# Package ‘pbdBASE’

February 28, 2020

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Programming with Big Data -- Base Wrappers for Distributed Matrices</td>
</tr>
<tr>
<td>Version</td>
<td>0.5-3</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
<tr>
<td>License</td>
<td>Mozilla Public License 2.0</td>
</tr>
<tr>
<td>Depends</td>
<td>R (&gt;= 3.6.0), methods</td>
</tr>
<tr>
<td>Imports</td>
<td>utils, pbdMPI (&gt;= 0.4-3), pbdSLAP (&gt;= 0.2-9)</td>
</tr>
<tr>
<td>SystemRequirements</td>
<td>OpenMPI (&gt;= 1.5.4) on Solaris, Linux, Mac, and FreeBSD, MS-MPI (Microsoft HPC Pack 2012) or MPICH2 (&gt;= 1.4.1p1) on Windows.</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
<tr>
<td>LazyData</td>
<td>yes</td>
</tr>
<tr>
<td>ByteCompile</td>
<td>yes</td>
</tr>
<tr>
<td>NeedsCompilation</td>
<td>yes</td>
</tr>
<tr>
<td>URL</td>
<td><a href="https://pbdr.org/">https://pbdr.org/</a></td>
</tr>
<tr>
<td>BugReports</td>
<td><a href="http://group.pbdr.org/">http://group.pbdr.org/</a></td>
</tr>
<tr>
<td>MailingList</td>
<td>Please send questions and comments regarding pbdR to <a href="mailto:RBigData@gmail.com">RBigData@gmail.com</a></td>
</tr>
<tr>
<td>Maintainer</td>
<td>Wei-Chen Chen <a href="mailto:wccsnow@gmail.com">wccsnow@gmail.com</a></td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>7.0.2</td>
</tr>
<tr>
<td>Author</td>
<td>Drew Schmidt [aut], Wei-Chen Chen [aut, cre], Sebastien Lamy de la Chapelle [aut], George Ostrouchov [aut],</td>
</tr>
</tbody>
</table>
Pragneshkumar Patel [aut],
Ewan Higgs [ctb]

Repository CRAN
Date/Publication 2020-02-28 10:30:05 UTC

R topics documented:

