The CLVTools Package

Walktrough for the CLVTools package

Setup the R environment

Install the stable version from CRAN:

```
install.packages("CLVTools")
```

Install the development version from GitHub (using the `devtools` [@devtools] package):

```
install.packages("devtools")
devtools::install_github("CLVTools", ref = "master")
```

Load the package

```
library("CLVTools")
```

Load sample data provided in the package

As Input data `CLVTools` requires customers' transaction history. Every transaction record consists of a purchase date and customer ID. Optionally, the price of the transaction may be included to allow for prediction of future customer spending using an additional Gamma/Gamma model[@Fader2005b; @Colombo1999]. Using the full history of transaction data allows for comprehensive plots and summary statistics, which allow the identification of possible issues prior to model estimation. Data may be provided as `data.frame` or `data.table` [@data.table].

It is common practice to split time series data into two parts, an estimation and a holdout period. The model is estimated based on the data from the estimation period while the data from the holdout period allows to rigorously assess model performance. Once model performance is checked on known data one can proceed to predict data without a holdout period. The length of the estimation period is heavily dependent on the characteristics of the analyzed dataset. We recommend to choose an estimation period that contains in minimum the lenght of the average inter-purchase time. Note that all customers in the dataset need to purchase at least once during the estimation period, i.e. these models do not account for prospects who have not yet a purchase record.

Some models included in `CLVTools` allow to model the impact of covariates. These covariates may explain heterogeneity among the customers and therefore increase the predictive accuracy of the model. At the same time we may also identify and quantify the effects of these covariates on customer purchase and customer attrition. `CLVTools` distinguishes between time-invariant and time-varying covariates. Time-invariant covariates include customer characteristics such as demographics that do not change over time. Time-varying covariates are allowed to change over time. They include for example direct marketing information or seasonal patterns.

For the following example, we use data of a retailer in the apparel industry. The dataset contains transactional detail records for every customer consisting of customer id, date of purchase and the total monetary value of the transaction.The apparel dataset is available in the `CLVTools` package. Use the `data(apparelTrans)` to load it:

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

```
#>          Id       Date  Price
#>    1:    1 2005-01-03  26.95
#>    2:    1 2005-03-29  38.95
#>    3:    1 2005-03-31  93.73
#>    4:    1 2005-05-04 224.96
#>    5:    1 2005-11-04 104.95
#>   ---
#> 3644: 1221 2005-05-10  62.93
#> 3645: 1221 2005-05-11   8.95
#> 3646: 1222 2005-01-03  45.95
#> 3647: 1222 2005-02-10   3.98
#> 3648: 1222 2005-02-21 189.95
```

Initialize the CLV-Object

Before we estimate a model, we are required to initialize a data object using the `clvdata()` command. The data object contains the prepared transactional data and is later used as input for model fitting. Make sure to store the generated object in a variable, e.g. in our example `clv.apparel`.

Through the argument `data.transactions` a `data.frame` or `data.table` which contains the transaction records, is specified. In our example this is `data.transactions=apparelTrans`. The argument `date.format` is used to indicate the format of the date variable in the data used. The date format in the apparel dataset is given as "year-month-day" (i.e., "2005-01-03"), therefore we set `date.format="ymd"`. Other combinations such as `date.format="dmy"` are possible. See the documentation of `lubridate` [@lubridate] for all details. `time.unit` is the scale used to measure time between two dates. For this dataset and in most other cases The argument `time.unit="week"` is the preferred choice. Abbreviations may be used (i.e. "w"). `estimation.split` indicates the length of the estimation period. Either the length of the estimation period (in previous specified time units) or the date at which the estimation period ends can be specified. If no value is provided, the whole dataset is used as estimation period (i.e. no holdout period). In this example, we use an estimation period of 40 weeks. Finally, the three name arguments indicate the column names for customer ID, date and price in the supplied dataset.

```
clv.apparel <- clvdata(apparelTrans,
                       date.format="ymd",
                       time.unit = "week",
                       estimation.split = 40,
                       name.id = "Id",
                       name.date = "Date",
                       name.price = "Price")
```

