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.

Note that the package does not use RLink -- it has purely *Mathematica* language implementations.

The following function signatures are implemented:

```
RowNames[_RSparseMatrix]
ColumnNames[_RSparseMatrix]
SetRowNames[_RSparseMatrix, {_String..}]
SetColumnNames[_RSparseMatrix, {_String..}]
DimensionNames[_RSparseMatrix]
Dimensions[_RSparseMatrix]
RowsCount[_RSparseMatrix]
ColumnsCount[_RSparseMatrix]
RowSums[_RSparseMatrix]
ColumnSums[_RSparseMatrix]
Total[_RSparseMatrix,___]
ArrayRules[_RSparseMatrix]
Transpose[_RSparseMatrix]
MatrixForm[_RSparseMatrix]
MatrixPlot[_RSparseMatrix]
Times[_RSparseMatrix, _RSparseMatrix]
Times[_,_RSparseMatrix]
Times[_RSparseMatrix, _]
Plus[_RSparseMatrix, _RSparseMatrix]
Plus[_,_RSparseMatrix]
Plus[_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"]
```

In[2]:=
```
(*Import["~/MathFiles/MathematicaForPrediction/Misc/RSparseMatrix.m"]*)
```

Exposition functions

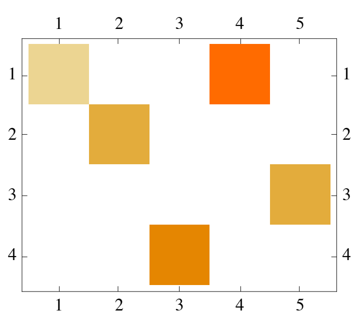
In[3]:=
```
Clear[ResultsGrid]
ResultsGrid[expressions_Inactive, opts___] :=
  Block[{t, t1},
   t = Map[HoldForm, expressions, {2}];
   t1 = ReleaseHold[Activate[t]];
   Grid[MapThread[Prepend,
     {{Activate[t], MatrixForm /@ t1, Head /@ t1}, Style[#, Blue, FontFamily → "Times"] & /@ {"expr", "result", "head"}}], opts]
  ];
```

Tests and experiments

▪ SparseArrays for comparisons

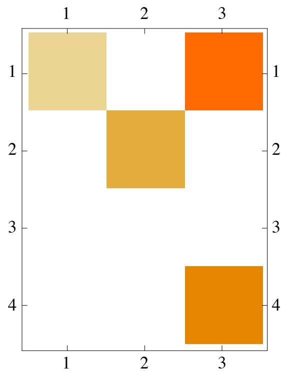
In[84]:=
```
mat = SparseArray[{{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 4} → 4, {3, 5} → 2}];
MatrixPlot[mat]
```

Out[85]=



In[86]:=
```
mat2 = SparseArray[{{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 3} → 4}];
MatrixPlot[mat2]
```

Out[87]=



This illustrates row binding and column binding corresponding to R’s function `cbind` and `rbind`. Here it is done over sparse arrays below it is done with `RSparseMatrix` objects.

In[88]:=
```
Grid[{{MatrixForm[mat], MatrixForm[Join[mat, mat]], MatrixForm[Transpose@Join[Transpose[mat], Transpose[mat]]]}}]
```

Out[88]=

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

■ Creation

In[89]:=
```
rmat = MakeRSparseMatrix[{{1, 1} → 1, {2, 2} → 2, {3, 3} → 3, {1, 4} → 4},
  "ColumnNames" → {"a", "b", "c"}, "RowNames" → {"A", "B", "C"}, "DimensionNames" → {"U", "V"}]
```

Out[89]=
```
$Failed
```

In[90]:=
```
rmat = MakeRSparseMatrix[
  {{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 4} → 4, {3, 5} → 2},
  "ColumnNames" → {"a", "b", "c", "d", "e"},
  "RowNames" → {"A", "B", "C", "D"},
  "DimensionNames" → {"U", "V"}]
```

Out[90]=
```
RSparseMatrix[⟨|sparseArray → SparseArray[ Specifiedelements 5
                                           Dimensions {4, 5} ],
  rownames → <|A → 1, B → 2, C → 3, D → 4|>, colnames → <|a → 1, b → 2, c → 3, d → 4, e → 5|>, dimnames → <|U → 1, V → 2|> |⟩]
```

