**Topics of exercises**

* Frequent itemsets, association rules

**Textbooks and readings**

Pang-Ning Tan, Michael Steinbach, Vipin Kumar: Introduction to Data Mining. Pearson Addison Wesley, 2006, chapter 6, <https://www-users.cs.umn.edu/~kumar001/dmbook/index.php>

**Exercise 5.1**

Consider the market basket transactions shown in the table.

| **TID** | **Items** |
| --- | --- |
| 1 | {Milk, Beer, Diapers} |
| 2 | {Bread, Butter, Milk} |
| 3 | {Milk, Diapers, Cookies} |
| 4 | {Bread, Butter, Cookies} |
| 5 | {Beer, Cookies, Diapers} |
| 6 | {Milk, Diapers, Bread, Butter} |
| 7 | {Bread, Butter, Diapers} |
| 8 | {Beer, Diapers} |
| 9 | {Milk, Diapers, Bread, Butter} |
| 10 | {Beer, Cookies} |

1. What is the maximum number of association rules that can be extracted from this data (including rules that have zero support)?
2. What is the maximum size of frequent itemsets that can be extracted (assuming minsup > 0)?
3. Find an itemset (of size 2 or larger) that has the largest support.

**Exercise 5.2**

Consider the short table with the market basket transactions (in a numeric representations of items) from some chosen places in the world.

| **TID** | **Items** |
| --- | --- |
| Puebla-El Parian | {1, 3, 4} |
| Paris-Galleries Lafayette | {2, 3, 5} |
| Moscow-Rublovka | {1, 2, 3, 5} |
| New York-5th Avenue Gallery | {2, 5} |

1. Exploit the **Apriori** Algorithm in order to generate the possible associations rules (assume the min support threshold = 2). (How many of them you can detect? Why?)
2. Detect the rules withh the same support and with the same conﬁdence. How many values of the confidence can you distinguish?
3. What about the confidence of 1-elemental candidate item sets? (Can you generalize your conclusions?)

**Exercise 5.3**

Consider the following dataset with 10 transactions over 5 total items.

| **TID** | **Items** |
| --- | --- |
| 1 | {a, b} |
| 2 | {b, c, d} |
| 3 | {a, c, d, e} |
| 4 | {a, d, e} |
| 5 | {a, b, c} |
| 6 | {a, b, c, d} |
| 7 | {a} |
| 8 | {a, b, c} |
| 9 | {a, b, d} |
| 10 | {b, c, e} |

1. Create its corresponding FP-tree.
2. Detect the most frequent items.

**Exercise 5.4**

Consider the market basket transactions shown in the table.

| **TID** | **Items** |
| --- | --- |
| 1 | {a, b, d, e} |
| 2 | {b, c, d} |
| 3 | {a, b, d, e} |
| 4 | {a, c, d, e} |
| 5 | {b, c, d, e} |
| 6 | {b, d, e} |
| 7 | {c, d} |
| 8 | {a, b, c} |
| 9 | {a, d, e} |
| 10 | {b, d} |

1. Build FP-tree for these data.
2. Find frequent itemsets ending in e.
3. Find frequent itemsets ending in b.

**Exercise 5.5**

1. Download and unpack the [titanic.raw.txt.zip](http://home.agh.edu.pl/~mszpyrka/lib/exe/fetch.php?media=lectures:dm:titanic.raw.txt.zip) file.
2. Load the file as **titanic\_raw** data frame. Display a few record using the **head** function.
3. Use **Apriori** algorithm to find association rules of length at least 2, support equal to 0.25 and confidence 0.5. Use the **inspect** function to display the rules. Sort the rules by **lift**. Find the rule with the highest lift.
4. Use **Apriori** algorithm to find association rules of length at least 2, support equal to 0.1, confidence 0.3. Moreover, include the following argument appearance = list(rhs = c(“Survived=No”, “Survived=Yes”), default = “lhs”). Use the **inspect** function to display the rules.

**Exercise 5.6**

1. Download and unpack the [lotto.txt.zip](http://home.agh.edu.pl/~mszpyrka/lib/exe/fetch.php?media=lectures:dm:lotto.txt.zip) file.
2. Load the file as **lotto** data frame.
3. Convert the data frame into list of vectors **lotto.list**. Each vector must contain 6 numbers represented by strings. **Remark:** Use a double **for** statement.
4. Convert the list into set of transactions: trs ← as(lotto.list, “transactions”)
5. Use the **Apriori** algorithm to calculate how many times each value has been drawn. **Remark:** Find all 1-itemsets.
6. Use the **Apriori** algorithm to find the frequent 2-itemsets and 3-itemsets.

**Exercise 5.7**

1. Download and unpack the [groceries.txt.zip](http://home.agh.edu.pl/~mszpyrka/lib/exe/fetch.php?media=lectures:dm:groceries.txt.zip) file.
2. Load the file as **groceries** data frame.
3. Find all frequent itemsets (support at least 1%) that
   * contain *whole milk* **or** *yogurt*
   * contain *whole milk* **and** *yogurt*
   * contain **only** *whole milk* **or** *yogurt*
4. Find all maximal frequent itemsets (support at least 1%) that contain *tropical fruit*
5. Find all closed frequent itemsets (support at least 1%) that contain *tropical fruit*
6. Find all association rules (support at least 1%, confidence at least 25%) that
   * **rhs** contains *whole milk* **or** *yogurt*
   * **lhs** contains *whole milk* **or** *yogurt*
   * **lhs** contains *whole milk* **and** *yogurt*
   * **lhs** contains **only** *whole milk* **or** *yogurt*

**Remark:** Use the **subset** function and operators: %in%, %ain%, %oin% (see **arules.pdf** manual)

**Exercise 5.8**

Consider the set of transactions defined as follows (*Each letter is an item.*):

POLAND

AGH

OLD

CITY

CASTLE

DRAGON

VISTULA

WINTER

1. Build FP-tree for these data using minimum support count = 2.
2. Use the FP-tree and find all frequent itemsets.

**Exercise 5.9**

1. Download and unpack the [40-transactions.txt.zip](http://home.agh.edu.pl/~mszpyrka/lib/exe/fetch.php?media=lectures:dm:40-transactions.txt.zip) file.
2. Load the file as **trs** data frame and transform as follows:
3. trs <- as(trs, "matrix")

trs <- as(trs, "transactions")

1. Use **apriori** method to find all frequent itemsets (support = 0.01).
2. Compute all-confidence measure for these itemsets. **Remark:** Use the **interestMeasure** function.
3. Find cross-support itemsets (h\_c = 0.14).

**Exercise 5.10**

**The Three Oddest Words**

by Wisława Szymborska (Nobel Prize 1996)

(English translation)

When I pronounce the word Future,

the first syllable already belongs to the past.

When I pronounce the word Silence,

I destroy it.

When I pronounce the word Nothing,

I make something no non-being can hold

1. Skip punctuation marks, join lines and assign the poem as a single string to a variable. Transform all capital letters two small ones.
2. Transform the variable to a matrix of transactions, such that each word is treated as a transaction that contains set of letters. **Remark:** The as(…, “transactions”) function will omit repeated letters.
3. Use the **Apriori** algorithm to find the frequent 2-itemsets and 3-itemsets of different letters.