Package 'neuromplex'

July 22, 2025

Version 1.0-1

Date 2021-04-21

Title Neural Multiplexing Analysis

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

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

Depends R (>= 3.6)

Imports BayesLogit, ggplot2, dplyr, tidyr, magrittr, gridExtra

Description Statistical methods for whole-trial and time-domain analysis of single cell neural response to multiple stimuli presented simultaneously. The package is based on the paper by C Glynn, ST Tokdar, A Zaman, VC Caruso, JT Mohl, SM Willett, and JM Groh (2021) ``Analyzing second order stochasticity of neural spiking under stimuli-bundle exposure'', is in press for publication by the Annals of Applied Statistics. A preprint may be found at <doi:10.48550/arXiv.1911.04387>.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2021-04-22 12:00:02 UTC

Contents

`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, w=c(1,1)),
     burnIn = 1e3, nsamp = 1e3, thin = 4,
     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`
`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

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)

References

Glynn, C., Tokdar, S.T., Zaman, A., Caruso, V.C., Mohl, J.T., Willett, S.M., and Groh, J.M. (2020+). Analyzing second order stochasticity of neural spiking under stimuli-bundle exposure. The Annals of Applied Statistics. Accepted.

See Also

plot.dapp, summary.dapp and predict.dapp.

Examples

```
## Note:
#### The example below uses a simpler synthetic data, a wider bin-width
#### and a shorter MCMC run to keep the run length less than 5s
#### Use ?plot.dapp or ?plot.summary for a more realistic example

## Generate 30 A and 30 B trials with rate functions
##    lambda.A(t) = 160*exp(-2*t/1000) + 40*exp(-0.2*t/1000)
##    lambda.B(t) = 40*exp(-2*t/1000)
## where time t is measured in ms. Then, generate 25 AB trials,
## roughly 2/3 with flat weight curves with a constant intensity
## either close to A, or close to B (equally likely). The
## remaining 1/3 curves are sinusoidal that snake between 0.01 and 0.99
## with a period randomly drawn between 400 and 1000

ntrials <- c(nA=30, nB=30, nAB=25)
```

```
flat.range <- list(A=c(0.85, 0.95),
                   B=c(0.05, 0.15))
flat.mix <- c(A=1/2, B=1/2)
wavy.span <- c(0.01, 0.99)
wavy.period <- c(400, 1000)

T.horiz <- 1000
rateB <- 40 * exp(-2*(1:T.horiz)/T.horiz)
rateA <- 4*rateB + 40 * exp(-0.2*(1:T.horiz)/T.horiz)

synth.data <- synthesis.dapp(ntrials = ntrials, pr.flat = 2/3,
                             intervals = flat.range, wts = flat.mix,
                             span = wavy.span, period.range = wavy.period,
                             lambda.A=rateA, lambda.B=rateB)

## Generate binned spike counts witb 100 ms bins
spike.counts <- mplex.preprocess(synth.data$spiketimes, bw=100, visualize=FALSE)

## Fit the DAPP model to data
#### A short MCMC run is done below to keep the run length short.
#### Use default or larger values for burn, nsamp and thin
#### for more reliable estimation
fit.post <- dapp(spike.counts, burn=10, nsamp=90, thin=1, verbose=FALSE)

## Visualize model fit
plot(fit.post)

## Post process results to assign second order stochasticity labels
summary(fit.post)
```

`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, w=c(1,1)), 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

```
prior <- dapp.simulate(1000, 25)
```

---

mplex.preprocess *Preprocessing Neural Multiplexing Data*

Description

Preprocess nueral spike train recording to preapre binned spike counts suitable for DAPP analysis

Usage

```
mplex.preprocess(spiketimes, start.time=0, end.time=1e3, bw=50,
    remove.zeros=FALSE, visualize=TRUE, ...)
```

Arguments

spiketimes	a list with 3 elements giving the 3 sets of spiketimes associated with experimental conditions A, B and AB
start.time	starting time for the observation window. See details below
end.time	ending time of the observations window. See details below

bw	bin width (in ms) used for binning. A single bin is used when bw equals or exceeds the length of the observation period (end.time - start.time). Single bin analysis is same as total spike count analysis
remove.zeros	logical indicating if trials with zero spike counts should be removed from the analysis
visualize	logical indicating if a graphical summary should be produced to visualize the three sets of trials
...	additional commands to be passed on to grid.arrange() for plotting. For example, adding 'top="PLOT TITLE"' will add a title at the top of the combined plot. See grid.arrange for more details.

