

# Package: RGenEDA (via r-universe)

October 30, 2026

**Title** Genomic Exploratory Data Analysis

**Version** 0.99.0

**Description** Provides a unified and reproducible framework for performing genomic exploratory data analysis (EDA). Highly repetitive functions and analyses have been collated into an easy-to-use S4 object that produces quick results and customizable visualizations to assess variation in your data. See <<https://github.com/mikemartinez99/RGenEDA>>.

**License** GPL-3

**URL** <https://github.com/mikemartinez99/RGenEDA>,  
<https://mikemartinez99.github.io/RGenEDA/>

**BugReports** <https://github.com/mikemartinez99/RGenEDA/issues>

**biocViews** Software, RNASeq, GeneExpression, Transcriptomics, Sequencing, DifferentialExpression, PrincipalComponent, QualityControl, Visualization

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Depends** methods, R (>= 4.4.0)

**Imports** cli, pheatmap, RColorBrewer, dplyr, ggplot2 (>= 3.4.0),  
ggrepel, grid, patchwork, scales, tidyr, vegan, stats,  
SummarizedExperiment, tools, utils

**Suggests** DESeq2, ggtext, pasilla, tidyverse, knitr, rmarkdown,  
testthat (>= 3.0.0), withr

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**Config/pak/sysreqs** libicu-dev zlib1g-dev

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

**Date/Publication** 2026-10-30 00:47:49 UTC

**RemoteUrl** <https://github.com/BiocStaging/RGenEDA>

**RemoteRef** HEAD

**RemoteSha** ad98b0ce95e1ec506bece35899da10a1e8d01b5e

## Contents

|                                |    |
|--------------------------------|----|
| as.geneda . . . . .            | 3  |
| DEDashboard . . . . .          | 4  |
| DEGs . . . . .                 | 6  |
| DimReduction . . . . .         | 7  |
| extractEigen . . . . .         | 7  |
| ExtractPCA . . . . .           | 8  |
| FilterDEGs . . . . .           | 9  |
| FindHVDEGs . . . . .           | 10 |
| FindVariableFeatures . . . . . | 11 |
| GenEDA . . . . .               | 12 |
| geneda-class . . . . .         | 13 |
| generatePCs . . . . .          | 13 |
| GenSave . . . . .              | 14 |
| getCounts . . . . .            | 15 |
| getMetadata . . . . .          | 16 |
| getNormalized . . . . .        | 16 |
| HVGs . . . . .                 | 17 |
| PlotCountDist . . . . .        | 18 |
| PlotDistances . . . . .        | 18 |
| PlotEigenCorr . . . . .        | 20 |
| PlotEigenHeatmap . . . . .     | 21 |
| PlotHVGVariance . . . . .      | 22 |
| PlotMA . . . . .               | 23 |
| PlotOrdCorr . . . . .          | 24 |
| PlotPCA . . . . .              | 25 |
| PlotPValHist . . . . .         | 26 |
| PlotScree . . . . .            | 27 |
| PlotVolcano . . . . .          | 28 |
| RunPCA . . . . .               | 29 |
| SetDEGs . . . . .              | 30 |
| show,geneda-method . . . . .   | 31 |
| Snail1KO . . . . .             | 31 |
| SummarizeDEGs . . . . .        | 32 |

**Index**

**34**

as.geneda

*Convert a SummarizedExperiment to a geneda object***Description**

Builds a geneda object from a SummarizedExperiment, including DESeqDataSet objects from DESeq2 and DESeqTransform objects returned by DESeq2::vst() or DESeq2::rlog(). Raw counts, normalized values, and sample metadata (colData) are carried over, so the object is ready for FindVariableFeatures(), RunPCA(), and the plotting functions.

**Usage**

```
as.geneda(x, ...)

## S4 method for signature 'SummarizedExperiment'
as.geneda(
  x,
  normalized = NULL,
  counts = "counts",
  transform = c("vst", "logcpm"),
  DEGs = NULL,
  ...
)
```

**Arguments**

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x          | A SummarizedExperiment (or subclass such as DESeqDataSet or DESeqTransform) with features as rows and samples as columns.                                                                                                                                                                                                                                                                                                                                                                      |
| ...        | Not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| normalized | Where the normalized values come from. One of: <ul style="list-style-type: none"> <li>• NULL (default): computed from the counts assay using transform. If x is a DESeqTransform, its transformed values are used instead.</li> <li>• A single assay name or index in x.</li> <li>• A SummarizedExperiment with the same features and samples as x, for example the output of DESeq2::vst(x). Its first assay is used.</li> <li>• A matrix with the same row and column names as x.</li> </ul> |
| counts     | Name or index of the raw counts assay in x, or NULL to leave the counts slot empty. Default "counts". If the assay is not present, the counts slot is left empty, unless counts are needed to compute normalized.                                                                                                                                                                                                                                                                              |
| transform  | Normalization used when normalized = NULL: "vst" (variance stabilizing transformation from DESeq2, blind = TRUE) or "logcpm" (log2 counts per million with a pseudocount of 1). Default "vst", which requires the DESeq2 package and integer counts.                                                                                                                                                                                                                                           |
| DEGs       | Optional named list of DEG tables (data.frame or DESeqResults) to store with SetDEGs(), using the list names as assay names.                                                                                                                                                                                                                                                                                                                                                                   |

## Details

Sample metadata are taken from `colData(x)` and converted to a `data.frame`. Columns added by DESeq2, such as `sizeFactor`, are kept and can be tested with `PlotEigenCorr()` like any other covariate.

