Package ‘echoice2’
November 20, 2023

Type Package
Title Choice Models with Economic Foundation
Version 0.2.4
Date 2023-11-20
Maintainer Nino Hardt <me@ninohardt.com>
License MIT + file LICENSE
BugReports https://github.com/ninohardt/echoice2/issues
URL https://github.com/ninohardt/echoice2,
http://ninohardt.de/echoice2/
Imports Rcpp, parallel, magrittr, stats, graphics, stringr, purrr, tibble, tidyselect, tidyr, rlang, forcats
Depends R (>= 3.5), dplyr, ggplot2
Suggests knitr, rmarkdown, testthat, bayesm
LinkingTo Rcpp, RcppArmadillo
Config/testthat/edition 3
Encoding UTF-8
RoxygenNote 7.2.3
VignetteBuilder knitr
NeedsCompilation yes
Author Nino Hardt [aut, cre] (<https://orcid.org/0000-0001-5215-6216>)
Repository CRAN
Date/Publication 2023-11-20 08:50:07 UTC
R topics documented:

- dd_dem
- dd_dem_sr
- dd_est_hmnl
- dd_est_hmnl_screen
- dd_LL
- dd_LL_sr
- dummify
- dummyvar
- ec_boxplot_MU
- ec_boxplot_screen
- ec_demcurve
- ec_demcurve_cond_dem
- ec_demcurve_inci
- ec_dem_aggregate
- ec_dem_eval
- ec_dem_summarise
- ec_draws_MU
- ec_draws_screen
- ec_estimates_MU
- ec_estimates_screen
- ec_estimates_SIGMA
- ec_estimates_SIGMA_corr
- ec_gen_err_ev1
- ec_gen_err_normal
- ec_lmd_NR
- ec_lol_tidy1
- ec_screenprob_sr
- ec_screen_summarise
- ec_summarize_attrlvl
- ec_trace_MU
- ec_trace_screen
- ec_undummy
- ec_undummy_lowhigh
- ec_undummy_lowmediumhigh
- ec_undummy_yesno
- ec_util_choice_to_long
- ec_util_dummy_mutualeclusive
- get_attr_lvl
- icecream
- icecream_discrete
- logMargDenNRu
- pizza
- prep_newprediction
- vd_add_prodid
- vd_dem_summarise
- vd_dem_vdm
Description

Discrete Choice Predictions (HMNL)

Usage

\[ \text{dd_dem}(\text{dd}, \text{est}, \text{prob} = \text{FALSE}, \text{cores} = \text{NULL}) \]

Arguments

- \text{dd} \quad \text{tibble with long-format choice data}
- \text{est} \quad \text{estimation object}
- \text{prob} \quad \text{logical, report probabilities instead of demand}
- \text{cores} \quad \text{cores}

Value

Draws of expected choice

See Also

\[ \text{dd_est_hmnl()} \] to generate demand predictions based on this model
Examples

```r
data(icecream_discrete)
icecream_est <- icecream_discrete %>% filter(id<10) %>% dd_est_hmnl(R=4, cores=2)
  # demand prediction
icecream_dempred <- icecream_discrete %>% filter(id<10) %>% 
  dd_dem(icecream_est, cores=2)
```

---

**dd_dem_sr**  
*Discrete Choice Predictions (HMNL with attribute-based screening)*

**Description**  
Discrete Choice Predictions (HMNL with attribute-based screening)

**Usage**  
`dd_dem_sr(dd, est, prob = FALSE, cores = NULL)`

**Arguments**  
- `dd` data
- `est` est
- `prob` logical, report probabilities instead of demand
- `cores` cores

**Value**  
Draws of expected choice

**See Also**  
`dd_est_hmnl_screen()` to generate demand predictions based on this model

**Examples**

```r
data(icecream_discrete)
icecream_est <- icecream_discrete %>% filter(id<20) %>% dd_est_hmnl_screen(R=10, cores=2)
  # demand prediction
icecream_dempred <- icecream_discrete %>% filter(id<20) %>% 
  dd_dem_sr(icecream_est, cores=2)
```
dd_est_hmnl

Estimate discrete choice model (HMNL)

Description

Estimate discrete choice model (HMNL)

Usage

dd_est_hmnl(
  dd,
  R = 1e+05,
  keep = 10,
  cores = NULL,
  control = list(include_data = TRUE)
)

Arguments

- **dd**: discrete choice data (long format)
- **R**: draws
- **keep**: thinning
- **cores**: no of CPU cores to use (default: auto-detect)
- **control**: list containing additional settings

Value

est ec-draw object (List)

See Also

- `dd_dem()` to generate demand predictions based on this model

Examples

```r
data(icecream_discrete)
icecream_est <- icecream_discrete %>% dd_est_hmnl(R=20, cores=2)
```
dd_est_hmnl_screen

Estimate discrete choice model (HMNL, attribute-based screening (not including price))

Description

Estimate discrete choice model (HMNL, attribute-based screening (not including price))

Usage

dd_est_hmnl_screen(
  dd,
  price_screen = TRUE,
  R = 1e+05,
  keep = 10,
  cores = NULL,
  control = list(include_data = TRUE)
)

Arguments

dd       discrete choice data (long format)
price_screen A logical, indicating whether price tag screening should be estimated
R         draws
keep      thinning
cores     no of CPU cores to use (default: auto-detect)
control   list containing additional settings

Value

est ec-draw object (List)

See Also

dd_dem_sr() to generate demand predictions based on this model

Examples

data(icecream_discrete)
icecream_est <- icecream_discrete %>% dplyr::filter(id<20) %>%
  dd_est_hmnl_screen(R=20, cores=2)
dd_LL

Log-Likelihood for compensatory hmnl model

Description

Log-Likelihood for compensatory hmnl model

Usage

dd_LL(draw, dd, fromdraw = 1)

Arguments

draw
A list, 'echoice2' draws object

dd
A tibble, tidy choice data (before dummy-coding)

fromdraw
An integer, from which draw onwards to compute LL (i.e., excl. burnin)

Value

N x Draws Matrix of log-Likelihood values

Examples

data(icecream_discrete)

# fit model
icecream_est <- icecream_discrete %>% dd_est_hmnl(R=10, keep=1, cores=2)

# compute likelihood for each subject in each draw
loglls<-dd_LL(icecream_est, icecream_discrete, fromdraw = 2)

dd_LL_sr

Log-Likelihood for screening hmnl model

Description

Log-Likelihood for screening hmnl model

Usage

dd_LL_sr(draw, dd, fromdraw = 1)

Arguments

draw
A list, 'echoice2' draws object

dd
A tibble, tidy choice data (before dummy-coding)

fromdraw
An integer, from which draw onwards to compute LL (i.e., excl. burnin)
Value

N x Draws Matrix of log-Likelihood values

Examples

data(icecream_discrete)
#fit model
icecream_est <- icecream_discrete %>% dd_est_hmnl_screen(R=10, keep=1, cores=2)
#compute likelihood for each subject in each draw
loglls<-dd_LL_sr(icecream_est, icecream_discrete, fromdraw = 2)

---

dummify

*Create dummy variables within a tibble*

Description

Create dummy variables within a tibble

Usage

dummify(dat, sel)

Arguments

- **dat** A tibble with the data.
- **sel** A character vector with the name(s) of the variables to be dummied.

