

# Package: PDATK (via r-universe)

July 4, 2024

**Type** Package

**Title** Pancreatic Ductal Adenocarcinoma Tool-Kit

**Version** 1.13.0

**Date** `r Sys.Date()`

**Description** Pancreatic ductal adenocarcinoma (PDA) has a relatively poor prognosis and is one of the most lethal cancers. Molecular classification of gene expression profiles holds the potential to identify meaningful subtypes which can inform therapeutic strategy in the clinical setting. The Pancreatic Cancer Adenocarcinoma Tool-Kit (PDATK) provides an S4 class-based interface for performing unsupervised subtype discovery, cross-cohort meta-clustering, gene-expression-based classification, and subsequent survival analysis to identify prognostically useful subtypes in pancreatic cancer and beyond. Two novel methods, Consensus Subtypes in Pancreatic Cancer (CSPC) and Pancreatic Cancer Overall Survival Predictor (PCOSP) are included for consensus-based meta-clustering and overall-survival prediction, respectively. Additionally, four published subtype classifiers and three published prognostic gene signatures are included to allow users to easily recreate published results, apply existing classifiers to new data, and benchmark the relative performance of new methods. The use of existing Bioconductor classes as input to all PDATK classes and methods enables integration with existing Bioconductor datasets, including the 21 pancreatic cancer patient cohorts available in the MetaGxPancreas data package. PDATK has been used to replicate results from Sandhu et al (2019) [<https://doi.org/10.1200/cci.18.00102>] and an additional paper is in the works using CSPC to validate subtypes from the included published classifiers, both of which use the data available in MetaGxPancreas. The inclusion of subtype centroids and prognostic gene signatures from these and other publications will enable researchers and clinicians to classify novel patient gene expression data, allowing the direct clinical application of the classifiers included in PDATK.

Overall, PDATAK provides a rich set of tools to identify and validate useful prognostic and molecular subtypes based on gene-expression data, benchmark new classifiers against existing ones, and apply discovered classifiers on novel patient data to inform clinical decision making.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.1), SummarizedExperiment

**Imports** data.table, MultiAssayExperiment, ConsensusClusterPlus, igraph, ggplotify, matrixStats, RColorBrewer, clusterRepro, CoreGx, caret, survminer, methods, S4Vectors, BiocGenerics, survival, stats, plyr, dplyr, MatrixGenerics, BiocParallel, rlang, piano, scales, survcomp, genefu, ggplot2, switchBox, reportROC, pROC, verification, utils

**Suggests** testthat (>= 3.0.0), msigdb, BiocStyle, rmarkdown, knitr, HDF5Array

**VignetteBuilder** knitr

**Roxygen** list(markdown = TRUE, r6=FALSE)

**biocViews** GeneExpression, Pharmacogenetics, Pharmacogenomics, Software, Classification, Survival, Clustering, GenePrediction

**BugReports** <https://github.com/bhk1ab/PDATAK/issues>

**RoxygenNote** 7.1.2

**Collate** 'class-S4Model.R' 'class-CohortList.R'  
 'class-SurvivalExperiment.R' 'class-SurvivalModel.R'  
 'class-ClinicalModel.R' 'class-ConsensusMetaclusteringModel.R'  
 'class-CoxModel.R' 'class-GeneFuModel.R'  
 'class-ModelComparison.R' 'class-NCSModel.R' 'class-PCOSP.R'  
 'class-RGAModel.R' 'class-RLSModel.R' 'classUnions.R' 'data.R'  
 'globals.R' 'methods-assignSubtypes.R'  
 'methods-barPlotModelComparison.R' 'methods-coerce.R'  
 'methods-compareModels.R'  
 'methods-densityPlotModelComparison.R'  
 'methods-dropNotCensored.R' 'methods-findCommonGenes.R'  
 'methods-findCommonSamples.R' 'methods-forestPlot.R'  
 'methods-getTopFeatures.R' 'methods-merge.R'  
 'methods-normalize.R' 'methods-plotNetworkGraph.R'  
 'methods-plotROC.R' 'methods-plotSurvivalCurves.R'  
 'methods-predictClasses.R' 'methods-rankFeatures.R'  
 'methods-runGSEA.R' 'methods-subset.R' 'methods-trainModel.R'  
 'methods-validateModel.R' 'utilities.R'

**Repository** <https://bioc.r-universe.dev>

**RemoteUrl** <https://github.com/bioc/PDATAK>

**RemoteRef** HEAD

**RemoteSha** e15f1cf85c0bb52ac4367ce0640d844bc740e130

## Contents

assignColDataColumn . . . . .	6
assignSubtypes . . . . .	6
assignSubtypes,CohortList,list-method . . . . .	7
assignSubtypes,SurvivalExperiment,data.frame-method . . . . .	8
barPlotModelComparison . . . . .	9
barPlotModelComparison,ClinicalModel,PCOSP_or_RLS_or_RGA-method . . . . .	10
birnbaum . . . . .	11
chen . . . . .	12
ClinicalModel . . . . .	12
ClinicalModel-class . . . . .	13
CohortList . . . . .	13
CohortList-class . . . . .	14
cohortSubtypeDFs . . . . .	14
compareModels . . . . .	14
compareModels,ModelComparison,SurvivalModel-method . . . . .	15
compareModels,SurvivalModel,SurvivalModel-method . . . . .	16
ConsensusMetaclusteringModel . . . . .	17
ConsensusMetaclusteringModel-class . . . . .	18
CoxModel . . . . .	18
CSPC_MAE . . . . .	19
densityPlotModelComparison . . . . .	19
densityPlotModelComparison,PCOSP_or_RLS_or_RGA,PCOSP_or_RLS_or_RGA-method . . . . .	20
dropNotCensored . . . . .	21
dropNotCensored,CohortList-method . . . . .	21
dropNotCensored,SurvivalExperiment-method . . . . .	22
existingClassifierData . . . . .	23
findCommonGenes . . . . .	23
findCommonGenes,CohortList-method . . . . .	24
findCommonGenes,MultiAssayExperiment-method . . . . .	24
findCommonSamples . . . . .	25
findCommonSamples,CohortList-method . . . . .	26
forestPlot . . . . .	26
forestPlot,ModelComparison-method . . . . .	27
forestPlot,PCOSP_or_ClinicalModel-method . . . . .	28
GeneFuModel . . . . .	30
GeneFuModel-class . . . . .	31
getModelSeed . . . . .	31
getModelSeed,SurvivalModel-method . . . . .	32
getTopFeatures . . . . .	32
getTopFeatures,MultiAssayExperiment-method . . . . .	33
getTopFeatures,PCOSP-method . . . . .	33
getTopFeatures,SummarizedExperiment-method . . . . .	34
haiderSigScores . . . . .	35
hasColDataColumns . . . . .	35
merge,SurvivalExperiment,SurvivalExperiment-method . . . . .	36
ModelComparison . . . . .	36

ModelComparison-class	37
modelParams	38
modelParams,S4Model-method	38
models	39
models,S4Model-method	39
models,SurvivalModel-method	40
models<- ,SurvivalModel,SimpleList-method	40
NCSModel-class	41
NetworkCommunitySearchModel	41
normalize,data.frame_or_matrix-method	42
normalize,DFrame-method	42
normalize,MultiAssayExperiment-method	43
normalize,SummarizedExperiment-method	44
normalsMAE	45
optimalKMinimizeAmbiguity	45
PCOSP	46
PCOSP-class	47
PCOSP_or_ClinicalModel-class	47
PCOSP_or_RLS_or_RGA-class	47
plotNetworkGraph	48
plotNetworkGraph,NCSModel-method	48
plotROC	49
plotROC,PCOSP-method	49
plotSurvivalCurves	50
plotSurvivalCurves,CoxModel-method	50
predictClasses	51
predictClasses,CohortList,ClinicalModel-method	52
predictClasses,CohortList,GeneFuModel-method	53
predictClasses,CohortList,PCOSP_or_RLS_or_RGA-method	53
predictClasses,ConsensusMetaclusteringModel,ANY-method	54
predictClasses,NCSModel,ANY-method	55
predictClasses,SurvivalExperiment,ClinicalModel-method	55
predictClasses,SurvivalExperiment,GeneFuModel-method	56
predictClasses,SurvivalExperiment,PCOSP_or_RLS_or_RGA-method	57
preprocessCaret	58
RandomGeneAssignmentModel	59
RandomLabelShufflingModel	59
rankFeatures	60
rankFeatures,MultiAssayExperiment-method	61
rankFeatures,SummarizedExperiment-method	62
removeColDataFactorColumns	63
removeFactorColumns	63
renameColDataColumns	64
renameColumns	64
RGAModel-class	65
RLSModel-class	65
runGSEA	65
runGSEA,PCOSP,data.frame-method	66

S4Model-class . . . . .	67
sampleClinicalModel . . . . .	67
sampleCohortList . . . . .	68
sampleCGCmicro . . . . .	68
samplePCOSPmodel . . . . .	69
samplePCOSPpredList . . . . .	69
samplePCSI survExp . . . . .	70
sampleRGAModel . . . . .	70
sampleRLSmodel . . . . .	71
sampleTrainedPCOSPmodel . . . . .	71
sampleValPCOSPmodel . . . . .	72
show,S4Model-method . . . . .	72
subset,CohortList-method . . . . .	73
SurvivalExperiment . . . . .	73
SurvivalExperiment-class . . . . .	74
SurvivalModel . . . . .	75
SurvivalModel-class . . . . .	75
trainData . . . . .	76
trainData,S4Model-method . . . . .	77
trainData<- . . . . .	77
trainModel . . . . .	78
trainModel,ClinicalModel-method . . . . .	79
trainModel,ConsensusMetaclusteringModel-method . . . . .	80
trainModel,CoxModel-method . . . . .	81
trainModel,GeneFuModel-method . . . . .	81
trainModel,NCSModel-method . . . . .	82
trainModel,PCOSP-method . . . . .	82
trainModel,RGAModel-method . . . . .	83
trainModel,RLSModel-method . . . . .	84
validateModel . . . . .	85
validateModel,ClinicalModel,CohortList-method . . . . .	86
validateModel,ClinicalModel,SurvivalExperiment-method . . . . .	87
validateModel,ConsensusMetaclusteringModel,ConsensusMetaclusteringModel-method . . . . .	88
validateModel,GeneFuModel,CohortList-method . . . . .	89
validateModel,GeneFuModel,SurvivalExperiment-method . . . . .	90
validateModel,PCOSP_or_RLS_or_RGA,CohortList-method . . . . .	90
validateModel,PCOSP_or_RLS_or_RGA,SurvivalExperiment-method . . . . .	91
validationData . . . . .	92
validationData,S4Model-method . . . . .	93
validationData,SurvivalModel-method . . . . .	93
validationData<- . . . . .	94
validationData<- ,SurvivalModel,CohortList-method . . . . .	94
validationStats . . . . .	95
validationStats,S4Model-method . . . . .	96
validationStats,SurvivalModel-method . . . . .	96
validationStats<- . . . . .	97
validationStats<- ,S4Model,DFrame_or_data.frame_data.table_or_NULL-method . . . . .	97
validationStats<- ,SurvivalModel,data.frame-method . . . . .	98

**Index****99**


---

assignColDataColumn	<i>assignColDataColumn</i> Assign a new column directly to the colData slot of a SummarizedExperiment.
---------------------	--

---

**Description**

assignColDataColumn

Assign a new column directly to the colData slot of a SummarizedExperiment.

**Usage**

```
assignColDataColumn(SE, colname, values)
```

**Arguments**

SE	A SummarizedExperiment object to assign colData columns to
colname	The name of the column to assign values to.
values	The values to assign to the col

**Value**

The SE object with the new column in the objects colData slot.

**Examples**

```
SE <- SummarizedExperiment(matrix(rnorm(100), 10, 10))
updatedSE <- assignColDataColumn(SE, 'test', rep(1, 10))
```

---

assignSubtypes	<i>Assign Sample Subtypes to an S4 Object</i>
----------------	---

---

**Description**

Assign Sample Subtypes to an S4 Object

**Usage**

```
assignSubtypes(object, subtypes, ...)
```

**Arguments**

object	An S4 object containing a slot representing samples or patients.
subtypes	A mapping to assign subtypes to the samples or patients in the object.
...	Allow new parameters to be defined for this generic.

**Value**

object with subtypes assigned to the sample metadata and the isSubtyped metadata item set to TRUE.

**Examples**

```
data(sampleICGcmicro)
data(cohortSubtypeDFs)

cohortList <- assignSubtypes(sampleICGcmicro,
  subtypes=cohortSubtypeDFs$ICGCMICRO,
  sampleCol='sample_name',
  subtypeCol='subtype')
```

---

assignSubtypes, CohortList, list-method

*Assign Subtype Annotations to a SurvivalExperiment Object*

---

**Description**

Assign Subtype Annotations to a SurvivalExperiment Object

**Usage**

```
## S4 method for signature 'CohortList,list'
assignSubtypes(
  object,
  subtypes,
  ...,
  sampleCol = "sample_name",
  subtypeCol = "subtype"
)
```

**Arguments**

object	A CohortList.
subtypes	A list of data.frame objects, one per cohort, with to subtypes to assign to the colData slot of cohorts with a matching name.
...	Catch unnamed parameters. Not used.
sampleCol	A character vector indicating the name of the column with sample identifiers in the subtype column. Must match the name of the sample identifier in colData.
subtypeCol	A character vector indicating the name of the column with the subtype labels in the subtypes data.frame.

**Value**

The CohortList with the subtypes in the subtypes column of the colData slot and a metadata item, hasSubtypes, set to TRUE for each SurvivalExperiment.

**Examples**

```
data(sampleCohortList)
data(cohortSubtypeDFs)

cohortList <- assignSubtypes(sampleCohortList,
  subtypes=cohortSubtypeDFs[names(sampleCohortList)],
  sampleCol='sample_name',
  subtypeCol='subtype')
```

---

assignSubtypes, SurvivalExperiment, data.frame-method

*Assign Subtype Annotations to a SurvivalExperiment Object*

---

**Description**

Assign Subtype Annotations to a SurvivalExperiment Object

**Usage**

```
## S4 method for signature 'SurvivalExperiment,data.frame'
assignSubtypes(
  object,
  subtypes,
  ...,
  sampleCol = "sample_name",
  subtypeCol = "subtype"
)
```

**Arguments**

object	A SurvivalExperiment object where the subtype annotations will be added to the colData slot as the subtype column.
subtypes	A data.frame with
...	Force subsequent arguments to be named. Not used.
sampleCol	A character vector specifying the name of the column containing the sample names. These must match the colnames of the SurvivalExperiment. If the sample names are the rownames of the subtypes data.frame then set this parameter to 'rownames'. Defaults to 'sample_name'.
subtypeCol	A character vector specifying the name of the subtype column in the subtypes data.frame object. Defaults to 'subtype'.



**Value**

The SurvivalExperiment with the subtypes in the subtypes column of the colData slot and a metadata item, hasSubtypes, set to TRUE.

