R Notebook

Blake Cuningham CNNBLA001

Contents


```
rm(list = ls())

suppressWarnings(suppressMessages(library(data.table)))
suppressWarnings(suppressMessages(library(tidyverse)))
suppressWarnings(suppressMessages(library(lubridate)))
suppressWarnings(suppressMessages(library(stringr)))
```

Get data into R

```
orders_orig <- data.frame(fread("data/orders.csv", header = T))
order_items <- data.frame(fread("data/order-items.csv", header = T))
```

Create separate data frames for orders, customers, items

```
customers <- orders_orig %>%
  select(customers_id, customers_gender, customers_dob) %>%
  unique() %>%
  mutate(customers_id = as.numeric(customers_id)) %>%
  arrange(customers_id)

orders <- orders_orig %>%
  select(-c(customers_gender, customers_dob)) %>%
  unique() %>%
  mutate(customers_id = as.numeric(customers_id)) %>%
  arrange(customers_id)
```

Clean the order items table

```
order_items <- order_items %>%
  mutate(products_price = as.numeric(products_price)) %>%
  mutate(products_quantity = as.numeric(products_quantity)) %>%
  filter(products_price > 0) %>%
```

```
  mutate(products_quantity = ifelse(str_detect(products_name, "case|CASE|Case") & !str_detect(products_
  mutate(products_quantity = ifelse(str_detect(products_name, "case|CASE|Case") & str_detect(products_na
  mutate(products_quantity = ifelse(str_detect(products_name, "12 Assorted"), products_quantity * 12, pr
  mutate(products_quantity = ifelse(str_detect(products_name, "Assorted"), products_quantity * 12, produ
  mutate(products_quantity = ifelse(str_detect(products_name, "Budget White"), products_quantity * 12, p
  mutate(products_quantity = ifelse(str_detect(products_name, "30 bottles"), products_quantity * 30, pro
  mutate(products_quantity = ifelse(str_detect(products_name, "Voucher"), products_quantity * 6, product


# TODO:

# mixed cases contain 12 bottles (6 red, 6 white)
# seems all order items with "case" in the name, and not a multiple of 6 (or 12?) are actually cases an
# assuming assorted is 12 too
# wynklub seems to be just one bottle
# gift vouchers are tricky ... assume the intention is to buy 6 bottles
# remove "as arranged by Johan"?
# remove wine years?

# table(order_items$products_quantity)
# order_items[order_items$products_quantity == 1 & !str_detect(order_items$products_name, "wynklub"),]
# order_items[str_detect(order_items$products_name, "Johan"),]
```

Add features to customers table

```
# number of orders

customers <- orders %>%
  count(customers_id) %>%
  right_join(customers)
```

```
## Joining, by = "customers_id"
```

```
colnames(customers)[colnames(customers) == "n"] <- "order_count"

# age and age group

customers <- customers %>%
  mutate(customers_dob = parse_date_time(customers_dob, "%Y-%m-%d H:%M:%S")) %>%
  mutate(customer_age =  as.period(interval(customers_dob, ymd("20170824")))) %>%
  mutate(customer_age_group = ifelse(customer_age >= years(60), "old", ifelse(customer_age >= years(35)
  mutate(customer_age_group = as.factor(customer_age_group))

# avg. per bottle
customer_item_prices <- left_join(orders, order_items) %>%
  mutate(total_paid = products_price * products_quantity) %>%
  group_by(customers_id) %>%
  summarise("order_amount" = sum(total_paid), "item_count" = sum(products_quantity)) %>%
  mutate(avg_item = order_amount / item_count)
```

```
## Joining, by = "orders_id"
```

```
customers <- customer_item_prices %>%
  select(customers_id, avg_item) %>%
  right_join(customers)
```

```
## Joining, by = "customers_id"
```

```
customers <- customers %>%
  mutate(customer_price_group = ifelse(avg_item >= 500, "luxury",
                                       ifelse(avg_item >= 150, "premium",
                                              ifelse(avg_item >= 50, "economy",
                                                     "budget")))) %>%
  mutate(customer_price_group = as.factor(customer_price_group))


# Add in country and payment method
customers <- customers %>%
  left_join(orders) %>%
  select(-c(date_purchased, orders_id, order_total)) %>%
  distinct(customers_id, .keep_all = T)
```

```
## Joining, by = "customers_id"
```

```
# Order count class
customers <- customers %>%
  mutate(order_freq = ifelse(order_count >= 3, "frequent",
                             ifelse(order_count >= 2, "twice",
                                    "once"))) %>%
  mutate(order_freq = as.factor(order_freq))
```

