

# Package ‘parafac4microbiome’

May 9, 2026

**Title** Parallel Factor Analysis Modelling of Longitudinal Microbiome Data

**Version** 1.3.2

**Description** Creation and selection of PARAllel FACtor Analysis (PARAFAC) models of longitudinal microbiome data. You can import your own data with our import functions or use one of the example datasets to create your own PARAFAC models. Selection of the optimal number of components can be done using `assessModelQuality()` and `assessModelStability()`. The selected model can then be plotted using `plotPARAFACmodel()`. The Parallel Factor Analysis method was originally described by Carroll and Chang (1970) <[doi:10.1007/BF02310791](https://doi.org/10.1007/BF02310791)> and Harshman (1970) <<https://www.psychology.uwo.ca/faculty/harshman/wpppfac0.pdf>>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** compositions, cowplot, doParallel, dplyr, foreach, ggplot2, ggpubr, lifecycle, magrittr, methods, multiway, parallel, pracma, rlang, rTensor, stats, tidy

**Depends** R (>= 2.10)

**LazyData** true

**Suggests** knitr, phyloseq, rmarkdown, TreeSummarizedExperiment (>= 2.16.1), SummarizedExperiment (>= 1.38.1), testthat (>= 3.0.0), withr, NPLStoolbox

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**URL** <https://grvanderploeg.com/parafac4microbiome/>,  
<https://github.com/GRvanderPloeg/parafac4microbiome/>

**BugReports** <https://github.com/GRvanderPloeg/parafac4microbiome/issues>

**NeedsCompilation** no

**Author** Geert Roelof van der Ploeg [aut, cre] (ORCID: <https://orcid.org/0009-0007-5204-3386>),  
 Johan Westerhuis [ctb] (ORCID: <https://orcid.org/0000-0002-6747-9779>),  
 Anna Heintz-Buschart [ctb] (ORCID: <https://orcid.org/0000-0002-9780-1933>),  
 Age Smilde [ctb] (ORCID: <https://orcid.org/0000-0002-3052-4644>),  
 University of Amsterdam [cph, fnd]

**Maintainer** Geert Roelof van der Ploeg <g.r.ploeg@uva.nl>

**Repository** CRAN

**Date/Publication** 2025-07-31 10:00:08 UTC

## Contents

assessModelQuality . . . . .	3
assessModelStability . . . . .	4
calculateFMS . . . . .	5
calculateSparsity . . . . .	6
calculateVarExp . . . . .	7
calcVarExpPerComponent . . . . .	7
corcondia . . . . .	8
fac_to_vect . . . . .	9
flipLoadings . . . . .	9
Fujita2023 . . . . .	10
importPhyloseq . . . . .	11
importTreeSummarizedExperiment . . . . .	12
initializePARAFAC . . . . .	13
multiwayCenter . . . . .	14
multiwayCLR . . . . .	14
multiwayScale . . . . .	15
parafac . . . . .	15
parafac_core_als . . . . .	16
parafac_fun . . . . .	17
plotModelMetric . . . . .	18
plotModelStability . . . . .	18
plotModelITCCs . . . . .	20
plotPARAFACmodel . . . . .	20
processDataCube . . . . .	22
reinflateFac . . . . .	23
reinflateTensor . . . . .	24
reshapeData . . . . .	24
Shao2019 . . . . .	25
sortComponents . . . . .	26
transformPARAFACloadings . . . . .	27
vanderPloeg2024 . . . . .	27
vect_to_fac . . . . .	28

<b>Index</b>	<b>29</b>
--------------	-----------

---

assessModelQuality	<i>Create randomly initialized models to determine the correct number of components by assessing model quality metrics.</i>
--------------------	---

---

**Description**

Create randomly initialized models to determine the correct number of components by assessing model quality metrics.

**Usage**

```
assessModelQuality(
  X,
  minNumComponents = 1,
  maxNumComponents = 5,
  numRepetitions = 100,
  ctol = 1e-04,
  maxit = 500,
  numCores = 1
)
```

**Arguments**

<code>X</code>	Input data
<code>minNumComponents</code>	Minimum number of components (default 1).
<code>maxNumComponents</code>	Maximum number of components (default 5).
<code>numRepetitions</code>	Number of randomly initialized models to create (default 100).
<code>ctol</code>	Relative change in loss tolerated to call the algorithm converged in the ALS case (default 1e-4).
<code>maxit</code>	Maximum number of iterations allowed without convergence (default 500).
<code>numCores</code>	Number of cores to use. If set larger than 1, it will run the job in parallel (default 1)

**Value**

A list object of the following:

- `plots`: Plots of all assessed metrics and an overview plot showing a summary of all of them.
- `metrics`: metrics of every created model (number of iterations, sum of squared errors, CORCONDIA score and variance explained).
- `models`: all created models.

