Storing data with Git

INTRODUCTION TO VERSION CONTROL WITH GIT

George Boorman
Curriculum Manager, DataCamp

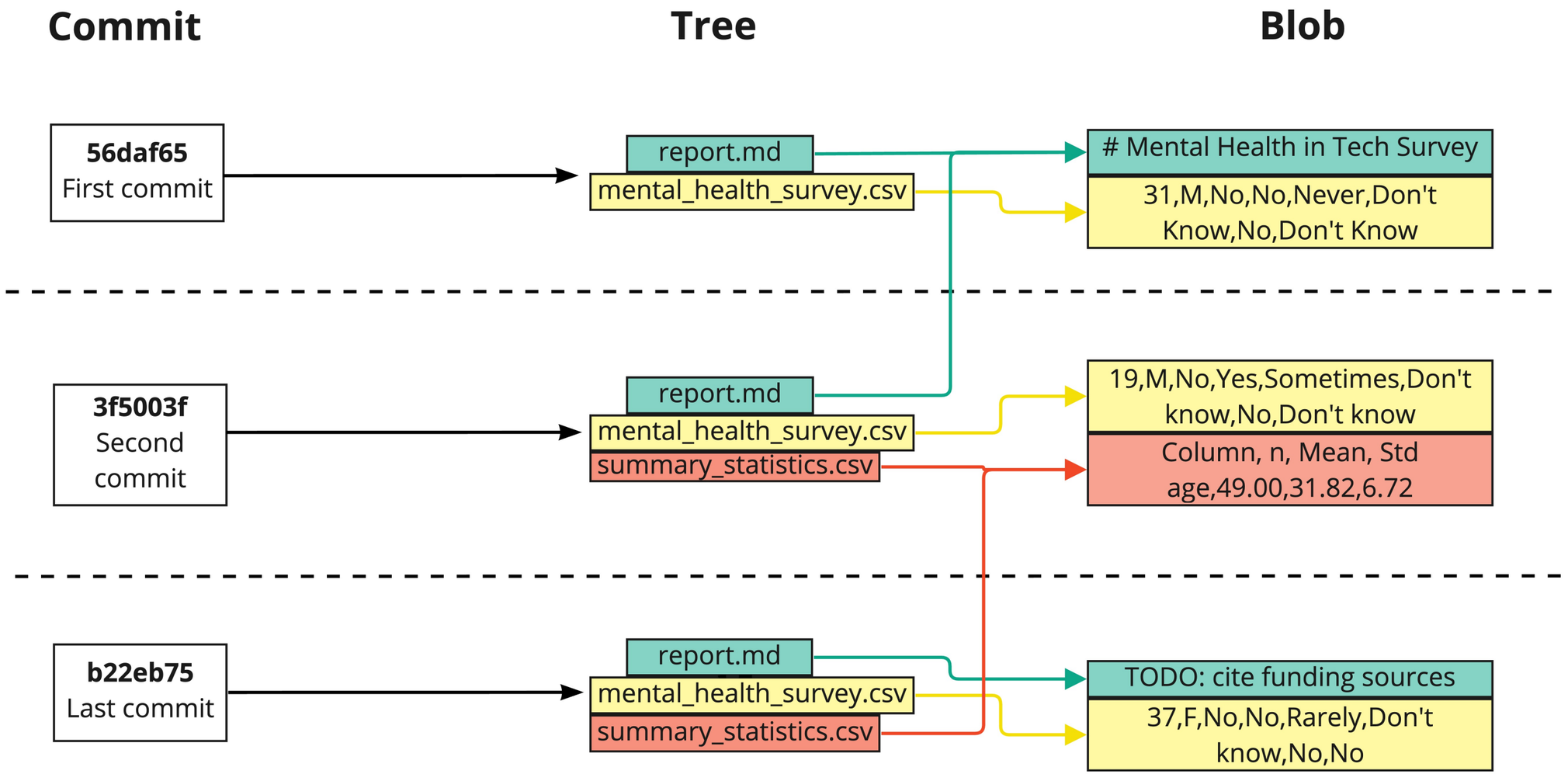
datacamp

The commit structure

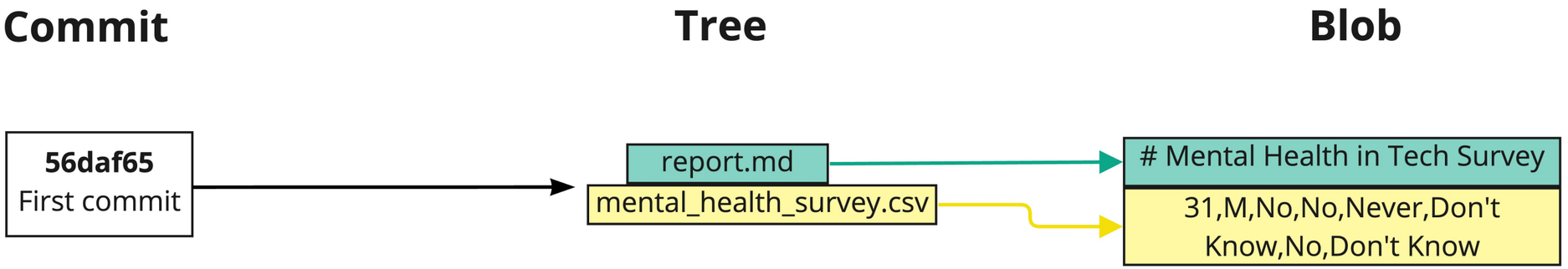
Git commits have **three** parts:

- **Commit**
  - contains the metadata
- **Tree**
  - tracks the names and locations in the repo
- **Blob**
  - binary large object
  - may contain data of any kind
  - compressed snapshot of a file's contents

