Package: ssMousetrack (via r-universe)

October 29, 2026

Title Bayesian State-Space Modeling of Mouse-Tracking Experiments via Stan

Version 1.1.7

Date 2025-03-16

Description Estimates previously compiled state-space modeling for mouse-tracking experiments using the 'rstan' package, which provides the R interface to the Stan C++ library for Bayesian estimation.

License GPL (>= 3)

Depends methods, R (>= 3.4.0), Rcpp (>= 1.0.0)

Imports rstan (>= 2.18.2), rstantools (>= 1.5.1), CircStats, dtw, ggplot2, cowplot, RcppParallel

LinkingTo BH (>= 1.66.0-1), Rcpp (>= 1.0.0), RcppEigen (>= 0.3.3.5.0), rstan (>= 2.18.2), StanHeaders (>= 2.18.0), RcppParallel

Encoding UTF-8

LazyData true

NeedsCompilation yes

SystemRequirements GNU make

RoxygenNote 7.2.3

BugReports https://github.com/antcalcagni/ssMousetrack/issues

Author Antonio Calcagnì [aut, cre], Massimiliano Pastore [ctb]

Maintainer Antonio Calcagnì <ant.calcagni@gmail.com>

Config/pak/sysreqs make

Repository https://cran.r-universe.dev

Date/Publication 2025-03-17 16:20:02 UTC

RemoteUrl https://github.com/cran/ssMousetrack

RemoteRef HEAD

RemoteSha 8bdf13c5106b9e87880d5fb9e5645e334a5841fd

Contents

ssMousetrack-package . . . 2
check_prior . . . 4
compute_D . . . 6
congruency . . . 6
evaluate_ssm . . . 7
generate_data . . . 8
generate_design . . . 10
generate_Z . . . 11
language . . . 12
prepare_data . . . 13
run_ssm . . . 14

Index **17**

`ssMousetrack-package` *Bayesian State-Space Modeling of Mouse-Tracking Experiments Via Stan*

Description

The **ssMousetrack** package allows analysing mouse-tracking experiments via Bayesian state-space modeling. The package estimates the model using Markov Chain Monte Carlo, variational approximations to the posterior distribution, or optimization, as implemented in the **rstan** package. The user can use the customary R modeling syntax to define equations of the model and Stan syntax to specify priors over the model parameters.

The sections below provide an overview of the state-space model implemented by the **ssMousetrack** package.

Details

(i) Mouse-tracking data

The raw data of a mouse-tracking experiment for *I* individuals and *J* stimuli consist of a collection of arrays $(x, y)_{ij} = (x_0, ..., x_{N_{ij}}; y_0, ..., y_{N_{ij}})$ which contain ordered N_{ij} x 1 sequences of x-y Cartesian coordinates as mapped to the computer-mouse pointer. The x-y coordinates are pre-processed according to the following steps:

1. *Realigning*: the arrays $(x, y)_{ij}$ are re-aligned on a common sampling scale, so that *N* indicates the cumulative amount of progressive time from 0% to N = 100%, with *N* being the same over $i = 1, ..., I$ and $j = 1, ..., J$
2. *Normalization*: the aligned arrays $(x, y)_{ij}$ are normalized so that $(x_0, y_0)_{ij} = (0, 0)$ and $(x_N, y_N)_{ij} = (1, 1)$ for each $i = 1, ..., I$ and $j = 1, ..., J$
3. *Translation*: the normalized arrays $(x, y)_{ij}$ are translated into the quadrant $[-1, 1]$x$[0, 1]$
4. *atan2 projection*: the final arrays $(x, y)_{ij}$ are projected onto a lower-subspace via the *atan2* function by getting the ordered collection of radians $(y)_{ij} = (y_0, ..., y_N)$ in the subset of reals $(0, \pi]^N$, for each $i = 1, ..., I$ and $j = 1, ..., J$.

