Package ‘CLVTools’

January 20, 2025

Title Tools for Customer Lifetime Value Estimation

Version 0.11.2

Date 2024-12-01

Depends R (>= 3.5.0), methods

Description A set of state-of-the-art probabilistic modeling approaches to derive estimates of individual customer lifetime values (CLV).
Commonly, probabilistic approaches focus on modelling 3 processes, i.e. individuals' attrition, transaction, and spending process.
Latent customer attrition models, which are also known as ``buy-'til-you-die models'', model the attrition as well as the transaction process.
They are used to make inferences and predictions about transactional patterns of individual customers such as their future purchase behavior.
Moreover, these models have also been used to predict individuals’ long-term engagement in activities such as playing an online game or
posting to a social media platform. The spending process is usually modelled by a separate probabilistic model. Combining these results yields in
lifetime values estimates for individual customers.
This package includes fast and accurate implementations of various probabilistic models for non-contractual settings
(e.g., grocery purchases or hotel visits). All implementations support time-invariant covariates, which can be used to control for e.g.,
socio-demographics. If such an extension has been proposed in literature, we further provide the possibility to control for time-varying
covariates to control for e.g., seasonal patterns.
Currently, the package includes the following latent attrition models to model individuals' attrition and transaction process:
[1] Pareto/NBD model (Pareto/Negative-Binomial-Distribution),
[2] the Extended Pareto/NBD model (Pareto/Negative-Binomial-Distribution with time-varying covariates),
[3] the BG/NBD model (Beta-Gamma/Negative-Binomial-Distribution) and the
[4] GGom/NBD (Gamma-Gompertz/Negative-Binomial-Distribution).
Further, we provide an implementation of the Gamma/Gamma model to model the spending process of individuals.

Imports data.table (>= 1.12.0), digest (>= 0.6.0), Formula (>= 1.2-4),
ggplot2 (>= 3.2.0), lubridate (>= 1.7.8), numDeriv (>=

2016.8-1.1), Matrix (>= 1.2-17), MASS, optimx (>= 2019-12.02), Rcpp(>= 0.12.12), stats, utils

Suggests covr, knitr, rmarkdown, xml2, testthat (>= 3.0.0), lmtest

License GPL-3

URL https://github.com/bachmannpatrick/CLVTools

BugReports https://github.com/bachmannpatrick/CLVTools/issues

NeedsCompilation yes

LinkingTo Rcpp, RcppArmadillo (>= 0.11.4.0.1), RcppGSL (>= 0.3.7), testthat

SystemRequirements GNU GSL

LazyLoad yes

Encoding UTF-8

Collate 'CLVTools.R' 'RcppExports.R' 'all_generics.R' 'catch-routine-registration.R' 'class_clv_time.R' 'class_clv_data.R' 'class_clv_model.R' 'class_clv_fitted.R' 'class_clv_fitted_transactions.R' 'class_clv_model_nocorrelation.R' 'class_clv_model_bgnbd.R' 'class_clv_bgnbd.R' 'class_clv_fitted_transactions_staticcov.R' 'class_clv_data_staticcovariates.R' 'class_clv_model_bgnbd_staticcov.R' 'class_clv_bgnbd_staticcov.R' 'class_clv_data_dynamiccovariates.R' 'class_clv_fitted_spending.R' 'class_clv_fitted_transactions_dynamiccov.R' 'class_clv_model_gg.R' 'class_clv_gg.R' 'class_clv_model_ggomnbd_nocov.R' 'class_clv_ggomnbd.R' 'class_clv_model_ggomnbd_staticcov.R' 'class_clv_ggomnbd_staticcov.R' 'class_clv_model_withcorrelation.R' 'class_clv_model_pnbd.R' 'class_clv_model_pnbd_staticcov.R' 'class_clv_model_pnbd_dynamiccov.R' 'class_clv_pnbd.R' 'class_clv_pnbd_dynamiccov.R' 'class_clv_pnbd_staticcov.R' 'class_clv_time_date.R' 'class_clv_time_datetime.R' 'class_clv_time_days.R' 'class_clv_time_hours.R' 'class_clv_time_weeks.R' 'class_clv_time_years.R' 'clv_template_controlflow_estimate.R' 'clv_template_controlflow_pmf.R' 'clv_template_controlflow_predict.R' 'data.R' 'f_DoExpectation.R' 'f_clvdata_inputchecks.R' 'f_clvfitted_inputchecks.R' 'f_generics_clvdata.R' 'f_generics_clvdatadyncov.R' 'f_generics_clvdatastaticcov.R' 'f_generics_clvfitted.R' 'f_generics_clvfitted_estimate.R' 'f_generics_clvfittedspending.R' 'f_generics_clvfittedtransactions.R' 'f_generics_clvfittedtransactionsdyncov.R'

'f_generics_clvfittedtransactionsstaticcov.R'
'f_generics_clvfittedtransactionsstaticcov_estimate.R'
'f_generics_clvpnbddyncov.R' 'f_interface_bgbb.R'
'f_interface_bgnbd.R' 'f_interface_bootstrappedapply.R'
'f_interface_clvdata.R' 'f_interface_gg.R'
'f_interface_ggomnbd.R' 'f_interface_latentattrition.R'
'f_interface_lrtest.R' 'f_interface_newcustomer.R'
'f_interface_pmf.R' 'f_interface_pnbd.R'
'f_interface_predict_clvfittedspending.R'
'f_interface_predict_clvfittedtransactions.R'
'f_interface_setdynamiccovariates.R'
'f_interface_setstaticcovariates.R' 'f_interface_spending.R'
'f_s3generics_clvdata.R' 'f_s3generics_clvdata_dynamiccov.R'
'f_s3generics_clvdata_plot.R'
'f_s3generics_clvdata_staticcov.R' 'f_s3generics_clvfitted.R'
'f_s3generics_clvfittedspending_plot.R'
'f_s3generics_clvfittedtransactions_plot.R'
'f_s3generics_clvfittedtransactions_staticcov.R'
'f_s3generics_clvtime.R' 'interlayer_callLL.R'
'interlayer_callnextinterlayer.R' 'interlayer_constraints.R'
'interlayer_correlation.R' 'interlayer_manager.R'
'interlayer_regularization.R' 'pnbd_dyncov_ABCD.R'
'pnbd_dyncov_BkSum.R' 'pnbd_dyncov_CET.R' 'pnbd_dyncov_DECT.R'
'pnbd_dyncov_createwalks.R' 'pnbd_dyncov_expectation.R'
'pnbd_dyncov_palive.R'

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/parallel false

Config/testthat/edition 3

Author Patrick Bachmann [cre, aut],
Niels Kuebler [aut],
Markus Meierer [aut],
Jeffrey Naef [aut],
E. Shin Oblander [aut],
Patrik Schilter [aut]

Maintainer Patrick Bachmann <pbachma@ethz.ch>

Repository CRAN

Date/Publication 2024-12-02 22:40:02 UTC

Contents

CLVTools-package 5
apparelDynCov 6
apparelDynCovFuture 6
apparelStaticCov 7

apparelTrans . . . 7
as.clv.data . . . 8
as.data.frame.clv.data . . . 9
as.data.table.clv.data . . . 11
bgbb . . . 12
bgnbd . . . 14
bgnbd_CET . . . 18
bgnbd_expectation . . . 19
bgnbd_LL . . . 20
bgnbd_PAlive . . . 22
bgnbd_pmf . . . 24
cdnow . . . 25
clv.bootstrapped.apply . . . 25
clvdata . . . 27
fitted.clv.fitted . . . 29
gg . . . 31
ggomnbd . . . 33
ggomnbd_CET . . . 37
ggomnbd_expectation . . . 38
ggomnbd_LL . . . 39
ggomnbd_PAlive . . . 41
ggomnbd_PMF . . . 42
gg_LL . . . 44
latentAttrition . . . 45
lrtest . . . 47
newcustomer . . . 48
nobs.clv.data . . . 51
nobs.clv.fitted . . . 52
plot.clv.data . . . 52
plot.clv.fitted.spending . . . 56
plot.clv.fitted.transactions . . . 58
pmf . . . 62
pnbd . . . 63
pnbd_CET . . . 69
pnbd_DERT . . . 71
pnbd_expectation . . . 73
pnbd_LL . . . 74
pnbd_PAlive . . . 76
pnbd_pmf . . . 77
predict.clv.fitted.spending . . . 78
predict.clv.fitted.transactions . . . 81
SetDynamicCovariates . . . 86
SetStaticCovariates . . . 87
spending . . . 89
subset.clv.data . . . 90
summary.clv.fitted . . . 92
vcov.clv.fitted . . . 94

Index **96**

CLVTools-package *Customer Lifetime Value Tools*

Description

CLVTools is a toolbox for various probabilistic customer attrition models for non-contractual settings. It provides a framework, which is capable of unifying different probabilistic customer attrition models. This package provides tools to estimate the number of future transactions of individual customers as well as the probability of customers being alive in future periods. Further, the average spending by customers can be estimated. Multiplying the future transactions conditional on being alive and the predicted individual spending per transaction results in an individual CLV value.

The implemented models require transactional data from non-contractual businesses (i.e. customers' purchase history).

Author(s)

Maintainer: Patrick Bachmann <pbachma@ethz.ch>

Authors:

- Niels Kuebler <niels.kuebler@uzh.ch>
- Markus Meierer <markus.meierer@business.uzh.ch>
- Jeffrey Naef <naef@stat.math.ethz.ch>
- E. Shin Oblander <eoblander23@gsb.columbia.edu>
- Patrik Schilter <patrik.schilter@gmail.com>

See Also

Development for CLVTools can be followed via the GitHub repository at https://github.com/bachmannpatrick/CLVTools.

Examples

```
data("cdnow")

# Create a CLV data object, split data in estimation and holdout sample
clv.data.cdnow <- clvdata(data.transactions = cdnow, date.format = "ymd",
                          time.unit = "week", estimation.split = 39, name.id = "Id")

# summary of data
summary(clv.data.cdnow)

# Fit a PNBD model without covariates on the first 39 periods
pnbd.cdnow <- pnbd(clv.data.cdnow,
                   start.params.model = c(r=0.5, alpha=8, s=0.5, beta=10))
# inspect fit
```

```
summary(pnbd.cdnow)

# Predict 10 periods (weeks) ahead from estimation end
#   and compare to actuals in this period
pred.out <- predict(pnbd.cdnow, prediction.end = 10)

# Plot the fitted model to the actual repeat transactions
plot(pnbd.cdnow)
```

---

apparelDynCov *Time-varying Covariates for the Apparel Retailer Dataset*

Description

This simulated data contains seasonal information and additional covariates on all 600 customers in the "apparelTrans" dataset. This information can be used as time-varying covariates.

Usage

```
data("apparelDynCov")
```

Format

A data.table with 187,800 rows and 5 variables

Id Customer Id

Cov.Date Date of contextual factor

High.Season Seasonal variable: 1 indicating a time-period that is considered "high season".

Gender 0=male, 1=female

Channel Acquisition channel: 0=online, 1=offline

---

apparelDynCovFuture *Future Time-varying Covariates for the Apparel Retailer Dataset*

Description

This simulated data contains seasonal information and additional covariates on all 600 customers in the "apparelTrans" after the last transaction in the dataset. This information can be used as time-varying covariates for prediction future customer behavior.

Usage

```
data("apparelDynCovFuture")
```

Format

A data.table with 56,400 rows and 5 variables

Id Customer Id

Cov.Date Date of contextual factor

High.Season Seasonal variable: 1 indicating a time-period that is considered "high season".

Gender 0=male, 1=female

Channel Acquisition channel: 0=online, 1=offline

`apparelStaticCov` *Time-invariant Covariates for the Apparel Retailer Dataset*

Description

This simulated data contains additional demographic information on all 600 customers in the "apparelTrans" dataset. This information can be used as time-invariant covariates.

Usage

```
data("apparelStaticCov")
```

Format

A `data.table` with 600 rows and 3 variables:

Id Customer Id

Gender 0=male, 1=female

Channel Acquisition channel: 0=online, 1=offline

`apparelTrans` *Apparel Retailer Dataset*

Description

This is a simulated dataset containing the entire purchase history of customers made their first purchase at an apparel retailer on January 2nd 2005. In total the dataset contains 600 customers who made 3,187 transactions between January 2005 and end of December 2010.

Usage

```
data("apparelTrans")
```

Format

A `data.table` with 3,187 rows and 3 variables:

`Id` Customer Id

`Date` Date of purchase

`Price` Price of purchase

`as.clv.data` *Coerce to clv.data object*

Description

Functions to coerce transaction data to a `clv.data` object.

Usage

```
as.clv.data(
  x,
  date.format = "ymd",
  time.unit = "weeks",
  estimation.split = NULL,
  name.id = "Id",
  name.date = "Date",
  name.price = "Price",
  ...
)

## S3 method for class 'data.frame'
as.clv.data(
  x,
  date.format = "ymd",
  time.unit = "weeks",
  estimation.split = NULL,
  name.id = "Id",
  name.date = "Date",
  name.price = "Price",
  ...
)

## S3 method for class 'data.table'
as.clv.data(
  x,
  date.format = "ymd",
  time.unit = "weeks",
  estimation.split = NULL,
  name.id = "Id",
```

```
    name.date = "Date",
    name.price = "Price",
    ...
  )
```

Arguments

`x`	Transaction data.
`date.format`	Character string that indicates the format of the date variable in the data used. See details.
`time.unit`	What time unit defines a period. May be abbreviated, capitalization is ignored. See details.
`estimation.split`	Indicates the length of the estimation period. See details.
`name.id`	Column name of the customer id in `x`.
`name.date`	Column name of the transaction date in `x`.
`name.price`	Column name of price in `x`. NULL if no spending data is present.
`...`	Ignored

Details

See section "Details" of clvdata for more details on parameters and usage.

Examples

```
 # dont test because ncpu=2 limit on cran (too fast)
data(cdnow)

# Turn data.table of transaction data into a clv.data object,
#  using default date format and column names but no holdout period
clv.cdnow <- as.clv.data(cdnow)
```

---

as.data.frame.clv.data *Coerce to a Data Frame*

Description

Extract a copy of the transaction data stored in the given clv.data object into a data.frame.

Usage

```
## S3 method for class 'clv.data'
as.data.frame(
  x,
  row.names = NULL,
  optional = NULL,
  ids = NULL,
  sample = c("full", "estimation", "holdout"),
  ...
)
```

Arguments

`x`	An object of class `clv.data`.
`row.names`	Ignored
`optional`	Ignored
`ids`	Character vector of customer ids for which transactions should be extracted. `NULL` extracts all.
`sample`	Name of sample for which transactions should be extracted, either "estimation", "holdout", or "full" (default).
`...`	Ignored

Value

A `data.frame` with columns `Id`, `Date`, and `Price` (if present).

Examples

```
data("cdnow")
clv.data.cdnow <- clvdata(data.transactions = cdnow,
                          date.format="ymd",
                          time.unit = "w",
                          estimation.split = 37)

# Extract all transaction data (all ids, estimation and holdout period)
df.trans <- as.data.frame(clv.data.cdnow)

# Extract transaction data of estimation period
df.trans <- as.data.frame(clv.data.cdnow, sample="estimation")

# Extract transaction data of ids "1", "2", and "999"
#  (estimation and holdout period)
df.trans <- as.data.frame(clv.data.cdnow, ids = c("1", "2", "999"))

# Extract transaction data of ids "1", "2", and "999" in estimation period
df.trans <- as.data.frame(clv.data.cdnow, ids = c("1", "2", "999"),
                          sample="estimation")
```

as.data.table.clv.data *Coerce to a Data Table*

Description

Extract a copy of the transaction data stored in the given clv.data object into a data.table.

Usage

```
## S3 method for class 'clv.data'
as.data.table(
  x,
  keep.rownames = FALSE,
  ids = NULL,
  sample = c("full", "estimation", "holdout"),
  ...
)
```

Arguments

x	An object of class `clv.data`.
keep.rownames	Ignored
ids	Character vector of customer ids for which transactions should be extracted. `NULL` extracts all.
sample	Name of sample for which transactions should be extracted, either "estimation", "holdout", or "full" (default).
...	Ignored

Value

A `data.table` with columns `Id`, `Date`, and `Price` (if present).

Examples

```
library(data.table)

data("cdnow")
clv.data.cdnow <- clvdata(data.transactions = cdnow,
                          date.format="ymd",
                          time.unit = "w",
                          estimation.split = 37)

# Extract all transaction data (all ids, estimation and holdout period)
dt.trans <- as.data.table(clv.data.cdnow)

# Extract transaction data of estimation period
```

```
dt.trans <- as.data.table(clv.data.cdnow, sample="estimation")

# Extract transaction data of ids "1", "2", and "999"
#  (estimation and holdout period)
dt.trans <- as.data.table(clv.data.cdnow, ids = c("1", "2", "999"))

# Extract transaction data of ids "1", "2", and "999" in estimation period
dt.trans <- as.data.table(clv.data.cdnow, ids = c("1", "2", "999"),
                          sample="estimation")
```

---

bgbb *BG/BB models - Work In Progress*

Description

Fits BG/BB models on transactional data with static and without covariates. Not yet implemented.

Usage

```
## S4 method for signature 'clv.data'
bgbb(
  clv.data,
  start.params.model = c(),
  optimx.args = list(),
  verbose = TRUE,
  ...
)

## S4 method for signature 'clv.data.static.covariates'
bgbb(
  clv.data,
  start.params.model = c(),
  optimx.args = list(),
  verbose = TRUE,
  names.cov.life = c(),
  names.cov.trans = c(),
  start.params.life = c(),
  start.params.trans = c(),
  names.cov.constr = c(),
  start.params.constr = c(),
  reg.lambdas = c(),
  ...
)

## S4 method for signature 'clv.data.dynamic.covariates'
bgbb(
```

```
  clv.data,
  start.params.model = c(),
  optimx.args = list(),
  verbose = TRUE,
  names.cov.life = c(),
  names.cov.trans = c(),
  start.params.life = c(),
  start.params.trans = c(),
  names.cov.constr = c(),
  start.params.constr = c(),
  reg.lambdas = c(),
  ...
)
```

Arguments

`clv.data` The data object on which the model is fitted.

`start.params.model` Named start parameters containing the optimization start parameters for the model without covariates.

`optimx.args` Additional arguments to control the optimization which are forwarded to `optimx::optimx`. If multiple optimization methods are specified, only the result of the last method is further processed.

`verbose` Show details about the running of the function.

`...` Ignored

`names.cov.life` Which of the set Lifetime covariates should be used. Missing parameter indicates all covariates shall be used.

`names.cov.trans` Which of the set Transaction covariates should be used. Missing parameter indicates all covariates shall be used.

`start.params.life` Named start parameters containing the optimization start parameters for all lifetime covariates.

`start.params.trans` Named start parameters containing the optimization start parameters for all transaction covariates.

`names.cov.constr` Which covariates should be forced to use the same parameters for the lifetime and transaction process. The covariates need to be present as both, lifetime and transaction covariates.

`start.params.constr` Named start parameters containing the optimization start parameters for the constraint covariates.

`reg.lambdas` Named lambda parameters used for the L2 regularization of the lifetime and the transaction covariate parameters. Lambdas have to be >= 0.

Value

No value is returned.

---

bgnbd *BG/NBD models*

Description

Fits BG/NBD models on transactional data without and with static covariates.

