Package: baker (via r-universe)

September 2, 2024

Type Package

Title ``Nested Partially Latent Class Models''

Version 1.0.3.9001

Date 2024-04-25

Description Provides functions to specify, fit and visualize nested partially-latent class models (Wu, Deloria-Knoll, Hammitt, and Zeger (2016) <doi:10.1111/rssc.12101>; Wu, Deloria-Knoll, and Zeger (2017) <doi:10.1093/biostatistics/kxw037>; Wu and Chen (2021) <doi:10.1002/sim.8804>) for inference of population disease etiology and individual diagnosis. In the motivating Pneumonia Etiology Research for Child Health (PERCH) study, because both quantities of interest sum to one hundred percent, the PERCH scientists frequently refer to them as population etiology pie and individual etiology pie, hence the name of the package.

Depends R(>= 4.3.0)

Imports rjags(>= 4-6), R2jags(>= 0.5), lubridate(>= 1.3), binom(>= 1.1), coda(>= 0.16), robCompositions(>= 2.0.3), ggplot2(>= 1.0), ggpubr(>= 0.4.0), gridExtra(>= 2.0), reshape2(>= 1.4), mgcv(>= 1.8-6), mvbutils(>= 2.7.4.1), shinyFiles(>= 0.6), shinydashboard(>= 0.5.1), stats, utils, abind

License MIT + file LICENSE

Language en-US

SystemRequirements JAGS (>= 4.3.2) (http://mcmc-jags.sourceforge.net)

Suggests spelling, knitr, testthat, rmarkdown, covr, knitcitations, sf

VignetteBuilder knitr

RoxygenNote 7.2.3

Encoding UTF-8

URL https://github.com/zhenkewu/baker, https://zhenkewu.com/baker/

BugReports https://github.com/zhenkewu/baker/issues

Roxygen list(markdown = TRUE)

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

RemoteUrl https://github.com/zhenkewu/baker

RemoteRef HEAD

RemoteSha ab64a489a728b4d151242b204fedf854a77e2b75

Contents

`add_meas_BrS_case_Nest_Slice` *add likelihood for a BrS measurement slice among cases (conditional dependence)*

Description

add likelihood for a BrS measurement slice among cases (conditional dependence)

Usage

```
add_meas_BrS_case_Nest_Slice(s, Mobs, cause_list, ppd = NULL)
```

Arguments

`s`	the slice
`Mobs`	See `data_nplcm` described in `nplcm()`
`cause_list`	the list of causes in `data_nplcm` described in `nplcm()`
`ppd`	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_NoNest_Slice_
add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_Slice_jags(), add_meas_BrS_case_NoNest_
add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_
add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(),
add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_S
add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNe
add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_NoNest_Slice_jags(),
add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_Slice_jags(), add_meas_BrS_case_NoNest_
add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_
add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(),
add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_S
add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNe
add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

`add_meas_BrS_case_Nest_Slice_jags` *add likelihood for a BrS measurement slice among cases (conditional dependence)*

Description

add likelihood for a BrS measurement slice among cases (conditional dependence)

Usage

```
add_meas_BrS_case_Nest_Slice_jags(s, Mobs, cause_list, ppd = NULL)
```

Arguments

`s`	the slice
`Mobs`	See `data_nplcm` described in `nplcm()`
`cause_list`	the list of causes in `data_nplcm` described in `nplcm()`
`ppd`	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is `plug`, the `.bug` code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(
add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_Slice_jags(), add_meas_BrS_case_NoNest_
add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_j
add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(),
add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_S
add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNe
add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(),
add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_Slice_jags(), add_meas_BrS_case_NoNest_
add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_j
add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(),
add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_S
add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNe
add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags

add likelihood component for a BrS measurement slice among cases

---

Description

regression model with no nested subclasses; discrete predictors

Usage

```
add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(
  s,
  Mobs,
  prior,
  cause_list,
  ppd = NULL
)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
prior	Prior specifications.
cause_list	the list of causes in data_nplcm described in nplcm()
ppd	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_ add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_ add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_case_NoNest_reg_Slice_jags

add likelihood component for a BrS measurement slice among cases

---

Description

regression model with no nested subclasses

Usage

```
add_meas_BrS_case_NoNest_reg_Slice_jags(s, Mobs, prior, cause_list, ppd = NULL)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
prior	Prior specifications.
cause_list	the list of causes in data_nplcm described in nplcm()
ppd	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_ add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_ add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

`add_meas_BrS_case_NoNest_Slice` *add a likelihood component for a BrS measurement slice among cases (conditional independence)*

Description

add a likelihood component for a BrS measurement slice among cases (conditional independence)

Usage

```
add_meas_BrS_case_NoNest_Slice(s, Mobs, cause_list, ppd = NULL)
```

Arguments

`s`	the slice
`Mobs`	See `data_nplcm` described in `nplcm()`
`cause_list`	the list of causes in `data_nplcm` described in `nplcm()`
`ppd`	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_reg_Slice_jags(), add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sli add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_reg_Slice_jags(), add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sli add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_case_NoNest_Slice_jags

add a likelihood component for a BrS measurement slice among cases (conditional independence)

Description

add a likelihood component for a BrS measurement slice among cases (conditional independence)

Usage

```
add_meas_BrS_case_NoNest_Slice_jags(s, Mobs, prior, cause_list, ppd = NULL)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
prior	Prior specifications.
cause_list	the list of causes in data_nplcm described in nplcm()
ppd	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_Slice_jags(), add_meas_BrS_case_NoNest_reg_Slice(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNest_reg_Slice(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_Slice_jags(), add_meas_BrS_case_NoNest_reg_Slice(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNest_reg_Slice(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_ctrl_Nest_Slice

add likelihood for a BrS measurement slice among controls (conditional independence)

Description

add likelihood for a BrS measurement slice among controls (conditional independence)

Usage

```
add_meas_BrS_ctrl_Nest_Slice(s, Mobs, cause_list, ppd = NULL)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
cause_list	the list of causes in data_nplcm described in nplcm()
ppd	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jag add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jag add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags

add a likelihood component for a BrS measurement slice among controls

---

Description

regression model without nested subclasses; discrete

Usage

```
add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(
  s,
  Mobs,
  cause_list,
  ppd = NULL
)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
cause_list	the list of causes in data_nplcm described in nplcm()
ppd	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_param_Nest_ add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNe add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_param_Nest_ add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNe add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_ctrl_NoNest_reg_Slice_jags

add a likelihood component for a BrS measurement slice among controls

---

Description

regression model without nested subclasses

Usage

```
add_meas_BrS_ctrl_NoNest_reg_Slice_jags(s, Mobs, cause_list, ppd = NULL)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
cause_list	the list of causes in data_nplcm described in nplcm()
ppd	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

`add_meas_BrS_ctrl_NoNest_Slice` *add a likelihood component for a BrS measurement slice among controls (conditional independence)*

Description

add a likelihood component for a BrS measurement slice among controls (conditional independence)

Usage

```
add_meas_BrS_ctrl_NoNest_Slice(s, Mobs, cause_list, ppd = NULL)
```

Arguments

`s`	the slice
`Mobs`	See `data_nplcm` described in `nplcm()`
`cause_list`	the list of causes in `data_nplcm` described in `nplcm()`
`ppd`	Default is NULL; Set to TRUE for enabling posterior predictive checking.

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jag add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_reg_discrete_predictor_Slice_jag add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_param_Nest_reg_Slice_jags

add parameters for a BrS measurement slice among cases and controls

---

Description

regression model with nested subclasses; called by insert_bugfile_chunk_reg_nest_meas

Usage

```
add_meas_BrS_param_Nest_reg_Slice_jags(
  s,
  Mobs,
  prior,
  cause_list,
  FPR_formula = NULL
)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
prior	Prior specifications.
cause_list	the list of causes in data_nplcm described in nplcm()
FPR_formula	False positive regression formula for slice s of BrS data. Check model_options$likelihood$FPR_formu

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_NoNest_Slic add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_NoNest_Slic add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_param_Nest_Slice

add parameters for a BrS measurement slice among cases and controls (conditional dependence)

Description

add parameters for a BrS measurement slice among cases and controls (conditional dependence)

Usage

```
add_meas_BrS_param_Nest_Slice(s, Mobs, cause_list)
```

Arguments

`s`	the slice
`Mobs`	See `data_nplcm` described in `nplcm()`
`cause_list`	the list of causes in `data_nplcm` described in `nplcm()`

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_No add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_No add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

`add_meas_BrS_param_Nest_Slice_jags` *add parameters for a BrS measurement slice among cases and controls (conditional dependence)*

Description

add parameters for a BrS measurement slice among cases and controls (conditional dependence)

Usage

```
add_meas_BrS_param_Nest_Slice_jags(s, Mobs, cause_list)
```

Arguments

`s`	the slice
`Mobs`	See `data_nplcm` described in `nplcm()`
`cause_list`	the list of causes in `data_nplcm` described in `nplcm()`

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_S add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNe add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Slice_jags(), add_meas_BrS_param_NoNest_S add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNe add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

`add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags`
add parameters for a BrS measurement slice among cases and controls

---

Description

regression model with no nested subclasses; discrete

Usage

```
add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(
  s,
  Mobs,
  prior,
  cause_list
)
```

Arguments

`s`	the slice
`Mobs`	See `data_nplcm` described in `nplcm()`
`prior`	Prior specifications.
`cause_list`	the list of causes in `data_nplcm` described in `nplcm()`

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_param_NoNest_reg_Slice_jags

add parameters for a BrS measurement slice among cases and controls

---

Description

regression model with no nested subclasses

Usage

```
add_meas_BrS_param_NoNest_reg_Slice_jags(
  s,
  Mobs,
  prior,
  cause_list,
  FPR_formula
)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
prior	Prior specifications.
cause_list	the list of causes in data_nplcm described in nplcm()
FPR_formula	False positive regression formula for slice s of BrS data. Check model_options$likelihood$FPR_formu

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

`add_meas_BrS_param_NoNest_Slice`

add parameters for a BrS measurement slice among cases and controls (conditional independence)

Description

add parameters for a BrS measurement slice among cases and controls (conditional independence)

Usage

```
add_meas_BrS_param_NoNest_Slice(s, Mobs, cause_list)
```

Arguments

`s`	the slice
`Mobs`	See `data_nplcm` described in `nplcm()`
`cause_list`	the list of causes in `data_nplcm` described in `nplcm()`

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

---

add_meas_BrS_param_NoNest_Slice_jags

add parameters for a BrS measurement slice among cases and controls (conditional independence)

Description

add parameters for a BrS measurement slice among cases and controls (conditional independence)

Usage

```
add_meas_BrS_param_NoNest_Slice_jags(s, Mobs, prior, cause_list)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
prior	Prior specifications.
cause_list	the list of causes in data_nplcm described in nplcm()

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_reg_Slice_jags(), add_meas_BrS_param_NoN add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case(), add_meas_SS_param()

`add_meas_BrS_subclass_Nest_Slice` *add subclass indicators for a BrS measurement slice among cases and controls (conditional independence)*

Description

add subclass indicators for a BrS measurement slice among cases and controls (conditional independence)

Usage

```
add_meas_BrS_subclass_Nest_Slice(s, Mobs, cause_list, ppd = NULL, reg = NULL)
```

Arguments

s	the slice
Mobs	See data_nplcm described in nplcm()
cause_list	the list of causes in data_nplcm described in nplcm()
ppd	Default is NULL; Set to TRUE for enabling posterior predictive checking.
reg	Default is NULL; set to TRUE if doing regression (double index of subclass weights: subject and subclass)

Value

a list of two elements: the first is plug, the .bug code; the second is parameters that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_SS_case(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_SS_case(), add_meas_SS_param()

`add_meas_SS_case` *add likelihood for a SS measurement slice among cases (conditional independence)*

Description

add likelihood for a SS measurement slice among cases (conditional independence)

Usage

```
add_meas_SS_case(nslice, Mobs, prior, cause_list)
```

Arguments

`nslice`	the total number of SS measurement slices
`Mobs`	see `data_nplcm` described in `nplcm()`
`prior`	see `model_options` described in `nplcm()`
`cause_list`	the list of causes in `model_options` described in `nplcm()`

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_param()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_param()

`add_meas_SS_param` *add parameters for a SS measurement slice among cases (conditional independence)*

Description

add parameters for a SS measurement slice among cases (conditional independence)

Usage

```
add_meas_SS_param(nslice, Mobs, prior, cause_list)
```

Arguments

`nslice`	the total number of SS measurement slices
`Mobs`	see `data_nplcm` described in `nplcm()`
`prior`	see `model_options` described in `nplcm()`
`cause_list`	the list of causes in `model_options` described in `nplcm()`

Value

a list of two elements: the first is `plug`, the .bug code; the second is `parameters` that stores model parameters introduced by this plugged measurement slice

See Also

Other likelihood specification functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case()

Other plug-and-play functions: add_meas_BrS_case_Nest_Slice_jags(), add_meas_BrS_case_Nest_Slice(), add_meas_BrS_case_NoNest_Slice_jags(), add_meas_BrS_case_NoNest_Slice(), add_meas_BrS_case_NoNest_reg_ add_meas_BrS_case_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_ctrl_Nest_Slice(), add_meas_BrS_ctrl_NoNest_Slice(), add_meas_BrS_ctrl_NoNest_reg_Slice_jags(), add_meas_BrS_ctrl_NoNest_ add_meas_BrS_param_Nest_Slice_jags(), add_meas_BrS_param_Nest_Slice(), add_meas_BrS_param_Nest_reg_Sl add_meas_BrS_param_NoNest_Slice_jags(), add_meas_BrS_param_NoNest_Slice(), add_meas_BrS_param_NoNest_ add_meas_BrS_param_NoNest_reg_discrete_predictor_Slice_jags(), add_meas_BrS_subclass_Nest_Slice(), add_meas_SS_case()

---

as.matrix_or_vec *convert one column data frame to a vector*

Description

convert one column data frame to a vector

Usage

```
as.matrix_or_vec(x)
```

Arguments

x	an one-column data.frame

Details

JAGS cannot accept a data frame with one column; This function converts it to a vector, which JAGS will allow.

Value

a vector

`assign_model` *Interpret the specified model structure*

Description

`assign_model` translates options specified by a user (e.g., in `model_options`) into information that can be understood by `baker`.

Usage

```
assign_model(model_options, data_nplcm, silent = TRUE)
```

Arguments

`model_options`	See `nplcm()` function.
`data_nplcm`	Data. See `nplcm()` function for data structure.
`silent`	Default is `TRUE` for no messages; `FALSE` otherwise.

Details

`assign_model` will be modified to check if data are conformable to specified model.

Value

A list of model specifications:

- `num_slice` A vector counting the No. of measurement slices for each level of measurement quality (e.g., MBS, MSS, MGS representing Bronze-Standard Measurements - case-control, Silver-Standard Measurements and Gold-Standard Measurements - case-only);
- `nested` Local dependence specification for modeling bronze-standard data. `TRUE` for nested models (conditional dependence given disease class); `FALSE` for non-nested models (conditional independence given disease class). One for each BrS slice.
- `regression`
  - `do_reg_Eti` `TRUE` for doing etiology regression. It means let the etiology fractions vary with explanatory variables. `FALSE` otherwise;
  - `do_reg_FPR` A vector whose names represent the slices of bronze-standard data. For each slice of BrS measurements, `TRUE` does false positive rate regression. It means the false positive rates, estimatable from controls, can vary with covariates; `FALSE` otherwise.
  - `is_discrete_predictor` A list of names "Eti", and the names for every slice of bronze-standard data. `TRUE` if all predictors are discrete; `FALSE` otherwise.

Examples

```
cause_list <- c(LETTERS[1:6])
J.BrS <- 6
model_options_no_reg <- list(
likelihood   = list(
  cause_list = cause_list,
  k_subclass = 2,
  Eti_formula = ~-1,
  # no covariate for the etiology regression
  FPR_formula = list(
    MBS1 =   ~-1)
    # no covariate for the subclass weight regression
),
use_measurements = c("BrS"),
# use bronze-standard data only for model estimation.
prior= list(
  Eti_prior = overall_uniform(1,cause_list),
  # Dirichlet(1,...,1) prior for the etiology.
  TPR_prior  = list(BrS = list(
    info  = "informative", # informative prior for TPRs
    input = "match_range",
    # specify the informative prior for TPRs by specifying a plausible range.
    val = list(MBS1 = list(up =  list(rep(0.99,J.BrS)),
    # upper ranges: matched to 97.5% quantile of a Beta prior
                           low = list(rep(0.55,J.BrS))))
                           # lower ranges: matched to 2.5% quantile of a Beta prior
  )
  )
)
)
data("data_nplcm_noreg")

assign_model(model_options_no_reg,data_nplcm_noreg)
```

---

baker *baker:* **B***ayesian* **A***nalytic* **K***it for* **E***tiology* **R***esearch*

Description

baker is designed for disease etiology studies from case-control data with multiple sources of measurements with potential errors. If you are interested in estimating the population etiology pie (a vector of fractions that sum to one), and the probability of each cause for a particular individual case, try baker.

Details

baker implements hierarchical Bayesian models to infer disease etiology for multivariate binary data. We created baker to catalyze effective communications between analysts and practicing clinicians that are vital to the success of etiology studies. The baker package offers modules to

- Import and tidy the PERCH data (the study that motivates the creation of this package),
- Transform, explore the data,
- Specify, automatically generate the model files, and fit the models (npLCM),
- Store and visualize posterior summaries for communicating scientific findings, and
- Check and compare the fitted models.

baker has implemented models for dependent measurements given disease status, regression analyses of etiology, multiple imperfect measurements, different priors for true positive rates among cases with differential measurement characteristics, and multiple-pathogen etiology. Scientists in Pneumonia Etiology Research for Child Health (PERCH) study usually refer to the etiology distribution as "population etiology pie" and "individual etiology pie" for their compositional nature, hence the name of the package (baking the pie).

Value

No returned value; documentation purpose only.

baker functions

nplcm()

See Also

- https://github.com/zhenkewu/baker for the source code and system/software requirements to use baker for your data.

---

beta_parms_from_quantiles

Pick parameters in the Beta distribution to match the specified range

---

Description

beta_parms_from_quantiles produces prior Beta parameters for the true positive rates (TPR)

Usage

```
beta_parms_from_quantiles(
  q,
  p = c(0.025, 0.975),
  precision = 0.001,
  derivative.epsilon = 0.001,
  start.with.normal.approx = TRUE,
  start = c(1, 1),
  plot = FALSE
)
```

Arguments

q	A vector of lower and upper bounds, in which Beta distribution will have quantiles specified by p. For example, q=c(0.5,0.99)
p	The lower and upper quantiles of the range one wants to specify.
precision	Approximation precisions.
derivative.epsilon	Precision of calculating derivative.
start.with.normal.approx	Default is TRUE, for normal approximation.
start	Starting values of beta parameters.
plot	Default is FALSE to suppress plotting of the beta density, otherwise, set to TRUE.

Value

A list containing the selected Beta parameters a, and b. Other elements of the list include some details about the computations involved in finding a and b.

References

http://www.medicine.mcgill.ca/epidemiology/Joseph/PBelisle/BetaParmsFromQuantiles.html

Examples

```
beta_parms_from_quantiles(c(0.5,0.99))
```

---

beta_plot *Plot beta density*

Description

Plot beta density

Usage

```
beta_plot(a, b)
```

Arguments

a	The first parameter
b	The second parameter

Value

None

Examples

```
beta_plot(2,2)
```

`bin2dec` *Convert a 0/1 binary-coded sequence into decimal digits*

Description

Useful when try to list all the binary patterns. One can group the binary sequences according to their equivalent decimal values.

Usage

```
bin2dec(binary_vector)
```

Arguments

`binary_vector`	a binary number

Value

a decimal number

Examples

```
bin2dec(c(1,0,1))
```

`check_dir_create` *check existence and create folder if non-existent*

Description

check existence and create folder if non-existent

Usage

```
check_dir_create(path)
```

Arguments

`path`	Folder path to check and create if not there.

Value

the same returned values for `dir.create()`

Examples

```
check_dir_create(tempdir())
```

`clean_combine_subsites` *Combine subsites in raw PERCH data set*

Description

In the Actual PERCH data set, a study site may have multiple subsites. `clean_combine_subsites` combines all the study subjects from the same site.

Usage

```
clean_combine_subsites(raw_meas_dir, subsites_list, newsites_vec)
```

Arguments

`raw_meas_dir`	The file path to the raw data file (.csv)
`subsites_list`	The list of subsite group names. Each group is a vector of subsites to be combined
`newsites_vec`	A vector of new site names. It has the same length as "`subsites_list`"

Value

A data frame with combined sites

`clean_perch_data` *Clean PERCH data*

Description

`clean_perch_data` transforms a raw data table (row for subjects, column for variables - named as `{pathogen name}_{specimen}{test}` for lab tests or other covariates) into a list. It is designed for PERCH data format.

Usage

```
clean_perch_data(clean_options)
```

Arguments

`clean_options` The list of options for cleaning PERCH data. Its elements are defined as follows:

`raw_meas_dir` : The file path to the raw data;

`case_def` : Variable name in raw data for **case** definition;

`case_def_val` : The value for **case** definition;

`ctrl_def` : Variable name in raw data for **control** definition;

`ctrl_def_val` : The value for **control** definition;

`X_strat` : A vector of variable names for stratifying the data to perform SEPARATE analyses;

`X_strat_val` : A list of values for `X_strat`. The output data only have individuals with `identical(X_strat,X_strat_val)==TRUE`. To perform analysis on a single site, say `"02GAM"`, use `X_strat="newSITE"` and `X_strat_val=list("02GAM")`;

`BrS_objects` : A list of BrS objects built by `make_meas_object()`;

`SS_objects` : A list of SS objects built by `make_meas_object()`;

`X_extra` : A vector of covariate names for regression or visualization;

`patho_taxo_dir` : The file path to the pathogen category or taxonomy information (.csv). The information should be as complete as possible for a particular analysis. If not, the pathogen without taxonomy information could not be assigned to bacterial or viral groups (see `plot_group_etiology()`); See `assign_taxo_cause_list()` that requires this taxonomy information..

Value

A List: `list(Mobs,Y,X)`

- `Mobs` A list of bronze- (`MBS`), silver- (`MSS`), and gold-standard (`MGS`, if available) measurements. See the formats of these measurements in `extract_data_raw()`.
- `Y` 1 for case; 0 for control;
- `X` Data frame of covariates for cases and controls. The covariate names are specified in `X_extra`;

This function does not re-order pathogens that only have silver-standard data.

See Also

make_meas_object for wrapping information about a particular type of measurement; extract_data_raw for reading raw data table and organizing them into `data_nplcm` format. Also see clean_combine_subsites for combining subsites and parse_date_time for parsing date.

`combine_data_nplcm` *combine multiple data_nplcm (useful when simulating data from regression models)*

Description

combine multiple data_nplcm (useful when simulating data from regression models)

Usage

