

Package: OncoSeq (via r-universe)

October 21, 2026

Type Package

Title End-to-End Analysis of Cancer Transcriptomes and Genomes

Version 0.99.1

Description OncoSeq provides a unified, end-to-end workflow for cancer transcriptomic and genomic data analysis. A single container class couples bulk RNA-sequencing counts with somatic mutation calls, gene-level copy-number alterations and clinical follow-up for the same set of tumour samples. The package covers data import from the Genomic Data Commons, cBioPortal and local quantification files; quality control and filtering; normalisation; differential expression through DESeq2, edgeR or limma-voom behind one interface; gene-set enrichment and over-representation testing; Kaplan-Meier and Cox survival modelling including genome-wide survival screens; mutation and copy-number summarisation; integration of genomic alterations with expression; and publication-ready visualisation. Every analysis is recorded in the object so that a complete study can be reproduced from one object.

License GPL-3

Encoding UTF-8

Depends R (>= 4.4.0), SummarizedExperiment

Imports methods, stats, utils, grDevices, S4Vectors, DESeq2, edgeR, limma, fgsea, BiocParallel, survival, ggplot2, ggrepel, matrixStats, httr2, jsonlite

Suggests TCGAbiolinks, maftools, clusterProfiler, org.Hs.eg.db, ComplexHeatmap, pheatmap, survminer, tximport, msigdb, apeglm, testthat (>= 3.0.0), knitr, rmarkdown, BiocStyle

biocViews Software, GeneExpression, DifferentialExpression, RNASeq, Transcriptomics, Sequencing, GeneSetEnrichment, Survival, SomaticMutation, CopyNumberVariation, Visualization, QualityControl, DataImport, Normalization, MultipleComparison

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://github.com/babasaraki/OncoSeq>

BugReports <https://github.com/babasaraki/OncoSeq/issues>

Collate 'AllGenerics.R' 'AllClasses.R' 'methods-OncoSeqExperiment.R'
 'utils.R' 'import-local.R' 'import-gdc.R' 'import-cbioportal.R'
 'qc.R' 'normalize.R' 'de.R' 'genesets.R' 'enrichment.R'
 'mutations.R' 'cnv.R' 'survival.R' 'integrate.R' 'plots.R'
 'pipeline.R' 'data.R' 'zzz.R'

Config/roxygen2/version 8.1.1

NeedsCompilation no

Config/pak/sysreqs libssl-dev zlib1g-dev

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-10-21 04:29:39 UTC

RemoteUrl <https://github.com/BiocStaging/OncoSeq>

RemoteRef HEAD

RemoteSha 0675d14d3025d43abed9aa6643f6b869414a05a2

Contents

alignLayers	3
asOncoSeq	4
blcaExample	5
cbioProfiles	6
cbioStudies	7
cnvExpressionCorrelation	7
cnvFrequency	8
cnvSummary	9
compareEngines	10
coxModel	11
detectOutliers	12
diffExpress	13
exampleOncoSeq	14
filterLowExpression	15
getAnalysis	16
importCBioPortal	17
importCounts	19
importGDC	20
importMAF	21
importTximport	22
integrateMutationExpression	23
msigdbSets	24
mutationFrequency	25
mutationStatus	26
mutationSummary	27
normalizeCounts	27

oncoplotData	28
OncoSeq-generics	29
OncoSeqExperiment	30
OncoSeqExperiment-class	31
OncoSeqExperiment-methods	32
plotCNVFrequency	34
plotEnrichmentBar	34
plotExpressionHeatmap	35
plotKM	36
plotMAplot	37
plotMutationFrequency	37
plotOncoprint	38
plotSamplePCA	39
plotVolcano	40
qcMetrics	41
readGMT	42
runGSEA	42
runOncoSeq	44
runORA	45
simExample	46
survivalFit	47
survivalScreen	48
tumourMutationBurden	49

Index	51
--------------	-----------

alignLayers	<i>Restrict an object to samples present in every layer</i>
-------------	---

Description

Intersects the sample sets of the expression, mutation, copy-number and clinical layers so that downstream multi-omic analyses operate on a consistent cohort, and reports what was dropped.

Usage

```
alignLayers(
  object,
  require = c("mutation", "cnv"),
  timeColumn = "os_time",
  eventColumn = "os_event",
  verbose = TRUE
)
```

Arguments

object	An OncoSeqExperiment .
require	Layers that a sample must have data for. Any of "mutation", "cnv", "survival".
timeColumn, eventColumn	Columns checked when "survival" is required.
verbose	Report the number of samples retained.

Value

The subset [OncoSeqExperiment](#).

Examples

```
ose <- exampleOncoSeq()
alignLayers(ose, require = c("mutation", "survival"))
```

asOncoSeq	<i>Convert an existing object to an OncoSeqExperiment</i>
-----------	---

Description

Wraps a [SummarizedExperiment](#), a 'DESeqDataSet' or an 'edgeR' 'DGEList' in an [OncoSeqExperiment](#) so that it can enter the OncoSeq workflow, optionally attaching mutation and copy-number layers.

Usage

```
asOncoSeq(object, mutations = NULL, cnv = NULL)
```

Arguments

object	A SummarizedExperiment , 'DESeqDataSet' or 'DGEList'.
mutations	Optional variant table.
cnv	Optional gene-level copy-number matrix or SummarizedExperiment .

Value

An [OncoSeqExperiment](#).

Examples

```
se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(counts = matrix(rpois(100, 30), nrow = 10,
    dimnames = list(paste0("G", 1:10), paste0("S", 1:10))))))
asOncoSeq(se)
```

blcaExample

*Bladder urothelial carcinoma multi-omic example cohort***Description**

A real, trimmed TCGA bladder cancer cohort assembled from the cBioPortal PanCancer Atlas study ‘blca_tcga_pan_can_atlas_2018’ with [importCBioPortal()]. It carries mRNA expression, somatic mutations, discrete copy-number calls and clinical follow-up for the same 400 tumours, and is the dataset used throughout the vignette for the genomic and survival sections.

Usage

```
data(blcaExample)
```

Format

An [OncoSeqExperiment](#) with 420 features and 400 samples:

assay ‘expression’ RSEM-normalised mRNA abundance.

‘colData’ clinical attributes including ‘AGE’, ‘SEX’, ‘AJCC_PATHOLOGIC_TUMOR_STAGE’, ‘SUBTYPE’, ‘TMB_NONSYNONYMOUS’, ‘ANEUPLOIDY_SCORE’, and the harmonised ‘os_time’/‘os_event’.

‘mutations’ 14,337 somatic variants with ‘sample’, ‘gene’, ‘variant_class’, position and protein change.

‘cnv’ discrete GISTIC gene-level copy-number calls.

Details

Genes were restricted to a 420-gene panel: the 400 genes mutated in the largest number of samples in the study, plus a curated list of urothelial-carcinoma drivers and luminal/basal markers. Samples were restricted to the 400 tumours carrying expression, mutation, copy-number and overall-survival data simultaneously.

The ‘expression’ assay holds RSEM-normalised values, not raw counts, so count-based engines will refuse to model it; use ‘engine = “limma”’ with ‘voom = FALSE’, or [exampleOncoSeq()] when a count-scale demonstration is needed. Copy-number values are GISTIC calls in ‘-2:2’. Survival is harmonised into ‘os_time’ (months) and ‘os_event’ (1 = death).

Source

cBioPortal study ‘blca_tcga_pan_can_atlas_2018’ (<https://www.cbioportal.org/study/summary?id=blca_tcga_pan_can_atlas_2018> derived from the TCGA PanCancer Atlas. The build script is ‘data-raw/make-blcaExample.R’.

See Also

[exampleOncoSeq()] for a simulated count-scale object.

Examples

```
data(blcaExample)
blcaExample
head(mutationFrequency(blcaExample, topN = 10))
```

cbioProfiles	<i>List the molecular profiles of a cBioPortal study</i>
--------------	--

Description

List the molecular profiles of a cBioPortal study

Usage

```
cbioProfiles(studyId, baseUrl = CBIOPORTAL_URL)
```

Arguments

studyId	cBioPortal study identifier, for example "blca_tcga_pan_can_atlas_2018".
baseUrl	Base URL of the cBioPortal API.

Value

A 'data.frame' of molecular profiles with their 'molecularProfileId', 'molecularAlterationType' and 'datatype'.

