Why code review is important in academia

- Mistakes happen
- ↑ code quality → reproducibilty
- ↑ research quality
- Opportunity to learn
- Discover bugs earlier
- ↑ maintainability → reproducibilty

Apply industry methods to academia



Types of code review



No review

Over-the-shoulder

Group review meeting

Peer review / asynchronous

Summary so far...

- Coding in industry vs academia is different
- Code review is probably also going to be different

Next...

- Methods used for asynchronous code review

Industry uses “pullrequests” for peer review

Image from: https://r-cubed-intro.rostools.org/sessions/version-control

What is git... git branches... pull requests?

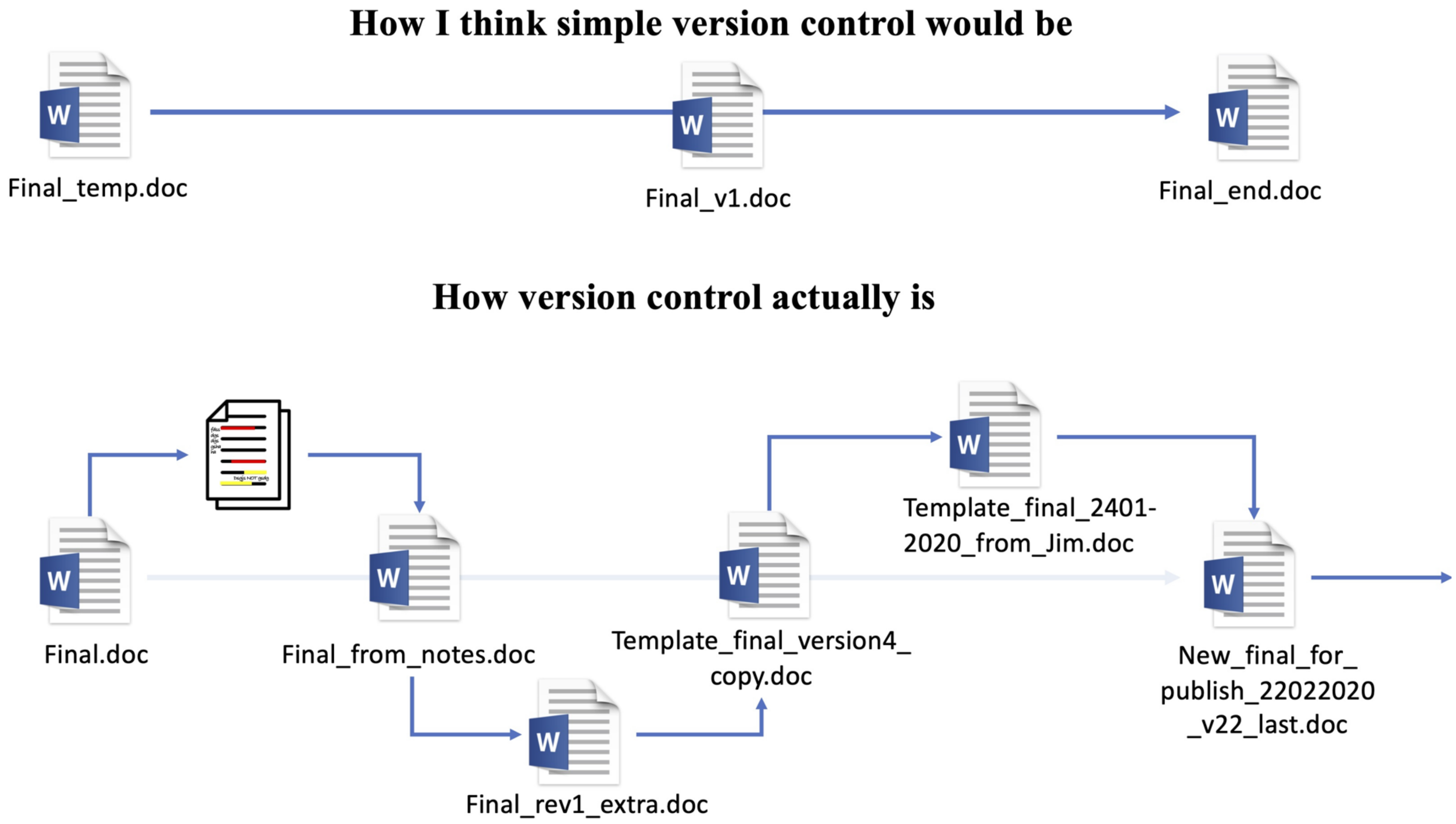

How I think simple version control would be
Final_temp.doc
Final_v1.doc
Final_end.doc
How version control actually is
Final.doc
Final_from_notes.doc
Final_rev1_extra.doc
Template_final_version4_ copy.doc
Template_final_2401-2020_from_Jim.doc
New_final_for_ publish_22022020 _v22_last.doc

Image from: https://r-cubed-intro.rostools.org/sessions/version-control

What is git... git branches... pull requests?

git = version control system

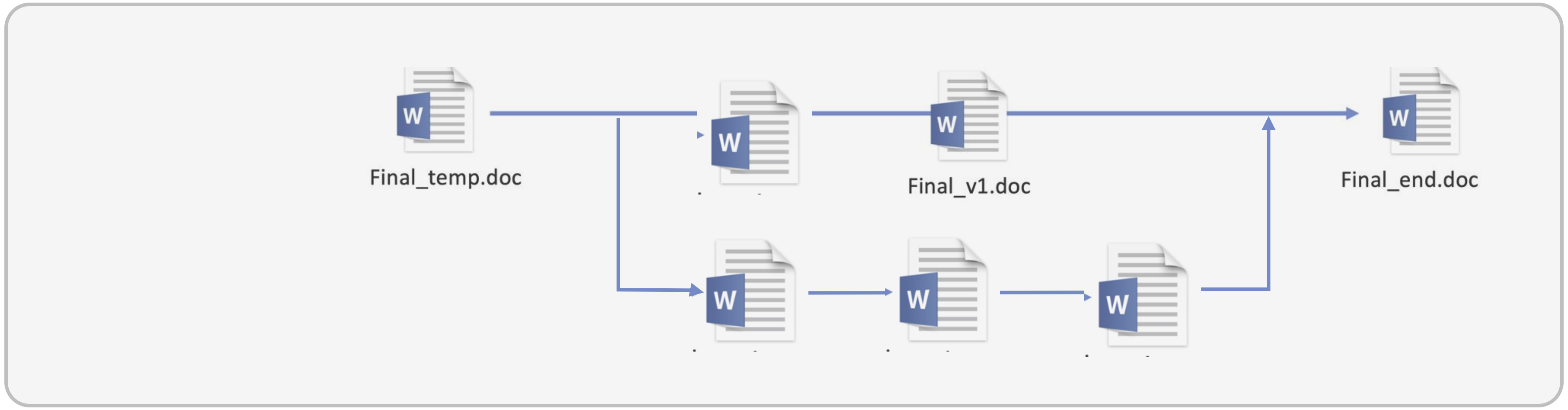
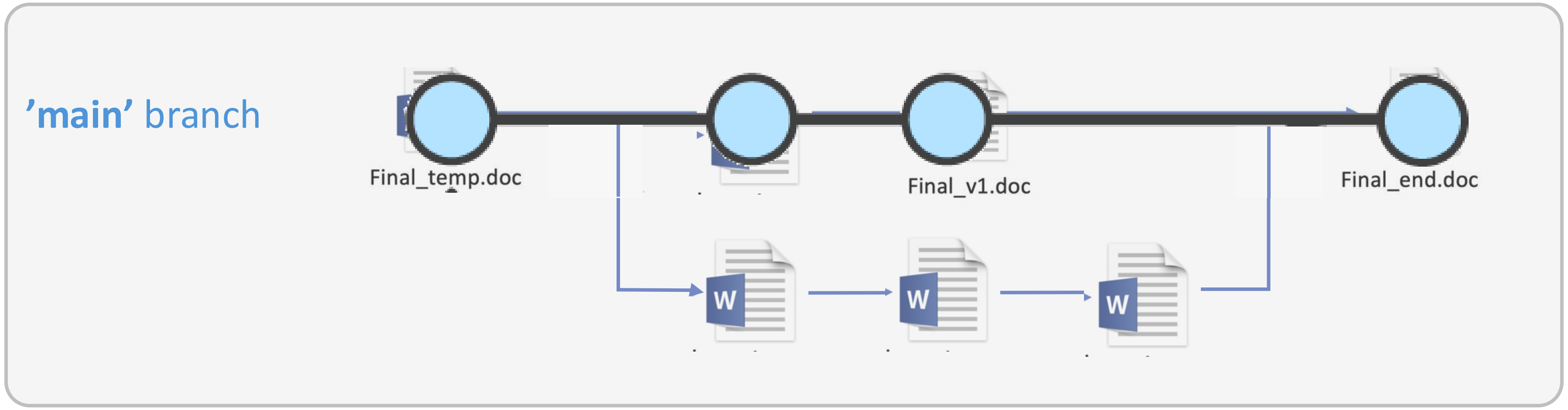


Image from: https://r-cubed-intro.rostools.org/sessions/version-control

What is git... git branches... pull requests?