- pbdBASE-package
- BASE Global Environment
- base.blacs_gridinit
- base.crossprod
- base.descinit
- base.det
- base.dgamx2d
- base.dgesd2d
- base.dhilbmk
- base.dim0
- base.free_blacs_system_handle
- base.igamx2d
- base.igsum2d
- base.indxg2p
- base.matexp
- base.maxdim
- base.minctxt
- base.mksubmat
- base.nbd
- base.numroc
- base.ownany
- base.pdchtri
- base.pdclvar
- base.pdilbmk
- base.pdmkcpn1
- base.pdmvsum
- base.pdsweep
- base.procgrid
- base.r_matexp_pade_wrap
- base.r_matpow_by_squaring_wrap
- base.rcolcpy
- base.rcolcpy2
- base.redist
- base.rl2blas
- base.rl2insert
- base.rpdgecon
- base.rpdgelqf
- base.rpdgels
- base.rpdgemm
- base.rpdgemr2d
### Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>base.pdgeqpf</td>
<td>30</td>
</tr>
<tr>
<td>base.pdgesv</td>
<td>30</td>
</tr>
<tr>
<td>base.pdgesvd</td>
<td>31</td>
</tr>
<tr>
<td>base.pdgetrf</td>
<td>32</td>
</tr>
<tr>
<td>base.pdgetri</td>
<td>32</td>
</tr>
<tr>
<td>base.pdlange</td>
<td>33</td>
</tr>
<tr>
<td>base.pdlaprt</td>
<td>33</td>
</tr>
<tr>
<td>base.pdorglq</td>
<td>34</td>
</tr>
<tr>
<td>base.pdorgqr</td>
<td>34</td>
</tr>
<tr>
<td>base.pdormqr</td>
<td>35</td>
</tr>
<tr>
<td>base.pdpotrf</td>
<td>36</td>
</tr>
<tr>
<td>base%pdsyevr</td>
<td>36</td>
</tr>
<tr>
<td>base.pdsyev</td>
<td>37</td>
</tr>
<tr>
<td>base.pdjtran</td>
<td>38</td>
</tr>
<tr>
<td>base.pdtrcon</td>
<td>38</td>
</tr>
<tr>
<td>base.rowcpy</td>
<td>39</td>
</tr>
<tr>
<td>base.rowcpy2</td>
<td>39</td>
</tr>
<tr>
<td>base.tri2zero</td>
<td>40</td>
</tr>
<tr>
<td>base.valid_context</td>
<td>40</td>
</tr>
<tr>
<td>blaceexit</td>
<td>41</td>
</tr>
<tr>
<td>blacs_aps</td>
<td>42</td>
</tr>
<tr>
<td>coords</td>
<td>42</td>
</tr>
<tr>
<td>coordspair</td>
<td>43</td>
</tr>
<tr>
<td>diag</td>
<td>44</td>
</tr>
<tr>
<td>finalizer</td>
<td>45</td>
</tr>
<tr>
<td>g2lcoord</td>
<td>46</td>
</tr>
<tr>
<td>g2l_coord</td>
<td>46</td>
</tr>
<tr>
<td>get.comm.from.ICTXT</td>
<td>47</td>
</tr>
<tr>
<td>gridexit</td>
<td>47</td>
</tr>
<tr>
<td>gridinfo</td>
<td>48</td>
</tr>
<tr>
<td>gridinit</td>
<td>49</td>
</tr>
<tr>
<td>InitGrid</td>
<td>50</td>
</tr>
<tr>
<td>l2g_coord</td>
<td>51</td>
</tr>
<tr>
<td>numroc2</td>
<td>51</td>
</tr>
<tr>
<td>pcoords</td>
<td>52</td>
</tr>
<tr>
<td>sys2blacs.handle</td>
<td>52</td>
</tr>
</tbody>
</table>

### Description

A package contains the basic methods for dealing with distributed data types, as well as the data types themselves.
Details
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

```
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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uplo</td>
<td>Triangle whose values to use.</td>
</tr>
<tr>
<td>trans</td>
<td>tcrossprod or crossprod.</td>
</tr>
<tr>
<td>x</td>
<td>Matrix to crossprod.</td>
</tr>
<tr>
<td>descx</td>
<td>ScaLAPACK descriptor array.</td>
</tr>
<tr>
<td>descc</td>
<td>ScaLAPACK descriptor array of output.</td>
</tr>
</tbody>
</table>

Details

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

Description

Creates ScaLAPACK descriptor array.

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim</td>
<td>Global dim.</td>
</tr>
<tr>
<td>bldim</td>
<td>Blocking dim.</td>
</tr>
<tr>
<td>ldim</td>
<td>Local dim.</td>
</tr>
<tr>
<td>ICTXT</td>
<td>BLACS context.</td>
</tr>
</tbody>
</table>

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()
```

```
spmd.code <- "
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**

**Description**

Min value across a process grid.

**Usage**

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

base.igamn2d(ICTXT, SCOPE, m, n, x, lda, RDEST, CDEST)

base.dgamn2d(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.dgesd2d**

**Description**

Sent value across a process grid.

**Usage**

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

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

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

Details

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

---

Description

Create Hilbert matrix.

Usage

base.dhilbmk(n)

Arguments

n \quad Size.

Details

For advanced users only. See pbdDMAT for high-level functions.
**base.dim0**

---

**maxdim**

---

**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

---

**base.free_blacs_system_handle**

*Free Blacs System Handle*

---

**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**

```
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**

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

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

```r
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**

