Package 'DGEobj'

January 20, 2025

Type Package

Title Differential Gene Expression (DGE) Analysis Results Data Object

Version 1.1.2

Description Provides a flexible container to manage and annotate Differential Gene Expression (DGE) analysis results (Smythe et. al (2015) <doi:10.1093/nar/gkv007>). The DGEobj has data slots for row (gene), col (samples), assays (matrix n-rows by m-samples dimensions) and metadata (not keyed to row, col, or assays). A set of accessory functions to deposit, query and retrieve subsets of a data workflow has been provided. Attributes are used to capture metadata such as species and gene model, including reproducibility information such that a 3rd party can access a DGEobj history to see how each data object was created or modified. Since the DGEobj is customizable and extensible it is not limited to RNA-seq analysis types of workflows -- it can accommodate nearly any data analysis workflow that starts from a matrix of assays (rows) by samples (columns).

Depends R (>= 3.5.0)

License GPL-3

Language en-US

Encoding UTF-8

Imports assertthat, magrittr, stringr, utils

Suggests biomaRt, conflicted, dplyr, edgeR, GenomicRanges, glue, knitr, rmarkdown, testthat

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Author John Thompson [aut], Connie Brett [aut, cre], Isaac Neuhaus [aut], Ryan Thompson [aut]

Maintainer Connie Brett <connie@aggregate-genius.com>

Repository CRAN

Date/Publication 2022-05-16 07:10:12 UTC

Contents

`DGEobj-package` *DGEobj Package Overview*

Description

DGEobj is an S3 data class that provides a flexible container for Differential Gene Expression (DGE) analysis results. The DGEobj class is designed to be extensible allowing definition of new data types as needed. A set of accessory functions to deposit, query and retrieve subsets of a data workflow has been provided. Attributes are used to capture metadata such as species and gene model, including reproducibility information such that a 3rd party can access a DGEobj history to see how each data object was created or modified.

Details

Operationally, the DGEobj is influenced by the RangedSummarizedExperiment (RSE). The DGEobj has data slots for row (gene), col (samples), assays (anything with n-rows by m-samples dimensions) and metadata (anything that can't be keyed to row, col or assay). The key motivation for creating the DGEobj data structure is that the RSE only allows one data item each in the row and col slots and thus is unsuitable for capturing the plethora of data objects created during a typical DGE workflow. The DGEobj data structure can hold any number of row and col data objects and thus is suitable for capturing the multiple steps of a downstream analysis.

Certain object types, primarily the count matrix and associated row and column info, are defined as unique which means only one instance of that type may be added to the DGEobj.

When multiple objects of one type are included in a DGEobj (e.g. two different fits), the concept of parent attributes is used to associate downstream data objects (e.g. contrasts) with the appropriate data object they are derived from.

More Information

```
browseVignettes(package = 'DGEobj')
```

`addItem` *Add a data item*

Description

Add a data item

Usage

```
addItem(
  dgeObj,
  item,
  itemName,
  itemType,
  funArgs = match.call(),
  itemAttr,
  parent = "",
  overwrite = FALSE,
  init = FALSE
)
```

Arguments

`dgeObj`	A class DGEobj created by function initDGEobj()
`item`	The data item to be deposited in the DGEobj
`itemName`	The user-assigned name for this data item
`itemType`	The type attribute. See showTypes() to see the predefined types – types are extensible with the newType() function.

funArgs	(optional) A text field to annotate how the data object was created. If the result of match.call() is passed as this argument, the name and arguments used in the current function are captured
itemAttr	(optional) A named list of attributes to add directly to the item
parent	(optional) itemName of the parent of this item
overwrite	Whether to overwrite a matching data object stored in the itemName slot (default = FALSE)
init	Internal Use (default = FALSE)

Value

A DGEobj

Examples

```
## Not run:
   myFunArgs <- match.call()  #  Capture calling function and arguments

   myDGEobj <- addItem(myDGEobj, item     = MyCounts,
                                 itemName = "counts",
                                 itemType = "counts",
                                 funArgs  = myFunArgs)

## End(Not run)
```

---

addItems *Add multiple data items*

Description

Add multiple data items

Usage

```
addItems(dgeObj, itemList, itemTypes, parents, itemAttr, overwrite = FALSE)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()
itemList	A named list of data items to add to DGEobj
itemTypes	A list of type values for each item on itemList
parents	(optional) A list of parent values for each item on itemList (optional, but highly recommended)
itemAttr	(optional) An named list of attributes to add to each item. These attributes will be attached to all items in the call.
overwrite	Whether to overwrite a matching data object stored in the itemName slot (default = FALSE)

Value

A DGEobj

Examples

