Package: jfa (via r-universe)

October 29, 2026

Title Statistical Methods for Auditing

Version 0.7.4

Date 2025-11-14

Description Provides statistical methods for auditing as implemented in JASP for Audit (Derks et al., 2021 <doi:10.21105/joss.02733>). First, the package makes it easy for an auditor to plan a statistical sample, select the sample from the population, and evaluate the misstatement in the sample compliant with international auditing standards. Second, the package provides statistical methods for auditing data, including tests of digit distributions and repeated values. Finally, the package includes methods for auditing algorithms on the aspect of fairness and bias. Next to classical statistical methodology, the package implements Bayesian equivalents of these methods whose statistical underpinnings are described in Derks et al. (2021) <doi:10.1111/ijau.12240>, Derks et al. (2024) <doi:10.2308/AJPT-2021-086>, Derks et al. (2022) <doi:10.31234/osf.io/8nf3e> Derks et al. (2024) <doi:10.31234/osf.io/tgq5z>, and Derks et al. (2025) <doi:10.31234/osf.io/b8tu2>.

Depends R (>= 3.5.0)

Imports bde (>= 1.0.1.1), extraDistr (>= 1.9.1), ggplot2 (>= 3.4.2), methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), rstantools (>= 2.2.0), stats, truncdist (>= 1.0-2)

Suggests benford.analysis (>= 0.1.5), BenfordTests (>= 1.2.0), BeyondBenford (>= 1.4), fairness, knitr, MUS (>= 0.1.6), rmarkdown, samplingbook (>= 1.2.4), testthat

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

VignetteBuilder knitr

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

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

License GPL (>= 3)

Encoding UTF-8

LazyData true

NeedsCompilation yes

SystemRequirements C++17, GNU make

Language en-US

RoxygenNote 7.3.3

Biarch true

Author Koen Derks [aut, cre] (ORCID: <https://orcid.org/0000-0002-5533-9349>), Lotte Mensink [ctb] (ORCID: <https://orcid.org/0009-0001-8321-4690>), Federica Picogna [ctb] (ORCID: <https://orcid.org/0009-0005-5587-6119>)

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

Config/pak/sysreqs make

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

Date/Publication 2025-11-15 12:56:45 UTC

RemoteUrl https://github.com/cran/jfa

RemoteRef HEAD

RemoteSha 2cae208c8e73cdd948517e70d1cc4bcf29fd450f

Contents

jfa-package . . . 3
accounts . . . 4
allowances . . . 5
auditPrior . . . 6
benchmark . . . 10
BuildIt . . . 11
carrier . . . 12
compas . . . 13
digit_test . . . 13
evaluation . . . 16
fairness_selection . . . 21
jfa-methods . . . 24
model_fairness . . . 27
planning . . . 31
repeated_test . . . 34
retailer . . . 36
sanitizer . . . 37
selection . . . 37
sinoForest . . . 40

Index **42**

`jfa-package` *jfa — Statistical Methods for Auditing*

Description

`jfa` is an R package that provides statistical methods for auditing. The package includes functions for planning, performing, and evaluating an audit sample compliant with international auditing standards, as well as functions for auditing data, such as testing the distribution of leading digits in the data against Benford's law. In addition to offering classical frequentist methods, `jfa` also provides a straightforward implementation of their Bayesian counterparts.

The functionality of the `jfa` package and its intended workflow are implemented with a graphical user interface in the Audit module of JASP, a free and open-source software program for statistical analyses.

For documentation on `jfa` itself, including the manual and user guide for the package, worked examples, and other tutorial information visit the package website.

Author(s)

Koen Derks (maintainer, author) <k.derks@nyenrode.nl>

Please use the citation provided by R when citing this package. A BibTex entry is available from `citation('jfa')`.

See Also

Useful links:

- The vignettes for worked examples.
- The issue page to submit a bug report or feature request.