```r
base.mksubmat(x, descx)
```

```r
base.mkgblmat(x, descx, rsrc, csrc)
```

**Arguments**

- `x` Matrix.
- `descx` ScaLAPACK descriptor array.
- `rsrc, csr` Row/column source.
base.nbd

Details

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

Examples

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, blim = blim)
descx <- base.descinit(dim = dim, blim = blim, ldim = ldim)

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

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

---

base.nbd  

Next Best Divisor

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

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" integer

Examples

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

  base.nbd(100, 10) # 10 divides 100, so 10 is returned
  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

```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)
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*

```r
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**  
*pchtri*

**Description**

Inverse of cholesky.

**Usage**

```r
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  

*Column Variances*

**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**

```r
base.pdclvar(x, descx)
```

**Arguments**

- `x` The matrix.
- `descx` ScaLAPACK descriptor array.

---

base.pdhilbmk  

*pdhilbmk*

**Description**

Create Hilbert matrix.

**Usage**

```r
base.pdhilbmk(descx)
```

**Arguments**

- `descx` ScaLAPACK descriptor matrix.

**Details**

For advanced users only. See pbdDMAT for high-level functions.
base.pdmkcpn1

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.

base.pdmvsum

R-like Matrix-Vector Sum

Description

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

Usage

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

Arguments

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

*pdsweep*

---

**Description**

Matrix-Vector Sweep

**Usage**

```
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**

*procgrid*

---

**Description**

"Optimal" process grid when nprow and ncol are empty

**Usage**

```
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.
**Description**

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

**Usage**

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

**Arguments**

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

---

**Description**

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

**Usage**

```r
glbase.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.rl2insert \hspace{1cm} R-like Matrix-Vector Insertion

**Description**

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

**Usage**

```r
base.rl2insert(x, descx, vec, i, j)
```

**Arguments**

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

base.rpdgecon \hspace{1cm} rpdgecon

**Description**

Inverse condition number of a general matrix.

**Usage**

```r
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.
**Description**

Matrix-Matrix Multiply.

**Usage**

```latex
base.rpdgemm(transx, transy, x, descx, y, descy, descc)
```

**Arguments**

- `transx, transy` 'T' or 'N' for transpose or not.
- `x, y` Matrix.
- `descx, descy, descc` ScaLAPACK descriptor array.

**Details**

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

---

**Description**

General 2d block cyclic redistribution function.

**Usage**

```latex
base.rpdgemr2d(x, descx, descy)
```

**Arguments**

- `x` Matrix.
- `descx, descy` ScaLAPACK descriptor array.

**Details**

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

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>base.rpdgeqpf(tol, m, n, x, descx, comm = .pbd_env$SPMD.CT$comm)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tol</code></td>
</tr>
<tr>
<td><code>m, n</code></td>
</tr>
<tr>
<td><code>x</code></td>
</tr>
<tr>
<td><code>descx</code></td>
</tr>
<tr>
<td><code>comm</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>For advanced users only. See pbdDMAT for high-level functions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A list contains QR results.</td>
</tr>
</tbody>
</table>

---

---

### base.rpdgesv  
**rpdpesv**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solving a (square) system of equations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>base.rpdgesv(n, nrhs, a, desca, b, descb)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
</tr>
<tr>
<td><code>nrhs</code></td>
</tr>
<tr>
<td><code>a, b</code></td>
</tr>
<tr>
<td><code>desca, descb</code></td>
</tr>
</tbody>
</table>
base.rpdgesvd

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

Description
SVD.

Usage
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.

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.
**base.rpdlange**  

*rpdlange*

---

**Description**

Matrix norms.

**Usage**

base.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.

---

**base.rpdlaprint**  

*rpdlaprint*

---

**Description**

Matrix printer.

