Package 'BayesianFROC'

June 11, 2019

Type Package

Title FROC Analysis by Bayesian Approaches

Version 0.1.3

Maintainer Issei Tsunoda <tsunoda.issei1111@gmail.com>

Description Provides new methods for the so-called Free-response Receiver Operating Characteristic (FROC) analysis. The ultimate aim of FROC analysis is to compare observer performances, which means comparing characteristics, such as area under the curve (AUC) or figure of merit (FOM). In this package, we only use the notion of AUC for modality comparison, where by ``modality'', we mean imaging methods such as Magnetic Resonance Imaging (MRI), Computed Tomography (CT), Positron Emission Tomography (PET), ..., etc. So there is a problem that which imaging method is better to detect lesions from shadows in radiographs. To solve modality comparison issues, this package provides new methods using hierarchical Bayesian models proposed by the author of this package. Using this package, one can obtain at least one conclusion that which imaging methods are better for finding lesions in radiographs with the case of your data. Fitting FROC statistical models is sometimes not so good, it can easily confirm by drawing FROC curves and comparing these curves and the points constructed by False Positive fractions (FPFs) and True Positive Fractions (TPFs), we can validate the goodness of fit intuitively. Such validation is also implemented by the Chi square goodness of fit statistics in the Bayesian context which means that the parameter is not deterministic, thus by integrating it with the posterior predictive measure, we get a desired value. To compare modalities (imaging methods: MRI, CT, PET, ... , etc), we evaluate AUCs for each modality. FROC is developed by Dev Chakraborty, his FROC model in his 1989 paper relies on the maximal likelihood methodology. The author modified and provided the alternative Bayesian FROC model. Strictly speaking, his model does not coincide with models in this package. In FROC context, we means by multiple reader and multiple case (MRMC) the case of the number of reader or modality is two or more. The MRMC data is available for functions of this package. I hope that medical researchers use not only the frequentist method but also alternative Bayesian methods. In medical research, many problems are considered under only frequentist methods, such as the notion of p-values. But p-value is sometimes misunderstood. Bayesian methods provide very simple, direct, intuitive answer for research questions. Combining frequentist methods with Bayesian methods, we can obtain more reliable answer for research questions. Please execute the following R scripts from the R (R studio) console, demo(demo_MRMC, package = ``BayesianFROC''); demo(demo_srsc, pack-

age = ``BayesianFROC''); demo(demo_stan, package = ``Bayesian-FROC''); demo(demo_drawcurves_srsc, package = ``Bayesian-FROC''); demo_Bayesian_FROC(); demo_Bayesian_FROC_without_pause(). References: Dev Chakraborty (1989) <doi:10.1118/1.596358> Maximum likelihood analysis of free - response receiver operating characteristic (FROC) data. Pre-print: Issei Tsunoda; Bayesian Models for free-response receiver operating characteristic analysis. See the vignettes for more details.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports knitr, readxl, xlsx, stats, graphics, tcltk, grDevices, ggplot2, methods, car, crayon, DiagrammeR, bridgesampling

Suggests rmarkdown, rstantools, openxlsx, hexbin, MASS

Depends rstan (>= 2.18.2), R (>= 3.5.0), Rcpp

NeedsCompilation yes

VignetteBuilder knitr

Collate 'Close_all_graphic_devices.R' 'ConfirmConvergence.R' 'DrawCurves.R' 'DrawCurves_MRMC.R' 'DrawCurves_MRMC_pairwise.R' 'DrawCurves_MRMC_pairwise_BlackWhite.R' 'DrawCurves_MRMC_pairwise_col.R' 'DrawCurves_srsc.R' 'Draw_an_area_of_AUC_for_srsc.R' 'Make_TeX_file_for_summary.R' 'Replicate_MRMC_data.R' 'Rprofile.R' 'Simulation_Based_Calibration.R' 'StatisticForANOVA.R' 'Test_Null_Hypothesis_that_all_modalities_are_same.R' 'apply_foo.R' 'check_rhat.R' 'clearWorkspace.R' 'color.R' 'convertFromJafroc.R' 'create_dataset.R' 'dark_theme.R' 'dataset_creator_new_version.R' 'demo_Bayesian_FROC.R' 'demo_Bayesian_FROC_without_pause.R' 'document_dataset.R' 'document_dataset_MRMC.R' 'draw_bi_normal.R' 'explanation_about_package_BayesianFROC.R' 'explanation_for_what_curves_are_drawn.R' 'extractAUC.R' 'extract_EAP_CI.R' 'extract_EAP_by_array.R' 'extract_estimates_MRMC.R' 'fffaaabbb.R' 'fit_Bayesian_FROC.R' 'fit_MRMC_test.R' 'fit_MRMC_versionTWO.R' 'fit_Null_hypothesis_model_to_.R' 'fit_srsc.R' 'fit_srsc_per_image_test.R' 'fit_srsc_per_lesion.R' 'foo_of_a_List_of_Arrays.R' 'give_name_srsc_data.R' 'initial_values_specification_for_stan_in_case_of_MRMC.R' 'install_imports.R' 'layout.R' 'metadata_srsc_per_image.R' 'metadata_to_DrawCurve_MRMC.R' 'metadata_to_fit_MRMC.R' 'stanfitExtended.R' 'methods.R' 'methods_print.R' 'modelComparison.R' 'p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit.R' 'packageStartupMessage.R' 'pairs_plot_if_divergent_transition_occurred.R' 'pause.R'

'plotFROC.R' 'showGraphicalModel.R' 'size_of_return_value.R'
'snippet_for_BayesianFROC.R' 'specify_priors.R'
'summarise_MRMC.R' 'summary_AUC_comparison_MRMC.R'
'summary_AUC_comparison_MRMC_with_crayon.R'
'summary_AUC_comparison_MRMC_without_crayon.R'
'summary_EAP_CI_srsc.R' 'the_row_number_of_logical_vector.R'
'validation.R' 'validation_MRMC.R' 'viewdata.R'
'viewdata_MRMC.R' 'viewdata_srsc.R' 'waic.R'

Author Issei Tsunoda [aut, cre]

Repository CRAN

Date/Publication 2019-06-11 13:10:03 UTC

R topics documented:

`check_rhat` *Diagnosis for convergence*

Description

This function evaluate your model's R hat statistics.

Usage

```
check_rhat(StanS4class)
```

Arguments

`StanS4class`	This is a return value of the function `rstan::stan()`, i.e., an object of the S4 class the so-called stanfit.

Value

TRUE of FALSE. If model converges then TRUE, and if not FALSE.

Author(s)

betanalpha. This is not my function. But I modified it.

References

Gelman A. \& Rubin, D.B. (1992). Inference from Iterative Simulation Using Multiple Sequences, Statistical Science, Volume 7, Number 4, 457-472.

`chi_square_goodness_of_fit`

The Goodness of Fit (Chi Square) Calculator

Description

Chi square goodness of fit statistics for each MCMC sample with a fixed dataset.

Our data is 2C categories, that is,

the number of hits :h[1], h[2], h[3],...,h[C] and

the number of false alarms: f[1],f[2], f[3],...,f[C].

Our model has C+2 parameters, that is,

the thresholds of the bi normal assumption z[1],z[2],z[3],...,z[C] and

the mean and standard deviation of the signal distribution.

So, the degree of freedom of this statistics is calculated by

2C-(C+2)-1 =C -3.

This differ from Chakraborty's result C-2. Why ?

Usage

```
chi_square_goodness_of_fit(StanS4class, dig = 3, h, f)
```

Arguments

StanS4class	An object of the inherited class of the S4 class stanfit. That is, the stanfitExtended S4 class defined in this package.
dig	An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,
h	The number of hits. The reason why we include this variable is to substitute the hits from the posterior predictive distributions. In famous Gelman's book, he explain how to use the test statistics in the Bayesian context. In this context I need to substitute the replication data from the posterior predictive distributions.
f	The number of false alarms. The reason why we include this variable is to substitute the false alarms from the posterior predictive distributions. In famous Gelman's book, he explain how to use the test statistics in the Bayesian context. In this context I need to substitute the replication data from the posterior predictive distributions.

Details

To calculate the chi square test statistics, the two quantities are needed, that is, data and parameter. In the classical chi square values, as the parameter, for example, MLE(maximal likelihood estimator) is chosen. In Bayesian sense, the parameter can be taken for all MCMC iterations, that is, parameter is not deterministic and we consider it is a random variable or samples from the posterior distribution. And such samples are obtained in the Hamiltonian Monte Carlo Simulation with the author's Bayesian Model. Thus we can calculate chi square values with MCMC samples.

Value

Chi squares for each MCMC samples. So, the return values is a vector whose length is number of MCMC iterations minus the warming up period. Of course if MCMC is not only one chain, then taking the summation over all chains.

Examples

```
#  Get the MCMC samples from a dataset.

      fit <- fit_Bayesian_FROC(BayesianFROC::dataList.Chakra.1,
                          ite = 1111,
                          summary =FALSE,
                          cha = 2)

#   The chi square discrepancies are calculated by the following code

        Chi.Square.for.each.MCMC.samples   <-   chi_square_goodness_of_fit(fit)

#  Get posterior mean of the chi square discrepancy.
```

```
                    m<-   mean(Chi.Square.for.each.MCMC.samples)

# Calculate the p-value for the posterior mean of the chi square discrepancy.

                     stats::pchisq(m,df=1)




# dottest
```

`chi_square_goodness_of_fit_from_input_all_param`
The Goodness of Fit (Chi Square) Calculator

Description

Chi square goodness of fit statistics for each MCMC sample with a fixed dataset.

Our data is 2C categories, that is,

the number of hits :h[1], h[2], h[3],...,h[C] and

the number of false alarms: f[1],f[2], f[3],...,f[C].

Our model has C+2 parameters, that is,

the thresholds of the bi normal assumption z[1],z[2],z[3],...,z[C] and

the mean and standard deviation of the signal distribution.

So, the degree of freedom of this statistics is calculated by

2C-(C+2)-1 =C -3.

This differ from Chakraborty's result C-2. Why ?

Usage

```
chi_square_goodness_of_fit_from_input_all_param(h, f, p, lambda, NL, NI,
  ModifiedPoisson = FALSE, dig = 3)
```

Arguments

h	The number of hits. The reason why we include this variable is to substitute the hits from the posterior predictive distributions. In famous Gelman's book, he explain how to use the test statistics in the Bayesian context. In this context I need to substitute the replication data from the posterior predictive distributions.

f	The number of false alarms. The reason why we include this variable is to substitute the false alarms from the posterior predictive distributions. In famous Gelman's book, he explain how to use the test statistics in the Bayesian context. In this context I need to substitute the replication data from the posterior predictive distributions.
p	Hit rate. A vector whose length is number of confidence levels.
lambda	False alarm rate. A vector whose length is number of confidence levels.
NL	Number of Lesions
NI	Number of Images
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
dig	An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

Details

To calculate the chi square test statistics, the two quantities are needed, that is, data and parameter. In the classical (frequentists) chi square values, as the estimates of parameter, for example, MLE (maximal likelihood estimator) is chosen. In Bayesian sense, the parameter can be taken for all MCMC iterations, that is, parameter is not deterministic and we consider it is a random variable or samples from the posterior distribution. And such samples are obtained in the Hamiltonian Monte Carlo Simulation with the author's Bayesian Model. Thus we can calculate chi square values with MCMC samples.

Value

Chi squares for each MCMC samples. So, the return values is a vector whose length is number of MCMC iterations minus the warming up period. Of course if MCMC is not only one chain, then taking the summation over all chains.

A number !! Not list nor dataframe nor vector !! Only A number represent the chi square for your input data.

Examples

```
#  Get the MCMC samples from a dataset.

       fit <- fit_Bayesian_FROC(BayesianFROC::dataList.Chakra.1,
                          ite = 1111,
                          summary =FALSE,
                          cha = 2)

#   The chi square discrepancies are calculated
```

```
#   by the following code by using each MCMC samples as a parameter.#


  NI          <-  fit@dataList$NI
  NL          <-  fit@dataList$NL
  f.observed  <-  fit@dataList$f
  h.observed  <-  fit@dataList$h
  C           <-  fit@dataList$C
  p <-  rstan::get_posterior_mean(fit, par=c("p"))
  lambda <- rstan::get_posterior_mean(fit, par=c("l"))


         Chi.Square <-   chi_square_goodness_of_fit_from_input_all_param(

                           h   =   h.observed,
                           f   =   f.observed,
                           p   =   p,
                      lambda   =   lambda,
                           NL  =   NL,
                           NI  =   NI
                                )

#  Get posterior mean of the chi square discrepancy.

                    Chi.Square

# Calculate the p-value for the posterior mean of the chi square discrepancy.

                     stats::pchisq(Chi.Square,df=1)







# dottest
```

---

clearWorkspace *Clear Work Space*

Description

If functions are masked in global environment, I use this. this function has no variables.

Usage

```
clearWorkspace()
```

Close_all_graphic_devices *Close the Graphic Device*

Description

Close the graphic device to avoid errors in R CMD check.

Usage

```
Close_all_graphic_devices()
```

Examples

```
 #    Open the graphic devices

  grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
  grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
  grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
  grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
  grDevices::dev.new();plot(stats::runif(100),stats::runif(100))
  grDevices::dev.new();plot(stats::runif(100),stats::runif(100))


#    Close the graphic device

      Close_all_graphic_devices()


#'# dottest
```

compare *model comparison*

Description

This is a model comparison.

Usage

```
compare(NI, ite = 1111)
```

Arguments

NI	images
ite	iteration

comparison *model comparison*

Description

This is a model comparison.

Usage

```
comparison(Number.of.variation.of.NL, Number.of.images, ite = 1111,
  DrawCurve = FALSE, dig = 3, e = 0)
```

Arguments

Number.of.variation.of.NL	Lesion
Number.of.images	images
ite	iteration
DrawCurve	Logical: TRUE of FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve =TRUE, if not then DrawCurve =FALSE. The reason why the author make this variable DrawCurve is that it takes long time in MRMC case to draw curves, and thus default value is FALSE.
dig	An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,
e	exp for false alarms

comparison_replicated_models_and_truth *Comparison of Model and Truth*

Description

Comparison of estimates from true distributions and the true model parameters

Usage

```
comparison_replicated_models_and_truth(initial.seed = 123,
  mu.truth = BayesianFROC::mu_truth, v.truth = BayesianFROC::v_truth,
  z.truth = BayesianFROC::z_truth, NI = 200, NL = 142,
  ModifiedPoisson = FALSE, No.of.replication = 2, summary = FALSE,
  ite = 1111)
```

Arguments

initial.seed	The variable initial.seed is used to replicate datasets. That is, if you take initial.seed = 1234, then the seed 1234, 1235, 1236, 1237, 1238, ... are for the first replication, the second replication, the third replication, ... If the n-th model does not converge for some n, then such model has no mean and thus the non-convergent models are omitted to calculate the errors.
mu.truth	array of dimension (M,Q). Mean of represents the signal distribution of bi-normal assumption.
v.truth	array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.
z.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
NI	Number of Images.
NL	Number of Lesions.
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
No.of.replication	For fixed number of lesions, images, the dataset of hits and false alarms are replicated, and the number of replicated datasets are specified by this variable.
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.
ite	An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

Value

list of estimates

ConfirmConvergence *Diagnosis For Convergence*

Description

This function evaluate convergence based on only the R hat statistics for a fitted model object.

Usage

```
ConfirmConvergence(StanS4class, summary = TRUE)
```

Arguments

StanS4class	An S4 object of the class stanfit. No need that it is the S4 class stanfitExtended.
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Value

Logical: TRUE of FALSE. If model converges then it is TRUE, and if not FALSE.

References

Gelman A. & Rubin, D.B. (1992). Inference from Iterative Simulation Using Multiple Sequences, Statistical Science, Volume 7, Number 4, 457-472.

See Also

check_rhat() made by Betanalpha.

Examples

```
#================The first example=====================================

            #((Primitive way)).
#1) Build the data for singler reader and single modality  case.

dat <- list(c=c(3,2,1),    #Confidence level
            h=c(97,32,31), #Number of hits for each confidence level
            f=c(1,14,74),  #Number of false alarms for each confidence level

            NL=259,       #Number of lesions
            NI=57,        #Number of images
            C=3)          #Number of confidence level

# where, c denotes Confidence level,
#        h denotes number of Hits for each confidence level,
#        f denotes number of False alarms for each confidence level,
#        NL denotes Number of Lesions,
#        NI denotes Number of Images,









#2) Fit the FROC model.
```

```
  #Since the above dataset  "dat" are single reader and single modality,
  #the following function fit the non hierarchical model.

          fit <-   BayesianFROC::fit_Bayesian_FROC(dat,ite=1111)


#  Where, the variable "ite" is the iteration of MCMC sampling.
#  More larger iteration is better.


#3.1) Confirm whether our estimates converge.



          ConfirmConvergence(fit)


# By the above R script,
# the diagnosis of convergence will be printed in the R (R-studio) console.
# Which diagnosis is based on only the R hat statistic.
# So someone might consider it is not sufficient, and use the
# Simulation based cariblation (SBC) or other things to diagnosis of the
# convergence or bias.
# Now, I try to implement SBC and it almost be made, but the randomized is not
# sufficient caused seed, so, I have to say it is under construction.
# I am tired,...



# It also return the logical vector whether the MCMC converge,
# if MCMC converges, then the return is TRUE and if not, then FALSE.

# This logical return value is used in this package development
# and the user should not be interested.

# The following was useful for programming.
#3.2) The return value is TRUE or FALSE.

     x <- ConfirmConvergence(fit)

#3.3) If you do not want to print the results, then

      x <- ConfirmConvergence(fit,summary=FALSE)



# 2019.05.21 Revised.
```

```
# dottest
```

`convertFromJafroc` *Convert from* `.xlsx` *file of* ***Jafroc*** *into* R *object*

Description

Convert an Excel file whose extension is **.xlsx** of Chakraborty's Jafroc formulation to an R object representing FROC data to which we will apply functions in this package such as `fit_Bayesian_FROC()`.

Convert

from .xlsx file of Jafroc

into R object

Usage

```
convertFromJafroc(No.of.Modalities, No.of.readers, No.of.confidence.levels)
```

Arguments

`No.of.Modalities`	Total number of modalities used your data.
`No.of.readers`	Total number of readers who are participants.
`No.of.confidence.levels`	The number of confidence levels.

Format

The `.xlsx` file of Jafroc must include three sheets named by ***TP, FP, Truth*** (other names never be permitted !!)

——————————————— *TP* ———————————————

A sheet named ***TP*** includes five columns named from the right hand side:

ReaderID, ModalityID, CaseID, LesionID, TP_Rating.

1) Note that the above word CaseID means the Image ID vectors indicating the ID of radiographs. That is "case = image = radiograph". 2) Note that the first row of `.xlsx` sheet devote for the names as follows:

An Example of a sheet named *TP* in a `.xlsx` file for the *Jafroc* software

ReaderID	ModalityID	CaseID	LesionID	TP_Rating.
1	1	1	1	5
1	2	2	1	4
1	3	4	1	5
1	1	8	1	3

1	2	8	2	4
1	3	9	1	4
1	1	9	2	3
1	2	9	3	5
1	3	11	1	3
2	1	1	1	4
2	2	4	1	4
2	3	5	1	4
2	1	8	1	1
2	2	8	2	2
2	3	8	3	2
2	1	10	1	3
2	2	10	2	2
2	3	11	1	2
:	:	:	:	:
:	:	:	:	:

FP

A sheet named **FP** includes four columns named from the right hand side: **ReaderID, ModalityID, CaseID, FP_Rating An Example of a sheet named FP in a** `.xlsx` **file for the Jafroc software**

ReaderID	ModalityID	CaseID	FP_Rating.
1	1	1	2
1	2	2	1
1	3	3	1
1	1	5	2
1	2	7	1
1	3	7	2
1	1	9	3
1	2	9	4
1	3	10	1
2	1	1	2
2	2	2	3
2	3	3	4
2	1	8	1
2	2	9	1
2	3	11	1
2	1	14	1
2	2	15	1
2	3	21	2
:	:	:	:
:	:	:	:

Truth

A sheet named **Truth** includes three columns named from the right hand side:**CaseID, LesionID, Weight** .

An Example of a sheet named Truth in a `.xlsx` **file for the Jafroc software**

CaseID	LesionID	Weight
1	1	0.3333...
1	2	0.3333...
1	3	0.3333...
2	1	0.5
2	2	0.5
3	1	1
4	1	0.25
4	2	0.25
4	3	0.25
4	4	0.25
5	1	0.5
5	2	0.5
6	1	0.3333...
6	2	0.3333...
6	3	0.3333...
7	1	0.3333...
7	2	0.3333...
7	3	0.3333...
8	1	0.25
8	2	0.25
8	3	0.25
8	4	0.25
:	:	:
:	:	:

Never change from these column names in any `.xlsx` file.

Note that the weght are used for each images influence a same effect on the esimates. If we consider the truth without weight, then the images including many targets (lesions) has very strong effect on the estimates. To avoid such bias, Jafroc uses weight. However, in this package, we do not use the information of weight. Since the theory of the author of this package did not consider such weight. In the future I have to include the notion of weight. Jafroc use the notion fo figure of metric as non parametric manner. So, it seems difficult to include it in the Bayesian model, since generally speaking, Bayesian methodology is parametric.

Details

The return values include the data list which are directly available to the main function `fit_Bayesian_FROC`. So, if user has data of Jafroc, then by running this function, user immediately can fit the author's Bayesian FROC model to the resulting R object.

