Package 'pbdBASE'

February 5, 2018

Type Package

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

Version 0.4-5.1

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 (>= 3.0.0), methods

Imports pbdMPI (>= 0.3-1), pbdSLAP(>= 0.2-1), utils

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 http://r-pbd.org/

BugReports http://group.r-pbd.org/

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

Maintainer ORPHANED

RoxygenNote 5.0.1

Author Drew Schmidt [aut, cre],
Wei-Chen Chen [aut],
George Ostrouchov [aut],
Pragneshkumar Patel [aut],
Ewan Higgs [ctb]

Repository CRAN

Date/Publication 2018-02-05 10:46:53 UTC

X-CRAN-Original-Maintainer Drew Schmidt <schmidt@math.utk.edu>

X-CRAN-Comment Orphaned and corrected on 2018-02-05 as C++ programming error reported by clang was not corrected despite reminders.

R topics documented:

`pbdBASE-package` *ScaLAPACK Wrappers and Utilities*

Description

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

Details

Package:	pbdBASE
Type:	Package
License:	MPL
LazyLoad:	yes

This package requires an MPI library (OpenMPI, MPICH2, or LAM/MPI).

Author(s)

Drew Schmidt <schmidt AT math.utk.edu>, Wei-Chen Chen, George Ostrouchov, and Pragneshkumar Patel.

References

Programming with Big Data in R Website: http://r-pbd.org/

BASE Global Environment

Global Environment for the pbdBASE Package

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.crossprod *crossprod*

Description

Crossproduct.

Usage

```
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.
descc	ScaLAPACK descriptor array of output.

Details

For advanced users only.

---

base.dallreduce *dallreduce*

Description

Allreduce

Usage

```
base.dallreduce(x, descx, op = "sum", scope = "All")
```

Arguments

x	Matrix.
descx	ScaLAPACK descriptor array.
op	Operation.
scope	Rows, columns, or both.

Details

For advanced users only.

base.descinit *descinit*

Description

Creates ScaLAPACK descriptor array.

Usage

```
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.

base.dgamx2d *BLACS Min*

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.

`base.dgesd2d` *BLACS Point to Poin*

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`	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.

`base.dhilbmk` *dhilbmk*

Description

Create Hilbert matrix.

Usage

```
base.dhilbmk(n)
```

Arguments

`n`	Size.

Details

For advanced users only.

`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.

`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.

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)
```

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.

base.indxg2p *indxg2p*

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 function.

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.

`base.matexp` *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.

`base.maxdim` *maxdim*

Description

Compute maximum dimension across all nodes

Usage

```
base.maxdim(dim)
```

Arguments

`dim`	Global dim.

Details

For advanced users only.

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.

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, rsrc, csrc)
```

Arguments

x	Matrix.
descx	ScaLAPACK descriptor array.
rsrc, csrc	Row/column source.

Details

For advanced users only.

`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

Suprisingly useful for thinking about processor grid shapes.

Examples

```
## Not run:
library(pbdBASE, quiet = TRUE)
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

## End(Not run)
```

`base.numroc` *numroc*

Description

NUMber of Rows Or Columns

Usage

```
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.

---

base.ownany *Determining Local Ownership of a Distributed Matrix*

Description

For advanced users only.

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 CTXT= 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.

Examples

```
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()

iown <- ownany(dim=c(4, 4), bldim=c(2, 2), CTXT=0)
```

```
comm.print(iown, all.rank=T)

finalize()

## End(Not run)
```

`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.

`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

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

Arguments

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

base.pdhilbmk *pdhilbmk*

Description

Create Hilbert matrix.

Usage

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

Arguments

descx	ScaLAPACK descriptor matrix.

Details

For advanced users only.

base.pdmkcpn1 *pdmkcpn1*

Description

Create Companion Matrix

Usage

```
base.pdmkcpn1(coef, descx)
```

Arguments

coef	Coefficients vector.
descx	ScaLAPACK descriptor array.

Details

For advanced users only.

base.pdmvsum *R-like Matrix-Vector Sum*

Description

For advanced users only.

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.

`base.procgrid` *procgrid*

Description

"Optimal" process grid when nprow and npcol are empty

Usage

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

Arguments

`nprocs`	Number of processors.

Details

For advanced users only.

`base.p_matexp_pade_wrap` *p_matexp_pade_wrap*

Description

Pade' expansion.

Usage

```
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.

base.p_matpow_by_squaring_wrap *p_matpow_by_squaring_wrap*

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.

base.rcolcpy *R Column Copy*

Description

For advanced users only.

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.

Usage

```
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

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

Arguments

desc	ScaLAPACK descriptor array.
A	Matrix.

base.rl2blas *Level 2 R-like BLAS*

Description

For advanced users only.

Usage

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

Arguments

x	Matrix.
descx	ScaLAPACK descriptor array.
vec	Global vector.
FUN	Function.

base.rl2insert *R-like Matrix-Vector Insertion*

Description

For advanced users only.

Usage

```
base.rl2insert(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.

base.rpdgels *rpdgels*

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.

base.rpdgemm *rpdgemm*

Description

Matrix-Matrix Multiply.