Usage

```
## S4 method for signature 'clv.data'
bgnbd(
  clv.data,
  start.params.model = c(),
  optimx.args = list(),
  verbose = TRUE,
  ...
)

## S4 method for signature 'clv.data.static.covariates'
bgnbd(
  clv.data,
  start.params.model = c(),
  optimx.args = list(),
  verbose = TRUE,
  names.cov.life = c(),
  names.cov.trans = c(),
  start.params.life = c(),
  start.params.trans = c(),
  names.cov.constr = c(),
  start.params.constr = c(),
  reg.lambdas = c(),
  ...
)
```

Arguments

`clv.data` The data object on which the model is fitted.

`start.params.model` Named start parameters containing the optimization start parameters for the model without covariates.

`optimx.args` Additional arguments to control the optimization which are forwarded to `optimx::optimx`. If multiple optimization methods are specified, only the result of the last method is further processed.

verbose
: Show details about the running of the function.

...
: Ignored

names.cov.life
: Which of the set Lifetime covariates should be used. Missing parameter indicates all covariates shall be used.

names.cov.trans
: Which of the set Transaction covariates should be used. Missing parameter indicates all covariates shall be used.

start.params.life
: Named start parameters containing the optimization start parameters for all lifetime covariates.

start.params.trans
: Named start parameters containing the optimization start parameters for all transaction covariates.

names.cov.constr
: Which covariates should be forced to use the same parameters for the lifetime and transaction process. The covariates need to be present as both, lifetime and transaction covariates.

start.params.constr
: Named start parameters containing the optimization start parameters for the constraint covariates.

reg.lambdas
: Named lambda parameters used for the L2 regularization of the lifetime and the transaction covariate parameters. Lambdas have to be >= 0.

Details

Model parameters for the BG/NBD model are `r, alpha, a, and b`.
`r`: shape parameter of the Gamma distribution of the purchase process.
`alpha`: scale parameter of the Gamma distribution of the purchase process.
`a`: shape parameter of the Beta distribution of the dropout process.
`b`: shape parameter of the Beta distribution of the dropout process.

If no start parameters are given, r = 1, alpha = 3, a = 1, b = 3 is used. All model start parameters are required to be > 0. If no start values are given for the covariate parameters, 0.1 is used.

Note that the DERT expression has not been derived (yet) and it consequently is not possible to calculated values for DERT and CLV.

The BG/NBD model: The BG/NBD is an "easy" alternative to the Pareto/NBD model that is easier to implement. The BG/NBD model slight adapts the behavioral "story" associated with the Pareto/NBD model in order to simplify the implementation. The BG/NBD model uses a beta-geometric and exponential gamma mixture distributions to model customer behavior. The key difference to the Pareto/NBD model is that a customer can only churn right after a transaction. This simplifies computations significantly, however has the drawback that a customer cannot churn until he/she makes a transaction. The Pareto/NBD model assumes that a customer can churn at any time.

BG/NBD model with static covariates: The standard BG/NBD model captures heterogeneity was solely using Gamma distributions. However, often exogenous knowledge, such as for example customer demographics, is available. The supplementary knowledge may explain part of the heterogeneity among the customers and therefore increase the predictive accuracy of the model.

In addition, we can rely on these parameter estimates for inference, i.e. identify and quantify effects of contextual factors on the two underlying purchase and attrition processes. For technical details we refer to the technical note by Fader and Hardie (2007).

The likelihood function is the likelihood function associated with the basic model where alpha, a, and b are replaced with alpha = alpha0*exp(-g1z1), a = a_0*exp(g2z2), and b = b0*exp(g3z2) while r remains unchanged. Note that in the current implementation, we constrain the covariate parameters and data for the lifetime process to be equal (g2=g3 and z2=z3).

Value

Depending on the data object on which the model was fit, bgnbd returns either an object of class clv.bgnbd or clv.bgnbd.static.cov.

The function `summary` can be used to obtain and print a summary of the results. The generic accessor functions `coefficients`, `vcov`, `fitted`, `logLik`, `AIC`, `BIC`, and `nobs` are available.

References

Fader PS, Hardie BGS, Lee KL (2005). ""Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model" Marketing Science, 24(2), 275-284.

Fader PS, Hardie BGS (2013). "Overcoming the BG/NBD Model's #NUM! Error Problem" URL `http://brucehardie.com/notes/027/bgnbd_num_error.pdf`.

Fader PS, Hardie BGS (2007). "Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models." URL `http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf`.

Fader PS, Hardie BGS, Lee KL (2007). "Creating a Fit Histogram for the BG/NBD Model" URL `https://www.brucehardie.com/notes/014/bgnbd_fit_histogram.pdf`

See Also

`clvdata` to create a clv data object, `SetStaticCovariates` to add static covariates to an existing clv data object.

`gg` to fit customer's average spending per transaction with the `Gamma-Gamma` model

`predict` to predict expected transactions, probability of being alive, and customer lifetime value for every customer

`plot` to plot the unconditional expectation as predicted by the fitted model

`pmf` for the probability to make exactly x transactions in the estimation period, given by the probability mass function (PMF).

`newcustomer` to predict the expected number of transactions for an average new customer.

The generic functions `vcov`, `summary`, `fitted`.

Examples

```
data("apparelTrans")
clv.data.apparel <- clvdata(apparelTrans, date.format = "ymd",
                            time.unit = "w", estimation.split = 52)
```

```
# Fit standard bgnbd model
bgnbd(clv.data.apparel)

# Give initial guesses for the model parameters
bgnbd(clv.data.apparel,
     start.params.model = c(r=0.5, alpha=15, a = 2, b=5))


# pass additional parameters to the optimizer (optimx)
#    Use Nelder-Mead as optimization method and print
#    detailed information about the optimization process
apparel.bgnbd <- bgnbd(clv.data.apparel,
                     optimx.args = list(method="Nelder-Mead",
                                        control=list(trace=6)))

# estimated coefs
coef(apparel.bgnbd)

# summary of the fitted model
summary(apparel.bgnbd)

# predict CLV etc for holdout period
predict(apparel.bgnbd)

# predict CLV etc for the next 15 periods
predict(apparel.bgnbd, prediction.end = 15)


# To estimate the bgnbd model with static covariates,
#   add static covariates to the data
data("apparelStaticCov")
clv.data.static.cov <-
 SetStaticCovariates(clv.data.apparel,
                     data.cov.life = apparelStaticCov,
                     names.cov.life = c("Gender", "Channel"),
                     data.cov.trans = apparelStaticCov,
                     names.cov.trans = c("Gender", "Channel"))

# Fit bgnbd with static covariates
bgnbd(clv.data.static.cov)

# Give initial guesses for both covariate parameters
bgnbd(clv.data.static.cov, start.params.trans = c(Gender=0.75, Channel=0.7),
                   start.params.life  = c(Gender=0.5, Channel=0.5))

# Use regularization
bgnbd(clv.data.static.cov, reg.lambdas = c(trans = 5, life=5))

# Force the same coefficient to be used for both covariates
bgnbd(clv.data.static.cov, names.cov.constr = "Gender",
                   start.params.constr = c(Gender=0.5))

# Fit model only with the Channel covariate for life but
```

```
# keep all trans covariates as is
bgnbd(clv.data.static.cov, names.cov.life = c("Channel"))
```

bgnbd_CET	*BG/NBD: Conditional Expected Transactions*

Description

Calculates the expected number of transactions in a given time period based on a customer's past transaction behavior and the BG/NBD model parameters.

bgnbd_nocov_CET Conditional Expected Transactions without covariates

bgnbd_staticcov_CET Conditional Expected Transactions with static covariates

Usage

```
bgnbd_nocov_CET(r, alpha, a, b, dPeriods, vX, vT_x, vT_cal)

bgnbd_staticcov_CET(
  r,
  alpha,
  a,
  b,
  dPeriods,
  vX,
  vT_x,
  vT_cal,
  vCovParams_trans,
  vCovParams_life,
  mCov_trans,
  mCov_life
)
```

Arguments

r	shape parameter of the Gamma distribution of the purchase process
alpha	scale parameter of the Gamma distribution of the purchase process
a	shape parameter of the Beta distribution of the lifetime process
b	shape parameter of the Beta distribution of the lifetime process
dPeriods	number of periods to predict
vX	Frequency vector of length n counting the numbers of purchases.
vT_x	Recency vector of length n.
vT_cal	Vector of length n indicating the total number of periods of observation.

vCovParams_trans
: Vector of estimated parameters for the transaction covariates.

vCovParams_life
: Vector of estimated parameters for the lifetime covariates.

mCov_trans
: Matrix containing the covariates data affecting the transaction process. One column for each covariate.

mCov_life
: Matrix containing the covariates data affecting the lifetime process. One column for each covariate.

Details

mCov_trans is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to vCovParams_trans at the respective position.

mCov_life is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to vCovParams_life at the respective position.

Value

Returns a vector containing the conditional expected transactions for the existing customers in the BG/NBD model.

References

Fader PS, Hardie BGS, Lee KL (2005). ""Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model" Marketing Science, 24(2), 275-284.

Fader PS, Hardie BGS (2013). "Overcoming the BG/NBD Model's #NUM! Error Problem" URL http://brucehardie.com/notes/027/bgnbd_num_error.pdf.

Fader PS, Hardie BGS (2007). "Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models." URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS, Lee KL (2007). "Creating a Fit Histogram for the BG/NBD Model" URL https://www.brucehardie.com/notes/014/bgnbd_fit_histogram.pdf

---

bgnbd_expectation *BG/NBD: Unconditional Expectation*

Description

Computes the expected number of repeat transactions in the interval (0, vT_i] for a randomly selected customer, where 0 is defined as the point when the customer came alive.

Usage

```
bgnbd_nocov_expectation(r, alpha, a, b, vT_i)

bgnbd_staticcov_expectation(r, vAlpha_i, vA_i, vB_i, vT_i)
```

Arguments

r	shape parameter of the Gamma distribution of the purchase process
alpha	scale parameter of the Gamma distribution of the purchase process
a	shape parameter of the Beta distribution of the lifetime process
b	shape parameter of the Beta distribution of the lifetime process
vT_i	Number of periods since the customer came alive
vAlpha_i	Vector of individual parameters alpha
vA_i	Vector of individual parameters a
vB_i	Vector of individual parameters b

Value

Returns the expected transaction values according to the chosen model.

References

Fader PS, Hardie BGS, Lee KL (2005). ““Counting Your Customers” the Easy Way: An Alternative to the Pareto/NBD Model” Marketing Science, 24(2), 275-284.

Fader PS, Hardie BGS (2013). “Overcoming the BG/NBD Model’s #NUM! Error Problem” URL http://brucehardie.com/notes/027/bgnbd_num_error.pdf.

Fader PS, Hardie BGS (2007). “Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models.” URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS, Lee KL (2007). “Creating a Fit Histogram for the BG/NBD Model” URL https://www.brucehardie.com/notes/014/bgnbd_fit_histogram.pdf

---

bgnbd_LL *BG/NBD: Log-Likelihood functions*

Description

Calculates the Log-Likelihood values for the BG/NBD model with and without covariates.

The function `bgnbd_nocov_LL_ind` calculates the individual log-likelihood values for each customer for the given parameters.

The function `bgnbd_nocov_LL_sum` calculates the log-likelihood value summed across customers for the given parameters.

The function bgnbd_staticcov_LL_ind calculates the individual log-likelihood values for each customer for the given parameters and covariates.

The function bgnbd_staticcov_LL_sum calculates the individual log-likelihood values summed across customers.

Usage

```
bgnbd_nocov_LL_ind(vLogparams, vX, vT_x, vT_cal)

bgnbd_nocov_LL_sum(vLogparams, vX, vT_x, vT_cal, vN)

bgnbd_staticcov_LL_ind(vParams, vX, vT_x, vT_cal, mCov_life, mCov_trans)

bgnbd_staticcov_LL_sum(vParams, vX, vT_x, vT_cal, vN, mCov_life, mCov_trans)
```

Arguments

vLogparams	vector with the BG/NBD model parameters at log scale. See Details.
vX	Frequency vector of length n counting the numbers of purchases.
vT_x	Recency vector of length n.
vT_cal	Vector of length n indicating the total number of periods of observation.
vN	The value ("number of times observed") with which the LL value of this observation is multiplied before summing across customers.
vParams	vector with the parameters for the BG/NBD model at log scale and the static covariates at original scale. See Details.
mCov_life	Matrix containing the covariates data affecting the lifetime process. One column for each covariate.
mCov_trans	Matrix containing the covariates data affecting the transaction process. One column for each covariate.

Details

vLogparams is a vector with model parameters r, alpha_0, a, b at log-scale, in this order.

vParams is vector with the BG/NBD model parameters at log scale, followed by the parameters for the lifetime covariates at original scale and then followed by the parameters for the transaction covariates at original scale

mCov_trans is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to vLogparams at the respective position.

mCov_life is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to vLogparams at the respective position.

Value

Returns the respective Log-Likelihood value(s) for the BG/NBD model with or without covariates.

References

Fader PS, Hardie BGS, Lee KL (2005). "“Counting Your Customers” the Easy Way: An Alternative to the Pareto/NBD Model" Marketing Science, 24(2), 275-284.

Fader PS, Hardie BGS (2013). “Overcoming the BG/NBD Model's #NUM! Error Problem” URL http://brucehardie.com/notes/027/bgnbd_num_error.pdf.

Fader PS, Hardie BGS (2007). “Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models.” URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS, Lee KL (2007). “Creating a Fit Histogram for the BG/NBD Model” URL https://www.brucehardie.com/notes/014/bgnbd_fit_histogram.pdf

---

`bgnbd_PAlive` *BG/NBD: Probability of Being Alive*

Description

Calculates the probability of a customer being alive at the end of the calibration period, based on a customer's past transaction behavior and the BG/NBD model parameters.

`bgnbd_nocov_PAlive` P(alive) for the BG/NBD model without covariates

`bgnbd_staticcov_PAlive` P(alive) for the BG/NBD model with static covariates

Usage

```
bgnbd_nocov_PAlive(r, alpha, a, b, vX, vT_x, vT_cal)

bgnbd_staticcov_PAlive(
  r,
  alpha,
  a,
  b,
  vX,
  vT_x,
  vT_cal,
  vCovParams_trans,
  vCovParams_life,
  mCov_trans,
  mCov_life
)
```

Arguments

`r`	shape parameter of the Gamma distribution of the purchase process
`alpha`	scale parameter of the Gamma distribution of the purchase process
`a`	shape parameter of the Beta distribution of the lifetime process

b	shape parameter of the Beta distribution of the lifetime process
vX	Frequency vector of length n counting the numbers of purchases.
vT_x	Recency vector of length n.
vT_cal	Vector of length n indicating the total number of periods of observation.
vCovParams_trans	Vector of estimated parameters for the transaction covariates.
vCovParams_life	Vector of estimated parameters for the lifetime covariates.
mCov_trans	Matrix containing the covariates data affecting the transaction process. One column for each covariate.
mCov_life	Matrix containing the covariates data affecting the lifetime process. One column for each covariate.

Details

mCov_trans is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to vCovParams_trans at the respective position.

mCov_life is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to vCovParams_life at the respective position.

Value

Returns a vector with the PAlive for each customer.

References

Fader PS, Hardie BGS, Lee KL (2005). ""Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model" Marketing Science, 24(2), 275-284.

Fader PS, Hardie BGS (2013). "Overcoming the BG/NBD Model's #NUM! Error Problem" URL http://brucehardie.com/notes/027/bgnbd_num_error.pdf.

Fader PS, Hardie BGS (2007). "Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models." URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS, Lee KL (2007). "Creating a Fit Histogram for the BG/NBD Model" URL https://www.brucehardie.com/notes/014/bgnbd_fit_histogram.pdf

bgnbd_pmf *BG/NBD: Probability Mass Function (PMF)*

Description

Calculate P(X(t)=x), the probability that a randomly selected customer makes exactly x transactions in the interval (0, t].

Usage

```
bgnbd_nocov_PMF(r, alpha, a, b, x, vT_i)

bgnbd_staticcov_PMF(r, x, vAlpha_i, vA_i, vB_i, vT_i)
```

Arguments

r	shape parameter of the Gamma distribution of the purchase process
alpha	scale parameter of the Gamma distribution of the purchase process
a	shape parameter of the Beta distribution of the lifetime process
b	shape parameter of the Beta distribution of the lifetime process
x	The number of transactions to calculate the probability for (unsigned integer).
vT_i	Number of periods since the customer came alive.
vAlpha_i	Vector of individual parameters alpha
vA_i	Vector of individual parameters a
vB_i	Vector of individual parameters b

Value

Returns a vector of probabilities.

References

Fader PS, Hardie BGS, Lee KL (2005). "“Counting Your Customers” the Easy Way: An Alternative to the Pareto/NBD Model" Marketing Science, 24(2), 275-284.

Fader PS, Hardie BGS (2013). “Overcoming the BG/NBD Model's #NUM! Error Problem” URL http://brucehardie.com/notes/027/bgnbd_num_error.pdf.

Fader PS, Hardie BGS (2007). “Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models.” URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS, Lee KL (2007). “Creating a Fit Histogram for the BG/NBD Model” URL https://www.brucehardie.com/notes/014/bgnbd_fit_histogram.pdf

`cdnow` *CDNOW dataset*

Description

A dataset containing the entire purchase history up to the end of June 1998 of the cohort of 23,570 individuals who made their first-ever purchase at CDNOW in the first quarter of 1997.

Usage

```
data("cdnow")
```

Format

A `data.table` with 6696 rows and 4 variables:

`Id` Customer Id

`Date` Date of purchase

`CDs` Amount of CDs purchased

`Price` Price of purchase

References

Fader, Peter S. and Bruce G.,S. Hardie, (2001), "Forecasting Repeat Sales at CDNOW: A Case Study," Interfaces, 31 (May-June), Part 2 of 2, p94-107.

`clv.bootstrapped.apply` *Bootstrapping: Fit a model again on sampled data and apply method*

Description

Given a fitted model, sample new data from the `clv.data` stored in it and re-fit the model on it. Which customers are selected into the new data is determined by `fn.sample`. The model is fit on the new data with the same options with which it was originally fit, including `optimx.args`, `verbose` and start parameters. If required, any option can be changed by passing it as `...`. After the model is fit, `fn.boot.apply` is applied to it and the value it returns is collected in a list which is eventually returned.

The estimation and holdout periods are preserved exactly as in the original data. This is regardless of how the actually sampled transactions would define these periods. This way, each customer's model summary data (`cbs`) generated from the sampled data remains the same as on the original data. This makes sampling from the `clv.data` object equivalent to sampling directly from the model summary data.

Note that the Id of customers which are sampled more than once gains a suffix "_BOOTSTRAP_ID_<number>".

Usage

```
clv.bootstrapped.apply(object, num.boots, fn.boot.apply, fn.sample = NULL, ...)
```

Arguments

object	Fitted model
num.boots	number of times to sample data and re-fit the model
fn.boot.apply	Method to apply on each model estimated on the sampled data. See examples.
fn.sample	Method sampling customer ids for creating the bootstrapped data. Receives and returns a vector of ids (string). If NULL, ids are sampled with replacement until reaching original length. See examples.
...	Passed to the model estimation method. See examples.

Value

Returns a list containing the results of fn.boot.apply

See Also

For possible inputs to ... see pnbd, ggomnbd, bgnbd.

Internal methods clv.data.create.bootstrapping.data to create a clv.data object of given customer ids and clv.fitted.estimate.same.specification.on.new.data to estimate a model again on new data with its original specification.

