ChannelAttribution package review

Overview

This vignette will present at a high-level exploration of the `ChannelAttribution` package. We will then compare ChannelAttribution to fractribution and last touch approaches.

First let's load the required package:

```
library(ChannelAttribution)
library(dplyr)
library(tidyr)
```

Package data

There is a pre-loaded dataset within the package (see `?ChannelAttribution::Data` for details).

```
data(PathData)
customer_paths <- as_tibble(Data)
```

Inspect the data:

```
customer_paths %>% glimpse()
#> Observations: 10,000
#> Variables: 4
#> $ path                  <chr> "eta > iota > alpha > eta", "iota > iot...
#> $ total_conversions     <int> 1, 2, 2, 1, 0, 1, 0, 1, 1, 1, 10, 1, 1,...
#> $ total_conversion_value <dbl> 0.24400, 3.19500, 6.75400, 2.40200, 0.0...
#> $ total_null            <dbl> 3, 6, 6, 3, 2, 3, 2, 2, 3, 3, 38, 3, 3,...
```

Let's inspect one of these, see that different conversion values mean multiple rows for the same path:

```
customer_paths %>% filter(path == 'beta > beta > eta')
#> # A tibble: 97 x 4
#>    path              total_conversions total_conversion_value total_null
#>    <chr>                         <int>                  <dbl>      <dbl>
#>  1 beta > beta > eta                 1                   8.72          2
#>  2 beta > beta > eta                 1                   2.56          3
#>  3 beta > beta > eta                 1                   6.81          2
#>  4 beta > beta > eta                 1                   5.68          2
#>  5 beta > beta > eta                 1                  10.6           3
#>  6 beta > beta > eta                 1                   3.54          2
#>  7 beta > beta > eta                 1                   4.45          3
#>  8 beta > beta > eta                 1                   3.94          2
#>  9 beta > beta > eta                 1                   7.51          3
#> 10 beta > beta > eta                 1                   1.3           3
#> # ... with 87 more rows
```

Channel

Let's extract the set of **channels**:

```
customer_paths %>%
  mutate(split_paths = stringr::str_split(path, " > ")) %>%
  select(split_paths) %>%
  unnest(split_paths) %>%
  count(split_paths, sort = TRUE)
#> # A tibble: 12 x 2
#>    split_paths     n
#>    <chr>       <int>
#>  1 iota        15593
#>  2 beta        10982
#>  3 eta         10219
#>  4 alpha        9674
#>  5 theta        4527
#>  6 lambda       4453
#>  7 epsilon      1561
#>  8 zeta         1187
#>  9 kappa         832
#> 10 gamma         411
#> 11 delta          13
#> 12 mi              2
```

Note this is a toy dataset.

Fit the Markov Model for attribution

There is a `markov_model()` function to do this. For the help page:

```
?markov_model
```

Order of Markov Model

A Markov model determines the probability that a user will transition from Sequence A to Sequence B based on the steps that each user takes through a site. The contents of these sequences are determined by the Markov order, which ranges from 0 to 4. Here are some guidelines to **determine what Markov Order is appropriate**:

- **Order 0**: Doesn't know where the user came from or what step the user is on, only the probability of going to any page.
- **Order 1**: Looks back zero steps. You are currently at Step A (Sequence A). The probability of going anywhere is based on being at that step.
- **Order 2**: Looks back one step. You came from Step A (Sequence A) and are currently at Step B (Sequence B). The probability of going anywhere is based on where you were and where you are.
- **Order 3**: Looks back two steps. You came from Step A > B (Sequence A) and are currently at Step C (Sequence B). The probability of going anywhere is based on where you were and where you are.
- **Order 4**: Looks back three steps. You came from Step A > B > C (Sequence A) and are currently at Step D (Sequence B). The probability of going anywhere is based on where you were and where you are.

According to the paper this package is based on (see help file) the greatest increase in accuracy is realized moving between second and third order, in contrast to the marginal lift moving between the third and fourth.

So let's go with **order 3**.

Fit

To fit we apply the function as follows:

```
mod <- markov_model(customer_paths, var_path = 'path',
                    var_conv = 'total_conversions',
                    var_value = 'total_conversion_value',
                    var_null = 'total_null', out_more = TRUE,
                    order = 3, sep = ">", seed = 101)
```

Get the results

Since we set `out_more = TRUE` this will return a list with the attribution results, the transition matrix generated and the removal effects.

```
# Inspect
str(mod)
#> List of 3
#>  $ result           :'data.frame':   12 obs. of  3 variables:
#>   ..$ channel_name          : Factor w/ 12 levels "alpha","beta",..: 5 7 1 2 11 9 8 12 4 6 ...
#>   ..$ total_conversions     : num [1:12] 3400 3785 5758 2408 1854 ...
#>   ..$ total_conversion_value: num [1:12] 12976 14786 19410 9810 7275 ...
#>  $ transition_matrix:'data.frame':   5604 obs. of  3 variables:
#>   ..$ channel_from          : Factor w/ 934 levels "(start)","alpha,alpha,alpha",..: 1 1 1 1 1 1 1 1
#>   ..$ channel_to            : Factor w/ 935 levels "(conversion)",..: 348 533 57 139 516 36 488 164 
#>   ..$ transition_probability: num [1:5604] 0.000622 0.048491 0.000724 0.020591 0.000419 ...
#>  $ removal_effects  :'data.frame':   12 obs. of  3 variables:
#>   ..$ channel_name                   : Factor w/ 12 levels "alpha","beta",..: 5 7 1 2 11 9 8 12 4 6
#>   ..$ removal_effects_conversion     : num [1:12] 0.323 0.359 0.547 0.229 0.176 ...
#>   ..$ removal_effects_conversion_value: num [1:12] 0.347 0.396 0.52 0.263 0.195 ...
```