Value

tibble with dummy variables

Examples

mytest=data.frame(A=factor(c('a','a','b','c','c')), B=1:5)
dummify(mytest,"A")
**dummyvar**

**Dummy-code a categorical variable**

**Description**

Dummy-code a categorical variable

**Usage**

dummyvar(data)

**Arguments**

data

One column of categorical data to be dummy-coded

**Value**

tibble with dummy variables

**Examples**

mytest=data.frame(attribute=factor(c('a','a','b','c','c')))
dummyvar(mytest)

**ec_boxplot_MU**

*Generate MU_theta boxplot*

**Description**

Generate MU_theta boxplot

**Usage**

ee_c_boxplot_MU(draws, burnin = 100)

**Arguments**

draws

A list, `echoice2` draws object

burnin

Burn-in to remove

**Value**

A ggplot2 plot containing traceplots of draws

**See Also**

e_c_trace_MU() to obtain traceplot
Generating Screening Probability Boxplot

### Description
Generate Screening Probability boxplot.

### Usage
```r
ec_boxplot_screen(draws, burnin = 100)
```

### Arguments
- **draws**: A list, `echoice2` draws object, from a model with attribute-based screening.
- **burnin**: Burn-in to remove.

### Value
A ggplot2 plot containing traceplots of draws.

### See Also
- `ec_draws_MU()` to obtain MU_theta draws,
- `ec_trace_screen()` to generate traceplot.

### Examples
```r
data(icecream)
#run MCMC sampler (use way more than 20 draws for actual use
icecream_scr_est <- icecream %>% dplyr::filter(id<20) %>% vd_est_vdm_screen(R=20, cores=2)
ec_boxplot_screen(icecream_scr_est, burnin = 1)
```
**ec_demcurve**

### Description

This helper function creates demand curves.

### Usage

```r
ec_demcurve(
  ec_long,
  focal_product,
  rel_pricerange,
  dem_fun,
  draws,
  epsilon_not = NULL
)
```

### Arguments

- **ec_long**: choice scenario (discrete or volumetric)
- **focal_product**: Logical vector picking the focal product for which to create a demand curve
- **rel_pricerange**: Price range, relative to base case price; this is used to create demand curve
- **dem_fun**: demand function (e.g., `dd_prob` for HMNL or `vd_dem_vdm` for volumetric demand). For discrete choice, use choice probabilities instead of choice predictions.
- **draws**: ec-draws object (e.g., output from `dd_est_hmnl` or `vd_est_vd`)
- **epsilon_not**: (optional) error realizations (this helps make curves look smoother for volumetric models)

### Value

List containing aggregate demand quantities for each scenario defined by `rel_pricerange`.

### See Also

- `ec_gen_err_normal()` to generate error realization from Normal distribution,
- `ec_gen_err_ev1()` to generate error realization from EV1 distribution

### Examples

```r
data(icecream)
# Run MCMC sampler (use way more than 50 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<100) %>%
  vd_est_vdm(R=20, keep=1, cores=2)
# Demand at different price points
```
dem_scenarios <- ec_demcurve(icecream %>% dplyr::filter(id<100),
  icecream %>% dplyr::filter(id<100) %>% pull('Brand')=='Store',
  c(.75,1,1.25), vd_dem_vdm, icecream_est)
#optional plot
# dem_scenarios %>%
# do.call('rbind',. ) %>%
# ggplot(aes(x=scenario,y=\'E(demand)\',color=Flavor)) + geom_line()

ec_demcurve_cond_dem  Create demand-incidence curves

Description
This helper function creates demand curves

Usage
ec_demcurve_cond_dem(
  ec_long,
  focal_product,
  rel_pricerange,
  dem_fun,
  draws,
  epsilon_not = NULL
)

Arguments
- **ec_long**  choice scenario (discrete or volumetric)
- **focal_product**  Logical vector picking the focal product for which to create a demand curve
- **rel_pricerange**  Price range, relative to base case price; this is used to create demand curve
- **dem_fun**  demand function (e.g., dd_prob for HMNL or vd_dem_vdm for volumetric demand). For discrete choice, use choice probabilities instead of choice predictions.
- **draws**  ec-draws object (e.g., output from dd_est_hmnl or vd_est_vd)
- **epsilon_not** (optional) error realisations (this helps make curves look smoother for volumetric models)

Value
List containing aggregate demand quantities for each scenario defined by rel_pricerange
**ec_demcurve_inci**

Create demand-incidence curves

**Description**

This helper function creates demand curves

**Usage**

```r
ec_demcurve_inci(
  ec_long,                      # choice scenario (discrete or volumetric)
  focal_product,                # Logical vector picking the focal product for which to create a demand curve
  rel_pricerange,              # Price range, relative to base case price; this is used to create demand curve
  dem_fun,                     # demand function (e.g., dd_prob for HMNL or vd_dem_vdm for volumetric demand). For discrete choice, use choice probabilities instead of choice predictions.
  draws,                       # ec-draws object (e.g., output from dd_est_hmnl or vd_est_vd)
  epsilon_not = NULL           # (optional) error realisatins (this helps make curves look smoother for voumetric models)
)
```

**Arguments**

- **ec_long**: choice scenario (discrete or volumetric)
- **focal_product**: Logical vector picking the focal product for which to create a demand curve
- **rel_pricerange**: Price range, relative to base case price; this is used to create demand curve
- **dem_fun**: demand function (e.g., dd_prob for HMNL or vd_dem_vdm for volumetric demand). For discrete choice, use choice probabilities instead of choice predictions.
- **draws**: ec-draws object (e.g., output from dd_est_hmnl or vd_est_vd)
- **epsilon_not**: (optional) error realisations (this helps make curves look smoother for volumetric models)

