Package ‘pbdBASE’
February 2, 2020

Type Package

Title Programming with Big Data -- Base Wrappers for Distributed Matrices

Version 0.5-2

Description An interface to and extensions for the 'PBLAS' and 'ScaLAPACK' numerical libraries. This enables R to utilize distributed linear algebra for codes written in the 'SPMD' fashion. This interface is deliberately low-level and mimics the style of the native libraries it wraps. For a much higher level way of managing distributed matrices, see the 'pbdDMAT' package.

License Mozilla Public License 2.0

Depends R (>= 3.6.0), methods

Imports utils, pbdMPI (>= 0.4-3), pbdSLAP (>= 0.2-9)

SystemRequirements OpenMPI (>= 1.5.4) on Solaris, Linux, Mac, and FreeBSD. MS-MPI (Microsoft HPC Pack 2012) or MPICH2 (>= 1.4.1p1) on Windows.

LazyLoad yes

LazyData yes

ByteCompile yes

NeedsCompilation yes

URL https://pbdr.org/

BugReports http://group.pbdr.org/

MailingList Please send questions and comments regarding pbdR to RBigData@gmail.com

Maintainer Wei-Chen Chen <wccsnow@gmail.com>

RoxygenNote 7.0.2

Author Drew Schmidt [aut], Wei-Chen Chen [aut, cre], Sebastien Lamy de la Chapelle [aut], George Ostrouchov [aut],

1
R topics documented:

pbddBASE-package .................................................. 3
BASE Global Environment ......................................... 5
base.blacs_gridinit ................................................ 6
base.crossprod ....................................................... 6
base.descinit ........................................................ 7
base.det ............................................................. 8
base.dgamx2d ....................................................... 9
base.dgesd2d ........................................................ 9
base.dhilbmk ........................................................ 10
base.dim0 ............................................................ 11
base.free_blacs_system_handle .................................. 11
base.igamx2d ........................................................ 12
base.igsum2d ........................................................ 12
base.indxg2p ........................................................ 13
base.matexp ........................................................ 14
base.maxdim ........................................................ 14
base.minctxt ....................................................... 15
base.mksubmat ...................................................... 15
base.nbd ............................................................. 16
base.numroc ........................................................ 17
base.ownany ........................................................ 18
base.pdchtri ........................................................ 19
base.pdclvar ....................................................... 20
base.pdhilbmk ..................................................... 20
base.pdmkcpn1 ..................................................... 21
base.pdmvsum ...................................................... 21
base.pdsweep ....................................................... 22
base.procgrid ....................................................... 22
base.p_matexp_pade_wrap ........................................ 23
base.p_matpow_by_squaring_wrap ................................ 24
base.rcolcpy ........................................................ 24
base.rcolcpy2 ..................................................... 25
base.redist ........................................................ 25
base.rl2blas ......................................................... 26
base.rl2insert ...................................................... 27
base.rpdgecon ...................................................... 27
base.rpdgelqf ...................................................... 28
base.rpdgels ....................................................... 28
base.rpdgemm ....................................................... 29
base.rpdgemr2d ..................................................... 29
Description

A package contains the basic methods for dealing with distributed data types, as well as the data types themselves.
This package requires an MPI library (OpenMPI, MPICH2, or LAM/MPI).

Author(s)

Drew Schmidt <wrathematics AT gmail.com>, Wei-Chen Chen, George Ostrouchov, and Pragneshkumar Patel.

References

Programming with Big Data in R Website: https://pbdr.org/

Description

The environment for the pbdBASE package where "global" variables are stored.

Usage

.pbdBASEEnv

Format

An object of class environment of length 0.

Details

The __blacs_gridinfo__ and __blacs_initialized objects are stored in this environment.
base.blacs_gridinit  Creating Grid From A System Context

Description

Creates a grid from a System Context obtained from a call to 'sys2blacs_handle'.

Usage

```r
base.blacs_gridinit(
  SYSCTX,  
  NPROW,  
  NPCOL,  
  nprocs = pbdMPI::comm.size(comm),  
  comm = .pbd_env$SPMD.CT$comm
)
```

Arguments

- **SYSCTX**: System context obtained from a call to 'sys2blacs_handle'
- **NPROW**: Number of rows in the process grid
- **NPCOL**: Number of columns in the process grid
- **nprocs**: Number of processors in the communicator
- **comm**: An MPI (not BLACS) communicator.

Value

A blacs context number

base.crossprod  crossprod

Description

Crossproduct.

Usage

```r
base.crossprod(uplo, trans, x, descx, descc)
```
Arguments

- **uplo**: Triangle whose values to use.
- **trans**: tcrossprod or crossprod.
- **x**: Matrix to crossprod.
- **descx**: ScaLAPACK descriptor array.
- **descx**: ScaLAPACK descriptor array of output.

Details

For advanced users only. See pbdDMAT for high-level functions.

---

Description

Creates ScaLAPACK descriptor array.

Usage

```r
crossprod(base.descinit(dim, bldim, ldim, ICTXT = 0))
```

Arguments

- **dim**: Global dim.
- **bldim**: Blocking dim.
- **ldim**: Local dim.
- **ICTXT**: BLACS context.

Details

For advanced users only. See pbdDMAT for high-level functions.

Value

A descriptor array.
Examples

```r
spmd.code <- 
  suppressMessages(library(pbdMPI))
  suppressMessages(library(pbdBASE))
  init.grid()

  ### Set desc.
  dim <- c(6L, 5L)
  bldim <- c(3L, 3L)
  ldim <- base.numroc(dim = dim, bldim = bldim)
  descx <- base.descinit(dim = dim, bldim = bldim, ldim = ldim)
  comm.print(descx)

  finalize()
```

```r
pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)
```

base.det  det

Description

Determinant.

Usage

```r
base.det(a, desca)
```

Arguments

- `a` : Matrix.
- `desca` : ScaLAPACK descriptor array.