Identify top and bottom customers by number of unique products bought

```
# justify choosing "41777" for a top customers
order_items %>%
  right_join(orders, by = "orders_id") %>%
  count(customers_id) %>%
  arrange(desc(n)) %>%
  head(5) %>%
  knitr::kable(caption = "5 customers with highest variety of products bought")
```

Table 1: 5 customers with highest variety of products bought

customers_id	n
1022	16
36892	16
29302	13
39263	13
41777	12

```
# justify choosings "411" for a bottom customer
order_items %>%
  right_join(orders, by = "orders_id") %>%
  count(customers_id) %>%
  arrange(n) %>%
  head() %>%
```

```
  knitr::kable(caption = "5 customers with lowest variety of products bought")
```

Table 2: 5 customers with lowest variety of products bought

customers_id	n
65	1
163	1
411	1
506	1
624	1
889	1

```
temp <- customers %>%
  filter(customers_id == "41777" |
           customers_id == "411") %>%
  select(-customer_age) %>% # doesn't seem possible to transpose if containing lubridate type
  # data.frame() %>%
  t() %>%
  data.frame()

colnames(temp) <- c("Low product variety", "High product variety")
knitr::kable(temp, caption = "Comparison of two example customers")
```

Table 3: Comparison of two example customers

	Low product variety	High product variety
customers_id	411	41777
avg_item	720.0000	137.9333
order_count	1	3
customers_gender	f	
customers_dob	1944-05-29	1950-12-25
customer_age_group	old	old
customer_price_group	luxury	economy
countries_name	Cape Town	Cape Town
payment_method	Bank Deposit/EFT	Credit Card Payment
order_freq	once	frequent

```
  # knitr::kable()
```

