SSparseMatrx

Anton Antonov
MathematicaForPrediction at GitHub
MathematicaForPrediction blog at WordPress.com
May 2015
October 2015
March 2018

Introduction

This notebook has the function implementations for manipulating objects with head SSparseMatrx 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 languages R [1]. (Similar to regular matrices in S and R.)

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 S and R, SSparseMatrx handles only strings as row names and column names.

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

The first version of the package SSparseMatrix.m is based on RSparseMatrix.m. Since the name `RSparseMatrix` hints to the internal functions and heads of RLink, the name was changed in `SSparseMatrix`. S precedes R and (I strongly assume that) S is the first of the two languages to have the named matrix rows and columns.

The following function signatures are implemented:

```
RowNames[_SSparseMatrx]
ColumnNames[_SSparseMatrx]
SetRowNames[_SSparseMatrx, {_String..}]
SetColumnNames[_SSparseMatrx, {_String..}]
DimensionNames[_SSparseMatrx]
```

```
Dimensions[_SSparseMatrx]
RowsCount[_SSparseMatrx]
ColumnsCount[_SSparseMatrx]
RowSums[_SSparseMatrx]
ColumnSums[_SSparseMatrx]
Total[_SSparseMatrx,___]
ArrayRules[_SSparseMatrx]
Transpose[_SSparseMatrx]
MatrixForm[_SSparseMatrx]
MatrixPlot[_SSparseMatrx]
Times[_SSparseMatrx, _SSparseMatrx]
Times[_,_SSparseMatrx]
Times[_SSparseMatrx, _]
Plus[_SSparseMatrx, _SSparseMatrx]
Plus[_,_SSparseMatrx]
Plus[_SSparseMatrx, _]
Dot[_SSparseMatrx, _SSparseMatrx]
Dot[_,_SSparseMatrx]
Dot[_SSparseMatrx, _]
Part[_SSparseMatrx, _String | {_String ..},___]
Part[_SSparseMatrx, _,_String | {_String ..}]
Part[_SSparseMatrx, _String | {_String ..}, _String | {_String ..}]
RowBind[_SSparseMatrx,_SSparseMatrx]
ColumnBind[_SSparseMatrx,_SSparseMatrx]
```

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

The package can be loaded from GitHub [3]:

In[27]:=
```
Import["https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/SSparseMatrix.m"]
```

Most of the examples below are turned into unit tests in the file GitHub file “SSparseMatrix-tests.wlt”, [6].

Exposition functions

This function is used to visualize the commands, the results, and the results’ heads.

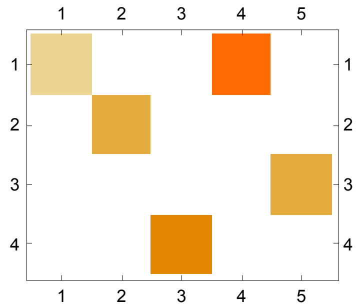
In[28]:=
```
Clear[ResultsGrid]
ResultsGrid[expressions_Inactive, opts___] :=
  Block[{t, t1},
   t = Map[HoldForm, expressions, {2}];
   t1 = ReleaseHold[Activate[t]];
   Grid[MapThread[Prepend, {{Activate[t], MatrixForm /@ t1, Head /@ t1},
      Style[#, Blue, FontFamily → "Times"] & /@ {"expr", "result", "head"}}], opts]
  ];
```

Tests and experiments

SparseArrays for comparisons

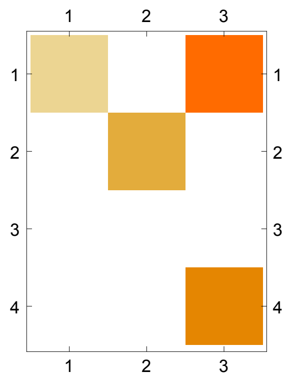
In[30]:=
```
mat = SparseArray[{{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 4} → 4, {3, 5} → 2}];
MatrixPlot[mat]
```

Out[31]=



In[32]:=
```
mat2 = SparseArray[{{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 3} → 4}];
MatrixPlot[mat2]
```

Out[33]=



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 SSparseMatrx objects.

