SOFTWARE TOOL ARTICLE

Using singscore to predict mutations in acute myeloid leukemia from transcriptomic signatures [version 1; peer review: awaiting peer review]

Dharmesh D. Bhuva1,2, Momeneh Foroutan3, Yi Xie1, Ruqian Lyu1, Joseph Cursons1,4, Melissa J. Davis1,4,5

1Bioinformatics Division, Walter and Eliza Hall Institute of Medical Research, Parkville, VIC, 3052, Australia
2School of Mathematics and Statistics, University of Melbourne, Parkville, VIC, 3010, Australia
3Department of Clinical Pathology, The University of Melbourne Centre for Cancer Research, Victorian Comprehensive Cancer Centre, Parkville, VIC, 3000, Australia
4Department of Medical Biology, University of Melbourne, Parkville, VIC, 3010, Australia
5Department of Biochemistry and Molecular Biology, Faculty of Medicine, Dentistry and Health Sciences, University of Melbourne, Parkville, VIC, 3010, Australia

Abstract
Advances in RNA sequencing (RNA-seq) technologies that measure the transcriptome of biological samples have revolutionised our ability to understand transcriptional regulatory programs that underpin diseases such as cancer. We recently published singscore - a single sample, rank-based gene set scoring method which quantifies how concordant the transcriptional profile of individual samples are relative to specific gene sets of interest. Here we demonstrate the application of singscore to investigate transcriptional profiles associated with specific mutations or genetic lesions in acute myeloid leukemia. Using matched genomic and transcriptomic data available through the TCGA we show that scoring of appropriate signatures can distinguish samples with corresponding mutations, reflecting the ability of these mutations to drive aberrant transcriptional programs involved in leukemogenesis. We believe the singscore method is particularly useful for studying heterogeneity within a specific subsets of cancers, and as demonstrated, we show the ability of singscore to identify where alternative mutations appear to drive similar transcriptional programs.

Keywords
single sample, gene set scoring, signature scoring, AML mutations, NPM1c mutation, mutation prediction, TCGA

This article is included in the Bioconductor gateway.
Corresponding authors: Joseph Cursons (cursons.j@wehi.edu.au), Melissa J. Davis (davis.m@wehi.edu.au)

Author roles: Bhuva DD: Conceptualization, Formal Analysis, Software, Writing – Original Draft Preparation, Writing – Review & Editing; Foroutan M: Conceptualization, Writing – Review & Editing; Xie Y: Writing – Review & Editing; Lyu R: Writing – Review & Editing; Cursons J: Conceptualization, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing; Davis MJ: Conceptualization, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work was partially supported by the National Health and Medical Research Council (NHMRC) Project Grant APP1128609. MJD was supported by the NBCF Career Development Fellowship ECF-14-043 and the Betty Smyth Centenary Fellowship in Bioinformatics.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2019 Bhuva DD et al. This is an open access article distributed under the terms of the Creative Commons Attribution Licence, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Bhuva DD, Foroutan M, Xie Y et al. Using signscore to predict mutations in acute myeloid leukemia from transcriptomic signatures [version 1; peer review: awaiting peer review] F1000Research 2019, 8:776 (https://doi.org/10.12688/f1000research.19236.1)

First published: 03 Jun 2019, 8:776 (https://doi.org/10.12688/f1000research.19236.1)
Introduction

The development of microarrays and more recently the rapid uptake of RNA-sequencing technologies have provided a platform to examine the transcriptional profile (or transcriptome) of biological samples. Transcriptomic analyses have traditionally focused on ‘differential expression’ of genes between sets of samples, however with the rapid growth of publicly available RNA data there has been increasing usage of ‘relative approaches’, which quantify the relative concordance of a sample or samples with a specific gene signature. While sequencing of genomic mutations has been important for classifying different tumour subsets based upon the presence of mutations or fusion genes, and identifying genetic lesions which may act as drivers of cancer progression, transcriptomic profiling can provide further information on the state or phenotype of cells carrying these mutations.

Cancers are a heterogeneous set of diseases with a number of clinical and pathological subtypes. In diseases such as breast cancer the primary clinical classifications relate to the expression of hormone receptors (estrogen receptor: ER; and progesterone receptor: PR) or the overexpression of Erb-B2 receptor tyrosine kinase (HER2), as these features can be directly targeted with therapeutic agents. A common example of transcriptomic or gene expression data informing clinical practice is the use of prediction analysis of microarray 50 (PAM50) signatures for distinguishing the intrinsic breast cancer subtypes [2, Cieslik and Chinnaian]. For many other cancers, subtype classification has largely relied upon identifying sets of recurrent mutations across large patient cohorts, with whole genome or whole exome sequencing studies helping to resolve the clinically significant subtypes [3, Papaemmanuil et al.].

Perhaps the most well-known ‘relative approach’ is single-sample gene set enrichment analysis (ssGSEA), often used through the GenePattern web-tool. Another common approach is gene set variation analysis (GSVA) which is available as an R/Bioconductor package that also includes functionality for ssGSEA, an alternative approach known as PLAGE, and a z-score based approach. Both ssGSEA and GSVA use a Kolmogorov-Smirnov like random-walk statistic to convert normalised gene ranks to the resulting score; however, this normalisation procedure means that the scores are not truly ‘single-sample’, and variations in the overall sample composition for a study (e.g. variations in the presence or relative frequency of different cancer subtypes) can lead to unexpected changes in sample scores. Furthermore, the resultant scores from these methods can vary in their range and absolute value, making them difficult to interpret without further processing. To overcome this, we have developed a single-sample gene set scoring method singscore, which simply uses the ranks of genes within a given set, normalised relative to the maximum and minimum theoretical scores for a gene set of a given size.

Through large-scale efforts such as The Cancer Genome Atlas (TCGA), transcriptomic data are available for thousands of clinical samples, often together with corresponding genomic or epigenomic (often DNA methylation) data. These transcriptomic data can help to characterise the functional effects of corresponding mutations, and provide a window to study the heterogeneity which arises within different subtypes of cancer due to genomic and transcriptional regulatory programs which can also influence cell behaviour. Here, we demonstrate that the single-sample gene set scoring method singscore can be used to classify TCGA AML samples using transcriptional ‘gene signatures’ for the NPM1c mutation, KMT2A (MLL) gene fusions, and PML-RARA gene fusions that were derived from independent studies. Without any need for parameter fitting or estimation, we show that gene set scoring with singscore can distinguish samples carrying these mutations. The case studies we present demonstrate the application of gene set scoring to examine not only the differences, but also the relative similarities between established subtypes of AML that impact clinical outcome. This workflow is available as a bioconductor workflow package from https://bioconductor.org/packages/release/workflows/html/SingscoreAMLMutations.html.

Description of the biological problem

As with most cancers, acute myeloid leukemia (AML) is a heterogeneous disease with a number of classified subtypes. Analysis of TCGA AML genomic data identified a number of subtypes based upon the presence or absence of specific ‘driver mutations’; recapitulating and expanding upon previously identified clinical subsets. A more recent study which focused primarily on genomic data has further refined the clinically significant AML subtypes, highlighting a number of co-occurring as well as mutually exclusive mutations.

Of note for this work, one of the most common mutations in clinical AML samples is a frameshift mutation within exon 12 of the nucleophosmin (NPM1) gene. This mutation leads to aberrant localisation of nucleophosmin with cytoplasmic accumulation rather than localising to the nucleolus, and accordingly this mutation is often referred to as the NPM1c mutation [4]. As noted by Verhaak et al., the NPM1c mutation is associated with dysregulated activity of the homeobox domain (Hox) family of transcription factors which are essential for
developmental patterning. The effects of this mutation in disease progression have been further demonstrated in recent work which showed that loss of NPM1c leads to differentiation of AML cells.

