Mining Association Rules From MusicBrainz

Ivo Jimenez   Vassilis Polychronopoulos

June 12, 2012

Abstract

MusicBrainz is a publicly available relational database that stores information about artists, releases, tracks and the relationship among them. We present the results of mining association rules from this dataset, with the aim of obtaining knowledge about artists and their work. We are able to obtain associations between features, such as native language, and quantify how likely it is for an artist to do the "cross-over" to another language; or to associate the volume of work an artist produces with respect to a specific music label or genre. The number of possible association rules is virtually infinite so we limit the scope of this work to a few meaningful ones. Nonetheless, there is plenty of room for future work on this.

Introduction

MusicBrainz is a publicly available relational database that stores information about artists, releases, tracks and the relationship among them. MusicBrainz is used by portable MP3 players and desktop multi-media software to show metadata of a particular song to the user, while it is being played. Thus, the typical use for the MusicBrainz dataset is operational, i.e. simple OLTP queries over it used to display an artist's info (eg. as in Rhythmbox); or to incorporate new information into the dataset (eg. Picard, the official MusicBrainz client).

To the best of our knowledge, there hasn't been any work on mining the MusicBrainz dataset. In this report, we present the results of mining association rules from this dataset, with the aim of obtaining knowledge about artists and their work. For example, we are able to obtain correlations between features, such as the native language of an artist, and how likely it is for her to do the "cross-over" to other language (eg. English). Another example is to associate the volume of work an artist produces with respect to a specific music label or genre. The number of possible association rules is virtually infinite so we limit the scope of this work to a few meaningful ones. Nonetheless, we include a list of associations that might be interesting to mine.

The report is organized as follows. We give a high level introduction of Association Rule mining. We then present the schematic structure of the MusicBrainz database in section. We then explain the query and transformation of the data necessary to perform mining using R, and finally we explain the motivation behind our experiments and present our results. The appendices contain detailed plots and lists of the results which are mentioned high-level and commented in the main body of the report.

Frequent Patterns and Association Rules

Identifying the frequent itemsets from a dataset and deriving association rules is a very useful method for knowledge discovery in a dataset that can be used in different domains. Association rules can reveal implicit connections between items of a dataset and can help identify useful patterns for behavior prediction and decision making. Strong association rules are the ones with high confidence and positive correlation. Various measures of interest that can be used to analyze and present strong rules have been described by Shapiro and Fawley [10]. To quote Agrawal et al. [1] the problem of mining association rules from a dataset is defined as follows:

> Let $I = i_1, i_2, \ldots, i_n$ be a set of n binary attributes called items. Let $D = t_1, t_2, \ldots, t_m$ be a set of transactions called the database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of the form $X \to Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$.

The standard nomenclature for the set of items X is the *antecedent* items or left-hand-side or LHS and for the Y itemset is *consequent* or right-hand-side or RHS of the rule.

In lectures and literature there is a standard example that is used to clarify the concepts, using the transactions made in a supermarket. Having a database that contains transactions, where each transaction is a tuple that contains the items bought, examples of transactions are $(1, milk, bread)$, $(2, bread, butter)$, $(3, milk, bread, butter)$, $(4, bread)$. An example of an association rule is $\{milk, butter\} \to \{bread\}$ which would indicate that with a high confidence a customer that buys milk and butter is also buying bread. A reference containing examples and conceptual information is [7].

To restrict our set of rules to a small one that contains meaningful and strong associations we use various measures of interest and significance. The most widely used ones are the minimum threshold on support and confidence. The support of an itemset, denoted $supp(X)$ is the proportion of the transactions in the entire dataset that contain all elements of the itemset. For example, itemset $X = \{item_1, item_2\}$ has support ${}^2/_5 = 0.6$ if it occurs in 60% of all transactions (3 out of 5 transactions).
Another interesting measure of correlation is the *lift*, defined as $\frac{supp(X \cup Y)}{supp(X) supp(Y)}$ essentially saying how many times more itemsets X and Y are found together than in the case where they were statistically independent. A lift that is smaller than 1 indicates a negative correlation, while the higher the lift of an association rule $X \to Y$ the stronger the association is.

The problem that needs to be solved in order to derive association rules is determining frequent datasets. In a database with a large number of transactions and items, performing a brute-search approach to determine the frequent itemsets would explode exponentially and is prohibitively costly. There have been significant contributions in the field in the past two decades. A benchmark that compares the currently fastest algorithms for computing the frequent itemsets was made by Goethals and Zaki [5] . Two of the algorithms tested are the Apriori and Eclat algorithms by Borgelt [3]. Apriori algorithm [2] uses a bread-first approach that counts transactions increasing the itemset length and pruning the search tree to decrease the size of the search space. Eclat [11] uses a different strategy involving equivalence classes and set intersection rather than counting. The algorithms can be used to mine frequent itemsets, maximal frequent itemsets and closed frequent itemsets. Apriori can also be used to generate the association rules. FP-growth algorithm [8] is capable of computing the frequent itemsets without generating the candidate sets avoiding that costly procedure.

In this work, we use the Apriori algorithm implemented in R [4] by the `arules` package [6].

MusicBrainz

The database

Musicbrainz is a project that aims to create an open content database containing music metadata. It is called a project rather than a database, because it is constantly updated and enriched with new features and additions to the schema. Also, the documentation for the database and the platform on which it runs is still incomplete. Belonging to the public domain, it is available to anyone for download. For our experiments, we downloaded a virtual machine containing an already configured PostgreSQL server with the database, performing queries by remotely connecting to the virtual machine. Some technical issues arose on configuring the connection with the virtual machine and other configuration settings of the virtual machine platform, but, overall, the setting up of the database did not pose significant technical challenges.

Schema general description

In this section we give a quick description of the schematic organization of the MusicBrainz database. The database is laid out in a star-schema fashion. Figure 1 corresponds to the UML-based diagram of the schema. There are 10 *fact* tables corresponding to the main entities of the database. Refer to [9] for more information.

Artist

An artist is generally a musician, group of musicians, a collaboration of multiple musicians or other music professional.

Artist credit

List of artists, variations of artist names and pieces of text to join the artist names. Examples:

- "Queen & David Bowie" – two artists ("Queen" and "David Bowie"), no name variations, joined with " & ".
- "Jean-Michel Jarre" – one artist ("Jean Michel Jarre"), name variation "Jean-Michel Jarre"
- "Tracy W. Bush Derek Duke Jason Hayes and Glenn Stafford" – four artists, no name variations, joined with commas and an "and".

Release group

Represents an abstract "album" entity. Technically it's a group of releases, with a specified type. Examples:

- Single "Under Pressure" by "Queen & David Bowie"
- Album "The Wall" by "Pink Floyd"

