Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Christian Burkert, Johanna Ansohn McDougall, Hannes Federrath

Data Minimisation Potential for Timestamps in Git: An Empirical Analysis of User Configurations

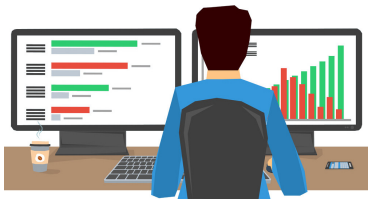
IFIP SEC 2022, June 14

Agenda

Timestamp Data Minimisation (in Git)

- Why do we need it?
- What is necessary timestamp data?
- What is the impact of minimisation on user privacy?

You have probably used Git



confgitour — christian@MacBook-Pro

..er/confgitour

```
➜  confgitour git:(master) ✗ git add paper.tex
➜  confgitour git:(master) ✗ git commit -m "Finalize camera ready"
```

Git is by far the most popular SCM/VCS

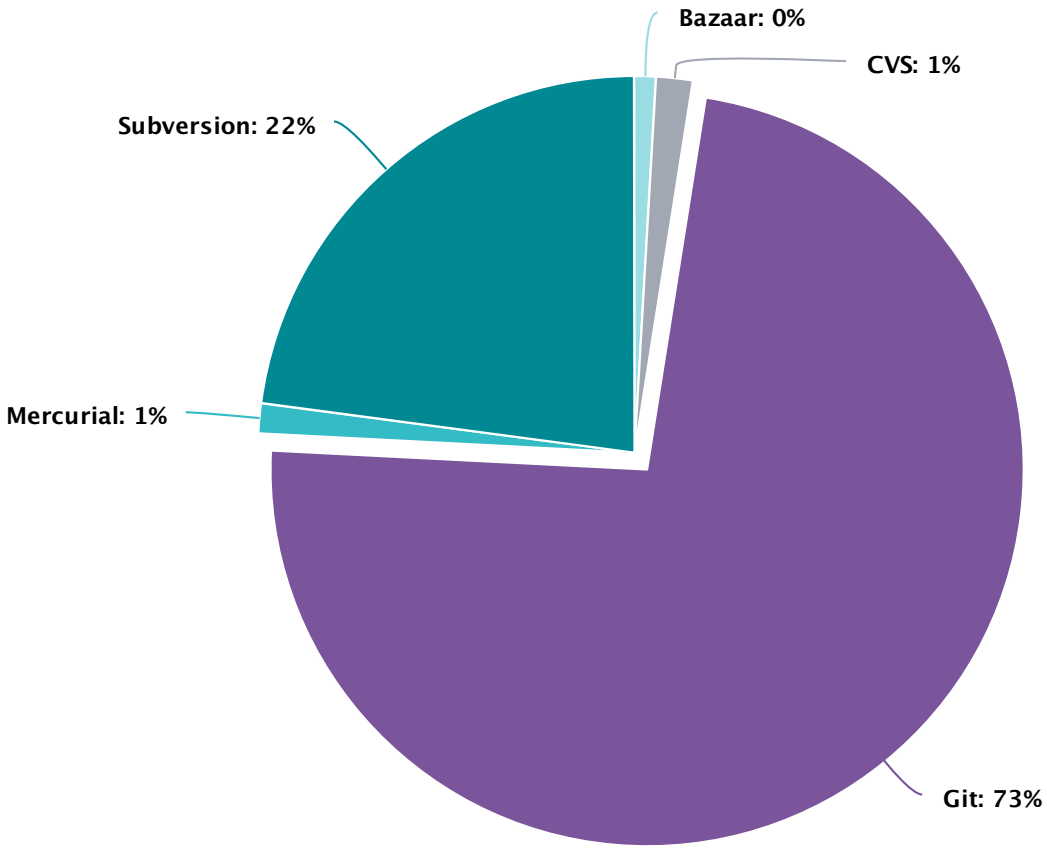


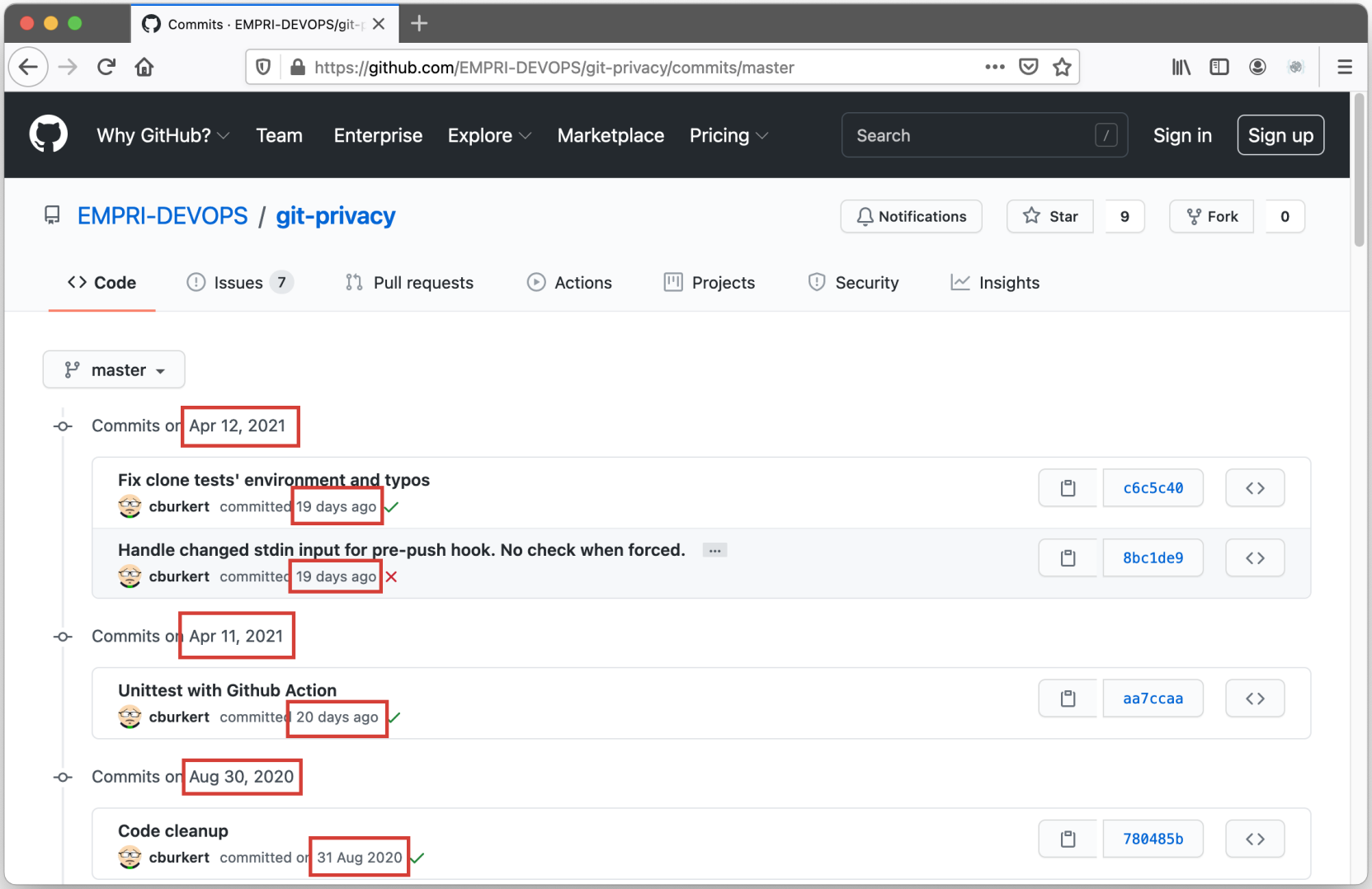
Figure: Git repos make up 73% of the projects indexed by OpenHub

