

# Package ‘CNVRG’

September 16, 2020

**Title** Dirichlet-Multinomial Modelling of Relative Abundance Data

**Version** 0.1

**Maintainer** Joshua Harrison <joshua.grant.harrison@gmail.com>

## Description

Implements Dirichlet multinomial modelling of relative abundance data using functionality provided by the 'Stan' software. The purpose of this package is to provide a user friendly way to interface with 'Stan' that is suitable for those new to modelling. For more regarding the modelling mathematics and computational techniques we use see our publication in Molecular Ecology Resources titled "Dirichlet multinomial modelling outperforms alternatives for analysis of ecological count data" (Harrison et al. 2020 <doi:10.1111/1755-0998.13128>).

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Biarch** true

**Depends** R (>= 3.4.0)

**Imports** methods, Rcpp (>= 0.12.0), rstan (>= 2.18.1), vegan

**LinkingTo** BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0),  
RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

**SystemRequirements** GNU make

**Repository** CRAN

**NeedsCompilation** yes

**Author** Joshua Harrison [aut, cre] (<<https://orcid.org/0000-0003-2524-0273>>),  
Vivaswat Shastry [aut] (<<https://orcid.org/0000-0002-7294-5607>>),  
W. John Calder [aut] (<<https://orcid.org/0000-0002-8923-1803>>),  
C. Alex Buerkle [aut] (<<https://orcid.org/0000-0003-4222-8858>>)

**Date/Publication** 2020-09-16 09:40:06 UTC

R topics documented:

CNVRG-package . . . . .	2
diff_abund . . . . .	2
diversity_calc . . . . .	3
extract_point_estimate . . . . .	4
fungi . . . . .	5
isd_transform . . . . .	6
varHMC . . . . .	7
varInf . . . . .	8
<b>Index</b>	<b>10</b>

---

CNVRG-package	<i>The 'CNVRG' package.</i>
---------------	-----------------------------

---

Description

This package implements Dirichlet multinomial modeling of relative abundance data using functionality provided by the 'Stan' software. The purpose of this package is to provide a user friendly way to interface with 'Stan' that is suitable for those new modelling.

References

Stan Development Team (2018). RStan: the R interface to Stan. R package version 2.18.2.

---

diff_abund	<i>@encoding UTF-8</i>
------------	------------------------

---

Description

Calculate features with different abundances between treatment groups

Usage

diff\_abund(model\_output, countData)

Arguments

- |              |  |
|--------------|--|
| model_output | Fitted 'Stan' object.  |
| countData    | Dataframe of count data that was modelled. Should be exactly the same as those data modelled! The first field should be sample name and integer count data should be in all other fields. This is passed in so that the names of fields can be used to make the output of differential relative abundance testing more readable. |

**Details**

This function determines which features within the matrix, be they taxa or molecules, differ in relative abundance among treatment groups. Pass in a model object, with samples for pi parameters, and the number of treatment groups. This function only works for pi parameters.

The output of this function gives the proportion of samples that were greater than zero after subtracting two posterior distributions. Therefore, values that are very large or very small denote a high certainty that the distributions subtracted differ.

**Value**

A dataframe with the first field denoting the treatment comparison (e.g., treatment 1 vs. 2) and subsequent fields stating the proportion of samples from the posterior that were greater than zero.

**Examples**

```
com_demo <- matrix(0, nrow = 10, ncol = 10)
com_demo[1:5,] <- c(rep(3,5), rep(7,5)) #Alternates 3 and 7
com_demo[6:10,] <- c(rep(7,5), rep(3,5)) #Reverses alternation
out <- varInf(com_demo, starts = c(1,6), ends=c(5,10))
diff_abund_test <- diff_abund(model_output = out, countData = com_demo)
```

diversity\_calc

*@encoding UTF-8 Calculate diversity entropies for each replicate***Description**

Calculate Shannon's or Simpson's entropies for each replicate while propagating uncertainty in relative abundance estimates through calculations.

**Usage**

```
diversity_calc(
  model_output,
  countData,
  params = "pi",
  entropy_measure = "shannon",
  equivalents = T
)
```

**Arguments**

model_output	Object from fitted 'Stan' model that has samples describing posteriors of focal parameters.
countData	Dataframe of count data that was modelled. Should be exactly the same as those data modelled! The first field should be sample name and integer count data should be in all other fields. This is passed in so that the names of fields can be used to make the output of differential relative abundance testing more readable.

params	Parameter for which to calculate diversity, can be 'p' or 'pi' or both (e.g., c("pi","p"))
entropy_measure	Diversity entropy to use, can be one of 'shannon' or 'simpson'
equivalents	Convert entropies into number equivalents. Defaults to true. See Jost (2006), "Entropy and diversity"

