# Package ‘ganGenerativeData’

October 13, 2022

**Type** Package

**Title** Generate Generative Data for a Data Source

**Version** 1.3.3

**Date** 2022-02-16

**Author** Werner Mueller

**Maintainer** Werner Mueller <werner.mueller5@chello.at>

**Description**

Generative Adversarial Networks are applied to generate generative data for a data source. In iterative training steps the distribution of generated data converges to that of the data source. Direct applications of generative data are the created functions for outlier detection and missing data completion. Reference: Goodfellow et al. (2014) <arXiv:1406.2661v1>.

**License** GPL (>= 2)

**Imports** Rcpp (>= 1.0.3), tensorflow (>= 2.0.0)

**LinkingTo** Rcpp

**RoxygenNote** 7.0.2

**SystemRequirements** TensorFlow (https://www.tensorflow.org)

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-02-16 13:40:16 UTC

## R topics documented:

ganGenerativeData-package . . . 2
dsActivateColumns . . . 8
dsCreateWithDataFrame . . . 8
dsDeactivateColumns . . . 9
dsGetActiveColumnNames . . . 10
dsGetInactiveColumnNames . . . 10
dsGetNumberOfRows . . . 11
dsGetRow . . . 11
dsRead . . . 12

dsWrite . . . . . . . . . . 12
gdCalculateDensityValue . . . . . . . . . . 13
gdCalculateDensityValueQuantile . . . . . . . . . . 13
gdCalculateDensityValues . . . . . . . . . . 14
gdComplete . . . . . . . . . . 15
gdGenerate . . . . . . . . . . 15
gdGetNumberOfRows . . . . . . . . . . 16
gdGetRow . . . . . . . . . . 17
gdKNearestNeighbors . . . . . . . . . . 17
gdPlot2dProjection . . . . . . . . . . 18
gdPlotDataSourceParameters . . . . . . . . . . 19
gdPlotParameters . . . . . . . . . . 19
gdRead . . . . . . . . . . 20

**Index** **21**

---

`ganGenerativeData-package`

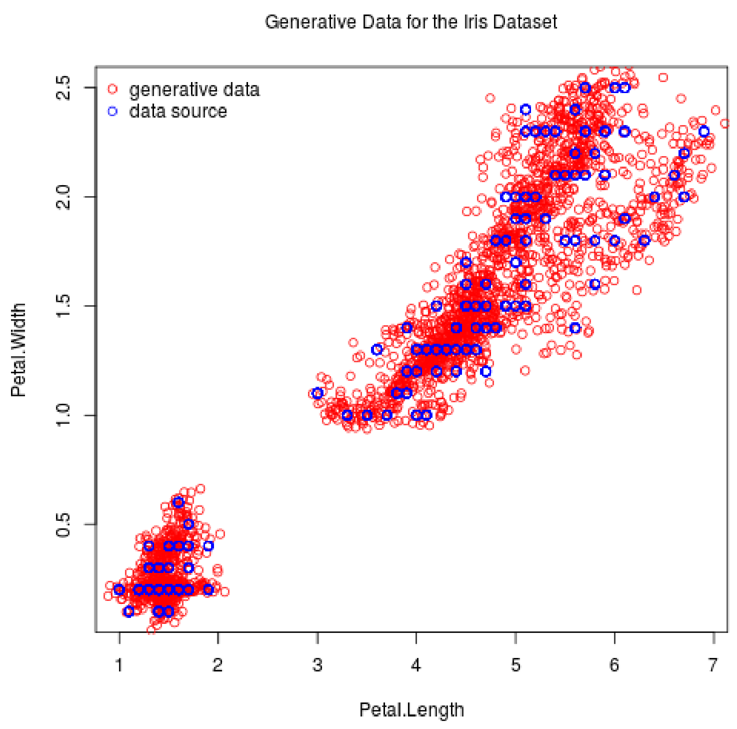
*Generate generative data for a data source*

