

# Package ‘cudaBayesreg’

February 14, 2012

**Version** 0.3-13

**Date** 2011-10-18

**Title** CUDA Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis

**Author** Adelino Ferreira da Silva <afs@fct.unl.pt>

**Maintainer** Adelino Ferreira da Silva <afs@fct.unl.pt>

**Depends** R (>= 2.9.2), cudaBayesregData, oro.nifti

**SystemRequirements** nvcc (release >= 3.1) (NVIDIA Cuda Compiler driver); Linux operating system.

**Description** Compute Unified Device Architecture (CUDA) is a software platform for massively parallel high-performance computing on NVIDIA GPUs. This package provides a CUDA implementation of a Bayesian multilevel model for the analysis of brain fMRI data. A fMRI data set consists of time series of volume data in 4D space. Typically, volumes are collected as slices of 64 x 64 voxels. Analysis of fMRI data often relies on fitting linear regression models at each voxel of the brain. The volume of the data to be processed, and the type of statistical analysis to perform in fMRI analysis, call for high-performance computing strategies. In this package, the CUDA programming model uses a separate thread for fitting a linear regression model at each voxel in parallel. The global statistical model implements a Gibbs Sampler for hierarchical linear models with a normal prior. This model has been proposed by Rossi, Allenby and McCulloch in ‘Bayesian Statistics and Marketing’, Chapter 3, and is referred to as ‘rhierLinearModel’ in the R-package bayesm. A notebook equipped with a NVIDIA ‘GeForce 8400M GS’ card having Compute Capability 1.1 has been used in the tests. The data sets used in the package’s examples are available in the separate package cudaBayesregData.

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2011-10-19 16:57:39

**R topics documented:**

buildzstat.volume . . . . .	2
cudaMultireg.slice . . . . .	4
cudaMultireg.volume . . . . .	7
plot.bayesm.mat . . . . .	9
plot.hcoef.post . . . . .	10
pmeans.hcoef . . . . .	11
post.overlay . . . . .	12
post.ppm . . . . .	13
post.randeff . . . . .	15
post.shrinkage.mean . . . . .	16
post.shrinkage.minmax . . . . .	18
post.simul.betadraw . . . . .	19
post.simul.hist . . . . .	20
post.tseries . . . . .	21
premask . . . . .	22
read.fmrillice . . . . .	23
read.Zsegslice . . . . .	25
regpostsim . . . . .	26
<b>Index</b>	<b>28</b>

---

buildzstat.volume	<i>Build a Posterior Probability Map (PPM) NIFTI volume</i>
-------------------	---

---

**Description**

buildzstat.volume builds a PPM statistical volume from slices processed by cudaMultireg.volume.

**Usage**

```
buildzstat.volume(fbase=NULL, vreg=2, nu.e=3,
  rg=c(NULL,NULL), swap=FALSE, blobsize=3, savedir=tempdir())
```

**Arguments**

fbase	If fbase is left unspecified (default NULL), then user datasets need to be provided as input. Otherwise, fbase indicates the dataset prefix of one of the two demo fMRI datasets to use. see read.fmrillice for a detailed description.
vreg	regression variable to represent in PPM; default(vreg=2)
nu.e	d.f. parameter for regression error variance prior (def: 3)
rg	rg=c(first, last): a vector containing the first and last numbers of the sequence of slices to be processed. If rg=c(NULL,NULL) (default), all slices in the volume are processed.
swap	logical variable (default = 'FALSE') for choosing the right/left data display convention consistent with FSLVIEW

blobsize	numeric value (default='3'). Applies spatial contextual information in a 3D immediate neighbourhood for eliminating 3D blobs with less than 'blobsize' active voxels.
savedir	Directory (def: 'tempdir()') where the MCMC simulations for all slices are saved.

## Details

The PPM volume is built by `buildzstat.volume` after all slices in the desired range 'rg' have been processed by `cudaBayesreg.volume`. The PPM volume has the dimension of the original volume. However, when a non-null range 'rg' is specified only the slices in the range are statistically processed. The remaining slices are assumed to contain non-activated voxels. To run the examples, the data sets from the R-package **cudaBayesregData** are required.

## Value

nactive	numeric vector containing the number of estimated active voxels for each slice in range 'rg'.
---------	---

## Author(s)

Adelino Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, Portugal, <afs@fct.unl.pt>.

## References

Adelino R. Ferreira da Silva (2011). “**cudaBayesreg**: Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis.” *Journal of Statistical Software*, **44**(4), 1–24. URL <http://www.jstatsoft.org/v44/i04/>.

Adelino Ferreira da Silva (2011). **cudaBayesregData**: *Data sets for the examples used in the package cudaBayesreg*. R package version 0.3-10. URL <http://CRAN.R-project.org/package=cudaBayesregData>.

Adelino Ferreira da Silva (2011). “A Bayesian Multilevel Model for fMRI Data Analysis.”, *Computer Methods and Programs in Biomedicine*, **102**(3), 238–252.

Adelino Ferreira da Silva (2010). “**cudaBayesreg**: Bayesian Computation in CUDA.”, *The R Journal*, **2/2**, 48-55. URL [http://journal.r-project.org/archive/2010-2/RJournal\\_2010-2\\_Ferreira~da~Silva.pdf](http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Ferreira~da~Silva.pdf).

Brandon Whitcher, Volker Schmid and Andrew Thornton (2011). **oro.nifti**: *Rigorous - NIfTI Input / Output*, R package version 0.2.5. URL <http://CRAN.R-project.org/package=oro.nifti>.

## See Also

[read.fmrillice](#), [buildzstat.volume](#), [cudaMultireg.slice](#), [post.overlay](#)

## Examples

```
## Not run:
## Simulation using the visual/auditory test dataset "fmri"
cudaMultireg.volume(fbase="fmri", R=2000, savedir=tempdir())
buildzstat.volume(fbase="fmri", vreg=2)
post.overlay(fbase="fmri", vreg=2, view="axial")
buildzstat.volume(fbase="fmri", vreg=4)
post.overlay(fbase="fmri", vreg=4, view="axial")
##
## simulation using the SPM auditory dataset "swrfM*"
cudaMultireg.volume(fbase="swrfM", R=2000, rg=c(13,16), savedir=tempdir())
buildzstat.volume(fbase="swrfM", rg=c(13,16))
post.overlay(fbase="swrfM", vreg=2, rg=c(13,16), view="axial")
##

## End(Not run)
```

---

cudaMultireg.slice      *CUDA Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis on a fMRI slice*

---

## Description

cudaMultireg.slice provides an interface to a CUDA implementation of a Bayesian multilevel model for the analysis of brain fMRI data. cudaMultireg.slice processes a single slice.