Channel attribution report

We can extract a channel attribution report from the `mod` object.

```
mod$result %>%
  mutate(perc_contribution = round(total_conversions / sum(total_conversions), 2))
#>    channel_name total_conversions total_conversion_value perc_contribution
#> 1           eta       3400.039298            12976.20247              0.17
#> 2          iota       3784.914764            14786.41908              0.19
#> 3         alpha       5757.501571            19410.13205              0.29
#> 4          beta       2407.602012             9809.60864              0.12
#> 5         theta       1854.464171             7275.30842              0.09
#> 6        lambda       1138.948901             4732.70024              0.06
#> 7         kappa        263.598499             1112.94525              0.01
#> 8          zeta        368.397617             1405.21573              0.02
#> 9       epsilon        623.898431             2450.37464              0.03
#> 10        gamma        179.184835              825.13236              0.01
#> 11        delta          4.566718               13.34864              0.00
#> 12           mi          1.883183                5.58406              0.00
```

Compare ChannelAttribution with Fractribution

Let's compare the channel attribution model with fractribution and last touch.

Run it all on the fractribution sample dataset:

```
# Load example data from Fractribution
path_summary <- fractribution.model::example_path_summary
```

Run channel attribution:

```
mod_2 <- markov_model(path_summary, var_path = 'path',
                      var_conv = 'total_conversions',
                      var_null = 'total_non_conversions',
                      out_more = TRUE, order = 3, sep = ">",
                      seed = 101)
```

Inspect:

```
# Put in variable
channel_attribution <- mod_2$result %>%
  mutate(channel_name = as.character(channel_name),
         total_conversions = round(total_conversions))

# Inspect
channel_attribution
#>    channel_name total_conversions
#> 1             B               669
#> 2             A               627
#> 3             C               355
#> 4             D               270
#> 5             H                29
#> 6             G                18
#> 7             I                 6
#> 8             F                 7
#> 9             E                 2
#> 10            J                 0
#> 11            K                 0
```

Run Fractribution:

```
# Load library
library(fractribution.model)

# Fit
fractional_attribution <- attribution_fit(example_path_summary,
                                          example_path_customer_map)

# Aggregate
fractribution <- channel_attribution_report(fractional_attribution)

# Inspect
fractribution
#> # A tibble: 11 x 2
#>    channel attributed_conversions
#>    <chr>                   <dbl>
#>  1 b                         683
#>  2 a                         644
```

```
#>  3 c                          344
#>  4 d                          255
#>  5 h                           25
#>  6 g                           17
#>  7 i                            8
#>  8 f                            5
#>  9 e                            2
#> 10 j                            0
#> 11 k                            0
```

And run last touch:

```
# Fit
last_touch <- last_touch_attribution(example_path_summary)

# Inspect
last_touch
#> # A tibble: 11 x 2
#>    last_channel last_touch_conversions
#>    <chr>                         <dbl>
#>  1 B                               655
#>  2 A                               648
#>  3 C                               361
#>  4 D                               258
#>  5 H                                28
#>  6 G                                17
#>  7 I                                 8
#>  8 F                                 6
#>  9 E                                 2
#> 10 J                                 0
#> 11 K                                 0
```

Join all together to compare:

```
# Make fractribution channels upper case
fractribution <- fractribution %>%
    mutate(channel = stringr::str_to_upper(channel))

# Join all together
combined_channel_report <- fractribution %>%
  inner_join(channel_attribution, by = c('channel' = 'channel_name')) %>%
  inner_join(last_touch, by = c('channel' = 'last_channel')) %>%
  select(channel,
         fractribution = attributed_conversions,
         channel_attribution = total_conversions,
         last_touch = last_touch_conversions)

# Inspect
combined_channel_report
#> # A tibble: 11 x 4
#>    channel fractribution channel_attribution last_touch
#>    <chr>           <dbl>               <dbl>      <dbl>
#>  1 B                 683                 669        655
#>  2 A                 644                 627        648
#>  3 C                 344                 355        361
```

```
#>  4 D                255                270                258
#>  5 H                 25                 29                 28
#>  6 G                 17                 18                 17
#>  7 I                  8                  6                  8
#>  8 F                  5                  7                  6
#>  9 E                  2                  2                  2
#> 10 J                  0                  0                  0
#> 11 K                  0                  0                  0
```

Little differences, same ranking. No harm running all and comparing.