Examples

```
data("cdnow")

clv.cdnow <- clvdata(data.transactions = cdnow, date.format="ymd",
                     time.unit = "weeks", estimation.split=37)

pnbd.cdnow <- pnbd(clv.cdnow)

# bootstrapped model coefs while sampling 50 percent
# of customers without replacement
clv.bootstrapped.apply(pnbd.cdnow, num.boots=5, fn.boot.apply=coef,
fn.sample=function(x){
sample(x, size = as.integer(0.5*length(x)), replace = FALSE)})

# sample customers with built-in standard logic and
# return predictions until end of holdout period in original
# data.
# prediction.end is not required because the bootstrapped
# data contains the same estimation and holdout periods
# as the original data, even if the transactions of the sampled
# customers .
clv.bootstrapped.apply(pnbd.cdnow, num.boots=5, fn.sample=NULL,
fn.boot.apply=function(x){predict(x)})
```

```
# return the fitted models
# forward additional arguments to the model fitting method
clv.bootstrapped.apply(pnbd.cdnow, num.boots=5, fn.sample=NULL,
fn.boot.apply=return,
# args for ..., forwarded to pnbd()
verbose=FALSE, optimx.args=list(method="Nelder-Mead"),
start.params.model=coef(pnbd.cdnow))
```

---

clvdata *Create an object for transactional data required to estimate CLV*

Description

Creates a data object that contains the prepared transaction data and that is used as input for model fitting. The transaction data may be split in an estimation and holdout sample if desired. The model then will only be fit on the estimation sample.

If covariates should be used when fitting a model, covariate data can be added to an object returned from this function.

Usage

```
clvdata(
  data.transactions,
  date.format,
  time.unit,
  estimation.split = NULL,
  name.id = "Id",
  name.date = "Date",
  name.price = "Price"
)
```

Arguments

`data.transactions`
: Transaction data as `data.frame` or `data.table`. See details.

`date.format`
: Character string that indicates the format of the date variable in the data used. See details.

`time.unit`
: What time unit defines a period. May be abbreviated, capitalization is ignored. See details.

`estimation.split`
: Indicates the length of the estimation period. See details.

`name.id`
: Column name of the customer id in `data.transactions`.

`name.date`
: Column name of the transaction date in `data.transactions`.

`name.price`
: Column name of price in `data.transactions`. NULL if no spending data is present.

Details

data.transactions A data.frame or data.table with customers' purchase history. Every transaction record consists of a purchase date and a customer id. Optionally, the price of the transaction may be included to also allow for prediction of future customer spending.

time.unit The definition of a single period. Currently available are "hours", "days", "weeks", and "years". May be abbreviated.

date.format A single format to use when parsing any date that is given as character input. This includes the dates given in data.transaction, estimation.split, or as an input to any other function at a later point, such as prediction.end in predict. The function parse_date_time of package lubridate is used to parse inputs and hence all formats it accepts in argument orders can be used. For example, a date of format "year-month-day" (i.e., "2010-06-17") is indicated with "ymd". Other combinations such as "dmy", "dym", "ymd HMS", or "HMS dmy" are possible as well.

estimation.split May be specified as either the number of periods since the first transaction or the timepoint (either as character, Date, or POSIXct) at which the estimation period ends. The indicated timepoint itself will be part of the estimation sample. If no value is provided or set to NULL, the whole dataset will used for fitting the model (no holdout sample).

Aggregation of Transactions:

Multiple transactions by the same customer that occur on the minimally representable temporal resolution are aggregated to a single transaction with their spending summed. For time units days and any other coarser Date-based time units (i.e. weeks, years), this means that transactions on the same day are combined. When using finer time units such as hours which are based on POSIXct, transactions on the same second are aggregated.

For the definition of repeat-purchases, combined transactions are viewed as a single transaction. Hence, repeat-transactions are determined from the aggregated transactions.

Value

An object of class clv.data. See the class definition clv.data for more details about the returned object.

The function summary can be used to obtain and print a summary of the data. The generic accessor function nobs is available to read out the number of customers.

See Also

SetStaticCovariates to add static covariates

SetDynamicCovariates for how to add dynamic covariates

plot to plot the repeat transactions

summary to summarize the transaction data

pnbd to fit Pareto/NBD models on a clv.data object

Examples

```
data("cdnow")
```

```
# create clv data object with weekly periods
#    and no splitting
clv.data.cdnow <- clvdata(data.transactions = cdnow,
                          date.format="ymd",
                          time.unit = "weeks")

# same but split after 37 periods
clv.data.cdnow <- clvdata(data.transactions = cdnow,
                          date.format="ymd",
                          time.unit = "w",
                          estimation.split = 37)

# same but estimation end on the 15th Oct 1997
clv.data.cdnow <- clvdata(data.transactions = cdnow,
                          date.format="ymd",
                          time.unit = "w",
                          estimation.split = "1997-10-15")

# summary of the transaction data
summary(clv.data.cdnow)

# plot the total number of transactions per period
plot(clv.data.cdnow)

## Not run:
# create data with the weekly periods defined to
#   start on Mondays

# set start of week to Monday
oldopts <- options("lubridate.week.start"=1)

# create clv.data while Monday is the beginning of the week
clv.data.cdnow <- clvdata(data.transactions = cdnow,
                          date.format="ymd",
                          time.unit = "weeks")

# Dynamic covariates now have to be supplied for every Monday

# set week start to what it was before
options(oldopts)

## End(Not run)
```

fitted.clv.fitted *Extract Unconditional Expectation*

Description

Extract the unconditional expectation (future transactions unconditional on being "alive") from a fitted clv model. This is the unconditional expectation data that is used when plotting the fitted model.

Usage

```
## S3 method for class 'clv.fitted'
fitted(object, prediction.end = NULL, verbose = FALSE, ...)
```

Arguments

`object`	A fitted clv model for which the unconditional expectation is desired.
`prediction.end`	Until what point in time to predict. This can be the number of periods (numeric) or a form of date/time object. See details.
`verbose`	Show details about the running of the function.
`...`	Ignored

Details

`prediction.end` indicates until when to predict or plot and can be given as either a point in time (of class `Date`, `POSIXct`, or `character`) or the number of periods. If `prediction.end` is of class character, the date/time format set when creating the data object is used for parsing. If `prediction.end` is the number of periods, the end of the fitting period serves as the reference point from which periods are counted. Only full periods may be specified. If `prediction.end` is omitted or NULL, it defaults to the end of the holdout period if present and to the end of the estimation period otherwise.

The first prediction period is defined to start right after the end of the estimation period. If for example weekly time units are used and the estimation period ends on Sunday 2019-01-01, then the first day of the first prediction period is Monday 2019-01-02. Each prediction period includes a total of 7 days and the first prediction period therefore will end on, and include, Sunday 2019-01-08. Subsequent prediction periods again start on Mondays and end on Sundays. If `prediction.end` indicates a timepoint on which to end, this timepoint is included in the prediction period.

Value

A `data.table` which contains the following columns:

`period.until`	The timepoint that marks the end (up until and including) of the period to which the data in this row refers.
`period.num`	The number of this period.
`expectation`	The value of the unconditional expectation for the period that ends on `period.until`.

See Also

`plot` to plot the unconditional expectation gg *Gamma/Gamma Spending model*

Description

Fits the Gamma-Gamma model on a given object of class clv.data to predict customers' mean spending per transaction.

Usage

```
## S4 method for signature 'clv.data'
gg(
  clv.data,
  start.params.model = c(),
  remove.first.transaction = TRUE,
  optimx.args = list(),
  verbose = TRUE,
  ...
)
```

Arguments

clv.data	The data object on which the model is fitted.
start.params.model	Named start parameters containing the optimization start parameters for the model without covariates.
remove.first.transaction	Whether customer's first transaction are removed. If TRUE all zero-repeaters are excluded from model fitting.
optimx.args	Additional arguments to control the optimization which are forwarded to optimx::optimx. If multiple optimization methods are specified, only the result of the last method is further processed.
verbose	Show details about the running of the function.
...	Ignored

Details

Model parameters for the G/G model are p, q, and gamma.
p: shape parameter of the Gamma distribution of the spending process.
q: shape parameter of the Gamma distribution to account for customer heterogeneity.
gamma: scale parameter of the Gamma distribution to account for customer heterogeneity.
If no start parameters are given, p=0.5, q=15, gamma=2 is used for all model parameters. All parameters are required to be > 0.

The Gamma-Gamma model cannot be estimated for data that contains negative prices. Customers with a mean spending of zero or a transaction count of zero are ignored during model fitting.

The G/G model: The G/G model allows to predict a value for future customer transactions. Usually, the G/G model is used in combination with a probabilistic model predicting customer transaction such as the Pareto/NBD or the BG/NBD model.

Value

An object of class clv.gg is returned.

The function `summary` can be used to obtain and print a summary of the results. The generic accessor functions `coefficients`, `vcov`, `fitted`, `logLik`, `AIC`, `BIC`, and `nobs` are available.

References

Colombo R, Jiang W (1999). "A stochastic RFM model." Journal of Interactive Marketing, 13(3), 2-12.

Fader PS, Hardie BG, Lee K (2005). "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." Journal of Marketing Research, 42(4), 415-430.

Fader PS, Hardie BG (2013). "The Gamma-Gamma Model of Monetary Value." URL http://www.brucehardie.com/notes/025/gamma_gamma.pdf.

See Also

`clvdata` to create a clv data object.

`plot` to plot diagnostics of the transaction data, incl. of spending.

`predict` to predict expected mean spending for every customer.

`plot` to plot the density of customer's mean transaction value compared to the model's prediction.

Examples

```
data("apparelTrans")
clv.data.apparel <- clvdata(apparelTrans, date.format = "ymd",
                            time.unit = "w", estimation.split = 52)

# Fit the gg model
gg(clv.data.apparel)

# Give initial guesses for the model parameters
gg(clv.data.apparel,
     start.params.model = c(p=0.5, q=15, gamma=2))


# pass additional parameters to the optimizer (optimx)
#    Use Nelder-Mead as optimization method and print
#    detailed information about the optimization process
apparel.gg <- gg(clv.data.apparel,
                     optimx.args = list(method="Nelder-Mead",
                                        control=list(trace=6)))

# estimated coefs
coef(apparel.gg)
```

```
# summary of the fitted model
summary(apparel.gg)

# Plot model vs empirical distribution
plot(apparel.gg)

# predict mean spending and compare against
#    actuals in the holdout period
predict(apparel.gg)
```

ggomnbd   *Gamma-Gompertz/NBD model*

Description

Fits Gamma-Gompertz/NBD models on transactional data with static and without covariates.

Usage

```
## S4 method for signature 'clv.data'
ggomnbd(
  clv.data,
  start.params.model = c(),
  optimx.args = list(),
  verbose = TRUE,
  ...
)

## S4 method for signature 'clv.data.static.covariates'
ggomnbd(
  clv.data,
  start.params.model = c(),
  optimx.args = list(),
  verbose = TRUE,
  names.cov.life = c(),
  names.cov.trans = c(),
  start.params.life = c(),
  start.params.trans = c(),
  names.cov.constr = c(),
  start.params.constr = c(),
  reg.lambdas = c(),
  ...
)
```

Arguments

- `clv.data` The data object on which the model is fitted.
- `start.params.model` Named start parameters containing the optimization start parameters for the model without covariates.
- `optimx.args` Additional arguments to control the optimization which are forwarded to `optimx::optimx`. If multiple optimization methods are specified, only the result of the last method is further processed.
- `verbose` Show details about the running of the function.
- `...` Ignored
- `names.cov.life` Which of the set Lifetime covariates should be used. Missing parameter indicates all covariates shall be used.
- `names.cov.trans` Which of the set Transaction covariates should be used. Missing parameter indicates all covariates shall be used.
- `start.params.life` Named start parameters containing the optimization start parameters for all lifetime covariates.
- `start.params.trans` Named start parameters containing the optimization start parameters for all transaction covariates.
- `names.cov.constr` Which covariates should be forced to use the same parameters for the lifetime and transaction process. The covariates need to be present as both, lifetime and transaction covariates.
- `start.params.constr` Named start parameters containing the optimization start parameters for the constraint covariates.
- `reg.lambdas` Named lambda parameters used for the L2 regularization of the lifetime and the transaction covariate parameters. Lambdas have to be >= 0.

Details

Model parameters for the GGompertz/NBD model are `r, alpha, beta, b` and `s`.
`r`: shape parameter of the Gamma distribution of the purchase process. The smaller `r`, the stronger the heterogeneity of the purchase process.
`alpha`: scale parameter of the Gamma distribution of the purchase process.
`beta`: scale parameter for the Gamma distribution for the lifetime process.
`b`: scale parameter of the Gompertz distribution (constant across customers).
`s`: shape parameter of the Gamma distribution for the lifetime process. The smaller `s`, the stronger the heterogeneity of customer lifetimes.

If no start parameters are given, r=0.5, alpha=2, b=0.1, s=1, beta=0.1 is used. All model start parameters are required to be > 0. If no start values are given for the covariate parameters, 0.1 is used.

Note that the DERT expression has not been derived (yet) and it consequently is not possible to calculated values for DERT and CLV.

The Gamma-Gompertz/NBD model: There are two key differences of the gamma/Gompertz/NBD (GGompertz/NBD) model compared to the relative to the well-known Pareto/NBD model: (i) its probability density function can exhibit a mode at zero or an interior mode, and (ii) it can be skewed to the right or to the left. Therefore, the GGompertz/NBD model is more flexible than the Pareto/NBD model. According to Bemmaor and Glady (2012) can indicate substantial differences in expected residual lifetimes compared to the Pareto/NBD. The GGompertz/NBD tends to be appropriate when firms are reputed and their offerings are differentiated.

Value

Depending on the data object on which the model was fit, `ggomnbd` returns either an object of class clv.ggomnbd or clv.ggomnbd.static.cov.

The function `summary` can be used to obtain and print a summary of the results. The generic accessor functions `coefficients`, `vcov`, `fitted`, `logLik`, `AIC`, `BIC`, and `nobs` are available.

References

Bemmaor AC, Glady N (2012). "Modeling Purchasing Behavior with Sudden "Death": A Flexible Customer Lifetime Model" Management Science, 58(5), 1012-1021.

Adler J (2022). "Comment on "Modeling Purchasing Behavior with Sudden "Death": A Flexible Customer Lifetime Model" Management Science 69(3):1929-1930.

The expression for the PMF was derived by Adler J (2024). (unpublished)

See Also

`clvdata` to create a clv data object, `SetStaticCovariates` to add static covariates to an existing clv data object.

`gg` to fit customer's average spending per transaction with the `Gamma-Gamma` model

`predict` to predict expected transactions, probability of being alive, and customer lifetime value for every customer

`plot` to plot the unconditional expectation as predicted by the fitted model

`pmf` for the probability to make exactly x transactions in the estimation period, given by the probability mass function (PMF).

`newcustomer` to predict the expected number of transactions for an average new customer.

The generic functions `vcov`, `summary`, `fitted`.

Examples

```
data("apparelTrans")
clv.data.apparel <- clvdata(apparelTrans, date.format = "ymd",
                            time.unit = "w", estimation.split = 52)

# Fit standard ggomnbd model
ggomnbd(clv.data.apparel)

# Give initial guesses for the model parameters
ggomnbd(clv.data.apparel,
```

```
    start.params.model = c(r=0.5, alpha=15, b=5, beta=10, s=0.5))


# pass additional parameters to the optimizer (optimx)
#    Use Nelder-Mead as optimization method and print
#    detailed information about the optimization process
apparel.ggomnbd <- ggomnbd(clv.data.apparel,
                     optimx.args = list(method="Nelder-Mead",
                                        control=list(trace=6)))

# estimated coefs
coef(apparel.ggomnbd)

# summary of the fitted model
summary(apparel.ggomnbd)

# predict CLV etc for holdout period
predict(apparel.ggomnbd)

# predict CLV etc for the next 15 periods
predict(apparel.ggomnbd, prediction.end = 15)


# To estimate the ggomnbd model with static covariates,
#   add static covariates to the data
data("apparelStaticCov")
clv.data.static.cov <-
 SetStaticCovariates(clv.data.apparel,
                     data.cov.life = apparelStaticCov,
                     names.cov.life = c("Gender", "Channel"),
                     data.cov.trans = apparelStaticCov,
                     names.cov.trans = c("Gender", "Channel"))

# Fit ggomnbd with static covariates
ggomnbd(clv.data.static.cov)

# Give initial guesses for both covariate parameters
ggomnbd(clv.data.static.cov, start.params.trans = c(Gender=0.75, Channel=0.7),
                   start.params.life  = c(Gender=0.5, Channel=0.5))

# Use regularization
ggomnbd(clv.data.static.cov, reg.lambdas = c(trans = 5, life=5))

# Force the same coefficient to be used for both covariates
ggomnbd(clv.data.static.cov, names.cov.constr = "Gender",
                   start.params.constr = c(Gender=0.5))

# Fit model only with the Channel covariate for life but
# keep all trans covariates as is
ggomnbd(clv.data.static.cov, names.cov.life = c("Channel"))
```

ggomnbd_CET *GGompertz/NBD: Conditional Expected Transactions*

Description

Calculates the expected number of transactions in a given time period based on a customer's past transaction behavior and the GGompertz/NBD model parameters.

`ggomnbd_nocov_CET` Conditional Expected Transactions without covariates

`ggomnbd_staticcov_CET` Conditional Expected Transactions with static covariates

Usage

```
ggomnbd_nocov_CET(r, alpha_0, b, s, beta_0, dPeriods, vX, vT_x, vT_cal)

ggomnbd_staticcov_CET(
  r,
  alpha_0,
  b,
  s,
  beta_0,
  dPeriods,
  vX,
  vT_x,
  vT_cal,
  vCovParams_trans,
  vCovParams_life,
  mCov_life,
  mCov_trans
)
```

Arguments

`r`	shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process.
`alpha_0`	scale parameter of the Gamma distribution of the purchase process.
`b`	scale parameter of the Gompertz distribution (constant across customers)
`s`	shape parameter of the Gamma distribution for the lifetime process The smaller s, the stronger the heterogeneity of customer lifetimes.
`beta_0`	scale parameter for the Gamma distribution for the lifetime process
`dPeriods`	number of periods to predict
`vX`	Frequency vector of length n counting the numbers of purchases.
`vT_x`	Recency vector of length n.
`vT_cal`	Vector of length n indicating the total number of periods of observation.

`vCovParams_trans`
: Vector of estimated parameters for the transaction covariates.

`vCovParams_life`
: Vector of estimated parameters for the lifetime covariates.

`mCov_life`
: Matrix containing the covariates data affecting the lifetime process. One column for each covariate.

`mCov_trans`
: Matrix containing the covariates data affecting the transaction process. One column for each covariate.

Details

`mCov_trans` is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vCovParams_trans` at the respective position.

`mCov_life` is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vCovParams_life` at the respective position.

Value

Returns a vector containing the conditional expected transactions for the existing customers in the GGompertz/NBD model.

References

Bemmaor AC, Glady N (2012). "Modeling Purchasing Behavior with Sudden "Death": A Flexible Customer Lifetime Model" Management Science, 58(5), 1012-1021.

Adler J (2022). "Comment on "Modeling Purchasing Behavior with Sudden "Death": A Flexible Customer Lifetime Model" Management Science 69(3):1929-1930.

