AM6

SA

12/03/2020

R Markdown

Loading Libraries

```r
library(tidyverse)
```

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

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------- tidyverse_con
## 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(ggthemes)
library(ggrepel)
library(RColorBrewer)
library(ChannelAttribution)
library(markovchain)
```

```
## Package:  markovchain
## Version:  0.8.2
## Date:     2020-01-10
## BugReport: http://github.com/spedygiorgio/markovchain/issues
```

```r
library(visNetwork)
library(expm)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
##
## Attaching package: 'expm'

## The following object is masked from 'package:Matrix':
##
##     expm
```

```
library(stringr)
library(purrrlyr)
```

#Generate and prepare data

```
set.seed(99669966)
df_raw <- data.frame(customer_id = paste0('id', sample(c(1:20000), replace = TRUE)), date = as.Date(rbe
  group_by(customer_id) %>%
  mutate(conversion = sample(c(0, 1), n(), prob = c(0.975, 0.025), replace = TRUE)) %>%
  ungroup() %>%
  dmap_at(c(1, 3), as.character) %>%
  arrange(customer_id, date)

df_raw
```

```
## # A tibble: 80,000 x 4
##    customer_id date       channel   conversion
##    <chr>       <date>     <chr>          <dbl>
##  1 id10        2016-01-01 channel_5          0
##  2 id10        2016-01-01 channel_1          0
##  3 id10        2016-01-03 channel_1          0
##  4 id10        2016-01-07 channel_1          0
##  5 id10        2016-01-09 channel_5          0
##  6 id10        2016-01-14 channel_5          0
##  7 id10        2016-01-18 channel_5          0
##  8 id10        2016-01-22 channel_4          0
##  9 id10003     2016-01-01 channel_1          0
## 10 id10003     2016-01-02 channel_4          0
## # ... with 79,990 more rows
```

```
df_raw <- df_raw %>%
  mutate(channel = ifelse(channel == 'channel_2', NA, channel))



df_paths <- df_raw %>%
  group_by(customer_id) %>%
  mutate(path_no = ifelse(is.na(lag(cumsum(conversion))), 0, lag(cumsum(conversion))) + 1) %>%
  ungroup()

df_paths
```

```
## # A tibble: 80,000 x 5
##    customer_id date       channel   conversion path_no
##    <chr>       <date>     <chr>          <dbl>   <dbl>
##  1 id10        2016-01-01 channel_5          0       1
##  2 id10        2016-01-01 channel_1          0       1
##  3 id10        2016-01-03 channel_1          0       1
##  4 id10        2016-01-07 channel_1          0       1
##  5 id10        2016-01-09 channel_5          0       1
##  6 id10        2016-01-14 channel_5          0       1
##  7 id10        2016-01-18 channel_5          0       1
##  8 id10        2016-01-22 channel_4          0       1
##  9 id10003     2016-01-01 channel_1          0       1
## 10 id10003     2016-01-02 channel_4          0       1
## # ... with 79,990 more rows
```

For first purchaser only

```
df_paths_1 <- df_paths %>%
  filter(path_no == 1) %>%
  select(-path_no)

df_paths_1
```

```
## # A tibble: 73,872 x 4
##    customer_id date       channel   conversion
##    <chr>       <date>     <chr>          <dbl>
##  1 id10        2016-01-01 channel_5          0
##  2 id10        2016-01-01 channel_1          0
##  3 id10        2016-01-03 channel_1          0
##  4 id10        2016-01-07 channel_1          0
##  5 id10        2016-01-09 channel_5          0
##  6 id10        2016-01-14 channel_5          0
##  7 id10        2016-01-18 channel_5          0
##  8 id10        2016-01-22 channel_4          0
##  9 id10003     2016-01-01 channel_1          0
## 10 id10003     2016-01-02 channel_4          0
## # ... with 73,862 more rows
```

replace some channels

```
##### replace some channels #####
df_path_1_clean <- df_paths_1 %>%
  # removing NAs
  filter(!is.na(channel)) %>%

  # adding order of channels in the path
  group_by(customer_id) %>%
  mutate(ord = c(1:n()),
         is_non_direct = ifelse(channel == 'channel_6', 0, 1),
         is_non_direct_cum = cumsum(is_non_direct)) %>%

  # removing Direct (channel_6) when it is the first in the path
  filter(is_non_direct_cum != 0) %>%

  # replacing Direct (channel_6) with the previous touch point
  mutate(channel = ifelse(channel == 'channel_6', channel[which(channel != 'channel_6')][is_non_direct_c

  ungroup() %>%
  select(-ord, -is_non_direct, -is_non_direct_cum)
```

```r
df_path_1_clean
```

```
## # A tibble: 70,181 x 4
##    customer_id date       channel   conversion
##    <chr>       <date>     <chr>          <dbl>
##  1 id10        2016-01-01 channel_5          0
##  2 id10        2016-01-01 channel_1          0
##  3 id10        2016-01-03 channel_1          0
##  4 id10        2016-01-07 channel_1          0
##  5 id10        2016-01-09 channel_5          0
##  6 id10        2016-01-14 channel_5          0
##  7 id10        2016-01-18 channel_5          0
##  8 id10        2016-01-22 channel_4          0
##  9 id10003     2016-01-01 channel_1          0
## 10 id10003     2016-01-02 channel_4          0
## # ... with 70,171 more rows
```

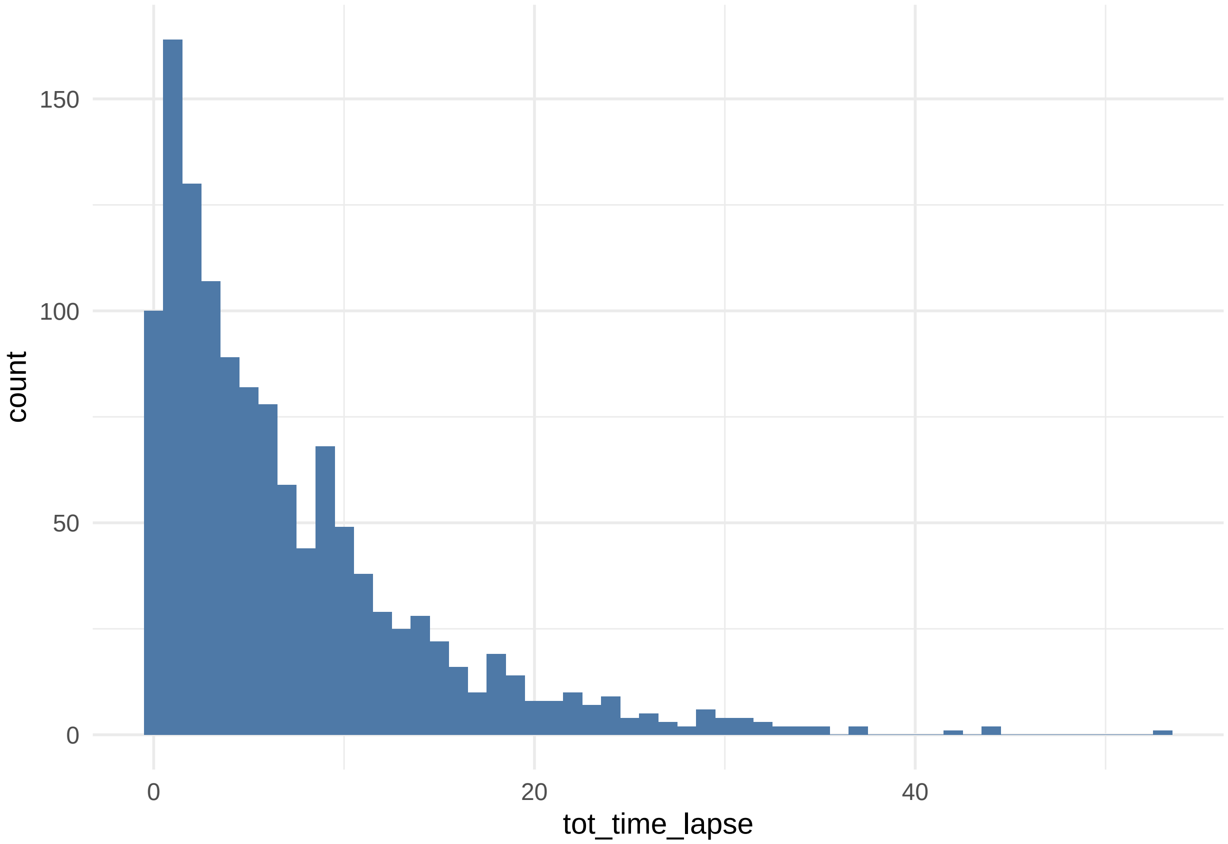
```r
df_path_1_clean <- df_path_1_clean %>%
  group_by(customer_id) %>%
  mutate(uniq_channel_tag = ifelse(length(unique(channel)) == 1, TRUE, FALSE)) %>%
  ungroup()