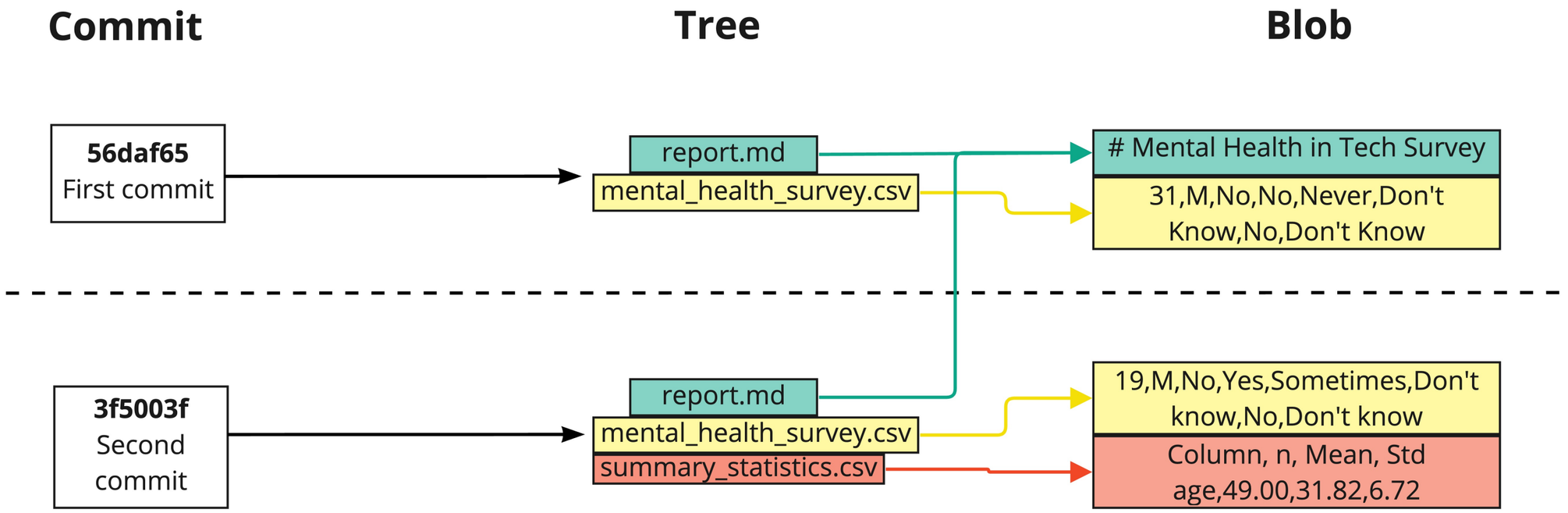
Visualizing the commit structure



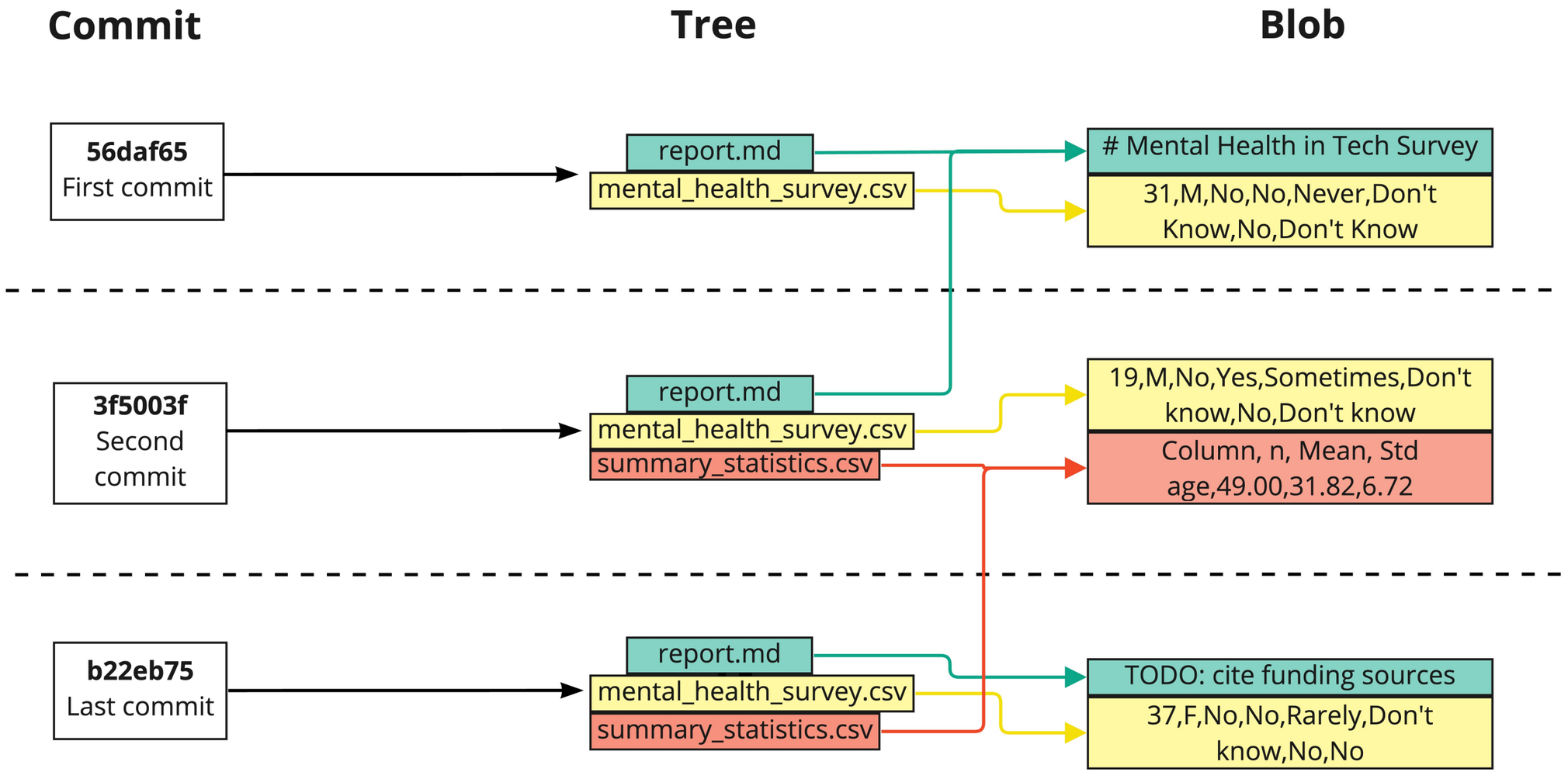
Visualizing the commit structure



Visualizing the commit structure



Visualizing the commit structure



Git log

```
git log
```

```
commit ad8accfe94cb924444c488132bdef7c54b9bca68
Author: Rep Loop <repl@datacamp.com>
Date:   Wed Jul 27 07:48:27 2022 +0000

    Added reminder to cite funding sources.
:
```

- Press `space` to show more recent commits
- Press `q` to quit the log and return to the terminal

Git hash

Last commit: `b22eb75a82a68b9c0f1c45b9f5a9b7abe281683a`

- Pseudo-random number generator—**hash** function
- Hashes allow data sharing between repos
  - If two files are the same,
    - then their hashes are the same
  - Git only needs to compare hashes

