Recommender

I have tried to build a Product recommendation algorithm. I followed Diego Usai’s article on towardsdatascience.com which can be found here https://towardsdatascience.com/clean-a-complex-dataset-for-modelling-with-recommendation-algorithms-c977f7ba28b1 (https://towardsdatascience.com/clean-a-complex-dataset-for-modelling-with-recommendation-algorithms-c977f7ba28b1)

This data set contains transactions occurring between 01/Dec/2010 and 09/Dec/2011 for a UK-based and registered online retail company. The company sells mainly unique all-occasion gifts and many of their customers are wholesalers.

Importing libraries

```
# install.packages("data.table")
# install.packages("readxl")
# install.packages("tidyverse")
# install.packages("lubridate")
# install.packages("skimr")
# install.packages("knitr")
# install.packages("treemap")
# install.packages("installr")
# install.packages("recommenderlab")

#
# library(installr)
#
# updateR()

#Getting Libraries
library(data.table)
library(readxl)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(skimr)
library(knitr)
library(treemap)
library(recommenderlab)
```

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

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

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

```
## Loading required package: arules
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
## Loading required package: proxy
```

```
##
## Attaching package: 'proxy'
```

```
## The following object is masked from 'package:Matrix':
##
##     as.matrix
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
## Loading required package: registry
```

```
## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

Importing raw data file

use read excel function and remove white space

```
retail <- read_excel("Online Retail.xlsx",
                     trim_ws = TRUE)
```

Inspect the data using skim() function

```
retail %>% skim()
```

Data summary

Name	Piped data
Number of rows	541909
Number of columns	8
_______________________	
Column type frequency:	
character	4
numeric	3
POSIXct	1
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
InvoiceNo	0	1	6	7	0	25900	0
StockCode	0	1	1	12	0	4070	0
Description	1454	1	1	35	0	4211	0
Country	0	1	3	20	0	38	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Quantity	0	1.00	9.55	218.08	-80995.00	1.00	3.00	10.00	80995	▁▁▇▁▁
UnitPrice	0	1.00	4.61	96.76	-11062.06	1.25	2.08	4.13	38970	▁▇▁▁▁
CustomerID	135080	0.75	15287.69	1713.60	12346.00	13953.00	15152.00	16791.00	18287	▇▇▇▇▇

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
InvoiceDate	0	1	2010-12-01 08:26:00	2011-12-09 12:50:00	2011-07-19 17:17:00	23260

Cleaning the data ^.^

Using the filter function

```
# a <- c(1,2,3,4) %>% as.data.frame()
# b <- c(T,T,F,T)
# a %>% filter(b)
```

Using grepl which generates TRUE/FALSE vector for each row by matching with a string, like Italy below

```
grepl("Italy", retail$Country) %>% table()
```

```
## .
##  FALSE   TRUE
## 541106    803
```

Using filter an grepl to remove rows that have a cancellation

Check how many rows have a cancellation using summarise() function

```
retail %>%
  filter(grepl("C", retail$InvoiceNo)) %>%
  summarise(Total = n())
```

```
## # A tibble: 1 x 1
##   Total
##   <int>
## 1  9288
```

Now remove the rows that have a cancellation

```
retail  <- retail %>%
  filter(!grepl("C", retail$InvoiceNo))
```

Making a table and using group_by function

```
# a <- c("Ant","Bat","Cow","Pig","Pig","Genda","Genda")
# b <- c(1,1,1,1,2,1,1)
# tbl <- cbind(a,b) %>% as.data.frame()
# colnames(tbl) <- c("animal","price")
# tbl %>% group_by(price) %>% summarise(count=n())
# tbl %>% group_by(animal) %>% summarise(count=n())
```

Group by 2 columns at once and get the count

```
# tbl %>% group_by(animal,price) %>% summarise(count=n())
# tbl %>% group_by(animal,price) %>% summarise(count=n()) %>% arrange(desc(count))
# tbl %>% group_by(animal,price) %>% summarise(count=n()) %>% arrange(desc(count)) %>% ungroup()
```

Using group_by to get quantities less than or equal to 0 and remove such rows

We group by description to see the reason for non positive quantities. Also group by unit price to check if there was a reason or were these just “adjustment codes”

```
retail %>%
  filter(Quantity <= 0) %>%
  group_by(Description, UnitPrice) %>%
  summarise(count =n()) %>%
  arrange(desc(count)) %>%
  ungroup()
```

```
## `summarise()` regrouping output by 'Description' (override with `.groups` argument)
```

```
## # A tibble: 139 x 3
##    Description            UnitPrice count
##    <chr>                      <dbl> <int>
##  1 <NA>                           0   862
##  2 check                          0   120
##  3 damages                        0    45
##  4 damaged                        0    42
##  5 ?                              0    41
##  6 sold as set on dotcom          0    20
##  7 Damaged                        0    14
##  8 thrown away                    0     9
##  9 Unsaleable, destroyed.         0     9
## 10 ??                             0     7
## # ... with 129 more rows
```

Remove the rows that have non positive quantities

```
retail  <- retail %>%
  filter(Quantity > 0)
```

Working with product codes

```
retail$StockCode
```

Evidently, the stock codes are numbers, so blog writer checked for weird stock codes that are not numbers

```
stc <- c('AMAZONFEE', 'BANK CHARGES', 'C2', 'DCGSSBOY',     'DCGSSGIRL', 'DOT', 'gift_0001_', 'PADS', 'POST')
```

Make a vector of all weird stock codes and concatenate them into a string using paste()

```
paste(stc, collapse="|")
```

```
## [1] "AMAZONFEE|BANK CHARGES|C2|DCGSSBOY|DCGSSGIRL|DOT|gift_0001_|PADS|POST"
```

Group by stock code and description to see what the deal is with these weird stock codes

```
retail %>%
  filter(grepl(paste(stc, collapse="|"), StockCode))  %>%
  group_by(StockCode, Description) %>%
  summarise(count =n()) %>%
  arrange(desc(count)) %>%
  ungroup()
```

```
## `summarise()` regrouping output by 'StockCode' (override with `.groups` argument)
```

```
## # A tibble: 19 x 3
##    StockCode    Description                         count
##    <chr>        <chr>                               <int>
##  1 POST         POSTAGE                              1126
##  2 DOT          DOTCOM POSTAGE                        708
##  3 C2           CARRIAGE                              141
##  4 DCGSSGIRL    GIRLS PARTY BAG                        13
##  5 BANK CHARGES Bank Charges                           12
##  6 DCGSSBOY     BOYS PARTY BAG                         11
##  7 gift_0001_20 Dotcomgiftshop Gift Voucher £20.00     9
##  8 gift_0001_10 Dotcomgiftshop Gift Voucher £10.00     8
##  9 gift_0001_30 Dotcomgiftshop Gift Voucher £30.00     7
## 10 gift_0001_50 Dotcomgiftshop Gift Voucher £50.00     4
## 11 PADS         PADS TO MATCH ALL CUSHIONS              4
## 12 POST         <NA>                                    4
## 13 gift_0001_40 Dotcomgiftshop Gift Voucher £40.00     3
## 14 AMAZONFEE    AMAZON FEE                              2
## 15 C2           <NA>                                    1
## 16 DOT          <NA>                                    1
## 17 gift_0001_10 <NA>                                    1
## 18 gift_0001_20 to push order througha s stock was     1
## 19 gift_0001_30 <NA>                                    1
```

Since these are less than 2000, a small number, these can be removed

```
retail <- filter(retail,
                 !grepl(paste(stc, collapse="|"), StockCode))
```

Working with the %in% operator

Make a new variable fav and assign it Pig. Now check if fav is in animal column of table

```
# tbl
# fav <- "Pig"
# fav %in% tbl$animal
```

Next, use the in operator the other way round. Is animal column of table in fav?

