Point Process Simulation

Estimated Parameters

```
set.seed(1899)
ad_types <- c("search", "display", "other", "purchase")
K <- length(ad_types)
alpha_mean <- matrix(c(2.817, 0.0860, 0.5381, 0.6167,
                       0.1614, 1.7818, 0.2055, 0.0845,
                       0.4647, 0.1270, 8.0526, 0.8384),
                     nrow = 3, ncol = 4, byrow = T)
row.names(alpha_mean) <- ad_types[1:3]
colnames(alpha_mean) <- ad_types

alpha_sd <- matrix(c(0.1765, 0.0214, 0.0562, 0.0633,
                       0.0496, 0.2314, 0.0572, 0.0347,
                       0.0654, 0.0367, 0.4117, 0.0867),
                   nrow = 3, ncol = 4, byrow = T)

beta_mean <- c(34.0188, 46.8854, 51.5114)
beta_sd <- c(1.7426, 4.9370, 2.3241)
names(beta_mean) <- ad_types[1:3]

theta_mu_mean <- c(-5.3926, -6.1027, -5.8063, -9.7704)
theta_mu_sd <- c(0.0166, 0.0212, 0.0221, 0.0762)
names(theta_mu_mean) <- ad_types
names(theta_mu_sd) <- ad_types

Sigma_mu_mean <- matrix(c(0.4584, -0.1197, -0.4942, 0.2256,
                       -0.1197, 0.5934, -0.3380, -0.4157,
                       -0.4942, -0.3380, 1.0014, 0.0762,
                       0.2256, -0.4157, 0.0762, 2.3914), nrow = 4, ncol = 4)

Sigma_mu_sd <- matrix(c(0.0246, -0.0212, 0.0257, 0.1228,
                       0.0212, 0.0335, 0.0304, 0.1665,
                       0.0257, 0.0304, 0.0365, 0.2440,
                       0.1228, 0.1665, 0.2440, 0.2575), nrow = 4, ncol = 4)

row.names(Sigma_mu_mean) <- ad_types
colnames(Sigma_mu_mean) <- ad_types

n <- 12000
#inv_df <- n - 1
inv_df <- 7
psi_mean <- c(-0.5664, -0.7556, -0.6235, 0.2787)
psi_sd <- c(0.1228, 0.2348, 0.1229, 0.2160)
names(psi_mean) <- ad_types
names(psi_sd) <- ad_types


first_click_prob <- c(1199, 418, 811) %>% prop.table()
```

Table 1: alpha

	search	display	other	purchase
search	2.8170	0.0860	0.5381	0.6167
display	0.1614	1.7818	0.2055	0.0845
other	0.4647	0.1270	8.0526	0.8384

Table 2: beta

	x
search	34.0188
display	46.8854
other	51.5114

Table 3: theta

	x
search	-5.3926
display	-6.1027
other	-5.8063
purchase	-9.7704

Table 4: Sigma

	search	display	other	purchase
search	0.4584	-0.1197	-0.4942	0.2256
display	-0.1197	0.5934	-0.3380	-0.4157
other	-0.4942	-0.3380	1.0014	0.0762
purchase	0.2256	-0.4157	0.0762	2.3914

