t-SNE

鈴木瑞人
東京大学大学院　新領域創成科学研究科
メディカル情報生命専攻
博士課程1年

Package ‘tsne’

July 15, 2016

Type Package

Title T-Distributed Stochastic Neighbor Embedding for R (t-SNE)

Version 0.1-3

Date 2016-06-04

Author Justin Donaldson <jdonaldson@gmail.com>

Maintainer Justin Donaldson <jdonaldson@gmail.com>

Description A ``pure R'' implementation of the t-SNE algorithm.

License GPL

LazyLoad yes

NeedsCompilation no

URL https://github.com/jdonaldson/rtsne/

BugReports https://github.com/jdonaldson/rtsne/issues

Repository CRAN

Date/Publication 2016-07-15 20:02:16

References

- L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9 (Nov) : 2579-2605, 2008.
- L.J.P. van der Maaten. Learning a Parametric Embedding by Preserving Local Structure. In Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR W&CP 5:384-391, 2009.

tsne	*The t-SNE method for dimensionality reduction*

Description

Provides a simple function interface for specifying t-SNE dimensionality reduction on R matrices or "dist" objects.

Usage

```
tsne(X, initial_config = NULL, k = 2, initial_dims = 30, perplexity = 30,
     max_iter = 1000, min_cost = 0, epoch_callback = NULL, whiten = TRUE,
     epoch=100)
```

Arguments

`X`	The R matrix or "dist" object
`initial_config`	an argument providing a matrix specifying the initial embedding for X. See Details.
`k`	the dimension of the resulting embedding.
`initial_dims`	The number of dimensions to use in reduction method.
`perplexity`	Perplexity parameter. (optimal number of neighbors)
`max_iter`	Maximum number of iterations to perform.

`min_cost`	The minimum cost value (error) to halt iteration.
`epoch_callback`	A callback function used after each epoch (an epoch here means a set number of iterations)
`whiten`	A boolean value indicating whether the matrix data should be whitened.
`epoch`	The number of iterations in between update messages.

Details

When the initial_config argument is specified, the algorithm will automatically enter the *final momentum* stage. This stage has less large scale adjustment to the embedding, and is intended for small scale tweaking of positioning. This can greatly speed up the generation of embeddings for various similar X datasets, while also preserving overall embedding orientation.

Value

An R object containing a *ydata* embedding matrix, as well as a the matrix of probabilities P

Author(s)

Justin Donaldson (jdonaldson@gmail.com)

References

L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research* 9 (Nov) : 2579-2605, 2008.

L.J.P. van der Maaten. Learning a Parametric Embedding by Preserving Local Structure. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics* (AISTATS), JMLR W&CP 5:384-391, 2009.