```
# tbl$animal %in% fav
```

Working with is.na function

This function returns true of a value is NA and false otherwise

```
# a <- c(1,2,3,4,5,NA,6)
# a %>% is.na
```

Use %in% and is.na to remove weird descriptions and NA description rows

Make a vector of all weird descriptions

```
descr <- c( "check", "check?", "?", "??", "damaged", "found",
            "adjustment", "Amazon", "AMAZON", "amazon adjust",
            "Amazon Adjustment", "amazon sales", "Found", "FOUND",
            "found box", "Found by jackie ","Found in w/hse","dotcom",
            "dotcom adjust", "allocate stock for dotcom orders ta", "FBA", "Dotcomgiftshop Gift Voucher £100.0
0", "on cargo order",
            "wrongly sold (22719) barcode", "wrongly marked 23343",
            "dotcomstock", "rcvd be air temp fix for dotcom sit",
            "Manual", "John Lewis", "had been put aside",
            "for online retail orders", "taig adjust", "amazon",
            "incorrectly credited C550456 see 47", "returned",
            "wrongly coded 20713", "came coded as 20713",
            "add stock to allocate online orders", "Adjust bad debt",
            "alan hodge cant mamage this section", "website fixed",
            "did  a credit  and did not tick ret", "michel oops",
            "incorrectly credited C550456 see 47", "mailout", "test",
            "Sale error",  "Lighthouse Trading zero invc incorr", "SAMPLES",
            "Marked as 23343", "wrongly coded 23343","Adjustment",
            "rcvd be air temp fix for dotcom sit", "Had been put aside." )
```

Remove all these rows

```
retail <- retail %>%
  filter(!(Description %in% descr))
```

Now check for NA descriptions

```
sum(is.na(retail$Description))
```

```
## [1] 584
```

Only 584 rows, can be removed

```
retail <- retail %>%
  filter(!is.na(Description))
```

So, there were 7 columns, We have cleaned InvoiceNumber, , StockCode, Description. Now we come to CustomerId

```
retail$CustomerID %>% skim()
```

Data summary

Name	Piped data
Number of rows	528148
Number of columns	1
________________________	
Column type frequency:	
numeric	1
________________________	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
data	131778	0.75	15301.6	1709.98	12346	13975	15159	16803	18287	▇▇▇▇▇▇▇

There are a significant number of NAs here, 1.3 lakh, so we will leave them as it is

Check for number of unique Invoice numbers, since customer ids are so many missing

```
sapply(retail[ ,c('InvoiceNo','CustomerID')],
               function(x) length(unique(x)))
```

```
##  InvoiceNo CustomerID
##      19792       4336
```

Changing data types of columns

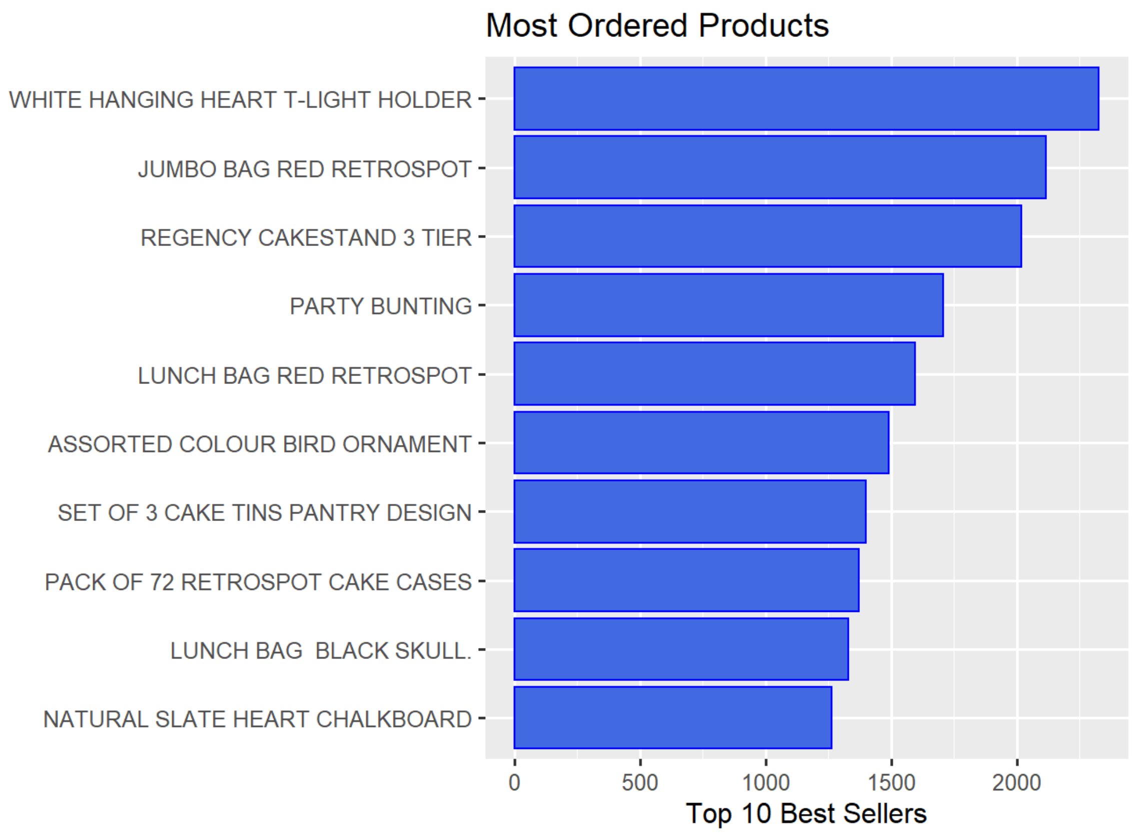
```
retail <- retail %>%
  mutate(Description = as.factor(Description)) %>%
  mutate(Country = as.factor(Country)) %>%
  mutate(InvoiceNo = as.numeric(InvoiceNo)) %>%
  mutate(Date = as.Date(InvoiceDate)) %>%
  mutate(Time = as.factor(format(InvoiceDate,"%H:%M:%S")))
```

Exploring the data

Checking the most popular items

```
retail %>%
  group_by(Description) %>%
  summarize(count = n()) %>%
  top_n(10, wt = count) %>%
  arrange(desc(count)) %>%
  ggplot(aes(x = reorder(Description, count), y = count))+
  geom_bar(stat = "identity", fill = "royalblue", colour = "blue") +
  labs(x = "", y = "Top 10 Best Sellers", title = "Most Ordered Products") +
  coord_flip() +
  theme_grey(base_size = 12)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Checking the percentage that top 10 items constitute

```
retail %>%
  group_by(Description) %>%
  summarize(count = n()) %>%
  mutate(pct=(count/sum(count))*100) %>%
  arrange(desc(pct)) %>%
  ungroup() %>%
  top_n(10, wt=pct)
```

