# Package: polySegratioMM
# Version: 0.5-1

July 18, 2008

## R topics documented:

---

`calculateDIC` *Compute DIC for fitted mixture model*

---

### Description

Computes and returns the Deviance Information Critereon (DIC) as suggested by Celeaux et al (2006) as their DIC$_4$ for Bayesian mixture models

### Usage

```
calculateDIC(mcmc.mixture, model, priors, seg.ratios, chain=1, print.DIC=FALSE)
```

## Arguments

| | |
|---|---|
| `mcmc.mixture` | Object of type `segratioMCMC` produced by `coda` usually by using `readJags` |
| `model` | object of class `modelSegratioMM` specifying model parameters, ploidy etc |
| `priors` | Object of class `priorsSegratioMM` |
| `seg.ratios` | Object of class `segRatio` contains the segregation ratios for dominant markers and other information such as the number of dominant markers per individual |
| `chain` | Which chain to use when compute dosages (Default: 1) |
| `print.DIC` | Whether to print DIC |

## Value

A scalar DIC is returned

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## References

G Celeaux et. al. (2006) Deviance Information Criteria for Missing Data Models *Bayesian Analysis* **4** 23pp

D Spiegelhalter et. el. (2002) Bayesian measures of model complexity and fit *JRSS B* **64** 583–640

## See Also

`dosagesMCMC` `readJags`

## Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)

## compute segregation ratios
sr <-  segregationRatios(a1$markers)

## set up model, priors, inits etc and write files for JAGS
x <- setModel(3,8)
x2 <- setPriors(x)
dumpData(sr, x)
inits <- setInits(x,x2)
dumpInits(inits)
writeJagsFile(x, x2, stem="test")

## run JAGS
small <- setControl(x, burn.in=200, sample=500)
writeControlFile(small)
rj <- runJags(small)  ## just run it
print(rj)

## read mcmc chains and print DIC
xj <- readJags(rj)
```

```
print(calculateDIC(xj, x, x2, sr))
```

---

## `diagnosticsJagsMix` *MCMC diagnostics for polyploid segregation ratio mixture models*

---

### Description

Produce and/or plot various diagnostic measures from `coda` package for Bayesian mixture models for assessing marker dosage in autopolyploids

### Usage

```
diagnosticsJagsMix(mcmc.mixture, diagnostics = TRUE, plots = FALSE,
 index = -c( grep("T\[", varnames(mcmc.mixture$mcmc.list)),
             grep("b\[",varnames(mcmc.mixture$mcmc.list)) ),
 trace.plots = FALSE, auto.corrs = FALSE, density.plots = FALSE,
 xy.plots = FALSE, hpd.intervals = FALSE, hdp.prob = 0.95,
 return.results = FALSE)
```

### Arguments

| | |
|---|---|
| `mcmc.mixture` | Object of class `segratioMCMC` or `runJagsWrapper` after `JAGS` run produced by `coda` |
| `diagnostics` | if TRUE then print several `coda` dignostic tests |
| `plots` | if TRUE then produce several `coda` dignostic plots |
| `index` | index of parameters for disgnostic tests/plots (Default: mixture model (and random effects) parameters) |
| `trace.plots` | if TRUE plot mcmc traces (default: FALSE) |
| `auto.corrs` | if TRUE produce autocorrelations of mcmc's (default: FALSE) |
| `density.plots` | if TRUE plot parameter densities (default: FALSE) |
| `xy.plots` | if TRUE plot traces using 'lattice' (default: FALSE) |
| `hpd.intervals` | if TRUE print and return highest posterior density intervals for parameters specified by `index` |
| `hdp.prob` | probability for `hpd.intervals` |
| `return.results` | if TRUE return results as list |

### Value

If `return.results` is TRUE then a list is returned with components depending on various settings of arguments

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

mcmc autocorr.diag raftery.diag geweke.diag gelman.diag trellisplots

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)

## fit simple model in one hit

x.run <- runSegratioMM(sr, x, burn.in=200, sample=500)
print(x.run)
diagnosticsJagsMix(x.run)
diagnosticsJagsMix(x.run, plot=TRUE)
```

---

## DistributionPlotBinomial

*Distribution Plot*

---

### Description

Plots probability density function given the parameters. May be useful when investigating parameter choice for prior distributions.

### Usage

```
DistributionPlotBinomial(size = 200, prob = 0.5,
xlab = "Number of Successes", ylab = "Probability Mass", signif.digits = 3,
main = paste("Binomial Distribution: n =", size, "p =",
signif(prob, digits = signif.digits)))

DistributionPlotGamma(shape = 1, rate = 1, length = 100, xlab = "x",
ylab = "Density", main = bquote(paste("Gamma Distribution: ", alpha,
"=", .(signif(shape, digits = signif.digits)), ",", beta, "=",
.(signif(rate, digits = signif.digits)))), signif.digits = 3)

DistributionPlotNorm(mean = 0, sd = 1, length = 100, xlab = "x", ylab =
"Density", main = bquote(paste("Normal Distribution: ", mu, "=",
.(signif(mean, digits = signif.digits)), ",", sigma, "=", .(signif(sd,
digits = signif.digits)))), signif.digits = 3)
```

### Arguments

| | |
|---|---|
| `size` | number of trials (Binomial) |
| `prob` | probability of success (Binomial) |
| `shape` | shape parameter. Must be strictly positive. (Gamma) |
| `rate` | an alternative way to specify the scale (Gamma) |

| | |
|---|---|
| `mean` | mean (Normal) |
| `sd` | standard deviation (Normal) |
| `xlab` | x-axis label |
| `ylab` | y-axis label |
| `signif.digits` | number of significant digits for default `main` title |
| `main` | title for plot |
| `length` | Number of points to use for obtaining a smooth curve |

## Details

Based on functions in package `Rcmdr`

## Value

None.