In[34]:=
```
Grid[{{MatrixForm[mat], MatrixForm[Join[mat, mat]], MatrixForm[Transpose@Join[Transpose[mat], Transpose[mat]]]}}]
```

Out[34]=

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

Creation

In[35]:=
```
MatrixQ[{{1, 1} → 1, {2, 2} → 2, {4, 3} → 3, {1, 4} → 4, {3, 5} → 2}]
```

Out[35]=
```
False
```

In[36]:=
```
rmat = MakeSSparseMatrix[
  {{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[36]=
```
SparseArray[ Specified elements: 5
             Dimensions: {4, 5} ]
```

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

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

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

In[37]:=
```
rmat // MatrixForm
```

Out[37]//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

Here is the full form of the object `rmat`:

In[38]:=
```
rmat // FullForm
```

Out[38]//FullForm=
```
SSparseMatrix[Association[Rule["SparseMatrix", SparseArray[Automatic, List[4, 5], 0,
    List[1, List[List[0, 2, 3, 4, 5], List[List[1], List[4], List[2], List[5], List[3]]], List[1, 4, 2, 2, 3]]]],
  Rule["RowNames", Association[Rule["A", 1], Rule["B", 2], Rule["C", 3], Rule["D", 4]]],
  Rule["ColumnNames", Association[Rule["a", 1], Rule["b", 2], Rule["c", 3], Rule["d", 4], Rule["e", 5]]],
  Rule["DimensionNames", Association[Rule["U", 1], Rule["V", 2]]]]]
```

The SSparseMatrx objects can be created from `SparseArray` objects :

In[39]:=
```
rmat = ToSSparseMatrix[SparseArray[rmat], "ColumnNames" → {"a", "b", "c", "d", "e"},
  "RowNames" → {"A", "B", "C", "D"}, "DimensionNames" → {"U", "V"}]
```

Out[39]=
```
SparseArray[ Specified elements: 5
             Dimensions: {4, 5} ]
```

In[40]:=
```
rmat // MatrixForm
```

Out[40]//MatrixForm=

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

Setting names

This section shows the setting of row and column names.

In[41]:=
```
rmat2 = rmat
```

Out[41]= SparseArray[Specified elements: 5 Dimensions: {4, 5}]

In[42]:=
```
MatrixForm[rmat2]
```

Out[42]//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[43]:=
```
SetRowNames[rmat2, ToString /@ Range[RowsCount[rmat]]]
```

Out[43]= SparseArray[Specified elements: 5 Dimensions: {4, 5}]

In[44]:=
```
MatrixForm[rmat2]
```

Out[44]//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[45]:=
```
SetColumnNames[rmat2, ToString /@ Range[ColumnsCount[rmat]]]
```

Out[45]= SparseArray[Specified elements: 5 Dimensions: {4, 5}]

In[46]:= `MatrixForm[rmat2]`

Out[46]//MatrixForm=

$$\left(\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 0 & 0 & 4 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 2 \\ 4 & 0 & 0 & 3 & 0 & 0 \end{array}\right)$$

In[47]:= `MatrixForm[rmat]`

Out[47]//MatrixForm=

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

Query functions

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

```
In[48]:= RowNames[rmat]
Out[48]= {A, B, C, D}
```

```
In[49]:= ColumnNames[rmat]
Out[49]= {a, b, c, d, e}
```

```
In[50]:= DimensionNames[rmat]
Out[50]= {U, V}
```

Functions that can be applied to sparse arrays follow.

```
In[51]:= Dimensions[rmat]
Out[51]= {4, 5}
```

```
In[52]:= ArrayRules[rmat]
Out[52]= {{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 SSparseMatrx objects.