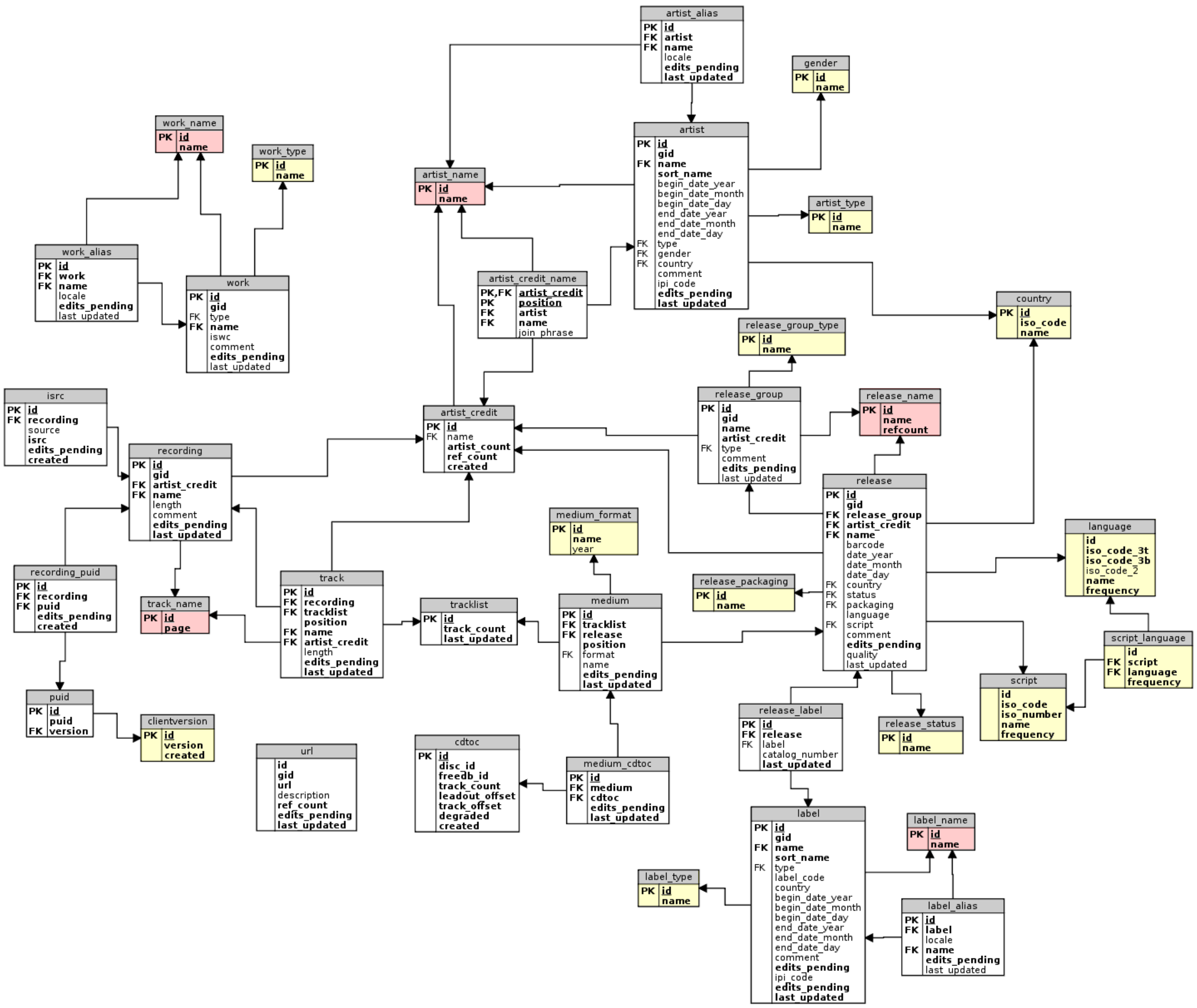

Figure 1: Relational schema of the MusicBrainz database

Release

Real-world release object you can buy in your music store. It has release date and country, list of catalog number and label pairs, packaging type and release status. Examples:

- 1984 US release of "The Wall" by "Pink Floyd", release on label "Columbia Records" with catalog number "C2K 36183" and UPC "074643618328", it's an official release and comes with two CDs in jewel case.

Medium

Piece of media, included in a release. Contains information about the format, position in the release and an optional title. Has attached CD TOCs. Examples:

- CD1 of the 1984 US release of "The Wall" by "Pink Floyd"
- CD2 of the 2005 UK release of "Aerial" by "Kate Bush", named "A Sky of Honey"

Tracklist

A tracklist is an ordered list of tracks that is linked to one or more mediums. Examples:

- Tracklist 703797 appears on release The Wall (medium 1/2)
- Tracklist 1085862 appears on releases Violet Cries (digital media) and Violet Cries (vinyl)

Track

This object is not visible to users on its own, only in the context of a tracklist. It contains a link to a recording, title, artist credit and its position on its tracklist.

Recording

Represents unique audio data. Has title, artist credit, duration, list of PUIDs and ISRCs Examples (all are different Recordings):

- Album version of the track "*Into the Blue*" by "*Moby*"
- Remix "*Into the Blue (Buzz Boys Main Room Mayhem mix)*" by "*Moby*"
- Remix "*Into the Blue (Underground mix)*" by "*Moby*"

Work

One layer above recordings ("song", "composition", etc.). While recording represents audio data, work represents the composition behind the recording. Advanced Relationships should be used to link recording and work.

- Song "*Into the Blue*" by "*Moby*" – all the recordings listed above will be linked to this object

Label

Labels represent mostly imprints.

Advanced relationships

For all the combination of major entities (artist, work, release etc.) there exist relations that attempt to grasp advanced relationships among them. There is a naming convention for each one of them, so advanced relation *l_artist_release* contains information about relationships between artists and recordings, *l_artist_label* contains information about relationships between artists and labels and likewise for the other entities. Each advanced relation contains attributes *entity0* and *entity1* containing the ids of the corresponding entities and an attribute called link, which joined with relations *Links* and *Links_name* can determine the kind of connection that the two entities have. Thus, in the case of an artist and a label, the link may indicate a recording contract or the fact that the artist is the founder of that label. For an artist and a release, the link may indicate that the artist did vocals, or played an instrument or was involved in mixing. So there is a number of possible relationship types among entities and the kind of link established between them is the main information that each tuple of this special relation contains.

Data preparation

Transactions can be represented as sparse matrices. We need to extend the notion of transaction to circumstances that go beyond the common use of the term, that is, in supermarkets and shops. Assuming we have a one-to-many mapping from an artist to the various roles he/she has played in her carreer (an artist may have served as a singer, lyricist, composer, producer, guitarist in the same or in different periods of his/her career) we can consider a transaction having the form (singer,lyricist, composer, producer, guitarist). Likewise, the fact that an artist has made recording contracts with various labels in the music industry, say EMI, Warner Bros and Sony can be thought of as a transaction (EMI,Warner Bros, Sony) in a transaction table that would contain that information for all artists. Thus, our task is to thoroughly explore the schema of the database, perform the right queries, and transform the query results into transaction tables suitable for mining by the apriori algorithm. The format of a transaction table on which rule extraction algorithms run is the following (see Table 1 or an alternative format in Table 2): it contains two columns, one identifying a transaction and another one with the item set that corresponds to each transaction. In its initial form the result of the query is likely to contain the same artist or other entity in the left column, similar to the format of Table 2, so the corresponding transaction is a list of all the elements in the right column where the same element is present in the left column. This kind of representation needs to first be transformed into a sparse array and then to a structure of the form "transaction" suitable for use by the methods in the *arules* R package.

ID	ITEM
1	{milk, bread}
2	{bread, butter}
3	{milk, bread, butter}
4	{bread}

Table 1: *Traditional* way of representing a transaction table

ID	ITEM
1	{milk}
1	{bread}
2	{bread}
2	{butter}
3	{milk}
3	{bread}
3	{butter}
4	{bread}

Table 2: An alternative way of representing a transaction table

Given that the MusicBrainz schema is starred, we have to prepare the data so that can work on the right format. For example, for a rule associating record companies, say $\{Columbia\} \rightarrow \{Sony\}$ (see next section for the rationale behind this association and for other meaningful ones), we have to get a transaction table whose schema is *Transaction(Artist, Label)*, that is, the id of an artist represents the id of a transaction and the id of the label corresponds to the item. The query we need to address to the musicbrainz database to get that result back would need to join 4 relations as we can see from the schema details of the database. The query is the following:
SELECT distinct entity0, ln.name
FROM l_artist_label as w,link as l, label as lb, label_name as ln, link_type as t
WHERE w.link=l.id and l.link_type=t.id and
t.name='recording contract' and entity1=lb.id and lb.id=ln.id

It is essential to thoroughly understand the database schema and make the right queries to the database to obtain the data that is suitable for association rule mining. Having transaction data in the form of Table 2, we need to mofidy them appropriately so that they can be made suitable for use with the apriori function of the arules package in R. An R script that would do this would need to first turn the table into a sparse array and then transform it into the form that can be used as input to the apriori function. The following script loads the required libraries, connects to the library, posts the query and gets back results and performs the necessary transformation to the query results:

```
# load libraries
> library(RJDBC)

# load the JDBC driver
> drv <- JDBC("org.postgresql.Driver", "pg.jar")

# create a connection
> conn <- dbConnect(drv, "jdbc:postgresql://localhost/musicbrainz_db")

# get data from the database
```

```
> data <- dbGetQuery(
           conn,
           "SELECT distinct entity0, ln.name
           FROM l_artist_label as w,link as l, label as lb, label_name as ln, link_type as t
           WHERE w.link=l.id and l.link_type=t.id and
            t.name='recording contract' and entity1=lb.id and lb.id=ln.id")

# get items and IDs:
#    ID   = artist
#    item = label

> id   <- data[, 1]
> item <- data[, 2]
```

We use the `RJDBC` library [@urbanek_rjdbc_2011] to manage all the connectivity to the underlying DBMS. The script first loads the driver of the DBMS (PostgreSQL [@postgresql_global_development_group_postgresql_2010]), creates a connection and then executes a query over the `l_artist_label`, which is the view described above. Consult [@musicbrainz_contributors_musicbrainz_2012] for more about the advanced views and relationships.

In the following section we describe the four rounds of experiments we performed attempting to mine various association rules for different query results from the database. and our results for various of them. One of them did not produce good results, the other 3 produced a number of strong association rules.

MusicBrainz Associations

Which rules would be interesting to mine from the MusicBrainz dataset? The number of possible association rules is virtually infinite so we limit the scope of this work to a few meaningful ones. We first define the set of interesting rules we explored as part of this project, as well as the rationale behind them. We then present the results.