df_path_1_clean_multi <- df_path_1_clean %>%
  filter(uniq_channel_tag == FALSE) %>%
  select(-uniq_channel_tag)
```

computing time lapses from the first contact to conversion/last contact

```r
# computing time lapses from the first contact to conversion/last contact
df_multi_paths_tl <- df_path_1_clean_multi %>%
  group_by(customer_id) %>%
  summarise(path = paste(channel, collapse = ' > '),
            first_touch_date = min(date),
            last_touch_date = max(date),
            tot_time_lapse = round(as.numeric(last_touch_date - first_touch_date)),
            conversion = sum(conversion)) %>%
  ungroup()
```

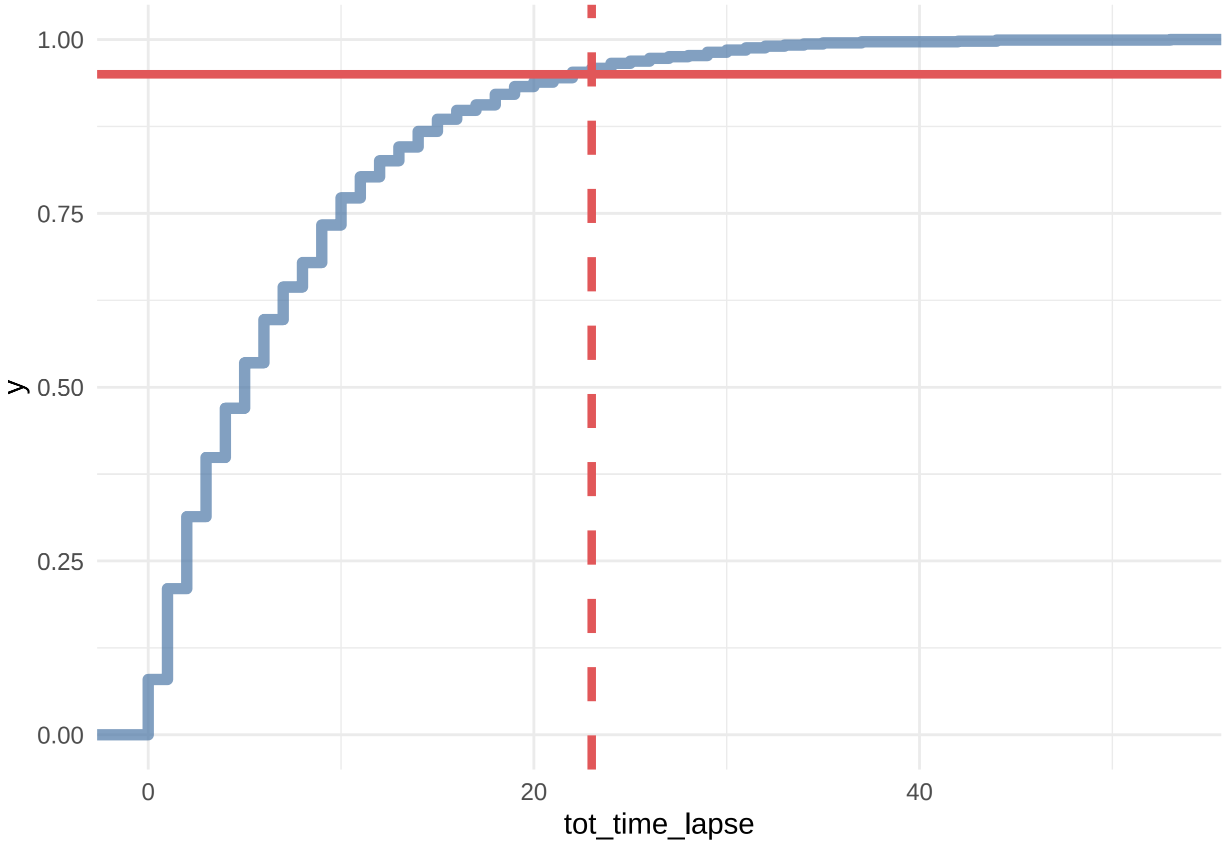
distribution plot

