Package 'neuromplex'

September 24, 2018

Version 0.0-1

Date 2018-09-23

Title Neural Multiplexing Analysis

Author Surya Tokdar <surya.tokdar@duke.edu>

Maintainer Surya Tokdar <surya.tokdar@duke.edu>

Depends R (>= 2.6), stats, graphics, grDevices, BayesLogit, weights

Description Statistical methods for whole-trial and time-
domain analysis of single cell neural response to multiple stimuli presented simultaneously.

License GPL-2

NeedsCompilation no

R topics documented:

`bin.counter` *Bin Counting*

Description

Fast bin counts of spike times

Usage

```
bin.counter(x, b)
```

Arguments

x	spike times
b	break points defining time bins. Must be an ordered vector with no duplications. Allowed to not cover the entire span of spike times

Value

Returns a vector giving the bin counts.

Examples

```
## generate 20 AB trials, roughl half with flat weight curves
## with a constant intensity either in (0,.1) or in (0.9, 1)
## (equally likely). The remaining curves are sinusoidal
## that snake between 0.1 and 0.9 with a period randomly
## drawn between 500 and 1500

synth.data <- synthesis.dapp(ntrials = c(15, 20, 20), pr.flat = 1,
                             intervals = list(c(0,.1), c(.45,.55), c(.9,1)),
                             wts = c(1/3, 1/3, 1/3), span = c(.1,.9),
                             period = c(500, 1500))

spike.counts <- list()
breaks <- seq(0, 1e3, 25)
spike.counts$Acounts <- sapply(synth.data$spiketimes$A, bin.counter, b = breaks)
spike.counts$Bcounts <- sapply(synth.data$spiketimes$B, bin.counter, b = breaks)
spike.counts$ABcounts <- sapply(synth.data$spiketimes$AB, bin.counter, b = breaks)
```

---

dapp *Dynamic Admixture of Poisson Process*

Description

Fits the DAPP model to binned spiking data

Usage

```
dapp(spike.counts, lengthScale = NULL, lsPrior = NULL,
     hyper = list(prec = c(1,1), sig0 = 1.87), burnIn = 1e3,
     nsamp = 1e3, thin = 4, plot = FALSE, verbose = TRUE,
     remove.zeros = FALSE)
```

Arguments

spike.counts	A list with the following items. 'Acounts': binned spike counts under condition A presented as a matrix. Rows are bins, columns are replicates (trials). 'Bcount': binned spike counts under condition B. 'ABcounts': binned spike counts under condition AB. 'bin.mids': an array giving the mid-points of the time bins. 'bin.width': a scalar giving the bin width.

lengthScale	an array giving the length scale parameter values to be used for Gaussian process prior. Defaults to `sort(0.16 * resp.horiz / c(4, 3, 2, 1, 0.5, 0.1))` where `resp.horiz` is the time horizon of the response period.
lsPrior	an array of the same length as `lengthScale` giving the prior probabilities of the length scale values.
hyper	a list of hyper parameters with the following iterms. 'prec': a 2-vector giving the shape and rate parameters of the gamma distribution on the Dirichlet precision parameter. 'sig0': a scalaer giving the scale of the (centered) logistic distribution used in transforming the Gaussian random curves into curves restricted between 0 and 1.
burnIn	number of MCMC iterations to discard as burn-in.
nsamp	number of MCMC draws to be saved for posterior inference.
thin	the thinning rate at which MCMC draws are to be saved. The total number of iterations equals `burnIn + nsamp * thin`
plot	logical indicating if a graphical update should be plotted during the course of the MCMC
verbose	logical indicating if some fit details should be printed during the course of the MCMC
remove.zeros	logical indicating if trials with zero spike count shuold be removed from the analysis

Details

To be added...