**Examples**

```r
data(icecream)
#run MCMC sampler (use way more draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<20) %>%
  vd_est_vdm(R=2, keep=1, cores=2)
#demand at different price points
conddem_scenarios<-
  ec_demcurve_cond_dem(icecream%>% dplyr::filter(id<20),
                     icecream%>% dplyr::filter(id<20) %>% pull('Brand')=='Store',
                     c(.75,1),vd_dem_vdm,icecream_est)
```

**See Also**

- `ec_gen_err_normal()` to generate error realization from Normal distribution, `ec_gen_err_ev1()` to generate error realization from EV1 distribution
**ec_dem_aggregate**

**Value**

List containing aggregate demand quantities for each scenario defined by `rel_pricerange`

**See Also**

`ec_gen_err_normal()` to generate error realization from Normal distribution, `ec_gen_err_ev1()` to generate error realization from EV1 distribution

**Examples**

```r
data(icecream)
#run MCMC sampler (use way more than 50 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<50) %>%
vd_est_vdm(R=20, keep=1, cores=2)
#demand at different price points
inci_scenarios<-
ec_demcurve_inci(icecream%>% dplyr::filter(id<50),
    icecream%>% dplyr::filter(id<50) %>% pull(`Var`)=="Store",
    c(.75,1,1.25),vd_dem_vdm,icecream_est)
```

---

**Description**

Aggregate demand draws, e.g. from individual-choice occasion-alternative level to individual level. (using the new demand draw format)

**Usage**

```r
ec_dem_aggregate(de, groupby)
```

**Arguments**

- `de` demand draws
- `groupby` groupby grouping variables (as (vector of) string(s))

**Value**

Aggregated demand predictions
Examples

data(icecream)
#run MCMC sampler (use way more than 50 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<10) %>% vd_est_vdm(R=4, keep=1, cores=2)
#Generate demand predictions
icecream_predicted_demand =
  icecream %>% dplyr::filter(id<10) %>%
  vd_dem_vdm(icecream_est)
#aggregate
brand_lvl_pred_demand <-
  icecream_predicted_demand %>% ec_dem_aggregate("Brand")

ec_dem_eval

Evaluate (hold-out) demand predictions

Description

This function obtains proper posterior fit statistics. It computes the difference between true demand and each draw from the demand posterior. Then, fit statistics are obtained.

Usage

ec_dem_eval(de)

Arguments

de  demand draws (output from vd_dem_x function)

Value

Predictive fit statistics (MAE, MSE, RAE, bias, hit-probability)
**ec_dem_summarise**  
*Summarize posterior draws of demand*

**Description**

Add summaries of posterior draws of demand to tibble. (using the new demand draw format)

**Usage**

```r
ec_dem_summarise(de, quantiles)
```

```r
ec_dem_summarize(de, quantiles = c(0.05, 0.95))
```

**Arguments**

- `de` demand draws
- `quantiles` Quantiles for Credibility Intervals (default: 90% interval)

**Value**

Summary of demand predictions

**Examples**

```r
data(icecream)
# run MCMC sampler (use way more than 10 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<10) %>% vd_est_vdm(R=10, keep=1, cores=2)
# Generate demand predictions
icecream_predicted_demand <- icecream %>% dplyr::filter(id<10) %>%
                          vd_dem_vdm(icecream_est)
# aggregate
brand_LVL_pred_demand <- icecream_predicted_demand %>% ec_dem_aggregate("Brand")
# summarise
brand_LVL_pred_demand %>% ec_dem_summarise()
```
ec_draws_MU

Obtain MU_theta draws

Description
Obtain MU_theta draws

Usage
ec_draws_MU(draws)

Arguments
draws A list, 'echoice2' draws object

Value
A tibble, long format, draws of MU

See Also
ec_draws_screen() to obtain screening parameter draws (where applicable), ec_trace_MU() to generate a traceplot of MU_theta draws

Examples
data(icecream)
# run MCMC sampler (use way more than 20 draws for actual use
icecream_est <- icecream %>% dplyr::filter(id<50) %>% vd_est_vdm(R=20, cores=2)
ec_draws_MU(icecream_est)

ec_draws_screen

Obtain Screening probability draws

Description
Obtain Screening probability draws

Usage
ec_draws_screen(draws)

Arguments
draws A list, 'echoice2' draws object
ec_estimates_MU

Obtain upper level model estimates

Description

Obtain upper level model estimates

Usage

ec_estimates_MU(est, quantiles = c(0.05, 0.95))

Arguments

est

is an 'choice2' draw object (list)

quantiles

quantile for CI

Value

tibble with MU (upper level) summaries

Examples

data(icecream)
#run MCMC sampler (use way more than 20 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<20) %>% vd_est_vdm(R=20, cores=2)
#Upper-level summary
icecream_est %>% ec_estimates_MU
**ec_estimates_screen**  
*Summarize attribute-based screening parameters*

**Description**
Summarize attribute-based screening parameters from an attribute-based screening model in 'echoice2'.

**Usage**
```
ec_estimates_screen(est, quantiles = c(0.05, 0.95))
```

**Arguments**
- `est` is an 'echoice2' draw object (list) from a model with attribute-based screening
- `quantiles` quantile for CI

**Value**
tibble with screening summaries

**Examples**
```r
#run MCMC sampler (use way more than 20 draws for actual use)
data(icecream)
est_scr_icecream <- vd_est_vdm_screen(icecream%>%dplyr::filter(id<30), R=20, cores=2)
#summarise draws of screening probabilities
ec_estimates_screen(est_scr_icecream)
#Note: There is no variance in this illustrative example - more draws are needed
```

---

**ec_estimates_SIGMA**  
*Obtain posterior mean estimates of upper level covariance*

**Description**
Obtain posterior mean estimates of upper level covariance

**Usage**
```
ec_estimates_SIGMA(est)
```

**Arguments**
- `est` is an 'echoice2' draw object (list)
Value

estimates of upper level covariance

Examples

data(icecream)
#run MCMC sampler (use way more than 20 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<50) %>% vd_est_vdm(R=20, cores=2)
icecream_est %>% ec_estimates_SIGMA %>% round(2)

ec_estimates_SIGMA_corr

Obtain posterior mean estimates of upper level correlations

Description

Obtain posterior mean estimates of upper level correlations

Usage

ec_estimates_SIGMA_corr(est)

Arguments

est is an 'echoice2' draw object (list)