Simple bought products approach (i.e. user-based collaborative filtering)

```
# get matrix of "bought wines" i.e. customer_id as rows and wines as columns

customer_products_bought_tall <- order_items %>%
  select(orders_id, products_name) %>%
  inner_join(orders %>% select(customers_id, orders_id)) %>% # use inner join to avoid customers withou
  select(customers_id, products_name) %>%
  unique() %>%
  mutate(bought = 1)
```

```
## Joining, by = "orders_id"
```

```
customer_products_bought_wide <- customer_products_bought_tall %>%
  complete(customers_id, products_name, fill = list(bought = 0)) %>%
  spread(key = products_name, value = bought)

# convert data to matrix form
sorted_customers_id <- as.character(unlist(customer_products_bought_wide[,1]))
customer_products_bought_wide <- as.matrix(customer_products_bought_wide[,-1])
row.names(customer_products_bought_wide) <- sorted_customers_id
```

```
cosine_sim <- function(a,b){crossprod(a,b)/sqrt(crossprod(a)*crossprod(b))}
temp_func <- Vectorize(function(x, y) cosine_sim(customer_products_bought_wide[x,], customer_products_b

customer_similarities1 <- outer(seq_len(nrow(customer_products_bought_wide)),
      seq_len(nrow(customer_products_bought_wide)),
      temp_func)

diag(customer_similarities1) <- 0
row.names(customer_similarities1) <- row.names(customer_products_bought_wide)
colnames(customer_similarities1) <- row.names(customer_products_bought_wide)

saveRDS(customer_similarities1, "output/customer_similarities1.rds")

# max(customer_products_bought_wide[263,])
# cosine_sim(customer_products_bought_wide[3,], customer_products_bought_wide[3,])
```

```
customer_similarities1 <- readRDS("output/customer_similarities1.rds")


# sort(apply(customer_products_bought_wide, 1, sum), decreasing = T)
# # Raka Spliced 2014
# # Le Bonheur Prima 2013 unlabelled
# # Knorhoek Shiraz/Cabernet Franc/Cabernet Sauvignon 2015
#
# crossprod(customer_products_bought_wide[, "Raka Spliced 2014"], customer_similarities1["39263",])
#
# temp <- customer_similarities1["39263",] %*% customer_products_bought_wide
# # customer_similarities1["39263",] %*% customer_products_bought_wide
# dim(customer_similarities1["39263",])
dim(customer_products_bought_wide)
```

```
## [1] 999 323
```

```
# dim(temp)

dim(customer_products_bought_wide[0,])
```

```
## [1]   0 323
```

```
customer_scores <- data.frame(product = colnames(customer_products_bought_wide),
                              score = as.vector(customer_similarities1["39263",] %*% customer_products_
                              bought = customer_products_bought_wide["39263",])

customer_scores %>%
  # filter(bought == 0) %>%
```

```
  arrange(desc(score)) %>%
  select(product, score, bought) %>%
  head()
```

```
##                                       product     score bought
## 1              Le Bonheur Prima 2013 unlabelled 18.811557      1
## 2 Le Bonheur Cabernet Sauvignon 2014 unlabelled  7.501758      0
## 3                      Clos Malverne Shiraz 2014  6.479492      1
## 4                              Raka Spliced 2014  3.309603      0
## 5           Le Bonheur Tricorne 2012 unlabelled  3.303285      0
## 6              Neil Ellis Sincerely Shiraz 2016  2.897555      1
```

```
customer_based_recommendations <- function(customer, customer_similarities, bought_wines){

  # turn into character if not already
  customer <- ifelse(is.character(customer),customer,as.character(customer))

  # get scores
  customer_scores <- data.frame(product = colnames(bought_wines),
                                score = as.vector(customer_similarities[customer,] %*% bought_wines),
                                bought = bought_wines[customer,])

  # sort unseen movies by score and remove the 'seen' column
  customer_recom <- customer_scores %>%
    filter(bought == 0) %>%
    arrange(desc(score)) %>%
    select(-bought)

  return(customer_recom)

}

head(customer_based_recommendations("39263", customer_similarities1, customer_products_bought_wide))
```

```
##                                             product    score
## 1          Le Bonheur Cabernet Sauvignon 2014 unlabelled 7.501758
## 2                                   Raka Spliced 2014 3.309603
## 3                    Le Bonheur Tricorne 2012 unlabelled 3.303285
## 4                      Nitida Roxia Sauvignon Blanc 2016 2.844950
## 5 Knorhoek Shiraz/Cabernet Franc/Cabernet Sauvignon 2015 2.817521
## 6                  Asara Vineyard Collection Shiraz 2012 2.695650
```

