Package 'bspam'

November 12, 2026

Type Package

Encoding UTF-8

Title Speed-Accuracy Psychometric Modeling for Binomial Count Outcome Data

Version 1.2.6

Maintainer Kuo Wang <wangkuo@nifty.com>

Description Provides functions to fit speed-accuracy psychometric models for binomial count outcome data. The models jointly represent accuracy and response time and can be used for model-based calibration and scoring of oral reading fluency (ORF) assessment data. The methods are based on Potgieter, Kamata and Kara (2017) <doi:10.48550/arXiv.1705.10446> and Kara, Kamata, Potgieter and Nese (2020) <doi:10.1177/0013164419900208>.

License GPL (>= 3)

Depends R (>= 3.5.0)

Imports tibble, rootSolve, tidyverse, tidyr, magrittr, doParallel, foreach, stringr, futile.logger, mvtnorm, nleqslv, tryCatchLog, parallel, miscTools, psych, dplyr, plotly, Matrix, statmod

Suggests knitr, rstan, BH, RcppEigen, runjags, rjags, rmarkdown, testthat (>= 3.0.0)

URL https://kamataak.github.io/bspam/, https://github.com/kamataak/bspam

BugReports https://github.com/kamataak/bspam/issues

RoxygenNote 7.3.3

LazyData yes

Config/testthat/edition 3

NeedsCompilation no

VignetteBuilder knitr

Author Kuo Wang [aut, cre] (ORCID: <https://orcid.org/0000-0002-4925-0591>),
Akihito Kamata [aut, ctb] (ORCID:
<https://orcid.org/0000-0001-9570-1464>),
Cornelis J. Potgieter [aut] (ORCID:
<https://orcid.org/0000-0002-1995-6817>),
Joseph F. T. Nese [aut] (ORCID:
<https://orcid.org/0000-0002-9878-7395>),
Yusuf Kara [aut] (ORCID: <https://orcid.org/0000-0003-0691-0630>),
Sarunya Somsong [aut] (ORCID: <https://orcid.org/0000-0002-4974-7346>),
Xin Qiao [aut] (ORCID: <https://orcid.org/0000-0002-3248-7859>)

Repository CRAN

Date/Publication 2026-11-12 15:40:02 UTC

Contents

agg.word . . . 3
desc.data . . . 4
fit.model . . . 5
fit.model.testlet . . . 8
get.cases . . . 10
passage.calib.bayes . . . 10
passage.calib.mcem . . . 11
passage2 . . . 12
plot.information . . . 13
plot.person . . . 14
plot.task . . . 15
plot.wcpm . . . 16
prep . . . 17
run.mcem . . . 19
run.scoring . . . 21
scoring . . . 23
sentence.cens.high . . . 26
sentence.cens.low . . . 27
sentence.level.data . . . 28
summary.bootstrap . . . 29
summary.fit.model . . . 30
summary.fit.model.testlet . . . 31
summary.scoring . . . 31

Index **33**

agg.word *Aggregate word-level ORF data*

Description

Aggregates word-level response and timing data to the sentence or passage level. The function calculates words read correctly, elapsed reading time, the number of words, and observed words-correct-per-minute (WCPM).

Usage

```
agg.word(
  data,
  agg.level = "sentence",
  person.id = "",
  passage.id = "",
  word.pos.sen = "",
  word.pos.pas = "",
  sen.pos.pas = "",
  start.time = "",
  end.time = "",
  time.scale = "centi",
  score = ""
)
```

Arguments

data	A data frame containing word-level response and timing data.
agg.level	Character string specifying the aggregation level. Available options are "sentence" and "passage". Default is "sentence".
person.id	Quoted variable name identifying persons.
passage.id	Quoted variable name identifying passages.
word.pos.sen	Quoted variable name giving each word's position within a sentence. Required when agg.level = "sentence".
word.pos.pas	Quoted variable name giving each word's position within a passage. Required when agg.level = "passage".
sen.pos.pas	Quoted variable name giving each sentence's position within a passage. Required when agg.level = "sentence".
start.time	Quoted variable name giving the word-level start time.
end.time	Quoted variable name giving the word-level end time.
time.scale	Character string indicating the unit of the input start and end times. Use "centi" for hundredths of a second or "sec" for seconds. Default is "centi".
score	Quoted variable name giving the word-level accuracy indicator, coded 1 for a correctly read word and 0 otherwise.

Details

When `agg.level = "sentence"`, the output contains one row per person-by-passage-by-sentence combination. When `agg.level = "passage"`, the output contains one row per person-by-passage combination.

Regardless of the input `time.scale`, the returned elapsed-time variable is expressed in seconds. Observed WCPM is calculated as the number of words read correctly divided by elapsed time in seconds, multiplied by 60.

Value

A data frame containing the aggregated response information. Sentence-level output includes `wrc`, `secs`, `nwords.sen`, and `wcpm.sen`. Passage-level output includes `wrc`, `secs`, `nwords.pas`, and `wcpm.pas`.

`desc.data` *Summarize response data*

Description

Computes descriptive summaries of task-level or sub-task-level response data. Summaries can be produced at the person level or aggregated across the sample. For oral reading fluency (ORF) data, observed WCPM summaries can also be included.

Usage

```
desc.data(
  data = NULL,
  person.id = NULL,
  task.id = NULL,
  max.counts = NULL,
  obs.counts = NULL,
  time = NULL,
  sub.task.id = NULL,
  desc.level = "sample",
  verbose = TRUE,
  type = "general",
  testlet = FALSE
)
```

Arguments

`data`	A data frame containing response data.
`person.id`	Quoted variable name identifying persons.
`task.id`	Quoted variable name identifying tasks. In the ORF assessment context, this is the passage identifier.

max.counts	Quoted variable name giving the maximum possible count for each task or sub-task. In the ORF assessment context, this is the number of words in the passage or sentence.
obs.counts	Quoted variable name giving the observed number of successful outcomes. In the ORF assessment context, this is the number of words read correctly.
time	Quoted variable name giving completion time, in seconds.
sub.task.id	Quoted variable name identifying sub-tasks. In the ORF assessment context, this is the sentence identifier. This argument is required when `testlet = TRUE`.
desc.level	Character string specifying the level of the descriptive output. Use `"sample"` for a sample-level summary or `"person"` for person-level summaries. Default is `"sample"`.
verbose	Logical controlling the amount of sample-level output. When `TRUE`, the full result from `psych::describe()` is returned. When `FALSE`, a reduced set of summary statistics is returned. Default is `TRUE`.
type	Character string specifying the output type. If `"general"`, general count and time summaries are produced. If `"orf"`, observed WCPM summaries are additionally produced. Default is `"general"`.
testlet	Logical. If `FALSE`, task-level data are summarized. If `TRUE`, sub-task-level data are summarized while retaining their task structure. Default is `FALSE`.