The Jafroc software's format includes suspicious locations of readers and true locations. Such data is *redundant* for our Bayesian statistical models. So, we reduce the information of data to the

number of false positives and number of hits for each confidence levels by this function.

minor comment or regret The author said the Jafroc data is redundant, but I should say more informative, and it cause limitation of our model. So, our model start to fit a model to this reduced data from Jafroc. So, reduction will cause the non accuracy evaluation of observer performance. The future research I should start the Jafroc formulation to build model not the R object of this function.

Author(s)

Issei Tsunoda

References

Bayesian Models for Free-response Receiver Operating Characteristic Analysis

See Also

Rjafroc

Examples

```
# Aim

# step 0) Prepare Jafroc .xlsx file being contained in this package
# step 1) Convert the .xlxs file obtained in step 0)
# step 2) Fit model to the data object obtained in step 1)
```

```
# ------------------------ step 0)   ------------------------------------------------

# By an xlsx file named JAFROC_data.xlsx in the director "inst/extdata" of this package,
# we can reconstruct it  as follows:(If someone can obtain the Excel file
# from the path BayesianFROC/inst/extdata/JAFROC_data.xlsx, then the following code
# is not required to run. If searching bother you, then run the R script to obtain the
# Excel file.)
# I do not know how to users refer the JAFROC_data.xlsx in this package,
# so I provide it by making the same xlsx file as the JAFROC_data.xlsx.
```

```
# Note that JAFROC_data.xlsx cannot remove,
# if it is removed, then devtools::run_examples() make an error.

Truth <- readxl::read_excel(system.file("extdata",
 "JAFROC_data.xlsx",
  package="BayesianFROC"),
   sheet = "Truth")
View(Truth)

TP <- readxl::read_excel( system.file("extdata",
                                      "JAFROC_data.xlsx",
                                       package="BayesianFROC"),
                          sheet = "TP")
View(TP)

FP <- readxl::read_excel( system.file("extdata",
                                      "JAFROC_data.xlsx",
                                        package="BayesianFROC"),
                         sheet = "FP")
View(FP)




sample <- list(TP=TP,FP=FP,Truth=Truth)
openxlsx::write.xlsx(sample,"JafrocDatasetExample.xlsx")



# Now, we get excel file named "JafrocDatasetExample.xlsx", which is same as
# the JAFROC_data.xlsx.
# whose format is available in the Jafroc software developed by Chakraborty.
# If you use your data, your data must has same format of "JafrocDatasetExample.xlsx".
# Note that other excel data must comply with the above format.

# Note that if you have proper format excel file for our package,
# this process does not need.



#    (0) From the above, we obtain  "JafrocDatasetExample.xlsx"
#    which is the multiple reader and multiple modality dataset
#    for Jfroc analysis which is NOT implemented in our package,
#     but Chakraborty's software called Jafroc or the R package Rjafroc.
```

```
# ------------------------ step 1)   ------------------------------------------------

# (1) Using "JafrocDatasetExample.xlsx" as an example excel file,
# we run the function to convert the excel file from Jafroc format
# to our format:



     dataList <- convertFromJafroc(
                                  No.of.Modalities =5,
                                  No.of.readers    =4,
                                  No.of.confidence.levels = 5
                                    )



# In the variable, there is no xlsx file, since it is selected by interactive manner.
# So, please select the xlsx file obtained in step 0).

# ------------------------ step 2)   ------------------------------------------------

#  (2)   Now, we obtain a data list as the return value.
#        Using this list, we run the function "fit_Bayesian_FROC":



          fit  <-  fit_Bayesian_FROC(dataList )


            # dontrun
```

`create_dataList_MRMC` *MRMC: One Dataset Creator (No Replication)*

Description

From threshold, mean and S.D., data of Hits and False Alarm are created.

Usage

```
create_dataList_MRMC(z.truth = BayesianFROC::z_truth,
  mu.truth = BayesianFROC::mu_truth, v.truth = BayesianFROC::v_truth,
  NI = 57, NL = 142, ModifiedPoisson = FALSE, seed = 123,
  summary = FALSE)
```

Arguments

`z.truth`	Vector of dimension = C represents the thresholds of bi-normal assumption.

mu.truth	array of dimension (M,Q). Mean of represents the signal distribution of bi-normal assumption.
v.truth	array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.
NI	The number of images,
NL	The number of lesions,
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
seed	The seed for creating hits which are generated by the binomial distributions with the specified seed.
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Examples

```
dataList  <- create_dataList_MRMC()


fit_Bayesian_FROC(dataList,summary = FALSE)


#  In the example, we use a default values for true parameters for
#  the distributions. The reason why the default values exists is difficulty
#  for the user who is not familiar with FROC data nor FROC model parameter regions.
#   So, in the Bayesian model is merely model for FROC data.
#   If user input the abnormal data, then the model does not fit nor converge
#   in the Hamiltonian Monte Carlo simulations.
```

---

create_dataset *Create a dataset*

Description

This is an interactive creator of dataset for FROC data. Using this return value, you can build the FROC model for your data by applying the function fit_Bayesian_FROC in this package.

Usage

```
create_dataset()
```

Value

A return value, i.e. a list, which are used to build FROC models.

`Credible_Interval_for_curve` *Draw FROC curves which means credible interval.*

Description

Plot FROC curves based on two parameters a and b.

Usage

```
Credible_Interval_for_curve(dataList, StanS4class.fit_MRMC_versionTWO,
  mesh.for.drawing.curve = 10000, upper_x = upper_x,
  upper_y = upper_y, lower_y = lower_y)
```

Arguments

`dataList` This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
          h = c(41,22,14,8,1),
          f = c(1,2,5,11,13),
          NL = 124,
          NI = 63,
          C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

———— **Single reader and single modality (SRSC) case.** ————

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating

new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

---

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	ModalityID m	Confidence levels c	No. of false alarms f	No. of hits. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14
1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1

2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

`StanS4class.fit_MRMC_versionTWO`
: A return value of `fit_MRMC_versionTWO`.

`mesh.for.drawing.curve`
: An integer indicating number of dots drawing the curves, default =10000.

`upper_x`
: The frame size of drawing picture.

`upper_y`
: The frame size of drawing picture.

`lower_y`
: The frame size of drawing picture.

---

d *Single reader and Single modality data*

Description

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

Details

This data is same as `dataList.Chakra.1.with.explantation`. The author name it `d` for the sake of simplicity.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

`dataList.Chakra.1.with.explantation`

dark_theme *Dark Theme*

Description

Putting it before plotting, plot area become the dark theme.

Usage

```
dark_theme(type = 1)
```

Arguments

type	1,2,3,4,5,6,7,...

Details

a function which indicates what color are used when we use graphic devices.

Value

Nothing

Examples

```
dark_theme(1)

graphics::plot(c(1,2,3),c(1,2,3))


dark_theme(2)

graphics::plot(c(1,2,3),c(1,2,3))


# 2019.05.21 Revised.

# dottest
```

data.hier.ficitious *Multiple reader and Multiple modality data*

Description

This is used to build a hierarchical FROC model.

Details

This data is fictitious.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

The author' preprint

data.MultiReaderMultiModality *Multiple reader and Multiple modality data*

Description

This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.Web.

Details

This data appeared in Chakraborty's paper (1988)

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

data.SingleReaderSingleModality *Single reader and Single modality data*

Description

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

Details

This data appeared in Chakraborty's paper (1988)

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

dataList.Chakra.1 *Single reader and Single modality data*

Description

This is used to build a non-hierarchical FROC model.

Details

This data appeared in Chakraborty's paper (1988)

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

---

dataList.Chakra.1.with.explantation

Single reader and Single modality data

Description

A list, representing FROC data of hits and false alarms. This is used to build a non-hierarchical FROC model.

Format

This data list contains the following integer vectors `f, h` and integers `NL, NI, C`.

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

Example data:

A single reader and single modality case

NI=57,NL=259 In R console ->	**confidence level** c	**No. of false alarms** f	**No. of hits** h
definitely present	3	1	97
probably present	2	14	32
questionable	1	74	31

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, the confidence level means present (deseased, positive) case only. Since each reader marks their suspicous location only and it generate the hits and false alarms for his confidenc level representing that lesion is present. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Note that The format for the above example data must be made by the following forms:

```
dat <- list(
           h = c(97,   32,  31 ),
           f = c(1 ,  14, 74 ),
           NL = 259,
           NI = 57,
           C = 3)
```

And using this object `dat`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dat)`.

Details

Note that the maximal number of confidence level, denoted by `C`, are included, however, confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. Should write down your hits and false alarms vector so that it is compatible with this automatically created vector `c`.

This data appeared in Chakraborty's paper (1988). This dataset is same as `dataList.Chakra.1`. The difference between two dataset is only explanations for vectors. That is I attached the name for each vector by `names()`. I hope it help user for understanding what it is.

Author(s)

Issei Tsunoda `<tsunoda.issei1111@gmail.com >`

Source

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

dataList.Chakra.2 *Single reader and Single modality data*

Description

This is used to build a non-hierarchical FROC model.

Details

This data appeared in Chakraborty's paper (1988)

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

dataList.Chakra.3 *Single reader and Single modality data*

Description

This is used to build a non-hierarchical FROC model.

Details

This data appeared in Chakraborty's paper (1988)

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

dataList.Chakra.4 *Single reader and Single modality data*

Description

A list, representing FROC data. This is used to build a non-hierarchical FROC model.

Details

This data appeared in Chakraborty's paper (1988)

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

dataList.Chakra.Web *Multiple Reader and Multiple Modality Data*

Description

A list, representing FROC data of MRMC.

Details

This data is based on in Chakraborty's JAFROC software in which example data exists. The author have calculated hits and false alarms from this Jafroc example data.

Contents:

Multiple readers and Multiple modalities case, i.e., MRMC case

ModalityID q	**ReaderID** m	**Confidence levels** c	**No. of false alarms** f	**No. of hits**. h
1	1	5	0	50
1	1	4	4	30
1	1	3	20	11

1	1	2	29	5
1	1	1	21	1
1	2	5	0	15
1	2	4	0	29
1	2	3	6	29
1	2	2	15	1
1	2	1	22	0
1	3	5	1	39
1	3	4	15	31
1	3	3	18	8
1	3	2	31	10
1	3	1	19	3
1	4	5	1	10
1	4	4	2	8
1	4	3	4	25
1	4	2	16	45
1	4	1	17	14
2	1	5	1	52
2	1	4	1	25
2	1	3	21	13
2	1	2	24	4
2	1	1	23	1
2	2	5	1	27
2	2	4	1	28
2	2	3	5	29
2	2	2	30	1
2	2	1	40	0
2	3	5	2	53
2	3	4	19	29
2	3	3	31	13
2	3	2	56	2
2	3	1	42	4
2	4	5	2	9
2	4	4	0	16
2	4	3	2	22
2	4	2	30	43
2	4	1	32	14
3	1	5	1	43
3	1	4	7	29
3	1	3	13	11
3	1	2	28	6
3	1	1	19	0
3	2	5	0	18
3	2	4	1	29
3	2	3	7	21
3	2	2	7	0
3	2	1	31	0
3	3	5	7	43

3	3	4	15	29
3	3	3	28	6
3	3	2	41	7
3	3	1	9	1
3	4	5	0	10
3	4	4	2	14
3	4	3	5	19
3	4	2	24	32
3	4	1	31	23
4	1	5	1	61
4	1	4	4	19
4	1	3	18	12
4	1	2	21	9
4	1	1	23	3
4	2	5	1	16
4	2	4	1	29
4	2	3	0	34
4	2	2	11	1
4	2	1	35	0
4	3	5	6	52
4	3	4	14	29
4	3	3	37	10
4	3	2	36	4
4	3	1	18	3
4	4	5	0	10
4	4	4	2	16
4	4	3	4	23
4	4	2	18	43
4	4	1	25	15
5	1	5	0	35
5	1	4	2	29
5	1	3	19	18
5	1	2	23	9
5	1	1	18	0
5	2	5	0	17
5	2	4	2	27
5	2	3	6	24
5	2	2	10	0
5	2	1	30	0
5	3	5	2	34
5	3	4	25	33
5	3	3	40	7
5	3	2	29	13
5	3	1	24	2
5	4	5	1	12
5	4	4	1	16
5	4	3	4	21
5	4	2	24	35

5	4	1	32	15

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Example data of Jafroc software

See Also

dataList.Chakra.Web.orderd d

Examples

```
viewdata(BayesianFROC::dataList.Chakra.Web)
```

---

dataList.Chakra.Web.orderd

Multiple reader and Multiple modality data

Description

This is used to build a hierarchical FROC model.

Details

This data appeared in Chakraborty's JAFROC. I have ordered so that the modality ID means the order of AUC. For example modality ID = 1 means its AUC is the highest. modalityID = 2 means its AUC is the second.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

dataList.High *Single reader and Single modality data*

Description

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

Details

This data is fictitious.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

dataList.high.ability *Single reader and Single modality data*

Description

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

Details

This data is fictitious.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

`dataList.Low` *Single reader and Single modality data*

Description

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

Details

This data is fictitious.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

`dataList.low.ability` *Single reader and Single modality data*

Description

A list, representing FROC data. This is used to build a hierarchical FROC model. This data is same as dataList.Chakra.1.

Details

This data is fictitious.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Maximum likelihood analysis of free-response receiver operating characteristic (FROC) data, Dev P. Chakraborty.

See Also

dataList.Chakra.1.with.explantation

---

dataList.one.modality *Multiple reader and one modality data for fit_MRMC_versionTWO*

Description

This is used to build a hierarchical FROC model.

Details

This data contains only one modality. If see = 12, then the model has converged.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Nothing in 2018

---

dataset_creator_new_version *Create a Dataset (version 2) Interactively*

Description

This is an interactive creator of dataset for FROC data.

Usage

```
dataset_creator_new_version()
```

Details

This provide the intaractive making of FROC dataset by using table to summarize hits and false alarm data.

Using this return value, you can build the FROC model for your data by applying the function fit_Bayesian_FROC() in this package.

Should carefully for the order of confidence levels.

Value

A list representing FROC data, to build FROC fitted model object by `fit_Bayesian_FROC()`.

---

dd *Multiple Reader and Multiple Modality Data*

Description

A list, representing FROC data of MRMC.

Details

This data is based on in Chakraborty's JAFROC software in which example data exists. The author have calculated hits and false alarms from this Jafroc example data.

Contents:

Multiple readers and multiple modalities case, i.e., MRMC case

ModalityID q	**ReaderID** m	**Confidence levels** c	**No. of false alarms** f	**No. of hits**. h
1	1	5	0	50
1	1	4	4	30
1	1	3	20	11
1	1	2	29	5
1	1	1	21	1
1	2	5	0	15
1	2	4	0	29
1	2	3	6	29
1	2	2	15	1
1	2	1	22	0
1	3	5	1	39
1	3	4	15	31
1	3	3	18	8
1	3	2	31	10
1	3	1	19	3
1	4	5	1	10
1	4	4	2	8
1	4	3	4	25
1	4	2	16	45
1	4	1	17	14
2	1	5	1	52
2	1	4	1	25
2	1	3	21	13
2	1	2	24	4
2	1	1	23	1

2	2	5	1	27
2	2	4	1	28
2	2	3	5	29
2	2	2	30	1
2	2	1	40	0
2	3	5	2	53
2	3	4	19	29
2	3	3	31	13
2	3	2	56	2
2	3	1	42	4
2	4	5	2	9
2	4	4	0	16
2	4	3	2	22
2	4	2	30	43
2	4	1	32	14
3	1	5	1	43
3	1	4	7	29
3	1	3	13	11
3	1	2	28	6
3	1	1	19	0
3	2	5	0	18
3	2	4	1	29
3	2	3	7	21
3	2	2	7	0
3	2	1	31	0
3	3	5	7	43
3	3	4	15	29
3	3	3	28	6
3	3	2	41	7
3	3	1	9	1
3	4	5	0	10
3	4	4	2	14
3	4	3	5	19
3	4	2	24	32
3	4	1	31	23
4	1	5	1	61
4	1	4	4	19
4	1	3	18	12
4	1	2	21	9
4	1	1	23	3
4	2	5	1	16
4	2	4	1	29
4	2	3	0	34
4	2	2	11	1
4	2	1	35	0
4	3	5	6	52
4	3	4	14	29
4	3	3	37	10

4	3	2	36	4
4	3	1	18	3
4	4	5	0	10
4	4	4	2	16
4	4	3	4	23
4	4	2	18	43
4	4	1	25	15
5	1	5	0	35
5	1	4	2	29
5	1	3	19	18
5	1	2	23	9
5	1	1	18	0
5	2	5	0	17
5	2	4	2	27
5	2	3	6	24
5	2	2	10	0
5	2	1	30	0
5	3	5	2	34
5	3	4	25	33
5	3	3	40	7
5	3	2	29	13
5	3	1	24	2
5	4	5	1	12
5	4	4	1	16
5	4	3	4	21
5	4	2	24	35
5	4	1	32	15

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

References

Example data of Jafroc software

See Also

dataList.Chakra.Web dataList.Chakra.Web.orderd d

Examples

```
viewdata(BayesianFROC::dd)
```

demo_Bayesian_FROC *demonstration*

Description

demonstration

Usage

```
demo_Bayesian_FROC()
```

Details

The author often forget the R script for execute the demos or bother to write the code to execute demo, thus I made this.

Value

none

Examples

```
 demo_Bayesian_FROC()

# 2019.05.21 Revised.
# dottest
```

demo_Bayesian_FROC_without_pause *demonstration without pause*

Description

demonstration without pause

Usage

```
demo_Bayesian_FROC_without_pause()
```

Value

none

Examples

```
demo_Bayesian_FROC_without_pause()

# dottest
```

`draw.CFP.CTP.from.dataList` *Plot the pairs of CFPs and CTPs*

Description

It plot the emipirical FROC curves (not depicted the line).

Usage

```
draw.CFP.CTP.from.dataList(dataList, ModifiedPoisson = FALSE,
  new.imaging.device = TRUE)
```

Arguments

`dataList` This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
           h = c(41,22,14,8,1),
           f = c(1,2,5,11,13),
           NL = 124,
           NI = 63,
           C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

———— Single reader and single modality (SRSC) case. ————

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

- `f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- `h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- `NL` A positive integer, representing Number of Lesions.
- `NI` A positive integer, representing Number of Images.
- `C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this

dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

---

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	**ModalityID** m	**Confidence levels** c	**No. of false alarms** f	**No. of hits**. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14

1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

`ModifiedPoisson`

Logical, that is `TRUE` or `FALSE`. If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm are `per lesion`, and if `ModifiedPoisson = FALSE`, then Poisson rate of false alarm are `per image`. To know detail, refer the author's paper in which I explained per image and per lesion.

`new.imaging.device`

Logical: `TRUE` of `FALSE`. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

Value

CFPs and CTPs

Examples

```
draw.CFP.CTP.from.dataList(dataList.Chakra.1)
```

`DrawCurves` *Draw the FROC curves*

Description

The function makes a plot of the FROC curve or the AFROC curve.

Usage

```
DrawCurves(StanS4class, modalityID, readerID, title = TRUE,
  indexCFPCTP = FALSE, upper_x, upper_y, new.imaging.device = TRUE,
  Colour = TRUE, DrawFROCcurve = TRUE, DrawAFROCcurve = FALSE,
  DrawCFPCTP = TRUE, Draw.Flexible.upper_y = TRUE,
  Draw.Flexible.lower_y = TRUE)
```

Arguments

StanS4class	fitted model object, from an output of rstan::sampling().
modalityID	A positive integer vector indicating modalityID. If it is not given, then the first modality is chosen.
readerID	A positive integer vector indicating readerID. If it is not given, then the first reader is chosen.
title	logical: TRUE or FALSE. If TRUE(default), then title of curves are drawn.
indexCFPCTP	TRUE of FALSE. If TRUE, then the cumulative false and hits are specified with its confidence level.
upper_x	This is a upper bound for the axis of the horisontal coordinate of FROC curve.
upper_y	This is a upper bound for the axis of the vertical coordinate of FROC curve.
new.imaging.device	Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.
Colour	logical: TRUE or FALSE. whether Colour of curves is dark theme or not.
DrawFROCcurve	logical: TRUE or FALSE. Whether or not FROC curves are shown.
DrawAFROCcurve	logical: TRUE or FALSE. Whether or not AFROC curves are shown.
DrawCFPCTP	logical: TRUE or FALSE. Whether or not CFPCTP points are shown.
Draw.Flexible.upper_y	logical: TRUE or FALSE. Whether or not the upper bounds of vertical axis are determined automatically.
Draw.Flexible.lower_y	logical: TRUE or FALSE. Whether or not the lower bounds of vertical axis are determined automatically.

Details

The function makes a plot of the FROC curves and AFROC curves for user's specified modality and user's specified reader. Using this function **repeatedly**, we can draw the different reader and modality in a **same** plane simultaneously. So, we can visualize the difference of modality (reader).

——— To read the tables in Stan S4 class ————————————-

* The AUC denoted by AA[modalityID , readerID] are shown by the function print().

* The column of 2.5% and 97.5% means the lower and upper bounds of the 95

* For example, AA[2,3] means the AUC of the 2 nd modality and the 3 rd reader.

