January 7, 2015

0.0.1 rmdHeader

`rmdHeader` generates the yaml metadata header for Rmd files as a character string to be inserted at the top of a file. It has default arguments specific to my own projects but can be changed. The output from this function is passed directly to `genRmd` below.

```
rmdHeader <- function(title = "INSERT_TITLE_HERE", author = "Matthew Leonawicz",
    theme = "united", highlight = "zenburn", toc = FALSE, keep.md = TRUE, ioslides = FALSE,
    include.pdf = FALSE) {
    if (toc)
        toc <- "true" else toc <- "false"
    if (keep.md)
        keep.md <- "true" else keep.md <- "false"
    if (ioslides)
        hdoc <- "ioslides_presentation" else hdoc <- "html_document"
    rmd.header <- "---\n"
    if (!is.null(title))
        rmd.header <- paste0(rmd.header, "title: ", title, "\n")
    if (!is.null(author))
        rmd.header <- paste0(rmd.header, "author: ", author, "\n")
    rmd.header <- paste0(rmd.header, "output:\n  ", hdoc, ":\n    toc: ", toc,
        "\n    theme: ", theme, "\n    highlight: ", highlight, "\n    keep_md: ",
        keep.md, "\n")
    if (ioslides)
        rmd.header <- paste0(rmd.header, "    widescreen: true\n")
    if (include.pdf)
        rmd.header <- paste0(rmd.header, "  pdf_document:\n    toc: ", toc,
            "\n    highlight: ", highlight, "\n")
    rmd.header <- paste0(rmd.header, "---\n")
    rmd.header
}
```

0.0.2 rmdknitrSetup

`rmdknitrSetup` generates the `knitr` global options setup code cunk for Rmd files as a character string to be inserted at the top of a file following the yaml header. The only option at this time is the ability to include or exclude a source reference to a project-related code flow diagram **R** script via `include.sankey`. The output from this function is passed directly to `genRmd` below.

```
rmdknitrSetup <- function(file, include.sankey = TRUE) {
    x <- paste0("\n```{r knitr_setup, echo=FALSE}\nopts_chunk$set(cache=FALSE, eval=FALSE, tidy=TRUE, me
    if (include.sankey)
        x <- paste0(x, "read_chunk(\"../../code/proj_sankey.R\")\n")
    x <- paste0(x, "read_chunk(\"../../code/", file, "\")\n```\n")
    x
```

```
}
```

0.0.3 genRmd

`genRmd` works on existing projects. It checks for existing **R** scripts. If no **R** files exist in the project's `code` directory, the function will abort. Otherwise it will generate Rmd template files for each of the **R** scripts it finds.

With `replace=TRUE` any existing Rmd files are regenerated with the provided template - be careful! With `replace=FALSE` (default) Rmd files are generated only for **R** scripts which do not yet have corresponding Rmd files. If `update.header=TRUE`, `replace` is ignored, and only existing Rmd files are regenerated, in this case strictly updating the yaml metadata header at the top of each Rmd file without altering any other Rmd content/documentation.

The Rmd files are placed in the `/docs/Rmd` directory. This function assumes this project directory exists.

```
genRmd <- function(path, replace = FALSE, header = rmdHeader(), knitrSetupChunk = rmdknitrSetup(),
    update.header = FALSE, ...) {
    stopifnot(is.character(path))
    files <- list.files(path, pattern = ".R$", full = TRUE)
    stopifnot(length(files) > 0)
    rmd <- gsub(".R", ".Rmd", basename(files))
    rmd <- file.path(dirname(path), "docs/Rmd", rmd)
    if (!(replace | update.header))
        rmd <- rmd[!sapply(rmd, file.exists)]
    if (update.header)
        rmd <- rmd[sapply(rmd, file.exists)]
    stopifnot(length(rmd) > 0)

    sinkRmd <- function(x, ...) {
        y1 <- header
        y2 <- kniterSetupChunk
        y3 <- list(...)$rmd.template
        if (is.null(y1))
            y1 <- rmd.header
        if (is.null(y2))
            y2 <- rmd.knitr.setup(gsub(".Rmd", ".R", basename(x)))
        if (is.null(y3))
            y3 <- rmd.template
        sink(x)
        sapply(c(y1, y2, y3), cat)
        sink()
    }

    swapHeader <- function(x) {
        l <- readLines(x)
        ind <- which(l == "---")
        l <- l[(ind[2] + 1):length(l)]
        l <- paste0(l, "\n")
        sink(x)
        sapply(c(header, l), cat)
        sink()
    }

    if (update.header) {
        sapply(rmd, swapHeader, ...)
        cat("yaml header updated for each .Rmd file.\n")
    } else {
        sapply(rmd, sinkRmd, ...)
        cat(".Rmd files created for each .R file.\n")
    }
}
```

