Package ‘BTYD’

November 17, 2021

Type Package

Title Implementing BTYD Models with the Log Sum Exp Patch

Version 2.4.3

Maintainer Gabi Huiber <ghuiber@gmail.com>

BugReports https://github.com/ghuiber/BTYD/issues

Description Functions for data preparation, parameter estimation, scoring, and plotting for the BG/BB (Fader, Hardie, and Shang 2010 <doi:10.1287/mksc.1100.0580>), BG/NBD (Fader, Hardie, and Lee 2005 <doi:10.1287/mksc.1040.0098>) and Pareto/NBD and Gamma/Gamma (Fader, Hardie, and Lee 2005 <doi:10.1509/jmkr.2005.42.4.415>) models.

License GPL-3

Collate 'data.R' 'BTYD.R' 'bgbb.R' 'bgnbd.R' 'pnbd.R' 'dc.R' 'spend.R'

Depends R(>= 3.5), hypergeo, optimx, dplyr

Imports Matrix

Suggests knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

Author Lukasz Dziurzynski [aut],
Edward Wadsworth [aut],
Peter Fader [ctb],
Elea McDonnell Feit [ctb],
Daniel McCarthy [aut, ctb],
Bruce Hardie [ctb],
Arun Gopalakrishnan [ctb],
Eric Schwartz [ctb],
Yao Zhang [ctb],
Gabi Huiber [ctb, cre]

NeedsCompilation no

Encoding UTF-8

RoxygenNote 7.1.2

Repository CRAN

Date/Publication 2021-11-17 19:30:02 UTC

R topics documented:

BTYD-package . . . 4
addLogs . . . 6
bgbb.ConditionalExpectedTransactions . . . 7
bgbb.DERT . . . 8
bgbb.EstimateParameters . . . 9
bgbb.Expectation . . . 11
bgbb.HeatmapHoldoutExpectedTrans . . . 12
bgbb.LL . . . 13
bgbb.PAlive . . . 14
bgbb.PlotDropoutRateHeterogeneity . . . 16
bgbb.PlotFrequencyInCalibration . . . 16
bgbb.PlotFrequencyInHoldout . . . 18
bgbb.PlotFreqVsConditionalExpectedFrequency . . . 19
bgbb.PlotRecVsConditionalExpectedFrequency . . . 21
bgbb.PlotTrackingCum . . . 23
bgbb.PlotTrackingInc . . . 25
bgbb.PlotTransactionRateHeterogeneity . . . 26
bgbb.pmf . . . 27
bgbb.pmf.General . . . 28
bgbb.PosteriorMeanDropoutRate . . . 30
bgbb.PosteriorMeanLmProductMoment . . . 31
bgbb.PosteriorMeanTransactionRate . . . 32
bgbb.rf.matrix.DERT . . . 33
bgbb.rf.matrix.LL . . . 35
bgbb.rf.matrix.PosteriorMeanDropoutRate . . . 36
bgbb.rf.matrix.PosteriorMeanTransactionRate . . . 37
bgnbd.cbs.LL . . . 38
bgnbd.ConditionalExpectedTransactions . . . 39
bgnbd.EstimateParameters . . . 41
bgnbd.Expectation . . . 43
bgnbd.ExpectedCumulativeTransactions . . . 44
bgnbd.generalParams . . . 45
bgnbd.LL . . . 46
bgnbd.PAlive . . . 47
bgnbd.PlotDropoutRateHeterogeneity . . . 48
bgnbd.PlotFrequencyInCalibration . . . 49
bgnbd.PlotFreqVsConditionalExpectedFrequency . . . 50
bgnbd.PlotRecVsConditionalExpectedFrequency . . . 52
bgnbd.PlotTrackingCum . . . 54
bgnbd.PlotTrackingInc . . . 55
bgnbd.PlotTransactionRateHeterogeneity . . . 57
bgnbd.pmf . . . 58
bgnbd.pmf.General . . . 59
cdnowElog . . . 60
cdnowSummary . . . 61
dc.BuildCBSFromCBTAndDates . . . 62

dc.BuildCBTFromElog . . . 63
dc.check.model.params . . . 64
dc.compress.cbs . . . 65
dc.CreateFreqCBT . . . 66
dc.CreateReachCBT . . . 67
dc.CreateSpendCBT . . . 68
dc.CumulativeToIncremental . . . 69
dc.DissipateElog . . . 69
dc.ElogToCbsCbt . . . 70
dc.FilterCustByBirth . . . 71
dc.GetFirstPurchasePeriodsFromCBT . . . 72
dc.GetLastPurchasePeriodsFromCBT . . . 73
dc.InputCheck . . . 73
dc.MakeRFmatrixCal . . . 74
dc.MakeRFmatrixHoldout . . . 76
dc.MakeRFmatrixSkeleton . . . 77
dc.MergeCustomers . . . 77
dc.MergeTransactionsOnSameDate . . . 79
dc.PlotLogLikelihoodContour . . . 80
dc.PlotLogLikelihoodContours . . . 83
dc.ReadLines . . . 85
dc.RemoveTimeBetween . . . 86
dc.SplitUpElogForRepeatTrans . . . 87
dc.WriteLine . . . 88
discreteSimElog . . . 88
donationsSummary . . . 89
h2f1 . . . 90
pnbd.cbs.LL . . . 91
pnbd.ConditionalExpectedTransactions . . . 92
pnbd.DERT . . . 93
pnbd.EstimateParameters . . . 95
pnbd.Expectation . . . 97
pnbd.ExpectedCumulativeTransactions . . . 98
pnbd.generalParams . . . 99
pnbd.LL . . . 100
pnbd.PAlive . . . 101
pnbd.Plot.DERT . . . 102
pnbd.PlotDropoutRateHeterogeneity . . . 104
pnbd.PlotFrequencyInCalibration . . . 105
pnbd.PlotFreqVsConditionalExpectedFrequency . . . 106
pnbd.PlotRateHeterogeneity . . . 108
pnbd.PlotRecVsConditionalExpectedFrequency . . . 109
pnbd.PlotTrackingCum . . . 110
pnbd.PlotTrackingInc . . . 112
pnbd.PlotTransactionRateHeterogeneity . . . 114
pnbd.pmf . . . 115
pnbd.pmf.General . . . 116
spend.EstimateParameters . . . 117

spend.expected.value 119
spend.generalParams 120
spend.LL 121
spend.marginal.likelihood 123
spend.plot.average.transaction.value 124
subLogs 125

Index **126**

`BTYD-package` *This project was funded and sponsored by* R*hrefhttps://wca.wharton.upenn.eduWharton Customer Analytics.*

Description

This package implements the BG/BB, BG/NBD and Pareto/NBD models, which capture/project customer purchase patterns in a typical non-contractual setting.

Details

While these models are developed on a customer-by-customer basis, they do not necessarily require data at such a granular level. The Pareto/NBD requires a "customer-by-sufficient-statistic" matrix (CBS), which consists of each customer's frequency, recency (the time of their last transactions) and total time observed - but the timing of each and every transaction (other than the last) is not needed by the model. If, however, you do have the granular data in the form of an event log (which contains at least columns for customer identification and the time of each transaction, and potentially more columns such as transaction amount), this package provides functions to convert it to a CBS. You can use `dc.ReadLines` to get your event log from a comma-delimited file to an event log usable by this package; it is possible to use read.table or read.csv, but formatting will be required afterwards. You can then convert the event log directly to a CBS (for both the calibration and holdout periods) using `dc.ElogToCbsCbt`. As the name suggests, this function also produces a customer-by-time matrix (CBT). This matrix consists of a row for every customer and a column for every date, and is populated by a statistic of your choice (reach, frequency, or spend). It is not necessary for any of the models presented in this package, but is used as a building block to produce the CBS.

The BG/NBD model requires all the same inputs as the Pareto/NBD model.

The BG/BB model requires the same information as the Pareto/NBD model, but as it models discrete transaction opportunities, this information can be condensed into a recency-frequency matrix. A recency-frequency matrix contains a row for every recency/frequency combination in the given time period, and each row contains the number of customers with that recency/frequency combination. Since frequency will always be less than or equal to recency, this matrix will contain (n)(n-1)/2 + 1 rows at most, with n as the number of transaction opportunities (of course, the maximum number of rows for pooled data - for customers with varying numbers of transaction opportunities - will be the sum of the above equation for each unique number of transaction opportunities). You can convert a CBS to recency-frequency matrices using `dc.MakeRFmatrixCal` and `dc.MakeRFmatrixHoldout`.

If you want to test the data contained in the package, or have data formatted as a customer-by-sufficient-statistic or recency-frequency matrix, a good starting place would be `pnbd.EstimateParameters`, `bgnbd.EstimateParameters`, or `bgbb.EstimateParameters`.

Following that, pnbd.PlotFrequencyInCalibration, bgnbd.PlotFrequencyInCalibration and bgbb.PlotFrequencyInCalibration will give a check that the model fits the data in-sample. Further plotting functions, comparing actual and expected results, are labelled "pnbd.Plot...", "bgnbd.Plot..." and "bgbb.Plot...". The building blocks of these functions are also provided: pnbd.LL, bgnbd.LL bgbb.LL, pnbd.pmf, bgnbd.pmf, bgbb.pmf, pnbd.Expectation, bgnbd.Expectation, bgbb.Expectation, pnbd.ConditionalExpectedTransactions, bgnbd.ConditionalExpectedTransactions, and bgbb.ConditionalExpec may be of particular interest.

This package uses the following conventions:

The time period used to estimate the model parameters is called the *calibration period*. Users may be accustomed to this being called the estimation period, or simply being referred to as "in-sample". Function parameter names generally follow this convention: for example, "n.cal" is used to refer to the number of transaction opportunities in the calibration period.

The time period used to validate model performance is called the *holdout period*. Users may be accustomed to this being called the validation period, or simply being referred to as "out-of-sample". Function parameters relating to this time period are generally appended with ".star". For example, n.star is used to refer to the number of transaction opportunities in the holdout period.

As described in the papers referenced below, the BG/BB, BG/NBD and Pareto/NBD models are generally concerned with repeat transactions, not total transactions. This means that a customer's first transaction in the calibration period is usually not part of the data being modeled - this is due to the fact that a new customer generally does not show up "on the company's radar" until after their first purchase has taken place. This means that the modal number of repeat purchases tends to be zero. If your data does not have a relatively large number of customers with zero transactions, but does have a relatively large number of customers with one transaction, and the estimation functions are struggling, the problem is most likely that you are including customers' very first transactions. Some of the data-conversion functions have examples illustrating how to work with data that includes this very first transaction. Note that this does not apply to the holdout period; in the holdout period, we already know about the customer and take all of their previous transactions into account.

Author(s)

Maintainer: Gabi Huiber <ghuiber@gmail.com> [contributor]

Authors:

- Lukasz Dziurzynski
- Edward Wadsworth
- Daniel McCarthy [contributor]

Other contributors:

- Peter Fader [contributor]
- Elea McDonnell Feit [contributor]
- Bruce Hardie [contributor]
- Arun Gopalakrishnan [contributor]
- Eric Schwartz [contributor]
- Yao Zhang [contributor]

References

See https://www.brucehardie.com for papers, notes, and datasets relating to applied probability models in marketing.

Fader, Peter S., and Bruce G.S. Hardie. "A Note on Deriving the Pareto/NBD Model and Related Expressions." November. 2005. Web. http://www.brucehardie.com/notes/008/

Fader, Peter S., Bruce G.S. Hardie, and Ka L. Lee. "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." *Journal of Marketing Research* Vol.42, pp.415-430. November. 2005. http://www.brucehardie.com/papers.html

Fader, Peter S., and Bruce G.S. Hardie. "Deriving an Expression for P (X(t) = x) Under the Pareto/NBD Model." September. 2006. Web. http://www.brucehardie.com/notes/012/

Fader, Peter S., and Bruce G.S. Hardie. "Creating an RFM summary using Excel." December. 2008. Web. http://www.brucehardie.com/notes/022/

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. http://www.brucehardie.com/papers/020/

Jerath, Kinshuk, Peter S. Fader, and Bruce G.S. Hardie. "Customer-Base Analysis on a 'Data Diet': Model Inference Using Repeated Cross-Sectional Summary (RCSS) Data." June. 2011. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1708562 or doi: 10.2139/ssrn.1708562

Fader, Peter S., Bruce G.S. Hardie, and Ka L. Lee. ""Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model." *Marketing Science* Vol.24, pp.275-284. Spring. 2005. http://www.brucehardie.com/papers.html

Fader, Peter S., Hardie, Bruce G.S., and Lee, Ka Lok. "Computing P(alive) Using the BG/NBD Model." December. 2008. Web. http://www.brucehardie.com/notes/021/palive_for_BGNBD.pdf

See Also

Useful links:

- Report bugs at https://github.com/ghuiber/BTYD/issues

`addLogs` *Add Logs*

Description

Given log(a) and log(b), returns log(a + b)

Usage

```
addLogs(loga, logb)
```

Arguments

`loga`	first number in log space.
`logb`	first number in log space.

`bgbb.ConditionalExpectedTransactions` *BG/BB Conditional Expected Transactions*

Description

Calculates the number of expected transactions in the holdout period, conditional on a customer's behavior in the calibration period.

Usage

```
bgbb.ConditionalExpectedTransactions(params, n.cal, n.star, x, t.x)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`n.cal`	number of transaction opportunities in the calibration period. Can also be a vector of calibration period transaction opportunities - see details.
`n.star`	number of transaction opportunities in the holdout period, or a vector of holdout period transaction opportunities.
`x`	the number of repeat transactions made by the customer in the calibration period. Can also be vector of frequencies - see details.
`t.x`	recency - the transaction opportunity in which the customer made their last transaction. Can also be a vector of recencies - see details.

Details

E(X(n, n+n*) | alpha, beta, gamma, delta, x, t.x, n). This function requires the holdout period to immediately follow the calibration period.

`n.cal`, `n.star`, `x`, and `t.x` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

The number of transactions a customer is expected to make in the `n.star` transaction opportunities following the calibration period, conditional on their behavior during the calibration period.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
params <- c(1.20, 0.75, 0.66, 2.78)
# the number of transactions a customer is expected
# to make in the 10 transaction opportunities
# following the calibration period, which consisted
# of 6 transaction opportunities (during which they
# made 3 transactions, the last of which occurred
# in the 4th opportunity)
bgbb.ConditionalExpectedTransactions(params, n.cal=6, n.star=10, x=3, t.x=4)

# We can also use vectors as input:
bgbb.ConditionalExpectedTransactions(params, n.cal=6, n.star=1:10, x=3, t.x=4)
bgbb.ConditionalExpectedTransactions(params, n.cal=6, n.star=10, x=1:4, t.x=4)
```

`bgbb.DERT` *BG/BB Discounted Expected Residual Transactions*

Description

Computes the number of discounted expected residual transactions by a customer, conditional on their behavior in the calibration period.

Usage

```
bgbb.DERT(params, x, t.x, n.cal, d)
```

Arguments

params	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
x	the number of repeat transactions made by the customer in the calibration period. Can also be vector of frequencies - see details.
t.x	recency - the transaction opportunity in which the customer made their last transaction. Can also be a vector of recencies - see details.
n.cal	number of transaction opportunities in the calibration period. Can also be a vector of calibration period transaction opportunities - see details.
d	discount rate.

Details

DERT(d | alpha, beta, gamma, delta, x, t.x, n). This is the present value of the expected future transaction stream for a customer with x transactions and a recency of t.x in n.cal transaction opportunities, discounted by a rate d.

`x`, `t.x`, and `n.cal` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be also be a vector.

Value

The present value of the expected future transaction stream for a particular customer.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web. See equation 14.

Examples

```
params <- c(1.20, 0.75, 0.66, 2.78)
# Compute DERT for a customer who made 3 transactions
# in the calibration period(consisting of 6 transaction
# opportunities), with the last transaction occurring
# during the 4th transaction opportunity, discounted at
# 10%.
bgbb.DERT(params, x=3, t.x=4, n.cal=6, d=0.1)

# We can also compare DERT for several customers:
bgbb.DERT(params, x=1:6, t.x=6, n.cal=6, d=0.1)
```

`bgbb.EstimateParameters` *BG/BB Parameter estimation*

Description

Estimates parameters for the BG/BB model.

Usage

```
bgbb.EstimateParameters(
  rf.matrix,
  par.start = c(1, 1, 1, 1),
  max.param.value = 1000
)
```

Arguments

`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`par.start`	initial BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`max.param.value`	the upper bound on parameters.

Details

The best-fitting parameters are determined using the `bgbb.rf.matrix.LL` function. The sum of the log-likelihood for each customer (for a set of parameters) is maximized in order to estimate paramaters.

A set of starting parameters must be provided for this method. If no parameters are provided, (1,1,1,1) is used as a default. It may be useful to use starting values for parameters that represent your best guess of the heterogeneity in the transaction and dropout rates of customers. It may be necessary to run the estimation from multiple starting points to ensure that it converges. To compare the log-likelihoods of different parameters, use `bgbb.rf.matrix.LL`.

The lower bound on the parameters to be estimated is always zero, since BG/BB parameters cannot be negative. The upper bound can be set with the max.param.value parameter.

Value

Vector of estimated paramaters.

See Also

`bgbb.rf.matrix.LL`

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)
# log-likelihood of estimated parameters
bgbb.rf.matrix.LL(est.params, rf.matrix)
```

`bgbb.Expectation` *BG/BB Expectation*

Description

Returns the number of transactions that a randomly chosen customer (for whom we have no prior information) is expected to make in the first n transaction opportunities.

Usage

```
bgbb.Expectation(params, n)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`n`	number of transaction opportunities; may also be a vector.

Details

E(X(n) | alpha, beta, gamma, delta)

Value

Mean of the BG/BB probability mass function.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
params <- c(1.20, 0.75, 0.66, 2.78)
# Expected number of transactions that a randomly chosen customer
# will make in the first 10 transaction opportunities.
bgbb.Expectation(params, n=10)

# We can also compare expected transactions over time:
bgbb.Expectation(params, n=1:10)
```

bgbb.HeatmapHoldoutExpectedTrans
BG/BB Heatmap of Holdout Period Expected Transactions

Description

Plots a heatmap based on the conditional expected holdout period frequency for each recency-frequency combination in the calibration period.

Usage

```
bgbb.HeatmapHoldoutExpectedTrans(
  params,
  n.cal,
  n.star,
  xlab = "Recency",
  ylab = "Frequency",
  xticklab = NULL,
  title = "Heatmap of Conditional Expected Transactions"
)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`n.cal`	number of transaction opportunities in the calibration period.
`n.star`	number of transaction opportunities in the holdout period.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

E(X(n, n+n*) | alpha, beta, gamma, delta, x, t.x, n). This function requires the holdout period to immediately follow the calibration period.

Value

A matrix containing the conditional expected transactions in the holdout period for each recency-frequency combination in the calibration period. The rows represent calibration period frequencies, and the columns represent calibration period recencies.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See Also

`bgbb.ConditionalExpectedTransactions`

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# Plot a heatmap of conditional expected transactions in
# a holdout period of 5 transaction opportunities, given
# that the calibration period consisted of 6 transaction
# opportunities.
bgbb.HeatmapHoldoutExpectedTrans(est.params, n.cal=6, n.star=5)
```

`bgbb.LL` *BG/BB Log-Likelihood*

Description

Calculates the log-likelihood of the BG/BB model.

Usage

```
bgbb.LL(params, x, t.x, n.cal)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`x`	the number of repeat transactions made by the customer in the calibration period. Can also be vector of frequencies - see details.
`t.x`	recency - the transaction opportunity in which the customer made their last transaction. Can also be a vector of recencies - see details.
`n.cal`	number of transaction opportunities in the calibration period. Can also be a vector of calibration period transaction opportunities - see details.

Details

x, t.x, and n.cal may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be also be a vector.

Value

A vector of log-likelihoods as long as the longest input vector (x, t.x, or n.cal).

