Viewing the version history

INTRODUCTION TO GIT



George Boorman
Curriculum Manager

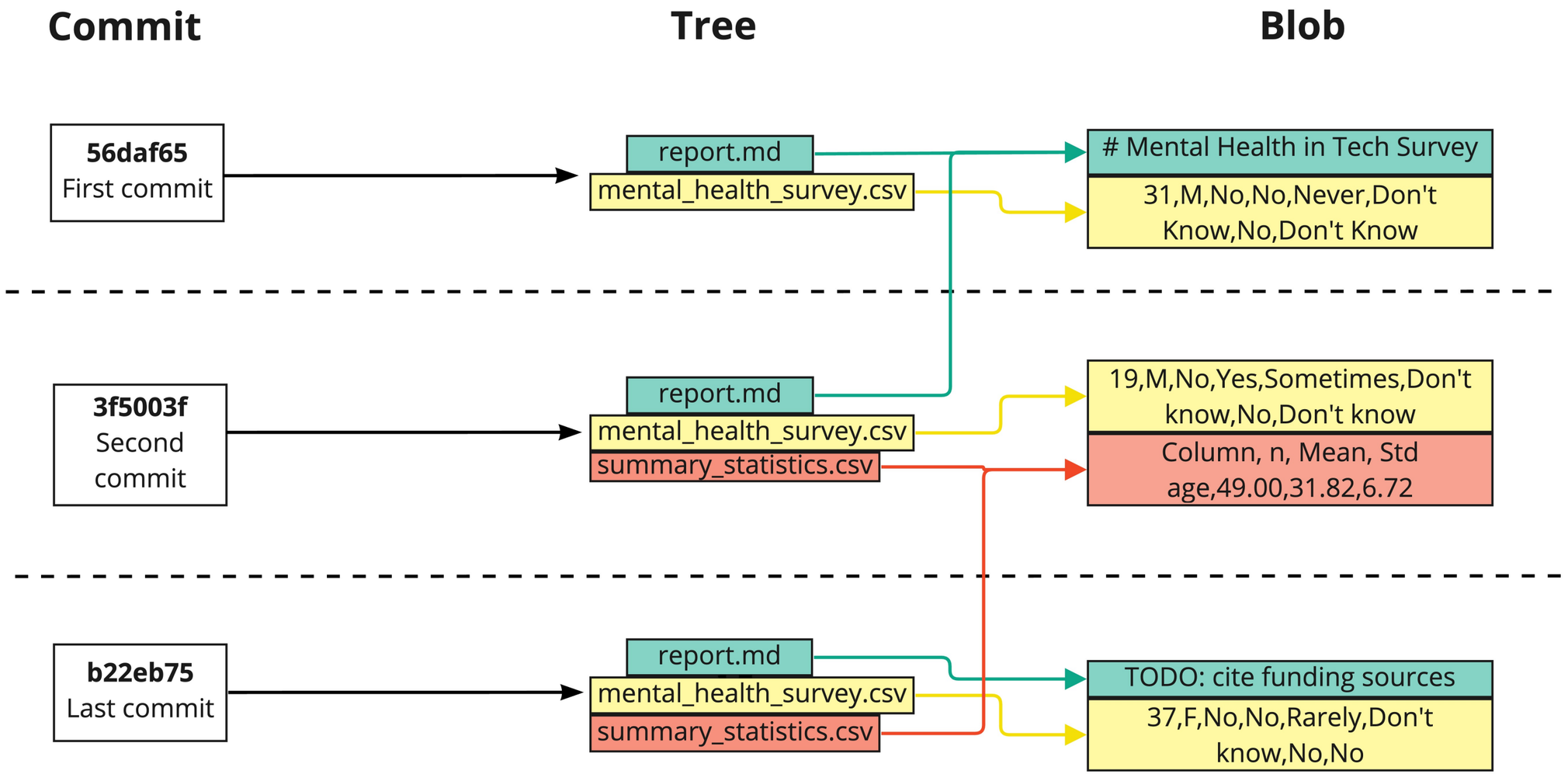
datacamp

The commit structure

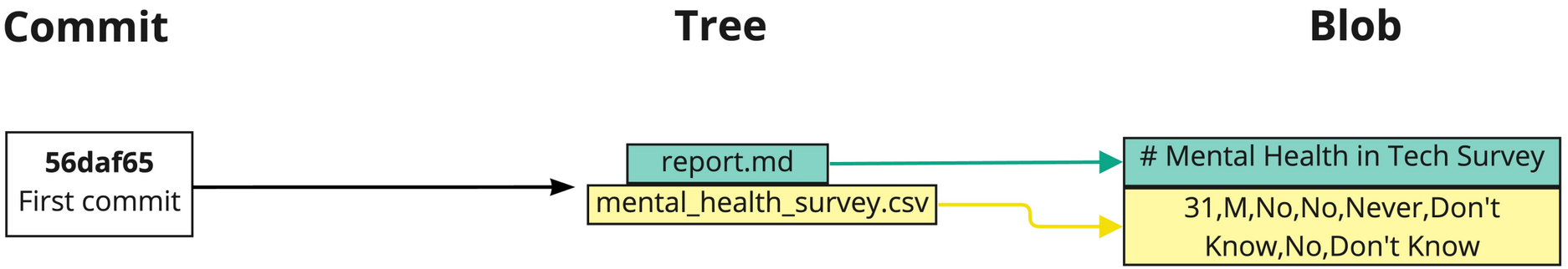
Git commits have **three** parts:

1. **Commit**
   - contains the metadata - author, log message, commit time
2. **Tree**
   - tracks the names and locations of files and directories in the repo
   - like a dictionary - mapping keys to files/directories
3. **Blob**
   - **B**inary **L**arge **OB**ject
   - may contain data of any kind
   - a compressed snapshot of a file's contents

