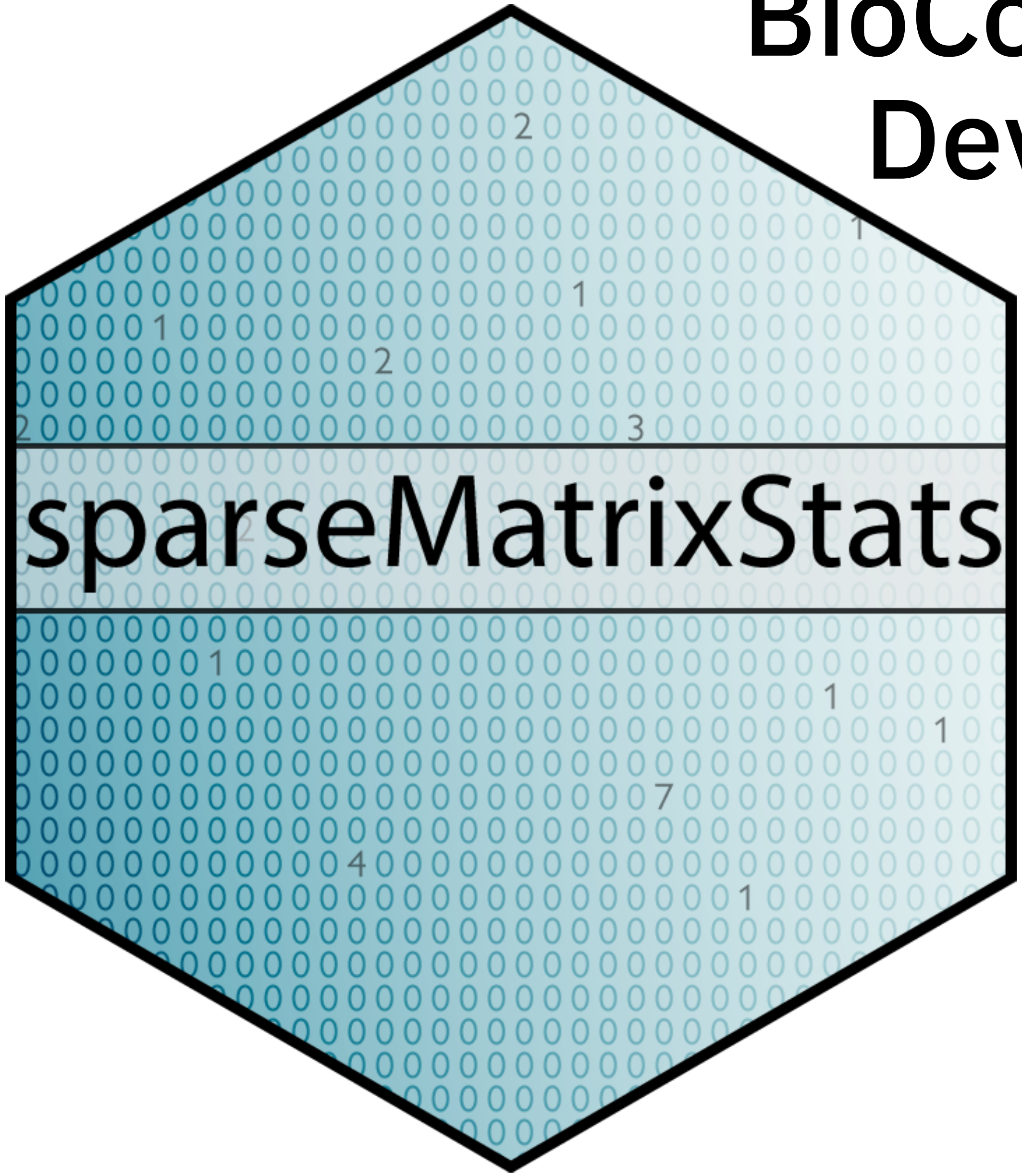


BioConductor Developers Forum

October 2020

Constantin Ahlmann-Eltze

@const_ae

About me

- PhD student at EMBL Heidelberg with Wolfgang Huber
- Working on single cell data analysis methods
- Using R since 2015
- Had three month of (not so interesting) lectures last year…

Motivation

```cpp
NumericVector row_var_dgcmatrix(NumericVector x, IntegerVector i, int rows, int cols) {
  NumericVector rowmean(rows, 0.0);
  for (int k=0; k<x.length(); ++k) {
    rowmean[i[k]] += x[k];
  }
  for (int k=0; k<rows; ++k) {
    rowmean[k] /= cols;
  }
  NumericVector rowvar(rows, 0.0);
  IntegerVector nzero(rows, cols);
  for (int k=0; k<x.length(); ++k) {
    rowvar[i[k]] += pow(x[k] - rowmean[i[k]], 2);
    nzero[i[k]] -= 1;
  }
  for (int k=0; k<rows; ++k) {
    rowvar[k] = (rowvar[k] + (pow(rowmean[k], 2) * nzero[k])) / (cols - 1);
  }
  return rowvar;
}
```

From Christoph Hafemeister, sctransform

```r
colVars_spm <- function( spm ) {
  stopifnot( methods::is( spm, "dgCMatrix" ) )
  ans <- sapply( base::seq.int(spm@Dim[2]), function(j) {
    if( spm@p[j+1] == spm@p[j] ) { return(0) } # all entries are 0: var is 0
    mean <- base::sum( spm@x[ (spm@p[j]+1):spm@p[j+1] ] ) / spm@Dim[1]
    sum( ( spm@x[ (spm@p[j]+1):spm@p[j+1] ] - mean )^2 ) +
      mean^2 * ( spm@Dim[1] - ( spm@p[j+1] - spm@p[j] ) ) } ) / ( spm@Dim[1] - 1 )
  names(ans) <- spm@Dimnames[[2]]
  ans
}
```

