# Faster TeX with Vim

Umut Özer and Afiq Hatta

# Faster TeX with Vim

Umut Özer and Afiq Hatta

# Outline

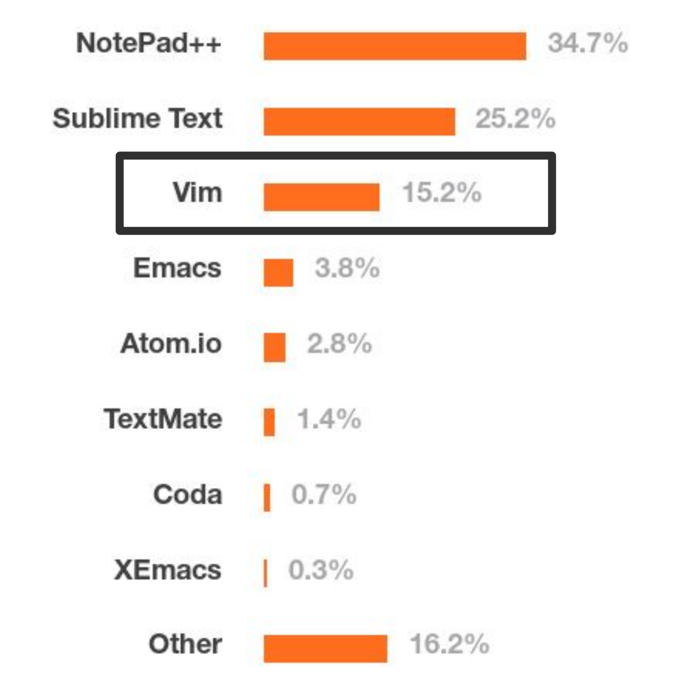
1. What’s Vim and how do I get it? (10 minutes - Umut)
    a. Installing Vim (Linux)
    b. Using MacVim for Mac - can demo
2. Useful plugins for writing LaTeX (10 minutes - Afiq)
    a. Installing plugins with VimPlug and the .vimrc
    b. VimTex and hotkeys with VimTex
3. The most important Plugin: UltiSnips and how to use it (10 minutes - Afiq and Umut)
    a. Demonstration
    b. Writing snippets - BIG DEMO
4. Getting code from Gilles Castel’s github (5 mins)
5. Boni! (Github linking if time)

# Vim in the olden days

- Based on vi (1976); every UNIX system has it
- Vim: Vi IMproved (1991) is a powerful all-purpose editor
    - Linux and Mac: already installed
        - Can also download MacVim (‘mvim’)
    - Windows: download at https://www.vim.org/download.php#pc

