Additional File 3: Overview of the package exposed functions

SLAPenrich is implemented as an open-source Bioconductor R-package (submission in progress) and it is public available on GitHub (at https://github.com/francescojm/SLAPenrich). It contains seven exposed functions available to the user, nine internal functions and two data objects.
The referenced equations are contained in the formal description of the statistical framework underlying SLAPenrich (Additional File 1).

Input/Output

Of the thirteen exposed functions, two are for data input/output: the first one,

```
SLAPE.readDataset(filename),
```

reads a dataset stored in a .csv file as a sparse binary matrix; the second one,

```
SLAPE.write.table(PFP,EM,filename='', fdrth=Inf,exclcovth=0,
                  PATH_COLLECTION, GeneLenghts),
```

extracts from the `PFP` (pathway fingerprints) object (outputted by the `SLAPE.analysis` function, see below) the subset of pathways (from the collection specified in the pathway collection specified in `PATH_COLLECTION`) whose enrichment false discovery rate is below the threshold specified in the parameter `fdrth` and whose exclusive coverage (see methods) is above the threshold specified in the parameter `exclcovth`. The extracted

enriched pathways are then assembled and written in the csv file specified in the parameter `filename`, together with other information such as, for example, the percentage of altered samples of the initial dataset (specified in the matrix `EM`) when considering individual genes in a given pathway, and total exonic block lengths of the enriched pathways (extracted from the data object specified in `GeneLenghts)`.

Core analysis

The core analysis function implementing the statistical framework described in the methods section is

```
SLAPE.analyse(EM, show_progress=TRUE, correctionMethod='fdr',
NSAMPLES=1, NGENES=1, accExLength=TRUE, BACKGROUNDpopulation=NULL,
PATH_COLLECTION, path_probability='Bernoulli', Rho=10^-6,GeneLenghts)
```

This function takes in input a dataset (the parameter `EM`) stored in a sparse binary matrix, or a sparse matrix with integer non-null entries. In this matrix the columns correspond to samples, the rows correspond to genes and a non-zero entry indicates the presence of a somatic mutations harbored by a given sample in a given gene. If the matrix contains integer entries then they are deemed as the number of somatic point mutations harbored by a given sample in a given gene (these values will be considered to account for the sample mutation rate if the analysis takes into account of the gene exonic lengths, or converted in binary values otherwise, see below).

For each pathway gene-set *P* in the pathway collection specified in the parameter `PATH_COLLECTION`, and an inputted genomic dataset (summarized by the parameter `EM`),

this function computes first of all a vector of probabilities $\pi = \{p_i\}$ quantifying how likely each sample is to harbor at least one somatic mutation in a gene belonging to P, by random chance.

These probabilities are computed by default using a Bernoulli model accounting for the total exonic block lengths of all the genes belonging to P, and the expected or observed background mutation rate (Additional File (AF) Equation 4) [1,2] (as specified by the parameter `Rho`, which in the second case should be set equal to `NULL`). Alternatively, these probabilities can be computed through a complementary cumulative hypergeometric distribution evaluated at $X = 0$ and taking into account of the mutation burden of the samples, the size of P in terms of number of genes (AF Equations 1 and 2, and `accExLength = FALSE`), or its total exonic content block length (AF Equation 3 and `accExLength = TRUE`, the default setting). In all the tests make use of a gene background population that can be defined by the user (through the parameter `BACKGROUNDpopulation`) or assembled pooling together all the genes belonging to at least one pathway of the collection specified in `PATH_COLLECTION`.

After π has been computed, this function computes a *pathway alteration* score at the population level, quantifying the deviance of the number of samples in the datasets harbouring at least a somatic mutation in at least one gene of P, $O(P)$ (Equation 6) from its random expectation $E(P)$ (AF Equation 7, which is computed summing the $\{p_i\}$ across all the samples, AF Equation 5). Finally, it computes the significance of this score with a p-value against the null hypothesis: "$O(P)$ is drawn from a Poisson binomial distribution with $\{p_i\}$ success probabilities" (AF Equation 8). This comes from the observation that if there is no tendency for a given pathway to be recurrently mutated

across m samples of the datasets, then each of these samples can be considered as the observation of a single Bernoulli trial (in a series of m of them), where the event under consideration in the i-th trial is "At least one gene belonging to P is mutated in the i-th sample". The success probability of this event is given by p_i. Worthy of note is that a Poisson binomial distribution should be considered instead of a simple binomial distribution because the p_i are, of course, not identical.