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
params <- c(1.20, 0.75, 0.66, 2.78)

# Returns the log likelihood of the parameters for a customer who
# made 3 transactions in a calibration period with 6 transaction opportunities,
# with the last transaction occurring during the 4th transaction opportunity.
bgbb.LL(params, x=3, t.x=4, n.cal=6)

# We can also give vectors as function parameters:
set.seed(7)
x <- sample(1:3, 10, replace = TRUE)
t.x <- sample(3:5, 10, replace = TRUE)
n.cal <- rep(5, 10)
bgbb.LL(params, x, t.x, n.cal)
```

`bgbb.PAlive` *BG/BB P(Alive)*

Description

Uses BG/BB model parameters and a customer's past transaction behavior to return the probability that they will be alive in the transaction opportunity following the calibration period.

Usage

```
bgbb.PAlive(params, x, t.x, n.cal)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`x`	the number of repeat transactions made by the customer in the calibration period. Can also be vector of frequencies - see details.
`t.x`	recency - the transaction opportunity in which the customer made their last transaction. Can also be a vector of recencies - see details.
`n.cal`	number of transaction opportunities in the calibration period. Can also be a vector of calibration period transaction opportunities - see details.

Details

`x`, `t.x`, and `n.cal` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

P(alive at n+1 | alpha, beta, gamma, delta, x, t.x, n)

Value

Probability that the customer is alive at the (n+1)th transaction opportunity. If `x`, `t.x`, and/or `n.cal` are of length greater than one, then this will be a vector of probabilities (containing one element matching each element of the longest input vector).

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
params <- c(1.20, 0.75, 0.66, 2.78)

# The probability that a customer who made 3 transactions in
# the calibration period (which consisted of 6 transaction
# opportunities), with the last transaction occurring at the
# 4th transaction opportunity, is alive at the 7th transaction
# opportunity
bgbb.PAlive(params, x=3, t.x=4, n.cal=6)

# The input parameters may also be vectors:
bgbb.PAlive(params, x=1, t.x=1:6, n.cal=6)
```

bgbb.PlotDropoutRateHeterogeneity

BG/BB Plot Dropout Rate Heterogeneity

Description

Plots and returns the estimated beta distribution of Theta (customers' propensities to drop out).

Usage

```
bgbb.PlotDropoutRateHeterogeneity(params)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.

Details

This returns the distribution of each customer's geometric parameter that determines their lifetime (using the BG/BB assumption that a customer's lifetime can be modeled with a geometric distribution).

Value

Distribution of customers' propensities to drop out.

Examples

```
params <- c(1.2, 0.75, 0.66, 2.78)
bgbb.PlotDropoutRateHeterogeneity(params)
params <- c(0.2, 1.5, 3.2, 6)
bgbb.PlotDropoutRateHeterogeneity(params)
```

bgbb.PlotFrequencyInCalibration

BG/BB Plot Frequency in Calibration Period

Description

Plots the actual and expected number of customers who made a certain number of repeat transactions in the calibration period. Also returns a matrix with this comparison.

Usage

```
bgbb.PlotFrequencyInCalibration(
  params,
  rf.matrix,
  censor = NULL,
  plotZero = TRUE,
  xlab = "Calibration period transactions",
  ylab = "Customers",
  title = "Frequency of Repeat Transactions"
)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`censor`	optional. Any calibration period frequency at this number, or above it, will be binned together. If the censor number is greater than the maximum recency in the recency-frequency matrix, the maximum recency will be used as the censor number.
`plotZero`	If FALSE, the histogram will exclude the zero bin.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`title`	title placed on the top-center of the plot.

Value

Calibration period repeat transaction frequency comparison matrix, actual vs. expected.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
```

```
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# plot actual vs. expected frequencies in the calibration period
bgbb.PlotFrequencyInCalibration(est.params, rf.matrix)
```

---

bgbb.PlotFrequencyInHoldout

BG/BB Plot Frequency in Holdout

Description

Plots the actual and expected number of customers who made a certain number of transactions in the holdout period, binned according to holdout period frequencies. Also returns a matrix with this comparison and the number of customers in each bin.

Usage

```
bgbb.PlotFrequencyInHoldout(
  params,
  n.cal,
  rf.matrix.holdout,
  censor = NULL,
  plotZero = TRUE,
  title = "Frequency of Repeat Transactions",
  xlab = "Holdout period transactions",
  ylab = "Customers"
)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`n.cal`	number of transaction opportunities in the calibration period.
`rf.matrix.holdout`	holdout period recency-frequency matrix. It must contain columns for frequency in the holdout period ("x.star"), the number of transaction opportunities in the holdout period ("n.star"), and the number of customers with each frequency ("custs").

`censor`	optional. Any calibration period frequency at this number, or above it, will be binned together. If the censor number is greater than the maximum recency in the recency-frequency matrix, the maximum recency will be used as the censor number.
`plotZero`	If FALSE, the histogram will exclude the zero bin.
`title`	title placed on the top-center of the plot.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.

Value

Holdout period repeat transaction frequency comparison matrix (actual vs. expected).

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
rf.matrix.holdout <- donationsSummary$rf.matrix.holdout
# donationsSummary$rf.matrix and donationsSummary$rf.matrix.holdout already
# have appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# number of periods in the calibration period
n.cal = max(rf.matrix[,"n.cal"])

bgbb.PlotFrequencyInHoldout(est.params, n.cal, rf.matrix.holdout)
```

`bgbb.PlotFreqVsConditionalExpectedFrequency`
BG/BB Plot Frequency vs Conditional Expected Frequency

Description

Plots the actual and conditional expected number of transactions made by customers in the holdout period, binned according to calibration period frequencies. Also returns a matrix with this comparison and the number of customers in each bin.

Usage

```
bgbb.PlotFreqVsConditionalExpectedFrequency(
  params,
  n.star,
  rf.matrix,
  x.star,
  trunc = NULL,
  xlab = "Calibration period transactions",
  ylab = "Holdout period transactions",
  xticklab = NULL,
  title = "Conditional Expectation"
)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`n.star`	number of transaction opportunities in the holdout period.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`x.star`	a vector containing the number of transactions made in the holdout period by the groups of customers with the same recency and frequency in the calibration period. It must be in the same order as the rf.matrix.
`trunc`	optional integer used to truncate the plot. In the plot, all calibration period frequencies above the truncation number will be removed. If the truncation number is greater than the maximum frequency, R will warn you and change it to the maximum frequency.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Value

Holdout period transaction frequency comparison matrix (actual vs. expected), binned by calibration period frequency.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# get the holdout period transactions
x.star <- donationsSummary$x.star

# number of transaction opportunities in the holdout period
n.star <- 5

# Plot holdout period transactions
bgbb.PlotFreqVsConditionalExpectedFrequency(est.params, n.star, rf.matrix, x.star, trunc=6)
```

---

bgbb.PlotRecVsConditionalExpectedFrequency
BG/BB Plot Recency vs Conditional Expected Frequency

Description

Plots the actual and conditional expected number of transactions made by customers in the holdout period, binned according to calibration period recencies. Also returns a matrix with this comparison and the number of customers in each bin.

Usage

```
bgbb.PlotRecVsConditionalExpectedFrequency(
  params,
  n.star,
  rf.matrix,
  x.star,
  trunc = NULL,
  xlab = "Calibration period recency",
  ylab = "Holdout period transactions",
  xticklab = NULL,
  title = "Conditional Expected Transactions by Recency"
)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`n.star`	number of transaction opportunities in the holdout period.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`x.star`	a vector containing the number of transactions made in the holdout period by the groups of customers with the same recency and frequency in the calibration period. It must be in the same order as the rf.matrix.
`trunc`	optional integer used to truncate the plot. In the plot, all calibration period frequencies above the truncation number will be removed. If the truncation number is greater than the maximum frequency, R will warn you and change it to the maximum frequency.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Value

Holdout period transaction frequency comparison matrix (actual vs. expected), binned by calibration period recency.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)
```

```
# get the holdout period transactions
x.star <- donationsSummary$x.star

# number of transaction opportunities in the holdout period
n.star <- 5

# Compare holdout period transactions.
bgbb.PlotRecVsConditionalExpectedFrequency(est.params, n.star, rf.matrix, x.star, trunc=6)
```

bgbb.PlotTrackingCum *BG/BB Tracking Cumulative Transactions Plot*

Description

Plots the actual and expected cumulative total repeat transactions by all customers for the calibration and holdout periods. Also returns a matrix with this comparison.

Usage

```
bgbb.PlotTrackingCum(
  params,
  rf.matrix,
  actual.cum.repeat.transactions,
  xlab = "Time",
  ylab = "Cumulative Transactions",
  xticklab = NULL,
  title = "Tracking Cumulative Transactions"
)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`actual.cum.repeat.transactions`	vector containing the cumulative number of repeat transactions made by customers in all transaction opportunities (both calibration and holdout periods). Its unit of time should be the same as the units of the recency-frequency matrix used to estimate the model parameters.

`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

The holdout period should immediately follow the calibration period. This function assumes that all customers' calibration periods end on the same date, rather than starting on the same date (thus customers' birth periods are determined using `max(n.cal) -n.cal` rather than assuming that they are all 0).

Value

Matrix containing actual and expected cumulative repeat transactions.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
data(donationsSummary)
# donationsSummary$rf.matrix already has appropriate column names
rf.matrix <- donationsSummary$rf.matrix

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# get the annual repeat transactions, and transform them into
# a cumulative form
actual.inc.repeat.transactions <- donationsSummary$annual.trans
actual.cum.repeat.transactions <- cumsum(actual.inc.repeat.transactions)

# set appropriate x-axis
x.tickmarks <- c( "'96","'97","'98","'99","'00","'01","'02","'03","'04","'05","'06" )

# plot actual vs. expected transactions. The calibration period was 6 periods long.
bgbb.PlotTrackingCum(est.params, rf.matrix, actual.cum.repeat.transactions, xticklab=x.tickmarks)
```

`bgbb.PlotTrackingInc` *BG/BB Tracking Incremental Transactions Plot*

Description

Plots the actual and expected incremental total repeat transactions by all customers for the calibration and holdout periods. Also returns a matrix of this comparison.

Usage

```
bgbb.PlotTrackingInc(
  params,
  rf.matrix,
  actual.inc.repeat.transactions,
  xlab = "Time",
  ylab = "Transactions",
  xticklab = NULL,
  title = "Tracking Incremental Transactions"
)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`actual.inc.repeat.transactions`	vector containing the incremental number of repeat transactions made by customers in all transaction opportunities (both calibration and holdout periods). Its unit of time should be the same as the units of the recency-frequency matrix used to estimate the model parameters.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

The holdout period should immediately follow the calibration period. This function assumes that all customers' calibration periods end on the same date, rather than starting on the same date (thus customers' birth periods are determined using `max(n.cal) -n.cal` rather than assuming that they are all 0).

Value

Matrix containing actual and expected incremental repeat transactions.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

Examples

```
data(donationsSummary)
# donationsSummary$rf.matrix already has appropriate column names
rf.matrix <- donationsSummary$rf.matrix

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# get the annual repeat transactions
actual.inc.repeat.transactions <- donationsSummary$annual.trans

# Set appropriate x-axis
x.tickmarks <- c( "'96","'97","'98","'99","'00","'01","'02","'03","'04","'05","'06" )

# Plot actual vs. expected transactions. The calibration period was 6 periods long.
bgbb.PlotTrackingInc(est.params, rf.matrix, actual.inc.repeat.transactions, xticklab=x.tickmarks)
```

`bgbb.PlotTransactionRateHeterogeneity`
BG/BB Plot Transaction Rate Heterogeneity

Description

Plots and returns the estimated beta distribution of P (customers' propensities to purchase).

Usage

```
bgbb.PlotTransactionRateHeterogeneity(params)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.

Details

This returns the distribution of each customer's Bernoulli parameter, which determines the level of their purchasing (using the BG/BB assumption that purchasing on the individual level can be modeled with a Bernoulli distribution).

Value

Distribution of customers' propensities to purchase.

Examples

```
params <- c(1.2, 0.75, 0.66, 2.78)
bgbb.PlotTransactionRateHeterogeneity(params)
params <- c(0.2, 1.5, 3.2, 6)
bgbb.PlotTransactionRateHeterogeneity(params)
```

`bgbb.pmf` *BG/BB Probability Mass Function*

Description

Probability mass function for the BG/BB.

Usage

```
bgbb.pmf(params, n, x)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`n`	number of transaction opportunities; may also be a vector.
`x`	number of transactions; may also be a vector.

Details

P(X(n)=x | alpha, beta, gamma, delta). Returns the probability that a customer makes x transactions in the first n transaction opportunities.

Parameters `n` and `x` may be vectors. The standard rules for vector operations apply - if they are not of the same length, the shorter vector will be recycled (start over at the first element) until it is as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Probability of X(n)=x, conditional on model parameters.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See Also

`bgbb.pmf.General`

Examples

```
params <- c(1.20, 0.75, 0.66, 2.78)
# The probability that a customer made 3 transactions in the first
# 6 transaction opportunities.
bgbb.pmf(params, n=6, x=3)

# Vectors may also be used as arguments:
bgbb.pmf(params, n=6, x=0:6)
```

`bgbb.pmf.General` *BG/BB General Probability Mass Function*

Description

Calculates the probability that a customer will make `x.star` transactions in the first `n.star` transaction opportunities following the calibration period.

Usage

```
bgbb.pmf.General(params, n.cal, n.star, x.star)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`n.cal`	number of transaction opportunities in the calibration period. Can also be a vector of calibration period transaction opportunities - see details.
`n.star`	number of transaction opportunities in the holdout period, or a vector of holdout period transaction opportunities.
`x.star`	number of transactions in the holdout period, or a vector of transaction frequencies.

Details

P(X(n, n + n*) = x* | alpha, beta, gamma, delta). This is a more generalized version of the bgbb.pmf. Setting `n.cal` to 0 reduces this function to the probability mass function in its usual format - the probability that a user will make x.star transactions in the first n.star transaction opportunities.

It is impossible for a customer to make a negative number of transactions, or to make more transactions than there are transaction opportunities. This function will throw an error if such inputs are provided.

`n.cal`, `n.star`, and `x.star` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Probability of X(n, n + n*) = x*, given BG/BB model parameters.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See Also

`bgbb.pmf`

Examples

```
params <- c(1.20, 0.75, 0.66, 2.78)
# Probability that a customer will make 3 transactions in the 10
# transaction opportunities following the 6 transaction opportunities
# in the calibration period, given BG/BB parameters.
bgbb.pmf.General(params, n.cal=6, n.star=10, x.star=3)

# Vectors may also be provided as input:
```

```
# Comparison between different frequencies:
bgbb.pmf.General(params, n.cal=6, n.star=10, x.star=1:10)
# Comparison between different holdout transaction opportunities:
bgbb.pmf.General(params, n.cal=6, n.star=5:15, x.star=3)
```

bgbb.PosteriorMeanDropoutRate

BG/BB Posterior Mean Dropout Rate

Description

Computes the mean value of the marginal posterior value of Theta, the geometric dropout process parameter.

Usage

```
bgbb.PosteriorMeanDropoutRate(params, x, t.x, n.cal)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`x`	the number of repeat transactions made by the customer in the calibration period. Can also be vector of frequencies - see details.
`t.x`	recency - the transaction opportunity in which the customer made their last transaction. Can also be a vector of recencies - see details.
`n.cal`	number of transaction opportunities in the calibration period. Can also be a vector of calibration period transaction opportunities - see details.

Details

E(Theta | alpha, beta, gamma, delta, x, t.x, n). This is calculated by setting `l = 0` and `m = 1` in `bgbb.PosteriorMeanLmProductMoment`.

`x`, `t.x`, and `n.cal` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be also be a vector.

Value

The posterior mean dropout rate.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See Also

`bgbb.rf.matrix.PosteriorMeanDropoutRate`

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# return the posterior mean dropout rate vector
bgbb.rf.matrix.PosteriorMeanDropoutRate(est.params, rf.matrix)
```

`bgbb.PosteriorMeanLmProductMoment`

BG/BB Posterior Mean (l,m)th Product Moment

Description

Computes the (l,m)th product moment of the joint posterior distribution of P (the Bernoulli transaction process parameter) and Theta (the geometric dropout process parameter).

Usage

```
bgbb.PosteriorMeanLmProductMoment(params, l, m, x, t.x, n.cal)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`l`	moment degree of P
`m`	moment degree of Theta
`x`	the number of repeat transactions made by the customer in the calibration period. Can also be vector of frequencies - see details.

`t.x`	recency - the transaction opportunity in which the customer made their last transaction. Can also be a vector of recencies - see details.
`n.cal`	number of transaction opportunities in the calibration period. Can also be a vector of calibration period transaction opportunities - see details.

Details

E((P)^l(Theta)^m | alpha, beta, gamma, delta, x, t.x, n)

`x`, `t.x`, and `n.cal` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be also be a vector.

Value

The expected posterior (l,m)th product moment.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See equation 17.

`bgbb.PosteriorMeanTransactionRate`
BG/BB Posterior Mean Transaction Rate

Description

Computes the mean value of the marginal posterior value of P, the Bernoulli transaction process parameter.

Usage

```
bgbb.PosteriorMeanTransactionRate(params, x, t.x, n.cal)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`x`	the number of repeat transactions made by the customer in the calibration period. Can also be vector of frequencies - see details.
`t.x`	recency - the transaction opportunity in which the customer made their last transaction. Can also be a vector of recencies - see details.

`n.cal`	number of transaction opportunities in the calibration period. Can also be a vector of calibration period transaction opportunities - see details.

Details

E(P | alpha, beta, gamma, delta, x, t.x, n). This is calculated by setting `l = 1` and `m = 0` in `bgbb.PosteriorMeanLmProductMom`

`x`, `t.x`, and `n.cal` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be also be a vector.

Value

The posterior mean transaction rate.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See Also

`bgbb.rf.matrix.PosteriorMeanTransactionRate`

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# return the posterior mean transaction rate vector
bgbb.rf.matrix.PosteriorMeanTransactionRate(est.params, rf.matrix)
```

`bgbb.rf.matrix.DERT` *BG/BB Discounted Expected Residual Transactions using a recency-frequency matrix*

Description

Computes the number of discounted expected residual transactions by a customer, conditional on their behavior in the calibration period.

Usage

```
bgbb.rf.matrix.DERT(params, rf.matrix, d)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`d`	discount rate.

Value

The present value of the expected future transaction stream for a particular customer.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web. See equation 14.

See Also

`bgbb.DERT`

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# compute DERT for a customer from every row in rf.matrix,
# discounted at 10%.
bgbb.rf.matrix.DERT(est.params, rf.matrix, d = 0.1)
```

`bgbb.rf.matrix.LL` *BG/BB Log-Likelihood using a recency-frequency matrix*

Description

Calculates the log-likelihood of the BG/BB model.

Usage

```
bgbb.rf.matrix.LL(params, rf.matrix)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.

Value

The total log-likelihood of the provided data in rf.matrix.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See Also

`bgbb.LL`

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

params <- c(1.20, 0.75, 0.66, 2.78)
bgbb.rf.matrix.LL(params, rf.matrix)
```

`bgbb.rf.matrix.PosteriorMeanDropoutRate` *BG/BB Posterior Mean Dropout Rate using a recency-frequency matrix*

Description

Computes the mean value of the marginal posterior value of Theta, the geometric dropout process parameter.

Usage

```
bgbb.rf.matrix.PosteriorMeanDropoutRate(params, rf.matrix)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.

Details

E(Theta | alpha, beta, gamma, delta, x, t.x, n). This is calculated by setting `l = 0` and `m = 1` in `bgbb.PosteriorMeanLmProductMoment`.

`rf.matrix` has columns x, t.x, and n.cal'.

Value

The posterior mean dropout rate.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See Also

`bgbb.PosteriorMeanDropoutRate`

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# return the posterior mean dropout rate vector
bgbb.rf.matrix.PosteriorMeanDropoutRate(est.params, rf.matrix)
```

`bgbb.rf.matrix.PosteriorMeanTransactionRate` *BG/BB Posterior Mean Transaction Rate using a recency-frequency matrix*

Description

Computes the mean value of the marginal posterior value of P, the Bernoulli transaction process parameter.

Usage

```
bgbb.rf.matrix.PosteriorMeanTransactionRate(params, rf.matrix)
```

Arguments

`params`	BG/BB parameters - a vector with alpha, beta, gamma, and delta, in that order. Alpha and beta are unobserved parameters for the beta-Bernoulli transaction process. Gamma and delta are unobserved parameters for the beta-geometric dropout process.
`rf.matrix`	recency-frequency matrix. It must contain columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and the number of customers with this combination of recency, frequency and transaction opportunities in the calibration period ("custs"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.

Details

`rf.matrix` has columns x, t.x, and n.cal‘.

Value

The posterior mean transaction rate.

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. Web.

See Also

`bgbb.PosteriorMeanTransactionRate`

Examples

```
data(donationsSummary)

rf.matrix <- donationsSummary$rf.matrix
# donationsSummary$rf.matrix already has appropriate column names

# starting-point parameters
startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
est.params <- bgbb.EstimateParameters(rf.matrix, startingparams)

# return the posterior mean transaction rate vector
bgbb.rf.matrix.PosteriorMeanTransactionRate(est.params, rf.matrix)
```

`bgnbd.cbs.LL` *BG/NBD Log-Likelihood Wrapper*

Description

Calculates the log-likelihood sum of the BG/NBD model.

Usage

```
bgnbd.cbs.LL(params, cal.cbs)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period. If your data is compressed (see `dc.compress.cbs`), a fourth column labeled "custs" (number of customers with a specific combination of recency, frequency and length of calibration period) is available.