The expression for the PMF was derived by Adler J (2024). (unpublished)

`ggomnbd_expectation` *GGompertz/NBD: Unconditional Expectation*

Description

Computes the expected number of repeat transactions in the interval (0, vT_i] for a randomly selected customer, where 0 is defined as the point when the customer came alive.

Usage

```
ggomnbd_nocov_expectation(r, alpha_0, b, s, beta_0, vT_i)

ggomnbd_staticcov_expectation(r, b, s, vAlpha_i, vBeta_i, vT_i)
```

Arguments

r	shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process.
alpha_0	scale parameter of the Gamma distribution of the purchase process.
b	scale parameter of the Gompertz distribution (constant across customers)
s	shape parameter of the Gamma distribution for the lifetime process The smaller s, the stronger the heterogeneity of customer lifetimes.
beta_0	scale parameter for the Gamma distribution for the lifetime process
vT_i	Number of periods since the customer came alive
vAlpha_i	Vector of individual parameters alpha
vBeta_i	Vector of individual parameters beta

Value

Returns the expected transaction values according to the chosen model.

References

Bemmaor AC, Glady N (2012). “Modeling Purchasing Behavior with Sudden “Death”: A Flexible Customer Lifetime Model” Management Science, 58(5), 1012-1021.

Adler J (2022). “Comment on “Modeling Purchasing Behavior with Sudden “Death”: A Flexible Customer Lifetime Model” Management Science 69(3):1929-1930.

The expression for the PMF was derived by Adler J (2024). (unpublished)

---

ggomnbd_LL *GGompertz/NBD: Log-Likelihood functions*

Description

Calculates the Log-Likelihood values for the GGompertz/NBD model with and without covariates.

The function `ggomnbd_nocov_LL_ind` calculates the individual log-likelihood values for each customer for the given parameters.

The function `ggomnbd_nocov_LL_sum` calculates the log-likelihood value summed across customers for the given parameters.

The function `ggomnbd_staticcov_LL_ind` calculates the individual log-likelihood values for each customer for the given parameters and covariates.

The function `ggomnbd_staticcov_LL_sum` calculates the individual log-likelihood values summed across customers.

Usage

```
ggomnbd_nocov_LL_ind(vLogparams, vX, vT_x, vT_cal)

ggomnbd_nocov_LL_sum(vLogparams, vX, vT_x, vT_cal, vN)

ggomnbd_staticcov_LL_ind(vParams, vX, vT_x, vT_cal, mCov_life, mCov_trans)

ggomnbd_staticcov_LL_sum(vParams, vX, vT_x, vT_cal, vN, mCov_life, mCov_trans)
```

Arguments

`vLogparams`	vector with the GGompertz/NBD model parameters at log scale. See Details.
`vX`	Frequency vector of length n counting the numbers of purchases.
`vT_x`	Recency vector of length n.
`vT_cal`	Vector of length n indicating the total number of periods of observation.
`vN`	The value ("number of times observed") with which the LL value of this observation is multiplied before summing across customers.
`vParams`	vector with the parameters for the GGompertz/NBD model at log scale and the static covariates at original scale. See Details.
`mCov_life`	Matrix containing the covariates data affecting the lifetime process. One column for each covariate.
`mCov_trans`	Matrix containing the covariates data affecting the transaction process. One column for each covariate.

Details

`vLogparams` is a vector with model parameters `r`, `alpha_0`, `b`, `s`, `beta_0` at log-scale, in this order.

`vParams` is vector with the GGompertz/NBD model parameters at log scale, followed by the parameters for the lifetime covariates at original scale and then followed by the parameters for the transaction covariates at original scale

`mCov_trans` is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vParams` at the respective position.

`mCov_life` is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vParams` at the respective position.

Value

Returns the respective Log-Likelihood value(s) for the GGompertz/NBD model with or without covariates.

References

Bemmaor AC, Glady N (2012). “Modeling Purchasing Behavior with Sudden “Death”: A Flexible Customer Lifetime Model” Management Science, 58(5), 1012-1021.

Adler J (2022). “Comment on “Modeling Purchasing Behavior with Sudden “Death”: A Flexible Customer Lifetime Model” Management Science 69(3):1929-1930.

The expression for the PMF was derived by Adler J (2024). (unpublished)

`ggomnbd_PAlive` *GGompertz/NBD: Probability of Being Alive*

Description

Calculates the probability of a customer being alive at the end of the calibration period, based on a customer's past transaction behavior and the GGompertz/NBD model parameters.

`ggomnbd_nocov_PAlive` P(alive) for the GGompertz/NBD model without covariates

`ggomnbd_staticcov_PAlive` P(alive) for the GGompertz/NBD model with static covariates

Usage

```
ggomnbd_staticcov_PAlive(
  r,
  alpha_0,
  b,
  s,
  beta_0,
  vX,
  vT_x,
  vT_cal,
  vCovParams_trans,
  vCovParams_life,
  mCov_life,
  mCov_trans
)

ggomnbd_nocov_PAlive(r, alpha_0, b, s, beta_0, vX, vT_x, vT_cal)
```

Arguments

`r`	shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process.
`alpha_0`	scale parameter of the Gamma distribution of the purchase process.
`b`	scale parameter of the Gompertz distribution (constant across customers)
`s`	shape parameter of the Gamma distribution for the lifetime process The smaller s, the stronger the heterogeneity of customer lifetimes.

beta_0	scale parameter for the Gamma distribution for the lifetime process
vX	Frequency vector of length n counting the numbers of purchases.
vT_x	Recency vector of length n.
vT_cal	Vector of length n indicating the total number of periods of observation.
vCovParams_trans	Vector of estimated parameters for the transaction covariates.
vCovParams_life	Vector of estimated parameters for the lifetime covariates.
mCov_life	Matrix containing the covariates data affecting the lifetime process. One column for each covariate.
mCov_trans	Matrix containing the covariates data affecting the transaction process. One column for each covariate.

Details

mCov_trans is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to vCovParams_trans at the respective position.

mCov_life is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to vCovParams_life at the respective position.

Value

Returns a vector with the PAlive for each customer.

References

Bemmaor AC, Glady N (2012). "Modeling Purchasing Behavior with Sudden "Death": A Flexible Customer Lifetime Model" Management Science, 58(5), 1012-1021.

Adler J (2022). "Comment on "Modeling Purchasing Behavior with Sudden "Death": A Flexible Customer Lifetime Model" Management Science 69(3):1929-1930.

The expression for the PMF was derived by Adler J (2024). (unpublished)

---

ggomnbd_PMF *GGompertz/NBD: Probability Mass Function (PMF)*

Description

Calculate P(X(t)=x), the probability that a randomly selected customer makes exactly x transactions in the interval (0, t].

Usage

```
ggomnbd_nocov_PMF(r, alpha_0, b, s, beta_0, x, vT_i)

ggomnbd_staticcov_PMF(
  r,
  alpha_0,
  b,
  s,
  beta_0,
  x,
  vCovParams_trans,
  vCovParams_life,
  mCov_life,
  mCov_trans,
  vT_i
)
```

Arguments

`r`	shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process.
`alpha_0`	scale parameter of the Gamma distribution of the purchase process.
`b`	scale parameter of the Gompertz distribution (constant across customers)
`s`	shape parameter of the Gamma distribution for the lifetime process The smaller s, the stronger the heterogeneity of customer lifetimes.
`beta_0`	scale parameter for the Gamma distribution for the lifetime process
`x`	The number of transactions to calculate the probability for (unsigned integer).
`vT_i`	Number of periods since the customer came alive.
`vCovParams_trans`	Vector of estimated parameters for the transaction covariates.
`vCovParams_life`	Vector of estimated parameters for the lifetime covariates.
`mCov_life`	Matrix containing the covariates data affecting the lifetime process. One column for each covariate.
`mCov_trans`	Matrix containing the covariates data affecting the transaction process. One column for each covariate.

Details

`mCov_trans` is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vCovParams_trans` at the respective position.

`mCov_life` is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vCovParams_life` at the respective position.

Value

Returns a vector of probabilities.

References

Bemmaor AC, Glady N (2012). “Modeling Purchasing Behavior with Sudden “Death”: A Flexible Customer Lifetime Model” Management Science, 58(5), 1012-1021.

Adler J (2022). “Comment on “Modeling Purchasing Behavior with Sudden “Death”: A Flexible Customer Lifetime Model” Management Science 69(3):1929-1930.

The expression for the PMF was derived by Adler J (2024). (unpublished)

---

gg_LL *Gamma-Gamma: Log-Likelihood Function*

Description

Calculates the Log-Likelihood value for the Gamma-Gamma model.

Usage

```
gg_LL(vLogparams, vX, vM_x, vN)
```

Arguments

`vLogparams`	a vector containing the log of the parameters p, q, gamma
`vX`	frequency vector of length n counting the numbers of purchases
`vM_x`	the observed average spending for every customer during the calibration time.
`vN`	The value ("number of times observed") with which the LL value of this observation is multiplied before summing across customers.

Details

`vLogparams` is a vector with the parameters for the Gamma-Gamma model. It has three parameters (p, q, gamma). The scale parameter for each transaction is distributed across customers according to a gamma distribution with parameters q (shape) and gamma (scale).

Value

Returns the Log-Likelihood value for the Gamma-Gamma model.

References

Colombo R, Jiang W (1999). "A stochastic RFM model." Journal of Interactive Marketing, 13(3), 2-12.

Fader PS, Hardie BG, Lee K (2005). "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." Journal of Marketing Research, 42(4), 415-430.

Fader PS, Hardie BG (2013). "The Gamma-Gamma Model of Monetary Value." URL http://www.brucehardie.com/notes/025/gamma_gamma.pdf.

`latentAttrition` *Formula Interface for Latent Attrition Models*

Description

Fit latent attrition models for transaction behavior, using a formula to specify the covariates.

Usage

```
latentAttrition(
  formula,
  family,
  data,
  optimx.args = list(),
  verbose = TRUE,
  ...
)
```

Arguments

`formula`	Formula to select and transform covariates in `data`. Has to be left empty if `data` contains no covariates. See Details.
`family`	A latentAttrition model. One of `pnbd`, `bgnbd`, or `ggomnbd`.
`data`	A `clv.data` object.
`optimx.args`	Additional arguments to control the optimization which are forwarded to `optimx::optimx`. If multiple optimization methods are specified, only the result of the last method is further processed.
`verbose`	Show details about the running of the function.
`...`	Forwarded to model specified in `family`.

Details

A two-part formula is used to select and transform the covariates stored in `data` before the model is estimated on it. May not be given if `data` contains no covariates.

The formula left hand side (LHS) has to remain empty and may never be specified.

The formula right hand side (RHS) follows a two-part notation using `|` as separator.

- 1st part: Which covariates to include for the lifetime process, potentially transforming them and adding interactions. The dot ('.') refers to all lifetime covariates.
- 2nd part: Which covariates to include for the transaction process, potentially transforming them and adding interactions. The dot ('.') refers to all transaction covariates

e.g: ~ `covlife | covtrans`

See the example section for illustrations on how to specify the formula parameter.

See Also

Models for inputs to family: pnbd, ggomnbd, bgnbd.

spending to fit spending models with a formula interface

Examples

```
data("apparelTrans")
data("apparelStaticCov")

clv.nocov <-
    clvdata(apparelTrans, time.unit="w", date.format="ymd")

# Create static covariate data with 2 covariates
clv.staticcov  <-
  SetStaticCovariates(clv.nocov,
                      data.cov.life  = apparelStaticCov,
                      names.cov.life = c("Gender", "Channel"),
                      data.cov.trans = apparelStaticCov,
                      names.cov.trans = c("Gender", "Channel"))

# Fit models without covariates.
# Note that NO formula may be specified in this case
latentAttrition(formula =, family=pnbd, data=clv.nocov)
latentAttrition(formula =, family=bgnbd, data=clv.nocov)
latentAttrition(formula =, family=ggomnbd, data=clv.nocov)

# Fit pnbd with start parameters and correlation
# required args are passed as part of '...'
latentAttrition(formula =, family=pnbd, data=clv.nocov,
                use.cor=TRUE,
                start.params.model=c(r=1, alpha=10, s=2, beta=8))


# Fit pnbd with all present covariates
latentAttrition(formula=~.|., family=pnbd, data=clv.staticcov)

# Fit pnbd with selected covariates
latentAttrition(formula=~Gender|Channel+Gender, family=pnbd,
                data=clv.staticcov)

# Fit pnbd with start parameters for covariates
```

```
latentAttrition(formula=~Gender|., family=pnbd,
                data=clv.staticcov,
                start.params.life = c(Gender = 0.6),
                start.params.trans = c(Gender = 0.6, Channel = 0.4))

# Fit pnbd with transformed covariate data
latentAttrition(formula=~Gender|I(log(Channel+2)), family=pnbd,
                data=clv.staticcov)

# Fit pnbd with all covs and regularization
latentAttrition(formula=~.|., family=pnbd, data=clv.staticcov,
                reg.lambdas = c(life=3, trans=8))

# Fit pnbd with all covs and constraint parameters for Channel
latentAttrition(formula=~.|., family=pnbd, data=clv.staticcov,
                names.cov.constr='Channel')
```

---

lrtest *Likelihood Ratio Test of Nested Models*

Description

`lrtest` carries out likelihood ratio tests to compare nested CLV models of the same family that were fitted on the same transaction data.

The method compares each two consecutive models. An asymptotic likelihood ratio test is carried out: Twice the difference in log-likelihoods is compared with a Chi-squared distribution.

Usage

```
## S3 method for class 'clv.fitted'
lrtest(object, ..., name = NULL)

lrtest(object, ...)

## S4 method for signature 'clv.fitted'
lrtest(object, ..., name = NULL)
```

Arguments

`object`	An fitted model object inheriting from `clv.fitted`.
`...`	Other models objects fitted on the same transaction data
`name`	A character vector of names to use for the models in the resulting output. If given, a name has to be provided for `object` and each model in `...`. If not given, the default model names are used.

Value

A data.frame of class "anova" which contains the log-likelihood, degrees of freedom, the difference in degrees of freedom, likelihood ratio Chi-squared statistic and corresponding p-value.

---

newcustomer   *New customer prediction data*

Description

The methods documented here are to be used together with predict (transactions) to obtain the expected number of transactions of an average newly alive customer and with predict (spending) to obtain the expected spending of an average newly alive customer. This prediction is only sensible for (fictional) customers without order history: Customers which just came alive and have not had the chance to reveal any more of their behavior.

The methods described here produce the data required as input to predict(newdata=) to make this new customer prediction. This is mostly covariate data for static and dynamic covariate models. See details for the required format.

newcustomer(), newcustomer.static(), newcustomer.dynamic(): To predict the number of transactions a single, fictional, average new customer is expected to make in the num.periods periods since making the first transaction ("coming alive").

newcustomer.spending(): To estimate how much a single, fictional, average new customer is expected to spend on average per transaction.

Usage

```
newcustomer(num.periods)

newcustomer.static(num.periods, data.cov.life, data.cov.trans)

newcustomer.dynamic(
  num.periods,
  data.cov.life,
  data.cov.trans,
  first.transaction
)

newcustomer.spending()
```

Arguments

num.periods	A positive, numeric scalar indicating the number of periods to predict.
data.cov.life	Numeric-only covariate data for the lifetime process for a single customer, data.table or data.frame. See details.
data.cov.trans	Numeric-only covariate data for the transaction process for a single customer, data.table or data.frame. See details.

`first.transaction`
: For dynamic covariate models only: The time point of the first transaction of the customer ("coming alive") for which a prediction is made. Has to be within the time range of the covariate data.

Details

The covariate data has to contain one column for every covariate parameter in the fitted model. Only numeric values are allowed, no factors or characters. No customer Id is required because the data on which the model was fit is not used for this prediction.

For `newcustomer.static()`: One column for every covariate parameter in the estimated model. No column `Id`. Exactly 1 row of numeric covariate data.
For example: `data.frame(Gender=1, Age=30, Channel=0)`.

For `newcustomer.dynamic()`: One column for every covariate parameter in the estimated model. No column `Id`. A column `Cov.Date` with time points that mark the start of the period defined by `time.unit`. For every `Cov.Date`, exactly 1 row of numeric covariate data.
For example for weekly covariates: `data.frame(Cov.Date=c("2000-01-03", "2000-01-10"), Gender=c(1,1), Channel=c(1, 1), High.Season=c(0,1,0))`
If `Cov.Date` is of type character, the `date.format` given when creating the the `clv.data` object is used to parse it. The data has to cover the time from the customer's first transaction `first.transaction` to the end of the prediction period given by `t`. It does not have to cover the same time range as when fitting the model. See examples.

For models with dynamic covariates, the time point of the first purchase (`first.transaction`) is additionally required because the exact covariates that are active during the prediction period have to be known.

Value

`newcustomer()`
: An object of class `clv.newcustomer.no.cov`

`newcustomer.static()`
: An object of class `clv.newcustomer.static.cov`

`newcustomer.dynamic()`
: An object of class `clv.newcustomer.dynamic.cov`

`newcustomer.spending()`
: An object of class `clv.newcustomer.spending`

See Also

predict (transactions) to use the output of the methods described here.

predict (spending) to use the output of the methods described here.

Examples

```
data("apparelTrans")
data("apparelStaticCov")
data("apparelDynCov")

clv.data.apparel <- clvdata(apparelTrans, date.format = "ymd",
```

```
                            time.unit = "w", estimation.split = 52)
clv.data.static.cov <-
 SetStaticCovariates(clv.data.apparel,
                    data.cov.life = apparelStaticCov,
                    names.cov.life = "Gender",
                    data.cov.trans = apparelStaticCov,
                    names.cov.trans = c("Gender", "Channel"))
clv.data.dyn.cov <-
  SetDynamicCovariates(clv.data = clv.data.apparel,
                      data.cov.life = apparelDynCov,
                      data.cov.trans = apparelDynCov,
                      names.cov.life = c("High.Season", "Gender"),
                      names.cov.trans = c("High.Season", "Gender"),
                      name.date = "Cov.Date")



# No covariate model
p.apparel <- pnbd(clv.data.apparel)

# Predict the number of transactions an average new
# customer is expected to make in the first 3.68 weeks
predict(
  p.apparel,
  newdata=newcustomer(num.periods=3.68)
)


# Spending model
gg.apparel <- gg(clv.data.apparel)
predict(gg.apparel, newdata = newcustomer.spending())



# Static covariate model
p.apparel.static <- pnbd(clv.data.static.cov)

# Predict the number of transactions an average new
# customer who is female (Gender=1) and who was acquired
# online (Channel=1) is expected to make in the first 3.68 weeks
predict(
  p.apparel.static,
  newdata=newcustomer.static(
    num.periods=3.68,
    # For the lifetime process, only Gender was used when fitting
    data.cov.life=data.frame(Gender=1),
    data.cov.trans=data.frame(Gender=1, Channel=0)
  )
)


## Not run:
# Dynamic covariate model
```

```
p.apparel.dyn <- pnbd(clv.data.dyn.cov)

# Predict the number of transactions an average new
# customer who is male (Gender=0), who did not purchase during
# high.season, and who was
# acquired on "2005-02-16" (first.transaction) is expected
# to make in the first 2.12 weeks.
# Note that the time range is very different from the one used
# when fitting the model. Cov.Date still has to match the
# beginning of the week.
predict(
  p.apparel.dyn,
  newdata=newcustomer.dynamic(
    num.periods=2.12,
    data.cov.life=data.frame(
      Cov.Date=c("2051-02-12", "2051-02-19", "2051-02-26"),
      Gender=c(0, 0, 0),
      High.Season=c(4, 0, 7)),
    data.cov.trans=data.frame(
      Cov.Date=c("2051-02-12", "2051-02-19", "2051-02-26"),
      Gender=c(0, 0, 0),
      High.Season=c(4, 0, 7)),
    first.transaction = "2051-02-16"
  )
)

## End(Not run)
```

`nobs.clv.data` *Number of observations*

Description

The number of observations is defined as the number of unique customers in the transaction data.