Value

Returns a list containting the following items.

Acounts	binned spike counts under condition A presented as a matrix. Rows are bins, columns are replicates (trials). In case of single bin analysis, i.e., with bw equal or larger than total observation window length, a vector of counts is returned.
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
time.horizon	a vector of length 2 giving the start and the end times of the observation period

Examples

```
## generate 25 A and 30 B trials with rate functions
##    lambda.A(t) = 160*exp(-2*t/1000) + 40*exp(-0.2*t/1000)
##    lambda.B(t) = 40*exp(-2*t/1000)
## where time t is measured in ms. Then, generate 40 AB trials,
## roughly half with flat weight curves with a constant intensity
## either close to A, or close to B or close to the 50-50 mark,
## (equally likely). The remaining curves are sinusoidal
## that snake between 0.01 and 0.99 with a period randomly
## drawn between 400 and 1000

ntrials <- c(nA=25, nB=30, nAB=40)
flat.range <- list(A=c(0.85, 0.95),
                   B=c(0.05, 0.15),
                   mid=c(0.45,0.55))
flat.mix <- c(A=1/3, B=1/3, mid=1/3)
wavy.span <- c(0.01, 0.99)
wavy.period <- c(400, 1000)

T.horiz <- 1000
rateB <- 40 * exp(-2*(1:T.horiz)/T.horiz)
rateA <- 4*rateB + 40 * exp(-0.2*(1:T.horiz)/T.horiz)

synth.data <- synthesis.dapp(ntrials = ntrials, pr.flat = 0.5,
```

```
                              intervals = flat.range, wts = flat.mix,
                              span = wavy.span, period.range = wavy.period,
                              lambda.A=rateA, lambda.B=rateB)

## Visualize data and generate binned spike counts
spike.counts <- mplex.preprocess(synth.data$spiketimes, visualize=TRUE,
top="Synthetic data: bin size=50ms")

## Not run:
## Visualize total spike counts data
spike.counts <- mplex.preprocess(synth.data$spiketimes, bw=Inf, visualize=TRUE,
top="Synthetic data: total spike counts")

## 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, tilt.prior = FALSE, mesh.tilt = 0.1,
     nprior = x$mcmc["nsamp"], ncurves = 10,
     simple.layout = FALSE, ...)
```

Arguments

`x`	a fitted model of the class 'dapp'
`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
`simple.layout`	logical indicating if a simpler graphical output should be returned with only predictive visualization
`...`	additional commands to be passed on to grid.arrange() for plotting. For example, adding 'top="PLOT TITLE"' will add a title at the top of the combined plot. See `grid.arrange` for more details.

Value

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

See Also

dapp, predict.dapp and summary.dapp.

Examples

```
## Not run:
## generate 25 A and 30 B trials with rate functions
##    lambda.A(t) = 160*exp(-2*t/1000) + 40*exp(-0.2*t/1000)
##    lambda.B(t) = 40*exp(-2*t/1000)
## where time t is measured in ms. Then, generate 40 AB trials,
## roughly half with flat weight curves with a constant intensity
## either close to A, or close to B or close to the 50-50 mark,
## (equally likely). The remaining curves are sinusoidal
## that snake between 0.01 and 0.99 with a period randomly
## drawn between 400 and 1000

ntrials <- c(nA=25, nB=30, nAB=40)
flat.range <- list(A=c(0.85, 0.95),
                   B=c(0.05, 0.15),
                   mid=c(0.45,0.55))
flat.mix <- c(A=1/3, B=1/3, mid=1/3)
wavy.span <- c(0.01, 0.99)
wavy.period <- c(400, 1000)

T.horiz <- 1000
rateB <- 40 * exp(-2*(1:T.horiz)/T.horiz)
rateA <- 4*rateB + 40 * exp(-0.2*(1:T.horiz)/T.horiz)

synth.data <- synthesis.dapp(ntrials = ntrials, pr.flat = 0.5,
                             intervals = flat.range, wts = flat.mix,
                             span = wavy.span, period.range = wavy.period,
                             lambda.A=rateA, lambda.B=rateB)

## Visualize data and generated binned spike counts
spike.counts <- mplex.preprocess(synth.data$spiketimes, visualize=FALSE)

## Fit the DAPP model to data
fit.post <- dapp(spike.counts, verbose=FALSE)

## Visualize model fit
plot(fit.post)

## Post process results to assign second order stochasticity labels
summary(fit.post)

## 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,
              gamma.pars = c(0.5, 2e-10), beta.pars = c(0.5, 0.5),
              nMC = 1000, plot = FALSE, add.poisson.fits = FALSE,
              method.screen = c('variance', 'bincount'), ...)
```

