December 24, 2014

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, `projman` 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.

0.0.5 convertDocs

`convertDocs` converts between Rmd and Rnw files. The project's `docs/Rmd` or `docs/Rnw` directory is specified. Any files of the same type as the directory are converted to the other type and saved to the other directory. The input files are not removed.

This function speeds up the process of duplicating files, e.g., when wanting to make PDFs from Rnw files when only Rmd files exist. This is almost exclusively what I use this function for. On less frequent occasions I have used it in the other direction when I have Rnw files which were once used to make PDFs but later I decide to put them on the web as a web page and not as a link to a PDF.

The user still makes specific changes by hand, for example, any required changes to `knitr` code chunk options that must differ for PDF output vs. html output. The primary benefit is in not having to fuss with large amounts of standard substitutions which can be automated, such as swapping code chunk enclosure styles and common file metadata.

```
convertDocs <- function(path, rmdChunkID = c("```{r", "}", "```"), rnwChunkID = c("<<",
    ">>=", "@"), emphasis = "replace", overwrite = FALSE, ...) {
    stopifnot(is.character(path))
    type <- basename(path)
    rmd.files <- list.files(path, pattern = ".Rmd$", full = TRUE)
    rnw.files <- list.files(path, pattern = ".Rnw$", full = TRUE)
    dots <- list(...)
    if (rmdChunkID[1] == "```{r")
        rmdChunkID[1] <- paste0(rmdChunkID[1], " ")
    gsbraces <- function(txt) gsub("\\{", "\\\\{", txt)
    if (type == "Rmd") {
        stopifnot(length(rmd.files) > 0)
        outDir <- file.path(dirname(path), "Rnw")
        if (is.null(doc.class <- dots$doc.class))
            doc.class <- "article"
        if (is.null(doc.packages <- dots$doc.packages))
            doc.packages <- "geometry"
        doc.class.string <- paste0("\\documentclass{", doc.class, "}")
        doc.packages.string <- paste0(sapply(doc.packages, function(x) paste0("\\usepackage{",
            x, "}")), collapse = "\n")
        if ("geometry" %in% doc.packages)
            doc.packages.string <- c(doc.packages.string, "\\geometry{verbose, tmargin=2.5cm, bmargin=2.
        header.rnw <- c(doc.class.string, doc.packages.string, "\\begin{document}\n")  #,
        # paste0('<<highlight, echo=FALSE>>=\nknit_theme$set(knit_theme$get('',
        # theme, ''))\n@\n'))
    } else if (type == "Rnw") {
        stopifnot(length(rnw.files) > 0)
```

```
    outDir <- file.path(dirname(path), "Rmd")
} else stop("path must end in 'Rmd' or 'Rnw'.")

swapHeadings <- function(from, to, x) {
    nc <- nchar(x)
    ind <- which(substr(x, 1, 8) == "\\section" | substr(x, 1, 4) == "\\sub")
    if (!length(ind)) {
        # assume Rmd file
        ind <- which(substr(x, 1, 1) == "#")
        ind.n <- rep(1, length(ind))
        for (i in 2:6) {
            ind.tmp <- which(substr(x[ind], 1, i) == substr("######", 1,
              i))
            if (length(ind.tmp))
              ind.n[ind.tmp] <- ind.n[ind.tmp] + 1 else break
        }
        for (i in 1:length(ind)) {
            n <- ind.n[i]
            input <- paste0(substr("######", 1, n), " ")
            h <- x[ind[i]]
            h <- gsub("\\*", "_", h)  # Switch any markdown boldface asterisks in headings to double
            heading <- gsub("\n", "", substr(h, n + 2, nc[ind[i]]))
            # h <- gsub(input, '', h)
            if (n <= 2)
              subs <- "\\" else if (n == 3)
              subs <- "\\sub" else if (n == 4)
              subs <- "\\subsub" else if (n >= 5)
              subs <- "\\subsubsub"