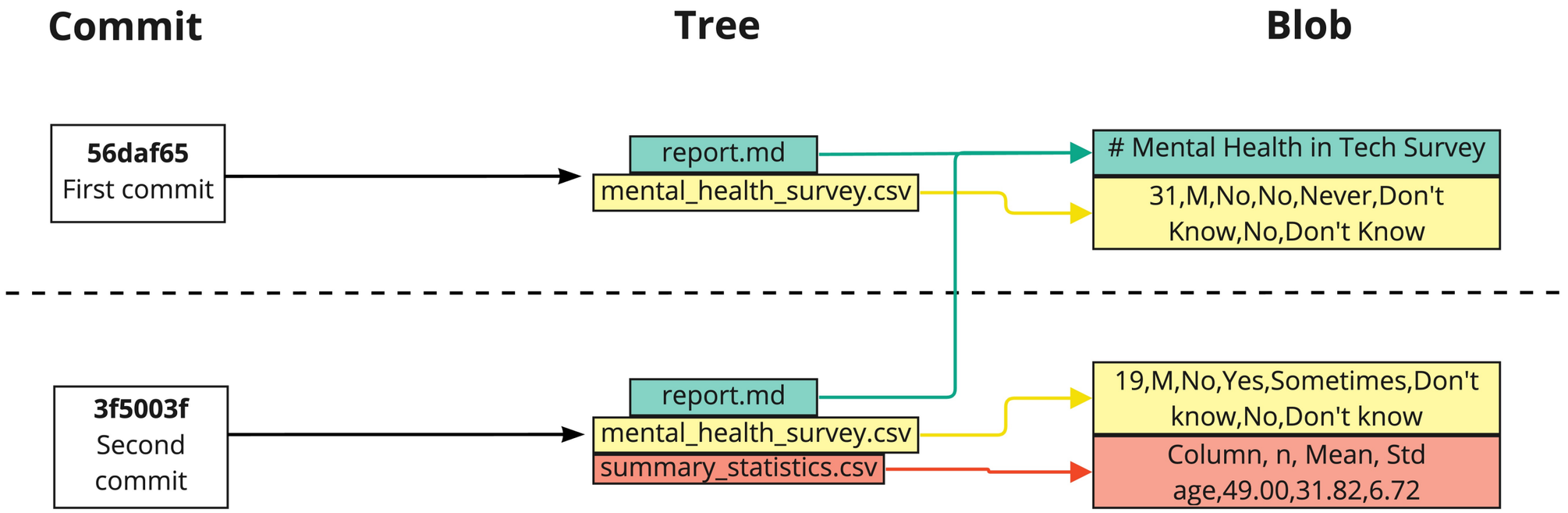
Visualizing the commit structure



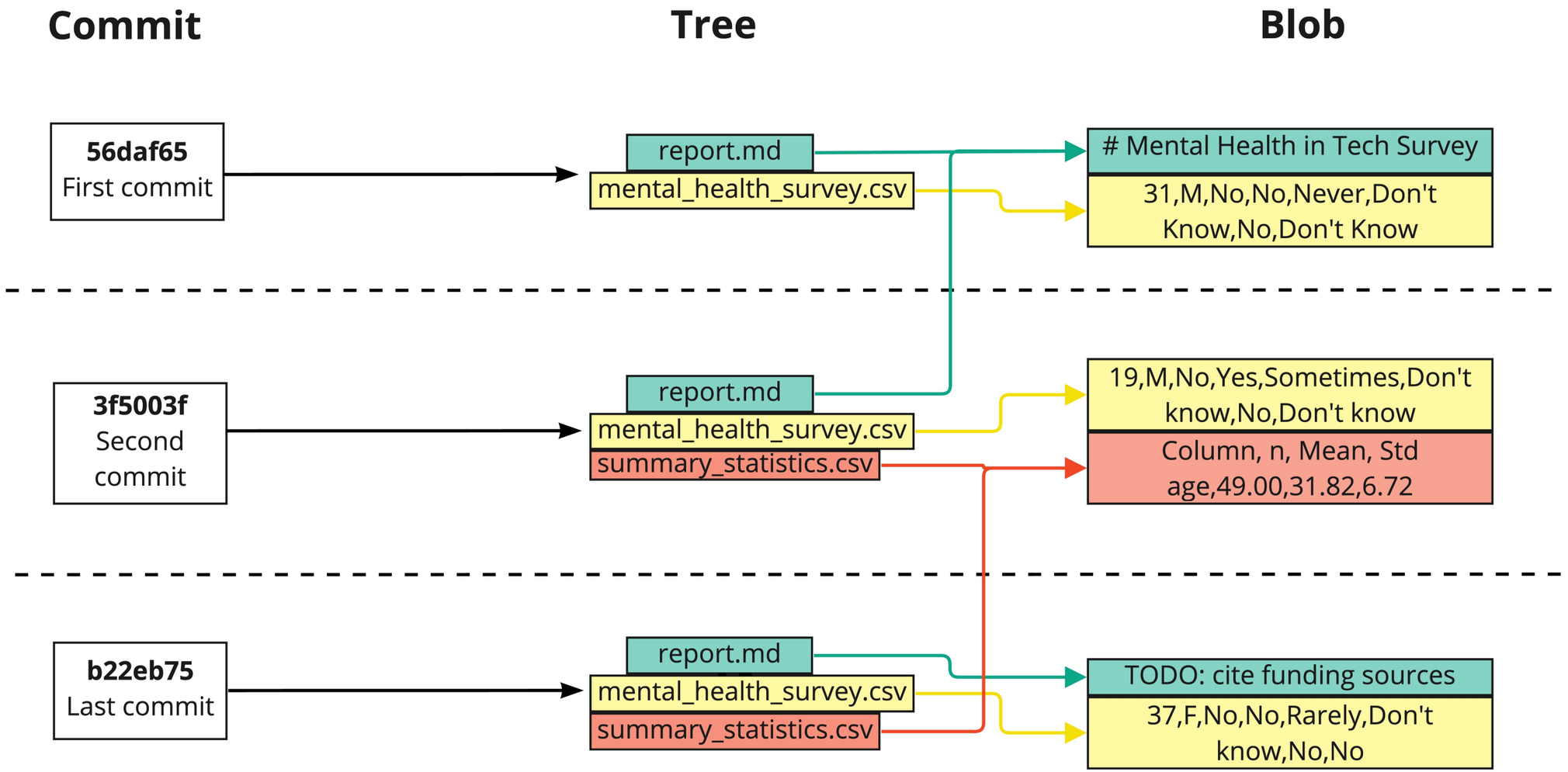
Visualizing the commit structure



Visualizing the commit structure



Visualizing the commit structure



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

Git log

```
git log
```

- Shows commits from newest to oldest

```
commit ad8accfe94cb924444c488132bdef7c54b9bca68
Author: Rep Loop <repl@datacamp.com>
Date:   Wed Jul 24 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

Let's practice!

INTRODUCTION TO GIT

Version history tips and tricks

INTRODUCTION TO GIT

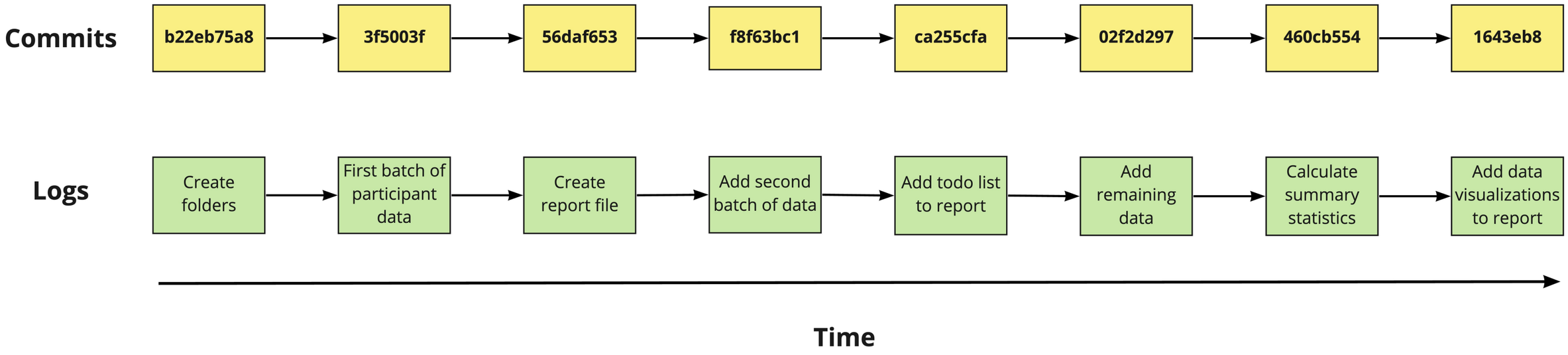


George Boorman
Curriculum Manager, DataCamp

datacamp

Projects grow!

- Larger project = more commits = larger output



Restricting the number of commits

- We can restrict the number of commits displayed using `-` :
- Restrict to the `3` most recent commits

```
git log -3
```

Restricting the file

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

```
git log report.md
```

Combining techniques

```
cd data
```

```
git log -2 mental_health_survey.csv
