Customer Scoring with BTYDplus

Michael Platzer

2016-09-28

Introduction

The BTYDplus package provides advanced statistical methods to describe and predict customers' purchase behavior in noncontractual setting. It fits probabilistic models to historic transaction records for computing customer-centric metrics of managerial interest.

The challenge of this task is threefold: For one, the churn event in a non-contractual customer relationship is not directly observable, but needs to be infered indirectly based on observed periods of inactivity. Second, with customers behaving differently, yet having oftentimes only few transactions recorded so far, we require statistical methods that can utilize cohort-level patterns as priors for estimating customer-level quantities. And third, we attempt to predict the (unseen) future, thus need assumptions regarding the future dynamics.

Figure 1 displays the complete transaction records of 30 sampled customers of an online grocery store. Each horizontal line represents a customer, and each circle a purchase event. The typical questions that arise are:

- How many customers does the firm still have?
- How many customers will be active in one year from now?
- How many transactions can be expected in next X weeks?
- Which customers can be considered to have churned?
- Which customers will provide the most value to the company going forward?

```
library(BTYDplus)
data("groceryElog")
set.seed(123)
# plot timing patterns of 30 sampled customers
plotTimingPatterns(groceryElog, n = 30, T.cal = "2007-05-15",
                   headers = c("Past", "Future"), title = "")
```

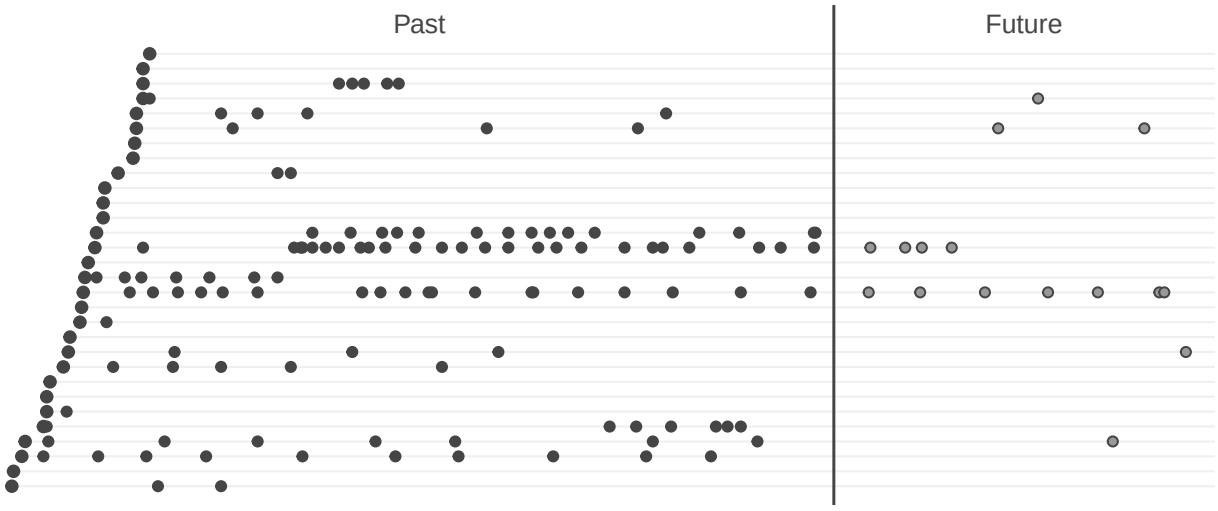


Figure 1: Timing Patterns for Sampled Grocery Customers

Fitting a buy-till-you-die model to a particular customer cohort not just allows analysts to describe it in terms of its heterogeneous distribution of purchase patterns and dropout probabilities, but also provides answers for all of the above stated questions. On aggregated level the estimated number of future transactions can then be, for example, used for capacity and production planning. The estimated future value of the cohort for assessing the return on investment for customer acquistion spends. On individual level the customer database can be enriched with estimates on a customer's status, future activity and future value. customer scores like these can be then utilized to adapt services, messages and offers with respect to customers' state and value. Given the accessibility and speed of the provided models, practitioners can score their customer base with these advanced statistical techniques on a continuous basis.

Models

The BTYD package already provides implementations for the **Pareto/NBD** (Schmittlein, Morrison, and Colombo (1987)), the **BG/NBD** (P. Fader, Hardie, and Lee (2005)) and the **BG/BB** (Fader, Hardie, and Shang (2010)) model. BTYDplus complements that package by providing several additional buy-till-you-die models, that have been published in the marketing literature, but whose implementation are complex and non-trivial. In order to create a consistent experience of users of both packages, the BTYDplus adopts method signatures from BTYD where possible.

The models provided as part of BTYDplus are:

- **NBD** - Ehrenberg (1959)
- **MBG/NBD** - Batislam, Denizel, and Filiztekin (2007)
- **BG/CNBD-k** - Platzer and Reutterer (n.d.)
- **MBG/CNBD-k** - Platzer and Reutterer (n.d.)
- **Pareto/NBD (HB)** - Ma and Liu (2007)
- **Pareto/NBD (Abe)** - Abe (2009)
- **Pareto/GGG** - Platzer and Reutterer (2016)

The number of implemented models raises the question, which one to use, and which one works best in a particular case. There is no simple answer to that, but practitioners could consider trying out all of them for a given dataset, assess data fit, calculate forecast accuracy based on an artificially withheld time period and then make a tradeoff between calculation speed, data fit and accuracy.

The implementation of the original *NBD* model from 1959 serves mainly as a basic benchmark. It assumes a heterogenous purchase process, but doesn't account for the possibility of a customer churning. The *Pareto/NBD* model, introduced in 1987, combines the NBD model for transactions of active customers with a heterogeneuos dropout process, and to this date can still be considered a gold standard for buy-till-you-die models. The *BG/NBD* model adjusts the Pareto/NBD assumptions with respect to the dropout process in order to speed up computation. It is able to retain a similar level of data fit and forecast accuracy, but also improves the robustness of the parameter search. However, the BG/NBD model particularly assumes that every customer without a repeat transaction has *not* churned yet, independent of the elapsed time of inactivity. This seems counterintuitive, particular when compared to customers with repeat transactions. Thus the *MBG/NBD* has been developed to eliminate this inconsistency by allowing customers without any activity to also remain inactive. Data fit and forecast accuracy are comparable to BG/NBD, yet it results in more plausible estimates for the dropout process. The more recently developed *BG/CNBD-k* and *MBG/CNBD-k* model classes extend BG/NBD and MBG/NBD each but allow for regularity within the transaction timings. If such regularity is present (even in a mild form), these models can yield significant improvements in terms of customer level forecasting accuracy, while the computational costs remain at a similar order of magnitude.