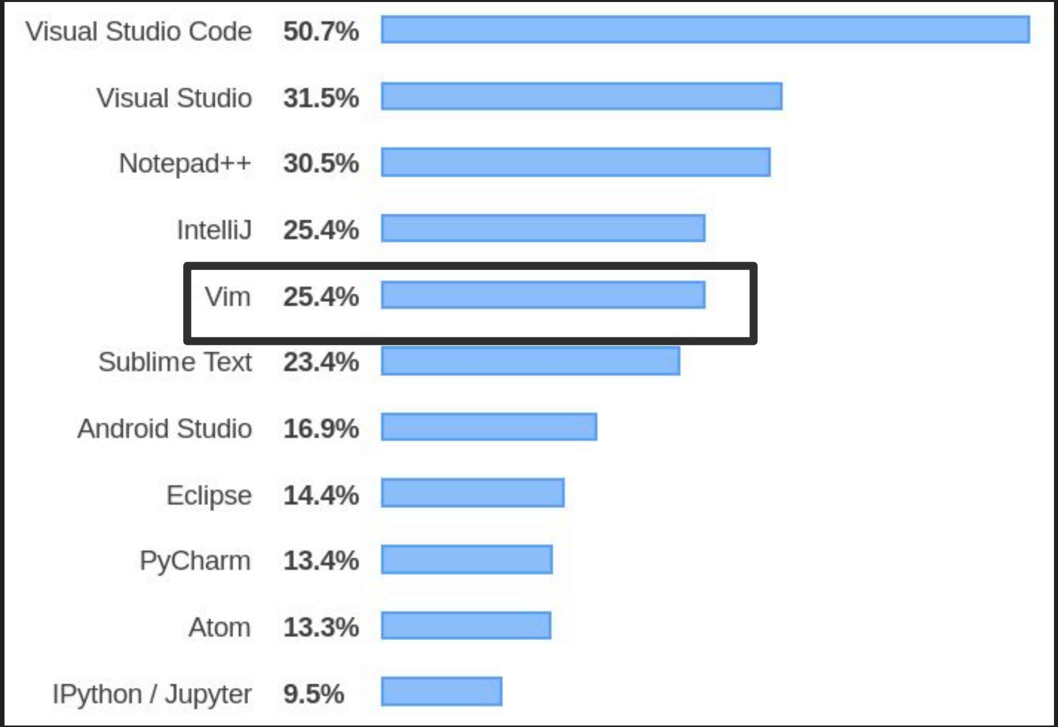
# Vim Today

## Most popular text editors (2015)



insights.stackoverflow.com/survey/2015

## Most popular IDEs (2019)



insights.stackoverflow.com/survey/2019

# What’s the big idea? Use the Keyboard!

- You never need to lift hands off the keyboard again!
- No disruption to the flow by having to use a mouse or arrow keys
- Fingers stay close to “home row” (asd…hjkl)

A
S
D
F
J
K
L
;
tab
caps lock
shift
Ctrl
delete
return
shift
Q
W
E
R
T
Y
U
I
O
P
G
H
V
B
N

# Additional Resources: First Steps with Vim

- Built-in tutorial: type “vimtutor” in command line
    - Unfortunately not available for Windows users
- Built-in help system: type “:help” from inside vim
- Use gVim: graphical user interface like an ordinary editor, but has the full power of Vim, too!

# DEMO BY UMUT

# Vim has four modes*

- Switch in and out of different modes to move around, make edits, record macros, use commands, …

| Mode | Use |
|---|---|
| Normal Mode | The default mode: Movement, general editing, macros, ... |
| Insert Mode (i) | Insert text like in any other text editor |
| Command Mode (:) | Useful commands like search / replace |
| Visual Mode (v) | Perform actions on big blocks of text |

* that we care about...

# Normal Mode: Basic Movement

- The cursor is moved with the hjkl keys of the right hand

h (left)  j (down)  k (up)  l (right)

~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 ( 9 ) 0 _ - + = delete
tab Q W E R T Y U I O P { [ } ] | \
caps lock A S D F G H J K L ; " ' return
shift V B N shift
Ctrl → ↓ ↑

# Normal Mode: Editing

- Perform general edits
  - Move things around: delete, copy, paste
  - Record macros on the fly for repetitive tasks

| Keys | Mnemonic | Action |
|---|---|---|
| x | ‘Cross out’ | Delete the character under the cursor |
| i | “insert” | Enter “insert” mode to insert text (press ESC to get back out to normal mode) |
| A (upper case) | “Append” | “Append” text at the end of the line |
| u | “undo” | Undo the last command, press ‘U’ to fix a whole line |

# Normal Mode: Motions

| Key | Motion |
|---|---|
| w | Until start of next “word” |
| e | To the “end” of current word |
| $ | To end of current line |

- Typing a number before a motion repeats it that many times
  - E.g.: type ‘2w’ to move cursor two words forward
  - Type ‘0’ to move to beginning of line

# Normal Mode: Operators and Motions

- General Vim principle: Combine operators and motions!

| Keys | Mnemonic | |
|---|---|---|
| dw | ‘Delete word’ | |
| ce | “Change to end of word” | |
| yas | “Yank a sentence” | |

