Package 'sft'

July 23, 2025

Version 2.4

Date 2025-04-18

Title Functions for Systems Factorial Technology Analysis of Data

Maintainer Joe Houpt <joseph.houpt@utsa.edu>

Depends R (>= 3.5), methods, fda, SuppDists

Description A series of tools for analyzing Systems Factorial Technology data. This includes functions for plotting and statistically testing capacity coefficient functions and survivor interaction contrast functions. Houpt, Blaha, McIntire, Havig, and Townsend (2013) <doi:10.3758/s13428-013-0377-3> provide a basic introduction to Systems Factorial Technology along with examples using the sft R package.

License GPL (>= 2)

RoxygenNote 6.0.1

NeedsCompilation no

Author Joe Houpt [aut, cre] (ORCID: <https://orcid.org/0000-0002-2784-5535>), Leslie Blaha [aut]

Repository CRAN

Date/Publication 2025-04-20 11:10:02 UTC

Contents

`assessment` *Workload Assessment Functions*

Description

Calculates the Workload Assessment Functions

Usage

```
assessment(RT, CR, stopping.rule=c("OR","AND", "STST"), correct=c(TRUE, FALSE),
  fast=c(TRUE, FALSE), detection=TRUE)
```

Arguments

`RT`	A list of response time arrays. The first array in the list is assumed to be the exhaustive condition.
`CR`	A list of correct/incorrect indicator arrays. If NULL, assumes all are correct.
`stopping.rule`	Indicates whether to compare performance to an OR, AND, or STST processing baseline.
`correct`	Indicates whether to assess performance on correct trials.
`fast`	Indicates whether to use cumulative distribution functions or survivor functions to assess performance.
`detection`	Indicates whether to use a detection task baseline or a discrimination task baseline.

Details

The assessment functions are a generalization of the workload capacity functions that account for incorrect responses. Townsend & Altieri (2012) derived four different assessment functions each for AND and OR tasks to compare performance with two targets with the performance of an unlimited-capacity, independent, parallel (UCIP) model. The correct assessment functions assess performance on correct trials and the incorrect assessment functions assess performance on the trials with incorrect responses. The fast assessment functions use the cumulative distribution functions, similar to

the AND capacity coefficient, and the slow assessment functions use the survivor functions, similar to the OR capacity coefficient.

In an OR task, the detection model assumes that the response will be correct if it is correct on either source, i.e., if either source is detected. In discrimination OR tasks, a participant may respond based on whichever source finishes first. Hence, the response will be incorrect if the first to finish is incorrect even if the second source would have been correct. This results in a slightly different baseline for performance assessment. See Donkin, Little and Houpt (2013) for details.

Value

`A` An object of class stepfun representing the estimated assessment function.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. and Altieri, N. (2012). An accuracy-response time capacity assessment function that measures performance against standard parallel predictions. *Psychological Review, 3*, 500-516.

Donkin, C, Little, D.R. and Houpt (2013). Assessing the effects of caution on the capacity of information processing. *Manuscript submitted for publication.*

See Also

`capacity.or` `capacity.and` `stepfun`

Examples

```
c1c.12 <- rexp(10000, .015)
c1i.12 <- rexp(10000, .01)
c1c    <- rexp(10000, .015)
c1i    <- rexp(10000, .01)

c2c.12 <- rexp(10000, .014)
c2i.12 <- rexp(10000, .01)
c2c    <- rexp(10000, .014)
c2i    <- rexp(10000, .01)

RT.1 <- pmin(c1c, c1i)
CR.1 <- c1c < c1i
RT.2 <- pmin(c2c, c2i)
CR.2 <- c2c < c2i

c1Correct <- c1c.12 < c1i.12
c2Correct <- c2c.12 < c2i.12

# OR Detection
CR.12 <- c1Correct | c2Correct
RT.12 <- rep(NA, 10000)
RT.12[c1Correct & c2Correct] <- pmin(c1c.12, c2c.12)[c1Correct & c2Correct]
```

```
RT.12[c1Correct & !c2Correct] <- c1c.12[c1Correct & !c2Correct]
RT.12[!c1Correct & c2Correct] <- c2c.12[!c1Correct & c2Correct]
RT.12[!c1Correct & !c2Correct] <- pmax(c1i.12, c2i.12)[!c1Correct & !c2Correct]

RT <- list(RT.12, RT.1, RT.2)
CR <- list(CR.12, CR.1, CR.2)
a.or.cf <- assessment(RT, CR, stopping.rule="OR", correct=TRUE, fast=TRUE, detection=TRUE)
a.or.cs <- assessment(RT, CR, stopping.rule="OR", correct=TRUE, fast=FALSE, detection=TRUE)
a.or.if <- assessment(RT, CR, stopping.rule="OR", correct=FALSE, fast=TRUE, detection=TRUE)
a.or.is <- assessment(RT, CR, stopping.rule="OR", correct=FALSE, fast=FALSE, detection=TRUE)

par(mfrow=c(2,2))
plot(a.or.cf, ylim=c(0,2))
plot(a.or.cs, ylim=c(0,2))
plot(a.or.if, ylim=c(0,2))
plot(a.or.is, ylim=c(0,2))


# AND
CR.12 <- c1Correct & c2Correct
RT.12 <- rep(NA, 10000)
RT.12[CR.12] <- pmax(c1c.12, c2c.12)[CR.12]
RT.12[c1Correct & !c2Correct] <- c2i.12[c1Correct & !c2Correct]
RT.12[!c1Correct & c2Correct] <- c1i.12[!c1Correct & c2Correct]
RT.12[!c1Correct & !c2Correct] <- pmin(c1i.12, c2i.12)[!c1Correct & !c2Correct]

RT <- list(RT.12, RT.1, RT.2)
CR <- list(CR.12, CR.1, CR.2)
a.and.cf <- assessment(RT, CR, stopping.rule="AND", correct=TRUE, fast=TRUE, detection=TRUE)
a.and.cs <- assessment(RT, CR, stopping.rule="AND", correct=TRUE, fast=FALSE, detection=TRUE)
a.and.if <- assessment(RT, CR, stopping.rule="AND", correct=FALSE, fast=TRUE, detection=TRUE)
a.and.is <- assessment(RT, CR, stopping.rule="AND", correct=FALSE, fast=FALSE, detection=TRUE)

par(mfrow=c(2,2))
plot(a.and.cf, ylim=c(0,2))
plot(a.and.cs, ylim=c(0,2))
plot(a.and.if, ylim=c(0,2))
plot(a.and.is, ylim=c(0,2))
```

`assessmentGroup` *Assessment Functions*

Description

Calculates the specified assessment function for each participant and each condition.

Usage

```
assessmentGroup(inData, stopping.rule=c("OR", "AND", "STST"), correct=c(TRUE, FALSE),
  fast=c(TRUE, FALSE), detection=TRUE, plotAt=TRUE, ...)
```

Arguments

inData	Data collected from a Double Factorial Paradigm experiment in standard form.
stopping.rule	Indicates whether to use OR, AND, or single-target-self-terminating (STST) processing baseline to calculate individual assessment functions.
plotAt	Indicates whether or not to generate plots of the assessment functions.
correct	Indicates whether to assess performance on correct trials.
fast	Indicates whether to use cumulative distribution functions or survivor functions to assess performance.
detection	Indicates whether to use a detection task baseline or a discrimination task baseline.
...	Arguments to be passed to plot function.

Details

For the details of the assessment functions, see assessment.

Value

A list containing the following components:

overview	A data frame indicating order of subject and condition for the assessment functions.
At.fn	Matrix with each row giving the values of the of the estimated assessment function for one participant in one condition for values of times. The rows match the ordering of statistic.
assessment	A list with the returned values from the assessment function for each participant and each condition.
times	Times at which the assessment functions are calculated in At.fn matrix.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. and Altieri, N. (2012). An accuracy-response time capacity assessment function that measures performance against standard parallel predictions. *Psychological Review, 3*, 500-516.

Donkin, C., Little, D. R., and Houpt, J. W. (2014). Assessing the speed-accuracy trade-off effect on the capacity of information processing. *Journal of Experimental Psychology: Human Perception and Performance, 40(3),* 1183.

See Also

assessment

Examples

```
## Not run:
data(dots)
assessmentGroup(subset(dots, Condition=="OR"),
  stopping.rule="OR", correct=TRUE, fast=FALSE,
  detection=TRUE)
assessmentGroup(subset(dots, Condition=="AND"),
  stopping.rule="AND", correct=TRUE, fast=TRUE, )

## End(Not run)
```

`capacity.and` *Capacity Coefficient for Exhaustive (AND) Processing*

Description

Calculates the Capacity Coefficient for Exhaustive (AND) Processing

Usage

```
capacity.and(RT, CR=NULL, ratio=TRUE)
```

Arguments

`RT`	A list of response time arrays. The first array in the list is assumed to be the exhaustive condition.
`CR`	A list of correct/incorrect indicator arrays. If NULL, assumes all are correct.
`ratio`	Indicates whether to return the standard ratio capacity coefficient or, if FALSE, the difference form.

Details

The AND capacity coefficient compares performance on task to an unlimited-capacity, independent, parallel (UCIP) model using cumulative reverse hazard functions. Suppose $K_i(t)$ is the cumulative reverse hazard function for response times when process i is completed in isolation and $K_i(t)$ is the cumulative reverse hazard function for response times when all processes must completed together. Then the AND capacity coefficient is given by,

$$C_{\mathrm{AND}}(t) = \frac{\sum_i K_i(t)}{K_{\mathrm{AND}}(t)}.$$

The numerator is the estimated cumulative reverse hazard function for the UCIP model, based on the response times for each process in isolation and the denominator is the actual performance.

$C_{\mathrm{AND}}(t) < 1$ implies worse performance than the UCIP model. This indicates that either there are limited processing resources, there is inhibition among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed serially).

$C_{\mathrm{AND}}(t) > 1$ implies better performance than the UCIP model. This indicates that either there are more processing resources available per process when there are more processes, that there is facilitation among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed coactively).

The difference form of the capacity coefficient (returned if ratio=FALSE) is given by,

$$C_{\mathrm{AND}}(t) = K_{\mathrm{AND}}(t) - \sum_i K_i(t).$$

Negative values indicate worse than UCIP performance and positive values indicate better than UCIP performance.