The final I x J x N array of data **Y** contains the mouse-tracking trajectories expressed in terms of angles. These trajectories lie on the arc defined by the union of two disjoint sets, namely the sets $\{y_0, ..., y_N : y_n \geq \pi/2\}$ (target's hemispace) and $\{y_0, ..., y_N : y_n < (3\pi)/4\}$ (distractor's hemispace), with $\pi/2$ and $(3\pi)/4$ being the location points for target and distractor, respectively.

Note that, the current version of **ssMousetrack** package requires the number of stimuli J to be the same over the subjects $i = 1, ..., I$.

The pre-processed mouse-tracking trajectories are analysed using the state-space modeling described below.

(ii) Model representation

The array **Y** contains the *observed data* expressed in angles. The *measurement equation* of the model is:

$$y_{ij}^{(n)} \sim vonMises\big(\mu_{ij}^{(n)}, \kappa_{ij}^{(n)}\big)$$

where $\mu_{ij}^{(n)}$ and $\kappa_{ij}^{(n)}$ are the location and the concentration parameters for the vonMises probability law. The moving mean on the arc $\mu_{ij}^{(n)}$ is defined as:

$$\mu_{ij}^{(n)} := G(\beta, x_i^{(n)})$$

with β being a J x 1 array of real parameters representing the contribution of the *j*-th stimulus on the observed trajectory $y_{ij} = (y^{(0)}, ..., y^{(N)})$ whereas G is a non-linear function mapping reals to the subset $(0, \pi]$ of the form: (i) $\big[(1 + \exp(\beta - x_i^{(n)}))\big]\pi^{-1}$ (logistic), (ii) $\big[\exp(-\beta \exp(-x_i^{(n)}))\big]\pi$ (gompertz). In the G equation, $x_i^{(n)}$ is a real random quantity obeying to the law:

$$x_i^{(n)} \sim Normal\big(x_i^{(n-1)}, \sigma_i^2\big)$$

which represents a random walk process with time-fixed variance σ_i^2. The terms $x_i = (x_i^{(0)}, ..., x_i^{(N)})$ are the individual latent dynamics unaffected by the stimuli (i.e., how individual differ in executing the task) whereas β contains the experimental effects regardless to the individual dynamic (i.e., how experimental variables act on the individual dynamics to produce the observed responses).

The terms $\beta = (\beta_1, ..., \beta_J)$ are defined according to the following linear combination:

$$\beta_j := \sum_{k=1}^{K} z_{jk}\gamma_k$$

where z_{jk} is an element of the J x K *dummy matrix* **Z** representing main and high-order effects of the experimental design.

The terms $\kappa_{ij} = (\kappa_{ij}^{(0)}, ..., \kappa_{ij}^{(N)})$ are computed as follows:

$$\kappa_{ij}^{(n)} := \exp^o\big(\delta_{ij}^{(n)}\big)$$

where $\delta_{ij}^{(n)} = |y_{ij}^{(n)} - (3\pi)/4|$ (if $y_{ij}^{(n)} < \pi/2$) or $\delta_{ij}^{(n)} = |y_{ij}^{(n)} - \pi/4|$ (if $y_{ij}^{(n)} \geq \pi/2$). The function $\exp^o$ is the exponential function scaled in the natural range of the parameters κ_{ij} (positive real numbers).

(iii) Bayesian formulation

The state-space model in the **ssMousetrack** package requires estimating the array of latent trajectories **X** and the K x 1 parameters γ. Let Θ representing both the unknown quantities, the posterior density after factorization is:

$$f(\Theta|Y) \propto f(\gamma) \prod_{i=1}^{I} \prod_{j=1}^{J} f(\gamma|y_{ij}) \prod_{i=1}^{I} \prod_{j=1}^{J} f(x_i|y_{ij})$$

Sampling from $f(\Theta|Y)$ is solved via *marginal MCMC* where the term $f(x_i|y_{ij})$ is approximated by means of Kalman filtering/smoothing. The marginal Likelihood of the model used for the rejection criterion of the MCMC sampler is approximated with the Normal distribution using the Kalman filter theory.