Usage

```
## S3 method for class 'clv.data'
nobs(object, ...)
```

Arguments

`object`	An object of class clv.data.
`...`	Ignored

Value

The number of customers.

nobs.clv.fitted — *Number of observations*

Description

The number of observations is defined as the number of unique customers for which the model was fit.

Usage

```
## S3 method for class 'clv.fitted'
nobs(object, ...)
```

Arguments

`object`	An object of class clv.fitted.
`...`	Ignored

Value

The number of customers.

plot.clv.data — *Plot Diagnostics for the Transaction data in a clv.data Object*

Description

Depending on the value of parameter `which`, one of the following plots will be produced. Note that the `sample` parameter determines the period for which the selected plot is made (either estimation, holdout, or full).

Tracking Plot: Plot the aggregated repeat transactions per period over the given time-horizon (`prediction.end`). See Details for the definition of plotting periods.

Frequency Plot: Plot the distribution of transactions or repeat transactions per customer, after aggregating transactions of the same customer on a single time point. Note that if `trans.bins` is changed, `label.remaining` usually needs to be adapted as well.

Spending Plot: Plot the empirical density of either customer's average spending per transaction or the value of every transaction in the data, after aggregating transactions of the same customer on a single time point. Note that in all cases this includes all transactions and not only repeat-transactions.

Interpurchase Time Plot: Plot the empirical density of customer's mean time (in number of periods) between transactions, after aggregating transactions of the same customer on a single time point. Note that customers without repeat-transactions are removed.

Transaction Timing Plot: Plot the transaction timings of selected or sampled customers on their respective timelines.

Usage

```
## S3 method for class 'clv.data'
plot(
  x,
  which = c("tracking", "frequency", "spending", "interpurchasetime", "timings"),
  prediction.end = NULL,
  cumulative = FALSE,
  trans.bins = 0:9,
  count.repeat.trans = TRUE,
  count.remaining = TRUE,
  label.remaining = "10+",
  mean.spending = TRUE,
  annotate.ids = FALSE,
  ids = c(),
  sample = c("estimation", "full", "holdout"),
  geom = "line",
  color = "black",
  plot = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	The clv.data object to plot
`which`	Which plot to produce, either "tracking", "frequency", "spending", "interpurchasetime", or "timings". May be abbreviated but only one may be selected. Defaults to "tracking".
`prediction.end`	"tracking": Until what point in time to plot. This can be the number of periods (numeric) or a form of date/time object. See details.
`cumulative`	"tracking": Whether the cumulative actual repeat transactions should be plotted.
`trans.bins`	"frequency": Vector of integers indicating the number of transactions (x axis) for which the customers should be counted.
`count.repeat.trans`	"frequency": Whether repeat transactions (TRUE, default) or all transactions (FALSE) should be counted.
`count.remaining`	"frequency": Whether the customers which are not captured with `trans.bins` should be counted in a separate last bar.
`label.remaining`	"frequency": Label for the last bar, if `count.remaining=TRUE`.
`mean.spending`	"spending": Whether customer's mean spending per transaction (`TRUE`, default) or the value of every transaction in the data (`FALSE`) should be plotted.
`annotate.ids`	"timings": Whether timelines should be annotated with customer ids.
`ids`	"timings": A character vector of customer ids or a single integer specifying the number of customers to sample. Defaults to `NULL` for which 50 random customers are selected.

`sample`	Name of the sample for which the plot should be made, either "estimation", "full", or "holdout". Defaults to "estimation". Not for "tracking" and "timing".
`geom`	"spending" and "interpurchasetime": The geometric object of ggplot2 to display the data. Forwarded to ggplot2::stat_density.
`color`	Color of resulting geom object in the plot. Not for "tracking" and "timing".
`plot`	Whether a plot should be created or only the assembled data returned.
`verbose`	Show details about the running of the function.
`...`	Forwarded to ggplot2::stat_density ("spending", "interpurchasetime") or ggplot2::geom_bar ("frequency"). Not for "tracking" and "timings".

Details

`prediction.end` indicates until when to predict or plot and can be given as either a point in time (of class `Date`, `POSIXct`, or `character`) or the number of periods. If `prediction.end` is of class character, the date/time format set when creating the data object is used for parsing. If `prediction.end` is the number of periods, the end of the fitting period serves as the reference point from which periods are counted. Only full periods may be specified. If `prediction.end` is omitted or NULL, it defaults to the end of the holdout period if present and to the end of the estimation period otherwise.

The first prediction period is defined to start right after the end of the estimation period. If for example weekly time units are used and the estimation period ends on Sunday 2019-01-01, then the first day of the first prediction period is Monday 2019-01-02. Each prediction period includes a total of 7 days and the first prediction period therefore will end on, and include, Sunday 2019-01-08. Subsequent prediction periods again start on Mondays and end on Sundays. If `prediction.end` indicates a timepoint on which to end, this timepoint is included in the prediction period.

If there are no repeat transactions until `prediction.end`, only the time for which there is data is plotted. If the data is returned (i.e. with argument `plot=FALSE`), the respective rows contain `NA` in column `Number of Repeat Transactions`.

Value

An object of class `ggplot` from package `ggplot2` is returned by default. If `plot=FALSE`, the data that would have been used to create the plot is returned. Depending on which plot was selected, this is a `data.table` which contains some of the following columns:

`Id`	Customer Id
`period.until`	The timepoint that marks the end (up until and including) of the period to which the data in this row refers.
`Spending`	Spending as defined by parameter `mean.spending`.
`mean.interpurchase.time`	Mean number of periods between transactions per customer, excluding customers with no repeat-transactions.
`num.transactions`	The number of (repeat) transactions, depending on `count.repeat.trans`.
`num.customers`	The number of customers.
`type`	"timings": Which purpose the value in this row is used for.

variable	"tracking": The number of actual repeat transactions in the period that ends at period.until. "timings": Coordinate (x or y) for which to use the value in this row for.
value	"timings": Date or numeric (stored as string) "tracking": numeric

See Also

ggplot2::stat_density and ggplot2::geom_bar for possible arguments to ...

plot to plot fitted transaction models

plot to plot fitted spending models

Examples

```
data("cdnow")
clv.cdnow <- clvdata(cdnow, time.unit="w",estimation.split=37,
                     date.format="ymd")

### TRACKING PLOT
# Plot the actual repeat transactions
plot(clv.cdnow)
# same, explicitly
plot(clv.cdnow, which="tracking")

# plot cumulative repeat transactions
plot(clv.cdnow, cumulative=TRUE)

# Dont automatically plot but tweak further
library(ggplot2) # for ggtitle()
gg.cdnow <- plot(clv.cdnow)
# change Title
gg.cdnow + ggtitle("CDnow repeat transactions")

# Dont return a plot but only the data from
#   which it would have been created
dt.plot.data <- plot(clv.cdnow, plot=FALSE)


### FREQUENCY PLOT
plot(clv.cdnow, which="frequency")

# Bins from 0 to 15, all remaining in bin labelled "16+"
plot(clv.cdnow, which="frequency", trans.bins=0:15,
     label.remaining="16+")

# Count all transactions, not only repeat
#  Note that the bins have to be adapted to start from 1
plot(clv.cdnow, which="frequency", count.repeat.trans = FALSE,
     trans.bins=1:9)
```

```
### SPENDING DENSITY
# plot customer's average transaction value
plot(clv.cdnow, which="spending", mean.spending = TRUE)

# distribution of the values of every transaction
plot(clv.cdnow, which="spending", mean.spending = FALSE)


### INTERPURCHASE TIME DENSITY
# plot as small points, in blue
plot(clv.cdnow, which="interpurchasetime",
     geom="point", color="blue", size=0.02)


### TIMING PATTERNS
# selected customers and annotating them
plot(clv.cdnow, which="timings", ids=c("123", "1041"), annotate.ids=TRUE)

# plot 25 random customers
plot(clv.cdnow, which="timings", ids=25)

# plot all customers
plot(clv.cdnow, which="timings", ids=nobs(clv.cdnow))
```

---

plot.clv.fitted.spending *Plot expected and actual mean spending per transaction*

Description

Compares the density of the observed average spending per transaction (empirical distribution) to the model's distribution of mean transaction spending (weighted by the actual number of transactions). See `plot.clv.data` to plot more nuanced diagnostics for the transaction data only.

Usage

```
## S3 method for class 'clv.fitted.spending'
plot(x, n = 256, verbose = TRUE, ...)

## S4 method for signature 'clv.fitted.spending'
plot(x, n = 256, verbose = TRUE, ...)
```

Arguments

x	The fitted spending model to plot

n	Number of points at which the empirical and model density are calculated. Should be a power of two.
verbose	Show details about the running of the function.
...	Ignored

Value

An object of class `ggplot` from package `ggplot2` is returned by default.

References

Colombo R, Jiang W (1999). "A stochastic RFM model." Journal of Interactive Marketing, 13(3), 2-12.

Fader PS, Hardie BG, Lee K (2005). "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." Journal of Marketing Research, 42(4), 415-430.

Fader PS, Hardie BG (2013). "The Gamma-Gamma Model of Monetary Value." URL http://www.brucehardie.com/notes/025/gamma_gamma.pdf.

See Also

`plot` for transaction models

`plot` for transaction diagnostics of `clv.data` objects

Examples

```
data("cdnow")

clv.cdnow <- clvdata(cdnow,
  date.format="ymd",
  time.unit = "week",
  estimation.split = "1997-09-30")

est.gg <- gg(clv.data = clv.cdnow)

# Compare empirical to theoretical distribution
plot(est.gg)

## Not run:
# Modify the created plot further
library(ggplot2)
gg.cdnow <- plot(est.gg)
gg.cdnow + ggtitle("CDnow Spending Distribution")

## End(Not run)
```

plot.clv.fitted.transactions
Plot Diagnostics for a Fitted Transaction Model

Description

Depending on the value of parameter which, one of the following plots will be produced. See plot.clv.data to plot more nuanced diagnostics for the transaction data only. For comparison, other models can be drawn into the same plot by specifying them in other.models (see examples).

Tracking Plot: Plot the actual repeat transactions and overlay it with the repeat transaction as predicted by the fitted model. Currently, following previous literature, the in-sample unconditional expectation is plotted in the holdout period. In the future, we might add the option to also plot the summed CET for the holdout period as an alternative evaluation metric. Note that only whole periods can be plotted and that the prediction end might not exactly match prediction.end. See the Note section for more details.

PMF Plot: Plot the actual and expected number of customers which made a given number of repeat transaction in the estimation period. The expected number is based on the PMF of the fitted model, the probability to make exactly a given number of repeat transactions in the estimation period. For each bin, the expected number is the sum of all customers' individual PMF value. Note that if trans.bins is changed, label.remaining needs to be adapted as well.

Usage

```
## S3 method for class 'clv.fitted.transactions'
plot(
  x,
  which = c("tracking", "pmf"),
  other.models = list(),
  prediction.end = NULL,
  cumulative = FALSE,
  trans.bins = 0:9,
  calculate.remaining = TRUE,
  label.remaining = "10+",
  newdata = NULL,
  transactions = TRUE,
  label = NULL,
  plot = TRUE,
  verbose = TRUE,
  ...
)

## S4 method for signature 'clv.fitted.transactions'
plot(
  x,
  which = c("tracking", "pmf"),
```

```
  other.models = list(),
  prediction.end = NULL,
  cumulative = FALSE,
  trans.bins = 0:9,
  calculate.remaining = TRUE,
  label.remaining = "10+",
  newdata = NULL,
  transactions = TRUE,
  label = NULL,
  plot = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	The fitted transaction model for which to produce diagnostic plots
which	Which plot to produce, either "tracking" or "pmf". May be abbreviated but only one may be selected. Defaults to "tracking".
other.models	List of fitted transaction models to plot. List names are used as colors, standard colors are chosen if unnamed (see examples). The `clv.data` object stored in each model is used if no `newdata` is given.
prediction.end	"tracking": Until what point in time to plot. This can be the number of periods (numeric) or a form of date/time object. See details.
cumulative	"tracking": Whether the cumulative expected (and actual) transactions should be plotted.
trans.bins	"pmf": Vector of positive integer numbers (>=0) indicating the number of repeat transactions (x axis) to plot. Should contain 0 in nearly all cases as it refers to repeat-transactions.
calculate.remaining	"pmf": Whether the probability for the remaining number of transactions not in `trans.bins` should be calculated.
label.remaining	"pmf": Label for the last bar, if `calculate.remaining=TRUE`.
newdata	An object of class clv.data for which the plotting should be made with the fitted model. If none or NULL is given, the plot is made for the data on which the model was fit. If `other.models` was specified, the data in each model is replaced with `newdata`.
transactions	Whether the actual observed repeat transactions should be plotted.
label	Character vector to label each model. If NULL, the model(s) internal name is used (see examples).
plot	Whether a plot is created or only the assembled data is returned.
verbose	Show details about the running of the function.
...	Ignored

Details

prediction.end indicates until when to predict or plot and can be given as either a point in time (of class Date, POSIXct, or character) or the number of periods. If prediction.end is of class character, the date/time format set when creating the data object is used for parsing. If prediction.end is the number of periods, the end of the fitting period serves as the reference point from which periods are counted. Only full periods may be specified. If prediction.end is omitted or NULL, it defaults to the end of the holdout period if present and to the end of the estimation period otherwise.

The first prediction period is defined to start right after the end of the estimation period. If for example weekly time units are used and the estimation period ends on Sunday 2019-01-01, then the first day of the first prediction period is Monday 2019-01-02. Each prediction period includes a total of 7 days and the first prediction period therefore will end on, and include, Sunday 2019-01-08. Subsequent prediction periods again start on Mondays and end on Sundays. If prediction.end indicates a timepoint on which to end, this timepoint is included in the prediction period.

The newdata argument has to be a clv data object of the exact same class as the data object on which the model was fit. In case the model was fit with covariates, newdata needs to contain identically named covariate data.

The use case for newdata is mainly two-fold: First, to estimate model parameters only on a sample of the data and then use the fitted model object to predict or plot for the full data set provided through newdata. Second, for models with dynamic covariates, to provide a clv data object with longer covariates than contained in the data on which the model was estimated what allows to predict or plot further. When providing newdata, some models might require additional steps that can significantly increase runtime.

Value

An object of class ggplot from package ggplot2 is returned by default. If plot=FALSE, the data that would have been used to create the plot is returned. Depending on which plot was selected, this is a data.table which contains the following columns:

For the Tracking plot:

period.until
: The timepoint that marks the end (up until and including) of the period to which the data in this row refers.

variable
: Type of variable that 'value' refers to. Either "model name" or "Actual" (if transactions=TRUE).

value
: Depending on variable either (Actual) the actual number of repeat transactions in the period that ends at period.until, or the unconditional expectation for the period that ends on period.until ("model name").

For the PMF plot:

num.transactions
: The number of repeat transactions in the estimation period (as ordered factor).

variable
: Type of variable that 'value' refers to. Either "model name" or "Actual" (if transactions=TRUE).

value
: Depending on variable either (Actual) the actual number of customers which have the respective number of repeat transactions, or the number of customers which are expected to have the respective number of repeat transactions, as by the fitted model ("model name").

Note

Because the unconditional expectation for a period is derived as the difference of the cumulative expectations calculated at the beginning and at end of the period, all timepoints for which the expectation is calculated are required to be spaced exactly 1 time unit apart.

If `prediction.end` does not coincide with the start of a time unit, the last timepoint for which the expectation is calculated and plotted therefore is not `prediction.end` but the start of the first time unit after `prediction.end`.

See Also

`plot.clv.fitted.spending` for diagnostics of spending models

`plot.clv.data` for transaction diagnostics of `clv.data` objects

`pmf` for the values on which the PMF plot is based

Examples

```
data("cdnow")

# Fit ParetoNBD model on the CDnow data
clv.cdnow <- clvdata(cdnow, time.unit="w",
                           estimation.split=37,
                           date.format="ymd")
pnbd.cdnow <- pnbd(clv.cdnow)

## TRACKING PLOT
# Plot actual repeat transaction, overlayed with the
#  expected repeat transactions as by the fitted model
plot(pnbd.cdnow)

# Plot cumulative expected transactions of only the model
plot(pnbd.cdnow, cumulative=TRUE, transactions=FALSE)

# Plot until 2001-10-21
plot(pnbd.cdnow, prediction.end = "2001-10-21")

# Plot until 2001-10-21, as date
plot(pnbd.cdnow,
     prediction.end = lubridate::dym("21-2001-10"))

# Plot 15 time units after end of estimation period
plot(pnbd.cdnow, prediction.end = 15)

# Save the data generated for plotting
#   (period, actual transactions, expected transactions)
plot.out <- plot(pnbd.cdnow, prediction.end = 15)

# A ggplot object is returned that can be further tweaked
library("ggplot2")
gg.pnbd.cdnow <- plot(pnbd.cdnow)
```

```
gg.pnbd.cdnow + ggtitle("PNBD on CDnow")


## PMF PLOT
plot(pnbd.cdnow, which="pmf")

# For transactions 0 to 15, also have
#  to change label for remaining
plot(pnbd.cdnow, which="pmf", trans.bins=0:15,
     label.remaining="16+")

# For transactions 0 to 15 bins, no remaining
plot(pnbd.cdnow, which="pmf", trans.bins=0:15,
     calculate.remaining=FALSE)

## MODEL COMPARISON
# compare vs bgnbd
bgnbd.cdnow <- bgnbd(clv.cdnow)
ggomnbd.cdnow <- ggomnbd(clv.cdnow)

# specify colors as names of other.models
# note that ggomnbd collapses into the pnbd on this dataset
plot(pnbd.cdnow, cumulative=TRUE,
     other.models=list(blue=bgnbd.cdnow, "#00ff00"=ggomnbd.cdnow))

# specify names as label, using standard colors
plot(pnbd.cdnow, which="pmf",
     other.models=list(bgnbd.cdnow),
     label=c("Pareto/NBD", "BG/NBD"))
```

---

pmf *Probability Mass Function*

Description

Calculate P(X(t)=x), the probability to make exactly x repeat transactions in the interval (0, t]. This interval is in the estimation period and excludes values of `t=0`. Note that here `t` is defined as the observation period `T.cal` which differs by customer.

Usage

```
## S4 method for signature 'clv.fitted.transactions'
pmf(object, x = 0:5)
```

Arguments

`object`	The fitted transaction model.
`x`	Vector of positive integer numbers (>=0) indicating the number of repeat transactions x for which the PMF should be calculated.

Value

Returns a `data.table` with ids and depending on `x`, multiple columns of PMF values, each column for one value in `x`.

`Id`	customer identification
`pmf.x.Y`	PMF values for Y number of transactions

See Also

The model fitting functions `pnbd`,`bgnbd`, `ggomnbd`.

`plot` to visually compare the PMF values against actuals.