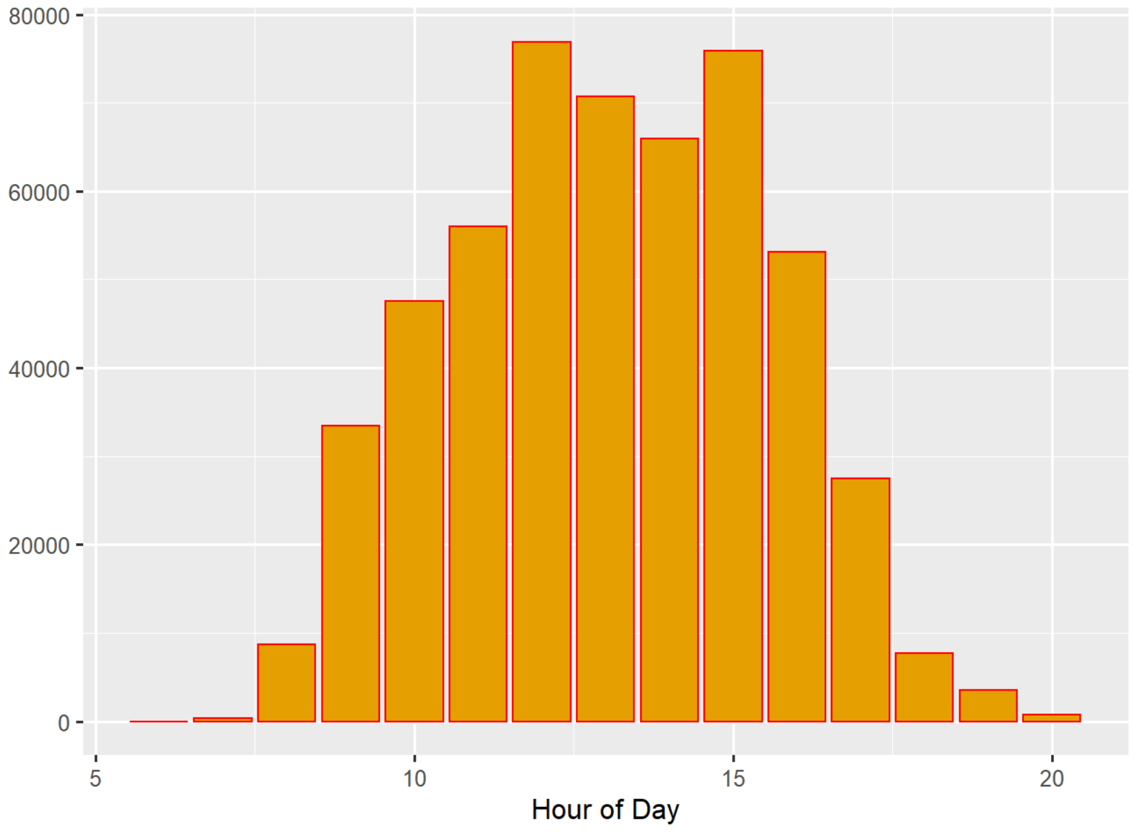
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 10 x 3
##    Description                        count   pct
##    <fct>                              <int> <dbl>
##  1 WHITE HANGING HEART T-LIGHT HOLDER  2327 0.441
##  2 JUMBO BAG RED RETROSPOT             2115 0.400
##  3 REGENCY CAKESTAND 3 TIER            2019 0.382
##  4 PARTY BUNTING                       1707 0.323
##  5 LUNCH BAG RED RETROSPOT             1594 0.302
##  6 ASSORTED COLOUR BIRD ORNAMENT       1489 0.282
##  7 SET OF 3 CAKE TINS PANTRY DESIGN    1399 0.265
##  8 PACK OF 72 RETROSPOT CAKE CASES     1370 0.259
##  9 LUNCH BAG  BLACK SKULL.             1328 0.251
## 10 NATURAL SLATE HEART CHALKBOARD      1263 0.239
```

Checking at what time of the day people most buy the products

```
retail %>%
  ggplot(aes(hour(hms(Time)))) +
  geom_histogram(stat = "count",fill = "#E69F00", colour = "red") +
  labs(x = "Hour of Day", y = "") +
  theme_grey(base_size = 12)
```

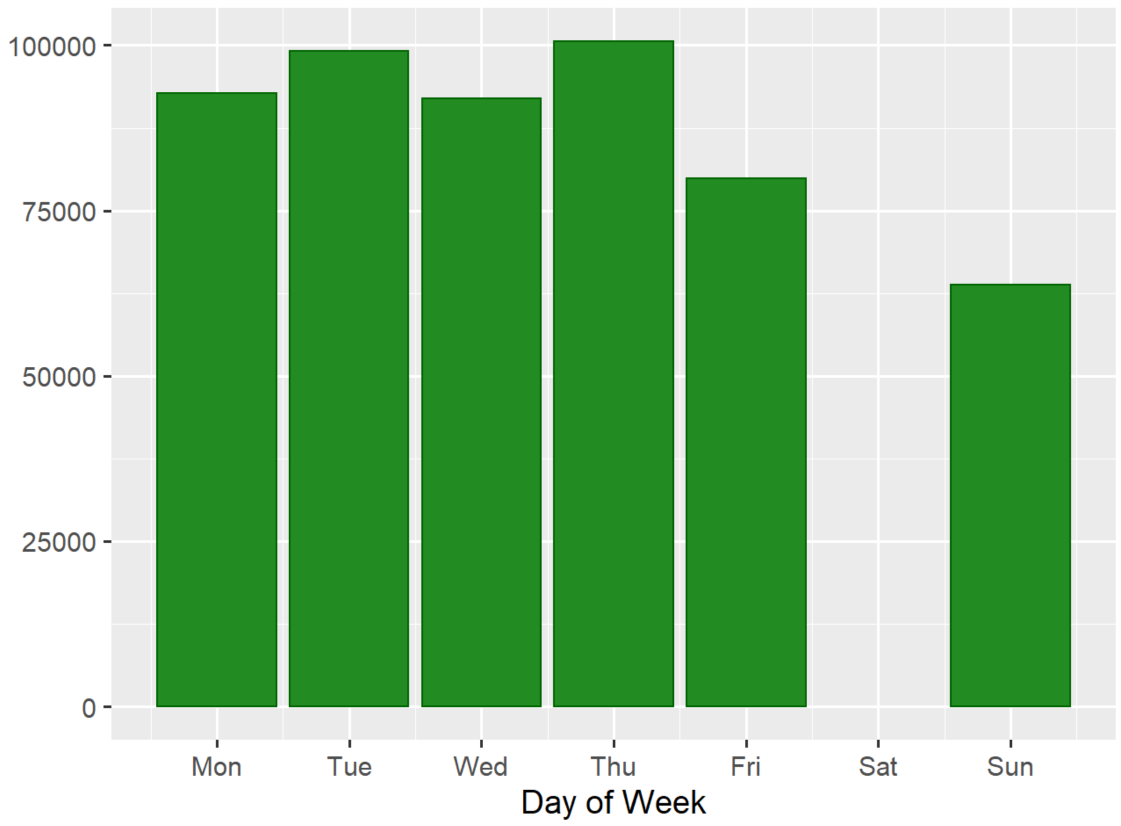
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



What day of the week people buy more often?

