Package ‘RCM’

May 30, 2024

Type Package

Title Fit row-column association models with the negative binomial distribution for the microbiome

Version 1.20.0

Description Combine ideas of log-linear analysis of contingency table, flexible response function estimation and empirical Bayes dispersion estimation for explorative visualization of microbiome datasets. The package includes unconstrained as well as constrained analysis. In addition, diagnostic plot to detect lack of fit are available.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports RColorBrewer, alabama, edgeR, reshape2, tseries, stats, VGAM, ggplot2 (>= 2.2.1.9000), nleqslv, phyloseq, tensor, MASS, grDevices, graphics, methods

Depends R (>= 4.0), DBI

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

biocViews Metagenomics, DimensionReduction, Microbiome, Visualization

BugReports https://github.com/CenterForStatistics-UGent/RCM/issues


git_url https://git.bioconductor.org/packages/release/bioc/vignettes/RCM

git_branch RELEASE_3_19

git_last_commit 363e900

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29

Author Stijn Hawinkel [cre, aut] (<https://orcid.org/0000-0002-4501-5180>)

Maintainer Stijn Hawinkel <stijn.hawinkel@psb.ugent.be>
## Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addOrthProjection</td>
<td>3</td>
</tr>
<tr>
<td>arrayprod</td>
<td>4</td>
</tr>
<tr>
<td>buildCentMat</td>
<td>5</td>
</tr>
<tr>
<td>buildConfMat</td>
<td>5</td>
</tr>
<tr>
<td>buildConfMat.character</td>
<td>6</td>
</tr>
<tr>
<td>buildConfMat.data.frame</td>
<td>6</td>
</tr>
<tr>
<td>buildCovMat</td>
<td>7</td>
</tr>
<tr>
<td>buildDesign</td>
<td>8</td>
</tr>
<tr>
<td>checkAlias</td>
<td>8</td>
</tr>
<tr>
<td>constrCorresp</td>
<td>9</td>
</tr>
<tr>
<td>correctXMissingness</td>
<td>10</td>
</tr>
<tr>
<td>deviances</td>
<td>10</td>
</tr>
<tr>
<td>dLR_nb</td>
<td>11</td>
</tr>
<tr>
<td>dNBabundsOld</td>
<td>12</td>
</tr>
<tr>
<td>dNBlibSizes</td>
<td>13</td>
</tr>
<tr>
<td>dNBlibcolNP</td>
<td>14</td>
</tr>
<tr>
<td>dNBlibcolOld</td>
<td>14</td>
</tr>
<tr>
<td>dNBlibcol_constr</td>
<td>16</td>
</tr>
<tr>
<td>dNBlibcol_constr_noLab</td>
<td>16</td>
</tr>
<tr>
<td>dNBlibrow</td>
<td>17</td>
</tr>
<tr>
<td>dNBpsis</td>
<td>18</td>
</tr>
<tr>
<td>ellipseCoord</td>
<td>19</td>
</tr>
<tr>
<td>estDisp</td>
<td>20</td>
</tr>
<tr>
<td>estNBparams</td>
<td>21</td>
</tr>
<tr>
<td>estNBparamsNoLab</td>
<td>22</td>
</tr>
<tr>
<td>estNPresp</td>
<td>23</td>
</tr>
<tr>
<td>extractCoord</td>
<td>25</td>
</tr>
<tr>
<td>extractE</td>
<td>26</td>
</tr>
<tr>
<td>filterConfounders</td>
<td>26</td>
</tr>
<tr>
<td>getDevianceRes</td>
<td>27</td>
</tr>
<tr>
<td>getDevMat</td>
<td>28</td>
</tr>
<tr>
<td>getInflCol</td>
<td>29</td>
</tr>
<tr>
<td>getInflRow</td>
<td>29</td>
</tr>
<tr>
<td>getInt</td>
<td>30</td>
</tr>
<tr>
<td>getLogLik</td>
<td>30</td>
</tr>
<tr>
<td>getModelMat</td>
<td>31</td>
</tr>
<tr>
<td>getRowMat</td>
<td>31</td>
</tr>
<tr>
<td>GramSchmidt</td>
<td>32</td>
</tr>
<tr>
<td>heq_nb</td>
<td>32</td>
</tr>
<tr>
<td>heq_nb_jac</td>
<td>33</td>
</tr>
<tr>
<td>indentPlot</td>
<td>33</td>
</tr>
<tr>
<td>inertia</td>
<td>34</td>
</tr>
<tr>
<td>JacCol_constr</td>
<td>34</td>
</tr>
<tr>
<td>JacCol_constr_noLab</td>
<td>35</td>
</tr>
<tr>
<td>lik</td>
<td>36</td>
</tr>
<tr>
<td>LR_nb</td>
<td>37</td>
</tr>
</tbody>
</table>
This function adds orthogonal projections to a given plot

Description

This function adds orthogonal projections to a given plot

Usage

```
addOrthProjection(
    RCMplot, sample = NULL, species = NULL, variable = NULL, Dims = c(1, 2),
    addLabel = FALSE, labPos = NULL
)
```
arrayprod

An auxiliary R function to 'array' multiply an array with a vector, kindly provided by Joris Meys

Description
An auxiliary R function to 'array' multiply an array with a vector, kindly provided by Joris Meys

Usage
arrayprod(x, y)

Arguments
x        a axbxc array
y        a vector of length c

Value
a axb matrix. The ij-th element equals sum(x[i,j]*y)

Arguments

RCMplot the RCMplot object
sample, species, variable
            names or approximate coordinates of sample, species or variable
Dims The dimensions of the solutions that have been plotted
addLabel a boolean, should the r-s-psi label be added?
labPos  the position of the label. Will be calculated if not provided

Value
a modified ggplot object that contains the geom_segment object that draws the projection

See Also
plot.RCM

Examples

data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[seq_len(100)],
    prune_samples(sample_names(Zeller)[seq_len(50)], Zeller))
zellerRCM = RCM(tmpPhy, k = 2, round = TRUE)
zellerPlot = plot(zellerRCM, returnCoords = TRUE)
addOrthProjection(zellerPlot, species = c(-0.35,1.1), sample = c(1,1.2))
buildCentMat

Description
A function to build a centering matrix based on a dataframe

Usage
buildCentMat(object)

Arguments
object an rcm object or dataframe

Value
a centering matrix consisting of ones and zeroes, or a list with components
centMat a centering matrix consisting of ones and zeroes
datFrame The dataframe with factors with one level removed

buildConfMat A function to build the confounder matrices

Description
A function to build the confounder matrices

Usage
buildConfMat(x, ...)

Arguments
x a matrix, data frame or character string
... further arguments passed on to other methods

For the preliminary trimming, we do not include an intercept, but we do include all the levels of the factors using contrasts=FALSE: we want to do the trimming in every subgroup, so no hidden reference levels For the filtering we just use a model with an intercept and treatment coding, here the interest is only in adjusting the offset
Value

a list with components

confModelMatTrim
A confounder matrix without intercept, with all levels of factors present. This will be used to trim out taxa that have zero abundances in any subgroup defined by confounders

confModelMat
A confounder matrix with intercept, and with reference levels for factors absent. This will be used to fit the model to modify the independence model, and may include continuous variables

Description

buildConfMat.character

Usage

## S3 method for class 'character'
buildConfMat(character, physeq)

Arguments

confounders a numeric matrix of confounders
physeq a physeq object with a sample_data slot

Value

see buildConfMat.numeric

Description

buildConfMat.data.frame

Usage

## S3 method for class 'data.frame'
buildConfMat(character, physeq, n)
**buildCovMat**

**Arguments**

- **confounders**: a data frame of confounders
- **n**: the number of rows of the count matrix

**Value**

see buildConfMat

---

**buildCovMat**  
A function to build the covariate matrix of the constraints

**Description**

A function to build the covariate matrix of the constraints

**Usage**

```r
buildCovMat(covariates, dat)
```

**Arguments**

- **covariates**: the covariates, either as dataframe or as character string
- **dat**: the phyloseq object

  In this case we will 1) Include dummy’s for every level of the categorical variable, and force them to sum to zero. This is needed for plotting and required for reference level independent normalization. 2) Exclude an intercept. The density function f() will provide this already.

**Value**

- a list with components
  - **covModelMat**: The model matrix
  - **datFrame**: The data frame used to construct the model matrix
buildDesign  
\textit{A function to build the design matrix}

\textbf{Description}
A function to build the design matrix

\textbf{Usage}
\begin{verbatim}
buildDesign(sampleScore, responseFun)
\end{verbatim}

\textbf{Arguments}
\begin{itemize}
  \item \texttt{sampleScore} a vector of environmental scores
  \item \texttt{responseFun} A character string, indicating the shape of the response function
\end{itemize}

For dynamic response function estimation, the same design matrix as for the quadratic one is returned. Will throw an error when an unknown response function is provided

\textbf{Value}
A design matrix of dimension n-by-f

checkAlias  
\textit{Check for alias structures in a dataframe, and throw an error when one is found}

\textbf{Description}
Check for alias structures in a dataframe, and throw an error when one is found

\textbf{Usage}
\begin{verbatim}
checkAlias(datFrame, covariatesNames)
\end{verbatim}

\textbf{Arguments}
\begin{itemize}
  \item \texttt{datFrame} the data frame to be checked for alias structure
  \item \texttt{covariatesNames} The names of the variables to be considered
\end{itemize}

\textbf{Value}
Throws an error when an alias structure is detected, returns invisible otherwise
Examples

# Make a dataframe with aliased variables
def = data.frame(foo = rnorm(10), baa = rep(c(TRUE, FALSE), each = 5),
foo2 = factor(rep(c("male", "female"), each = 5))
checkAlias(df, c("foo", "baa"))
# Check test files for the error being thrown

---

constrCorresp | Constrained correspondence analysis with adapted powers

Description

Constrained correspondence analysis with adapted powers

Usage

constrCorresp(
  X,
  Y,
  rowExp,
  colExp,
  muMarg = outer(rowSums(X), colSums(X))/sum(X)
)

Arguments

X  outcome matrix
Y  constraining matrix
rowExp, colExp  see ?RCM_NB
muMarg  mean matrix under independence model

Details

the vegan version, adapted for flexible powers rowExp and colExp

Value

a list with eigenvalues, aliased variables and environmentam gradients
**correctXMissingness**

*Replace missing entries in X by their expectation to set their contribution to the estimating equations to zero*

**Description**

Replace missing entries in X by their expectation to set their contribution to the estimating equations to zero.

