Project Management

Matthew Leonawicz

December 31, 2014

1 Related items

Currently there is only this unpackaged **R** script, accompanying code for a projects hierarchy diagram and a code flow diagram based on the current development of this project, and a simple script for generating documents based on project **R** code.

1.1 Files and Data

This project does not use any data. It does make use of supplemental libraries for formatting during html document generation. `proj_sankey.R` and `code_sankey.R` are used to produce of project hierarchy diagram of my current projects and a code flow diagram for this project, respectively. `drg.R` is used to assist in dynamic report generation.

1.2 Code flow

The Sankey diagram has become part of project management. Each project has its own, detailing the relationships among **R** code and data relevant to the project, and in some cases, how they relate to code and data files which are more general and span multiple projects. In general, for my projects I would only provide the code flow diagram here among the rest of the project documentation, but since this is the project management project and I am introducing its use, in this case I will also show the code I use to make the diagram.

2 R code

Here is the code used to generate the code flow diagram.

2.1 Initial setup

Two packages are required.

```
require(igraph)
require(rCharts)
```

Project file names can be loaded using `pattern` arguments with `list.files` pointing to both the `code` and relevant `docs` directories. However, there is a substantial amount of hardcoding involved for any project as shown here. For example, the `rpm.R` script contains all the `rpm` functions. Although this script has a complete, corresponding `rpm.Rmd` file, I decided to subsequently break out the functions and other content from `rpm.Rmd` into multiple Rmd files. Such ad hoc project-specific decisions require that I take note here and make the related distinctions.

```r
setwd("C:/github/ProjectManagement/docs/Rmd")

c0 <- list.files("../../code", pattern = ".R$")
c1a <- c("_output.yaml", "navbar.html", "leonawicz.github.io")
c1b <- "supporting libraries"
c1c <- "in_header.html"

c2 <- list.files(pattern = ".Rmd$")
rmd.primary <- which(gsub(".Rmd", ".R", c2) %in% c0)
c2a <- c2[rmd.primary]
c2b <- c2[-rmd.primary]

c3 <- gsub(".Rmd", ".md", c2)
c4 <- gsub(".Rmd", ".html", c2)
```

Directional connections must be made among files. The connections are expressed by element-wise comparison of the equal-length `to` and `from` vectors.

```r
from <- c(
        c0,     rep("rpm.Rmd", length(c2b)), c2, c2, rep("pm.R", length(c1a)), c1b, rep(c(c1a, c1c), eac

)
to <- c(
        c2a, c2b, c3, c4, c1a, "pm.R", rep(c4, length(c(c1a, c1c)))
)
```

2.2 Create a directional tree graph

The vectors are combined in a data frame and the `igraph` package is used to grow the tree diagram.

```r
relations <- data.frame(from = from, to = to)
g <- graph.data.frame(relations, directed = T, vertices = data.frame(c(c0, c1a,
    c1b, c1c, c2, c3, c4)))

gw <- get.data.frame(g)
gw$value <- 1
colnames(gw) <- c("source", "target", "value")
gw$source <- as.character(gw$source)
gw$target <- as.character(gw$target)
```

2.3 Display with rCharts

The `rcharts` package has functionality for turning this into an interactive D3 visualization, which is nice, particularly the mouseover interactivity, since there can be so much visual overlap among files for complex projects. Additional javascript can be included to alter the colors. My strengths are in **R** so I borrowed this code snippet from online, but if you have skills with javascript and D3 you could probably do better with color control and opacity I imagine.

```r
p <- rCharts$new()
p$setLib("http://timelyportfolio.github.io/rCharts_d3_sankey/libraries/widgets/d3_sankey")
p$setTemplate(script = "http://timelyportfolio.github.io/rCharts_d3_sankey/libraries/widgets/d3_sankey/l
p$set(data = gw, nodeWidth = 15, nodePadding = 10, layout = 32, width = 900,
    height = 800, margin = list(right = 20, left = 20, bottom = 50, top = 50),
    title = "Code Flow")

p$setTemplate(afterScript = "\n<script>\n  var cscale = d3.scale.category20b();\n  d3.selectAll('#{{ cha
```

Embed the chart in a document when rendering.

```
p$show("iframesrc", cdn=T)
```