## Usage

```
cudaMultireg.slice(slicedata, ymaskdata, R, keep = 5, nu.e = 3,
fsave = NA, zprior=FALSE, rng = 0)
```

## Arguments

slicedata	list(slice=slice, niislicets=niislicets, mask=mask, dsgn=dsgn); input slice data used in simulation as returned by read.fmrlice. See read.fmrlice for indications on how to process user defined datasets.
ymaskdata	list(yn = yn, kin = kin, nreg = nreg); masked and standardised slice data as returned by premask
R	number of MCMC draws
keep	MCMC thinning parameter: keep every keepth draw (def: 5)
nu.e	d.f. parameter for regression error variance prior (def: 3)
fsave	filename for saving the MCMC simulation (def: NULL do not save)
zprior	boolean {T,F}; default {F} - use just a mean for Z
rng	integer {0,1,2}: type of RNG to use {0-Marsaglia Multicarry, 1-R. P. Brent xorgens, 2-Mersenne Twister MT19937-64}; (def. 0-Marsaglia Multicarry)

## Details

The statistical model implemented in CUDA was specified as a Gibbs Sampler for hierarchical linear models with a normal prior. This model has been analysed by Rossi, Allenby and McCulloch in *Bayesian Statistics and Marketing*, Chapt. 3, and is referred to as `rhierLinearModel` in the R package **bayesm**. The main computational work is done in parallel on a CUDA capable GPU. Each thread is responsible for fitting a general linear model at each voxel. The CUDA implementation has the following system requirements: `nvcc` NVIDIA Cuda Compiler driver, `g++` GNU compiler (`nvcc` compatible version). The package includes source code files to build the library `'newmat11.so'`. This is a matrix library by R. B. Davies used in the package's host C++ code. The package includes three optional *CUDA-based* RNGs. Marsaglia's `multicarry` RNG follows the R implementation, is the fastest one, and is used by default; Brent's RNG has higher quality but is not-so-fast; Matsumoto's Mersenne Twister is slow. The data sets used in the examples are available in the R package **cudaBayesregData**.

## Value

a list containing

<code>betadraw</code>	<code>nreg x nvar x R/keep</code> array of individual regression coef draws
<code>taudraw</code>	<code>R/keep x nreg</code> array of error variance draws
<code>Deltadraw</code>	<code>R/keep x nz x nvar</code> array of <code>Deltadraws</code>
<code>Vbetadraw</code>	<code>R/keep x nvar*nvar</code> array of <code>Vbeta</code> draws

## Note

The statistical model may be specified as follows.

Model: `length(regdata)` regression equations.

$y_i = X_i \beta_i + e_i$ .  $e_i \sim N(0, \tau_i)$ . `nvar` X vars in each equation.

Priors:

$\tau_i \sim \text{nu.e} * \text{ssq}_i / \chi_{\text{nu.e}}^2$ .  $\tau_i$  is the variance of  $e_i$ .

$\beta_i \sim N(\text{ZDelta}[i,], V_{\beta_i})$ .

Note: `ZDelta` is the matrix `Z * Delta`; `[i,]` refers to `i`th row of this product.

$\text{vec}(\text{Delta})$  given  $V_{\beta_i} \sim N(\text{vec}(\text{Deltabar}), V_{\beta_i}(x)A^{-1})$ .

$V_{\beta_i} \sim \text{IW}(\text{nu}, V)$ .

`Delta`, `Deltabar` are `nz x nvar`. `A` is `nz x nz`.  $V_{\beta_i}$  is `nvar x nvar`.

By default we suppose that we don't have any `z` vars, `Z=iota` (`nreg x 1`).

Simulated objects are specified as in **bayesm** with classes `bayesm.mat` and `bayesm.var`. S3 methods to summarize marginal distributions given an array of draws are then compatible with those of **bayesm** (see Examples).

Summaries will be invoked by a call to the generic summary function as in `summary(object)` where `object` is of class `bayesm.mat` or `bayesm.var`.

A new S3 method (`hcoef.post`) is specified for dispatching `betadraw` plots.

## Author(s)

Adelino Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia, Portugal, <afs@fct.unl.pt>.

## References

Adelino R. Ferreira da Silva (2011). “**cudaBayesreg**: Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis.” *Journal of Statistical Software*, **44**(4), 1–24. URL <http://www.jstatsoft.org/v44/i04/>.

Adelino Ferreira da Silva (2011). **cudaBayesregData**: *Data sets for the examples used in the package cudaBayesreg*, R package version 0.3-10. URL <http://CRAN.R-project.org/package=cudaBayesregData>.

Adelino Ferreira da Silva (2011). “A Bayesian Multilevel Model for fMRI Data Analysis.”, *Computer Methods and Programs in Biomedicine*, **102**(3), 238–252.

Adelino Ferreira da Silva (2010). “**cudaBayesreg**: Bayesian Computation in CUDA.”, *The R Journal*, **2/2**, 48-55. URL [http://journal.r-project.org/archive/2010-2/RJournal\\_2010-2\\_Ferreira~da~Silva.pdf](http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Ferreira~da~Silva.pdf).

Rossi, Allenby and McCulloch. *Bayesian Statistics and Marketing*, Chapter 3. URL <http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html>.

Davies, R.B. (1994). *Writing a matrix package in C++*. In OON-SKI'94: The second annual object-oriented numerics conference, pp 207-213. Rogue Wave Software, Corvallis. URL [http://www.robertnz.net/cpp\\_site.html](http://www.robertnz.net/cpp_site.html).

Richard. P. Brent. *Some long-period random number generators using shifts and xors*, Preprint: 2 July 2007.

Brandon Whitcher, Volker Schmid and Andrew Thornton (2011). **oro.nifti**: *Rigorous - NIfTI Input / Output*, R package version 0.2.5. URL <http://CRAN.R-project.org/package=oro.nifti>.

