

# Package ‘BCT’

December 7, 2020

**Type** Package

**Title** Bayesian Context Trees for Discrete Time Series

**Version** 1.1

**Date** 2020-12-04

**Author** Ioannis Papageorgiou, Valentinian Mihai Lungu, Ioannis Kontoyiannis

**Maintainer** Valentinian Mihai Lungu <valentinian.mihai@gmail.com>

## Description

An implementation of a collection of tools for exact Bayesian inference with discrete times series. This package contains functions that can be used for prediction, model selection, estimation, and other statistical tasks. Specifically, the functions provided can be used for the exact computation of the prior predictive likelihood of the data, for the identification of the a posteriori most likely (MAP) variable-memory Markov models, for calculating the exact posterior probabilities and the AIC and BIC scores of these models, and for prediction with respect to log-loss and 0-1 loss. All the functions here (except `generate_data`) are implementations of deterministic algorithms that have linear complexity in the length of the input data. Example data sets from finance, genetics and animal communication are also provided. Detailed descriptions of the underlying theory and algorithms can be found in [Kontoyiannis et al. 'Bayesian Context Trees: Modelling and exact inference for discrete time series.' <arXiv:2007.14900> [stat.ME], July 2020].

**License** GPL (>= 2)

**LazyData** true

**Encoding** UTF-8

**Imports** Rcpp (>= 1.0.5), stringr, igraph, grDevices, graphics

**LinkingTo** Rcpp

**Depends** R (>= 4.0)

**RoxygenNote** 7.1.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-12-07 22:30:16 UTC

## R topics documented:

BCT	2
compute_counts	4
CTW	5
draw_models	6
generate_data	6
gene_s	7
kBCT	8
log_loss	10
MAP_parameters	11
ML	12
pewee	13
prediction	13
sars_cov_2	14
show_tree	15
SP500	16
zero_one_loss	17
<b>Index</b>	<b>19</b>

---

BCT

*Bayesian Context Trees (BCT) algorithm*

---

### Description

Finds the maximum a posteriori probability (MAP) tree model.

### Usage

```
BCT(input_data, depth, beta = NULL)
```

### Arguments

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section on how to transform any dataset to a "character" object.
depth	maximum memory length.
beta	hyper-parameter of the model prior. Takes values between 0 and 1. If not initialised in the call function, the default value is $1 - 2^{-m+1}$ , where $m$ is the size of the alphabet; for more information see <a href="#">Kontoyiannis et al. (2020)</a> .

### Value

returns a list object which includes:

Contexts	MAP model given as a list object containing the contexts of its leaves.
Results	a dataframe with the following columns: prior probability, log(prior probability), posterior probability, log(posterior probability), number of leaves, maximum depth, BIC score, AIC score and maximum log-likelihood.

**See Also**[kBCT](#)**Examples**

```
# Finding the MAP model with maximum depth <= 10
# for the SP500 dataset (with default value beta):

BCT(SP500, 10)

# For custom beta (e.g. 0.7):

BCT(SP500, 10, 0.7)

# The type of the input dataset is "character"
# If the dataset is contained within a vector:

q <- c(1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0)

# Convert a vector to a "character" object:
s <- paste(q, collapse = "")

BCT(s, 2)

# Reading a file using the readChar function

# Example 1: The dataset is stored in a .txt file

# fileName <- '~/example_data.txt' # fileName stores the path to the dataset

# s<-readChar(fileName, file.info(fileName)$size)

# Make sure that s does not contain any "\n" at the end of the string
# To remove last entry:
# s<-gsub('.$', '', s)

# To remove any unwanted characters (e.g. "\n"):
# s<-gsub('\n', '', s)

# Example 2: The dataset is stored in a .csv file

# fileName <- '~/example_data.csv' # fileName stores the path to the dataset

# s<-readChar(fileName, file.info(fileName)$size)

# Depending on the running environment,
# s might contain unwanted characters such as: "\n" or "\r\n".
# Remove any unwanted characters (e.g. "\r\n"):
# s<-gsub('\r\n', '', s)

# Always make sure that s does not contain any unwanted characters
```

