Rainbows

RStudio > Tools > Global Options...
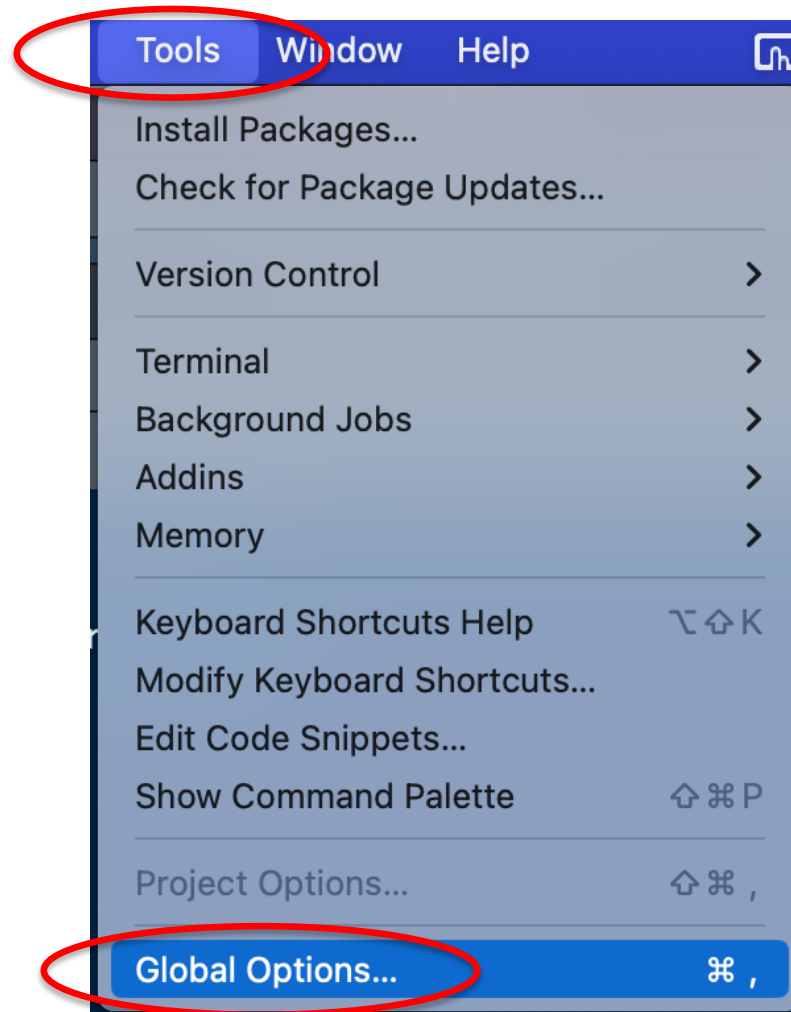