git = version control system

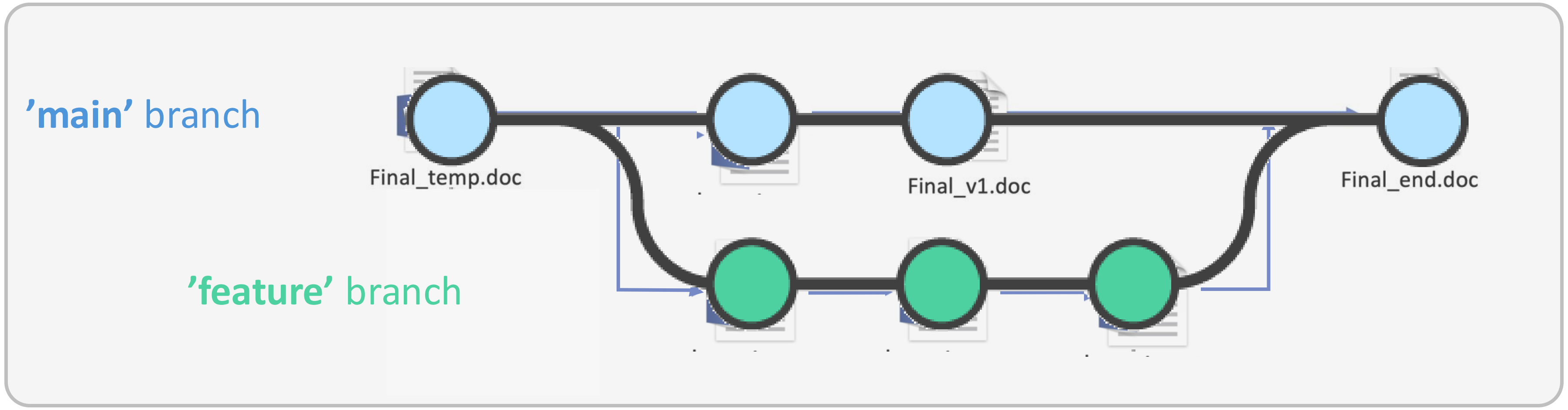


● = git commit
(change in file)

Image from: https://r-cubed-intro.rostools.org/sessions/version-control

What is git... **git branches...** pull requests?

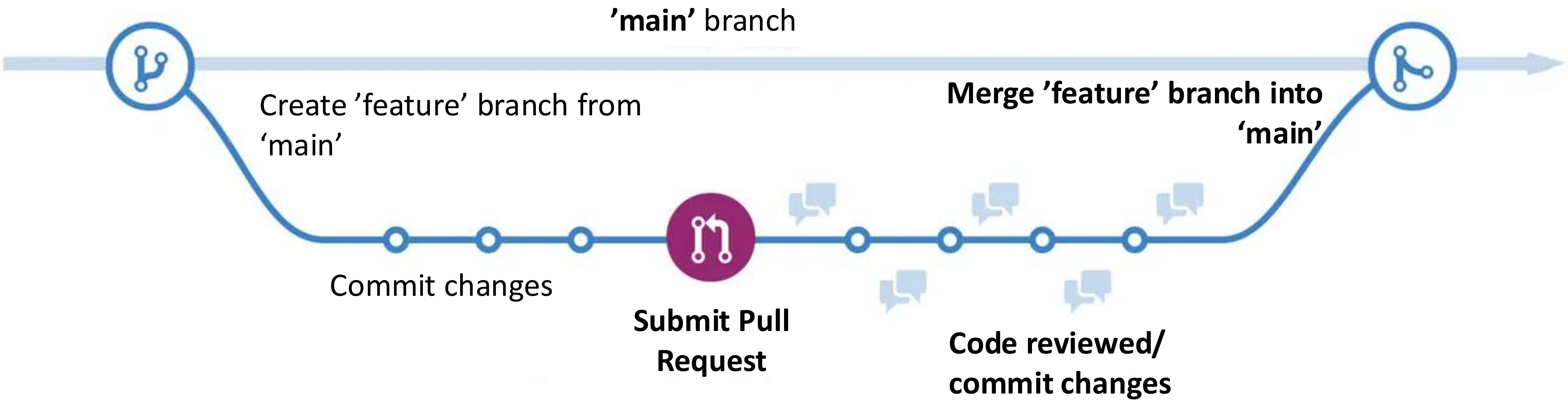
git = version control system



● = git commit
(change in file)

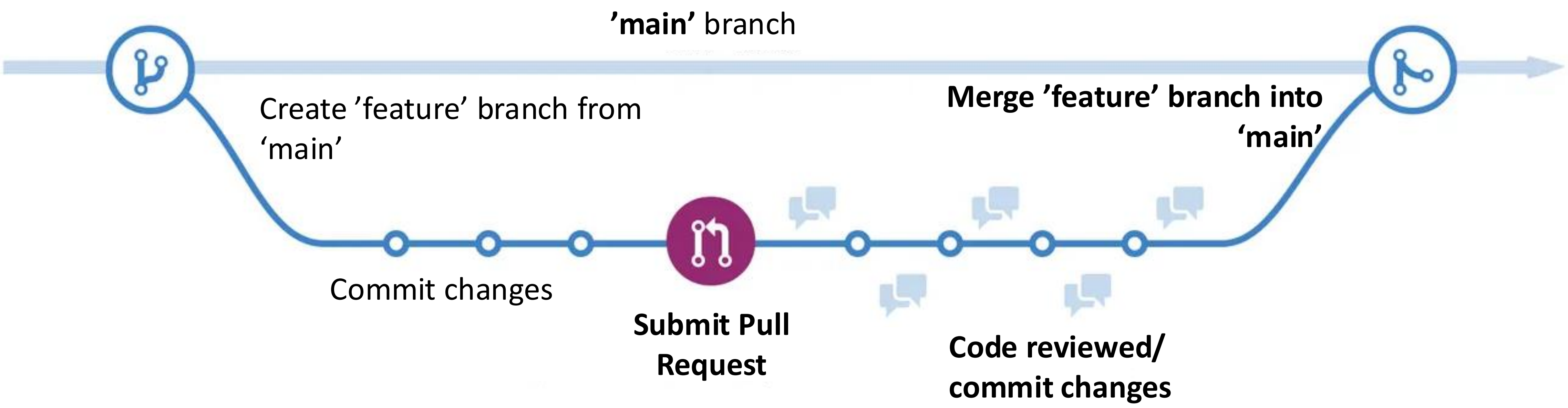
Image from: https://r-cubed-intro.rostools.org/sessions/version-control

What is git... git branches... **pull requests?**


’main’ branch
Create ’feature’ branch from ‘main’
Merge ’feature’ branch into ‘main’
Commit changes
Submit Pull Request
Code reviewed/ commit changes

Pull requests are useful for reviewing changes made to an existing code base

Industry uses “pull requests” for peer review


’main’ branch
Create ’feature’ branch from ‘main’
Merge ’feature’ branch into ‘main’
Commit changes
Submit Pull Request
Code reviewed/ commit changes

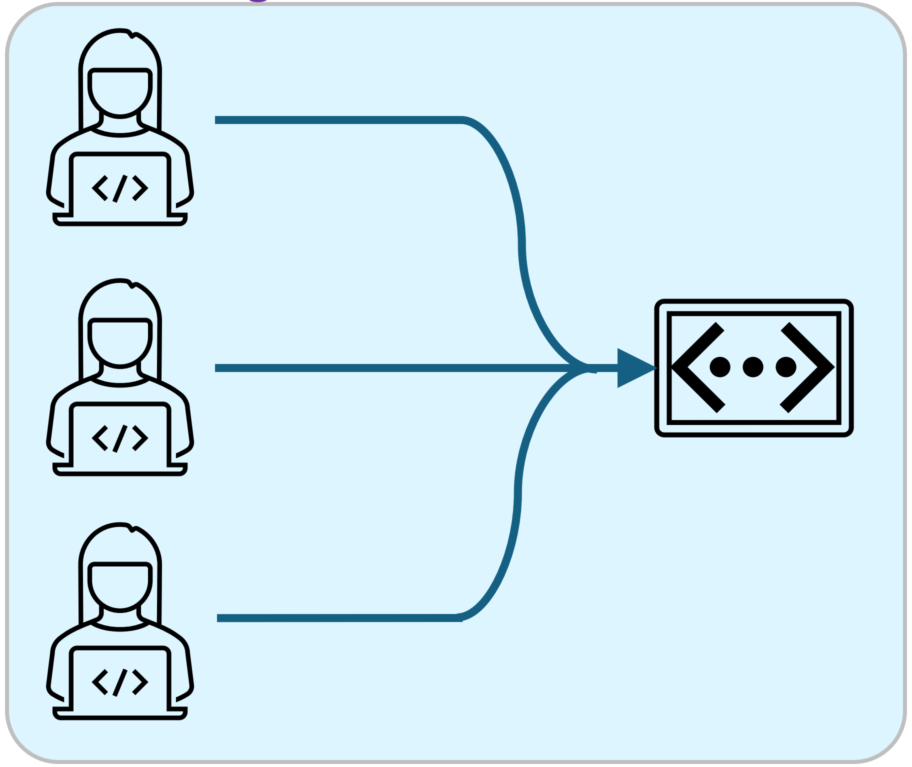
Pull requests are useful for reviewing changes made to an existing code base

“Pull requests” for peer review in academia?

