

# Package ‘countland’

February 1, 2024

**Title** Analysis of Biological Count Data, Especially from Single-Cell RNA-Seq

**Version** 0.1.2

**Description** A set of functions for applying a restricted linear algebra to the analysis of count-based data. See the accompanying preprint manuscript: “Normalizing need not be the norm: count-based math for analyzing single-cell data” Church et al (2022) <[doi:10.1101/2022.06.01.494334](https://doi.org/10.1101/2022.06.01.494334)> This tool is specifically designed to analyze count matrices from single cell RNA sequencing assays. The tools implement several count-based approaches for standard steps in single-cell RNA-seq analysis, including scoring genes and cells, comparing cells and clustering, calculating differential gene expression, and several methods for rank reduction. There are many opportunities for further optimization that may prove useful in the analysis of other data. We provide the source code freely available at <<https://github.com/shchurch/countland>> and encourage users and developers to fork the code for their own purposes.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.0

**URL** <https://github.com/shchurch/countland>

**BugReports** <https://github.com/shchurch/countland/issues>

**Imports** methods, rlang, Matrix, ggplot2

**Suggests** tidyverse, viridis, gridExtra, igraph, RSpectra, matrixTests, rdist, stats, Seurat, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Church Samuel H. [aut, cre] (<<https://orcid.org/0000-0002-8451-103X>>)

**Maintainer** Church Samuel H. <samuelhchurch@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-02-01 18:00:02 UTC

## R topics documented:

Center . . . . .	3
Cluster . . . . .	3
CountIndex . . . . .	4
countland . . . . .	4
countland-class . . . . .	5
Dot . . . . .	6
Embed . . . . .	7
IMA . . . . .	7
IMA_Compute_Init_Scaled . . . . .	8
IMA_init . . . . .	8
IMA_params . . . . .	9
IMA_Update_Factor . . . . .	9
listCols . . . . .	10
Log . . . . .	10
Normalize . . . . .	11
PlotEigengap . . . . .	11
PlotEmbedding . . . . .	12
PlotGeneCounts . . . . .	12
PlotIMA . . . . .	13
PlotIMAEIbow . . . . .	13
PlotMarker . . . . .	14
PlotSharedCounts . . . . .	14
PrintGeneNumber . . . . .	15
RankMarkerGenes . . . . .	15
RemoveEmpty . . . . .	16
RescaleVariance . . . . .	16
RestoreCounts . . . . .	17
RunIMA . . . . .	17
ScikitManifoldSpectralEmbedding . . . . .	18
ScoreCells . . . . .	19
ScoreGenes . . . . .	19
SharedCounts . . . . .	20
Subsample . . . . .	21
SubsampleCol . . . . .	21
SubsetCells . . . . .	22
SubsetGenes . . . . .	22

**Index**

**24**

---

Center	<i>Recapitulate Seurat centering scaled and transformed data</i>
--------	--

---

**Description**

Recapitulate Seurat centering scaled and transformed data

**Usage**

```
Center(C)
```

**Arguments**

C                    countland object

**Value**

countland object with slots centered\_counts

---

Cluster	<i>Perform spectral clustering on dot products.</i>
---------	---

---

**Description**

Perform spectral clustering on dot products.

**Usage**

```
Cluster(C, n_clusters, n_components = NULL)
```

**Arguments**

C                    countland object  
n\_clusters          number of clusters, integer  
n\_components        number of components from spectral embedding to use (default NULL, will be set to n\_clusters), integer

**Value**

countland object with slot cluster\_labels

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- Dot(C)
C <- Embed(C, n_components=5)
C <- Cluster(C, n_clusters=3)
```

---

CountIndex	<i>Internal function for calculating count index.</i>
------------	---

---

**Description**

Internal function for calculating count index.

**Usage**

```
CountIndex(lm)
```

**Arguments**

lm	column vector
----	---------------

**Value**

count index = largest n where n cells have  $\geq n$  counts

---

countland	<i>Initialize a countland object from a dgCMatix</i>
-----------	--

---

**Description**

Initialize a countland object from a dgCMatix

**Usage**

```
countland(m, remove_empty = TRUE, verbose = TRUE)
```

**Arguments**

m	A matrix of counts (dense or sparse)
remove_empty	filter out cells and genes with no observed counts (default=TRUE)
verbose	show stderr message statements (default=TRUE)

**Value**

countland object

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
```