Value

`Ct`	An object of class approxfun representing the estimated AND capacity coefficient.
`Var`	An object of class approxfun representing the variance of the estimated AND capacity coefficient. Only returned if ratio=FALSE.
`Ctest`	A list with class "htest" that is returned from `ucip.test` and contains the statistic and p-value.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Wenger, M.J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review, 111*, 1003–1035.

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

Houpt, J.W., Blaha, L.M., McIntire, J.P., Havig, P.R. and Townsend, J.T. (2013). Systems Factorial Technology with R. *Behavior Research Methods*.

See Also

`ucip.test` `capacityGroup` `capacity.or` `estimateUCIPand` `estimateNAK` `approxfun`

Examples

```
rate1 <- .35
rate2 <- .3
RT.pa <- rexp(100, rate1)
RT.ap <- rexp(100, rate2)
RT.pp.limited <- pmax( rexp(100, .5*rate1), rexp(100, .5*rate2))
```

```
RT.pp.unlimited <- pmax( rexp(100, rate1), rexp(100, rate2))
RT.pp.super <- pmax( rexp(100, 2*rate1), rexp(100, 2*rate2))
tvec <- sort(unique(c(RT.pa, RT.ap, RT.pp.limited, RT.pp.unlimited, RT.pp.super)))

cap.limited <- capacity.and(RT=list(RT.pp.limited, RT.pa, RT.ap))
print(cap.limited$Ctest)
cap.unlimited <- capacity.and(RT=list(RT.pp.unlimited, RT.pa, RT.ap))
cap.super <- capacity.and(RT=list(RT.pp.super, RT.pa, RT.ap))

matplot(tvec, cbind(cap.limited$Ct(tvec), cap.unlimited$Ct(tvec), cap.super$Ct(tvec)),
  type='l', lty=1, ylim=c(0,3), col=2:4, main="Example Capacity Functions", xlab="Time",
  ylab="C(t)")
abline(1,0)
legend('topright', c("Limited", "Unlimited", "Super"), lty=1, col=2:4)
```

`capacity.id` *Capacity Coefficient for Full Identification (ID) Exhaustive Processing*

Description

Calculates the capacity coefficient for exhaustive processing accounting for processing differences in no-responses to single targets. The motivation for this version of the AND capacity coefficient is described in Howard et. al (2020).

Usage

```
capacity.id(dt.rt, nt.rt, st.rts, dt.cr, nt.cr, st.crs, ratio=TRUE)
```

Arguments

`dt.rt`	Response times from trials on which all signals indicate targets.
`nt.rt`	Response times from trials on which all signals are either absent or are non-targets.
`st.rts`	A list of arrays of response times from with each list containing response times for trials on which a distinct signal is the only signal present or that is indicating the target response.
`dt.cr`	A vector of indicators from trials on which all signals indicate targets indicating whether the participant correctly responded to the corresponding trial. If NULL, assumes all are correct.
`nt.cr`	A vector of indicators from trials on which all signals are either absent or are non-targets indicating whether the participant correctly responded to the corresponding trial. If NULL, assumes all are correct.
`st.crs`	A list of vectors of indicators from with each list containing response times for trials on which a distinct signal is the only signal present or that is indicating the target response indicating whether the participant correctly responded to the corresponding trial. If NULL, assumes all are correct.

`ratio` Indicates whether to return the standard ratio capacity coefficient or, if FALSE, the difference form.

Details

The identification-AND capacity coefficient compares performance on task to an unlimited-capacity, independent, parallel (UCIP) model using cumulative reverse hazard functions. Suppose $K_i(t)$ is the cumulative reverse hazard function for response times when process i indicates a target-present response but all other signals imply a non-target response, $K_{\mathrm{AND}}(t)$ is the cumulative reverse hazard function for response times when all target processes must completed together, and $K_{\mathrm{NT}}(t)$ si teh cumulative reverse hazard function for response times when all non-target response processes are completed together. Then the ID capacity coefficient is given by,

$$C_{\mathrm{ID}}(t) = \frac{\sum_i K_i(t)}{K_{\mathrm{AND}}(t) + K_{\mathrm{ID}}}.$$

The numerator is the estimated cumulative reverse hazard function for the UCIP model, based on the response times for each process in isolation and the denominator is the actual performance with a correction for the non-target response processes present in the single-target conditions.

$C_{\mathrm{ID}}(t) < 1$ implies worse performance than the UCIP model. This indicates that either there are limited processing resources, there is inhibition among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed serially).

$C_{\mathrm{ID}}(t) > 1$ implies better performance than the UCIP model. This indicates that either there are more processing resources available per process when there are more processes, that there is facilitation among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed coactively).

The difference form of the capacity coefficient (returned if ratio=FALSE) is given by,

$$C_{\mathrm{ID}}(t) = K_{\mathrm{ID}}(t) + K_{\mathrm{NT}}(t) - \sum_i K_i(t).$$

Negative values indicate worse than UCIP performance and positive values indicate better than UCIP performance.

Value

`Ct` An object of class approxfun representing the estimated ID capacity coefficient.

`Var` An object of class approxfun representing the variance of the estimated ID capacity coefficient. Only returned if ratio=FALSE.

`Ctest` A list with class "htest" that is returned from `ucip.test` and contains the statistic and p-value.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Wenger, M.J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review, 111*, 1003–1035.

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Howard, Z. L., Garrett, P., Little, D. R., Townsend, J. T., & Eidels, A. (2021). A show about nothing: No-signal processes in systems factorial technology. Psychological Review, 128(1), 187-201. doi:https://doi.org/10.1037/rev0000256

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

Houpt, J.W., Blaha, L.M., McIntire, J.P., Havig, P.R. and Townsend, J.T. (2013). Systems Factorial Technology with R. *Behavior Research Methods*.

See Also

ucip.test capacityGroup capacity.and estimateUCIPand estimateNAK approxfun

Examples

```
rate1p <- .35
rate1a <- .25
rate2p <- .35
rate2a <- .25
RT.pa <- pmax(rexp(100, rate1p), rexp(100, rate2a))
RT.ap <- pmax(rexp(100, rate2p), rexp(100, rate1a))
RT.nt <- pmax(rexp(100, rate1a), rexp(100, rate2a))
RT.pp.limited <- pmax( rexp(100, .5*rate1p), rexp(100, .5*rate2p))
RT.pp.unlimited <- pmax( rexp(100, rate1p), rexp(100, rate2p))
RT.pp.super <- pmax( rexp(100, 2*rate1p), rexp(100, 2*rate2p))
tvec <- sort(unique(c(RT.pa, RT.ap, RT.pp.limited, RT.pp.unlimited, RT.pp.super)))

cap.limited <- capacity.id(dt.rt=RT.pp.limited, nt.rt=RT.nt,
   st.rts=list(RT.pa, RT.ap))
print(cap.limited$Ctest)
cap.unlimited <- capacity.id(dt.rt=RT.pp.unlimited, nt.rt=RT.nt,
     st.rts=list(RT.pa, RT.ap))
cap.super <- capacity.id(dt.rt=RT.pp.super, nt.rt=RT.nt,
 st.rts=list(RT.pa, RT.ap))

matplot(tvec, cbind(cap.limited$Ct(tvec), cap.unlimited$Ct(tvec), cap.super$Ct(tvec)),
  type='l', lty=1, ylim=c(0,3), col=2:4, main="Example Capacity Functions", xlab="Time",
  ylab="C(t)")
abline(1,0)
legend('topright', c("Limited", "Unlimited", "Super"), lty=1, col=2:4)
```

capacity.or *Capacity Coefficient for First-Terminating (OR) Processing*

Description

Calculates the Capacity Coefficient for First-Terminating (OR) Processing

Usage

```
capacity.or(RT, CR=NULL, ratio=TRUE)
```

Arguments

RT	A list of response time arrays. The first array in the list is assumed to be the exhaustive condition.
CR	A list of correct/incorrect indicator arrays. If NULL, assumes all are correct.
ratio	Indicates whether to return the standard ratio capacity coefficient or, if FALSE, the difference form.

Details

The OR capacity coefficient compares performance on task to an unlimited-capacity, independent, parallel (UCIP) model using cumulative hazard functions. Suppose $H_i(t)$ is the cumulative hazard function for response times when process i is completed in isolation and $H_i(t)$ is the cumulative hazard function for response times when all processes occur together and a response is made as soon as any of the processes finish. Then the OR capacity coefficient is given by,

$$C_{\mathrm{OR}}(t) = \frac{H_{\mathrm{OR}}(t)}{\sum_i H_i(t)}.$$

The denominator is the estimated cumulative hazard function for the UCIP model, based on the response times for each process in isolation and the numerator is the actual performance.

$C_{\mathrm{OR}}(t) < 1$ implies worse performance than the UCIP model. This indicates that either there are limited processing resources, there is inhibition among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed serially).

$C_{\mathrm{OR}}(t) > 1$ implies better performance than the UCIP model. This indicates that either there are more processing resources available per process when there are more processes, that there is facilitation among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed coactively).

The difference form of the capacity coefficient is given by,

$$C_{\mathrm{OR}}(t) = H_{\mathrm{OR}}(t) - \sum_i H_i(t).$$

Negative values indicate worse than UCIP performance and positive values indicate better than UCIP performance.

Value

Ct	An object of class approxfun representing the OR capacity coefficient.
Var	An object of class approxfun representing the variance of the estimated OR capacity coefficient. Only returned if ratio=FALSE.
Ctest	A list with class "htest" that is returned from ucip.test and contains the statistic and p-value.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

Houpt, J.W., Blaha, L.M., McIntire, J.P., Havig, P.R. and Townsend, J.T. (2013). Systems Factorial Technology with R. *Behavior Research Methods*.