Examples

```
## Not run:
cbioProfiles("blca_tcga_pan_can_atlas_2018")[, c("molecularProfileId",
                                                "molecularAlterationType")]

## End(Not run)
```

cbioStudies

List studies available from cBioPortal

Description

Queries a cBioPortal instance for the studies it serves, optionally filtered by a keyword matched against the study identifier, name and cancer type.

Usage

```
cbioStudies(keyword = NULL, baseUrl = CBIOPORTAL_URL)
```

Arguments

keyword	Optional case-insensitive search term, for example "bladder".
baseUrl	Base URL of the cBioPortal API.

Value

A 'data.frame' with one row per study, including 'studyId', 'name', 'cancerTypeId' and the sample counts.

See Also

```
[cbioProfiles()], [importCBioPortal()]
```

Examples

```
## Not run:
bladder <- cbioStudies("bladder")
head(bladder[, c("studyId", "name", "allSampleCount")])

## End(Not run)
```

cnvExpressionCorrelation

Correlate copy number with expression

Description

Tests the dosage effect of copy-number alteration on transcript abundance gene by gene, over the samples that carry both layers. A positive correlation is the expected signature of a copy-number-driven expression change and is used to separate passenger from dosage-sensitive genes.

Usage

```
cnvExpressionCorrelation(
  object,
  genes = NULL,
  exprAssay = NULL,
  cnvAssay = 1L,
  method = "spearman",
  minSamples = 10L
)
```

Arguments

object	An OncoSeqExperiment with expression and copy-number layers.
genes	Genes to test; defaults to every gene present in both layers.
exprAssay	Expression assay to use; log-scale values are preferred and are derived from counts when necessary.
cnvAssay	Name or index of the copy-number assay.
method	Correlation method, passed to [stats::cor.test()].
minSamples	Minimum number of complete pairs required to test a gene.

Value

A 'data.frame' with 'gene', 'n', 'estimate', 'pvalue' and 'padj', ordered by adjusted p-value.

Examples

```
ose <- exampleOncoSeq()
head(cnvExpressionCorrelation(ose))
```

cnvFrequency

Copy-number alteration frequency

Description

Per-gene frequency of gain and loss across the cohort, in the long format expected by [plotCNVFrequency()].

Usage

```
cnvFrequency(
  object,
  genes = NULL,
  topN = 20,
  ampThreshold = 2,
  delThreshold = -2,
  assay = 1L
)
```

Arguments

object	An OncoSeqExperiment with a copy-number layer.
genes	Optional character vector restricting the output.
topN	Keep the ‘topN’ genes with the highest alteration frequency.
ampThreshold	Values greater than or equal to this count as a gain.
delThreshold	Values less than or equal to this count as a loss.
assay	Name or index of the copy-number assay.

Value

A ‘data.frame’ with columns ‘gene’, ‘alteration’ (“gain” or “loss”) and ‘frequency’.

Examples

```
ose <- exampleOncoSeq()
head(cnvFrequency(ose, topN = 5))
```

cnvSummary	<i>Summarise gene-level copy-number alterations</i>
------------	---

Description

Counts amplifications and deletions per gene and per sample from the discrete copy-number layer. Thresholds follow the GISTIC convention in which ‘2’ and ‘-2’ denote high-level amplification and deep deletion and ‘1’/‘-1’ denote single-copy gain and loss.

Usage

```
cnvSummary(object, ampThreshold = 2, delThreshold = -2, assay = 1L)
```

Arguments

object	An OncoSeqExperiment with a copy-number layer.
ampThreshold	Values greater than or equal to this count as a gain.
delThreshold	Values less than or equal to this count as a loss.
assay	Name or index of the copy-number assay.

Value

A list with ‘perGene’ and ‘perSample’ ‘data.frame’s.

Examples

```
ose <- exampleOncoSeq()
cs <- cnvSummary(ose)
head(cs$perGene)
```

compareEngines	<i>Run the same contrast through every engine and compare</i>
----------------	---

Description

Fits the same comparison with DESeq2, edgeR and limma-voom and returns the overlap of their significant calls together with a merged table of log fold changes. Concordance across engines is a practical check that a signature is not an artefact of one model's assumptions.

Usage

```
compareEngines(
  object,
  design,
  contrast = NULL,
  coef = NULL,
  engines = c("DESeq2", "edgeR", "limma"),
  alpha = 0.05,
  assay = NULL
)
```

Arguments

object	An OncoSeqExperiment .
design	A one-sided formula over 'colData()' columns, for example '~ condition' or '~ batch + condition'.
contrast	A character triple 'c(variable, numerator, denominator)' or a numeric contrast vector over the design-matrix columns.
coef	Name or index of a single design-matrix coefficient to test; an alternative to 'contrast'.
engines	Engines to run.
alpha	Adjusted-p-value threshold used for the overlap counts.
assay	Assay to model. Count engines default to 'counts'; 'limma' uses 'counts' with voom unless 'voom = FALSE'.

Value

A list with 'merged' (one row per feature, log2FC and padj from each engine), 'overlap' (a named integer vector of significant counts per engine and the size of their intersection) and 'consensus' (the features significant in every engine).

Examples

```
ose <- exampleOncoSeq()
ose <- filterLowExpression(ose, group = "condition", verbose = FALSE)
cmp <- compareEngines(ose, ~ condition,
                      contrast = c("condition", "tumour", "normal"))
cmp$overlap
```

coxModel

Multivariable Cox proportional-hazards model

Description

Fits `[survival::coxph()]` with covariates drawn from `'colData()'`, from expression (prefix a gene name with `'expr:'`) or from mutation status (prefix with `'mut:'`), and returns a tidy table of hazard ratios.

Usage

```
coxModel(
  object,
  covariates,
  timeColumn = "os_time",
  eventColumn = "os_event",
  assay = NULL,
  name = "cox"
)
```

Arguments

<code>object</code>	An OncoSeqExperiment .
<code>covariates</code>	Character vector of terms, for example <code>'c("age", "stage", "expr:TP53", "mut:RB1")'</code> .
<code>timeColumn, eventColumn</code>	Follow-up and event columns in <code>'colData()'</code> .
<code>assay</code>	Expression assay used for <code>'expr:'</code> terms.
<code>name</code>	Name under which the result is stored in the registry; <code>'NULL'</code> returns the table instead of the object.

Value

The object with the tidy table stored, or the table itself when `'name'` is `'NULL'`. Columns are `'term'`, `'coef'`, `'HR'`, `'ci_lower'`, `'ci_upper'`, `'se'`, `'z'` and `'pvalue'`. The fitted model is attached as the `'model'` attribute.

Examples

```
ose <- exampleOncoSeq()
coxModel(ose, c("expr:TP53", "mut:TP53"), name = NULL)
```

detectOutliers	<i>Flag candidate outlier samples</i>
----------------	---------------------------------------

Description

Identifies samples that sit far from the bulk of the data in two complementary ways: distance from the centroid in the space of the leading principal components, and low median correlation with all other samples. Both criteria are reported; nothing is removed.

Usage

```
detectOutliers(
  object,
  assay = NULL,
  nTopGenes = 500L,
  nPC = 2L,
  madThreshold = 3,
  corMethod = "spearman"
)
```

Arguments

object	An OncoSeqExperiment .
assay	Assay to use. When absent, log2 counts-per-million are computed from the counts.
nTopGenes	Number of most variable features used for the principal component analysis.
nPC	Number of principal components used for the distance.
madThreshold	A sample is flagged when its centroid distance exceeds the median distance by this many median absolute deviations.
corMethod	Correlation method for the second criterion.

Value

A [DataFrame](#) with one row per sample giving the centroid distance, the median inter-sample correlation and a logical ‘outlier’ column.

Examples

```
ose <- exampleOncoSeq()
out <- detectOutliers(ose)
sum(out$outlier)
```

diffExpress

*Differential expression with DESeq2, edgeR or limma***Description**

Fits a differential-expression model with one of three engines behind a single interface and returns one harmonised result table. The engines are the standard implementations: [DESeq2::DESeq()] with a negative-binomial Wald test, [edgeR::glmQLFTest()] quasi-likelihood F tests, and [limma::eBayes()] on [limma::voom()] precision weights (or on an already transformed assay with 'limma-trend').

