Package ‘jfa’

March 23, 2021

Title Bayesian and Classical Audit Sampling

Version 0.5.2

Date 2021-03-15

Description Implements the audit sampling workflow as discussed in Derks et al. (2019) <doi:10.31234/osf.io/9f6ub>. The package makes it easy for an auditor to plan an audit sample, sample from the population, and evaluating that sample using various confidence bounds according to the International Standards on Auditing. Furthermore, the package implements Bayesian equivalents of these methods.

BugReports https://github.com/koenderks/jfa/issues

URL https://github.com/koenderks/jfa, https://koenderks.github.io/jfa/

Suggests testthat, knitr, rmarkdown, kableExtra

Language en-US

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

R topics documented:

auditPrior *Create a Prior Distribution with Existing Audit Information*

Description

This function creates a prior distribution with audit information to be used in the `planning()` and `evaluation()` functions via their `prior` argument. The returned object is of class `jfaPrior` and has a `print()` and `plot()` method.

For more details on how to use this function see the package vignette: `vignette('jfa',package = 'jfa')`

Usage

```
auditPrior(confidence = 0.95, likelihood = 'binomial', method = 'none',
           expectedError = 0, N = NULL, materiality = NULL, ir = 1, cr = 1,
           pHmin = NULL, pHplus = NULL, factor = 1, sampleN = 0, sampleK = 0)
```

Arguments

`confidence`	a value between 0 and 1 specifying the confidence level desired for the sample planning. Defaults to 0.95 for 95% confidence.
`likelihood`	a character specifying the likelihood assumed when updating the prior distribution. This can be either `binomial` for the binomial likelihood and beta prior distribution, `poisson` for the Poisson likelihood and gamma prior distribution, or `hypergeometric` for the hypergeometric likelihood and beta-binomial prior distribution. See the details section for more information about the available likelihoods.
`method`	a character specifying the method by which the prior distribution is constructed. Defaults to the `none` method, which incorporates no existing information. Other options are `median`, `hypotheses`, `arm`, `sample` or `factor`. See the details section for more information about these methods.
`expectedError`	a value between 0 and 1 specifying the expected errors in the sample relative to the total sample size, or a number (>= 1) that represents the number of expected errors in the sample. It is advised to set this value conservatively to minimize the probability of the observed errors exceeding the expected errors, which would imply that insufficient work has been done in the end.
`N`	an integer (or value) specifying the total population size (or value). Only required when `likelihood = 'hypergeometric'`.
`materiality`	a value between 0 and 1 specifying the performance materiality (i.e., maximum upper limit) of the audit as a fraction of the total size (or value). Can be `NULL` for some methods.
`ir`	if `method = 'arm'`, a value between 0 and 1 specifying the inherent risk probability from the audit risk model. Defaults to 1 for 100% risk.
`cr`	if `method = 'arm'`, a value between 0 and 1 specifying the internal control risk probability from the audit risk model. Defaults to 1 for 100% risk.
`pHmin`	if `method = 'hypotheses'`, a value between 0 and 1 specifying the prior probability of the hypothesis $\theta <$ materiality.

pHplus	if method = 'hypotheses', a value between 0 and 1 specifying the prior probability of the hypothesis $\theta >$ materiality.
factor	if method = 'factor', a value between 0 and 1 specifying the weighting factor for the results of the earlier sample.
sampleN	if method = 'sample' or method = 'factor', an integer specifying the number of transactions that were inspected in the previous sample.
sampleK	if sample or factor, a value specifying the sum of errors in the previous sample.

Details

This section elaborates on the available likelihoods and corresponding prior distributions for the likelihood argument.

- poisson: The Poisson likelihood is used as a likelihood for monetary unit sampling (MUS). Its likelihood function is defined as:

  $$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

  The conjugate *gamma*(α, β) prior has probability density function:

  $$f(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$$

- binomial: The binomial likelihood is used as a likelihood for record sampling *with* replacement. Its likelihood function is defined as:

  $$p(x) = \binom{n}{k} p^k (1-p)^{n-k}$$

  The conjugate *beta*(α, β) prior has probability density function:

  $$f(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

- hypergeometric: The hypergeometric likelihood is used as a likelihood for record sampling *without* replacement. Its likelihood function is defined as:

  $$p(x = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

  The conjugate *beta-binomial*(α, β) prior (Dyer and Pierce, 1993) has probability density function:

  $$f(k|n, \alpha, \beta) = \binom{n}{k} \frac{B(k+\alpha, n-k+\beta)}{B(\alpha, \beta)}$$

This section elaborates on the available methods for constructing a prior distribution.