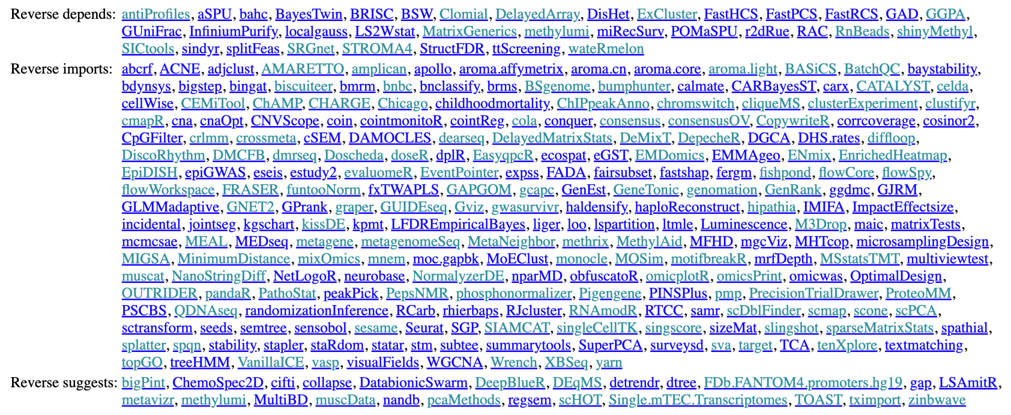
From Felix Frauhammer, scUtils

sparseMatrixStats ports the matrixStats API to dgCMatrix objects

matrixStats: Functions that Apply to Rows and Columns of Matrices (and to Vectors)

High-performing functions operating on rows and columns of matrices, e.g. col / rowMedians(), col / rowRanks(), and col / rowSds()

downloads 409K/month



Reverse depends:	antiProfiles, aSPU, bahc, BayesTwin, BRISC, BSW, Clomial, DelayedArray, DisHet, ExCluster, FastHCS, FastPCS, FastRCS, GAD, GGPA, GUniFrac, InfiniumPurify, localgauss, LS2Wstat, MatrixGenerics, methylumi, miRecSurv, POMaSPU, r2dRue, RAC, RnBeads, shinyMethyl, SICtools, sindyr, splitFeas, SRGnet, STROMA4, StructFDR, ttScreening, wateRmelon
Reverse imports:	abcrf, ACNE, adjclust, AMARETTO, amplican, apollo, aroma.affymetrix, aroma.cn, aroma.core, aroma.light, BASiCS, BatchQC, baystability, bdynsys, bigstep, bingat, biscuiteer, bmrm, bnbc, bnclassify, brms, BSgenome, bumphunter, calmate, CARBayesST, carx, CATALYST, celda, cellWise, CEMiTool, ChAMP, CHARGE, Chicago, childhoodmortality, ChIPpeakAnno, chromswitch, cliqueMS, clusterExperiment, clustifyr, cmapR, cna, cnaOpt, CNVScope, coin, cointmonitoR, cointReg, cola, conquer, consensus, consensusOV, CopywriteR, corrcoverage, cosinor2, CpGFilter, crlmm, crossmeta, cSEM, DAMOCLES, dearseq, DelayedMatrixStats, DeMixT, DepecheR, DGCA, DHS.rates, diffloop, DiscoRhythm, DMCFB, dmrseq, Doscheda, doseR, dplR, EasyqpcR, ecospat, eGST, EMDomics, EMMAgeo, ENmix, EnrichedHeatmap, EpiDISH, epiGWAS, eseis, estudy2, evaluomeR, EventPointer, expss, FADA, fairsubset, fastshap, fergm, fishpond, flowCore, flowSpy, flowWorkspace, FRASER, funtooNorm, fxTWAPLS, GAPGOM, gcapc, GenEst, GeneTonic, genomation, GenRank, ggdmc, GJRM, GLMMadaptive, GNET2, GPrank, graper, GUIDEseq, Gviz, gwasurvivr, haldensify, haploReconstruct, hipathia, IMIFA, ImpactEffectsize, incidental, jointseg, kgschart, kissDE, kpmt, LFDREmpiricalBayes, liger, loo, lspartition, ltmle, Luminescence, M3Drop, maic, matrixTests, mcmcsae, MEAL, MEDseq, metagene, metagenomeSeq, MetaNeighbor, methrix, MethylAid, MFHD, mgcViz, MHTcop, microsamplingDesign, MIGSA, MinimumDistance, mixOmics, mnem, moc.gapbk, MoEClust, monocle, MOSim, motifbreakR, mrfDepth, MSstatsTMT, multiviewtest, muscat, NanoStringDiff, NetLogoR, neurobase, NormalyzerDE, nparMD, obfuscatoR, omicplotR, omicsPrint, omicwas, OptimalDesign, OUTRIDER, pandaR, PathoStat, peakPick, PepsNMR, phosphonormalizer, Pigengene, PINSPlus, pmp, PrecisionTrialDrawer, ProteoMM, PSCBS, QDNAseq, randomizationInference, RCarb, rhierbaps, RJcluster, RNAmodR, RTCC, samr, scDblFinder, scmap, scone, scPCA, sctransform, seeds, semtree, sensobol, sesame, Seurat, SGP, SIAMCAT, singleCellTK, singscore, sizeMat, slingshot, sparseMatrixStats, spathial, splatter, spqn, stability, stapler, staRdom, statar, stm, subtee, summarytools, SuperPCA, surveysd, sva, target, TCA, tenXplore, textmatching, topGO, treeHMM, VanillaICE, vasp, visualFields, WGCNA, Wrench, XBSeq, yarn
Reverse suggests:	bigPint, ChemoSpec2D, cifti, collapse, DatabionicSwarm, DeepBlueR, DEqMS, detrendr, dtree, FDb.FANTOM4.promoters.hg19, gap, LSAmitR, metavizr, methylumi, MultiBD, muscData, nandb, pcaMethods, regsem, scHOT, Single.mTEC.Transcriptomes, TOAST, tximport, zinbwave

matrixStats API

colAlls()
colAnyMissings()
colAnyNAs()
colAnys()
colAvgsPerRowSet()
colCollapse()
colCounts()
colCummaxs()
colCummins()
colCumprods()
colCumsums()
colDiffs()
colIQRDiffs()
colIQRs()
colLogSumExps()
colMadDiffs()
colMads()
colMaxs()
colMeans2()
colMedians()
colMins()
colOrderStats()
colProds()
colQuantiles()
colRanges()
colRanks()
colSdDiffs()
colSds()
colsum()
colSums2()
colTabulates()
colVarDiffs()
colVars()
colWeightedMads()
colWeightedMeans()
colWeightedMedians()
colWeightedSds()
colWeightedVars()

rowAlls()
rowAnyMissings()
rowAnyNAs()
rowAnys()
rowAvgsPerColSet()
rowCollapse()
rowCounts()
rowCummaxs()
rowCummins()
rowCumprods()
rowCumsums()
rowDiffs()
rowIQRDiffs()
rowIQRs()
rowLogSumExps()
rowMadDiffs()
rowMads()
rowMaxs()
rowMeans2()
rowMedians()
rowMins()
rowOrderStats()
rowProds()
rowQuantiles()
rowRanges()
rowRanks()
rowSdDiffs()
rowSds()
rowsum()
rowSums2()
rowTabulates()
rowVarDiffs()
rowVars()
rowWeightedMads()
rowWeightedMeans()
rowWeightedMedians()
rowWeightedSds()
rowWeightedVars()

matrixStats::colSums2()

```
mat <- matrix(rpois(n = 6 * 3, lambda = 0.3), nrow = 6, ncol = 3)
mat
#>      [,1] [,2] [,3]
#> [1,]    0    0    0
#> [2,]    1    0    0
#> [3,]    1    0    0
#> [4,]    0    0    0
#> [5,]    0    1    1
#> [6,]    0    0    0

matrixStats::colSums2(mat)
#> [1] 2 1 1

matrixStats::rowSums2(mat)
#> [1] 0 1 1 0 2 0
```