            output <- paste0("\\", subs, "section{", heading, "}\n")
            x[ind[i]] <- gsub(h, output, h)
        }
    } else {
        # assume Rnw file
        ind <- which(substr(x, 1, 8) == "\\section")
        if (length(ind)) {
            for (i in 1:length(ind)) {
              h <- x[ind[i]]
              heading <- paste0("## ", substr(h, 10, nchar(h) - 2))
              x[ind[i]] <- gsub(gsbraces(h), heading, h)
            }
        }
        ind <- which(substr(x, 1, 4) == "\\sub")
        if (length(ind)) {
            for (i in 1:length(ind)) {
              h <- x[ind[i]]
              z <- substr(h, 2, 10)
              if (z == "subsubsub") {
                p <- "##### "
                n <- 18
              } else if (substr(z, 1, 6) == "subsub") {
                p <- "#### "
                n <- 15
              } else if (substr(z, 1, 3) == "sub") {
```

```
                p <- "### "
                n <- 12
              }
              heading <- paste0(p, substr(h, n, nchar(h) - 2))
              x[ind[i]] <- gsub(gsbraces(h), heading, h)
            }
        }
    }
    x
}

swapChunks <- function(from, to, x) {
    nc <- nchar(x)
    chunk.start.open <- substr(x, 1, nchar(from[1])) == from[1]
    chunk.start.close <- substr(x, nc - 1 - nchar(from[2]) + 1, nc - 1) ==
        from[2]
    chunk.start <- which(chunk.start.open & chunk.start.close)
    chunk.end <- which(substr(x, 1, nchar(from[3])) == from[3] & nc == nchar(from[3]) +
        1)
    x[chunk.start] <- gsub(from[2], to[2], gsub(gsbraces(from[1]), gsbraces(to[1]),
        x[chunk.start]))
    x[chunk.end] <- gsub(from[3], to[3], x[chunk.end])
    chunklines <- as.numeric(unlist(mapply(seq, chunk.start, chunk.end)))
    list(x, chunklines)
}

# I know I use '**' strictly for bold font in Rmd files.  For now, this
# function assumes: 1. The only emphasis in a doc is boldface or typewriter.
# 2. These instances are always preceded by a space, a carriage return, or
# an open bracket, 3. and followed by a space, period, comma, or closing
# bracket.
swapEmphasis <- function(x, emphasis = "remove", pat.remove = c("`", "\\*\\*",
    "__"), pat.replace = pat.remove, replacement = c("\\\\texttt\\{", "\\\\textbf\\{",
    "\\\\textbf\\{", "\\}", "\\}", "\\}")) {

    stopifnot(emphasis %in% c("remove", "replace"))
    n <- length(pat.replace)
    rep1 <- replacement[1:n]
    rep2 <- replacement[(n + 1):(2 * n)]
    prefix <- c(" ", "^", "\\{", "\\(")
    suffix <- c(" ", ",", "-", "\n", "\\.", "\\}", "\\)")
    n.p <- length(prefix)
    n.s <- length(suffix)
    pat.replace <- c(paste0(rep(prefix, n), rep(pat.replace, each = n.p)),
        paste0(rep(pat.replace, each = n.s), rep(suffix, n)))
    replacement <- c(paste0(rep(gsub("\\^", "", prefix), n), rep(rep1, each = n.p)),
        paste0(rep(rep2, each = n.s), rep(suffix, n)))
    if (emphasis == "remove")
        for (k in 1:length(pat.remove)) x <- sapply(x, function(v, p, r) gsub(p,
            r, v), p = pat.remove[k], r = "")
    if (emphasis == "replace")
        for (k in 1:length(pat.replace)) x <- sapply(x, function(v, p, r) gsub(p,
            r, v), p = pat.replace[k], r = replacement[k])
    x
```

```
}

swap <- function(file, header = NULL, outDir, ...) {
    title <- list(...)$title
    author <- list(...)$author
    highlight <- list(...)$highlight
    ext <- tail(strsplit(file, "\\.")[[1]], 1)
    l <- readLines(file)
    l <- l[substr(l, 1, 7) != "<style>"]  # Strip any html style lines
    if (ext == "Rmd") {
        hl.default <- "solarized-light"
        out.ext <- "Rnw"
        h.ind <- 1:which(l == "---")[2]
        h <- l[h.ind]
        t.ind <- which(substr(h, 1, 7) == "title: ")
