Log for eCommerce in R

by Ava Lee 2/14/2021

```
> # Read data
> custData<- read.csv(file.choose(), header=T)
>
> # Descriptive statistics
> head(custData)
  InvoiceNo StockCode                          Description Quantity InvoiceDate
1    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER        6    29-Nov-16
2    536365     71053                  WHITE METAL LANTERN        6    29-Nov-16
3    536365    84406B       CREAM CUPID HEARTS COAT HANGER        8    29-Nov-16
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6    29-Nov-16
5    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.        6    29-Nov-16
6    536365     22752         SET 7 BABUSHKA NESTING BOXES        2    29-Nov-16
  UnitPrice CustomerID        Country  X
1      2.55      17850 United Kingdom NA
2      3.39      17850 United Kingdom NA
3      2.75      17850 United Kingdom NA
4      3.39      17850 United Kingdom NA
5      3.39      17850 United Kingdom NA
6      7.65      17850 United Kingdom NA
> glimpse(custData)
Rows: 541,909
Columns: 9
$ InvoiceNo   <chr> "536365", "536365", "536365", "536365", "536365", "536365…
$ StockCode   <chr> "85123A", "71053", "84406B", "84029G", "84029E", "22752",…
$ Description <chr> "WHITE HANGING HEART T-LIGHT HOLDER", "WHITE METAL LANTER…
$ Quantity    <int> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3, 3,…
$ InvoiceDate <chr> "29-Nov-16", "29-Nov-16", "29-Nov-16", "29-Nov-16", "29-N…
$ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85, 1.6…
$ CustomerID  <int> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850, 1…
$ Country     <chr> "United Kingdom", "United Kingdom", "United Kingdom", "Un…
$ X           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
> summary(custData)
  InvoiceNo          StockCode          Description           Quantity
 Length:541909      Length:541909      Length:541909      Min.   :-80995.00
 Class :character   Class :character   Class :character   1st Qu.:     1.00
 Mode  :character   Mode  :character   Mode  :character   Median :     3.00
                                                          Mean   :     9.55
                                                          3rd Qu.:    10.00
                                                          Max.   : 80995.00

  InvoiceDate          UnitPrice           CustomerID       Country
 Length:541909      Min.   :-11062.06   Min.   :12346    Length:541909
 Class :character   1st Qu.:     1.25   1st Qu.:13953    Class :character
 Mode  :character   Median :     2.08   Median :15152    Mode  :character
                    Mean   :     4.61   Mean   :15288
                    3rd Qu.:     4.13   3rd Qu.:16791
                    Max.   : 38970.00   Max.   :18287
                                        NA's   :135080
   X
```

```
 Mode:logical
 NA's:541909




> dim(custData)
[1] 541909      9
>
> # Plot missing values using the DataExplorer package
> options(repr.plot.width=7, repr.plot.height=3)
> plot_missing(custData)
>
> # Drop missing values and check dimensions
> drop  <- c("X")
> custData = custData[!names(custData)%in%drop]
> custData <- na.omit(custData)
> dim(custData)
[1] 406829      8
> head(custData)
  InvoiceNo StockCode                          Description Quantity InvoiceDate
1    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER        6   29-Nov-16
2    536365     71053                  WHITE METAL LANTERN        6   29-Nov-16
3    536365    84406B       CREAM CUPID HEARTS COAT HANGER        8   29-Nov-16
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6   29-Nov-16
5    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.        6   29-Nov-16
6    536365     22752         SET 7 BABUSHKA NESTING BOXES        2   29-Nov-16
  UnitPrice CustomerID        Country
1      2.55      17850 United Kingdom
2      3.39      17850 United Kingdom
3      2.75      17850 United Kingdom
4      3.39      17850 United Kingdom
5      3.39      17850 United Kingdom
6      7.65      17850 United Kingdom
>
>
> # create date, month and year components of invoice date
