Setting up R, vimtex and Ultisnips in vim on a Mac

Ronald (Ryy) Glenn Thomas

2025-05-13

Table of contents

1 Introduction

Start by installing vim (neovim), R tex vimtex ultisnips

See post "Setting up a minimal neovim…" for details on installing plugins with Neovim.

2 Sections

1. vim sections
2. ALE, lSP, completion, lintr, fix

- R
- julia
- python

2. ftplugins
3. ultisnips
    - dynamic
4. vimtex

- completion

3 Install and configure plugins (source file ~/.vimrc)

```
set runtimepath^=~/vimplugins
syntax enable
filetype plugin indent on
let mapleader = ","
let maplocalleader = " "
" source ~/.vimplugins
call plug#begin('~/vimplugins')
Plug 'rakr/vim-one'
Plug 'mhartington/oceanic-next'
Plug 'jasonccox/vim-wayland-clipboard'
Plug 'junegunn/vim-easy-align'
Plug 'rafi/awesome-vim-colorschemes'
Plug  'mbbill/undotree'
Plug 'jpalardy/vim-slime'
" let g:slime_target = "vimterminal"
" let g:slime_no_mappings = 1
" let g:slime_vimterminal_config = { "vertical": 1 }
" let g:slime_vimterminal_cmd = "R"
" nmap <C-c>v <Plug>SlimeConfig
" nmap <localleader>l <Plug>SlimeCellsSendAndGoToNext
" nmap <localleader>j <Plug>SlimeCellsNext
" nmap <localleader>k <Plug>SlimeCellsPrev
Plug 'lervag/vimtex'
Plug 'tpope/vim-vividchalk'
let g:vimtex_complete_close_braces=1
let g:vimtex_quickfix_mode=0
Plug 'sirver/ultisnips'
```

```
let g:UltiSnipsExpandTrigger="<C-tab>"
let g:UltiSnipsJumpForwardTrigger="<tab>"
let g:UltiSnipsJumpBackwardTrigger="<S-tab>"
nnoremap <leader>U <Cmd>call UltiSnips#RefreshSnippets()<CR>
nnoremap <leader>u :UltiSnipsEdit<cr>
Plug 'dense-analysis/ale'
" let g:ale_completion_enabled = 1
let g:ale_virtualtext_cursor = 'disabled'
let g:ale_set_balloons = 1
highlight clear ALEErrorSign
highlight clear ALEWarningSign
let g:ale_sign_error = ' '
let g:ale_sign_warning = '.'
let g:ale_linters = {'python': ['pylsp']}
" Set this variable to 1 to fix files when you save them.
let g:ale_fix_on_save = 1
let g:ale_fixers = {
\   '*': ['remove_trailing_lines', 'trim_whitespace'],
\   'r': ['styler'],
\   'rmd': ['styler'],
\   'quarto': ['styler'],
\   'python': ['black','isort'],
\   'javascript': ['eslint'],
\}
nnoremap <leader>n :ALENext<CR>
Plug 'junegunn/fzf', { 'do': { -> fzf#install() } }
Plug 'junegunn/fzf.vim'
nnoremap <leader>z :Files<CR>
nnoremap <Leader>' :Marks<CR>
nnoremap <Leader>/ :BLines<CR>
nnoremap <Leader>b :Buffers<CR>
nnoremap <Leader>r :Rg<CR>
nnoremap <Leader>s :Snippets<CR>
tnoremap <leader>b <C-w>:Buffers<cr>
tnoremap <leader>r <C-w>:Rg<cr>
tnoremap <leader>z <C-w>:Files<cr>
Plug 'junegunn/vim-peekaboo'
Plug 'tpope/vim-unimpaired'
Plug 'tpope/vim-obsession'
Plug 'tpope/vim-repeat'
Plug 'tpope/vim-commentary'
autocmd FileType quarto setlocal commentstring=#\ %s
Plug 'tpope/vim-surround'
Plug 'justinmk/vim-sneak'
let g:sneak#label = 1
let g:sneak#s_next = 1
let g:sneak#use_ic_scs = 1
```

```
Plug 'machakann/vim-highlightedyank'
Plug 'vim-airline/vim-airline'
let g:airline#extensions#ale#enabled = 1
let g:airline#extensions#tabline#enabled = 1
let g:airline#extensions#fzf#enabled = 1
Plug 'rgt47/rgt-R'
Plug 'girishji/vimcomplete'
call plug#end()
 if $COLORTERM == 'truecolor'
   set termguicolors
 endif
 set scrolloff=7
set iskeyword-=. "add '.' to break word list
set completeopt=menu,menuone,popup,noinsert,noselect
set complete+=k
