

# Package ‘hNMF’

June 1, 2018

**Title** Hierarchical Non-Negative Matrix Factorization

**Version** 0.6

**Author** Nicolas Sauwen

**Maintainer** Nicolas Sauwen <nicolas.sauwen@openanalytics.eu>

**Description** Hierarchical and single-level non-negative matrix factorization. Several NMF algorithms are available.

**Depends** R (>= 3.3.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** NMF, oro.nifti, tcltk, nnls, R.matlab, spatialfil, rasterImage, stats, graphics, grDevices, MASS

**RoxygenNote** 6.0.1.9000

**Suggests** testthat

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-06-01 12:30:06 UTC

## R topics documented:

HALSacc . . . . .	2
hNMF . . . . .	2
imoverlay . . . . .	3
initializeNMF . . . . .	4
initializeSPA . . . . .	4
oneLevelNMF . . . . .	5
PGNMF . . . . .	6
preProcesInputData . . . . .	7
residualNMF . . . . .	7
scaleNMFResult . . . . .	8
semiNMF . . . . .	8
<b>Index</b>	<b>10</b>

---

HALSacc	<i>Accelerated hierarchical alternating least squares NMF. For a reference to the method, see N. Gillis, Nonnegative matrix factorization: complexity, algorithms and applications [Section 4.2, Algo. 6], PhD thesis, Université catholique de Louvain, February 2011.</i>
---------	---

---

### Description

Accelerated hierarchical alternating least squares NMF. For a reference to the method, see N. Gillis, Nonnegative matrix factorization: complexity, algorithms and applications [Section 4.2, Algo. 6], PhD thesis, Université catholique de Louvain, February 2011.

### Usage

```
HALSacc(X, nmfMod, alpha = 1, maxiter = 1000, checkDivergence = FALSE)
```

### Arguments

X	Input data matrix, each column represents one observation and the rows correspond to the different features
nmfMod	Valid NMF model, containing initialized factor matrices (in accordance with the NMF package definition)
alpha	Nonnegative parameter of the accelerated method
maxiter	Maximum number of iterations
checkDivergence	currently not in use, to be implemented

### Value

Resulting NMF model (in accordance with the NMF package definition)

### Author(s)

nsauwen

---

hNMF	<i>Hierarchical non-negative matrix factorization.</i>
------	--

---

### Description

Hierarchical non-negative matrix factorization.

### Usage

```
hNMF(nmfInput, nmfMethod = "HALSacc")
```

**Arguments**

nmfInput            List with NMF input attributes  
 nmfMethod          String referring to the NMF algorithm to be used.

**Value**

Resulting NMF model (in accordance with NMF package definition)

**Author(s)**

Nicolas Sauwen

**Examples**

```
# create nmfInput object
X <- matrix(runif(10*20), 10,20)
bgImageTensor <- array(0,dim=dim(X))
selectVect <- array(1,dim=dim(X))
nmfInput <- NULL
nmfInput$numRows <- nrow(X)
nmfInput$numCols <- ncol(X)
nmfInput$numSlices <- 1
nmfInput$bgImageTensor <- bgImageTensor
nmfInput$selectVect <- selectVect

# run NMF with default algorithm, 5 runs with random initialization
NMfresult1 <- oneLevelNMF(X, rank=2, nruns=5)

# run NMF with specified algorithm and with initialized sources
W0 <- initializeSPA(X,3)
NMfresult2 <- oneLevelNMF(X, rank=3, method="HALSacc", initData = W0)
```

---

imoverlay

---

*Overlay a mask or a color scaled image on top of a background image*


---

**Description**

Overlay a mask or a color scaled image on top of a background image

**Usage**

```
imoverlay(image, overlay, selectVect = NULL, color = c(0, 1, 0))
```

**Arguments**

image	A matrix, background image
overlay	A matrix, serving as the overlay mask or figure
selectVect	A matrix (binary values), specifying which matrix elements are to be overlaid
color	3-element vector, defining the RGB color to be used in case the overlay is a mask

**Author(s)**

Nicolas Sauwen

---

initializeNMF	<i>Initialize NMF model with initial spectral data</i>
---------------	--

---

**Description**

Initialize NMF model with initial spectral data

**Usage**

```
initializeNMF(X, initData = NULL)
```

**Arguments**

X	input matrix
initData	source or abundance matrix with initial values

---

initializeSPA	<i>The successive projection algorithm, a useful method for initializing the NMF source matrix</i>
---------------	--

---

**Description**

The successive projection algorithm, a useful method for initializing the NMF source matrix

**Usage**

```
initializeSPA(data, nSources)
```

**Arguments**

data	Input data matrix. The columns correspond to the data points, each row represents one feature
nSources	Number of sources to be obtained

**Value**

Matrix with initialized sources as its columns

**Author(s)**

Nicolas Sauwen

**Examples**

```
# random data
X <- matrix(runif(10*20), 10,20)

# Create initial source matrix for 3 sources
W0 <- initializeSPA(X,3)
```

