RSparseMatrix

Anton Antonov
MathematicaForPrediction at GitHub
MathematicaForPrediction blog at WordPress.com
May 2015
October 2015

Introduction

This notebook has the function implementations for manipulating objects with head `RSparseMatrix` that behave like `SparseArray` objects but have the added functionalities to use row names and column names in a manner similar to that of the sparse arrays objects from the base library Matrix [2] for the programming language R [1].

The idea is fairly simple: we can use associations or replacement rules to map row names and column names into integers. Similarly to how it is done in R, `RSparseMatrix` handles only strings as row names and column names.

The following function signatures are implemented:

```
RowNames[_RSparseMatrix]
ColumnNames[_RSparseMatrix]
DimensionNames[_RSparseMatrix]
Dimensions[_RSparseMatrix]
RowsCount[_RSparseMatrix]
ColumnsCount[_RSparseMatrix]
RowSum[_RSparseMatrix]
ColumnSum[_RSparseMatrix]
Total[_RSparseMatrix,___]
ArrayRules[_RSparseMatrix]
Transpose[_RSparseMatrix]
MatrixForm[_RSparseMatrix]
MatrixPlot[_RSparseMatrix]
Dot[_RSparseMatrix, _RSparseMatrix]
Dot[_,_RSparseMatrix]
Dot[_RSparseMatrix, _]
Part[_RSparseMatrix, _String | {_String ..},___]
Part[_RSparseMatrix, _,_String | {_String ..}]
Part[_RSparseMatrix, _String | {_String ..}, _String | {_String ..}]
RowBind[_RSparseMatrix,_RSparseMatrix]
ColumnBind[_RSparseMatrix,_RSparseMatrix]
```

Note that assignment (with `Set[__]`) is not implemented.

The package can be loaded from GitHub [3]:

In[1]:=
```
Import[
 "https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/Misc/
   RSparseMatrix.m"]
```

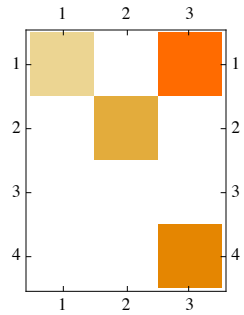
Implementation (and explanations)

Tests and experiments

SparseArrays to compare with

In[2]:=
```
mat = SparseArray[{{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 3} → 4}];
MatrixPlot[mat]
```

Out[3]=



In[4]:=
```
Grid[{{MatrixForm[mat], MatrixForm[Join[mat, mat]],
   MatrixForm[Transpose@Join[Transpose[mat], Transpose[mat]]]}}]
```

Out[4]=

$$\begin{pmatrix} 1 & 0 & 4 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 4 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \\ 1 & 0 & 4 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 4 & 1 & 0 & 4 \\ 0 & 2 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 3 \end{pmatrix}$$

Creation

In[5]:=
```
rmat = MakeRSparseMatrix[
   {{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 4} → 4}, "ColumnNames" → {"a", "b", "c"},
   "RowNames" → {"A", "B", "C"}, "DimensionNames" → {"U", "V"}]
```

Out[5]=
```
$Failed
```

In[6]:=
```
rmat = MakeRSparseMatrix[
   {{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 4} → 4}, "ColumnNames" → {"a", "b", "c", "d"},
   "RowNames" → {"A", "B", "C", "D"}, "DimensionNames" → {"U", "V"}];
```