Rainbows
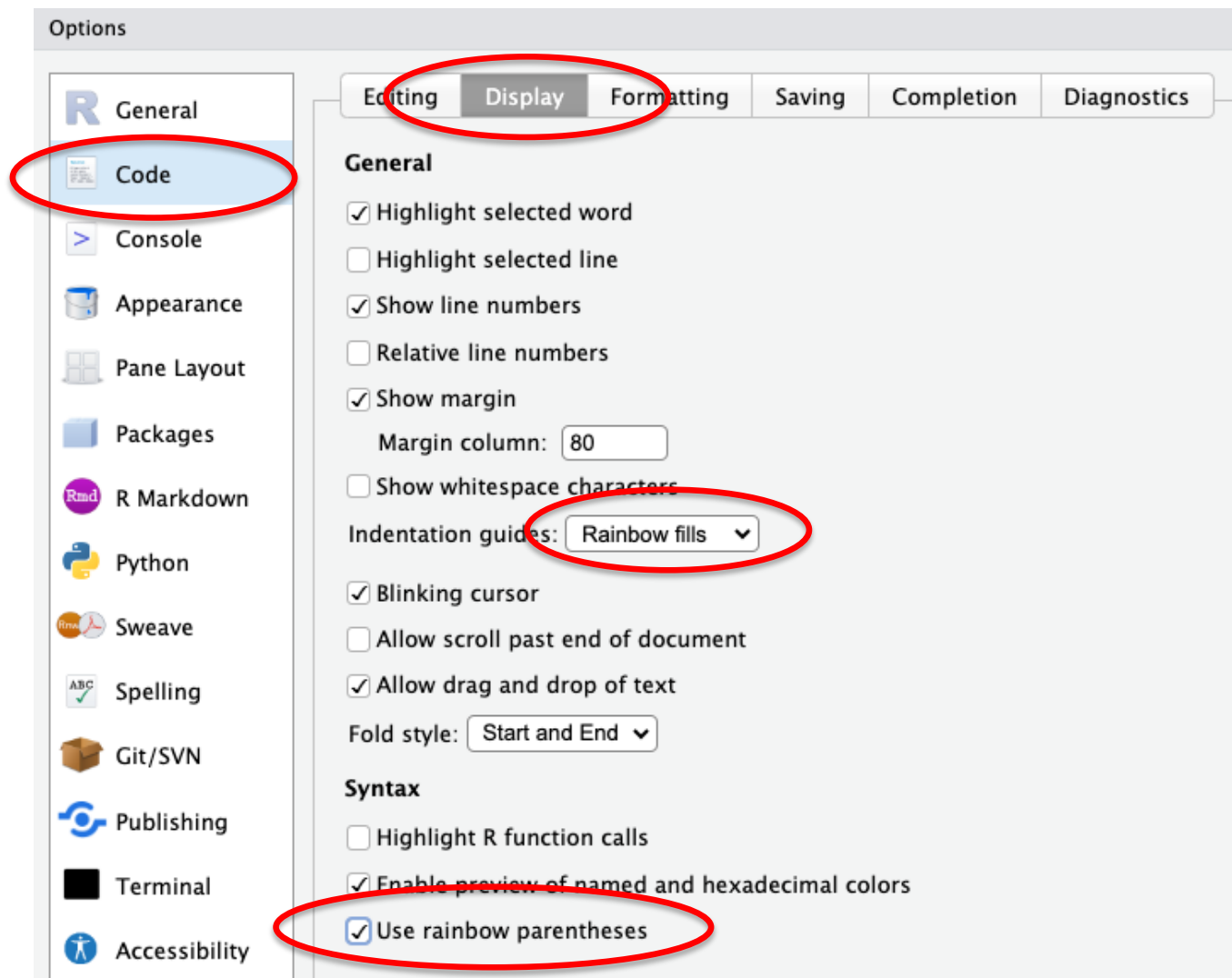
Connected Symbols