Examples

```
data("cdnow")

# Fit the ParetoNBD model on the CDnow data
pnbd.cdnow <- pnbd(clvdata(cdnow, time.unit="w",
                           estimation.split=37,
                           date.format="ymd"))

# Calculate the PMF for 0 to 10 transactions
#  in the estimation period
pmf(pnbd.cdnow, x=0:10)

# Compare vs. actuals (CBS in estimation period):
# x    mean(pmf)    actual percentage of x
# 0    0.616514     1432/2357= 0.6075519
# 1    0.168309     436/2357 = 0.1849809
# 2    0.080971     208/2357 = 0.0882478
# 3    0.046190     100/2357 = 0.0424268
# 4    0.028566     60/2357  = 0.0254561
# 5    0.018506     36/2357  = 0.0152737
# 6    0.012351     27/2357  = 0.0114552
# 7    0.008415     21/2357  = 0.0089096
# 8    0.005822     5/2357   = 0.0021213
# 9    0.004074     4/2357   = 0.0016971
# 10   0.002877     7/2357   = 0.0029699
```

---

pnbd *Pareto/NBD models*

Description

Fits Pareto/NBD models on transactional data with and without covariates.

Usage

```
## S4 method for signature 'clv.data'
pnbd(
  clv.data,
  start.params.model = c(),
  use.cor = FALSE,
  start.param.cor = c(),
  optimx.args = list(),
  verbose = TRUE,
  ...
)

## S4 method for signature 'clv.data.static.covariates'
pnbd(
  clv.data,
  start.params.model = c(),
  use.cor = FALSE,
  start.param.cor = c(),
  optimx.args = list(),
  verbose = TRUE,
  names.cov.life = c(),
  names.cov.trans = c(),
  start.params.life = c(),
  start.params.trans = c(),
  names.cov.constr = c(),
  start.params.constr = c(),
  reg.lambdas = c(),
  ...
)

## S4 method for signature 'clv.data.dynamic.covariates'
pnbd(
  clv.data,
  start.params.model = c(),
  use.cor = FALSE,
  start.param.cor = c(),
  optimx.args = list(),
  verbose = TRUE,
  names.cov.life = c(),
  names.cov.trans = c(),
  start.params.life = c(),
  start.params.trans = c(),
  names.cov.constr = c(),
  start.params.constr = c(),
  reg.lambdas = c(),
  ...
)
```

Arguments

- `clv.data` The data object on which the model is fitted.
- `start.params.model` Named start parameters containing the optimization start parameters for the model without covariates.
- `use.cor` Whether the correlation between the transaction and lifetime process should be estimated.
- `start.param.cor` Start parameter for the optimization of the correlation.
- `optimx.args` Additional arguments to control the optimization which are forwarded to `optimx::optimx`. If multiple optimization methods are specified, only the result of the last method is further processed.
- `verbose` Show details about the running of the function.
- `...` Ignored
- `names.cov.life` Which of the set Lifetime covariates should be used. Missing parameter indicates all covariates shall be used.
- `names.cov.trans` Which of the set Transaction covariates should be used. Missing parameter indicates all covariates shall be used.
- `start.params.life` Named start parameters containing the optimization start parameters for all lifetime covariates.
- `start.params.trans` Named start parameters containing the optimization start parameters for all transaction covariates.
- `names.cov.constr` Which covariates should be forced to use the same parameters for the lifetime and transaction process. The covariates need to be present as both, lifetime and transaction covariates.
- `start.params.constr` Named start parameters containing the optimization start parameters for the constraint covariates.
- `reg.lambdas` Named lambda parameters used for the L2 regularization of the lifetime and the transaction covariate parameters. Lambdas have to be >= 0.

Details

Model parameters for the Pareto/NBD model are `r, alpha, s, and beta`.
`s`: shape parameter of the Gamma distribution for the lifetime process. The smaller s, the stronger the heterogeneity of customer lifetimes.
`beta`: rate parameter for the Gamma distribution for the lifetime process.
`r`: shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process.
`alpha`: rate parameter of the Gamma distribution of the purchase process.

Based on these parameters, the average purchase rate while customers are active is r/alpha and the average dropout rate is s/beta.

Ideally, the starting parameters for r and s represent your best guess concerning the heterogeneity of customers in their buy and die rate. If covariates are included into the model additionally parameters for the covariates affecting the attrition and the purchase process are part of the model.

If no start parameters are given, r=0.5, alpha=15, s=0.5, beta=10 is used for all model parameters and 0.1 for covariate parameters. The model start parameters are required to be > 0.

The Pareto/NBD model: The Pareto/NBD is the first model addressing the issue of modeling customer purchases and attrition simultaneously for non-contractual settings. The model uses a Pareto distribution, a combination of an Exponential and a Gamma distribution, to explicitly model customers' (unobserved) attrition behavior in addition to customers' purchase process.
In general, the Pareto/NBD model consist of two parts. A first process models the purchase behavior of customers as long as the customers are active. A second process models customers' attrition. Customers live (and buy) for a certain unknown time until they become inactive and "die". Customer attrition is unobserved. Inactive customers may not be reactivated. For technical details we refer to the original paper by Schmittlein, Morrison and Colombo (1987) and the detailed technical note of Fader and Hardie (2005).

Pareto/NBD model with static covariates: The standard Pareto/NBD model captures heterogeneity was solely using Gamma distributions. However, often exogenous knowledge, such as for example customer demographics, is available. The supplementary knowledge may explain part of the heterogeneity among the customers and therefore increase the predictive accuracy of the model. In addition, we can rely on these parameter estimates for inference, i.e. identify and quantify effects of contextual factors on the two underlying purchase and attrition processes. For technical details we refer to the technical note by Fader and Hardie (2007).

Pareto/NBD model with dynamic covariates: In many real-world applications customer purchase and attrition behavior may be influenced by covariates that vary over time. In consequence, the timing of a purchase and the corresponding value of at covariate a that time becomes relevant. Time-varying covariates can affect customer on aggregated level as well as on an individual level: In the first case, all customers are affected simultaneously, in the latter case a covariate is only relevant for a particular customer. For technical details we refer to the paper by Bachmann, Meierer and Näf (2020).

Value

Depending on the data object on which the model was fit, `pnbd` returns either an object of class clv.pnbd, clv.pnbd.static.cov, or clv.pnbd.dynamic.cov.

The function `summary` can be used to obtain and print a summary of the results. The generic accessor functions `coefficients`, `vcov`, `fitted`, `logLik`, `AIC`, `BIC`, and `nobs` are available.

Note

The Pareto/NBD model with dynamic covariates can currently not be fit with data that has a temporal resolution of less than one day (data that was built with time unit `hours`).

References

Schmittlein DC, Morrison DG, Colombo R (1987). "Counting Your Customers: Who-Are They and What Will They Do Next?" Management Science, 33(1), 1-24.

Bachmann P, Meierer M, Naef, J (2021). "The Role of Time-Varying Contextual Factors in Latent Attrition Models for Customer Base Analysis" Marketing Science 40(4). 783-809.

Fader PS, Hardie BGS (2005). "A Note on Deriving the Pareto/NBD Model and Related Expressions." URL http://www.brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf.

Fader PS, Hardie BGS (2007). "Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models." URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS (2020). "Deriving an Expression for P(X(t)=x) Under the Pareto/NBD Model." URL https://www.brucehardie.com/notes/012/pareto_NBD_pmf_derivation_rev.pdf

See Also

clvdata to create a clv data object, SetStaticCovariates to add static covariates to an existing clv data object.

gg to fit customer's average spending per transaction with the Gamma-Gamma model

predict to predict expected transactions, probability of being alive, and customer lifetime value for every customer

plot to plot the unconditional expectation as predicted by the fitted model

pmf for the probability to make exactly x transactions in the estimation period, given by the probability mass function (PMF).

newcustomer to predict the expected number of transactions for an average new customer.

The generic functions vcov, summary, fitted.

SetDynamicCovariates to add dynamic covariates on which the pnbd model can be fit.

Examples

```
data("apparelTrans")
clv.data.apparel <- clvdata(apparelTrans, date.format = "ymd",
                            time.unit = "w", estimation.split = 52)

# Fit standard pnbd model
pnbd(clv.data.apparel)

# Give initial guesses for the model parameters
pnbd(clv.data.apparel,
     start.params.model = c(r=0.5, alpha=15, s=0.5, beta=10))


# pass additional parameters to the optimizer (optimx)
#    Use Nelder-Mead as optimization method and print
#    detailed information about the optimization process
apparel.pnbd <- pnbd(clv.data.apparel,
                     optimx.args = list(method="Nelder-Mead",
                                        control=list(trace=6)))
```

```
# estimated coefs
coef(apparel.pnbd)

# summary of the fitted model
summary(apparel.pnbd)

# predict CLV etc for holdout period
predict(apparel.pnbd)

# predict CLV etc for the next 15 periods
predict(apparel.pnbd, prediction.end = 15)


# Estimate correlation as well
pnbd(clv.data.apparel, use.cor = TRUE)


# To estimate the pnbd model with static covariates,
#   add static covariates to the data
data("apparelStaticCov")
clv.data.static.cov <-
 SetStaticCovariates(clv.data.apparel,
                     data.cov.life = apparelStaticCov,
                     names.cov.life = c("Gender", "Channel"),
                     data.cov.trans = apparelStaticCov,
                     names.cov.trans = c("Gender", "Channel"))

# Fit pnbd with static covariates
pnbd(clv.data.static.cov)

# Give initial guesses for both covariate parameters
pnbd(clv.data.static.cov, start.params.trans = c(Gender=0.75, Channel=0.7),
                   start.params.life  = c(Gender=0.5, Channel=0.5))

# Use regularization
pnbd(clv.data.static.cov, reg.lambdas = c(trans = 5, life=5))

# Force the same coefficient to be used for both covariates
pnbd(clv.data.static.cov, names.cov.constr = "Gender",
                   start.params.constr = c(Gender=0.5))

# Fit model only with the Channel covariate for life but
# keep all trans covariates as is
pnbd(clv.data.static.cov, names.cov.life = c("Channel"))

# Add dynamic covariates data to the data object
#   add dynamic covariates to the data

## Not run:
data("apparelDynCov")
clv.data.dyn.cov <-
  SetDynamicCovariates(clv.data = clv.data.apparel,
                       data.cov.life = apparelDynCov,
```

```
                        data.cov.trans = apparelDynCov,
                        names.cov.life = c("High.Season", "Gender", "Channel"),
                        names.cov.trans = c("High.Season", "Gender", "Channel"),
                        name.date = "Cov.Date")


# Fit PNBD with dynamic covariates
pnbd(clv.data.dyn.cov)

# The same fitting options as for the
#  static covariate are available
pnbd(clv.data.dyn.cov, reg.lambdas = c(trans=10, life=2))

## End(Not run)
```

---

pnbd_CET *Pareto/NBD: Conditional Expected Transactions*

Description

Calculates the expected number of transactions in a given time period based on a customer's past transaction behavior and the Pareto/NBD model parameters.

`pnbd_nocov_CET` Conditional Expected Transactions without covariates

`pnbd_staticcov_CET` Conditional Expected Transactions with static covariates

Usage

```
pnbd_nocov_CET(r, alpha_0, s, beta_0, dPeriods, vX, vT_x, vT_cal)

pnbd_staticcov_CET(
  r,
  alpha_0,
  s,
  beta_0,
  dPeriods,
  vX,
  vT_x,
  vT_cal,
  vCovParams_trans,
  vCovParams_life,
  mCov_trans,
  mCov_life
)
```

Arguments

r	shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process
alpha_0	rate parameter of the Gamma distribution of the purchase process
s	shape parameter of the Gamma distribution for the lifetime process. The smaller s, the stronger the heterogeneity of customer lifetimes
beta_0	rate parameter for the Gamma distribution for the lifetime process.
dPeriods	number of periods to predict
vX	Frequency vector of length n counting the numbers of purchases.
vT_x	Recency vector of length n.
vT_cal	Vector of length n indicating the total number of periods of observation.
vCovParams_trans	Vector of estimated parameters for the transaction covariates.
vCovParams_life	Vector of estimated parameters for the lifetime covariates.
mCov_trans	Matrix containing the covariates data affecting the transaction process. One column for each covariate.
mCov_life	Matrix containing the covariates data affecting the lifetime process. One column for each covariate.

Details

`mCov_trans` is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vCovParams_trans` at the respective position.

`mCov_life` is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vCovParams_life` at the respective position.

Value

Returns a vector containing the conditional expected transactions for the existing customers in the Pareto/NBD model.

References

Schmittlein DC, Morrison DG, Colombo R (1987). “Counting Your Customers: Who-Are They and What Will They Do Next?” Management Science, 33(1), 1-24.

Bachmann P, Meierer M, Naef, J (2021). “The Role of Time-Varying Contextual Factors in Latent Attrition Models for Customer Base Analysis” Marketing Science 40(4). 783-809.

Fader PS, Hardie BGS (2005). “A Note on Deriving the Pareto/NBD Model and Related Expressions.” URL http://www.brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf.

Fader PS, Hardie BGS (2007). "Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models." URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS (2020). "Deriving an Expression for P(X(t)=x) Under the Pareto/NBD Model." URL https://www.brucehardie.com/notes/012/pareto_NBD_pmf_derivation_rev.pdf

---

pnbd_DERT *Pareto/NBD: Discounted Expected Residual Transactions*

Description

Calculates the discounted expected residual transactions.

pnbd_nocov_DERT Discounted expected residual transactions for the Pareto/NBD model without covariates

pnbd_staticcov_DERT Discounted expected residual transactions for the Pareto/NBD model with static covariates

Usage

```
pnbd_nocov_DERT(
  r,
  alpha_0,
  s,
  beta_0,
  continuous_discount_factor,
  vX,
  vT_x,
  vT_cal
)

pnbd_staticcov_DERT(
  r,
  alpha_0,
  s,
  beta_0,
  continuous_discount_factor,
  vX,
  vT_x,
  vT_cal,
  mCov_life,
  mCov_trans,
  vCovParams_life,
  vCovParams_trans
)
```

Arguments

- `r` shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process
- `alpha_0` rate parameter of the Gamma distribution of the purchase process
- `s` shape parameter of the Gamma distribution for the lifetime process. The smaller s, the stronger the heterogeneity of customer lifetimes
- `beta_0` rate parameter for the Gamma distribution for the lifetime process.
- `continuous_discount_factor` continuous discount factor to use
- `vX` Frequency vector of length n counting the numbers of purchases.
- `vT_x` Recency vector of length n.
- `vT_cal` Vector of length n indicating the total number of periods of observation.
- `mCov_life` Matrix containing the covariates data affecting the lifetime process. One column for each covariate.
- `mCov_trans` Matrix containing the covariates data affecting the transaction process. One column for each covariate.
- `vCovParams_life` Vector of estimated parameters for the lifetime covariates.
- `vCovParams_trans` Vector of estimated parameters for the transaction covariates.

Details

`mCov_trans` is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vCovParams_trans` at the respective position.

`mCov_life` is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to `vCovParams_life` at the respective position.

Value

Returns a vector with the DERT for each customer.

References

Schmittlein DC, Morrison DG, Colombo R (1987). “Counting Your Customers: Who-Are They and What Will They Do Next?” Management Science, 33(1), 1-24.

Bachmann P, Meierer M, Naef, J (2021). “The Role of Time-Varying Contextual Factors in Latent Attrition Models for Customer Base Analysis” Marketing Science 40(4). 783-809.

Fader PS, Hardie BGS (2005). “A Note on Deriving the Pareto/NBD Model and Related Expressions.” URL http://www.brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf.

Fader PS, Hardie BGS (2007). "Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models." URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS (2020). "Deriving an Expression for P(X(t)=x) Under the Pareto/NBD Model." URL https://www.brucehardie.com/notes/012/pareto_NBD_pmf_derivation_rev.pdf

---

`pnbd_expectation` *Pareto/NBD: Unconditional Expectation*

Description

Computes the expected number of repeat transactions in the interval (0, vT_i] for a randomly selected customer, where 0 is defined as the point when the customer came alive.

Usage

```
pnbd_nocov_expectation(r, s, alpha_0, beta_0, vT_i)

pnbd_staticcov_expectation(r, s, vAlpha_i, vBeta_i, vT_i)
```

Arguments

`r`	shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process
`s`	shape parameter of the Gamma distribution for the lifetime process. The smaller s, the stronger the heterogeneity of customer lifetimes
`alpha_0`	rate parameter of the Gamma distribution of the purchase process
`beta_0`	rate parameter for the Gamma distribution for the lifetime process.
`vT_i`	Number of periods since the customer came alive
`vAlpha_i`	Vector of individual parameters alpha
`vBeta_i`	Vector of individual parameters beta

Value

Returns the expected transaction values according to the chosen model.

References

Schmittlein DC, Morrison DG, Colombo R (1987). "Counting Your Customers: Who-Are They and What Will They Do Next?" Management Science, 33(1), 1-24.

Bachmann P, Meierer M, Naef, J (2021). "The Role of Time-Varying Contextual Factors in Latent Attrition Models for Customer Base Analysis" Marketing Science 40(4). 783-809.

Fader PS, Hardie BGS (2005). "A Note on Deriving the Pareto/NBD Model and Related Expressions." URL http://www.brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf.

Fader PS, Hardie BGS (2007). "Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models." URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS (2020). "Deriving an Expression for P(X(t)=x) Under the Pareto/NBD Model." URL https://www.brucehardie.com/notes/012/pareto_NBD_pmf_derivation_rev.pdf

---

pnbd_LL *Pareto/NBD: Log-Likelihood functions*

Description

Calculates the Log-Likelihood values for the Pareto/NBD model with and without covariates.

The function `pnbd_nocov_LL_ind` calculates the individual log-likelihood values for each customer for the given parameters.

The function `pnbd_nocov_LL_sum` calculates the log-likelihood value summed across customers for the given parameters.

The function `pnbd_staticcov_LL_ind` calculates the individual log-likelihood values for each customer for the given parameters and covariates.

The function `pnbd_staticcov_LL_sum` calculates the individual log-likelihood values summed across customers.

Usage

```
pnbd_nocov_LL_ind(vLogparams, vX, vT_x, vT_cal)

pnbd_nocov_LL_sum(vLogparams, vX, vT_x, vT_cal, vN)

pnbd_staticcov_LL_ind(vParams, vX, vT_x, vT_cal, mCov_life, mCov_trans)

pnbd_staticcov_LL_sum(vParams, vX, vT_x, vT_cal, vN, mCov_life, mCov_trans)
```

Arguments

vLogparams	vector with the Pareto/NBD model parameters at log scale. See Details.
vX	Frequency vector of length n counting the numbers of purchases.
vT_x	Recency vector of length n.
vT_cal	Vector of length n indicating the total number of periods of observation.
vN	The value ("number of times observed") with which the LL value of this observation is multiplied before summing across customers.

vParams	vector with the parameters for the Pareto/NBD model at log scale and the static covariates at original scale. See Details.

mCov_life	Matrix containing the covariates data affecting the lifetime process. One column for each covariate.

mCov_trans	Matrix containing the covariates data affecting the transaction process. One column for each covariate.

Details

vLogparams is a vector with model parameters r, alpha_0, s, beta_0 at log-scale, in this order.

vParams is vector with the Pareto/NBD model parameters at log scale, followed by the parameters for the lifetime covariates at original scale and then followed by the parameters for the transaction covariates at original scale

mCov_trans is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to vParams at the respective position.

mCov_life is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to vParams at the respective position.

Value

Returns the respective Log-Likelihood value(s) for the Pareto/NBD model with or without covariates.

References

Schmittlein DC, Morrison DG, Colombo R (1987). “Counting Your Customers: Who-Are They and What Will They Do Next?” Management Science, 33(1), 1-24.