Industry:

Changes made to an existing code base

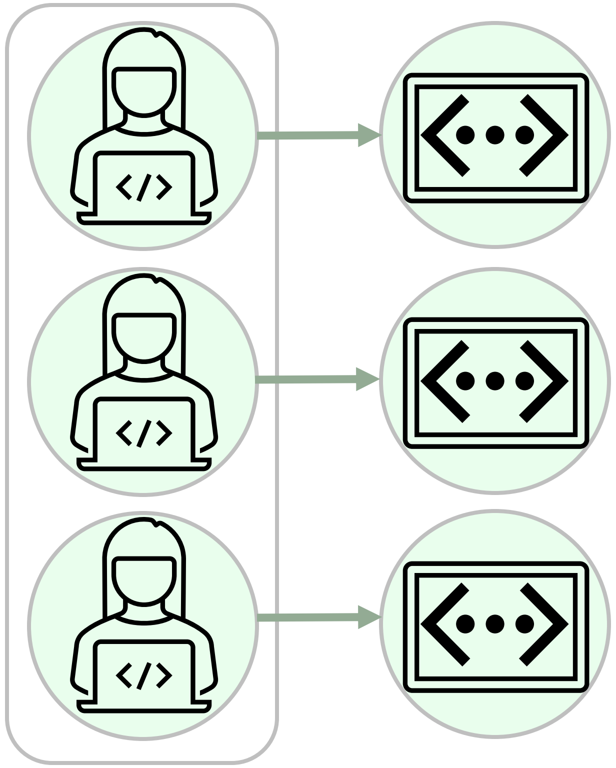
Changes to code are reviewed



Academia:

Each researcher/project = new code base

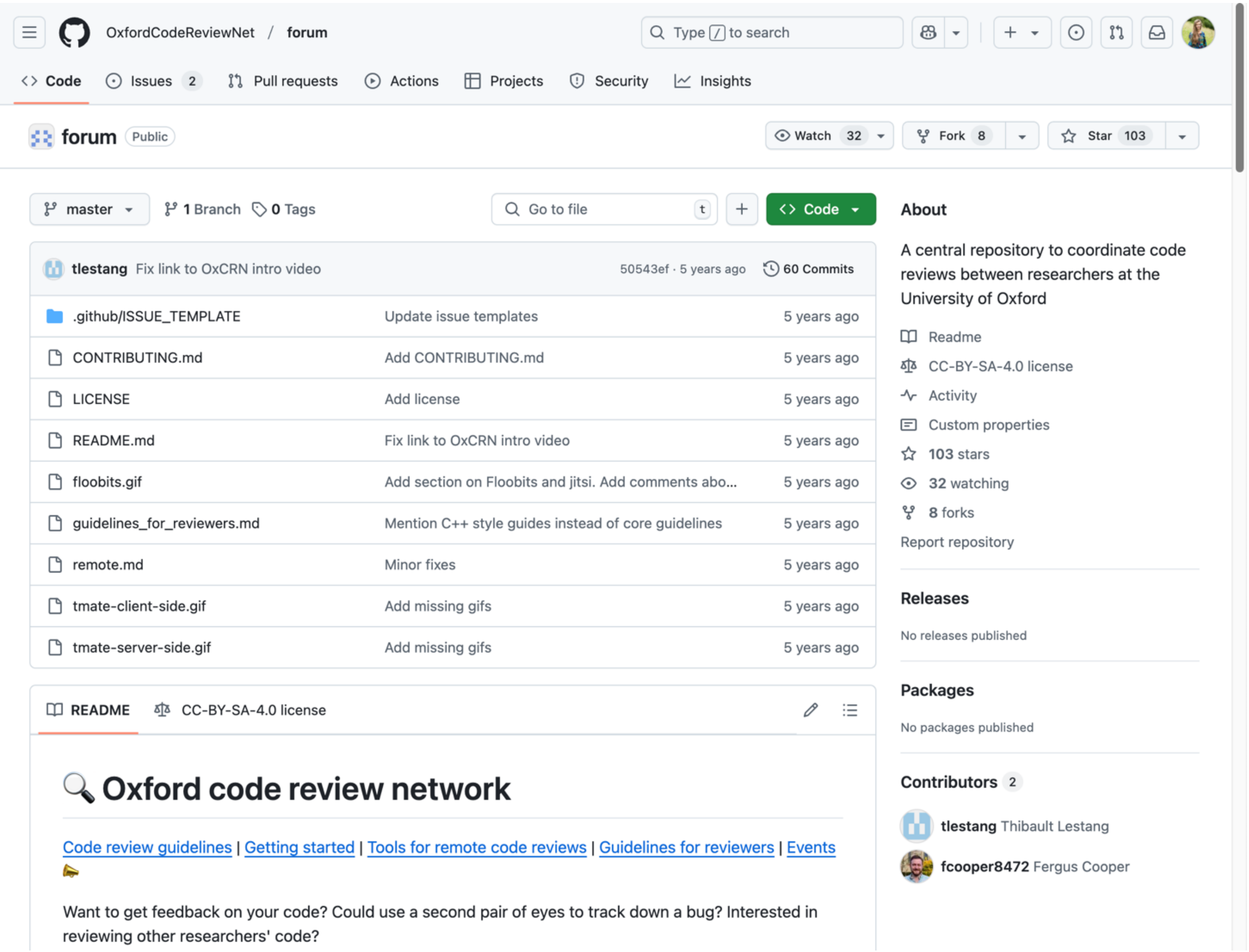
Not just changes that need reviewing



Do we need something less advanced and more suitable for academia?

Oxford Code Review Network

https://github.com/OxfordCodeReviewNet/forum



tlestang Fix link to OxCRN intro video 50543ef · 5 years ago 60 Commits

.github/ISSUE_TEMPLATE	Update issue templates	5 years ago
CONTRIBUTING.md	Add CONTRIBUTING.md	5 years ago
LICENSE	Add license	5 years ago
README.md	Fix link to OxCRN intro video	5 years ago
floobits.gif	Add section on Floobits and jitsi. Add comments abo...	5 years ago
guidelines_for_reviewers.md	Mention C++ style guides instead of core guidelines	5 years ago
remote.md	Minor fixes	5 years ago
tmate-client-side.gif	Add missing gifs	5 years ago
tmate-server-side.gif	Add missing gifs	5 years ago

🔍 Oxford code review network

Code review guidelines | Getting started | Tools for remote code reviews | Guidelines for reviewers | Events 📣

Want to get feedback on your code? Could use a second pair of eyes to track down a bug? Interested in reviewing other researchers' code?

About

A central repository to coordinate code reviews between researchers at the University of Oxford

Releases

No releases published

Packages

No packages published

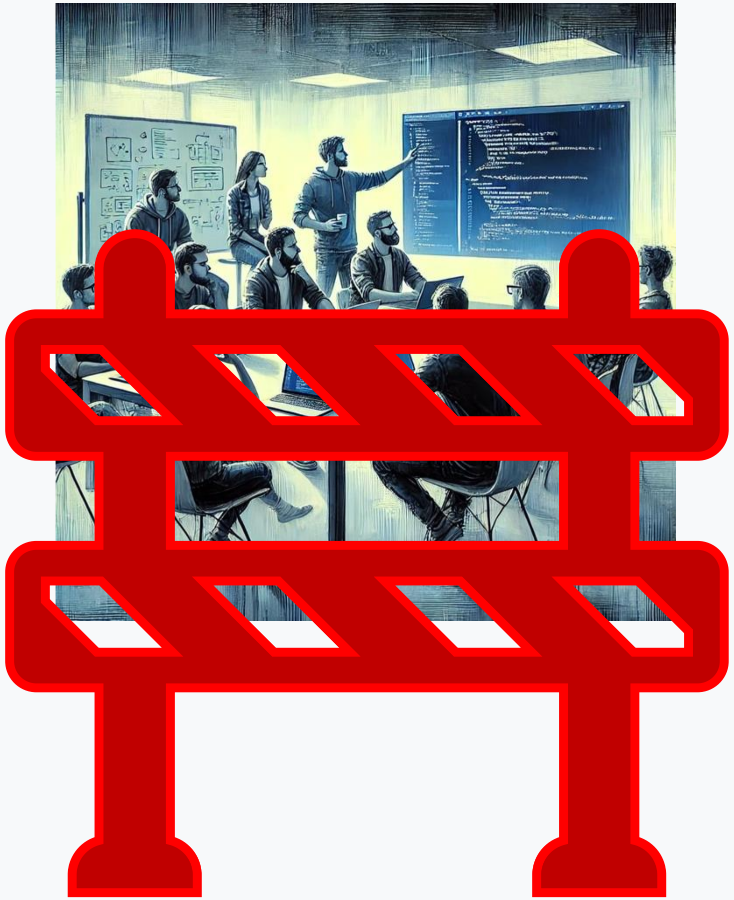
Contributors 2

tlestang Thibault Lestang

fcooper8472 Fergus Cooper

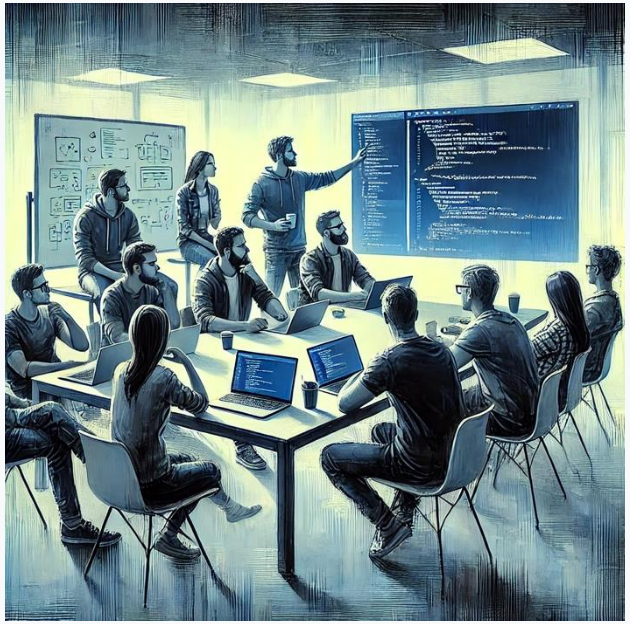
Barriers for code review

git



GitHub

Barriers for code review



Reviewing the code is more important

than the tools used to review

Personal experience of code review

- Didn’t use pull requests or GitHub issues…
- My code was on a shared GitHub repository
- New coding language to me - nextflow
- My colleague looked at the script I was working on
- We had regular meetings where I would walkthrough/explain the code
- Restructured logic of code execution, learnt new functions, etc.