tsneパッケージ

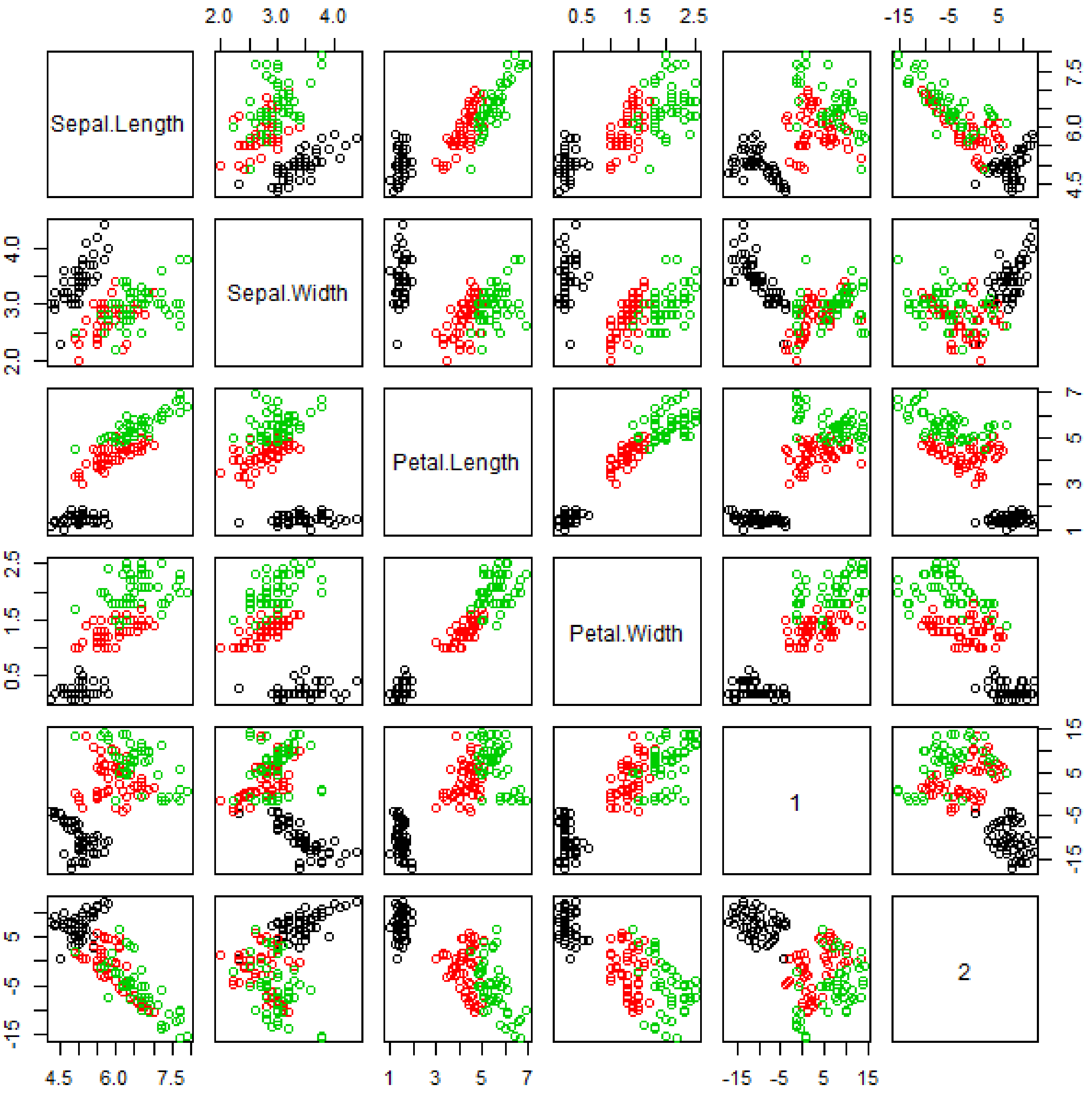
```
install.packages("tsne",dependencies=T)
library(tsne)
```

tsne_iris = tsne(iris[,1:4])

```
> library(tsne)
> tsne_iris = tsne(iris[,1:4])
sigma summary: Min. : 0.4865 |1st Qu. : 0.5879 |Median : 0.6149 |Mean : 0.6231 |3rd Qu. : 0.6549 |Max.
Epoch: Iteration #100 error is: 12.7292704163109
Epoch: Iteration #200 error is: 0.232290830193848
Epoch: Iteration #300 error is: 0.231638445928024
Epoch: Iteration #400 error is: 0.231618364669694
Epoch: Iteration #500 error is: 0.231617866665524
Epoch: Iteration #600 error is: 0.231617842641033
Epoch: Iteration #700 error is: 0.231617841601354
Epoch: Iteration #800 error is: 0.231617841553128
Epoch: Iteration #900 error is: 0.231617841550944
Epoch: Iteration #1000 error is: 0.231617841550844
```