Bachmann P, Meierer M, Naef, J (2021). “The Role of Time-Varying Contextual Factors in Latent Attrition Models for Customer Base Analysis” Marketing Science 40(4). 783-809.

Fader PS, Hardie BGS (2005). “A Note on Deriving the Pareto/NBD Model and Related Expressions.” URL http://www.brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf.

Fader PS, Hardie BGS (2007). “Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models.” URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS (2020). “Deriving an Expression for P(X(t)=x) Under the Pareto/NBD Model.” URL https://www.brucehardie.com/notes/012/pareto_NBD_pmf_derivation_rev.pdf

pnbd_PAlive *Pareto/NBD: Probability of Being Alive*

Description

Calculates the probability of a customer being alive at the end of the calibration period, based on a customer's past transaction behavior and the Pareto/NBD model parameters.

`pnbd_nocov_PAlive` P(alive) for the Pareto/NBD model without covariates

`pnbd_staticcov_PAlive` P(alive) for the Pareto/NBD model with static covariates

Usage

```
pnbd_nocov_PAlive(r, alpha_0, s, beta_0, vX, vT_x, vT_cal)

pnbd_staticcov_PAlive(
  r,
  alpha_0,
  s,
  beta_0,
  vX,
  vT_x,
  vT_cal,
  vCovParams_trans,
  vCovParams_life,
  mCov_trans,
  mCov_life
)
```

Arguments

`r`	shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process
`alpha_0`	rate parameter of the Gamma distribution of the purchase process
`s`	shape parameter of the Gamma distribution for the lifetime process. The smaller s, the stronger the heterogeneity of customer lifetimes
`beta_0`	rate parameter for the Gamma distribution for the lifetime process.
`vX`	Frequency vector of length n counting the numbers of purchases.
`vT_x`	Recency vector of length n.
`vT_cal`	Vector of length n indicating the total number of periods of observation.
`vCovParams_trans`	Vector of estimated parameters for the transaction covariates.
`vCovParams_life`	Vector of estimated parameters for the lifetime covariates.

mCov_trans — Matrix containing the covariates data affecting the transaction process. One column for each covariate.

mCov_life — Matrix containing the covariates data affecting the lifetime process. One column for each covariate.

Details

mCov_trans is a matrix containing the covariates data of the time-invariant covariates that affect the transaction process. Each column represents a different covariate. For every column a gamma parameter needs to added to vCovParams_trans at the respective position.

mCov_life is a matrix containing the covariates data of the time-invariant covariates that affect the lifetime process. Each column represents a different covariate. For every column a gamma parameter needs to added to vCovParams_life at the respective position.

Value

Returns a vector with the PAlive for each customer.

References

Schmittlein DC, Morrison DG, Colombo R (1987). “Counting Your Customers: Who-Are They and What Will They Do Next?” Management Science, 33(1), 1-24.

Bachmann P, Meierer M, Naef, J (2021). “The Role of Time-Varying Contextual Factors in Latent Attrition Models for Customer Base Analysis” Marketing Science 40(4). 783-809.

Fader PS, Hardie BGS (2005). “A Note on Deriving the Pareto/NBD Model and Related Expressions.” URL http://www.brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf.

Fader PS, Hardie BGS (2007). “Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models.” URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS (2020). “Deriving an Expression for P(X(t)=x) Under the Pareto/NBD Model.” URL https://www.brucehardie.com/notes/012/pareto_NBD_pmf_derivation_rev.pdf

---

pnbd_pmf — *Pareto/NBD: Probability Mass Function (PMF)*

Description

Calculate P(X(t)=x), the probability that a randomly selected customer makes exactly x transactions in the interval (0, t].

Usage

```
pnbd_nocov_PMF(r, alpha_0, s, beta_0, x, vT_i)

pnbd_staticcov_PMF(r, s, x, vAlpha_i, vBeta_i, vT_i)
```

Arguments

`r`	shape parameter of the Gamma distribution of the purchase process. The smaller r, the stronger the heterogeneity of the purchase process
`alpha_0`	rate parameter of the Gamma distribution of the purchase process
`s`	shape parameter of the Gamma distribution for the lifetime process. The smaller s, the stronger the heterogeneity of customer lifetimes
`beta_0`	rate parameter for the Gamma distribution for the lifetime process.
`x`	The number of transactions to calculate the probability for (unsigned integer).
`vT_i`	Number of periods since the customer came alive.
`vAlpha_i`	Vector of individual parameters alpha.
`vBeta_i`	Vector of individual parameters beta.

Value

Returns a vector of probabilities.

References

Schmittlein DC, Morrison DG, Colombo R (1987). "Counting Your Customers: Who-Are They and What Will They Do Next?" Management Science, 33(1), 1-24.

Bachmann P, Meierer M, Naef, J (2021). "The Role of Time-Varying Contextual Factors in Latent Attrition Models for Customer Base Analysis" Marketing Science 40(4). 783-809.

Fader PS, Hardie BGS (2005). "A Note on Deriving the Pareto/NBD Model and Related Expressions." URL http://www.brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf.

Fader PS, Hardie BGS (2007). "Incorporating time-invariant covariates into the Pareto/NBD and BG/NBD models." URL http://www.brucehardie.com/notes/019/time_invariant_covariates.pdf.

Fader PS, Hardie BGS (2020). "Deriving an Expression for P(X(t)=x) Under the Pareto/NBD Model." URL https://www.brucehardie.com/notes/012/pareto_NBD_pmf_derivation_rev.pdf

---

predict.clv.fitted.spending

Infer customers' spending

Description

Infer customer's mean spending per transaction and compare it to the actual mean spending in the holdout period.

New customer prediction: The fitted model can also be used to estimate the spending that a single, (fictional), average newly alive customer is expected to make at the moment of the first transaction. This is, for a customer which has no existing order history and that just "came alive". The data on which the model was fit and which is stored in it is NOT used for this prediction. See examples and newcustomer.spending for more details.

Usage

```
## S3 method for class 'clv.fitted.spending'
predict(
  object,
  newdata = NULL,
  uncertainty = c("none", "boots"),
  level = 0.9,
  num.boots = 100,
  verbose = TRUE,
  ...
)

## S4 method for signature 'clv.fitted.spending'
predict(
  object,
  newdata = NULL,
  uncertainty = c("none", "boots"),
  level = 0.9,
  num.boots = 100,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	A fitted spending model for which prediction is desired.
`newdata`	A `clv.data` object or data for the new customer prediction (see newcustomer.spending). If none or NULL is given, predictions are made for the data on which the model was fit.
`uncertainty`	Method to produce confidence intervals of the predictions (parameter uncertainty). Either "none" (default) or "boots".
`level`	Required confidence level, if `uncertainty="boots"`.
`num.boots`	Number of bootstrap repetitions, if `uncertainty="boots"`. A low number may not produce intervals for all customers if they are not sampled.
`verbose`	Show details about the running of the function.
`...`	Ignored

Details

If `newdata` is provided, the individual customer statistics underlying the model are calculated the same way as when the model was fit initially. Hence, if `remove.first.transaction` was `TRUE`, this will be applied to `newdata` as well.

To predict for new customers, the output of newcustomer.spending has to be given to `newdata`. See examples.

Value

An object of class `data.table` with columns:

- `Id` The respective customer identifier
- `actual.mean.spending` Actual mean spending per transaction in the holdout period. Only if there is a holdout period otherwise it is not reported.
- `predicted.mean.spending` The mean spending per transaction as predicted by the fitted spending model.

If predicting for new customers (using `newcustomer.spending()`), a numeric scalar indicating the expected spending is returned instead.

Uncertainty Estimates

Bootstrapping is used to provide confidence intervals of all predicted metrics. These provide an estimate of parameter uncertainty. To create bootstrapped data, customer ids are sampled with replacement until reaching original length and all transactions of the sampled customers are used to create a new `clv.data` object. A new model is fit on the bootstrapped data with the exact same specification as used when fitting `object` (incl. start parameters and 'optimx.args') and it is then used to predict on this data.

It is highly recommended to fit the original model (`object`) with a robust optimization method, such as Nelder-Mead (`optimx.args=list(method='Nelder-Mead')`). This ensures that the model can also be fit on the bootstrapped data.

All prediction parameters, incl `prediction.end` and `continuous.discount.factor`, are forwarded to the prediction on the bootstrapped data. Per customer, the boundaries of the confidence intervals of each predicted metric are the sample quantiles (`quantile(x, probs=c((1-level)/2, 1-(1-level)/2)`).

See clv.bootstrapped.apply to create a custom bootstrapping procedure.

See Also

models to predict spending: gg.

models to predict transactions: pnbd, bgnbd, ggomnbd.

`predict` for transaction models

`newdata.spending` to create data to predict for customers without order history

Examples

```
data("apparelTrans")

# Fit gg model on data
apparel.holdout <- clvdata(apparelTrans, time.unit="w",
                           estimation.split = 52, date.format = "ymd")
apparel.gg <- gg(apparel.holdout)

# Estimate customers' mean spending per transaction
predict(apparel.gg)
```

```
# Estimate the mean spending per transaction a single,
# fictional, average new customer is expected to make
# See ?newcustomer.spending() for more examples
predict(apparel.gg, newdata=newcustomer.spending())
```

---

predict.clv.fitted.transactions
Predict CLV from a fitted transaction model

Description

Probabilistic customer attrition models predict in general three expected characteristics for every customer:

- "conditional expected transactions" (CET), which is the number of transactions to expect from a customer during the prediction period,
- "probability of a customer being alive" (PAlive) at the end of the estimation period and
- "discounted expected residual transactions" (DERT) for every customer, which is the total number of transactions for the residual lifetime of a customer discounted to the end of the estimation period. In the case of time-varying covariates, instead of DERT, "discounted expected conditional transactions" (DECT) is predicted. DECT does only cover a finite time horizon in contrast to DERT. For continuous.discount.factor=0, DECT corresponds to CET.

In order to derive a monetary value such as CLV, customer spending has to be considered. If the clv.data object contains spending information, customer spending can be predicted using a Gamma/Gamma spending model for parameter predict.spending and the predicted CLV is be calculated (if the transaction model supports DERT/DECT). In this case, the prediction additionally contains the following two columns:

- "predicted.mean.spending", the mean spending per transactions as predicted by the spending model.
- "CLV", the customer lifetime value. CLV is the product of DERT/DECT and predicted spending.

Uncertainty estimates are available for all predicted quantities using bootstrapping.

New customer prediction: The fitted model can also be used to predict the number of transactions a fictional, single, average newly alive customer is expected to make at the moment of the first transaction ("coming alive"). This is, for a customer which has no existing order history. For covariate models, the prediction is for an average customer with the given covariates.

The individual-level unconditional expectation that is also used for the tracking plot is used to obtain this prediction. For models without covariates, the prediction hence is the same for all customers and independent of when a customer comes alive. For models with covariates, the prediction is the same for all customers with the same covariates.

The data on which the model was fit and which is stored in it is NOT used for this prediction. See examples and newcustomer for more details.

Usage

```
## S3 method for class 'clv.fitted.transactions'
predict(
  object,
  newdata = NULL,
  prediction.end = NULL,
  predict.spending = gg,
  continuous.discount.factor = log(1 + 0.1),
  uncertainty = c("none", "boots"),
  level = 0.9,
  num.boots = 100,
  verbose = TRUE,
  ...
)

## S4 method for signature 'clv.fitted.transactions'
predict(
  object,
  newdata = NULL,
  prediction.end = NULL,
  predict.spending = gg,
  continuous.discount.factor = log(1 + 0.1),
  uncertainty = c("none", "boots"),
  level = 0.9,
  num.boots = 100,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	A fitted clv transaction model for which prediction is desired.
`newdata`	A clv data object or data for the new customer prediction (see newcustomer) for which predictions should be made with the fitted model. If none or NULL is given, predictions are made for the data on which the model was fit.
`prediction.end`	Until what point in time to predict. This can be the number of periods (numeric) or a form of date/time object. See details.
`predict.spending`	Whether and how to predict spending and based on it also CLV, if possible. See details.
`continuous.discount.factor`	continuous discount factor to use to calculate `DERT/DECT`. Defaults to a 10% continuous annual rate. See details.
`uncertainty`	Method to produce confidence intervals of the predictions (parameter uncertainty). Either "none" (default) or "boots".
`level`	Required confidence level, if `uncertainty="boots"`.

num.boots	Number of bootstrap repetitions, if uncertainty="boots". A low number may not produce intervals for all customers if they are not sampled.
verbose	Show details about the running of the function.
...	Ignored

Details

predict.spending indicates whether to predict customers' spending and if so, the spending model to use. Accepted inputs are either a logical (TRUE/FALSE), a method to fit a spending model (i.e. gg), or an already fitted spending model. If provided TRUE, a Gamma-Gamma model is fit with default options. If argument newdata is provided, the spending model is fit on newdata. Predicting spending is only possible if the transaction data contains spending information. See examples for illustrations of valid inputs.

The newdata argument has to be a clv data object of the exact same class as the data object on which the model was fit. In case the model was fit with covariates, newdata needs to contain identically named covariate data.

The use case for newdata is mainly two-fold: First, to estimate model parameters only on a sample of the data and then use the fitted model object to predict or plot for the full data set provided through newdata. Second, for models with dynamic covariates, to provide a clv data object with longer covariates than contained in the data on which the model was estimated what allows to predict or plot further. When providing newdata, some models might require additional steps that can significantly increase runtime.

To predict for new customers, the output of newcustomer has to be given to newdata. See examples.

prediction.end indicates until when to predict or plot and can be given as either a point in time (of class Date, POSIXct, or character) or the number of periods. If prediction.end is of class character, the date/time format set when creating the data object is used for parsing. If prediction.end is the number of periods, the end of the fitting period serves as the reference point from which periods are counted. Only full periods may be specified. If prediction.end is omitted or NULL, it defaults to the end of the holdout period if present and to the end of the estimation period otherwise.

The first prediction period is defined to start right after the end of the estimation period. If for example weekly time units are used and the estimation period ends on Sunday 2019-01-01, then the first day of the first prediction period is Monday 2019-01-02. Each prediction period includes a total of 7 days and the first prediction period therefore will end on, and include, Sunday 2019-01-08. Subsequent prediction periods again start on Mondays and end on Sundays. If prediction.end indicates a timepoint on which to end, this timepoint is included in the prediction period.

continuous.discount.factor is the continuous rate used to discount the expected residual transactions (DERT/DECT). An annual rate of (100 x d)% equals a continuous rate delta = ln(1+d). To account for time units which are not annual, the continuous rate has to be further adjusted to delta=ln(1+d)/k, where k are the number of time units in a year.

Value

An object of class data.table with columns:

Id	The respective customer identifier
period.first	First timepoint of prediction period

period.last	Last timepoint of prediction period
period.length	Number of time units covered by the period indicated by period.first and period.last (including both ends).
PAlive	Probability to be alive at the end of the estimation period
CET	The Conditional Expected Transactions: The number of transactions expected until prediction.end.
DERT or DECT	Discounted Expected Residual Transactions or Discounted Expected Conditional Transactions for dynamic covariates models
actual.x	Actual number of transactions until prediction.end. Only if there is a holdout period and the prediction ends in it, otherwise not reported.
actual.total.spending	Actual total spending until prediction.end. Only if there is a holdout period and the prediction ends in it, otherwise not reported.
predicted.mean.spending	The mean spending per transactions as predicted by the spending model.
predicted.total.spending	The predicted total spending until prediction.end (CET*predicted.mean.spending).
predicted.CLV	Customer Lifetime Value based on DERT/DECT and predicted.mean.spending.

If predicting for new customers (using newcustomer()), a numeric scalar indicating the expected number of transactions is returned instead.

Uncertainty Estimates

Bootstrapping is used to provide confidence intervals of all predicted metrics. These provide an estimate of parameter uncertainty. To create bootstrapped data, customer ids are sampled with replacement until reaching original length and all transactions of the sampled customers are used to create a new clv.data object. A new model is fit on the bootstrapped data with the exact same specification as used when fitting object (incl. start parameters and 'optimx.args') and it is then used to predict on this data.

It is highly recommended to fit the original model (object) with a robust optimization method, such as Nelder-Mead (optimx.args=list(method='Nelder-Mead')). This ensures that the model can also be fit on the bootstrapped data.

All prediction parameters, incl prediction.end and continuous.discount.factor, are forwarded to the prediction on the bootstrapped data. Per customer, the boundaries of the confidence intervals of each predicted metric are the sample quantiles (quantile(x, probs=c((1-level)/2, 1-(1-level)/2)).

See clv.bootstrapped.apply to create a custom bootstrapping procedure.

See Also

models to predict transactions: pnbd, bgnbd, ggomnbd.

models to predict spending: gg.

predict for spending models

clv.bootstrapped.apply for bootstrapped model estimation

newcustomer to create data to predict for newly alive customers.

Examples

```
data("apparelTrans")
# Fit pnbd standard model on data, WITH holdout
apparel.holdout <- clvdata(apparelTrans, time.unit="w",
                           estimation.split=52, date.format="ymd")
apparel.pnbd <- pnbd(apparel.holdout)

# Predict until the end of the holdout period
predict(apparel.pnbd)

# Predict until 10 periods (weeks in this case) after
#   the end of the 37 weeks fitting period
predict(apparel.pnbd, prediction.end = 10) # ends on 2010-11-28

# Predict until 31th Dec 2016 with the timepoint as a character
predict(apparel.pnbd, prediction.end = "2016-12-31")

# Predict until 31th Dec 2016 with the timepoint as a Date
predict(apparel.pnbd, prediction.end = lubridate::ymd("2016-12-31"))


# Predict future transactions but not spending and CLV
predict(apparel.pnbd, predict.spending = FALSE)

# Predict spending by fitting a Gamma-Gamma model
predict(apparel.pnbd, predict.spending = gg)

# Fit a spending model separately and use it to predict spending
apparel.gg <- gg(apparel.holdout, remove.first.transaction = FALSE)
predict(apparel.pnbd, predict.spending = apparel.gg)


# Fit pnbd standard model WITHOUT holdout
pnc <- pnbd(clvdata(apparelTrans, time.unit="w", date.format="ymd"))

# This fails, because without holdout, a prediction.end is required
## Not run:
predict(pnc)

## End(Not run)

# But it works if providing a prediction.end
predict(pnc, prediction.end = 10) # ends on 2016-12-17

# Predict the number of transactions a single, fictional, average new
# customer is expected to make in the first 3.45 weeks since coming alive
# See ?newcustomer() for more examples
predict(apparel.pnbd, newdata = newcustomer(num.periods=3.45))
```

SetDynamicCovariates *Add Dynamic Covariates to a CLV data object*

Description

Add dynamic covariate data to an existing data object of class `clv.data`. The returned object can be used to fit models with dynamic covariates.

No covariate data can be added to a clv data object which already has any covariate set.

At least 1 covariate is needed for both processes and no categorical covariate may be of only a single category.

Usage

```
SetDynamicCovariates(
  clv.data,
  data.cov.life,
  data.cov.trans,
  names.cov.life,
  names.cov.trans,
  name.id = "Id",
  name.date = "Date"
)
```

Arguments

`clv.data`	CLV data object to add the covariates data to.
`data.cov.life`	Dynamic covariate data as `data.frame` or `data.table` for the lifetime process.
`data.cov.trans`	Dynamic covariate data as `data.frame` or `data.table` for the transaction process.
`names.cov.life`	Vector with names of the columns in `data.cov.life` that contain the covariates.
`names.cov.trans`	Vector with names of the columns in `data.cov.trans` that contain the covariates.
`name.id`	Name of the column to find the Id data for both, `data.cov.life` and `data.cov.trans`.
`name.date`	Name of the column to find the Date data for both, `data.cov.life` and `data.cov.trans`.