Examples

```
# Load the jfa package
library(jfa)

################################
### Example 1: Audit sampling ###
################################

# Load the BuildIt population
data('BuildIt')

# Stage 1: Planning
stage1 <- planning(materiality = 0.03, expected = 0.01)
summary(stage1)
```

```
# Stage 2: Selection
stage2 <- selection(data = BuildIt, size = stage1,
                    units = 'values', values = 'bookValue',
                    method = 'interval', start = 1)
summary(stage2)

# Stage 3: Execution
sample <- stage2[['sample']]

# Stage 4: Evaluation
stage4 <- evaluation(data = sample, method = 'stringer.binomial',
                     values = 'bookValue', values.audit = 'auditValue')
summary(stage4)

################################
### Example 2: Data auditing ####
################################

# Load the sinoForest data set
data('sinoForest')

# Test first digits in the data against Benford's law
digit_test(sinoForest[["value"]], check = "first", reference = "benford")

#####################################
### Example 3: Algorithm auditing ####
#####################################

# Load the compas data set
data('compas')

# Test algorithmic fairness against Caucasian ethnicity
model_fairness(compas, "Ethnicity", "TwoYrRecidivism", "Predicted",
               privileged = "Caucasian", positive = "yes")
```

`accounts` *Accounts Receivable*

Description

Audit sample obtained from a population of N = 87 accounts receivable, totaling $612,824 in book value (Higgins and Nandram, 2009; Lohr, 2021).

Usage

```
data(accounts)
```

Format

A data frame with 20 rows and 3 variables.

account account number (between 1 - 87)

bookValue booked value of the account

auditValue audited (i.e., true) value of the account

References

Higgins, H. N., & Nandram, B. (2009). Monetary unit sampling: Improving estimation of the total audit error *Advances in Accounting*, 25(2), 174-182. doi:10.1016/j.adiac.2009.06.001

Lohr, S. L. (2021). *Sampling: Design and Analysis*. CRC press.

Examples

```
data(accounts)
```

`allowances` *Legitimacy Audit*

Description

Fictional population from a legitimacy audit, containing 4189 records with identification numbers, stratum identifiers, 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(allowances)
```

Format

A data frame with 4189 rows and 5 variables.

item a unique record identification number.

branch the stratum identifier / branch number.

bookValue the item book value in US dollars.

auditValue the item audit (i.e., true) value in US dollars.

times a sample selection indicator (`0` = not in sample).

Examples

```
data(allowances)
```

auditPrior *Audit Sampling: Prior Distributions*

Description

auditPrior() is used to create a prior distribution for Bayesian audit sampling. The interface allows a complete customization of the prior distribution as well as a formal translation of pre-existing audit information into a prior distribution. The function returns an object of class jfaPrior that can be used in the planning() and evaluation() functions via their prior argument. Objects with class jfaPrior can be further inspected via associated summary() and plot() methods.

Usage