References

Calcagnì, A., Lombardi, L., D'Alessandro, M., & Freuli F. (2019). A state space approach to dynamic modeling of mouse-tracking data. *Frontiers in Psychology: Quantitative Psychology and Measurement*

Särkkä, S. (2013). Bayesian Filtering and Smoothing. *Cambridge University Press*

Durbin, J., & Koopman, S. J. (2012). Time series analysis by state space methods (Vol. 38). *Oxford University Press*

Andrieu, C., Doucet, A., & Holenstein, R. (2010). Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3), 269s-342

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). Bayesian Data Analysis (Second edition). Chapman & Hall/CRC.

See Also

https://mc-stan.org/ for more information on the Stan C++ language used by **ssMousetrack** package

Jokkala, J. (2016). Github repository: *kalman-stan-randomwalk*, https://github.com/juhokokkala/kalman-stan-randomwalk

---

`check_prior` *Check prior distributions*

Description

Check prior distributions

Usage

```
check_prior(priors = NULL)
```

Arguments

`priors`	(list) a list of arguments specifying priors for each parameter involved in the model (see Details). If `priors=NULL` then predefined priors will be returned.

Details

The function is used to specify the prior-related arguments of the state-space modeling function `run_ssm`. Priors are specified in terms of distributions and associated parameters as implemented in the **rstan** package. The available options are as follows:

- `lognormal(mu,sigma)` (code = 1, required parameters = 2)
- `normal(mu,sigma)` (code = 2, required parameters = 2)
- `normal(mu,sigma)T(min,max)` (code = 201, required parameters = 4)
- `chi_square(df)` (code = 3, required parameters = 1)
- `inv_chi_square(df)` (code = 4, required parameters = 1)
- `gamma(alpha,beta)` (code = 5, required parameters = 2)
- `pareto(min,alpha)` (code = 6, required parameters = 2)
- `uniform(min,max)` (code = 7, required parameters = 2)

This is an internal function, generally not to be called by the user.

Value

a matrix contatining priors (numeric codes, see Details) and their parameters

Examples

```
## Not run:
## Define priors for all the paramaters
priors_list <- list("lognormal(1,1)","normal(2,3)T(0,10)","normal(3,1)")
priors_out <- check_prior(priors_list)
print(priors_out)

## Define priors for some of the paramaters
priors_list <- list(NULL,"pareto(1,1.2)",NULL)
priors_out <- check_prior(priors_list)
print(priors_out)

## Use pre-defined vague priors for all the parameters
priors_list <- list(NULL,NULL,NULL)
priors_out <- check_prior(priors_list)
print(priors_out)

## End(Not run)
```

compute_D *Compute the matrix of distances D for kappa parameters*

Description

Compute the matrix of distances D for kappa parameters

Usage

```
compute_D(Y = NULL, y_T = pi/4, y_D = (3 * pi)/4)
```

Arguments

Y	(matrix) N x JI matrix of observed trajectories
y_T	(numeric) position in angles of the target
y_D	(numeric) position in angles of the distractor

Details

The function compute the distance of the Y-trajectories from the distractor and target points. This is an internal function, generally not to be called by the user.

Value

a N x JI matrix containing the delta values for each data point in Y

Examples

```
## Generate a generic matrix Y of I = 5 individuals and J = 1 trajectories (N = 61)
I <- 5; N <- 61
y_T <- pi/4; y_D <- (3*pi)/4
Y <- matrix(stats::rnorm(n = N*I,mean = (y_T+y_D)/2,sd = 10),N,I)
DY <- compute_D(Y=Y,y_T=y_T,y_D=y_D)
```

congruency *Mouse-tracking experiment of a memory task*

Description

This dataset contains a subset of data originally presented in Coco & Duran (2016). In this task participants see sentence and scene pairs that varied in plausibility and are requested to classify the pairs as congruent or incongruent. The experimental variables are *congruency* with two categorical levels (i.e., congruent, incongruent) and *plausibility* with two categorical levels (i.e., plausible, implausible). Participants have to classify each stimulus as belonging to one of these four levels.