Details

Note: do not use a compressed `cal.cbs` matrix. It makes quicker work for Pareto/NBD estimation as implemented in this package, but the opposite is true for BG/NBD. For proof, compare the definition of the `bgnbd.cbs.LL` to that of `pnbd.cbs.LL`.

Value

The total log-likelihood of the provided data.

See Also

`bgnbd.EstimateParameters`

`bgnbd.LL`

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# random assignment of parameters
params <- c(0.5, 6, 1.2, 3.3)
# returns the log-likelihood of the given parameters
bgnbd.cbs.LL(params, cal.cbs)

# compare the speed and results to the following:
cal.cbs.compressed <- dc.compress.cbs(cal.cbs)
bgnbd.cbs.LL (params, cal.cbs.compressed)

# Returns the log likelihood of the parameters for a customer who
# made 3 transactions in a calibration period that ended at t=6,
# with the last transaction occurring at t=4.
bgnbd.LL(params, x=3, t.x=4, T.cal=6)

# We can also give vectors as function parameters:
set.seed(7)
x <- sample(1:4, 10, replace = TRUE)
t.x <- sample(1:4, 10, replace = TRUE)
T.cal <- rep(4, 10)
bgnbd.LL(params, x, t.x, T.cal)
```

`bgnbd.ConditionalExpectedTransactions`

BG/NBD Conditional Expected Transactions

Description

E[X(T.cal, T.cal + T.star) | x, t.x, r, alpha, a, b]

Usage

```
bgnbd.ConditionalExpectedTransactions(
  params,
  T.star,
  x,
  t.x,
  T.cal,
  hardie = TRUE
)
```

Arguments

params	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
T.star	length of time for which we are calculating the expected number of transactions.
x	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
t.x	time of most recent repeat transaction, or a vector of recencies.
T.cal	length of calibration period, or a vector of calibration period lengths.
hardie	if TRUE, use `h2f1` instead of `hypergeo`.

Details

`T.star`, `x`, `t.x` and `T.cal` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Number of transactions a customer is expected to make in a time period of length t, conditional on their past behavior. If any of the input parameters has a length greater than 1, this will be a vector of expected number of transactions.

References

Fader, Peter S.; Hardie, Bruce G.S.and Lee, Ka Lok. “Computing P(alive) Using the BG/NBD Model.” December. 2008. Web. `http://www.brucehardie.com/notes/021/palive_for_BGNBD.pdf`

See Also

`bgnbd.Expectation`

Examples

```
params <- c(0.243, 4.414, 0.793, 2.426)
# Number of transactions a customer is expected to make in 2 time
# intervals, given that they made 10 repeat transactions in a time period
# of 39 intervals, with the 10th repeat transaction occurring in the 35th
# interval.
bgnbd.ConditionalExpectedTransactions(params, T.star=2, x=10, t.x=35, T.cal=39)

# We can also compare expected transactions across different
# calibration period behaviors:
bgnbd.ConditionalExpectedTransactions(params, T.star=2, x=5:20, t.x=25, T.cal=39)
```

`bgnbd.EstimateParameters`
BG/NBD Parameter Estimation

Description

Estimates parameters for the BG/NBD model.

Usage

```
bgnbd.EstimateParameters(
  cal.cbs,
  par.start = c(1, 3, 1, 3),
  max.param.value = 10000,
  method = "L-BFGS-B",
  hessian = FALSE
)
```

Arguments

`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`par.start`	initial BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`max.param.value`	the upper bound on parameters.
`method`	the optimization method(s) passed along to `optimx`.
`hessian`	set it to TRUE if you want the Hessian matrix, and then you might as well have the complete `optimx` object returned.

Details

The best-fitting parameters are determined using the bgnbd.cbs.LL function. The sum of the log-likelihood for each customer (for a set of parameters) is maximized in order to estimate parameters.

A set of starting parameters must be provided for this method. If no parameters are provided, (1,3,1,3) is used as a default. These values are used because they provide good convergence across data sets. It may be useful to use starting values for r and alpha that represent your best guess of the heterogeneity in the buy and die rate of customers. It may be necessary to run the estimation from multiple starting points to ensure that it converges. To compare the log-likelihoods of different parameters, use bgnbd.cbs.LL.

The lower bound on the parameters to be estimated is always zero, since BG/NBD parameters cannot be negative. The upper bound can be set with the max.param.value parameter.

This function may take some time to run.

Value

Vector of estimated parameters.

References

Fader, Peter S.; Hardie, and Bruce G.S.. "Overcoming the BG/NBD Model's #NUM! Error Problem." December. 2013. Web. http://brucehardie.com/notes/027/bgnbd_num_error.pdf

See Also

bgnbd.cbs.LL

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# starting-point parameters
startingparams <- c(1.0, 3, 1.0, 3)

# estimated parameters
est.params <- bgnbd.EstimateParameters(cal.cbs = cal.cbs,
                                       par.start = startingparams)

# complete object returned by \code{\link[optimx]{optimx}}
optimx.set <- bgnbd.EstimateParameters(cal.cbs = cal.cbs,
                                       par.start = startingparams,
                                       hessian = TRUE)

# log-likelihood of estimated parameters
bgnbd.cbs.LL(est.params, cal.cbs)
```

`bgnbd.Expectation` *BG/NBD Expectation*

Description

Returns the number of repeat transactions that a randomly chosen customer (for whom we have no prior information) is expected to make in a given time period.

Usage

```
bgnbd.Expectation(params, t, hardie = TRUE)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`t`	length of time for which we are calculating the expected number of repeat transactions.
`hardie`	if TRUE, use `h2f1` instead of `hypergeo`.

Details

E(X(t) | r, alpha, a, b)

Value

Number of repeat transactions a customer is expected to make in a time period of length t.

References

Fader, Peter S.; Hardie, Bruce G.S.and Lee, Ka Lok. “Computing P(alive) Using the BG/NBD Model.” December. 2008. Web. `http://www.brucehardie.com/notes/021/palive_for_BGNBD.pdf`

See Also

`bgnbd.ConditionalExpectedTransactions`

Examples

```
params <- c(0.243, 4.414, 0.793, 2.426)

# Number of repeat transactions a customer is expected to make in 2 time intervals.
bgnbd.Expectation(params, t=2, hardie = FALSE)

# We can also compare expected transactions over time:
bgnbd.Expectation(params, t=1:10)
```

bgnbd.ExpectedCumulativeTransactions

BG/NBD Expected Cumulative Transactions

Description

Calculates the expected cumulative total repeat transactions by all customers for the calibration and holdout periods.

Usage

```
bgnbd.ExpectedCumulativeTransactions(
  params,
  T.cal,
  T.tot,
  n.periods.final,
  hardie = TRUE
)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`T.cal`	a vector to represent customers' calibration period lengths (in other words, the "T.cal" column from a customer-by-sufficient-statistic matrix).
`T.tot`	end of holdout period. Must be a single value, not a vector.
`n.periods.final`	number of time periods in the calibration and holdout periods. See details.
`hardie`	if TRUE, use h2f1 instead of hypergeo.

Details

The function automatically divides the total period up into n.periods.final time intervals. n.periods.final does not have to be in the same unit of time as the T.cal data. For example: - if your T.cal data is in weeks, and you want cumulative transactions per week, n.periods.final would equal T.star. - if your T.cal data is in weeks, and you want cumulative transactions per day, n.periods.final would equal T.star * 7.

The holdout period should immediately follow the calibration period. This function assume that all customers' calibration periods end on the same date, rather than starting on the same date (thus customers' birth periods are determined using max(T.cal) - T.cal rather than assuming that it is 0).

Value

Vector of expected cumulative total repeat transactions by all customers.

See Also

bgnbd.Expectation

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

params <- c(0.243, 4.414, 0.793, 2.426)

# Returns a vector containing cumulative repeat transactions for 273 days.
# All parameters are in weeks; the calibration period lasted 39 weeks.
bgnbd.ExpectedCumulativeTransactions(params,
                                     T.cal = cal.cbs[,"T.cal"],
                                     T.tot = 39,
                                     n.periods.final = 273,
                                     hardie = TRUE)
```

---

bgnbd.generalParams *Define general parameters*

Description

This is to ensure consistency across all functions that require common bits and bobs.

Usage

```
bgnbd.generalParams(params, func, x, t.x, T.cal, T.star = NULL, hardie = NULL)
```

Arguments

params	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
func	function calling dc.InputCheck
x	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
t.x	time of most recent repeat transaction, or a vector of recencies.
T.cal	length of calibration period, or a vector of calibration period lengths.
T.star	length of time for which we are calculating the expected number of transactions.
hardie	if TRUE, use h2f1 instead of hypergeo when you call this function from within bgnbd.ConditionalExpectedTransactions.

Value

a list with things you need for `bgnbd.LL`, `bgnbd.PAlive` and `bgnbd.ConditionalExpectedTransactions`

See Also

`bgnbd.LL`

`bgnbd.PAlive`

`bgnbd.ConditionalExpectedTransactions`

`bgnbd.LL` *BG/NBD Log-Likelihood*

Description

Calculates the log-likelihood of the BG/NBD model.

Usage

```
bgnbd.LL(params, x, t.x, T.cal)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`x`	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
`t.x`	time of most recent repeat transaction, or a vector of recencies.
`T.cal`	length of calibration period, or a vector of calibration period lengths.

Details

`x`, `t.x` and `T.cal` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be also be a vector.

Value

A vector of log-likelihoods as long as the longest input vector (x, t.x, or T.cal).

See Also

`bgnbd.EstimateParameters`

`bgnbd.cbs.LL`

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# random assignment of parameters
params <- c(0.5, 6, 1.2, 3.3)
# returns the log-likelihood of the given parameters
bgnbd.cbs.LL (params, cal.cbs)

# compare the speed and results to the following:
cal.cbs.compressed <- dc.compress.cbs(cal.cbs)
bgnbd.cbs.LL(params, cal.cbs.compressed)

# Returns the log likelihood of the parameters for a customer who
# made 3 transactions in a calibration period that ended at t=6,
# with the last transaction occurring at t=4.
bgnbd.LL(params, x=3, t.x=4, T.cal=6)

# We can also give vectors as function parameters:
set.seed(7)
x <- sample(1:4, 10, replace = TRUE)
t.x <- sample(1:4, 10, replace = TRUE)
T.cal <- rep(4, 10)
bgnbd.LL(params, x, t.x, T.cal)
```

`bgnbd.PAlive` *BG/NBD P(Alive)*

Description

Uses BG/NBD model parameters and a customer's past transaction behavior to return the probability that they are still alive at the end of the calibration period.

Usage

```
bgnbd.PAlive(params, x, t.x, T.cal)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`x`	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
`t.x`	time of most recent repeat transaction, or a vector of recencies.
`T.cal`	length of calibration period, or a vector of calibration period lengths.

Details

P(Alive | X=x, t.x, T.cal, r, alpha, a, b)

x, t.x, and T.cal may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Probability that the customer is still alive at the end of the calibration period. If x, t.x, and/or T.cal has a length greater than one, then this will be a vector of probabilities (containing one element matching each element of the longest input vector).

References

Fader, Peter S.; Hardie, Bruce G.S.and Lee, Ka Lok. “Computing P(alive) Using the BG/NBD Model.” December. 2008. Web.

Examples

```
params <- c(0.243, 4.414, 0.793, 2.426)

bgnbd.PAlive(params, x=23, t.x=39, T.cal=39)
# P(Alive) of a customer who has the same recency and total
# time observed.

bgnbd.PAlive(params, x=5:20, t.x=30, T.cal=39)
# Note the "increasing frequency paradox".

# To visualize the distribution of P(Alive) across customers:

data(cdnowSummary)
cbs <- cdnowSummary$cbs
params <- bgnbd.EstimateParameters(cbs, par.start = c(0.243, 4.414, 0.793, 2.426))
p.alives <- bgnbd.PAlive(params, cbs[,"x"], cbs[,"t.x"], cbs[,"T.cal"])
plot(density(p.alives))
```

---

bgnbd.PlotDropoutRateHeterogeneity

BG/NBD Plot Dropout Probability Heterogeneity

Description

Plots and returns the estimated gamma distribution of p (customers’ probability of dropping out immediately after a transaction).

Usage

```
bgnbd.PlotDropoutRateHeterogeneity(params, lim = NULL)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`lim`	upper-bound of the x-axis. A number is chosen by the function if none is provided.

Value

Distribution of customers' probabilities of dropping out.

Examples

```
params <- c(0.243, 4.414, 0.793, 2.426)
bgnbd.PlotDropoutRateHeterogeneity(params)
params <- c(0.243, 4.414, 1.33, 2.426)
bgnbd.PlotDropoutRateHeterogeneity(params)
```

`bgnbd.PlotFrequencyInCalibration` *BG/NBD Plot Frequency in Calibration Period*

Description

Plots a histogram and returns a matrix comparing the actual and expected number of customers who made a certain number of repeat transactions in the calibration period, binned according to calibration period frequencies.

Usage

```
bgnbd.PlotFrequencyInCalibration(
  params,
  cal.cbs,
  censor,
  plotZero = TRUE,
  xlab = "Calibration period transactions",
  ylab = "Customers",
  title = "Frequency of Repeat Transactions"
)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x") and total time observed ("T.cal").
`censor`	integer used to censor the data. See details.
`plotZero`	If FALSE, the histogram will exclude the zero bin.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`title`	title placed on the top-center of the plot.

Details

This function requires a censor number, which cannot be higher than the highest frequency in the calibration period CBS. The output matrix will have (censor + 1) bins, starting at frequencies of 0 transactions and ending at a bin representing calibration period frequencies at or greater than the censor number. The plot may or may not include a bin for zero frequencies, depending on the plotZero parameter.

Value

Calibration period repeat transaction frequency comparison matrix (actual vs. expected).

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# parameters estimated using bgnbd.EstimateParameters
est.params <- c(0.243, 4.414, 0.793, 2.426)
# the maximum censor number that can be used
max(cal.cbs[,"x"])

bgnbd.PlotFrequencyInCalibration(est.params, cal.cbs, censor=7)
```

`bgnbd.PlotFreqVsConditionalExpectedFrequency`
BG/NBD Plot Frequency vs. Conditional Expected Frequency

Description

Plots the actual and conditional expected number transactions made by customers in the holdout period, binned according to calibration period frequencies. Also returns a matrix with this comparison and the number of customers in each bin.

Usage

```
bgnbd.PlotFreqVsConditionalExpectedFrequency(
  params,
  T.star,
  cal.cbs,
  x.star,
  censor,
  xlab = "Calibration period transactions",
  ylab = "Holdout period transactions",
  xticklab = NULL,
  title = "Conditional Expectation"
)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`T.star`	length of then holdout period.
`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`x.star`	vector of transactions made by each customer in the holdout period.
`censor`	integer used to censor the data. See details.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

This function requires a censor number, which cannot be higher than the highest frequency in the calibration period CBS. The output matrix will have (censor + 1) bins, starting at frequencies of 0 transactions and ending at a bin representing calibration period frequencies at or greater than the censor number.

Value

Holdout period transaction frequency comparison matrix (actual vs. expected).

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
```

```
# cal.cbs already has column names required by method

# number of transactions by each customer in the 39 weeks
# following the calibration period
x.star <- cal.cbs[,"x.star"]

# parameters estimated using bgnbd.EstimateParameters
est.params <- c(0.243, 4.414, 0.793, 2.426)
# the maximum censor number that can be used
max(cal.cbs[,"x"])

# plot conditional expected holdout period frequencies,
# binned according to calibration period frequencies
bgnbd.PlotFreqVsConditionalExpectedFrequency(est.params,
                                             T.star = 39,
                                             cal.cbs,
                                             x.star,
                                             censor = 7)
```

---

bgnbd.PlotRecVsConditionalExpectedFrequency

BG/NBD Plot Actual vs. Conditional Expected Frequency by Recency

---

Description

Plots the actual and conditional expected number of transactions made by customers in the holdout period, binned according to calibration period recencies. Also returns a matrix with this comparison and the number of customers in each bin.

Usage

```
bgnbd.PlotRecVsConditionalExpectedFrequency(
  params,
  cal.cbs,
  T.star,
  x.star,
  xlab = "Calibration period recency",
  ylab = "Holdout period transactions",
  xticklab = NULL,
  title = "Actual vs. Conditional Expected Transactions by Recency"
)
```

Arguments

params	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.

`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`T.star`	length of then holdout period.
`x.star`	vector of transactions made by each customer in the holdout period.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

This function does bin customers exactly according to recency; it bins customers according to integer units of the time period of cal.cbs. Therefore, if you are using weeks in your data, customers will be binned as follows: customers with recencies between the start of the calibration period (inclusive) and the end of week one (exclusive); customers with recencies between the end of week one (inclusive) and the end of week two (exclusive); etc.

The matrix and plot will contain the actual number of transactions made by each bin in the holdout period, as well as the expected number of transactions made by that bin in the holdout period, conditional on that bin's behavior during the calibration period.

Value

Matrix comparing actual and conditional expected transactions in the holdout period.

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# number of transactions by each customer in the 39 weeks following
# the calibration period
x.star <- cal.cbs[,"x.star"]

# parameters estimated using bgnbd.EstimateParameters
est.params <- c(0.243, 4.414, 0.793, 2.426)

# plot conditional expected holdout period transactions,
# binned according to calibration period recencies
bgnbd.PlotRecVsConditionalExpectedFrequency(est.params,
                                            cal.cbs,
                                            T.star = 39,
                                            x.star)
```

bgnbd.PlotTrackingCum *BG/NBD Tracking Cumulative Transactions Plot*

Description

Plots the actual and expected cumulative total repeat transactions by all customers for the calibration and holdout periods, and returns this comparison in a matrix.

Usage

```
bgnbd.PlotTrackingCum(
  params,
  T.cal,
  T.tot,
  actual.cu.tracking.data,
  n.periods.final = NA,
  hardie = TRUE,
  xlab = "Week",
  ylab = "Cumulative Transactions",
  xticklab = NULL,
  title = "Tracking Cumulative Transactions"
)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`T.cal`	a vector to represent customers' calibration period lengths (in other words, the "T.cal" column from a customer-by-sufficient-statistic matrix).
`T.tot`	end of holdout period. Must be a single value, not a vector.
`actual.cu.tracking.data`	vector containing the cumulative number of repeat transactions made by customers for each period in the total time period (both calibration and holdout periods). See details.
`n.periods.final`	number of time periods in the calibration and holdout periods. See details.
`hardie`	if TRUE, use h2f1 instead of hypergeo.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

actual.cu.tracking.data does not have to be in the same unit of time as the T.cal data. T.tot will automatically be divided into periods to match the length of actual.cu.tracking.data. See bgnbd.ExpectedCumulativeTransactions.

The holdout period should immediately follow the calibration period. This function assume that all customers' calibration periods end on the same date, rather than starting on the same date (thus customers' birth periods are determined using max(T.cal) - T.cal rather than assuming that it is 0).

Value

Matrix containing actual and expected cumulative repeat transactions.

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# Cumulative repeat transactions made by all customers across calibration
# and holdout periods
cu.tracking <- cdnowSummary$cu.tracking

# parameters estimated using bgnbd.EstimateParameters
est.params <- c(0.243, 4.414, 0.793, 2.426)

# All parameters are in weeks; the calibration period lasted 39
# weeks and the holdout period another 39.
bgnbd.PlotTrackingCum(est.params,
                      T.cal = cal.cbs[,"T.cal"],
                      T.tot = 78,
                      actual.cu.tracking.data = cu.tracking,
                      hardie = TRUE)
```

`bgnbd.PlotTrackingInc` *BG/NBD Tracking Incremental Transactions Comparison*

Description

Plots the actual and expected incremental total repeat transactions by all customers for the calibration and holdout periods, and returns this comparison in a matrix.

