

# Package ‘pbapply’

February 15, 2012

**Type** Package

**Title** Adding Progress Bar to ‘\*apply’ Functions

**Version** 1.0-4

**Date** 2010-09-08

**Author** Peter Solymos

**Maintainer** Peter Solymos <solymos@ualberta.ca>

**Description** A lightweight package that adds progress bar to vectorized R functions (\*apply). The implementation can easily be added to functions, where showing the progress is useful for the user (e.g. bootstrap).

**License** GPL-2

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2011-10-24 06:17:12

## R topics documented:

pbapply . . . . .	2
pboptions . . . . .	4

<b>Index</b>	<b>8</b>
--------------	----------

**Description**

Adding progress bar to \*apply functions

**Usage**

```

pbapply(X, FUN, ...)
pbapply(X, MARGIN, FUN, ...)
pbsapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
pbreplicate(n, expr, simplify = TRUE)

```

**Arguments**

X	For pbsapply and pbapply, a vector (atomic or list) or an expressions vector (other objects including classed objects will be coerced by <code>as.list()</code> ). For pbapply an array, including a matrix.
MARGIN	A vector giving the subscripts which the function will be applied over. 1 indicates rows, 2 indicates columns, <code>c(1, 2)</code> indicates rows and columns.
FUN	The function to be applied to each element of X: see <a href="#">apply</a> , <a href="#">sapply</a> , and <a href="#">lapply</a> .
...	Optional arguments to FUN.
simplify	Logical; should the result be simplified to a vector or matrix if possible?
USE.NAMES	Logical; if TRUE and if X is character, use X as names for the result unless it had names already.
n	Number of replications.
expr	Expression (language object, usually a call) to evaluate repeatedly.

**Details**

The behaviour of the progress bar is controlled by the option `type` in [pboptions](#), it can take values `c("txt", "win", "tk", "none",)` on Windows, and `c("txt", "tk", "none",)` on Unix systems.

Other options have elements that are arguments used in the functions [txtProgressBar](#) (for `"pbapply.txt"` option), and [tkProgressBar](#).

See [pboptions](#) for how to conveniently set these.

**Value**

Similar to the value returned by the standard \*apply functions.

A progress bar is showed as a side effect.

**Note**

Progress bar can add an overhead to the computation.

**Author(s)**

Peter Solymos <solymos@ualberta.ca>

**See Also**

Progress bars used in the functions: [txtProgressBar](#), [tkProgressBar](#)

Standard \*apply functions: [apply](#), [sapply](#), [lapply](#), [replicate](#)

Setting the options: [pboptions](#)

Conveniently add progress bar to for-like loops: [startpb](#), [setpb](#), [getpb](#), [closepb](#)

**Examples**

```
## simple linear model simulation
n <- 200
x <- rnorm(n)
y <- rnorm(n, model.matrix(~x) %*% c(0,1), sd=0.5)
d <- data.frame(y, x)
## model fitting and bootstrap
mod <- lm(y~x, d)
ndat <- model.frame(mod)
B <- 100
bid <- sapply(1:B, function(i) sample(nrow(ndat), nrow(ndat), TRUE))
fun <- function(z) {
  ndat <- ndat[sample(nrow(ndat), nrow(ndat), TRUE),]
  coef(lm(mod$call$formula, data=ndat[z,]))
}
## standard '*apply' functions
system.time(res1 <- lapply(1:B, function(i) fun(bid[i])))
system.time(res2 <- sapply(1:B, function(i) fun(bid[i])))
system.time(res3 <- apply(bid, 2, fun))
## 'pb*apply' functions
## try different settings:
## "none", "txt", "tk"
options("pbapply.pb"="txt")
system.time(res4 <- pblapply(1:B, function(i) fun(bid[i])))
system.time(res5 <- pbsapply(1:B, function(i) fun(bid[i])))
system.time(res6 <- pbapply(bid, 2, fun))

## Examples taken from standard '*apply' functions

## sapply and lapply

require(stats); require(graphics)

x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
# compute the list mean for each list element
pblapply(x,mean)
```

```

# median and quartiles for each list element
pblapply(x, quantile, probs = 1:3/4)
pbsapply(x, quantile)
i39 <- pbsapply(3:9, seq) # list of vectors
pbsapply(i39, fivenum)

## replicate

foo <- function(x=1, y=2) c(x,y)
# does not work: bar <- function(n, ...) replicate(n, foo(...))
bar <- function(n, x) pbreplicate(n, foo(x=x))
bar(5, x=3)

## apply

## Compute row and column sums for a matrix:
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
dimnames(x)[[1]] <- letters[1:8]
pbapply(x, 2, mean, trim = .2)
col.sums <- pbapply(x, 2, sum)
row.sums <- pbapply(x, 1, sum)
rbind(cbind(x, Rtot = row.sums), Ctot = c(col.sums, sum(col.sums)))

stopifnot( pbapply(x, 2, is.vector))

## Sort the columns of a matrix
pbapply(x, 2, sort)

##- function with extra args:
cave <- function(x, c1, c2) c(mean(x[c1]), mean(x[c2]))
pbapply(x,1, cave, c1="x1", c2=c("x1","x2"))

ma <- matrix(c(1:4, 1, 6:8), nrow = 2)
ma
pbapply(ma, 1, table) #--> a list of length 2
pbapply(ma, 1, stats::quantile)# 5 x n matrix with rownames

stopifnot(dim(ma) == dim(pbapply(ma, 1:2, sum)))

## Example with different lengths for each call
z <- array(1:24, dim=2:4)
zseq <- pbapply(z, 1:2, function(x) seq_len(max(x)))
zseq          ## a 2 x 3 matrix
typeof(zseq) ## list
dim(zseq)    ## 2 3
zseq[1,]
pbapply(z, 3, function(x) seq_len(max(x)))
# a list without a dim attribute