---

compute_counts	<i>Compute empirical frequencies of all contexts</i>
----------------	--

---

### Description

Computes the count vectors of all contexts up to a certain length ( $D$ ) for a given dataset. The first  $D$  characters are used to construct the initial context and the counting is performed on the remaining characters. These counts are needed for intermediate computations in BCT and kBCT, and can also be viewed as maximum likelihood estimates of associated parameters; see [Kontoyiannis et al. \(2020\)](#).

### Usage

```
compute_counts(input_data, depth)
```

### Arguments

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section of the BCT/kBCT functions on how to transform any dataset to a "character" object.
depth	maximum memory length.

### Value

a list containing the counts of all contexts of length  $\leq$  depth. If a context with a smaller length than the maximum depth is not contained in the output, its associated count vector is 0. 'Root' indicates the empty context.

### See Also

[BCT](#), [generate\\_data](#)

### Examples

```
# For the pewee dataset:  
compute_counts(pewee, 3)
```

---

CTW

*Context Tree Weighting (CTW) algorithm*

---

## Description

Computes the prior predictive likelihood of the data.

## Usage

```
CTW(input_data, depth, beta = NULL)
```

## Arguments

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section of the BCT/kBCT functions on how to transform any dataset to a "character" object.
depth	maximum memory length.
beta	hyper-parameter of the model prior. Takes values between 0 and 1. If not initialised in the call function, the default value is $1 - 2^{-m+1}$ , where $m$ is the size of the alphabet; for more information see <a href="#">Kontoyiannis et al. (2020)</a> .

## Value

returns the natural logarithm of the prior predictive likelihood of the data.

## See Also

[BCT](#), [kBCT](#)

## Examples

```
# For the gene_s dataset with a maximum depth of 10 (with default value of beta):  
CTW(gene_s, 10)  
  
# For custom beta (e.g. 0.8):  
CTW(gene_s, 10, 0.8)
```

---

draw_models	<i>Plot the results of the BCT and kBCT functions</i>
-------------	---

---

**Description**

This function plots the models produced by the BCT and kBCT functions.

**Usage**

```
draw_models(lst)
```

**Arguments**

lst                    output of the BCT/kBCT function.

**Value**

plots of the BCT/kBCT output models.

**See Also**

[show\\_tree](#), [BCT](#), [kBCT](#)

**Examples**

```
# Use the pewee dataset as an example:
q <- BCT(pewee, 5) # maximum depth of 5

draw_models(q)
r <- kBCT(pewee, 5, 3)

# maximum depth of 5, and k = 3 (top 3 a posteriori most likely models)
draw_models(r)
```

---

generate_data	<i>Sequence generator</i>
---------------	---------------------------

---

**Description**

Generates a simulated sequence of data according to a given model and associated parameters. An initial context of length equal to the maximum depth of the model is first generated uniformly and independently, and it is deleted after the desired number of samples has been generated.

**Usage**

```
generate_data(ct_theta, N)
```

**Arguments**

- ct\_theta            a list containing the contexts that specify a model, and also a parameter vector for each context.
- N                    length of the sequence to be generated.

**Value**

a simulated sequence as a "character" object

**See Also**

[BCT](#), [kBCT](#), [MAP\\_parameters](#)

**Examples**

```
# Create a list containing contexts and associated parameters.
d1 <- list("0" = c(0.2, 0.8), "10" = c(0.9, 0.1), "11" = c(0,1))

# The contexts need to correspond to the leaves of a proper tree.
# The key of each vector is the context.
# For example:
# For context "0":  $P(x_{i+1} = 0 \mid x_i = 0) = 0.2$ 
# and  $P(x_{i+1} = 1 \mid x_i = 0) = 0.8$ 

# If a dataset containing only letters is desired:
d2 <- list("ab" = c(0.3, 0.7), "b" = c(0.8, 0.2), "aa" = c(0.5,0.5))

# Generate data from d2
gd <- generate_data(d2, 10000)

# Use the BCT function to find the MAP model
BCT(gd, 10) # maximum depth of 10

# or the kBCT function can be used:
kBCT(gd, 10, 5) # maximum depth of 10 and top 5 models
```