Usage

```
diffExpress(
  object,
  design,
  contrast = NULL,
  coef = NULL,
  engine = c("DESeq2", "edgeR", "limma"),
  assay = NULL,
  voom = NULL,
  shrink = FALSE,
  name = "de",
  alpha = 0.05,
  verbose = TRUE,
  ...
)
```

Arguments

object	An OncoSeqExperiment .
design	A one-sided formula over 'colData()' columns, for example '~ condition' or '~ batch + condition'.
contrast	A character triple 'c(variable, numerator, denominator)' or a numeric contrast vector over the design-matrix columns.
coef	Name or index of a single design-matrix coefficient to test; an alternative to 'contrast'.
engine	One of "DESeq2", "edgeR" or "limma".
assay	Assay to model. Count engines default to 'counts'; 'limma' uses 'counts' with voom unless 'voom = FALSE'.
voom	For 'engine = "limma"', apply [limma::voom()] to counts ('TRUE'), or fit 'limma-trend' to the supplied transformed assay ('FALSE'). 'NULL' chooses automatically from the assay.
shrink	For 'engine = "DESeq2"', apply [DESeq2::lfcShrink()] with 'apeglm'; requires the 'apeglm' package and a 'coef'.
name	Name under which the result is stored in the analysis registry.

alpha	Significance level used when reporting the number of differentially expressed features.
verbose	Print a one-line summary of the fit.
...	Further arguments passed to the engine's fitting function ([DESeq2::DESeq()], [edgeR::glmQLFit()] or [limma::eBayes()]).

Details

The contrast may be given in three ways:

- a character triple 'c(variable, numerator, denominator)', which tests 'numerator - denominator' for that 'colData()' variable;
- a coefficient name present in the design matrix, through 'coef';
- a numeric vector over the columns of the design matrix.

Additional terms in 'design' are adjusted for. The same contrast specification gives the same comparison in all three engines, so results can be compared directly.

Value

The [OncoSeqExperiment](#) with the result table stored in 'analyses(object)[[name]]'. Retrieve it with [getAnalysis()]. The table is a 'data.frame' with columns 'feature', 'baseMean', 'log2FC', 'lfcSE', 'stat', 'pvalue', 'padj' and 'engine', ordered by 'padj'.

See Also

[compareEngines()], [runGSEA()], [plotVolcano()]

Examples

```
ose <- exampleOncoSeq()
ose <- filterLowExpression(ose, group = "condition", verbose = FALSE)
ose <- diffExpress(ose, ~ condition,
                  contrast = c("condition", "tumour", "normal"),
                  engine = "edgeR", name = "de_edger")
head(getAnalysis(ose, "de_edger"))
```

exampleOncoSeq

A small simulated cancer multi-omic dataset

Description

Returns [simExample], a compact [OncoSeqExperiment](#) with expression counts, somatic mutations, gene-level copy number and overall-survival follow-up, optionally trimmed to fewer features or samples. It is the object used by the examples and unit tests.

Usage

```
exampleOncoSeq(nGenes = NULL, nPerGroup = NULL)
```

Arguments

nGenes	Number of features to keep, taken from the start of the object so that the named driver genes are always present. ‘NULL’ keeps all 300.
nPerGroup	Number of tumour and of normal samples to keep. ‘NULL’ keeps all 20 of each.

Details

The data are simulated, not measured: a negative-binomial count model with a planted differential signal between tumour and normal, mutations planted at frequencies resembling those of urothelial carcinoma, and survival times from an exponential model whose rate depends on ‘TP53’ mutation status. Gene symbols are real so that gene sets and driver names behave sensibly in examples.

The object is built once, by ‘data-raw/make-simExample.R’, and shipped with the package. This function therefore draws no random numbers and leaves the caller’s random stream untouched; repeated calls return identical objects.

Value

An [OncoSeqExperiment](#) with a ‘counts’ assay, ‘condition’, ‘age’, ‘stage’, ‘os_time’ and ‘os_event’ in ‘colData()’, a mutation table and a copy-number layer.

See Also

[simExample] for the full object, [importCounts()] to build one from your own data.

Examples

```
ose <- exampleOncoSeq()
ose
table(SummarizedExperiment::colData(ose)$condition)
dim(exampleOncoSeq(nGenes = 50, nPerGroup = 5))
```

filterLowExpression	<i>Remove weakly expressed features</i>
---------------------	---

Description

Drops features that carry too little signal to be tested. With a grouping variable the filter is [edgeR::filterByExpr()], the standard choice for count data, which keeps features with enough counts in a number of samples equal to the smallest group. Without a grouping variable a simple counts-per-million rule is applied.

Usage

```
filterLowExpression(
  object,
  group = NULL,
  assay = NULL,
  minCount = 10,
  minCPM = 1,
  minSamples = 2L,
  verbose = TRUE
)
```

Arguments

object	An OncoSeqExperiment .
group	Name of a 'colData()' column giving the experimental groups, or a vector of the same length as the number of samples. 'NULL' applies the CPM rule.
assay	Name of the count assay; defaults to 'counts'.
minCount	Minimum count required, passed to [edgeR::filterByExpr()].
minCPM	CPM threshold used when 'group' is 'NULL'.
minSamples	Number of samples that must exceed 'minCPM' when 'group' is 'NULL'; defaults to the size of the smallest conceivable group, 2.
verbose	Report how many features were removed.

Value

The filtered [OncoSeqExperiment](#).

Examples

```
ose <- exampleOncoSeq()
dim(ose)
dim(filterLowExpression(ose, group = "condition"))
```

getAnalysis

Retrieve a stored analysis result

Description

Extract one result from the analysis registry of an [OncoSeqExperiment](#), with an informative error when the name is not present.

Usage

```
getAnalysis(object, name)
```

Arguments

object An [OncoSeqExperiment](#).
 name Name of the stored result. When missing, the available names are returned.

Value

The stored result object, or a character vector of available names when ‘name’ is missing.

Examples

```
ose <- exampleOncoSeq()
getAnalysis(ose)
ose <- diffExpress(ose, ~ condition,
                  contrast = c("condition", "tumour", "normal"),
                  engine = "edgeR", verbose = FALSE)
head(getAnalysis(ose, "de"))
```

importCBioPortal	<i>Import a cBioPortal study</i>
------------------	----------------------------------

Description

Assembles a curated cBioPortal study into an [OncoSeqExperiment](#): an mRNA expression matrix, the somatic mutation table, discrete (GISTIC-style) gene-level copy-number calls and the clinical table, all restricted to samples shared by the requested layers.

Usage

```
importCBioPortal(
  studyId,
  genes = NULL,
  expressionProfile = NULL,
  mutationProfile = NULL,
  cnvProfile = NULL,
  mutations = TRUE,
  cnv = TRUE,
  clinical = TRUE,
  chunkSize = 500L,
  baseUrl = CBIOPORTAL_URL
)
```

Arguments

studyId cBioPortal study identifier; see [cbioStudies()].
 genes Optional character vector of HUGO symbols to restrict the expression and copy-number layers to. ‘NULL’ fetches every gene in the profile, which for a whole study is a large download.

expressionProfile	Molecular profile identifier for expression. When 'NULL' the first continuous mRNA profile of the study is used.
mutationProfile	Molecular profile identifier for mutations, or 'NULL' to use the study's extended mutation profile. Set 'mutations' to 'FALSE' to skip.
cnvProfile	Molecular profile identifier for discrete copy number, or 'NULL' to use the study's GISTIC profile. Set 'cnv' to 'FALSE' to skip.
mutations, cnv, clinical	Logical flags controlling which optional layers are retrieved.
chunkSize	Number of genes per API request.
baseUrl	Base URL of the cBioPortal API.

Details

cBioPortal serves processed expression values rather than raw counts: for TCGA PanCancer Atlas studies the default profile holds RSEM-normalised values. The resulting assay is therefore named 'expression', not 'counts', and count-based differential-expression engines will refuse to run on it; use 'engine = "limma"' on log-transformed values, or [importGDC()] when raw counts are required.

Clinical attributes arrive in long form and are pivoted to one column per attribute. Overall survival is harmonised into 'os_time' and 'os_event' when the study carries 'OS_MONTHS'/'OS_STATUS'.

Value

An [OncoSeqExperiment](#) with an 'expression' assay.