```
combine_data_nplcm(data_nplcm_list)
```

Arguments

`data_nplcm_list`
: a list of data_nplcm in `nplcm()`

Value

a list with each element resulting from row binding of each corresponding element in the input `data_nplcm_list`.

See Also

Other data operation functions: `merge_lists()`, `subset_data_nplcm_by_index()`

Examples

```
N=100
Y = rep(c(1,0),times=50) # simulate two cases and two controls.
out_list <- vector("list",length=N)
J = 3                          # number of causes
cause_list = c(LETTERS[1:J])   # cause list
K = 2                          # number of subclasses
lambda = c(.8,.2)                # subclass weights for control group
eta = c(.9,.1)                   # subclass weights for case group

for (i in 1:N){
  #setup parameters for the present individual:
  set_parameter <- list(
    cause_list      = cause_list,
    etiology        = c(0.5,0.2,0.3), # only meaningful for cases
    pathogen_BrS    = LETTERS[1:J],
    pathogen_SS     = LETTERS[1:2],
    meas_nm         = list(MBS = c("MBS1"),MSS=c("MSS1")),
    Lambda          = lambda,         # for BrS
    Eta             = t(replicate(J,eta)),  # case subclass weight for BrS
    PsiBS           = cbind(c(0.15,0.3,0.35),
                            c(0.25,0.2,0.15)), # FPR
```

```
    PsiSS           = cbind(rep(0,J),rep(0,J)),
    ThetaBS         = cbind(c(0.95,0.9,0.85),    # TPR
                            c(0.95,0.9,0.85)),
    ThetaSS         = cbind(c(0.25,0.10),
                            c(0.25,0.10)),
    Nd      =     1,
    Nu      =     1
  )
  simu_out   <- simulate_nplcm(set_parameter)
  out <- simu_out$data_nplcm
  out_list[[i]] <- out
}

# extract cases and controls and combine all the data into one:
data_nplcm_list <- lapply(1:N, function(s) subset_data_nplcm_by_index(out_list[[s]],2-Y[s]))
data_nplcm_unordered      <- combine_data_nplcm(data_nplcm_list)
```

`compute_logOR_single_cause`

Calculate marginal log odds ratios

Description

This only works for single-agent causes

Usage

```
compute_logOR_single_cause(set_parameter)
```

Arguments

`set_parameter` True model parameters in an npLCM specification:

- `cause_list` a vector of disease class names among cases (since the causes could be multi-agent (e.g., multiple pathogens may cause an individual case's pneumonia), so its length could be longer than the total number of unique causative agents)
- `etiology` a vector of proportions that sum to 100 percent
- `pathogen_BrS` a vector of putative causative agents' names measured in bronze-standard (BrS) data. This function simulates only one slice defined by `specimen``test``pathogen`
- `pathogen_SS` a vector of pathogen names measured in silver-standard (SS) data.
- `meas_nm` a list of `specimen``test` names e.g., `list(MBS = c("NPPCR"),MSS="BCX")` for nasopharyngeal (NP) specimen tested by polymerase chain reaction (PCR) - `NPPCR` and blood (B) tested by culture (Cx) - `BCX`
- `Lambda` controls' subclass weights $\nu_1, \nu_2, \ldots, \nu_K$ a vector of `K` probabilities that sum to 1.

`Eta` a matrix of dimension `length(cause_list)` by `K`; each row represents a disease class (among cases); the values in that row are subclass weights $\eta_1, \eta_2, \ldots, \eta_K$ for that disease class, so needs to sum to one. In Wu et al. 2016 (JRSS-C), the subclass weights are the same across disease classes across rows. But when simulating data, one can specify rows with distinct subclass weights - it is a matter whether we can recover these parameters (possible when some cases' true disease classes are observed)

`PsiBS/PsiSS` False positive rates for Bronze-Standard data and for Silver-Standard data. For example, the rows of `PsiBS` correspond to the dimension of the particular slice of BrS measures, e.g., `10` for 10 causative agents measured by NPPCR; the columns correspond to `K` subclasses; generically, the dimension is `J` by `K` `PsiSS` is supposed to be a vector of all zeros (perfect specificity in silver-standard measures).

`ThetaBS/ThetaSS` True positive rates Θ for Bronze-Standard data and for Silver-Standard data. Dimension is `J` by `K` (can contain `NA` if the total number of causative agents measured by BrS or SS exceeds the measured causative agents in SS. For example, in PERCH study, nasopharyngeal polymerase chain reaction (NPPCR; bronze-standard) may target 30 distinct pathogens, but blood culture (BCX; silver-standard) may only target a subset of the 30, so we have to specify `NA` in `ThetaSS`for those pathogens not targeted by BCX).

`Nu` the number of control subjects

`Nd` the number of case subjects

Value

a matrix of log odds ratio. See the example for a figure showing pairwise odds ratios for cases (upper right, solid lines) and controls (lower left, broken lines) as the first subclass weight increases from 0 to 1. Pairwise independence is represented by the dotted horizontal lines for reference.

Examples

```
K.true  <- 2   # no. of latent subclasses in actual simulation.
               # If eta = c(1,0), effectively, it is K.true=1
J       <- 5   # no. of pathogens.
N       <- 500 # no. of cases/controls.

col_seq_cause <-  c("#DB9D85","#A2B367","#47BEA2",
"#70B3DA","#CD99D8")#colorspace::rainbow_hcl(5, start = 30, end = 300)

subclass_mix_seq <- seq(0,1,by=0.05)
res      <- array(NA,c(J,J,length(subclass_mix_seq)))
res_cond <- array(NA,c(J,J,length(subclass_mix_seq),J))

it <- layout(matrix(1:J^2,nrow=J,ncol=J,byrow=TRUE),
            heights = rep(3,J),
            widths  = rep(3,J))

oldpar <- par(oma=c(8,10,8,3));
```

```
pch_seq_cause <- LETTERS[1:J]
lty_seq_cause <- 1+(1:J)
pch_pos_seq   <- c(0.01); gap = 0.15
adj_seq <- c(0.15,0.5,0.85) # for roman numerals:
cex1       <- 2
cex_label1 <- 1
cex2       <- 2
cex_label2 <- 2
cex_margin_marks <- 2

for (scn in c(1,2,3)){
 for (iter in seq_along(subclass_mix_seq)){
   curr_mix <- subclass_mix_seq[iter]
   lambda <- c(curr_mix,1-curr_mix)
   eta    <- c(curr_mix,1-curr_mix)
   # if it is c(1,0),then it is conditional independence model, and
   # only the first column of parameters in PsiBS, ThetaBS matter!

   seed_start <- 20150923

   # set fixed simulation sequence:
   set.seed(seed_start)

   if (scn == 3){
     ThetaBS_withNA <- cbind(c(0.95,0.9,0.1,0.5,0.5),
                             c(0.95,0.1,0.9,0.5,0.5))
     PsiBS_withNA   <- cbind(c(0.4,0.4,0.05,0.2,0.2),
                             c(0.05,0.05,0.4,0.05,0.05))
   }

   if (scn == 2){
     ThetaBS_withNA <- cbind(c(0.95,0.5,0.5,0.5,0.5),
                             c(0.95,0.5,0.5,0.5,0.5))
     PsiBS_withNA   <- cbind(c(0.4,0.4,0.05,0.2,0.2),
                             c(0.05,0.05,0.4,0.05,0.05))
   }

   if (scn == 1){
     ThetaBS_withNA <- cbind(c(0.95,0.5,0.5,0.5,0.5),
                             c(0.95,0.5,0.5,0.5,0.5))
     PsiBS_withNA   <- cbind(c(0.3,0.3,0.15,0.2,0.2),
                             c(0.15,0.15,0.3,0.05,0.05))
   }

   # the following paramter names are set using names in the 'baker' package:
   set_parameter0 <- list(
     cause_list      = c(LETTERS[1:J]),
     etiology        = c(0.5,0.2,0.15,0.1,0.05), #same length as cause_list
     #etiology        = rep(0.2,J), #same length as cause_list
     pathogen_BrS    = LETTERS[1:J],
     meas_nm         = list(MBS = c("MBS1")),
     Lambda          = lambda,              #ctrl mix
     Eta             = t(replicate(J,eta)), #case mix, row number equal to Jcause.
```

```
    PsiBS           = PsiBS_withNA,
    ThetaBS         = ThetaBS_withNA,
    Nu      =     N, # control size.
    Nd      =     N  # case size.
  )

  res[,,iter] <- round(compute_logOR_single_cause(set_parameter0),2)

  for (pick in 1:J){
    set_parameter <- set_parameter0
    set_parameter$ThetaBS <- set_parameter0$PsiBS
    set_parameter$ThetaBS[pick,] <- set_parameter0$ThetaBS[pick,]
    set_parameter$etiology <- rep(0,J); set_parameter$etiology[pick] <- 1
    res_cond[,,iter,pick] <- round(compute_logOR_single_cause(set_parameter),2)
  }
}

ind <- sapply(c(0,0.5,1),function(x) which(subclass_mix_seq==x))
logOR_lim <- c(-2.15,2.15)
col_seq <- c("dodgerblue2","orange")
logOR_seq <- log(c(0.25,0.5,1,2,4))
pick_one <- 3

print(paste0("==Shading pairs of ",pch_seq_cause[pick_one]," and others.==="))
for (j in 1:J){
  for (l in 1:J){

    par(mar=c(0,0,0,0));
    if (j==J){
      par(mar=c(0,0,0,0))
    }
    if (l%%J==0){
      par(mar=c(0,0,0,1))
    }
    if (l%%J==1){
      par(mar=c(0,1,0,0))
    }
    if (!(j==l)){
      plot(res[j,l,],type="l",xlab="",ylab="",
           ylim=logOR_lim, lwd=5,
           xaxt="n",
           yaxt="n",
           col=col_seq[1+(l>j)],
           #lty=c(2,1)[1+(l>j)],
           lty=1,
           bty="n"
      )
      box(col="lightgray")
      abline(h=0,col="lightgray",lwd=3,lty=3)

      if (j<l){
        matplot(res_cond[j,l,,],type="l",add=TRUE,pch=LETTERS[1:J],lwd=2,lty=2,
                col=col_seq_cause)
```

```
}
lab_ord <- c(j,l); if (j>l){lab_ord <- rev(lab_ord)}
mtext(paste0("(",set_parameter$pathogen_BrS[lab_ord[1]],",",
             set_parameter$pathogen_BrS[lab_ord[2]],")"),
      side=3, adj=0.1,line=-2)

if (l%%J==1){
  axis(2,at = logOR_seq,
       labels = round(exp(logOR_seq),1),
       las=2,cex.axis=cex1)
}

if (l%%J==0){
  axis(4,at = logOR_seq,
       labels = round(exp(logOR_seq),1),
       las=2,cex.axis=cex1)
}

if (j==J){
  axis(1,at=seq_along(subclass_mix_seq)[ind],
  labels=rep("",length(ind)),cex.axis = cex1,las=1)
  axis(1,at=seq_along(subclass_mix_seq)[ind]+c(1,rep(0,length(ind)-2),-1),
  labels=subclass_mix_seq[ind],cex.axis = cex1,las=1,tick=FALSE)
}
if (j==1){
  axis(3,at=seq_along(subclass_mix_seq)[ind],
  labels=rep("",length(ind)),cex.axis = cex1,las=1)
  axis(3,at=seq_along(subclass_mix_seq)[ind]+c(1,rep(0,length(ind)-2),-1),
  labels=subclass_mix_seq[ind],cex.axis = cex1,las=1,tick=FALSE)
}
if (j==5 & l==1){
  mtext(expression(atop("Odds Ratio","(log-scale)")), side = 2, line = 4,
        cex=cex_label1, las=2)
}
if (j==5){
  mtext(expression(lambda[o]),side=1,line=4,cex=cex_label1)
}

if ((j<l) && (l==pick_one | j==pick_one )){
  # add shading cells for oen picked pathogen among cases:
  color <- rgb(190, 190, 190, alpha=80, maxColorValue=255)
  rect(par("usr")[1], par("usr")[3], par("usr")[2],
       par("usr")[4], density = 100, col = color)

matplot(res_cond[j,l,,],type="l",add=TRUE,lwd=2,col=col_seq_cause,lty=lty_seq_cause)
  for (ell in 1:J){
    where_add_letter <- quantile(seq_along(subclass_mix_seq),pch_pos_seq+gap*ell)
 points(where_add_letter, res_cond[j,l,where_add_letter,ell], pch=pch_seq_cause[ell])
  }
  mtext(paste0("(",set_parameter$pathogen_BrS[lab_ord[1]],",",
               set_parameter$pathogen_BrS[lab_ord[2]],")"),
        side=3, adj=0.1,line=-2)
}
```

```
      }else{

        plot(1, type="n", axes=FALSE, xlab="", ylab="", bty="n",
             xlim=c(0,1),ylim=c(0,1))


        if (j==3){
          text(labels=expression(CASES%up%""),x=.7,
               y=0.55,srt=-49,col=col_seq[2],cex=1.8,adj=0.5,font=4)
          text(labels=expression(CONTROLS%down%""),x=.42,
               y=0.38,srt=-49,col=col_seq[1],cex=1.8,adj=0.5,font=4)
        }
        if (j!=1 & j!=J){
          dg <- par("usr")
          segments(dg[1],dg[4],dg[2],dg[3], col='lightgray',lwd=3)
        }
        if (j==J){
          legend("top",LETTERS[1:J],lty=2,col=col_seq_cause,cex = 1.5,lwd=2,
                 bty="n",horiz=FALSE)
        }
      }
    }
  }
}
par(oldpar)
```

`compute_marg_PR_nested_reg` *compute positive rates for nested model with subclass mixing weights that are the same across* `Jcause` *classes for each person (people may have different weights.)*

Description

The array version of this function (compute_marg_PR_nested_reg_array) is used in plot_etiology_regression

Usage

```
compute_marg_PR_nested_reg(ThetaBS, PsiBS, pEti_mat, subwt_mat, case, template)
```

Arguments

`ThetaBS`	True positive rates for `JBrS` measures (rows) among `K` subclasses (columns)
`PsiBS`	False positive rates; dimension same as above
`pEti_mat`	a matrix of etiology pies for `N` subjects (rows) and `Jcause` causes (columns) rows sum to ones.

subwt_mat	a matrix of subclass weights for cases and controls. N by K. Rows sum to ones.
case	a N-vector of 1s (cases) and 0s (controls)
template	a binary matrix with Jcause+1 rows (Jcause classes of cases and 1 class of controls) and JBrS columns for the Bronze-standard measurement (say, pick one type/slice). The ones in each row indicate the measurements that will show up more frequently in cases given the cause.

Value

a matrix of values between 0 and 1 (need not to have row sums of ones); of dimension (number of subjects, dimension of the bronze-standard measurement slice).

---

compute_marg_PR_nested_reg_array

compute positive rates for nested model with subclass mixing weights that are the same across Jcause *classes for each person (people may have different weights.)*

Description

This is an array-version of compute_marg_PR_nested_reg. This is used in plot_etiology_regression

Usage

```
compute_marg_PR_nested_reg_array(
  ThetaBS_array,
  PsiBS_array,
  pEti_mat_array,
  subwt_mat_array,
  case,
  template
)
```

Arguments

ThetaBS_array	An array of: True positive rates for JBrS measures (rows) among K subclasses (columns)
PsiBS_array	An array of: False positive rates; dimension same as above
pEti_mat_array	An array of: a matrix of etiology pies for N subjects (rows) and Jcause causes (columns) rows sum to ones.
subwt_mat_array	An array of: a matrix of subclass weights for cases and controls. N by K. Rows sum to ones.
case	a N-vector of 1s (cases) and 0s (controls)
template	a binary matrix with Jcause+1 rows (Jcause classes of cases and 1 class of controls) and JBrS columns for the Bronze-standard measurement (say, pick one type/slice). The ones in each row indicate the measurements that will show up more frequently in cases given the cause.

Value

An array of: a matrix of values between 0 and 1 (need not to have row sums of ones); of dimension (number of subjects, dimension of the bronze-standard measurement slice).

`create_bugs_regressor_Eti` *create regressor summation equation used in regression for etiology*

Description

`create_bugs_regressor_Eti` creates linear product of coefficients and a row of design matrix used in regression

Usage

```
create_bugs_regressor_Eti(
  n,
  dm_nm = "dm_Eti",
  b_nm = "betaEti",
  ind_nm = "j",
  sub_ind_nm = "k"
)
```

Arguments

`n`	the length of coefficients
`dm_nm`	name of design matrix; default `"dm_Eti"`
`b_nm`	name of the coefficients; default `"betaEti"`
`ind_nm`	name of the coefficient iterator; default `"j"`
`sub_ind_nm`	name of the subject iterator; default `"k"`

Value

a character string with linear product form

create_bugs_regressor_FPR *create regressor summation equation used in regression for FPR*

Description

`create_bugs_regressor_FPR` creates linear product of coefficients and a row of design matrix used in regression

Usage

```
create_bugs_regressor_FPR(
  n,
  dm_nm = "dm_FPR",
  b_nm = "b",
  ind_nm = "j",
  sub_ind_nm = "k"
)
```

Arguments

`n`	the length of coefficients
`dm_nm`	name of design matrix; default `"dm_FPR"`
`b_nm`	name of the coefficients; default `"b"`
`ind_nm`	name of the coefficient iterator; default `"j"`
`sub_ind_nm`	name of the subject iterator; default `"k"`

Value

a character string with linear product form

data_nplcm_noreg *Simulated dataset that is structured in the format necessary for an `nplcm()` without regression*

Description

Data set for illustrating regression functionalities

Usage

```
data("data_nplcm_noreg")
```

Format

A list containing three items

Mobs BrS level measurements: N = 600 (half cases and half controls); one slice of BrS measurements (6 dimensional, A-F); one slice of SS measurements (2 dimensional, A and B)

Y case-control status

Value

No returned value; just loading data into the working space.

---

data_nplcm_reg_nest *Simulated dataset that is structured in the format necessary for an* nplcm() *with regression*

Description

Data set for illustrating regression functionalities

Usage

```
data("data_nplcm_reg_nest")
```

Format

A list containing three items

Mobs BrS level measurements: N = 1,200 (half cases and half controls); one slice of BrS measurements (6 dimensional, A-F); one slice of SS measurements (2 dimensional, A and B)

Y case-control status

X matrix of covariates (N by 4); columns: SITE (1 and 2, each with 600 subjects), DATE (index from 1:300), std_date (standardized DATE), ENRLDATE (actual dates)

Value

No returned value; just loading data into the working space.

delete_start_with *Deletes a pattern from the start of a string, or each of a vector of strings.*

Description

`delete_start_with` is used for clean the column names in raw data. For example, R adds "X" at the start of variable names. This function deletes "X_"s from the column names. This can happen if the raw data have column names such as "`_CASE_ABX`". Check `clean_perch_data()` for its actual usage.

Usage

```
delete_start_with(s, vec)
```

Arguments

`s`	the pattern (a single string) to be deleted from the start.
`vec`	a vector of strings with unwanted starting strings (specified by `s`).

Value

string(s) with deleted patterns from the start.

Examples

```
delete_start_with("X_",c("X_hello"))
delete_start_with("X_",c("X_hello","hello2"))
delete_start_with("X_",c("X_hello","hello2","X_hello3"))
```

dm_Rdate_Eti *Make etiology design matrix for dates with R format.*

Description

`dm_Rdate_Eti` creates design matrices for etiology regressions.

Usage

```
dm_Rdate_Eti(Rdate, Y, num_knots_Eti, basis_Eti = "ncs")
```

Arguments

Rdate	a vector of dates of R format
Y	binary case/control status; 1 for case; 0 for controls
num_knots_Eti	number of knots for etiology regression
basis_Eti	the type of basis functions to use for etiology regression. It can be "ncs" (natural cubic splines) or "tprs" (thin-plate regression splines). Default is "ncs". "tprs" will be implemented later.

Details

It is used in `model_options$likeihood$Eti_formula`. For example, one can specify it as:

`~ AGECAT+HIV+dm_Rdate_Eti(ENRLDATE,Y,5)`

to call an etiology regression with intercept, main effects for 'AGECAT' and 'HIV', and natural cubic spline bases for 'ENRLDATE' using 5 knots defined as 5 equal-probability-spaced sample quantiles.

Value

Design matrix for etiology regression:

- `Z_Eti` transformed design matrix for etiology regression

See Also

`nplcm()`

---

dm_Rdate_FPR *Make FPR design matrix for dates with R format.*

Description

`dm_Rdate_FPR` creates design matrices for false positive rate regressions; can also be used to standardize dates.

Usage

```
dm_Rdate_FPR(Rdate, Y, effect = "fixed", num_knots_FPR = NULL)
```

Arguments

Rdate	a vector of dates of R format
Y	binary case/control status; 1 for case; 0 for controls
effect	The design matrix for "random" or "fixed" effect; Default is "fixed". When specified as "fixed", it produces standardized R-format dates using control's mean and standard deviation; When specified as "random", it produces num_knots_FPR columns of design matrix for thin-plate regression splines (TPRS) fitting. One needs both "fixed" and "random" in a FPR regression formula in model_options to enable TPRS fitting. For example, model_options$likelihood$FPR_formula can be ~ AGECAT+HIV+dm_Rdate_FPR(ENRLDATE,Y,"fixed")+dm_Rdate_FPR(ENRLDATE,Y,"random",10) means FPR regression with intercept, main effects for 'AGECAT' and 'HIV', and TPRS bases for 'ENRLDATE' using 10 knots placed at 10 equal-probability-spaced sample quantiles.
num_knots_FPR	number of knots for FPR regression; default is NULL to accommodate fixed effect specification.

Value

Design matrix for FPR regression:

- Z_FPR_ctrl transformed design matrix for FPR regression for controls
- Z_FPR_case transformed design matrix for borrowing FPR regression from controls to cases. It is obtained using control-standardization, and square-root the following matrix (Ω]) with (j_1,j_2) element being

$$\Omega_{j_1 j_2} = \|knots_{j_1} - knots_{j_2}\|^3$$

.

See Also

nplcm()

---

expit *expit function*

Description

expit function

Usage

