CLV PREDICTIONS USING THE sBG METHOD

Step 0 - You will need the file retentionforsbgclv.csv dataset for this exercise. Please load the required libraries as well before attempting this exercise. The dataset is a simple record of retention rates observed for a subscription based service such as Spotify. With a $1 per month subscription the service has a record of identifying four groups (segments) each with differing retention rates on a month by month basis. The information is available for the six months this service has been in operation since launch. For each month, the company has identified for case 01 and case 02 the active and lost customers. For case 03 and case 04 the numbers were taken from initial reports of a competing service (such as Pandora as a competitor to Spotify). The purpose of this exercise is to predict the retention rates at 24 months after launch and estimate the CLV.

Step 1 – Load the libraries and the data

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------------

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ---------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
library(MLmetrics)
```

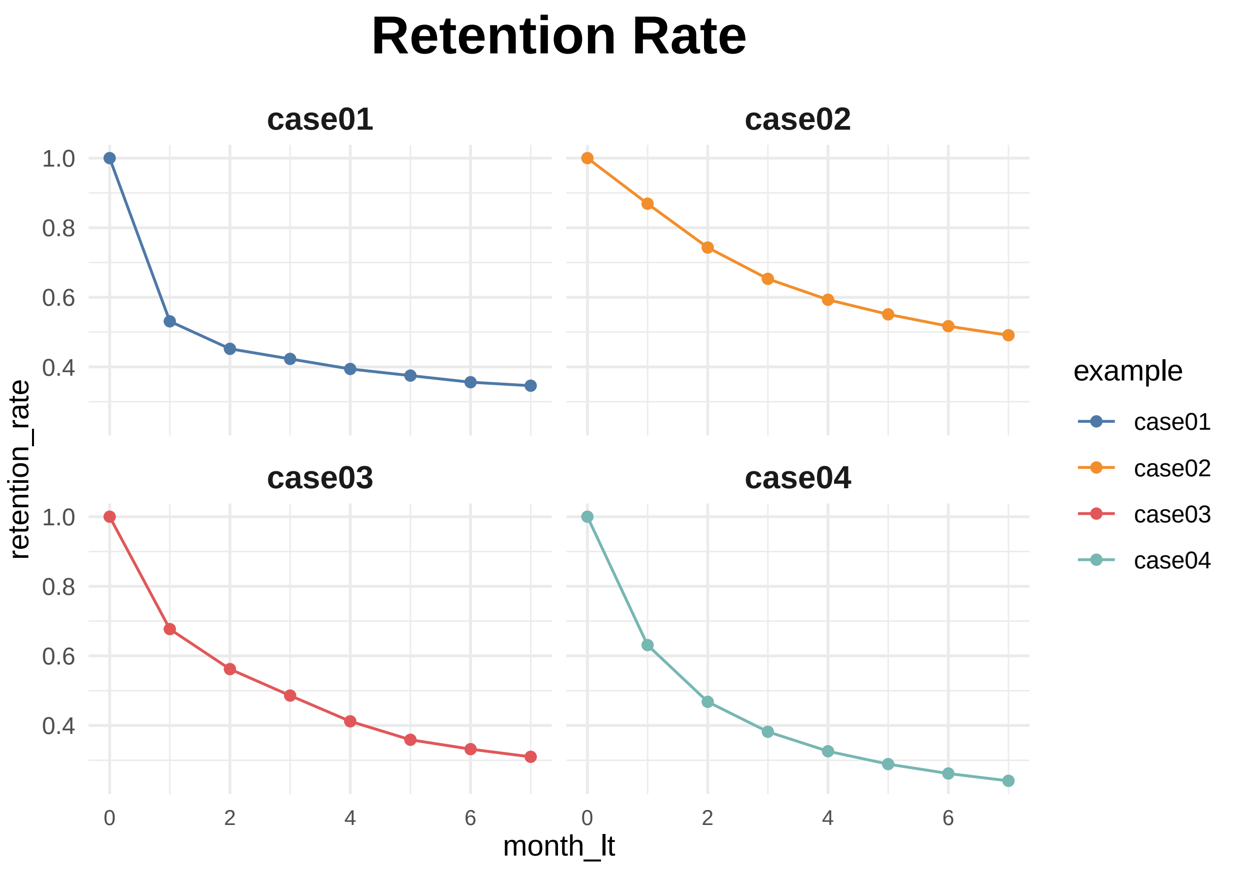
```
##
```