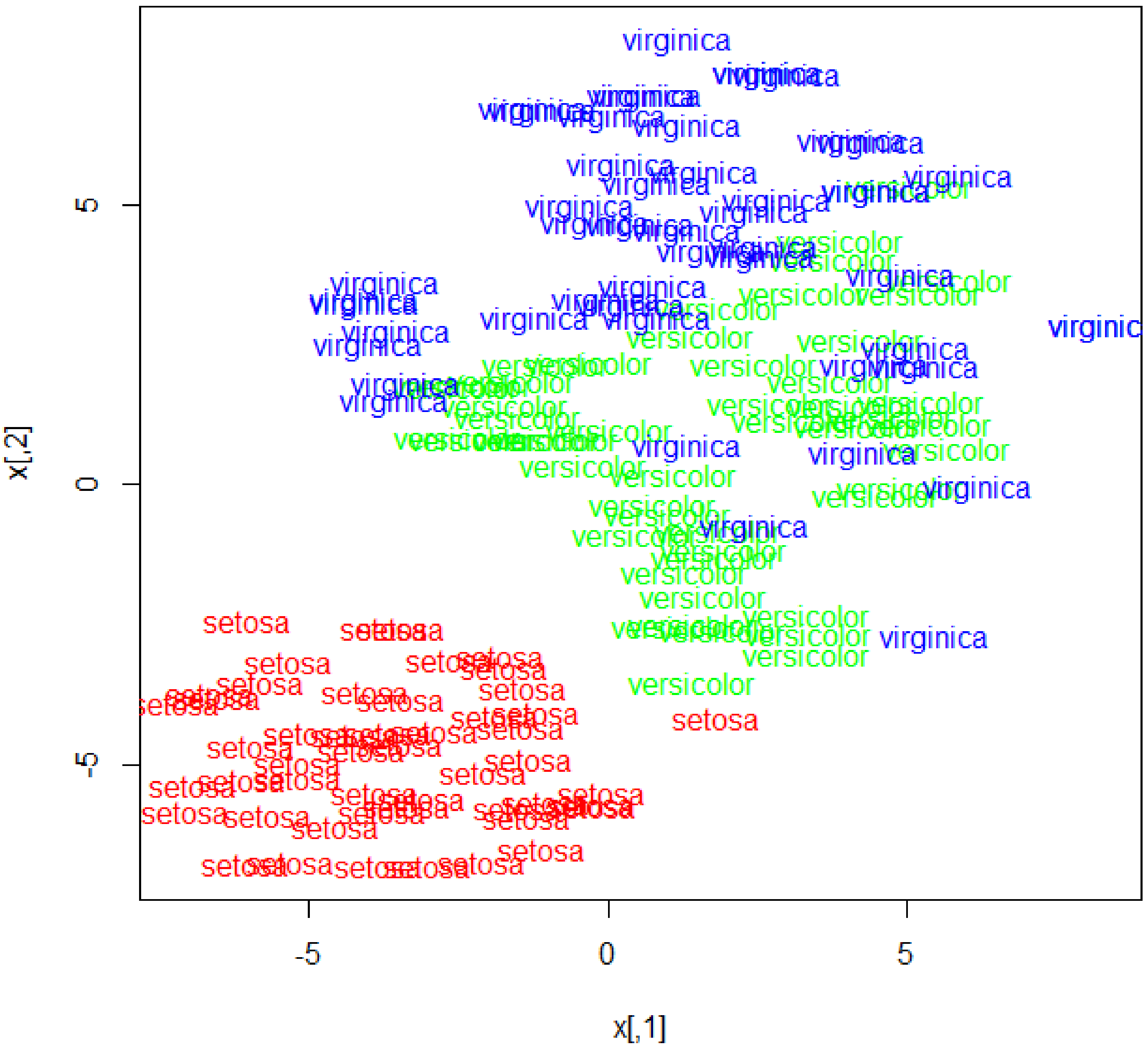
```
> tsne_iris
              [,1]        [,2]
 [1,] -10.98973474  7.49262716
 [2,]  -7.84621195  3.59544599
 [3,]  -7.58935475  6.62858434
 [4,]  -5.67591894  7.58956779
 [5,] -12.80539450  8.22018033
 [6,] -13.27932142 10.03995551
 [7,]  -6.68169151  9.69168386
 [8,]  -9.25604715  7.66907250
 [9,]  -3.94173557  7.05463816
[10,]  -6.28195051  5.45809394
[11,] -11.37102794  9.70250162
[12,] -15.57301134  7.64398412
[13,]  -6.41587220  4.71583114
[14,]  -4.03548040  7.88542046
[15,] -10.70601617 11.88941310
[16,] -13.43981814 12.02887269
[17,] -13.82423468  5.01899094
[18,] -11.51974786  6.03013178
[19,] -11.29133891 10.75665544
[20,] -13.75888831  9.06464611
[21,]  -9.83084552  8.97065701
[22,] -12.78490820  6.74851720
```

```
tsne_iris
x=cbind(iris[,1:4],tsne_iris)
x
plot(x,col=iris$Species)
```