Note that the output is for mated to look like sparse array object. The package has this definition:

```
Format[RSparseMatrix[obj_]] := obj["sparseArray"];
```

The function `MatrixForm` shows the `RSparseMatrix` objects with their row and column names:

In[91]:=
```
rmat // MatrixForm
```

Out[91]//MatrixForm=

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

In[92]:=
```
rmat = ToRSparseMatrix[SparseArray[rmat], "ColumnNames" → {"a", "b", "c", "d", "e"},
  "RowNames" → {"A", "B", "C", "D"}, "DimensionNames" → {"U", "V"}]
```

Out[92]=
```
RSparseMatrix[⟨|sparseArray → SparseArray[ Specifiedelements 5
                                           Dimensions {4, 5} ],
  rownames → <|A → 1, B → 2, C → 3, D → 4|>, colnames → <|a → 1, b → 2, c → 3, d → 4, e → 5|>, dimnames → <|U → 1, V → 2|> |⟩]
```

In[93]:=
```
rmat // MatrixForm
```

Out[93]//MatrixForm=

$$\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}$$

■ Setting names

In[94]:=
```
rmat2 = rmat
```

Out[94]=
```
RSparseMatrix[⟨|sparseArray → SparseArray[ Specifiedelements 5
                                           Dimensions {4, 5} ],
  rownames → <|A → 1, B → 2, C → 3, D → 4|>, colnames → <|a → 1, b → 2, c → 3, d → 4, e → 5|>, dimnames → <|U → 1, V → 2|>|⟩]
```

In[95]:=
```
MatrixForm[rmat2]
```

Out[95]//MatrixForm=

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

In[96]:=
```
SetRowNames[rmat2, ToString /@ Range[RowsCount[rmat]]]
```

Out[96]=
```
RSparseMatrix[⟨|sparseArray → SparseArray[ Specifiedelements 5
                                           Dimensions {4, 5} ],
  rownames → <|1 → 1, 2 → 2, 3 → 3, 4 → 4|>, colnames → <|a → 1, b → 2, c → 3, d → 4, e → 5|>, dimnames → <|U → 1, V → 2|>|⟩]
```

In[97]:=
```
MatrixForm[rmat2]
```

Out[97]//MatrixForm=

	a	b	c	d	e
1	1	0	0	4	0
2	0	2	0	0	0
3	0	0	0	0	2
4	0	0	3	0	0

In[98]:=
```
SetColumnNames[rmat2, ToString /@ Range[ColumnsCount[rmat]]]
```

Out[98]=
```
RSparseMatrix[⟨|sparseArray → SparseArray[ Specifiedelements 5
                                           Dimensions {4, 5} ],
  rownames → <|1 → 1, 2 → 2, 3 → 3, 4 → 4|>, colnames → <|1 → 1, 2 → 2, 3 → 3, 4 → 4, 5 → 5|>, dimnames → <|U → 1, V → 2|>|⟩]
```

In[99]:= **MatrixForm[rmat2]**

Out[99]//MatrixForm=

	1	2	3	4	5
1	1	0	0	4	0
2	0	2	0	0	0
3	0	0	0	0	2
4	0	0	3	0	0

In[100]:= **MatrixForm[rmat]**

Out[100]//MatrixForm=

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

■ Query functions

These functions can be used to names of rows, columns, and dimensions. They correspond to R's functions `rownames`, `colnames`, `dimnames`.

```
In[101]:= RowNames[rmat]
          ColumnNames[rmat]
          DimensionNames[rmat]

Out[101]= {A, B, C, D}

Out[102]= {a, b, c, d, e}

Out[103]= {U, V}
```

Functions that can be applied to sparse arrays follow.

```
In[104]:= Dimensions[rmat]

Out[104]= {4, 5}

In[105]:= ArrayRules[rmat]

Out[105]= {{1, 1} → 1, {1, 4} → 4, {2, 2} → 2, {3, 5} → 2, {4, 3} → 3, {_, _} → 0}
```

■ Visualization

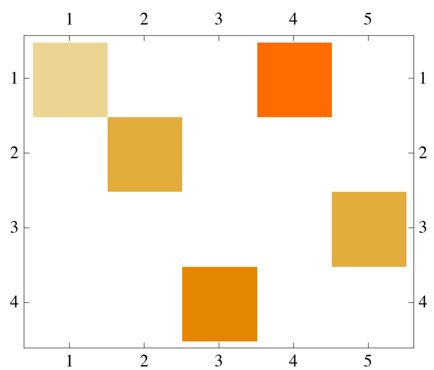
The redefinitions of `MatrixForm` and `MatrixPlot` are very useful for visualizing the `RSparseMatrix` objects.