See Also

ucip.test capacityGroup capacity.and estimateUCIPor estimateNAH approxfun

Examples

```
rate1 <- .35
rate2 <- .3
RT.pa <- rexp(100, rate1)
RT.ap <- rexp(100, rate2)
RT.pp.limited <- pmin( rexp(100, .5*rate1), rexp(100, .5*rate2))
RT.pp.unlimited <- pmin( rexp(100, rate1), rexp(100, rate2))
RT.pp.super <- pmin( rexp(100, 2*rate1), rexp(100, 2*rate2))
tvec <- sort(unique(c(RT.pa, RT.ap, RT.pp.limited, RT.pp.unlimited, RT.pp.super)))

cap.limited <- capacity.or(RT=list(RT.pp.limited, RT.pa, RT.ap))
print(cap.limited$Ctest)
cap.unlimited <- capacity.or(RT=list(RT.pp.unlimited, RT.pa, RT.ap))
cap.super <- capacity.or(list(RT=RT.pp.super, RT.pa, RT.ap))

matplot(tvec, cbind(cap.limited$Ct(tvec), cap.unlimited$Ct(tvec), cap.super$Ct(tvec)),
  type='l', lty=1, ylim=c(0,3), col=2:4, main="Example Capacity Functions", xlab="Time",
  ylab="C(t)")
abline(1,0)
legend('topright', c("Limited", "Unlimited", "Super"), lty=1, col=2:4)
```

`capacity.stst` *Capacity Coefficient for Single-Target Self-Terminating (STST) Processing*

Description

Calculates the Capacity Coefficient for Single-Target Self-Terminating (STST) Processing

Usage

```
capacity.stst(RT, CR=NULL, ratio=TRUE)
```

Arguments

`RT`	A list of response time arrays. The first array in the list is assumed to be the single-target among N distractors condition.
`CR`	A list of correct/incorrect indicator arrays. If NULL, assumes all are correct.
`ratio`	Indicates whether to return the standard ratio capacity coefficient or, if FALSE, the difference form.

Details

The STST capacity coefficient compares performance on task to an unlimited-capacity, independent, parallel (UCIP) model using cumulative reverse hazard functions. Suppose $K_{i,1}(t)$ is the cumulative reverse hazard function for response times when single-target process i is completed in isolation and $K_{i,n}(t)$ is the cumulative reverse hazard function for response times when the single-target i is processed among n other processes, all completed together. Then the STST capacity coefficient is given by,

$$C_{\mathrm{STST}}(t) = \frac{K_{i,1}(t)}{K_{i,n}(t)}.$$

The numerator is the estimated cumulative reverse hazard function for the UCIP model, based on the response times for the i process in isolation and the denominator is the actual performance on the i process among n distractors or other active channels.

$C_{\mathrm{STST}}(t) < 1$ implies worse performance than the UCIP model. This indicates that either there are limited processing resources, there is inhibition among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed serially).

$C_{\mathrm{STST}}(t) > 1$ implies better performance than the UCIP model. This indicates that either there are more processing resources available per process when there are more processes, that there is facilitation among the subprocesses, or the items are not processed in parallel (e.g., the items may be processed coactively).

The difference form of the capacity coefficient (returned if ratio=FALSE) is given by,

$$C_{\mathrm{STST}}(t) = K_{i,n}(t) - K_{i,1}(t).$$

Negative values indicate worse than UCIP performance and positive values indicate better than UCIP performance.

Value

Ct	An object of class approxfun representing the estimated STST capacity coefficient.
Var	An object of class approxfun representing the variance of the estimated STST capacity coefficient. Only returned if ratio=FALSE.
Ctest	A list with class "htest" that is returned from ucip.test and contains the statistic and p-value.

Author(s)

Leslie Blaha <leslie.blaha@us.af.mil>

Joe Houpt <joseph.houpt@utsa.edu>

References

Blaha, L.M. & Townsend, J.T. (under review). On the capacity of single-target self-terminating processes.

Houpt, J.W. & Townsend, J.T. (2012). Statistical measures for workload capacity analysis. *Journal of Mathematical Psychology, 56*, 341-355.

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Townsend, J.T. & Wenger, M.J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review, 111*, 1003–1035.

See Also

ucip.test capacityGroup capacity.or capacity.and estimateNAK approxfun

Examples

```
rate1 <- .35
RT.pa <- rexp(100, rate1)
RT.pp.limited <- rexp(100, .5*rate1)
RT.pp.unlimited <- rexp(100, rate1)
RT.pp.super <- rexp(100, 2*rate1)
tvec <- sort(unique(c(RT.pa, RT.pp.limited, RT.pp.unlimited, RT.pp.super)))

cap.limited <- capacity.stst(RT=list(RT.pp.limited, RT.pa))
print(cap.limited$Ctest)
cap.unlimited <- capacity.stst(RT=list(RT.pp.unlimited, RT.pa))
cap.super <- capacity.stst(RT=list(RT.pp.super, RT.pa))

matplot(tvec, cbind(cap.limited$Ct(tvec), cap.unlimited$Ct(tvec), cap.super$Ct(tvec)),
  type='l', lty=1, ylim=c(0,5), col=2:4, main="Example Capacity Functions", xlab="Time",
  ylab="C(t)")
abline(1,0)
```

```
legend('topright', c("Limited", "Unlimited", "Super"), lty=1, col=2:4, bty="n")
```

---

capacityGroup *Capacity Analysis*

Description

Performs workload capacity analysis on each participant and each condition. Plots each capacity coefficient individually and returns the results of the nonparametric null-hypothesis test for unlimited capacity independent parallel performance.

Usage

```
capacityGroup(inData, acc.cutoff=.9, ratio=TRUE, OR=NULL,
  stopping.rule=c("OR", "AND", "STST"), plotCt=TRUE, ...)
```

Arguments

`inData`	Data collected from a Double Factorial Paradigm experiment in standard form.
`acc.cutoff`	Minimum accuracy for each stimulus category used in calculating the capacity coefficient.
`OR`	Indicates whether to compare performance to an OR or AND processing baseline. Provided for backwards compatibility for package version < 2.
`stopping.rule`	Indicates whether to use OR, AND or Single Target Self Terminating (STST) processing baseline to calculate individual capacity functions.
`ratio`	Indicates whether to return the standard ratio capacity coefficient or, if FALSE, the difference form.
`plotCt`	Indicates whether or not to generate plots of the capacity coefficients.
`...`	Arguments to be passed to plot function.

Details

For the details of the capacity coefficients, see `capacity.or`, `capacity.and` and `capacity.stst`. If accuracy in any of the stimulus categories used to calculate a capacity coefficient falls below the cutoff, NA is retuned for that value in both the statistic and the Ct matrix.

Value

A list containing the following components:

`overview`	A data frame indicating whether the OR and AND capacity coefficients significantly above baseline (super), below baseline (limited) or neither (unlimited) both at the individual level and at the group level in each condition. NA is returned for any participant that had performance below the accuracy cutoff in a condition.

`Ct.fn`	Matrix with each row giving the values of the of the estimated capacity coefficient for one participant in one condition for values of times. The rows match the ordering of statistic.
`Ct.var`	Matrix with each row giving the values of the of the variance of the estimated capacity coefficient for one participant in one condition for values of times. Only returned if ratio=FALSE.
`capacity`	A list with the returned values from the capacity function for each participant and each condition.
`times`	Times at which the matrix capacity coefficients are calculated in Ct.fn matrix.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Wenger, M.J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review, 111*, 1003–1035.

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

Houpt, J.W., Blaha, L.M., McIntire, J.P., Havig, P.R. and Townsend, J.T. (2013). Systems Factorial Technology with R. *Behavior Research Methods*.

See Also

`capacity.and` `capacity.or` `capacity.stst` `ucip.test`

Examples

```
## Not run:
data(dots)
capacityGroup(subset(dots, Condition=="OR"),
  stopping.rule="OR")
capacityGroup(subset(dots, Condition=="AND"),
  stopping.rule="AND")

## End(Not run)
```

dots *RT and and Accuracy from a Simple Detection Task*

Description

Data from a simple Double Factorial Paradigm task.

Usage

```
data(dots)
```

Format

A data frame with 57600 observations on the following 6 variables.

`Subject` A character vector indicating the participant ID.

`Condition` A character vector indicating whether participants could respond as soon as they detected either dot (OR) or both dots (AND).

`Correct` A logical vector indicating whether or not the participant responded correctly.

`RT` A numeric vector indicating the response time on a given trial.

`Channel1` A numeric vector indicating the stimulus level for the upper dot. 0: Absent; 1: Low contrast (slow); 2: High contrast (fast).

`Channel2` A numeric vector indicating the stimulus level for the lower dot. 0: Absent; 1: Low contrast (slow); 2: High contrast (fast).

Details

These data include response time and accuracy from nine participants that completed two versions of a Double Factorial Paradigm task. Stimuli were either two dots, one above fixation and one below, a single dot above fixation, a single dot below fixation, or a blank screen. Each dot could be presented either high or low contrast when present. In the OR task, participants were instructed to respond 'yes' whenever they saw either dot and 'no' otherwise. In the AND task, participants were instructed to respond 'yes' only when both dots were present and 'no' otherwise. See Eidels et al. (2012) or Houpt & Townsend (2010) for a more thorough description of the task.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

Source

Eidels, A., Townsend, J. T., Hughes, H. C., & Perry, L. A. (2012). Complementary relationship between response times, response accuracy, and task requirements in a parallel processing system. *Journal Cognitive Psychology*. Manuscript (submitted for publication).

References

Houpt, J.W. & Townsend, J.T. (2010). The statistical properties of the survivor interaction contrast. *Journal of Mathematical Psychology, 54*, 446-453.

Examples

```
data(dots)
summary(dots)
## Not run:
sicGroup(dots)
capacityGroup(dots)

## End(Not run)
```

`estimate.bounds` *Bounds on Response Time Cumulative Distribution Functions for Parallel Processing Models*

Description

Calculates the bounds on the range of cumulative distribution functions for response time data for parallel processing models under specified stopping rules (OR, AND, or Single-Target Self-Terminating).

Usage

```
estimate.bounds(RT, CR = NULL, stopping.rule = c("OR","AND","STST"),
  assume.ID=FALSE, numchannels=NULL, unified.space=FALSE)
```

Arguments

`RT`	A list of numeric response time arrays for the individual processing channels
`CR`	A list of correct/incorrect indicator arrays. If NULL, assumes all are correct.
`stopping.rule`	A character string specifying the stopping rule for the parallel processing model; must be one of "OR", "AND", "STST". If NULL, then "OR" is the default model.
`assume.ID`	A logical indicating whether the individual channel distributions are assumed to be Identically Distributed (ID). If FALSE, non-ID distributions are estimated from the RT data. If TRUE, only the first array in RT is used, together with numchannels, to estimate the distributions.
`numchannels`	Number of channels in the parallel processing model when all channels are active. If NULL,number channels will be estimated equal to length of RT.
`unified.space`	A logical indicating whether the unified capacity space version of the bounds should be estimated.