## See Also

[read.fmrillice](#), [read.Zsegslice](#), [premask](#), [pmeans.hcoef](#), [regpostsim](#), [plot.hcoef.post](#), [post.simul.hist](#), [post.ppm](#), [post.tseries](#), [post.randeff](#), [post.shrinkage.mean](#)

## Examples

```
## Not run:
## Simulation using the visual/auditory test dataset "fmri"
slicedata <- read.fmrillice(fbase="fmri", slice=3, swap=FALSE)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"/simultest1",fileext = ".sav", sep="")
out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=3,
  fsave=fsave, zprior=FALSE, rng=0 )
## Post-processing simulation
post.ppm(out=out, slicedata=slicedata, ymaskdata=ymaskdata, vreg=2)
post.ppm(out=out, slicedata=slicedata, ymaskdata=ymaskdata, vreg=4)
## "bayesm" summaries
require("bayesm")
summary(out$betadraw)
summary(out$Deltadraw)
plot(out$Deltadraw)
summary(out$Vbetadraw)
##
## Random effects simulation using the SPM auditory dataset "swrfm*"
fbase <- "swrfm"
```

```

slice <- 21
slicedata <- read.fmrisslice(fbase=fbase, slice=slice, swap=FALSE )
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"/simultest3",fileext = ".sav", sep="")
out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=3,
  fsave=fsave, zprior=TRUE, rng=1)
post.ppm(out=out, slicedata=slicedata, ymaskdata=ymaskdata, vreg=2)

## End(Not run)

```

---

cudaMultireg.volume     *CUDA Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis on a fMRI NIFTI volume*

---

## Description

cudaMultireg.volume provides an interface to a CUDA implementation of a Bayesian multilevel model for the analysis of brain fMRI data. Data is processed on a slice-by-slice basis. Data volumes in gzipped NIFTI format are used.

## Usage

```

cudaMultireg.volume(fbase=NULL, R=2000, keep=5, nu.e=3,
  zprior=FALSE, rng=0, rg=c(NULL,NULL), swap=FALSE, savedir=tempdir())

```

## Arguments

**fbase**                    If fbase is left unspecified (default NULL), then user datasets need to be provided as input. Otherwise, fbase indicates the dataset prefix of one of the two demo fMRI datasets to use. Three data files are required as input. User specified data files must have the names generated by the FSL/FEAT pre-processing tool, namely  
‘filtered\_func\_data.nii.gz’, ‘mask.nii.gz’, and ‘design.mat’.  
‘filtered\_func\_data.nii.gz’ specifies the dataset to be analyzed,  
‘mask.nii.gz’ specifies the dataset to be used as mask.  
‘design.mat’ specifies the dataset to be used as design matrix.  
Typically, these datasets are obtained using the FSL/FEAT pre-processing tool, or other similar tool.  
In **cudaBayesreg**, versions 10+, read.fmrisslice uses the ‘design.mat’ format from FSL/FEAT.  
The prefix fbase applies to the demo data files provided in the complementary package **cudaBayesregData**:  
‘{fbase}\_filtered\_func.nii.gz’,  
‘{fbase}\_mask.nii.gz’, and  
‘{fbase}\_design.mat’.  
Two test datasets are included in the package: one with prefix ‘fmri’, the other with prefix ‘swrfm’. The prefix ‘swrfm’ is used in the random effects example. See also read.Zsegslice for user-defined segmented masks.

R	number of MCMC draws
keep	MCMC thinning parameter: keep every keepth draw (def: 5)
nu.e	d.f. parameter for regression error variance prior (def: 3)
zprior	Boolean {T,F}; default {F} - use just a mean for Z (see model description in <code>cudaMultireg.slice</code> ).
rng	integer {0,1,2}: type of RNG to use {0-Marsaglia Multicarry, 1-R. P. Brent xorgens, 2-Mersenne Twister MT19937-64}; (def. 0-Marsaglia Multicarry)
rg	rg=c(first, last): a vector containing the first and last numbers of the sequence of slices to be processed. If rg=c(NULL,NULL) (default), all slices in the volume are processed.
swap	logical variable (default = FALSE) for choosing the right/left data display convention consistent with FSLVIEW
savedir	Directory (def: <code>tempdir()</code> ) where the MCMC simulations for all slices are going to be saved.

## Details

The statistical model implemented in CUDA was specified as a Gibbs Sampler for hierarchical linear models with a normal prior. The main computational work is done in parallel on a CUDA capable GPU. Each thread is responsible for fitting a general linear model at each voxel. Data volumes are processed on a slice-by-slice basis, before reconstructing the processed volume, using `build.zstatvolume`. The statistical model is specified in `cudaMultireg.slice`. To run the examples, the data sets from the R-package **cudaBayesregData** are required.

## Author(s)

Adelino Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia, Portugal, <afs@fct.unl.pt>.

## References

- Adelino R. Ferreira da Silva (2011). “**cudaBayesreg**: Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis.” *Journal of Statistical Software*, **44**(4), 1–24. URL <http://www.jstatsoft.org/v44/i04/>.
- Adelino Ferreira da Silva (2011). **cudaBayesregData**: *Data sets for the examples used in the package cudaBayesreg*, R package version 0.3-10. URL <http://CRAN.R-project.org/package=cudaBayesregData>.
- Adelino Ferreira da Silva (2011). “A Bayesian Multilevel Model for fMRI Data Analysis.”, *Computer Methods and Programs in Biomedicine*, **102**(3), 238–252.
- Adelino Ferreira da Silva (2010). “**cudaBayesreg**: Bayesian Computation in CUDA.”, *The R Journal*, **2/2**, 48-55. URL [http://journal.r-project.org/archive/2010-2/RJournal\\_2010-2\\_Ferreira-da-Silva.pdf](http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Ferreira-da-Silva.pdf).
- Rossi, Allenby and McCulloch. *Bayesian Statistics and Marketing*, Chapter 3. URL <http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html>.

Davies, R.B. (1994). *Writing a matrix package in C++*. In OON-SKI'94: The second annual object-oriented numerics conference, pp 207-213. Rogue Wave Software, Corvallis. URL [http://www.robertnz.net/cpp\\_site.html](http://www.robertnz.net/cpp_site.html).