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## See Also

`Rcmdr` `Binomial` `Normal` `GammaDist`

## Examples

```
## Binomial distribution
DistributionPlotBinomial()
DistributionPlotBinomial(size=20, prob=0.2)

## Gamma distribution
DistributionPlotGamma()

## Normal distribution
DistributionPlotNorm()
```

---

`dosagesJagsMix` *Compute dosages under specified Bayesian mixture model*

---

## Description

Computes and returns estimated dosages under specified model using posterior probabilities derived from mcmc chains by the proportion of samples in each dosage class.

## Usage

```
dosagesJagsMix(mcmc.mixture, jags.control, seg.ratio, chain = 1,
max.post.prob = TRUE, thresholds = c(0.5, 0.6, 0.7, 0.8, 0.9, 0.95,
0.99), print = FALSE, print.warning = TRUE, index.sample = 20)
```

## Arguments

| | |
|---|---|
| `mcmc.mixture` | Object of type `segratioMCMC` produced by `coda` usually by using `readJags` |
| `jags.control` | Object of class `jagsControl` for setting up `JAGS` command file |
| `seg.ratio` | Object of class `segRatio` contains the segregation ratios for dominant markers and other information such as the number of dominant markers per individual |
| `chain` | Which chain to use when compute dosages (Default: 1) |
| `max.post.prob` | Logical for producing dose allocations based on the maximum posterior probability (Default: `TRUE`) |
| `thresholds` | Numeric vector of thresholds for allocating dosages when the posterior probabilty to a particular dosage class is above the threshold |
| `print` | Logical indicating whether or not to print intermediate results (Default: `FALSE`) |
| `print.warning` | Logical to print warnings if there is more than one marker with the maximum posterior probability |
| `index.sample` | Numeric vector indicating which markers to print if `print` is `TRUE`. If `index.sample` is of length 1 then a random sample of size `index.sample` is selected |

## Value

An object of class `dosagesMCMC` is returned with components:

| | |
|---|---|
| `p.dosage` | Matrix of posterior probabilities of dosages for each marker dosage |
| `dosage` | Matrix of allocated dosages based on posterior probabilities. The columns correspond to different 'thresholds' and if requested, the last column is allocated on basis of `max.post` |
| `thresholds` | vector of cutoff probabilities for dosage class |
| `chain` | Chain used to compute dosages |
| `max.post` | maximum dosage posterior probabilties for each marker |
| `index.sample` | Numeric vector indicating which markers to print if `print` is `TRUE`. If `index.sample` is of length 1 then a random sample of size `index.sample` is selected |

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## See Also

`dosagesMCMC` `readJags`

## Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)

## compute segregation ratios
sr <-  segregationRatios(a1$markers)

## set up model, priors, inits etc and write files for JAGS
x <- setModel(3,8)
```

```
x2 <- setPriors(x)
dumpData(sr, x)
inits <- setInits(x,x2)
dumpInits(inits)
writeJagsFile(x, x2, stem="test")

## run JAGS
small <- setControl(x, burn.in=200, sample=500)
writeControlFile(small)
rj <- runJags(small)  ## just run it
print(rj)

## read mcmc chains and produce dosage allocations
xj <- readJags(rj)
dd <- dosagesJagsMix(xj, small, sr)
print(dd)
```

---

## `dumpData` *Dumps segregation ratio data to file for subsequent JAGS run*

---

### Description

Given segregation ratio data provided as an object of class `segRatio`, data are dumped in R format for use by `JAGS`

### Usage

```
dumpData(seg.ratio, model, stem = "test", fix.one = TRUE,
 data.file = paste(stem, "-data.R", sep = ""))
```

### Arguments

| | |
|---|---|
| `seg.ratio` | Object of class `segRatio` contains the segregation ratios for dominant markers and other information such as the number of dominant markers per individual |
| `model` | Object of class `modelSegratioMM` containing mixture model information |
| `stem` | File name stem for data file (default “test”) |
| `fix.one` | Logical to fix the dosage of the observation closest to the centre of each component on the logit scale. This can greatly assist with convergence (Default: `TRUE`) |
| `data.file` | Data file name which is automatically generated from `stem` if not specified |

### Value

None.

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

`segRatio` `dump`

## Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)

## compute segregation ratios
sr <-  segregationRatios(a1$markers)

## set up model for 3 components of autooctoploid
x <- setModel(3,8)

dumpData(sr, x)
```

---

## plotFitted *Plot observed segregation ratios and fitted and theoretical models*

---

### Description

Plots histogram of observed segregation ratios on logit scale along with scaled density of fitted components corresponding to dosage classes. Plots of expected theoretical distributions can be plotted with or without segregation ratio data.