**Examples**

```
X = Fujita2023$data

# Run assessModelQuality with less strict convergence parameters as example
assessment = assessModelQuality(X,
                                minNumComponents=1,
                                maxNumComponents=3,
                                numRepetitions=5)

assessment$plots$overview
```

---

`assessModelStability` *Bootstrapping procedure to determine PARAFAC model stability for a given number of components.*

---

**Description**

Bootstrapping procedure to determine PARAFAC model stability for a given number of components.

**Usage**

```
assessModelStability(
  dataset,
  minNumComponents = 1,
  maxNumComponents = 5,
  numFolds = dim(dataset$data)[1],
  considerGroups = FALSE,
  groupVariable = "",
  colourCols = NULL,
  legendTitles = NULL,
  xLabels = NULL,
  legendColNums = NULL,
  arrangeModes = NULL,
  maxit = 500,
  ctol = 1e-04,
  numCores = 1
)
```

**Arguments**

<code>dataset</code>	See <a href="#">Fujita2023</a> , <a href="#">Shao2019</a> or <a href="#">vanderPloeg2024</a> .
<code>minNumComponents</code>	Minimum number of components (default 1).
<code>maxNumComponents</code>	Maximum number of components (default 5).
<code>numFolds</code>	Number of bootstrapped models to create.

considerGroups	Consider subject groups in calculating sparsity (default FALSE)
groupVariable	Column name in dataset\$mode1 that should be used to consider groups (default "")
colourCols	Vector of strings stating which column names should be factorized for colours per mode.
legendTitles	Vector of strings stating the legend title per mode.
xLabels	Vector of strings stating the x-axis labels per mode.
legendColNums	Vector of integers stating the desired number of columns for the legends per mode.
arrangeModes	Vector of boolean values per mode, stating if the loadings should be arranged according to colourCols (TRUE) or not (FALSE).
maxit	Maximum number of iterations allowed without convergence (default 500).
ctol	Relative change in loss tolerated to call the algorithm converged in the ALS case (default 1e-4).
numCores	Number of cores to use. If set larger than 1, it will run the job in parallel (default 1)

### Value

A list containing the following:

- models: All stabilized sign-flipped bootstrapped PARAFAC models.
- modelPlots: A list of plots of the median model with error bars for each number of components.
- FMSplot: A bar plot showing the Factor Match Scores per number of components (see Li et al., 2024).
- FMS: FMS values that the FMS plot is based on.

### Examples

```
processedFujita = processDataCube(Fujita2023, sparsityThreshold=0.99, centerMode=1, scaleMode=2)
modelStability = assessModelStability(processedFujita,
                                     minNumComponents=1,
                                     maxNumComponents=2)
```

---

calculateFMS

*Calculate Factor Match Score for all initialized models.*

---

### Description

Calculate Factor Match Score for all initialized models.

### Usage

```
calculateFMS(models)
```

**Arguments**

models                    Output of `parafac()` using `output="all"`.

**Value**

Vector containing FMS scores of all comparisons

**Examples**

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
X = reinflateTensor(A, B, C)
models = parafac(X, 2, initialization="random", nstart=10, maxit=2, output="all")
calculateFMS(models)
```

---

calculateSparsity            *Calculate sparsity across the feature mode of a multi-way array.*

---

**Description**

Calculate sparsity across the feature mode of a multi-way array.

**Usage**

```
calculateSparsity(dataset, considerGroups = FALSE, groupVariable = "")
```

**Arguments**

dataset                    See [Fujita2023](#), [Shao2019](#) or [vanderPloeg2024](#).

considerGroups          Consider subject groups in calculating sparsity (default FALSE)

groupVariable          Column name in `dataset$mode1` that should be used to consider groups (default "")

**Value**

Vector of sparsity fractions (N x J) where N is the number of groups and J is the number of features.

**Examples**

```
# No groups
sparsity = calculateSparsity(Fujita2023)
length(sparsity)
hist(sparsity)

# Consider groups
colnames(Shao2019$mode1)
sparsity = calculateSparsity(Shao2019, considerGroups=TRUE, groupVariable="Delivery_mode")
```

```
dim(sparsity)
hist(sparsity[,1])
hist(sparsity[,2])
```

---

calculateVarExp	<i>Calculate the variation explained by a PARAFAC model.</i>
-----------------	--

---

**Description**

Calculate the variation explained by a PARAFAC model.

**Usage**