**Usage**

```r
correctXMissingness(X, mu, allowMissingness, naId)
```

**Arguments**

- `X`: the matrix of counts
- `mu`: the matrix of expectations
- `allowMissingness`: A boolean, are missing values present
- `naId`: The numeric index of the missing values in X

**Value**

The matrix X with the NA entries replaced by the corresponding entries in mu.

**Note**

This may seem like a hacky approach, but it avoids having to deal with NAs in functions like `crossprod()`.

**deviances**

*A function to extract deviances for all dimension, including after filtering on confounders*

**Description**

A function to extract deviances for all dimension, including after filtering on confounders.

**Usage**

```r
deviances(rcm, squaredSum = FALSE)
```
Arguments

rcm an object of the RCM class
squaredSum a boolean, should total deviance be returned?

Total deviances can be deceptive and not correspond to the differences in log-likelihood. As the dispersion is different for each model. To compare models it is better to compare likelihoods.

Value

If Sum is FALSE, a named array of deviance residuals of the independence model and all models with dimension 1 to k, including after filtering on confounders. Otherwise a table with total deviances (the sum of squared deviance residuals), deviance explained and cumulative deviance explained.

Description

A function that returns the value of the partial derivative of the log-likelihood ratio to alpha, keeping the response functions fixed

Usage

dLR_nb(
  Alpha,
  X,
  CC,
  responseFun = c("linear", "quadratic", "nonparametric", "dynamic"),
  psi,
  NB_params,
  NB_params_noLab,
  d,
  alphaK,
  k,
  centMat,
  nLambda,
  nLambda1s,
  thetaMat,
  muMarg,
  ncols,
  envGradEst,
  allowMissingness,
  naId,
  ...
)
Arguments

**Alpha**
a vector of length $d + k^{*}(2+(k-1)/2)$, the environmental gradient plus the lagrangian multipliers

**X**
the n-by-p count matrix

**CC**
a n-by-d covariate vector

**responseFun**
a character string indicating the type of response function

**psi**
a scalar, an importance parameter

**NB_params**
Starting values for the NB_params

**NB_params_noLab**
Starting values for the NB_params without label

**d**
an integer, the number of covariate parameters

**alphaK**
a matrix of environmental gradients of lower dimensions

**k**
an integer, the current dimension

**centMat**
a nLambda1s-by-d centering matrix

**nLambda**
an integer, number of lagrangian multipliers

**nLambda1s**
an integer, number of centering restrictions

**thetaMat**
a matrix of size n-by-p with estimated dispersion parameters

**muMarg**
an n-by-p offset matrix

**ncols**
a scalar, the number of columns of X

**envGradEst**
a character string, indicating how the environmental gradient should be fitted. ‘LR’ using the likelihood-ratio criterion, or ‘ML’ a full maximum likelihood solution

**allowMissingness**
A boolean, are missing values present

**naId**
The numeric index of the missing values in X

... further arguments passed on to other methods

Value

: The value of the lagrangian and the constraining equations

---

**dNBabundsOld**

A score function for the column components of the independence model (mean relative abundances)

Description

A score function for the column components of the independence model (mean relative abundances)

Usage

`dNBabundsOld(beta, X, reg, thetas, allowMissingness, naId)`
**dNBlibSizes**

**Arguments**

- **beta**: a vector of length p with current abundance estimates
- **X**: a n-by-p count matrix
- **reg**: a vector of length n with library sizes estimates
- **thetas**: a n-by-p matrix with overdispersion estimates in the rows
- **allowMissingness**: A boolean, are missing values present
- **naId**: The numeric index of the missing values in X

**Value**

a vector of length p with evaluations of the score function

---

**dNBlibSizes**

A score function for the row components of the independence model (library sizes)

---

**Description**

A score function for the row components of the independence model (library sizes)

**Usage**

dNBlibSizes(beta, X, reg, thetas, allowMissingness, naId)

**Arguments**

- **beta**: a vector of length n with current library size estimates
- **X**: a n-by-p count matrix
- **reg**: a vector of length p with relative abundance estimates
- **thetas**: a n-by-p matrix with overdispersion estimates in the rows
- **allowMissingness**: A boolean, are missing values present
- **naId**: The numeric index of the missing values in X

**Value**

a vector of length n with evaluations of the score function
**dNB1lcolNP**  
*Estimation of the parameters of a third degree GLM*

**Description**

Estimation of the parameters of a third degree GLM

**Usage**

\[
dNB1lcolNP(beta, X, reg, theta, muMarg, allowMissingness, naId, \ldots)
\]

**Arguments**

- **beta**: A vector of any length
- **X**: the data vector of length \( n \)
- **reg**: a \( nx\text{length}(beta) \) regressor matrix
- **theta**: a scalar, the overdispersion
- **muMarg**: the offset of length \( n \)
- **allowMissingness**: A boolean, are missing values present
- **naId**: The numeric index of the missing values in \( X \)
- **...**: further arguments passed on to the jacobian

**Value**

A vector of the same length as \( beta \) with evaluations of the score function

**dNB1lcolOldId**  
*A score function for the estimation of the column scores in an unconstrained RC(M) model*

**Description**

A score function for the estimation of the column scores in an unconstrained RC(M) model
Usage

dNB1lcolOld(
  beta,  
  X,      
  reg,    
  thetas, 
  muMarg, 
  k,      
  p,      
  n,      
  colWeights,  
  nLambda, 
  cMatK,   
  allowMissingness, 
  naId,    
  ...)
)

Arguments

beta     vector of length p+1+1+(k-1): p row scores, 1 centering, one normalization and (k-1) orthogonality lagrangian multipliers
X         the n x p data matrix
reg      a n x 1 regressor matrix: outer product of rowScores and psis
thetas   n x p matrix with the dispersion parameters (converted to matrix for numeric reasons)
muMarg   the n x p offset
k         an integer, the dimension of the RC solution
p         an integer, the number of taxa
n         an integer, the number of samples
colWeights the weights used for the restrictions
nLambda   an integer, the number of restrictions
cMatK     the lower dimensions of the colScores
allowMissingness
           A boolean, are missing values present
naId      The numeric index of the missing values in X
...        further arguments passed on to the jacobian

Value

A vector of length p+1+1+(k-1) with evaluations of the derivative of lagrangian
dNBllcol_constr
The score function of the response function for 1 taxon at the time

Description
The score function of the response function for 1 taxon at the time

Usage
dNBllcol_constr(betas, X, reg, theta, muMarg, psi, allowMissingness, naId)

Arguments
betas a vector of v parameters of the response function of a single taxon
X the count vector of length n
reg a n-by-v matrix of regressors
theta The dispersion parameter of this taxon
muMarg offset of length n
psi a scalar, the importance parameter
allowMissingness A boolean, are missing values present
naId The numeric index of the missing values in X

Even though this approach does not imply normalization over the parameters of all taxa, it is very fast and they can be normalized afterwards

Value
A vector of length v with the evaluation of the score functions

dNBllcol_constr_noLab
The score function of the general response function

Description
The score function of the general response function
Usage

dNBllcol_constr_noLab(
    betas,
    X,
    reg,
    thetasMat,
    muMarg,
    psi,
    allowMissingness,
    naId,
    ...
)

Arguments

  betas         a vector of regression parameters with length \( v \)
  X             the \( nxp \) data matrix
  reg           a matrix of regressors of dimension \( nxv \)
  thetasMat     A matrix of dispersion parameters
  muMarg        offset matrix of dimension \( nxp \)
  psi           a scalar, the importance parameter
  allowMissingness
                 A boolean, are missing values present
  naId          The numeric index of the missing values in \( X \)
  ...           further arguments passed on to the jacobian

Value

The evaluation of the score functions (a vector length \( v \))

\[ \text{dNBllrow} \quad \text{A score function of the NB for the row scores} \]

Description

A score function of the NB for the row scores

Usage

\[ \text{dNBllrow}(\]
  beta,
  X,
  reg,
  thetas,
\]
dNBpsis

Arguments

beta: a vector of length n + k +1 regression parameters to optimize
X: the data matrix of dimensions nxp
reg: a 1xp regressor matrix: outer product of column scores and psis
thetas: an n x p matrix with the dispersion parameters (converted to matrix for numeric reasons)
muMarg: an n x p offset matrix
k: a scalar, the dimension of the RC solution
n: a scalar, the number of samples
p: a scalar, the number of taxa
rowWeights: a vector of length n, the weights used for the restrictions
nLambda: an integer, the number of lagrangian multipliers
rMatK: the lower dimension row scores
allowMissingness: A boolean, are missing values present
naId: The numeric index of the missing values in X
... Other arguments passed on to the jacobian

Value

A vector of length n + k +1 with evaluations of the derivative of the lagrangian

Description

A score function for the psi of a given dimension

Usage

dNBpsis(beta, X, reg, theta, muMarg, allowMissingness, naId, ...)