```
expit(x)
```

Arguments

x	A real number

Value

a Probability between 0 and 1

Examples

```
expit(-0.1)
```

`extract_data_raw` *Import Raw PERCH Data* `extract_data_raw` *imports and converts the raw data to analyzable format*

Description

Import Raw PERCH Data

`extract_data_raw` imports and converts the raw data to analyzable format

Usage

```
extract_data_raw(
  dat_prepared,
  strat_nm,
  strat_val,
  meas_object,
  extra_covariates = NULL
)
```

Arguments

dat_prepared	The data set prepared in `clean_perch_data`.
strat_nm	The vector of covariate names to separately extract data. For example, in PERCH data cleaning, `X = c("newSITE","CASECONT")`.
strat_val	The list of covariate values to stratify data. Each element corresponds to elements in `X`. For example, in PERCH data cleaning, `Xval = list("02GAM","1")`.
meas_object	A list of bronze-standard or silver-standard measurement objects made by function `make_meas_object()`.
extra_covariates	The vector of covariate name for regression purposes. The default is NULL, which means not reading in any covariate.

Value

A list of data.

Mobs **MBS** A list of Bronze-Standard (BrS) measurements. The names of the list take the form of `specimen_test`. Each element of the list is a data frame. The rows of the data frame are for subjects; the columns are for measured pathogens.

MSS A list of Silver-Standard (SS) measurements. The formats are the same as `MBS` above.

MGS A list of Gold-Standard (GS) measurements. It equals `NULL` if no GS data exist.

X A data frame with columns specified by `extra_covariates`.

See Also

`clean_perch_data()`

Other raw data importing functions: `read_meas_object()`

`get_coverage` *Obtain coverage status from a result folder*

Description

Obtain coverage status from a result folder

Usage

```
get_coverage(DIR_NPLCM, truth)
```

Arguments

`DIR_NPLCM`	Path to where Bayesian results are stored
`truth`	True etiologic fraction vector (must sum to 1) used to generate data.

Value

A logic vector of length as `truth`. 1 for covered; 0 for not.

`get_direct_bias` *Obtain direct bias that measure the discrepancy of a posterior distribution of pie and a true pie.*

Description

Obtain direct bias that measure the discrepancy of a posterior distribution of pie and a true pie.

Usage

```
get_direct_bias(DIR_list, truth = NULL, silent = FALSE)
```

Arguments

`DIR_list`	The list of where Bayesian results are stored
`truth`	True etiologic fraction vector (must sum to 1) used to generate data; Default is `NULL`. If a vector is supplied, then only the first path in `DIR_LIST` is used.
`silent`	Default is FALSE. To suppress printing messages, set to TRUE.

Value

a list of length two. `diff` is the direct differences; `prb` is the percent relative bias.

`get_fitted_mean_nested` *get fitted mean for nested model with subclass mixing weights that are the same among cases*

Description

get fitted mean for nested model with subclass mixing weights that are the same among cases

Usage

```
get_fitted_mean_nested(
  slice,
  res_nplcm,
  model_options,
  data_nplcm,
  clean_options
)
```

Arguments

slice	the slice of BrS data that are modeled
res_nplcm	matrix of MCMC samples
model_options	see nplcm()
data_nplcm	see nplcm()
clean_options	see clean_perch_data()

Value

a matrix of no. of rows equal to retained MCMC samples, no. of columns equal to the no. of measurement dimensions within a slice.

---

get_fitted_mean_no_nested

get model fitted mean for conditional independence model

Description

get model fitted mean for conditional independence model

Usage

```
get_fitted_mean_no_nested(
  slice,
  res_nplcm,
  model_options,
  data_nplcm,
  clean_options
)
```

Arguments

slice	the slice of BrS data that are modeled
res_nplcm	matrix of MCMC samples
model_options	see nplcm()
data_nplcm	see nplcm()
clean_options	see clean_perch_data()

Value

a list with model fitted means

get_individual_data *get individual data*

Description

get individual data

Usage

```
get_individual_data(i, data_nplcm)
```

Arguments

i	index of individual as appeared in data_nplcm
data_nplcm	the data for nplcm; see nplcm()

Value

a list of the same structure as data_nplcm; just with one row of values

Examples

```
data(data_nplcm_noreg)
get_individual_data(2,data_nplcm_noreg)
```

get_individual_prediction *get individual prediction (Bayesian posterior)*

Description

must set individual.pred = TRUE in MCMC options (see the example of this function)

Usage

```
get_individual_prediction(x)
```

Arguments

x	an nplcm object; it contains the file path DIR_NPLCM to where the model results and specifications are stored. The function first reads a list from this folder by nplcm_read_folder()

Value

a matrix of individual predictions; rows for cases, columns for causes specified in `model_options$likelihood$cause_list`
See `nplcm()`

Examples

```
data(data_nplcm_noreg)
cause_list <- LETTERS[1:6]
J.BrS      <- 6
model_options_no_reg <- list(
  likelihood   = list(
    cause_list = cause_list,
    k_subclass = 2,
    Eti_formula = ~-1, # no covariate for the etiology regression
    FPR_formula = list(
      MBS1 =   ~-1)    # no covariate for the subclass weight regression
  ),
  use_measurements = c("BrS"),
  # use bronze-standard data only for model estimation.
  prior= list(
    Eti_prior = overall_uniform(1,cause_list),
    # Dirichlet(1,...,1) prior for the etiology.
    TPR_prior  = list(BrS = list(
      info  = "informative", # informative prior for TPRs
      input = "match_range",
      # specify the informative prior for TPRs by specifying a plausible range.
      val = list(MBS1 = list(up =  list(rep(0.99,J.BrS)),
                             # upper ranges: matched to 97.5% quantile of a Beta prior
                             low = list(rep(0.55,J.BrS))))
      # lower ranges: matched to 2.5% quantile of a Beta prior
    )
    )
  )
)


set.seed(1)
# include stratification information in file name:
thedir    <- paste0(tempdir(),"_no_reg")

# create folders to store the model results
dir.create(thedir, showWarnings = FALSE)
result_folder_no_reg <- file.path(thedir,paste("results",collapse="_"))
thedir <- result_folder_no_reg
dir.create(thedir, showWarnings = FALSE)

# options for MCMC chains:
mcmc_options_no_reg <- list(
  debugstatus = TRUE,
  n.chains = 1,
  n.itermcmc = as.integer(200),
  n.burnin = as.integer(100),
```

```
  n.thin = 1,
  individual.pred = TRUE, # <- must set to TRUE!
  ppd = FALSE,
  result.folder = thedir,
  bugsmodel.dir = thedir
)

BrS_object_1 <- make_meas_object(patho = LETTERS[1:6],
                                 specimen = "MBS", test = "1",
                                 quality = "BrS", cause_list = cause_list)
clean_options <- list(BrS_objects = make_list(BrS_object_1))
# place the nplcm data and cleaning options into the results folder
dput(data_nplcm_noreg,file.path(thedir,"data_nplcm.txt"))
dput(clean_options, file.path(thedir, "data_clean_options.txt"))

rjags::load.module("glm")

fitted_nplcm_noreg <- nplcm(data_nplcm_noreg,model_options_no_reg,mcmc_options_no_reg)
image(get_individual_prediction(fitted_nplcm_noreg))
```

`get_latent_seq` *get index of latent status*

Description

get index of latent status

Usage

```
get_latent_seq(cause_list, ord, select_latent = NULL, exact = TRUE)
```

Arguments

`cause_list`	see mode_options in `nplcm()`
`ord`	order of cause_list according to posterior mean
`select_latent`	Default is NULL
`exact`	Default is TRUE

Value

a vector of indices

`get_marginal_rates_nested` *get marginal TPR and FPR for nested model*

Description

get marginal TPR and FPR for nested model

Usage

```
get_marginal_rates_nested(slice, res_nplcm, model_options, data_nplcm)
```

Arguments

`slice`	the slice of BrS data that are modeled
`res_nplcm`	matrix of MCMC samples
`model_options`	see `nplcm()`
`data_nplcm`	see `nplcm()`

Value

a matrix of no. of rows equal to retained MCMC samples, no. of columns equal to the no. of measurement dimensions within a slice.

`get_marginal_rates_no_nested` *get marginal TPR and FPR for no nested model*

Description

get marginal TPR and FPR for no nested model

Usage

```
get_marginal_rates_no_nested(slice, res_nplcm, model_options, data_nplcm)
```

Arguments

`slice`	the slice of BrS data that are modeled
`res_nplcm`	matrix of MCMC samples
`model_options`	see `nplcm()`
`data_nplcm`	see `nplcm()`

Value

a matrix of no. of rows equal to retained MCMC samples, no. of columns equal to the no. of measurement dimensions within a slice.

get_metric — *Obtain Integrated Squared Aitchison Distance, Squared Bias and Variance (both on Central Log-Ratio transformed scale) that measure the discrepancy of a posterior distribution of pie and a true pie.*

Description

The result is equivalent to Euclidean-type calculation after the compositional vector (e.g., etiologic fraction) is centered-log-ratio (CLRB) transformed. For simulation only.

Usage

```
get_metric(DIR_NPLCM, truth)
```

Arguments

DIR_NPLCM	File path where Bayesian results are stored
truth	True etiologic fraction vector (must sum to 1) used to generate data

Value

a vector of (Integrated Squared Aitchison Distance (ISAD), bias-squared, variance, truth)

get_pEti_samp — *get etiology samples by names (no regression)*

Description

get etiology samples by names (no regression)

Usage

```
get_pEti_samp(res_nplcm, model_options)
```

Arguments

res_nplcm	result from model fits
model_options	model specification

Value

A list:

pEti_mat: a matrix of posterior samples (iteration by cause); overall etiology latent_nm: a vector of character strings representing the names of the causes

get_plot_num *get the plotting positions (numeric) for the fitted means; 3 positions for each cell*

Description

get the plotting positions (numeric) for the fitted means; 3 positions for each cell

Usage

```
get_plot_num(e, height)
```

Arguments

e	Integer index from 1 to length(cause_list)
height	the total number of causes

Value

a triple with numerical plotting positions

get_plot_pos *get a list of measurement index where to look for data*

Description

get a list of measurement index where to look for data

Usage

```
get_plot_pos(template)
```

Arguments

template	See nplcm()

Value

a list of index vectors

get_postsd *Obtain posterior standard deviation from a result folder*

Description

Obtain posterior standard deviation from a result folder

Usage

```
get_postsd(DIR_NPLCM)
```

Arguments

DIR_NPLCM	Path to where Bayesian results are stored

Value

a vector of positive numbers

get_top_pattern *get top patterns from a slice of bronze-standard measurement*

Description

get top patterns from a slice of bronze-standard measurement

Usage

```
get_top_pattern(BrS_dat, Y, case_status, n_pat, exclude_missing = TRUE)
```

Arguments

BrS_dat	bronze-standard data, which is usually data_nplcm$Mobs$MBS[[1]]
Y	A vector of case/control status: 1 for case; 0 for control
case_status	1 for case; 0 for controls
n_pat	the number of top patterns one wants to show
exclude_missing	DEFAULT is TRUE for excluding any individual with missing measurements.

Value

a list of results: obs_pat - observed rates; pattern_names; exist_other - if actual no. of patterns is larger than n_pat; N- No. of individuals with Y = case_status.

See Also

Other exploratory data analysis functions: plot_logORmat(), show_individual(), summarize_BrS(), summarize_SS(), visualize_season()

Examples

```
data(data_nplcm_noreg)
get_top_pattern(data_nplcm_noreg$Mobs$MBS[[1]],data_nplcm_noreg$Y,1,5,FALSE)

data(data_nplcm_noreg)
get_top_pattern(data_nplcm_noreg$Mobs$MBS$MBS1,data_nplcm_noreg$Y,case_status=1,n_pat=5)
```

H *Shannon entropy for multivariate discrete data*

Description

Shannon entropy for multivariate discrete data

Usage

```
H(px)
```

Arguments

px	a vector of positive numbers sum to 1

Value

a non-negative number

Examples

```
H(c(0.5,0.3,0.2))
```

has_non_basis *test if a formula has terms not created by [s_date_Eti() or s_date_FPR()*

Description

test if a formula has terms not created by [s_date_Eti() or s_date_FPR()

Usage

```
has_non_basis(form)
```

Arguments

form	a formula

Value

logical TRUE (if having terms not created by [s_date_Eti() or s_date_FPR()); FALSE otherwise.

Examples

```
form1 <- as.formula(~ -1+s_date_FPR(DATE,Y,basis = "ps",10) + as.factor(SITE))
form2 <- as.formula(~ -1+s_date_FPR(DATE,Y,basis = "ps",10))
form3 <- as.formula(~ s_date_FPR(DATE,Y,basis = "ps",10))

has_non_basis(form1)
has_non_basis(form2)
has_non_basis(form3)
```

I2symb *Convert 0/1 coding to pathogen/combinations*

Description

Reverse to symb2I()

Usage

```
I2symb(binary_code, pathogen_list)
```

Arguments

binary_code	Binary indicators for pathogens
pathogen_list	The complete list of pathogen names

Value

The name of pathogen or pathogen combination indicated by "code"

Examples

```
I2symb("001",c("A","B","C"))
I2symb("000",c("A","B","C"))
```

`Imat2cat` *Convert a matrix of binary indicators to categorical variables*

Description

Convert a matrix of binary indicators to categorical variables

Usage

```
Imat2cat(binary_mat, cause_list, pathogen_list)
```

Arguments

`binary_mat`	The matrix of binary indicators. Rows for subjects, columns for pathogens in the "`pathogen.list`"
`cause_list`	The list of causes
`pathogen_list`	The complete list of pathogen names

Value

A vector of categorical variables. Its length equals the length of "`allowed.list`"

Examples

```
Imat2cat(rbind(diag(3),c(1,1,0),c(0,0,0)),c("A","B","C","A+B","NoA"),c("A","B","C"))
```

init_latent_jags_multipleSS

Initialize individual latent status (for JAGS)

Description

Initialize individual latent status (for JAGS)

Usage

```
init_latent_jags_multipleSS(
  MSS_list,
  cause_list,
  patho = unlist(lapply(MSS_list, colnames))
)
```

Arguments

MSS_list	A list of silver-standard measurement data, possibly with more than one slices; see data_nplcm argument in nplcm()
cause_list	See model_options arguments in nplcm()
patho	A vector of measured pathogen name for MSS; default is colnames(MSS)

Details

In JAGS 3.4.0, if an initial value contradicts the probabilistic specification, e.g. MSS_1[i,j] ~ dbern(mu_ss_1[i,j]), where MSS_1[i,j]=1 but mu_ss_1[i,j]=0, then JAGS cannot understand it. In PERCH application, this is most likely used when the specificity of the silver-standard data is 1. Note: this is not a problem in WinBUGS.

Value

a list of numbers, indicating categories of individual latent causes.

insert_bugfile_chunk_noreg_etiology

insert distribution for latent status code chunk into .bug file

Description

insert distribution for latent status code chunk into .bug file

Usage

```
insert_bugfile_chunk_noreg_etiology(ppd = NULL)
```

Arguments

ppd	Default is NULL; set to TRUE for posterior predictive checking

Value

a long character string to be inserted into .bug model file as distribution specification for latent status

---

insert_bugfile_chunk_noreg_meas

Insert measurement likelihood (without regression) code chunks into .bug model file

---

Description

Insert measurement likelihood (without regression) code chunks into .bug model file

Usage

```
insert_bugfile_chunk_noreg_meas(
  k_subclass,
  Mobs,
  prior,
  cause_list,
  use_measurements = "BrS",
  ppd = NULL,
  use_jags = FALSE
)
```

Arguments

k_subclass	the number of subclasses for the slices that require conditional dependence modeling (only applicable to BrS data); its length is of the same value as the number of BrS slices.
Mobs	measurement data in the form of data_nplcm
prior	prior specification from model_options
cause_list	a list of latent status names (crucial for building templates; see make_template())
use_measurements	"BrS", or "SS"
ppd	Default is NULL; set to TRUE for posterior predictive checking
use_jags	Default is FALSE; set to TRUE if want to use JAGS for model fitting.

Value

a long character string to be inserted into .bug model file as measurement likelihood

See Also

It is used in write_model_NoReg for constructing a .bug file along with specification of latent status distribution (insert_bugfile_chunk_noreg_etiology)

`insert_bugfile_chunk_reg_discrete_predictor_etiology`
insert etiology regression for latent status code chunk into .bug file; discrete predictors

---

Description

insert etiology regression for latent status code chunk into .bug file; discrete predictors

Usage

```
insert_bugfile_chunk_reg_discrete_predictor_etiology(Jcause, ppd = NULL)
```

Arguments

`Jcause`	The number of distinct causes, i.e., categories of latent health status; equals `length(model_options$likelihood$cause_list)`.
`ppd`	Default is NULL; set to TRUE for posterior predictive checking

Value

a long character string to be inserted into .bug model file as distribution specification for latent status

---

`insert_bugfile_chunk_reg_discrete_predictor_nonest_meas`
Insert measurement likelihood (with regression; discrete) code chunks into .bug model file

---

Description

Insert measurement likelihood (with regression; discrete) code chunks into .bug model file

Usage

```
insert_bugfile_chunk_reg_discrete_predictor_nonest_meas(
  Mobs,
  prior,
  cause_list,
  use_measurements = "BrS",
  ppd = NULL,
  use_jags = FALSE
)
```

Arguments

Mobs	Measurement data in the form of data_nplcm
prior	Prior specification from model_options
cause_list	A list of latent status names (crucial for building templates; see make_template())
use_measurements	"BrS", or "SS"
ppd	Default is NULL; set to TRUE for posterior predictive checking
use_jags	Default is FALSE; set to TRUE if want to use JAGS for model fitting.

Value

A long character string to be inserted into .bug model file as measurement likelihood

See Also

It is used in write_model_Reg_NoNest for constructing a .bug file along with specification of latent status regression (insert_bugfile_chunk_reg_etiology)

---

insert_bugfile_chunk_reg_etiology

insert etiology regression for latent status code chunk into .bug file

---

Description

insert etiology regression for latent status code chunk into .bug file

Usage

```
insert_bugfile_chunk_reg_etiology(Eti_formula, Jcause, ppd = NULL)
```

Arguments

Eti_formula	Etiology regression formula; Check model_options$likelihood$Eti_formula.
Jcause	The number of distinct causes, i.e., categories of latent health status; equals length(model_options$likelihood$cause_list).
ppd	Default is NULL; set to TRUE for posterior predictive checking

Value

a long character string to be inserted into .bug model file as distribution specification for latent status

`insert_bugfile_chunk_reg_nest_meas` *Insert measurement likelihood (nested model+regression) code chunks into .bug model file*

Description

Insert measurement likelihood (nested model+regression) code chunks into .bug model file

Usage

```
insert_bugfile_chunk_reg_nest_meas(
  Mobs,
  prior,
  cause_list,
  FPR_formula,
  use_measurements = "BrS",
  ppd = NULL,
  use_jags = FALSE
)
```

Arguments

`Mobs`	Measurement data in the form of `data_nplcm`
`prior`	Prior specification from `model_options`
`cause_list`	A list of latent status names (crucial for building templates; see `make_template()`)
`FPR_formula`	A list of FPR regression formula; check `model_options$likelihood$FPR_formula`
`use_measurements`	"BrS", or "SS"
`ppd`	Default is NULL; set to TRUE for posterior predictive checking
`use_jags`	Default is FALSE; set to TRUE if want to use JAGS for model fitting.

Value

A long character string to be inserted into .bug model file as measurement likelihood

See Also

Called by write_model_Reg_NoNest for constructing a `.bug` file. This is usually called along with specification of latent status regression (insert_bugfile_chunk_reg_etiology).

`insert_bugfile_chunk_reg_nonest_meas` *Insert measurement likelihood (with regression) code chunks into .bug model file*

Description

Insert measurement likelihood (with regression) code chunks into .bug model file

Usage

```
insert_bugfile_chunk_reg_nonest_meas(
  Mobs,
  prior,
  cause_list,
  FPR_formula,
  use_measurements = "BrS",
  ppd = NULL,
  use_jags = FALSE
)
```

Arguments

`Mobs`	Measurement data in the form of `data_nplcm`
`prior`	Prior specification from `model_options`
`cause_list`	A list of latent status names (crucial for building templates; see `make_template()`)
`FPR_formula`	A list of FPR regression formula; check `model_options$likelihood$FPR_formula`
`use_measurements`	"BrS", or "SS"
`ppd`	Default is NULL; set to TRUE for posterior predictive checking
`use_jags`	Default is FALSE; set to TRUE if want to use JAGS for model fitting.

Value

A long character string to be inserted into .bug model file as measurement likelihood

See Also

It is used in write_model_Reg_NoNest for constructing a .bug file along with specification of latent status regression (insert_bugfile_chunk_reg_etiology)

is.error *Test for 'try-error' class*

Description

Test for 'try-error' class

Usage

```
is.error(x)
```

Arguments

x	An object to be test if it is "try-error"

Value

Logical. TRUE for "try-error"; FALSE otherwise

References

http://adv-r.had.co.nz/Exceptions-Debugging.html

is_discrete *Check if covariates are discrete*

Description

is_discrete checks if the specified covariates could be regarded as discrete variables.

Usage

```
is_discrete(X, X_reg)
```

Arguments

X	A data frame of covariates
X_reg	The vector of covariates that will stratify the analyses. These variables have to be categorical. Or a formula (can be tested by is.formula in plyr), e.g., ~as.factor(SITE8) + as.factor(AGECAT > 1).

Details

Note that this function should be used with caution. It used

$$nrow(X)/nrow(unique(X[, X_reg, drop = FALSE])) > 10$$

as an *ad hoc* criterion. It is not the same as is.discrete() in plyr

Value

TRUE for all being discrete; FALSE otherwise.

---

is_intercept_only *check if the formula is intercept only*

Description

outputs logical values for a formula; to identify intercept-only formula.

Usage

```
is_intercept_only(form)
```

Arguments

form	Regression formula

Value

TRUE for intercept-only; FALSE otherwise

---

is_jags_folder *See if a result folder is obtained by JAGS*

Description

See if a result folder is obtained by JAGS

Usage

```
is_jags_folder(DIR_NPLCM)
```

Arguments

DIR_NPLCM	directory to the folder with results. "mcmc_options.txt" must be in the folder.

Value

TRUE for from JAGS; FALSE otherwise.

Examples