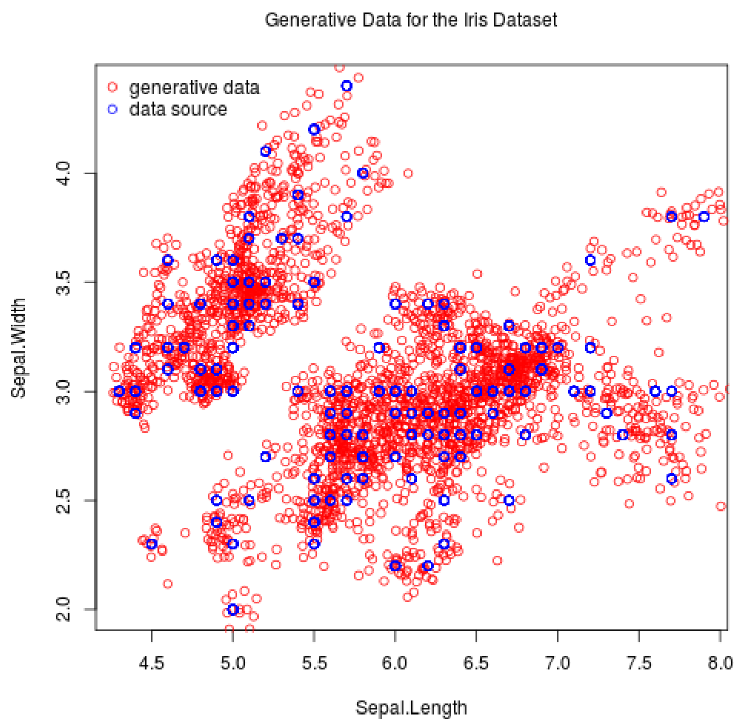
---

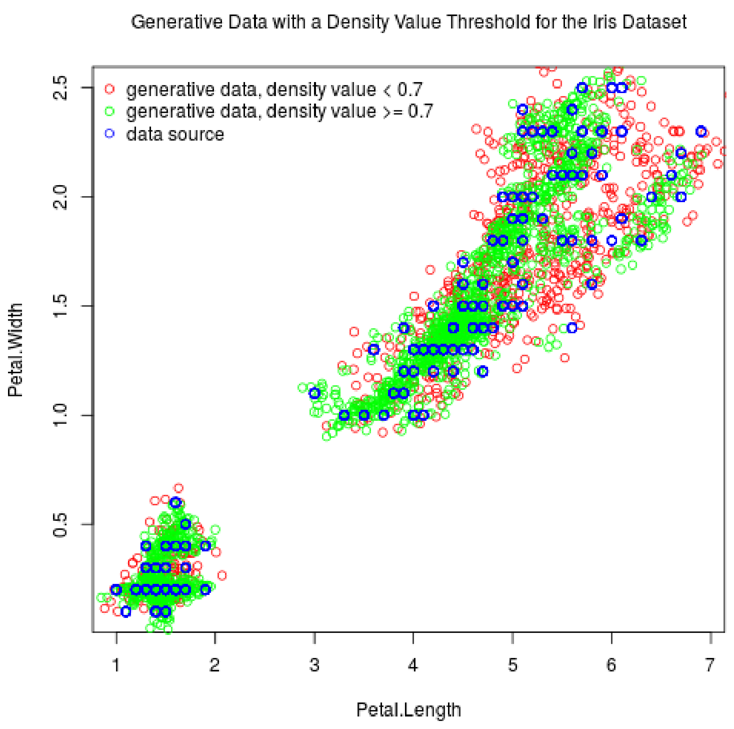
## Description

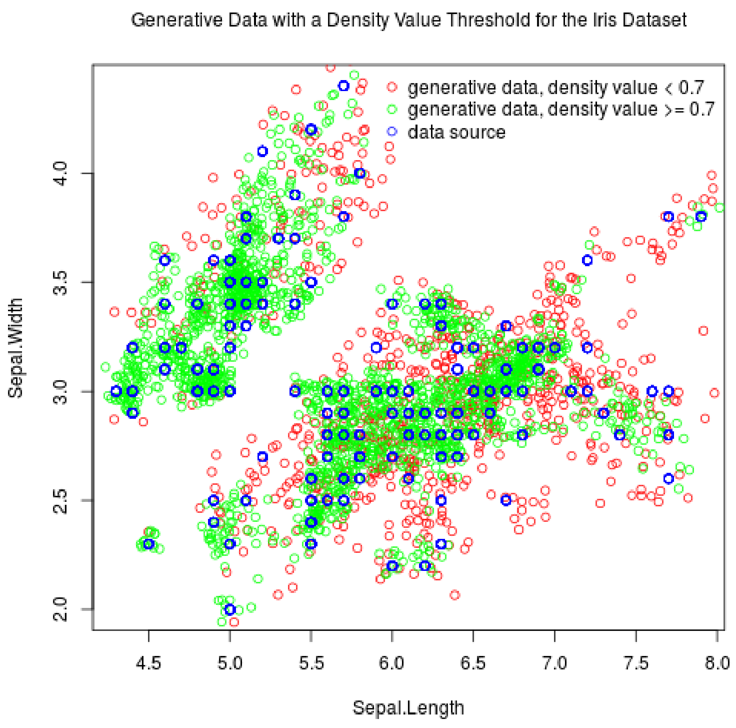
Generative Adversarial Networks are applied to generate generative data for a data source. In iterative training steps the distribution of generated data converges to that of the data source. Direct applications of generative data are the created functions for outlier detection and missing data completion.

The inserted images show two-dimensional projections of generative data for the iris dataset:


Generative Data for the Iris Dataset
○ generative data
○ data source
Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7


Generative Data for the Iris Dataset
○ generative data
○ data source
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Generative Data with a Density Value Threshold for the Iris Dataset
generative data, density value < 0.7
generative data, density value >= 0.7
data source
Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7


Generative Data with a Density Value Threshold for the Iris Dataset
generative data, density value < 0.7
generative data, density value >= 0.7
data source
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

## Details

The API includes functions for topics "definition of data source" and "generation of generative data". Main function of first topic is `dsCreateWithDataFrame()` which creates a data source with passed data frame. Main function of second topic is `gdGenerate()` which generates generative data for a data source.

### 1. Definition of data source

`dsCreateWithDataFrame()` Create a data source with passed data frame.

`dsActivateColumns()` Activate columns of a data source in order to include them in generation of generative data. By default columns are active.