Value

Returns a list of class "dapp" containting the following items.

lsProb	posterior preditctive draws of length scale
lambda.A	posterior draws of lambda.A at bin mid-points
lambda.B	posterior draws of lambda.B at bin mid-points
alpha	posterior draws of the alpha curves at bin mid-points
A	posterior draws of the latent variable A which gives the AB spike counts (by bin) that are to be attributed to signal A (the remaining are attributed to signal B)
prec	posterior draws of precision
alpha.pred	posterior predictive draws of alpha (of a future trial)
psl.pred	posterior predictive draw of the feature parameters (phi, psi, ell) (of a future trial)
details	mcmc details given as an array of `c(niter, nsamp, burnIn, thin, MH acceptance rate)`
hyper	hyper parameters used in model fitting
lengthScale	length scale set used in model fitting
lsPrior	length scale prior
bin.mids	bin mid-points
bin.width	bin width
mcmc	mcmc controls (burn-in length, thinning rate and number of saved draws)

Examples

```
## Not run:
synth.data <- synthesis.dapp(ntrials = c(15, 20, 20), pr.flat = 1,
                             intervals = list(c(0,.1), c(.45,.55), c(.9,1)),
                             wts = c(1/3, 1/3, 1/3), span = c(.1,.9),
                             period = c(500, 1500))

spike.counts <- list()
breaks <- seq(0, 1e3, 25)
spike.counts$Acounts <- sapply(synth.data$spiketimes$A, bin.counter, b = breaks)
spike.counts$Bcounts <- sapply(synth.data$spiketimes$B, bin.counter, b = breaks)
spike.counts$ABcounts <- sapply(synth.data$spiketimes$AB, bin.counter, b = breaks)
spike.counts$bin.mids <- breaks[-1] - mean(diff(breaks))/2
spike.counts$bin.width <- diff(breaks)[1]

fit.post <- dapp(spike.counts)
plot(fit.post, synth.data = synth.data)
## reanalyze forcing a uniform prior on range(alpha)
plot(fit.post, synth.data = synth.data, tilt = TRUE)
print(summary(fit.post, tilt = TRUE))


## End(Not run)
```

`dapp.simulate` *Simulate from Dynamic Admixture of Poisson Process*

Description

Simulate spike trains from DAPP model to binned spiking data

Usage

```
dapp.simulate(horizon = 1000, bin.width = 25, lengthScale,
     lsPrior = rep(1/length(lengthScale),length(lengthScale)),
     hyper = list(prec = c(1,1), sig0 = 1.87), nsamp = 1e3)
```

Arguments

`horizon`	time horizon of the response period (in ms)
`bin.width`	width of the time bins (in ms) to be used to aggregate spike counts
`lengthScale`	an array giving the length scale parameter values to be used for Gaussian process prior. Defaults to `sort(0.16 * resp.horiz / c(4, 3, 2, 1, 0.5, 0.1))` where `resp.horiz` is the time horizon of the response period.
`lsPrior`	an array of the same length as `lengthScale` giving the prior probabilities of the length scale values.
`hyper`	a list of hyper parameters with the following iterms. 'prec': a 2-vector giving the shape and rate parameters of the gamma distribution on the Dirichlet precision parameter. 'sig0': a scalaer giving the scale of the (centered) logistic distribution used in transforming the Gaussian random curves into curves restricted between 0 and 1.
`nsamp`	number of priors draws to be made

Details

Primarily intended to be used internally by the summary.dapp and plot.dapp functions. Could also be use to draw directly from the model.

Value

Returns a list of class "dapp" containting the following items.

lsProb	draws of length scale
alpha.pred	prior predictive draws of alpha
prec	draws of precision

Examples

```
## Not run:
prior <- dapp.simulate(1000, 25)

## End(Not run)
```

---

plot.dapp *Plotting Method for Dynamic Admixture of Poisson Process*

Description

Visually summarizes model fit of the DAPP model to binned spiking data

Usage

```
## S3 method for class 'dapp'
plot(x, add.prior = TRUE, synth.data = NULL,
     tilt.prior = FALSE, mesh.tilt = 0.1, nprior = x$mcmc["nsamp"],
     ncurves = 10, ...)
```

Arguments

x	a fitted model of the class 'dapp'
add.prior	logical indicating if prior predictive should be visualized
synth.data	for synthetic data provided, true alpha curves can be predicted
tilt.prior	lofical giving whether the prior should be tilted to mimic an analysis done with a uniform prior on the range(alpha)
mesh.tilt	a tuning parameter that controls how exactly tilting is done. Shorter mesh value gives tighter match but will require more Monte Carlo simulations
nprior	number of prior draws to be used for display
ncurves	number of curves to be shown individually
...	no addiitonal parameters used at this point

Details

To be added..