Assays stored as sparse or delayed matrices are converted to base matrices with `as.matrix()`.

## Value

A `geneda` object.

## See Also

[GenEDA\(\)](#) to build a `geneda` object from matrices.

## Examples

```
data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample

se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(counts = counts),
  colData = meta
)
obj <- as.geneda(se, transform = "logcpm")
obj

# From a DESeqDataSet, with VST-normalized values and DEGs
if (requireNamespace("DESeq2", quietly = TRUE)) {
  meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail_KO"))
  dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
  dds <- DESeq2::DESeq(dds, quiet = TRUE)
  obj <- as.geneda(dds,
    normalized = DESeq2::vst(dds),
    DEGs = list(KO_vs_WT = DESeq2::results(dds))
  )
  obj
}
```

## Description

Generate a multi-panel differential expression dashboard for a single assay in a geneda object. The dashboard includes:

1. P-value histogram
2. MA plot
3. Volcano plot

## Usage

```
DEDashboard(object, assay, alpha = 0.05, l2fc = 1, num = NULL, den = NULL)
```

## Arguments

|        |                                                                                                                       |
|--------|-----------------------------------------------------------------------------------------------------------------------|
| object | A geneda object containing DEGs in the DEGs slot.                                                                     |
| assay  | Character. Name of the assay to visualize.                                                                            |
| alpha  | Numeric. Significance threshold for highlighting p-values and DEGs (default 0.05).                                    |
| l2fc   | Numeric. Log2 fold-change threshold for MA and volcano plots.                                                         |
| num    | Character. Name of the numerator group for volcano plot labeling. If NULL (default), the legend uses "numerator".     |
| den    | Character. Name of the denominator group for volcano plot labeling. If NULL (default), the legend uses "denominator". |

## Value

A patchwork object combining plots.

## Examples

```
data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail_KO"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail_KO vs WT)
dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
db <- DEDashboard(obj,
  assay = "KO_vs_WT", alpha = 0.05, l2fc = 1,
  num = "Snail_KO", den = "WT"
)
# The dashboard needs a large device, e.g.
```

```
# ggplot2::ggsave("dashboard.png", db, width = 14, height = 10)
class(db)
```

---

|      |                              |
|------|------------------------------|
| DEGs | <i>Access DEGs container</i> |
|------|------------------------------|

---

**Description**

Access DEGs container

**Usage**

```
DEGs(object, assay)
```

**Arguments**

- object            A geneda object
- assay            The name of the DEGs slot to return

**Value**

The DEG data.frame stored under assay, or NULL if it is not set

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail1_K0"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail1_K0 vs WT)
dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
head(DEGs(obj, "KO_vs_WT"))
```

---

|              |                                                  |
|--------------|--------------------------------------------------|
| DimReduction | <i>Access dimensional reduction results list</i> |
|--------------|--------------------------------------------------|

---

**Description**

Access dimensional reduction results list

**Usage**

```
DimReduction(object)
```

**Arguments**

object                      A geneda object

**Value**

List with Loadings, Eigenvectors, percent\_var

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 1000)
obj <- RunPCA(obj)
pca <- DimReduction(obj)
names(pca)
head(pca$Scores[, 1:3])
```

---

|              |                     |
|--------------|---------------------|
| extractEigen | <i>extractEigen</i> |
|--------------|---------------------|

---

**Description**

Extract gene-level Eigen-vectors for PCs of interest and calculate percent variance.

**Usage**

```
extractEigen(object, component)
```

**Arguments**

|           |                                                                 |
|-----------|-----------------------------------------------------------------|
| object    | A geneda object containing PCA results in the DimReduction slot |
| component | Principal component to use (i.e., PC1)                          |

**Value**

A data-frame containing gene loading information and percent variation per gene

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 1000)
obj <- RunPCA(obj)
pc1 <- extractEigen(obj, component = "PC1")
head(pc1[order(-abs(pc1$EigenVector)), ])
```

---

ExtractPCA

---

*Extract PCA Loadings and Metadata*


---

**Description**

This function extracts PCA loadings stored in the DimReduction slot of a geneda object and combines them with the associated metadata. It ensures that the metadata has valid rownames and aligns the PCA loadings accordingly.

**Usage**

