Contingency tables creation examples

Anton Antonov
MathematicaForPrediction at GitHub
MathematicaVsR project at GitHub
October, 2016

Introduction

In statistics contingency tables are matrices used to show the co-occurrence of variable values of multi-dimensional data. They are fundamental in many types of research. This document shows how to use several functions implemented in *Mathematica* for the construction of contingency tables.

Code

In this document we are going to use the implementations in the package MathematicaForPredictionUtilities.m from MathematicaForPrediction at GitHub, [1].

```
Import[
 "https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/
   MathematicaForPredictionUtilities.m"]
```

The implementation of `CrossTabulate` in [1] is based on `Tally`, `GatherBy`, and `SparseArray`. The implementation of `xtabsViaRLink` in [1] is based on R's function `xtabs` called via RLink.

Other package used in this document are [2] and [4] imported with these commands:

```
Import[
 "https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/
   MosaicPlot.m"]
Import[
 "https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/
   Misc/RSparseMatrix.m"]
```

For a different approach to implementing cross-tabulation than those taken in [1] see the Stack Overflow answer http://stackoverflow.com/a/8101951 by Mr.Wizard.

Using Titanic data

Getting data

```
In[783]:= titanicData = Flatten@*List @@@ ExampleData[{"MachineLearning", "Titanic"}, "Data"];
titanicData = DeleteCases[titanicData, {___, _Missing, ___}];
```

```
In[785]:= titanicColumnNames = Flatten@*List @@
             ExampleData[{"MachineLearning", "Titanic"}, "VariableDescriptions"];
          aTitanicColumnNames = AssociationThread[
             titanicColumnNames → Range[Length[titanicColumnNames]]];
```

Note that we have removed the records with missing data (for simpler exposition).

Data summary

```
In[787]:= Dimensions[titanicData]
Out[787]= {1046, 4}
In[788]:= RecordsSummary[titanicData, titanicColumnNames]
```

Out[788]=

1 passenger class	
3rd	501
1st	284
2nd	261

2 passenger age	
Min	0.1667
1st Qu	21.
Median	28.
Mean	29.8811
3rd Qu	39.
Max	80.

3 passenger sex	
male	658
female	388

4 passenger survival	
died	619
survived	427

Using CrossTabulate

Assume that we want to group the people according to their passenger class and survival and we want to find the average age for each group.

We can do that by

1. finding the counts contingency table *C* for the variables "passenger class" and "passenger survival",

2. finding the age contingency table *A* for the same variables, and

3. do the element-wise division *A*/*C*.

```
In[789]:= ctCounts = CrossTabulate[titanicData[[All,
              aTitanicColumnNames /@ {"passenger class", "passenger survival"}]]];
          MatrixForm[#1, TableHeadings → {#2, #3}] & @@ ctCounts
```

Out[790]//MatrixForm=

	died	survived
1st	103	181
2nd	146	115
3rd	370	131

```
In[791]:= ctTotalAge = CrossTabulate[titanicData[[All, aTitanicColumnNames /@
               {"passenger class", "passenger survival", "passenger age"}]]];
          MatrixForm[#1, TableHeadings → {#2, #3}] & @@ ctTotalAge
```

Out[792]//MatrixForm=

	died	survived
1st	4454.5	6666.92
2nd	4842.5	2858.75
3rd	9610.58	2822.42

In[793]:=
```
MatrixForm[ctTotalAge[[1]] / Normal[ctCounts[[1]]],
 TableHeadings → Values[Rest[ctTotalAge]]]
```

Out[793]//MatrixForm=

	died	survived
1st	43.2476	36.8338
2nd	33.1678	24.8587
3rd	25.9745	21.5452

(We have to make the sparse array `ctCounts` a regular array because otherwise we get warnings for division by zero because of the sparse array's default value.)

Let us repeat the steps above by dividing the passengers before-hand according to their sex.