Richard. P. Brent. *Some long-period random number generators using shifts and xors*, Preprint: 2 July 2007.

Brandon Whitcher, Volker Schmid and Andrew Thornton (2011). **oro.nifti**: *Rigorous - NIFTI Input / Output*, R package version 0.2.5. URL <http://CRAN.R-project.org/package=oro.nifti>.

### See Also

[cudaMultireg.slice](#), [buildzstat.volume](#), [read.fmrislice](#), [read.Zsegslice](#), [premask](#), [pmeans.hcoef](#), [regpostsim](#), [plot.hcoef.post](#), [post.simul.hist](#), [post.ppm](#), [post.tseries](#), [post.randeff](#), [post.shrinkage.mean](#)

### Examples

```
## Not run:
## simulation using the SPM auditory dataset "swrfM*"
cudaMultireg.volume(fbase="swrfM", R=2000, rg=c(13,16), savedir=tempdir())
buildzstat.volume(fbase="swrfM", rg=c(13,16))
post.overlay(fbase="swrfM", vreg=2, rg=c(13,16), view="axial")
##
## Random effects simulation using the SPM auditory dataset "swrfM*"
cudaMultireg.volume(fbase="swrfM", R=2000, zprior=TRUE, rng=1,
  rg=c(17,21), savedir=tempdir())
buildzstat.volume(fbase="swrfM", rg=c(17,21))
post.overlay(fbase="swrfM", vreg=2, rg=c(17,21), view="axial")
##
## Simulation using the visual/auditory test dataset "fmri"
cudaMultireg.volume(fbase="fmri", R=2000, savedir=tempdir())
buildzstat.volume(fbase="fmri", vreg=2)
post.overlay(fbase="fmri", vreg=2, view="axial")
buildzstat.volume(fbase="fmri", vreg=4)
post.overlay(fbase="fmri", vreg=4, view="axial")

## End(Not run)
```

---

plot.bayesm.mat

*Plot Method for Arrays of MCMC Draws*

---

### Description

plot.bayesm.mat is an S3 method to plot sequence plots of MCMC draws and acfs. The columns in the array correspond to parameters and the rows to MCMC draws.

### Usage

```
## S3 method for class 'bayesm.mat'
plot(x, names, ...)
```

**Arguments**

x	An object of either S3 class, bayesm.mat, or S3 class, mcmc
names	optional character vector of names for coefficients
...	standard graphics parameters

**Details**

Typically, `plot.bayesm.mat` will be invoked by a call to the generic plot function as in `plot(object)` where `object` is of class `bayesm.mat`. This function is a simplified version of the equivalent function in **bayesm**. The original **bayesm::plot** function may be used instead in the **cudaBayesreg** context. See description of similar function in `bayesm::plot.bayesm.mat`.

**Author(s)**

Peter Rossi, Graduate School of Business, University of Chicago.

---

<code>plot.hcoef.post</code>	<i>Plot Method for Hierarchical Model Coefficients</i>
------------------------------	--

---

**Description**

`plot.hcoef.post` plots arrays of hierarchical coefficients.

**Usage**

```
plot.hcoef.post(x, spmname, spm, burnin=trunc(.1*R), nsamp=30, ...)
```

**Arguments**

x	'betadraw' object generated by the MCMC simulation
spmname	name associated with the thresholded voxels, e.g. "activated", "non-activated"
spm	threshold active, or non-active voxel, coefficients
burnin	n. of draws to burnin, def: .1*R
nsamp	number of random voxels to use in the plots (default: 30)
...	standard graphics parameters

**Details**

See description of similar function in `bayesm::plot.bayesm.hcoef`.

**See Also**

[cudaMultireg.slice](#), [pmeans.hcoef](#), [regpostsim](#), [post.simul.hist](#), [post.simul.betadraw](#)

**Examples**

```

## Not run:
## load simulation
fsave <- paste(tempdir(),"simultest1",fileext = ".sav", sep="")
load(file=fsave)
cat("loaded",fsave,"\n")
vreg <- 2
pmeans <- pmeans.hcoef(out$betadraw)
px <- regpostsim(pmeans, vreg=vreg)
spma <- px$spma # active voxels
spmnm <- px$spmnm # non-active voxels
plot(out$betadraw, spmname="activated", spm=spma)
plot(out$betadraw, spmname="non-activated", spm=spmnm)

## End(Not run)

```

---

pmeans.hcoef	<i>Posterior mean for each regression variable</i>
--------------	--

---

**Description**

pmeans.hcoef processes the MCMC simulation to evaluate the posterior mean of the regression variables.

**Usage**

```
pmeans.hcoef(x, burnin=trunc(.1*R))
```

**Arguments**

x	‘betadraw’ object generated by the MCMC simulation
burnin	n. of draws to burnin, def: .1*R

**Details**

Post-process MCMC simulation

**Value**

pmeans	Posterior Means of Coefficients
--------	---------------------------------

**See Also**

[cudaMultireg.slice](#), [read.fmrlice](#), [regpostsim](#), [post.simul.betadraw](#), [post.simul.hist](#)

## Examples

```
## Not run:
slicedata <- read.fmrisclice(fbase="fmri", slice=3, swap=FALSE)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"/simultest1",fileext = ".sav", sep="")
answ <- readline("Run MCMC simulation first ? ")
run <- FALSE
if (substr(answ, 1, 1) == "y") { run <- TRUE }
if(run) {
  out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=3,
    fsave=fsave, zprior=FALSE)
} else {
  load(file=fsave)
  cat("loaded",fsave,"\n")
}
pmeans <- pmeans.hcoef(out$betadraw)

## End(Not run)
```

---

post.overlay

*Rendering a Posterior Probability Map (PPM) volume*

---

## Description

post.overlay overlays a statistical PPM volume of voxel activations on the original fMRI volume to visualise medical imaging data.

## Usage

```
post.overlay(fbase=NULL, vreg=2, nu.e=3, rg=c(NULL,NULL),
  view="axial", savedir=tempdir())
```

## Arguments

fbase	If fbase is left unspecified (default NULL), then user datasets need to be provided as input. Otherwise, fbase indicates the dataset prefix of one of the two demo fMRI datasets to use. see read.fmrisclice for a detailed description.
vreg	regression variable to represent in PPM; default(vreg=2)
nu.e	d.f. parameter for regression error variance prior (def: 3)
rg	rg=c(first, last): a vector containing the first and last numbers of the sequence of slices to be processed. If rg=c(NULL,NULL) (default), all slices in the volume are processed.
view	choice among the three orthogonal views c("axial", "coronal", "sagittal") to use for the rendered image, (def: "axial").
savedir	Directory (def: 'tempdir()') where the (PPM) NIFTI volume built by buildzstat.volume is located.