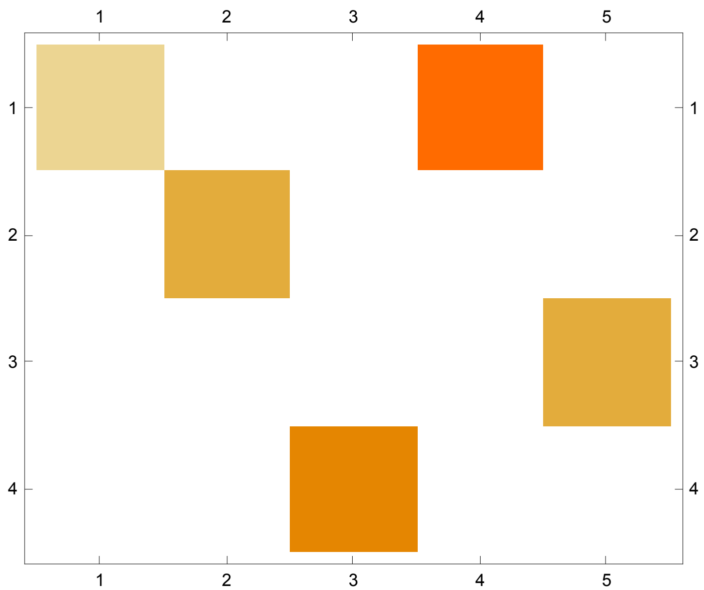
In[53]:=
```
MatrixForm[rmat]
```

Out[53]//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[54]:=
```
MatrixPlot[rmat]
```

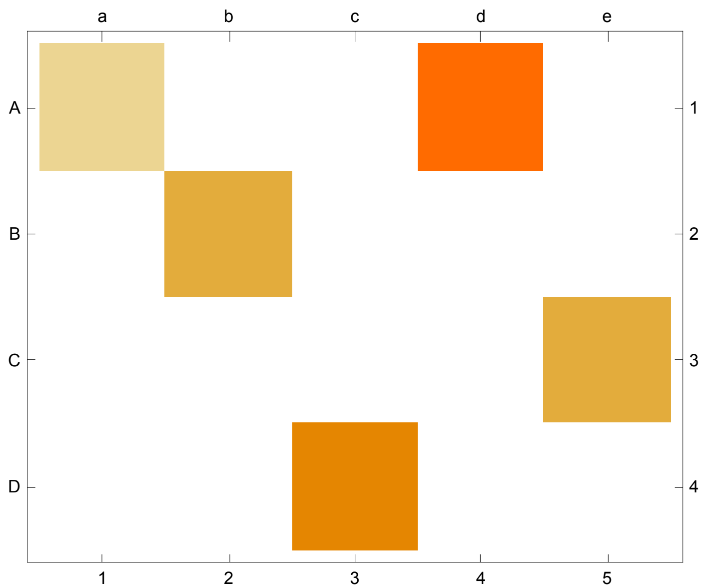
Out[54]=



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

In[55]:=
```
MatrixPlot[rmat, FrameTicks → {{Transpose[{Range[RowsCount[rmat]], RowNames[rmat]}], All},
  {All, Transpose[{Range[ColumnsCount[rmat]], ColumnNames[rmat]}]}}]
```

Out[55]=



Transpose

In[56]:= `MatrixForm[Transpose[rmat]]`

Out[56]//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
e	0	0	2	0

In[57]:= `DimensionNames[Transpose[rmat]]`

Out[57]= {V, U}

Sums

In[58]:= `MatrixForm[rmat]`

Out[58]//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[59]:= `Total[rmat, 2]`

Out[59]= 12

In[60]:= `RowSums[rmat]`

Out[60]= {5, 2, 2, 3}

In[61]:= `ColumnSums[rmat]`

Out[61]= {1, 2, 3, 4, 2}

Dot product

In order to make the SSparseMatrix objects really useful we have to implement matrix-vector and matrix-matrix operations for them. (With other SSparseMatrix objects and with SparseArray objects.)

Matrix by vector

Out[62]=

<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>SSparseMatrix</td><td>SSparseMatrix</td><td>SSparseMatrix</td></tr>
</table>

Out[64]=

<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>SSparseMatrix</td><td>SparseArray</td><td>SSparseMatrix</td></tr>
</table>

Matrix by matrix

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

Out[66]=

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

This creates another SSparseMatrx object with no row and column names:

In[68]:=

```
rmat2 = ToSSparseMatrix[SparseArray[RandomInteger[{0, 4}, {ColumnsCount[rmat], RowsCount[rmat]}]]];
```

Next we look into two dot products of two SSparseMatrx objects.