The Matrix package contains S4 classes for efficiently storing sparse matrices

```
mat
#>      [,1] [,2] [,3]
#> [1,]    0    0    0
#> [2,]    1    0    0
#> [3,]    1    0    0
#> [4,]    0    0    0
#> [5,]    0    1    1
#> [6,]    0    0    0
```

```
library(Matrix)
dgC_mat <- as(mat, "dgCMatrix")
```

```
dgC_mat
#> 6 x 3 sparse Matrix of class "dgCMatrix"
#>
#> [1,] . . .
#> [2,] 1 . .
#> [3,] 1 . .
#> [4,] . . .
#> [5,] . 1 1
#> [6,] . . .
```

sparseMatrixStats::colSums2()

```
dgC_mat
#> 6 x 3 sparse Matrix of class
#>  "dgCMatrix"
#>
#> [1,] . 1 .
#> [2,] . . .
#> [3,] . . 1
#> [4,] 1 . .
#> [5,] . . .
#> [6,] 1 . 2

sparseMatrixStats::colSums2(dgC_mat)
#> [1] 2 1 3

sparseMatrixStats::rowSums2(dgC_mat)
#> [1] 1 0 1 1 0 3
```

sparseMatrixStats is a drop-in replacement for matrixStats API

```
library(matrixStats)
library(sparseMatrixStats)

colSums2(mat)
#> [1] 2 1 1
colSums2(dgC_mat)
#> [1] 2 1 1



rowVarDiffs(mat)
#> [1] 0.00 0.25 0.25 0.00 0.25 0.00
rowVarDiffs(dgC_mat)
#> [1] 0.00 0.25 0.25 0.00 0.25 0.00
```

Benchmarks

```r
se <- TENxPBMCData::TENxPBMCData("pbmc4k")
large_mat <- as.matrix(assay(se))
dgC_large_mat <- as(large_mat, "dgCMatrix")


sum(large_mat == 0) / prod(dim(large_mat))
#> [1] 0.9608314


bench::mark(matrixStats = colMeans2(large_mat),
            sparseMatrixStats = colMeans2(dgC_large_mat),
            Matrix = colMeans(dgC_large_mat))
#> # A tibble: 3 x 6
#>   expression              min   median `itr/sec` mem_alloc `gc/sec`
#>   <bch:expr>         <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
#> 1 matrixStats        162.36ms 165.86ms      6.06   297.2KB        0
#> 2 sparseMatrixStats    9.03ms   9.61ms    103.      36.4KB        0
#> 3 Matrix              13.04ms  13.85ms     71.5       34KB        0
```

25x

17x

```
bench::mark(matrixStats = colVars(large_mat),
            sparseMatrixStats = colVars(dgC_large_mat))
#> # A tibble: 2 x 6
#>   expression             min   median `itr/sec` mem_alloc `gc/sec`
#>   <bch:expr>        <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
#> 1 matrixStats        410.3ms  427.9ms      2.34   428.9KB        0
#> 2 sparseMatrixStats   16.4ms   17.4ms     57.5     36.4KB        0
```

40x

```
bench::mark(matrixStats = colMedians(large_mat),
            sparseMatrixStats = colMedians(dgC_large_mat))
#> # A tibble: 2 x 6
#>   expression             min   median `itr/sec` mem_alloc `gc/sec`
#>   <bch:expr>        <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
#> 1 matrixStats          726ms  725.9ms      1.38     438KB        0
#> 2 sparseMatrixStats     14ms   14.6ms     67.7     41.3KB        0
```

20x

```
bench::mark(matrixStats = colQuantiles(large_mat),
            sparseMatrixStats = colQuantiles(dgC_large_mat))
#> # A tibble: 2 x 6
#>   expression             min   median `itr/sec` mem_alloc `gc/sec`
#>   <bch:expr>        <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
#> 1 matrixStats           5.7s     5.7s     0.175    3.81GB    0.175
#> 2 sparseMatrixStats  960.8ms  960.8ms     1.04   341.65KB    0
```

6x

```
bench::mark(matrixStats = rowVars(large_mat),
            sparseMatrixStats = rowVars(dgC_large_mat))
#> # A tibble: 2 x 6
#>   expression             min   median `itr/sec` mem_alloc `gc/sec`
#>   <bch:expr>        <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
#> 1 matrixStats          1.82s    1.82s     0.551     314KB        0
#> 2 sparseMatrixStats  34.01ms  35.78ms    27.7       534KB        0
```

54x

```
bench::mark(matrixStats = rowMedians(large_mat),
            sparseMatrixStats = rowMedians(dgC_large_mat))
#> # A tibble: 2 x 6
#>   expression             min   median `itr/sec` mem_alloc `gc/sec`
#>   <bch:expr>        <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
#> 1 matrixStats          1.94s    1.94s     0.516   297.3KB        0
#> 2 sparseMatrixStats 179.63ms 193.44ms     5.27     66.5MB        0
```

10x

```
bench::mark(matrixStats = rowQuantiles(large_mat),
            sparseMatrixStats = rowQuantiles(dgC_large_mat))
#> # A tibble: 2 x 6
#>   expression             min   median `itr/sec` mem_alloc `gc/sec`
#>   <bch:expr>        <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
#> 1 matrixStats          5.68s    5.68s     0.176    1.65GB    0.352
#> 2 sparseMatrixStats 939.34ms 939.34ms     1.06    68.76MB    0
```