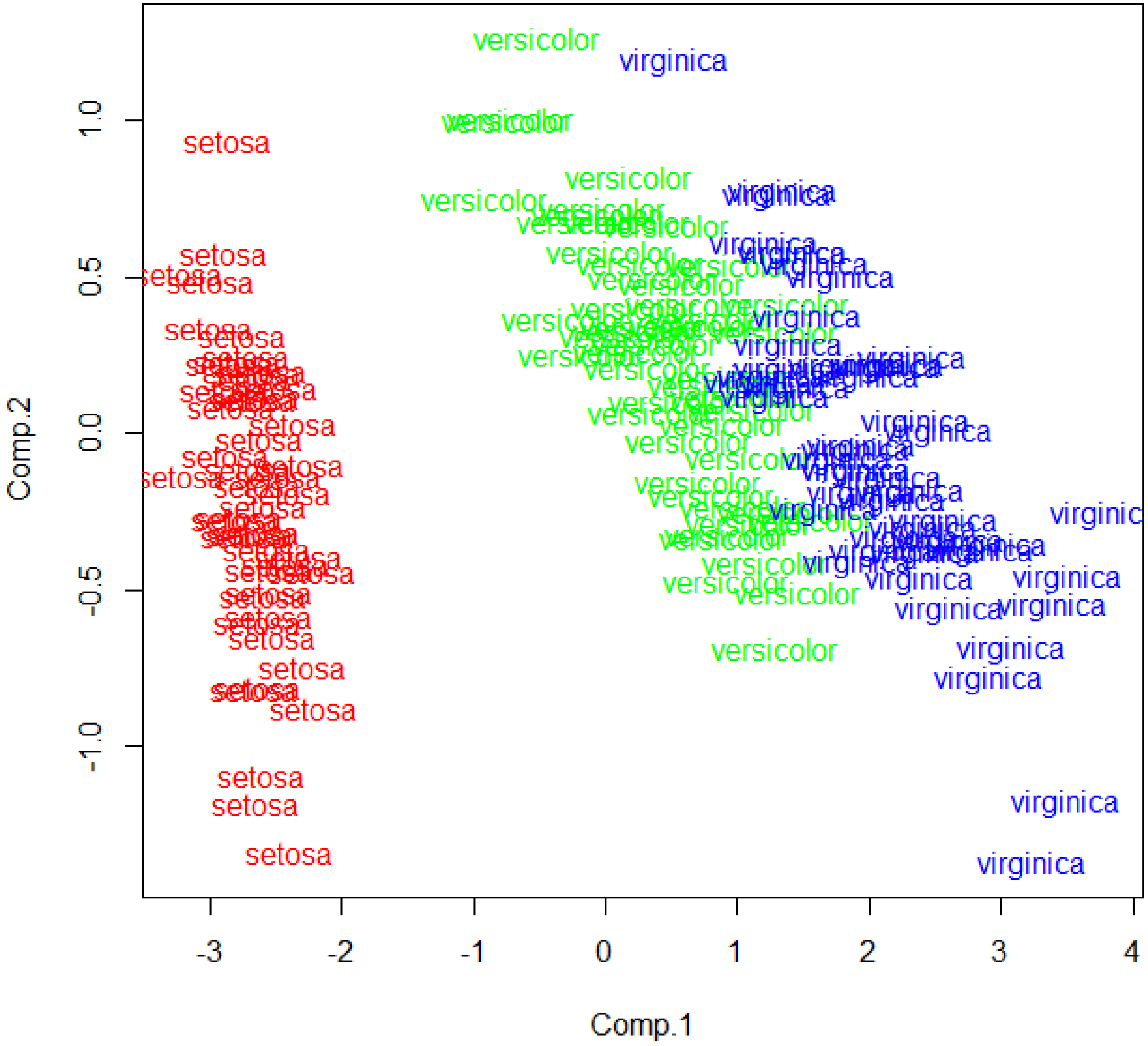
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
1
2
2.0
3.0
4.0
0.5
1.5
2.5
-15
-5
5
7.5
6.0
4.5
4.0
3.0
2.0
7
5
3
1
2.5
1.5
0.5
15
5
-5
-15
5
-5
-15
4.5
6.0
7.5
1
3
5
7
-15
-5
5
15

```
colors = rainbow(length(unique(iris$Species)))
names(colors) = unique(iris$Species)
ecb = function(x,y){ plot(x,t='n'); text(x,labels=iris$Species,
col=colors[iris$Species]) }
tsne_iris = tsne(iris[,1:4], epoch_callback = ecb, perplexity=50)
```



