Building the R package GMAfunctions

Gabriel Altschuler

September 28, 2015

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

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.

```
> dir<-"../"
> source("../scripts/gabriel functions.R")
```

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<-"GMAfunctions"
> package.skeleton(list = ls()[-grep("^dir$", ls())],
                   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, this will be done in full for the final distribution. For this version, some editing is still required to remove a line in 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.

```
> # 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)
 }
> # Add description, namespace and datalist files from master directory
> system(paste("cp ../../", packageName,
              "_masterFiles/DESCRIPTION ../DESCRIPTION", sep = ""))
> system(paste("cp ../../", packageName,
              "_masterFiles/NAMESPACE ../NAMESPACE", sep = ""))
> system(paste("cp ../../", packageName,
              "_masterFiles/datalist ../data/datalist", sep = ""))
> system(paste("cp ../../", packageName,
               "_masterFiles/manFiles/GMAfunctions-package.Rd",
               " ", "GMAfunctions-package.Rd", sep = ""))
> # Build package
> setwd(wd)
> setwd("../package")
> system(paste("rm ",packageName, "/Read-and-delete-me", sep = ""))
> # Add in the a manual compiled by collating all the Rd files
> system(paste("mkdir ", packageName, "/inst", sep = ""))
> system(paste("mkdir ", packageName, "/inst/doc", 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 = ""))
> system(paste("cp ",packageName, "_masterFiles/nodePathways.pdf ", packageName,
              "/inst/doc/nodePathways.pdf", sep = ""))
> # If required the package can be tested
> # system(paste("R CMD check ", packageName, sep = ""))
> system(paste("R CMD build ", packageName, sep = ""))
> setwd(wd)
```

This package can now be installed from the command line, for example when installing on `hpc111` to a local library

```
> # install.packages(
> # "/home/galtschu2/fingerprint/package/pathprint.tar.gz",
> # lib = .libPaths()[3], repos = NULL, type = "source")
```