**Author(s)**

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia,  
<afs@fct.unl.pt>.

**References**

Adelino Ferreira da Silva (2011). "A Bayesian Multilevel Model for fMRI Data Analysis.", *Computer Methods and Programs in Biomedicine*, **102**,(3), 238–252.

**See Also**

[cudaMultireg.volume](#), [buildzstat.volume](#), [read.fmrislice](#)

**Examples**

```
## Not run:
## Simulation using the visual/auditory test dataset "fmri"
cudaMultireg.volume(fbase="fmri", R=2000, savedir=tempdir())
buildzstat.volume(fbase="fmri", vreg=2)
post.overlay(fbase="fmri", vreg=2, view="axial")
buildzstat.volume(fbase="fmri", vreg=4)
post.overlay(fbase="fmri", vreg=4, view="axial")

## End(Not run)
```

---

post.ppm

*Posterior Probability Map (PPM) image*

---

**Description**

post.ppm computes the PPM image of voxel activations in a slice.

**Usage**

```
post.ppm(out, slicedata, ymaskdata, vreg=2, swap=FALSE, plot=TRUE,
         col=heat.colors(256))
```

**Arguments**

out	output of MCMC simulation
slicedata	list(slice=slice, niislicets=niislicets, mask=mask, dsgn=dsgn); input slice data used in simulation as returned by read.fmrislice
ymaskdata	list(yn = yn, kin = kin, nreg = nreg); masked and standardised slice data as returned by premask
vreg	regression variable to represent in PPM; default(vreg=2)
swap	image in radiological convention (default=FALSE)
plot	show ppm images (with overlay) ?: (default=TRUE)
col	a list of colors such as that generated by "heat.colors", "gray" or similar functions.

**Details**

Use the MCMC draws to estimate the Posterior Probability Map (PPM) image. The number of regression variables used in the simulation is equal to the number of columns specified in the design matrix, plus an intercept term; vreg=1 represents the intercept term in regression.

**Value**

a list containing

ppm	ppm image as matrix
nactive	n. of active voxels

**Author(s)**

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia, <afs@fct.unl.pt>.

**References**

Adelino Ferreira da Silva (2011). "A Bayesian Multilevel Model for fMRI Data Analysis.", *Computer Methods and Programs in Biomedicine*, **102**,(3), 238–252.

**See Also**

[cudaMultireg.slice](#), [regpostsim](#), [post.simul.hist](#), [post.tseries](#)

**Examples**

```
## Not run:
slicedata <- read.fmrlice(fbase="fmri", slice=3, swap=FALSE)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"/simultest1",fileext = ".sav", sep="")
answ <- readline("Run MCMC simulation first ? ")
run <- FALSE
if (substr(answ, 1, 1) == "y") { run <- TRUE }
if(run) {
out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5,
nu.e=3, zprior=FALSE)
} else {
load(file=fsave)
cat("loaded",fsave,"\n")
}
post.ppm(out=out, slicedata=slicedata, ymaskdata=ymaskdata, vreg=2)
post.ppm(out=out, slicedata=slicedata, ymaskdata=ymaskdata, vreg=4)

## End(Not run)
```

---

post.randeff	<i>Plots of the random effects distribution</i>
--------------	---

---

### Description

Plots draws of the random effects distribution, following the specification of cross-sectional units (group information) in the Z matrix of the statistical model.

### Usage

```
post.randeff(out, classnames=NULL, climits=TRUE)
```

### Arguments

<code>out</code>	output of MCMC simulation
<code>classnames</code>	default=NULL; concatenation of unit (class member) names used in the Z matrix specification. The argument may be a subvector of all unit names, but the ‘classnames’ given in the argument must match the order used in the Z matrix specification. If no class names are given (default) only the draws of the mean of the random effects distribution are plotted.
<code>climits</code>	logical variable (default = ‘TRUE’): if TRUE plots for the class draws use a common ‘ylim’ parameter.

### Details

The statistical model allows for the analysis of random effects through the specification of the Z matrix in the prior,

$$\beta_{i_j} \sim N(Z\Delta[i,], V_{\beta_{i_j}}).$$

The example included in the package (‘fbase="swrfm"’) defines a partition of the fMRI dataset in 3 classes, associated with 3 brain regions: CSF, gray matter and white matter (see examples).

### Author(s)

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia, <afs@fct.unl.pt>.

### References

Adelino R. Ferreira da Silva (2011). “**cudaBayesreg**: Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis.” *Journal of Statistical Software*, **44**(4), 1–24. URL <http://www.jstatsoft.org/v44/i04/>.

Adelino Ferreira da Silva (2011). “A Bayesian Multilevel Model for fMRI Data Analysis.”, *Computer Methods and Programs in Biomedicine*, **102**(3), 238–252.

Adelino Ferreira da Silva (2010). “**cudaBayesreg**: Bayesian Computation in CUDA.”, *The R Journal*, **2/2**, 48-55. URL [http://journal.r-project.org/archive/2010-2/RJournal\\_2010-2\\_Ferreira~da~Silva.pdf](http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Ferreira~da~Silva.pdf).

**See Also**

[cudaMultireg.slice](#), [read.Zsegslice](#), [read.fmrlice](#)

**Examples**

```
## Not run:
## Random effects simulation using the SPM auditory dataset "swrfM*"
fbase <- "swrfM"
slice <- 21
slicedata <- read.fmrlice(fbase=fbase, slice=slice)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"/simultest3",fileext = ".sav", sep="")
out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=3,
  fsave=fsave, zprior=TRUE, rng=1)
## show random effects for 3 classes
post.randeff(out, classnames=c("CSF", "GRY", "WHT"))

## End(Not run)
```

---

post.shrinkage.mean    *Computes shrinkage of fitted estimates over regressions*

---

**Description**

post.shrinkage.mean computes the mean fitted estimates as a function of the mean regression coefficient estimates over all regressions.

