Let there be vim

A gentle proof for its superiority

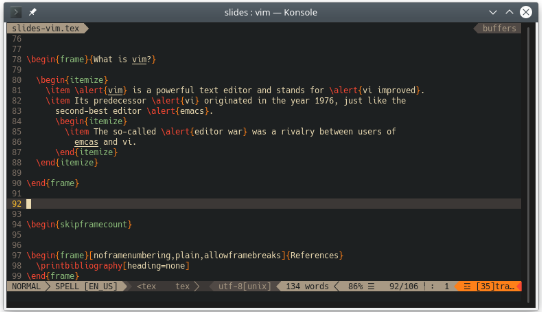
Stefan Huber

ITS, FH Salzburg

Nov 6, 2019

What is vim?

- vim is a powerful text editor and stands for vi improved.
- Its predecessor vi originated in the year 1976, just like the second-best editor emacs.
  - The so-called editor war is (was?) a rivalry between users of emcas and vi.
- Neovim is a fork with a bunch of infrastructure improvements.


slides : vim — Konsole

```
slides-vim.tex                                                                  buffers


\begin{frame}{What is vim?}

  \begin{itemize}
    \item \alert{vim} is a powerful text editor and stands for \alert{vi improved}.
    \item Its predecessor \alert{vi} originated in the year 1976, just like the
      second-best editor \alert{emacs}.
      \begin{itemize}
        \item The so-called \alert{editor war} was a rivalry between users of
          emcas and vi.
      \end{itemize}
  \end{itemize}

\end{frame}




\begin{skipframecount}


\begin{frame}[noframenumbering,plain,allowframebreaks]{References}
  \printbibliography[heading=none]
\end{frame}
NORMAL > SPELL [EN_US] > <tex   tex > utf-8[unix] < 134 words < 86% ☰ 92/106 ¦ : 1 < ☰ [35]tra…
```

It us *just* and editor, right?

Yes.

It us *just* and editor, right?

Yes.

- But ask Jamie Oliver about his knives or Rafael Nadal about his tennis racket. *Just* knives, *just* rackets.

It us *just* and editor, right?

Yes.

- But ask Jamie Oliver about his knives or Rafael Nadal about his tennis racket. *Just* knives, *just* rackets.

Developers, administrators, and other computer users edit text files all day long:

- Programming in all kind of languages, including web (CSS, SASS, . . .)
- Linux configuration files, patch files, git commit messages
- LaTeX, markdown, other text
- Mail
- Todo manager
- Mass renaming of files (moreutils), man page viewer

This is, why I not only care about my editor but also about my keyboard.

What makes vim different?

I believe the power of vim originates from these sources:

- Purely keyboard-driven with a rich logic behind its shortcuts. (Details later.)
  - Concept of modes
  - A verb-subject shortcut logic and the ability to combine operations.
    Ex: `3gq}` reformats the next three paragraphs and `yyp` swaps two lines.

  ⋮

- A powerful configuration system.
- A scripting language `vimscript` – but also `Python` – that generated an addon ecosystem.

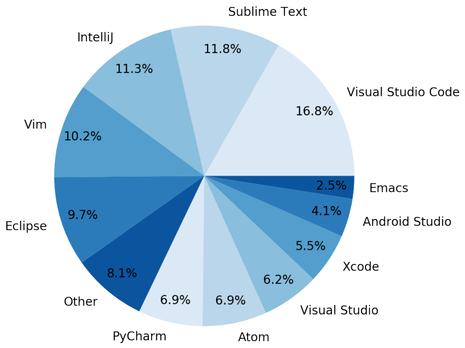
The last two points can transform vim into something IDE-like, if requested.

Evidence for its superiority

From [Triplebyte2018]:

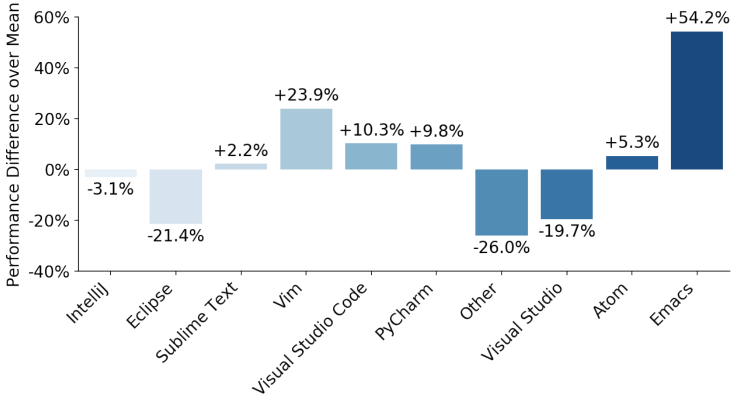
> *Triplebyte interviews hundreds of engineers every week. For each interview, we record the editor, language and operating system used. We don't use this information to decide who passes our interview (I don't think that would be fair). However, it is fascinating data!*

Editor usage



- Data conducted over 2018.
- People do use vim, but Emacs not so much.
- VS Code is a rising star. The year before (2017) it was among “others”.