Summary so far...

- Tools for code review balanced with reducing barriers

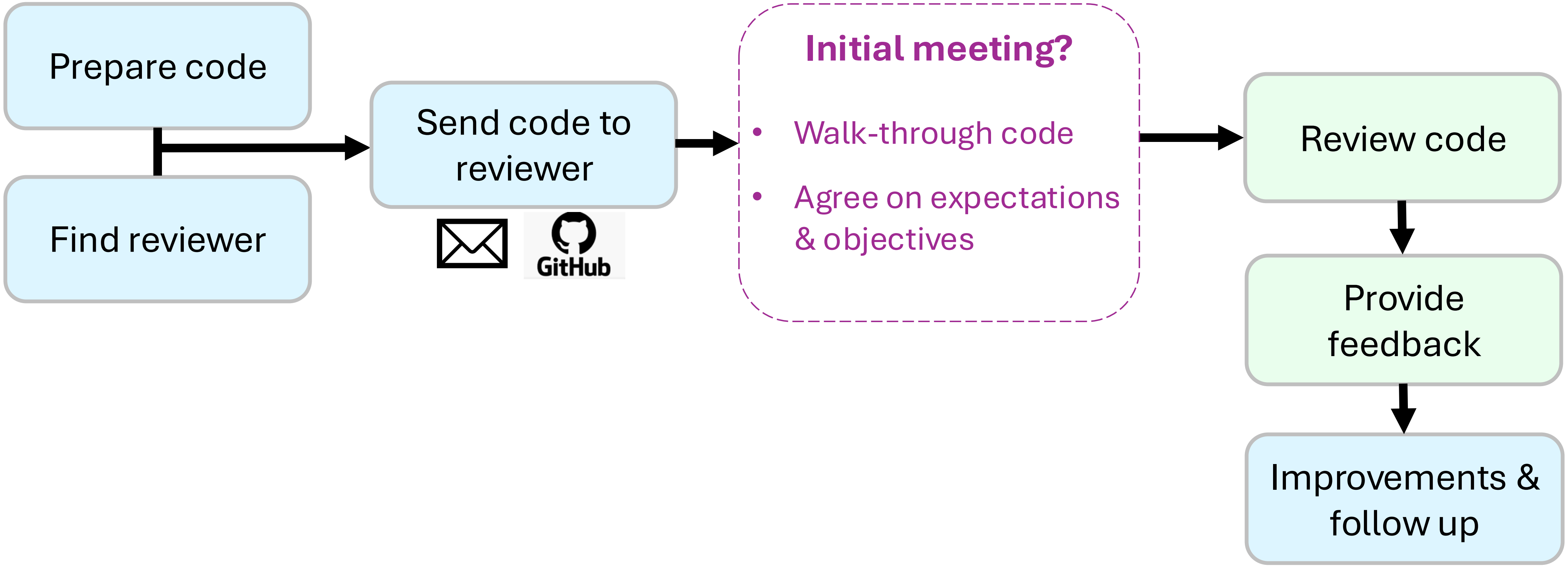
Next...

- Explain how a code review is done (role of author & reviewer)
- Things to look for in code when reviewing (checklist)

Overview - How to do a code review

Role of code author

Role of code reviewer



Prepare code

Find reviewer

Send code to reviewer

Initial meeting?

- Walk-through code
- Agree on expectations & objectives

Review code

Provide feedback

Improvements & follow up

Preparing code for review

Role of code author

- Identify **50 to 200 lines** of code you want reviewed
- Make it easy to review: include **comments and documentation**
- Decide the **focus of the review** in advance
- Ask for **specific feedback** if needed
- Be prepared to **walk through your code** in the logical order of execution

Finding a reviewer

Role of code author

- Any researcher that codes is a good candidate
- Any coding experience is useful
- Colleagues in your research group,
  collaborators,
  mentors,
  students…

How to do a code review

Role of code reviewer

How to do a code review

Role of code reviewer

- **Simply ask questions**
- Get the code author to explain their code in detail..

“Can you help me understand what this line of code does?”

- Leads to them finding bugs & areas of improvements themselves

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

How to do a code review

Role of code reviewer

- **Check code logic**
- Does the code do what is expected?

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

How to do a code review

Role of code reviewer

Be kind, give **personal opinions** rather than imperative statements

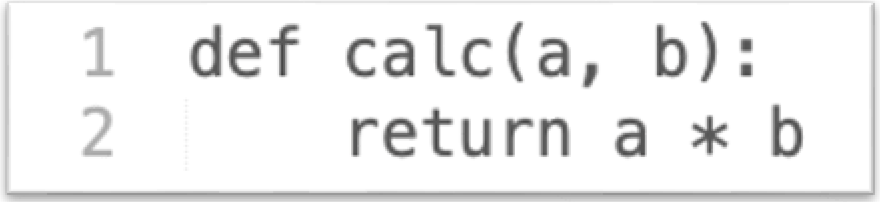
“I think this function’s name could be improved”
NOT
“You should rename this function”

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

Variable names

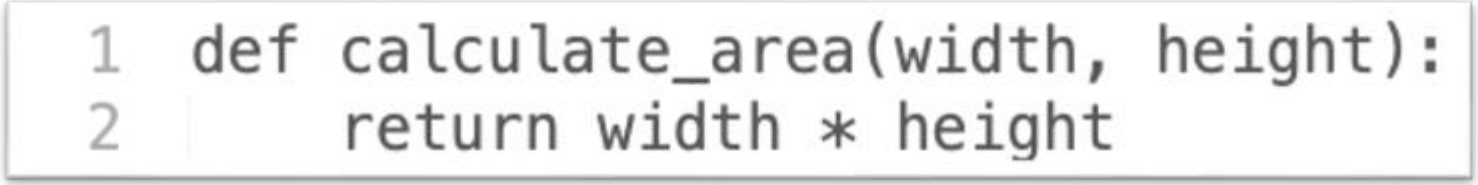
Use descriptive names that convey intent

👎



```
def calc(a, b):
    return a * b
```

👍



```
def calculate_area(width, height):
    return width * height
```

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

Hard coded values

Avoid “magic numbers”

👎

```
for (int i = 0; i < 26; i++) {
```

👍

```
alphabetLength = 26;
for (int i = 0; i < alphabetLength; i++) {
```

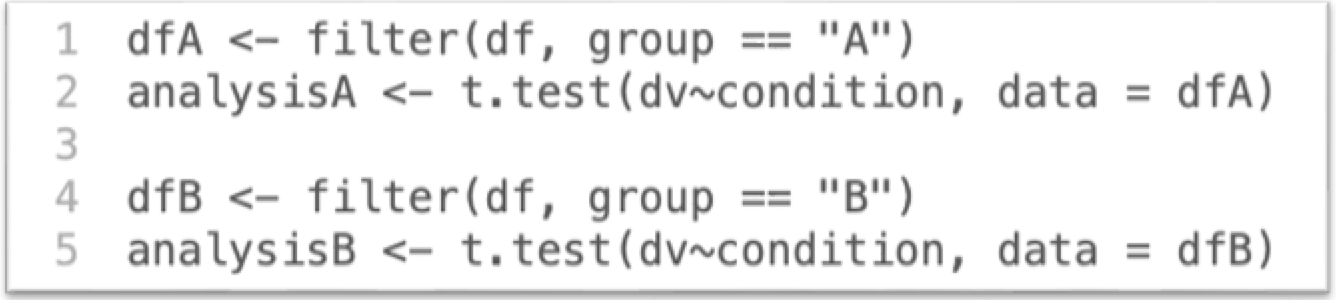
👍👍

```
alphabetLength = alphabetData.size();
for (int i = 0; i < alphabetLength; i++) {
```

Duplicated code

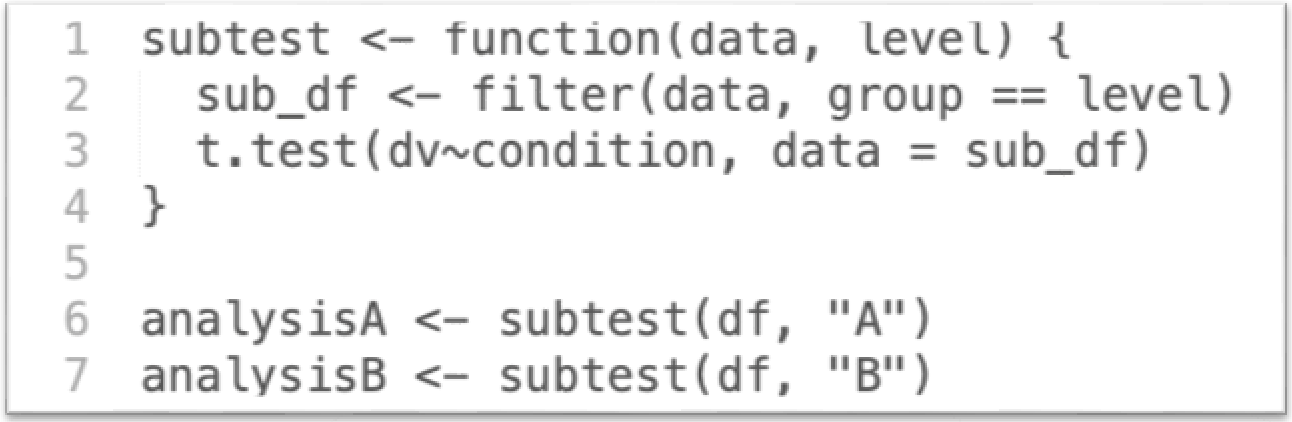
Don’t repeat yourself (DRY)