In[946]:=
```
Association@
 Map[
  (mCount = CrossTabulate[
      #[[All, aTitanicColumnNames /@ {"passenger class", "passenger survival"}]]];
    mAge = CrossTabulate[#[[All, aTitanicColumnNames /@
         {"passenger class", "passenger survival", "passenger age"}]]];
    #[[1, aTitanicColumnNames["passenger sex"]]] -> MatrixForm[
      mAge[[1]] / Normal[mCount[[1]]], TableHeadings → Values[Rest[mAge]]]) &,
  GatherBy[titanicData, #[[aTitanicColumnNames["passenger sex"]]] &]]
```

Out[946]= ⟨| female →

	died	survived
1st	35.2	37.1094
2nd	34.0909	26.7111
3rd	23.4188	20.8148

, male →

	died	survived
1st	43.6582	36.1682
2nd	33.0926	17.4493
3rd	26.6796	22.4364

|⟩

Using R's xtabs (via RLink)

The alternative of `CrossTabulate` is `xtabsViaRLink` that is uses R's function `xtabs` via RLink.

In[795]:=
```
Needs["RLink`"]
RLinkResourcesInstall[]
InstallR[]
```

Out[796]= {Paclet[RLinkRuntime, 9.0.0.0, <>]}

In[798]:=
```
ctCounts = FromRXTabsForm@xtabsViaRLink[titanicData[[All,
      aTitanicColumnNames /@ {"passenger class", "passenger survival"}]],
    {"passenger.sex", "passenger.survival"},
    " ~ passenger.sex + passenger.survival"];
MatrixForm[#1, TableHeadings → {#2, #3}] &@@ ctCounts
```

Out[799]//MatrixForm=

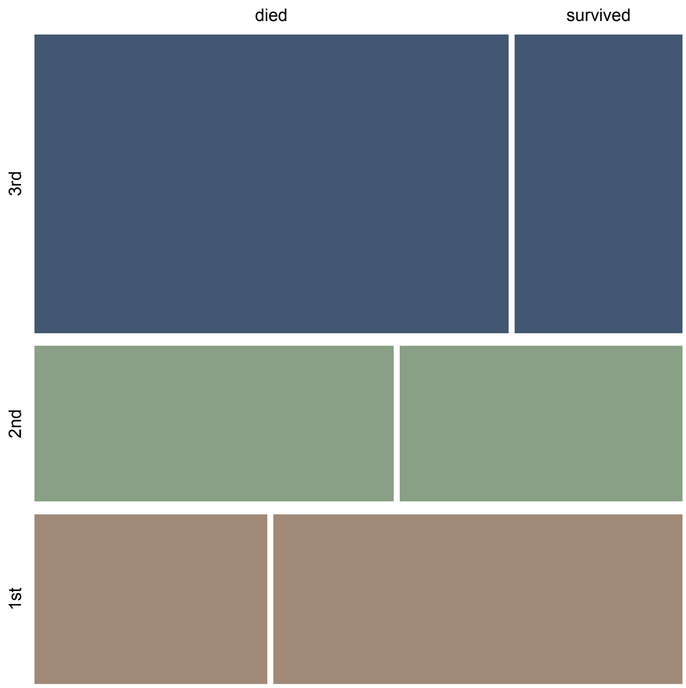
	died	survived
1st	103.	181.
2nd	146.	115.
3rd	370.	131.

Relation to mosaic plots

The graphical visualization of a dataset with mosaic plots, [2,3], is similar in spirit to contingency tables. Compare the following mosaic plot with the contingency table in the last section.

In[1046]:=
```
MosaicPlot[
 titanicData⟦All, aTitanicColumnNames /@ {"passenger class", "passenger survival"}⟧]
```

Out[1046]=



Using larger data

Let us consider an example with larger data that has larger number of unique values in its columns.

Getting online retail invoices data

The following dataset is taken from [6].