After alteration scores and corresponding significance have been assessed for all the pathways considered in the analysis, resulting p-values are corrected for multiple hypothesis testing with a user-defined method, specified by the parameter `correctionMethod`. Possible values for this parameter are all the admissible values of the parameter `method` in the built-in function `p.adjust` of R, plus *qvalue* through which the user can select the Storey-Tibshirani [3,4] correction method.

Through the parameters `NSAMPLES` and `NGENES` the minimal values that the number of samples harbouring a mutation in the pathway P, and the number of genes in P mutated in at least one sample should assume in order for P to be included in the analysis can be specified, respectively. The default value for these two parameters is 1.

As mentioned, the two parameters `accExLength` and `BACKGROUNDpopulation` specify whether the gene exonic lengths should be taken into account while defining the probabilities $\{p_i\}$ described above, and the collection of official symbols of the genes that should be included in the background population in the used statistical framework, respectively. If the value of `accExLength` is `TRUE` (default) then the non-null values of the matrix coding for the inputted dataset (`EM`) are deemed to indicate the number of somatic point mutations harbored by a given gene in a give sample.

`BACKGROUNDpopulation` could be, for example, all the genes whose mutational status is accounted in the inputted dataset `EM`. If the value of `BACKGROUNDpopulation` is `NULL` (default) then the set of all the genes included in at least one pathway of the collection included in the analysis is used as background population.
Finally, the parameter `show_progress` determines if a progress bar should be visualized during the execution of the analysis.

For an inputted dataset of m samples and a collection of p pathways included in the analysis, `SLAPE.Analyse` outputs also (i) a $p \times m$ binary *pathway alteration matrix* where rows indicate pathways, columns indicate samples and non-null entries indicate the presence of at least a somatic mutation in at least one gene of a given pathway in a given sample; (ii) a $p \times m$ *pathway mutation probability matrix*, where the j-th row contains the vector of probabilities $\{p_{j,i}\}$ of the i-th sample harboring at least a somatic mutation in at least one gene of the j-th pathway, by random chance; (iii) a vector of *pathway alteration expectations* (with an element for each pathway) with an estimation of the expected number of samples harbouring at least one somatic mutation in at least on gene of a given analysed pathway; (iv) a vector of *pathway exclusive coverage scores* quantifying the tendency of the genes composing each of the analysed pathway to be mutated in a mutual exclusive fashion; (v) a list of individual *binary pathway alteration matrices* (one for each analysed pathway), where the generic matrix M_i has dimensions $k \times m$, where k is the number of genes in the i-th pathway, m is the number of samples in the analysed dataset and a generic non-null entry in position h, j is equal to 1 if the h-th

gene of the *i*-th pathway of the analyzed collection harbors at least one somatic mutation in the *j*-th sample; (vi) a vector of *numerical pathway identifiers*.

Visualisation

Storing the results outputted by the `SLAPE.Analyse` function in a list, it is possible to visualize them systematically and to produce pdf files with resulting plots using the function

```
SLAPE.serialPathVis(EM, PFP, fdrth=5, exCovTh=50, PATH='./',
                    PATH_COLLECTION).
```

This function extracts from the list of results outputted by `SLAPE.Analyse` (specified by the `PFP` parameter) those pathways (from the collection specified by the parameter `PATH_COLLECTION`) with an enrichment false discovery rate (FDR) below the user defined threshold value specified by the parameter `fdrth`, and with an exclusive coverage score (AF Equation 9) above the threshold value specified by the parameter `exCovTh`. All the figures produced by this function are stored in the directory specified by the parameter `PATH`.