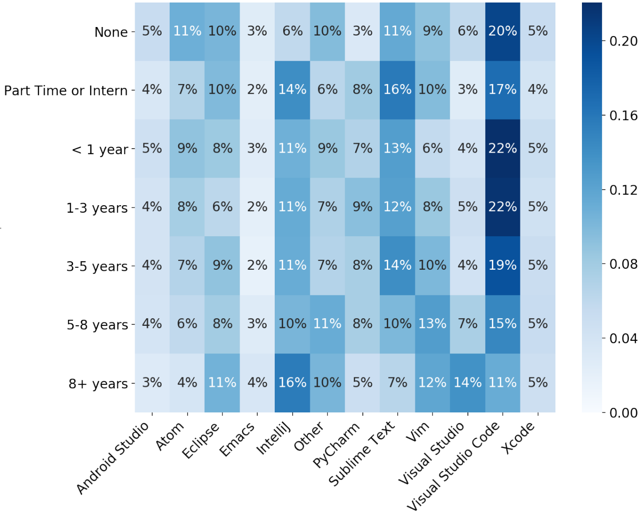
Interview pass rates by editor



- Emacs and vim users perform very well.
  - Age? Ambition and frustration tolerance? Correlation with language, OS, et cetera?
- Eclipse and Visual Studio users don't.
  - IDEs are bad? But what about PyCharm? Is it about language?

Editor by experience

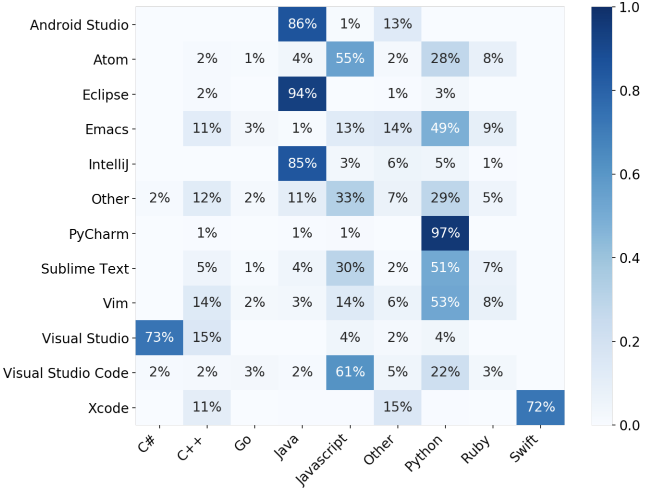
- Vim users among all experience levels (expressed in years, not ability).
- Juniors like VS Code.



	Android Studio	Atom	Eclipse	Emacs	IntelliJ	Other	PyCharm	Sublime Text	Vim	Visual Studio	Visual Studio Code	Xcode
None	5%	11%	10%	3%	6%	10%	3%	11%	9%	6%	20%	5%
Part Time or Intern	4%	7%	10%	2%	14%	6%	8%	16%	10%	3%	17%	4%
< 1 year	5%	9%	8%	3%	11%	9%	7%	13%	6%	4%	22%	5%
1-3 years	4%	8%	6%	2%	11%	7%	9%	12%	8%	5%	22%	5%
3-5 years	4%	7%	9%	2%	11%	7%	8%	14%	10%	4%	19%	5%
5-8 years	4%	6%	8%	3%	10%	11%	8%	10%	13%	7%	15%	5%
8+ years	3%	4%	11%	4%	16%	10%	5%	7%	12%	14%	11%	5%

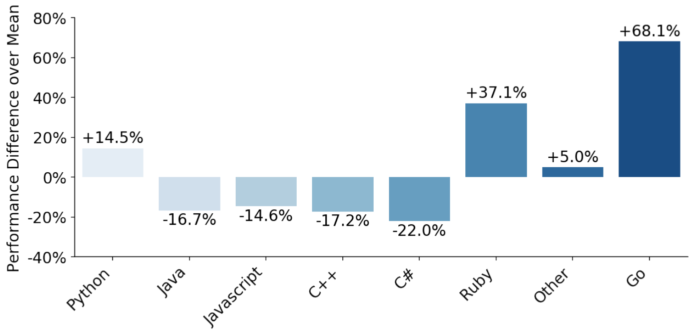
Editor by language

- Vim users code mostly in C++, Javascript and Python.
- Users of Eclipse and Visual Studio perform bad and mostly do C# and Java.



