# Attribution Modeling

Howard Nguyen

2023-04-19

## Load needed libraries

```
#Load the libraries
library("ChannelAttribution")
library("ggplot2")
library("reshape")
library("dplyr")
library("plyr")
library("reshape2")
library("markovchain")
library("plotly")
```

## Read the dataset

```
channel <- read.csv("Channel_attribution.csv", header = T)
head(channel)
```

```
##   R05A.01 R05A.02 R05A.03 R05A.04 R05A.05 R05A.06 R05A.07 R05A.08 R05A.09
## 1      16       4       3       5      10       8       6       8      13
## 2       2       1       9      10       1       4       3      21      NA
## 3       9      13      20      16      15      21      NA      NA      NA
## 4       8      15      20      21      NA      NA      NA      NA      NA
## 5      16       9      13      20      21      NA      NA      NA      NA
## 6       1      11       8       4       9      21      NA      NA      NA
##   R05A.10 R05A.11 R05A.12 R05A.13 R05A.14 R05A.15 R05A.16 R05A.17 R05A.18
## 1      20      21      NA      NA      NA      NA      NA      NA      NA
## 2      NA      NA      NA      NA      NA      NA      NA      NA      NA
## 3      NA      NA      NA      NA      NA      NA      NA      NA      NA
## 4      NA      NA      NA      NA      NA      NA      NA      NA      NA
## 5      NA      NA      NA      NA      NA      NA      NA      NA      NA
## 6      NA      NA      NA      NA      NA      NA      NA      NA      NA
##   R05A.19 R05A.20 Output
## 1      NA      NA     NA
## 2      NA      NA     NA
## 3      NA      NA     NA
## 4      NA      NA     NA
## 5      NA      NA     NA
## 6      NA      NA     NA
```

```
# Loop through each row in the channel data frame.
for(row in 1:nrow(channel))
{
  if(21 %in% channel[row,]){channel$convert[row] = 1} # Check whether the value 21 is present in the cu
}
column = colnames(channel)
channel$path = do.call(paste, c(channel[column], sep = " > "))
# Show the first few rows of the path column to check that it was created correctly.
head(channel$path)
```

```
## [1] "16 > 4 > 3 > 5 > 10 > 8 > 6 > 8 > 13 > 20 > 21 > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA
## [2] "2 > 1 > 9 > 10 > 1 > 4 > 3 > 21 > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA
## [3] "9 > 13 > 20 > 16 > 15 > 21 > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA
## [4] "8 > 15 > 20 > 21 > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA
## [5] "16 > 9 > 13 > 20 > 21 > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA
## [6] "1 > 11 > 8 > 4 > 9 > 21 > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA > NA >
```

Create a new variable called column containing the column names of the channel data frame. Then, create a new column called path by applying the paste() function to all the columns in each row. The do.call() function is used to pass all the columns as separate arguments to paste(), with the separator " > " specified by the sep argument.

```
for(row in 1:nrow(channel)) {
  channel$path[row] = strsplit(channel$path[row], " > 21")[[1]][1]
}
channel_fin = channel[,c(23,22)]
channel_fin = ddply(channel_fin,~path,summarise, conversion= sum(convert))
head(channel_fin)
```

```
##                           path conversion
## 1               1 > 1 > 1 > 20          1
## 2              1 > 1 > 12 > 12          1
## 3    1 > 1 > 14 > 13 > 12 > 20          1
## 4      1 > 1 > 3 > 13 > 3 > 20          1
## 5          1 > 1 > 3 > 17 > 17          1
## 6 1 > 1 > 6 > 1 > 12 > 20 > 12          1
```

1- In the first loop, the code checks if the value "21" exists in each row of the channel data frame. If "21" is present, then the corresponding value in the "convert" column is set to 1. This assumes that "21" represents the final conversion step in each path.

2- The next line creates a new column called "path" by concatenating all the column values from the first column up to the "convert" column using ">" as a separator.

3- The second loop splits each path in the "path" column by the "> 21" string and only keeps the first part of the path. This is because we are only interested in the path that leads up to the final conversion step.

