



# Coding: Best Practices & Tools

Towards painless programming for researchers

---

L. MANZARI – M. GABORIT

MWL Lunch Seminar – June 1, 2016

# Contents

1. Better practices in coding: why?
2. Editors & IDEs
3. Versioning for fun and profit
4. Documentation is not a myth
5. What's next?

Better practices in coding: why?

---

## Better practices in coding: why?

*Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.*

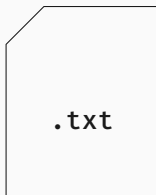
— D. Knuth

## Editors & IDEs

---

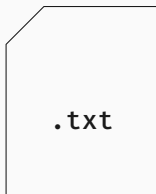
# File types in research

# File types in research

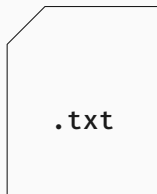


data

# File types in research



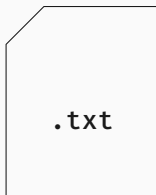
data



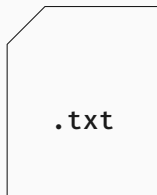
config



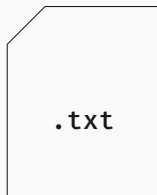
## File types in research



data

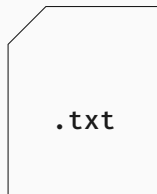


config

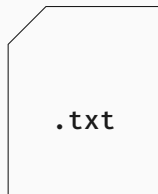


$\text{\LaTeX}$

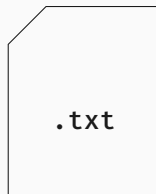
## File types in research



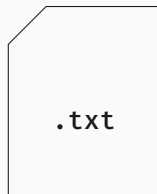
data



config

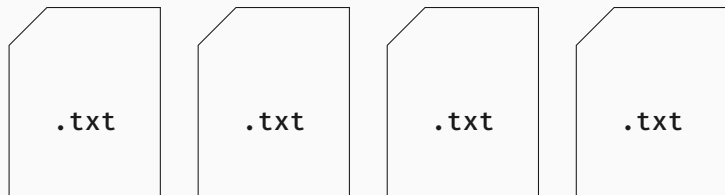


$\text{\LaTeX}$



code

## File types in research



data

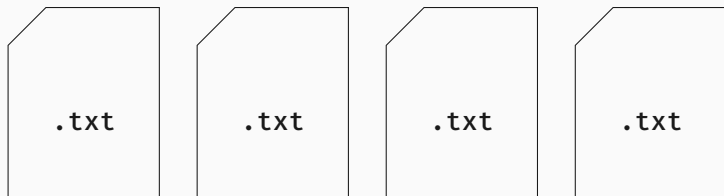
config

$\text{\LaTeX}$

code

These are all **text files**...

## File types in research



data

config

TEX

code

These are all **text files**...  
...let's use **a good text editor**!

# What makes a text editor *good*?

A good text editor has to be...

- Transparent
- Intelligent
- Customizable
- Extensible

# All the fuss about Vlm and GNU Emacs

- Pros**
  - learn once, use forever
  - available everywhere
  - (im)proved with time
- Cons**
  - at the beginning, intimidating

## Characteristics

- It's not a text editor, it's a Lisp interpreter
- Highly context-dependent behavior

## Characteristics

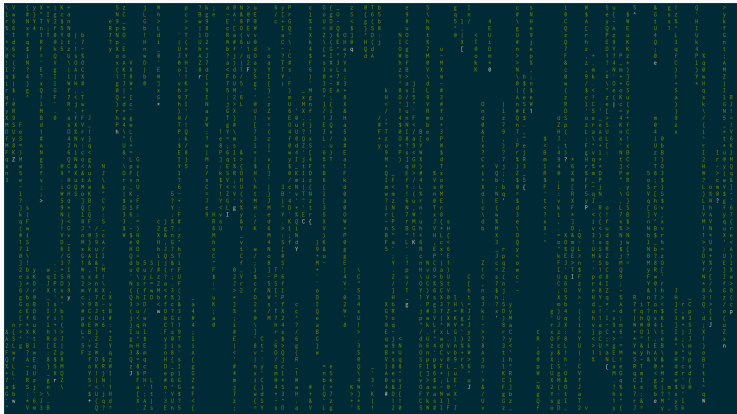
- It's not a text editor, it's a Lisp interpreter
- Highly context-dependent behavior

## How to get started

1. use it as a regular text editor
2. customize according to need
3. extend it using packages



What people think it is...