Further recurrent genetic lesions in AML relevant for this work include lysine methyl transferase 2A (KMT2A; previously known as MLL) fusion genes, partial tandem duplications within KMT2A (KMT2A-PTD), and fusion genes between promyelocytic leukemia and retinoic acid receptor alpha (PML-RARA). Given the role of NPM1c in dysregulating the Hox gene family, it is interesting to note that AML samples with MLL fusion genes also show dysregulated expression of Hox family genes [13, Ross et al.]; however, samples with MLL-PTD appear to show a relatively distinct phenotype from MLL-fusion samples. While there is good evidence demonstrating the role of NPM1c mutations and other genetic lesions in blocking AML cell differentiation, the PML-RARA fusion subset is diagnostic for a specific subset of AML known as acute promyelocytic leukemia (APL). This clinically distinct subtype of AML is associated with a specific morphology under the French-American-British (FAB) classification of AML, FAB-M3, with cells showing a distinct morphology due to a differentiation block at the promyelocyte stage.

In this workflow we demonstrate the ability of the singscore method for single sample gene set scoring to classify tumour ‘driver mutations’ from transcriptomic data. We use a previously identified gene signature for the NPM1c mutation. We also use signatures for PML-RARA gene fusions and MLL-fusions that were derived using pediatric AML samples but shown to work well for classifying adult AML samples with similar lesions; although we note that there is evidence of relatively large differences in the mutational profiles of adult and pediatric cancers. Using these signatures, which are included within the molecular signatures database (MSigDB), we demonstrate that a bi-directional scoring approach can classify TCGA AML samples with corresponding mutations with a good precision and recall. A particularly useful feature of gene set scoring is the ability to project samples onto 2D or higher-order landscapes defined by corresponding phenotypic signatures. Accordingly, by comparing scores for both the NPM1c and KMT2A-MLL-fusion signatures, we show that this classification likely arises through the shared downstream biological effects of Hox family dysregulation. We also compare the NPM1c mutation signature to the PML-RARA signature and show a clear separation of these subtypes reflecting their divergent phenotypes and the mutually exclusive nature of these mutations.

**Downloading and preparing the data**

Data from TCGA project is made available through the Genomic Data Commons (GDC). Open access data from the project can be accessed in multiple pre-processed formats. Transcriptomic data can be downloaded either at the count level or as FPKM transformed abundance, before or after upper quantile normalisation. Other pre-processed version can be found from sources such as the cBioPortal and FireBrowse. The GDC data used STAR to perform a two-pass alignment followed by quantification using HTSeq. Data from the GDC can be downloaded using the GDC data transfer tool which allows users to select the specific files of interest using the GDC portal. These files then have to be read, merged, annotated and processed into a data structure that simplifies downstream analysis. Alternatively, all the above mentioned steps, including the download, can be performed using the R package TCGAbiolinks. The package supports data download using the GDC API and the GDC client. We will use the TCGAbiolinks package to download, annotate and process the data into a SummarizedExperiment R object.

The following steps need to be performed to prepare the data:

1. Create a query to select the files to download
2. Execute the query and download the data
3. Read the data into R
4. Filter out genes with low expression
5. Normalise the data for compositional bias and transform to account for gene-length biases as outlined in the singscore manuscript

**Querying the GDC database**

The first step in any analysis should be to determine and report the data version and the service used to download the data. The getGDCInfo() function returns the release date of all data on the GDC along with a version.
library(SingscoreAMLMutations)
library(TCGAbiolinks)

# get GDC version information
gdc_info = getGDCInfo()
gdc_info

## $commit
## [1] "e588f035feefee17f562b3a1bc2816c49a2b2b19"
##
## $data_release
## [1] "Data Release 16.0 - March 26, 2019"
##
## $status
## [1] "OK"
##
## $tag
## [1] "1.20.0"
##
## $version
## [1] 1

A query then needs to be run, using the GDC to identify the specific files for download. This step is similar to generating a MANIFEST file using the GDC portal. The first parameter of the query specifies the project - available projects can be accessed using getGDCprojects() or from https://portal.gdc.cancer.gov/projects.

The TCGA acute myeloid leukemia data is part of the TCGA-LAML project. Following this, the data category, data type and workflow type need to be specified. The query formed below selects files containing the count level transcriptomic measurements. Values for different parameters of the query can be identified from “searching arguments” section of the “query” vignette: vignette("query", package = "TCGAbiolinks"). The result of this query will be a dataframe containing filenames and additional annotations related to the files.

Read count level data are selected instead of the processed FPKM data as one of the downstream pre-processing analysis results in filtering out of genes. A general recommendation is to compute FPKM values after filtering genes out so as to ensure counts are normalised by the corresponding library sizes. In cases where count-level data is not available, filtering can be performed directly on FPKM values, provided that the library size is large enough.

# form a query for the RNAseq data
query_rna = GDCquery(
  #getGDCprojects()
  project = 'TCGA-LAML',
  #TCGAbiolinks:::getProjectSummary('TCGA-LAML')
  data.category = 'Transcriptome Profiling',
  data.type = 'Gene Expression Quantification',
  workflow.type = 'HTSeq - Counts'
)

# extract results of the query
rnaseq_res = getResults(query_rna)
dim(rnaseq_res)

## [1] 151 28

colnames(rnaseq_res)

## [1] "data_release"      "data_type"
## [3] "updated_datetime"   "file_name"
## [5] "submitter_id"       "file_id"
## [7] "file_size"                 "cases"
## [9] "id"                        "created_datetime"
## [11] "md5sum"                    "data_format"
## [13] "access"                    "state"
## [15] "version"                   "data_category"
## [17] "type"                      "experimental_strategy"
## [19] "project"                   "analysis_id"
## [21] "analysis_updated_datetime" "analysis_created_datetime"
## [23] "analysis_submitter_id"     "analysis_state"
## [25] "analysis_workflow_link"    "analysis_workflow_type"
## [27] "analysis_workflow_version" "tissue.definition"

### Downloading the TCGA AML RNA-seq read counts
The `GDCdownload` function then executes the query on the GDC database and begins downloading the data using the GDC API. The download method should be changed to “client”, if the size of the data is expected to be large, e.g. for RNA-seq read data or methylation data. It is good practice to specify the directory for data storage - we store all the data in the “GDCdata” directory in the temporary directory. Users should store their data in a permanent storage to retain the data. The function downloads the data and organises it into the folder based on parameters specified in the query. This allows multiple different levels and types of data to be stored in the same directory structure. Files with counts are stored at `TEMPDIR/GDCdata/TCGA-LAML/harmonized/Transcriptome_Profiling/Gene_Expression_Quantification/`.

```r
datapath = file.path(tempdir(), 'GDCdata')
GDCdownload(query_rna, directory = datapath) #{size: 39MB}
```

### Reading count-level data into R
The `GDCprepare` function reads and processes the downloaded data into a `RangedSummarizedExperiment` object from the `SummarizedExperiment` package which allows patient annotations, gene annotations and count data to be stored in one object. Patient annotations are downloaded upon calling this function and subsequently mapped and attached to the resulting object. A `RangedSummarizedExperiment` object is similar to an `ExpressionSet` object but provides added functionality such as indexing with genomic coordinates and storing multiple data matrices with the same structure. Feature annotations used to annotate the data are stored in an RDA/RDATA file.

```r
aml_se = GDCprepare(query_rna, directory = datapath)
```