### Usage

```
## S3 method for class 'runJagsWrapper':
plot(x, theoretical=FALSE, ...)

plotFitted(seg.ratios, summary.mixture, add.random.effect=TRUE,
  theoretical=FALSE, model=NULL, theory.col="red",
  xaxis=c("logit","raw"), ylim=NULL, NCLASS=NULL, n.seq=100,
  xlab="logit(Segregation Ratio)", ylab="Density", density.plot=FALSE,
  fitted.lwd=2, fitted.col="blue", bar.col="lightgreen", cex=1,
  warnings = FALSE, main=NULL, ...)

plotTheoretical(ploidy.level=8, seg.ratios=NULL, n.components=NULL,
  expected.segratio=NULL, proportions=c(0.65,0.2,0.1,0.03,0.01,0.01, 0, 0),
  n.individuals=200, xaxis=c("raw","logit"),
  type.parents=c("heterogeneous","homozygous"), xlim=c(0,1),
  NCLASS=NULL, xlab="Segregation Ratio", ylab="Density",
  density.plot=FALSE, fitted.lwd=2, fitted.col="blue", cex=1,
  warnings = TRUE, main=NULL, ...)
```

### Arguments

| | |
|---|---|
| `x` | object of class `runJagsWrapper` produced by using `runSegratioMM` to set up and fit mixture model |
| `seg.ratios` | segregation ratios as class `segRatio` |
| `summary.mixture` | mcmc summary data produce by `summary.segratioMCMC` |

| | |
|---|---|
| `add.random.effect` | add random variance component to fitted distribution plot if model includes a random effect (default: TRUE) |
| `theoretical` | whether to plot the expected theoretical distribution under the fitted model (default: FALSE) |
| `model` | object of class `modelSegratioMM` specifying model if plotting expected theoretical distribution |
| `theory.col` | colour for expected theoretical distribution (default: "red") |
| `ploidy.level` | the number of homologous chromosomes |
| `n.components` | number of components for mixture model |
| `expected.segratio` | may be specified or automatically calculated from ploidy level etc |
| `xaxis` | whether to plot on "logit" or "raw" scale. Defaults to "logit" if plotting segregation ratios or "raw" for theoretical distributions |
| `proportions` | for no. of markers in each component of theoretical distribution plot |
| `n.individuals` | for theoretical distribution plot - taken from segregation ratios if supplied |
| `type.parents` | "heterogeneous" if parental markers are 0,1 or "homogeneous" if parental markers are both 1 |
| `ylim` | c(lower,upper) yaxis limits for histogram of segregation ratios |
| `xlim` | c(lower,upper) xaxis limits for segregation ratios |
| `NCLASS` | number of classes for histogram (Default: 100) |
| `n.seq` | number of points to use for plotting fitted mixture |
| `xlab` | x-axis label |
| `ylab` | y-axis label |
| `density.plot` | whether to plot a smoothed density as well as segregation data and fitted and/or theoretical distributions (default: FALSE) |
| `main` | title for plot |
| `fitted.lwd` | width for fitted line |
| `fitted.col` | colour for fitted line |
| `bar.col` | colour for histogram |
| `cex` | character expansion for text (see `par`) |
| `warnings` | print warnings like number of components etc (Default: FALSE) |
| `...` | extra options for `plot` |

## Details

`plotFitted` plot histogram of observed segregation ratios on logit scale along with scaled density of fitted components corresponding to dosage classes using trellis

`plotTheoretical` plot expected distribution of autopolyploid dominant markers on probability (0,1) scale. Segregation ratios may also be plotted

`plot.runJagsWrapper` plots the fitted values of object of class `runJagsWrapper` which has been produced by using `runSegratioMM` to set up and fit mixture model

Note that since trellis graphics are employed, plots may need to be printed in order to see them

## Value

None.

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## See Also

`summary.mcmc` `mcmc` `segratioMCMC` `readJags` `diagnosticsJagsMix` `runSegratioMM`

## Examples

```
## simulate small autooctaploid data set
plotTheoretical(8, proportion=c(0.7,0.2,0.1),n.individuals=50)
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)

## fit simple model in one hit and summarise

x.run <- runSegratioMM(sr, x, burn.in=200, sample=500)
print(x.run)

## plot fitted model using 'plotFitted'
plotFitted(sr, x.run$summary)
a.plot <- plotFitted(sr, x.run$summary, density.plot=TRUE)
print(a.plot)
## or the easier way
plot(x.run, theoretical=TRUE)
```

---

`plot.segratioMCMC` *MCMC plots for segregation ratio mixture models*

---

## Description

Standard MCMC trace and density plots for specified mixure model parameters and posterior probability distributions for specified markers

## Usage

```
plot.segratioMCMC(x, ..., row.index = c(1:10), var.index = c(1:6),
marker.index = c(1:8))
```

## Arguments

| | |
|---|---|
| `x` | object of class `segratioMCMC` |
| `...` | extra options for printing |
| `row.index` | which rows to print (Default: first 10) |
| `var.index` | which mixture model variable to summarise (Default: all) |
| `marker.index` | which markers to summarise (Default: 1:8) |

### Value

None.