A score function for the psi of a given dimension
**Arguments**

- **beta** a scalar, the initial estimate
- **X** the n-by-p count matrix
- **reg** the regressor matrix, the outer product of current row and column scores
- **theta** a n-by-p matrix with the dispersion parameters
- **muMarg** the nxp offset matrix
- **allowMissingness** A boolean, are missing values present
- **naId** The numeric index of the missing values in X
- **...** other arguments passed on to the jacobian

**Value**

The evaluation of the score function at beta, a scalar

**Description**

A function that returns the coordinates of an ellipse

**Usage**

`ellipseCoord(a, b, c, quadDrop = 0.95, nPoints = 100)`

**Arguments**

- **a, b, c** parameters of the quadratic function $a^2x + bx + c$
- **quadDrop** A scalar, fraction of peak height at which to draw the ellipse
- **nPoints** an integer, number of points to use to draw the ellipse

**Value**

a matrix with x and y coordinates of the ellipse
estDisp  

*Estimate the overdispersion*

---

**Description**

Estimate the overdispersion

**Usage**

```r
estDisp(
  X,
  cMat = NULL,
  rMat = NULL,
  muMarg,
  psis,
  trended.dispersion = NULL,
  prior.df = 10,
  dispWeights = NULL,
  rowMat = NULL,
  allowMissingness = FALSE,
  naId
)
```

**Arguments**

- `X`  
  the data matrix of dimensions n x p

- `cMat`  
  a 1 x p column scores matrix

- `rMat`  
  a n x 1 row scores matrix, if unconstrained

- `muMarg`  
  an n x p offset matrix

- `psis`  
  a scalar, the current psi estimate

- `trended.dispersion`  
  a vector of length p with pre-calculated trended.dispersion estimates. They do not vary in function of the offset anyway

- `prior.df`  
  an integer, number of degrees of freedom of the prior for the Bayesian shrinkage

- `dispWeights`  
  Weights for estimating the dispersion in a zero-inflated model

- `rowMat`  
  matrix of row scores in case of constrained ordination

- `allowMissingness`  
  A boolean, are missing values present

- `naId`  
  The numeric index of the missing values in X

**Details**

Information between taxa is shared with empirical Bayes using the edgeR package, where the time-limiting steps are programmed in C.
estNBparams

**Value**

A vector of length p with dispersion estimates

---

**Description**

A function to estimate the taxon-wise NB-params

**Usage**

```r
estNBparams(
  design,  
  thetas,  
  muMarg, 
  psi, 
  X, 
  nleqslv.control, 
  ncols, 
  initParam, 
  v, 
  dynamic = FALSE, 
  envRange, 
  allowMissingness, 
  naId
)
```

**Arguments**

- `design`: an n-by-v design matrix
- `thetas`: a vector of dispersion parameters of length p
- `muMarg`: an offset matrix
- `psi`: a scalar, the importance parameter
- `X`: the data matrix
- `nleqslv.control`: a list of control elements, passed on to nleqslv()
- `ncols`: an integer, the number of columns of X
- `initParam`: a v-by-p matrix of initial parameter estimates
- `v`: an integer, the number of parameters per taxon
- `dynamic`: a boolean, should response function be determined dynamically? See details
- `envRange`: a vector of length 2, giving the range of observed environmental scores
- `allowMissingness`: A boolean, are missing values present
The numeric index of the missing values in X
If dynamic is TRUE, quadratic response functions are fitted for every taxon. If the optimum falls outside of the observed range of environmental scores, a linear response function is fitted instead.

Value

a v-by-p matrix of parameters of the response function

description

A function to estimate the NB-params ignoring the taxon labels

Usage

estNBparamsNoLab(
  design,
  thetasMat,
  muMarg,
  psi,
  X,
  nleqslv.control,
  initParam,
  n,
  v,
  dynamic,
  envRange,
  preFabMat,
  allowMissingness,
  naId
)

Arguments

design an n-by-v design matrix
thetasMat A matrix of dispersion parameters
muMarg an offset matrix
psi a scalar, the importance parameter
X the data matrix
nleqslv.control a list of control elements, passed on to nleqslv()
initParam a vector of length v of initial parameter estimates
### estNPresp

**n**  
an integer, the number of samples

**v**  
an integer, the number of parameters per taxon

**dynamic**  
a boolean, should response function be determined dynamically? See details

**envRange**  
a vector of length 2, giving the range of observed environmental scores

**prefabMat**  
a pre-fabricated auxiliary matrix

**allowMissingness**  
A boolean, are missing values present

**naId**  
The numeric index of the missing values in X

If dynamic is TRUE, quadratic response functions are fitted for every taxon. If the optimum falls outside of the observed range of environmental scores, a linear response function is fitted instead

**Value**

a \( v \)-by-\( p \) matrix of parameters of the response function

### Description

Estimate the taxon-wise response functions non-parametrically

### Usage

```r
estNPresp(
  sampleScore,
  muMarg,
  X,
  ncols,
  thetas,
  n,
  coefInit,
  coefInitOverall,
  dfSpline,
  vgamMaxit,
  degree,
  verbose,
  allowMissingness,
  naId,
  ...
)
```
Arguments

- `sampleScore`: a vector of length n with environmental scores
- `muMarg`: the offset matrix
- `X`: the n-by-p data matrix
- `ncols`: an integer, the number of columns of X
- `thetas`: a vector of length p with dispersion parameters
- `n`: an integer, the number of samples
- `coefInit`: a 2-by-p matrix with current taxon-wise parameter estimates
- `coefInitOverall`: a vector of length 2 with current overall parameters
- `dfSpline`: a scalar, the degrees of freedom for the smoothing spline.
- `vgamMaxit`: Maximal number of iterations in the fitting of the GAM model
- `degree`: The degree if the parametric fit if the VGAM fit fails
- `verbose`: a boolean, should number of failed fits be reported
- `allowMissingness`: A boolean, are missing values present
- `naId`: The numeric index of the missing values in X
- `...`: further arguments, passed on to the VGAM::vgam() function

The negative binomial likelihood is still maximized, but now the response function is a non-parametric one. To avoid a perfect fit and overly flexible functions, we enforce smoothness restrictions. In practice we use a generalized additive model (GAM), i.e. with splines. The same fitting procedure is carried out ignoring species labels. We do not normalize the parameters related to the splines: the psis can be calculated afterwards.

Value

A list with components

- `taxonCoef`: The fitted coefficients of the sample-wise response curves
- `splinesList`: A list of all the B-spline objects
- `rowMar`: The row matrix
- `overall`: The overall fit ignoring taxon labels, as a list of coefficients and a spline
- `rowVecOverall`: The overall row vector, ignoring taxon labels
extractCoord  A function to extract plotting coordinates, either for plot.RCM or to export to other plotting software

Description

A function to extract plotting coordinates, either for plot.RCM or to export to other plotting software

Usage

extractCoord(RCM, Dim = c(1, 2))

Arguments

RCM  an RCM object

Dim  an integer vector of required dimensions

The parameters for the ellipses of the quadratic response function come from the parametrization \( f(x) = a*x^2 + b*x + c \) For an unconstrained object the row and column coordinates are returned in separate matrices. The row names will correspond to the labels. For a constrained analysis also the variable points are returned. All variables still need to be scaled to optimally fill the available space

Value

A list with components

samples  A dataframe of sample scores

species  A dataframe of column scores, with origin, slope, end and ellipse coordinates as needed

variables  A dataframe of variable scores, loadings of the environmental gradient

Examples

data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[1:100], prune_samples(sample_names(Zeller)[1:50], Zeller))
zellerRCM = RCM(tmpPhy, k = 2, round = TRUE)
coordsZeller = extractCoord(zellerRCM)
Filter out the effect of known confounders. This is done by fitting interactions of every taxon with the levels of the confounders. It returns a modified offset matrix for the remainder of the fitting procedure.

Description

Filters out the effect of known confounders. This is done by fitting interactions of every taxon with the levels of the confounders. It returns a modified offset matrix for the remainder of the fitting procedure.

Usage

filterConfounders(
  muMarg,
  confMat,
  X,
  thetas,
  p,
  n,
  nleqslv.control,
  trended.dispersion,
  tol = 0.001,
)
getDevianceRes

maxIt = 20,
allowMissingness,
naId
)

Arguments

muMarg a nxp matrix, the current offset
confMat a nxt confounder matrix
X the nxp data matrix
thetas a vector of length p with the current dispersion estimates
p an integer, the number of columns of X
n an integer, the number of rows of X
nleqslv.control see nleqslv()
trended.dispersion a vector of length p with trended dispersion estimates
tol a scalar, the convergence tolerance
maxIt maximum number of iterations
allowMissingness A boolean, are missing values present
naId The numeric index of the missing values in X

Fits the negative binomial mean parameters and overdispersion parameters iteratively. Convergence is determined based on the L2-norm of the absolute change of mean parameters

Value

a list with components:

thetas new theta estimates
NB_params The estimated parameters of the interaction terms

getDevianceRes A function to calculate the matrix of deviance residuals.