Symbols before

`<- -> >= <= |> != == |`

Symbols after

`← → ≥ ≤ ▷ ≠ = |`

Install font: 'FiraCode-Regular'
https://github.com/tonsky/FiraCode

OR

Install font: 'JetBrainsMono-Regular'
https://www.jetbrains.com/lp/mono/

Connected Symbols

Background Jobs

Snippets
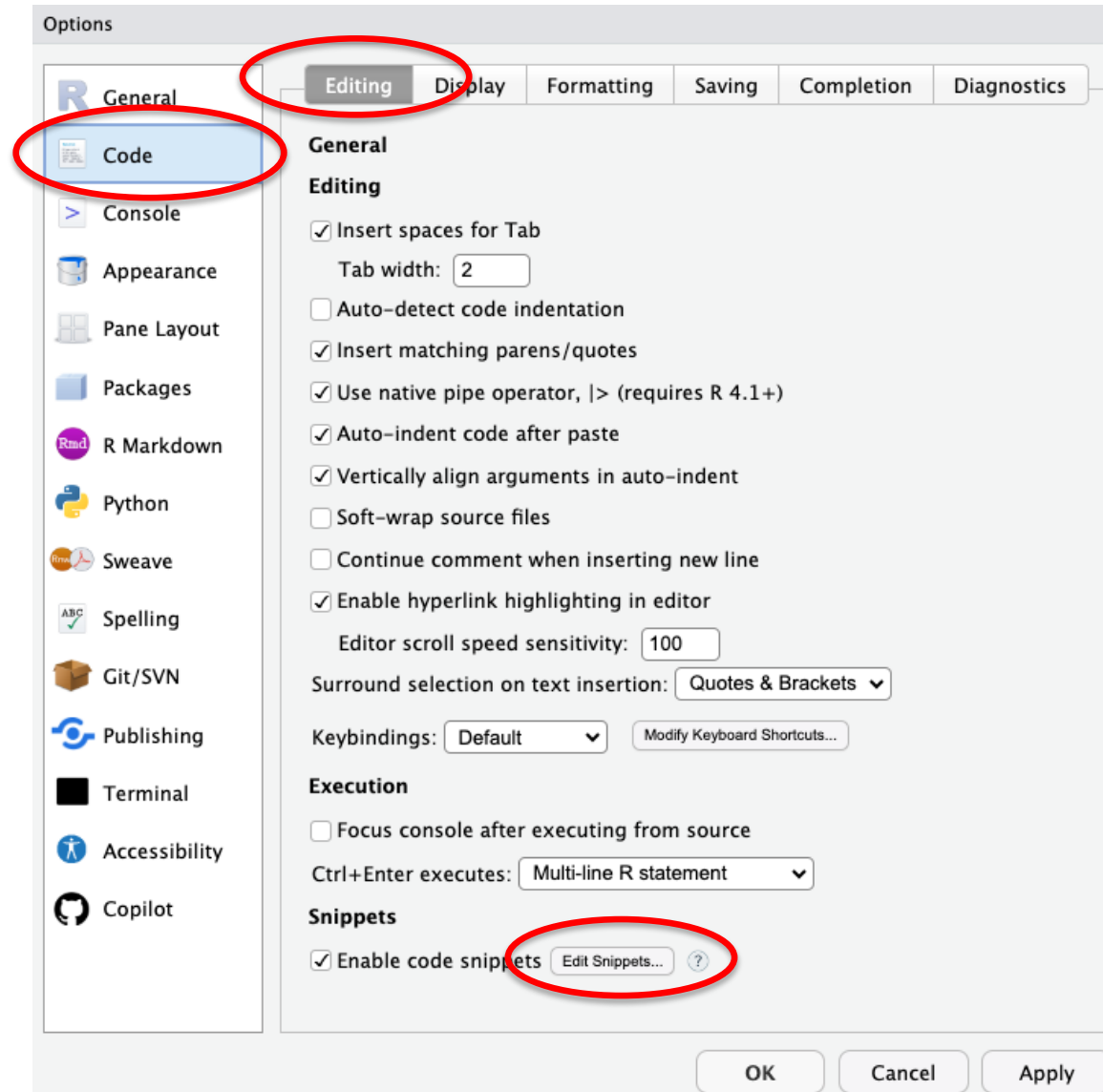
Snippets

- R
- C/C++
- Markdown
- TeX
- JavaScript
- HTML
- CSS
- SQL
- Java
- Python
- Stan
- YAML

```
144 ▾ snippet dplyr
145       dplyr::
146   |
147 ▾ snippet mutate
148       dplyr::mutate(${1})
149
150 ▾ snippet filter
151       dplyr::filter(${1})
152
153 ▾ snippet select
154       dplyr::select(${1})
155
156 ▾ snippet drop
157       tidyr::drop_na(${1})
158
159 ▾ snippet distinct
160       dplyr::distinct(${1})
161
162 ▾ snippet pivot_wider
163 ▾     tidyr::pivot_wider(id_cols = ${1:vector_of_col_names},
164                             names_from = ${2:column_name},
165                             values_from = ${3:vector_name})
166
167 ▾ snippet pivot_longer
168 ▾     tidyr::pivot_longer(cols = ${1:vector_of_col_names},
169                             names_to = "${2:column_name}",
170                             values_to = "${3:column_name}")
171
```

Using Code Sn

Save

Git Version Control

Git was created in 2005 by Linus Torvalds. He writes....

git can mean anything, depending on your mood.....

- random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of **get** may or may not be relevant.

- stupid. contemptible and despicable. simple. Take your pick from the dictionary of slang

- "**g**lobal **i**nformation **t**racker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room

- "**g**oddamn **i**diotic **t**ruckload of sh*t": when it breaks

This is a stupid (but extremely fast) directory content manager. It doesn't do a whole lot, but what it **does** do is track directory contents efficiently.

Git Version Control

Git Version Control using RStudio

| Introduction | Stage 1 | 2 | 3 | 4 | 5 |

Version control software is a tool to use when you write code.