virginica
versicolor
setosa
5
0
-5
-5
0
5
x[,2]
x[,1]

```
dev.new()
pca_iris = princomp(iris[,1:4])$scores[,1:2]
plot(pca_iris, t='n')
text(pca_iris, labels=iris$Species,col=colors[iris$Species])
```



Comp.1
Comp.2
-3
-2
-1
0
1
2
3
4
-1.0
-0.5
0.0
0.5
1.0
setosa
versicolor
virginica

Package ‘Rtsne’

August 29, 2016

Type Package

Title T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation

Version 0.11

Description An R wrapper around the fast T-distributed Stochastic Neighbor Embedding implementation by Van der Maaten.

License BSD_3_clause + file LICENSE

URL https://github.com/jkrijthe/Rtsne

Imports Rcpp (>= 0.11.0)

LinkingTo Rcpp

Suggests testthat

RoxygenNote 5.0.1

NeedsCompilation yes

Author Jesse Krijthe [aut, cre],
Laurens van der Maaten [cph] (Author of original C++ code)

Maintainer Jesse Krijthe <jkrijthe@gmail.com>

Repository CRAN

Date/Publication 2016-06-30 13:41:40

`Rtsne`	*Barnes-Hut implementation of t-Distributed Stochastic Neighbor Embedding*

Description

Wrapper for the C++ implementation of Barnes-Hut t-Distributed Stochastic Neighbor Embedding. t-SNE is a method for constructing a low dimensional embedding of high-dimensional data, distances or similarities. Exact t-SNE can be computed by setting theta=0.0.