In[106]:=
```
MatrixForm[rmat]
```

Out[106]//MatrixForm=

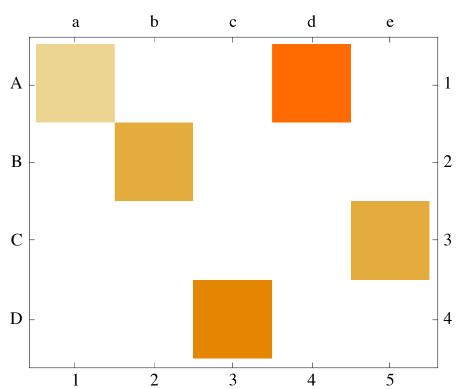
	a	b	c	d	e
A	1	0	0	4	0
B	0	2	0	0	0
C	0	0	0	0	2
D	0	0	3	0	0

In[107]:=
```
MatrixPlot[rmat]
```

Out[107]=


Here is another version of the `MatrixPlot` call that uses the row and column names as ticks.

In[108]:=
```
MatrixPlot[rmat, FrameTicks → {{Transpose[{Range[RowsCount[rmat]], RowNames[rmat]}], All},
   {All, Transpose[{Range[ColumnsCount[rmat]], ColumnNames[rmat]}]}}]
```

Out[108]=


■ Transpose

In[109]:= `MatrixForm[Transpose[rmat]]`

Out[109]//MatrixForm=

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline a & 1 & 0 & 0 & 0 \\ b & 0 & 2 & 0 & 0 \\ c & 0 & 0 & 0 & 3 \\ d & 4 & 0 & 0 & 0 \\ e & 0 & 0 & 2 & 0 \end{array}$$

In[110]:= `DimensionNames[Transpose[rmat]]`

Out[110]= {V, U}

■ Sums

In[111]:= `Total[rmat, 2]`

Out[111]= 12

In[112]:= `RowSums[rmat]`

Out[112]= {1, 2, 3, 4, 2}

In[113]:= `ColumnSums[rmat]`

Out[113]= {5, 2, 2, 3}

Dot product

Matrix by vector

Out[114]=

<table>
<tr><td>expr</td><td>rmat</td><td>Transpose[rmat⟦{1}, All⟧]</td><td>rmat.Transpose[rmat⟦{1}, All⟧]</td></tr>
<tr><td>result</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|c} & A \\ \hline a & 1 \\ b & 0 \\ c & 0 \\ d & 4 \\ e & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|c} & A \\ \hline A & 17 \\ B & 0 \\ C & 0 \\ D & 0 \end{array}\right)$</td></tr>
<tr><td>head</td><td>RSparseMatrix</td><td>RSparseMatrix</td><td>RSparseMatrix</td></tr>
</table>

Out[116]=

<table>
<tr><td>expr</td><td>rmat</td><td>rmat⟦1, All⟧</td><td>rmat.rmat⟦1, All⟧</td></tr>
<tr><td>result</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 4 \\ 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|c} A & 17 \\ B & 0 \\ C & 0 \\ D & 0 \end{array}\right)$</td></tr>
<tr><td>head</td><td>RSparseMatrix</td><td>SparseArray</td><td>RSparseMatrix</td></tr>
</table>

Matrix by matrix

First we look into a dot product to the right of `_RSparseMatrix` with a sparse array and a dot product to the left of `_RSparseMatrix` with a sparse array.

Out[118]=

<table>
<tr><td>expr</td><td>rmat</td><td>mat</td><td>rmat.Transpose[mat]</td><td>Transpose[mat].rmat</td></tr>
<tr><td>result</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{ccccc} 1 & 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|cccc} A & 17 & 0 & 0 & 0 \\ B & 0 & 4 & 0 & 0 \\ C & 0 & 0 & 4 & 0 \\ D & 0 & 0 & 0 & 9 \end{array}\right)$</td><td>$\left(\begin{array}{ccccc} a & b & c & d & e \\ \hline 1 & 0 & 0 & 4 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 4 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{array}\right)$</td></tr>
<tr><td>head</td><td>RSparseMatrix</td><td>SparseArray</td><td>RSparseMatrix</td><td>RSparseMatrix</td></tr>
</table>

This creates another `RSparseMatrix` object with no row and column names:

In[120]:=
```
rmat2 = ToRSparseMatrix[SparseArray[RandomInteger[{0, 4}, {ColumnsCount[rmat], RowsCount[rmat]}]]];
```

Next we look into two dot products of two `RSparseMatrix` objects.