```
is_jags_folder(tempdir()) # just an illustration.
```

is_length_all_one *check if a list has elements all of length one*

Description

check if a list has elements all of length one

Usage

```
is_length_all_one(x)
```

Arguments

x	a list

Value

TRUE or FALSE

Examples

```
l = list(a = 5, b = 1:2)
is_length_all_one(l) # FALSE
l = list(a = 5, b = 1)
is_length_all_one(l) # TRUE
```

jags2_baker *Run* JAGS *from R*

Description

The jags function takes data and starting values as input. It automatically writes a jags script, calls the model, and saves the simulations for easy access in R. Check the R2jags::jags2 for details about the argument.

Usage

```
jags2_baker(
  data,
  inits,
  parameters.to.save,
  model.file = "model.bug",
  n.chains = 3,
  n.iter = 2000,
  n.burnin = floor(n.iter/2),
  n.thin = max(1, floor((n.iter - n.burnin)/1000)),
```

```
  DIC = TRUE,
  jags.path = "",
  working.directory = NULL,
  clearWD = TRUE,
  refresh = n.iter/50
)
```

Arguments

data	(1) a vector or list of the names of the data objects used by the model, (2) a (named) list of the data objects themselves, or (3) the name of a "dump" format file containing the data objects, which must end in ".txt", see example below for details.
inits	a list with n.chains elements; each element of the list is itself a list of starting values for the BUGS model, *or* a function creating (possibly random) initial values. If inits is NULL, JAGS will generate initial values for parameters.
parameters.to.save	character vector of the names of the parameters to save which should be monitored.
model.file	file containing the model written in BUGS code. Alternatively, as in **R2WinBUGS**, model.file can be an R function that contains a BUGS model that is written to a temporary model file (see tempfile) using write.model
n.chains	number of Markov chains (default: 3)
n.iter	number of total iterations per chain (including burn in; default: 2000)
n.burnin	length of burn in, i.e. number of iterations to discard at the beginning. Default is n.iter/2, that is, discarding the first half of the simulations. If n.burnin is 0, jags() will run 100 iterations for adaption.
n.thin	thinning rate. Must be a positive integer. Set n.thin > 1 to save memory and computation time if n.iter is large. Default is max(1, floor(n.chains * (n.iter-n.burnin) / 1000)) which will only thin if there are at least 2000 simulations.
DIC	logical; if TRUE (default), compute deviance, pD, and DIC. The rule pD=var(deviance) / 2 is used.
jags.path	directory that contains the JAGS executable. The default is "".
working.directory	sets working directory during execution of this function; This should be the directory where model file is.
clearWD	indicating whether the files 'data.txt', 'inits[1:n.chains].txt', 'codaIndex.txt', 'jagsscript.txt', and 'CODAchain[1:nchains].txt' should be removed after jags has finished, default=TRUE.
refresh	refresh frequency for progress bar, default is n.iter/50

Details

This modifies the jags2 function in R2jags package.

Value

Same as R2jags::jags()

See Also

R2jags::jags()

`line2user` *convert line to user coordinates*

Description

Here's a version that works with log-scale and linear scale axes. The trick is to express line locations in npc coordinates rather than user coordinates, since the latter are of course not linear when axes are on log scales.

Usage

```
line2user(line, side)
```

Arguments

`line`	integer
`side`	integer; 1-4

Details

`par('cin')[2] * par('cex') * par('lheight')` returns the current line height in inches, which we convert to user coordinates by multiplying by `diff(grconvertX(0:1, 'inches', 'user'))`, the length of an inch in user coordinates (horizontally, in this case - if interested in the vertical height of a line in user coords we would use `diff(grconvertY(0:1, 'inches', 'user'))`).

Value

a numeric vector of the same length as `line`; the values represent the coordinates in the current plot and are converted from `line`.

References

https://stackoverflow.com/questions/29125019/get-margin-line-locations-mgp-in-user-coordinates

Examples

```
setup_plot <- function(log = "") {
  oldpar <- par(mar = c(2, 10, 2, 2), oma = rep(2, 4))
  plot.new()
  plot.window(xlim = c(1, 10), ylim = c(1, 10), log = log)
  box(which = "plot", lwd = 2, col = "gray40")
  box(which = "figure", lwd = 2, col = "darkred")
  box(which = "outer", lwd = 2, col = "darkgreen")
  text(x = 0.5, y = 0.5,
       labels = "Plot Region",
       col = "gray40", font = 2)
  mtext(side = 3, text = "Figure region", line = 0.5, col = "darkred", font = 2)
  mtext(side = 3, text = "Device region", line = 2.5, col = "darkgreen", font = 2)
  for (i in 0:9) {
    mtext(side = 2, col = "darkred", text = paste0("Line", i), line = i)
  }
  par(oldpar)
}
# And here are a couple of examples, applied to your setup_plot with mar=c(5, 5, 5, 5):
setup_plot()
axis(1, line=5)
axis(2, line=5)
abline(h=line2user(0:4, 1), lty=3, xpd=TRUE)
abline(v=line2user(0:4, 2), lty=3, xpd=TRUE)
abline(h=line2user(0:4, 3), lty=3, xpd=TRUE)
abline(v=line2user(0:4, 4), lty=3, xpd=TRUE)

setup_plot(log='x')
axis(1, line=5)
axis(2, line=5)
abline(h=line2user(0:4, 1), lty=3, xpd=TRUE)
abline(v=line2user(0:4, 2), lty=3, xpd=TRUE)
abline(h=line2user(0:4, 3), lty=3, xpd=TRUE)
abline(v=line2user(0:4, 4), lty=3, xpd=TRUE)


setup_plot(log='y')
axis(1, line=5)
axis(2, line=5)
abline(h=line2user(0:4, 1), lty=3, xpd=TRUE)
abline(v=line2user(0:4, 2), lty=3, xpd=TRUE)
abline(h=line2user(0:4, 3), lty=3, xpd=TRUE)
abline(v=line2user(0:4, 4), lty=3, xpd=TRUE)

setup_plot(log='xy')
axis(1, line=5)
axis(2, line=5)
abline(h=line2user(0:4, 1), lty=3, xpd=TRUE)
abline(v=line2user(0:4, 2), lty=3, xpd=TRUE)
abline(h=line2user(0:4, 3), lty=3, xpd=TRUE)
abline(v=line2user(0:4, 4), lty=3, xpd=TRUE)
```

loadOneName *load an object from .RDATA file*

Description

load an object from .RDATA file

Usage

```
loadOneName(objName, file, envir = parent.frame(), assign.on.exit = TRUE)
```

Arguments

objName	the name of the object
file	the file path
envir	environment; default is calling environment: parent.frame
assign.on.exit	default is TRUE

Value

a new environment

logit *logit function*

Description

logit function

Usage

```
logit(p)
```

Arguments

p	Probability between 0 and 1

Value

A real number

Examples

```
logit(0.5)
```

logOR *calculate pairwise log odds ratios*

Description

Case at upper triangle; control at lower triangle

Usage

```
logOR(MBS.case, MBS.ctrl)
```

Arguments

MBS.case	Case Bronze-Standard (BrS) data; rows for case subjects; columns contain JBrS measurements
MBS.ctrl	Control Bronze-Standard (BrS) data; rows for control subjects; columns contain JBrS measurements

Value

a list of two elements: logOR (JBrS by JBrS matrix of log odds ratios for each pair among JBrS measurements) and logOR.se (same dimension as logOR, but representing the standard errors of the corresponding estimated log odds ratios in logOR).

logsumexp *log sum exp trick*

Description

log sum exp trick

Usage

```
logsumexp(x)
```

Arguments

x	a vector of numbers

Value

a numeric value

Examples

```
logsumexp(c(-20,-30))
```

lookup_quality *Get position to store in data_nplcm$Mobs:*

Description

Get position to store in data_nplcm$Mobs:

Usage

```
lookup_quality(quality_nm)
```

Arguments

quality_nm	names of quality: can be "BrS", "SS" or "GS"

Details

also works for a vector

Value

position of the quality name: "BrS"-1; "SS"-2; "GS"-3.

See Also

extract_data_raw()

make_filename *Create new file name*

Description

Create new file name

Usage

```
make_filename(parameter_names, parameter_vals, format)
```

Arguments

parameter_names	The parameters that distinguish this folder's scenario
parameter_vals	The actual parameter values
format	The suffix ".XXX" in the end to specify the file format

Value

A string for file name

Examples

```
make_filename(c("theta","alpha"),c(0.9,2),"csv")
```

`make_foldername` *Create new folder name*

Description

Create new folder name

Usage

```
make_foldername(parent_path, parameter_names, parameter_vals, sep = "/")
```

Arguments

`parent_path`	The parent directory where to put the new folder
`parameter_names`	The parameters that distinguish this folder's scenario
`parameter_vals`	The actual parameter values
`sep`	file name separator - default to `"/"` for OSX; `"\\"` for Windows.

Value

A string for folder name

Examples

```
make_foldername("/user",c("theta","alpha","beta"),c(1,2,3))
```

make_list — *Takes any number of R objects as arguments and returns a list whose names are derived from the names of the R objects.*

Description

Roger Peng's listlabeling challenge from http://simplystatistics.tumblr.com/post/11988685443/computing-on-the-language. Code copied from https://gist.github.com/ajdamico/1329117/0134148987859856fcecbe4446cfd37e500e4272

Usage

```
make_list(...)
```

Arguments

...	any R objects

Value

a list as described above

Examples

```
#create three example variables for a list
x <- 1
y <- 2
z <- "hello"
#display the results
make_list( x , y , z )
```

make_meas_object — *Make measurement slice*

Description

Wrap the information about a particular type of measurement, e.g., NPPCR. NB: add example! copy some from the vignette file.

Usage

```
make_meas_object(patho, specimen, test, quality, cause_list, sep_char = "_")
```

Arguments

patho	A vector of pathogen names
specimen	Specimen name
test	Test name
quality	Quality category: any of "BrS", "SS" or "GS".
cause_list	The vector of potential latent status
sep_char	a character string that separate the pathogen names and the specimen-test pair; Default to "_"

Value

A list with measurement information

- quality same as argument
- patho same as argument
- name_in_data the names used in the raw data to locate these measurements
- template a mapping from patho to cause_list. NROW = length(cause_list)+1; NCOL = length(patho). This value is crucial in model fitting to determine which measurements are informative of a particular category of latent status.
- specimen same as argument
- test same as argument
- nm_spec_test paste specimen and test together

See Also

make_template()

Examples

```
make_meas_object(
patho = c("A","B","C","D","E","F"),
specimen = "MBS",
test = "1",
quality = "BrS",
cause_list = c("A","B","C","D","E"))
```

make_numbered_list *Make a list with numbered names*

Description

To collect multiple measurements within the same category, e.g., bronze-standard.

Usage

```
make_numbered_list(...)
```

Arguments

...	any R object

Value

a list with names numbered

make_template *make a mapping template for model fitting*

Description

`make_template` creates a mapping matrix (binary values). Each pathogen in a measurement slice (e.g., nasal-pharyngeal PCR test) is mapped to inform one category of latent status. All the possible categories (e.g., causes of pneumonia) remain the same regardless of the measurement slice used (e.g., NPPCR or BCX).

Usage

```
make_template(patho, cause_list)
```

Arguments

`patho`	A vector of pathogen names for a particular measurement slice. `patho` must be a substring of some elements in `cause_list`, e.g., "PNEU" is a substring of "PNEU_VT13". Also see Examples for this function.
`cause_list`	A vector of characters; Potential categories of latent statuses.

Details

The first argument has to be character substrings from the second argument. For example, the two arguments can respectively be "A" and "A_1", or "A" and "A+B".The second argument can have character strings not matched in the first argument. If so, it means some causes of diseases are not directly measured in the current measurement slice. For each element of patho, the function matches from the start of the strings of cause_list. Therefore, make sure that latent statuses from the same family (e.g., "PNEU_VT13" and "PNEU_NOVT13") need to start with the same family name (e.g., "PNEU") followed by subcategories (e.g., "_VT13" and "_NOVT13").

Value

a mapping from patho to cause_list. NROW = length(cause_list)+1; NCOL = length(patho). This value is crucial in model fitting to determine which measurements are informative of a particular category of latent status.

Examples

```
cause_list <- c("HINF","PNEU_VT13","PNEU_NOVT13","SAUR","HMPV_A_B","FLU_A",
"PARA_1","PARA_3","PARA_4","PV_EV","RHINO","RSV", "ENTRB","TB")

patho_BrS_NPPCR <- c("HINF","PNEU","SAUR","HMPV_A_B","FLU_A","PARA_1",
"PARA_3","PARA_4","PV_EV","RHINO","RSV")
make_template(patho_BrS_NPPCR,cause_list)


 cause = c("A","B1","B2","C","A+C","B+C")
 patho = c("A","B","C")
 make_template(patho,cause)

 cause = c("A","B1","B2","C","A+C","B+C","other")
 patho = c("A","B","C")
 make_template(patho,cause)


 cause = c("A","B1","B2","X_B","Y_B","C","A+C","B+C","other")
 patho = c("A","B","C","X_B","Y_B")
 make_template(patho,cause)
```

---

marg_H *Shannon entropy for binary data*

Description

Shannon entropy for binary data

Usage

```
marg_H(m_px)
```

Arguments

m_px	a number between 0 and 1

Value

a non-negative number

Examples

```
marg_H(0.1)
```

---

match_cause *Match latent causes that might have the same combo but different specifications*

Description

@details In our cause_list, "A+B" represents the same cause as "B+A". It is used for plotting side-by-side posterior sample comparisons

Usage

```
match_cause(pattern, vec)
```

Arguments

pattern	a vector of latent cause names, e.g., from a particular fit
vec	a vector of latent cause names, e.g., usually a union of cause names from several model fits. Usually, it is also the display order that one wants to show.

Value

A vector of length `length(vec)`; `NA` means no pattern matches vec; 1 at position 10 means the first element of `pattern` matches the 10th element of `vec`.

Examples

```
pattern <- c("X+Y","A+Z","C")
vec     <- c(LETTERS[1:26],"Y+Z","Y+X","Z+A")
match_cause(pattern,vec)
```

merge_lists *For a list of many sublists each of which has matrices as its member, we combine across the many sublists to produce a final list*

Description

For a list of many sublists each of which has matrices as its member, we combine across the many sublists to produce a final list

Usage

```
merge_lists(list_of_lists)
```

Arguments

list_of_lists	a list of sublists

Value

a list after merge

See Also

Other data operation functions: combine_data_nplcm(), subset_data_nplcm_by_index()

Examples

```
DT1 = list(A=1:3,B=letters[1:3])
DT2 = list(A=4:5,B=letters[4:5])
DT3 = list(A=1:4,B=letters[1:4])
DT4 = list(A=4:7,B=letters[4:7])
l = list(DT1,DT2);names(l) <- c("haha","hihi")
l2 = list(DT3,DT4);names(l2) <- c("haha","hihi")
listoflists <- list(l,l2);names(listoflists) <- c("dude1","dude2")
listoflists
merge_lists(listoflists)
```

my_reorder *Reorder the measurement dimensions to match the order for display*

Description

Reorder the measurement dimensions to match the order for display

Usage

```
my_reorder(disp_order, raw_nm)
```

Arguments

disp_order	The vector of names to be displayed (order matters)
raw_nm	The vector of names from raw measurements (order matters)

Value

A permuted vector from 1 to length(raw_nm). For example, if its first element is 3, it means that the 3rd pathogen in raw_nm should be arranged to the first in the raw measurements.

Examples

```
disp_order <- c("B","E","D","C","F","A")
raw_nm <- c("C","A","E")
my_reorder(disp_order,raw_nm)
```

NA2dot *convert 'NA' to '.'*

Description

convert 'NA' to '.'

Usage

```
NA2dot(s)
```

Arguments

s	A string of characters that may contain "NA"

Value

A string of characters without 'NA'

nplcm *Fit nested partially-latent class models (highest-level wrapper function)*

Description

Uses JAGS (OSX or Windows) operating system for Bayesian posterior inference (see README file for an instruction to install JAGS). If running JAGS on windows, please go to control panel to add the directory to JAGS into ENVIRONMENTAL VARIABLE.

Usage

```
nplcm(data_nplcm, model_options, mcmc_options)
```

Arguments

data_nplcm — Cases are on top of controls in the rows of diagnostic test results and the covariate matrix. This is assumed by baker to automatically write model files (.bug).

- Mobs A list of measurements of distinct qualities (Bronze-, Silver, and Gold-Standard: MBS,MSS,MGS). The elements of the list should include MBS, MSS, and MGS. If any of the component is not available, please specify it as, e.g., MGS=NULL (effectively deleting MGS from Mobs).
  - MBS a list of data frame of bronze-standard (BrS) measurements. For each data frame (referred to as a 'slice'), rows are subjects, columns are causative agents (e.g., pathogen species). We use list here to accommodate the possibility of multiple sets of BrS data. They have imperfect sensitivity/specificity (e.g. nasopharyngeal polymerase chain reaction - NPPCR).
  - MSS a list of data frame of silver-standard (SS) measurements. Rows are subjects, columns are causative agents measured in specimen (e.g. blood culture). These measurements have perfect specificity but imperfect sensitivity.
  - MGS a list of data frame of gold-standard (GS) measurements. Rows are subject, columns are measured causative agents These measurements have perfect sensitivity and specificity.
- Y Vector of disease status: 1 for case, 0 for control.
- X Covariate matrix. A subset of columns are primary covariates in cause-specific- case-fraction (CSCF) functions and hence must be available for cases, and another subset are covariates that are available in the cases and the controls. The two sets of covariates may be identical, overlapping or completely different. In general, this is not the design matrix for regression models, because for enrollment date in a study which may have non-linear effect, basis expansion is often needed for approximation.

model_options — A list of model options: likelihood and prior.

use_measurements A vector of characters strings; can be one or more from "BrS", "SS", "GS".

likelihood **cause_list** The vector of causes (NB: specify);

k_subclass The number of nested subclasses in each disease class (one of case classes or the control class; the same k_subclass is assumed for each class) and each slice of BrS measurements. 1 for conditional independence; larger than 1 for conditional dependence. It is only available for BrS measurements. It is a vector of length equal to the number of slices of BrS measurements;

Eti_formula Formula for etiology regressions. You can use s_date_Eti() to specify the design matrix for R format enrollment date; it will produce natural cubic spline basis. Specify ~ 1 if no regression is intended.

FPR_formula formula for false positive rates (FPR) regressions; see formula(). You can use s_date_FPR() to specify part of the design matrix for R format enrollment date; it will produce penalized-spline basis (based on B-splines). Specify ~ 1 if no regression is intended. (NB: If effect="fixed", dm_Rdate_FPR() will just specify a design matrix with appropriately standardized dates.)

prior **Eti_prior** Description of etiology prior (e.g., overall_uniform - all hyperparameters are 1; or 0_1 - all hyperparameters are 0.1);

TPR_prior Description of priors for the measurements (e.g., informative vs non-informative). Its length should be the same as use_measurements above. Please see examples for how to specify. The package can also handle multiple slices of BrS, SS data, so separate specification of the TPR priors are needed.

mcmc_options A list of Markov chain Monte Carlo (MCMC) options.

- debugstatus Logical - whether to pause WinBUGS after it finishes model fitting; (NB: is this obsolete? Test.)
- n.chains Number of MCMC chains;
- n.burnin Number of burn-in iterations;
- n.thin To keep every other n.thin samples after burn-in period;
- individual.pred TRUE to perform individual prediction (Icat variables in the .bug file); FALSE otherwise;
- ppd TRUE to simulate new data (XXX.new variables in the .bug file) from the posterior predictive distribution (ppd); FALSE otherwise;
- get.pEti TRUE for getting posterior samples of individual etiologic fractions; FALSE otherwise. For non-regression, or regression models with all discrete predictors, by default this is TRUE, so no need to specify this entry. It is only relevant for regression models with non-discrete covariates. Because individuals have distinct CSCFs at their specific covariate values, it's easier to just store the posterior samples of the regression coefficients and reconstruct the pies afterwards, rather than storing them through JAGS.
- result.folder Path to folder storing the results;
- bugsmodel.dir Path to .bug model files;
- jags.dir Path to where JAGS is installed; if NULL, this will be set to jags.dir="".

Value

A JAGS output result, fitted by function R2jags::jags2() from R2jags. It is an object of class nplcm and bugs. Current implemented models follow the hierarchy below:

- no regression: Fitted by at low level by nplcm_fit_NoReg
- regression: Given disease class (control or a class of cases with the same subset of causative agents):
  - local independence model for BrS measures: Fitted at lower level by
    - nplcm_fit_Reg_NoNest deals with the setting with two sets of covariates, one for CSCF regression and the other for FPR regression. The two sets of covariates may be identical, overlapping or non-overlapping. This function is called when there exists one or more than one discrete covariate among the union of the two covariate sets. The method implemented by this function directly lets FPR depend upon covariates. This is different from Wu and Chen (2021), which let the subclass weights depend upon covariates. We implemented this function for methods comparison.
    - nplcm_fit_Reg_discrete_predictor_NoNest deals with the setting with all discrete covariates for FPRs and CSCFs. The strata defined by the two sets of covariates need not be identical, e.g., as a result of distinct sets of covariates. Again, this is directly to let FPR be stratified by covariates, hence different from Wu and Chen (2020+) We implemented this function for methods comparison.
  - local dependence model for BrS measures: Fitted at lower level by nplcm_fit_Reg_Nest: This is the method introduced in Wu and Chen (2021): CSCF regression + case/control subclass weight regression. It does not provide a specialized function for the setting with all discrete covariates.

Examples

```
data(data_nplcm_noreg)
cause_list <- LETTERS[1:6]
J.BrS      <- 6
model_options_no_reg <- list(
  likelihood   = list(
    cause_list = cause_list,
    k_subclass = 2,
    Eti_formula = ~-1, # no covariate for the etiology regression
    FPR_formula = list(
      MBS1 =   ~-1)    # no covariate for the subclass weight regression
  ),
  use_measurements = c("BrS"),
  # use bronze-standard data only for model estimation.
  prior= list(
    Eti_prior = overall_uniform(1,cause_list),
    # Dirichlet(1,...,1) prior for the etiology.
    TPR_prior  = list(BrS = list(
      info  = "informative", # informative prior for TPRs
      input = "match_range",
      # specify the informative prior for TPRs by specifying a plausible range.
      val = list(MBS1 = list(up =  list(rep(0.99,J.BrS)),
```