Table 5: psi

	x
search	-0.5664
display	-0.7556
other	-0.6235
purchase	0.2787

Simulate Data

Draw Parameters

1. Draw $\alpha, \beta, \psi, \theta_\mu, \Sigma_\mu$ and generate $\mu^i \sim MVN_K(\theta_\mu, \Sigma_\mu)$ to simulate the behaviour of a representative customer. Draw $\mu^i \sim log\text{-}MVN_K(\theta_\mu, \Sigma_\mu)$.

```r
# calculate gamma and inv wishart params from means and SDs
gammaShRaFromMeanSD <- function( mean , sd ) {
  if ( mean <=0 ) stop("mean must be > 0")
  if ( sd <=0 ) stop("sd must be > 0")
  shape = mean^2/sd^2
  rate = mean/sd^2
  return( list( shape=shape , rate=rate ) )
}

invParamsWishartFromMeanPAndDF <- function(Sigma_mu_mean, df, dim){
  scale_matrix <- Sigma_mu_mean*df - Sigma_mu_mean*dim - Sigma_mu_mean
  return(list(scale_matrix=scale_matrix, df=df, dim=dim))
}

draw_params <- function() {
  alpha <- matrix(nrow = nrow(alpha_mean), ncol = ncol(alpha_mean))
  for(j in c(1:nrow(alpha_mean))) {
    for(k in c(1:ncol(alpha_mean))) {
      gam <- gammaShRaFromMeanSD(mean = alpha_mean[j,k], sd = alpha_sd[j,k])
      alpha[j,k] <- rgamma(n=1, shape = gam$shape, rate = gam$rate)
    }
  }

  beta <- numeric()
  for(j in c(1:length(beta_mean))) {
      gam <- gammaShRaFromMeanSD(mean = beta_mean[j], sd = beta_sd[j])
      beta[j] <- rgamma(n=1, shape = gam$shape, rate = gam$rate)
  }

  # assumption that all psi_ks are independet we can sample with the diagonal of sds,
  # in appendix they calculate psi with the Identity matrix which assumes independence!
  psi <- mvrnorm(mu = psi_mean,
                 Sigma = psi_sd^2 * diag(nrow = length(psi_sd)))
  theta_mu <- mvrnorm(mu = theta_mu_mean,
                      Sigma = theta_mu_sd^2 * diag(nrow = length(theta_mu_sd)))

  inv_wishart_params <- invParamsWishartFromMeanPAndDF(Sigma_mu_mean = Sigma_mu_mean,
                                                       df = inv_df, dim = length(ad_types))
  Sigma_mu <- riwish(v = inv_wishart_params$df, S = inv_wishart_params$scale_matrix)
  mu <- exp(mvrnorm(mu = theta_mu, Sigma = Sigma_mu))
  list(alpha = alpha, beta = beta, psi = psi,
       theta_mu = theta_mu, Sigma_mu = Sigma_mu, mu = mu)
}
```

2. Simulate point process in $[0, T]$ given $\alpha, \beta, \psi, \mu^i$ and realized type j_0 at $t_0 = 0$,($j_0 = 1, \ldots, K-1$).
   a. initialize $t = 0, n = 0, n_K = 0, m = \sum_{k=1}^{K}(\mu_k^i + \alpha_{j_o k})$
   b. Repeat until $t > T$
      i) Simulate $s \sim Exp(m)$
      ii) Set $t = t + s$
      iii) If $t < T$, calculate

$$\lambda_k = \mu_k^i \exp(\psi_k n_K) + \alpha_{j_0 k} \exp(-\beta_{j_0} t) + \sum_{l=1, j_l \neq K}^{n} \alpha_{j_l k} \exp(-\beta_{j_l}(t - t_l))$$

and let $\lambda = \sum_{k=1}^{K} \lambda_k$ and generate $U \sim Unif(0,1)$.

1. If $U \leq \frac{\lambda}{m}$, $n = n+1$, $t_n = t$. Simulate $j_n \sim multinomial(1, \lambda_1/\lambda, \ldots, \lambda_K/\lambda)$.
   - If $j_n = K$ then
   $$n_K = n_K + 1$$
   $$m = \lambda - \mu_k^i \exp(\psi_k(n_K - 1)) + \mu_k^i \exp(\psi_k n_K)$$
   - else
   $$m = \lambda + \alpha_{j_n k}$$
2. If $U > \frac{\lambda}{m}$ then
   $$m = \lambda$$