Usage

```
Rtsne(X, ...)

## Default S3 method:
Rtsne(X, dims = 2, initial_dims = 50, perplexity = 30,
  theta = 0.5, check_duplicates = TRUE, pca = TRUE, max_iter = 1000,
  verbose = FALSE, is_distance = FALSE, Y_init = NULL, ...)

## S3 method for class 'dist'
Rtsne(X, ..., is_distance = TRUE)

## S3 method for class 'data.frame'
Rtsne(X, ...)
```

Arguments

`X`	matrix; Data matrix
`...`	Other arguments that can be passed to Rtsne
`dims`	integer; Output dimensionality (default: 2)
`initial_dims`	integer; the number of dimensions that should be retained in the initial PCA step (default: 50)
`perplexity`	numeric; Perplexity parameter
`theta`	numeric; Speed/accuracy trade-off (increase for less accuracy), set to 0.0 for exact TSNE (default: 0.5)
`check_duplicates`	logical; Checks whether duplicates are present. It is best to make sure there are no duplicates present and set this option to FALSE, especially for large datasets (default: TRUE)
`pca`	logical; Whether an initial PCA step should be performed (default: TRUE)
`max_iter`	integer; Number of iterations (default: 1000)
`verbose`	logical; Whether progress updates should be printed (default: FALSE)
`is_distance`	logical; Indicate whether X is a distance matrix (experimental, default: FALSE)
`Y_init`	matrix; Initial locations of the objects. If NULL, random initialization will be used (default: NULL). Note that when using this, the initial stage with exaggerated perplexity values and a larger momentum term will be skipped.

Details

Given a distance matrix D between input objects (which by default, is the euclidean distances between two objects), we calculate a similarity score in the original space p_ij.

$$p_{j|i} = \frac{\exp(-\|D_{ij}\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|D_{ij}\|^2/2\sigma_i^2)}$$

which is then symmetrized using:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

. The σ for each object is chosen in such a way that the perplexity of p_jli has a value that is close to the user defined perplexity. This value effectively controls how many nearest neighbours are taken into account when constructing the embedding in the low-dimensional space. For the low-dimensional space we use the Cauchy distribution (t-distribution with one degree of freedom) as the distribution of the distances to neighbouring objects:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l} 1 + \|y_k - y_l\|^2)^{-1}}$$

. By changing the location of the objects y in the embedding to minimize the Kullback-Leibler divergence between these two distributions q_{ij} and p_{ij}, we create a map that focusses on small-scale structure, due to the assymetry of the KL-divergence. The t-distribution is chosen to avoid the crowding problem: in the original high dimensional space, there are potentially many equidistant objects with moderate distance from a particular object, more than can be accounted for in the low dimensional representation. The t-distribution makes sure that these objects are more spread out in the new representation.

Value

List with the following elements:

`Y`	Matrix containing the new representations for the objects
`N`	Number of objects
`origD`	Original Dimensionality before TSNE
`perplexity`	See above
`theta`	See above
`costs`	The cost for every object after the final iteration
`itercosts`	The total costs (KL-divergence) for all objects in every 50th + the last iteration

Methods (by class)

- `default`: Default Interface
- `dist`: tsne on given dist object
- `data.frame`: tsne on data.frame

References

Maaten, L. Van Der, 2014. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research, 15, p.3221-3245.

