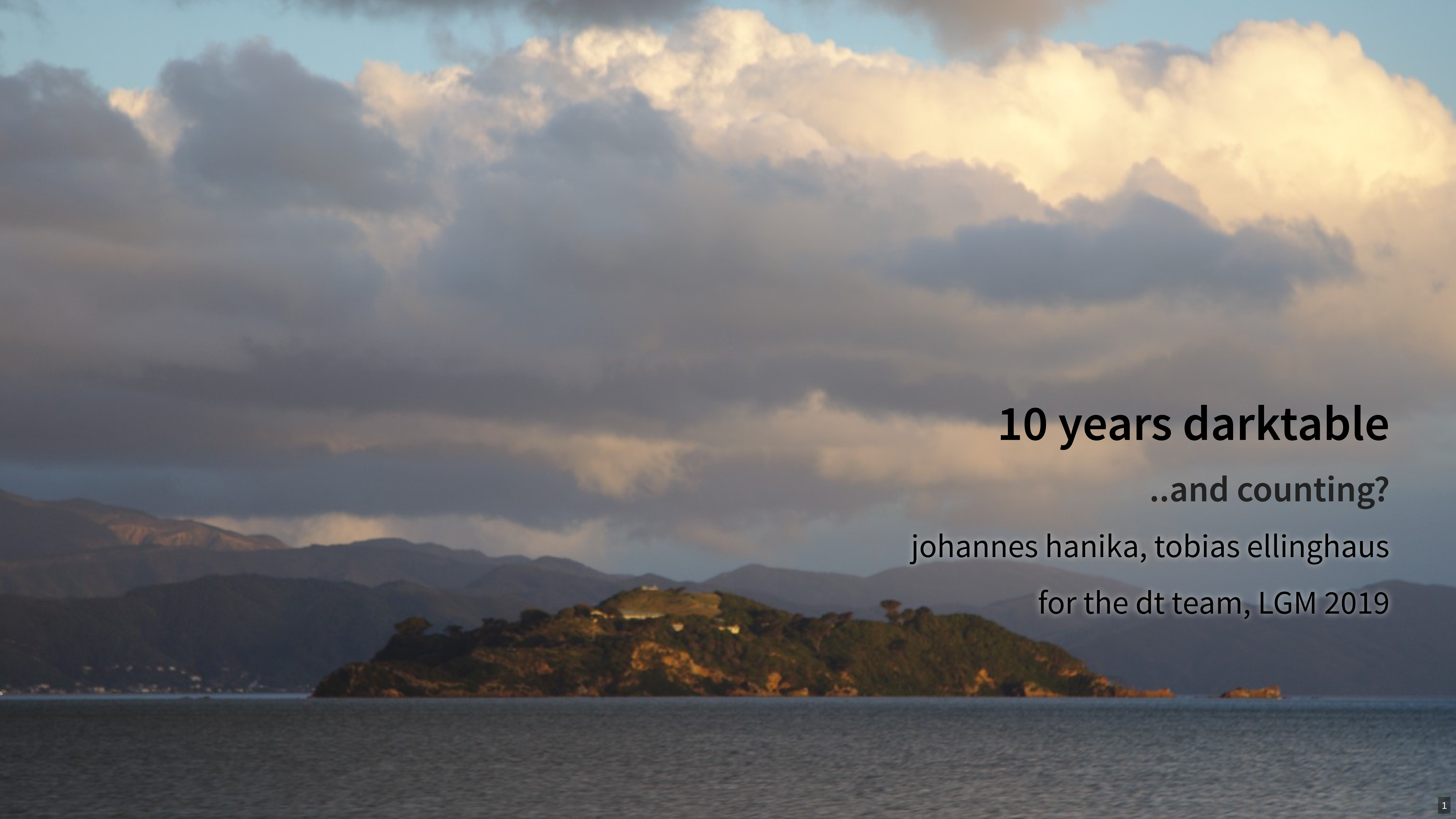


10 years darktable

..and counting?