```
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall
```

```
df_ret <- read.csv("retentionforsbgclv.csv") # load the retentionforsbgclv.csv
```

Step 1 – Plot the retention curves for the four cases we have in the dataset. Color values are optional but it does make it easier to compare plots in the later stages.

```
ggplot(df_ret, aes(x = month_lt, y = retention_rate, group = example, color = example)) +
     theme_minimal() +
     facet_wrap(~ example) +
     scale_color_manual(values = c('#4e79a7', '#f28e2b', '#e15759', '#76b7b2')) +
     geom_line() +
     geom_point() +
     theme(plot.title = element_text(size = 20, face = 'bold', vjust = 2, hjust = 0.5),
           axis.text.x = element_text(size = 8, hjust = 0.5, vjust = .5, face = 'plain'),
           strip.text = element_text(face = 'bold', size = 12)) +
    ggtitle('Retention Rate')
```



Step 2 – The following are the functions from Fader - Hardie used to create sBG dist. There are packages that can run this but the authors were nice enough to provide the functions in R for easy application.

functions for sBG distribution

```
churnBG <- Vectorize(function(alpha, beta, period) {
     t1 = alpha / (alpha + beta)
     result = t1
     if (period > 1) {
         result = churnBG(alpha, beta, period - 1) * (beta + period - 2) / (alpha + beta + period - 1)
     }
     return(result)
 }, vectorize.args = c("period"))

 survivalBG <- Vectorize(function(alpha, beta, period) {
     t1 = 1 - churnBG(alpha, beta, 1)
     result = t1
     if(period > 1){
         result = survivalBG(alpha, beta, period - 1) - churnBG(alpha, beta, period)
     }
     return(result)
}, vectorize.args = c("period"))

MLL <- function(alphabeta) {
     if(length(activeCust) != length(lostCust)) {
         stop("Variables activeCust and lostCust have different lengths: ",
              length(activeCust), " and ", length(lostCust), ".")
     }
     t = length(activeCust) # number of periods
     alpha = alphabeta[1]
     beta = alphabeta[2]
     return(-as.numeric(
         sum(lostCust * log(churnBG(alpha, beta, 1:t))) +
             activeCust[t]*log(survivalBG(alpha, beta, t))
     ))
 }
```

Step 3 – take retention data and predict outcomes using the Fader-Hardie functions.

```
df_ret <- df_ret %>%
     group_by(example) %>%
     mutate(activeCust = 1000 * retention_rate,
            lostCust = lag(activeCust) - activeCust,
            lostCust = ifelse(is.na(lostCust), 0, lostCust)) %>%
     ungroup()
```

```r
 ret_preds01 <- vector('list', 7)
 for (i in c(1:7)) {

     df_ret_filt <- df_ret %>%
         filter(between(month_lt, 1, i) == TRUE & example == 'case01')

     activeCust <- c(df_ret_filt$activeCust)
     lostCust <- c(df_ret_filt$lostCust)

     opt <- optim(c(1, 1), MLL)
retention_pred <- round(c(1, survivalBG(alpha = opt$par[1], beta = opt$par[2], c(1:7))), 3)

     df_pred <- data.frame(month_lt = c(0:7),
                           example = 'case01',
                           fact_months = i,
                           retention_pred = retention_pred)
}
```

```
## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(survivalBG(alpha, beta, t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(survivalBG(alpha, beta, t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(survivalBG(alpha, beta, t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(survivalBG(alpha, beta, t)): NaNs produced
```