**Examples**

```
data(sampleICGcmicro)
data(cohortSubtypeDFs)

cohortList <- assignSubtypes(sampleICGcmicro,
  subtypes=cohortSubtypeDFs$ICGCMICRO,
  sampleCol='sample_name',
  subtypeCol='subtype')
```

---

barPlotModelComparison

*Make A Bar Plot Comparing Performance Between Two S4 Objects  
Representing Mathematical Models.*

---

**Description**

Make A Bar Plot Comparing Performance Between Two S4 Objects Representing Mathematical Models.

**Usage**

```
barPlotModelComparison(model1, model2, ...)
```

**Arguments**

model1	AnS4 object containing results of a mathematical model
model2	An S4 object containing results of a different mathematical model, but with the same or overlapping samples.
...	Allow new parameters to be defined for this generic.

**Value**

A bar plot comparing some aspect of model1 and model2

**Examples**

```

data(sampleCohortList)
data(sampleValPCOSPmodel)
data(sampleICGCmicro)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Setup the models
set.seed(1987)
clinicalModel <- ClinicalModel(sampleICGCmicro,
  formula='prognosis ~ sex + age + T + N + M + grade',
  randomSeed=1987)

# Train the models
trainedClinicalModel <- trainModel(clinicalModel)

# Make predictions
clinicalPredCohortList <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)

# Validate the models
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=clinicalPredCohortList)

# Plot the comparison
modelCompBarPlot <- barPlotModelComparison(validatedClinicalModel,
  sampleValPCOSPmodel, stat='AUC')

```

---

```
barPlotModelComparison, ClinicalModel, PCOSP_or_RLS_or_RGA-method
```

*Make a Bar Plot Comparison Model Performance Between a ClinicalModel and a PCOSP, RLSModel or RGAModel object.*

---

**Description**

Make a Bar Plot Comparison Model Performance Between a ClinicalModel and a PCOSP, RLSModel or RGAModel object.

**Usage**

```
## S4 method for signature 'ClinicalModel, PCOSP_or_RLS_or_RGA'
barPlotModelComparison(model1, model2, stat, ...)
```

**Arguments**

model1            A ClinicalModel object.  
model2            A PCOSP or RLSModel or RGAModel object.

`stat` A character vector specifying which statistic to compare the models using. Options are 'AUC', 'log\_D\_index' or 'concordance\_index'.  
`...` Not used.

**Value**

A ggplot2 object showing a barplot coloured by the model and comparing the stat between all cohorts that both models were validated with.

**Examples**

```
data(sampleValPCOSPmodel)
data(sampleCohortList)
data(sampleICGCmicro)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Setup the models
clinicalModel <- ClinicalModel(sampleICGCmicro,
  formula='prognosis ~ sex + age + T + N + M + grade',
  randomSeed=1987)

# Train the models
trainedClinicalModel <- trainModel(clinicalModel)

# Make predictions
clinicalPredCohortList <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)

# Validate the models
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=clinicalPredCohortList)

# Plot the comparison
modelCompBarPlot <- barPlotModelComparison(validatedClinicalModel,
  sampleValPCOSPmodel, stat='AUC')
```

---

 birnbaum

---

*Published classifier gene signature for Birnbaum*


---

**Description**

The genes and coefficients for the gene signature from Birnbaum *et al.* (2017)

**Examples**

```
# Loads chen, birnbaum and haiderSigScores objects
data(existingClassifierData)
```

---

chen	<i>Published classifier gene signature for Chen</i>
------	---

---

**Description**

The genes and coefficients for the gene signature from Chen *et al.* (2015)

**Examples**

```
# Loads chen, birnbaum and haiderSigScores objects
data(existingClassifierData)
```

---

ClinicalModel	<i>Constructor for the ClinicalModel Class</i>
---------------	--

---

**Description**

Constructor for the ClinicalModel Class

**Usage**

```
ClinicalModel(trainData, formula, minDaysSurvived = 365, ..., randomSeed)
```

**Arguments**

trainData	A SurvivalExperiment or CohortList object to construct a clinical model using
formula	A formula object or a character vector coercible to one. All columns specified in the formula must be in the colData slot of the all SurvivalExperiments in trainData.
minDaysSurvived	An integer specifying the minimum number of days required to be 'good' prognosis. Default is 365.
...	Force all subsequent parameters to be named. Not used.
randomSeed	An integer randomSeed that was used to train the model. Users should specify this when initializing a model to ensure reproducibility.

**Value**

A ClinicalModel object.

**Examples**

```
data(sampleICGcmicro)
set.seed(1987)
clinicalModel <- ClinicalModel(sampleICGcmicro,
  formula='prognosis ~ sex + age + T + N + M + grade', randomSeed=1987)
```

---

ClinicalModel-class      *ClinicalModel Class Definition*

---

**Description**

An S4 class with a number of predefined methods for accessing slots relevant to a survival model. More specific model types will inherit from this class for their accessor methods and constructor.

**Examples**

```
data(sampleICGcmicro)
set.seed(1987)
survModel <- SurvivalModel(sampleICGcmicro, minDaysSurvived=385,
  randomSeed=1987)
```

---

CohortList                      *Constructor for the CohortList class, a specialized list for storing SurvivalExperiment objects.*

---

**Description**

Constructor for the CohortList class, a specialized list for storing SurvivalExperiment objects.

**Usage**

```
CohortList(..., mDataTypes)
```

**Arguments**

...                      One or more SurvivalExperiment objects.

mDataTypes              A character vector with the same length as ... which indicates the molecular data type in each cohort. This will be assigned to mcols for the CohortList as well as to the metadata of each item in the list as mDataType.

**Value**

A CohortList object containing one or more SurvivalExperiment objects.

**Examples**

```
data(sampleICGcmicro)
set.seed(1987)
cohortList <- CohortList(list(survExp1=sampleICGcmicro,
  survExp2=sampleICGcmicro), mDataTypes=c('rna_micro', 'rna_micro'))
```

---

CohortList-class	<i>CohortList Class Definition</i>
------------------	------------------------------------

---

**Description**

A list containing only SurvivalExperiment objects.

---

cohortSubtypeDFs	<i>A list of sample subtypes for the data in sampleCohortList</i>
------------------	---

---

**Description**

A list of data.frames containing clinical subtypes for the data in the cohortSubtypeDFs object.

**Examples**

```
data(cohortSubtypeDFs)
```

---

compareModels	<i>Compare Two Mathematical Models Represented as S4 Objects</i>
---------------	--

---

**Description**

Compare Two Mathematical Models Represented as S4 Objects

**Usage**

```
compareModels(model1, model2, ...)
```

**Arguments**

model1	A S4 object representing some kind of mathematical model.
model2	A S4 object representing some kind of mathematical model.
...	Allow new parameters to be defined for this generic.

**Value**

A S4 object with statistics about the performance of each model.

**Examples**

```
data(sampleValPCOSModel)
data(sampleClinicalModel)
data(sampleCohortList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train the model
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Predict risk/risk-class
ClinicalPredPCSI <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)

# Validate the models
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=ClinicalPredPCSI)

# Compare the models
modelComp <- compareModels(sampleValPCOSModel, validatedClinicalModel)
head(modelComp)
```

---

compareModels, ModelComparison, SurvivalModel-method

*Compare Two SurvivalModel Objects, Returning A ModelComparison Object With Statistics Comparing the Performance of Each Model.*

---

**Description**

Compare Two SurvivalModel Objects, Returning A ModelComparison Object With Statistics Comparing the Performance of Each Model.

**Usage**

```
## S4 method for signature 'ModelComparison, SurvivalModel'
compareModels(model1, model2, model2Name)
```

**Arguments**

model1	An object inheriting from the SurvivalModel class.
model2	Another object inheriting from the SurvivalModel class
model2Name	A character vector with the name of the second model.

**Value**

A ModelComparison object with statistics comparing the two models.

**Examples**

```

data(sampleValPCOSPmodel)
data(sampleClinicalModel)
data(sampleCohortList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train the models
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Predict risk/risk-class
clinicalPredCohortL <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)

# Validate the models
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=clinicalPredCohortL)

# Compare the models
modelComp <- compareModels(sampleValPCOSPmodel, validatedClinicalModel)
head(modelComp)

# Compare model comparison to another model
modelCompComp <- compareModels(modelComp, sampleValPCOSPmodel)

```

---

compareModels, SurvivalModel, SurvivalModel-method

*Compare Two SurvivalModel Objects, Returning A ModelComparison Object With Statistics Comparing the Performance of Each Model.*

---

**Description**

Compare Two SurvivalModel Objects, Returning A ModelComparison Object With Statistics Comparing the Performance of Each Model.

**Usage**

```

## S4 method for signature 'SurvivalModel, SurvivalModel'
compareModels(model1, model2, modelNames)

```



**Arguments**

model1	An object inheriting from the SurvivalModel class.
model2	Another object inheriting from the SurvivalModel class
modelName	Optional character vector with a name for each model. Defaults to the class of the model plus 1 and 2 if missing.

**Value**

A ModelComparison object with statistics comparing the two models.

**Examples**

```

data(sampleValPCOSModel)
data(sampleClinicalModel)
data(sampleCohortList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train the model
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Predict risk/risk-class
ClinicalPredPCSI <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)

# Validate the models
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=ClinicalPredPCSI)

# Compare the models
modelComp <- compareModels(sampleValPCOSModel, validatedClinicalModel)
head(modelComp)

```

---

ConsensusMetaclusteringModel

*Constructor for a ConsensusClusterModel Object.*

---

**Description**

Constructor for a ConsensusClusterModel Object.

**Usage**

```
ConsensusMetaclusteringModel(trainData, ..., randomSeed)
```

**Arguments**

trainData	A MultiAssayExperiment or SummarizedExperiment containing molecular data to be used for consensus clustering with ConsensusClusterPlus::ConsensusClusterPlus
...	Force subsequent parameters to be named. Not used.
randomSeed	An integer randomSeed that will be passed to the randomSeed parameter of the ConsensusClusterPlus::ConsensusClusterPlus function when training the model.

**Value**

A ConsensusMetaclusteringModel object containing the training data and ready to be trained.

**See Also**

[ConsensusClusterPlus::ConsensusClusterPlus](#)

**Examples**

```
data(CSPC_MAE)
set.seed(1987)
conMetaclustModel <- ConsensusMetaclusteringModel(CSPC_MAE, randomSeed=1987)
```

---

ConsensusMetaclusteringModel-class

*An S4Model Containing Molecular Data to be Consensus Clustered*

---

**Description**

An S4 wrapper class providing an interface to the ConsensusClusterPlus function from the package with the same name.

---

CoxModel

*CoxModel constructor*

---

**Description**

Build a CoxModel object from a MultiAssayExperiment of SurvivalExperiments. Allows easy application of the survival::coxph function to many SurvivalExperiment at once, assuming they share the survivalPredictor column.

**Usage**

```
CoxModel(object, survivalPredictor = "metacluster_labels")
```

**Arguments**

`object` A `MultiAssayExperiment` containing only `SurvivalExperiment` objects.

`survivalPredictor` A character vector indicating the name of the one or more columns in the `colData` slot of each `SurvivalExperiment` to use for testing survival differences between different groups. Must be a valid column in the `colData` of ALL experiments.

**Value**

A `CoxModel` object, with `object` in the `trainData` slot.

**Examples**

```
library(MultiAssayExperiment)
data(CSPC_MAE)
experiments(CSPC_MAE) <- endoapply(experiments(CSPC_MAE), SurvivalExperiment,
  event_occurred='vital_status', survival_time='days_to_death')
coxModel <- CoxModel(CSPC_MAE, survivalPredictor='sample_type')
```

---

CSPC_MAE	<i>A MultiAssayExperiment containing cohorts of pancreatic cancer patients, for use in package examples.</i>
----------	--

---

**Description**

A `MultiAssayExperiment` containing cohorts of pancreatic cancer patients, for use in package examples.

**Examples**

```
data(CSPC_MAE)
```

---

<code>densityPlotModelComparison</code>	<i>Render A Density Plot of Model Performance for an S4 Object</i>
---	--

---

**Description**

Render A Density Plot of Model Performance for an S4 Object

**Usage**

```
densityPlotModelComparison(object, refModel, ...)
```

**Arguments**

object            An S4 object representing a statistical or ML model.  
 refModel        An S4 object representing a statistical or ML model to compare object against.  
 ...              Allow additional parameters to be defined for this generic.

**Value**

A ggplot object containing the plot.

---

*densityPlotModelComparison, PCOSP\_or\_RLS\_or\_RGA, PCOSP\_or\_RLS\_or\_RGA-method*  
*Render a Density Plot Comparing Model Performance Between Two*  
*PCOSP, RLSModel or RGAModel object.*

---

**Description**

Render a Density Plot Comparing Model Performance Between Two PCOSP, RLSModel or RGAModel object.

**Usage**

```
## S4 method for signature 'PCOSP_or_RLS_or_RGA, PCOSP_or_RLS_or_RGA'
densityPlotModelComparison(
  object,
  refModel,
  ...,
  title,
  xlab,
  ylab,
  mDataTypeLabels
)
```

**Arguments**

object            A PCOSP, RLSModel or RGAModel object.  
 refModel        A PCOSP, RLSModel or RGAModel object to compare performance against.  
 ...              Catch unnamed parameters. Not used.  
 title            Optional character vector with plot title.  
 xlab            Optional character vector with x-axis label.  
 ylab            Optional character vector with y-axis label.  
 mDataTypeLabels  
                  Optional character vector whos names are one or more existing mDataTypes  
                  in object and refModel and whos values are the desired mDataType labels in  
                  the plot facets.

**Value**

A ggplot object with a density plot of model AUCs for object and a vertical line for the average AUC of refModel, faceted by mDataType.

---

dropNotCensored	<i>Remove Censored Patient Samples from An S4 Object.</i>
-----------------	---

---

**Description**

Remove Censored Patient Samples from An S4 Object.

**Usage**

```
dropNotCensored(object, ...)
```

**Arguments**

object	An S4 object containing survival data which needs to have patients who were not censored before some criteria.
...	Allow new parameters to be defined for this generic.

**Value**

S4 The object subset to only those patients which pass the censoring criteria.

**Examples**

```
data(sampleICGcmicro)
ICGcmicro <- dropNotCensored(sampleICGcmicro)
```

---

dropNotCensored, CohortList-method	<i>Remove Censored Patients from Each SurvivalExperiemnt in a CohortList</i>
------------------------------------	--

---

**Description**

Remove Censored Patients from Each SurvivalExperiemnt in a CohortList

**Usage**

```
## S4 method for signature 'CohortList'
dropNotCensored(object, minDaysSurvived = 365)
```

**Arguments**

`object` A CohortList for which to drop patients who died before each SurvivalExperiment item a specified date.

`minDaysSurvived` An integer specifying the minimum number of days a patient needs to have survived to be included in the cohort.

**Value**

The CohortList with censored samples removed.

