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

Ronald (Ryy) Glenn Thomas

11/17/23

Table of contents

Figure 1: vim setup

1 Introduction

Start by installing vim (neovim), R vimtex ultisnips

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

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/

2 Sections

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

- R
- julia
- python

2. ftplugins
3. ultisnips
    - dynamic
4. vimtex

- completion

3 .vimrc

```
unlet! skip_defaults_vim
source $VIMRUNTIME/defaults.vim
set scrolloff=2
syntax enable
filetype plugin indent on
let mapleader = ","
let maplocalleader = " "
" ESSENTIAL plugins
"
call plug#begin()
Plug 'jpalardy/vim-slime'
Plug 'klafyvel/vim-slime-cells'
Plug 'lervag/vimtex'
Plug 'sirver/ultisnips'
" Plug 'jalvesaq/Nvim-R'

"
let g:vimtex_complete_close_braces=1
let g:vimtex_quickfix_mode=0
" let g:UltiSnipsSnippetDirectories = ['~/.vim/UltiSnips']
```

```
" let g:UltiSnipsJumpForwardTrigger="<tab>"
" let g:UltiSnipsJumpBackwardTrigger="<c-k>"
"
"
"plugins for IDE like enhancements
"
" let g:ale_linters_explicit = 1
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 = '.'
" In ~/.vim/vimrc, or somewhere similar.
let g:ale_linters = {
\   'python': ['pylsp'],
\}
let g:ale_fixers = {
\   '*': ['remove_trailing_lines', 'trim_whitespace'],
\   'r': ['styler'],
\   'python': ['black','isort'],
\   'javascript': ['eslint'],
\}
" Set this variable to 1 to fix files when you save them.
let g:ale_fix_on_save = 1
Plug 'dense-analysis/ale'
"
Plug 'vifm/vifm.vim'
Plug 'junegunn/fzf', { 'do': { -> fzf#install() } }
Plug 'junegunn/fzf.vim'
Plug 'junegunn/vim-peekaboo'
Plug 'tpope/vim-unimpaired'
" Plug 'ervandew/supertab'
Plug 'justinmk/vim-sneak'
Plug 'tpope/vim-obsession'
Plug 'tpope/vim-repeat'
Plug 'tpope/vim-commentary'
Plug 'tpope/vim-surround'
```

```
Plug 'machakann/vim-highlightedyank'
Plug 'vim-airline/vim-airline'
let g:airline#extensions#ale#enabled = 1
let g:sneak#s_next = 1
let g:sneak#use_ic_scs = 1
let g:airline#extensions#tabline#enabled = 1
let g:airline#extensions#fzf#enabled = 1
call plug#end()
"
" optional settings
"
if $COLORTERM == 'truecolor'
  set termguicolors
endif
colorscheme lightning
set tabstop=2
set shiftwidth=2
set autoindent
set expandtab
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=72
set colorcolumn=80
set cursorline
set hlsearch
set splitright
set hidden
set incsearch
set noswapfile
set showmatch
```

```
set ignorecase
set smartcase
set gdefault
"
" optional mappings
"
inoremap <expr> <TAB> pumvisible() ? "\<C-n>" : "\<TAB>"
" inoremap <expr> <S-TAB> pumvisible() ? "\<C-p>" : "\<TAB>"
inoremap <F10> <C-x><C-k>
inoremap <F12> <C-x><C-o>
inoremap <silent> <Esc> <Esc>`^
"
nnoremap <leader>1 <C-w>:b1<CR>
nnoremap <leader>f :Files<CR>
nnoremap <Leader>' :Marks<CR>
nnoremap <Leader>/ :BLines<CR>
nnoremap <Leader>b :Buffers<CR>
tnoremap <leader>b <C-w>:Buffers<cr>
nnoremap <Leader>r :Rg<CR>
nnoremap <Leader>s :Snippets<CR>