> custData$date <- sapply(custData$InvoiceDate, FUN = function(x) {strsplit(x, split =
'[-]')[[1]][1]})
> custData$month <- sapply(custData$InvoiceDate, FUN = function(x) {strsplit(x, split =
'[-]')[[1]][2]})
> custData$year <- sapply(custData$InvoiceDate, FUN = function(x) {strsplit(x, split =
'[-]')[[1]][3]})
>
> # check the first three entries
> head(custData, n =3)
  InvoiceNo StockCode                          Description Quantity InvoiceDate
1    536365    85123A WHITE HANGING HEART T-LIGHT HOLDER        6   29-Nov-16
```

```
2    536365    71053              WHITE METAL LANTERN       6   29-Nov-16
3    536365    84406B     CREAM CUPID HEARTS COAT HANGER       8   29-Nov-16
  UnitPrice CustomerID       Country date month year
1      2.55      17850 United Kingdom   29   Nov   16
2      3.39      17850 United Kingdom   29   Nov   16
3      2.75      17850 United Kingdom   29   Nov   16
>
> # Convert date variable into datetime format
> custData$dateIndex <- as.Date(custData$InvoiceDate, format="%d-%b-%y")
> head(custData)
  InvoiceNo StockCode                       Description Quantity InvoiceDate
1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6   29-Nov-16
2    536365    71053              WHITE METAL LANTERN       6   29-Nov-16
3    536365    84406B     CREAM CUPID HEARTS COAT HANGER       8   29-Nov-16
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE       6   29-Nov-16
5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.       6   29-Nov-16
6    536365    22752        SET 7 BABUSHKA NESTING BOXES       2   29-Nov-16
  UnitPrice CustomerID       Country date month year  dateIndex
1      2.55      17850 United Kingdom   29   Nov   16 2016-11-29
2      3.39      17850 United Kingdom   29   Nov   16 2016-11-29
3      2.75      17850 United Kingdom   29   Nov   16 2016-11-29
4      3.39      17850 United Kingdom   29   Nov   16 2016-11-29
5      3.39      17850 United Kingdom   29   Nov   16 2016-11-29
6      7.65      17850 United Kingdom   29   Nov   16 2016-11-29
>
> # Create dayOfWeek variable
> custData <- custData %>% mutate(lineTotal = Quantity * UnitPrice)
> custData$dayOfWeek <- wday(custData$dateIndex, label=TRUE)
> head(custData)
  InvoiceNo StockCode                       Description Quantity InvoiceDate
1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6   29-Nov-16
2    536365    71053              WHITE METAL LANTERN       6   29-Nov-16
3    536365    84406B     CREAM CUPID HEARTS COAT HANGER       8   29-Nov-16
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE       6   29-Nov-16
5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.       6   29-Nov-16
6    536365    22752        SET 7 BABUSHKA NESTING BOXES       2   29-Nov-16
  UnitPrice CustomerID       Country date month year  dateIndex lineTotal
1      2.55      17850 United Kingdom   29   Nov   16 2016-11-29     15.30
2      3.39      17850 United Kingdom   29   Nov   16 2016-11-29     20.34
3      2.75      17850 United Kingdom   29   Nov   16 2016-11-29     22.00
4      3.39      17850 United Kingdom   29   Nov   16 2016-11-29     20.34
5      3.39      17850 United Kingdom   29   Nov   16 2016-11-29     20.34
6      7.65      17850 United Kingdom   29   Nov   16 2016-11-29     15.30
  dayOfWeek
1       Tue
2       Tue
3       Tue
4       Tue
5       Tue
```

```
6       Tue
>
> # Convert variables into factors
> custData$Country <- as.factor(custData$Country)
> custData$date <- as.factor(custData$date)
> custData$month <- as.factor(custData$month)
> custData$year <- as.factor(custData$year)
> range(custData$InvoiceDate)
[1] "1-Apr-17" "9-Sep-17"
> levels(custData$year) <- c(2016,2017)
> custData$dayOfWeek <- as.factor(custData$dayOfWeek)
>
> head(custData, n=5)
  InvoiceNo StockCode                         Description Quantity InvoiceDate
1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6   29-Nov-16
2    536365     71053                 WHITE METAL LANTERN        6   29-Nov-16
3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8   29-Nov-16
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6   29-Nov-16
