Package ‘RSpectra’

October 12, 2022

Type Package

Title Solvers for Large-Scale Eigenvalue and SVD Problems

Version 0.16-1

Date 2022-04-24

Description R interface to the 'Spectra' library

<https://spectralib.org/> for large-scale eigenvalue and SVD problems. It is typically used to compute a few eigenvalues/vectors of an n by n matrix, e.g., the k largest eigenvalues, which is usually more efficient than eigen() if k << n. This package provides the 'eigs()' function that does the similar job as in 'Matlab', 'Octave', 'Python SciPy' and 'Julia'. It also provides the 'svds()' function to calculate the largest k singular values and corresponding singular vectors of a real matrix. The matrix to be computed on can be dense, sparse, or in the form of an operator defined by the user.

License MPL (>= 2)

URL https://github.com/yixuan/RSpectra

BugReports https://github.com/yixuan/RSpectra/issues

Depends R (>= 3.0.2)

Imports Matrix (>= 1.1-0), Rcpp (>= 0.11.5)

Suggests knitr, rmarkdown, prettydoc

LinkingTo Rcpp, RcppEigen (>= 0.3.3.3.0)

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.1.2

NeedsCompilation yes

Author Yixuan Qiu [aut, cre],
Jiali Mei [aut] (Function interface of matrix operation),
Gael Guennebaud [ctb] (Eigenvalue solvers from the 'Eigen' library),
Jitse Niesen [ctb] (Eigenvalue solvers from the 'Eigen' library)

Maintainer Yixuan Qiu <yixuan.qiu@cos.name>

Repository CRAN

Date/Publication 2022-04-24 19:50:02 UTC
R topics documented:

- `eigs` ................................................. 2
- `svds` ............................................. 6

Index

| eigs | Find a Specified Number of Eigenvalues/vectors of a Square Matrix |

Description

Given an $n \times n$ matrix $A$, function `eigs()` can calculate a specified number of eigenvalues and eigenvectors of $A$. Users can specify the selection criterion by argument `which`, e.g., choosing the $k$ largest or smallest eigenvalues and the corresponding eigenvectors.

Currently `eigs()` supports matrices of the following classes:

- `matrix` The most commonly used matrix type, defined in the `base` package.
- `dgeMatrix` General matrix, equivalent to `matrix`, defined in the `Matrix` package.
- `dgCMatrix` Column oriented sparse matrix, defined in the `Matrix` package.
- `dgRMatrix` Row oriented sparse matrix, defined in the `Matrix` package.
- `dsyMatrix` Symmetric matrix, defined in the `Matrix` package.
- `dsCMatrix` Symmetric column oriented sparse matrix, defined in the `Matrix` package.
- `dsRMatrix` Symmetric row oriented sparse matrix, defined in the `Matrix` package.
- `function` Implicitly specify the matrix through a function that has the effect of calculating $f(x) = Ax$. See section Function Interface for details.

`eigs_sym()` assumes the matrix is symmetric, and only the lower triangle (or upper triangle, which is controlled by the argument `lower`) is used for computation, which guarantees that the eigenvalues and eigenvectors are real, and in general results in faster and more stable computation. One exception is when $A$ is a function, in which case the user is responsible for the symmetry of the operator.

`eigs_sym()` supports "matrix", "dgeMatrix", "dgCMatrix", "dgRMatrix" and "function" typed matrices.

Usage

```
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'matrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dgeMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dgCMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dgRMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dsyMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dsCMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dsRMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)
```
## S3 method for class 'dgCMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dsCMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dgRMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class 'dsRMatrix'
eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

## S3 method for class `\"function\"`
eigs(
  A,
  k,
  which = "LM",
  sigma = NULL,
  opts = list(),
  ..., 
  n = NULL,
  args = NULL
)

eigs_sym(A, k, which = "LM", sigma = NULL, opts = list(),
  lower = TRUE, ...)