Usage

```
bgnbd.PlotTrackingInc(
  params,
  T.cal,
  T.tot,
  actual.inc.tracking.data,
  n.periods.final = NA,
  hardie = TRUE,
  xlab = "Week",
  ylab = "Transactions",
  xticklab = NULL,
  title = "Tracking Weekly Transactions"
)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`T.cal`	a vector to represent customers' calibration period lengths (in other words, the "T.cal" column from a customer-by-sufficient-statistic matrix).
`T.tot`	end of holdout period. Must be a single value, not a vector.
`actual.inc.tracking.data`	vector containing the incremental number of repeat transactions made by customers for each period in the total time period (both calibration and holdout periods). See details.
`n.periods.final`	number of time periods in the calibration and holdout periods. See details.
`hardie`	if TRUE, use h2f1 instead of hypergeo.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

actual.inc.tracking.data does not have to be in the same unit of time as the T.cal data. T.tot will automatically be divided into periods to match the length of actual.inc.tracking.data. See bgnbd.ExpectedCumulativeTransactions

The holdout period should immediately follow the calibration period. This function assume that all customers' calibration periods end on the same date, rather than starting on the same date (thus customers' birth periods are determined using max(T.cal) - T.cal rather than assuming that it is 0).

Value

Matrix containing actual and expected incremental repeat transactions.

Examples

```
data(cdnowSummary)
cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# Cumulative repeat transactions made by all customers across calibration
# and holdout periods
cu.tracking <- cdnowSummary$cu.tracking
# make the tracking data incremental
inc.tracking <- dc.CumulativeToIncremental(cu.tracking)

# parameters estimated using bgnbd.EstimateParameters
est.params <- c(0.243, 4.414, 0.793, 2.426)

# All parameters are in weeks; the calibration period lasted 39
# weeks and the holdout period another 39.
bgnbd.PlotTrackingInc(est.params, ,
                      T.cal = cal.cbs[,"T.cal"],
                      T.tot = 78,
                      actual.inc.tracking.data = inc.tracking,
                      hardie = TRUE)
```

---

bgnbd.PlotTransactionRateHeterogeneity

BG/NBD Plot Transaction Rate Heterogeneity

Description

Plots and returns the estimated gamma distribution of lambda (customers' propensities to purchase).

Usage

```
bgnbd.PlotTransactionRateHeterogeneity(params, lim = NULL)
```

Arguments

params	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
lim	upper-bound of the x-axis. A number is chosen by the function if none is provided.

Details

This returns the distribution of each customer's Poisson parameter, which determines the rate at which each customer buys.

Value

Distribution of customers' propensities to purchase.

Examples

```
params <- c(0.243, 4.414, 0.793, 2.426)
bgnbd.PlotTransactionRateHeterogeneity(params)
params <- c(0.53, 4.414, 0.793, 2.426)
bgnbd.PlotTransactionRateHeterogeneity(params)
```

`bgnbd.pmf` *BG/NBD Probability Mass Function*

Description

Probability mass function for the BG/NBD.

Usage

```
bgnbd.pmf(params, t, x)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`t`	length end of time period for which probability is being computed. May also be a vector.
`x`	number of repeat transactions by a random customer in the period defined by t. May also be a vector.

Details

P(X(t)=x | r, alpha, a, b). Returns the probability that a customer makes x repeat transactions in the time interval (0, t].

Parameters t and x may be vectors. The standard rules for vector operations apply - if they are not of the same length, the shorter vector will be recycled (start over at the first element) until it is as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Probability of X(t)=x conditional on model parameters. If t and/or x has a length greater than one, a vector of probabilities will be returned.

References

Fader, Peter S.; Hardie, Bruce G.S.and Lee, Ka Lok. “Computing P(alive) Using the BG/NBD Model.” December. 2008. Web.

Examples

```
params <- c(0.243, 4.414, 0.793, 2.426)
# probability that a customer will make 10 repeat transactions in the
# time interval (0,2]
bgnbd.pmf(params, t=2, x=10)
# probability that a customer will make no repeat transactions in the
# time interval (0,39]
bgnbd.pmf(params, t=39, x=0)

# Vectors may also be used as arguments:
bgnbd.pmf(params, t=30, x=11:20)
```

`bgnbd.pmf.General` *Generalized BG/NBD Probability Mass Function*

Description

Generalized probability mass function for the BG/NBD.

Usage

```
bgnbd.pmf.General(params, t.start, t.end, x)
```

Arguments

`params`	BG/NBD parameters - a vector with r, alpha, a, and b, in that order. r and alpha are unobserved parameters for the NBD transaction process. a and b are unobserved parameters for the Beta geometric dropout process.
`t.start`	start of time period for which probability is being calculated. It can also be a vector of values.
`t.end`	end of time period for which probability is being calculated. It can also be a vector of values.
`x`	number of repeat transactions by a random customer in the period defined by (t.start, t.end]. It can also be a vector of values.

Details

P(X(t.start, t.end)=x | r, alpha, a, b). Returns the probability that a customer makes x repeat transactions in the time interval (t.start, t.end].

It is impossible for a customer to make a negative number of repeat transactions. This function will return an error if it is given negative times or a negative number of repeat transactions. This function will also return an error if t.end is less than t.start.

t.start, t.end, and x may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Probability of x transaction occuring between t.start and t.end conditional on model parameters. If t.start, t.end, and/or x has a length greater than one, a vector of probabilities will be returned.

References

Fader, Peter S.; Hardie, Bruce G.S.and Lee, Ka Lok. "Computing P(alive) Using the BG/NBD Model." December. 2008. Web.

Examples

```
params <- c(0.243, 4.414, 0.793, 2.426)
# probability that a customer will make 10 repeat transactions in the
# time interval (1,2]
bgnbd.pmf.General(params, t.start=1, t.end=2, x=10)
# probability that a customer will make no repeat transactions in the
# time interval (39,78]
bgnbd.pmf.General(params, t.start=39, t.end=78, x=0)
```

---

cdnowElog *CDNOW event log data*

Description

Data representing the purchasing behavior of 2,357 CDNOW customers between January 1997 and June 1998, in event log format.

Format

A comma-delimited file representing an event log with 6,919 entries. It has 5 columns: The customer's ID in the master dataset, the customer's ID in this dataset (which represents 1/10th of the master dataset), the date of the transaction in the format "%Y%m%d" (e.g. 19970225), the number of CDs purchased, and the dollar value of the transaction.

Details

The customers in this data represent 1/10th of the cohort of customers who made their first transactions with CDNOW in the first quarter of 1997. CDNOW was an online retailer, selling music and related products on the web since 1994.

Source

Can be found online.

`cdnowSummary` *CDNOW repeat transaction data summary*

Description

Data representing the purchasing behavior of 2,357 CDNOW customers between January 1997 and June 1998, summarized as a customer-by-time matrix and a vector of cumulative weekly transactions.

Usage

```
data(cdnowSummary)
```

Format

A named list of four elements:

- `cbs` A customer-by-time matrix with four columns: frequency ("x"), recency ("t.x"), length of observation in the calibration period ("T.cal"), and number of transactions in the holdout period ("x.star"). Each row represents a customer.
- `cu.tracking` A vector containing cumulative transactions for every week in both the calibration and estimating periods (78 weeks total). This vector contains the sum of transactions across all customers.
- `est.params` A vector containing estimated values for the four Pareto/NBD parameters: r, alpha, s, and beta, in that order. This estimation was made using `pnbd.EstimateParameters`, and is included here to avoid having to run the relatively time-consuming parameter estimation function in examples.
- `m.x` A vector containing the average value of each customer's repeat transactions. Used in examples for spend functions.

Details

The customers in this data represent 1/10th of the cohort of customers who made their first transactions with CDNOW in the first quarter of 1997. CDNOW was an online retailer, selling music and related products on the web since 1994.

Source

The data was put together using data conversion functions included in this package. The original event log is included (see `cdnowElog`).

dc.BuildCBSFromCBTAndDates

Build CBS matrix from CBT matrix

Description

Given a customer-by-time matrix, yields the resulting customer-by-sufficient-statistic matrix.

Usage

```
dc.BuildCBSFromCBTAndDates(cbt, dates, per, cbt.is.during.cal.period = TRUE)
```

Arguments

`cbt`	customer-by-time matrix. This is a matrix consisting of a row per customer and a column per time period. It should contain numeric information about a customer's transactions in every time period - either the number of transactions in that time period (frequency), a 1 to indicate that at least 1 transaction occurred (reach), or the average/total amount spent in that time period.
`dates`	if cbt.is.during.cal.period is TRUE, then dates is a data frame with three columns: 1. the dates when customers made their first purchases 2. the dates when customers made their last purchases 3. the date of the end of the calibration period. if cbt.is.during.cal.period is FALSE, then dates is a vector with two elements: 1. the date of the beginning of the holdout period 2. the date of the end of the holdout period.
`per`	interval of time for customer-by-sufficient-statistic matrix. May be "day", "week", "month", "quarter", or "year".
`cbt.is.during.cal.period`	if TRUE, indicates the customer-by-time matrix is from the calibration period. If FALSE, indicates the customer-by-time matrix is from the holdout period.

Details

The customer-by-sufficient statistic matrix will contain the sum of the statistic included in the customer-by-time matrix (see the cbt parameter), the customer's last transaction date, and the total time period for which the customer was observed.

Value

Customer-by-sufficient-statistic matrix, with three columns: frequency("x"), recency("t.x") and total time observed("T.cal"). See details. Frequency is total transactions, not repeat transactions.

Examples

```
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5)
elog[,"date"] <- as.Date(elog[,"date"], "%Y%m%d")

# Transaction-flow models are about interpurchase times. Since we
# only know purchase times to the day, we merge all transaction on
# the same day. This example uses dc.MergeTransactionsOnSameDate
# to illustrate this; however, we could have simply used dc.CreateReachCBT
# instead of dc.CreateFreqCBT to obtain the same result.
merged.elog <- dc.MergeTransactionsOnSameDate(elog)
cutoff.date <- as.Date("1997-09-30")
freq.cbt <- dc.CreateFreqCBT(merged.elog)
cal.freq.cbt <- freq.cbt[,as.Date(colnames(freq.cbt)) <= cutoff.date]
holdout.freq.cbt <- freq.cbt[,as.Date(colnames(freq.cbt)) > cutoff.date]

cal.start.dates.indices <- dc.GetFirstPurchasePeriodsFromCBT(cal.freq.cbt)
cal.start.dates <- as.Date(colnames(cal.freq.cbt)[cal.start.dates.indices])
cal.end.dates.indices <- dc.GetLastPurchasePeriodsFromCBT(cal.freq.cbt)
cal.end.dates <- as.Date(colnames(cal.freq.cbt)[cal.end.dates.indices])
T.cal.total <- rep(cutoff.date, nrow(cal.freq.cbt))
cal.dates <- data.frame(cal.start.dates,
                        cal.end.dates,
                        T.cal.total)

# Create calibration period customer-by-sufficient-statistic data frame,
# using weeks as the unit of time.
cal.cbs <- dc.BuildCBSFromCBTAndDates(cal.freq.cbt,
                                      cal.dates,
                                      per="week",
                                      cbt.is.during.cal.period=TRUE)
# Force the calibration period customer-by-sufficient-statistic to only contain
# repeat transactions (required by BG/BB and Pareto/NBD models)
cal.cbs[,"x"] <- cal.cbs[,"x"] - 1

holdout.start <- cutoff.date+1
holdout.end <- as.Date(colnames(holdout.freq.cbt)[ncol(holdout.freq.cbt)])
holdout.dates <- c(holdout.start, holdout.end)

# Create holdout period customer-by-sufficient-statistic data frame, using weeks
# as the unit of time.
holdout.cbs <- dc.BuildCBSFromCBTAndDates(holdout.freq.cbt,
                                          holdout.dates,
                                          per="week",
                                          cbt.is.during.cal.period=FALSE)
```

`dc.BuildCBTFromElog` *Build Customer-by-Time Matrix from Event Log*

Description

Creates a customer-by-time matrix from an event log.

Usage

```
dc.BuildCBTFromElog(elog, statistic = "freq")
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction.. For the total spend and average spend matrices, the event log must have a "sales" column. If the dates are not formatted to be in the order year-month-day, the columns of the customer-by-time matrix may not be ordered chronologically if the "date" column does not consist of date objects (R will order them alphabetically). This will cause problems with other functions, so it is better to convert the date column to date objects before running this function.
`statistic`	either "freq", "reach", "total.spend", or "average.spend". This determines what type of customer-by-time matrix is returned.

Value

Customer-by-time matrix.

`dc.check.model.params` *Check model params*

Description

Check model parameters for correctness.

Usage

```
dc.check.model.params(printnames, params, func)
```

Arguments

`printnames`	Names to print parameter errors.
`params`	Model parameters.
`func`	Function calling dc.check.model.params.

Value

Stops the program if there is something wrong with the parameters.

`dc.compress.cbs` *Compress Customer-by-Sufficient-Statistic (CBS) Matrix*

Description

Combines all customers with the same combination of recency, frequency and length of calibration period in the customer-by-sufficient-statistic matrix, and adds a fourth column labelled "custs" (with the number of customers belonging in each row).

Usage

```
dc.compress.cbs(cbs, rounding = 3)
```

Arguments

`cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`rounding`	the function tries to ensure that there are similar customers by rounding the customer-by-sufficient-statistic matrix first. This parameter determines how many decimal places are left in the data. Negative numbers are allowed; see the documentation for round in the base package. As of the time of writing, that documentation states: "Rounding to a negative number of digits means rounding to a power of ten, so for example round(x, digits = -2) rounds to the nearest hundred."

Details

This function is meant to be used to speed up log-likelihood and parameter estimation functions in the Pareto/NBD (pnbd) set of functions. How much faster those function run depends on how similar customers are. You can used compressed CBS matrices in BG/NBD estimation too, but there will be no speed gains there over using un-compressed CBS data.

This function only takes columns "x", "t.x", and "T.cal" into account. All other columns will be added together - for example, if you have a spend column, the output's spend column will contain the total amount spent by all customers with an identical recency, frequency, and time observed.

Value

A customer-by-sufficient-statistic matrix with an additional column "custs", which contains the number of customers with each combination of recency, frequency and length of calibration period.

Examples

```
# Create a sample customer-by-sufficient-statistic matrix:
set.seed(7)
x <- sample(1:4, 10, replace = TRUE)
t.x <- sample(1:4, 10, replace = TRUE)
T.cal <- rep(4, 10)
ave.spend <- sample(10:20, 10, replace = TRUE)
cbs <- cbind(x, t.x, T.cal, ave.spend)
cbs

# If cbs is printed, you would note that the following
# sets of rows have the same x, t.x and T.cal:
# (1, 6, 8); (3, 9)

dc.compress.cbs(cbs, 0)   # No rounding necessary

# Note that all additional columns (in this case, ave.spend)
# are aggregated by sum.
```

`dc.CreateFreqCBT` *Create Frequency Customer-by-Time Matrix*

Description

Creates a customer-by-time matrix with total number of transactions per time period.

Usage

```
dc.CreateFreqCBT(elog)
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction. If the dates are not formatted to be in the order year-month-day, the columns of the customer-by-time matrix may not be ordered chronologically if the "date" column does not consist of date objects (R will order them alphabetically). This will cause problems with other functions, so it is better to convert the date column to date objects before running this function.

Value

Frequency customer-by-time matrix.

Examples

```
# Create event log from file "cdnowElog.csv", which has
# customer IDs in the second column, dates in the third column, and
# sales numbers in the fifth column.
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5)

# Given that the dates are in the order year-month-day,
# it is not strictly necessary to convert the date column
# to date formats. However, it is good practice:
elog[,"date"] <- as.Date(elog[,"date"], "%Y%m%d")

freq.cbt <- dc.CreateFreqCBT(elog)
```

`dc.CreateReachCBT` *Create Reach Customer-by-Time Matrix*

Description

Creates a customer-by-time matrix with 1's in periods that a customer made a transaction and 0's otherwise.

Usage

```
dc.CreateReachCBT(elog)
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction. If the dates are not formatted to be in the order year-month-day, the columns of the customer-by-time matrix may not be ordered chronologically if the "date" column does not consist of date objects (R will order them alphabetically). This will cause problems with other functions, so it is better to convert the date column to date objects before running this function.

Value

Reach customer-by-time matrix.

Examples

```
# Create event log from file "cdnowElog.csv", which has
# customer IDs in the second column, dates in the third column, and
# sales numbers in the fifth column.
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5)

# Given that the dates are in the order year-month-day,
# it is not strictly necessary to convert the date column
# to date formats. However, it is good practice:
```

```
elog[,"date"] <- as.Date(elog[,"date"], "%Y%m%d")

reach.cbt <- dc.CreateReachCBT(elog)
```

dc.CreateSpendCBT *Create Spend Customer-by-Time Matrix*

Description

Creates a customer-by-time matrix with spend per time period.

Usage

```
dc.CreateSpendCBT(elog, is.avg.spend = FALSE)
```

Arguments

elog	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction. If the dates are not formatted to be in the order year-month-day, the columns of the customer-by-time matrix may not be ordered chronologically if the "date" column does not consist of date objects (R will order them alphabetically). This will cause problems with other functions, so it is better to convert the date column to date objects before running this function.
is.avg.spend	if TRUE, return average spend customer-by-time matrix; else, return total spend customer-by-time matrix.

Value

Spend customer-by-time matrix.

Examples

```
# Create event log from file "cdnowElog.csv", which has
# customer IDs in the second column, dates in the third column, and
# sales numbers in the fifth column.
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5);

# Given that the dates are in the order year-month-day,
# it is not strictly necessary to convert the date column
# to date formats. However, it is good practice:
elog[,"date"] <- as.Date(elog[,"date"], "%Y%m%d")

spend.cbt <- dc.CreateSpendCBT(elog)
```

`dc.CumulativeToIncremental` *Cumulative to Incremental*

Description

Converts a vector of cumulative transactions to a vector of incremental transactions.

Usage

```
dc.CumulativeToIncremental(cu)
```

Arguments

`cu`	A vector containing cumulative transactions over time.

Value

Vector of incremental transactions.

`dc.DissipateElog` *Dissipate Event Log*

Description

Filters an event log, keeping a fraction of the original event log.

Usage

```
dc.DissipateElog(elog, dissipate.factor)
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction.
`dissipate.factor`	integer indicating how much of the dataset to eliminate. It must be greater than 1 for the function to work. (dissipate.factor-1)/(dissipate.factor) events will be removed from the event log. For example, if 2 is provided, 1/2 of the event log is eliminated, and if 10 is provided, 9/10 of the event log is eliminated.

Value

Reduced event log.

dc.ElogToCbsCbt *Convert Event Log to CBS and CBT Matrices*

Description

Uses an event log to return calibration period CBT and CBS, holdout period CBT and CBS, and summary data for each customer (including times of first and last transactions).

Usage

```
dc.ElogToCbsCbt(
  elog,
  per = "week",
  T.cal = max(elog$date),
  T.tot = max(elog$date),
  merge.same.date = TRUE,
  cohort.birth.per = T.cal,
  dissipate.factor = 1,
  statistic = "freq"
)
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction. The "date" column must contain date objects, not character strings or factors.
`per`	interval of time for customer-by-sufficient-statistic matrix. May be "day", "week", "month", "quarter", or "year".
`T.cal`	R date object indicating when the calibration period ends.
`T.tot`	T.tot R date object indicating when holdout period ends.
`merge.same.date`	If TRUE, transactions from the same period count as a single transaction instead of counting as multiple transactions.
`cohort.birth.per`	Time interval used to filter the event log. Can be specified as a Date object or a vector of two Dates. If one date object is used, the birth period is from the minimum date in the dataset through the given date. If two dates are given, the birth period is set between (inclusive) the two dates.
`dissipate.factor`	integer indicating how much of the dataset to eliminate. If left as 1, none of the dataset is eliminated. (dissipate.factor-1)/(dissipate.factor) events will be removed from the event log. For example, if 2 is provided, 1/2 of the event log is eliminated, and if 10 is provided, 9/10 of the event log is eliminated.
`statistic`	Determines type of CBT returned: can be: "reach", "freq", "total.spend", or "average.spend." (note: spend requires $sales column in elog)

Details

This function automatically removes customers' first transactions, meaning that the output matrices will only contain repeat transaction information.

Value

A list of items: - $cal list with CBS and CBT from the calibration period - $holdout list with CBS and CBT from holdout period - $cust.data data frame with each customer's first and last transaction details

Examples

```
# Create event log from file "cdnowElog.csv", which has
# customer IDs in the second column, dates in the third column, and
# sales numbers in the fifth column.
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5)

elog[,"date"] <- as.Date(elog[,"date"], "%Y%m%d")

data <- dc.ElogToCbsCbt(elog, per="week", T.cal=as.Date("1997-09-30"))
```

`dc.FilterCustByBirth` *Filter Customer by Birth*

Description

Filters an event log, keeping all transactions made by customers who made their first transactions in the given time interval.

Usage

```
dc.FilterCustByBirth(elog, cohort.birth.per)
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction. The date column must be formatted as Date objects.
`cohort.birth.per`	Time interval used to filter the event log. Can be specified as a Date object or a vector of two Dates. If one date object is used, the birth period is from the minimum date in the dataset through the given date. If two dates are given, the birth period is set between (inclusive) the two dates.

