Lessons learned from running a local RSE group: research-software.uit.no

Radovan Bast, UiT The Arctic University of Norway

Nordic RSE conference 2024

Not about

- Why research software engineering is important
- Why we thought that our university should have an RSE group
- Why every university should have an RSE group

D'oh!

This talk is about

- How we started
- Growing pains
- What we know now that we wish we knew earlier
- Most common problems we see

How we started: research-software.uit.no

1. Pitch the idea to the boss
2. Create a website
3. Get a nice subdomain
4. Talk about it and repeat yourself
5. Send an email to all department with "cake" in the subject
6. Invite yourself to other department seminars
7. Infoscreens

Our workflow

- Weekly office hours (2 hours/week).
- Attract projects ranging from 1 hour to days.
- Track projects:

```
name: Mathematica on Fram
contact: Jane Doe <jane.doe@example.org>
department: Department of Mathematics and Statistics
staff: Staff who worked on it
summary: |
  A sentence/paragraph summarizing what this is/was about.
status: finished
```

  - For each project write at least a sentence on the website.
  - For larger projects write a blog post.
  - "If it isn't documented, it didn't happen".

Our workflow

- Selecting projects: Currently we say "yes" to everything that we know we can solve.
- Status: 40 projects. Roughly 1 new project per office hour.

Growing pains

- Started with only a personal email and then lost overview -> Create a **group email address** and connect it to an issue tracker.

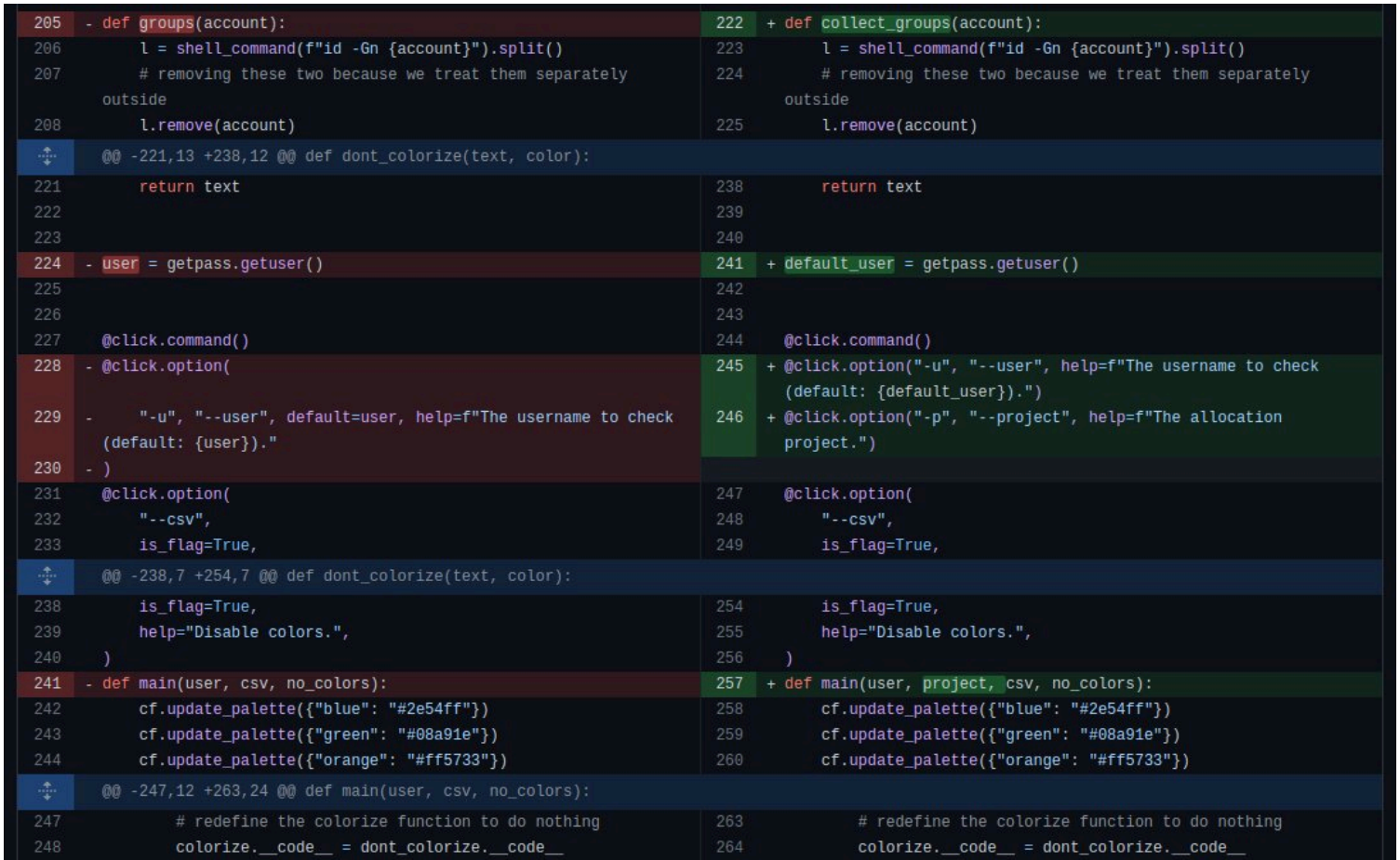
- **Too many cooks**: Student asking and 3 staff answering which can feel like a cross-examination -> Let one staff lead the discussion and help even when you feel like you know a "better" answer.