set dictionary=/usr/share/dict/words
highlight Pmenu  guifg=Black guibg=cyan gui=bold
highlight PmenuSel  gui=bold guifg=White guibg=blue
set gfn=Monaco:h14
set encoding=utf-8
set lazyredraw
set autochdir
set number relativenumber
set clipboard=unnamed
set textwidth=80
set colorcolumn=80
set cursorline
set hlsearch
set incsearch
set ignorecase
set smartcase
set showmatch
set noswapfile
set hidden
set gdefault
set splitright
set wildmenu
set wildignorecase
set wildmode=list:full
" experiment to map TAB to navigate pop ups
inoremap <expr> <tab> pumvisible() ? "\<C-n>" : "\<tab>"
inoremap <expr> <S-tab> pumvisible() ? "\<C-p>" : "\<S-tab>"

nnoremap <leader>o <C-w>:b1<CR>
nnoremap <leader>t <C-w>:b2<CR>
nnoremap <leader>h <C-w>:b3<CR>
nnoremap <leader><leader> <C-w>w
```

```
nnoremap <leader>a ggVG
nnoremap <leader>m vipgq
nnoremap <leader>f :tab split<cr>
nnoremap <leader>v :edit ~/.vimrc<cr>
nnoremap <localleader><leader> <C-u>
nnoremap <localleader><localleader> <C-d>
noremap - $
noremap : ;
noremap ; :
inoremap <F10> <C-x><C-k>
inoremap <F12> <C-x><C-o>
inoremap <expr> <TAB> pumvisible() ? "\<C-n>" : "\<TAB>"
inoremap <silent> <Esc> <Esc>``

tnoremap <F1> <C-\><C-n>
tnoremap <leader>o <C-w>:b1<CR>
tnoremap <leader>t <C-w>:b2<CR>
tnoremap <leader>h <C-w>:b3<CR>
tnoremap <leader><leader> <C-w>w
tnoremap lf ls()<CR>
" nnoremap <silent> <C-CR> :let @c = getline(".") . "\n" \|
"           \ :call term_sendkeys(term_list()[0], @c)<CR><CR>
" vnoremap  <silent> <C-v> y \| :let @c=@" . "\n" <CR> \|
"   \ :call term_sendkeys(term_list()[0], @c)<CR>
" vmap <silent> <C-CR> <C-v>}
" " vnoremap <C-CR> :w temp.R<CR> \| :let @x = "source('temp.R')" \|
" "         \ :call term_sendkeys(term_list()[0], @x)<CR> \| :+1<CR>}
" vnoremap <F2> :w! temp.R<CR>
" nnoremap <F3> :let @x = "source('temp.R',echo=T)" . "\n"<CR>
" nnoremap <F4> :call term_sendkeys(term_list()[0], @x)<CR>
" nnoremap <F5> :r !R -q --no-echo -e 'options(echo=F); source("temp.R",ec=T)' \|
"       \ sed 's/^/\# /g'<CR>
" nnoremap <F6> :let @y = "sink('temp2.txt'); source('temp.R',echo=T); sink()" . "\n"<CR>
" nnoremap <F7> :call term_sendkeys(term_list()[0], @y)<CR>
" nnoremap <F8> :r !cat temp2.txt \| sed 's/^/\# /g'<CR>


" " control-j to move to next chunk
" nnoremap <C-j> /```{<CR>j
" " control-l to highlight and run current chunk
" nmap <C-l> ?```{<CR>jV/```<CR>k<C-v>/```{<CR>j/zqzq<CR>

" " control-k to move to prev chunk
" nnoremap <C-k> 2?```{<CR>j
" " control-;' to highlight from cursor to end of chunk
" nnoremap <C-h> V/```<CR>k
" " nnoremap <C-l> V3j
```

```
"
au FocusGained * :let @z=@*
 " autocmd TextChanged,TextChangedI <buffer> silent write
set updatetime=1000
autocmd CursorHold,CursorHoldI * update
autocmd FileType rmd setlocal commentstring=#\ %s
" In vim configuration file
let $FZF_DEFAULT_OPTS = '--bind "ctrl-j:down,ctrl-k:up,j:preview-down,k:preview-up"'
" test turning off autocomments
:set formatoptions-=c formatoptions-=r formatoptions-=o

" Experimental --------------------
" Start interactive EasyAlign in visual mode (e.g. vipga)
xmap ga <Plug>(EasyAlign)

" Start interactive EasyAlign for a motion/text object (e.g. gaip)
nmap ga <Plug>(EasyAlign)
colorscheme one
" colorscheme  OceanicNextLight
" set clipboard+=unnamedplus
" set background=light
set background=dark
```