Item-based approach

```
products_customers_wide <- t(customer_products_bought_wide)
```

```
temp_func <- Vectorize(function(x, y) cosine_sim(products_customers_wide[x,], products_customers_wide[y

product_similarities1 <- outer(seq_len(nrow(products_customers_wide)),
      seq_len(nrow(products_customers_wide)),
      temp_func)

diag(product_similarities1) <- 0
```

```
row.names(product_similarities1) <- row.names(products_customers_wide)
colnames(product_similarities1) <- row.names(products_customers_wide)

saveRDS(product_similarities1, "output/product_similarities1.rds")
```

```
product_similarities1 <- readRDS("output/product_similarities1.rds")
```

```
user_bought <- customer_products_bought_tall %>%
        filter(customers_id == "39263") %>%
        select(products_name) %>%
        unlist() %>%
        as.character()

product_scores <- tibble(product = row.names(product_similarities1),
                      score = apply(product_similarities1[,user_bought],1,sum),
                      bought = products_customers_wide[,as.character(39263)])

product_scores %>%
  filter(bought == 0) %>%
  arrange(desc(score)) %>%
  head()
```

```
## # A tibble: 6 x 3
##                                         product     score bought
##                                           <chr>     <dbl>  <dbl>
## 1             Vrede en Lust White Mischief 2016 1.0773503      0
## 2              Jordan The Prospector Syrah 2015 0.8912014      0
## 3                 Vrede en Lust Cotes De Savoye 0.8625711      0
## 4 Le Bonheur Cabernet Sauvignon 2014 unlabelled 0.8198409      0
## 5                       Iona One Man Band 2011 0.7618017      0
## 6                   Remhoogte Estate Cape Blend 0.7618017      0
```

```
product_based_recommendations <- function(customer, product_similarities, bought_products){

  # turn into character if not already
  customer <- ifelse(is.character(customer),customer,as.character(customer))

  # get scores
  customer_bought <- row.names(product_similarities)[bought_products[,customer] == TRUE]
  customer_scores <- tibble(product = row.names(product_similarities),
                        score = apply(data.frame(product_similarities[,customer_bought]),1,sum),
                        bought = bought_products[,customer])
  # sort unseen movies by score and remove the 'seen' column
  customer_recom <- customer_scores %>% filter(bought == 0) %>% arrange(desc(score)) %>% select(-bought)

  return(customer_recom)

}

head(product_based_recommendations("39263", product_similarities1, products_customers_wide))
```

```
## # A tibble: 6 x 2
##                                         product     score
##                                           <chr>     <dbl>
## 1             Vrede en Lust White Mischief 2016 1.0773503
```

```
## 2              Jordan The Prospector Syrah 2015 0.8912014
## 3                 Vrede en Lust Cotes De Savoye 0.8625711
## 4 Le Bonheur Cabernet Sauvignon 2014 unlabelled 0.8198409
## 5                        Iona One Man Band 2011 0.7618017
## 6                   Remhoogte Estate Cape Blend 0.7618017
```

```
# undebug(product_based_recommendations)
# test <- "411"
# test <- row.names(product_similarities1)[products_customers_wide[,test] == TRUE]
# tibble(product = row.names(product_similarities1),
#                         score = apply(data.frame(product_similarities1[,test]),1,sum),
#                         bought = products_customers_wide[,"411"]) %>%
#   arrange(desc(score))


# head(product_based_recommendations("39263", product_similarities1, products_customers_wide)) %in% hea
```

Matrix factorization

Proxy for wine rating? Number of bottles?

