queries

# queries

source('functions.R')

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':  
  
 filter, lag

The following objects are masked from 'package:base':  
  
 intersect, setdiff, setequal, union

Attaching package: 'rlang'

The following objects are masked from 'package:purrr':  
  
 %@%, flatten, flatten\_chr, flatten\_dbl, flatten\_int, flatten\_lgl,  
 flatten\_raw, invoke, splice

### Example (projection query)

The input data frame represents the table specifying a weighted relation on three variables: X, Y, and Z. The projection query projects the input relation on the input variables X and Y. It’s important to note that the query uses summation as the aggregation function, which plays a vital role in the calculation process.

r1\_X\_Y\_Z <- data.frame(X=c(0,0,0,1,1),  
 Y=c(0,0,1,0,0),  
 Z=c(0,1,1,0,1),  
 Weight = c(0.3,0.3,0.2,0.1,0.1),  
 stringsAsFactors = FALSE)  
result <- project(r1\_X\_Y\_Z, c("Y","Z"))  
print(result)

Y Z Weight  
0\_0 0 0 0.4  
0\_1 0 1 0.4  
1\_1 1 1 0.2

### Example (projection query)

The input data frame is a table of a weighted relation on X, Y, and Z. The following projects the relation on X and Y. We explicitly specify the `FUN` parameter as `max`, which results in applying the `max` of the weights to obtain the maximum utility for X and Y overall values of Z.

r2\_X\_Y\_Z <-  
 data.frame(X = c(0,0,0,0,1,1,1,1),  
 Y = c(0,0,1,1,0,0,1,1),  
 Z = c(0,1,0,1,0,1,0,1),  
 Weight = c(1,2,5,3,7,4,6,5),  
 stringsAsFactors = FALSE)  
  
result1 <- project(r2\_X\_Y\_Z, c("X","Y"),FUN=max)  
print(result1)

X Y Weight  
0\_0 0 0 2  
0\_1 0 1 5  
1\_0 1 0 7  
1\_1 1 1 6

### Example (conditioning query)

The following uses the `cond` operator on the weighted relation `r2\_X\_Y\_Z` to select only tuples with value assignment `X=0` and `Y=1`.

r2\_X\_Y\_Z <-  
 data.frame(X = c(0,0,0,0,1,1,1,1),  
 Y = c(0,0,1,1,0,0,1,1),  
 Z = c(0,1,0,1,0,1,0,1),  
 Weight = c(1,2,5,3,7,4,6,5),  
 stringsAsFactors = FALSE)  
  
result <- cond(r2\_X\_Y\_Z, c("X","Y"), c(0, 1))  
print(result)

X Y Z Weight  
1 0 1 0 5  
2 0 1 1 3

### Example (combination query)

The example combines two potential functions: 1) phi\_X\_Y on the variables X and Y, and 2) phi\_Y\_Z on Y and Z. The comb function joins phi\_X\_Y and phi\_Y\_Z, multiplying the “Weight” attribute of joined tuples, tuples matching on the value of the shared Y attribute. The result is a joint probability on the combined variables X, Y, and Z.

phi\_X\_Y <- data.frame(X = c(0,0,1,1),  
 Y = c(0,1,0,1),  
 Weight = c(0.15, 0.6, 0.225, 0.025),  
 stringsAsFactors = FALSE)  
  
phi\_Y\_Z <- data.frame(Y = c(0,1,1),  
 Z = c(0,0,1),  
 Weight = c(1, 0.25, 0.75),  
 stringsAsFactors = FALSE)  
  
# Run the function  
rel <- comb(phi\_X\_Y, phi\_Y\_Z, FUN="\*")  
  
# Print the resulting relation.  
print(rel)

Y X Z Weight  
1 0 0 0 0.15000  
2 0 1 0 0.22500  
3 1 0 0 0.15000  
4 1 0 1 0.45000  
5 1 1 0 0.00625  
6 1 1 1 0.01875

### Example (combination query)

Consider two utility functions, f1(X, Y) and f2(Y, Z), represented by the two weighted relations r1 and r2 in the code below. We combine the two relations to get the sum of the two functions, a function of the three variables f(X, Y, Z), defined by the result of the combination query. Note that we specify the combining function as “+” for summing the “Weights” of the relations r1 and r2 joined tuples instead of multiplying them as in the case by default.

r1 <- data.frame(X = c(0,0,1,1),  
 Y = c(0,1,0,1),  
 Weight = c(4,8,0,5),  
 stringsAsFactors = FALSE)  
  
r2 <- data.frame(Y = c(0,1,1),  
 Z = c(0,0,1),  
 Weight = c(0,5,6),  
 stringsAsFactors = FALSE)   
# Run the function  
rel <- comb(r1, r2, FUN="+")  
  
# Print the resulting relation.  
print(rel)

Y X Z Weight  
1 0 0 0 4  
2 0 1 0 0  
3 1 0 0 13  
4 1 0 1 14  
5 1 1 0 10  
6 1 1 1 11

### Example (Combining a list of relations)

We consider an example of a Bayesian network with conditional probabilities as weighted relations. We use the comb\_list function to combine the conditional probabilities for the three-node network to compute the joint probability distribution.

P\_X <- data.frame(X = c(0,1),  
 Weight = c(0.75, 0.25),  
 stringsAsFactors = FALSE)  
P\_Y\_X <- data.frame(X = c(0,0,1,1),  
 Y = c(0,1,0,1),  
 Weight = c(0.2,0.8,0.9,0.1),  
 stringsAsFactors = FALSE)  
P\_Z\_Y <- data.frame(Y = c(0,0,1,1),  
 Z = c(0,1,0,1),  
 Weight = c(1,0,0.25,0.75),  
 stringsAsFactors = FALSE)  
JPD <- comb\_list(list(P\_X,P\_Y\_X,P\_Z\_Y))  
print(JPD)

Y X Z Weight  
1 0 0 0 0.15000  
2 0 0 1 0.00000  
3 0 1 0 0.22500  
4 0 1 1 0.00000  
5 1 0 0 0.15000  
6 1 0 1 0.45000  
7 1 1 0 0.00625  
8 1 1 1 0.01875

### Example (Add Relation Remove Variable)

The following example shows how to use the ‘assoc’ function to create the variable-relations association dictionary and how to update the dictionary throughout the execution of the QCG algorithm by using the ‘add\_relation\_remove\_var’ function to add the induced relations from the elimination step to the dictionary and also remove the eliminated variable from the dictionary.

f\_X <- data.frame(X = c(0,0,1), Weight = c(3,5,7), stringsAsFactors = FALSE)  
f\_Y <- data.frame(Y = c(1,0,1), Weight = c(2,10,6), stringsAsFactors = FALSE)  
f\_Z <- data.frame(Z = c(1,1,0), Weight = c(4,3,1), stringsAsFactors = FALSE)  
  
dict = assoc(list(f\_X,f\_Y,f\_Z),c("X","Y","Z"))  
f\_X\_Y <- data.frame(X = c(0,0,1,1), Y = c(0,1,0,1), Weight = c(9,3,4,7), stringsAsFactors = FALSE)  
  
dict2 <- add\_relation\_remove\_var(dict, f\_X\_Y, "Z")  
print(dict2)

$X  
$X[[1]]  
 X Weight  
1 0 3  
2 0 5  
3 1 7  
  
  
$Y  
$Y[[1]]  
 Y Weight  
1 1 2  
2 0 10  
3 1 6  
  
$Y[[2]]  
 X Y Weight  
1 0 0 9  
2 0 1 3  
3 1 0 4  
4 1 1 7