Details

For advanced users only. See pbdDMAT for high-level functions.
**base.dgamx2d**

**BLACS Min**

**Description**

Min value across a process grid.

**Usage**

base.dgamx2d(ICTX, SCOPE, m, n, x, lda, RDEST, CDEST)

base.igamm2d(ICTX, SCOPE, m, n, x, lda, RDEST, CDEST)

base.dgamn2d(ICTX, SCOPE, m, n, x, lda, RDEST, CDEST)

**Arguments**

ICTXT  
BLACS ICTXT.

SCOPE  
Rows, cols, or both.

m, n  
Problem size.

x  
Local values.

lda  
Leading dimension.

RDEST  
Row destination.

CDEST  
Col destination.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

**base.dgesd2d**  

**BLACS Point to Point**

**Description**

Sent value across a process grid.

**Usage**

base.dgesd2d(ICTX, SCOPE, m, n, x, lda, RDEST, CDEST)

base.dgerv2d(ICTX, SCOPE, m, n, x, lda, RDEST, CDEST)
**Arguments**

- **ICTXT**: BLACS ICTXT.
- **SCOPE**: Rows, cols, or both.
- **m, n**: Problem size.
- **x**: Local values.
- **lda**: Leading dimension.
- **RDEST**: Row destination.
- **CDEST**: Col destination.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

---

**Description**

Create Hilbert matrix.

**Usage**

base.dhilbmk(n)

**Arguments**

- **n**: Size.

**Details**

For advanced users only. See pbdDMAT for high-level functions.
**Description**

Compute dimensions on process MYROW=MYCOL=0

**Usage**

```
base.dim0(dim, ICTXT = 0)
```

**Arguments**

- **dim**
  Global dim.
- **ICTXT**
  BLACS context.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

**Value**

Dimension on MYROW=MYCOL=0

---

**Description**

Free Blacs System Handle

**Usage**

```
base.free_blacs_system_handle(SHANDLE)
```

**Arguments**

- **SHANDLE**
  A system handle. Obtained via a call to 'sys2blacs.handle'

**Value**

None
**base.igamx2d**  

**BLACS Max**

**Description**

Max value across a process grid.

**Usage**

```c
base.igamx2d(ICTXT, SCOPE, m, n, x, lda, RDEST, CDEST)
```

**Arguments**

- **ICTXT**: BLACS ICTXT.
- **SCOPE**: Rows, cols, or both.
- **m, n**: Problem size.
- **x**: Local values.
- **lda**: Leading dimension.
- **RDEST**: Row destination.
- **CDEST**: Col destination.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

---

**base.igsum2d**

**BLACS Sums**

**Description**

Sum across a process grid.

**Usage**

```c
base.igsum2d(ICTXT, SCOPE, m, n, x, lda, RDEST, CDEST)
```

```c
base.dgsum2d(ICTXT, SCOPE, m, n, x, lda, RDEST, CDEST)
```
base.indxg2p

Arguments

ICTXT    BLACS ICTXT.
SCOPE    Rows, cols, or both.
m, n    Problem size.
x    Local values.
lda    Leading dimension.
RDEST    Row destination.
CDEST    Col destination.

Details

For advanced users only. See pbdDMAT for high-level functions.

Description

Computes the process coordinate which contains the entry of a distributed matrix specified by a
global index INDXGLOB. Simplified reimplementation of the ScaLAPACK aux INDXG2P func-
tion.

Usage

base.indxg2p(INDXGLOB, NB, NPROCS)

Arguments

INDXGLOB    Global index.
NB    Block size.
NPROCS    Total number of processors over which matrix is distributed.

Details

For advanced users only. See pbdDMAT for high-level functions.

Value

The process coordinate.
base.matexp

Description
Serial matrix exponentiation.

Usage
base.matexp(A, p = 6, t = 1)

Arguments
A Matrix to exponentiate.
p Pade’ expansion size.
t Scaling factor.

Details
For advanced users only. See pbdDMAT for high-level functions.

Value
exp(A)

base.maxdim

Description
Compute maximum dimension across all nodes

Usage
base.maxdim(dim)

Arguments
dim Global dim.

Details
For advanced users only. See pbdDMAT for high-level functions.

Value
Maximum dimension.
**base.minctxt**  
*Get BLACS Context Grid Information*

**Description**  
Finds the smallest integers for creating a new BLACS context.

**Usage**  
```
base.minctxt(after = 0)
```

**Arguments**  
- `after`: ignores all values below this integer as possibilities

**Details**  
For advanced users only. See pbdDMAT for high-level functions.  
Returns the smallest integer which could become a new BLACS context value.  
For example, if contexts 0, 1, and 2 are taken, and `after=0`, then the function returns 3. If 0, 1, 2, and 5 are taken, the function returns 3 if `after=0`, but returns 6 if `after=4`.  
The function is useful when a transitory grid is needed, such as for reading in data onto a subset of processors before distributing out to the full grid.

**Value**  
Returns the minimum value.

---

**base.mksubmat**  
*(Un)Distribute*

**Description**  
*(Un)Distribute matrix.*

**Usage**  
```
base.mksubmat(x, descx)
base.mkgblmat(x, descx, rsrcc, csrcc)
```

**Arguments**  
- `x`: Matrix.  
- `descx`: ScaLAPACK descriptor array.  
- `rsrcc, csrcc`: Row/column source.
Details

For advanced users only. See pbdDMAT for high-level functions.

Examples

```r
spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

### Set data matrix and desc.
x <- matrix(as.double(1:30), nrow = 6, ncol = 5)
dim <- dim(x)
blim <- c(3L, 3L)
ldim <- base.numroc(dim = dim, bldim = blim)
descx <- base.descinit(dim = dim, bldim = blim, ldim = ldim)

### Redistribute from rank 0.
dx <- base.mksubmat(x, descx)
comm.print(dx, all.rank = TRUE)

finalize()
"
pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)
```

Description

Given integers n and d, with n > d, this function finds the "next best divisor" of n which is greater than or equal to d.

