`CancerMutationAnalysis` package

Giovanni Parmigiani
Dana-Farber Cancer Institute and
Harvard School of Public Health
email: gp@jimmy.harvard.edu,

Simina M. Boca
Georgetown University Medical Center
email: smb310@georgetown.edu

May 3, 2016

Contents

1 Overview

This package contains software which implements a variety of methods for gene and gene-set level analysis in somatic mutation studies of cancer. The methods implemented are described in **?** and **?**, for the gene level analysis, and **?**, for the gene-set level analysis. The gene level methods considered were developed to distinguish between "driver genes" and "passenger genes;" the former play an active role in tumorigenesis, having mutations which are selected for in cancer, while the latter are mutated in tumor samples, but have no role in tumorigenesis. They incorporate the study design used in **?????**: The complete collection of coding exons (the exome) was sequenced in several "discovery samples," with a smaller number of genes being further sequenced in a larger number of "prevalence samples," based on the results of the discovery screen, and possibly, on prior knowledge about their role in cancer.
The gene-set methods implemented carry out the "patient-oriented" approach, which calculates gene-set scores for each sample, then combines them across samples. By comparison, gene-oriented methods calculate a score for each gene across all samples, then combine them into gene-set scores. The four patient-oriented methods described in **?** are included, as well as the gene-oriented method described in **?** (see also **?**), which is in the `limma` package and performs a Wilcoxon test. Data from several comprehensive studies are also distributed with this package (**?????**).
This vignette represents an introduction to the `CancerMutationAnalysis` package. The primary functions are: `cma.scores`, which calculates a variety of gene scores; `cma.fdr`, which estimates passenger probabilities and false discovery rates; and `cma.set.stat`, which implements a number of gene-set level methods. The function `cma.set.sim` performs gene-set simulations using either the permutation null or the passenger null (see **?**). The functions `extract.sims.method` and `combine.sims` are used to manipulate the objects resulting from `cma.set.sim`.

2 Data objects

We first load the data and look at them:

```
> library(CancerMutationAnalysis)
> data(WoodBreast07)
> data(WoodColon07)
> data(JonesPancreas08)
> data(ParsonsGBM08)
> data(ParsonsMB11)
```

The datasets are labelled according to the publication year of the study and the tumor type. Thus, data is available for studies of breast cancer, colon cancer, pancreatic cancer, glioblastoma multiforme (GBM), and medulloblastoma (MB).
The objects containing information on the somatic mutations are labelled `GeneAlter*`. As an example, consider the `GeneAlterGBM` object:

```
> head(GeneAlterGBM)
```

```
           Gene Type Sample Screen WTNuc Context MutNuc
1 MRPL55@@15731  Mut  BR11P   Disc     C     CpG      T
2     USH2A@@45  Mut  BR20P   Disc     G     GpA      T
3   KCNH1@@6084  Mut  BR09P   Disc     G              T
4   LAMB3@@1338  Mut  BR17X   Disc     G              A
5    PIGR@@7060  Mut  BR08X   Disc     G     CpG      A
6     REN@@7303  Mut  BR20P   Disc     G     CpG      A
```

The columns represent the transcript, mutation type (`Mut`, `Amp`, `Del`, for point mutations, amplifications, and deletions, respectively), sample ID, screen (`Disc` or `Prev` for "Discovery" or "Prevalence"), wild type nucleotype, context, and nucleotide after mutation. Only point mutations are used to calculate gene scores, estimate passenger probabilities, and perform gene-set level analyses. Only discovery samples are used to perform the set analyses.
The `GeneCov*` objects contain data on the number of nucleotides of each type successfully sequenced for each transcript:

```
> head(GeneCovGBM)
```

```
           Gene Screen WTNuc Context Coverage
1 15E1.2@@13849   Disc     C     CpG 387.0147
2 15E1.2@@13849   Prev     C     CpG   0.0000
3 15E1.2@@13849   Disc     G     CpG 465.2525
4 15E1.2@@13849   Prev     G     CpG   0.0000
5 15E1.2@@13849   Disc     G     GpA 470.0150
6 15E1.2@@13849   Prev     G     GpA   0.0000
```

The `GeneSamp*` objects store the number of discovery and prevalence samples for each transcript:

```
> head(GeneSampGBM)
```

```
           Gene Screen NrSamp
1 15E1.2@@13849   Disc     21
2 15E1.2@@13849   Prev      0
3 2'-PDE@@16564   Disc     21
4 2'-PDE@@16564   Prev      0
5  3'HEXO@@9816   Disc     21
6  3'HEXO@@9816   Prev      0
```

The `BackRates*` objects contain estimated passenger probabilities used in the original studies.

