Week 7 Exercises

James Quacinella

10/11/2015

```
library(hash)
```

```
## hash-2.2.6 provided by Decision Patterns
```

```
library(stringr)
```

6.1.1 (section 6.1.5)

Suppose there are 100 items, numbered 1 to 100, and also 100 baskets, also numbered 1 to 100. Item i is in basket b if and only if i divides b with no remainder. Thus, item 1 is in all the baskets, item 2 is in all fifty of the even-numbered baskets, and so on. Basket 12 consists of items {1, 2, 3, 4, 6, 12}, since these are all the integers that divide 12. Answer the following questions:

(a) If the support threshold is 5, which items are frequent?

(b) If the support threshold is 5, which pairs of items are frequent?

(c) What is the sum of the sizes of all the baskets?

Answer

(a) Well, we need to find the items that will appear in 5 or more buckets. Since there are 100 buckets, and the rule is that item i does to bucket b if i divides b evenly, it seems obvious that the numbers up until 20 will be frequent. For example, item 1 goes into 100 buckets. Item 2 will go into 50 buckets (all the even numbers). Item 20 will go to buckets 20, 40, 60, 80 and 100. However, once we hit 21, we do not meet the support requirement. Item 21 will only be in buckets 21, 42, 63, and 84.

(b) We only need to investigate the items numbered from 1 - 20, due to the monoticity property. So we need to know which of thesse pairs end up in the same buckets.

There is some weird logic here, but here are two ways of finding pairs of numbers from 1 - 20 that will also form pairs that are frequent. If the pair (a, b) is of the form such that $bmoda$ is 0 (a divides b evenly) since the itemset for this pair would be equivalent to the itemset for b, which we know is frequent.

Otherwise, if the pair is of the form (a, b) where $a * b <= 20$, then the pair should also be frequent. For example, (2,7) should be frequent since 2 will appear in all even numbered buckets, and 7 appears in bckets that are multiples of 7. The pair, therefore, would appear in the intersection of these sets, i.e. buckets 14, 28, 42, etc. This is equivalent to the item set for 14, which we know is frequent since its $<$ 20. Here is R code to produce the pairs:

```
# Find all pairs of ints from 1 to 20
pairs <- combn(1:20, 2, simplify = FALSE)

# Find all frequent pairs by using our conditions above
```

```r
frequent_pairs <- lapply(pairs, function(pair) {
  if(pair[[2]] %% pair[[1]] == 0) { return(pair); }
  else if(pair[[2]] * pair[[1]] <= 20) { return(pair) }
})

# Remove NULLs
frequent_pairs <- frequent_pairs[!sapply(frequent_pairs, is.null)]

# Convert to dataframe and print
frequent_pairs.df <- as.data.frame(frequent_pairs)
colnames(frequent_pairs.df) <- 1:length(frequent_pairs)
frequent_pairs.df
```

```
##   1 2 3 4 5 6 7 8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
## 1 1 1 1 1 1 1 1 1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2
## 2 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  3  4  5  6  7  8  9 10
##   28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
## 1  2  2  2  2  2  3  3  3  3  3  3  3  4  4  4  4  4  5  5  5  6  6  7  8
## 2 12 14 16 18 20  4  5  6  9 12 15 18  5  8 12 16 20 10 15 20 12 18 14 16
##   52 53
## 1  9 10
## 2 18 20
```

(c)

```r
# Stolen from http://rosettacode.org/wiki/Factors_of_an_integer#R
# Returns list of factors of n
factors <- function(n)
{
   if(length(n) > 1)
   {
      lapply(as.list(n), factors)
   } else
   {
      one.to.n <- seq_len(n)
      one.to.n[(n %% one.to.n) == 0]
   }
}

# Answer is 482
sum(unlist(lapply(1:100, function(i) { length(factors(i)) } )))
```

```
## [1] 482
```

We take the bucket numbers from 1 - 100, find the factors of each of them and return the number of factors via lapply, unlist it to flatten it, and then sum the lengths via sum for final answer of **482**.

6.1.5 (section 6.1.5)

For the data of Exercise 6.1.1, what is the confidence of the following association rules?