The dataset contains two participants (I=2), each measured along three trials, two categorical variables (Q=2) each with two levels (K=2). The total number of trials is J=12. Mouse-tracking trajectories are raw-data, i.e. they have not been previously pre-processed.

Usage

```
congruency
```

Format

A long-format dataframe of 728 observations containing information on the following variables.

sbj The ID number of participants

trial The ID number of trials

congruency A factor of levels `congruent`, `incongruent`

plausibility A factor of levels `plausible`, `implausible`

timestep The ID number of the recorded x-y trajectories

x The recorded x-trajectories

y The recorded y-trajectories

Source

Coco, M. I., & Duran, N. D. (2016). When expectancies collide: Action dynamics reveal the interaction between stimulus plausibility and congruency. *Psychonomic bulletin & review*, 23(6), 1920-1931.

`evaluate_ssm` *Evaluate the adequacy of the state-space model to reproduce the observed data*

Description

Evaluate the adequacy of the state-space model to reproduce the observed data

Usage

```
evaluate_ssm(ssmfit = NULL, M = 100, plotx = TRUE)
```

Arguments

`ssmfit`	(list) output of `run_ssm` function
`M`	(integer) number of replications
`plotx`	(boolean) if `plotx=TRUE` the function returns a graphical representation for the fit indices

Details

The function implements a simulated-based method for assessing the adequacy of the model to reproduce the observed data. In particular, the function provides two type of model adequacy, i.e. overall (PA_ov) and by-subject (PA_sbj). In the overall case the function provides the total amount of data reconstruction based on the I x J x N matrix Y of observed data. By contrast, in the second case the function provides the adequacy of the model to reconstruct the individual-based set of data as it works on the matrix J x N over i=1,...,I. Both the indices are in the range 0% - 100%, with 100% indicating perfect fit. In addition, the function returns a by-subject distance-based index (Dynamic Timw Warp distance) between observed and reproduced trajectories using `dtw` function.

Value

a datalist containing the adequacy indices

Examples

```
## Not run:
## Fit a state-space model using simulated data
# Generate mouse-tracking data for an univariate experimental design with K = 3 categorical levels,
## J = 12 trials, I = 5 subjects
X1 <- generate_data(I=5,J=12,K=3,Z.formula="~Z1")
iid <- 23 # keep just one dataset from the simulated set of datasets
# Run the state-space model on the chosen dataset
X1_fit <- run_ssm(N = X1$N,I = X1$I,J = X1$J,Y = X1$data$Y[iid,,],D = X1$data$D[iid,,],
Z = X1$data$Z,niter=100,nwarmup=25)
# Evaluate the state-space model
evaluate_ssm(ssmfit = X1_fit,M = 10,plotx=FALSE)

## End(Not run)
```

---

generate_data *Generate datasets according to the model structure*

Description

Generate datasets according to the model structure

Usage

```
generate_data(
  M = 100,
  N = 61,
  I = 10,
  J = 12,
  K = c(4),
  Z.type = c("symmetric"),
  Z.contrast = "treatment",
  Z.formula = NULL,
```

```
  sigmax = 1,
  lambda = 1,
  yT = pi/4,
  yD = (3 * pi)/4,
  kappa_bnds = c(120, 300),
  priors = "default",
  gfunction = c("logistic", "gompertz"),
  ...
)
```