👎



```
dfA <- filter(df, group == "A")
analysisA <- t.test(dv~condition, data = dfA)

dfB <- filter(df, group == "B")
analysisB <- t.test(dv~condition, data = dfB)
```

👍

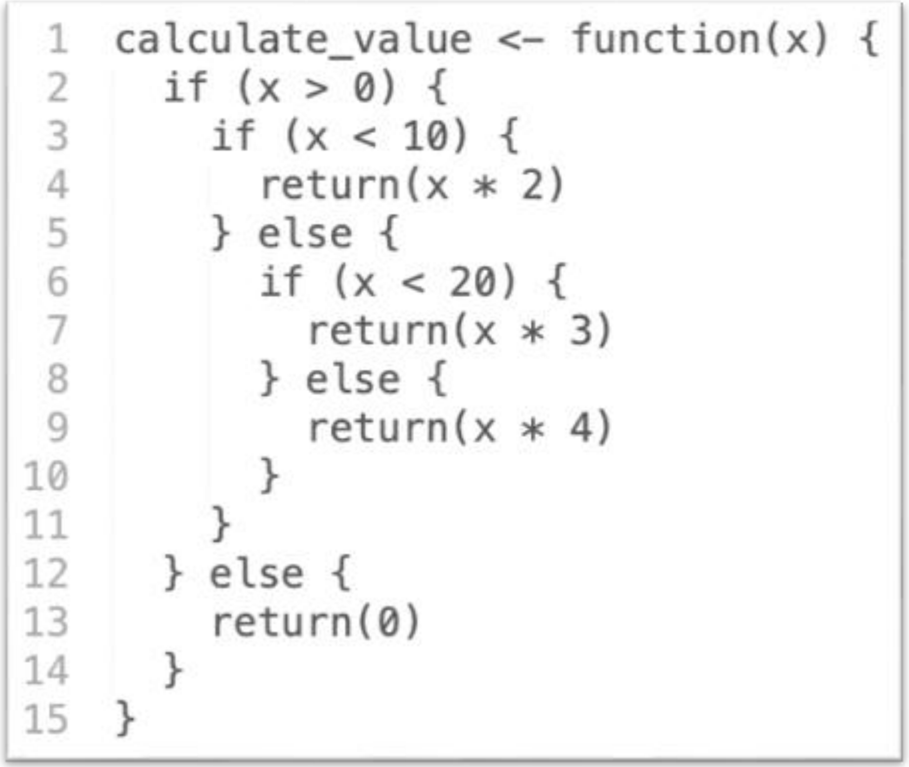


```
subtest <- function(data, level) {
  sub_df <- filter(data, group == level)
  t.test(dv~condition, data = sub_df)
}

analysisA <- subtest(df, "A")
analysisB <- subtest(df, "B")
```

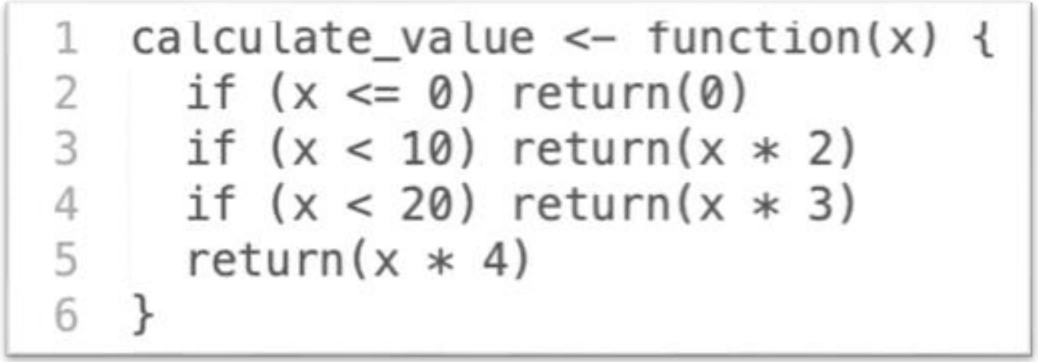
Slide adapted from materials: CC-BY: DeBruine, Lisa (2022) Intro to Code Review. *https://debruine.github.io/code-review/*

Complex if else statements

Flatten nested conditional statements with guard clauses



```
calculate_value <- function(x) {
  if (x > 0) {
    if (x < 10) {
      return(x * 2)
    } else {
      if (x < 20) {
        return(x * 3)
      } else {
        return(x * 4)
      }
    }
  } else {
    return(0)
  }
}
```



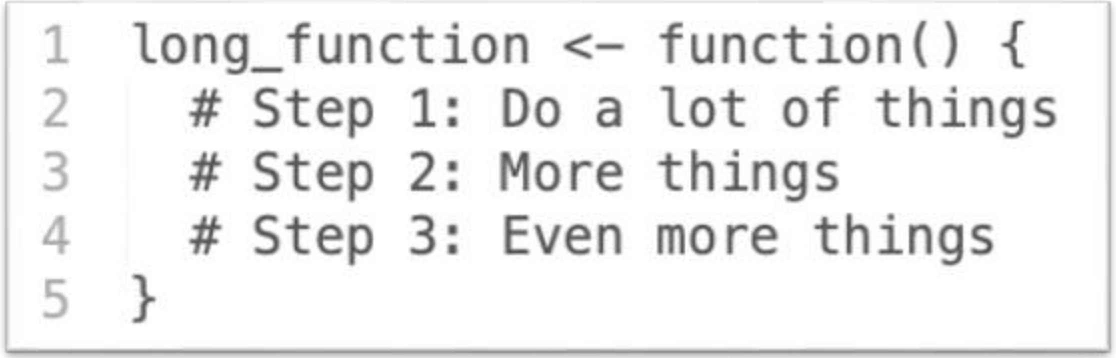
```
calculate_value <- function(x) {
  if (x <= 0) return(0)
  if (x < 10) return(x * 2)
  if (x < 20) return(x * 3)
  return(x * 4)
}
```

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

Long functions

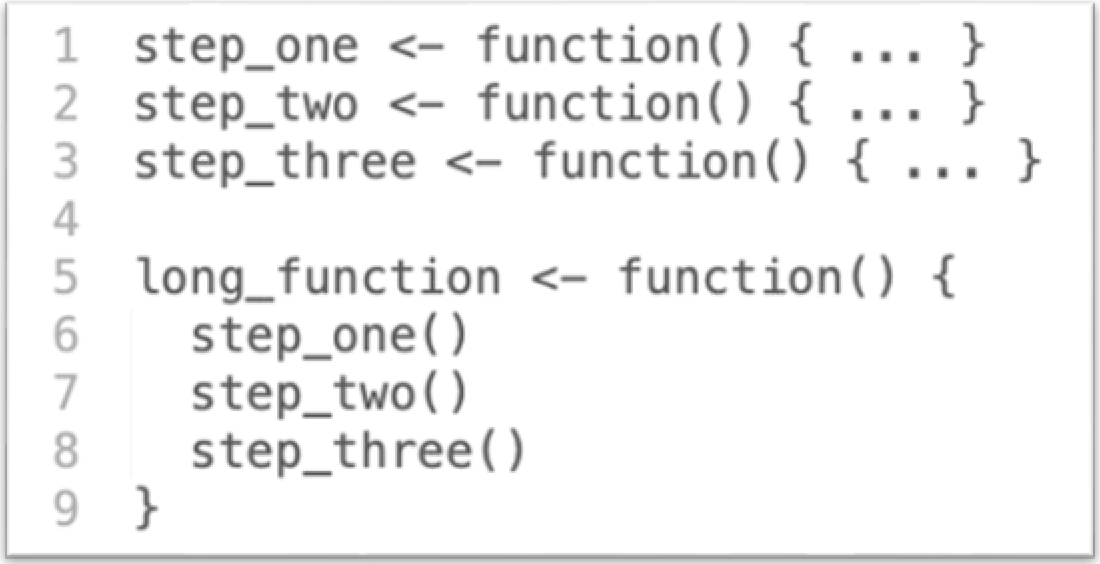
Functions should be short and do one thing

👎



```
long_function <- function() {
  # Step 1: Do a lot of things
  # Step 2: More things
  # Step 3: Even more things
}
```

👍



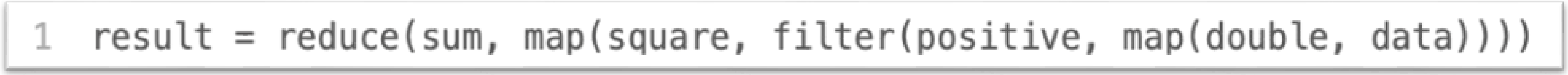
```
step_one <- function() { ... }
step_two <- function() { ... }
step_three <- function() { ... }

long_function <- function() {
  step_one()
  step_two()
  step_three()
}
```