```r
ggplot(df_multi_paths_tl %>% filter(conversion == 1), aes(x = tot_time_lapse)) +
  theme_minimal() +
  geom_histogram(fill = '#4e79a7', binwidth = 1)
```



cumulative distribution plot

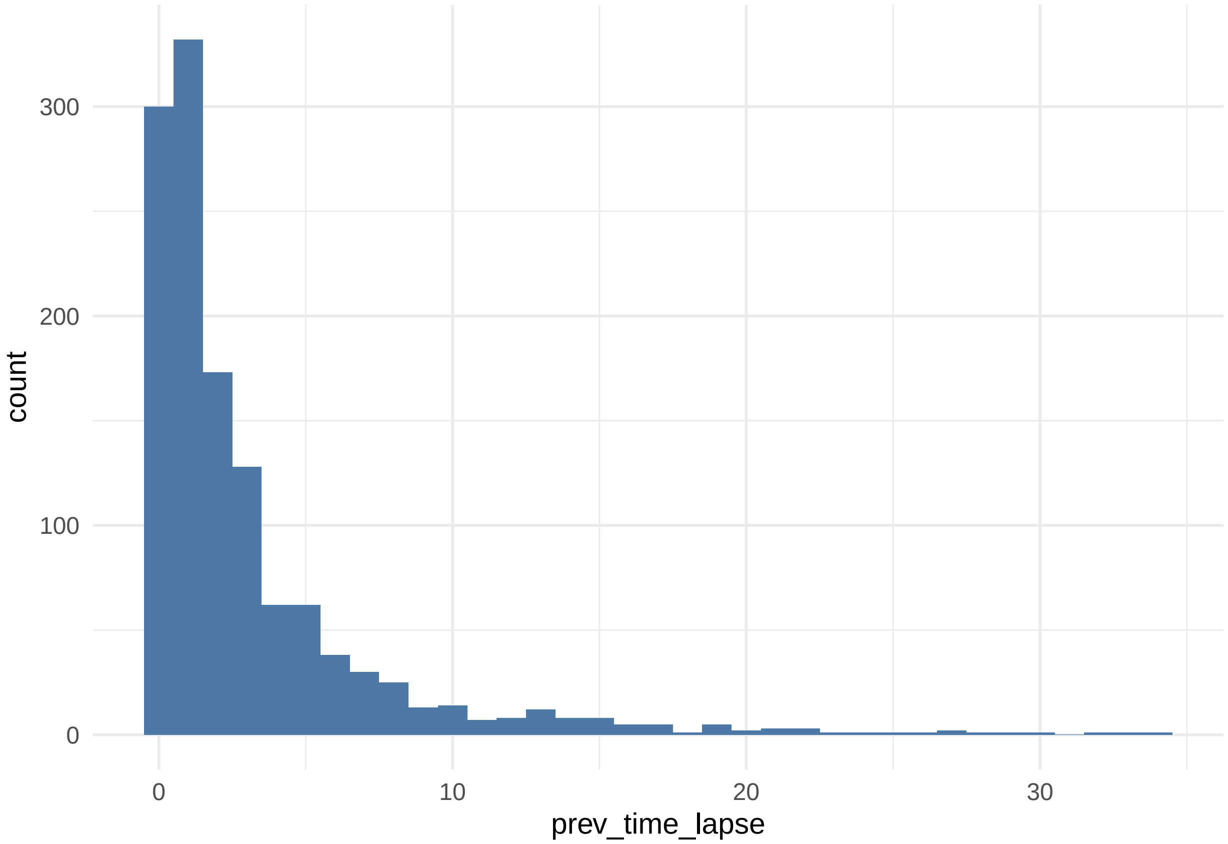
```
ggplot(df_multi_paths_tl %>% filter(conversion == 1), aes(x = tot_time_lapse)) +
  theme_minimal() +
  stat_ecdf(geom = 'step', color = '#4e79a7', size = 2, alpha = 0.7) +
  geom_hline(yintercept = 0.95, color = '#e15759', size = 1.5) +
  geom_vline(xintercept = 23, color = '#e15759', size = 1.5, linetype = 2)
```



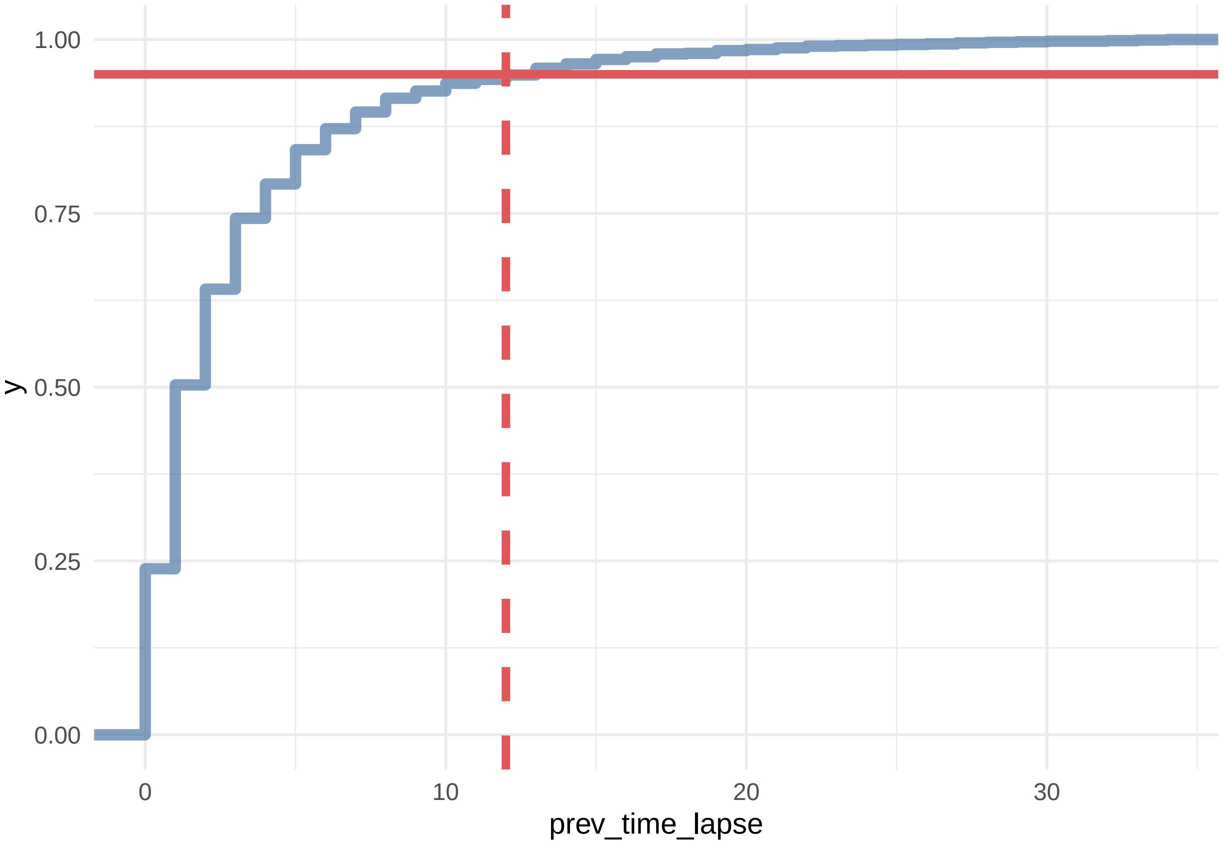
data for

1) time lapse from the first contact,
2) time lapse between the conversion date and a previous contact.

```
#1) time lapse from the first contact,

#2) time lapse between the conversion date and a previous contact.


df_multi_paths_tl_2 <- df_path_1_clean_multi %>%
  group_by(customer_id) %>%
  mutate(prev_touch_date = lag(date)) %>%
  ungroup() %>%
  filter(conversion == 1) %>%
  mutate(prev_time_lapse = round(as.numeric(date - prev_touch_date)))

# distribution
ggplot(df_multi_paths_tl_2, aes(x = prev_time_lapse)) +
  theme_minimal() +
  geom_histogram(fill = '#4e79a7', binwidth = 1)
```



```
# cumulative distribution
ggplot(df_multi_paths_tl_2, aes(x = prev_time_lapse)) +
  theme_minimal() +
  stat_ecdf(geom = 'step', color = '#4e79a7', size = 2, alpha = 0.7) +
  geom_hline(yintercept = 0.95, color = '#e15759', size = 1.5) +
  geom_vline(xintercept = 12, color = '#e15759', size = 1.5, linetype = 2)
```



#subsetting data for For tot_time_lapse > 20 & prev_touch > 10