Value

estimates of upper level correlations

Examples

data(icecream)
#run MCMC sampler (use way more than 20 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<20) %>% vd_est_vdm(R=20, cores=2)
icecream_est %>% ec_estimates_SIGMA_corr %>% round(2)
**ec_gen_err_ev1**  
*Simulate error realization from EV1 distribution*

**Description**

Simulate error realization from EV1 distribution

**Usage**

```r
ec_gen_err_ev1(ec_dem, draws, seed = NULL)
```

**Arguments**

- `ec_dem`: discrete or volumetric choice data, with or without x  
- `draws`: draws from volumetric demand model  
- `seed`: seed for reproducible error realisations; set is automatically reset of running this function

**Value**

error realizations

**Examples**

```r
data(icecream)
#run MCMC sampler (use way more than 50 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<100) %>%
vd_est_vdm(R=100, keep=1, cores=2)
#generate error realizations
errs<- ec_gen_err_ev1(icecream %>% dplyr::filter(id<100), icecream_est, seed=123)
```

---

**ec_gen_err_normal**  
*Simulate error realization from Normal distribution*

**Description**

Simulate error realization from Normal distribution

**Usage**

```r
ec_gen_err_normal(ec_dem, draws, seed = NULL)
```
Arguments

- **ec_dem**: discrete or volumetric choice data, with or without x
- **draws**: draws from volumetric demand model
- **seed**: seed for reproducible error realisations; set is automatically reset of running this function

Value

error realizations

Examples

data(icecream)
#run MCMC sampler (use way more than 50 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<10) %>%
vd_est_vdm(R=10, keep=1, error_dist = "Normal", cores=2)
#generate error realizations
errs<- ec_gen_err_normal(icecream %>% dplyr::filter(id<10), icecream_est, seed=123)

Description

This is a helper function to quickly obtain log marginal density from a draw object

Usage

ec_lmd_NR(est)

Arguments

- **est**: 'echoice2' draw object

Details

Draws are split in 4 equal parts from start to finish, and LMD is computed for each part. This helps to double-check convergence.

Value

tibble with LMDs (first 25% of draws, next 25% of draws, ...)

Obtain Log Marginal Density from draw objects

ec_lmd_NR
Examples

```r
data(icecream)
#run MCMC sampler (use way more than 50 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<100) %>% vd_est_vdm(R=20, cores=2)
#obtain LMD by quartile of draws
ec_lmd_NR(icecream_est)
```

ec_lol_tidy1

Convert "list of lists" format to long "tidy" format

Description

Convert "list of lists" format to long "tidy" format

Usage

```r
ec_lol_tidy1(data_lol, X = "X", y = "y")
```

Arguments

- `data_lol`: A list of data frames containing design matrices and response vectors
- `X`: The column name of the design matrix, default: "X"
- `y`: The column name of the response vector, default: "y"

Value

A tidy data frame with columns for each design matrix column, the response vector, and an id column indicating which data frame the row came from

Examples

```r
loldata=list()
loldata[[1]]=list()
loldata[[1]]$y = c(1,2)
loldata[[1]]$X = data.frame(brand1=c(1,0, 1,0),brand2=c(0,1, 0,1),price=c(1,2))
loldata[[2]]=list()
loldata[[2]]$y = c(1,1)
loldata[[2]]$X = data.frame(brand1=c(1,0, 1,0),brand2=c(0,1, 0,1),price=c(1,2))
ec_lol_tidy1(loldata)
```
### ec_screenprob_sr  
**Screening probabilities of choice alternatives**

**Description**

Obtain draws of screening probabilities of choice alternatives

**Usage**

```r
ec_screenprob_sr(xd, est, cores=NULL)
```

**Arguments**

- **xd**: data
- **est**: ec-model draws
- **cores**: (optional) cores

**Value**

Draws of screening probabilities of choice alternatives

**Examples**

```r
data(icecream)
icecream_est <- icecream %>% filter(id<10) %>% vd_est_vdm_screen(R=10, price_screen=TRUE, cores=2)
ec_screenprob_sr(icecream %>% filter(id<10), icecream_est, cores=2)
```

### ec_screen_summarise  
**Summarize posterior draws of screening**

**Description**

Adds summaries of posterior draws of demand to tibble. (using the new demand draw format)

**Usage**

```r
ec_screen_summarise(sc, quantiles = c(0.05, 0.95))
```

**Arguments**

- **sc**: tibble containing screening draws in .screendraws
- **quantiles**: Quanitiles for Credibility Intervals (default: 90% interval)
**ec_summarize_attrlvls**

**Value**

Summary of screening probabilities

**Examples**

```r
data(icecream)

icecream_est <- icecream %>% vd_est_vdm_screen(R=20, price_screen=TRUE, cores=2)

#consideration set by respondent

cons_ss <-
  ec_screenprob_sr(icecream, icecream_est, cores=2) %>%
  group_by(id, task) %>%
  summarise(.screendraws=list(purrr::reduce(.screendraws ,'+'))) %>%
  ec_screen_summarise() %>%
  group_by(id) %>%
  summarise(n_screen=mean('E(screening)'))
```

---

**ec_summarize_attrlvls  Summarize attributes and levels**

**Description**

Summarize attributes and levels in tidy choice data containing categorical attributes (before dummy-coding)

**Usage**

```r
ec_summarize_attrlvls(data_in)
```

**Arguments**

- `data_in`  A tibble, containing long-format choice data

**Details**

This function looks for categorical attributes and summarizes their levels. This is helpful when evaluating a new choice data file.

**Value**

A tibble with one row per attribute, and a list of the levels

**Examples**

```r
data(icecream)

ec_summarize_attrlvls(icecream)
```
ec_trace_MU

Generate MU_theta traceplot

Description
Generate MU_theta traceplot

Usage
ec_trace_MU(draws, burnin = 100)

Arguments
- draws: A list, 'echoice2' draws object
- burnin: burn-in to remove

Value
A ggplot2 plot containing traceplots of draws

See Also
ec_boxplot_MU() to obtain boxplot

Examples
## Not run:
data(icecream)
#run MCMC sampler (use way more than 20 draws for actual use
icecream_est <- icecream %>% dplyr::filter(id<10) %>% vd_est_vdm(R=10, cores=2)
ec_trace_MU(icecream_est)
## End(Not run)

ec_trace_screen

Generate Screening probability traceplots

Description
Generate Screening probability traceplots

Usage
ec_trace_screen(draws, burnin = 100)
**Arguments**

- **draws**
  A list, 'echoice2' draws object, from a model with attribute-based screening

- **burnin**
  burn-in to remove

**Value**

A ggplot2 plot containing traceplots of draws

**See Also**

- `ec_draws_MU()` to obtain MU.theta draws, `ec_boxplot_screen()` to generate boxplot

**Examples**

```r
## Not run:
data(icecream)
#run MCMC sampler (use way more than 20 draws for actual use
icecream_scr_est <- icecream %>% dplyr::filter(id<20) %>% vd_est_vdm_screen(R=20, cores=2)
ec_trace_screen(icecream_scr_est, burnin=1)
## End(Not run)
```

---

**ec_undummy**

Converts a set of dummy variables into a single categorical variable

**Description**

Given a set of dummy variables, this function converts them into a single categorical variable. The categorical variable is created by determining which variables are active (i.e. have a value of 1) for each observation and assigning a category based on the set of active variables. If necessary, a reference level can be specified to ensure that all possible categories are represented. Often, all brands of a brand attribute are added as brand intercepts, while other categorical attributes are coded with respect to a reference level.