```r
ret_preds01[[i]] <- df_pred


ret_preds01 <- as.data.frame(do.call('rbind', ret_preds01))

ret_preds02 <- vector('list', 7)
 for (i in c(1:7)) {

     df_ret_filt <- df_ret %>%
         filter(between(month_lt, 1, i) == TRUE & example == 'case02')

     activeCust <- c(df_ret_filt$activeCust)
     lostCust <- c(df_ret_filt$lostCust)

     opt <- optim(c(1, 1), MLL)
retention_pred <- round(c(1, survivalBG(alpha = opt$par[1], beta = opt$par[2], c(1:7))), 3)

     df_pred <- data.frame(month_lt = c(0:7),
                           example = 'case02',
                           fact_months = i,
                           retention_pred = retention_pred)
```

```r
    ret_preds02[[i]] <- df_pred
}
```

```
## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced

## Warning in log(churnBG(alpha, beta, 1:t)): NaNs produced
```

```r
ret_preds02 <- as.data.frame(do.call('rbind', ret_preds02))

ret_preds03 <- vector('list', 7)
 for (i in c(1:7)) {

     df_ret_filt <- df_ret %>%
         filter(between(month_lt, 1, i) == TRUE & example == 'case03')

     activeCust <- c(df_ret_filt$activeCust)
     lostCust <- c(df_ret_filt$lostCust)

     opt <- optim(c(1, 1), MLL)
     retention_pred <- round(c(1, survivalBG(alpha = opt$par[1], beta = opt$par[2], c(1:7))), 3)

     df_pred <- data.frame(month_lt = c(0:7),
                           example = 'case03',
                           fact_months = i,
                           retention_pred = retention_pred)
     ret_preds03[[i]] <- df_pred
 }

 ret_preds03 <- as.data.frame(do.call('rbind', ret_preds03))

ret_preds04 <- vector('list', 7)
for (i in c(1:7)) {

     df_ret_filt <- df_ret %>%
         filter(between(month_lt, 1, i) == TRUE & example == 'case04')

     activeCust <- c(df_ret_filt$activeCust)
     lostCust <- c(df_ret_filt$lostCust)

     opt <- optim(c(1, 1), MLL)
retention_pred <- round(c(1, survivalBG(alpha = opt$par[1], beta = opt$par[2], c(1:7))), 3)

     df_pred <- data.frame(month_lt = c(0:7),
                           example = 'case04',
                           fact_months = i,
                              retention_pred = retention_pred)
     ret_preds04[[i]] <- df_pred
 }

 ret_preds04 <- as.data.frame(do.call('rbind', ret_preds04))


