Package 'mewAvg'

October 13, 2022

Version 0.3.1

Title A Fixed Memeory Moving Expanding Window Average

Author Adam L. Pintar and Zachary H. Levine

Maintainer Adam L. Pintar <adam.pintar@nist.gov>

Depends methods

Description Compute the average of a sequence of random vectors
in a moving expanding window using a fixed amount of memory.

License GPL (>= 2.0)

NeedsCompilation yes

ByteCompile TRUE

RoxygenNote 7.1.2

Repository CRAN

Date/Publication 2022-04-25 15:10:02 UTC

R topics documented:

mewAvg-package *A fixed memory moving expanding window average*

Description

This package provides the tools to calculate an average in a moving expanding window (MEW) using a fixed amount of memory.

Details

See the examples for the functions `mewMean` and `mewAvg` for the details of use.

References

Levine, Z. H., & Pintar, A. L. (2015). A fixed-memory moving, expanding window for obtaining scatter corrections in x-ray CT and other stochastic averages. Computer Physics Communications, 196, 455-459.

mewAccum *Update the class* `mewTyp`

Description

Update an S4 object of class `mewTyp` with a new data point

Usage

```
mewAccum(xx, av)
```

Arguments

`xx`	(vector double) The vector of data with which to update the MEW aveage
`av`	(class mewTyp) The current state of the MEW average

Details

If `av` is an S4 object of class `mewTyp` that contains the current state of the MEW average and `xx` is a new vector of data, the function `mewAccum` updates the MEW average with `xx`.

Value

The updated instance of `av`

Examples

```
n_iter <- 1000

av <- mewInit(n_bin = 4, n_xx = 1, ff = 0.5)

for (i in 1:n_iter) {

  value <- runif(n=2)
  value[1] <- ((cos(value[1]*2*pi))^2)*(1 - exp(-0.01*i))
  value[2] <- (-((sin(value[2]*2*pi))^2))*(1 - exp(-0.01*i))
  value <- as.double(value)

  av <- mewAccum(xx = value, av = av)
}
```

mewAvg *Convenience wrapper for the MEW process*

Description

Packages the process of calling `mewInit`, looping through the random vectors calling `mewAccum` for each one and calling `mewMean` when desired.

Usage

```
mewAvg(f, n.bin, n.xx, ff, n.save = NULL, n.iter = NULL, i.to.save, ...)
```

Arguments

`f` (function) A user defined R function. See the 'Details' section for more on defining this function

`n.bin` (scalar integer) The fixed number of bins to use to define the moving expanding window

`n.xx` (scalar integer) The length of the numeric vector returned by `f`

`ff` (scalar double) The fraction of the samples to included in each window

`n.save` (scalar integer OR NULL)The number of estimates to save and return. The default value is NULL since this argument can be derived from `i.to.save`. The argument is kept for compatibility with older versions of this package

`n.iter` (scalar integer OR NULL) The number of times to call `f`. The default value is NULL since this argument can be derived from `i.to.save`. The argument is kept for compatibility with older versions of this package

`i.to.save` (vector integer length n.iter) A vector of zeros and ones of length `n.iter` where position `i` is 1 if an average should be calculated and saved at iteration i, and zero otherwise

`...` The initial named arguments to `f`.

Details

The function f should generate the sequence of random vectors one at a time. The returned value from a single call should be a list with at least one element. The first element should be a numeric vector of length n.xx (the next vector in the sequence), and the remaining elements should be the updated arguments for the next call to f, named appropriately for the argument of f to update. The 'Examples' section provides further guidance.

The downfall of this interface is that the user cannot run the algorithm for some number of iterations, pause, assess convergence of the mean and then pick up from where they paused. To accomplish that see the examples associated with the mewMean function.

Value

A matrix of dimension n.save by n.xx containing the saved averages

Examples

```
MyFun <- function (k) {

 value <- runif(n=2)
 value[1] <- ((cos(value[1]*2*pi))^2)*(1 - exp(-0.01*k))
 value[2] <- (-((sin(value[2]*2*pi))^2))*(1 - exp(-0.01*k))

 k <- k + 1

 return(list(value=value, k=k))
}

i.to.save <- seq(from=1, to=1025, by=32)
tmp <- rep(x=0, times=1025)
tmp[i.to.save] <- 1
i.to.save <- tmp

mean.vals <- mewAvg(f=MyFun,
                    n.bin=4,
                    n.xx=2,
                    ff=0.5,
                    n.save=sum(i.to.save),
                    n.iter=length(i.to.save),
                    i.to.save=i.to.save,
                    k=1)

plot(c(1:sum(i.to.save),
       1:sum(i.to.save)),
     c(mean.vals[, 1],
       mean.vals[, 2]),
     type="n",
     xlab="Saved Iter",
     ylab="Mean")
points(1:sum(i.to.save),
       mean.vals[, 1])
points(1:sum(i.to.save),
       mean.vals[, 2])

## an AR(1) process

ArOne <- function (x.old, phi, sig.eps) {

  value <- phi*x.old + rnorm(n=1, mean=0, sd=sig.eps)

  return(list(value=value, x.old=value))
}

mean.vals.ar1 <- mewAvg(f=ArOne,
                        n.bin=4,
                        n.xx=1,
                        ff=0.5,
                        n.save=sum(i.to.save),
                        n.iter=length(i.to.save),
                        i.to.save=i.to.save,
                        x.old=0,
                        phi=0.5,
                        sig.eps=1)

plot(x=c(1, sum(i.to.save)),
     y=c(-0.5, 0.5),
     xlab="Saved Iter",
     ylab="Mean",
     type="n")
points(x=1:sum(i.to.save),
       y=mean.vals.ar1)
abline(h=0, col="red")
```

mewGetMean *Extract MEW average value*

Description

Return the current value of the moving expanding window (MEW) average if it is up-to-date; otherwise, raise an error

Usage

```
mewGetMean(av)
```

Arguments

av	The current state of the MEW average

Value

(vector double length n_xx) the current value of the MEW average if it is up-to-date

Examples

```
## see the examples for the function \code{mewMean}
```

---

mewInit *Create an S4 object of class* `mewTyp`

Description

Call this function to create an S4 object of class `mewTyp`.