```
                              # upper ranges: matched to 97.5% quantile of a Beta prior
                              low = list(rep(0.55,J.BrS))))
      # lower ranges: matched to 2.5% quantile of a Beta prior
    )
    )
  )
)


set.seed(1)
# include stratification information in file name:
thedir    <- paste0(tempdir(),"_no_reg")

# create folders to store the model results
dir.create(thedir, showWarnings = FALSE)
result_folder_no_reg <- file.path(thedir,paste("results",collapse="_"))
thedir <- result_folder_no_reg
dir.create(thedir, showWarnings = FALSE)

# options for MCMC chains:
mcmc_options_no_reg <- list(
  debugstatus = TRUE,
  n.chains = 1,
  n.itermcmc = as.integer(200),
  n.burnin = as.integer(100),
  n.thin = 1,
  individual.pred = TRUE, # <- must set to TRUE! <------- NOTE!
  ppd = FALSE,
  result.folder = thedir,
  bugsmodel.dir = thedir
)

BrS_object_1 <- make_meas_object(patho = LETTERS[1:6],
                                 specimen = "MBS", test = "1",
                                 quality = "BrS", cause_list = cause_list)
clean_options <- list(BrS_objects = make_list(BrS_object_1))
# place the nplcm data and cleaning options into the results folder
dput(data_nplcm_noreg,file.path(thedir,"data_nplcm.txt"))
dput(clean_options, file.path(thedir, "data_clean_options.txt"))

rjags::load.module("glm")

nplcm_noreg <- nplcm(data_nplcm_noreg,model_options_no_reg,mcmc_options_no_reg)
```

---

nplcm_fit_NoReg *Fit nested partially-latent class model (low-level)*

Description

This function prepares data, specifies hyperparameters in priors (true positive rates and etiology fractions), initializes the posterior sampling chain, writes the model file (for JAGS or WinBUGS with slight differences in syntax), and fits the model. Features:

- no regression;
- no nested subclasses

Usage

```
nplcm_fit_NoReg(data_nplcm, model_options, mcmc_options)
```

Arguments

`data_nplcm` Cases are on top of controls in the rows of diagnostic test results and the covariate matrix. This is assumed by `baker` to automatically write model files (`.bug`).

- `Mobs` A list of measurements of distinct qualities (Bronze-, Silver, and Gold-Standard: `MBS`,`MSS`,`MGS`). The elements of the list should include `MBS`, `MSS`, and `MGS`. If any of the component is not available, please specify it as, e.g., `MGS=NULL` (effectively deleting `MGS` from `Mobs`).
  - `MBS` a list of data frame of bronze-standard (BrS) measurements. For each data frame (referred to as a 'slice'), rows are subjects, columns are causative agents (e.g., pathogen species). We use `list` here to accommodate the possibility of multiple sets of BrS data. They have imperfect sensitivity/specificity (e.g. nasopharyngeal polymerase chain reaction - NPPCR).
  - `MSS` a list of data frame of silver-standard (SS) measurements. Rows are subjects, columns are causative agents measured in specimen (e.g. blood culture). These measurements have perfect specificity but imperfect sensitivity.
  - `MGS` a list of data frame of gold-standard (GS) measurements. Rows are subject, columns are measured causative agents These measurements have perfect sensitivity and specificity.
- `Y` Vector of disease status: `1` for case, `0` for control.
- `X` Covariate matrix. A subset of columns are primary covariates in cause-specific- case-fraction (CSCF) functions and hence must be available for cases, and another subset are covariates that are available in the cases and the controls. The two sets of covariates may be identical, overlapping or completely different. In general, this is not the design matrix for regression models, because for enrollment date in a study which may have non-linear effect, basis expansion is often needed for approximation.

`model_options` A list of model options: likelihood and prior.

`use_measurements` A vector of characters strings; can be one or more from `"BrS"`, `"SS"`, `"GS"`.

`likelihood` **cause_list** The vector of causes (NB: specify);

k_subclass The number of nested subclasses in each disease class (one of case classes or the control class; the same `k_subclass` is assumed for each class) and each slice of BrS measurements. `1` for conditional independence; larger than `1` for conditional dependence. It is only available for BrS measurements. It is a vector of length equal to the number of slices of BrS measurements;

Eti_formula Formula for etiology regressions. You can use `s_date_Eti()` to specify the design matrix for `R` format enrollment date; it will produce natural cubic spline basis. Specify `~ 1` if no regression is intended.

FPR_formula formula for false positive rates (FPR) regressions; see `formula()`. You can use `s_date_FPR()` to specify part of the design matrix for `R` format enrollment date; it will produce penalized-spline basis (based on B-splines). Specify `~ 1` if no regression is intended. (NB: If `effect="fixed"`, `dm_Rdate_FPR()` will just specify a design matrix with appropriately standardized dates.)

`prior` **Eti_prior** Description of etiology prior (e.g., `overall_uniform` - all hyperparameters are `1`; or `0_1` - all hyperparameters are `0.1`);

TPR_prior Description of priors for the measurements (e.g., informative vs non-informative). Its length should be the same as `use_measurements` above. Please see examples for how to specify. The package can also handle multiple slices of BrS, SS data, so separate specification of the TPR priors are needed.

`mcmc_options` A list of Markov chain Monte Carlo (MCMC) options.

- `debugstatus` Logical - whether to pause WinBUGS after it finishes model fitting; (NB: is this obsolete? Test.)
- `n.chains` Number of MCMC chains;
- `n.burnin` Number of burn-in iterations;
- `n.thin` To keep every other `n.thin` samples after burn-in period;
- `individual.pred` `TRUE` to perform individual prediction (`Icat` variables in the `.bug` file); `FALSE` otherwise;
- `ppd` `TRUE` to simulate new data (`XXX.new` variables in the `.bug` file) from the posterior predictive distribution (ppd); `FALSE` otherwise;
- `get.pEti` `TRUE` for getting posterior samples of individual etiologic fractions; `FALSE` otherwise. For non-regression, or regression models with all discrete predictors, by default this is `TRUE`, so no need to specify this entry. It is only relevant for regression models with non-discrete covariates. Because individuals have distinct CSCFs at their specific covariate values, it's easier to just store the posterior samples of the regression coefficients and reconstruct the pies afterwards, rather than storing them through `JAGS`.
- `result.folder` Path to folder storing the results;
- `bugsmodel.dir` Path to `.bug` model files;
- `jags.dir` Path to where JAGS is installed; if `NULL`, this will be set to `jags.dir=""`.

Value

BUGS fit results.

See Also

write_model_NoReg for constructing .bug model file; This function then put it in the folder mcmc_options$bugsmodel.dir.

Other model fitting functions: nplcm_fit_Reg_Nest(), nplcm_fit_Reg_NoNest(), nplcm_fit_Reg_discrete_predictor

---

`nplcm_fit_Reg_discrete_predictor_NoNest` *Fit nested partially-latent class model with regression (low-level)*

Description

Fit nested partially-latent class model with regression (low-level)

Usage

```
nplcm_fit_Reg_discrete_predictor_NoNest(
  data_nplcm,
  model_options,
  mcmc_options
)
```

Arguments

`data_nplcm` Cases are on top of controls in the rows of diagnostic test results and the covariate matrix. This is assumed by baker to automatically write model files (.bug).

- Mobs A list of measurements of distinct qualities (Bronze-, Silver, and Gold-Standard: MBS,MSS,MGS). The elements of the list should include MBS, MSS, and MGS. If any of the component is not available, please specify it as, e.g., MGS=NULL (effectively deleting MGS from Mobs).
  - MBS a list of data frame of bronze-standard (BrS) measurements. For each data frame (referred to as a 'slice'), rows are subjects, columns are causative agents (e.g., pathogen species). We use list here to accommodate the possibility of multiple sets of BrS data. They have imperfect sensitivity/specificity (e.g. nasopharyngeal polymerase chain reaction - NPPCR).
  - MSS a list of data frame of silver-standard (SS) measurements. Rows are subjects, columns are causative agents measured in specimen (e.g. blood culture). These measurements have perfect specificity but imperfect sensitivity.
  - MGS a list of data frame of gold-standard (GS) measurements. Rows are subject, columns are measured causative agents These measurements have perfect sensitivity and specificity.
- Y Vector of disease status: 1 for case, 0 for control.

- `X` Covariate matrix. A subset of columns are primary covariates in cause-specific- case-fraction (CSCF) functions and hence must be available for cases, and another subset are covariates that are available in the cases and the controls. The two sets of covariates may be identical, overlapping or completely different. In general, this is not the design matrix for regression models, because for enrollment date in a study which may have non-linear effect, basis expansion is often needed for approximation.

`model_options` A list of model options: likelihood and prior.

`use_measurements` A vector of characters strings; can be one or more from `"BrS"`, `"SS"`, `"GS"`.

`likelihood` **cause_list** The vector of causes (NB: specify);

k_subclass The number of nested subclasses in each disease class (one of case classes or the control class; the same `k_subclass` is assumed for each class) and each slice of BrS measurements. `1` for conditional independence; larger than `1` for conditional dependence. It is only available for BrS measurements. It is a vector of length equal to the number of slices of BrS measurements;

Eti_formula Formula for etiology regressions. You can use `s_date_Eti()` to specify the design matrix for `R` format enrollment date; it will produce natural cubic spline basis. Specify `~ 1` if no regression is intended.

FPR_formula formula for false positive rates (FPR) regressions; see `formula()`. You can use `s_date_FPR()` to specify part of the design matrix for `R` format enrollment date; it will produce penalized-spline basis (based on B-splines). Specify `~ 1` if no regression is intended. (NB: If `effect="fixed"`, `dm_Rdate_FPR()` will just specify a design matrix with appropriately standardized dates.)

`prior` **Eti_prior** Description of etiology prior (e.g., `overall_uniform` - all hyperparameters are `1`; or `0_1` - all hyperparameters are `0.1`);

TPR_prior Description of priors for the measurements (e.g., informative vs non-informative). Its length should be the same as `use_measurements` above. Please see examples for how to specify. The package can also handle multiple slices of BrS, SS data, so separate specification of the TPR priors are needed.

`mcmc_options` A list of Markov chain Monte Carlo (MCMC) options.

- `debugstatus` Logical - whether to pause WinBUGS after it finishes model fitting; (NB: is this obsolete? Test.)
- `n.chains` Number of MCMC chains;
- `n.burnin` Number of burn-in iterations;
- `n.thin` To keep every other `n.thin` samples after burn-in period;
- `individual.pred` `TRUE` to perform individual prediction (`Icat` variables in the `.bug` file); `FALSE` otherwise;
- `ppd` `TRUE` to simulate new data (`XXX.new` variables in the `.bug` file) from the posterior predictive distribution (ppd); `FALSE` otherwise;
- `get.pEti` `TRUE` for getting posterior samples of individual etiologic fractions; `FALSE` otherwise. For non-regression, or regression models with all discrete predictors, by default this is `TRUE`, so no need to specify this entry.

It is only relevant for regression models with non-discrete covariates. Because individuals have distinct CSCFs at their specific covariate values, it's easier to just store the posterior samples of the regression coefficients and reconstruct the pies afterwards, rather than storing them through JAGS.

- result.folder Path to folder storing the results;
- bugsmodel.dir Path to .bug model files;
- jags.dir Path to where JAGS is installed; if NULL, this will be set to jags.dir="".

Details

This function prepares data, specifies hyperparameters in priors (true positive rates and etiology fractions), initializes the posterior sampling chain, writes the model file (for JAGS or WinBUGS with slight differences in syntax), and fits the model. Features:

- regression;
- no nested subclasses, i.e. conditional independence of multivariate measurements given disease class and covariates;
- multiple BrS + multiple SS.

If running JAGS on windows, please go to control panel to add the directory to jags into ENVIRONMENTAL VARIABLE!

Value

BUGS fit results.

See Also

write_model_NoReg for automatically generate .bug model file; This present function store it in location: mcmc_options$bugsmodel.dir.

Other model fitting functions: nplcm_fit_NoReg(), nplcm_fit_Reg_Nest(), nplcm_fit_Reg_NoNest()

---

nplcm_fit_Reg_Nest *Fit nested partially-latent class model with regression (low-level)*

Description

Called by nplcm() upon being assigned to this nested regression by assign_model()

Usage

```
nplcm_fit_Reg_Nest(data_nplcm, model_options, mcmc_options)
```

Arguments

`data_nplcm` Cases are on top of controls in the rows of diagnostic test results and the covariate matrix. This is assumed by `baker` to automatically write model files (`.bug`).

- `Mobs` A list of measurements of distinct qualities (Bronze-, Silver, and Gold-Standard: `MBS`,`MSS`,`MGS`). The elements of the list should include `MBS`, `MSS`, and `MGS`. If any of the component is not available, please specify it as, e.g., `MGS=NULL` (effectively deleting `MGS` from `Mobs`).
  - `MBS` a list of data frame of bronze-standard (BrS) measurements. For each data frame (referred to as a 'slice'), rows are subjects, columns are causative agents (e.g., pathogen species). We use `list` here to accommodate the possibility of multiple sets of BrS data. They have imperfect sensitivity/specificity (e.g. nasopharyngeal polymerase chain reaction - NPPCR).
  - `MSS` a list of data frame of silver-standard (SS) measurements. Rows are subjects, columns are causative agents measured in specimen (e.g. blood culture). These measurements have perfect specificity but imperfect sensitivity.
  - `MGS` a list of data frame of gold-standard (GS) measurements. Rows are subject, columns are measured causative agents These measurements have perfect sensitivity and specificity.
- `Y` Vector of disease status: `1` for case, `0` for control.
- `X` Covariate matrix. A subset of columns are primary covariates in cause-specific- case-fraction (CSCF) functions and hence must be available for cases, and another subset are covariates that are available in the cases and the controls. The two sets of covariates may be identical, overlapping or completely different. In general, this is not the design matrix for regression models, because for enrollment date in a study which may have non-linear effect, basis expansion is often needed for approximation.

`model_options` A list of model options: likelihood and prior.

`use_measurements` A vector of characters strings; can be one or more from `"BrS"`, `"SS"`, `"GS"`.

`likelihood` **cause_list** The vector of causes (NB: specify);

k_subclass The number of nested subclasses in each disease class (one of case classes or the control class; the same `k_subclass` is assumed for each class) and each slice of BrS measurements. `1` for conditional independence; larger than `1` for conditional dependence. It is only available for BrS measurements. It is a vector of length equal to the number of slices of BrS measurements;

Eti_formula Formula for etiology regressions. You can use `s_date_Eti()` to specify the design matrix for `R` format enrollment date; it will produce natural cubic spline basis. Specify `~ 1` if no regression is intended.

FPR_formula formula for false positive rates (FPR) regressions; see `formula()`. You can use `s_date_FPR()` to specify part of the design matrix for `R` format enrollment date; it will produce penalized-spline basis (based on B-splines). Specify `~ 1` if no regression is intended. (NB: If `effect="fixed"`,

dm_Rdate_FPR() will just specify a design matrix with appropriately standardized dates.)

prior **Eti_prior** Description of etiology prior (e.g., overall_uniform - all hyperparameters are 1; or 0_1 - all hyperparameters are 0.1);

TPR_prior Description of priors for the measurements (e.g., informative vs non-informative). Its length should be the same as use_measurements above. Please see examples for how to specify. The package can also handle multiple slices of BrS, SS data, so separate specification of the TPR priors are needed.

mcmc_options A list of Markov chain Monte Carlo (MCMC) options.

- debugstatus Logical - whether to pause WinBUGS after it finishes model fitting; (NB: is this obsolete? Test.)
- n.chains Number of MCMC chains;
- n.burnin Number of burn-in iterations;
- n.thin To keep every other n.thin samples after burn-in period;
- individual.pred TRUE to perform individual prediction (Icat variables in the .bug file); FALSE otherwise;
- ppd TRUE to simulate new data (XXX.new variables in the .bug file) from the posterior predictive distribution (ppd); FALSE otherwise;
- get.pEti TRUE for getting posterior samples of individual etiologic fractions; FALSE otherwise. For non-regression, or regression models with all discrete predictors, by default this is TRUE, so no need to specify this entry. It is only relevant for regression models with non-discrete covariates. Because individuals have distinct CSCFs at their specific covariate values, it's easier to just store the posterior samples of the regression coefficients and reconstruct the pies afterwards, rather than storing them through JAGS.
- result.folder Path to folder storing the results;
- bugsmodel.dir Path to .bug model files;
- jags.dir Path to where JAGS is installed; if NULL, this will be set to jags.dir="".

Details

This function prepares data, specifies hyperparameters in priors (true positive rates and etiology fractions), initializes the posterior sampling chain, writes the model file (for JAGS), and fits the model. Features:

- regression (not all discrete covariates);
- nested subclasses, i.e. conditional dependence of multivariate measurements given disease class and covariates;
- multiple BrS + multiple SS.

Value

BUGS fit results.

See Also

write_model_Reg_Nest for constructing .bug model file; This function then put it in the folder mcmc_options$bugsmodel.dir.

Other model fitting functions: nplcm_fit_NoReg(), nplcm_fit_Reg_NoNest(), nplcm_fit_Reg_discrete_predictor_N

---

nplcm_fit_Reg_NoNest *Fit nested partially-latent class model with regression (low-level)*

Description

Fit nested partially-latent class model with regression (low-level)

Usage

```
nplcm_fit_Reg_NoNest(data_nplcm, model_options, mcmc_options)
```

Arguments

data_nplcm Cases are on top of controls in the rows of diagnostic test results and the covariate matrix. This is assumed by baker to automatically write model files (.bug).

- Mobs A list of measurements of distinct qualities (Bronze-, Silver, and Gold-Standard: MBS,MSS,MGS). The elements of the list should include MBS, MSS, and MGS. If any of the component is not available, please specify it as, e.g., MGS=NULL (effectively deleting MGS from Mobs).
  - MBS a list of data frame of bronze-standard (BrS) measurements. For each data frame (referred to as a 'slice'), rows are subjects, columns are causative agents (e.g., pathogen species). We use list here to accommodate the possibility of multiple sets of BrS data. They have imperfect sensitivity/specificity (e.g. nasopharyngeal polymerase chain reaction - NPPCR).
  - MSS a list of data frame of silver-standard (SS) measurements. Rows are subjects, columns are causative agents measured in specimen (e.g. blood culture). These measurements have perfect specificity but imperfect sensitivity.
  - MGS a list of data frame of gold-standard (GS) measurements. Rows are subject, columns are measured causative agents These measurements have perfect sensitivity and specificity.
- Y Vector of disease status: 1 for case, 0 for control.
- X Covariate matrix. A subset of columns are primary covariates in cause-specific- case-fraction (CSCF) functions and hence must be available for cases, and another subset are covariates that are available in the cases and the controls. The two sets of covariates may be identical, overlapping or completely different. In general, this is not the design matrix for regression models, because for enrollment date in a study which may have non-linear effect, basis expansion is often needed for approximation.

`model_options` A list of model options: likelihood and prior.

`use_measurements` A vector of characters strings; can be one or more from `"BrS"`, `"SS"`, `"GS"`.

`likelihood` **cause_list** The vector of causes (NB: specify);

k_subclass The number of nested subclasses in each disease class (one of case classes or the control class; the same `k_subclass` is assumed for each class) and each slice of BrS measurements. `1` for conditional independence; larger than `1` for conditional dependence. It is only available for BrS measurements. It is a vector of length equal to the number of slices of BrS measurements;

Eti_formula Formula for etiology regressions. You can use `s_date_Eti()` to specify the design matrix for `R` format enrollment date; it will produce natural cubic spline basis. Specify `~ 1` if no regression is intended.

FPR_formula formula for false positive rates (FPR) regressions; see `formula()`. You can use `s_date_FPR()` to specify part of the design matrix for `R` format enrollment date; it will produce penalized-spline basis (based on B-splines). Specify `~ 1` if no regression is intended. (NB: If `effect="fixed"`, `dm_Rdate_FPR()` will just specify a design matrix with appropriately standardized dates.)

`prior` **Eti_prior** Description of etiology prior (e.g., `overall_uniform` - all hyperparameters are `1`; or `0_1` - all hyperparameters are `0.1`);

TPR_prior Description of priors for the measurements (e.g., informative vs non-informative). Its length should be the same as `use_measurements` above. Please see examples for how to specify. The package can also handle multiple slices of BrS, SS data, so separate specification of the TPR priors are needed.

`mcmc_options` A list of Markov chain Monte Carlo (MCMC) options.

- `debugstatus` Logical - whether to pause WinBUGS after it finishes model fitting; (NB: is this obsolete? Test.)
- `n.chains` Number of MCMC chains;
- `n.burnin` Number of burn-in iterations;
- `n.thin` To keep every other `n.thin` samples after burn-in period;
- `individual.pred` `TRUE` to perform individual prediction (`Icat` variables in the `.bug` file); `FALSE` otherwise;
- `ppd` `TRUE` to simulate new data (`XXX.new` variables in the `.bug` file) from the posterior predictive distribution (ppd); `FALSE` otherwise;
- `get.pEti` `TRUE` for getting posterior samples of individual etiologic fractions; `FALSE` otherwise. For non-regression, or regression models with all discrete predictors, by default this is `TRUE`, so no need to specify this entry. It is only relevant for regression models with non-discrete covariates. Because individuals have distinct CSCFs at their specific covariate values, it's easier to just store the posterior samples of the regression coefficients and reconstruct the pies afterwards, rather than storing them through `JAGS`.
- `result.folder` Path to folder storing the results;
- `bugsmodel.dir` Path to `.bug` model files;
- `jags.dir` Path to where JAGS is installed; if `NULL`, this will be set to `jags.dir=""`.

Details

This function prepares data, specifies hyperparameters in priors (true positive rates and CSCFs), initializes the posterior sampling chain, writes the model file (for JAGS or WinBUGS with slight differences in syntax), and fits the model. Features:

- regression (not all discrete covariates);
- no nested subclasses, i.e. conditional independence of multivariate measurements given disease class and covariates;
- multiple BrS + multiple SS.

Value

BUGS fit results from `JAGS`.

See Also

write_model_NoReg for constructing `.bug` model file; This function then puts it in the folder `mcmc_options$bugsmodel.dir`.

Other model fitting functions: `nplcm_fit_NoReg()`, `nplcm_fit_Reg_Nest()`, `nplcm_fit_Reg_discrete_predictor_NoN`

---

nplcm_read_folder *Read data and other model information from a folder that stores model results.*

Description

Read data and other model information from a folder that stores model results.

Usage