**Examples**

```
data(sampleCohortList)
valCohortList <- dropNotCensored(sampleCohortList)
```

---

`dropNotCensored, SurvivalExperiment-method`

*Remove Censored Patients from A SurvivalExperiment Object*

---

**Description**

Remove Censored Patients from A SurvivalExperiment Object

**Usage**

```
## S4 method for signature 'SurvivalExperiment'
dropNotCensored(object, minDaysSurvived = 365)
```

**Arguments**

`object` A SurvivalExperiment to censor.

`minDaysSurvived` An integer specifying the minimum number of days a patient needs to have survived to be included in the cohort.

**Details**

Censored means no event before end of measurement. Since we want not censored, we keep patients who had an event before `minDaysSurvived`. Therefore we keep individuals surviving  $>$  `minDaysSurvived`, or who had an event (died) before `minDaysSurvived`.

**Value**

The SurvivalExperiment with censored samples removed.

**Examples**

```
data(sampleICGcMicro)
ICGcMicro <- dropNotCensored(sampleICGcMicro)
```

---

```
existingClassifierData
  existingClassifierData
```

---

**Description**

existingClassifierData

**Value**

Three objects:

- chen: The genes and coefficients for the gene signature from Chen *et al.* (2015)
- birnbaum: The genes and coefficients for the gene signature from Birnbaum *et al.* (2017)
- haiderSigScores: The classifier risk scores from Haider *et al.* (2014)

**Examples**

```
# Loads chen, birnbaum and haiderSigScores objects
data(existingClassifierData)
```

---

```
findCommonGenes      Find the common genes in an S4 object.
```

---

**Description**

Find the common genes in an S4 object.

**Usage**

```
findCommonGenes(object, ...)
```

**Arguments**

object	An S4 object to find common genes for.
...	Allow new parameters to be defined for this generic.

**Value**

A character vector of common gene names.

**Examples**

```
data(sampleCohortList)
commonGenes <- findCommonGenes(sampleCohortList)
head(commonGenes)
```

---

`findCommonGenes,CohortList-method`

*Intersect Gene Names for All SurvivalExperiments in a CohortList*

---

**Description**

Intersect Gene Names for All SurvivalExperiments in a CohortList

**Usage**

```
## S4 method for signature 'CohortList'
findCommonGenes(object)
```

**Arguments**

`object` A CohortList of SurvivalExperiments to find common genes between.

**Value**

A character vector of genes common to all SurvivalExperiments in the CohortList.

**Examples**

```
data(sampleCohortList)
commonGenes <- findCommonGenes(sampleCohortList)
head(commonGenes)
```

---

`findCommonGenes,MultiAssayExperiment-method`

*Intersect Gene Names for All experiments in a MultiAssayExperiment*

---

**Description**

Intersect Gene Names for All experiments in a MultiAssayExperiment



**Usage**

```
## S4 method for signature 'MultiAssayExperiment'  
findCommonGenes(object)
```

**Arguments**

object            A MultiAssayExperiment where rownames represent genes.

**Value**

A character vector of genes common to all experiments in the MutliAssayExperiment.

---

<code>findCommonSamples</code>	<i>Find Common Samples in a List-like S4 Object where The Columns of Each Item Represent Samples</i>
--------------------------------	--

---

**Description**

Find Common Samples in a List-like S4 Object where The Columns of Each Item Represent Samples

**Usage**

```
findCommonSamples(object, ...)
```

**Arguments**

object            A S4 object, where the columns of each element represent samples.

...                Allow new parameters to be defined for this generic.

**Value**

A character vector of common sample names.

**Examples**

```
data(sampleCohortList)  
commonSamples <- findCommonSamples(sampleCohortList)  
head(commonSamples)
```

---

findCommonSamples, CohortList-method

*Find Common Samples in a CohortList Object where The Columns of Each Item Represent Samples*

---

### Description

Find Common Samples in a CohortList Object where The Columns of Each Item Represent Samples

### Usage

```
## S4 method for signature 'CohortList'
findCommonSamples(object)
```

### Arguments

object            A CohortList for which we want to find common samples between all SurvivalExperiment objects.

### Value

A character vector of common sample names.

### Examples

```
data(sampleCohortList)
commonSamples <- findCommonSamples(sampleCohortList)
head(commonSamples)
```

---

forestPlot

*Generate a forest plot from an S4 object*

---

### Description

Generate a forest plot from an S4 object

### Usage

```
forestPlot(object, ...)
```

### Arguments

object            An S4 object to create a forest plot of.  
 ...                Allow new parameters to this generic.

**Value**

None, draws a forest plot.

**Examples**

```
data(sampleValPCOSPmodel)

# Plot
dIndexForestPlot <- forestPlot(sampleValPCOSPmodel, stat='log_D_index')
```

---

forestPlot, ModelComparison-method

*Render a forest plot from the validationStats slot of a PCOSP model object.*

---

**Description**

Render a forest plot from the validationStats slot of a PCOSP model object.

**Usage**

```
## S4 method for signature 'ModelComparison'
forestPlot(
  object,
  stat,
  groupBy = "cohort",
  colourBy = "model",
  vline,
  ...,
  xlab,
  ylab,
  transform,
  colours,
  title
)
```

**Arguments**

object	A ModelComparison object to forest plot.
stat	A character vector specifying a statistic to plot.
groupBy	A character vector with one or more columns in validationStats to group by. These will be the facets in your forestplot.
colourBy	A character vector specifying the columns in validationStats to colour by.
vline	An integer value on the x-axis to place a dotted vertical line.
...	Force subsequent parameters to be named, not used.

xlab	A character vector specifying the desired x label. Automatically guesses based on the stat argument.
ylab	A character vector specifying the desired y label. Defaults to 'Cohort (P-value)'.
transform	The name of a numeric function to transform the statistic before making the forest plot.
colours	A character vector of colours to pass into <code>ggplot2::scale_fill_manual</code> , which modify the colourBy argument.
title	A character vector with a title to add to the plot.

**Value**

A `ggplot2` object.

**Examples**

```

data(sampleValPCOSPmodel)
data(sampleClinicalModel)
data(sampleCohortList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train the models
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Predict risk/risk-class
ClinicalPredCohortL <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)

# Validate the models
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=ClinicalPredCohortL)

# Compare the models
modelComp <- compareModels(sampleValPCOSPmodel, validatedClinicalModel)

# Make the forest plot
modelComp <- modelComp[modelComp$summary == TRUE, ]
modelCindexCompForestPlot <- forestPlot(modelComp, stat='concordance_index')

```

---

forestPlot, PCOSP\_or\_ClinicalModel-method

*Render a forest plot from the validationStats slot of a PCOSP model object.*

---

**Description**

Render a forest plot from the validationStats slot of a PCOSP model object.

**Usage**

```
## S4 method for signature 'PCOSP_or_ClinicalModel'
forestPlot(
  object,
  stat,
  groupBy = "mDataType",
  colourBy = "isSummary",
  vline,
  ...,
  xlab,
  ylab,
  transform,
  colours,
  title
)
```

**Arguments**

object	A PCOSP model which has been validated with validateModel and therefore has data in the validationStats slot.
stat	A character vector specifying a statistic to plot.
groupBy	A character vector with one or more columns in validationStats to group by. These will be the facets in your forestplot.
colourBy	A character vector specifying the columns in validationStats to colour by.
vline	An integer value on the x-axis to place a dotted vertical line.
...	Force subsequent parameters to be named, not used.
xlab	A character vector specifying the desired x label. Automatically guesses based on the stat argument.
ylab	A character vector specifying the desired y label. Defaults to 'Cohort (P-value)'.
transform	The name of a numeric function to transform the statistic before making the forest plot.
colours	A character vector of colours to pass into ggplot2::scale_fill_manual, which modify the colourBy argument.
title	A character vector with a title to add to the plot.

**Value**

A ggplot2 object.

**Examples**

```
data(sampleValPCOSPmodel)

# Plot
dIndexForestPlot <- forestPlot(sampleValPCOSPmodel, stat='log_D_index')
```

GeneFuModel

GeneFuModel *Constructor Method***Description**

GeneFuModel Constructor Method

**Usage**

```
GeneFuModel(
  trainCohorts = SurvivalExperiment(),
  minDaysSurvived = 365,
  ...,
  randomSeed
)
```

**Arguments**

trainCohorts	A CohortList or SurvivalExperiment containing training data for the genefu model. If you don't have training data, but have a trained model this will default to an empty SurvivalExperiment. You can then assign the model using the models setter method.
minDaysSurvived	An integer specifying the minimum days survived to be considered in the 'good' survival prognosis group.
...	Fall through paramater to SurvivalModel constructor.
randomSeed	An integer randomSeed that was used to train the model. Users should specify this when initializing a model to ensure reproducibility.

**Value**

A GeneFuModel object, with model parameters in the

**Examples**

```
set.seed(1987)
geneFuModel <- GeneFuModel(randomSeed=1987)
```

---

GeneFuModel-class	<i>A SurvivalModel Sub-class Designed to Hold A Survival Model Generated Using the geneFu R package.</i>
-------------------	--

---

**Description**

A SurvivalModel Sub-class Designed to Hold A Survival Model Generated Using the geneFu R package.

---

getModelSeed	<i>Generic for retrieving the randomSeed parameter from a SurvivalModel object.</i>
--------------	---

---

**Description**

This should be used to set the seed before model training to ensure reproducible results.

**Usage**

```
getModelSeed(object)
```

**Arguments**

object            An S4 object to get the seed from.

**Value**

An integer seed to be used when training the a SurvivalModel.

**Examples**

```
data(samplePCOSModel)  
getModelSeed(samplePCOSModel)
```

---

getModelSeed, SurvivalModel-method

*Method for retrieving the random seed used for training a specific survival model object.*

---

### Description

This should be used to set the seed before model training to ensure reproducible results.

### Usage

```
## S4 method for signature 'SurvivalModel'
getModelSeed(object)
```

### Arguments

object            A SurvivalModel object to get the seed from.

### Value

An integer seed to be used when training the a SurvivalModel.

### Examples

```
data(samplePCOSPmodel)
getModelSeed(samplePCOSPmodel)
```

---

getTopFeatures            *Get the Top Predictive Features from an S4 Object*

---

### Description

Get the Top Predictive Features from an S4 Object

### Usage

```
getTopFeatures(object, ...)
```

### Arguments

object            An S4 object to get top scoring features from.  
 ...                Allow additional parameters to be defined for this generic.

### Value

A character vector of top predictive features.



**Examples**

```
data(sampleTrainedPCOSPmodel)

# Get the top features
topFeatures <- getTopFeatures(sampleTrainedPCOSPmodel, numModels=2)
```

---

getTopFeatures,MultiAssayExperiment-method

*Get the Top Ranked Features in a MultiAssayExperiment object*

---

**Description**

Get the Top Ranked Features in a MultiAssayExperiment object

**Usage**

```
## S4 method for signature 'MultiAssayExperiment'
getTopFeatures(object, numFeats, ...)
```

**Arguments**

object	A SummarizedExperiment to extract top features from
numFeats	An integer number of top ranked features to extract.
...	Fall through arguments to rankFeatures.

**Value**

A character vector of top ranked features, with the features in order of rank ascending.

**See Also**

[rankFeatures,MultiAssayExperiment-method](#)

---

getTopFeatures,PCOSP-method

*Get the top features for classification using a PCOSP model object.*

---

**Description**

Get the top features for classification using a PCOSP model object.

**Usage**

```
## S4 method for signature 'PCOSP'
getTopFeatures(object, numModels)
```

**Arguments**

`object` A PCOSP model object which has been trained with `trainModel`.

`numModels` An integer specifying the number of top models to use features from. Defaults to top 10% of KTSPs models.

**Value**

A character vector of gene names representing the unique genes from the top `numModels` KTSPs models in the model object.

**Examples**

```
data(sampleTrainedPCOSPmodel)

# Get the top features
topFeatures <- getTopFeatures(sampleTrainedPCOSPmodel, numModels=5)
```

---

`getTopFeatures, SummarizedExperiment-method`

*Get the Top Ranked Features in a SummarizedExperiment object*

---

**Description**

Get the Top Ranked Features in a `SummarizedExperiment` object

**Usage**

```
## S4 method for signature 'SummarizedExperiment'
getTopFeatures(object, numFeats, ..., rankCol = "feature_rank")
```

**Arguments**

`object` A `SummarizedExperiment` to extract top features from

`numFeats` An integer number of top ranked features to extract.

`...` Fall through arguments to `rankFeatures`, which is used to calculate the ranks if `rankCol` is not present the object `rowData`.

`rankCol` The name of the column containing the integer feature ranks. Defaults to `feature_rank`, as calculated with `rankFeatures`, but users can alternatively specify their own custom rank column if desired.

**Value**

A character vector of top ranked features, with the features in order of rank ascending.

**Examples**

```
data(sampleICGmicro)
getTopFeatures(sampleICGmicro, numFeats=20)
```

---

haiderSigScores	<i>Classifier survival scores for Haider</i>
-----------------	--

---

**Description**

The classifier risk scores from Haider *et al.* (2014)

**Examples**

```
# Loads chen, birnbaum and haiderSigScores objects
data(existingClassifierData)
```

---

hasColDataColumns	<i>Check for Column Names in the colData Slot of a SummarizedExperiment</i>
-------------------	---

---

**Description**

Check for Column Names in the colData Slot of a SummarizedExperiment

**Usage**

```
hasColDataColumns(SE, values)
```

**Arguments**

SE	A SummarizedExperiment object to check for the existence of colData columns.
values	A character vector with one or more column name to check for in the column data.

**Value**

logical True if all of values are column names in the SummarizedExperiment object, FALSE otherwise.

**Examples**

```
SE <- SummarizedExperiment(matrix(rnorm(100), 10, 10),
  colData=data.frame(test=rep(1, 10)))
hasColDataColumns(SE, 'test')
```

---

```
merge, SurvivalExperiment, SurvivalExperiment-method
      Merge two SurvivalExperiments, subsetting to shared rows and
      columns
```

---

**Description**

Merge two SurvivalExperiments, subsetting to shared rows and columns

**Usage**

```
## S4 method for signature 'SurvivalExperiment, SurvivalExperiment'
merge(x, y, cohortNames)
```

**Arguments**

x                    A SurvivalExperiment.  
y                    A SurvivalExperiment.  
cohortNames        An optional character vector specifying the a name for each SurvivalExperiment.

**Value**

A SurvivalExperiment object with merge data from x and y, and the assay from each in the assays slot.

**Examples**

```
data(sampleICGCMicro)
survExp2 <- sampleICGCMicro
mergedSurvExp <- merge(survExp2, sampleICGCMicro,
  cohortNames=c('copyICGCMicro', 'ICGCMicro'))
mergedSurvExp
```

---

```
ModelComparison        ModelComparison Constructor
```

---

**Description**

ModelComparison Constructor

**Usage**

```
ModelComparison(model1, model2, ...)
```

**Arguments**

model1	An object with a validationStats method which returns a data.table. Probably this object should inherit from SurvivalModel.
model2	An object with a validationStats method which returns a data.table. Probably this object should inherit from SurvivalModel.
...	Not used.