All of the aforementioned models benefit from closed-form solutions for key expressions and thus can be efficiently estimated via means of maximum likelihood estimation (MLE). However, the necessity of deriving closed-form expressions restricts the model builder from relaxing the underlying behavioral assumptions. An

alternative estimation method for probabilistic models is via Markov-Chain-Monte-Carlo (MCMC) simulation. MCMC comes at significantly higher costs in terms of implementation complexity and computation time, but it allows for more flexible assumptions. Additionally one gains the benefits of (1) estimated marginal posterior distributions rather than point estimates, (2) individual-level parameter estimates, and thus (3) straightforward simulations of customer-level metrics of managerial interest. The hierarchical bayes variant of Pareto/NBD (i.e., *Pareto/NBD (HB)*) served as a proof-of-concept for the MCMC approach, but doesn't yet take advantage of the gained flexibility, as it sticks to the original Pareto/NBD assumptions. In contrast, Abe's variant of the Pareto/NBD (termed here *Pareto/NBD (Abe)*) relaxes the independence of purchase and dropout process, plus is capable of incorporating customer covariates. Particularly the latter can turn out to be very powerful, if any of such known covariates helps in explaining the heterogeneity within the customer cohort. Finally, the *Pareto/GGG* is another generalization of Pareto/NBD, which allows for a varying degree of regularity within the transaction timings. Analogous to (M)BG/CNBD-k, incorporating regularity can yield significant improvements in forecast accuracy, if such regularity is present in the data.

Analytical Workflow

The typical analysis process starts out by reading in a complete log of all events or transactions of an existing customer cohort. It is up to the analyst to define how a customer base is split into cohorts, but typically these are defined based on customers' first transaction date and/or the acquisition channel. The data requirements for such an event log are minimal, and only consist of a customer identifier field `cust`, and a `date` field of class `Date` or `POSIXt`. If the analysis should also cover the monetary component, the event log needs to contain a corresponding field `sales`. In order to get started quickly, BTYDplus provides an event log for customers of an online grocery store (`data("groceryElog")`). Further, for each BTYDplus model data generators are available (`*.GenerateData`), which allow to create artificial transaction logs, that follow the assumptions of a particular model.

Table 1: Transaction Log Example

cust	date	sales
4	1997-01-01	29.33
4	1997-01-18	29.73
4	1997-08-02	14.96
4	1997-12-12	26.48
21	1997-01-01	63.34
21	1997-01-13	11.77

Once the transaction log has been obtained, it needs to be converted into a customer-by-sufficient-statistic summary table (via the `elog2cbs` method), so that the data can be consumed by model-specific parameter estimation methods (`*.EstimateParameters` for MLE- and `*.DrawParameters` for MCMC-models). The estimated parameters already provide insights regarding the purchase and dropout process, e.g. mean purchase frequency, mean lifetime, variation in dropout probability, etc. For MLE-estimated models we can further report the maximized log-likelihood (via the `*.cbs.LL` methods) to benchmark the models in terms of their data fit to a particular dataset. Further, estimates for the conditional and unconditional expected number of transactions (`*.pmf`, `*.Expectation`, `*.ConditionalExpectedTransactions`), as well as for the (unobservable) status of a customer (`*.PAlive`) can be computed based on the parameters. Such estimates can then be analyzed either on individual level, or be aggregated on to cohort level.

Helper Methods

BTYDplus provides various model-independent helper methods for handling and describing customers' transaction logs.

Convert Event Log to Weekly Transactions

Before starting to fit probabilistic models, an analyst might be interested in reporting the total number of transactions over time, to gain a first understanding of the dynamics at a cohort level. For this purpose the methods `elog2cum` and `elog2inc` are provided. These take an event log as a first argument, and count for each time unit the cumulated or incremental number of transactions. If argument `first` is set to TRUE, then a customer's initial transaction will be included, otherwise not.

```
data("groceryElog")
op <- par(mfrow = c(1, 2), mar = c(2.5, 2.5, 2.5, 2.5))
# incremental
weekly_inc_total  <- elog2inc(groceryElog, by = 7, first = TRUE)
weekly_inc_repeat <- elog2inc(groceryElog, by = 7, first = FALSE)
plot(weekly_inc_total, typ = "l", frame = FALSE, main = "Incremental")
lines(weekly_inc_repeat, col = "red")
# cumulative
weekly_cum_total  <- elog2cum(groceryElog, by = 7, first = TRUE)
weekly_cum_repeat <- elog2cum(groceryElog, by = 7, first = FALSE)
plot(weekly_cum_total, typ = "l", frame = FALSE, main = "Cumulative")
lines(weekly_cum_repeat, col = "red")
par(op)
```

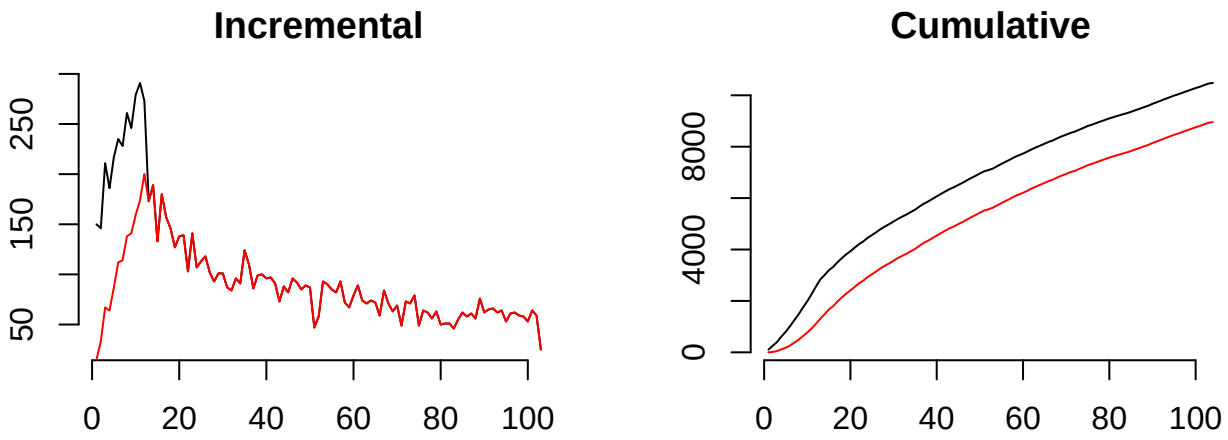


Figure 2: Weekly Trends

Convert Transaction Log to CBS format

The `elog2cbs` method is an efficient implementation for the conversion of an event log into a customer-by-sufficient-statistic (CBS) `data.frame`, with a row for each customer. This is the required data format for estimating model parameters.

```
data("groceryElog")
cbs <- elog2cbs(groceryElog)
head(cbs, 5)
#>   cust  x      t.x      litt      first T.cal
#> 1    1  0  0.00000  0.000000 2006-01-01   104
#> 2    2  1 50.28571  3.917721 2006-01-01   104
#> 3    3 33 99.28571 32.887165 2006-01-01   104
```

```
#> 4    4  0  0.00000  0.000000 2006-01-01   104
#> 5    5  5 89.85714 14.353181 2006-01-01   104
```

Returned field `cust` is the unique customer identifier, `x` the number of repeat transactions, `t.x` the time of the last recorded transaction, `litt` the sum over logarithmic intertransaction times (required for estimating regularity), `first` the date of the first transaction, and `T.cal` the duration between the first transaction and the end of the calibration period. If the provided `elog` data.frame contains a field `sales`, then this will be summed up, and returned as an additional field, named `sales`. Note, that transactions with identical `cust` and `date` field are counted as a single transaction, but with `sales` being summed up.