Description

A function to calculate the matrix of deviance residuals.

Usage

getDevianceRes(RCM, Dim = RCM$k)
getDevMat

Arguments

RCM an RCM object
Dim The dimensions to use
For the deviance residuals we use the overdispersions from the reduced model.
Standard dimensions used are only first and second, since these are also plotted

Value
A matrix with deviance residuals of the same size as the original data matrix

Examples

data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[1:120],
prune_samples(sample_names(Zeller)[1:75], Zeller))
# Subset for a quick fit
zellerRCM = RCM(tmpPhy, k = 2, round = TRUE, prevCutOff = 0.03)
devRes = getDevianceRes(zellerRCM)

getDevMat X Calculate the matrix of deviance residuals

Description
X Calculate the matrix of deviance residuals

Usage
getDevMat(X, thetaMat, mu)

Arguments
X the data matrix
thetaMat the matrix of dispersions
mu the matrix of means

Value
The matrix of deviance residuals
getInflCol

A function to extract the influence for a given parameter index

Description

A function to extract the influence for a given parameter index

Usage

getInflCol(score, InvJac, taxon)

Arguments

- score: a score matrix
- InvJac: The inverted jacobian
- taxon: The taxon name or index

Value

A matrix with all observations’ influence on the given taxon

getInflRow

Extract the influence of all observations on a given row score

Description

Extract the influence of all observations on a given row score

Usage

getInflRow(score, InvJac, sample)

Arguments

- score: the score function evaluated for every observation
- InvJac: The inverse jacobian
- sample: the row score or sample index

Value

A matrix with all observations’ influence on the row score
getInt

**Integrate the spline of an vgam object**

**Description**

Integrate the spline of an vgam object

**Usage**

```r
getInt(coef, spline, sampleScore, stop.on.error = FALSE, ...)
```

**Arguments**

- `coef`: A vector of coefficients
- `spline`: The cubic smoothing spline
- `sampleScore`: the observed environmental scores
- `stop.on.error`: see ?integrate
- `...`: additional arguments passed on to integrate()

**Value**

A scalar, the value of the integral

getLogLik

**Extract the logged likelihood of every count**

**Description**

Extract the logged likelihood of every count

**Usage**

```r
getLogLik(rcm, Dim)
```

**Arguments**

- `rcm`: an RCM object
- `Dim`: A vector of integers indicating which dimensions to take along, or Inf for the saturated model, or 0 for the independence model

**Value**

A matrix with logged likelihood of the size of the data matrix
getModelMat

A function to construct a model matrix of a certain degree

Description

A function to construct a model matrix of a certain degree

Usage

getModelMat(y, degree)

Arguments

y the variable
degree the degree

Value

A model matrix with degree+1 columns and as many rows as length(y)

ggetRowMat

Return a matrix of row scores

Description

Return a matrix of row scores

Usage

ggetRowMat(sampleScore, responseFun, NB_params, taxonCoef, spline)

Arguments

sampleScore a vector of length n with sample scores
responseFun a character string, the type of response function, either 'linear' or 'quadratic'
NB_params a v-by-p matrix of parameters of the response function
taxonCoef A vector of coefficients
spline The cubic smoothing spline

Multiplying the old offset with the exponent matrix times the importance parameter obtains the new one based on lower dimension

Value

a n-by-p matrix of scores
GramSchmidt

*Gram-Schmidt orthogonalization of vectors*

**Description**

Gram-Schmidt orthogonalization of vectors

**Usage**

```r
GramSchmidt(x, otherVecs, weights = rep(1, length(x)))
```

**Arguments**

- **x**: The vector that is to be orthogonalized
- **otherVecs**: a matrix; x is orthogonalized with respect to its rows
- **weights**: The weights used in the orthogonalization

**Value**

The orthogonalized vector

---

heq_nb

*Define linear equality constraints for env. gradient*

**Description**

Define linear equality constraints for env. gradient

**Usage**

```r
heq_nb(Alpha, alphaK, d, k, centMat, ...)
```

**Arguments**

- **Alpha**: the current estimate of the environmental gradient
- **alphaK**: a matrix with the environmental gradients of the lower dimensions
- **d**: an integer, the number of environmental variables, including dummies
- **k**: an integer, the current dimension
- **centMat**: a centering matrix
- **...**: further arguments for other methods, not needed in this one

The centering matrix centMat ensures that the parameters of the dummies of the same categorical variable sum to zero

**Value**

a vector of with current values of the constraints, should evolve to zeroes only
The jacobian of the linear equality constraints

Description
The jacobian of the linear equality constraints

Usage
heq_nb_jac(Alpha, alphaK, d, k, centMat, ...)

Arguments
- **Alpha**: the current estimate of the environmental gradient
- **alphaK**: a matrix with the environmental gradients of the lower dimensions
- **d**: an integer, the number of environmental variables, including dummies
- **k**: an integer, the current dimension
- **centMat**: a centering matrix
- **...**: further arguments for other methods, not needed in this one

Value
The jacobian matrix

Functions to indent the plot to include the entire labels

Description
Functions to indent the plot to include the entire labels

Usage
indentPlot(plt, xInd = 0, yInd = 0)

Arguments
- **plt**: a ggplot object
- **xInd**: a scalar or a vector of length 2, specifying the indentation left and right of the plot to allow for the labels to be printed entirely
- **yInd**: a a scalar or a vector of length 2, specifying the indentation top and bottom of the plot to allow for the labels to be printed entirely

Value
a ggplot object, squared
Calculate the log-likelihoods of all possible models

**Arguments**

- `rcm` an object of the RCM class

**Value**

A table with inertias, proportion inertia explained and cumulative proportion of inertia explained.

**Examples**

```r
data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[1:100],
prune_samples(sample_names(Zeller)[1:50], Zeller))
zellerRCM = RCM(tmpPhy, round = TRUE)
inertia(zellerRCM)
```

---

Jacobian of the constrained analysis with linear response function.

**Description**

Jacobian of the constrained analysis with linear response function.

**Usage**

```r
JacCol_constr(betas, X, reg, theta, muMarg, psi, allowMissingness, naId)
```
Arguments

- betas: a vector of v parameters of the response function of a single taxon
- X: the count vector of length n
- reg: a n-by-v matrix of regressors
- theta: The dispersion parameter of this taxon
- muMarg: offset of length n
- psi: a scalar, the importance parameter
- allowMissingness: A boolean, are missing values present
- naId: The numeric index of the missing values in X

Even though this approach does not imply normalization over the parameters of all taxa, it is very fast and they can be normalized afterwards.

Value

The jacobian, a square symmetric matrix of dimension v

---

**JacCol_constr_noLab**  
*The jacobian of the response function without taxon labels*

---

Description

The jacobian of the response function without taxon labels

Usage

```r
JacCol_constr_noLab(
  betas,
  X,
  reg,
  thetasMat,
  muMarg,
  psi,
  n,
  v,
  preFabMat,
  allowMissingness,
  naId
)
```
Arguments

- `betas`: a vector of regression parameters with length `v`
- `X`: the `n x p` data matrix
- `reg`: a matrix of regressors of dimension `n x v`
- `thetasMat`: A matrix of dispersion parameters
- `muMarg`: offset matrix of dimension `n x p`
- `psi`: a scalar, the importance parameter
- `n`: an integer, number of rows of `X`
- `v`: an integer, the number of parameters of the response function
- `prefabMat`: a prefabricated matrix
- `allowMissingness`: A boolean, are missing values present
- `naId`: The numeric index of the missing values in `X`

Value

The jacobian (a `v-by-v` matrix)

---

**liks**

*Calculate the log-likelihoods of all possible models*

Description

Calculate the log-likelihoods of all possible models

Usage

```r
liks(rcm, Sum = TRUE)
```

Arguments

- `rcm`: an object of the RCM class
- `Sum`: a boolean, should log-likelihoods be summed?

Value

If `Sum` is `FALSE`, a named array log-likelihoods of the independence model and all models with dimension 1 to `k`, including after filtering on confounders. Otherwise a table with log-likelihoods, deviance explained and cumulative deviance explained.
Examples

```r
data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[1:100],
  prune_samples(sample_names(Zeller)[1:50], Zeller))
zellerRCM = RCM(tmpPhy, round = TRUE)
liks(zellerRCM)
```

**LR_nb**

Get the value of the log-likelihood ratio of alpha

**Description**

Get the value of the log-likelihood ratio of alpha

**Usage**

```r
LR_nb(
  Alpha,
  X,
  CC,
  responseFun = c("linear", "quadratic", "nonparametric", "dynamic"),
  muMarg,
  psi,
  nleqslv.control = list(trace = FALSE),
  n,
  NB_params,
  NB_params_noLab,
  thetaMat,
  ncols,
  nonParamRespFun,
  envGradEst,
  ...
)
```

**Arguments**

- **Alpha**: a vector of length d, the environmental gradient
- **X**: the n-by-p count matrix
- **CC**: the n-by-d covariate matrix
- **responseFun**: a character string indicating the type of response function
- **muMarg**: an n-by-p offset matrix
- **psi**: a scalar, an importance parameter
- **nleqslv.control**: the control list for the nleqslv() function
LR_nb_Jac