Examples

```
#================The first example=====================================

#1) Build the S4 class object by the following:


  fit <- fit_Bayesian_FROC(
```

```
  BayesianFROC::dataList.Chakra.Web,    # data to which fit the model
                           ite=1111     # iteration of MCMC, very few, should be more.
                           )


# The object "fit" is an S4 class object
# whose S4 class is an inherited class from stanfit in the rstan package.

#<<Minor comments>>
#Note that return value "fit" is not an stanfit S4 object generated by rstan::stan(),
#but some inherited S4 class object which is an S4 object of
# some inherited S4 class from the stanfit class.




#2) Now, we obtain the S4 class object named "fit".
  # Using this S4 class object, we draw the curves by:



          DrawCurves(fit,modality = 1,reader = 4)

#From this code, FROC curve for the first modality and fourth reader is drawn.







#3) By changing, e.g., the modality,
   #we can draw the curves for different  modalities.
   #This shows the comparison of modalites.
   # In the following R script,
   #  the first draw curve for the 2 nd modality and the fourth reader,
   # and the second R script draw for the 3rd modality and the 4 th reader,
   # respectively.



            DrawCurves(fit,modality = 2,reader = 4)
            DrawCurves(fit,modality = 3,reader = 4)



# Curves are overwritten for the sake of comparison.
# When comparing modalities fitted by the hierarchical Bayesian Model to the same data,
# the upper FROC curve or AFROC curve, the better the AUC.
```

```
#4) By applying the function with respect to different modalities
#   in this manner, we can draw  AFROC (FROC) curves in the same plain.



#5) If you want to draw the FROC curves
#for reader ID =1,2,3,4 and modality ID =1,2, then the code is as follows;

                 DrawCurves(
                           fit,
                           modalityID = c(1,2,3,4),
                           readerID = c(1,2)
                           )
# Each color of curves corresponds to the modality ID.
# So, even if curves are different readers and same modality, then color is same.







#6) If you want to see only data points, then by DrawFROCcurve = F, it will be done.

DrawCurves(fit,
           DrawCFPCTP = TRUE,      # This implies data points are ploted.
           DrawFROCcurve = FALSE,  # From this, the curves are not drawn.
           modalityID = c(1,2,3,4),
           readerID  = c(1)
           )







#7) If you use the plot in submission and it is not allowed to use color, then
#   by Colour =F, you can get black and white plots, e.g.,


DrawCurves(fit,
          DrawCFPCTP = TRUE,
          DrawFROCcurve = TRUE,
          modalityID = c(1,2,3,4),
          readerID  = c(1),
          Colour = FALSE    # From this, you can get plots without colors.
```

```
          )




#'#8)  For AFROC, use DrawAFROCcurve = T

DrawCurves(fit,
          DrawFROCcurve = FALSE,
          DrawAFROCcurve = TRUE,
          modalityID = c(1,2,3,4),
          readerID  = c(1))
#================The Second Example=====================================
#This function is available in the case of single reader and single modality.
#The reason why the maintainer separate the fitting and drawing curves is, in MRMC case,
#It tooks a time to drawing, but in the single reader and single modality case, drawing
# the curve is very fast, so in fitting process the curves are also depicted, however
# by this function user can draw the FROC curves.
#First, we prepare the data endowed with this package.

        dat  <- get(data("dataList.Chakra.1"))




#Second, we run the stan funtion
#with data named "dat"  and the author's Bayesian model.


         fit <-  fit_srsc_per_lesion(dat)

# Drawing the curves by

              DrawCurves(fit)







#      Close the graphic device to avoid errors in R CMD check.

         Close_all_graphic_devices()
# dottest
```

DrawCurves_MRMC *Draw the FROC curves for all modalities and readers*

Description

Draw the FROC curves and AFROC curves for all modalities and readers, if many modalities and readers exists, then so very confused plots will be drawn.

Usage

```
DrawCurves_MRMC(StanS4class)
```

Arguments

StanS4class	Fitted model object created by `fit_Bayesian_FROC()`. The class of this object is stanfitExtended and not the class stanfit.

Examples

```
  fit <- fit_Bayesian_FROC(
                           dataList.Chakra.Web.orderd,
                           ite = 1111,
                           summary =FALSE
                           )

              DrawCurves_MRMC(fit)



# 2019.05.21 Revised.

# dottest
```

DrawCurves_MRMC_pairwise *Draw the FROC curves with Colour*

Description

Draw *FROC curves* and *AFROC curves* for user's specified modalities and user's specified readers. Using this function **repeatedly**, we can draw the different reader and modality in a **same** plane simultaneously.

Usage

```
DrawCurves_MRMC_pairwise(StanS4class, modalityID, readerID,
  Colour = TRUE, DrawFROCcurve = TRUE, DrawAFROCcurve = FALSE,
  DrawCFPCTP = TRUE, Draw.Flexible.upper_y = TRUE,
  Draw.Flexible.lower_y = TRUE, new.imaging.device = TRUE)
```

Arguments

StanS4class	This is an R object of class stanfitExtended inherited from the S4 class stanfit.
modalityID	This is a vector indicating modalityID whose component is natural namber.
readerID	This is a vector indicating readerID whose component is natural namber.
Colour	Logical, that is TRUE or FALSE. Whether plot of curves are with dark theme. Default is TRUE indicating dark theme.
DrawFROCcurve	Logical: TRUE of FALSE. Whether the FROC curve is to be drawn.
DrawAFROCcurve	Logical: TRUE of FALSE. Whether the AFROC curve is to be drawn.
DrawCFPCTP	Logical: TRUE of FALSE. Whether the CFPCTP points are to be drawn.
Draw.Flexible.upper_y	Logical, that is TRUE or FALSE. Whether or not the upper bounds of vertical axis are determined automatically.
Draw.Flexible.lower_y	Logical, that is TRUE or FALSE. Whether or not the lower bounds of vertical axis are determined automatically.
new.imaging.device	Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

Details

By drawing different modality FROC curves in the same plane, we can compare the modality. E.g., if some modality FROC curve is upper then other modality curves, then we may say that the upper modality is better observer performance, i.e., higher AUC.

Examples

```
#1) Build the S4 class object by the following:


  fit <- fit_Bayesian_FROC(dataList.Chakra.Web)


# The object "fit" is an S4 class object
# whose S4 class name is stanfit in the rstan package.
```

```
#<<Minor comments>>
#Note that return value "fit" is not an stanfit S4 object generated by rstan::stan(),
#but some inherited S4 class object which is an S4 object of
# some inherited S4 class from stanfit class. So, we can consider it as an object of
#an S4 class of rstan::stan().
#2) Now, we obtain the S4 class object named "fit".
  # Using this S4 class object, we draw the curves by:



 DrawCurves_MRMC_pairwise(fit,
                          modality = 1,
                          reader = 4
                          )



#3) By changing, e.g., the modality,
   #we can draw the curves for different  modalities.
   #This shows the comparison of modalites.



 DrawCurves_MRMC_pairwise(fit,
                          modality = 2,
                          reader = 4
                          )

 DrawCurves_MRMC_pairwise(fit,
                          modality = 3,
                          reader = 4
                          )


#4) By repeat the running with respect to different modalities
#   in this manner, we can draw  AFROC (FROC) curves.



#5) If you want to draw the FROC curves
#for reader ID =1,2,3,4 and modality ID =1,2, then the code is as follows;

DrawCurves_MRMC_pairwise(
                           fit,
                           modalityID = c(1,2,3,4),
                           readerID = c(1,2)
                           )
# Each color of curves corresponds to the modality ID.
# So, even if curves are different readers and same modality, then color is same.
```

```
#   Close the graphic device
    Close_all_graphic_devices()
 # dottest
```

DrawCurves_MRMC_pairwise_BlackWhite

Draw the FROC curves without colour

Description

Plot curves without any color (dark theme), that is, black and white (white backgroud with black curves). Draw FROC curves and AFROC curves for user's specified modality and user's specified reader. Using this function **repeatedly**, we can draw or compare the different reader and modality in a **same** plane simultaneously. So, we can visualize the difference of modality (reader).

Usage

```
DrawCurves_MRMC_pairwise_BlackWhite(StanS4class, modalityID, readerID,
  new.imaging.device = TRUE, DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE, DrawCFPCTP = TRUE,
  Draw.Flexible.upper_y = TRUE, Draw.Flexible.lower_y = TRUE)
```

Arguments

StanS4class	This is an R object of class stanfitExtended inherited from the S4 class stanfit.
modalityID	This is a vector indicating modalityID whose component is natural namber.
readerID	This is a vector indicating readerID whose component is natural namber.
new.imaging.device	Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.
DrawFROCcurve	Logical: TRUE of FALSE. Whether the FROC curve is to be drawn.
DrawAFROCcurve	Logical: TRUE of FALSE. Whether the AFROC curve is to be drawn.
DrawCFPCTP	Logical: TRUE of FALSE. Whether the CFPCTP points are to be drawn.
Draw.Flexible.upper_y	Logical, that is TRUE or FALSE. Whether or not the upper bounds of vertical axis are determined automatically.

`Draw.Flexible.lower_y`
: Logical, that is `TRUE` or `FALSE`. Whether or not the lower bounds of vertical axis are determined automatically.

`DrawCurves_MRMC_pairwise_col` *Draw the FROC curves with Colour*

Description

Draw an FROC curves and an AFROC curves for user's specified modality and user's specified reader. Using this function **repeatedly**, we can draw the different reader and modality in a **same** plane simultaneously. So, we can visualize the difference of modality (reader).

——— To read the tables in Stan S4 class ————————————-

* The AUC denoted by AA[modalityID , readerID] are shown.

* The column of 2.5% and 97.5% means the lower and upper bounds of the 95

* For example, AA[2,3] means the AUC of the 2 nd modality and the 3 rd reader.

Usage

```
DrawCurves_MRMC_pairwise_col(StanS4class, modalityID, readerID,
  new.imaging.device = TRUE, DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE, DrawCFPCTP = TRUE,
  Draw.Flexible.upper_y = TRUE, Draw.Flexible.lower_y = TRUE)
```

Arguments

`StanS4class`
: This is an R object of class `stanfitExtended` inherited from the S4 class `stanfit`.

`modalityID`
: This is a vector indicating modalityID whose component is natural namber.

`readerID`
: This is a vector indicating readerID whose component is natural namber.

`new.imaging.device`
: Logical: `TRUE` of `FALSE`. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

`DrawFROCcurve`
: Logical: `TRUE` of `FALSE`. Whether the FROC curve is to be drawn.

`DrawAFROCcurve`
: Logical: `TRUE` of `FALSE`. Whether the AFROC curve is to be drawn.

`DrawCFPCTP`
: Logical: `TRUE` of `FALSE`. Whether the CFPCTP points are to be drawn.

`Draw.Flexible.upper_y`
: Logical, that is `TRUE` or `FALSE`. Whether or not the upper bounds of vertical axis are determined automatically.

`Draw.Flexible.lower_y`
: Logical, that is `TRUE` or `FALSE`. Whether or not the lower bounds of vertical axis are determined automatically.

DrawCurves_srsc *Draw the FROC curves*

Description

Draw an FROC curves and an AFROC curves.

Usage

```
DrawCurves_srsc(StanS4class, title = TRUE, indexCFPCTP = FALSE,
  upper_x, upper_y, new.imaging.device = TRUE, Drawcol = TRUE,
  DrawFROCcurve = TRUE, DrawAFROCcurve = FALSE, DrawCFPCTP = TRUE,
  Draw.inner.circle.for.CFPCTPs = TRUE)
```

Arguments

StanS4class	fitted model object, from an output of `rstan::sampling()`.
title	logical: TRUE or FALSE. If TRUE(default), then title of curves are drawn.
indexCFPCTP	TRUE of FALSE. If TRUE, then the cumulative false and hits are specified with its confidence level.
upper_x	This is a upper bound for the axis of the horisontal coordinate of FROC curve.
upper_y	This is a upper bound for the axis of the vertical coordinate of FROC curve.
new.imaging.device	Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.
Drawcol	Logical: TRUE of FALSE. Whether the (A)FROC curve is to be drawn by using color of dark theme. The default value is a TRUE.
DrawFROCcurve	logical: TRUE or FALSE. Whether or not FROC curves are shown.
DrawAFROCcurve	logical: TRUE or FALSE. Whether or not AFROC curves are shown.
DrawCFPCTP	logical: TRUE or FALSE. Whether or not CFPCTP points are shown.
Draw.inner.circle.for.CFPCTPs	TRUE or FALSE. If true, then to plot the cumulative false positives and true positives the plot points is depicted by two way, one is a large circle and one is a small circle. By see the small circle, user can see the more precise position of these points.

Draw_an_area_of_AUC_for_srsc *Draw a Region of the area under the AFROC curve*

Description

Draw a Region of the area under the AFROC curve

Usage

```
Draw_an_area_of_AUC_for_srsc(StanS4class)
```

Arguments

StanS4class	This is an R object of class stanfitExtended inherited from the S4 class stanfit.

Value

None

Examples

```
fit <- fit_Bayesian_FROC(dataList.Chakra.1)

Draw_an_area_of_AUC_for_srsc(fit)


# dottest
```

Draw_a_prior_sample *Draw One Sample from Prior*

Description

Draw One Sample from Prior

Usage

```
Draw_a_prior_sample(sd = 5, C = 5,
  seed.for.drawing.a.prior.sample = 1111)
```

Arguments

sd	Standard deviation of priors. Very large number.
C	No. of Confidence level
seed.for.drawing.a.prior.sample	seed

Value

w, v, m, dz, z

Examples

```
    Draw.a.prior.sample <- Draw_a_prior_sample()

# dottest
```

Draw_a_simulated_data_set

Draw a simulated dataset from model distributions with specified parameters from priors

---

Description

Draw a simulated dataset from model distributions with specified parameters from priors

Usage

```
Draw_a_simulated_data_set(sd = 5, C = 5,
  seed.for.drawing.a.prior.sample = 1111, fun = stats::var, NI = 259,
  NL = 259, initial.seed.for.drawing.a.data = 1234,
  ModifiedPoisson = FALSE, ite = 1111)
```

Arguments

sd	Standard Deviation of priors
C	No. of Confidence levels
seed.for.drawing.a.prior.sample	seed
fun	An one dimensional real valued function defined on the parameter space. This is used in the definition of the rank statistics. Generally speaking, the element of the parameter space is a vector, so the function should be defined on vectors. In my model parameter is mean, standard deviation, C thresholds of the latent Gaussian, so this function should be defined on the C+2 dimensional Euclidean space.

`NI` No. of images

`NL` No. of Lesions

`initial.seed.for.drawing.a.data`
seed

`ModifiedPoisson`
Logical, that is `TRUE` or `FALSE`. If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm are `per lesion`, and if `ModifiedPoisson = FALSE`, then Poisson rate of false alarm are `per image`. To know detail, refer the author's paper in which I explained per image and per lesion.

`ite` An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

Value

One simulated dataset

Examples

```
   one.dataList  <-  Draw_a_simulated_data_set()

# dottest
```

`Draw_a_simulated_data_set_and_Draw_posterior_samples`
Draw a dataset and MCMC samples

Description

Draw a dataset and MCMC samples

Usage

```
Draw_a_simulated_data_set_and_Draw_posterior_samples(sd = 5, C = 5,
  seed.for.drawing.a.prior.sample = 1111, fun = stats::var, NI = 259,
  NL = 259, initial.seed.for.drawing.a.data = 1234,
  ModifiedPoisson = FALSE, PreciseLogLikelihood = TRUE, ite = 1111,
  DrawCurve = FALSE)
```

Arguments

sd	Standard Deviation of priors
C	No. of Confidence levels
seed.for.drawing.a.prior.sample	seed
fun	An one dimensional real valued function defined on the parameter space. This is used in the definition of the rank statistics. Generally speaking, the element of the parameter space is a vector, so the function should be defined on vectors. In my model parameter is mean, standard deviation, C thresholds of the latent Gaussian, so this function should be defined on the C+2 dimensional Euclidean space.
NI	No. of images
NL	No. of Lesions
initial.seed.for.drawing.a.data	seed
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
PreciseLogLikelihood	Logical, that is TRUE or FALSE. If PreciseLogLikelihood = TRUE(default), then Stan calculates the precise log likelihood with target formulation. If PreciseLogLikelihood = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warnning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.
ite	An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.
DrawCurve	Logical: TRUE of FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve =TRUE, if not then DrawCurve =FALSE. The reason why the author make this variable DrawCurve is that it takes long time in MRMC case to draw curves, and thus default value is FALSE.

Value

Draw.a.prior.sample The Return value of Draw_a_prior_sample

A dataList and an object of the stanfit S4 class with respect to the dataList

See Also

hits_false_alarms_creator_from_thresholds

Examples

```
#  Draw a curve for various seeds and various number of confidence levels.
#  Changing the seed, we can draw a parameter from priors and using this sample,
#  we can draw the datasets from our model whose parameters are
#  the priors samples.


#     1. draw a prior sample,
#     2. draw a data from the model at the prior sample drawn in step 1,
#     3. draw a posterior sample for the data drawn in step 2.



 Draw_a_simulated_data_set_and_Draw_posterior_samples(
 seed.for.drawing.a.prior.sample = 1234,
 C=8)

   Draw_a_simulated_data_set_and_Draw_posterior_samples(
 seed.for.drawing.a.prior.sample = 12345,
 C=7)

    Draw_a_simulated_data_set_and_Draw_posterior_samples(
 seed.for.drawing.a.prior.sample = 123456,
 C=6)


    Draw_a_simulated_data_set_and_Draw_posterior_samples(
 seed.for.drawing.a.prior.sample = 1234567,
 C=5)


# dottest
```

`draw_bi_normal` *Visualization of Latent Gaussians*

Description

Visualization of Latent Gaussians

Usage

```
draw_bi_normal(StanS4class, dark_theme = TRUE, dig = 3, mesh = 1000)
```

Arguments

StanS4class	This is an R object of class stanfitExtended inherited from the S4 class stanfit.
dark_theme	TRUE or FALSE
dig	Digit for print of the outputs in the R console.
mesh	Mesh for painting the area

Value

Information of Latent Gaussians, such as mean and S.D. of the signal distributions and thresholds.

Examples

```
#            ----- High AUC case --------

     viewdata(dataList.High)

     fit.High <- fit_Bayesian_FROC(dataList.High,ite=111)

     draw_bi_normal(fit.High)



#            ----- Low AUC case --------

     viewdata(dataList.Low)

     fit.Low <- fit_Bayesian_FROC(dataList.Low)

     draw_bi_normal(fit.Low)

# dottest
```

---

explanation_about_package_BayesianFROC
Explanation of this package

Description

In R console, explanation are shown.

Usage

```
explanation_about_package_BayesianFROC()
```

Examples

```
explanation_about_package_BayesianFROC()
```

`explanation_for_what_curves_are_drawn` *Print out about what curves are drawn*

Description

For package developer.

Usage

```
explanation_for_what_curves_are_drawn(modalityID, readerID)
```

Arguments

`modalityID`	A vector.
`readerID`	A vector..

Value

Nothing

Examples

```
#================The first example=====================================


  modalityID <-c(1,2)
  readerID <-c(1,2,3)

  explanation_for_what_curves_are_drawn( modalityID, readerID )


#================The second example=====================================


  modalityID <- 1
  readerID <-c(1,2,3)

  explanation_for_what_curves_are_drawn( modalityID, readerID )


# dottest
```

extractAUC *Extract AUC for each modality from hierarchical Bayesian models.*

Description

EAPs

Usage

```
extractAUC(StanS4class, dig = 3, summary = TRUE)
```

Arguments

StanS4class	This is an output of rstan::stan for a single reader and a single modality. More precisely, this is an object of some inherited class from the S4 class called stanfit in the rstan package.
dig	digits of estimates.
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Value

The estimates of AUC with respect to modalities.

extract_EAP_by_array *Extract Etimates Preserving Array Format.*

Description

Extract EAP extimates by array format.

Usage

```
extract_EAP_by_array(StanS4class, name.of.parameter)
```

Arguments

StanS4class	fitted model object, from an output of rstan::sampling().
name.of.parameter	An parameter name (given as a character string, should not surround by ""). The name of parameter which user want to extract. Parameters are contained in the parameter block of each Stan file in the path: inst/extdata.

Details

If an estimate is an array, then this function extract estimated parameters preserving an array format. The rstan also has such function, i.e., rstan::get_posterior_mean(). However this function does not extract paramter as an array but coerce to the class matrix.

Value

A list of datalists from the posterior predictive distribution

Examples

```
#================================The first example: MRMC case ========================
# Make a fitted model object of class stanfitExtended
# which is inherited from the S4class stanfit.
# The following example, fitted model is the hierarchical Bayesian FROC model
# which is used to compare modality.

 fit <- fit_Bayesian_FROC( ite  = 1111 ,
                           summary = FALSE   ,
                           dataList = dataList.Chakra.Web.orderd,cha=1
                            )

#  Extract one dimensional array "z = z[]",

                  z   <- extract_EAP_by_array(
                                                fit,  # The above fitted model object
                                                z     # The parameter contained in "fit"
                                                )



#  Extract two dimensional array "AA = AA[ , ]",

                  AA  <- extract_EAP_by_array(
                                               fit,
                                               AA
                                               )


#  Extract three dimensional array "ppp = ppp[ , , ]",

                  ppp <- extract_EAP_by_array(fit,ppp)



# ####1#### ####2#### ####3#### ####4#### ####5#### ####6#### ####7#### ####8#### ####9####
#================= The second example: singler reader and single modality ==============



#   Of course, for the case of srsc, it is also available.
```

```
#   We shall show the case of srsc in which case the parameters are not array,
#   but in such a case we can extract estimates preserving its format such as vector.

 fit <- fit_Bayesian_FROC( ite  = 1111 ,
                           summary = FALSE   ,
                           dataList = dataList.Chakra.1,
                           cha=2
                            )

#  To extract the posterior mean for parameter "A" representing AUC, we run the following;


          A <- extract_EAP_by_array(
                                     fit,
                                      A
                                      )




#  To extract the posterior mean for parameter "z" indicating decision thresholds;


          z <- extract_EAP_by_array(
                                      fit,
                                      z
                                      )



# 2019.05.21 Revised.
# dottest
```

`extract_EAP_CI` *MRMC: Extract Estimates of a vector from stanfitExtended object*

Description

We extract the EAPs and CIs from the stanfitExtended S4 class which is an inherited class of the stanfit S4 class.