# Normal Mode: Jumping Motions

| Keys | Movement |
|---|---|
| gg or G | Beginning and end of file respectively |
| 506G | Jump to line 506 |
| / PHRASE | Search for PHRASE, press ENTER to jump to it |
| n | “next” match of PHRASE, ‘N’ finds previous match. |
| CTRL-O | “out” one level; back to where we jumped from. ‘CRTL-I’ goes back “in”. |
| % | Positioned on ‘(‘, ‘[‘ or ‘{‘, jump to matching ‘)’, ‘]’ or ‘}’. |
| ) or } or ]] | Go a sentence ) , paragraph }, or section ]] forward. Use ( { and [[ to go backwards. |

# Normal Mode: Mnemonic Editing

- Use key combinations to perform more complicated actions

| Examples: | Mnemonic | Action |
|---|---|---|
| ci{ | “Change in braces” | Changes text inside {...} |
| da( | “Delete around parens” | Deletes all of ‘(...)’ |
| yas | “Yank a sentence” | Yank is synonym to “copy” in vim.<br>(c already taken for “change”) |

- Guess: which character is used to “paste” the yanked sentence?
- The same key will “put” the “deleted” ‘(...)’ back after the cursor.

# Command Mode

| Key | Mnemonic | Function |
|---|---|---|
| :w | ‘write’ | Save changes to the current file |
| :q | ‘quit’ | Exit vim (use :q! to force quit, discarding changes) |
| :s/X/Y/ | ‘search’ | ‘Search’ for ‘X’ and replace with ‘Y’ |
| :h | ‘help’ | Built-in help facility for commands and features |

- Use ‘:w FILENAME’ to save to a new file
- Type ‘:h ~’ to find out what ‘~’ does
- Completion with <TAB> and CTRL-D

# More stuff: Reference Slide (peruse :h in own time)

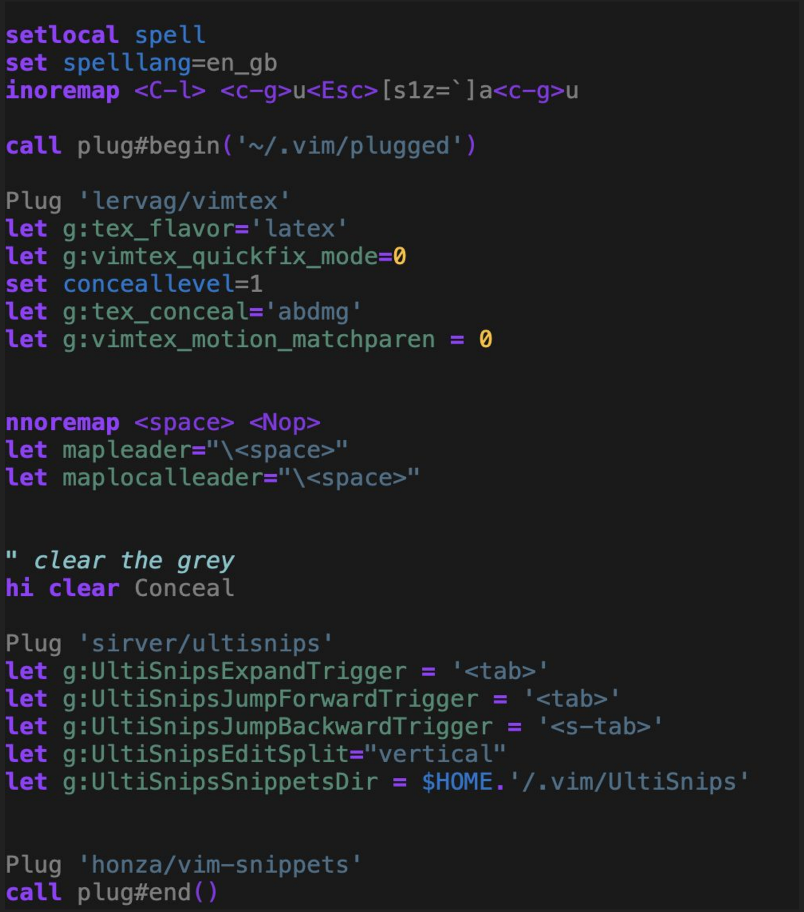
- **Spell checking** with ‘:set spell spelllang=en_gb’ (good to put in .vimrc)
    - Press ‘z=’ to fix spelling mistake under cursor
    - Press ‘zg’ (“good”) to add a word to the dictionary
    - Quickly fix last typo with CTRL-L in insert mode (Castell snippet)
- **Folding** with ‘zf’ (“fold”), and ‘zo’ to “open” a fold
- Line numbers, colorscheme, and other **customisation** in .vimrc
- Create **new mappings** using :map
- Use ‘.’ to repeat previous action. Can be combined with a [count].
- Start **recording a macro** with ‘qx’, end with ‘q’. Execute with ‘@x’.
- “Mark” a location with ‘mx’, then **jump to marked spot** with `x and ’x
    - Can use any letter in place of ‘x’