Details

For task-level data, person-level summaries include the number of observed tasks and totals for maximum counts, observed counts, and time. With `type = "orf"`, the function also calculates passage-level observed WCPM, the average passage WCPM for each person, and an overall observed WCPM based on total correct words and total time.

For sub-task-level data, person-level summaries additionally include the number of distinct sub-tasks. With `type = "orf"`, sentence-level observations are first aggregated within passages before passage-level and overall observed WCPM summaries are calculated.

Value

If `desc.level = "person"`, returns a person-level data frame. If `desc.level = "sample"`, returns descriptive statistics generated by `psych::describe()`; when `verbose = FALSE`, only the variables, sample size, mean, standard deviation, minimum, and maximum are retained.

---

fit.model *Fit the speed-accuracy psychometric model*

Description

This is the main interface function for estimating parameters of the speed-accuracy psychometric model for repeatedly measured count outcome data. The model jointly describes task accuracy and completion time. Model parameters can be estimated using the Monte Carlo EM (MCEM) algorithm described in Potgieter et al. (2017) or a fully Bayesian approach described in Kara et al. (2020).

Usage

```
fit.model(
  data = NA,
  person.id = "",
  task.id = "",
  sub.task.id = "",
  max.counts = "",
  obs.counts = "",
  time = "",
  k.in = 5,
  reps.in = 50,
  ests.in = NA,
  est = "mcem",
  se = "none",
  verbose = FALSE,
  testlet = FALSE
)
```

Arguments

`data`	A data frame containing the response data or an object created by `prep`. When a prepared-data object is supplied, the column-name arguments `person.id`, `task.id`, `max.counts`, `obs.counts`, and `time` should be left unspecified.
`person.id`	Quoted variable name in `data` identifying persons.
`task.id`	Quoted variable name in `data` identifying tasks. In the ORF assessment context, this is the passage identifier.
`sub.task.id`	Quoted variable name in `data` identifying sub-tasks. In the ORF assessment context, this is the sentence identifier. This argument is required when `testlet = TRUE`.
`max.counts`	Quoted variable name in `data` giving the maximum possible count for each task. In the ORF assessment context, this is the number of words in the passage or sentence.
`obs.counts`	Quoted variable name in `data` giving the observed number of successful outcomes for each task. In the ORF assessment context, this is the number of words read correctly.
`time`	Quoted variable name in `data` giving the completion time, in seconds, for each task.
`k.in`	Numeric value or vector indicating the number of Monte Carlo imputations used in the MCEM algorithm. Default is `5`.
`reps.in`	Numeric value or vector indicating the number of Monte Carlo EM iterations. Default is `50`.
`ests.in`	An optional list containing initial values for the model parameters. If initial values are not supplied, they are generated internally using the method-of-moments procedure.
`est`	Character string specifying the estimation method. Available options are `"mcem"` and `"bayes"`. Default is `"mcem"`.

se	Character string specifying the method used to obtain standard errors for MCEM parameter estimates. Available options are "none", "analytic", and "bootstrap". Default is "none".
verbose	Logical. If TRUE, a summary of the fitted model is printed. Default is FALSE.
testlet	Logical. If TRUE, a sub-task-level testlet model is fitted using fit.model.testlet. In the ORF assessment context, this models sentences nested within passages. Default is FALSE.

Details

In the oral reading fluency (ORF) assessment context, tasks correspond to reading passages, accuracy is represented by the number of words read correctly, and speed is represented by reading time.

For task-level analyses (testlet = FALSE), fit.model() estimates task parameters for the accuracy and time components together with population-level parameters describing the latent speed-accuracy distribution.

The function accepts either raw long-format data or data previously prepared with prep. When raw data are supplied, the relevant columns are identified through the column-name arguments. When a prep object is supplied, the standardized data stored in that object are used directly.

With est = "mcem", model parameters are estimated by Monte Carlo EM. Standard errors may optionally be obtained using the method specified by se. With est = "bayes", model parameters are estimated using the Bayesian implementation.

When testlet = TRUE, the function fits the sentence-level testlet model through fit.model.testlet rather than the task-level model.

Value

Invisibly returns a fitted model object. For task-level analyses, the returned object has class "fit.model" and contains task.param, with estimated task parameters, and hyper.param, with estimated population-level parameters. Standard-error columns are included when applicable. When testlet = TRUE, an object of class "fit.model.testlet" is returned containing the estimated sub-task and testlet-model parameters.

References

Potgieter, C. J., Kamata, A., & Kara, Y. (2017). An EM algorithm for estimating an oral reading speed and accuracy model. *arXiv preprint arXiv:1705.10446*. https://arxiv.org/abs/1705.10446

Kara, Y., Kamata, A., Potgieter, C. J., & Nese, J. F. T. (2020). Estimating model-based oral reading fluency: A Bayesian approach with a binomial-lognormal joint latent model. *Educational and Psychological Measurement, 80*(5), 847-869. doi:10.1177/0013164419900208

See Also

prep for preparing response data, scoring for estimating person scores, and fit.model.testlet for fitting the sub-task-level testlet model.

Examples

```
MCEM_run <- fit.model(
  data = passage2,
  person.id = "id.student",
  task.id = "id.passage",
  max.counts = "numwords.pass",
  obs.counts = "wrc",
  time = "sec",
  k.in = 5,
  reps.in = 50,
  est = "mcem"
)
```

`fit.model.testlet` *Fit the speed-accuracy testlet model*

Description

Fits a sub-task-level speed-accuracy psychometric model that accounts for the nesting of sub-tasks within tasks. In oral reading fluency (ORF) applications, sub-tasks correspond to sentences and tasks correspond to passages.

