Package 'DMCfun'

July 2, 2020

Type Package

Title Diffusion Model of Conflict (DMC) in Reaction Time Tasks

Version 0.12.1

Date 2020-06-20

Description DMC model simulation detailed in Ulrich, R., Schroeter, H., Leuthold, H., & Birngruber, T. (2015). Automatic and controlled stimulus processing in conflict tasks: Superimposed diffusion processes and delta functions. Cognitive Psychology, 78, 148-174. Ulrich et al. (2015) <doi:10.1016/j.cogpsych.2015.02.005>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5.0)

Imports Rcpp (>= 0.12.16), dplyr (>= 1.0.0), optimr, parallel, pbapply, tibble, tidyr,

Suggests testthat

LinkingTo Rcpp, BH

RoxygenNote 7.1.0

LazyData true

SystemRequirements C++11

NeedsCompilation yes

Author Mackenzie G. Ian [cre, aut]

Maintainer Mackenzie G. Ian <ian.mackenzie@uni-tuebingen.de>

Repository CRAN

Date/Publication 2020-07-02 07:30:07 UTC

R topics documented:

`DMCfun-package` *DMCfun*

Description

DMC model simulation detailed in Ulrich, R., Schroeter, H., Leuthold, H., & Birngruber, T. (2015). Automatic and controlled stimulus processing in conflict tasks: Superimposed diffusion processes and delta functions. Cognitive Psychology, 78, 148-174. Ulrich et al. (2015) <doi:10.1016/j.cogpsych.2015.02.005>.

Author(s)

Maintainer: Mackenzie G. Ian <ian.mackenzie@uni-tuebingen.de>

addDataDF *addDataDF*

Description

Add simulated ex-gaussian reaction-time (RT) data and binary error (Error = 1, Correct = 0) data to an R DataFrame. This function can be used to create simulated data sets.

Usage

```
addDataDF(dat, RT = NULL, Error = NULL)
```

Arguments

dat	DataFrame (see createDF)
RT	RT parameters (see rtDist)
Error	Error parameters (see errDist)

Value

DataFrame with RT (ms) and Error (bool) columns

Examples

```
# Example 1: default dataframe
dat <- createDF()
dat <- addDataDF(dat)
head(dat)
hist(dat$RT, 100)
table(dat$Error)

# Example 2: defined overall RT parameters
dat <- createDF(nVP = 50, nTrl = 50, design = list("Comp" = c("comp", "incomp")))
dat <- addDataDF(dat, RT = c(500, 150, 100))
boxplot(dat$RT ~ dat$Comp)
table(dat$Comp, dat$Error)

# Example 3: defined RT + Error parameters across conditions
dat <- createDF(nVP = 50, nTrl = 50, design = list("Comp" = c("comp", "incomp")))
dat <- addDataDF(dat,
                 RT = list("Comp_comp"   = c(500, 80, 100),
                           "Comp_incomp" = c(600, 80, 140)),
                 Error = list("Comp_comp"   = 5,
                              "Comp_incomp" = 15))
boxplot(dat$RT ~ dat$Comp)
table(dat$Comp, dat$Error)

# Example 4:
# create dataframe with defined RT + Error parameters across different conditions
```

```
dat <- createDF(nVP = 50, nTrl = 50, design = list("Comp" = c("comp", "incomp", "neutral")))
dat <- addDataDF(dat,
                 RT = list("Comp_comp"      = c(500, 150, 100),
                           "Comp_neutral"   = c(550, 150, 100),
                           "Comp_incomp"    = c(600, 150, 100)),
                 Error = list("Comp_comp"    =  5,
                              "Comp_neutral" = 10,
                              "Comp_incomp"  = 15))
boxplot(dat$RT ~ dat$Comp)
table(dat$Comp, dat$Error)

# Example 5:
# create dataframe with defined RT + Error parameters across different conditions
dat <- createDF(nVP = 50, nTrl = 50,
                design = list("Hand" = c("left", "right"),
                              "Side" = c("left", "right")))
dat <- addDataDF(dat,
                 RT = list("Hand:Side_left:left"   = c(400, 150, 100),
                           "Hand:Side_left:right"  = c(500, 150, 100),
                           "Hand:Side_right:left"  = c(500, 150, 100),
                           "Hand:Side_right:right" = c(400, 150, 100)),
                 Error = list("Hand:Side_left:left"   = c(5,4,2,2,1),
                              "Hand:Side_left:right"  = c(15,4,2,2,1),
                              "Hand:Side_right:left"  = c(15,7,4,2,1),
                              "Hand:Side_right:right" = c(5,8,5,3,1)))

boxplot(dat$RT ~ dat$Hand + dat$Side)
table(dat$Error, dat$Hand, dat$Side)
```

`addErrorBars` *addErrorBars*

Description

Add error bars to current plot (uses base arrows function).

Usage

```
addErrorBars(xpos, ypos, errorSize, arrowSize = 0.1)
```

Arguments

`xpos`	x-position of data-points
`ypos`	y-position of data-points
`errorSize`	+- size of error bars
`arrowSize`	Width of the errorbar arrow

Value

Plot

Examples

```
# Example 1
plot(c(1, 2), c(450, 500), xlim = c(0.5, 2.5), ylim = c(400, 600), type = "o")
addErrorBars(c(1, 2), c(450, 500), errorSize = c(20, 20))

# Example 2
plot(c(1, 2), c(450, 500), xlim = c(0.5, 2.5), ylim = c(400, 600), type = "o")
addErrorBars(c(1, 2), c(450, 500), errorSize = c(20, 40), arrowSize = 0.2)
```

`calculateCAF` *calculateCAF*

Description

Calculate conditional accuracy function (CAF). The DataFrame should contain columns defining the participant, compatibility condition, RT and error (Default column names: "VP", "Comp", "RT", "Error"). The "Comp" column should define compatibility condition (Default: c("comp", "incomp")) and the "Error" column should define if the trial was an error or not (Default: c(0, 1)).