**Value**

A ModelComparison object, which is a wrapper for DataFrame which is used for method dispatch.

**Examples**

```
data(sampleValPCOSModel)
data(sampleClinicalModel)
data(sampleCohortList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train the models
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Predict risk/risk-class
clinicalPredCohortL <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)

# Validate the models
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=clinicalPredCohortL)

# Compare the models
modelComp <- ModelComparison(sampleValPCOSModel, validatedClinicalModel)
head(modelComp)
```

---

ModelComparison-class *ModelComparison Class Definition*

---

**Description**

ModelComparison Class Definition

---

 modelParams

*Generic for Accessing the Model Parameters of an S4 Object*


---

**Description**

Generic for Accessing the Model Parameters of an S4 Object

**Usage**

```
modelParams(object, ...)
```

**Arguments**

object	A S4 Object.
...	Allow additional arguments to be defined for this generic.

**Value**

A List- or list-like object containing all the parameters needed to reproduce a specific model training run, including environmental settings like the random seed and RNGkind.

**Examples**

```
data(CSPC_MAE)
set.seed(1987)
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)
modelParams(metaclustModel)
```

---

 modelParams,S4Model-method

*Accessor for the Model Parameters of an S4Model Object*


---

**Description**

Accessor for the Model Parameters of an S4Model Object

**Usage**

```
## S4 method for signature 'S4Model'
modelParams(object)
```

**Arguments**

object	A S4Model Object
--------	------------------

**Value**

A List- or list-like object containing all the parameters needed to reproduce a specific model training run, including environmental settings like the random seed and RNGkind.

---

models	<i>Accessor for the models slot of an S4 object</i>
--------	---

---

**Description**

Accessor for the models slot of an S4 object

**Usage**

```
models(object, ...)
```

**Arguments**

object	An S4 object to retrieve models from.
...	Allow new parameters to be defined for this generic.

**Value**

An S4 object representing a model.

**Examples**

```
data(CSPC_MAE)
set.seed(1987)
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)
models(metaclustModel)
```

---

models, S4Model-method	<i>Getter for the models slot of an S4Model Object</i>
------------------------	--

---

**Description**

Getter for the models slot of an S4Model Object

**Usage**

```
## S4 method for signature 'S4Model'
models(object)
```

**Arguments**

object            An S4Model object to retrieve models from.

**Value**

An List- or list-like object representing a model.

---

*models*,*SurvivalModel*-method

*Getter for the models slot of a SurvivalModel object*

---

**Description**

Getter for the models slot of a SurvivalModel object

**Usage**

```
## S4 method for signature 'SurvivalModel'
models(object)
```

**Arguments**

object            A SurvivalModel model object to retrieve the models slot from.

**Value**

A SimpleList of top scoring KTSPmodels

**Examples**

```
data(samplePCOSPmodel)
models(samplePCOSPmodel)
```

---

*models<-*,*SurvivalModel*,*SimpleList*-method

*Setter for the models slot of a SurvivalModel object*

---

**Description**

Setter for the models slot of a SurvivalModel object

**Usage**

```
## S4 replacement method for signature 'SurvivalModel,SimpleList'
models(object) <- value
```



**Arguments**

object            A SurvivalModel object to update  
 value            A SimpleList of trained KTSP models

**Value**

None, updates the object.

**Examples**

```
data(samplePCOSPmodel)
models(samplePCOSPmodel) <- SimpleList(model1=NA)
```

---

NCSModel-class            *S4Model Class for Metaclustering Using Network Community Search*

---

**Description**

S4Model Class for Metaclustering Using Network Community Search

---

NetworkCommunitySearchModel  
*Constructor for a NetworkCommunitySearchModel (NCSModel)*

---

**Description**

Use `igraph::fastgreedy.community` cross-cohort metaclusters for a `ConsensusMetaclusteringModel` which has been validated with `validateModel`.

**Usage**

```
NetworkCommunitySearchModel(model)
```

**Arguments**

model            A validated `ConsensusMetaclusteringModel` object.

**Value**

An `NCSModel` object containing the relevant data from the `ConsensusMetaclusteringModel`.

---

normalize,data.frame\_or\_matrix-method  
*Normalize a data.frame Object*

---

### Description

Normalize a data.frame Object

### Usage

```
## S4 method for signature 'data.frame_or_matrix'
normalize(object, MARGIN = 2, FUN = "proprocessCaret", ...)
```

### Arguments

object	A data.frame object to normalize.
MARGIN	An integer indicating if rows (1) or columns (2) should be normalized. Defaults to 2 for columns.
FUN	A function to normalize your data with. Should accept a rectangular object such as a matrix, data.frame, or data.table and return an object of the same class with the data normalized using FUN.
...	Fall through parameters to FUN. For the default FUN, these are passed to caret::preProcess to allow configuration of the normalization method. Omitting any arguments with the default FUN will scale and center the data.

### Value

The data.frame normalized.

---

normalize,DFrame-method  
*Normalize a S4 DFrame Object*

---

### Description

Normalize a S4 DFrame Object

### Usage

```
## S4 method for signature 'DFrame'
normalize(object, MARGIN = 2, FUN = "preprocessCaret", ...)
```

**Arguments**

object	A DFrame or DataFrame object to normalize.
MARGIN	An integer indicating if rows (1) or columns (2) should be normalized. Defaults to 2 for columns.
FUN	A function to normalize your data with. Should accept a rectangular object such as a matrix, data.frame, or data.table and return an object of the same class with the data normalized using FUN.
...	Fall through parameters to FUN. For the default FUN, these are passed to caret::preProcess to allow configuration of the normalization method. Omitting any arguments with the default FUN will scale and center the data.

**Value**

A normalized DFrame object.

---

normalize,MultiAssayExperiment-method

*Normalize the assays of a MutliAssayExperiment Object*

---

**Description**

For this method to work, there must be a normalize method defined for all classes of experiments in the MultiAssayExperiment

**Usage**

```
## S4 method for signature 'MultiAssayExperiment'
normalize(
  object,
  MARGIN = 2,
  FUN = "preprocessCaret",
  ...,
  whichAssays = seq_along(assays(object))
)
```

**Arguments**

object	A SummarizedExperiment object with assays to normalize.
MARGIN	An integer indicating if rows (1) or columns (2) should be normalized. Defaults to 2 for columns.
FUN	A function to normalize your data with. Should accept a rectangular object such as a matrix, data.frame, or data.table and return an object of the same class with the data normalized using FUN.
...	Fall through parameters to FUN. For the default FUN, these are passed to caret::preProcess to allow configuration of the normalization method. Omitting any arguments with the default FUN will scale and center the data.

`whichAssays` A numeric or character vector specifying the indices of the assays to normalize. Defaults to all assays.

### Details

When using the default FUN, it is also possible to impute missing values. See `?caret::preProcess` for information on available methods.

### Value

The `MultiAssayExperiment` with one or more of the assays normalized and information about the normalization method in the `normalization` item of the object metadata.

### See Also

[preprocessCaret](#), `caret::preProcess`

---

normalize, SummarizedExperiment-method

*Normalize the assays in a SummarizedExperiment Object*

---

### Description

Normalize the assays in a `SummarizedExperiment` Object

### Usage

```
## S4 method for signature 'SummarizedExperiment'
normalize(
  object,
  MARGIN = 2,
  FUN = "preprocessCaret",
  ...,
  whichAssays = seq_along(assays(object))
)
```

### Arguments

`object` A `SummarizedExperiment` object with assays to normalize.

`MARGIN` An integer indicating if rows (1) or columns (2) should be normalized. Defaults to columns. Defaults to 2.

`FUN` A function to normalize your data with. The function should accept a matrix, normalize the columns then return a matrix. The data will be transposed before applying FUN if `MARGIN=1`.

`...` Fall through parameters to FUN. When using the default FUN, the data is scaled and centered when no additional arguments are specified.

`whichAssays` A numeric or character vector specifying the indices or names of the assays to normalize. Defaults to all assays.

**Details**

When using the default FUN, it is also possible to impute missing values. See `?caret::preProcess` for information on available methods.

**Value**

The SummarizedExperiment with one or more of the matrices in assays normalized and the normalization details in the normalization item of the object metadata.

**See Also**

[preprocessCaret](#), `caret::preProcess`

---

normalsMAE	<i>A MultiAssayExperiment containing cohorts of normal patients, for package examples.</i>
------------	--

---

**Description**

A MultiAssayExperiment containing cohorts of normal patients, for package examples.

**Examples**

```
data(normalsMAE)
```

---

optimalKMinimizeAmbiguity	<i>Predict optimal K values by minimizing the difference between the ECDF of clustering consensus at two points in a subinterval.</i>
---------------------------	---

---

**Description**

Predict optimal K values by minimizing the difference between the ECDF of clustering consensus at two points in a subinterval.

**Usage**

```
optimalKMinimizeAmbiguity(assayClusters, subinterval = c(0.1, 0.9))
```

**Arguments**

- `assayClusters` A `SimpleList` of clustering results from a `ConsensusMetaclusteringModel`, as returned by `models(object)` where `object` is a trained `ConsensusMetaclusteringModel` object.
- `subinterval` A numeric vector of two float values, the first being the lower and second being the upper limit of the subinterval to compare cluster ambiguity over. Default is `c(0.1, 0.9)`, i.e. comparing the 10th and 90th percentile of cluster consensus to calculate the ambiguity of a given clustering solution. This is the value used to select the optimal K value from the potential solutions for each assay in the training data.

**Value**

A numeric vector the same length as `assayClusters`, with an optimal K prediction for each assay in the `rawdata` slot of the trained `ConsensusMetaclusteringModel` object which `assayClusters` came from.

---

 PCOSP

---

*Pancreatic Cancer Overall Survival Predictor (PCOSP) Constructor*


---

**Description**

Pancreatic Cancer Overall Survival Predictor (PCOSP) Constructor

**Usage**

```
PCOSP(trainCohorts, minDaysSurvived = 365, ..., randomSeed)
```

**Arguments**

- `trainCohorts` A `CohortList` or `SurvivalExperiment` containing the training data for the PCOSP model.
- `minDaysSurvived` An integer indicating the minimum number of day required to be in the 'good' survival group. Any patients below this cut-off will be considered low survival. Default is 365 days.
- `...` Force subsequent parameters to be named. This parameter is not used.
- `randomSeed` An integer `randomSeed` that was used to train the model. Users should specify this when initializing a model to ensure reproducibility.

**Details**

This function assumes there is only 1 assay per `SurvivalExperiment`.

**Value**

A PCOSP object with training data in the assays slot, concatenating together the molecular data types and labelling the genes with the data type to ensure the results are easily interpretable.

**Examples**

```
data(sampleICGcmicro)
set.seed(1987)
PCOSPmodel <- PCOSP(sampleICGcmicro, minDaysSurvived=365, randomSeed=1987)
```

---

PCOSP-class

*Pancreatic Cancer Overall Survival Predictor (PCOSP) Class*


---

**Description**

Pancreatic Cancer Overall Survival Predictor (PCOSP) Class

---

PCOSP\_or\_ClinicalModel-class

*Class Union for PCOSP and ClinicalModel Types*


---

**Description**

Class union used for method dispatch without inheritance

---

PCOSP\_or\_RLS\_or\_RGA-class

*Class Union for PCOSP, RLSModel and RGAModel Types*


---

**Description**

Class union used for method dispatch without inheritance

---

plotNetworkGraph      *A Generic for Plotting a Network Graph From an S4 Object*

---

**Description**

A Generic for Plotting a Network Graph From an S4 Object

**Usage**

```
plotNetworkGraph(object, ...)
```

**Arguments**

object            An S4 object with a valid plotNetworkGraph method set.  
 ...              Allow additional arguments to be defined for this generic.

**Value**

A network plot, either as an object or via side effects.

---

plotNetworkGraph,NCSModel-method  
*Plot a Network Graph for a Classified NCSModel Object*

---

**Description**

Visualize metaclusters predicted using network community search on the consensus clustering results for a MultiAssayExperiment of patient cohorts.

**Usage**

```
## S4 method for signature 'NCSModel'  

plotNetworkGraph(object, ..., palette = "Set1", clusterLabels)
```

**Arguments**

object            A classified NCSModel object, as returned by the predictClasses method.  
 ...              Not used, force subsequent arguments to be named.  
 palette          character A valid palette for use in RColourBrewer::brewer.pal  
 clusterLabels   A character vector of names for the metaclusters. Defaults to the cluster number.

**Value**

A ggplot object containing the network graph, showing the relative edge distances between each cluster in each cohort along with the predicted metacluster label.



---

plotROC	<i>Plot ROC curves for an S4 object</i>
---------	---

---

**Description**

Plot ROC curves for an S4 object

**Usage**

```
plotROC(object, ...)
```

**Arguments**

object	An S4 object with a defined plotROC method.
...	Allow new parameters to be added to this generic.

**Value**

A ggplot object containing the ROC curves.

---

plotROC,PCOSP-method	<i>Plot ROC curves for a PCOSP model object.</i>
----------------------	--

---

**Description**

Plot ROC curves for a PCOSP model object.

**Usage**

```
## S4 method for signature 'PCOSP'
plotROC(object, alpha = 0.05, ..., xlabel, ylabel, title)
```

**Arguments**

object	A PCOSP model which has been validated with with the validateModel method.
alpha	A float specifying the significance level for the plot. Non- significant cohorts will have dotted lines.
...	Catch unnamed parameters. Not used.
xlabel	A character vector specifying the x label.
ylabel	A character vector specifying the y label.
title	A character vector speciyfing the plot tile.

**Value**

A ggplot object containing the ROC curves.

**Examples**

```
data(sampleValPCOSModel)

# Plot ROC curves
AUROCplot <- plotROC(sampleValPCOSModel)
```

---

plotSurvivalCurves      *Generic for Plotting Survival Curves from an S4 Object*

---

**Description**

Generic for Plotting Survival Curves from an S4 Object

**Usage**

```
plotSurvivalCurves(object, ...)
```

**Arguments**

object            An S4 object to plot survival curves for.  
...                Allow new parameters to be defined on this generic.

**Value**

A plot, either via side-effects or as the return value.

---

plotSurvivalCurves,CoxModel-method  
                          *Plot Survival Curves from a Fit CoxModel Object*

---

**Description**

Plot Survival Curves from a Fit CoxModel Object

**Usage**

```
## S4 method for signature 'CoxModel'  
plotSurvivalCurves(object, byCohort = TRUE, ..., facet.by = "cohort")
```

**Arguments**

object	A CoxModel object with survival curves fit via the trainModel method.
byCohort	TRUE to return a single plot object faceted by facet.by, FALSE to get a list of individual survival curves per cohort.
...	Fall through parameters to survminer::ggsurvplot function.
facet.by	What column of the object modelDT to use for faceting the survival plot. Defaults to 'cohorts'. Only used if byCohort is TRUE.

**Value**

A ggplot or list of ggplot objects containing the survival curves for each cohort in the trainData slot of the CoxModel.