Source: `https://www.openhub.net/repositories/compare`

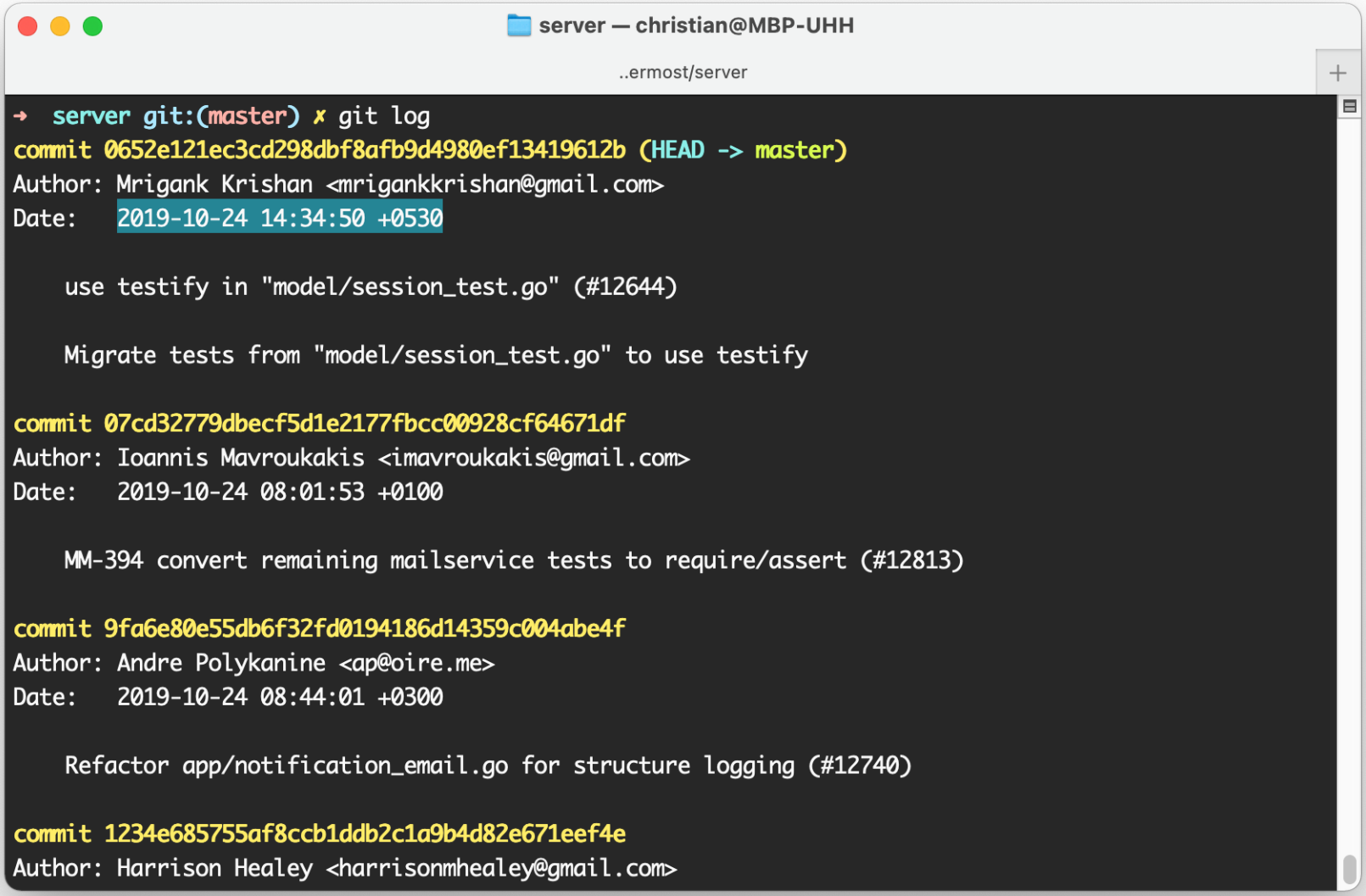
Timestamp in Git

```
1 # git cat-file -p 4a3c5d8119c25804ef94521688fa277a83e0beec
2 tree 0601d4dd701fbefdcdf69c9a9cf71a3b61cfcd28
3 parent 36d60dfe2c66dfdb09767e11511c9d7e6483202d
4 author Chris Lattner <clattner@apple.com> 1485549958 -0800
5 committer Chris Lattner <clattner@apple.com> 1485549958 -0800
6
7 Joe is taking over as the overall maintainer of Swift, thanks.
```

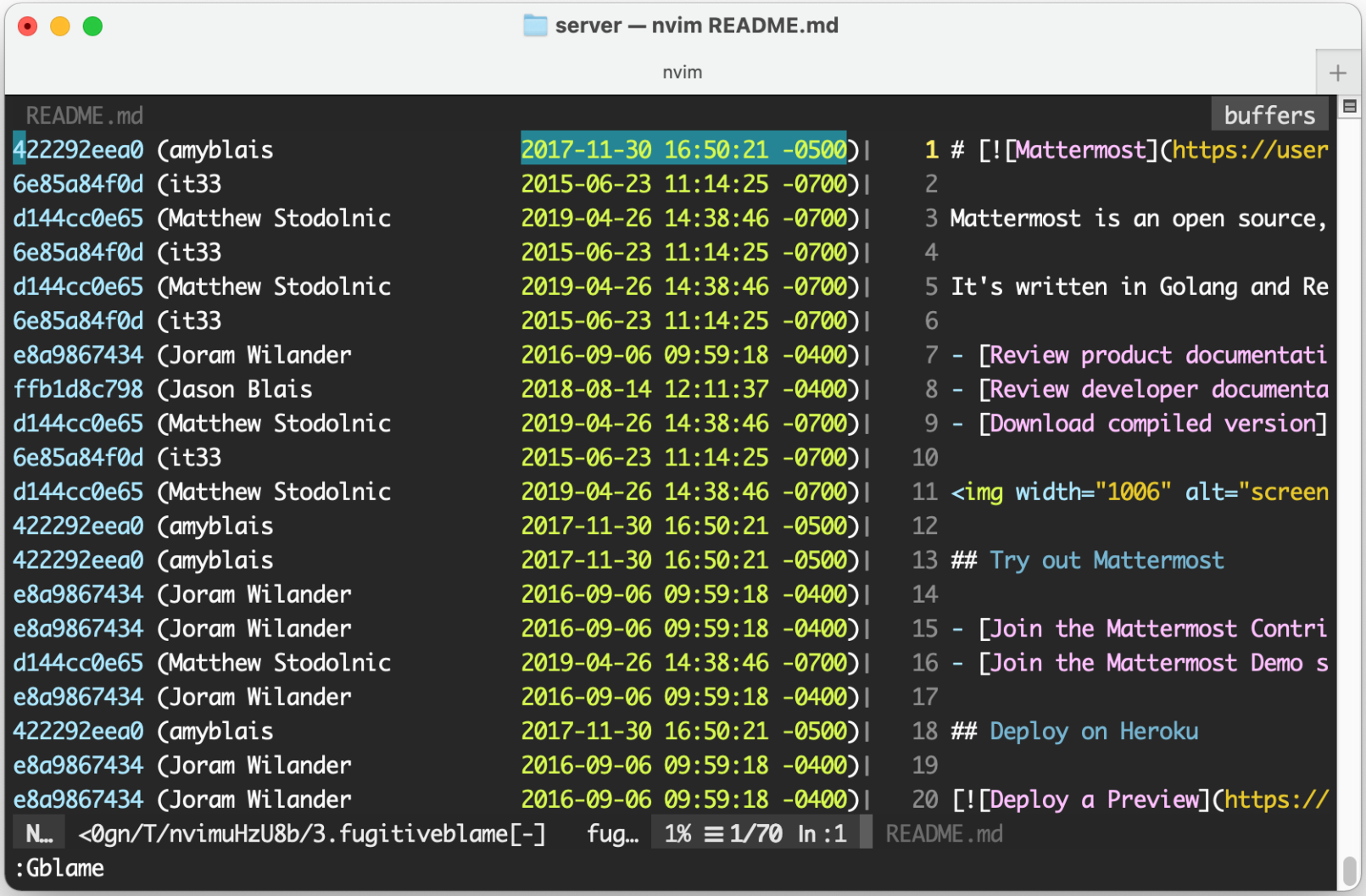
Timestamp as User Info in Git



Timestamp as User Info in Git



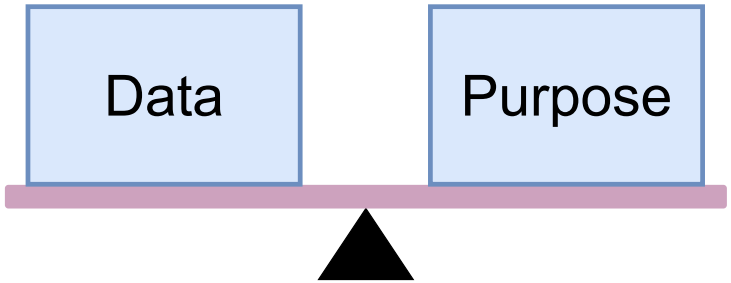
Timestamp as User Info in Git



Background: Data Minimisation

Definition (Data Minimisation)

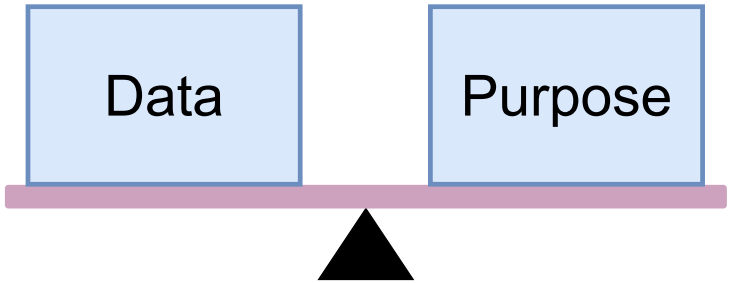
Personal data shall be adequate, relevant and limited to what is **necessary in relation to the purposes** for which they are processed. (Article 5 GDPR)

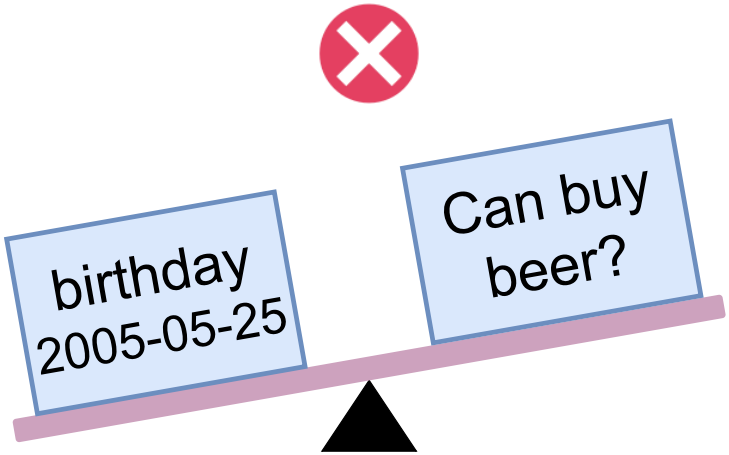


Background: Data Minimisation

Definition (Data Minimisation)