- none: This method constructs a prior distribution according to the principle of minimum information.
- median: This method constructs a prior distribution so that the prior probabilities of tolerable and intolerable misstatement are equal.
- hypotheses: This method constructs a prior distribution with specified prior probabilities for the hypotheses of tolerable and intolerable misstatement. Requires specification of the pHmin and pHplus arguments.

- `arm`: This method constructs a prior distribution according to the assessed risks in the audit risk model. Requires specification of the `ir` and `cr` arguments.
- `sample`: This method constructs a prior distribution on the basis of an earlier sample. Requires specification of the `sampleN` and `sampleK` arguments.
- `factor`: This method constructs a prior distribution on the basis of last year's results and a weighting factor. Requires specification of the `factor`, `sampleN` and `sampleK` arguments.

Value

An object of class `jfaPrior` containing:

`confidence`	a fraction indicating the confidence level for the desired population statement.
`likelihood`	a character indicating the specified likelihood.
`method`	a character indicating the method by which the prior distribution is constructed.
`expectedError`	a value indicating the input for the number of expected errors.
`N`	if `N` is specified, an integer indicating the population size.
`materiality`	if `materiality` is specified, a value between 0 and 1 indicating the materiality that was used to construct the prior distribution.
`description`	a list containing a description of the prior distribution, including parameters and the implicit sample.
`statistics`	a list containing statistics of the prior distribution, including the mean, mode, median, and upper bound.
`specifics`	a list containing optional specifications of the prior distribution that vary depending on the `method` used.
`hypotheses`	if `materiality` is specified, a list containing information about the hypotheses, including prior probabilities and odds.

Author(s)

Koen Derks, <k.derks@nyenrode.nl>

References

Derks, K., de Swart, J., Wagenmakers, E.-J., Wille, J., & Wetzels, R. (2019). JASP for audit: Bayesian Tools for the Auditing Practice.

Derks, K., de Swart, J., van Batenburg, P. Wagenmakers, E.-J., & Wetzels, R. (2020). Priors in a Bayesian Audit: How Integrating Information into the Prior Distribution can Improve Audit Transparency and Efficiency.

See Also

`planning` `selection` `evaluation` `report`

Examples

```
library(jfa)

# Specify the materiality, confidence, and expected errors:
materiality   <- 0.05   # 5%
confidence    <- 0.95   # 95%
expectedError <- 0.025  # 2.5%
```

```
# Specify the inherent risk (ir) and control risk (cr):
ir <- 1     # 100%
cr <- 0.6   # 60%

# Create a beta prior distribution according to the Audit Risk Model (arm)
# and a binomial likelihood:
prior <- auditPrior(confidence = confidence, likelihood = 'binomial',
                method = 'arm', expectedError = expectedError, materiality = materiality,
                    ir = ir, cr = cr)
print(prior)

# ------------------------------------------------------------
#         jfa Prior Distribution Summary (Bayesian)
# ------------------------------------------------------------
# Input:
#
# Confidence:              0.95
# Expected sample errors:  0.02
# Likelihood:              binomial
# Specifics:           Inherent risk = 1; Internal control risk = 0.6; Detection risk = 0.08
# ------------------------------------------------------------
# Output:
#
# Prior distribution:      beta(2.275, 50.725)
# Implicit sample size:    51
# Implicit errors:         1.27
# ------------------------------------------------------------
# Statistics:
#
# Upper bound:             0.1
# Precision:               7.1%
# Mode:                    0.02
# Mean:                    0.04
# Median:                  0.04
# ------------------------------------------------------------
```

`BuildIt` *BuildIt Construction financial statements*

Description

Fictional data from a construction company in the United States, containing 3500 observations identification numbers, book values, and audit values. The audit values are added for illustrative purposes, as these would need to be assessed by the auditor in the execution stage of the audit.