After selecting the pathways following the user definitions, this function systematically calls for each of them (distinguished by their numerical identifier, `Id`) the sub-routine:

```
SLAPE.pathVis(EM, PFP, Id, i=NULL,PATH='./', PATH_COLLECTION).
```

Before producing the plots, `SLAPE.pathVis` rearranges rows and columns of the *alteration matrix* of the `Id` pathway through a heuristic mutual-exclusivity sorting procedure (detailed in the method), which highlights the tendency of the composing genes to be mutated in a mutual exclusive fashion across the samples of the analyzed dataset. This sorting is implemented in the function

```
SLAPE.heuristic_mut_ex_sorting(EM),
```

which is exported, therefore available to the user and suitable for sorting any type of binary matrix. After this re-arrangement the *alteration matrix* of the `Id` pathway is visualized and stored in a pdf file as a binary heatmap with genes on the rows, samples on the columns, and blue entries indicating the presence of somatic mutations in given gene/sample combinations (an example is reported in figure 2A). Additionally, *mutation probabilities* and *alteration expectations* are visualized and saved as bar diagrams, together with other statistical scores in a separate figure file.

Extraction of core-components from the enriched pathways

The function

```
SLAPE.core_components(PFP, EM, PATH='./',
fdrth=Inf, exclcovth=0, PATH_COLLECTION),
```

identifies sets of core-components genes frequently altered and shared by multiple enriched pathways identified by the `SLAPE.Analyse` function (and specified in `PFP`). These core-component gene sets are supposed to lead the enrichment outcomes. To detect such core components, the function executes a fast greedy community detection algorithm [5,6], implemented in the `fastgreedy.community` function of the `iGraph` package [7-9]. The analysis performed by this function considers only enrichments at a false discovery rate lower than the threshold value specified in the parameter `fdrth` and corresponding to pathways with an exclusive coverage greater than the threshold value specified in the parameter `exclcovth`. After these core-component gene-sets have been identified, this function visualizes them together with the pathways they belong to through a set of *membership matrices*: binary heatmaps with pathways on the columns, a set of core-component genes on the rows and non-empty entries specifying to which enriched pathway each gene belongs to (an example is provided in figure 2B). These heatmaps are stored in individual pdf files and saved in the directory specified by the `PATH` parameter.

Differential pathway enrichment analysis

The function

```
SLAPE.diff_SLAPE_analysis (EM, contrastMatrix, positiveCondition,
negativeCondition, SLAPE.FDRth=5, ...)
```

performs a differential enrichment analysis of pathway alterations at the sample population level between two sample subsets of dataset specified in the parameter `EM` (defined as for the previous function). The parameter `contrastMatrix` specifies which samples are included in each sub-population and it is a binary matrix with sample identifiers on the rows and condition identifiers on the columns. The sample identifiers should match those of the initial datasets, i.e. the column headers of the `EM` matrix. A 1 in the position i, j of such a matrix indicates that the i-th sample is included in the sub-population corresponding to the j-th condition.

The two sub-populations to be contrasted are specified by the parameters `positiveCondition` and `negativeCondition` that should match two different column headers of the `contrastMatrix`. This function first performs two independent SLAPenrich analyses on the two user-defined sub-populations of samples with an experimental setting specified by the additional parameters (not listed in the function signature above), which are the same of the `SLAPE.analyse` function. Then it selects the pathways with an enrichment FDR smaller than the value specified in the `SLAPE.FDRth=5` in at least one of the two independent analyses. For this set of pathways a differential enrichment score is computed, as detailed in the method, and summary heatmaps are visualised, as shown in figure 3.