Details

The *estimate.bounds* function uses the response times from individual channels processing in isolation to estimae the response time distributions for an n-channel parallel model. The input argument RT must be a list of numeric arrays, containing either one array for each of the n channels to be estimated (so length(RT)=n), or it can have length(RT)=1 and the bounds can be found under an assumption that the n channels are identically distributed to the data in RT. For this latter case, assume.ID=TRUE and numchannels=n (where $n \geq 2$) must be specified.

Standard unlimited capacity parallel processing models for n simultaneously operating channels can produce a range of behavior, which is bounded by various functions derived from the probability distributions on each of the i, for $i = 1, \ldots, n$, channels operating in isolation. These bounds depend on the stopping rule under which is the parallel model is assumed to be operating (OR, AND, or STST).

stopping.rule="OR"

Let $F_n(t) = P[RT \leq t] = P[min(T_i) \leq t]$ for $i = 1, ..., n$, denote the cumulative distribution of response times under a minimum time (logical OR) stopping rule. The general bounds for n-channel parallel processing under an OR stopping rule are:

$$max_i F^i_{n-1}(t) \leq F_n(t) \leq min_{i,j}[F^i_{n-1}(t) + F^j_{n-1}(t) - F^{ij}_{n-2}(t)]$$

Under the assumption or conditions that the individual channels are identically distributed, this inequality chain simplifies to

$$F_{n-1}(t) \leq F_n(t) \leq [2 * F_{n-1}(t) - F_{n-2}(t)]$$

When the model under scrutiny has only $n = 2$ channels, the inequality chain takes the form:

$$F^i_1(t) \leq F_2(t) \leq [F^i_1(t) + F^j_1(t)]$$

stopping.rule="AND"

Let $G_n(t) = P[RT \leq t] = P[max(T_i) \leq t]$, for $i = 1, ..., n$, denote the cumulative distribution of repsonse times under a maximum time (logical AND, ehxaustive) stopping rule. The general bounds for n-channel parallel processing under an AND stopping rule are:

$$max_{i,j}[G^i_{n-1}(t) + G^j_{n-1}(t) - G^{ij}_{n-2}(t)] \leq G_n(t) \leq min_i G^i_{n-1}(t)$$

Under the assumption or conditions that the individual channels are identically distributed, this inequality chain simplifies to

$$[2 * G_{n-1}(t) - G_{n-2}(t)] \leq G_n(t) \leq G_{n-1}(t)$$

When the model under scrutiny has only $n = 2$ channels, the inequality chain takes the form:

$$[G^i_1(t) + G^j_1(t) - 1] \leq G_2(t) \leq G^i_1(t)$$

stopping.rule="STST"

Let $F_k(t) = P[RT \leq t]$ denote the cumulative distribution of repsonse times under a single-target self-terminating (STST) stopping rule, where the target of interest is on processing channel k among

n active channels. The general bounds for n-channel parallel processing under an STST stopping rule are:

$$\prod_{i=1}^{n} F_1(t) \leq F_k(t) \leq \sum_{i=1}^{n} F_1(t)$$

Under the assumption or conditions that the individual channels are identically distributed, this inequality chain simplifies to

$$[F_1(t)]^n \leq F_k(t) \leq n * F_1(t)$$

When the model under scrutiny has only $n = 2$ channels, the inequality chain takes the form:

$$[F_1^i(t) * F_1^j(t)] \leq F_k(t) \leq [F_1^i(t) + F_1^j(t)]$$

Note that in this case, $k = i$ or $k = j$, but this may not be specifiable *a priori* depending on experimental design.

Across all stopping rule conditions, violation of the upper bound indicates performance that is faster than can be predicted by an unlimited capacity parallel model. This may arise from positive (facilitatory) crosstalk between parallel channels, super capacity parallel processing, or some form of co-active architecture in the measured human response time data.

Violation of the lower bound indicates performance that is slower than predictd by an unlimited capacity parallel model. This may arise from negative (inhibitory) crosstalk between parallel channels, fixed capacity or limited capacity processing, or some form of serial architecture in the measured human response time data.

Value

A list containing the following components:

`Upper.Bound` An object of class "approxfun"" representing the estimated upper bound on the cumulative distribution function for an unlimited capacity parallel model.

`Lower.Bound` A object of class "approxfun" representing the estimated lower bound on the cumulative distribution function for an unlimited capacity parallel model.

Author(s)

Leslie Blaha <leslie.blaha@us.af.mil>

Joe Houpt <joseph.houpt@utsa.edu>

References

Blaha, L.M. & Townsend, J.T. (under review). On the capacity of single-target self-terminating processes.

Colonius, H. & Vorberg, D. (1994). Distribution inequalities for parallel models with unlimited capacity. *Journal of Mathematical Psychology, 38*, 35-58.

Grice, G.R., Canham, L., & Gwynne, J.W. (1984). Absense of a redundant-signals effect in a raction time task with divided attention. *Perception & Psychophysics, 36*, 565-570.

Miller, J. (1982). Divided attention: Evidence for coactivation with redundant signals. *Cognitive Psychology, 14*, 247-279.

Townsend, J.T. & Eidels, A. (2011). Workload capacity spaces: a unified methodology for response time measures of efficiency as workload is varied. *Psychonomic Bulletin & Review, 18*, 659-681.

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Townsend, J.T. & Wenger, M.J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review, 111*, 1003–1035.

See Also

ucip.test capacity.or capacity.and capacity.stst approxfun

Examples

```
#randomly generated data
rate1 <- .35
rate2 <- .3
rate3 <- .4
RT.paa <- rexp(100, rate1)
RT.apa <- rexp(100, rate2)
RT.aap <- rexp(100, rate3)
RT.or <- pmin(rexp(100, rate1), rexp(100, rate2), rexp(100, rate3))
RT.and <- pmax(rexp(100, rate1), rexp(100, rate2), rexp(100, rate3))
tvec <- sort(unique(c(RT.paa, RT.apa, RT.aap, RT.or, RT.and)))

or.bounds <- estimate.bounds(RT=list(RT.paa, RT.apa, RT.aap), CR=NULL, assume.ID=FALSE,
  unified.space=FALSE)
and.bounds <- estimate.bounds(RT=list(RT.paa, RT.apa, RT.aap))

## Not run:
#plot the or bounds together with a parallel OR model
matplot(tvec,
  cbind(or.bounds$Upper.Bound(tvec), or.bounds$Lower.Bound(tvec), ecdf(RT.or)(tvec)),
  type='l', lty=1, ylim=c(0,1), col=2:4, main="Example OR Bounds", xlab="Time",
  ylab="P(T<t)")
abline(1,0)
legend('topright', c("Upper Bound", "Lower Bound", "Parallel OR Model"),
  lty=1, col=2:4, bty="n")

#using the dots data set in sft package
data(dots)
attach(dots)
RT.A <- dots[Subject=='S1' & Condition=='OR' & Channel1==2 & Channel2==0, 'RT']
RT.B <- dots[Subject=='S1' & Condition=='OR' & Channel1==0 & Channel2==2, 'RT']
RT.AB <- dots[Subject=='S1' & Condition=='OR' & Channel1==2 & Channel2==2, 'RT']
tvec <- sort(unique(c(RT.A, RT.B, RT.AB)))
Cor.A <- dots[Subject=='S1' & Condition=='OR' & Channel1==2 & Channel2==0, 'Correct']
Cor.B <- dots[Subject=='S1' & Condition=='OR' & Channel1==0 & Channel2==2, 'Correct']
Cor.AB <- dots[Subject=='S1' & Condition=='OR' & Channel1==2 & Channel2==2, 'Correct']
capacity <- capacity.or(list(RT.AB,RT.A,RT.B), list(Cor.AB,Cor.A,Cor.B), ratio=TRUE)
```

```
bounds <- estimate.bounds(list(RT.A,RT.B), list(Cor.A,Cor.B), unified.space=TRUE)

#plot unified capacity coefficient space
plot(tvec, capacity$Ct(tvec), type="l", lty=1, col="red", lwd=2)
lines(tvec, bounds$Upper.Bound(tvec), lty=2, col="blue", lwd=2)
lines(tvec, bounds$Lower.Bound(tvec), lty=4, col="blue", lwd=2)
abline(h=1, col="black", lty=1)

## End(Not run)
```

`estimateNAH` *Neslon-Aalen Estimator of the Cumulative Hazard Function*

Description

Computes the Nelson-Aalen estimator of a cumulative hazard function.

Usage

```
estimateNAH(RT, CR)
```

Arguments

`RT`	A vector of times at which an event occurs (e.g., a vector of response times).
`CR`	A vector of status indicators, 1=normal, 0=censored. For response time data, this corresponds to 1=correct, 0=incorrect.

Details

The Nelson-Aalen estimator of the cumulative hazard function is a step function with jumps at each event time. The jump size is given by the number at risk up until immediately before the event. If Y(t) is the number at risk immediately before t, then the N-A estimator is given by:

$$H(t) = \sum_{s \in \{\text{EventTimes} < t\}} \frac{1}{Y(s)}$$

Value

`H`	A function of class "stepfun" that returns the Nelson-Aalen estimator of the cumulative hazard function.
`Var`	A function of class "stepfun" that returns the estimated variance of the Nelson-Aalen estimator of the cumulative hazard function.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Aalen, O. O., Borgan, O., & Gjessing, H. K. (2008). *Survival and event history analysis: A process point of view*. New York: Springer.

See Also

estimateNAK stepfun

Examples

```
x <- rexp(50, rate=.5)
censoring <- runif(50) < .90
H.NA <- estimateNAH(x, censoring)

# Plot the estimated cumulative hazard function
plot(H.NA$H,
  main="Cumulative Hazard Function\n X ~ Exp(.5)    n=50",
  xlab="X", ylab="H(x)")

# Plot 95% Confidence intervals
times <- seq(0,10, length.out=100)
lines(times, H.NA$H(times) + sqrt(H.NA$Var(times))*qnorm(1-.05/2), lty=2)
lines(times, H.NA$H(times) - sqrt(H.NA$Var(times))*qnorm(1-.05/2), lty=2)

# Plot the true cumulative hazard function
abline(0,.5, col='red')
```

---

estimateNAK *Neslon-Aalen Estimator of the Reverse Cumulative Hazard Function*

Description

Computes the Nelson-Aalen estimator of a reverse cumulative hazard function.