Some combination of quantity and price . . . i.e. something may be really good, but expensive. Something may be mediocre but cheap. Number of bottles bias towards cheap, price per bottle bias toward expensive. Some combination of the two? And probably a log of that combination in order to reduce impact of MASSIVE orders.

```
product_ratings <- order_items %>%
  mutate(log_order_val = log(products_quantity * products_price)) %>% # create ratings proxy
  inner_join(orders) %>%
  select(customers_id, products_name, log_order_val) %>%
  group_by(customers_id, products_name) %>%
  summarise(rating_proxy = mean(log_order_val)) %>%
  ungroup() %>%
  # complete(customers_id, products_name, fill = list(ratings_proxy = NA))
  complete(customers_id, products_name) %>%
  spread(key = products_name, value = rating_proxy)
```

```
## Joining, by = "orders_id"
```

```
# product_ratings %>%
#   gather(-customers_id, key = products_name, value = rating_proxy)

products_list <- as.character(unlist(product_ratings[,1]))
product_ratings <- as.matrix(product_ratings[,-1])
row.names(product_ratings) <- products_list
```

```
ranks <- c(1, 2, 3, 4, 5, 10, 20, 50, 100, 200)
# ranks <- c(1, 2, 3)
mf_acc <- NULL
mf_test_acc <- NULL

set.seed(1)
pr_sample_ind <- sample(length(product_ratings), ceiling(0.2*length(product_ratings)))
pr_train <- cbind(product_ratings)
pr_train[pr_sample_ind] <- NA
```

```r
pr_test <- cbind(product_ratings)
pr_test[-pr_sample_ind] <- NA
```

```r
for (rank in ranks){
  set.seed(1)

  decomp <- nnmf(A = pr_train,
                 alpha = 3e-3,
                 beta = 3e-3,
                 method = "scd",
                 k = rank,
                 max.iter = 10000
                 )
  mf_observed <- decomp$W %*% decomp$H

  # Train accuracy
  mf_train_errors <- (mf_observed - pr_train)^2
  mf_train_accuracy <- sqrt(mean(mf_train_errors[!is.na(pr_train)]))

  mf_acc <- append(mf_acc, mf_train_accuracy)

  # Test accuracy
  mf_test_errors <- (mf_observed - pr_test)^2
  mf_test_accuracy <- sqrt(mean(mf_test_errors[!is.na(pr_test)]))

  mf_test_acc <- append(mf_test_acc, mf_test_accuracy)

}

rank_test_results <- data.frame("Rank" = ranks, "Train_acc" = mf_acc, "Test_acc" = mf_test_acc)

saveRDS(rank_test_results, "output/rank_test_results.rds")
```

```r
check_matrix <- matrix(rep(mean(pr_train, na.rm = T), length(pr_test)), dim(pr_test)[1], dim(pr_test)[2]
check_errors <- (check_matrix - pr_test)^2
check_accuracy <- sqrt(mean(check_errors[!is.na(pr_test)]))
check_accuracy
```

```
## [1] 1.036206
```

```r
rank_test_results <- readRDS("output/rank_test_results.rds")

ggplot(rank_test_results) +
  geom_line(aes(x = Rank, y = Train_acc, color = "Train")) +
  geom_line(aes(x = Rank, y = Test_acc, color = "Test")) +
  geom_vline(aes(xintercept = 1), linetype = 2) +
  # scale_color_manual(values = c("Train", "Test")) +
  scale_color_discrete("Accuracy") +
  theme_light()
```

```r
sum(is.na(product_ratings)) / length(product_ratings)
```

```
## [1] 0.9910902
```