The time unit for expressing `t.x`, `T.cal` and `litt` are determined via the argument `units`, which is passed forward to method `difftime`, and defaults to `weeks`.

Argument `T.tot` allows one to specify the end of the observation period, i.e., the last possible date of an event to still be included in the event log. If `T.tot` is not provided, then the date of the last recorded event will be assumed to coincide with the end of the observation period. If `T.tot` is provided, then any event that occurs after that date is discarded.

Argument `T.cal` allows one to calculate the summary statistics for a calibration and a holdout period separately. This is particularly useful for evaluating forecasting accuracy for a given dataset. If `T.cal` is not provided, then the whole observation period is considered, and is then subsequently used for for estimating model parameters. If it is provided, then the returned `data.frame` contains two additional fields, with `x.star` representing the number of repeat transactions during the holdout period of length `T.star`. And only those customers are contained, who have had at least one event during the calibration period.

```
data("groceryElog")
range(groceryElog$date)
#> [1] "2006-01-01" "2007-12-30"
cbs <- elog2cbs(groceryElog, T.cal = "2006-12-31")
head(cbs, 5)
#>   cust  x      t.x      litt      first T.cal T.star x.star
#> 1    1  0  0.00000  0.000000 2006-01-01    52     52      0
#> 2    2  1 50.28571  3.917721 2006-01-01    52     52      0
#> 3    3 19 48.57143 16.952179 2006-01-01    52     52     14
#> 4    4  0  0.00000  0.000000 2006-01-01    52     52      0
#> 5    5  2 40.42857  6.012667 2006-01-01    52     52      3
```

Estimate Regularity

The models BG/CNBD-k, MBG/CNBD-k and Pareto/GGG are capable of leveraging regularity within transaction timings for improving forecast accuracy. The method `estimateRegularity` provides a quick check for the degree of regularity in the event timings. A return value of close to 1 supports the assumption of exponentially distributed intertransaction times, whereas values significantly larger than 1 reveal the presence of regularity. Estimation is either done by 1) assuming a same degree of regularity across all customers (`method = "wheat"`), or 2) by estimating regularity for each customer separately, as the shape parameter of a fitted gamma distribution, and then return the median across estimates. The latter methods, though, require sufficient (≥ 10) transactions per customer.

Wheat and Morrison (1990)'s method calculates for each customer a statistic M based on her last two number of intertransaction times as $M := \text{ITT}_1/(\text{ITT}_1 + \text{ITT}_2)$. That measure is known to follow a Beta(k, k) distribution, if the intertransaction times of customers follow Gamma(k, λ) with a shared k but potentially varying λ, and k can then be estimated as $(1 - 4 \cdot Var(M))/(8 \cdot Var(M))$. The corresponding diagnostic plot shows the actual distribution of M vs. the theoretical distribution for Exponential, respectively for Erlang-2 distributed ITTs.

```
data("groceryElog")
op <- par(mfrow = c(1, 2))
(k.wheat <- estimateRegularity(groceryElog, method = "wheat",
                               plot = TRUE, title = "Wheat & Morrison"))
#> [1] 1.82444
(k.mle <- estimateRegularity(groceryElog, method = "mle",
                             plot = TRUE, title = "Maximum Likelihood"))
#> [1] 3.262912
par(op)
```

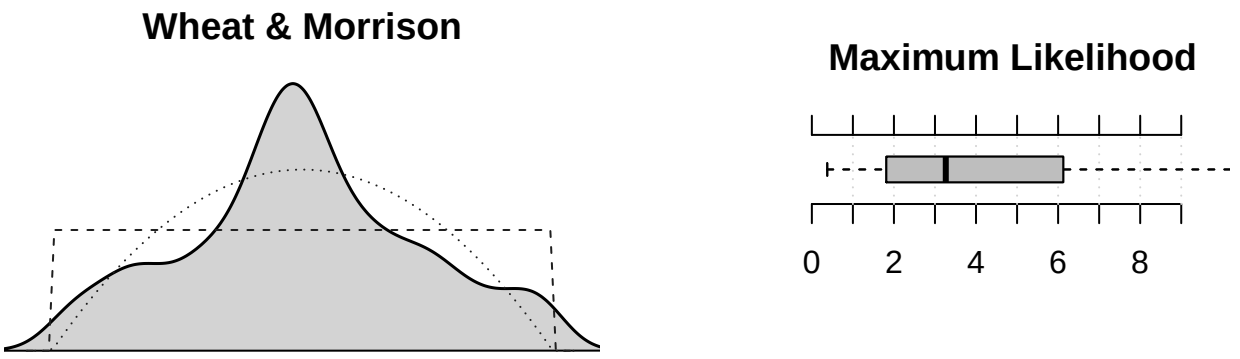


Figure 3: Diagnostic Plots for Estimating Regularity

Applied to the online grocery dataset the Wheat & Morrison estimator reports a regularity estimate of close to 2, suggesting that a Erlang-2 might be more appropriate than the exponential distribution for modelling intertransaction times in this case. The peak in the plotted distribution additionally hints at a subset of customers which are exhibiting an even stronger degree of regularity.

The maximum likelihood estimation method fits separate gamma distributions to the intertransaction times of each customer with more than 10 events. The reported median estimate of k=3.26 also indicate stronger degrees of regularity for this subset of highly active customers. The boxplot then gives a deeper understanding of the distribution of `k` estimates, revealing a heterogeneity within regularity across the cohort, thus suggesting that this dataset is a good candidate for the Pareto/GGG model.

Maximum Likelihood Estimated Models

NBD

The NBD model by Ehrenberg (1959) assumes a heterogenous, yet constant purchasing process, with expontentially distributed intertransaction times, whereas its purchase rate λ is $Gamma(r, \alpha)$-distributed across customers.

Fitting the model requires converting the event log first to a CBS format and passing the dataset to `nbd.EstimateParameters`. The method searches (by using `stats::optim`) for that pair of (r, α) heterogeneity parameters, that maximizes the log-likelihood function (`nbd.cbs.LL`) given the data.

```
data("groceryElog")
cbs <- elog2cbs(groceryElog, T.cal = "2006-12-31")
round(params.nbd <- nbd.EstimateParameters(cbs), 3)
#>     r alpha
#> 0.420 5.245
nbd.cbs.LL(params.nbd, cbs)
#> [1] -16376.86
```

With the mean of the Gamma distribution being r/α, the mean estimate for λ is 0.08, which translates to a mean intertransaction time of $1/\lambda$ of 12.48 weeks.