Editors: VIm

## What it does actually look like

[illegible]

## Characteristics

- Command line or GUI
- Modal (one mode per class of action : edition, displacement, selection, etc...)
- Based on shortcuts and commands
- Fully customizable through plugin or embedded language
- Painless integration with any standard UNIX toolchain

## Where to start

- `http://vim-adventures.com/ & $ vimtutor`
- `http://vimcasts.org/`
- `https://vimebook.com/en`

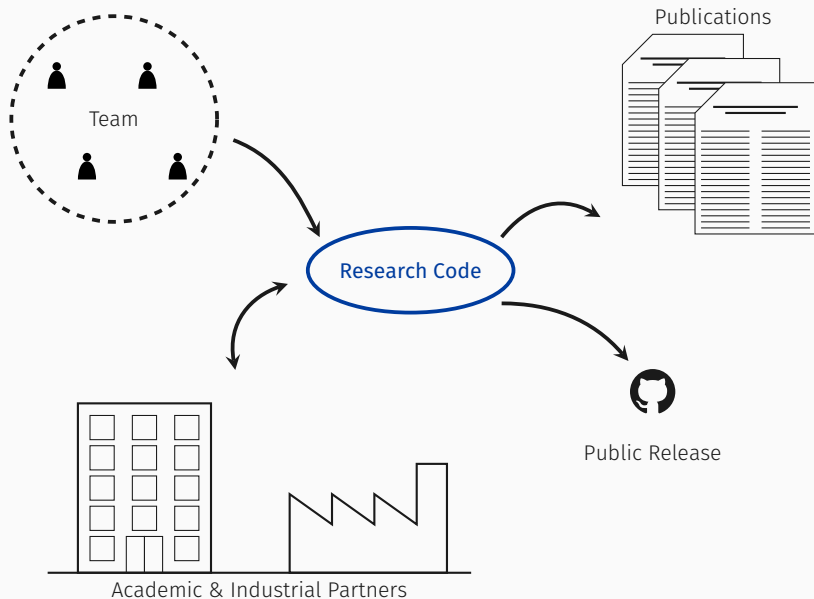
Multi-platform, open-source and actively maintained

- Atom [atom.io](https://atom.io)
- Light Table [lighttable.com](https://lighttable.com)
- Sublime Text [sublimetext.com](https://sublimetext.com)

## Versioning for fun and profit

---

# Contribution to a research code



We need to **edit** files every single day.

We need to **collaborate on** files every single day.



We need to **review** files every single day.

We need to **explore** files every single day.

We need to **share** files every single day.

We need a reliable versioning system!

## Versioning for fun and profit

---

The old old way : name it, zip it, mail it

The first versioning attempts traditionally rely on 3 pillars :

- Naming conventions
- Regular backups
- Careful data management

## Versioning: the old old way

The first versioning attempts traditionally rely on 3 pillars :

- Naming conventions
- Regular backups
- Careful data management

“C’mon, I even mailed myself the latest version:  
how could this go wrong?”