# Even more stuff: Reference Slide

- **Auto-complete** with CTRL-P (plugin)
- Yank to the x register with “xy and then paste with “xp
    - Your **clipboard** (usual copy/paste) is the “**+** register (requires gVim)
- **Visual mode** (entered with ‘v’) and its variants for **bulk-editing**!

# The ~/.vimrc

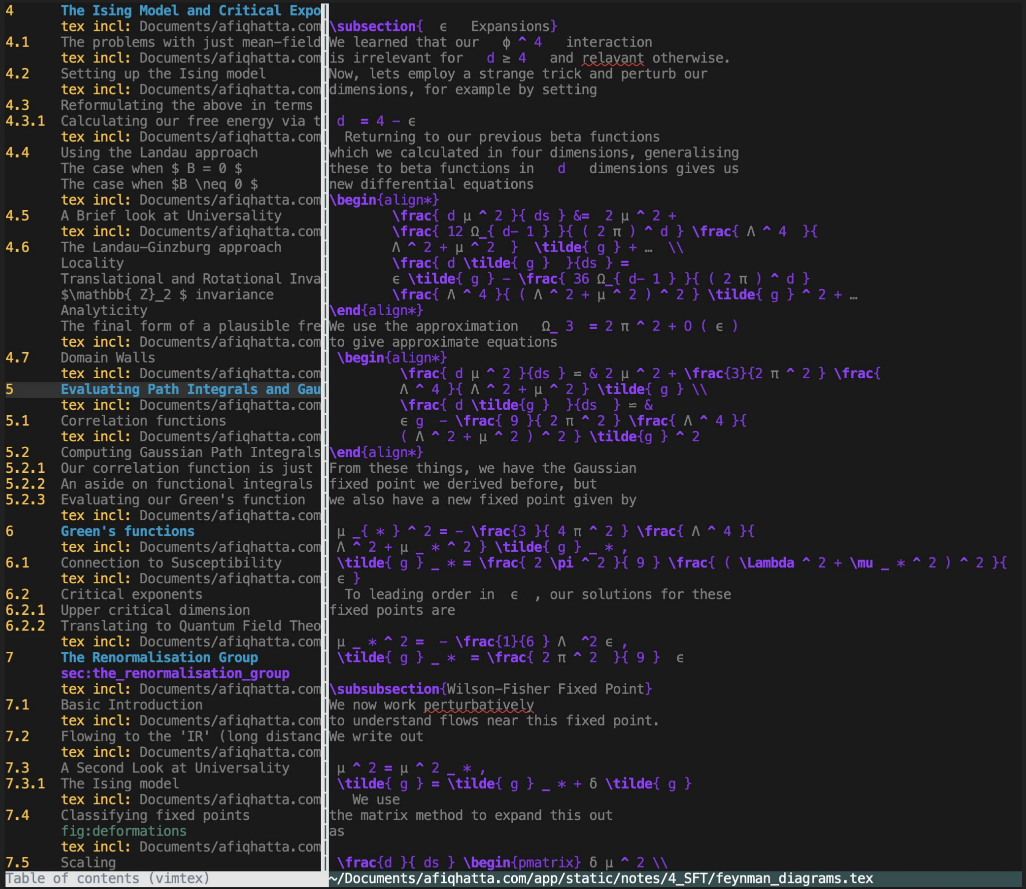
- Startup script: configure features
- Install vim-plug at https://github.com/junegunn/vim-plug
- This will be on my github (https://github.com/afiqhatta)
- Activate the configuration with :source ~/.vimrc



```vim
setlocal spell
set spelllang=en_gb
inoremap <C-l> <c-g>u<Esc>[s1z=`]a<c-g>u

call plug#begin('~/.vim/plugged')

Plug 'lervag/vimtex'
let g:tex_flavor='latex'
let g:vimtex_quickfix_mode=0
set conceallevel=1
let g:tex_conceal='abdmg'
let g:vimtex_motion_matchparen = 0


nnoremap <space> <Nop>
let mapleader="\<space>"
let maplocalleader="\<space>"


" clear the grey
hi clear Conceal

Plug 'sirver/ultisnips'
let g:UltiSnipsExpandTrigger = '<tab>'
let g:UltiSnipsJumpForwardTrigger = '<tab>'
let g:UltiSnipsJumpBackwardTrigger = '<s-tab>'
let g:UltiSnipsEditSplit="vertical"
let g:UltiSnipsSnippetsDir = $HOME.'/.vim/UltiSnips'


Plug 'honza/vim-snippets'
call plug#end()
```