Interesting Associations

Experiment 1:Rules for contracts with different record companies

One interesting association is related to how often an artist migrates from one record company to another and whether there exist a label for which is very likely that an artist leaves it. This type of information can be used by companies, since it might be an indication of how good is the public image of it, or even reflect that something is not doing well internally. A company can also use that information to identify its adversaries. The SQL query used is the one we used in the example above Thus, for this association, we're interested in finding which (and how many) labels imply having more than one label. We can get a high-level idea of this by doing a quick analysis on the data. The `arules` package has many features that allow the user to explore and visualize the data which we used to preprocess the transaction tables. We also used the summary command to get information about the transaction table. Applying the itemFrequencyplot command we had an image image of the frequent itemsets in the database which can be found in the appendix. We then used the apriori function to do the mining. The results were not as expected:

```
> rules <- apriori(trans, parameter = list(support = 0.0015, confidence = 0.6))
..
```

```
..
writing ... [0 rule(s)] done [0.00s].
```

Lowering the support to extremely low levels, close to zero, we had thousands of meaningless association rules whose support corresponds to exactly one transaction and therefore cannot be considered rules. Assessing the results, we concluded that there is most likely two principal reasons that there are no such rules. First, the mobility of the artists may be very low, and artists probably stick to the company that they had their first contract with, and second, even when there are rules reported for very few transactions, they may refer to the same company under alias name. Unfortunately, though the database contains a relation with label aliases, information was very sparse, and we could not exploit it in some meaningful way.

Experiment 2: Roles of artists in different releases

A different type of association rules we mined is the association between the roles an artist has played in his career across different releases. Somebody may have served as a vocalist and a guitarist and later as a producer or mixer, so it would be interesting to see whether there are association rules there. We need to perform the following query to get the data needed:
SELECT distinct entity0, t.name
FROM l_artist_release as w,link as l,link_type as t
WHERE w.link=l.id and l.link_type=t.id

using the advanced view l_artist_release. We plotted the most frequent roles which can be seen in the plot in the appendix. We got a number of 18 rules, all with high lift, so there was no need to be pruned any further. The support was chosen at 0.005, which is suitable for the relative frequencies present in the dataset, and the confidence was at 0.6. Results can be seen in the appendix and they are quite interesting. So, we can deduce for example by rule 11 that a person who has participated in the mix of a recording and has composed some recording has with high confidence (>90%) produced some recording.

Experiment 3: Roles of artists in different works

As indicated above in the description of the schema, the relation *work* is one level above recordings and releases. So, while a relase and a recording represent audio data, a work represents the composition behind the recording. We mined association rules for artists and work in the same way as before. The frequent itemsets are fewer this time, as can be seen from the frequency plot in the appendix (composer, lyricist, writer etc.). Mining produced 2 rules. Overall, the results are less interesting than the previous experiment. Results can be found in the appendix.

Experiment 4: Associations among languages used by artists

The final experiment examined the association among languages that an artist has released into. The value *multiple languages* in the language attribute of a release indicates that the release contains more than one languages, which are unspecified. The SQL query to obtain the results to be further processed for mining is the following:
SELECT distinct entity0, l.name
FROM l_artist_release as ar,release as r, language as l
WHERE entity1=r.id and r.language=l.id
We performed the mining and got a very big number of association rules, many of them with negative correlation (lift <1), so we used the subset command to obtain only the rules with lift >1.5, essentially pruning the rules that have the English language to the right side. English is ubiquitous, it has a very

big support, and therefore the rules having it at the right side have low lift in general and do not provide much information. Doing that we get a number of rules. We plot them using the group setting to get a visualization of the strongest rules as can be seen at the plot in the appendix.

Conclusion

It has been a fruitful round of experiments that mine association rules from musicbrainz. The work allowed us to master the details of the musicbrainz database schema and profoundly understand the problem of association rules and their mining.
Musicbrainz lacks a number of information, such as popularity and genre, which could make our experiments even more interesting by examining associations among different music genres and or use classification techniques trying to predict popularity.
In general, future work could use information on works of music from different sources and combine them with the information in musicbrainz to apply and test various classification techniques that would classify artists, works, tracks and labels, building the training data from the external source.

References

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. page 487–499, 1994.

[3] C. Borgelt. Efficient implementations of apriori and eclat. page 90, 2003.

[4] M. J. Crawley. *The R Book*. John Wiley & Sons, June 2007.

[5] B. Goethals and M. J. Zaki. Advances in frequent itemset mining implementations: report on FIMI'03. *SIGKDD Explor. Newsl.*, 6(1):109–117, June 2004.

[6] M. Hahsler, B. Grün, and K. Hornik. Introduction to arules: Mining association rules and frequent item sets. *SIGKDD EXPLORATIONS*, 2:0–4, 2007.

[7] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, Third Edition*, volume 1. Morgan Kaufmann, Jan. 2001.

[8] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[9] MusicBrainz contributors. MusicBrainz Database/Schema. http://wiki.musicbrainz.org/MusicBrainz_Database/Schema, 2012.

[10] G. Piatetsky-Shapiro, G. Piatetsky-Shapiro, and W. Frawley. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.

[11] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. Technical report, University of Rochester, Rochester, NY, USA, 1997.

Appendix

Experiment 1: Failed

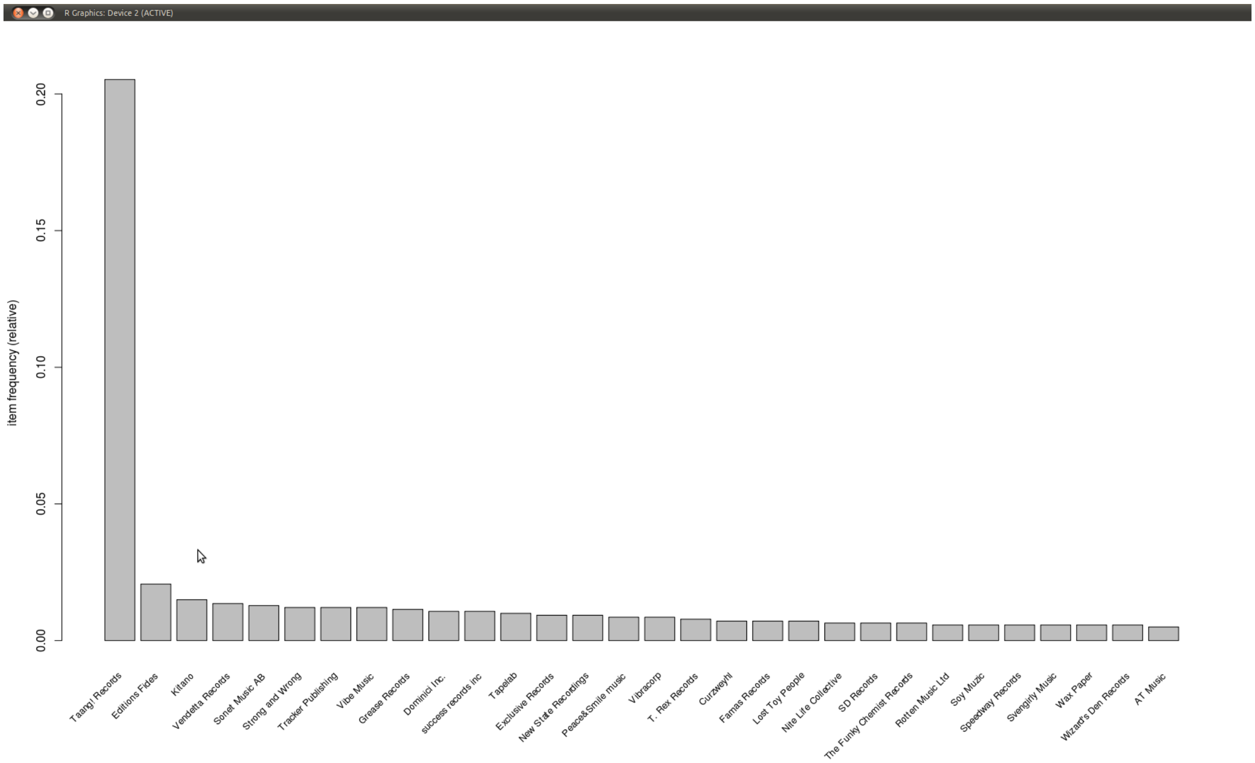

Figure 2: Frequency plot for most frequent items in experiment 1

```
itemMatrix in sparse format with
 1403 rows (elements/transactions) and
 641 columns (items)
```

```
> rules <- apriori(trans, parameter = list(support = 0.0015, confidence = 0.6))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
        0.6    0.1    1 none FALSE            TRUE  0.0015      1     10  rules
   ext
 FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