```
retail %>%
  ggplot(aes(wday(Date,
                  week_start = getOption("lubridate.week.start", 1)))) +
  geom_histogram(stat = "count" , fill = "forest green", colour = "dark green") +
  labs(x = "Day of Week", y = "") +
  scale_x_continuous(breaks = c(1,2,3,4,5,6,7),
                     labels = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")) +
  theme_grey(base_size = 14)
```

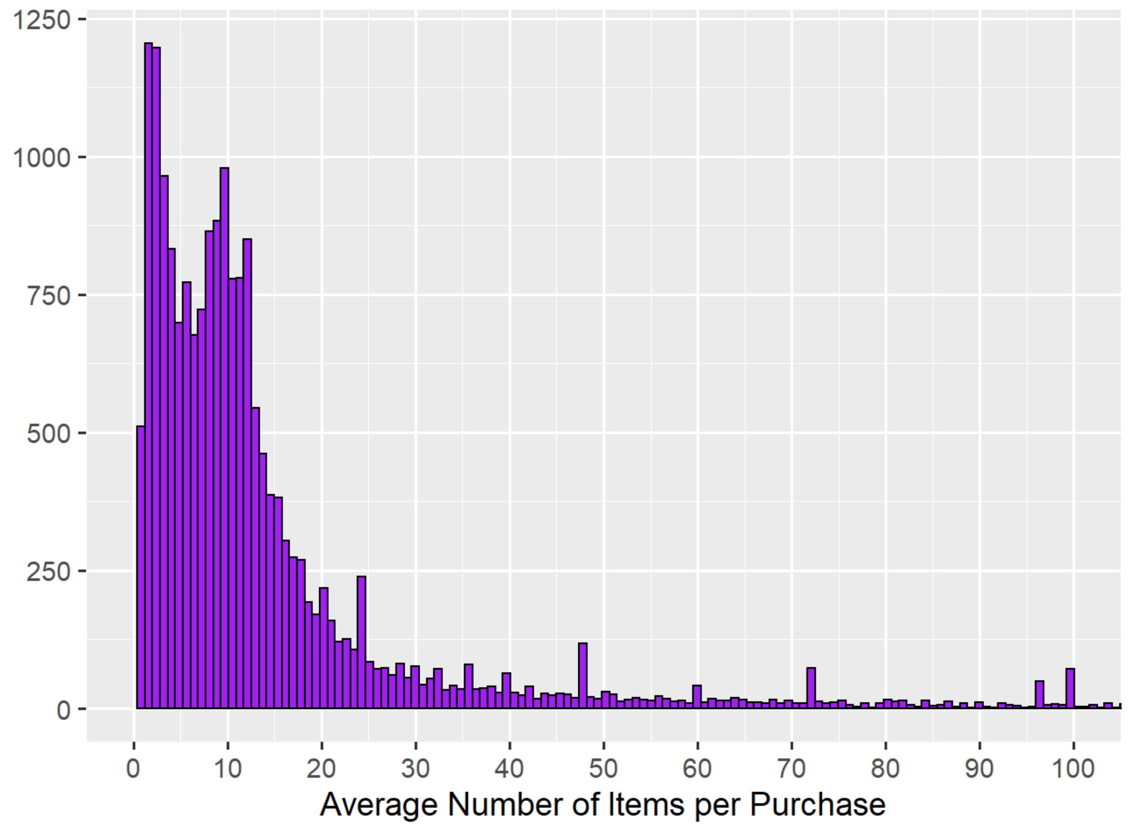
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



How many items each customer buy?

```
retail %>%
  group_by(InvoiceNo) %>%
  summarise(n = mean(Quantity)) %>%
  ggplot(aes(x=n)) +
  geom_histogram(bins = 100000,fill = "purple",colour = "black") +
  coord_cartesian(xlim=c(0,100)) +
  scale_x_continuous(breaks=seq(0,100,10)) +
  labs(x = "Average Number of Items per Purchase", y = "") +
  theme_grey(base_size = 14)
```

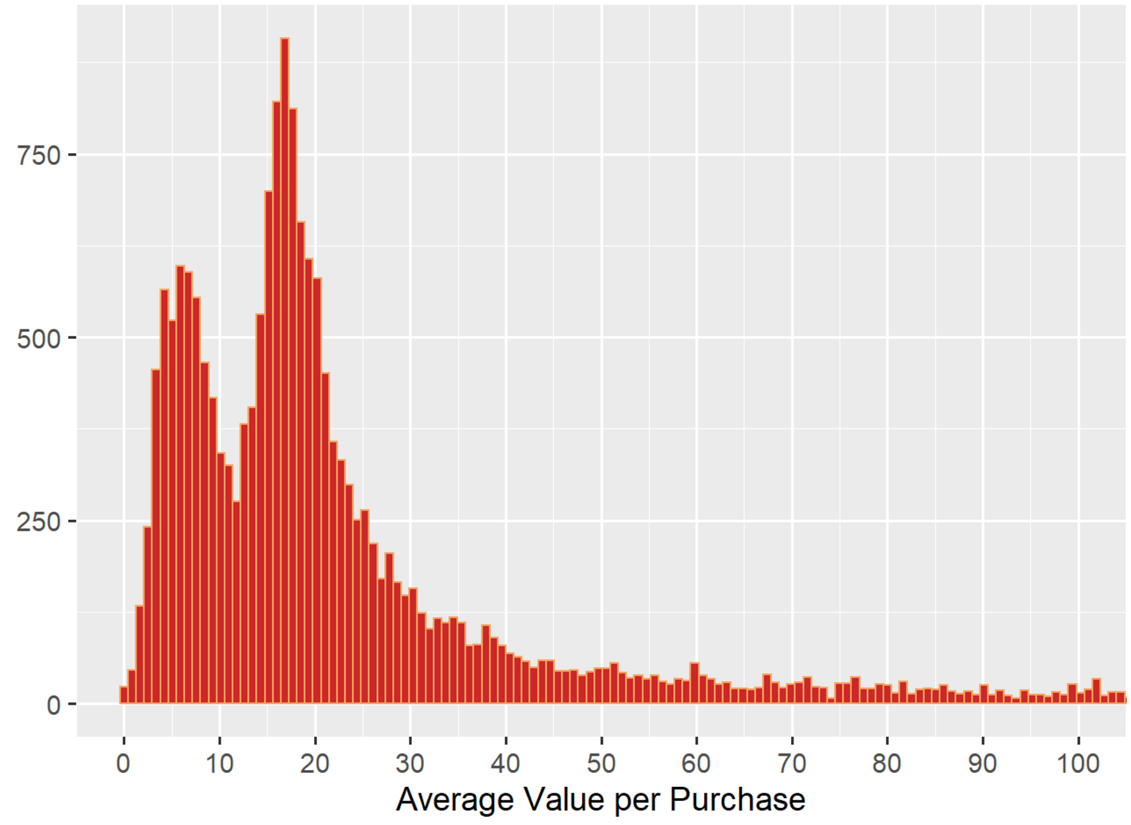
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Average value per order

```
retail %>%
  mutate(Value = UnitPrice * Quantity) %>%
  group_by(InvoiceNo) %>%
  summarise(n = mean(Value)) %>%
  ggplot(aes(x=n)) +
  geom_histogram(bins = 200000, fill="firebrick3", colour = "sandybrown") +
  coord_cartesian(xlim=c(0,100)) +
  scale_x_continuous(breaks=seq(0,100,10)) +
  labs(x = "Average Value per Purchase", y = "") +
  theme_grey(base_size = 14)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Applying various product recommendation models