Personal data shall be adequate, relevant and limited to what is **necessary in relation to the purposes** for which they are processed. (Article 5 GDPR)

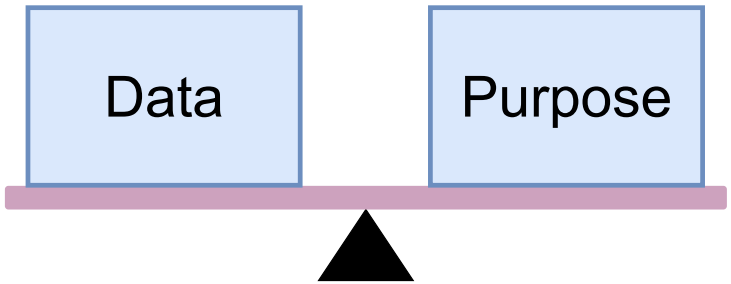


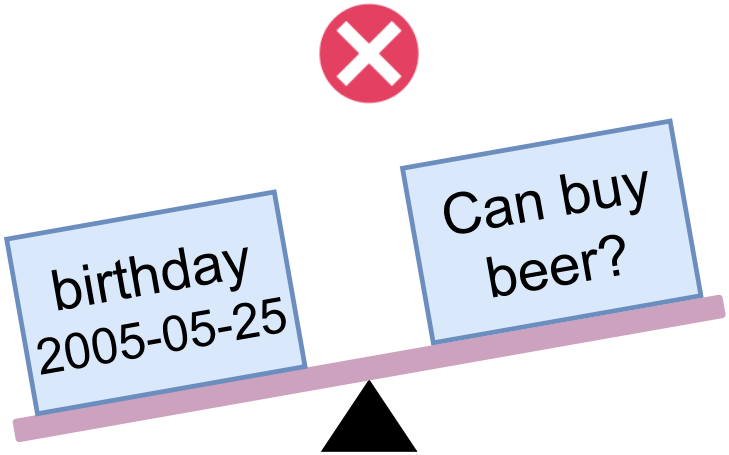


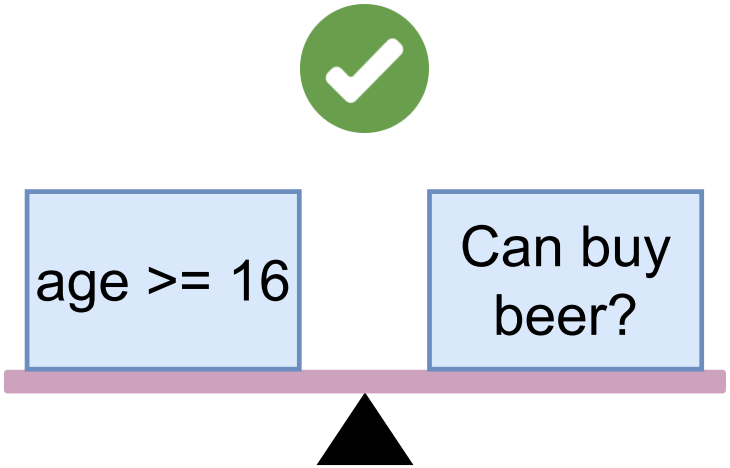
Background: Data Minimisation

Definition (Data Minimisation)

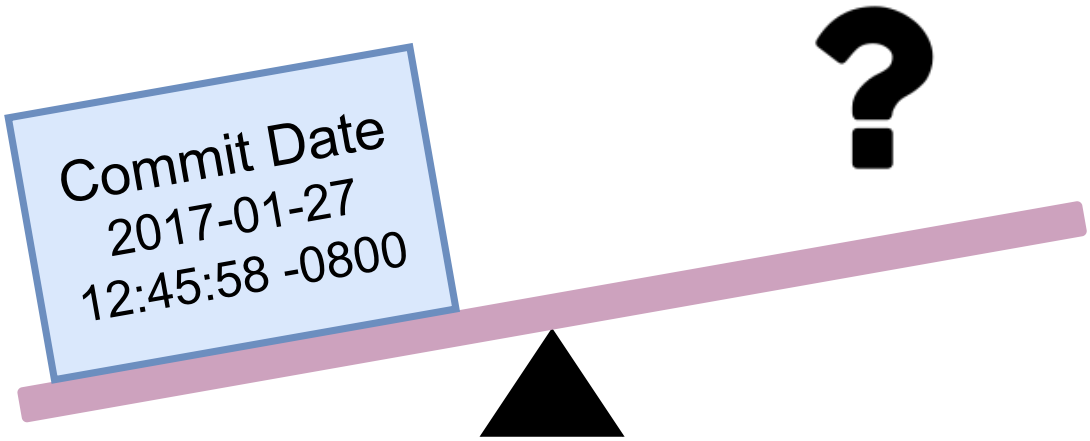
Personal data shall be adequate, relevant and limited to what is **necessary in relation to the purposes** for which they are processed. (Article 5 GDPR)







Goal: Understand Potential for Data Minimisation

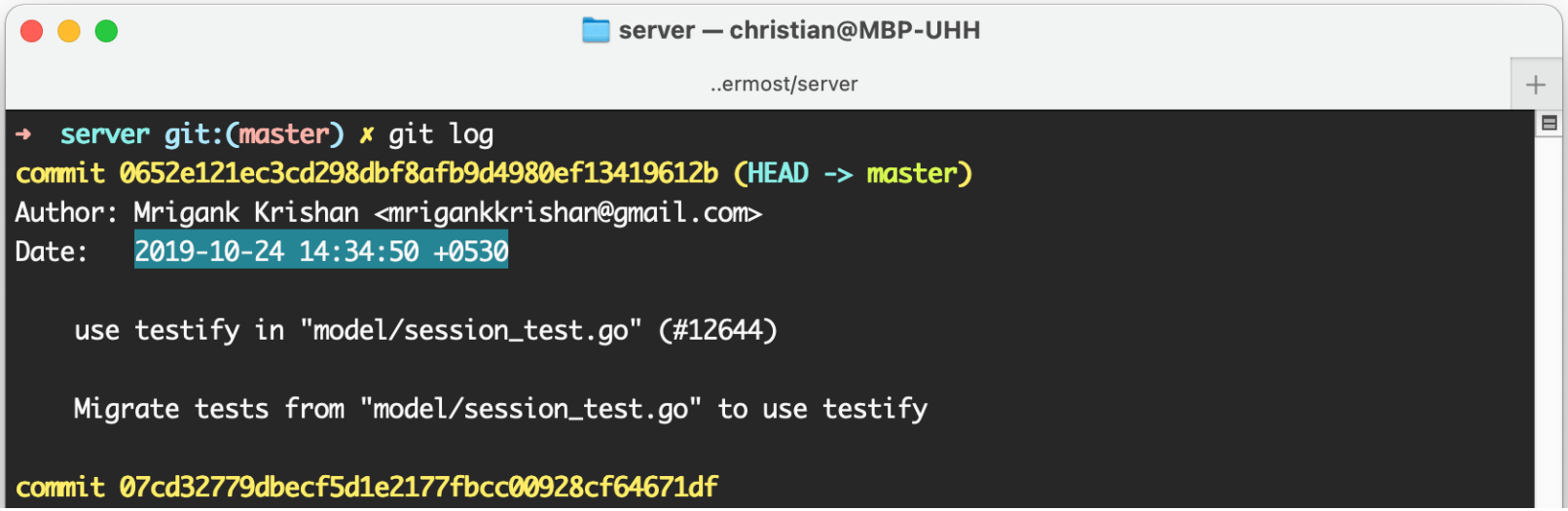


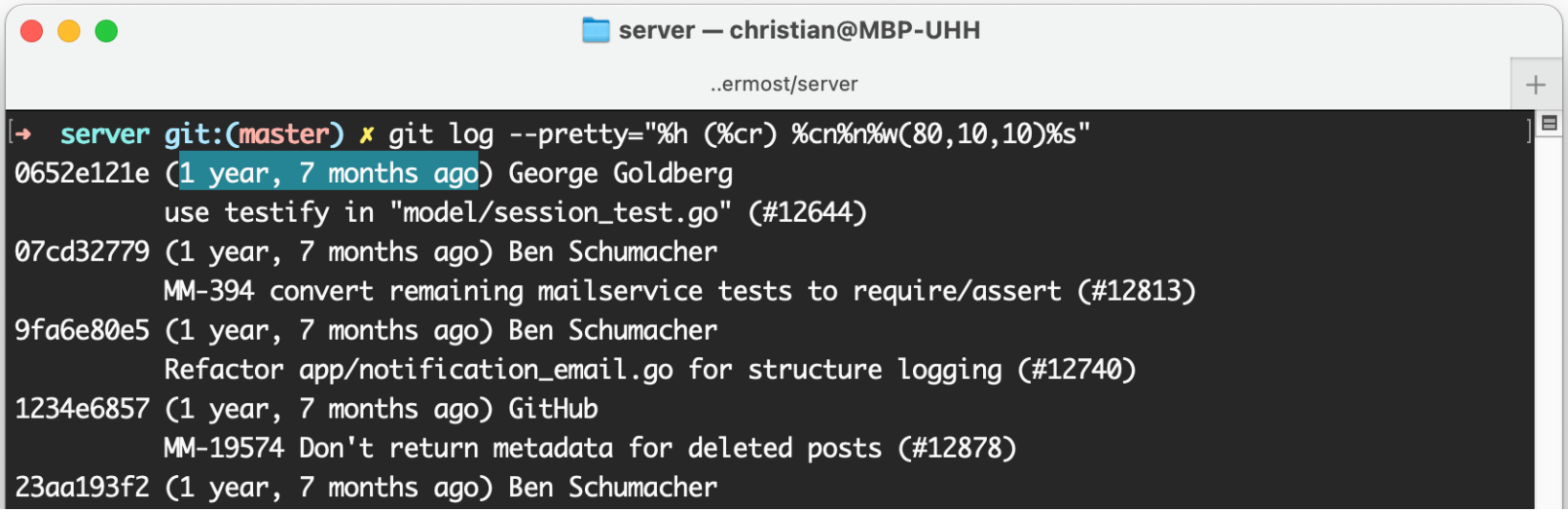

```
# git cat-file -p 4a3c5d8119c25804ef94521688fa277a83e0beec
tree 0601d4dd701fbefdcdf69c9a9cf71a3b61cfcd28
parent 36d60dfe2c66dfdb09767e11511c9d7e6483202d
author Chris Lattner <clattner@apple.com> 1485549958 -0800
committer Chris Lattner <clattner@apple.com> 1485549958 -0800

Joe is taking over as the overall maintainer of Swift, thanks.
```