`dsDeactivateColumns()` Deactivate columns of a data source in order to exclude them in generation of generative data. Note that in this version only columns with values of type double or float can be used in generation of generative data. All columns with values of other type have to be deactivated.

`dsGetActiveColumnNames()` Get names of active columns of a data source.

`dsGetInactiveColumnNames()` Get names of inactive columns of a data source.

`dsWrite()` Write created data source including settings of active columns to a file in binary format. This file will be used as input in functions of topic "generation of generative data".

`dsRead()` Read a data source from a file that was written with `dsWrite()`.

`dsGetNumberOfRows()` Get number of rows in a data source.

`dsGetRow()` Get a row in a data source.

### 2. Generation of generative data

`gdGenerate()` Read a data source from a file, generate generative data for the data source in iterative training steps and write generated data to a file in binary format.

`gdCalculateDensityValues()` Read generative data from a file, calculate density values and write generative data with density values to original file.

`gdRead()` Read generative data and data source from specified files.

`gdPlotParameters()` Specify plot parameters for generative data.

`gdPlotDataSourceParameters()` Specify plot parameters for data source.

`gdPlot2dProjection()` Create an image file containing two-dimensional projections of generative data and data source.

`gdGetNumberOfRows()` Get number of rows in generative data.

`gdGetRow()` Get a row in generative data.

`gdCalculateDensityValue()` Calculate density value for a data record.

`gdCalculateDensityValueQuantile()` Calculate density value quantile for a percent value.

`gdKNearestNeighbors()` Search for k nearest neighbors.

`gdCompletee()` Complete incomplete data record.