```
calculateVarExp(Fac, X)
```

**Arguments**

Fac	Fac object output from the <code>parafac()</code> function.
X	Input data of the PARAFAC model.

**Value**

The variation explained by the model, expressed as a fraction (between 0-1).

**Examples**

```
X = Fujita2023$data
model = parafac(X, nfac=1, nstart=1)
calculateVarExp(model$Fac, X)
```

---

calcVarExpPerComponent	<i>Calculate the variance explained of a PARAFAC model, per component</i>
------------------------	---

---

**Description**

Calculate the variance explained of a PARAFAC model, per component

**Usage**

```
calcVarExpPerComponent(Fac, X)
```

**Arguments**

Fac	Fac object output of a model
X	Input dataset

**Value**

Vector of scalars of the percentage of variation explained per component

**Examples**

```
X = array(rnorm(108*100*10), c(108,100,10))
model = parafac(X, 2)
calcVarExpPerComponent(model$Fac, X)
```

---

corcondia

*Core Consistency Diagnostic (CORCONDIA) calculation*

---

**Description**

Core Consistency Diagnostic (CORCONDIA) calculation

**Usage**

```
corcondia(X, Fac)
```

**Arguments**

X	Input data matrix
Fac	PARAFAC model Fac object

**Value**

Scalar of the CORCONDIA value

**Examples**

```
X = Fujita2023$data
model = parafac(X, 2)
corcondia(X, model$Fac)
```

---

fac_to_vect	<i>Vectorize Fac object</i>
-------------	-----------------------------

---

**Description**

Vectorize Fac object

**Usage**

```
fac_to_vect(Fac)
```

**Arguments**

Fac                      Fac object output of [parafac](#).

**Value**

Vectorized Fac object

**Examples**

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
Fac = list(A, B, C)
v = fac_to_vect(Fac)
```

---

flipLoadings	<i>Sign flip the loadings of many randomly initialized models to make consistent overview plots.</i>
--------------	--

---

**Description**

Sign flip the loadings of many randomly initialized models to make consistent overview plots.

**Usage**

```
flipLoadings(models, X)
```

**Arguments**

models                      Output of [parafac](#).  
X                              Input dataset of parafac modelling procedure.

**Value**

models with sign flipped components where applicable.

**Examples**

```
A = array(rnorm(108*2), c(108,2))
B = array(rnorm(100*2), c(100,2))
C = array(rnorm(10*2), c(10,2))
X = reinflateTensor(A, B, C)
models = parafac(X, 2, nstart=10, output="all", sortComponents=TRUE)
flippedModels = flipLoadings(models, X)
```

---

Fujita2023

*Fujita2023 longitudinal microbiome data*

---

**Description**

The Fujita2023 longitudinal microbiome dataset as a three-dimensional array, with replicates in mode 1, microbial abundances in mode 2 and time in mode 3.

**Usage**

```
Fujita2023
```

**Format**

**Fujita2023:**

A list object with three elements:

**data** Array object of the data cube

**mode1** Dataframe with all the subject metadata, ordered the same as the rows in the data cube.

**mode2** Taxonomic classification of the microbiota, ordered the same as the columns in the data cube.

**mode3** Dataframe with the time metadata, ordered the same as the third dimension in the array.

...

**Source**

[doi:10.1186/s40168023014745](https://doi.org/10.1186/s40168023014745)

---

importPhyloseq      *Import Phyloseq object for PARAFAC modelling*

---

## Description

Import Phyloseq object for PARAFAC modelling

## Usage

```
importPhyloseq(phyloseqObject, subjectIDs, thirdMode)
```

## Arguments

phyloseqObject	Phyloseq object containing at least an otu table and sample data, preferably also taxonomic information.
subjectIDs	Column name in sam_data corresponding to the subject IDs.
thirdMode	Column name in sam_data corresponding to the study design aspect to put in the third mode of the data cube.