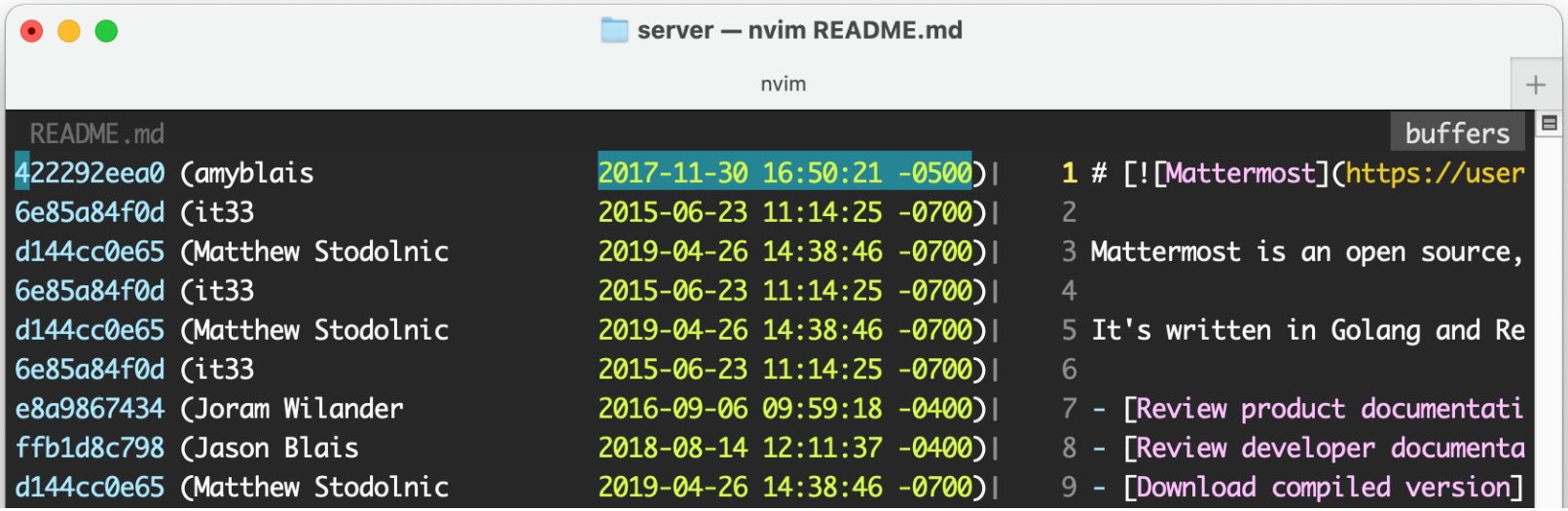
Background:
Customisability of Git's Date Handling

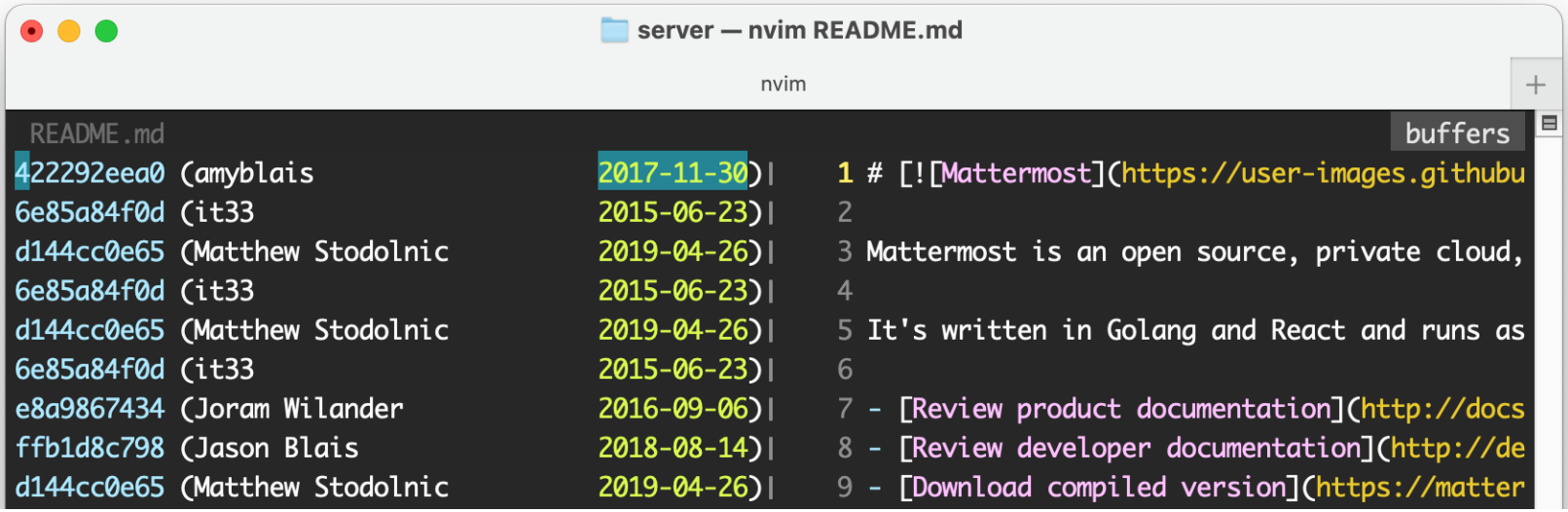
Git's Presentation is Customisable





Git's Presentation is Customisable





Background: Git Options relating Timestamps

Table: Date-related features in Git.

Option	Description
`--pretty[=<format>]`	Custom object output format
`--date=<format>`	Change date output format
`--until=<date>`	Only show objects until `date`
`--since=<date>`	Only show objects since `date`

Background: Git Options relating Timestamps

Table: Date-related features in Git.

Option	Description
`--pretty[=<format>]`	Custom object output format
`--date=<format>`	Change date output format
`--until=<date>`	Only show objects until `date`
`--since=<date>`	Only show objects since `date`

Table: Support for date-related features in Git subcommands.

	annotate	blame	log	shortlog	show	...
`--date`	✔†	✔	✔		✔*	
`--pretty`			✔		✔	
`--since`/`--until`		✔	✔	✔	✔*	

*=undocumented, †=disfunctional

Background: Git's Date Formatting Options (`--date`)

Customisation of the date formatting can also influences precision.

Name	Suffix	Precision	Example(s)
default	-	sec	Wed Sep 22 14:57:31 2021 +0200
human	-	day to	Sep 21 2021
		sec	7 seconds ago
iso	i/I	sec	2021-09-22 14:57:31 +0200
raw	-	sec	1632315451 +0200
relative	r	year to	7 years ago
		sec	7 seconds ago
rfc	D	sec	Wed, 22 Sep 2021 14:57:31 +0200
short	s	day	2021-10-04
unix	t	sec	1632315451

Background: Git's Pretty Formatting Options (`--pretty`)

Allows customisation of

- included date types (author, commit, both, none)
- date formatting (like with `--date`).

