January 7, 2015

0.0.1 genAppDiv

`genAppDiv` generates an html file storing a container div element which organizes Shiny web applications. The function scans a directory of Shiny app subdirectories. This apps directory should be a local repository.

Specifically, `genAppDiv` looks for a named directory of image files. There should be one image per app, named exactly as the respective app directory is named. Only apps with corresponding images are built into the html container. If you wish to leave out, say, a developmental app from being linked to on you Github user website, do not include an image file for that app.

The container element includes an image link to each app's url as well as a link to the source code on Github. Although the function scans for images in directory inside a local repository, the images referenced in the output html are of course not local. They point to the same images stored on Github, hence why it is useful for the local directory of apps to be a Github repository. As an example, a repository may contain the directories, `app1`, `app2`, `app3`, and `images`.

This function will probably be removed in favor of the more general `genPanelDiv` function.

```
# Functions for Github user website
genAppDiv <- function(file = "C:/github/leonawicz.github.io/assets/apps_container.html",
    type = "apps", main = "Shiny Apps", apps.url = "http://shiny.snap.uaf.edu",
    github.url = "https://github.com/ua-snap/shiny-apps/tree/master", apps.dir = "C:/github/shiny-apps",
    img.loc = "_images/cropped", ...) {

    apps.img <- list.files(file.path(apps.dir, img.loc))
    apps <- sapply(strsplit(apps.img, "\\."), "[[", 1)
    x <- paste0("<div class=\"container\">\n  <div class=\"row\">\n    <div class=\"col-lg-12\">\n
        type, "\">", main, "</h3>\n      </div>\n    </div>\n  </div>\n  ")

    fillRow <- function(i, ...) {
        app <- apps[i]
        app.url <- file.path(apps.url, app)
        dots <- list(...)
        if (is.null(dots$col))
            col <- "warning" else col <- dots$col
        if (is.null(dots$panel.main))
            panel.main <- gsub("_", " ", app) else panel.main <- dots$panel.main
        if (length(panel.main) > 1)
            panel.main <- panel.main[i]
        x <- paste0("<div class=\"col-lg-4\">\n\t\t  <div class=\"bs-component\">\n\t\t\t<div class=\"pa
            col, "\">\n\t\t\t  <div class=\"panel-heading\"><h3 class=\"panel-title\">",
            panel.main, "</h3>\n\t\t\t  </div>\n\t\t\t  <div class=\"panel-body\"><a href=\"",
            app.url, "\" target=\"_blank\"><img src=\"", file.path(gsub("/tree/",
                "/raw/", github.url), img.loc, apps.img[i]), "\" alt=\"", apps[i],
            "\" width=100% height=200px></a><p></p>\n\t\t\t\t<div class=\"btn-group btn-group-justified\
            app.url, "\" target=\"_blank\" class=\"btn btn-success\">Launch</a>\n\t\t\t\t  <a href=\"",
            file.path(github.url, app), "\" target=\"_blank\" class=\"btn btn-info\">Github</a>\n\t\t\t\
    }
    n <- length(apps)
    seq1 <- seq(1, n, by = 3)
    y <- c()
    for (j in 1:length(seq1)) {
        ind <- seq1[j]:(seq1[j] + 2)
        ind <- ind[ind %in% 1:n]
        y <- c(y, paste0("<div class=\"row\">\n", paste0(sapply(ind, fillRow,
            ...), collapse = "\n"), "</div>\n"))
    }
    z <- "</div>\n"
    sink(file)
    sapply(c(x, y, z), cat)
    sink()
    cat("div container html created for Shiny Apps.\n")
}
```

0.0.2 genPanelDiv

`genPanelDiv` generates an html file storing a container div element which in its current state of development organizes two types of content: **R** projects and Shiny web applications.

The `type` argument can be one of `projects`, `apps`, `datavis`, or `gallery`. The purpose of the function is to generate an html file defining a container div element to display and reference either my **R** projects, my Shiny apps, or my example visualization galleries.

