Version Control

Outline

- Brief overview
    - What it is
    - Why to use it
- Main topic: What (not) to save
- Discussion

Version Control

- Maintain different *versions* of your project, e.g.,
    - Stable
    - Development

Version Control

- Maintain different *versions* of your project, e.g.,
    - Stable
    - Development

- Maintain a *history* of your project

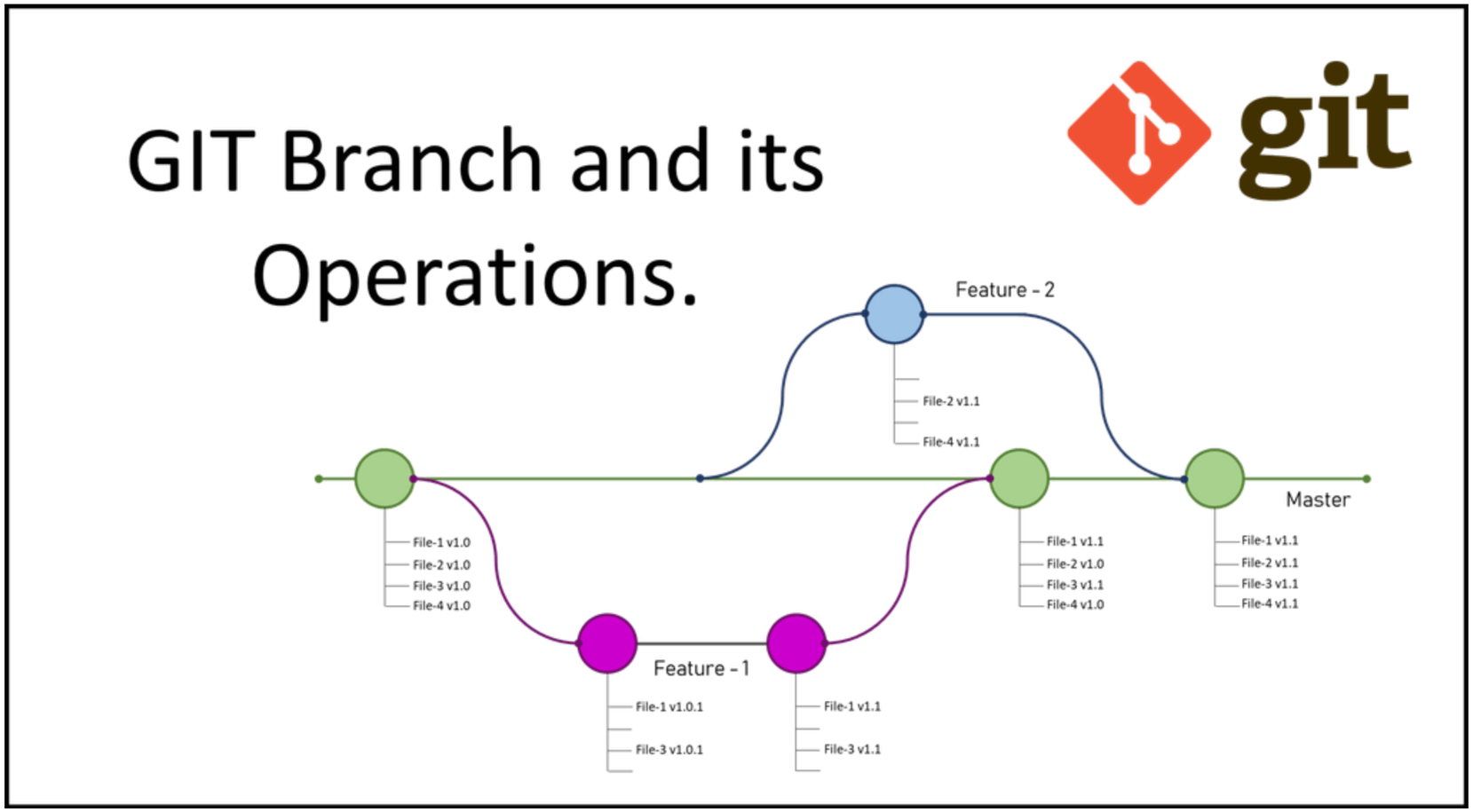
Version Control

- Maintain different *versions* of your project, e.g.,
    - Stable
    - Development