---

countland-class	<i>An S4 class to represent a countland object</i>
-----------------	--

---

**Description**

An S4 class to represent a countland object

**Slots**

**counts** A dgCMatrx with rows as cells, columns as genes.

**names\_genes** A character vector of column names.

**names\_cells** A character vector of row names.

**raw\_counts** The count dgCMatrx as originally loaded.

**raw\_names\_genes** The gene name character vector as originally loaded.

**raw\_names\_cells** The cell name character vector as originally loaded.

**subsample** A dgCMatrx with row sums equal.

**cell\_scores** A data.frame of cell count measures.

**gene\_scores** A data.frame of gene expression measures.

**dots** A similarity dgCMatrx of dot products.

**eigenvals** An vector of eigenvalues from spectral embedding

**embedding** An array of two columns (spectral embeddings).

**cluster\_labels** A numeric vector of cluster assignments of length n cells.

**marker\_full** A list of data.frames with genes ranked for each cluster.

**marker\_genes** A data.frame of top ten marker genes per cluster.

**matrixU** A dgCMatrx of dimensions cells x features.

**matrixV** A dgCMatrx of dimensions genes x features.

**matrixLambda** A diagonal dgCMatrx of scaling factors.

**sharedcounts** A similarity dgCMatrx of shared counts between genes.

**sum\_sharedcounts** A dgCMatrx with counts summed within gene clusters.

**sum\_sharedcounts\_all** A dgCMatrx with counts summed and including all genes not present in any cluster.

norm\_factor A numeric vector of cell normalization factors.  
norm\_counts A dgCMatrix of normalized counts.  
log\_counts A dgCMatrix of log transformed counts.  
scaled\_counts A dgCMatrix of counts scaled by gene unit variance.  
centered\_counts A dgCMatrix of counts centered at zero.  
verbose A T/F object for suppressing messages

---

Dot *Calculate pairwise dot products of counts between all cells.*

---

### Description

Calculate pairwise dot products of counts between all cells.

### Usage

```
Dot(C, subsample = FALSE)
```

### Arguments

C countland object  
subsample if TRUE, use subsampled counts, otherwise use counts (default=FALSE)

### Value

countland object with slot dots

### Examples

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- Dot(C)
```

---

Embed *Perform spectral embedding on dot products.*

---

**Description**

Perform spectral embedding on dot products.

**Usage**

```
Embed(C, n_components = 10)
```

**Arguments**

C                    countland object  
n\_components        number of components, integer (default=10)

**Value**

countland object with slot embedding, eigenvals

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)  
gold.data <- Seurat::Read10X(data.dir = gold_path)  
C <- countland(gold.data)  
C <- Dot(C)  
C <- Embed(C, n_components=5)
```

---

IMA *run integer matrix approximation*

---

**Description**

run integer matrix approximation

**Usage**

```
IMA(X, params)
```

**Arguments**

X                    observed data matrix  
params                parameter object

**Value**

U, V, and Lambda matrix factors

---

IMA\_Compute\_Init\_Scaled  
*rescale if max val is above upper bound*

---

**Description**

rescale if max val is above upper bound

**Usage**

```
IMA_Compute_Init_Scaled(h, l_bound, u_bound)
```

**Arguments**

h	matrix to be rescaled
l_bound	lower bound
u_bound	upper bound

**Value**

rescaled matrix

---

IMA\_init                    *function to initialize U, V, and Lambda*

---

**Description**

function to initialize U, V, and Lambda

**Usage**

```
IMA_init(X, params)
```

**Arguments**

X	observed data matrix
params	parameter object

**Value**

initialized U, V, and Lambda matrices



---

IMA_params	<i>Parameter class for IMA</i>
------------	--------------------------------

---

**Description**

Parameter class for IMA

**Usage**

```
IMA_params(
    rank,
    u_bounds,
    l_bounds = c(0, 0),
    maxiter = 1e+06,
    stop_crit = 1e-04
)
```

**Arguments**

rank	target number of features in final matrices
u_bounds	upper bounds on integers
l_bounds	lower bounds on integers (default = c(0,0))
maxiter	maximum number of iterations (default = 1000000)
stop_crit	criterion of difference at which to stop (default = 0.0001)

**Value**

parameter object

---

IMA_Update_Factor	<i>Update factor matrix - see SUSTain code</i>
-------------------	--

---

**Description**

Update factor matrix - see SUSTain code

**Usage**

```
IMA_Update_Factor(M, coeff, mkrp, mode, lambda_, params)
```

**Arguments**

M	matrix to be updated (either U or V)
coeff	matrix used in updating algorithm
mkrp	matrix used in updating algorithm
mode	whether update U or V
lambda_	scaling matrix
params	parameter object

**Value**

updated matrix and scaling factors

---

listCols	<i>Split dgCMatrix into column vectors.</i>
----------	---

---

**Description**

Split dgCMatrix into column vectors.