```

git log output

```
commit f35b9487c063d3facc853c1789b0b77087a859fa
Author: Rep Loop <repl@datacamp.com>
Date:   Fri Jul 26 15:14:32 2024 +0000

    Add two new participants' data.

commit 7f71eadea60bf38f53c8696d23f8314d85342aaf
Author: Rep Loop <repl@datacamp.com>
Date:   Fri Jul 19  09:58:21 2024 +0000

    Adding fresh data for the survey.
```

Customizing the date range

- Restrict `git log` by date:

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

- Commits since 2nd April 2024:

```
git log --since='Apr 2 2024'
```

- Commits between 2nd and 11th April:

```
git log --since='Apr 2 2024' --until='Apr 11 2024'
```

Acceptable filter formats

Natural language

- `"2 weeks ago"`
- `"3 months ago"`
- `"yesterday"`

Date format

- `"07-15-2024"`
  - Recommend ISO Format 6801 `"YYYY-MM-DD"`
  - Check system settings for compatibility, e.g., `12-06-2024` could be `6th Dec` or `12th June`!
- `"15 Jul 2024"` or `"15 July 2024"`
  - Invalid: `"15 Jul, 2024"`

[1] https://www.iso.org/iso-8601-date-and-time-format.html

Finding a particular commit

```
git log
```