Background: Git's Pretty Formatting Options (`--pretty`)

Allows customisation of

- included date types (author, commit, both, none)
- date formatting (like with `--date`).

```
$ git log --pretty=fuller
commit e71a7ef2090484177c75726b4e92a5d8945f0fb2
Author:     Christian Burkert <burkert@example.com>
AuthorDate: 2021-05-20 12:00:01 +0200
Commit:     John Doe <doe@example.com>
CommitDate: 2021-05-22 00:00:00 +0500

    I'm a commit message headline

```

Background: Git's Pretty Formatting Options (`--pretty`)

Allows customisation of

- included date types (author, commit, both, none)
- date formatting (like with `--date`).

```
$ git log --pretty=fuller
commit e71a7ef2090484177c75726b4e92a5d8945f0fb2
Author:     Christian Burkert <burkert@example.com>
AuthorDate: 2021-05-20 12:00:01 +0200
Commit:     John Doe <doe@example.com>
CommitDate: 2021-05-22 00:00:00 +0500

    I'm a commit message headline

$ git log --pretty="%h <%an> (%cr)%n%w(80,8,8)%s"
e71a7ef <Christian Burkert> (5 days ago)
        I'm a commit message headline
```

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example: 1 hour 30 minutes ago | yesterday 5pm

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example: $\underbrace{\text{1 hour}}_{\text{hour}}\ \underbrace{\text{30 minutes}}_{\text{minute}}\ \text{ago}$ | yesterday 5pm

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example: $\underbrace{\text{1 hour}}_{\text{hour}}\ \underbrace{\text{30 minutes}}_{\text{minute}}\ \text{ago}$ | $\underbrace{\text{yesterday}}_{\text{day}}\ \underbrace{\text{5pm}}_{\text{hour}}$

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example: $\underbrace{\text{1 hour}}_{\text{hour}}\ \underbrace{\text{30 minutes}}_{\text{minute}}$ ago

Precisions: hour < minute

Example: $\underbrace{\text{yesterday}}_{\text{day}}\ \underbrace{\text{5pm}}_{\text{hour}}$

Precisions: day < hour

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example:	$\underbrace{\text{1 hour}}_{\text{hour}}\ \underbrace{\text{30 minutes}}_{\text{minute}}$ ago	$\underbrace{\text{yesterday}}_{\text{day}}\ \underbrace{\text{5pm}}_{\text{hour}}$
Precisions:	hour < minute	day < hour
Highest:	minute	hour

Background: Git Configuration Files

Customisations can also be made in config files (e.g., `~/.gitconfig`).

```
1 [alias]
2   lf = log --pretty=fuller --date=iso
3   hi = log --pretty="%h <%an>%d (%cr)%n%w(80,8,8)%s"
4   bs = blame --date=short
5   sl = shortlog --since="1 day ago"
6
```

Background: Git Configuration Files

Customisations can also be made in config files (e.g., `~/.gitconfig`).

```
1 [alias]
2   lf = log --pretty=fuller --date=iso
3   hi = log --pretty="%h <%an>%d (%cr)%n%w(80,8,8)%s"
4   bs = blame --date=short
5   sl = shortlog --since="1 day ago"
6
7 [pretty]
8   my = %h %an (%ar)
9
```

Background: Git Configuration Files

Customisations can also be made in config files (e.g., `~/.gitconfig`).

```
[alias]
  lf = log --pretty=fuller --date=iso
  hi = log --pretty="%h <%an>%d (%cr)%n%w(80,8,8)%s"
  bs = blame --date=short
  sl = shortlog --since="1 day ago"

[pretty]
  my = %h %an (%ar)

[format]
  pretty = my
[log]
  date = iso
[blame]
  date = short
```

Evaluation

Empirical Analysis of Git Config Files

Guiding Are second-precision timestamps in Git necessary?

Empirical Analysis of Git Config Files

Guiding Are second-precision timestamps in Git necessary?

Dataset GitHub hosts over 20 000 config files that contain alias definitions

Empirical Analysis of Git Config Files

Guiding Are second-precision timestamps in Git necessary?

Dataset GitHub hosts over 20 000 config files that contain alias definitions

RQ 1. What precision demands result from user customisations?

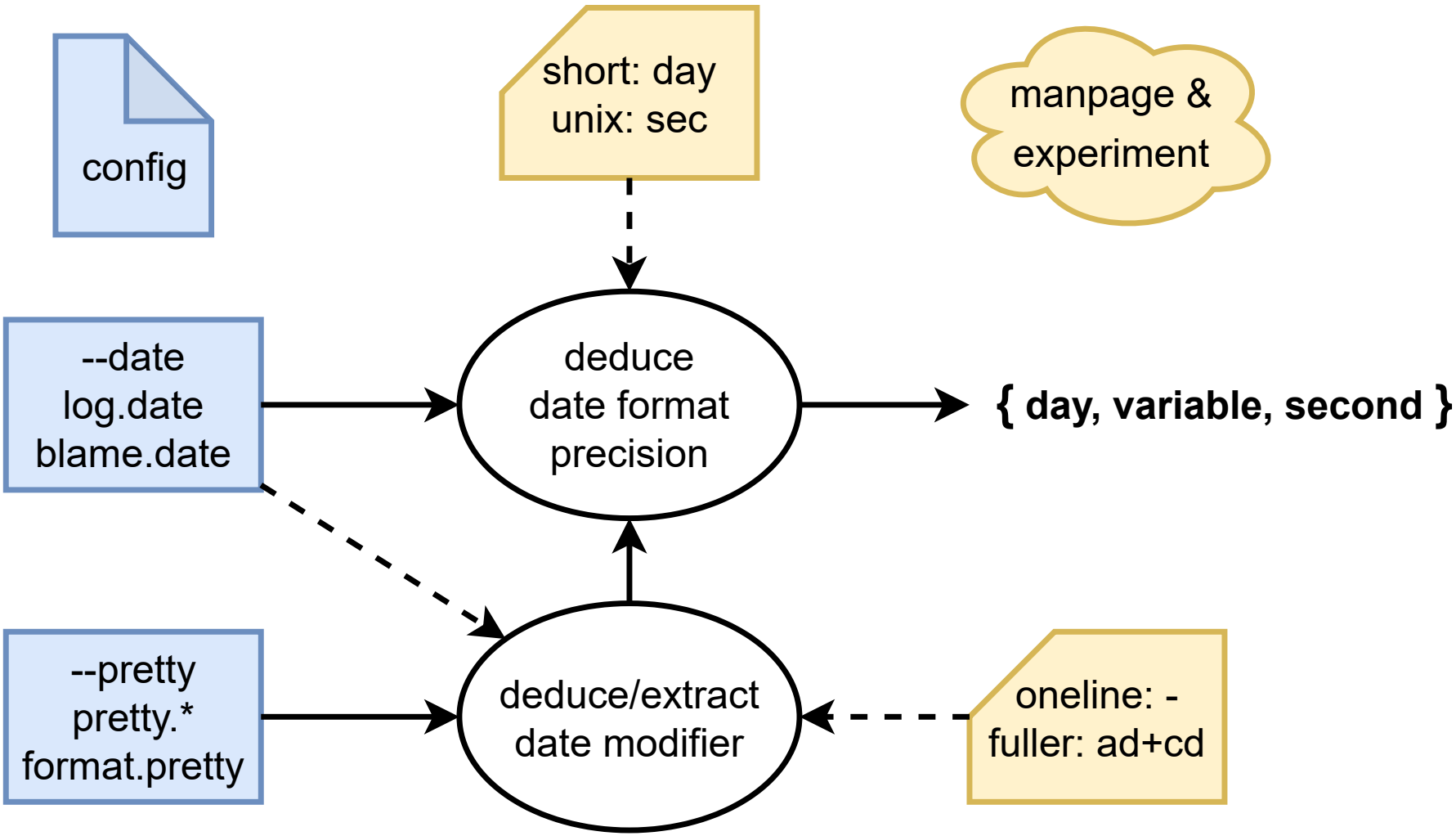
Empirical Analysis of Git Config Files

Guiding Are second-precision timestamps in Git necessary?

Dataset GitHub hosts over 20 000 config files that contain alias definitions

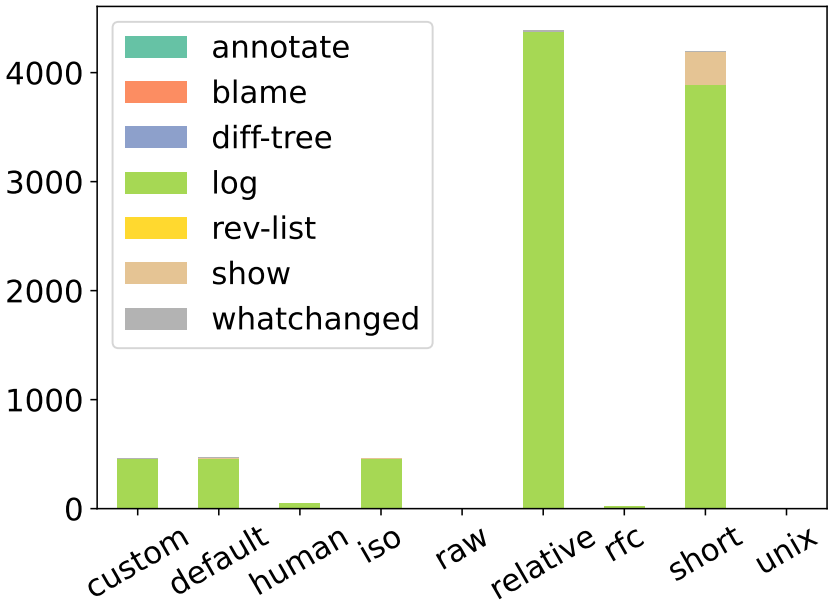
RQ
1. What precision demands result from user customisations?
2. Would a small to moderate precision reduction have an impact on user privacy?