## Value

List object containing:

- 'data': data cube
- 'mode1': metadata of the subject mode
- 'mode2': taxonomy information
- 'mode3': metadata of the third mode

## Examples

```
library(phyloseq)
data(GlobalPatterns)
GP = GlobalPatterns

# Add custom subject IDs to the sample data to make this example work
alteredSampleData = sample_data(GP)
alteredSampleData$subjectID = c(1,2,3,1,2,1,2,3,1,2,1,2,1,2,3,1,2,3,1,2,3,4,5,1,2,3)
df = phyloseq(otu_table(GP), tax_table(GP), alteredSampleData)

# Make a data cube with SampleType (soil, feces, etc.) as the third mode.
result = importPhyloseq(df, subjectIDs = "subjectID", thirdMode="SampleType")
```

---

```
importTreeSummarizedExperiment
  Import TreeSummarizedExperiment object for PARAFAC modelling
```

---

## Description

Import TreeSummarizedExperiment object for PARAFAC modelling

## Usage

```
importTreeSummarizedExperiment(
  treeObject,
  subjectIDs,
  thirdMode,
  taxa_are_rows
)
```

## Arguments

treeObject	TreeSummarizedExperiment object containing at least an OTU table and sample information, preferably also taxonomic information.
subjectIDs	Column name in the sample information corresponding to the subject IDs.
thirdMode	Column name in the sample information corresponding to the study design aspect to put in the third mode of the data cube.
taxa_are_rows	Boolean specifying if the taxa are in the rows of the OTU table (TRUE) or not (FALSE).

## Value

List object containing:

- 'data': data cube
- 'mode1': metadata of the subject mode
- 'mode2': taxonomy information
- 'mode3': metadata of the third mode

## Examples

```
library(TreeSummarizedExperiment)

fakeOTU = t(rTensor::k_unfold(rTensor::as.tensor(Fujita2023$data), 2)@data)
fakeTaxa = as.matrix(Fujita2023$mode2)
fakeSam = as.data.frame(cbind(rep(1:8, 110), rep(1:110, each=8)))
colnames(fakeSam) = c("replicate.id", "timepoint")

fakeTreeObj = TreeSummarizedExperiment(assays = list(Count = fakeOTU),
```

```

                                rowData = fakeSam,
                                colData = fakeTaxa)
dataset = importTreeSummarizedExperiment(fakeTreeObj,
                                         subjectIDs="replicate.id",
                                         thirdMode="timepoint",
                                         taxa_are_rows=FALSE)

```

---

initializePARAFAC      *Initialize PARAFAC algorithm input vectors*

---

### Description

Initialize PARAFAC algorithm input vectors

### Usage

```
initializePARAFAC(Tensor, nfac, initialization = "random", output = "Fac")
```

### Arguments

Tensor	Input dataset matrix or tensor
nfac	Number of components to initialize.
initialization	Either "random" for random initialization or "svd" for svd based.
output	Output the initialized components as a Fac object ("Fac", default) or as a vector ("vect").

### Value

Fac or vector with initialized components.

### Examples

```

A = array(rnorm(108,2), c(108,2))
B = array(rnorm(100,2), c(100,2))
C = array(rnorm(10,2), c(10,2))
Tensor = reinflateTensor(A, B, C, returnAsTensor=TRUE)
init = initializePARAFAC(Tensor, 2)

```

---

multiwayCenter      *Center a multi-way array*

---

**Description**

Center a multi-way array

**Usage**

```
multiwayCenter(X, mode = 1)
```

**Arguments**

X	Multi-way array
mode	Mode to center across (default 1).

**Value**

Centered multi-way array

**Examples**

```
cube_cnt = multiwayCenter(Fujita2023$data)
```

---

multiwayCLR      *Perform a centered log-ratio transform over a multi-way array*

---

**Description**

Note: Propagates NAs corresponding to missing samples.

**Usage**

```
multiwayCLR(X, pseudocount = 1)
```

**Arguments**

X	Multi-way array of counts
pseudocount	Pseudocount value to use (default 1).

**Value**

CLRed cube

**Examples**

```
cubeCLR = multiwayCLR(Fujita2023$data)
```

---

multiwayScale	<i>Scale a multi-way array</i>
---------------	--------------------------------

---

**Description**

Scale a multi-way array

**Usage**

```
multiwayScale(X, mode = 2)
```

**Arguments**

X	Multi-way array
mode	Mode to scale within: 1=subjects,2=features,3=time (default 2).

**Value**

Scaled multi-way array

**Examples**

```
cube_scl = multiwayCenter(Fujita2023$data)
```

---

parafac	<i>Parallel Factor Analysis</i>
---------	---------------------------------

---

**Description**

Parallel Factor Analysis

**Usage**

```
parafac(  
  Tensor,  
  nfac,  
  nstart = 1,  
  maxit = 500,  
  ctol = 1e-04,  
  initialization = "random",  
  output = "best",  
  sortComponents = FALSE  
)
```

**Arguments**

Tensor	3-way matrix of numeric data
nfac	Number of factors (components) to fit.
nstart	Number of models to randomly initialize (default 1).
maxit	Maximum number of iterations allowed without convergence (default 500).
ctol	Relative change in loss tolerated to call the algorithm converged in the ALS case (default 1e-4).
initialization	"Random" for randomly initialized input vectors or "nvec" for svd-based best guess.
output	String ("best"/"all") Return only the best model of the nstart models ("best") or return all of them in a list object ("all").
sortComponents	Boolean to sort the components based on their variance explained (default FALSE)

**Value**

List object of the PARAFAC model or models.

