Mining drugs that are taken together using the openFDA API

This function calls the adverse events endpoint of the openFDA API to retrieve reports submitted between 01-01-2016 and 01-06-20117. There is a limit of 100 reports you can retrieve per query. More reports are extracted by using the skip parameter.

```
get_data <- function(k){

  api_url <- "https://api.fda.gov/drug/event.json?search=receivedate:[20160101+TO+20170601]&limit=100"
  data <- jsonlite::fromJSON(api_url, simplifyDataFrame = T)
  data <- data$results
  data <- data$patient
  data <- data$drug

  data <- sapply(data,function(x) x$medicinalproduct)
  all_data <- data

  for(i in 1:k){
    api_url <- paste("https://api.fda.gov/drug/event.json?search=receivedate:[20160101+TO+20170601]&lim
    data <- jsonlite::fromJSON(api_url, simplifyDataFrame = T)
    data <- data$results
    data <- data$patient
    data <- data$drug

    data <- sapply(data,function(x) x$medicinalproduct)
    all_data <- c(all_data,data)
  }
  return(all_data)
}
```

Get the get the first 100+99*100 reports between 01-01-2016 and 01-06-20117 and clean the data (drug names).

```
all_reports <- get_data(99)
#remove dots
all_reports <- sapply(all_reports, function(x) gsub("\\.","",x))
#remove curly brackets
all_reports <- sapply(all_reports, function(x) gsub("\\{","",x))
```

The idea is to try to find associations among drugs, i.e. which drugs tend to be taken together, by using the same method applied in Market Basket Analysis - Association Rules.

We will use the arules package for the analysis and on top of that the arulesViz package for visualizations.

We make the following assumption: An adverse event report (ae report) is one transaction and the drugs listed in the report are the purchased items. An itemset then is a set of one or more drugs.

Association measures:

1. Support. Measures the proportion of ae reports in which an itemset (containing one or more drugs) appears.
2. Confidence. Measures how likely drug B is taken when drug A is taken, Confidence (A->B) = Support(A,B)/Support(A).Caveat with confidence: it only accounts for how popular drug A is. If drug

B is also very popular in the dataset, there will be a higher chance that an ae report containing drug A will also contain drug B, thus inflating the confidence.

3. Lift, measures how likely drug B is taken when drug A is taken while accounting for how popular drug B is in the dataset. Lift measures how many times more often drug A and B are taken together than expected if they were statistically independent. A lift value of 1 indicates independence between drug A and B.

Load the retrieved data from the API in the appropriate format.

```
require(arules)

#convert the list to class transactions to use with the arules package
meds_list <- as(all_reports, "transactions")
```

Start by defining the minimum support in the dataset. Check the summary of the "transactions"

```
summ_meds_list <- summary((meds_list))
```

We see that the most frequent item(medicine) is ENBREL and occurs 624 times in the dataset. It means that this particular medicine has a support of 0.0624. For the purposes of this brief exploratory analysis we want to consider medicines that appear at least 5 times in these 10000 reports, therefore we set the minimum support to $5/10000 = 0.0005$. We also set the minimum confidence at 0.5 and run the apriori algorithm to obtain the association rules for the dataset.

```
#maxlen defines the maximum number or items in a rule and minlen the minimum. minlen is set to 2 to avo
rules <- apriori(meds_list,parameter = list(supp=0.0005, conf = 0.5, maxlen = 20, minlen = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5   5e-04      2
##  maxlen target   ext
##      20  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 5
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[5200 item(s), 10000 transaction(s)] done [0.01s].
## sorting and recoding items ... [1028 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 11 done [0.01s].
## writing ... [23603 rule(s)] done [0.01s].
## creating S4 object  ... done [0.01s].
```

We get 23603 rules. We can remove the redundant rules. A rule is redundant if a more general rule with the same or a higher confidence exists. A rule is more general if it has the same RHS but one or more items removed from the LHS.

```
#remove redundant rules,
rules <- rules[!is.redundant(rules, measure = "confidence")]
```

Sorting the remaining rules by their lift, we can inspect the top 10 rules with the highest lift. We focus on lift here, because the higher the lift gets the lower the probability that the relationship between the two itemsets

is a coincidence.

```
#sort rules by lift in decreasing order
sorted_rules <- sort(rules, by = "lift", decreasing = T)

sorted_lhs <- as(lhs(sorted_rules), "list")
sorted_rhs <- as(rhs(sorted_rules), "list")
```

```
inspect(head(sorted_rules, n=10))
```

```
##      lhs                                      rhs            support confidence     lift count
## [1]  {FERRIPROX}                           => {DEFEROXAMINE}   6e-04      1.000 1666.667     6
## [2]  {DEFEROXAMINE}                        => {FERRIPROX}      6e-04      1.000 1666.667     6
## [3]  {EMTRICITABINE,
##       RISPERIDONE}                         => {TENOVIR}        6e-04      1.000 1666.667     6
## [4]  {RISPERIDONE,
##       RITONAVIR}                           => {TENOVIR}        5e-04      1.000 1666.667     5
## [5]  {EXVIERA}                             => {VIEKIRAX}       7e-04      1.000 1250.000     7
## [6]  {VIEKIRAX}                            => {EXVIERA}        7e-04      0.875 1250.000     7
## [7]  {LAMIVUDINE,
##       NEVIRAPINE}                          => {ZIDOVUDINE}     5e-04      1.000 1250.000     5
## [8]  {PROCARBAZINE,
##       SULFAMETHOXAZOLE-TRIMETHOPRIM (SMZ)} => {G-CSF}          8e-04      1.000 1250.000     8
## [9]  {BLEOMYCIN,
##       SULFAMETHOXAZOLE-TRIMETHOPRIM (SMZ)} => {G-CSF}          8e-04      1.000 1250.000     8
## [10] {ETOPOSIDE,
##       SULFAMETHOXAZOLE-TRIMETHOPRIM (SMZ)} => {G-CSF}          8e-04      1.000 1250.000     8
```

From the rules we can observe some patterns such as:

If someone took DEFEROXAMINE is likely to have taken FERRIPROX as well.