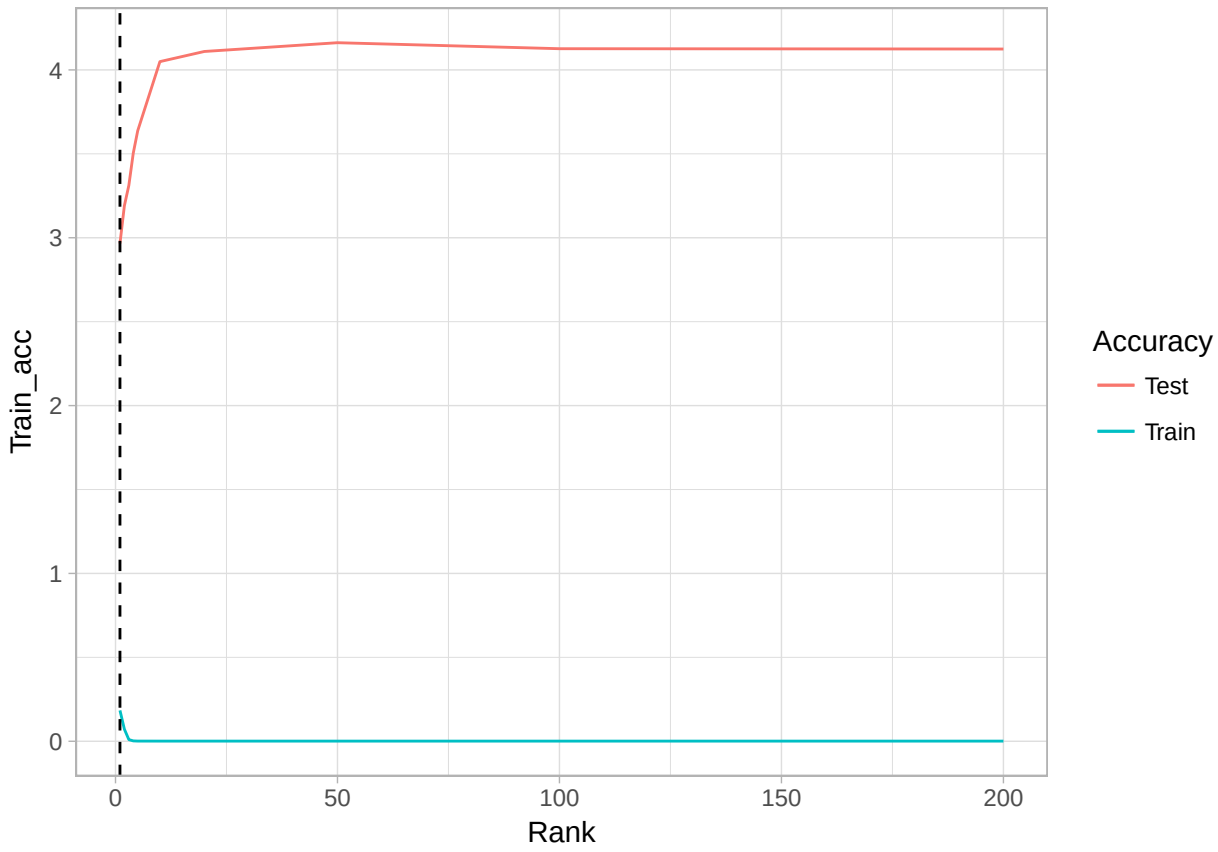

Figure 1: Mean squared error of matrix ranks

```r
library(NNLM)

set.seed(7)

mf1_decomp <- nnmf(A = product_ratings,
                   alpha = 3e-3,
                   beta = 3e-3,
                   method = "scd",
                   k = 1,
                   max.iter = 10000
                   )

saveRDS(mf1_decomp, "models/mf1_decomp.rds")
```

```r
mf1_decomp <- readRDS("models/mf1_decomp.rds")

# Top choices for our user
mf1_predicted <- mf1_decomp$W %*% mf1_decomp$H
mf1_predicted[0,head(order(mf1_predicted["411",], decreasing = T))]
```

```
##      Everyday Red || 12 Assorted Bin 1 Budget Red || 12 Assorted
##      Luxury Red || Assorted Budget Mixed || Assorted White & Red
##      TOP SELLING May RED Case JORDAN Lifestyle Case 12-pack
```

```r
# Top wines based on H factor
mf1_predicted[0,head(order(mf1_decomp$H, decreasing = T))]
```

```
##      Everyday Red || 12 Assorted Bin 1 Budget Red || 12 Assorted
##      Luxury Red || Assorted Budget Mixed || Assorted White & Red
##      TOP SELLING May RED Case JORDAN Lifestyle Case 12-pack
```