Process of Precision Evaluation

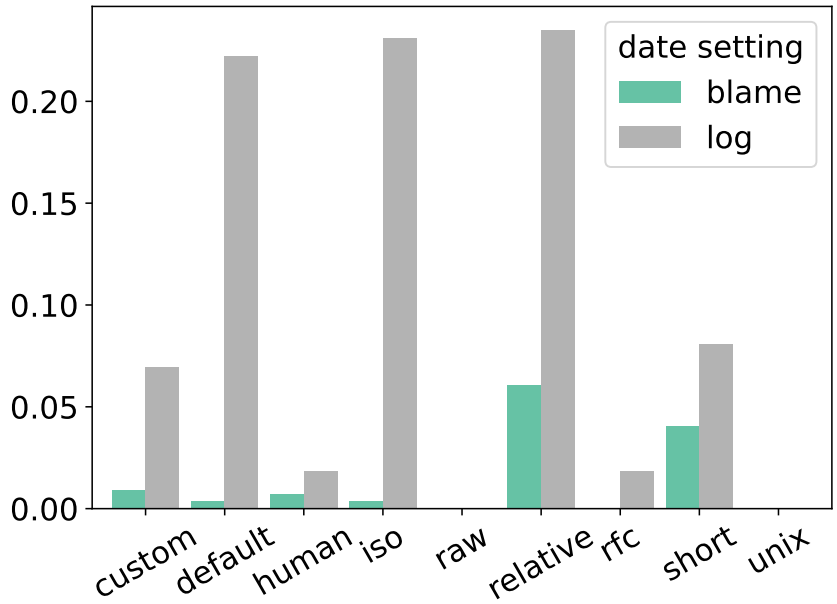


Results for Date Formatting (`--date`)

- Popularity of date formats depends on context (see *short*)
- Format *relative* popular in all contexts



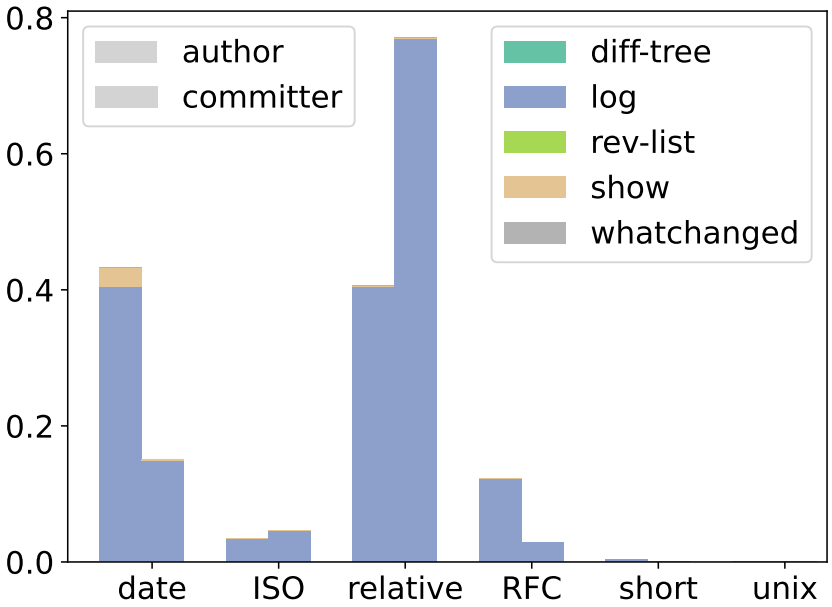
(a) used with `--date` in command aliases



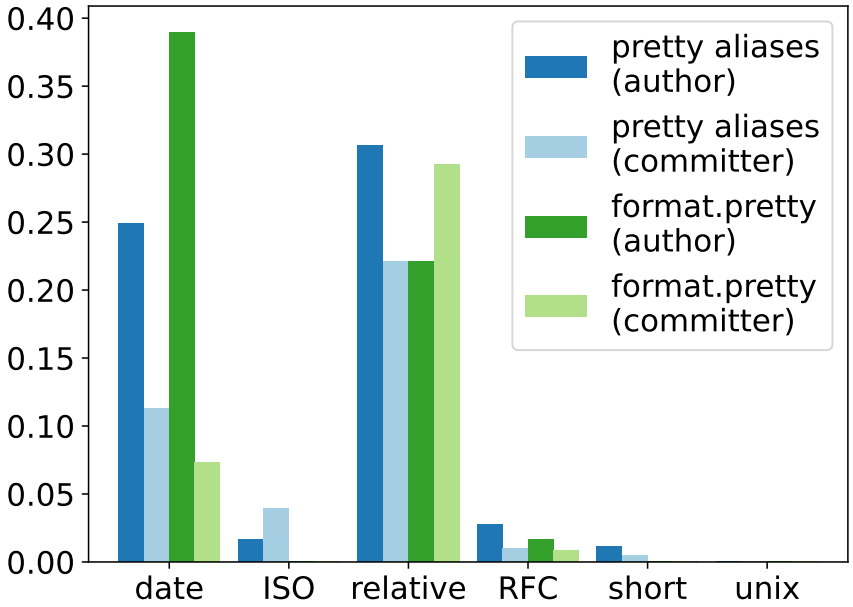
(b) used for config settings

Results for Pretty Formatting (`--pretty`)

- ~40% of formats use neither date type
- otherwise *date* and *relative* are popular

