Package ‘hacksig’

October 13, 2022

Type Package

Title A Tidy Framework to Hack Gene Expression Signatures

Version 0.1.2

Description A collection of cancer transcriptomics gene signatures as well as a simple and tidy interface to compute single sample enrichment scores either with the original procedure or with three alternatives: the ``combined z-score'' of Lee et al. (2008) <doi:10.1371/journal.pcbi.1000217>, the ``single sample GSEA'' of Barbie et al. (2009) <doi:10.1038/nature08460> and the ``singscore'' of Foroutan et al. (2018) <doi:10.1186/s12859-018-2435-4>. The 'get_sig_info()' function can be used to retrieve information about each signature implemented.

License MIT + file LICENSE

URL `https://github.com/Acare/hacksig`

BugReports `https://github.com/Acare/hacksig/issues`

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 4.0)

Imports dplyr (>= 1.0.7), future.apply (>= 1.8.1), rlang (>= 0.4.11), stats (>= 4.0.5), tibble (>= 3.1.5), tidyr (>= 1.1.4)

Suggests covr, future (>= 1.22.1), ggplot2 (>= 3.3.5), knitr (>= 1.36), msigdbr (>= 7.4.1), purrr (>= 0.3.4), rmarkdown (>= 2.11)

VignetteBuilder knitr

NeedsCompilation no

Author Andrea Carenzo [aut, cre] (<`https://orcid.org/0000-0001-6804-9984`>), Loris De Cecco [aut] (<`https://orcid.org/0000-0002-7066-473X`>), Federico Pistore [aut]

Maintainer Andrea Carenzo <`andrea.carenzo@gmail.com`>

Repository CRAN

Date/Publication 2022-02-17 14:22:02 UTC

R topics documented:

check_sig 2
get_sig_genes 3
get_sig_info 3
hack_cinsarc 4
hack_class 5
hack_estimate 6
hack_immunophenoscore 8
hack_sig 10
test_expr 13

Index **14**

check_sig *Check signatures feasibility*

Description

`check_sig()` is a helper function that shows useful information about signatures that you want to test on your gene expression matrix.

Usage

```
check_sig(expr_data, signatures = "all")
```

Arguments

`expr_data`	A normalized gene expression matrix (or data frame) with gene symbols as row names and samples as columns.
`signatures`	It can be a list of signatures or a character vector indicating keywords for a group of signatures. The default (`"all"`) will cause the function to check for all the signatures implemented in `hacksig`.

Value

A tibble with a number of rows equal to the number of input signatures and five columns:

- `n_genes` gives the number of genes composing a signature;
- `n_present` and `frac_present` are the number and fraction of genes in a signature which are present in `expr_data`, respectively;
- `missing_genes` returns a named list of missing gene symbols for each signature.

See Also

`get_sig_info()`, `hack_sig()`

Examples

```
check_sig(test_expr)
check_sig(test_expr, "estimate")
```

get_sig_genes *Get signature gene identifiers*

Description

Obtain gene signatures implemented in `hacksig` as a named list of gene symbols.

Usage

```
get_sig_genes(keywords = "all")
```

Arguments

`keywords`	A character vector indicating keywords for a group of signatures. The default (`"all"`) will cause the function to check for all the signatures implemented in `hacksig`.

Value

A named list of gene signatures.

See Also

`get_sig_info()` to get valid keywords for signatures.

Examples

```
get_sig_genes()
get_sig_genes("estimate")
```

get_sig_info *Display available gene signatures*

Description

`get_sig_info()` returns information about all gene signatures implemented in `hacksig`.

Usage

```
get_sig_info()
```

Value

A tibble with one row per signature and four columns:

- `signature_id`, a unique identifier associated to a signature;
- `signature_keywords`, valid keywords to use in the `signatures` argument of `hack_sig()` and `check_sig()` as well as in the `keywords` argument of `get_sig_genes()`;
- `publication_doi`, the original publication DOI;
- `description`, a brief description about the signature.

See Also

`check_sig()`, `hack_sig()`, `get_sig_genes()`

Examples

```
get_sig_info()
```

`hack_cinsarc` *Hack the CINSARC classification*

Description

Given a gene expression matrix and a 0-1 vector indicating the distant metastasis status of samples, `hack_cinsarc()` classifies samples into one of two risk classes, C1 or C2, using the CINSARC signature as implemented in *Chibon et al., 2010*.