Out[69]=

expr	rmat	rmat2	rmat.rmat2	rmat2.rmat	rmat.rmat2.rmat
result	$\left(\begin{array}{c\|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 3 & 0 & 4 \\ 3 & 0 & 3 & 3 \\ 0 & 3 & 4 & 2 \\ 4 & 0 & 4 & 0 \end{pmatrix}$	$\left(\begin{array}{c\|cccc} A & 1 & 12 & 17 & 9 \\ B & 0 & 6 & 0 & 8 \\ C & 8 & 0 & 8 & 0 \\ D & 9 & 0 & 9 & 9 \end{array}\right)$	$\left(\begin{array}{ccccc} a & b & c & d & e \\ \hline 1 & 0 & 3 & 4 & 2 \\ 0 & 6 & 12 & 0 & 0 \\ 3 & 0 & 9 & 12 & 6 \\ 0 & 6 & 6 & 0 & 8 \\ 4 & 0 & 0 & 16 & 8 \end{array}\right)$	$\left(\begin{array}{c\|ccccc} & a & b & c & d & e \\ \hline A & 1 & 24 & 27 & 4 & 34 \\ B & 0 & 12 & 24 & 0 & 0 \\ C & 8 & 0 & 0 & 32 & 16 \\ D & 9 & 0 & 27 & 36 & 18 \end{array}\right)$
head	SSparseMatrix	SSparseMatrix	SSparseMatrix	SSparseMatrix	SSparseMatrix

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

In[71]:=

```
DimensionNames[rmat.Transpose[rmat]]
```

Out[71]= {U}

Verification:

Out[72]=

expr	SparseArray[rmat]	SparseArray[rmat2]	SparseArray[rmat]. SparseArray[rmat2]	SparseArray[rmat2]. SparseArray[rmat]	SparseArray[rmat]. SparseArray[rmat2]. SparseArray[rmat]
result	$\begin{pmatrix} 1 & 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 3 & 0 & 4 \\ 3 & 0 & 3 & 3 \\ 0 & 3 & 4 & 2 \\ 4 & 0 & 4 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 12 & 17 & 9 \\ 0 & 6 & 0 & 8 \\ 8 & 0 & 8 & 0 \\ 9 & 0 & 9 & 9 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 3 & 4 & 2 \\ 0 & 6 & 12 & 0 & 0 \\ 3 & 0 & 9 & 12 & 6 \\ 0 & 6 & 6 & 0 & 8 \\ 4 & 0 & 0 & 16 & 8 \end{pmatrix}$	$\begin{pmatrix} 1 & 24 & 27 & 4 & 34 \\ 0 & 12 & 24 & 0 & 0 \\ 8 & 0 & 0 & 32 & 16 \\ 9 & 0 & 27 & 36 & 18 \end{pmatrix}$
head	SparseArray	SparseArray	SparseArray	SparseArray	SparseArray

Arithmetic operations

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

Out[74]=

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	SSparseMatrix	SSparseMatrix	SSparseMatrix	SSparseMatrix

Out[76]=

expr	rmat	Transpose[rmat2]	rmat + Transpose[rmat2]	rmat Transpose[rmat2]
result	$\left(\begin{array}{c\|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$	$\begin{pmatrix} 1 & 0 & 3 & 0 & 4 \\ 0 & 3 & 0 & 3 & 0 \\ 1 & 0 & 3 & 4 & 4 \\ 1 & 4 & 3 & 2 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & 0 & 3 & 4 & 4 \\ 0 & 5 & 0 & 3 & 0 \\ 1 & 0 & 3 & 4 & 6 \\ 1 & 4 & 6 & 2 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 9 & 0 & 0 \end{pmatrix}$
head	SSparseMatrix	SSparseMatrix	SSparseMatrix	SSparseMatrix

In[78]:=
```
rmat3 = rmat2;
```

In[79]:=
```
SetRowNames[rmat3, ColumnNames[rmat]];
SetColumnNames[rmat3, RowNames[rmat]];
```