```
#For tot_time_lapse > 20 & prev_touch > 10


df_multi_paths_tl_3 <- df_path_1_clean_multi %>%
  group_by(customer_id) %>%
  mutate(prev_time_lapse = round(as.numeric(date - lag(date)))) %>%
  summarise(path = paste(channel, collapse = ' > '),
            tot_time_lapse = round(as.numeric(max(date) - min(date))),
            prev_touch_tl = prev_time_lapse[which(max(date) == date)],
            conversion = sum(conversion)) %>%
  ungroup() %>%
  mutate(is_fruitless = ifelse(conversion == 0 & tot_time_lapse > 20 & prev_touch_tl > 10, TRUE, FALSE))
  filter(conversion == 1 | is_fruitless == TRUE)

df_multi_paths_tl_3
```

```
## # A tibble: 3,033 x 6
##    customer_id path         tot_time_lapse prev_touch_tl conversion is_fruitless
##    <chr>       <chr>                 <dbl>         <dbl>      <dbl> <lgl>
##  1 id10010     channel_3 >~             22            13          0 TRUE
##  2 id10012     channel_0 >~             31            21          0 TRUE
##  3 id10017     channel_5 >~             33            27          0 TRUE
##  4 id10020     channel_1 >~              4             0          1 FALSE
##  5 id10032     channel_0 >~             14            14          1 FALSE
##  6 id10033     channel_1 >~             24            19          0 TRUE
##  7 id10034     channel_5 >~              0             0          1 FALSE
##  8 id10039     channel_7 >~               35            11           0 TRUE
##  9 id10041     channel_7 >~                1             1           1 FALSE
## 10 id10042     channel_4 >~                7             3           1 FALSE
## # ... with 3,023 more rows
```

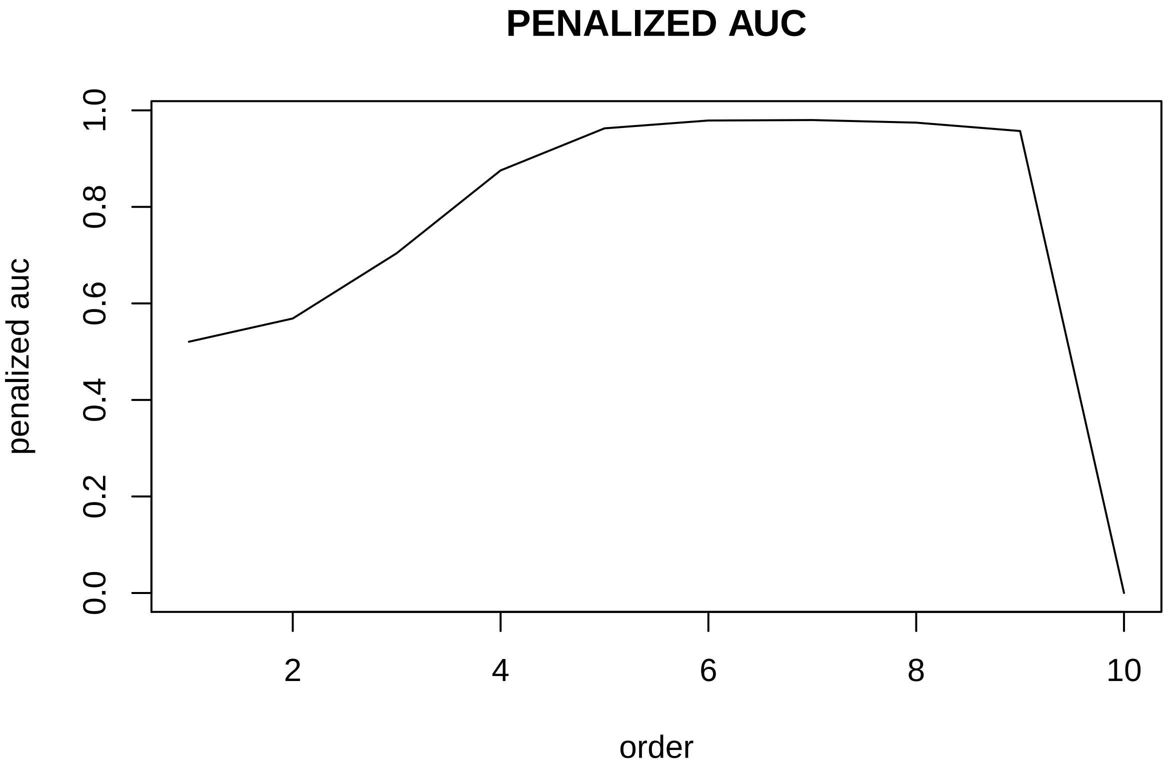
#models for #multi-channel paths only for the reporting period (e.g. 90 days) in the example. Therefore, paths include a minimum of 2 touches with 2 dates and last touch date is equal to conversion date. 1) Criteria #1 - 23 days period between those 2 or more dates (between 1st and last touches) covers 95% of customers with conversions. 2) Criteria #2 - 12 days period between last 2 touches (between last and previous touches) covers 95% of customers with conversions.

```
##### Generic Probabilistic Model #####
df_all_paths_compl <- df_multi_paths_tl_3 %>%

mutate(null_conversion = ifelse(conversion == 1, 0, 1))
```

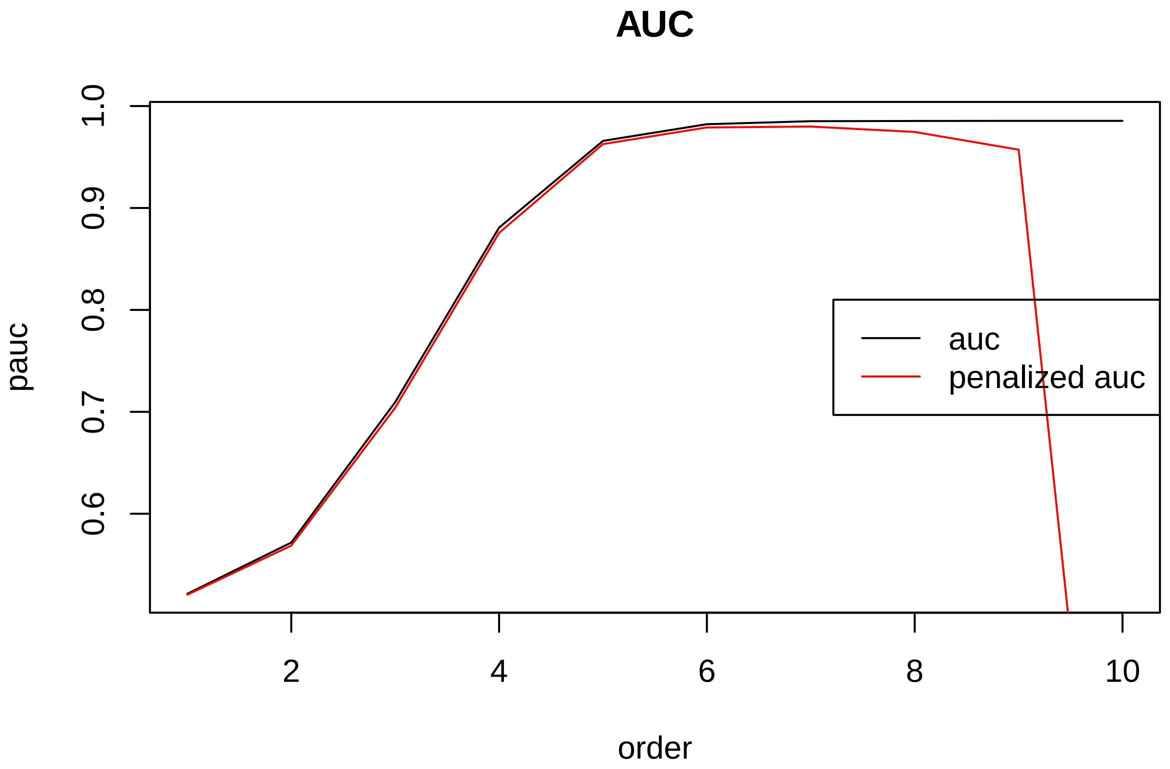
#Finding the order of the model