See Also

[importGDC()], [cbioStudies()]

Examples

```
## Not run:
ose <- importCBioPortal("blca_tcga_pan_can_atlas_2018",
                       genes = c("TP53", "RB1", "FGFR3", "ERBB2"))
ose

## End(Not run)
```

importCounts

*Import counts and sample annotation from local data***Description**

Builds an [OncoSeqExperiment](#) from a gene-by-sample count matrix and a table of sample annotation held in memory or on disk. Optional somatic-variant and copy-number layers are attached and checked against the expression sample identifiers.

Usage

```
importCounts(
  counts,
  colData = NULL,
  rowData = NULL,
  mutations = NULL,
  cnv = NULL,
  sep = "\t",
  intersectSamples = TRUE
)
```

Arguments

counts	A numeric matrix (genes in rows, samples in columns), a ‘data.frame’, or a path to a delimited text file whose first column holds feature identifiers.
colData	A ‘data.frame’/ DataFrame of sample annotation, or a path to a delimited file whose first column holds sample identifiers. Rows are matched to the columns of ‘counts’ by name when names are available, otherwise by position.
rowData	Optional feature annotation, as an object or a file path.
mutations	Optional variant table, as an object or a path to a MAF or delimited file (see [importMAF()]).
cnv	Optional gene-level copy-number matrix, as an object or a file path.
sep	Field separator used when reading files.
intersectSamples	When ‘TRUE’ (the default) the object is restricted to samples present in both ‘counts’ and ‘colData’. When ‘FALSE’ a mismatch is an error.

Value

An [OncoSeqExperiment](#).

See Also

[\[importGDC\(\)\]](#), [\[importCBioPortal\(\)\]](#), [\[importTximport\(\)\]](#)

Examples

```
counts <- matrix(rpois(300, 40), nrow = 30,
                 dimnames = list(paste0("G", 1:30), paste0("S", 1:10)))
meta <- data.frame(condition = rep(c("tumour", "normal"), each = 5),
                  row.names = colnames(counts))
ose <- importCounts(counts, meta)
ose
```

importGDC

Import a TCGA/GDC project

Description

Downloads and assembles a Genomic Data Commons project into a single [OncoSeqExperiment](#): STAR gene-expression counts, the indexed clinical table with harmonised overall-survival columns, and optionally simple somatic mutations and gene-level copy-number scores. Layers are keyed on the TCGA patient barcode so that expression, genomic and clinical data for one tumour line up.

Usage

```
importGDC(
  project = "TCGA-BLCA",
  sampleType = NULL,
  mutations = FALSE,
  cnv = FALSE,
  clinical = TRUE,
  directory = "GDCdata",
  files.per.chunk = 50L
)
```

Arguments

project	GDC project identifier, for example "TCGA-BLCA" for bladder urothelial carcinoma.
sampleType	Character vector of sample types to keep, for example 'c("Primary Tumor", "Solid Tissue Normal")'. 'NULL' keeps all.
mutations	Logical; also download simple somatic mutations.
cnv	Logical; also download gene-level copy-number scores.
clinical	Logical; attach the indexed clinical table.
directory	Download cache directory passed to [TCGAbiolinks::GDCdownload()].
files.per.chunk	Number of files per download chunk; lower this on unstable connections.

Details

The GDC retired the "HTSeq - Counts" workflow when it moved to Data Release 32; this function requests "STAR - Counts", which is the current gene-expression workflow. Downloaded files are cached by 'TCGAbiolinks' in 'directory', so repeating a call is inexpensive.

Sample barcodes are truncated to the 12-character patient barcode so that expression, mutation, copy-number and clinical layers share identifiers. When a patient contributes more than one aliquot the first is retained and a message reports how many were dropped.

Value

An [OncoSeqExperiment](#) with a 'counts' assay (plus 'tpm_unstrand' and 'fpkm_unstrand' when the GDC supplies them), clinical columns in 'colData', and the mutation and copy-number layers when requested.

Requirements

Needs the 'TCGAbiolinks' package and network access to 'api.gdc.cancer.gov'.

See Also

[importCBioPortal()] for a lighter, API-based alternative.

Examples

```
# Downloads several hundred megabytes; not run during package checks.
## Not run:
blca <- importGDC("TCGA-BLCA", sampleType = "Primary Tumor",
                 mutations = TRUE, cnv = TRUE)

blca

## End(Not run)
```

importMAF

Read a MAF file

Description

Reads a Mutation Annotation Format file (optionally gzip-compressed) and harmonises its column names to the OncoSeq vocabulary, keeping all original columns.

Usage

```
importMAF(file, sep = "\t", comment = "#")
```

Arguments

file	Path to a MAF or MAF-like delimited file.
sep	Field separator.
comment	Lines beginning with this character are skipped; MAF files from the GDC begin with a version header.

Value

A [DataFrame](#) with at least ‘sample’, ‘gene’ and, when present, ‘variant_class’.

See Also

[mutationFrequency()], [oncplotData()]

Examples

```
maf <- system.file("extdata", "example_mutations.maf", package = "OncoSeq")
head(importMAF(maf))
```

importTximport

Import transcript-level quantifications through tximport

Description

Summarises transcript-level abundance files produced by salmon, kallisto, RSEM or similar to the gene level with [tximport::tximport()] and returns an [OncoSeqExperiment](#) holding the gene-level counts, the average transcript lengths and the TPM estimates.

Usage

```
importTximport(
  files,
  tx2gene,
  type = "salmon",
  colData = NULL,
  countsFromAbundance = "lengthScaledTPM",
  ...
)
```

Arguments

files	Named character vector of quantification files; the names become sample identifiers.
tx2gene	A two-column ‘data.frame’ mapping transcript to gene identifiers.
type	Quantification tool, passed to [tximport::tximport()].

colData	Optional sample annotation, matched to 'names(files)'.
countsFromAbundance	Passed to [tximport::tximport()]; the default "lengthScaledTPM" yields counts suitable for direct use in differential expression.
...	Further arguments passed to [tximport::tximport()].

Value

An [OncoSeqExperiment](#) with assays 'counts', 'abundance' and 'length'.

Examples

```
# Requires the tximport package and quantification files on disk:
if (requireNamespace("tximport", quietly = TRUE)) {
  # files <- c(S1 = "S1/quant.sf", S2 = "S2/quant.sf")
  # ose <- importTximport(files, tx2gene = t2g, type = "salmon")
}
```

integrateMutationExpression

Differential expression between mutant and wild-type tumours

Description

Adds the mutation status of a driver gene to 'colData()' and runs [diffExpress()] comparing mutant with wild-type samples, optionally adjusting for clinical covariates. This is the standard way to ask what transcriptional programme accompanies an alteration.

Usage

```
integrateMutationExpression(
  object,
  gene,
  covariates = NULL,
  engine = "edgeR",
  nonsynonymousOnly = TRUE,
  minGroupSize = 3L,
  name = paste0("de_", gene),
  ...
)
```

Arguments

object	An OncoSeqExperiment with expression and mutation layers.
gene	Driver gene whose status defines the comparison.
covariates	Optional 'colData()' columns to adjust for.

engine	Differential-expression engine; see [diffExpress()].
nonsynonymousOnly	Count only protein-altering variants.
minGroupSize	Refuse to run when either group is smaller than this.
name	Name under which the result is stored.
...	Passed to [diffExpress()].

Value

The object with the result stored in the registry and a ‘<gene>_status’ column added to ‘colData()’.

See Also

[diffExpress()], [mutationStatus()]

Examples

```
ose <- exampleOncoSeq()
ose <- filterLowExpression(ose, group = "condition", verbose = FALSE)
ose <- integrateMutationExpression(ose, "TP53", engine = "edgeR")
head(getAnalysis(ose, "de_TP53"))
```

msigdbSets

Fetch a gene-set collection through msigdb

Description

Convenience wrapper that converts an MSigDB collection obtained with the ‘msigdb’ package into the named-list format used by OncoSeq.

Usage

```
msigdbSets(species = "Homo sapiens", collection = "H", subcollection = NULL)
```

Arguments

species	Species name accepted by [msigdb::msigdb()], for example “Homo sapiens”.
collection	Collection identifier, for example “H” for the hallmark sets or “C2” for curated sets.
subcollection	Optional sub-collection, for example “CP:REACTOME”.

Value

A named list of gene symbols.

Requirements

Needs the 'msigdbR' package, which is not a hard dependency of OncoSeq.