```
ExtractPCA(object)
```

**Arguments**

|        |                                                                                                                 |
|--------|-----------------------------------------------------------------------------------------------------------------|
| object | A geneda object containing PCA results in the DimReduction slot and sample-level metadata in the metadata slot. |
|--------|-----------------------------------------------------------------------------------------------------------------|

**Details**

The function performs several checks:

- Ensures the input object is of class geneda.
- Verifies that the DimReduction slot contains PCA loadings.
- Confirms that the metadata has valid rownames.
- Reorders the PCA loadings to match the order of metadata rows.

If metadata rownames are missing or invalid, the function throws an error.

**Value**

A data.frame combining PCA loadings and sample metadata, where rows correspond to samples and columns include principal component loadings and metadata fields.

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 1000)
obj <- RunPCA(obj)
pca_df <- ExtractPCA(obj)
head(pca_df[, c("PC1", "PC2", "Genotype")])
```

---

FilterDEGs

---

*Filter DEGs by padj and absolute log2FoldChange*


---

**Description**

Filters the unfiltered DEGs in DEGs[[assay]] and stores the filtered results in a new named slot DEGs[[saveAssay]]. Multiple filtered result sets can be stored with different assay names.

**Usage**

```
FilterDEGs(object, assay, alpha = 0.05, l2fc = 1, saveAssay)
```

**Arguments**

|           |                                                                  |
|-----------|------------------------------------------------------------------|
| object    | A geneda object with a DEG table set via SetDEGs()               |
| assay     | The DEG slot to filter                                           |
| alpha     | Adjusted p-value threshold ( $\leq$ )                            |
| l2fc      | Absolute log2 fold change threshold ( $\geq$ )                   |
| saveAssay | Character name for the filtered result set (e.g., "padj05_lfc1") |

**Value**

Updated geneda object with filtered results stored in DEGs[[saveAssay]]

**Examples**

```

data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail1_KO"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail1_KO vs WT)
dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
obj <- FilterDEGs(obj,
  assay = "KO_vs_WT", alpha = 0.05, l2fc = 1,
  saveAssay = "KO_vs_WT_filtered"
)
nrow(DEGs(obj, "KO_vs_WT_filtered"))

```

FindHVDEGs

*Find DEGs Overlapping With HVGs***Description**

Looks for DEGs after filtering that overlap with HVGs.

**Usage**

```
FindHVDEGs(object, assay, direction = c("positive", "negative", "both"))
```

**Arguments**

|           |                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------|
| object    | A geneda object with a DEG table set via SetDEGs()                                                      |
| assay     | The DEG slot to pull results from (typically a result of FilterDEGs)                                    |
| direction | One of positive, negative, or both (indicating the direction of DEG fold change to intersect with HVGs) |

**Value**

For direction = "positive" or "negative", a character vector of gene IDs that are both DEGs in that direction and HVGs. For direction = "both", a list with elements positive, negative, and both (all overlapping genes).

**Examples**

```

data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail1_KO"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail1_KO vs WT)
dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
obj <- FilterDEGs(obj, assay = "KO_vs_WT", saveAssay = "KO_vs_WT_filtered")
obj <- FindVariableFeatures(obj, nfeatures = 1000)
hv <- FindHVDEGs(obj, assay = "KO_vs_WT_filtered", direction = "both")
lengths(hv)

```

---

FindVariableFeatures    *Find and store highly variable genes (HVGs)*

---

**Description**

Computes per-feature variance on normalized(object), ranks genes by decreasing variance, and stores the top nfeatures gene IDs into the HVGs slot. Returns the updated object.

**Usage**

```
FindVariableFeatures(object, nfeatures)
```

**Arguments**

|           |                         |
|-----------|-------------------------|
| object    | A geneda object         |
| nfeatures | Number of HVGs to store |

**Value**

Updated geneda object

**Examples**

```

data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample

```

```

norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 1000)
length(HVGs(obj))

```

---

GenEDA

---

*Construct a geneda object*


---

## Description

Construct a geneda object

## Usage

```
GenEDA(normalized, metadata, counts = NULL, DEGs = NULL)
```

## Arguments

|            |                                                                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| normalized | Normalized expression matrix (features x samples).                                                                                                                           |
| metadata   | Sample metadata data.frame with row names matching colnames(normalized).                                                                                                     |
| counts     | Optional counts matrix (features x samples).                                                                                                                                 |
| DEGs       | Optional named list of DEG tables (data.frame or DESeqResults), stored with SetDEGs() using the list names as assay names. Each table needs log2FoldChange and padj columns. |

## Details

This constructor does not compute HVGs or PCA. Use FindVariableFeatures() and RunPCA() afterward.

## Value

A geneda object.

## Examples

```

data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj

```

---

geneda-class

*geneda S4 class*


---

### Description

An S4 container for exploratory genomic data analysis. Stores optional raw counts, normalized data, sample metadata, highly variable genes (HVGs), and PCA dimensionality reduction results.