Usage

```
extract_EAP_CI(StanS4class, parameter.name, dimension.of.parameter,
  dig = 5, summary = TRUE)
```

Arguments

StanS4class	An S4 object of the class stanfit. No need that it is the S4 class stanfitExtended.
parameter.name	character vector. E.g., use as "aaa" . for names of parameter described in the parameter block of stan file.
dimension.of.parameter	If parameter aaa is vector, i.e.,aaa[1],aaa[2],...aaa[6] then dimension.of.parameter = 6
dig	An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Details

To validate our model has no bias, that is comparison of true parameters of distributions and EAPs, we have to extract the estimates from the stanfitExtended object. And this function do it.

Value

EAPs, CI.

See Also

extract_estimates_MRMC

Examples

```
# First we create the following fitted model object of  class stanfitExtend.


   fit <- fit_Bayesian_FROC(
                      BayesianFROC::dataList.Chakra.Web.orderd, # data
                             ite = 1111,                        # MCMC iteration
                             summary = FALSE                    # vervose
   )



# Second, to extract the EAPs of the parameter z,
# we also have to specify the dimension of vector z as follows.


       extract_EAP_CI(

                 fit,  #  The above fitted model object
```

```
                  "z",  #  The parameter name described in parameter block of stan file
                   5    #  The dimension of vector z
                      )



# One more example: to extract the EAPs of the parameter dz,
# we also have to specify its dimension of vector dz as follows.

          list.of.dz <-extract_EAP_CI(fit,"dz",4)

# One more example: to extract the EAPs of the parameter w,
#     we also have to specify its dimension of vector w as follows.

           list.w  <-extract_EAP_CI(fit,"w",1)


# Note that this function can extract only parameter of "vector" and not "array" !!
# To extract such array please use "extract_estimates_MRMC()"
# which extract all parameters from a hierarchical Bayesian model
# estimated from user data. So, this function is no longer meaningless,
# and I will delete this.


# I forgot where I use this function
# 2019.05.21 Revised.
# dottest
```

`extract_estimates_MRMC` *MRMC: Extract All Estimates from stanfitExtended object*

Description

Extract estimates, preserving its format, such as array, vector. From MRMC models, it extract the EAPs and CIs.

Usage

```
extract_estimates_MRMC(StanS4class, dig = 3)
```

Arguments

`StanS4class`	An S4 class object. The Name of S4 class is stanfitExtended.
`dig`	An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

Details

To validate our model has no bias, that is comparison of true parameters of distributions and EAPs, we have to extract the estimates from the stanfitExtended object. And this function do it.

Value

EAPs, CIs which preserving its format, such as array, vector.

See Also

extract_EAP_CI() is used in the function `extract_estimates_MRMC()`.

Examples

```
 fit <- fit_Bayesian_FROC(
             BayesianFROC::dataList.Chakra.Web.orderd,
             summary = FALSE,
             ite=111)

 EAPs <- extract_estimates_MRMC(fit)

# dottest
```

`extract_parameters_from_replicated_models`
Extract Estimates From Replicated MRMC Model

Description

Extract Estimates From Replicated MRMC Model

Usage

```
extract_parameters_from_replicated_models(initial.seed = 123,
  mu.truth = BayesianFROC::mu_truth, v.truth = BayesianFROC::v_truth,
  z.truth = BayesianFROC::z_truth, NI = 200, NL = 142,
  ModifiedPoisson = FALSE, No.of.replication = 2, summary = FALSE,
  ite = 1111)
```

Arguments

initial.seed	The variable initial.seed is used to replicate datasets. That is, if you take initial.seed = 1234, then the seed 1234, 1235, 1236, 1237, 1238, ... are for the first replication, the second replication, the third replication, ... If the n-th model does not converge for some n, then such model has no mean and thus the non-convergent models are omitted to calculate the errors.
mu.truth	array of dimension (M,Q). Mean of represents the signal distribution of bi-normal assumption.
v.truth	array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.
z.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
NI	Number of Images.
NL	Number of Lesions.
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
No.of.replication	For fixed number of lesions, images, the dataset of hits and false alarms are replicated, and the number of replicated datasets are specified by this variable.
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.
ite	An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

Value

list of estimates

Examples

```
list.of.estimates <- extract_parameters_from_replicated_models()
```

`false_and_its_rate_creator` *MRMC False Alarm Creator.*

Description

From threshold, mean and S.D., data of False Alarm are created.

Usage

```
false_and_its_rate_creator(z.truth = BayesianFROC::z_truth, NI = 333,
  NL = 111, ModifiedPoisson = FALSE, seed = 12345)
```

Arguments

`z.truth`	Vector of dimension = C represents the thresholds of bi-normal assumption.
`NI`	The number of images.
`NL`	The number of lesions.
`ModifiedPoisson`	Logical, that is `TRUE` or `FALSE`. If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm are `per lesion`, and if `ModifiedPoisson = FALSE`, then Poisson rate of false alarm are `per image`. To know detail, refer the author's paper in which I explained per image and per lesion.
`seed`	The seed for creating false alarm which are generated by the Poisson distributions using the specified seed.

Details

From threshold, mean and S.D. of the latent Gaussian noise distribution in the bi-normal assumption, data of False Alarm are created. For the process of this drawing false alarm samples, its rate are also created. So, in the return values of the function, the rates for each confidence level is also attached.

Examples

```
false.rate <- false_and_its_rate_creator()
```

`false_and_its_rate_creator_MRMC` *MRMC: False Alarm Creator For each Modality and each Reader.*

Description

From threshold, mean and S.D., data of False Alarm are created.#'@details In our hierarchical model, false alarm rate does not depend on the readers or modalities. Thus this sampling function merely draws samples from the Poisson distribution of the same false alarm rate. Of course, this same false rate of the Poisson distributions is not desired one. Since we should assume that each reader with different modality should differ. To accomplish this, we have to assume that threshold parameter of Gaussian assumption should depend on the reader and modality. But such model does not converge in the Hamiltonian Monte Carlo simulation.

Usage

```
false_and_its_rate_creator_MRMC(z.truth = BayesianFROC::z_truth,
  NI = 333, NL = 111, ModifiedPoisson = FALSE, seed = 12345,
  M = 5, Q = 4, summary = TRUE)
```

Arguments

`z.truth`	Vector of dimension = C represents the thresholds of bi-normal assumption.
`NI`	The number of images.
`NL`	The number of lesions.
`ModifiedPoisson`	Logical, that is `TRUE` or `FALSE`. If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm are `per lesion`, and if `ModifiedPoisson = FALSE`, then Poisson rate of false alarm are `per image`. To know detail, refer the author's paper in which I explained per image and per lesion.
`seed`	The seed for creating false alarm which are generated by the Poisson distributions using the specified seed.
`M`	Number of modalities
`Q`	Number of readers
`summary`	Logical: `TRUE` of `FALSE`. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Value

Vector for false alarms as an element of list of MRMC data.

Examples

```
false_and_its_rate_creator_MRMC()
```

`fffaaabbb` *Package Development tools and memo.*

Description

This is for the author of this package. project option build and reload ^ ^ ^ ^

Usage

```
fffaaabbb()
```

`fit_Bayesian_FROC` *Fit model to data*

Description

Creates a fitted model object of class `stanfitExtended`.

Usage

```
fit_Bayesian_FROC(dataList, ModifiedPoisson = FALSE,
  PreciseLogLikelihood = TRUE, DrawCurve = length(dataList$m) == 0,
  Drawcol = TRUE, summary = TRUE,
  make.csv.file.to.draw.curve = FALSE, mesh.for.drawing.curve = 10000,
  significantLevel = 0.7, new.imaging.device = TRUE, cha = 1,
  ite = 10000, DrawFROCcurve = TRUE, DrawAFROCcurve = FALSE,
  DrawCFPCTP = TRUE, dig = 5, war = floor(ite/5), see = 1234567,
  Null.Hypothesis = FALSE)
```

Arguments

`dataList` This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
            h = c(41,22,14,8,1),
            f = c(1,2,5,11,13),
            NL = 124,
```

```
NI = 63,
C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

———— Single reader and single modality (SRSC) case. ————

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22

equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

—

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

---

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So,

to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	ModalityID m	Confidence levels c	No. of false alarms f	No. of hits. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14
1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

`ModifiedPoisson`

Logical, that is `TRUE` or `FALSE`. If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm are `per lesion`, and if `ModifiedPoisson = FALSE`, then Poisson rate of false alarm are `per image`. To know detail, refer the author's paper in which I explained per image and per lesion.

`PreciseLogLikelihood`

Logical, that is `TRUE` or `FALSE`. If `PreciseLogLikelihood = TRUE`(default), then Stan calculates the precise log likelihood with target formulation.

If `PreciseLogLikelihood = FALSE`, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warnning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.

`DrawCurve`

Logical: `TRUE` of `FALSE`. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set `DrawCurve =TRUE`, if not then `DrawCurve =FALSE`. The reason why the author make this variable

DrawCurve is that it takes long time in MRMC case to draw curves, and thus default value is FALSE.

Drawcol Logical: TRUE of FALSE. Whether the (A)FROC curve is to be drawn by using color of dark theme. The default value is a TRUE.

summary Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

make.csv.file.to.draw.curve
Logical: TRUE of FALSE. Whether to create a csv file. If TRUE then csv file is created in your desktop to draw an FROC curve and cumulative hits and false alarms by scatter plot. Default is FALSE since it took times to create csv files.

mesh.for.drawing.curve
An integer indicating number of dots drawing the curves, default =10000.

significantLevel
This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

new.imaging.device
Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

cha An argument of rstan::sampling() in which it is named chains. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, default = 1.

ite An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

DrawFROCcurve Logical: TRUE of FALSE. Whether the FROC curve is to be drawn.

DrawAFROCcurve Logical: TRUE of FALSE. Whether the AFROC curve is to be drawn.

DrawCFPCTP Logical: TRUE of FALSE. Whether the CFPCTP points are to be drawn.

dig An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

war An argument of rstan::sampling() in which it is named warmup. A positive integer representing the Burn in period, which must be less than ite. Defaults to war = floor(ite/5)=10000/5=2000,

see An argument of rstan::sampling() in which it is named seed. A positive integer representing seed used in stan, default = 1234567.

Null.Hypothesis
Logical, that is TRUE or FALSE. If Null.or.Alternative.Hypothesis = FALSE(default), then fit the *alternative model* to dataList If Null.or.Alternative.Hypothesis = TRUE, then fit the *null model* to dataList. Note that the null model is constructed under the null hypothesis that all modality are same observer performance ability. The alternative model is made under the assumption that all modality are not

same. The reason why author creats this parameter is to test the null hypothesis by the Bayes factor. But the result of test is not desired one for me. Thus the test is under construction.

Details

This function can be handled in MRMC data and It automatically detects whether the data was MRMC or not and adapts itself accordingly.

It also plots FROC curves if singler reader and single modality case.

Build the S4 object by Stan to fit the author's Bayesian models introduced in the author's paper. The output of the `rstan::sampling()` is an object of the S4 class called `stanfit`. But, in this package, we extended the *stanfit* class to an S4 class named *stanfitExtended*. The new S4 class `stanfitExtended` included new slots for sequential analysis. So, the return value of the function is not the S4 class *stanfit* but the new S4 class `stanfitExtended`. Thus, to apply the functions in the **rstan** package for fitted model objects , we have to change the class of the S4 fitted model objects using the function `methods::as()` such as by the code `methods::as( object = fitted.model.object, "stanfit")`.

The following items are main substances of this function.

What we can do by this function

———————————*[FIT]*——————————— * `rstan::sampling()` runs to demonstrate the author's Bayeisan model. Return is an object of class "stanfitExtended" that represents an Bayesiain FROC model.

———————————*[PLOT CURVES]*——————————— * If dataset is a single reader and single modality, then the curves are drawn in default. But in the MRMC case,it tooks long time, so the plot is not done for this case.

———————————*[Create A fitted model object]*——————————— * The return value of this function is an S4 object whose class is inherited from the S4 class of the rstan package, which called stanfit.

———————————*[PRINT]*——————————— * Estimates are printed in the R (Studio) console.

This function `fit_Bayesian_FROC` is available both single reader and single modality case and multiple reader and multiple modality case. Confidence level vector is not required but it is implicitly refered as the decreasing oreder, For example, if C=3, then it would be a form `c=c(3,2,1,3,2,1,...)`. Even if you write your data according to the order `c=c(1,2,3,1,2,3,...)`, the program does not consider as your order, but `c=c(3,2,1,3,2,1,...)` instead.

Value

An object of class `stanfitExtended` which is an inherited S4 class from the S4 class `stanfit` By `rstan::sampling`, the function fit the author's FROC Bayesian models to user data.

Use this fitted model object for sequential analysis, such as drawing the FROC curve and alternative FROC (AFROC) curves.

———————————————————————————————————————

—- Notations and simbols for the **Outputs of single reader and single modality case** ——

———————————————————————————————————————-

In the following, the notations for estimated parameters are shown.

`w` A real number representing ***the lowest threshold*** of the Gaussian assumption (binormal assumption). so `w=z[1]`.

`dz[1]` A real number representing ***the difference of the first and second threshold*** of the Gaussian assumption: `dz[1] := z[2] - z[1]`.

`dz[2]` A real number representing the difference of the second and third threshold of the Gaussian assumption: `dz[2] := z[3] - z[2]`.

`dz[3]` A real number representing the difference of the third and fourth threshold of the Gaussian assumption: `dz[3] := z[4] - z[3]`.

...

`m` A real number representing the The ***mean*** of the Latent Gaussian distribution for diseased images. In TeX, it denoted by μ

`v` A positive real number representing the ***standard deviation*** of the Latent Gaussian distribution for diseased images.In TeX, it will be denoted by σ, not the square of σ.

`p[1]` A real number representing the Hit rate with confidence level 1.

`p[2]`A real number representing the Hit rate with confidence level 2.

`p[3]`A real number representing the Hit rate with confidence level 3.

...

`l[1]`A positive real number representing the (Cumulative) False positive rate with confidence level 1. In TeX, it will be denoted by λ_1.

`l[2]`A positive real number representing the (Cumulative) False positive rate with confidence level 2. In TeX, it will be denoted by λ_2.

`l[3]`A positive real number representing the (Cumulative) False positive rate with confidence level 3. In TeX, it will be denoted by λ_3.

`l[4]`A positive real number representing the (Cumulative) False positive rate with confidence level 4. In TeX, it will be denoted by λ_4.

...

`dl[1]`A positive real number representing the difference `l[1] - l[2]`.

`dl[2]`A positive real number representing the difference `l[2] - l[3]`.

`dl[3]`A positive real number representing the difference `l[3] - l[4]`.

...

`z[1]` A real number representing the lowest threshold of the (Gaussian) binormal assumption.

`z[2]` A real number representing the 2nd threshold of the (Gaussian) binormal assumption.

`z[3]` A real number representing the 3rd threshold of the (Gaussian) binormal assumption.

`z[4]` A real number representing the fourth threshold of the (Gaussian) binormal assumption.

`a` A real number defined by `m/v`, please contact the author's paper for detail.

`b` A real number representing defined by `1/v`, please contact the author's paper for detail.

`A` A positive real number between 0 and 1, representing AUC, i.e., the area under the alternative ROC curve.

`lp__` The logarithmic likelihood of our model for your data.

—- **Notations and simbols:** Outputs of Multiple Reader and Multiple Modality case ——

`w` The lowest threshold of the Gaussian assumption (binormal assumption). so `w=z[1]`.

`dz[1]` The difference of the first and second threshold of the Gaussian assumption.

`dz[2]` The difference of the second and third threshold of the Gaussian assumption.

`dz[3]` The difference of the third and fourth threshold of the Gaussian assumption.

...

`mu` The mean of the Latent Gaussian distribution for diseased images.

`v` The variance of the Latent Gaussian distribution for diseased images.

`ppp[1,1,1]` Hit rate with confidence level 1, modaity 1, reader 1.

`ppp[2,1,1]` Hit rate with confidence level 2, modality 1, reader 1.

`ppp[3,1,1]` Hit rate with confidence level 3, modality 1, reader 1.

...

`l[1]` (Cumulative) False positive rate with confidence level 1.

`l[2]` (Cumulative) False positive rate with confidence level 2.

`l[3]` (Cumulative) False positive rate with confidence level 3.

`l[4]` (Cumulative) False positive rate with confidence level 4.

...

`dl[1]` This is defined by the difference `l[1] - l[2]`.

`dl[2]` This is defined by the difference `l[2] - l[3]`.

`dl[3]` This is defined by the difference `l[3] - l[4]`.

...

`z[1]` The lowest threshold of the (Gaussian) binormal assumption.

`z[2]` The 2nd threshold of the (Gaussian) binormal assumption.

`z[3]` The 3rd threshold of the (Gaussian) binormal assumption.

`z[4]` The fourth threshold of the (Gaussian) binormal assumption.

`aa` This is defined by `m/v`, please see the author's paper for more detail.

`bb` This is defined by `1/v`, please see the author's paper for more detail.

`AA` The area under alternative FROC curve associated to reader and modality.

`A` The area under alternative FROC curve associated to modality.

`hyper_v` Standard deviation of `AA` around `A`.

`lp__` The logarithmic likelihood of our model for your data.

Author(s)

Issei Tsunoda

References

Bayesian Models for Free-response Receiver Operating Characteristic Analysis; Pre-print

See Also

——— **Before fitting:** *create a dataset*

Convert from JAFROC format xlsx file to the author's format

`convertFromJafrodataset_creator_new_version` Create an R object which represent user data.

`create_dataset` Create an R object which represent user data.

——— **Further subsequential analysis: Plot curves** Using the result of fitting a Bayesian FROC model, we can go sequential analysis.

`DrawCurves` for drawing free response ROC curves.

——— **Further subsequential analysis: Validation of the Model**

`p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit` Calculation of a p-value in the Bayesian paradigm.

——— R **objects of example datasets from real world or fictious:**

`dataList.Chakra.1` A `list` for an example dataset of single reader and single modality data. The word Chakra in the dataset name means that it appears in the paper of Chakraborty.

`dataList.Chakra.2` A `list` for an example dataset of single reader and single modality data. The word Chakra in the dataset name means that it appears in the paper of Chakraborty.

`dataList.Chakra.3` A `list` for an example dataset of single reader and single modality data. The word Chakra in the dataset name means that it appears in the paper of Chakraborty.

`dataList.Chakra.4` A `list` for an example dataset of single reader and single modality data. The word Chakra in the dataset name means that it appears in the paper of Chakraborty.

`dataList.high.ability` A `list` for an example dataset of single reader and single modality data

`dataList.low.ability` A `list` for an example dataset of single reader and single modality data

`dataList.Chakra.Web` A `list` for an example dataset of multiple readers and multiple modalities data. The word Chakra in the dataset name means that it appears in the paper of Chakraborty.

`data.hier.ficitious` A `list` for an example dataset of multiple readers and multiple modalities data

`dataList.High` A `list` for an example dataset of single reader and single modality data whose AUC is high.

`dataList.Low` A `list` for an example dataset of single reader and single modality data whose AUC is low.

* **

See Vignettes for more details.