---

predictClasses	<i>Predict Classes for New Data Based on a Train Classifier Model</i>
----------------	---

---

**Description**

Predict Classes for New Data Based on a Train Classifier Model

**Usage**

```
predictClasses(object, model, ...)
```

**Arguments**

object	An S4 object containing data to predict classes from.
model	An S4 object containing one or more trained classification models.
...	Allow further parameters to be defined on this generic.

**Value**

The S4 object with class predictions added to the metadata.

**Examples**

```
data(sampleTrainedPCOSPmodel)
data(samplePCSIurvExp)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Make predictions
PCOSPpredSurvExp <- predictClasses(samplePCSIurvExp,
  model=sampleTrainedPCOSPmodel)
head(colData(PCOSPpredSurvExp))
```

---

predictClasses,CohortList,ClinicalModel-method

*Use a Clinical GLM to Predict Classes for a CohortList of SurvivalExperiment Objects.*

---

## Description

Use a Clinical GLM to Predict Classes for a CohortList of SurvivalExperiment Objects.

## Usage

```
## S4 method for signature 'CohortList,ClinicalModel'
predictClasses(object, model, ..., na.action = "na.exclude", type = "response")
```

## Arguments

object	A CohortList with SurvivalExperiments to predict classes for. The colData slot in ALL SurvivalExperiments must have column names which match the formula in the model object.
model	A trained ClinicalModel object, as return by trainModel.
...	Fall through parameters to <code>stats::predict</code> .
na.action	The na.action paramter passed to <code>stats::predict.glm</code> .
type	The type parameter passed to <code>stats::predict.glm</code>

## Value

A CohortList with the model predictions in the colData slot as clinical\_prob\_good for each SurvivalExperiment, and the model in the metadata as predictionModel.

## Examples

```
data(sampleClinicalModel)
data(sampleCohortList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train Model
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Make predictions
ClinicalPredCohortList <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)
head(colData(ClinicalPredCohortList[[1]]))
```

---

predictClasses, CohortList, GeneFuModel-method

*Use a Gene Signature Based Prediction Model from the geneFu Package to Predict Signature Scores for Each Sample*

---

### Description

Use a Gene Signature Based Prediction Model from the geneFu Package to Predict Signature Scores for Each Sample

### Usage

```
## S4 method for signature 'CohortList, GeneFuModel'
predictClasses(object, model, ..., annot = NA)
```

### Arguments

object	A CohortList with SurvivalExperiments to predict classes for.
model	A trained GeneFuModel object.
...	Fall through parameters to <code>geneFu::sig.score</code> .
annot	The annot parameter passed to <code>geneFu::sig.score</code> . Defaults to NA, which assumes your assay rowname match the gene labels in the model.

### Value

A CohortList with the model predictions in the colData slot as `geneFu_score` for each SurvivalExperiment, and the model in the metadata as `predictionModel`.

---

predictClasses, CohortList, PCOSP\_or\_RLS\_or\_RGA-method

*Predict Survival Prognosis Classes and Risk Scores for A CohortList Using a PCOSP, RLSModel or RGAModel object.*

---

### Description

Predict Survival Prognosis Classes and Risk Scores for A CohortList Using a PCOSP, RLSModel or RGAModel object.

### Usage

```
## S4 method for signature 'CohortList, PCOSP_or_RLS_or_RGA'
predictClasses(object, model, ...)
```

**Arguments**

object            A CohortList with SurvivalExperiments to predict classes for.  
 model            A trained PCOSP model to use for predicting classes.  
 ...              Fall through arguments to BiocParallel::bplapply for configuring parallelization settings.

**Value**

A CohortList with the model predictions attached to each SurvivalExperiment in the metadata slot and the prob\_good\_survival column added to the colData slot.

**Examples**

```
data(sampleTrainedPCOSPmodel)
data(sampleCohortList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Make predictions
PCOSPpredCohortList <- predictClasses(sampleCohortList[seq_len(2)],
  model=sampleTrainedPCOSPmodel)
head(colData(PCOSPpredCohortList[[1]]))
```

---

*predictClasses, ConsensusMetaclusteringModel, ANY-method*

*Compute the Optimal Clustering Solution for a Trained Consensus-MetaclusteringModel*

---

**Description**

Compute the optimal clustering solution out of possibilities generated with trainModel. Assigns the cluster labels to the MultiAssayExperiment object.

**Usage**

```
## S4 method for signature 'ConsensusMetaclusteringModel,ANY'
predictClasses(object, ..., optimal_k_function = optimalKMinimizeAmbiguity)
```

**Arguments**

object            A MutliAssayExperiment object  
 ...              Fall through arguments to optimal\_k\_function. For the default optimal\_k\_function, you can specify subinterval argument which defines the interval of the ECDF to minimize ambiguity over. Defaults to c(0.1, 0.9) if not specified. See ?optimalKMinimizeAmbiguity for more details on the subinterval parameter.

optimal\_k\_function

A function which accepts as its input models(object) of a trained ConsensusMetaclusteringModel object, and returns a vector of optimal K values, one for each assay in rawdata(object). The default method is optimalKMinimizeAmbiguity, see ?optimalKMinimizeAmbiguity for more details. Please note this argument must be named or it will not work.

### Value

A object ConsensusMetaclusteringModel, with class predictions assigned to the colData of trianData

predictClasses,NCSModel,ANY-method

*Predict Metacluster Labels for a NetworkCommunitySearchModel*

### Description

Predict Metacluster Labels for a NetworkCommunitySearchModel

### Usage

```
## S4 method for signature 'NCSModel,ANY'
predictClasses(object)
```

### Arguments

object            A NCSModel which has been trained.

### Value

The object model with

predictClasses,SurvivalExperiment,ClinicalModel-method

*Predict Survival Prognosis Classes and Risk Scores for A SurvivalModel Using a ClinicalModel Object.*

### Description

Predict Survival Prognosis Classes and Risk Scores for A SurvivalModel Using a ClinicalModel Object.

### Usage

```
## S4 method for signature 'SurvivalExperiment,ClinicalModel'
predictClasses(object, model, ..., na.action = "na.exclude", type = "response")
```

**Arguments**

object	A SurvivalExperiment object with the correct columns in colData to match the formula for the ClinicalModel object.
model	A trained ClinicalModel object, as return by trainModel.
...	Fall through parameters to <code>stats::predict</code> .
na.action	The na.action paramter passed to <code>stats::predict.glm</code> .
type	The type parameter passed to <code>stats::predict.glm</code>

**Value**

A SurvivalExperiment with the model predictions in the colData slot as clinical\_prob\_good.

**Examples**

```
data(sampleClinicalModel)
data(samplePCSIurvExp)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train Model
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Make predictions
ClinicalPredSurvExp <- predictClasses(samplePCSIurvExp,
  model=trainedClinicalModel)
head(colData(ClinicalPredSurvExp))
```

---

predictClasses, SurvivalExperiment, GeneFuModel-method

*Use a Gene Signature Based Prediciton Model from the genefu Package to Predict Signature Scores for Each Sample.*

---

**Description**

Use a Gene Signature Based Prediciton Model from the genefu Package to Predict Signature Scores for Each Sample.

**Usage**

```
## S4 method for signature 'SurvivalExperiment, GeneFuModel'
predictClasses(object, model, ..., annot = NA)
```



**Arguments**

object	A SurvivalExperiment to predict classes for.
model	A GeneFuModel object to predict classes with.
...	Fall through parameters to <code>genefu::sig.score</code> .
annot	A named parameter with annotations mapping from the gene identifiers in the <code>genefu</code> model.

**Details**

A signature score should be interpreted as unit-less continuous risk predictor.

**Value**

The SurvivalExperiment passed to the object argument with the `genefu_score` column added to the objects `colData` slot.

---

predictClasses, SurvivalExperiment, PCOSP\_or\_RLS\_or\_RGA-method

*Predict Survival Prognosis Classes and Risk Scores for A CohortList Using a PCOSP, RLSModel or RGAModel object.*

---

**Description**

Predict Survival Prognosis Classes and Risk Scores for A CohortList Using a PCOSP, RLSModel or RGAModel object.

**Usage**

```
## S4 method for signature 'SurvivalExperiment,PCOSP_or_RLS_or_RGA'
predictClasses(object, model, ...)
```

**Arguments**

object	A SurvivalExperiment object to predict classes for.
model	A trained PCOSP model to use for predicting classes.
...	Fall through arguments to <code>BiocParallel::bplapply</code> for configuring parallelization settings.

**Value**

A SurvivalExperiment with the predictions in its metadata and a column in `colData`, `prob_good_survival`, which contains the proportion of models which predicted good prognosis for each sample.

**See Also**

[BiocParallel::bplapply](#), [switchBox::SWAP.KTSP.Classify](#)

## Examples

```
data(sampleTrainedPCOSPmodel)
data(samplePCSIurvExp)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Make predictions
PCOSPpredSurvExp <- predictClasses(samplePCSIurvExp,
  model=sampleTrainedPCOSPmodel)
head(colData(PCOSPpredSurvExp))
```

---

preprocessCaret	<i>Preprocess Data Using the caret::preProcess method, then return the normalized data using predict.</i>
-----------------	---

---

## Description

Preprocess Data Using the caret::preProcess method, then return the normalized data using predict.

## Usage

```
preprocessCaret(x, ...)
```

## Arguments

x	The data to be normalized with caret::preProcess and caret::predict.
...	Fall through parameters to caret::preProcess. This can be used to apply a range of different preprocessing methods from that package.

## Value

x preprocessed according to the arguments in ...

## See Also

[caret::preProcess](#), [stats::predict](#)

---

`RandomGeneAssignmentModel`*RandomGeneAssignmentModel Constructor*

---

**Description**

RandomGeneAssignmentModel Constructor

**Usage**

```
RandomGeneAssignmentModel(trainCohorts, minDaysSurvived = 365, ..., randomSeed)
```

**Arguments**

<code>trainCohorts</code>	A 'SurvivalExperiment' containing training data for the SurvivalModel object.
<code>minDaysSurvived</code>	An integer minimum number of days survived to be classified as a 'good' prognosis.
<code>...</code>	Force subsequent parameters to be named. Not used.
<code>randomSeed</code>	An integer randomSeed that was used to train the model. Users should specify this when initializing a model to ensure reproducibility.

**Value**

A SurvivalModel object.

**Examples**

```
data(sampleICGCmicro)
set.seed(1987)
RGAModel <- RGAModel(sampleICGCmicro, minDaysSurvived=365, randomSeed=1987)
```

---

`RandomLabelShufflingModel`*RandomLabelShufflingModel Constructor*

---

**Description**

RandomLabelShufflingModel Constructor

**Usage**

```
RandomLabelShufflingModel(trainCohorts, minDaysSurvived = 365, ..., randomSeed)
```

**Arguments**

trainCohorts	A 'SurvivalExperiment' containing training data for the SurvivalModel object.
minDaysSurvived	An integer minimum number of days survived to be classified as a 'good' prognosis.
...	Force subsequent parameters to be named. Not used.
randomSeed	An integer randomSeed that was used to train the model. Users should specify this when initializing a model to ensure reproducibility.

**Value**

A SurvivalModel object.

**Examples**

```
data(sampleICGCmicro)
set.seed(1987)
RLSmodel <- RLSModel(sampleICGCmicro, minDaysSurvived=365, randomSeed=1987)
```

---

rankFeatures

*Rank the Features in a S4 Object*

---

**Description**

Rank the Features in a S4 Object

**Usage**

```
rankFeatures(object, ...)
```

**Arguments**

object	An S4 object where rows represent features.
...	Allow new parameters to be defined for this generic.

**Value**

The object with the features per value and ranking of the features in the rowData slot of the object.

**Examples**

```
data(sampleICGCmicro)
rankFeatures(sampleICGCmicro)
```

---

rankFeatures,MultiAssayExperiment-method

*Rank the Features in a MultiAssayExperiment Object*


---

### Description

Rank the Features in a MultiAssayExperiment Object

### Usage

```
## S4 method for signature 'MultiAssayExperiment'
rankFeatures(
  object,
  FUN = "mad",
  RANK_FUN = "dense_rank",
  ...,
  descending = TRUE,
  weights
)
```

### Arguments

object	A MultiAssayExperiment to rank the features in.
FUN	A vectorized feature scoring function, such as var or mad. Defaults to mad from the BiocGenerics package.
RANK_FUN	A ranking function, such as rank or dense_rank. Defaults to dense_rank from dplyr.
...	Fall through arguments to FUN, such as na.rm=TRUE.
descending	Should your rank function be called with - before the values from FUN. Defaults to TRUE, which should be used if high values returned from FUN are good.
weights	A named numeric weighting vector with a weight for each experiment in the MultiAssayExperiment object. Names must match the names(experiments(object)). Passed to matrixStats::weightedMedian when aggregating feature scores per assay. Defaults to the sample size of an assay relative to the largest sample size when this paramter is missing.

### Value

The MultiAssayExperiment with the item featureRanks in the object metadata, which stores a DataFrame containing ranks across all assays for each unique feature and the additional columns feature\_score and feature\_rank, as calculated with FUN and RANK\_FUN, respectively. Information about which functions were used for each column can be found in the object mcols in the calculated\_with column.

### See Also

[BiocGenerics::mad](#), [dplyr::dense\\_rank](#), [matrixStats::weightedMedian](#)

---

 rankFeatures, SummarizedExperiment-method

*Rank the Features in a SummarizedExperiment Object*


---

## Description

Rank the Features in a SummarizedExperiment Object

## Usage

```
## S4 method for signature 'SummarizedExperiment'
rankFeatures(
  object,
  FUN = "rowMads",
  RANK_FUN = "dense_rank",
  ...,
  descending = TRUE,
  assay = 1
)
```

## Arguments

object	A SummarizedExperiment to rank the features in.
FUN	A row-wise summary function, such as rowVars, rowMads, etc. defaults to MatrixGenerics::rowMads.
RANK_FUN	A ranking function, such as rank or dense_rank. Defaults to dplyr::dense_rank.
...	Fall through arguments to FUN, such as na.rm=TRUE.
descending	Should your rank function be called with - before the values from FUN. Defaults to TRUE.
assay	integer assay to use for the ranking, as passed to the SummarizedExperiment::assay function. Defaults to the first assay.

## Value

The SummarizedExperiment with the column feature\_score and feature\_rank in the rowData slot. Information about which functions were used for each column can be found in the object mcols in the calculated\_with column.

## Examples

```
data(sampleICGCMicro)
rankFeatures(sampleICGCMicro, FUN='rowMads', RANK_FUN='dense_rank')
```

---

```
removeColDataFactorColumns
```

*Remove any factor columns from the colData of an S4 object*

---

**Description**

Remove any factor columns from the colData of an S4 object

**Usage**

```
removeColDataFactorColumns(x)
```

**Arguments**

x                    An S4 object with a colData method defined for it.

**Value**

x with colData factor columns converted to either integer or character, as appropriate.

**Examples**

```
data(sampleICGCMicro)
removeColDataFactorColumns(sampleICGCMicro)
```

---

```
removeFactorColumns    Convert factor columns in a rectangular object
```

---

**Description**

Convert factor columns in a rectangular object

**Usage**

```
removeFactorColumns(x)
```

**Arguments**

x                    A list-like rectangular object such as a data.frame, data.table, or DataFrame.