The object contains data for 56,925 features and 151 samples. The original data files contain 60,483 features, some of which (3,767) could not be mapped to ENSEMBL GRCh38.p12. Feature and sample annotations can be accessed using `rowData(se)` and `colData(se)`, respectively, and the counts data can be accessed using `assay(se)`. TCGA data usually contains some formalin-fixed paraffin-embedded (FFPE) samples which should be discarded from the analysis as the protocol introduces biological artefacts. This procedure is only performed on solid tumours and not leukemias, therefore, no filtering is required for this data set.

```r
aml_se
```

```r
## class: RangedSummarizedExperiment
## dim: 56734 151
## metadata(1): data_release
## assays(1): HTSeq - Counts
## rownames(56734): ENSG00000000003 ENSG00000000005 ... ENSG0000000281912 ENSG00000281920
## rowData names(3): ensembl_gene_id external_gene_name original_ensembl_gene_id
## colData names(56): sample patient ... name is_ffpe
```
Filter out genes with low counts

The edgeR package contains methods that assist in the data normalisation and transformation required for filtering and subsequent steps. The methods require a DGEList object therefore we begin by creating a DGEList for the AML data from the SummarizedExperiment. Creation of a DGEList object is initially unsuccessful due to duplicated row names in the data matrix (107 features). The raw count files for individual samples do not have any duplicated features indicating this is introduced by the GDCprepare function. This is further supported by the fact that the counts for a single feature are the same for duplicated entries. An example of such a feature is shown below and we have verified duplicated entries of all other features to be the same.

```r
library(SummarizedExperiment)
#example check on one feature ID
dup_feature = 'ENSG00000011454'
dup_data = assay(aml_se[rownames(aml_se) %in% dup_feature, ])
identical(dup_data[1, ], dup_data[2, ])
```

## [1] TRUE

As such, it is safe to discard duplicated entries.

```r
library(edgeR)
aml_se = aml_se[!duplicated(rownames(aml_se)), ]
aml_dge = DGEList(counts = assay(aml_se), genes = rowData(aml_se))
```

Genes with low counts across most samples are discarded from the analysis. This is a standard step in differential expression analysis as inclusion of such genes in the analysis could skew estimates of dispersion. It is also motivated in rank-based analysis, such as with singscore, to avoid rank duplication. Rank duplication reduces the discriminant power of scores as the number of unique ranks is reduced. A commonly used filter is to select only those genes that have CPMs above a certain threshold across a proportion of samples. Filtering is performed on the CPMs rather than raw counts as the former accounts for variation in library sizes, therefore, is unbiased. For instance, a CPM of 1 would equate to read counts between 19 and 50 for samples in the AML data where library sizes vary between 18.6 and 49.7 million reads. Here, we retain genes that have a CPM > 1 across more than 50% of the samples. Other methods to filter out genes with low counts exist and may be preferable in specific applications. Chen et al.\textsuperscript{19} and Law et al.\textsuperscript{20} filter genes based on the experimental design whereby the proportion of samples with enough read counts are evaluated per experimental group. As the AML data have many samples, filtering is performed across all samples rather than within sub-groups. Group specific filtering would be recommended for the study of rare groups. The distribution of logCPMs is much closer to the expected log-normal distribution after filtering out genes with low counts as seen in Figure 1.

```r
prop_expressed = rowMeans(cpm(aml_dge) > 1)
keep = prop_expressed > 0.5
op = par(no.readonly = TRUE)
par(mfrow = c(1, 2))
hist(cpm(aml_dge, log = TRUE), main = 'Unfiltered', xlab = 'logCPM')
abline(v = log(1), lty = 2, col = 2)
hist(cpm(aml_dge[keep, ], log = TRUE), main = 'Filtered', xlab = 'logCPM')
abline(v = log(1), lty = 2, col = 2)
par(op)
```

#subset the data
aml_dge = aml_dge[keep, , keep.lib.sizes = FALSE]
aml_se = aml_se[keep, ]
Transformation to FPKM values and normalisation

Singscore requires gene expression measurements to be comparable between genes within a sample, therefore, correction for gene length bias needs to be performed. Transformations such as transcripts per million (TPM) and reads/fragments per kilobase per million (RPKM/FPKM), that normalise by gene length, may be used. Both TPM and RPKM/FPKM values should produce similar results when applying singscore provided that the library size is large enough, which they are here. RPKM values are generally computed after correcting for compositional biases. The `calcNormFactors` function in edgeR provides three methods to do so, TMM normalisation being the default. Chen et al. and Law et al. discuss the implications of normalisation prior to down-stream processing such as differential expression analysis. Normalisation is generally performed for cross-sample analysis where samples need to be comparable. Singscores are invariant to data normalisation as the analysis is contained within the sample of interest. The idea extends to any transformation that preserves the relative ranks of genes within a sample such as a log transformation. Here, we use TMM normalisation solely for visualisation purposes.

Data transformation to TPM or RPKM/FPKM requires the lengths for all genes to be calculated. Gene lengths need to be computed based on the alignment and quantification parameters. The TCGA transcriptomic data has been aligned using STAR and quantified using HTSeq (details of the pipeline available at https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline/). HTSeq quantifies reads mapping to the exons of each gene, therefore, effective gene lengths can be calculated as the sum of all exons spanning the gene. The GENCODE v22 annotation file was used for quantification therefore the same file needs to be used to compute gene lengths.

```r
#download v22 of the GENCODE annotation
gencode_file = 'gencode.v22.annotation.gtf.gz'
gencode_link = paste('ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_22', gencode_file, sep = '/')
download.file(gencode_link, gencode_file, method = 'libcurl') #(size: 39MB)
```

The `rtracklayer` R package provides functions to help parse GTF files.

```r
library(rtracklayer)
library(plyr)

# download v22 of the GENCODE annotation
# (size: 39MB)

gtf = import.gtf(gencode_file, format = 'gtf', genome = 'GRCm38.71', feature.type = 'exon')
#split records by gene to group exons of the same gene
grl = reduce(split(gtf, elementMetadata(gtf)$gene_id))
gene_lengths = ldply(grl, function(x) {
  # sum up the length of individual exons
  return(c('gene_length' = sum(width(x))))
}, .id = 'ensembl_gene_id')
```

---

**Figure 1.** Histogram of logCPM values for the AML data before and after filtering. Filtering results in fewer zeros in the data. Most genes with CPM less than 1, logCPM < 0; (red line) across the majority of samples get discarded, resulting in an approximately log-normal distribution.
Genes are also annotated with their biotype for further analysis. The annotation file uses Ensembl IDs with versions as keys to records, which then need to be converted to Ensembl IDs. This is simply achieved by truncating the trailing version number.

```r
# extract information on gene biotype
genotype = unique(elementMetadata(gtf)[, c('gene_id', 'gene_type')])
colnames(genotype)[1] = 'ensembl_gene_id'
gene_lengths = merge(genotype, gene_lengths)

# remove ENSEMBL ID version numbers
gene_lengths$ensembl_gene_id = gsub('\.[0-9]*', '', gene_lengths$ensembl_gene_id)
saveRDS(gene_lengths, file = 'gene_lengths_HTSeq_gencodev22.rds')
gene_lengths
```

## DataFrame with 60483 rows and 3 columns
##
## | ensembl_gene_id | gene_type       | gene_length |
## |-----------------|-----------------|-------------|
## | ENSG00000000003  | protein_coding | 4535        |
## | ENSG00000000005  | protein_coding | 1610        |
## | ENSG000000000419 | protein_coding | 1207        |
## | ENSG000000000457 | protein_coding | 6883        |
## | ENSG000000000460 | protein_coding | 5967        |
## | ENSG000000000472 | misc_RNA       | 290         |
## | ENSG000000000473 | miRNA          | 68          |
## | ENSG000000000474 | miRNA          | 64          |
## | ENSG000000000475 | lincRNA        | 515         |
## | ENSG000000000476 | antisense      | 484         |