```
res=choose_order(df_all_paths_compl, var_path="path", var_conv="conversion",
                 var_null="null_conversion")
```

```
## [1] "Suggested order: 7"
```



```
#plot auc and penalized auc
plot(res$auc$order,res$auc$auc,type="l",xlab="order",ylab="pauc",main="AUC")
lines(res$auc$order,res$auc$pauc,col="red")
legend("right", legend=c("auc","penalized auc"),
       col=c("black","red"),lty=1)
```



```
## End(Not run)
```

#model with order = 1 and best order

```
mod_attrib_complete <- markov_model(
  df_all_paths_compl,
  var_path = 'path',
  var_conv = 'conversion',
  var_null = 'null_conversion', order = 1,
  out_more = TRUE
)

mod_attrib_complete_best <- markov_model(
  df_all_paths_compl,
  var_path = 'path',
  var_conv = 'conversion',
  var_null = 'null_conversion', order = res$suggested_order,
  out_more = TRUE
)
```

#For Model with order = 1

```
trans_matrix_prob <- mod_attrib_complete$transition_matrix %>%
  dmap_at(c(1, 2), as.character)

trans_matrix_prob
```

```
##    channel_from   channel_to transition_probability
## 1       (start)    channel_3             0.07088691
## 2       (start)    channel_0             0.22551929
## 3       (start)    channel_5             0.26640290
## 4       (start)    channel_1             0.15100561
## 5       (start)    channel_7           0.10121991
## 6       (start)    channel_4           0.18496538
## 7     channel_3    channel_5           0.23040000
## 8     channel_3       (null)           0.12080000
## 9     channel_3 (conversion)           0.09120000
## 10    channel_3    channel_4           0.16960000
## 11    channel_3    channel_1           0.12240000
## 12    channel_3    channel_0           0.19280000
## 13    channel_3    channel_7           0.07280000
## 14    channel_5    channel_4           0.18938347
## 15    channel_5    channel_0           0.24218966
## 16    channel_5       (null)           0.14597733
## 17    channel_5    channel_1           0.14929500
## 18    channel_5    channel_7           0.09538291
## 19    channel_5 (conversion)           0.09344761
## 20    channel_5    channel_3           0.08432403
## 21    channel_4    channel_0           0.21947326
## 22    channel_4    channel_5           0.26695930
## 23    channel_4       (null)           0.13048683
## 24    channel_4    channel_3           0.08379888
## 25    channel_4 (conversion)           0.09138069
## 26    channel_4    channel_7           0.08499601
## 27    channel_4    channel_1           0.12290503
## 28    channel_0    channel_1           0.14014175
## 29    channel_0    channel_4           0.16784794
## 30    channel_0       (null)           0.12822165
## 31    channel_0    channel_5           0.29832474
## 32    channel_0 (conversion)           0.08988402
## 33    channel_0    channel_3           0.08569588
## 34    channel_0    channel_7           0.08988402
## 35    channel_1    channel_3           0.08458921
## 36    channel_1    channel_7           0.08410306
## 37    channel_1    channel_5           0.26835197
## 38    channel_1    channel_0           0.21876519
## 39    channel_1    channel_4           0.15216334
## 40    channel_1 (conversion)           0.08507535
## 41    channel_1       (null)           0.10695187
## 42    channel_7    channel_0           0.21519886
## 43    channel_7    channel_5           0.26562500
## 44    channel_7    channel_3           0.05681818
## 45    channel_7 (conversion)           0.08593750
## 46    channel_7    channel_1           0.11576705
## 47    channel_7    channel_4           0.15198864
## 48    channel_7       (null)           0.10866477
```

```
mod_attrib_complete$removal_effects
```

```
##   channel_name removal_effects
## 1    channel_3       0.3267779
## 2    channel_5       0.6742605
## 3    channel_4       0.5434039
## 4    channel_0       0.6186207
## 5    channel_1       0.4780710
## 6    channel_7       0.3625341
```

```
mod_attrib_complete$result
```

```
##   channel_name total_conversions
## 1    channel_3          136.6439
## 2    channel_5          281.9456
## 3    channel_4          227.2273
## 4    channel_0          258.6796
## 5    channel_1          199.9080
## 6    channel_7          151.5956
```

#For Model with order = Best

```
mod_attrib_complete_best$removal_effects
```

```
##   channel_name removal_effects
## 1    channel_3       0.2999831
## 2    channel_5       0.7363272
## 3    channel_4       0.5634169
## 4    channel_0       0.6525260
## 5    channel_1       0.4746096
## 6    channel_7       0.3541746
```

```
mod_attrib_complete_best$result
```

```
##   channel_name total_conversions
## 1    channel_3          122.2896
## 2    channel_5          300.1674
## 3    channel_4          229.6797
## 4    channel_0          266.0054
## 5    channel_1          193.4769
## 6    channel_7          144.3810
```

Calculate ROAS and CPA - for model with order 1

```
calculation = mod_attrib_complete$result

calculation <- data.frame(total_cost = c(15000, 21000, 22000, 10000, 20000, 5000), calculation)

calculation$chanel_weight <- calculation$total_conversions / sum(calculation$total_conversions)

calculation$cost_weight <- calculation$total_cost / sum(calculation$total_cost)