```
apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[641 item(s), 1403 transaction(s)] done [0.00s].
sorting and recoding items ... [105 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [0 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s]
```

Experiment 2

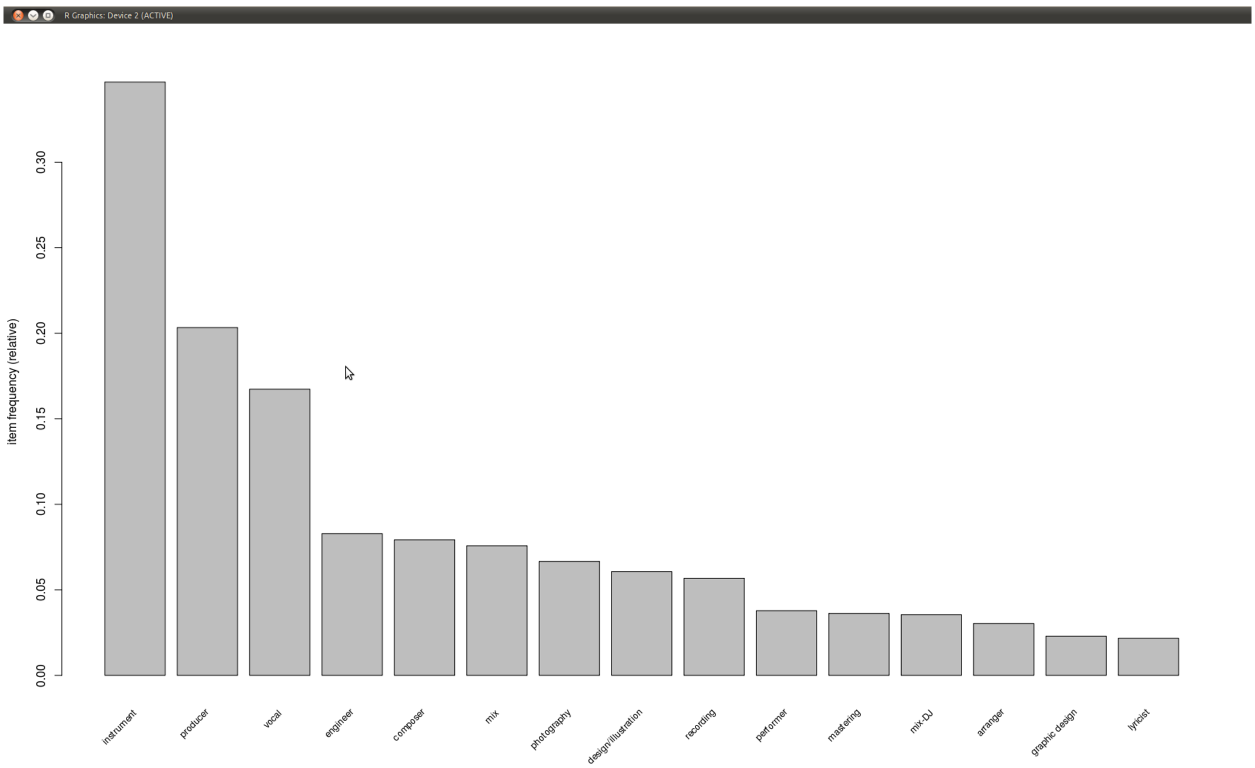

Figure 3: Frequency plot for most frequent items in experiment 2

```
transactions as itemMatrix in sparse format with
 66702 rows (elements/itemsets/transactions) and
 37 columns (items) and a density of 0.03972651

most frequent items:
instrument   producer      vocal   engineer   composer    (Other)
     23135      13561      11159       5524       5282      39383

element (itemset/transaction) length distribution:
sizes
```

```
    1     2     3     4     5     6     7     8     9    10    11    12    13
48615 10971  3878  1730   790   375   178    87    34    17    16     7     2
   14    15
    1     1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.00    1.00    1.47    2.00   15.00

includes extended item information - examples:
         labels
1      arranger
2 art direction
3         audio

includes extended transaction information - examples:
  transactionID
1             1
2             4
3             9
> itemFrequencyPlot(trans, topN = 20, cex.names = 0.8)

> rules <- apriori(trans, parameter = list(support = 0.005, confidence = 0.6))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
        0.6    0.1    1 none FALSE            TRUE   0.005      1     10  rules
   ext
 FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[37 item(s), 66702 transaction(s)] done [0.01s].
sorting and recoding items ... [26 item(s)] done [0.01s].
creating transaction tree ... done [0.02s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [18 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
> inspect(rules)
   lhs             rhs              support confidence      lift
1  {lyricist,
    producer}   => {composer}   0.005067314  0.7647059  9.656837
2  {mastering,
    recording}  => {mix}        0.005577044  0.8416290 11.116502
3  {mastering,
    producer}   => {mix}        0.005502084  0.6427320  8.489409
4  {arranger,
```

```
    composer}   => {producer}   0.007630956  0.7654135  3.764812
5  {arranger,
    mix}        => {producer}   0.006266679  0.9106754  4.479306
6  {arranger,
    instrument} => {producer}   0.006881353  0.6219512  3.059169
7  {engineer,
    recording}  => {mix}        0.010044676  0.6399236  8.452314
8  {producer,
    recording}  => {mix}        0.013837666  0.6771827  8.944443
9  {instrument,
    recording}  => {producer}   0.005786933  0.6509275  3.201693
10 {engineer,
    instrument} => {producer}   0.006446583  0.6677019  3.284201
11 {composer,
    mix}        => {producer}   0.009594915  0.9155937  4.503498
12 {composer,
    vocal}      => {instrument} 0.007645948  0.6938776  2.000563
13 {mix,
    vocal}      => {producer}   0.005696981  0.8102345  3.985271
14 {mix,
    vocal}      => {instrument} 0.005547060  0.7889126  2.274564
15 {instrument,
    mix}        => {producer}   0.010749303  0.7515723  3.696732
16 {producer,
    vocal}      => {instrument} 0.015351864  0.7447273  2.147171
17 {engineer,
    mix,
    recording}  => {producer}   0.006026806  0.6000000  2.951198
18 {engineer,
    producer,
    recording}  => {mix}        0.006026806  0.8072289 10.662135
```

Experiment 3

```
transactions as itemMatrix in sparse format with
 48564 rows (elements/itemsets/transactions) and
 11 columns (items) and a density of 0.113539

most frequent items:
    composer     lyricist       writer orchestrator   librettist      (Other)
       36392        17724         5015          480          344          698

element (itemset/transaction) length distribution:
sizes
    1     2     3     4     5
37651  9797  1060    52     4