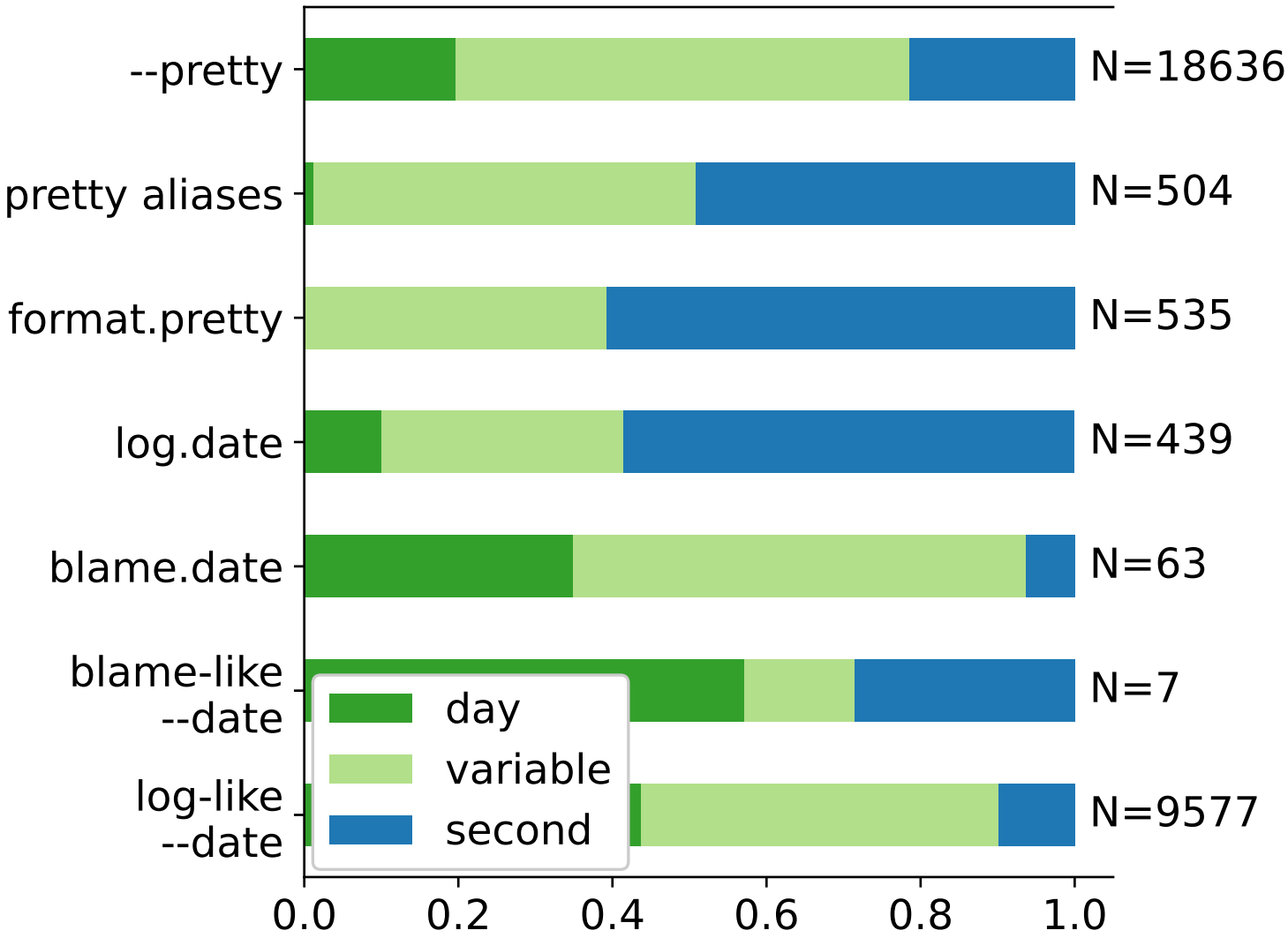


(a) custom formats in subcommand aliases

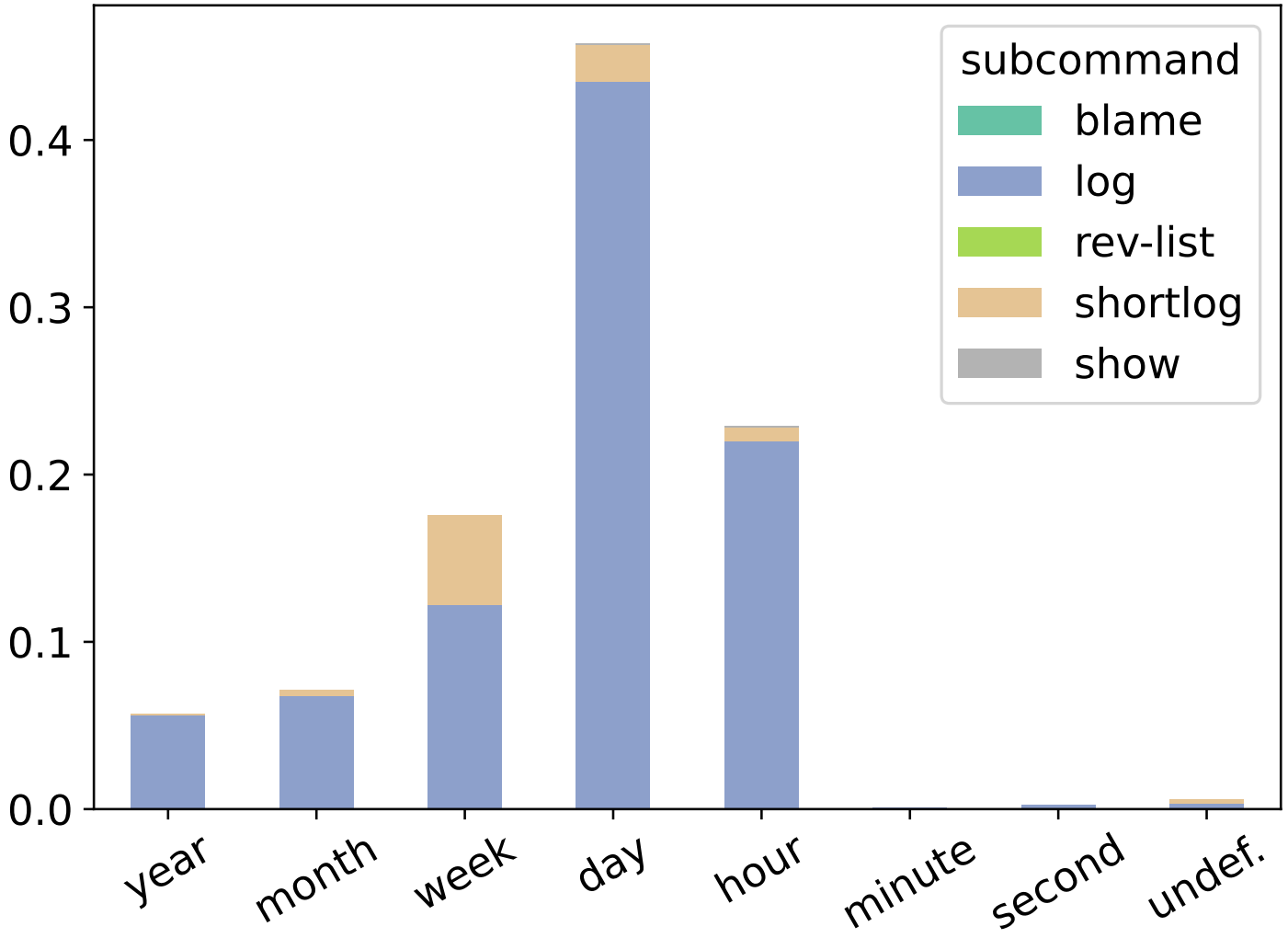


(b) format aliases and `format.pretty`

Results: Date Precision in Output Formatting (Overview)



Results: Precision in Filters



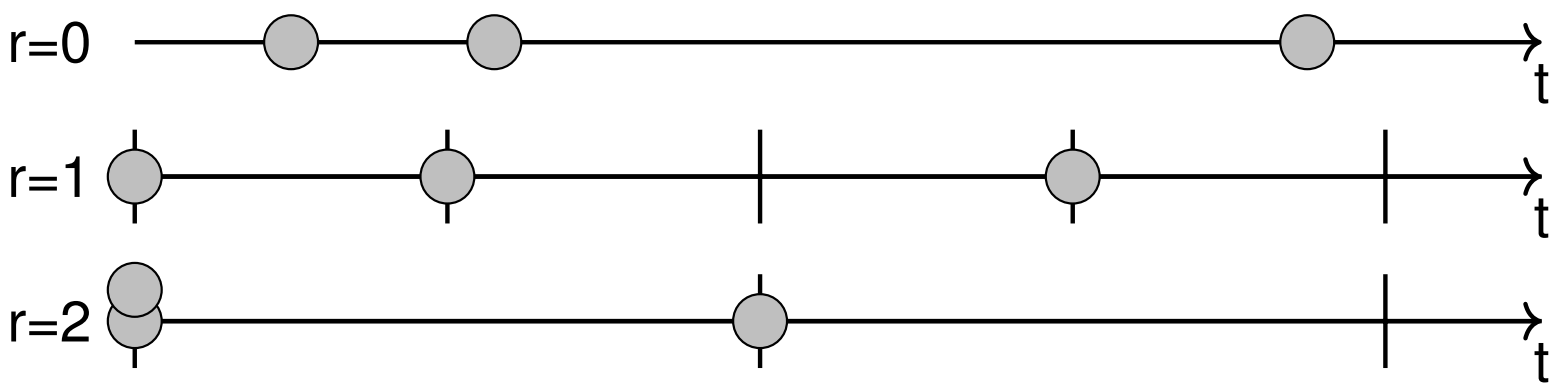
RQ 2: Assessing the Privacy Impact of Precision Reduction

RQ Would a small to moderate precision reduction have an impact on user privacy?

RQ 2: Assessing the Privacy Impact of Precision Reduction

RQ Would a small to moderate precision reduction have an impact on user privacy?

Assumption Making chronologically adjacent activities temporally indistinguishable reduces privacy impact

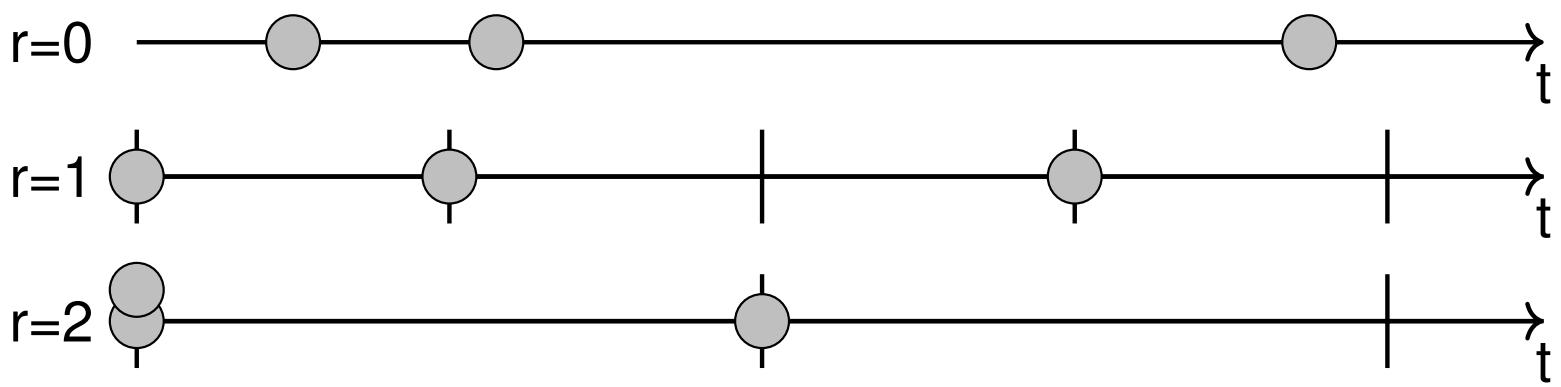


RQ 2: Assessing the Privacy Impact of Precision Reduction

RQ Would a small to moderate precision reduction have an impact on user privacy?

Assumption Making chronologically adjacent activities temporally indistinguishable reduces privacy impact

Dataset GitHub snapshot with 360 million commits of medium-active users (1k to 10k total commits)



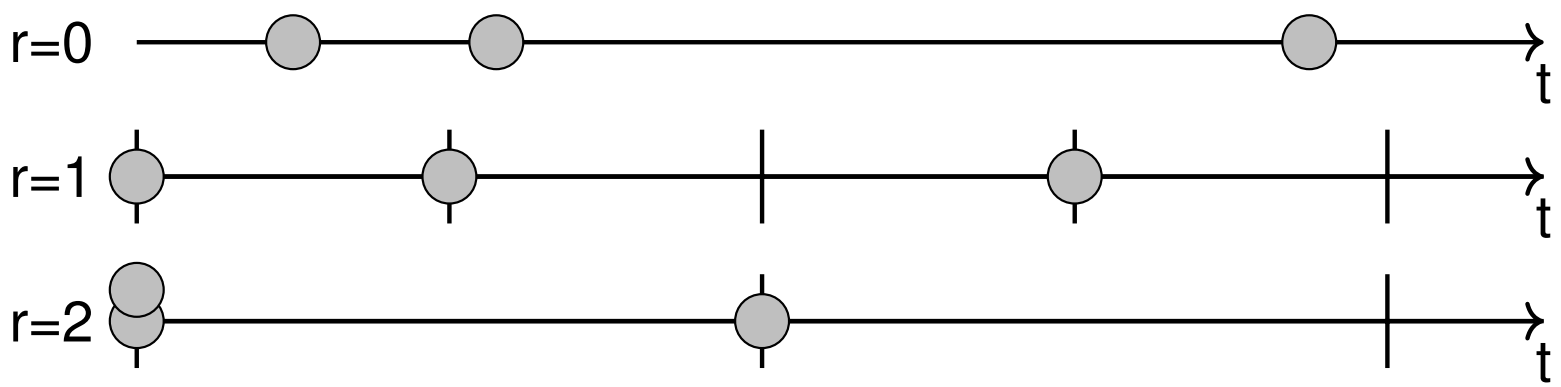
RQ 2: Assessing the Privacy Impact of Precision Reduction

RQ Would a small to moderate precision reduction have an impact on user privacy?