In[800]:=
```
data = Import["/Volumes/WhiteSlimSeagate/Datasets/UCI
      Online Retail Data Set/Online Retail.csv"];
columnNames = First[data];
data = Rest[data];
```

In[803]:=
```
aColumnNames = AssociationThread[columnNames → Range[Length[columnNames]]];
```

Data summary

We have ≈ 66 000 rows and 8 columns:

In[804]:=
```
Dimensions[data]
```
Out[804]=
```
{65 499, 8}
```

Here is a summary of the columns:

In[1047]:=
```
Magnify[#, 0.75] &@RecordsSummary[data, columnNames]
```
Out[1047]=
```
{ 1 InvoiceNo          2 StockCode         3 Description
  537 434   675        85123A   353        WHITE HANGING HEART T-LIGHT HOLDER  358
  538 071   652        22423    279        REGENCY CAKESTAND 3 TIER            278
  538 349   620    ,   22469    224    ,   HEART OF WICKER SMALL               224   ,
  537 638   601        22834    213        HAND WARMER BABUSHKA DESIGN         213
  537 237   597        22111    207        SCOTTIE DOG HOT WATER BOTTLE        207
  536 876   593        85099B   205        JUMBO BAG RED RETROSPOT             205
  (Other)   61 761     (Other)  64 018     (Other)                             64 014

  4 Quantity           5 InvoiceDate              6 UnitPrice          7 CustomerID         8 Country
  Min       -74 215    12/6/10 16:57   675        Min       0                    25 281     United Kingdom  61 186
  1st Qu    1          12/9/10 14:09   652        1st Qu    1.25       12 748    695        Germany         982
  Median    2      ,   12/10/10 14:59  621    ,   Median    2.51   ,   17 841    481    ,   France          967
  3rd Qu    8          12/7/10 15:28   601        3rd Qu    4.24       14 606    421        EIRE            504       }
  Mean      182 660/21 833  12/6/10 9:58  597     Mean      5.85759    15 311    418        Spain           355
  Max       74 215     12/3/10 11:36   593        Max       16 888.    14 911    377        Portugal        212
                       (Other)         61 760                          (Other)   37 826     (Other)         1293
```

Contingency tables

Country vs. StockCode

There is no one-to-one correspondence between the values of the column “Description” and the column “StockCode” which can be seen with this command:

In[1035]:=
```
MinMax@Map[Length@*Union,
  GatherBy[data⟦All, aColumnNames /@ {"Description", "StockCode"}⟧, First]]
```
Out[1035]=
```
{1, 144}
```

The way in which the column “StockCode” was ingested made it have multiple types for its values:

In[1038]:=
```
Tally[NumberQ /@ data⟦All, aColumnNames["StockCode"]⟧]
```
Out[1038]=
```
{{False, 9009}, {True, 56 490}}
```

So let us convert it to all strings:

In[1039]:=
```
data⟦All, aColumnNames["StockCode"]⟧ =
  ToString /@ data⟦All, aColumnNames["StockCode"]⟧;
```

Here we find the contingency table for “Country” and “StockCode” over “Quantity” using `CrossTabulate`:

In[1040]:=
```
AbsoluteTiming[
 ctRes =
  CrossTabulate[data⟦All, aColumnNames /@ {"Country", "StockCode", "Quantity"}⟧];
]
```
Out[1040]=
```
{0.256339, Null}
```

Here we find the contingency table for “Country” and “StockCode” over “Quantity” using `xtabsViaRLink`:

In[1041]:=
```
AbsoluteTiming[rres =
   xtabsViaRLink[data⟦All, aColumnNames /@ {"Country", "StockCode", "Quantity"}⟧,
    {"Country", "StockCode", "Quantity"}, "Quantity ~ Country + StockCode"];
  ctRRes = FromRXTabsForm[rres];
]
```

Out[1041]=
```
{0.843621, Null}
```

Both functions produce the same result:

In[1042]:=
```
ctRRes["matrix"] == N@ctRRes[[1]]
```

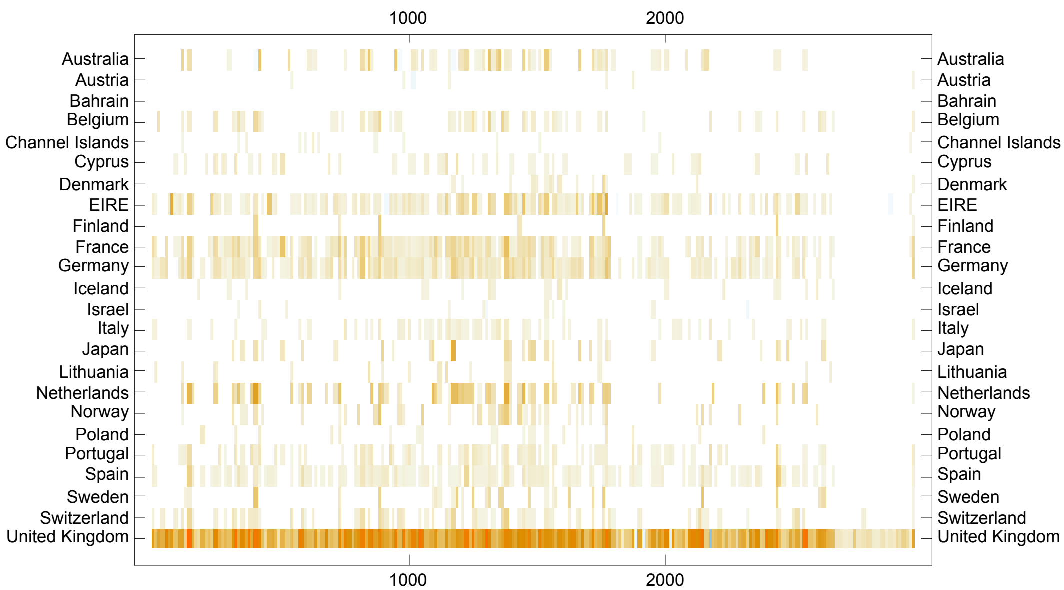
Out[1042]=
```
True
```

Note that `xtabsViaRLink` is slower but still fairly quick.

Here we plot the contingency table using `MatrixPlot` :

In[1043]:=
```
MatrixPlot[ctRRes["matrix"], AspectRatio → 1 / 1.5, FrameTicks →
  {{#, #} &@Table[{i, ctRRes["rownames"]⟦i⟧}, {i, Length[ctRRes["rownames"]]}],
   {Automatic, Automatic}}, ImageSize → 550]
```

Out[1043]=



Country vs. Quarter

Let us extend the data with columns that have months and quarters corresponding to the invoice dates.

The following commands computing date objects and extracting month and quarter values from them are too slow.

```
(*AbsoluteTiming[
 dobjs=DateObject[{#,{"Month","/","Day","/","Year"," ","Hour",":","Minute"}}]&/@
    data⟦All,aColumnNames["InvoiceDate"]⟧;
]*)
```

Out[975]= {30.2595, Null}

```
(*AbsoluteTiming[
 dvals=DateValue[dobjs,{"MonthName","QuarterNameShort"}];
]*)
```

Out[976]= {91.1732, Null}

We can use the following ad hoc computation instead.

In[1050]:=
```
dvals = StringSplit[#, {"/", " ", ":"}] & /@ data⟦All, aColumnNames["InvoiceDate"]⟧;
```

This summary shows that the second value in the dates is day of month, and the first value is most likely month.

In[1052]:=
```
Magnify[#, 0.75] &@RecordsSummary[dvals⟦All, 1 ;; 3⟧, "MaxTallies" → 16]
```

Out[1052]=

1 column 1	
12	42481
1	23018

2 column 2	
6	5710
17	5672
7	4757
10	4734
5	4468
9	4008
13	3728
14	3597
12	3260
1	3108
20	2851
8	2647
16	2436
3	2202
2	2109
(Other)	10212

3 column 3	
10	42481
11	23018

These commands extend the data and the corresponding column-name-to-index association.

```
ms = DateValue[Table[DateObject[{2016, i, 1}], {i, 12}], "MonthName"];
dvals =
  Map[{ms⟦#⟧, "Q" <> ToString[Quotient[#, 4] + 1]} &, ToExpression@dvals⟦All, 1⟧];
dataM = MapThread[Join[#1, #2] &, {data, dvals}];
aColumnNamesM = Join[aColumnNames, <|"MonthName" → (Length[aColumnNames] + 1),
    "QuarterNameShort" → (Length[aColumnNames] + 2)|>];
```

Out[1002]= {0.054877, Null}

Here is the contingency table for “Country” vs “QuarterNameShort” over “Quantity”.