- Maintain a *history* of your project

- Standard: git

Version Control



Version control is for *you*

Our goals:

1. Exactly reproducible
2. User friendly
3. Transparent
4. Reusable
5. Archived
6. Version controlled

Version control is for *you*

Our goals:

1. Exactly reproducible
2. User friendly
3. Transparent
4. Reusable
5. Archived
6. Version controlled

- Most of these are for *sharing* your analysis
- Version control is mainly *for you*

It’s worth it

- Git is powerful, and so also fairly complex
- But the basic functionality is pretty simple
- It is well worth learning, and using regularly
- Many, many tutorials online
    - See, e.g., Karl Broman's: https://kbroman.org/github_tutorial/

Main topic: What (not) to save

An inherent trade-off

You don't normally save every file in your git repository.

An inherent trade-off

You don't normally save every file in your git repository.

- If you saved everything you did -- every edit, every plot, etc. -- your git archive would balloon in size

An inherent trade-off

You don't normally save every file in your git repository.

- If you saved everything you did -- every edit, every plot, etc. -- your git archive would balloon in size

- If you don't save enough, you may not be able to piece together what you did

An inherent trade-off

You don't normally save every file in your git repository.

- If you saved everything you did -- every edit, every plot, etc. -- your git archive would balloon in size

- If you don't save enough, you may not be able to piece together what you did

What should you keep?

An inherent trade-off

You don't normally save every file in your git repository.

- If you saved everything you did -- every edit, every plot, etc. -- your git archive would balloon in size

- If you don't save enough, you may not be able to piece together what you did

What should you keep?

Note:

- A `.gitignore` file lets you specify what (not) to save
- It is important to set this up at the outset of your project.

  Once a file has been committed to your repository, it generally can't be removed.

Git – designed for text files

- When editing text files, git only saves *differences*

  This makes archiving simple text files very space efficient
- Other files get completely re-saved at each commit

What to keep (track)

- markdown files
- scripts
- makefiles
- simple text documentation
- etc.

What to ignore

- binary files
    - pdf, jpg, etc.
- data
- automatically generated text files
    - latex `.aux`, `.log`, etc.
- **notebook files that include output**
    - `.ipynb`
    - `.html`

What to ignore

- binary files
    - pdf, jpg, etc.
- data
- automatically generated text files
    - latex `.aux`, `.log`, etc.
- **notebook files that include output**
    - `.ipynb`
    - `.html`

- **Use jupytext**
- Only track the markdown file, not the `.ipynb`

But do save your notebooks!

- Notebooks save code and output together (`.ipynb` or `.html`)
- It is *highly valuable* to archive notebooks
- Just not with every git commit!

But do save your notebooks!

- Notebooks save code and output together (`.ipynb` or `.html`)
- It is *highly valuable* to archive notebooks
- Just not with every git commit!

- Develop some other strategy for saving notebooks, e.g., a special directory where you put copies of "milestone" notebooks
- Include helpful archival information within your notebook, e.g.,
  - `date()`
  - `file.info(list.files(recursive=T))`
  - `installed.packages()`

Discussion

- What strategy would work *for you* to archive "milestone notebooks"?
- Examples:
    - Have a special directory where you put notebooks (when?)
    - Instead of sending plots by email, send a notebook

Discussion

- What strategy would work *for you* to archive "milestone notebooks"?
- Examples:
    - Have a special directory where you put notebooks (when?)
    - Instead of sending plots by email, send a notebook

- Realistically, what would be your biggest obstacle to actually following that strategy?

Discussion

- What strategy would work *for you* to archive "milestone notebooks"?
- Examples:
    - Have a special directory where you put notebooks (when?)
    - Instead of sending plots by email, send a notebook

- Realistically, what would be your biggest obstacle to actually following that strategy?
- What might you do to lessen that obstacle?