The SummarizedExperiment object allows feature annotations to be stored, therefore, information on gene length and biotypes should be added to the existing annotations. Similarly, annotations need to be added to the DGEList object. The column containing lengths should include “length” in its name.

```r
# allocate rownames for ease of indexing
rownames(gene_lengths) = gene_lengths$ensembl_gene_id
rowData(aml_se)$gene_length = gene_lengths[rownames(aml_se), 'gene_length']
rowData(aml_se)$gene_biotype = gene_lengths[rownames(aml_se), 'gene_type']

# annotate gene lengths for the DGE object
aml_dge$genes:length = gene_lengths[rownames(aml_dge), 'gene_length']
```

RPKM/FPKM values can now be calculated with the computed gene lengths after computing the normalisation factors. The SummarizedExperiment object can store multiple levels of the data simultaneously, provided that the number of features and samples remains the same across measurements. As such, FPKM values are appended to the existing object.

```r
aml_dge_tmm = calcNormFactors(aml_dge, method = 'TMM')

# compute FPKM values and append to assays
assay(aml_se, 'logFPKM_TMM') = rpkm(aml_dge_tmm, log = TRUE)
aml_se
```

## class: RangedSummarizedExperiment
## dim: 17425 151
## metadata(1): data_release
## assays(2): HTSeq - Counts logFPKM_TMM
## rownames(17425): ENSG00000000419 ENSG000000000457 ...
Annotate samples with mutation data

For this analysis we have used the curated mutation list from the original TCGA AML publication \(^3\) (Supplemental Table 01 at [https://gdc.cancer.gov/node/87]) rather than variant calls from the standard TCGA pipeline (available through the National Cancer Institute Genomic Data Common) and readers should note that there are some discrepancies between these. For genetic lesions of interest (NPM1c, \(KMT2A-MLL\), \(KMT2A-PTD\) and \(PML-RARA\)), patients were identified by the following criteria:

- **Patient ID**: The ‘TCGA Patient ID’ column was extracted directly
- **NPM1c**: TRUE if the ‘NPM1’ column contains the strings ‘p.W287fs’ or ‘p.W288fs’
- **KMT2A-fusion**: TRUE if the ‘MLL-partner’ column contains the string ‘MLL-’ or ‘-MLL’ (note that the official gene symbol for \(MLL\) is now \(KMT2A\))
- **KMT2A-PTD**: TRUE if the ‘MLL-PTD’ column contains the string ‘exons’
- **PML-RARA**: TRUE if the ‘PML-RARA’ column contains the string ‘PML-RARA’

```r
# DataFrame with 151 rows and 4 columns
#              NPM1c_Mut KMT2A_Fusion KMT2A_PTD  PML_RARA
# TCGA-AB-2808-03A-01T-0734-13    FALSE    FALSE    FALSE    FALSE
# TCGA-AB-2808-03A-01T-0734-13    FALSE    FALSE    FALSE    FALSE
# TCGA-AB-2834-03A-01T-0734-13    FALSE    TRUE    FALSE    FALSE
# TCGA-AB-2834-03A-01T-0734-13    FALSE    TRUE    FALSE    FALSE
# ...        ...        ...        ...        ...
# TCGA-AB-2821-03A-01T-0735-13    FALSE    FALSE    FALSE    FALSE
# TCGA-AB-2861-03A-01T-0736-13    TRUE    FALSE    FALSE    FALSE
# TCGA-AB-2835-03A-01T-0736-13    TRUE    FALSE    FALSE    FALSE
# TCGA-AB-2818-03A-01T-0734-13    TRUE    FALSE    FALSE    FALSE
# TCGA-AB-2869-03A-01T-0735-13    TRUE    FALSE    FALSE    FALSE
```

Map Ensembl IDs to Entrez IDs

Ensembl annotations (Ensembl IDs) have higher coverage of the genome which may be useful in applications such as variant calling and similar exploratory analysis\(^22\). However, RefSeq annotations (Entrez IDs) may be better suited to RNA-seq analyses which require a stable reference annotation for comparison\(^23\). As such, we choose to map Ensembl IDs to Entrez IDs and discard any unmapped features.

Mapping can be performed using the Ensembl Biomart service, which can be queried using the biomartR bioconductor package. This would provide the most up to date annotations. Alternatively, mapping could be performed with the bi-annually updated org.Hs.eg.db annotation R package which provides a stable set of annotations, thereby enhancing reproducibility. Mapping is performed with the mapIds function in the AnnotationDbi R package.
Multimapped Ensembl IDs are replaced by \( \text{NAs} \), then discarded to enforce unique mapping. Similarly, Entrez IDs that map to multiple Ensembl IDs are identified from the mapping, and discarded. Only features with unique Ensembl ID to Entrez ID mappings remain.

```r
#select genes with mapped Entrez IDs
keep = !is.na(gene_annot$entrezgene)

#select genes with unique Entrez IDs
dup_entrez = gene_annot$entrezgene[duplicated(gene_annot$entrezgene)]
keep = keep & !gene_annot$entrezgene %in% dup_entrez

#Biotype of discarded genes (due to non-unique mapping)
head(sort(table(gene_annot[!keep, 'gene_biotype'])), decreasing = TRUE, n = 10)

##
##         processed_pseudogene                          antisense
##                         1017                                907
##                      lincRNA                                TEC
##                          451                                263
##             sense_intronic                     protein_coding
##                          252                                139
##       unprocessed_pseudogene transcribed_unprocessed_pseudogene
##                          128                                 93
##         processed_transcript   transcribed_processed_pseudogene
##                           72                                 72
```

Transcriptional signatures to predict mutation status
The signature by Verhaak et al.\textsuperscript{12} is now used to predict the mutation status of the NPM1c mutation. This is done by quantifying the concordance of genes in the signature with their expression in each sample. As such, high expression of up-regulated genes and low expression of down-regulated genes would result in higher scores. This single value can then be used to predict the mutation status of individual samples if these data were unavailable.

The signatures of interest are first downloaded from the MSigDB and read into GeneSet objects from the GSEABase R package. We then use the singscore R/Bioconductor package to quantify each sample for the Verhaak signature. Some of the visualisation and diagnostic tools within the singscore package are used to interpret the signatures and scores. Finally, we use a simple logistic regression model on the scores to predict the mutation status.

Download signature and load into R
The Verhaak et al.\textsuperscript{12} signature is composed of an up-regulated and a down-regulated gene set. Many signatures are developed in such a manner to improve discrimination of samples. MSigDB stores such signatures using names with suffixes “\_UP” and “\_DN” representing the independent components of the signature. Here, we form the download links for the signature with the base name “VERHAAK\_AML\_WITH\_NPM1\_MUTATED”.
#create signature names

```r
verhaak_names = paste('VERHAAK_AML_WITH_NPM1_MUTATED', c('UP', 'DN'), sep = '_')
verhaak_names
```

## [1] "VERHAAK_AML_WITH_NPM1_MUTATED_UP" "VERHAAK_AML_WITH_NPM1_MUTATED_DN"

The signatures are then downloaded using the links, resulting in an XML file for each component of the signature. The `mapply` function is used to run the download function on all pairs of link-output arguments.

```r
#generate URLs
verhaak_links = paste0(  
  'http://software.broadinstitute.org/gsea/msigdb/download_geneset.jsp?geneSetName=',  
  verhaak_names,  
  '&fileType=xml'
)
#download files
verhaak_files = paste0(verhaak_names, '.xml')
mapply(download.file, verhaak_links, verhaak_files, method = 'libcurl')
```