```
## Not run:
   # NOTE: Requires the edgeR package

   # Add normalized counts and log2CPM as additional "assay" items in the DGEobj
   dgeObj  <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
   dgeList <- edgeR::calcNormFactors(edgeR::DGEList(dgeObj$counts), method="TMM")
   log2cpm <- edgeR::cpm(dgeList, log = TRUE)

   dgeObj <- addItems(dgeObj,
                      itemList = list(newDgelist = dgeList, Log2CPM = log2cpm),
                      itemTypes = list("assay", "assay"),
                      parents = list("counts", "newDgelist")
   )
   inventory(dgeObj)

## End(Not run)
```

---

annotateDGEobj *Add annotations*

Description

Reads an annotation file containing key/value pairs or a named list and attaches them attributes to a DGEobj. If a file is used, it should be a text file containing key/value pairs separated by an equals sign. The keys argument specifies which keys we want to capture as attributes on the DGEobj.

Usage

```
annotateDGEobj(dgeObj, annotations, keys = NULL)
```

Arguments

dgeObj	A object of class DGEobj created by function initDGEobj()
annotations	Either a character string path to a file with annotations given as key/value pairs separated by an equal sign, or a named list of key/value pairs
keys	By default (value = NULL), all keys are read in and applied as DGEobj attributes. Use the keys argument to specify a specific list of keys to read from the file.

Value

A DGEobj

Examples

```
   MyDgeObj <- system.file("exampleObj.RDS", package = "DGEobj")

## Not run:
   #using a text file file of key=value pairs
   annotationFile <- "/location/to/myAnnotations.txt"
   MyDgeObj <- annotateDGEobj(MyDgeObj, annotationFile)

   #using a named list of key/values
   annotations <- list(Title     = "Rat Liver Slices from Bile Duct Ligation animals",
                       Organism  = "Rat",
                       GeneModel = "Ensembl.R89")
   MyDgeObj <- annotateDGEobj(MyDgeObj, annotations)

## End(Not run)
```

`as.list.DGEobj` *Cast as a simple list*

Description

Cast as a simple list

Usage

```
## S3 method for class 'DGEobj'
as.list(x, ...)
```

Arguments

`x`	A DGEobj
`...`	Additional parameters

Value

A named list

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

mylist <- as.list(exObj)
```

baseType *Get the baseType of an internal data item*

Description

Get the baseType of an internal data item

Usage

```
baseType(dgeObj, type)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()
type	An item type for which you want the baseType

Value

character string

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

baseType(exObj, type = "DGEList")
```

baseTypes *Get a list of the available baseTypes*

Description

Get a list of the available baseTypes

Usage

```
baseTypes(dgeObj)
```

Arguments

dgeObj	(optional) A class DGEobj created by function initDGEobj()

Value

A character vector of baseTypes

Examples

```
# Global definition of baseTypes
baseTypes()

# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

# Basetypes from a specific DGEobj
baseTypes(exObj)
```

`dim.DGEobj` *Get the "assay" dimensions (row/genes by col/samples)*

Description

Returns the dimensions of the assay data (baseType)

Usage

```
## S3 method for class 'DGEobj'
dim(x)
```

Arguments

`x`	A class DGEobj created by function initDGEobj()

Value

An integer vector [r,c] with a length of 2.

`dimnames.DGEobj` *Get the "assay" names (row/genes by col/samples)*

Description

Returns a list of length 2 containing the the assay data names (baseType)

Usage

```
## S3 method for class 'DGEobj'
dimnames(x)
```

Arguments

`x`	A class DGEobj created by function initDGEobj()

Value

A list of length 2 containing rownames and colnames of the DGEobj

---

`getAttributes` *Get all attributes*

Description

Get all user-defined attributes from a DGEobj except for any listed in the excludeList argument.

Usage

```
getAttributes(
  dgeObj,
  excludeList = list("dim", "dimnames", "names", "row.names", "class")
)
```

Arguments

`dgeObj`	A DGEobj
`excludeList`	A list of attribute names to exclude from the output (default = list("dim", "dimnames", "names", "row.names"))

Value

A named list

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

getAttributes(exObj)

# Get the formula attribute from the design (if set)
attr(exObj$design, "formula")
```

getBaseType *Retrieve data items by baseType*

Description

Retrieve data items by baseType

Usage

```
getBaseType(dgeObj, baseType)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()
baseType	One or more of: ["row", "col", "assay", "meta"]

Value

A list of data items

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

Assays <- getBaseType(exObj, baseType = "assay")
AssaysAndMeta <- getBaseType(exObj, c("assay", "meta"))
```

getItem *Retrieve a data item by name*

Description

Retrieve a data item by name

Usage