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
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
nMC	number of Monte Carlo samples to be used in numerical approximations.
plot	logical indicating if a visualization plot should be made
add.poisson.fits	logical indicating if a fitted Poisson pmfs will be overlaid in the visualization. Ignored when plot=FALSE.
method.screen	a character string, default is 'variance' which uses the Poisson variance test to assess whether a Poisson distribution fits a sample of counts. Alternative choice is 'bincount' which uses an binned histogram based nonparametric chi-square goodness of fit test
...	additional commands to be passed on to grid.arrange() for plotting. For example, adding 'top="PLOT TITLE"' will add a title at the top of the combined plot. See grid.arrange for more details.

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

```
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, nMC=200, plot=FALSE)
```

`predict.dapp` *Predict Method for Dynamic Admixture of Poisson Process*

Description

Summarizes predictive draws of weight curves from a fitted DAPP model

Usage

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

Arguments

`object`	a fitted model of the class 'dapp'
`tilt.prior`	logical 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

This function is intended to be mostly used through `predict.dapp`.

Value

Gives prior and posterior summaries of the range and average predicted alpha curves. Also gives the same for the posterior draws of alpha for each recorded AB trial.

See Also

`dapp`, `plot.dapp` and `summary.dapp`.

Examples

```
## Not run:
## generate 25 A and 30 B trials with rate functions
##    lambda.A(t) = 160*exp(-2*t/1000) + 40*exp(-0.2*t/1000)
##    lambda.B(t) = 40*exp(-2*t/1000)
## where time t is measured in ms. Then, generate 40 AB trials,
## roughly half with flat weight curves with a constant intensity
## either close to A, or close to B or close to the 50-50 mark,
## (equally likely). The remaining curves are sinusoidal
## that snake between 0.01 and 0.99 with a period randomly
## drawn between 400 and 1000

ntrials <- c(nA=25, nB=30, nAB=40)
flat.range <- list(A=c(0.85, 0.95),
                   B=c(0.05, 0.15),
                   mid=c(0.45,0.55))
flat.mix <- c(A=1/3, B=1/3, mid=1/3)
wavy.span <- c(0.01, 0.99)
wavy.period <- c(400, 1000)

T.horiz <- 1000
rateB <- 40 * exp(-2*(1:T.horiz)/T.horiz)
rateA <- 4*rateB + 40 * exp(-0.2*(1:T.horiz)/T.horiz)

synth.data <- synthesis.dapp(ntrials = ntrials, pr.flat = 0.5,
                             intervals = flat.range, wts = flat.mix,
                             span = wavy.span, period.range = wavy.period,
                             lambda.A=rateA, lambda.B=rateB)

## Visualize data and generated binned spike counts
spike.counts <- mplex.preprocess(synth.data$spiketimes, visualize=TRUE)

## Fit the DAPP model to data
fit.post <- dapp(spike.counts, verbose=FALSE)

## Prediction
pp <- predict(fit.post)

## Visualizing (range, ave) of alpha(t) for each recorded AB trial
te <- pp$trial.est
ggplot(te, aes(x=ave, y=range)) +
    stat_density_2d(aes(fill = ..level..), h=0.2, geom = "polygon") +
    scale_fill_viridis_c() +
    theme_bw() +
    facet_wrap(~as.factor(trial))

## Post process results to assign second order stochasticity labels
summary(fit.post)

## End(Not run)
```

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

Description

Presents post-processing labels from a DAPP model fit to binned spiking data

Usage

```
## S3 method for class 'dapp'
summary(object, flat.cut = 0.15, wavy.cut = 0.85,
    extreme.cut = 0.25, ...)
```