In[1026]:=
```
ctRes = CrossTabulate[
   dataM⟦All, aColumnNamesM /@ {"Country", "QuarterNameShort", "Quantity"}⟧];
Magnify[#, 0.75] &@MatrixForm[#1, TableHeadings → {#2, #3}] &@@ ctRes
```

Out[1027]=

	Q1	Q4
Australia	5534	454
Austria	0	3
Bahrain	-54	54
Belgium	675	1755
Channel Islands	0	80
Cyprus	144	917
Denmark	0	454
EIRE	7796	5381
Finland	0	1254
France	6581	4978
Germany	4495	6723
Iceland	0	319
Israel	100	-56
Italy	745	293
Japan	-45	4093
Lithuania	0	652
Netherlands	14 570	6811
Norway	0	3582
Poland	288	140
Portugal	1841	945
Spain	3091	867
Sweden	292	3714
Switzerland	2653	714
United Kingdom	157 046	298 101

Uniform tables

Often when making contingency tables over subsets of the data we obtain contingency tables with different rows and columns. For various reasons (programming, esthetics, comprehension) it is better to have the tables with the same rows and columns.

Here is an example of non-uniform contingency tables derived from the online retail data of the previous section. We split the data over the countries and find contingency tables of “MonthName” vs ”QuarterNameShort” over ”Quantity”.

In[985]:=
```
tbs = Association@Map[
    (xtab = CrossTabulate[
        #⟦All, aColumnNamesM /@ {"MonthName", "QuarterNameShort", "Quantity"}⟧];
      #⟦1, aColumnNamesM["Country"]⟧ → xtab) &,
    GatherBy[dataM, #⟦aColumnNamesM["Country"]⟧ &]];
```

```
Magnify[#, 0.75] &@Map[
  MatrixForm[#["Matrix"], TableHeadings → (# /@ {"RowNames", "ColumnNames"})] &, tbs]
```

Out[1015]=

United Kingdom →

	Q1	Q4
December	0	298 101
January	157 046	0

France →

	Q1	Q4
December	0	4978
January	6581	0

Australia →

	Q1	Q4
December	0	454
January	5534	0

Netherlands →

	Q1	Q4
December	0	6811
January	14 570	0

Germany →

	Q1	Q4
December	0	6723
January	4495	0

Norway →

	Q4
December	3582

EIRE →

	Q1	Q4
December	0	5381
January	7796	0

Switzerland →

	Q1	Q4
December	0	714
January	2653	0

Spain →

	Q1	Q4
December	0	867
January	3091	0

Poland →

	Q1	Q4
December	0	140
January	288	0

Portugal →

	Q1	Q4
December	0	945
January	1841	0

Italy →

	Q1	Q4
December	0	293
January	745	0

Belgium →

	Q1	Q4
December	0	1755
January	675	0

Lithuania →

	Q4
December	652

Japan →

	Q1	Q4
December	0	4093
January	-45	0

Iceland →

	Q4
December	319

Channel Islands →

	Q4
December	80

Denmark →

	Q4
December	454

Cyprus →

	Q1	Q4
December	0	917
January	144	0

Sweden →

	Q1	Q4
December	0	3714
January	292	0

Austria →

	Q4
December	3

Israel →

	Q1	Q4
December	0	-56
January	100	0

Finland →

	Q4
December	1254

Bahrain →

	Q1	Q4
December	0	54
January	-54	0

Using the object `RSparseMatrix`, see [4,5], we can impose row and column names on each table.