Value

event log with only rows from customers who made their first transaction within the birth period.

Examples

```
# Create event log from file "cdnowElog.csv", which has
# customer IDs in the second column, dates in the third column, and
# sales numbers in the fifth column.
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5)

# converting the date column to Date objects is
# necessary for this function.
elog$date <- as.Date(elog$date, "%Y%m%d")

# starting date. Note that it must be a Date object.
start.date <- as.Date("1997-01-01")
# ending date. Note that it must be a Date object.
end.date <- as.Date("1997-01-31")

# Filter the elog to include only customers who made their
# first transaction in January 1997
filtered.elog <- dc.FilterCustByBirth(elog, c(start.date, end.date))
```

---

dc.GetFirstPurchasePeriodsFromCBT

Get First Purchase Periods from Customer-by-Time Matrix

Description

Uses a customer-by-time matrix to return a vector containing the periods in which customers made their first purchase.

Usage

```
dc.GetFirstPurchasePeriodsFromCBT(cbt)
```

Arguments

cbt	customer-by-time matrix. This is a matrix consisting of a row per customer and a column per time period. It should contain numeric information about a customer's transactions in every time period - either the number of transactions in that time period (frequency), a 1 to indicate that at least 1 transaction occurred (reach), or the average/total amount spent in that time period.

Value

a vector containing the indices of periods in which customers made their first transactions. To convert to actual dates (if your customer-by-time matrix has dates as column names), use colnames(cbt)[RESULT]

`dc.GetLastPurchasePeriodsFromCBT` *Get Last Purchase Periods from Customer-by-Time Matrix*

Description

Uses a customer-by-time matrix to return a vector containing the periods in which customers made their last purchase.

Usage

```
dc.GetLastPurchasePeriodsFromCBT(cbt)
```

Arguments

`cbt`	customer-by-time matrix. This is a matrix consisting of a row per customer and a column per time period. It should contain numeric information about a customer's transactions in every time period - either the number of transactions in that time period (frequency), a 1 to indicate that at least 1 transaction occurred (reach), or the average/total amount spent in that time period.

Value

a vector containing the indices of periods in which customers made their last transactions. To convert to actual dates (if your customer-by-time matrix has dates as column names), use colnames(cbt)[RESULT]

`dc.InputCheck` *Check the inputs to functions that use this common pattern*

Description

A bunch of functions whose names start with `pnbd` take a set of four parameters as their first argument, and then a set of vectors or scalars such as `x` or `T.cal` as their subsequent arguments. This function started out as pnbd.InputCheck() and it was meant to run input checks for any number of such subsequent vector arguments, as long as they all met the same requirements as `x`, `t.x` and `T.cal` in `pnbd.LL`: meaning, the length of the longest of these vectors is a multiple of the lengths of all others, and all vectors are numeric and positive.

Usage

```
dc.InputCheck(params, func, printnames = c("r", "alpha", "s", "beta"), ...)
```

Arguments

`params`	If used by `pnbd.[...]` functions, Pareto/NBD parameters – a vector with r, alpha, s, and beta, in that order. See `pnbd.LL`. If used by `bgnbd.[...]` functions, BG/NBD parameters – a vector with r, alpha, a, and b, in that order. See `bgnbd.LL`. If used by `bgbb.[...]` functions, BG/BB parameters – a vector with alpha, beta, gamma, and delta, in that order. See `bgbb.LL`. If used by `spend.[...]` functions, a vector of gamma-gamma parameters – p, q, and gamma, in that order. See `spend.LL`.
`func`	Function calling dc.InputCheck
`printnames`	a string vector with the names of parameters to pass to `dc.check.model.params`
`...`	other arguments

Details

With an extra argument, `printnames`, pnbd.InputCheck() could also accommodate input checks for functions whose names start with bgbb, bgnbd, and spend so it was basically useful everywhere. That's when it became `dc.InputCheck()`. `params` can have any length as long as that length is the same as the length of `printnames`, so `dc.InputCheck()` can probably handle mixtures of distributions for modeling BTYD behavior that are not yet implemented.

By other arguments ... here we mean a bunch of named vectors that are used by functions that call `dc.InputCheck`, such as x, t.x, T.cal, etc. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. Vector recycling is a good way to get into trouble. Keep vectors to the same length and use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

If all is well, a data frame with everything you need in it, with nrow() equal to the length of the longest vector in `...`

See Also

`pnbd.LL` `pnbd.ConditionalExpectedTransactions`

`dc.MakeRFmatrixCal` *Make Calibration Period Recency-Frequency Matrix*

Description

Make a calibration period recency-frequency matrix.

Usage

```
dc.MakeRFmatrixCal(
  frequencies,
  periods.of.final.purchases,
  num.of.purchase.periods,
  holdout.frequencies = NULL
)
```

Arguments

`frequencies`	vector which indicates the number of repeat transactions made by customers in the calibration period.
`periods.of.final.purchases`	a vector indicating in which period customers made their final purchases.
`num.of.purchase.periods`	the number of transaction opportunities in the calibration period.
`holdout.frequencies`	an optional vector indicating the number of transactions made by customers in the holdout period.

Value

A matrix with all possible frequency-recency combinations, and the number of customers with each combination. It contains columns for frequency ("x"), recency ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), number of customers with this combination of recency, frequency, and number of periods observed ("custs"), and optionally, number of transactions in the holdout period ("x.star").

Examples

```
elog <- dc.ReadLines(system.file("data/discreteSimElog.csv", package="BTYD"),1,2)
elog[,"date"] <- as.Date(elog[,"date"])

cutoff.date <- as.Date("1977-01-01")
cbt <- dc.CreateReachCBT(elog)
cal.cbt <- cbt[,as.Date(colnames(cbt)) <= cutoff.date]
holdout.cbt <- cbt[,as.Date(colnames(cbt)) > cutoff.date]

cal.start.dates.indices <- dc.GetFirstPurchasePeriodsFromCBT(cal.cbt)
cal.start.dates <- as.Date(colnames(cal.cbt)[cal.start.dates.indices])
cal.end.dates.indices <- dc.GetLastPurchasePeriodsFromCBT(cal.cbt)
cal.end.dates <- as.Date(colnames(cal.cbt)[cal.end.dates.indices])
T.cal.total <- rep(cutoff.date, nrow(cal.cbt))
cal.dates <- data.frame(cal.start.dates, cal.end.dates, T.cal.total)

# Create calibration period customer-by-sufficient-statistic data frame,
# using years as the unit of time.
cal.cbs <- dc.BuildCBSFromCBTAndDates(cal.cbt,
                                      cal.dates,
                                      per="year",
                                    cbt.is.during.cal.period=TRUE)

holdout.start <- as.Date(colnames(holdout.cbt)[1])
holdout.end <- as.Date(tail(colnames(holdout.cbt),n=1))
# The (-1) below is to remove the effect of the birth period - we are only
# interested in repeat transactions in the calibration period.
frequencies <- (cal.cbs[,"x"] - 1)
periods.of.final.purchases <- cal.cbs[,"t.x"]
num.of.purchase.periods <- ncol(cal.cbt) - 1

# Create a calibration period recency-frequency matrix
cal.rf.matrix <- dc.MakeRFmatrixCal(frequencies,
                                    periods.of.final.purchases,
                                    num.of.purchase.periods)
```

---

dc.MakeRFmatrixHoldout

Make Holdout Period Recency-Frequency Matrix

Description

Creates a recency-frequency matrix for the holdout period.

Usage

```
dc.MakeRFmatrixHoldout(holdout.cbt)
```

Arguments

`holdout.cbt`	holdout period frequency customer-by-time matrix. This is a matrix consisting of a row per customer and a column per time period. It should contain the number of transactions each customer made per time period.

Value

recency-frequency matrix for the holdout period, with three columns: frequency ("x.star"), recency ("t.x.star"), number of transaction opportunities in the holdout period ("n.star"), and the number of customers with each frequency-recency combination ("custs").

`dc.MakeRFmatrixSkeleton` *Make Recency-Frequency Matrix Skeleton*

Description

Creates a matrix with all possible recency and frequency combinations.

Usage

```
dc.MakeRFmatrixSkeleton(n.periods)
```

Arguments

`n.periods`	number of transaction opportunities in the calibration period.

Details

Makes the structure in which to input data for recency-frequency matrices.

Value

Matrix with two columns: frequency ("x") and recency ("t.x"). All possible recency-frequency combinations in the calibration period are represented.

`dc.MergeCustomers` *Merge Customers*

Description

Takes two CBT or CBS matrices and ensures that the second one has the same row names as the first.

Usage

```
dc.MergeCustomers(data.correct, data.to.correct)
```

Arguments

`data.correct`	CBT or CBS with the correct customer IDs as row names. Usually from the calibration period.
`data.to.correct`	CBT or CBS which needs to be fixed (customer IDs inserted). Usually from the holdout period.

Details

Care should be taken in using this function. It inserts zero values in all rows that were not in the original holdout period data. This behavior does not cause a problem if using CBT matrices, but will cause a problem if using CBS matrices (for example, the output will report all customers with a holdout period length of zero). However, this particular issue is easily fixed (see examples) and should not cause problems.

A work-around to avoid using this function is presented in the example for `dc.BuildCBSFromCBTAndDates` - build the full CBT and only use the columns applying to each particular time period to construct separate CBTs, and from them, CBSs. That is a much cleaner and less error-prone method; however, on occasion the data will not be available in event log format and you may not be able to construct a CBT for both time periods together.

Value

Updated holdout period CBT or CBS.

Examples

```
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5)
elog[,"date"] <- as.Date(elog[,"date"], "%Y%m%d")
cutoff.date <- as.Date("1997-09-30")
cal.elog <- elog[which(elog[,"date"] <= cutoff.date),]
holdout.elog <- elog[which(elog[,"date"] > cutoff.date),]

# Create calibration period CBT from cal.elog
cal.reach.cbt <- dc.CreateReachCBT(cal.elog)
# Create holdout period CBT from holdout.elog
holdout.reach.cbt <- dc.CreateReachCBT(holdout.elog)

# Note the difference:
nrow(cal.reach.cbt)            # 2357 customers
nrow(holdout.reach.cbt)        # 684 customers

# Create a "fixed" holdout period CBT, with the same number
# of customers in the same order as the calibration period CBT
fixed.holdout.reach.cbt <- dc.MergeCustomers(cal.reach.cbt, holdout.reach.cbt)
nrow(fixed.holdout.reach.cbt)  # 2357 customers

# You can verify that the above is correct by turning these into a CBS
# (see \code{\link{dc.BuildCBSFromCBTAndDates}} and using
# \code{\link{pnbd.PlotFreqVsConditionalExpectedFrequency}}, for example

# Alternatively, we can fix the CBS, instead of the CBS:

cal.start.dates.indices <- dc.GetFirstPurchasePeriodsFromCBT(cal.reach.cbt)
cal.start.dates <- as.Date(colnames(cal.reach.cbt)[cal.start.dates.indices])
cal.end.dates.indices <- dc.GetLastPurchasePeriodsFromCBT(cal.reach.cbt)
cal.end.dates <- as.Date(colnames(cal.reach.cbt)[cal.end.dates.indices])
T.cal.total <- rep(cutoff.date, nrow(cal.reach.cbt))
cal.dates <- data.frame(cal.start.dates, cal.end.dates, T.cal.total)
```

```
# Create calibration period customer-by-sufficient-statistic data frame,
# using weeks as the unit of time.
cal.cbs <- dc.BuildCBSFromCBTAndDates(cal.reach.cbt,
                                      cal.dates,
                                      per="week",
                                      cbt.is.during.cal.period=TRUE)

# Force the calibration period customer-by-sufficient-statistic to only
#  contain repeat transactions (required by BG/BB and Pareto/NBD models)
cal.cbs[,"x"] <- cal.cbs[,"x"] - 1

holdout.start <- cutoff.date+1
holdout.end <- as.Date(colnames(fixed.holdout.reach.cbt)[ncol(fixed.holdout.reach.cbt)])
holdout.dates <- c(holdout.start, holdout.end)

# Create holdout period customer-by-sufficient-statistic data frame,
# using weeks as the unit of time.
holdout.cbs <- dc.BuildCBSFromCBTAndDates(holdout.reach.cbt,
                                          holdout.dates,
                                          per="week",
                                          cbt.is.during.cal.period=FALSE)

# Note the difference:
nrow(cal.cbs)            # 2357 customers
nrow(holdout.cbs)        # 684 customers

# Create a "fixed" holdout period CBS, with the same number
# of customers in the same order as the calibration period CBS
fixed.holdout.cbs <- dc.MergeCustomers(cal.cbs, holdout.cbs)
nrow(fixed.holdout.cbs)  # 2357 customers

# Furthermore, this function will assign a zero value to all fields
# that were not in the original holdout period CBS. Since T.star is the
# same for all customers in the holdout period, we should fix that:
fixed.holdout.cbs[,"T.star"] <- rep(max(fixed.holdout.cbs[,"T.star"]),nrow(fixed.holdout.cbs))
```

---

dc.MergeTransactionsOnSameDate

Merge Transactions on Same Day

Description

Updates an event log; any transactions made by the same customer on the same day are combined into one transaction.

Usage

```
dc.MergeTransactionsOnSameDate(elog)
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction.

Value

Event log with transactions made by the same customer on the same day merged into one transaction.

`dc.PlotLogLikelihoodContour` *Plot Log-Likelihood Contour*

Description

Makes a contour plot of a loglikelihood function that varies over two designated parameters, centered around a set of previously estimated parameters.

Usage

```
dc.PlotLogLikelihoodContour(
  loglikelihood.fcn,
  vary.or.fix.param,
  predicted.params,
  ...,
  n.divs = 3,
  new.dev = FALSE,
  num.contour.lines = 10,
  zoom.percent = 0.9,
  allow.neg.params = FALSE,
  param.names = c("param 1", "param 2", "param 3", "param 4")
)
```

Arguments

`loglikelihood.fcn`	log-likelihood function to plot.
`vary.or.fix.param`	a vector of strings containing either "vary" or "fix". The parameters in the same indices as "vary" will be plotted while the other parameters will remain fixed at the estimated values. See details.
`predicted.params`	estimated parameters.
`...`	all additional arguments required by the log-likelihood function. For example, `bgbb.rf.matrix.LL` requires rf.matrix; `pnbd.cbs.LL` requires cal.cbs and hardie (defaults to TRUE); and `bgnbd.cbs.LL` requires cal.cbs.

`n.divs`	integer representing how fine-grained the contour plot is. A higher value will produce a higher resolution plot with smoother contour lines, but will take longer to plot. n.divs also affects the boundaries of the contour plot; see details.
`new.dev`	if TRUE, makes a new window for each contour plot.
`num.contour.lines`	number of contour lines to plot in the window.
`zoom.percent`	determines boundaries of contour plot. See details.
`allow.neg.params`	if FALSE, the contour plot will not include negative values (see details). This should be set to false for the BG/BB and Pareto/NBD models.
`param.names`	a vector containing parameter names.

Details

The contour plot will have the first parameter labelled "vary" on the x-axis, and the second parameter labelled "vary" on the y-axis. It will extend out by (n.divs * zoom.percent) in both directions and both dimensions from the estimated parameter values. The exception is if allow.neg.params is FALSE. In this case, the contour plot will end at zero if it would have extended into negative parameter values.

The estimated parameter values will be indicated by the intersection of two red lines.

See Also

`dc.PlotLogLikelihoodContours`

Examples

```
# **Examples for BG/BB model:
data(donationsSummary)
rf.matrix <- donationsSummary$rf.matrix

# starting-point parameters
bgbb.startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
bgbb.est.params <- bgbb.EstimateParameters(rf.matrix, bgbb.startingparams)

# set up parameter names for a more descriptive result
bgbb.param.names <- c("alpha", "beta", "gamma", "delta")

# plot a log-likelihood contour of alpha and beta, the unobserved
# parameters for the beta-Bernoulli transaction process of the BG/BB.
# Note that allow.neg.params has been set to false as BG/BB parameters
# cannot be negative:
dc.PlotLogLikelihoodContour(bgbb.rf.matrix.LL,
                            c("vary", "vary", "fix", "fix"),
                            bgbb.est.params,
                            rf.matrix = rf.matrix,
                            n.divs = 15,
                            num.contour.lines = 15,
                            zoom.percent = 0.2,
                            allow.neg.params = FALSE,
                            param.names = bgbb.param.names)

# plot a log-likelihood contour of gamma and delta, the unobserved
# parameters for the beta-geometric dropout process of the BG/BB.
# Note that allow.neg.params has been set to false as BG/BB parameters
# cannot be negative:
dc.PlotLogLikelihoodContour(bgbb.rf.matrix.LL,
                            c("fix", "fix", "vary", "vary"),
                            bgbb.est.params,
                            rf.matrix = rf.matrix,
                            n.divs = 15,
                            num.contour.lines = 15,
                            zoom.percent = 0.2,
                            allow.neg.params = FALSE,
                            param.names = bgbb.param.names)

# **Example for Pareto/NBD model:
data(cdnowSummary)
cbs <- cdnowSummary$cbs

# Speed up calculations:
cbs <- dc.compress.cbs(cbs)

# parameters estimated using pnbd.EstimateParameters
pnbd.est.params <- cdnowSummary$est.params

# set up parameter names for a more descriptive result
pnbd.param.names <- c("r", "alpha", "s", "beta")

# plot a log-likelihood contour of r and s, the shape parameters
# of the transaction and dropout process models (respectively).
# Note that allow.neg.params has been set to false as Pareto/NBD
# parameters cannot be negative:
dc.PlotLogLikelihoodContour(pnbd.cbs.LL,
                            c("vary", "fix", "vary", "fix"),
                            pnbd.est.params,
                            cal.cbs = cbs,
                            hardie = TRUE,
                            n.divs = 20,
                            num.contour.lines = 20,
                            zoom.percent = 0.1,
                            allow.neg.params = FALSE,
                            param.names = pnbd.param.names)

# **Example for BG/NBD model:
data(cdnowSummary)
cbs <- cdnowSummary$cbs

# parameters estimated using bgnbd.EstimateParameters
bgnbd.est.params <- cdnowSummary$est.params
```

```
# set up parameter names for a more descriptive result
bgnbd.param.names <- c("r", "alpha", "s", "beta")

# plot a log-likelihood contour of r and s, the shape parameters
# of the transaction and dropout process models (respectively).
# Note that allow.neg.params has been set to false as BG/NBD
# parameters cannot be negative:
dc.PlotLogLikelihoodContour(bgnbd.cbs.LL,
                            c("vary", "fix", "vary", "fix"),
                            bgnbd.est.params,
                            cal.cbs = cbs,
                            n.divs = 20,
                            num.contour.lines = 20,
                            zoom.percent = 0.1,
                            allow.neg.params = FALSE,
                            param.names = bgnbd.param.names)
```

---

dc.PlotLogLikelihoodContours *Plot Log-Likelihood Contours*

Description

Creates a set of contour plots, such that there is a contour plot for every pair of parameters varying.

Usage