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

`dosagesMCMC` `readJags`

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)

## fit simple model in one hit and summarise

x.run <- runSegratioMM(sr, x, burn.in=200, sample=500)
plot(x.run$mcmc.mixture)
```

---

## `polySegratioMM-package` *Marker dosage for autoployploids by Bayesian mixture models*

---

### Description

These functions provide tools for estimating marker dosage for dominant markers in regular autopolyploids via Bayesian mixture model. Wrappers are provided for generating MCMC samples using the `JAGS` software. Convergence diagnostics and posterior distribution densities are provided by the `coda` package.

### Details

| | |
|---|---|
| Package: | polySegratioMM |
| Type: | Package |
| Version: | 0.4-1 |
| Date: | 2008-01-03 |
| License: | Copyright CSIRO and all rights reserved until further notice. Discussions needed. |

The simplest way to fit a model is to use `runSegratioMM`. Given segregation ratios and a ploidy level, a mixture model is constructed with default priors and initial values and `JAGS` run to produce an MCMC sample for statistical inference.

A standard model may be set up with `setModel` where two parameters are set, namely `ploidy.level` or the number of homologous chromosomes set either as a numeric or as a character string and also `n.components` or the number of components for mixture model (less than or equal to maximum number of possible dosages).

Vague or strong priors may be constructed automatically using `setPriors`. Plots of standard conjugate distributions may be obtained using `DistributionPlotBinomial` `DistributionPlotGamma` and `DistributionPlotNorm`.

If necessary, other operations like setting up initial values or the control files for `JAGS` may be set using `setInits` `setControl` `dumpData` `dumpInits` `writeControlFile` `writeJagsFile`. Once the `BUGS` files and `JAGS` control files are set up then `JAGS` may be run using `runJags` and results read using `readJags`.

Convergence diagnostics may be carried out using `\coda` or the convenience wrapper `diagnosticsJagsMix`.

Dose allocation can be carried out using `dosagesJagsMix`.

Plots may be produced and objects printed and summarised using standard `print` and `plot` methods. Plots of theoretical binomial distributions with different ploidy levels and sample sizes may be obtained with `plotFitted`. In addition, `plotFitted` produces a lattice plot of the observed segregation ratios and fitted mixture model on the logit scale.

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## References

J B S Haldane (1930) Theoretical genetics of autopolyploids. *Journal of genetics* **22** 359–372

Ripol, M I et al (1999) Statistical aspects of genetic mapping in autopolyploids. *Gene* **235** 31–41

JAGS http://www-fis.iarc.fr/ martyn/software/jags/ and http://streaming.stat.iastate.edu/wiki/index.php/JAGS_Guide

## Examples

```
## simulate small autooctaploid data set of 100 markers for 50 individuals
## with %70 Single, %20 Double and %10 Triple Dose markers
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=75)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)  # autooctapolid mode with 3 components

## fit simple model in one hit with default priors, inits etc
## warning: this is too small an MCMC sample so should give inaccurate
## answers but it could still take quite a while
x.run <- runSegratioMM(sr, x, burn.in=200, sample=500)
print(x.run)

## plot observed segregation ratios, fitted model and expected distribution
plot(x.run, theoretical=TRUE)
```

## `print.dosagesMCMC` *Doses from Bayesian mixture model*

### Description

Prints objects of S3 class `dosagesMCMC` or `segratioMCMC`

### Usage

```
## S3 method for class 'dosagesMCMC':
print(x, ..., index.sample = 20)

## S3 method for class 'segratioMCMC':
print(x, ..., row.index = c(1:10), var.index = c(1:6), marker.index
 = c(1:8), chain = 1)
```

### Arguments

| | |
|---|---|
| `x` | object of class `dosagesMCMC` or `segratioMCMC` |
| `...` | extra options for printing |
| `index.sample` | which markers to print (Default: 20 markers at random) |
| `row.index` | which rows to print (Default: first 10) |
| `var.index` | which mixture model variable to summarise (Default: all) |
| `marker.index` | which markers to summarise (Default: 1:8) |
| `chain` | which chain to print (Default: 1) |

### Value

None.

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

`dosagesMCMC` `readJags`

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)

## fit simple model in one hit

x.run <- runSegratioMM(sr, x, burn.in=200, sample=500)

print(x.run$doses)
```

## `print.runJags` *Running JAGS*

### Description

Print details and timing of `JAGS` run and summaries of results

### Usage

```
## S3 method for class 'runJags':
print(x, ...)
## S3 method for class 'runJagsWrapper':
print(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | Objects of class `runJags` or `runJagsWrapper` |
| `...` | extra printing options |

### Details

`print.runJags` can be employed when `runJags` is called directly and reports timimgs and dates while `print.runJagsWrapper` provides summary statistics when `runSegratioMM` is used.

### Value

None.

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

`runJags` `runSegratioMM`

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)

## fit simple model in one hit

x.run <- runSegratioMM(sr, x, burn.in=200, sample=500)
print(x.run)
```

## `readJags` *Read MCMC sample(s) from a JAGS run*

### Description

wrapper to `read.openbugs` which returns object of class `mcmc.list` and so can be used to specify the start and end iterations for the MCMC sample(s) and also specify thinning

### Usage

```
readJags(run.jags, quiet = TRUE, ...)
```

### Arguments

| | |
|---|---|
| `run.jags` | object of class `runJAGS` produced by running JAGS |
| `quiet` | logical to return program output (Default: TRUE) |
| `...` | other options for `read.openbugs` |