johannes hanika, tobias ellinghaus

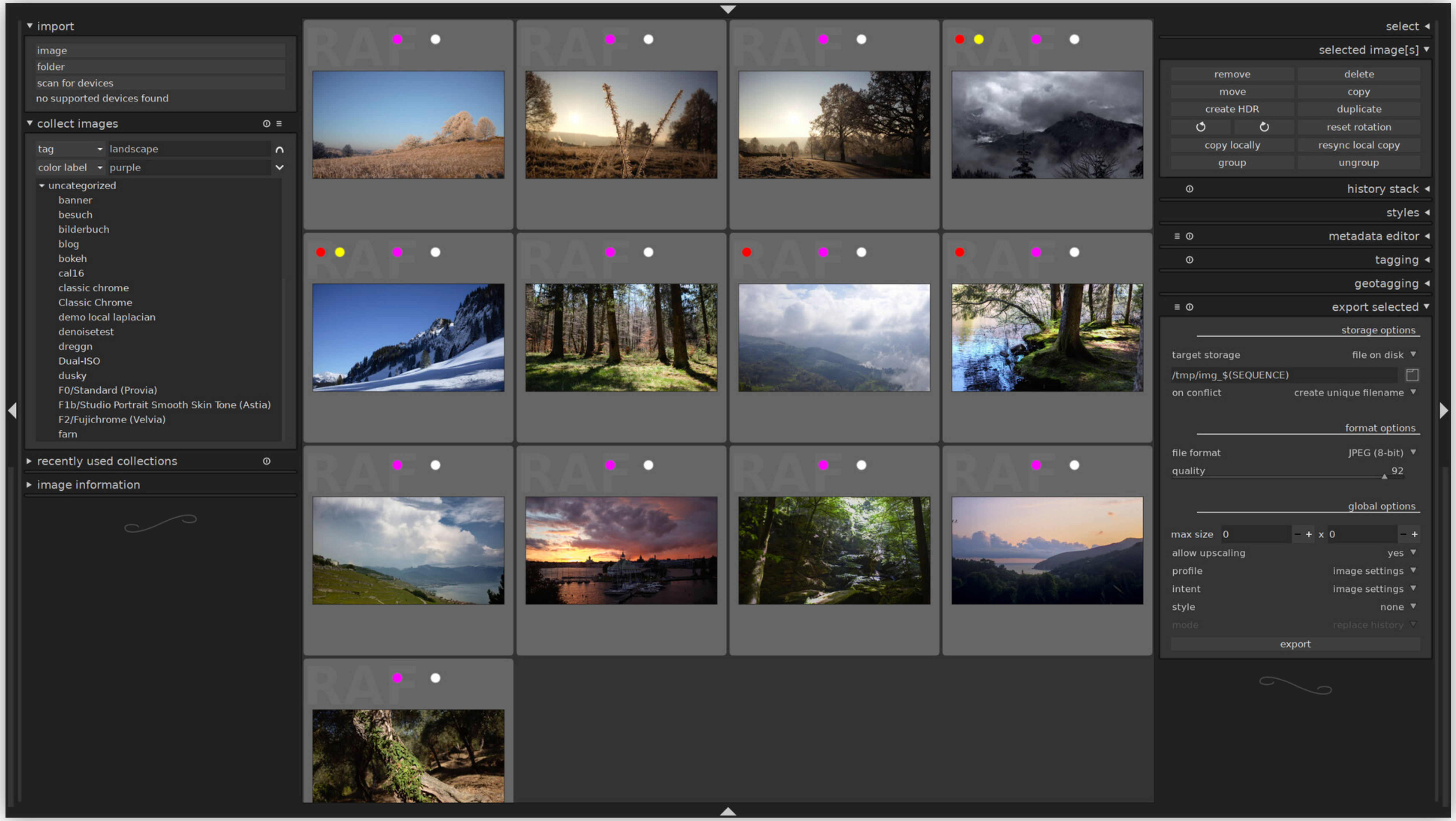
for the dt team, LGM 2019

contents of this talk

- introduction what is dt
- historic overview
- organisational aspects:
  - how did we (or not) manage dt as a project
  - how did that go for us the last 10 years?

what is darktable?