Out[81]=

expr	rmat	Transpose[rmat3]	rmat + Transpose[rmat3]	rmat Transpose[rmat3]
result	$\left(\begin{array}{c\|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$	$\left(\begin{array}{c\|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 3 & 0 & 4 \\ B & 0 & 3 & 0 & 3 & 0 \\ C & 1 & 0 & 3 & 4 & 4 \\ D & 1 & 4 & 3 & 2 & 0 \end{array}\right)$	$\left(\begin{array}{c\|ccccc} & a & b & c & d & e \\ \hline A & 2 & 0 & 3 & 4 & 4 \\ B & 0 & 5 & 0 & 3 & 0 \\ C & 1 & 0 & 3 & 4 & 6 \\ D & 1 & 4 & 6 & 2 & 0 \end{array}\right)$	$\left(\begin{array}{c\|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 0 & 0 \\ B & 0 & 6 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 8 \\ D & 0 & 0 & 9 & 0 & 0 \end{array}\right)$
head	SSparseMatrix	SSparseMatrix	SSparseMatrix	SSparseMatrix

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

Out[85]=

expr	rmat	Transpose[rmat3]	rmat + Transpose[rmat3]	rmat Transpose[rmat3]
result	$\left(\begin{array}{c\|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 2 \\ D & 0 & 0 & 3 & 0 & 0 \end{array}\right)$	$\left(\begin{array}{c\|ccccc} & s.a & s.b & s.c & s.d & s.e \\ \hline A & 1 & 0 & 3 & 0 & 4 \\ B & 0 & 3 & 0 & 3 & 0 \\ C & 1 & 0 & 3 & 4 & 4 \\ D & 1 & 4 & 3 & 2 & 0 \end{array}\right)$	$\begin{pmatrix} 2 & 0 & 3 & 4 & 4 \\ 0 & 5 & 0 & 3 & 0 \\ 1 & 0 & 3 & 4 & 6 \\ 1 & 4 & 6 & 2 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 9 & 0 & 0 \end{pmatrix}$
head	SSparseMatrix	SSparseMatrix	SSparseMatrix	SSparseMatrix

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 sparse 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[87]=

<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>SSparseMatrix</td><td>SparseArray</td><td>SparseArray</td><td>SSparseMatrix</td><td>SparseArray</td><td>SparseArray</td><td>Integer</td></tr>
</table>

Permutation of both row names and column names

Out[89]=

<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>SSparseMatrix</td><td>SSparseMatrix</td><td>SSparseMatrix</td></tr>
</table>

Various subsets

Out[91]=

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

Out[93]=

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

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[95]:=
```
rmat2 = ToSSparseMatrix[rmat, "RowNames" → Map["s." <> # &, RowNames[rmat]]];
```

In[96]:=
```
rmat3 = ToSSparseMatrix[rmat, "ColumnNames" → Map["t." <> # &, ColumnNames[rmat]]];
```

In[97]:=
```
ResultsGrid[Inactive[{rmat, rmat2, rmat3}], Dividers → All]
```

Out[97]=

expr	rmat	rmat2	rmat3
result	see below	see below	see below
head	SSparseMatrix	SSparseMatrix	SSparseMatrix

rmat:

	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

rmat2:

	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

rmat3:

	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[98]:=
```
ResultsGrid[Inactive[{RowBind[rmat, rmat], RowBind[rmat, rmat2]}], Dividers → All]
```

Out[98]=

expr	RowBind[rmat, rmat]	RowBind[rmat, rmat2]
result	see below	see below
head	SSparseMatrix	SSparseMatrix

RowBind[rmat, rmat]:

	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

RowBind[rmat, rmat2]:

	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[99]:=
```
ResultsGrid[Inactive[{ColumnBind[rmat, rmat2], MatrixForm[ColumnBind[rmat, rmat3]]}], Dividers → All]
```

Out[99]=

expr	ColumnBind[rmat, rmat2]	ColumnBind[rmat, rmat3]
result	(see matrices below)	(see matrices below)
head	SSparseMatrix	MatrixForm

