

# Package ‘mfa’

May 16, 2024

**Title** Bayesian hierarchical mixture of factor analyzers for modelling genomic bifurcations

**Version** 1.27.0

**Description** MFA models genomic bifurcations using a Bayesian hierarchical mixture of factor analysers.

**Depends** R (>= 3.4.0)

**Imports** methods, stats, ggplot2, Rcpp, dplyr, ggmcmc, MCMCpack, MCMCglmm, coda, magrittr, tibble, Biobase

**LinkingTo** Rcpp

**License** GPL (>= 2)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**biocViews** ImmunoOncology, RNASeq, GeneExpression, Bayesian, SingleCell

**Suggests** knitr, rmarkdown, BiocStyle, testthat

**VignetteBuilder** knitr

**NeedsCompilation** yes

**git\_url** <https://git.bioconductor.org/packages/mfa>

**git\_branch** devel

**git\_last\_commit** 9505ca7

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-05-15

**Author** Kieran Campbell [aut, cre]

**Maintainer** Kieran Campbell <kieranrcampbell@gmail.com>

## Contents

calculate_chi . . . . .	2
create_synthetic . . . . .	3
cs_sigmoid . . . . .	4
empirical_lambda . . . . .	4
log_sum_exp . . . . .	5
map_branch . . . . .	5
mcmcify . . . . .	6
mfa . . . . .	6
plot_chi . . . . .	8
plot_dropout_relationship . . . . .	9
plot_mfa_autocorr . . . . .	9
plot_mfa_trace . . . . .	10
posterior . . . . .	10
print.mfa . . . . .	11
summary.mfa . . . . .	12
to_ggmcmc . . . . .	13
transient . . . . .	13

<b>Index</b>	<b>14</b>
--------------	-----------

---

calculate_chi	<i>Calculate posterior chi precision parameters</i>
---------------	-----------------------------------------------------

---

### Description

Calculates a data frame of the MAP estimates of  $\chi$ .

### Usage

```
calculate_chi(m)
```

### Arguments

`m` A fit returned from `mfa`

### Value

A `data_frame` with one entry for the feature names and one for the MAP estimates of `chi` (using the `posterior.mode` function from `MCMCglmm`).

### Examples

```
synth <- create_synthetic(C = 20, G = 5)
m <- mfa(synth$X)
chi_map <- calculate_chi(m)
```

---

create\_synthetic      *Create synthetic data*

---

### Description

Create synthetic bifurcating data for two branches. Optionally incorporate zero inflation and transient gene expression.

### Usage

```
create_synthetic(C = 100, G = 40, p_transient = 0, zero_negative = TRUE,
  model_dropout = FALSE, lambda = 1)
```

### Arguments

C	Number of cells to simulate
G	Number of genes to simulate
p_transient	Proportion of genes that exhibit transient expression
zero_negative	Logical: should expression generated less than zero be set to zero? This will zero-inflate the data
model_dropout	Logical: if true, expression will be set to zero with the exponential dropout formula dependent on the latent expression using dropout parameter lambda
lambda	The dropout parameter

### Value

A list with the following entries:

- X A cell-by-feature expression matrix
- branch A vector of length C assigning cells to branches
- pst A vector of pseudotimes for each cell
- k The  $k$  parameters
- phi The  $\phi$  parameters
- delta The  $\delta$  parameters
- p\_transient The proportion of genes simulated as transient according to the original function call

### Examples

```
synth <- create_synthetic()
```

---

cs_sigmoid	<i>Sigmoid function for activations - renamed to avoid naming conflict</i>
------------	----------------------------------------------------------------------------

---

**Description**

Sigmoid function for activations - renamed to avoid naming conflict

**Usage**

```
cs_sigmoid(t, phi, k, delta)
```

**Value**

Sigmoid function given the parameters

---

empirical_lambda	<i>Estimate the dropout parameter</i>
------------------	---------------------------------------

---

**Description**

Estimate the dropout parameter

**Usage**

```
empirical_lambda(y, lower_limit = 0)
```

**Arguments**

y	A cell-by-gene expression matrix
lower_limit	The limit below which expression counts as 'dropout'

**Value**

The estimated lambda

**Examples**

```
synth <- create_synthetic(C = 20, G = 5, zero_negative = TRUE, model_dropout = TRUE)
lambda <- empirical_lambda(synth$X)
```

---

log_sum_exp	<i>Log sum of exponentials</i>
-------------	--------------------------------

---

**Description**

Log sum of exponentials

**Usage**

```
log_sum_exp(x)
```

**Arguments**

x                      Vector of quantities

**Value**

Log sum of exponentials in a numerically stable manner.

---

map_branch	<i>Find the MAP branch and uncertainty</i>
------------	--------------------------------------------

---

**Description**

Find the MAP branch and uncertainty

**Usage**

```
map_branch(g)
```

**Arguments**

g                      The trace slot from an mfa fit

**Value**

A data frame with a column corresponding to the MAP branch and a further column corresponding to the proportion of samples assigned to that branch (a measure of uncertainty).