c. Simulation output is $\{t_1, \ldots, t_n\}$ and $\{j_1, \ldots, j_n\}$

3. Repeat Step 1 and 2 R times.

```
T <- 120
```

```
simulate_user_clickstream <- function(alpha, beta, psi, theta_mu, Sigma_mu, mu) {
  t = 0; n = 0; n_K = 0; m = c(alpha[j_0,] + mu) %>% sum; j = integer(); t_j = numeric()

  hist_intensity_decay <- function(a, b, events, current_time, event_times) {
    result <- numeric(K)
    names(result) <- ad_types
    if(length(events) == 0) return(result)

    for(k in 1:K){
      for(idx in 1:length(events)){
        j_l <- events[idx]
        if(j_l != K) { # do not consider purchases (K) here
          result[k] <- result[k] +
            alpha[j_l, k] * exp(-1 * beta[j_l] * (t - event_times[idx]))
        }
      }
    }
    return(result)
  }

  repeat {
    s <- rexp(n = 1, rate = m)
    t <- t + s
    if(!is.nan(t) && t < T){
      lambda <- mu * exp(psi*n_K) + alpha[j_0, ] * exp(-1 * beta[j_0] * t)
      lambda <- lambda + hist_intensity_decay(alpha, beta, j, t, t_j)
      U <- runif(n = 1)
      if(is.na(m) || is.na(U)  || NA %in% lambda || is.infinite(m) || Inf %in% lambda) {
        print(paste("Value is NA ", c(m, U, lambda)))
      }
      if(U <= sum(lambda) / m){
        n <- n + 1
        t_j[n] <- t
        j[n] <- which(rmultinom(n = 1, size = 1,
                                prob = lambda/sum(lambda))[,1] %in% c(1))
        if(j[n] == K) { # if purchase
          n_K <- n_K + 1
```

```
          m <- sum(lambda - mu * exp(psi * (n_K -1)) + mu * exp(psi * n_K))
        } else {
          m <- sum(lambda + alpha[j[n], ])
        }
      } else {
        m <- sum(lambda)
      }
    }
    # in some cases n_K becomes greater then a few thousand.
    # hence, m tends so go to Inf as s becomes 0.
    # this breaks the code and so we break at a certain purchase count.
    if(t > T || n_K >= 10) return(list(timestamp = t_j, event = ad_types[j]))
  }
}
```

Simulate click stream data

Simulate for one user

```
j_0 <- sample(length(ad_types) - 1, size = 1, prob = first_click_prob)
params <- draw_params()
data <- do.call(simulate_user_clickstream, params)
```

j_0 is set to type "other" with $T = 120$.

	time	event
1	93.27	other
2	93.28	other
3	105.77	other
4	105.79	other

Table 6: Simulated click stream

Simulate for multiple users and plot their click streams.