- Only need the first 8-10 characters of the `hash`

```
git show c27fa856
```

[1] https://git-scm.com/book/en/v2/Git-Tools-Revision-Selection#Short-SHA-1

git show output

```
commit c27fa85646794b92c5de310395493ebcc3e15cc0 (HEAD -> main)
Author: Rep Loop <repl@datacamp.com>
Date:   Thu Aug 11 07:57:09 2022 +0000

    Adding 50th participant's data

diff --git a/data/mental_health_survey.csv b/data/mental_health_survey.csv
index e034015..17ff40f 100644
--- a/data/mental_health_survey.csv
+++ b/data/mental_health_survey.csv
@@ -48,3 +48,4 @@ age,gender,family_history,treatment,work_interfere,benefits,mental_health_interv
 29,F,No,Yes,Rarely,Don't know,No,Don't know
 23,M,Yes,No,Sometimes,No,No,No
 25,M,Yes,Yes,Sometimes,Yes,No,Don't know
+F,56,Yes,Rarely,No,Don't know,Often,No
```

Log

Diff

Data entry error

Let's practice!

INTRODUCTION TO GIT

Comparing versions

INTRODUCTION TO GIT



George Boorman
Curriculum Manager, DataCamp

datacamp

git show output

```
commit c27fa85646794b92c5de310395493ebcc3e15cc0 (HEAD -> main)
Author: Rep Loop <repl@datacamp.com>
Date:   Thu Aug 11 07:57:09 2022 +0000