Customer similarities with user data

```r
# get matrix of "bought wines" i.e. customer_id as rows and wines as columns

customer_products_bought_tall_2 <- order_items %>%
  select(orders_id, products_name) %>%
  inner_join(orders %>% select(customers_id, orders_id)) %>% # use inner join to avoid customers withou
  select(customers_id, products_name) %>%
  unique() %>%
  mutate(bought = 1)
```

```
## Joining, by = "orders_id"
```

```r
# create customer attributes data to adD
## customer age group
customer_products_bought_tall_2 <- customers %>%
  select(customers_id, customer_age_group) %>%
  mutate(bought = 1) %>%
  complete(customers_id, customer_age_group, fill = list(bought = 0)) %>%
  rename(products_name = customer_age_group) %>%
  rbind(customer_products_bought_tall_2)

## customer gender
```

```r
customer_products_bought_tall_2 <- customers %>%
  select(customers_id, customers_gender) %>%
  mutate(bought = 1) %>%
  complete(customers_id, customers_gender, fill = list(bought = 0)) %>%
  rename(products_name = customers_gender) %>%
  rbind(customer_products_bought_tall_2)

## customer price group
customer_products_bought_tall_2 <- customers %>%
  select(customers_id, customer_price_group) %>%
  mutate(bought = 1) %>%
  complete(customers_id, customer_price_group, fill = list(bought = 0)) %>%
  rename(products_name = customer_price_group) %>%
  drop_na() %>%
  rbind(customer_products_bought_tall_2)


## customer countries
customer_products_bought_tall_2 <- customers %>%
  select(customers_id, countries_name) %>%
  mutate(bought = 1) %>%
  complete(customers_id, countries_name, fill = list(bought = 0)) %>%
  rename(products_name = countries_name) %>%
  drop_na() %>%
  filter(products_name != "") %>%
  rbind(customer_products_bought_tall_2)



## customer payment method
customer_products_bought_tall_2 <- customers %>%
  select(customers_id, payment_method) %>%
  mutate(bought = 1) %>%
  complete(customers_id, payment_method, fill = list(bought = 0)) %>%
  rename(products_name = payment_method) %>%
  drop_na() %>%
  filter(products_name != "") %>%
  rbind(customer_products_bought_tall_2)


## order frequency
customer_products_bought_tall_2 <- customers %>%
  select(customers_id, order_freq) %>%
  mutate(bought = 1) %>%
  complete(customers_id, order_freq, fill = list(bought = 0)) %>%
  rename(products_name = order_freq) %>%
  drop_na() %>%
  filter(products_name != "") %>%
  rbind(customer_products_bought_tall_2)


## spread data into matrix shape
customer_products_bought_wide_2 <- customer_products_bought_tall_2 %>%
```

```
  complete(customers_id, products_name, fill = list(bought = 0)) %>%
  spread(key = products_name, value = bought)

# convert data to matrix form
sorted_customers_id <- as.character(unlist(customer_products_bought_wide_2[,1]))
customer_products_bought_wide_2 <- as.matrix(customer_products_bought_wide_2[,-1])
row.names(customer_products_bought_wide_2) <- sorted_customers_id
```

```
cosine_sim <- function(a,b){crossprod(a,b)/sqrt(crossprod(a)*crossprod(b))}
temp_func_2 <- Vectorize(function(x, y) cosine_sim(customer_products_bought_wide_2[x,], customer_produc

customer_similarities2 <- outer(seq_len(nrow(customer_products_bought_wide_2)),
      seq_len(nrow(customer_products_bought_wide_2)),
      temp_func_2)

diag(customer_similarities2) <- 0
row.names(customer_similarities2) <- row.names(customer_products_bought_wide_2)
colnames(customer_similarities2) <- row.names(customer_products_bought_wide_2)

saveRDS(customer_similarities2, "output/customer_similarities2.rds")

# max(customer_products_bought_wide[263,])
# cosine_sim(customer_products_bought_wide[3,], customer_products_bought_wide[3,])
```

```
customer_similarities2 <- readRDS("output/customer_similarities2.rds")

head(customer_based_recommendations("39263",
                                    customer_similarities1,
                                    customer_products_bought_wide))
```

```
##                                              product    score
## 1          Le Bonheur Cabernet Sauvignon 2014 unlabelled 7.501758
## 2                                       Raka Spliced 2014 3.309603
## 3                     Le Bonheur Tricorne 2012 unlabelled 3.303285
## 4                       Nitida Roxia Sauvignon Blanc 2016 2.844950
## 5 Knorhoek Shiraz/Cabernet Franc/Cabernet Sauvignon 2015 2.817521
## 6                   Asara Vineyard Collection Shiraz 2012 2.695650
```

```
cs_2_indexedBy1 <- customer_similarities2[row.names(customer_similarities1),colnames(customer_similariti
head(customer_based_recommendations("39263",
                                    cs_2_indexedBy1,
                                    customer_products_bought_wide))
```

```
##                                              product    score
## 1                                 Personalised Mixed Case 29.61147
## 2 Knorhoek Shiraz/Cabernet Franc/Cabernet Sauvignon 2015 28.80357
## 3                                       Raka Spliced 2014 26.29577
## 4          Le Bonheur Cabernet Sauvignon 2014 unlabelled 25.81791
## 5                       Nitida Roxia Sauvignon Blanc 2016 20.43521
## 6                     Phizante Kraal Sauvignon Blanc 2016 17.85821
```