Usage

```
calculateCAF(
  dat,
  stepCAF = 20,
  columns = c("VP", "Comp", "RT", "Error"),
  compCoding = c("comp", "incomp"),
  errorCoding = c(0, 1)
)
```

Arguments

`dat`	DataFrame with columns containing the participant number, condition compatibility, RT data (in ms) and an Error column.
`stepCAF`	Step size for the CAF bins. For example, a step size of 20 would result in 5 CAF bins centered on 10, 30, 50, 70, and 90%.
`columns`	Name of required columns Default: c("VP", "Comp", "RT", "Error")
`compCoding`	Coding for compatibility Default: c("comp", "incomp")
`errorCoding`	Coding for errors Default: c(0, 1))

Value

DataFrame (tibble)

Examples

```
# Example 1
dat <- createDF(nVP = 1, nTrl = 100, design = list("Comp" = c("comp", "incomp")))
dat <- addDataDF(dat,
                 RT = list("Comp_comp"   = c(500, 80, 100),
                           "Comp_incomp" = c(600, 80, 140)),
                 Error = list("Comp_comp"   = c(5, 4,3,2,1),
                              "Comp_incomp" = c(20, 8, 6, 4, 2)))
caf <- calculateCAF(dat)

# Example 2
dat <- createDF(nVP = 1, nTrl = 100, design = list("Congruency" = c("cong", "incong")))
dat <- addDataDF(dat,
                 RT = list("Congruency_cong"   = c(500, 80, 100),
                           "Congruency_incong" = c(600, 80, 140)),
                 Error = list("Congruency_cong"   = c(5, 4,3,2,1),
                              "Congruency_incong" = c(20, 8, 6, 4, 2)))
head(dat)
caf <- calculateCAF(dat, columns = c("VP", "Congruency", "RT", "Error"),
                    compCoding = c("cong", "incong"))
```

`calculateCostValue` *calculateCostValue*

Description

Calculate cost value (fit) from combination of RT and error rate.

Usage

```
calculateCostValue(resTh, resOb)
```

Arguments

`resTh`	list containing caf values for comp/incomp conditions (nbins*2*3) and delta values for comp/incomp conditions (nbins*5). See output from dmcSim (.$caf).
`resOb`	list containing caf values for comp/incomp conditions (n*2*3) and delta values for comp/incomp conditions (nbins*5). See output from dmcSim ($delta).

Value

cost value (RMSE)

Examples

```
# Example 1:
resTh <- dmcSim()
resOb <- dmcSim()
cost  <- calculateCostValue(resTh, resOb)

# Example 2:
resTh <- dmcSim()
resOb <- dmcSim(tau = 150)
cost  <- calculateCostValue(resTh, resOb)
```

`calculateDelta` *calculateDelta*

Description

Calculate delta plot. Here RTs are split into n bins (Default: 5) for compatible and incompatible trials separately. Mean RT is calculated for each condition in each bin then subtracted (incompatible - compatible) to give a compatibility effect (delta) at each bin.

Usage

```
calculateDelta(
  dat,
  stepDelta = 5,
  columns = c("VP", "Comp", "RT"),
  compCoding = c("comp", "incomp"),
  quantileType = 5
)
```

Arguments

`dat`	DataFrame with columns containing the participant number, condition compatibility, and RT data (in ms).
`stepDelta`	Step size for the Delta bins. For example, a step size of 5 would result in 19 CAF bins positioned at 5, 10, 15, ... 85, 90, 95%.
`columns`	Name of required columns Default: c("VP", "Comp", "RT")
`compCoding`	Coding for compatibility Default: c("comp", "incomp")
`quantileType`	Argument (1-9) from R function quantile specifying the algorithm (?quantile)

Value

DataFrame (tibble)

Examples

```
# Example 1
dat <- createDF(nVP = 50, nTrl = 100, design = list("Comp" = c("comp", "incomp")))
dat <- addDataDF(dat,
                 RT = list("Comp_comp"   = c(500, 80, 100),
                           "Comp_incomp" = c(600, 80, 140)))
delta <- calculateDelta(dat)

# Example 2
dat <- createDF(nVP = 50, nTrl = 100, design = list("Congruency" = c("cong", "incong")))
dat <- addDataDF(dat,
                 RT = list("Congruency_cong"   = c(500, 80, 100),
                           "Congruency_incong" = c(600, 80, 140)))
head(dat)
delta <- calculateDelta(dat, columns = c("VP", "Congruency", "RT"),
                        compCoding = c("cong", "incong"))
```

---

createDF *createDF*

Description

Create dataframe (see also addDataDF)

Usage

```
createDF(
  nVP = 20,
  nTrl = 50,
  design = list(A = c("A1", "A2"), B = c("B1", "B2"))
)
```

Arguments

nVP	Number of participants
nTrl	Number of trials per factor/level for each participant
design	Factors and levels

Value

dataframe

Examples

```
# Example 1
dat <- createDF()

# Example 2
dat <- createDF(nVP = 50, nTrl = 50, design = list("Comp" = c("comp", "incomp")))

# Example 3
dat <- createDF(nVP = 50, nTrl = 50, design = list("Comp" = c("comp", "incomp"),
                                                   "Side" = c("left", "right", "middle")))
```

---

dmcCppR *dmcCppR*

Description

dmcCppR

---

dmcFitAgg *dmcFitAgg*

Description

Fit theoretical data generated from dmcSim to observed data by minimizing the root-mean-square error (RMSE) between a weighted combination of the CAF and CDF functions.