Usage

```
data(BuildIt)
```

Format

A data frame with 3500 rows and 3 variables.

ID unique record identification number.

bookValue book value in US dollars ($14.47–$2,224.40).

auditValue true value in US dollars ($14.47–$2,224.40).

References

Derks, K., de Swart, J., Wagenmakers, E.-J., Wille, J., & Wetzels, R. (2019). JASP for audit: Bayesian tools for the auditing practice.

Examples

```
data(BuildIt)
```

`evaluation` *Evaluation of Audit Samples using Confidence / Credible Bounds*

Description

This function takes a data frame (using `sample`, `bookValues`, and `auditValues`) or summary statistics (using `nSumstats` and `kSumstats`) and evaluates the audit sample according to the specified method. The returned object is of class `jfaEvaluation` and can be used with associated `print()` and `plot()` methods.

For more details on how to use this function see the package vignette: `vignette("jfa",package = "jfa")`

Usage

```
evaluation(confidence = 0.95, method = "binomial", N = NULL,
            sample = NULL, bookValues = NULL, auditValues = NULL, counts = NULL,
            nSumstats = NULL, kSumstats = NULL,
            materiality = NULL, minPrecision = NULL,
            prior = FALSE, nPrior = 0, kPrior = 0,
        rohrbachDelta = 2.7, momentPoptype = "accounts", populationBookValue = NULL,
            csA = 1, csB = 3, csMu = 0.5)
```

Arguments

`confidence`	the required confidence level for the bound. Default is 0.95 for 95% confidence.
`method`	the method that is used to evaluate the sample. This can be either one of `poisson`, `binomial`, `hypergeometric`, `mpus`, `stringer`, `stringer-meikle`, `stringer-lta`, `stringer-pvz`, `rohrbach`, `moment`, `direct`, `difference`, `quotient`, or `regression`.
`N`	an integer specifying the total number of units (transactions or monetary units) in the population.
`sample`	a data frame containing at least a column of Ist values and a column of Soll (true) values.
`bookValues`	a character specifying the column name for the Ist values in the sample.
`auditValues`	a character specifying the column name for the Soll values in the sample.
`counts`	a integer vector of the number of times each transaction in the sample is to be evaluated (due to it being selected multiple times for the sample).

`nSumstats` an integer specifying the number of transactions in the sample. If specified, overrides the `sample`, `bookValues` and `auditValues` arguments and assumes that the data come from summary statistics specified by both `nSumstats` and `kSumstats`.

`kSumstats` a value specifying the sum of taints (proportional errors) found in the sample. If specified, overrides the `sample`, `bookValues` and `auditValues` arguments and assumes that the data come from summary statistics specified by both `kSumstats` and `nSumstats`.

`materiality` a value specifying the performance materiality as a fraction of the total value (or size) of the population (a value between 0 and 1). If specified, the function also returns the conclusion of the analysis with respect to the performance materiality. The value is discarded when `direct`, `difference`, `quotient`, or `regression` method is chosen.

`minPrecision` a value specifying the required minimum precision. If specified, the function also returns the conclusion of the analysis with respect to the required minimum precision. This value must be specified as a fraction of the total value of the population (a value between 0 and 1).

`prior` a logical indicating whether to use a prior distribution when evaluating. Defaults to `FALSE` for frequentist evaluation. If `TRUE`, the prior distribution is updated by the corresponding likelihood. Chooses a conjugate gamma distribution for the Poisson likelihood, a conjugate beta distribution for the binomial likelihood, and a conjugate beta-binomial distribution for the hypergeometric likelihood.

`nPrior` a value for the prior parameter β (number of transactions in the assumed prior sample).

`kPrior` a value for the prior parameter α (total tainting in the assumed prior sample).

`rohrbachDelta` a value specifying Δ in Rohrbach's augmented variance bound (Rohrbach, 1993).

`momentPoptype` a character specifying the type of population for the modified moment method (Dworin and Grimlund, 1984). Can be either one of `accounts` or `inventory`. Options result in different methods for calculating the central moments.