Assumption Making chronologically adjacent activities temporally indistinguishable reduces privacy impact

Dataset GitHub snapshot with 360 million commits of medium-active users (1k to 10k total commits)

Variable Ratio of a user's temporally distinguishable activities depending on precision reduction level



RQ 2: Assessing the Privacy Impact of Precision Reduction

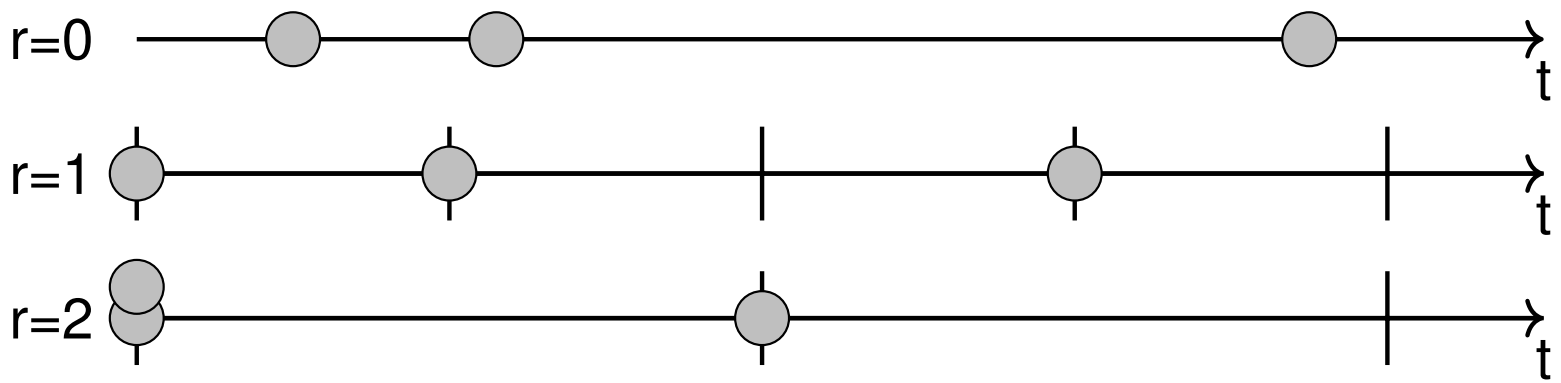
RQ Would a small to moderate precision reduction have an impact on user privacy?

Assumption Making chronologically adjacent activities temporally indistinguishable reduces privacy impact

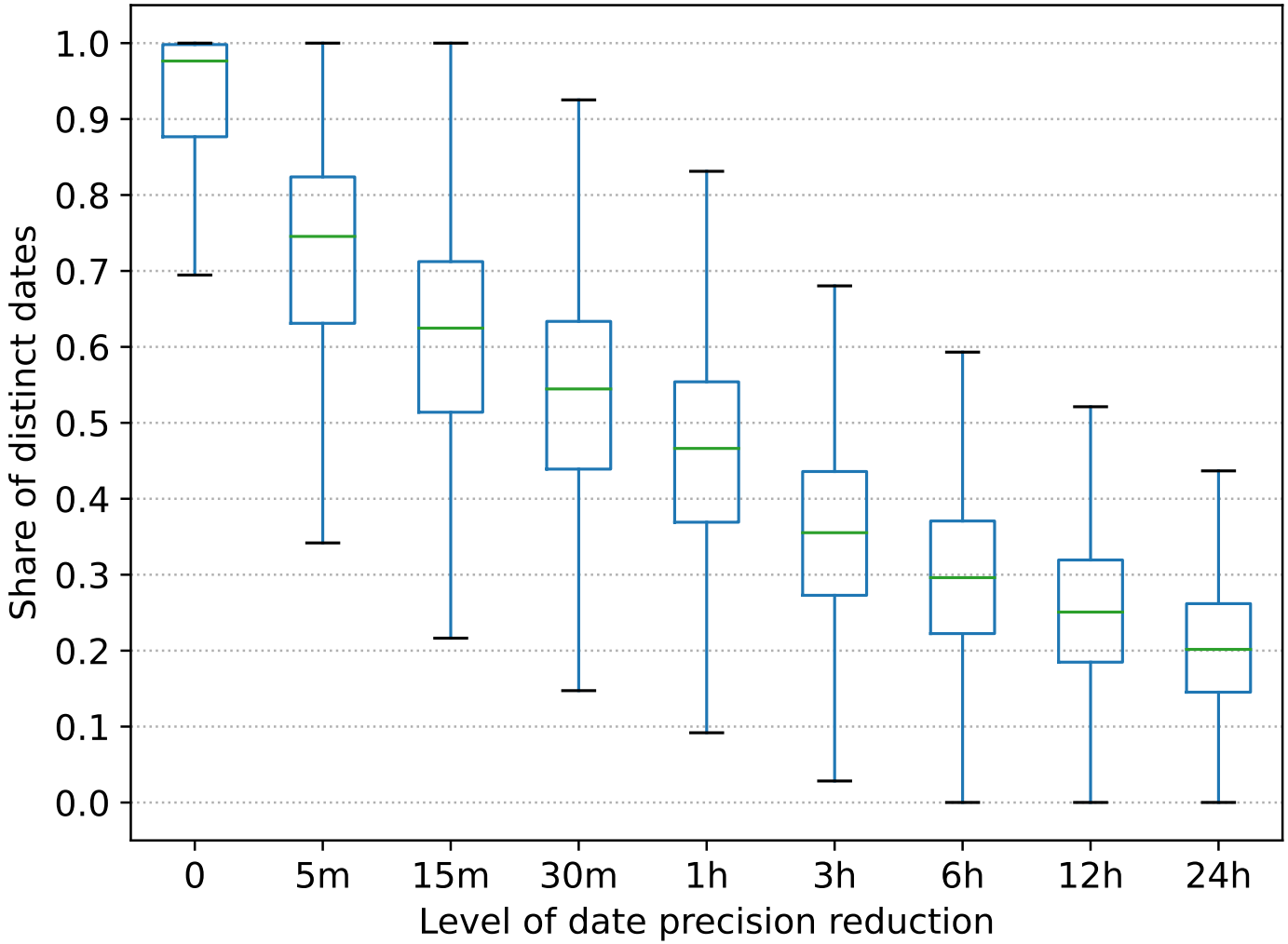
Dataset GitHub snapshot with 360 million commits of medium-active users (1k to 10k total commits)

Variable Ratio of a user's temporally distinguishable activities depending on precision reduction level

Result Ratio of distinguishable activities declines fast: Precision 1h: $<50\%$ distinguishable (median)



RQ 2: Assessing the Privacy Impact of Precision Reduction II



Limitations

- Git configs as indicators for precision demand
  - Biased towards desire for non-default behaviour?

Limitations

- Git configs as indicators for precision demand
  - Biased towards desire for non-default behaviour?
  - Biased towards more experienced users (discoverability)?

Limitations

- Git configs as indicators for precision demand
  - Biased towards desire for non-default behaviour?
  - Biased towards more experienced users (discoverability)?
  - Format might be chosen counter to demand (e. g., ISO)

Limitations

- Git configs as indicators for precision demand
  - Biased towards desire for non-default behaviour?
  - Biased towards more experienced users (discoverability)?
  - Format might be chosen counter to demand (e. g., ISO)
- Privacy impact assessment
  - Ratio of distinguishable time points only considers risks of direct interval inference,
    i. e., how long did it that between X and Y?

Summary: Findings & Contribution

We contributed empirical evidence on

- timestamp precision demand (in Git)
  - analysed >20 000 public Git configs
  - <10% of `--date` in aliases use seconds
  - ~10% of `--pretty` in aliases use sec
  - ~0.5% of filters use min or sec
  - published dataset of config features

Summary: Findings & Contribution

We contributed empirical evidence on

- timestamp precision demand (in Git)
  - analysed >20 000 public Git configs
  - <10% of `--date` in aliases use seconds
  - ~10% of `--pretty` in aliases use sec
  - ~0.5% of filters use min or sec
  - published dataset of config features
- privacy impact of precision reduction
  - analysed >360 million commits
  - distinguishable: 75% at 5min, 50% at 1h

Summary: Findings & Contribution

We contributed empirical evidence on

- timestamp precision demand (in Git)
  - analysed >20 000 public Git configs
  - <10% of `--date` in aliases use seconds
  - ~10% of `--pretty` in aliases use sec
  - ~0.5% of filters use min or sec
  - published dataset of config features
- privacy impact of precision reduction
  - analysed >360 million commits
  - distinguishable: 75% at 5min, 50% at 1h

Developed `git-privacy` for timestamp redaction

`https://github.com/EMPRI-DEVOPS/git-privacy`

Contact

Christian Burkert

Tel. +49 40 42883-2406
Mail christian.burkert@uni-hamburg.de

I'd be happy to hear from you!

OpenPGP Fingerprint:

`9B97 CC4B 5FF4 7BA3 EF7B 1966 A5FB 6E0B 41AC CDFB`