****

Examples

```
#============================The first example=====================================

            # Making FROC Data and Fitting a Model to the data

  #                         Notations
  #            h = hits = TP = True Positives
  #            f = False alarms = FP = False Positives

#1) Build the data for singler reader and single modality  case.

dat <- list(c=c(3,2,1),    #Confidence level. Note that c is ignored.
            h=c(97,32,31), #Number of hits for each confidence level
            f=c(1,14,74),  #Number of false alarms for each confidence level

            NL=259,        #Number of lesions
            NI=57,         #Number of images
            C=3)           #Number of confidence level



#  where,
#        c denotes confidence level, i.e., rating of reader.
#                3 = Definitely deseased,
#                2 = subtle,.. deseased
#                1 = very subtle
#        h denotes number of hits (True Positives: TP) for each confidence level,
#       f denotes number of false alarms (False Positives: FP) for each confidence level,
#        NL denotes number of lesions,
#        NI denotes number of images,


# For example, in the above example data,
#  the number of hits with confidence level 3 is 97,
#  the number of hits with confidence level 2 is 32,
#  the number of hits with confidence level 1 is 31,

#  the number of false alarms with confidence level 3 is 1,
#  the number of false alarms with confidence level 2 is 14,
#  the number of false alarms with confidence level 1 is 74,

#2) Fit the FROC model.
  #Since dataset named dat are single reader and single modality,
  #the function build the such model by running the following code.
```

```
          fit <-   BayesianFROC::fit_Bayesian_FROC(
                               dat,       # dataset
                               ite=1111,  #To run in time <5s.
                               cha=1      # number of chains, it is better more large.
                               )
```

```
#3) Using the S4 object fit, we can go further step, such as calculation of the
# Chisquare and the p value of the Bayesian version for testing the goodness of fit.
# I think p value has problems that it relies on the sample size with monotonicity.
# But it is well used, thus I hate but I implement the p value.
```

```
# (( REMARK ))
# Should not write the above data as follows:

# MANNER (A)   dat <- list(c=c(1,2,3),h=c(31,32,97),f=c(74,14,1),NL=259,NI=57,C=3)


# Even if user write data in the above MANNER (A),
# the program interpret it as the following MANNER (B);

# MANNER (B)   dat <- list(c=c(3,2,1),h=c(31,32,97),f=c(74,14,1),NL=259,NI=57,C=3)

# Because the vecetor c is ingored in the program,
# and it is generated by  rep(C:1) automatically  in the internal of the function.
# So, we can omit c from the list.
```

```
#This package is very rigid format, so please be sure that your format is
#exactly same to the data in this package.
#More precisely, the confidence level vector should be denoted rep(C:1) (Not rep(1:C)).
#  Note that confidence level vector c  should not be specified.
#   If specified, will be ignored ,
#  since it is created by   c <-c(rep(C:1)) in the program and
#  do not refer from user input confidecnce level vector,
#  where C is the highest number of confidence levels.
```

```
#=======  The Second Example:  From data endowed in this package to fitting ============


#    (1)First, we prepare the data from this package.


                dat  <- BayesianFROC::dataList.Chakra.1


#    (2)Second, we run fit_Bayesian_FROC() in which the rstan::stan() is implemented.
#    with data named "dat"  and the author's Bayesian model.


                fit <-  fit_Bayesian_FROC(dat)




#   Now, we get the stan's out put, i.e.,  an S4 class object named "fit".

# << Minor Comments>>
#  More precisely, this is an S4 object of some inherited class (named stanfitExtended)
#  which is extended using stan's S4 class named "stanfit". This new S4 class
#  has new slots for the informations such as user data, plotting data for FROC curves,
#  input data to run this function, etc.

#  Using the output "fit",

#  we can use the functions in the "rstan" package, for example, as follows;


         rstan::stan_trace(fit)# stochastic process of a posterior estimate
         rstan::stan_hist(fit) # Histogram of a posterior estimate
         rstan::stan_rhat(fit) # Histogram of rhat for all parameters
         rstan::summary(fit)   # summary of fit by rstan




#================The third example:   Hand made data and fitting  ==========

#    Fit a model to a hand made data
```

```
#     1) Build the data for singler reader and single modality  case.

   dat <- list(
            c=c(3,2,1),    #  Confidence level, which is ignored.
            h=c(97,32,31), #  Number of hits for each confidence level
            f=c(1,14,74),  #  Number of false alarms for each confidence level

            NL=259,       #   Number of lesions
            NI=57,        #   Number of images
            C=3)          #   Number of confidence level




#  where,
#        c denotes confidence level, , each components indicates that
#                3 = Definitely lesion,
#                2 = subtle,
#                1 = very subtle
#          That is the high number indicates the high confidence level.
#        h denotes number of hits
#          (True Positives: TP) for each confidence level,
#        f denotes number of false alarms
#          (False Positives: FP) for each confidence level,
#        NL denotes number of lesions,
#        NI denotes number of images,


#     2) Fit  and draw FROC and AFROC curves.



           fit <-   fit_Bayesian_FROC(dat, DrawCurve = TRUE)



# (( REMARK ))
#           Changing the hits and false alarms denoted by h and  f
#           in the dataset denoted by dat, respectively,
#           user can draw the verious curves.
#           Enjoy drawing the curves for various single reader and single modality data.


#==========           The 4th example:  the data created by a function       ==========


#         1) Build the data by create_dataset() which endowed in this package.

                     dataList <-  create_dataset()
```

```
#Now, as as a return value of create_dataset(), we get the FROC data (list) named dataList.

#        2) Fit an MRMC or srsc FROC model.

                      fit <-  fit_Bayesian_FROC(dataList)




#========     The 5-th example:Comparison of the posterior probability for AUC ==========


# This example shows how to use the stanfit (stanfit.Extended) object.
# Using stanfit object, we can extract posterior samples and using these samples,
# we can calculate the posterior probability of research questions.


    fit <- fit_Bayesian_FROC(dataList.Chakra.Web.orderd,ite = 1111,summary =FALSE)


#    For example, we shall show the code to compute the posterior probability of the evet
#    that  the AUC of modality 1 is larger than that of modality 2:


                            e <- extract(fit)


# This code means that the MCMC samples are retained in the object e for all parameters.
# For example, the AUC is extracted by the code e$A and it is a two dimensional array.
# The first component indicates the MCMC samples and
# the second component indicate the modality ID.

# For example, the code e$A[,1] means the vector of MCMC samples of the 1 st modality.
# For example, the code e$A[,2] means the vector of MCMC samples of the 2 nd modality.
# For example, the code e$A[,3] means the vector of MCMC samples of the 3 rd modality.
#    To calculate the posterior probability of the evet
#    that the AUC of modality 1 is larger than that of modality 2,
#    we excute the following R script:

                        mean(e$A[,1] > e$A[,2])


#    Similarly, to compute the posterior probability that
#     the AUC of modality 1 is larger  than  that of modality 3:

                        mean(e$A[,1] > e$A[,3])
```

```
#    Similarly, to compute the posterior probability that
#     the AUC of modality 1 is larger  than  that of modality 4:

                      mean(e$A[,1] > e$A[,4])


#    Similarly, to compute the posterior probability that
#     the AUC of modality 1 is larger  than  that of modality 5:

                      mean(e$A[,1] > e$A[,5])


#    Similarly, to compute the posterior probability that
#     the AUC of modality 1 is larger  than  that of modality 5 at least 0.01


                      mean(e$A[,1] > e$A[,5]+0.01)


#      Similarly,

                mean( e$A[,1] > e$A[,5] + 0.01 )
                mean( e$A[,1] > e$A[,5] + 0.02 )
                mean( e$A[,1] > e$A[,5] + 0.03 )
                mean( e$A[,1] > e$A[,5] + 0.04 )
                mean( e$A[,1] > e$A[,5] + 0.05 )
                mean( e$A[,1] > e$A[,5] + 0.06 )
                mean( e$A[,1] > e$A[,5] + 0.07 )
                mean( e$A[,1] > e$A[,5] + 0.08 )



# Since any posterior distribution tends to the Dirac measure whose center is
# true parameter under the assumption that the model is correct in the sense that the
# true distribution is belongs to a family of models.
# Thus using this procedure, we will get
# the true parameter if any more large sample size we can take.


#      Close the graphic device to avoid errors in R CMD check.

                    Close_all_graphic_devices()


# dottest
```

`fit_MRMC_test` *Fit and Draw the FROC models (curves)*

Description

Fit and Draw the FROC models (curves).

Usage

```
fit_MRMC_test(dataList, DrawCurve = FALSE,
  PreciseLogLikelihood = FALSE, summary = TRUE,
  mesh.for.drawing.curve = 10000, significantLevel = 0.7, cha = 1,
  war = floor(ite/5), ite = 10000, dig = 3, see = 1234569,
  Null.Hypothesis = FALSE)
```

Arguments

`dataList`

This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
           h = c(41,22,14,8,1),
           f = c(1,2,5,11,13),
           NL = 124,
           NI = 63,
           C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

——— **Single reader and single modality (SRSC) case.** ———

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	**ModalityID** m	**Confidence levels** c	**No. of false alarms** f	**No. of hits**. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14
1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

—

DrawCurve Logical: TRUE of FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve =TRUE, if not then DrawCurve =FALSE. The reason why the author make this variable DrawCurve is that it takes long time in MRMC case to draw curves, and thus default value is FALSE.

PreciseLogLikelihood
Logical, that is TRUE or FALSE. If PreciseLogLikelihood = TRUE(default), then Stan calculates the precise log likelihood with target formulation.
If PreciseLogLikelihood = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warnning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.

summary Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

mesh.for.drawing.curve
An integer indicating number of dots drawing the curves, default =10000.

significantLevel
This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

cha An argument of rstan::sampling() in which it is named chains. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, default = 1.

war An argument of rstan::sampling() in which it is named warmup. A positive integer representing the Burn in period, which must be less than ite. Defaults to war = floor(ite/5)=10000/5=2000,

ite An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

dig An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

see An argument of rstan::sampling() in which it is named seed. A positive integer representing seed used in stan, default = 1234567.

Null.Hypothesis
Logical, that is TRUE or FALSE. If Null.or.Alternative.Hypothesis = FALSE(default), then fit the *alternative model* to dataList If Null.or.Alternative.Hypothesis = TRUE, then fit the *null model* to dataList. Note that the null model is constructed under the null hypothesis that all modality are same observer performance ability. The alternative model is made under the assumption that all modality are not same. The reason why author creats this parameter is to test the null hypothesis

by the Bayes factor. But the result of test is not desired one for me. Thus the test is under construction.

`fit_MRMC_versionTWO` *Fit and Draw the FROC models (curves) version2.*

Description

Fit and Draw the FROC models (curves). This model is aimed to draw a free-response ROC curves for multiple readers, that is, resulting FROC curve is one for multiple readers and reflects their hits and false alarms.

Usage

```
fit_MRMC_versionTWO(dataList, DrawFROCcurve = TRUE, DrawCFPCTP = TRUE,
  version = 2, mesh.for.drawing.curve = 10000,
  significantLevel = 0.7, cha = 1, war = floor(ite/5), ite = 10000,
  dig = 5, see = 1234569)
```

Arguments

`dataList`

This is a variable in the function `rstan::sampling()` in which it is named `data`.

For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
            h = c(41,22,14,8,1),
            f = c(1,2,5,11,13),
            NL = 124,
            NI = 63,
            C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

———— **Single reader and single modality (SRSC) case.** ————

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be speci-

fied. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

---

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	**ModalityID** m	**Confidence levels** c	**No. of false alarms** f	**No. of hits**. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14
1	1	4	2	13
1	2	3	2	5
1	3	2	5	3

2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

`DrawFROCcurve` Logical: `TRUE` of `FALSE`. Whether the FROC curve is to be drawn.

`DrawCFPCTP` Logical: `TRUE` of `FALSE`. Whether the CFPCTP points are to be drawn.

`version` 2 or 3

`mesh.for.drawing.curve`
An integer indicating number of dots drawing the curves, default =10000.

`significantLevel`
This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

`cha` An argument of `rstan::sampling()` in which it is named `chains`. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, default = 1.

`war` An argument of `rstan::sampling()` in which it is named `warmup`. A positive integer representing the Burn in period, which must be less than `ite`. Defaults to war = floor(ite/5)=10000/5=2000,

`ite` An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

`dig` An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

`see` An argument of `rstan::sampling()` in which it is named `seed`. A positive integer representing seed used in stan, default = 1234567.

See Also

Example data:

BayesianFROC::dataList.one.modality

This dataset is a single modality dataset with multiple readers.

Examples

```
## Not run:

#================The first example=====================================
#(1)First, we prepare the data from this package.

    dat  <- BayesianFROC::dataList.one.modality



#(2)Second, we run fit_Bayesian_FROC() in which the rstan::stan() is implemented.
#with data named "dat"  and the author's Bayesian model.


         fit <-  fit_MRMC_versionTWO(dat,see = 12,ite=2222)

 #      It needs a lot of memory and so, in this example we take the small iteration,
 #      i.e., ite =2222. However if user execute this, then the ite =30000 is recommended
 #      for getting reliable estimates.







#Note that we change the seed from default to 12 to get a convergence model.
#If users enconter the convergence issues,
#then please consider changing the seed like this example.

#The resulting FROC curve means the summarizing curve over all readers

#================The Second example=====================================
#(1)First, we prepare the data from this package.

        dat  <- BayesianFROC::dataList.Chakra.Web



#(2)Second, we run fit_Bayesian_FROC() in which the rstan::stan() is implemented.
#with data named "dat"  and the author's Bayesian model.


         fit <-  fit_MRMC_versionTWO(dataList.Chakra.Web ,ite=2222)

#The resulting FROC curve means the summarizing curve over all readers

 #      It needs a lot of memory and so, in this example we take the small iteration,
 #      i.e., ite =2222. However if user execute this, then the ite =30000 is recommended
 #      for getting reliable estimates.
```

```
#'#      Close the graphic device to avoid errors in R CMD check.

if (!grDevices::dev.cur()>=2) {

   for (i in 1:grDevices::dev.cur()-1) {message("The",i,"-th graphic device is omitted.")
    grDevices::dev.off()
     }
     }


## End(Not run)#\dontrun
```

`fit_Null_hypothesis_model_to_` *Fit the null model*

Description

Fit the null model, representing the null hypothesis that all modalities are same.

Usage

```
fit_Null_hypothesis_model_to_(dataList, DrawCurve = FALSE,
  PreciseLogLikelihood = FALSE, summary = TRUE,
  mesh.for.drawing.curve = 10000, significantLevel = 0.7, cha = 1,
  war = floor(ite/5), ite = 10000, dig = 3, see = 1234569)
```

Arguments

`dataList` This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
            h = c(41,22,14,8,1),
            f = c(1,2,5,11,13),
            NL = 124,
            NI = 63,
            C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

——— **Single reader and single modality (SRSC) case.** ———

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

———

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

—

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

---

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case ———————

ReaderID	ModalityID	Confidence levels	No. of false alarms	No. of hits.

q	m	c	f	h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14
1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

`DrawCurve` Logical: `TRUE` of `FALSE`. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set `DrawCurve =TRUE`, if not then `DrawCurve =FALSE`. The reason why the author make this variable `DrawCurve` is that it takes long time in MRMC case to draw curves, and thus default value is `FALSE`.

`PreciseLogLikelihood`

Logical, that is `TRUE` or `FALSE`. If `PreciseLogLikelihood = TRUE`(default), then Stan calculates the precise log likelihood with target formulation.

If `PreciseLogLikelihood = FALSE`, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warnning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.

`summary` Logical: `TRUE` of `FALSE`. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

`mesh.for.drawing.curve`

An integer indicating number of dots drawing the curves, default =10000.

`significantLevel`

This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.

cha — An argument of rstan::sampling() in which it is named chains. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, default = 1.

war — An argument of rstan::sampling() in which it is named warmup. A positive integer representing the Burn in period, which must be less than ite. Defaults to war = floor(ite/5)=10000/5=2000,

ite — An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

dig — An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

see — An argument of rstan::sampling() in which it is named seed. A positive integer representing seed used in stan, default = 1234567.

---

fit_srsc — *Draw an FROC curve and an AFROC curve using Bayesian approach. Build S4 classes by stan in the case of Single reader and Single modality (srsc)*

Description

Build the S4 class by stan with your single reader and single modality data dataList.

The model is the author's Bayesian model introduced the author's paper.

Before running the function stan_srsc, you should confirm that your dataset is correctly formulated by the function viewdataSRSC().

Usage

```
fit_srsc(dataList, new.imaging.device = TRUE, DrawCurve = T,
  ModifiedPoisson = FALSE, PreciseLogLikelihood = FALSE,
  Drawcol = TRUE, summary = TRUE, DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE, DrawCFPCTP = TRUE,
  mesh.for.drawing.curve = 10000, make.csv.file.to.draw.curve = FALSE,
  cha = 4, ite = 3000, dig = 5, war = floor(ite/5), see = 1234)
```

Arguments

dataList — it should include f, h, NL, NI, C. The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by C, are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, you should confirm the orders by the function

new.imaging.device
: Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

DrawCurve
: This is a dichotomous, i.e., TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve =TRUE, if not then DrawCurve =FALSE. The reason why the author make this variable DrawCurve is that it take long time to draw curves, and thus default value is FALSE.

ModifiedPoisson
: This is dichotomous, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, user read the author's paper in which I explained per image and per lesion.

PreciseLogLikelihood
: If PreciseLogLikelihood = TRUE, then Stan calculates the precise log likelihood.

  If PreciseLogLikelihood = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function.

Drawcol
: Logical: TRUE of FALSE. Whether the (A)FROC curve is to be drawn by using color of dark theme. The default value is a TRUE.

summary
: Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

DrawFROCcurve
: Logical: TRUE of FALSE. Whether the FROC curve is to be drawn.

DrawAFROCcurve
: Logical: TRUE of FALSE. Whether the AFROC curve is to be drawn.

DrawCFPCTP
: Logical: TRUE of FALSE. Whether the CFPCTP points are to be drawn.

mesh.for.drawing.curve
: An integer indicating number of dots drawing the curves, default =10000.

make.csv.file.to.draw.curve
: Logical: TRUE of FALSE. Whether to create a csv file. If TRUE then csv file is created in your desktop to draw an FROC curve and cumulative hits and false alarms by scatter plot. Default is FALSE since it took times to create csv files.

cha
: An argument of rstan::stan, which means the number of chains generated by Hamiltonian Monte Carlo method, and, default = 4.

ite
: An argument of rstan::stan, which means the number of samples generated by Hamiltonian Monte Carlo method, and, default = 3000. If your model could not converge, then raise this number.

dig
: An argument of rstan::stan, which means the Significant digits, used in stan Cancellation. default = 3,

war
: An argument of rstan::stan, which means the Burn in period, default = 1000,

see
: An argument of rstan::stan, which means a seed used in stan,

  default = 1234. If your model could not converge, then change this number.

Value

An S4 class, created by `rstan::stan`. Using this S4 class , you can go ahead to the next step, that is, drawing the FROC curve and alternative FROC (AFROC) curves.

Examples

```
#First, we prepare the data endowed with this package.

        dat  <- get(data("dataList.Chakra.1"))



#Second, we run the stan funtion
#with data named "dat"  and the author's Bayesian model.


           fit <-  fit_srsc(dat)




#Now, we get the stan's out put S4 class named "fit".

#Using the return value "fit",

#we can use the functions in the "rstan" package,
# To do so, we have to change the class from the inherited class to
# the class of the stanfit, as follows;


        fit <- as(fit, "stanfit")

#   e.g.,

          rstan::stan_trace(fit)
          rstan::stan_hist(fit)
          rstan::stan_rhat(fit)

#donttest
```

`fit_srsc_per_image_test` *Build S4 classes with Drawing by "per image" model*

Description

Build the S4 class by stan with your single reader and single modality data dataList, with Drawing.

The model is the author's Bayesian model introduced the author's paper.

Before running the function curve_srsc, you should confirm that your dataset is correctly formatted by the function viewdataSRSC().

Usage

```
fit_srsc_per_image_test(dataList, new.imaging.device = TRUE,
  DrawCurve = TRUE, PreciseLogLikelihood = TRUE, Drawcol = TRUE,
  make.csv.file.to.draw.curve = FALSE, mesh.for.drawing.curve = 10000,
  summary = TRUE, DrawFROCcurve = TRUE, DrawAFROCcurve = FALSE,
  DrawCFPCTP = TRUE, cha = 4, ite = 3000, dig = 5,
  war = floor(ite/5), see = 1234)
```

Arguments

dataList	it should include f, h, NL, NI, C. The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by C, are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, you should confirm the orders by the function
new.imaging.device	Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.
DrawCurve	This is a dichotomous, i.e., TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve =TRUE, if not then DrawCurve =FALSE. The reason why the author make this variable DrawCurve is that it take long time to draw curves, and thus default value is FALSE.
PreciseLogLikelihood	Logical, that is TRUE or FALSE. If PreciseLogLikelihood = TRUE(default), then Stan calculates the precise log likelihood with target formulation. If PreciseLogLikelihood = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function. In past, I distinct the stan file, one is target formulation and the another is not. But non-target formulation cause some Jacobian warnning, thus I made all stanfile with target formulation when I uploaded to CRAN. Thus this variable is now meaningless.
Drawcol	Logical: TRUE of FALSE. Whether the (A)FROC curve is to be drawn by using color of dark theme. The default value is a TRUE.
make.csv.file.to.draw.curve	Logical: TRUE of FALSE. Whether to create a csv file. If TRUE then csv file is created in your desktop to draw an FROC curve and cumulative hits and false alarms by scatter plot. Default is FALSE since it took times to create csv files.
mesh.for.drawing.curve	An integer indicating number of dots drawing the curves, default =10000.

summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.
DrawFROCcurve	Logical: TRUE of FALSE. Whether the FROC curve is to be drawn.
DrawAFROCcurve	Logical: TRUE of FALSE. Whether the AFROC curve is to be drawn.
DrawCFPCTP	Logical: TRUE of FALSE. Whether the CFPCTP points are to be drawn.
cha	An argument of rstan::stan, which means the number of chains generated by Hamiltonian Monte Carlo method, and, default = 4.
ite	An argument of rstan::stan, which means the number of samples generated by Hamiltonian Monte Carlo method, and, default = 3000. If your model could not converge, then raise this number.
dig	An argument of rstan::stan, which means the Significant digits, used in stan Cancellation. default = 3,
war	An argument of rstan::stan, which means the Burn in period, default = 1000,
see	An argument of rstan::stan, which means a seed used in stan, default = 1234. If your model could not converge, then change this number.

Value

stan.srsc This is S4 class!! More precisely this is a S4 class, created by rstan::stan.

The stan S4 class object are build, named by stan.srsc.

Using S4 class stan.srsc, you can go ahead to the next step, that is, drawing the FROC curve and alternative FROC (AFROC) curves.