### Slots

counts Optional counts matrix (features x samples). Can be NULL.

normalized Normalized expression matrix (features x samples).

metadata Sample-level metadata data.frame (rows = samples).

HVGs Character vector of selected highly variable gene IDs (row names).

DimReduction List for PCA results with Scores, Eigenvectors, percent\_var.

DEGs List container for differential expression results.

### Examples

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
is(obj, "geneda")
```

---

generatePCs

*generatePCs*


---

### Description

Generate principal component analysis data that can be used in downstream analyses.

### Usage

```
generatePCs(mat, vars, nFeatures)
```

### Arguments

|           |                                                                                      |
|-----------|--------------------------------------------------------------------------------------|
| mat       | A data matrix where rows are features and columns are samples                        |
| vars      | A vector of gene variances (can calculate using RGenEDA::plotVariance) or gene names |
| nFeatures | Number of top features to generate principal components on.                          |

**Value**

A list consisting of 3 slots: Loadings, Eigenvectors, and percent\_var

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
vars <- apply(norm, 1, var)
pcs <- generatePCs(norm, vars, nFeatures = 1000)
pcs$percent_var
```

---

GenSave

---

*Save a pheatmap Objects generated from RGenEDA to File*


---

**Description**

Saves a pheatmap derived RGenEDA heatmap object to a file in various formats including PNG, JPEG, TIFF, BMP, PDF, and SVG.

**Usage**

```
GenSave(
  pheatmap_obj,
  filename,
  width = 8,
  height = 6,
  units = "in",
  res = 300,
  ...
)
```

**Arguments**

|              |                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| pheatmap_obj | A pheatmap object returned by an RGenEDA or pheatmap function.                                                                                   |
| filename     | Character string specifying the path and filename where the heatmap should be saved.                                                             |
| width        | Numeric, the width of the output image. Default is 8.                                                                                            |
| height       | Numeric, the height of the output image. Default is 6.                                                                                           |
| units        | Character, units for width and height: one of "in", "cm", "mm", or "px". Default is "in". Values are converted to inches for PDF and SVG output. |
| res          | Numeric, the resolution (in dpi) for raster images. Default is 300.                                                                              |
| ...          | Additional arguments passed to the graphics device function.                                                                                     |

**Details**

This function is similar in spirit to ggsave() but works specifically with pheatmap objects.

**Value**

Invisibly returns the filename of the saved heatmap.

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
dists <- PlotDistances(obj, meta_cols = "Genotype")
out <- tempfile(fileext = ".png")
GenSave(dists, out, width = 6, height = 5)
file.exists(out)
```

---

getCounts

---

Access counts matrix (avoid name clash with DESeq2::counts)

---

**Description**

Access counts matrix (avoid name clash with DESeq2::counts)

**Usage**

```
getCounts(object)
```

**Arguments**

object                      A geneda object

**Value**

Matrix or NULL

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
getCounts(obj)[1:5, ]
```

---

|             |                                                            |
|-------------|------------------------------------------------------------|
| getMetadata | <i>Access metadata (avoid name clash with S4 generics)</i> |
|-------------|------------------------------------------------------------|

---

**Description**

Access metadata (avoid name clash with S4 generics)

**Usage**

```
getMetadata(object)
```

**Arguments**

|        |                 |
|--------|-----------------|
| object | A geneda object |
|--------|-----------------|

**Value**

data.frame

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
getMetadata(obj)
```

---

|               |                                                                  |
|---------------|------------------------------------------------------------------|
| getNormalized | <i>Access normalized matrix (avoid name clash with generics)</i> |
|---------------|------------------------------------------------------------------|

---

**Description**

Access normalized matrix (avoid name clash with generics)

**Usage**

```
getNormalized(object)
```

**Arguments**

|        |                 |
|--------|-----------------|
| object | A geneda object |
|--------|-----------------|

**Value**

Matrix

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
getNormalized(obj)[1:5, ]
```

HVGs

*Access HVG IDs***Description**

Access HVG IDs

**Usage**

HVGs(object)

**Arguments**

object            A geneda object

**Value**

Character vector

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 500)
head(HVGs(obj))
```

---

|               |                      |
|---------------|----------------------|
| PlotCountDist | <i>PlotCountDist</i> |
|---------------|----------------------|

---

**Description**

Generates violin plots with embedded boxplots showing the distribution of normalized or VST-transformed RNA-Seq counts for each sample in a geneda object. Optionally, plots can be faceted by a metadata variable.

**Usage**

```
PlotCountDist(object, split_by = NULL)
```

**Arguments**

- object           A geneda object containing normalized expression values in @normalized and sample metadata in @metadata.
- split\_by        Character. Optional column name from object@metadata used for faceting (default: NULL, no faceting).

**Value**

A ggplot2 object displaying the distribution of counts per sample.

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
PlotCountDist(obj)
PlotCountDist(obj, split_by = "Genotype")
```

---

|               |                                          |
|---------------|------------------------------------------|
| PlotDistances | <i>Plot a Euclidean Distance Heatmap</i> |
|---------------|------------------------------------------|

---

**Description**

Compute a sample-by-sample distance matrix from a geneda object and plot a ggplot2 heatmap. Optionally subset metadata columns and reorder rows/columns by hierarchical clustering.