Examples

```
## Not run:
hallmark <- msigdbRSets("Homo sapiens", "H")

## End(Not run)
```

mutationFrequency	<i>Mutation frequency across the cohort</i>
-------------------	---

Description

For each gene, the proportion of samples carrying at least one mutation, with an exact binomial (Clopper-Pearson) confidence interval. The denominator is the number of samples in the object, so genes absent from the mutation table are correctly treated as unmutated.

Usage

```
mutationFrequency(
  object,
  genes = NULL,
  topN = 20,
  nonsynonymousOnly = TRUE,
  conf.level = 0.95
)
```

Arguments

object	An OncoSeqExperiment .
genes	Optional character vector restricting the output.
topN	Return only the 'topN' most frequently mutated genes; 'Inf' returns all.
nonsynonymousOnly	Restrict to protein-altering variants.
conf.level	Confidence level for the interval.

Value

A 'data.frame' with 'gene', 'n_mutated', 'n_samples', 'frequency', 'ci_lower' and 'ci_upper', ordered by frequency.

Examples

```
ose <- exampleOncoSeq()
mutationFrequency(ose, topN = 5)
```

mutationStatus	<i>Mutation status of a gene as a factor</i>
----------------	--

Description

Returns a two-level factor giving, for every sample in the object, whether the gene carries a somatic mutation. Samples absent from the mutation table are treated as wild-type, which is correct when the whole cohort was sequenced; check this assumption for cohorts with partial sequencing.

Usage

```
mutationStatus(object, gene, nonsynonymousOnly = TRUE)
```

Arguments

object	An OncoSeqExperiment .
gene	Gene symbol.
nonsynonymousOnly	Count only protein-altering variants.

Value

A factor with levels ‘wild-type’ and ‘mutant’, named by sample.

Examples

```
ose <- exampleOncoSeq()
table(mutationStatus(ose, "TP53"))
```

mutationSummary	<i>Summarise the somatic mutation layer</i>
-----------------	---

Description

Produces per-sample and per-gene summaries of the mutation table: the number of variants, the number of distinct genes or samples affected, and the breakdown by variant classification.

Usage

```
mutationSummary(object, nonsynonymousOnly = TRUE)
```

Arguments

object An [OncoSeqExperiment](#).
nonsynonymousOnly Restrict to protein-altering variants.

Value

A list with elements 'perSample', 'perGene' and 'byClass', each a 'data.frame'.

Examples

```
ose <- exampleOncoSeq()
s <- mutationSummary(ose)
head(s$perGene)
```

normalizeCounts	<i>Normalise expression data</i>
-----------------	----------------------------------

Description

Computes a normalised expression assay and adds it to the object, leaving the raw counts untouched. Count-scale methods wrap the standard implementations in 'edgeR' and 'DESeq2'.

Usage

```
normalizeCounts(
  object,
  method = c("logcpm", "cpm", "tmm", "upperquartile", "rle", "vst", "rlog", "log2tpm"),
  assay = NULL,
  name = NULL,
  priorCount = 1,
  lengthColumn = NULL,
  blind = TRUE
)
```

Arguments

object	An OncoSeqExperiment .
method	Normalisation method; see Details.
assay	Name of the count assay to transform.
name	Name under which the new assay is stored; defaults to the method name.
priorCount	Prior count for the log transformations.
lengthColumn	Name of a 'rowData()' column holding feature lengths in bases, required for 'method = "log2tpm"'.
blind	Passed to [DESeq2::vst()] / [DESeq2::rlog()].

Details

Supported methods:

"cpm" Counts per million.

"logcpm" log2 counts per million with a prior count.

"tmm" log2 CPM after trimmed mean of M-values scaling ([edgeR::normLibSizes()]).

"upperquartile", **"rle"** log2 CPM after the corresponding 'edgeR' scaling factors.

"vst" Variance-stabilising transformation ([DESeq2::vst()]); the standard choice for clustering and principal-component analysis of count data.

"rlog" Regularised log ([DESeq2::rlog()]); better behaved than 'vst' for small experiments but much slower.

"log2tpm" log2 transcripts per million, which requires a feature-length column.

Value

The [OncoSeqExperiment](#) with an added assay.

Examples

```
ose <- exampleOncoSeq()
ose <- normalizeCounts(ose, method = "logcpm")
SummarizedExperiment::assayNames(ose)
```

oncoplotData

Build the matrix behind an oncoprint

Description

Returns a gene-by-sample character matrix whose entries are the variant classification carried by that gene in that sample, empty where the gene is not mutated. Samples carrying several variants in one gene are labelled "Multi_Hit", the convention used by oncoprint renderers.

Usage

```
oncoplotData(
  object,
  genes = NULL,
  topN = 20,
  nonsynonymousOnly = TRUE,
  sortSamples = TRUE
)
```

Arguments

object	An OncoSeqExperiment .
genes	Genes to include; defaults to the ‘topN’ most frequently mutated.
topN	Number of genes when ‘genes’ is ‘NULL’.
nonsynonymousOnly	Restrict to protein-altering variants.
sortSamples	Order samples by mutation load, so the staircase pattern of an oncoprint emerges.

Value

A character matrix with genes in rows and samples in columns, ready for `[ComplexHeatmap::oncoPrint()]` or `[plotOncoprint()]`.

Examples

```
ose <- exampleOncoSeq()
m <- oncoplotData(ose, topN = 5)
m[, 1:4]
```

OncoSeq-generics

OncoSeq generics

Description

Generic functions defining the OncoSeq accessor interface. Methods are documented with the [OncoSeqExperiment](#) class.

Usage

```
mutations(x, ...)

mutations(x) <- value

cnv(x, ...)
```

```

cnv(x) <- value

analyses(x, ...)

analyses(x) <- value

provenance(x, ...)

provenance(x) <- value

```

Arguments

<code>x</code>	An OncoSeqExperiment object.
<code>...</code>	Further arguments passed to methods.
<code>value</code>	Replacement value.

Value

Accessors return the corresponding layer; replacement functions return the modified object.

Examples

```

ose <- exampleOncoSeq()
head(mutations(ose))
cnv(ose)
analyses(ose)
names(provenance(ose))

```

OncoSeqExperiment	<i>Construct an OncoSeqExperiment</i>
-------------------	---------------------------------------

Description

Builds an [OncoSeqExperiment](#) from an expression matrix or an existing [SummarizedExperiment](#), optionally attaching somatic mutations, gene-level copy number and clinical data for the same samples.

Usage

```

OncoSeqExperiment(
  assays,
  colData = NULL,
  rowData = NULL,
  mutations = NULL,
  cnv = NULL,
  provenance = list(),
  ...
)

```

Arguments

assays	A named list of matrices with genes in rows and samples in columns, or a SummarizedExperiment . When a plain list is given the first element is treated as raw counts and should be named ‘counts’.
colData	A ‘data.frame’ or DataFrame of sample annotation whose rows correspond to the columns of ‘assays’. Ignored when ‘assays’ is a SummarizedExperiment .
rowData	Optional feature annotation.
mutations	Optional somatic variant table in long format with columns ‘sample’ and ‘gene’ (see the class documentation).
cnv	Optional gene-level copy-number data: a matrix or SummarizedExperiment with samples in columns.
provenance	Optional named list describing the data source.
...	Further arguments passed to [SummarizedExperiment::SummarizedExperiment()].

Value

An [OncoSeqExperiment](#).

Examples

```
counts <- matrix(rpois(200, 50), nrow = 20,
                 dimnames = list(paste0("G", 1:20), paste0("S", 1:10)))
meta <- data.frame(group = rep(c("tumour", "normal"), each = 5),
                  row.names = colnames(counts))
muts <- data.frame(sample = c("S1", "S2"), gene = c("TP53", "RB1"),
                  variant_class = c("Missense_Mutation", "Nonsense_Mutation"))
ose <- OncoSeqExperiment(list(counts = counts), colData = meta,
                        mutations = muts)
ose
```

OncoSeqExperiment-class

Container for a cancer multi-omic study

Description

‘OncoSeqExperiment’ extends [SummarizedExperiment](#) so that bulk expression data, somatic mutations, gene-level copy-number alterations, clinical follow-up and the results of every analysis performed on them live in a single object. All layers are keyed on the same sample identifiers, which are the column names of the object; validity checking enforces this.