Usage

```r
base.nbd(n, d)
```

Arguments

- **n**  
The divident (number divided into).
- **d**  
The candidate divisor.

Details

Surprisingly useful for thinking about processor grid shapes.
Value
The "next best divisor" interger

Examples

```r
spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

text <- base.nbd(100, 10) # 10 divides 100, so 10 is returned
text <- base.nbd(100, 11) # 11 does not, so the 'next best' divisor, 20, is returned

finalize()
"
pbdMPI::execmpi(spmd.code = spmd.code, nranks = 1L)
```

Description
NUMber of Rows Or Columns

Usage

```r
base.numroc(dim, bldim, ICTXT = 0, fixme = TRUE)
```

Arguments

dim Global dim.
bldim Blocking dim.
ICTXT BLACS context.
fixme Should ldims be "rounded" to 0 or not.

Details
For advanced users only. See pbdDMAT for high-level functions.

Value
A vector of local dim.
Examples

spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

### Set desc.
dim <- c(6L, 5L)
bldim <- c(3L, 3L)
ldim <- base.numroc(dim = dim, bldim = bldim)
comm.print(ldim)

finalize()
"

pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)

---

base.ownany  

Determining Local Ownership of a Distributed Matrix

Description

For advanced users only. See pbdDMAT for high-level functions.

Usage

base.ownany(dim, bldim, ICTXT = 0)

Arguments

dim  global dimension
bldim  blocking dimension
ICTXT  BLACS context

Details

A simple wrapper of numroc. The return is the answer to the question 'do I own any of the global matrix?'. Passing a distributed matrix is allowed, but often it is convenient to determine that information without even having a distributed matrix on hand. In this case, explicitly passing the appropriate information to the arguments dim=, bldim= (and ICTXT= as necessary, since it defaults to 0) while leaving x missing will produce the desired result. See the examples below for more clarity.

The return for each function is local.

Value

TRUE or FALSE
Examples

spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

iown <- base.ownany(dim=c(4, 4), bldim=c(4, 4), ICTXT=0)
comm.print(iown, all.rank = TRUE)

finalize()
"
pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)

base.pdchtri  pdchtri

Description

Inverse of cholesky.

Usage

base.pdchtri(uplo, x, descx, descc)

Arguments

uplo  Triangle whose values to use.
x     Matrix to crossprod.
descx ScaLAPACK descriptor array.
descc ScaLAPACK descriptor array of output.

Details

For advanced users only. See pbdDMAT for high-level functions.
base.pdclvar

Description

Computes the variances of a ScaLAPCK-like distributed matrix. Significantly faster than using apply(), even in compared to the performance differences you would find comparing these two approaches using just base R.

Usage

base.pdclvar(x, descx)

Arguments

<table>
<thead>
<tr>
<th>x</th>
<th>The matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>descx</td>
<td>ScaLAPACK descriptor array.</td>
</tr>
</tbody>
</table>

base.pdhilbmk

Description

Create Hilbert matrix.

Usage

base.pdhilbmk(descx)

Arguments

| descx   | ScaLAPACK descriptor matrix. |

Details

For advanced users only. See pbdDMAT for high-level functions.
Description

Create Companion Matrix

Usage

base.pdmkcpn1(coef, descx)

Arguments

coef Coefficients vector.
descx ScaLAPACK descriptor array.

Details

For advanced users only. See pbdDMAT for high-level functions.

Description

R-like Matrix-Vector Sum

Usage

base.pdmvsum(x, descx, y, descy)

Arguments

x Matrix.
descx, descy ScaLAPACK descriptor array.
y Vector.
**base.pdsweep**

Description

Matrix-Vector Sweep

Usage

```r
base.pdsweep(x, descx, vec, MARGIN, FUN)
```

Arguments

- `x`: Matrix.
- `descx`: ScaLAPACK descriptor array.
- `vec`: Vector
- `MARGIN`: Rows or columns.
- `FUN`: Function.

Details

For advanced users only. See pbdDMAT for high-level functions.

**base.procgrid**

Description

"Optimal" process grid when nprow and ncol are empty

Usage

```r
base.procgrid(nprocs)
```

Arguments

- `nprocs`: Number of processors.

Details

For advanced users only. See pbdDMAT for high-level functions.

Value

A list contains nprow and ncol.
Examples

```r
spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

opt <- base.procgrid(4)
comm.print(opt)

opt <- base.procgrid(6)
comm.print(opt)

opt <- base.procgrid(8)
comm.print(opt)

finalize()
"
pbdMPI::execmpi(spmd.code = spmd.code, nranks = 1L)
```

Description

Pade' expansion.

Usage

```r
base.p_matexp_pade_wrap(A, desca, p = 6)
```

Arguments

- **A**: Matrix.
- **desca**: ScaLAPACK descriptor array.
- **p**: Order of the Pade’ approximation.

Details

For advanced users only. See pbdDMAT for high-level functions.

Value

Results of Pade’ expansion.
**Description**

Matrix power by squaring.

**Usage**

base.p_matpow_by_squaring_wrap(A, desca, b = 1)

**Arguments**

- **A**  
  Matrix.
- **desca**  
  ScaLAPACK descriptor array.
- **b**  
  Power.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

**Value**

A powered matrix.

---

**Description**

For advanced users only. See pbdDMAT for high-level functions.

**Usage**

base.rcolcpy(x, descx, y, descy, xcol, ycol)

**Arguments**

- **x, y**  
  Matrix.
- **descx, descy**  
  ScaLAPACK descriptor array.
- **xcol, ycol**  
  Columns.
**base.rcolcpy2**

**R Column Copy-2**

---

**Description**

For advanced users only. See pbdDMAT for high-level functions.