Check the `clvdata` Object

To get details on the `clvdata` object, print it to the console

```
clv.apparel
#> CLV Transaction Data
#>
#> Call:
#> clvdata(data.transactions = apparelTrans, date.format = "ymd",
#>     time.unit = "week", estimation.split = 40, name.id = "Id",
#>     name.date = "Date", name.price = "Price")
#>
#> Total # customers    250
#> Total # transactions 3366
#>
```

```
#>
#> Time unit          Weeks
#>
#> Estimation start   2005-01-03
#> Estimation end     2005-10-10
#> Estimation length 40.0000 Weeks
#>
#> Holdout start      2005-10-11
#> Holdout end        2006-07-16
#> Holdout length     39.71429 Weeks
```

Alternatively the `summary()` command provides full detailed summary statistics for the provided transactional detail. `summary()` is available at any step in the process of estimating a probabilistic customer attrition model with `CLVTools`. The result output is updated accordingly and additional information is added to the summary statistics.`nobs()` extracts the number of observations.

```
summary(clv.apparel)
#> CLV Transaction Data
#>
#> Time unit          Weeks
#> Estimation length 40.0000 Weeks
#> Holdout length     39.71429 Weeks
#>
#> Transaction Data Summary
#>                                  Estimation      Holdout         Total
#> Number of customers              -               -               250
#> First Transaction in period      2005-01-03      2005-10-11      2005-01-03
#> Last Transaction in period       2005-10-10      2006-07-16      2006-07-16
#> Total # Transactions             1989            1377            3366
#> Mean # Transactions per cust     7.956           14.196          13.464
#> (SD)                             10.953          14.395          20.654
#> Mean Spending per Transaction    218.610         217.099         217.992
#> (SD)                             241.898         243.393         242.476
#> Total Spending                   434815.030      298944.740      733759.770
#> Total # zero repeaters           68              153             67
#> Percentage # zero repeaters      0.272           0.612           0.268
```

For the apparel dataset we observe a total of 250 customers who made in total 3366 repeat purchases. Approximately 67% of the customers are zero repeaters, which means that the majority of the customers do not return to the store after their first purchase.

Estimate Model Parameters

After initializing the object, we are able to estimate the first probabilistic latent attrtition model. We start with the standard Pareto/NBD model [@Schmittlein1987] and therefore use the command `pnbd()` to fit the model and estimate model parameters. `clv.data` specifies the initialized object prepared in the last step. Optionally, starting values for the model parameters and control settings for the optimization algorithm may be provided: The argument `start.params.model` allows to assign a vector (e.g. `c(alpha=1, beta=2, s=1, beta=2)` in the case of the Pareto/NBD model) of starting values for the optimization. This is useful if prior knowledge on the parameters of the distributions are available. By default starting values are set to 1 for all parameters. The argument `optimx.args` provides an option to control settings for the optimization routine. It passes a list of arguments to the optimizer. All options known from the package `optimx` [@optimx1; @optimx2] may be used. This option enables users to specify specific optimization algorithms, set upper and/or lower limits or enable tracing information on the progress of the optimization. In the case of the standard Pareto/NBD model, `CLVTools` uses by default the optimization method `L-BFGS-G`

[@byrd1995limited]. If the result of the optimization is infeasible, the optimization automatically switches to the more robust but often slower `Nelder-Mead` method [@nelder1965simplex]. `verbose` shows additional output.

```
est.pnbd <- pnbd(clv.data = clv.apparel)
#> Starting estimation...
#> Estimation finished!
est.pnbd
#> Pareto NBD Standard  Model
#>
#> Call:
#> suppressPackageStartupMessages({
#>     oldLC <- Sys.getlocale(category = "LC_COLLATE")
#>     Sys.setlocale(category = "LC_COLLATE", locale = "C")
#>     on.exit(Sys.setlocale(category = "LC_COLLATE", locale = oldLC))
#>     devtools::document(roclets = c("rd", "collate", "namespace",
#>         "vignette"))
#> })
#>
#> Coefficients:
#>       r    alpha        s     beta
#>  0.6650   2.6195   0.5563  17.3040
#> KKT1: TRUE
#> KKT2: TRUE
#>
#> Used Options:
#> Correlation:     FALSE
```

If we assign starting parameters and addititonal arguments for the optimizer we use:

```
est.pnbd <- pnbd(clv.data = clv.apparel,
                     start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                     optimx.args = list(control=list(trace=5),
                                        method="Nelder-Mead",
                                        upper=NULL,
                                        lower=NULL))
```

