Tidy eval Cheatsheet

Leonardo Uchôa Pedreira

1/13/2022

General Recipe

Looking not deep, it's kinda simple to create new functions in the tidyverse way. The strategy to make it work is just to `quote` (make it become a symbol) and `unquote` (evaluate the symbol) the function arguments. Let's look an example.

The next function call won't work.

```
grouped_mean <- function(data, group_var, summary_var) {
    data %>%
    group_by(group_var) %>%
    summarise(mean = mean(summary_var))
}

grouped_mean(mtcars, cyl, mpg)

# Error: Must group by variables found in `.data`.
# * Column `group_var` is not found.
# Run `rlang::last_error()` to see where the error occurred.
```

To make it work we need to use the `quote` and `unquote` strategy. So first we quote the arguments with `enquo` and then unquote them with `!!`.

```
grouped_mean <- function(data, group_var, summary_var) {
  group_var <- enquo(group_var)
  summary_var <- enquo(summary_var)

  data %>%
    group_by(!!group_var) %>%
    summarise(mean = mean(!!summary_var))
}

grouped_mean(mtcars, cyl, mpg)
```

```
## # A tibble: 3 x 2
##     cyl  mean
##   <dbl> <dbl>
## 1     4  26.7
## 2     6  19.7
## 3     8  15.1
```

Some Extras

There some key functions when talking about tidy eval. They are:

- `quote`: used outside functions to make objects become symbols
- `enquo`: used inside functions to make objects become symbols. (There's also `enquos`).
- `sym`: used inside and outside functions to make strings become symbols. There's also `syms`, used for vectors, like `syms(c('cyl','disp'))`. Example:
  - this works `syms(c('cyl','disp'))`
  - this doesn't `sym(c('cyl','disp'))`
- `!!`: used to unquote symbols
- `!!!`: used to unquote vectorized symbols. For example:
  - this works `group_by(mtcars,!!!syms(c('cyl','disp')))`
  - this doesn't `group_by(mtcars,!!syms(c('cyl','disp')))`

Using with `ggplot2`

If we're creating a plotting function like `plt <- function(df,col)`, we have two approaches. The first is to make `col` be a string and the second is to make it behave like an object, the dplyr style.

First: *col* is called as a string

```
col_summary <- function(df, col) {
  ggplot(df) +
    geom_bar(aes(x = .data[[col]])) +
    coord_flip()
}

col_summary(mpg, "drv")
```

Second: *col* is called like an object

```
col_summary <- function(df, col) {
  ggplot(df) +
    geom_bar(aes(x = {{ col }})) +
    coord_flip()
}

col_summary(mpg, drv)
```

Sources

- tidy eval book
- programming with dplyr
- ggplot2 functions
- rlang cheatsheet