`populationBookValue`
a value specifying the total value of the transactions in the population. Required when `method` is one of `direct`, `difference`, `quotient`, or `regression`, but optional otherwise.

`csA` if `method = "coxsnell"`, the α parameter of the prior distribution on the mean taint. Default is set to 1, as recommended by Cox and Snell (1979).

`csB` if `method = "coxsnell"`, the β parameter of the prior distribution on the mean taint. Default is set to 3, as recommended by Cox and Snell (1979).

`csMu` if `method = "coxsnell"`, the mean of the prior distribution on the mean taint. Default is set to 0.5, as recommended by Cox and Snell (1979).

Details

This section lists the available options for the `methods` argument.

- `poisson`: The confidence bound taken from the Poisson distribution. If combined with `prior = TRUE`, performs Bayesian evaluation using a *gamma* prior and posterior.
- `binomial`: The confidence bound taken from the binomial distribution. If combined with `prior = TRUE`, performs Bayesian evaluation using a *beta* prior and posterior.

- `hypergeometric`: The confidence bound taken from the hypergeometric distribution. If combined with `prior = TRUE`, performs Bayesian evaluation using a *beta-binomial* prior and posterior.
- `mpu`: Mean per unit estimator using the observed sample taints.
- `stringer`: The Stringer bound (Stringer, 1963).
- `stringer-meikle`: Stringer bound with Meikle's correction for understatements (Meikle, 1972).
- `stringer-lta`: Stringer bound with LTA correction for understatements (Leslie, Teitlebaum, and Anderson, 1979).
- `stringer-pvz`: Stringer bound with Pap and van Zuijlen's correction for understatements (Pap and van Zuijlen, 1996).
- `rohrbach`: Rohrbach's augmented variance bound (Rohrbach, 1993).
- `moment`: Modified moment bound (Dworin and Grimlund, 1984).
- `coxsnell`: Cox and Snell bound (Cox and Snell, 1979).
- `direct`: Confidence interval using the direct method (Touw and Hoogduin, 2011).
- `difference`: Confidence interval using the difference method (Touw and Hoogduin, 2011).
- `quotient`: Confidence interval using the quotient method (Touw and Hoogduin, 2011).
- `regression`: Confidence interval using the regression method (Touw and Hoogduin, 2011).

Value

An object of class `jfaEvaluation` containing:

`confidence`	a value specifying the confidence level of the result.
`method`	the evaluation method that was used.
`N`	if `N` is specified, the population size that is used.
`n`	an integer specifying the sample size used in the evaluation.
`k`	an integer specifying the number of transactions that contained an error.
`t`	a value specifying the sum of observed taints.
`materiality`	if `materiality` is specified, the performance materiality used.
`minPrecision`	if `minPrecision` is specified, the minimum required precision used.
`mle`	a value specifying the most likely error in the population as a proportion.
`precision`	a value specifying the difference between the mle and the upper confidence bound as a proportion.
`popBookvalue`	if specified as input, the total Ist value of the population.
`pointEstimate`	if method is one of `direct`, `difference`, `quotient`, or `regression`, the value of the point estimate.
`lowerBound`	if method is one of `direct`, `difference`, `quotient`, or `regression`, the value of the lower bound of the interval.
`upperBound`	if method is one of `direct`, `difference`, `quotient`, or `regression`, the value of the upper bound of the interval.
`confBound`	the upper confidence bound on the error percentage.
`conclusion`	if `materiality` is specified, the conclusion about whether to approve or not approve the population.

populationK	the assumed total errors in the population. Used in inferences with hypergeometric method.
prior	an object of class 'jfaPrior' to represents the prior distribution.
posterior	an object of class 'jfaPosterior' to represents the posterior distribution.
data	a data frame containing the relevant columns from the sample input.

Author(s)

Koen Derks, <k.derks@nyenrode.nl>

References

Cox, D. and Snell, E. (1979). On sampling and the estimation of rare errors. *Biometrika*, 66(1), 125-132.

Dworin, L. D. and Grimlund, R. A. (1984). Dollar-unit Sampling for accounts receivable and inventory. *The Accounting Review*, 59(2), 218–241

Leslie, D. A., Teitlebaum, A. D., & Anderson, R. J. (1979). *Dollar-unit sampling: a practical guide for auditors*. Copp Clark Pitman; Belmont, Calif.: distributed by Fearon-Pitman.