Usage

```
fit.model.testlet(
  data = NULL,
  person.id = "",
  sub.task.id = "",
  obs.counts = "",
  time = "",
  task.id = "",
  max.counts = ""
)
```

Arguments

`data`	A data frame containing sub-task-level response data.
`person.id`	Quoted variable name in `data` identifying persons.
`sub.task.id`	Quoted variable name in `data` identifying sub-tasks. In the ORF assessment context, this is the sentence identifier.
`obs.counts`	Quoted variable name in `data` giving the observed number of successful outcomes for each sub-task. In the ORF assessment context, this is the number of words read correctly in the sentence.
`time`	Quoted variable name in `data` giving the completion time, in seconds, for each sub-task.

task.id	Quoted variable name in data identifying the task to which each sub-task belongs. In the ORF assessment context, this is the passage identifier.
max.counts	Quoted variable name in data giving the maximum possible count for each sub-task. In the ORF assessment context, this is the number of words in the sentence.

Details

The model jointly describes accuracy and completion time at the sentence level while incorporating passage-level testlet effects. It estimates sentence parameters together with population-level parameters describing the latent accuracy, speed, and testlet components.

fit.model.testlet() is the sub-task-level calibration routine used by fit.model when testlet = TRUE. The input data are converted to person-by-sub-task count and time structures, with completion times expressed as natural-log time standardized to 10 units.

The model includes sentence-level accuracy parameters a and b, sentence-level time parameters alpha and beta, and passage-level testlet effects for the accuracy and speed components. Population-level parameters include sigma, gamma1, gamma2, rho.theta, and rho.testlet.

Each task must be represented by observations from at least two persons. If a task is represented by only one person, or if required column names are not supplied, the function stops the calibration procedure and returns without a fitted model.

Value

An object of class "fit.model.testlet" containing:

task.param A data frame containing the estimated sub-task parameters a, b, alpha, and beta, together with task.id and sub.task.id.

hyper.param A data frame containing the estimated population-level parameters sigma, gamma1, gamma2, rho.theta, and rho.testlet.

Y The person-by-sub-task observed count matrix used for calibration.

logT10 The person-by-sub-task matrix of natural-log completion times standardized to 10 units.

N The vector of maximum counts for the sub-tasks.

See Also

fit.model for the main model-fitting interface and prep for preparing task- or sub-task-level response data.

Examples

```
fit_testlet <- fit.model.testlet(
  data = sentence.level.data,
  person.id = "id.student",
  sub.task.id = "id.sentence",
  obs.counts = "wrc",
  time = "sec",
  task.id = "id.passage",
  max.counts = "numwords.sent"
)
```

get.cases *Create person-by-occasion case identifiers*

Description

Creates unique case identifiers by combining the `person.id` and `occasion` variables in prepared response data. These identifiers are used by scoring functions to identify person-by-occasion records.

Usage

```
get.cases(data)
```

Arguments

`data`	A data frame containing `person.id` and `occasion` variables.

Value

Invisibly returns a one-column data frame containing unique case identifiers in the variable `cases`.

passage.calib.bayes *Task calibration example output object by Bayes*

Description

This is an example calibration output object obtained using the Bayesian estimator in `fit.model`, with `est = "bayes"`. The object contains calibrated oral reading fluency passage parameters together with population-level hyperparameters.

Usage

```
passage.calib.bayes
```

Format

A list of two elements: `$pass.param` is a data frame with 150 rows and 10 variables, and `$hyper.param` is a data frame with 1 row and 4 variables.

`$pass.param`

`a` parameter controlling binomial success probabilities

`b` parameter controlling binomial success probabilities

`alpha` parameter controlling reading times

`beta` parameter controlling reading times

`se_a` standard error of a

`se_b` standard error of b

`se_alpha` standard error of alpha

`se_beta` standard error of beta

`passage.id` passage ID

`nwords.p` total number of words in the passage

`$hyper.param`

`vartau` variance of latent speed ability tau

`rho` correlation between the two latent variables

`se_vartau` standard error of vartau

`se_rho` standard error of rho

Source

https://jnese.github.io/core-blog/

References

Nese, J. F. T. & Kamata, A. (2014-2018). Measuring Oral Reading Fluency: Computerized Oral Reading Evaluation (Project No. R305A140203) [Grant]. Institute of Education Sciences, U.S. Department of Education.

`passage.calib.mcem` *Task calibration example output object by MCEM*

Description

This is an example calibration output object obtained using the MCEM estimator in `fit.model`. The object contains calibrated oral reading fluency passage parameters together with population-level hyperparameters.

Usage

```
passage.calib.mcem
```

Format

A list of two elements: `$pass.param` is a data frame with 150 rows and 10 variables, and `$hyper.param` is a data frame with 1 row and 4 variables.

`$pass.param`

`a` parameter controlling binomial success probabilities

`b` parameter controlling binomial success probabilities

`alpha` parameter controlling reading times

`beta` parameter controlling reading times

`se_a` standard error of a

`se_b` standard error of b

`se_alpha` standard error of alpha

`se_beta` standard error of beta

`passage.id` passage ID

`nwords.p` total number of words in the passage

`$hyper.param`

`vartau` variance of latent speed ability tau

`rho` correlation between the two latent variables

`se_vartau` standard error of vartau

`se_rho` standard error of rho

Source

https://jnese.github.io/core-blog/

References

Nese, J. F. T. & Kamata, A. (2014-2018). Measuring Oral Reading Fluency: Computerized Oral Reading Evaluation (Project No. R305A140203) [Grant]. Institute of Education Sciences, U.S. Department of Education.

---

passage2 *Passage-level Oral Reading Fluency assessment data set*

Description

This is an example data set. It is a passage-level Oral Reading Fluency assessment data set for 85 students who were assigned to read 2 to 12 passages among the same 12 passages. The data is a small subset of the data collected by Nese and Kamata (2014-2018).

Usage

```
passage2
```

Format

A data frame with 847 rows and 7 variables.