Functions in the GSEABase package help with reading, parsing and processing the signatures. Signatures from an MSigDB XML file can be read using the `getBroadSets` function which results in a `GeneSet` object. Gene symbols, Entrez IDs and affymetrix chip IDs from the original experiment (HG-U133A in this case) are stored in the XML file. Entrez IDs are read from the file as these can be mapped directly to our data. Conversions to other identifiers can be achieved using the `mapIdentifiers` function from GSEABase and an annotation package that contains the mappings. The advantage of using this function instead of the `mapIds` function from the AnnotationDbi package is that the former retains the GeneSet object after conversion of IDs.

```r
library(GSEABase)
verhaak_sigs = getBroadSets(verhaak_files, membersId = 'MEMBERS_EZID')
verhaak_sigs
```

## GeneSetCollection
## names: VERHAAK_AML_WITH_NPM1_MUTATED_UP, VERHAAK_AML_WITH_NPM1_MUTATED_DN (2 total)
## unique identifiers: 10051, 10135, ..., 9828 (429 total)
## types in collection:
## geneIdType: SymbolIdentifier (1 total)
## collectionType: BroadCollection (1 total)

To make data indexing easier during signature scoring, row names of the `SummarisedExperiment` object are changed to Entrez IDs which are already part of the row annotations.

```r
rownames(aml_se) = rowData(aml_se)$entrezgene
```

**Score TCGA AML samples using the Verhaak signature**

Singscore is a rank based metric of gene set enrichment in single samples. Scores for multiple signatures make use of the same ranked expression per sample. As such, it makes sense to compute the ranks only once and re-use them for scoring different signatures. The implementation separates these two phases of the analysis to reduce the computational cost of scoring. The `rankGenes` function will compute ranks from expression data in the form of either a numeric matrix, numeric data frame, ExpressionSet object, DGEList object or a SummarizedExperiment object. Users also have to specify what method should be used to break ties. The default is ‘min’ and we recommend this be used for RNA-seq data which may have many genes with zero counts. This will reduce the effect of zeros in the scores, however, appropriate pre-filtering of genes with low counts will still be required.
library(singscore)

# apply the rankGenes method to each version of the dataset, excluding counts
aml_ranked = rankGenes(assay(aml_se, 'logFPKM_TMM'))

Singscores can be computed using three modes, depending on the properties of the gene signature. The first mode of operation is applied when two directed gene sets (expected up- and down-regulated gene sets) form the transcriptomic signature. Many signatures in the MSigDB, including the Verhaak et al. signature come in such pairs. This mode can be invoked by passing the up- and down-regulated gene sets to the arguments upSet and downSet respectively. In some cases, only one set of genes forms the signature. If all genes in the gene set are up-regulated or all down-regulated, the second mode of operation applies and is invoked by passing the gene set to the upSet argument. For sets of down-regulated genes, the score would simply be inverted (-score if scores are centered, 1 - score otherwise). Finally, if the user is unsure of the composition of genes in the gene-set, such that, the gene set may contain both up- and down-regulated genes, the final mode of singscore applies. The gene set is passed to the upSet argument similar to the previous mode with the additional argument knownDirection set to FALSE.

By default, singscores are centered such that the range of scores is [-1, 1] and [-0.5, 0.5] for the first two modes respectively. Negative scores indicate an inverse enrichment of signatures, that is, expected up-regulated genes are in fact down-regulated and vice-versa. Scores from the last mode can not be centered and have the range [0, 1]. In this mode, high scores are obtained when ranks of genes are distant from the median rank. If scores are centered in this scenario, it would lead to the conclusion that a negative score shows inverted enrichment, which is not the case. Score centering only serves the purpose of easing interpretation for users, a simple linear transformation is applied to achieve it.

Scores for the NPM1c mutation signature are computed using the default settings, with the first mode of operation being used due to the presence of an up- and down-regulated gene set. The function returns a data frame reporting the score and dispersion of ranks for the up-regulated gene set, down-regulated gene set and the combination of both. Dispersion of the combined gene set in this mode is simply the mean of the independent dispersion estimates. If any gene names/IDs are present in the signature but missing in the expression data, a warning will be reported.

# apply the scoring function
verhaak_scores = simpleScore(aml_ranked,
upSet = verhaak_sigs[[1]],
downSet = verhaak_sigs[[2]])

## Warning in checkGenes(upSet, rownames(rankData)): 22 genes missing: 10265, 108, 10924, 11025, 11026, 1646, 1672, 200315, 2215, 3215, 3216, 3569, 3627, 3759, 50486, 6346, 6364, 861, 8843, 9103, 9518, 9627

## Warning in checkGenes(downSet, rownames(rankData)): 27 genes missing: 100505650, 10232, 10267, 2122, 221981, 2258, 23532, 23630, 24141, 25907, 2697, 28517, 28639, 3386, 3848, 3934, 4070, 445, 4680, 4681, 5457, 5790, 6091, 7441, 8277, 862, 8788

It should be noted that singscores are composed of two components, an enrichment score and a dispersion estimate of ranks. The quantity of interest in gene set enrichment is the distribution of the expression or ranks of genes in the signature. In an ideal scenario, all expected up-regulated genes would have high expression therefore higher values of ranks. As such, ranks would be distributed on the higher end of the entire rank spectrum. Singscore aims to quantify this distribution of ranks, therefore, computes and reports the average and dispersion of ranks of genes in the signature relative to all other genes. The first quantity is similar to scores computed from all other single sample scoring methods. We determined a two component score to be a more appropriate and accurate representation of the distribution of ranks of signature genes. The default and recommended measure of dispersion is the median absolute deviation (MAD) due to its non-parametric property. Other appropriate measure of dispersion could be the inter-quartile range (IQR) and can be used by passing the IQR function as an argument to the dispersionFun argument.
## head(verhaak_scores)

<table>
<thead>
<tr>
<th></th>
<th>TotalScore</th>
<th>TotalDispersion</th>
<th>UpScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCGA-AB-2808-03A-01T-0734-13</td>
<td>-0.03592658</td>
<td>5685.771</td>
<td>0.003475178</td>
</tr>
<tr>
<td>TCGA-AB-2865-03A-01T-0736-13</td>
<td>0.01467909</td>
<td>5129.055</td>
<td>0.115085222</td>
</tr>
<tr>
<td>TCGA-AB-2930-03A-01T-0740-13</td>
<td>-0.04123012</td>
<td>5281.762</td>
<td>-0.040130701</td>
</tr>
<tr>
<td>TCGA-AB-2844-03A-01T-0736-13</td>
<td>0.15265046</td>
<td>4872.565</td>
<td>0.051436521</td>
</tr>
<tr>
<td>TCGA-AB-2834-03A-01T-0734-13</td>
<td>0.13797648</td>
<td>5038.616</td>
<td>0.053861255</td>
</tr>
<tr>
<td>TCGA-AB-3011-03A-01T-0736-13</td>
<td>0.04738017</td>
<td>5310.673</td>
<td>-0.075273109</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>UpDispersion</th>
<th>DownScore</th>
<th>DownDispersion</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCGA-AB-2808-03A-01T-0734-13</td>
<td>5282.504</td>
<td>-0.039401756</td>
<td>6089.038</td>
</tr>
<tr>
<td>TCGA-AB-2865-03A-01T-0736-13</td>
<td>5058.631</td>
<td>-0.100406133</td>
<td>5199.478</td>
</tr>
<tr>
<td>TCGA-AB-2930-03A-01T-0740-13</td>
<td>4999.327</td>
<td>-0.001099423</td>
<td>5564.198</td>
</tr>
<tr>
<td>TCGA-AB-2844-03A-01T-0736-13</td>
<td>5239.508</td>
<td>0.101213941</td>
<td>4505.621</td>
</tr>
<tr>
<td>TCGA-AB-2834-03A-01T-0734-13</td>
<td>5700.597</td>
<td>0.084115222</td>
<td>4376.635</td>
</tr>
<tr>
<td>TCGA-AB-3011-03A-01T-0736-13</td>
<td>5782.140</td>
<td>0.122653278</td>
<td>4839.206</td>
</tr>
</tbody>
</table>