Arguments

M	(integer) number of simulated datasets
N	(integer) length of the Y-trajectories
I	(integer) number of individuals
J	(integer) number of trials
K	(array of integers) list of length Q of the number of levels for each categorical variable
Z.type	(array of characters) list of length Q of the methods (symmetric or random) to generate the matrix (see generate_Z)
Z.contrast	(character) type of contrasts (default: treatment) for the model matrix Z (see model.matrix)
Z.formula	(character) a formula of the contrasts for the model matrix Z (see model.matrix)
sigmax	(numeric) fixed value for the model parameter sigmax
lambda	(numeric) fixed value for the model parameter lambda
yT	(numeric) position in angles of the target
yD	(numeric) position in angles of the distractor
kappa_bnds	(array) array containing the lower and upper bounds for the kappa parameter (default = c(120,300))
priors	(list) a list of arguments specifying priors for each parameter involved in the model (see check_prior). If priors="default" then pre-defined priors will be used.
gfunction	(character) type of link function between latent states and observed data: 'logistic', 'gompertz' (default = 'logistic').
...	other stan arguments (e.g., 'init', 'algorithm', 'sample_file'. See sampling)

Details

The function generates simulated datasets via Stan according to the model structure.

Value

a datalist containing simulated data and parameters

Examples

```
## Not run:
## Generate mouse-tracking data for an univariate experimental design
## with K = 3 categorical levels, J = 30 trials, I = 8 subjects
X1 <- generate_data(I=5,J=12,K=3,Z.formula="~Z1",M=50)

## Generate mouse-tracking data for an univariate experimental design
## by varying priors of parameters
priors_list = list("normal(0,1)T(0,Inf)","normal(0,1)","normal(-2,0.5)")
X1 <- generate_data(I=5,J=12,K=3,Z.formula="~Z1",M=50,priors=priors_list)

## Generate mouse-tracking data with two experimental factors Z1 and Z2, J = 9 trials,
## K_Z1 = 3, K_Z2 = 3, I = 5 subjects
X2 <- generate_data(I=5,J=9,K=c(3,3),Z.formula="~Z1*Z2",
Z.type=c("symmetric","random"),M=50) # design with interaction

## End(Not run)
```

`generate_design` *Generate the design of a mouse-tracking experiment*

Description

Generate the design of a mouse-tracking experiment

Usage

```
generate_design(I = 10, J = 12, K = c(4), Z.type = c("symmetric"))
```

Arguments

`I`	(integer) number of individuals
`J`	(integer) number of trials
`K`	(list of integers) list of length Q of the number of levels for each categorical variable
`Z.type`	(list of characters) list of length Q of the methods (symmetric or random) to generate the matrix (see `generate_Z`)

Details

The function generates a dataframe containing the experimental design of a mouse-tracking study. The design is of the order (sbj,trial,variable1,...variableQ), where variable1,...,variableQ are Q categorical variables each with K_1,...,K_Q levels. The levels are codified using hundreds. This is an internal function, generally not to be called by the user.

Value

a dataframe of the order (sbj,trial,variable1,...variableQ)

Examples

```
## Generate a design with Q = 2 categorical variables:
## the first variable has K = 4 levels generated via symmetric method
## the second variable has K = 3 levels generated via random method.
X <- generate_design(I = 10, J = 12, K = c(4,3), Z.type = c("symmetric","random"))
print(X)
```

`generate_Z` *Generate a row-wise stacked boolean partition matrix of JI rows and K columns*

Description

Generate a row-wise stacked boolean partition matrix of JI rows and K columns

Usage

```
generate_Z(I, J, K, type = c("symmetric", "random"))
```

Arguments

`I`	(integer) number of individuals
`J`	(integer) number of trials
`K`	(integer) number of levels for a categorical variables
`type`	(character) method to generate the matrix: symmetric (default) or random

Details

The function generates a (JI x K) boolean partition matrix for I individuals, J stimuli and K categories. Note that J and K must be chosen so that J This is an internal function, generally not to be called by the user.

Value

a (JI x K) boolean matrix

Examples

```
Z <- generate_Z(I = 2,J = 12,K = 4,type="symmetric")
print(Z)
```

`language` *Mouse-tracking experiment of a lexical decision task*

Description

This dataset contains a subset of data originally presented in Barca & Pezzullo (2012). In this task participants see a printed stimulus on the screen (e.g., water) and are requested to perform a dichotomous choice task where the stimulus can be classified as word or non-word. The experimental variable is the *stimulus type* with four categorical levels (i.e., high-frequency word, low-frequency word, pseudowords, and strings of letters). Participants have to classify each stimulus as belonging to word or non-word categories.