5    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.       6   29-Nov-16
  UnitPrice CustomerID        Country date month year  dateIndex lineTotal
1      2.55      17850 United Kingdom   29   Nov 2016 2016-11-29     15.30
2      3.39      17850 United Kingdom   29   Nov 2016 2016-11-29     20.34
3      2.75      17850 United Kingdom   29   Nov 2016 2016-11-29     22.00
4      3.39      17850 United Kingdom   29   Nov 2016 2016-11-29     20.34
5      3.39      17850 United Kingdom   29   Nov 2016 2016-11-29     20.34
  dayOfWeek
1       Tue
2       Tue
3       Tue
4       Tue
5       Tue
>
> #------------------ Clustering ----------------------#
>
> # standardisation (or normalization)
> cust <- custData %>% select(UnitPrice, Quantity, lineTotal)
> head(cust)
  UnitPrice Quantity lineTotal
1      2.55        6     15.30
2      3.39        6     20.34
3      2.75        8     22.00
4      3.39        6     20.34
5      3.39        6     20.34
6      7.65        2     15.30
>
> fit<-hclust(custData, method='complete')
Error in if (is.na(n) || n > 65536L) stop("size cannot be NA nor exceed 65536") :
  missing value where TRUE/FALSE needed
> groups<-cutree(fit,k=3)
```

Log for eCommerce in R

by Ava Lee 2/14/2021

```
> groups
 [1] 1 2 3 1 1 2 2 1 2 2 2 1 1 3 2 3 2 3 1 1 3 1 1 3 2
>
> ## KMeans clustering
>
> km <- kmeans(custData, 3)
Error in do_one(nmeth) : NA/NaN/Inf in foreign function call (arg 1)
In addition: Warning message:
In storage.mode(x) <- "double" : NAs introduced by coercion
> head(km)
$cluster
 [1] 1 2 4 1 4 1 1 4 1 2 1 4 4 3 1 3 1 3 4 4 4 4 4 3 1

$centers
        SAT       Top10     Accept     SFRatio   Expenses    GradRate
1  0.7865158  0.66710444 -0.8831526 -0.42287980  0.4317231  0.77919531
2  0.8634201  0.56705021 -0.2382484 -1.52925136  2.3393604 -0.30029442
3 -1.8912923 -1.94145231  1.5612876  1.60546806 -1.2086753 -1.65272331
4 -0.1240313  0.06277688  0.2179720  0.04425486 -0.3729528  0.01987242

$totss
[1] 144

$withinss
[1]  5.323568  2.113429  7.089134 17.782037

$tot.withinss
[1] 32.30817

$betweenss
[1] 111.6918

>
> cust_kmeans<-data.frame(custData, km$cluster)
Error in data.frame(custData, km$cluster) :
  arguments imply differing number of rows: 406829, 25
> cust_kmeans
Error: object 'cust_kmeans' not found
> # cluster profiling
>
> #cluster profiling
> names(km)
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
>
> fviz_cluster(km, data = cust,
+              palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
+              geom = "point",
+              ellipse.type = "convex",
```

```
+             ggtheme = theme_bw()
+ )
Error in data.frame(..., check.names = FALSE) :
  arguments imply differing number of rows: 406829, 25
>
>
> #----------------------------Clustering End ----------------------------#
>
> # Plot revenue in over time
> options(repr.plot.width=8, repr.plot.height=3)
> custData %>%
+   group_by(dateIndex) %>%
+   summarise(revenue = sum(lineTotal)) %>%
+   ggplot(aes(x = dateIndex, y = revenue)) + geom_line() + geom_smooth(method = 'auto', se
= FALSE) + labs(x = 'Date', y = 'Revenue (£)', title = 'Revenue by Date')
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
> # Plot revenue by dayOfWeek
> custData %>%
+   group_by(dayOfWeek) %>%
+   summarise(revenue = sum(lineTotal)) %>%
+   ggplot(aes(x = dayOfWeek, y = revenue)) + geom_col() + labs(x = 'Day of Week', y =
'Revenue (£)', title = 'Revenue by Day of Week')
>
> # Transaction table by dayOfWeek
> weekdaySummary <- custData %>%
+   group_by(dateIndex, dayOfWeek) %>%
+   summarise(revenue = sum(lineTotal), transactions = n_distinct(InvoiceNo)) %>%
+   mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
+   ungroup()
`summarise()` has grouped output by 'dateIndex'. You can override using the `.groups` argument.
>
> head(weekdaySummary, n = 10)
# A tibble: 10 x 5
   dateIndex  dayOfWeek revenue transactions aveOrdVal
   <date>     <ord>       <dbl>        <int>     <dbl>
 1 2016-11-29 Tue       46051.         127      363.
 2 2016-11-30 Wed       45775.         160      286.