---

gene\_s

*SARS-CoV-2 gene S*

---

**Description**

This dataset contains the spike (S) gene, in positions 21,563–25,384 of the SARS-CoV-2 genome. The importance of this gene is that it codes for the surface glycoprotein whose function was identified in Yan et al. (2020) and Lan et al. (2020) as critical, in that it binds onto the Angiotensin Converting Enzyme 2 (ACE2) receptor on human epithelial cells, giving the virus access to the cell and thus facilitating the COVID-19 disease. The gene sequence is mapped to the alphabet 0,1,2,3 via the obvious map A->0, C->1, G->2, T->3.

**Usage**

```
gene_s
```

**Format**

An object of class "character".

**References**

Wu, F., S. Zhao, B. Yu, et al. (2020). A new coronavirus associated with human respiratory disease in China. *Nature* 579(7798), 265–269. ([PMC](#))

Yan, R., Y. Zhang, Y. Li, L. Xia, Y. Guo, and Q. Zhou (2020). Structural basis for the recognition of SARS-CoV-2 by full-length human ACE2. *Science* 367(6485), 1444–1448. ([aaas](#))

Lan, J., J. Ge, J. Yu, et al. (2020). Structure of the SARS-CoV-2 spike receptor-binding domain bound to the ACE2 receptor. *Nature* 581, 215–220. ([nature](#))

**Examples**

```
BCT(gene_s, 5)
```

---

 kBCT

---

*k-Bayesian Context Trees (kBCT) algorithm*


---

**Description**

Finds the top k a posteriori most likely tree models.

**Usage**

```
kBCT(input_data, depth, k, beta = NULL)
```

**Arguments**

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section on how to transform any dataset to a "character" object.
depth	maximum memory length.
k	number of the a posteriori most likely tree models to be identified.
beta	hyper-parameter of the model prior. Takes values between 0 and 1. If not initialised in the call function, the default value is $1 - 2^{-m+1}$ , where $m$ is the size of the alphabet; for more information see: <a href="#">Kontoyiannis et al. (2020)</a> .



**Value**

a list object which includes:

Contexts	top k a posteriori most likely models. Each model given as a list object containing the contexts of its leaves.
Results	a dataframe with the following columns: prior probability, log(prior probability), posterior probability, log(posterior probability), posterior odds, number of leaves, maximum depth, BIC score, AIC score and maximum log-likelihood.

**See Also**

[BCT](#)

**Examples**

```
# Finding the first 5 a posteriori most likely models with maximum depth <= 5
# for the SP500 dataset (with default value beta):

kBCT(SP500, 5, 2)

# For custom beta (e.g. 0.8):

kBCT(SP500, 5, 2, 0.8)

# The type of the input dataset is "character"
# If the dataset is contained within a vector:

q <- c(1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0)

# Convert a vector to a "character" object:
s <- paste(q, collapse = "")

kBCT(s, 2, 2)

# Reading a file using the readChar function

# Example 1: The dataset is stored in a .txt file

# fileName <- '~/example_data.txt' # fileName stores the path to the dataset

# s<-readChar(fileName, file.info(fileName)$size)

# Make sure that s does not contain any "\n" at the end of the string
# To remove last entry:
# s<-gsub('.$', '', s)

# To remove any unwanted characters (e.g. "\n"):
# s<-gsub('\n', '', s)

# Example 2: The dataset is stored in a .csv file
```

```
# fileName <- '~/example_data.csv' # fileName stores the path to the dataset

# s<-readChar(fileName, file.info(fileName)$size)

# Depending on the running environment,
# s might contain unwanted characters such as: "\n" or "\r\n".
# Remove any unwanted characters (e.g. "\r\n"):
# s<-gsub('\r\n', '', s)

# Always make sure that s does not contain any unwanted characters
```