**Usage**

```
post.shrinkage.mean(out, X, vreg, plot=T)
```

**Arguments**

out	output of MCMC simulation
X	regression matrix used in the simulation
vreg	number of the regression coefficient
plot	{T,F} output plot (default=T)

**Details**

To assess the influence of the hyperparameter  $nu$  on the dispersion of the fitted estimates and regression coefficient estimates two plots are available in the package: one in terms of means values, the other in terms of maximum and minimum values. These plots help visualizing shrinkage by analyzing the influence of the hyperparameter  $nu$  on the estimates. Different shrinkage plots may be compared for simulations with different  $nu$  values.

**Value**

a list containing

yrecmean	mean of fitted values
beta	mean of estimated coefficients over all regressions

**Author(s)**

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia,  
<afs@fct.unl.pt>.

**See Also**

[cudaMultireg.slice](#)

**Examples**

```
## Not run:
slicedata <- read.fmrislice(fbase="fmri", slice=3, swap=FALSE)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"simultest1",fileext = ".sav", sep="")
nu1 <- 3
out1 <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=nu1,
  fsave=fsave1, zprior=FALSE, rng=1 )
fsave <- paste(tempdir(),"simultest2",fileext = ".sav", sep="")
nu2 <- slicedata$nobs
out2 <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=nu2,
  fsave=fsave2, zprior=FALSE, rng=1 )
vreg <- 2
x1 <- post.shrinkage.mean(out1, slicedata$X, vreg=vreg, plot=F)
x2 <- post.shrinkage.mean(out2, slicedata$X, vreg=vreg, plot=F)
par(mfrow=c(1,2), mar=c(4,4,1,1)+0.1)
xlim=range(c(x1$beta, x2$beta))
ylim=range(c(x1$yrecmean, x2$yrecmean))
plot(x1$beta, x1$yrecmean,type="p", pch="+", col="violet", ylim=ylim,
  xlim=xlim, xlab=expression(beta), ylab="y")
legend("topright", expression(paste(nu,"=3")), bg="seashell")
plot(x2$beta, x2$yrecmean,type="p", pch="+", col="blue", ylim=ylim,
  xlim=xlim, xlab=expression(beta), ylab="y")
legend("topright", expression(paste(nu,"=45")), bg="seashell")
par(mfrow=c(1,1))

## End(Not run)
```

---

post.shrinkage.minmax *Computes shrinkage of fitted estimates over regressions*

---

### Description

post.shrinkage.minmax computes the maximum and minimum fitted estimates, as a function of the mean regression coefficient estimates over all regressions.

### Usage

```
post.shrinkage.minmax(out, X, vreg, plot=T)
```

### Arguments

out	output of MCMC simulation
X	regression matrix used in the simulation
vreg	number of the regression coefficient
plot	{T,F} output plot (default=T)

### Details

The plot helps visualizing shrinkage by analyzing the influence of the hyperparameter  $nu$  on the dispersion of the fitted maximum and minimum estimates. Different shrinkage plots may be compared for simulations with different  $nu$  values.

### Value

a list containing

yrecmin	minimum values of fitted values
yrecmax	maximum values of fitted values
beta	mean of estimated coefficients over all regressions

### Author(s)

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia, <afs@fct.unl.pt>.

### See Also

[cudaMultireg.slice](#), [read.fmrlice](#)

**Examples**

```
## Not run:
slicedata <- read.fmrlice(fbase="fmri", slice=3, swap=FALSE)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"/simultest1",fileext = ".sav", sep="")
nu1 <- 3
out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=nu1,
  fsave=fsave1, zprior=FALSE, rng=1)
vreg <- 2
post.shrinkage.minmax(out, slicedata$X, vreg=vreg)

## End(Not run)
```

---

post.simul.betadraw    *Postprocessing of MCMC simulation*

---

**Description**

Postprocessing of MCMC simulation. Boxplots of posterior distributions for regressor coefficient `beta[vreg]` in two cases: estimates for 30 time series of random voxels in active cortex areas; estimates for 30 time series of random voxels in non-active cortex areas.

**Usage**

```
post.simul.betadraw(out, vreg = 2)
```

**Arguments**

<code>out</code>	List of output objects of MCMC simulation
<code>vreg</code>	regression variable to map; default 'vreg=2'

**Details**

Post-process analysis

**See Also**

[cudaMultireg.slice](#), [regpostsim](#), [post.ppm](#), [post.tseries](#)

**Examples**

```
## Not run:
slicedata <- read.fmrlice(fbase="fmri", slice=3, swap=FALSE)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"/simultest1",fileext = ".sav", sep="")
answ <- readline("Run MCMC simulation first ? ")
run <- FALSE
if (substr(answ, 1, 1) == "y") { run <- TRUE }
```

```

if(run) {
out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=3,
  fsave=fsave, zprior=FALSE)
} else {
load(file=fsave)
cat("loaded", fsave, "\n")
}
post.simul.betadraw(out=out, vreg=2)
post.simul.betadraw(out=out, vreg=4)

## End(Not run)

```

---

post.simul.hist

*Histogram of the posterior distribution of a regression coefficient*


---

### Description

Postprocessing of MCMC simulation. Histogram of the posterior distribution of regression coefficient beta[vreg].

### Usage

```
post.simul.hist(out, vreg = 2)
```

### Arguments

out	list of output objects of MCMC simulation
vreg	regression variable to map; default 'vreg=2'

### Details

Post-process analysis

### See Also

[cudaMultireg.slice](#), [regpostsim](#), [post.ppm](#), [post.tseries](#)

### Examples

```

## Not run:
## load MCMC simulation
fsave <- paste(tempdir(), "/simultest1", fileext = ".sav", sep="")
load(fsave)
post.simul.hist(out=out, vreg=2)
post.simul.hist(out=out, vreg=4)

## End(Not run)

```

---

post.tseries	<i>Show fitted time series of active voxel</i>
--------------	--

---

**Description**

post.tseries plots the fitted time series of a voxel estimated as active.

**Usage**

```
post.tseries(out, slicedata, ymaskdata, vreg=2)
```

**Arguments**

out	output of MCMC simulation
slicedata	list(slice=slice, niislicets=niislicets, mask=mask, dsgn=dsgn); input slicedata used in simulation as returned by read.fmrislice
ymaskdata	list(yn = yn, kin = kin, nreg = nreg); masked and standardised slice data as returned by premask
vreg	number of the active variable to visualize; default(vreg=2).