Projects

For projects, the function scans a directory of local repositories and takes any directories found to be the names of projects. There is an `exclude` argument for dropping any known directories that are to be avoided. My defaults are `exclude="leonawicz.github.io", "shiny-apps"` since the first is just a local repository for my Github user account web site and not a ”project” in the same sense of my other projects and the second is the local repository which is scanned by `genPanelDiv` when `type="apps"`.

Apps

For apps, the function scans a directory of Shiny app subdirectories. Unlike for projects, where `genPanelDiv` scans a directory of multiple local repositories, this apps directory should be a specific local repository. The apps contained within are not individual repositories. I have taken this approach for now simply because this is how my apps tend to be stored.

Specifically, the `genAppDiv` looks for a named directory of image files. There should be one image per app, named exactly as the respective app directory is named. Only apps with corresponding images are built into the html container. If you wish to leave out, say, a developmental app from being linked to on you Github user website, do not include an image file for that app.

The container element includes an image link to each app's url as well as a link to the source code on Github. Although the app scans for images in a local repository, the images referenced in the output html are of course not local. They point to the same images stored on Github, hence why it is useful for the local directory of apps to be a Github repository.

DataVis and Galleries

Whereas the first three types generate containers for the main Github user web page, I use `type="gallery"` to make a separate container html file of graphics for each panel occurring in my `datavis` container. These containers tend to be added to unique web pages. `datavis` is for highlighting a number of galleries whereas `gallery` is for the galleries' respective contents.

In order to use `type="datavis"` there must be a data visualization local repository. Mine is named `DataVisualizationExamples`, evident from the hardcoding currently in place within this function. Similar to when `type="apps"`, this repository includes a directory of images, in t his case one image for each gallery. Each image in this directory is named such that it identically matches another the name of a gallery images directory containing multiple images. As with `type="apps"`, gallery directories are only included if a corresponding thumbnail image in the images directory exists.

When `type="gallery"`, the behavior of `genPanelDiv` is most unique. For each gallery which exists, the function will make a unique html file with a gallery container element.