**Usage**

`ec_undummy(data_in, set_members, attribute_name, ref_level = NULL)`

**Arguments**

- **data_in** a data frame containing the dummy variables
- **set_members** a character vector of the names of the dummy variables
- **attribute_name** a character string representing the name of the new categorical variable
- **ref_level** a character string representing the name of the reference level. If specified, a new dummy variable will be created for this level, and it will be used as the reference category for the categorical variable. Defaults to NULL.
Value

A data frame with the same columns as `data_in`, except for the dummy variables in `set_members`, which are replaced with the new categorical variable `attribute_name`.

Examples

```r
minidata = structure(list(id = c("1", "1", "1", "1", "2", "2", "2", "2"),
               task = c(1L, 1L, 2L, 2L, 3L, 3L, 4L, 4L),
               alt = c(1L, 2L, 1L, 2L, 1L, 2L, 1L, 2L),
               brand1 = c(1L, 0, 1, 0, 1, 0, 1, 0),
               brand2 = c(0, 1, 0, 1, 0, 1, 0, 1),
               price = c(1, 2, 1, 2, 1, 2, 1, 2),
               x = c(1, 0, 1, 1, 0, 1, 0, 1)),
               class = c("tbl_df", "tbl", "data.frame"), row.names = c(NA, -8L))

minidata %>% ec_undummy(c('brand1','brand2'),'brand')
```

---

**ec_undummy_lowhigh**  
*Convert dummy-coded variables to low/high factor*

**Description**

Convert dummy-coded variables to low/high factor.

**Usage**

```r
ec_undummy_lowhigh(vec_in)
```

**Arguments**

- `vec_in`: A vector of dummy-coded variables (0/1)

**Value**

A factor vector with levels "low" and "high"

**Examples**

```r
ec_undummy_lowhigh(c(0,1,0,1,1))
```
**ec_undummy_lowmediumhigh**

*Convert dummy-coded variables to low/medium/high factor*

**Description**
Convert dummy-coded variables to low/medium/high factor

**Usage**
```
ec_undummy_lowmediumhigh(vec_in)
```

**Arguments**
- `vec_in` A vector of dummy-coded variables (0/1/2)

**Value**
A factor vector with levels "low", "medium" and "high"

**Examples**
```
ec_undummy_lowmediumhigh(c(0,1,2,1,0,2))
```

---

**ec_undummy_yesno**

*Convert dummy-coded variables to yes/no factor*

**Description**
Convert dummy-coded variables to yes/no factor

**Usage**
```
ec_undummy_yesno(vec_in)
```

**Arguments**
- `vec_in` A vector of dummy-coded variables (0/1)

**Value**
A factor vector with levels "no" and "yes"
**ec_util_choice_to_long**

Convert a vector of choices to long format

**Usage**

ec_util_choice_to_long(myvec, all_index)

**Arguments**

myvec A vector of choices, where each element represents the index of the chosen alternative.

all_index A vector of all the possible alternative indices.

**Value**

A tibble with columns 'x', 'task', and 'alt', where 'x' is a binary indicator of whether the alternative was chosen or not, 'task' is the task index, and 'alt' is the alternative index.

**Examples**

#There are 3 alternatives in this task.
#Since there are 3 observations in myvec, there are 3 tasks total.
ec_util_choice_to_long(c(1, 2, 1), c(1, 2, 3))
Find mutually exclusive columns

Description

This function finds pairs of columns in a data frame that are mutually exclusive, i.e., that never have positive values at the same time.

Usage

ec_util_dummy_mutualeclusive(data_in, filtered = TRUE)

Arguments

data_in: A data frame containing the data.
filtered: A logical value indicating whether to return only the mutually exclusive pairs (TRUE) or all pairs (FALSE). Default is TRUE.

Value

A tibble containing all pairs of mutually exclusive columns in the data frame.

Examples

minidata=structure(list(id = c("1", "1", "1", "1", "2", "2", "2", "2"),
  task = c(1L, 1L, 2L, 2L, 3L, 3L, 4L, 4L),
  alt = c(1L, 2L, 1L, 2L, 1L, 2L, 1L, 2L),
  brand1 = c(1, 0, 1, 0, 1, 0, 1, 0),
  brand2 = c(0, 1, 0, 1, 0, 1, 0, 1),
  price = c(1, 2, 1, 2, 1, 2, 1, 2),
  x = c(1, 0, 1, 1, 0, 1, 0),
  class = c("tbl_df", "tbl", "data.frame"),
  row.names = c(NA, -8L))
ec_util_dummy_mutualeclusive(minidata)

Obtain attributes and levels from tidy choice data with dummies

Description

Obtain attributes and levels from tidy choice data with dummies

Usage

get_attr_lvl(tdc)
Arguments

tdc  A tibble with choice data

Value

tibble

Examples

mytest=data.frame(A=factor(c('a', 'a', 'b', 'c', 'c')), B=1:5)
dummied_data = dummify(mytest,"A")
get_attr_lvl(dummied_data)

Description

Volumetric Conjoint data, ice cream category

Details

Data from volumetric conjoint analysis in the ice cream category. 300 respondents total. Volumetric demand in units of 4 ounces each. Attributes include brand name, flavor, and container size.

Description

Discrete-Choice Conjoint data, ice cream category

Details

Data from discrete choice conjoint analysis in the ice cream category. 300 respondents total. Attributes include brand name, flavor, and container size.
logMargDenNRu

Description
This function uses the quick-and-dirty Newton-Raftery approximation for log-marginal-density.

Usage
logMargDenNRu(ll)

Arguments
ll  A vector of log-likelihood values (i.e., draws)

Details
Approximation of LMD based on Newton-Raftery. It is not the most accurate, but a very fast method.