    Adding 50th participant's data

diff --git a/data/mental_health_survey.csv b/data/mental_health_survey.csv
index e034015..17ff40f 100644
--- a/data/mental_health_survey.csv
+++ b/data/mental_health_survey.csv
@@ -48,3 +48,4 @@ age,gender,family_history,treatment,work_interfere,benefits,mental_health_interv
 29,F,No,Yes,Rarely,Don't know,No,Don't know
 23,M,Yes,No,Sometimes,No,No,No
 25,M,Yes,Yes,Sometimes,Yes,No,Don't know
+F,56,Yes,Rarely,No,Don't know,Often,No
```

Log

Diff

Data entry error

git diff

- `git diff` - Difference between versions

- Compare last committed version with latest version **not in** the staging area

```
git diff report.md
```

Comparing to an unstaged file

```
diff --git a/report.md b/report.md
index 6218b4e..066f447 100644
--- a/report.md
+++ b/report.md
@@ -1,5 +1,5 @@
 # Mental Health in Tech Survey
-TODO: write executive summary.
 TODO: include link to raw data.
 TODO: add references.
 TODO: add summary statistics.
+TODO: cite funding sources.
```

git diff output

Line changes



```
diff --git a/report.md b/report.md
index 6218b4e..066f447 100644
--- a/report.md
+++ b/report.md
@@ -1,5 +1,5 @@
 # Mental Health in Tech Survey
-TODO: write executive summary.
 TODO: include link to raw data.
 TODO: add references.
 TODO: add summary statistics.
+TODO: cite funding sources.
```

git diff output

Line changes

Line in version a

```
diff --git a/report.md b/report.md
index 6218b4e..066f447 100644
--- a/report.md
+++ b/report.md
@@ -1,5 +1,5 @@
 # Mental Health in Tech Survey
-TODO: write executive summary.
 TODO: include link to raw data.
 TODO: add references.
 TODO: add summary statistics.
+TODO: cite funding sources.
```

git diff output

Line changes

Line in version a

Line in version b

```
diff --git a/report.md b/report.md
index 6218b4e..066f447 100644
--- a/report.md
+++ b/report.md
@@ -1,5 +1,5 @@
 # Mental Health in Tech Survey
-TODO: write executive summary.
 TODO: include link to raw data.
 TODO: add references.
 TODO: add summary statistics.
+TODO: cite funding sources.
```

Comparing to a staged file

- Add `report.md` to the staging area

```
git add report.md
```

- Compare last committed version of `report.md` with the version **in the staging area**

```
git diff --staged report.md
```

Comparing to a staged file

```
diff --git a/report.md b/report.md
index 6218b4e..066f447 100644
--- a/report.md
+++ b/report.md
@@ -1,5 +1,5 @@
 # Mental Health in Tech Survey
-TODO: write executive summary.
 TODO: include link to raw data.
 TODO: add references.
 TODO: add summary statistics.
+TODO: cite funding sources.
```

Comparing multiple staged files

- Compare **all staged files** to versions in the last commit

```
git diff --staged
```

```
diff --git a/mh_tech_survey.csv b/mh_tech_survey.csv
index 4208ed3..d758efb 100644
--- a/mh_tech_survey.csv
+++ b/mh_tech_survey.csv
@@ -47,3 +47,4 @@ age,gender,family_history,treatment,work_interfere,
ntal_health_interv
 28,M,No,Yes,Rarely,Yes,No,Yes
 29,F,No,Yes,Rarely,Don't know,No,Don't know
 23,M,Yes,No,Sometimes,No,No,No
+37,F,No,No,Rarely,Don't know,No,No
diff --git a/report.md b/report.md
index 6218b4e..066f447 100644
--- a/report.md
+++ b/report.md
@@ -1,5 +1,5 @@
 # Mental Health in Tech Survey
-TODO: write executive summary.
 TODO: include link to raw data.
 TODO: add references.
 TODO: add summary statistics.
+TODO: cite funding sources.
```

Comparing two commits

- Find the commit hashes