**Examples**

```
X = array(rnorm(108*100*10), c(108,100,10))
model = parafac(X, 2)
```

---

parafac\_core\_als      *Internal PARAFAC alternating least-squares (ALS) core algorithm*

---

**Description**

Internal PARAFAC alternating least-squares (ALS) core algorithm

**Usage**

```
parafac_core_als(Tensor, nfac, init, maxit = 500, ctol = 1e-04)
```

**Arguments**

Tensor	Tensor data object
nfac	Number of components to compute
init	Initialization from <a href="#">initializePARAFAC</a> .
maxit	Maximum number of iterations to run (default 500).
ctol	Loss function tolerance for convergence (default 1e-4)

**Value**

List containing a Fac object and the loss per iteration

**Examples**

```

A = array(rnorm(108*2), c(108,2))
B = array(rnorm(100*2), c(100,2))
C = array(rnorm(10*2), c(10,2))
Tensor = reinflateTensor(A, B, C)
init = initializePARAFAC(Tensor, 2)
model = parafac_core_als(Tensor, 2, init)

```

---

parafac_fun	<i>PARAFAC loss function calculation</i>
-------------	--

---

**Description**

PARAFAC loss function calculation

**Usage**

```
parafac_fun(x, Tensor, lambdas = NULL)
```

**Arguments**

x	Vector of fitted loadings generated by the PARAFAC algorithm, can also be a Fac object
Tensor	input data
lambdas	If lambdas (from the kruskal tensor case) are generated to make the Fac norm 1, they can be supplied.

**Value**

Scalar value of the loss function

**Examples**

```

A = array(rnorm(108*2), c(108,2))
B = array(rnorm(100*2), c(100,2))
C = array(rnorm(10*2), c(10,2))
X = reinflateTensor(A, B, C)
model = parafac(X, 2)
f = parafac_fun(model$Fac, X)

```

---

plotModelMetric      *Plot diagnostics of many initialized PARAFAC models.*

---

### Description

Plot diagnostics of many initialized PARAFAC models.

### Usage

```
plotModelMetric(
  metric,
  plottingMode = "box",
  ylabel = "metric",
  titleString = ""
)
```

### Arguments

metric	Matrix of metrics per initialized model (number of models x number of components).
plottingMode	Plot the metrics as a box plot ("box", default) or as a bar plot ("bar").
ylabel	String of the y axis label (default "metric").
titleString	String of the plot title (default "").

### Value

A plot of the metrics

### Examples

```
varExp = array(runif(100*2, min=50, max=100), c(100,2))
plotModelMetric(varExp, plottingMode="box", ylabel="Variation explained (%)")
```

---

plotModelStability      *Plot a summary of the loadings of many initialized parafac models.*

---

### Description

Plot a summary of the loadings of many initialized parafac models.

**Usage**

```
plotModelStability(
  models,
  dataset,
  colourCols = NULL,
  legendTitles = NULL,
  xLabels = NULL,
  legendColNums = NULL,
  arrangeModes = NULL,
  continuousModes = NULL,
  overallTitle = ""
)
```

**Arguments**

models	Models list output from <code>parafac()</code> using <code>output="all"</code> .
dataset	A longitudinal microbiome dataset, ideally processed with <code>processDataCube()</code> , formatted as follows: <b>data</b> Array object of the data cube <b>mode1</b> Dataframe with all the subject metadata, ordered the same as the rows in the data cube. <b>mode2</b> Taxonomic classification of the microbiota, ordered the same as the columns in the data cube. <b>mode3</b> Dataframe with the time metadata, ordered the same as the third dimension in the array.
colourCols	Vector of strings stating which column names should be factorized for colours per mode.
legendTitles	Vector of strings stating the legend title per mode.
xLabels	Vector of strings stating the x-axis labels per mode.
legendColNums	Vector of integers stating the desired number of columns for the legends per mode.
arrangeModes	Vector of boolean values per mode, stating if the loadings should be arranged according to <code>colourCols</code> (TRUE) or not (FALSE).
continuousModes	Vector of boolean values per mode, stating if the loadings should be plotted as a line plot (TRUE) or a bar plot (FALSE).
overallTitle	Overall title of the plot.