Make a column joining invoice no and description and remove duplicate rows

```
retail <- retail %>%
    mutate(InNo_Desc = paste(InvoiceNo, Description, sep = ' '))
    retail <- retail[!duplicated(retail$InNo_Desc), ] %>%
    select(-InNo_Desc)
```

Create a ratings matrix

```
ratings_matrix <- retail %>%

  select(InvoiceNo, Description) %>%

  mutate(value = 1) %>%

  spread(Description, value, fill = 0) %>%
  select(-InvoiceNo) %>%

  as.matrix() %>%

  as("binaryRatingMatrix")
```

Setting the evaluation scheme: 5-fold cross validation

```
scheme <- ratings_matrix %>%
  evaluationScheme(method = "cross",
                   k      = 5,
                   train  = 0.8,
                   given  = -1)
```

```
## Warning in .local(data, ...): The following users do not have enough items
## leaving no given items: 5, 7, 10, 15, 26, 33, 36, 38, 52, 62, 82, 89, 99, 105,
## 112, 114, 115, 116, 145, 153, 163, 168, 172, 187, 217, 224, 228, 239, 244, 245,
## 246, 251, 256, 257, 284, 287, 298, 302, 304, 305, 325, 335, 362, 364, 394, 398,
## 419, 427, 428, 429, 446, 448, 469, 473, 489, 508, 510, 529, 534, 535, 542, 544,
## 559, 571, 585, 607, 610, 618, 647, 652, 679, 691, 696, 697, 701, 708, 709, 719,
## 745, 775, 780, 781, 802, 837, 846, 849, 850, 851, 881, 883, 892, 902, 925, 935,
## 947, 963, 969, 1015, 1017, 1033, 1034, 1035, 1037, 1045, 1062, 1073, 1091, 1105,
## 1109, 1141, 1144, 1152, 1156, 1161, 1174, 1177, 1178, 1189, 1203, 1212, 1216,
## 1233, 1259, 1272, 1284, 1290, 1310, 1311, 1324, 1330, 1333, 1351, 1352, 1370,
## 1393, 1404, 1410, 1412, 1422, 1423, 1428, 1444, 1445, 1446, 1451, 1454, 1460,
## 1461, 1466, 1467, 1468, 1473, 1476, 1480, 1483, 1485, 1501, 1506, 1507, 1510,
## 1515, 1526, 1547, 1549, 1557, 1569, 1574, 1592, 1601, 1616, 1617, 1618, 1631,
## 1640, 1648, 1650, 1661, 1683, 1688, 1694, 1701, 1712, 1726, 1735, 1744, 1801,
## 1804, 1805, 1813, 1824, 1856, 1859, 1872, 1883, 1886, 1890, 1910, 1944, 1960,
## 1972, 1978, 1980, 1986, 1987, 2027, 2033, 2035, 2036, 2043, 2062, 2086, 2090,
## 2094, 2095, 2099, 2104, 2105, 2110, 2142, 2151, 2221, 2227, 2260, 2282, 2301,
## 2369, 2390, 2398, 2403, 2411, 2412, 2426, 2436, 2449, 2455, 2537, 2546, 2547,
## 2567, 2582, 2599, 2602, 2626, 2632, 2633, 2675, 2686, 2692, 2695, 2702, 2715,
## 2737, 2738, 2801, 2835, 2884, 2907, 2908, 2915, 2933, 2937, 2948, 2950, 2963,
## 3003, 3012, 3036, 3038, 3041, 3043, 3054, 3061, 3069, 3103, 3119, 3121, 3137,
## 3139, 3156, 3180, 3181, 3186, 3203, 3219, 3221, 3222, 3237, 3246, 3252, 3306,
## 3310, 3316, 3317, 3352, 3353, 3354, 3357, 3361, 3364, 3391, 3394, 3426, 3427,
## 3431, 3434, 3482, 3483, 3498, 3509, 3513, 3521, 3523, 3525, 3535, 3558, 3582,
## 3586, 3595, 3610, 3616, 3657, 3663, 3678, 3689, 3718, 3733, 3738, 3750, 3756,
## 3785, 3792, 3798, 3810, 3814, 3817, 3820, 3825, 3830, 3838, 3852, 3887, 3933,
## 3948, 3957, 3975, 3982, 3989, 4003, 4008, 4022, 4059, 4061, 4069, 4075, 4079,
## 4100, 4109, 4127, 4134, 4172, 4180, 4185, 4186, 4198, 4214, 4217, 4226, 4261,
## 4266, 4289, 4297, 4303, 4313, 4335, 4361, 4367, 4371, 4375, 4397, 4412, 4413,
## 4427, 4444, 4455, 4456, 4472, 4504, 4516, 4517, 4525, 4607, 4608, 4623, 4631,
## 4684, 4685, 4699, 4720, 4735, 4751, 4813, 4818, 4826, 4827, 4847, 4856, 4862,
## 4885, 4895, 4950, 4952, 4966, 4973, 5028, 5036, 5037, 5039, 5041, 5044, 5063,
## 5104, 5107, 5195, 5200, 5203, 5206, 5207, 5210, 5214, 5215, 5217, 5221, 5230,
## 5244, 5266, 5274, 5283, 5300, 5331, 5349, 5367, 5371, 5388, 5389, 5416, 5417,
## 5494, 5511, 5524, 5527, 5528, 5530, 5542, 5547, 5585, 5591, 5593, 5606, 5607,
## 5616, 5621, 5624, 5679, 5685, 5710, 5720, 5723, 5737, 5744, 5751, 5772, 5817,
## 5818, 5852, 5887, 5909, 5910, 5938, 5946, 5959, 5970, 5997, 6012, 6025, 6035,
## 6036, 6069, 6076, 6083, 6085, 6093, 6120, 6139, 6169, 6176, 6178, 6181, 6233,
## 6243, 6251, 6254, 6255, 6258, 6260, 6274, 6275, 6288, 6292, 6307, 6313, 6320,
## 6328, 6330, 6335, 6340, 6379, 6385, 6386, 6403, 6412, 6418, 6426, 6431, 6443,
## 6447, 6449, 6453, 6459, 6481, 6492, 6497, 6522, 6534, 6535, 6540, 6556, 6567,
## 6610, 6622, 6633, 6654, 6658, 6670, 6679, 6680, 6691, 6695, 6746, 6804, 6805,
## 6823, 6845, 6852, 6853, 6854, 6865, 6883, 6886, 6906, 6908, 6909, 6959, 6960,
## 6979, 6986, 6987, 7005, 7016, 7034, 7050, 7059, 7086, 7111, 7114, 7123, 7127,
## 7129, 7131, 7136, 7139, 7150, 7170, 7172, 7174, 7232, 7233, 7250, 7264, 7274,
## 7291, 7303, 7310, 7311, 7312, 7319, 7334, 7335, 7364, 7365, 7371, 7381, 7399,
## 7405, 7419, 7448, 7462, 7478, 7486, 7488, 7502, 7505, 7508, 7514, 7529, 7550,
## 7553, 7567, 7574, 7588, 7597, 7601, 7602, 7610, 7611, 7613, 7628, 7645, 7677,
## 7683, 7701, 7703, 7704, 7714, 7716, 7721, 7747, 7757, 7783, 7802, 7804, 7828,
## 7830, 7835, 7838, 7843, 7845, 7883, 