In[7]:=
```
rmat // MatrixForm
```

Out[7]//MatrixForm=

	a	b	c	d
A	1	0	0	4
B	0	2	0	0
C	0	0	0	0
D	0	0	3	0

In[8]:=
```
SparseArray[rmat] // MatrixForm
```

Out[8]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{pmatrix}$$

In[9]:=
```
SparseArray[rmat[[All, RotateLeft@ColumnNames[rmat]]] // MatrixForm
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4 & 1 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

Query methods

In[10]:=
```
RowNames[rmat]
ColumnNames[rmat]
DimensionNames[rmat]
```

Out[10]= {A, B, C, D}

Out[11]= {a, b, c, d}

Out[12]= {U, V}

In[13]:=
```
Dimensions[rmat]
```

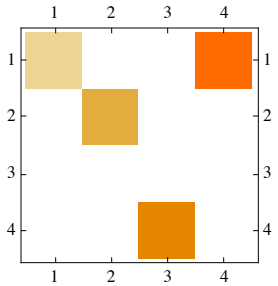
Out[13]= {4, 4}

In[14]:=
```
ArrayRules[rmat]
```

Out[14]= {{1, 1} → 1, {1, 4} → 4, {2, 2} → 2, {4, 3} → 3, {_, _} → 0}

In[15]:=
```
MatrixPlot[rmat]
```

Out[15]=



In[16]:=
```
MatrixForm[rmat]
```

Out[16]//MatrixForm=

	a	b	c	d
A	1	0	0	4
B	0	2	0	0
C	0	0	0	0
D	0	0	3	0

Transpose

In[17]:=
```
MatrixForm[Transpose[rmat]]
```

Out[17]//MatrixForm=

	A	B	C	D
a	1	0	0	0
b	0	2	0	0
c	0	0	0	3
d	4	0	0	0

In[18]:=
```
DimensionNames[Transpose[rmat]]
```

Out[18]= {V, U}

Sums

In[19]:=
```
Total[rmat, 2]
```

Out[19]= 10

In[20]:=
```
RowSum[rmat]
```

Out[20]= {1, 2, 3, 4}

In[21]:=
```
ColumnSum[rmat]
```

Out[21]= {5, 2, 0, 3}

■ Dot product

In[22]:=
```
MatrixForm[rmat.mat]
```

Out[22]//MatrixForm=

A	1	0	16
B	0	4	0
C	0	0	0
D	0	0	0

In[23]:=
```
MatrixForm[Transpose[mat].rmat]
```

Out[23]//MatrixForm=

	a	b	c	d
1	0	0	4	
0	4	0	0	
4	0	9	16	

In[24]:=
```
Grid[{{MatrixForm[rmat.rmat], MatrixForm[rmat.Transpose[rmat]]}}]
```

Out[24]=

	a	b	c	d
A	1	0	12	4
B	0	4	0	0
C	0	0	0	0
D	0	0	0	0

	A	B	C	D
A	17	0	0	0
B	0	4	0	0
C	0	0	0	0
D	0	0	0	9

In[25]:=
```
MatrixForm[(rmat.rmat).rmat]
```

Out[25]//MatrixForm=

	a	b	c	d
A	1	0	12	4
B	0	8	0	0
C	0	0	0	0
D	0	0	0	0

In[26]:=
```
Grid[{{MatrixForm[rmat], MatrixForm[rmat.rmat.rmat]}}]
```

Out[26]=

	a	b	c	d
A	1	0	0	4
B	0	2	0	0
C	0	0	0	0
D	0	0	3	0

	a	b	c	d
A	1	0	12	4
B	0	8	0	0
C	0	0	0	0
D	0	0	0	0

Here Associations “swallows” the second value “U” because they are the same.

In[27]:=
```
DimensionNames[rmat.Transpose[rmat]]
```

Out[27]= {U}

■ Arithmetic operations

In[28]:=
```
MatrixForm[rmat + 1]
```

Out[28]//MatrixForm=

	a	b	c	d
A	2	1	1	5
B	1	3	1	1
C	1	1	1	1
D	1	1	4	1

In[29]:=
```
MatrixForm[rmat - 1]
```

Out[29]//MatrixForm=

	a	b	c	d
A	0	−1	−1	3
B	−1	1	−1	−1
C	−1	−1	−1	−1
D	−1	−1	2	−1

In[30]:=
```
MatrixForm[rmat * 2.33]
```

Out[30]//MatrixForm=

	a	b	c	d
A	2.33	0.	0.	9.32
B	0.	4.66	0.	0.
C	0.	0.	0.	0.
D	0.	0.	6.99	0.

In[31]:=
```
Grid[{MatrixForm /@ {rmat, rmat + 2, rmat * 10, 100 * rmat + 2.33 * rmat}}]
```

Out[31]=

	a	b	c	d
A	1	0	0	4
B	0	2	0	0
C	0	0	0	0
D	0	0	3	0

	a	b	c	d
A	3	2	2	6
B	2	4	2	2
C	2	2	2	2
D	2	2	5	2

	a	b	c	d
A	10	0	0	40
B	0	20	0	0
C	0	0	0	0
D	0	0	30	0

	a	b	c	d
A	102.33	0.	0.	409.32
B	0.	204.66	0.	0.
C	0.	0.	0.	0.
D	0.	0.	306.99	0.

In[32]:=
```
Grid[{MatrixForm /@ {rmat, Transpose[rmat], rmat + Transpose[rmat], rmat * Transpose[rmat]}}]
```

Out[32]=

	a	b	c	d
A	1	0	0	4
B	0	2	0	0
C	0	0	0	0
D	0	0	3	0

	A	B	C	D
a	1	0	0	0
b	0	2	0	0
c	0	0	0	3