The expected number of (future) transactions for a customer, conditional on her past (`x` and `T.cal`), can be computed with `nbd.ConditionalExpectedTransactions`. By passing the whole CBS we can easily generate estimates for all customers in the cohort.

```
# predict customers who've had 1 to 5 transactions in first 52 weeks
est <- nbd.ConditionalExpectedTransactions(
         params.nbd, T.star = 52, x = 1:5, T.cal = 52)
names(est) <- 1:5
round(est, 3)
#>     1     2     3     4     5
#> 1.290 2.199 3.107 4.015 4.924

# predict whole customer cohort
cbs$x.star.nbd <- nbd.ConditionalExpectedTransactions(
                    params.nbd, T.star = 52, cbs$x, cbs$T.cal)
# compare predictions with actuals at aggregated level
c(`Actuals` = sum(cbs$x.star),
  `NBD`     = sum(cbs$x.star.nbd))
#>  Actuals      NBD
#> 3389.000 6354.746
```

As can be seen, the NBD model heavily overforecasts the actual number of transactions (by 87.5%), which can be explained by the lack of a dropout process in the model assumptions. All customers are assumed to remain just as active in the second year, as they have been in their first year. However, figure 2 shows clearly a downward trend in the incremental transaction counts for the online grocery customers, thus mandating a different model.

Pareto/NBD

The Pareto/NBD model (Schmittlein, Morrison, and Colombo 1987) combines the NBD model with the possibility of customers becoming inactive. A customer's state, however, is not directly observable, and the model needs to draw inferences based on the observed elapsed time since a customer's last activity, i.e., `T.cal - t.x`. Pareto/NBD in particular assumes a customer's lifetime τ to be exponential distributed with parameter μ, whereas μ is Gamma(s, β)-distributed across customers.

The Pareto/NBD implementation is part of the BTYD package, but the workflow of fitting the model and making predictions is analogous to BTYDplus (respectively vice versa).

```
data("groceryElog")
cbs <- elog2cbs(groceryElog, T.cal = "2006-12-31")
params.pnbd <- BTYD::pnbd.EstimateParameters(cbs)
names(params.pnbd) <- c("r", "alpha", "s", "beta")
round(params.pnbd, 3)
#>     r alpha     s  beta
#> 0.786 5.679 0.386 5.717
BTYD::pnbd.cbs.LL(params.pnbd, cbs)
#> [1] -15782.39
```

For one, we can note, that the maximized log-likelihood of Pareto/NBD is higher than for the NBD model, implying that its data fit is better. And second, by estimating a mean lifetime $1/(\beta/s)$ of 14.8 weeks, the estimated mean intertransaction times change from 12.48 to 7.22 weeks, when compared to NBD.

Let's now again compute the conditional expected transactions for six customers with an increasing number of observed transactions, but all with an observed overly long period of recent inactivity.

```
# predict customers who've had 1 to 5 transactions in first 12 weeks,
# but then remained inactive for 40 weeks
est <- BTYD::pnbd.ConditionalExpectedTransactions(
         params.pnbd, T.star = 52, x = 1:5, t.x = 12, T.cal = 52)
names(est) <- 1:5
round(est, 3)
#>     1     2     3     4     5
#> 0.448 0.363 0.217 0.109 0.049
```

```
# predict whole customer cohort
cbs$x.star.pnbd <- BTYD::pnbd.ConditionalExpectedTransactions(
                      params.pnbd, T.star = 52, cbs$x, cbs$t.x, cbs$T.cal)
# compare predictions with actuals at aggregated level
c(`Actuals`    = sum(cbs$x.star),
  `Pareto/NBD` = sum(cbs$x.star.pnbd))
#>    Actuals Pareto/NBD
#>   3389.000   3990.767
```

As expected, the Pareto/NBD yields overall lower and thus more realistic estimates than the NBD. However, the results also reveal an interesting pattern, which might seem at first sight counter intuitive. Customers with a very active purchase history receive lower estimates than customers which have been less active in the past. P. S. Fader, Hardie, and Lee (2005) discuss this apparent paradox in more detail, yet the underlying mechanism can be easily explained by looking at the model's assessment of the latent activity state.

```
# P(alive) for customers who've had 1 to 5 transactions in first 12 weeks,
# but then remained inactive for 40 weeks
palive <- BTYD::pnbd.PAlive(params.pnbd, x = 1:5, t.x = 12, T.cal = 52)
names(palive) <- 1:5
round(palive, 3)
#>     1     2     3     4     5
#> 0.319 0.165 0.073 0.029 0.011
```

The probability of still being alive after a 40 week purchase hiatus drops from 31.9% for the one-time-repeating customer to 1.1% for the customer which has had already 5 transactions. The elapsed time of inactivity is a stronger indication of churn for the highly frequent than for the less frequent purchasing customer, as a low purchase frequency also allows for the possibility of such long intertransaction times as the observed 40 weeks.

BG/CNBD-k and MBG/CNBD-k

The BG/NBD (P. Fader, Hardie, and Lee 2005) and the MBG/NBD (Batislam, Denizel, and Filiztekin 2007) models are contained in the larger class of (M)BG/CNBD-k models (Platzer and Reutterer, n.d.), and thus presented here together in this section. The MBG/CNBD-k model assumptions are as follows: A customer's intertransaction times, while being active, are Erlang-k distributed, with purchase rate λ being Gamma(r, α)-distributed across customers. After each transaction a customer can become inactive (for good) with a constant dropout probability of p, whereas p is Beta(a, b)-distributed across customers. The BG/CNBD-k only differs in that respect, that the customer is not allowed to drop out at the initial transaction, but only at repeat transactions.

```
data("groceryElog")
cbs <- elog2cbs(groceryElog, T.cal = "2006-12-31")
params.bgnbd   <- BTYD::bgnbd.EstimateParameters(cbs) # BG/NBD
params.bgcnbd  <- bgcnbd.EstimateParameters(cbs)      # BG/CNBD-k
params.mbgnbd  <- mbgnbd.EstimateParameters(cbs)      # MBG/NBD
params.mbgcnbd <- mbgcnbd.EstimateParameters(cbs)     # MBG/CNBD-k
rbind(`BG/NBD` = c(k=1, round(params.bgnbd, 3),  LL = BTYD::bgnbd.cbs.LL(params.bgnbd, cbs)),
      `BG/CNBD-k` = c(round(params.bgcnbd, 3), LL = bgcnbd.cbs.LL(params.bgcnbd, cbs)),
      `MBG/NBD` = c(round(params.mbgnbd, 3), LL = mbgcnbd.cbs.LL(params.mbgnbd, cbs)),
      `MBG/CNBD-k` = c(round(params.mbgcnbd, 3), LL = mbgcnbd.cbs.LL(params.mbgcnbd, cbs)))
#>             k                                LL
#> BG/NBD      1 0.349 2.581 0.504 3.644 -15836.57
#> BG/CNBD-k   2 0.359 1.129 0.523 2.785 -15054.17
#> MBG/NBD     1 1.257 6.038 0.368 0.716 -15781.94
#> MBG/CNBD-k  2 1.329 2.813 0.425 0.790 -14978.21
```

The MLE method searches across a five dimensional parameter space (k, r, α, a, b) to find the optimum of the log-likelihood function. As can be seen from the reported log-likelihood values, the MBG/CNBD-k is able to provide a better fit than NBD, Pareto/NBD, BG/NBD, MBG/NBD and BG/CNBD-k for the given dataset. Further, the estimate for regularity parameter k is 2 and implies that regularity is present, and that Erlang-2 is deemed more suitable for the intertransaction times than the exponential distribution ($k = 1$).