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

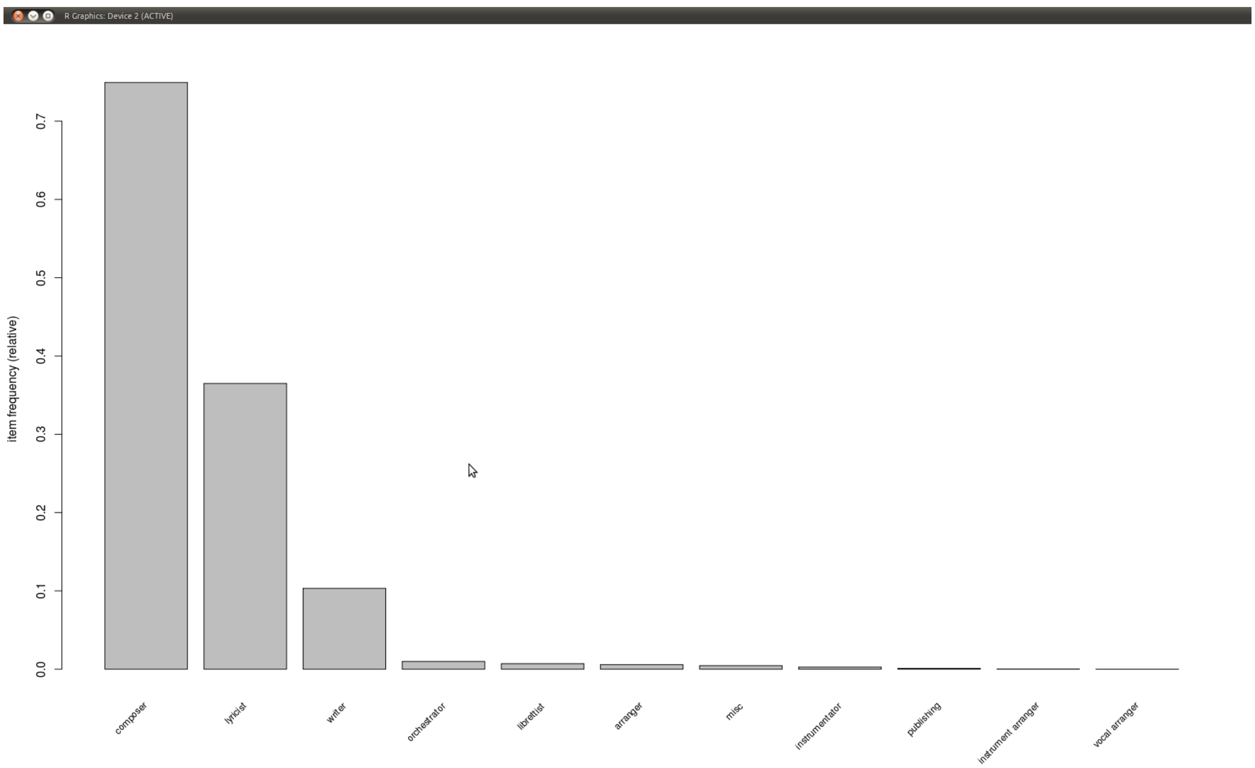

Figure 4: Frequency plot for most frequent items in experiment 3

```
  1.000   1.000   1.000   1.249   1.000   5.000

includes extended item information - examples:
               labels
1            arranger
2            composer
3 instrument arranger

includes extended transaction information - examples:
  transactionID
1            10
2            15
3            16
> itemFrequencyPlot(trans, topN = 20, cex.names = 0.8)
> rules <- apriori(trans, parameter = list(support = 0.001, confidence = 0.6))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
        0.6    0.1    1 none FALSE            TRUE   0.001      1     10  rules
   ext
 FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[11 item(s), 48564 transaction(s)] done [0.01s].
sorting and recoding items ... [9 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [3 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
> inspect(rules)
  lhs               rhs            support confidence     lift
1 {}             => {composer} 0.749361667  0.7493617 1.000000
2 {lyricist,
   orchestrator} => {composer} 0.001235483  0.9523810 1.270923
3 {lyricist,
   writer}       => {composer} 0.019335310  0.8376450 1.117811
```

Experiment 4

```
> summary(trans)
transactions as itemMatrix in sparse format with
 66251 rows (elements/itemsets/transactions) and
 89 columns (items) and a density of 0.0122632
```

