Project Management

Matthew Leonawicz

January 7, 2015

1 Example usage

This is how `rpm` functions can be used to create and manipulate a project, using the `rpm` project itself as an example. The code below is not intended to be run in full directly, but serves as a guide.

1.1 Dynamic report generation

The script, `pm.R`, is used to compile web sites and reports in various formats based on project documentation, namely Rmd files. Using this project management project as an example, markdown and html files are generated for existing Rmd files. There is also optional conversion from Rmd to Rnw and subsequent PDF generation.

1.2 Github user website

Although not a part of the `pm.R` scripts associated with other **R** projects, this example also includes creation of my Github user website, `leonawicz.github.io`, as part of overall project management. This is in addition to the `rpm`-specific project website.

2 R code

2.1 Create a project

Note that I use my own default path for storing a project when creating a new project. See the `rpm` [default objects](objects.html "default objects") and [creating a new project](func_new.html "new project") for more details.

```
source("C:/github/ProjectManagement/code/rpm.R")  # Eventually load rpm package instead
proj.name <- "ProjectManagement"  # Project name
proj.location <- matt.proj.path  # Use default file location

docDir <- c("Rmd/include", "md", "html", "Rnw", "pdf", "timeline")
newProject(proj.name, proj.location, docs.dirs = docDir, overwrite = T)  # create a new project

rfile.path <- file.path(proj.location, proj.name, "code")  # path to R scripts
docs.path <- file.path(proj.location, proj.name, "docs")
rmd.path <- file.path(docs.path, "Rmd")

# generate Rmd files from existing R scripts using default yaml front-matter
genRmd(path = rfile.path, header = rmdHeader())
```

2.2 Update a project

Functions can be used to create, read, or update. See [Rmd-related functions](func_rmd.html "Rmd-related functions").

```
# update yaml front-matter only
genRmd(path = rfile.path, header = rmdHeader(), knitrSetupChunk = rmdknitrSetup(),
    update.header = TRUE)

# obtain knitr code chunk names in existing R scripts
chunkNames(path = file.path(proj.location, proj.name, "code"))

# append new knitr code chunk names found in existing R scripts to any Rmd
# files which are outdated
chunkNames(path = file.path(proj.location, proj.name, "code"), append.new = TRUE)
```

2.3 Prepare a project website

With some additional project-specific setup, files can be generated which will assist in creating a project website. See [website-related functions](func_website.html "website-related functions").

```
# Setup for generating a project website
index.url <- file.path(rmd.path, "proj_intro.html")  # temporary
file.copy(index.url, file.path(rmd.path, "index.html"))

proj.title <- "Project Management"
proj.menu <- c("rpm", "R Code", "All Projects")

proj.submenu <- list(c("About rpm", "Introduction", "Related items", "Example usage"),
    c("Default objects", "divider", "Functions", "Start a new project", "Working with Rmd files",
        "Document conversion", "Organize documents", "Make a project website",
        "Github user website"), c("Projects diagram", "divider", "About", "Other"))

proj.files <- list(c("header", "proj_intro.html", "code_sankey.html", "pm.html"),
    c("objects.html", "divider", "header", "func_new.html", "func_rmd.html",
        "func_convert.html", "func_organize.html", "func_website.html", "func_user_website.html"),
    c("proj_sankey.html", "divider", "proj_intro.html", "proj_intro.html"))

user <- "leonawicz"
proj.github <- file.path("https://github.com", user, proj.name)

# generate navigation bar html file common to all pages
genNavbar(htmlfile = file.path(proj.location, proj.name, "docs/Rmd/include/navbar.html"),
    title = proj.title, menu = proj.menu, submenus = proj.submenu, files = proj.files,
    title.url = "index.html", home.url = "index.html", site.url = proj.github,
    include.home = FALSE)

# generate _output.yaml file Note that external libraries are expected,
# stored in the 'libs' directory below
yaml.out <- file.path(proj.location, proj.name, "docs/Rmd/_output.yaml")
libs <- "libs"
common.header <- "include/in_header.html"
genOutyaml(file = yaml.out, lib = libs, header = common.header, before_body = "include/navbar.html")
```