### Value

Returns object of class `segratioMCMC` with components

| | |
|---|---|
| `run.jags` | object of class `runJAGS` produced by running JAGS |
| `mcmc.list` | object of class `mcmc.list` containing the MCMC sample(s) |

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

`mcmc.list` `setPriors` `setInits` `expected.segRatio` `segRatio` `setControl` `dumpData` `dumpInits` or for an easier way to run a segregation ratio mixture model see `runSegratioMM`

### Examples

```
library(polySegratio)

## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)
x2 <- setPriors(x)
cat(x$bugs.code,x2$bugs.code,sep="\n")

x3 <- setModel(3,8, random.effect = TRUE)
x4 <- setPriors(x3, type="strong")

dumpData(sr, x3)
inits <- setInits(x,x2)
dumpInits(inits)
```

```
##x.priors <- setPriors(x, "vague")
writeJagsFile(x, x2, stem="test")

small <- setControl(x, burn.in=20, sample=50)
writeControlFile(small)
rj <- runJags(small)  ## just run it

xj <- readJags(rj)
print(xj)
```

---

## runJags — *Run JAGS to create MCMC sample for segregation ratio mixture model*

---

### Description

Runs external program `JAGS` and returns MCMC list for processing by `coda`.

### Usage

```
runJags(jags.control, jags = "jags", quiet = FALSE,
 cmd.file = paste(jags.control$stem, ".cmd", sep = ""), timing = TRUE)
```

### Arguments

| | |
|---|---|
| `jags.control` | Object of class `jagsControl` containing MCMC burn in, sample and thinning as well as relavant files for BUGS commands, inits and data |
| `jags` | Name of `JAGS` program assumed to be in PATH. However, jags may explicitly set here to include the full path name |
| `quiet` | Locial to return program output (Default: FALSE) |
| `cmd.file` | `JAGS` .cmd commad file (Default: deduced from `jags.control`) |
| `timing` | Logical to return timing information such as date started and ended and elapsed user and system time |

### Value

Returns object of class `runJAGS` with components

| | |
|---|---|
| `jags.control` | Object of class `jagsControl` |
| `exit` | integer indicating return error (0 if no errors) |
| `cmd.file` | `JAGS` command file |
| `start.time` | time JAGS run started |
| `end.time` | time JAGS run finished |
| `elapsed.time` | elapsed user and system time |
| `call` | function call |

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## See Also

setPriors setInits expected.segRatio segRatio setControl dumpData dumpInits
or for an easier way to run a segregation ratio mixture model see runSegratioMM

## Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
sr <-  segregationRatios(a1$markers)

## set up model with 3 components
x <- setModel(3,8)
x2 <- setPriors(x)
dumpData(sr, x)
inits <- setInits(x,x2)
dumpInits(inits)
##x.priors <- setPriors(x, "vague")
writeJagsFile(x, x2, stem="test")

small <- setControl(x, burn.in=20, sample=50)
writeControlFile(small)
rj <- runJags(small)  ## just run it
print(rj)
```

---

`runSegratioMM` *Run a Bayesian mixture model for marker dosage with minimal effort*

---

## Description

Given segregation ratios and a ploidy level, a mixture model is constructed with default priors and initial values and `JAGS` run to produce an MCMC sample for statistical inference. Returns an object of S3 class `runJagsWrapper`

## Usage

```
runSegratioMM(seg.ratios, model, priors = setPriors(model),
 inits = setInits(model, priors), jags.control =
 setControl(model, stem, burn.in = burn.in, sample = sample, thin = thin),
 burn.in = 2000, sample = 5000, thin = 1, stem = "test", fix.one = TRUE,
 print = TRUE, plots = TRUE, print.diagnostics = TRUE,
 plot.diagnostics = TRUE, run.diagnostics.later=FALSE )
```

## Arguments

| | |
|---|---|
| `seg.ratios` | Object of class `segRatio` contains the segregation ratios for dominant markers and other information such as the number of dominant markers per individual |
| `model` | object of class `modelSegratioMM` specifying model parameters, ploidy etc |
| `priors` | object of class `priorsSegratioMM` indicating priors that are “vague”, “strong” or “specified” |
| `inits` | A list of initial values usually produced by `setInits` |

| | |
|---|---|
| `jags.control` | Object of class `jagsControl` containing MCMC burn in, sample and thinning as well as relavant files for BUGS commands, inits and data |
| `burn.in` | size of MCMC burn in (Default: 2000) |
| `sample` | size of MCMC sample (default: 5000) |
| `thin` | thinning interval between consecutive observations (default: 1 or no thinning) |
| `stem` | text to be used as part of `JAGS` .cmd file name |
| `fix.one` | Logical to fix the dosage of the observation closest to the centre of each component on the logit scale. This can greatly assist with convergence (Default: `TRUE`) |
| `print` | logical for printing monitoring and summary information (default: TRUE) |
| `plots` | logical to plotting MCMC posterior distributions (default: TRUE) |
| `print.diagnostics` | logical for printing disagnostic statistics (default: TRUE) |
| `plot.diagnostics` | logical for diagnostic plots (default: TRUE) |
| `run.diagnostics.later` | should diagnostics be run later which may help if there are convergence problems (Default: FALSE) |

## Value

Returns object of class `runJagsWrapper` with components

| | |
|---|---|
| `seg.ratios` | Object of class `segRatio` contains the segregation ratios for dominant markers |
| `model` | object of class `modelSegratioMM` specifying model parameters, ploidy etc |
| `priors` | Object of class `priorsSegratioMM` specifying prior distributions |
| `inits` | A list of initial values usually produced by `setInits` |
| `jags.control` | Object of class `jagsControl` containing MCMC burn in, sample and thinning as well as relavant files for BUGS commands, inits and data |
| `stem` | text to be used as part of `JAGS` .cmd file name and other files |
| `fix.one` | Logical to fix the dosage of the observation closest to the centre of each component on the logit scale. This can greatly assist with convergence (Default: `TRUE`) |
| `run.jags` | object of class `runJAGS` produced by running JAGS |
| `mcmc.mixture` | Object of type `segratioMCMC` produced by `coda` usually by using `readJags` |
| `diagnostics` | list containing various diagnostic summaries and statistics produced by `coda` |
| `summary` | summaries of posterior distributions of model parameters |
| `doses` | object of class `dosagesMCMC` containing posterior probabilities of dosages for each marker dosage and allocated dosages |
| `DIC` | Deviance Information Critereon |

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

setPriors setInits expected.segRatio segRatio setControl dumpData dumpInits and diagnosticsJagsMix

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)

## fit simple model in one hit

x.run <- runSegratioMM(sr, x, burn.in=200, sample=500)
print(x.run)
```