Usage

```
dmcFitAgg(
  resOb,
  nTrl = 1e+05,
  startVals = list(),
  minVals = list(),
  maxVals = list(),
  fixedFit = list(),
  fitInitialGrid = TRUE,
  fitInitialGridN = 10,
  fixedGrid = list(),
  stepCAF = 20,
  stepDelta = 5,
  printInputArgs = TRUE,
  printResults = FALSE
)
```

Arguments

resOb	Observed data (see flankerData and simonTask for data format)
nTrl	Number of trials to use within dmcSim.
startVals	Starting values for to-be estimated parameters. This is a list with values specified individually for amp, tau, mu, bnds, resMean, resSD, aaShape, spShape, sigm (e.g., startVals = list(amp = 20, tau = 200, mu = 0.5, bnds = 75, resMean = 300, resSD = 30, aaShape = 2, spShape = 3, sigm = 4)).
minVals	Minimum values for the to-be estimated parameters. This is a list with values specified individually for amp, tau, mu, bnds, resMean, resSD, aaShape, spShape, sigm (e.g., minVals = list(amp = 10, tau = 5, mu = 0.1, bnds = 20, resMean = 200, resSD = 5, aaShape = 1, spShape = 2, sigm = 1)).
maxVals	Maximum values for the to-be estimated parameters. This is a list with values specified individually for amp, tau, mu, bnds, resMean, resSD, aaShape, spShape, sigm (e.g., maxVals = list(amp = 40, tau = 300, mu = 1.0, bnds = 150, resMean = 800, resSD = 100, aaShape = 3, spShape = 4, sigm = 10))
fixedFit	Fix parameter to starting value. This is a list with bool values specified individually for amp, tau, mu, bnds, resMean, resSD, aaShape, spShape, sigm (e.g., fixedFit = list(amp = F, tau = F, mu = F, bnds = F, resMean = F, resSD = F, aaShape = F, spShape = F, sigm = T))
fitInitialGrid	TRUE/FALSE
fitInitialGridN	10 reduce if searching more than 1 initial parameter
fixedGrid	Fix parameter for initial grid search. This is a list with bool values specified individually for amp, tau, mu, bnds, resMean, resSD, aaShape, spShape, sigm (e.g., fixedGrid = list(amp = T, tau = F, mu = T, bnds = T, resMean = T, resSD = T, aaShape = T, spShape = T, sigm = T))
stepCAF	Step size for the CAF bins. For example, a step size of 20 would result in 5 CAF bins centered on 10, 30, 50, 70, and 90%.
stepDelta	Step size for the Delta bins. For example, a step size of 5 would result in 19 CAF bins positioned at 5, 10, 15, ... 85, 90, 95%.
printInputArgs	TRUE/FALSE
printResults	TRUE/FALSE

Value

dmcfit

The function returns a list with the relevant results from the fitting procedure. The list is accessed with obj$name with the the following:

obj$means	Condition means for reaction time and error rate
obj$caf	Accuracy per bin for compatible and incompatible trials
obj$delta	Mean RT and compatibility effect per bin
obj$sim	Individual trial data points (reaction times/error) and activation vectors from simulation
obj$par	The fitted model parameters + final RMSE of the fit

Examples

```
# Example 1: Flanker data from Ulrich et al. (2015)
fit <- dmcFitAgg(flankerData)  # only initial search tau
plot(fit, flankerData)
summary(fit)

# Example 2: Simon data from Ulrich et al. (2015)
fit <- dmcFitAgg(simonData)    # only initial search tau
plot(fit, simonData)
summary(fit)

# Example 3: Flanker data from Ulrich et al. (2015) with non-default
# start vals and some fixed values
fit <- dmcFitAgg(flankerData,
                 startVals = list(mu = 0.6, aaShape = 2.5),
                 fixedFit = list(mu = TRUE, aaShape = TRUE))

# Example 4: Simulated Data (+ve going delta function)
dat <- createDF(nVP = 20, nTrl = 500, design = list("Comp" = c("comp", "incomp")))
dat <- addDataDF(dat,
                 RT = list("Comp_comp"   = c(510, 100, 100),
                           "Comp_incomp" = c(540, 130, 85)),
                 Error = list("Comp_comp"   = c(4, 3, 2, 1, 1),
                              "Comp_incomp" = c(20, 4, 3, 1, 1)))
datOb <- dmcObservedData(dat, columns = c("VP", "Comp", "RT", "Error"))
plot(datOb)
fit <- dmcFitAgg(datOb, nTrl = 5000)
plot(fit, datOb)
summary(fit)
```

`dmcFitVPs` *dmcFitVPs*

Description

Fit theoretical data generated from dmcSim to observed data by minimizing the root-mean-square error (RMSE) between a weighted combination of the CAF and CDF functions.