# VimTex

- The best way to handle TeX files in vim
- Includes syntax highlighting, content selection and more
- (Umut to explain auto-compile and local leader hotkeys)



```
4       The Ising Model and Critical Expo
        tex incl: Documents/afiqhatta.com
4.1     The problems with just mean-field
        tex incl: Documents/afiqhatta.com
4.2     Setting up the Ising model
        tex incl: Documents/afiqhatta.com
4.3     Reformulating the above in terms
4.3.1   Calculating our free energy via t
        tex incl: Documents/afiqhatta.com
4.4     Using the Landau approach
        The case when $ B = 0 $
        The case when $B \neq 0 $
        tex incl: Documents/afiqhatta.com
4.5     A Brief look at Universality
        tex incl: Documents/afiqhatta.com
4.6     The Landau–Ginzburg approach
        Locality
        Translational and Rotational Inva
        $\mathbb{ Z}_2 $ invariance
        Analyticity
        The final form of a plausible fre
        tex incl: Documents/afiqhatta.com
4.7     Domain Walls
        tex incl: Documents/afiqhatta.com
5       Evaluating Path Integrals and Gau
        tex incl: Documents/afiqhatta.com
5.1     Correlation functions
        tex incl: Documents/afiqhatta.com
5.2     Computing Gaussian Path Integrals
5.2.1   Our correlation function is just
5.2.2   An aside on functional integrals
5.2.3   Evaluating our Green's function
        tex incl: Documents/afiqhatta.com
6       Green's functions
        tex incl: Documents/afiqhatta.com
6.1     Connection to Susceptibility
        tex incl: Documents/afiqhatta.com
6.2     Critical exponents
6.2.1   Upper critical dimension
6.2.2   Translating to Quantum Field Theo
        tex incl: Documents/afiqhatta.com
7       The Renormalisation Group
        sec:the_renormalisation_group
        tex incl: Documents/afiqhatta.com
7.1     Basic Introduction
        tex incl: Documents/afiqhatta.com
7.2     Flowing to the 'IR' (long distanc
        tex incl: Documents/afiqhatta.com
7.3     A Second Look at Universality
7.3.1   The Ising model
        tex incl: Documents/afiqhatta.com
7.4     Classifying fixed points
        fig:deformations
        tex incl: Documents/afiqhatta.com
7.5     Scaling
Table of contents (vimtex)
```

```

\subsection{  ϵ   Expansions}
We learned that our   ϕ ^ 4   interaction
is irrelevant for   d ≥ 4   and relavant otherwise.
Now, lets employ a strange trick and perturb our