```
git log
```

- Compare them

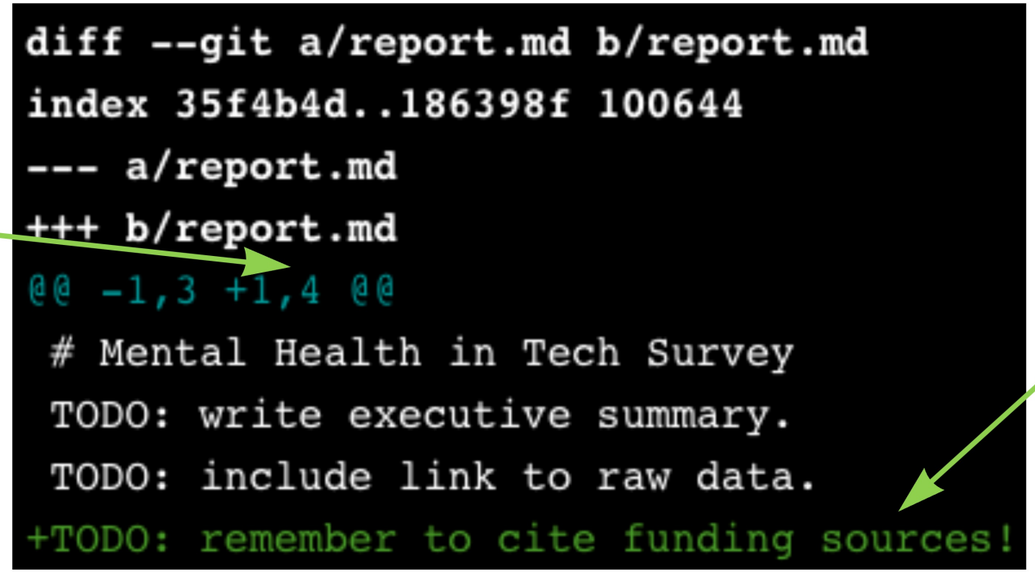
```
git diff 35f4b4d 186398f
```

- What changed **from** first hash **to** second hash
  - Put most recent hash second
- State in latest commit = `HEAD`
- Compare second most recent with the most recent commit

```
git diff HEAD~1 HEAD
```

Comparing two commits

Version B has an extra line



Contents of the new line

Summary

Command	Function
`git diff`	Show changes between all unstaged files and the latest commit
`git diff report.md`	Show changes between an unstaged file and the latest commit
`git diff --staged`	Show changes between all staged files and the latest commit
`git diff --staged report.md`	Show changes between a staged file and the latest commit
`git diff 35f4b4d 186398f`	Show changes between two commits using hashes
`git diff HEAD~1 HEAD~2`	Show changes between two commits using `HEAD` instead of commit hashes

Let's practice!

INTRODUCTION TO GIT

Restoring and reverting files

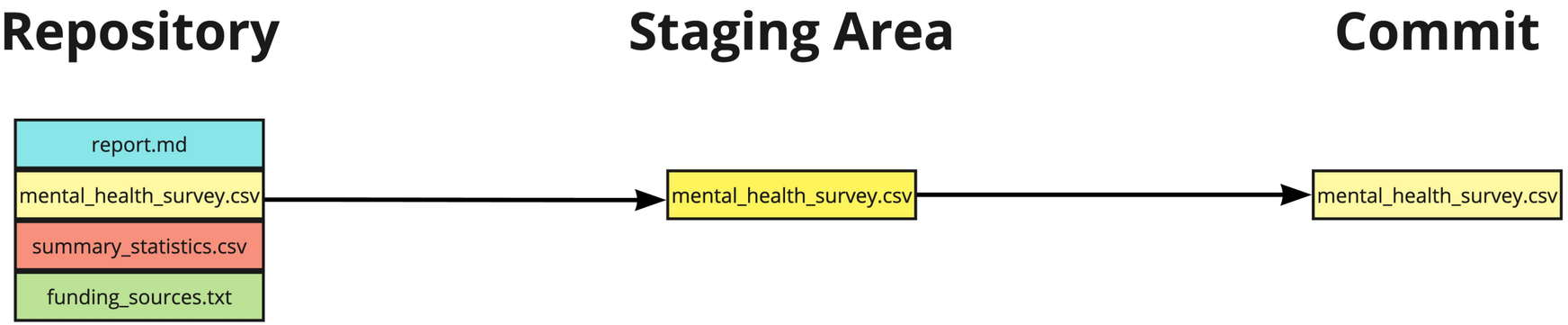
INTRODUCTION TO GIT



George Boorman
Curriculum Manager, DataCamp

datacamp

Making an error



Reverting files

- Restoring a repo to the state prior to the previous commit
- `git revert`
  - Reinstates previous versions and makes a commit
  - Restores **all files updated in the given commit**
  - `a845edcb`, `ebe93178`, etc
  - `HEAD`, `HEAD~1`, etc

```
git revert HEAD
```

Reverting files

```
git revert HEAD
```

```
Revert "Adding fresh data for the survey."