4x

dgCMatrix objects store *column-sparse* representation of the data

```
mat2
#>       [,1] [,2] [,3] [,4]
#>  [1,]    3    0    0    0
#>  [2,]    0    0    0    0
#>  [3,]    0    0    5    0
#>  [4,]    1    6    0    4
#>  [5,]    0    0    0    0
#>  [6,]    0    0    0    0
#>  [7,]    0    0    0    0
#>  [8,]    0    7    0    0
#>  [9,]    0    0    0    2
#> [10,]    0    0    0    0
```

```
dgC_mat2 <- as(mat2, "dgCMatrix")
str(dgC_mat2)
#> Formal class 'dgCMatrix' [package
"Matrix"] with 6 slots
#>   ..@ i       : int [1:7] 0 3 3 7 2 3 8
#>   ..@ p       : int [1:5] 0 2 4 5 7
#>   ..@ Dim     : int [1:2] 10 4
#>   ..@ Dimnames:List of 2
#>   .. ..$ : NULL
#>   .. ..$ : NULL
#>   ..@ x       : num [1:7] 3 1 6 7 5 4 2
#>   ..@ factors : list()
```

Internal structure of a dgCMatrix

```
dgC_mat2 <- as(mat2, "dgCMatrix")
str(dgC_mat2)
#> Formal class 'dgCMatrix' [package
"Matrix"] with 6 slots
#>     ..@ i        : int [1:7] 0 3 3 7 2 3 8
#>     ..@ p        : int [1:5] 0 2 4 5 7
#>     ..@ Dim      : int [1:2] 10 4
#>     ..@ Dimnames:List of 2
#>     .. ..$ : NULL
#>     .. ..$ : NULL
#>     ..@ x        : num [1:7] 3 1 6 7 5 4 2
#>     ..@ factors : list()
```

p	0	2	4	5	7

Column Pointers

	0	1	2	3	4	5	6
x	3	1	6	7	5	4	1
i	0	3	3	7	2	3	8

Values (x)
Row index (i)

