Package ‘klin’

February 20, 2015

Type Package
Title Linear equations with Kronecker structure
Version 2007-02-05
Date 2007-02-05
Author Tamas K Papp <tpapp@princeton.edu>
Maintainer ORPHANED
Depends R (>= 2.0.0), Matrix (>= 0.9975-8)
Description The package implements efficient ways to evaluate and
solve equations of the form Ax=b, where A is a kronecker
product of matrices. Functions to solve least squares problems
of this type are also included.
License GPL (>= 2)
X-CRAN-Comment Orphaned on 2011-08-11 as requested by the original
maintainer, Tamas K Papp <tpapp@princeton.edu>.
Repository CRAN
NeedsCompilation no
X-CRAN-Original-Maintainer Tamas K Papp <tpapp@princeton.edu>
Date/Publication 2012-10-29 08:59:03

R topics documented:

  klin-package ......................................................... 2
  incseq ............................................................... 2
  klin.eval ........................................................... 3
  klin.klist .......................................................... 4
  klin.ls ............................................................... 5
  klin.preparels ...................................................... 6
  klin.solve .......................................................... 7

Index 9
klin-package

*Calculate and solve linear equations with Kronecker structure.*

**Description**

The package implements efficient ways to evaluate and solve equations of the form $Ax=b$, where $A$ is a kronecker product of matrices. Functions to solves least squares problems of this type are also included.

**Details**

<table>
<thead>
<tr>
<th>Package:</th>
<th>klin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>2007-01-08</td>
</tr>
<tr>
<td>Date:</td>
<td>2007-01-08</td>
</tr>
<tr>
<td>License:</td>
<td>GPL version 2 or newer</td>
</tr>
</tbody>
</table>

The most important functions are `klin.eval` and `klin.solve`, which evaluate $A \ %*% \ x$ or solve for $x$ in $A \ %*% \ x = b$ where $A$ is a Kronecker product and $x$ and $b$ are conforming vectors.

Convenience functions for solving least squares problems with Kronecker structure (`klin.ls` and `klin.preparels`) are also included. The function `klin.klist` forms the Kronecker product of a list of matrices.

**Author(s)**

Author and Maintainer: Tamas K Papp <tpapp@princeton.edu>

**References**


---

**incseq**

*Only returns increasing sequences*

**Description**

Given two numbers $a$ and $b$, returns $a:b$ if $a\leq b$, otherwise `numeric(0)`.

**Usage**

`incseq(a,b)`
**Arguments**

- **a**: lower endpoint, an integer
- **b**: upper endpoint, an integer

**Value**

If $a \leq b$, $a:b$, otherwise numeric(0).

**Note**

The function does not check whether a and b are indeed integers.

**Author(s)**

Tamas K Papp <tpapp@princeton.edu>

---

**klin.eval**

_Evaluate Kronecker product times a vector_

**Description**

Computes the product $A \%x% x$, where $A$ is a Kronecker product of matrices.

**Usage**

`klin.eval(A, x, transpose = FALSE)`

**Arguments**

- **A**: A list that contains the matrices, preferably of class Matrix.
- **x**: A conformable numeric vector.
- **transpose**: If TRUE, the transpose of the matrices in A is used (implemented by calling crossprod).

**Details**

The matrices in the list A should be of the class Matrix. This allows the user to take advantage of their special structure (eg sparsity).

**Value**

A vector which equals $(A[[1]] \%x% ... \%x% A[[length(A)]] \%x% x$.

**Note**

The algorithm (given in the reference) is orders of magnitude more efficient (both in terms of CPU and memory usage) than computing the Kronecker product and doing the matrix multiplication.
**Author(s)**

Tamas K Papp <tpapp@princeton.edu>

**References**


**See Also**

`klin.solve`, `klin.preparels`, `klin.ls`, `klin.klist`.

**Examples**

```r
## dimensions
n <- c(6,5,3)
m <- c(4,7,2)
## make random matrices
A <- lapply(seq_along(n),
  function(i) Matrix(rnorm(m[i]*n[i]),m[i],n[i]))
x <- rnorm(prod(n)) # make random x
b1 <- klin.klist(A) %*% x # brute force way
b2 <- klin.eval(A, x) # using klin.eval
range(b1-b2) # should be small
```

---

**klin.klist**  
*Calculates the Kronecker product of a list of matrices*

**Description**

Given a list `A` of matrices, the function calculates `A[[1]] %x% ... %x% A[[length(A)]]`.

**Usage**

`klin.klist(A)`

**Arguments**

`A`  
A list that contains the matrices, preferably of class `Matrix`.

**Value**

A matrix, the dimensions are the product of the dimensions of the matrices in `A`.