## Author(s)

Werner Mueller

Maintainer: Werner Mueller <werner.mueller5@chello.at>

## References

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014), *"Generative Adversarial Nets"*, <arXiv:1406.2661v1>

## Examples

```
# Generate generative data for the iris dataset

# Package ganGenerativeData imports package tensorflow
# which offers an interface to machine learning framework TensorFlow.
# It is used for training of neural networks and must be installed by the user.
## Not run: library(tensorflow)
install_tensorflow()
## End(Not run)

# Load library
library(ganGenerativeData)

# 1. Definition of data source for the iris dataset

# Create a data source with iris data frame.
dsCreateWithDataFrame(iris)

# Deactivate the column with name Species and index 5 in order to exclude it in
# generation of generative data.
```

```
dsDeactivateColumns(c(5))

# Get the active column names: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width.
dsGetActiveColumnNames()

# Write the data source including settings of active columns to file "iris4d.bin" in binary format.
## Not run: dsWrite("iris4d.bin")

# 2. Generation of generative data for the iris data source

# Read data source from file "iris4d.bin",
# generate generative data in iterative training steps and
# write generated generative data to file "gd.bin".
## Not run: gdGenerate("iris4d.bin", "gd.bin", 2500, 0.95, c(1, 2))

# Read generative data from file "gd.bin", calculate density values and
# write generative data with density values to original file.
## Not run: gdCalculateDensityValues("gd.bin")

# Read generative data from file "gd.bin" and data source from "iris4d.bin"
## Not run: gdRead("gd.bin", "iris4d.bin")

# Create an image showing two-dimensional projections of generative data and
# data source for column indices 3, 4 and write it to file "gd34d.png"
## Not run: gdPlot2dProjection("gd34d.png",
"Generative Data for the Iris Dataset", c(3, 4),
gdPlotParameters(250000),
gdPlotDataSourceParameters(2500))
## End(Not run)

# Create an image showing two-dimensional projections of generative data and data source for
# column indices 3, 4 with density value threshold 0.71 and write it to file "gd34ddv.png"
## Not run: gdPlot2dProjection("gd34ddv.png",
"Generative Data with a Density Value Threshold for the Iris Dataset", c(3, 4),
gdPlotParameters(250000, c(0.71), c("red", "green")),
gdPlotDataSourceParameters(2500))
## End(Not run)

# Get number of rows in generative data
## Not run: gdGetNumberOfRows()

# Get row with index 1000 in generative data
## Not run: gdGetRow(1000)

# Calculate density value for a data record
## Not run: gdCalculateDensityValue(list(6.1, 2.6, 5.6, 1.4))

# Calculate density value quantile for 50 percent
## Not run: gdCalculateDensityValueQuantile(50)

# Search for k nearest neighbors for a data record
## Not run: gdKNearestNeighbors(list(5.1, 3.5, 1.4, 0.2), 3)
```

```
# Complete incomplete data record containing an NA value
## Not run: gdComplete(list(5.1, 3.5, 1.4, NA))
```

---

## dsActivateColumns *Activate columns*

---

### Description

Activate columns of a data source in order to include them in generation of generative data. By default columns are active.

### Usage

```
dsActivateColumns(columnVector)
```

### Arguments

| | |
|---|---|
| `columnVector` | Vector of column indices |

### Value

None

### Examples

```
dsCreateWithDataFrame(iris)
dsGetActiveColumnNames()
dsDeactivateColumns(c(5))
dsGetActiveColumnNames()
dsActivateColumns(c(5))
dsGetActiveColumnNames()
```

---

## dsCreateWithDataFrame *Create a data source with passed data frame*

---

### Description

Create a data source with passed data frame.

### Usage

```
dsCreateWithDataFrame(dataFrame)
```

### Arguments

| | |
|---|---|
| `dataFrame` | Name of data frame |

### Value

None

### Examples

```
# Create a data source and with built in iris data frame.
dsCreateWithDataFrame(iris)
```

---

## `dsDeactivateColumns` *Deactivate columns*

---

### Description

Deactivate columns of a data source in order to exclude them in generation of generative data. Note that in this version only columns with values of type double or float can be used in generation of generative data. All columns with values of other type have to be deactivated.

