AnalysisDemo

Paul Shannon

December 9, 2014

1 Introduction

We wish to encourage computational biology and biostatistical contributions to Oncoscape. The general scheme is

- Modify the present package "AnalysisDemo" to embody your algorithm
- Use unit tests to ensure the code works with a variety of inputs.
- Send us your package, and we will wrap it up so that it appears as a computational service within Oncoscape.
- Or host it yourself, letting us help as needed to get the websockets/JSON communication working.

2 Run AnalysisDemo in its Distributed Form

First, install the package. If you are working in a shell from the command line:

```
R CMD INSTALL AnalysisDemo_0.99.0.tar.gz
```

You will want to explore the source code, so unpack the tarball, creating an **AnalysisDemo** directory:

```
tar zxf AnalysisDemo_0.99.0.tar.gz
```

Now change directories to unitTests, and run those tests:

```
cd AnalysisDemo/inst/unitTests
ls -l testPackage.R
```

Start R, "source" that file, then run the tests it includes:

```
R> source("testPackage.R")
R> runTests()  # this is a convenience function which runs five tests (see below)
```

You will see this output in your R console:

```
[1] --- test_noArgs_constructor
[1] --- test_args_constructor
[1] --- test_setExpression
[1] --- test_.trimMatrix
found 5/10 overlapping samples in the expession data, 8/11 overlapping genes
[1] --- test_score
found 8/10 overlapping samples in the expession data, 8/11 overlapping genes
found 8/10 overlapping samples in the expession data, 6/9 overlapping genes
[1] TRUE
```

3 Package Architecture and Design

This package defines a single S4 class, **AnalysisDemo** defined in AnalysisDemo/R/AnalysisDemo-class.R. If you are new to R's S4 classes, they will seem a bit odd at first, inspired as they are by the Common Lisp Object System (CLOS). We hope that the working example provided here gets you past most of the confusion.

Sample data is provided in two files found within the AnalysisDemo/data directory:

- **tbl.mrnaUnified.TCGA.GBM.RData**: public gene expression data for 325 patients and 12k genes
- **msigdb.RData**: genesets from the Broad Institute, lists of gene symbols associated with published analyses, Gene Ontology terms, KEGG pathways, etc.

Unit tests are provided with in a single file, as demonstrated above. These are extremely useful in at least these two ways:

- They make software engineering robust, and much simpler!
  See `http://www.bioconductor.org/developers/how-to/unitTesting-guidelines`
- For the programmer who wishes to use or modify the code, they effectively and compactly document, explain and demonstrate what the code does.

4 What the Unit Tests Do

- **test_noArgs_constructor**: establishes that the package infrastructure is sound, by creating an empty (and thus not otherwise useful) AnalysisDemo object.
- **test_args_constructor**: makes sure both data items (the expression array, and the msigdb genesets) can be read, that they can be used to create an AnalysisDemo object, and that accessors (getSampleIDs, getGeneSet) work as expected.
- **test_setExpression**: shows how to specify the expression matrix
- **test_score**: runs the naive and simple analysis we provide as a demonstration, using the specified geneset and patient (sample) ids, getting the expected result. The scoring method we use does a t-test comparing gene expression for the specified patients, in the specified geneset (here, "YAMANAKA_GLIOBLASTOMA_SURVIVAL_UP") against an equal number of randomly selected genes. The t-test is repeated for each of the patients.

5 Please Return a Rich and Detailed Result

Please return a rich and detailed result from your analysis: it will be packaged up in JSON, and returned via a websocket to the user's web browser, where we will display it for the user to explore.

For example, here is a brief session with AnalysisDemo (taken from the unitTests). Once the result is calculated and returned, we print it out to the console to demonstrate the level of detail we ask you to return.

```
data("msigdb")                      # variable "genesets" is loaded
data("tbl.mrnaUnified.TCGA.GBM")    # variabole "tbl.mrna" is loaded

   # tcga gbm samples with survival >6 years post diagnsosis, by inspection in Oncoscape
longSurvivors <- list("TCGA.02.0014", "TCGA.02.0021", "TCGA.02.0028", "TCGA.02.0080", "TCGA.02.0114",
                      "TCGA.06.6693", "TCGA.08.0344", "TCGA.12.0656", "TCGA.12.0818", "TCGA.12.1088")

geneset.of.interest <- genesets["YAMANAKA_GLIOBLASTOMA_SURVIVAL_UP"]
demo <- AnalysisDemo(sampleIDs=longSurvivors,
                     geneSet=geneset.of.interest,
                     sampleDescription="TCGA GBM long survivors",
                     geneSetDescription="msgidb:YAMANAKA_GLIOBLASTOMA_SURVIVAL_UP")

demo <- setExpressionData(demo, tbl.mrna)

set.seed(123)
scores <- score(demo)
   found 8/10 overlapping samples in the expession data, 8/11 overlapping genes
```

```
scores

  $sample.title
  [1] "TCGA GBM long survivors"

  $geneSet.title
  [1] "msgidb:YAMANAKA_GLIOBLASTOMA_SURVIVAL_UP"

  $actual.samples.used
  [1] "TCGA.02.0014" "TCGA.02.0021" "TCGA.02.0028" "TCGA.02.0080" "TCGA.02.0114"
  [6] "TCGA.08.0344" "TCGA.12.0656" "TCGA.12.1088"

  $actual.genes.used
  [1] "PKM2"   "RAB32"  "SLN"    "LDHC"   "SLC2A3" "ITGA5"  "DYRK3"  "TPI1"

  $unmatched.samples
  [1] "TCGA.06.6693" "TCGA.12.0818"

  $unmatched.genes
  [1] "FGFBP2"  "STK40"   "EMILIN2"

  $pvals
  $pvals$TCGA.02.0014
  [1] 0.01191561

  $pvals$TCGA.02.0021
  [1] 0.8945468

  $pvals$TCGA.02.0028
  [1] 0.8122718

  $pvals$TCGA.02.0080
  [1] 0.01048628

  $pvals$TCGA.02.0114
  [1] 0.02630814

  $pvals$TCGA.08.0344
  [1] 0.2464011

  $pvals$TCGA.12.0656
  [1] 0.3623026

  $pvals$TCGA.12.1088
  [1] 0.6808923
```