Slots

- mutations** A [DataFrame](#) of somatic variants in long format, one row per variant per sample. Must contain at least the columns 'sample' and 'gene'; 'variant_class' is used by the summarisation functions when present. Sample identifiers must be a subset of 'colnames(object)'.
- cnv** A [SummarizedExperiment](#) of gene-level copy-number calls (genes in rows, samples in columns), or 'NULL'. The first assay is taken to hold discrete GISTIC-style calls in '-2:2'. Column names must be a subset of 'colnames(object)'.
- analyses** A [SimpleList](#) registry of analysis results. Each element is named and carries an 'analysis' attribute describing how it was produced.
- provenance** A list recording the import source and every processing step applied to the object.

Accessors

'mutations()', 'cnv()', 'analyses()' and 'provenance()' return the corresponding slot; each has a replacement counterpart. All [SummarizedExperiment](#) accessors ('assay()', 'colData()', 'rowData()', 'rowRanges()') work unchanged.

Subsetting

'x[i, j]' subsets genes and samples. Subsetting samples also filters the mutation table and the copy-number layer to the retained samples, so the object stays internally consistent.

See Also

[OncoSeqExperiment()] to construct an object, [importCounts()] and [importGDC()] to build one from data.

OncoSeqExperiment-methods

Methods for OncoSeqExperiment objects

Description

Accessors, replacement methods, subsetting and display for [OncoSeqExperiment](#).

Usage

```
## S4 method for signature 'OncoSeqExperiment'
mutations(x, ...)

## S4 replacement method for signature 'OncoSeqExperiment'
mutations(x) <- value

## S4 method for signature 'OncoSeqExperiment'
cnv(x, ...)
```

```
## S4 replacement method for signature 'OncoSeqExperiment'
cnv(x) <- value

## S4 method for signature 'OncoSeqExperiment'
analyses(x, ...)

## S4 replacement method for signature 'OncoSeqExperiment'
analyses(x) <- value

## S4 method for signature 'OncoSeqExperiment'
provenance(x, ...)

## S4 replacement method for signature 'OncoSeqExperiment'
provenance(x) <- value

## S4 method for signature 'OncoSeqExperiment,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'OncoSeqExperiment'
show(object)
```

Arguments

x, object	An OncoSeqExperiment .
...	Further arguments, currently unused.
value	Replacement value.
i, j	Row (feature) and column (sample) subscripts.
drop	Ignored; retained for signature compatibility.

Value

‘mutations()’ a [DataFrame](#); ‘cnv()’ a [SummarizedExperiment](#) or ‘NULL’; ‘analyses()’ a [SimpleList](#); ‘provenance()’ a list; ‘[’ an [OncoSeqExperiment](#); ‘show()’ is called for its side effect.

Examples

```
ose <- exampleOncoSeq()
head(mutations(ose))
cnv(ose)
ose[1:5, 1:4]
```

plotCNVFrequency	<i>Copy-number alteration frequency plot</i>
------------------	--

Description

Copy-number alteration frequency plot

Usage

```
plotCNVFrequency(
  object,
  genes = NULL,
  topN = 20L,
  ampThreshold = 2,
  delThreshold = -2
)
```

Arguments

object	An OncoSeqExperiment with a copy-number layer.
genes	Optional genes to show.
topN	Number of genes with the highest alteration frequency.
ampThreshold, delThreshold	Discrete thresholds for gain and loss.

Value

A ‘ggplot’ object.

Examples

```
ose <- exampleOncoSeq()
plotCNVFrequency(ose, topN = 10)
```

plotEnrichmentBar	<i>Bar chart of enrichment results</i>
-------------------	--

Description

Bar chart of enrichment results

Usage

```
plotEnrichmentBar(object, name = "gsea", topN = 15L, alpha = NULL)
```

Arguments

object	An OncoSeqExperiment .
name	Name of a stored [runGSEA()] or [runORA()] result.
topN	Number of gene sets to show.
alpha	Only sets with an adjusted p-value below this are shown; 'NULL' shows the top sets regardless.

Value

A 'ggplot' object.

Examples

```

ose <- exampleOncoSeq()
ose <- diffExpress(ose, ~ condition,
                  contrast = c("condition", "tumour", "normal"),
                  engine = "edgeR", verbose = FALSE)
sets <- readGMT(system.file("extdata", "example_sets.gmt",
                           package = "OncoSeq"))
ose <- runGSEA(ose, sets)
plotEnrichmentBar(ose, "gsea")

```

plotExpressionHeatmap *Heatmap of selected features*

Description

Draws a per-feature standardised (z-scored) heatmap of the most variable or explicitly chosen features, with optional sample annotation shown as a strip above the matrix.

Usage

```

plotExpressionHeatmap(
  object,
  features = NULL,
  nTopGenes = 50L,
  assay = NULL,
  annotationColumn = NULL,
  scale = TRUE,
  clusterSamples = TRUE
)

```

Arguments

object	An OncoSeqExperiment .
features	Features to draw; defaults to the 'nTopGenes' most variable.
nTopGenes	Number of features when 'features' is 'NULL'.
assay	Assay to use.
annotationColumn	Optional 'colData()' column used to order and annotate samples.
scale	Standardise each feature across samples.
clusterSamples	Order samples by hierarchical clustering within annotation groups.

Value

A 'ggplot' object.

Examples

```
ose <- exampleOncoSeq()
plotExpressionHeatmap(ose, nTopGenes = 20, annotationColumn = "condition")
```

plotKM	<i>Kaplan-Meier curves</i>
--------	----------------------------

Description

Draws the survival curves from a [survivalFit()] result, annotated with the log-rank p-value and the number at risk in each group.

Usage

```
plotKM(fit, showPValue = TRUE, palette = NULL, timeLabel = "Time")
```

Arguments

fit	An object returned by [survivalFit()].
showPValue	Annotate the panel with the log-rank p-value.
palette	Optional vector of colours, one per group.
timeLabel	Axis label for follow-up time.

Value

A 'ggplot' object.

Examples

```
ose <- exampleOncoSeq()
plotKM(survivalFit(ose, "TP53", by = "mutation"))
```

plotMAplot	<i>Mean-difference (MA) plot of a differential-expression result</i>
------------	--

Description

Mean-difference (MA) plot of a differential-expression result

Usage

```
plotMAplot(object, deName = "de", alpha = 0.05, lfcThreshold = 0)
```

Arguments

object	An OncoSeqExperiment .
deName	Name of the stored differential-expression result.
alpha	Adjusted-p-value threshold for calling a feature significant.
lfcThreshold	Absolute log2 fold-change threshold.

Value

A ‘ggplot’ object.

Examples

```
ose <- exampleOncoSeq()
ose <- diffExpress(ose, ~ condition,
                  contrast = c("condition", "tumour", "normal"),
                  engine = "edgeR", verbose = FALSE)
plotMAplot(ose)
```

plotMutationFrequency	<i>Mutation frequency plot with confidence intervals</i>
-----------------------	--

Description

Mutation frequency plot with confidence intervals

Usage

```
plotMutationFrequency(object, topN = 20L, nonsynonymousOnly = TRUE)
```

Arguments

object An [OncoSeqExperiment](#).
 topN Number of genes to show.
 nonsynonymousOnly Restrict to protein-altering variants.

Value

A ‘ggplot’ object.

Examples

```
ose <- exampleOncoSeq()
plotMutationFrequency(ose, topN = 8)
```

plotOncoprint	<i>Oncoprint-style mutation landscape</i>
---------------	---

Description

Tile plot of the matrix returned by [oncoplotData()]: genes in rows, samples in columns, coloured by variant classification, with the per-gene mutation frequency on the right.

Usage

```
plotOncoprint(object, genes = NULL, topN = 20L, nonsynonymousOnly = TRUE)
```

Arguments

object An [OncoSeqExperiment](#).
 genes Genes to show; defaults to the ‘topN’ most frequently mutated.
 topN Number of genes when ‘genes’ is ‘NULL’.
 nonsynonymousOnly Restrict to protein-altering variants.

Value

A ‘ggplot’ object.

Examples

```
ose <- exampleOncoSeq()
plotOncoprint(ose, topN = 8)
```

plotSamplePCA	<i>Principal component plot of samples</i>
---------------	--

Description

Projects samples onto their leading principal components, computed from the most variable features of a log-scale assay, and colours them by a sample annotation.