Details

data.cov.life and data.cov.trans are data.frames or data.tables that each contain exactly 1 row for every combination of timepoint and customer. For each customer appearing in the transaction data there needs to be covariate data at every timepoint that marks the start of a period as defined by time.unit. It has to range from the start of the estimation sample (timepoint.estimation.start) until the end of the period in which the end of the holdout sample (timepoint.holdout.end) falls. See the the provided data apparelDynCov for illustration. Covariates of class character or factor are converted to k-1 numeric dummies.

Date as character If the Date column in the covariate data is of type character, the date.format given when creating the the clv.data object is used for parsing.

Value

An object of class clv.data.dynamic.covariates. See the class definition clv.data.dynamic.covariates for more details about the returned object.

Examples

```
## Not run:
data("apparelTrans")
data("apparelDynCov")

# Create a clv data object without covariates
clv.data.apparel <- clvdata(apparelTrans, time.unit="w",
                            date.format="ymd")

# Add dynamic covariate data
clv.data.dyn.cov  <-
   SetDynamicCovariates(clv.data.apparel,
                       data.cov.life  = apparelDynCov,
                       names.cov.life = c("High.Season", "Gender", "Channel"),
                       data.cov.trans = apparelDynCov,
                       names.cov.trans = c("High.Season", "Gender", "Channel"),
                       name.id = "Id",
                       name.date = "Cov.Date")

# summary output about covariates data
summary(clv.data.dyn.cov)

# fit pnbd model with dynamic covariates
pnbd(clv.data.dyn.cov)

## End(Not run)
```

SetStaticCovariates *Add Static Covariates to a CLV data object*

Description

Add static covariate data to an existing data object of class `clv.data`. The returned object then can be used to fit models with static covariates.

No covariate data can be added to a clv data object which already has any covariate set.

At least 1 covariate is needed for both processes and no categorical covariate may be of only a single category.

Usage

```
SetStaticCovariates(
  clv.data,
  data.cov.life,
  data.cov.trans,
  names.cov.life,
  names.cov.trans,
  name.id = "Id"
)
```

Arguments

`clv.data`	CLV data object to add the covariates data to.
`data.cov.life`	Static covariate data as `data.frame` or `data.table` for the lifetime process.
`data.cov.trans`	Static covariate data as `data.frame` or `data.table` for the transaction process.
`names.cov.life`	Vector with names of the columns in `data.cov.life` that contain the covariates.
`names.cov.trans`	Vector with names of the columns in `data.cov.trans` that contain the covariates.
`name.id`	Name of the column to find the Id data for both, `data.cov.life` and `data.cov.trans`.

Details

`data.cov.life` and `data.cov.trans` are `data.frames` or `data.tables` that each contain exactly one single row of covariate data for every customer appearing in the transaction data. Covariates of class `character` or `factor` are converted to k-1 numeric dummy variables.

Value

An object of class `clv.data.static.covariates`. See the class definition clv.data.static.covariates for more details about the returned object.

Examples

```
data("apparelTrans")
data("apparelStaticCov")

# Create a clv data object without covariates
```

```
clv.data.apparel <- clvdata(apparelTrans, time.unit="w",
                            date.format="ymd")

# Add static covariate data
clv.data.apparel.cov  <-
   SetStaticCovariates(clv.data.apparel,
                       data.cov.life  = apparelStaticCov,
                       names.cov.life = "Gender",
                       data.cov.trans = apparelStaticCov,
                       names.cov.trans = "Gender",
                       name.id = "Id")

# more summary output
summary(clv.data.apparel.cov)

# fit model with static covariates
pnbd(clv.data.apparel.cov)
```

---

spending *Formula Interface for Spending Models*

Description

Fit models for customer spending (currently only the Gamma-Gamma model).

Usage

```
spending(family, data, optimx.args = list(), verbose = TRUE, ...)
```

Arguments

`family`	A spending model (currently only gg).
`data`	A `clv.data` object.
`optimx.args`	Additional arguments to control the optimization which are forwarded to `optimx::optimx`. If multiple optimization methods are specified, only the result of the last method is further processed.
`verbose`	Show details about the running of the function.
`...`	Forwarded to model specified in `family`.

Value

Returns an object of the respective model which was fit.

See Also

Spending models for `family`: gg.

latentAttrition to fit latent attrition models with a formula interface

Examples

```
data("cdnow")
clv.cdnow <- clvdata(data.transactions = cdnow, date.format="ymd",
                     time.unit = "weeks")

# Fit gg
spending(family=gg, data=clv.cdnow)

# Fit gg with start params
spending(family=gg, data=clv.cdnow,
         start.params.model=c(p=0.5, q=15, gamma=2))

# Fit gg, do not remove first transaction
spending(family=gg, data=clv.cdnow, remove.first.transaction=FALSE)


## No formula may be given to specify covariates because currently
## no spending model uses covariates
```

`subset.clv.data` *Subsetting clv.data*

Description

Returns a subset of the transaction data stored within the given `clv.data` object which meet conditions. The given expression are forwarded to the `data.table` of transactions. Possible rows to subset and select are `Id`, `Date`, and `Price` (if present).

Usage

```
## S3 method for class 'clv.data'
subset(x, subset, select, sample = c("full", "estimation", "holdout"), ...)
```

Arguments

x	`clv.data` to subset
subset	logical expression indicating rows to keep
select	expression indicating columns to keep
sample	Name of sample for which transactions should be extracted,
...	further arguments passed to `data.table::subset`

Value

A copy of the data.table of selected transactions. May contain columns Id, Date, and Price.

See Also

data.table's subset

Examples

```
 # dont test because ncpu=2 limit on cran (too fast)
library(data.table) # for between()
data(cdnow)

clv.cdnow <- clvdata(cdnow,
  date.format="ymd",
  time.unit = "week",
  estimation.split = "1997-09-30")

# all transactions of customer "1"
subset(clv.cdnow, Id=="1")
subset(clv.cdnow, subset = Id=="1")

# all transactions of customer "111" in the estimation period...
subset(clv.cdnow, Id=="111", sample="estimation")
# ... and in the holdout period
subset(clv.cdnow, Id=="111", sample="holdout")

# all transactions of customers "1", "2", and "999"
subset(clv.cdnow, Id %in% c("1","2","999"))

# all transactions on "1997-02-16"
subset(clv.cdnow, Date == "1997-02-16")

# all transactions between "1997-02-01" and "1997-02-16"
subset(clv.cdnow, Date >= "1997-02-01" & Date <= "1997-02-16")
# same using data.table's between
subset(clv.cdnow, between(Date, "1997-02-01","1997-02-16"))

# all transactions with a value between 50 and 100
subset(clv.cdnow, Price >= 50 & Price <= 100)
# same using data.table's between
subset(clv.cdnow, between(Price, 50, 100))

# only keep Id of transactions on "1997-02-16"
subset(clv.cdnow, Date == "1997-02-16", "Id")
```

summary.clv.fitted — *Summarizing a fitted CLV model*

Description

Summary method for fitted CLV models that provides statistics about the estimated parameters and information about the optimization process. If multiple optimization methods were used (for example if specified in parameter `optimx.args`), all information here refers to the last method/row of the resulting `optimx` object.

Usage

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

## S3 method for class 'clv.fitted.transactions.static.cov'
summary(object, ...)

## S3 method for class 'summary.clv.fitted'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

`object`	A fitted CLV model
`...`	Ignored for `summary`, forwarded to `printCoefmat` for `print`.
`x`	an object of class `"summary.clv.no.covariates"`, usually, a result of a call to `summary.clv.no.covariates`.
`digits`	the number of significant digits to use when printing.
`signif.stars`	logical. If TRUE, 'significance stars' are printed for each coefficient.

Value

This function computes and returns a list of summary information of the fitted model given in `object`. It returns a list of class `summary.clv.no.covariates` that contains the following components:

`name.model`	the name of the fitted model.
`call`	The call used to fit the model.
`tp.estimation.start`	Date or POSIXct indicating when the fitting period started.

tp.estimation.end
: Date or POSIXct indicating when the fitting period ended.

estimation.period.in.tu
: Length of fitting period in time.units.

time.unit
: Time unit that defines a single period.

coefficients
: a px4 matrix with columns for the estimated coefficients, its standard error, the t-statistic and corresponding (two-sided) p-value.

estimated.LL
: the value of the log-likelihood function at the found solution.

AIC
: Akaike's An Information Criterion for the fitted model.

BIC
: Schwarz' Bayesian Information Criterion for the fitted model.

KKT1
: Karush-Kuhn-Tucker optimality conditions of the first order, as returned by optimx.

KKT2
: Karush-Kuhn-Tucker optimality conditions of the second order, as returned by optimx.

fevals
: The number of calls to the log-likelihood function during optimization.

method
: The last method used to obtain the final solution.

additional.options
: A list of additional options used for model fitting.

  Correlation Whether the correlation between the purchase and the attrition process was estimated.

  estimated.param.cor Correlation coefficient measuring the correlation between the two processes, if used.

For models fits with static covariates, the list additionally is of class summary.clv.static.covariates and the list in additional.options contains the following elements:

additional.options
: **Regularization** Whether L2 regularization for parameters of contextual factors was used.

  lambda.life The regularization lambda used for the parameters of the Lifetime process, if used.

  lambda.trans The regularization lambda used for the parameters of the Transaction process, if used.

  Constraint covs Whether any covariate parameters were forced to be the same for both processes.

  Constraint params Name of the covariate parameters which were constraint, if used.

See Also

The model fitting functions pnbd.

Function coef will extract the coefficients matrix including summary statistics and function vcov will extract the vcov from the returned summary object.

Examples

```
data("apparelTrans")

# Fit pnbd standard model, no covariates
clv.data.apparel <- clvdata(apparelTrans, time.unit="w",
                                estimation.split=52, date.format="ymd")
pnbd.apparel <- pnbd(clv.data.apparel)

# summary about model fit
summary(pnbd.apparel)

# Add static covariate data
data("apparelStaticCov")
data.apparel.cov  <-
   SetStaticCovariates(clv.data.apparel,
                       data.cov.life  = apparelStaticCov,
                       names.cov.life = "Gender",
                       data.cov.trans = apparelStaticCov,
                       names.cov.trans = "Gender",
                       name.id = "Id")

# fit model with covariates and regualization
pnbd.apparel.cov <- pnbd(data.apparel.cov,
                         reg.lambdas = c(life=2, trans=4))

# additional summary about covariate parameters
#   and used regularization
summary(pnbd.apparel.cov)
```

`vcov.clv.fitted` *Calculate Variance-Covariance Matrix for CLV Models fitted with Maximum Likelihood Estimation*

Description

Returns the variance-covariance matrix of the parameters of the fitted model object. The variance-covariance matrix is derived from the Hessian that results from the optimization procedure. First, the Moore-Penrose generalized inverse of the Hessian is used to obtain an estimate of the variance-covariance matrix. Next, because some parameters may be transformed for the purpose of restricting their value during the log-likelihood estimation, the variance estimates are adapted to be comparable to the reported coefficient estimates. If the result is not positive definite, Matrix::nearPD is used with standard settings to find the nearest positive definite matrix.

If multiple estimation methods were used, the Hessian of the last method is used.

Usage

```
## S3 method for class 'clv.fitted'
vcov(object, ...)
```

Arguments

`object`	a fitted clv model object
`...`	Ignored

Value

A matrix of the estimated covariances between the parameters of the model. The row and column names correspond to the parameter names given by the `coef` method.

See Also

MASS::ginv, Matrix::nearPD

Index

∗ **datasets**
apparelDynCov, 6
apparelDynCovFuture, 6
apparelStaticCov, 7
apparelTrans, 7
cdnow, 25

apparelDynCov, 6
apparelDynCovFuture, 6
apparelStaticCov, 7
apparelTrans, 7
as.clv.data, 8
as.data.frame.clv.data, 9
as.data.table.clv.data, 11

bgbb, 12
bgbb,clv.data-method *(*bgbb*)*, 12
bgbb,clv.data.dynamic.covariates-method *(*bgbb*)*, 12
bgbb,clv.data.static.covariates-method *(*bgbb*)*, 12
bgnbd, 14, *26*, *46*, *63*, *80*, *84*
bgnbd,clv.data-method *(*bgnbd*)*, 14
bgnbd,clv.data.dynamic.covariates-method *(*bgnbd*)*, 14
bgnbd,clv.data.static.covariates-method *(*bgnbd*)*, 14
bgnbd_CET, 18
bgnbd_expectation, 19
bgnbd_LL, 20
bgnbd_nocov_CET *(*bgnbd_CET*)*, 18
bgnbd_nocov_expectation *(*bgnbd_expectation*)*, 19
bgnbd_nocov_LL_ind *(*bgnbd_LL*)*, 20
bgnbd_nocov_LL_sum *(*bgnbd_LL*)*, 20
bgnbd_nocov_PAlive *(*bgnbd_PAlive*)*, 22
bgnbd_nocov_PMF *(*bgnbd_pmf*)*, 24
bgnbd_PAlive, 22
bgnbd_pmf, 24
bgnbd_staticcov_CET *(*bgnbd_CET*)*, 18
bgnbd_staticcov_expectation *(*bgnbd_expectation*)*, 19
bgnbd_staticcov_LL_ind *(*bgnbd_LL*)*, 20
bgnbd_staticcov_LL_sum *(*bgnbd_LL*)*, 20
bgnbd_staticcov_PAlive *(*bgnbd_PAlive*)*, 22
bgnbd_staticcov_PMF *(*bgnbd_pmf*)*, 24

cdnow, 25
clv.bgnbd, *16*
clv.bgnbd.static.cov, *16*
clv.bootstrapped.apply, 25, *80*, *84*
clv.data, *28*
clv.data.dynamic.covariates, *87*
clv.data.static.covariates, *88*
clv.gg, *32*
clv.ggomnbd, *35*
clv.ggomnbd.static.cov, *35*
clv.pnbd, *66*
clv.pnbd.dynamic.cov, *66*
clv.pnbd.static.cov, *66*
clvdata, *9*, *16*, 27, *32*, *35*, *67*
CLVTools *(*CLVTools-package*)*, 5
CLVTools-package, 5

fitted, *16*, *32*, *35*, *66*, *67*
fitted.clv.fitted, 29

gg, *16*, 31, *35*, *67*, *80*, *83*, *84*, *89*
gg,clv.data-method *(*gg*)*, 31
gg_LL, 44
ggomnbd, *26*, 33, *46*, *63*, *80*, *84*
ggomnbd,clv.data-method *(*ggomnbd*)*, 33
ggomnbd,clv.data.dynamic.covariates-method *(*ggomnbd*)*, 33
ggomnbd,clv.data.static.covariates-method *(*ggomnbd*)*, 33
ggomnbd_CET, 37
ggomnbd_expectation, 38
ggomnbd_LL, 39

ggomnbd_nocov_CET (ggomnbd_CET), 37
ggomnbd_nocov_expectation (ggomnbd_expectation), 38
ggomnbd_nocov_LL_ind (ggomnbd_LL), 39
ggomnbd_nocov_LL_sum (ggomnbd_LL), 39
ggomnbd_nocov_PAlive (ggomnbd_PAlive), 41
ggomnbd_nocov_PMF (ggomnbd_PMF), 42
ggomnbd_PAlive, 41
ggomnbd_PMF, 42
ggomnbd_staticcov_CET (ggomnbd_CET), 37
ggomnbd_staticcov_expectation (ggomnbd_expectation), 38
ggomnbd_staticcov_LL_ind (ggomnbd_LL), 39
ggomnbd_staticcov_LL_sum (ggomnbd_LL), 39
ggomnbd_staticcov_PAlive (ggomnbd_PAlive), 41
ggomnbd_staticcov_PMF (ggomnbd_PMF), 42
ggplot2::geom_bar, *54, 55*
ggplot2::stat_density, *54, 55*

latentAttrition, 45, *89*
lrtest, 47
lrtest,clv.fitted-method (lrtest), 47
lrtest.clv.fitted (lrtest), 47

MASS::ginv, *95*
Matrix::nearPD, *94, 95*

newcustomer, *16*, *35*, 48, *67*, *81–84*
newcustomer.spending, *78, 79*
newdata.spending, *80*
nobs.clv.data, 51
nobs.clv.fitted, 52

optimx::optimx, *13, 14*, *31*, *34*, *45*, *65*, *89*

parse_date_time, *28*
plot, *16*, *28*, *30*, *32*, *35*, *55*, *57*, *63*, *67*
plot (plot.clv.fitted.transactions), 58
plot,clv.fitted.spending-method (plot.clv.fitted.spending), 56
plot,clv.fitted.transactions-method (plot.clv.fitted.transactions), 58
plot.clv.data, 52, *56*, *58*, *61*
plot.clv.fitted.spending, 56, *61*
plot.clv.fitted.transactions, 58
pmf, *16*, *35*, *61*, 62, *67*
pmf,clv.fitted.transactions-method (pmf), 62
pnbd, *26*, *28*, *46*, *63*, 63, *80*, *84*, *93*
pnbd,clv.data-method (pnbd), 63
pnbd,clv.data.dynamic.covariates-method (pnbd), 63
pnbd,clv.data.static.covariates-method (pnbd), 63
pnbd_CET, 69
pnbd_DERT, 71
pnbd_expectation, 73
pnbd_LL, 74
pnbd_nocov_CET (pnbd_CET), 69
pnbd_nocov_DERT (pnbd_DERT), 71
pnbd_nocov_expectation (pnbd_expectation), 73
pnbd_nocov_LL_ind (pnbd_LL), 74
pnbd_nocov_LL_sum (pnbd_LL), 74
pnbd_nocov_PAlive (pnbd_PAlive), 76
pnbd_nocov_PMF (pnbd_pmf), 77
pnbd_PAlive, 76
pnbd_pmf, 77
pnbd_staticcov_CET (pnbd_CET), 69
pnbd_staticcov_DERT (pnbd_DERT), 71
pnbd_staticcov_expectation (pnbd_expectation), 73
pnbd_staticcov_LL_ind (pnbd_LL), 74
pnbd_staticcov_LL_sum (pnbd_LL), 74
pnbd_staticcov_PAlive (pnbd_PAlive), 76
pnbd_staticcov_PMF (pnbd_pmf), 77
predict, *16*, *32*, *35*, *67*, *80*, *84*
predict (predict.clv.fitted.transactions), 81
predict (spending), *48, 49*
predict (transactions), *48, 49*
predict,clv.fitted.spending-method (predict.clv.fitted.spending), 78
predict,clv.fitted.transactions-method (predict.clv.fitted.transactions), 81
predict.clv.fitted.spending, 78
predict.clv.fitted.transactions, 81
print.summary.clv.fitted (summary.clv.fitted), 92

SetDynamicCovariates, *28*, *67*, 86
SetStaticCovariates, *16*, *28*, *35*, *67*, 87
spending, *46*, 89
subset, *91*
subset *(*subset.clv.data*)*, 90
subset.clv.data, 90
summary, *16*, *28*, *32*, *35*, *66, 67*
summary.clv.fitted, 92

tracking plot, *81*

vcov, *16*, *32*, *35*, *66, 67*
vcov.clv.fitted, 94