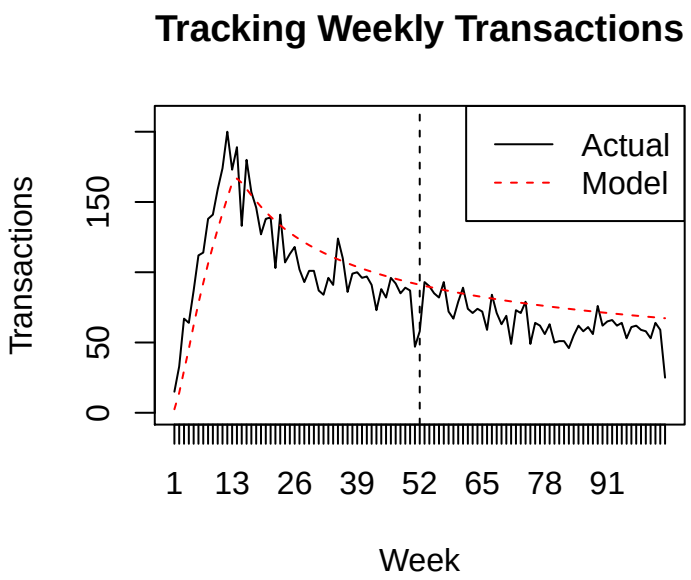
```
# predict customers who've had 1 to 5 transactions in first 12 weeks,
# but then remained inactive for 40 weeks
est <- mbgcnbd.ConditionalExpectedTransactions(
         params.mbgcnbd, T.star = 52, x = 1:5, t.x = 12, T.cal = 52)
names(est) <- 1:5
round(est, 3)
#>     1     2     3     4     5
#> 0.282 0.075 0.013 0.002 0.000

# P(alive) for customers who've had 1 to 5 transactions in first 12 weeks,
# but then remained inactive for 40 weeks
palive <- mbgcnbd.PAlive(params.mbgcnbd, x = 1:5, t.x = 12, T.cal = 52)
names(palive) <- 1:5
round(palive, 3)
#>     1     2     3     4     5
#> 0.156 0.029 0.004 0.000 0.000
```

Predicting transactions for 5 assumed customers, each with a long purchase hiatus but with a varying number of past transactions, we see the same pattern as for Pareto/NBD, except that the forecasted numbers are even lower. This results from the long period of inactivity being now, in the presence of regularity, an even stronger indiciation for churn, as the Erlang-2 allows for less variation in the intertransaction times.

```
# predict whole customer cohort
cbs$x.star.mbgcnbd <- mbgcnbd.ConditionalExpectedTransactions(
                        params.mbgcnbd, T.star = 52, cbs$x, cbs$t.x, cbs$T.cal)
# compare predictions with actuals at aggregated level
c(`Actuals`    = sum(cbs$x.star),
  `MBG/CNBD-k` = sum(cbs$x.star.mbgcnbd))
#>    Actuals MBG/CNBD-k
#>   3389.000   3970.013
```

Comparing the predictions at an aggregate level, we see that also the MBG/CNBD-k remains overly optimistic for the online grocery dataset. This can also be visualized with the help of `mbgcnbd.PlotTrackingInc`.

```
nil <- mbgcnbd.PlotTrackingInc(params.mbgcnbd,
         T.cal = cbs$T.cal,
         T.tot = max(cbs$T.cal + cbs$T.star),
         actual.inc.tracking = elog2inc(groceryElog))
```



However, when assessing the error at individual level, by calculating mean absolute error (MAE) of our predictions, we see a significant improvement in forecast accuracy. In particular considering that the data requirements have not increased, and still only leverage the historic event log.

```
cbs$x.star.pnbd <- BTYD::pnbd.ConditionalExpectedTransactions(
                      params = BTYD::pnbd.EstimateParameters(cbs),
                      T.star = 52, cbs$x, cbs$t.x, cbs$T.cal)
mae <- function(act, est) sum(abs(act-est)) / sum(act) # mean absolute error
c(`MAE Pareto/NBD` = mae(cbs$x.star, cbs$x.star.pnbd),
  `MAE MBG/CNBD-k` = mae(cbs$x.star, cbs$x.star.mbgcnbd))
#> MAE Pareto/NBD MAE MBG/CNBD-k
#>      0.6785325      0.6439486
lift <- 1 - mae(cbs$x.star, cbs$x.star.mbgcnbd) / mae(cbs$x.star, cbs$x.star.pnbd)
cat("Lift in MAE:", round(100*lift, 1), "%")
#> Lift in MAE: 5.1 %
```

MCMC Estimated Models

This chapter presents three buy-till-you-die model variants which rely on Markov-Chain-Monte-Carlo simulation for parameter estimation. Implementation complexity as well as computational costs are significantly higher, and despite an efficient MCMC implementation in C++ applying these models requires much longer compute time when compared to the before presented ML-estimated models. On the upside, we gain flexibility in our model assumptions, and get estimated distributions, even for individual-level parameters. Thus, the return object for parameter estimation (`param.draws <- *.mcmc.DrawParameters(...)`) not only returns the point estimates of the heterogeneity parameters (`params <- *.EstimateParameters(...)` did), but provides samples from the marginal posterior distributions, both at cohort- (`param.draws$level_1`)

as well as on customer-level (`param.draws$level_2`). Based on these parameter draws, we can then easily sample the posterior distributions of any derived quantity, for example the number of transactions (`mcmc.DrawFutureTransactions`) or the probability of being active in a given period.

Generally speaking, MCMC works by constructing a Markov chain which has the desired target (posterior) distribution as its equilibrium distribution. The algorithm then performs random walks on that Markov chain and will eventually (after some "burnin" phase) produce draws from the posterior. In order to assess MCMC convergence one can run multiple MCMC chains (in parallel) and check whether these provide similar distributions. Due to the high auto-correlation between subsequent iteration steps in the MCMC chains, it is also advisable to keep only every x-th step. The MCMC default settings for parameter draws (`*.mcmc.DrawParameters(..., mcmc = 2500, burnin = 500, thin = 50, chains = 2)`) should work well in many empirical settings. Depending on your platform, the code will either use a single core (on Windows OS), or multiple cores in parallel (on Unix/MacOS) to run the MCMC chains. To speed up convergence, the MCMC chains will be automatically initialized with the maximum likelihood estimates of Pareto/NBD. The sampled draws are wrapped as `coda::mcmc.list` object, and the `coda` package provides various helper methods (e.g. `as.matrix.mcmc.list`, `HPDinterval`, etc.) for performing output analysis and diagnostics for MCMC (cf. `help(package="coda")`).

Pareto/NBD (HB)

The Pareto/NBD (HB) is identical to Pareto/NBD with respect to its model assumptions, but differs in its estimation method. Rossi and Allenby (2003) provided a blueprint for applying a full Bayes approach (in contrast to an empirical Bayes approach) to hierarchical models such as Pareto/NBD. Ma and Liu (2007) then published a specific MCMC scheme, comprised of Gibbs sampling with slice sampling to draw from the conditional distributions. Later Abe (2009) suggested then in their technical appendix to augment the parameter space with the unobserved lifetime τ and activity status z in order to decouple the sampling of the transaction process from the dropout process. This allows the sampling scheme to take advantage of conjugate priors for drawing λ and μ. Both methods are available, yet the latter (`use_data_augmentation=TRUE`) should be the preferred choice because it is significantly faster.