Meikle, G. R. (1972). *Statistical Sampling in an Audit Context: An Audit Technique*. Canadian Institute of Chartered Accountants.

Pap, G., and van Zuijlen, M. C. (1996). On the asymptotic behavior of the Stringer bound 1. *Statistica Neerlandica*, 50(3), 367-389.

Rohrbach, K. J. (1993). Variance augmentation to achieve nominal coverage probability in sampling from audit populations. *Auditing*, 12(2), 79.

Stringer, K. W. (1963). Practical aspects of statistical sampling in auditing. *In Proceedings of the Business and Economic Statistics Section* (pp. 405-411). American Statistical Association.

Touw, P., and Hoogduin, L. (2011). *Statistiek voor Audit en Controlling*. Boom uitgevers Amsterdam.

See Also

auditPrior planning selection

Examples

```
library(jfa)
set.seed(1)

# Generate some audit data (N = 1000):
data <- data.frame(ID = sample(1000:100000, size = 1000, replace = FALSE),
                   bookValue = runif(n = 1000, min = 700, max = 1000))

# Using monetary unit sampling, draw a random sample from the population.
s1 <- selection(population = data, sampleSize = 100, units = "mus",
                 bookValues = "bookValue", algorithm = "random")
s1_sample <- s1$sample
s1_sample$trueValue <- s1_sample$bookValue
s1_sample$trueValue[2] <- s1_sample$trueValue[2] - 500 # One overstatement is found

# Using summary statistics, calculate the upper confidence bound according
# to the binomial distribution:
```

```
e1 <- evaluation(nSumstats = 100, kSumstats = 1, method = "binomial",
                 materiality = 0.05)
print(e1)

# ------------------------------------------------------------
#             jfa Evaluation Summary (Frequentist)
# ------------------------------------------------------------
# Input:
#
# Confidence:              95%
# Materiality:             5%
# Minium precision:        Not specified
# Sample size:             100
# Sample errors:           1
# Sum of taints:           1
# Method:                  binomial
# ------------------------------------------------------------
# Output:
#
# Most likely error:       1%
# Upper bound:             4.66%
# Precision:               3.66%
# Conclusion:              Approve population
# ------------------------------------------------------------

# Evaluate the raw sample using the stringer bound and the sample counts:

e2 <- evaluation(sample = s1_sample, bookValues = "bookValue", auditValues = "trueValue",
                 method = "stringer", materiality = 0.05, counts = s1_sample$counts)
print(e2)

# ------------------------------------------------------------
#             jfa Evaluation Summary (Frequentist)
# ------------------------------------------------------------
# Input:
#
# Confidence:              95%
# Materiality:             5%
# Minium precision:        Not specified
# Sample size:             100
# Sample errors:           1
# Sum of taints:           1
# Method:                  stringer
# ------------------------------------------------------------
# Output:
#
# Most likely error:       0.69%
# Upper bound:             4.12%
# Precision:               3.44%
# Conclusion:              Approve population
# ------------------------------------------------------------
```

---

planning *Frequentist and Bayesian Planning for Audit Sampling*

Description

This function calculates the required sample size for an audit, based on the Poisson, binomial or hypergeometric likelihood. A prior distribution can be specified to perform Bayesian planning. The returned object is of class `jfaPlanning` and has a `print()` and `plot()` method.

For more details on how to use this function see the package vignette: `vignette('jfa',package = 'jfa')`

Usage

```
planning(confidence = 0.95, expectedError = 0, likelihood = 'poisson', N = NULL,
         materiality = NULL, minPrecision = NULL,
         prior = FALSE, kPrior = 0, nPrior = 0,
         increase = 1, maxSize = 5000)
```