```
auditPrior(
  method = c(
    "default", "param", "strict", "impartial", "hyp",
    "arm", "bram", "sample", "power", "nonparam"
  ),
  likelihood = c(
    "poisson", "binomial", "hypergeometric",
    "normal", "uniform", "cauchy", "t", "chisq",
    "exponential"
  ),
  N.units = NULL,
  alpha = NULL,
  beta = NULL,
  materiality = NULL,
  expected = 0,
  ir = NULL,
  cr = NULL,
  ub = NULL,
  p.hmin = NULL,
  x = NULL,
  n = NULL,
  delta = NULL,
  samples = NULL,
  conf.level = 0.95
)
```

Arguments

method	a character specifying the method by which the prior distribution is constructed. Possible options are default, strict, impartial, param, arm, bram, hyp, sample, and power. See the details section for more information.
likelihood	a character specifying the likelihood for updating the prior distribution. Possible options are poisson (default) for a conjugate gamma prior distribution,

	binomial for a conjugate beta prior distribution, or hypergeometric for a conjugate beta-binomial prior distribution. See the details section for more information.
N.units	a numeric value larger than 0 specifying the total number of units in the population. Required for the hypergeometric likelihood.
alpha	a numeric value specifying the α parameter of the prior distribution. Required for method param.
beta	a numeric value specifying the β parameter of the prior distribution. Required for method param.
materiality	a numeric value between 0 and 1 specifying the performance materiality (i.e., the maximum tolerable misstatement in the population) as a fraction. Required for methods impartial, arm, and hyp.
expected	a numeric value between 0 and 1 specifying the expected (tolerable) misstatements in the sample relative to the total sample size. Required for methods impartial, arm, bram, and hyp.
ir	a numeric value between 0 and 1 specifying the inherent risk (i.e., the probability of material misstatement occurring due to inherent factors) in the audit risk model. Required for method arm.
cr	a numeric value between 0 and 1 specifying the internal control risk (i.e., the probability of material misstatement occurring due to internal control systems) in the audit risk model. Required for method arm.
ub	a numeric value between 0 and 1 specifying the conf.level-% upper bound for the prior distribution as a fraction. Required for method bram.
p.hmin	a numeric value between 0 and 1 specifying the prior probability of the hypothesis of tolerable misstatement (H1: $\theta <$ materiality). Required for method hyp.
x	a numeric value larger than, or equal to, 0 specifying the sum of proportional misstatements (taints) in a prior sample. Required for methods sample and power.
n	a numeric value larger than 0 specifying the number of units in a prior sample. Required for methods sample and power.
delta	a numeric value between 0 and 1 specifying the weight of a prior sample specified via x and n. Required for method power.
samples	a numeric vector containing samples of the prior distribution. Required for method nonparam.
conf.level	a numeric value between 0 and 1 specifying the confidence level (1 - audit risk).

Details

To perform Bayesian audit sampling you must assign a prior distribution to the parameter in the model, i.e., the population misstatement θ. The prior distribution can incorporate pre-existing audit information about θ into the analysis, which consequently allows for a more efficient or more accurate estimates. The default priors used by jfa are indifferent towards the possible values of θ, while still being proper. Note that the default prior distributions are a conservative choice of prior since they, in most cases, assume all possible misstatement to be equally likely before seeing a data

sample. It is recommended to construct an informed prior distribution based on pre-existing audit information when possible.

This section elaborates on the available input options for the `method` argument.

- `default`: This method produces a *gamma(1, 1)*, *beta(1, 1)*, *beta-binomial(N, 1, 1)*, *normal(0.5, 1000)* , *cauchy(0, 1000)*, *student-t(1)*, or *chi-squared(1)* prior distribution. These prior distributions are mostly indifferent about the possible values of the misstatement.
- `param`: This method produces a custom `gamma(alpha, beta)`, `beta(alpha, beta)`, `beta-binomial(N, alpha, beta)` prior distribution, *normal(alpha, beta)*, *cauchy(alpha, beta)*, *student-t(alpha)*, or *chi-squared(alpha)*. The alpha and beta parameters must be set using `alpha` and `beta`.
- `strict`: This method produces an improper *gamma(1, 0)*, *beta(1, 0)*, or *beta-binomial(N, 1, 0)* prior distribution. These prior distributions match sample sizes and upper limits from classical methods and can be used to emulate classical results.
- `impartial`: This method produces an impartial prior distribution. These prior distributions assume that tolerable misstatement ($\theta <$ materiality) and intolerable misstatement ($\theta >$ materiality) are equally likely.
- `hyp`: This method translates an assessment of the prior probability for tolerable misstatement ($\theta <$ materiality) to a prior distribution.
- `arm`: This method translates an assessment of inherent risk and internal control risk to a prior distribution.
- `bram`: This method translates an assessment of the expected most likely error and *x*-% upper bound to a prior distribution.
- `sample`: This method translates the outcome of an earlier sample to a prior distribution.
- `power`: This method translates and weighs the outcome of an earlier sample to a prior distribution (i.e., a power prior).
- `nonparam`: This method takes a vector of samples from the prior distribution (via `samples`) and constructs a bounded density (between 0 and 1) on the basis of these samples to act as the prior.

This section elaborates on the available input options for the `likelihood` argument and the corresponding conjugate prior distributions used by `jfa`.

- `poisson`: The Poisson distribution is an approximation of the binomial distribution. The Poisson distribution is defined as:

$$f(\theta, n) = \frac{\lambda^{\theta} e^{-\lambda}}{\theta!}$$

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

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

  .
- `binomial`: The binomial distribution is an approximation of the hypergeometric distribution. The binomial distribution is defined as:

$$f(\theta, n, x) = \binom{n}{x} \theta^{x} (1-\theta)^{n-x}$$

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

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

.

- `hypergeometric`: The hypergeometric distribution is defined as:

$$f(x, n, K, N) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}$$

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

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

.

Value

An object of class `jfaPrior` containing:

`prior`	a string describing the functional form of the prior distribution.
`description`	a list containing a description of the prior distribution, including the parameters of the prior distribution and the implicit sample on which the prior distribution is based.
`statistics`	a list containing statistics of the prior distribution, including the mean, mode, median, and upper bound of the prior distribution.
`specifics`	a list containing specifics of the prior distribution that vary depending on the `method`.
`hypotheses`	if `materiality` is specified, a list containing information about the hypotheses, including prior probabilities and odds for the hypothesis of tolerable misstatement (H1) and the hypothesis of intolerable misstatement (H0).
`method`	a character indicating the method by which the prior distribution is constructed.
`likelihood`	a character indicating the likelihood of the data.
`materiality`	if `materiality` is specified, a numeric value between 0 and 1 giving the materiality used to construct the prior distribution.
`expected`	a numeric value larger than, or equal to, 0 giving the input for the number of expected misstatements.
`conf.level`	a numeric value between 0 and 1 giving the confidence level.
`N.units`	if `N.units` is specified, the number of units in the population.

Author(s)

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

References

Derks, K., de Swart, J., van Batenburg, P., Wagenmakers, E.-J., & Wetzels, R. (2021). Priors in a Bayesian audit: How integration of existing information into the prior distribution can improve audit transparency and efficiency. *International Journal of Auditing*, 25(3), 621-636. doi:10.1111/ijau.12240

Derks, K., de Swart, J., Wagenmakers, E.-J., Wille, J., & Wetzels, R. (2021). JASP for audit: Bayesian tools for the auditing practice. *Journal of Open Source Software*, *6*(68), 2733. doi:10.21105/joss.02733

Derks, K., de Swart, J., Wagenmakers, E.-J., & Wetzels, R. (2022). An impartial Bayesian hypothesis test for audit sampling. *PsyArXiv*. doi:10.31234/osf.io/8nf3e

See Also

`planning` `selection` `evaluation`

Examples