The dataset contains five participants (I=5), each measured along three trials, one categorical variable (Q=1) with four levels (K=4). The total number of trials is J=12. Mouse-tracking trajectories have previously been pre-processed with N=101 timesteps, translated into the first quadrant, and rotated so that the Target point (y_T) is always on the right-side.

Usage

```
language
```

Format

A long-format dataframe of 6060 observations containing information on the following variables.

sbj The ID number of participants

condition A factor of levels `HF`, `LF`, `PW`, `NW` indicating the type of stimulus

timestep The ID number of the recorded x-y trajectories

x The recorded x-trajectories

y The recorded y-trajectories

trial The ID number of trials

Source

Barca, L., & Pezzulo, G. (2012). Unfolding visual lexical decision in time. *PloS one*, 7(4), e35932.

prepare_data *Prepare mouse-tracking trajectories for state-space modeling via Stan*

Description

Prepare mouse-tracking trajectories for state-space modeling via Stan

Usage

```
prepare_data(
  X = NULL,
  preprocess = TRUE,
  N = 61,
  Z.formula = NULL,
  Z.contrast = "treatment",
  yT = "AUTO",
  yD = "AUTO"
)
```

Arguments

X	(dataframe) a data frame of x-y trajectories and experimental design (see Details)
preprocess	(boolean) indicates whether x-y trajectories should be pre-processed (default preprocess=TRUE)
N	(integer) number of timesteps for trajectory normalization (default N=61)
Z.formula	(character) a formula of the contrasts for the model matrix Z (see model.matrix)
Z.contrast	(character) type of contrasts (default: treatment) for the model matrix Z (see model.matrix)
yT	(numeric) position in angles of the target. The default option yT="AUTO" will automatically determine the target position from the observed data
yD	(numeric) position in angles of the distractor. The default option yD="AUTO" will automatically determine the target position from the observed data

Details

The function prepares the mouse-tracking trajectories to be modeled for the state-space analysis. It automatically processes trajectories according to time-normalization, translation, and atan2 conversion. Users can skip pre-processing by setting preprocess=FALSE.

The input dataframe X needs to be organized using the long format with information being organized as nested. In particular, X must contains the following variables:

sbj The ID number of participants

trial The ID number of trials

factors 1,...,Q factors for the categorical variables of the design. They may have different levels.

timestep The ID number of the recorded x-y trajectories

x The recorded x-trajectories associated to trials and experimental levels

y The recorded y-trajectories associated to trials and experimental levels

See `language` and `congruency` as examples of datasets format required by **ssMousetrack** package.

Value

a list containing (i) the new dataframe of the pre-processed dataset (`X_processed`) and (ii) the needed data for `run_ssm`

Examples

```
data(congruency)
dataout <- prepare_data(X = congruency,preprocess = TRUE,Z.formula = "~congruency*plausibility")
str(dataout)
```

`run_ssm` *State-space modeling of mouse-tracking trajectories via Stan*

Description

State-space modeling of mouse-tracking trajectories via Stan

Usage

```
run_ssm(
  N,
  I,
  J,
  Y = NULL,
  D = NULL,
  Z = NULL,
  sigmax = 1,
  lambda = 1,
  y_T = pi/4,
  y_D = (3 * pi)/4,
  priors = "default",
  gfunction = c("logistic", "gompertz"),
  kappa_bnds = c(5, 300),
  nchains = 1,
  niter = 2000,
  nwarmup = 500,
  ncores = "AUTO",
  stan_object = FALSE,
  ...
)
```