---

mcmcify	<i>Turn a matrix's columns into informative names</i>
---------	-------------------------------------------------------

---

**Description**

Turn a matrix's columns into informative names

**Usage**

```
mcmcify(m, name)
```

**Arguments**

m	A fit returned from mfa
name	The name of the parameter

**Value**

The input with consistent naming.

---

mfa	<i>Fit a MFA object</i>
-----	-------------------------

---

**Description**

Perform Gibbs sampling inference for a hierarchical Bayesian mixture of factor analysers to identify bifurcations in single-cell expression data.

**Usage**

```
mfa(y, iter = 2000, thin = 1, burn = iter/2, b = 2,
    zero_inflation = FALSE, pc_initialise = 1, prop_collapse = 0,
    scale_input = !zero_inflation, lambda = NULL, eta_tilde = NULL,
    alpha = 0.1, beta = 0.1, theta_tilde = 0, tau_eta = 1,
    tau_theta = 1, tau_c = 1, alpha_chi = 0.01, beta_chi = 0.01,
    w_alpha = 1/b, clamp_pseudotimes = FALSE)
```

**Arguments**

y	A cell-by-gene single-cell expression matrix or an ExpressionSet object
iter	Number of MCMC iterations
thin	MCMC samples to thin
burn	Number of MCMC samples to throw away
b	Number of branches to model

zero_inflation	Logical, should zero inflation be enabled?
pc_initialise	Which principal component to initialise pseudotimes to
prop_collapse	Proportion of Gibbs samples which should marginalise over c
scale_input	Logical. If true, input is scaled to have mean 0 variance 1
lambda	The dropout parameter - by default estimated using the function <code>empirical_lambda</code>
eta_tilde	Hyperparameter
alpha	Hyperparameter
beta	Hyperparameter
theta_tilde	Hyperparameter
tau_eta	Hyperparameter
tau_theta	Hyperparameter
tau_c	Hyperparameter
alpha_chi	Hyperparameter
beta_chi	Hyperparameter
w_alpha	Hyperparameter
clamp_pseudotimes	This clamps the pseudotimes to their initial values and doesn't perform sampling. Should be FALSE except for diagnostics.

## Details

The column names of Y are used as feature (gene/transcript) names while the row names are used as cell names. If either of these is undefined then the corresponding names are set to `cell_x` or `feature_y`.

It is recommended the form of Y is analogous to log-expression to mitigate the impact of outliers.

In the absence of prior information, three valid local maxima in the posterior likelihood exist (see manuscript). Setting the initial values to a principal component typically fixes sampling to one of them, analogous to specifying a root cell in similar methods.

The hyper-parameter `eta_tilde` represents the expected expression in the absence of any actual expression measurements. While a Bayesian purist might reason this based on knowledge of the measurement technology, simply taking the mean of the input matrix in an Empirical Bayes style seems reasonable.

The degree of shrinkage of the factor loading matrices to a common value is given by the gamma prior on  $\chi$ . The mean of this is  $\alpha_{\chi} / \beta_{\chi}$  while the variance  $\alpha_{\chi} / \beta_{\chi}^2$ . Therefore, to obtain higher levels of shrinkage increase  $\alpha_{\chi}$  with respect to  $\beta_{\chi}$ .

The collapsed Gibbs sampling option given by `collapse` involves marginalising out c (the factor loading intercepts) when updating the branch assignment parameters  $\gamma$  which tends to soften the branch assignments.

If zero inflation is enabled using the `zero_inflation` parameter then scaling should *not* be enabled.

**Value**

An S3 structure with the following entries:

- `traces` A list of iteration-by-dim trace matrices for several important variables
- `iter` Number of iterations
- `thin` Thinning applied
- `burn` Burn period at the start of MCMC
- `b` Number of branches modelled
- `prop_collapse` Proportion of updates for gamma that are collapsed
- `N` Number of cells
- `G` Number of features (genes/transcripts)
- `feature_names` Names of features
- `cell_names` Names of cells

**Examples**

```
synth <- create_synthetic(C = 20, G = 5)
m <- mfa(synth$X)
```

---

plot\_chi

*Plot posterior precision parameters*

---

**Description**

Plot posterior precision parameters

**Usage**

```
plot_chi(m, nfeatures = m$G)
```

**Arguments**

<code>m</code>	A fit returned from <code>mfa</code>
<code>nfeatures</code>	Top number of

**Value**

A `ggplot2` bar-plot showing the map estimates of  $\chi^{-1}$

**Examples**

```
synth <- create_synthetic(C = 20, G = 5)
m <- mfa(synth$X)
plot_chi(m)
```

---

plot\_dropout\_relationship  
*Plot the dropout relationship*

---

**Description**

Plot the dropout relationship

**Usage**

```
plot_dropout_relationship(y, lambda = empirical_lambda(y))
```

**Arguments**

y	The input data matrix
lambda	The estimated value of lambda

**Value**

A ggplot2 plot showing the estimated dropout relationship

**Examples**

```
synth <- create_synthetic(C = 20, G = 5, zero_negative = TRUE, model_dropout = TRUE)
lambda <- empirical_lambda(synth$X)
plot_dropout_relationship(synth$X, lambda)
```

---

plot\_mfa\_autocorr      *Plot MFA autocorrelation*

---

**Description**

Plots the autocorrelation of the posterior log-likelihood.