`id.student` unique student identifier

`occasion` identifier for longitudinal assessment occasions; here a triannual assessment administered in the fall, winter, and spring of a school year

`grade` student grade level

`id.passage` unique passage identifier

`numwords.pass` total number of words in the passage

`wrc` words read correct

`sec` seconds to read the passage

Source

https://jnese.github.io/core-blog/

References

Nese, J. F. T. & Kamata, A. (2014-2018). Measuring Oral Reading Fluency: Computerized Oral Reading Evaluation (Project No. R305A140203) [Grant]. Institute of Education Sciences, U.S. Department of Education.

`plot.information` *Plot model-based WCPM information*

Description

Creates an interactive three-dimensional visualization of model-based WCPM information over a grid of latent accuracy and speed values.

Usage

```
## S3 method for class 'information'
plot(x, ...)
```

Arguments

`x`	A calibrated model object containing `task.param` and `hyper.param`.
`...`	Additional arguments reserved for the plotting function.

Details

The current implementation visualizes WCPM information for the first calibrated task only. The axes are `theta`, `tau`, and WCPM information.

This function is intended to be called directly as `plot.information(...)`. It is not designed for automatic dispatch through `plot()`.

Value

A three-dimensional plotly scatter plot.

Examples

```
data("passage.calib.mcem")
plot.information(passage.calib.mcem)
```

---

plot.person *Plot estimated person scores*

Description

Creates an interactive plot of person-level scores from a "scoring" object.

Usage

```
## S3 method for class 'person'
plot(x, person = NULL, parameter, show.se = TRUE, sort = FALSE, ...)
```

Arguments

x	A scoring object of class "scoring".
person	Optional vector of person identifiers. If NULL, all persons are included.
parameter	Character vector specifying one or two score types: "theta", "tau", or "wcpm".
show.se	Logical. If TRUE, approximately 95 percent intervals based on estimate plus or minus 1.96 standard errors are shown.
sort	Logical. For a single score, if TRUE, persons are sorted by the estimated score.
...	Additional arguments reserved for the plotting function.

Details

One score produces a person-by-score plot; two scores produce a scatter plot. For "wcpm", the observed WCPM column is excluded and the model-based WCPM estimate is used.

This function is intended to be called directly as plot.person(...). It is not designed for automatic dispatch through plot().

Value

A plotly object.

Examples

```
scores <- data.frame(
  person.id = c("1", "2"),
  theta.map = c(-0.5, 0.6),
  tau.map = c(-0.2, 0.3),
  se.theta.map = c(0.2, 0.2),
  se.tau.map = c(0.1, 0.1)
)
class(scores) <- c("scoring", "data.frame")
plot.person(scores, parameter = "theta")
plot.person(scores, parameter = c("theta", "tau"))
```

`plot.task` *Plot calibrated task parameters*

Description

Creates an interactive plot of calibrated task parameters from a `"fit.model"` or `"fit.model.testlet"` object.

Usage

```
## S3 method for class 'task'
plot(x, task = NULL, parameter, sort = FALSE, ...)
```

Arguments

`x`	A fitted calibration object of class `"fit.model"` or `"fit.model.testlet"`.
`task`	Optional vector of task identifiers to include. If `NULL`, all available tasks are included.
`parameter`	Character vector specifying one or two parameters: `"a"`, `"b"`, `"alpha"`, or `"beta"`.
`sort`	Logical. For a single parameter, if `TRUE`, tasks are ordered by the parameter estimate. Default is `FALSE`.
`...`	Additional arguments reserved for the plotting function.

Details

With one parameter, a horizontal bar plot is returned. With two parameters, a scatter plot is returned with one point per selected task.

This function is intended to be called directly as `plot.task(...)`. It is not designed for automatic dispatch through `plot()`. For objects of class `"fit.model.testlet"`, each row of `task.param` corresponds to a sub-task, and the plot displays `sub.task.id` on the vertical axis. The `task` argument still filters by `task.id`.

Value

A plotly object.

Examples

```
data("passage.calib.mcem")
plot.task(passage.calib.mcem, parameter = "a")
plot.task(passage.calib.mcem, parameter = c("a", "b"))
```

---

plot.wcpm *Plot observed versus model-based WCPM*

Description

Creates an interactive scatter plot comparing observed WCPM with model-based WCPM estimates from a "scoring" object produced with type = "orf".

Usage

```
## S3 method for class 'wcpm'
plot(x, person = NULL, show.se = TRUE, show.abline = TRUE, ...)
```

Arguments

x	A scoring object of class "scoring" containing wcpm.obs and a model-based WCPM estimate.
person	Optional vector of person identifiers. If NULL, all persons are included.
show.se	Logical. If TRUE, vertical error bars show approximately 95 percent intervals based on 1.96 standard errors.
show.abline	Logical. If TRUE, the identity line y = x is added.
...	Additional arguments reserved for the plotting function.

Details

The x-axis is observed WCPM and the y-axis is model-based WCPM. The function requires scoring results generated with type = "orf".

This function is intended to be called directly as plot.wcpm(...). It is not designed for automatic dispatch through plot().

Value

A plotly scatter plot.

Examples

```
scores <- data.frame(
  person.id = c("1", "2"),
  wcpm.obs = c(100, 120),
  wcpm.map = c(105, 118),
  se.wcpm.map = c(5, 6)
)
class(scores) <- c("scoring", "data.frame")
plot.wcpm(scores)
```

`prep` *Prepare response data for bspam analyses*

Description

Prepares long-format response data for model fitting and scoring with **bspam**. The function standardizes the required variable names and constructs the data structures used internally by the package.