7893, 7897, 7898, 7930, 7945, 7957, 7960,
## 7961, 7974, 7980, 7982, 7995, 8032, 8042, 8051, 8052, 8057, 8113, 8124, 8156,
## 8159, 8181, 8183, 8191, 8196, 8197, 8210, 8233, 8236, 8263, 8281, 8296, 8300,
## 8334, 8335, 8354, 8355, 8361, 8362, 8368, 8372, 8394, 8402, 8450, 8467, 8481,
## 8503, 8504, 8505, 8523, 8583, 8594, 8624, 8640, 8642, 8643, 8645, 8647, 8684,
## 8694, 8701, 8705, 8708, 8751, 8754, 8761, 8765, 8784, 8787, 8797, 8800, 8807,
## 8820, 8821, 8829, 8849, 8857, 8875, 8881, 8895, 8910, 8912, 8924, 8928, 8950,
## 8954, 8969, 8983, 8984, 8996, 8999, 9015, 9061, 9072, 9076, 9084, 9087, 9088,
## 9089, 9092, 9102, 9106, 9121, 9125, 9128, 9130, 9145, 9147, 9155, 9188, 9212,
## 9225, 9234, 9251, 9276, 9280, 9298, 9302, 9332, 9362, 9368, 9376, 9391, 9407,
## 9423, 9429, 9433, 9435, 9440, 9446, 9458, 9463, 9475, 9476, 9478, 9482, 9483,
## 9496, 9517, 9530, 9536, 9544, 9545, 9546, 9558, 9563, 9576, 9590, 9610, 9612,
## 9624, 9643, 9644, 9647, 9698, 9721, 9731, 9733, 9737, 9741, 9777, 9820, 9823,
## 9830, 9840, 9865, 9871, 9874, 9875, 9880, 9911, 9933, 9938, 9967, 9983, 10007,
## 10009, 10011, 10023, 10038, 10056, 10068, 10077, 10089, 10111, 10122, 10137,
## 10142, 10156, 10165, 10190, 10194, 10206, 10233, 10252, 10290, 10291, 10296,
## 10302, 10352, 10377, 10389, 10392, 10426, 10430, 10448, 10468, 10470, 10471,
## 10473, 10479, 10518, 10529, 10533, 10536, 10561, 10573, 10576, 10588, 10603,
## 10607, 10609, 10644, 10646, 10647, 10670, 10689, 10691, 10694, 10713, 10730,
## 10779, 10807, 10826, 10857, 10864, 10890, 10893, 10908, 10916, 10917, 10932,
## 10942, 10948, 10949, 10952, 10971, 11001, 11002, 11003, 11048, 11074, 11076,
## 11100, 11112, 11122, 11127, 11176, 11191, 11193, 11197, 11198, 11224, 11230,
## 11262, 11265, 11271, 11273, 11276, 11279, 11282, 11294, 11297, 11319, 11320,
## 11323, 11335, 11361, 11364, 11379, 11382, 11442, 11443, 11460, 11464, 11480,
## 11482, 11498, 11501, 11505, 11558, 11564, 11580, 11583, 11596, 11600, 11610,
## 11641, 11643, 11644, 11659, 11691, 11707, 11718, 11759, 11771, 11783, 11788,
## 11792, 11795, 11801, 11813, 11851, 11864, 11866, 11908, 11915, 11924, 11953,
## 11955, 11977, 11997, 11998, 11999, 12018, 12032, 12051, 12054, 12055, 12085,
## 12102, 12139, 12147, 12154, 12172, 12173, 12176, 12180, 12210, 12238, 12242,
## 12243, 12247, 12267, 12276, 12278, 12282, 12288, 12296, 12329, 12330, 12335,
## 12359, 12364, 12369, 12373, 12378, 12380, 12388, 12398, 12403, 12422, 12429,
## 12432, 12442, 12443, 12446, 12465, 12523, 12531, 12532, 12539, 12545, 12575,
## 12609, 12618, 12622, 12631, 12642, 12674, 12676, 12686, 12692, 12704, 12709,
## 12724, 12730, 12741, 12762, 12794, 12803, 12820, 12830, 12831, 12867, 12869,
## 12883, 12897, 12905, 12908, 12931, 12940, 12987, 13004, 13033, 13034, 13054,
## 13069, 13079, 13121, 13141, 13148, 13167, 13190, 13212, 13217, 13264, 13267,
## 13306, 13314, 13348, 13392, 13414, 13416, 13425, 13444, 13468, 13469, 13532,
## 13533, 13554, 13576, 13588, 13639, 13640, 13643, 13672, 13792, 13811, 13878,
## 13889, 13900, 13919, 13938, 13971, 13991, 14008, 14033, 14049, 14050, 14083,
## 14093, 14094, 14100, 14104, 14119, 14132, 14176, 14177, 14178, 14182, 14184,
## 14191, 14209, 14248, 14291, 14292, 14297, 14304, 14314, 14316, 14331, 14340,
## 14356, 14364, 14373, 14378, 14393, 14397, 14398, 14401, 14425, 14432, 14439,
## 14447, 14448, 14463, 14476, 14483, 14488, 14503, 14518, 14530, 14566, 14575,
## 14576, 14593, 14598, 14628, 14669, 14672, 14678, 14679, 14686, 14700, 14719,
## 14741, 14752, 14767, 14776, 14786, 14809, 14822, 14830, 14833, 14835, 14837,
## 14840, 14856, 14865, 14896, 14928, 14930, 14935, 14942, 14945, 14960, 14984,
## 14987, 15029, 15043, 15049, 15055, 15057, 15073, 15080, 15095, 15096, 15107,
## 15110, 15112, 15132, 15135, 15136, 15148, 15150, 15153, 15155, 15187, 15233,
## 15247, 15281, 15295, 15298, 15344, 15347, 15372, 15390, 15410, 15424, 15427,
## 15432, 15441, 15442, 15477, 15487, 15494, 15500, 15502, 15505, 15524, 15572,
## 15639, 15646, 15657, 15695, 15709, 15751, 15776, 15797, 15799, 15811, 15820,
## 15831, 15837, 15841, 15894, 15897, 15902, 15904, 15945, 15951, 15958, 15988,
## 15996, 16002, 16005, 16011, 16013, 16014, 16017, 16027, 16051, 16052, 16057,
## 16060, 16063, 16065, 16127, 16147, 16160, 16187, 16220, 16293, 16309, 16314,
## 16318, 16325, 16332, 16346, 16362, 16423, 16426, 16439, 16440, 16443, 16451,
## 16474, 16484, 16487, 16525, 16527, 16552, 16568, 16579, 16580, 16605, 16607,
## 16627, 16647, 16662, 16667, 16689, 16697, 16714, 16734, 1680
```

```
scheme
```

```
## Evaluation scheme using all-but-1 items
## Method: 'cross-validation' with 5 run(s).
## Good ratings: NA
## Data set: 19792 x 4001 rating matrix of class 'binaryRatingMatrix' with 517354 ratings.
```