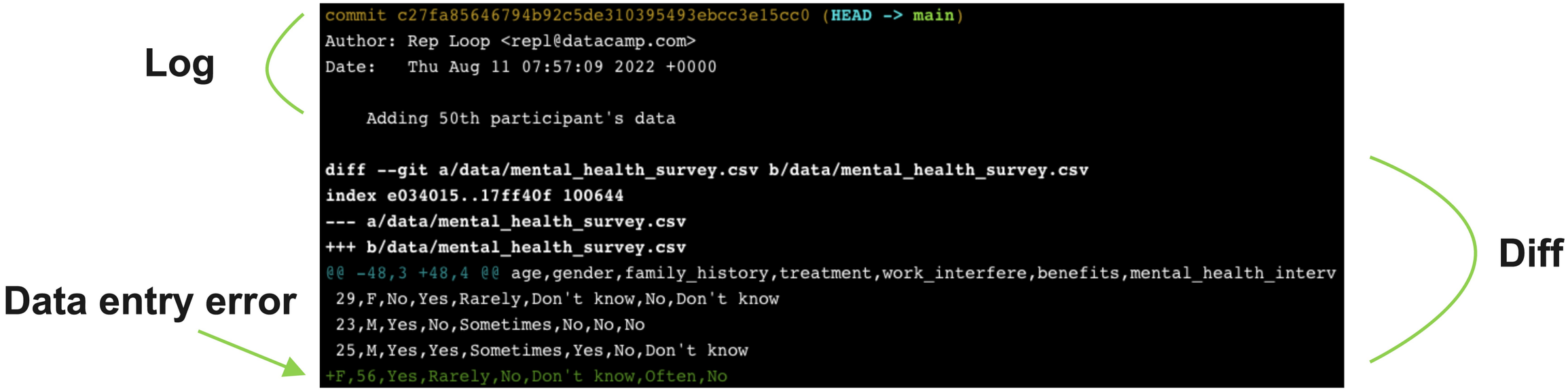
Finding a particular commit

```
git log
```

- Only need the first 6-8 characters of the `hash`

```
git show c27fa856
```

Git show output



Let's practice!

INTRODUCTION TO VERSION CONTROL WITH GIT

Viewing changes

INTRODUCTION TO VERSION CONTROL WITH GIT

George Boorman
Curriculum Manager, DataCamp

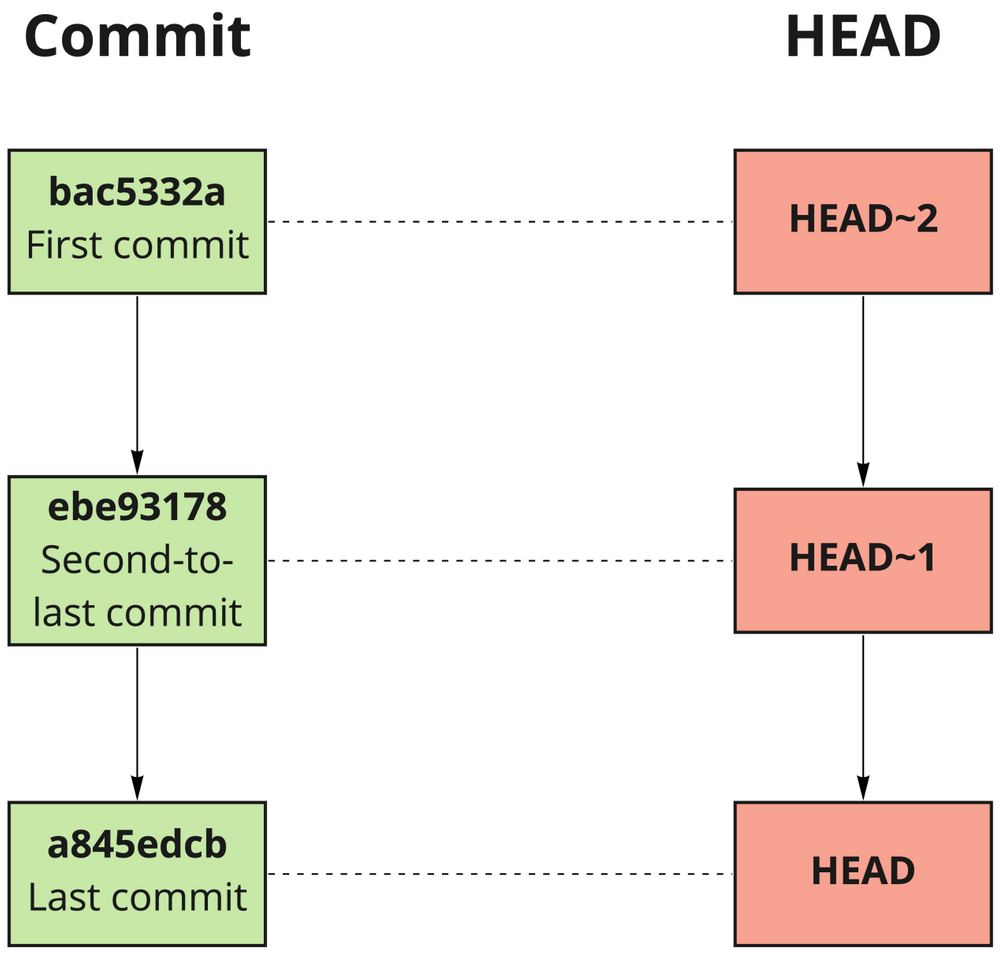
datacamp

The HEAD shortcut

```
git diff -r HEAD
```

- Compares staged files to the version in the last commit
- Use a tilde `~` to pick a specific commit to compare versions
  - `HEAD~1` = the second most recent commit
  - `HEAD~2` = the commit before that
- Must not use spaces before or after the tilde `~`