```
dc.PlotLogLikelihoodContours(
  loglikelihood.fcn,
  predicted.params,
  ...,
  n.divs = 2,
  multiple.screens = FALSE,
  num.contour.lines = 10,
  zoom.percent = 0.9,
  allow.neg.params = FALSE,
  param.names = c("param 1", "param 2", "param 3", "param 4")
)
```

Arguments

`loglikelihood.fcn`	log-likelihood function to plot.
`predicted.params`	estimated parameters.
`...`	all additional arguments required by the log-likelihood function. For example, `bgbb.rf.matrix.LL` requires rf.matrix; `pnbd.cbs.LL` requires cal.cbs and hardie (defaults to TRUE); and `bgnbd.cbs.LL` requires cal.cbs.

`n.divs`	integer representing how fine-grained the contour plot is. A higher value will produce a higher resolution plot with smoother contour lines, but will take longer to plot. n.divs also affects the boundaries of the contour plot; see details.
`multiple.screens`	if TRUE, plots each contour plot on a separate R graphics window.
`num.contour.lines`	number of contour lines to plot in the window.
`zoom.percent`	determines boundaries of contour plot. See details.
`allow.neg.params`	if FALSE, the contour plot will not include negative values (see details). This should be set to false for the BG/BB and Pareto/NBD models.
`param.names`	a vector containing parameter names.

Details

For each contour plot, the non-varying parameters are kept constant at the predicted values.

The contour will extend out by (n.divs * zoom.percent) in both directions and both dimensions from the estimated parameter values. The exception is if allow.neg.params is FALSE. In this case, the contour plot will end at zero if it would have extended into negative parameter values.

The estimated parameter values will be indicated by the intersection of two red lines.

See Also

`dc.PlotLogLikelihoodContour`

Examples

```
# **Example for BG/BB model:
data(donationsSummary)
rf.matrix <- donationsSummary$rf.matrix

# starting-point parameters
bgbb.startingparams <- c(1, 1, 0.5, 3)
# estimated parameters
bgbb.est.params <- bgbb.EstimateParameters(rf.matrix, bgbb.startingparams)

# set up parameter names for a more descriptive result
bgbb.param.names <- c("alpha", "beta", "gamma", "delta")

# plot-log likelihood contours:
dc.PlotLogLikelihoodContours(bgbb.rf.matrix.LL,
                             bgbb.est.params,
                             rf.matrix = rf.matrix,
                             n.divs = 5,
                             num.contour.lines = 8,
                             zoom.percent = 0.3,
                             allow.neg.params = FALSE,
                             param.names = bgbb.param.names)
```

```
# **Example for Pareto/NBD model:
data(cdnowSummary)
cbs <- cdnowSummary$cbs

# Speed up calculations:
cbs <- dc.compress.cbs(cbs)

# parameters estimated using pnbd.EstimateParameters
pnbd.est.params <- cdnowSummary$est.params

# set up parameter names for a more descriptive result
pnbd.param.names <- c("r", "alpha", "s", "beta")

# plot log-likelihood contours:
dc.PlotLogLikelihoodContours(pnbd.cbs.LL,
                             pnbd.est.params,
                             cal.cbs = cbs,
                             hardie = TRUE,
                             n.divs = 5,
                             num.contour.lines = 15,
                             zoom.percent = 0.3,
                             allow.neg.params = FALSE,
                             param.names = pnbd.param.names)

# **Example for BG/NBD model:
data(cdnowSummary)
cbs <- cdnowSummary$cbs

# parameters estimated using bgnbd.EstimateParameters
bgnbd.est.params <- cdnowSummary$est.params

# set up parameter names for a more descriptive result
bgnbd.param.names <- c("r", "alpha", "s", "beta")

# plot log-likelihood contours:
dc.PlotLogLikelihoodContours(bgnbd.cbs.LL,
                             bgnbd.est.params,
                             cal.cbs = cbs,
                             n.divs = 5,
                             num.contour.lines = 15,
                             zoom.percent = 0.3,
                             allow.neg.params = FALSE,
                             param.names = bgnbd.param.names)
```

`dc.ReadLines` *Read Lines*

Description

Given a .csv file that throws errors when read in by the usual read.csv and read.table methods, loops through the file line-by-line and picks out the customer, date, and sales (optional) transaction data to return an event log.

Usage

```
dc.ReadLines(csv.filename, cust.idx, date.idx, sales.idx = -1)
```

Arguments

`csv.filename`	The name of the comma-delimited file to be read. This file must contain headers.
`cust.idx`	The index of the customer ID column in the comma-delimited file.
`date.idx`	The index of the date column in the comma-delimited file.
`sales.idx`	The index of the sales column in the comma-delimited file.

Details

Once this function has been run, you may need to convert the date column to Date objects for the event log to work with other functions in this library. See the as.Date function in the `base` R package for more details.

Examples

```
# Create event log from file "cdnowElog.csv", which has
# customer IDs in the second column, dates in the third column, and
# sales numbers in the fifth column.
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5)

# convert date column to date objects, as required by some other functions
elog$date <- as.Date(elog$date, "$Y%m%d")
```

`dc.RemoveTimeBetween` *Remove Time Between*

Description

This function creates a new event log, with time in the middle removed. Used, for example, in sports with off-seasons.

Usage

```
dc.RemoveTimeBetween(elog, day1, day2, day3, day4)
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction. The "date" column must consist of date objects, not character strings.
`day1`	date of beginning of first period. Must be a date object.
`day2`	date of end of first period. Must be a date object.
`day3`	date of beginning of second period. Must be a date object.
`day4`	date of third period. Must be a date object.

Details

The four date parameters must be in ascending order.

Value

list - `elog1` the event log with all elog$date entries between day1 and day2 - `elog2` the event with all elog$date entries between day3 and day4 - `elog3` elog1 combined with elog2, with all dates from elog2 reduced by the time removed between elog1 and elog2

Examples

```
elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD"),2,3,5)
elog[,"date"] <- as.Date(elog[,"date"], "%Y%m%d")

# Use the cdnow data to return a 6 month event log for January, February,
# March, October, November, December.
period.one.start <- as.Date("1997-01-01")
period.one.end <- as.Date("1997-03-31")
period.two.start <- as.Date("1997-10-01")
period.two.end <- as.Date("1997-12-31")
reduced.elog <- dc.RemoveTimeBetween(elog, period.one.start, period.one.end,
                                     period.two.start, period.two.end)

# Note that the new elog will go up to June 30 at a maximum, since we
# are only using 6 months of data starting on January 1
max(reduced.elog$elog3$date)  # "1997-06-30"
```

`dc.SplitUpElogForRepeatTrans` *Split Up Event Log for Repeat Transactions*

Description

Turns an event log into a repeat transaction event log, removing customers' first transactions. Also returns a data frame with information about customers' first and last transactions.

Usage

```
dc.SplitUpElogForRepeatTrans(elog)
```

Arguments

`elog`	event log, which is a data frame with columns for customer ID ("cust"), date ("date"), and optionally other columns such as "sales". Each row represents an event, such as a transaction. The "date" column must contain date objects, not character strings or factors.

Value

A named list: - `repeat.trans.elog` an event log containing only repeat transactions - `cust.data` data frame containing the first and last transaction information for each customer

`dc.WriteLine` *Write Line*

Description

Writes any number of arguments to the console.

Usage

```
dc.WriteLine(...)
```

Arguments

`...`	objects to print to the R console.

Details

The code is literally: cat(..., fill=TRUE); flush.console();

`discreteSimElog` *Discrete simulated annual event log data*

Description

Data simulated using BG/BB model assumptions. Contains annual transaction behavior for a period of 14 years, for a cohort of 10,000 customers who made their first transactions in 1970.

Format

A comma-delimited file representing an event log with 52,432 entries. It has 2 columns: The customer's ID and the date of the transaction in standard R date format.

Details

This dataset was simulated in order to illustrate certain data-conversion functions (see `dc.MakeRFmatrixCal`).

`donationsSummary` *Discrete donation data summary*

Description

This dataset contains a recency-frequency matrix capturing the discrete transaction behavior of 11,104 customers over 6 transaction opportunities, summarized as a recency-frequency matrix and a vector of annual transactions.

Usage

```
data(donationsSummary)
```

Format

A named list:

$rf.matrix A matrix with 22 rows (for each possible recency-frequency combination in 6 calibration period transaction opportunities) and 4 columns: number of transactions during the calibration period ("x"), recency in the calibration period ("t.x"), number of transaction opportunities in the calibration period ("n.cal"), and number of customers with this recency-frequency combination in the calibration period ("custs").

$rf.matrix.holdout A matrix with 15 rows (for each possible recency-frequency combination in 5 holdout period transaction opportunities) and 4 columns: number of transactions during the holdout period ("x.star"), recency in the holdout period ("t.x.star"), number of transaction opportunities in the holdout period ("n.star"), and number of customers with the recency-frequency combination in the holdout period ("custs").

$x.star A vector with 22 elements, containing the number of transactions made by each calibration period recency-frequency bin in the holdout period. It is in the same order as `$rf.matrix`.

$annual.sales A vector with 11 elements, containing the number of transactions made by all customers in each time period in both the calibration and holdout periods.

Details

Data from "a major nonprofit organization located in the midwestern United States that is funded in large part by donations from individuals. In 1995 the organization "acquired" 11,104 first-time supporters; in each of the following six years, these individuals either did or did not support the organization."

This dataset contains, for each possible in-sample recency/frequency combination in the 1995 cohort, the number of customers and the number of transactions they made during the validation period.

Source

Data can be found online at `http://www.brucehardie.com/notes/010/` (Associated Excel spreadsheet)

References

Fader, Peter S., Bruce G.S. Hardie, and Jen Shang. "Customer-Base Analysis in a Discrete-Time Noncontractual Setting." *Marketing Science* 29(6), pp. 1086-1108. 2010. INFORMS. http://www.brucehardie.com/notes/020/

---

h2f1 *Use Bruce Hardie's Gaussian hypergeometric implementation*

Description

In benchmarking `pnbd.LL` runs more quickly and it returns the same results if it uses this helper instead of `hypergeo`, which is the default. But `h2f1` is such a barebones function that in some edge cases it will keep going until you get a segfault, where `hypergeo` would have failed with a proper error message.

Usage

```
h2f1(a, b, c, z)
```

Arguments

`a,`	counterpart to A in `hypergeo`
`b,`	counterpart to B in `hypergeo`
`c,`	counterpart to C in `hypergeo`
`z,`	counterpart to z in `hypergeo`

References

Fader, Peter S., and Bruce G.S. Hardie. "A Note on Deriving the Pareto/NBD Model and Related Expressions." November. 2005. Web. http://www.brucehardie.com/notes/008/

See Also

`hypergeo`

`pnbd.cbs.LL` *Pareto/NBD Log-Likelihood*

Description

Calculates the log-likelihood of the Pareto/NBD model.

Usage

```
pnbd.cbs.LL(params, cal.cbs, hardie = TRUE)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period. If your data is compressed (see `dc.compress.cbs`), a fourth column labelled "custs" (number of customers with a specific combination of recency, frequency and length of calibration period) will make this function faster.
`hardie`	if TRUE, use `h2f1` instead of `hypergeo`.

Value

The log-likelihood of the provided data.

References

Fader, Peter S., and Bruce G.S. Hardie. "A Note on Deriving the Pareto/NBD Model and Related Expressions." November. 2005. Web. `http://www.brucehardie.com/notes/008/`

See Also

`pnbd.EstimateParameters`

`pnbd.LL`

Examples

```
data(cdnowSummary)
cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# random assignment of parameters
```

```
params <- c(0.5, 8, 0.7, 10)
# returns the log-likelihood of the given parameters
pnbd.cbs.LL (params, cal.cbs, TRUE)

# compare the speed and results to the following:
cal.cbs.compressed <- dc.compress.cbs(cal.cbs)
pnbd.cbs.LL (params, cal.cbs.compressed, TRUE)
```

---

pnbd.ConditionalExpectedTransactions
Pareto/NBD Conditional Expected Transactions

Description

Uses Pareto/NBD model parameters and a customer's past transaction behavior to return the number of transactions they are expected to make in a given time period.

Usage

```
pnbd.ConditionalExpectedTransactions(
  params,
  T.star,
  x,
  t.x,
  T.cal,
  hardie = TRUE
)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`T.star`	length of time for which we are calculating the expected number of transactions.
`x`	number of repeat transactions in the calibration period T.cal, or a vector of calibration period frequencies.
`t.x`	time of most recent repeat transaction, or a vector of recencies.
`T.cal`	length of calibration period, or a vector of calibration period lengths.
`hardie`	if TRUE, have `pnbd.PAlive` use `h2f1` instead of `hypergeo`.

Details

E[X(T.cal, T.cal + T.star) | x, t.x, r, alpha, s, beta]

`T.star`, `x`, `t.x`, and `T.cal` may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Number of transactions a customer is expected to make in a time period of length t, conditional on their past behavior. If any of the input parameters has a length greater than 1, this will be a vector of expected number of transactions.

References

Fader, Peter S., and Bruce G.S. Hardie. "A Note on Deriving the Pareto/NBD Model and Related Expressions." November. 2005. Web. http://www.brucehardie.com/notes/008/

See Also

pnbd.Expectation

Examples

```
params <- c(0.55, 10.56, 0.61, 11.64)
# Number of transactions a customer is expected to make in 2 time
# intervals, given that they made 10 repeat transactions in a time period
# of 39 intervals, with the 10th repeat transaction occurring in the 35th
# interval.
pnbd.ConditionalExpectedTransactions(params,
                                     T.star = 2,
                                     x = 10,
                                     t.x = 35,
                                     T.cal = 39,
                                     hardie = TRUE)

# We can also compare expected transactions across different
# calibration period behaviors:
pnbd.ConditionalExpectedTransactions(params,
                                     T.star = 2,
                                     x = 5:20,
                                     t.x = 25,
                                     T.cal = 39,
                                     hardie = TRUE)
```

---

pnbd.DERT *Pareto/NBD Discounted Expected Residual Transactions*

Description

Calculates the discounted expected residual transactions of a customer, given their behavior during the calibration period.

Usage

```
pnbd.DERT(params, x, t.x, T.cal, d, hardie = TRUE)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`x`	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
`t.x`	time of most recent repeat transaction, or a vector of recencies.
`T.cal`	length of calibration period, or a vector of calibration period lengths.
`d`	the discount rate to be used. Make sure that it matches up with your chosen time period (do not use an annual rate for monthly data, for example).
`hardie`	if TRUE, use `h2f1` instead of `hypergeo`.

Details

DERT(d | r, alpha, s, beta, X = x, t.x, T.cal)

x, t.x, T.cal may be vectors. The standard rules for vector operations apply

- if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be also be a vector.

Value

The number of discounted expected residual transactions for a customer with a particular purchase pattern during the calibration period.

References

Fader, Peter S., Bruce G.S. Hardie, and Ka L. Lee. "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." Journal of Marketing Research Vol.42, pp.415-430. November. 2005. http://www.brucehardie.com/papers.html

See equation 2.

Note that this paper refers to what this package is calling discounted expected residual transactions (DERT) simply as discounted expected transactions (DET).

Examples

```
# elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD2"),2,3)
# elog[, 'date'] <- as.Date(elog[, 'date'], format = '%Y%m%d')
# cal.cbs <- dc.ElogToCbsCbt(elog)$cal$cbs
# params <- pnbd.EstimateParameters(cal.cbs, hardie = TRUE)
params <- c(0.5629966, 12.5590370, 0.4081095, 10.5148048)

# 15% compounded annually has been converted to 0.0027 compounded continuously,
# as we are dealing with weekly data and not annual data.
d <- 0.0027
```

```
# calculate the discounted expected residual transactions of a customer
# who made 7 transactions in a calibration period that was 77.86
# weeks long, with the last transaction occurring at the end of
# the 35th week.
pnbd.DERT(params,
          x = 7,
          t.x = 35,
          T.cal = 77.86,
          d,
          hardie = TRUE)

# We can also use vectors to compute DERT for several customers:
pnbd.DERT(params,
          x = 1:10,
          t.x = 30,
          T.cal = 77.86,
          d,
          hardie = TRUE)
```

---

pnbd.EstimateParameters

Pareto/NBD Parameter Estimation

Description

The best-fitting parameters are determined using the pnbd.cbs.LL function. The sum of the log-likelihood for each customer (for a set of parameters) is maximized in order to estimate parameters.

Usage

```
pnbd.EstimateParameters(
  cal.cbs,
  par.start = c(1, 1, 1, 1),
  max.param.value = 10000,
  method = "L-BFGS-B",
  hardie = TRUE,
  hessian = FALSE
)
```

Arguments

cal.cbs	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period. If your data is compressed (see dc.compress.cbs), a fourth column labelled "custs" (number of customers with a specific combination of recency, frequency and length of calibration period) will make this function faster.

`par.start`	initial Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`max.param.value`	the upper bound on parameters.
`method`	the optimization method(s).
`hardie`	if TRUE, have `pnbd.LL` use `h2f1` instead of `hypergeo`.
`hessian`	set it to TRUE if you want the Hessian matrix, and then you might as well have the complete `optimx` object returned.

Details

A set of starting parameters must be provided for this method. If no parameters are provided, (1,1,1,1) is used as a default. It may be useful to use starting values for r and s that represent your best guess of the heterogeneity in the buy and die rate of customers. It may be necessary to run the estimation from multiple starting points to ensure that it converges. To compare the log-likelihoods of different parameters, use `pnbd.cbs.LL`.

The lower bound on the parameters to be estimated is always zero, since Pareto/NBD parameters cannot be negative. The upper bound can be set with the max.param.value parameter.

This function may take some time to run. It uses `optimx` for maximum likelihood estimation, not `optim`.

Value

Unnamed vector of estimated parameters by default, `optimx` object with everything if `hessian` is TRUE.

References

Fader, Peter S.; Hardie, and Bruce G.S.. "Overcoming the BG/NBD Model's #NUM! Error Problem." December. 2013. Web. http://brucehardie.com/notes/027/bgnbd_num_error.pdf

See Also

`pnbd.cbs.LL`

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# starting-point parameters
startingparams <- c(0.5, 6, 0.9, 8)

# estimated parameters
est.params <- pnbd.EstimateParameters(cal.cbs = cal.cbs,
                                            par.start = startingparams,
                                            method = 'L-BFGS-B',
                                            hardie = TRUE)

# complete object returned by \code{\link[optimx]{optimx}}
optimx.set <- pnbd.EstimateParameters(cal.cbs = cal.cbs,
                                      par.start = startingparams,
                                      hardie = TRUE,
                                      hessian = TRUE)

# log-likelihood of estimated parameters
pnbd.cbs.LL(est.params, cal.cbs, TRUE)
```

pnbd.Expectation *Pareto/NBD Expectation*

Description

Returns the number of repeat transactions that a randomly chosen customer (for whom we have no prior information) is expected to make in a given time period.

Usage

```
pnbd.Expectation(params, t)
```

Arguments

params	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
t	The length of time for which we are calculating the expected number of repeat transactions.

Details

E(X(t) | r, alpha, s, beta)

Value

Number of repeat transactions a customer is expected to make in a time period of length t.

References

Fader, Peter S., and Bruce G.S. Hardie. "A Note on Deriving the Pareto/NBD Model and Related Expressions." November. 2005. Web. http://www.brucehardie.com/notes/008/

See Also

pnbd.ConditionalExpectedTransactions

Examples

```
params <- c(0.55, 10.56, 0.61, 11.64)

# Number of repeat transactions a customer is expected to make in 2 time intervals.
pnbd.Expectation(params = params,
                 t = 2)

# We can also compare expected transactions over time:
pnbd.Expectation(params = params,
                 t = 1:10)
```

pnbd.ExpectedCumulativeTransactions

Pareto/NBD Expected Cumulative Transactions

Description

Calculates the expected cumulative total repeat transactions by all customers for the calibration and holdout periods.

Usage

```
pnbd.ExpectedCumulativeTransactions(params, T.cal, T.tot, n.periods.final)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`T.cal`	length of calibration period, or a vector of calibration period lengths.
`T.tot`	End of holdout period. Must be a single value, not a vector.
`n.periods.final`	Number of time periods in the calibration and holdout periods. See details.

Details

The function automatically divides the total period up into n.periods.final time intervals. n.periods.final does not have to be in the same unit of time as the T.cal data. For example: - if your T.cal data is in weeks, and you want cumulative transactions per week, n.periods.final would equal T.star. - if your T.cal data is in weeks, and you want cumulative transactions per day, n.periods.final would equal T.star * 7.

The holdout period should immediately follow the calibration period. This function assume that all customers' calibration periods end on the same date, rather than starting on the same date (thus customers' birth periods are determined using max(T.cal) - T.cal rather than assuming that it is 0).

Value

Vector of expected cumulative total repeat transactions by all customers.

See Also

pnbd.Expectation

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

params <- c(0.55, 10.56, 0.61, 11.64)

# Returns a vector containing cumulative repeat transactions for 546 days.
# All parameters are in weeks; the calibration period lasted 39 weeks
# and the holdout period another 39.
pnbd.ExpectedCumulativeTransactions(params = params,
                                    T.cal = cal.cbs[,"T.cal"],
                                    T.tot = 78,
                                    n.periods.final = 546)
```

---

pnbd.generalParams *Define general parameters*

Description

This is to ensure consistency across all functions that require the likelihood function, or the log of it, and to make sure that the same implementation of the hypergeometric function is used everywhere for building `A0`.