A function that returns the Jacobian of the likelihood ratio

Description

A function that returns the Jacobian of the likelihood ratio

Usage

LR_nb_Jac(
  Alpha,
  X,
  CC,
  responseFun = c("linear", "quadratic", "nonparametric", "dynamic"),
  psi,
  NB_params,
  NB_params_noLab,
  d,
  alphaK,
  k,
  centMat,
  nLambda,
  nLambdas,
  thetaMat,
  muMarg,
  n,
Arguments

Alpha
a vector of length $d + k*(2+(k-1)/2)$, the environmental gradient plus the lagrangian multipliers

X
the n-by-p count matrix

CC
a n-by-d covariate vector

responseFun
a character string indicating the type of response function

psi
a scalar, an importance parameter

NB_params
Starting values for the NB_params

NB_params_noLab
Starting values for the NB_params without label

d
an integer, the number of covariate parameters

alphaK
a matrix of environmental gradients of lower dimensions

k
an integer, the current dimension

centMat
a nLambda1s-by-d centering matrix

nLambda
an integer, number of lagrangian multipliers

nLambda1s
an integer, number of centering restrictions

thetaMat
a matrix of size n-by-p with estimated dispersion parameters

muMarg
an n-by-p offset matrix

n
an integer, the number of rows of X

ncols
a scalar, the number of columns of X

preFabMat
a prefabricated matrix

envGradEst
a character string, indicating how the environmental gradient should be fitted. 'LR' using the likelihood-ratio criterion, or 'ML' a full maximum likelihood solution

allowMissingness
A boolean, are missing values present

naId
The numeric index of the missing values in X

... Further arguments passed on to other functions

Value

A symmetric matrix, the evaluated Jacobian
NBalphaInfl  
*Calculate the components of the influence functions*

**Description**

Calculate the components of the influence functions

**Usage**

```
NBalphaInfl(rcm, Dim)
```

**Arguments**

- `rcm`: an `rcm` object
- `Dim`: the required dimension

**Value**

An n-by-p-by-d array with the influence of every observation on every alpha parameter

--

NBcolInfl  
*The influence function for the column scores*

**Description**

The influence function for the column scores

**Usage**

```
NBcolInfl(rcm, Dim = 1)
```

**Arguments**

- `rcm`: an `rcm` object
- `Dim`: the required dimension

**Value**

A list with components

- `score`: a matrix with components of the score function
- `InvJac`: A square matrix of dimension p with the components of the Jacobian related to the column scores
NBjacobianAbundsOld  Jacobian for the column components of the independence model

Description
Jacobian for the column components of the independence model

Usage
NBjacobianAbundsOld(beta, X, reg, thetas, allowMissingness, naId)

Arguments
- beta: a vector of length p with current abundance estimates
- X: a n-by-p count matrix
- reg: a vector of length n with library sizes estimates
- thetas: a n-by-p matrix with overdispersion estimates in the rows
- allowMissingness: A boolean, are missing values present
- naId: The numeric index of the missing values in X

Value
a diagonal matrix of dimension p with evaluations of the jacobian function

NBjacobianColNP  Jacobian function for the estimation of a third degree GLM

Description
Jacobian function for the estimation of a third degree GLM

Usage
NBjacobianColNP(beta, X, reg, theta, muMarg)

Arguments
- beta: vector of any length
- X: the data vector of length n
- reg: a nxlength(beta) regressor matrix
- theta: a scalar, the overdispersion
- muMarg: the offset of length n
Value

A matrix of dimension 8-by-8

Description

Jacobian for the estimation of the column scores

Usage

\[
\text{NBjacobianColOld}(\beta, X, \text{reg}, \text{thetas}, \text{muMarg}, k, n, p, \text{colWeights}, n\text{Lambda}, c\text{MatK}, \text{preFabMat}, \text{Jac}, \text{allowMissingness}, \text{naId})
\]

Arguments

- **beta**: vector of length \(p+1+1+(k-1)\): \(p\) row scores, 1 centering, one normalization and \((k-1)\) orthogonality lagrangian multipliers
- **X**: the \(nxp\) data matrix
- **reg**: a \(nx1\) regressor matrix: outer product of rowScores and psis
- **thetas**: \(nxp\) matrix with the dispersion parameters (converted to matrix for numeric reasons)
- **muMarg**: the \(nxp\) offset
- **k**: an integer, the dimension of the RC solution
- **n**: an integer, the number of samples
- **p**: an integer, the number of taxa
- **colWeights**: the weights used for the restrictions
- **nLambda**: an integer, the number of restrictions
NBjacobianLibSizes

---

**Description**

Jacobian for the raw components of the independence model

**Usage**

```r
NBjacobianLibSizes(beta, X, reg, thetas, allowMissingness, naId)
```

**Arguments**

- `beta`: a vector of length n with current library size estimates
- `X`: a n-by-p count matrix
- `reg`: a vector of length p with relative abundance estimates
- `thetas`: a n-by-p matrix with overdispersion estimates in the rows
- `allowMissingness`: A boolean, are missing values present
- `naId`: The numeric index of the missing values in X

**Value**

a diagonal matrix of dimension n: the Fisher information matrix
NBjacobianPsi

*Description*

Jacobian for the psi of a given dimension

*Usage*

```
NBjacobianPsi(beta, X, reg, muMarg, theta, preFabMat, allowMissingness, naId)
```

*Arguments*

- **beta**: a scalar, the current estimate
- **X**: the n-by-p count matrix
- **reg**: the regressor matrix, the outer product of current row and column scores
- **muMarg**: the nxp offset matrix
- **theta**: a n-by-p matrix with the dispersion parameters
- **preFabMat**: a prefab matrix, (1+X/thetas)
- **allowMissingness**: A boolean, are missing values present
- **naId**: The numeric index of the missing values in X