**Details**

Use the estimated regression coefficients to visualize the fitted time series in an active voxel.

**Author(s)**

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia,  
<afs@fct.unl.pt>.

**See Also**

[cudaMultireg.slice](#), [post.simul.hist](#), [post.simul.betadraw](#), [post.ppm](#)

**Examples**

```
## Not run:
## read data and load MCMC simulation
slicedata <- read.fmrislice(fbase="fmri", slice=3, swap=TRUE)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(), "/simultest1", fileext = ".sav", sep="")
load(fsave)
post.tseries(out=out, slicedata=slicedata, ymaskdata=ymaskdata, vreg=2)

## End(Not run)
```

premask

*Mask out voxels with constant time-series*

---

**Description**

premask applies a pre-defined mask to a fMRI slice in order to select regions of interest (ROIs) for processing

**Usage**

```
premask(slicedata)
```

**Arguments**

```
slicedata      list(slicedata).
```

**Details**

Mask out fMRI nifti data as read by `cudaBayesreg::read.fmrlice(slice)`. Pixels with constant time series are masked out.

**Value**

a list containing

<code>yn</code>	voxels values
<code>kin</code>	indices of voxels in mask
<code>nreg</code>	number of regressions

**Author(s)**

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia, <afs@fct.unl.pt>.

**References**

Adelino Ferreira da Silva (2011). **cudaBayesregData**: *Data sets for the examples used in the package cudaBayesreg*, R package version 0.3-10. URL <http://CRAN.R-project.org/package=cudaBayesregData>.

FSL/FEAT Analysis tool, FMRIB Software Library (FSL). URL [www.fmrib.ox.ac.uk/fsl](http://www.fmrib.ox.ac.uk/fsl).

**See Also**

[cudaMultireg.slice](#), [read.fmrlice](#)

## Examples

```
## Not run:
slicedata <- read.fmrlice(fbase="fmri", slice=3)
ymaskdata <- premask(slicedata)
print(str(ymaskdata))

## End(Not run)
```

---

read.fmrlice	<i>Read fMRI data</i>
--------------	-----------------------

---

## Description

read.fmrlice reads pre-filtered fMRI data, mask data, and the design matrix to be used in fMRI data processing.

## Usage

```
read.fmrlice(fbase=NULL, slice=NULL, swap=FALSE)
```

## Arguments

fbase	<p>If fbase is left unspecified (default NULL), then user datasets need to be provided as input. Otherwise, fbase indicates the dataset prefix of one of the two demo fMRI datasets to use. Three data files are required as input. User specified data files must have the names generated by the FSL/FEAT pre-processing tool, namely</p> <p>‘filtered_func_data.nii.gz’, ‘mask.nii.gz’, and ‘design.mat’.</p> <p>‘filtered_func_data.nii.gz’ specifies the dataset to be analyzed, ‘mask.nii.gz’ specifies the dataset to be used as mask. ‘design.mat’ specifies the dataset to be used as design matrix.</p> <p>Typically, these datasets are obtained using the FSL/FEAT pre-processing tool, or other similar tool.</p> <p>In <b>cudaBayesreg</b>, versions 10+, read.fmrlice uses the ‘design.mat’ format from FSL/FEAT.</p> <p>The prefix fbase applies to the demo data files provided in the complementary package <b>cudaBayesregData</b>:</p> <p>‘{fbase}_filtered_func.nii.gz’,</p> <p>‘{fbase}_mask.nii.gz’, and</p> <p>‘{fbase}_design.mat’.</p> <p>Two test datasets are included in the package: one with prefix ‘fmri’, the other with prefix ‘swrfm’. The prefix ‘swrfm’ is used in the random effects example. See also read.Zsegslice for user-defined segmented masks.</p>
slice	<p>the number of the slice to use. If a slice number is not specified a central slice from the provided dataset (mid-brain, in general) is assumed (default NULL).</p>
swap	<p>logical variable (default = FALSE) for choosing the right/left data display convention consistent with FSLVIEW.</p>

## Details

The FSL/FEAT Analysis tool may be used to generate the prefiltered fMRI data (niislicets), the mask (mask), and the design matrix (dsgn) required as data input. The FSL-design file *design.mat* is simply a ASCII textfile with the fields */NumWaves*, */NumPoints*, */PPheights*, and */Matrix*. Therefore, it may easily edited, if required, to prepare user specific design matrices without the FSL/FEAT tool. The package **oro.nifti** is required for reading gzipped NIFTI files.

## Value

a list containing

fbase	dataset prefix of the dataset used in the analysis
slice	slice number
niislicets	slice data at all timepoints
mask	slice mask
X	full design matrix
nvar	number of regression variables
nobs	number of observations
swap	relative orientation used in the data setup

## Author(s)

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia,  
<afs@fct.unl.pt>.

## References

Adelino Ferreira da Silva (2011). **cudaBayesregData**: *Data sets for the examples used in the package cudaBayesreg*. R package version 0.3-10. URL <http://CRAN.R-project.org/package=cudaBayesregData>.

FSL/FEAT Analysis tool, FMRIB Software Library (FSL). URL [www.fmrib.ox.ac.uk/fsl](http://www.fmrib.ox.ac.uk/fsl).

Brandon Whitcher, Volker Schmid and Andrew Thornton (2011). **oro.nifti**: *Rigorous - NIfTI Input / Output*, R package version 0.2.5. URL <http://CRAN.R-project.org/package=oro.nifti>.

## See Also

[cudaMultireg.slice](#) [read.Zsegslice](#) [premask](#)

## Examples

```
## Not run:
slicedata <- read.fmrlice(fbase="fmri", slice=3)
print(str(slicedata))

## End(Not run)
```

---

read.Zsegslice	<i>Read brain segmented data based on structural regions for CSF, gray, and white matter.</i>
----------------	---

---

### Description

read.Zsegslice builds the Z matrix of the statistical model, based on the brain segmented regions CSF/GRY/WHT for a given fMRI dataset.