**Usage**

```r
base.rcolcpy2(x, descx, y, descy, xcol, ycol)
```

**Arguments**

- `x, y`: Matrix.
- `descx, descy`: ScaLAPACK descriptor array.
- `xcol, ycol`: Columns.

---

**base.redist**

**base.redist**

---

**Description**

Redistribute a matrix from rank 0 to all ranks in block cyclic fashion.

**Usage**

```r
base.redist(desc, A)
```

**Arguments**

- `desc`: ScaLAPACK descriptor array.
- `A`: Matrix.

**Value**

A block cyclic matrix of the input matrix A from rank 0.
Examples

```r
spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

### Set data matrix A and desc.
A <- matrix(as.double(1:30), nrow = 6, ncol = 5)
if (comm.rank() != 0)
  A <- matrix(as.double(0), nrow = 6, ncol = 5)
dim <- dim(A)
bldim <- c(3L, 3L)
ldim <- base.numroc(dim = dim, bldim = bldim)
desc <- base.descinit(dim = dim, bldim = bldim, ldim = ldim)

### Redistribute from rank 0.
dA <- base.redist(desc, A)
comm.print(dA, all.rank = TRUE)

finalize()
"

pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)
```

---

**base.rl2blas**

**Level 2 R-like BLAS**

**Description**

For advanced users only. See pbdDMAT for high-level functions.

**Usage**

```r
base.rl2blas(x, descx, vec, FUN)
```

**Arguments**

- `x`: Matrix.
- `descx`: ScaLAPACK descriptor array.
- `vec`: Global vector.
- `FUN`: Function.
base.r12insert  

**R-like Matrix-Vector Insertion**

---

**Description**

For advanced users only. See pbdDMAT for high-level functions.

**Usage**

```
base.r12insert(x, descx, vec, i, j)
```

**Arguments**

- `x`: Matrix.
- `descx`: ScaLAPACK descriptor array.
- `vec`: Global vector.
- `i, j`: Indices.

---

base.rpdgecon  

**rpdgecon**

---

**Description**

Inverse condition number of a general matrix.

**Usage**

```
base.rpdgecon(norm, m, n, a, desca)
```

**Arguments**

- `norm`: Type of norm.
- `m, n`: Problem size
- `a`: Matrix.
- `desca`: ScaLAPACK descriptor array.

**Details**

For advanced users only. See pbdDMAT for high-level functions.
**Description**
LQ.

**Usage**
base.rpdgelqf(m, n, x, descx)

**Arguments**
m, n  Problem size.
x  Matrix.
descx  ScaLAPACK descriptor array.

**Details**
For advanced users only. See pbdDMAT for high-level functions.

---

**Description**
Linear model fitter via rank-revealing QR (with pivoting).

**Usage**
base.rpdgels(tol, m, n, nrhs, a, desca, b, descb)

**Arguments**
tol  Numerical tolerance for the QR.
m, n  Problem size.
nrhs  Number of right hand sides.
a  Left hand side.
desca  ScaLAPACK descriptor array.
b  Right hand side.
descb  ScaLAPACK descriptor array.

**Details**
For advanced users only. See pbdDMAT for high-level functions.
**base.rpdgemm**

<table>
<thead>
<tr>
<th>Description</th>
<th>Matrix-Matrix Multiply.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage</strong></td>
<td>base.rpdgemm(transx, transy, x, descx, y, descy, descc)</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>transx, transy</td>
</tr>
<tr>
<td></td>
<td>x, y</td>
</tr>
<tr>
<td></td>
<td>descx, descy, descc</td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>For advanced users only. See pbdDMAT for high-level functions.</td>
</tr>
</tbody>
</table>

**base.rpdgemr2d**

<table>
<thead>
<tr>
<th>Description</th>
<th>General 2d block cyclic redistribution function.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage</strong></td>
<td>base.rpdgemr2d(x, descx, descy)</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>descx, descy</td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>For advanced users only. See pbdDMAT for high-level functions.</td>
</tr>
</tbody>
</table>
base.rpdgeqpf

Description
QR.

Usage
base.rpdgeqpf(tol, m, n, x, descx, comm = .pbd_env$SPMD.CT$comm)

Arguments
tol Numerical tolerance for the QR.
m, n Problem size.
x Matrix.
descx ScaLAPACK descriptor array.
comm An MPI (not BLACS) communicator.

Details
For advanced users only. See pbdDMAT for high-level functions.

Value
A list contains QR results.

base.rpdgesv

Description
Solving a (square) system of equations.

Usage
base.rpdgesv(n, nrhs, a, desca, b, descb)

Arguments
n Problem size.
nrhs Number of right hand sides.
a, b Matrix.
desca, descb ScaLAPACK descriptor array.
Description

SVD.

Usage

```r
base.rpdgesvd(
  jobu,
  jobvt,
  m,
  n,
  a,
  desca,
  descu,
  descvt,
  ...,
  inplace = FALSE,
  comm = .pbd_env$SPMD.CT$comm
)
```

Arguments

- `jobu, jobvt` Control for u/vt return.
- `m, n` Problem size.
- `a` Matrix.
- `desca, descu, descvt` ScaLAPACK descriptor array.
- `...` Ignored
- `inplace` Should the computation be done in-place or not. For REALLY advanced users only.
- `comm` An MPI (not BLACS) communicator.

Details

For advanced users only. See pbdDMAT for high-level functions.
Description
LU factorization.

Usage
base.rpdgetrf(a, desca)

Arguments
a Matrix.
desca ScaLAPACK descriptor array.

Details
For advanced users only. See pbdDMAT for high-level functions.

Description
Matrix inversion.

Usage
base.rpdgetri(n, a, desca)

Arguments
n Problem size.
a Matrix.
desca ScaLAPACK descriptor array.

Details
For advanced users only. See pbdDMAT for high-level functions.
**Description**
Matrix norms.