Usage

```
dmcFitVPs(
  resOb,
  nTrl = 1e+05,
  startVals = list(),
  minVals = list(),
  maxVals = list(),
```

```
  fixedFit = list(),
  fitInitialGrid = TRUE,
  fitInitialGridN = 10,
  fixedGrid = list(),
  stepCAF = 20,
  stepDelta = 5,
  VP = c(),
  printInputArgs = TRUE,
  printResults = FALSE
)
```

Arguments

resOb	Observed data (see flankerData, simonData for data format)
nTrl	Number of trials to use within dmcSim
startVals	Starting values for to-be estimated parameters
minVals	Minimum values for the to-be estimated parameters
maxVals	Maximum values for the to-be estimated parameters
fixedFit	Fix parameter to starting value
fitInitialGrid	TRUE/FALSE (NB. overrides fitInitialTau)
fitInitialGridN	10
fixedGrid	Fix parameter for initial grid search
stepCAF	Step size for the CAF bins. For example, a step size of 20 would result in 5 CAF bins centered on 10, 30, 50, 70, and 90%.
stepDelta	Step size for the Delta bins. For example, a step size of 5 would result in 19 CAF bins positioned at 5, 10, 15, ... 85, 90, 95%.
VP	NULL (aggregated data across all participants) or integer for participant number
printInputArgs	TRUE/FALSE
printResults	TRUE/FALSE

Value

dmcfitvp List of dmcfit per participant fitted (see dmcFitAgg)

Examples

```
# Example 1: Flanker data from Ulrich et al. (2015)
fit <- dmcFitVPs(flankerData, nTrl = 1000, VP = c(1, 2))
plot(fit, flankerData, VP = 1)
plot(fit, flankerData, VP = 2)
summary(fit)
fitAgg <- mean(fit)
plot(fitAgg, flankerData)
```

```
# Example 2: Simon data from Ulrich et al. (2015)
fit <- dmcFitVPs(simonData, nTrl = 1000, VP = c(1, 2))
plot(fit, simonData, VP = 1)
plot(fit, simonData, VP = 2)
summary(fit)
fitAgg <- mean(fit)
plot(fitAgg, simonData)
```

---

dmcObservedData *dmcObservedData*

Description

Basic example analysis script to create data object required for observed data. Example raw *.txt files are flankerData.txt and simonData.txt. There are four critical columns: A column containing participant number A column coding for compatible or incompatible A column with RT (in ms) A column indicating of the response was correct

Usage

```
dmcObservedData(
  dat,
  stepCAF = 20,
  stepDelta = 5,
  outlier = c(200, 1200),
  columns = c("VP", "Comp", "RT", "Error"),
  compCoding = c("comp", "incomp"),
  errorCoding = c(0, 1),
  quantileType = 5,
  delim = "\t",
  skip = 0
)
```

Arguments

dat	Text file(s) containing the observed data or an R DataFrame (see createDF/addDataDF)
stepCAF	Step size for the CAF bins. For example, a step size of 20 would result in 5 CAF bins centered on 10, 30, 50, 70, and 90%.
stepDelta	Step size for the Delta bins. For example, a step size of 5 would result in 19 CAF bins positioned at 5, 10, 15, ... 85, 90, 95%.
outlier	Outlier limits in ms (e.g., c(200, 1200))
columns	Name of required columns DEFAULT = c("VP", "Comp", "RT", "Error")
compCoding	Coding for compatibility DEFAULT = c("comp", "incomp")
errorCoding	Coding for errors DEFAULT = c(0, 1))

quantileType	Argument (1-9) from R function quantile specifying the algorithm (?quantile)
delim	Single character used to separate fields within a record if reading from external text file.
skip	Number of lines to skip before reading data if reading from external text file.

Value

DataFrame

Examples

```
# Example 1
plot(flankerData)  # flanker data from Ulrich et al. (2015)

# Example 2
plot(simonData)    # simon data from Ulrich et al. (2015)

# Example 3 (Basic behavioural analysis from Ulrich et al. 2015)
flankerDat <- tibble::add_column(Task = "flanker", flankerData$summaryVP, .before = 2)
simonDat   <- tibble::add_column(Task = "simon",   simonData$summaryVP,   .before = 2)
datAgg     <- rbind(rbind(flankerDat, simonDat))

datAgg$VP   <- factor(datAgg$VP)
datAgg$Task <- factor(datAgg$Task)
datAgg$Comp <- factor(datAgg$Comp)

aovErr <- aov(perErr ~ Comp*Task + Error(VP/(Comp*Task)), datAgg)
summary(aovErr)
model.tables(aovErr, type = "mean")

aovRt <- aov(rtCor ~ Comp*Task + Error(VP/(Comp*Task)), datAgg)
summary(aovRt)
model.tables(aovRt, type = "mean")

# Example 4
dat <- createDF(nVP = 50, nTrl = 500, design = list("Comp" = c("comp", "incomp")))
dat <- addDataDF(dat,
                 RT = list("Comp_comp"    = c(500, 75, 120),
                           "Comp_incomp"  = c(530, 75, 100)),
                 Error = list("Comp_comp" = c(3, 2, 2, 1, 1),
                              "Comp_incomp" = c(21, 3, 2, 1, 1)))
datOb <- dmcObservedData(dat)
plot(datOb)
plot(datOb, VP = 1)

# Example 5
dat <- createDF(nVP = 50, nTrl = 500, design = list("Congruency" = c("cong", "incong")))
dat <- addDataDF(dat,
                 RT = list("Congruency_cong"   = c(500, 75, 100),
                           "Congruency_incong" = c(530, 100, 110)),
                 Error = list("Congruency_cong"   = c(3, 2, 2, 1, 1),
                              "Congruency_incong" = c(21, 3, 2, 1, 1)))
```

```
datOb <- dmcObservedData(dat, stepCAF = 20, stepDelta = 10,
                         columns = c("VP", "Congruency", "RT", "Error"),
                         compCoding = c("cong", "incong"))
plot(datOb, labels = c("Congruent", "Incongruent"))
plot(datOb, VP = 1)
```