calculation$roas <- calculation$chanel_weight / calculation$cost_weight

calculation$optimal_budget = calculation$total_cost * calculation$roas

calculation$CPA = calculation$total_cost / calculation$total_conversions



calculation$CPA
```

```
## [1] 109.77435  74.48244  96.81936  38.65787 100.04603  32.98249
```

```
calculation$optimal_budget
```

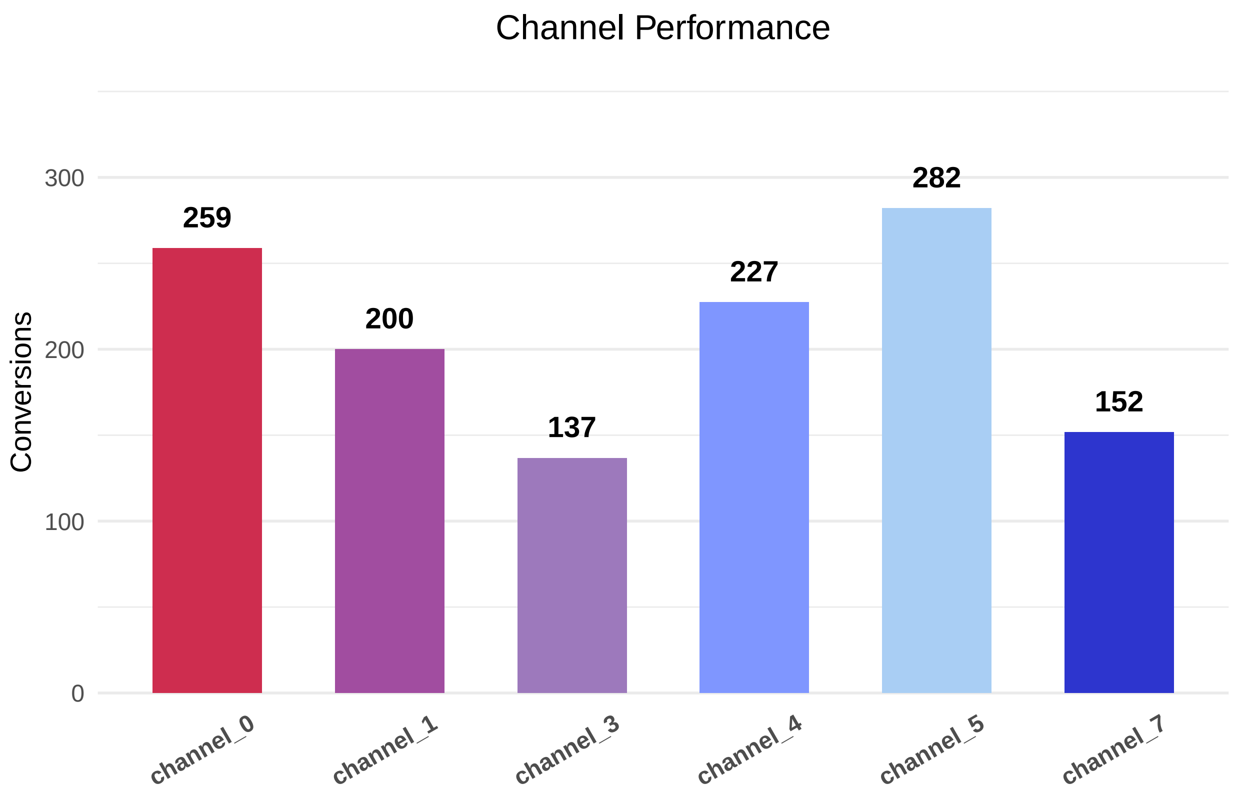
```
## [1] 10117.74 20876.55 16824.95 19153.82 14802.10 11224.83
```

```
calculation$total_cost
```

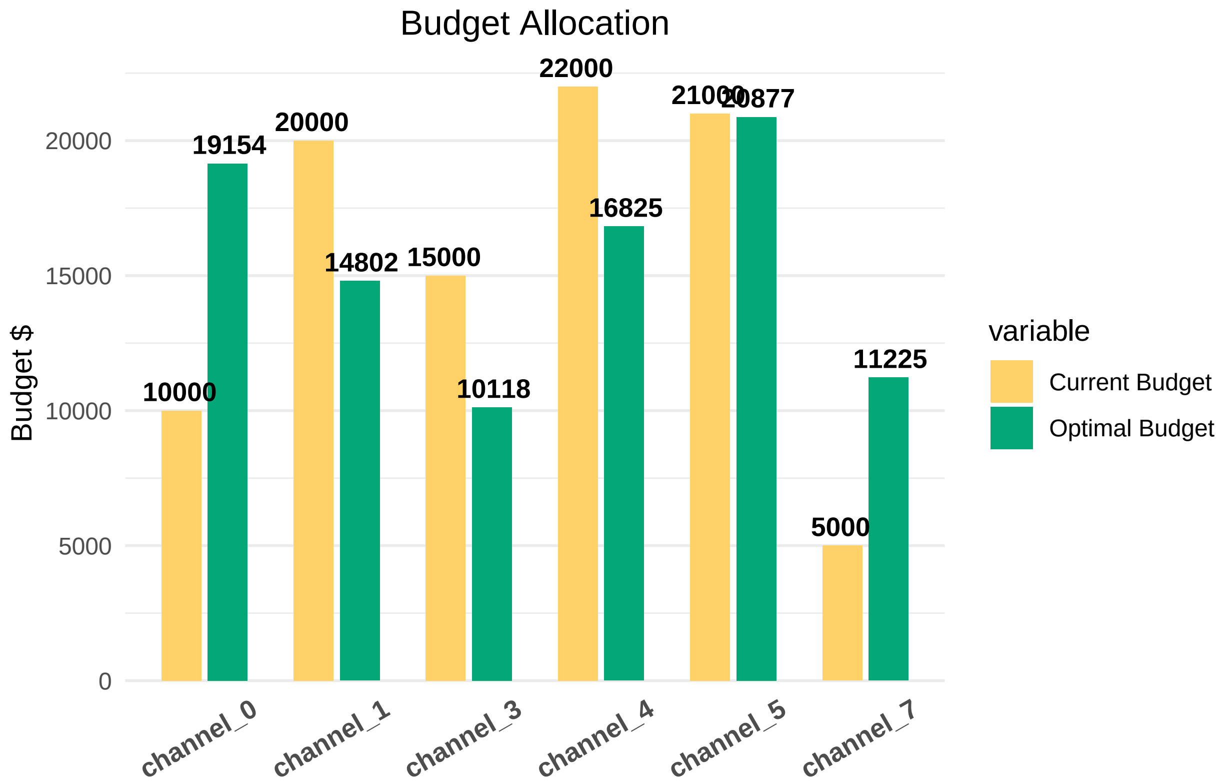
```
## [1] 15000 21000 22000 10000 20000  5000
```

Create an ordered graph showing conversions attributed to each channel

```
# Create an ordered graph showing conversions attributed to each channel
g_channel_performance <- ggplot(calculation, aes(x = channel_name, y = total_conversions, fill = channel
  geom_bar(stat = "identity", width = 0.6) +
  ylim(0, 350) +
  scale_fill_manual(values = c("#CE2D4F",
                               "#A14DA0",
                               "#9D79BC",
                               "#7F96FF",
                               "#A9CEF4",
                               "#2d35ce")) +
  theme_minimal() +
  theme(axis.text.x = element_text(size = 9, angle = 30, hjust = 0.6, face = "bold")) +
  theme(panel.grid.major.x = element_blank()) +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_text(aes(label = round(total_conversions, 0)), fontface = "bold", size = 4, vjust = -1) +
  labs(x = "", y = "Conversions") +
  ggtitle("Channel Performance") +
  guides(fill=FALSE)