```
nplcm_read_folder(DIR_NPLCM)
```

Arguments

`DIR_NPLCM`	File path to the folder containing posterior samples

Value

A list with data, options and posterior samples.

- `bugs.dat`
- `model_options`
- `clean_otions`
- `Nd`; `Nu`; `Y`; `Mobs`;
- `res_nplcm`.

Examples

```
data(data_nplcm_noreg)
cause_list <- LETTERS[1:6]
J.BrS      <- 6
model_options_no_reg <- list(
  likelihood   = list(
    cause_list = cause_list,
    k_subclass = 2,
    Eti_formula = ~-1, # no covariate for the etiology regression
    FPR_formula = list(
      MBS1 =   ~-1)    # no covariate for the subclass weight regression
  ),
  use_measurements = c("BrS"),
  # use bronze-standard data only for model estimation.
  prior= list(
    Eti_prior = overall_uniform(1,cause_list),
    # Dirichlet(1,...,1) prior for the etiology.
    TPR_prior  = list(BrS = list(
      info  = "informative", # informative prior for TPRs
      input = "match_range",
      # specify the informative prior for TPRs by specifying a plausible range.
      val = list(MBS1 = list(up =  list(rep(0.99,J.BrS)),
                             # upper ranges: matched to 97.5% quantile of a Beta prior
                             low = list(rep(0.55,J.BrS))))
      # lower ranges: matched to 2.5% quantile of a Beta prior
    )
    )
  )
)


set.seed(1)
# include stratification information in file name:
thedir    <- paste0(tempdir(),"_no_reg")

# create folders to store the model results
dir.create(thedir, showWarnings = FALSE)
result_folder_no_reg <- file.path(thedir,paste("results",collapse="_"))
thedir <- result_folder_no_reg
dir.create(thedir, showWarnings = FALSE)

# options for MCMC chains:
mcmc_options_no_reg <- list(
  debugstatus = TRUE,
  n.chains = 1,
  n.itermcmc = as.integer(200),
  n.burnin = as.integer(100),
  n.thin = 1,
  individual.pred = FALSE,
  ppd = TRUE,
  result.folder = thedir,
```

```
  bugsmodel.dir = thedir
)

BrS_object_1 <- make_meas_object(patho = LETTERS[1:6],
                                 specimen = "MBS", test = "1",
                                 quality = "BrS", cause_list = cause_list)
clean_options <- list(BrS_objects = make_list(BrS_object_1))
# place the nplcm data and cleaning options into the results folder
dput(data_nplcm_noreg,file.path(thedir,"data_nplcm.txt"))
dput(clean_options, file.path(thedir, "data_clean_options.txt"))

rjags::load.module("glm")

nplcm_noreg <- nplcm(data_nplcm_noreg,model_options_no_reg,mcmc_options_no_reg)

res <- nplcm_read_folder(nplcm_noreg$DIR_NPLCM)
```

`null_as_zero` *Convert* `NULL` *to zero.*

Description

`null_as_zero` make `NULL` to be zero.

Usage

```
null_as_zero(x)
```

Arguments

`x`	A number (usually a member of a list) that might be `NULL`

Value

A number

`order_post_eti` *order latent status by posterior mean*

Description

order latent status by posterior mean

Usage

```
order_post_eti(res_nplcm, model_options)
```

Arguments

res_nplcm	result from model fits
model_options	model specification

Value

a list with order (ord) and ordered posterior samples (by column)

---

overall_uniform *specify overall uniform (symmetric Dirichlet distribution) for etiology prior*

Description

specify overall uniform (symmetric Dirichlet distribution) for etiology prior

Usage

```
overall_uniform(alpha, cause_list)
```

Arguments

alpha	any positive number, usually 1.
cause_list	a list of latent status

Value

a vector of length length(cause_list)

See Also

Other prior specification functions: set_prior_tpr_BrS_NoNest(), set_prior_tpr_SS()

Examples

```
overall_uniform(1,c("A","B","C"))
```

parse_nplcm_reg *parse regression components (either false positive rate or etiology regression) for fitting npLCM; Only use this when formula is not* NULL.

Description

parse regression components (either false positive rate or etiology regression) for fitting npLCM; Only use this when formula is not NULL.

Usage

```
parse_nplcm_reg(form, data_nplcm, silent = TRUE)
```

Arguments

form	regression formula
data_nplcm	data object for nplcm(); may contain covariates X; must have case-control status Y.
silent	Default is TRUE for no message about covariates; FALSE otherwise.

Value

TRUE for doing regression; FALSE otherwise.

pathogen_category_perch *pathogens and their categories in PERCH study (virus or bacteria)*

Description

231 rows indicating bacteria, virus, fungi, or other categories.

Usage

```
data("pathogen_category_perch")
```

Format

A matrix of two columns

pathogen names of the pathogens

pathogen_type category of the pathogens, B for bacterium, V for virus, F for fungus, O for "not categorized"

Value

No returned value; just loading data into the working space.

pathogen_category_simulation *Hypothetical pathogens and their categories (virus or bacteria)*

Description

This is used in simulations where the pathogen names are from the alphabet, and we hope to plot etiologies grouped by virus or bacteria

Usage

```
data("pathogen_category_simulation")
```

Format

A matrix of two columns

pathogen names of the hypothetical pathogens, A-Z

pathogen_type category of the hypothetical pathogens, `B` for bacterium, `V` for virus, which are randomly assigned.

Value

No returned value; just loading data into the working space.

plot.nplcm `plot.nplcm` *plot the results from* `nplcm()`.

Description

`plot.nplcm` plot the results from `nplcm()`.

Usage

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

Arguments

`x`	Output from `nplcm()`.
`...`	Arguments passed to summary and printing methods.

Value

a figure

See Also

Other visualization functions: plot_BrS_panel(), plot_SS_panel(), plot_check_common_pattern(), plot_check_pairwise_SLORD(), plot_etiology_regression(), plot_etiology_strat(), plot_panels(), plot_pie_panel(), plot_subwt_regression()

`plot_BrS_panel` *Plot bronze-standard (BrS) panel*

Description

Plot bronze-standard (BrS) panel

Usage

```
plot_BrS_panel(
  slice,
  data_nplcm,
  model_options,
  clean_options,
  bugs.dat,
  res_nplcm,
  bg_color,
  select_latent = NULL,
  exact = TRUE,
  top_BrS = 1.3,
  cexval = 1,
  srtval = 0,
  prior_shape = "interval",
  silent = TRUE
)
```

Arguments

slice	the index of measurement slice for BrS.
data_nplcm	See nplcm()
model_options	See nplcm()
clean_options	See clean_perch_data()
bugs.dat	Data input for the model fitting.
res_nplcm	See nplcm_read_folder()
bg_color	A list with names "BrS", "SS", "pie" to specify background colors
select_latent	a vector of character strings representing latent status. It is used for just plotting a subset of latent status. For example, you can specify select_latent = "HINF" to plot all latent status information relevant to "HINF".

exact	Default is TRUE to use select_latent as exact names of causes. If you want to specify a name and plot all single or combo causes with that name, specify it to be FALSE.
top_BrS	Numerical value to specify the rightmost limit on the horizontal axis for the BrS panel.
cexval	Default is 1 - size of text of the BrS percentages.
srtval	Default is 0 - the direction of the text for the BrS percentages.
prior_shape	interval or boxplot - for how to represent prior/posteriors of the TPR/FPRs of measurements.
silent	Default is TRUE to not print any warning messages; FALSE otherwise.

Value

plotting function.

See Also

Other visualization functions: plot.nplcm(), plot_SS_panel(), plot_check_common_pattern(), plot_check_pairwise_SLORD(), plot_etiology_regression(), plot_etiology_strat(), plot_panels(), plot_pie_panel(), plot_subwt_regression()

---

plot_case_study *visualize the PERCH etiology regression with a continuous covariate*

Description

This function is specifically designed for PERCH data, e.g., (NB: dealing with NoA, multiple-pathogen causes, other continuous covariates? also there this function only plots the first slice - so generalization may be useful - give users an option to choose slice s; currently default to the first slice.)

Usage

```
plot_case_study(
  DIR_NPLCM,
  stratum_bool = stratum_bool,
  bugs.dat = NULL,
  slice = 1,
  RES_NPLCM = NULL,
  do_plot = TRUE,
  do_rug = FALSE,
  return_metric = TRUE
)
```

Arguments

DIR_NPLCM	File path to the folder containing posterior samples
stratum_bool	integer; for this function, indicates which strata to plot
bugs.dat	The posterior samples (loaded into the environment to save time) -> default is NULL
slice	integer; specifies which slice of bronze-standard data to visualize; Default to 1.
RES_NPLCM	pre-read res_nplcm; default to NULL.
do_plot	TRUE for plotting
do_rug	TRUE for plotting
return_metric	TRUE for showing overall mean etiology, quantiles, s.d., and if truth$Eti is supplied, coverage, bias, truth and integrated mean squared errors (IMSE).

Value

A figure of etiology regression curves and some marginal positive rate assessment of model fit; See example for the legends.

---

plot_check_common_pattern

Posterior predictive checking for the nested partially class models - frequent patterns in the BrS data. (for multiple folders)

Description

At each MCMC iteration, we generate a new data set based on the model and parameter values at that iteration. The sample size of the new data set equals that of the actual data set, i.e. the same number of cases and controls.

Usage

```
plot_check_common_pattern(
  DIR_list,
  slice_vec = rep(1, length(DIR_list)),
  n_pat = 10,
  dodge_val = 0.8
)
```

Arguments

DIR_list	The list of directory paths, each storing a model output.
slice_vec	Default are 1s, for the first slice of BrS data.
n_pat	Number of the most common BrS measurement pattern among cases and controls. Default is 10.
dodge_val	Default is 0.8; For width of boxplots.

Value

A figure of posterior predicted frequencies compared with the observed frequencies of the most common patterns for the BrS data.

See Also

Other visualization functions: plot.nplcm(), plot_BrS_panel(), plot_SS_panel(), plot_check_pairwise_SLORD(), plot_etiology_regression(), plot_etiology_strat(), plot_panels(), plot_pie_panel(), plot_subwt_regression()

Examples

```
data(data_nplcm_noreg)
cause_list <- LETTERS[1:6]
J.BrS      <- 6
model_options_no_reg <- list(
  likelihood   = list(
    cause_list = cause_list,
    k_subclass = 2,
    Eti_formula = ~-1, # no covariate for the etiology regression
    FPR_formula = list(
      MBS1 =   ~-1)    # no covariate for the subclass weight regression
  ),
  use_measurements = c("BrS"),
  # use bronze-standard data only for model estimation.
  prior= list(
    Eti_prior = overall_uniform(1,cause_list),
    # Dirichlet(1,...,1) prior for the etiology.
    TPR_prior  = list(BrS = list(
      info  = "informative", # informative prior for TPRs
      input = "match_range",
      # specify the informative prior for TPRs by specifying a plausible range.
      val = list(MBS1 = list(up =  list(rep(0.99,J.BrS)),
                             # upper ranges: matched to 97.5% quantile of a Beta prior
                             low = list(rep(0.55,J.BrS))))
      # lower ranges: matched to 2.5% quantile of a Beta prior
    )
    )
  )
)


set.seed(1)
# include stratification information in file name:
thedir    <- paste0(tempdir(),"_no_reg")

# create folders to store the model results
dir.create(thedir, showWarnings = FALSE)
result_folder_no_reg <- file.path(thedir,paste("results",collapse="_"))
thedir <- result_folder_no_reg
dir.create(thedir, showWarnings = FALSE)
```

```
# options for MCMC chains:
mcmc_options_no_reg <- list(
  debugstatus = TRUE,
  n.chains = 1,
  n.itermcmc = as.integer(200),
  n.burnin = as.integer(100),
  n.thin = 1,
  individual.pred = FALSE,
  ppd = TRUE,
  result.folder = thedir,
  bugsmodel.dir = thedir
)

BrS_object_1 <- make_meas_object(patho = LETTERS[1:6],
                                 specimen = "MBS", test = "1",
                                 quality = "BrS", cause_list = cause_list)
clean_options <- list(BrS_objects = make_list(BrS_object_1))
# place the nplcm data and cleaning options into the results folder
dput(data_nplcm_noreg,file.path(thedir,"data_nplcm.txt"))
dput(clean_options, file.path(thedir, "data_clean_options.txt"))

rjags::load.module("glm")

nplcm_noreg <- nplcm(data_nplcm_noreg,model_options_no_reg,mcmc_options_no_reg)

plot_check_common_pattern(nplcm_noreg$DIR_NPLCM)
```

`plot_check_pairwise_SLORD` *Posterior predictive checking for nested partially latent class models - pairwise log odds ratio (only for bronze-standard data)*

Description

At each MCMC iteration, we generate a new data set based on the model and parameter values at that iteration. The sample size of the new data set equals that of the actual data set, i.e. the same number of cases and controls.

Usage

```
plot_check_pairwise_SLORD(DIR_NPLCM, slice = 1)
```

Arguments

`DIR_NPLCM`	File path to the folder that stores results from npLCM fit.
`slice`	Default is 1, for the first slice of BrS data.

Value

A figure of posterior predicted log odds ratio compared with the observed log odds ratio for the BrS data. The function generates this figure in your working directory automatically.

See Also

Other visualization functions: plot.nplcm(), plot_BrS_panel(), plot_SS_panel(), plot_check_common_pattern(), plot_etiology_regression(), plot_etiology_strat(), plot_panels(), plot_pie_panel(), plot_subwt_regression()

Examples

```
 data(data_nplcm_noreg)
cause_list <- LETTERS[1:6]
J.BrS      <- 6
model_options_no_reg <- list(
  likelihood   = list(
    cause_list = cause_list,
    k_subclass = 2,
    Eti_formula = ~-1, # no covariate for the etiology regression
    FPR_formula = list(
      MBS1 =   ~-1)    # no covariate for the subclass weight regression
  ),
  use_measurements = c("BrS"),
  # use bronze-standard data only for model estimation.
  prior= list(
    Eti_prior = overall_uniform(1,cause_list),
    # Dirichlet(1,...,1) prior for the etiology.
    TPR_prior  = list(BrS = list(
      info  = "informative", # informative prior for TPRs
      input = "match_range",
      # specify the informative prior for TPRs by specifying a plausible range.
      val = list(MBS1 = list(up =  list(rep(0.99,J.BrS)),
                             # upper ranges: matched to 97.5% quantile of a Beta prior
                             low = list(rep(0.55,J.BrS))))
      # lower ranges: matched to 2.5% quantile of a Beta prior
    )
    )
  )
)


set.seed(1)
# include stratification information in file name:
thedir    <- paste0(tempdir(),"_no_reg")

# create folders to store the model results
dir.create(thedir, showWarnings = FALSE)
result_folder_no_reg <- file.path(thedir,paste("results",collapse="_"))
thedir <- result_folder_no_reg
dir.create(thedir, showWarnings = FALSE)
```

```
# options for MCMC chains:
mcmc_options_no_reg <- list(
  debugstatus = TRUE,
  n.chains = 1,
  n.itermcmc = as.integer(200),
  n.burnin = as.integer(100),
  n.thin = 1,
  individual.pred = FALSE,
  ppd = TRUE,
  result.folder = thedir,
  bugsmodel.dir = thedir
)

BrS_object_1 <- make_meas_object(patho = LETTERS[1:6],
                                 specimen = "MBS", test = "1",
                                 quality = "BrS", cause_list = cause_list)
clean_options <- list(BrS_objects = make_list(BrS_object_1))
# place the nplcm data and cleaning options into the results folder
dput(data_nplcm_noreg,file.path(thedir,"data_nplcm.txt"))
dput(clean_options, file.path(thedir, "data_clean_options.txt"))

rjags::load.module("glm")

nplcm_noreg <- nplcm(data_nplcm_noreg,model_options_no_reg,mcmc_options_no_reg)

plot_check_pairwise_SLORD(nplcm_noreg$DIR_NPLCM,slice=1)
```

`plot_etiology_regression` *visualize the etiology regression with a continuous covariate*

Description

This function visualizes the etiology regression against one continuous covariate, e.g., enrollment date. (NB: dealing with NoA, multiple-pathogen causes, other continuous covariates? also there this function only plots the first slice - so generalization may be useful - give users an option to choose slice s; currently default to the first slice.)

Usage

```
plot_etiology_regression(
  DIR_NPLCM,
  stratum_bool,
  slice = 1,
  plot_basis = FALSE,
  truth = NULL,
  RES_NPLCM = NULL,
  do_plot = TRUE,
```

```
  do_rug = TRUE,
  return_metric = TRUE,
  plot_ma_dots = FALSE
)
```

Arguments

DIR_NPLCM	File path to the folder containing posterior samples
stratum_bool	a vector of TRUE/FALSE with TRUE indicating the rows of subjects to include
slice	integer; specifies which slice of bronze-standard data to visualize; Default to 1.
plot_basis	TRUE for plotting basis functions; Default to FALSE
truth	a list of truths computed from true parameters in simulations; elements: Eti, FPR, PR_case,TPR; All default to NULL in real data analyses. Currently only works for one slice of bronze-standard measurements (in a non-nested model). • Eti matrix of # of rows = # of subjects, # columns: length(cause_list) for Eti • FPR matrix of # of rows = # of subjects, # columns: ncol(data_nplcm$Mobs$MBS$MBS1) • PR_case matrix of # of rows = # of subjects, # columns: ncol(data_nplcm$Mobs$MBS$MBS1) • TPR a vector of length identical to PR_case
RES_NPLCM	pre-read res_nplcm; default to NULL.
do_plot	TRUE for plotting
do_rug	TRUE for plotting
return_metric	TRUE for showing overall mean etiology, quantiles, s.d., and if truth$Eti is supplied, coverage, bias, truth and integrated mean squared errors (IMSE).
plot_ma_dots	plot moving averages among case and controls if TRUE; Default to FALSE.

Value

A figure of etiology regression curves and some marginal positive rate assessment of model fit; See example for the legends.

References

See example figures

- A Figure using simulated data for six pathogens: https://github.com/zhenkewu/baker/blob/master/inst/figs/visualize_etiology_regression_SITE=1.pdf
- The legends for the figure above: https://github.com/zhenkewu/baker/blob/master/inst/figs/legends_visualize_etiology_regression.png

See Also

Other visualization functions: plot.nplcm(), plot_BrS_panel(), plot_SS_panel(), plot_check_common_pattern(), plot_check_pairwise_SLORD(), plot_etiology_strat(), plot_panels(), plot_pie_panel(), plot_subwt_regression()

plot_etiology_strat *visualize the etiology estimates for each discrete levels*

Description

This function visualizes the etiology estimates against one discrete covariate, e.g., age groups.

Usage

```
plot_etiology_strat(
  DIR_NPLCM,
  strata_weights = "empirical",
  truth = NULL,
  RES_NPLCM = NULL,
  show_levels = 0,
  is_plot = TRUE,
  VERBOSE = TRUE
)
```

Arguments

DIR_NPLCM	File path to the folder containing posterior samples
strata_weights	a vector of weights that sum to one; for each pathogen the weights specify how the j-th etiology fraction should be combined across all levels of the discrete predictors in the data; default is "empirical" to use empirical weights (observed fractions of subjects across strata).
truth	a list of true values, e.g., truth=list(allEti = <a list of etiology fractions, each of identical if available, will be shown in thicker red solid vertical lines.
RES_NPLCM	pre-read res_nplcm; default to NULL.
show_levels	a vector of integers less than or equal to the total number of levels of strata; default to 0 for overall.
is_plot	default to TRUE, plotting the figures; if FALSE only returning summaries
VERBOSE	default to TRUE, print actual meanings of the levels

Value

plotting function

See Also

Other visualization functions: plot.nplcm(), plot_BrS_panel(), plot_SS_panel(), plot_check_common_pattern(), plot_check_pairwise_SLORD(), plot_etiology_regression(), plot_panels(), plot_pie_panel(), plot_subwt_regression()

plot_leftmost *plotting the labels on the left margin for panels plot*

Description

plotting the labels on the left margin for panels plot

Usage

```
plot_leftmost(model_options, height)
```

Arguments

model_options	See nplcm()
height	no. of rows in the panels plot; commonly set as length(select_latent)

Value

a plot

See Also

plot_panels

plot_logORmat *Visualize pairwise log odds ratios (LOR) for data that are available in both cases and controls*

Description

Visualize pairwise log odds ratios (LOR) for data that are available in both cases and controls

Usage

```
plot_logORmat(data_nplcm, pathogen_display, BrS_slice = 1, logOR_rounding = 2)
```

Arguments

data_nplcm	See assign_model().
pathogen_display	The pathogen vector in desired order for display. It can be of larger length than that of pathogen_BrS.
BrS_slice	Default is 1 - the set of BrS data to visualize.
logOR_rounding	Rounding number of the log odds ratio. Default is 2.

Details

plot_logORmat visualizes a matrix of pairwise log odds ratios (LOR) for cases (upper) and controls (lower). LOR is at the top of the cell. Below it, its standard error is in smaller type, using the same color as the LOR. Then the estimate is divided by its standard error. We put the actual value when the Z-statistics has an absolute value greater than 2; a plus (red) or minus (blue) if between 1 and 2; blank otherwise.

Value

Figure of LOR matrix and relevant s.e. and significance information.

See Also

Other exploratory data analysis functions: get_top_pattern(), show_individual(), summarize_BrS(), summarize_SS(), visualize_season()

Examples

```
data(data_nplcm_noreg)
plot_logORmat(data_nplcm_noreg,names(data_nplcm_noreg$Mobs$MBS[[1]]))
```

---

plot_panels *Plot three-panel figures for nested partially-latent model results*

Description

plot_panels() visualizes the model outputs for communicating how the data inform final latent disease status (etiology). It works for singleton or combo etiologies.

Usage