Usage

```
prep(
  data = data,
  person.id = "",
  task.id = "",
  sub.task.id = "",
  occasion = "",
  group = "",
  max.counts = "",
  obs.counts = "",
  time = "",
  cens = "",
  sentence_level = FALSE
)
```

Arguments

`data`	A data frame containing the response data in long format.
`person.id`	Quoted variable name in `data` identifying persons.
`task.id`	Quoted variable name in `data` identifying tasks. In the ORF assessment context, this is the passage identifier.
`sub.task.id`	Quoted variable name in `data` identifying sub-tasks. In the ORF assessment context, this is the sentence identifier. This argument is required when `sentence_level = TRUE`.

occasion	Quoted variable name in data identifying measurement occasions. If omitted, all observations are assigned to a single occasion.
group	Quoted variable name in data identifying groups. If omitted, all observations are assigned to a single group.
max.counts	Quoted variable name in data giving the maximum possible count for each task or sub-task. In the ORF assessment context, this is the number of words in the passage or sentence.
obs.counts	Quoted variable name in data giving the observed number of successful outcomes. In the ORF assessment context, this is the number of words read correctly.
time	Quoted variable name in data giving the observed completion time, in seconds.
cens	Optional quoted variable name in data containing censoring indicators. For sentence-level data, if this argument is omitted, all observations are treated as fully observed and a censoring indicator equal to 0 is created internally.
sentence_level	Logical. If FALSE, task-level data are prepared. If TRUE, sub-task-level data are prepared. Default is FALSE.

Details

Both task-level and sub-task-level data are supported. In oral reading fluency (ORF) applications, task-level data correspond to passage-level observations, whereas sub-task-level data correspond to sentence-level observations nested within passages.

For task-level data, prep() reshapes the observed count and time variables into person-by-task structures used by the task-level speed-accuracy model. Reading or completion times are transformed to the natural logarithm of time standardized to 10 task units. In ORF applications, this corresponds to log reading time per 10 words.

For sub-task-level data, the function preserves the task and sub-task structure in the standardized long-format data and constructs corresponding person-by-observation matrices for sentence-level analyses.

If occasion or group is not supplied, a constant value of 1 is created for that variable. For sentence-level data, if cens is not supplied, a censoring variable containing zeros is created, indicating that all observations are treated as fully observed.

Value

A prepared-data object containing two components:

data.long A standardized long-format data frame using the variable names expected internally by **bspam**.

data.wide A list of matrices and vectors used by the model-fitting and scoring functions.

For task-level data, the returned object has class "prepared.task". Its data.wide component contains Y, the observed count matrix; logT10, the matrix of log-transformed completion times standardized to 10 units; N, the vector of task maximum counts; and I, the number of tasks.

For sub-task-level data, the returned object has class "prepared.sub.task". Its data.wide component contains the corresponding count, time, and maximum-count structures used for sentence-level analyses.

See Also

fit.model for model calibration and scoring for person-level scoring.

Examples

```
prepared <- prep(
  data = passage2,
  person.id = "id.student",
  task.id = "id.passage",
  occasion = "occasion",
  group = "grade",
  max.counts = "numwords.pass",
  obs.counts = "wrc",
  time = "sec"
)
```

---

run.mcem *Fit the speed-accuracy model by Monte Carlo EM*

Description

Lower-level routine for estimating parameters of the task-level speed-accuracy psychometric model using the Monte Carlo EM (MCEM) algorithm described in Potgieter et al. (2017). Most users should use fit.model, which provides the main model-fitting interface.

Usage

```
run.mcem(
  Y,
  logT10,
  N,
  I,
  k.in = 5,
  reps.in = 50,
  ests.in = NA,
  verbose = FALSE
)
```

Arguments

Y	Numeric matrix of observed accuracy counts with persons in rows and tasks in columns. Missing values are allowed.
logT10	Numeric matrix of completion times with the same dimensions as Y. Values are natural-log completion times standardized to 10 task units. In oral reading fluency (ORF) applications, this is the natural logarithm of reading time per 10 words. Missing values are allowed.

N	Numeric vector giving the maximum possible count for each task. In the ORF assessment context, this is the number of words in each passage.
I	Numeric value giving the number of tasks, corresponding to the number of columns in Y and logT10.
k.in	Numeric value or vector specifying the number of Monte Carlo imputations used at each stage of the MCEM procedure. Default is 5.
reps.in	Numeric value or vector specifying the number of MCEM iterations associated with each value of k.in. Values of approximately 50 to 100 are generally recommended. Default is 50.
ests.in	Optional list containing initial values for the model parameters. If initial values are not supplied, they are generated internally using the method-of-moments procedure.
verbose	Logical. If TRUE, progress information is printed. Default is FALSE.

Details

The model jointly describes task accuracy and completion time through two correlated latent person variables. For each task, the count component is parameterized by accuracy parameters a and b, and the time component is parameterized by alpha and beta. The population-level parameters include vartau, the variance of the latent speed component, and rho, the correlation between the latent accuracy and speed components.

The MCEM algorithm alternates between Monte Carlo imputation of the latent person variables and updates of the task and population-level parameters. The arguments k.in and reps.in may be vectors, allowing the number of Monte Carlo imputations to change across stages of the algorithm.

This function expects the wide-format matrices used internally by **bspam**. Users with long-format response data should generally call fit.model directly or first prepare the data using prep.

Value

A list containing the estimated task and population-level parameters, including a, b, alpha, beta, vartau, and rho.

References

Potgieter, C. J., Kamata, A., & Kara, Y. (2017). An EM algorithm for estimating an oral reading speed and accuracy model. *arXiv preprint arXiv:1705.10446*. https://arxiv.org/abs/1705.10446

See Also

fit.model for the main model-fitting interface and prep for preparing response data.

`run.scoring` *Estimate person scores by likelihood-based methods*

Description

Lower-level routine for estimating person-level latent accuracy and speed scores from a calibrated task-level speed-accuracy model using likelihood-based approaches. The supported estimators are maximum likelihood estimation (MLE), maximum a posteriori estimation (MAP), and expected a posteriori estimation (EAP). Most users should use `scoring`, which provides the main scoring interface.