Let's apply the Pareto/NBD (HB), with the default MCMC settings in place, for the online grocery dataset. First we draw parameters with `pnbd.mcmc.DrawParameters`, and then pass these forward to the model-independent methods `mcmc.DrawFutureTransactions`, `mcmc.PActive` and `mcmc.PAlive`.

```
data("groceryElog")
cbs <- elog2cbs(groceryElog, T.cal = "2006-12-31") # 52 weeks calibration
# generate parameter draws
pnbd.draws <- pnbd.mcmc.DrawParameters(cbs) # ~13secs on 2015 MacBook Pro
#> set param_init: 0.7463, 5.4544, 0.3817, 6.9221
#> running in parallel on 2 cores
# generate draws for holdout period
pnbd.xstar.draws <- mcmc.DrawFutureTransactions(cbs, pnbd.draws)
# create point estimates by taking average across samples
cbs$x.star.pnbd  <- apply(pnbd.xstar.draws, 2, mean)
cbs$pactive.pnbd <- mcmc.PActive(pnbd.xstar.draws)
cbs$palive.pnbd  <- mcmc.PAlive(pnbd.draws)
# show estimates for first few customers
head(cbs[, c("x", "t.x", "x.star", "x.star.pnbd", "pactive.pnbd", "palive.pnbd")])
#>    x      t.x x.star x.star.pnbd pactive.pnbd palive.pnbd
#> 1  0  0.00000      0        0.10         0.08        0.23
#> 2  1 50.28571      0        1.27         0.63        1.00
#> 3 19 48.57143     14       15.11         0.91        0.92
#> 4  0  0.00000      0        0.18         0.09        0.24
#> 5  2 40.42857      3        1.97         0.72        0.89
#> 6  5 47.57143      6        4.93         0.93        0.97
```

As can be seen, the basic application of an MCMC-estimated model is just as straightforward as for ML-estimated models. However, the return object of `pnbd.mcmc.DrawParameters` allows for further analysis. Its return object is a 2-element list: `level_1` is a list of `coda::mcmc.lists`, one for each customer, with draws for customer-level parameters (λ, μ, τ, z), and `level_2` a `coda::mcmc.list` with draws for cohort-level parameters (r, α, s, β). In total we have 100 samples for nrow(cbs) * 4 + 4 = 6104 parameters, and for each we can inspect the MCMC traces, the estimated distributions and calculate summary statistics.

```
class(pnbd.draws$level_2)
#> [1] "mcmc.list"

# convert cohort-level draws from coda::mcmc.list to a matrix, with each parameter
# becoming a column, and each draw a row
cohort.draws <- pnbd.draws$level_2
head(as.matrix(cohort.draws), 5)
#>              r    alpha         s     beta
#> [1,] 0.7433038 5.882881 0.3889020 6.826054
#> [2,] 0.8421824 6.144940 0.3640106 5.814168
#> [3,] 0.8405980 6.022312 0.3391057 4.148905
#> [4,] 0.8055730 5.798655 0.3143380 4.086252
#> [5,] 0.8336055 6.196172 0.3604559 4.898867

# compute median across draws, and compare to ML estimates
rbind(`Pareto/NBD (HB)` = apply(as.matrix(cohort.draws), 2, median),
      `Pareto/NBD`      = BTYD::pnbd.EstimateParameters(cbs))
#>                          r    alpha         s     beta
#> Pareto/NBD (HB) 0.7894316 5.717755 0.3634955 4.841299
#> Pareto/NBD      0.7864354 5.678976 0.3862209 5.717118

# plot trace- and density-plots for heterogeneity parameters
op <- par(mfrow = c(2, 4), mar = c(2.5, 2.5, 2.5, 2.5))
coda::traceplot(pnbd.draws$level_2)
coda::densplot(pnbd.draws$level_2)
par(op)
```

```
class(pnbd.draws$level_1)
#> [1] "list"
length(pnbd.draws$level_1)
#> [1] 1525

customer4 <- "4"
customer4.draws <- pnbd.draws$level_1[[customer4]]
head(as.matrix(customer4.draws), 5)
#>            lambda           mu          tau z
#> [1,] 0.0008224448 3.336727e-02    19.304104 0
#> [2,] 0.0058029809 6.453994e-05  3525.447722 1
#> [3,] 0.0267553252 1.677446e-03  1244.467271 1
#> [4,] 0.0237583309 2.684788e-04 13581.716333 1
#> [5,] 0.0366261862 2.522227e-01     0.376544 0
round(apply(as.matrix(customer4.draws), 2, median), 3)
#> lambda     mu    tau      z
#>  0.026  0.050  5.976  0.000
```

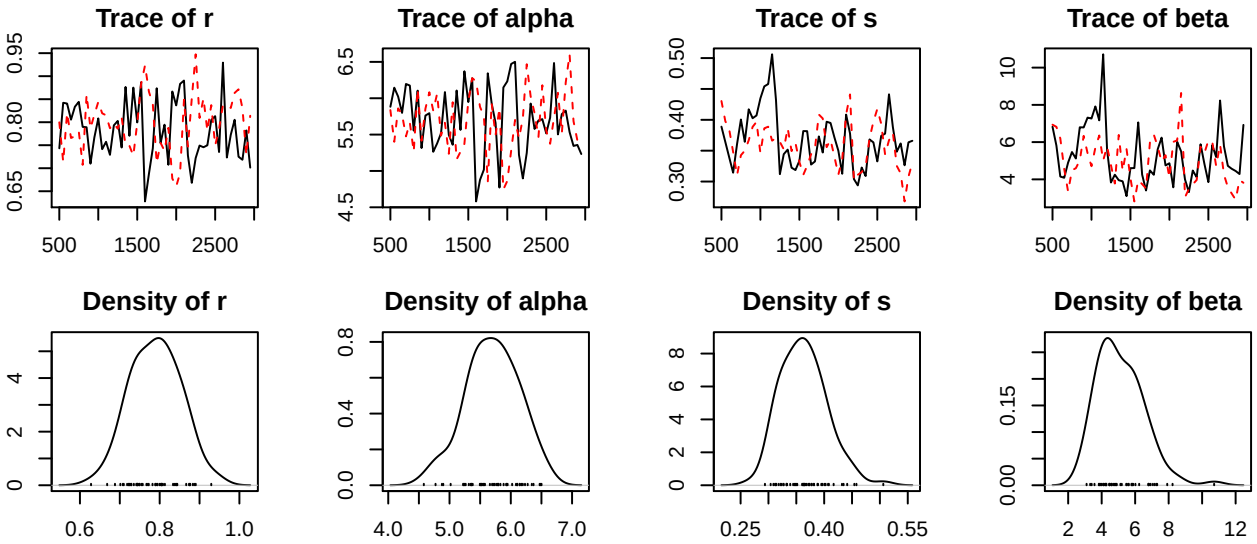


Figure 4: MCMC traces and parameter distributions of cohort-level parameters

```
# plot trace- and density-plots for customer4 parameters
op <- par(mfrow = c(2, 4), mar = c(2.5, 2.5, 2.5, 2.5))
coda::traceplot(pnbd.draws$level_1[[customer4]])
coda::densplot(pnbd.draws$level_1[[customer4]])
par(op)
```