Parameter estimates may be reported by either printing the estimated object (i.e. `est.pnbd`) directly in the console or by calling `summary(est.pnbd)` to get a more detailed report including the likelihood value as well as AIC and BIC. Alternatively parameters may be directly extracted using `coef(est.pnbd)`. Also `loglik()`, `confint()` and `vcov()` are available to directly access the Loglikelihood value, confidence intervals for the parameters and to calculate the Variance-Covariance Matrix for the fitted model. For the standard Pareto/NBD model, we get 4 parameters r, α, s and β. where r, α represent the shape and scale parameter of the gamma distribution that determines the purchase rate and s, β of the attrition rate across individual customers. r/α can be interpreted as the mean purchase and s/β as the mean attrition rate. A significance level is provided for each parameter estimates. In the case of the apparelTrans dataset we observe a an average purchase rate of $r/\alpha = 0.254$ transactions and an average attrition rate of $s/\beta = 0.032$ per customer per week. KKT 1 and 2 indicate the Karush-Kuhn-Tucker optimality conditions of the first and second order [@KKT]. If those criteria are not met, the optimizer has probably not arrived at an optimal solution. If this is the case it is usually a good idea to rerun the estimation using alternative starting values.

```
#Full detailed summary of the parameter estimates
summary(est.pnbd)
#> Pareto NBD Standard  Model
#>
#> Call:
```

```
#> suppressPackageStartupMessages({
#>     oldLC <- Sys.getlocale(category = "LC_COLLATE")
#>     Sys.setlocale(category = "LC_COLLATE", locale = "C")
#>     on.exit(Sys.setlocale(category = "LC_COLLATE", locale = oldLC))
#>     devtools::document(roclets = c("rd", "collate", "namespace",
#>         "vignette"))
#> })
#>
#> Fitting period:
#> Estimation start  2005-01-03
#> Estimation end    2005-10-10
#> Estimation length 40.0000 Weeks
#>
#> Coefficients:
#>       Estimate Std. Error z-val Pr(>|z|)
#> r      0.66501    0.09287 7.160 8.05e-13 ***
#> alpha  2.61947    0.39919 6.562 5.31e-11 ***
#> s      0.55628    0.28072 1.982   0.0475 *
#> beta  17.30401   13.83691 1.251   0.2111
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Optimization info:
#> LL      -3506.5134
#> AIC     7021.0267
#> BIC     7035.1126
#> KKT 1   TRUE
#> KKT 2   TRUE
#> fevals 18.0000
#> Method L-BFGS-B
#>
#> Used Options:
#> Correlation FALSE

#Extract the coefficients only
coef(est.pnbd)
#>          r      alpha          s       beta
#>  0.6650103  2.6194708  0.5562817 17.3040094
#Alternative: oefficients(est.pnbd.obj)
```

To extract only the coefficients, we can use `coef()`. To access the confidence intervals for all parameters `confint()` is available.

```
#Extract the coefficients only
coef(est.pnbd)
#>          r      alpha          s       beta
#>  0.6650103  2.6194708  0.5562817 17.3040094
#Alternative: oefficients(est.pnbd.obj)

#Extrac the confidence intervals
confint(est.pnbd)
#>              2.5 %     97.5 %
#> r      0.482979103  0.8470415
#> alpha  1.837070256  3.4018713
```

```
#> s      0.006089078  1.1064743
#> beta  -9.815831673 44.4238505
```

In order to get the Likelihood value and the corresponding Variance-Covariance Matrix we use the follwing commands:

```
# LogLikelihood at maximum
logLik(est.pnbd)
#> 'log Lik.' -3506.513 (df=4)

# Variance-Covariance Matrix at maximum
vcov(est.pnbd)
#>                  r       alpha             s        beta
#> r      0.008625721  0.03038651 -0.006451792  -0.4506927
#> alpha  0.030386507  0.15935368 -0.016367942  -1.1955275
#> s     -0.006451792 -0.01636794  0.078801294   3.7481918
#> beta  -0.450692726 -1.19552752  3.748191828 191.4600192
```

Predicting Customer Behavior

Once the model parameters are estimated, we are able to predict future customer behavior on an individual level. To do so, we use `predict()` on the object with the estimated parameters (i.e. `est.pnbd`). The prediction period may be varied by specifying `prediction.end`. It is possible to provide either an end-date or a duration using the same time unit as specified when initializing the object (i.e `prediction.end = "2006-05-08"` or `prediction.end = 30`). By default, the prediction is made until the end of the dataset specified in the `clvdata()` command. The argument `continuous.discount.factor` allows to adjust the discount rate used to estimated the discounted expected transactions (DERT). The default value is `0.1` (=10%). Probabilistic customer attrition model predict in general three expected characteristics for every customer:

- "conditional expected transactions" (CET), which is the number of transactions to expect form 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.