### Usage

```
dsDeactivateColumns(columnVector)
```

### Arguments

| | |
|---|---|
| `columnVector` | Vector of column indices |

### Value

None

### Examples

```
dsCreateWithDataFrame(iris)
dsDeactivateColumns(c(5))
dsGetInactiveColumnNames()
```

## dsGetActiveColumnNames

*Get active column names*

### Description

Get active column names of a data source

### Usage

```
dsGetActiveColumnNames()
```

### Value

Vector of names of active columns

### Examples

```
dsCreateWithDataFrame(iris)
dsDeactivateColumns(c(5))
dsGetActiveColumnNames()
```

## dsGetInactiveColumnNames

*Get inactive column names*

### Description

Get inactive column names of a data source

### Usage

```
dsGetInactiveColumnNames()
```

### Value

Vector of names of inactive columns

### Examples

```
dsCreateWithDataFrame(iris)
dsDeactivateColumns(c(5))
dsGetInactiveColumnNames()
```

## `dsGetNumberOfRows` *Get number of rows*

### Description

Get number of rows in a data source

### Usage

```
dsGetNumberOfRows()
```

### Value

Number of rows

### Examples

```
dsCreateWithDataFrame(iris)
dsGetNumberOfRows()
```

## `dsGetRow` *Get a row in a data source*

### Description

Get a row in a data source for a row index.

### Usage

```
dsGetRow(index)
```

### Arguments

| | |
|---|---|
| `index` | Index of row |

### Value

List containing row in data source

### Examples

```
dsCreateWithDataFrame(iris)
dsGetRow(1)
```

## dsRead *Read a data source from file*

### Description

Read a data source from a file in binary format

### Usage

```
dsRead(fileName)
```

### Arguments

| | |
|---|---|
| `fileName` | Name of data source file |

### Value

None

### Examples

```
## Not run: dsCreateWithDataFrame(iris)
dsDeactivateColumns(c(5))
dsWrite("iris4d.bin")
dsRead("iris4d.bin")
## End(Not run)
```

## dsWrite *Write a data source to file*

### Description

Write a data source including settings of active columns to a file in binary format. This file will be used as input in functions for generation of generative data.

### Usage

```
dsWrite(fileName)
```

### Arguments

| | |
|---|---|
| `fileName` | Name of data source file |

### Value

None

## Examples

```
## Not run: dsCreateWithDataFrame(iris)
dsDeactivateColumns(c(5))
dsWrite("iris4d.bin")
## End(Not run)
```

---

`gdCalculateDensityValue` *Calculate density value for a data record*

---

## Description

Calculate density value for a data record. By default for the calculation a linear search is performed on generative data. When a search tree is used search is performed on a tree for generative data which is built once in the first function call.

## Usage

```
gdCalculateDensityValue(dataRecord, useSearchTree = FALSE)
```

## Arguments

| | |
|---|---|
| `dataRecord` | List containing a data record |
| `useSearchTree` | Boolean value indicating if a search tree should be used. |

## Value

Normalized density value number

## Examples

```
## Not run: gdRead("gd.bin")
gdCalculateDensityValue(list(6.1, 2.6, 5.6, 1.4))
## End(Not run)
```

---

`gdCalculateDensityValueQuantile` *Calculate density value quantile*

---

## Description

Calculate density value quantile for a percent value.

## Usage

```
gdCalculateDensityValueQuantile(percent)
```

### Arguments

| | |
|---|---|
| `percent` | Percent value |

### Value

Normalized density value quantile number

### Examples

```
## Not run: gdRead("gd.bin")
gdCalculateDensityValueQuantile(50)
## End(Not run)
```

---

## gdCalculateDensityValues

*Calculate density values for generative data*

---

### Description

Read generative data from a file, calculate density values and write generative data with density values to original file. Calculated density values are used to classiy generative data. In function gdPlotParameters() density value thresholds with assigned colors can be passed to draw generative data for different density value ranges.