	C#	C++	Go	Java	Javascript	Other	Python	Ruby	Swift
Android Studio				86%	1%	13%			
Atom		2%	1%	4%	55%	2%	28%	8%	
Eclipse		2%		94%		1%	3%		
Emacs		11%	3%	1%	13%	14%	49%	9%	
IntelliJ				85%	3%	6%	5%	1%	
Other	2%	12%	2%	11%	33%	7%	29%	5%	
PyCharm		1%		1%	1%		97%		
Sublime Text		5%	1%	4%	30%	2%	51%	7%	
Vim		14%	2%	3%	14%	6%	53%	8%	
Visual Studio	73%	15%			4%	2%	4%		
Visual Studio Code	2%	2%	3%	2%	61%	5%	22%	3%	
Xcode		11%				15%			72%

Interview pass rates by language



A two-edged sword

Recommending vim is a two-edged sword:

- It is a powerful tool once mastered.
- However, unlike most editors, vim (and emacs alike) come with a learning curve.

A two-edged sword

Recommending vim is a two-edged sword:

- It is a powerful tool once mastered.
- However, unlike most editors, vim (and emacs alike) come with a learning curve.

- Within 4 years and 9 months, 1000082 people viewed the article *How to exit the Vim editor?*
- Fun fact: Most of them are jquery, css and angularjs developers.
- *Stack Overflow: Helping One Million Developers Exit Vim*. May 2017. URL: https://stackoverflow.blog/2017/05/23/stack-overflow-helping-one-million-developers-exit-vim/ (visited on Oct. 29, 2019)

First vim steps

Vi is organized in different modes:

Normal	Running commands, e.g., navigation, searching, replacing, operating
Insert	Inserting (replacing) text
Visual	Selecting text

`<esc>` brings you in normal mode, `i` in insert mode and `v` in visual mode.

Normal mode

Most power is in the normal mode. Virtually every keystroke is a command.

- Motion commands: `j`, `k`, `h`, `l` for down/up, left/right. (Or cursor keys.)
- More motion: `b`, `w` for word-wise backward, forward and `{`, `}` for paragraph-wise.
- Delete, yank (copy), cut and paste is `d`, `y`, `x`, `p`.
- These can combined in a verb-subject logic:
  - `y}` means yank text until end of paragraph and `yy` means yank this line.
  - `3p` means paste three times, e.g., `yy20p` duplicates the line 20 times.
  - `dd` means delete current line and `ddp` therefore moves a line down (delete and paste below).

The zen of vim

A few thoughts on “groking vim”:
`https://stackoverflow.com/questions/1218390/what-is-your-most-productive-shortcut-with-vim/1220118#1220118`

How I personally use vim

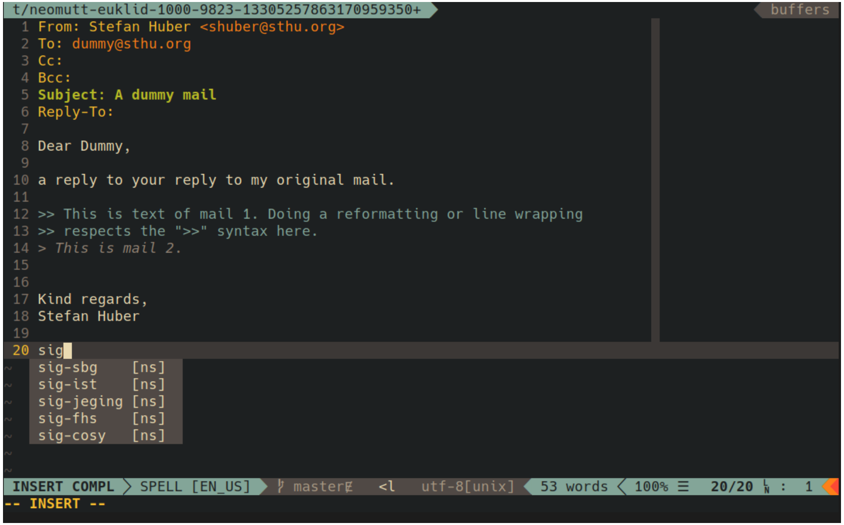
A modular vim configuration, Aug 2018:

- `https://www.sthu.org/code/codesnippets/vimconf.html`

Now some action. . .

- Disclaimer: Many features shown have been adopted by other editors in the last decades.