```
# Default beta prior
auditPrior(likelihood = "binomial")

# Impartial prior
auditPrior(method = "impartial", materiality = 0.05)

# Non-conjugate prior
auditPrior(method = "param", likelihood = "normal", alpha = 0, beta = 0.1)
```

`benchmark` *Benchmark Analysis of Sales Versus Cost of Sales*

Description

Fictional data from a benchmark analysis comparing industry sales versus the industry cost of sales.

Usage

```
data(benchmark)
```

Format

A data frame with 100 rows and 2 variables.

sales book value in US dollars ($100,187,432–$398,280,933).

costofsales true value in US dollars ($71,193,639–$309,475,784).

References

Derks, K., de Swart, J., van Batenburg, P., Wagenmakers, E.-J., & Wetzels, R. (2021). Priors in a Bayesian audit: How integration of existing information into the prior distribution can improve audit transparency and efficiency. *International Journal of Auditing*, 25(3), 621-636. doi:10.1111/ijau.12240

Examples

```
data(benchmark)
```

`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. (2021). JASP for audit: Bayesian tools for the auditing practice. *Journal of Open Source Software*, *6*(68), 2733.

Examples

```
data(BuildIt)
```

carrier *Carrier Company Financial Statements*

Description

Fictional data from a carrier company in Europe, containing 202 ledger items across 10 company entities.

Usage

```
data(carrier)
```

Format

A data frame with 202 rows and 12 variables.

description description of the ledger item.

entity1 recorded values for entity 1, in US dollars.

entity2 recorded values for entity 2, in US dollars.

entity3 recorded values for entity 3, in US dollars.

entity4 recorded values for entity 4, in US dollars.

entity5 recorded values for entity 5, in US dollars.

entity6 recorded values for entity 6, in US dollars.

entity7 recorded values for entity 7, in US dollars.

entity8 recorded values for entity 8, in US dollars.

entity9 recorded values for entity 9, in US dollars.

entity10 recorded values for entity 10, in US dollars.

total total value, in US dollars.

Source

https://medium.com/towards-data-science/data-driven-audit-1-automated-sampling-using-python-52e833

Examples

```
data(carrier)
```

compas *COMPAS Recidivism Prediction*

Description

This data was used to predict recidivism (whether a criminal will reoffend or not) in the USA.

Usage

```
data(compas)
```

Format

A data frame with 100 rows and 2 variables.

TwoYrRecidivism yes/no for recidivism or no recidivism.

AgeAboveFourtyFive yes/no for age above 45 years or not

AgeBelowTwentyFive yes/no for age below 25 years or not

Gender female/male for gender

Misdemeanor yes/no for having recorded misdemeanor(s) or not

Ethnicity Caucasian, African American, Asian, Hispanic, Native American or Other

Predicted yes/no, predicted values for recidivism

References

https://www.kaggle.com/danofer/compass https://cran.r-project.org/package=fairness

Examples

```
data(compas)
```

digit_test *Data Auditing: Digit Distribution Test*

Description

This function extracts and performs a test of the distribution of (leading) digits in a vector against a reference distribution. By default, the distribution of leading digits is checked against Benford's law.

Usage