**Usage**
```r
textbase.rpdlange(norm, m, n, a, desca)
```

**Arguments**
- `norm`: Type of norm.
- `m, n`: Problem size
- `a`: Matrix.
- `desca`: ScaLAPACK descriptor array.

**Details**
For advanced users only. See pbdDMAT for high-level functions.

---

**Description**
Matrix printer.

**Usage**
```r
textbase.rpdlaprint(m, n, a, desca)
```

**Arguments**
- `m, n`: Number rows/cols.
- `a`: Matrix.
- `desca`: ScaLAPACK descriptor array.

**Details**
For advanced users only. See pbdDMAT for high-level functions.
**Description**
Recover Q.

**Usage**
```c
base.rpdorglq(m, n, k, lq, desc, tau)
```

**Arguments**
- **m, n**  
  Problem size.
- **k**  
  Number of elementary reflectors.
- **lq**  
  QR decomposition.
- **desc**  
  ScaLAPACK descriptor array.
- **tau**  
  Elementary reflectors.

**Details**
For advanced users only. See pbdDMAT for high-level functions.

**Value**
Q matrix of the QR decomposition.

**Description**
Recover Q.

**Usage**
```c
base.rpdorgqr(m, n, k, qr, descqr, tau)
```

**Arguments**
- **m, n**  
  Problem size.
- **k**  
  Number of elementary reflectors.
- **qr**  
  QR decomposition.
- **descqr**  
  ScaLAPACK descriptor array.
- **tau**  
  Elementary reflectors.
base.rpdormqr

Details
For advanced users only. See pbdDMAT for high-level functions.

Value
Q matrix of the QR decomposition.

Description
op(Q) * y.

Usage
base.rpdormqr(side, trans, m, n, k, qr, descqr, tau, c, descc)

Arguments
side 'L' or 'R', for left or right application of Q matrix.
trans Q or Q^T.
m, n Problem size.
k Number of elementary reflectors.
qr QR decomposition.
descqr ScaLAPACK descriptor array.
tau Elementary reflectors.
c Vector.
descc ScaLAPACK descriptor array.

Details
For advanced users only. See pbdDMAT for high-level functions.
**base.rpdpotrf**

**Rpdpotrf**

**Description**

Cholesky factorization.

**Usage**

```r
classbase.rpdpotrf(uplo, n, a, desca)
```

**Arguments**

- `uplo` Triangle where the information is stored (in the symmetric matrix).
- `n` Problem size.
- `a` Matrix.
- `desca` ScaLAPACK descriptor array.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

---

**base.rpdseyevr**

**Rpdseyevr**

**Description**

Symmetric eigenvalue decomposition.

**Usage**

```r
classbase.rpdseyevr(jobz, uplo, n, a, desca, descz)
```

**Arguments**

- `jobz` Control for if vectors/values/both are returned.
- `uplo` Triangle where the information is stored (in the symmetric matrix).
- `n` Problem size.
- `a` Matrix.
- `desca`, `descz` ScaLAPACK descriptor array.

**Details**

For advanced users only. See pbdDMAT for high-level functions.
Description

Generalized eigenvalue problem.

Usage

base.rpdsyevx(
    jobz,  
    range,  
    n,  
    a,  
    desca,  
    vl,  
    vu,  
    il,  
    iu,  
    abstol = 1e-08,  
    orfac = 0.001  
)

Arguments

jobz Control for if vectors/values/both are returned.
range Parameter to determine the search criteria for eigenvalues.
n Problem size.
a Matrix.
desca ScaLAPACK descriptor array.
vl, vu Endpoints of the interval subset of the real line in which to search for eigenvalues, if specified by range.
il, iu Eigenvalues with indices il, ..., iu will be found, if specified by range.
abstol Absolute error tolerance for the eigenvalues.
orfac Eigenvectors with eigenvalues below orfac*norm(a) of each other are reorthogonalized.

Details

For advanced users only. See pbdDMAT for high-level functions.
**base.rpdtran**

**Description**

Transpose.

**Usage**

base.rpdtran(a, desca, descc)

**Arguments**

- **a**: Matrix.
- **desca, descc**: ScaLAPACK descriptor array.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

---

**base.rpdtrcon**

**Description**

Inverse condition number of a triangular matrix.

**Usage**

base.rpdtrcon(norm, uplo, diag, n, a, desca)

**Arguments**

- **norm**: Type of norm.
- **uplo**: Triangle where information is stored.
- **diag**: Specifies if the matrix is unit triangular or not.
- **n**: Problem size
- **a**: Matrix.
- **desca**: ScaLAPACK descriptor array.

**Details**

For advanced users only. See pbdDMAT for high-level functions.
**base.rrowcpy**  
*R Row Copy*

**Description**

For advanced users only. See pbdDMAT for high-level functions.

**Usage**

```r
base.rrowcpy(x, descx, y, descy, xrow, yrow)
```

**Arguments**

- `x, y` Matrix.
- `descx, descy` ScaLAPACK descriptor array.
- `xrow, yrow` Rows.

---

**base.rrowcpy2**  
*R Row Copy-2*

**Description**

For advanced users only. See pbdDMAT for high-level functions.

**Usage**

```r
base.rrowcpy2(x, descx, y, descy, xrow, yrow)
```

**Arguments**

- `x, y` Matrix.
- `descx, descy` ScaLAPACK descriptor array.
- `xrow, yrow` Rows.
**base.tri2zero**  
*tri2zero*

**Description**
Zero Triangle

**Usage**
```r
base.tri2zero(x, descx, uplo = "L", diag = "N")
```

**Arguments**
- `x` Matrix.
- `descx` ScaLAPACK descriptor array.
- `uplo` Triangle.
- `diag` Zero diagonal as well.

**Details**
For advanced users only. See pbdDMAT for high-level functions.

---

**base.valid_context**  
*BLACS Context Validation*

**Description**
Checks if a supplied ICTXT is valid.

**Usage**
```r
base.valid_context(ICTXT, ..., override = FALSE)
```

**Arguments**
- `ICTXT` BLACS context number.
- `...` Not used.
- `override` If `override=FALSE`, the context number will produce an error if it is any of the reserved contexts (0, 1, or 2).
Description
Shuts down all BLACS communicators.