## S3 method for class `\"function\"`
eigs_sym(
  A,
  k,
  which = "LM",
  sigma = NULL,
  opts = list(),
  lower = TRUE,
  ..., 
  n = NULL,
  args = NULL
)

### Arguments

- **A**: The matrix whose eigenvalues/vectors are to be computed. It can also be a function which receives a vector \( x \) and calculates \( Ax \). See section **Function Interface** for details.

- **k**: Number of eigenvalues requested.

- **which**: Selection criterion. See **Details** below.

- **sigma**: Shift parameter. See section **Shift-And-Invert Mode**.
opts            Control parameters related to the computing algorithm. See Details below.
...            Arguments for specialized S3 function calls, for example lower, n and args.
n            Only used when A is a function, to specify the dimension of the implicit matrix. See section Function Interface for details.
args            Only used when A is a function. This argument will be passed to the A function when it is called. See section Function Interface for details.
lower            For symmetric matrices, should the lower triangle or upper triangle be used.

Details
The which argument is a character string that specifies the type of eigenvalues to be computed. Possible values are:
"LM" The $k$ eigenvalues with largest magnitude. Here the magnitude means the Euclidean norm of complex numbers.
"SM" The $k$ eigenvalues with smallest magnitude.
"LR" The $k$ eigenvalues with largest real part.
"SR" The $k$ eigenvalues with smallest real part.
"LI" The $k$ eigenvalues with largest imaginary part.
"SI" The $k$ eigenvalues with smallest imaginary part.
"LA" The $k$ largest (algebraic) eigenvalues, considering any negative sign.
"SA" The $k$ smallest (algebraic) eigenvalues, considering any negative sign.
"BE" Compute $k$ eigenvalues, half from each end of the spectrum. When $k$ is odd, compute more from the high and then from the low end.

eigs() with matrix types "matrix", "dgeMatrix", "dgCMatrix" and "dgRMatrix" can use "LM", "SM", "LR", "SR", "LI" and "SI".
eigs_sym() with all supported matrix types, and eigs() with symmetric matrix types ("dsyMatrix", "dsCMatrix", and "dsRMatrix") can use "LM", "SM", "LA", "SA" and "BE".
The opts argument is a list that can supply any of the following parameters:
ncv            Number of Lanzcos basis vectors to use. More vectors will result in faster convergence, but with greater memory use. For general matrix, ncv must satisfy $k + 2 \leq ncv \leq n$, and for symmetric matrix, the constraint is $k < ncv \leq n$. Default is $\min(n, \max(2k+1, 20))$.
tol            Precision parameter. Default is 1e-10.
maxitr            Maximum number of iterations. Default is 1000.
retvec            Whether to compute eigenvectors. If FALSE, only calculate and return eigenvalues.
initvec            Initial vector of length $n$ supplied to the Arnoldi/Lanczos iteration. It may speed up the convergence if initvec is close to an eigenvector of $A$.

Value
A list of converged eigenvalues and eigenvectors.
values            Computed eigenvalues.
vectors            Computed eigenvectors. vectors[, j] corresponds to values[j].
nconv            Number of converged eigenvalues.
niter            Number of iterations used in the computation.
nops            Number of matrix operations used in the computation.
Shift-And-Invert Mode

The sigma argument is used in the shift-and-invert mode. When sigma is not NULL, the selection criteria specified by argument which will apply to

\[
\frac{1}{\lambda - \sigma}
\]

where \(\lambda\)'s are the eigenvalues of \(A\). This mode is useful when users want to find eigenvalues closest to a given number. For example, if \(\sigma = 0\), then which = "LM" will select the largest values of \(1/|\lambda|\), which turns out to select eigenvalues of \(A\) that have the smallest magnitude. The result of the shift-and-invert mode can be found in the SciPy document, https://docs.scipy.org/doc/scipy/tutorial/arpack.html.

Function Interface

The matrix \(A\) can be specified through a function with the definition

```r
function(x, args)
{
  ## should return A %*% x
}
```

which receives a vector \(x\) as an argument and returns a vector of the same length. The function should have the effect of calculating \(Ax\), and extra arguments can be passed in through the args parameter. In `eigs()`, user should also provide the dimension of the implicit matrix through the argument \(n\).