Combine recommendations

```
ensemble_recommender <- function(customer, context_similarities, no_context_similarities, product_simila

  # Convert to character
  customer <- ifelse(is.character(customer), customer, as.character(customer))

  # Context containing similarity
  rank_1 <- customer_based_recommendations(customer,
                                           context_similarities,
                                           wide_customer_products_matrix) %>%
    mutate(p_rank = rank(-score)) %>%
    select(product, p_rank)

  # User bought similarity only
  rank_2 <- customer_based_recommendations(customer,
                                           no_context_similarities,
                                           wide_customer_products_matrix) %>%
    mutate(p_rank = rank(-score)) %>%
    select(product, p_rank)

  # Item similarity
  rank_3 <- product_based_recommendations(customer,
                                          product_similarities,
                                          t(wide_customer_products_matrix)) %>%
    mutate(p_rank = rank(-score)) %>%
    select(product, p_rank)

  # Matrix factorization
  rank_4 <- data.frame(score = predicted_ratings[customer,]) %>%
    rownames_to_column("product") %>%
    mutate(p_rank = rank(-score)) %>%
    select(product, p_rank)

  # Combination of methods
  customer_recs <- inner_join(inner_join(inner_join(rank_1, rank_2, by = "product"), rank_3, by = "produ
    mutate(overall = p_rank.x + p_rank.y + p_rank.x.x + p_rank.y.y) %>%
    select(product, overall) %>%
    mutate(p_rank = rank(overall)) %>%
    select(product, p_rank) %>%
    arrange(p_rank)

  return(customer_recs)
}

head(suppressWarnings(ensemble_recommender("39263",
                     context_similarities = cs_2_indexedBy1,
                     no_context_similarities = customer_similarities1,
                     product_similarities = product_similarities1,
                     wide_customer_products_matrix = customer_products_bought_wide,
                     predicted_ratings = mf1_predicted))) %>%
  knitr::kable()
```

product	p_rank
Le Bonheur Tricorne 2012 unlabelled	1
Le Bonheur Cabernet Sauvignon 2014 unlabelled	2
Asara Vineyard Collection Shiraz 2012	3
Flagstone Dragon Tree Cape Blend 2014	4
Capaia One Flagship Red 2010	5
Vrede en Lust Lady J Syrah 2013	6

```
# debug(ensemble_recommender)
# undebug(ensemble_recommender)

# test <- "411"
# data.frame(score = mf1_predicted[test,]) %>%
#     rownames_to_column("product") %>%
#     mutate(p_rank = rank(-score)) %>%
#     select(product, p_rank)

# test <- "411"
# product_based_recommendations(test,
#                                   product_similarities1,
#                                   t(customer_products_bought_wide)) %>%
#     mutate(p_rank = rank(-score)) %>%
#     select(product, p_rank)

# customer_similarities1["411",] %>% data.frame() %>% rownames_to_column() %>% filter(. > 0) # customer
# customer_products_bought_wide["411",] %>% data.frame() %>% rownames_to_column() %>% arrange(desc(.)) 
# cs_2_indexedBy1["411",] %>% data.frame() %>% rownames_to_column() %>% filter(. > 0) # customer exists
# mf1_predicted["411",] %>% data.frame() %>% rownames_to_column() %>% filter(. > 0) # customer exists a

# customer_products_bought_wide["411",] %>% data.frame() %>% rownames_to_column() %>% arrange(desc(.)) 
# product_similarities1["TOP SELLING July Everyday Case",]  %>% data.frame() %>% rownames_to_column() %
```