Matrix

Internal structure of a dgCMatrix

	0	1	2	3
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				

`Matrix`

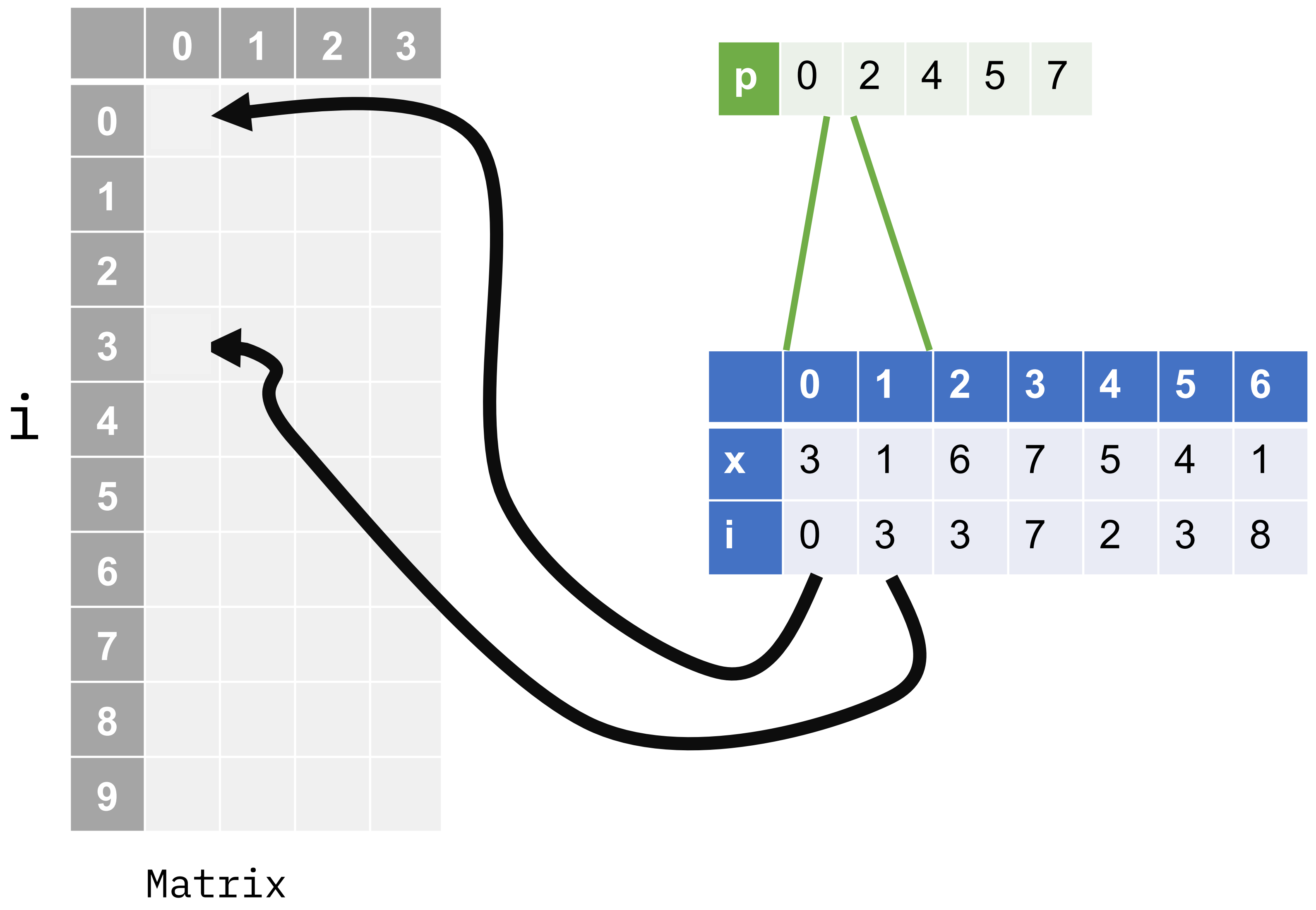
p	0	2	4	5	7

`Column Pointers`

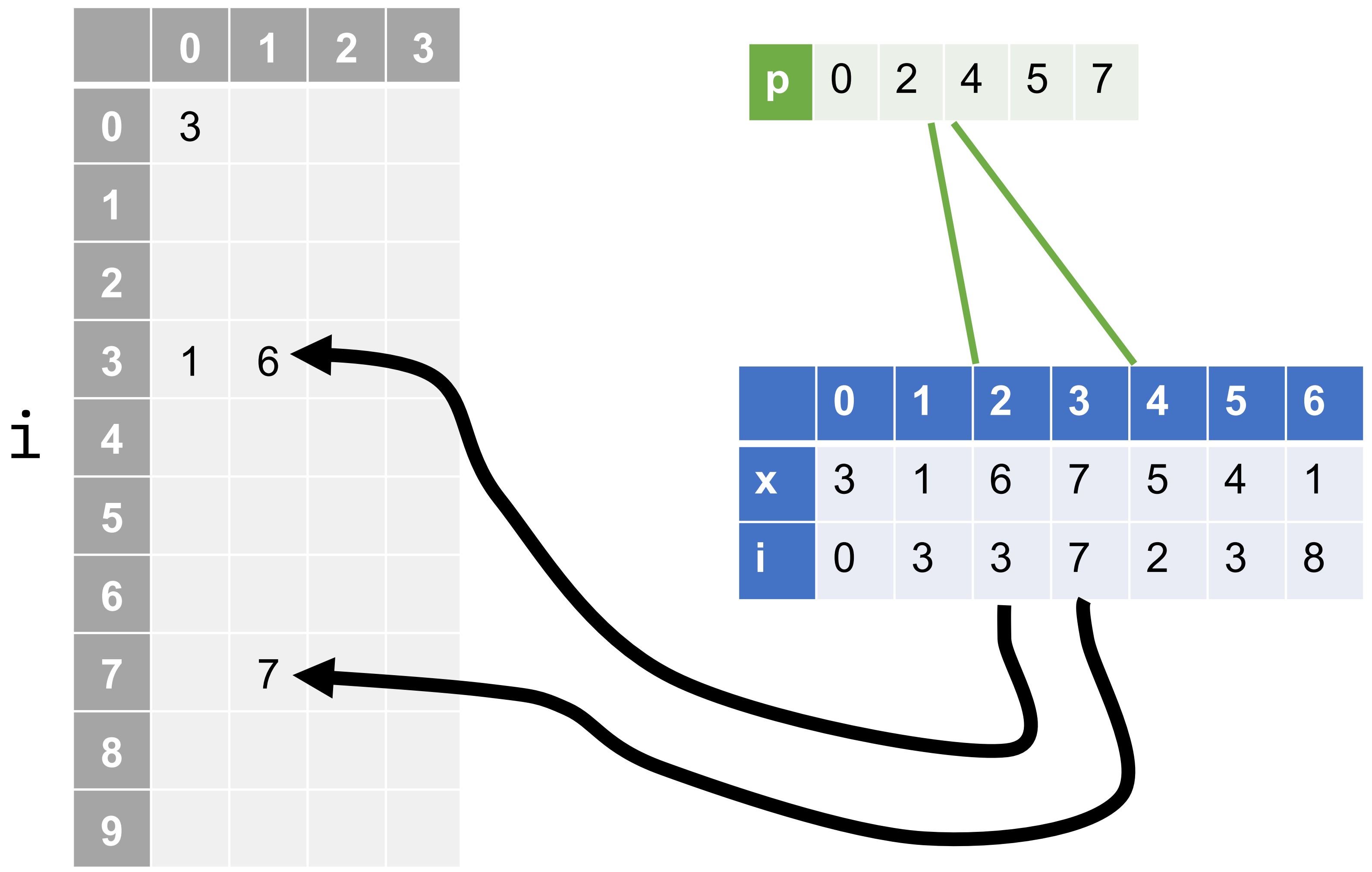
	0	1	2	3	4	5	6
x	3	1	6	7	5	4	1
i	0	3	3	7	2	3	8

```
Values (x)
Row index (i)
```

Internal structure of a dgCMatrix



Internal structure of a dgCMatrix



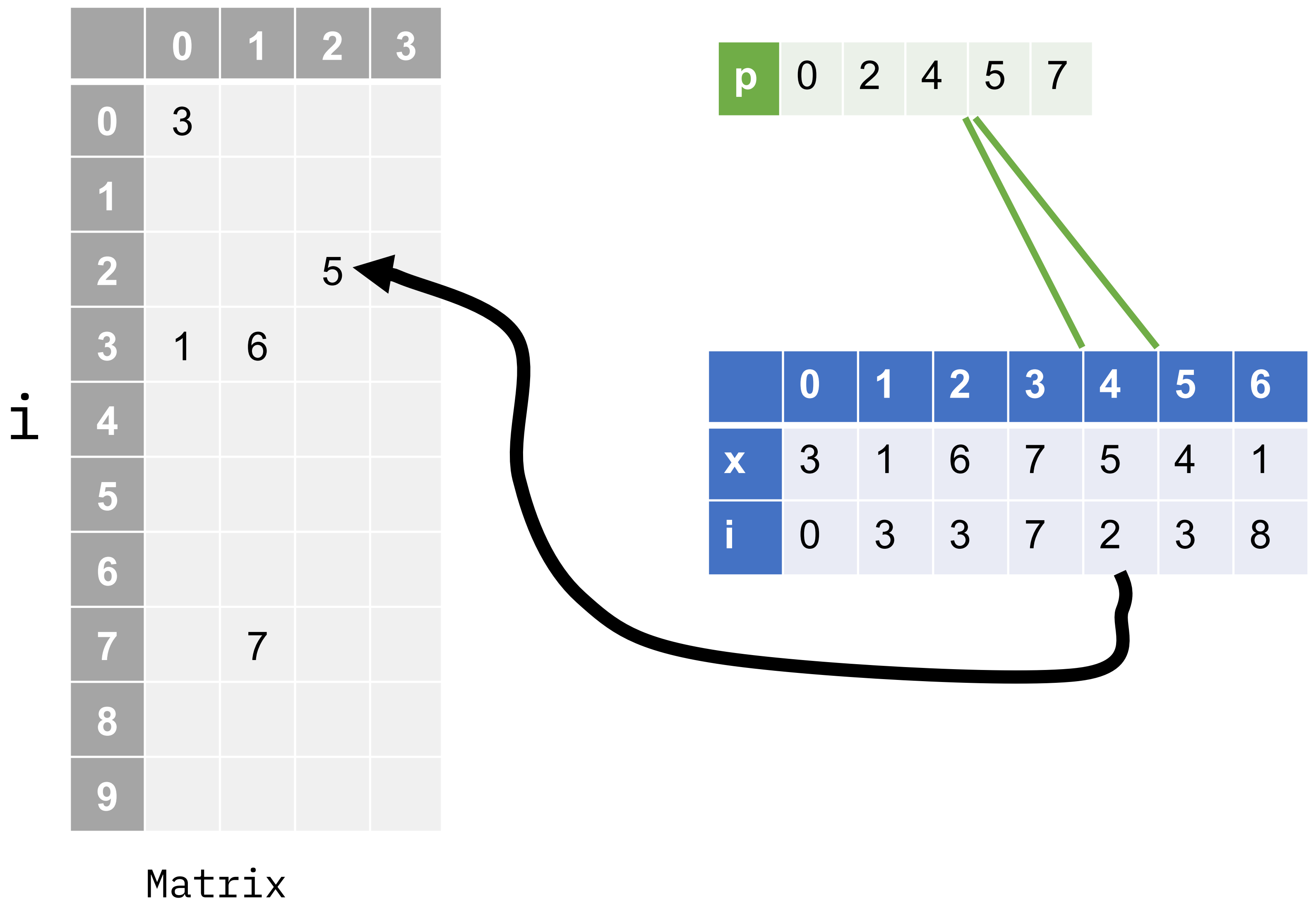
	0	1	2	3
0	3			
1				
2				
3	1	6		
4				
5				
6				
7		7		
8				
9				