- digital darkroom and **lighttable**

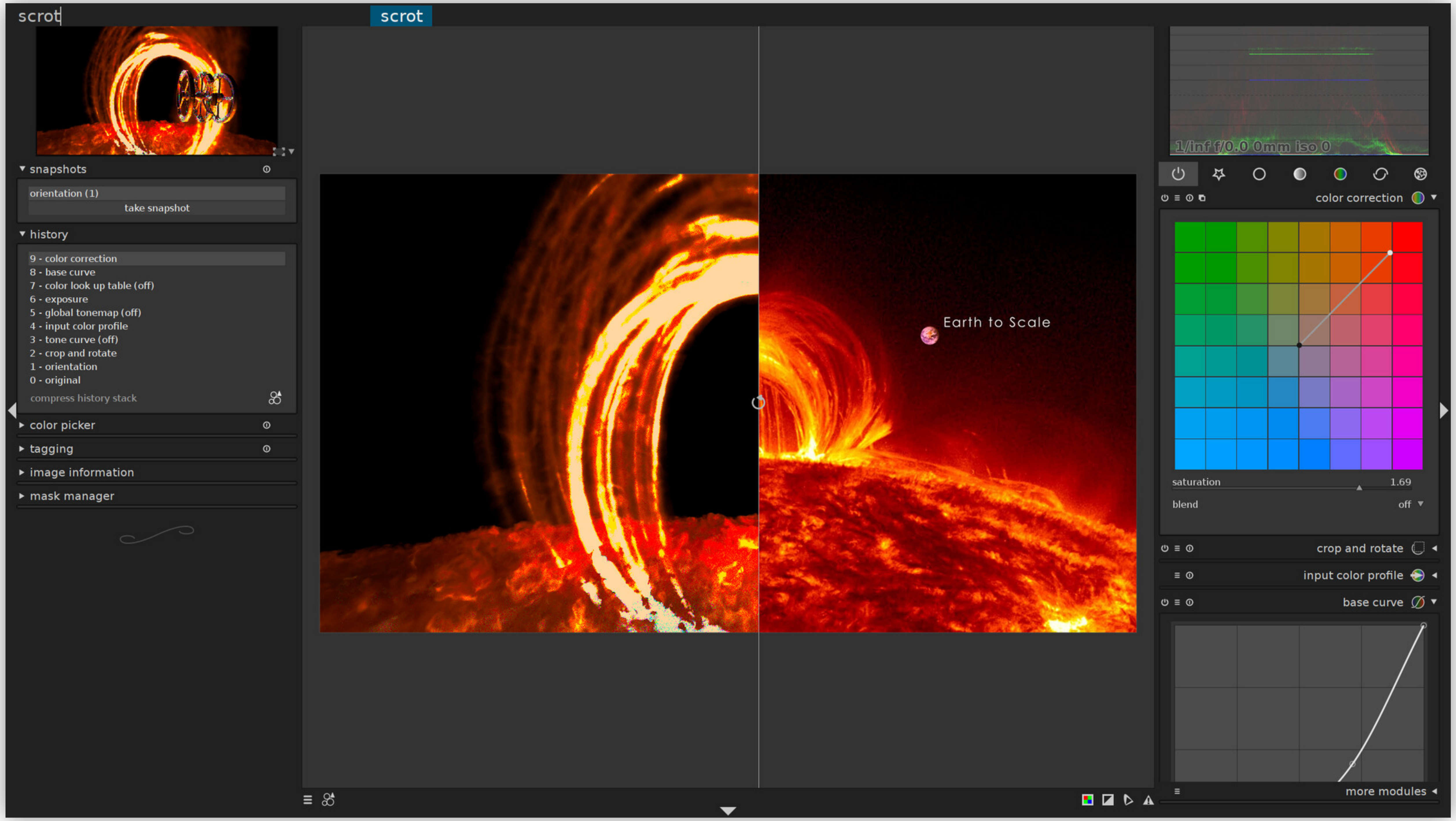


tag landscape
color label purple

target storage: file on disk
/tmp/img_$(SEQUENCE)
on conflict: create unique filename
file format: JPEG (8-bit)
quality: 92
max size: 0 x 0
allow upscaling: yes
profile: image settings
intent: image settings
style: none
mode: replace history

what is darktable?

- digital **darkroom** and lighttable



orientation (1)

9 - color correction
8 - base curve
7 - color look up table (off)
6 - exposure
5 - global tonemap (off)
4 - input color profile
3 - tone curve (off)
2 - crop and rotate
1 - orientation
0 - original

compress history stack

saturation 1.69

blend off

what is darktable?

key features:

- non-destructive photography workflow
  - long list of image operation modules for raw edits
  - local edits via parametric and drawn masks
  - GPU accelerated via opencl
- manage medium size collections of images (10s of 1000s)
- printing mode
- lua scripting
- extensible modular architecture: many plugins for this and that
- ..

what is darktable?

community!

historic overview

- first commit

```
commit f52987d28426f6da69907bdd190d2c379ce9077d
Author: jhanika <jhanika@2165daf7-ed08-4abd-b3da-b2de91a31a1b>
Tue Apr 7 07:13:23 2009 +0000
```

- almost exactly 10 years ago!