---

dmcSim *dmcSim*

Description

DMC model simulation detailed in Ulrich, R., Schroeter, H., Leuthold, H., & Birngruber, T. (2015). Automatic and controlled stimulus processing in conflict tasks: Superimposed diffusion processes and delta functions. Cognitive Psychology, 78, 148-174. This function is essentially a wrapper around the c++ function runDMC

Usage

```
dmcSim(
  amp = 20,
  tau = 30,
  mu = 0.5,
  bnds = 75,
  resMean = 300,
  resSD = 30,
  aaShape = 2,
  spShape = 3,
  sigm = 4,
  nTrl = 1e+05,
  tmax = 1000,
  varSP = FALSE,
  spLim = c(-75, 75),
  varDR = FALSE,
  drShape = 3,
  drLim = c(0.1, 0.7),
  fullData = FALSE,
  nTrlData = 5,
  stepDelta = 5,
  stepCAF = 20,
  printInputArgs = TRUE,
  printResults = TRUE,
  setSeed = FALSE
)
```

Arguments

amp	amplitude of automatic activation

tau	time to peak automatic activation
mu	drift rate of controlled processes
bnds	+- response criterion
resMean	mean of non-decisional component
resSD	standard deviation of non-decisional component
aaShape	shape parameter of automatic activation
spShape	shape parameter of starting point
sigm	diffusion constant
nTrl	number of trials
tmax	number of time points per trial
varSP	true/false variable starting point
spLim	limit range of distribution of starting point
varDR	true/false variable drift rate NB. In DMC, drift rate across trials is always constant.
drShape	shape parameter of drift rate
drLim	limit range of distribution of drift rate
fullData	TRUE/FALSE (Default: FALSE)
nTrlData	Number of trials to plot
stepDelta	Number of delta bins
stepCAF	Number of CAF bins
printInputArgs	TRUE/FALSE
printResults	TRUE/FALSE
setSeed	TRUE/FALSE

Value

dmcsim

The function returns a list with the relevant results from the simulation. The list is accessed with obj$name with the the following:

obj$means	Condition means for reaction time and error rate
obj$caf	Accuracy per bin for compatible and incompatible trials
obj$delta	Mean RT and compatibility effect per bin
obj$sim	Individual trial data points (reaction times/error) and activation vectors from simulation
obj$trials	Example individual trial timecourse for n compatible and incompatible trials
obj$prms	The input parameters used in the simulation

Examples

```
# Example 1
dmc <- dmcSim(fullData = TRUE)  # full data only required for activation plot (top left)
plot(dmc)
dmc <- dmcSim() # faster
plot(dmc)

# Example 2
dmc <- dmcSim(tau = 130)
plot(dmc)

# Example 3
dmc <- dmcSim(tau = 90)
plot(dmc)

# Example 4
dmc <- dmcSim(varSP = TRUE)
plot(dmc, "delta")

# Example 5
dmc <- dmcSim(tau = 130, varDR = TRUE)
plot(dmc, "caf")

# Example 6
dmc <- dmcSim(stepDelta = 10, stepCAF = 10)
plot(dmc)
```

`dmcSims` *dmcSims*

Description

Run dmcSim with range of input parameters.

Usage

```
dmcSims(params, printInputArgs = FALSE, printResults = FALSE)
```

Arguments

`params`	(list of parameters to dmcSim)
`printInputArgs`	Print DMC input arguments to console
`printResults`	Print DMC output to console

Value

list of dmcsim

Examples

```
# Example 1
params <- list(amp = seq(10, 20, 5), tau = c(50, 100, 150), nTrl = 50000)
dmc <- dmcSims(params)
plot(dmc[[1]])    # full combination 1
plot(dmc)         # delta plots for all combinations
plot(dmc[c(1:3)]) # delta plots for specific combinations

# Example 2
params <- list(amp = seq(10, 20, 5), tau = seq(20, 40, 20), bnds = seq(50, 100, 25))
dmc <- dmcSims(params)
plot(dmc[[1]])  # combination 1
plot(dmc, ncol = 2)       # delta plots for all combinations
plot(dmc[c(1:3)]) # delta plots for specific combinations
```

`errDist` *errDist*

Description

Returns a random vector of 0's (correct) and 1's (incorrect) with defined proportions (default = 10% errors).

Usage

```
errDist(n = 10000, proportion = 10)
```

Arguments

`n`	Number
`proportion`	Approximate proportion of errors in percentage

Value

double

Examples

```
# Example 1
x <- errDist(1000, 10)
table(x)
```

`flankerData` *A summarised dataset: see raw data file flankerDataRaw and dmcObservedData.R This is the summarised Flanker Task data from Ulrich et al. (2015)*

Description

- $summary –> Reaction time correct, standard deviation correct, percentage error, reaction time incorrect, and standard deviation for incorrect trials for both compatible and incompatible trials
- $caf –> Proportion correct for compatible and incompatible trials across 5 bins
- $delta –> Compatible reactions times, incompatible mean reaction times, mean reaction times, incompatible - compatible reaction times (delta), and standard deviation + standard error of this difference across 10 bins.

