Churn Management

Günter J. Hitsch

February 8, 2017

```r
library(bit64)
library(data.table)
library(ggplot2)
library(broom)
library(knitr)
library(Hmisc)
```

Overview

Cell2Cell is a wireless telecom company (the name was changed to disguise the exact identity) that attempts to mitigate customer churn. The goal is to develop a model to predict customer churn at Cell2Cell and use the insights from the model to develop a targeted incentive plan to lower the rate at which customers churn.

Address these key issues:

1. Is customer churn at Cell2Cell predictable from the customer information that Cell2Cell maintains?
2. What factors drive customer churn? Which factors are particularly important?
3. What incentives should Cell2Cell offer to prevent customer churn?
4. What is the economic value of a proposed targeted plan to prevent churn, and how does this value differ across customer segments?

Data

All data are contained in the file `Cell2Cell.RData`. Please consult the file `Cell2Cell-Database-Documentation.xlsx` for a description of the data and some summary statistics. Note that *calibration sample* is an alternative term for *training* or *estimation* sample.

Inspect the data. We see that the calibration sample was selected using *oversampling*—half of the observations are 1 (churn) and half of the observations are 0. The purpose of oversampling was to obtain more precise estimates (lower standard errors) when estimating a logistic regression model. The validation sample, on the other hand, was not created using oversampling and represents the *true churn rate* in the data.

As you can see, some variables have missing values, which—as you know by now—is common and of no concern (unless the missing values indicate some *systematic* flaws or bias in how the data were constructed). Most estimation methods in R will automatically delete rows with missing values before estimating the model. However, the `predict` methods will yield `NA` values if a row in the data used for prediction contains missing values. Hence, in a situation where you don't need to keep the full data I recommend to remove any observations with missing values before you conduct the main analysis.

Model estimation

Estimate a logit model to predict the conditional churn probability.

You can inspect the regression output using methods you already used, such as `summary` or `stargazer`. Having said this, especially when you have a large number of inputs, it can be convenient to store the regression estimates in a table. A simple way to do this is to install the broom package that has the purpose of cleaning up messy R output.

Using the `tidy` function in the `broom` package it is trivial to capture the regression output in the form of a data.table:

```
results_DT = as.data.table(tidy(fit))
kable(results_DT, digits = 5)
```

For `kable` to work, you need to load the `knitr` library. Inspect the estimation output, and keep the `results_DT` table for later use.

Prediction: Accounting for oversampling

In order to correct the scale of the predicted response (in this example: churn) in the validation sample we need to supply an *offset variable* to the logistic regression model. An offset is a known number that is added to the right-hand side of the regression when estimating the model, and adding the offset will correspondingly change the estimate of the intercept. The offset takes the form:

$$\text{offset} = [\log(\bar{p}_e) - \log(1 - \bar{p}_e)] - [\log(\bar{p}_v) - \log(1 - \bar{p}_v)]$$

Here, $\bar{p}_e$ is the average response rate in the estimation (training) sample and $\bar{p}_v$ is the average response rate in the validation sample.

Create an `offset_var` variable and add it to the data set. Then re-estimate the logistic regression. To tell `glm` that you want to use `offset_var` you need to use a formula of the form:

```
y ~ offset(offset_var) + ...
```

Where you place `offset()` on the right-hand side of the formula is irrelevant.

Before predicting the response rate in the validation sample you need to set the offset to 0. Then, when you invoke the `predict` function, supply the data with the offset set to 0 using the `newdata` option. Compare the average predicted response with the average observed response rate in the validation sample.

Predictive power: Lift

We evaluate the predictive fit of the logistic regression model using a lift table and lift chart. To develop reusable code we develop a function that returns a lift table. The function (call it `liftTable`) will need to take the following inputs:

- Predicted outcome or score
- Observed outcome
- Number of segments to be created based on the score

`liftTable` will return a data.table that contains:

- An index (`score_group`) for each segment that was created based on the score
- The average score value (predicted outcome) in the `score_group`
- The average observed outcome in the `score_group`
- A lower and upper bound for a 95 percent confidence interval for the average observed outcome
- The lift factor

You can calculate the 95 percent confidence interval using a normal approximation. If you do not remember how to do this from your statistics classes see the corresponding Wikipedia entry on binomial proportion confidence intervals. If you are adventurous, you can implement the slightly more accurate Wilson score interval method that is the default in the binconf function in the `Hmisc` package.

