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].

```
In[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:

```
In[2]:= 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[4]:= titanicData = Flatten@*List @@@ ExampleData[{"MachineLearning", "Titanic"}, "Data"];
titanicData = DeleteCases[titanicData, {___, _Missing, ___}];
```

In[6]:=
```
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[8]:=
```
Dimensions[titanicData]
```

Out[8]=
```
{1046, 4}
```

In[9]:=
```
RecordsSummary[titanicData, titanicColumnNames]
```

Out[9]=

1 passenger class		2 passenger age		3 passenger sex		4 passenger survival	
3rd	501	Min	0.1667	male	658	died	619
1st	284	1st Qu	21.	female	388	survived	427
2nd	261	Median	28.				
		Mean	29.8811				
		3rd Qu	39.				
		Max	80.				

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[10]:=
```
ctCounts = CrossTabulate[titanicData⟦All,
     aTitanicColumnNames /@ {"passenger class", "passenger survival"}⟧];
MatrixForm[#1, TableHeadings → {#2, #3}] &@@ ctCounts
```

Out[11]//MatrixForm=

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

In[12]:=
```
ctTotalAge = CrossTabulate[titanicData⟦All, aTitanicColumnNames /@
      {"passenger class", "passenger survival", "passenger age"}⟧];
MatrixForm[#1, TableHeadings → {#2, #3}] &@@ ctTotalAge
```

Out[13]//MatrixForm=

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

In[14]:=
```
MatrixForm[ctTotalAge[[1]] / Normal[ctCounts[[1]]],
 TableHeadings → Values[Rest[ctTotalAge]]]
```

Out[14]//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[15]:=
```
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[15]= ⟨| 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[16]:=
```
Needs["RLink`"]
RLinkResourcesInstall[]
InstallR[]
```

Out[17]= {Paclet[RLinkRuntime, 9.0.0.0, <>]}

In[19]:=
```
ctCounts = FromRXTabsForm@xtabsViaRLink[titanicData[[All,
      aTitanicColumnNames /@ {"passenger class", "passenger survival"}]],
    {"passenger.sex", "passenger.survival"},
    " ~ passenger.sex + passenger.survival"];
MatrixForm[#1, TableHeadings → {#2, #3}] & @@ ctCounts
```

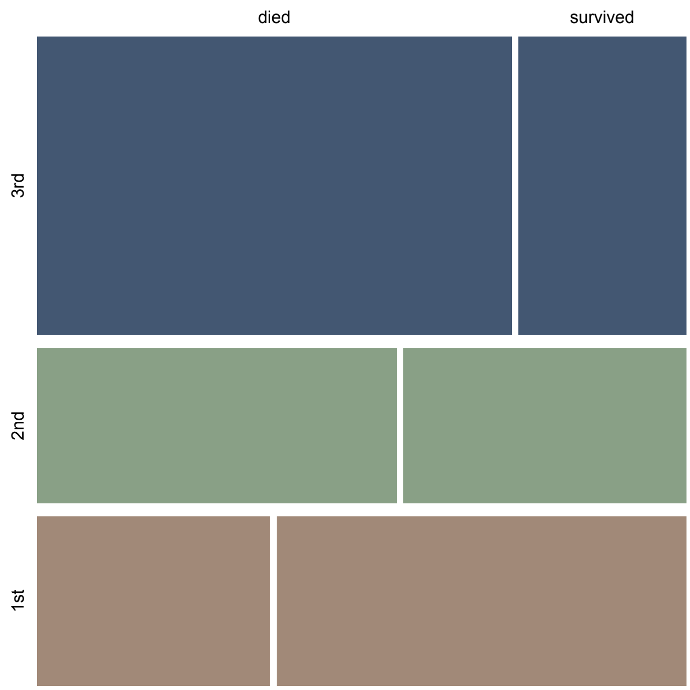
Out[20]//MatrixForm=

	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[21]:=
```
MosaicPlot[
 titanicData⟦All, aTitanicColumnNames /@ {"passenger class", "passenger survival"}⟧]
```

Out[21]=


Straightforward calling of MatrixForm

At this point we might want to be able to call `MatrixForm` more directly for the output of `CrossTabulate` and `FromRXTabsForm`. One way to do this is to define an up-value for `Association`.

In[22]:=
```
Unprotect[Association];
MatrixForm[
    x_Association /; (KeyExistsQ[x, "XTABMatrix"] || KeyExistsQ[x, "XTABTensor"])] ^:=
   (MatrixForm[#1, TableHeadings → Rest[{##}]] & @@ x);
Protect[Association];
```

Now we can do this:

In[25]:=
```
MatrixForm@CrossTabulate[titanicData[[All,
   aTitanicColumnNames /@ {"passenger class", "passenger survival"}]]]
```

Out[25]//MatrixForm=

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

Remark: Because of this up-value definition for `Association` with `MatrixForm` we have the associations returned by `CrossTabulate` and `FromRXTabsForm` to have the key "XTABMatrix" instead of "Matrix", the former assumed to be much more rarely to be used than the latter.

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[26]:=
```
data = Import["/Volumes/WhiteSlimSeagate/Datasets/UCI
     Online Retail Data Set/Online Retail.csv"];
columnNames = First[data];
data = Rest[data];
```

In[29]:=
```
aColumnNames = AssociationThread[columnNames → Range[Length[columnNames]]];
```

Data summary

We have ≈ 66 000 rows and 8 columns:

In[30]:=
```
Dimensions[data]
```

Out[30]=
```
{65 499, 8}
```

Here is a summary of the columns:

```
In[31]:= Magnify[#, 0.75] &@RecordsSummary[data, columnNames]
```

Out[31]=

1 InvoiceNo		2 StockCode		3 Description	
537434	675	85123A	353	WHITE HANGING HEART T-LIGHT HOLDER	358
538071	652	22423	279	REGENCY CAKESTAND 3 TIER	278
538349	620	22469	224	HEART OF WICKER SMALL	224
537638	601	22834	213	HAND WARMER BABUSHKA DESIGN	213
537237	597	22111	207	SCOTTIE DOG HOT WATER BOTTLE	207
536876	593	85099B	205	JUMBO BAG RED RETROSPOT	205
(Other)	61761	(Other)	64018	(Other)	64014

4 Quantity	
Min	-74215
1st Qu	1
Median	2
3rd Qu	8
Mean	$\frac{182660}{21833}$
Max	74215

5 InvoiceDate	
12/6/10 16:57	675
12/9/10 14:09	652
12/10/10 14:59	621
12/7/10 15:28	601
12/6/10 9:58	597
12/3/10 11:36	593
(Other)	61760

6 UnitPrice	
Min	0
1st Qu	1.25
Median	2.51
3rd Qu	4.24
Mean	5.85759
Max	16888.

7 CustomerID	
	25281
12748	695
17841	481
14606	421
15311	418
14911	377
(Other)	37826

8 Country	
United Kingdom	61186
Germany	982
France	967
EIRE	504
Spain	355
Portugal	212
(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[32]:= MinMax@Map[Length@*Union,
           GatherBy[data[[All, aColumnNames /@ {"Description", "StockCode"}]], First]]

Out[32]= {1, 144}
```

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

```
In[33]:= Tally[NumberQ /@ data[[All, aColumnNames["StockCode"]]]]

Out[33]= {{False, 9009}, {True, 56490}}
```

So let us convert it to all strings:

```
In[34]:= data[[All, aColumnNames["StockCode"]]] =
           ToString /@ data[[All, aColumnNames["StockCode"]]];
```

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

```
In[35]:= AbsoluteTiming[
          ctRes =
            CrossTabulate[data[[All, aColumnNames /@ {"Country", "StockCode", "Quantity"}]]];
         ]

Out[35]= {0.252337, Null}
```