*Value*

The evaluation of the jacobian function at beta, a 1-by-1 matrix

NBjacobianRow

*Description*

A jacobian function of the NB for the row scores

*Usage*

```
NBjacobianRow(
  beta, 
  X, 
  reg, 
  thetas, 
  muMarg, 
  k,
)
Arguments

- **beta**: a vector of length \( n + k + 1 \) regression parameters to optimize
- **X**: the data matrix of dimensions \( nxp \)
- **reg**: a \( 1 \times p \) regressor matrix: outer product of column scores and psis
- **thetas**: \( nxp \) matrix with the dispersion parameters (converted to matrix for numeric reasons)
- **muMarg**: an \( nxp \) offset matrix
- **k**: a scalar, the dimension of the RC solution
- **n**: a scalar, the number of samples
- **p**: a scalar, the number of taxa
- **rowWeights**: a vector of length \( n \), the weights used for the restrictions
- **nLambda**: an integer, the number of lagrangian multipliers
- **rMatK**: the lower dimension row scores
- **prefFabMat**: a prefab matrix, \((1 + X/thetas)\)
- **Jac**: an empty Jacobian matrix
- **allowMissingness**: A boolean, are missing values present
- **naId**: The numeric index of the missing values in \( X \)

Value

- a symmetric jacobian matrix of size \( n+k+1 \)

Description

The influence function for the psis
**Usage**

`NBpsiInfl(rcm, Dim = 1)`

**Arguments**

- `rcm` an `rcm` object
- `Dim` the required dimensions

**Value**

The influence of every single observation on the psi value of this dimension

---

**Description**

The influence function for the row scores

**Usage**

`NBrowInfl(rcm, Dim = 1)`

**Arguments**

- `rcm` an `rcm` object
- `Dim` the required dimension

**Value**

A list with components

- `score` a matrix with components of the score function
- `InvJac` A square matrix of dimension n with the components of the Jacobian related to the row scores
plot.RCM

Plot RC(M) ordination result with the help of ggplot2

Description

Plot RC(M) ordination result with the help of ggplot2

Usage

## S3 method for class 'RCM'
plot(
x, 
..., 
Dim = c(1, 2), 
plotType = c("samples", "species", "variables"), 
samColour = if (is.null(inflVar)) NULL else "Influence", 
taxNum = if (all(plotType == "species") || !is.null(taxRegExp)) { ncol(x$X) } 
else { 10 }, 
taxRegExp = NULL, 
varNum = 15, 
arrowSize = 0.25, 
inflDim = 1, 
inflVar = NULL, 
returnCoords = FALSE, 
alpha = TRUE, 
varPlot = NULL, 
colLegend = if (!is.null(inflVar)) paste0("Influence on\n", inflVar, 
  "\nparameter \nin dimension", inflDim) else samColour, 
samShape = NULL, 
shapeLegend = samShape, 
samSize = 2, 
scalingFactor = NULL, 
quadDrop = 0.995, 
plotEllipse = TRUE, 
taxaScale = 0.5, 
Palette = if (!all(plotType == "species")) "Set1" else "Paired", 
taxLabels = !all(plotType == "species"), 
taxDots = FALSE, 
taxCol = "blue", 
taxColSingle = "blue", 
nudge_y = 0.08, 
axesFixed = TRUE, 
aspRatio = 1, 
xInd = if (all(plotType == "samples")) c(0, 0) else c(-0.75, 0.75), 
yInd = c(0, 0), 
taxLabSize = 4, 
varLabSize = 3.5,
alphaRange = c(0.2, 1),
varExpFactor = 10,
manExpFactorTaxa = 0.975,
nPhyl = 10,
phy1Other = c(""),
legendSize = samSize,
noLegend = is.null(samColour),
crossSize = 4,
contCol = c("orange", "darkgreen"),
legendLabSize = 15,
legendTitleSize = 16,
axisLabSize = 14,
axisTitleSize = 16,
plotPsi = "psi",
brackChar = "\n"
)

Arguments

x an RCM object

... further arguments, passed on to aes in the ggplot() function

Dim An integer vector of length two, which dimensions to plot

plotType a character string: which components to plot. Can be any combination of 'samples','species' and 'variables'

samColour a character string, the variable to use for the colour of the sample dots. Can also be a richness measure, or "influence". Alternatively, a vector equal to the number of samples in the RCM object can be supplied. See details.

taxNum an integer, the number of taxa to be plotted

taxRegExp a character vector indicating which taxa to plot. Any taxa matching this regular expression will be plotted

varNum an integer, number of variable arrows to draw

arrowSize a scalar, the size of the arrows

inflDim an integer, the dimension for which the influence should be calculated

inflVar the variable on which the influence should be plotted. See details.

returnCoords a boolean, should final coordinates be returned?

alpha a boolean, should small arrows be made transparent?

varPlot the names of the variable arrows to plot. Overrides the varNum argument

colLegend a character string, the legend text for the sample colour. Defaults to the name of the colour variable

samShape a character string, the variable to use for the shape of the sample dots

shapeLegend a character string, the text to use for the shapeLegend. Defaults to the name of the shape variable

samSize a scalar, the size of the sample dots
scalingFactor  a scalar, a user supplied scaling factor for the taxon arrows. If not supplied it will be calculated to make sample and taxon plots on the same scale
quadDrop  a number between 0 and 1. At this fraction of the peak height are the ellipses of the quadratic response functions drawn
plotEllipse  a boolean, whether to add the ellipses
taxaScale  a scalar, by which to scale the rectangles of the quadratic taxon plot
Palette  the colour palette
taxLabels  a boolean, should taxon labels be plotted?
taxDots  a boolean, should taxa be plotted as dots?
taxCol  the taxon colour
taxColSingle  the taxon colour if there is only one
nudge_y  a scalar, the offset for the taxon labels
axesFixed  A boolean, should the aspect ratio of the plot (the scale between the x and y-axis) be fixed. It is highly recommended to keep this argument at TRUE for honest representation of the ordination. If set to FALSE, the plotting space will be optimally used but the plot may be deformed in the process.
aspRatio  The aspect ratio of the plot when 'axesfixed' is TRUE (otherwise this argument is ignored), passed on to ggplot2::coord_fixed(). It is highly recommended to keep this argument at 1 for honest representation of the ordination.
xInd  a scalar or a vector of length 2, specifying the indentation left and right of the plot to allow for the labels to be printed entirely. Defaults to 0.75 at every side
yInd  a scalar or a vector of length 2, specifying the indentation top and bottom of the plot to allow for the labels to be printed entirely. Defaults to 0 at every side
taxLabSize  the size of taxon labels
varLabSize  the size of the variable label
alphaRange  The range of transparency
varExpFactor  a scalar, the factor by which to expand the variable coordinates
manExpFactorTaxa  a manual expansion factor for the taxa. Setting it to a high value allows you to plot the taxa around the samples
nPhyl  an integer, number of phylogenetic levels to show
phylOther  a character vector of phylogenetic levels to be included in the 'other' group
legendSize  a size for the coloured dots in the legend
noLegend  a boolean indicating you do not want a legend
crossSize  the size of the central cross
contCol  a character vector of length two, giving the low and high values of the continuous colour scale
legendLabSize  size of the legend labels
legendTitleSize  size of the legend title
axisLabSize    size of the axis labels
axisTitleSize size of the axis title
plotPsi       a character vector, describing what to plot on the axis. Can be either 'psi', 'none' or 'loglik'. The latter plots the log-likelihood explained
breakChar    a character string indicating how the taxon names should be broken

Details

This function relies on the ggplot2 machinery to produce the plots, and the result can be modified accordingly. Monoplots, biplots and for constrained analysis even triplots can be produced, depending on the 'plotType' argument.

When one of either 'Observed', 'Chao1', 'ACE', 'Shannon', 'Simpson', 'InvSimpson' or 'Fisher' are supplied to the 'samColour' argument, the according richness measure (as calculated by phyloseq::estimate_richness) is mapped to the sample colour. When "influence" is supplied, the influence on the variable supplied is plotted. This 'inflVar' variable should be either "psi", or a variable name.

Value

plots a ggplot2-object to output

Note

Supplying only few categorical variables as constraining variables may cause the samples to be plotted on top of each other, since the number of unique sample scores is limited. The plot is still valid, but consider adding more sample variables to spread out the samples

See Also

RCM, addOrthProjection, extractCoord, plotRespFun

Examples

data(Zeller)
require(phylloseq)
.tmpPhy = prune_taxa(taxa_names(Zeller)[1:100],
  prune_samples(sample_names(Zeller)[1:50], Zeller))
# Subset for a quick fit
zellerRCM = RCM(tmpPhy)
plot(zellerRCM)
Description

Plots a number of response functions over the observed range of the environmental score. If no taxa are provided those who react most strongly to the environmental score are chosen.

Usage

plotRespFun(
  RCM,
  taxa = NULL,
  type = "link",
  logTransformYAxis = FALSE,
  addSamples = TRUE,
  samSize = NULL,
  Dim = 1L,
  nPoints = 100L,
  labSize = 2.5,
  yLocVar = NULL,
  yLocSam = NULL,
  Palette = "Set3",
  addJitter = FALSE,
  nTaxa = 9L,
  angle = 90,
  legendLabSize = 15,
  legendTitleSize = 16,
  axisLabSize = 14,
  axisTitleSize = 16,
  lineSize = 0.75,
  ...
)

Arguments

RCM: an RCM object

taxa: a character vector of taxa to be plotted

type: a character string, plot the response function on the log-scale ('link') or the abundance scale 'response', similar to predict.glm().

logTransformYAxis: a boolean, should y-axis be log transformed?

addSamples: a boolean, should sample points be shown?

samSize: a sample variable name or a vector of length equal to the number of samples, for the sample sizes
plotRespFun

Dim
An integer, the dimension to be plotted

nPoints
the number of points to be used to plot the lines

labSize
the label size for the variables

yLocVar
the y-location of the variables, recycled if necessary

yLocSam
the y-location of the samples, recycled if necessary

Palette
which color palette to use

addJitter
A boolean, should variable names be jittered to make them more readable

nTaxa
an integer, number of taxa to plot

angle
angle at which variable labels should be turned

legendLabSize
size of the legend labels

legendTitleSize
size of the legend title

axisLabSize
size of the axis labels

axisTitleSize
size of the axis title

lineSize
size of the response function lines

... Other arguments passed on to the ggplot() function

Value
Plots a ggplot2-object to output

See Also
RCM, plot.RCM, residualPlot

Examples

data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[1:100],
prune_samples(sample_names(Zeller)[1:50], Zeller))
# Subset for a quick fit
zellerRCMnp = RCM(tmpPhy, k = 2,
covariates = c('BMI', 'Age', 'Country', 'Diagnosis', 'Gender'),
round = TRUE, responseFun = 'nonparametric')
plotRespFun(zellerRCMnp)
**Description**

This is a wrapper function, which currently only fits the negative binomial distribution, but which could easily be extended to other ones.

**Usage**

```
RCM(dat, ...)  
## S4 method for signature 'phyloseq'
RCM(dat, covariates = NULL, confounders = NULL, ...)

## S4 method for signature 'matrix'
RCM(
  dat,
  k = 2,
  round = FALSE,
  prevCutOff = 0.05,
  minFraction = 0.1,
  rowWeights = "uniform",
  colWeights = "marginal",
  confModelMat = NULL,
  confTrimMat = NULL,
  covModelMat = NULL,
  centMat = NULL,
  allowMissingness = FALSE,
 ...)
```

**Arguments**

- **dat**
  - an nxp count matrix or a phyloseq object with an otu_table slot
- **...**
  - Further arguments passed on to the RCM.NB() function
- **covariates**
  - In case 'dat' is a phyloseq object, the names of the sample variables to be used as covariates in the constrained analysis, or 'all' to indicate all variables to be used. In case 'dat' is a matrix, a nxf matrix or dataframe of covariates. Character variables will be converted to factors, with a warning. Defaults to NULL, in which case an unconstrained analysis is carried out.
- **confounders**
  - In case 'dat' is a phyloseq object, the names of the sample variables to be used as confounders to be filtered out. In case 'dat' is a matrix, a nxf dataframe of confounders. Character variables will be converted to factors, with a warning. Defaults to NULL, in which case no filtering occurs.
- **k**
  - an integer, the number of dimensions of the RCM solution
round    a boolean, whether to round to nearest integer. Defaults to FALSE.
prevCutOff a scalar, the prevalence cutoff for the trimming. Defaults to 2.5e-2
minFraction a scalar, each taxon’s total abundance should equal at least the number of samples n times minFraction, otherwise it is trimmed. Defaults to 10%
rowWeights, colWeights character strings, the weighting procedures for the normalization of row and column scores. Defaults to ‘uniform’ and ‘marginal’ respectively
confTrimMat, confModelMat, covModelMat, centMat Dedicated model matrices constructed based on phyloseq object.
allowMissingness A boolean, should NA values be tolerated?

Details

This function should be called on a raw count matrix, without rarefying or normalization to proportions. This function trims on prevalence and total abundance to avoid instability of the algorithm. Covariate and confounder matrices are constructed, so that everything is passed on to the workhorse function RCM.NB() as matrices.

Value

see RCM_NB

See Also

RCM_NB, plot.RCM, residualPlot, plotRespFun

Examples

data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[1:100],
prune_samples(sample_names(Zeller)[1:50], Zeller))
zellerRCM = RCM(tmpPhy, round = TRUE)
Usage