```

**Description**

Creating progress bar and setting options.

**Usage**

```
pboptions(...)
startpb(min = 0, max = 1)
setpb(pb, value)
getpb(pb)
closepb(pb)
dopb()
```

**Arguments**

...	Arguments in tag = value form, or a list of tagged values. The tags must come from the parameters described below.
pb	A progress bar object created by startpb.
min, max	Finite numeric values for the extremes of the progress bar. Must have min < max.
value	New value for the progress bar.

**Details**

pboptions is a convenient way of handling options related to progress bar.

Other functions can be used for conveniently adding progress bar to for-like loops (see Examples).

**Value**

When parameters are set by pboptions, their former values are returned in an invisible named list. Such a list can be passed as an argument to pboptions to restore the parameter values. Tags are the following:

type	Type of the progress bar: text("txt"), Windows("win"), TclTk("tk"), or none("none"). Default value is "txt".
char	The character (or character string) to form the progress bar. Default value is "+".
txt.width	The width of the text based progress bar, as a multiple of the width of char. If NA, the number of characters is that which fits intogetOption("width"). Default value is 50.
gui.width	The width of the GUI based progress bar in pixels: the dialog box will be 40 pixels wider (plus frame). Default value is 300.
style	The style of the bar, see <a href="#">txtProgressBar</a> . Default value is 3.
initial	Initial value for the progress bar. Default value is 0.
title	Character string giving the window title on the GUI dialog box. Default value is "R progress bar".

label            Character string giving the window label on the GUI dialog box. Default value is "".

For startpb a progress bar object.

For getpb and setpb, a length-one numeric vector giving the previous value (invisibly for setpb). The return value is NULL if the progress bar is turned off by getOption("pbapply.pb") ("none" or NULL value).

dopb returns a logical value if progress bar is to be shown based on the option getOption("pbapply.pb"). It is FALSE if the type of progress bar is "none" or NULL.

For closepb closes the connection for the progress bar.

### Author(s)

Peter Solymos <solymos@ualberta.ca>

### See Also

Progress bars used in the functions: [txtProgressBar](#), [tkProgressBar](#)

### Examples

```
## for loop
fun1 <- function() {
  pb <- startpb(0, 10)
  for (i in 1:10) {
    Sys.sleep(0.2)
    setpb(pb, i)
  }
  closepb(pb)
}

## while loop
fun2 <- function() {
  pb <- startpb(0, 10-1)
  i <- 1
  while (i < 10) {
    Sys.sleep(0.2)
    setpb(pb, i)
    i <- i + 1
  }
  closepb(pb)
}

## using original settings
fun1()
## resetting pboptions
opb <- pboptions(style=1, char=">")
## check new settings
getOption("pboptions")
## running again with new settings
fun2()
## resetting original
pboptions(opb)
```

*poptions*

7

```
## check reset  
getOption("poptions")  
fun1()
```

# Index

## \*Topic **IO**

    pboptions, 4

## \*Topic **manip**

    pbapply, 2

## \*Topic **utilities**

    pbapply, 2

    pboptions, 4

apply, 2, 3

as.list, 2

closepb, 3

closepb (pboptions), 4

dopb (pboptions), 4

getpb, 3

getpb (pboptions), 4

lapply, 2, 3

pbapply, 2

pbapply (pbapply), 2

pboptions, 2, 3, 4

pbreplicate (pbapply), 2

pbsapply (pbapply), 2

replicate, 3

sapply, 2, 3

setpb, 3

setpb (pboptions), 4

startpb, 3

startpb (pboptions), 4

tkProgressBar, 2, 3, 6

txtProgressBar, 2, 3, 5, 6