### Diagnostics of the Verhaak signature

The `singscore` package provides a set of visualisation tools that enable diagnostics of the gene signature. For instance, these tools may be used to determine the importance of each component for a bidirectional signature (up- and down-regulated gene sets) to the total score, determine the importance of each gene of a signature in discriminating between the classes of interest, and to investigate the relationship between the final score and the dispersion of signature gene ranks. Annotations of interest can be overlayed on each plot. Singscore supports both continuous and categorical annotations, which can either be input as a vector, or as a string specifying a column within the score data frame. We begin by investigating the relationship between the score and dispersion of ranks for the up-regulated gene signature, down-regulated gene signature and the full signature. The `plotDispersion` functions generates a diagnostic plot with annotations overlaid. Annotations can be discrete or continuous, and can be passed as independent variables, or as a column name when the data is appended to the score data frame. It should be noted that all plotting functions in `singscore` can be made interactive by setting the `isInteractive` argument to `TRUE`.

```r
# relative size of text in the figure
relSize = 0.7

# create annotation
mutated_gene = rep('Other', ncol(aml_se))
mutated_gene[aml_se$NPM1c.Mut] = 'NPM1c Mut'
mutated_gene[aml_se$KMT2A.Fusion | aml_se$KMT2A.PTD] = 'MLL Fusion/PTD'

p1 = plotDispersion(verhaak_scores, annot = mutated_gene, textSize = relSize)
p1
```

Figure 2 shows that the set of down-regulated genes has more discriminant power in separating NPM1c mutated samples from those without. Moreover, the signature is able to discriminate `MLL (KMT2A)` fusions and PTDs from the other samples. NPM1c mutations produce higher scores for the down-regulated gene set while `MLL` fusions and PTDs produce moderate scores. Similar trends are observed with the set of up-regulated genes; however, despite the range of scores increasing, the discriminant power drops moderately. In fact, the signature of up-regulated genes is able to discriminate samples without the genomic changes of interest (‘Other’) better by producing negative scores for most of these samples. Negative scores for the expected up-regulated gene set indicate that these genes are expressed below median expression, therefore, likely down-regulated within corresponding samples. Another observation from these plots is based on the trend of the dispersion (MAD). The dispersion is generally expected to be higher for scores close to zero. Zero scores are achieved in three possible scenarios: when genes are expressed at median expression level; when genes are evenly distributed at both ends of the expression spectrum; or, the more likely scenario whereby the rank of expression of all genes are uniformly distributed. The last two scenarios would result in a high dispersion. To explain these ideas, we select 3 samples and plot the ranks of genes in both signatures. The sample with the highest total score, lowest total score and highest dispersion are chosen.
library(gridExtra)

# select samples with the properties required
select_samples = c(
  'Max Total Score' = which.max(verhaak_scores$TotalScore),
  'Min Total Score' = which.min(verhaak_scores$TotalScore),
  'Max Dispersion' = which.max(verhaak_scores$TotalDispersion)
)

# plotRankDensity applied to each sample
rank_plots = lapply(names(select_samples), function(x) {
  # get the sample index
  aml_sample = select_samples[x]
  # drop = FALSE is required to ensure the data.frame is intact
  p1 = plotRankDensity(rankData = aml_ranked[, aml_sample, drop = FALSE],
                       upSet = verhaak_sigs[[1]],
                       downSet = verhaak_sigs[[2]],
                       textSize = relSize)
  # overwrite title of the plot with the description of the sample
  # this is possible because singscore uses ggplot2
  p1 = p1 + ggtitle(paste(x, mutated_gene[aml_sample], sep = '\n')) +
       guides(colour = guide_legend(ncol = 1))
  return(p1)
})

# create a multipanel plot
grid.arrange(grobs = rank_plots, nrow = 1)

Figure 3 shows that both the up- and down-regulated gene sets contribute to the score of the top scoring sample as these genes are at the extremes ends of the rank spectrum. Similarly, the lowest scoring sample has the ranks of genes in each set inverted. As observed in the previous plot, the up-regulated genes improve discrimina-
tion between NPM1c mutated samples and other samples and may be a stronger indicator of wild-type NPM1c samples than the down-regulated genes. Finally, the sample with the maximum dispersion exhibits uniformly distributed down-regulated gene ranks and a bimodal distribution of the up-regulated gene ranks with the modes at the extremes of the spectrum.

Combined, these plots show that both the up- and down-regulated genes play an important role in discriminating between NPM1c mutated (and MLL fusion/PTD samples), and wild-type samples. These diagnostic plots help in determining the importance of genes in signatures with respect to the samples of interest and should be used prior to application of signatures. In some cases, signatures may have been developed in specific contexts due to inherent biases in datasets and yet, described or applied in a generalised setting. These diagnostic plots may help in validating their application in specific contexts and possibly assist in characterising the contexts of all existing signatures.

Predicting mutations using the Verhaak signature

Mutation status can be predicted from singescores using a logistic regression model with a “logit” link function. The benefits of this model over one where each gene in the signature is used as a term in the model is the simplicity of the model. The Verhaak et al.\textsuperscript{12} signature consists of 429 genes which would result in a regression model with 429 predictors. As there are fewer samples than the predictors, some feature selection method would be required which may result in the loss of information. Moreover, automating model development on a larger selection of gene signatures would be limited with gene-level models. Additionally, models trained using singe-score would inherit its non-parametric properties to some extent. For instance, such models would be invariant to all data transformations that retain within sample ranks of genes. The main limitation of such models would be the loss of accuracy due to summarisation of the data (information on 429 genes is summarised into a two-component score).

In any case, our aim here is not to discuss the various models that can be used to predict mutation status, but to exhibit the discriminant power of singescore and transcriptomic signatures. consequently, the data used to train the model are used to evaluate the basic performance properties. We begin by combining the scores with mutation annotations.

Figure 3. Rank distribution of genes reveals that expected up-regulated genes are up-regulated in high scoring samples and down-regulated in low-scoring samples with a similar, but inverse, observation for the expected down-regulated genes. Samples with the highest score, lowest score and highest dispersion (MAD) are show from left to right. The barcode plot for ranks of genes in each signature along with the density of the ranks is shown in each plot. Up-regulated genes are represented in green and down-regulated genes in purple. Gaussian distributions are observed at the extremes for the lowest and highest scores. High dispersion results in scores close to 0 because all genes are either evenly distributed (down-regulated gene set) or bi-modally distributed at the extrema (up-regulated gene set).
# create a dataframe with the data required: scores and mutations
scoredf = as.data.frame(colData(aml_se)[, aml_mutations])
scoredf$Score = verhaak_scores$TotalScore
scoredf$Dispersion = verhaak_scores$TotalDispersion

Before training a model, we can visualise how well scores resulting from the Verhaak et al.\cite{12} signature separates the mutants from wild-type samples for some genes of interest. Figure 4 shows that NPM1c signature scores can discriminate between NPM1c wild-type and mutants. Similar observations are made for MLL (KMT2A) fusions and PML-RARA fusions.

# restructure the data for plotting
plotdf = melt(scoredf, id.var = c('Score', 'Dispersion'), variable.name = 'Mutation', value.name = 'Status')
# convert TRUE-FALSE values to Mut-WT
plotdf$Status = factor(plotdf$Status, labels = c('WT', 'Mut'))
pl = ggplot(plotdf, aes(Mutation, Score, fill = Status)) + geom_boxplot(position = 'dodge', alpha = 0.6) + scale_fill_brewer(palette = 'Set2') + current_theme + theme(axis.text.x = element_text(angle = 45, vjust = 0.5))
pl

To quantify the above observations, we fit logistic regression models with each genomic lesion as the response variable and the score as the predictor. Coefficient estimates, standard errors, z-statistics, and p-values of each model are listed below.