Usage

```
hack_cinsarc(expr_data, dm_status)
```

Arguments

`expr_data`	A normalized gene expression matrix (or data frame) with gene symbols as row names and samples as columns.
`dm_status`	A numeric vector specifying whether a sample has either (1) or not (0) developed distant metastasis.

Details

CINSARC (*Complexity INdex in SARComas*) is a prognostic 67-gene signature related to mitosis and control of chromosome integrity. It was developed to improve metastatic outcome prediction in soft tissue sarcomas over the FNCLCC (*Fédération Francaise des Centres de Lutte Contre le Cancer*) grading system.

Value

A tibble with one row for each sample in `expr_data` and two columns: `sample_id` and `cinsarc_class`.

Algorithm

The CINSARC method implemented in `hacksig` makes use of leave-one-out cross validation (LOOCV) to classify samples into C1/C2 risk groups (see *Lesluyes & Chibon, 2020*). First, gene expression values are centered by their mean across samples. Then, for each iteration of the LOOCV, mean normalized gene values are computed by metastasis group (i.e. compute the metastatic centroids). Then, one minus the Spearman's correlation between centered samples and metastatic centroids are computed. Finally, if a sample is more correlated to the non-metastatic centroid, then it is assigned to the C1 class (low risk). Conversely, if a sample is more correlated to the metastatic centroid, then it is assigned to the C2 class (high risk).

Source

codeocean.com/capsule/4933686/tree/v4

References

Chibon, F., Lagarde, P., Salas, S., Pérot, G., Brouste, V., Tirode, F., Lucchesi, C., de Reynies, A., Kauffmann, A., Bui, B., Terrier, P., Bonvalot, S., Le Cesne, A., Vince-Ranchère, D., Blay, J. Y., Collin, F., Guillou, L., Leroux, A., Coindre, J. M., & Aurias, A. (2010). Validated prediction of clinical outcome in sarcomas and multiple types of cancer on the basis of a gene expression signature related to genome complexity. *Nature medicine*, 16(7), 781–787. doi: 10.1038/nm.2174.

Lesluyes, T., & Chibon, F. (2020). A Global and Integrated Analysis of CINSARC-Associated Genetic Defects. *Cancer research*, 80(23), 5282–5290. doi: 10.1158/00085472.CAN200512.

Examples

```
# generate random distant metastasis outcome
set.seed(123)
test_dm_status <- sample(c(0, 1), size = ncol(test_expr), replace = TRUE)

hack_cinsarc(test_expr, test_dm_status)
```

`hack_class` *Hack signature classes*

Description

`hack_class()` is supposed to be used in combination after `hack_sig()` in order to classify your samples in one of two or more signature classes.

Usage

```
hack_class(sig_data, cutoff = "original")
```

Arguments

sig_data A tibble result of a call to hack_sig().

cutoff A character specifying which function to use to categorize samples by signature scores. Can be one of:

- "original" (default), apply the original publication method; if categorization is not expected, the median score is used as a threshold;
- "mean"/"median", samples will be classified as "low" or "high" with respect to the mean/median signature score, respectively;
- "tertiles", samples will be classified as "<= T1" (score lower than first tertile), "(T1, T2]" (score between first and second tertiles), "> T2" (score higher than second tertile);
- "quartiles", samples will be classified as "<= Q1" (score lower than first quartile), "(Q1, Q2]" (score between first and second quartiles), "(Q2, Q3]" (score between second and third quartiles), "> Q3" (score higher than third quartile).

Value

A tibble with the same dimension as sig_data, a column sample_id indicating sample identifiers and one column for each input signature giving signature classes.

See Also

hack_sig()

Examples

```
library(dplyr)
hack_sig(test_expr, "immune") %>% hack_class()
```

hack_estimate *Hack the ESTIMATE scores*

Description

Obtain *Immune*, *Stroma*, *ESTIMATE* and *Tumor Purity* scores from a cohort of samples, using the method implemented in *Yoshihara et al., 2013*.

Usage

```
hack_estimate(expr_data)
```

Arguments

expr_data A normalized gene expression matrix (or data frame) with gene symbols as row names and samples as columns.

Details

The ESTIMATE (*Estimation of STromal and Immune cells in MAlignant Tumors using Expression data*) method was developed with the aim to estimate the fraction of tumor cells in a sample by using gene expression instead of copy number data. The fundamental assumption of this method is that the tumor microenvironment is a very rich and dynamic ecosystem, in which immune infiltrating cells and stroma play a major role. The ESTIMATE score is defined as the combination (i.e. sum) of immune and stroma scores and can be thought of as a *"non-tumor score"*. Consequently, a high ESTIMATE enrichment gives a low tumor purity score and viceversa.