## Versioning: the old old way






Awesome\_project







## Versioning: the old old way

- 📁 Awesome\_project
- 📁 Awesome\_project\_2






## Versioning: the old old way

-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup







## Versioning: the old old way

-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test








## Versioning: the old old way

-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test
-  Awesome\_project\_test









## Versioning: the old old way

-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test
-  Awesome\_project\_test
-  Awesome\_project\_backup\_2










# Versioning: the old old way

-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test
-  Awesome\_project\_test
-  Awesome\_project\_backup\_2
-  Awesome\_project\_idea\_sunday

# Versioning: the old old way










-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test
-  Awesome\_project\_test
-  Awesome\_project\_backup\_2
-  Awesome\_project\_idea\_sunday
-  Awesome\_project\_publi

# Versioning: the old old way









-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test
-  Awesome\_project\_test
-  Awesome\_project\_backup\_2
-  Awesome\_project\_idea\_sunday
-  Awesome\_project\_publi
-  Awesome\_project\_publi\_fix



# Versioning: the old old way




-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test
-  Awesome\_project\_test
-  Awesome\_project\_backup\_2
-  Awesome\_project\_idea\_sunday
-  Awesome\_project\_publi
-  Awesome\_project\_publi\_fix
-  Awesome\_project\_not\_good

# Versioning: the old old way

-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test
-  Awesome\_project\_test
-  Awesome\_project\_backup\_2
-  Awesome\_project\_idea\_sunday
-  Awesome\_project\_publi
-  Awesome\_project\_publi\_fix
-  Awesome\_project\_not\_good

...

# Versioning: the old old way

-  Awesome\_project
-  Awesome\_project\_2
-  Awesome\_project\_backup
-  Awesome\_project\_test
-  Awesome\_project\_test
-  Awesome\_project\_backup\_2
-  Awesome\_project\_idea\_sunday
-  Awesome\_project\_publi
-  Awesome\_project\_publi\_fix
-  Awesome\_project\_not\_good

...

# Versioning: the old old way

Plus, this does not behave well when scaling things up...

- How to handle team-work?
- Modification conflicts: what to do?
- No continuous time travel
- ...and a lot more issues

# Versioning for fun and profit

---

The new old way : {Drop,}Box and Changelogs

The Cloud: one more step towards real-time collaboration!

- Free & paid plans
- Auto-sync even for big files
- Team notifications
- Mobile access to files
- A carefully maintained changelog *may* keep track of changes

The Cloud: one more step towards real-time collaboration!

- Free & paid plans
- Auto-sync even for big files
- Team notifications
- Mobile access to files
- A carefully maintained changelog *may* keep track of changes

But...

The Cloud: one more step towards real-time collaboration!

- Free & paid plans
- Auto-sync even for big files
- Team notifications
- Mobile access to files
- A carefully maintained changelog *may* keep track of changes

But... does the "Cloud" really share the spirit that drives research?



*These and other features may require our systems to  
access, store and scan Your Stuff.*

— DropBox TOS

*You hereby grant Box and its contractors the right to **transmit, use and disclose** Content posted on the Service solely to the extent necessary to provide the Service, as otherwise permitted by these Terms, or to **comply with any request** of a governmental or regulatory body (including subpoenas or court orders)*

— Box TOS

*When you upload, submit, store, send or receive content to or through our Services, you give Google (and those we work with) a worldwide license to **use, host, store, reproduce, modify, create derivative works, communicate, publish, publicly perform, publicly display and distribute** such content. [...] This license continues even if you stop using our Services [...]*

— Google TOS

What about iCloud?

*Apple reserves the right at all times to determine whether Content is appropriate and in compliance with this Agreement, and may pre-screen, **move, refuse, modify and/or remove** Content at any time, without prior notice and in its sole discretion [...]*

— iCloud TOS

# Terms of Service: chosen moments

What about iCloud?

*However, by submitting or posting such Content on areas of the Service that are accessible by the public or other users with whom you consent to share such Content, you grant Apple a worldwide, royalty-free, non-exclusive license to **use, distribute, reproduce, modify, adapt, publish, translate, publicly perform and publicly display** such Content on the Service solely for the purpose for which such Content was submitted or made available, without any compensation or obligation to you.*

— iCloud TOS

# Versioning for fun and profit

---

The badass way: **git**

# Versioning: the badass way

**git** was built to handle huge projects, like the Linux kernel:

- distributed (everyone has a full copy)
- efficient (only changes are transferred)
- commit-wise versioning
- supporting branches (new ideas in a enclosed space) and tagging
- potentially server-less & self-hostable
- simultaneous modifications & smart merging
- usable through GUI (tig, gitk, tortoise-git & others) or command-line