First we convert the contingency tables into `RSparseMatrix` objects:

In[1021]:=

```
tbs2 = Map[ToRSparseMatrix[#["Matrix"],
    "RowNames" → #["RowNames"], "ColumnNames" → #["ColumnNames"]] &, tbs];
```

And then we impose the desired row and column names:

In[1044]:=
```
tbs2 = Map[ImposeColumnNames[
    ImposeRowNames[#, {"January", "December"}], {"Q1", "Q2", "Q3", "Q4"}] &, tbs2];
Magnify[#, 0.75] &@ (MatrixForm /@ tbs2)
```

Out[1045]=

United Kingdom →

	Q1	Q2	Q3	Q4
January	157 046	0	0	0
December	0	0	0	298 101

France →

	Q1	Q2	Q3	Q4
January	6581	0	0	0
December	0	0	0	4978

Australia →

	Q1	Q2	Q3	Q4
January	5534	0	0	0
December	0	0	0	454

Netherlands →

	Q1	Q2	Q3	Q4
January	14 570	0	0	0
December	0	0	0	6811

Germany →

	Q1	Q2	Q3	Q4
January	4495	0	0	0
December	0	0	0	6723

Norway →

	Q1	Q2	Q3	Q4
January	0	0	0	0
December	0	0	0	3582

EIRE →

	Q1	Q2	Q3	Q4
January	7796	0	0	0
December	0	0	0	5381

Switzerland →

	Q1	Q2	Q3	Q4
January	2653	0	0	0
December	0	0	0	714

Spain →

	Q1	Q2	Q3	Q4
January	3091	0	0	0
December	0	0	0	867

Poland →

	Q1	Q2	Q3	Q4
January	288	0	0	0
December	0	0	0	140

Portugal →

	Q1	Q2	Q3	Q4
January	1841	0	0	0
December	0	0	0	945

Italy →

	Q1	Q2	Q3	Q4
January	745	0	0	0
December	0	0	0	293

Belgium →

	Q1	Q2	Q3	Q4
January	675	0	0	0
December	0	0	0	1755

Lithuania →

	Q1	Q2	Q3	Q4
January	0	0	0	0
December	0	0	0	652

Japan →

	Q1	Q2	Q3	Q4
January	-45	0	0	0
December	0	0	0	4093

Iceland →

	Q1	Q2	Q3	Q4
January	0	0	0	0
December	0	0	0	319

Channel Islands →

	Q1	Q2	Q3	Q4
January	0	0	0	0
December	0	0	0	80

Denmark →

	Q1	Q2	Q3	Q4
January	0	0	0	0
December	0	0	0	454

Cyprus →

	Q1	Q2	Q3	Q4
January	144	0	0	0
December	0	0	0	917

Sweden →

	Q1	Q2	Q3	Q4
January	292	0	0	0
December	0	0	0	3714

Austria →

	Q1	Q2	Q3	Q4
January	0	0	0	0
December	0	0	0	3

Israel →

	Q1	Q2	Q3	Q4
January	100	0	0	0
December	0	0	0	-56

Finland →

	Q1	Q2	Q3	Q4
January	0	0	0	0
December	0	0	0	1254

Bahrain →

	Q1	Q2	Q3	Q4
January	-54	0	0	0
December	0	0	0	54

References

[1] Anton Antonov, MathematicaForPrediction utilities, (2014), source code MathematicaForPrediction at GitHub, package MathematicaForPredictionUtilities.m.

[2] Anton Antonov, Mosaic plot for data visualization implementation in Mathematica, (2014), MathematicaForPrediction at GitHub, package MosaicPlot.m.

[3] Anton Antonov, "Mosaic plots for data visualization", (2014), MathematicaForPrediction at WordPress blog. URL: https://mathematicaforprediction.wordpress.com/2014/03/17/mosaic-plots-for-data-visualization/ .

[4] Anton Antonov, RSparseMatrix Mathematica package, (2015) MathematicaForPrediction at GitHub. URL: https://github.com/antononcube/MathematicaForPrediction/blob/master/Misc/RSparseMatrix.m .

[5] Anton Antonov, “RSparseMatrix for sparse matrices with named rows and columns”, (2015), MathematicaForPrediction at WordPress blog. URL: https://mathematicaforprediction.wordpress.com/2015/10/08/rsparsematrix-for-sparse-matrices-with-named-rows-and-columns/ .

[6] Daqing Chen, Online Retail Data Set, (2015), UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/ml/datasets/Online+Retail .