Matrix

p	0	2	4	5	7

	0	1	2	3	4	5	6
x	3	1	6	7	5	4	1
i	0	3	3	7	2	3	8

Internal structure of a dgCMatrix

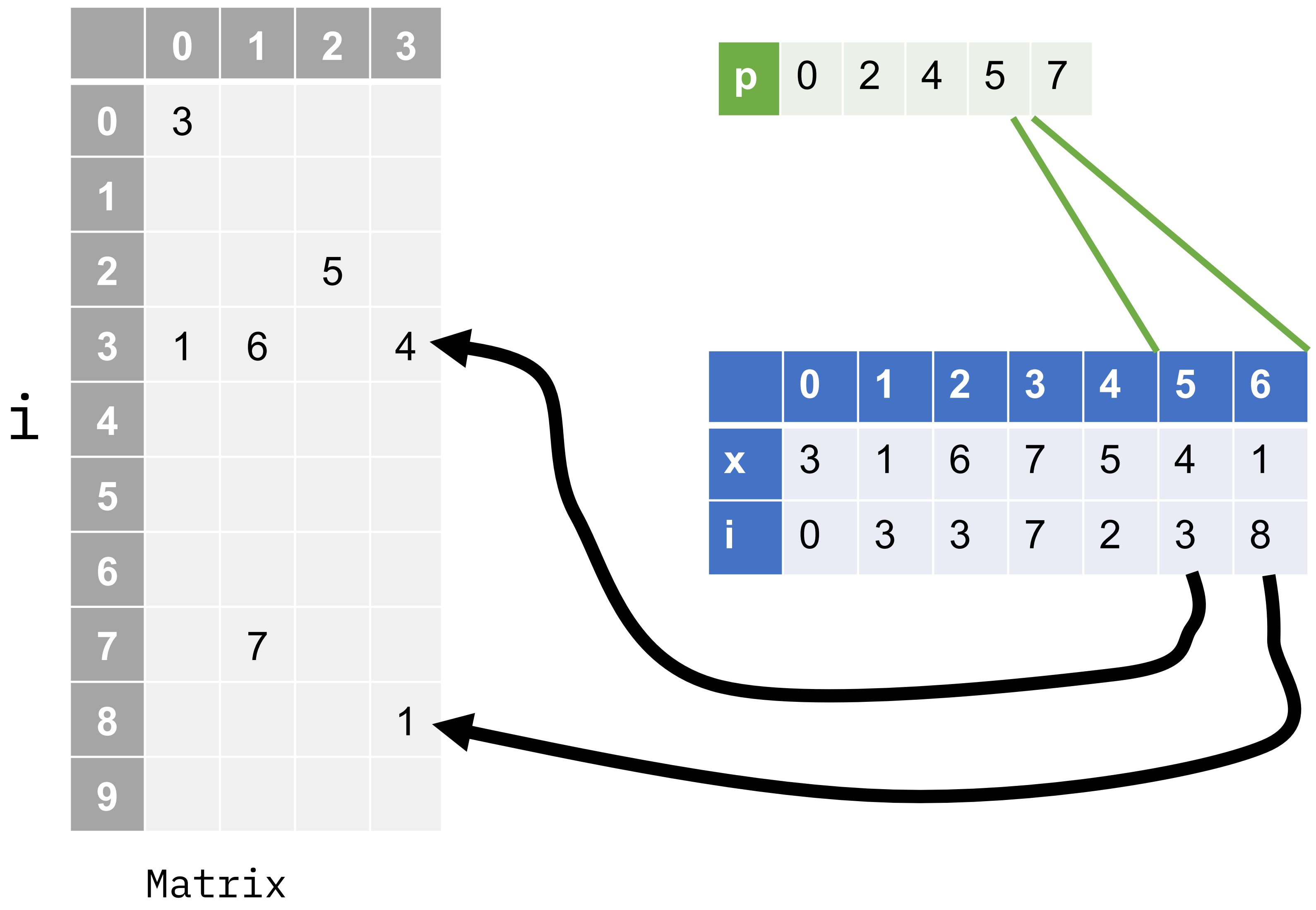


	0	1	2	3
0	3			
1				
2			5	
3	1	6		
4				
5				
6				
7		7		
8				
9				

p	0	2	4	5	7

	0	1	2	3	4	5	6
x	3	1	6	7	5	4	1
i	0	3	3	7	2	3	8

Internal structure of a dgCMatrix



	0	1	2	3
0	3			
1				
2			5	
3	1	6		4
4				
5				
6				
7		7		
8				1
9				

p	0	2	4	5	7

	0	1	2	3	4	5	6
x	3	1	6	7	5	4	1
i	0	3	3	7	2	3	8

Triplet sparse format achieves less compression

`dgTMatrix`

Row	Column	Value
0	0	3
3	0	1
3	1	6
7	1	7
2	2	5
3	3	4
8	3	2

```r
se <- TENxPBMCData::TENxPBMCData("pbmc4k")
large_mat <- as.matrix(assay(se))

pryr::object_size(large_mat)
#> 587 MB


pryr::object_size(as(large_mat, "dgCMatrix"))
#> 71.2 MB


pryr::object_size(as(large_mat, "dgTMatrix"))
#> 94.1 MB
```

sparseMatrixStats Implementation

```
// [[Rcpp::export]]
NumericVector dgCMatrix_colMeans2(S4 matrix, bool na_rm){
  return reduce_matrix_double(matrix, na_rm,
              [](auto values, auto row_indices, int number_of_zeros) -> double{
    return sp_mean(values, number_of_zeros);
  });
}
```