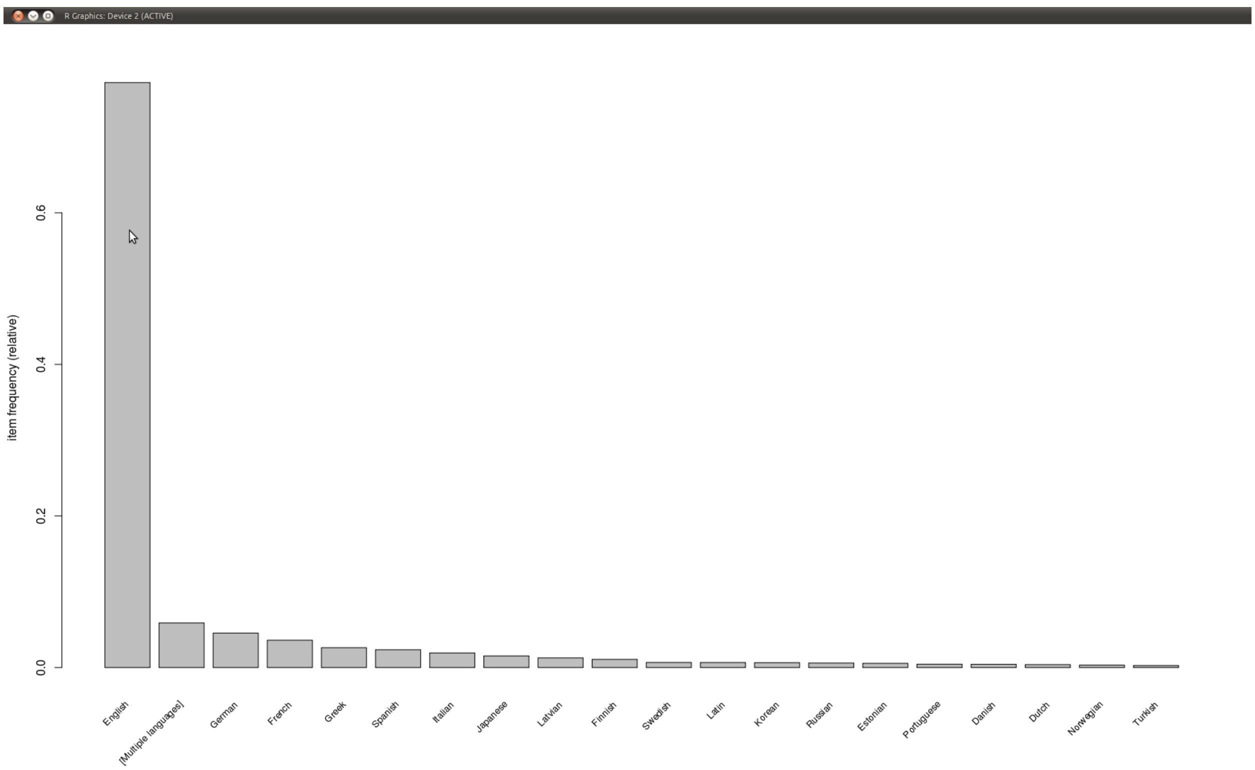

Figure 5: Frequency plot for most frequent languages

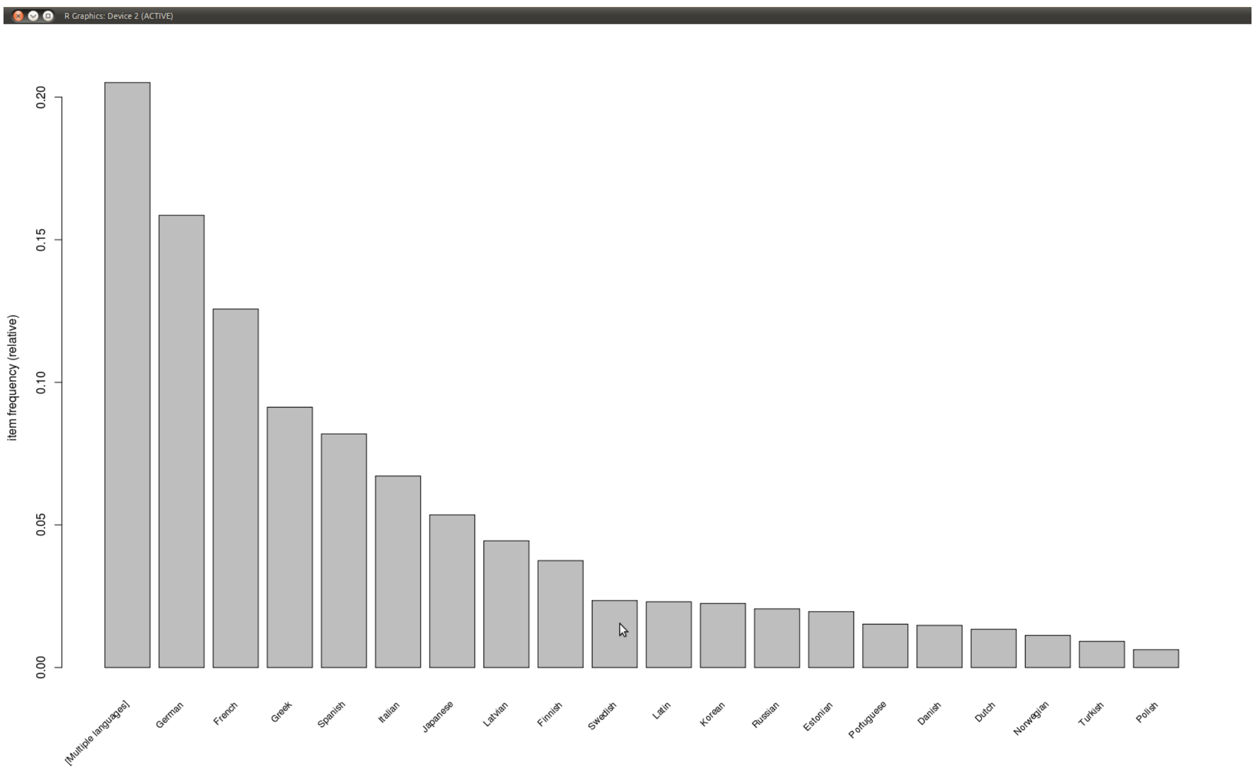

Figure 6: Frequency plot for most frequent languages excluding english

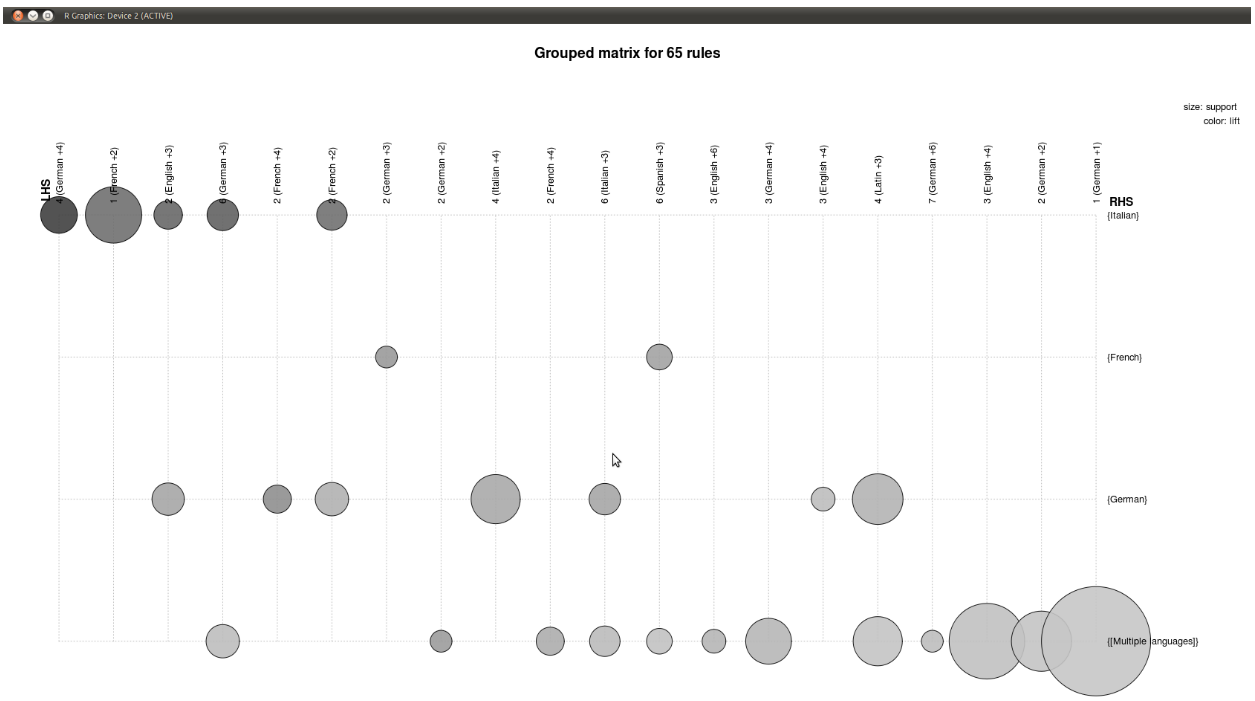

Figure 7: Grouped plot for association rules with languages

```
most frequent items:
             English [Multiple languages]               German
               51136                 3896                 3012
              French                Greek              (Other)
                2388                 1734                10142

element (itemset/transaction) length distribution:
sizes
    1     2     3     4     5     6     7     8     9    11
61691  3553   681   223    66    22    10     3     1     1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   1.000   1.091   1.000  11.000

includes extended item information - examples:
     labels
1 Afrikaans
2      Akan
3   Amharic

includes extended transaction information - examples:
  transactionID
1             1
2             4
3             9

> rules <- apriori(trans, parameter = list(support = 0.0001, confidence = 0.5))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target
        0.5    0.1    1 none FALSE            TRUE   1e-04      1     10  rules
   ext
 FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[89 item(s), 66251 transaction(s)] done [0.00s].
sorting and recoding items ... [58 item(s)] done [0.01s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.00s].
writing ... [136 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
> plot(rules, method="grouped")
> inspect(rules)
    lhs                        rhs                          support confidence       lift
```