Usage

```
mewInit(n_bin, n_xx, ff)
```

Arguments

`n_bin`	(scalar integer) The fixed number of bins to use to define the moving expanding window
`n_xx`	(scalar integer) The length of each vector in the sequence to be averaged
`ff`	(scalar double) The fraction of the samples to included in each window

Details

If it is necessary to directly call mewAccum and mewMean an S4 object of class `mewTyp` should be created first using this function. The user should never create an S4 object of class `mewTyp` using the `new` function provided by the `methods` package.

Value

An initialized instance of the class `mewTyp`

Examples

```
av <- mewInit(n_bin = 4, n_xx = 2, ff = 0.5)
```

mewMean *Update the moving expanding window average*

Description

When desired, the x_mean slot in an S4 object of class mewTyp may be updated to contain the correct moving expanding window (MEW) average (it is not updated by the function mewAccum to save computation). If the slot know_mean is unity, the slot x_mean is up-to-date; otherwise; it is not.

Usage

```
mewMean(av)
```

Arguments

av	(class mewTyp) the current state of the MEW average

Value

the updated instance of the argument av

Examples

```
n_iter <- 100

i_to_print <- 10

results <- matrix(data = double(2*n_iter/i_to_print),
                  nrow = n_iter/i_to_print,
                  ncol = 2)

av <- mewInit(n_bin = 4, n_xx = 2, ff = 0.5)

for (i in 1:n_iter) {

  value <- runif(n=2)
  value[1] <- ((cos(value[1]*2*pi))^2)*(1 - exp(-0.01*i))
  value[2] <- (-((sin(value[2]*2*pi))^2))*(1 - exp(-0.01*i))

  av <- mewAccum(xx = value, av = av)

  if (i%%i_to_print == 0) {

    av <- mewMean(av)
    show(av)
    results[i/i_to_print, ] <- mewGetMean(av)
  }
}

## plot the results
```

```
plot(c(1, (n_iter/i_to_print)),
     c(min(results), max(results)),
     type = "n")
points(1:(n_iter/i_to_print), results[, 1])
points(1:(n_iter/i_to_print), results[, 2])

## Now, a larger example, and we pause part way through to assess
## convergence

n_iter <- 1000
av <- mewInit(n_bin = 4, n_xx = 5000, ff = 0.5)
for (i in 1:n_iter) {

  new_samp <- runif(n = 5000)
  av <- mewAccum(xx = new_samp, av = av)
}

av <- mewMean(av = av)

## of course each element of the mean sould converge to 0.5.  After
## 1000 iterations, the first six elements of the mean vector are
show(av)

## run another 1000 iterations
for (i in 1:1000) {

  new_samp <- runif(n = 5000)
  av <- mewAccum(xx = new_samp, av = av)
}

av <- mewMean(av)

## check the mean of the first six elements again
show(av)
```

mewTyp-class *The state of the moving expanding window average*

Description

The class holds the current state of the moving expanding window (MEW) average

Details

The user should never create, update or access an instance of this class themselves. An instance of the class should be created with the function `mewInit` and updated with the functions `mewAccum` and `mewMean`. The user can extract the current value of the MEW average with the function `mewGetMean`, and print the first six elements of the mean vector to the screen with either the `show` or `print` functions.

Slots

- `i_new` (scalar integer) The index of the bin to add the current sample to
- `i_old` (scalar integer) The index of the bin to deweight
- `know_mean` (scalar integer) flag 0: mean not known 1: mean known
- `n_bin` (scalar integer) The number of bins to use in the MEW process
- `n_bin_use` (scalar integer) The number of bins currently in use
- `n_xx` (scalar integer) The length of a vector in the sequence being averaged
- `n_part` (scalar integer) The number of samples in the bins that are not being added to or deweighted
- `m_sample` (vector integer length - n_bin) The maximum number of samples allowed in each of the bins
- `n_sample` (vector integer length - n_bin) The number of samples currently in each bin
- `x_mean` (vector double length - n_xx) The current value of the MEW average (which is up-to-date only if `know_mean == 1`)
- `x_sum_part` (vector double length - n_xx) The sum in the bins not being added to or deweighted
- `xx` (matrix dimension - n_xx × n_bin) The bin sums
- `ff` (scalar double) The fraction of samples to retain in the MEW average
- `ww` (scalar double) The factor of increase in the number of samples from one bin to the next
- `a_sample` (scalar double) The ideal number of samples in a bin (before rounding)

`show,mewTyp-method` *Print the MEW average to the screen*

Description

Print to the screen the first six elements of the current value (if it is up-to-date) of the moving expanding window (MEW) average. An error is raised if the MEW average is not up-to-date.

Usage

```
## S4 method for signature 'mewTyp'
show(object)
```

Arguments

`object`	(class mewTyp) The current state of the MEW average

Value

Upon successful exit, zero is returned invisibly.

Examples

```
## see the examples for the function mewMean
```

Index