Examples

```
#First, we prepare the data endowed with this package.

        dat  <- get(data("dataList.Chakra.1"))



#Second, we run the stan funtion
#with data named "dat"  and the author's Bayesian model.


           fit <-    fit_srsc_per_image_test(dat)




#      Close the graphic device to avoid errors in R CMD check.
```

```
Close_all_graphic_devices()

# dottest
```

`fit_srsc_per_lesion` *Build S4 classes with Drawing by "per lesion" model*

Description

Build the S4 class by stan with your single reader and single modality data `dataList`, with Drawing.

The model is the author's Bayesian model introduced the author's paper.

Before running the function `curve_srsc`, you should confirm that your dataset is correctly formatted by the function `viewdataSRSC()`.

Usage

```
fit_srsc_per_lesion(dataList, DrawCurve = TRUE,
  PreciseLogLikelihood = TRUE, Drawcol = TRUE, summary = TRUE,
  make.csv.file.to.draw.curve = FALSE, new.imaging.device = TRUE,
  mesh.for.drawing.curve = 10000, DrawFROCcurve = TRUE,
  DrawAFROCcurve = FALSE, DrawCFPCTP = TRUE, cha = 4, ite = 3000,
  dig = 5, war = floor(ite/5), see = 1234)
```

Arguments

`dataList`	it should include `f, h, NL, C`, note that the number of images `NI` is not necessary. The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by `C`, are included, however, `c` means the confidence level is not required, it is created by `c <-c(rep(C:1))`, where `C` is the number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector. So, to confirm your false positives and hits are correctly correspondence to confidence levels, you should confirm the orders by the function
`DrawCurve`	This is a dichotomous, i.e., TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set `DrawCurve =TRUE`, if not then `DrawCurve =FALSE`. The reason why the author make this variable `DrawCurve` is that it take long time to draw curves, and thus default value is `FALSE`.
`PreciseLogLikelihood`	If `PreciseLogLikelihood` = TRUE, then Stan calculates the precise log likelihood. If `PreciseLogLikelihood` = FALSE, then Stan calculates the log likelihood by dropping the constant terms in the likelihood function.
`Drawcol`	Logical: `TRUE` of `FALSE`. Whether the (A)FROC curve is to be drawn by using color of dark theme. The default value is a `TRUE`.

summary — Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

make.csv.file.to.draw.curve — Logical: TRUE of FALSE. Whether to create a csv file. If TRUE then csv file is created in your desktop to draw an FROC curve and cumulative hits and false alarms by scatter plot. Default is FALSE since it took times to create csv files.

new.imaging.device — Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.

mesh.for.drawing.curve — An integer indicating number of dots drawing the curves, default =10000.

DrawFROCcurve — Logical: TRUE of FALSE. Whether the FROC curve is to be drawn.

DrawAFROCcurve — Logical: TRUE of FALSE. Whether the AFROC curve is to be drawn.

DrawCFPCTP — Logical: TRUE of FALSE. Whether the CFPCTP points are to be drawn.

cha — An argument of rstan::stan, which means the number of chains generated by Hamiltonian Monte Carlo method, and, default = 4.

ite — An argument of rstan::stan, which means the number of samples generated by Hamiltonian Monte Carlo method. If your model could not converge, then raise this number. The default is 3000.

dig — An argument of rstan::stan, which means the Significant digits, used in stan Cancellation. The default is 3.

war — An argument of rstan::stan, which means the Burn in period, The default is 1000.

see — An argument of rstan::stan, which means a seed used in stan,
The default is 1234. If your model could not converge, then change this number.

Value

stan.srsc.per.lesion This is S4 class!! More precisely this is a S4 class, created by rstan::stan.

The stan S4 class object are build, named by stan.srsc.per.lesion.

Using S4 class stan.srsc.per.lesion, you can go ahead to the next step, that is, drawing the FROC curve and alternative FROC (AFROC) curves.

Examples

```
#First, we prepare the data endowed with this package.

        dat  <- get(data("dataList.Chakra.1"))



#Second, we run the stan funtion
```

```
#with data named "dat"  and the author's Bayesian model.

         fit <-  fit_srsc_per_lesion(dat)



# dottest
```

`foo_of_a_List_of_Arrays` *Variance of a List of Arrays*

Description

Then the function calculates the variance over all list for each array component.

Usage

```
foo_of_a_List_of_Arrays(x, name.of.function)
```

Arguments

`x`	A List of Arrays. The dimension of array is fixed for all list component.
`name.of.function`	This is an operator, such as `mean`, `var`, `sum`,... Note that user no need to surround the input by "". For example, mean instead of "mean".

Details

Of course variance can change to `sum` or `mean` or any other functions whose entry is a vector.One can find this function in the Stack over flow, since I ask there, and thus the example given in here can also find also there. In my hierarchical Bayesian Model, the estimates has the format arrays. For example the hit rate are array whose subscript is confidence level, modality, and reader. So, when one desire to validate the estimates, it needs to calculate such variance of arrays. When I validate the estimates, I used the function.

Value

An array being reduced form use input list of array via user input operator such as `mean`, `var`, `sum`,...

Examples

```
#Suppose that x is the following list of arrays:

  a <- array(1,c(2,3,4));
  b <- array(2,c(2,3,4));
  c <- array(3,c(2,3,4));
  d <- array(4,c(2,3,4));
  x <- list(a=a,b=b,c=c,d=d)

foo_of_a_List_of_Arrays(x,sum)
foo_of_a_List_of_Arrays(x,mean)
foo_of_a_List_of_Arrays(x,stats::var)



#Note that the component of list can be vectors with fixed same length.

  y <- list(c(1,2,3),
            c(11,22,33),
            c(1111,2222,3333))


  a <- foo_of_a_List_of_Arrays(y,sum)
```

---

from_array_to_vector *Transform from an array to a vector*

Description

In the stan files, the number of hits, false alarms and hit rates in binomial assumption for MRMC case are written in the three indexed array format. Three index are constituted from confidence levels, modality ID, reader ID. However the data substituting the function BayesianFROC::fit_Bayesian_FROC() are written in the vector. So, to connect these different format, the author make this function.

Usage

```
from_array_to_vector(Three.dim.array)
```

Arguments

`Three.dim.array`
: Three dimensional array, such as the number of hits for each confidence level, modality and reader. Or false alarms. Since the author construct the substituting data list as one dimensional (one index) array, it needs to reconstruct to the three indexed array from one dimensional array whose subscript is [confidence level, modality, reader] or vice versa.

Value

One dimensional array transformed from user input three dimensional array.

Examples

```
h.array.etc <- hits_from_thresholds()

h.vector     <- from_array_to_vector(h.array.etc$h)
```

---

get_samples_from_Posterior_Predictive_distribution

Get Samples from the Predictive Posterior Distribution (PPD).

Description

Get samples from the posterior predictive distribution.

Usage

```
get_samples_from_Posterior_Predictive_distribution(StanS4class,
  new.imaging.device = TRUE, upper_x, upper_y, Colour = TRUE,
  plot.replicated.points = TRUE)
```

Arguments

StanS4class	fitted model object, from an output of `rstan::sampling()`.
new.imaging.device	Logical: TRUE of FALSE. If TRUE (default), then open a new device to draw curve. Using this we can draw curves in same plain by new.imaging.device=FALSE.
upper_x	This is a upper bound for the axis of the horisontal coordinate of FROC curve.
upper_y	This is a upper bound for the axis of the vertical coordinate of FROC curve.
Colour	logical: TRUE or FALSE. whether Colour of curves is dark theme or not.
plot.replicated.points	TRUE or FALSE. If true, then plot replicated points (hits, false alarms) by the scatter plot. This process will takes a long times. So if user has no time, then FALSE will help you.

Details

This methods to draw from the PPD is described in Gelman book, Bayesian Data Analysis. The aim of this function is to evaluate the chi square test statistics as a Bayesian sense. According to Gelman book, the chi square test need the samples from the PPD. So, we use this function to accomplish this task.

Value

A list of datalists from the posterior predictive distribution

Examples

```
fit <- fit_Bayesian_FROC(
 ite  = 1111,
  summary = FALSE ,
 dataList = BayesianFROC::dataList.Chakra.1 )




#=======  The first example ========================================================
 TPs.FPs <- get_samples_from_Posterior_Predictive_distribution(fit)


#=======  The Second Example: Short cut    ========================================
# If user has no time, then  plot.replicated.points=FALSE will help you.
# By setting FALSE, the replicated data from the posterior predictive
# distribution does not draw, and hence the running time of function become shorter.

 TPs.FPs <- get_samples_from_Posterior_Predictive_distribution(fit,
                                   plot.replicated.points =  FALSE)




#      Close the graphic device to avoid errors in R CMD check.

     grDevices::dev.new();plot(stats::runif(100),stats::runif(100))



#================The third example:  From Hand made data to fitting  ==========
#  If user want to use the scatter plots of hits and false alarms from the posterior
#  predictive distribution for the submission, then the color plot is not appropriate.
#  So, by setting the argument Colour = FALSE, the scatter plot become black and white.
#  So, user can use this scatter plot for submission.


 get_samples_from_Posterior_Predictive_distribution(fit,Colour = FALSE)

  g <-get_samples_from_Posterior_Predictive_distribution(fit)

  x <- g$CFP
```

```
  y <- g$CTP


 plot(   hexbin::hexbin(unlist(x),unlist(y))   )


#      Close the graphic device to avoid errors in R CMD check.

         Close_all_graphic_devices()
# dottest
```

---

ggplotFROC *Draw FROC curves by two parameters a and b*

Description

Plot FROC curves based on two parameters a and b.

Usage

```
ggplotFROC(a, b, mesh.for.drawing.curve = 10000, upper_x = 1,
  upper_y = 1, lower_y = 0, dataList, StanS4class)
```

Arguments

a	See paper for definition.
b	See paper for definition.
mesh.for.drawing.curve	An integer indicating number of dots drawing the curves, default =10000.
upper_x	The frame size of drawing picture.
upper_y	The frame size of drawing picture.
lower_y	The frame size of drawing picture.
dataList	This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and single modality data, the `dataList` is the following forms:

```
 dataList.Example <- list(
           h = c(41,22,14,8,1),
           f = c(1,2,5,11,13),
           NL = 124,
           NI = 63,
           C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

———— Single reader and single modality (SRSC) case. ————

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

———

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

—

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	ModalityID m	Confidence levels c	No. of false alarms f	No. of hits. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14
1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

StanS4class — This is an R object of class `stanfitExtended` inherited from the S4 class `stanfit`.

`ggplotFROC.EAP` *Draw FROC curves by two parameters a and b*

Description

Plot FROC curves based on two parameters a and b.

Usage

```
ggplotFROC.EAP(a, b, mesh.for.drawing.curve = 10000, upper_x = 1,
  upper_y = 1, lower_y = 0, dataList, StanS4class)
```

Arguments

a	See paper for definition.
b	See paper for definition.
mesh.for.drawing.curve	An integer indicating number of dots drawing the curves, default =10000.

upper_x The frame size of drawing picture.

upper_y The frame size of drawing picture.

lower_y The frame size of drawing picture.

dataList This is a variable in the function `rstan::sampling()` in which it is named `data`.

For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
           h = c(41,22,14,8,1),
           f = c(1,2,5,11,13),
           NL = 124,
           NI = 63,
           C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

———— Single reader and single modality (SRSC) case. ————

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence

levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector.

Example data:

A single reader and single modality case

NI=63,NL=124 In R console ->	**confidence level** c	**No. of false alarms** f	**No. of hits** h
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`

`m` A positive integer vector, representing the **modality** ID vector.

`q` A positive integer vector, representing the **reader** ID vector.

`h` A positive integer vector, representing the number of **hits** vector.

`f` A positive integer vector, representing the number of **false alarm** vector.

`NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	**ModalityID** m	**Confidence levels** c	**No. of false alarms** f	**No. of hits**. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14
1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

`StanS4class` This is an R object of class `stanfitExtended` inherited from the S4 class `stanfit`.

give_name_srsc_CFP_CTP_vector *Give a Name For CTP CFP vector*

Description

Give a Name for a vector representing cumulative true positives (CTPs) or cumulative false positives (CFPs).

Usage

```
give_name_srsc_CFP_CTP_vector(vector, CFP.or.CTP = "CFP",
  ModifiedPoisson = FALSE)
```

Arguments

vector	A vector representing cumulative true positives (CTPs) or cumulative false positives (CFPs).
CFP.or.CTP	"CFP" or "CTP". Default value is "CFP".
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.

Details

Some function in this package give the return values of vectors representing the CFP or CTPs. Using this function, we specify what the components of vector means. This is important since its order is not deterministic, that is, its order give two case, one is decreasing and one is increasing order. So, to avoid such confusion, the name should be specified. Of course this function is no needed for user to know or to use it.

Value

A vector representing cumulative true positives (CTPs) or cumulative false positives (CFPs) with its name.

Examples

```
h <- BayesianFROC::dataList.Chakra.1$h

NL <- BayesianFROC::dataList.Chakra.1$NL

CTP.vector <- cumsum(h)/NL

CTP.vector.with.name <- give_name_srsc_CFP_CTP_vector(CTP.vector)
```

give_name_srsc_data *Give a name for srsc data list component*

Description

By specifying the data,the names are given for each component vectors.

Usage

```
give_name_srsc_data(dataList)
```

Arguments

dataList

This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
           h = c(41,22,14,8,1),
           f = c(1,2,5,11,13),
           NL = 124,
           NI = 63,
           C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

———— Single reader and single modality (SRSC) case. ————

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

`f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.

`h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.

`NL` A positive integer, representing Number of Lesions.

`NI` A positive integer, representing Number of Images.

`C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	**ModalityID** m	**Confidence levels** c	**No. of false alarms** f	**No. of hits**. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14
1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13

2	2	4	2	4
2	3	3	1	2

Details

This is only available on singler reader and single modality case, not available on MRMC case.

Examples

```
#>dataList.Chakra.2
#$f
#[1]  4 13 44
#
#$h
#[1] 122  31  20
#
#$NL
#[1] 269
#
#$NI
#[1] 57
#
#$C
#[1] 3

   dataList.with.name <- give_name_srsc_data(dataList.Chakra.2)
```

```
#> dataList.with.name
# $f
# f(3) f(2) f(1)
# 4   13   44
#
# $h
# h(3) h(2) h(1)
# 122   31   20
#
# $NL
# Number of Lesions
# 269
#
# $NI
# Number of Images
# 57
#
```

```
# $C
# Number of Confidence levels
# 3



# dottest
```

`hits_creator_from_rate` *MRMC Dataset Creator From Hit Rate.*

Description

From hit rates, data of hits are created.

Usage

```
hits_creator_from_rate(NL = 252, seed = 123,
  p.truth = BayesianFROC::p_truth)
```

Arguments

NL	Number of Lesions.
seed	The seed for creating hits which are generated by the binomial distributions with the specified seed.
p.truth	Array of dimension (C, M, Q), where C = number of confidence levels, M = number of modalities, Q = number of readers.

Value

Hits Data

Examples

```
#  Using the default hit values, hit data are created as follows;

           hits <- hits_creator_from_rate()


# If user want to use their own hit rates, then please input as same as
# the default values manner.
#================The third example=====================================

#     The hits rate cannot take any values, since there is a trend that a hit rate of
#   a higher confidence level is a higher. So, If it is difficult for user to create
#   a true hit rates, then  by taking estimates as true parameters,
```

```
#   user can replicate datasets.
#     To do so, work follow is first fitting, secondly extracting estimates,
#   thirdly apply this function (hits_creator_from_rate() ).


# * Fitting

     fit <- fit_Bayesian_FROC(
             dataList.Chakra.Web.orderd,
             ite = 1111,  #  For simplicity, we take small MCMC samples.
             summary =FALSE)

# * Extracting

        estimates <- extract_estimates_MRMC(fit)

         ppp <- estimates$ppp.EAP

#      Note that ppp is an array
#    whose dimension is constituted by number of confidence levels, modalities, readers.


# *  Replicating as an true values is ppp


        hits  <-   hits_creator_from_rate(p.truth = ppp )


# <<Remark>>
#     ppp is an array. But ignoring its dimension, we can write that

#          hits ~ Binomial(ppp, NL)

#    Where NL is a number of lesions.

#   By writing its component explicitly, we can write

#        Hits[c,m,r] ~ Binomial(ppp[c,m,r], NL)

#        Where c means the c-th confidence level,
#              m  means the m-th modality,
#              r means the r-th reader.
# dottest
```

`hits_false_alarms_creator_from_thresholds`
Hits and False Alarms Creator

Description

From the parameter of the bi-normal assumptions, hits and false alarms are generated.

Usage

```
hits_false_alarms_creator_from_thresholds(replicate.datset = 3,
  ModifiedPoisson = FALSE, mean.truth = 0.6, sd.truth = 5.3,
  z.truth = c(-0.8, 0.7, 2.38), NL = 259, NI = 57, summary = TRUE,
  initial.seed = 12345)
```

Arguments

replicate.datset	A Number indicate that how many you replicate dataset from user's specified dataset.
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
mean.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
sd.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
z.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
NL	Number of Lesions.
NI	Number of Images.
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.
initial.seed	Replicated datasets are created using a continuous sequence of seeds and its initial seed is specified by this argument. For example, if you choose initial.seed =12300, then the replicated datasets are created from using the sequence of seeds: 12301,12302,12303,12304,...

Details

From the fixed parameters of bi-normal assumptions, we replicate data, that is, we draw the data from the distributions whose parameters are known. Especially, we interest the hits and false alarms since the number of images, lesions and confidence level is same for all replications. So, it is sufficient to check the hits and false alarms.

Value

Datasets Including Hits and False Alarms

Examples

```
#================The first example=====================================
#      Replication of Data from Fixed ( specified) Parameters.
```

```
 a <- hits_false_alarms_creator_from_thresholds(replicate.datset = 1)

#  Extract the first replicated dataset:

 a[[1]]$NL
 a[[1]]$NI
 a[[1]]$f
 a[[1]]$h
 a[[1]]$C


#================The second example=====================================
#      Replication of Data from Fixed ( specified) Parameters.

 b <- hits_false_alarms_creator_from_thresholds(replicate.datset = 2)


#  Extract the first replicated dataset:

 b[[1]]$NL
 b[[1]]$NI
 b[[1]]$f
 b[[1]]$h
 b[[1]]$C


#  Extract the second replicated dataset:

 b[[2]]$NL
 b[[2]]$NI
 b[[2]]$f
 b[[2]]$h
 b[[2]]$C


#================The Third example=====================================
#      Replication of Data from Fixed ( specified) Parameters.


 c <- hits_false_alarms_creator_from_thresholds(replicate.datset = 3)


#  Extract the first replicated dataset:

 c[[1]]$NL
 c[[1]]$NI
 c[[1]]$f
 c[[1]]$h
 c[[1]]$C


#  Extract the second replicated dataset:
```

```
 c[[2]]$NL
 c[[2]]$NI
 c[[2]]$f
 c[[2]]$h
 c[[2]]$C

#  Extract the third replicated dataset:

 c[[3]]$NL
 c[[3]]$NI
 c[[3]]$f
 c[[3]]$h
 c[[3]]$C



# dottest
```

`hits_from_thresholds` *MRMC Hit Creator from thresholds, mean and S.D.*

Description

From threshold, mean and S.D., data of hit rate are created.

Usage

```
hits_from_thresholds(z.truth = BayesianFROC::z_truth,
  mu.truth = BayesianFROC::mu_truth, v.truth = BayesianFROC::v_truth,
  NL = 252, seed = 123)
```

Arguments

`z.truth`	Vector of dimension = C represents the thresholds of bi-normal assumption.
`mu.truth`	array of dimension (M,Q). Mean of represents the signal distribution of bi-normal assumption.
`v.truth`	array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.
`NL`	Number of Lesions.
`seed`	The seed for creating hits which are generated by the binomial distributions with the specified seed.

Value

Hits Data for MRMC. The reason that hits is multiple reader and multiple modalities arise from the multiple indices of mean and S.D. of signal distribution of the bi-normal assumption.

Examples

```
hits.rate.p <-hits_from_thresholds()


#donttest
```

`hits_rate_creator` *MRMC Hit Rates Creator from Thresholds, Mean and S.D.*

Description

From thresholds, data of hit rate are created.

Usage

```
hits_rate_creator(z.truth = BayesianFROC::z_truth,
  mu.truth = BayesianFROC::mu_truth, v.truth = BayesianFROC::v_truth)
```

Arguments

`z.truth`	Vector of dimension = C represents the thresholds of bi-normal assumption.
`mu.truth`	array of dimension (M,Q). Mean of represents the signal distribution of bi-normal assumption.
`v.truth`	array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.

Value

ppp:= hit rate, which is an array with dimension

Examples

```
#================The first example=====================================

#     Using default values for hit rates, we can create a data of hits as follows:

        hits.rate <-hits_rate_creator()

#================The second example=====================================

#     Using the hit rate from the hits_rate_creator(), we can get the hits data:

        hits_creator_from_rate(p.truth =hits_rate_creator() )

#================The remark for example=================================
```

```
# The author does not show how to specify the hit rates or threshods.
# For the details of it, please see the default values of such a quantities.