---

## setControl *Set up controls for a JAGS segregation ratio model run*

---

### Description

Sets up directives for running JAGS which are subsequently put into a .cmd file. MCMC attributes such as the size of burn in, length of MCMC and thinning may be specified

### Usage

```
setControl(model, stem = "test", burn.in = 2000, sample = 5000, thin = 1,
 bugs.file = paste(stem, ".bug", sep = ""),
 data.file = paste(stem, "-data.R", sep = ""),
 inits.file = paste(stem, "-inits.R", sep = ""),
 monitor.var = model$monitor.var, seed=1)
```

### Arguments

| | |
|---|---|
| model | object of class modelSegratioMM specifying model parameters, ploidy etc |
| stem | text to be used as part of JAGS .cmd file name |
| burn.in | size of MCMC burn in (Default: 2000) |
| sample | size of MCMC sample (default: 5000) |
| thin | thinning interval between consecutive observations. Thinning may be a scalar or specified for each variable set by specifying a vector (default: 1 or no thinning) |
| bugs.file | name of .bug file |
| data.file | name of R data file |
| inits.file | name of R inits file |
| monitor.var | which variables to be monitored (Default: as per model) |
| seed | seed for JAGS run for Windows only (for unix set seed in setInits) |

### Value

Returns an object of class `jagsControl` which is a list with components

| | |
|---|---|
| `jags.code` | Text containing control statements for `JAGS` .cmd file |
| `model` | object of class `modelSegratioMM` specifying model parameters, ploidy etc |
| `stem` | text to be used as part of `JAGS` .cmd file name |
| `burn.in` | size of MCMC burn in (Default: 2000) |
| `sample` | size of MCMC sample (default: 5000) |
| `thin` | thinning interval between consecutive observations |
| `bugs.file` | name of .bug file |
| `data.file` | name of R data file |
| `inits.file` | name of R inits file |
| `monitor.var` | which variables to be monitored |
| `call` | function call |

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

`setModel` `setInits` `expected.segRatio` `segRatio` `setControl` `dumpData` `dumpInits`
or for an easier way to run a segregation ratio mixture model see `runSegratioMM`

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)

## set up model with 3 components
x <- setModel(3,8)
x2 <- setPriors(x)

jc <- setControl(x)
print(jc)
```

---

## setInits — *Set up and dump initial values given the model and prior*

---

### Description

Given a model of class `modelSegratioMM` and priors of class `priorsSegratioMM`, initial values are computed using approximate expected values by `setInits` and then written to file by `dumpInits`

### Usage

```
setInits(model, priors, seed = 1)

dumpInits(inits, stem = "test", inits.file = paste(stem, "-inits.R",
  sep = ""))
```

## Arguments

| | |
|---|---|
| `model` | Object of class `modelSegratioMM` providing model attributes like the number of components and ploidy level |
| `priors` | Object of class `priorsSegratioMM` |
| `seed` | Seed to be used for `JAGS` runs. If a number of chains are to be run a vector of starting values may be specified. However, see note below. |
| `inits` | A list of initial values usually produced by `setInits` |
| `stem` | File name stem for inits file (default “test”) |
| `inits.file` | Inits file name which is automatically generated from `stem` if not specified |

## Value

Returns a list with the following initial values:

| | |
|---|---|
| `mu` | Mean of dosage classes on logit scale: usually c(0,NA,NA,. . . ,NA) |
| `P` | Initial value for proportion in each dosage class |
| `tau` | Precision of means which depends on whether priors are strong or weak |
| `theta` | Differences in means (for parameterisation employed for better convergence) |
| `seed` | Sets seed for each MCMC chain (Default:1) |
| `taub` | If the model contains a random effect then sets initial value of precision of random effect *b* which is normally distributed with mean 0 and precision `taub` |

## Note

*Warning*: If a number of chains are to be produced then several seeds may be specified. Currently, this is largely untested and so it is highly unlikely that this will actually work for all functions in this package.

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## See Also

`setModel` `setPriors` `setControl` `dumpInits`

## Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)

## set up model, priors, inits etc and write files for JAGS
x <- setModel(3,8)
x2 <- setPriors(x)
inits <- setInits(x,x2)
dumpInits(inits)
```