As its most basic level, it lets you capture snapshots of your progress, along with any commentary notes you want to record about the development you've done so far and why you've done it that way.

It keeps a record of all of the changes to individual lines of code, so you can easily see what changed between snapshots. These snapshots are called 'commits' and can be seen as stepping stones along the journey you take with your code. You can use version control to branch off in new directions as well as retrace your steps to an earlier commit from your journey.

The gains from using version control with a good workflow is that it will streamline working with your most important collaborator: Future You!

This document aims to gradually introduce you to using version control software with RStudio, over several stages:

Stage 1 - The basics

Stage 2 - Setup Local Version Control in RStudio

Stage 3 - Commits

Stage 4 - Reverts

Stage 5 - Branching

github.com/NevilHopley/10Things

List input now controls graphical output

5754dd0b SQAnevilhopley 14 Nov 2024 at 18:48

ui.R now has select from list input

19de3da4 SQAnevilhopley 14 Nov 2024 at 18:46

Slider now controls graphical output

f26cd9e3 SQAnevilhopley 14 Nov 2024 at 18:38

ui.R now has slider input

5a9b3b09 SQAnevilhopley 14 Nov 2024 at 18:35

Radio button now controls graphical output

566d760d SQAnevilhopley 14 Nov 2024 at 17:00

ui.R now has radio button selector

57574f80 SQAnevilhopley 14 Nov 2024 at 16:04

Initial starting position - blank Shiny App, with s...

9dbd5ae5 SQAnevilhopley 14 Nov 2024 at 15:51

server.R -1+2
/app/server.R

View

```
 7           # whilst `df` exists in the global environment
 8           df_plot = df |>
 9             filter(level == input$level &
-                       year %in% input$year_range[1]:input$year_range[2])
10 +                     year %in% input$year_range[1]:input$year_range[2] &
11 +                     subject %in% input$subjects)
12
13           # generate ggplot object
14           ggplot(data = df_plot,
```

Titles and Legends
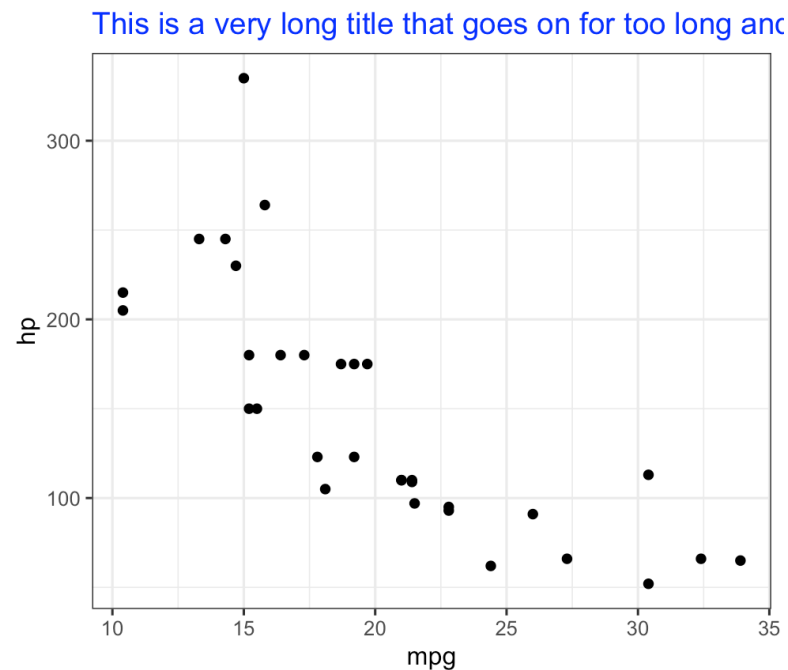
This is a very long title that goes on for too long and


```r
ggplot2::ggplot(data = mtcars,
                mapping = ggplot2::aes(x = mpg,
                                       y = hp)) +
  ggplot2::geom_point() +
  ggplot2::theme_bw() +
  ggplot2::labs(title = "This is a very long title that goes on for too long and does not stop") +
  ggplot2::theme(plot.title = ggplot2::element_text(colour = "blue"))
```