0.0.4 chunkNames

`chunkNames` can be used in two ways. It can return a list with length equal to the number of **R** files, where each list element is a vector of **R** code chunk names found in each **R** script.

Alternatively, with `append.new=TRUE`, this list has each vector matched element-wise against chunk names found in existing Rmd files. If no Rmd files have yet been generated, the function will abort. Otherwise, for the Rmd files which do exist (and this may correspond to a subset of the **R** files), these Rmd files are appended with a list of code chunk names found in the current corresponding **R** files which have not yet been integrated into the current state of the Rmd files. This fascilitates updating of Rmd documentation when it falls behind scripts which have been updated.

```
chunkNames <- function(path, rChunkID = "# @knitr", rmdChunkID = "```{r", append.new = FALSE) {
    files <- list.files(path, pattern = ".R$", full = TRUE)
    stopifnot(length(files) > 0)
    l1 <- lapply(files, readLines)
    l1 <- rapply(l1, function(x) x[substr(x, 1, nchar(rChunkID)) == rChunkID],
        how = "replace")
    l1 <- rapply(l1, function(x, p) gsub(paste0(p, " "), "", x), how = "replace",
        p = rChunkID)
    if (!append.new)
        return(l1)

    appendRmd <- function(x, rmd.files, rChunks, rmdChunks, ID) {
        r1 <- rmdChunks[[x]]
        r2 <- rChunks[[x]]
        r.new <- r2[!(r2 %in% r1)]
        if (length(r.new)) {
            r.new <- paste0(ID, " ", r.new, "}\n```\n", collapse = "")  # Hard coded brace and backticks
            sink(rmd.files[x], append = TRUE)
            cat("\nNEW_CODE_CHUNKS\n")
            cat(r.new)
            sink()
            paste(basename(rmd.files[x]), "appended with new chunk names from .R file")
        } else paste("No new chunk names appended to", basename(rmd.files[x]))
    }

    rmd <- gsub(".R", ".Rmd", basename(files))
    rmd <- file.path(dirname(path), "docs/Rmd", rmd)
    rmd <- rmd[sapply(rmd, file.exists)]
    stopifnot(length(rmd) > 0)  # Rmd files must exist
    files.ind <- match(gsub(".Rmd", "", basename(rmd)), gsub(".R", "", basename(files)))  # Rmd exist fo
    l2 <- lapply(rmd, readLines)
    l2 <- rapply(l2, function(x) x[substr(x, 1, nchar(rmdChunkID)) == rmdChunkID],
        how = "replace")
    l2 <- rapply(l2, function(x, p) gsub(paste0(p, " "), "", x), how = "replace",
```

```
        p = gsub("\\{", "\\\\{", rmdChunkID))
    l2 <- rapply(l2, function(x) gsub("}", "", sapply(strsplit(x, ","), "[[",
        1)), how = "replace")
    sapply(1:length(rmd), appendRmd, rmd.files = rmd, rChunks = l1[files.ind],
        rmdChunks = l2, ID = rmdChunkID)
}
```

Regarding the creation and updating of Rmd files, `rpm` simply assumes that there will be one **R** Markdown file corresponding to one **R** script. This is not always the case for a given project, but again, the purpose is to generate basic templates. Unnecessary files can always be deleted later, or edits made such that one **R** Markdown file reads multiple **R** scripts, as is the case with the Rmd file used to generate this document.