**Value**

x with factor columns converted to either integer or character, as appropriate.

**Examples**

```
x <- data.frame(a=factor(LETTERS[1:5]), b=factor(runif(5, 0, 1)))
removeFactorColumns(x)
```

---

renameColDataColumns    *Rename the columns in the colData slot, or do nothing if they don't match*

---

**Description**

Rename the columns in the colData slot, or do nothing if they don't match

**Usage**

```
renameColDataColumns(x, values)
```

**Arguments**

x	An S4 object with a colData method.
values	A character vector where names are the existing column names and values are the new column names.

**Value**

x with updated column names, if they match any existing columns.

**Examples**

```
data(sampleICGcmicro)
renameColDataColumns(sampleICGcmicro, c(event_occurred='days_survived'))
```

---

renameColumns            *Rename columns or do nothing if the names don't match*

---

**Description**

Rename columns or do nothing if the names don't match

**Usage**

```
renameColumns(x, values)
```



**Arguments**

- `x` An object for which `colnames` is defined, probably a `data.frame` or other similar object.
- `values` A character vector where names are the old column names and values are the new column names. Uses `gsub` internally to do the renaming.

**Value**

`x` with the updated column names if they are present. Does not fail if the column names are missing.

**Examples**

```
x <- data.frame(a=factor(LETTERS[1:5]), b=factor(runif(5, 0, 1)))
renameColumns(x, c(a='c'))
```

---

RGAModel-class	<i>RGAModel Class Definition</i>
----------------	----------------------------------

---

**Description**

RGAModel Class Definition

---

RLSModel-class	<i>RLSModel Class Definition</i>
----------------	----------------------------------

---

**Description**

RLSModel Class Definition

---

runGSEA	<i>Run Gene Set Enrichment Analysis</i>
---------	---

---

**Description**

Run Gene Set Enrichment Analysis

**Usage**

```
runGSEA(object, geneSet, ...)
```

**Arguments**

object	An S4 object to conduct Gene Set Enrichment Analysis (GSEA) with.
geneSet	An object representing a gene set, such as a <code>data.frame</code> .
...	Allow additional parameters to be defined for this generic.

**Value**

A `data.frame` containing the significantly enriched gene sets.

---

`runGSEA,PCOSP,data.frame-method`

*Run Gene Set Enrichment Analysis On A PCOSP Model Object.*

---

**Description**

Run Gene Set Enrichment Analysis On A PCOSP Model Object.

**Usage**

```
## S4 method for signature 'PCOSP,data.frame'
runGSEA(object, geneSet, numModels, ..., adjMethod = "fdr", allResults = FALSE)
```

**Arguments**

object	A PCOSP model which has been trained with <code>trainModel</code> .
geneSet	A <code>data.frame</code> with two columns, the first being the name of the gene and the second the gene set. The gene names must match the rownames of <code>object</code> . Additional columns will be dropped.
numModels	The number of models to use when selecting the top features from the PCOSP model in <code>object</code> . If missing will default to the top 10% of models.
...	Force subsequent parameters to be named. Not used.
adjMethod	An optional parameter specifying the multiple testing correction to use in <code>piano::runGSAhyper</code> . This parameter must be named.
allResults	Return the full results from <code>piano::runGSAhyper</code> instead of a <code>data.frame</code> of significant results? Default is <code>FALSE</code> . This parameter must be named.

**Value**

A `data.table` containing the significantly enriched gene sets.

---

S4Model-class	<i>An S4 Virtual Class For the Concept of a Statistical or ML Model</i>
---------------	---

---

### Description

An S4 Virtual Class For the Concept of a Statistical or ML Model

### Slots

`trainData` An object inheriting from `List` or `list` representing the training data for the model.

`modelParams` An object inheriting from `List` or `list` representing the parameters needed to train the model.

`models` An object inheriting from `List` or `list` representing the trained models.

`validationStats` An object inheriting from `DFrame` or `data.frame` and storing statistics assessing model performance.

`validationData` An object inheriting `List` or `list` representing the data used to validate or evaluate the performance of a model.

`elementMetadata` A `DataFrame` or `'data.frame'` of item metadata for the `models` slot.

`metadata` A `List` or `list` of model level metadata.

---

<code>sampleClinicalModel</code>	<i>Sample ClinicalModel Containing the ICGC micro-array cohort from MetaGxPancreas as training data.</i>
----------------------------------	--

---

### Description

Sample ClinicalModel Containing the ICGC micro-array cohort from MetaGxPancreas as training data.

### See Also

`MetaGxPancreas::loadPancreasDatasets`

### Examples

```
data(sampleClinicalModel)
```

---

sampleCohortList	<i>A Set of Example Patient Cohorts</i>
------------------	---

---

**Description**

A CohortList object containing sample data for the PCOSP vignette. This data is a subset of the the Pancreas datasets available in MetaGxPancreas.

**See Also**

MetaGxPancreas::loadPancreasDatasets

**Examples**

```
data(sampleCohortList)
```

---

sampleICGCmicro	<i>A Sample SurvivalExperiment Containing Data from the ICGC micro-array cohort from MetaGxPancreas</i>
-----------------	---

---

**Description**

A Sample SurvivalExperiment Containing Data from the ICGC micro-array cohort from MetaGxPancreas

**See Also**

MetaGxPancreas::loadPancreasDatasets

**Examples**

```
data(sampleICGCmicro)
```

---

samplePCOSPmodel	<i>A Sample PCOSP Model Containing the ICGC micro-array cohort from MetaGxPancreas as training data.</i>
------------------	--

---

**Description**

A Sample PCOSP Model Containing the ICGC micro-array cohort from MetaGxPancreas as training data.

**See Also**

MetaGxPancreas::loadPancreasDatasets

**Examples**

```
data(samplePCOSPmodel)
```

---

samplePCOSPpredList	<i>Sample CohortList with PCOSP Risk Predictions</i>
---------------------	--

---

**Description**

Sample CohortList with PCOSP Risk Predictions

**See Also**

MetaGxPancreas::loadPancreasDatasets

**Examples**

```
data(samplePCOSPpredList)
```

---

samplePCSIurvExp	<i>Sample SurvivalExperiment Containing the PCSI rna-sequencing cohort from MetaGxPancreas.</i>
------------------	---

---

**Description**

Used as validation data for modelling examples

**See Also**

MetaGxPancreas::loadPancreasDatasets

**Examples**

```
data(samplePCSIurvExp)
```

---

sampleRGAmode1	<i>Sample RGA Model Containing the ICGC micro-array cohort from MetaGxPancreas as training data.</i>
----------------	--

---

**Description**

Sample RGA Model Containing the ICGC micro-array cohort from MetaGxPancreas as training data.

**See Also**

MetaGxPancreas::loadPancreasDatasets

**Examples**

```
data(sampleRGAmode1)
```

---

sampleRLSmodel	<i>Sample RLS Model Containing the ICGC micro-array cohort from MetaGxPancreas as training data.</i>
----------------	--

---

### **Description**

Sample RLS Model Containing the ICGC micro-array cohort from MetaGxPancreas as training data.

### **See Also**

MetaGxPancreas::loadPancreasDatasets

### **Examples**

```
data(sampleRLSmodel)
```

---

sampleTrainedPCOSPmodel	<i>A Sample Trained PCOSP Model Containing the ICGC micro-array cohort from MetaGxPancreas as training data.</i>
-------------------------	--

---

### **Description**

A Sample Trained PCOSP Model Containing the ICGC micro-array cohort from MetaGxPancreas as training data.

### **See Also**

MetaGxPancreas::loadPancreasDatasets

### **Examples**

```
data(sampleTrainedPCOSPmodel)
```

---

sampleValPCOSPmodel    *Sample Validated PCOSP Model for Plotting Examples*

---

**Description**

Sample Validated PCOSP Model for Plotting Examples

**See Also**

MetaGxPancreas::loadPancreasDatasets

**Examples**

```
data(sampleValPCOSPmodel)
```

---

show, S4Model-method    *Show method for Classes Inheriting from S4Model*

---

**Description**

Show method for Classes Inheriting from S4Model

**Usage**

```
## S4 method for signature 'S4Model'  
show(object)
```

**Arguments**

object            A S4Model derivative to show.

**Value**

None, prints to console.

**Examples**

```
data(CSPC_MAE)  
set.seed(1987)  
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)  
metaclustModel
```



---

 subset, CohortList-method

*Subset method for a CohortList*


---

### Description

Works using endoapply of `[]` over the list `SurvivalExperiments`

### Usage

```
## S4 method for signature 'CohortList'
subset(x, subset = TRUE, select = TRUE, invert = FALSE)
```

### Arguments

<code>x</code>	A <code>CohortList</code> object
<code>subset</code>	The row query. Defaults to <code>TRUE</code> , i.e., select all.
<code>select</code>	The column query. Defaults to <code>TRUE</code> , i.e., select all.
<code>invert</code>	A logical vector indicating if the matches should be inverted. Default is <code>FALSE</code> .

### Value

A `CohortList` containing only the rows and columns selected in `i` and `j`, respectively.

### Examples

```
data(sampleCohortList)
commonGenes <- findCommonGenes(sampleCohortList)
commonGenesCohortList <- subset(sampleCohortList, subset=commonGenes)
```

---

<code>SurvivalExperiment</code>	<i>Constructor for SurvivalExperiment Class Builds a SurvivalExperiment object, which is just a wrapper for a SummarizedExperiment with mandatory survival metadata numeric columns survival_time and event_occurred.</i>
---------------------------------	---

---

### Description

Constructor for `SurvivalExperiment` Class

Builds a `SurvivalExperiment` object, which is just a wrapper for a `SummarizedExperiment` with mandatory survival metadata numeric columns `survival_time` and `event_occurred`.

**Usage**

```
SurvivalExperiment(
  ...,
  survival_time = "survival_time",
  event_occurred = "event_occurred"
)
```

**Arguments**

... pairlist Fall through arguments to the SummarizedExperiment constructor. If the first argument to dots is a SummarizedExperiment, that object is used instead.

survival\_time A character vector indicating the column name in colData which contains the integer number of days a patient has survived since treatment at the time of data collection. If event\_occurred is 1/TRUE, then this is the number of days the patient lived.

event\_occurred A character vector indicating the column name in colData which contains logical or integer values where 0/FALSE means a patient is alive and 1/TRUE means a patient is deceased.

**Value**

A SurvivalExperiment object.

**Examples**

```
data(sampleICGCmicro)

# build a SurvivalExperiment from raw data
ICGCmicro <- SurvivalExperiment(assays=assays(sampleICGCmicro),
  rowData=rowData(sampleICGCmicro), colData=colData(sampleICGCmicro),
  metadata=metadata(sampleICGCmicro), survival_time='survival_time',
  event_occurred='event_occurred')

# build a SurvivalExperiment from an existig SummarizedExperment
ICGCmicroSumExp <- as(sampleICGCmicro, 'SummarizedExperiment')
ICGCmicro <- SurvivalExperiment(ICGCmicroSumExp,
  survival_time='survival_time', event_occurred='event_occurred')
```

---

SurvivalExperiment-class

*SurvivalExperiment Class*

---

**Description**

A SummarizedExperiment with mandatory numeric survival metadata columns survival\_time and event\_occurred.

---

SurvivalModel	<i>Constructor for a SurvivalModel Object.</i>
---------------	--

---

**Description**

Constructor for a SurvivalModel Object.

**Usage**

```
SurvivalModel(trainCohorts, minDaysSurvived = 365, ..., randomSeed)
```

**Arguments**

trainCohorts	A 'SurvivalExperiment' containing training data for the SurvivalModel object.
minDaysSurvived	An integer minimum number of days survived to be classified as a 'good' prognosis.
...	Force subsequent parameters to be named. Not used.
randomSeed	An integer randomSeed that was used to train the model. Users should specify this when initializing a model to ensure reproducibility.

**Value**

A SurvivalModel object.

**Examples**

```
data(sampleICGcmicro)
set.seed(1987)
survModel <- SurvivalModel(sampleICGcmicro, minDaysSurvived=365,
  randomSeed=1987)
```

---

SurvivalModel-class	<i>A Generic Container for Storing Mathematical Models of Survival Experiments</i>
---------------------	--

---

**Description**

An S4 class with a number of predefined methods for accessing slots relevant to a survival model. More specific model types will inherit from this class for their accessor methods and constructor.

**Slots**

`models` A `SimpleList` containing one or more model object.

`validationData` A `CohortList` containing one or more `SurvivalExperiment` objects used to validate the model. This slot is populated by the when the `validateModel` method is called on a `SurvivalModel` object.

`validationStats` A `data.frame` object containing validation statistics calculated by the `validateModel` method.

**Examples**

```
data(sampleICGCmicro)
set.seed(1987)
survModel <- SurvivalModel(sampleICGCmicro, minDaysSurvived=385,
  randomSeed=1987)
```

---

trainData

*Generic for Accessing the Training Data of an S4 Object*

---

**Description**

Generic for Accessing the Training Data of an S4 Object

**Usage**

```
trainData(object, ...)
```

**Arguments**

`object` An S4 object to retrieve training data from.  
`...` Allow new parameters to be defined for this generic.

**Value**

The training data for an S4 object.

**Examples**

```
data(CSPC_MAE)
set.seed(1987)
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)
```

---

`trainData,S4Model-method`*Accessor for the Training Data in a S4Model Object*

---

**Description**

Accessor for the Training Data in a S4Model Object

**Usage**

```
## S4 method for signature 'S4Model'  
trainData(object)
```

**Arguments**

`object` An S4Model object to retrieve training data from.

**Value**

The training data for an S4Model Object.

---

`trainData<-`*Generic for Accessing the Training Data of an S4 Object*

---

**Description**

Generic for Accessing the Training Data of an S4 Object

**Usage**

```
trainData(object, ...) <- value
```

**Arguments**

`object` An S4 object to retrieve training data from.  
`...` Allow new parameters to be defined for this generic.  
`value` An object to place in the objects training data slot.

**Value**

None, updates the object.

**Examples**

```
data(CSPC_MAE)
set.seed(1987)
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)
trainData(metaclustModel) <- CSPC_MAE
```

---

trainModel	<i>Train a Model Based on the Data in an S4 Object</i>
------------	--

---

**Description**

Train a Model Based on the Data in an S4 Object

**Usage**

```
trainModel(object, ...)
```

**Arguments**

object	An S4 object representing an untrained statistical or machine. learning model.
...	Allow new method to be defined for this generic.

**Value**

The same object with the @model slot populated with the fit model

**Examples**

```
data(samplePCOSPmodel)
set.seed(getModelSeed(samplePCOSPmodel))

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

trainModel(samplePCOSPmodel, numModels=5, minAccuracy=0.6)
```

---

trainModel,ClinicalModel-method

*Fit a GLM Using Clinical Predictors Specified in a ClinicalModel Object.*

---

## Description

Fit a GLM Using Clinical Predictors Specified in a ClinicalModel Object.