From commit hash to HEAD



Using HEAD with git show

```
git show HEAD~3
```

```
commit 39aec30df997f15650037da6a3df291097233ddb (tag: add-report-as-markdown)
Author: Rep Loop <repl@datacamp.com>
Date:   Wed Jul 6 14:36:44 2022 +0000

    Added summary report file.

diff --git a/report.md b/report.md
new file mode 100644
index 0000000..35f4b4d
--- /dev/null
+++ b/report.md
@@ -0,0 +1,3 @@
+# Mental Health in Tech Survey
+TODO: write executive summary.
+TODO: include link to raw data.
```

What changed between two commits?

- `git show` is useful for viewing changes made in a particular commit
- `git diff` compares changes between two commits

What changed between two commits?

- To compare the the fourth and third most recent commits

```
git diff 35f4b4d 186398f
```

or

```
git diff HEAD~3 HEAD~2
```

What changed between two commits?

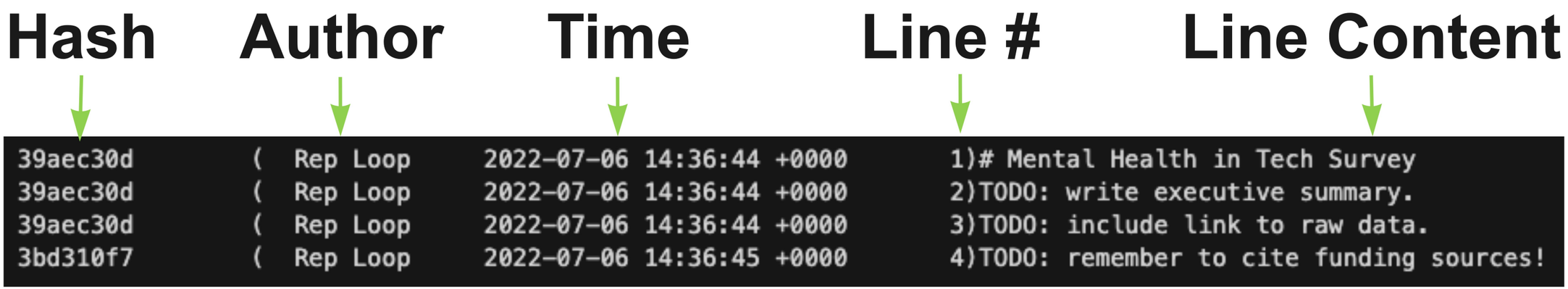
Add a fourth line

```
diff --git a/report.md b/report.md
index 35f4b4d..186398f 100644
--- a/report.md
+++ b/report.md
@@ -1,3 +1,4 @@
 # Mental Health in Tech Survey
 TODO: write executive summary.
 TODO: include link to raw data.
+TODO: remember to cite funding sources!
```

Contents of the new line

Changes per document by line

```
git annotate report.md
```



[1] Image credit: https://unsplash.com/@hannahbusing

Summary

Command	Function
`git show HEAD~1`	Show what changed in the second most recent commit
`git diff 35f4b4d 186398f`	Show changes between two commits
`git diff HEAD~1 HEAD~2`	Show changes between two commits
`git annotate file`	Show line-by-line changes and associated metadata

Let's practice!