Author(s)

Yixuan Qiu https://statr.me
Jiali Mei <vermouthmj1@gmail.com>

See Also

eigen(), svd(), svds()

Examples

```r
library(Matrix)

n = 20
k = 5

## general matrices have complex eigenvalues
set.seed(111)
A1 = matrix(rnorm(n^2), n) ## class "matrix"
A2 = Matrix(A1) ## class "dgeMatrix"
```
eigs(A1, k)
eigs(A2, k, opts = list(retvec = FALSE))  ## eigenvalues only

## Sparse matrices
A1[sample(n^2, n^2 / 2)] = 0
A3 = as(A1, "dgCMatrix")
A4 = as(A1, "dgRMatrix")
eigs(A3, k)
eigs(A4, k)

## Function interface
f = function(x, args)
{
  as.numeric(args %*% x)
}
eigs(f, k, n = n, args = A3)

## Symmetric matrices have real eigenvalues
A5 = crossprod(A1)
eigs_sym(A5, k)

## Find the smallest (in absolute value) k eigenvalues of A5
eigs_sym(A5, k, which = "SM")

## Another way to do this: use the sigma argument
eigs_sym(A5, k, sigma = 0)

## The results should be the same,
## but the latter method is far more stable on large matrices

---

svds

Find the Largest k Singular Values/Vectors of a Matrix

---

**Description**

Given an \( m \) by \( n \) matrix \( A \), function `svds()` can find its largest \( k \) singular values and the corresponding singular vectors. It is also called the Truncated SVD or Partial SVD since it only calculates a subset of the whole singular triplets.

Currently `svds()` supports matrices of the following classes:

### matrix
The most commonly used matrix type, defined in the **base** package.

### dgeMatrix
General matrix, equivalent to matrix, defined in the **Matrix** package.

### dgCMatrix
Column oriented sparse matrix, defined in the **Matrix** package.

### dgRMatrix
Row oriented sparse matrix, defined in the **Matrix** package.

### dsyMatrix
Symmetric matrix, defined in the **Matrix** package.

### dsCMatrix
Symmetric column oriented sparse matrix, defined in the **Matrix** package.

### dsRMatrix
Symmetric row oriented sparse matrix, defined in the **Matrix** package.

### function
Implicitly specify the matrix through two functions that calculate \( f(x) = Ax \) and \( g(x) = A'x \). See section **Function Interface** for details.
Note that when \( A \) is symmetric and positive semi-definite, SVD reduces to eigen decomposition, so you may consider using `eigs()` instead. When \( A \) is symmetric but not necessarily positive semi-definite, the left and right singular vectors are the same as the left and right eigenvectors, but the singular values and eigenvalues will not be the same. In particular, if \( \lambda \) is a negative eigenvalue of \( A \), then \(|\lambda|\) will be the corresponding singular value.

Usage

```r
svds(A, k, nu = k, nv = k, opts = list(), ...)
```

## S3 method for class 'matrix'
```r
svds(A, k, nu = k, nv = k, opts = list(), ...)
```

## S3 method for class 'dgeMatrix'
```r
svds(A, k, nu = k, nv = k, opts = list(), ...)
```

## S3 method for class 'dgCMatrix'
```r
svds(A, k, nu = k, nv = k, opts = list(), ...)
```

## S3 method for class 'dgRMatrix'
```r
svds(A, k, nu = k, nv = k, opts = list(), ...)
```

## S3 method for class 'dsyMatrix'
```r
svds(A, k, nu = k, nv = k, opts = list(), ...)
```

## S3 method for class 'dsCMatrix'
```r
svds(A, k, nu = k, nv = k, opts = list(), ...)
```

## S3 method for class 'dsRMatrix'
```r
svds(A, k, nu = k, nv = k, opts = list(), ...)
```

## S3 method for class `function`
```r
svds(A, k, nu = k, nv = k, opts = list(), ..., Atrans, dim, args = NULL)
```

Arguments

- **A**
  The matrix whose truncated SVD is to be computed.