ColumnBind[rmat, rmat2]:

	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

ColumnBind[rmat, rmat3]:

	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

Unit tests

Here are the results of the unit test running with [6].

```
In[100]:= testReport = TestReport["~/MathematicaForPrediction/UnitTests/SSparseMatrix-tests.wlt"]

Out[100]= TestReportObject[ Title: Test Report: SSparseMatrix-tests.wlt
                            Success rate: 98%    Tests run: 46 ]

In[101]:= testReport["TestsSucceededCount"]

Out[101]= 45

In[102]:= testReport["TestsFailedCount"]

Out[102]= 1
```

The failed test

```
In[103]:= testReport["TestsFailed"]

Out[103]= <|TestsFailedWrongResults → <|37 → TestResultObject[ Outcome: Failure
                                                               Test ID: PermutationPart-2 ]|>,
           TestsFailedWithMessages → <||>, TestsFailedWithErrors → <||>|>
```

I am aware of the failing test. It happens when `Part` is called with a list of integers:

```
In[104]:= rmat[[{1, 2}]]

Part: Part {1, 2} of SparseArray[ Specified elements: 5
                                  Dimensions: {4, 5} ] does not exist.

Out[104]= SparseArray[ Specified elements: 5
                       Dimensions: {4, 5} ][[{1, 2}]]
```

The workaround is to use `All`:

In[105]:=
```
rmat[[{1, 2}, All]] // MatrixForm
```

Out[105]//MatrixForm=

$$\left(\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline A & 1 & 0 & 0 & 4 & 0 \\ B & 0 & 2 & 0 & 0 & 0 \end{array}\right)$$

Profiling

In this section we show simple profiling tests based on matrix-matrix multiplication.

Note that difference between using SparseArray objects and SSparseMatrix objects is negligent.

In[106]:=
```
smat = SparseArray[RandomReal[{0, 1}, {1000, 120}]];
```

In[107]:=
```
rmat = ToSSparseMatrix[smat, "RowNames" → Map["A" <> ToString[#] &, Range[Dimensions[smat][[1]]]],
   "ColumnNames" → Map["b" <> ToString[#] &, Range[Dimensions[smat][[2]]]]];
```

Using SparseArray objects:

In[108]:=
```
n = 100;
tres =
 AbsoluteTiming[
  Do[sres = smat.Transpose[smat], {i, n}]
 ]
tres[[1]] / n
```

Out[109]=
```
{4.45523, Null}
```

Out[110]=
```
0.0445523
```

Using SSparseMatrix objects:

In[111]:=
```
tres =
 AbsoluteTiming[
  Do[rres = rmat.Transpose[rmat], {i, n}]
 ]
tres〚1〛 / n
```

Out[111]= {5.22466, Null}

Out[112]= 0.0522466

Same results are obtained:

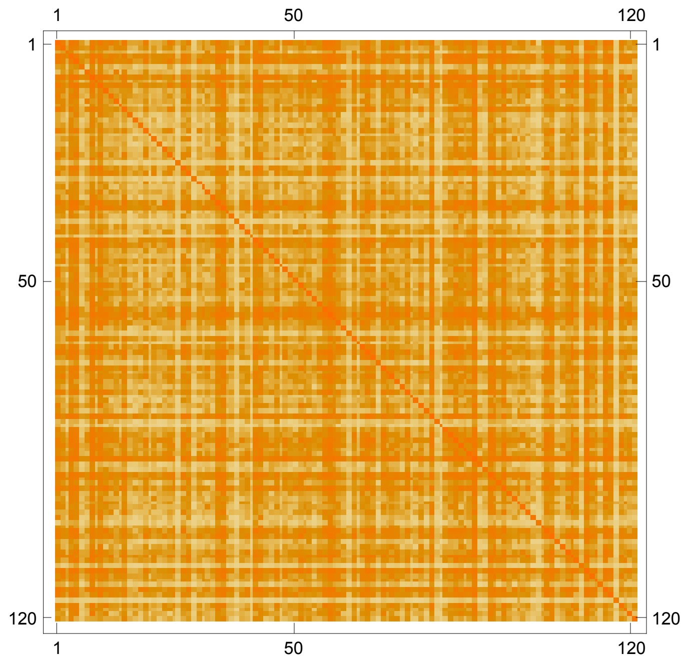
In[113]:=
```
Norm[sres〚1 ;; 120, 1 ;; 120〛 - SparseArray[rres〚1 ;; 120, 1 ;; 120〛]]
```