## setModel *Set characteristics of the Bayesian mixture model for dosages*

### Description

Used to automatically set up Bayesian finite mixture models for dosage allocation of dominant markers in autopolyploids given the number of components and ploidy level

### Usage

```
setModel(n.components, ploidy.level, random.effect = FALSE, seg.ratios =NULL,
 ploidy.name = NULL, equal.variances=TRUE,
 type.parents = c("heterogeneous", "homozygous"))
```

### Arguments

| | |
|---|---|
| `n.components` | number of components for mixture model (less than or equal to maximum number of possible dosages) |
| `ploidy.level` | the number of homologous chromosomes, either as numeric or as a character string |
| `random.effect` | Logical indicating whether model contains random effect (Default: `FALSE`) |
| `seg.ratios` | segregation proportions for each marker provided as S3 class `segRatio` |
| `ploidy.name` | Can overide ploidy name here or allow it to be determined from `ploidy.level` |
| `equal.variances` | Logical indicating whether model contains separate or common variances for each component (Default: `TRUE`) |
| `type.parents` | "heterogeneous" if parental markers are 0,1 or "homogeneous" if parental markers are both 1 |

### Value

Returns object of class `modelSegratioMM` with components

| | |
|---|---|
| `bugs.code` | text to be used by `JAGS` in the .bug file but without statements pertaining to priors |
| `n.components` | number of components for mixture model |
| `monitor.var` | names of variables to be monitored in `JAGS` run |
| `ploidy.level` | ploidy level |
| `random.effect` | Logical indicating whether model contains random effect (Default: `FALSE`) |
| `equal.variances` | Logical indicating equal or separate variances for each component |
| `E.segRatio` | Expected segregation ratios |
| `type.parents` | "heterogeneous" if parental markers are 0,1 or "homogeneous" if parental markers are both 1 |
| `call` | function call |

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

setPriors setInits expected.segRatio segRatio setControl dumpData dumpInits
or for an easier way to run a segregation ratio mixture model see runSegratioMM

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)

## set up model with 3 components
x <- setModel(3,8)
print(x)
```

---

## setPriors — *Set prior distributions for parameters of Bayesian mixture model for dosages*

---

### Description

May be used to automatically set up vague or strong priors or explicitly set them for Bayesian finite mixture model specified as an object of class `modelSegratioMM` using `setModel`

### Usage

```
setPriors(model, type.prior = c("strong",
                "vague", "strong.tau","strong.s", "specified"),
mean.vague = 0.1, prec.vague = 0.1, A.vague = 0.1, B.vague = 0.1,
prec.strong=400, n.individuals=200, reffect.A = 44, reffect.B = 0.8,
M.sd = 0.025, STRONG.PREC=c(0.025, 0.975), UPPER = 0.995,  PREC.INT=0.2,
params = NULL, segRatio = NULL)
```

### Arguments

| | |
|---|---|
| `model` | object of class `modelSegratioMM` specifying model parameters, ploidy etc |
| `type.prior` | The type of prior required being one of “strong”, “vague”, “strong.tau” “strong.s” or “specified”. The first four prior types will automatically set prior distributions whereas for the last, namely “specified”, the prior distribution parameters must be set explicitly. Note that strong priors get progressively stronger from “strong” to “strong.s” |
| `mean.vague` | The mean of Normal priors for a “vague” prior |
| `prec.vague` | The precision of Normal priors for a “vague” prior |
| `A.vague` | The shape parameter of the Gamma prior for the precision parameters for a “vague” prior |
| `B.vague` | The rate (scale) parameter of the Gamma prior for the precision parameters for a “vague” prior |

| | |
|---|---|
| `prec.strong` | Precision for Normal mean parameters when `type.prior` is “strong”. Note that on logit scale default is equivalent to having a 95%CI as +/- 0.1 |
| `n.individuals` | Used for Binomial calculations to set prior precision parameters when `type.prior` is “strong”. |
| `reffect.A` | The shape parameter of the Gamma prior for the precision parameter of the random.effect for a “vague” prior |
| `reffect.B` | The rate (scale) parameter of the Gamma prior for the precision parameter of the random.effect for a “vague” prior |
| `M.sd` | Approximate standard deviation for the mean segregation ratios on raw probability scale - this is set to 0.025 which would give an approximate 95% interval of 0.1 for the segregation ratio |
| `UPPER` | Cutoff for guessing parameters on logit scale noting that logit(1) is undefined |
| `STRONG.PREC` | Interval on raw probabilty scale used to set strong priors on the the precision distribution parameters of the segregation ratios by using a 95% interval on the theoretical distribution and equating this on the logit scale (Default: c(0.025, 0.975)) |
| `PREC.INT` | Multiplier or setting prior for precision on logit scale corresponding to approx confidence region being precision*(1 - PREC.INT, 1 + PREC.INT) Default:0.2 |
| `params` | if `type.prior` is “specified” then a list of priors parameters must be set containing components M for means, A and B for gamma prior parameters and if the model contains a random.effect then reffect.A, and reffect.B for the gamma prior for the precision of random effect `taub`. Note that the lengths of M, prec, A and B should be equal to the number of components |
| `segRatio` | If specified, this value overides the automatically generated value which is set as the expected segregation ratio given the ploidy level |

## Value

Returns an object of class `priorsSegratioMM` which is a list with components

| | |
|---|---|
| `type` | Type of prior: one of “vague”, “strong” or “specified” |
| `bugs.code` | Text containing prior statements for `BUGS` file |
| `random.effect` | Logical indicating whether model contains random effect (Default: `FALSE`) |
| `equal.variances` | Logical indicating equal or separate variances for each component |
| `params` | List containing Normal means on logit scale `logit.means`, precision on logit scale `logit.prec`, and Gamma parameters `A` and `B` and finally `reffect.A` and `reffect.B` if the model contains a random effect |
| `call` | function call |

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## See Also

`setModel` `setInits` `expected.segRatio` `segRatio` `setControl` `dumpData` `dumpInits` or for an easier way to run a segregation ratio mixture model see `runSegratioMM`

## Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)

## set up model with 3 components
x <- setModel(3,8)
x2 <- setPriors(x)
print(x2)

x2b <- setPriors(x, "strong")
print(x2b)
```