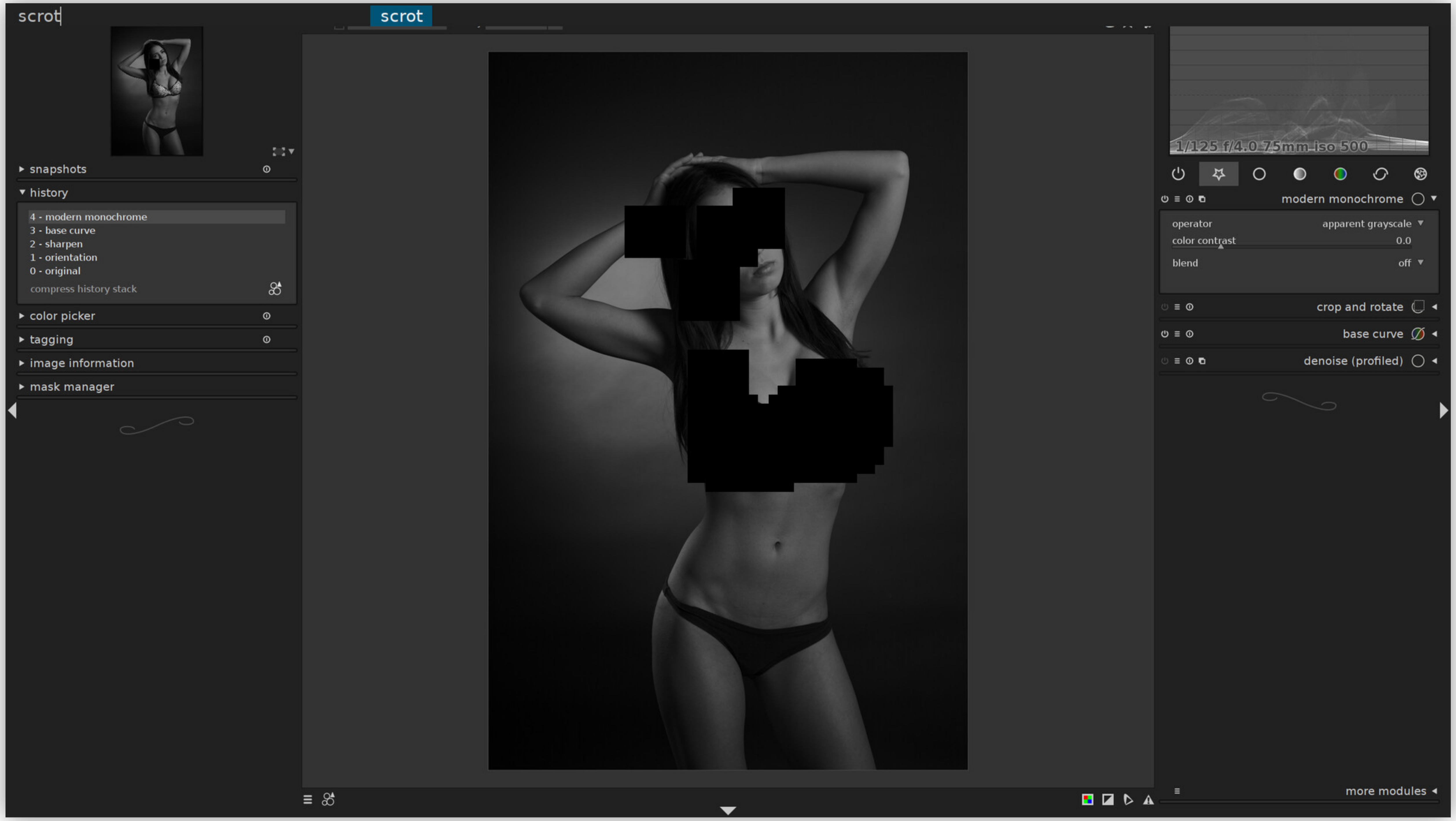
historic overview:

- lots of debugging of broken images

- (broken addressing scheme in laplace pyramids)

historic overview:

- lots of debugging of broken images



scrot

history

4 - modern monochrome
3 - base curve
2 - sharpen
1 - orientation
0 - original

compress history stack

1/125 f/4.0 75mm iso 500

modern monochrome

operator: apparent grayscale
color contrast: 0.0
blend: off

- builtin AI based cencorship (NaN propagate..)

historic overview:

- large, ever changing group of active contributors
- some datamining:
- **who commits code?**
  - many people with direct commit rights (24)
  - we run an *author credits* script before every release

historic overview:

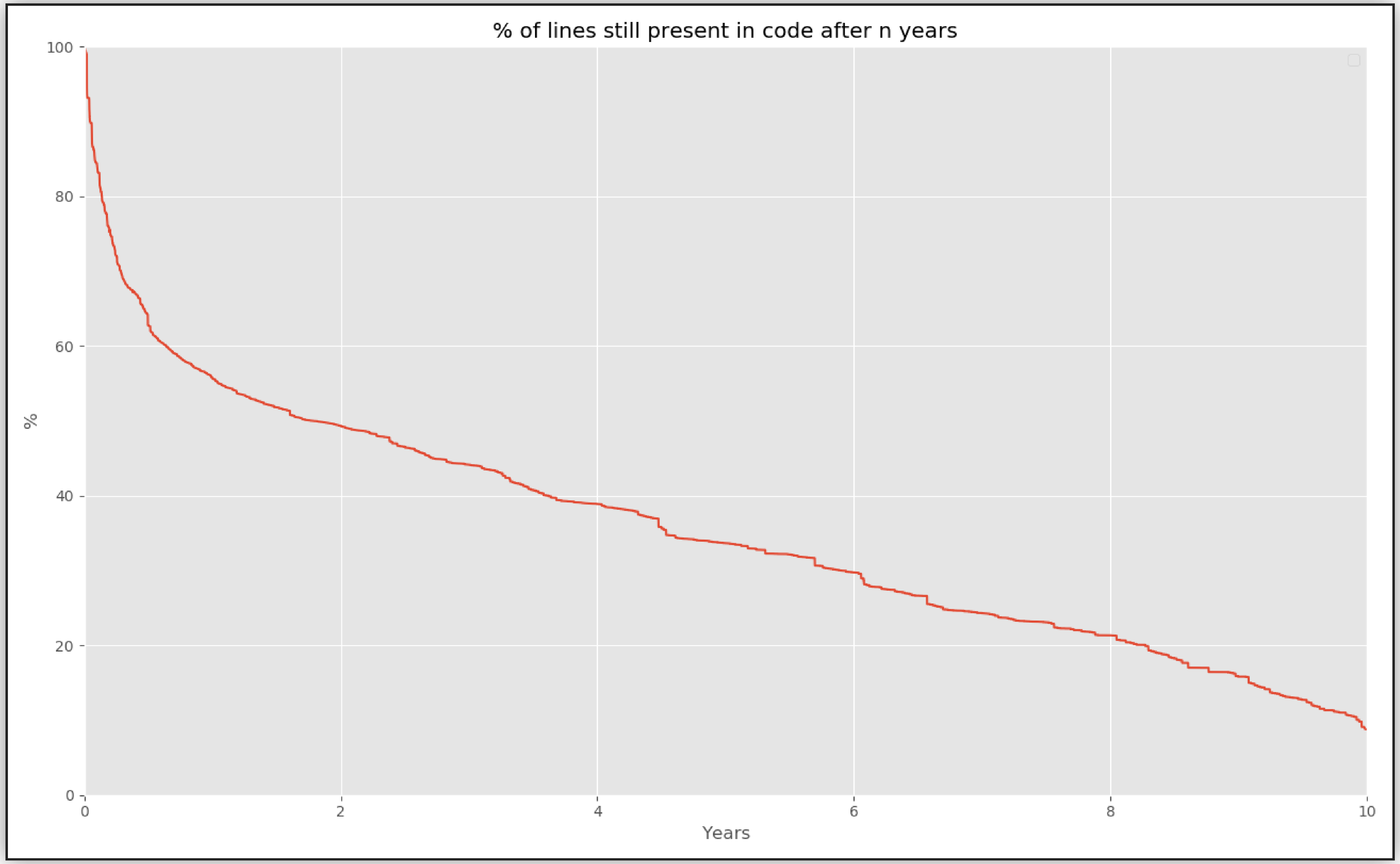
commits are nice, but.

- who actually owns the lines of code? do they remain? are they replaced?
- git blame!