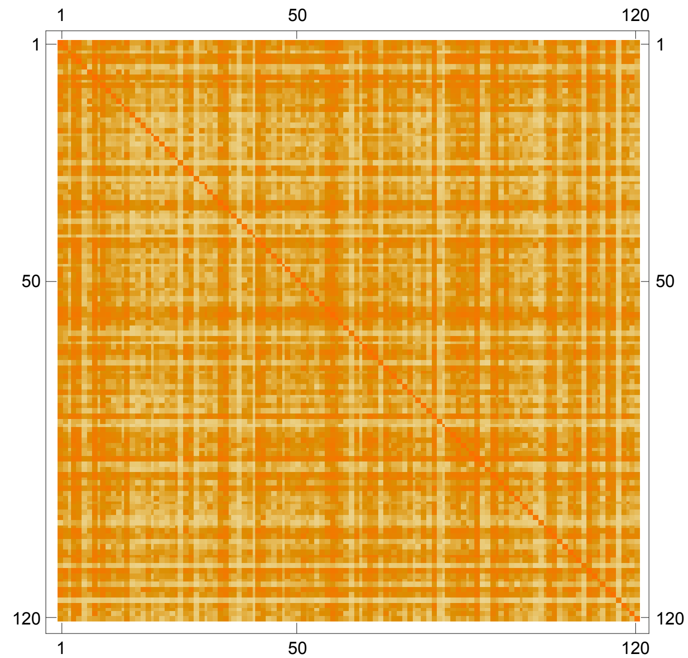
Out[113]= 0.

Visualization:

In[114]:=
```
Grid[{{
   MatrixPlot[rres⟦1 ;; 120, 1 ;; 120⟧, ImageSize → 350],
   MatrixPlot[rres⟦"A" <> ToString[#] & /@ Range[120], 1 ;; 120⟧, ImageSize → 350]}}]
```