---

`summary.segratioMCMC` *Summary statistics for an segratioMCMC object*

---

## Description

Wrapper for `summary.mcmc` processing only mixture model parameters although markers may also easily be summarised. The mean, standard deviation, naive standard error of the mean (ignoring autocorrelation of the chain) and time-series standard error based on an estimate of the spectral density at 0. For details see `summary.mcmc`

## Usage

```
## S3 method for class 'segratioMCMC':
summary(object, ..., row.index = c(1:10),
 var.index = NULL,
 marker.index = c(1:8))
```

## Arguments

| | |
|---|---|
| `object` | object of class `segratioMCMC` |
| `...` | extra options for `summary.mcmc` |
| `row.index` | which rows to print (Default: first 10) |
| `var.index` | which mixture model variable to summarise (Default: all) |
| `marker.index` | which markers to summarise (Default: 1:8) |

## Value

An object of class `summarySegratioMCMC` is returned which contains summary statistics for parameters and some markers. For details see `summary.mcmc`

## Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

## See Also

`summary.mcmc` `mcmc` `segratioMCMC` `readJags` `diagnosticsJagsMix`

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
##print(a1)
sr <-  segregationRatios(a1$markers)
x <- setModel(3,8)

## fit simple model in one hit and summarise

x.run <- runSegratioMM(sr, x, burn.in=200, sample=500)
print(summary(x.run$mcmc.mixture))
print(summary(x.run$mcmc.mixture, var.index=c(1:3), marker.index=c(1:4)))
```

---

## `writeControlFile` *Write JAGS .cmd file for running JAGS*

---

### Description

Write JAGS .cmd file to disk

### Usage

```
writeControlFile(jags.control,
     file = paste(jags.control$stem, ".cmd", sep = ""))
```

### Arguments

| | |
|---|---|
| `jags.control` | Object of class `jagsControl` containing MCMC burn in, sample and thinning as well as relavant files for BUGS commands, inits and data |
| `file` | JAGS .cmd file name |

### Value

None.

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

`setControl` `runJags`

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)
sr <-  segregationRatios(a1$markers)

## set up model with 3 components
x <- setModel(3,8)
x2 <- setPriors(x)
```

```
dumpData(sr, x)
inits <- setInits(x,x2)
dumpInits(inits)
##x.priors <- setPriors(x, "vague")
writeJagsFile(x, x2, stem="test")

small <- setControl(x, burn.in=20, sample=50)
writeControlFile(small)
```

---

## `writeJagsFile` *Writes BUGS file for processing by JAGS*

---

### Description

Given the model and priors a file is written to disk for subsequent `JAGS` run. BUGS code contained in the model and priors objects is combined and alterered if necessary

### Usage

```
writeJagsFile(model, priors, stem = "test")
```

### Arguments

| | |
|---|---|
| `model` | object of class `modelSegratioMM` specifying model parameters, ploidy etc |
| `priors` | Object of class `priorsSegratioMM` specifying priors |
| `stem` | File name stem for BUGS file (default “test”) |

### Value

None.

### Author(s)

Peter Baker ⟨Peter.Baker@csiro.au⟩

### See Also

`segRatio` `dump`

### Examples

```
## simulate small autooctaploid data set
a1 <- sim.autoMarkers(8,c(0.7,0.2,0.1),n.markers=100,n.individuals=50)

## compute segregation ratios
sr <-  segregationRatios(a1$markers)

## set up model for 3 components of autooctoploid
x <- setModel(3,8)
x2 <- setPriors(x)

dumpData(sr, x)
inits <- setInits(x,x2)
```

```
dumpInits(inits)
##x.priors <- setPriors(x, "vague")
writeJagsFile(x, x2, stem="test")
```

# Index