**Usage**

base.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.
base.rpdorglq  \hspace{1cm} \texttt{rpdorglq}

\begin{itemize}
  \item \textbf{Description} \hspace{1cm} Recover Q.
  \item \textbf{Usage} \hspace{1cm} \texttt{base.rpdorglq(m, n, k, lq, desc, tau)}
  \item \textbf{Arguments} \hspace{1cm}
    \begin{itemize}
      \item \hspace{1cm} \texttt{m, n} \hspace{1cm} Problem size.
      \item \hspace{1cm} \texttt{k} \hspace{1cm} Number of elementary reflectors.
      \item \hspace{1cm} \texttt{lq} \hspace{1cm} QR decomposition.
      \item \hspace{1cm} \texttt{desc} \hspace{1cm} ScaLAPACK descriptor array.
      \item \hspace{1cm} \texttt{tau} \hspace{1cm} Elementary reflectors.
    \end{itemize}
  \item \textbf{Details} \hspace{1cm} For advanced users only. See \texttt{pbdDMAT} for high-level functions.
  \item \textbf{Value} \hspace{1cm} Q matrix of the QR decomposition.
\end{itemize}

---

base.rpdorgqr  \hspace{1cm} \texttt{rpdorgqr}

\begin{itemize}
  \item \textbf{Description} \hspace{1cm} Recover Q.
  \item \textbf{Usage} \hspace{1cm} \texttt{base.rpdorgqr(m, n, k, qr, descqr, tau)}
  \item \textbf{Arguments} \hspace{1cm}
    \begin{itemize}
      \item \hspace{1cm} \texttt{m, n} \hspace{1cm} Problem size.
      \item \hspace{1cm} \texttt{k} \hspace{1cm} Number of elementary reflectors.
      \item \hspace{1cm} \texttt{qr} \hspace{1cm} QR decomposition.
      \item \hspace{1cm} \texttt{descqr} \hspace{1cm} ScaLAPACK descriptor array.
      \item \hspace{1cm} \texttt{tau} \hspace{1cm} Elementary reflectors.
    \end{itemize}
\end{itemize}
Details

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

Value

Q matrix of the QR decomposition.

---

Description

\( \text{op}(Q) \ast y \).

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.
Description

Cholesky factorization.

Usage

base.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.

Description

Symmetric eigenvalue decomposition.

Usage

base.rpdsyevr(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

```r
code
base.rpd svexv (
  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**

```plaintext
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**

```plaintext
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.
**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.

---

**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).
Blacsexit

Blacsexit

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

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, .pdbBASEEnv 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.

cordspair | Global to Local/Local to Global Pair Indexing

Description
Get the local index-pair given global information.

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

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

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)
bldim <- 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()"

"
### 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
**Description**

Global to local coordinates with explicit ownership given.

**Usage**

\[ \text{g2lcoord}(\text{dim}, \text{bldim}, \text{gi}, \text{gj}, \text{gridinfo}) \]

**Arguments**

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

**Value**

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

---

**Description**

Global to local coords.

**Usage**

\[ \text{base.g2l Coord}(\text{ind}, \text{bldim}, \text{ICTXT} = 0, \text{dim} = \text{NULL}) \]

**Arguments**

- **ind**: Matrix indices.
- **bldim**: Blocking dimension.
- **ICTXT**: BLACS context.
- **dim**: Ignored; will be removed in a future version.
**get.comm.from.ICTXT**

**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

**Description**
Frees a BLACS context.

**Usage**
```
base.gridexit(ICTXT, override = FALSE)
ggridexit(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 pbdDMAT 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.

---

gridinfo Get BLACS Context Grid Information
---

Description

Grabs the existing BLACS context grid information.

Usage

base.blacs(ICTXT = 0)

blacs(ICTXT = 0)

Arguments

ICTXT BLACS context number.

Details

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.

The function effectively serves as a shorthand for

eval(parse(text=paste(".__blacs_gridinfo_",ICTXT,sep="")))

Value

Returns a list with 5 elements: NPROW and NPCOL, the number of process rows and columns respectively; ICTXT, the associated BLACS context number; MYROW and MYCOL, the current process’ row and column position in the process grid.
gridinit

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)
```

---

blacs_init

description

BLACS grid initialization.

Usage

```r
base.blacs_init(ICXT, NPROW, NPCOL, ..., quiet = FALSE)
blacs_init(ICXT, NPROW, NPCOL, ..., quiet = FALSE)
blacs_gridinit(ICXT, 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


InitGrid

.Initialize Process Grid.