**Value**

Plot of loadings with error bars

**Examples**

```
processedFujita = processDataCube(Fujita2023, sparsityThreshold=0.99, centerMode=1, scaleMode=2)
models = parafac(processedFujita$data, 2, nstart=10, output="all")
plotModelStability(models, processedFujita)
```

plotModelTCCs                    *Plots Tucker Congruence Coefficients of randomly initialized models.*

---

**Description**

Plots Tucker Congruence Coefficients of randomly initialized models.

**Usage**

```
plotModelTCCs(models)
```

**Arguments**

models                    Models list output of `parafac()` using `output="all"`.

**Value**

Plot of TCCs

**Examples**

```
processedFujita = processDataCube(Fujita2023, sparsityThreshold=0.99, centerMode=1, scaleMode=2)
models = parafac(processedFujita$data, 3, nstart=10, output="all")
plotModelTCCs(models)
```

---

plotPARAFACmodel                    *Plot a PARAFAC model*

---

**Description**

Plot a PARAFAC model

**Usage**

```
plotPARAFACmodel(
  model,
  dataset,
  numComponents,
  colourCols = NULL,
  legendTitles = NULL,
  xLabels = NULL,
  legendColNums = NULL,
  arrangeModes = NULL,
  continuousModes = NULL,
  overallTitle = ""
)
```

**Arguments**

model	Model output from <code>parafac()</code> .
dataset	A longitudinal microbiome dataset, ideally processed with <code>processDataCube()</code> , formatted as follows: <b>data</b> Array object of the data cube <b>mode1</b> Dataframe with all the subject metadata, ordered the same as the rows in the data cube. <b>mode2</b> Taxonomic classification of the microbiota, ordered the same as the columns in the data cube. <b>mode3</b> Dataframe with the time metadata, ordered the same as the third dimension in the array.
numComponents	Number of PARAFAC components in the model.
colourCols	Vector of strings stating which column names should be factorized for colours per mode.
legendTitles	Vector of strings stating the legend title per mode.
xLabels	Vector of strings stating the x-axis labels per mode.
legendColNums	Vector of integers stating the desired number of columns for the legends per mode.
arrangeModes	Vector of boolean values per mode, stating if the loadings should be arranged according to colourCols (TRUE) or not (FALSE).
continuousModes	Vector of boolean values per mode, stating if the loadings should be plotted as a line plot (TRUE) or a bar plot (FALSE).
overallTitle	Overall title of the plot.

**Value**

Plot object

**Examples**

```
library(multiway)
library(dplyr)
library(ggplot2)
set.seed(0)

# Process the data
processedFujita = processDataCube(Fujita2023, sparsityThreshold=0.9, centerMode=1, scaleMode=2)

# Make PARAFAC model
model = parafac(processedFujita$data, nfac=2, nstart=10, verbose=FALSE)

# Make plot
plotPARAFACmodel(model, processedFujita,
  numComponents = 2,
  colourCols = c("", "Genus", ""))
```

```

legendTitles = c("", "Genus", ""),
xLabels = c("Replicate", "Feature index", "Time point"),
legendColNums = c(0,5,0),
arrangeModes = c(FALSE, TRUE, FALSE),
continuousModes = c(FALSE,FALSE,TRUE),
overallTitle = "Fujita PARAFAC model")

```

---

processDataCube	<i>Process a multi-way array of count data.</i>
-----------------	---

---

### Description

Process a multi-way array of count data.

### Usage

```

processDataCube(
  dataset,
  sparsityThreshold = 1,
  considerGroups = FALSE,
  groupVariable = "",
  CLR = TRUE,
  centerMode = 0,
  scaleMode = 0
)

```

### Arguments

dataset	A longitudinal microbiome dataset, formatted as follows: <b>data</b> Array object of the data cube filled with counts <b>mode1</b> Dataframe with all the subject metadata, ordered the same as the rows in the data cube. <b>mode2</b> Taxonomic classification of the microbiota, ordered the same as the columns in the data cube. <b>mode3</b> Dataframe with the time metadata, ordered the same as the third dimension in the array. See <a href="#">Fujita2023</a> , <a href="#">Shao2019</a> or <a href="#">vanderPloeg2024</a> for more information.
sparsityThreshold	Maximum sparsity for a feature to be selected (default=1, i.e. do not select features).
considerGroups	Consider groups when calculating sparsity (default=FALSE).
groupVariable	Column name in dataset\$mode1 that should be used to consider groups (default="").
CLR	Perform a centered log-ratio transformation of the count data (default=TRUE).

centerMode	Mode to center across: 1=subjects,2=features,3=time (default 0, i.e. do not center). See <a href="#">multiwayCenter()</a> for more information.
scaleMode	Mode to scale within: 1=subjects,2=features,3=time (default 0, i.e. do not scale). See <a href="#">multiwayScale()</a> for more information.