Usage

```
flankerData
```

Format

dmcob

`flankerDataRaw` *Raw flanker data from Ulrich et al. (2015)*

Description

- VP Subject number
- Comp comp vs. incomp
- RT
- Error 0 = correct, 1 = error

Usage

```
flankerDataRaw
```

`mean.dmcfitvp` *mean.dmcfitvp*

Description

Aggregated simulation results from dmcFitVPs.

Usage

```
## S3 method for class 'dmcfitvp'
mean(x, ...)
```

Arguments

`x`	Output from dmcFitVPs
`...`	pars

Value

dmcfit

The function returns a list with the relevant aggregated results dmcFitVPs. The list is accessed with obj$name and so on with the the following:

`obj$means`	means
`obj$delta`	delta
`obj$caf`	caf
`obj$prms`	par

Examples

```
# Example 1: Fit individual data then aggregate
fitVPs <- dmcFitVPs(flankerData, nTrl = 1000, VP = c(2))
fitAgg <- mean(fitVPs)
plot(fitAgg, flankerData)
```

plot.dmcfit *plot.dmcfit*

Description

Plot the simulation results from the output of dmcFitAgg. The plot can be an overall summary, or individual plots (activation, trials, pdf, cdf, caf, delta, all). Plot type summary1 contains an activation plot, example individual trials, the probability distribution function (PDF), the cumulative distribution function (CDF), the conditional accuracy function (CAF) and delta plots. This required that dmcSim is run with fullData = TRUE. Plot type summary2 contains only the PDF, CDF, CAF and delta plots and does not require that dmcSim is run with fullData = TRUE.

Usage

```
## S3 method for class 'dmcfit'
plot(
  x,
  y,
  figType = "summary",
  legend = TRUE,
  labels = c("Compatible", "Incompatible", "Observed", "Predicted"),
  cols = c("black", "green", "red"),
  ylimRt = c(200, 800),
  ylimEr = c(0, 20),
  ylimCAF = c(0, 1),
  cafBinLabels = FALSE,
  ylimDelta = c(-50, 100),
  xlimDelta = c(200, 1000),
  xlabs = TRUE,
  ylabs = TRUE,
  xaxts = TRUE,
  yaxts = TRUE,
  ...
)
```

Arguments

x	Output from dmcFitAgg
y	Observed data
figType	summary, rtCorrect, errorRate, rtErrors, cdf, caf, delta, all
legend	TRUE/FALSE (or FUNCTION) plot legend on each plot
labels	Condition labels c("Compatible", "Incompatible", "Observed", "Predicted") default
cols	Condition colours c("green", "red") default
ylimRt	ylimit for Rt plots

ylimEr	ylimit for error rate plots
ylimCAF	ylimit for CAF plot
cafBinLabels	TRUE/FALSE
ylimDelta	ylimit for delta plot
xlimDelta	xlimit for delta plot
xlabs	TRUE/FALSE
ylabs	TRUE/FALSE
xaxts	TRUE/FALSE
yaxts	TRUE/FALSE
...	additional plot pars

Examples

```
# Example 1
resTh <- dmcFitAgg(flankerData, nTrl = 5000)
plot(resTh, flankerData)

# Example 2
resTh <- dmcFitAgg(flankerData, nTrl = 5000)
plot(resTh, flankerData)
plot(resTh, flankerData, figType = "all")

# Example 3
resTh <- dmcFitAgg(simonData, nTrl = 5000)
plot(resTh, simonData)
```

`plot.dmcfitvp` *plot.dmcfitvp*

Description

Plot the simulation results from the output of dmcFitVPs. The plot can be an overall summary, or individual plots (activation, trials, pdf, cdf, caf, delta, all). Plot type summary1 contains an activation plot, example individual trials, the probability distribution function (PDF), the cumulative distribution function (CDF), the conditional accuracy function (CAF) and delta plots. This required that dmcSim is run with fullData = TRUE. Plot type summary2 contains only the PDF, CDF, CAF and delta plots and does not require that dmcSim is run with fullData = TRUE.

Usage

```
## S3 method for class 'dmcfitvp'
plot(
  x,
  y,
  figType = "summary",
  VP = NULL,
  legend = TRUE,
  labels = c("Compatible", "Incompatible", "Observed", "Predicted"),
  cols = c("black", "green", "red"),
  ylimRt = c(200, 800),
  ylimEr = c(0, 20),
  ylimCAF = c(0, 1),
  cafBinLabels = FALSE,
  ylimDelta = c(-50, 100),
  xlimDelta = c(200, 1000),
  xlabs = TRUE,
  ylabs = TRUE,
  xaxts = TRUE,
  yaxts = TRUE,
  ...
)
```

Arguments

x	Output from dmcFitVPs
y	Observed data
figType	summary, rtCorrect, errorRate, rtErrors, cdf, caf, delta, all
VP	NULL (aggregated data across all participants) or integer for participant number
legend	TRUE/FALSE (or FUNCTION) plot legend on each plot
labels	Condition labels c("Compatible", "Incompatible", "Observed", "Predicted") default
cols	Condition colours c("green", "red") default
ylimRt	ylimit for Rt plots
ylimEr	ylimit for error rate plots
ylimCAF	ylimit for CAF plot
cafBinLabels	TRUE/FALSE
ylimDelta	ylimit for delta plot
xlimDelta	xlimit for delta plot
xlabs	TRUE/FALSE
ylabs	TRUE/FALSE
xaxts	TRUE/FALSE
yaxts	TRUE/FALSE
...	additional plot pars