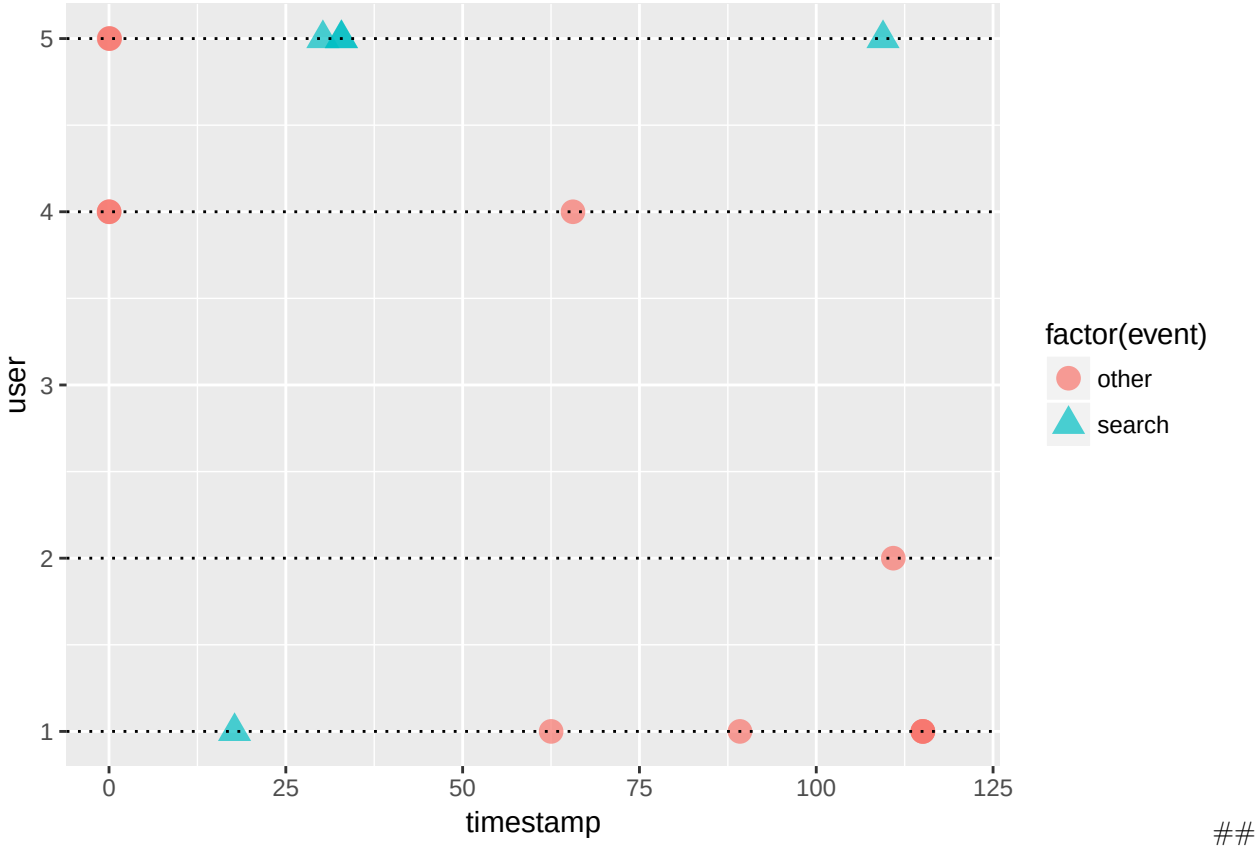
```
sim_cnt <- 5
```

Simulate for 5 users.

```
sim_data <- function(count) {
  users <- data.frame()
  for(i in 1:count) {
    j_0 <- sample(length(ad_types) - 1, size = 1, prob = first_click_prob)
    params <- draw_params()
    data <- do.call(simulate_user_clickstream, params)
    users <- users %>%
      rbind(data_frame(user = rep.int(i, length(data$event)),
                       event = data$event,
                       timestamp=data$timestamp))
  }
```

```
  users
}
```

```
users <- sim_data(sim_cnt)
ggplot(users, aes(x = timestamp, y = user,
                  label = event, color = factor(event),
                  shape = factor(event),
                  xmax = T)) +
  geom_point(size = 4, alpha = .7) +
  geom_hline(aes(yintercept = user), lty="dotted") +
  xlim(0, T) +
  ylim(1, sim_cnt)
```



##

Simulate 12,000 users

```
data <- sim_data(n)
```

```
data %>%
  filter(event != 'purchase') %>%
  group_by(event) %>%
  summarise(n = n()) %>%
  mutate(ratio = n / sum(n)) %>%
  knitr::kable(caption = "Ad click ratio")
```

Table 7: Ad click ratio

event	n	ratio
display	6600	0.2354032
other	12408	0.4425580

event	n	ratio
search	9029	0.3220387

```
data %>%
  mutate(event_short = recode(event, 'purchase' = 'P', 'other' = 'O',
                              'search' = 'S', 'display' = 'D')) %>%
  group_by(user) %>%
  summarise(stream = paste(event_short, collapse = "")) %>%
  mutate(stream = stream %>% gsub(x = ., "(P).*","\\1")) %>%
  filter(stream %>% nchar() > 2, stream %>% endsWith('P')) %>%
  group_by(stream) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  head(n=20) %>%
  knitr::kable(caption = "Most frequent click combinations ending with a purchase")
```

Table 8: Most frequent click combinations ending with a purchase

stream	n
OOP	44
SSP	34
OSP	22
OOOP	20
SSSP	17
SOP	15
DSP	9
OOOOP	9
OOSP	8
DOP	5
OOOOOOP	4
SDP	4
SSSSP	4
DDP	3
DSSP	3
ODSP	3
OSSP	3
SOOP	3
DOOOP	2
DOOP	2