Usage

```
estimateNAK(RT, CR)
```

Arguments

RT	A vector of times at which an event occurs (e.g., a vector of response times).
CR	A vector of status indicators, 1=normal, 0=censored. For response time data, this corresponds to 1=correct, 0=incorrect.

Details

The Nelson-Aalen estimator of the cumulative reverse hazard function is a step function with jumps at each event time. The jump size is given by the number of events that have occurred up to and including the event. If G(t) is the number events that have occurred up to and including t, then the N-A estimator of the cumulative reverse hazard function is given by:

$$K(t) = - \sum_{s \in \{\text{EventTimes} > t\}} \frac{1}{G(s)}$$

Value

K	A function of class "stepfun" that returns the Nelson-Aalen estimator of the cumulative reverse hazard function.
Var	A function of class "stepfun" that returns estimated variance of the Nelson-Aalen estimator of the cumulative reverse hazard function.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Aalen, O. O., Borgan, O., & Gjessing, H. K. (2008). *Survival and event history analysis: A process point of view*. New York: Springer.

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

See Also

estimateNAH stepfun

Examples

```
x <- rexp(50, rate=.5)
censoring <- runif(50) < .90
K.NA <- estimateNAK(x, censoring)

# Plot the estimated cumulative reverse hazard function
plot(K.NA$K,
  main="Cumulative Reverse Hazard Function\n X ~ Exp(.5)    n=50",
  xlab="X", ylab="K(x)")

# Plot 95% Confidence intervals
times <- seq(0,10, length.out=100)
lines(times, K.NA$K(times) + sqrt(K.NA$Var(times))*qnorm(1-.05/2), lty=2)
lines(times, K.NA$K(times) - sqrt(K.NA$Var(times))*qnorm(1-.05/2), lty=2)

# Plot the true cumulative reverse hazard function
lines(times, log(pexp(times, .5)), col='red')
```

`estimateUCIPand` *UCIP Performance on AND Tasks*

Description

Estimates the reverse cumulative hazard function of an unlimited capacity, independent, parallel process on an AND task.

Usage

```
estimateUCIPand(RT, CR)
```

Arguments

`RT`	A list of arrays of response times. Each list is used to estimate the response time distribution of a separate channel.
`CR`	A list of arrays of correct (1) or incorrect (0) indicators corresponding to each element of the list RT.

Details

This function concerns the processing time of an unlimited capacity, independent, parallel (UCIP) system. This means that the completion time for each processing channel does not vary based on the presence of other processes. Thus, the performance on tasks with a single process can be used to estimate performance of the UCIP model with multiple processes occurring.

For example, in a two channel UCIP system the probability that both processes have finished (AND processing) is the product of the probabilities of that each channel has finished.

$$P(T_{ab} \leq t) = P(T_a \leq t)P(T_b \leq t)$$

We are interested in the cumulative reverse hazard function, which is the natural log of the cumulative distribution function. Because the log of a product is the sum of the logs, this gives us the following equality for the two channel AND process.

$$K_{ab}(t) = K_a(t) + K_b(t)$$

In general, the cumulative reverse hazard function of a UCIP AND process is estimated by the sum of the cumulative reverse hazard functions of each sub-process.

$$K_{\mathrm{UCIP}}(t) = \sum_{i=1}^{n} K_i(t)$$

The cumulative reverse hazard functions of the sub-processes are estimated using the Nelson-Aalen estimator. The Nelson-Aalen estimator is a Gaussian martingale, so the estimate of the UCIP performance is also a Gaussian martingale and the variance of the estimator can be estimated with the sum of variance estimates for each sub-process.

Value

K	A function of class "stepfun" that returns the Nelson-Aalen estimator of the cumulative reverse hazard function of a UCIP model on an exhaustive (AND) task.
Var	A function of class "stepfun" that returns the estimated variance of the Nelson-Aalen estimator of the cumulative reverse hazard function of a UCIP model on an exhaustive (AND) task.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Wenger, M.J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review, 111*, 1003-1035.

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

See Also

estimateNAK

Examples

```
# Channel completion times and accuracy
rt1 <- rexp(100, rate=.5)
cr1 <- runif(100) < .90
rt2 <- rexp(100, rate=.4)
cr2 <- runif(100) < .95
Kucip = estimateUCIPand(list(rt1, rt2), list(cr1, cr2))


# Plot the estimated UCIP cumulative reverse hazard function
plot(Kucip$K, do.p=FALSE,
  main="Estimated UCIP Cumulative Reverse Hazard Function\n
    X~max(X1,X2)    X1~Exp(.5)    X2~Exp(.4)",
  xlab="X", ylab="K_UCIP(x)")
# Plot 95% Confidence intervals
times <- seq(0,10, length.out=100)
lines(times, Kucip$K(times) + sqrt(Kucip$Var(times))*qnorm(1-.05/2), lty=2)
lines(times, Kucip$K(times) - sqrt(Kucip$Var(times))*qnorm(1-.05/2), lty=2)
# Plot true UCIP cumulative reverse hazard function
lines(times[-1], log(pexp(times[-1], .5)) + log(pexp(times[-1], .4)), col='red')
```

`estimateUCIPor` *UCIP Performance on OR Tasks*

Description

Estimates the cumulative hazard function of an unlimited capacity, independent, parallel process on an OR task.

Usage

```
estimateUCIPor(RT, CR)
```

Arguments

`RT`	A list of arrays of response times. Each list is used to estimate the response time distribution of a separate channel.
`CR`	A list of arrays of correct (1) or incorrect (0) indicators corresponding to each element of the list RT.

Details

This function concerns the processing time of an unlimited capacity, independent, parallel (UCIP) system. This means that the completion time for each processing channel does not vary based on the presence of other processes. Thus, the performance on tasks with a single process can be used to estimate performance of the UCIP model with multiple processes occurring.

For example, in a two channel UCIP system the probability that no process has finished (OR processing) is the product of the probabilities of that each channel has not finished.

$$P(T_{ab} > t) = P(T_a > t)P(T_b > t)$$

We are interested in the cumulative hazard function, which is the natural log of the suvivor function (which is one minus the cumulative distribution function). Because the log of a product is the sum of the logs, this gives us the following equality for the two channel OR process.

$$H_{ab}(t) = H_a(t) + H_b(t)$$

In general, the cumulative hazard function of a UCIP OR process is estimated by the sum of the cumulative hazard functions of each sub-process.

$$H_{\mathrm{UCIP}}(t) = \sum_{i=1}^{n} H_i(t)$$

The cumulative hazard functions of the sub-processes are estimated using the Nelson-Aalen estimator. The Nelson-Aalen estimator is a Gaussian martingale, so the estimate of the UCIP performance is also a Gaussian martingale and the variance of the estimator can be estimated with the sum of variance estimates for each sub-process.

Value

H	A function of class "stepfun" that returns the Nelson-Aalen estimator of the cumulative hazard function of a UCIP model on a first-terminating (OR) task.
Var	A function of class "stepfun" that returns the estimated variance of the Nelson-Aalen estimator of the cumulative reverse hazard function of a UCIP model on a first-terminating (OR) task.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

See Also

estimateNAH

Examples

```
# Channel completion times and accuracy
rt1 <- rexp(100, rate=.5)
cr1 <- runif(100) < .90
rt2 <- rexp(100, rate=.4)
cr2 <- runif(100) < .95
Hucip = estimateUCIPor(list(rt1, rt2), list(cr1, cr2))


# Plot the estimated UCIP cumulative hazard function
plot(Hucip$H, do.p=FALSE,
  main="Estimated UCIP Cumulative Hazard Function\n
    X~min(X1,X2)    X1~Exp(.5)    X2~Exp(.4)",
  xlab="X", ylab="H_UCIP(t)")
# Plot 95% Confidence intervals
times <- seq(0,10, length.out=100)
lines(times, Hucip$H(times) + sqrt(Hucip$Var(times))*qnorm(1-.05/2), lty=2)
lines(times, Hucip$H(times) - sqrt(Hucip$Var(times))*qnorm(1-.05/2), lty=2)
#Plot true UCIP cumulative hazard function
abline(0,.9, col='red')
```

`fPCAassessment` *Functional Principal Components Analysis for the Assessment Functions*

Description

Calculates the principle functions and scores for the workload assessment measure of performance by each individual in each condition.

Usage

```
fPCAassessment(sftData, dimensions, stopping.rule=c("OR", "AND", "STST"),
               correct=c(TRUE,FALSE), fast=c(TRUE,FALSE), detection=TRUE,
               register=c("median","mean","none"), plotPCs=FALSE, ...)
```

Arguments

`sftData`	Data collected from a Double Factorial Paradigm experiment in standard form.
`dimensions`	The number of principal functions with which to represent the data.
`stopping.rule`	Indicates whether to use OR, AND or Single Target Self Terminating (STST) processing baseline to calculate individual assessment functions.
`correct`	Indicates whether to assess performance on correct trials.
`fast`	Indicates whether to use cumulative distribution functions or survivor functions to assess performance.
`detection`	Indicates whether to use a detection task baseline or a discrimination task baseline.
`register`	Indicates value to use for registering the assessment data.
`plotPCs`	Indicates whether or not to generate plots of the principal functions.
`...`	Arguments to be passed to plot function.

Details

Functional principal components analysis (fPCA) is an extension of standard principal components analysis to infinite dimensional (function) spaces. Just as in standard principal components analysis, fPCA is a method for finding a basis set of lower dimensionality than the original space to represent the data. However, in place of basis vectors, fPCA has basis functions. Each function in the original dataset can then be represented by a linear combination of those bases, so that given the bases, the each datum is represented by a vector of its coefficients (or scores) in that linear combination.

The assessment coefficient is a function across time, so the differences among assessment coefficients from different participants and/or conditions may be quite informative. fPCA gives a well motivated method for representing those differences in a concise way. The factor scores can be used to examine differences among assessment coefficients, accounting for variation across the entire function.