Description

Manages the creation of BLACS context grids.

Usage

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

Arguments

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

Details

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

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

\begin{itemize}
  \item .\_\_blacs_gridinfo\_0
  \item .\_\_blacs_gridinfo\_1
  \item .\_\_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 \text{init.grid()} in this fashion, namely with \text{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 \text{NPROW} by \text{NPCOL} processes; contexts 1 is the process grid consisting of 1 process row and \text{NPROW*NPCOL} processes columns; context 2 is the process grid consisting of \text{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 \text{NPROW} and \text{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 .\_\_blacs_gridinfo\_ICTXT are just storage mechanisms to avoid needing to directly invoke the BLACS routine 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

**Description**

Local to global coords.

**Usage**

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

```r
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.

---

**numroc2**

<table>
<thead>
<tr>
<th>Description</th>
<th>A better version of NUMROC (NUMber Rows Or Columns). Returns the local dimension given global matrix + distribution parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td><code>numroc2(N, NB, IPROC, NPROCS)</code></td>
</tr>
</tbody>
</table>
| 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**

<table>
<thead>
<tr>
<th>Description</th>
<th>Interchange Between Process Number and BLACS Coordinates</th>
</tr>
</thead>
</table>
| Usage       | `base.pnum(ICTX, PROW, PCOL)`  
              | `base.pcoord(ICTX, PNUM)` |
| Description | Grabs the existing BLACS context grid information. |

---
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 \(c(MYPROW, MYPCOL)\), while the latter does the reverse.

Value

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

Examples

```r
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)
```

---

**sys2blacs.handle**  
*Context Within a Given Communicator*

**Description**

Creates a context that will be valid for a given communicator

**Usage**

```r
sys2blacs.handle(comm)
```
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.dgam2d (base.dgamx2d), 9
base.dgamx2d, 9
base.dger2d (base.dgesd2d), 9
base.dgesd2d, 9
base.igam2d (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.igam2d (base.dgamx2d), 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.mkgbmat (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.pcoor (pcoords), 52
base.pdchtri, 19
base.pdc1var, 20
base.pdhilbmk, 20
base.pdkcpn1, 21
base.pdmvsum, 21
base.pdsweep, 22
base.pnum (pcoords), 52
base.procgrid, 22
base.rcolcpy, 24
base.rcolcpy2, 25
base.redist, 25
base.r12blas, 26
base.r12insert, 27
base.rpdgecon, 27
base.rpdgelqf, 28
base.rpdgels, 28
base.rpdgemm, 29
base.rpdgemr2d, 29
base.rpdgeqpf, 30
base.rpdgetrf, 30
base.rpdgetri, 30
base.rpdgesv, 30
base.rpdgesvd, 30
base.rpdgesv, 31
base.rpdgetrf, 32
base.rpdgetri, 32
base.rpdlange, 33
base.rpdlaprt, 33
base.rpdorglq, 34
base.rpdorgqr, 34
base.rpdormqr, 35
base.rpdpotrf, 36
base.rpdsyevr, 36
base.rpdsyevx, 37
base.rpdttran, 38
base.rpdtrcon, 38
base.rrowcpy, 39
base.rrowcpy2, 39
base.tri2zero, 40
base.valid_context, 40
blacs(gridinfo), 48
blacs_apts, 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_apts(blacs_apts), 42
get.comm.from.ICTXT, 47
gridexit, 47
gridinfo, 48
gridinit, 49
indx2l(coords), 42
indx2l2g(coords), 42
init.grid(InitGrid), 50
InitGrid, 50
l2g_coord, 51
l2gpair(coordspair), 43
numroc2, 52

pbdBASE-package, 3
pcoords, 52

set.blacs_apts(blacs_apts), 42
sys2blacs.handle, 53