Out[114]=

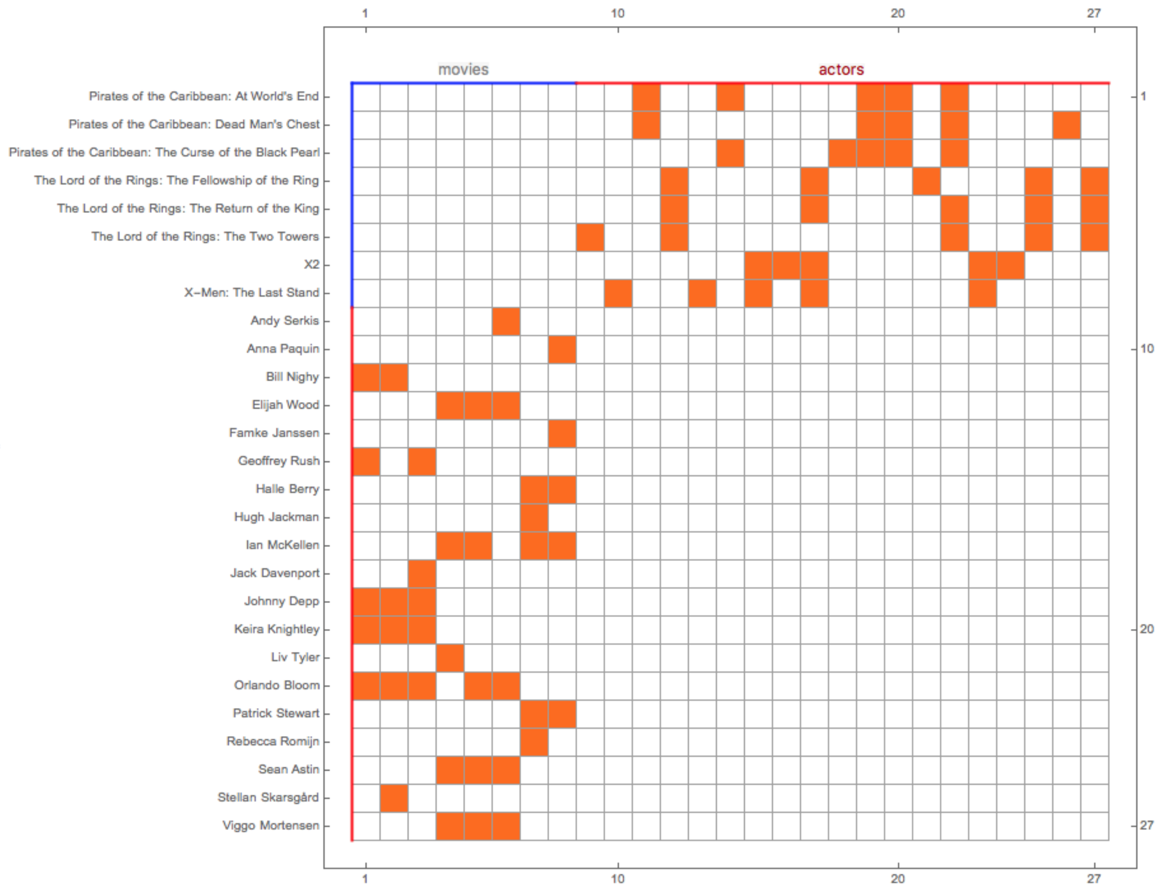




Neat example

Consider this incidence matrix, `rBiMat0`, that represents a bi-partite graph of actors starring in movies relationships:

Out[116]=



We can use a SSparseMatrx object of it with named rows and columns (`rBiMat01`).

If we want to see which actors have participated in movies together with Orlando Bloom we can do the following:

In[117]:=
```
(*Magnify[#,0.7]&@MatrixForm[rBiMat01.rBiMat01⟦All,{"Orlando Bloom"}⟧]*)
```

In[207]:=
```
MatrixForm[rBiMat.rBiMat[[All, {"Orlando Bloom"}]]
```

Out[207]//MatrixForm=

Out[118]=

	Orlando Bloom
Pirates of the Caribbean: At World's End	0
Pirates of the Caribbean: Dead Man's Chest	0
Pirates of the Caribbean: The Curse of the Black Pearl	0
The Lord of the Rings: The Fellowship of the Ring	0
The Lord of the Rings: The Return of the King	0
The Lord of the Rings: The Two Towers	0
X2	0
X-Men: The Last Stand	0
Andy Serkis	1
Anna Paquin	0
Bill Nighy	2
Elijah Wood	2
Famke Janssen	0
Geoffrey Rush	2
Halle Berry	0
Hugh Jackman	0
Ian McKellen	1
Jack Davenport	1
Johnny Depp	3
Keira Knightley	3
Liv Tyler	0
Orlando Bloom	5
Patrick Stewart	0
Rebecca Romijn	0
Sean Astin	2
Stellan Skarsgård	1
Viggo Mortensen	2

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, RSparseMatrx *Mathematica* package, (2015), *Mathematica*ForPrediction project at GitHub.
URL: https://github.com/antononcube/MathematicaForPrediction/blob/master/Misc/SSparseMatrx.m .

[4] A. Antonov, SSparseMatrx Mathematica package, (2018), *Mathematica*ForPrediction project at GitHub.
URL: https://github.com/antononcube/MathematicaForPrediction/blob/master/SSparseMatrx.m .

[5] A. Antonov, SSparseMatrix Mathematica unit tests, (2018), MathematicaForPrediction at GitHub.
URL: https://github.com/antononcube/MathematicaForPrediction/blob/master/UnitTests/SSparseMatrix-tests.wlt

[6] A. Antonov, SSparseMatrix Mathematica unit tests, (2018), MathematicaForPrediction at GitHub.
URL: https://github.com/antononcube/MathematicaForPrediction/blob/master/UnitTests/SSparseMatrix-tests.wlt .