Arguments

`confidence`	a value between 0 and 1 specifying the confidence level desired for the sample planning. Defaults to 0.95 for 95% confidence.
`expectedError`	a value between 0 and 1 specifying the expected errors in the sample relative to the total sample size, or a number (>= 1) that represents the number of expected errors in the sample. It is advised to set this value conservatively to minimize the probability of the observed errors exceeding the expected errors, which would imply that insufficient work has been done in the end.
`likelihood`	a character specifying the likelihood assumed in the calculation. This can be either `binomial` for the binomial likelihood, `poisson` for the Poisson likelihood, or `hypergeometric` for the hypergeometric likelihood. See the details section for more information about the available likelihoods.
`N`	an integer (or value) specifying the total population size (or value). Only required when `likelihood = 'hypergeometric'`.
`materiality`	a value between 0 and 1 specifying the performance materiality (i.e., maximum upper limit) of the audit as a fraction of the total size (or value). Can be `NULL`, but `minPrecision` should be specified in that case.
`minPrecision`	a value between 0 and 1 specifying the minimum precision (i.e., upper bound minus most likely error) to be obtained. Can be `NULL`, but `materiality` should be specified in that case.
`prior`	a logical specifying whether to use a prior distribution when planning, or an object of class 'jfaPrior' containing the prior distribution. Defaults to `FALSE` for frequentist planning. If `TRUE`, a negligible prior distribution is chosen by default, but can be adjusted using the 'kPrior' and 'nPrior' arguments. Chooses a conjugate gamma distribution for the Poisson likelihood, a conjugate beta distribution for the binomial likelihood, and a conjugate beta-binomial distribution for the hypergeometric likelihood.
`kPrior`	if `prior = TRUE`, a value specifying the assumed sum of errors in the implicit sample on which the prior distribution is based.
`nPrior`	if `prior = TRUE`, a value specifying the number of observations in the implicit sample on which the prior distribution is based.
`increase`	an integer specifying the desired increase step for the sample size calculation.
`maxSize`	an integer specifying the maximum sample size that is considered in the calculation. Defaults to 5000 for efficiency. Increase this value if the sample size cannot be found due to it being too large (e.g., for a low materiality).

Details

This section elaborates on the available likelihoods and corresponding prior distributions for the `likelihood` argument.

- `poisson`: The Poisson likelihood is used as a likelihood for monetary unit sampling (MUS). Its likelihood function is defined as:

  $$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

  The conjugate *gamma*(α, β) prior has probability density function:

  $$f(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$$

- `binomial`: The binomial likelihood is used as a likelihood for record sampling *with* replacement. Its likelihood function is defined as:

  $$p(x) = \binom{n}{k} p^k (1-p)^{n-k}$$

  The conjugate *beta*(α, β) prior has probability density function:

  $$f(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

- `hypergeometric`: The hypergeometric likelihood is used as a likelihood for record sampling *without* replacement. Its likelihood function is defined as:

  $$p(x = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

  The conjugate *beta-binomial*(α, β) prior (Dyer and Pierce, 1993) has probability density function:

  $$f(k|n, \alpha, \beta) = \binom{n}{k} \frac{B(k+\alpha, n-k+\beta)}{B(\alpha, \beta)}$$

Value

An object of class `jfaPlanning` containing:

`confidence`	a fraction indicating the confidence level for the desired population statement.
`expectedError`	the specified number of errors as a fraction or as a number.
`likelihood`	a character indicating the specified likelihood.
`N`	an integer indicating the population size (only returned if it is specified).
`materiality`	a fraction indicating the specified materiality. Can be `NULL`.
`minPrecision`	a fraction indicating the minimum precision to be obtained. Can be `NULL`.
`sampleSize`	an integer indicating the required sample size.
`errorType`	a character indicating whether the expected errors where specified as a percentage or as an integer.
`expectedSampleError`	a value indicating the number of errors that are allowed in the sample.

`expectedBound` a value indicating the expected upper bound if the sample goes according to plan.

`expectedPrecision` a value indicating the expected precision if the sample goes according to plan.

`populationK` if `likelihood = 'hypergeometric'`, an integer indicating the assumed population errors.

`prior` if a prior distribution is specified, an object of class `jfaPrior` that contains the prior distribution.

`expectedPosterior` if a prior distribution is specified, an object of class `jfaPosterior` that contains the expected posterior distribution.

Author(s)

Koen Derks, <k.derks@nyenrode.nl>

References

Dyer, D. and Pierce, R.L. (1993). On the Choice of the Prior Distribution in Hypergeometric Sampling. *Communications in Statistics - Theory and Methods*, 22(8), 2125 - 2146.