1	{}	=>	{English}	0.7718525003	0.7718525	1.0000000
2	{Malay}	=>	{English}	0.0001056588	0.5384615	0.6976223
3	{Sami, Northern}	=>	{English}	0.0001509411	0.9090909	1.1778039
4	{Czech, Italian}	=>	{English}	0.0001509411	0.9090909	1.1778039
5	{Czech, German}	=>	{English}	0.0001056588	1.0000000	1.2955843
6	{Czech, [Multiple languages]}	=>	{English}	0.0001509411	1.0000000	1.2955843
7	{German, Norwegian}	=>	{[Multiple languages]}	0.0001056588	1.0000000	17.0048768
8	{German, Norwegian}	=>	{English}	0.0001056588	1.0000000	1.2955843
9	{[Multiple languages], Norwegian}	=>	{English}	0.0002113176	0.7000000	0.9069090
10	{Dutch, German}	=>	{English}	0.0001358470	0.5625000	0.7287662
11	{Dutch, [Multiple languages]}	=>	{English}	0.0001660352	0.7333333	0.9500952
12	{Italian, Portuguese}	=>	{English}	0.0001056588	0.8750000	1.1336363
13	{French, Portuguese}	=>	{English}	0.0002113176	0.9333333	1.2092120
14	{German, Portuguese}	=>	{English}	0.0001056588	0.7777778	1.0076767
15	{[Multiple languages], Portuguese}	=>	{English}	0.0003773528	0.6944444	0.8997113
16	{Estonian, Finnish}	=>	{English}	0.0001207529	0.8888889	1.1516305
17	{Latvian, Russian}	=>	{English}	0.0003018822	0.5263158	0.6818865
18	{Latin, Russian}	=>	{English}	0.0001056588	1.0000000	1.2955843
19	{French, Russian}	=>	{English}	0.0001207529	0.6666667	0.8637229
20	{German, Russian}	=>	{[Multiple languages]}	0.0001358470	0.5625000	9.5652432
21	{German, Russian}	=>	{English}	0.0001811293	0.7500000	0.9716882
22	{[Multiple languages], Russian}	=>	{English}	0.0002716940	0.6000000	0.7773506
23	{Spanish, Swedish}	=>	{English}	0.0001056588	1.0000000	1.2955843
24	{French, Swedish}	=>	{English}	0.0001207529	0.8888889	1.1516305
25	{German, Swedish}	=>	{[Multiple languages]}	0.0001056588	0.6363636	10.8212852
26	{German, Swedish}	=>	{English}	0.0001660352	1.0000000	1.2955843
27	{[Multiple languages], Swedish}	=>	{English}	0.0002867881	0.7307692	0.9467732

```
28  {Finnish,
     [Multiple languages]} => {English}              0.0002716940  0.6923077  0.8969430
29  {Latvian,
     [Multiple languages]} => {English}              0.0003169763  0.5121951  0.6635920
30  {Latin,
     Spanish}              => {[Multiple languages]} 0.0001207529  0.6153846 10.4645396
31  {Latin,
     Spanish}              => {English}              0.0001660352  0.8461538  1.0962637
32  {Italian,
     Latin}                => {German}               0.0006943291  0.5974026 13.1402787
33  {Italian,
     Latin}                => {[Multiple languages]} 0.0006792350  0.5844156  9.9379150
34  {Italian,
     Latin}                => {English}              0.0009660232  0.8311688  1.0768493
35  {French,
     Latin}                => {English}              0.0007697997  0.7611940  0.9861911
36  {German,
     Latin}                => {English}              0.0012528113  0.8469388  1.0972806
37  {Latin,
     [Multiple languages]} => {English}              0.0014188465  0.7966102  1.0320756
38  {French,
     Japanese}             => {English}              0.0001056588  0.8750000  1.1336363
39  {German,
     Japanese}             => {[Multiple languages]} 0.0001056588  0.5833333  9.9195115
40  {German,
     Japanese}             => {English}              0.0001660352  0.9166667  1.1876190
41  {Japanese,
     [Multiple languages]} => {English}              0.0004528234  0.5555556  0.7197691
42  {Italian,
     Spanish}              => {English}              0.0002113176  0.7368421  0.9546411
43  {German,
     Spanish}              => {French}               0.0001660352  0.5500000 15.2588149
44  {French,
     Spanish}              => {English}              0.0004981057  0.7674419  0.9942856
45  {German,
     Spanish}              => {[Multiple languages]} 0.0001660352  0.5500000  9.3526822
46  {German,
     Spanish}              => {English}              0.0002867881  0.9500000  1.2308051
47  {[Multiple languages],
     Spanish}              => {English}              0.0008452703  0.5137615  0.6656213
48  {French,
     Italian}              => {English}              0.0016150700  0.7133333  0.9241835
49  {German,
     Italian}              => {[Multiple languages]} 0.0014792230  0.5240642  8.9116467
50  {German,
     Italian}              => {English}              0.0022792109  0.8074866  1.0461670
51  {Italian,
     [Multiple languages]} => {English}              0.0027320342  0.8116592  1.0515729
52  {French,
     German}               => {[Multiple languages]} 0.0012829995  0.5592105  9.5093061
53  {French,
```

```
     German}               => {English}               0.0019169522  0.8355263  1.0824948
54  {French,
     [Multiple languages]} => {English}               0.0030188224  0.7380074  0.9561508
55  {German,
     [Multiple languages]} => {English}               0.0042565395  0.7401575  0.9589364
56  {German,
     [Multiple languages],
     Norwegian}            => {English}               0.0001056588  1.0000000  1.2955843
57  {English,
     German,
     Norwegian}            => {[Multiple languages]} 0.0001056588  1.0000000 17.0048768
58  {English,
     [Multiple languages],
     Norwegian}            => {German}                0.0001056588  0.5000000 10.9978420
59  {German,
     [Multiple languages],
     Russian}              => {English}               0.0001358470  1.0000000  1.2955843
60  {English,
     German,
     Russian}              => {[Multiple languages]} 0.0001358470  0.7500000 12.7536576
61  {English,
     [Multiple languages],
     Russian}              => {German}                0.0001358470  0.5000000 10.9978420
62  {German,
     [Multiple languages],
     Swedish}              => {English}               0.0001056588  1.0000000  1.2955843
63  {English,
     German,
     Swedish}              => {[Multiple languages]} 0.0001056588  0.6363636 10.8212852
64  {Latin,
     [Multiple languages],
     Spanish}              => {English}               0.0001207529  1.0000000  1.2955843
65  {English,
     Latin,
     Spanish}              => {[Multiple languages]} 0.0001207529  0.7272727 12.3671831
66  {French,
     Italian,
     Latin}                => {German}                0.0002565999  0.6800000 14.9570651
67  {French,
     German,
     Latin}                => {Italian}               0.0002565999  0.5666667 29.4218130
68  {French,
     Italian,
     Latin}                => {[Multiple languages]} 0.0002415058  0.6400000 10.8831211
69  {French,
     Latin,
     [Multiple languages]} => {Italian}               0.0002415058  0.5000000 25.9604232
70  {French,
     Italian,
     Latin}                => {English}               0.0003018822  0.8000000  1.0364675
71  {German,
```

```
     Italian,
     Latin}                 => {[Multiple languages]} 0.0004830116  0.6956522 11.8294795
72  {Italian,
     Latin,
     [Multiple languages]} => {German}               0.0004830116  0.7111111 15.6413752
73  {German,
     Latin,
     [Multiple languages]} => {Italian}              0.0004830116  0.6808511 35.3503635
74  {German,
     Italian,
     Latin}                 => {English}              0.0006641409  0.9565217  1.2392546
75  {English,
     Italian,
     Latin}                 => {German}               0.0006641409  0.6875000 15.1220327
76  {English,
     German,
     Latin}                 => {Italian}              0.0006641409  0.5301205 27.5243041
77  {Italian,
     Latin,
     [Multiple languages]} => {English}              0.0006490468  0.9555556  1.2380028
78  {English,
     Italian,
     Latin}                 => {[Multiple languages]} 0.0006490468  0.6718750 11.4251516
79  {French,
     German,
     Latin}                 => {[Multiple languages]} 0.0002867881  0.6333333 10.7697553
80  {French,
     Latin,
     [Multiple languages]} => {German}               0.0002867881  0.5937500 13.0599373
81  {French,
     German,
     Latin}                 => {English}              0.0004226351  0.9333333  1.2092120
82  {English,
     French,
     Latin}                 => {German}               0.0004226351  0.5490196 12.0760618
83  {French,
     Latin,
     [Multiple languages]} => {English}              0.0003924469  0.8125000  1.0526623
84  {English,
     French,
     Latin}                 => {[Multiple languages]} 0.0003924469  0.5098039  8.6691529
85  {German,
     Latin,
     [Multiple languages]} => {English}              0.0006792350  0.9574468  1.2404531
86  {English,
     German,
     Latin}                 => {[Multiple languages]} 0.0006792350  0.5421687  9.2195115
87  {German,
     Japanese,
     [Multiple languages]} => {English}              0.0001056588  1.0000000  1.2955843
88  {English,
```

```
     German,
     Japanese}              => {[Multiple languages]} 0.0001056588  0.6363636 10.8212852
89  {French,
     Italian,
     Spanish}               => {English}              0.0001207529  1.0000000  1.2955843
90  {English,
     Italian,
     Spanish}               => {French}               0.0001207529  0.5714286 15.8533142
91  {Italian,
     [Multiple languages],
     Spanish}               => {English}              0.0001207529  0.8888889  1.1516305
92  {English,
     Italian,
     Spanish}               => {[Multiple languages]} 0.0001207529  0.5714286  9.7170725
93  {French,
     German,
     Spanish}               => {[Multiple languages]} 0.0001056588  0.6363636 10.8212852
94  {German,
     [Multiple languages],
     Spanish}               => {French}               0.0001056588  0.6363636 17.6548272
95  {French,
     German,
     Spanish}               => {English}              0.0001660352  1.0000000  1.2955843
96  {English,
     German,
     Spanish}               => {French}               0.0001660352  0.5789474 16.0619104
97  {French,
     [Multiple languages],
     Spanish}               => {English}              0.0002264117  0.8333333  1.0796536
98  {German,
     [Multiple languages],
     Spanish}               => {English}              0.0001660352  1.0000000  1.2955843
99  {English,
     German,
     Spanish}               => {[Multiple languages]} 0.0001660352  0.5789474  9.8449287
100 {French,
     German,
     Italian}               => {[Multiple languages]} 0.0006490468  0.7166667 12.1868284
101 {French,
     Italian,
     [Multiple languages]} => {German}               0.0006490468  0.5972222 13.1363112
102 {French,
     German,
     [Multiple languages]} => {Italian}              0.0006490468  0.5058824 26.2658399
103 {French,
     German,
     Italian}               => {English}              0.0007999879  0.8833333  1.1444328
104 {French,
     Italian,
     [Multiple languages]} => {English}              0.0009509290  0.8750000  1.1336363
105 {English,
```

```
     French,
     Italian}              => {[Multiple languages]} 0.0009509290  0.5887850 10.0122172
106 {German,
     Italian,
     [Multiple languages]} => {English}              0.0013735642  0.9285714  1.2030426
107 {English,
     German,
     Italian}              => {[Multiple languages]} 0.0013735642  0.6026490 10.2479721
108 {English,
     Italian,
     [Multiple languages]} => {German}               0.0013735642  0.5027624 11.0586035
109 {French,
     German,
     [Multiple languages]} => {English}              0.0011924348  0.9294118  1.2041313