4 .vimrc

5 ~/.vim/ftplugin/rmd.vim

```
tnoremap ZD quarto::quarto_render(output_format = "pdf")<CR>
tnoremap ZO source("<C-W>"%")
tnoremap ZQ q('no')<C-\><C-n>:q!<CR>
tnoremap ZR render("<C-W>"%")<CR>
nnoremap ZT :!R -e 'render("<C-r>%", output_format="pdf_document")'<CR>
tnoremap ZS style_dir()<CR>
tnoremap ZX exit<CR>
tnoremap ZZ q('no')<C-\><C-n>:q!<CR>



nnoremap <localleader>d :let @c=expand("<cword>") \| :let @d="dim(".@c.")"."\n"   \| :call ter
```

```
nnoremap <localleader>h :let @c=expand("<cword>") \| :let @d="head(".@c.")"."\n"   \| :call ter
nnoremap <localleader>s :let @c=expand("<cword>") \| :let @d="str(".@c.")"."\n"   \| :call term
nnoremap <localleader>p :let @c=expand("<cword>") \| :let @d="print(".@c.")"."\n"   \| :call te
nnoremap <localleader>n :let @c=expand("<cword>") \| :let @d="names(".@c.")"."\n"   \| :call te



" autocmd BufEnter * if &ft ==# 'rmd'  && !exists('b:entered') | execute('let b:entered = 1 |
autocmd BufEnter * if &ft ==# 'rmd'  && !exists('b:entered') | execute('let b:entered = 1 | :te
```

6 ~/.vim/ftplugin/r.vim

```
tnoremap ZD quarto::quarto_render(output_format = "pdf")<CR>
tnoremap ZO source("<C-W>"#")
tnoremap ZQ q('no')<C-\><C-n>:q!<CR>
tnoremap ZR render("<C-W>"#")
tnoremap ZS style_dir()<CR>
tnoremap ZX exit<CR>
tnoremap ZZ q('no')<C-\><C-n>:q!<CR>

" nnoremap <localleader>d "byiw \| :let @a="dim(".@b.")"."\n" <CR>  \| :call term_sendkeys(term
" nnoremap <localleader>h "byiw \| :let @a="head(".@b.")"."\n" <CR>  \| :call term_sendkeys(ter
" nnoremap <localleader>s "byiw \| :let @a="str(".@b.")"."\n" <CR>  \| :call term_sendkeys(term
" nnoremap <localleader>p "byiw \| :let @a="print(".@b.")"."\n" <CR>  \| :call term_sendkeys(te
" nnoremap <localleader>n "byiw \| :let @a="names(".@b.")"."\n" <CR>  \| :call term_sendkeys(te



nnoremap <localleader>d :let @c=expand("<cword>") \| :let @d="dim(".@c.")"."\n"   \| :call term
nnoremap <localleader>h :let @c=expand("<cword>") \| :let @d="head(".@c.")"."\n"   \| :call ter
nnoremap <localleader>s :let @c=expand("<cword>") \| :let @d="str(".@c.")"."\n"   \| :call term
nnoremap <localleader>p :let @c=expand("<cword>") \| :let @d="print(".@c.")"."\n"   \| :call te
nnoremap <localleader>n :let @c=expand("<cword>") \| :let @d="namers(".@c.")"."\n"   \| :call t



" autocmd BufEnter * if &ft ==# 'r'    && !exists('b:entered') | execute('let b:entered = 1 |
autocmd BufEnter * if &ft ==# 'r'    && !exists('b:entered') | execute('let b:entered = 1 | :te
```

7 Practical application

1) set analysis goal: Logistic regression of Palmer Penguins data set predicting gender.

Start with a barebones system. i.e. only vim, R and latex installed.

Step one: add the minimum to vim to allow rmarkdown development. Simplest approach: open vim with empty analysis file. p.Rmd

```
> cd ~/prj/qblog/posts/setup_R_vimtex_ultisnips/penguins
> vim -u .myvimrc p.Rmd
# first line for analysis is to load the penguin data
# enter insert mode and type first R command
inside_vim_normal_mode> i
inside_vim_insert_mode> library(palmerpenguins)
# exit insert mode
inside_vim_insert_mode> C-c
# yank (copy) line into register (unnamed register ")
inside_vim_normal_mode> yy
# open a terminal inside vim and run the R repl
inside_vim_command_mode> term
inside_vim_term> R
# paste last yank (stored in register ") from p.Rmd buffer to R repl
inside_vim_terminal_running_R> C-w ""
```