dimensions, for example by setting

 d  = 4 − ϵ
  Returning to our previous beta functions
which we calculated in four dimensions, generalising
these to beta functions in   d   dimensions gives us
new differential equations
\begin{align*}
        \frac{ d μ ^ 2 }{ ds } &=  2 μ ^ 2 +
        \frac{ 12 Ω_{ d− 1 } }{ ( 2 π ) ^ d } \frac{ Λ ^ 4  }{
        Λ ^ 2 + μ ^ 2  }  \tilde{ g } + …  \\
        \frac{ d \tilde{ g }  }{ds } =
        ϵ \tilde{ g } − \frac{ 36 Ω_{ d− 1 } }{ ( 2 π ) ^ d }
        \frac{ Λ ^ 4 }{ ( Λ ^ 2 + μ ^ 2 ) ^ 2 } \tilde{ g } ^ 2 + …
\end{align*}
We use the approximation   Ω_ 3  = 2 π ^ 2 + 0 ( ϵ )
to give approximate equations
 \begin{align*}
         \frac{ d μ ^ 2 }{ds } ≕ & 2 μ ^ 2 + \frac{3}{2 π ^ 2 } \frac{
         Λ ^ 4 }{ Λ ^ 2 + μ ^ 2 } \tilde{ g } \\
         \frac{ d \tilde{g }  }{ds  } ≕ &
         ϵ g  − \frac{ 9 }{ 2 π ^ 2 } \frac{ Λ ^ 4 }{
         ( Λ ^ 2 + μ ^ 2 ) ^ 2 } \tilde{g } ^ 2
\end{align*}
From these things, we have the Gaussian
fixed point we derived before, but
we also have a new fixed point given by

 μ _{ ∗ } ^ 2 = − \frac{3 }{ 4 π ^ 2 } \frac{ Λ ^ 4 }{
 Λ ^ 2 + μ _ ∗ ^ 2 } \tilde{ g } _ ∗ ,
 \tilde{ g } _ ∗ = \frac{ 2 \pi ^ 2 }{ 9 } \frac{ ( \Lambda ^ 2 + \mu _ ∗ ^ 2 ) ^ 2 }{
 ϵ }
  To leading order in  ϵ  , our solutions for these
fixed points are

 μ _ ∗ ^ 2 =  − \frac{1}{6 } Λ  ^2 ϵ ,
 \tilde{ g } _ ∗  = \frac{ 2 π ^ 2  }{ 9 }  ϵ

\subsubsection{Wilson–Fisher Fixed Point}
We now work perturbatively
to understand flows near this fixed point.
We write out

 μ ^ 2 = μ ^ 2 _ ∗ ,
 \tilde{ g } = \tilde{ g } _ ∗ + δ \tilde{ g }
   We use
the matrix method to expand this out
as

 \frac{d }{ ds } \begin{pmatrix} δ μ ^ 2 \\
~/Documents/afiqhatta.com/app/static/notes/4_SFT/feynman_diagrams.tex
```

# UltiSnips - a great hotkey tool for vim

- This is where most of our speed gains come from!
- We store snippets in a file which we specify in our ~/.vimrc
- Has a rich syntax in which to describe when hotkeys are activated
- Edit snippets with :UltiSnipsEdit

DEMOS by Afiq and Umut

# Vimtex additions to: Normal mode

- Gives “surrounding” classification for actions
- Recognises TeX environments

| Examples: | Mnemonic | Action |
|---|---|---|
| cse | “Change surrounding environment” | Changes stuff in \begin{...} \end{...}<br>E.g. “equation” -> “align” |
| tsd | “Toggle surrounding delimiter” | (...) <-> \left( … \right) |
| di$ | “Delete in dollars” | $...$ -> $$ |