Usage

```
run.scoring(
  object,
  person.data,
  task.data,
  cases,
  perfect.cases,
  zero.cases,
  est = "map",
  lo = -12,
  hi = 12,
  q = 100,
  kappa = 1,
  external = NULL,
  type = NULL
)
```

Arguments

`object`	A fitted calibration object of class `"fit.model"`, typically obtained from `fit.model`.
`person.data`	A long-format data frame containing the standardized person response data used for scoring.
`task.data`	A data frame containing calibrated task parameters, including `a`, `b`, `alpha`, `beta`, `task.id`, and `max.counts`.
`cases`	A one-column data frame containing the person-by-occasion identifiers to be scored, typically created by `get.cases`.
`perfect.cases`	A data frame identifying person-by-occasion cases with perfect observed accuracy.
`zero.cases`	A data frame identifying person-by-occasion cases with zero observed correct counts.
`est`	Character string specifying the estimator. Available options are `"mle"`, `"map"`, and `"eap"`. Default is `"map"`.
`lo`	Numeric lower bound used in the numerical scoring procedures. Default is `-12`.

hi	Numeric upper bound used in the numerical scoring procedures. Default is 12.
q	Numeric value controlling the number of quadrature points used for EAP scoring. Default is 100.
kappa	Numeric tuning parameter retained for compatibility with the scoring workflow. Default is 1.
external	Optional vector of calibrated task identifiers used to calculate model-based outcomes on a common task set. If NULL, model-based outcomes are calculated using the tasks observed for each person.
type	Character string specifying the output type. If "general", latent scores and model-based count and time outcomes are returned without WCPM. If "orf", observed and model-based WCPM variables are additionally returned.

Details

The calibration parameters in object and task.data are treated as fixed while person-level latent accuracy and speed scores are estimated. The function computes observed count and time summaries for each person-by-occasion case and applies the estimator selected by est.

For MLE, the latent accuracy and speed components are estimated by likelihood maximization. For MAP, the joint posterior mode of the two latent components is obtained by a Newton-Raphson procedure. For EAP, numerical integration over a two-dimensional quadrature grid is used.

When external is supplied, the latent person scores are still based on the observed response data, but model-based count, time, and WCPM outcomes are calculated using the specified calibrated tasks rather than the tasks observed for that person.

Scoring is performed in parallel across person-by-occasion cases.

Value

Invisibly returns an object of class "scoring". The object contains one row per scored person-by-occasion case, with identifiers, descriptive observed-data summaries, estimated latent accuracy and speed scores, corresponding standard errors, and model-based outcomes. When type = "orf", observed and model-based WCPM variables are also included.

References

Potgieter, C. J., Qiao, X., Kamata, A., & Kara, Y. (2024). Likelihood-based estimation of model-derived oral reading fluency. *Journal of Educational Measurement, 61*(3), 542-559. doi:10.1111/jedm.12404

Qiao, X., Evudottir, E., Potgieter, C. J., & Kamata, A. (2026). Likelihood-based estimation of model-derived oral reading fluency under overdispersed count-time data. *Measurement: Interdisciplinary Research and Perspectives*. doi:10.1080/15366367.2026.2643183

See Also

scoring for the main scoring interface, fit.model for model calibration, and get.cases for constructing person-by-occasion identifiers.

`scoring` *Estimate person scores from a calibrated speed-accuracy model*

Description

This is the main interface function for estimating person-level scores from a calibrated speed-accuracy psychometric model. It estimates latent accuracy and speed scores using likelihood-based methods, including maximum likelihood estimation (MLE), maximum a posteriori estimation (MAP), and expected a posteriori estimation (EAP), or a fully Bayesian approach.

Usage

```
scoring(
  calib.data = NA,
  data = NA,
  person.id = "",
  task.id = "",
  sub.task.id = "",
  occasion = "",
  group = "",
  max.counts = "",
  obs.counts = "",
  time = "",
  cens = "",
  cases = NULL,
  est = "map",
  se = "analytic",
  failsafe = 0,
  bootstrap = 100,
  external = NULL,
  type = "general",
  censoring = FALSE,
  testlet = FALSE
)
```

Arguments

`calib.data`	A fitted calibration object obtained from `fit.model`. This should be an object of class `"fit.model"` or `"fit.model.testlet"`, depending on the scoring model being used.
`data`	A data frame containing response data or an object created by `prep`. When raw data are supplied, the relevant columns are identified using the column-name arguments below.
`person.id`	Quoted variable name in `data` identifying persons.
`task.id`	Quoted variable name in `data` identifying tasks. In the ORF assessment context, this is the passage identifier.

`sub.task.id` Quoted variable name in `data` identifying sub-tasks. In the ORF assessment context, this is the sentence identifier. This argument is required when `testlet = TRUE`.

`occasion` Quoted variable name in `data` identifying measurement occasions. If not supplied, a single occasion is assumed.

`group` Quoted variable name in `data` identifying groups. If not supplied, a single group is assumed.

`max.counts` Quoted variable name in `data` giving the maximum possible count for each task or sub-task. In the ORF assessment context, this is the number of words in the passage or sentence.

`obs.counts` Quoted variable name in `data` giving the observed number of successful outcomes. In the ORF assessment context, this is the number of words read correctly.

`time` Quoted variable name in `data` giving the observed completion time, in seconds.

`cens` Quoted variable name in `data` containing censoring indicators, where `1` indicates a censored observation and `0` indicates a fully observed response. This argument is used when `censoring = TRUE` or `testlet = TRUE`.

`cases` Optional set of person-by-occasion cases for which scores are requested. If `NULL`, scores are estimated for all available cases in `data`.

`est` Character string specifying the scoring estimator. Available options are `"mle"`, `"map"`, `"eap"`, and `"bayes"`. Default is `"map"`.

`se` Character string specifying the standard-error method for non-censored scoring. Available options are `"analytic"` and `"bootstrap"`. Default is `"analytic"`.

`failsafe` Numeric value specifying the number of additional attempts allowed when a bootstrap calculation fails for a case. Values from `0` to `50` may be used. Default is `0`.

`bootstrap` Numeric value specifying the number of bootstrap replications when `se = "bootstrap"`. Default is `100`.

`external` An optional vector of task identifiers used to define the tasks for model-based score calculation. If `NULL`, model-based outcomes are calculated using the tasks assigned to or observed for each person. If supplied, the specified calibrated tasks are used instead.

`type` Character string specifying the output type. If `"general"`, latent accuracy and speed scores and related model-based outcomes are returned without WCPM. If `"orf"`, model-based WCPM is additionally returned. Default is `"general"`.