        a.ind <- which(substr(h, 1, 8) == "author: ")
        highlight.ind <- which(substr(h, 1, 11) == "highlight: ")
        if (is.null(title) & length(t.ind))
            title <- substr(h[t.ind], 8, nchar(h[t.ind])) else if (is.null(title))
            title <- ""
        if (is.null(author) & length(a.ind))
            author <- substr(h[a.ind], 9, nchar(h[a.ind])) else if (is.null(author))
            author <- ""
        if (is.null(highlight) & length(highlight.ind))
            highlight <- substr(h[highlight.ind], 12, nchar(h[highlight.ind])) else if (is.null(high
            highlight <- hl.default else if (!(highlight %in% knit_theme$get()))
            highlight <- hl.default
        if (!is.null(title))
            header <- c(header, paste0("\\title{", title, "}"))
        if (!is.null(author))
            header <- c(header, paste0("\\author{", author, "}"))
        if (!is.null(title))
            header <- c(header, "\\maketitle\n")
        header <- c(header, paste0("<<highlight, echo=FALSE>>=\nknit_theme$set(knit_theme$get('",
            highlight, "'))\n@\n"))
    } else if (ext == "Rnw") {
        hl.default <- "tango"
        out.ext <- "Rmd"
        begin.doc <- which(l == "\\begin{document}")
        make.title <- which(l == "\\maketitle")
        if (length(make.title))
            h.ind <- 1:make.title else h.ind <- 1:begin.doc
        h <- l[h.ind]
        t.ind <- which(substr(h, 1, 6) == "\\title")
        a.ind <- which(substr(h, 1, 7) == "\\author")
        highlight.ind <- which(substr(l, 1, 11) == "<<highlight")
        if (is.null(title) & length(t.ind))
            title <- substr(h[t.ind], 8, nchar(h[t.ind]) - 1)
        if (is.null(author) & length(a.ind))
            author <- substr(h[a.ind], 9, nchar(h[a.ind]) - 1)
        if (length(highlight.ind)) {
            l1 <- l[highlight.ind + 1]
            h1 <- substr(l1, nchar("knit_theme$set(knit_theme$get('") +
```

```
                    1, nchar(l1) - nchar("'))\n"))
                if (!(h1 %in% knit_theme$get()))
                    h1 <- hl.default
            }
            if (is.null(highlight) & length(highlight.ind))
                highlight <- h1 else if (is.null(highlight))
                highlight <- hl.default else if (!(highlight %in% knit_theme$get()))
                highlight <- hl.default
            header <- rmdHeader(title = title, author = author, highlight = highlight)
        }
        header <- paste0(header, collapse = "\n")
        l <- paste0(l[-h.ind], "\n")
        if (ext == "Rmd") {
            from <- rmdChunkID
            to <- rnwChunkID
        }
        if (ext == "Rnw") {
            from <- rnwChunkID
            to <- rmdChunkID
        }
        l <- swapHeadings(from = from, to = to, x = l)
        chunks <- swapChunks(from = from, to = to, x = l)
        l <- chunks[[1]]
        if (ext == "Rmd")
            l <- swapEmphasis(x = l, emphasis = emphasis)
        if (ext == "Rmd")
            l[-chunks[[2]]] <- sapply(l[-chunks[[2]]], function(v, p, r) gsub(p,
                r, v), p = "_", r = "\\\\_")
        l <- c(header, l, "\n\\end{document}\n")
        outfile <- file.path(outDir, gsub(paste0("\\.", ext), paste0("\\.",
            out.ext), basename(file)))
        if (overwrite || !file.exists(outfile)) {
            sink(outfile)
            sapply(l, cat)
            sink()
            print(paste("Writing", outfile))
        }
    }

    if (type == "Rmd") {
        sapply(rmd.files, swap, header = header.rnw, outDir = outDir, ...)
        cat(".Rmd to .Rnw file conversion complete.\n")
    } else {
        sapply(rnw.files, swap, header = NULL, outDir = outDir, ...)
        cat(".Rnw to .Rmd file conversion complete.\n")
    }
}
```