**Usage**

```
plot_mfa_autocorr(m)
```

**Arguments**

m	A fit returned from mfa
---	-------------------------

**Value**

A ggplot2 plot returned by the ggcmc package plotting the autocorrelation of the posterior log-likelihood.

**Examples**

```
synth <- create_synthetic(C = 20, G = 5)
m <- mfa(synth$X)
plot_mfa_autocorr(m)
```

---

plot_mfa_trace	<i>Plot MFA trace</i>
----------------	-----------------------

---

**Description**

Plots the trace of the posterior log-likelihood.

**Usage**

```
plot_mfa_trace(m)
```

**Arguments**

m                    A fit returned from mfa

**Value**

A ggplot2 plot plotting the trace of the posterior log-likelihood.

**Examples**

```
synth <- create_synthetic(C = 20, G = 5)
m <- mfa(synth$X)
plot_mfa_trace(m)
```

---

posterior	<i>Calculate the log-posterior during inference</i>
-----------	-----------------------------------------------------

---

**Description**

Calculate the log-posterior during inference

**Usage**

```
posterior(y, c, k, pst, tau, gamma, theta, eta, chi, w, tau_c, r, alpha, beta,
  theta_tilde, eta_tilde, tau_theta, tau_eta, alpha_chi, beta_chi,
  zero_inflation = FALSE, lambda = NULL)
```

**Arguments**

y	Cell-by-gene gene expression matrix
c	Factor loading parameter
k	Factor loading parameter
pst	Pseudotime vector
tau	Precision parameter
gamma	Branch responsibility parameter
theta	Factor loading parameter
eta	Factor loading parameter
chi	ARD-like precision
w	Branch responsibility prior (simplex)
tau_c	Hyperparameter
r	Hyperparameter
alpha	Hyperparameter
beta	Hyperparameter
theta_tilde	Hyperparameter
eta_tilde	Hyperparameter
tau_theta	Hyperparameter
tau_eta	Hyperparameter
alpha_chi	Hyperparameter
beta_chi	Hyperparameter
zero_inflation	Logical - was zero inflation modelled?
lambda	Zero inflation parameter

**Value**

The posterior log-likelihood.

---

print.mfa	<i>Print an mfa fit</i>
-----------	-------------------------

---

**Description**

Print an mfa fit

**Usage**

```
## S3 method for class 'mfa'
print(x, ...)
```

**Arguments**

x                    An MFA fit returned by mfa  
 ...                 Additional arguments

**Value**

A string representation of an mfa object.

**Examples**

```
synth <- create_synthetic(C = 20, G = 5)
m <- mfa(synth$X)
print(m)
```

---

summary.mfa

*Summarise an mfa fit*

---

**Description**

Returns summary statistics of an mfa fit, including MAP pseudotime and branch allocations along with uncertainties.

**Usage**

```
## S3 method for class 'mfa'
summary(object, ...)
```

**Arguments**

object              An MFA fit returned by a call to mfa  
 ...                 Additional arguments

**Value**

A data\_frame with the following columns:

- pseudotime The MAP pseudotime estimate
- branch The MAP branch estimate
- branch\_certainty The proportion of traces for which the cell is assigned to its MAP branch
- pseudotime\_lower The lower bound on the 95 (HPD) credible interval
- pseudotime\_upper The upper bound on the 95

**Examples**

```
synth <- create_synthetic(C = 20, G = 5)
m <- mfa(synth$X)
ms <- summary(m)
```

---

to_ggmcmc	<i>Turn a trace list to a ggmcmc object</i>
-----------	---------------------------------------------

---

**Description**

Turn a trace list to a ggmcmc object

**Usage**

```
to_ggmcmc(g)
```

**Arguments**

g                    A list of trace matrices

**Value**

The trace list converted into a ggs object for input to ggmcmc.

---

transient	<i>Transient mean function</i>
-----------	--------------------------------

---

**Description**

Transient mean function

**Usage**

```
transient(t, location = 0.5, scale = 0.01, reverse = FALSE)
```

**Value**

Transient mean function given the parameters.

# Index

## \* internal

- cs\_sigmoid, 4
- log\_sum\_exp, 5
- map\_branch, 5
- mcmcify, 6
- posterior, 10
- transient, 13

- calculate\_chi, 2
- create\_synthetic, 3
- cs\_sigmoid, 4

- empirical\_lambda, 4

- log\_sum\_exp, 5

- map\_branch, 5
- mcmcify, 6
- mfa, 6

- plot\_chi, 8
- plot\_dropout\_relationship, 9
- plot\_mfa\_autocorr, 9
- plot\_mfa\_trace, 10
- posterior, 10
- print.mfa, 11

- summary.mfa, 12

- to\_ggmcmc, 13
- transient, 13