Usage
base.blacsexit(CONT = TRUE)
blacsexit(CONT = TRUE)

Arguments
CONT logical; determines whether or not to shut down all MPI communicators

Details
If the user wishes to shut down BLACS communicators but still have access to MPI, then call this function with CONT=TRUE. Calling blacsexit(CONT=FALSE) will shut down all MPI communicators, equivalent to calling

> blacsexit(CONT=TRUE) > finalize(mpi.finalize=TRUE)

This function is automatically invoked if BLACS communicators are running and finalize() is called.

Value
Has an invisible return of 0 when successful.

Examples

```r
spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

### Do something with BLACS here.

### Don't use this unless you know what to do after this.
# blacsexit()

### Then, do others without BLACS here.

finalize() # This should be off since blacexit().
"
pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)
```

blacs_apts  Functions to set and get BLACS_APTS

Description
To set and get BLACS array/object/whatever pointers needed in and from R. Because other packages
has it's own memory stack vision that may not be visiable by this package or vice versa.

Usage
set.blacs.apts()
get.blacs.apts()

Details
The 'set.blacs.apts()' is for advanced users. This one is needed to be called within R from 'pbd-
BASE' package to set the pointers to the memory where BLACS had initialized so that the pointers
are set to the right address of the memory stack.
The 'get.blacs.apts()' is for debugging only. The advanced user mainly calls the C version 'get_BLACS_APTS_from_R()'
in 'src/export_blacs/pkg_ools.c'.
I am lazy to use .C(), but should not hurt performance here. Eventually,.pbdBASEEnv should pass
to .C() and set/get pointers from it instead of .GlobalEnv.

coords  Local to Global/Global to Local Indexing

Description
Get the local index given global information.

Usage
 indxg2l(INDXGLOB, NB, IPROC, ISRCPROC, NPROCS)
 indxl2g(INDXLOC, NB, IPROC, ISRCPROC, NPROCS)

Arguments
INDXGLOB  Global index.
NB         Block size.
IPROC      Coordinate of the process whose local info is to be determined.
ISRCPROC   The coordinate of the process that possesses the first row/column of the dis-
            tributed matrix. That’s always 0 pbdDMAT.
NPROCS     Total number of processors over which matrix is distributed.
INDXLOC    Local index.
coordspair

Details

For advanced users only. See pbdDMAT for high-level functions.

Value

The local index.

---

<table>
<thead>
<tr>
<th>coordspair</th>
<th>Global to Local/Local to Global Pair Indexing</th>
</tr>
</thead>
</table>

Description

Get the local index-pair given global information.

Usage

\[
g2lpair(gi, gj, bldim, ICTXT) \\
l2gpair(i, j, bldim, ICTXT)
\]

Arguments

\[
\begin{align*}
gi, gj & \quad \text{Global indices.} \\
bldim & \quad \text{Blocking dimension} \\
ICTXT & \quad \text{BLACS context.} \\
i, j & \quad \text{Local indices.}
\end{align*}
\]

Details

For advanced users only. See pbdDMAT for high-level functions.

Value

The local index-pair.
Description

Grab diagonal or create distributed diagonal matrix.

Usage

base.ddiagtk(x, descx, proc.dest = "all")
base.ddiagmk(diag, descx)

Arguments

x Matrix.
descx ScaLAPACK descriptor array.
proc.dest Who owns the result.
diag Diagonal.

Details

For advanced users only. See pbdDMAT for high-level functions.

Value

diagonal elements of matrix or a diagonal matrix

Examples

spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

### Set data matrix and desc.
x <- matrix(as.double(1:25), nrow = 5, ncol = 5)
dim <- dim(x)
blbdim <- c(3L, 3L)
ldim <- base.numroc(dim = dim, bldim = bldim)
descx <- base.descinit(dim = dim, bldim = bldim, ldim = ldim)

### Get diagonal
diag.x <- base.ddiagtk(x, descx)
comm.print(diag.x)

finalize()
"
"
```
finalizer

pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)

spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

### Set data matrix and desc.
x <- matrix(as.double(1:25), nrow = 5, ncol = 5)
dim <- dim(x)
bdim <- c(3L, 3L)
ldim <- base.numroc(dim = dim, bldim = bdim)
descx <- base.descinit(dim = dim, bldim = bdim, ldim = ldim)

### Set diagonal
diag.x <- base.ddiagtk(x, descx)
new.x <- base.ddiagmk(diag.x, descx)
comm.print(new.x, all.rank = TRUE)

finalize()
"
pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)
```

---

**finalizer**

---

**Finalizer**

---

**Description**

A replacement for `pbdMPI::finalize()` that automatically shuts BLACS communicators down.

**Usage**

```r
base.finalize(mpi.finalize = .pbd_env$SPMD.CT$mpi.finalize)
finalize(mpi.finalize = .pbd_env$SPMD.CT$mpi.finalize)
```

**Arguments**

- `mpi.finalize`: If MPI should be shut down.

**Value**

None
**g2lcoord**

---

**Description**

Global to local coordinates with explicit ownership given.

**Usage**

```
g2lcoord(dim, bldim, gi, gj, gridinfo)
```

**Arguments**

- `dim`: Global dimension.
- `bldim`: Blocking dimension.
- `gi, gj`: Global row and column indices, respectively.
- `gridinfo`: The return of `base.blacs(ICTX(x))`. See the Details section for more information.

**Value**

For the process that owns the desired local data at global indices \((gi, gj)\), the return is the local index. Otherwise, `NA` is returned.

---

**g2l_coord**

---

**Description**

Global to local coords.