g_channel_performance
```

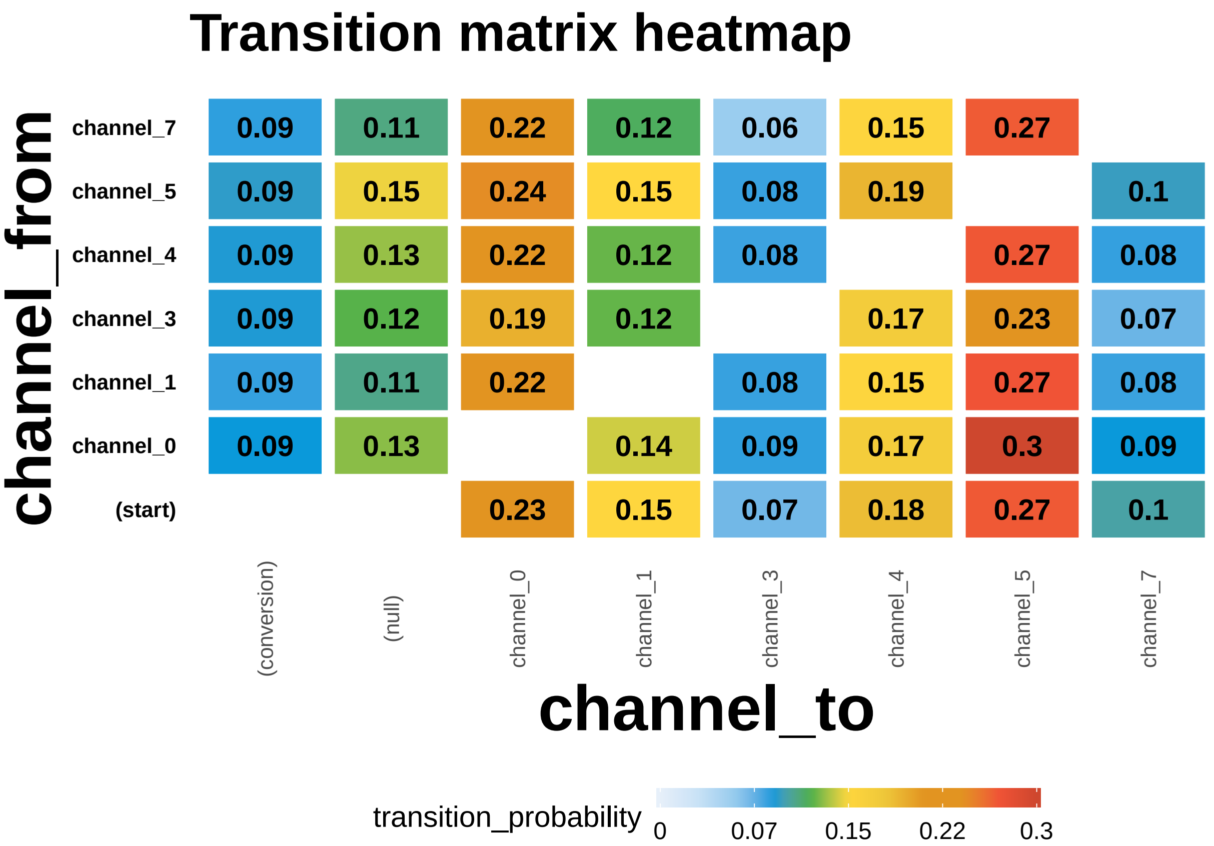


```
df_g2 = calculation[, c("channel_name", "total_cost", "optimal_budget")]
df_g2 = melt(df_g2, id = "channel_name")

g_budget_allocation <- ggplot(df_g2, aes(x = channel_name, y = value, fill = variable)) +
  geom_bar(stat = "identity", width = 0.6, position = position_dodge(width = 0.7)) +
  scale_fill_manual(labels = c("Current Budget", "Optimal Budget"), values = c("#FFD166", "#04A777")) +
  theme_minimal() +
  theme(axis.text.x = element_text(size = 10, angle = 30, hjust = 0.6, face = "bold")) +
  theme(panel.grid.major.x = element_blank()) +
  geom_text(aes(label = round(value, 0)),
            fontface = "bold", size = 3.5,
            vjust = -0.5, position = position_dodge(width = 0.75)) +
  labs(x = "", y = "Budget $") +
  ggtitle("Budget Allocation") +
  theme(plot.title = element_text(hjust = 0.5))

g_budget_allocation
```



transition matrix heatmap

```r
############## visualizations ##############
# transition matrix heatmap for "real" data
df_plot_trans <- mod_attrib_complete$transition_matrix

cols <- c("#e7f0fa", "#c9e2f6", "#95cbee", "#0099dc", "#4ab04a", "#ffd73e", "#eec73a",
          "#e29421", "#e29421", "#f05336", "#ce472e")
t <- max(df_plot_trans$transition_probability)

ggplot(df_plot_trans, aes(y = channel_from, x = channel_to, fill = transition_probability)) +
  theme_minimal() +
  geom_tile(colour = "white", width = .9, height = .9) +
  scale_fill_gradientn(colours = cols, limits = c(0, t),
                       breaks = seq(0, t, by = t/4),
                       labels = c("0", round(t/4*1, 2), round(t/4*2, 2), round(t/4*3, 2), round(t/4*4, 
                       guide = guide_colourbar(ticks = T, nbin = 50, barheight = .5, label = T, barwidt
  geom_text(aes(label = round(transition_probability, 2)), fontface = "bold", size = 4) +
  theme(legend.position = 'bottom',
        legend.direction = "horizontal",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        plot.title = element_text(size = 20, face = "bold", vjust = 2, color = 'black', lineheight = 0.
        axis.title.x = element_text(size = 24, face = "bold"),
        axis.title.y = element_text(size = 24, face = "bold"),
        axis.text.y = element_text(size = 8, face = "bold", color = 'black'),
        axis.text.x = element_text(size = 8, angle = 90, hjust = 0.5, vjust = 0.5, face = "plain")) +
  ggtitle("Transition matrix heatmap")
```



Calculate ROAS and CPA - for model with order = Best

```
calculation = mod_attrib_complete_best$result

calculation <- data.frame(total_cost = c(15000, 21000, 22000, 10000, 20000, 5000), calculation)

calculation$chanel_weight <- calculation$total_conversions / sum(calculation$total_conversions)

calculation$cost_weight <- calculation$total_cost / sum(calculation$total_cost)