Usage

```
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.

base.rpdgemr2d *rpdgemr2d*

Description

General 2d block cyclic redistribution function.

Usage

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

Arguments

x	Matrix.
descx, descy	ScaLAPACK descriptor array.

Details

For advanced users only.

base.rpdgeqpf *rpdgeqpf*

Description

QR.

Usage

```
base.rpdgeqpf(tol, m, n, x, descx)
```

Arguments

tol	Numerical tolerance for the QR.
m, n	Problem size.
x	Matrix.
descx	ScaLAPACK descriptor array.

Details

For advanced users only.

base.rpdgesv *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.

Details

For advanced users only.

base.rpdgesvd *rpdgesvd*

Description

SVD.

Usage

```
base.rpdgesvd(jobu, jobvt, m, n, a, desca, descu, descvt, ...,
  inplace = FALSE)
```

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.

Details

For advanced users only.

base.rpdgetrf *rpdgetrf*

Description

LU factorization.

Usage

```
base.rpdgetrf(a, desca)
```

Arguments

a	Matrix.
desca	ScaLAPACK descriptor array.

Details

For advanced users only.

base.rpdgetri *rpdgetri*

Description

Matrix inversion.

Usage

```
base.rpdgetri(n, a, desca)
```

Arguments

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

Details

For advanced users only.

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.

base.rpdlaprnt *rpdlaprnt*

Description

Matrix printer.

Usage

```
base.rpdlaprnt(m, n, a, desca)
```

Arguments

m, n	Number rows/cols.
a	Matrix.
desca	ScaLAPACK descriptor array.

Details

For advanced users only.

base.rpdorgqr *rpdorgqr*

Description

Recover Q.

Usage

```
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.

Details

For advanced users only.

base.rpdormqr *rpdormqr*

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 righth 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.

base.rpdpotrf *rpdpotrf*

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.

`base.rpdsyevr` *rpdsyevr*

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.

`base.rpdsyevx` *rpdsyevx*

Description

Genearlized 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.

---

base.rpdtran *rpdtran*

Description

Transpose.

Usage

```
base.rpdtran(a, desca, descc)
```

Arguments

a	Matrix.
desca, descc	ScaLAPACK descriptor array.

Details

For advanced users only.

base.rpdtrcon *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.

base.rrowcpy *R Row Copy*

Description

For advanced users only.

Usage

```
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.

Usage

```
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

```
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.

base.valid_context *BLACS Context Validation*

Description

Checks if a supplied ICTXT is valid.

Usage

```
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 *BLACS Exit*

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

```
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()

blacsexit()

finalize()

## End(Not run)
```

---

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 distributed matrix. That's always 0 pbdDMAT.
NPROCS	Total number of processors over which matrix is distributed.
INDXLOC	Local index.

Details

For advanced users only.

coordspair *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

gi, gj	Global indices.
bldim	Blocking dimension
ICTXT	BLACS context.
i, j	Local indices.

Details

For advanced users only.

diag *diag*

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.

finalizer *Finalizer*

Description

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

Usage

```
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.

g2lcoord *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(ICTXT(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 *g2l_coord*

Description

Global to local coords.

Usage

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

g2l_coord(ind, dim, bldim, ICTXT = 0)
```

Arguments

ind	Matrix indices.
dim	Global dim.
bldim	Blocking dimension.
ICTXT	BLACS context.

Details

For advanced users only.

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.

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.

Examples

```
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()

mygrid <- blacs(0)

pbdMPI::comm.print(mygrid)

finalize()

## End(Not run)
```

`gridinit` *blacs_gridinit*

Description

BLACS grid initialization.

Usage

```
base.blacs_gridinit(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.

InitGrid *Initialize Process Grid*

Description

Manages the creation of BLACS context grids.

Usage

```
init.grid(NPROW, NPCOL, ICTXT, quiet = FALSE)
```

Arguments

NPROW	number of process rows. Can be missing; see details.
NPCOL	number of process columns. Can be missing; see details.
ICTXT	BLACS context number.
quiet	logical; controls whether or not information about grid size should be printed.

Details

`blacs_gridinit()` is for experienced users only. It is a shallow wrapper of the BLACS routine `BLACS_GRIDINIT`, with the addition of creating the `.__blacs_gridinfo_ICTXT` objects, as described below.

The remainder of this section applies only to `init.grid()`.

If `ICTXT` is missing, three variables will be created in the `.pbdBASEEnv` environment:

`.__blacs_gridinfo_0`

`.__blacs_gridinfo_1`

`.__blacs_gridinfo_2`

These variables store the BLACS process grid information for the BLACS context corresponding to the trailing digit of the variable. Most users should invoke `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 >= 3.

Context 0 is the "full" process grid of `NPROW` by `NPCOL` processes; contexts 1 is the process grid consisting of 1 process row and `NPROW*NPCOL` processes columns; context 2 is the process grid consisting of `NPROW*NPCOL` processes rows and 1 process column. These contexts can be redundant depending on the number of prcesses 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 `NPROW` and `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

```
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()

finalize()

## End(Not run)
```

---

l2g_coord *l2g_coord*

Description

Local to global coords.

Usage

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

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

Arguments

ind	Matrix indices.
dim	Global dim.
bldim	Blocking dimension.
ICTXT	BLACS context.

Details

For advanced users only.

numroc2 *numroc2*

Description

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

Usage

```
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.

pcoords *Interchange Between Process Number and BLACS Coordinates*

Description

Grabs the existing BLACS context grid information.

Usage

```
base.pnum(ICTXT, PROW, PCOL)

base.pcoord(ICTXT, 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, MYPCOL), while the latter does the reverse.

Value

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

Examples

```
## Not run:
# Save code in a file "demo.r" and run with 2 processors by
# > mpiexec -np 2 Rscript demo.r

library(pbdBASE, quiet = TRUE)
init.grid()

blacs_ <- blacs(ICTXT = 0)

# get the ICTXT = 0 BLACS coordsinates for process 0
myCoords <- pcoord(ICTXT = 0, PNUM = 0)

comm.print(myCoords)

finalize()

## End(Not run)
```

Index