Ignoring Things

What if we have files that we do not want Git to track for us, like backup files created by our editor or intermediate files created during data analysis? Let's create a few dummy files:

```
$ mkdir results
$ touch a.dat b.dat c.dat results/a.out results/b.out
```

and see what Git says:

```
$ git status
```

Output:

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    a.dat
    b.dat
    c.dat
    results/

nothing added to commit but untracked files present (use "git add" to track)
```

Putting these files under version control would be a waste of disk space. What's worse, having them all listed could distract us from changes that actually matter, so let's tell Git to ignore them.

We do this by creating a file in the root directory of our project called `.gitignore`:

```
$ nano .gitignore
$ cat .gitignore
```

Output:

```
*.dat
results/
```

These patterns tell Git to ignore any file whose name ends in `.dat` and everything in the `results` directory. (If any of these files were already being tracked, Git would continue to track them.)

Once we have created this file, the output of `git status` is much cleaner:

```
$ git status
```

Output:

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

The only thing Git notices now is the newly-created `.gitignore` file. You might think we wouldn't want to track it, but everyone we're sharing our repository with will probably want to ignore the same things that we're ignoring. Let's add and commit `.gitignore`:

```
$ git add .gitignore
$ git commit -m "Ignore data files and the results folder."
$ git status
```

Output:

```
On branch master
nothing to commit, working directory clean
```

As a bonus, using `.gitignore` helps us avoid accidentally adding to the repository files that we don't want to track:

```
$ git add a.dat
```

Output:

```
The following paths are ignored by one of your .gitignore files:
a.dat
Use -f if you really want to add them.
```

If we really want to override our ignore settings, we can use `git add -f` to force Git to add something. For example, `git add -f a.dat`. We can also always see the status of ignored files if we want:

```
$ git status --ignored
```

Output:

```
On branch master
Ignored files:
 (use "git add -f <file>..." to include in what will be committed)

        a.dat
        b.dat
        c.dat
        results/

nothing to commit, working directory clean
```

Ignoring Nested Files

Given a directory structure that looks like:

```
results/data
results/plots
```

How would you ignore only `results/plots` and not `results/data`?

Solution

If you only want to ignore the contents of `results/plots`, you can change your `.gitignore` to ignore only the `/plots/` subfolder by adding the following line to your .gitignore:

```
results/plots/
```

This line will ensure only the contents of `results/plots` is ignored, and not the contents of `results/data`.

As with most programming issues, there are a few alternative ways that one may ensure this ignore rule is followed. The "Ignoring Nested Files: Variation" exercise has a slightly different directory structure that presents an alternative solution. Further, the discussion page has more detail on ignore rules.

Including Specific Files

How would you ignore all `.dat` files in your root directory except for `final.dat`? Hint: Find out what `!` (the exclamation point operator) does

Solution

You would add the following two lines to your .gitignore:

```
*.dat           # ignore all data files
!final.dat      # except final.data
```

The exclamation point operator will include a previously excluded entry.

Note also that because you've previously committed `.dat` files in this lesson they will not be ignored with this new rule. Only future additions of `.dat` files added to the root directory will be ignored.

Ignoring Nested Files: Variation

Given a directory structure that looks similar to the earlier Nested Files exercise, but with a slightly different directory structure:

```
results/data
results/images
results/plots
results/analysis
```

How would you ignore all of the contents in the results folder, but not `results/data`?

Hint: think a bit about how you created an exception with the `!` operator before.

Solution

If you want to ignore the contents of `results/` but not those of `results/data/`, you can change your `.gitignore` to ignore the contents of results folder, but create an exception for the contents of the `results/data` subfolder. Your .gitignore would look like this:

```
results/*               # ignore everything in results folder
!results/data/          # do not ignore results/data/ contents
```

Ignoring all data Files in a Directory

Assuming you have an empty .gitignore file, and given a directory structure that looks like:

```
results/data/position/gps/a.dat
results/data/position/gps/b.dat
results/data/position/gps/c.dat
results/data/position/gps/info.txt
results/plots
```

What's the shortest `.gitignore` rule you could write to ignore all `.dat` files in `result/data/position/gps`? Do not ignore the `info.txt`.

Solution

Appending `results/data/position/gps/*.dat` will match every file in `results/data/position/gps` that ends with `.dat`. The file `results/data/position/gps/info.txt` will not be ignored.

The Order of Rules

Given a `.gitignore` file with the following contents:

```
*.dat
!*.dat
```

What will be the result?

Solution

> The `!` modifier will negate an entry from a previously defined ignore pattern. Because the `!*.dat` entry negates all of the previous `.dat` files in the `.gitignore`, none of them will be ignored, and all `.dat` files will be tracked.

Log Files

You wrote a script that creates many intermediate log-files of the form `log_01`, `log_02`, `log_03`, etc. You want to keep them but you do not want to track them through `git`.

1. Write **one** `.gitignore` entry that excludes files of the form `log_01`, `log_02`, etc.
2. Test your "ignore pattern" by creating some dummy files of the form `log_01`, etc.
3. You find that the file `log_01` is very important after all, add it to the tracked files without changing the `.gitignore` again.
4. Discuss with your neighbor what other types of files could reside in your directory that you do not want to track and thus would exclude via `.gitignore`.

Solution

1. append either `log_*` or `log*` as a new entry in your .gitignore
2. track `log_01` using `git add -f log_01`

Key Points

- The `.gitignore` file tells Git what files to ignore.