## Usage

```
## S4 method for signature 'ClinicalModel'
trainModel(
  object,
  ...,
  family = binomial(link = "logit"),
  na.action = na.exclude
)
```

## Arguments

object	A ClinicalModel object, with survival data for the model in the colData slot.
...	Fall through parameters to the <code>stats::glm</code> function.
family	Argument to the family parameter of <code>stats::glm</code> . Defaults to <code>binomial(link='logit')</code> . This parameter must be named.
na.action	Argument to the na.action parameter of <code>stats::glm</code> . Defaults to 'na.omit', dropping rows with NA values in one or more of the formula variables.

## Value

A ClinicalModel object with a glm object in the models slot.

## Examples

```
data(sampleClinicalModel)
set.seed(getModelSeed(sampleClinicalModel))

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

trainedClinicalModel <- trainModel(sampleClinicalModel)
```

---

 trainModel,ConsensusMetaclusteringModel-method

*Train A ConsensusMetaclusteringModel*


---

### Description

Since consensus clustering is an unsupervised learning method, there isn't really a 'training step' per se. Instead this method computes the consensus clusters and stores the results in the models slot.

### Usage

```
## S4 method for signature 'ConsensusMetaclusteringModel'
trainModel(
  object,
  maxK = 5,
  reps = 10,
  distance = "pearson",
  clusterAlg = "hc",
  plot = NULL,
  ...
)
```

### Arguments

object	A ConsensusMetaclusteringModel to train.
maxK	The maximum number of clusters to test. Defaults to 5.
reps	How many random samples should clustering be repeated on? Default is 10, but 1000+ is recommended for real world use.
distance	The distance method to use. Defaults to 'pearson'. See ?ConsensusClusterPlus::ConsensusClusterPlus for more options.
clusterAlg	The clustering algorithm to use. Defaults to 'hc'. See ?ConesnsusClusterPlus::ConsensusClusterPlus for more options.
plot	An optional path to output the plots generated by each call to ConsensusClusterPlus::ConsensusClusterPlus. Default is NULL, which suppresses all plots, otherwise passed to the clustering function.
...	Fall through parameters to BiocParallel::bplapply. This can be used to customize your parallelization using BPPARAM or to pass additional arguments to ConsensusClusterPlus.

### Value

The ConsensusMetaclusteringModel with the clustering results in the models slot.



---

`trainModel,CoxModel-method`*Fit Models to the trainData in a CoxModel Object*

---

**Description**

Computes models with the survival package for coxph, survfit, survdiff as well as computes the fit p-values using pchisq with the chisq values from survdiff. Modelling data is stored in modelData, as well as a data.table with all model data merged in modelDT. These items are all assigned to the models slot.

**Usage**

```
## S4 method for signature 'CoxModel'  
trainModel(object)
```

**Arguments**

object            A CoxModel object to fit models for.

**Value**

A CoxModel object with the results of coxph, survfit and survdiff in the models slot as lists where each item corresponds to the data in modelData. For convenience, all the model data has also been merged into a single data.table in the modelDT item of models.

---

`trainModel,GeneFuModel-method`*Train a GeneFuModel Object*

---

**Description**

Train a GeneFuModel Object

**Usage**

```
## S4 method for signature 'GeneFuModel'  
trainModel(object)
```

**Arguments**

object            A GeneFuModel object to train.

**Value**

An error message, since we have not finished implementing this functionality yet.

---

 trainModel,NCSModel-method

*Train a NetworkCommunitySearchModel*


---

### Description

Train a NetworkCommunitySearchModel

### Usage

```
## S4 method for signature 'NCSModel'
trainModel(object, alpha = 0.05, minRepro = 0.5, minCor = 0)
```

### Arguments

object	An NCSModel object, created from a ConsensusMetaclusteringModel.
alpha	A float specifying the significance level for cluster reproducibility. Default is 0.05.
minRepro	A float specifying the minimum in-group proportion (IGP) for a cluster to be included in the metacluster labels. Default is 0.5.
minCor	A float specifying the minimum correlation between a centroid and assay cluster to be included in the metacluster labels. Default is 0.0.

### Value

The NCSModel from object with the networkEdges item of the models slot filtered based on the specified criteria. The criteria are also stored in the modelParam slots to ensure reproducibility.

---

 trainModel,PCOSP-method

*Train a PCOSP Model Based on The Data the assay trainMatrix.*


---

### Description

Uses the switchBox SWAP.Train.KTSP function to fit a number of k top scoring pair models to the data, filtering the results to the best models based on the specified paramters.

### Usage

```
## S4 method for signature 'PCOSP'
trainModel(object, numModels = 10, minAccuracy = 0.6, ...)
```

**Arguments**

object	A PCOSP object to train.
numModels	An integer specifying the number of models to train. Defaults to 10. We recommend using 1000+ for good results.
minAccuracy	A float specifying the balanced accuracy required to consider a model 'top scoring'. Defaults to 0.6. Must be in the range [0, 1].
...	Fall through arguments to BiocParallel::bplapply. Use this to configure parallelization options. By default the settings inferred in BiocParallel::bpparam() will be used.

**Details**

This function is parallelized with BiocParallel, thus if you wish to change the back-end for parallelization, number of threads, or any other parallelization configuration please pass BPPARAM to bplapply.

**Value**

A PCOSP object with the trained model in the model slot.

**See Also**

switchBox::SWAP.KTSP.Train BiocParallel::bplapply

**Examples**

```
data(samplePCOSPmodel)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

set.seed(getModelSeed(samplePCOSPmodel))
trainModel(samplePCOSPmodel, numModels=2, minAccuracy=0.6)
```

---

trainModel, RGAModel-method

*Train a RGAModel Based on the Data in the assays slot.*

---

**Description**

Uses the switchBox SWAP.Train.KTSP function to fit a number of k top scoring pair models to the data, filtering the results to the best models based on the specified parameters.

**Usage**

```
## S4 method for signature 'RGAModel'
trainModel(object, numModels = 10, minAccuracy = 0, ...)
```

**Arguments**

object	A RGAModel object to train.
numModels	An integer specifying the number of models to train. Defaults to 10. We recommend using 1000+ for good results.
minAccuracy	A float specifying the balanced accuracy required to consider a model 'top scoring'. Defaults to 0. Must be in the range 0 to 1.
...	Fall through arguments to BiocParallel::bplapply.

**Details**

This function is parallelized with BiocParallel, thus if you wish to change the back-end for parallelization, number of threads, or any other parallelization configuration please pass BPPARAM to bplapply.

**Value**

A RGAModel object with the trained model in the model slot.

**See Also**

switchBox::SWAP.KTSP.Train BiocParallel::bplapply

**Examples**

```
data(sampleRGAModel)
set.seed(getModelSeed(sampleRGAModel))

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

trainedRGAModel <- trainModel(sampleRGAModel, numModels=2, minAccuracy=0)
```

---

trainModel,RLSModel-method

*Train a PCOSP Model Based on The Data the assay trainMatrix.*

---

**Description**

Uses the switchBox SWAP.Train.KTSP function to fit a number of k top scoring pair models to the data, filtering the results to the best models based on the specified paramters.

**Usage**

```
## S4 method for signature 'RLSModel'
trainModel(object, numModels = 10, minAccuracy = 0, ...)
```

**Arguments**

object	A PCOSP object to train.
numModels	An integer specifying the number of models to train. Defaults to 10. We recommend using 1000+ for good results.
minAccuracy	This parameter should be set to zero, since we do not expect the permuted models to perform well. Setting this higher will result in an ensemble with very few models included.
...	Fall through arguments to <code>BiocParallel::bplapply</code> . Use this to configure parallelization options. By default the settings inferred in <code>BiocParallel::bpparam()</code> will be used.

**Details**

This function is parallelized with `BiocParallel`, thus if you wish to change the back-end for parallelization, number of threads, or any other parallelization configuration please pass `BPPARAM` to `bplapply`.

**Value**

A PCOSP object with the trained model in the `model` slot.

**See Also**

`switchBox::SWAP.KTSP.Train` `BiocParallel::bplapply`

**Examples**

```
data(sampleRLSmodel)
set.seed(getModelSeed(sampleRLSmodel))

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

trainedRLSmodel <- trainModel(sampleRLSmodel, numModels=2)
```

---

validateModel	<i>Perform Validation on an S4 Object Representing a Trained Model</i>
---------------	--

---

**Description**

Perform Validation on an S4 Object Representing a Trained Model

**Usage**

```
validateModel(model, valData, ...)
```

**Arguments**

model            An S4 object.  
 valData         Any Data to verify the model with.  
 ...             Allow new parameters to be defined for this generic.

**Value**

The S4 object with added model performance metadata.

**Examples**

```
data(sampleTrainedPCOSPmodel)
data(samplePCOSPpredList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Validate model
validatedPCOSPmodel <- validateModel(sampleTrainedPCOSPmodel,
  valData=samplePCOSPpredList[[1]])
```

---

validateModel,ClinicalModel,CohortList-method

*Evaluate the Performance of a List of Trained KTSP Models from a PCOSP Model*

---

**Description**

Evaluate the Performance of a List of Trained KTSP Models from a PCOSP Model

**Usage**

```
## S4 method for signature 'ClinicalModel,CohortList'
validateModel(model, valData, ...)
```

**Arguments**

model            A trained ClinicalModel object, as returned by the trainModel method.  
 valData         A CohortList containing one or more SurvivalExperiments. The first assay in each SurvivalExperiment will be classified using all top scoring KTSP models in models(model).  
 ...             Fallthrough arguments to BiocParallel::bplapply, use this to configure the parallelization settings for this function. For example to specify BPARAM.

**Value**

The model object with the validationStats and validationData slots occupied.

**See Also**

[BiocParallel::bplapply](#), [switchBox::SWAP.KTSP.Classify](#)

**Examples**

```
data(sampleClinicalModel)
data(samplePCSIurvExp)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train Model
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Make predictions
clinicalPredSurvExp <- predictClasses(samplePCSIurvExp,
  model=trainedClinicalModel)

# Validate model
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=clinicalPredSurvExp)
```

---

validateModel, ClinicalModel, SurvivalExperiment-method

*Validate a ClinicalModel object with a single SurvivalExperiment object.*

---

**Description**

Validate a ClinicalModel object with a single SurvivalExperiment object.

**Usage**

```
## S4 method for signature 'ClinicalModel, SurvivalExperiment'
validateModel(model, valData)
```

**Arguments**

model            A ClinicalModel object which has been trained using trainModel.  
valData         A SurvivalExperiment to validate the model with.

**Value**

The ClinicalModel with the validation statistics in the validationStats slot and the validation data in the validationData slot.

**Examples**

```

data(sampleClinicalModel)
data(sampleCohortList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Train Model
trainedClinicalModel <- trainModel(sampleClinicalModel)

# Make predictions
clinicalPredCohortList <- predictClasses(sampleCohortList[c('PCSI', 'TCGA')],
  model=trainedClinicalModel)

# Validate model
validatedClinicalModel <- validateModel(trainedClinicalModel,
  valData=clinicalPredCohortList)

```

---

```

validateModel,ConsensusMetaclusteringModel,ConsensusMetaclusteringModel-method
  Compute the Inter-Cohort Cluster Correlation and Clustering Reproducibility of All Clusters in Each Cohort.

```

---

**Description**

Compute the Inter-Cohort Cluster Correlation and Clustering Reproducibility of All Clusters in Each Cohort.

**Usage**

```

## S4 method for signature
## 'ConsensusMetaclusteringModel,ConsensusMetaclusteringModel'
validateModel(model, valData, ...)

```

**Arguments**

model	A ConsensusMetaclusteringModel object with cluster_labels in assigned to each experiment, as returned by predictClasses.
valData	A ConsensusMetaclusteringModel object with cluster_labels assigned to each experiment, as returned by predictClasses. This consensus cluster should contain outgroup cohorts, such as normal patients to be compared against the disease cohorts being used for class discovery.
...	Fallthrough parameters to BiocParallel::bpmapply. This can also be used to customize the call to stats::cor.test used for calculating the cluster thresholds.



**Value**

The ConsensusMetaclusteringModel from object, with the training data from valData in the validationData slot, the models from the valData object appended to the models of object, and the validationStats slot populated with pair-wise comparisons between all experiments in both model and valData.

---

validateModel, GeneFuModel, CohortList-method

*Evaluate the Performance of a List of Trained KTSP Models from a PCOSP Model*

---

**Description**

Evaluate the Performance of a List of Trained KTSP Models from a PCOSP Model

**Usage**

```
## S4 method for signature 'GeneFuModel, CohortList'
validateModel(model, valData, ...)
```

**Arguments**

model	A GeneFuModel with a DataFrame of gene coefficients in the models slot.
valData	A CohortList containing one or more SurvivalExperiments. The first assay in each SurvivalExperiment will be classified using all top scoring KTSP models in models(model).
...	Fallthrough arguments to BiocParallel::bplapply, use this to configure the parallelization settings for this function. For example to specify BPARAM.

**Value**

The model object with the validationStats and validationData slots occupied.

**See Also**

[BiocParallel::bplapply](#), [switchBox::SWAP.KTSP.Classify](#)

**Examples**

```
data(sampleTrainedPCOSPmodel)
data(samplePCOSPpredList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Validate model
validatedPCOSPmodel <- validateModel(sampleTrainedPCOSPmodel,
  valData=samplePCOSPpredList)
```

---

validateModel, GeneFuModel, SurvivalExperiment-method

*Validate a GeneFuModel object with a single SurvivalExperiment object.*

---

### Description

Validate a GeneFuModel object with a single SurvivalExperiment object.

### Usage

```
## S4 method for signature 'GeneFuModel, SurvivalExperiment'
validateModel(model, valData)
```

### Arguments

model	A GeneFuModel object which has been trained using trainModel.
valData	A SurvivalExperiment to validate the model with.

### Value

The GeneModel with the validation statistics in the validationStats slot and the validation data in the validationData slot.

---

validateModel, PCOSP\_or\_RLS\_or\_RGA, CohortList-method

*Evaluate the Performance of a List of Trained KTSP Models from a PCOSP Model*

---

### Description

Evaluate the Performance of a List of Trained KTSP Models from a PCOSP Model

### Usage

```
## S4 method for signature 'PCOSP_or_RLS_or_RGA, CohortList'
validateModel(model, valData, ...)
```

### Arguments

model	A PCOSP model which has been trained using trainModel.
valData	A CohortList containing one or more SurvivalExperiments. The first assay in each SurvivalExperiment will be classified using all top scoring KTSP models in models(model).
...	Fallthrough arguments to BiocParallel::bplapply, use this to configure the parallelization settings for this function. For example to specify BPARAM.

**Value**

The model object with the validationStats and validationData slots occupied.