**Usage**

```
PlotDistances(
  object,
  method = "euclidean",
  reorder = TRUE,
  meta_cols = NULL,
  palettes = NULL,
  return = c("object", "plot")
)
```

**Arguments**

|                        |                                                                                                                                                  |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>object</code>    | A geneda object                                                                                                                                  |
| <code>method</code>    | Distance method for <code>stats::dist</code> (default "euclidean")                                                                               |
| <code>reorder</code>   | Logical; if TRUE, reorder rows/cols by hclust on the distance matrix. Default TRUE                                                               |
| <code>meta_cols</code> | Optional character vector of metadata columns to display in the title/subtitle for quick reference (no annotation strips are drawn).             |
| <code>palettes</code>  | Optional named list of color vectors that can be used downstream (returned) if the caller wants to annotate elsewhere.                           |
| <code>return</code>    | One of <code>c("object", "plot")</code> . If "plot", draws the heatmap immediately via grid and still returns the result list. Default "object". |

**Value**

A list with elements: `dist_matrix`, `order` (character vector of sample names), `heatmap` (pheatmap object), `palettes` (as passed through)

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
colorList <- list(Genotype = c(WT = "grey40", Snail_K0 = "firebrick3"))
dists <- PlotDistances(obj,
  meta_cols = "Genotype", palettes = colorList,
  return = "plot"
)
```

**Description**

Calculate Eigen-correlations from a geneda object and plot a publication-quality heatmap using ggplot2 with numeric labels and significance stars. For continuous variables, Pearson correlation is used. P values follow this convention:  $p < 0.001$ ,  $p < 0.01$ ,  $p < 0.05$  = three, two, one stars, respectively.

Two-level categorical variables are coded by factor level order, so the sign of the correlation follows the level order. Categorical variables with more than two levels are summarized by the multiple correlation R (the square root of  $R^2$  from a one-way linear model, `score ~ variable`). R is unsigned (0 to 1) and its p-value comes from the model F-test, so results do not depend on how levels are labelled. Variables that cannot be tested (a single level, constant values, or no residual degrees of freedom, e.g. a sample ID column) are dropped.

Requires PCA scores in `DimReduction(object)`; does not recompute PCA.

**Usage**

```
PlotEigenCorr(object, num_pcs = 10, meta_cols = NULL)
```

**Arguments**

|                        |                                                                                                      |
|------------------------|------------------------------------------------------------------------------------------------------|
| <code>object</code>    | A geneda object containing normalized and metadata, and optionally <code>DimReduction</code> scores. |
| <code>num_pcs</code>   | Number of principal components to correlate.                                                         |
| <code>meta_cols</code> | Optional character vector of metadata column names to include. Defaults to all metadata columns.     |

**Value**

A list with elements: `cor_matrix`, `pval_matrix`, `stars`, `plot` (ggplot)

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 1000)
obj <- RunPCA(obj)
ec <- PlotEigenCorr(obj, num_pcs = 5, meta_cols = "Genotype")
ec$plot
```

---

|                  |                         |
|------------------|-------------------------|
| PlotEigenHeatmap | <i>PlotEigenHeatmap</i> |
|------------------|-------------------------|

---

### Description

Generates a pheatmap of the top genes contributing to a specific principal component from a geneda object. Genes are selected by absolute eigenvector loading for the chosen PC, and expression values are Z-score scaled per gene. Optionally annotate columns with metadata variables and add row annotations showing gene-level percent variance.

### Usage

```
PlotEigenHeatmap(
  object,
  pc = "PC1",
  n = 25,
  annotate_by = NULL,
  annotate_colors = NULL,
  return = c("object", "plot")
)
```

### Arguments

|                 |                                                                                   |
|-----------------|-----------------------------------------------------------------------------------|
| object          | A geneda object containing normalized expression data and PCA in DimReduction.    |
| pc              | Character. Principal component to visualize (e.g., "PC1").                        |
| n               | Integer. Number of genes to select by absolute loading (default: 25).             |
| annotate_by     | Character vector of metadata column names for column annotations (optional).      |
| annotate_colors | Named list of color vectors for metadata columns. Names should match annotate_by. |
| return          | One of c("object", "plot"). If "plot", draws the heatmap via grid.                |

### Value

A list with topGenes (data.frame with EigenVector and PctVar), expression (scaled matrix), and heatmap (pheatmap object)

### Examples

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 1000)
obj <- RunPCA(obj)
```

```

res <- PlotEigenHeatmap(obj,
  pc = "PC1", n = 25,
  annotate_by = "Genotype",
  annotate_colors = list(Genotype = c(
    WT = "grey40", Snail_KO =
      "firebrick3"
  )),
  return = "plot"
)

```

---

PlotHVGVariance

*PlotHVGVariance*


---

### Description

Plot variance for all genes (features) on a geneda object using ggplot2.

### Usage