Titles and Legends
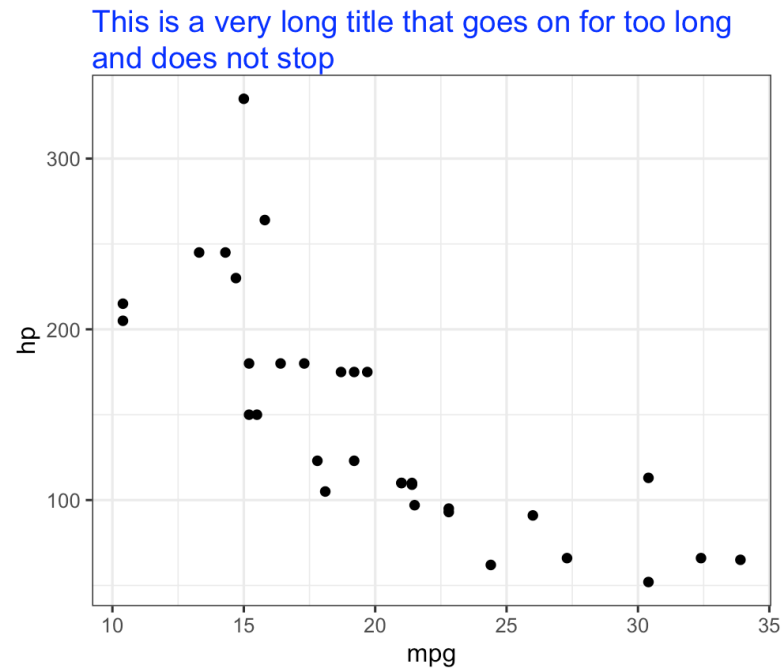

```
ggplot2::ggplot(data = mtcars,
                mapping = ggplot2::aes(x = mpg,
                                       y = hp)) +
  ggplot2::geom_point() +
  ggplot2::theme_bw() +
  ggplot2::labs(title = "This is a very long title that goes on for too long and does not stop") +
  ggplot2::theme(plot.title = ggtext::element_textbox_simple(colour = "blue"))
```

2024 Higher Mathematics grade

A   B   C   D   No Award

2024 Higher Mathematics grade

A   B   C   D   No Award

2024 Higher Mathematics grade

A   B   C   D   No Award

2024 Higher Mathematics grade

A   B   C   D   No Award
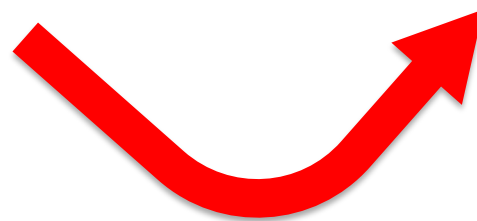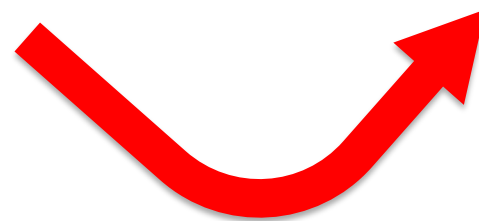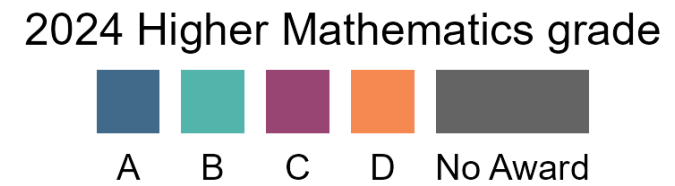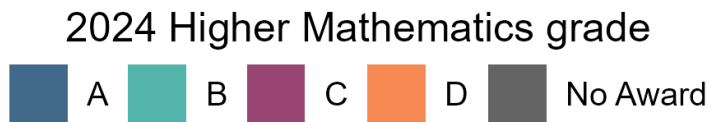
```
legend.text.position = "bottom",
```

2024 Higher Mathematics grade

A   B   C   D   No Award

2024 Higher Mathematics grade

A   B   C   D   No Award
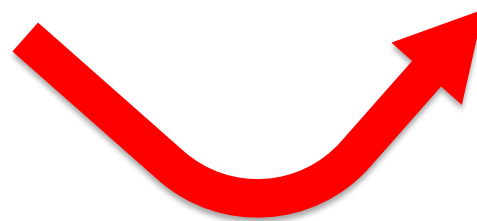
```r
legend.text = ggplot2::element_text(colour = "white",
                                    vjust = 10.5,
                                    face = "bold"),
```