#================The 4-th example=======================================

    p.truth.array <- hits_rate_creator()

# dottest
```

`initial_values_specification_for_stan_in_case_of_MRMC` *Initial values for HMC (Hamiltonian Moncte Carlo Markov Chains)*

Description

Internal function. Should not be interested.

Usage

```
initial_values_specification_for_stan_in_case_of_MRMC(dataList)
```

Arguments

`dataList`	This is a variable in the function `rstan::sampling()` in which it is named `data`. For the single reader and single modality data, the `dataList` is the following forms:

```
dataList.Example <- list(
          h = c(41,22,14,8,1),
          f = c(1,2,5,11,13),
          NL = 124,
          NI = 63,
          C = 5)
```

And using this object `dataList.Example`, we can apply `fit_Bayesian_FROC()` such as `fit_Bayesian_FROC(dataList.Example)`.

To make this R object `dataList`, this package provide three functions: please use one of the following codes to obtain an R object representing FROC data:

`convertFromJafroc()` If data is a **JAFROC xlsx** formulation.

`dataset_creator_new_version()` Enter TP and FP data **by table**.

`create_dataset()` Enter TP and FP data by **interactive** manner.

This package includes FROC datasets. Before running the function, we can confirm dataset is correctly formulated by the function `viewdata`.

-

———— Single reader and single modality (SRSC) case. ————

In single reader and single modality case (srsc), it should be a list which includes `f, h, NL, NI, C`. This list contains the following numeric vectors `f, h` and numerics `NL, NI, C`:

- `f` Non-negative integer vector specifying number of False Alarms associated with each confidence level. The first component corresponding to the highest confidence level.
- `h` Non-negative integer vector specifying number of Hits associated with each confidence level. The first component corresponding to the highest confidence level.
- `NL` A positive integer, representing Number of Lesions.
- `NI` A positive integer, representing Number of Images.
- `C` A positive integer, representing Number of Confidence level.

The detail of these dataset, see the datasets endowed with this package. 'Note that the maximal number of confidence level, denoted by `C`, are included, however, Note that confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` in the program and do not refer from user input data, where `C` is the highest number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created `c` vector.

Example data:

A single reader and single modality case

`NI=63,NL=124` In R console ->	**confidence level** `c`	**No. of false alarms** `f`	**No. of hits** `h`
definitely present	5	1	41
probably present	4	2	22
equivocal	3	5	14
perhaps present	2	11	8
questionable	1	13	1

* *false alarms* = False Positives = FP

* *hits* = True Positives = TP

Note that in FROC data, all confidence level means *present* (*deseased, lesion*) case only, no confidence level indicating absent.. Since each reader marks their suspicous location only and it generate the hits and false alarms, *thus* his confidenc level representing that lesion is *present*. In the absent case, reader dose not mark any locations and hence, the absent cofidence level does not relate this

dataset. So, if reader think it is no lesion, then in such case confidence level is not needed.

Note that the first column of confidence level vector `c` should not be specified. If specified, will be ignored , since it is created by `c <-c(rep(C:1))` automatically in the program and do not refer from user input data even if it is specified explicitly, where `C` is the highest number of confidence levels. So you should check the compatibility of your data and the program's generating new confidence level vector by a table which can be displayed by the function `viewdata()`.

Multiple readers and multiple modalities case, i.e., MRMC case

---

For multiple readers and multiple modalities case, i.e., MRMC case, to apply the function `fit_Bayesian_FROC()`, an R list object representing FROC data must have components `m,q,c,h,f,NL,C,M,Q`:

- `C` A positive integer, representing the **highest** number of confidence level, this is a scalar.
- `M` A positive integer vector, representing the number of **modalities**.
- `Q` A positive integer, representing the number of **readers**.
- `c` A positive integer vector, representing the **confidence level**. This vector must be made by `rep(rep(C:1), M*Q)`
- `m` A positive integer vector, representing the **modality** ID vector.
- `q` A positive integer vector, representing the **reader** ID vector.
- `h` A positive integer vector, representing the number of **hits** vector.
- `f` A positive integer vector, representing the number of **false alarm** vector.
- `NL` A positive integer, representing the Total number of **lesions** for all images, this is a scalar.

The detail of these dataset, please see the example datasets (the section **See Also** in the below) in this package.

Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata()`.

Example data:

Multiple readers and multiple modalities case, i.e., MRMC case

ReaderID q	ModalityID m	Confidence levels c	No. of false alarms f	No. of hits. h
1	1	5	1	15
1	2	4	3	14
1	3	3	5	5
1	1	2	5	3
1	2	1	9	4
1	3	5	1	14

1	1	4	2	13
1	2	3	2	5
1	3	2	5	3
2	1	1	6	4
2	2	5	1	14
2	3	4	1	4
2	1	3	1	1
2	2	2	2	2
2	3	1	3	2
2	1	5	1	13
2	2	4	2	4
2	3	3	1	2

Details

This attempt failed, that is, I cannot specify the initial values so that the `rstan::sampling()` does not say the following:

Rejecting initial value:

Log probability evaluates to log(0), i.e. negative infinity.

Stan can't start sampling from this initial value.

Value

Initial values specification. See the detailed documentation for the init argument in `stan()`.

Examples

```
init <- initial_values_specification_for_stan_in_case_of_MRMC(dataList.Chakra.Web)


# Where init is the variable of the rstan::stan() or rstan::sampling()
```

`install_imports` *Installer.*

Description

This is an installer for required packages in this package. To install this package BayesianFROC, we use the package xlsx which require the Java. So, if use buy a new computer and and it does not have installed the Java, then please install Java.

Usage

```
install_imports()
```

`make_TeX` *Make a TeX file for summary*

Description

Under Construction... "This only inner funtion, in the future I run this in the `fit_Bayesian_FROC()`.

Usage

```
make_TeX()
```

Value

TeX file reflected the analysis

`metadata_srsc_per_image` *Create metadata for MRMC data.*

Description

From data of number of hits and false alarms, we calculate the number of cumulative false positives and hits. Since there are three subscripts, reader, modality, and image, we create array format and vector format etc...

Usage

```
metadata_srsc_per_image(dataList)
```

Arguments

dataList	it should include m,q,c,h,f,NL,C,M,Q which means from the right c means the confidence level is not required, however it is created by c <-c(rep(C:1)), where C is the number of confidence levels. So, you should write down your hits and false alarms vector so that it is compatible with this automatically created c vector. h means the number of hits f means the number of false alarm NL means the Total number of lesions for all images C means the highest number of confidence level The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by C, are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, you should confirm the orders by the function viewdata_MRMC.

Value

A metadata such as number of cumulative false alarms and hits to create and draw the curve.

Examples

```
#First, we prepare the data endowed with this package.

        dat  <- BayesianFROC::dataList.Chakra.Web



#Second, we run the stan funtion
#with data named "dat"  and the author's Bayesian model.


             metadata_srsc_per_image(dat)




#Now, we get a metadata.
# dottest
```

metadata_to_DrawCurve_MRMC

Create metadata for MRMC data

Description

From data of number of hits and false alarms, we calculate the number of cumulative false positives and hits. Since there are three subscripts, reader, modality, and image, we create array format and vector format etc...

Usage

```
metadata_to_DrawCurve_MRMC(StanS4class, mesh.for.drawing.curve = 5000)
```

Arguments

StanS4class	This is an output of rstan::stan.
mesh.for.drawing.curve	An integer indicating number of dots drawing the curves, default =10000.

Value

A metadata such as number of cumulative false alarms and hits to create and draw the curve.

metadata_to_fit_MRMC *Create metadata for MRMC data*

Description

From data of number of hits and false alarms, we calculate the number of cumulative false positives and hits. Since there are three subscripts, reader, modality, and image, we create array format and vector format etc...

Usage

```
metadata_to_fit_MRMC(dataList)
```

Arguments

dataList	it should include m,q,c,h,f,NL,C,M,Q which means from the right
	m means the modality ID vector
	q means the reader ID vector
	c means the confidence level
	h means the number of hits
	f means the number of false alarm

NL means the Total number of lesions for all images

C means the highest number of confidence level

M means the number of modalities

Q means the number of readers.

The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by C, are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, you should confirm the orders by the function viewdata_MRMC.

Value

A metadata such as number of cumulative false alarms and hits to create and draw the curve.

Examples

```
#First, we prepare the data endowed with this package.

        dat  <- get(data("dataList.Chakra.Web"))



#Second, we run the stan funtion
#with data named "dat"  and the author's Bayesian model.


             metadata_to_fit_MRMC(dat)




#Now, we get a metadata.

# dottest
```

---

mu_truth *Mean data of MRMC*

Description

For the default value of a variable of a function.

Details

Mean Rate data of some MRMC data to use as a default value of the function `hits_creator_from_rate`. This is an array obtained from estimates of some data contained in this package. To simulate a replication of dataset, the default values should be used from an actual values. Thus the author prepare this data.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

See Also

`hits_creator_from_rate`

---

pairs_plot_if_divergent_transition_occurred

Pairs plot for divergent transition

Description

If divergent transition occurs, the author often forget the variable par or pars. So, I made this to avoid such confusion.

Usage

```
pairs_plot_if_divergent_transition_occurred(StanS4class,
  character.representing.paramter = "z")
```

Arguments

`StanS4class`	This is an R object of class `stanfitExtended` inherited from the S4 class `stanfit`.
`character.representing.paramter`	Character, surrounded by "", indicating the paramter of model.

Examples

```
# Create a fitted model object of class stanfitExtended  inherited from stanfit.

 fit <- fit_Bayesian_FROC( ite  = 1111,
                           summary = FALSE,
                           cha = 1,
                           Null.Hypothesis = FALSE,
                           dataList = dd )
```

```
# Pairs plot to examine the divergent transition.

   pairs_plot_if_divergent_transition_occurred(fit)
```

---

pause *Pause for Demo*

Description

Pause for Demo

pause()

Usage

```
pause()
```

---

plot-methods *A generic function plot()*

Description

A generic function plot()

---

plotFROC *Draw FROC curves by two parameters a and b*

Description

Plot FROC curves based on two parameters a and b.

Usage

```
plotFROC(a, b, mesh.for.drawing.curve = 10000, upper_x = 1,
  upper_y = 1, lower_y = 0)
```

Arguments

a	See paper for definition.
b	See paper for definition.
mesh.for.drawing.curve	An integer indicating number of dots drawing the curves, default =10000.
upper_x	The frame size of drawing picture.
upper_y	The frame size of drawing picture.
lower_y	The frame size of drawing picture.

---

plot_test *# Definition of a method for the inherited class stanfitExtended from stanfit*

Description

This is a function for a method in the generic function plot.

Usage

```
plot_test(x)
```

Arguments

x	This is an object of an S4 class named stanfitExtended which is an inherited S4 class from the stanfit S4 class in the rstan package.

---

print *A method for a generic function* print() *for class "*stanfitExtended*"*

Description

This is a method for print and stanfitExtended S4 class.

Arguments

x	An S4 object of class stanfitExtended inherited from the class stanfit in the rstan package.

Examples

```
# How to use a new method for generic function "print".
#============================The First Example=====================================


#(1)First, we prepare the example data from this package.


                    dat  <- BayesianFROC::dataList.Chakra.1


# The R object named dat is a list which contains the hits and false alarms representing
# an FROC dataset. To confirm it, the function viewdata() can be used;


                           viewdata(dat)



#(2)Second, we run fit_Bayesian_FROC() in which the rstan::sampling() is implemented.
#Fit to data named "dat"   the author's Bayesian model by


                       fit <-  fit_Bayesian_FROC(dat)



#(3)Thirdly, we obtain the R object fit of  S4 class
# named stanfitExtended that is an inherited class from the  S4 class stanfit
# defined in the package rstan.
# For the S4 class stanfitExtended defined in this package, we can use
# the generic function print for this new S4 class.


                            print(fit)


# To use the generic functin print() as a  object of class "stanfit",
#  we coerce class of fit into stanfit from stanfitExtended as follows;
```

```
                              fitt <- methods::as(fit,"stanfit")



  # THe R object "fitt" is a fitted model object of class stanfit,
  # thus we can also apply the generic function print() as follows:


                                 print(fitt)


  #============================The Second Example=====================================


  #(1)First, we prepare the example data from this package.

                         dat  <- BayesianFROC::dataList.Chakra.Web


  #(2)Second, we run fit_Bayesian_FROC() in which the rstan::sampling() is implemented.
  #Fit to data named "dat"   the author's Bayesian model by


                           fit <-  fit_Bayesian_FROC(dat)

  #(3)Thirdly, we obtain the R object fit of  S4 class
  # named stanfitExtended that is an inherited class from the  S4 class stanfit
  # defined in the package rstan.
  # For the S4 class stanfitExtended defined in this package, we can use
  # the generic function print for this new S4 class.


                              print(fit)




  # 2019.05.21 Revised.


  # dottest
```

print_stanfitExtended *Definition of a method for the inherited class stanfitExtended from stanfit*

Description

This is a function for a method for a generic function `print()` for class "`stanfitExtended`"

Usage

```
print_stanfitExtended(x)
```

Arguments

x	This is an object of an S4 class named stanfitExtended which is an inherited S4 class from the stanfit S4 class in the rstan package.

p_truth *Hit Rate data*

Description

For the default value of a variable of a function.

Details

Hit Rate data of some MRMC data to use as a default value of the function `hits_creator_from_rate`. This is an array obtained from estimates of some data contained in this package. To simulate a replication of dataset, the default values should be used from an actual values. Thus the author prepare this data.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

See Also

`hits_creator_from_rate`

`p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit`
Get the Chi square values

Description

Get the Chi square values with indexed by the all possible pair of replicated data and MCMC samples.

Usage

```
p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(StanS4class,
  dig = 3, Colour = TRUE, plot.replicated.points = FALSE,
  head.only = FALSE)
```

Arguments

`StanS4class`	fitted model object, from an output of `rstan::sampling()`.
`dig`	An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,
`Colour`	logical: TRUE or FALSE. whether Colour of curves is dark theme or not.
`plot.replicated.points`	TRUE or FALSE. If true, then plot replicated points (hits, false alarms) by the scatter plot. This process will takes a long times. So if user has no time, then `FALSE` will help you.
`head.only`	Logical: `TRUE` of `FALSE`. Whether head part or entire. If `TRUE`, only head part are shown. Default is `FALSE`

Details

Here, we briefly review how to get the chi square samples in the Bayesian paradigm.

First, Let

$$f(y|\theta)$$

be a model (likelihood) with future data y and model parameter θ. Let

$$p(\theta|D)$$

be the posterior for given data D. In this situation, the Hamiltonian Monte Carlo method is performed to get the MCMC samples of size N, say

$$\theta_1, \theta_2, \theta_3, ..., \theta_N$$

from posterior $p(\theta|D)$ of given data D. Alternatively, we get the sequence of models

$$f(y|\theta_1), f(y|\theta_2), f(y|\theta_3), ..., f(y|\theta_N).$$

To get the samples

$$y1, y2, ..., yN$$

from the posterior probability distribution, we merely draw the $y1, y2, ..., yN$ from $f(y|\theta_1), f(y|\theta_2), f(y|\theta_3), ..., f(y|\theta_N)$, respectively. That is for all I y_i is drawn from the distribution $f(y|\theta_i)$. Once, we get the samples from the posterior predictive density, we can calculate an arbitrary integral with the posterior measure by the law of large number, or it sometimes is called MonteCarlo integral. Recall that the chi square goodness of fit statistics χ is dependent of the model parameter θ and data D. that is,

$$\chi = \chi(D|\theta)$$

. So, by integarating

$$\chi(D|\theta)$$

with the posterior predictive measure, we get the

$$\chi(D)$$

which depends only of the data D, that is,

p value :=

$$\chi(D) := \int 1[\chi(Data|\theta) > \chi(D|\theta)]f(\theta|Data)d\theta d(Data)$$

So, in the return value of this function is p value.

My hand, especially right has ache, so I quit this documentation, Good Luck, 2019 may 29. I do not have confidence whether my explanation sucess.

In this manner we get the two sequence of samples, one is from the posterior distribution and one is the posterior predictive distribution. Using these two kind of samples, we can calculate the test statistics as the Bayesian manner. That is, in frequentist method, the test statistics are calculated by the fixed model parameters, such as the maximal likelihood estimators. However, in Bayesian context, the parameter is not deterministic and hence we should calculate test statistics with the posterior measure. To accomplish this task, this package include the function.

Value

The main return is a nonnegative real number indicating p value of the Chi square goodness of fit. And the other components to calculate p values.

See Also

get_samples_from_Posterior_Predictive_distribution, chi_square_goodness_of_fit_from_input_all_param

Examples

```
# First, fit the model to data. The number of sampling of the Hamiltonian Monte Carlo
# methods should be a little number, if user computer has low ability,
# since the calculation of the posterior predictive p values is heavy.


 fit <- fit_Bayesian_FROC(BayesianFROC::dataList.Chakra.1 ,ite = 1111)
```

```
#  Next, extract the posterior predictive p value from the fitted model object "fit",
#  and to do so, we have to make a object "output".


 output <-  p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(fit)



# From the above R script, the table will appear in the R cosole.
# If the TRUE is more, then model fitting is better.
# Finaly, we obtain the following p value;


                p.value  <-   output$p.values.for.chisquare


#  The significant level of p value is 0.05 in frequentist paradium, but,
# In this p value I think it should be more greater, and
# should use e.g., 0.6 instead of 0.05 for significant level.
# If significant level is 0.5, then test

                       p.value > 0.5

# If it is FALSE, then the fitting is bad.
# If p value is more greater than the fitting is more better.


# If user has no time, then  plot.replicated.points=FALSE will help you.
# By setting FALSE, the replicated data from the posterior predictive
# distribution does not draw, and hence the running time of function become shorter.

 TPs.FPs <- p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(fit,
                                  plot.replicated.points =  FALSE)


#  If user want to use the scatter plots of hits and false alarms from the posterior
#  predictive distribution for the submission, then the color plot is not appropriate.
#  So, by setting the argument Colour = FALSE, the scatter plot become black and white.
#  So, user can use this scatter plot for submission.


 p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(fit,Colour = FALSE)



# Since p values are depend on data only, so it is better to show this dependency more
# explicitly as follows;

   p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit(
```

```
   fit_Bayesian_FROC(dataList.High)
    )



       #    Close the graphic device

       Close_all_graphic_devices()


# dottest
```

`rank_statistics_with_two_parameters` *Rank Statistics*

Description

Rank Statistics

Usage

```
rank_statistics_with_two_parameters(values.of.f.at.one.MCMC.samples,
  values.of.f.at.a.sample.from.priors)
```

Arguments

`values.of.f.at.one.MCMC.samples`	The value of f at a vector whose components are constructed by the all parameters at one MCMC sample.
`values.of.f.at.a.sample.from.priors`	The value of f at a vector of model parameters from the prior distribution.

Value

The value of the Rank Statistics

Examples

```
#======== The first example   ========================================

  rank_statistics_with_two_parameters(c(1,2,3,4,5),4)

#=======  The Second Example  ========================================

       a <- Draw_a_simulated_data_set_and_Draw_posterior_samples()
```

```
    rank_statistics_with_two_parameters(
         a$MCMC.samples.sended.by.fun,
         a$prior.samples.sended.by.fun
         )


# dottest
```

---

replicate_model_MRMC *Replicate Models*

Description

Replicate Models For Replicated Data From True Distributions.

Usage

```
replicate_model_MRMC(initial.seed = 123,
  mu.truth = BayesianFROC::mu_truth, v.truth = BayesianFROC::v_truth,
  z.truth = BayesianFROC::z_truth, NI = 200, NL = 142,
  ModifiedPoisson = FALSE, No.of.replication = 2, summary = FALSE,
  ite = 1111)
```

Arguments

initial.seed	The variable initial.seed is used to replicate datasets. That is, if you take initial.seed = 1234, then the seed 1234, 1235, 1236, 1237, 1238, ... are for the first replication, the second replication, the third replication, ... If the n-th model does not converge for some n, then such model has no mean and thus the non-convergent models are omitted to calculate the errors.
mu.truth	array of dimension (M,Q). Mean of represents the signal distribution of bi-normal assumption.
v.truth	array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.
z.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
NI	Number of Images.
NL	Number of Lesions.
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
No.of.replication	For fixed number of lesions, images, the dataset of hits and false alarms are replicated, and the number of replicated datasets are specified by this variable.

summary — Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

ite — An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

Examples

```
list.of.fit  <- replicate_model_MRMC()
```

---

replicate_MRMC_dataList

MRMC: Replicate Datasets

Description

From threshold, mean and S.D., it replicate datasets.

Usage

```
replicate_MRMC_dataList(initial.seed = 123,
  mu.truth = BayesianFROC::mu_truth, v.truth = BayesianFROC::v_truth,
  z.truth = BayesianFROC::z_truth, NI = 200, NL = 142,
  ModifiedPoisson = FALSE, No.of.replication = 2, summary = FALSE)
```

Arguments

initial.seed — The variable initial.seed is used to replicate datasets. That is, if you take initial.seed = 1234, then the seed 1234, 1235, 1236, 1237, 1238, ... are for the first replication, the second replication, the third replication, ... If the n-th model does not converge for some n, then such model has no mean and thus the non-convergent models are omitted to calculate the errors.

mu.truth — array of dimension (M,Q). Mean of represents the signal distribution of bi-normal assumption.

v.truth — array of dimension (M,Q). Standard Deviation of represents the signal distribution of bi-normal assumption.

z.truth — This is a parameter of the latent Gaussian assumption for the noise distribution.

NI Number of Images.

NL Number of Lesions.

ModifiedPoisson
Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.

No.of.replication
For fixed number of lesions, images, the dataset of hits and false alarms are replicated, and the number of replicated datasets are specified by this variable.

summary Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

---

showGM *Show the Graphical Model for a single reader and single modality*

Description

This function shows the graphical model for a single reader and single modality FROC statistical model.

Usage

```
showGM()
```

Examples

```
  showGM()



