Building the R package pathprint

Gabriel Altschuler

June 14, 2012

This document outlines the steps required to build or update the R package pathprint.

1 Loading the data

The first step is to load all the relevant files into the workspace. This includes the scripts to process a gene expression array to a pathway fingerprint, dataframes containing the annotation files, metadata and fingerprint database, and the pathway sets that define the fingerprint.

```
> # building pathprint R package
> scripts<-c("exprs2fingerprint",
           "consensusDistance",
           "consensusFingerprint",
           "customCDFAnn",
           "diffPathways",
           "single.chip.enrichment",
           "thresholdFingerprint",
           "hyperPathway"
           )
> data<-c("chipframe",
         "GEO.fingerprint.matrix",
         "GEO.metadata.matrix",
         "platform.thresholds",
         "pluripotents.frame",
         "genesets"
         )
> pathways<-c("pathprint.Ce.gs",
             "pathprint.Dm.gs",
             "pathprint.Dr.gs",
             "pathprint.Hs.gs",
             "pathprint.Mm.gs",
             "pathprint.Rn.gs"
             )
> dir <- "../"
> sapply(scripts, function(x){
   source(paste(dir, "scripts/", x, ".R", sep = ""))})
        exprs2fingerprint consensusDistance
value   ?                 ?
visible FALSE             FALSE
        consensusFingerprint customCDFAnn diffPathways
value   ?                    ?            ?
visible FALSE                FALSE        FALSE
        single.chip.enrichment thresholdFingerprint
value   ?                      ?
visible FALSE                  FALSE
        hyperPathway
value   ?
visible FALSE
> sapply(data, function(x){
   load(paste(dir, "data/", x, ".RData", sep = ""), .GlobalEnv)})
               chipframe   GEO.fingerprint.matrix
             "chipframe" "GEO.fingerprint.matrix"
     GEO.metadata.matrix      platform.thresholds
   "GEO.metadata.matrix"    "platform.thresholds"
      pluripotents.frame                 genesets
    "pluripotents.frame"               "genesets"
> sapply(pathways, function(x){
   load(paste(dir, "data/", x, sep = ""), .GlobalEnv)})
  pathprint.Ce.gs   pathprint.Dm.gs   pathprint.Dr.gs
"pathprint.Ce.gs" "pathprint.Dm.gs" "pathprint.Dr.gs"
  pathprint.Hs.gs   pathprint.Mm.gs   pathprint.Rn.gs
"pathprint.Hs.gs" "pathprint.Mm.gs" "pathprint.Rn.gs"
```

2 Compile source package

The `utils` script `package.skeleton` compiles all workspace objects into a new package. The argument `force = TRUE` specifies that the previous package directory is overwritten if it exists.

```
> packageName<-"pathprint"
> package.skeleton(list = c(scripts, data, pathways),
                  name = packageName, force = TRUE,
                  path = paste(dir, "package", sep = "")
                  )
```

3 Build package

Each object in an R package is described by a man file, this contains information on the type of object and any additional text that will appear in the help page. The next step is to edit these man files. Even for a help-file free version, some editing is still required to remove a dummy line inserted into the description file without which the package will not compile. A quick fix using a shell script to strip out the problematic lines is sufficient for an initial build. A complete package requires a manual edit of each man file.
Some additional points to consider

- To facilitate subsequent builds it is useful to keep a set of man files in a separate master directory, along with the description and datalist files.
- The description file contains information that is read and used to build the package. Ensure that fields such as `Version:` number, `Depends:` (for extra packages that are required), and `Suggests:` (for extra packages only in the examples).
- The R function `resaveRdaFiles` from the `tools` package should be used to optimize the compression of all data objects prior to the final build step (N.B. compressing before runnning `package.skeleton` does not help as all files are loaded and then re-compressed using gzip). Switching the compression type from gzip to bzip2 or xz can dramatically reduce file sizes for some objects. Note that this is a time consuming step and also requires that the line `Depends: R (>= 2.10)` is included in the `DESCRIPTION` file.
- In order to pass all tests run by `R CMD check` all objects must be ASCII encoded. Other encodings can be switched using the function `iconv`.
- In addition, `check` does not like objects being called using `data()` within scripts. As an alternative, use lazyloading of objects by specifying `LazyData: yes` in the `DESCRIPTION` file. This also requires a file named `datalist` to be added to the data directory that contains the names of all objects to be lazyloaded.
- If a Windows binary file is required and there is no access to a Windows machine, use the upload service, `http://win-builder.r-project.org/`.

```
> # Edit man files
> wd<-getwd()
> setwd(paste(dir, "package/", packageName, "/man", sep = ""))
> man<-dir()
> man<-man[-grep("package.Rd", man)]
> for (i in 1:length(man)){
   code<-paste("sh ", "../../../scripts/mansimple.sh " , man[i], sep = "")
   system(code)
 }
```

```
> # alternative is to copy the man files from the manfiles dir
> man<-dir()
> for (i in 1:length(man)){
   code<-paste("cp ../../", packageName,
               "_masterFiles/manFiles/", man[i],
               " ", man[i], sep = "")
   system(code)
   }
> # Add description and datalist files from master directory
> system(paste("cp ../../", packageName,
              "_masterFiles/DESCRIPTION ../DESCRIPTION", sep = ""))
> system(paste("cp ../../", packageName,
              "_masterFiles/datalist ../data/datalist", sep = ""))
> # Build package
> setwd(wd)
> setwd("../package")
> # Remove placeholder
> system(paste("rm ",packageName, "/Read-and-delete-me", sep = ""))
> # Add in the a manual compiled by collating all the Rd files
> system(paste("R CMD Rd2pdf ",packageName, sep = ""))
> system(paste("mkdir ", packageName, "/inst", sep = ""))
> system(paste("mkdir ", packageName, "/inst/doc", sep = ""))
> system(paste("mv ",packageName, ".pdf ", packageName, "/inst/doc/",
              packageName, ".pdf", sep = ""))
> # Include index html file that references the manual (and any other docs)
> system(paste("cp ", packageName, "_masterFiles/index.html ",
              packageName, "/inst/doc/index.html", sep = ""))
> # Re-compress data files (this step takes time)
> library(tools)
> resaveRdaFiles(paste(packageName, "/data", sep = ""))
> # Build the package
> system(paste("R CMD build ", packageName, sep = ""))
> # The package can also be tested (this step takes even more time)
> system(paste("R CMD check ", packageName, sep = ""))
> setwd(wd)
```