### Usage

```
gdCalculateDensityValues(generativeDataFileName)
```

### Arguments

| | |
|---|---|
| `generativeDataFileName` | Name of generative data file name |

### Value

None

### Examples

```
## Not run: gdCalculateDensityValues("gd.bin")
```

## gdComplete — *Complete incomplete data record*

### Description

Search for first nearest neighbor in generative data for incomplete data record containing NA values. Found row in generative data is then used to replace NA values in inccomplete data record. This function calls gdKNearestNeighbor() with parameter k equal to 1.

### Usage

```
gdComplete(dataRecord, useSearchTree = FALSE)
```

### Arguments

| | |
|---|---|
| `dataRecord` | List containing incomplete data record |
| `useSearchTree` | Boolean value indicating if a search tree should be used. |

### Value

List containing completed data record

### Examples

```
## Not run: gdRead("gd.bin")
gdComplete(list(5.1, 3.5, 1.4, NA))
## End(Not run)
```

## gdGenerate — *Generate generative data for a data source*

### Description

Read a data source from a file, generate generative data for the data source in iterative training steps and write generated data to a file in binary format. When a higher number of iterations is used the distribution of generated data gets closer to that of the data source.

### Usage

```
gdGenerate(
  dataSourceFileName,
  generativeDataFileName,
  numberOfIterations,
  keepProbability,
  columnIndices
)
```

### Arguments

| | |
|---|---|
| `dataSourceFileName` | Name of data source file |
| `generativeDataFileName` | Name of generative data file |
| `numberOfIterations` | Number of iterations. In this version the limit of number of iterations is set to 50000. |
| `keepProbability` | Value in the range of 0 to 1 which is used in training of neural networks to train generalized networks. |
| `columnIndices` | Vector of two column indices that are used to plot two-dimensional projections of normalized generated generative data and data source for a training step. Indices refer to indices of active columns of data source. |

### Value

None

### Examples

```
## Not run: gdGenerate("iris4d.bin", "gd.bin", 2500, 0.95, c(1, 2))
```

---

## `gdGetNumberOfRows` *Get number of rows*

---

### Description

Get number of rows in generative data

### Usage

```
gdGetNumberOfRows()
```

### Value

Number of rows

### Examples

```
## Not run: gdRead("gd.bin")
gdGetNumberOfRows()
## End(Not run)
```

## gdGetRow *Get a row in generative data*

### Description

Get a row in generative data for a row index

### Usage

```
gdGetRow(index)
```

### Arguments

| | |
|---|---|
| `index` | Index of row |

### Value

List containing row in generative data

### Examples

```
## Not run: gdRead("gd.bin")
gdGetRow(1000)
## End(Not run)
```

## gdKNearestNeighbors *Search for k nearest neighbors*

### Description

Search for k nearest neighbors in generative data for a data record. When the data record contains NA values only the non-NA values are considered in search. By default a linear search is performed. When a search tree is used search is performed on a tree which is built once in the first function call. Building a tree is also triggered when NA values in data records change in subsequent function calls.

### Usage

```
gdKNearestNeighbors(dataRecord, k = 1L, useSearchTree = FALSE)
```

### Arguments

| | |
|---|---|
| `dataRecord` | List containing a data record |
| `k` | Number of nearest neighbors |
| `useSearchTree` | Boolean value indicating if a search tree should be used. |

## Value

A list of rows in generative data

## Examples

```
## Not run: gdRead("gd.bin")
gdKNearestNeighbors(list(5.1, 3.5, 1.4, 0.2), 3)
## End(Not run)
```

---

## `gdPlot2dProjection` *Create an image file for generative data and data source*

---

## Description

Create an image file containing two-dimensional projections of generative data and data source. Plot parameters for generative data and data source are passed by functions gdPlotParameters() and gdPlotDataSourceParameters(). Data points of data source are drawn above data points of generative data.