OR

If someone takes ZIDOVUDINE is likely to have taken LAMIVUDINE, NEVIRAPINE as well.

These patterns emerge from rules with high confidence AND lift.

Visualisations

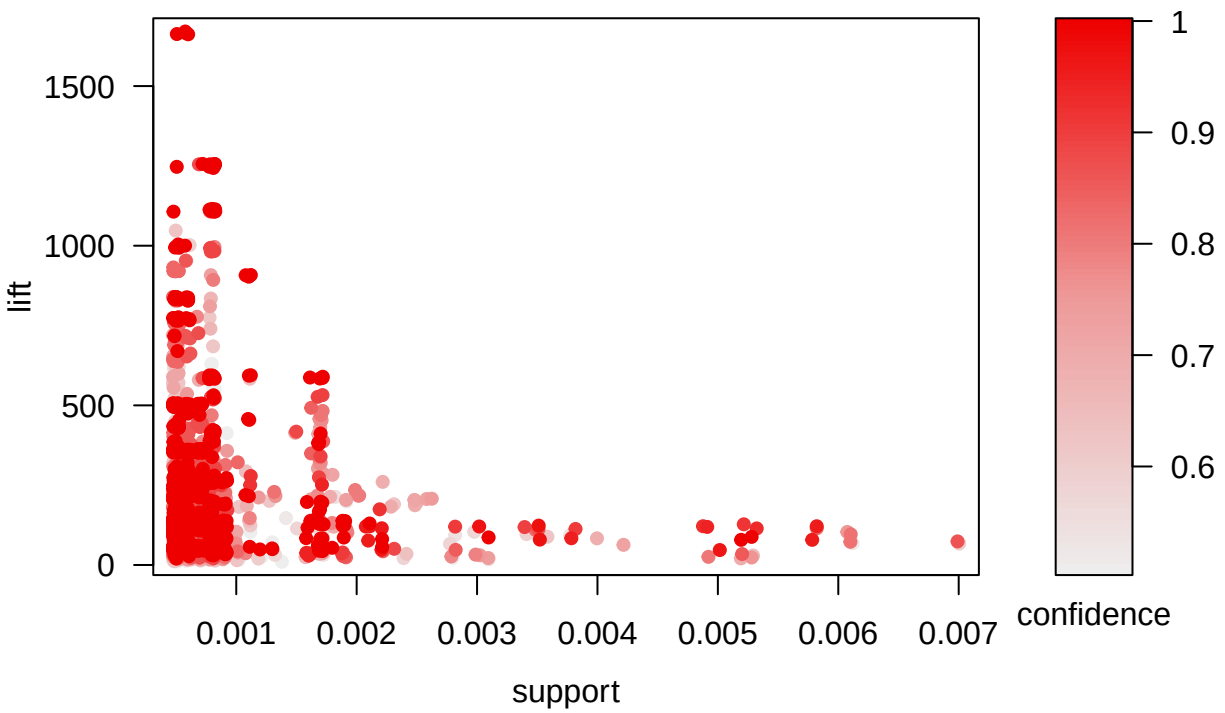
Using the arulesViz package we can quickly produce some visualisations.

To get an overview of the rules we can use a scatter plot with two interest measures on the axes, in our case, lift(on y) and support(on x). Moreover, a third measure (confidence) is used as the color of the points.

In the following plot we can see that rules with high lift have typically a relatively low support, as expected. This plot can also be interactive, i.e. by hovering over the points in the plot we see the underlying rules.

```
require(arulesViz)
require(igraph)

plot(rules, measure=c("support", "lift"), shading="confidence")
```



Graph based visualisations can also be utilised, albeit for a small portion of the extracted rules. Here, rules are represented as nodes. Node size is proportional to the rule's support, while the darker the node is colored the higher its lift. In the following graph we can observe the same 10 rules as before. LHS itemsets are incoming edges in vertices, while RHS are outgoing.

```
rules_to_plot <- head(sorted_rules,10)
graph <- plot(rules_to_plot, method = "graph", control = list(cex = .6, alpha = 0.9, igraph::layout_nic
```

```
## Warning: Unknown control parameters:
```

```
## Available control parameters (with default values):
## main  =  Graph for 10 rules
## nodeColors    =  c("#66CC66E6", "#9999CCE6")
## nodeCol   =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE121
## edgeCol   =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#53535
## alpha     =  0.5
## cex   =  1
## itemLabels    =  TRUE
## labelCol  =  #000000B3
## measureLabels     =  FALSE
## precision     =  3
## layout    =  NULL
## layoutParams  =  list()
## arrowSize     =  0.5
## engine    =  igraph
## plot  =  TRUE
## plot_options  =  list()
## max   =  100
## verbose   =  FALSE
```

Graph for 10 rules

size: support (5e−04 − 8e−04)
color: lift (1250 − 1666.6666666667)