`censoring` Logical. If `TRUE`, scoring accommodates censored task- or sub-task-level observations using the corresponding censoring model. Default is `FALSE`.

`testlet` Logical. If `TRUE`, sub-task-level scoring is performed using the testlet model. In the ORF assessment context, this corresponds to sentence-level observations nested within passages. Setting `testlet = TRUE` also activates the censoring-scoring pathway. Default is `FALSE`.

Details

In oral reading fluency (ORF) applications, the function can additionally produce model-based words-correct-per-minute (WCPM) scores. Scoring can be performed at the task level or, when a testlet model is used, at the sub-task level while accounting for the nesting of sentences within passages. Censored observations can also be accommodated.

The calibration parameters supplied through `calib.data` are treated as fixed when estimating person-level latent accuracy and speed scores. For standard task-level scoring without censoring, `est` selects among MLE, MAP, EAP, and fully Bayesian scoring.

When `se = "analytic"`, standard errors are obtained as part of the standard scoring procedure. When `se = "bootstrap"`, case-level parametric bootstrap calculations are performed, with the number of replications controlled by `bootstrap`. The `failsafe` argument controls how many additional attempts are made for a case when a bootstrap calculation fails.

The `external` argument can be used to calculate model-based outcomes on a specified set of calibrated tasks rather than on the tasks observed for each person. This can be useful when scores are to be expressed on a common task set.

When `censoring = TRUE`, censored and fully observed responses are distinguished using the variable supplied through `cens`. When `testlet = TRUE`, scoring is performed at the sub-task level using the testlet model and the censoring-scoring pathway is activated automatically.

Value

The returned object depends on the scoring procedure. For standard analytic or Bayesian scoring without censoring, an object of class `"scoring"` is returned containing person identifiers, descriptive response information, estimated latent accuracy and speed scores, their standard errors, and model-based outcomes appropriate to the selected `type`.

With `se = "bootstrap"`, an object of class `"bootstrap"` is returned. It contains `bootstrap.out`, with successfully estimated bootstrap results, and `error_case`, identifying cases for which bootstrap estimation was unsuccessful.

With censoring or testlet scoring, an object of class `"scoring.censoring"` is returned containing the corresponding Bayesian score estimates, posterior standard deviations, and model-based outcomes.

References

Kara, Y., Kamata, A., Potgieter, C. J., & Nese, J. F. T. (2020). Estimating model-based oral reading fluency: A Bayesian approach. *Educational and Psychological Measurement, 80*(5), 847-869. doi:10.1177/0013164419900208

Potgieter, C. J., Qiao, X., Kamata, A., & Kara, Y. (2024). Likelihood-based estimation of model-derived oral reading fluency. *Journal of Educational Measurement, 61*(3), 542-559. doi:10.1111/jedm.12404

Qiao, X., Evudottir, E., Potgieter, C. J., & Kamata, A. (2026). Likelihood-based estimation of model-derived oral reading fluency under overdispersed count-time data. *Measurement: Interdisciplinary Research and Perspectives*. doi:10.1080/15366367.2026.2643183

See Also

fit.model for model calibration, prep for preparing response data, and summary.scoring for summarizing scoring results.

Examples

```
WCPM_all <- scoring(
  calib.data = passage.calib.mcem,
  data = passage2,
  person.id = "id.student",
  occasion = "occasion",
  group = "grade",
  task.id = "id.passage",
  max.counts = "numwords.pass",
  obs.counts = "wrc",
  time = "sec",
  est = "map",
  se = "analytic",
  type = "general"
)
```

---

sentence.cens.high *Sentence-level data set with high censoring*

Description

This is an example data set. It is a sentence-level Oral Reading Fluency assessment data set with high proportion of censoring data for 58 students who were assigned to read 4 passages with a total of 23 sentences. The data is a small subset of the data collected by Nese and Kamata (2014-2018).

Usage

```
sentence.cens.high
```

Format

1334 rows and 9 variables:

`id.student` unique student identifier

`grade` student grade level

`id.passage` unique passage identifier

`ind.passage` passage index

`id.sentence` unique sentence sequence

`numwords.sent` the total number of words in the sentence

`wrc` the number of words read correct

`sec` time to read the sentence in seconds

`cens` censoring indicator, where 1 indicates a censored observation and 0 indicates a fully observed observation

Source

https://jnese.github.io/core-blog/

References

Nese, J. F. T. & Kamata, A. (2014-2018). Measuring Oral Reading Fluency: Computerized Oral Reading Evaluation (Project No. R305A140203) [Grant]. Institute of Education Sciences, U.S. Department of Education.

---

sentence.cens.low *Sentence-level data set with low censoring*

Description

This is an example data set. It is a sentence-level Oral Reading Fluency assessment data set with low proportion of censoring data for 58 students who were assigned to read 4 passages with a total of 23 sentences. The data is a small subset of the data collected by Nese and Kamata (2014-2018).

Usage

```
sentence.cens.low
```

Format

1334 rows and 9 variables:

`id.student` unique student identifier

`grade` student grade level

`id.passage` unique passage identifier

`ind.passage` passage index

`id.sentence` unique sentence sequence

`numwords.sent` the total number of words in the sentence

`wrc` the number of words read correct

`sec` time to read the sentence in seconds

`cens` censoring indicator, where 1 indicates a censored observation and 0 indicates a fully observed observation

Source

https://jnese.github.io/core-blog/

References

Nese, J. F. T. & Kamata, A. (2014-2018). Measuring Oral Reading Fluency: Computerized Oral Reading Evaluation (Project No. R305A140203) [Grant]. Institute of Education Sciences, U.S. Department of Education.

`sentence.level.data` *Sentence-level Oral Reading Fluency assessment data set*

Description

This is an example data set. It is a sentence-level Oral Reading Fluency assessment data set for 58 students who were assigned to read 4 passages with a total of 23 sentences. The data is a small subset of the data collected by Nese and Kamata (2014-2018).