Value
A single numeric value representing the log marginal density

Examples
logll_values <- c(-4000, -4001, -4002)
logMargDenNRu(logll_values)

pizza

Description
Volumetric Conjoint data, pizza category

Details
Data from volumetric conjoint analysis in the frozen pizza category.
**prep_newprediction**  
*Match factor levels between two datasets*

**Description**

Makes sure the factor levels in `data_new` are aligned with `data_old` This is helpful for demand simulations.

**Usage**

```
prep_newprediction(data_new, data_old)
```

**Arguments**

- `data_new`  
  New long-format choice data  
- `data_old`  
  Old long-format choice data

**Value**

long-format choice data

**Examples**

```
data(icecream)
prep_newprediction(icecream, icecream)
```

---

**vd_add_prodid**  
*Add product id to demand draws*

**Description**

This adds a unique product identifier to demand draw objects.

**Usage**

```
vd_add_prodid(de)
```

**Arguments**

- `de`  
  demand draws

**Value**

`est`
Examples

data(icecream)
#run MCMC sampler (use way more than 10 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<10) %>% vd_est_vdm(R=4, keep=1, cores=2)
#Generate demand predictions
icecream_predicted_demand =
  icecream %>% dplyr::filter(id<10) %>%
    vd_dem_vdm(icecream_est)
#add prodid
icecream_predicted_demand_w_id <- icecream_predicted_demand %>% vd_add_prodid

vd_dem_summarise  Summarize posterior draws of demand (volumetric models only)

Description

   Adds summaries of posterior draws of demand to tibble. (using the new demand draw format)

Usage

   vd_dem_summarise(de, quantiles = c(0.05, 0.95))
   vd_dem_summarize(de, quantiles = c(0.05, 0.95))

Arguments

   de       demand draws
   quantiles Quantiles for Credibility Intervals (default: 90% interval)

Value

   Summary of demand predictions

Examples

data(icecream)
#run MCMC sampler (use way more than 10 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<10) %>% vd_est_vdm(R=10, keep=1, cores=2)
#Generate demand predictions
icecream_predicted_demand =
  icecream %>% dplyr::filter(id<10) %>%
    vd_dem_vdm(icecream_est)
#aggregate
brand_lvl_pred_demand <- icecream_predicted_demand %>% ec_dem_aggregate("Brand")
vd_dem_vdm

#summarise
brand_lvl_pred_demand %>% vd_dem_summarise()

vd_dem_vdm

Demand Prediction (Volumetric Demand Model)

Description

Generating demand predictions for volumetric demand model. Reminder: there is no closed-form solution for demand, thus we need to integrate not only over the posterior distribution of parameters and the error distribution. The function outputs a tibble containing id, task, alt, p, attributes, draws from the posterior of demand. Error realizations can be pre-supplied to the epsilon_not. This helps create smooth demand curves or conduct optimization.

Usage

vd_dem_vdm(
  vd,
  est,
  epsilon_not = NULL,
  error_dist = NULL,
  tidy = TRUE,
  cores = NULL
)

Arguments

vd  data
est ec-model draws
epsilon_not  (optional) error realizations
error_dist  (optional) A string defining the error term distribution (default: 'EV1')
tidy  (optional) apply 'echoice2' tidier (default: TRUE)
cores  (optional) cores (default: auto-detect)

Value

Draws of expected demand

See Also

prep_newprediction() to match vd’s factor levels, ec_gen_err_ev1() for pre-generating error realizations and vd_est_vdm() for estimating the corresponding model
Examples

```r
data(icecream)
#run MCMC sampler (use way more than 10 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<20) %>% vd_est_vdm(R=10, keep=1, cores=2)
#Generate demand predictions
icecream_predicted_demand = icecream %>% dplyr::filter(id<20) %>%
  vd_dem_vdm(icecream_est, cores=2)
#column .demdraws contains draws from posterior of predicted demand
```

Description

Generating demand predictions for volumetric demand model with attribute-based screening. Reminder: there is no closed-form solution for demand, thus we need to integrate not only over the posterior distribution of parameters and the error distribution. The function outputs a tibble containing id, task, alt, p, attributes, draws from the posterior of demand. Eerror realisations can be pre-supplied to the epsilon_not. This helps create smooth demand curves or conduct optimization.

Usage

```r
vd_dem_vdm_screen(vd, est, epsilon_not = NULL, error_dist = NULL, cores = NULL)
```

Arguments

- `vd`  
  data
- `est`  
  ec-model draws
- `epsilon_not`  
  (optional) error realizations
- `error_dist`  
  (optional) A string defining the error term distribution (default: 'EV1')
- `cores`  
  (optional) cores

Value

Draws of expected demand

See Also

`prep_newprediction()` to match `vd`'s factor levels, `ec_gen_err_normal()` for pre-generating error realizations and `vd_est_vdm_screen()` for estimating the corresponding model
Examples

```r
data(icecream)
#run MCMC sampler (use way more than 20 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<20) %>% vd_est_vdm_screen(R=20, keep=1, cores=2)
#Generate demand predictions
icecream_predicted_demand=
  icecream %>% dplyr::filter(id<20) %>%
  vd_dem_vdm_screen(icecream_est, cores=2)
#column .demdraws contains draws from posterior of predicted demand
```

---

vd_dem_vdm_ss

**Demand Prediction (Volumetric demand, accounting for set-size variation, EV1 errors)**

Description

Generating demand predictions for volumetric demand model with set-size adjustment. Reminder: there is no closed-form solution for demand, thus we need to integrate not only over the posterior distribution of parameters and the error distribution. The function outputs a tibble containing id, task, alt, p, attributes, draws from the posterior of demand. Error realizations can be pre-supplied to the epsilon_not. This helps create smooth demand curves or conduct optimization.