2024 Higher Mathematics grade

A B C D No Award

2024 Higher Mathematics grade
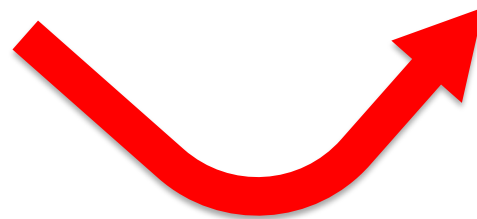
A B C D No Award

```
legend.key.width = unit(c(5.5, 5.5, 5.5, 5.5, 17.5), 'mm'),
```

2024 Higher Mathematics grade

A B C D No Award

2024 Higher Mathematics grade
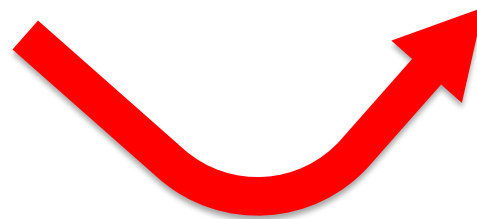
A B C D No Award

```
legend.margin = ggplot2::margin(t = 0, r = 0, b = -25, l = 0, unit = "pt")
```

2024 Higher Mathematics grade

A B C D No Award


```
ggplot2::theme(legend.text.position = "bottom",
        legend.text = ggplot2::element_text(colour = "white",
                                            vjust = 10.5,
                                            face = "bold"),
        legend.key.width = unit(c(5.5, 5.5, 5.5, 5.5, 17.5), 'mm'),
        legend.margin = ggplot2::margin(t = 0, r = 0, b = -25, l = 0, unit = "pt"),
```

Rename and Select

```
data_frame |>
  dplyr::rename(new_name_1 = old_name_1,
                new_name_2 = old_name_2) |>
  dplyr::select(new_name_1,
                new_name_2)
```

```
data_frame |>
  dplyr::select(new_name_1 = old_name_1,
                new_name_2 = old_name_2)
```

Multiple Sourcing

```r
source("R/function_1.R")
source("R/function_2.R")
source("R/function_3.R")
source("R/function_4.R")
source("R/function_5.R")
```

```r
purrr::walk(.x = c("R/function_1.R",
                   "R/function_3.R",
                   "R/function_2.R",
                   "R/function_5.R",
                   "R/function_4.R"),
            .f = ~ source(file = .x),
            .progress = TRUE)
```

Multiple Sourcing

```r
purrr::walk(.x = c("R/function_1.R",
                   "R/function_3.R",
                   "R/function_2.R",
                   "R/function_5.R",
                   "R/function_4.R"),
            .f = ~ source(file = .x),
            .progress = TRUE)
```

```r
purrr::walk(.x = list.files(path = "R",
                            pattern = "*.R",
                            full.names = TRUE),
            .f = ~ source(file = .x),
            .progress = TRUE)
```

Replace all NAs

```r
tidyr::replace_na(replace = list(col_1_name = 0))

tidyr::replace_na(replace = list(col_2_name = ""))
```

```r
dplyr::mutate(dplyr::across(.cols = tidyselect::where(is.numeric),
                            .fns = ~ tidyr::replace_na(., 0)))

dplyr::mutate(dplyr::across(.cols = tidyselect::where(is.character),
                            .fns = ~ tidyr::replace_na(., "")))
```

Conditional Piping


```
syntax for conditional pipe step:

`{\(x) if(<condition>) <function>(x, <function_arguments>) else x}() |>`
or
`{\(y) if(<condition>) <function>(y, <function_arguments>) else y}() |>`
or
use any variable name as the 'anonymous function variable' to carry the data frame
through to the next stage of the pipe if the <condition> is not met

An extension of this that allows for two different things to happen, pending on the
`<condition>` is
```

```
{\(x) if(<condition>)
    <function_if_condition_true>(x, <function_arguments>)
  else
    <function_if_condition_false>(x, <function_arguments>)}() |>
```

Positron

https://positron.posit.co