# Package 'banocc'

September 26, 2024

**Type** Package

**Title** Bayesian ANalysis Of Compositional Covariance

**Version** 1.29.0

**Date** 2022-04-20

**Maintainer** George Weingart <george.weingart@gmail.com>,
Curtis Huttenhower <chuttenh@hsph.harvard.edu>

**Description** BAnOCC is a package designed for compositional data, where
each sample sums to one. It infers the approximate covariance
of the unconstrained data using a Bayesian model coded with
`rstan`. It provides as output the `stanfit` object as well as
posterior median and credible interval estimates for each
correlation element.

**License** MIT + file LICENSE

**Depends** R (>= 3.5.1), rstan (>= 2.17.4)

**Imports** coda (>= 0.18.1), mvtnorm, stringr

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

**VignetteBuilder** knitr

**RoxygenNote** 5.0.1

**biocViews** ImmunoOncology, Metagenomics, Software, Bayesian

**git_url** https://git.bioconductor.org/packages/banocc

**git_branch** devel

**git_last_commit** 3a8575f

**git_last_commit_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-09-26

**Author** Emma Schwager [aut, cre],
Curtis Huttenhower [aut]

# Contents

---

banocc                      *banocc: A package for Bayesian ANalysis of Compositional Correlation*

---

## Description

BAnOCC is a package for inferring correlations between features in compositional data, where each sample sums to one. It provides one object, `banocc_model` and one function, `run_banocc`

## banocc objects

`banocc_model` has the `stan` model code to be compiled using `rstan::stan`.

## banocc functions

`run_banocc` takes a compiled model, and returns the 'stanfit' object resulting from a call to `rstan::sampling`

`get_banocc_output` takes a 'stanfit' object or the output of `run_banocc` and returns a list with the posterior median and credible interval estimates

---

banocc_model                      *The stan model used in the Bayesian fit*

---

## Description

This is the literal model used for fitting in Stan

## Usage

```
banocc_model
```

## Format

An object of class `character` of length 1.

## Value

The BAnOCC model as a string to be compiled with `rstan::stan_model`

## Examples

```
data(compositions_null)
## Not run:
  compiled_banocc_model <- rstan::stan_model(model_code = banocc_model)

## End(Not run)
```

---

compositions_hard_null

*Simulated compositional data with no feature correlations*

---

### Description

These are the normalized samples corresponding to `counts_hard_null`. They should have a very different correlation structure from the counts. In particular, there should be one strong, positive association which is not present in the count correlation structure.

### Usage

```
compositions_hard_null
```

### Format

A data frame with 1000 rows (compositional samples) and 9 variables (the features)

### Value

A data frame with 1000 compositional samples from 9 features, generated by dividing each row of `counts_hard_null` by its sum.

---

compositions_neg_spike

*Simulated compositional data with a negative count correlation*

---

### Description

These are the normalized data corresponding to `counts_neg_spike`. The count data have one negative feature correlation, but the compositional correlation strucutre should be different.

### Usage

```
compositions_neg_spike
```

### Format

A data frame with 1000 rows (compositional samples) and 9 variables (the features)

### Value

A data frame with 1000 compositional samples from 9 features, generated by dividing each row of `counts_neg_spike` by its sum.

---

compositions_null        *Simulated compositional data with no feature correlations*

---

#### Description

These are the normalized samples corresponding to counts_null. They should have a similar (but not identical) correlation structure.

#### Usage

```
compositions_null
```

#### Format

A data frame with 1000 rows (compositional samples) and 9 variables (the features)

#### Value

A data frame with 1000 compositional samples from 9 features, generated by dividing each row of counts_null by its sum.

---

compositions_pos_spike

*Simulated compositional data with a positive count correlation*

---

#### Description

These are the normalized data corresponding to counts_pos_spike. The count data have one positive feature correlation, but the compositional correlation structure should be different.

#### Usage

```
compositions_pos_spike
```

#### Format

A data frame with 1000 rows (compositional samples) and 9 variables (the features)

#### Value

A data frame with 1000 compositional samples from 9 features, generated by dividing each row of counts_pos_spike by its sum.

---

counts_hard_null *Simulated count data with no feature correlations*

---

### Description

Nine features are draw independently from very different log-normal distributions whose means and variances are positively correlated. This means that the compositions generated from this dataset (see compositions_hard_null) should be have a correlation structure very different from that of these counts.

### Usage

```
counts_hard_null
```

### Format

A data frame with 1000 rows (samples) and 9 variables (the features)

### Value

A data frame with 1000 unconstrained samples from 9 features.

---

counts_neg_spike *Simulated count data with one negative feature correlation*

---

### Description

Nine features are drawn from a log-normal distribution with one negative correlation. The resulting compositions are in compositions_neg_spike

### Usage

```
counts_neg_spike
```

### Format

A data frame with 1000 rows (samples) and 9 variables (the features)

### Value

A data frame with 1000 unconstrained samples from 9 features.

---

counts_null                           *Simulated count data with no feature correlations*

---

## Description

Nine features are drawn independently from similar log-normal distributions to generate null count data. Because the feature distributions are very similar, the compositions generated from this dataset (see compositions_null), should have a correlation structure similar to that of the counts.

## Usage

```
counts_null
```

## Format

A data frame with 1000 rows (the samples) and 9 variables (the features)

## Value

A data frame with 1000 unconstrained samples from 9 features.

---

counts_pos_spike                       *Simulated count data with one positive feature correlation*

---

## Description

Nine features are drawn from a log-normal distribution with one positive correlation. The resulting compositions are in compositions_pos_spike.