Figure 4. NPM1c signature scores are able to separate mutants from wild-types. Boxplot of scores from the NPM1c signature split by different types of mutations.
#fit GLMs to each mutation
glms = lapply(aml_mutations, function(x) {
    #generate a formula for the fit
    form = as.formula(paste0(x, ' ~ Score'))
    glm1 = glm(form, data = scoredf, family = binomial(link = 'logit'))
    return(glm1)
})
names(glms) = aml_mutations

#extract coefficients
coops = lapply(glms, function(x) coef(summary(x)))
ldply(coops, function(x) x[2, ], .id = 'Mutation')

##       Mutation  Estimate Std. Error    z value     Pr(>|z|)
## 1    NPM1c.Mut 12.771875   2.119444  6.0260494 1.680158e-09
## 2 KMT2A.Fusion  4.519615   2.125312  2.1265654 3.345621e-02
## 3    KMT2A.PTD  1.746473   2.048774  0.8524478 3.939656e-01
## 4     PML.RARA -6.419008   2.332427 -2.7520720 5.921949e-03

NPM1c mutations are significantly associated with the score. Samples carrying a MLL gene fusion are also associated with the score reflecting their shared effects on Hox gene dysregulation, although with a lower significance. Interestingly the PML-RARA fusion carries a negative coefficient, likely reflecting the distinct cellular morphology/phenotype of acute promyelocytic leukemia relative to other subsets of AML, as noted above.

Evaluate prediction model performance
The above statistics give insight on the models trained but not on their performance. Precision and recall provide insight on a models predictive performance. The dcanr package provides functions to compute these metrics along with other measures of performance.

library(dcanr)

#assess sensitivity and specificity
prec_rec = ldply(glms, function(glm1) {
    #predict mutations for the data used in training and convert to binary format
    prediction = as.numeric(predict(glm1) > 0)
    observed = glm1$y
    prec = performanceMeasure(prediction, observed, 'precision')
    recall = performanceMeasure(prediction, observed, 'recall')
    f1 = performanceMeasure(prediction, observed)
    return(c('Precision' = prec, 'Recall' = recall, 'F1' = f1))
}, .id = 'Mutation')
prec_rec

##       Mutation Precision    Recall        F1
## 1    NPM1c.Mut 0.6774194 0.5675676 0.6176471
## 2 KMT2A.Fusion       NaN 0.0000000 0.0000000
## 3    KMT2A.PTD       NaN 0.0000000 0.0000000
## 4     PML.RARA       NaN 0.0000000 0.0000000

Precision for predictions of all genomic changes other than NPM1c is undefined because all samples are predicted to be wild-types. Consequently, their recall is zero. Precision, recall and the F1 score of the NPM1c mutation model are high as expected. As singscores are two-components scores, performance may be further improved by including the dispersion of ranks. This is observed in Figure 2.

As observed from the model below, using both components of singscores significantly improves the predictive performance.
#include dispersion in the model

```r
glm_npm1c = glm('NPM1c.Mut ~ Score + Dispersion',
                  data = scoredf,
                  family = binomial(link = 'logit'))
```

# evaluate performance of the new model

```r
prediction = as.numeric(predict(glm_npm1c) > 0)
observed = glm_npm1c$y
c(
  'Precision' = performanceMeasure(prediction, observed, 'precision'),
  'Recall' = performanceMeasure(prediction, observed, 'recall'),
  'F1' = performanceMeasure(prediction, observed)
)
```

```r
## Precision    Recall        F1
## 0.7631579 0.7837838 0.7733333
```

**Signature landscapes with multiple signatures**

Often, we are interested in the relationship between two dependent or independent phenotypes, for instance, the epithelial and mesenchymal phenotypes. The role of most signatures is to estimate an unobservable molecular phenotype so they may be considered as proxies of phenotypes. As such, we could investigate the relationship between two phenotypes using corresponding signatures. Foroutan *et al.* first introduced the idea of molecular signature landscapes to investigate the relationship between signatures related to the epithelial-mesenchymal transition (EMT) and TGF-β-induced EMT. Subsequently, Cursons *et al.* computed epithelial and mesenchymal phenotype signature singscores and demonstrated an epithelial phenotype shift following miR-200c transfection into a mesenchymal cell line using a signature landscape. Foroutan *et al.* used signature landscapes to stratify breast cancer subtypes along the epithelial-mesenchymal axis and included it as part of the singscore package. Here, we show how such landscapes can be used beyond the current application of the epithelial-mesenchymal axis. We demonstrate how transcriptomic signatures of different mutations can be used to stratify AML samples.

**Ross MLL fusion signature vs. Verhaak signature landscape**

We now use the Ross *et al.* MLL-fusion signatures to score the TCGA AML samples. Unlike the NPM1c signature, this signature is composed of genes that discriminate samples with MLL-fusion genes. We download and parse the signature as demonstrated with the NPM1c signature.

```r
# create signature names
rossmll_name = 'ROSS_AML_WITH_MLL_FUSIONS'
# generate URLs
rossmll_link = paste0(
  'http://software.broadinstitute.org/gsea/msigdb/downloadgeneset.jsp?geneSetName=',
  rossmll_name,
  '&fileType=xml'
)
# download files
rossmll_file = paste0(rossmll_name, '.xml')
download.file(rossmll_link, rossmll_file, method = 'libcurl')
rossmll_sig = getBroadSets(rossmll_file, membersId = 'MEMBERS_EZID')
rossmll_sig
```

```r
# GeneSetCollection
## names: ROSS AML WITH MLL FUSIONS (1 total)
## unique identifiers: 10113, 10479, ..., 9961 (78 total)
## types in collection:
## geneIdType: SymbolIdentifier (1 total)
## collectionType: BroadCollection (1 total)
```

The gene set is a composition of both up- and down-regulated genes as genes were selected based on their ability to discriminate mutants from wild-types. We use the third mode of singscore, which does not require the direction of genes in the gene set to be known. The knownDirection parameter of the simpleScore function is set to 'null' to reflect this. We evaluate the performance of the new model using performanceMeasure function.
function is set to `FALSE` to induce this mode. Ranks computed previously can be reused to compute scores with the new signature.

```r
rossmll_scores = simpleScore(aml_ranked, rossmll_sig[[1]], knownDirection = FALSE)
```

The `plotScoreLandscape` function plots a hexbin plot to visualise the two-dimensional distribution of scores. Scores computed using the Verhaak et al.\textsuperscript{12} signature and the Ross et al.\textsuperscript{14} MLL-fusions signature are passed as arguments. Both scores should have been computed on the same samples with the order of samples retained. Names of the scores should be passed as arguments to the `scorenames` argument. The `textSize` argument can be used to specify the size of all text relative to the plot size. This may prove useful when plots are being generated for scientific posters, publications and presentations, all of which require different image sizes.

```r
p_mll_npm1c = plotScoreLandscape(
  verhaak_scores,
  rossmll_scores,
  scorenames = c('VERHAAK_AML_WITH_NPM1_MUTATED', 'ROSS_AML_WITH_MLL_FUSIONS'),
  textSize = relSize
)
p_mll_npm1c
```

Figure 5 shows a strong positive association between scores from the two signatures (Spearman’s $\rho = 0.611$) despite only 16 genes being shared across the two signatures (signature sizes are 78 and 429 genes). In such an analysis, we may be interested in projecting new data points onto the landscape as done by Cursons et al.\textsuperscript{25}. Alternatively, we may want to overlay some existing data points to investigate sample stratification using scores.