3 Calculating gene scores

Gene scores are obtained using the `cma.scores` function. These scores include the Cancer Mutation Prevalence (CaMP) score (**?**) and the log Likelihood Ratio (logLRT) score (**?**). They take into account the mutation profile and the total number of nucleotides successfully sequenced for each gene, as well as an estimated "background passenger rate" of mutation:

```
> ScoresGBM <- cma.scores(cma.alter = GeneAlterGBM,
+                         cma.cov = GeneCovGBM,
+                         cma.samp = GeneSampGBM,
+                         passenger.rates = BackRatesGBM["MedianRates",])
> head(ScoresGBM)
```

```
                   CaMP   logLRT
A2M@@408      0.9028709 2.158645
A4GALT@@21470 0.5706862 1.678895
ABCA10@@388   1.0423011 2.575103
ABCA4@@140    0.7044877 1.853172
ABCA7@@194    0.1076564 1.157538
ABCB6@@2071   0.5499152 1.655159
```

4 Estimating passenger probabilities

Passenger probabilities for individual genes are estimated using the `cma.fdr` function, which performs an empirical Bayes (EB) analysis. The output of this function is a list, each score having an entry which, for each gene, gives the corresponding score, the number of genes with scores greater or equal in the dataset (`F`), the average number of genes with scores greater or equal in the simulated datasets (`F0`), the estimated false discovery rate (`Fdr`), the estimated local false discovery rate (`fdr`), and the estimated value of the prior probability of the gene being null (`p0`). The estimated passenger probabilities for individual genes are the estimate local false discovery rates. If `showFigure=TRUE`, a plot is also generated for each score, displaying the right tail of the density of null scores and a 1-D "rug plot" histogram of the observed scores. A cutoff is chosen so that the false discovery rate has the largest possible value below it. See Figure **??** for an example.

```
> set.seed(188310)
> FdrGBM <-  cma.fdr(cma.alter = GeneAlterGBM,
+                    cma.cov = GeneCovGBM,
+                    cma.samp = GeneSampGBM,
+                    scores = "logLRT",
+                    passenger.rates = BackRatesGBM["MedianRates",],
+                    showFigure=TRUE,
+                    cutoffFdr=0.1,
+                    M = 5)
```

```
Score: logLRT.
```

```
> head(FdrGBM[["logLRT"]])
```

```
                            Score     F    F0 Fdr fdr p0
PCOTH@@18196          5.837195e-05 20671 20671   1   1  1
```

```
PMS2L3@@9470           9.693291e-05 20670 20670   1   1  1
ENST00000360216@@16327 1.011182e-04 20669 20669   1   1  1
CCL4L2@@14948          1.029200e-04 20668 20668   1   1  1
Q6PDB4_HUMAN@@17521    1.036599e-04 20667 20667   1   1  1
LOC149643@@15875       1.085358e-04 20666 20666   1   1  1
```

5 Performing gene-set analyses

The function `cma.set.stat` returns a data-frame with p-values and q-values for all the methods selected. By default, all the methods are implemented; however this takes several minutes. Setting the `gene.method` parameter to `TRUE` implements the gene-oriented method. The other method parameters refer to the patient-oriented methods: `perm.null.method` refers to the permutation null without heterogeneity, `perm.null.het.method` to the permutation null with heterogeneity, `pass.null.method` to the passenger null without heterogeneity, and `pass.null.het.method` to the passenger null with heterogeneity. See **?** for more details on all these methods and **?** for a related method.
We use gene-sets from KEGG (**?**) in this example through the `KEGG.db` package (**?**):

```
> library(KEGG.db)
> KEGGPATHID2EXTID

PATHID2EXTID map for KEGG (object of class "AnnDbBimap")
```

Since the genes in the GBM datasets are annotated by gene names, while `KEGGPATHID2EXTID` uses EntrezGene identifiers, we also need to load the vector mapping the identifiers to the names, which we obtained by using the `biomaRt` package (**?**).