Editing mails in vim



```
t/neomutt-euklid-1000-9823-13305257863170959350+                                  buffers
From: Stefan Huber <shuber@sthu.org>
To: dummy@sthu.org
Cc:
Bcc:
Subject: A dummy mail
Reply-To:

Dear Dummy,

a reply to your reply to my original mail.

>> This is text of mail 1. Doing a reformatting or line wrapping
>> respects the ">>" syntax here.
> This is mail 2.


Kind regards,
Stefan Huber

sig
sig-sbg     [ns]
sig-ist     [ns]
sig-jeging  [ns]
sig-fhs     [ns]
sig-cosy    [ns]


INSERT COMPL  SPELL [EN_US]  master  <l  utf-8[unix]  53 words  100% ☰  20/20  :  1
-- INSERT --
```

Editing mails in vim

- Vim knows `filetype` dependent configuration.
  - The filetype `mail` sets `textwidth` to 78.
  - Spell checking is enabled. Language is auto-detected.
  - According syntax highlighting is used.
  - If I mail patch files the command `:set ft=c` quickly changes to C source filetype.
- The command `gq` does reformatting to textwidth.
  - I frequently call `gq}` to reformat a text paragraph.
- Snippet completion known from IDEs completes common patterns:
  - `lg` completes `Liebe Grüße, Stefan`.
  - `sig-sbg` completes to my signature for my private address.
- vim is text based: Can run it remotely in my mail server.

LaTeX in vim

- I collaborate in paper editing using git.
  - And addon shows me +/∼/- marks to indicate `git diff`.
  - The command `:GitBlame` tells me which line was last modified by whom.
- The command `\ll` starts `latexmk` in background in continuous mode and fires up a PDF viewer.
  - Whenever I save a tex file or other dependencies the viewer updates automatically.
- Syntax checkers run all kind of lint-like software, also for latex.
  - They also hint to possibly wrong typesetting.
- Snippet completion accelerates
  - creating latex beamer slides
  - inserting common environments like itemize, figure, table, . . .

Programming in vim

Programming is a lot about navigation. Marks help:

- `mA` sets the mark `A` to the current position.
- `'C` jumps to mark `C`.
- File-local marks are lower case.
- `x'b` cuts everything until mark `b`.
- `:help marks` fires up the built-in documentation to marks.

Navigation is a lot about using monitor real-estate. Splitting helps:

- `:split` splits the window horizontally, `:vsplit` does vertically.
- `\wk` moves focus to the window below, `\w=` equalizes the window sizes.
- `F11` does a vertical split and opens `.h` file to a `.c*` file and vice versa. It is smart enough to find the include directory.

Programming in vim

Bad whitespace is bad:

- Mixing tabs and spaces. Trailing whitespace.
- I let vim highlight them.
- The command `:set list` shows whitespace explicitly.

Respecting whitespace and syntax:

- Vim is syntax-aware when wrapping lines or reformatting in multi-line comments.
- The command `=` performs auto-formatting, e.g., fixing indentation.
- An addon detects the indentation format, e.g., 4 spaces or one tab.
    - `:retab` changes tabs to spaces or vice versa accordingly.

Programming in vim

Programming is typically line-wise editing:

- Many commands operate on lines, like `dd`, `yy` and such.
- `A` jumps to end of line and switches to insert mode.
- `o` adds an empty line below with right indentation and switches to insert mode.
- There is a line-based visual mode to select whole lines.

Sometimes we wish to edit columns-wise:

- Layouting a table, ASCII art, whatever.
- Vim also knows a column-based visual mode.
- This allows for simultaneous edits of multiple lines at once.

Config files

When editing config files I often need to do mass operations:

- `:g/^#/d` removes (`d`) all lines (`:g`) that match the pattern “start with hash” (`^#`).
- `:%s/\s\+$//g` removes all whitespace. That is, it replaces on all lines (`:%s`) whitespace at the end of a line (`\s\+$`) with nothing.

Vimwiki

Vimwiki is a vim-integrated wiki system.

- For self-organization.
- I use it to keep TODOs and let them being displayed on desktop wallpaper, synced via nextcloud to my mobile devices.

How to start or improve

I do not explicitly recommend vim.

- But for me it is *by far* the best editor.

If you are still interested:

- Start with the interactive learning program `vimtutor` to gather the basics in like 30 minutes.
- Then learn new tricks over the next years. Different people use different features for their different habits and use cases.

Configuration

Much of the power of vim comes from its configuration:

- Configurations are very personal.
- If you start with vim then think about starting with a minimal configuration and let it grow by your personal needs over the next years.

The antithesis to the above said is `https://spacevim.org/`

- Maybe have a look there, too.

A more scientific talk

Persistent homology in data science and more.

- The theory behind `https://www.sthu.org/code/libstick/`

References I

[so-exitvim-2017] *Stack Overflow: Helping One Million Developers Exit Vim*. May 2017. URL: https://stackoverflow.blog/2017/05/23/stack-overflow-helping-one-million-developers-exit-vim/ (visited on Oct. 29, 2019).

[Triplebyte2018] *The Rise of Microsoft Visual Studio Code*. Dec. 2018. URL: https://triplebyte.com/blog/editor-report-the-rise-of-visual-studio-code (visited on Oct. 29, 2019).