# dottest
```

Simulation_Based_Calibration_histogram *Draw a histogram of the rank statistics*

Description

To validate that the MCMC procedure is correct or not, we show the histogram of rank statistics. If the resulting histogram is uniformly distributed, then we can conclude that the MCMC sampling is correct. If the histogram is far from uniformity, then the MCMC sampling or specification of priors is not correct or not appropriate.

Usage

```
Simulation_Based_Calibration_histogram(N = 3, sd = 5, C = 5,
  initial.seed.for.drawing.a.rank.statistics = 1234567,
  fun = stats::var, NI = 259, NL = 259,
  initial.seed.for.drawing.a.data = 1234, ModifiedPoisson = FALSE,
  ite = 1111, DrawCurve = FALSE)
```

Arguments

N	samples size of the rank statistics.
sd	Standard Deviation of priors
C	No. of Confidence levels
initial.seed.for.drawing.a.rank.statistics	seed
fun	An one dimensional real valued function defined on the parameter space. This is used in the definition of the rank statistics. Generally speaking, the element of the parameter space is a vector, so the function should be defined on vectors. In my model parameter is mean, standard deviation, C thresholds of the latent Gaussian, so this function should be defined on the C+2 dimensional Euclidean space.
NI	No. of images
NL	No. of Lesions
initial.seed.for.drawing.a.data	seed
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
ite	An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.
DrawCurve	Logical: TRUE of FALSE. Whether the curve is to be drawn. TRUE or FALSE. If you want to draw the FROC and AFROC curves, then you set DrawCurve =TRUE, if not then DrawCurve =FALSE. The reason why the author make this variable DrawCurve is that it takes long time in MRMC case to draw curves, and thus default value is FALSE.

Value

samples of rank statistics

Examples

```
## Not run:
  g <-Simulation_Based_Calibration_histogram(N=2,ite = 2222)

  graphics::hist(g$rank.statistics)



  g <- Simulation_Based_Calibration_histogram(
  NI=1111111,
  NL=1111111,
  # N =100 would be better  more than N =10
  # But this is only example, we take very small N
  N=10,
  ite=3333,
  sd=1,
  initial.seed.for.drawing.a.rank.statistics = 123456789,
  DrawCurve = T
  )



   g <- Simulation_Based_Calibration_histogram(
   NI=1111111,
   NL=1111111,
  # N =100 would be better  more than N =10
  # But this is only example, we take very small N
   N=10,
   ite=3333,
   sd=1,initial.seed.for.drawing.a.rank.statistics = 123456789,
   DrawCurve = T,
   C=11)
#=======      The Second Example:      ================================================

# If you want to see the replicated data, then the following code is available.
# In the following, I extract the dataset which is very small rank statistics, e.g.
# less than 10. And draw the CFP and CTP for observation of dataset.



gggg <- Simulation_Based_Calibration_histogram(
NI=1111111,
NL=1111111,
N=22,
ite=2222)


 a <- gggg$rank.statistics<10

 aa <- the_row_number_of_logical_vector(a)
```

```
  draw.CFP.CTP.from.dataList(gggg$fit.list[[  aa[1]  ]]@dataList)


## End(Not run)#\dontrun
```

`snippet_for_BayesianFROC` *Edit Snippet*

Description

Snippet for the package BayesianFROC. Copy and paste to the snippet edition tools in your R studio for the conforable usage of the package BayesianFROC. This is under construction. To edit snippet, you should open the editor located in Tools > Global options > Code > Edit snippets.

Usage

```
snippet_for_BayesianFROC()
```

Value

nothing

snippet_for_BayesianFROC()

`specify_priors` *Define priors*

Description

Specification of the support for priors

Usage

```
specify_priors()
```

Value

numerical vector indicating the characteristics of the priors.

stanfitExtended *stanfitExtended (S4 class)*

Description

Inherits from the class stanfit.

Note that stanfit is an S4 class defined in the package ***rstan*** :

Slots

plotdataMRMC Plot data for MRMC case.

plotdata This is a data frame with four components which is used to draw curves such as FROC curves and AFROC curves.

dataList This is a dataset. Using the dataset, the fitting has done.

studyDesign This is character, e.g., "srsc.per.image", "srsc.per.lesion",

metadata This is additional data calculated from dataList, such as cumulative hits and false alarms,...,etc.

WAIC This is a WAIC calculated by the function waic .

convergence This is TRUE or FALSE. If TRUE, then it means your model is good in the R hat criterion.

PreciseLogLikelihood This is TRUE or FALSE. If TRUE, then target formulation is used in the stan file.

chisquare This is a chi square calculated with Expected A Posterior estimates, i.e., the posterior mean estimates. Note that this is *not* calculated by integrating the posterior predictive measure. Do not confuse with the p value calculated with the posterior predicitive measure implemented in the function p_value_of_the_Bayesian_sense_for_chi_square_goodness_of_fit()

index THis is for programming phase.

Divergences This is a number of the divergence transitions in the MCMC simulation.

MCMC.Iterations A MCMC iterations which does not count the burn-in period.

Divergence.rate A divergence rate, that is the number of the divergence iterations over total MCMC iterations. Burn-in period is not included.

model_name This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

model_pars This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

par_dims This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

mode This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

sim This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

inits This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

stan_args This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

stanmodel This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

date This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

.MISC This is a slot from the stanfit which is an S4 class defined in the ***rstan*** package.

StatisticForANOVA *Statistic for ANOVA*

Description

Provides a statistic to test the null hypothesis that all modalities are same.

Usage

```
StatisticForANOVA()
```

Value

None

summarize_MRMC *Summarize the estimates for MRMC case*

Description

Summarize the estimates for MRMC case

Usage

```
summarize_MRMC(StanS4class, dig = 3)
```

Arguments

StanS4class	fitted model object, from an output of rstan::sampling().
dig	An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

Value

Nothing

Examples

```
fit <- fit_Bayesian_FROC(
        dataList.Chakra.Web.orderd,
        ite = 1111,
        summary =FALSE
        )

summarize_MRMC(fit)


# dottest
```

`summary_AUC_comparison_MRMC` *Print summary for AUC comparisons for MRMC*

Description

This is print the results of AUC comparison for MRMC data.

Usage

```
summary_AUC_comparison_MRMC(StanS4class, significantLevel = 0.8,
  dig = 3)
```

Arguments

`StanS4class`	This is an R object of class `stanfitExtended` inherited from the S4 class `stanfit`.
`significantLevel`	This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.
`dig`	An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

---

`summary_AUC_comparison_MRMC_without_crayon` *Print summary for AUC comparisons for MRMC without color*

Description

This is print the results of AUC comparison for MRMC data.

Usage

```
summary_AUC_comparison_MRMC_without_crayon(StanS4class,
  significantLevel = 0.8, dig = 3)
```

Arguments

StanS4class	This is an R object of class stanfitExtended inherited from the S4 class stanfit.
significantLevel	This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.
dig	An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

---

summary_AUC_comparison_MRMC_with_crayon
Print summary for AUC comparisons for MRMC hier with color

Description

This is print the results of AUC comparison for MRMC data.

Usage

```
summary_AUC_comparison_MRMC_with_crayon(StanS4class,
  significantLevel = 0.8, dig = 3)
```

Arguments

StanS4class	This is an R object of class stanfitExtended inherited from the S4 class stanfit.
significantLevel	This is a number between 0 and 1. The results are shown if posterior probabilities are greater than this quantity.
dig	An argument of rstan::sampling() in which it is named iter. A positive integer representing the Significant digits, used in stan Cancellation. default = 5,

summary_EAP_CI_srsc *Summary*

Description

EAP and CI

Usage

```
summary_EAP_CI_srsc(StanS4class, dig = 5, summary = TRUE)
```

Arguments

StanS4class	This is an output of rstan::stan for a single reader and a single modality. More precisely, this is an object of some inherited class from the S4 class called stanfit in the rstan package.
dig	digits of estimates.
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Value

The estimates

Examples

```
#================The first example=======================================


#1) Build the data for singler reader and single modality  case.

dat <- list(c=c(3,2,1),    #Confidence level
            h=c(97,32,31), #Number of hits for each confidence level
            f=c(1,14,74),  #Number of false alarms for each confidence level

            NL=259,       #Number of lesions
            NI=57,        #Number of images
            C=3)          #Number of confidence level

# where, c denotes Confidence level,
#        h denotes number of Hits for each confidence level,
#        f denotes number of False alarms for each confidence level,
#        NL denotes Number of Lesions,
#        NI denotes Number of Images,
```

```
#2) Fit the FROC model.
  #Since dataset named dat are single reader and single modality,
  #the function build the such model by running the following code.

          fit <-   BayesianFROC::fit_Bayesian_FROC(dat)

 #3) Extract estimates, that is posterior means and 95% credible intervals


        estimates <- summary_EAP_CI_srsc(  fit )


# dottest
```

`Test_Null_Hypothesis_that_all_modalities_are_same`
Test the Null hypothesis that all modalities are same

Description

Test null hypothesis that all modalities have same observer performance ablity, using Bayes factor.

@details From input data `dataList`, the two objects of class `stanfit` are created. one is fitted to the null hypothesis model and the another one representing alternative hypothesis. These two `stanfit`. objects are compared based on Bayes factor.

Usage

```
Test_Null_Hypothesis_that_all_modalities_are_same(dataList, ite = 111,
  cha = 1, summary = FALSE)
```

Arguments

`dataList`	MRMC case only.
`ite`	An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.
`cha`	An argument of `rstan::sampling()` in which it is named `chains`. A positive integer representing the number of chains generated by Hamiltonian Monte Carlo method, and, default = 1.
`summary`	Logical: `TRUE` of `FALSE`. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Value

none

Examples

```
Test_Null_Hypothesis_that_all_modalities_are_same(BayesianFROC::dd)

#donttest
```

`the_row_number_of_logical_vector` *Extract the row number from a logical vector*

Description

Extract the row number from a logical vector

Usage

```
the_row_number_of_logical_vector(vector.logical)
```

Arguments

`vector.logical`	vector with logical component

Value

the row number of logical component

Examples

```
a <-c(TRUE,FALSE,FALSE,TRUE)

the_row_number_of_logical_vector(a)
```

validation.dataset_srsc *Dataset creator from known distributions*

Description

By specifying the parameters of bi-normal assumptions, (that is, mean and sd of the signal latent Gaussian distribution) the dataset are created from this known distributions. The number of lesions and the number of images corresponds the sample size of each replicated dataset.

This function also fit a srsc FROC model for each replicated dataset.

Usage

```
validation.dataset_srsc(replicate.datset = 3, ModifiedPoisson = FALSE,
  mean.truth = 0.6, sd.truth = 5.3, z.truth = c(-0.8, 0.7, 2.38),
  NL = 259, NI = 57, ite = 1111, summary = TRUE)
```

Arguments

`replicate.datset`	A Number indicate that how many you replicate dataset from user's specified dataset.
`ModifiedPoisson`	Logical, that is `TRUE` or `FALSE`. If `ModifiedPoisson = TRUE`, then Poisson rate of false alarm are `per lesion`, and if `ModifiedPoisson = FALSE`, then Poisson rate of false alarm are `per image`. To know detail, refer the author's paper in which I explained per image and per lesion.
`mean.truth`	This is a parameter of the latent Gaussian assumption for the noise distribution.
`sd.truth`	This is a parameter of the latent Gaussian assumption for the noise distribution.
`z.truth`	This is a parameter of the latent Gaussian assumption for the noise distribution.
`NL`	Number of Lesions.
`NI`	Number of Images.
`ite`	An argument of `rstan::sampling()` in which it is named `iter`. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.
`summary`	Logical: `TRUE` of `FALSE`. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Value

Return values is,

stanfit objects, more precisely some inherited class, for each replicated dataset.

Examples

```
#================The first example=====================================



#   Using the default values, the code run, i.e.:

   datasets <- validation.dataset_srsc()



#================The second example=====================================

#   If user do not familiar with the values of thresholds, then
#   it would be better to use the actual estimated values
#    as an example of true parameters. In the following,
#     I explain this.

# First, get estimators by

  fit <- fit_Bayesian_FROC(dataList.Chakra.1,ite = 1111,summary =FALSE,cha=3)

#  Secondly, extract the expected a posterior estimators (EAPs) from the object fit

  z <- rstan::get_posterior_mean(fit,par=c("z"))[,"mean-all chains"]

#  Thirdly we use this z as a true values. If user does not know
#   the scale or ordinal values of thersholds, by using fitted values,
#     user can use this function to replicate datasets.

   datasets <- validation.dataset_srsc(z.truth = z)



# dottest
```

`validation.dataset_srsc_for_different_NI_NL`
Dataset creator from known distributions.

Description

By specifying the parameters of binormal assumptions, the dataset are created from this known distributions.

Usage

```
validation.dataset_srsc_for_different_NI_NL(NLvector = c(100, 10000,
```

```
1e+06, 1e+08), ratio = 2, replicate.datset = 3,
ModifiedPoisson = FALSE, mean.truth = 0.6, sd.truth = 5.3,
z.truth = c(-0.8, 0.7, 2.38), ite = 111)
```

Arguments

NLvector	Vector indicates the number of Lesions.
ratio	Number of Images is determined by converting ratio * NLvector to a integer.
replicate.datset	A Number indicate that how many you replicate dataset from user's specified dataset.
ModifiedPoisson	Logical, that is TRUE or FALSE. If ModifiedPoisson = TRUE, then Poisson rate of false alarm are per lesion, and if ModifiedPoisson = FALSE, then Poisson rate of false alarm are per image. To know detail, refer the author's paper in which I explained per image and per lesion.
mean.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
sd.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
z.truth	This is a parameter of the latent Gaussian assumption for the noise distribution.
ite	An argument of rstan::sampling() in which it is named iter. A positive integer representing the number of samples generated by Hamiltonian Monte Carlo method, and, default = 10000. If your model could not converge, then raise this number. Must be greater for more reliable estimates.

Examples

```
  datasets <-validation.dataset_srsc_for_different_NI_NL(
              NLvector = c(100,10000000,1000000000),
              ite = 2222
              )


 # By the following, we can extract only datasets whose
 # model has converged.
   datasets$convergent.dataList.as.dataframe

# dottest
```

---

validation.draw_srsc *Draw Curves for validation dataset*

Description

drawing curves.

Usage

```
validation.draw_srsc(validation.data, mesh.for.drawing.curve = 11111,
  upper_y = 1, DrawFROCcurve = TRUE)
```

Arguments

validation.data	This is a return value of the function validation.dataset_srsc.
mesh.for.drawing.curve	An integer indicating number of dots drawing the curves, default =10000.
upper_y	This is a upper bound for the axis of the vertical coordinate of FROC curve.
DrawFROCcurve	logical: TRUE or FALSE. Whether or not FROC curves are shown.

Examples

```
    datasets <- validation.dataset_srsc()

    validation.draw_srsc(datasets)

# dottest
```

---

viewdata *Build a table of FROC data*

Description

Create a tabular representation of FROC data from FROC data object.

Usage

```
viewdata(dataList, summary = TRUE, head.only = FALSE)
```

Arguments

dataList	———————————————— Single reader and single modality (SRSC) case. ———————————————— In single reader and single modality case, it should include f, h, NL, NI, C. The detail of these dataset, see the datasets endowed with this package. Note that the maximal number of confidence level, denoted by C, are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, you should confirm the orders by the function

---

Multiple readers and multiple modalities case, i.e., MRMC case

---

In multiple readers and multiple modalities case, i.e., MRMC case, it should include `m,q,c,h,f,NL,C,M,Q` which means from the right

`C` means the highest number of confidence level, this is a scalar.

`M` means the number of modalities

`Q` means the number of readers.

`c` means the confidence level vector. This vector must be made by `rep(rep(C:1), M*Q)`.

`m` means the modality ID vector.

`q` means the reader ID vector.

`h` means the number of hits vector.

`f` means the number of false alarm vector.

`NL` means the Total number of lesions for all images, this is a scalar.

The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by `C`, are included, however, its each confidence level vector also created in the program by `C`. So, to confirm your false positives and hits are correctly correspond to confidence levels, you should confirm the orders by the function `viewdata_MRMC`.

`summary` TRUE or FALSE, if true then results are printed, if FALSE this function do nothing.

`head.only` Logical: `TRUE` of `FALSE`. Whether head part or entire. If `TRUE`, only head part are shown. Default is `FALSE`

Value

Nothing

In order to confirm your data, please view table before fitting. Confidence level vector are created in my program regardless of user's confidence level vectors.

Examples

```
# The first example, we prepare the data endowed with this package.
             dat  <- get(data("dataList.Chakra.1"))



             viewdata(dat)


# The second examle, we consider the multiple reader and multiple dataset

             dat <-  get(data("dataList.Chakra.Web"))
```

```
viewdata(dat)

# dottest
```

---

viewdata_MRMC *View MRMC data*

Description

Build a table for data dataList.

Usage

```
viewdata_MRMC(dataList, summary = TRUE, head.only = FALSE)
```

Arguments

dataList	it should include m,q,c,h,f,NL,C,M,Q which means from the right m means the modality ID vector q means the reader ID vector c means the confidence level h means the number of hits f means the number of false alarm NL means the Total number of lesions for all images C means the highest number of confidence level M means the number of modalities Q means the number of readers. The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by C, are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, you should confirm the orders by the function viewdata_MRMC.
summary	TRUE or FALSE, if true then results are printed, if FALSE this function do nothing.
head.only	Logical: TRUE of FALSE. Whether head part or entire. If TRUE, only head part are shown. Default is FALSE

viewdata_srsc *Build a table of data in the case of Single reader and Single modality (srsc)*

Description

In order to confirm your data, please view table. my program makes new column of confidence levels which are used in my program. So, it is possible that your order of confidence level and Program's order of confidence level are inverse. This function's result table are the one which are used in program.

Usage

```
viewdata_srsc(dataList, summary = TRUE)
```

Arguments

dataList	it should include f, h, NL, NI, C. The detail of these dataset, please see the endowed datasets. Note that the maximal number of confidence level, denoted by C, are included, however, its each confidence level should not included your data. So, to confirm your false positives and hits are correctly correspondence to confidence levels, user should confirm the orders by the function.
summary	TRUE or FALSE, if true then results are printed, if FALSE this function do nothing.

Examples

```
#   First, we prepare an example FROC data "dataList.Chakra.1" in this package.
# Note that this data should be formed as a single reader and single case (modality).
# If your data are a multiple reader and multiple case (modality), i.e.,MRMC-data,
# then another function named viewdataMRMC is available for MRMC-data.

             dat  <- get(data("dataList.Chakra.1"))




# Show data named "dat";


             viewdata_srsc(dat)



#The Reason why the author made this \code{viewdata_srsc} is
#the code does not refer your confidence level.
#More precisely, my program made the column vector of confidence levels
#from the its highest number,
```

```
#so, it may be occur the interpretaion of code for hits and false alarm
#are inverse order compared with your data.


# dottest
```

---

v_truth *Standard Deviation data of MRMC*

Description

For the default value of a variable of a function.

Details

Standard Deviation Rate data of some MRMC data to use as a default value of the function `hits_creator_from_rate`. This is an array obtained from estimates of some data contained in this package. To simulate a replication of dataset, the default values should be used from an actual values. Thus the author prepare this data.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

See Also

`hits_creator_from_rate`

---

waic *WAIC calculator*

Description

Using the fitted object of class satnfit whose stan file described using `target +=` , the function calculates the WAIC.

Usage

```
waic(StanS4classwithTargetFormulation, dig = 4, summary = TRUE)
```

Arguments

StanS4classwithTargetFormulation	This is a fitted model object built by stan whose model block is described by target formulation function in the rstan package. This object is avaliable both S4 class, stanfit and stanfitExtended. In this package, we make a new S4 class "stanfitExtended" which is inherited class of rstan's S4 class named "stanfit". This function is available for stanfit S4 object.
dig	The number of significant digits of waic.
summary	Logical: TRUE of FALSE. Whether to print the verbose summary, i.e., logical; whether to list verbose data of estimates when printing. If TRUE then verbose summary and additional estimates are printed in the R console. If FALSE, the output is minimal. I regret, this variable name should be verbose.

Value

A real number, representing the value of WAIC.

Examples

```
#First, we prepare the data endowed with this package:

        dat  <- get(data("dataList.Chakra.1"))



#Second,  create a fitted model object;

            fit <- fit_Bayesian_FROC(dat, PreciseLogLikelihood = TRUE)


#Using the fitted model object "fit", we obtain the WAIC


                waic(fit)


#The Author provide two model for FROC for single reader and single modality case.
#One is false alarm rates means "per lesion" and the other means "per image".
#The above "fit" is "per image". Now we shall consider to compare these two model
#by WAIC. To do so, next we shall fit the "per lesion" model as follows:

fit2 <- fit_Bayesian_FROC(dat, PreciseLogLikelihood = TRUE, ModifiedPoisson=TRUE)

waic(fit2)
```

```
# By compare two model's WAIC we can say which model is better.
# Note that the smaller WAIC is better.

waic(fit)     # per lesion model
waic(fit2)    # per image model



# 2019.05.21 Revised.
# dottest
```

---

z_truth *Threshold data of MRMC*

Description

For the default value of a variable of a function.

Details

Threshold Rate data of some MRMC data to use as a default value of the function `hits_creator_from_rate`. This is an array obtained from estimates of some data contained in this package. To simulate a replication of dataset, the default values should be used from an actual values. Thus the author prepare this data.

Author(s)

Issei Tsunoda <tsunoda.issei1111@gmail.com >

See Also

`hits_creator_from_rate`

Index