```
plot_panels(
  DIR_NPLCM,
  slices = "all",
  bg_color = list(BrS = "lavenderblush", SS = "mistyrose", pie = "antiquewhite"),
  select_latent = NULL,
  exact = TRUE,
  SS_upperlimit = 1,
  eti_upperlimit = 1,
  silent = TRUE,
  ref_eti0 = NULL,
  is_plot = TRUE
)
```

Arguments

DIR_NPLCM	File path to the folder containing posterior samples
slices	DEFAULT is "all" - to plot all measurements; Otherwise, one can specify a list: `list(MBS=c(1,3),MSS=1)` means to plot the 1st and 3rd slice of BrS measurements and 1st of SS measurement.
bg_color	A list with names "BrS", "SS", "pie" to specify background colors. The current default is `list(BrS = "lavenderblush", SS = "mistyrose", pie="antiquewhite")`. If no background is intended, specify as NULL or for a particular measurement, e.g., `BrS = NULL`.
select_latent	a vector of character strings representing latent status. It is used for just plotting a subset of latent status. For example, you can specify `select_latent = "HINF"` to plot all latent status information relevant to `"HINF"`.
exact	Default is `TRUE` to use `select_latent` as exact names of causes. If you want to specify a name and plot all single or combo causes with that name, specify it to be `FALSE`.
SS_upperlimit	The upper limit of horizontal bar for the silver-standard subpanel (the middle panel). The default value is .25.
eti_upperlimit	The upper limit of horizontal bar for the etiology posterior subpanel (the rightmost panel). The default value is .4
silent	Default is TRUE to not print any warning messages; FALSE otherwise.
ref_eti0	reference quantiles and means; a list: pEti_ref_q, pEti_ref_mean_ord
is_plot	default to `TRUE` for plotting only; set to `FALSE` if to get summary.

Details

Missing data for BrS or SS are dropped when calculating observed measurement positive rates

Value

A figure with two or three columns (if `is_plot=TRUE`); otherwise, it provide posterior summaries of Etiology information to used by `print.summary.nplcm.no_reg()`

See Also

Other visualization functions: `plot.nplcm()`, `plot_BrS_panel()`, `plot_SS_panel()`, `plot_check_common_pattern()`, `plot_check_pairwise_SLORD()`, `plot_etiology_regression()`, `plot_etiology_strat()`, `plot_pie_panel()`, `plot_subwt_regression()`

`plot_pie_panel` *Plot etiology (pie) panel*

Description

Plot etiology (pie) panel

Usage

```
plot_pie_panel(
  model_options,
  res_nplcm,
  bugs.dat,
  bg_color,
  select_latent = NULL,
  exact = TRUE,
  top_pie = 1,
  label_size = 1,
  ref_eti = NULL,
  is_plot = TRUE
)
```

Arguments

`model_options`	See `nplcm()`
`res_nplcm`	See `nplcm_read_folder()`
`bugs.dat`	Data input for the model fitting.
`bg_color`	A list with names "BrS", "SS", "pie" to specify background colors
`select_latent`	a vector of character strings representing latent status. It is used for just plotting a subset of latent status. For example, you can specify `select_latent = "HINF"`
`exact`	Default is `TRUE` to use `select_latent` as exact names of causes. If you want to specify a name and plot all single or combo causes with that name, specify it to be `FALSE`. to plot all latent status information relevant to `"HINF"`.
`top_pie`	Numerical value to specify the rightmost limit on the horizontal axis for the pie panel.
`label_size`	the size of latent status labels on the right margin
`ref_eti`	reference quantiles and means; a list: pEti_ref_q, pEti_ref_mean_ord
`is_plot`	default to `TRUE` for plotting only; set to `FALSE` if to get summary.

Value

plotting function.

See Also

Other visualization functions: plot.nplcm(), plot_BrS_panel(), plot_SS_panel(), plot_check_common_pattern(), plot_check_pairwise_SLORD(), plot_etiology_regression(), plot_etiology_strat(), plot_panels(), plot_subwt_regression()

`plot_SS_panel` *Plot silver-standard (SS) panel*

Description

Plot silver-standard (SS) panel

Usage

```
plot_SS_panel(
  slice,
  data_nplcm,
  model_options,
  clean_options,
  bugs.dat,
  res_nplcm,
  bg_color,
  select_latent = NULL,
  exact = TRUE,
  top_SS = 1,
  cexval = 1,
  srtval = 0,
  prior_shape = "interval"
)
```

Arguments

slice	the index of measurement slice for SS.
data_nplcm	See nplcm()
model_options	See nplcm()
clean_options	See clean_perch_data()
bugs.dat	Data input for the model fitting.
res_nplcm	See nplcm_read_folder()
bg_color	A list with names "BrS", "SS", "pie" to specify background colors
select_latent	a vector of character strings representing latent status. It is used for just plotting a subset of latent status. For example, you can specify select_latent = "HINF" to plot all latent status information relevant to "HINF".
exact	Default is TRUE to use select_latent as exact names of causes. If you want to specify a name and plot all single or combo causes with that name, specify it to be FALSE.

top_SS	Numerical value to specify the rightmost limit on the horizontal axis for the SS panel.
cexval	Default is 1 - size of text of the SS percentages.
srtval	Default is 0 - the direction of the text for the SS percentages.
prior_shape	interval or boxplot - for how to represent prior/posteriors of the TPR/FPRs of measurements.

Value

plotting function

See Also

Other visualization functions: plot.nplcm(), plot_BrS_panel(), plot_check_common_pattern(), plot_check_pairwise_SLORD(), plot_etiology_regression(), plot_etiology_strat(), plot_panels(), plot_pie_panel(), plot_subwt_regression()

---

plot_subwt_regression *visualize the subclass weight regression with a continuous covariate*

Description

visualize the subclass weight regression with a continuous covariate

Usage

```
plot_subwt_regression(
  DIR_NPLCM,
  stratum_bool,
  case = 0,
  slice = 1,
  truth = NULL,
  RES_NPLCM = NULL
)
```

Arguments

DIR_NPLCM	File path to the folder containing posterior samples
stratum_bool	a vector of TRUE/FALSE with TRUE indicating the rows of subjects to include
case	1 for plotting cases, 0 for plotting controls; default to 0.
slice	integer; specifies which slice of bronze-standard data to visualize; Default to 1.
truth	a list of truths computed from true parameters in simulations; elements: Eti, FPR, PR_case,TPR; All default to NULL in real data analyses. Currently only works for one slice of bronze-standard measurements (in a non-nested model). • truth_subwt matrix of # of rows = # of subjects, # columns: number of true subclasses
RES_NPLCM	pre-read res_nplcm; default to NULL.

Value

A figure of subclass regression curves

See Also

Other visualization functions: plot.nplcm(), plot_BrS_panel(), plot_SS_panel(), plot_check_common_pattern(), plot_check_pairwise_SLORD(), plot_etiology_regression(), plot_etiology_strat(), plot_panels(), plot_pie_panel()

---

print.nplcm print.nplcm *summarizes the results from* nplcm().

Description

print.nplcm summarizes the results from nplcm().

Usage

```
## S3 method for class 'nplcm'
print(x, ...)
```

Arguments

x	Output from nplcm().
...	Arguments passed to summary and printing methods.

Value

Summary of object output by nplcm() — need details.

See Also

Other nplcm results: print.summary.nplcm.no_reg(), print.summary.nplcm.reg_nest_strat(), print.summary.nplcm.reg_nest(), print.summary.nplcm.reg_nonest_strat(), print.summary.nplcm.reg_nones
summary.nplcm()

print.summary.nplcm.no_reg *Compact printing of* nplcm() *model fits*

Description

print.summary.nplcm is a print method for class summary.nplcm.NoReg.

Usage

```
## S3 method for class 'summary.nplcm.no_reg'
print(x, ...)
```

Arguments

x	output from summary.nplcm with summary.nplcm.no_reg as the output object class.
...	Not used.

Value

see print.nplcm()

See Also

Other nplcm results: print.nplcm(), print.summary.nplcm.reg_nest_strat(), print.summary.nplcm.reg_nest(), print.summary.nplcm.reg_nonest_strat(), print.summary.nplcm.reg_nonest(), summary.nplcm()

print.summary.nplcm.reg_nest *Compact printing of* nplcm() *model fits*

Description

print.summary.nplcm is a print method for class summary.nplcm.reg_nest.

Usage

```
## S3 method for class 'summary.nplcm.reg_nest'
print(x, ...)
```

Arguments

x	output from summary.nplcm with summary.nplcm.reg_nest as the output object class.
...	Not used.

Value

see print.nplcm()

See Also

Other nplcm results: print.nplcm(), print.summary.nplcm.no_reg(), print.summary.nplcm.reg_nest_strat(), print.summary.nplcm.reg_nonest_strat(), print.summary.nplcm.reg_nonest(), summary.nplcm()

---

print.summary.nplcm.reg_nest_strat

Compact printing of nplcm() *model fits*

Description

Same as print.summary.nplcm.reg_nonest_strat()

Usage

```
## S3 method for class 'summary.nplcm.reg_nest_strat'
print(x, ...)
```

Arguments

x	output from summary.nplcm with summary.nplcm.reg_nest_strat as the output object class.
...	Not used.

Details

print.summary.nplcm is a print method for class summary.nplcm.reg_nest_strat.

Value

see print.nplcm()

See Also

Other nplcm results: print.nplcm(), print.summary.nplcm.no_reg(), print.summary.nplcm.reg_nest(), print.summary.nplcm.reg_nonest_strat(), print.summary.nplcm.reg_nonest(), summary.nplcm()

print.summary.nplcm.reg_nonest

Compact printing of nplcm() *model fits*

Description

print.summary.nplcm is a print method for class summary.nplcm.reg_nonest.

Usage

```
## S3 method for class 'summary.nplcm.reg_nonest'
print(x, ...)
```

Arguments

x	output from summary.nplcm with summary.nplcm.reg_nonest as the output object class.
...	Not used.

Value

see print.nplcm()

See Also

Other nplcm results: print.nplcm(), print.summary.nplcm.no_reg(), print.summary.nplcm.reg_nest_strat(), print.summary.nplcm.reg_nest(), print.summary.nplcm.reg_nonest_strat(), summary.nplcm()

print.summary.nplcm.reg_nonest_strat

Compact printing of nplcm() *model fits*

Description

print.summary.nplcm is a print method for class summary.nplcm.reg_nonest_strat.

Usage

```
## S3 method for class 'summary.nplcm.reg_nonest_strat'
print(x, ...)
```

Arguments

x	output from summary.nplcm with summary.nplcm.reg_nonest_strat as the output object class.
...	Not used.

Value

see print.nplcm()

See Also

Other nplcm results: print.nplcm(), print.summary.nplcm.no_reg(), print.summary.nplcm.reg_nest_strat(), print.summary.nplcm.reg_nest(), print.summary.nplcm.reg_nonest(), summary.nplcm()

---

read_meas_object *Read measurement slices*

Description

NB: add example, small data

Usage

```
read_meas_object(object, data)
```

Arguments

object	Outputs from make_meas_object()
data	Raw data with column names {pathogen name}_{specimen}{test}

Value

A list with two elements: meas-data frame with measurements; position-see lookup_quality()

See Also

Other raw data importing functions: extract_data_raw()

---

rvbern *Sample a vector of Bernoulli variables.*

Description

Sample a vector of Bernoulli variables with higher speed (same length with "p"). The Bernoulli random variables can have different means.

Usage

```
rvbern(p)
```

Arguments

p	A vector of probabilities, each being the head probability of an independent coin toss

Value

A vector of 1s (head) and 0s (tail)

Examples

```
rvbern(c(0.9,0.1,0.2,0.3))
```

---

set_prior_tpr_BrS_NoNest

Set true positive rate (TPR) prior ranges for bronze-standard (BrS) data

---

Description

`set_prior_tpr_BrS_NoNest` is for for conditional independence models. We currently also use it for conditional dependence model: subclass TPRs are independently assigned a beta prior. Ongoing work will enable specifying priors for the marginal TPR, i.e. TPRs for a disease class, not for the finer subclass.

Usage

```
set_prior_tpr_BrS_NoNest(slice, model_options, data_nplcm)
```

Arguments

slice	The BrS measurement slice under consideration.
model_options	See `nplcm()` function.
data_nplcm	See `assign_model()` function.

Value

Parameters for the BrS dta TPR priors. It is a list of two lists (alpha and beta). Alpha and beta are of the same length, the number of BrS measurement slices. Each element of the alpha (beta) list is a numeric vector for alpha (beta) parameters as in BETA distribution.

See Also

Other prior specification functions: `overall_uniform()`, `set_prior_tpr_SS()`

set_prior_tpr_SS *Set true positive rate (TPR) prior ranges for silver-standard data.*

Description

Set true positive rate (TPR) prior ranges for silver-standard data.

Usage

```
set_prior_tpr_SS(model_options, data_nplcm)
```

Arguments

model_options	See nplcm() function.
data_nplcm	See assign_model() function.

Value

Parameters for the SS data TPR priors. It is a list of two lists (alpha and beta). Alpha and beta are of the same length, the number of BrS measurement slices. Each element of the alpha (beta) list is a numeric vector for alpha (beta) parameters to specify Beta prior for TPRs.

See Also

Other prior specification functions: overall_uniform(), set_prior_tpr_BrS_NoNest()

set_strat *Stratification setup by covariates*

Description

set_strat makes group indicators based on model_options$X_reg_*

Usage

```
set_strat(X, X_reg)
```

Arguments

X	A data frame of covariates
X_reg	The vector of covariates that will stratify the analyses. These variables have to be categorical.

Details

the results from this function will help stratify etiology or FPR for different strata; the ways of stratification for etiology and FPR can be based on different covariates.

Value

A list with following elements:

- `N_group` The number of groups
- `group` A vector of group indicator for every observation

`show_dep` *Show function dependencies*

Description

Show function dependencies

Usage

```
show_dep(fname, pckg = "package:baker", ...)
```

Arguments

`fname`	Character string for one function
`pckg`	Package name; default is `"package:baker"`
`...`	Other parameters accepted by `mvbutils::foodweb()`

Value

A figure showing function dependencies

Examples

```
show_dep("nplcm",ancestor=FALSE)
show_dep("nplcm")
```

show_individual *get an individual's data from the output of* clean_perch_data()

Description

get an individual's data from the output of clean_perch_data()

Usage

```
show_individual(data_nplcm, ID)
```

Arguments

data_nplcm	data for fitting nplcm; See nplcm()
ID	patient id: patid.

Value

a list with the inquired patient's data

See Also

Other exploratory data analysis functions: get_top_pattern(), plot_logORmat(), summarize_BrS(), summarize_SS(), visualize_season()

Examples

```
data(data_nplcm_noreg)
data_nplcm_noreg$X$patid <- paste("PAT",1:length(data_nplcm_noreg$Y0),sep="")
data_nplcm_noreg$X <- as.data.frame(data_nplcm_noreg$X)
subset_data_nplcm_by_index(data_nplcm_noreg,which(data_nplcm_noreg$X$patid%in%c("PAT12","PAT408")))
data_nplcm_noreg$X <- NULL
```

simulate_brs *Simulate Bronze-Standard (BrS) Data*

Description

Simulate Bronze-Standard (BrS) Data

Usage

```
simulate_brs(set_parameter, latent_samples)
```

Arguments

`set_parameter` True model parameters in an npLCM specification:

`cause_list` a vector of disease class names among cases (since the causes could be multi-agent (e.g., multiple pathogens may cause an individual case's pneumonia), so its length could be longer than the total number of unique causative agents)

`etiology` a vector of proportions that sum to 100 percent

`pathogen_BrS` a vector of putative causative agents' names measured in bronze-standard (BrS) data. This function simulates only one slice defined by `specimen``test``pathogen`

`pathogen_SS` a vector of pathogen names measured in silver-standard (SS) data.

`meas_nm` a list of `specimen``test` names e.g., `list(MBS = c("NPPCR"),MSS="BCX")` for nasopharyngeal (NP) specimen tested by polymerase chain reaction (PCR) - `NPPCR` and blood (B) tested by culture (Cx) - `BCX`

`Lambda` controls' subclass weights $\nu_1, \nu_2, \ldots, \nu_K$ a vector of `K` probabilities that sum to 1.

`Eta` a matrix of dimension `length(cause_list)` by `K`; each row represents a disease class (among cases); the values in that row are subclass weights $\eta_1, \eta_2, \ldots, \eta_K$ for that disease class, so needs to sum to one. In Wu et al. 2016 (JRSS-C), the subclass weights are the same across disease classes across rows. But when simulating data, one can specify rows with distinct subclass weights - it is a matter whether we can recover these parameters (possible when some cases' true disease classes are observed)

`PsiBS/PsiSS` False positive rates for Bronze-Standard data and for Silver-Standard data. For example, the rows of `PsiBS` correspond to the dimension of the particular slice of BrS measures, e.g., `10` for 10 causative agents measured by NPPCR; the columns correspond to `K` subclasses; generically, the dimension is `J` by `K` `PsiSS` is supposed to be a vector of all zeros (perfect specificity in silver-standard measures).

`ThetaBS/ThetaSS` True positive rates Θ for Bronze-Standard data and for Silver-Standard data. Dimension is `J` by `K` (can contain `NA` if the total number of causative agents measured by BrS or SS exceeds the measured causative agents in SS. For example, in PERCH study, nasopharyngeal polymerase chain reaction (NPPCR; bronze-standard) may target 30 distinct pathogens, but blood culture (BCX; silver-standard) may only target a subset of the 30, so we have to specify `NA` in `ThetaSS`for those pathogens not targeted by BCX).

`Nu` the number of control subjects

`Nd` the number of case subjects

`latent_samples` simulated latent status for all the subjects, for use in simulating BrS measurements.

Value

a data frame with first column being case-control status (case at top) and columns of bronze-standard measurements

See Also

Other internal simulation functions: `simulate_latent()`, `simulate_ss()`

`simulate_latent` *Simulate Latent Status:*

Description

Simulate Latent Status:

Usage

```
simulate_latent(set_parameter)
```

Arguments

`set_parameter` True model parameters in an npLCM specification:

`cause_list` a vector of disease class names among cases (since the causes could be multi-agent (e.g., multiple pathogens may cause an individual case's pneumonia), so its length could be longer than the total number of unique causative agents)

`etiology` a vector of proportions that sum to 100 percent

`pathogen_BrS` a vector of putative causative agents' names measured in bronze-standard (BrS) data. This function simulates only one slice defined by `specimen``test``pathogen`

`pathogen_SS` a vector of pathogen names measured in silver-standard (SS) data.

`meas_nm` a list of `specimen``test` names e.g., `list(MBS = c("NPPCR"),MSS="BCX")` for nasopharyngeal (NP) specimen tested by polymerase chain reaction (PCR) - `NPPCR` and blood (B) tested by culture (Cx) - `BCX`

`Lambda` controls' subclass weights $\nu_1, \nu_2, \ldots, \nu_K$ a vector of `K` probabilities that sum to 1.

`Eta` a matrix of dimension `length(cause_list)` by `K`; each row represents a disease class (among cases); the values in that row are subclass weights $\eta_1, \eta_2, \ldots, \eta_K$ for that disease class, so needs to sum to one. In Wu et al. 2016 (JRSS-C), the subclass weights are the same across disease classes across rows. But when simulating data, one can specify rows with distinct subclass weights - it is a matter whether we can recover these parameters (possible when some cases' true disease classes are observed)

`PsiBS/PsiSS` False positive rates for Bronze-Standard data and for Silver-Standard data. For example, the rows of `PsiBS` correspond to the dimension of the particular slice of BrS measures, e.g., `10` for 10 causative agents measured by NPPCR; the columns correspond to `K` subclasses; generically, the dimension is `J` by `K` `PsiSS` is supposed to be a vector of all zeros (perfect specificity in silver-standard measures).

ThetaBS/ThetaSS True positive rates Θ for Bronze-Standard data and for Silver-Standard data. Dimension is J by K (can contain NA if the total number of causative agents measured by BrS or SS exceeds the measured causative agents in SS. For example, in PERCH study, nasopharyngeal polymerase chain reaction (NPPCR; bronze-standard) may target 30 distinct pathogens, but blood culture (BCX; silver-standard) may only target a subset of the 30, so we have to specify NA in ThetaSSfor those pathogens not targeted by BCX).

Nu the number of control subjects

Nd the number of case subjects

Value

a list of latent status samples for use in simulating measurements. It also includes a template to look up measurement parameters for each disease class.

See Also

Other internal simulation functions: simulate_brs(), simulate_ss()

---

simulate_nplcm *Simulate data from nested partially-latent class model (npLCM) family*

Description

Simulate data from nested partially-latent class model (npLCM) family

Usage

```
simulate_nplcm(set_parameter)
```

Arguments

set_parameter True model parameters in an npLCM specification:

cause_list a vector of disease class names among cases (since the causes could be multi-agent (e.g., multiple pathogens may cause an individual case's pneumonia), so its length could be longer than the total number of unique causative agents)

etiology a vector of proportions that sum to 100 percent

pathogen_BrS a vector of putative causative agents' names measured in bronze-standard (BrS) data. This function simulates only one slice defined by specimen``test``pathogen

pathogen_SS a vector of pathogen names measured in silver-standard (SS) data.

meas_nm a list of specimen``test names e.g., list(MBS = c("NPPCR"),MSS="BCX") for nasopharyngeal (NP) specimen tested by polymerase chain reaction (PCR) - NPPCR and blood (B) tested by culture (Cx) - BCX

`Lambda` controls' subclass weights $\nu_1, \nu_2, \ldots, \nu_K$ a vector of `K` probabilities that sum to 1.

`Eta` a matrix of dimension `length(cause_list)` by `K`; each row represents a disease class (among cases); the values in that row are subclass weights $\eta_1, \eta_2, \ldots, \eta_K$ for that disease class, so needs to sum to one. In Wu et al. 2016 (JRSS-C), the subclass weights are the same across disease classes across rows. But when simulating data, one can specify rows with distinct subclass weights - it is a matter whether we can recover these parameters (possible when some cases' true disease classes are observed)

`PsiBS/PsiSS` False positive rates for Bronze-Standard data and for Silver-Standard data. For example, the rows of `PsiBS` correspond to the dimension of the particular slice of BrS measures, e.g., `10` for 10 causative agents measured by NPPCR; the columns correspond to `K` subclasses; generically, the dimension is `J` by `K` `PsiSS` is supposed to be a vector of all zeros (perfect specificity in silver-standard measures).

`ThetaBS/ThetaSS` True positive rates Θ for Bronze-Standard data and for Silver-Standard data. Dimension is `J` by `K` (can contain `NA` if the total number of causative agents measured by BrS or SS exceeds the measured causative agents in SS. For example, in PERCH study, nasopharyngeal polymerase chain reaction (NPPCR; bronze-standard) may target 30 distinct pathogens, but blood culture (BCX; silver-standard) may only target a subset of the 30, so we have to specify `NA` in `ThetaSS`for those pathogens not targeted by BCX).

`Nu` the number of control subjects

`Nd` the number of case subjects

Value

A list of diagnostic test measurements, true latent statues:

`data_nplcm` a list of structured data (see `nplcm()` for description).

`template` a matrix: rows for causes (may comprise a single or multiple causative agents), columns for measurements; generated as a lookup table to match disease-class specific parameters (true and false positive rates)

`latent_cat` integer values to indicate the latent category. The integer code corresponds to the order specified in `set_parameter$etiology`. Controls are coded as `length(set_parameter$etiology)+1`.)

See Also

simulate_latent for simulating discrete latent status, given which simulate_brs simulates bronze-standard data.

Examples