**Usage**

```
listCols(m)
```

**Arguments**

m	dgCMatrix
---	-----------

**Value**

list of column vectors, numeric

---

Log	<i>Recapitulate Seurat log transformation</i>
-----	---

---

**Description**

Recapitulate Seurat log transformation

**Usage**

```
Log(C)
```

**Arguments**

C	countland object
---	------------------

**Value**

countland object with slots log\_counts

---

Normalize	<i>Recapitulate Seurat normalization</i>
-----------	--

---

**Description**

Recapitulate Seurat normalization

**Usage**

```
Normalize(C)
```

**Arguments**

C                    countland object

**Value**

countland object with slots norm\_factor, norm\_counts

---

PlotEigengap	<i>Plots eigenvalues to investigate the optimal number of clusters</i>
--------------	--

---

**Description**

Plots eigenvalues to investigate the optimal number of clusters

**Usage**

```
PlotEigengap(C)
```

**Arguments**

C                    countland object

**Value**

generates plot of eigenvalues by number of components

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- Dot(C)
C <- Embed(C, n_components=5)
PlotEigengap(C)
```

---

PlotEmbedding	<i>Plot cells using spectral embedding of dot products.</i>
---------------	---

---

**Description**

Plot cells using spectral embedding of dot products.

**Usage**

```
PlotEmbedding(C, colors = color_palette)
```

**Arguments**

C	countland object
colors	color palette for ggplot2, default=palette of 11 colors

**Value**

generates plot of cells in two spectral embedding dimensions

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- Dot(C)
C <- Embed(C,n_components=5)
C <- Cluster(C,n_clusters=3)
PlotEmbedding(C)
```

---

PlotGeneCounts	<i>Generate a strip plot for counts across selected genes</i>
----------------	---

---

**Description**

Generate a strip plot for counts across selected genes

**Usage**

```
PlotGeneCounts(C, gene_indices, colors = color_palette)
```

**Arguments**

C	countland object
gene_indices	vector of gene index values
colors	color palette for ggplot2, default=palette of 11 colors

**Value**

generates plot of gene count distributions

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
PlotGeneCounts(C, gene_indices=1:10)
```

---

 PlotIMA

*Plot cells using integer matrix approximation*


---

**Description**

Plot cells using integer matrix approximation

**Usage**

```
PlotIMA(C, x = 1, y = 2, colors = color_palette, subsample = TRUE)
```

**Arguments**

C	countland object
x	feature on x-axis, integer (default=1)
y	feature on y-axis, integer (default=2)
colors	color palette for ggplot2, default=palette of 11 colors
subsample	if TRUE, use subsampled counts (default), otherwise use counts

**Value**

generates plot of cells using integer matrix approximation

---

 PlotIMAEIbow

*Plot the difference between the observed and reconstructed count matrix using integer matrix approximation and a series of total features.*


---

**Description**

Plot the difference between the observed and reconstructed count matrix using integer matrix approximation and a series of total features.

**Usage**

```
PlotIMAEIbow(C, max_features, u_bounds, subsample = TRUE)
```

**Arguments**

C	countland object
max_features	maximum number of features to assess, integer
u_bounds	upper bounds for U and V matrices, vector of length 2
subsample	if TRUE, use subsampled counts (default), otherwise use counts

**Value**

generates elbow plot for the difference between observed and reconstructed matrices as number of features increases

---

PlotMarker	<i>Plot cell using spectral embedding and display counts in a given gene.</i>
------------	---

---

**Description**

Plot cell using spectral embedding and display counts in a given gene.

**Usage**

```
PlotMarker(C, gene_index, colors = color_palette)
```

**Arguments**

C	countland object
gene_index	index value for gene to visualize
colors	color palette for ggplot2, default=palette of 11 colors

**Value**

generates plot of cells with spectral embedding, colored by marker gene counts

---

PlotSharedCounts	<i>Plot cells using matrix of counts summed by clusters of genes.</i>
------------------	---

---

**Description**

Plot cells using matrix of counts summed by clusters of genes.

**Usage**