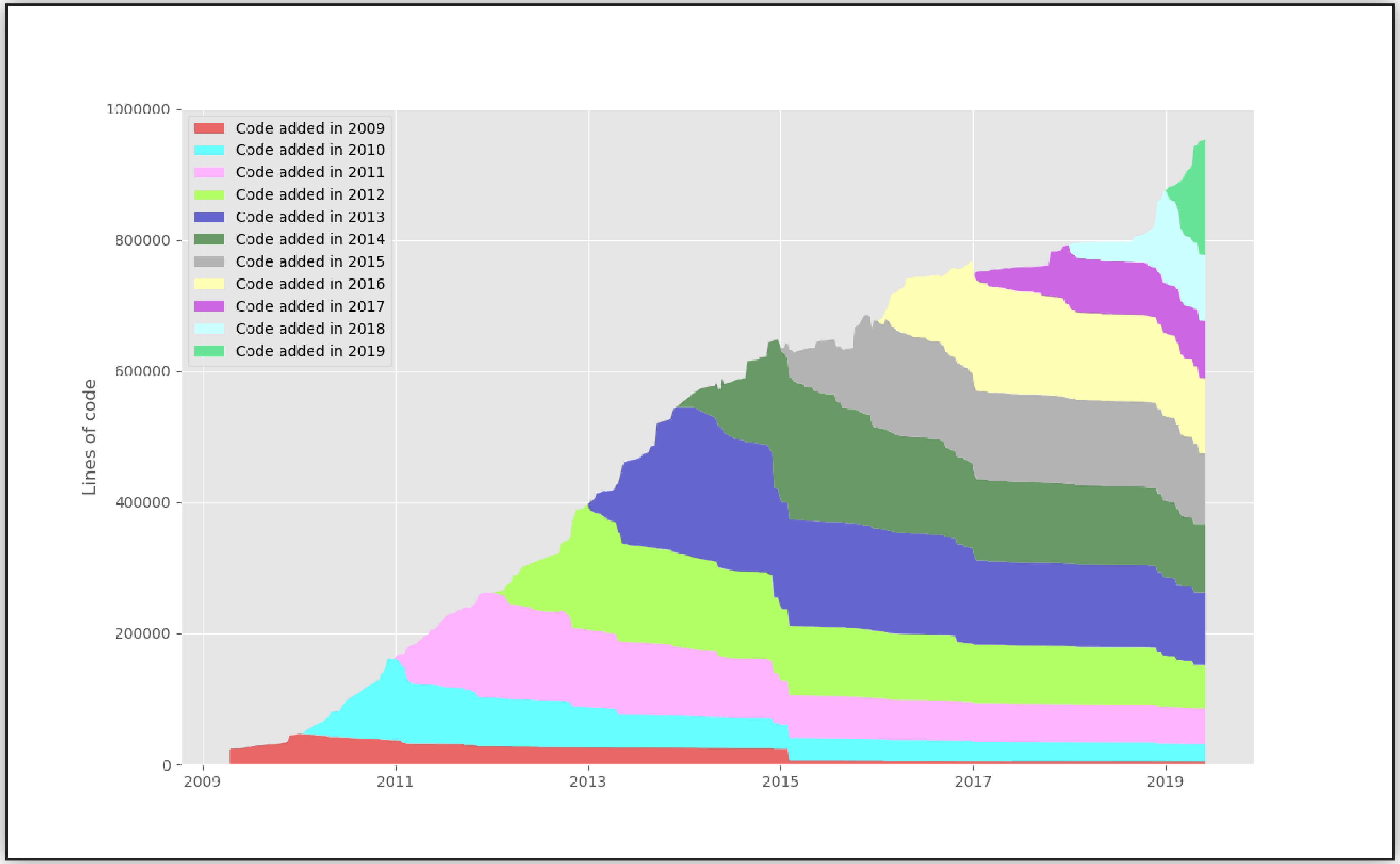
```bash
#!/bin/bash
for year in $(seq 2009 1 2019)
do
for month in 03 06 09 12
do
  qrev=$(git rev-list -n 1 --first-parent --before="$year-$month-01 13:37" master)
  git ls-tree -r -z --name-only $qrev -- src/*/*.{h,c} data/kernels/*.cl| xargs -0 -n1 git blame --li
done
done
```

- .. actually pippin pointed out a nicer way:

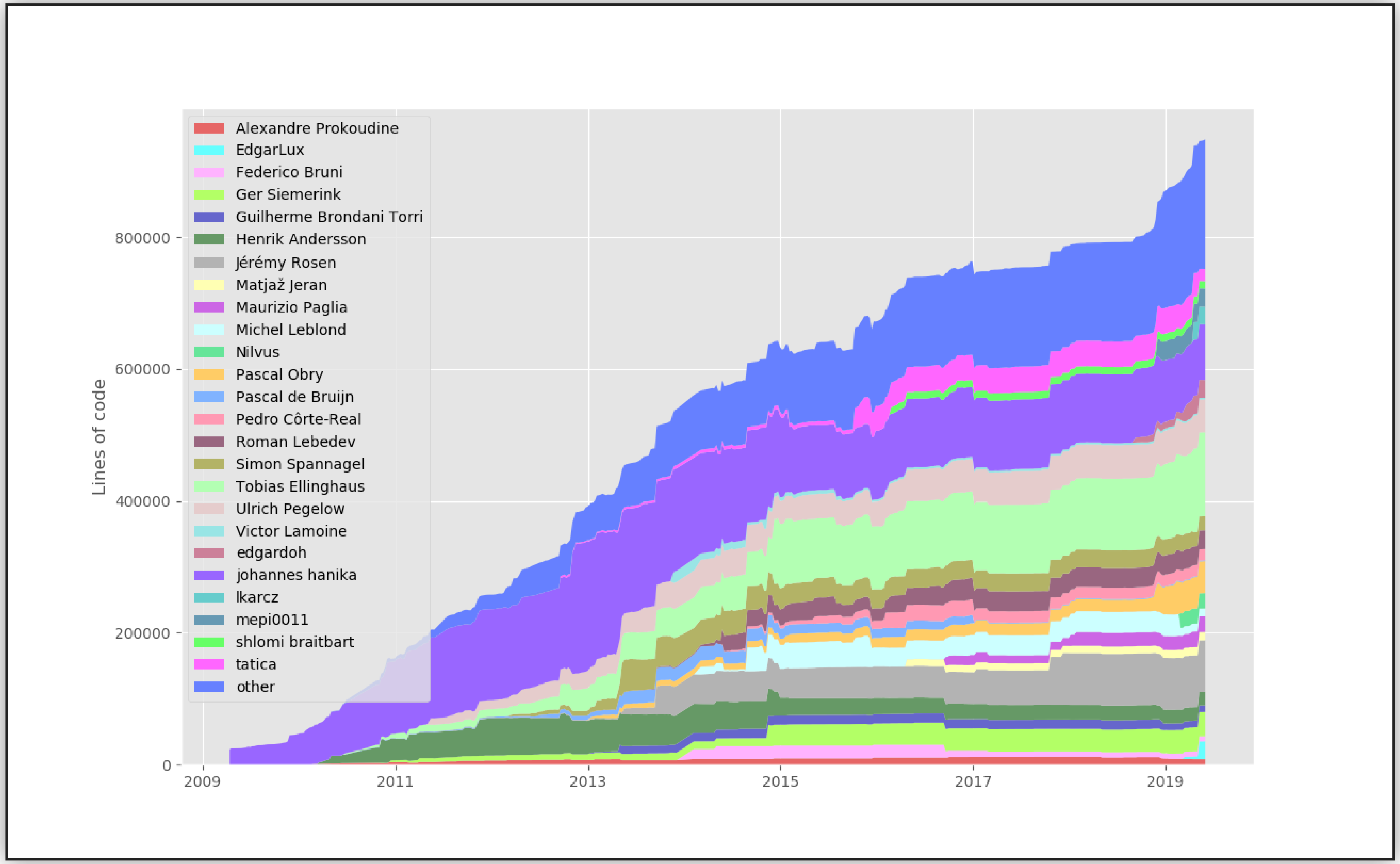
git mining: survival plot



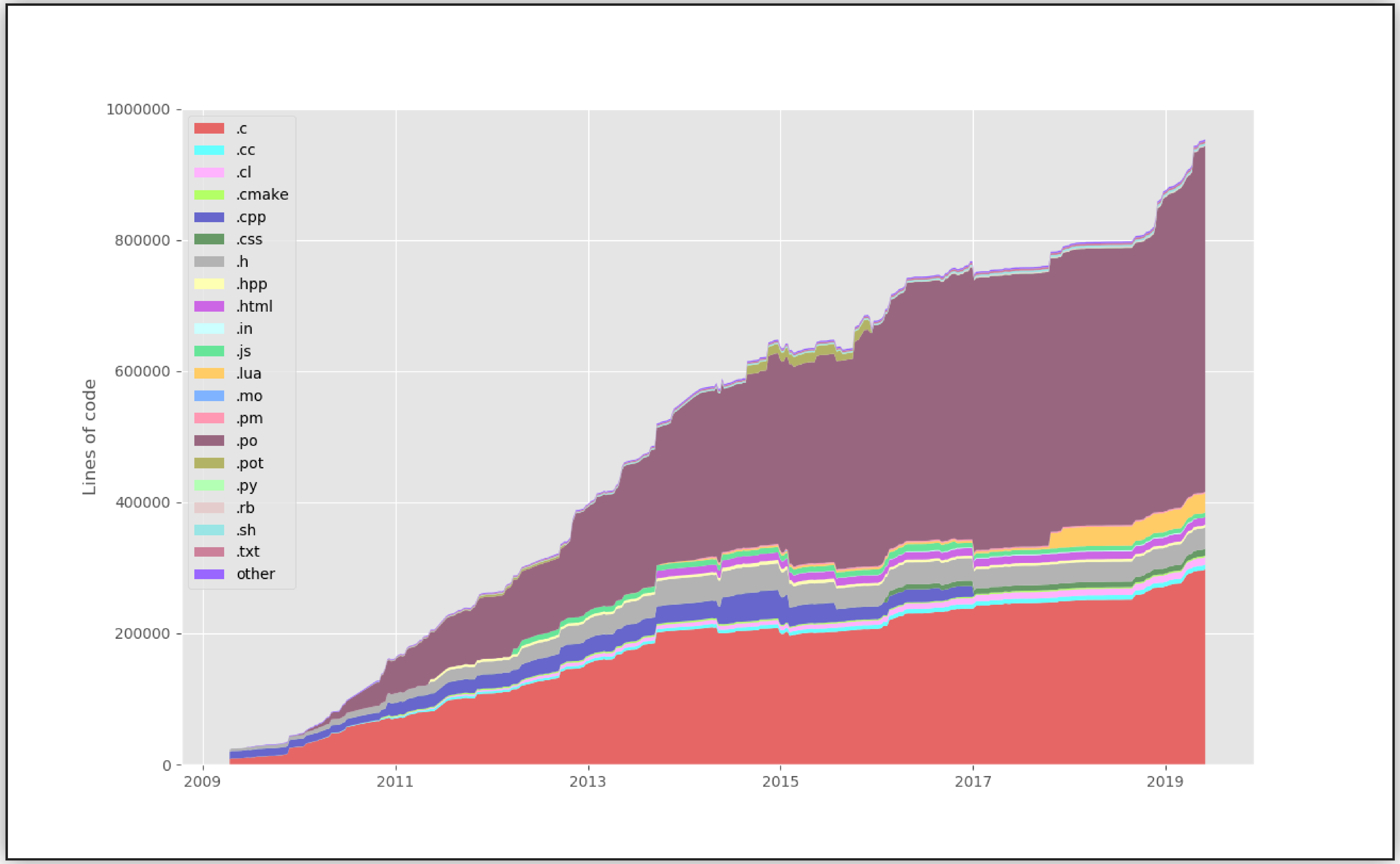
git mining: cohorts



git mining: authors



git mining: file types



git mining: loc ownership

conclusion

- code lasts surprisingly long!
  - even considering that houz ran clang-format..
  - henrik's code after years of inactivity (last: jan 2016) only down to 50%
- means we usually only fix and maintain? or:
  - a) everybody only writes perfect code
  - b) we pile up technical debt without reconsideration

community and organisation

- large number of contributors (just shy of 200)
- different time zones/nationalities/technical background/mindset
  - testers
  - suggesters
  - translators
  - techies
  - colour folks
  - .. sometimes photographers ;)

community and organisation

do-ocracy

- this is a spare time project, we want this to be fun!
- if there's a rule, it's

  don't stop anyone from coding when they're having fun

community and organisation

responsibility

- whether we like it or not: people depend on our software to get their work done
- the non-destructive workflow promise:

  we will support your old edits indefinitely

lessons learned

gained what from laissez-faire?

- **quick** development from ground up
- **features** i never thought of
- translations to **languages** i didn't know existed
- technical debt :/

lessons learned

non-destructive: cannot deprecate things?

- 10 years of ufraw style pipeline
- whole other talk about piling up a stack of technical debts

lessons learned

why are we doing this? fun!

- compatible people
- personalities you need:
  - to advance code (antisocial rockstars)
  - to maintain it after those^ disappeared: with a sense for social responsibility

- communication channels?

outlook/conclusions

hmm, anyone?

- let's discuss here at LGM

thanks!

anyways a huge thanks to all our supporters!

- this has been fun
- let's go for the next 10 years :)