Examples

```
# Example 1
resTh <- dmcFitVPs(flankerData, nTrl = 5000, VP = c(2, 3))
plot(resTh, flankerData)
plot(flankerData, VP = 2)
plot(resTh, flankerData)
```

---

plot.dmclist *plot.dmclist*

Description

Plot the simulation results from the output of dmcSim. The plot can be an overall summary, or individual plots (activation, trials, pdf, cdf, caf, delta, all). Plot type summary1 contains an activation plot, example individual trials, the probability distribution function (PDF), the cumulative distribution function (CDF), the conditional accuracy function (CAF) and delta plot. This requires that dmcSim is run with fullData = TRUE. Plot type summary2 contains only the PDF, CDF, CAF and delta plots and does not require that dmcSim is run with fullData = TRUE.

Usage

```
## S3 method for class 'dmclist'
plot(
  x,
  ylim = c(-50, 150),
  xlim = NULL,
  col = c("black", "lightgrey"),
  lineType = "l",
  legendPos = "topleft",
  ncol = 1,
  ...
)
```

Arguments

x	Output from dmcSims
ylim	ylimit for delta plot
xlim	xlimit for delta plot
col	# color range start/end color
lineType	line type ("l", "b", "o") for delta plot
legendPos	legend position
ncol	number of legend columns
...	pars for legend

Examples

```
# Example 1
params <- list(amp = seq(20, 30, 2))
dmc <- dmcSims(params)
plot(dmc, ncol = 2, xlim = c(0, 1300), ylim = c(-100, 200))

# Example 2
params <- list(amp=c(10, 20), tau = seq(20, 80, 40), mu = seq(0.2, 0.6, 0.2), nTrl = 50000)
dmc <- dmcSims(params)
plot(dmc, ncol = 2, col=c("green", "blue"), lineType = "l")
```

`plot.dmcob` *plot.dmcob*

Description

Plot results from the output of dmcSim. The plot can be an overall summary, or individual plots (activation, trials, pdf, cdf, caf, delta, all). Plot type summary1 contains an activation plot, example individual trials, the probability distribution function (PDF), the cumulative distribution function (CDF), the conditional accuracy function (CAF) and delta plots. This required that dmcSim is run with fullData = TRUE. Plot type summary2 contains only the PDF, CDF, CAF and delta plots and does not require that dmcSim is run with fullData = TRUE.

Usage

```
## S3 method for class 'dmcob'
plot(
  x,
  figType = "summary",
  VP = NULL,
  legend = TRUE,
  labels = c("Compatible", "Incompatible"),
  cols = c("black", "green", "red"),
  errorBars = FALSE,
  errorBarType = "sd",
  ylimRt = c(200, 800),
  ylimEr = c(0, 20),
  ylimCAF = c(0, 1),
  cafBinLabels = FALSE,
  ylimDelta = c(-50, 100),
  xlimDelta = c(200, 1000),
  xlabs = TRUE,
  ylabs = TRUE,
```

```
  xaxts = TRUE,
  yaxts = TRUE,
  ...
)
```

Arguments

x	Output from fitDMC
figType	summary, rtCorrect, errorRate, rtErrors, cdf, caf, delta, all
VP	NULL (aggregated data across all participants) or integer for participant number
legend	TRUE/FALSE (or FUNCTION) plot legend on each plot
labels	Condition labels c("Compatible", "Incompatible") default
cols	Condition colours c("green", "red") default
errorBars	TRUE(default)/FALSE Plot errorbars
errorBarType	sd(default), or se
ylimRt	ylimit for Rt plots
ylimEr	ylimit for error rate plots
ylimCAF	ylimit for CAF plot
cafBinLabels	TRUE/FALSE
ylimDelta	ylimit for delta plot
xlimDelta	xlimit for delta plot
xlabs	TRUE/FALSE
ylabs	TRUE/FALSE
xaxts	TRUE/FALSE
yaxts	TRUE/FALSE
...	additional plot pars

Examples

```
# Example 1 (real dataset)
plot(flankerData)
plot(flankerData, errorBars = TRUE, errorBarType = "se")
plot(flankerData, figType = "delta")
plot(flankerData, figType = "caf")

# Example 2 (real dataset)
plot(simonData)
plot(simonData, errorBars = TRUE, errorBarType = "se")
plot(simonData, figType = "delta", errorBars = TRUE, errorBarType = "sd")

# Example 3 (simulated dataset)
dat <- createDF(nVP = 50, nTrl = 50,
                design = list("Comp" = c("comp", "incomp")))
dat <- addDataDF(dat,
```

```
                  RT = list("Comp_comp"   = c(420, 100, 80),
                            "Comp_incomp" = c(470, 100, 95)),
                  Error = list("Comp_comp"   = c(5, 3, 2, 1, 2),
                               "Comp_incomp" = c(15, 8, 4, 2, 2)))
datOb <- dmcObservedData(dat)
plot(datOb, errorBars = TRUE, errorBarType = "sd")

# Example 4 (simulated dataset)
dat <- createDF(nVP = 50, nTrl = 50,
                design = list("Comp" = c("comp", "incomp")))
dat <- addDataDF(dat,
                 RT = list("Comp_comp"   = c(420, 100, 150),
                           "Comp_incomp" = c(470, 100, 120)),
                 Error = list("Comp_comp"   = c(5, 3, 2, 1),
                              "Comp_incomp" = c(15, 8, 4, 2)))
datOb <- dmcObservedData(dat, stepCAF = 25)
plot(datOb)
```

---

plot.dmcsim *plot.dmcsim*

Description

Plot the simulation results from the output of dmcSim. The plot can be an overall summary, or individual plots (activation, trials, pdf, cdf, caf, delta, all). Plot type summary1 contains an activation plot, example individual trials, the probability distribution function (PDF), the cumulative distribution function (CDF), the conditional accuracy function (CAF) and delta plot. This requires that dmcSim is run with fullData = TRUE. Plot type summary2 contains only the PDF, CDF, CAF and delta plots and does not require that dmcSim is run with fullData = TRUE.