```
PlotHVGVariance(object, transform = NULL, dropTopN = 0)
```

### Arguments

|           |                                                                             |
|-----------|-----------------------------------------------------------------------------|
| object    | A geneda object created by GenEDA().                                        |
| transform | One of c("log") to log-transform variance, or NULL for none. Default = NULL |
| dropTopN  | Number of top-most variable genes to drop prior to plotting. Default = 0    |

### Value

A ggplot2 object visualizing the variance of HVGs

### Examples

```

data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
PlotHVGVariance(obj, transform = "log")

```

PlotMA

*PlotMA***Description**

Plot MA plot from a geneda object. Requires that you have ran SetDEGs on your object following differential expression in DESeq2.

**Usage**

```
PlotMA(object, assay, alpha, l2fc, title = NULL)
```

**Arguments**

|        |                                                                            |
|--------|----------------------------------------------------------------------------|
| object | A geneda object containing DEGs from SetDEGs method                        |
| assay  | The DEG slot to use for visualization                                      |
| alpha  | Threshold for adjusted p-values (padj column from DESeq2)                  |
| l2fc   | Absolute value log2Fold-change magnitude threshold (log2FoldChange column) |
| title  | Optional character vector of what plot should be titled.                   |

**Details**

Thresholds are inclusive (padj <= alpha, |log2FoldChange| >= l2fc), matching FilterDEGs().

**Value**

A ggplot2 object

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail_KO"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail_KO vs WT)
dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
PlotMA(obj, assay = "KO_vs_WT", alpha = 0.05, l2fc = 1)
```

**Description**

Calculate correlations between NMDS ordination axes (from beta diversity distances) and metadata, and plot a ggplot2 heatmap with numeric labels and significance stars.

For continuous variables, Pearson correlation is used. P values follow this convention:  $p < 0.001$ ,  $p < 0.01$ ,  $p < 0.05$  = three, two, one stars, respectively.

Two-level categorical variables are coded by factor level order, so the sign of the correlation follows the level order. Categorical variables with more than two levels are summarized by the multiple correlation R (the square root of  $R^2$  from a one-way linear model,  $\text{score} \sim \text{variable}$ ). R is unsigned (0 to 1) and its p-value comes from the model F-test, so results do not depend on how levels are labelled. Variables that cannot be tested (a single level, constant values, or no residual degrees of freedom, e.g. a sample ID column) are dropped.

Default num\_mds is 10, but function will internally set max number of NMDS to be  $n-1$  where  $n$  is your number of samples.

**Usage**

```
PlotOrdCorr(object, num_mds = 10, meta_cols = NULL, distance = "bray")
```

**Arguments**

|           |                                                                                                  |
|-----------|--------------------------------------------------------------------------------------------------|
| object    | A geneda object containing normalized and metadata.                                              |
| num_mds   | Number of NMDS axes to correlate. Default = 10                                                   |
| meta_cols | Optional character vector of metadata column names to include. Defaults to all metadata columns. |
| distance  | Distance metric for vegdist (default "bray").                                                    |

**Value**

A list with elements: cor\_matrix, pval\_matrix, stars, plot (ggplot), and variance explained (squared correlations between ordination distances and observed dissimilarities.)

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
set.seed(42)
oc <- PlotOrdCorr(obj, num_mds = 3, meta_cols = "Genotype")
oc$plot
```

PlotPCA

*Plot PCA results from a GenEDA object***Description**

Visualize principal component analysis (PCA) results stored within a GenEDA object. This function extracts PCA data via [ExtractPCA](#) and provides a flexible ggplot2-based visualization interface.

**Usage**

```
PlotPCA(
  object,
  x = 1,
  y = 2,
  color_by,
  colors = NULL,
  split_by = NULL,
  shape_by = NULL,
  return_data = FALSE
)
```

**Arguments**

|                          |                                                                                             |
|--------------------------|---------------------------------------------------------------------------------------------|
| <code>object</code>      | A GenEDA object containing PCA results in the <code>DimReduction</code> slot.               |
| <code>x</code>           | Numeric or character. The principal component to plot on the x-axis (e.g., 1 or "PC1").     |
| <code>y</code>           | Numeric or character. The principal component to plot on the y-axis (e.g., 2 or "PC2").     |
| <code>color_by</code>    | Character. Column name in the metadata used to color points.                                |
| <code>colors</code>      | Vector. Custom colors to use for plotting                                                   |
| <code>split_by</code>    | Character (optional). Column name in metadata used for faceting (creates separate panels).  |
| <code>shape_by</code>    | Character (optional). Column name in metadata used to control point shape.                  |
| <code>return_data</code> | Logical (default = FALSE), whether or not to return pca dataframe for more custom plotting. |

**Value**

A ggplot object displaying the PCA scatter plot, or a list of `pca_df` and plot if `return_data = TRUE`

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
```

```

meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 1000)
obj <- RunPCA(obj)
PlotPCA(obj,
  x = 1, y = 2, color_by = "Genotype",
  colors = c(WT = "grey40", Snail1_KO = "firebrick3")
)

```

---

PlotPValHist

*PlotPValHist*


---

### Description

Generate a p-value histogram from a DEG table inside a geneda object. This function uses raw p-values only and highlights p-values below a given threshold (default 0.05).

