Extracted from:

Pragmatic Guide to Git

The Pragmatic Bookshelf

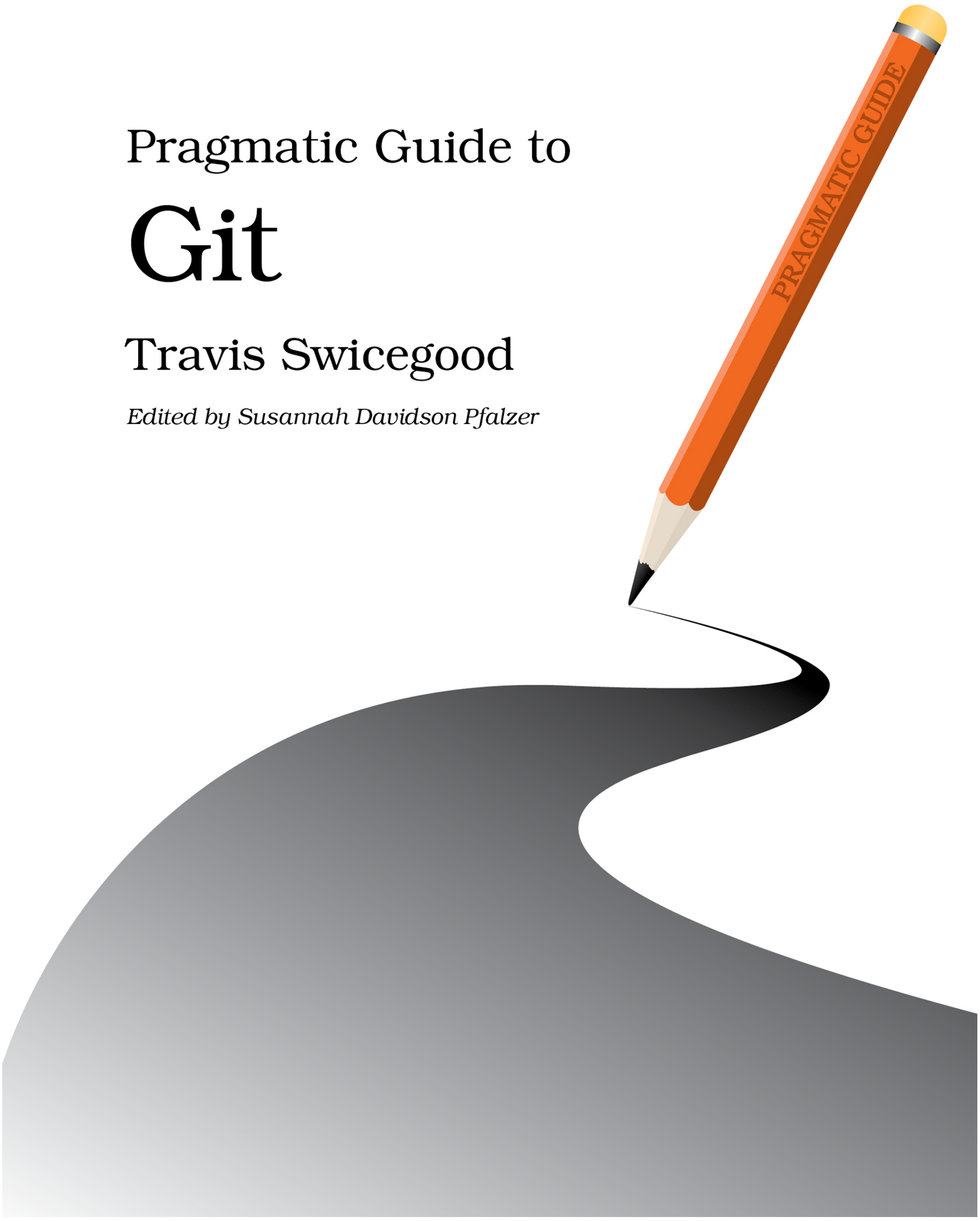
Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic Guide to Git

Travis Swicegood

Edited by Susannah Davidson Pfalzer



The team that produced this book includes:

Susannah Davidson Pfalzer (editor)
Potomac Indexing, LLC (indexer)
Kim Wimpsett (copyeditor)
Steve Peter (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-934356-72-2
Printed on acid-free paper.
Book version: P3.0—January 2012

33 Generating Statistics About Changes

The individual commit's changes are important, but viewing those changes in aggregate through statistics can provide you with a unique view of the project. Git helps you do that through the various statistical outputs it generates.