### Usage

```
read.Zsegslice(slicedata, ymaskdata)
```

### Arguments

slicedata	list of data values returned by the call to read.fmrlice
ymaskdata	list of data values returned by the call to premask

### Details

The FSL tools may be used to obtain the segmented masks for brain parcellation in three main regions according to tissue type (CSF,GRY,WHT). If fbase has been left unspecified in reading slicedata, then three user specified segmented datasets in gzipped NIFTI format must be provided as input, with the names 'csf.nii.gz', 'gry.nii.gz', and 'wht.nii.gz'. Otherwise, fbase indicates the dataset prefix of one of the three segmented masks provided as a group effects example for 'swrfM\_filtered\_func\_data.nii.gz': 'swrfM\_csf.nii.gz', 'swrfM\_gry.nii.gz', and 'swrfM\_wht'. Only 'swrfM' segmented masks are provided in **cudaBayesregData**. The segmented masks included in the package were obtained by applying FAST to the structural high-resolution image 'sM00223\*', followed by FLIRT for low-resolution registration to 'fM00223\*'. The 'sM00223\*' and 'fM00223\*' datasets are available from the SPM site, and are described in chapter 28 of the SPM8 manual. The fMRI dataset 'swrfM\_filtered\_func\_data.nii.gz' is a filtered version of 'fM00223\*'.

### Value

Z	centered matrix specifying the characteristics of cross-sectional units (optional group information)
---	--

### Author(s)

A. Ferreira da Silva, Universidade Nova de Lisboa, Faculdade de Ciencias e Tecnologia, <afs@fct.unl.pt>.

## References

Adelino Ferreira da Silva (2011). **cudaBayesregData**: *Data sets for the examples used in the package cudaBayesreg*, R package version 0.3-10. URL <http://CRAN.R-project.org/package=cudaBayesregData>.

FSL/FEAT Analysis tool, FMRIB Software Library (FSL). URL [www.fmrib.ox.ac.uk/fsl](http://www.fmrib.ox.ac.uk/fsl).

John Ashburner et. al.. *SPM8 Manual*, Functional Imaging Laboratory, Institute of Neurology, UCL, London. URL <http://www.fil.ion.ucl.ac.uk/spm/>.

## See Also

[cudaMultireg.slice](#), [post.randeff](#), [premask](#), [post.ppm](#)

## Examples

```
## Not run:
fbase <- "swrfM"
slice <- 21
slicedata <- read.fmrlice(fbase=fbase, slice=slice, swap=FALSE )
ymaskdata <- premask(slicedata)
Z <- read.Zsegslice(slicedata, ymaskdata )
## Random effects simulation
fsave <- paste(tempdir(),"simultest2",fileext = ".sav", sep="")
out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=3,
  fsave=fsave, zprior=TRUE)
post.ppm(out=out, slicedata=slicedata, ymaskdata=ymaskdata, vreg=2)

## End(Not run)
```

---

regpostsim

*Estimation of voxel activations*

---

## Description

regpostsim estimates voxel activation and plots the posterior distribution of a regression coefficient.

## Usage

```
regpostsim(pmeans, vreg, plot=T)
```

## Arguments

pmeans	Posterior Means of Coefficients as processed by pmeans.hcoef()
vreg	regression variable to process
plot	plot the histogram, (default=T)

**Details**

Estimate the active and non-active voxels based on the highest posterior density (HPD) of the coefficients simulated by the multilevel method. Plot the histogram of the posterior distribution of regression coefficient ‘vreg’

**See Also**

[cudaMultireg.slice](#), [pmeans.hcoef](#), [plot.hcoef.post](#), [post.simul.hist](#), [post.simul.betadraw](#), [post.ppm](#), [post.tseries](#)

**Examples**

```
## Not run:
slicedata <- read.fmrlice(fbase="fmri", slice=3, swap=FALSE)
ymaskdata <- premask(slicedata)
fsave <- paste(tempdir(),"simultest1",fileext = ".sav", sep="")
answ <- readline("Run MCMC simulation first ? ")
run <- FALSE
if (substr(answ, 1, 1) == "y") { run <- TRUE }
if(run) {
  out <- cudaMultireg.slice(slicedata, ymaskdata, R=2000, keep=5, nu.e=3,
    fsave=fsave, zprior=FALSE)
} else {
  load(file=fsave)
  cat("loaded", fsave, "\n")
}
##
pmeans <- pmeans.hcoef(out$betadraw)
px <- regpostsim(pmeans, vreg=2)
pm2 <- pmeans[,vreg]
spma <- px$spma # active voxels
spmn <- px$spmn # non-active voxels

## End(Not run)
```

# Index

## \*Topic **IO**

read.fmrislice, 23  
read.Zsegslice, 25

## \*Topic **dplot**

post.overlay, 12  
post.ppm, 13  
post.randeff, 15  
post.shrinkage.mean, 16  
post.shrinkage.minmax, 18  
post.tseries, 21

## \*Topic **hplot**

plot.bayesm.mat, 9  
plot.hcoef.post, 10  
post.simul.betadraw, 19  
post.simul.hist, 20

## \*Topic **regression**

cudaMultireg.slice, 4  
cudaMultireg.volume, 7

## \*Topic **utilities**

buildzstat.volume, 2  
pmeans.hcoef, 11  
premask, 22  
regpostsim, 26

buildzstat.volume, 2, 3, 9, 13

cudaMultireg.slice, 3, 4, 9–11, 14, 16–22,  
24, 26, 27

cudaMultireg.volume, 7, 13

plot.bayesm.mat, 9

plot.hcoef.post, 6, 9, 10, 27

pmeans.hcoef, 6, 9, 10, 11, 27

post.overlay, 3, 12

post.ppm, 6, 9, 13, 19–21, 26, 27

post.randeff, 6, 9, 15, 26

post.shrinkage.mean, 6, 9, 16

post.shrinkage.minmax, 18

post.simul.betadraw, 10, 11, 19, 21, 27

post.simul.hist, 6, 9–11, 14, 20, 21, 27

post.tseries, 6, 9, 14, 19, 20, 21, 27

premask, 6, 9, 22, 24, 26

read.fmrislice, 3, 6, 9, 11, 13, 16, 18, 22, 23

read.Zsegslice, 6, 9, 16, 24, 25

regpostsim, 6, 9–11, 14, 19, 20, 26