```
getItem(dgeObj, itemName)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()
itemName	Name of item to retrieve

Value

The requested data item

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

MyCounts <- getItem(exObj, "counts")
```

`getItems` *Retrieve multiple data items by name*

Description

Retrieve multiple data items by name

Usage

```
getItems(dgeObj, itemNames)
```

Arguments

`dgeObj`	A class DGEobj created by function initDGEobj()
`itemNames`	A character string, character vector, or list names to retrieve

Value

A list

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

myList <- getItems(exObj, list("counts", "geneData"))
names(myList)
```

`getType` *Retrieve data items by type*

Description

Retrieve data items by type

Usage

```
getType(dgeObj, type, parent)
```

Arguments

`dgeObj`	A class DGEobj created by function initDGEobj()
`type`	A type or list of types to retrieve
`parent`	(optional) Filter return list for common parent (e.g. useful to select one set of contrast results when multiple fits have been performed)

Value

A list of data items

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

MyRawData      <- getType(exObj, type = list("counts", "design", "geneData"))
```

`initDGEobj` *Initialize with base data (primaryAssayData, row annotations, col annotations)*

Description

Initialize with base data (primaryAssayData, row annotations, col annotations)

Usage

```
initDGEobj(
  primaryAssayData,
  rowData,
  colData,
  level,
  customAttr,
  allowShortSampleIDs = FALSE,
  DGEobjDef = initDGEobjDef()
)
```

Arguments

primaryAssayData	A numeric matrix or dataframe with row and colnames. Each column represents a sample. Each row represents and assay. This is typically the counts matrix in a DGE RNA-Seq experiment.
rowData	Gene, exon, isoform or protein level annotation. Rownames must match the rownames in primaryAssayData
colData	A dataframe describing the experiment design. Rownames much match the colnames(primaryAssayData)
level	One of "gene", "exon", "isoform" or "protein"
customAttr	(optional) Named list of attributes
allowShortSampleIDs	Using sequential integer rownames (even if typed as character) is discouraged and by default will abort the DGEobj creation. If you have a legitimate need to have short sample names composed of numeric characters, you can set this argument to TRUE (default = FALSE)
DGEobjDef	An object definition. Defaults to the global DGEobj definition (initDGEobjDef()) and you usually shouldn't change this unless you're customizing the object for new data types.

Value

A DGEobj

Examples

```
dgeObj   <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))
MyCounts <- dgeObj$counts
geneinfo <- dgeObj$geneData
sampinfo <- dgeObj$design

myDgeObj <- initDGEobj(primaryAssayData = MyCounts,
                       rowData          = geneinfo,
                       colData          = sampinfo,
                       level            = "gene",
                       customAttr       = list (Genome = "Rat.B6.0",
```

```
                                        GeneModel = "Ensembl.R89"))
```

`initDGEobjDef` *Instantiate a class DGEobjDef object.*

Description

Instantiate a class DGEobjDef object.

Usage

```
initDGEobjDef(levels, primaryAssayNames, types, uniqueTypes)
```

Arguments

`levels`	A character string or vector providing names for new levels
`primaryAssayNames`	A character string or vector, must be the same length as levels This argument supplies the primaryAssayNames for the corresponding levels.
`types`	A named character vector of new types where the values indicate the basetype for each named type (optional)
`uniqueTypes`	A name or vector of names to add to the uniqueType list (optional)

Value

A class DGEobjDef object suitable for use with initDGEobj

Examples

```
# return the default DGEobj definition
myDGEobjDef <- initDGEobjDef()

# Optionally add some new types and levels for metabolomics data
 myDGEobjDef <- initDGEobjDef(levels = "metabolomics",
                              primaryAssayNames = "intensity",
                              types <- c(normalizedIntensity = "assay"))

# When a new level is defined, the itemNames and types for the
# rowData and colData are automatically established.  The
# types argument is only needed to define downstream workflow objects.
```

inventory *Retrieve the object inventory*

Description

Retrieve the object inventory

Usage

```
inventory(dgeObj, verbose = FALSE)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()
verbose	Include funArgs column in the output (default = FALSE)

Value

A data.frame summarizing the data contained in the DGEobj

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

inventory(exObj)
```

newType *Add a new type definition to a DGEobj*

Description

Add a new type definition to a DGEobj

Usage

```
newType(dgeObj, itemType, baseType, uniqueItem = FALSE)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()
itemType	The name of the new type to create
baseType	The baseType of the new item. One of [row, col, assay, meta]
uniqueItem	If set to TRUE, only one instance of the new type is allowed in a DGEobj