Usage

```
pnbd.generalParams(params, x, t.x, T.cal, func, hardie = TRUE)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`x`	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
`t.x`	time of most recent repeat transaction, or a vector of recencies.
`T.cal`	length of calibration period, or a vector of calibration period lengths.
`func`	name of the function calling dc.InputCheck; either `pnbd.LL` or `pnbd.PAlive`.
`hardie`	if TRUE, use `h2f1` instead of `hypergeo`.

Details

This function is only ever called by either pnbd.LL or pnbd.PAlive so it returns directly the output that is expected from those calling functions: either the log likelihood for a set of customers, or the probability that a set of customers with characteristics given by `x`, `t.x` and `T.cal`, having estimated a set of `params`, is still alive. Either set of customers can be of size 1.

Value

A vector of log likelihood values if `func` is `pnbd.LL`, or a vector of probabilities that a customer is still alive if `func` is `pnbd.PAlive`.

See Also

pnbd.LL

pnbd.PAlive

pnbd.DERT

---

pnbd.LL *Pareto/NBD Log-Likelihood*

Description

Calculates the log-likelihood of the Pareto/NBD model.

Usage

```
pnbd.LL(params, x, t.x, T.cal, hardie = TRUE)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`x`	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
`t.x`	time of most recent repeat transaction, or a vector of recencies.
`T.cal`	length of calibration period, or a vector of calibration period lengths.
`hardie`	if TRUE, use h2f1 instead of hypergeo.

Value

A vector of log-likelihoods as long as the longest input vector (x, t.x, or T.cal).

References

Fader, Peter S., and Bruce G.S. Hardie. "A Note on Deriving the Pareto/NBD Model and Related Expressions." November. 2005. Web. http://www.brucehardie.com/notes/008/

See Also

pnbd.EstimateParameters

Examples

```
# Returns the log likelihood of the parameters for a customer who
# made 3 transactions in a calibration period that ended at t=6,
# with the last transaction occurring at t=4.
pnbd.LL(params, x=3, t.x=4, T.cal=6, hardie = TRUE)

# We can also give vectors as function parameters:
set.seed(7)
x <- sample(1:4, 10, replace = TRUE)
t.x <- sample(1:4, 10, replace = TRUE)
T.cal <- rep(4, 10)
pnbd.LL(params, x, t.x, T.cal, hardie = TRUE)
```

`pnbd.PAlive` *Pareto/NBD P(Alive)*

Description

Uses Pareto/NBD model parameters and a customer's past transaction behavior to return the probability that they are still alive at the end of the calibration period.

Usage

```
pnbd.PAlive(params, x, t.x, T.cal, hardie = TRUE)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`x`	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
`t.x`	time of most recent repeat transaction, or a vector of recencies.
`T.cal`	length of calibration period, or a vector of calibration period lengths.
`hardie`	if TRUE, use `h2f1` instead of `hypergeo`.

Details

P(Alive | X=x, t.x, T.cal, r, alpha, s, beta)

x, t.x, and T.cal may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Probability that the customer is still alive at the end of the calibration period. If x, t.x, and/or T.cal has a length greater than one, then this will be a vector of probabilities (containing one element matching each element of the longest input vector).

References

Fader, Peter S., and Bruce G.S. Hardie. "A Note on Deriving the Pareto/NBD Model and Related Expressions." November. 2005. Web. http://www.brucehardie.com/notes/008/

Examples

```
data(cdnowSummary)
cbs <- cdnowSummary$cbs
params <- pnbd.EstimateParameters(cbs, hardie = TRUE)

pnbd.PAlive(params, x=0, t.x=0, T.cal=39, TRUE)
# 0.2941633; P(Alive) of a customer who made no repeat transactions.

pnbd.PAlive(params, x=23, t.x=39, T.cal=39, TRUE)
# 1; P(Alive) of a customer who has the same recency and total
# time observed.

pnbd.PAlive(params, x=5:20, t.x=30, T.cal=39, TRUE)
# Note the "increasing frequency paradox".

# To visualize the distribution of P(Alive) across customers:
p.alives <- pnbd.PAlive(params, cbs[,"x"], cbs[,"t.x"], cbs[,"T.cal"], TRUE)
plot(density(p.alives))
```

`pnbd.Plot.DERT` *Pareto/NBD Plot Discounted Expected Residual Transactions*

Description

Plots discounted expected residual transactions for different combinations of calibration period frequency and recency.

Usage

```
pnbd.Plot.DERT(params, x, t.x, T.cal, d, hardie = TRUE, type = "persp")
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.

`x`	number of repeat transactions in the calibration period T.cal, or a vector of transaction frequencies.
`t.x`	time of most recent repeat transaction, or a vector of recencies.
`T.cal`	length of calibration period, or a vector of calibration period lengths.
`d`	the discount rate to be used. Make sure that it matches up with your chosen time period (do not use an annual rate for monthly data, for example).
`hardie`	if TRUE, use `h2f1` instead of `hypergeo`.
`type`	must be either "persp" (perspective - 3 dimensional) or "contour". Determines the type of plot produced by this function.

Details

The length of the calibration period `T.cal` must be a single value, not a vector.

Value

A matrix with discounted expected residual transaction values for every combination of calibration period frequency `x` and calibration period recency `t.x`.

References

Fader, Peter S., Bruce G.S. Hardie, and Ka L. Lee. "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." Journal of Marketing Research Vol.42, pp.415-430. November. 2005. http://www.brucehardie.com/papers.html

Note that this paper refers to what this package is calling discounted expected residual transactions (DERT) simply as discounted expected transactions (DET).

Examples

```
# The RFM and CLV paper uses all 78 weeks of the cdnow data to
# estimate parameters. These parameters can be estimated as follows:
# elog <- dc.ReadLines(system.file("data/cdnowElog.csv", package="BTYD2"),2,3)
# elog[, 'date'] <- as.Date(elog[, 'date'], format = '%Y%m%d')
# cal.cbs <- dc.ElogToCbsCbt(elog)$cal$cbs
# pnbd.EstimateParameters(cal.cbs, hardie = TRUE)

# (The final function was run several times with its own output as
# input for starting parameters, to ensure that the result converged).

params <- c(0.5629966, 12.5590370, 0.4081095, 10.5148048)

# 15% compounded annually has been converted to 0.0027 compounded continously,
# as we are dealing with weekly data and not annual data.
d <- 0.0027

pnbd.Plot.DERT(params = params,
               x = 0:14,
               t.x = 0:77,
               T.cal = 77.86,
               d = d,
               hardie = TRUE,
               type = "persp")
pnbd.Plot.DERT(params = params,
               x = 0:14,
               t.x = 0:77,
               T.cal = 77.86,
               d = d,
               hardie = TRUE,
               type="contour")
```

pnbd.PlotDropoutRateHeterogeneity

Pareto/NBD Plot Dropout Rate Heterogeneity

Description

Plots and returns the estimated gamma distribution of mu (customers' propensities to drop out).

Usage

```
pnbd.PlotDropoutRateHeterogeneity(params, lim = NULL)
```

Arguments

params	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
lim	The upper-bound of the x-axis. A number is chosen by the function if none is provided.

Details

This returns the distribution of each customer's exponential parameter that determines their lifetime (using the Pareto/NBD assumption that a customer's lifetime can be modeled with an exponential distribution).

Value

Distribution of customers' propensities to drop out.

Examples

```
params <- c(0.55, 10.56, 0.61, 11.64)
pnbd.PlotDropoutRateHeterogeneity(params)
params <- c(0.55, 10.56, 3, 11.64)
pnbd.PlotDropoutRateHeterogeneity(params)
```

pnbd.PlotFrequencyInCalibration
Pareto/NBD Plot Frequency in Calibration Period

Description

Plots a histogram and returns a matrix comparing the actual and expected number of customers who made a certain number of repeat transactions in the calibration period, binned according to calibration period frequencies.

Usage

```
pnbd.PlotFrequencyInCalibration(
  params,
  cal.cbs,
  censor,
  hardie = TRUE,
  plotZero = TRUE,
  xlab = "Calibration period transactions",
  ylab = "Customers",
  title = "Frequency of Repeat Transactions"
)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x") and total time observed ("T.cal").
`censor`	integer used to censor the data. See details.
`hardie`	if TRUE, have `pnbd.pmf` use `h2f1` instead of `hypergeo`.
`plotZero`	if FALSE, the histogram will exclude the zero bin.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`title`	title placed on the top-center of the plot.

Details

This function requires a censor number, which cannot be higher than the highest frequency in the calibration period CBS. The output matrix will have (censor + 1) bins, starting at frequencies of 0 transactions and ending at a bin representing calibration period frequencies at or greater than the censor number. The plot may or may not include a bin for zero frequencies, depending on the plotZero parameter.

Value

Calibration period repeat transaction frequency comparison matrix (actual vs. expected).

Examples

```
data(cdnowSummary)
cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# parameters estimated using pnbd.EstimateParameters
est.params <- cdnowSummary$est.params
# the maximum censor number that can be used
max(cal.cbs[,"x"])

pnbd.PlotFrequencyInCalibration(params = est.params,
                                cal.cbs = cal.cbs,
                                censor = 7,
                                hardie = TRUE)
```

---

pnbd.PlotFreqVsConditionalExpectedFrequency

Pareto/NBD Plot Frequency vs. Conditional Expected Frequency

Description

Plots the actual and conditional expected number transactions made by customers in the holdout period, binned according to calibration period frequencies. Also returns a matrix with this comparison and the number of customers in each bin.

Usage

```
pnbd.PlotFreqVsConditionalExpectedFrequency(
  params,
  T.star,
  cal.cbs,
  x.star,
  censor,
  hardie = TRUE,
  xlab = "Calibration period transactions",
  ylab = "Holdout period transactions",
  xticklab = NULL,
  title = "Conditional Expectation"
)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`T.star`	length of the holdout period. It must be a scalar for this plot's purposes: you have one holdout period of a given length.
`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`x.star`	vector of transactions made by each customer in the holdout period.
`censor`	integer used to censor the data. See details.
`hardie`	if TRUE, have `pnbd.ConditionalExpectedTransactions` use `h2f1` instead of `hypergeo`.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

This function requires a censor number, which cannot be higher than the highest frequency in the calibration period CBS. The output matrix will have (censor + 1) bins, starting at frequencies of 0 transactions and ending at a bin representing calibration period frequencies at or greater than the censor number.

Value

Holdout period transaction frequency comparison matrix (actual vs. expected).

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# number of transactions by each customer in the 39 weeks
# following the calibration period
x.star <- cal.cbs[,"x.star"]

# parameters estimated using pnbd.EstimateParameters
est.params <- cdnowSummary$est.params
# the maximum censor number that can be used
max(cal.cbs[,"x"])
```

```
# plot conditional expected holdout period frequencies,
# binned according to calibration period frequencies
pnbd.PlotFreqVsConditionalExpectedFrequency(params = est.params,
                                            T.star = 39,
                                            cal.cbs = cal.cbs,
                                            x.star = x.star,
                                            censor = 7,
                                            hardie = TRUE)
```

pnbd.PlotRateHeterogeneity

Plot Pareto/NBD Rate Heterogeneity

Description

A helper for plotting either the estimated gamma distribution of mu (customers' propensities to drop out), or the estimated gamma distribution of lambda (customers' propensities to purchase).

Usage

```
pnbd.PlotRateHeterogeneity(params, func, lim = NULL)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`func`	A string that is either "pnbd.PlotDropoutRateHeterogeneity" or "pnbd.PlotTransactionRateHeterogeneity"
`lim`	The upper-bound of the x-axis. A number is chosen by the function if none is provided.

Value

Depending on the value of `func`, either the distribution of customers' propensities to purchase or the distribution of customers' propensities to drop out.

See Also

`pnbd.PlotDropoutRateHeterogeneity`

`pnbd.PlotTransactionRateHeterogeneity`

`pnbd.PlotRecVsConditionalExpectedFrequency` *Pareto/NBD Plot Actual vs. Conditional Expected Frequency by Recency*

Description

Plots the actual and conditional expected number of transactions made by customers in the holdout period, binned according to calibration period recencies. Also returns a matrix with this comparison and the number of customers in each bin.

Usage

```
pnbd.PlotRecVsConditionalExpectedFrequency(
  params,
  cal.cbs,
  T.star,
  x.star,
  hardie = TRUE,
  xlab = "Calibration period recency",
  ylab = "Holdout period transactions",
  xticklab = NULL,
  title = "Actual vs. Conditional Expected Transactions by Recency"
)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`cal.cbs`	calibration period CBS (customer by sufficient statistic). It must contain columns for frequency ("x"), recency ("t.x"), and total time observed ("T.cal"). Note that recency must be the time between the start of the calibration period and the customer's last transaction, not the time between the customer's last transaction and the end of the calibration period.
`T.star`	length of the holdout period. It must be a scalar for this plot's purposes: you have one holdout period of a given length.
`x.star`	vector of transactions made by each customer in the holdout period.
`hardie`	if TRUE, have `pnbd.ConditionalExpectedTransactions` use `h2f1` instead of `hypergeo`.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`xticklab`	vector containing a label for each tick mark on the x axis.
`title`	title placed on the top-center of the plot.

Details

This function does bin customers exactly according to recency; it bins customers according to integer units of the time period of cal.cbs. Therefore, if you are using weeks in your data, customers will be binned as follows: customers with recencies between the start of the calibration period (inclusive) and the end of week one (exclusive); customers with recencies between the end of week one (inclusive) and the end of week two (exlusive); etc.

The matrix and plot will contain the actual number of transactions made by each bin in the holdout period, as well as the expected number of transactions made by that bin in the holdout period, conditional on that bin's behavior during the calibration period.

Value

Matrix comparing actual and conditional expected transactions in the holdout period.

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# number of transactions by each customer in the 39 weeks following
# the calibration period
x.star <- cal.cbs[,"x.star"]

# parameters estimated using pnbd.EstimateParameters
est.params <- cdnowSummary$est.params

# plot conditional expected holdout period transactions, binned according to
# calibration period recencies
pnbd.PlotRecVsConditionalExpectedFrequency(params = est.params,
                                           cal.cbs = cal.cbs,
                                           T.star = 39,
                                           x.star = x.star,
                                           hardie = TRUE)
```

`pnbd.PlotTrackingCum` *Pareto/NBD Tracking Cumulative Transactions Plot*

Description

Plots the actual and expected cumulative total repeat transactions by all customers for the calibration and holdout periods, and returns this comparison in a matrix.

Usage

```
pnbd.PlotTrackingCum(
  params,
  T.cal,
  T.tot,
  actual.cu.tracking.data,
  n.periods.final = NA,
  xlab = "Week",
  ylab = "Cumulative Transactions",
  xticklab = NULL,
  title = "Tracking Cumulative Transactions"
)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`T.cal`	length of calibration period, or a vector of calibration period lengths.
`T.tot`	End of holdout period. Must be a single value, not a vector.
`actual.cu.tracking.data`	A vector containing the cumulative number of repeat transactions made by customers for each period in the total time period (both calibration and holdout periods). See details.
`n.periods.final`	Number of time periods in the calibration and holdout periods. See details.
`xlab`	Descriptive label for the x axis.
`ylab`	Descriptive label for the y axis.
`xticklab`	Vector containing a label for each tick mark on the x axis.
`title`	Title placed on the top-center of the plot.

Details

actual.cu.tracking.data does not have to be in the same unit of time as the T.cal data. T.tot will automatically be divided into periods to match the length of actual.cu.tracking.data. See `pnbd.ExpectedCumulativeTransactio`

The holdout period should immediately follow the calibration period. This function assume that all customers' calibration periods end on the same date, rather than starting on the same date (thus customers' birth periods are determined using max(T.cal) - T.cal rather than assuming that it is 0).

Value

Matrix containing actual and expected cumulative repeat transactions.

Examples

```
data(cdnowSummary)

cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# Cumulative repeat transactions made by all customers across calibration
# and holdout periods
cu.tracking <- cdnowSummary$cu.tracking

# parameters estimated using pnbd.EstimateParameters
est.params <- cdnowSummary$est.params

# All parameters are in weeks; the calibration period lasted 39
# weeks and the holdout period another 39.
pnbd.PlotTrackingCum(params = est.params,
                     T.cal = cal.cbs[,"T.cal"],
                     T.tot = 78,
                     actual.cu.tracking.data = cu.tracking)
```

---

pnbd.PlotTrackingInc *Pareto/NBD Tracking Incremental Transactions Comparison*

Description

Plots the actual and expected incremental total repeat transactions by all customers for the calibration and holdout periods, and returns this comparison in a matrix.

Usage

```
pnbd.PlotTrackingInc(
  params,
  T.cal,
  T.tot,
  actual.inc.tracking.data,
  n.periods.final = NA,
  xlab = "Week",
  ylab = "Transactions",
  xticklab = NULL,
  title = "Tracking Weekly Transactions"
)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`T.cal`	length of calibration period, or a vector of calibration period lengths.

`T.tot`	End of holdout period. Must be a single value, not a vector.
`actual.inc.tracking.data`	A vector containing the incremental number of repeat transactions made by customers for each period in the total time period (both calibration and holdout periods). See details.
`n.periods.final`	Number of time periods in the calibration and holdout periods. See details.
`xlab`	Descriptive label for the x axis.
`ylab`	Descriptive label for the y axis.
`xticklab`	Vector containing a label for each tick mark on the x axis.
`title`	Title placed on the top-center of the plot.

Details

actual.inc.tracking.data does not have to be in the same unit of time as the T.cal data. T.tot will automatically be divided into periods to match the length of actual.inc.tracking.data. See `pnbd.ExpectedCumulativeTransactic`

The holdout period should immediately follow the calibration period. This function assume that all customers' calibration periods end on the same date, rather than starting on the same date (thus customers' birth periods are determined using max(T.cal) - T.cal rather than assuming that it is 0).

Value

Matrix containing actual and expected incremental repeat transactions.

Examples

```
data(cdnowSummary)
cal.cbs <- cdnowSummary$cbs
# cal.cbs already has column names required by method

# Cumulative repeat transactions made by all customers across calibration
# and holdout periods
cu.tracking <- cdnowSummary$cu.tracking
# make the tracking data incremental
inc.tracking <- dc.CumulativeToIncremental(cu.tracking)

# parameters estimated using pnbd.EstimateParameters
est.params <- cdnowSummary$est.params

# All parameters are in weeks; the calibration period lasted 39
# weeks and the holdout period another 39.
pnbd.PlotTrackingInc(params = est.params,
                     T.cal = cal.cbs[,"T.cal"],
                     T.tot = 78,
                     actual.inc.tracking.data = inc.tracking)
```

pnbd.PlotTransactionRateHeterogeneity
Pareto/NBD Plot Transaction Rate Heterogeneity

Description

Plots and returns the estimated gamma distribution of lambda (customers' propensities to purchase).

Usage

```
pnbd.PlotTransactionRateHeterogeneity(params, lim = NULL)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`lim`	The upper-bound of the x-axis. A number is chosen by the function if none is provided.

Details

This returns the distribution of each customer's Poisson parameter, which determines the level of their purchasing (using the Pareto/NBD assumption that purchasing on the individual level can be modeled with a Poisson distribution).

Value

Distribution of customers' propensities to purchase.

Examples

```
params <- c(0.55, 10.56, 0.61, 11.64)
pnbd.PlotTransactionRateHeterogeneity(params)
params <- c(3, 10.56, 0.61, 11.64)
pnbd.PlotTransactionRateHeterogeneity(params)
```

`pnbd.pmf` *Pareto/NBD Probability Mass Function*

Description

Probability mass function for the Pareto/NBD.

Usage

```
pnbd.pmf(params, t, x, hardie = TRUE)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`t`	length end of time period for which probability is being computed. May also be a vector.
`x`	number of repeat transactions by a random customer in the period defined by t. May also be a vector.
`hardie`	if TRUE, have `pnbd.pmf.General` use `h2f1` instead of `hypergeo`.

Details

P(X(t)=x | r, alpha, s, beta). Returns the probability that a customer makes x repeat transactions in the time interval (0, t].