This function implements the steps outlines in Burns, Houpt, Townsend and Endres (2013) applied to the assessment functions defined in Townsend and Altieri (2012) and Donkin, Little, and Houpt

(2013). First, the data are shifted by subtracting the median response time within each condition for each participant, but across both single target and multiple target trials, so that the assessment curves will be registered. Second, each assessment coefficient is calculated with the shifted response times. Next, the mean assessment coefficient is subtracted from each assessment coefficient, then the representation of the resulting assessment coefficients are translated to a b-spline basis. The fPCA procedure extracts the basis function from the bspline space that accounts for the largest variation across the assessment coefficients, then the next basis function which must be orthogonal to the first but explains the most amount of variation in the assessment coefficients given that constraint and so on until the indicated number of basis have been extracted. Once the assessment functions are represented in the reduced space, a varimax rotation is applied.

The assessment functions can be registered to the mean or median response time across all levels of workload but within each participant and condition, or the analyses can be performed without registration.

For details on fPCA for the assessment coefficient, see Burns, Houpt, Townsend and Endres (2013). For details on fPCA in general using R, see Ramsay, Hooker and Graves (2009).

Value

`Scores`	Data frame containing the Loading values for each participant and condition.
`MeanAT`	Object of class approxfun representing the mean At function.
`PF`	List of objects of class approxfun representing the principal functions.
`medianRT`	Size of shift used to register each assessment curve (median RT).

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Burns, D.M., Houpt, J.W., Townsend, J.T. & Endres, M.J. (2013). Functional principal components analysis of workload assessment functions. *Behavior Research Methods*

Donkin, C, Little, D.R. and Houpt (2013). Assessing the effects of caution on the capacity of information processing. *Manuscript submitted for publication.*

Ramsay, J., Hooker, J. & Graves, S. (2009). Functional Data Analysis with R and MATLAB. New York, NY: Springer.

Townsend, J.T. and Altieri, N. (2012). An accuracy-response time capacity assessment function that measures performance against standard parallel predictions. *Psychological Review, 3*, 500-516.

See Also

`assessment` `fda`

Examples

```
## Not run:
data(dots)
fPCAassessment(dots, dimensions=2, stopping.rule="OR", register="median",
```

```
                correct=TRUE, fast=FALSE, detection=TRUE, plotPCs=TRUE)

## End(Not run)
```

`fPCAcapacity` *Functional Principal Components Analysis for the Capacity Coefficient*

Description

Calculates the principle functions and scores for the workload capacity measure of performance by each individual in each condition.

Usage

```
fPCAcapacity(sftData, dimensions, acc.cutoff=.75, OR=NULL,
  stopping.rule=c("OR","AND","STST"), ratio=TRUE,
  register=c("median","mean","none"), plotPCs=FALSE, ...)
```

Arguments

`sftData`	Data collected from a Double Factorial Paradigm experiment in standard form.
`dimensions`	The number of principal functions with which to represent the data.
`acc.cutoff`	Minimum accuracy needed to be included in the analysis.
`OR`	Indicates whether to compare performance to an OR or AND processing baseline. Provided for backwards compatibility for package version < 2.
`stopping.rule`	Indicates whether to use OR, AND or Single Target Self Terminating (STST) processing baseline to calculate individual capacity functions.
`ratio`	Whether to use ratio Ct or difference Ct.
`register`	Indicates value to use for registering the capacity data.
`plotPCs`	Indicates whether or not to generate plots of the principal functions.
`...`	Arguments to be passed to plot function.

Details

Functional principal components analysis (fPCA) is an extension of standard principal components analysis to infinite dimensional (function) spaces. Just as in standard principal components analysis, fPCA is a method for finding a basis set of lower dimensionality than the original space to represent the data. However, in place of basis vectors, fPCA has basis functions. Each function in the original dataset can then be represented by a linear combination of those bases, so that given the bases, the each datum is represented by a vector of its coefficients (or scores) in that linear combination.

The capacity coefficient is a function across time, so the differences among capacity coefficients from different participants and/or conditions may be quite informative. fPCA gives a well motivated method for representing those differences in a concise way. The factor scores can be used to examine differences among capacity coefficients, accounting for variation across the entire function.

This function implements the steps outlines in Burns, Houpt, Townsend and Endres (2013). First, the data are shifted by subtracting the median response time within each condition for each participant, but across both single target and multiple target trials, so that the capacity curves will be registered. Second, each capacity coefficient is calculated with the shifted response times. Next, the mean capacity coefficient is subtracted from each capacity coefficient, then the representation of the resulting capacity coefficients are translated to a b-spline basis. The fPCA proceedure extracts the basis function from the bspline space that accounts for the largest variation across the capacity coefficients, then the next basis function which must be orthogonal to the first but explains the most amount of variation in the capacity coefficients given that constraint and so on until the indicated number of basis have been extracted. Once the capacity functions are represented in the reduced space, a varimax rotation is applied.

The capacity functions can be registered to the mean or median response time across all levels of workload but within each participant and condition, or the analyses can be performed without registration.

For details on fPCA for the capacity coefficient, see Burns, Houpt, Townsend and Endres (2013). For details on fPCA in general using R, see Ramsay, Hooker and Graves (2009).

Value

`Scores`	Data frame containing the Loading values for each participant and condition.
`MeanCT`	Object of class approxfun representing the mean Ct function.
`PF`	List of objects of class approxfun representing the principal functions.
`medianRT`	Size of shift used to register each capacity curve (median RT).

Author(s)

Joe Houpt <joseph.houpt@utsa.edu> Devin Burns <burnsde@mst.edu>

References

Burns, D.M., Houpt, J.W., Townsend, J.T. & Endres, M.J. (2013). Functional principal components analysis of workload capacity functions. *Behavior Research Methods*

Ramsay, J., Hooker, J. & Graves, S. (2009). Functional Data Analysis with R and MATLAB. New York, NY: Springer.

See Also

`capacity.and` `capacity.or` `fda`

Examples

```
## Not run:
data(dots)
fPCAcapacity(dots, dimensions=2,stopping.rule="OR",
  plotPCs=TRUE)

## End(Not run)
```

`mic.test` *Test of the Mean Interaction Contrast*

Description

Performs either an Adjusted Rank Transform or ANOVA test for an interaction at the mean level.

Usage

```
mic.test(HH, HL, LH, LL, method=c("art", "anova"))
```

Arguments

`HH`	Response times from the High–High condition.
`HL`	Response times from the High–Low condition.
`LH`	Response times from the Low–High condition.
`LL`	Response times from the Low–Low condition.
`method`	If "art", use the adjusted rank transform test. If "anova" use ANOVA.

Details

The mean interaction contrast (MIC) indicates the architecture of a process. Serial processes result in MIC equal to zero. Parallel-OR and Coactive process have a positive MIC. Parallel-AND process have a negative MIC. A test for a significant MIC can be done with a nonparametric adjusted rank transform test (described below) or an ANOVA.

The Adjusted Rank Transform is a nonparametric test for an interaction between two discrete variables. The test is carried out by first subtracting the mean effect of the salience level on each channel. Suppose, $m_{H,\cdot}$ = E[RT; Level of Channel 1 is Fast], $m_{L,\cdot}$ = E[RT; Level of Channel 1 is Slow], $m_{\cdot,H}$ = E[RT; Level of Channel 2 is Fast], $m_{\cdot,L}$ = E[RT; Level of Channel 2 is Slow]. Then for each response time from the fast–fast condition, $m_{H,\cdot}$ and $m_{\cdot,H}$ are subtracted. Likewise, for each of the other conditions, the appropriate m is subtracted. Next, each mean subtracted response time is replaced with its rank across all conditions (e.g., the fastest time of all conditions would be replaced with a 1). In this implementation, tied response times are assigned using the average rank. Finally, a standard ANOVA on the ranks is done on the ranks and the p-value of that test is returned. This test was recommended by Sawilowsky (1990) based on a survey of a number of nonparametric tests for interactions. He credits Reinach (1965) for first developing the test.

Value

A list of class "htest" containing:

`statistic`	The value of the MIC.
`p.value`	The p-value of the ART or ANOVA test.
`alternative`	A description of the alternative hypothesis.
`method`	A string indicating that the Houpt-Townsend statistic was used.
`data.name`	A string indicating which data were used for which input.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Reinach, S.G. (1965). A nonparametric analysis for a multiway classification with one element per cell. *South African Journal of Agricultural Science, 8*, 941–960.

Sawilowsky, S.S. (1990). Nonparametric tests of interaction in experimental design. *Review of Educational Research, 60*, 91–126.

Houpt, J.W. & Townsend, J.T. (2010). The statistical properties of the survivor interaction contrast. *Journal of Mathematical Psychology, 54*, 446-453.

Examples

```
T1.h <- rweibull(300, shape=2 , scale=400 )
T1.l <- rweibull(300, shape=2 , scale=800 )
T2.h <- rweibull(300, shape=2 , scale=400 )
T2.l <- rweibull(300, shape=2 , scale=800 )

Serial.hh <- T1.h + T2.h
Serial.hl <- T1.h + T2.l
Serial.lh <- T1.l + T2.h
Serial.ll <- T1.l + T2.l
mic.test(HH=Serial.hh, HL=Serial.hl, LH=Serial.lh, LL=Serial.ll)

Parallel.hh <- pmax(T1.h, T2.h)
Parallel.hl <- pmax(T1.h, T2.l)
Parallel.lh <- pmax(T1.l, T2.h)
Parallel.ll <- pmax(T1.l, T2.l)
mic.test(HH=Parallel.hh, HL=Parallel.hl, LH=Parallel.lh, LL=Parallel.ll, method="art")
```

---

sic *Calculate the Survivor Interaction Contrast*

Description

Function to calculate survivor interaction contrast and associated measures.

Usage

```
sic(HH, HL, LH, LL, domtest="ks", sictest="ks", mictest=c("art", "anova"))
```

Arguments

HH	Response times from the High–High condition.
HL	Response times from the High–Low condition.
LH	Response times from the Low–High condition.
LL	Response times from the Low–Low condition.
sictest	Which type of hypothesis test to use for SIC form.
domtest	Which type of hypothesis test to use for testing stochastic dominance relations, either as series of KS tests ("ks") or the dominance test based on Dirichlet process priors ("dp"). DP not yet implemented.
mictest	Which type of hypothesis test to use for the MIC, either adjusted rank transform or ANOVA.