Next we want to add some plugins to .myvimrc to allow ultisnips snippets for rmd files. Install the vim-plug Vim plugin manager

```
sh> curl -fLo ~/.vim/autoload/plug.vim --create-dirs \
    https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim

in vimrc> call plug#begin

in vimrc> call plug#end()

2) mkdir ~/sbx/penguins
3) mvim peng.Rmd
4) type rheader on the first line and hit TAB. The will match the
snippet string in Ultisnips and insert a text template with YAML markup
and latex text (for bibtex use). The snippet has X tabstops to allow
customization of the text block. Enter text in the first block
indicating the project name and then hit C-j to navigate to the next tab
stop: the title. Repeat the process to provide the project specific
information in the YAML and bibliography call. NB: don't leave insert
mode when navigating between tabstops or the ultisnip process with exit.
```

8 Rmd template

8.1 YAML header

The RMD file contains a YAML metadata header delineated with the lines "—" above and below. For this example we want to generate a pdf formatted output file.

The YAML can be as simple as one line specifying the output as pdf.

```
---
output: pdf_document
---
```

Which results in a simple output file as follows:

NB. to invoke file completion in vim for the rmd (or quarto) change the vim filetype using the command:

```
:set filetype=tex
```

then enter, e.g., `\includegraphics{` or `\input{` followed by `C-x C-o`. and a pop-up menu with possible completions with appear.

```
---
title: "Penguins data analysis"
author: "R.G. Thomas"
date: "`r Sys.Date()`"
output:
  pdf_document:
    keep_tex: true
    includes:
header-includes:
    - \usepackage{lipsum, fancyhdr, titling, currfile}
    - \usepackage[export]{adjustbox}
    - \pagestyle{fancy}
    - \pretitle{
    - \begin{flushright} \includegraphics[width=3cm,valign=c]{sudoku.pdf}
    - \end{flushright}
    - \noindent\rule{\linewidth}{2pt}\begin{flushleft}\LARGE}
    - \posttitle{\end{flushleft}\noindent\rule{\linewidth}{2pt}}
---
```

9 Introduction

Begin by loading the palmerpenguins package.

```
df1 <- palmerpenguins::penguins
```

and

```
---
```

10 Appendix

https://www.reddit.com/r/vim/comments/7c7wd9/vim_vimtex_zathura_on_macos/

https://stackoverflow.com/questions/40077211/e185-cannot-find-color-scheme

https://github.com/morhetz/gruvbox/issues/219

https://github.com/junegunn/vim-plug/issues/325

https://github.com/dylanaraps/pywal/wiki/Getting-Started

https://github.com/dylanaraps/wal.vim

https://github.com/dylanaraps/pywal/wiki/Customization

https://github.com/lervag/vimtex/issues/1420

https://latextools.readthedocs.io/en/latest/install/

https://mg.readthedocs.io/latexmk.html

https://gist.github.com/LucaCappelletti94/920186303d71c85e66e76ff989ea6b62

https://github.com/lervag/vimtex/issues/1420

https://latextools.readthedocs.io/en/latest/install/

https://github.com/lervag/vimtex/issues/1420

https://github.com/lervag/vimtex/issues/940

https://github.com/lervag/vimtex/issues/663

http://www.math.cmu.edu/~gautam/sj/blog/20140310-zathura-fsearch.html

https://gitter.im/SirVer/ultisnips

https://github.com/SirVer/ultisnips/issues/1107

https://github.com/SirVer/ultisnips/issues/1022

https://github.com/SirVer/ultisnips/issues/850

https://superuser.com/questions/1115159/how-do-i-install-vim-on-osx-with-python-3-support

https://jdhao.github.io/2020/01/05/ultisnips_python_interpolation/

http://witkowskibartosz.com/blog/python_snippets_in_vim_with_ultisnips.html#.Xnw9gtP7TRY

https://germaniumhq.com/2019/02/07/2019-02-07-Vim-Ultimate-Editing:-UltiSnips/

http://vimcasts.org/episodes/ultisnips-python-interpolation/

https://wraihan.com/posts/vimtex-and-zathura/

11 Appendix

Use tab for a) ultisnip snippet navigation (move from one tabstop to the next) and b) for navigation forward inside popup menu

```
let g:UltiSnipsExpandTrigger="<C-tab>"
let g:UltiSnipsJumpForwardTrigger="<tab>"
let g:UltiSnipsJumpBackwardTrigger="<S-tab>"

inoremap <expr> <tab> pumvisible() ? "\<C-n>" : "\<tab>"
inoremap <expr> <S-tab> pumvisible() ? "\<C-p>" : "\<S-tab>"
```

11.1 Prerequisites

In development

11.2 Step-by-Step Implementation

In development

11.3 Key Takeaways

In development

11.4 Further Reading

In development