Parameters `t` and `x` may be vectors. The standard rules for vector operations apply - if they are not of the same length, the shorter vector will be recycled (start over at the first element) until it is as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Probability of X(t)=x conditional on model parameters. If t and/or x has a length greater than one, a vector of probabilities will be returned.

References

Fader, Peter S., and Bruce G.S. Hardie. “Deriving an Expression for P (X(t) = x) Under the Pareto/NBD Model.” Sept. 2006. Web. `http://www.brucehardie.com/notes/012/`

Examples

```
params <- c(0.55, 10.56, 0.61, 11.64)
# probability that a customer will make 10 repeat transactions in the
# time interval (0,2]
pnbd.pmf(params, t=2, x=10, hardie = TRUE)
# probability that a customer will make no repeat transactions in the
# time interval (0,39]
pnbd.pmf(params, t=39, x=0, hardie = TRUE)

# Vectors may also be used as arguments:
pnbd.pmf(params = params,
         t = 30,
         x = 11:20,
         hardie = TRUE)
```

---

pnbd.pmf.General *Generalized Pareto/NBD Probability Mass Function*

Description

Generalized probability mass function for the Pareto/NBD.

Usage

```
pnbd.pmf.General(params, t.start, t.end, x, hardie = TRUE)
```

Arguments

`params`	Pareto/NBD parameters - a vector with r, alpha, s, and beta, in that order. r and alpha are unobserved parameters for the NBD transaction process. s and beta are unobserved parameters for the Pareto (exponential gamma) dropout process.
`t.start`	start of time period for which probability is being calculated. It can also be a vector of values.
`t.end`	end of time period for which probability is being calculated. It can also be a vector of values.
`x`	number of repeat transactions by a random customer in the period defined by (t.start, t.end]. It can also be a vector of values.
`hardie`	if TRUE, use `h2f1` instead of `hypergeo`.

Details

P(X(t.start, t.end)=x | r, alpha, s, beta). Returns the probability that a customer makes x repeat transactions in the time interval (t.start, t.end].

It is impossible for a customer to make a negative number of repeat transactions. This function will return an error if it is given negative times or a negative number of repeat transactions. This function will also return an error if t.end is less than t.start.

`t.start`, `t.end`, and x may be vectors. The standard rules for vector operations apply - if they are not of the same length, shorter vectors will be recycled (start over at the first element) until they are as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

Probability of x transaction occuring between t.start and t.end conditional on model parameters. If t.start, t.end, and/or x has a length greater than one, a vector of probabilities will be returned.

References

Fader, Peter S., and Bruce G.S. Hardie. "Deriving an Expression for P (X(t) = x) Under the Pareto/NBD Model." Sept. 2006. Web. http://www.brucehardie.com/notes/012/

Fader, Peter S., Bruce G.S. Hardie, and Kinshuk Jerath. "Deriving an Expression for P (X(t, t + tau) = x) Under the Pareto/NBD Model." Sept. 2006. Web. http://www.brucehardie.com/notes/013/

Examples

```
# probability that a customer will make 10 repeat transactions in the
# time interval (1,2]
data("cdnowSummary")
cal.cbs <- cdnowSummary$cbs
params <- pnbd.EstimateParameters(cal.cbs = cal.cbs,
                                  method = "L-BFGS-B",
                                  hardie = TRUE)
pnbd.pmf.General(params, t.start=1, t.end=2, x=10, hardie = TRUE)
# probability that a customer will make no repeat transactions in the
# time interval (39,78]
pnbd.pmf.General(params,
                 t.start = 39,
                 t.end = 78,
                 x = 0,
                 hardie = TRUE)
```

`spend.EstimateParameters` *Spend Parameter Estimation*

Description

Estimates parameters for the gamma-gamma spend model.

Usage

```
spend.EstimateParameters(
  m.x.vector,
  x.vector,
  par.start = c(1, 1, 1),
  max.param.value = 10000
)
```

Arguments

`m.x.vector`	a vector with each customer's average observed transaction value in the calibration period.
`x.vector`	a vector with the number of transactions each customer made in the calibration period. Must correspond to m.x.vector in terms of ordering of customers and length of the vector.
`par.start`	initial vector of gamma-gamma parameters: p, q, and gamma, in that order. p is the shape parameter for each transaction. The scale parameter for each transaction is distributed across customers according to a gamma distribution with parameters q (shape) and gamma (scale).
`max.param.value`	the upper bound on parameters.

Details

The best-fitting parameters are determined using the spend.LL function. The sum of the log-likelihood for each customer (for a set of parameters) is maximized in order to estimate parameters.

A set of starting parameters must be provided for this method. If no parameters are provided, (1,1,1,1) is used as a default. It may be necessary to run the estimation from multiple starting points to ensure that it converges. To compare the log-likelihoods of different parameters, use spend.LL.

The lower bound on the parameters to be estimated is always zero, since gamma-gamma parameters cannot be negative. The upper bound can be set with the max.param.value parameter.

Value

Vector of estimated parameters.

Examples

```
## Not run:
data(cdnowSummary)
ave.spend <- cdnowSummary$m.x
tot.trans <- cdnowSummary$cbs[,"x"]

# There will be many warnings due to the zeroes that are
# included in the data above. To avoid them, use the following:
# (see example for spend.LL)

ave.spend <- ave.spend[which(tot.trans > 0)]
```

```
tot.trans <- tot.trans[which(tot.trans > 0)]

# We will let the spend function use default starting parameters
spend.EstimateParameters(ave.spend, tot.trans)

## End(Not run)
```

`spend.expected.value` *Conditional expected transaction value*

Description

Calculates the expected transaction value for a customer, conditional on the number of transaction and average transaction value during the calibration period.

Usage

```
spend.expected.value(params, m.x, x)
```

Arguments

`params`	a vector of gamma-gamma parameters: p, q, and gamma, in that order. p is the shape parameter for each transaction. The scale parameter for each transaction is distributed across customers according to a gamma distribution with parameters q (shape) and gamma (scale).
`m.x`	the customer's average observed transaction value in the calibration period. May also be a vector of average observed transaction values - see details.
`x`	the number of transactions the customer made in the calibration period. May also be a vector of frequencies - see details.

Details

E(M | p, q, gamma, m.x, x).

m.x and x may be vectors. The standard rules for vector operations apply - if they are not of the same length, the shorter vector will be recycled (start over at the first element) until it is as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

Value

The expected transaction value for a customer conditional on their transaction behavior during the calibration period. If m.x or x has a length greater than one, then a vector of expected transaction values will be returned.

References

Fader, Peter S., Bruce G.S. Hardie, and Ka L. Lee. “RFM and CLV: Using Iso-Value Curves for Customer Base Analysis.” Journal of Marketing Research Vol.42, pp.415-430. November. 2005. Web.

Examples

```
## Not run:
data(cdnowSummary)
ave.spend <- cdnowSummary$m.x
tot.trans <- cdnowSummary$cbs[,"x"]
# params <- c(6, 4, 16); # in original documentation. rounded values of:
params <- spend.EstimateParameters(m.x.vector = ave.spend, x.vector = tot.trans);
# calculate the expected transaction value of a customer
# who spent an average of $35 over 3 transactions.
spend.expected.value(params, m.x=35, x=3)

# m.x and x may be vectors:
spend.expected.value(params, m.x=30:40, x=3)
spend.expected.value(params, m.x=35, x=1:10)
spend.expected.value(params, m.x=30:40, x=1:11)

## End(Not run)
```

`spend.generalParams` *Define general parameters*

Description

This is to ensure consistency across all spend functions.

Usage

```
spend.generalParams(params, func, m.x, x)
```

Arguments

`params`	a vector of gamma-gamma parameters: p, q, and gamma, in that order. p is the shape parameter for each transaction. The scale parameter for each transaction is distributed across customers according to a gamma distribution with parameters q (shape) and gamma (scale).
`func`	name of the function calling `dc.InputCheck`.
`m.x`	the customer’s average observed transaction value in the calibration period. May also be a vector of average observed transaction values - see details.
`x`	the number of transactions the customer made in the calibration period. May also be a vector of frequencies - see details.

Details

This function is only ever called by functions defined in the original BTYD package, such as `spend.LL`, `spend.marginal.likelihood` or `spend.expected.value` so it returns directly the output that is expected from those calling functions.

Value

That depends on `func`: 1. If `func` is `spend.marginal.likelihood`, the marginal distribution of a customer's average transaction value (if m.x or x has a length greater than 1, a vector of marginal likelihoods will be returned). 2. If `func` is `spend.LL`, the log-likelihood of the gamma-gamma model; if m.x or x has a length greater than 1, this is a vector of log-likelihoods. 3. If `func` is `spend.expected.value`, the expected transaction value for a customer conditional on their transaction behavior during the calibration period. If m.x or x has a length greater than one, then a vector of expected transaction values will be returned.

See Also

`spend.LL`

`spend.marginal.likelihood`

`spend.LL` *Spend Log-Likelihood*

Description

Calculates the log-likelihood of the gamma-gamma model for customer spending.

Usage

```
spend.LL(params, m.x, x)
```

Arguments

`params`	a vector of gamma-gamma parameters: p, q, and gamma, in that order. p is the shape parameter for each transaction. The scale parameter for each transaction is distributed across customers according to a gamma distribution with parameters q (shape) and gamma (scale).
`m.x`	the customer's average observed transaction value in the calibration period. May also be a vector of average observed transaction values - see details.
`x`	the number of transactions the customer made in the calibration period. May also be a vector of frequencies - see details.

Details

m.x and x may be vectors. The standard rules for vector operations apply - if they are not of the same length, the shorter vector will be recycled (start over at the first element) until it is as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of log-likelihoods.

Value

The log-likelihood of the gamma-gamma model. If m.x or x has a length greater than 1, this is a vector of log-likelihoods.

References

Fader, Peter S., Bruce G.S. Hardie, and Ka L. Lee. "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." Journal of Marketing Research Vol.42, pp.415-430. November. 2005. Web.

Examples

```
## Not run:
data(cdnowSummary)
ave.spend <- cdnowSummary$m.x
tot.trans <- cdnowSummary$cbs[,"x"]
# params <- c(6.25, 3.74, 15.44) # in original documentation. check below:
params <- spend.EstimateParameters(m.x.vector = ave.spend, x.vector = tot.trans)
# get the total log-likelihood of the data and parameters
# above. There will be many warnings due to the zeroes that are
# included in the data. If you wish to avoid these warnings, use:

# ave.spend <- ave.spend[which(tot.trans > 0)]
# tot.trans <- tot.trans[which(tot.trans > 0)]

# Note that we used tot.trans to remove the zeroes from ave.spend.
# This is because we need the vectors to be the same length, and it
# is possible that your data include customers who made transactions
# worth zero dollars (in which case the vector lengths would differ
# if we used ave.spend to remove the zeroes from ave.spend).

sum(spend.LL(params, ave.spend, tot.trans))

# This log-likelihood may be different than mentioned in the
# referenced paper; in the paper, a slightly different function
# which relies on total spend (not average spend) is used.

## End(Not run)
```

spend.marginal.likelihood

Gamma-gamma marginal likelihood

Description

Calculates the marginal likelihood of a customer's average transaction value.

Usage

```
spend.marginal.likelihood(params, m.x, x)
```

Arguments

`params`	a vector of gamma-gamma parameters: p, q, and gamma, in that order. p is the shape parameter for each transaction. The scale parameter for each transaction is distributed across customers according to a gamma distribution with parameters q (shape) and gamma (scale).
`m.x`	the customer's average observed transaction value in the calibration period. May also be a vector of average observed transaction values - see details.
`x`	the number of transactions the customer made in the calibration period. May also be a vector of frequencies - see details.

Details

m.x and x may be vectors. The standard rules for vector operations apply - if they are not of the same length, the shorter vector will be recycled (start over at the first element) until it is as long as the longest vector. It is advisable to keep vectors to the same length and to use single values for parameters that are to be the same for all calculations. If one of these parameters has a length greater than one, the output will be a vector of probabilities.

This function will issue a warning if any of m.x or x is 0, and will return a marginal likelihood of 0 for those values.

f(m.x | p, q, gamma, x).

Value

The marginal distribution of a customer's average transaction value. If m.x or x has a length greater than 1, a vector of marginal likelihoods will be returned.

References

Fader, Peter S., Bruce G.S. Hardie, and Ka L. Lee. "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." Journal of Marketing Research Vol.42, pp.415-430. November. 2005. Web.

See equation 3.

Examples

```
params <- c(6, 4, 16)

# calculate the marginal distribution of the average transaction value
# of a customer who spent an average of $35 over 3 transactions.
spend.marginal.likelihood(params, m.x=35, x=3)

# Several values can also be computed at once:
spend.marginal.likelihood(params, m.x=30:40, x=3)
spend.marginal.likelihood(params, m.x=35, x=1:10)
spend.marginal.likelihood(params, m.x=30:40, x=1:11)
```

`spend.plot.average.transaction.value`
Plot Actual vs. Expected Average Transaction Value

Description

Plots the actual and expected densities of average transaction values, and returns a vector with each customer's average transaction value probability.

Usage

```
spend.plot.average.transaction.value(
  params,
  m.x.vector,
  x.vector,
  xlab = "Average Transaction Value",
  ylab = "Marginal Distribution of Average Transaction Value",
  title = "Actual vs. Expected Average Transaction Value Across Customers"
)
```

Arguments

`params`	a vector of gamma-gamma parameters: p, q, and gamma, in that order. p is the shape parameter for each transaction. The scale parameter for each transaction is distributed across customers according to a gamma distribution with parameters q (shape) and gamma (scale).
`m.x.vector`	a vector with each customer's average observed transaction value in the calibration period.
`x.vector`	a vector with the number of transactions each customer made in the calibration period. Must correspond to m.x.vector in terms of ordering of customers and length of the vector.
`xlab`	descriptive label for the x axis.
`ylab`	descriptive label for the y axis.
`title`	title placed on the top-center of the plot.

Value

a vector with the probability of each customer’s average transaction value.

See Also

`spend.marginal.likelihood`

Examples

```
## Not run:
data(cdnowSummary)
ave.spend <- cdnowSummary$m.x
tot.trans <- cdnowSummary$cbs[,"x"]
# params <- c(6.25, 3.74, 15.44) # in original documentation. check below:
params <- spend.EstimateParameters(m.x.vector = ave.spend, x.vector = tot.trans)

# Plot the actual and expected average transaction value across customers.
f.m.x <- spend.plot.average.transaction.value(params, ave.spend, tot.trans)

## End(Not run)
```

`subLogs` *Subtract Logs*

Description

Given log(a) and log(b), returns log(a - b)

Usage

```
subLogs(loga, logb)
```

Arguments

`loga`	first number in log space.
`logb`	first number in log space.

Index

∗ **datasets**
  cdnowElog, 60
  cdnowSummary, 61
  discreteSimElog, 88
  donationsSummary, 89

addLogs, 6

bgbb.ConditionalExpectedTransactions, *5*, 7, *13*
bgbb.DERT, 8, *34*
bgbb.EstimateParameters, *4*, 9
bgbb.Expectation, *5*, 11
bgbb.HeatmapHoldoutExpectedTrans, 12
bgbb.LL, *5*, 13, *35*, *74*
bgbb.PAlive, 14
bgbb.PlotDropoutRateHeterogeneity, 16
bgbb.PlotFrequencyInCalibration, *5*, 16
bgbb.PlotFrequencyInHoldout, 18
bgbb.PlotFreqVsConditionalExpectedFrequency, 19
bgbb.PlotRecVsConditionalExpectedFrequency, 21
bgbb.PlotTrackingCum, 23
bgbb.PlotTrackingInc, 25
bgbb.PlotTransactionRateHeterogeneity, 26
bgbb.pmf, *5*, 27, *29*
bgbb.pmf.General, *28*, 28
bgbb.PosteriorMeanDropoutRate, 30, *36*
bgbb.PosteriorMeanLmProductMoment, *30*, 31, *33*, *36*
bgbb.PosteriorMeanTransactionRate, 32, *38*
bgbb.rf.matrix.DERT, 33
bgbb.rf.matrix.LL, *10*, 35, *80*, *83*
bgbb.rf.matrix.PosteriorMeanDropoutRate, *31*, 36
bgbb.rf.matrix.PosteriorMeanTransactionRate, *33*, 37
bgnbd.cbs.LL, 38, *39*, *42*, *46*, *80*, *83*
bgnbd.ConditionalExpectedTransactions, *5*, 39, *43*, *45*, *46*
bgnbd.EstimateParameters, *4*, *39*, 41, *46*
bgnbd.Expectation, *5*, *40*, 43, *45*
bgnbd.ExpectedCumulativeTransactions, 44, *55*, *56*
bgnbd.generalParams, 45
bgnbd.LL, *5*, *39*, *46*, 46, *74*
bgnbd.PAlive, *46*, 47
bgnbd.PlotDropoutRateHeterogeneity, 48
bgnbd.PlotFrequencyInCalibration, *5*, 49
bgnbd.PlotFreqVsConditionalExpectedFrequency, 50
bgnbd.PlotRecVsConditionalExpectedFrequency, 52
bgnbd.PlotTrackingCum, 54
bgnbd.PlotTrackingInc, 55
bgnbd.PlotTransactionRateHeterogeneity, 57
bgnbd.pmf, *5*, 58
bgnbd.pmf.General, 59
BTYD *(*BTYD-package*)*, 4
BTYD-package, 4

cdnowElog, 60, *61*
cdnowSummary, 61

dc.BuildCBSFromCBTAndDates, 62, *78*
dc.BuildCBTFromElog, 63
dc.check.model.params, 64, *74*
dc.compress.cbs, *38*, 65, *91*, *95*
dc.CreateFreqCBT, 66
dc.CreateReachCBT, 67
dc.CreateSpendCBT, 68
dc.CumulativeToIncremental, 69
dc.DissipateElog, 69
dc.ElogToCbsCbt, *4*, 70
dc.FilterCustByBirth, 71
dc.GetFirstPurchasePeriodsFromCBT, 72

dc.GetLastPurchasePeriodsFromCBT, 73
dc.InputCheck, 73, *120*
dc.MakeRFmatrixCal, *4*, 74, *88*
dc.MakeRFmatrixHoldout, *4*, 76
dc.MakeRFmatrixSkeleton, 77
dc.MergeCustomers, 77
dc.MergeTransactionsOnSameDate, 79
dc.PlotLogLikelihoodContour, 80, *84*
dc.PlotLogLikelihoodContours, *81*, 83
dc.ReadLines, *4*, 85
dc.RemoveTimeBetween, 86
dc.SplitUpElogForRepeatTrans, 87
dc.WriteLine, 88
discreteSimElog, 88
donationsSummary, 89

h2f1, *40*, *43*, *45*, 90, *91*, *92*, *94*, *96*, *99–101*, *103*, *105*, *107*, *109*, *115*, *116*
hypergeo, *40*, *43*, *45*, *90–92*, *94*, *96*, *99–101*, *103*, *105*, *107*, *109*, *115*, *116*

optim, *96*
optimx, *41*, *96*

pnbd.cbs.LL, *39*, *80*, *83*, 91, *95*, *96*
pnbd.ConditionalExpectedTransactions, *5*, *74*, 92, *97*, *107*, *109*
pnbd.DERT, 93, *100*
pnbd.EstimateParameters, *4*, *61*, *91*, 95, *101*
pnbd.Expectation, *5*, *93*, 97, *99*
pnbd.ExpectedCumulativeTransactions, 98, *111*, *113*
pnbd.generalParams, 99
pnbd.LL, *5*, *73*, *74*, *90*, *91*, *96*, *100*, 100
pnbd.PAlive, *92*, *100*, 101
pnbd.Plot.DERT, 102
pnbd.PlotDropoutRateHeterogeneity, 104, *108*
pnbd.PlotFrequencyInCalibration, *5*, 105
pnbd.PlotFreqVsConditionalExpectedFrequency, 106
pnbd.PlotRateHeterogeneity, 108
pnbd.PlotRecVsConditionalExpectedFrequency, 109
pnbd.PlotTrackingCum, 110
pnbd.PlotTrackingInc, 112
pnbd.PlotTransactionRateHeterogeneity, *108*, 114
pnbd.pmf, *5*, *105*, 115
pnbd.pmf.General, *115*, 116

spend.EstimateParameters, 117
spend.expected.value, 119, *121*
spend.generalParams, 120
spend.LL, *74*, *118*, *121*, 121
spend.marginal.likelihood, *121*, 123, *125*
spend.plot.average.transaction.value, 124
subLogs, 125