Out[121]=

<table>
<tr><td>expr</td><td>rmat</td><td>rmat2</td><td>rmat.rmat2</td><td>rmat2.rmat</td><td>rmat.rmat2.rmat</td></tr>
<tr><td>result</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{cccc} 1 & 0 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 3 & 4 & 0 & 1 \\ 1 & 2 & 2 & 3 \\ 1 & 0 & 2 & 4 \end{array}\right)$</td><td>$\left(\begin{array}{c|cccc} A & 5 & 8 & 10 & 13 \\ B & 4 & 4 & 4 & 2 \\ C & 2 & 0 & 4 & 8 \\ D & 9 & 12 & 0 & 3 \end{array}\right)$</td><td>$\left(\begin{array}{ccccc} a & b & c & d & e \\ \hline 1 & 0 & 3 & 4 & 4 \\ 2 & 4 & 3 & 8 & 4 \\ 3 & 8 & 3 & 12 & 0 \\ 1 & 4 & 9 & 4 & 4 \\ 1 & 0 & 12 & 4 & 4 \end{array}\right)$</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 5 & 16 & 39 & 20 & 20 \\ B & 4 & 8 & 6 & 16 & 8 \\ C & 2 & 0 & 24 & 8 & 8 \\ D & 9 & 24 & 9 & 36 & 0 \end{array}\right)$</td></tr>
<tr><td>head</td><td>RSparseMatrix</td><td>RSparseMatrix</td><td>RSparseMatrix</td><td>RSparseMatrix</td><td>RSparseMatrix</td></tr>
</table>

Here Associations “swallows” the second value “U” because they are the same. (This is a bug.)

In[123]:=
```
DimensionNames[rmat.Transpose[rmat]]
```

Out[123]=
```
{U}
```

Arithmetic operations

The first three tables in this sub-section should be self-explanatory.

Out[124]=

expr	rmat + 1	rmat − 2	rmat 10	10 rmat + 2.33 rmat
result	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 2 & 1 & 1 & 5 & 1 \\ B & 1 & 3 & 1 & 1 & 1 \\ C & 1 & 1 & 1 & 1 & 3 \\ D & 1 & 1 & 4 & 1 & 1 \end{array}\right)$	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & -1 & -2 & -2 & 2 & -2 \\ B & -2 & 0 & -2 & -2 & -2 \\ C & -2 & -2 & -2 & -2 & 0 \\ D & -2 & -2 & 1 & -2 & -2 \end{array}\right)$	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 10 & 0 & 0 & 40 & 0 \\ B & 0 & 20 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 20 \\ D & 0 & 0 & 30 & 0 & 0 \end{array}\right)$	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 12.33 & 0. & 0. & 49.32 & 0. \\ B & 0. & 24.66 & 0. & 0. & 0. \\ C & 0. & 0. & 0. & 0. & 24.66 \\ D & 0. & 0. & 36.99 & 0. & 0. \end{array}\right)$
head	RSparseMatrix	RSparseMatrix	RSparseMatrix	RSparseMatrix

Out[126]=

expr	rmat	Transpose[rmat2]	rmat + Transpose[rmat2]	rmat Transpose[rmat2]
result	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$	$\begin{pmatrix} 1 & 2 & 3 & 1 & 1 \\ 0 & 2 & 4 & 2 & 0 \\ 2 & 2 & 0 & 2 & 2 \\ 1 & 1 & 1 & 3 & 4 \end{pmatrix}$	$\begin{pmatrix} 2 & 2 & 3 & 5 & 1 \\ 0 & 4 & 4 & 2 & 0 \\ 2 & 2 & 0 & 2 & 4 \\ 1 & 1 & 4 & 3 & 4 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 4 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 \end{pmatrix}$
head	RSparseMatrix	RSparseMatrix	RSparseMatrix	RSparseMatrix