110 {English,
     French,
     German}               => {[Multiple languages]} 0.0011924348  0.6220472 10.5778367
111 {French,
     German,
     Italian,
     Latin}                => {[Multiple languages]} 0.0002113176  0.8235294 14.0040162
112 {French,
     Italian,
     Latin,
     [Multiple languages]} => {German}               0.0002113176  0.8750000 19.2462234
113 {French,
     German,
     Latin,
     [Multiple languages]} => {Italian}              0.0002113176  0.7368421 38.2574658
114 {French,
     German,
     Italian,
     Latin}                => {English}              0.0002415058  0.9411765  1.2193735
115 {English,
     French,
     Italian,
     Latin}                => {German}               0.0002415058  0.8000000 17.5965471
116 {English,
     French,
     German,
     Latin}                => {Italian}              0.0002415058  0.5714286 29.6690551
117 {French,
     Italian,
     Latin,
     [Multiple languages]} => {English}              0.0002113176  0.8750000  1.1336363
118 {English,
     French,
     Italian,
     Latin}                => {[Multiple languages]} 0.0002113176  0.7000000 11.9034138
119 {English,
     French,
```

```
     Latin,
     [Multiple languages]} => {Italian}              0.0002113176  0.5384615 27.9573788
120 {German,
     Italian,
     Latin,
     [Multiple languages]} => {English}              0.0004679175  0.9687500  1.2550973
121 {English,
     German,
     Italian,
     Latin}                => {[Multiple languages]} 0.0004679175  0.7045455 11.9807087
122 {English,
     Italian,
     Latin,
     [Multiple languages]} => {German}               0.0004679175  0.7209302 15.8573535
123 {English,
     German,
     Latin,
     [Multiple languages]} => {Italian}              0.0004679175  0.6888889 35.7676942
124 {French,
     German,
     Latin,
     [Multiple languages]} => {English}              0.0002716940  0.9473684  1.2273957
125 {English,
     French,
     German,
     Latin}                => {[Multiple languages]} 0.0002716940  0.6428571 10.9317065
126 {English,
     French,
     Latin,
     [Multiple languages]} => {German}               0.0002716940  0.6923077 15.2277812
127 {French,
     German,
     [Multiple languages],
     Spanish}              => {English}              0.0001056588  1.0000000  1.2955843
128 {English,
     French,
     German,
     Spanish}              => {[Multiple languages]} 0.0001056588  0.6363636 10.8212852
129 {English,
     German,
     [Multiple languages],
     Spanish}              => {French}               0.0001056588  0.6363636 17.6548272
130 {French,
     German,
     Italian,
     [Multiple languages]} => {English}              0.0005886704  0.9069767  1.1750649
131 {English,
     French,
     German,
     Italian}              => {[Multiple languages]} 0.0005886704  0.7358491 12.5130225
132 {English,
```

```
     French,
     Italian,
     [Multiple languages]} => {German}              0.0005886704  0.6190476 13.6163758
133 {French,
     German,
     Italian,
     Latin,
     [Multiple languages]} => {English}             0.0001962235  0.9285714  1.2030426
134 {English,
     French,
     German,
     Italian,
     Latin}                => {[Multiple languages]} 0.0001962235  0.8125000 13.8164624
135 {English,
     French,
     Italian,
     Latin,
     [Multiple languages]} => {German}              0.0001962235  0.9285714 20.4245637
136 {English,
     French,
     German,
     Latin,
     [Multiple languages]} => {Italian}             0.0001962235  0.7222222 37.4983891

> subrules <- rules[quality(rules)$lift > 1.5]
> inspect(subrules)
   lhs                        rhs                         support confidence      lift
1  {German,
    Norwegian}            => {[Multiple languages]} 0.0001056588  1.0000000 17.004877
2  {German,
    Russian}              => {[Multiple languages]} 0.0001358470  0.5625000  9.565243
3  {German,
    Swedish}              => {[Multiple languages]} 0.0001056588  0.6363636 10.821285
4  {Latin,
    Spanish}              => {[Multiple languages]} 0.0001207529  0.6153846 10.464540
5  {Italian,
    Latin}                => {German}               0.0006943291  0.5974026 13.140279
6  {Italian,
    Latin}                => {[Multiple languages]} 0.0006792350  0.5844156  9.937915
7  {German,
    Japanese}             => {[Multiple languages]} 0.0001056588  0.5833333  9.919511
8  {German,
    Spanish}              => {French}               0.0001660352  0.5500000 15.258815
9  {German,
    Spanish}              => {[Multiple languages]} 0.0001660352  0.5500000  9.352682
10 {German,
    Italian}              => {[Multiple languages]} 0.0014792230  0.5240642  8.911647
11 {French,
    German}               => {[Multiple languages]} 0.0012829995  0.5592105  9.509306
12 {English,
    German,
```

```
    Norwegian}            => {[Multiple languages]} 0.0001056588  1.0000000 17.004877
13 {English,
    [Multiple languages],
    Norwegian}            => {German}               0.0001056588  0.5000000 10.997842
14 {English,
    German,
    Russian}              => {[Multiple languages]} 0.0001358470  0.7500000 12.753658
15 {English,
    [Multiple languages],
    Russian}              => {German}               0.0001358470  0.5000000 10.997842
16 {English,
    German,
    Swedish}              => {[Multiple languages]} 0.0001056588  0.6363636 10.821285
17 {English,
    Latin,
    Spanish}              => {[Multiple languages]} 0.0001207529  0.7272727 12.367183
18 {French,
    Italian,
    Latin}                => {German}               0.0002565999  0.6800000 14.957065
19 {French,
    German,
    Latin}                => {Italian}              0.0002565999  0.5666667 29.421813
20 {French,
    Italian,
    Latin}                => {[Multiple languages]} 0.0002415058  0.6400000 10.883121
21 {French,
    Latin,
    [Multiple languages]} => {Italian}              0.0002415058  0.5000000 25.960423
22 {German,
    Italian,
    Latin}                => {[Multiple languages]} 0.0004830116  0.6956522 11.829480
23 {Italian,
    Latin,
    [Multiple languages]} => {German}               0.0004830116  0.7111111 15.641375
24 {German,
    Latin,
    [Multiple languages]} => {Italian}              0.0004830116  0.6808511 35.350364
25 {English,
    Italian,
    Latin}                => {German}               0.0006641409  0.6875000 15.122033
26 {English,
    German,
    Latin}                => {Italian}              0.0006641409  0.5301205 27.524304
27 {English,
    Italian,
    Latin}                => {[Multiple languages]} 0.0006490468  0.6718750 11.425152
28 {French,
    German,
    Latin}                => {[Multiple languages]} 0.0002867881  0.6333333 10.769755
29 {French,
    Latin,
```

```
    [Multiple languages]} => {German}              0.0002867881  0.5937500 13.059937
30 {English,
    French,
    Latin}                => {German}              0.0004226351  0.5490196 12.076062
31 {English,
    French,
    Latin}                => {[Multiple languages]} 0.0003924469  0.5098039  8.669153
32 {English,
    German,
    Latin}                => {[Multiple languages]} 0.0006792350  0.5421687  9.219512
33 {English,
    German,
    Japanese}             => {[Multiple languages]} 0.0001056588  0.6363636 10.821285
34 {English,
    Italian,
    Spanish}              => {French}              0.0001207529  0.5714286 15.853314
35 {English,
    Italian,
    Spanish}              => {[Multiple languages]} 0.0001207529  0.5714286  9.717072
36 {French,
    German,
    Spanish}              => {[Multiple languages]} 0.0001056588  0.6363636 10.821285
37 {German,
    [Multiple languages],
    Spanish}              => {French}              0.0001056588  0.6363636 17.654827
38 {English,
    German,
    Spanish}              => {French}              0.0001660352  0.5789474 16.061910
39 {English,
    German,
    Spanish}              => {[Multiple languages]} 0.0001660352  0.5789474  9.844929
40 {French,
    German,
    Italian}              => {[Multiple languages]} 0.0006490468  0.7166667 12.186828
41 {French,
    Italian,
    [Multiple languages]} => {German}              0.0006490468  0.5972222 13.136311
42 {French,
    German,
    [Multiple languages]} => {Italian}             0.0006490468  0.5058824 26.265840
43 {English,
    French,
    Italian}              => {[Multiple languages]} 0.0009509290  0.5887850 10.012217
44 {English,
    German,
    Italian}              => {[Multiple languages]} 0.0013735642  0.6026490 10.247972
45 {English,
    Italian,
    [Multiple languages]} => {German}              0.0013735642  0.5027624 11.058604
46 {English,
    French,
```

```
    German}               => {[Multiple languages]} 0.0011924348  0.6220472 10.577837
47 {French,
    German,
    Italian,
    Latin}                => {[Multiple languages]} 0.0002113176  0.8235294 14.004016
48 {French,
    Italian,
    Latin,
    [Multiple languages]} => {German}               0.0002113176  0.8750000 19.246223
49 {French,
    German,
    Latin,
    [Multiple languages]} => {Italian}              0.0002113176  0.7368421 38.257466
50 {English,
    French,
    Italian,
    Latin}                => {German}               0.0002415058  0.8000000 17.596547
51 {English,
    French,
    German,
    Latin}                => {Italian}              0.0002415058  0.5714286 29.669055
52 {English,
    French,
    Italian,
    Latin}                => {[Multiple languages]} 0.0002113176  0.7000000 11.903414
53 {English,
    French,
    Latin,
    [Multiple languages]} => {Italian}              0.0002113176  0.5384615 27.957379
54 {English,
    German,
    Italian,
    Latin}                => {[Multiple languages]} 0.0004679175  0.7045455 11.980709
55 {English,
    Italian,
    Latin,
    [Multiple languages]} => {German}               0.0004679175  0.7209302 15.857354
56 {English,
    German,
    Latin,
    [Multiple languages]} => {Italian}              0.0004679175  0.6888889 35.767694
57 {English,
    French,
    German,
    Latin}                => {[Multiple languages]} 0.0002716940  0.6428571 10.931707
58 {English,
    French,
    Latin,
    [Multiple languages]} => {German}               0.0002716940  0.6923077 15.227781
59 {English,
    French,
```

```
    German,
    Spanish}              => {[Multiple languages]} 0.0001056588  0.6363636 10.821285
60 {English,
    German,
    [Multiple languages],
    Spanish}              => {French}               0.0001056588  0.6363636 17.654827
61 {English,
    French,
    German,
    Italian}              => {[Multiple languages]} 0.0005886704  0.7358491 12.513023
62 {English,
    French,
    Italian,
    [Multiple languages]} => {German}               0.0005886704  0.6190476 13.616376
63 {English,
    French,
    German,
    Italian,
    Latin}                => {[Multiple languages]} 0.0001962235  0.8125000 13.816462
64 {English,
    French,
    Italian,
    Latin,
    [Multiple languages]} => {German}               0.0001962235  0.9285714 20.424564
65 {English,
    French,
    German,
    Latin,
    [Multiple languages]} => {Italian}              0.0001962235  0.7222222 37.498389
```