Create a list of algorithms

```
algorithms <- list(
  "association rules" = list(name  = "AR",
                        param = list(supp = 0.01, conf = 0.01)),
  "random items"      = list(name  = "RANDOM",  param = NULL),
  "popular items"     = list(name  = "POPULAR", param = NULL),
  "item-based CF"     = list(name  = "IBCF", param = list(k = 5)),
  "user-based CF"     = list(name  = "UBCF",
                        param = list(method = "Cosine", nn = 500))
                   )
algorithms
```

```
## $`association rules`
## $`association rules`$name
## [1] "AR"
##
## $`association rules`$param
## $`association rules`$param$supp
## [1] 0.01
##
## $`association rules`$param$conf
## [1] 0.01
##
##
##
## $`random items`
## $`random items`$name
## [1] "RANDOM"
##
## $`random items`$param
## NULL
##
##
## $`popular items`
## $`popular items`$name
## [1] "POPULAR"
##
## $`popular items`$param
## NULL
##
##
## $`item-based CF`
## $`item-based CF`$name
## [1] "IBCF"
##
## $`item-based CF`$param
## $`item-based CF`$param$k
## [1] 5
##
##
##
## $`user-based CF`
## $`user-based CF`$name
## [1] "UBCF"
##
## $`user-based CF`$param
## $`user-based CF`$param$method
## [1] "Cosine"
##
## $`user-based CF`$param$nn
## [1] 500
```

Estimate the models using evaluate() function

```
results <- recommenderlab::evaluate(scheme,
                                    algorithms,
                                    type  = "topNList",
                                    n     = c(1, 3, 5, 10, 15, 20)
                                    )
```

```
## AR run fold/sample [model time/prediction time]
##   1  [1.71sec/109.97sec]
##   2  [0.64sec/122.88sec]
##   3  [0.9sec/122.61sec]
##   4  [0.54sec/107.69sec]
##   5  [0.51sec/110.11sec]
## RANDOM run fold/sample [model time/prediction time]
##   1  [0sec/37.59sec]
##   2  [0sec/37.55sec]
##   3  [0sec/37.26sec]
##   4  [0sec/42.56sec]
##   5  [0sec/62.37sec]
## POPULAR run fold/sample [model time/prediction time]
##   1  [0.01sec/60.38sec]
##   2  [0.01sec/65.91sec]
##   3  [0.02sec/43.04sec]
##   4  [0.01sec/44.21sec]
##   5  [0.01sec/62.33sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [577.82sec/13.08sec]
##   2  [553.18sec/11.57sec]
##   3  [564.63sec/5.26sec]
##   4  [501.22sec/10.22sec]
##   5  [507.4sec/15.23sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.01sec/936.58sec]
##   2  [0sec/1019.92sec]
##   3  [0sec/1094.25sec]
##   4  [0.02sec/1147.71sec]
##   5  [0sec/1069.66sec]
```

```
results
```

```
## List of evaluation results for 5 recommenders:
## Evaluation results for 5 folds/samples using method 'AR'.
## Evaluation results for 5 folds/samples using method 'RANDOM'.
## Evaluation results for 5 folds/samples using method 'POPULAR'.
## Evaluation results for 5 folds/samples using method 'IBCF'.
## Evaluation results for 5 folds/samples using method 'UBCF'.
```

Visualizing the results

Pull all confusion matrix information for one model in a list

```
tmp <- results$`user-based CF` %>%
  getConfusionMatrix()  %>%
  as.list()
```

Average of 5 cross-validation rounds, add a column for number of recommendations, and select only needed columns

```
  as.data.frame( Reduce("+",tmp) / length(tmp)) %>%
  mutate(n = c(1, 3, 5, 10, 15, 20)) %>%
  select('n', 'precision', 'recall', 'TPR', 'FPR')
```

```
##    n    precision       recall          TPR          FPR
## 1  1 0.0007071090 0.0007070707 0.0007070707 0.0002497355
## 2  3 0.0006565954 0.0012626263 0.0012626263 0.0007492445
## 3  5 0.0006894297 0.0020202020 0.0020202020 0.0012486904
## 4 10 0.0008409601 0.0051515152 0.0051515152 0.0024969641
## 5 15 0.0007938161 0.0068181818 0.0068181818 0.0037456040
## 6 20 0.0007803490 0.0086868687 0.0086868687 0.0049941934
```

```
avg_conf_matr <- function(results) {
  tmp <- results %>%
    getConfusionMatrix()  %>%
    as.list()
    as.data.frame(Reduce("+",tmp) / length(tmp)) %>%
    mutate(n = c(1, 3, 5, 10, 15, 20)) %>%
    select('n', 'precision', 'recall', 'TPR', 'FPR')
}
```

Use map() function to get all results in a tidy format

```
results_tbl <- results %>%
  map(avg_conf_matr) %>% enframe() %>%  unnest()
```

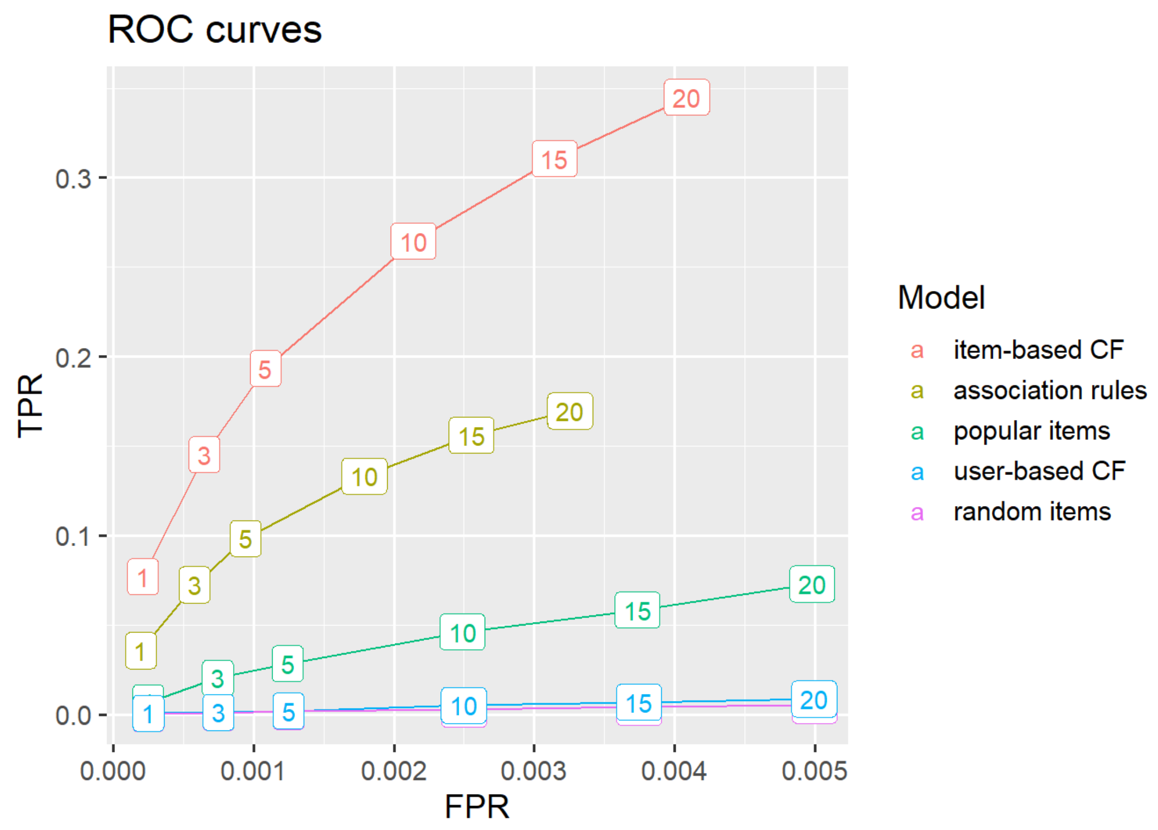
```
## Warning: `cols` is now required when using unnest().
## Please use `cols = c(value)`
```

```
results_tbl
```

```
## # A tibble: 30 x 6
##    name                  n precision   recall      TPR      FPR
##    <chr>             <dbl>     <dbl>    <dbl>    <dbl>    <dbl>
##  1 association rules     1  0.0434   0.0357   0.0357   0.000197
##  2 association rules     3  0.0323   0.0725   0.0725   0.000579
##  3 association rules     5  0.0285   0.0984   0.0984   0.000943
##  4 association rules    10  0.0229   0.133    0.133    0.00179
##  5 association rules    15  0.0206   0.156    0.156    0.00255
##  6 association rules    20  0.0191   0.170    0.170    0.00325
##  7 random items          1  0.000455 0.000455 0.000455 0.000250
##  8 random items          3  0.000337 0.00101  0.00101  0.000750
##  9 random items          5  0.000333 0.00167  0.00167  0.00125
## 10 random items         10  0.000298 0.00298  0.00298  0.00250
## # ... with 20 more rows
```

