

A simple vim package for interfacing with a REPL

Ronald (Ryy) Glenn Thomas

2024-06-28

Table of contents

1	Introduction	1
2	Experiment	4
2.1	Title: Add the normal mode mapping ZY for quarto files.	4
2.2	Introduction:	4

1 Introduction

Start with youtube Chris T.

<https://www.youtube.com/watch?v=lwD8G1P52Sk>

Start on [zz.tools.vim-R.vim](https://zz.tools/vim-R.vim)

Here is the code for the plugin:

From .vimrc

```
" vim - How to get visually selected text in VimScript - Stack Overflow
" https://stackoverflow.com/questions/1533565/how-to-get-visually-selected-text-in-vimscript
"

function! SelectChunk()
```



```

:execute "normal! ?```\<cr>jV/```\<cr>k"
endfunction

function! MoveNextChunk()
:execute "normal! /```\<CR>j"
:noh
endfunction

function! Raction(action)
:let @c = expand("<cword>")
:let @d=a:action . "(".@c.")\n"
:call term_sendkeys(term_list()[0], @d)
endfunction

function! SubmitLine()
:let @c = getline(".") . "\n"
:call term_sendkeys(term_list()[0], @c)
endfunction

function! GetVisualSelection(mode)
" call with visualmode() as the argument
let [line_start, column_start] = getpos("'"<")[1:2]
let [line_end, column_end] = getpos("'">")[1:2]
let lines = getline(line_start, line_end)
if a:mode ==# 'v'
" Must trim the end before the start, the beginning will shift left.
let lines[-1] = lines[-1][: column_end - (&selection == 'inclusive' ? 1 : 2)]
let lines[0] = lines[0][column_start - 1:]
elseif a:mode ==# 'V'
" Line mode no need to trim start or end
elseif a:mode == "\<c-v>"
" Block mode, trim every line
let new_lines = []
let i = 0
for line in lines
let lines[i] = line[column_start - 1: column_end - (&selection == 'inclusive' ? 1 : 2)]
let i = i + 1
endfor
else

```

```

        return ''
    endif
    for line in lines
        echom line
    endfor
    return join(lines, "\n")
endfunction

function! SubmitSelTest()
y
:let @c=@ " . "\n"
:call term_sendkeys(term_list()[0], @c)
endfunction

function! SubmitSel()
:let @c= GetVisualSelection(visualmode()) . "\n"
:call writefile(getreg('c', 1, 1), "temp.R")
:let @y = "source('temp.R',echo=T)" . "\n"
:call term_sendkeys(term_list()[0], @y)
endfunction

nnoremap <silent> <CR> :call SubmitLine()<CR><CR>
vnoremap <silent> <CR> :call SubmitSel()<CR><CR>
vnoremap <silent> <S-CR> :call SubmitSelTest()<CR><CR>

noremap <silent> <localleader>l :call SelectChunk()<CR> \|| :call SubmitSelTest()<CR>
noremap <silent> <localleader>; :call SelectChunk()<CR> \|| :call SubmitSelTest()<CR> \|| /```{<
nnoremap <silent> <C-CR> :call MoveNextChunk()<CR>

nnoremap <localleader>k 2?```{<CR>j
nnoremap <localleader>j /```{<CR>j

nnoremap <silent> <localleader>r :vert term R --no-save<CR><c-w>:wincmd p<CR>
nnoremap ZT :!R --quiet -e 'render("<C-r>%", output_format="pdf_document")'<CR>
nnoremap ZY :!R --quiet -e 'quarto_render("<C-r>%", output_format="pdf")'<CR>

tnoremap ZD quarto::quarto_render(output_format = "pdf")<CR>

```

```

tnoremap ZO source("<C-W>\"%)
tnoremap ZR render("<C-W>\"%)<CR>
tnoremap ZS style_dir()<CR>
tnoremap ZQ q('no')<C-\\><C-n>:q!<CR>
tnoremap ZZ q('no')<C-\\><C-n>:q!<CR>
tnoremap lf ls()<CR>

nnoremap <localleader>d :call Raction("dim")<CR>
nnoremap <localleader>h :call Raction("head")<CR>
nnoremap <localleader>s :call Raction("str")<CR>
nnoremap <localleader>p :call Raction("print")<CR>
nnoremap <localleader>n :call Raction("names")<CR>
nnoremap <localleader>f :call Raction("length")<CR>
vnoremap <silent> <localleader>z :w! temp.R<CR> \\
\\ :let @y = "sink('temp.txt'); source('temp.R',echo=T); sink()" . "\\n"<CR>
\\ :call term_sendkeys(term_list()[0], @y)<CR> \\
\\ :r !cat temp.txt \\ sed 's/^/\\# /g'<CR>

```

2 Experiment

2.1 Title: Add the normal mode mapping ZY for quarto files.

2.2 Introduction:

The goal is to allow quarto filetypes to render to pdf using a mapping called from the qmd file.

- start by constructing a mapping in .vimrc: (easier to develop there) map ZY to a shell escape and call to quarto_render. (use ZT map in rgt-R.vim as a template).
- test using any index.qmd file in posts. e.g. ~/config_ultisnips/index.qmd.
- once the mapping works then move it to the plugin and add a autocommand that only adds the mapping for quarto filetype files.

- open `~/prj/qblog/posts/vim_plugin_zz.tools.vim-R/rgt-R/plugin/rgt-R.vim`
- copy ZT mapping to ZY
- modify ZY to render quarto files with `render_quarto` command.