Details

SIC(t) = (S_LL - S_LH) - (S_HL - S_HH)

This function calculates the Survivor Interaction Contrast (SIC; Townsend & Nozawa, 1995). The SIC indicates the architecture and stopping-rule of the underlying information processing system. An entirely positive SIC indicates parallel first-terminating processing. An entirely negative SIC indicates parallel exhaustive processing. An SIC that is always zero indicates serial first-terminating processing. An SIC that is first positive then negative indicates either serial exhaustive or coactive processing. To distinguish between these two possibilities, an additional test of the mean interaction contrast (MIC) is used; coactive processing leads to a positive MIC while serial processing leads to an MIC of zero.

For the SIC function to distinguish among the processing types, the salience manipulation on each channel must selectively influence its respective channel (although see Eidels, Houpt, Altieri, Pei & Townsend, 2010 for SIC prediction from interactive parallel models). Although the selective influence assumption cannot be directly tested, one implication is that the distribution the HH response times stochastically dominates the HL and LH distributions which each in turn stochastically dominate the LL response time distribution. This implication is automatically tested in this function. The KS dominance test uses eight two-sample Kolmogorov-Smirnov tests: HH < HL, HH < LH, HL < LL, LH < LL should be significant while HH > HL, HH > LH, HL > LL, LH > LL should not. The DP uses four tests to determine which relation has the highest Bayes factor assuming a Dirichlet process prior for each of (HH, HL), (HH, LH), (HL, LL) and (LH, LL). See Heathcote, Brown, Wagenmakers & Eidels, 2010, for more details.

This function also performs a statistical analysis to determine whether the positive and negative parts of the SIC are significantly different from zero. Currently the only statistical test is based on the generalization of the two-sample Kolmogorov-Smirnov test described in Houpt & Townsend, 2010. This test performs two separate null-hypothesis tests: One test for whether the largest positive value of the SIC is significantly different from zero and one test for whether the largest negative value is significantly different from zero.

Value

SIC	An object of class stepfun representing the SIC.

Dominance	A data frame with the first column indicating which ordering was tested, the second column indicating the test statistic and the third indicating the p-value for that value of the statistic.
Dvals	A Matrix containing the values of the test statistic and the associated p-values.
MIC	Results of an adjusted rank transform test of the mean interaction contrast.
N	The scaling factor used for the KS test of the SIC form.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Houpt, J.W. & Townsend, J.T. (2010). The statistical properties of the survivor interaction contrast. *Journal of Mathematical Psychology, 54*, 446-453.

Houpt, J.W., Blaha, L.M., McIntire, J.P., Havig, P.R. and Townsend, J.T. (2013). Systems Factorial Technology with R. *Behavior Research Methods*.

See Also

stepfun sicGroup mic.test sic.test

Examples

```
T1.h <- rexp(50, .2)
T1.l <- rexp(50, .1)
T2.h <- rexp(50, .21)
T2.l <- rexp(50, .11)

SerialAND.hh <- T1.h + T2.h
SerialAND.hl <- T1.h + T2.l
SerialAND.lh <- T1.l + T2.h
SerialAND.ll <- T1.l + T2.l
SerialAND.sic <- sic(HH=SerialAND.hh, HL=SerialAND.hl, LH=SerialAND.lh,
  LL=SerialAND.ll)
print(SerialAND.sic$Dvals)
plot(SerialAND.sic$SIC, do.p=FALSE, ylim=c(-1,1))

p1 <- runif(200) < .3
SerialOR.hh <- p1[1:50]    * T1.h + (1-p1[1:50]   )*T2.h
SerialOR.hl <- p1[51:100]  * T1.h + (1-p1[51:100] )*T2.l
SerialOR.lh <- p1[101:150] * T1.l + (1-p1[101:150])*T2.h
SerialOR.ll <- p1[151:200] * T1.l + (1-p1[151:200])*T2.l
SerialOR.sic <- sic(HH=SerialOR.hh, HL=SerialOR.hl, LH=SerialOR.lh, LL=SerialOR.ll)
print(SerialOR.sic$Dvals)
plot(SerialOR.sic$SIC, do.p=FALSE, ylim=c(-1,1))
```

```
ParallelAND.hh <- pmax(T1.h, T2.h)
ParallelAND.hl <- pmax(T1.h, T2.l)
ParallelAND.lh <- pmax(T1.l, T2.h)
ParallelAND.ll <- pmax(T1.l, T2.l)
ParallelAND.sic <- sic(HH=ParallelAND.hh, HL=ParallelAND.hl, LH=ParallelAND.lh,
  LL=ParallelAND.ll)
print(ParallelAND.sic$Dvals)
plot(ParallelAND.sic$SIC, do.p=FALSE, ylim=c(-1,1))

ParallelOR.hh <- pmin(T1.h, T2.h)
ParallelOR.hl <- pmin(T1.h, T2.l)
ParallelOR.lh <- pmin(T1.l, T2.h)
ParallelOR.ll <- pmin(T1.l, T2.l)
ParallelOR.sic <- sic(HH=ParallelOR.hh, HL=ParallelOR.hl, LH=ParallelOR.lh,
  LL=ParallelOR.ll)
print(ParallelOR.sic$Dvals)
plot(ParallelOR.sic$SIC, do.p=FALSE, ylim=c(-1,1))
```

`sic.test` *Statistical test of the SIC.*

Description

Function to test for statistical significance of the positive and negative parts of a SIC.

Usage

```
sic.test(HH, HL, LH, LL, method="ks")
```

Arguments

`HH`	Response times from the High–High condition.
`HL`	Response times from the High–Low condition.
`LH`	Response times from the Low–High condition.
`LL`	Response times from the Low–Low condition.
`method`	Which type of hypothesis test to use for SIC form.

Details

SIC(t) = (S_LL - S_LH) - (S_HL - S_HH)

This function performs a statistical analysis to determine whether the positive and negative parts of the SIC are significantly different from zero. Currently the only statistical test is based on the generalization of the two-sample Kolmogorov-Smirnov test described in Houpt & Townsend, 2010. This test performs two separate null-hypothesis tests: One test for whether the largest positive value of the SIC is significantly different from zero and one test for whether the largest negative value is significantly different from zero.

Value

positive	A list of class "htest" containing the statistic and p-value along with descriptions of the alternative hypothesis, method and data names for the test of a significant positive portion of the SIC.
negative	A list of class "htest" containing the statistic and p-value along with descriptions of the alternative hypothesis, method and data names for the test of a significant negative portion of the SIC.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Houpt, J.W. & Townsend, J.T. (2010). The statistical properties of the survivor interaction contrast. *Journal of Mathematical Psychology, 54*, 446-453.

See Also

stepfun sicGroup sic mic.test

Examples

```
T1.h <- rexp(50, .2)
T1.l <- rexp(50, .1)
T2.h <- rexp(50, .21)
T2.l <- rexp(50, .11)

SerialAND.hh <- T1.h + T2.h
SerialAND.hl <- T1.h + T2.l
SerialAND.lh <- T1.l + T2.h
SerialAND.ll <- T1.l + T2.l
sic.test(HH=SerialAND.hh, HL=SerialAND.hl, LH=SerialAND.lh, LL=SerialAND.ll)

p1 <- runif(200) < .3
SerialOR.hh <- p1[1:50]    * T1.h + (1-p1[1:50]   )*T2.h
SerialOR.hl <- p1[51:100]  * T1.h + (1-p1[51:100] )*T2.l
SerialOR.lh <- p1[101:150] * T1.l + (1-p1[101:150])*T2.h
SerialOR.ll <- p1[151:200] * T1.l + (1-p1[151:200])*T2.l
sic.test(HH=SerialOR.hh, HL=SerialOR.hl, LH=SerialOR.lh, LL=SerialOR.ll)

ParallelAND.hh <- pmax(T1.h, T2.h)
ParallelAND.hl <- pmax(T1.h, T2.l)
ParallelAND.lh <- pmax(T1.l, T2.h)
ParallelAND.ll <- pmax(T1.l, T2.l)
sic.test(HH=ParallelAND.hh, HL=ParallelAND.hl, LH=ParallelAND.lh, LL=ParallelAND.ll)
```

```
ParallelOR.hh <- pmin(T1.h, T2.h)
ParallelOR.hl <- pmin(T1.h, T2.l)
ParallelOR.lh <- pmin(T1.l, T2.h)
ParallelOR.ll <- pmin(T1.l, T2.l)
sic.test(HH=ParallelOR.hh, HL=ParallelOR.hl, LH=ParallelOR.lh, LL=ParallelOR.ll)
```

`sicGroup` *SIC Analysis for a Group*

Description

Calculates the SIC for each individual in each condition of a DFP experiment. The function will plot each individuals SIC and return the results of the test for stochastic dominance and the statistical test of SIC form.

Usage

```
sicGroup(inData, sictest="ks", mictest=c("art", "anova"), domtest="ks",
         alpha.sic=.05, plotSIC=TRUE, ...)
```

Arguments

`inData`	Data collected from a Double Factorial Paradigm experiment in standard form.
`sictest`	Which type of hypothesis test to use for SIC form. "ks" is the only test currently implemented.
`mictest`	Which type of hypothesis test to use for the MIC. The adjusted rank transform (art) and analysis of variance (anova) are the only tests currently implemented.
`domtest`	Which type of hypothesis test to use for testing stochastic dominance relations, either as series of KS tests ("ks") or the dominance test based on Dirichlet process priors ("dp"). DP not yet implemented.
`alpha.sic`	Alpha level for determining a difference from zero used by the SIC overview.
`plotSIC`	Indicates whether or not to generate plots of the survivor interaction contrasts.
`...`	Arguments to be passed to plot function.

Details

See the help page for the `sic` function for details of the survivor interaction contrast.

Value

`overview`	Data frame summarizing the test outcomes for each participant and condition.
`Subject`	The participant identifier from inData.
`Condition`	The condition identifier from inData.

`Selective.Influence`
: The results of the survivor function dominance test for selective influence. Pass indicates HH < HL, LH and LL > LH, HL, but not HL, LH < HH and not LH, HL > LL, where A < B indicates that A is significantly faster than B at the level of the distribution. Ambiguous means neither HL, LH < HH, nor LH, HL > LL, but at least one of HH < HL, LH or LL >HL, LH did not hold. Fail means that at least one of HL, LH < HH or HL, LH > LL.

`Positive.SIC`
: Indicates whehter the SIC is significantly positive at any time.

`Negative.SIC`
: Indicates whehter the SIC is significantly negative at any time.

`MIC`
: Indicates whether or not the MIC is significantly non-zero.