If spending information was provided when initializing the object, `CLVTools` provides prediction for

- predicted spending estimated by a Gamma/Gamma model [@Colombo1999; @Fader2005c] and
- the customer lifetime value (CLV).

If a holdout period is available additionally the true numbers of transactions ("actual.x") and true spendings ("actual.spendings") during the holdout period are reported.

To use the parameter estimates on new data (e.g., an other customer cohort), the argument `newdata` optionally allows to provide a new `clvdata` object.

```
results <- predict(est.pnbd)
#> Predicting from 2005-10-11 until (incl.) 2006-07-16 (39.86 Weeks).
print(results)
#>        Id period.first period.last period.length actual.x actual.spending
#>   1:    1   2005-10-11  2006-07-16      39.85714        6         1410.75
#>   2:   10   2005-10-11  2006-07-16      39.85714        0            0.00
#>   3:  100   2005-10-11  2006-07-16      39.85714       10         2852.01
#>   4: 1000   2005-10-11  2006-07-16      39.85714       37         5874.07
#>   5: 1001   2005-10-11  2006-07-16      39.85714        0            0.00
#>  ---
```

```
#> 246: 1219   2005-10-11  2006-07-16      39.85714       18         3464.05
#> 247:  122   2005-10-11  2006-07-16      39.85714        2          130.37
#> 248: 1220   2005-10-11  2006-07-16      39.85714       18         2376.36
#> 249: 1221   2005-10-11  2006-07-16      39.85714        0            0.00
#> 250: 1222   2005-10-11  2006-07-16      39.85714        0            0.00
#>            PAlive          CET        DERT predicted.Spending predicted.CLV
#>   1: 0.3594502470  1.053881929 0.284919834           208.2820    59.3436681
#>   2: 0.2951895657  0.157038934 0.042455901           215.5326     9.1506296
#>   3: 0.9439761524 12.584749704 3.402321163           197.7055   672.6575408
#>   4: 0.9677557087 22.966151139 6.208961155           209.4217  1300.2912061
#>   5: 0.0019472323  0.010382371 0.002806902           211.3943     0.5933632
#>  ---
#> 246: 0.9668374243  6.701939589 1.811887517           224.7197   407.1667845
#> 247: 0.9584121734  5.876828766 1.588816572           202.5585   321.8282926
#> 248: 0.9985801580 10.916184535 2.951220051           208.3185   614.7935991
#> 249: 0.0005219825  0.006541308 0.001768460           228.7500     0.4045353
#> 250: 0.1166214301  0.248630971 0.067218056           209.9300    14.1110878
```

To change the duration of the prediction time, we use the `predicton.end` argument. We can either provide a time period (30 weeks in this example):

```
predict(est.pnbd, prediction.end = 30)
```

or provide a date indication the end of the prediction period:

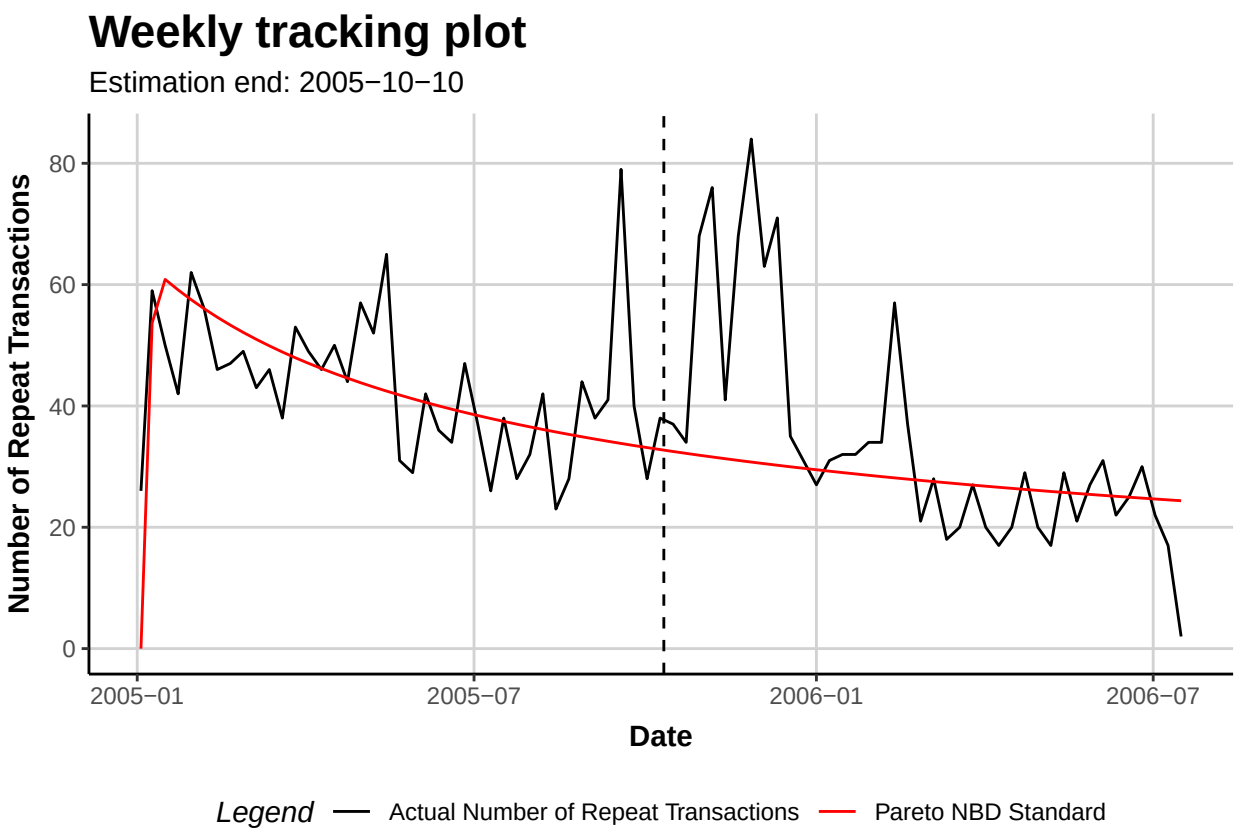
```
predict(est.pnbd, prediction.end = "2006-05-08")
```

Model Plotting

`clvdata` objects may be plotted using the `plot()` command. Similar to `summary()`, the output of `plot()` adapts on the current modeling step. It provides a descriptive plot of the actual transactional data if the model has not yet been fitted. Once the model has been estimated, `plot()` provides an aggregated incremental tracking plot of the actual data and the model based on the estimated parameters. The timespan for the plot may be altered using the `prediction.end` argument by providing either a duration or an end date. By default the plot is generated for the entire timespan of the provided dataset specified in the `clvdata()` command. The dashed line indicates the end of the estimation period. Alternatively cumulative actual and expected transactions can be plotted by setting `cumulative` to `TRUE`. The argument `transactions` disable for plotting actual transactions (`transactions=FALSE`). For further plotting options see the documentation

plot(x, prediction.end = NULL, cumulative = FALSE, transactions = TRUE, label = NULL, plot = TRUE, verbose = TRUE, ...)

```
plot(est.pnbd)
#> Plotting from 2005-01-03 until 2006-07-16.
```



To plot the *cumulative* expected transactions 30 time units (30 weeks in this example) ahead of the end of the estimation plot, we use:

```
plot(est.pnbd, prediction.end = 30, cumulative = TRUE)
```

Alternatively, it is possible to specify a date for the `prediction.end`argument. Note that dates are rounded to the next full time unit (i.e. week):

```
plot(est.pnbd, prediction.end = "2006-05-08", cumulative = TRUE)
```

Covariates

`CLVTools` provides the option to include covariates into probabilistic customer attrition models. Covariates may affect the purchase or the attrition process, or both. It is also possible to include different covariates for the two processes. However, support for covariates is dependent on the model. Not all implemented models provide the option for covariates. In general, `CLVTools` distinguishes between two types of covariates: time-invariant and time-varying. The former include factors that do not change over time such as customer demographics or customer acquisition information. The latter may change over time and include marketing activities or seasonal patterns.