See Also

`auditPrior` `selection` `evaluation` `report`

Examples

```
library(jfa)

# Using the binomial distribution, calculates the required sample size for a
# materiality of 5% when 2.5% mistakes are expected to be found in the sample.

# Frequentist planning with binomial likelihood:

p1 <- planning(confidence = 0.95, expectedError = 0.025, likelihood = 'binomial',
               materiality = 0.05)
print(p1)

# ------------------------------------------------------------
#              jfa Planning Summary (Frequentist)
# ------------------------------------------------------------
# Input:
#
# Confidence:              95%
# Materiality:             5%
# Minimum precision:       Not specified
# Likelihood:              binomial
# Expected sample errors:  6
# ------------------------------------------------------------
# Output:
#
# Sample size:             234
# ------------------------------------------------------------
# Statistics:
#
```

```
# Expected upper bound:    5%
# Expected precision:      2.43%
# ------------------------------------------------------------

# Bayesian planning with prior:

prior <- auditPrior(confidence = 0.95, likelihood = 'binomial', method = 'arm',
                    expectedError = 0.025, materiality = 0.05, cr = 0.6)

p2 <- planning(confidence = 0.95, expectedError = 0.025, materiality = 0.05,
               prior = prior)
print(p2)

# ------------------------------------------------------------
#              jfa Planning Summary (Bayesian)
# ------------------------------------------------------------
# Input:
#
# Confidence:              95%
# Materiality:             5%
# Minimum precision:       Not specified
# Likelihood:              binomial
# Prior distribution:      beta(2.275, 50.725)
# Expected sample errors:  4.23
# ------------------------------------------------------------
# Output:
#
# Sample size:             169
# Posterior distribution:  beta(6.5, 215.5)
# ------------------------------------------------------------
# Statistics:
#
# Expected upper bound:    4.99%
# Expected precision:      2.49%
# Expected Bayes factor-+: 9.32
# ------------------------------------------------------------
```

`report` *Generate an Audit Report*

Description

This function takes an object of class `jfaEvaluation`, creates a report containing the results, and saves the report to a file in your working directory.

For more details on how to use this function see the package vignette: `vignette("jfa",package = "jfa")`

Usage

```
report(object = NULL, file = NULL, format = "html_document")
```

Arguments

- `object` an object of class `jfaEvaluation` as returned by the `evaluation()` function.
- `file` a string that gives the desired name of the file (e.g. `"report.html"`). The report is created in your current working directory.
- `format` can be either one of `"html_document"` or `"pdf_document"` (compiling to pdf requires MikTex).

Value

A html or pdf report containing the results of the evaluation.

Author(s)

Koen Derks, <k.derks@nyenrode.nl>

See Also

evaluation

Examples

```
library(jfa)
set.seed(1)

# Generate some audit data (N = 1000):
data <- data.frame(ID = sample(1000:100000, size = 1000, replace = FALSE),
                   bookValue = runif(n = 1000, min = 700, max = 1000))

# Using monetary unit sampling, draw a random sample from the population.
s1 <- selection(population = data, sampleSize = 100, units = "mus",
                 bookValues = "bookValue", algorithm = "random")
s1_sample <- s1$sample
s1_sample$trueValue <- s1_sample$bookValue
s1_sample$trueValue[2] <- s1_sample$trueValue[2] - 500 # One overstatement is found

e2 <- evaluation(sample = s1_sample, bookValues = "bookValue", auditValues = "trueValue",
                 method = "stringer", materiality = 0.05, counts = s1_sample$counts)

# Generate report
# report(e2, file = "myFile.html")
```

---

selection — *Selecting a Sample from an Audit Population*

Description

This function takes a data frame and performs sampling according to one of three popular algorithms: random sampling, cell sampling, or fixed interval sampling. Sampling is done in combination with one of two sampling units: records or monetary units The returned object is of class `jfaSelection` and can be used with associated `print()` and `plot()` methods.