Value

A DGEobj

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

exObj <- newType(exObj,
                 itemType   = "AffyRMA",
                 baseType   = "assay",
                 uniqueItem = TRUE)
```

---

print.DGEobj *Print the Inventory*

Description

Print the Inventory

Usage

```
## S3 method for class 'DGEobj'
print(x, ..., verbose = FALSE)
```

Arguments

x	A class DGEobj created by function initDGEobj()
...	Additional parameters
verbose	Include funArgs column in the output (default = FALSE)

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

print(exObj)
```

resetDGEobj *Reset to original data*

Description

During a workflow, a DGEobj typically gets filtered down to remove samples that fail QC or non-expressed genes. The resetDGEobj() function produces a new DGEobj with the original unfiltered data. Resetting an object does not restore changes to attributes or class, but does revert changes made with addItem() and rmItem(). Reset requires that *_orig data is still in the DGEobj.

Usage

```
resetDGEobj(dgeObj)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()

Value

A DGEobj

Examples

```
#example object
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

# subset to first 10 rows to show reset functionality
exObj <- exObj[c(1:10), ]

exObj <- resetDGEobj(exObj)
dim(exObj)
```

rmItem *Removes a named data item*

Description

Removes a named data item

Usage

```
rmItem(dgeObj, itemName)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()
itemName	Name of the item to remove

Value

A DGEobj

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

exObj <- rmItem(exObj, "design")
```

---

setAttributes *Set attributes*

Description

Set one or more attributes on a DGEobj or on a specific item within a DGEobj.

Usage

```
setAttributes(dgeObj, attribs)
```

Arguments

dgeObj	A DGEobj
attribs	A named list of attribute/value pairs

Details

This function adds attributes without deleting the attributes that are already present. Any named attribute that already exists in the object will be updated. To remove an attribute from an object pass NULL as the attribute value.

Value

A DGEobj

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

# Assign attributes to a DGEobj
MyAttributes <- list(Platform       = "RNA-Seq",
                     Instrument     = "HiSeq",
                     Vendor         = "Unknown",
                     readType       = "PE",
                     readLength     = 75,
                     strandSpecific = TRUE)
exObj <- setAttributes(exObj, MyAttributes)

# Set attributes on an item inside a DGEobj
MyAttributes <- list(normalized   = FALSE,
                     LowIntFilter = "FPK >5 in >= 1 group")
exObj[["counts"]] <- setAttributes(exObj[["counts"]], MyAttributes)
```

---

showAttributes *Print attributes*

Description

This function prints all attributes regardless of the class of the attribute value.

Usage

```
showAttributes(
  dgeObj,
  skipList = c("dim", "dimnames", "rownames", "colnames", "listData", "objDef")
)
```

Arguments

dgeObj	A DGEobj
skipList	A character vector of attributes to skip. Use this to avoid printing certain lengthy attributes like rownames. Defaults to c("dim", "dimnames", "rownames", "colnames", "listData", "objDef")

Details

Note Use showMeta() to only retrieve attributes that are key/value pairs.

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

showAttributes(exObj)
```

---

showMeta *Retrieve the Key/Value metadata attributes that have a character value and length of 1*

Description

Retrieve the Key/Value metadata attributes that have a character value and length of 1

Usage

```
showMeta(dgeObj)
```

Arguments

dgeObj	A DGEobj with attributes

Value

A data.frame with "Attribute" and "Value" columns

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

showMeta(exObj)
```

---

showTypes *Returns and prints the list of all defined types*

Description

Returns and prints the list of all defined types

Usage

```
showTypes(dgeObj)
```

Arguments

dgeObj	A class DGEobj created by function initDGEobj()

Value

data.frame

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

showTypes(exObj)
```

---

subset.DGEobj *Subset internal row or column data*

Description

Subset internal row or column data

Usage

```
## S3 method for class 'DGEobj'
subset(x, ..., row, col, drop = FALSE, debug = FALSE)
```

Arguments

x	A class DGEobj created by function initDGEobj()
...	Additional parameters
row	Row index for the subset
col	Col index for the subset
drop	Included for compatibility only
debug	(default = FALSE) Set to TRUE to get additional information on the console if subsetting a DGEobj fails with a dimension error.

Value

A DGEobj

Examples

```
# example DGEobj
exObj <- readRDS(system.file("miniObj.RDS", package = "DGEobj"))

exObj <- subset(exObj, 1:10, 5:50)
```

`[.DGEobj` *Subset with square brackets*

Description

Subset with square brackets

Usage

```
## S3 method for class 'DGEobj'
x[...]
```

Arguments

`x`	A DGEobj
`...`	Additional parameters

Value

A DGEobj

Index