In[128]:=

```
rmat3 = rmat2;
```

In[129]:=

```
SetRowNames[rmat3, ColumnNames[rmat]];
SetColumnNames[rmat3, RowNames[rmat]];
```

Out[131]=

expr	rmat	Transpose[rmat3]	rmat + Transpose[rmat3]	rmat Transpose[rmat3]
result	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 2 & 3 & 1 & 1 \\ B & 0 & 2 & 4 & 2 & 0 \\ C & 2 & 2 & 0 & 2 & 2 \\ D & 1 & 1 & 1 & 3 & 4 \end{array}\right)$	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 2 & 2 & 3 & 5 & 1 \\ B & 0 & 4 & 4 & 2 & 0 \\ C & 2 & 2 & 0 & 2 & 4 \\ D & 1 & 1 & 4 & 3 & 4 \end{array}\right)$	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 4 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 4 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$
head	RSparseMatrix	RSparseMatrix	RSparseMatrix	RSparseMatrix

When an arithmetic operation can be performed on the underlying sparse arrays but the row names or column names do not coincide the names are dropped.

In[133]:=

```
SetRowNames[rmat3, "s." <> # & /@ ColumnNames[rmat]];
SetColumnNames[rmat3, RowNames[rmat]];
```

Out[135]=

expr	rmat	Transpose[rmat3]	rmat + Transpose[rmat3]	rmat Transpose[rmat3]
result	$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$	$\left(\begin{array}{c|ccccc} & s.a & s.b & s.c & s.d & s.e \\ \hline A & 1 & 2 & 3 & 1 & 1 \\ B & 0 & 2 & 4 & 2 & 0 \\ C & 2 & 2 & 0 & 2 & 2 \\ D & 1 & 1 & 1 & 3 & 4 \end{array}\right)$	$\begin{pmatrix} 2 & 2 & 3 & 5 & 1 \\ 0 & 4 & 4 & 2 & 0 \\ 2 & 2 & 0 & 2 & 4 \\ 1 & 1 & 4 & 3 & 4 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 4 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 \end{pmatrix}$
head	RSparseMatrix	RSparseMatrix	RSparseMatrix	RSparseMatrix

Part

A major useful feature is to have `Part` work with row and column names. The implementation of that additional functionality for `Part` is demonstrated below.

In the cases when the dimension drops sparse arrays or numbers are returned. In R the operation “[“ has the parameter “drop” -- the expression “smat[1,,drop=F]” is going to be a spare matrix, the expression “smat[1,,drop=T]” is going to be a dense vector. The corresponding implementation is to have the option “Drop→True|False” for `Part`, but that does not seem a good idea.

In the tables with examples below the last rows show the heads of the results.

Single row or column retrieval

Out[137]=

<table>
<tr><td>expr</td><td>rmat</td><td>rmat⟦A⟧</td><td>rmat⟦All, a⟧</td><td>rmat⟦{A}⟧</td><td>rmat⟦A, All⟧</td><td>rmat⟦All, a⟧</td><td>rmat⟦A, d⟧</td></tr>
<tr><td>result</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 4 \\ 0 \end{array}\right)$</td><td>$\left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 4 \\ 0 \end{array}\right)$</td><td>$\left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \end{array}\right)$</td><td>4</td></tr>
<tr><td>head</td><td>RSparseMatrix</td><td>SparseArray</td><td>SparseArray</td><td>RSparseMatrix</td><td>SparseArray</td><td>SparseArray</td><td>Integer</td></tr>
</table>

Permutation of both row names and column names

Out[139]=

<table>
<tr><td>expr</td><td>rmat</td><td>rmat⟦{C, D, A, B}⟧</td><td>rmat⟦{C, D, A, B}, {c, d, e, a, b}⟧</td></tr>
<tr><td>result</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \\ A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|ccccc} & c & d & e & a & b \\ \hline C & 0 & 0 & 2 & 0 & 0 \\ D & 3 & 0 & 0 & 0 & 0 \\ A & 0 & 4 & 0 & 1 & 0 \\ B & 0 & 0 & 0 & 0 & 2 \end{array}\right)$</td></tr>
<tr><td>head</td><td>RSparseMatrix</td><td>RSparseMatrix</td><td>RSparseMatrix</td></tr>
</table>