Usage

```r
vd_dem_vdm_ss(vd, est, epsilon_not = NULL, cores = NULL)
```

Arguments

- **vd**: data
- **est**: ec-model draws
- **epsilon_not**: (optional) error realizations
- **cores**: (optional) cores

Value

Draws of expected demand

See Also

`prep_newprediction()` to match vd’s factor levels, `ec_gen_err_ev1()` for pre-generating error realizations and `vd_est_vdm_ss()` for estimating the corresponding model
vd_est_vdm

Examples

data(icecream)
#run MCMC sampler (use way more than 10 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<10) %>% vd_est_vdm_ss(R=10, keep=1, cores=2)
#Generate demand predictions
icecream_predicted_demand=
  icecream %>% dplyr::filter(id<10) %>%
  vd_dem_vdm_ss(icecream_est, cores=2)
#column .demdraws contains draws from posterior of predicted demand

vd_est_vdm

Estimate volumetric demand model

Description

Estimate volumetric demand model

Usage

vd_est_vdm(
  vd,
  tidy = TRUE,
  R = 1e+05,
  keep = 10,
  cores = NULL,
  error_dist = "EV1",
  control = list(include_data = TRUE)
)

Arguments

vd A tibble, containing volumetric demand data (long format)
tidy A logical, whether to apply 'echoice2' tidier function (default: TRUE)
R A numeric, no of draws
keep A numeric, thinning factor
cores An integer, no of CPU cores to use (default: auto-detect)
error_dist A string defining the error term distribution, 'EV1' or 'Normal'
control A list containing additional settings

Value

An 'echoice2' draw object, in the form of a list
See Also

- `vd_dem_vdm()` to generate demand predictions based on this model
- `vd_est_vdm_screen()` to estimate a volumetric demand model with screening

Examples

```r
data(icecream)
#run MCMC sampler (use way more than 10 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<50) %>% vd_est_vdm(R=10, cores=2)
```

---

vd_est_vdm_screen  Estimate volumetric demand model with attribute-based conjunctive screening

Description

Estimate volumetric demand model with attribute-based conjunctive screening

Usage

```r
vd_est_vdm_screen(
  vd,  # volumetric demand data (long format)
  R = 1e+05,  # draws
  keep = 10,  # thinning
  cores = NULL,  # no of CPU cores to use (default: auto-detect)
  error_dist = "EV1",  # A string defining the error term distribution, ‘EV1’ or ‘Normal’ (default: ‘EV1’)
  price_screen = TRUE,  # A logical, indicating whether price tag screening should be estimated (default: TRUE)
  control = list(include_data = TRUE)  # list containing additional settings
)
```

Arguments

- **vd**: volumetric demand data (long format)
- **R**: draws
- **keep**: thinning
- **cores**: no of CPU cores to use (default: auto-detect)
- **error_dist**: A string defining the error term distribution, ‘EV1’ or ‘Normal’ (default: ‘EV1’)
- **price_screen**: A logical, indicating whether price tag screening should be estimated (default: TRUE)
- **control**: list containing additional settings

Value

- est ec-draw object (List)
Examples

```r
data(icecream)
icecream_est <- icecream %>% vd_est_vdm_screen(R=10, cores=2)
```

**vd_est_vdm_ss**

Estimate volumetric demand model accounting for set size variation (1st order)

Description

This model **REQUIRES** variation in choice-set size

Usage

```r
vd_est_vdm_ss(
    vd,
    order = 1,
    R = 1e+05,
    keep = 10,
    cores = NULL,
    control = list(include_data = TRUE)
)
```

Arguments

- **vd**: volumetric demand data (long format) with set size variation
- **order**: integer, either 1 or 2 (for now), indicating linear or quadratic set-size effect
- **R**: draws
- **keep**: thinning
- **cores**: no of CPU cores to use (default: auto-detect)
- **control**: list containing additional settings

Value

*est* ec-draw object (List)

Examples

```r
data(icecream)
#note that for this example dataset, the model is not identified
#because the data lacks variation in set size
icecream_est <- icecream %>% vd_est_vdm_ss(R=10, cores=2)
```
vd_LL_vdm  

Log-Likelihood for compensatory volumetric demand model

Description

Log-Likelihood for compensatory volumetric demand model

Usage

vd_LL_vdm(draw, vd, fromdraw = 1)

Arguments

draw A list, 'echoice2' draws object
vd A tibble, tidy choice data (before dummy-coding)
fromdraw An integer, from which draw onwards to compute LL (i.e., excl. burnin)

Value

N x Draws Matrix of log-Likelihood values

Examples

data(icecream)
#fit model
icecream_est <- icecream %>% vd_est_vdm(R=10, keep=1, cores=2)
#compute likelihood for each subject in each draw
loglls<-vd_LL_vdm(icecream_est, icecream, fromdraw = 2)
dim(loglls)

vd_LL_vdmss  

Log-Likelihood for volumetric demand model with set-size variation

Description

Log-Likelihood for volumetric demand model with set-size variation

Usage

vd_LL_vdmss(draw, vd, fromdraw = 1)

Arguments

draw A list, 'echoice2' draws object
vd A tibble, tidy choice data (before dummy-coding)
fromdraw An integer, from which draw onwards to compute LL (i.e., excl. burnin)
Value

N x Draws Matrix of log-Likelihood values

Examples

data(icecream)
# fit model
# note: this is just for demo purposes
# on this demo dataset, the model is not identified
due to a lack of set size variation
icecream_est <- icecream %>% vd_est_vdm_screen(R=10, keep=1, cores=2)
# compute likelihood for each subject in each draw
loglls<- vd_LL_vdm_screen(icecream_est, icecream, fromdraw = 2)
# 300 respondents, 10 draws
dim(loglls)

---

vd_LL_vdm_screen  
Log-Likelihood for conjunctive-screening volumetric demand model

Description

Log-Likelihood for conjunctive-screening volumetric demand model

Usage

vd_LL_vdm_screen(draw, vd, fromdraw = 1)

Arguments

draw  
A list, 'echoice2' draws object

vd  
A tibble, tidy choice data (before dummy-coding)

fromdraw  
An integer, from which draw onwards to compute LL (i.e., excl. burnin)

Value

N x Draws Matrix of log-Likelihood values

Examples

data(icecream)
# fit model
icecream_est <- icecream %>% filter(id<20) %>% vd_est_vdm_screen(R=10, keep=1, cores=2)
# compute likelihood for each subject in each draw
loglls<- vd_LL_vdm_screen(icecream_est, icecream%>%filter(id<20), fromdraw = 2)
dim(loglls)
vd_long_tidy  Generate tidy choice data with dummies from long-format choice data

Description
Generate tidy choice data with dummies from long-format choice data

Usage
vd_long_tidy(longdata)

Arguments
longdata  tibble

Value
tibble

Examples
data(icecream)
vd_long_tidy(icecream)

vd_prepare  Prepare choice data for analysis

Description
This utility function prepares tidy choice data for fast MCMC samplers.