```
PlotSharedCounts(C, x = 1, y = 2, colors = color_palette)
```

**Arguments**

C	countland object
x	gene cluster to plot on x-axis, integer (default=1)
y	gene cluster to plot on y-axis, integer (default=2)
colors	color palette for ggplot2, default=palette of 11 colors

**Value**

generates plot of cells using shared counts

---

PrintGeneNumber	<i>Restore count matrix to original state</i>
-----------------	---

---

**Description**

Restore count matrix to original state

**Usage**

PrintGeneNumber(C)

**Arguments**

C	countland object
---	------------------

**Value**

countland object

---

RankMarkerGenes	<i>Rank the top marker genes for each cluster from spectral clustering.</i>
-----------------	---

---

**Description**

Rank the top marker genes for each cluster from spectral clustering.

**Usage**

RankMarkerGenes(C, method = "prop-zero", subsample = FALSE)

**Arguments**

C	countland object
method	prop-zero to rank by proportion of cells that are non-zero (default), or rank-sums to rank using Wilcoxon rank-sums test
subsample	if TRUE, use subsampled counts, otherwise use counts (default=FALSE)

**Value**

countland object with slots marker\_genes and marker\_full

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- Dot(C)
C <- Embed(C,n_components=5)
C <- Cluster(C,n_clusters=3)
C <- RankMarkerGenes(C,method='prop-zero',subsample=FALSE)
```

---

RemoveEmpty

*Internal function to remove empty columns and rows*

---

**Description**

Internal function to remove empty columns and rows

**Usage**

RemoveEmpty(C)

**Arguments**

C                    countland object

**Value**

countland object, count matrix updated

---

RescaleVariance

*Recapitulate Seurat scaling to unit variance*

---

**Description**

Recapitulate Seurat scaling to unit variance

**Usage**

RescaleVariance(C)

**Arguments**

C                    countland object



**Value**

countland object with slots scaled\_counts

---

RestoreCounts	<i>Restore count matrix to original state</i>
---------------	---

---

**Description**

Restore count matrix to original state

**Usage**

```
RestoreCounts(C)
```

**Arguments**

C                    countland object

**Value**

countland object

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- SubsetGenes(C, gene_indices=1:200)
C <- SubsetCells(C, cell_indices=1:50)
C <- RestoreCounts(C)
```

---

RunIMA	<i>Perform integer matrix approximation on count matrix.</i>
--------	--

---

**Description**

Perform integer matrix approximation on count matrix.

**Usage**

```
RunIMA(
  C,
  features,
  u_bounds,
  l_bounds = c(0, 0),
  maxiter = 1e+06,
  stop_crit = 1e-04,
  subsample = TRUE
)
```

**Arguments**

C	countland object
features	target number of features, integer
u_bounds	upper bounds for U and V matrices, vector of length 2
l_bounds	lower bounds for U and V matrices, vector of length 2 (default=c(0,0))
maxiter	maximum number of iterations, integer (default=1000000)
stop_crit	criterion for stopping based on difference between iterations, numeric (default=0.0001)
subsample	if TRUE, use subsampled counts (default), otherwise use counts

**Value**

countland object with slots matrixU, matrixV, matrixLambda

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- RunIMA(C, features=10, u_bounds=c(10, 10), subsample=FALSE)
```

---

ScikitManifoldSpectralEmbedding

*Recapitulate scikit.manifold.spectral\_embedding from python.*

---

**Description**

Recapitulate scikit.manifold.spectral\_embedding from python.

**Usage**

```
ScikitManifoldSpectralEmbedding(A, n_components)
```

**Arguments**

A	similarity matrix, dgCMatrix
n_components	number of eigenvectors to retain, integer

**Value**

matrix of eigenvectors

---

ScoreCells	<i>Calculate several scores for counts across cells</i>
------------	---

---

**Description**

Calculate several scores for counts across cells

**Usage**

```
ScoreCells(C, gene_string = NULL)
```

**Arguments**

C	countland object
gene_string	string with regular expression expression matching gene names of interest (default=NULL)

**Value**

countland object with slot cell\_scores

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- ScoreCells(C, gene_string="*149932$")
```

---

ScoreGenes	<i>Calculate several scores for count-based gene expression.</i>
------------	--

---

**Description**

Calculate several scores for count-based gene expression.

**Usage**

```
ScoreGenes(C, subsample = FALSE)
```

**Arguments**

C	countland object
subsample	if TRUE, use subsampled counts, otherwise use counts (default=FALSE)

**Value**

countland object with slot gene\_scores

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- ScoreGenes(C)
```

---

SharedCounts	<i>Combine groups of genes with similar counts by clustering and summing.</i>
--------------	---

---

**Description**

Combine groups of genes with similar counts by clustering and summing.