**Value**

A list that has samples from posterior distributions of entropy metrics

**Examples**

```
com_demo <- matrix(0, nrow = 10, ncol = 10)
com_demo[1:5,] <- c(rep(3,5), rep(7,5)) #Alternates 3 and 7
com_demo[6:10,] <- c(rep(7,5), rep(3,5)) #Reverses alternation
names(com_demo) <- rep("name", 10)
out <- varInf(com_demo, starts = c(1,6), ends=c(5,10))
diversity_calc(model_output = out, params = c("pi", "p"),
countData = com_demo, entropy_measure = 'shannon')
```

---

extract\_point\_estimate

*@encoding UTF-8 Extract point estimates of pi parameters*

---

**Description**

Takes the mean value of pi parameters for each feature and separately by treatment group.

**Usage**

```
extract_point_estimate(modelOut, countData, treatments)
```

**Arguments**

modelOut	Fitted Stan object
countData	The count data modelled. Must be in exactly the same format.
treatments	An integer describing how many treatment groups were modelled.

**Value**

A dataframe specifying point estimates for each feature in each treatment group.

**Examples**

```
com_demo <- matrix(0, nrow = 10, ncol = 10)
com_demo[1:5,] <- c(rep(3,5), rep(7,5)) #Alternates 3 and 7
com_demo[6:10,] <- c(rep(7,5), rep(3,5)) #Reverses alternation
out <- varInf(com_demo, starts = c(1,6), ends=c(5,10))
extract_point_estimate(modelOut = out, countData = com_demo, treatments = 2)
```

---

fungi*Fungal endophytes of Astragalus lentiginosus grown near Reno, NV*

---

**Description**

Fungal endophytes of Astragalus lentiginosus grown near Reno, NV

**Usage**

fungi

**Format**

A data frame with columns:

**treatment** A categorical variable describing if a plant was treated with a slurry of endophyte inoculum and whether it was positive or negative for Alternaria fulva.

**Otu10** Contains count data.

**Otu100** Contains count data.

**Otu11** Contains count data.

**Otu12** Contains count data.

**Otu4** Contains count data.

**Otu40** Contains count data.

**Otu42** Contains count data.

**Otu54** Contains count data.

**Otu58** Contains count data.

**Otu6** Contains count data.

**Otu62** Contains count data.

**Otu7** Contains count data.

**Otu70** Contains count data.

**Otu71** Contains count data.

**Otu72** Contains count data.

**Otu74** Contains count data.

**Otu76** Contains count data.

**Otu77** Contains count data.

**Otu79** Contains count data.

**Otu86** Contains count data.

**Otu9** Contains count data.

**Otu92** Contains count data.

**Otu94** Contains count data.

**Otu96** Contains count data.

**Otu97** Contains count data.

**Otu99** Contains count data.

**Source**

Joshua G. Harrison, <https://www.biorxiv.org/content/10.1101/608729v1.full>

**Examples**

```
## Not run:
fungi

## End(Not run)
```

---

isd_transform	<i>@encoding UTF-8 Transform data into estimates of absolute abundances using an ISD</i>
---------------	--

---

**Description**

If an internal standard (ISD) has been added to samples such that the counts for that standard are representative of the same absolute abundance, then the ISD can be used to transform relative abundance data into estimates of absolute abundances (Harrison et al. 2020). This function performs this division while preserving uncertainty in relative abundance estimates of both the ISD and the other features present.

**Usage**

```
isd_transform(model_output, isd_index, countData)
```

**Arguments**

model_output	A fitted 'Stan' object.
isd_index	The index for the field with information for the internal standard.
countData	The count data modelled. Must be in exactly the same format as those modelled.

**Details**

An index for the ISD must be provided. This should be the field index that corresponds with the ISD. Remember that the index should mirror what has been modelled in that it there is NOT an initial field that contains the sample name. If the wrong index is passed in, the output of this function will be incorrect, but there will not be a fatal error or warning.

A simple check is to examine the output and make sure that the field that should correspond with the ISD is one (signifying that the ISD was divided by itself).

Harrison et al. 2020. The quest for absolute abundance: the use of internal standards for DNA-based community ecology. Molecular Ecology Resources.

**Value**

A dataframe specifying point estimates for each feature in each treatment group.

## Examples

```
com_demo <- matrix(0, nrow = 10, ncol = 10)
com_demo[1:5,] <- c(rep(3,5), rep(7,5)) #Alternates 3 and 7
com_demo[6:10,] <- c(rep(7,5), rep(3,5)) #Reverses alternation
names(com_demo) <- rep("name", 10)
out <- varInf(com_demo, starts = c(1,6), ends=c(5,10))
transformed_data <- isd_transform(model_output = out, countData = com_demo, isd_index = 2)
```

---

varHMC

*@encoding UTF-8 Perform Hamiltonian Monte Carlo sampling*


---

## Description

This function uses a compiled Dirichlet-multinomial model and performs Hamiltonian Monte Carlo sampling of posteriors using 'Stan'. After sampling it is important to check convergence. Use the summary function and shinystan to do this. If you use this function then credit 'Stan' and 'RStan' along with this package. Warning: data must be input in the correct organized format or this function will not provide accurate results. See vignette if you are unsure how to organize data. Warning: depending upon size of data to be analyzed this function can take a very long time to run.