The values for each column are handled as iterators

```
template<typename Iterator>
inline double sp_mean(Iterator values, int number_of_zeros){
  LDOUBLE sum = 0.0;
  int size = number_of_zeros;
  for(double d : values){
    R_CHECK_USER_INTERRUPT(++size);
    sum += d;
  }
  if(NumericVector::is_na(sum)){
    return sum;
  }else if(size == 0){
    return R_NaN;
  }else{
    return sum / size;
  }
}
```

C++14 Lambda Expressions + auto reduce boilerplate code a lot

```cpp
// [[Rcpp::export]]
NumericVector dgCMatrix_colMeans2(S4 matrix, bool na_rm){
  return reduce_matrix_double(matrix, na_rm,
              [](auto values, auto row_indices,
                  int number_of_zeros) -> double{
    return sp_mean(values, number_of_zeros);
  });
}
```

Lambda Expression
C++11 Feature

Templated Lambda / auto-template
parameter C++14 feature

Rest of the implementation

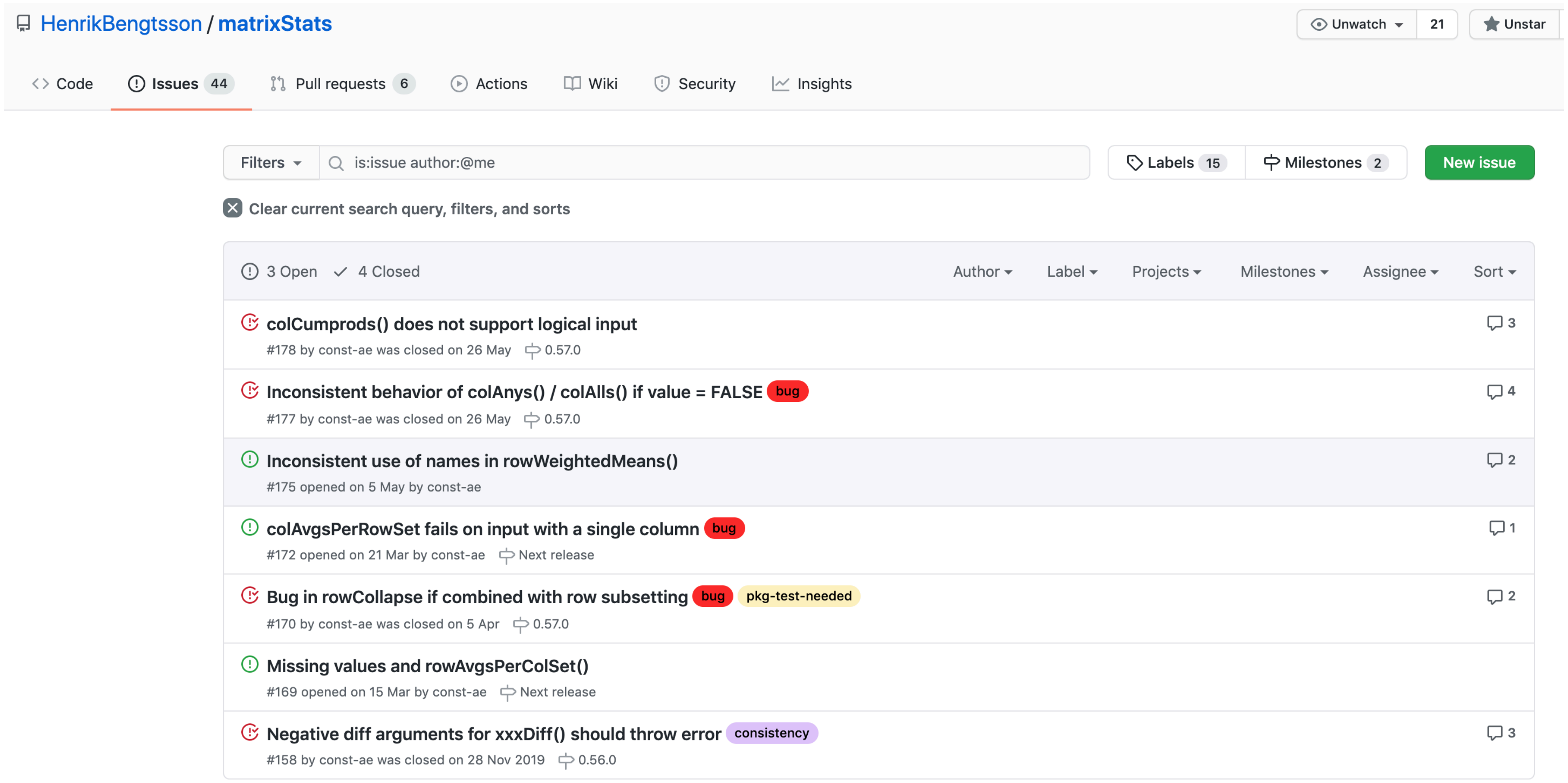
- The other 38 column functions look *kind of* similar
- Challenge how to test this large number of functions
  - 1438 unit tests