```r
RCM_NB(
  X,
  k,
  rowWeights = "uniform",
  colWeights = "marginal",
  tol = 0.001,
  maxItOut = 1000L,
  Psitol = 0.001,
  verbose = FALSE,
  global = "dbldog",
  nleqslv.control = list(maxit = 500L, cndtol = 1e-16),
  jacMethod = "Broyden",
  dispFreq = 10L,
  convNorm = 2,
  prior.df = 10,
  marginEst = "MLE",
  confModelMat = NULL,
  confTrimMat = NULL,
  prevCutOff,
  minFraction = 0.1,
  covModelMat = NULL,
  centMat = NULL,
  responseFun = c("linear", "quadratic", "dynamic", "nonparametric"),
  record = FALSE,
  control.outer = list(trace = FALSE),
  control.optim = list(),
  envGradEst = "LR",
  dfSpline = 3,
  vgamMaxit = 100L,
  degree = switch(responseFun[1], nonparametric = 3, NULL),
  rowExp = if (is.null(covModelMat)) 1 else 0.5,
  colExp = rowExp,
  allowMissingness = FALSE
)
```

Arguments

- **X**: a n x p data matrix
- **k**: an scalar, number of dimensions in the RC(M) model
- **rowWeights**: a character string, either 'uniform' or 'marginal' row weights.
- **colWeights**: a character string, either 'uniform' or 'marginal' column weights.
- **tol**: a scalar, the relative convergende tolerance for the row scores and column scores parameters.
- **maxItOut**: an integer, the maximum number of iterations in the outer loop.
- **Psitol**: a scalar, the relative convergence tolerance for the psi parameters.
verbose: a boolean, should information on iterations be printed?

global: global strategy for solving non-linear systems, see \texttt{nleqslv}

\texttt{nleqslv\_control}: a list with control options, see \texttt{nleqslv}

\texttt{jacMethod}: Method for solving non-linear equations, \texttt{?see nleqslv}. Defaults to Broyden. The difference with the newton method is that the Jacobian is not recalculated at every iteration, thereby speeding up the algorithm

\texttt{dispFreq}: an integer, how many iterations the algorithm should wait before reestimating the dispersions.

\texttt{convNorm}: a scalar, the norm to use to determine convergence

\texttt{prior\_df}: an integer, see \texttt{estDisp()}

\texttt{marginEst}: a character string, either \texttt{‘MLE’} or \texttt{‘marginSums’}, indicating how the independence model should be estimated

\texttt{confModelMat}: an \texttt{n}x\texttt{g} matrix with confounders, with no reference levels and with intercept

\texttt{confTrimMat}: an \texttt{n}x\texttt{h} matrix with confounders for filtering, with all levels and without intercept

\texttt{prevCutOff}: a scalar the minimum prevalence needed to retain a taxon before the the confounder filtering

\texttt{minFraction}: a scalar, total taxon abundance should equal \texttt{minFraction} \times \texttt{n} if it wants to be retained before the confounder filtering

\texttt{covModelMat}: an \texttt{n}x\texttt{d} matrix with covariates. If set to null an unconstrained analysis is carried out, otherwise a constrained one. Factors must have been converted to dummy variables already

\texttt{centMat}: a \texttt{fxd} matrix containing the contrasts to center the categorical variables. \texttt{f} equals the number of continuous variables + the total number of levels of the categorical variables.

\texttt{responseFun}: a character string indicating the shape of the response function

\texttt{record}: A boolean, should intermediate parameter estimates be stored?

\texttt{control\_outer}: a list of control options for the outer loop \texttt{constrOptim.nl} function

\texttt{control\_optim}: a list of control options for the \texttt{optim()} function

\texttt{envGradEst}: a character string, indicating how the environmental gradient should be fitted. \texttt{‘LR’} using the likelihood-ratio criterion, or \texttt{‘ML’} a full maximum likelihood solution

\texttt{dfSpline}: a scalar, the number of degrees of freedom for the splines of the non-parametric response function, see \texttt{VGAM::s()}

\texttt{vgam\_Maxit}: an integer, the maximum number of iteration in the \texttt{vgam()} function

\texttt{degree}: an integer, the degree of the polynomial fit if the spline fit fails

\texttt{rowExp}, \texttt{colExp}: exponents for the row and column weights of the singular value decomposition used to calculate starting values. Can be played around with in case of numerical troubles.

\texttt{allow\_Missingness}: See \texttt{RCM()}

\texttt{allow\_Missingness}
Details
Includes fitting of the independence model, filtering out the effect of confounders and fitting the RC(M) components in a constrained or an unconstrained way for any dimension k. Not intended to be called directly but only through the RCM() function.

Value
A list with elements:

- **converged**: a vector of booleans of length k indicating if the algorithm converged for every dimension.
- **rMat**: if not constrained a nxk matrix with estimated row scores.
- **cMat**: a kxp matrix with estimated column scores.
- **psis**: a vector of length k with estimates for the importance parameters psi.
- **thetas**: a vector of length p with estimates for the overdispersion.
- **rowRec**: (if not constrained) a n x k x maxItOut array with a record of all rMat estimates through the iterations.
- **colRec**: a k x p x maxItOut array with a record of all cMat estimates through the iterations.
- **psiRec**: a k x maxItOut array with a record of all psi estimates through the iterations.
- **thetaRec**: a matrix of dimension pxmaxItOut with estimates for the overdispersion along the way.
- **iter**: number of iterations.
- **Xorig**: (if confounders provided) the original fitting matrix.
- **X**: the trimmed matrix if confounders provided, otherwise the original one.
- **fit**: type of fit, either 'RCM_NB' or 'RCM_NB_constr'.
- **lambdaRow**: (if not constrained) vector of Lagrange multipliers for the rows.
- **lambdaCol**: vector of Lagrange multipliers for the columns.
- **rowWeights**: (if not constrained) the row weights used.
- **colWeights**: the column weights used.
- **alpha**: (if constrained) the kxd matrix of environmental gradients.
- **alphaRec**: (if constrained) the kdxmaxItOut array of alpha estimates along the iterations.
- **covariates**: (if constrained) the matrix of covariates.
- **libSizes**: a vector of length n with estimated library sizes.
- **abunds**: a vector of length p with estimated mean relative abundances.
- **confounders**: (if provided) the confounder matrix.
- **confParams**: the parameters used to filter out the confounders.
- **nonParamRespFun**: A list of the non parametric response functions.
- **degree**: The degree of the alternative parametric fit.
- **NApresent**: A boolean, were NA values present?
Note
Plotting is not supported for quadratic response functions

See Also
RCM

Examples
data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[seq_len(100)],
prune_samples(sample_names(Zeller)[seq_len(50)], Zeller))
mat = as(otu_table(tmpPhy), "matrix")
mat = mat[rowSums(mat)>0, colSums(mat)>0]
zellerRCM = RCM_NB(mat, k = 2)
#Needs to be called directly onto a matrix

---

residualPlot

Make residual plots

Description
Make residual plots

Usage
residualPlot(
  RCM,
  Dim = 1,
  whichTaxa = "response",
  resid = "Deviance",
  numTaxa = 9,
  mfrow = NULL,
  samColour = NULL,
  samShape = NULL,
  legendLabSize = 15,
  legendTitleSize = 16,
  axisLabSize = 14,
  axisTitleSize = 16,
  taxTitle = TRUE,
  h = 0
)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCM</td>
<td>an RCM object</td>
</tr>
<tr>
<td>Dim</td>
<td>an integer, which dimension?</td>
</tr>
<tr>
<td>whichTaxa</td>
<td>a character string or a character vector, for which taxa to plot the diagnostic plots</td>
</tr>
<tr>
<td>resid</td>
<td>the type of residuals to use, either ‘Deviance’ or ’Pearson’</td>
</tr>
<tr>
<td>numTaxa</td>
<td>an integer, the number of taxa to plot</td>
</tr>
<tr>
<td>mfrow</td>
<td>passed on to par(). If not supplied will be calculated based on numTaxa</td>
</tr>
</tbody>
</table>
| samColour, samShape | Vectors or character strings denoting the sample colour and shape respectively. 
                        | If character string is provided, the variables with this name is extracted from the 
                        | phyloseq object in RCM                                                      |
| legendLabSize | size of the legend labels                                                    |
| legendTitleSize | size of the legend title                                                     |
| axisLabSize | size of the axis labels                                                     |
| axisTitleSize | size of the axis title                                                      |
| taxTitle   | A boolean, should taxon title be printed                                     |
| h          | Position of reference line. Set to NA for no line                            |

Details

If whichTaxa is ‘run’ or ‘response’ the taxa with the highest run statistics or steepest slopes of the response function are plotted, numTax indicates the number. If whichTaxa is a character vector, these are interpreted as taxon names to plot. This function is mainly meant for linear response functions, but can be used for others too. The runs test statistic from the tseries package is used.

Value

Plots a ggplot2-object to output

See Also

RCM

Examples

data(Zeller)
require(phyloseq)
tmpPhy = prune_taxa(taxa_names(Zeller)[1:120],
prune_samples(sample_names(Zeller)[1:75], Zeller))
#Subset for a quick fit
zellerRCMlin = RCM(tmpPhy, k = 2,
covariates = c('BMI','Age','Country','Diagnosis','Gender'),
responseFun = 'linear', round = TRUE, prevCutOff = 0.03)
residualPlot(zellerRCMlin)
respFunJacMat

Calculates the Jacobian of the parametric response functions

Description

Calculates the Jacobian of the parametric response functions

Usage

respFunJacMat(
  betas, 
  X, 
  reg, 
  thetaMat, 
  muMarg, 
  psi, 
  v, 
  p, 
  IDmat, 
  IndVec, 
  allowMissingness, 
  naId
)

Arguments

betas a vector of length (deg+1)*(p+1) with regression parameters with deg the degree of the response function and the lagrangian multipliers
X the n*p data matrix
reg a vector of regressors with the dimension n-by-v
thetaMat The n-by-p matrix with dispersion parameters
muMarg offset matrix of size n*p
psi a scalar, the importance parameter
v an integer, one plus the degree of the response function
p an integer, the number of taxa
IDmat an logical matrix with indices of non-zero elements
IndVec a vector with indices with non-zero elements
allowMissingness A boolean, are missing values present
naId The numeric index of the missing values in X

Value

The jacobian, a square matrix of dimension (deg+1)*(p+1)
respFunScoreMat

Derivative of the Lagrangian of the parametric response function

**Description**

Derivative of the Lagrangian of the parametric response function

**Usage**