4- The last two lines create a new data frame called "channel_fin" that only contains the "path" and "convert" columns. The ddply() function is used to group the data by "path" and calculate the sum of the "convert" values for each path. This gives us the total number of conversions for each path. 5- The resulting data frame contains two columns: "path" and "conversion". The "path" column lists all the unique paths in the channel data frame, and the "conversion" column contains the total number of conversions for each path.

```
Data = channel_fin
head(Data)
```

```
##                           path conversion
## 1               1 > 1 > 1 > 20          1
## 2              1 > 1 > 12 > 12          1
## 3    1 > 1 > 14 > 13 > 12 > 20          1
## 4      1 > 1 > 3 > 13 > 3 > 20          1
## 5          1 > 1 > 3 > 17 > 17          1
## 6 1 > 1 > 6 > 1 > 12 > 20 > 12          1
```

```
H <- heuristic_models(Data, 'path', 'conversion', var_value='conversion')
H
```

```
##    channel_name first_touch_conversions first_touch_value
## 1             1                     130               130
## 2            20                       0                 0
## 3            12                      75                75
## 4            14                      34                34
## 5            13                     320               320
## 6             3                     168               168
## 7            17                      31                31
## 8             6                      50                50
## 9             8                      56                56
## 10           10                     547               547
## 11           11                      66                66
## 12           16                     111               111
## 13            2                     199               199
## 14            4                     231               231
## 15            7                      26                26
## 16            5                      62                62
## 17            9                     250               250
## 18           15                      22                22
## 19           18                       4                 4
## 20           19                      10                10
##    last_touch_conversions last_touch_value linear_touch_conversions
## 1                      18               18                73.773661
## 2                    1701             1701               473.998171
## 3                      23               23                76.127863
## 4                      25               25                56.335744
## 5                      76               76               204.039552
## 6                      21               21               117.609677
## 7                      47               47                76.583847
## 8                      20               20                54.707124
## 9                      17               17                53.677862
## 10                     42               42               211.822393
## 11                     33               33               107.109048
## 12                     95               95               156.049086
## 13                     18               18                94.111668
## 14                     88               88               250.784033
## 15                     15               15                33.435991
## 16                     23               23                74.900402
## 17                     71               71               194.071690
```

```
## 18                    47              47               65.159225
## 19                     2               2                5.026587
## 20                    10              10               12.676375
##    linear_touch_value
## 1           73.773661
## 2          473.998171
## 3           76.127863
## 4           56.335744
## 5          204.039552
## 6          117.609677
## 7           76.583847
## 8           54.707124
## 9           53.677862
## 10         211.822393
## 11         107.109048
## 12         156.049086
## 13          94.111668
## 14         250.784033
## 15          33.435991
## 16          74.900402
## 17         194.071690
## 18          65.159225
## 19           5.026587
## 20          12.676375
```

```
M <- markov_model(Data, 'path', 'conversion', var_value='conversion', order = 1)
```

```
##
## Number of simulations: 100000 - Convergence reached: 2.05% < 5.00%
##
## Percentage of simulated paths that successfully end before maximum number of steps (17) is reached: 9
```

```
M
```

```
##    channel_name total_conversion total_conversion_value
## 1             1        82.805970              82.805970
## 2            20       439.582090             439.582090
## 3            12        81.253731              81.253731
## 4            14        64.238806              64.238806
## 5            13       197.791045             197.791045
## 6             3       122.328358             122.328358
## 7            17        86.985075              86.985075
## 8             6        58.985075              58.985075
## 9             8        60.656716              60.656716
## 10           10       209.850746             209.850746
## 11           11       115.402985             115.402985
## 12           16       159.820896             159.820896
## 13            2        97.074627              97.074627
## 14            4       222.149254             222.149254
## 15            7        40.597015              40.597015
## 16            5        80.537313              80.537313
## 17            9       178.865672             178.865672
## 18           15        72.358209              72.358209
```

```
## 19           18         6.567164                6.567164
## 20           19        14.149254               14.149254
```