INTRODUCTION TO VERSION CONTROL WITH GIT

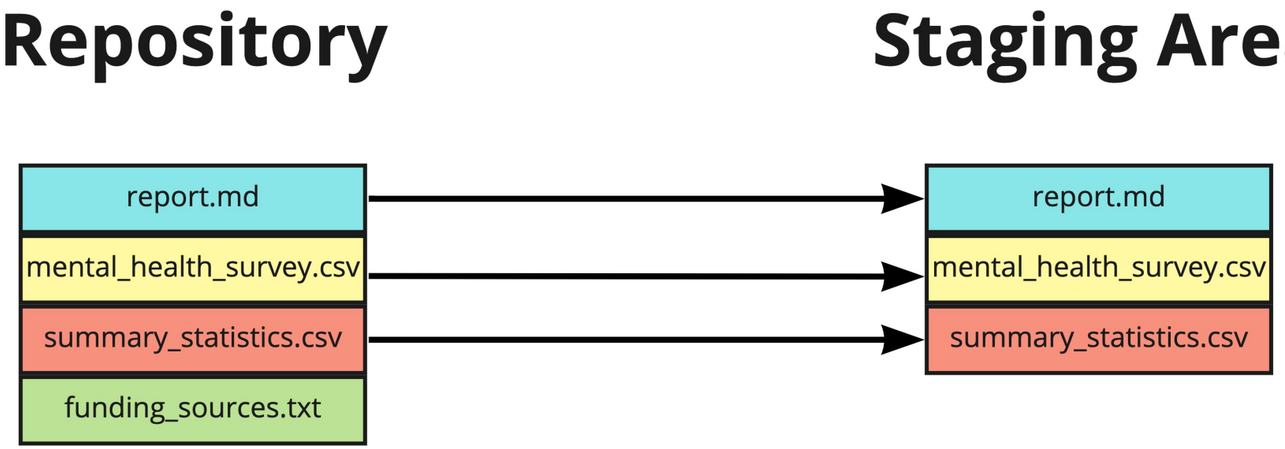
Undoing changes before committing

INTRODUCTION TO VERSION CONTROL WITH GIT

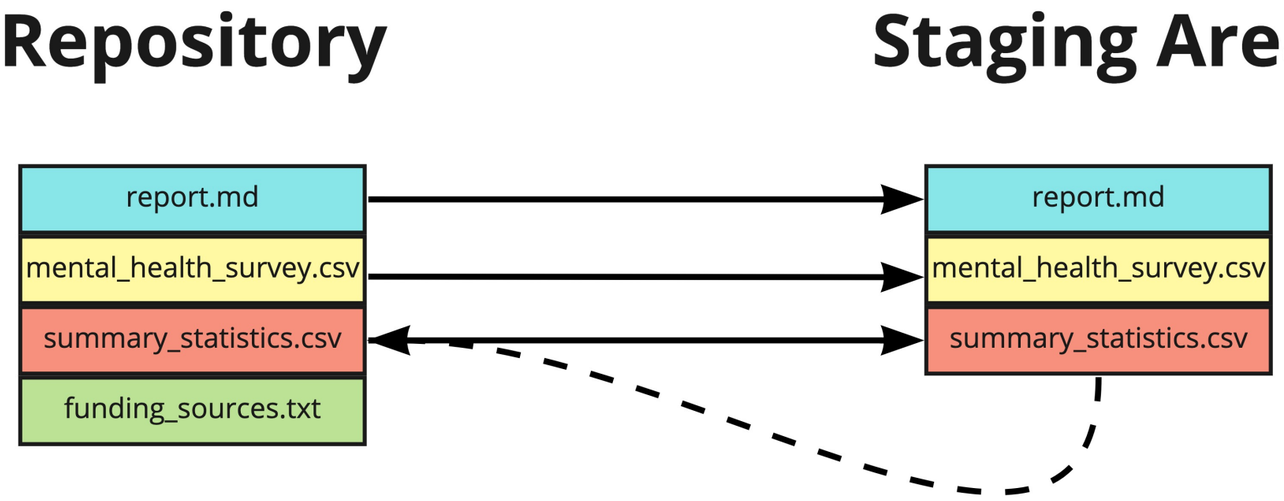
George Boorman
Curriculum Manager, DataCamp