Usage
vd_prepare(dt, Af = NULL)

Arguments
dt  tidy choice data (columns: id, task, alt, x, p, attributes)
Af  (optional) contains a full design matrix (for attribute-based screening), or, more generally, a design matrix used for attribute-based screening

Details
Note: This function is only exported because it makes it easier to tinker with this package. This function re-arranges choice data for fast access in highly-optimized MCMC samplers. It pre-computes task-wise total expenditures sumpsx and generates indices xfr, xto, lfr, lto for fast data access.
vd_prepare_nox

Value

list containing information for estimation functions

Examples

#minimal data example
dt <- structure(list(id = c(1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L,
                        2L, 2L),
task = c(1L, 1L, 1L, 2L, 2L, 2L, 1L, 1L, 1L, 2L, 2L, 2L),
alt = c(1L, 2L, 3L, 1L, 2L, 3L, 1L, 2L, 3L, 1L, 2L, 3L),
x = c(1, 0, 2, 1, 0, 1, 2, 3, 1, 1, 0, 1),
p = c(0, 1, 1, 1, 2, 0, 2, 2, 1, 2, 1, 1),
attr2 = c(1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0),
attr1 = c(0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1)),
class = c("tbl_df", "tbl", "data.frame"), row.names = c(NA,-12L))

#run prep function
test <- dt %>% vd_prepare

---

vd_prepare_nox  Prepare choice data for analysis (without x being present)

Description

This utility function prepares tidy choice data (without x) for fast data access.

Usage

vd_prepare_nox(dt, Af = NULL)

Arguments

dt  tidy choice data (columns: id, task, alt, p, attributes)
Af  (optional) contains a full design matrix (for attribute-based screening), or, more generally, a design matrix used for attribute-based screening

Details

Note: This function is only exported because it makes it easier to tinker with this package. This function re-arranges choice data for fast access, mainly for demand prediction.

Value

list containing information for prediction functions
Examples

# Minimal example:
# One attribute with 3 levels, 2 subjects, 3 alternatives, 2 tasks
dt <- structure(list(id = c(1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L,
                        2L, 2L),
  task = c(1L, 1L, 1L, 2L, 2L, 2L, 1L, 1L, 1L, 2L, 2L, 2L),
  alt = c(1L, 2L, 3L, 1L, 2L, 3L, 1L, 2L, 3L, 1L, 2L, 3L),
  x = c(1, 0, 2, 1, 0, 1, 2, 3, 1, 1, 0, 1),
  p = c(0, 1, 1, 2, 0, 2, 2, 1, 2, 1, 1),
  attr2 = c(1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0),
  attr1 = c(0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0)),
  class = c("tbl_df", "tbl", "data.frame"), row.names = c(NA,-12L))
test <- dt %>% dplyr::select(-all_of("x")) %>% vd_prepare_nox()

vd_thin_draw

Thin 'choice2'-vd draw objects

Description

Thin 'choice2'-vd draw objects

Usage

vd_thin_draw(est, burnin_perc = 0.5, total_draws = NULL)

Arguments

est is an 'choice2' draw object (list)
burnin_perc how much burn-in to remove
total_draws how many draws to keep after thinning

Value

thinned 'choice2' draw object (list)

Examples

data(icecream)
# run MCMC sampler (use way more than 50 draws for actual use)
icecream_est <- icecream %>% dplyr::filter(id<100) %>% vd_est_vdm(R=10, keep = 1, cores=2)
# without thinning, yields R=50 draws
dim(icecream_est$MUDraw)
icecream_est_thinned <- vd_thin_draw(icecream_est,.5)
# 26 draws left after thinning about half
dim(icecream_est_thinned$MUDraw)
Description
Get the attribute of an object

Usage
obj %.% attrname

Arguments
obj The object to get the attribute from.
attrname The name of the attribute to get.

Value
The attribute of the object.

Examples
obj <- list(a = 1, b = 2)
attributes(obj)$test="hello"
'%.%'(obj, "test")
Index

%, 47

dd_dem, 3
dd_dem(), 5
dd_dem_sr, 4
dd_dem_sr(), 6
dd_est_hmn1, 5
dd_est_hmn1(), 3
dd_est_hmn1_screen, 6
dd_est_hmn1_screen(), 4
dd_LL, 7
dd_LL_sr, 7
dummify, 8
dummyvar, 9
ec_boxplot_MU, 9
ec_boxplot_MU(), 26
ec_boxplot_screen, 10
ec_boxplot_screen(), 27
dc_dem_aggregate, 14
dc_dem_eval, 15
dc_dem_summarise, 16
dc_dem_summarise(ec_dem_summarise), 16
dc_demcurve, 11
dc_demcurve_cond_dem, 12
dc_demcurve_inci, 13
dc_draws_MU, 17
dc_draws_MU(), 10, 18, 27
dc_draws_screen, 17
dc_draws_screen(), 17
dc_estimates_MU, 18
dc_estimates_screen, 19
dc_estimates_SIGMA, 19
dc_estimates_SIGMA_corr, 20
dc_gen_err_evl, 21
dc_gen_err_evl(), 11, 13, 14, 36, 38
dc_gen_err_normal, 21
dc_gen_err_normal(), 11, 13, 14, 37
dc_lmd_NR, 22
dc_lol_tidyl, 23
dc_screen_summarise, 24
dc_screen_summarize
(ec_screen_summarise), 24
dc_screenprob_sr, 24
dc_summarise_attrlvls
(ec_summarise_attrlvls), 25
dc_summarize_attrlvls, 25
dc_trace_MU, 26
dc_trace_MU(), 9, 17
dc_trace_screen, 26
dc_trace_screen(), 10, 18
dc_undummy, 27
dc_undummy_lowhigh, 28
dc_undummy_lowmediumhigh, 29
dc_undummy_yeso, 29
dc_util_choice_to_long, 30
dc_util_dummy_mutualeclusive, 31
gtc_attr_lvl, 31
icecream, 32
icecream_discrete, 32
logMargDenNRu, 33
pizza, 33
prep_newprediction, 34
prep_newprediction(), 36–38
vd_add_prodid, 34
vd_dem_summarise, 35
vd_dem_summarize(vd_dem_summarise), 35
vd_dem_vdm, 36
vd_dem_vdm(), 40
vd_dem_vdm_screen, 37
vd_dem_vdm_ss, 38
vd_est_vdm, 39
vd_est_vdm(), 36
vd_est_vdm_screen, 40
vd_est_vdm_screen(), 37, 40
vd_est_vdm_ss, 41
INDEX

vd_est_vdm_ss(), 38
vd_LL_vdm, 42
vd_LL_vdm_screen, 43
vd_LL_vdms, 42
vd_long_tidy, 44
vd_prepare, 44
vd_prepare_nox, 45
vd_thin_draw, 46