 3 2016-12-01 Thu       22598.          64      353.
 4 2016-12-03 Sat       31381.          94      334.
 5 2016-12-04 Sun       30465.         111      274.
 6 2016-12-05 Mon       53126.          79      672.
 7 2016-12-06 Tue       38049.         134      284.
 8 2016-12-07 Wed       37178.         132      282.
 9 2016-12-08 Thu       32005.          78      410.
10 2016-12-10 Sat       17218.          50      344.
>
> # Box plot of revenue by day of week
```

```
> ggplot(weekdaySummary, aes(x = dayOfWeek, y = revenue)) + geom_boxplot() + labs(x =
'Day of the Week', y = 'Revenue', title = 'Revenue by Day of the Week')
> ggplot(weekdaySummary, aes(x = dayOfWeek, y = transactions)) + geom_boxplot() + labs(x
= 'Day of the Week', y = 'Number of Daily Transactions', title = 'Number of Transactions by Day
of the Week')
> ggplot(weekdaySummary, aes(x = dayOfWeek, y = aveOrdVal)) + geom_boxplot() + labs(x
= 'Day of the Week', y = 'Average Order Value', title = 'Average Order Value by Day of the
Week')
>
> # Density plot of transactions by day of week
> ggplot(weekdaySummary, aes(transactions, fill = dayOfWeek)) + geom_density(alpha = 0.2)
>
> # Apply non-parametric test for distributions other than normal based on skewness of the
density plot
> kruskal.test(transactions ~ dayOfWeek, data = weekdaySummary)

        Kruskal-Wallis rank sum test

data:  transactions by dayOfWeek
Kruskal-Wallis chi-squared = 71.744, df = 5, p-value = 4.441e-14

> # Discover which day of a week incurs significantly higher or lower revenues
> kruskal(weekdaySummary$transactions, weekdaySummary$dayOfWeek, console = TRUE)

Study: weekdaySummary$transactions ~ weekdaySummary$dayOfWeek
Kruskal-Wallis test's
Ties or no Ties

Critical Value: 71.7443
Degrees of freedom: 5
Pvalue Chisq  : 4.440892e-14

weekdaySummary$dayOfWeek,  means of the ranks

    weekdaySummary.transactions  r
Mon                   160.0769 52
Sat                    72.3600 50
Sun                   162.4574 47
Thu                   135.0100 50
Tue                   170.2170 53
Wed                    213.5000 53

Post Hoc Analysis

t-Student: 1.96793
Alpha    : 0.05
Groups according to probability of treatment differences and alpha level.

Treatments with the same letter are not significantly different.
```

```
   weekdaySummary$transactions groups
Wed                 213.5000      a
Tue                 170.2170      b
Sun                 162.4574     bc
Mon                 160.0769     bc
Thu                 135.0100      c
Sat                  72.3600      d
>
>
> # Valued Customer Analysis by Month
> custData %>%
+   group_by(month) %>%
+   summarise(revenue = sum(lineTotal)) %>%
+   ggplot(aes(x = month, y = revenue)) + geom_col() + labs(x = 'Month', y = 'Revenue (£)', title
= 'Revenue by Month Of Year')
>
> custData %>%
+   group_by(month) %>%
+   summarise(transactions = n_distinct(InvoiceNo)) %>%
+   ggplot(aes(x = month, y = transactions)) + geom_col() + labs(x = 'Month', y = 'Number of
Transactions', title = 'Transactions by Month Of Year')
>
> # Most valuable customers by country
> countrySummary <- custData %>%
+   group_by(Country) %>%
+   summarise(revenue = sum(lineTotal), transactions = n_distinct(InvoiceNo)) %>%
+   mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
+   ungroup() %>%
+   arrange(desc(revenue))
>
> head(countrySummary, n = 10)