d	4	0	0	0

2	0	0	4
0	4	0	0
0	0	0	3
4	0	3	0

1	0	0	0
0	4	0	0
0	0	0	0
0	0	0	0

■ Part

In[33]:=
```
MatrixForm[rmat[[{"A"}]]]
```

Out[33]//MatrixForm=

	a	b	c	d
A	1	0	0	4

In[34]:=
```
MatrixForm[rmat[[{"A", "B"}, {"a", "c", "d"}]]]
```

Out[34]//MatrixForm=

	a	c	d
A	1	0	4
B	0	0	0

In[35]:=
```
MatrixForm[rmat[[2 ;; 3, 1 ;; 2]]]
```

Out[35]//MatrixForm=

	a	b
B	0	2
C	0	0

In[36]:=
```
MatrixForm[rmat[[{"A", "B"}]]]
```

Out[36]//MatrixForm=

	a	b	c	d
A	1	0	0	4
B	0	2	0	0

In[37]:=
```
MatrixForm[rmat[[{"A", "B"}, 1 ;; 2]]]
```

Out[37]//MatrixForm=

	a	b
A	1	0
B	0	2

In[38]:=
```
MatrixForm[rmat[[All, {"a", "c"}]]]
```

Out[38]//MatrixForm=

	a	c
A	1	0
B	0	0
C	0	0
D	0	3

■ RowBind, ColumnBind

In[39]:=
```
MatrixForm[RowBind[rmat, rmat]]
```

Out[39]//MatrixForm=

	a	b	c	d
A.1	1	0	0	4
B.1	0	2	0	0
C.1	0	0	0	0
D.1	0	0	3	0
A.2	1	0	0	4
B.2	0	2	0	0
C.2	0	0	0	0
D.2	0	0	3	0

In[40]:=
```
rmat2 = ToRSparseMatrix[rmat, "RowNames" → Map["s." <> # &, RowNames[rmat]]];
MatrixForm[rmat2]
```

Out[41]//MatrixForm=

	a	b	c	d
s.A	1	0	0	4
s.B	0	2	0	0
s.C	0	0	0	0
s.D	0	0	3	0

In[42]:=
```
rmat3 = ToRSparseMatrix[rmat, "ColumnNames" → Map["t." <> # &, ColumnNames[rmat]]];
MatrixForm[rmat3]
```

Out[43]//MatrixForm=

	t.a	t.b	t.c	t.d
A	1	0	0	4
B	0	2	0	0
C	0	0	0	0
D	0	0	3	0

In[44]:=
```
MatrixForm[RowBind[rmat, rmat2]]
```

Out[44]//MatrixForm=

	a	b	c	d
A	1	0	0	4
B	0	2	0	0
C	0	0	0	0
D	0	0	3	0
s.A	1	0	0	4
s.B	0	2	0	0
s.C	0	0	0	0
s.D	0	0	3	0

In[45]:=
```
{MatrixForm[ColumnBind[rmat, rmat2]], MatrixForm[ColumnBind[rmat, rmat3]]}
```

Out[45]=

	a.1	b.1	c.1	d.1	a.2	b.2	c.2	d.2
A	1	0	0	4	1	0	0	4
B	0	2	0	0	0	2	0	0
C	0	0	0	0	0	0	0	0
D	0	0	3	0	0	0	3	0

,

	a	b	c	d	t.a	t.b	t.c	t.d
A	1	0	0	4	1	0	0	4
B	0	2	0	0	0	2	0	0
C	0	0	0	0	0	0	0	0
D	0	0	3	0	0	0	3	0

Profiling

In[46]:=
```
smat = SparseArray[RandomReal[{0, 1}, {1000, 120}]];
```

In[47]:=
```
rmat =
  ToRSparseMatrix[smat, "RowNames" → Map["A" <> ToString[#] &, Range[Dimensions[smat][[1]]]],
   "ColumnNames" → Map["b" <> ToString[#] &, Range[Dimensions[smat][[2]]]]];
```

In[48]:=
```
n = 100;
tres =
 AbsoluteTiming[
  Do[sres = smat.Transpose[smat], {i, n}]
 ]
tres[[1]] / n
```

Out[49]= {5.33297, Null}

Out[50]= 0.0533297

In[51]:=
```
tres =
 AbsoluteTiming[
  Do[rres = rmat.Transpose[rmat], {i, n}]
 ]
tres[[1]] / n
```

Out[51]= {5.3034, Null}

Out[52]= 0.053034

In[53]:=
```
Norm[sres[[1 ;; 120, 1 ;; 120]] - SparseArray[rres[[1 ;; 120, 1 ;; 120]]]
```

Out[53]= 0.

In[54]:=
```
Grid[{{
   MatrixPlot[rres[[1 ;; 120, 1 ;; 120]], ImageSize → 350],
   MatrixPlot[rres[["A" <> ToString[#] & /@ Range[120], 1 ;; 120]], ImageSize → 350]}}]
```

References

[1] The R Core Team, R Language Definition, (2015).
URL: https://cran.r-project.org/doc/manuals/r-release/R-lang.pdf

[2] D. Bates, M. Maechler, Sparse and Dense Matrix Classes and Methods, Package 'Matrix', (2015).
URL: https://cran.r-project.org/web/packages/Matrix/Matrix.pdf.

[3] A. Antonov, RSparseMatrix *Mathematica* packages, *Mathematica*ForPrediction project at GitHub, (2015).
URL: https://github.com/antononcube/MathematicaForPrediction/blob/master/Misc/RSparseMatrix.m .