For more details on how to use this function see the package vignette: `vignette("jfa",package = "jfa")`

Usage

```
selection(population, sampleSize, units = "records", algorithm = "random",
          bookValues = NULL, intervalStartingPoint = 1, ordered = TRUE,
          ascending = TRUE, withReplacement = FALSE, seed = 1)
```

Arguments

population	a data frame containing the population the auditor wishes to sample from.
sampleSize	the number of observations that need to be selected from the population. Can also be an object of class `jfaPlanning`.
units	can be either `records` (default) for record sampling, or `mus` for monetary unit sampling.
algorithm	can be either one of `random` (default) for random sampling, `cell` for cell sampling, or `interval` for fixed interval sampling.
bookValues	a character specifying the name of the column containing the book values (as in the population data).
intervalStartingPoint	the starting point in the interval (used only in fixed interval sampling)
ordered	if `TRUE` (default), the population is first ordered according to the value of their book values.
ascending	if `TRUE` (default), order the population in ascending order.
withReplacement	whether sampling should be performed with replacement. Defaults to `FALSE`.
seed	seed to reproduce results. Default is 1.

Details

This first part of this section elaborates on the possible options for the `units` argument:

- `records`: In record sampling, each observation in the population is seen as a sampling unit. An observation of $5000 is therefore equally likely to be selected as an observation of $500.
- `mus`: In monetary unit sampling, each monetary unit in the population is seen as a sampling unit. An observation of $5000 is therefore ten times more likely to be selected as an observation of $500.

This second part of this section elaborates on the possible options for the `algorithm` argument:

- `random`: In random sampling each sampling unit in the population is drawn with equal probability.
- `cell`: In cell sampling the sampling units in the population are divided into a number (equal to the sample size) of intervals. From each interval one sampling unit is selected with equal probability.
- `interval`: In fixed interval sampling the sampling units in the population are divided into a number (equal to the sample size) of intervals. From each interval one sampling unit is selected according to a fixed starting point (`intervalStartingPoint`).

Value

An object of class `jfaSelection` containing:

`population`	a data frame containing the input population.
`sample`	a data frame containing the selected observations.
`units`	the sampling units that were used for sampling.
`algorithm`	the algorithm that was used for sampling.
`bookValues`	if specified, the name of the specified book value column.

Author(s)

Koen Derks, <k.derks@nyenrode.nl>

References

Wampler, B., & McEacharn, M. (2005). Monetary-unit sampling using Microsoft Excel. *The CPA journal*, 75(5), 36.

See Also

auditPrior planning evaluation

Examples

```
library(jfa)
set.seed(1)

# Generate some audit data (N = 1000).
population <- data.frame(ID = sample(1000:100000, size = 1000, replace = FALSE),
                         bookValue = runif(n = 1000, min = 700, max = 1000))

# Draw a custom sample of 100 from the population (via random record sampling):

s1 <- selection(population = population, sampleSize = 100, algorithm = "random",
                units = "records", seed = 1)
print(s1)

# ------------------------------------------------------------
#                  jfa Selection Summary
# ------------------------------------------------------------
# Input:
#
# Population size:          1000
# Requested sample size:    100
# Sampling units:           Records
# Algorithm:                Random sampling
# ------------------------------------------------------------
# Output:
#
# Obtained sample size:     100
# ------------------------------------------------------------
# Statistics:
#
# Proportion n/N:           0.1
```

```
# ------------------------------------------------------------

# Use the result from the planning stage in the sampling stage:

p1 <- planning(materiality = 0.05, confidence = 0.95, expectedError = 0.025,
               likelihood = "binomial")

# Draw a sample via random monetary unit sampling:
s2 <- selection(population = population, sampleSize = p1, algorithm = "random",
                units = "mus", seed = 1, bookValues = "bookValue")
print(s2)

# ------------------------------------------------------------
#                  jfa Selection Summary
# ------------------------------------------------------------
# Input:
#
# Population size:         1000
# Requested sample size:   234
# Sampling units:          Monetary units
# Algorithm:               Random sampling
# ------------------------------------------------------------
# Output:
#
# Obtained sample size:    234
# ------------------------------------------------------------
# Statistics:
#
# Proportion n/N:          0.23
# Percentage of value:     23.06%
# ------------------------------------------------------------
```

Index