(a) {5, 7} -> 2

(b) {2, 3, 4} -> 5

Answer

The definition of the confidence of the rule I $\rightarrow$ j to be the ratio of the support for I union {j} to the support for I. That is, the confidence of the rule is the fraction of the baskets with all of I that also contain j

(a) First lets find out which baskets contain both 5 and 7. For item 5, we have baskets 5, 10, 15, 20, 25, . . . 70, 75, 80, 85, 90, 95, 100. For item 7, we have baskets 7, 14, 21, 28, 35, . . . , 70, 77, 84, 91, 98. There are two overlaps, buckets 35 and 70.

```
factors(35)    # 1   5   7   35
factors(70)    # 1   2   5   7 10 14 35 70
```

So, the confidence will be **1/2**. The support for I in this case is 2, and the support for I union j is 1 (bucket 70)

(b) Lets not do this by hand, so lets write some code to find the support for I in this case:

```
unlist(lapply(1:100, function(i) {if(i%%2==0 & i%%3==0 & i%%4==0) {return(i)}})) # 12 24 36 48 60 72 84
```

Therefore, the support for I is 8. Out of those 8, only 1 of them would include 5, bucket 60. Therefore, the confidence is **1/8**.

6.3.1 (section 6.3.4)

Here is a collection of twelve baskets. Each contains three of the six items 1 through 6:

{1, 2, 3} {1, 3, 5} {3, 5, 6} {2, 3, 4} {2, 4, 6} {1, 2, 4} {3, 4, 5} {1, 3, 4} {2, 3, 5} {4, 5, 6} {2, 4, 5} {3, 4, 6}

Suppose the support threshold is 4. On the first pass of the PCY Algorithm we use a hash table with 11 buckets, and the set {i, j} is hashed to bucket i x j mod 11.

(a) By any method, compute the support for each item and each pair of items.

(b) Which pairs hash to which buckets?

(c) Which buckets are frequent?

(d) Which pairs are counted on the second pass of the PCY Algorithm?

Answer

(a) Usnig the hash table:

```r
buckets <- list(c(1, 2, 3),
c(1, 3, 5),
c(3, 5, 6),
c(2, 3, 4),
c(2, 4, 6),
c(1, 2, 4),
c(3, 4, 5),
c(1, 3, 4),
c(2, 3, 5),
c(4, 5, 6),
c(2, 4, 5),
c(3, 4, 6))

# Lets use hash library to count pairs
library(hash)
pair_counts <- hash()

# Generate all pairs as strings to be used as keys
pairs <- unlist(lapply(buckets, function(x) {return(c(paste(x[1], x[2]), paste(x[1], x[3]), paste(x[2],

# Count pairs using lapply
temp <- lapply(pairs, function(pair) {
  if(pair %in% keys(pair_counts)) {
    pair_counts[[pair]] <- pair_counts[[pair]] + 1;
  } else {
    pair_counts[[pair]] <- 1;
  }
})
```

Lets look at the results for singletons (singleton in first row, count in second row):

```r
# Supports for single items
base:::table(unlist(buckets))
```

```
##
## 1 2 3 4 5 6
## 4 6 8 8 6 4
```

Lets see the results for pairs (pairs are keys, values are counts):

```r
# Print pair counts
pair_counts
```

```
## <hash> containing 14 key-value pair(s).
##   1 2 : 2
##   1 3 : 3
##   1 4 : 2
##   1 5 : 1
##   2 3 : 3
##   2 4 : 4
##   2 5 : 2
##   2 6 : 1
##   3 4 : 4
##   3 5 : 4
##   3 6 : 2
##   4 5 : 3
##   4 6 : 3
##   5 6 : 2
```

(b)

```r
get_hash <- function(x, y) {
  return((x * y) %% 11);
}

unlist(lapply(keys(pair_counts), function(pair) {
  items <- str_split(pair, " ")[[1]];
  return( paste(as.integer(items[1]), as.integer(items[2]), "==>", get_hash(as.integer(items[1]), as.in
}))
```

```
##  [1] "1 2 ==> 2"  "1 3 ==> 3"  "1 4 ==> 4"  "1 5 ==> 5"  "2 3 ==> 6"
##  [6] "2 4 ==> 8"  "2 5 ==> 10" "2 6 ==> 1"  "3 4 ==> 1"  "3 5 ==> 4"
## [11] "3 6 ==> 7"  "4 5 ==> 9"  "4 6 ==> 2"  "5 6 ==> 8"
```

Notice that the pair 1, 6 never occurs so is not calculated.