Usage

```
plotSamplePCA(  
  object,  
  colourBy = NULL,  
  shapeBy = NULL,  
  assay = NULL,  
  nTopGenes = 500L,  
  components = c(1L, 2L),  
  label = FALSE  
)
```

Arguments

object	An OncoSeqExperiment .
colourBy	'colData()' column used for point colour.
shapeBy	Optional 'colData()' column used for point shape.
assay	Assay to use; log2 counts-per-million are derived from counts when no transformed assay is present.
nTopGenes	Number of most variable features used.
components	Length-two integer vector giving which components to draw.
label	Draw sample names next to the points.

Value

A 'ggplot' object.

Examples

```
ose <- exampleOncoSeq()  
plotSamplePCA(ose, colourBy = "condition")
```

`plotVolcano`*Volcano plot of a differential-expression result*

Description

Volcano plot of a differential-expression result

Usage

```
plotVolcano(  
  object,  
  deName = "de",  
  alpha = 0.05,  
  lfcThreshold = 1,  
  nLabel = 10L,  
  labelGenes = NULL  
)
```

Arguments

<code>object</code>	An OncoSeqExperiment .
<code>deName</code>	Name of the stored differential-expression result.
<code>alpha</code>	Adjusted-p-value threshold for calling a feature significant.
<code>lfcThreshold</code>	Absolute log2 fold-change threshold.
<code>nLabel</code>	Number of top features to label, by adjusted p-value.
<code>labelGenes</code>	Explicit features to label, overriding ‘nLabel’.

Value

A ‘ggplot’ object.

Examples

```
ose <- exampleOncoSeq()  
ose <- diffExpress(ose, ~ condition,  
  contrast = c("condition", "tumour", "normal"),  
  engine = "edgeR", verbose = FALSE)  
plotVolcano(ose)
```

readGMT

Read a gene-set collection from a GMT file

Description

Parses a Broad-style GMT file into a named list of character vectors, the format expected by [runGSEA()] and [runORA()].

Usage

```
readGMT(file, minSize = 1L, maxSize = Inf)
```

Arguments

`file` Path to a ‘.gmt’ file.
`minSize, maxSize` Discard sets outside this size range.

Value

A named list of gene identifiers.

See Also

[msigdbrSets()]

Examples

```
gmt <- system.file("extdata", "example_sets.gmt", package = "OncoSeq")
sets <- readGMT(gmt)
lengths(sets)
```

runGSEA

Gene set enrichment analysis on a differential-expression result

Description

Runs [fgsea::fgseaMultilevel()] on features ranked by a statistic taken from a stored differential-expression table. The multilevel algorithm is the current fgsea default and gives accurate p-values far into the tail.

Usage

```
runGSEA(
  object,
  geneSets,
  deName = "de",
  rankBy = c("signed_log_p", "stat", "log2FC"),
  minSize = 5L,
  maxSize = 500L,
  name = "gsea",
  BPPARAM = BiocParallel::SerialParam(),
  ...
)
```

Arguments

<code>object</code>	An OncoSeqExperiment holding a differential-expression result.
<code>geneSets</code>	A named list of gene identifiers; see <code>[readGMT()]</code> .
<code>deName</code>	Name of the stored differential-expression result.
<code>rankBy</code>	Statistic used to rank features. <code>"signed_log_p"</code> is <code>'sign(log2FC) * -log10(pvalue)'</code> , which separates direction from magnitude better than the fold change alone; <code>"log2FC"</code> and <code>"stat"</code> are also available.
<code>minSize, maxSize</code>	Gene-set size limits.
<code>name</code>	Name under which the result is stored.
<code>BPPARAM</code>	A BiocParallelParam instance controlling parallelisation. The default is serial evaluation, which is reproducible and avoids the socket set-up that fails in sandboxed environments; pass <code>[BiocParallel::MulticoreParam()]</code> to use several cores.
<code>...</code>	Further arguments passed to <code>[fgsea::fgseaMultilevel()]</code> .

Value

The object with a result table stored in the registry, carrying columns `'set'`, `'size'`, `'ES'`, `'NES'`, `'pvalue'`, `'padj'`, `'leadingEdge'` and `'method'`, ordered by adjusted p-value.

See Also

`[runORA()]`, `[plotEnrichment()]`

Examples

```
ose <- exampleOncoSeq()
ose <- diffExpress(ose, ~ condition,
  contrast = c("condition", "tumour", "normal"),
  engine = "edgeR", verbose = FALSE)
sets <- readGMT(system.file("extdata", "example_sets.gmt",
  package = "OncoSeq"))
ose <- runGSEA(ose, sets)
```

```
head(getAnalysis(ose, "gsea")[, c("set", "NES", "padj")])
```

runOncoSeq

Run the full OncoSeq workflow in one call

Description

Executes the standard sequence of steps on an existing object: quality metrics, low-expression filtering, normalisation, differential expression, enrichment, mutation burden and a survival screen, storing every result in the analysis registry. Steps whose prerequisites are absent are skipped with a message rather than failing, so the same call works on expression-only and full multi-omic objects.

Usage

```
runOncoSeq(
  object,
  design,
  contrast = NULL,
  geneSets = NULL,
  engine = c("DESeq2", "edgeR", "limma"),
  normalisation = "vst",
  survivalGenes = NULL,
  timeColumn = "os_time",
  eventColumn = "os_event",
  alpha = 0.05,
  verbose = TRUE
)
```

Arguments

object	An OncoSeqExperiment .
design	A one-sided formula for the differential-expression model.
contrast	Contrast specification; see [diffExpress()] .
geneSets	Optional named list of gene sets enabling the enrichment step.
engine	Differential-expression engine.
normalisation	Normalisation method passed to [normalizeCounts()] .
survivalGenes	Genes to carry through [survivalScreen()] ; 'NULL' screens every feature, which is slow on a full transcriptome.
timeColumn, eventColumn	Survival columns in 'colData()'. Survival columns in 'colData()'.
alpha	Significance threshold used throughout.
verbose	Report progress.

Value

The [OncoSeqExperiment](#) with results stored under the names 'de', 'gsea', 'ora' and 'survival_screen', QC columns in 'colData()' and the normalised assay added.

See Also

[diffExpress()], [runGSEA()], [survivalScreen()]

Examples

```
ose <- exampleOncoSeq()
sets <- readGMT(system.file("extdata", "example_sets.gmt",
                           package = "OncoSeq"))
res <- runOncoSeq(ose, ~ condition,
                 contrast = c("condition", "tumour", "normal"),
                 geneSets = sets, engine = "edgeR", verbose = FALSE)
names(analyses(res))
```

runORA

Over-representation analysis of a differentially expressed gene list

Description

Tests whether each gene set is over-represented among the significant features of a stored differential-expression result, using the hypergeometric distribution with the tested features as the universe and Benjamini-Hochberg correction across sets.

Usage

```
runORA(
  object,
  geneSets,
  deName = "de",
  alpha = 0.05,
  lfcThreshold = 0,
  direction = c("both", "up", "down"),
  universe = NULL,
  minSize = 5L,
  maxSize = 500L,
  name = "ora"
)
```

Arguments

object	An OncoSeqExperiment .
geneSets	A named list of gene identifiers.
deName	Name of the stored differential-expression result.
alpha	Adjusted-p-value cut-off defining the significant list.
lfcThreshold	Minimum absolute log2 fold change for inclusion.
direction	Restrict the list to "up", "down" or "both".
universe	Character vector of background features; defaults to all features in the differential-expression table, which is the correct universe because only those were testable.
minSize, maxSize	Gene-set size limits, applied after intersecting each set with the universe.
name	Name under which the result is stored.

Value

The object with a result table in the registry, carrying 'set', 'size', 'overlap', 'expected', 'foldEnrichment', 'pvalue', 'padj', 'genes' and 'method'.

See Also

[runGSEA()]

Examples

```
ose <- exampleOncoSeq()
ose <- diffExpress(ose, ~ condition,
                  contrast = c("condition", "tumour", "normal"),
                  engine = "edgeR", verbose = FALSE)
sets <- readGMT(system.file("extdata", "example_sets.gmt",
                           package = "OncoSeq"))
ose <- runORA(ose, sets, alpha = 0.1)
head(getAnalysis(ose, "ora")[, c("set", "overlap", "padj")])
```

simExample

Simulated cancer multi-omic cohort

Description

The object returned by [exampleOncoSeq()]: 300 features by 40 samples, 20 simulated tumours and 20 simulated normals, with counts, somatic mutations, gene-level copy number and overall survival. The values are simulated; see [exampleOncoSeq()] for the generative model and 'data-raw/make-simExample.R' for the code that produced it.