Data for time-invariant covariates must contain a unique customer ID and a single value for each covariate. It should be supplied as a `data.frame` or `data.table`. In the example of the apparel retailer we use demographic information "gender" as time-invariant and information on the acquisition channel as covariate for both, the purchase and the attrition process. Use the `data("apparelStaticCov")` command to load the time-invariant covariates. In this example gender is coded as a dummy variable with `male=0` and `female=1` and channel with `online=0` and `offline=1`.

```
data("apparelStaticCov")
apparelStaticCov
#>        Id Gender Channel
```

```
#>   1:    1      0       0
#>   2:   10      0       0
#>   3:  100      1       0
#>   4: 1000      1       1
#>   5: 1001      1       0
#>  ---
#> 246: 1219      0       1
#> 247:  122      0       0
#> 248: 1220      0       0
#> 249: 1221      1       1
#> 250: 1222      1       0
```

Data for time-varying contextual factors requires a time-series of contextual factor values for every customer. I.e. if the time-varying contextual factors are allowed to change every week, a value for every customer for every week is required. Note that all contextual factors are required to use the same time intervals for the time-series. In the example of the apparel retailer we use information on direct marketing (`Marekting`) as time-varying contextual factors. Additionally, we add gender as time-invariant contextual factors. Note that the data structure of invariant contextual factors needs to be aligned with the structure of time-varying contextual factors. Use `data("apparelDynCov")` command to load the time-varying contextual factors. In this example `Cov.Date` indicates the first day of a weekly contextual factor time-interval.

```
data("apparelDynCov")
apparelDynCov
#>          Id   Cov.Date Marketing Gender Channel
#>     1:    1 2004-12-26         1      0       0
#>     2:    1 2005-01-02         1      0       0
#>     3:    1 2005-01-09         0      0       0
#>     4:    1 2005-01-16         1      0       0
#>     5:    1 2005-01-23         2      0       0
#>    ---
#> 20496: 1222 2006-06-18         0      1       0
#> 20497: 1222 2006-06-25         0      1       0
#> 20498: 1222 2006-07-02         0      1       0
#> 20499: 1222 2006-07-09         0      1       0
#> 20500: 1222 2006-07-16         0      1       0
```

To add the covariates to an initialized `clvdata` object the commands `SetStaticCovariates()` and `SetDynamicCovariates()` are available. The two commands are mutually exclusive. The argument `clv.data` specifies the initialized object and the argument `data.cov.life` respectively `data.cov.trans` specifies the data source for the covariates for the attrition and the purchase process. Covariates are added separately for the purchase and the attrition process. Therefore if a covariate should affect both processes it has to be added in both arguments: `data.cov.life` *and* `data.cov.trans`. The arguments `names.cov.life` and `names.cov.trans` specify the column names of the contextual factors for the two processes. In our example, we use the same covariates for both processes. Accordingly, we specify the time-invariant covariate "gender" as follows:

```
pnbd.static<- SetStaticCovariates(clv.data = clv.apparel,
                                  data.cov.life = apparelStaticCov,
                                  data.cov.trans = apparelStaticCov,
                                  names.cov.life = c("Gender", "Channel"),
                                  names.cov.trans =c("Gender", "Channel"),
                                  name.id = "Id")
```

To specify the time-varying contextual factors for seasonal patterns and direct marketing, we use the following:

```
pnbd.dyn <- SetDynamicCovariates(clv.data = clv.apparel,
                                 data.cov.life = apparelDynCov,
                                 data.cov.trans = apparelDynCov,
                                 names.cov.life = c("Marketing", "Gender", "Channel"),
                                 names.cov.trans = c("Marketing", "Gender", "Channel"),
                                 name.id = "Id",
                                 name.date = "Cov.Date")
```

In order to include time-invariant covariates in a time-varying model, they may be recoded as a time-varying covariate with a constant value in every time period.

Once the covariates are added to the model the estimation process is almost identical to the standard model without contextual factors. The only difference is that the provided object now data for contains either time-invariant or time-varying covariates and the option to define start parameters for the covariates of both processes using the arguments `start.params.life` and `start.params.trans`. If not set, the staring values are set to 1. To define starting parameters for the covariates, the name of the corresponding factor has to be used. For example in the case of time-invariant covariates:

```
est.pnbd.static <- pnbd(pnbd.static,
                        start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                        start.params.life = c(Gender=0.6, Channel=0.4),
                        start.params.trans = c(Gender=0.6, Channel=0.4))
#> Starting estimation...
#> Estimation finished!
```