ret_preds <- bind_rows(ret_preds01, ret_preds02, ret_preds03, ret_preds04)
```

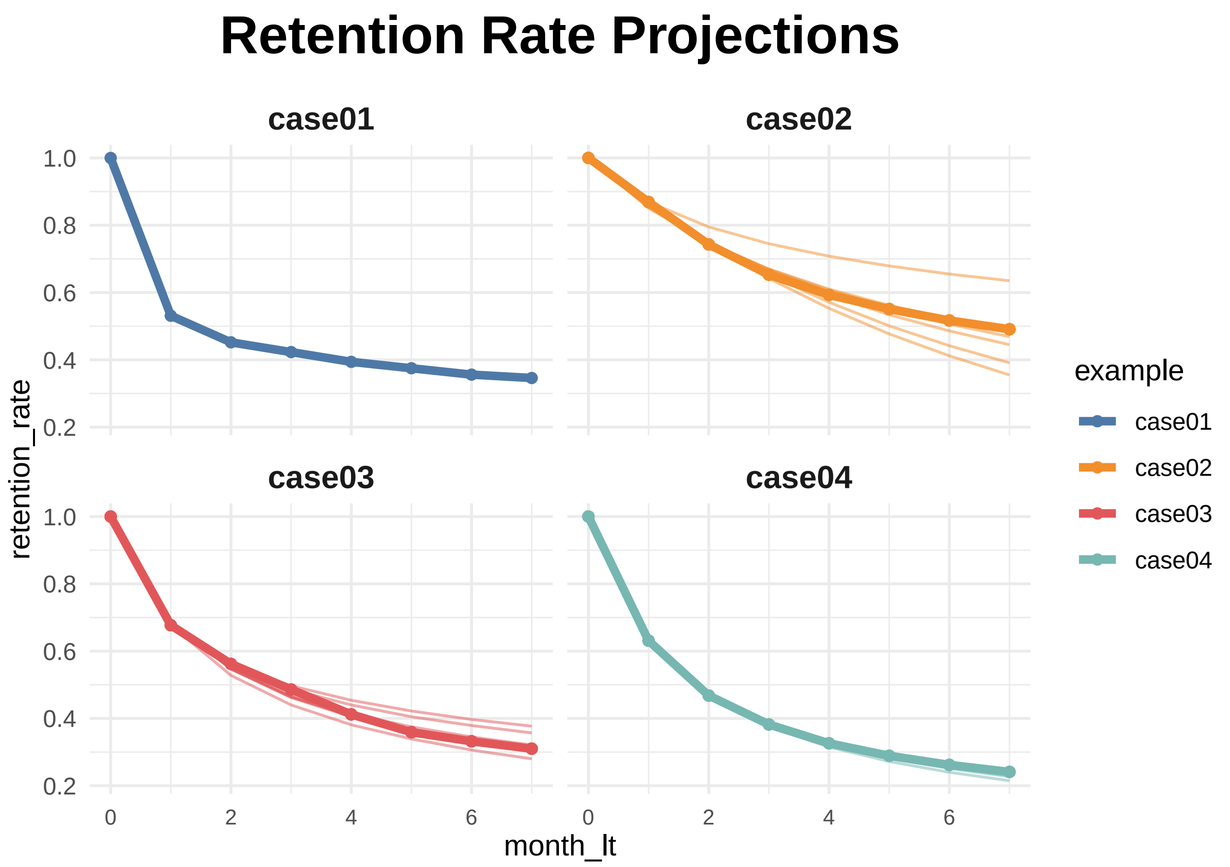
```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector
```

```
df_ret_all <- df_ret %>%
     select(month_lt, example, retention_rate) %>%
     left_join(., ret_preds, by = c('month_lt', 'example'))
```

```
## Warning: Column `example` joining factor and character vector, coercing into
## character vector
```

Step 4 – Plot the retention curves again to see how the predicted curves differ from the observed data curves. The visualization of the predicted retention curves and mean average percentage error (MAPE) that you get as output here shows how robust the sBG approach is in completing the retention curves even with limited data. As you can see from the graphs, in most cases, the accuracy of the predictions is very high even with just the first month's data and it significantly improves when adding historical periods.

```
ggplot(df_ret_all, aes(x = month_lt, y = retention_rate, group = example, color = example)) +
     theme_minimal() +
     facet_wrap(~ example) +
     scale_color_manual(values = c('#4e79a7', '#f28e2b', '#e15759', '#76b7b2')) +
     geom_line(size = 1.5) +
     geom_point(size = 1.5) +
     geom_line(aes(y = retention_pred, group = fact_months), alpha = 0.5) +
     theme(plot.title = element_text(size = 20, face = 'bold', vjust = 2, hjust = 0.5),
           axis.text.x = element_text(size = 8, hjust = 0.5, vjust = .5, face = 'plain'),
           strip.text = element_text(face = 'bold', size = 12)) +
     ggtitle('Retention Rate Projections')
```



#
Step 5 – Predict LTV using the predicted retentions and add to dataset. To obtain the average LTV prediction, we simply need to multiply the retention rate by the subscription price and calculate the cumulative amount for the required period. Let's start by needing to calculate the average LTV for case03 based on two historical months with a forecast horizon of 24 months (2 years) and a subscription price of $1. We make these assumptions for simplicity and to keep the code easy to understand.