![Signature landscape](image)

**Figure 5.** Signature landscape between the MLL fusion and NPM1c signatures. A positive association is revealed between the two signatures in AML.
Here, we overlay MLL fusions, MLL PTDs, PML-RARA fusions and NPM1c mutations onto the landscape. First, we build the annotation vector.

```
#new annotation - modify previously used annotations
mutated_gene[aml_se$KMT2A.Fusion] = 'MLL Fusion'
mutated_gene[aml_se$KMT2A.PTD] = 'MLL PTD'
mutated_gene[aml_se$PML.RARA] = 'PML-RARA'
```

Points are projected onto an existing landscape using the `projectScoreLandscape` function. This function uses the p_mll_npm1c plot generated using the `plotScoreLandscape` and overlays new data points onto it. Scores for the new data must be computed with the same signatures that were used to compute the landscape. As we are using existing data, scores computed in earlier sections are re-used. The `subSamples` can be used to select a subset of samples to project. Here, we select samples with the mutations we are interested in annotating.

```
select_aml = !mutated_gene %in% 'Other'
#project above mutations onto the landscape
p1 = projectScoreLandscape(p_mll_npm1c, verhaak_scores, rossmll_scores, subSamples = select_aml, annot = mutated_gene[select_aml])
p1 + theme(legend.box = 'vertical')
```

Figure 6 shows that the MLL fusions and MLL PTDs exhibit variation across different axes. MLL PTDs have a lower score than MLL fusions for the MLL fusion signature as expected. These sets of samples do not cluster along the two axes, consistent with observations by Ross et al.\textsuperscript{14}. Overlaying samples annotations onto the plot assists in interpreting different regions of the landscape.

**Ross PML-RARA fusion signature vs. Verhaak signature landscape**

It is evident from the previous analysis that PML-RARA samples differ from all other samples examined here. We repeat the analysis with the PML-RARA signature from Ross et al.\textsuperscript{14} to verify this distinction. The signature is available on the MSigDB and is named “ROSS_AML_WITH_PML_RARA_FUSION”. We download and parse the signature, score all samples against it, plot a landscape with scores from the Verhaak et al.\textsuperscript{12} signature, and finally, project samples onto the plot. The signature was constructed in a similar manner to the MLL fusions signature, therefore, samples are scored using the same settings.

Figure 7 shows a completely different landscape from what was observed with the MLL fusion signature. The PML-RARA signature forms a clear separation between the PML-RARA and NPM1c samples such that PML-RARA samples are the only ones with a high score for this signature. Moreover, no association is observed between the two signatures. As discussed above (Description of the biological problem) the PML-RARA fusion is diagnostic of a specific subtype of AML known as acute promyelocytic leukemia, with a highly distinct cell phenotype reflecting a block on differentiation at the promyelocyte stage\textsuperscript{15}. The distinct features of this subtype are correspondingly reflected in the L-shaped landscape for these two signatures and the different mechanisms by which these lesions drive leukemogenesis.

**Summary**

The singscore package provides an easy interface to apply gene set scoring methods within the R/Bioconductor environment. The TCGAbiolinks package allows relatively easy access to large clinically relevant data sets such as TCGA, together with appropriate annotation functions for interpreting biological data. Diagnostic and plotting functions included with singscore allow the user to investigate gene sets of interest to determine their power for distinguishing differences between samples. Different gene signatures can then be combined to explore how different cellular phenotypes are associated across a large cohort of cancer samples. As demonstrated, when appropriate gene set signatures are used, metrics calculated by singscore can be used for sample classification and this may be useful for further interrogation of large transcriptomic data sets where no genomic data are available.
**Figure 6.** The landscape reveals a distinction between the MLL fusions and PTDs as previously reported. Different mutations occupy different regions of the landscape, possessing different types of molecular traits.

**Figure 7.** The PML-RARA signature is independent from the NPM1c signature in AML. The L-shaped landscape suggests the molecular mechanisms underlying these mutations are mutually exclusive.
Packages used
This workflow depends on various packages from version 3.9 of the Bioconductor project, running on R version 3.6.0 (2019-04-26) or higher. The complete list of the packages used for this workflow are shown below:

```
sessionInfo()
```

```r
## R version 3.6.0 (2019-04-26)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 16299)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Australia.1252 LC_CTYPE=English_Australia.1252
## [3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Australia.1252
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils    datasets
## [8] methods   base
##
## other attached packages:
## [1] hexbin_1.27.3               dcanr_1.0.0
## [3] gridExtra_2.3               reshape2_1.4.3
## [5] singscore_1.4.0             GSEABase_1.46.0
## [7] graph_1.62.0                annotate_1.62.0
## [9] XML_3.98-1.19               org.Hs.eg.db_3.8.2
## [11] AnnotationDbi_1.46.0       plyr_1.8.4
## [13] rtracklayer_1.44.0          edgeR_3.26.3
## [15] limma_3.40.2                SummarizedExperiment_1.14.0
## [17] DelayedArray_0.10.0        BiocParallel_1.17.18
## [19] matrixStats_0.54.0          BiocRegister_1.46.0
## [21] GenomicRanges_1.36.0        GenomeInfoDb_1.20.0
## [23] IRanges_2.18.0              S4Vectors_0.22.0
## [25] BiocGenerics_0.30.0         TCGAbiolinks_2.12.1
## [27] ggplot2_3.1.1               SingscoreAMLMutations_1.0.0
##
## loaded via a namespace (and not attached):
## [1] backports_1.1.4              circlize_0.4.6
## [3] aroma.light_3.14.0           igraph_1.2.4.1
## [5] selectr_0.4-1                ConsensusClusterPlus_1.48.0
## [7] lazyeval_0.2.2                splines_3.6.0
## [9] usethis_1.5.0                 sva_3.32.0
## [11] digest_0.6.19                 foreach_1.4.4
## [13] htmltools_0.3.6               magrittr_1.5
## [15] memoise_1.1.0                 cluster_2.0.9
## [17] doParallel_1.0.14            ComplexHeatmap_2.0.0
## [19] BiocStrings_2.52.0            readr_1.3.1
## [21] R.utils_2.8.0                 prettyunits_1.0.2
## [23] colorspace_1.4-1             blob_1.1.1
## [25] rvest_0.3.4                   ggrepel_0.8.1
## [27] BiocWorkflowTools_1.10.0     xfun_0.7
## [29] dplyr_0.8.1                   crayon_1.3.4
## [31] RCurl_1.95-4.12               jsonlite_1.6
## [33] genefilter_1.66.0             survival_2.44-1.1
## [35] zoo_1.8-5                     iterators_1.0.10
## [37] glue_1.3.1                    survminer_0.4.4
## [39] registry_0.5-1               gtable_0.3.0
```
All data analyzed in the workflow is read automatically from public websites as part of the code. Mutation data for the samples in this study are available as part of the R/Bioconductor package `SingscoreAMLMutations` (v1.0.0).

Software availability


Source code available from: https://github.com/DavisLaboratory/SingscoreAMLMutations/.

Archived source code at time of publication: https://doi.org/10.5281/zenodo.3108807.

License: Artistic License 2.0.

Author information

DDB, JC and MJD wrote the article. DDB wrote the code for the workflow and the associated R/Bioconductor workflow package `SingscoreAMLMutations`. MF, JC, DDB and MJD conceived the idea for the workflow. YX translated the vignette for the package vignette into chinese. YX and RL reviewed the chinese translation. All authors have tested and reviewed the workflow. This work was supervised by JC and MJD.
Acknowledgments

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com