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

Obscure lines

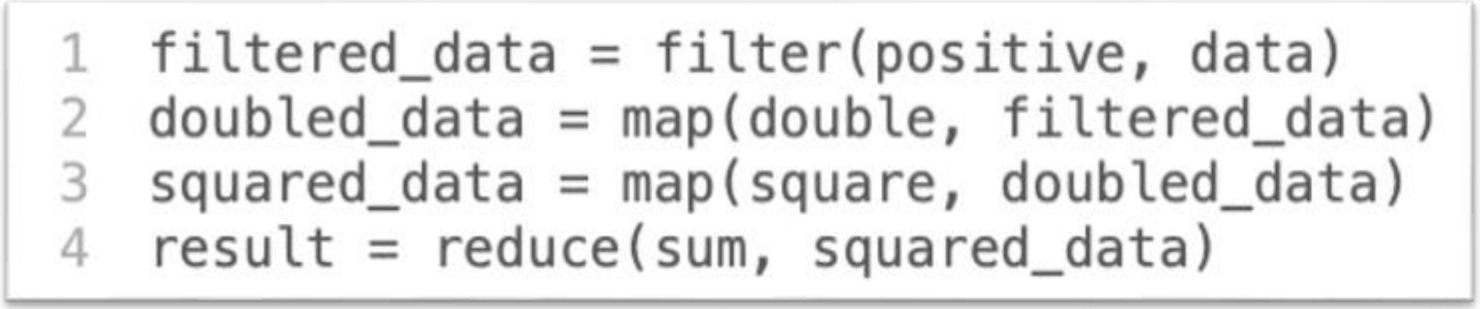
Resist clever one-liners

👎



```
result = reduce(sum, map(square, filter(positive, map(double, data))))
```

👍



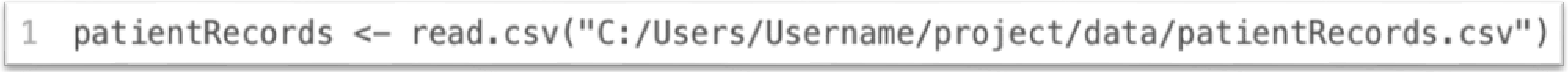
```
filtered_data = filter(positive, data)
doubled_data = map(double, filtered_data)
squared_data = map(square, doubled_data)
result = reduce(sum, squared_data)
```

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

File paths

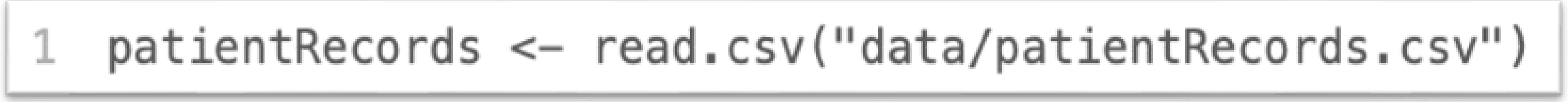
All file references should use relative paths, not absolute paths.

👎



```
patientRecords <- read.csv("C:/Users/Username/project/data/patientRecords.csv")
```

👍



```
patientRecords <- read.csv("data/patientRecords.csv")
```

“here” R package: https://here.r-lib.org/

Slide adapted from materials: CC-BY: DeBruine, Lisa (2022) Intro to Code Review. https://debruine.github.io/code-review/

File names

Name files so both people and computers can easily find things

👎

myabstract.docx
Joe’s Filenames Use Spaces and Punctuation.xlsx
figure 1.png
fig 2.png
JW7d^(2sl@deletethisandyourcareerisoverWx2*.txt

👍

2014-06-08_abstract-for-sla.docx
joes-filenames-are-getting-better.xlsx
fig01_scatterplot-talk-length-vs-interest.png
fig02_histogram-talk-attendance.png
1986-01-28_raw-data-from-challenger-o-rings.txt

three principles for (file) names

machine readable

human readable

plays well with default ordering

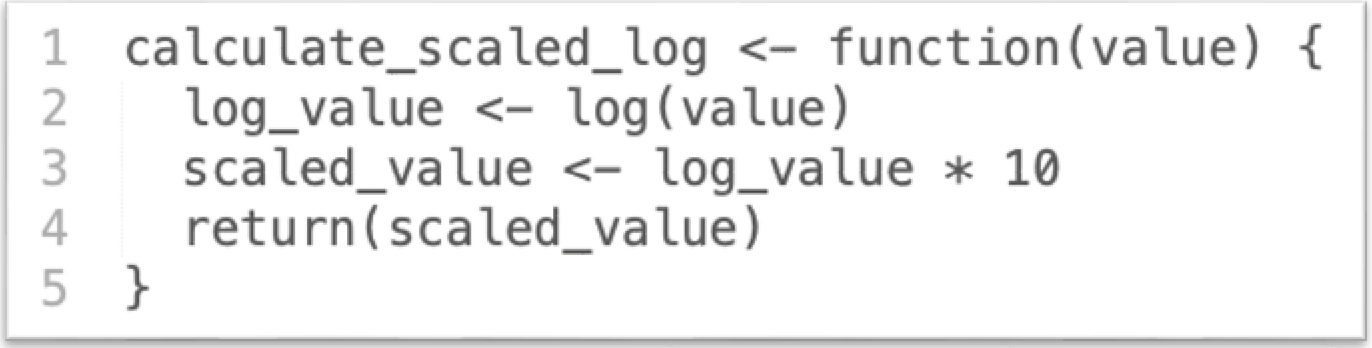
Use YYYY-MM-DD format for dates

Slide adapted from materials: CC-BY: DeBruine, Lisa (2022) Intro to Code Review. https://debruine.github.io/code-review/
https://speakerdeck.com/jennybc/how-to-name-files

Unintended behaviour

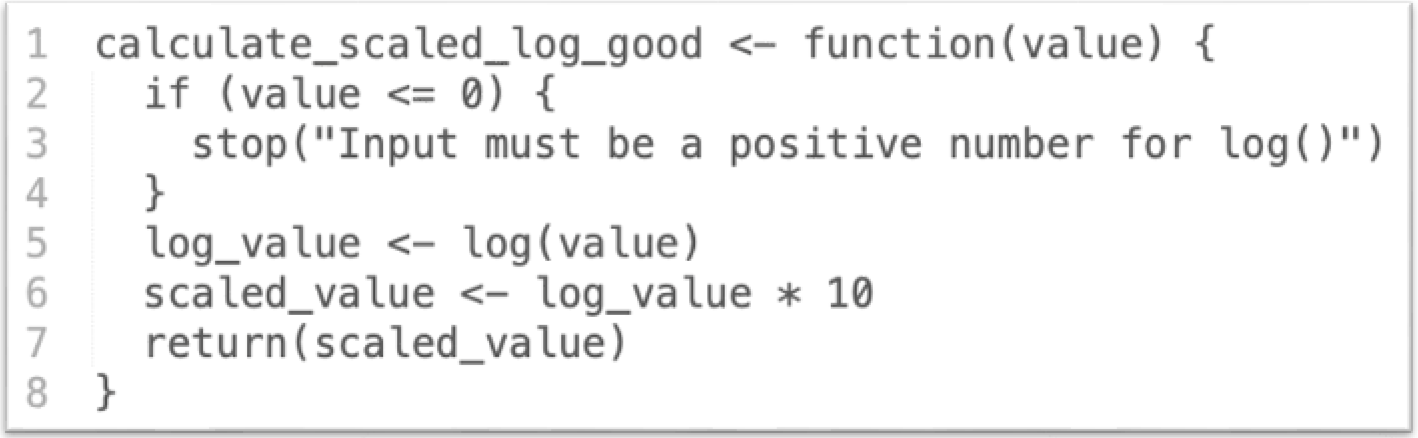
Validate inputs to prevent unintended behaviour or errors

👎



```r
calculate_scaled_log <- function(value) {
  log_value <- log(value)
  scaled_value <- log_value * 10
  return(scaled_value)
}
```

👍



```r
calculate_scaled_log_good <- function(value) {
  if (value <= 0) {
    stop("Input must be a positive number for log()")
  }
  log_value <- log(value)
  scaled_value <- log_value * 10
  return(scaled_value)
}
```

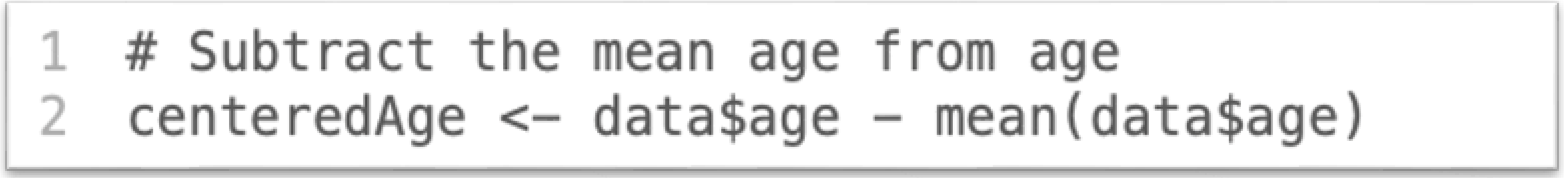
Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

Comments

Explain the “why” not the “what”