Value

Gives prior and posterior summaries of the range and average predicted alpha curves

Examples

```
## Not run:
synth.data <- synthesis.dapp(ntrials = c(15, 20, 20), pr.flat = 1,
                             intervals = list(c(0,.1), c(.45,.55), c(.9,1)),
                             wts = c(1/3, 1/3, 1/3), span = c(.1,.9),
                            period = c(500, 1500))

spike.counts <- list()
breaks <- seq(0, 1e3, 25)
spike.counts$Acounts <- sapply(synth.data$spiketimes$A, bin.counter, b = breaks)
spike.counts$Bcounts <- sapply(synth.data$spiketimes$B, bin.counter, b = breaks)
spike.counts$ABcounts <- sapply(synth.data$spiketimes$AB, bin.counter, b = breaks)
spike.counts$bin.mids <- breaks[-1] - mean(diff(breaks))/2
spike.counts$bin.width <- diff(breaks)[1]

fit.post <- dapp(spike.counts)
plot(fit.post, synth.data = synth.data)
## reanalyze forcing a uniform prior on range(alpha)
plot(fit.post, synth.data = synth.data, tilt = TRUE)
print(summary(fit.post, tilt = TRUE))


## End(Not run)
```

---

poisson.tests *Poisson Tests for Whole Trial Spike Counts*

Description

Carries out various Poisson related tests for double-stimuli spike count distribution.

Usage

```
poisson.tests(xA, xB, xAB, labels = c("A", "B", "AB"),
     remove.zeros = FALSE, plot = FALSE,
     gamma.pars = c(0.5, 2e-10), beta.pars = c(0.5, 0.5))
```

Arguments

xA	an array of whole-trial spike counts under stimulus 1
xB	an array of whole-trial spike counts under stimulus 2
xAB	an array of whole-trial spike counts when both stimuli are present together
labels	labels for stimlus conditions
remove.zeros	whether to remove trials with zero spike counts
plot	logical indicating if a visualization plot should be made
gamma.pars	shape and rate parameters of the gamma prior on Poisson mean
beta.pars	shape parameters of the beta prior for the mixture/intermediate parameter

Details

To be added...

Value

Returns a list with the following items:

`separation.logBF`	the (log) Bayes factor for testing that that two single stimulus distributions are different
`post.prob`	posterior probabilities of the four hypotheses (Mixture, Intermediate, Outside, Single) under equal prior probabilities
`pois.pvalue`	minimum of the two p-values checking for Poisson-ness of each single stimulus distribution
`sample.sizes`	three trial counts for A, B and AB conditions

Examples

```
## Not run:
nA <- 20; nB <- 15; nAB <- 25
muA <- 25; muB <- 40
Acounts <- rpois(nA, muA)
Bcounts <- rpois(nB, muB)
ABcounts <- rpois(nAB, sample(c(muA, muB), nAB, replace = TRUE))
poisson.tests(Acounts, Bcounts, ABcounts)

## End(Not run)
```

---

summary.dapp *Summary Method for Dynamic Admixture of Poisson Process*

Description

Summarizes model fit of the DAPP model to binned spiking data

Usage

```
## S3 method for class 'dapp'
summary(object, cut.width = 0.1, tilt.prior = FALSE,
     mesh.tilt = 0.1, nprior = object$mcmc["nsamp"], ...)
```

Arguments

`object`	a fitted model of the class 'dapp'
`cut.width`	width of the bins to be used to cut range(alpha) and average(alpha) characteristics
`tilt.prior`	lofical giving whether the prior should be tilted to mimic an analysis done with a uniform prior on the range(alpha)
`mesh.tilt`	a tuning parameter that controls how exactly tilting is done. Shorter mesh value gives tighter match but will require more Monte Carlo simulations
`nprior`	number of prior draws to be used for display
`...`	no addiitonal parameters used at this point

Details

To be added...