```r
respFunScoreMat(
  betas,
  X,
  reg,
  thetaMat,
  muMarg,
  psi,
  p,
  v,
  allowMissingness,
  naId,
  ...
)
```

**Arguments**

- `betas` a vector of length (deg+1)*(p+1) with regression parameters with deg the degree of the response function and the lagrangian multipliers
- `X` the nxp data matrix
- `reg` a matrix of regressors with the dimension nx(deg+1)
- `thetaMat` The n-by-p matrix with dispersion parameters
- `muMarg` offset matrix of size nxp
- `psi` a scalar, the importance parameter
- `p` an integer, the number of taxa
- `v` an integer, one plus the degree of the response function
- `allowMissingness` A boolean, are missing values present
- `naId` The numeric index of the missing values in X
- `...` further arguments passed on to the jacobian

The parameters are restricted to be normalized, i.e. all squared intercepts, first order and second order parameters sum to 1

**Value**

The evaluation of the score functions, a vector of length (p+1)* (deg+1)
**rowMultiply**

*Description*

A function to efficiently row multiply a matrix and a vector

*Usage*

```r
rowMultiply(matrix, vector)
```

*Arguments*

- **matrix**: a numeric matrix of dimension a-by-b
- **vector**: a numeric vector of length b
  
  `t(t(matrix))%*%vector` but then faster

*Details*

Memory intensive but that does not matter with given matrix sizes

*Value*

a matrix, row multiplied by the vector

---

**seq_k**

*Description*

A small auxiliary function for the length of the lambdas

*Usage*

```r
seq_k(y, nLambda1s = 1)
```

*Arguments*

- **y**: an integer, the current dimension
- **nLambda1s**: the number of centering restrictions

*Value*

a vector containing the ranks of the current lagrangian multipliers
trimOnConfounders

Description
Trim based on confounders to avoid taxa with only zero counts

Usage
trimOnConfounders(confounders, X, prevCutOff, minFraction, n)

Arguments
- confounders: a ntx confounder matrix
- X: the nxp data matrix
- prevCutOff: a scalar between 0 and 1, the prevalence cut off
- minFraction: a scalar between 0 and 1, each taxon’s total abundance should equal at least the number of samples n times minFraction, otherwise it is trimmed
- n: the number of samples
Should be called prior to fitting the independence model

Value
A trimmed data matrix nxp'

Zeller

Microbiomes of colorectal cancer patients and healthy controls

Description
Microbiome sequencing data of colorectal cancer patients, patients with small adenoma and healthy controls, together with other baseline covariates

Usage
Zeller

Format
A phyloseq object with an OTU-table and sample data
- otu_table: Count data matrix of 709 taxa in 194 samples
- sample_data: Data frame of patient covariates

Source
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4299606/
## Index

* datasets  
  Zeller, 63

<table>
<thead>
<tr>
<th>Function</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>addOrthProjection</td>
<td>3, 50</td>
</tr>
<tr>
<td>arrayProd</td>
<td>4</td>
</tr>
<tr>
<td>buildCentMat</td>
<td>5</td>
</tr>
<tr>
<td>buildConfMat</td>
<td>5</td>
</tr>
<tr>
<td>buildConfMat.character</td>
<td>6</td>
</tr>
<tr>
<td>buildConfMat.data.frame</td>
<td>6</td>
</tr>
<tr>
<td>buildCovMat</td>
<td>7</td>
</tr>
<tr>
<td>buildDesign</td>
<td>8</td>
</tr>
<tr>
<td>checkAlias</td>
<td>8</td>
</tr>
<tr>
<td>constrCorresp</td>
<td>9</td>
</tr>
<tr>
<td>correctXMissingness</td>
<td>10</td>
</tr>
<tr>
<td>deviances</td>
<td>10</td>
</tr>
<tr>
<td>dLR_nb</td>
<td>11</td>
</tr>
<tr>
<td>dNBabundsOld</td>
<td>12</td>
</tr>
<tr>
<td>dNBlibSizes</td>
<td>13</td>
</tr>
<tr>
<td>dNB1col_constr</td>
<td>16</td>
</tr>
<tr>
<td>dNB1col_constr_noLab</td>
<td>16</td>
</tr>
<tr>
<td>dNB1colNP</td>
<td>14</td>
</tr>
<tr>
<td>dNB1colOld</td>
<td>14</td>
</tr>
<tr>
<td>dNB1row</td>
<td>17</td>
</tr>
<tr>
<td>dNBpsis</td>
<td>18</td>
</tr>
<tr>
<td>ellipseCoord</td>
<td>19</td>
</tr>
<tr>
<td>estDisp</td>
<td>20</td>
</tr>
<tr>
<td>estNBparams</td>
<td>21</td>
</tr>
<tr>
<td>estNBparamsNoLab</td>
<td>22</td>
</tr>
<tr>
<td>estNPresp</td>
<td>23</td>
</tr>
<tr>
<td>extractCoord</td>
<td>25, 50</td>
</tr>
<tr>
<td>extractE</td>
<td>26</td>
</tr>
<tr>
<td>filterConfounders</td>
<td>26</td>
</tr>
<tr>
<td>getDevianceRes</td>
<td>27</td>
</tr>
<tr>
<td>getDevMat</td>
<td>28</td>
</tr>
<tr>
<td>getInfCol</td>
<td>29</td>
</tr>
<tr>
<td>getInfRow</td>
<td>29</td>
</tr>
<tr>
<td>getInt</td>
<td>30</td>
</tr>
<tr>
<td>getLogLik</td>
<td>30</td>
</tr>
<tr>
<td>getModelMat</td>
<td>31</td>
</tr>
<tr>
<td>getRowMat</td>
<td>31</td>
</tr>
<tr>
<td>GramSchmidt</td>
<td>32</td>
</tr>
<tr>
<td>heq_nb</td>
<td>32</td>
</tr>
<tr>
<td>heq_nb_jac</td>
<td>33</td>
</tr>
<tr>
<td>indentPlot</td>
<td>33</td>
</tr>
<tr>
<td>inertia</td>
<td>34</td>
</tr>
<tr>
<td>JacCol_constr</td>
<td>34</td>
</tr>
<tr>
<td>JacCol_constr_noLab</td>
<td>35</td>
</tr>
<tr>
<td>liks</td>
<td>36</td>
</tr>
<tr>
<td>LR_nb</td>
<td>37</td>
</tr>
<tr>
<td>LR_nb_Jac</td>
<td>38</td>
</tr>
<tr>
<td>NBAlphaInf1</td>
<td>40</td>
</tr>
<tr>
<td>NBcolInf1</td>
<td>40</td>
</tr>
<tr>
<td>NBjacobianAbundsOld</td>
<td>41</td>
</tr>
<tr>
<td>NBjacobianColNP</td>
<td>41</td>
</tr>
<tr>
<td>NBjacobianColOld</td>
<td>42</td>
</tr>
<tr>
<td>NBjacobianLibSizes</td>
<td>43</td>
</tr>
<tr>
<td>NBjacobianPsi</td>
<td>44</td>
</tr>
<tr>
<td>NBjacobianRow</td>
<td>44</td>
</tr>
<tr>
<td>NBpsiInf1</td>
<td>45</td>
</tr>
<tr>
<td>NBrowInf1</td>
<td>46</td>
</tr>
<tr>
<td>plot.RCM</td>
<td>4, 47, 52, 54</td>
</tr>
<tr>
<td>plotRespFun</td>
<td>50, 51, 54</td>
</tr>
<tr>
<td>RCM, matrix-method (RCM)</td>
<td>53</td>
</tr>
<tr>
<td>RCM, phyloseq-method (RCM)</td>
<td>53</td>
</tr>
<tr>
<td>RCM NB</td>
<td>54, 54</td>
</tr>
<tr>
<td>residualPlot</td>
<td>52, 54, 58</td>
</tr>
<tr>
<td>respFunJacMat</td>
<td>60</td>
</tr>
<tr>
<td>respFunScoreMat</td>
<td>61</td>
</tr>
</tbody>
</table>
INDEX

rowMultiply, 62
seq_k, 62
trimOnConfounders, 63
Zeller, 63