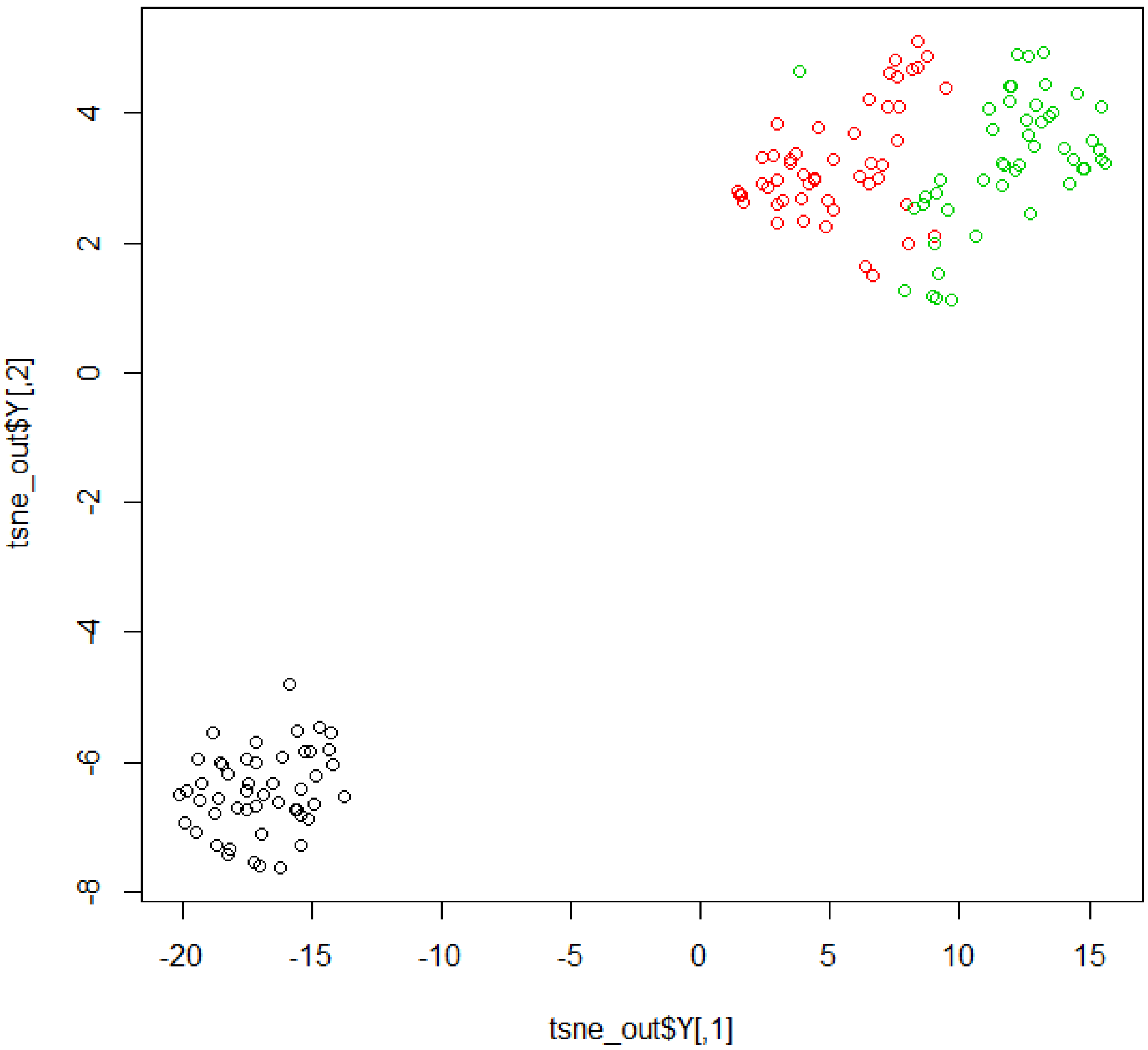
van der Maaten, L.J.P. & Hinton, G.E., 2008. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research, 9, pp.2579-2605.

Rtsneパケージ(こちらの方が早そう)

```
install.packages("Rtsne",dependencies=T)
library(Rtsne)
```