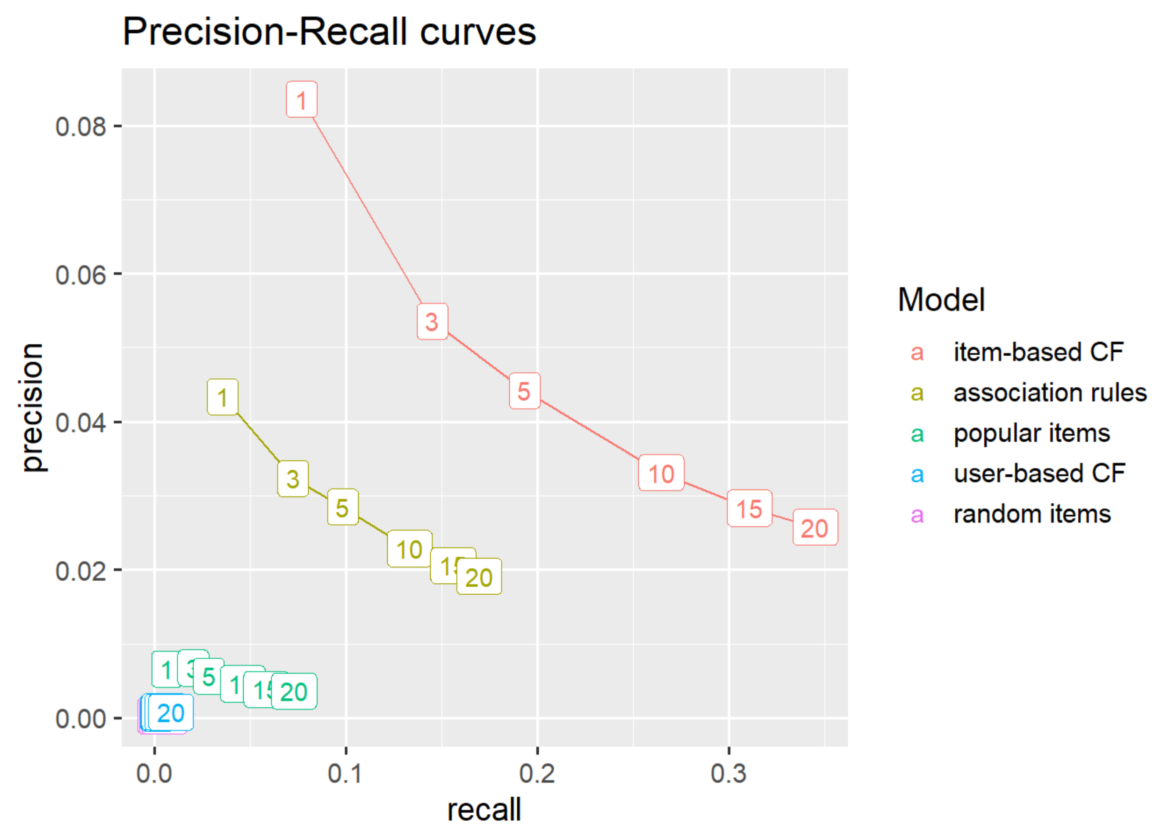
Plotting the ROC curve

```
results_tbl %>%
  ggplot(aes(FPR, TPR,
             colour = fct_reorder2(as.factor(name),
                          FPR, TPR))) +
  geom_line() +
  geom_label(aes(label = n))  +
  labs(title = "ROC curves", colour = "Model") +
  theme_grey(base_size = 14)
```



Plotting the Precision-Recall Curve

```
results_tbl %>%
  ggplot(aes(recall, precision,
             colour = fct_reorder2(as.factor(name),
                          precision, recall))) +
  geom_line() +
  geom_label(aes(label = n))  +
  labs(title = "Precision-Recall curves", colour = "Model") +
  theme_grey(base_size = 14)
```



Predictions for a new user

Finally, we generate predictions for a new user. First create a random customer order.

```
customer_order <- c("GREEN REGENCY TEACUP AND SAUCER",
                    "SET OF 3 BUTTERFLY COOKIE CUTTERS",
                    "JAM MAKING SET WITH JARS",
                    "SET OF TEA COFFEE SUGAR TINS PANTRY",
                    "SET OF 4 PANTRY JELLY MOULDS")
```

Put this order in a format that recommenderlab accepts.

```
new_order_rat_matrx <- retail %>%
select(Description) %>%
  unique() %>%   mutate(value = as.numeric(Description %in% customer_order)) %>%
  spread(key = Description, value = value) %>%
  as.matrix() %>%
  as("binaryRatingMatrix")
```

Now create a recommender, with the best method found above, i.e., Item Based Collaborative Filtering

```
recomm <- Recommender(getData(scheme, 'train'),
                      method = "IBCF",
                      param = list(k = 5))
recomm
```

```
## Recommender of type 'IBCF' for 'binaryRatingMatrix'
## learned using 15832 users.
```

Create a list of top 10 recommendations

```
pred <- predict(recomm,
                newdata = new_order_rat_matrx,
                n       = 10)
```

Inspect the suggested items as a list

```
as(pred, 'list')
```

```
## $`1`
##  [1] "ROSES REGENCY TEACUP AND SAUCER"   "PINK REGENCY TEACUP AND SAUCER"
##  [3] "SET OF 3 HEART COOKIE CUTTERS"     "REGENCY CAKESTAND 3 TIER"
##  [5] "JAM MAKING SET PRINTED"            "SET OF 3 CAKE TINS PANTRY DESIGN"
##  [7] "GINGERBREAD MAN COOKIE CUTTER"     "RECIPE BOX PANTRY YELLOW DESIGN"
##  [9] "SET OF 3 REGENCY CAKE TINS"        "SET OF 6 SPICE TINS PANTRY DESIGN"
```