0.0.6 moveDocs

`moveDocs` relocates files by renaming with a new file path. Specifically, it scans for md and html files in the `docs/Rmd` directory and/or pdf files in the `docs/Rnw` directory. If such files are found in the respective locations, they are moved to `docs/md`, `docs/html`, and `docs/pdf`, respectively.

The intent is to clean up the Rmd and Rnw directories after `knitr` has been used to knit documents in place. I do this because I have more success knitting documents with the confluence of `RStudio`, `rmarkdown`, `knitr`, `pandoc`, and `LaTeX` when the knitting occurs all within the directory of the originating files. The process is more prone to throwing errors when trying to specify alternate locations for outputs.

`moveDocs` makes a nominal effort to replace a possible relative path with a full file path before proceeding, if the former is supplied. Default arguments include `move=TRUE` which will call `file.rename` and `copy=FALSE` which, if `TRUE` (and `move=FALSE`), will alternatively call `file.copy`. If both are `TRUE`, any files found are moved.

This function will always overwrite any existing file versions previously moved to the output directories, by way of `file.rename`. To keep the behavior consistent, when `move=FALSE` and `copy=TRUE`, `file.copy` always executes with its argument, `overwrite=TRUE`. This should never cause problems because in the context I intend for this function, the types of files being moved or copied from `docs/Rmd` and `docs/Rnw` are never used as inputs to other files, functions, or processes, nor are they meant to be edited by hand after being generated.

If there are LaTeX-associated files present (.TeX, .aux, and .txt files with the same file names as local pdf files.), these files will be removed if `remove.latex=TRUE` (default). If `FALSE`, the default `latexDir="LaTeX"` means that these files will be moved to the `docs/LaTeX` directory rather than deleted. If this directory does not exist, it will be created. An alternate location can be specified, such as "pdf" if you want to keep these files with the related pdf files after those are moved by `moveDocs` as well to `docs/pdf`.

```
moveDocs <- function(path.docs, type = c("md", "html", "pdf"), move = TRUE,
    copy = FALSE, remove.latex = TRUE, latexDir = "latex") {
    if (any(!(type %in% c("md", "html", "pdf"))))
        stop("type must be among 'md', 'html', and 'pdf'")
    stopifnot(move | copy)
    if (path.docs == "." | path.docs == "./")
        path.docs <- getwd()
    if (strsplit(path.docs, "/")[[1]][1] == "..") {
        tmp <- strsplit(path.docs, "/")[[1]][-1]
        if (length(tmp))
            path.docs <- file.path(getwd(), paste0(tmp, collapse = "/")) else stop("Check path.docs argu
    }
    for (i in 1:length(type)) {
        if (type[i] == "pdf")
            origin <- "Rnw" else origin <- "Rmd"
        path.i <- file.path(path.docs, origin)
        infiles <- list.files(path.i, pattern = paste0("\\.", type[i], "$"),
            full = TRUE)
        if (type[i] == "pdf") {
            extensions <- c("tex", "aux", "log")
            all.pdfs <- basename(list.files(path.docs, pattern = ".pdf$", full = T,
                recursive = T))
            pat <- paste0("^", rep(gsub("pdf", "", all.pdfs), length(extensions)),
                rep(extensions, each = length(all.pdfs)), "$")
            latex.files <- sapply(1:length(pat), function(p, path, pat) list.files(path,
                pattern = pat[p], full = TRUE), path = path.i, pat = pat)
            if (!is.list(latex.files)) {
                if (remove.latex) {
                  unlink(latex.files)
                } else {
                  dir.create(file.path(path.docs, latexDir), showWarnings = FALSE,
                    recursive = TRUE)
                  file.rename(latex.files, file.path(path.docs, latexDir, basename(latex.files)))
                }
```

```
            }
        }
        if (length(infiles)) {
            infiles <- infiles[basename(dirname(infiles)) == origin]
            if (length(infiles)) {
                if (type[i] == "html") {
                  html.dirs <- gsub("\\.html", "_files", infiles)
                  dirs <- list.dirs(path.i, recursive = FALSE)
                  ind <- which(dirs %in% html.dirs)
                  if (length(ind)) {
                    html.dirs <- dirs[ind]
                    html.dirs.recur <- list.dirs(html.dirs)
                    for (p in 1:length(html.dirs.recur)) dir.create(gsub("/Rmd",
                      "/html", html.dirs.recur[p]), recursive = TRUE, showWarnings = FALSE)
                    subfiles <- unique(unlist(lapply(1:length(html.dirs.recur),
                      function(p, path) list.files(path[p], full = TRUE), path = html.dirs.recur)))
                    subfiles <- subfiles[!(subfiles %in% html.dirs.recur)]
                    file.copy(subfiles, gsub("/Rmd", "/html", subfiles), overwrite = TRUE)
                    if (move)
                      unlink(html.dirs, recursive = TRUE)
                  }
                }
                outfiles <- file.path(path.docs, type[i], basename(infiles))
                if (move)
                  file.rename(infiles, outfiles) else if (copy)
                  file.copy(infiles, outfiles, overwrite = TRUE)
            }
        }
    }
}
```