~160 individual unit tests run on 9 different input matrices

```r
descriptions <- list("diverse",
                     "named",
                     "zero row",
                     "zero col",
                     "empty",
                     "only zeros inside",
                     "numerical precision challenge",
                     "dense matrix",
                     "plus/minus Inf")
```

```r
test_that("rowSums works", {
  sp_mat2 <- t(sp_mat)
  expect_equal(rowSums2(sp_mat2), matrixStats::colSums2(mat))
  expect_equal(rowSums2(sp_mat2, na.rm=TRUE), matrixStats::colSums2(mat, na.rm=TRUE))
  expect_equal(rowSums2(sp_mat2, cols = row_subset, rows = col_subset), matrixStats::colSums2(mat, rows = row_subset, cols = col_subset))
})



test_that("colMeans works", {
  expect_equal(colMeans2(sp_mat), matrixStats::colMeans2(mat))
  expect_equal(colMeans2(sp_mat, na.rm=TRUE), matrixStats::colMeans2(mat, na.rm=TRUE))
  expect_equal(colMeans2(sp_mat, rows = row_subset, cols = col_subset), matrixStats::colMeans2(mat, rows = row_subset, cols = col_subset))
})

test_that("rowMeans works", {
  sp_mat2 <- t(sp_mat)
  expect_equal(rowMeans2(sp_mat2), matrixStats::colMeans2(mat))
  expect_equal(rowMeans2(sp_mat2, na.rm=TRUE), matrixStats::colMeans2(mat, na.rm=TRUE))
  expect_equal(rowMeans2(sp_mat2, cols = row_subset, rows = col_subset), matrixStats::colMeans2(mat, rows = row_subset, cols = col_subset))
})

test_that("colMedians works", {
  expect_equal(colMedians(sp_mat), matrixStats::colMedians(mat))
  expect_equal(colMedians(sp_mat, na.rm=TRUE), matrixStats::colMedians(mat, na.rm=TRUE))
  expect_equal(colMedians(sp_mat, rows = row_subset, cols = col_subset), matrixStats::colMedians(mat, rows = row_subset, cols = col_subset))
})



test_that("colVars works", {
  expect_equal(colVars(sp_mat), matrixStats::colVars(mat))
  expect_equal(colVars(sp_mat, na.rm=TRUE), matrixStats::colVars(mat, na.rm=TRUE))
  expect_equal(colVars(sp_mat, rows = row_subset, cols = col_subset), matrixStats::colVars(mat, rows = row_subset, cols = col_subset))
})

test_that("rowVars works", {
  sp_mat2 <- t(sp_mat)
  expect_equal(rowVars(sp_mat2), matrixStats::colVars(mat))
  expect_equal(rowVars(sp_mat2, na.rm=TRUE), matrixStats::colVars(mat, na.rm=TRUE))
  expect_equal(rowVars(sp_mat2, cols = row_subset, rows = col_subset), matrixStats::colVars(mat, rows = row_subset, cols = col_subset))
```

Reporting back upstream



HenrikBengtsson / matrixStats

is:issue author:@me

Clear current search query, filters, and sorts

3 Open ✓ 4 Closed

colCumprods() does not support logical input
#178 by const-ae was closed on 26 May 0.57.0 — 3

Inconsistent behavior of colAnys() / colAlls() if value = FALSE bug
#177 by const-ae was closed on 26 May 0.57.0 — 4

Inconsistent use of names in rowWeightedMeans()
#175 opened on 5 May by const-ae — 2

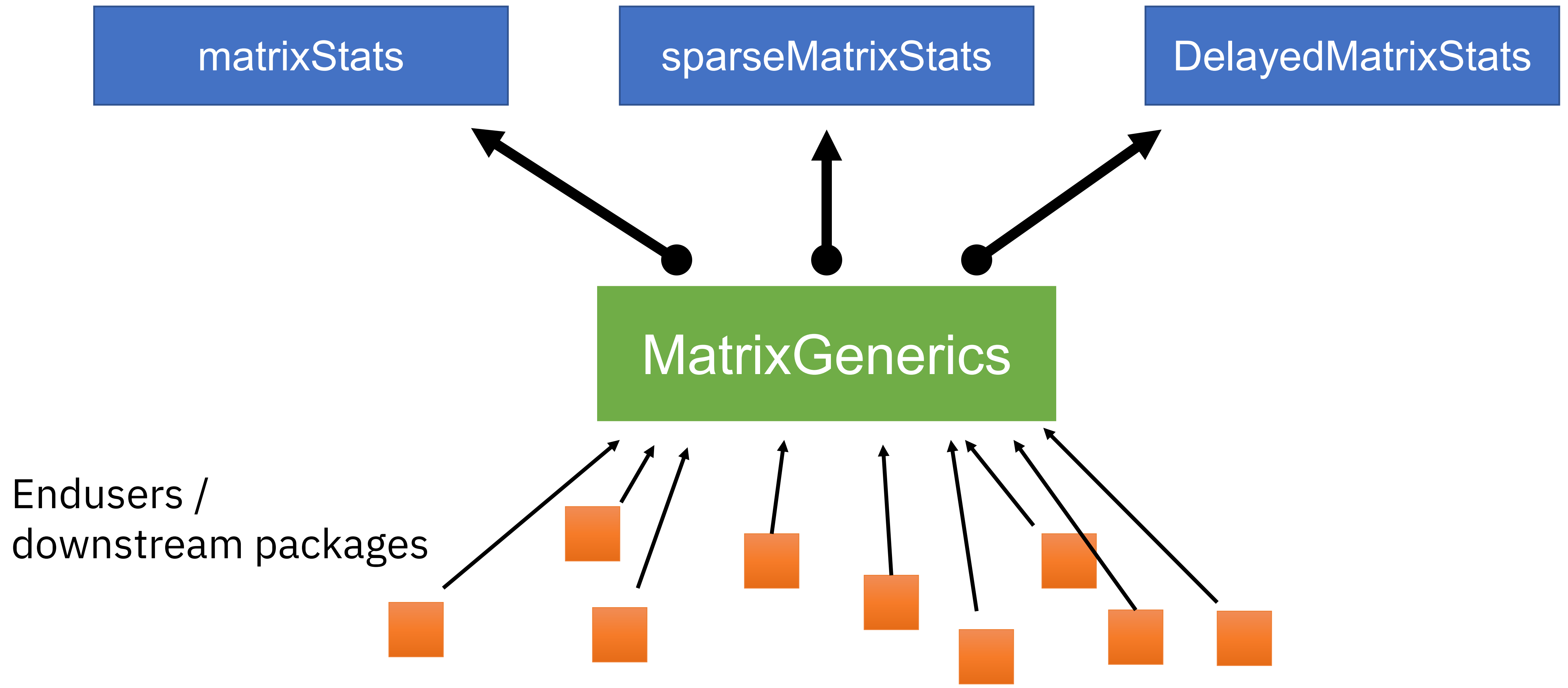
colAvgsPerRowSet fails on input with a single column bug
#172 opened on 21 Mar by const-ae Next release — 1

Bug in rowCollapse if combined with row subsetting bug pkg-test-needed
#170 by const-ae was closed on 5 Apr 0.57.0 — 2

Missing values and rowAvgsPerColSet()
#169 opened on 15 Mar by const-ae Next release

Negative diff arguments for xxxDiff() should throw error consistency
#158 by const-ae was closed on 28 Nov 2019 0.56.0 — 3

MatrixGenerics provide S4 functions to call the appropriate package for any matrix type



Thanks to…

- Hervé Pagès
- Pete Hickey
- Henrik Bengtsson
- Aaron Lun

- Lori Shepherd, Michael Lawrence and Martin Morgan for submission review
- Bioconductor for the community and infrastructure
- Mike Smith for inviting me to give a talk