This reverts commit 7f71eadea60bf38f53c8696d23f8314d85342aaf.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Changes to be committed:
#       modified:   data/mental_health_survey.csv
#
```

- Save: `Ctrl + O` , then `Enter`
- Exit: `Ctrl + X`

Reverting files

```
[main 7d11f79] Revert "Adding fresh data for the survey."
 Date: Tue Jul 30 14:17:56 2024 +0000
 1 file changed, 3 deletions(-)
```

git revert flags

- Avoid opening the text editor

```
git revert --no-edit HEAD
```

- Revert without committing (bring files into the staging area)

```
git revert -n HEAD
```

Revert a single file

- `git revert` works on commits, not individual files
- To revert a single file:
  - `git checkout`
  - Use commit hash or `HEAD` syntax

```
git checkout HEAD~1 -- report.md
```

Checking the checkout

```
git status
```

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)

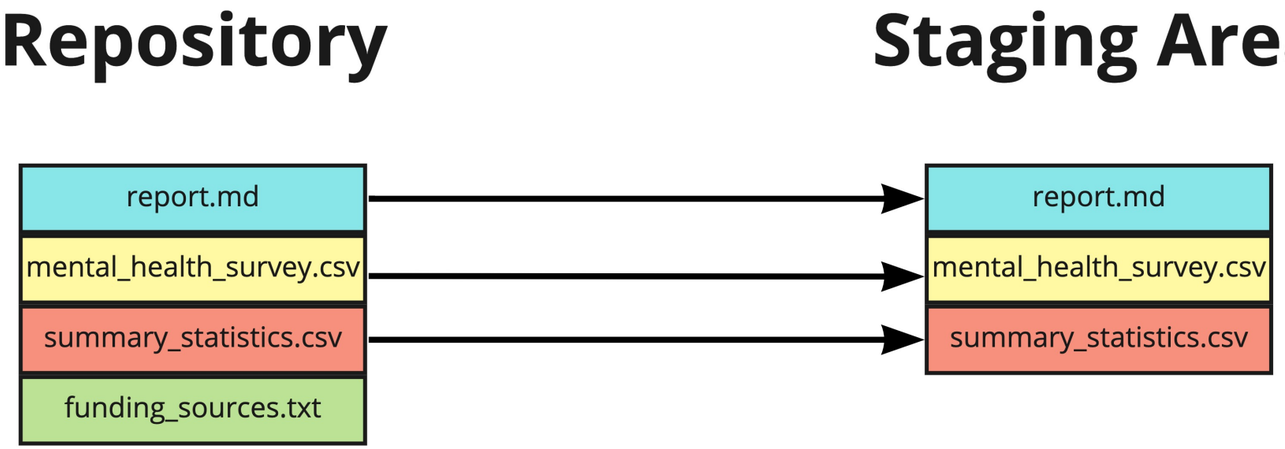
        modified:   report.md
```

Making a commit

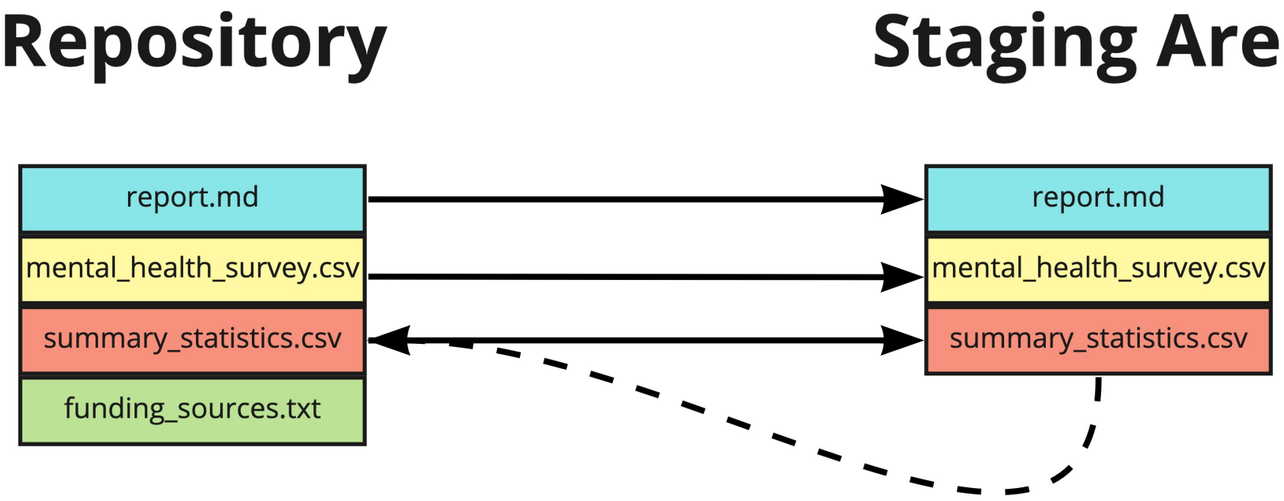
```
git commit -m "Checkout previous version of report.md"
```

```
[main daa6c87] Checkout previous version of report.md
 1 file changed, 1 deletion(-)
```