This function makes the more specific `genAppDiv` redundant and will likely replace it.

```r
genPanelDiv <- function(outDir, type = "projects", main = "Projects", github.user = "leonawicz",
    prjs.dir = "C:/github", exclude = c("leonawicz.github.io", "shiny-apps",
        "DataVisExamples", ".git", "_images"), img.loc = "_images/small", lightbox = FALSE,
    include.buttons = TRUE, include.titles = TRUE, ...) {

    stopifnot(github.user %in% c("leonawicz", "ua-snap"))
    if (type == "apps") {
        filename <- "apps_container.html"
        web.url <- "http://shiny.snap.uaf.edu"
        gh.url.tail <- "shiny-apps/tree/master"
        atts <- " target=\"_blank\""
        go.label <- "Launch"
        prjs.dir <- file.path(prjs.dir, "shiny-apps")
        prjs.img <- list.files(file.path(prjs.dir, img.loc))
        prjs <- sapply(strsplit(prjs.img, "\\."), "[[", 1)
    }
    if (type == "projects") {
        filename <- "projects_container.html"
        web.url <- paste0("http://", github.user, ".github.io")
        gh.url.tail <- ""
        atts <- ""
        go.label <- "Website"
        prjs <- list.dirs(prjs.dir, full = TRUE, recursive = FALSE)
        prjs <- prjs[!(basename(prjs) %in% exclude)]
        prjs.img <- sapply(1:length(prjs), function(i, a) list.files(file.path(a[i],
            "plots"), pattern = paste0("^_", basename(a)[i])), a = prjs)
        prjs <- basename(prjs)
    }
    if (type == "datavis") {
        filename <- "data-visualizations_container.html"
        web.url <- paste0("http://", github.user, ".github.io")
        gh.url.tail <- "DataVisExamples/tree/master"
        atts <- ""
        go.label <- "See More"
        prjs.dir <- file.path(prjs.dir, "DataVisExamples")
        prjs.img <- list.files(file.path(prjs.dir, img.loc))
        prjs <- sapply(strsplit(prjs.img, "\\."), "[[", 1)
    }
    if (type == "gallery") {
        web.url <- paste0("http://", github.user, ".github.io")
        gh.url.tail <- "DataVisExamples/tree/master"
        if (lightbox)
            atts1 <- " data-lightbox=\"ID\"" else atts1 <- ""
        go.label <- "Expand"
        prjs <- list.dirs(file.path(prjs.dir, "DataVisExamples"), full = T,
            recursive = F)
        prjs <- prjs[!(basename(prjs) %in% exclude)]
        prjs.img <- lapply(1:length(prjs), function(x, files, imgDir) list.files(path = file.path(files
            imgDir), recursive = FALSE), files = prjs, imgDir = img.loc)
        prjs <- basename(prjs)
        filename <- tolower(paste0("gallery-", gsub(" ", "-", gsub(" - ", " ",
            prjs)), ".html"))
    }
    gh.url <- file.path("https://github.com", github.user, gh.url.tail)

    fillRow <- function(i, ...) {
        prj <- panels[i]
        dots <- list(...)
        if (is.null(dots$col))
            col <- "warning" else col <- dots$col
        if (is.null(dots$panel.main))
            panel.main <- gsub(" - ", ": ", gsub("_", " ", prj)) else panel.main <- dots$panel.main
        if (length(panel.main) > 1)
            panel.main <- panel.main[i]
        if (type == "apps")
            img.src <- file.path(gsub("/tree/", "/raw/", gh.url), img.loc, prjs.img[i])
        if (type == "projects")
            img.src <- file.path(gh.url, prj, "raw/master/plots", prjs.img[i])
        if (type == "datavis")
            img.src <- file.path(gsub("/tree/", "/raw/", gh.url), img.loc, prjs.img[i])
        if (type != "gallery") {
            if (type == "datavis")
                pfx <- "gallery-" else pfx <- ""
            web.url <- file.path(web.url, tolower(paste0(pfx, gsub("_", "-",
                gsub("_-_", "-", prj)), ".html")))
        } else {
            prj <- prjs[p]
            img.src <- file.path(gsub("/tree/", "/raw/", gh.url), prjs[p], img.loc,
                panels[i])
            web.url <- file.path(gsub("/tree/", "/raw/", gh.url), prjs[p], panels[i])
            if (lightbox)
                atts <- gsub("ID", gsub(" - ", ": ", gsub("_", " ", prjs[p])),
                  atts1) else atts <- atts1
        }
        if (include.titles) {
            panel.title <- paste0("<div class=\"panel-heading\"><h3 class=\"panel-title\">",
                panel.main, "</h3>\n          </div>\n          ")
        } else panel.title <- ""
        if (include.buttons) {
            panel.buttons <- paste0("<div class=\"btn-group btn-group-justified\">\n\t\t\t<a href=\"",
                web.url, "\"", atts, " class=\"btn btn-success\">", go.label,
                "</a>\n\t\t\t<a href=\"", file.path(gh.url, prj), "\" class=\"btn btn-info\">Github</a>\
        } else panel.buttons <- ""
        x <- paste0("    <div class=\"col-lg-4\">\n      <div class=\"bs-component\">\n        <div clas
            col, "\">\n          ", panel.title, "<div class=\"panel-body\"><a href=\"",
            web.url, "\"", atts, "><img src=\"", img.src, "\" alt=\"", panel.main,
            "\" width=100% height=200px></a><p></p>\n          ", panel.buttons,
            "  </div>\n        </div>\n      </div>\n    </div>\n  ")
    }

    for (p in 1:length(filename)) {
        if (type == "gallery") {
            panels <- prjs.img[[p]]
```

```
            main <- gsub(" - ", ": ", gsub("_", " ", prjs[p]))
        } else panels <- prjs
        n <- length(panels)
        seq1 <- seq(1, n, by = 3)
        x <- paste0("<div class=\"container\">\n  <div class=\"row\">\n    <div class=\"col-lg-12\">\n
            type, "\">", main, "</h3>\n      </div>\n    </div>\n  </div>\n  ")
        y <- c()
        for (j in 1:length(seq1)) {
            ind <- seq1[j]:(seq1[j] + 2)
            ind <- ind[ind %in% 1:n]
            y <- c(y, paste0("<div class=\"row\">\n", paste0(sapply(ind, fillRow,
                ...), collapse = "\n"), "</div>\n"))
        }
        z <- "</div>\n"
        sink(file.path(outDir, filename[p]))
        sapply(c(x, y, z), cat)
        sink()
        cat("div container html file created.\n")
    }
}
```