## Usage

```
varHMC(
  countData,
  starts,
  ends,
  algorithm = "NUTS",
  chains = 2,
  burn = 500,
  samples = 1000,
  thinning_rate = 2,
  cores = 1,
  params_to_save = c("pi", "p")
)
```

## Arguments

- |           |   |
|-----------|---|
| countData | A matrix or data frame of counts, must only include numeric data. Data should be arranged so that the first n rows correspond to one treatment group and the next n rows correspond with the next treatment group, and so on. The row indices for the first and last sample in these groups are fed into this function via 'starts' and 'ends'. |
| starts    | A vector defining the indices that correspond to the first sample in each treatment group.  |
| ends      | A vector defining the indices that correspond to the last sample in each treatment group.   |

algorithm	The algorithm to use when sampling. Either 'NUTS' or 'HMC' or 'Fixed_param'. If unsure, then pick NUTS. This is "No U-turn sampling". The abbreviation is from 'Stan'.
chains	The number of chains to run.
burn	The warm-up or 'burn-in' time.
samples	How many samples from the posterior to save.
thinning_rate	Thinning rate to use during sampling.
cores	The number of cores to use.
params_to_save	The parameters from which to save samples. Can be 'p', 'pi', 'theta'.

### Value

A fitted 'Stan' object that includes the samples from the parameters designated.

### Examples

```
com_demo <- matrix(0, nrow = 10, ncol = 10)
com_demo[1:5,] <- c(rep(3,5), rep(7,5)) #Alternates 3 and 7
com_demo[6:10,] <- c(rep(7,5), rep(3,5)) #Reverses alternation
names(com_demo) <- rep("name", 10)
#These are toy data, many more samples, multiple chains, and a longer burn
#are likely advisable for real data.
fitstan_HMC <- varHMC(com_demo, starts = c(1,6),
ends=c(5,10),
chains = 1,
burn = 100,
samples = 150,
thinning_rate = 2)
```

---

varInf

---

*@encoding UTF-8 Perform variational inference sampling*


---

### Description

This function uses a compiled Dirichlet multinomial model and performs variational inference estimation of posteriors using 'Stan'. Evaluating the performance of variational inference is currently under development per our understanding. Please roll over to the 'Stan' website and see if new diagnostics are available. If you use this function then credit 'Stan' and 'RStan' along with this package.

### Usage

```
varInf(
  countData,
  starts,
  ends,
```



```

    algorithm = "meanfield",
    output_samples = 500,
    params_to_save = c("pi", "p")
  )

```

## Arguments

countData	A matrix or data frame of counts, must only include numeric data. Data should be arranged so that the first $n$ rows correspond to one treatment group and the next $n$ rows correspond with the next treatment group, and so on. The row indices for the first and last sample in these groups are fed into this function via 'starts' and 'ends'.
starts	A vector defining the indices that correspond to the first sample in each treatment group.
ends	A vector defining the indices that correspond to the last sample in each treatment group.
algorithm	The algorithm to use when performing variational inference. Either 'meanfield' or 'fullrank'. The former is the default.
output_samples	The number of samples from the approximated posterior to save.
params_to_save	The parameters from which to save samples. Can be 'p', 'pi', 'theta'.

## Details

Warning: data must be input in the correct organized format or this function will not provide accurate results. See vignette if you are unsure how to organize data. Warning: depending upon size of data to be analyzed this function can take a very long time to run.

## Value

A fitted 'Stan' object that includes the samples from the parameters designated.

## Examples

```

com_demo <- matrix(0, nrow = 10, ncol = 10)
com_demo[1:5,] <- c(rep(3,5), rep(7,5)) #Alternates 3 and 7
com_demo[6:10,] <- c(rep(7,5), rep(3,5)) #Reverses alternation
names(com_demo) <- rep("name", 10)
varInf(com_demo, starts = c(1,6), ends=c(5,10))

```

# Index

## \* **datasets**

fungi, [5](#)

CNVRG (CNVRG-package), [2](#)

CNVRG-package, [2](#)

diff\_abund, [2](#)

diversity\_calc, [3](#)

extract\_point\_estimate, [4](#)

fungi, [5](#)

isd\_transform, [6](#)

varHMC, [7](#)

varInf, [8](#)