**Note**

This is merely a convenience function, it does not employ any special algorithm, just calls `%x%` repeatedly.
**klin.ls**

**Author(s)**
Tamas K Papp <tpapp@princeton.edu>

**See Also**

- klin.eval
- klin.solve
- klin.ls
- klin.preparels

---

### klin.ls

Solves a least squares problem where the matrix is a Kronecker product

### Description

Computes the least squares estimate \( \hat{x} \) which minimizes the Euclidian norm of \( A \%x\% x = b \), where \( A \) is a Kronecker product of matrices.

### Usage

```r
klin.ls(A, b)
```

### Arguments

- **A**
  A list that contains the matrices, preferably of class `matrix`, or a list of class `klin.prepls` (see Details).

- **b**
  A conformable numeric vector.

### Details

The matrices in the list \( A \) should be of the class `Matrix`. This allows the user to take advantage of their special structure (e.g., sparsity).

This function is just glue for `klin.preparels` and `klin.solve`. If you are using the same \( A \) multiple times, it is suggested that you call `klin.preparels` and save the result. This allows `Matrix` to memoize the factors of `crossprod(A[[i]])` where needed.

### Value

A numeric vector.

### Note

The algorithm (given in the reference) is orders of magnitude more efficient (both in terms of CPU and memory usage) than computing the Kronecker product and calling `crossprod` and `solve`.

### Author(s)

Tamas K Papp <tpapp@princeton.edu>
References


See Also

`klin.eval`, `klin.solve`, `klin.preparels`, `klin.klist`.

Examples

```r
## dimensions
n <- c(2,4,3)
m <- n+c(1,0,2) # we need m >= n
## make random matrices
A <- lapply(seq_along(n),
            function(i) Matrix(rnorm(m[i]*n[i]),m[i],n[i]))
b <- rnorm(prod(m)) # make random b
x <- klin.ls(A,b)
```

`klin.preparels` *Prepares matrices for the least squares solver*

Description

You should use this function whenever you are calling `klin.ls` repeatedly with the same matrices.

Usage

`klin.preparels(A)`

Arguments

- `A` A list that contains the matrices, preferably of class `Matrix`.

Details

To compute the least squares estimate, we are solving

\[(A_1 \times A_2 \times \ldots \times A_K)^T(A_1 \times A_2 \times \ldots \times A_K) = (A_1 \times A_2 \times \ldots \times A_K)\,^T b\]

However, for square $A_i$ matrices, one can premultiply both sides by the Kronecker product of the inverse of $A_i^T$ (in the corresponding place) and identity matrices, making the problem simpler. `klin.prepls` calculates the matrices needed on both sides, but does not evaluate the Kronecker product.

Value

A list of class `klin.preparels`, contains matrices for the left and right hand side.
klin.solve

Author(s)
Tamas K Papp <tpapp@princeton.edu>

See Also
klin.eval, klin.solve, klin.ls, klin.klist.

Examples
```r
## dimensions
n <- c(2,4,3)
m <- n+c(1,0,2) # we need m >= n
## make random matrices
A <- lapply(seq_along(n),
function(i) Matrix(rnorm(m[i]*n[i]),m[i],n[i]))
b <- rnorm(prod(m)) # make random b
prepa <- klin.preparels(A)
x <- klin.ls(prepa,b)
```

---

**klin.solve**

**Solve linear systems where the matrix is a Kronecker product**

**Description**
Solves the system \( A \times x = b \) for \( x \) given \( A \) and \( b \), where \( A \) is a Kronecker product of matrices.

**Usage**

```r
klin.solve(A, b)
```

**Arguments**

- `A` A list that contains the matrices, preferably of class `Matrix`.
- `b` A conformable numeric vector.

**Details**
The matrices in the list \( A \) should be square matrices of the class `Matrix`. This allows the user to take advantage of their special structure (e.g., sparsity), also, their factors will be memoized by `Matrix`.

**Value**
A numeric vector \( x \) which solves the system.

**Note**
The algorithm (given in the reference) is orders of magnitude more efficient (both in terms of CPU and memory usage) than computing the Kronecker product and calling `solve`.
Author(s)
Tamas K Papp <tpapp@princeton.edu>

References

See Also
klin.eval, klin.preparels, klin.ls, klin.klist.

Examples
```r
## dimensions
m <- c(4,7,2)
## make random matrices
A <- lapply(seq_along(m),
  function(i) Matrix(rnorm(m[i]^2),m[i],m[i]))
b <- rnorm(prod(m)) # make random b
x1 <- solve(klin.klist(A),b) # brute force way
x2 <- klin.solve(A, b) # using klin.eval
range(x1-x2) # should be small
```
Index

*Topic **arith**
  incseq, 2

*Topic **array**
  klin.eval, 3
  klin.klist, 4
  klin.ls, 5
  klin.preparels, 6
  klin.solve, 7

*Topic **package**
  klin-package, 2

incseq, 2

klin (klin-package), 2
klin-package, 2
klin.eval, 3, 5–8
klin.klist, 4, 4, 6–8
klin.ls, 4, 5, 5, 7, 8
klin.preparels, 4–6, 6, 8
klin.solve, 4–7, 7