---

oneLevelNMF	<i>Perform Non-Negative Matrix factorization</i>
-------------	--

---

**Description**

Perform Non-Negative Matrix factorization

**Usage**

```
oneLevelNMF(X, rank, initData = NULL, method = "PGNMF", nruns = 10,
  checkDivergence = TRUE)
```

**Arguments**

X	input matrix. Each column represents one observation and the rows correspond to the different features
rank	number of NMF components to be found
initData	either of the NMF factor matrices, with initial values
method	name of the NMF method to be used. "PGNMF" (default) and "HALSacc" are available by default. Any method from the NMF package can also be specified
nruns	number of NMF runs. It is recommended to run the NMF analyses multiple times when random seeding is used, to avoid a suboptimal solution
checkDivergence	Boolean indicating whether divergence checking should be performed

**Value**

Scaled NMF model (in accordance with the NMF package definition)

**Author(s)**

Nicolas Sauwen

**Examples**

```
# random data
X <- matrix(runif(10*20), 10,20)

# run NMF with default algorithm, 5 runs with random initialization
NMFresult1 <- oneLevelNMF(X, rank=2, nruns=5)

# run NMF with specified algorithm and with initialized sources
W0 <- initializeSPA(X,3)
NMFresult2 <- oneLevelNMF(X, rank=3, method="HALSacc", initData = W0)
```

---

PGNMF	<i>NMF by alternating non-negative least squares using projected gradients. For a reference to the method, see C.-J. Lin, "Projected Gradient Methods for Non-negative Matrix Factorization", Neural computation 19.10 (2007): 2756-2779.</i>
-------	---

---

**Description**

NMF by alternating non-negative least squares using projected gradients. For a reference to the method, see C.-J. Lin, "Projected Gradient Methods for Non-negative Matrix Factorization", Neural computation 19.10 (2007): 2756-2779.

**Usage**

```
PGNMF(X, nmfMod, tol = 1e-05, maxIter = 500, timeLimit = 300,
      checkDivergence = TRUE)
```

**Arguments**

X	Input data matrix, each column represents one data point and the rows correspond to the different features
nmfMod	Valid NMF model, containing initialized factor matrices (in accordance with the NMF package definition)
tol	Tolerance for a relative stopping condition
maxIter	Maximum number of iterations
timeLimit	Limit of time duration NMF analysis
checkDivergence	Boolean indicating whether divergence checking should be performed Default is TRUE, but it should be set to FALSE when using random initialization

**Value**

Resulting NMF model (in accordance with the NMF package definition)

**Author(s)**

nsauwen

---

preProcesInputData	<i>Condition input data matrix properly for NMF</i>
--------------------	---

---

**Description**

Condition input data matrix properly for NMF

**Usage**

```
preProcesInputData(X)
```

**Arguments**

X	input matrix
---	--------------

**Value**

matrix with non-zero elements

---

residualNMF	<i>Computation of relative NMF residual per observation</i>
-------------	---

---

**Description**

Computation of relative NMF residual per observation

**Usage**

```
residualNMF(X, nmfFit)
```

**Arguments**

X	Input data matrix, each column represents one observation
nmfFit	NMF model fitted to the input data in X

**Value**

Relative residual per observation, returned as a vector

**Author(s)**

nsauwen

---

scaleNMFResult	<i>Apply fixed scaling to NMF model matrices by normalizing the basis vectors</i>
----------------	---

---

**Description**

Apply fixed scaling to NMF model matrices by normalizing the basis vectors

**Usage**

```
scaleNMFResult(NMFResult)
```

**Arguments**

NMFResult	Fitted NMF model
-----------	------------------

**Value**

NMFResult Rescaled NMF model

**Author(s)**

Nicolas Sauwen

---

semiNMF	<i>Semi-NMF based on multiplicative update rules. Reference: C. Ding, T. Li, and M.I. Jordan, "Convex and semi-nonnegative matrix factorizations", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 1, pp. 45-55, 2010.</i>
---------	---

---

**Description**

Semi-NMF based on multiplicative update rules. Reference: C. Ding, T. Li, and M.I. Jordan, "Convex and semi-nonnegative matrix factorizations", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 1, pp. 45-55, 2010.

**Usage**

```
semiNMF(X, nmfMod, maxiter = 2000, checkDivergence = FALSE)
```



**Arguments**

<code>X</code>	Input data matrix, each column represents one observation and the rows correspond to the different features
<code>nmfMod</code>	Valid NMF model, containing initialized factor matrices (in accordance with the NMF package definition)
<code>maxiter</code>	Maximum number of iterations
<code>checkDivergence</code>	currently not in use, to be implemented

**Value**

Resulting NMF model (in accordance with the NMF package definition)

**Author(s)**

nsauwen

# Index

HALSacc, [2](#)

hNMF, [2](#)

imoverlay, [3](#)

initializeNMF, [4](#)

initializeSPA, [4](#)

oneLevelNMF, [5](#)

PGNMF, [6](#)

preProcesInputData, [7](#)

residualNMF, [7](#)

scaleNMFResult, [8](#)

semiNMF, [8](#)