Usage

```
data(simExample)
```

Format

An [OncoSeqExperiment](#) with 300 features and 40 samples.

Source

Simulated by 'data-raw/make-simExample.R'.

See Also

[blcaExample] for a real cohort.

Examples

```
data(simExample)
simExample
```

survivalFit

Kaplan-Meier analysis of a gene or clinical variable

Description

Splits samples into groups by expression, mutation status, copy-number state or a clinical variable, fits Kaplan-Meier curves and tests them with the log-rank statistic, and reports the hazard ratio from the corresponding Cox model.

Usage

```
survivalFit(
  object,
  feature,
  by = c("auto", "expression", "mutation", "cnv", "clinical"),
  split = c("median", "tertile", "quartile", "optimal"),
  timeColumn = "os_time",
  eventColumn = "os_event",
  assay = NULL,
  minGroupSize = 5L
)
```

Arguments

object	An OncoSeqExperiment .
feature	Gene identifier or 'colData()' column name to stratify on.
by	Layer the feature belongs to: "expression", "mutation", "cnv", "clinical", or "auto" to search them in that order.
split	Grouping rule for continuous features: "median", "tertile", "quartile" or "optimal".
timeColumn, eventColumn	'colData()' columns holding follow-up time and the event indicator (1 = event, 0 = censored).
assay	Expression assay to use; log-scale values are derived from counts when needed.
minGroupSize	Refuse to test when a group is smaller than this.

Details

For expression the split may be at the median, by tertiles, by quartiles (comparing the outer two), or at an optimal cut-point. The optimal cut-point scans candidate thresholds between the 25th and 75th percentiles and keeps the one maximising the log-rank statistic. That search inflates significance, so the reported 'p_logrank_adjusted' applies the Altman approximation; the unadjusted value is reported alongside it and should not be quoted on its own.

Value

A list of class 'oncoseq_survival' with the 'survfit' object, the grouping factor, the log-rank test, the Cox hazard ratio with confidence interval, and the cut-point when one was chosen.

See Also

[coxModel()], [survivalScreen()], [plotKM()]

Examples

```
ose <- exampleOncoSeq()
fit <- survivalFit(ose, "TP53", by = "mutation")
fit$p_logrank
```

survivalScreen

Genome-wide screen for survival-associated genes

Description

Fits a Cox proportional-hazards model for every gene, with expression as a continuous covariate and optional clinical adjustment, and corrects the resulting p-values across genes with Benjamini-Hochberg.

Usage

```
survivalScreen(
  object,
  genes = NULL,
  covariates = NULL,
  timeColumn = "os_time",
  eventColumn = "os_event",
  assay = NULL,
  scale = TRUE,
  minVariance = 0,
  name = "survival_screen"
)
```

Arguments

object	An OncoSeqExperiment .
genes	Genes to test; defaults to all features.
covariates	Optional clinical covariates to adjust for.
timeColumn, eventColumn	Follow-up and event columns.
assay	Expression assay; log-scale values are derived from counts when needed.
scale	Standardise each gene before fitting, so that the hazard ratio refers to a one standard-deviation increase and is comparable across genes.
minVariance	Skip genes whose variance is at or below this value.
name	Name under which the result is stored; 'NULL' returns the table.

Value

The object with a table of 'gene', 'HR', 'ci_lower', 'ci_upper', 'z', 'pvalue' and 'padj', ordered by adjusted p-value.

Examples

```
ose <- exampleOncoSeq()
res <- survivalScreen(ose, genes = rownames(ose)[1:50], name = NULL)
head(res)
```

tumourMutationBurden	<i>Tumour mutation burden</i>
----------------------	-------------------------------

Description

Non-synonymous somatic variants per megabase of sequenced territory for every sample. The default capture size of 38 Mb is the figure conventionally used for whole-exome panels; supply the actual footprint of your assay when it is known.

Usage

```
tumourMutationBurden(  
  object,  
  captureSize = 38,  
  nonsynonymousOnly = TRUE,  
  store = TRUE  
)
```

Arguments

object	An OncoSeqExperiment .
captureSize	Size of the sequenced territory in megabases.
nonsynonymousOnly	Restrict to protein-altering variants.
store	Add a 'tmb' column to 'colData()' and return the object.

Value

The object with a 'tmb' column, or a 'data.frame' of burdens when 'store = FALSE'.

Examples

```
ose <- exampleOncoSeq()  
tmb <- tumourMutationBurden(ose, store = FALSE)  
summary(tmb$tmb)
```

Index

- * **datasets**
 - blcaExample, [5](#)
 - simExample, [46](#)
- [, OncoSeqExperiment, ANY, ANY, ANY-method
(OncoSeqExperiment-methods), [32](#)
- alignLayers, [3](#)
- analyses (OncoSeq-generics), [29](#)
- analyses, OncoSeqExperiment-method
(OncoSeqExperiment-methods), [32](#)
- analyses<- (OncoSeq-generics), [29](#)
- analyses<-, OncoSeqExperiment-method
(OncoSeqExperiment-methods), [32](#)
- asOncoSeq, [4](#)
- BiocParallelParam, [43](#)
- blcaExample, [5](#)
- cbioProfiles, [6](#)
- cbioStudies, [7](#)
- cnv (OncoSeq-generics), [29](#)
- cnv, OncoSeqExperiment-method
(OncoSeqExperiment-methods), [32](#)
- cnv<- (OncoSeq-generics), [29](#)
- cnv<-, OncoSeqExperiment-method
(OncoSeqExperiment-methods), [32](#)
- cnvExpressionCorrelation, [7](#)
- cnvFrequency, [8](#)
- cnvSummary, [9](#)
- compareEngines, [10](#)
- coxModel, [11](#)
- DataFrame, [12](#), [19](#), [22](#), [31–33](#), [41](#)
- detectOutliers, [12](#)
- diffExpress, [13](#)
- exampleOncoSeq, [14](#)
- filterLowExpression, [15](#)
- getAnalysis, [16](#)
- importCBioPortal, [17](#)
- importCounts, [19](#)
- importGDC, [20](#)
- importMAF, [21](#)
- importTximport, [22](#)
- integrateMutationExpression, [23](#)
- msigdbSets, [24](#)
- mutationFrequency, [25](#)
- mutations (OncoSeq-generics), [29](#)
- mutations, OncoSeqExperiment-method
(OncoSeqExperiment-methods), [32](#)
- mutations<- (OncoSeq-generics), [29](#)
- mutations<-, OncoSeqExperiment-method
(OncoSeqExperiment-methods), [32](#)
- mutationStatus, [26](#)
- mutationSummary, [27](#)
- normalizeCounts, [27](#)
- oncoplotData, [28](#)
- OncoSeq-generics, [29](#)
- OncoSeqExperiment, [4](#), [5](#), [8–23](#), [25–29](#), [30](#),
[30–41](#), [43–50](#)
- OncoSeqExperiment-class, [31](#)
- OncoSeqExperiment-methods, [32](#)
- plotCNVFrequency, [34](#)
- plotEnrichmentBar, [34](#)
- plotExpressionHeatmap, [35](#)
- plotKM, [36](#)
- plotMAplot, [37](#)
- plotMutationFrequency, [37](#)
- plotOncoprint, [38](#)
- plotSamplePCA, [39](#)
- plotVolcano, [40](#)
- provenance (OncoSeq-generics), [29](#)
- provenance, OncoSeqExperiment-method
(OncoSeqExperiment-methods), [32](#)
- provenance<- (OncoSeq-generics), [29](#)

provenance<- , OncoSeqExperiment-method
 (OncoSeqExperiment-methods), [32](#)

qcMetrics, [41](#)

readGMT, [42](#)
runGSEA, [42](#)
runOncoSeq, [44](#)
runORA, [45](#)

show, OncoSeqExperiment-method
 (OncoSeqExperiment-methods), [32](#)

simExample, [46](#)
SimpleList, [32](#), [33](#)
SummarizedExperiment, [4](#), [30–33](#)
survivalFit, [47](#)
survivalScreen, [48](#)

tumourMutationBurden, [49](#)