Arguments

object	a fitted model of the class 'dapp'
flat.cut	maximum range allowed to be labelled 'flat'
wavy.cut	minimum range allowed to be labelled 'wavy'
extreme.cut	for flat curves, maximum deviation from extremes (0 or 1) allowed to be labelled flat.B or flat.A (respectivel)
...	additional parameters passed on to the call of predict.dapp

Details

The summary function analyzes the prior and posterior predictive draws of the weight curves alpha(t). Each draw is assigned with one of the following labels: 'flat.A', 'flat.B', 'flat.Mid', 'wavy', or 'others'. The proportions of these categories are printed for the prior and posterior sets. Additionally, posterior draws of alpha(t), for each recorded AB trial, are also analyzed in the same way to produce similar labels for each trial, and, the trial is given the label that has the maximum posterior probability.

Value

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

See Also

dapp, plot.dapp and predict.dapp.

Examples

```
## Not run:
## generate 25 A and 30 B trials with rate functions
##    lambda.A(t) = 160*exp(-2*t/1000) + 40*exp(-0.2*t/1000)
##    lambda.B(t) = 40*exp(-2*t/1000)
## where time t is measured in ms. Then, generate 40 AB trials,
## roughly half with flat weight curves with a constant intensity
```

```
## either close to A, or close to B or close to the 50-50 mark,
## (equally likely). The remaining curves are sinusoidal
## that snake between 0.01 and 0.99 with a period randomly
## drawn between 400 and 1000

ntrials <- c(nA=25, nB=30, nAB=40)
flat.range <- list(A=c(0.85, 0.95),
                   B=c(0.05, 0.15),
                   mid=c(0.45,0.55))
flat.mix <- c(A=1/3, B=1/3, mid=1/3)
wavy.span <- c(0.01, 0.99)
wavy.period <- c(400, 1000)

T.horiz <- 1000
rateB <- 40 * exp(-2*(1:T.horiz)/T.horiz)
rateA <- 4*rateB + 40 * exp(-0.2*(1:T.horiz)/T.horiz)

synth.data <- synthesis.dapp(ntrials = ntrials, pr.flat = 0.5,
                             intervals = flat.range, wts = flat.mix,
                             span = wavy.span, period.range = wavy.period,
                             lambda.A=rateA, lambda.B=rateB)

## Visualize data and generated binned spike counts
spike.counts <- mplex.preprocess(synth.data$spiketimes, visualize=TRUE)

## Fit the DAPP model to data
fit.post <- dapp(spike.counts, verbose=FALSE)

## Visualize model fit
plot(fit.post)

## Post process results to assign second order stochasticity labels
summary(fit.post)

## 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	a vector of 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 with 3 elements giving the 3 sets of spiketimes associated with experimental conditions A, B and AB
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 25 A and 30 B trials with rate functions
##    lambda.A(t) = 160*exp(-2*t/1000) + 40*exp(-0.2*t/1000)
##    lambda.B(t) = 40*exp(-2*t/1000)
## where time t is measured in ms. Then, generate 40 AB trials,
## roughly half with flat weight curves with a constant intensity
## either close to A, or close to B or close to the 50-50 mark,
## (equally likely). The remaining curves are sinusoidal
## that snake between 0.01 and 0.99 with a period randomly
## drawn between 400 and 1000

ntrials <- c(nA=25, nB=30, nAB=40)
flat.range <- list(A=c(0.85, 0.95),
                   B=c(0.05, 0.15),
                   mid=c(0.45,0.55))
flat.mix <- c(A=1/3, B=1/3, mid=1/3)
wavy.span <- c(0.01, 0.99)
wavy.period <- c(400, 1000)

T.horiz <- 1000
rateB <- 40 * exp(-2*(1:T.horiz)/T.horiz)
rateA <- 4*rateB + 40 * exp(-0.2*(1:T.horiz)/T.horiz)
```

```
synth.data <- synthesis.dapp(ntrials = ntrials, pr.flat = 0.5,
                             intervals = flat.range, wts = flat.mix,
                             span = wavy.span, period.range = wavy.period,
                             lambda.A=rateA, lambda.B=rateB)

## Visualize data and generate binned spike counts
spike.counts <- mplex.preprocess(synth.data$spiketimes, visualize=TRUE, top="Synthetic Data")
```

Index