Accessory functions and data objects

The SLAPenrich package contains additional exposed functions allowing users to:

- check the consistency of (and possibly update the) gene symbol identifiers in both genomic datasets and pathway collection data object with the Hugo gene nomenclature (HGNC) catalogue, including approved gene symbols and previously used synonyms, contained in the `SLAPE.hgnc.table_20160210` data object;
- update or create a new HGNC catalogue, by downloading the most up-to-date version from the HUGO Gene Nomenclature Committee web-site (www.genenames.org);
- compute the total length of the exonic block of a given gene, making use of the gene exon attribute data object `SLAPE.all_genes_exonic_lengths_ensemble_20160209`, containing the genomic coordinates of all the exons for all the genes (the genome-wide total exonic block lengths are already precomputed and available in the `SLAPE.all_genes_exonic_content_block_lengths_ensemble_20160209` data object, and can be updated with a dedicated function);
- update the gene exon attribute data object, by making use of functions from the biomaRt R package [10-12].

Additionally, different collections of pathway gene sets from the Pathway Commons data portal (v4-201311) [7,9,13-16], and their post-processed versions computed as detailed in the methods to reduce redundancies are embedded in the package as R objects. These objects contain, for each pathway, multiple information such as uniprot identifiers of the composing genes, official data source, and pathway title/definition.

Finally the genomic datasets and corresponding patient clinical information used in the case study described in the following sections are also provided as R objects.

References

1. Stratton MR, Campbell PJ, Futreal PA. The cancer genome. Nature. Nature Publishing Group; 2009;458:719–24.

2. Wendl MC, Barbazuk WB. Extension of Lander-Waterman theory for sequencing filtered DNA libraries. BMC bioinformatics. 2005;6:245.

3. Yeang CH, McCormick F, Levine A. Combinatorial patterns of somatic gene mutations in cancer. The FASEB Journal. 2008;22:2605–22.

4. Storey JD, Tibshirani R. Statistical significance for genomewide studies. Proceedings of the National Academy of Sciences of the United States of America. 2003;100:9440–5.

5. Schubert M, Iorio F. Exploiting combinatorial patterns in cancer genomic data for personalized therapy and new target discovery. Pharmacogenomics. 2014;15:1943–6.

6. Newman MEJ. Fast algorithm for detecting community structure in networks. Phys Rev E Stat Nonlin Soft Matter Phys. 2004;69:066133.

7. Ciriello G, Cerami E, Sander C, Schultz N. Mutual exclusivity analysis identifies oncogenic network modules. Genome Research. 2012;22:398–406.

8. Csardi G, Nepusz T. The igraph software package for complex network research. InterJournal, Complex Systems. 2006;1695:38.

9. Vandin F, Upfal E, Raphael BJ. De novo discovery of mutated driver pathways in cancer. Genome Research. 2012;22:375–85.

10. Jerby-Arnon L, Pfetzer N, Waldman YY, McGarry L, James D, Shanks E, et al. Predicting Cancer-Specific Vulnerability via Data-Driven Detection of Synthetic Lethality. Cell. Elsevier Inc; 2014;158:1199–209.

11. Durinck S, Spellman PT, Birney E, Huber W. Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. Nature Protocols. 2009;4:1184–91.

12. Srihari S, Singla J, Wong L, Ragan MA. Inferring synthetic lethal interactions from mutual exclusivity of genetic events in cancer. Biology Direct. Biology Direct; 2015;:1–18.

13. Cerami EG, Gross BE, Demir E, Rodchenkov I, Babur Ö, Anwar N, et al. Pathway Commons, a web resource for biological pathway data. Nucleic Acids Res. 2011;39:D685–90.

14. Li HT, Zhang J, Xia J, Zheng CH. Identification of driver pathways in cancer based on combinatorial patterns of somatic gene mutations. Neoplasma. 2016;63:57–63.

15. Lu S, Lu KN, Cheng S-Y, Hu B, Ma X, Nystrom N, et al. Identifying Driver Genomic Alterations in Cancers by Searching Minimum-Weight, Mutually Exclusive Sets. Beerenwinkel N, editor. PLoS Comput Biol. 2015;11:e1004257.

16. Constantinescu S, Szczurek E, Mohammadi P, Rahnenführer J, Beerenwinkel N. TiMEx: a waiting time model for mutually exclusive cancer alterations. Bioinformatics. 2015.