```
K.true  <- 2   # no. of latent subclasses in actual simulation.
               # If eta = c(1,0), effectively, it is K.true=1.
J       <- 21   # no. of pathogens.
N       <- 600 # no. of cases/controls.
```

```
eta <- c(1,0)
# if it is c(1,0),then it is conditional independence model, and
# only the first column of parameters in PsiBS, ThetaBS matter!

seed_start <- 20150202
print(eta)

# set fixed simulation sequence:
set.seed(seed_start)

ThetaBS_withNA <- c(.75,rep(c(.75,.75,.75,NA),5))
PsiBS_withNA <- c(.15,rep(c(.05,.05,.05,NA),5))

ThetaSS_withNA <- c(NA,rep(c(0.15,NA,0.15,0.15),5))
PsiSS_withNA <- c(NA,rep(c(0,NA,0,0),5))

set_parameter <- list(
  cause_list      = c(LETTERS[1:J]),
  etiology        = c(c(0.36,0.1,0.1,0.1,0.1,0.05,0.05,0.05,
                 0.05,0.01,0.01,0.01,0.01),rep(0.00,8)),
                 #same length as cause_list.
  pathogen_BrS    = LETTERS[1:J][!is.na(ThetaBS_withNA)],
  pathogen_SS     = LETTERS[1:J][!is.na(ThetaSS_withNA)],
  meas_nm         = list(MBS = c("MBS1"),MSS="MSS1"),
  Lambda          = eta, #ctrl mix
  Eta             = t(replicate(J,eta)), #case mix, row number equal to Jcause.
  PsiBS           = cbind(PsiBS_withNA[!is.na(PsiBS_withNA)],
                          rep(0,sum(!is.na(PsiBS_withNA)))),
  ThetaBS         = cbind(ThetaBS_withNA[!is.na(ThetaBS_withNA)],
                          rep(0,sum(!is.na(ThetaBS_withNA)))),
  PsiSS           = PsiSS_withNA[!is.na(PsiSS_withNA)],
  ThetaSS         = ThetaSS_withNA[!is.na(ThetaSS_withNA)],
  Nu      =     N, # control size.
  Nd      =     N  # case size.
)
 simu_out <- simulate_nplcm(set_parameter)
 data_nplcm <- simu_out$data_nplcm

 pathogen_display <- rev(set_parameter$pathogen_BrS)
 plot_logORmat(data_nplcm,pathogen_display)
 # more examples are provided in the vignette, including settings with
 # covariates.
```

`simulate_ss` *Simulate Silver-Standard (SS) Data*

Description

Simulate Silver-Standard (SS) Data

Usage

```
simulate_ss(set_parameter, latent_samples)
```

Arguments

`set_parameter` True model parameters in an npLCM specification:

- `cause_list` a vector of disease class names among cases (since the causes could be multi-agent (e.g., multiple pathogens may cause an individual case's pneumonia), so its length could be longer than the total number of unique causative agents)
- `etiology` a vector of proportions that sum to 100 percent
- `pathogen_BrS` a vector of putative causative agents' names measured in bronze-standard (BrS) data. This function simulates only one slice defined by `specimen``test``pathogen`
- `pathogen_SS` a vector of pathogen names measured in silver-standard (SS) data.
- `meas_nm` a list of `specimen``test` names e.g., `list(MBS = c("NPPCR"),MSS="BCX")` for nasopharyngeal (NP) specimen tested by polymerase chain reaction (PCR) - `NPPCR` and blood (B) tested by culture (Cx) - `BCX`
- `Lambda` controls' subclass weights $\nu_1, \nu_2, \ldots, \nu_K$ a vector of `K` probabilities that sum to 1.
- `Eta` a matrix of dimension `length(cause_list)` by `K`; each row represents a disease class (among cases); the values in that row are subclass weights $\eta_1, \eta_2, \ldots, \eta_K$ for that disease class, so needs to sum to one. In Wu et al. 2016 (JRSS-C), the subclass weights are the same across disease classes across rows. But when simulating data, one can specify rows with distinct subclass weights - it is a matter whether we can recover these parameters (possible when some cases' true disease classes are observed)
- `PsiBS/PsiSS` False positive rates for Bronze-Standard data and for Silver-Standard data. For example, the rows of `PsiBS` correspond to the dimension of the particular slice of BrS measures, e.g., `10` for 10 causative agents measured by NPPCR; the columns correspond to `K` subclasses; generically, the dimension is `J` by `K` `PsiSS` is supposed to be a vector of all zeros (perfect specificity in silver-standard measures).
- `ThetaBS/ThetaSS` True positive rates Θ for Bronze-Standard data and for Silver-Standard data. Dimension is `J` by `K` (can contain `NA` if the total number of causative agents measured by BrS or SS exceeds the measured causative agents in SS. For example, in PERCH study, nasopharyngeal polymerase chain reaction (NPPCR; bronze-standard) may target 30 distinct pathogens, but blood culture (BCX; silver-standard) may only target a subset of the 30, so we have to specify `NA` in `ThetaSS`for those pathogens not targeted by BCX).
- `Nu` the number of control subjects
- `Nd` the number of case subjects

`latent_samples` simulated latent status for all the subjects, for use in simulating SS measurements.

Value

a data frame with first column being case-control status (case at top) and columns of silver-standard measurements

See Also

Other internal simulation functions: `simulate_brs()`, `simulate_latent()`

`softmax` *softmax*

Description

uses logsumexp trick to prevent numerical overflow

Usage

```
softmax(x)
```

Arguments

`x`	a vector of numbers

Value

a vector of positive values that sum to one.

Examples

```
softmax2 <- function(x) exp(x) / sum(exp(x))
softmax(c(1, 2, 3) * 1000)  # NaN NaN NaN
softmax2(c(1, 2, 3) * 1000)  # 0 0 1
```

`subset_data_nplcm_by_index`
subset data from the output of `clean_perch_data()`

Description

It is particularly useful in simulating data from a regression model where one generates a case and control at a particular covariate value, and just choose a case or control to retain in the simulated data.

Usage

```
subset_data_nplcm_by_index(data_nplcm, index)
```

Arguments

data_nplcm	data for fitting nplcm; See nplcm()
index	a vector of indices indicating the observations you hope to subset; it will subset in all the sublists of data_nplcm

Value

a list with the requested data, in the order determined by 'index'

See Also

Other data operation functions: combine_data_nplcm(), merge_lists()

Examples

```
J = 3                           # number of causes
cause_list = c(LETTERS[1:J])    # cause list
K = 2                           # number of subclasses
lambda = c(1,0)                 # subclass weights for control group
eta = c(1,0)                    # subclass weights for case group

# setup parameters for the present individual:
set_parameter <- list(
 cause_list      = cause_list,
 etiology        = c(0.5,0.2,0.3), # only meaningful for cases
 pathogen_BrS    = LETTERS[1:J],
 pathogen_SS     = LETTERS[1:2],
 meas_nm         = list(MBS = c("MBS1"),MSS=c("MSS1")),
 Lambda          = lambda,         # for BrS
 Eta             = t(replicate(J,eta)),  # case subclass weight for BrS
 PsiBS           = cbind(c(0.15,0.3,0.35),
                         c(0.25,0.2,0.15)), # FPR
 PsiSS           = cbind(rep(0,J),rep(0,J)),
 ThetaBS         = cbind(c(0.95,0.9,0.85),    # TPR
                         c(0.95,0.9,0.85)),
 ThetaSS         = cbind(c(0.25,0.10),
                         c(0.25,0.10)),
 Nd      =     5,
 Nu      =     3
)
simu_out   <- simulate_nplcm(set_parameter)
out <- simu_out$data_nplcm
out
subset_data_nplcm_by_index(out,c(1,4,5))
subset_data_nplcm_by_index(out,2)
```

summarize_BrS *summarize bronze-standard data*

Description

summarize bronze-standard data

Usage

```
summarize_BrS(BrS_dat, Y)
```

Arguments

BrS_dat	bronze-standard data, which is usually data_nplcm$Mobs$MBS[[1]]
Y	A vector of case/control status: 1 for case; 0 for control

Value

a list of summaries for BrS data

See Also

Other exploratory data analysis functions: get_top_pattern(), plot_logORmat(), show_individual(), summarize_SS(), visualize_season()

Examples

```
data(data_nplcm_noreg)
summarize_BrS(data_nplcm_noreg$Mobs$MBS[[1]], data_nplcm_noreg$Y)
```

summarize_SS *silver-standard data summary*

Description

silver-standard data summary

Usage

```
summarize_SS(SS_dat, Y)
```

Arguments

SS_dat	a data frame of silver-standard data. It can usually be obtained by data_nplcm$Mobs$MSS[[1]], meaning the first SS measurement slice.
Y	a vector of case control status: 1 for case; 0 for control.

Value

a vector of number of positives

See Also

Other exploratory data analysis functions: `get_top_pattern()`, `plot_logORmat()`, `show_individual()`, `summarize_BrS()`, `visualize_season()`

Examples

```
data(data_nplcm_noreg)
summarize_BrS(data_nplcm_noreg$Mobs$MBS[[1]], data_nplcm_noreg$Y)
summarize_SS(data_nplcm_noreg$Mobs$MSS[[1]], data_nplcm_noreg$Y)
```

---

summary.nplcm *summary.nplcm summarizes the results from* `nplcm()`.

Description

`summary.nplcm` summarizes the results from `nplcm()`.

Usage

```
## S3 method for class 'nplcm'
summary(object, ...)
```

Arguments

`object`	Output from `nplcm()`. An object of class "nplcm"
`...`	Not used.

Value

see `print.nplcm()`

See Also

Other nplcm results: `print.nplcm()`, `print.summary.nplcm.no_reg()`, `print.summary.nplcm.reg_nest_strat()`, `print.summary.nplcm.reg_nest()`, `print.summary.nplcm.reg_nonest_strat()`, `print.summary.nplcm.reg_nones

symb2I *Convert names of pathogen/combinations into 0/1 coding*

Description

Convert names of pathogen/combinations into 0/1 coding

Usage

```
symb2I(pathogen_name, pathogen_list)
```

Arguments

pathogen_name	The allowed pathogen name (can be a combination of pathogens in "pathlist")
pathogen_list	The complete list of pathogen names

Value

A 1 by length(pathlist) matrix of binary code (usually for pathogen presence/absence)

Examples

```
symb2I("A",c("A","B","C"))
symb2I("A+B",c("A","B","C"))
symb2I("NoA",c("A","B","C"))
symb2I(c("A","B+C"),c("A","B","C")) # gives a 2 by 3 matrix.
```

sym_diff_month *get symmetric difference of months from two vector of R-format dates*

Description

sym_diff_month evaluates the symmetric difference between two sets of R-formatted date

Usage

```
sym_diff_month(Rdate1, Rdate2)
```

Arguments

Rdate1, Rdate2	R-formatted R dates. See as.Date()

Value

NULL if no difference; the set of different months otherwise.

s_date_Eti *Make Etiology design matrix for dates with R format.*

Description

s_date_Eti creates design matrices for etiology regressions;

Usage

```
s_date_Eti(Rdate, Y, basis = "ps", dof = ifelse(basis == "ncs", 5, 10), ...)
```

Arguments

Rdate	a vector of dates of R format
Y	Binary case/control status; 1 for case; 0 for controls
basis	ncs for natural cubic splines; ps for penalized-splines based on B-spline basis functions (NB: baker does not recommend setting ncs using this function; use splines::ns)
dof	Degree-of-freedom for the bases. For ncs basis, dof is the number of columns; For ps basis, the number of columns is dof if intercept=TRUE; dof-1 if FALSE.
...	Other arguments as in splines::bs()

Value

- Z_Eti design matrix for etiology regression on dates.

See Also

nplcm()

Examples

```
data("data_nplcm_reg_nest")
s_date_Eti(data_nplcm_reg_nest$X$DATE,data_nplcm_reg_nest$Y,basis='ps',dof=7)
```

s_date_FPR *Make false positive rate (FPR) design matrix for dates with R format.*

Description

s_date_FPR creates design matrices for FPR regressions;

Usage

```
s_date_FPR(Rdate, Y, basis = "ps", dof = 10, ...)
```

Arguments

Rdate	a vector of dates of R format
Y	Binary case/control status; 1 for case; 0 for controls
basis	"ps" for penalized-splines based on B-spline basis functions
dof	Degree-of-freedom for the bases.For "ps" basis, the number of columns is dof if intercept=TRUE; dof-1 if FALSE.
...	Other arguments as in splines::bs()

Value

Design matrix for FPR regression, with cases' rows on top of controls'.

See Also

nplcm()

Examples

```
data(data_nplcm_reg_nest)
s_date_FPR(data_nplcm_reg_nest$X$DATE,data_nplcm_reg_nest$Y,basis='ps',dof=7)
```

tsb *generate stick-breaking prior (truncated) from a vector of random probabilities*

Description

generate stick-breaking prior (truncated) from a vector of random probabilities

Usage

```
tsb(u)
```

Arguments

u	a vector of probabilities, with the last element 1.

Value

a vector of the same length as u; sum to 1.

Examples

```
oldpar <- graphics::par(mfrow=c(3,3),oma=c(0,1,5,0),
   mar=c(1,2,1,1))
for (iter in 1:9){
 u   <- c(rbeta(9,1,0.8),1)
 res <- tsb(u)
 barplot(res,ylim=c(0,1),main=paste0("Random Sample #", iter),ylab="Probability")
}
graphics::mtext("Truncated Stick-Breaking Dist. (10 segments)",3,
     outer=TRUE,cex=1.5,line=1.5)
par(oldpar)
```

---

unfactor *Convert factor to numeric without losing information on the label*

Description

Convert factor to numeric without losing information on the label

Usage

```
unfactor(f)
```

Arguments

f	A factor

Value

A numeric vector

Examples

```
unfactor(factor(c("1","3","3"),levels=c("1","3")))
# contrast this to:
as.numeric(factor(c("1","3","3"),levels=c("1","3")))
```

unique_cause *get unique causes, regardless of the actual order in combo*

Description

get unique causes, regardless of the actual order in combo

Usage

```
unique_cause(cause_vec)
```

Arguments

cause_vec	a vector of characters with potential combo repetitions written in scrambled orders separated by "+"

Value

a vector of characters with unique meanings for latent causes

Examples

```
x <- c("A","B","A","CC+DD","DD+CC","E+F+G","B")
unique_cause(x)
```

unique_month *Get unique month from Date*

Description

unique_month converts observed dates into unique months to help visualize sampled months

Usage

```
unique_month(Rdate)
```

Arguments

Rdate	standard date format in R

Value

a vector of characters with month-year, e.g., 4-2012.

`visualize_case_control_matrix` *Visualize matrix for a quantity measured on cases and controls (a single number)*

Description

Special to case-control visualization: upper right for cases, lower left for controls.

Usage

```
visualize_case_control_matrix(
  mat,
  dim_names = ncol(mat),
  cell_metrics = "",
  folding_line = TRUE,
  axes = FALSE,
  xlab = "",
  ylab = "",
  asp = 1,
  title = ""
)
```

Arguments

`mat`	matrix of values: upper for cases, lower for controls;
`dim_names`	names of the columns, from left to right. It is also the names of the rows, from bottom to top. Default is 1 through `ncol(mat)`;
`cell_metrics`	the meaning of number in every cell;
`folding_line`	Default is `TRUE` for adding dashed major diagonal line.
`axes`	plot axes; default is `FALSE`;
`xlab`	label for x-axis
`ylab`	label for y-axis
`asp`	aspect ratio; default is `1` to ensure square shape
`title`	text for the figure

Value

plotting function; no returned value.

`visualize_season` *visualize trend of pathogen observation rate for NPPCR data (both cases and controls)*

Description

visualize trend of pathogen observation rate for NPPCR data (both cases and controls)

Usage

```
visualize_season(data_nplcm, patho, slice = 1, slice_SS = 1)
```

Arguments

`data_nplcm`	Data set produced by `clean_perch_data()`
`patho`	the index of pathogen
`slice`	the slice of BrS data for visualization; default is 1.
`slice_SS`	the slice of SS data to add onto BrS plots; default is 1, usually representing blood culture measurements.

Details

This function shows observed positive rate for continuous covariates,e.g., enrollment date in PERCH application. Smoothing is done by penalized splines implemented by `mgcv` package. The penalized spline smoothing term is constructed by `mgcv::smooth.construct.ps.smooth.spec()`. The window size of the moving averages currently is set to 60 days.

Value

A figure with smoothed positive rate and confidence bands for cases and controls, respectively. The right margin shows marginal positive rates; all rates are only among the subset of case subjects who had non-missing responses for a measured agent (e.g., pathogen); similarly for controls.

See Also

Other exploratory data analysis functions: `get_top_pattern()`, `plot_logORmat()`, `show_individual()`, `summarize_BrS()`, `summarize_SS()`

write.model *function to write bugs model (copied from R2WinBUGS)*

Description

function to write bugs model (copied from R2WinBUGS)

Usage

```
write.model(model, con = "model.bug", digits = 5)
```

Arguments

model	R / S-PLUS function containing the BUGS model in the BUGS model language, for minor differences see Section Details.
con	passed to writeLines which actually writes the model file
digits	number of significant digits used for WinBUGS input, see formatC

Value

write text lines to a connection; same as writeLines()

write_model_NoReg *Write .bug model file for model without regression*

Description

write_model_NoReg automatically generates model file according to model_options

Usage

```
write_model_NoReg(
  k_subclass,
  Mobs,
  prior,
  cause_list,
  use_measurements,
  ppd = NULL,
  use_jags = FALSE
)
```

Arguments

k_subclass	the number of subclasses for the slices that require conditional dependence modeling (only applicable to BrS data); its length is of the same value as the number of BrS slices.
Mobs	measurement data in the form of data_nplcm
prior	prior specification from model_options
cause_list	a list of latent status names (crucial for building templates; see make_template())
use_measurements	"BrS", or "SS"
ppd	Default is NULL; set to TRUE for posterior predictive checking
use_jags	Default is FALSE; set to TRUE if want to use JAGS for model fitting.

Value

a long character string to be written into .bug file.

See Also

insert_bugfile_chunk_noreg_meas for inserting .bug file chunk for measurements (plug-and-play); insert_bugfile_chunk_noreg_etiology for inserting .bug file chunk for distribution of latent status (etiology).

Other model generation functions: write_model_Reg_Nest(), write_model_Reg_NoNest(), write_model_Reg_discrete

---

write_model_Reg_discrete_predictor_NoNest

Write .bug model file for regression model without nested subclasses

---

Description

write_model_Reg_discrete_predictor_NoNest automatically generates model file according to model_options

Usage

```
write_model_Reg_discrete_predictor_NoNest(
  Mobs,
  prior,
  cause_list,
  use_measurements,
  ppd = NULL,
  use_jags = FALSE
)
```

Arguments

Mobs	Measurement data in the form of data_nplcm
prior	Prior specification from model_options
cause_list	A list of latent status names (crucial for building templates; see make_template())
use_measurements	"BrS", or "SS"
ppd	Default is NULL; set to TRUE for posterior predictive checking
use_jags	Default is FALSE; set to TRUE if want to use JAGS for model fitting.

Value

a long character string to be written into .bug file.

See Also

insert_bugfile_chunk_noreg_meas for inserting .bug file chunk for measurements (plug-and-play); insert_bugfile_chunk_noreg_etiology for inserting .bug file chunk for distribution of latent status (etiology).

Other model generation functions: write_model_NoReg(), write_model_Reg_Nest(), write_model_Reg_NoNest()

---

write_model_Reg_Nest *Write* .bug *model file for regression model WITH nested subclasses*

Description

write_model_Reg_Nest automatically generates model file according to model_options; This is called within nplcm_fit_Reg_Nest.

Usage

```
write_model_Reg_Nest(
  Mobs,
  prior,
  cause_list,
  Eti_formula,
  FPR_formula,
  use_measurements,
  ppd = NULL,
  use_jags = FALSE
)
```

Arguments

Mobs	Measurement data in the form of data_nplcm
prior	Prior specification from model_options
cause_list	A list of latent status names (crucial for building templates; see make_template())
Eti_formula	Etiology regression formula; Check model_options$likelihood$Eti_formula.
FPR_formula	A list of FPR regression formula; check model_options$likelihood$FPR_formula
use_measurements	"BrS", or "SS"
ppd	Default is NULL; set to TRUE for posterior predictive checking
use_jags	Default is FALSE; set to TRUE if want to use JAGS for model fitting.

Value

a long character string to be written into .bug file.

See Also

insert_bugfile_chunk_noreg_meas for inserting .bug file chunk for measurements (plug-and-play.R); insert_bugfile_chunk_noreg_etiology for inserting .bug file chunk for distribution of latent status (etiology).

Other model generation functions: write_model_NoReg(), write_model_Reg_NoNest(), write_model_Reg_discrete_pr

---

write_model_Reg_NoNest

Write .bug model file for regression model without nested subclasses

---

Description

write_model_Reg_NoNest automatically generates model file according to model_options

Usage

```
write_model_Reg_NoNest(
  Mobs,
  prior,
  cause_list,
  Eti_formula,
  FPR_formula,
  use_measurements,
  ppd = NULL,
  use_jags = FALSE
)
```

Arguments

Mobs	Measurement data in the form of data_nplcm
prior	Prior specification from model_options
cause_list	A list of latent status names (crucial for building templates; see make_template())
Eti_formula	Etiology regression formula; Check model_options$likelihood$Eti_formula.
FPR_formula	A list of FPR regression formula; check model_options$likelihood$FPR_formula
use_measurements	"BrS", or "SS"
ppd	Default is NULL; set to TRUE for posterior predictive checking
use_jags	Default is FALSE; set to TRUE if want to use JAGS for model fitting.

Value

a long character string to be written into .bug file.

See Also

insert_bugfile_chunk_noreg_meas for inserting .bug file chunk for measurements (plug-and-play); insert_bugfile_chunk_noreg_etiology for inserting .bug file chunk for distribution of latent status (etiology).

Other model generation functions: write_model_NoReg(), write_model_Reg_Nest(), write_model_Reg_discrete_prec

Index