Value

A tibble with one row for each sample in `expr_data` and five columns: `sample_id`, `immune_score`, `stroma_score`, `estimate_score` and `purity_score`.

Algorithm

Raw immune and stromal signatures scores are computed using single sample GSEA with rank normalization (*Barbie et al., 2009*). Then, the ESTIMATE score is computed by summing the immune and stroma scores. Finally, the tumor purity score is obtained with the following formula:

$$Purity = cos(0.6049872018 + 0.0001467884 * ESTIMATE)$$

Source

bioinformatics.mdanderson.org/public-software/estimate/

References

Barbie, D. A., Tamayo, P., Boehm, J. S., Kim, S. Y., Moody, S. E., Dunn, I. F., Schinzel, A. C., Sandy, P., Meylan, E., Scholl, C., Fröhling, S., Chan, E. M., Sos, M. L., Michel, K., Mermel, C., Silver, S. J., Weir, B. A., Reiling, J. H., Sheng, Q., Gupta, P. B., . . . Hahn, W. C. (2009). Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. *Nature*, 462(7269), 108–112. doi: 10.1038/nature08460.

Yoshihara, K., Shahmoradgoli, M., Martínez, E., Vegesna, R., Kim, H., Torres-Garcia, W., Treviño, V., Shen, H., Laird, P. W., Levine, D. A., Carter, S. L., Getz, G., Stemke-Hale, K., Mills, G. B., & Verhaak, R. G. (2013). Inferring tumour purity and stromal and immune cell admixture from expression data. *Nature communications*, 4, 2612. doi: 10.1038/ncomms3612.

Examples

```
hack_estimate(test_expr)
```

hack_immunophenoscore *Hack the Immunophenoscore*

Description

Obtain various immune biomarkers scores, which combined together give the immunophenoscore (*Charoentong et al., 2017*).

Usage

```
hack_immunophenoscore(expr_data, extract = "ips")
```

Arguments

`expr_data`	A normalized gene expression matrix (or data frame) with gene symbols as row names and samples as columns.
`extract`	A string controlling which type of biomarker scores you want to obtain. Possible choices are: • `"ips"` (default), only raw and discrete IPS scores; • `"class"`, IPS scores together with the four summary class scores; • `"all"`, all possible biomarker scores.

Details

The immunophenoscore is conceived as a quantification of tumor immunogenicity. It is obtained by aggregating multiple immune biomarkers scores, which are grouped into four major classes:

- *MHC molecules* (**MHC**), expression of MHC class I, class II, and non-classical molecules;
- *Immunomodulators* (**CP**), expression of certain co-inhibitory and co-stimulatory molecules;
- *Effector cells* (**EC**), infiltration of activated CD8+/CD4+ T cells and Tem (effector memory) CD8+/CD4+ cells;
- *Suppressor cells* (**SC**), infiltration of immunosuppressive cells (Tregs and MDSCs).

The table below shows in detail the 26 immune biomarkers and cell types grouped by class together with the number of genes which represent them:

Class |	Biomarker/cell type |	No. genes
MHC	B2M	1
MHC	HLA-A	1
MHC	HLA-B	1
MHC	HLA-C	1
MHC	HLA-DPA1	1
MHC	HLA-DPB1	1
MHC	HLA-E	1
MHC	HLA-F	1
MHC	TAP1	1

MHC	TAP2	1
CP	CD27	1
CP	CTLA-4	1
CP	ICOS	1
CP	IDO1	1
CP	LAG3	1
CP	PD1	1
CP	PD-L1	1
CP	PD-L2	1
CP	TIGIT	1
CP	TIM3	1
EC	Act CD4	24
EC	Act CD8	26
EC	Tem CD4	27
EC	Tem CD8	25
SC	MDSC	20
SC	Treg	20

Value

A tibble with one row for each sample in `expr_data`, a column `sample_id` indicating sample identifiers and a number of additional columns depending on the choice of `extract`.

Algorithm

Samplewise gene expression z-scores are obtained for each of 26 immune cell types and biomarkers. Then, weighted averaged z-scores are computed for each class and the raw immunophenoscore ($IPS - raw$) results as the sum of the four class scores. Finally, the immunophenoscore (IPS) is given as an integer value between 0 and 10 in the following way:

- $IPS = 0$, if $IPS - raw \leq 0$;
- $IPS = [10 * (IPS - raw/3)]$, if $0 < IPS - raw < 3$;
- $IPS = 10$, if $IPS - raw \geq 3$.