# Versioning: the badass way

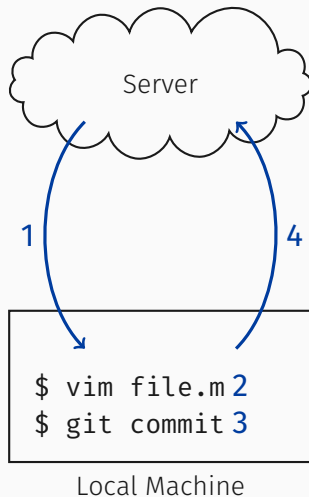
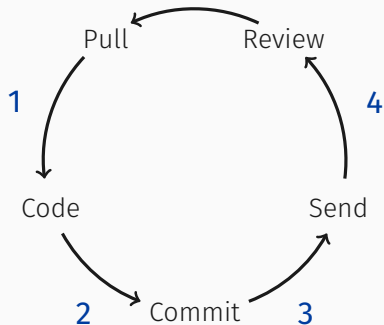
Concerning commits...

18ab98fc...	Unique (SHA1) identifier
Deletes Harkonnens	Changes Summary
Leto Atreides <leto@atreides.ar>	Committer & Author
Dec. 25 2015 21:42:21	Date & Time
- - + - - + +	Per-file additions/deletions
file doc.tex new file mode 10644 file stuff.sh old mode 100644 new mode 100755	Mode changes & file additions/deletions

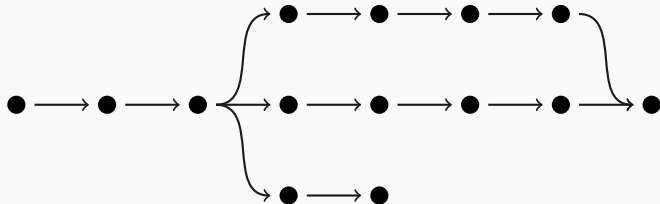
All the needed information is here!