👎

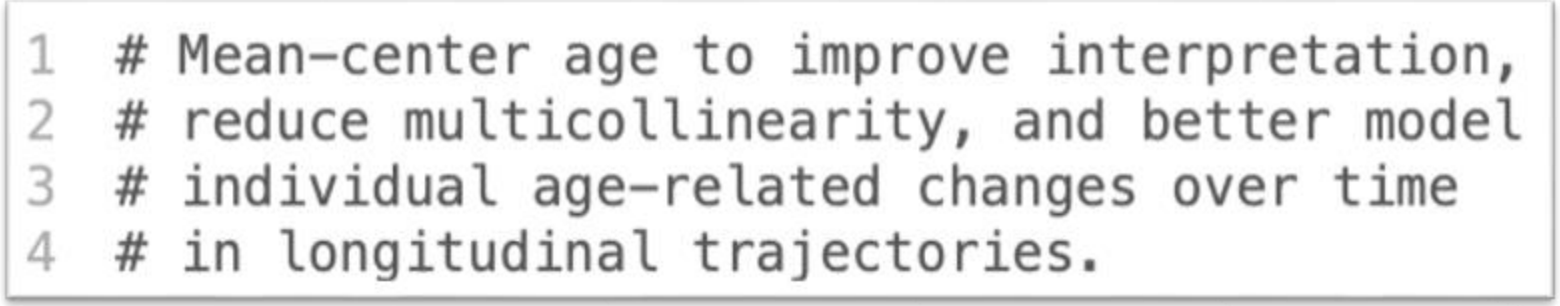
- Redundant comments
- Complicated comments



```
# Subtract the mean age from age
centeredAge <- data$age - mean(data$age)
```

👍

- Warnings of consequences
- Assumptions made

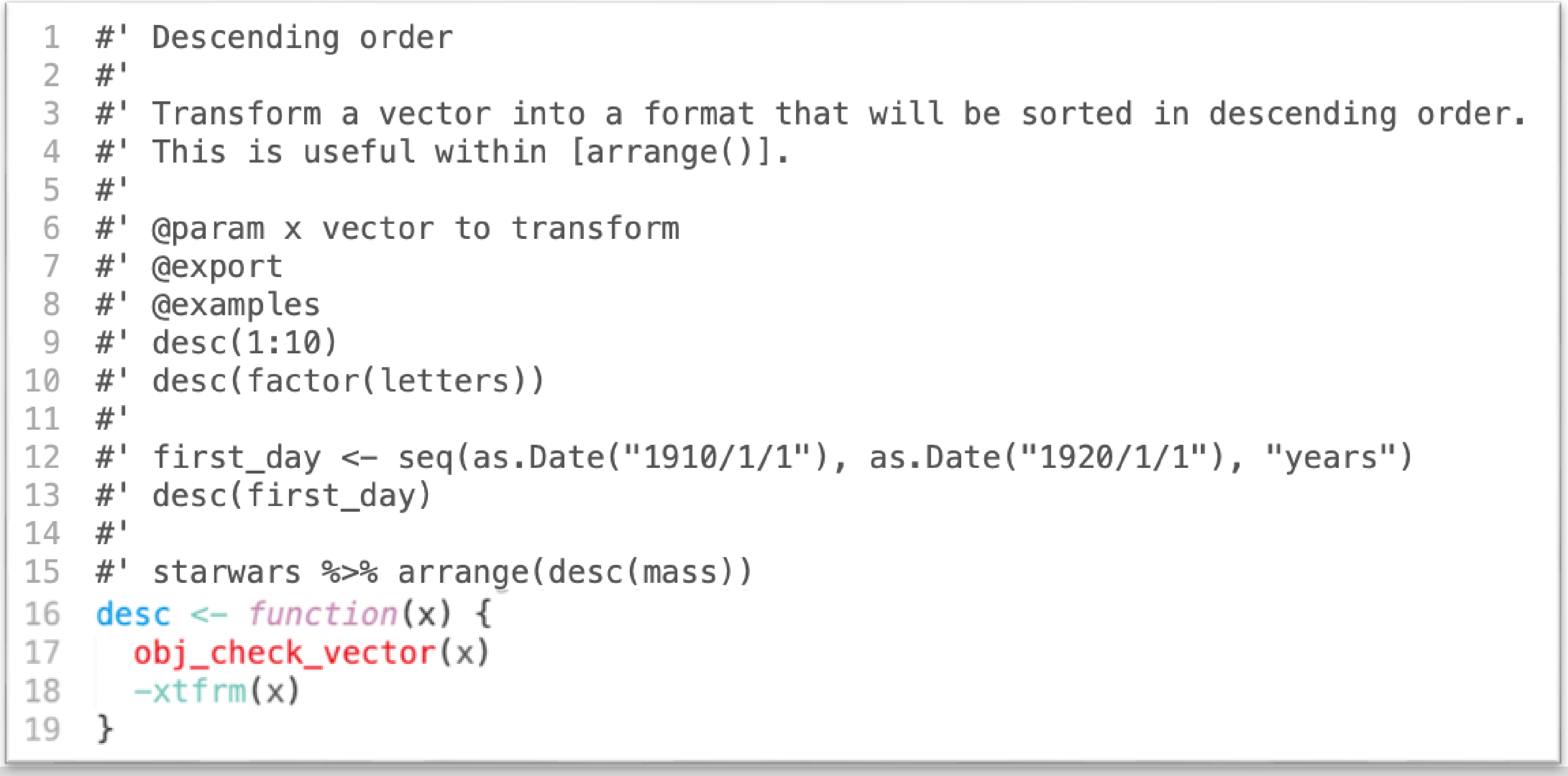


```
# Mean-center age to improve interpretation,
# reduce multicollinearity, and better model
# individual age-related changes over time
# in longitudinal trajectories.
```

Further info on writing good comments here: https://stackoverflow.blog/2021/12/23/best-practices-for-writing-code-comments/

Documented code

Functions and classes should contain docstrings



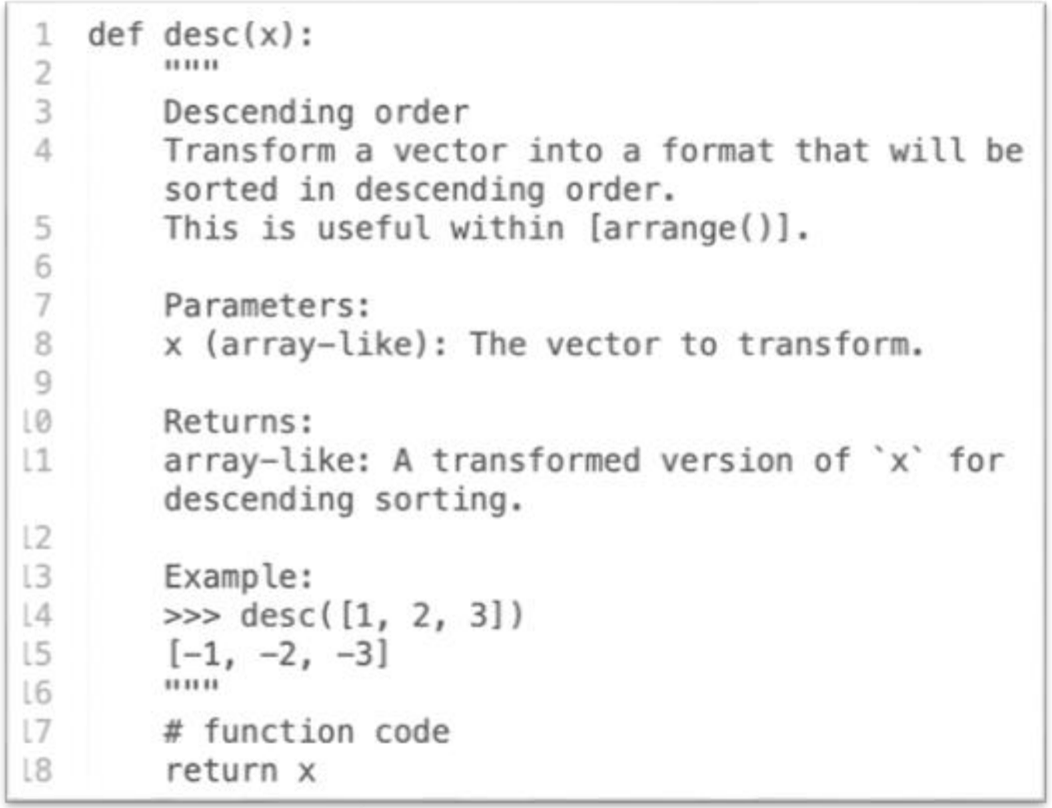
```r
#' Descending order
#'
#' Transform a vector into a format that will be sorted in descending order.
#' This is useful within [arrange()].
#'
#' @param x vector to transform
#' @export
#' @examples
#' desc(1:10)
#' desc(factor(letters))
#'
#' first_day <- seq(as.Date("1910/1/1"), as.Date("1920/1/1"), "years")
#' desc(first_day)
#'
#' starwars %>% arrange(desc(mass))
desc <- function(x) {
  obj_check_vector(x)
  -xtfrm(x)
}
```

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

Documented code

Functions and classes should contain docstrings

👍



```
def desc(x):
    """
    Descending order
    Transform a vector into a format that will be
    sorted in descending order.
    This is useful within [arrange()].

    Parameters:
    x (array-like): The vector to transform.

    Returns:
    array-like: A transformed version of `x` for
    descending sorting.

    Example:
    >>> desc([1, 2, 3])
    [-1, -2, -3]
    """
    # function code
    return x
```

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

Documented data

Be clear, consistent, and provide context

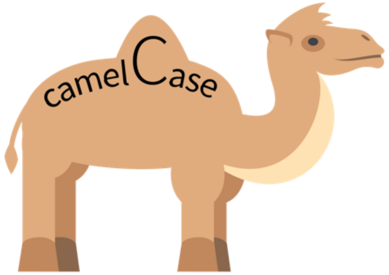
- Names (i.e., the column names)
- Labels/description
- Codings (e.g., 1 = always, 5 = never)
- Data type (e.g., binary, continuous)
- Descriptives (e.g., min, max)
- Data units (e.g., mg/L, months)
- Missing values (e.g., NA, 999)

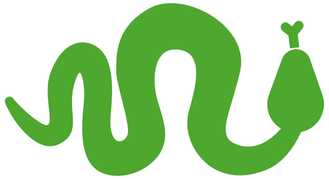
Slide adapted from materials: CC-BY: DeBruine, Lisa (2022) Intro to Code Review. *https://debruine.github.io/code-review/*

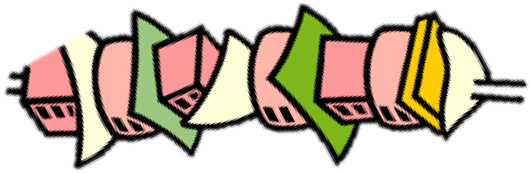
Coding Style

> “Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread.”