Usage

```
## S3 method for class 'dmcsim'
plot(
  x,
  figType = "summary1",
  ylimCAF = c(0, 1),
  cafBinLabels = FALSE,
  ylimDelta = c(-50, 150),
  ylimRt = c(200, 800),
  ylimErr = c(0, 20),
  legend = TRUE,
  labels = c("Compatible", "Incompatible"),
  cols = c("black", "green", "red"),
  errorBars = FALSE,
  xlabs = TRUE,
```

```
  ylabs = TRUE,
  xaxts = TRUE,
  yaxts = TRUE,
  ...
)
```

Arguments

x	Output from dmcSim
figType	summary1, summary2, summary3, activation, trials, pdf, cdf, caf, delta, rtCorrect, rtErrors, errorRate, all
ylimCAF	ylimit for CAF plot
cafBinLabels	TRUE/FALSE
ylimDelta	ylimit for delta plot
ylimRt	ylimit for rt plot
ylimErr	ylimit for er plot
legend	TRUE/FALSE (or FUNCTION) plot legend on each plot
labels	Condition labels c("Compatible", "Incompatible") default
cols	Condition colours c("green", "red") default
errorBars	TRUE/FALSE
xlabs	TRUE/FALSE
ylabs	TRUE/FALSE
xaxts	TRUE/FALSE
yaxts	TRUE/FALSE
...	additional plot pars

Examples

```
# Example 1
dmc = dmcSim(fullData = TRUE)
plot(dmc)

# Example 2
dmc = dmcSim()
plot(dmc)

# Example 3
dmc = dmcSim()
plot(dmc, figType = "all")

# Example 4
dmc = dmcSim()
plot(dmc, figType = "summary3")
```

rtDist *rtDist*

Description

Returns value(s) from a distribution appropriate to simulate reaction times. The distribution is a combined exponential and gaussian distribution called an exponentially modified Gaussian (EMG) distribution or ex-gaussian distribution.

Usage

```
rtDist(n = 10000, gaussMean = 600, gaussSD = 50, expRate = 200)
```

Arguments

n	Number of observations
gaussMean	Mean of the gaussian distribution
gaussSD	SD of the gaussian distribution
expRate	Rate of the exponential function

Value

double

Examples

```
# Example 1
x <- rtDist()
hist(x, 100)

# Example 2
x <- rtDist(n=20000, gaussMean=800, gaussSD=50, expRate=100)
hist(x, 100)
```

---

simonData *A summarised dataset: see raw data file simonDataRaw and dmcObservedData.R This is the summarised Simon Task data from Ulrich et al. (2015)*

Description

- $summary –> Reaction time correct, standard deviation correct, percentage error, reaction time incorrect, and standard deviation for incorrect trials for both compatible and incompatible trials
- $caf –> Proportion correct for compatible and incompatible trials across 5 bins
- $delta –> Compatible reactions times, incompatible mean reaction times, mean reaction times, incompatible - compatible reaction times (delta), and standard deviation + standard error of this difference across 10 bins.

Usage

```
simonData
```

Format

dmcob

`simonDataRaw` *Raw simon data from Ulrich et al. (2015)*

Description

- VP Subject number
- Comp comp vs. incomp
- RT
- Error 0 = correct, 1 = error

Usage

```
simonDataRaw
```

`summary.dmcfit` *summary.dmcfit*

Description

Summary of the simulation results from dmcFitAgg

Usage

```
## S3 method for class 'dmcfit'
summary(object, digits = 2, ...)
```

Arguments

object	Output from dmcFitAgg
digits	Number of digits in the output
...	pars

Value

DataFrame (tibble)

Examples

```
# Example 1
fitAgg <- dmcFitAgg(flankerData, nTrl = 1000)
summary(fitAgg)
```

---

summary.dmcfitvp *summary.dmcfitvp*

Description

Summary of the simulation results from dmcFitVPs

Usage

```
## S3 method for class 'dmcfitvp'
summary(object, digits = 2, ...)
```

Arguments

object	Output from dmcFitVPs
digits	Number of digits in the output
...	pars

Value

list of DataFrames (tibbles) with the first being individual participant fitted parameters and the second being the mean fitted parameters

Examples

```
# Example 1
fitVPs <- dmcFitVPs(flankerData, nTrl = 1000, VP = c(1, 10))
summary(fitVPs)
fit <- mean(fitVPs)
```

---

summary.dmcsim *summary.dmcsim*

Description

Summary of the overall results from dmcSim

Usage

```
## S3 method for class 'dmcsim'
summary(object, digits = 1, ...)
```

Arguments

object	Output from dmcSim
digits	Number of digits in the output
...	pars

Value

list

Examples

```
# Example 1
dmc <- dmcSim()
summary(dmc)

# Example 2
dmc <- dmcSim(tau = 90)
summary(dmc)
```

Index