### Usage

```
PlotPValHist(object, assay, bins = 40, alpha = 0.05, title = NULL)
```

### Arguments

|        |                                                                                                |
|--------|------------------------------------------------------------------------------------------------|
| object | A geneda object containing DEGs in the DEGs slot.                                              |
| assay  | The name of the DEG table to use (must contain a "pvalue" column).                             |
| bins   | Number of histogram bins (default 40).                                                         |
| alpha  | Significance threshold (default 0.05). Bars representing p-values < alpha will be highlighted. |
| title  | Optional title for the plot.                                                                   |

### Value

A ggplot2 object

### Examples

```

data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts) >= 10] >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail1_KO"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail1_KO vs WT)

```

```

dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
PlotPValHist(obj, assay = "KO_vs_WT")

```

---

PlotScree

*PlotScree*


---

## Description

Generate a scree plot

## Usage

```
PlotScree(object)
```

## Arguments

**object**                      A geneda object containing PCA results in the DimReduction slot.

## Value

A ggplot2 object

## Examples

```

data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts) >= 10] >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- FindVariableFeatures(obj, nfeatures = 1000)
obj <- RunPCA(obj)
PlotScree(obj)

```

PlotVolcano

*PlotVolcano***Description**

Plot volcano plot from a geneda object. Requires that you have ran SetDEGs on your object following differential expression in DESeq2.

**Usage**

```
PlotVolcano(object, assay, alpha, l2fc, den = NULL, num = NULL, title = NULL)
```

**Arguments**

|        |                                                                                                    |
|--------|----------------------------------------------------------------------------------------------------|
| object | A geneda object containing DEGs from SetDEGs method                                                |
| assay  | The DEG slot to use for visualization                                                              |
| alpha  | Threshold for adjusted p-values (padj column from DESeq2)                                          |
| l2fc   | Absolute value log2Fold-change magnitude threshold (log2FoldChange column)                         |
| den    | Denominator (reference level for comparison, num vs. den). If NULL, the legend uses "denominator". |
| num    | Numerator (numerator level for comparison, num vs. den). If NULL, the legend uses "numerator".     |
| title  | Optional character vector of what plot should be titled.                                           |

**Details**

Thresholds are inclusive ( $\text{padj} \leq \alpha$ ,  $|\log_2\text{FoldChange}| \geq \text{l2fc}$ ), matching FilterDEGs(). Genes with a missing log2FoldChange or padj are not plotted.

**Value**

A ggplot2 object

**Examples**

```
data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail_KO"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail_KO vs WT)
dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
```

```

res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
PlotVolcano(obj,
  assay = "KO_vs_WT", alpha = 0.05, l2fc = 1,
  den = "WT", num = "Snail_KO"
)

```

RunPCA

*Run PCA and store in DimReduction***Description**

Uses generatePCs() under the hood. If HVGs are empty, selects HVGs via FindVariableFeatures(object, nfeatures) with default nfeatures = 2000. If HVGs are present, the PCA uses NFEATURES = length(HVGs(object)). This aligns the PCA feature count with the HVG selection while allowing an explicit override of nfeatures when empty.

**Usage**

```
RunPCA(object, nfeatures = 2000)
```

**Arguments**

|           |                                                               |
|-----------|---------------------------------------------------------------|
| object    | A geneda object                                               |
| nfeatures | Number of features to use when HVGs are empty. Default = 2000 |

**Value**

Updated geneda object with DimReduction filled

**Examples**

```

data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- RunPCA(obj, nfeatures = 1000)
DimReduction(obj)$percent_var

```

---

|         |                                          |
|---------|------------------------------------------|
| SetDEGs | <i>Set unfiltered DEGs on the object</i> |
|---------|------------------------------------------|

---

## Description

Set unfiltered DEGs on the object

## Usage

```
SetDEGs(object, deg_table, assay)
```

## Arguments

|           |                                                                                         |
|-----------|-----------------------------------------------------------------------------------------|
| object    | A geneda object                                                                         |
| deg_table | A data.frame of DESeq2-like results containing at least columns log2FoldChange and padj |
| assay     | What to name the DEG slot                                                               |

## Value

Updated geneda object with DEGs[[assay]] set

## Examples

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail1_K0"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail1_K0 vs WT)
dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
obj
```

---

|                     |                        |
|---------------------|------------------------|
| show, geneda-method | Show method for geneda |
|---------------------|------------------------|

---

**Description**

Prints a short summary of a geneda object: the number of features, samples, and HVGs, whether PCA results and counts are present, and the names of stored DEG tables.

**Usage**

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

**Arguments**

object                    A geneda object

**Value**

Invisibly returns object; called for its printed summary.

**Examples**