nnoremap <leader><leader> <C-w>w
nnoremap <leader>U <Cmd>call UltiSnips#RefreshSnippets()<CR>
nnoremap <leader>a ggVG
nnoremap <leader>d :EditVifm<cr>
nnoremap <leader>n :ALENext<CR>
nnoremap <leader>t :tab split<cr>
nnoremap <leader>u :UltiSnipsEdit<cr>
nnoremap <leader>v :edit ~/.vimrc<cr>
nnoremap <localleader><leader> <C-u>
nnoremap <localleader><localleader> <C-d>
nnoremap <leader>m vipgq
nnoremap Q @@
noremap - $
noremap : ;
noremap ; :
"
tnoremap <F1> <C-\><C-n>
tnoremap <leader>1 <C-w>:b1<CR>
" tnoremap <leader>0 <C-w>:ls<cr>:b<localleader>
```

```
" tnoremap <leader>2 <C-w>:b2<cr>
" tnoremap <leader>3 <C-w>:b3<cr>
tnoremap <leader><leader> <C-w>w
tnoremap ZQ q('no')<C-\><C-n>:q!<CR>
tnoremap ZS style_dir()<CR>
tnoremap ZX exit<CR>
tnoremap ZZ q('no')<C-\><C-n>:q!<CR>
tnoremap ZD quarto::quarto_render(output_format = "pdf")<CR>

let g:slime_target = "vimterminal"
let g:slime_no_mappings = 1
let g:slime_vimterminal_config = { "vertical": 1 }
let g:slime_vimterminal_cmd = "R"
" match either ### (R) or ``` (rmd, quarto) to delimit cells
let g:slime_cell_delimiter = "^\\s*\\(###\\|```\\)"
nmap <C-c>v <Plug>SlimeConfig
nmap <localleader>l <Plug>SlimeCellsSendAndGoToNext
nmap <localleader>j <Plug>SlimeCellsNext
nmap <localleader>k <Plug>SlimeCellsPrev

" send current line to R
autocmd FileType r,rmd,quarto,tex,julia nnoremap        <CR>   V<Plug>SlimeRegionSend+
" send current paragraph to R
autocmd FileType r,rmd,quarto,tex,julia nnoremap <space><CR> vip<Plug>SlimeRegionSend}
"
" send chunk to R
" autocmd FileType r,rmd,quarto,tex nnoremap <leader><space><CR> <Plug>SlimeSendCell<CR>
"
" send chunk to R  (alternate)
" autocmd FileType r,rmd,quarto,tex nnoremap <localleader>l <Plug>SlimeSendCell<CR>
"
" send selected text to R
autocmd FileType r,rmd,quarto,tex,julia xnoremap        <CR>    <Plug>SlimeRegionSend+
"
" <space>k send from current line to end of chunk to R. constructed as map
" autocmd FileType r,rmd,quarto,tex nnoremap <space>k @k
"
" maps for R and RMD, space + letter. def as macros. stored in " ~/.viminfo
" j jump to next chunk
```

```
" h print head() of object
" p print() object
" d dim() of object
" i dim() of object
" n names() of object
" s str() of object
" o str() of object
" b length() of object
" k send from current line to end of chunk to R
" ' render rmd or qmd file to pdf
" r render rmd or qmd file to pdf
autocmd FileType r,rmd,quarto,tex nnoremap <space>' @a
autocmd FileType r,rmd,quarto,tex nnoremap <space>b @b
autocmd FileType r,rmd,quarto,tex nnoremap <space>d @d
autocmd FileType r,rmd,quarto,tex nnoremap <space>h @h
autocmd FileType r,rmd,quarto,tex nnoremap <space>i @i
" autocmd FileType r,rmd,quarto,tex nnoremap <space>j @j
autocmd FileType r,rmd,quarto,tex nnoremap <space>n @n
autocmd FileType r,rmd,quarto,tex nnoremap <space>p @p
autocmd FileType r,rmd,quarto,tex nnoremap <space>s @s
autocmd FileType r,rmd,quarto,tex nnoremap <space>r @r
autocmd FileType r,rmd,quarto,tex nnoremap <space>a @a
```

