Matrix operations

Cecina Babich Morrow

2023-11-28

Matrices

A matrix in R is a two-dimensional data structure consisting of columns and rows. They can be constructed using the `matrix()` function, with entries specified by column by default.

Numerical stability

R has an error of order 10^{-16}. The `all.equal()` function checks for "near equality" up to a given tolerance, set to 1.5×10^{-8} by default, which we can compare to `==`:

```
one_plus <- 1 + 1e-15
one <- 1
one_plus
```

```
## [1] 1
```

```
one
```

```
## [1] 1
```

```
one_plus == one
```

```
## [1] FALSE
```

```
all.equal(one_plus, one)
```

```
## [1] TRUE
```

R can also store integers as a "long" type, as compared to the numeric type:

```
class(1)
```

```
## [1] "numeric"
```

```
class(1L)
```

```
## [1] "integer"
```

Dense matrices

In the `Matrix` package, `dgeMatrix` is the S4 class for dense matrices.

```
library(Matrix)
dense_matrix <- Matrix(c(1, 2, 4, 3, 5, 8, 7, 7, 7), nrow=3, ncol=3)
class(dense_matrix)
```

```
## [1] "dgeMatrix"
## attr(,"package")
## [1] "Matrix"
```

The following code demonstrates some basic matrix operations for dense matrices:

```
# Invert a matrix:
solve(dense_matrix)
```

```
## 3 x 3 Matrix of class "dgeMatrix"
##            [,1]       [,2]       [,3]
## [1,]  3.0000000 -5.0000000  2.0000000
## [2,] -2.0000000  3.0000000 -1.0000000
## [3,]  0.5714286 -0.5714286  0.1428571
```

```
# Find the rank of a matrix:
rankMatrix(dense_matrix)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 6.661338e-16
```

```
# Give the reciprocal condition number (reciprocal of product of norm of matrix and norm of inverse):
rcond(dense_matrix)
```

```
## [1] 0.005555556
```

```
1 / (norm(dense_matrix) * norm(solve(dense_matrix)))
```

```
## [1] 0.005555556
```

Sparse matrices

Sparse matrices are matrices where most entries are 0. They are typically stored as class `dgCMatrix`, which are stored in a compressed, sparse, column-oriented format to save memory.

```r
set.seed(17)
nrows <- 1000
ncols <- 1000
vals <- sample(x=c(0, 1, 2), prob=c(0.98, 0.01, 0.01), size=nrows*ncols, replace=TRUE)
sparse_matrix <- Matrix(vals, nrow=nrows, ncol=ncols, sparse=TRUE)
class(sparse_matrix)
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

The following code demonstrates some basic matrix operations for sparse matrices:

```r
# Invert a matrix:
sparse_inverse <- solve(sparse_matrix)
class(sparse_inverse) # still dgCMatrix
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

```r
# Multiply by a vector:
class(sparse_inverse %*% c(1, rep(0, 998), 1)) # dgeMatrix
```

```
## [1] "dgeMatrix"
## attr(,"package")
## [1] "Matrix"
```

```r
# Multiply by that vector represented as a sparse matrix:
class(sparse_inverse %*% Matrix(c(1, rep(0, 998), 1), nrow=1000, ncol=1, sparse=TRUE))
```

```
## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"
```

Dependency graphs

We can use symmetric sparse matrices as adjacency graphs for graphs:

```r
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```r
sym_sparse <- rsparsematrix(nrow = 10,
                            ncol = 10,
                            density = 0.1,
                            symmetric = TRUE)
adj_graph <- graph_from_adjacency_matrix(sym_sparse, weighted = TRUE)
plot(adj_graph)
```

```
## Warning in v(graph): Non-positive edge weight found, ignoring all weights
## during graph layout.
```