**Usage**

```
base.g2l_coord(ind, bldim, ICTXT = 0, dim = NULL)
g2l_coord(ind, bldim, ICTXT = 0, dim = NULL)
```

**Arguments**

- `ind`: Matrix indices.
- `bldim`: Blocking dimension.
- `ICTXT`: BLACS context.
- `dim`: Ignored; will be removed in a future version.
Details
For advanced users only. See pbdDMAT for high-level functions.

Value
Local coords.

get.comm.from.ICTXT  Getting Communicator From BLACS Context

Description
Blacs context are associated with a certain communicator. It can be useful to retrieve this communicator to manipulate the matrix accordingly.

Usage
get.comm.from.ICTXT(ICTXT)

Arguments
ICTXT  a BLACS context

Value
A communicator

gridexit  gridexit

Description
Frees a BLACS context.

Usage
base.gridexit(ICTXT, override = FALSE)
gridexit(ICTXT, override = FALSE)

Arguments
ICTXT  BLACS context number.
override  logical; if TRUE, ignores normal check preventing the closing of ICTXT values of 0, 1, and 2. This could cause things to go crazy and I do not recommend it.
Details

For advanced users only. See pbDMAT for high-level functions.

The function frees the requested BLACS context. It is a trivial wrapper for the BLACS routine BLACS_GRIDEXIT. Also removes the object .__blacs_gridinfo_ICTXT.

Contexts 0, 1, and 2 can not be freed in this way unless the argument override=FALSE. This will probably break something and I do not recommend it.

Value

Silently returns 0 when successful. Silently returns 1 when requested ICTXT does not exist.
**Examples**

```r
spmd.code <- "
  suppressMessages(library(pbdMPI))
  suppressMessages(library(pbdBASE))
  init.grid()

  mygrid <- blacs(0)
  comm.print(mygrid, all.rank = TRUE)

  finalize()
"

pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)
```

---

**Description**

BLACS grid initialization.

**Usage**

- `base.blacs_init(ICTXT, NPROW, NPCOL, ..., quiet = FALSE)`
- `blacs_init(ICTXT, NPROW, NPCOL, ..., quiet = FALSE)`
- `blacs_gridinit(ICTXT, NPROW, NPCOL, ..., quiet = FALSE)`

**Arguments**

- **ICTXT**: BLACS context.
- **NPROW, NPCOL**: Number of process rows/cols.
- **...**: Additional arguments.
- **quiet**: Verbose initialization or not.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

**Value**

None
Description

Manages the creation of BLACS context grids.

Usage

\texttt{init.grid(NPROW, NPCOL, ICTXT, quiet = FALSE)}

Arguments

\begin{itemize}
\item \texttt{NPROW} number of process rows. Can be missing; see details.
\item \texttt{NPCOL} number of process columns. Can be missing; see details.
\item \texttt{ICTXT} BLACS context number.
\item \texttt{quiet} logical; controls whether or not information about grid size should be printed.
\end{itemize}

Details

\texttt{blacs_init()} is for experienced users only. It is a shallow wrapper of the BLACS routine \texttt{BLACS_INIT}, with the addition of creating the \texttt{.__blacs_gridinfo_ICTXT} objects, as described below.

The remainder of this section applies only to \texttt{init.grid()}. If ICTXT is missing, three variables will be created in the .pbdBASEEnv environment:

\begin{itemize}
\item \texttt{.__blacs_gridinfo_0}
\item \texttt{.__blacs_gridinfo_1}
\item \texttt{.__blacs_gridinfo_2}
\end{itemize}

These variables store the BLACS process grid information for the BLACS context corresponding to the trailing digit of the variable. Most users should invoke \texttt{init.grid()} in this fashion, namely with ICTXT missing, and only do so once.

Contexts 0, 1, and 2 are reserved. Additional custom contexts are possible to create, but they must be integers \( \geq 3 \).

Context 0 is the “full” process grid of \texttt{NPROW} by \texttt{NPCOL} processes; contexts 1 is the process grid consisting of 1 process row and \texttt{NPROW*NPCOL} processes columns; context 2 is the process grid consisting of \texttt{NPROW*NPCOL} processes rows and 1 process column. These contexts can be redundant depending on the number of processes available.

BLACS contexts have important internal use, and advanced users familiar with ScaLAPACK might find some advantage in directly manipulating these process grids. Most users should not need to directly manage BLACS contexts, in this function or elsewhere.

If the \texttt{NPROW} and \texttt{NPCOL} values are missing, then a best process grid will be chosen for the user based on the total available number of processes. Here “best” means as close to a square grid as possible.

The variables \texttt{.__blacs_gridinfo_ICTXT} are just storage mechanisms to avoid needing to directly invoke the BLACS routine \texttt{BLACS_GRIDINFO}. 
Additionally, another variable is created in the .pbdBASEEnv environment, namely .__blacs_initialized. Its existence is to alert finalize() to shut down BLACS communicators, if necessary, to prevent memory leaks.

**Value**

Silently returns 0 when successful. Additionally, several variables are created in the .pbdBASEEnv environment. See Details section.

**Examples**

```r
spmd.code <- "
  suppressMessages(library(pbdMPI))
  suppressMessages(library(pbdBASE))
  init.grid()

  ### Do something here. For example, below.
  comm.print(ls(.pbdBASEEnv))

  finalize()
"

pbdMPI::execmpi(spmd.code = spmd.code, nranks = 2L)
```

---

**l2g_coord**

Local to global coords.

**Usage**

```r
base.l2g_coord(ind, bldim, ICTXT = 0, dim = NULL)

l2g_coord(ind, bldim, ICTXT = 0, dim = NULL)
```

**Arguments**

- `ind`  Matrix indices.
- `bldim`  Blocking dimension.
- `ICTXT`  BLACS context.
- `dim`  Ignored; will be removed in a future version.

**Details**

For advanced users only. See pbdDMAT for high-level functions.
**Value**

Global coords.

---

**Description**

A better version of NUMROC (NUMber Rows Or Columns). Returns the local dimension given global matrix + distribution parameters.