### Merges the two data frames on the "channel_name" column.

```
R <- merge(H, M, by='channel_name')
```

### Select only relevant columns

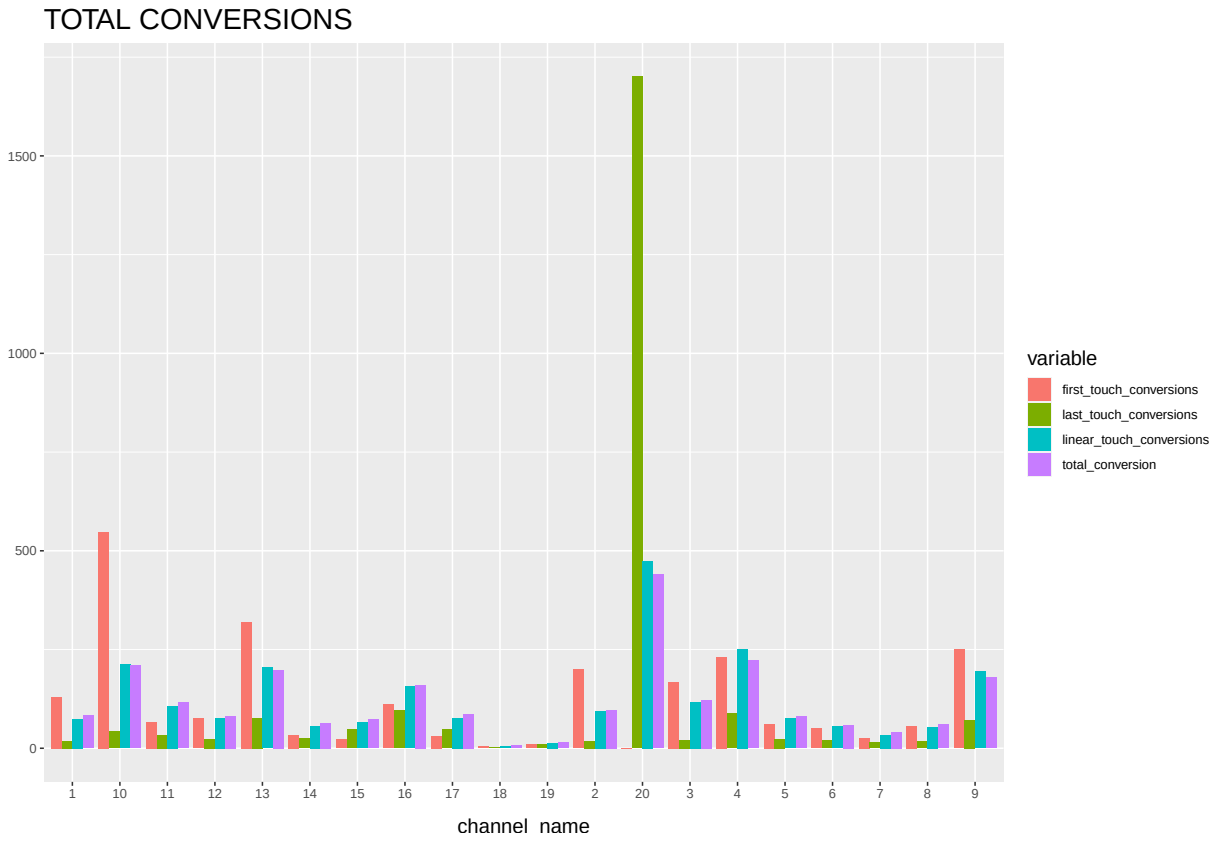
```
R1 <- R[, (colnames(R) %in% c('channel_name', 'first_touch_conversions',
                              'last_touch_conversions', 'linear_touch_conversions',
                              'total_conversion'))]
```

### Transforms the dataset into a data frame that ggplot2 can use to plot

```
# plot the outcomes
R1 <- melt(R1, id='channel_name')
```

### Plot the total conversions

```
ggplot(R1, aes(channel_name, value, fill = variable)) +
  geom_bar(stat='identity', position='dodge') +
  ggtitle('TOTAL CONVERSIONS') +
  theme(axis.title.x = element_text(vjust = -2)) +
  theme(axis.title.y = element_text(vjust = +2)) +
  theme(title = element_text(size = 14)) +
  theme(plot.title=element_text(size = 20)) +
  ylab("")
```


TOTAL CONVERSIONS
1500
1000
500
0
1
10
11
12
13
14
15
16
17
18
19
2
20
3
4
5
6
7
8
9
channel name
variable
first_touch_conversions
last_touch_conversions
linear_touch_conversions
total_conversion