calculation$roas <- calculation$chanel_weight / calculation$cost_weight

calculation$optimal_budget = calculation$total_cost * calculation$roas

calculation$CPA = calculation$total_cost / calculation$total_conversions



calculation$CPA
```

```
## [1] 122.65967  69.96096  95.78558  37.59322 103.37150  34.63059
```

```
calculation$optimal_budget
```

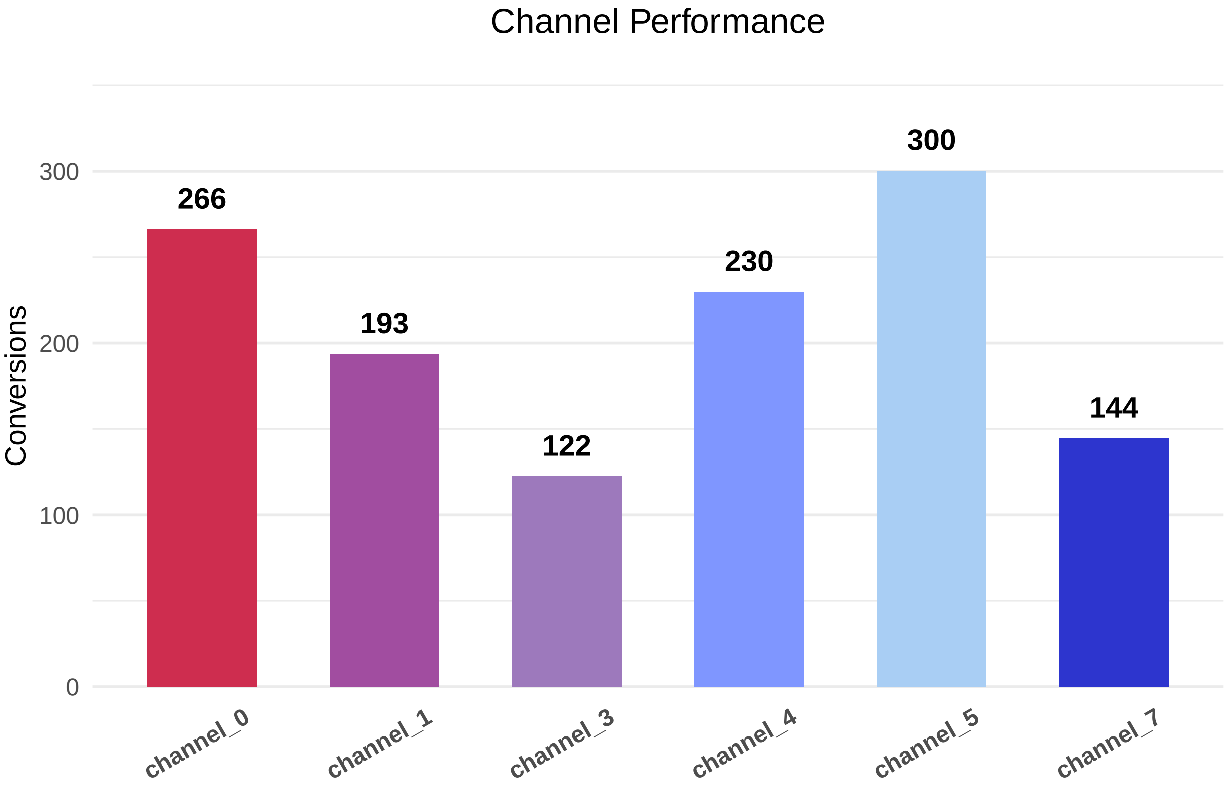
```
## [1]  9054.881 22225.770 17006.536 19696.261 14325.918 10690.632
```

```
calculation$total_cost
```

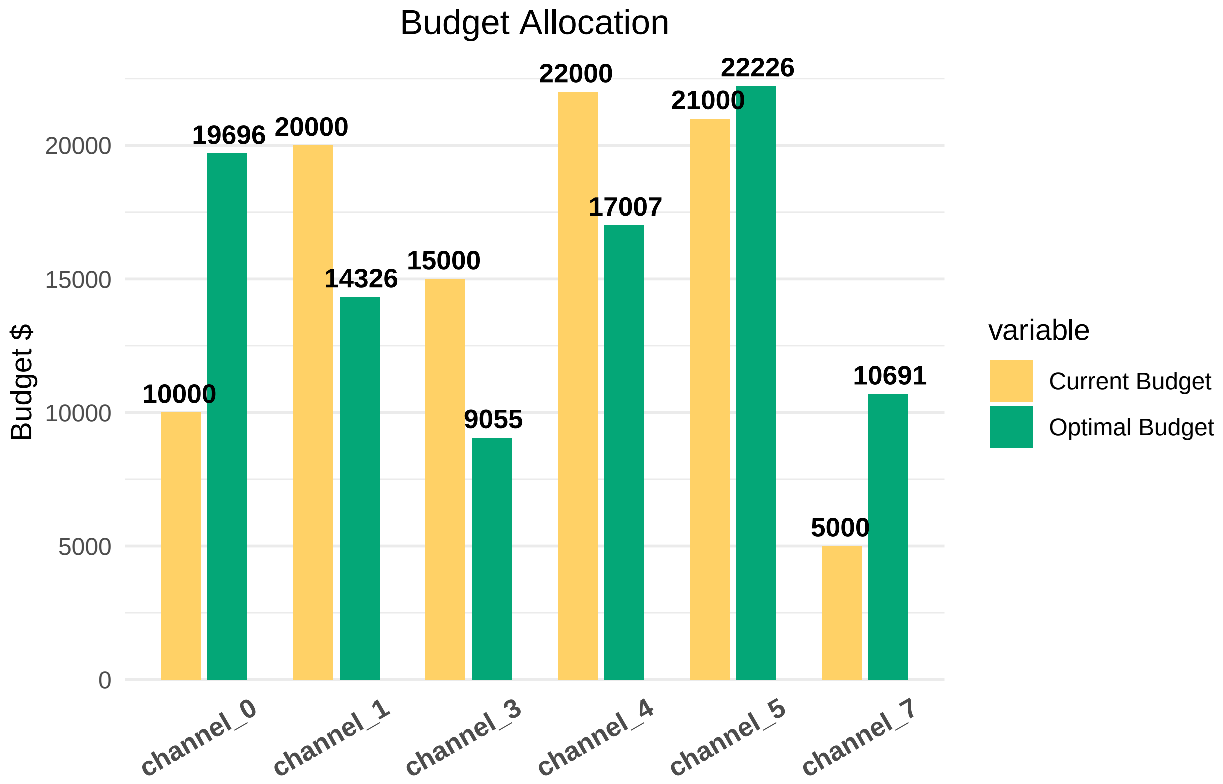
```
## [1] 15000 21000 22000 10000 20000  5000
```

Create an ordered graph showing conversions attributed to each channel

```
# Create an ordered graph showing conversions attributed to each channel
g_channel_performance <- ggplot(calculation, aes(x = channel_name, y = total_conversions, fill = channel
  geom_bar(stat = "identity", width = 0.6) +
  ylim(0, 350) +
  scale_fill_manual(values = c("#CE2D4F",
                               "#A14DA0",
                               "#9D79BC",
                               "#7F96FF",
                               "#A9CEF4",
                               "#2d35ce")) +
  theme_minimal() +
  theme(axis.text.x = element_text(size = 9, angle = 30, hjust = 0.6, face = "bold")) +
  theme(panel.grid.major.x = element_blank()) +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_text(aes(label = round(total_conversions, 0)), fontface = "bold", size = 4, vjust = -1) +
  labs(x = "", y = "Conversions") +
  ggtitle("Channel Performance") +
  guides(fill=FALSE)

g_channel_performance
```



```
df_g2 = calculation[, c("channel_name", "total_cost", "optimal_budget")]
df_g2 = melt(df_g2, id = "channel_name")

g_budget_allocation <- ggplot(df_g2, aes(x = channel_name, y = value, fill = variable)) +
  geom_bar(stat = "identity", width = 0.6, position = position_dodge(width = 0.7)) +
  scale_fill_manual(labels = c("Current Budget", "Optimal Budget"), values = c("#FFD166", "#04A777")) +
  theme_minimal() +
  theme(axis.text.x = element_text(size = 10, angle = 30, hjust = 0.6, face = "bold")) +
  theme(panel.grid.major.x = element_blank()) +
  geom_text(aes(label = round(value, 0)),
            fontface = "bold", size = 3.5,
            vjust = -0.5, position = position_dodge(width = 0.75)) +
  labs(x = "", y = "Budget $") +
  ggtitle("Budget Allocation") +
  theme(plot.title = element_text(hjust = 0.5))

g_budget_allocation
```


Budget Allocation
Budget $
channel_0
channel_1
channel_3
channel_4
channel_5
channel_7
variable
🟨 Current Budget
🟩 Optimal Budget