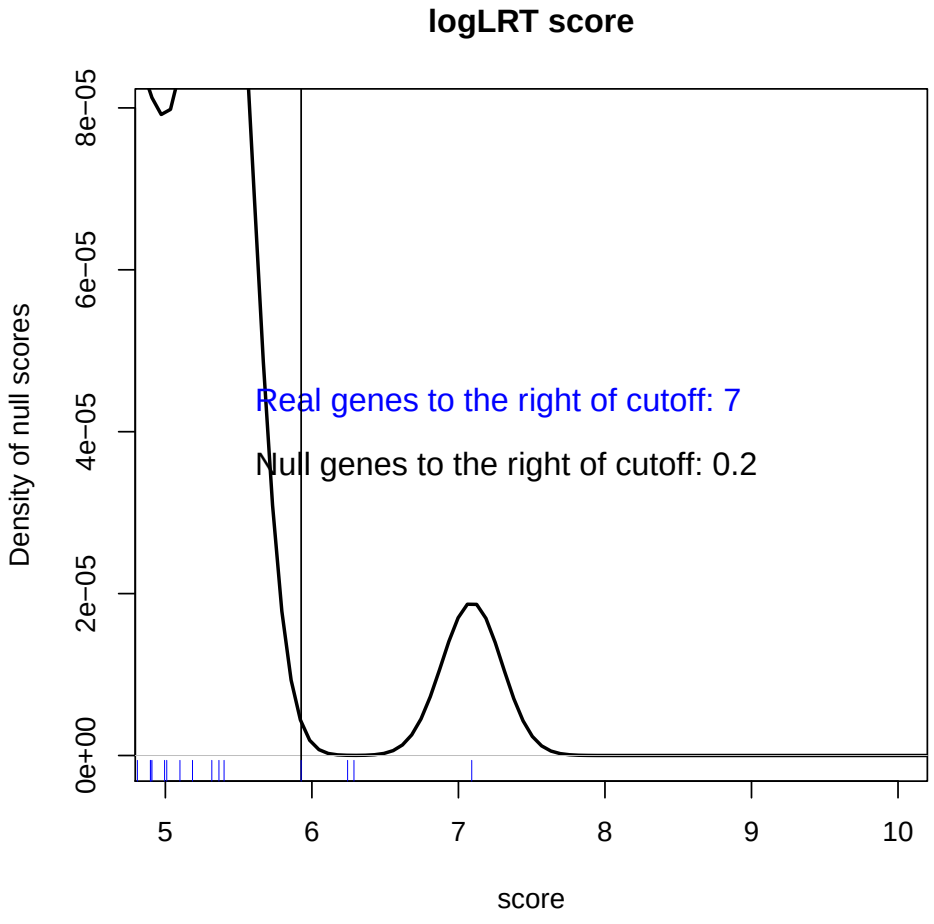
```
> data(EntrezID2Name)
```

We now use the function `cma.set.stat` to perform the gene-set analyses on three KEGG sets:

```
> as.character(KEGGPATHNAME2ID[c("Endometrial cancer",
+                                "Non-small cell lung cancer",
+                                "Alanine, aspartate and glutamate metabolism")])

Alanine, aspartate and glutamate metabolism
                                   "00250"
                         Endometrial cancer
                                   "05213"
                  Non-small cell lung cancer
                                   "05223"

> SetResultsGBM <-
+   cma.set.stat(cma.alter = GeneAlterGBM,
+                cma.cov = GeneCovGBM,
+                cma.samp = GeneSampGBM,
+                GeneSets = KEGGPATHID2EXTID[c("hsa05213",
+                  "hsa05223",  "hsa00250")],
+                ID2name = EntrezID2Name,
```

Figure 1: Example of figure obtained by setting `showFigure=TRUE` in the `cma.fdr` function.

```
Score: logLRT.

                              Score     F    F0 Fdr fdr p0
PCOTH@@18196           5.837195e-05 20671 20671   1   1  1
PMS2L3@@9470           9.693291e-05 20670 20670   1   1  1
ENST00000360216@@16327 1.011182e-04 20669 20669   1   1  1
CCL4L2@@14948          1.029200e-04 20668 20668   1   1  1
Q6PDB4_HUMAN@@17521    1.036599e-04 20667 20667   1   1  1
LOC149643@@15875       1.085358e-04 20666 20666   1   1  1
```



```
+                 gene.method = FALSE,
+                 perm.null.method = TRUE,
+                 perm.null.het.method = FALSE,
+                 pass.null.method = TRUE,
+                 pass.null.het.method = FALSE)
```