- Where do we **track projects** that take longer than an afternoon?

- Office hours work well. But should we do office hours **in-person or online**?

Help with improving your scripts/code

- Code review (we discuss code in a constructive way)
- Making code more reusable
- Good practices for documentation



```
205 - def groups(account):
206       l = shell_command(f"id -Gn {account}").split()
207       # removing these two because we treat them separately
    outside
208       l.remove(account)
    @@ -221,13 +238,12 @@ def dont_colorize(text, color):
221       return text
222
223
224 - user = getpass.getuser()
225
226
227   @click.command()
228 - @click.option(
229 -     "-u", "--user", default=user, help=f"The username to check
    (default: {user})."
230 - )
231   @click.option(
232       "--csv",
233       is_flag=True,
    @@ -238,7 +254,7 @@ def dont_colorize(text, color):
238       is_flag=True,
239       help="Disable colors.",
240   )
241 - def main(user, csv, no_colors):
242       cf.update_palette({"blue": "#2e54ff"})
243       cf.update_palette({"green": "#08a91e"})
244       cf.update_palette({"orange": "#ff5733"})
    @@ -247,12 +263,24 @@ def main(user, csv, no_colors):
247           # redefine the colorize function to do nothing
248           colorize.__code__ = dont_colorize.__code__
```

```
222 + def collect_groups(account):
223       l = shell_command(f"id -Gn {account}").split()
224       # removing these two because we treat them separately
    outside
225       l.remove(account)
238       return text
239
240
241 + default_user = getpass.getuser()
242
243
244   @click.command()
245 + @click.option("-u", "--user", help=f"The username to check
    (default: {default_user}).")
246 + @click.option("-p", "--project", help=f"The allocation
    project.")
247   @click.option(
248       "--csv",
249       is_flag=True,
254       is_flag=True,
255       help="Disable colors.",
256   )
257 + def main(user, project, csv, no_colors):
258       cf.update_palette({"blue": "#2e54ff"})
259       cf.update_palette({"green": "#08a91e"})
260       cf.update_palette({"orange": "#ff5733"})
263           # redefine the colorize function to do nothing
264           colorize.__code__ = dont_colorize.__code__
```

What we know now that we wish we knew earlier

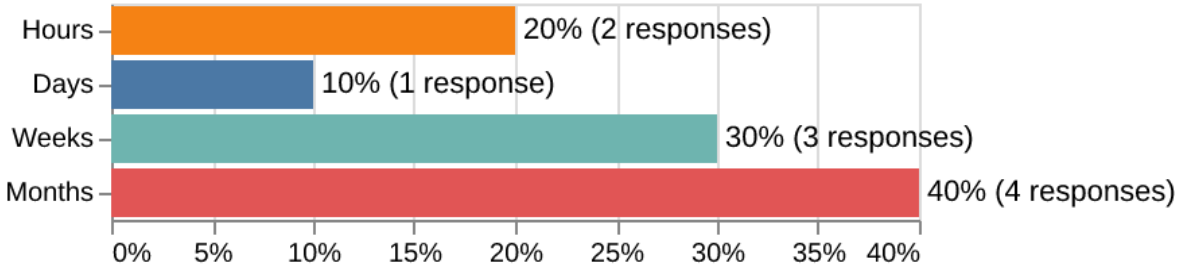
What we know now that we wish we knew earlier

- We expected more questions about code review and code structure.
- Surprisingly many questions about statistics and AI model choices and "niche" libraries.
- We can see that people are happy but we need to measure it.
- Document as you go (like when you visit your general practitioner).
- Students and researchers love it but university "has no budget for it".
- It helps to have sponsors high up in the administration. Spend time on explaining it well.

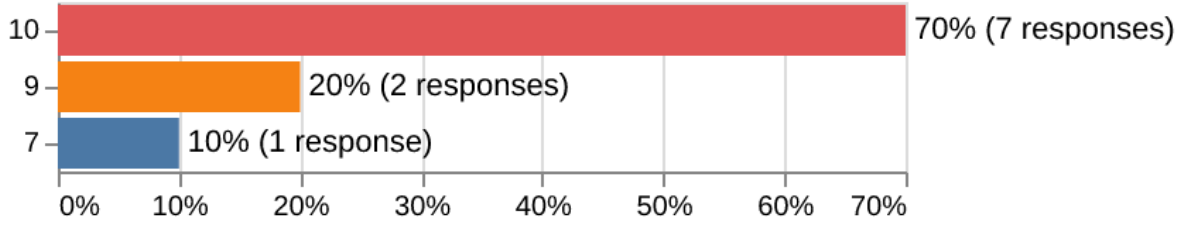
Most common problems we see

- Code is a 7k lines long notebook (Jupyter or R Markdown)
- Not reusable: Dependencies not documented. No functions. Example only runs by running code sections in a specific order.
- Hard-coded paths. No command-line interface.
- Execution takes a long time and/or consumes lots of memory.
- Lengthy independent work chunks are computed in a loop.
- Code repetition.
- It takes us a lot of effort to create a minimal working example.

Survey results



In your estimate, how much time have you saved as a result of working with us?



How likely is it that you would recommend our support to a friend or colleague?

0 means definitely not. 10 means definitely yes.