Unstaging a file



Unstaging a file



Unstaging a single file

- To unstage a single file:

```
git restore --staged summary_statistics.csv
```

- Edit the file

```
git add summary_statistics.csv
```

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

Unstaging all files

- To unstage all files:

```
git restore --staged
```

Summary

Command	Result
`git revert HEAD`	Revert all files from a given commit
`git revert HEAD --no-edit`	Revert without opening a text editor
`git revert HEAD -n`	Revert without making a new commit
`git checkout HEAD~1 -- report.md`	Revert a single file from the previous commit
`git restore --staged report.md`	Remove a single file from the staging area
`git restore --staged`	Remove all files from the staging area

Let's practice!

INTRODUCTION TO GIT

Congratulations

INTRODUCTION TO GIT



George Boorman
Curriculum Manager, DataCamp

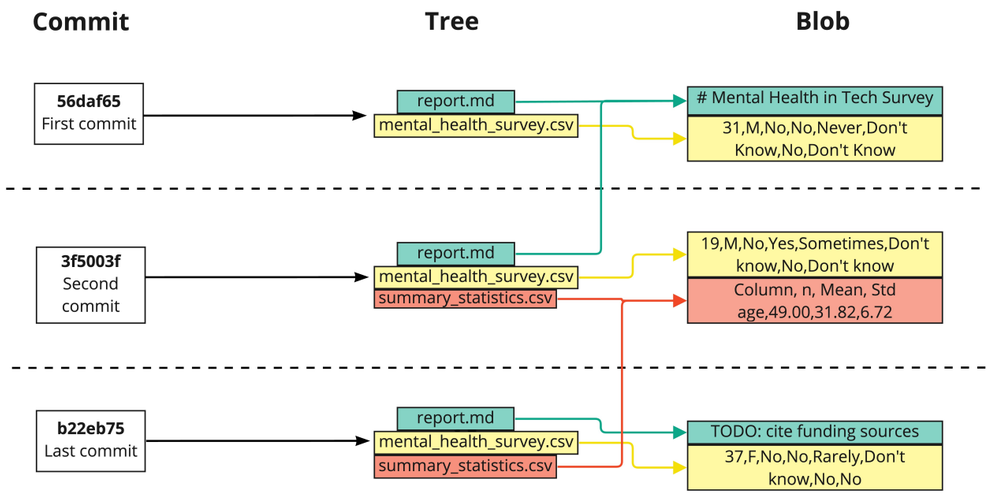
datacamp

Chapter 1 recap

- Benefits and applications of Git for version control
- Navigating the terminal - `ls` , `cd`
- How to initiate a Git project - `git init`
- How to use Git to track your files - `git add .` , `git commit -m`

Chapter 2 recap

- How Git stores data
  1. **Commit**
  2. **Tree**
  3. **Blob**
- `git log`
  - `3`
  - `--since`
  - `--until`
- `git show c27fa856`



Chapter 2 recap

Command	Function
`git diff report.md`	Show changes between unstaged file and the latest commit
`git diff --staged report.md`	Show changes between staged file and the latest commit
`git diff 35f4b4d 186398f`	Show changes between two commits using commit hashes
`git diff HEAD~1 HEAD~2`	Show changes between two commits using HEAD syntax

Chapter 2 recap

Command	Result
`git revert HEAD`	Revert all files from a given commit
`git revert HEAD --no-edit`	Revert without opening a text editor
`git revert HEAD -n`	Revert without making a new commit
`git checkout HEAD~1 -- report.md`	Revert a single file from the previous commit
`git restore --staged report.md`	Remove a single file from the staging area
`git restore --staged`	Remove all files from the staging area

What's next?

- Branches
- Remote repos
- Rebasing
- Bisecting
- Submodules

Congratulations

INTRODUCTION TO GIT