The resulting object is a data-frame containing p-values and q-values for the methods which were implemented:

```
> SetResultsGBM
```

```
         p.values.perm.null q.values.perm.null p.values.pass.null
hsa05213       9.198666e-16       2.759600e-15       2.108377e-10
hsa05223       1.678783e-15       2.518175e-15       3.403831e-11
hsa00250       3.109024e-01       3.109024e-01       6.340137e-01
         q.values.pass.null
hsa05213       3.162565e-10
hsa05223       1.021149e-10
hsa00250       6.340137e-01
```

6 Simulating data for gene-sets

We now demonstrate the use of the `cma.set.sim` function, which simulates datasets under either the permutation or passenger null (see `?`), depending on whether `pass.null` is set to `TRUE` or `FALSE`, and calculates the p-values and q-values for those datasets for the selected methods. The simulations may also include spiked-in gene-sets, by using the `perc.samples` and `spiked.set.sizes` options. For example, if one desires to have two spiked-in gene-sets, both having 50 genes, but one having the probability of being altered in any given sample of 0.75 and the other of 0.95, then these parameters should be set to `perc.samples = c(75, 90)` and `spiked.set.sizes = 50`. The spiked-in gene-sets are artificially created with hypothetical genes (for more details on how they are generated, see `?`). To simulate the data without spiked-in sets, under the permutation or passenger null hypotheses, the parameters should be set as following: `perc.samples = NULL, spiked.set.sizes = NULL`. The object outputted by `cma.set.sim` is of the class `SetMethodsSims`. Note that this code takes several minutes to run:

```
> set.seed(831984)
> resultsSim <-
+     cma.set.sim(cma.alter = GeneAlterGBM,
+                 cma.cov = GeneCovGBM,
+                 cma.samp = GeneSampGBM,
+                 GeneSets = KEGGPATHID2EXTID[c("hsa05213",
+                   "hsa05223", "hsa00250")],
+                 ID2name = EntrezID2Name,
+                 nr.iter = 2,
+                 pass.null = TRUE,
+                 perc.samples = c(75, 95),
+                 spiked.set.sizes = c(50),
+                 show.iter = TRUE,
+                 gene.method = FALSE,
```

```
+                 perm.null.method = TRUE,
+                 perm.null.het.method = FALSE,
+                 pass.null.method = TRUE,
+                 pass.null.het.method = FALSE)
> resultsSim

Simulation results for gene-set analysis of mutations
  Data-generating mechanism : Passenger null
  Number of simulations     : 2
  Number of gene-sets       : 5
     Original  : 3
     Spiked-in : 2
  Spiked-in sets            :
     Probability of being mutated in a set : 0.75 0.95
     Length (as number of genes)           : 50

> slotNames(resultsSim)

[1] "null.dist"        "perc.samples"     "spiked.set.sizes" "GeneSets"
[5] "cma.alter"        "cma.cov"          "cma.samp"         "Scores"
[9] "results"

> resultsSim@null.dist

[1] "Passenger null"
```

6.1 Manipulating the `SetMethodsSims` objects

We provide 2 functions to help manipulate the `SetMethodsSims` objects which result from the `cma.set.sim` function: `extract.sims.method` and `combine.sims`. The `extract.sims.method` function is used to obtain a single data frame with the p-values or q-values from one of the specific methods. For instance, the command to get the p-values for the permutation null with no heterogeneity method is:

```
> extract.sims.method(resultsSim,
+                     "p.values.perm.null")

                       [,1]         [,2]
hsa05213       1.790842e-02 3.326939e-01
hsa05223       2.142885e-02 6.030915e-01
hsa00250       5.652371e-01 2.824730e-01
gene.set.50.75 8.521175e-10 4.210103e-14
gene.set.50.95 1.048898e-17 8.187030e-16

>
```

The function `combine.sims` may be used to combine 2 simulations:

```
> combine.sims(resultsSim, resultsSim)
```

```
Simulation results for gene-set analysis of mutations
  Data-generating mechanism : Passenger null
  Number of simulations     : 4
  Number of gene-sets       : 5
     Original  : 3
     Spiked-in : 2
  Spiked-in sets            :
     Probability of being mutated in a set : 0.75 0.95
     Length (as number of genes)           : 50
```

7 Acknowledgments

We would like to thank Michael A. Newton and Luigi Marchionni for generously sharing their code and Benjamin Langmead, Fernando Pineda, and Michael Ochs for some very helpful discussions.