**Value**

CLRed, centered and scaled cube

**Examples**

```
processedCube = processDataCube(Fujita2023)
```

---

reinflateFac	<i>Calculate Xhat from a model Fac object</i>
--------------	---

---

**Description**

Calculate Xhat from a model Fac object

**Usage**

```
reinflateFac(Fac, X, returnAsTensor = FALSE)
```

**Arguments**

Fac	Fac object from parafac
X	Input data X
returnAsTensor	Boolean to return Xhat as rTensor tensor (TRUE) or matrix (default, FALSE).

**Value**

Xhat

**Examples**

```
processedFujita = processDataCube(Fujita2023, sparsityThreshold=0.99, centerMode=1, scaleMode=2)
model = parafac(processedFujita$data, nfac=1, nstart=1)
Xhat = reinflateFac(model$Fac, processedFujita$data)
```

---

reinflateTensor	<i>Create a tensor out of a set of matrices similar to a component model.</i>
-----------------	---

---

**Description**

Create a tensor out of a set of matrices similar to a component model.

**Usage**

```
reinflateTensor(A, B, C, returnAsTensor = FALSE)
```

**Arguments**

A	I x N matrix corresponding to loadings in the first mode for N components.
B	J x N matrix corresponding to loadings in the second mode for N components.
C	K x N matrix corresponding to loadings in the third mode for N components.
returnAsTensor	Boolean return as rTensor S4 tensor object (default FALSE).

**Value**

M, an I x J x K tensor.

**Examples**

```
A = rnorm(108)
B = rnorm(100)
C = rnorm(10)
M = reinflateTensor(A,B,C)
```

---

reshapeData	<i>Reorganize longitudinal microbiome into a data cube ready for PARAFAC modelling.</i>
-------------	---

---

**Description**

Reorganize longitudinal microbiome into a data cube ready for PARAFAC modelling.

**Usage**

```
reshapeData(
  Xlong,
  subjectMetadata,
  featureMetadata,
  timepointMetadata,
  timepointOrder = sort(unique(timepointMetadata))
)
```

**Arguments**

**Xlong** Longitudinal microbiome count data in matrix (long) format.  
**subjectMetadata** Vector containing the subjects corresponding to the measurements in Xlong.  
**featureMetadata** Taxonomic classification of the microbiota, ordered the same as the columns in Xlong.  
**timepointMetadata** Vector containing the time points corresponding to the measurements in Xlong.  
**timepointOrder** Vector containing the required order of the timepoints in timepointMetadata (default: `sort(unique(timepointMetadata))`).

**Value**

#' A list object containing

**data** Array object of the data cube

**mode1** Dataframe with the subjects, ordered the same as the rows in the data cube.

**mode2** Taxonomic classification of the microbiota, ordered the same as the columns in the data cube.

**mode3** Dataframe with the time metadata, ordered the same as the third dimension in the data cube.

...

**Examples**

```
library(dplyr)

Xlong = array(rnorm(108*5*10), c(108*5, 10))
subjects = rep(1:108, 5)
features = rep(1:10)
timepoints = rep(1:5, each=108)

dataset = reshapeData(Xlong, subjects, features, timepoints)
```

---

Shao2019

*Shao2019 longitudinal microbiome data*

---

**Description**

The Shao2019 longitudinal microbiome dataset as a three-dimensional array, with subjects in mode 1, microbial abundances in mode 2 and time in mode 3. Note: only time points 4, 7, 21 and Infancy are used. Note: all-zero microbial abundances have been removed to save disk space.

**Usage**

Shao2019

**Format**

Shao2019:

A list object with three elements:

**data** Array object of the data cube

**mode1** Dataframe with all the subject metadata, ordered the same as the rows in the data cube.

**mode2** Taxonomic classification of the microbiota, ordered the same as the columns in the data cube.

**mode3** Dataframe with the time metadata, ordered the same as the third dimension in the array.

...

**Source**

[doi:10.1038/s4158601915601](https://doi.org/10.1038/s4158601915601)

---

sortComponents	<i>Sort PARAFAC components based on variance explained per component.</i>
----------------	---

---

**Description**

Sort PARAFAC components based on variance explained per component.