datacamp

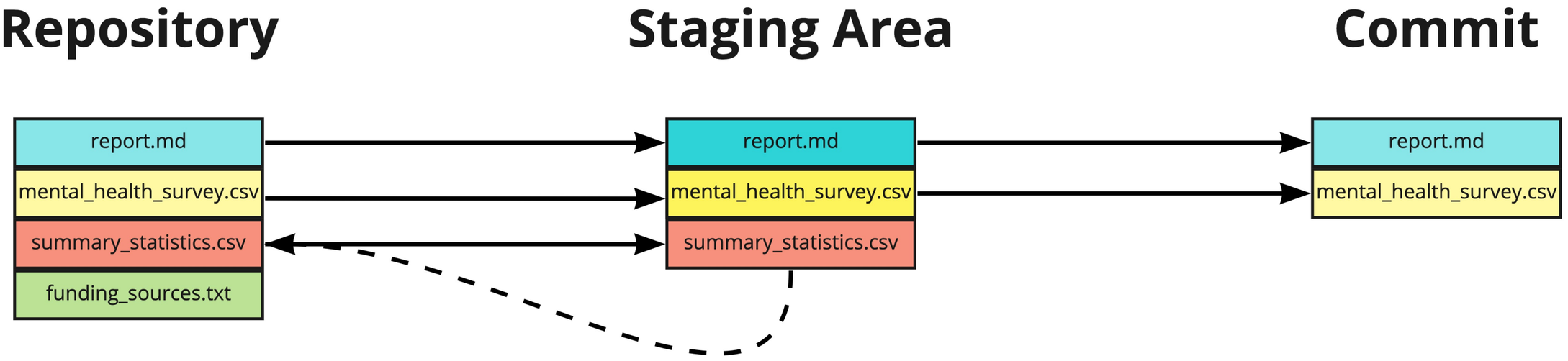
Unstaging a file



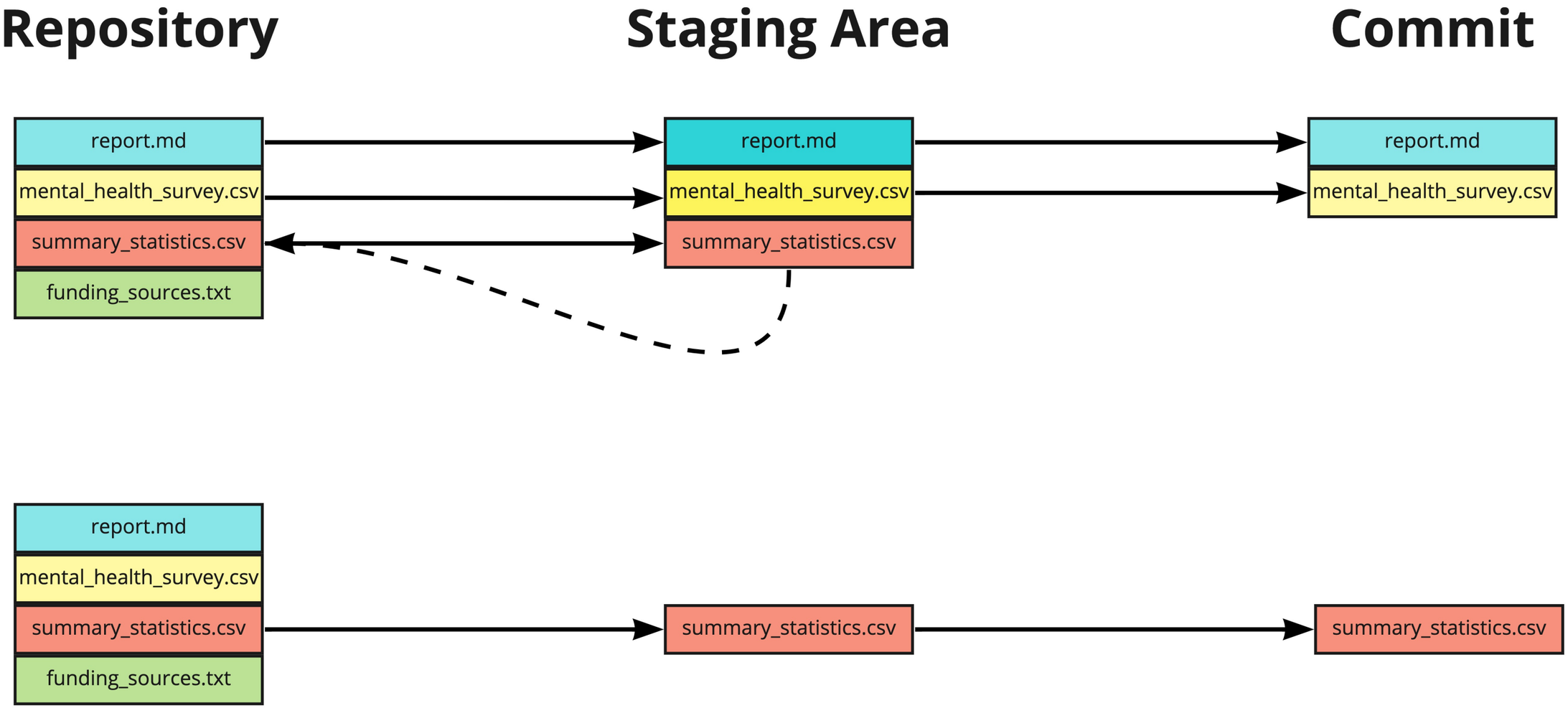
Unstaging a file



Making a commit



Committing the third file



Unstaging a single file in Git

- To unstage a single file:

```
git reset HEAD summary_statistics.csv
```

```
nano summary_statistics.csv
```

```
git add summary_statistics.csv
```

```
git commit -m "Adding age summary statistics"
```