Source

github.com/icbi-lab/Immunophenogram

References

Charoentong, P., Finotello, F., Angelova, M., Mayer, C., Efremova, M., Rieder, D., Hackl, H., & Trajanoski, Z. (2017). Pan-cancer Immunogenomic Analyses Reveal Genotype-Immunophenotype Relationships and Predictors of Response to Checkpoint Blockade. *Cell reports*, 18(1), 248–262. doi: 10.1016/j.celrep.2016.12.019.

See Also

hack_sig() to compute Immunophenoscore biomarkers in different ways (e.g. use signatures = "ips" and method = "singscore").

check_sig() to check if all/most of the Immunophenoscore biomarkers are present in your expression matrix (use signatures = "ips").

Examples

```
hack_immunophenoscore(test_expr)
hack_immunophenoscore(test_expr, extract = "class")
```

---

hack_sig *Score samples by gene signatures*

Description

Compute gene signature single sample scores in one of different ways. You can choose to apply either the *original* procedure or one of three single sample scoring methods: the combined z-score (*Lee et al., 2008*), the single sample GSEA (*Barbie et al., 2009*) or the singscore method (*Foroutan et al., 2018*).

Usage

```
hack_sig(
  expr_data,
  signatures = "all",
  method = "original",
  direction = "none",
  sample_norm = "raw",
  rank_norm = "none",
  alpha = 0.25
)
```

Arguments

expr_data
: A normalized gene expression matrix (or data frame) with gene symbols as row names and samples as columns.

signatures
: It can be a list of signatures or a character vector indicating keywords for a group of signatures. The default ("all") will cause the function to compute single sample scores for all the signatures implemented in hacksig.

method
: A character string specifying which method to use for computing the single sample score for each signature. You can choose one of:

- "original", the original method used by the authors of the signature;
- "zscore", the combined z-score method;
- "ssgsea", the single sample GSEA method;

- `"singscore"`, the singscore method;

`direction` A character string specifying the **singscore** computation method depending on the direction of the signatures. Can be on of:

- `"none"` (default), undirected signatures, that is you do not know whether the genes are up- or down-regulated;
- `"up"`, all genes in the signature are supposed to be up-regulated;
- `"down"`, all genes in the signature are supposed to be down-regulated;

`sample_norm` A character string specifying the type of normalization affecting the **single sample GSEA** scores. Can be one of:

- `"raw"` (default), obtain raw scores;
- `"separate"`, normalize raw scores in $[0, 1]$ across samples for each signature separately.
- `"all"`, normalize raw scores both across samples and signatures.

`rank_norm` A character string specifying how gene expression ranks should be normalized in the **single sample GSEA** procedure. Valid choices are:

- `"none"` (default), no rank normalization;
- `"rank"`, ranks are multiplied by `10000 / nrow(expr_data)`;
- `"logrank"`, normalized ranks are logged.

`alpha` A numeric scalar. Exponent in the running sum of the **single sample GSEA** score calculation which weighs the gene ranks. Defaults to $\alpha = 0.25$.

Details

For `"original"` method, it is intended the procedure used in the original publication by the authors for computing the signature score. `hack_sig()` can compute signature scores with the original method only if this is a relatively simple procedure (e.g weighted sum of fitted model coefficients and expression values). For more complex methods, such as CINSARC, ESTIMATE and Immunophenoscore, use the dedicated functions.

If `signatures` is a custom list of gene signatures, then the `"ssgsea"` method will be applied by default.

Value

A tibble with one row for each sample in `expr_data`, a column `sample_id` indicating sample identifiers and one column for each input signature giving single sample scores.

Algorithm

This section gives a brief explanation of how single sample scores are obtained from different methods.

Combined z-score:

Gene expression values are centered by their mean value and scaled by their standard deviation across samples for each gene (z-scores). Then, for each sample and signature, corresponding z-scores are added up and divided by the square root of the signature size (i.e. the number of genes composing a signature).

The combined z-score method is also implemented in the R package GSVA (*Hänzelmann et al., 2013*).

Single sample GSEA:

For each sample, genes are ranked by expression value in increasing order and rank normalization may follow (see argument rank_norm). Then, two probability-like vectors are computed for each sample and signature:

- P_{in}, the cumulative sum of weighted ranks divided by their total sum for genes in the signature;
- P_{out}, the cumulative sum of ones (indicating genes not in the signature) divided by the number of genes not in the signature.