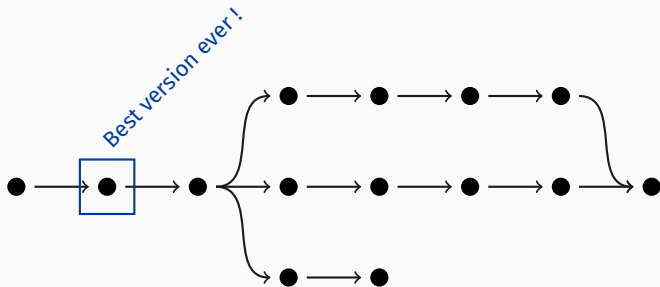
# Versioning: the badass way



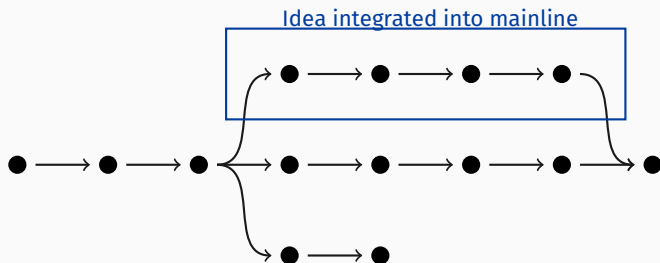
## Versioning: the badass way



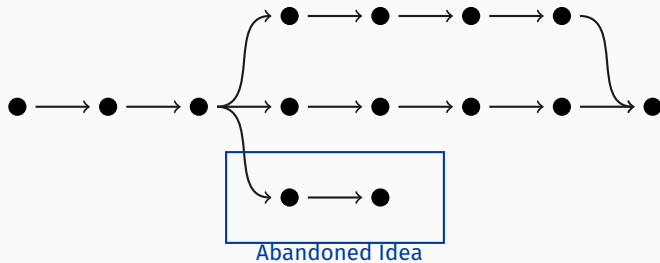
## Versioning: the badass way



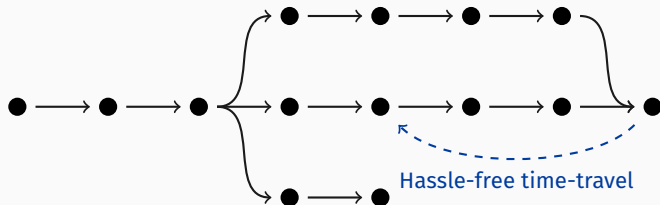
## Versioning: the badass way



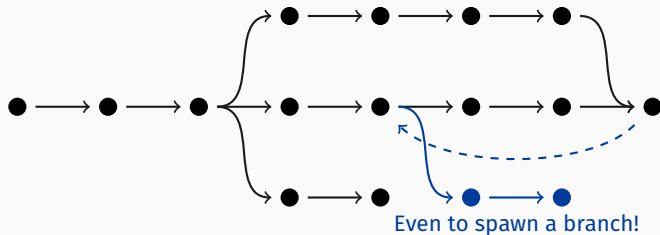
## Versioning: the badass way



## Versioning: the badass way



## Versioning: the badass way



Documentation is not a myth

---



Write The Fucking Manual

Write The ~~Fucking~~ Full Manual

## Write The ~~Fucking~~ Full Manual

Reasons ? Really ?

- Be able to reuse the code later (even after years)
- Be able to share without a full training for the receiver
- Allow reviewing & understanding the code
- Publish the code (y'know for reproducibility...)

## Documentation: right in the code

Nobody has **time** to write documentation...

but everybody can include comments in the code !

# Documentation: right in the code

Nobody has **time** to write documentation...

but everybody can include comments in the code !

## Different comments for different things

**File** What it does ? Bib. references ?

**Function** How is the task done ? How to use it ?

**Lines** Why is this operation done this way ?

# Documentation: right in the code

Nobody has **time** to write documentation...

but everybody can include comments in the code !

## Different comments for different things

**File** What it does ? Bib. references ?

**Function** How is the task done ? How to use it ?

**Lines** Why is this operation done this way ?

## Pro-tip

Start every single code file with a comment

*A computer program is just another way to express an idea.*

... and you have plenty of space to do so !

# Documentation: Variables naming matters

*A computer program is just another way to express an idea.*

... and you have plenty of space to do so !

Choose clear & meaningful names for variables and functions (avoid `tmp`, `data`, `this`, `process_data()`, `a-z` (except in loops), `id`, *etc.*)



*A computer program is just another way to express an idea.*

... and you have plenty of space to do so !

Choose clear & meaningful names for variables and functions (avoid `tmp`, `data`, `this`, `process_data()`, `a-z` (except in loops), `id`, etc.)

## Conventions

Formalized naming conventions exists (and some also address code styling issues, see PEP8 for example).

Use them as much as you can!

*When a task is repetitive... build a machine to do it.*

Documenting is boring but it's always the same !

Tools exist to automate doc generation :

**Doxygen** Doc-generator for C/C++/Java & others

**Sphinx** Doc-oriented static site generator build for the Python project (docutils) with EPUB, PDF,  $\text{\LaTeX}$  capabilities

**publish()** HTML/PDF publishing tool included in MATLAB

*etc.*

It's easy to publish it too : see [readthedocs.org](http://readthedocs.org)

What's next?

---

We couldn't fit it all in... so here is the rest:

**Github** Public Git server for free software (paid plans exist) with webpage publishing service (good to host the homepage of a project).

**Agility** Project management methods centered on human relations more than on tools and processes. See Kanban or Scrum for an introduction.

**TDD & BDD** Test- & Behavior Driven Development. Coding paradigms focusing on certain aspects (reliability & spec. compliance). Useful to scale from a research PoC to a distributed software.

Thank you!

Questions?

manzari@kth.se – gaborit@kth.se