(c) Going to do this by hand combining both outputs from above:

0 ==> No pairs

1 ==> (2,6), (3,4) ==> 1 + 4 = 5 total occurances of pairs hashed here

2 ==> (1,2), (4,6) ==> 2 + 3 = 5 total occurances of pairs hashed here

3 ==> (1,3) ==> 3 total occurances of pairs hashed here

4 ==> (1,4), (3,5) ==> 2 + 4 = 6 total occurances of pairs hashed here

5 ==> (1,5) ==> 1 total occurances of pairs hashed here

6 ==> (1,6), (2,3) ==> 3 total occurances of pairs hashed here

7 ==> (3,6) ==> 2 total occurances of pairs hashed here

8 ==> (2,4), (5,6) ==> 4 + 2 = 6 total occurances of pairs hashed here

9 ==> (4,5) ==> 3 total occurances of pairs hashed here

10 ==> (2,5) ==> 2 total occurances of pairs hashed here

Therefore, buckets 1, 2, 4, 8 are frequent for support = 4.

(d) Well, for singletons, all are frequent, since they all support >= 4. For pairs, the candidates are the pairs in frequent buckets (p217 of the book):

(2,6), (3,4), (1,2), (4,6), (1,4), (3,5), (2,4), (5,6)

6.3.2 (section 6.3.4)

Suppose we run the Multistage Algorithm on the data of Exercise 6.3.1, with the same support threshold of 4. The first pass is the same as in that exercise, and for the second pass, we hash pairs to nine buckets, using the hash function that hashes $\{i, j\}$ to bucket $i + j \bmod 9$. Determine the counts of the buckets on the second pass. Does the second pass reduce the set of candidate pairs? Note that all items are frequent, so the only reason a pair would not be hashed on the second pass is if it hashed to an infrequent bucket on the first pass.

Answer

```r
get_second_hash <- function(x, y) {
  return((x + y) %% 9);
}

unlist(lapply(keys(pair_counts), function(pair) {
  items <- str_split(pair, " ")[[1]];
  return( paste(as.integer(items[1]), as.integer(items[2]), "==>", get_second_hash(as.integer(items[1])
}))
```

```
##  [1] "1 2 ==> 3" "1 3 ==> 4" "1 4 ==> 5" "1 5 ==> 6" "2 3 ==> 5"
##  [6] "2 4 ==> 6" "2 5 ==> 7" "2 6 ==> 8" "3 4 ==> 7" "3 5 ==> 8"
## [11] "3 6 ==> 0" "4 5 ==> 0" "4 6 ==> 1" "5 6 ==> 2"
```

Summarizing by hand, like above, but here we will exclude pairs that did not map to a frequent bucket on the first pass:

0 ==> (3,6), (4,5) ==> all pairs filtered ==> 0 total occurances of pairs hashed here

1 ==> (4,6) ==> (4,6) after filtering ==> 3 total occurances of pairs hashed here

2 ==> (5,6) ==> (5,6) after filtering ==> 2 total occurances of pairs hashed here

3 ==> (1,2) ==> all pairs filtered ==> 0 total occurances of pairs hashed here

4 ==> (1,3) ==> all pairs filtered ==> 0 total occurances of pairs hashed here

5 ==> (1,4), (2,3) ==> (1,4) after filtering ==> 2 total occurances of pairs hashed here

6 ==> (1,5), (2,4) ==> (2,4) after filtering => 4 total occurances of pairs hashed here

7 ==> (2,5), (3,4) ==> (3,4) after filtering ==> 4 total occurances of pairs hashed here

8 ==> (2,6), (3,5) ==> (2,6), (3,5) after filtering ==> 1 + 4 = 5 total occurances of pairs hashed here

9 ==> None ==> total occurances of pairs hashed here

The only frequent bucket is 8, which has only two pairs associated with it (2,6), and (3,5)

NOTE I feel like I might be misinterpreting the algorithm