Usage

```
sentence.level.data
```

Format

1334 rows and 8 variables:

`id.student` unique student identifier

`grade` student grade level

`id.passage` unique passage identifier

`ind.passage` passage index

`id.sentence` unique sentence sequence

`numwords.sent` the total number of words in the sentence

`wrc` the number of words read correct

`sec` time to read the sentence in seconds

Source

https://jnese.github.io/core-blog/

References

Nese, J. F. T. & Kamata, A. (2014-2018). Measuring Oral Reading Fluency: Computerized Oral Reading Evaluation (Project No. R305A140203) [Grant]. Institute of Education Sciences, U.S. Department of Education.

summary.bootstrap *Summarize bootstrap scoring results*

Description

Prints and returns bootstrap-based scoring results from an object of class "bootstrap". The method can alternatively report cases for which the bootstrap procedure produced an error.

Usage

```
## S3 method for class 'bootstrap'
summary(
  object,
  digits = 4,
  geterror = FALSE,
  verbose = TRUE,
  factor.scores = FALSE,
  ...
)
```

Arguments

object	A bootstrap scoring object of class "bootstrap".
digits	Integer retained for compatibility with the summary method. The current screen output is formatted to three decimal places. Default is 4.
geterror	Logical. If TRUE, the method reports object$error_case instead of the bootstrap scoring table. Default is FALSE.
verbose	Logical. If TRUE, the formatted bootstrap summary is printed. Processed results are returned invisibly. Default is TRUE.
factor.scores	Logical. If FALSE, columns whose names contain "theta" or "tau" are removed. Default is FALSE.
...	Additional arguments for the summary method. Currently not used.

Details

With geterror = FALSE, the method summarizes object$bootstrap.out. Numeric scoring columns are formatted to three decimal places for screen display, while returned values remain numeric.

With geterror = TRUE, existing error cases are printed and returned invisibly. If no error cases are present, the method prints a message. If the bootstrap output contains no columns, it prints "Bootstrap has 0 obs.".

Value

With nonempty bootstrap output, invisibly returns the processed bootstrap scoring table. With geterror = TRUE and available error cases, invisibly returns object$error_case. Otherwise an informative message is printed.

See Also

scoring and summary.scoring.

---

summary.fit.model *Summarize a fitted bspam model*

Description

Prints the calibrated task parameters and population-level hyperparameters from an object of class "fit.model".

Usage

```
## S3 method for class 'fit.model'
summary(object, digits = 4, ...)
```

Arguments

object	A fitted model object of class "fit.model", typically returned by fit.model.
digits	Integer passed to the table-printing method. The current implementation formats reported parameter estimates to three decimal places before printing. Default is 4.
...	Additional arguments for the summary method. Currently not used.

Details

The printed output first reports the first four calibrated task parameters, corresponding to a, b, alpha, and beta. It then reports vartau, the variance of the latent speed component, and rho, the correlation between latent accuracy and speed.

Value

Invisibly returns NULL after printing the model summary.

See Also

fit.model and summary.fit.model.testlet.

summary.fit.model.testlet *Summarize a fitted testlet model*

Description

Prints the estimated sub-task parameters and population-level hyperparameters from an object of class "fit.model.testlet".

Usage

```
## S3 method for class 'fit.model.testlet'
summary(object, digits = 4, ...)
```

Arguments

object	A fitted testlet-model object of class "fit.model.testlet", typically returned by fit.model.testlet or by fit.model with testlet = TRUE.
digits	Integer passed to the table-printing method. The current implementation formats reported parameter estimates to three decimal places before printing. Default is 4.
...	Additional arguments for the summary method. Currently not used.

Details

The printed output first reports the first four estimated sub-task parameters from the fitted model. It then prints sigma, gamma1, gamma2, rho.theta, and rho.testlet.

Value

Invisibly returns NULL after printing the model summary.

See Also

fit.model.testlet and fit.model.

summary.scoring *Summarize bspam scoring results*

Description

Prints and returns a person-level summary from an object of class "scoring". The method supports both task-level and testlet scoring output.

Usage

```
## S3 method for class 'scoring'
summary(
  object,
  digits = 4,
  verbose = TRUE,
  factor.scores = TRUE,
  show = "short",
  ...
)
```

Arguments

object	A scoring object of class "scoring", typically returned by scoring.
digits	Integer retained for compatibility with the summary method. The current screen output is formatted to three decimal places. Default is 4.
verbose	Logical. If TRUE, the formatted summary is printed. The processed summary data are returned invisibly regardless of this setting. Default is TRUE.
factor.scores	Logical. If FALSE, columns whose names contain "theta" or "tau" are removed. Default is TRUE.
show	Character string controlling the amount of information. With "short" (default), observed-data and design-summary columns are omitted. With "long", those columns are retained.
...	Additional arguments for the summary method. Currently not used.

Details

The method automatically distinguishes task-level scoring output from testlet scoring output. For task-level results, the short display removes occasion, group, task count, maximum-count total, observed-count total, observed time, and observed WCPM when present. For testlet results, the corresponding sub-task and testlet-specific fields are handled as well.

When verbose = TRUE, a formatted table is printed. The invisibly returned object contains the corresponding numeric summary data rather than the display-formatted values.

Value

Invisibly returns a data frame containing the processed person-level scoring summary. Included columns depend on factor.scores, show, scoring type, and whether ORF/WCPM output is present.

See Also

scoring and summary.bootstrap.

Index

∗ **datasets**
passage.calib.bayes, 10
passage.calib.mcem, 11
passage2, 12
sentence.cens.high, 26
sentence.cens.low, 27
sentence.level.data, 28

agg.word, 3

desc.data, 4

fit.model, 5, *9*, *19–23*, *26*, *30, 31*
fit.model.testlet, *7*, 8, *31*

get.cases, 10, *21, 22*

passage.calib.bayes, 10
passage.calib.mcem, 11
passage2, 12
plot.information, 13
plot.person, 14
plot.task, 15
plot.wcpm, 16
prep, *6, 7*, *9*, 17, *20*, *23*, *26*

run.mcem, 19
run.scoring, 21

scoring, *7*, *19*, *21, 22*, 23, *30*, *32*
sentence.cens.high, 26
sentence.cens.low, 27
sentence.level.data, 28
summary.bootstrap, 29, *32*
summary.fit.model, 30
summary.fit.model.testlet, *30*, 31
summary.scoring, *26*, *30*, 31