- Notation and naming




snake_case


kebab-case

- Syntax (spacing, indentations, line length)
- Commenting guidelines
- And more…

Google style guide on many languages: https://google.github.io/styleguide/
R style guide: https://style.tidyverse.org/

Slide adapted from materials: https://github.com/OxfordCodeReviewNet/forum/blob/master/guidelines_for_reviewers.md

Automate what can be automated

Code author can use a linter to automate coding style checks before review

- **R –** lintr R package

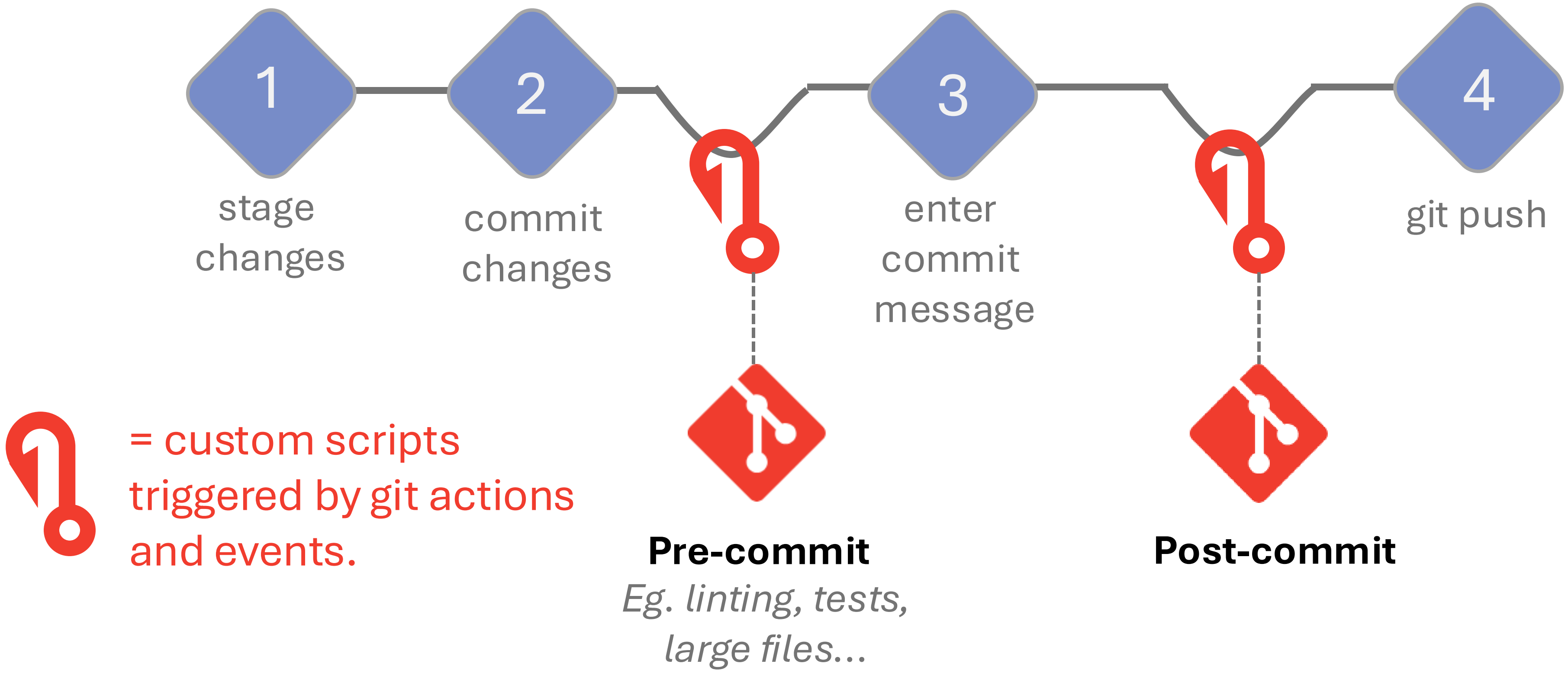
“lintr provides static code analysis for R. It checks for adherence to a given style, identifying syntax errors and possible semantic issues, then reports them to you so you can take action.”

- **Python** - list of linters:

https://github.com/vintasoftware/python-linters-and-code-analysis

Using git hooks and pre-commits

Code author can use automated pre-commit checks before review



= custom scripts triggered by git actions and events.

Pre-commit

Eg. linting, tests, large files...

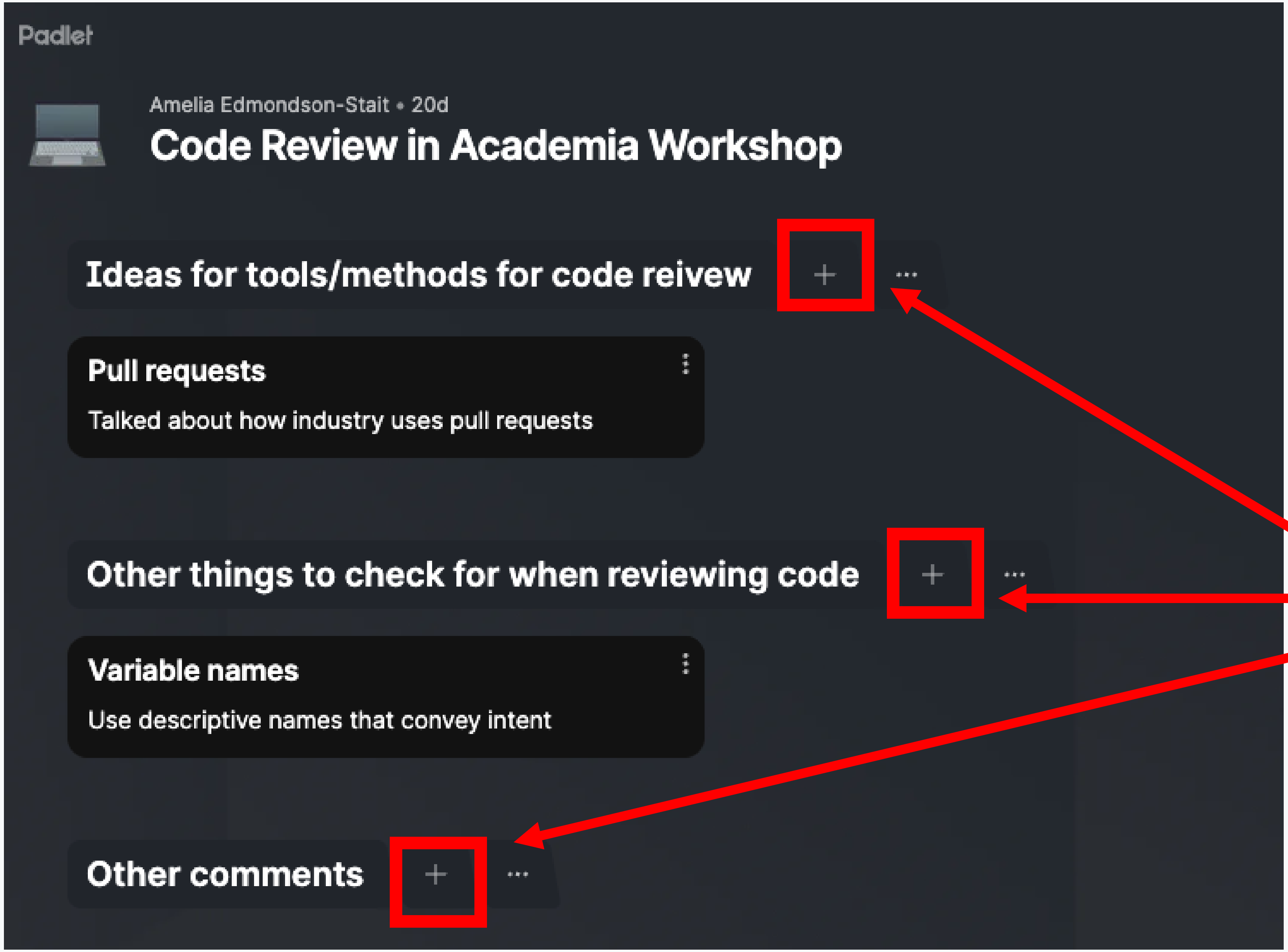
Post-commit

https://git-scm.com/book/ms/v2/Customizing-Git-Git-Hooks
https://pre-commit.com/

Summary of checklist

- Variable names
- Hard coded values/magic numbers
- Duplicated code
- Complex if else statements
- Long functions
- Obscure lines
- File paths
- File names
- Unintended behavior
- Comments
- Documented code
- Documented data
- Coding style (with automated checks using linter/git hooks)

Other things for a reviewer to check?



Amelia Edmondson-Stait • 20d

Code Review in Academia Workshop

Ideas for tools/methods for code reivew

Pull requests

Talked about how industry uses pull requests

Other things to check for when reviewing code

Variable names

Use descriptive names that convey intent

Other comments

Please add your comments and ideas