## Usage

```
gdPlot2dProjection(
  imageFileName,
  title,
  columnIndices,
  generativeDataParameters = gdPlotParameters(numberOfRandomPoints = 0,
    densityValueThresholds = c(), densityValueColors = c("red")),
 dataSourceParameters = gdPlotDataSourceParameters(numberOfRandomPoints = 0, color =
    "blue")
)
```

## Arguments

| | |
|---|---|
| `imageFileName` | Name of image file |
| `title` | Title of image |
| `columnIndices` | Vector of two column indices that are used for the two-dimensional projections. Indices refer to indices of active columns of data source. |
| `generativeDataParameters` | Plot generative data parameters, see function gdPlotParameters(). |
| `dataSourceParameters` | Plot data source parameters, see function gdPlotDataSourceParameters(). |

## Value

None

### Examples

```
## Not run: gdRead("gd.bin", "iris4d.bin")
gdPlot2dProjection("gd12ddv.png",
 "Generative Data with a Density Value Threshold for the Iris Dataset", c(1, 2),
gdPlotParameters(250000, c(0.71), c("red", "green")),
gdPlotDataSourceParameters(2500))
gdPlot2dProjection("gd34ddv.png",
"Generative Data with a Density Value Threshold for the Iris Dataset", c(3, 4),
gdPlotParameters(250000, c(0.71), c("red", "green")),
gdPlotDataSourceParameters(2500))
## End(Not run)
```

---

## `gdPlotDataSourceParameters` *Specify plot parameters for data source*

---

### Description

Specify plot parameters for data source passed to function gdPlot2dProjection().

### Usage

```
gdPlotDataSourceParameters(numberOfRandomPoints = 0, color = "blue")
```

### Arguments

| | |
|---|---|
| `numberOfRandomPoints` | Number of randomly selected rows in data source |
| `color` | Colour for data points of data source |

### Examples

```
## Not run: gdPlotDataSourceParameters(2500)
```

---

## `gdPlotParameters` *Specify plot parameters for generative data*

---

### Description

Specify plot parameters for generative data passed to function gdPlot2dProjection(). When density value thresholds with assigned colors are specified generative data is drawn for density value ranges in increasing order.

### Usage

```
gdPlotParameters(
  numberOfRandomPoints = 0,
  densityValueThresholds = c(),
  densityValueColors = c("red")
)
```

### Arguments

| | |
|---|---|
| `numberOfRandomPoints` | Number of randomly selected rows in generative data |
| `densityValueThresholds` | Vector of density value thresholds |
| `densityValueColors` | Vector of colors assigned to density value thresholds. The size must be the size of densityValueThresholds plus one. |

### Examples

```
## Not run: gdPlotParameters(250000, c(0.75), c("red", "green"))
```

## gdRead *Read generative data and data source*

### Description

Read generative data and data source from specified files. Read in generative data and data source are accessed in gdPlot2dProjection(), generative data is accessed in gdGetRow(), gdCalculateDensityValue() and gdCalculateDensityValueQuantile().

### Usage

```
gdRead(generativeDataFileName, dataSourceFileName = "")
```

### Arguments

| | |
|---|---|
| `generativeDataFileName` | Name of generative data file |
| `dataSourceFileName` | Name of data source file |

### Value

None

### Examples

```
## Not run: gdRead("gd.bin", "iris4d.bin")
```

# Index

∗ **package**
    ganGenerativeData-package, 2

dsActivateColumns, 8
dsCreateWithDataFrame, 8
dsDeactivateColumns, 9
dsGetActiveColumnNames, 10
dsGetInactiveColumnNames, 10
dsGetNumberOfRows, 11
dsGetRow, 11
dsRead, 12
dsWrite, 12

ganGenerativeData
        *(*ganGenerativeData-package*)*, 2
ganGenerativeData-package, 2
gdCalculateDensityValue, 13
gdCalculateDensityValueQuantile, 13
gdCalculateDensityValues, 14
gdComplete, 15
gdGenerate, 15
gdGetNumberOfRows, 16
gdGetRow, 17
gdKNearestNeighbors, 17
gdPlot2dProjection, 18
gdPlotDataSourceParameters, 19
gdPlotParameters, 19
gdRead, 20