0.0.3 htmlHead

`htmlHead` is useful for including javascript and CSS stylesheets in the head of an html document. Stylesheet arguments can be passed along as well in proper order.

```
htmlHead <- function(author = "Matthew Leonawicz", title = author, script.paths = NULL,
    stylesheet.paths, stylesheet.args = vector("list", length(path.stylesheets)),
    ...) {
    x <- paste0("<!DOCTYPE html>\n\n<html xmlns=\"http://www.w3.org/1999/xhtml\">\n\n<head>\n\n<meta cha
        author, " />\n\n<title>", title, "</title>\n")

    if (is.character(script.paths))
        x <- c(x, paste0(paste0("<script src=\"", script.paths, "\"></script>",
            collapse = "\n"), "\n"))

    x <- c(x, "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\" />\n")

    if (is.character(stylesheet.paths)) {
        n <- length(stylesheet.paths)
        stopifnot(is.list(stylesheet.args))
        stopifnot(length(stylesheet.args) == n)
        for (i in 1:n) {
            string <- ""
            if (is.list(stylesheet.args[i])) {
                v <- stylesheet.args[i]
                arg <- names(v)
                if (is.character(arg) && all(arg != ""))
                  string <- paste0(" ", paste(arg, paste0("\"", v, "\""), sep = "=",
                    collapse = " "))
            }
            x <- c(x, paste0("<link rel=\"stylesheet\" href=\"", stylesheet.paths[i],
                "\"", string, ">\n"))
        }
    }

    x <- c(x, "</head>\n")
    x

}
```

0.0.4 htmlBodyTop

`htmlBodyTop` currently is used for including custom CSS and a background image in the html body. CSS can be included as a text string or as a path to a CSS file.

```
htmlBodyTop <- function(css.file = NULL, css.string = NULL, background.image = "",
    include.default = TRUE, ...) {
    x <- "<style type = \"text/css\">\n"

    default <- paste0("\n\t.main-container {\n\t  max-width: 940px;\n\t  margin-left: auto;\n\t  margin-
        background.image, "\");\n\t  background-attachment: fixed;\n\t  background-size: 1920px 1080px;\

    if (!is.null(css.file))
        y <- readLines(css.file) else y <- ""
    if (!is.null(css.string))
        y <- c(y, css.string)
    if (include.default)
        y <- c(default, y)

    z <- "\n</style>\n\t<div class=\"container-fluid main-container\">\n\t"

    c(x, y, z)
}
```

0.0.5 htmlBottom

`htmlBottom` does not do anything else at this time other than close up the html document.

```
htmlBottom <- function(...) {
    # temporary
    "</body>\n\t</html>"
}
```

0.0.6 genUserPage

`genUserPage` generates a Github user account web page by combining precompiled html files of container elements made using `genPanelDiv` as well as various lingering hardcoded elements for my own work. I use this function to produce my main Github user page, the `index.html`, as well as supplemental gallery pages.

```
genUserPage <- function(file = "C:/github/leonawicz.github.io/index.html", containers = NULL,
    navbar = "", ...) {
    x1 <- htmlHead(...)
    x2 <- htmlBodyTop(...)
    if (!is.null(containers))
        x3 <- sapply(containers, function(x) paste0(paste0(readLines(x), collapse = "\n"),
            "\n\n")) else x3 <- ""
    x4 <- htmlBottom(...)
    nb <- if (file.exists(navbar) && substr(navbar, nchar(navbar) - 4, nchar(navbar)) ==
        ".html")
        nb <- paste0(paste0(readLines(navbar), collapse = "\n"), "\n\n")
    sink(file)
    sapply(c(x1, x2, nb, x3, x4), cat)
    sink()
    cat("Github User page html file created.\n")
}
```