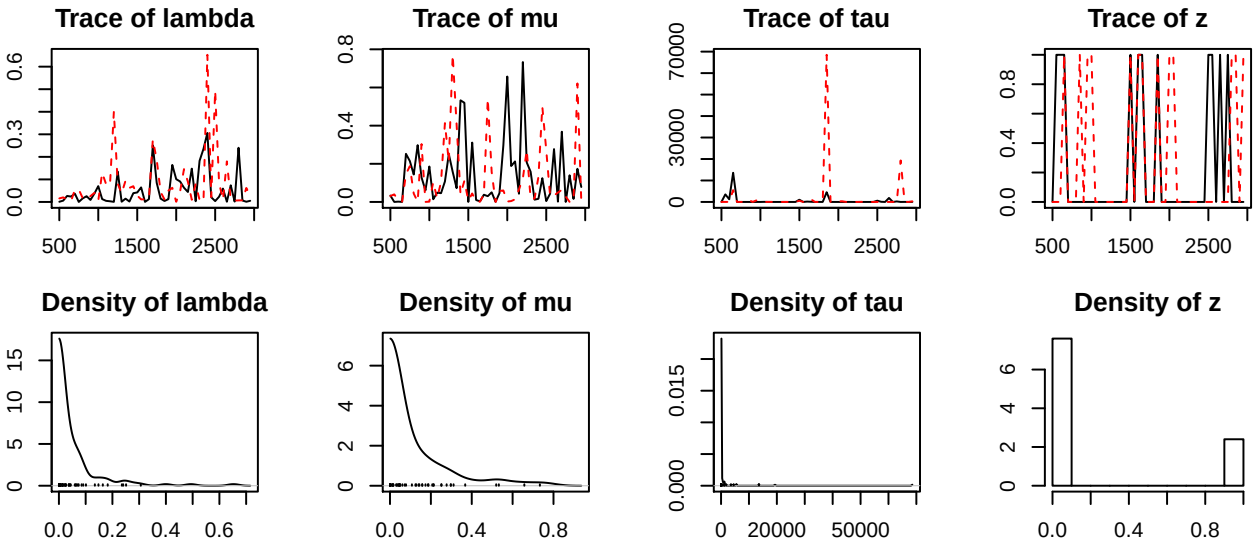


Figure 5: MCMC traces and parameter distributions of individual-level parameters for customer 1

Analogous to MLE-based models, we can also plot weekly transaction counts, as well as frequency plots at an aggregated level. These methods can be applied to all provided MCMC-based models in the following way.

```
op <- par(mfrow = c(1, 2))
nil <- mcmc.PlotFrequencyInCalibration(pnbd.draws, cbs)
nil <- mcmc.PlotTrackingInc(pnbd.draws,
```

```
        T.cal = cbs$T.cal,
        T.tot = max(cbs$T.cal + cbs$T.star),
        actual.inc.tracking.data = elog2inc(groceryElog))
par(op)
```

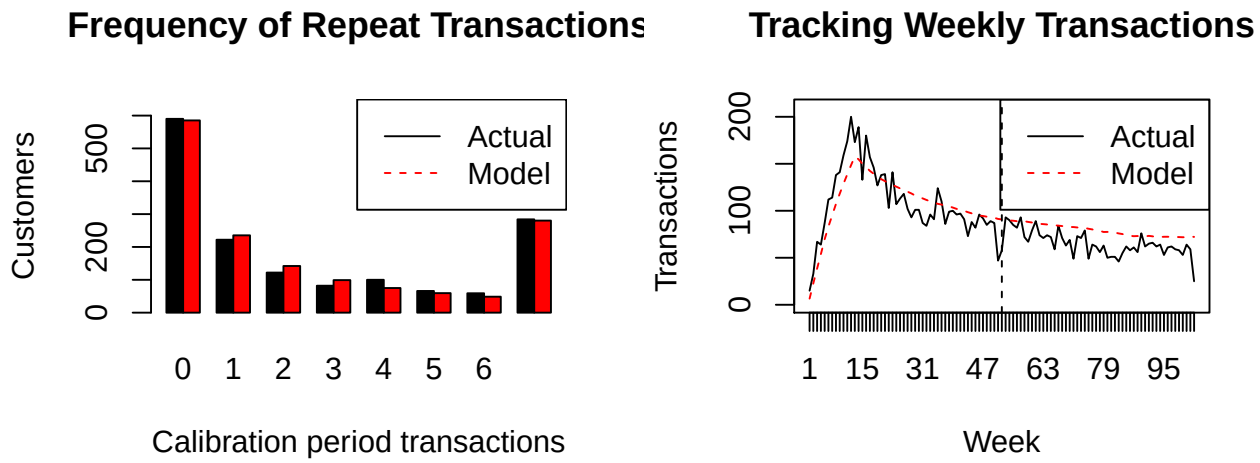


Figure 6: Diagnostic Plots

Pareto/NBD (Abe)

Abe (2009) introduced a variant of Pareto/NBD by replacing the two independent gamma distributions for individuals' purchase rates λ and dropout rates μ with a multivariate lognormal distribution. The BTYDplus package refers to this model variant as Pareto/NBD (Abe). The multivariate lognormal distribution permits a correlation between purchase and dropout processes, but even more importantly, can be easily extended to a linear regression model to incorporate customer-level covariates. This flexibility can significantly boost inference, if any of the captured covariates indeed helps in explaining the heterogeneity within the customer cohort.