2.4 Knit documents

Both Rmd and Rnw files can be knitted to various formats. Rmd and Rnw files can also be converted back and forth, with notable limitations. Files can be reorganized after knitting.

```
library(rmarkdown)
library(knitr)
setwd(rmd.path)

# R scripts files.r <- list.files('../../code', pattern='.R$', full=T)

# Rmd files
files.Rmd <- list.files(pattern = ".Rmd$", full = T)

# potential non-Rmd directories for writing various output files outtype <-
# file.path(dirname(getwd()), list.dirs('../', full=F, recursive=F)) outtype
# <- outtype[basename(outtype)!='Rmd']
```

```
# write all yaml front-matter-specified outputs to Rmd directory for all Rmd
# files
lapply(files.Rmd, render, output_format = "all")
moveDocs(path.docs = docs.path)

# if also making PDFs for a project, speed up the Rmd to Rnw file
# conversion/duplication
rnw.path <- file.path(docs.path, "Rnw")
setwd(rnw.path)
# themes <- knit_theme$get()
highlight <- "solarized-dark"
convertDocs(path = rmd.path, emphasis = "replace", overwrite = TRUE, highlight = highlight)  # Take care
lapply(list.files(pattern = ".Rnw$"), knit2pdf)
moveDocs(path.docs = docs.path, type = "pdf", remove.latex = FALSE)
```

2.5 Prepare a Github user website

As mentioned, the same functions can be applied to the generation of a Github user account website just as they are used to build individual project sites. I am currently using two somewhat incompatible bootstrap CSS themes for my user site vs. my project sites. I have made some ugly "generalizations" to the code which generates container elements for each in order to accommodate this frustrating desire. This is primarily an issue with differences in navbar construction.

I have only tested the code, and namely the navbar, with two themes, `united` and `cyborg`, and used these to generalize some of the code. I suspect all the other common Bootswatch themes will work once I make them permissible.

```
# Assuming project and app repos exist and are properly prepared
user <- "leonawicz"
user.site <- paste0(user, ".github.io")
mainDir <- file.path(proj.location, user.site)
setwd(file.path(mainDir, "assets"))

# create projects container html file
genPanelDiv(outDir = getwd(), type = "projects", main = "Projects", github.user = user,
    col = "primary")
```

```
# create Shiny apps container html file
genPanelDiv(outDir = getwd(), type = "apps", main = "Shiny Apps", github.user = "ua-snap")
# create Data Visualizations master container html file
genPanelDiv(outDir = getwd(), type = "datavis", main = "Data Visualizations",
    github.user = user, col = "default")
# create all Gallery container html files
genPanelDiv(outDir = getwd(), type = "gallery", main = "Gallery", github.user = user,
    col = "default", img.loc = "small", lightbox = TRUE, include.buttons = FALSE,
    include.titles = FALSE)

# Specify libraries for html head 'assets' is first because it resides in
# the top-level directory where the web site html files reside.
scripts = c("assets/libs/jquery-1.11.0/jquery.min.js", "assets/js/bootstrap.min.js",
    "assets/js/bootswatch.js", "assets/js/lightbox.min.js")
styles <- c("cyborg/bootstrap.css", "assets/css/bootswatch.min.css", "assets/libs/font-awesome-4.1.0/css
    "assets/css/lightbox.css")
styles.args <- list("", list(media = "screen"), "", "")

# check
htmlHead(script.paths = scripts, stylesheet.paths = styles, stylesheet.args = styles.args)

# Add a background image
back.img <- "assets/img/frac23.jpg"
# check
htmlBodyTop(background.image = back.img)

github.url <- file.path("https://github.com", user, user.site)

# Prepare navbar
nb.menu <- c("Projects", "Apps", "Data Visualizations", "Test1")

sub.menu <- list(c("empty"), c("empty"), c("empty"), c("A title", "A page",
    "divider", "Another title", "Page 1", "Page 2"))

files.menu <- list(c("index.html#projects"), c("index.html#apps"), c("index.html#datavis"),
    c("header", "#", "divider", "header", "#", "#"))


# Create navbar.html
genNavbar(htmlfile = "navbar.html", title = user.site, menu = nb.menu, submenus = sub.menu,
    files = files.menu, theme = "cyborg", title.url = "index.html", home.url = "index.html",
    site.url = github.url, include.home = FALSE)

# check
htmlBottom()

# Specify div container elements to include in body
all.containers <- list.files(pattern = "_container.html$")
keep.main <- c("about", "updates", "projects", "apps", "data-visualizations")
keep.main.ind <- match(keep.main, sapply(strsplit(all.containers, "_"), "[[",
    1))
main.containers <- all.containers[keep.main.ind]

gallery.containers <- list.files(pattern = "^gallery.*.html$")
```

```
# Create web pages
genUserPage(file = file.path(proj.location, user.site, "index.html"), navbar = "navbar.html",
    containers = main.containers, script.paths = scripts, stylesheet.paths = styles,
    stylesheet.args = styles.args, background.image = back.img)

files.out <- gsub("_", "-", gsub("_-_", "-", gallery.containers))
for (i in 1:length(gallery.containers)) {
    genUserPage(file = file.path(proj.location, user.site, files.out[i]), navbar = "navbar.html",
        containers = gallery.containers[i], script.paths = scripts, stylesheet.paths = styles,
        stylesheet.args = styles.args, background.image = back.img)
}
```