4 .zshrc

```
export ZSH="$HOME/.oh-my-zsh"
export EDITOR="vim"
ZSH_THEME="muse"
DISABLE_UNTRACKED_FILES_DIRTY="true"
plugins=(zsh-autosuggestions z git pass vi-mode scd common-aliases you-should-use)

source $ZSH/oh-my-zsh.sh

bindkey -v
alias mm='mutt'
alias sk='open -a Skim'
alias vc='vim ~/.vimrc'
```

```
alias vz='vim ~/.zshrc'
alias sz='source ~/.zshrc'
alias p2='enscript -C -2 -r -j --media=Letter'
alias p1='enscript  -j --media=Letter'
alias yr="yabai --restart-service"
alias lt='eza -lrFha -sold'
alias mvim="/Applications/MacVim.app/Contents/bin/mvim"
alias tp='trash-put -v'
alias rm='echo "This is not the command you are looking for."; false'
alias s='scd'
alias ZZ='exit'
alias r="radian"
alias nt="nvim"
alias ng="neovim-qt"
mma () { /Applications/Mathematica.app/Contents/MacOS/WolframKernel -script $1 }
function gz() {
    git add .
    git commit -a -m "$1"
    git push
}

cdpath=($HOME/Dropbox/prj $HOME/Dropbox/sbx $HOME/Dropbox/work )
export TEXINPUTS='.:/Users/zennjshr/images:/Users/zenn/shr:'
export PATH=".:/opt/homebrew/sbin:/opt/homebrew/bin:$PATH:$HOME/bin"

export vpc_id="vpc-14814b73"
export subnet_id="subnet-f02c90ab"
export ami_id="ami-014d05e6b24240371"
export keypair_name="power1_app"
export proj_name="power1_app"
export instance_type="t2.micro"
export storage_size="30"
export ami_id="ami-014d05e6b24240371"
export security_grp='sg-0b6184a66019ebffe'
export static_ip='54.176.238.177'

if type rg &> /dev/null; then
  export FZF_DEFAULT_COMMAND='rg --files --hidden --no-ignore-vcs'
  export FZF_DEFAULT_OPTS='-m --height 50% --border'
fi
```

```
ZSH_AUTOSUGGEST_HIGHLIGHT_STYLE="fg=011,bg=black,bold,underline"

test -e "${HOME}/.iterm2_shell_integration.zsh" && source "${HOME}/.iterm2_shell_integration
```

```
" Send current line to R.
function SendLineToR(godown, ...)
    let lnum = get(a:, 1, ".")
    let line = getline(lnum)
    if strlen(line) == 0
        if a:godown =~ "down"
            call GoDown()
        endif
        return
    endif

    if &filetype == "rnoweb"
        if line == "@"
            if a:godown =~ "down"
                call GoDown()
            endif
            return
        endif
        if line =~ "^<<.*child *= *"
            call KnitChild(line, a:godown)
            return
        endif
        if RnwIsInRCode(1) != 1
            return
        endif
    endif

    if &filetype == "rmd" || &filetype == "quarto"
        if line == "```"
            if a:godown =~ "down"
                call GoDown()
            endif
            return
        endif
        if line =~ "^```.*child *= *"
```

```
        call KnitChild(line, a:godown)
        return
    endif
    let line = substitute(line, "^(\\`\\`)\\?", "", "")
    if RmdIsInRCode(0) != 1
        if RmdIsInPythonCode(0) == 0
            call RWarningMsg("Not inside an R code chunk.")
            return
        else
            let line = 'reticulate::py_run_string("' . substitute(line, '"', '\\"', 'g')
        endif
    endif
endif

if &filetype == "rrst"
    if line == ".. .."
        if a:godown =~ "down"
            call GoDown()
        endif
        return
    endif
    if line =~ "^\.\. {r.*child *= *"
        call KnitChild(line, a:godown)
        return
    endif
    let line = substitute(line, "^\\.\\. \\?", "", "")
    if RrstIsInRCode(1) != 1
        return
    endif
endif

if &filetype == "rdoc"
    if getline(1) =~ '^The topic'
        let topic = substitute(line, '.*::', '', "")
        let package = substitute(line, '::.*', '', "")
        call AskRDoc(topic, package, 1)
        return
    endif
    if RdocIsInRCode(1) != 1
        return
```

```
    endif
endif

if &filetype == "rhelp" && RhelpIsInRCode(1) != 1
    return
endif

if &filetype == "r"
    let line = CleanOxygenLine(line)
endif

let block = 0
if g:R_parenblock
    let chunkend = ""
    if &filetype == "rmd" || &filetype == "quarto"
        let chunkend = "```"
    elseif &filetype == "rnoweb"
        let chunkend = "@"
    elseif &filetype == "rrst"
        let chunkend = ".. .."
    endif
    let rpd = RParenDiff(line)
    let has_op = substitute(line, '#.*', '', '') =~ g:rplugin.op_pattern
    if rpd < 0
        let line1 = line(".")
        let cline = line1 + 1
        while cline <= line("$")
            let txt = getline(cline)
            if chunkend != "" && txt == chunkend
                break
            endif
            let rpd += RParenDiff(txt)
            if rpd == 0
                let has_op = substitute(getline(cline), '#.*', '', '') =~ g:rplugin.op_p
                for lnum in range(line1, cline)
                    if g:R_bracketed_paste
                        if lnum == line1 && lnum == cline
                            let ok = g:SendCmdToR("\x1b[200~" . getline(lnum) . "\x1b[20
                        elseif lnum == line1
```

```
                        let ok = g:SendCmdToR("\x1b[200~" . getline(lnum))
                    elseif lnum == cline
                        let ok = g:SendCmdToR(getline(lnum) . "\x1b[201~\n", 0)
                    else
                        let ok = g:SendCmdToR(getline(lnum))
                    endif
                else
                    let ok = g:SendCmdToR(getline(lnum))
                end
                if !ok
                    " always close bracketed mode upon failure
                    if g:R_bracketed_paste
                        call g:SendCmdToR("\x1b[201~\n", 0)
                    end
                    return
                endif
            endfor
            call cursor(cline, 1)
            let block = 1
            break
        endif
        let cline += 1
    endwhile
  endif
endif

if !block
  if g:R_bracketed_paste
    let ok = g:SendCmdToR("\x1b[200~" . line . "\x1b[201~\n", 0)
  else
    let ok = g:SendCmdToR(line)
  end
endif

if ok
  if a:godown =~ "down"
    call GoDown()
    if exists('has_op') && has_op
      call SendLineToR(a:godown)
    endif
```

```
        else
            if a:godown == "newline"
                normal! o
            endif
        endif
    endif
endfunction
```