To inspect the estimated model we use `summary()`, however all other commands such as `print()`, `coef()`, `loglike()`, `confint()` and `vcov()` are also available. Now, output contains also parameters for the covariates for both processes. Since covariates are added separately for the purchase and the attrition process, there are also separate model parameters for the two processes. These parameters are directly interpretable as rate elasticity of the corresponding factors: A 1% change in a contextual factor $\mathbf{X^P}$ or $\mathbf{X^L}$ changes the purchase or the attrition rate by $\gamma_{purch}\mathbf{X^P}$ or $\gamma_{life}\mathbf{X^L}$ percent, respectively [@Gupta1991]. In the example of the apparel retailer, we observe that female customer significantly purchase more (`trans.Gender=1.1772`) and customer acquired offline purchase slightly more(`trans.Channel=0.5996`). Before interpreting the parameters, check the Karush-Kuhn-Tucker optimality conditions of the first and second order [@KKT] (KKT1 and KKT1). If those criteria are not met, the optimizer has probably not arrived at an optimal solution. If this is the case it is usually a good idea to rerun the estimation using alternative starting values.

```
summary(est.pnbd.static)
#> Pareto NBD with Static Covariates  Model
#>
#> Call:
#> suppressPackageStartupMessages({
#>     oldLC <- Sys.getlocale(category = "LC_COLLATE")
#>     Sys.setlocale(category = "LC_COLLATE", locale = "C")
#>     on.exit(Sys.setlocale(category = "LC_COLLATE", locale = oldLC))
#>     devtools::document(roclets = c("rd", "collate", "namespace",
#>         "vignette"))
#> })
#>
#> Fitting period:
#> Estimation start  2005-01-03
#> Estimation end    2005-10-10
#> Estimation length 40.0000 Weeks
#>
#> Coefficients:
```

```
#>                Estimate Std. Error  z-val Pr(>|z|)
#> r                0.9134     0.1528  5.977 2.28e-09 ***
#> alpha           12.5136     2.7492  4.552 5.32e-06 ***
#> s                0.5509     0.2545  2.164 0.030436 *
#> beta            25.7751    25.0566  1.029 0.303632
#> life.Gender      0.7941     0.5852  1.357 0.174826
#> life.Channel    -0.3323     0.3832 -0.867 0.385758
#> trans.Gender     1.1772     0.1874  6.280 3.38e-10 ***
#> trans.Channel    0.5996     0.1569  3.821 0.000133 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Optimization info:
#> LL      -3481.6392
#> AIC     6979.2784
#> BIC     7007.4501
#> KKT 1   TRUE
#> KKT 2   TRUE
#> fevals 41.0000
#> Method L-BFGS-B
#>
#> Used Options:
#> Correlation     FALSE
#> Regularization  FALSE
#> Constraint covs FALSE
```

To predict future customer behavior we use `predict()`. Time-varying covariates have to be available for the entire prediction prediction period. If the data initially provided in the `SetDynamicCovariates()` command does not cover the complete prediction period, the argument `new.data` offers the possibility to supply new data for the time-varying covariates in the from of a `clvdata` object.

Add Correlation to the model

To relax the assumption of independence between the purchase and the attrition process, `CLVTools` provides the option to specify the argument `use.cor` in the command t fit the model (i.e. `pnbd`). In case of `use.cor=TRUE`, a Sarmanov approach is used to correlate the two processes. `start.param.cor` allows to optionally specify a starting value for the correlation parameter.

```
est.pnbd.cor <- pnbd(clv.apparel,
                     use.cor= TRUE)
summary(est.pnbd.cor)
```

The parameter `Cor(life,trans)` is added to the parameter estimates that may be directly interpreted as a correlation. In the example of the apparel retailer the correlation parameter is not significant and the correlation is very close to zero, indicating that the purchase and the attrition process are independent.

Advanced Options for Contextual Factors

`CLVTools` provides two additional estimation options for models containing covariates (time-invariant or time-varying): regularization and constraints for the parameters of the covariates. Both options are included in the command to fit the model (i.e., `pnbd()`. Support for this option is dependent on the model. They may be used simultaneously.

- The argument `reg.lambdas` provides the possibility to specify separate `\lambda_{reg}` for the two processes (i.e. `reg.lambdas = c(trans=100, life=100)`. The larger the `\lambda_{reg}` the stronger

the effects of the regularization. Regularization only affects the parameters of the covariates.