Arguments

N	(integer) length of the Y-trajectories
I	(integer) number of individuals
J	(integer) number of trials
Y	(matrix) N x JI matrix of observed trajectories
D	(matrix) N x JI matrix of delta values for the observed trajectories
Z	(matrix) matrix of contrasts associated to the experimental design (see generate_design)
sigmax	(numeric) fixed value for the model parameter sigmax
lambda	(numeric) fixed value for the model parameter lambda
y_T	(numeric) position in angles of the target
y_D	(numeric) position in angles of the distractor
priors	(list) a list of arguments specifying priors for each parameter involved in the model (see check_prior). If priors="default" then pre-defined tpriors will be used.
gfunction	(character) type of link function between latent states and observed data: 'logistic', 'gompertz' (default = 'logistic').
kappa_bnds	(array) array containing the lower and upper bounds for the kappa parameter (default = c(5,300))
nchains	(integer) number of chains for the MCMC algorithm
niter	(integer) number of iterations for each chain
nwarmup	(integer) number of warmup/burnin iterations per chain
ncores	(integer) number of cores to use when executing the chains in parallel. The default option ncores="AUTO" will automatically determine the number of cores via the parallel package
stan_object	(boolean) if stan_object=TRUE, the object of S4 class stanfit representing the fitted results will be saved as stan_object.rda
...	other stan arguments (e.g., 'init', 'algorithm', 'sample_file'. See sampling)

Details

The function drawns samples from the posterior distribution of the model parameters. Note that, the current version of **ssMousetrack** package requires the number of stimuli J to be the same over the subjects $i = 1, ..., I$.

Value

a datalist containing the posterior samples for the model parameters along with the main Stan output

Examples

```
## Not run:
## Fit a state-space model using simulated data
# Generate mouse-tracking data for an univariate experimental design with K = 3 categorical levels,
# J = 12 trials, I = 5 subjects
X1 <- generate_data(I=5,J=12,K=3,Z.formula="~Z1")
iid <- 23 # keep just one dataset from the simulated set of datasets
# Run the state-space model on the chosen dataset
X1_fit <- run_ssm(N = X1$N,I = X1$I,J = X1$J,Y = X1$data$Y[iid,,],D = X1$data$D[iid,,],
Z = X1$data$Z)

## Fit a state-space model using the experimental dataset language
# The dataset is ready to be used and it does not need to be pre-processed (preprocess=FALSE).
# In this case, the function prepare_data just computes the observed radians from
# the x-y trajectories
X2 <- prepare_data(X = language, preprocess = FALSE, Z.formula = "~condition")
# Run the state-space model on the chosen dataset
X2_fit <- run_ssm(N = X2$N,I = X2$I,J = X2$J,Y = X2$Y,D = X2$D,Z = X2$Z,
niter=5000,nchains=2)

## Fit a state-space model using the experimental dataset congruency
# The dataset needs to be pre-processed (preprocess=TRUE)
X3 <- prepare_data(X = congruency, preprocess = TRUE,
Z.formula = "~congruency+plausibility") # additive design
# Define priors of the model parameters
KK <- dim(X3$Z)[2] # number of model parameters implied by the design matrix Z
priors_list <- list("lognormal(1,0.5)","pareto(3,5.25)","normal(0,2.5)")
# note that length(priors_list) = KK
# Run the state-space model on the chosen dataset
X3_fit <- run_ssm(N = X3$N,I = X3$I,J = X3$J,Y = X3$Y,D = X3$D,Z = X3$Z,
niter=10000,nwarmup=3500,priors=priors_list,nchains=4)

## End(Not run)
```

Index

∗ **datasets**
congruency, 6
language, 12

check_prior, 4, *9*, *15*
compute_D, 6
congruency, 6, *14*

dtw, *8*

evaluate_ssm, 7

generate_data, 8
generate_design, 10, *15*
generate_Z, *9, 10*, 11

language, 12, *14*

model.matrix, *9*, *13*

prepare_data, 13

run_ssm, *5*, *7*, 14, *14*

sampling, *9*, *15*
ssMousetrack *(*ssMousetrack-package*)*, 2
ssMousetrack-package, 2