# A tibble: 10 x 4
   Country        revenue transactions aveOrdVal
   <fct>            <dbl>        <int>     <dbl>
 1 United Kingdom 6767873.        19857      341.
 2 Netherlands     284662.          101     2818.
 3 EIRE            250285.          319      785.
 4 Germany         221698.          603      368.
 5 France          196713.          458      430.
 6 Australia       137077.           69     1987.
 7 Switzerland      55739.           71      785.
 8 Spain            54775.          105      522.
 9 Belgium          40911.          119      344.
10 Sweden           36596.           46      796.
> unique(countrySummary$Country)
 [1] United Kingdom       Netherlands          EIRE
 [4] Germany              France               Australia
 [7] Switzerland          Spain                Belgium
```

```
[10] Sweden               Japan                Norway
[13] Portugal             Finland              Channel Islands
[16] Denmark              Italy                Cyprus
[19] Austria              Singapore            Poland
[22] Israel               Greece               Iceland
[25] Canada               Unspecified          Malta
[28] United Arab Emirates USA                  Lebanon
[31] Lithuania            European Community    Brazil
[34] RSA                  Czech Republic       Bahrain
[37] Saudi Arabia
37 Levels: Australia Austria Bahrain Belgium Brazil Canada ... USA
>
> countryCustSummary <- custData %>%
+   group_by(Country) %>%
+   summarise(revenue = sum(lineTotal), customers = n_distinct(CustomerID)) %>%
+   mutate(aveCustVal = (round((revenue / customers),2))) %>%
+   ungroup() %>%
+   arrange(desc(revenue))
>
> head(countryCustSummary, n = 10)
# A tibble: 10 x 4
   Country        revenue customers aveCustVal
   <fct>            <dbl>     <int>      <dbl>
 1 United Kingdom 6767873.      3950      1713.
 2 Netherlands     284662.         9     31629.
 3 EIRE            250285.         3     83428.
 4 Germany         221698.        95      2334.
 5 France          196713.        87      2261.
 6 Australia       137077.         9     15231.
 7 Switzerland      55739.        21      2654.
 8 Spain            54775.       31      1767.
 9 Belgium          40911.       25      1636.
10 Sweden           36596.        8      4574.
>
> # Top five most valued countries
> topFiveCountries <- custData %>%
+   filter(Country == 'United Kingdom' | Country == 'Netherlands' | Country == 'EIRE' | Country
== 'Germany' | Country == 'France')
>
> topFiveCountrySummary <- topFiveCountries %>%
+   group_by(Country, dateIndex) %>%
+   summarise(revenue = sum(lineTotal), transactions = n_distinct(InvoiceNo), customers =
n_distinct(CustomerID)) %>%
+   mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
+   ungroup() %>%
+   arrange(desc(revenue))
`summarise()` has grouped output by 'Country'. You can override using the `.groups` argument.
>
> head(topFiveCountrySummary)
```

Log for eCommerce in R

by Ava Lee 2/14/2021

```
# A tibble: 6 x 6
  Country        dateIndex  revenue transactions customers aveOrdVal
  <fct>          <date>       <dbl>        <int>     <int>     <dbl>
1 United Kingdom 2017-09-18 100460.          63        49     1595.
2 United Kingdom 2017-09-13  65611.          99        85      663.
3 United Kingdom 2017-10-01  59782.          70        50      854.
4 United Kingdom 2017-11-01  56295.         100        85      563.
5 United Kingdom 2017-11-07  55634.         118        97      471.
6 United Kingdom 2017-11-21  55013.         130       102      423.
>
> ggplot(topFiveCountrySummary, aes(x = Country, y = revenue)) + geom_col() + labs(x = '
Country', y = 'Revenue (£)', title = 'Revenue by Country')
> ggplot(topFiveCountrySummary, aes(x = dateIndex, y = revenue, colour = Country)) +
geom_smooth(method = 'auto', se = FALSE) + labs(x = ' Country', y = 'Revenue (£)', title =
'Revenue by Country over Time')
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> ggplot(topFiveCountrySummary, aes(x = Country, y = aveOrdVal)) + geom_boxplot() + labs(x
= ' Country', y = 'Average Order Value (£)', title = 'Average Order Value by Country') +
scale_y_log10()
Warning messages:
1: In self$trans$transform(x) : NaNs produced
2: Transformation introduced infinite values in continuous y-axis
3: Removed 73 rows containing non-finite values (stat_boxplot).
> ggplot(topFiveCountrySummary, aes(x = Country, y = transactions)) + geom_boxplot() +
labs(x = ' Country', y = 'Transactions', title = 'Number of Daily Transactions by Country')
>
> # Segmentation
> custSummary <- custData %>%
+   group_by(CustomerID) %>%
+   summarise(revenue = sum(lineTotal), transactions = n_distinct(InvoiceNo)) %>%
+   mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
+   ungroup() %>%
+   arrange(desc(revenue))
>
> head(custSummary, n = 10)
# A tibble: 10 x 4
   CustomerID revenue transactions aveOrdVal
        <int>   <dbl>        <int>     <dbl>
 1      14646 279489.           77     3630.