## Usage

```
counts_pos_spike
```

## Format

A data frame with 1000 rows (samples) and 9 variables (the features)

## Value

A data frame with 1000 unconstrained samples from 9 features.

get_banocc_output    *Takes a model fit from BAnOCC, evaluates convergence and generates appropriate convergence metrics and inference*

### Description

Takes a model fit from BAnOCC, evaluates convergence and generates appropriate convergence metrics and inference

### Usage

```
get_banocc_output(banoccfit, conf_alpha = 0.05, get_min_width = FALSE,
  calc_snc = TRUE, eval_convergence = TRUE, verbose = FALSE,
  num_level = 0)
```

### Arguments

banoccfit
:   Either a `stanfit` object (the `Fit` element returned by `run_banocc`), or the list returned by a call to `run_banocc`.

conf_alpha
:   The percentage of the posterior density outside the credible interval. That is, a `1-conf_alpha * 100`% credible interval will be returned.

get_min_width
:   A boolean value: should the minimum CI width that includes zero be calculated?

calc_snc
:   Boolean: should the scaled neighborhood criterion be calculated?

eval_convergence
:   Boolean: if 'TRUE', convergence will be evaluated using the Rhat statistic, and the fit output (estimates, credible intervals, etc.) will be missing if this statistic does not indicate convergence.

verbose
:   Print informative statements as the function executes?

num_level
:   The number of indentations to add to the output when `verbose = TRUE`.

### Value

Returns a named list with the following elements:

*CI* The `1-conf_alpha * 100`% credible intervals

*Estimates.median* The correlation estimates, which are the marginal posterior medians

*Min.width* Only present if the `get_min_width` argument is TRUE. The minimum CI width that includes zero for each correlation.

*SNC* Only present if the `calc_snc` argument is TRUE. The scaled neighborhood criterion for each correlation.

*Fit* The `stanfit` object returned by the call to `run_banocc`.

*Data* Only present if the `banoccfit` argument is specified as the output of a call to `run_banocc`. It will be missing if `banoccfit` is specified as a `stanfit` object.

### See Also

`vignette("banocc-vignette")` for more examples.

## Examples

```
data(compositions_null)
  ## Not run:
    compiled_banocc_model <- rstan::stan_model(model_code=banocc_model)
    b_fit <- run_banocc(C=compositions_null,
                           compiled_banocc_model=compiled_banocc_model)
    b_output <- get_banocc_output(banoccfit=b_fit)

  ## End(Not run)
```

---

run_banocc                     *Runs BAnOCC to fit the model and generate appropriate convergence*
                               *metrics and inference.*

---

## Description

Runs BAnOCC to fit the model and generate appropriate convergence metrics and inference.

## Usage

```
run_banocc(compiled_banocc_model, C, n = rep(0, ncol(C)), L = 10 *
  diag(ncol(C)), a = 0.5, b = 0.01, cores = getOption("mc.cores", 1L),
  chains = 4, iter = 50, warmup = floor(iter/2), thin = 1,
  init = NULL, control = NULL, verbose = FALSE, num_level = 0)
```

## Arguments

compiled_banocc_model

                The compiled stan model (as with `stan_model(model_code = banocc_model)`).

C               The dataset as a data frame or matrix. This should be N by P with N samples as
                the rows and P features as the columns.

n               The prior mean for m; vectors of length less than P (the number of features/columns
                of C) will be recycled.

L               The prior variance-covariance for m (must be positive-definite with dimension
                PxP where P=number of features/columns in C), or a vector of length p of vari-
                ances for m. If a vector of length less than P is given, it will be recycled.

a               The shape parameter of a gamma distribution (the prior on the shrinkage param-
                eter lambda)

b               The rate parameter of a gamma distribution (the prior on the shrinkage parameter
                lambda)

cores           Number of cores to use when executing the chains in parallel, which defaults to
                1 but we recommend setting the `mc.cores` option to be as many processors as
                the hardware and RAM allow (up to the number of chains).

chains          A positive integer specifying the number of Markov chains. The default is 4.

iter            A positive integer specifying the number of iterations for each chain (including
                warmup). The default is 2000.

| | |
|---|---|
| warmup | A positive integer specifying the number of warmup (aka burnin) iterations per chain. If step-size adaptation is on (which it is by default), this also controls the number of iterations for which adaptation is run (and hence these warmup samples should not be used for inference). The number of warmup iterations should not be larger than iter and the default is iter/2. |
| thin | A positive integer specifying the period for saving samples. The default is 1, which is usually the recommended value. |
| init | The initial values as a list (see [sampling](#) in the rstan package). Default value is NULL, which means that initial values are sampled from the priors for parameters m and lambda while O is set to the identity matrix. |
| control | A named list of parameters to control the sampler's behavior. See the details in the documentation for the control argument in [stan](#). |
| verbose | Print informative statements as the function executes? |
| num_level | The number of indentations to add to the output when verbose = TRUE. |

## Value

Returns a named list with the following elements:

*Data* The data formatted as a named list that includes the input data (C) and the prior parameters (n, L, a, b)

*Fit* The stanfit object returned by the call to [sampling](#)

## See Also

vignette("banocc-vignette") for more examples.

## Examples

```
  data(compositions_null)
  ## Not run:
    compiled_banocc_model <- rstan::stan_model(model_code=banocc_model)
    b_stanfit <- run_banocc(C=compositions_null,
                            compiled_banocc_model=compiled_banocc_model)

## End(Not run)
```

# Index