Git's git diff --stat is the most straightforward of the displays. It takes one or two commits—remember, git diff assumes HEAD as its second commit if you don't specify it—and displays stats regarding the changes rather than displaying the diff output. It includes file-by-file changes in addition to the summary statistics. Git uses the diffstat command for this, so the output may look familiar if you're used to that program.

Sometimes all you need is the final line of the stats output—the number of files changed, the number of insertions, and the number of deletions. Use --shortstat to display that information. It's the same information displayed at the last line of the main --stat output but without the file-by-file breakdown.

The output generated by the --stat and --shortstat parameters is not easily parseable by a computer. You can use the --numstat parameter to generate a three-column output that is easy to parse. The first column is the number of inserted lines, the second is the number of deleted lines, and the final column is the name of the file in question.

You can generate stats using any range of revisions that work with git diff. Adding any of the stat parameters tells Git to output the stats without showing the actual differences. Add -p to the command to show both the statistical output and patch or to show the differences.

We've talked about viewing statistical information in the context of git diff, but you can also view it from git log along with each log message. Add --stat, --shortstat, or --numstat to any git log command to add the respective statistical output to the log output.

➤ Show change stats between the last ten commits.

```
prompt> git diff --stat HEAD~10
... or ...
prompt> git diff --stat HEAD~10 HEAD
```

➤ Show statistics between two commits.

```
prompt> git diff --stat first  second
... example ...
prompt> git diff --stat 423d021 1e85ac3
```

➤ Show the number of files changed, inserts, and deletes in the last ten commits.

```
prompt> git diff --shortstat HEAD~10
```

➤ Show stats in a parseable format.

```
prompt> git diff --numstat HEAD~10
```

➤ Show the patch in addition to the statistical information.

```
prompt> git diff --stat -p HEAD^
```

➤ Show statistics about commits in the log.

For full stats by file:

```
prompt> git log --stat
```

To display cumulative stats only, use this:

```
prompt> git log --shortstat
```

Related Tasks:

- Task 30, *Viewing the Log,* on page ?
- Task 32, *Comparing Differences,* on page ?

34 Assigning Blame

Despite its combative name, `git blame` is a useful tool for determining what the original developer was thinking. Most bugs manifest themselves with an error at a specific point. You can use `git blame` to find out when the problem line was introduced into the repository and use that as a jumping-off point for further investigation.

`git blame` displays all or a portion of a file with annotations showing when the change was made, by who, and, more importantly, in what revision the change was made. Armed with that, you can inspect the log to determine what the original author intended.

`git blame` outputs the following information:

- Short commit ID
- Author's name
- Date and time of commit
- Line number

By default, the entire file is displayed. You can limit the portion of the file displayed by using the `-L` parameter. It requires one parameter: a number or POSIX regular expression.

You can specify the point to stop, as well, by providing a second value as part of a comma-separated string. Make sure there's no space between the start, the comma, and the second value. The second value can be another line number, a regular expression, or a number with a plus (+) or minus (-) before it.

The plus sign shows the start plus the number of lines; the minus sign adjusts the start to show the number of lines before the start. Remember that the plus and minus are zero-indexed. For example, `-L 10,+10` shows lines 10 through 19, not lines 10 through 20.

Git can track content that moves around in a file or is copied from one file to another. You can use `git blame` to show content that has moved around by adding the `-M` parameter.

You can also track changes copied from another file by using the `-C` parameter. It checks the changes in the file against other changes in the repository to see whether it was copied from somewhere else.

➤ Display file with entire line-by-line commit information.

```
prompt> git blame some/file
```

➤ Start the output of blame at line 10.

```
prompt> git blame -L 10 some/file
```

➤ Limit the output of blame to lines 10 through 20.

```
prompt> git blame -L 10,20 some/file
... or ...
prompt> git blame -L 10,+11 some/file
... or ...
prompt> git blame -L 20,-11 some/file
```

➤ Show ten lines of output from blame starting at a POSIX regular expression.

```
prompt> git blame -L "/def to_s/",+10 some/file
```

➤ Check the history to see whether the change was moved within the file, and display that information.

```
prompt> git blame -M some/file
```

➤ Check the history to see whether the change was copied from somewhere else or moved around within the file, and display that information.

```
prompt> git blame -C some/file
```

Related Tasks:

- Task 30, *Viewing the Log*, on page ?
- Task 32, *Comparing Differences*, on page ?