 2      18102 256438.           62     4136.
 3      17450 187482.           55     3409.
 4      14911 132573.          248      535.
 5      12415 123725.           26     4759.
 6      14156 113384.           66     1718.
 7      17511  88125.           46     1916.
 8      16684  65892.           31     2126.
 9      13694  62653.           60     1044.
10      15311  59419.          118      504.
>
```

Log for eCommerce in R

by Ava Lee 2/14/2021

```
> # Revenue per customer
> ggplot(custSummary, aes(revenue)) + geom_histogram(binwidth = 10) + labs(x = 'Revenue', y
= 'Count of Customers', title = 'Histogram of Revenue per customer')
> ggplot(custSummary, aes(revenue)) + geom_histogram() + scale_x_log10() + labs(x =
'Revenue', y = 'Count of Customers', title = 'Histogram of Revenue per customer (Log Scale)')
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
Warning messages:
1: In self$trans$transform(x) : NaNs produced
2: Transformation introduced infinite values in continuous x-axis
3: Removed 55 rows containing non-finite values (stat_bin).
> ggplot(custSummary, aes(transactions)) + geom_histogram() + scale_x_log10() + labs(x =
'Number of Transactions', y = 'Count of Customers', title = 'Histogram of Transactions per
customer')
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
>
> # Group B
> custSummaryB <- custData %>%
+   group_by(CustomerID, InvoiceNo) %>%
+   summarise(revenue = sum(lineTotal), transactions = n_distinct(InvoiceNo)) %>%
+   mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
+   ungroup() %>%
+   arrange(revenue) %>%
+   mutate(cumsum=cumsum(revenue))
`summarise()` has grouped output by 'CustomerID'. You can override using the `.groups` argument.
>
> head(custSummaryB, n =10)
# A tibble: 10 x 6
   CustomerID InvoiceNo  revenue transactions aveOrdVal   cumsum
        <int> <chr>        <dbl>        <int>     <dbl>    <dbl>
 1      16446 C581484   -168470.            1 -168470. -168470.
 2      12346 C541433    -77184.            1  -77184. -245653.
 3      15098 C556445    -38970             1  -38970  -284623.
 4      15749 C550456    -22998.            1  -22998. -307622.
 5      16029 C570556    -11817.            1  -11817. -319438.
 6      12536 C573079     -8322.            1   -8322. -327760.
 7      16029 C551685     -8143.            1   -8143. -335903.
 8      16029 C551699     -6930             1   -6930  -342833.
 9      12744 C571750     -6068.            1   -6068. -348901.
10      14911 C562375     -4345.            1   -4345. -353246.
>
> custData %>% filter(CustomerID == 16446)
  InvoiceNo StockCode                Description Quantity InvoiceDate
1    553573     22980      PANTRY SCRUBBING BRUSH        1   16-May-17
2    553573     22982        PANTRY PASTRY BRUSH         1   16-May-17
3    581483     23843 PAPER CRAFT , LITTLE BIRDIE    80995    7-Dec-17
4   C581484     23843 PAPER CRAFT , LITTLE BIRDIE   -80995    7-Dec-17
  UnitPrice CustomerID        Country date month year  dateIndex  lineTotal
1      1.65      16446 United Kingdom   16   May 2017 2017-05-16       1.65
2      1.25      16446 United Kingdom   16   May 2017 2017-05-16       1.25
```

```
3      2.08      16446 United Kingdom    7   Dec 2017 2017-12-07  168469.60
4      2.08      16446 United Kingdom    7   Dec 2017 2017-12-07 -168469.60
  dayOfWeek
1       Tue
2       Tue
3       Thu
4       Thu
>
> custSummaryB <- custData %>%
+   group_by(InvoiceNo, CustomerID, Country, dateIndex, month, year, dayOfWeek) %>%
+   summarise(orderVal = sum(lineTotal)) %>%
+   mutate(recent = Sys.Date() - dateIndex) %>%
+   ungroup()
`summarise()` has grouped output by 'InvoiceNo', 'CustomerID', 'Country', 'dateIndex', 'month',
'year'. You can override using the `.groups` argument.
