Introduction to `rofi`

Getting started

`rofi` is a R package to help structure and streamline "Opportunities for improvement" (OFI) research projects using data from the Karolinska University Hospital trauma registry and trauma care quality. `rofi` is quite opinionated about directory structure and coding style that should make it easier for those just starting out to do just that, get started. database.

Let's start writing some `rofi` code of your own.

In your working directory, there should be a file called `main.R`. If there is no such file, please create it, for example using `file.create("./main.R")`. This file structures the rest of the code. Think of it as a table of contents, a file that anyone should be able to read and quickly get an idea about the steps required to get to the end result.

There might be some code in your `main.R` already, but consider revising it to look like this:

```
## Load packages
library(rofi)

## Import data
imported.data <- import_data()

## Merge data
data <- merge_data(imported.data)

## Prepare data
prepared.data <- prepare_data(data)
```

Now, `import_data`, `merge_data` and `prepare_data` are all functions that are specific to the `rofi` package. They do things with the data. If you want to understand what a particular function does, you can use `?rofi::merge_data`. This will open the documentation for the `merge_data` function.

We believe that R code is best expressed in functions. A function is basically a reusable piece of code, that uses some input (provided in the form of arguments) to produce some output. Often, functions also work in isolation and without side effects, meaning that whatever happens inside a function does not affect the environment outside of that function.

We think that there should be one function per file (with few exceptions, for example if you want to collect many very short and related functions in one file). This style makes it easier for several persons to work together on the same codebase.

So, let's create your first function. At some point you probably want to select just a few of all the variables in the complete dataset, because you do not need all of them for your particular project. So:

```
create_function("Select variables")
```

This should create a file called `select_variables.R` in the directory `functions` and open that file in your editor. It should look something like this:

```
#' Select Variables
#'
```

```r
#' Write a short description of your function here
#' @param x Describe argument x here
#' @export
select_variables <- function(x) {
    ## Replace with the contents of your function
    y <- x + 1
    return (y)
}
```

Now. Replace "Write a short description of your function here" with a short description of what your function does (maybe you wonder what the weird "#"' and "@param" and so on comes from. Those are from `roxygen`, a way of documenting R code for packages. See https://r-pkgs.org/man.html for details). You could for example write "Select only the variables I need for my analysis". The file would then look like this:

```r
#' Select Variables
#'
#' Select only the variables I need for my analysis
#' @param x Describe argument x here
#' @export
prepare_data <- function(x) {
    ## Replace with the contents of your function
    y <- x + 1
    return (y)
}'
```

The "@param" denotes documentation of a single argument, in this case called `x`. In this case `x` is a very uninformative argument name, so let's rename it to something more informative and intuitive, for example `dataset`. Let's also describe this argument, for example "The complete dataset", and replace `x` in the function definition with `dataset`:

```r
#' Select Variables
#'
#' Select only the variables I need for my analysis
#' @param dataset The complete dataset.
#' @export
prepare_data <- function(dataset) {
    ## Replace with the contents of your function
    y <- x + 1
    return (y)
}
```

Finally, let's do something useful with this function, namely selecting only the variables that we're going to use and return this smaller dataset:

```r
#' Select Variables
#'
#' Select only the variables I need for my analysis
#' @param dataset The complete dataset.
#' @export
prepare_data <- function(dataset) {
    variables.to.include <- c("pt_Gender", "pt_age_yrs", "ed_sbp_value", "ed_rr_value", "ed_gcs_sum", "
    prepared.dataset <- dataset[, variables.to.include]
    return (prepared.dataset)
}
```

Now that we have this function, let's go ahead and add it to our main script. Open the file `main.R` and add this function to where you want to use it. For example just after importing your data:

```r
## Load packages
library(rofi)

## Import data
imported.data <- import_data()

## Merge data
data <- merge_data(imported.data)

## Prepare data
prepared.data <- prepare_data(data)

## Select only variables required for the analysis
selected.data <- select_variables(prepared.data)
```

Note that if you give your function explicit names the comments are quite unnecessary, the function names themselves indicate what the functions do, and your main script becomes a roadmap or analysis plan showing what happens when, but with the bulk of the code contained in functions, kept in separate files.

Next steps

Congratulations, you should now be ready to write some project code of your own! To see all vignettes in this package, run:

```r
browseVignettes("rofi")
```

And to open a specific vignette, run:

```r
vignette("introduction", "rofi")
```