---

log\_loss

*Calculating the log-loss incurred in prediction*


---

### Description

Compute the log-loss incurred in BCT prediction with memory length  $D$ . Given an initial context  $(x_{-D+1}, \dots, x_0)$  and training data  $(x_1, \dots, x_n)$ , the log-loss is computed in sequentially predicting the test data  $(x_{n+1}, \dots, x_{n+T})$ . The function outputs the cumulative, normalized (per-sample) log-loss, at each prediction step; for more information see [Kontoyiannis et al.\(2020\)](#).

### Usage

```
log_loss(input_data, depth, train_size, beta = NULL)
```

### Arguments

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section of BCT/kBCT functions on how to transform any dataset to a "character" object.
depth	maximum memory length.
train_size	number of samples used in the training set. The training set size should be at least equal to the depth.
beta	hyper-parameter of the model prior. Takes values between 0 and 1. If not initialised in the call function, the default value is $1 - 2^{-m+1}$ , where $m$ is the size of the alphabet; for more information see <a href="#">Kontoyiannis et al. (2020)</a> .

### Value

returns a vector containing the averaged log-loss incurred in the sequential prediction at each time-step.

### See Also

[prediction](#), [zero\\_one\\_loss](#)

**Examples**

```
# Compute the log-loss in the prediction of the last 10 elements
# of a dataset.
log_loss(pewee, 5, nchar(pewee) - 10)

# For custom beta (e.g. 0.7):
log_loss(pewee, 5, nchar(pewee) - 10, 0.7)
```

---

MAP_parameters	<i>Parameters of the MAP model</i>
----------------	------------------------------------

---

**Description**

Returns the parameters of each leaf contained in the MAP model.

**Usage**

```
MAP_parameters(input_data, depth, beta = NULL)
```

**Arguments**

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section of BCT/kBCT functions on how to transform any dataset to a "character" object.
depth	maximum memory length.
beta	hyper-parameter of the model prior. Takes values between 0 and 1. If not initialised in the call function, the default value is $1 - 2^{-m+1}$ , where $m$ is the size of the alphabet; for more information see <a href="#">Kontoyiannis et al. (2020)</a> .

**Value**

list of parameters for each of the context within the MAP model.

**See Also**

[BCT](#), [kBCT](#), [generate\\_data](#)

**Examples**

```
# Use the gene_s dataset:
q <- BCT(gene_s, 10)
expected_contexts <- q[['Contexts']]
expected_contexts
# [1] "3" "1" "0" "23" "20" "21" "22"

# For default beta:
v <- MAP_parameters(gene_s, 10)
```

```
# For custom beta (e.g. 0.8):
MAP_parameters(gene_s, 10, 0.8)

# generate a sequence of data using the generate_data function
s <- generate_data(v, 20000)

# Use BCT:
r <- BCT(s, 10)

# Check the resulting contexts:
r[['Contexts']]
# [1] "3" "0" "1" "20" "22" "23" "21"

# The resulting contexts are as expected
```

---

ML

*Maximum Likelihood*

---

### Description

Computes the logarithm of the likelihood of the observations, maximised over all models and parameters.

### Usage

```
ML(input_data, depth)
```

### Arguments

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section of the BCT/kBCT functions on how to transform any dataset to a "character" object.
depth	maximum memory length.

### Value

returns the natural logarithm of the maximum likelihood.

### See Also

[BCT](#), [kBCT](#)

### Examples

```
# Computing the maximum likelihood of the gene_s dataset
# with a maximum depth of 5:
ML(gene_s, 5)
```

---

pewee

*Pewee birdsong*

---

### Description

The twilight song of the wood pewee bird can be described as a sequence consisting of an arrangement of musical phrases taken from an alphabet of three specific, distinct phrases. The dataset consists of a single continuous song by a wood pewee, of length  $n = 1327$  phrases.