`Model`
: Indicates which model would predict the pattern of data, assuming selective influence.

`SICfn`
: Matrix with each row giving the values of the of the estimated SIC for one participant in one condition for values of times. The rows match the ordering of statistic.

`sic`
: List with each element giving the result applying sic() to an individual in a condition. sic has the same ordering as overview.

`times`
: Times at which the SICs in SICfn are calculated.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Houpt, J.W. & Townsend, J.T. (2010). The statistical properties of the survivor interaction contrast. *Journal of Mathematical Psychology, 54*, 446-453.

Heathcote, A., Brown, S.D., Wagenmakers, E-J. & Eidels, A. (2010) Distribution-free tests of stochastic dominance for small samples. *Journal of Mathematical Psychology, 54*, 454-463.

Houpt, J.W., Blaha, L.M., McIntire, J.P., Havig, P.R. and Townsend, J.T. (2013). Systems Factorial Technology with R. *Behavior Research Methods*.

See Also

`sic` `capacityGroup`

Examples

```
## Not run:
data(dots)
sicGroup(dots)

## End(Not run)
```

`siDominance` *Dominance Test for Selective Influence*

Description

Function to test for the survivor function ordering predicted by the selective influence of the salience manipulation.

Usage

```
siDominance(HH, HL, LH, LL, method="ks")
```

Arguments

`HH`	Response times from the High–High condition.
`HL`	Response times from the High–Low condition.
`LH`	Response times from the Low–High condition.
`LL`	Response times from the Low–Low condition.
`method`	Which type of hypothesis test to use for testing stochastic dominance relations, either as series of KS tests ("ks") or the dominance test based on Dirichlet process priors ("dp"). DP not yet implemented.

Details

For an SIC function to distinguish among the processing types, the salience manipulation on each channel must selectively influence its respective channel (although see Eidels, Houpt, Altieri, Pei & Townsend, 2010 for SIC prediction from interactive parallel models). Although the selective influence assumption cannot be directly tested, one implication is that the distribution the HH response times stochastically dominates the HL and LH distributions which each in turn stochastically dominate the LL response time distribution. This implication is automatically tested in this function. The KS dominance test uses eight two-sample Kolmogorov-Smirnov tests: HH < HL, HH < LH, HL < LL, LH < LL should be significant while HH > HL, HH > LH, HL > LL, LH > LL should not. The DP uses four tests to determine which relation has the highest Bayes factor assuming a Dirichlet process prior for each of (HH, HL), (HH, LH), (HL, LL) and (LH, LL). See Heathcote, Brown, Wagenmakers & Eidels, 2010, for more details.

Value

A data frame with the first column indicating which ordering was tested, the second column indicating the test statistic and the third indicating the p-value for that value of the statistic.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Townsend, J.T. & Nozawa, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial and coactive theories. *Journal of Mathematical Psychology, 39*, 321-360.

Houpt, J.W. & Townsend, J.T. (2010). The statistical properties of the survivor interaction contrast. *Journal of Mathematical Psychology, 54*, 446-453.

Dzhafarov, E.N., Schweickert, R., & Sung, K. (2004). Mental architectures with selectively influenced but stochastically interdependent components. *Journal of Mathematical Psychology*, 48, 51-64.

See Also

`ks.test` `sic` `sicGroup` `mic.test`

Examples

```
T1.h <- rexp(50, .2)
T1.l <- rexp(50, .1)
T2.h <- rexp(50, .21)
T2.l <- rexp(50, .11)

HH <- T1.h + T2.h
HL <- T1.h + T2.l
LH <- T1.l + T2.h
LL <- T1.l + T2.l
siDominance(HH, HL, LH, LL)
```

---

ucip.id.test *A Statistical Test for Super or Limited Capacity*

Description

A nonparametric test for capacity values significantly different than those predicted by the estimated unlimited capacity, independent parallel model.

Usage

```
ucip.id.test(dt.rt, nt.rt, st.rts, dt.cr=NULL, nt.cr=NULL, st.crs=NULL)
```

Arguments

`dt.rt`	Response times from trials on which all signals indicate targets.
`nt.rt`	Response times from trials on which all signals are either absent or are non-targets.
`st.rts`	A list of arrays of response times from with each list containing response times for trials on which a distinct signal is the only signal present or that is indicating the target response.

dt.cr	A vector of indicators from trials on which all signals indicate targets indicating whether the participant correctly responded to the corresponding trial. If NULL, assumes all are correct.
nt.cr	A vector of indicators from trials on which all signals are either absent or are non-targets indicating whether the participant correctly responded to the corresponding trial. If NULL, assumes all are correct.
st.crs	A list of vectors of indicators from with each list containing response times for trials on which a distinct signal is the only signal present or that is indicating the target response indicating whether the participant correctly responded to the corresponding trial. If NULL, assumes all are correct.

Details

The test is based on the Nelson-Aalen formulation of the log-rank test. The function takes a weighted difference between estimated unlimited capacity, independent parallel performance, based on a participants single source response times, and the participants true performance when all sources are present. Use this statistic with caution. It has not been thorougly tested nor subjected to peer review.

Value

A list of class "htest" containing:

statistic	Z-score of a null-hypothesis test for UCIP performance.
p.val	p-value of a two-tailed null-hypothesis test for UCIP performance.
alternative	A description of the alternative hypothesis.
method	A string indicating whether the ART or ANOVA was used.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

Howard, Z. L., Garrett, P., Little, D. R., Townsend, J. T., & Eidels, A. (2021). A show about nothing: No-signal processes in systems factorial technology. Psychological Review, 128(1), 187-201. doi:https://doi.org/10.1037/rev0000256

See Also

capacity.or capacity.and capacity.id estimateUCIPor estimateUCIPand estimateNAH estimateNAK

Examples

```
rate1p <- .35
rate1a <- .25
rate2p <- .35
rate2a <- .25
RT.pa <- pmax(rexp(100, rate1p), rexp(100, rate2a))
RT.ap <- pmax(rexp(100, rate2p), rexp(100, rate1a))
RT.nt <- pmax(rexp(100, rate1a), rexp(100, rate2a))
RT.pp.limited <- pmax( rexp(100, .5*rate1p), rexp(100, .5*rate2p))
RT.pp.unlimited <- pmax( rexp(100, rate1p), rexp(100, rate2p))
RT.pp.super <- pmax( rexp(100, 2*rate1p), rexp(100, 2*rate2p))

z.limited  <- ucip.id.test(dt.rt=RT.pp.limited,  nt.rt=RT.nt, st.rts=list(RT.pa, RT.ap))
z.unlimited <- ucip.id.test(dt.rt=RT.pp.unlimited, nt.rt=RT.nt, st.rts=list(RT.pa, RT.ap))
z.super    <- ucip.id.test(dt.rt=RT.pp.super,    nt.rt=RT.nt, st.rts=list(RT.pa, RT.ap))
```

---

ucip.test *A Statistical Test for Super or Limited Capacity*

Description

A nonparametric test for capacity values significantly different than those predicted by the estimated unlimited capacity, independent parallel model.

Usage

```
ucip.test(RT, CR=NULL, OR=NULL, stopping.rule=c("OR","AND","STST"))
```

Arguments

RT	A list of response time arrays. The first array in the list is assumed to be the exhaustive condition.
CR	A list of correct/incorrect indicator arrays. If NULL, assumes all are correct.
OR	Indicates whether to compare performance to an OR or AND processing baseline. Provided for backwards compatibility for package version < 2.
stopping.rule	Indicates whether to compare performance to an OR, AND or Single Target Self Terminating (STST) processing baseline.

Details

The test is based on the Nelson-Aalen formulation of the log-rank test. The function takes a weighted difference between estimated unlimited capacity, independent parallel performance, based on a participants single source response times, and the participants true performance when all sources are present.

Value

A list of class "htest" containing:

`statistic`	Z-score of a null-hypothesis test for UCIP performance.
`p.val`	p-value of a two-tailed null-hypothesis test for UCIP performance.
`alternative`	A description of the alternative hypothesis.
`method`	A string indicating whether the ART or ANOVA was used.
`data.name`	A string indicating which data were used for which input.

Author(s)

Joe Houpt <joseph.houpt@utsa.edu>

References

Houpt, J.W. & Townsend, J.T. (2012). Statistical Measures for Workload Capacity Analysis. *Journal of Mathematical Psychology, 56*, 341-355.

See Also

`capacity.or` `capacity.and` `estimateUCIPor` `estimateUCIPand` `estimateNAH` `estimateNAK`

Examples

```
rate1 <- .35
rate2 <- .3
RT.pa <- rexp(100, rate1)
RT.ap <- rexp(100, rate2)

CR.pa <- runif(100) < .98
CR.ap <- runif(100) < .98
CR.pp <- runif(100) < .96
CRlist <- list(CR.pp, CR.pa, CR.ap)

#  OR Processing
RT.pp.limited <- pmin( rexp(100, .5*rate1), rexp(100, .5*rate2))
RT.pp.unlimited <- pmin( rexp(100, rate1), rexp(100, rate2))
RT.pp.super <- pmin( rexp(100, 2*rate1), rexp(100, 2*rate2))
z.limited  <- ucip.test(RT=list(RT.pp.limited, RT.pa, RT.ap), CR=CRlist, stopping.rule="OR")
z.unlimited <- ucip.test(RT=list(RT.pp.unlimited, RT.pa, RT.ap), CR=CRlist, stopping.rule="OR")
z.super    <- ucip.test(RT=list(RT.pp.super, RT.pa, RT.ap), CR=CRlist, stopping.rule="OR")

#  AND Processing
RT.pp.limited <- pmax( rexp(100, .5*rate1), rexp(100, .5*rate2))
RT.pp.unlimited <- pmax( rexp(100, rate1), rexp(100, rate2))
RT.pp.super <- pmax( rexp(100, 2*rate1), rexp(100, 2*rate2))
z.limited  <- ucip.test(RT=list(RT.pp.limited, RT.pa, RT.ap), CR=CRlist, stopping.rule="AND")
z.unlimited <- ucip.test(RT=list(RT.pp.unlimited, RT.pa, RT.ap), CR=CRlist, stopping.rule="AND")
z.super    <- ucip.test(RT=list(RT.pp.super, RT.pa, RT.ap), CR=CRlist, stopping.rule="AND")
```

Index