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

In[36]:=
```
AbsoluteTiming[rres =
   xtabsViaRLink[data⟦All, aColumnNames /@ {"Country", "StockCode", "Quantity"}⟧,
    {"Country", "StockCode", "Quantity"}, "Quantity ~ Country + StockCode"];
  ctRRes = FromRXTabsForm[rres];
]
```

Out[36]= `{0.864462, Null}`

Both functions produce the same result:

In[37]:=
```
ctRRes["XTABMatrix"] == N@ctRRes[[1]]
```

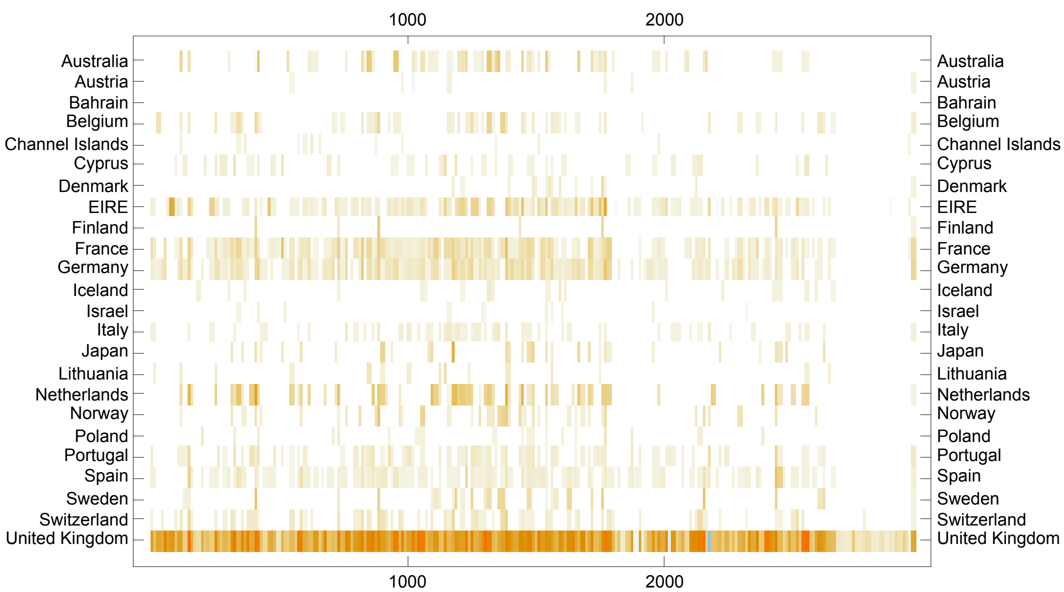
Out[37]= `True`

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

Here we plot the contingency table using `MatrixPlot` :

In[38]:=
```
MatrixPlot[ctRRes["XTABMatrix"], AspectRatio → 1 / 1.5, FrameTicks →
   {{#, #} &@Table[{i, ctRRes["RowNames"]⟦i⟧}, {i, Length[ctRRes["RowNames"]]}],
    {Automatic, Automatic}}, ImageSize → 550]
```

Out[38]=



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.

In[39]:=
```
(*AbsoluteTiming[
 dobjs=DateObject[{#,{"Month","/","Day","/","Year"," ","Hour",":","Minute"}}]&/@
    data⟦All,aColumnNames["InvoiceDate"]⟧;
]*)
```

In[40]:=
```
(*AbsoluteTiming[
 dvals=DateValue[dobjs,{"MonthName","QuarterNameShort"}];
]*)
```

We can use the following ad hoc computation instead.