**Usage**

```fortran
numroc2(N, NB, IPROC, NPROCS)
```

**Arguments**

- **N**: Global number of rows/cols.
- **NB**: Block size.
- **IPROC**: Coordinate of the process whose local info is to be determined.
- **NPROCS**: Total number of processors over which matrix is distributed.

**Details**

For advanced users only. See pbdDMAT for high-level functions.

**Value**

The local dimension.

---

**pcoords**

*Interchange Between Process Number and BLACS Coordinates*

**Description**

Grabs the existing BLACS context grid information.

**Usage**

```fortran
base.pnumICTXT, PROW, PCOL)
```

```fortran
base.pcoordICTXT, PNUM)
```
Arguments
ICTXT  BLACS context number.
PROW, PCOL  BLACS grid location row/column
PNUM  process rank

Details
For advanced users only. These functions are simple recreations of the BLACS routines BLACS_PNUM and BLACS_PCOORD. The former gets the process number associated with the BLACS process grid location c(MYPROW, MYPOL), while the latter does the reverse.

Value
pnum returns an integer; pcoord returns a list containing elements PROW and PCOL.

Examples
spmd.code <- "
suppressMessages(library(pbdMPI))
suppressMessages(library(pbdBASE))
init.grid()

### get the ICTXT = 0 BLACS coordinates for process 3
myCoords <- base.pcoord(ICTXT = 0, PNUM = 3)
comm.print(myCoords)

### get the ICTXT = 1 BLACS coordinates for process 3
myCoords <- base.pcoord(ICTXT = 1, PNUM = 3)
comm.print(myCoords)

### get the ICTXT = 2 BLACS coordinates for process 3
myCoords <- base.pcoord(ICTXT = 2, PNUM = 3)
comm.print(myCoords)

finalize()
"
pbdMPI::execmpi(spmd.code = spmd.code, nranks = 4L)
Arguments

comm Communicator for which you want to set the BLACS context

Value

A system handle, i.e. the system context number. System contexts can be used to have ScalaPACK methods run in different communicators.

See Also

base.free_blacs_system_handle, base.blacs_gridinit
Index

*Topic **BLACS**
  base.minctxt, 15
  base.ownany, 18
  blacsexit, 41
  gridexit, 47
  gridinfo, 48
  InitGrid, 50
  pcoords, 52
*Topic **Data**
  base.ownany, 18
*Topic **Distributing**
  base.ownany, 18
*Topic **Package**
  pbdBASE-package, 3
*Topic **datasets**
  BASE Global Environment, 5
  .pbdBASEEnv (BASE Global Environment), 5
  BASE Global Environment, 5
  base.blacs (gridinfo), 48
  base.blacs_gridinit, 6
  base.blacs_init (gridinit), 49
  base.blacsexit (blacsexit), 41
  base.crossprod, 6
  base.ddiagmk (diag), 44
  base.ddiagtk (diag), 44
  base.descinit, 7
  base.det, 8
  base.dgamm2d (base.dgmx2d), 9
  base.dgax2d, 9
  base.dger2d (base.dgesd2d), 9
  base.dgesd2d, 9
  base.dgsum2d (base.igsum2d), 12
  base.dhilbmk, 10
  base.dim0, 11
  base.finalize (finalizer), 45
  base.free_blacs_system_handle, 11
  base.g2l_coord (g2l_coord), 46
  base.gridexit (gridexit), 47
  base.igamm2d (base.dgmx2d), 9
  base.igamx2d, 12
  base.igsum2d, 12
  base.indxg2p, 13
  base.12g_coord (12g_coord), 51
  base.matexp, 14
  base.maxdim, 14
  base.minctxt, 15
  base.mkblmat (base.mksubmat), 15
  base.mksubmat, 15
  base.nbd, 16
  base.numroc, 17
  base.ownany, 18
  base.p_matexp_pade_wrap, 23
  base.p_matpow_by_squaring_wrap, 24
  base.pcoord (pcoords), 52
  base.pdcchtri, 19
  base.pdc1var, 20
  base.pdhilbmk, 20
  base.pdmkcpn1, 21
  base.pdmvsum, 21
  base.pdsweep, 22
  base.pnum (pcoords), 52
  base.procgrid, 22
  base.rcolcpy, 24
  base.rcolcpy2, 25
  base.redist, 25
  base.rl2blas, 26
  base.rl2insert, 27
  base.rpdgecon, 27
  base.rpdgelqf, 28
  base.rpdgels, 28
  base.rpdgemm, 29
  base.rpdgemr2d, 29
  base.rpdgesv, 30
  base.rpdgesvd, 31
  base.rpdgetrf, 32
  base.rpdgetri, 32
  base.rpdilange, 33
base.rpdlaprt, 33
base.rpdorglq, 34
base.rpdorgqr, 34
base.rpdormqr, 35
base.rpdpotrf, 36
base.rpdsev, 36
base.rpdsevpx, 37
base.rpdtran, 38
base.rpdtrcon, 38
base.rrowcpy, 39
base.rrowcpy2, 39
base.tri2zero, 40
base.valid_context, 40
blacs(gridinfo), 48
blacs_cepts, 42
blacs_gridinit(gridinit), 49
blacs_init(gridinit), 49
blacsexit, 41
coords, 42
coordspair, 43
diag, 44
finalize(finalizer), 45
finalizer, 45
g2l_coord, 46
g2lcoord, 46
g2lpair(coordspair), 43
get.blacs.cepts(blacs_cepts), 42
get.comm.from.ICTXT, 47
gridexit, 47
gridinfo, 48
gridinit, 49
indx2l(coords), 42
indx2g(coords), 42
init.grid(InitGrid), 50
InitGrid, 50
l2g_coord, 51
l2gpair(coordspair), 43
numroc2, 52
pbdBASE-package, 3
pcoords, 52
set.blacs.cepts(blacs_cepts), 42
sys2blacs.handle, 53