Unstaging all files

- To unstage all files:

```
git reset HEAD
```

Undo changes to an unstaged file

- Suppose we need to undo changes to a file in the repository

```
git checkout -- summary_statistics.csv
```

- `checkout` means switching to a different version
  - Defaults to the last commit
- This means losing all changes made to the unstaged file **forever**

Undo changes to all unstaged files

```
git checkout .
```

- `.` refers to the current directory when used in a shell command
- Undo changes to all unstaged files in the current directory and subdirectories
- This command must be run in the main directory i.e., `mh_survey`

Unstaging and undoing

```
git reset HEAD
```

```
git checkout .
```

```
git add .
```

```
git commit -m "Restore repo to previous commit"
```

Let's practice!

INTRODUCTION TO VERSION CONTROL WITH GIT

Restoring and reverting

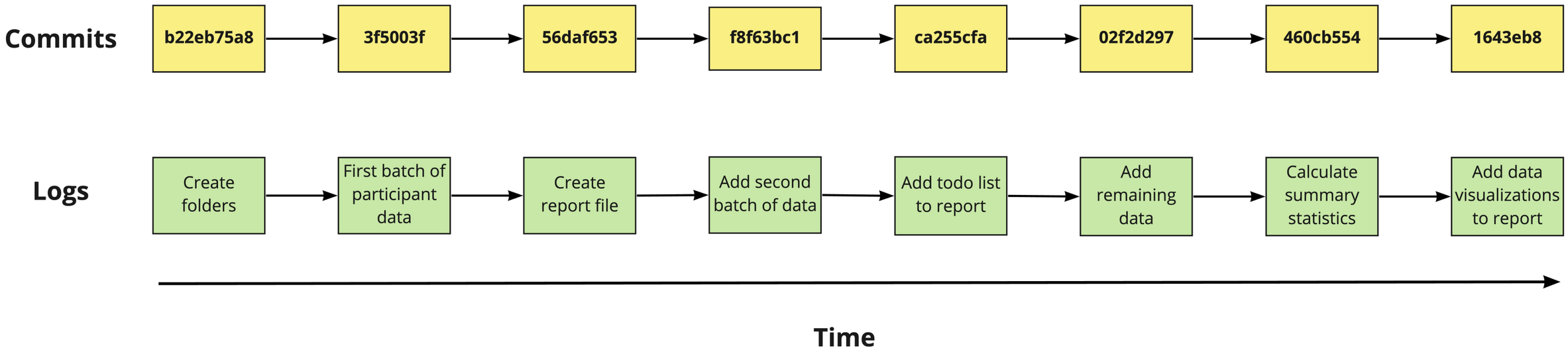
INTRODUCTION TO VERSION CONTROL WITH GIT

George Boorman
Curriculum Manager, DataCamp

datacamp

Project scale

- `git log` is useful to find the commit we want to revert to
- Larger project = more commits = larger output



Customizing the log output

- We can restrict the number of commits displayed using `-` :

```
git log -3
```

- To only look at the commit history of one file:

```
git log -3 report.md
```

Customizing the log output

- Restrict `git log` by date:

```
git log --since='Month Day Year'
```

- Commits since 2nd April 2022:

```
git log --since='Apr 2 2022'
```

- Commits between 2nd and 11th April:

```
git log --since='Apr 2 2022' --until='Apr 11 2022'
```

Restoring an old version of a file

```
git log --since='Jul 6 2022'
```

```
commit dc9d8fac8b314b8bd455e3ede491ab138775ee17
Author: Rep Loop <repl@datacamp.com>
Date:   Wed Jul 6 14:44:31 2022 +0000

    Adding fresh data for the survey.
```

Restoring an old version of a file

```
git checkout -- filename
```

- To revert to a version from a specific commit:

```
git checkout dc9d8fac mental_health_survey.csv
```

- This was the second to last commit, so another approach is:

```
git checkout HEAD~1 mental_health_survey.csv
```

Restoring a repo to a previous state

```
git checkout dc9d8fac
```

- Alternatively:

```
git checkout HEAD~1
```

Cleaning a repository

- See what files are not being tracked:

```
git clean -n
```

- Delete those files:

```
git clean -f
```

-  `git clean -f` cannot be undone!

Let's practice!

INTRODUCTION TO VERSION CONTROL WITH GIT