>
> custSummaryB$recent <- as.character(custSummaryB$recent)
> custSummaryB$recentDays <- sapply(custSummaryB$recent, FUN = function(x) {strsplit(x, split
= '[-]')[[1]][1]})
> custSummaryB$recentDays <- as.integer(custSummaryB$recentDays)
>
> head(custSummaryB, n = 5)
# A tibble: 5 x 10
  InvoiceNo CustomerID Country dateIndex  month year  dayOfWeek orderVal recent
  <chr>          <int> <fct>   <date>     <fct> <fct> <ord>        <dbl> <chr>
1 536365         17850 United… 2016-11-29 Nov   2016  Tue          139.  1538
2 536366         17850 United… 2016-11-29 Nov   2016  Tue           22.2 1538
3 536367         13047 United… 2016-11-29 Nov   2016  Tue          279.  1538
4 536368         13047 United… 2016-11-29 Nov   2016  Tue           70.1 1538
5 536369         13047 United… 2016-11-29 Nov   2016  Tue           17.8 1538
# … with 1 more variable: recentDays <int>
>
> # Customer Breakdown
> customerBreakdown <- custSummaryB %>%
+   group_by(CustomerID, Country) %>%
+   summarise(orders = n_distinct(InvoiceNo), revenue = sum(orderVal), meanRevenue =
round(mean(orderVal), 2), medianRevenue = median(orderVal),
+             mostDay = names(which.max(table(dayOfWeek))), mostMonth =
names(which.max(table(month))),
+             recency = min(recentDays))%>%
+   ungroup()
`summarise()` has grouped output by 'CustomerID'. You can override using the `.groups` argument.
>
> head(customerBreakdown)
# A tibble: 6 x 9
  CustomerID Country orders revenue meanRevenue medianRevenue mostDay mostMonth
       <int> <fct>    <int>   <dbl>       <dbl>         <dbl> <chr>   <chr>
1      12346 United…      2      0           0             0  Mon     Jan
2      12347 Iceland      7   4310         616.          585. Mon     Dec
```

```
3      12348 Finland      4   1797.       449.        338. Mon     Apr
4      12349 Italy        1   1758.      1758.       1758. Sun     Nov
5      12350 Norway       1    334.       334.        334. Tue     Jan
6      12352 Norway      11   1545.       140.        160. Mon     Feb
# … with 1 more variable: recency <int>
>
> custBreakSum <- customerBreakdown %>%
+   filter(orders > 1, revenue > 50)
>
> head(custBreakSum)
# A tibble: 6 x 9
  CustomerID Country orders revenue meanRevenue medianRevenue mostDay mostMonth
       <int> <fct>    <int>   <dbl>       <dbl>         <dbl> <chr>   <chr>
1      12347 Iceland      7   4310        616.          585. Mon     Dec
2      12348 Finland      4   1797.       449.          338. Mon     Apr
3      12352 Norway      11   1545.       140.          160. Mon     Feb
4      12356 Portug…      3   2811.       937.          481. Mon     Apr
5      12358 Austria      2   1168.       584.          584. Mon     Dec
6      12359 Cyprus       6   6246.      1041.          828. Tue     Apr
# … with 1 more variable: recency <int>
> dim(custBreakSum)
[1] 3032    9
>
> # Heatmap
>
> custMat <- custBreakSum %>%
+   select(recency, revenue, meanRevenue, medianRevenue, orders) %>%
+   as.matrix()
>
> rownames(custMat) <- custBreakSum$CustomerID
>
> head(custMat)
      recency revenue meanRevenue medianRevenue orders
12347    1167 4310.00      615.71        584.91      7
12348    1240 1797.24      449.31        338.50      4
12352    1201 1545.41      140.49        160.33     11
12356    1187 2811.43      937.14        481.46      3
12358    1166 1168.06      584.03        584.03      2
12359    1172 6245.53     1040.92        828.41      6
> class(custMat)
[1] "matrix" "array"
>
> options(repr.plot.width=12, repr.plot.height=7)
> heatmap(scale(custMat), cexCol = 0.7)
>
>
> # Reference
> # https://www.kaggle.com/chrisbow/e-commerce-eda-and-segmentation-with-r
```

> # https://www.datanovia.com/en/blog/k-means-clustering-visualization-in-r-step-by-step-guide/