- **k**
  Number of singular values requested.

- **nu**
  Number of left singular vectors to be computed. This must be between 0 and \( k \).

- **nv**
  Number of right singular vectors to be computed. This must be between 0 and \( k \).

- **opts**
  Control parameters related to the computing algorithm. See `Details` below.

- **...**
  Arguments for specialized S3 function calls, for example `Atrans`, `dim` and `args`.

- **Atrans**
  Only used when \( A \) is a function. \( A \) is a function that calculates the matrix multiplication \( Ax \), and \( Atrans \) is a function that calculates the transpose multiplication \( A^\prime x \).
dim Only used when A is a function, to specify the dimension of the implicit matrix. A vector of length two.

args Only used when A is a function. This argument will be passed to the A and Atrans functions.

Details

The opts argument is a list that can supply any of the following parameters:

ncv Number of Lanczos basis vectors to use. More vectors will result in faster convergence, but with greater memory use. ncv must be satisfy $k < ncv \leq p$ where $p = \min(m, n)$. Default is $\min(p, \max(2*k+1, 20))$.

tol Precision parameter. Default is 1e-10.

maxitr Maximum number of iterations. Default is 1000.

center Either a logical value (TRUE/FALSE), or a numeric vector of length n. If a vector c is supplied, then SVD is computed on the matrix $A - 1c'$, in an implicit way without actually forming this matrix. center = TRUE has the same effect as center = colMeans(A). Default is FALSE.

scale Either a logical value (TRUE/FALSE), or a numeric vector of length n. If a vector s is supplied, then SVD is computed on the matrix $(A - 1c')S$, where c is the centering vector and $S = diag(1/s)$. If scale = TRUE, then the vector s is computed as the column norm of $A - 1c'$. Default is FALSE.

Value

A list with the following components:

d A vector of the computed singular values.

u An m by nu matrix whose columns contain the left singular vectors. If nu == 0, NULL will be returned.

v An n by nv matrix whose columns contain the right singular vectors. If nv == 0, NULL will be returned.

nconv Number of converged singular values.

niter Number of iterations used.

nops Number of matrix-vector multiplications used.

Function Interface

The matrix A can be specified through two functions with the following definitions

```r
A <- function(x, args) {
  ## should return A %*% x
}
Atrans <- function(x, args)
```
They receive a vector \( x \) as an argument and returns a vector of the proper dimension. These two functions should have the effect of calculating \( Ax \) and \( A'x \) respectively, and extra arguments can be passed in through the \( \text{args} \) parameter. In \texttt{svds()}\(,\) user should also provide the dimension of the implicit matrix through the argument \( \text{dim} \).

The function interface does not support the center and scale parameters in \( \text{opts} \).

**Author(s)**

Yixuan Qiu <https://statr.me>

**See Also**

\texttt{eigen()}, \texttt{svd()}, \texttt{eigs()}.

**Examples**

```
m = 100
n = 20
k = 5
set.seed(111)
A = matrix(rnorm(m * n), m)
svds(A, k)
svds(t(A), k, nu = 0, nv = 3)

## Sparse matrices
library(Matrix)
A[sample(m * n, m * n / 2)] = 0
Asp1 = as(A, "dgCMatrix")
Asp2 = as(A, "dgRMatrix")
svds(Asp1, k)
svds(Asp2, k, nu = 0, nv = 0)

## Function interface
Af = function(x, args)
{
  as.numeric(args %*% x)
}
Atf = function(x, args)
{
  as.numeric(crossprod(args, x))
}
svds(Af, k, Atrans = Atf, dim = c(m, n), args = Asp1)
```
Index

* array
  eigs, 2
  svds, 6

eigen, 5, 9
eigs, 2, 7, 9
eigs_sym(eigs), 2

svd, 5, 9
svds, 5, 6