### Usage

pewee

### Format

An object of class "character".

### References

W. Craig. The song of the wood pewee (*Myiochanes virens* Linnaeus): A study of bird music. New York State Museum Bulletin No. 334. University of the State of New York, Albany, NY, 1943. ([craig](#))

### Examples

BCT(pewee, 5)

---

prediction

*Prediction*

---

### Description

Computes the posterior predictive distribution at each time step, and predicts the next symbol as its most likely value. Given an initial context  $(x_{-D+1}, \dots, x_0)$  and training data  $(x_1, \dots, x_n)$ , the posterior predictive distribution is computed sequentially for the test data  $(x_{n+1}, \dots, x_{n+T})$ . The function outputs the predicted distribution at each time step, along with the most likely symbol; for more information see [Kontoyiannis et al.\(2020\)](#).

### Usage

prediction(input\_data, depth, train\_size, beta = NULL)

**Arguments**

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section on how to transform any dataset to a "character" object.
depth	maximum memory length.
train_size	number of samples used for training.
beta	hyper-parameter of the model prior. Takes values between 0 and 1. If not initialised in the call function, the default value is $1 - 2^{-m+1}$ , where $m$ is the size of the alphabet; for more information see: <a href="#">Kontoyiannis et al. (2020)</a> .

**Value**

returns a "list" containing the posterior predictive distribution at each time step. The last entry in the list, named "Prediction", contains the most likely character at each time step according to the posterior predictive distribution.

**See Also**

[log\\_loss](#), [zero\\_one\\_loss](#)

**Examples**

```
# Predicting the 2 last characters of a dataset using a model with a maximum depth of 5
# The training size is the total number of characters within the dataset minus 2: nchar(pewee) - 2

q <- prediction(pewee, 5, nchar(pewee) - 2)

q
# [[1]]
# [1] 0.56300039 0.05899728 0.37800233

# [[2]]
# [1] 0.08150306 0.76293065 0.15556628

# $Prediction
# [1] "0" "1"

# To access the "Prediction" from result list q:
q[["Prediction"]]

# For custom beta (e.g. 0.8):
prediction(pewee, 5, nchar(pewee) - 10, 0.8)
```

**Description**

The severe acute respiratory syndrome coronavirus, SARS-CoV-2, is the novel coronavirus responsible for the Covid-19 global pandemic in 2019-20. This dataset contains the SARS-CoV-2 genome, available in the GenBank database as the sequence MN908947.3. It consists of  $n = 29903$  base pairs. The gene sequence is mapped to the alphabet 0,1,2,3 via the obvious map A->0, C->1, G->2, T->3.

**Usage**

```
sars_cov_2
```

**Format**

An object of class "character".

**References**

K. Clark, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, and E.W. Sayers. GenBank. Nucleic Acids Research, 44(D1): D67–D72, January 2016. ([ncbi](#))

F. Wu, S. Zhao, B. Yu, et al. A new coronavirus associated with human respiratory disease in China. Nature, 579(7798):265–269, February 2020. ([PubMed](#))

**Examples**

```
BCT(sars_cov_2, 5)
```

---

show\_tree

*Plot tree with given contexts*

---

**Description**

Plots a tree depicting a model with the given set of contexts.

**Usage**

```
show_tree(s)
```

**Arguments**

`s` vector containing the contexts of the leaves of the desired tree.

**Value**

plot of the desired tree model.

**See Also**

[BCT](#), [draw\\_models](#)

**Examples**

```
# Construct an example vector:
r <- c("a", "ab", "aab", "b", "ba")

show_tree(r)

# If the input contains digits:
q <- c(11,1,0)

show_tree(q)
```

---

SP500

*Daily changes in the S&P 500 index*

---

**Description**

This dataset contains the quantised daily changes  $x_i$  in the Standard & Poor's index price, from January 2, 1928 until October 7, 2016. The price changes are quantised to 7 values as follows: If the change between two successive trading days ( $i - 1$ ) and  $i$  is smaller than  $-3\%$ ,  $x_i$  is set equal to 0; if the change is between  $-3\%$  and  $-2\%$ ,  $x_i = 1$ ; for changes in the intervals  $(-2\%, -1\%]$ ,  $(-1\%, 1\%]$ ,  $(1\%, 2\%]$ , and  $(2\%, 3\%]$   $x_i$  is set equal to 2,3,4 and 5, respectively; and for changes greater than  $3\%$ ,  $x_i = 6$ .