**Usage**

```
sortComponents(Fac, X)
```

**Arguments**

Fac	Fac object output of a <a href="#">parafac</a> model
X	Input data

**Value**

Fac object of sorted components

**Examples**

```
X = array(rnorm(108*100*10), c(108,100,10))
model = parafac(X, 2)
sortedFac = sortComponents(model$Fac, X)
```

---

`transformPARAFACloadings`

*Transform PARAFAC loadings to an orthonormal basis. Note: this function only works for 3-way PARAFAC models.*

---

### Description

Transform PARAFAC loadings to an orthonormal basis. Note: this function only works for 3-way PARAFAC models.

### Usage

```
transformPARAFACloadings(Fac, modeToCorrect, moreOutput = FALSE)
```

### Arguments

<code>Fac</code>	Fac object from a PARAFAC object, see <a href="#">parafac()</a> .
<code>modeToCorrect</code>	Correct the subject (1), feature (2) or time mode (3).
<code>moreOutput</code>	Give orthonormal basis and transformation matrices as part of output (default FALSE).

### Value

Corrected loadings of the specified mode.

### Examples

```
processedFujita = processDataCube(Fujita2023, sparsityThreshold=0.99, centerMode=1, scaleMode=2)
model = parafac(processedFujita$data, nfac=2, nstart=1)
transformedA = transformPARAFACloadings(model$Fac, 1)
```

---

`vanderPloeg2024`

*vanderPloeg2024 longitudinal multi-omics dataset*

---

### Description

The vanderPloeg2024 longitudinal multi-omics dataset containing six oral microbiome niches, as well as salivary metabolomics and oral health measurements.

### Usage

```
vanderPloeg2024
```

**Format**

vanderPloeg2024:

Each measured dataset contains three elements:

**data** Array object of the data cube

**mode1** Dataframe with all the subject metadata, ordered the same as the rows in the data cube.

**mode2** Taxonomic classification of the microbiota, ordered the same as the columns in the data cube.

**mode3** Dataframe with the time metadata, ordered the same as the third dimension in the array.

...

**Source**

[doi:10.1101/2024.03.18.585469](https://doi.org/10.1101/2024.03.18.585469)

---

vect_to_fac	<i>Convert vectorized output of PARAFAC to a Fac list object with all loadings per mode.</i>
-------------	--

---

**Description**

Convert vectorized output of PARAFAC to a Fac list object with all loadings per mode.

**Usage**

```
vect_to_fac(vect, X, sortComponents = FALSE)
```

**Arguments**

**vect** Vectorized output of PARAFAC modelling

**X** Input data

**sortComponents** Sort the order of the components by variation explained (default FALSE).

**Value**

Fac: list object with all loadings in all components per mode, ordered the same way as Z\$modes.

**Examples**

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))

X = reinflateTensor(A, B, C)
result = initializePARAFAC(X, 2, initialization="random", output="vect")
Fac = vect_to_fac(result, X)
```

# Index

## \* datasets

Fujita2023, [10](#)  
Shao2019, [25](#)  
vanderPloeg2024, [27](#)

assessModelQuality, [3](#)  
assessModelStability, [4](#)

calculateFMS, [5](#)  
calculateSparsity, [6](#)  
calculateVarExp, [7](#)  
calcVarExpPerComponent, [7](#)  
corcondia, [8](#)

fac\_to\_vect, [9](#)  
flipLoadings, [9](#)  
Fujita2023, [4](#), [6](#), [10](#), [22](#)

importPhyloseq, [11](#)  
importTreeSummarizedExperiment, [12](#)  
initializePARAFAC, [13](#), [16](#)

multiwayCenter, [14](#)  
multiwayCenter(), [23](#)  
multiwayCLR, [14](#)  
multiwayScale, [15](#)  
multiwayScale(), [23](#)

parafac, [9](#), [15](#), [26](#)  
parafac(), [6](#), [7](#), [19–21](#), [27](#)  
parafac\_core\_als, [16](#)  
parafac\_fun, [17](#)  
plotModelMetric, [18](#)  
plotModelStability, [18](#)  
plotModelTCCs, [20](#)  
plotPARAFACmodel, [20](#)  
processDataCube, [22](#)  
processDataCube(), [19](#), [21](#)

reinflateFac, [23](#)  
reinflateTensor, [24](#)

reshapeData, [24](#)

Shao2019, [4](#), [6](#), [22](#), [25](#)  
sortComponents, [26](#)

transformPARAFACloadings, [27](#)

vanderPloeg2024, [4](#), [6](#), [22](#), [27](#)  
vect\_to\_fac, [28](#)