The single sample GSEA score is obtained by adding up the elements of the vector difference $P_{in} - P_{out}$. Finally, single sample scores could be normalized either across samples or across gene signatures and samples.

The single sample GSEA method is also implemented in the R package GSVA (*Hänzelmann et al., 2013*).

Singscore:

For signatures whose genes are supposed to be up- or down-regulated, genes are ranked by expression value in increasing or decreasing order, respectively. For signatures whose direction is unknown, genes are ranked by absolute expression in increasing order and are median-centered. Enrichment scores are then computed for each sample and signature by averaging gene ranks for genes in the signature. Finally, normalized scores are obtained by subtracting the theoretical minimum mean rank from the score and dividing by the difference between the theoretical maximum and minimum mean ranks.

The hacksig implementation of this method works only with unidirectional (i.e. all genes up- or down-regulated) and undirected gene signatures. If you want to get single sample scores for bidirectional gene signatures (i.e. signatures composed of both up- and down-regulated genes), please use the R package singscore (*Foroutan et al., 2018*).

References

Barbie, D. A., Tamayo, P., Boehm, J. S., Kim, S. Y., Moody, S. E., Dunn, I. F., Schinzel, A. C., Sandy, P., Meylan, E., Scholl, C., Fröhling, S., Chan, E. M., Sos, M. L., Michel, K., Mermel, C., Silver, S. J., Weir, B. A., Reiling, J. H., Sheng, Q., Gupta, P. B., . . . Hahn, W. C. (2009). Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. *Nature*, 462(7269), 108–112. doi: 10.1038/nature08460.

Foroutan, M., Bhuva, D. D., Lyu, R., Horan, K., Cursons, J., & Davis, M. J. (2018). Single sample scoring of molecular phenotypes. *BMC bioinformatics*, 19(1), 404. doi: 10.1186/s1285901824354.

Hänzelmann, S., Castelo, R., & Guinney, J. (2013). GSVA: gene set variation analysis for microarray and RNA-seq data. *BMC bioinformatics*, 14, 7. doi: 10.1186/14712105147.

Lee, E., Chuang, H. Y., Kim, J. W., Ideker, T., & Lee, D. (2008). Inferring pathway activity toward precise disease classification. *PLoS computational biology*, 4(11), e1000217. doi: 10.1371/journal.pcbi.1000217.

See Also

`get_sig_info()` to get information about all implemented signatures.

`check_sig()` to check if signatures are applicable to your data.

`hack_cinsarc()` to apply the original CINSARC procedure.

`hack_estimate()` to obtain the original ESTIMATE scores.

`hack_immunophenoscore()` to apply the original Immunophenoscore procedure.

Examples

```
# Raw ssGSEA scores for all implemented signatures can be obtained with:
hack_sig(test_expr, method = "ssgsea")

# To obtain 0-1 normalized ssGSEA scores, use:
hack_sig(test_expr, method = "ssgsea", sample_norm = "separate")

# You can also change the exponent of the ssGSEA running sum with:
hack_sig(test_expr, method = "ssgsea", sample_norm = "separate", alpha = 0.5)

# To obtain combined z-scores for custom gene signatures, use:
custom_list <- list(rand_sig1 = rownames(test_expr)[1:5],
                    rand_sig2 = c(rownames(test_expr)[6:8], "RANDOMGENE"))
hack_sig(test_expr, custom_list, method = "zscore")
```

`test_expr` *A toy gene expression matrix*

Description

A gene expression matrix simulating expression profiles of 20 samples. It should be used only for testing purpose.

Usage

```
test_expr
```

Format

A random normal data matrix with 20000 genes as rows and 20 samples as columns.

Examples

```
class(test_expr)
dim(test_expr)
check_sig(test_expr)
```

Index

∗ **datasets**
    test_expr, 13

check_sig, 2
check_sig(), *4*, *10*, *13*

get_sig_genes, 3
get_sig_genes(), *4*
get_sig_info, 3
get_sig_info(), *2, 3*, *13*

hack_cinsarc, 4
hack_cinsarc(), *13*
hack_class, 5
hack_estimate, 6
hack_estimate(), *13*
hack_immunophenoscore, 8
hack_immunophenoscore(), *13*
hack_sig, 10
hack_sig(), *2*, *4–6*, *10*

test_expr, 13