**Usage**

SP500

**Format**

An object of class "character".

**References**

Yahoo! finance. ([yahoo\\_finance](#))

**Examples**

BCT(SP500, 10)



zero\_one\_loss

*Calculating the 0-1 loss incurred in prediction***Description**

Compute the 0-1 loss, i.e., the proportion of incorrectly predicted values, incurred in BCT prediction with memory length  $D$ . Given an initial context  $(x_{-D+1}, \dots, x_0)$  and training data  $(x_1, \dots, x_n)$ , the 0-1 loss is computed in sequentially predicting the test data  $(x_{n+1}, \dots, x_{n+T})$ . The function outputs the cumulative, normalized (per-sample) 0-1 loss, at each prediction step; for more information see [Kontoyiannis et al. \(2020\)](#).

**Usage**

```
zero_one_loss(input_data, depth, train_size, beta = NULL)
```

**Arguments**

input_data	the sequence to be analysed. The sequence needs to be a "character" object. See the examples section of kBCT/BCT functions on how to transform any dataset to a "character" object.
depth	maximum memory length.
train_size	number of samples used in the training set. The training set size should be at least equal to the depth.
beta	hyper-parameter of the model prior. Takes values between 0 and 1. If not initialised in the call function, the default value is $1 - 2^{-m+1}$ , where $m$ is the size of the alphabet; for more information see <a href="#">Kontoyiannis et al. (2020)</a>

**Value**

returns a vector containing the averaged number of errors at each timestep.

**See Also**

[log\\_loss](#), [prediction](#)

**Examples**

```
# Use the pewee dataset and look at the last 8 elements:
  substring(pewee, nchar(pewee)-7, nchar(pewee))
# [1] "10001001"

# Predict last 8 elements using the prediction function
pred <- prediction(pewee, 10, nchar(pewee)-8)[["Prediction"]]
# Taking only the "Prediction" vector:

pred
# [1] "1" "0" "0" "1" "1" "0" "0" "1"
```

```
# To transform the result of the prediction function into a "character" object:
paste(pred, collapse = "")
# [1] "10011001"

# As observed, there is only 1 error (the sixth predicted element is 1 instead of a 0).
# Thus, up to the 4th place, the averaged error is 0
# and the sixth averaged error is expected to be 1/4.
# Indeed, the zero_one_loss function yields the expected answer:

zero_one_loss(pewee, 10, nchar(pewee)-8)
# [1] 0.0000000 0.0000000 0.0000000 0.2500000 0.2000000 0.1666667 0.1428571 0.1250000
```

# Index

## \* datasets

gene\_s, [7](#)

pewee, [13](#)

sars\_cov\_2, [14](#)

SP500, [16](#)

BCT, [2](#), [4-7](#), [9](#), [11](#), [12](#), [15](#)

compute\_counts, [4](#)

CTW, [5](#)

draw\_models, [6](#), [15](#)

gene\_s, [7](#)

generate\_data, [4](#), [6](#), [11](#)

kBCT, [3](#), [5-7](#), [8](#), [11](#), [12](#)

log\_loss, [10](#), [14](#), [17](#)

MAP\_parameters, [7](#), [11](#)

ML, [12](#)

pewee, [13](#)

prediction, [10](#), [13](#), [17](#)

sars\_cov\_2, [14](#)

show\_tree, [6](#), [15](#)

SP500, [16](#)

zero\_one\_loss, [10](#), [14](#), [17](#)