Value

Gives prior and posterior summaries of the range and average predicted alpha curves

Examples

```
## Not run:
synth.data <- synthesis.dapp(ntrials = c(15, 20, 20), pr.flat = 1,
                             intervals = list(c(0,.1), c(.45,.55), c(.9,1)),
                             wts = c(1/3, 1/3, 1/3), span = c(.1,.9),
                             period = c(500, 1500))

spike.counts <- list()
breaks <- seq(0, 1e3, 25)
spike.counts$Acounts <- sapply(synth.data$spiketimes$A, bin.counter, b = breaks)
spike.counts$Bcounts <- sapply(synth.data$spiketimes$B, bin.counter, b = breaks)
spike.counts$ABcounts <- sapply(synth.data$spiketimes$AB, bin.counter, b = breaks)
spike.counts$bin.mids <- breaks[-1] - mean(diff(breaks))/2
spike.counts$bin.width <- diff(breaks)[1]

fit.post <- dapp(spike.counts)
plot(fit.post, synth.data = synth.data)
## reanalyze forcing a uniform prior on range(alpha)
plot(fit.post, synth.data = synth.data, tilt = TRUE)
print(summary(fit.post, tilt = TRUE))


## End(Not run)
```

`synthesis.dapp` *Simulate Multiplexing Data for DAPP Analysis*

Description

Simulate spike trains from controlled DAPP setting with flat and sinusoidal weight curves

Usage

```
synthesis.dapp(ntrials = c(10, 10, 10), time.bins = 0:1000, lambda.A = 400,
     lambda.B = 100, pr.flat = 0.5, intervals = list(c(0,1)),
     wts = 1, span = c(0,1), period.range = c(400, 1000))
```

Arguments

`ntrials`	an array with 3 elements giving the trial counts for conditions A, B and AB
`time.bins`	time bins (in ms) giving the break points of the time bins in which Poisson draws should be made to mimic a Poisson process generation
`lambda.A`	a flat intensity (in Hz) for condition A

lambda.B	a flat intensity (in Hz) for condition B
pr.flat	proportion of flat weight curves to be generated
intervals	a list of sub-intervals (each represented by the 2-vector giving the sub-interval end-points) which determine the ranges of the flat weight curves
wts	the relative weights of the sub-intervals above
span	a two-vector giving the range of the sinusoidal weight curves
period.range	the range from which the sinusoidal periods are drawn randomly (and uniformly)

Value

Returns a list containting the following items.

spiketimes	a list of spiketimes for each AB trial
alphas	true underlying weight curves for each AB trial
lambdas	corresponding intensity curves for each AB trial
time.pts	time points associated with alphas and lambdas

Examples

```
## generate 20 AB trials, roughl half with flat weight curves
## with a constant intensity either in (0,.1) or in (0.9, 1)
## (equally likely). The remaining curves are sinusoidal
## that snake between 0.1 and 0.9 with a period randomly
## drawn between 500 and 1500

synth.data <- synthesis.dapp(ntrials = c(15, 20, 20), pr.flat = 1,
                             intervals = list(c(0,.1), c(.45,.55), c(.9,1)),
                             wts = c(1/3, 1/3, 1/3), span = c(.1,.9),
                             period = c(500, 1500))

spike.counts <- list()
breaks <- seq(0, 1e3, 25)
spike.counts$Acounts <- sapply(synth.data$spiketimes$A, bin.counter, b = breaks)
spike.counts$Bcounts <- sapply(synth.data$spiketimes$B, bin.counter, b = breaks)
spike.counts$ABcounts <- sapply(synth.data$spiketimes$AB, bin.counter, b = breaks)
spike.counts$bin.mids <- breaks[-1] - mean(diff(breaks))/2
spike.counts$bin.width <- diff(breaks)[1]
```

Index