Various subsets

Out[141]=

<table>
<tr><td>expr</td><td>rmat</td><td>rmat⟦{A, B}, {a, c, d}⟧</td><td>rmat⟦2 ;; 3, 1 ;; 2⟧</td><td>rmat⟦{A, B}, 1 ;; 2⟧</td><td>rmat⟦All, {a, c}⟧</td></tr>
<tr><td>result</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|ccc} & a & c & d \\ \hline A & 1 & 0 & 4 \\ B & 0 & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|cc} & a & b \\ \hline B & 0 & 2 \\ C & 0 & 0 \end{array}\right)$</td><td>$\left(\begin{array}{c|cc} & a & b \\ \hline A & 1 & 0 \\ B & 0 & 2 \end{array}\right)$</td><td>$\left(\begin{array}{c|cc} & a & c \\ \hline A & 1 & 0 \\ B & 0 & 0 \\ C & 0 & 0 \\ D & 0 & 3 \end{array}\right)$</td></tr>
<tr><td>head</td><td>RSparseMatrix</td><td>RSparseMatrix</td><td>RSparseMatrix</td><td>RSparseMatrix</td><td>RSparseMatrix</td></tr>
</table>

Out[143]=

<table>
<tr><td>expr</td><td>rmat</td><td>rmat⟦{A}, 1⟧</td><td>rmat⟦{A}, {a}⟧</td><td>rmat⟦2, 1 ;; 2⟧</td><td>rmat⟦C, All⟧</td><td>rmat⟦All, All⟧</td></tr>
<tr><td>result</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td><td>$(1)$</td><td>$\left(\begin{array}{c|c} & a \\ \hline A & 1 \end{array}\right)$</td><td>$\left(\begin{array}{c} 0 \\ 2 \end{array}\right)$</td><td>$\left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 2 \end{array}\right)$</td><td>$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$</td></tr>
<tr><td>head</td><td>RSparseMatrix</td><td>SparseArray</td><td>RSparseMatrix</td><td>SparseArray</td><td>SparseArray</td><td>RSparseMatrix</td></tr>
</table>

RowBind, ColumnBind

Row and column binding are useful in various data analysis scenarios.

When using row and column names there are couple of questions to be answered.

1. How duplication of row (column) names is handled?

2. How can we specify to ignore the row (column) names when the doing the binding?

In[145]:=
```
MatrixForm[RowBind[rmat, rmat]]
```

Out[145]//MatrixForm=

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

In[146]:=
```
rmat2 = ToRSparseMatrix[rmat, "RowNames" → Map["s." <> # &, RowNames[rmat]]];
MatrixForm[rmat2]
```

Out[147]//MatrixForm=

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

In[148]:=
```
rmat3 = ToRSparseMatrix[rmat, "ColumnNames" → Map["t." <> # &, ColumnNames[rmat]]];
MatrixForm[rmat3]
```

Out[149]//MatrixForm=

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

In[150]:=
```
MatrixForm[RowBind[rmat, rmat2]]
```

Out[150]//MatrixForm=

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

In[151]:=
```
{MatrixForm[ColumnBind[rmat, rmat2]], MatrixForm[ColumnBind[rmat, rmat3]]}
```

Out[151]=

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

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

Profiling

In[73]:=
```
smat = SparseArray[RandomReal[{0, 1}, {1000, 120}]];
```

In[74]:=
```
rmat = ToRSparseMatrix[smat, "RowNames" → Map["A" <> ToString[#] &, Range[Dimensions[smat]〚1〛]],
   "ColumnNames" → Map["b" <> ToString[#] &, Range[Dimensions[smat]〚2〛]]];
```

In[75]:=
```
n = 100;
tres =
 AbsoluteTiming[
  Do[sres = smat.Transpose[smat], {i, n}]
 ]
tres〚1〛 / n
```

Out[76]= {5.4868, Null}

Out[77]= 0.054868

In[78]:=
```
tres =
 AbsoluteTiming[
  Do[rres = rmat.Transpose[rmat], {i, n}]
 ]
tres〚1〛 / n
```

Out[78]= {5.50556, Null}

Out[79]= 0.0550556

In[80]:=
```
Norm[sres〚1 ;; 120, 1 ;; 120〛 - SparseArray[rres〚1 ;; 120, 1 ;; 120〛]]
```

Out[80]= 0.

In[81]:=
```
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 .