```
data(Snail1K0)
counts <- as.matrix(Snail1K0$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1K0$metadata
rownames(meta) <- meta$Sample
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)
obj <- RunPCA(obj, nfeatures = 1000)
show(obj)
```

---

|          |                  |
|----------|------------------|
| Snail1K0 | Snail1K0 dataset |
|----------|------------------|

---

**Description**

Dataset from Maturi, et al. 2018, (ENA:ERP019920, E-MTAB-5244)

**Usage**

```
data(Snail1K0)
```

**Format**

A list containing raw counts and metadata

**rawCounts** A dataframe of raw counts expression matrix

**metadata** A dataframe of associated metadata

**Source**

Maturi et al., 2018; ENA: ERP019920; E-MTAB-5244

**Examples**

```
data(Snail1KO)
dim(Snail1KO$rawCounts)
Snail1KO$metadata
```

---

|               |                                              |
|---------------|----------------------------------------------|
| SummarizeDEGs | <i>Summarize DEGs Across Selected Assays</i> |
|---------------|----------------------------------------------|

---

**Description**

Summarizes the number of DEGs in each assay of a geneda object according to thresholds for adjusted p-value and log2 fold-change, separately reporting upregulated and downregulated genes. Thresholds are inclusive ( $\text{padj} \leq \alpha$ ,  $|\log_2\text{FoldChange}| \geq \text{lfc}$ ), matching `FilterDEGs()`. Genes with a missing `log2FoldChange` or `padj` are not counted; missing values in other columns (e.g., gene annotations) are ignored.

**Usage**

```
SummarizeDEGs(object, alpha = 0.05, lfc1 = 1, lfc2 = 2, assays = NULL)
```

**Arguments**

|                     |                                                                                                             |
|---------------------|-------------------------------------------------------------------------------------------------------------|
| <code>object</code> | A geneda object containing DEGs in its DEGs slot.                                                           |
| <code>alpha</code>  | Numeric; adjusted p-value threshold. Default is 0.05.                                                       |
| <code>lfc1</code>   | Numeric; lower log2 fold-change threshold. Default is 1.                                                    |
| <code>lfc2</code>   | Numeric; higher log2 fold-change threshold. Default is 2.                                                   |
| <code>assays</code> | Optional character vector of assay names to summarize. Defaults to all assays in <code>object@DEGs</code> . |

**Value**

A data.frame of DEG counts by assay and criteria.

**Examples**

```
data(Snail1KO)
counts <- as.matrix(Snail1KO$rawCounts)
counts <- counts[rowSums(counts >= 10) >= 3, ]
meta <- Snail1KO$metadata
rownames(meta) <- meta$Sample
meta$Genotype <- factor(meta$Genotype, levels = c("WT", "Snail_KO"))
norm <- log2(t(t(counts) / colSums(counts)) * 1e6 + 1)
obj <- GenEDA(normalized = norm, metadata = meta, counts = counts)

# Differential expression with DESeq2 (Snail_KO vs WT)
dds <- DESeq2::DESeqDataSetFromMatrix(counts, meta, design = ~Genotype)
dds <- DESeq2::DESeq(dds, quiet = TRUE)
res <- as.data.frame(DESeq2::results(dds))
obj <- SetDEGs(obj, res, assay = "KO_vs_WT")
SummarizeDEGs(obj, alpha = 0.05, lfc1 = 1, lfc2 = 2)
```

# Index

## \* datasets

Snail1KO, [31](#)

as.geneda, [3](#)

as.geneda, SummarizedExperiment-method  
(as.geneda), [3](#)

DEDashboard, [4](#)

DEGs, [6](#)

DimReduction, [7](#)

extractEigen, [7](#)

ExtractPCA, [8](#), [25](#)

FilterDEGs, [9](#)

FindHVDEGs, [10](#)

FindVariableFeatures, [11](#)

GenEDA, [12](#)

GenEDA(), [4](#)

geneda-class, [13](#)

generatePCs, [13](#)

GenSave, [14](#)

getCounts, [15](#)

getMetadata, [16](#)

getNormalized, [16](#)

HVGs, [17](#)

PlotCountDist, [18](#)

PlotDistances, [18](#)

PlotEigenCorr, [20](#)

PlotEigenHeatmap, [21](#)

PlotHVGVariance, [22](#)

PlotMA, [23](#)

PlotOrdCorr, [24](#)

PlotPCA, [25](#)

PlotPValHist, [26](#)

PlotScree, [27](#)

PlotVolcano, [28](#)

RunPCA, [29](#)

SetDEGs, [30](#)

show, geneda-method, [31](#)

Snail1KO, [31](#)

SummarizeDEGs, [32](#)