In[41]:=
```
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[42]:=
```
Magnify[#, 0.75] &@RecordsSummary[dvals[[All, 1 ;; 3]], "MaxTallies" → 16]
```

Out[42]=

1 column 1	
12	42 481
1	23 018

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)	10 212

3 column 3	
10	42 481
11	23 018

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

In[43]:=
```
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)|>];
```

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

In[47]:=
```
ctRes = CrossTabulate[
   dataM⟦All, aColumnNamesM /@ {"Country", "QuarterNameShort", "Quantity"}⟧];
Magnify[#, 0.75] &@MatrixForm[#1, TableHeadings → {#2, #3}] &@@ ctRes
```

Out[48]=

	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[49]:=
```
tbs = Association@Map[
    (xtab = CrossTabulate[
        #⟦All, aColumnNamesM /@ {"MonthName", "QuarterNameShort", "Quantity"}⟧];
      #⟦1, aColumnNamesM["Country"]⟧ → xtab) &,
    GatherBy[dataM, #⟦aColumnNamesM["Country"]⟧ &]];
```

In[50]:=
```
Magnify[#, 0.75] &@Map[MatrixForm[#] &, tbs]
```

Out[50]=

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[51]:=
```
tbs2 = Map[ToRSparseMatrix[#["XTABMatrix"],
    "RowNames" → #["RowNames"], "ColumnNames" → #["ColumnNames"]] &, tbs];
```

And then we impose the desired row and column names:

In[52]:=
```
tbs2 = Map[ImposeColumnNames[
    ImposeRowNames[#, {"January", "December"}], {"Q1", "Q2", "Q3", "Q4"}] &, tbs2];
Magnify[#, 0.75] &@ (MatrixForm /@ tbs2)
```

Out[53]= ⟨|

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

|⟩

Generalization : CrossTensorate

A generalization of `CrossTabulate` is the function `CrossTensorate` implemented in [1] that takes a “formula” argument similar to R’s xtabs.

This finds number of people of different sub-groups of Titanic data:

In[54]:=
```
ctRes =
  CrossTensorate[Count == "passenger survival" + "passenger sex" + "passenger class",
   titanicData, aTitanicColumnNames];
MatrixForm[
 ctRes]
```

Out[55]//MatrixForm=

	female		male	
died	1st	5	1st	98
	2nd	11	2nd	135
	3rd	80	3rd	290
survived	1st	128	1st	53
	2nd	92	2nd	23
	3rd	72	3rd	59

We can verify the result using `Count`:

In[56]:=
```
Count[titanicData, {"1st", _, "female", "died"}]
```

Out[56]= 5

In[57]:=
```
Count[titanicData, {"2nd", _, "male", "survived"}]
```

Out[57]= 23

To split the cross-tensor across its first variable we can use this command:

In[58]:=
```
sctRes =
  Association@MapThread[Rule[#1, Join[<|"XTABTensor" → #2|>, Rest@Rest@ctRes]] &,
    {ctRes[[2]], # & /@ ctRes["XTABTensor"]}];
MatrixForm /@
 sctRes
```

Out[59]= ⟨| died →

	1st	2nd	3rd
female	5	11	80
male	98	135	290

, survived →

	1st	2nd	3rd
female	128	92	72
male	53	23	59

|⟩

Or we can call the more general function `CrossTensorateSplit` implemented in [1]:

In[66]:=
```
Map[MatrixForm /@ CrossTensorateSplit[ctRes, #] &, Rest@Keys[ctRes]]
```

Out[66]= {⟨| died →

	1st	2nd	3rd
female	5	11	80
male	98	135	290

, survived →

	1st	2nd	3rd
female	128	92	72
male	53	23	59

|⟩,

⟨| female →

	1st	2nd	3rd
died	5	11	80
survived	128	92	72

, male →

	1st	2nd	3rd
died	98	135	290
survived	53	23	59

|⟩,

⟨| 1st →

	female	male
died	5	98
survived	128	53

,

2nd →

	female	male
died	11	135
survived	92	23

, 3rd →

	female	male
died	80	290
survived	72	59

|⟩}

`CrossTensorateSplit` can also be called with one argument that is a variable name.This will produce a splitting function. For example, the above command can be re-written as :

In[67]:=
```
Map[MatrixForm /@ CrossTensorateSplit[#] @ctRes &, Rest@Keys[ctRes]]
```

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 .