A simple vim package for interfacing with a REPL

Ronald (Ryy) Glenn Thomas

2024-06-23

Table of contents

1 Introduction

Start with youtube Chris T.

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

Start on zz.tools.vim-R.vim

```
vim9script noclear
```

2 Appendix

```
source("~/shr/zz.tools.R")
library(pacman)

p_load(dplyr, gapminder, thematic, palmerpenguins,
       tidyverse, knitr, lubridate, readxl)
```

The downloaded binary packages are in /var/folders/j3/x1bzm4fn28qcq0hrv3kndvwm0000gn/T//RtmpTldRCR/d

```
gapminder installed
```

The downloaded binary packages are in /var/folders/j3/x1bzm4fn28qcq0hrv3kndvwm0000gn/T//RtmpTldRCR/d

```
thematic installed
```

```
knitr::opts_chunk$set(collapse = T)
set.seed(101)
dat <- palmerpenguins::penguins %>%
  fil(!is.na(sex))
dat1 <- slice_sample(dat, n = 10) |>
  sel(species, island, bill_length_mm)
```

Here is the code for the plugin:

From .vimrc

```
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 <F2> :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>
```

```
" 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




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>r :vert term R <CR><c-w>:wincmd p<CR>

nnoremap <localleader>d :let @c=expand("<cword>") \|
\ :let @d="dim(".@c.")"."\n"   \| :call term_sendkeys(term_list()[0], @d)<CR>
nnoremap <localleader>h :let @c=expand("<cword>") \|
\ :let @d="head(".@c.")"."\n"   \| :call term_sendkeys(term_list()[0], @d)<CR>
nnoremap <localleader>s :let @c=expand("<cword>") \|
\ :let @d="str(".@c.")"."\n"   \| :call term_sendkeys(term_list()[0], @d)<CR>
nnoremap <localleader>p :let @c=expand("<cword>") \|
\ :let @d="print(".@c.")"."\n"   \| :call term_sendkeys(term_list()[0], @d)<CR>
nnoremap <localleader>n :let @c=expand("<cword>") \|
\ :let @d="names(".@c.")"."\n"   \| :call term_sendkeys(term_list()[0], @d)<CR>
nnoremap <localleader>l :let @c=expand("<cword>") \|
\ :let @d="length(".@c.")"."\n"   \| :call term_sendkeys(term_list()[0], @d)<CR>

tnoremap lf ls()<CR> "list files
```

3 Experiment

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

3.2 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.