```
df_ltv_03 <- df_ret %>%
     filter(between(month_lt, 1, 2) == TRUE & example == 'case03')

activeCust <- c(df_ltv_03$activeCust)
lostCust <- c(df_ltv_03$lostCust)

opt <- optim(c(1, 1), MLL)
retention_pred <- round(c(survivalBG(alpha = opt$par[1], beta = opt$par[2], c(3:24))), 3)


df_pred <- data.frame(month_lt = c(3:24), retention_pred = retention_pred)

df_ltv_03 <- df_ret %>%
    filter(between(month_lt, 0, 2) == TRUE & example == 'case03') %>%
     select(month_lt, retention_rate) %>%
     bind_rows(., df_pred) %>%
     mutate(retention_rate_calc = ifelse(is.na(retention_rate), retention_pred, retention_rate),
            ltv_monthly = retention_rate_calc * 1,
            ltv_cum = round(cumsum(ltv_monthly), 2))
```

Step 6 – examine the dataset for cumulative LTV for each case. Keep

```
df_ltv_03 # $9.33 cumulative CLV, look at data after 24 months.
```

```
## # A tibble: 25 x 6
##    month_lt retention_rate retention_pred retention_rate_ca~ ltv_monthly ltv_cum
##       <int>          <dbl>          <dbl>              <dbl>       <dbl>   <dbl>
##  1        0          1             NA                  1           1        1
##  2        1          0.677         NA                  0.677       0.677    1.68
##  3        2          0.562         NA                  0.562       0.562    2.24
##  4        3         NA              0.497              0.497       0.497    2.74
##  5        4         NA              0.454              0.454       0.454    3.19
##  6        5         NA              0.422              0.422       0.422    3.61
##  7        6         NA              0.397              0.397       0.397    4.01
##  8        7         NA              0.377              0.377       0.377    4.39
##  9        8         NA              0.361              0.361       0.361    4.75
## 10        9         NA              0.346              0.346       0.346    5.09
## # ... with 15 more rows
```

Interpretation of the final output month_lt, retention_rate, retention_pred, retention_rate_calc, ltv_monthly, ltv_cum 1 0 1.000 NA 1.000 1.000 1.00 2 1 0.677 NA 0.677 0.677 1.68 3 2 0.562 NA 0.562 0.562 2.24 4 3 NA 0.497 0.497 0.497 2.74 5 4 NA 0.454 0.454 0.454 3.19 6 5 NA 0.422 0.422 0.422 3.61 7 6 NA 0.397 0.397 0.397 4.01 8 7 NA 0.377 0.377 0.377 4.39 9 8 NA 0.361 0.361 0.361 4.75 10 9 NA 0.346 0.346 0.346 5.09 11 10 NA 0.334 0.334 0.334 5.43 12 11 NA 0.323 0.323 0.323 5.75 13 12 NA 0.314 0.314 0.314 6.06 14 13 NA 0.305 0.305 0.305 6.37 15 14 NA 0.297 0.297 0.297 6.67 16 15 NA 0.290 0.290 0.290 6.96 17 16 NA 0.284 0.284 0.284 7.24 18 17 NA 0.278 0.278 0.278 7.52 19 18 NA 0.273 0.273 0.273 7.79 20 19 NA 0.267 0.267 0.267 8.06 21 20 NA 0.263 0.263 0.263 8.32 22 21 NA 0.258 0.258 0.258 8.58 23 22 NA 0.254 0.254 0.254 8.83 24 23 NA 0.250 0.250 0.250 9.08 25 24 NA 0.246 0.246 0.246 9.33 Interpretation is straightforward, just read the table. While we only have two periods of data used in the calculation of LTV values for case03, we find that the sBG estimation of the retention curve remains robust. You can calculate the other three cases as well to see if case01 differs from the case04 group in any meaningful manner. Please note that this calculation is primarily focused on the retention curves and less on the calculation of the CLV. However, CLV is so heavily dependent on the retention rate that it is a natural extension of the retention rates. This calculation as demonstrated ignores costs, so it would help to have costs factored into the subscription price (basically use gross margin) for more robust and reliable calculations.