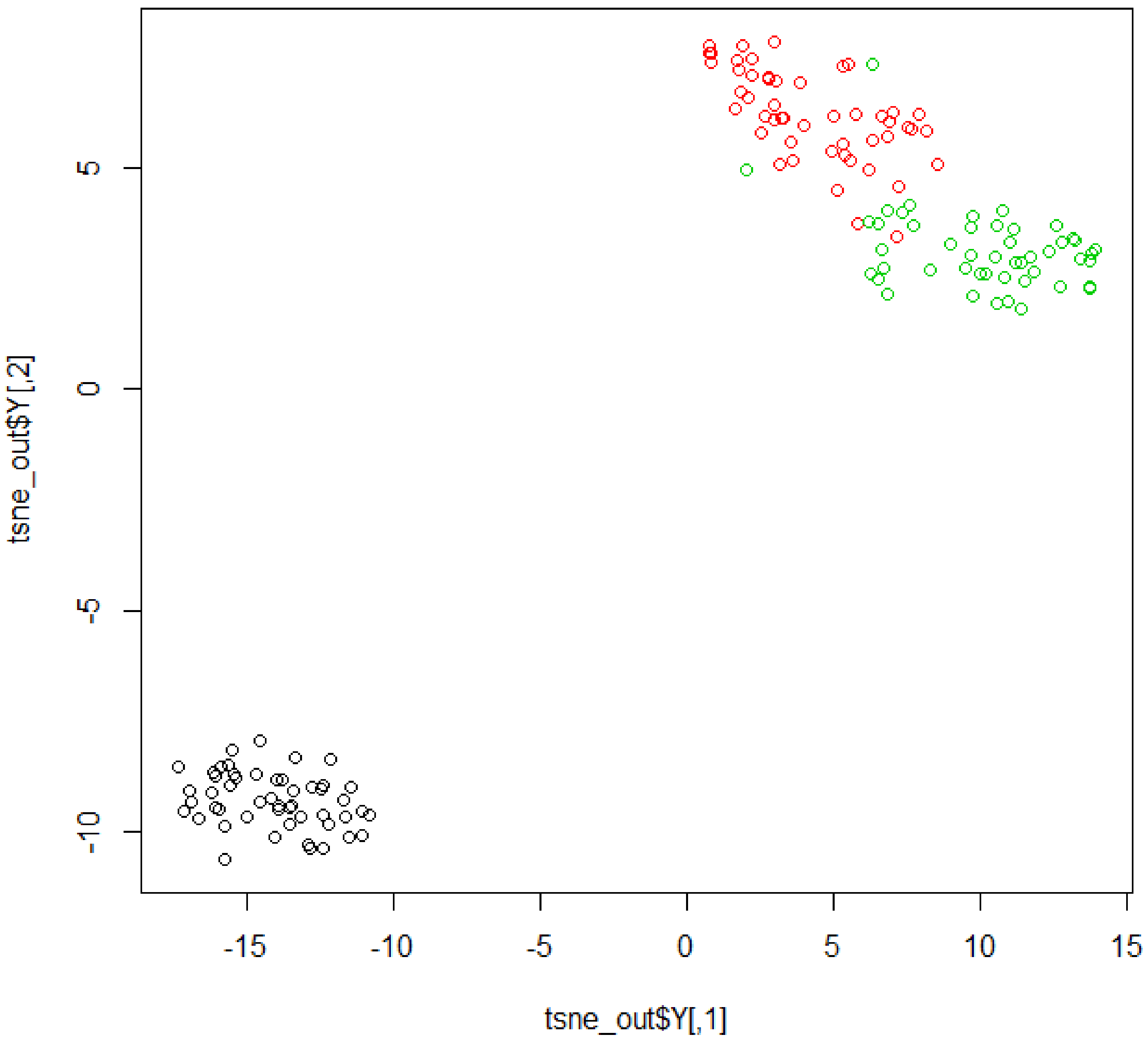
```
iris_unique <- unique(iris) # Remove duplicates
iris_matrix <- as.matrix(iris_unique[,1:4])
set.seed(42) # Set a seed if you want reproducible results
tsne_out <- Rtsne(iris_matrix) # Run TSNE
```

```
# Show the objects in the 2D tsne representation
plot(tsne_out$Y,col=iris_unique$Species)
```



4
2
0
-2
-4
-6
-8
tsne_out$Y[,2]
-20
-15
-10
-5
0
5
10
15
tsne_out$Y[,1]

```
# Using a dist object
tsne_out <- Rtsne(dist(iris_matrix))
plot(tsne_out$Y,col=iris_unique$Species)
```


tsne_out$Y[,1]
tsne_out$Y[,2]
-15
-10
-5
0
5
10
15
5
0
-5
-10