To code the `liftTable` I recommend to use the `cut_number` function in the ggplot2 package. `cut_number` takes a variable `x` and creates `n` groups with an approximately equal number of observations. Observations are assigned to the groups based on their ranking along the variable `x`. The format is:

```
cut_number(x, n = <no. of groups>)
```

To illustrate, we draw 10,000 random numbers from a uniform distribution on $[0, 5]$. `cut_number` assigns each number to one of five (because we set `n = 5`) groups.

```
set.seed(123)
DT = data.table(x = runif(10000, min = 0, max = 5))
DT[, group    := cut_number(x, n = 5)]
DT[, group_no := as.integer(group)]
```

```
head(DT)
```

```
           x        group group_no
1: 1.4378876     (1,2.01]        2
2: 3.9415257  (2.98,3.98]        4
3: 2.0448846  (2.01,2.98]        3
4: 4.4150870     (3.98,5]        5
5: 4.7023364     (3.98,5]        5
6: 0.2277825 [0.000327,1]        1
```

```
table(DT$group)
```

```

[0.000327,1]     (1,2.01]  (2.01,2.98]  (2.98,3.98]     (3.98,5]
        2000         2000         2000         2000         2000
```

As expected, because `x` is uniformly distributed on $[0, 5]$, the five groups created by `cut_number` correspond almost exactly to a $[k, k + 1]$ interval ($k = 0, 1, \ldots, 4$), and each of these intervals contains exactly 20 percent of all observations based on the rank of the `x` values. The `group` variable that we created is a factor that we converted to an integer score.

Calculate a lift table for 20 segments. Inspect the lift table. Then provide two charts. First, plot the `score_group` segments on the x-axis versus the observed churn rate on the y-axis, and also add bars for the 95 percent confidence intervals. Second, plot the segments versus the lift factor, and add a horizontal line at $y = 100$. How to do this in ggplot2 is explained in the ggplot2 document that I created.

Why do customers churn? — Effect sizes

We would like to understand *why* customers churn, which can help us to come up with incentives to prevent customer churn. Hence, we translate the logistic regression estimates into easily interpretable effect sizes for all inputs. Ultimately we will then know which inputs are strongly associated with customer churn.

One conceptually simple solution is to calculate the marginal effects using the `erer` package, copy and paste the output into an Excel file, and then predict the effect sizes in Excel. For bonus points, however, I recommend to perform the calculations directly in R. Once again, this ensures that you have easily reusable code. Below are some pointers on how to create a data.table that contains the predicted effect sizes of all variables.

First, start with the `results_DT` table that contains the logistic regression output. This table contains all variable names in the `term` column.

Second, recall the marginal effects formula, $\beta \cdot p \cdot (1-p)$. Using this formula, add a `marginal_effect` column to the results table. Although the results are largely immune to the calculation details, I recommend to take the average over all observation-specific marginal effects.

Third, calculate the standard deviation for all variables in the data. This can be tricky if you don't know the principle, so here is some help:

```r
sd_list = lapply(cell2cell_DT, sd)
sd_DT   = data.table(term    = names(sd_list),
                     std_dev = unlist(sd_list))
```

Note that the `lapply` function calculates a list of standard deviations for all columns in the data.table. We then create a data.table that contains the name of each variable in the column `term`, together with a column including all standard deviations.

You can now merge `results_DT` and `sd_DT` to evaluate the effect sizes and manually distinguish between the effect sizes of dummy variables and the effect sizes of other (continuous) variables.

Bonus: If you want to be extra-elegant, you can think of creating a function that detects if a column is a dummy variable that only contains 0/1 values. Hint: Such a function would check if the column contains exactly two unique values, and if 0 and 1 are contained among those unique values (the `all` function may come in handy for this task).

Economics of churn management

We would like to predict the value of a proposed churn management program in order to assess the maximum amount that we would spend to prevent a customer from churning for one year.

The prediction will depend on several parameters and assumptions. We consider a planning horizon of 6 years (the current year and five additional years), and an annual discount rate of 10 percent. Also, we will predict the churn management value for 10 groups, but it is good practice to make the code work for an arbitrary number of groups.

Also, we predict the program value for ten customer segments based on the predicted churn rate. We create these segments based on the validation sample data. We predict current and future customer profits at the year-level. Hence, we also need to convert both the monthly churn rate and the revenue data to the year-level.

Predict the economic value of a churn management program with success probability `gamma`. You can try multiple values, but I suggest to compare the results for $\gamma = 0.25$ and $\gamma = 0.5$.

Hint: It is easy to make little mistakes in the lifetime value predictions. Hence, be very clear about what your code is supposed to achieve, and check that every step is correct.