**Usage**

```
SharedCounts(C, n_clusters, n_cells = 100, subsample = TRUE)
```

**Arguments**

C	countland object
n_clusters	number of clusters
n_cells	number of cells to sample for gene clustering
subsample	if TRUE, use subsampled counts (default), otherwise use counts

**Value**

countland object with slots shared\_counts, sum\_sharedcounts, sum\_sharedcounts\_all

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- SharedCounts(C, n_clusters=10, subsample=FALSE)
```

---

Subsample	<i>Subsample cells to a standard number of counts by randomly sampling observations without replacement.</i>
-----------	--

---

**Description**

Subsample cells to a standard number of counts by randomly sampling observations without replacement.

**Usage**

```
Subsample(C, gene_counts = NA, cell_counts = NA)
```

**Arguments**

C	countland object
gene_counts	maximum total counts for genes
cell_counts	sequencing depth for all cells, or "min" to use the minimum cell total

**Value**

countland object with slot subsample

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- Subsample(C, gene_counts=250, cell_counts=100)
```

---

SubsampleCol	<i>Internal function for subsampling a column from a sparse matrix.</i>
--------------	---

---

**Description**

Internal function for subsampling a column from a sparse matrix.

**Usage**

```
SubsampleCol(lm, li, j, n_counts)
```

**Arguments**

lm	column vector
li	row positions
j	column index
n_counts	count to sample

**Value**

subsampled column as dgTMatrix components

---

SubsetCells                      *Subsets cells using a vector of cell indices*

---

**Description**

Subsets cells using a vector of cell indices

**Usage**

```
SubsetCells(C, cell_indices, remove_empty = TRUE)
```

**Arguments**

C                      countland object  
 cell\_indices        vector of cell index values  
 remove\_empty       filter out cells and genes with no observed counts (default=TRUE)

**Value**

countland object, count matrix updated

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- SubsetCells(C, cell_indices=1:50)
```

---

SubsetGenes                      *Subsets genes using a vector of gene indices*

---

**Description**

Subsets genes using a vector of gene indices

**Usage**

```
SubsetGenes(C, gene_indices, remove_empty = TRUE)
```

**Arguments**

<code>C</code>	countland object
<code>gene_indices</code>	vector of gene index values
<code>remove_empty</code>	filter out cells and genes with no observed counts (default=TRUE)

**Value**

countland object, count matrix updated

**Examples**

```
gold_path <- system.file("testdata", package = "countland", mustWork = TRUE)
gold.data <- Seurat::Read10X(data.dir = gold_path)
C <- countland(gold.data)
C <- SubsetGenes(C, gene_indices=1:200)
```

# Index

Center, [3](#)  
Cluster, [3](#)  
CountIndex, [4](#)  
countland, [4](#)  
countland-class, [5](#)

Dot, [6](#)

Embed, [7](#)

IMA, [7](#)  
IMA\_Compute\_Init\_Scaled, [8](#)  
IMA\_init, [8](#)  
IMA\_params, [9](#)  
IMA\_Update\_Factor, [9](#)

listCols, [10](#)  
Log, [10](#)

Normalize, [11](#)

PlotEigengap, [11](#)  
PlotEmbedding, [12](#)  
PlotGeneCounts, [12](#)  
PlotIMA, [13](#)  
PlotIMAElbow, [13](#)  
PlotMarker, [14](#)  
PlotSharedCounts, [14](#)  
PrintGeneNumber, [15](#)

RankMarkerGenes, [15](#)  
RemoveEmpty, [16](#)  
RescaleVariance, [16](#)  
RestoreCounts, [17](#)  
RunIMA, [17](#)

ScikitManifoldSpectralEmbedding, [18](#)  
ScoreCells, [19](#)  
ScoreGenes, [19](#)  
SharedCounts, [20](#)  
Subsample, [21](#)  
SubsampleCol, [21](#)  
SubsetCells, [22](#)  
SubsetGenes, [22](#)