**See Also**

[BiocParallel::bplapply](#), [switchBox::SWAP.KTSP.Classify](#)

**Examples**

```
data(sampleTrainedPCOSPmodel)
data(samplePCOSPpredList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Validate model
validatedPCOSPmodel <- validateModel(sampleTrainedPCOSPmodel,
  valData=samplePCOSPpredList)
```

---

validateModel,PCOSP\_or\_RLS\_or\_RGA,SurvivalExperiment-method

*Validate a PCOSP model with a single SurvivalExperiment object.*

---

**Description**

Validate a PCOSP model with a single SurvivalExperiment object.

**Usage**

```
## S4 method for signature 'PCOSP_or_RLS_or_RGA,SurvivalExperiment'
validateModel(model, valData)
```

**Arguments**

model	A PCOSP model which has been trained using trainModel.
valData	A SurvivalExperiment to validate the model with.

**Value**

The PCOSPmodel with the validation statistics in the validationStats slot and the validation data in the validationData slot.

**Examples**

```

data(sampleTrainedPCOSModel)
data(samplePCOSPpredList)

# Set parallelization settings
BiocParallel::register(BiocParallel::SerialParam())

# Validate model
validatedPCOSModel <- validateModel(sampleTrainedPCOSModel,
  valData=samplePCOSPpredList)

```

---

validationData	<i>Accessor for the validationData slot of an S4 object</i>
----------------	---

---

**Description**

Accessor for the validationData slot of an S4 object

**Usage**

```
validationData(object, ...)
```

**Arguments**

object	An S4 object
...	Allow definition of new arguments to this generic.

**Value**

A List- or list-like object containing one or more sets of validation data.

**Examples**

```

data(CSPC_MAE)
set.seed(1987)
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)
validationData(metaclustModel)

```

---

validationData,S4Model-method

*Accessor for the validationData slot of an S4Model object*

---

**Description**

Accessor for the validationData slot of an S4Model object

**Usage**

```
## S4 method for signature 'S4Model'  
validationData(object)
```

**Arguments**

object            An S4Model object

**Value**

A List- or list-like object containing one or more sets of validation data.

---

validationData,SurvivalModel-method

*Accessor for the validationData slot of a SurvivalModel object.*

---

**Description**

Accessor for the validationData slot of a SurvivalModel object.

**Usage**

```
## S4 method for signature 'SurvivalModel'  
validationData(object)
```

**Arguments**

object            A SurvivalModel object.

**Value**

A CohortList object containing the datasets used to compute validation statistics for this model.

**Examples**

```
data(samplePCOSPmodel)  
validationData(samplePCOSPmodel)
```

---

validationData<-            *Generic for setting the validationData slot on an S4 object*

---

### Description

Generic for setting the validationData slot on an S4 object

### Usage

```
validationData(object, ...) <- value
```

### Arguments

object	An S4 object.
...	Allow definition of additional parameters to this generic.
value	A data. frame of validation statistics.

### Value

None, updates the object

### Examples

```
data(CSPC_MAE)
set.seed(1987)
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)
validationData(metaclustModel) <- list(cohort1='This should be cohort data')
```

---

validationData<- ,SurvivalModel,CohortList-method  
*Setter for the validationData slot of a SurvivalModel object with a CohortList.*

---

### Description

Setter for the validationData slot of a SurvivalModel object with a CohortList.

### Usage

```
## S4 replacement method for signature 'SurvivalModel,CohortList'
validationData(object) <- value
```

### Arguments

object	A SurvivalModel model.
value	A CohortList of validation cohorts for the SurvivalModel model.

**Value**

None, updates the object.

**Examples**

```
data(samplePCOSModel)
validationData(samplePCOSModel) <- validationData(samplePCOSModel)
```

---

validationStats	<i>Accessor for the validationStats slot of an S4 object</i>
-----------------	--

---

**Description**

Accessor for the validationStats slot of an S4 object

**Usage**

```
validationStats(object, ...)
```

**Arguments**

object	An S4 object
...	Allow definition of new arguments to this generic.

**Value**

A data.frame of validation statistics for the validation data provided to validateModel function for a given S4 object.

**Examples**

```
data(CSPC_MAE)
set.seed(1987)
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)
validationStats(metaclustModel)
```

---

validationStats, S4Model-method

*Accessor for the validationStats slot of an S4Model Object*

---

### Description

Accessor for the validationStats slot of an S4Model Object

### Usage

```
## S4 method for signature 'S4Model'
validationStats(object)
```

### Arguments

object            An S4 object

### Value

A data.frame of validation statistics for the validation data provided to validateModel function for a given S4Model object.

---

validationStats, SurvivalModel-method

*Accessor for the validationStats slot of a SurvivalModel object.*

---

### Description

Accessor for the validationStats slot of a SurvivalModel object.

### Usage

```
## S4 method for signature 'SurvivalModel'
validationStats(object)
```

### Arguments

object            A SurvivalModel object to get validation statistics from.

### Value

A data.table of validation statistics for the SurvivalModel object.

### Examples

```
data(samplePCOSPmodel)
validationStats(samplePCOSPmodel)
```



---

```
validationStats<-      Setter for the validationStats slot on an S4 object
```

---

**Description**

Setter for the validationStats slot on an S4 object

**Usage**

```
validationStats(object, ...) <- value
```

**Arguments**

object	An S4 object.
...	Allow definition of additional parameters to this generic.
value	A DataFrame- or data.frame-like object of validation statistics.

**Value**

None, updates the object

**Examples**

```
data(CSPC_MAE)
set.seed(1987)
metaclustModel <- ConMetaclustModel(CSPC_MAE, randomSeed=1987)
validationStats(metaclustModel) <- data.frame()
```

---

```
validationStats<- ,S4Model,DFrame_or_data.frame_data.table_or_NULL-method
      Setter for the validationStats slot on an S4Model object
```

---

**Description**

Setter for the validationStats slot on an S4Model object

**Usage**

```
## S4 replacement method for signature 'S4Model,DFrame_or_data.frame_data.table_or_NULL'
validationStats(object) <- value
```

**Arguments**

object	An S4Model object.
value	A DataFrame- or data.frame-like object of validation statistics.

**Value**

None, updates the object

---

```
validationStats<-
```

,*SurvivalModel*,*data.frame*-method

*Setter for the validationStats slot of a SurvivalModel object with a data.frame*

---

**Description**

Setter for the validationStats slot of a SurvivalModel object with a data.frame

**Usage**

```
## S4 replacement method for signature 'SurvivalModel,data.frame'  
validationStats(object) <- value
```

**Arguments**

object	A SurvivalModel model.
value	A data.frame of validation statistics for a SurvivalModel object.

**Value**

None, updated the object.

**Examples**

```
data(samplePCOSPmodel)  
validationStats(samplePCOSPmodel) <- data.frame()
```

# Index

- [.ClinicalModel \(ClinicalModel-class\), 13](#)
- [.CohortList \(CohortList-class\), 14](#)
- [.ConsensusMetaclusteringModel \(ConsensusMetaclusteringModel-class\), 18](#)
- [.GeneFuModel \(GeneFuModel-class\), 31](#)
- [.ModelComparison \(ModelComparison-class\), 37](#)
- [.NCSModel \(NCSModel-class\), 41](#)
- [.PCOSP \(PCOSP-class\), 47](#)
- [.RGAModel \(RGAModel-class\), 65](#)
- [.RLSModel \(RLSModel-class\), 65](#)
- [.S4Model \(S4Model-class\), 67](#)
- [.SurvivalExperiment \(SurvivalExperiment-class\), 74](#)
- [.SurvivalModel \(SurvivalModel-class\), 75](#)
  
- [assignColDataColumn, 6](#)
- [assignSubtypes, 6](#)
- [assignSubtypes, CohortList, list-method, 7](#)
- [assignSubtypes, SurvivalExperiment, data.frame-method, 8](#)
  
- [barPlotModelComparison, 9](#)
- [barPlotModelComparison, ClinicalModel, PCOSP\\_or\\_RLS\\_or\\_RGA-method, 10](#)
- [BiocGenerics::mad, 61](#)
- [BiocParallel::bplapply, 57, 87, 89, 91](#)
- [birnbaum, 11](#)
  
- [caret::preProcess, 44, 45, 58](#)
- [chen, 12](#)
- [ClinicalModel, 12](#)
- [ClinicalModel-class, 13](#)
- [CohortList, 13](#)
- [CohortList-class, 14](#)
- [cohortSubtypeDFs, 14](#)
- [compareModels, 14](#)
- [compareModels, ModelComparison, SurvivalModel-method, 15](#)
- [compareModels, SurvivalModel, SurvivalModel-method, 16](#)
- [ConMetaclustModel \(ConsensusMetaclusteringModel\), 17](#)
- [ConsensusClusterPlus::ConsensusClusterPlus, 18](#)
- [ConsensusMetaclusteringModel, 17](#)
- [ConsensusMetaclusteringModel-class, 18](#)
- [CoxModel, 18](#)
- [CSPEC\\_MAE, 19](#)
  
- [densityPlotModelComparison, 19](#)
- [densityPlotModelComparison, PCOSP\\_or\\_RLS\\_or\\_RGA, PCOSP\\_or\\_RLS\\_or\\_RGA-method, 20](#)
- [dplyr::dense\\_rank, 61](#)
- [dropNotCensored, 21](#)
- [dropNotCensored, CohortList-method, 21](#)
- [dropNotCensored, SurvivalExperiment-method, 22](#)
- [existingClassifierData, 23](#)
  
- [findCommonGenes, 23](#)
- [findCommonGenes, RLS\\_or\\_RGA-method, 24](#)
- [findCommonGenes, CohortList-method, 24](#)
- [findCommonGenes, MultiAssayExperiment-method, 24](#)
- [findCommonSamples, 25](#)
- [findCommonSamples, CohortList-method, 26](#)
- [forestPlot, 26](#)
- [forestPlot, ModelComparison-method, 27](#)
- [forestPlot, PCOSP\\_or\\_ClinicalModel-method, 28](#)
  
- [genefu::sig.score, 53](#)
- [GeneFuModel, 30](#)
- [GeneFuModel-class, 31](#)

- getModelSeed, 31
- getModelSeed, SurvivalModel-method, 32
- getTopFeatures, 32
- getTopFeatures, MultiAssayExperiment-method, 33
- getTopFeatures, PCOSP-method, 33
- getTopFeatures, SummarizedExperiment-method, 34
- haiderSigScores, 35
- hasColDataColumns, 35
- matrixStats::weightedMedian, 61
- merge, SurvivalExperiment, SurvivalExperiment-method, 36
- ModelComparison, 36
- ModelComparison-class, 37
- modelParams, 38
- modelParams, S4Model-method, 38
- models, 39
- models, S4Model-method, 39
- models, SurvivalModel-method, 40
- models<-, SurvivalModel, SimpleList-method, 40
- NCSModel (NetworkCommunitySearchModel), 41
- NCSModel-class, 41
- NetworkCommunitySearchModel, 41
- normalize, data.frame\_or\_matrix-method, 42
- normalize, DFrame-method, 42
- normalize, MultiAssayExperiment-method, 43
- normalize, SummarizedExperiment-method, 44
- normalsMAE, 45
- optimalKMinimizeAmbiguity, 45
- PCOSP, 46
- PCOSP-class, 47
- PCOSP\_or\_ClinicalModel-class, 47
- PCOSP\_or\_RLS\_or\_RGA-class, 47
- piano::runGSAhyper, 66
- plotNetworkGraph, 48
- plotNetworkGraph, NCSModel-method, 48
- plotROC, 49
- plotROC, PCOSP-method, 49
- plotSurvivalCurves, 50
- plotSurvivalCurves, CoxModel-method, 50
- predictClasses, 51
- predictClasses, CohortList, ClinicalModel-method, 52
- predictClasses, CohortList, GeneFuModel-method, 53
- predictClasses, CohortList, PCOSP\_or\_RLS\_or\_RGA-method, 53
- predictClasses, ConsensusMetaclusteringModel, ANY-method, 54
- predictClasses, NCSModel, ANY-method, 55
- predictClasses, SurvivalExperiment, ClinicalModel-method, 55
- predictClasses, SurvivalExperiment, GeneFuModel-method, 56
- predictClasses, SurvivalExperiment, PCOSP\_or\_RLS\_or\_RGA-method, 57
- preprocessCaret, 44, 45, 58
- RandomGeneAssignmentModel, 59
- RandomLabelShufflingModel, 59
- rankFeatures, 60
- rankFeatures, MultiAssayExperiment-method, 61
- rankFeatures, SummarizedExperiment-method, 62
- removeColDataFactorColumns, 63
- removeFactorColumns, 63
- renameColDataColumns, 64
- renameColumns, 64
- RGAModel (RandomGeneAssignmentModel), 59
- RGAModel-class, 65
- RLSModel (RandomLabelShufflingModel), 59
- RLSModel-class, 65
- runGSEA, 65
- runGSEA, PCOSP, data.frame-method, 66
- S4Model-class, 67
- sampleClinicalModel, 67
- sampleCohortList, 68
- sampleICGmicro, 68
- samplePCOSPmodel, 69
- samplePCOSPpredList, 69
- samplePCSIurvExp, 70
- sampleRGAModel, 70
- sampleRLSmodel, 71
- sampleTrainedPCOSPmodel, 71
- sampleValPCOSPmodel, 72

show, S4Model-method, 72  
stats::glm, 79  
stats::predict, 52, 56, 58  
stats::predict.glm, 52, 56  
subset, CohortList-method, 73  
SurvivalExperiment, 73  
SurvivalExperiment-class, 74  
SurvivalModel, 75  
SurvivalModel-class, 75  
switchBox::SWAP.KTSP.Classify, 57, 87,  
89, 91

trainData, 76  
trainData, S4Model-method, 77  
trainData<-, 77  
trainModel, 78  
trainModel, ClinicalModel-method, 79  
trainModel, ConsensusMetaclusteringModel-method,  
80  
trainModel, CoxModel-method, 81  
trainModel, GeneFuModel-method, 81  
trainModel, NCSModel-method, 82  
trainModel, PCOSP-method, 82  
trainModel, RGAModel-method, 83  
trainModel, RLSModel-method, 84

validateModel, 85  
validateModel, ClinicalModel, CohortList-method,  
86  
validateModel, ClinicalModel, SurvivalExperiment-method,  
87  
validateModel, ConsensusMetaclusteringModel, ConsensusMetaclusteringModel-method,  
88  
validateModel, GeneFuModel, CohortList-method,  
89  
validateModel, GeneFuModel, SurvivalExperiment-method,  
90  
validateModel, PCOSP\_or\_RLS\_or\_RGA, CohortList-method,  
90  
validateModel, PCOSP\_or\_RLS\_or\_RGA, SurvivalExperiment-method,  
91  
validationData, 92  
validationData, S4Model-method, 93  
validationData, SurvivalModel-method,  
93  
validationData<-, 94  
validationData<-, SurvivalModel, CohortList-method,  
94  
validationStats, 95  
validationStats, S4Model-method, 96  
validationStats, SurvivalModel-method,  
96  
validationStats<-, 97  
validationStats<-, S4Model, DFrame\_or\_data.frame\_data.table,  
97  
validationStats<-, SurvivalModel, data.frame-method,  
98