- The argument `names.cov.constr` implements equality constraints for contextual factors with regards to the two processes. For example the variable "gender" is forced to have the same effect on the purchase as well as on the attrition process. To do so, the option `names.cov.constr` is available (i.e. `names.cov.constr=c("Gender")`). To provide starting parameters for the constrained variable use `start.params.constr`.

To enable regularization for the covariates, we use the following command:

```
est.pnbd.reg <- pnbd(pnbd.static,
                         start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                         reg.lambdas = c(trans=100, life=100))
#> Starting estimation...
#> Estimation finished!
summary(est.pnbd.reg)
#> Pareto NBD with Static Covariates  Model
#>
#> Call:
#> suppressPackageStartupMessages({
#>     oldLC <- Sys.getlocale(category = "LC_COLLATE")
#>     Sys.setlocale(category = "LC_COLLATE", locale = "C")
#>     on.exit(Sys.setlocale(category = "LC_COLLATE", locale = oldLC))
#>     devtools::document(roclets = c("rd", "collate", "namespace",
#>         "vignette"))
#> })
#>
#> Fitting period:
#> Estimation start  2005-01-03
#> Estimation end    2005-10-10
#> Estimation length 40.0000 Weeks
#>
#> Coefficients:
#>                  Estimate Std. Error  z-val Pr(>|z|)
#> r              6.676e-01  1.477e+00  0.452    0.651
#> alpha          2.632e+00  6.346e+00  0.415    0.678
#> s              5.553e-01  4.429e+00  0.125    0.900
#> beta           1.720e+01  2.177e+02  0.079    0.937
#> life.Gender    9.254e-06  7.071e-02  0.000    1.000
#> life.Channel  -7.075e-05  7.071e-02 -0.001    0.999
#> trans.Gender   4.714e-04  7.070e-02  0.007    0.995
#> trans.Channel  4.458e-04  7.069e-02  0.006    0.995
#>
#> Optimization info:
#> LL     -14.0260
#> AIC    44.0520
#> BIC    72.2237
#> KKT 1  TRUE
#> KKT 2  TRUE
#> fevals 105.0000
#> Method L-BFGS-B
#>
#> Used Options:
#> Correlation     FALSE
#> Regularization  TRUE
#>    lambda.life  100.0000
```

```
#>    lambda.trans 100.0000
#> Constraint covs FALSE
```

To constrain "Gender" to have the same effect on both processes we use the following command. Note, that the output now only contains one parameter for "Gender" as it is constrained to be the same for both processes.

```
est.pnbd.constr <- pnbd(pnbd.static,
                        start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                        start.params.constr = c(Gender=0.6),
                        names.cov.constr=c("Gender"))
#> Starting estimation...
#> Estimation finished!
summary(est.pnbd.constr)
#> Pareto NBD with Static Covariates  Model
#>
#> Call:
#> suppressPackageStartupMessages({
#>     oldLC <- Sys.getlocale(category = "LC_COLLATE")
#>     Sys.setlocale(category = "LC_COLLATE", locale = "C")
#>     on.exit(Sys.setlocale(category = "LC_COLLATE", locale = oldLC))
#>     devtools::document(roclets = c("rd", "collate", "namespace",
#>         "vignette"))
#> })
#>
#> Fitting period:
#> Estimation start  2005-01-03
#> Estimation end    2005-10-10
#> Estimation length 40.0000 Weeks
#>
#> Coefficients:
#>               Estimate Std. Error  z-val Pr(>|z|)
#> r               0.8937     0.1448  6.171 6.79e-10 ***
#> alpha          12.4534     2.7623  4.508 6.53e-06 ***
#> s               0.5453     0.2470  2.207  0.02728 *
#> beta           36.3180    30.4110  1.194  0.23238
#> life.Channel   -0.3416     0.3945 -0.866  0.38663
#> trans.Channel   0.6038     0.1578  3.826  0.00013 ***
#> constr.Gender   1.1844     0.1909  6.204 5.50e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Optimization info:
#> LL     -3481.8508
#> AIC    6977.7017
#> BIC    7002.3519
#> KKT 1  TRUE
#> KKT 2  TRUE
#> fevals 36.0000
#> Method L-BFGS-B
#>
#> Used Options:
#> Correlation          FALSE
#> Regularization       FALSE
```

```
#> Constraint covs      TRUE
#>    Constraint params Gender
```

Literature