The online grocery dataset doesn't contain any additional covariates, so for demonstration purposes we will apply Pareto/NBD (Abe) to the CDNow dataset and reproduce the findings from the original paper. First we estimate a model without covariates (M1), and then, we incorporate the dollar amount of the first purchase as a customer-level covariate (M2).

```
# load CDNow event log from BTYD package
cdnowElog <- read.csv(system.file("data/cdnowElog.csv", package = "BTYD"),
                      stringsAsFactors = FALSE,
                      col.names = c("cust", "sampleid", "date", "cds", "sales"))
cdnowElog$date <- as.Date(as.character(cdnowElog$date), format = "%Y%m%d")
# convert to CBS, and split into 39 weeks calibration, and 39 weeks holdout
cdnowCbs <- elog2cbs(cdnowElog, T.cal = "1997-09-30", T.tot = "1998-06-30")

# estimate Pareto/NBD (Abe) without covariates; see model M1 in Abe (2009)
draws.m1 <- abe.mcmc.DrawParameters(cdnowCbs,
              mcmc = 7500, burnin = 2500) # ~33secs on 2015 MacBook Pro
quant <- function(x) round(quantile(x, c(0.025, 0.5, 0.975)), 2)
t(apply(as.matrix(draws.m1$level_2), 2, quant))
#>                        2.5%   50% 97.5%
#> log_lambda            -3.70 -3.54 -3.32
```

```
#> log_mu                -3.96 -3.59 -3.26
#> var_log_lambda         1.10  1.34  1.65
#> cov_log_lambda_log_mu -0.20  0.13  0.74
#> var_log_mu             1.44  2.62  5.05

#' append dollar amount of first purchase to use as covariate
first <- aggregate(sales ~ cust, cdnowElog, function(x) x[1] * 10^-3)
names(first) <- c("cust", "first.sales")
cdnowCbs <- merge(cdnowCbs, first, by = "cust")

#' estimate with first purchase spend as covariate; see model M2 in Abe (2009)
draws.m2 <- abe.mcmc.DrawParameters(cdnowCbs, covariates = c("first.sales"),
              mcmc = 7500, burnin = 2500) # ~33secs on 2015 MacBook Pro
t(apply(as.matrix(draws.m2$level_2), 2, quant))
#>                          2.5%   50% 97.5%
#> log_lambda_intercept   -4.02 -3.77 -3.19
#> log_mu_intercept       -4.37 -3.73 -2.69
#> log_lambda_first.sales  0.04  6.04  9.39
#> log_mu_first.sales     -9.02  1.73  7.90
#> var_log_lambda          0.01  1.35  1.79
#> cov_log_lambda_log_mu  -0.35  0.22  0.76
#> var_log_mu              0.55  2.59  4.97
```

The parameter estimates for model M1 and M2 match roughly the numbers reported in Table 3 of Abe (2009). There are some discrepancies for the parameters `log_lambda_first.sales` and `log_mu_first.sales`, but the high level result remains unaltered: The dollar amount of a customer's initial purchase correlates positively with purchase frequency, but doesn't influence the dropout process.

Note, that the BTYDplus package can establish via simulations that its provided implementation is indeed correctly able to reidentify underlying data generating parameters, including the regression coefficients for the covariates.

Pareto/GGG

Platzer and Reutterer (2016) presented another extension of the Pareto/NBD model. The Pareto/GGG generalizes the distribution for the intertransaction times from the exponential to the Gamma distribution, whereas its shape parameter k is also allowed to vary across customers following a Gamma(t, γ) distribution. Hence, the purchase process follows a Gamma-Gamma-Gamma (GGG) mixture distribution, that is capable of capturing a varying degree of regularity across customers. For datasets which exhibit regularity in their timing patterns, leveraging that information can yield significant improvements in terms of forecasting accuracy. This results from improved inferences about customers' latent state in the presence of regularity.

```
# estimte Pareto/GGG
pggg.draws <- pggg.mcmc.DrawParameters(cbs) # ~2mins on 2015 MacBook Pro
# generate draws for holdout period
pggg.xstar.draws <- mcmc.DrawFutureTransactions(cbs, pggg.draws)
# create point estimates by taking average across samples
cbs$x.star.pggg  <- apply(pggg.xstar.draws, 2, mean)
cbs$pactive.pggg <- mcmc.PActive(pggg.xstar.draws)
cbs$palive.pggg  <- mcmc.PAlive(pggg.draws)
# show estimates for first few customers
head(cbs[, c("x", "t.x", "x.star", "x.star.pggg", "pactive.pggg", "palive.pggg")])
#>     x      t.x x.star x.star.pggg pactive.pggg palive.pggg
```

```
#>  1  0  0.00000      0        0.08         0.06        0.10
#>  2  1 50.28571      0        0.96         0.57        0.97
#>  3 19 48.57143     14       15.45         0.85        0.87
#>  4  0  0.00000      0        0.11         0.07        0.10
#>  5  2 40.42857      3        2.21         0.86        0.93
#>  6  5 47.57143      6        4.57         0.96        0.98

# report median parameter estimates
round(apply(as.matrix(pggg.draws$level_2), 2, median), 3)
#>      t gamma     r alpha     s  beta
#>  1.671 0.368 0.942 5.150 0.435 4.610
median.est <- sapply(pggg.draws$level_1, function(draw) {
  apply(as.matrix(draw), 2, median)
})
round(apply(median.est, 1, mean), 3)
#>      k lambda     mu    tau      z
#>  3.889  0.159  0.064 69.440  0.314
```

Summarizing the estimated parameter distributions shows that regularity parameter `k` is estimated significantly larger than 1, and that it varies substantially across customers.

```
(mae <- c(`Pareto/GGG` = mean(abs(cbs$x.star - cbs$x.star.pggg)),
          `Pareto/NBD` = mean(abs(cbs$x.star - cbs$x.star.pnbd))))
#> Pareto/GGG Pareto/NBD
#>   1.389548   1.513515
```

Comparing the mean absolute error of Pareto/GGG and Pareto/NBD predictions shows a lift of 8.2%. This confirms that the Pareto/GGG is indeed able to leverage the detected regularity within the online grocery dataset for improving the forecasting accuracy.

References

Abe, Makoto. 2009. "'Counting Your Customers' One by One: A Hierarchical Bayes Extension to the Pareto/NBD Model." *Marketing Science* 28 (3). INFORMS: 541–53.

Batislam, Emine Persentili, Meltem Denizel, and Alpay Filiztekin. 2007. "Empirical Validation and Comparison of Models for Customer Base Analysis." *International Journal of Research in Marketing* 24 (3). Elsevier: 201–9.

Ehrenberg, Andrew SC. 1959. "The Pattern of Consumer Purchases." *Applied Statistics*. JSTOR, 26–41.

Fader, P.S., B.G.S. Hardie, and K.L. Lee. 2005. "Counting Your Customers the Easy Way: An Alternative to the Pareto/NBD Model." *Marketing Science* 24: 275–84.

Fader, Peter S, Bruce GS Hardie, and Ka Lok Lee. 2005. "RFM and CLV: Using Iso-Value Curves for Customer Base Analysis." *Journal of Marketing Research* 42 (4). American Marketing Association: 415–30.

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

Ma, Shao-Hui, and Jin-Lan Liu. 2007. "The MCMC Approach for Solving the Pareto/NBD Model and Possible Extensions." In *Third International Conference on Natural Computation (ICNC 2007)*, 2:505–12. IEEE.

Platzer, Michael, and Thomas Reutterer. 2016. "Ticking Away the Moments: Timing Regularity Helps to

Better Predict Customer Activity." *Marketing Science*. INFORMS.

———. n.d. "Leveraging Purchase Regularity for Predicting Customer Behavior, the Easy Way."

Rossi, Peter E, and Greg M Allenby. 2003. "Bayesian Statistics and Marketing." *Marketing Science* 22 (3). INFORMS: 304–28.

Schmittlein, D.C., D.G. Morrison, and R. Colombo. 1987. "Counting Your Customers: Who Are They and What Will They Do Next?" *Management Science* 33 (1). INFORMS Institute for Operations Research; the Management Sciences (INFORMS), Linthicum, Maryland, USA: 1–24.

Wheat, R.D., and D.G. Morrison. 1990. "Estimating Purchase Regularity with Two Interpurchase Times." *Journal of Marketing Research* 27 (1): 87–93.