```
digit_test(
  x,
  check = c("first", "last", "firsttwo", "lasttwo"),
  reference = "benford",
  conf.level = 0.95,
  prior = FALSE
)
```

Arguments

x	a numeric vector.
check	location of the digits to analyze. Can be first, last, firsttwo, or lasttwo.
reference	which character string given the reference distribution for the digits, or a vector of probabilities for each digit. Can be benford for Benford's law, uniform for the uniform distribution. An error is given if any entry of reference is negative. Probabilities that do not sum to one are normalized.
conf.level	a numeric value between 0 and 1 specifying the confidence level (i.e., 1 - audit risk / detection risk).
prior	a logical specifying whether to use a prior distribution, or a numeric value equal to or larger than 1 specifying the prior concentration parameter, or a numeric vector containing the prior parameters for the Dirichlet distribution on the digit categories.

Details

Benford's law is defined as $p(d) = log10(1/d)$. The uniform distribution is defined as $p(d) = 1/d$.

Value

An object of class jfaDistr containing:

data	the specified data.
conf.level	a numeric value between 0 and 1 giving the confidence level.
observed	the observed counts.
expected	the expected counts under the null hypothesis.
n	the number of observations in x.
statistic	the value the chi-squared test statistic.
parameter	the degrees of freedom of the approximate chi-squared distribution of the test statistic.
p.value	the p-value for the test.
check	checked digits.
digits	vector of digits.
reference	reference distribution

match	a list containing the row numbers corresponding to the observations matching each digit.
deviation	a vector indicating which digits deviate from their expected relative frequency under the reference distribution.
prior	a logical indicating whether a prior distribution was used.
data.name	a character string giving the name(s) of the data.

Author(s)

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

References

Benford, F. (1938). The law of anomalous numbers. *In Proceedings of the American Philosophical Society*, 551-572.

Preece, D. A. (1981). Distributions of final digits in data. *Journal of the Royal Statistical Society Series D: The Statistician*, 30(1), 31-60. doi:10.2307/2987702

Dlugosz, S., & Müller-Funk, U. (2009). The value of the last digit: Statistical fraud detection with digit analysis. *Advances in data analysis and classification*, 3(3), 281-290. doi:10.1007/s11634-00900485

See Also

repeated_test

Examples

```
set.seed(1)
x <- rnorm(100)

# First digit analysis against Benford's law
digit_test(x, check = "first", reference = "benford")

# Bayesian first digit analysis against Benford's law
digit_test(x, check = "first", reference = "benford", prior = TRUE)

# Last digit analysis against the uniform distribution
digit_test(x, check = "last", reference = "uniform")

# Bayesian last digit analysis against the uniform distribution
digit_test(x, check = "last", reference = "uniform", prior = TRUE)

# First digit analysis against a custom distribution
digit_test(x, check = "last", reference = 1:9)

# Bayesian first digit analysis against a custom distribution
digit_test(x, check = "last", reference = 1:9, prior = TRUE)
```

`evaluation` *Audit Sampling: Evaluation*

Description

`evaluation()` is used to perform statistical inference about the misstatement in a population after auditing a statistical sample. It allows specification of statistical requirements for the sample with respect to the performance materiality or the precision. The function returns an object of class `jfaEvaluation` that can be used with associated `summary()` and `plot()` methods.

Usage

```
evaluation(
  materiality = NULL,
  method = c(
    "poisson", "binomial", "hypergeometric",
    "inflated.poisson", "hurdle.beta",
    "stringer.poisson", "stringer.binomial", "stringer.hypergeometric",
    "stringer.meikle", "stringer.lta", "stringer.pvz", "stringer",
    "rohrbach", "moment", "coxsnell", "mpu", "pps",
    "direct", "difference", "quotient", "regression"
  ),
  alternative = c("less", "two.sided", "greater"),
  conf.level = 0.95,
  data = NULL,
  values = NULL,
  values.audit = NULL,
  strata = NULL,
  times = NULL,
  x = NULL,
  n = NULL,
  N.units = NULL,
  N.items = NULL,
  pooling = c("none", "complete", "partial"),
  prior = FALSE
)
```

Arguments

`materiality`	a numeric value between 0 and 1 specifying the performance materiality (i.e., the maximum tolerable misstatement in the population) as a fraction of the total number of units in the population. Can be `NULL`. Not used for methods `direct`, `difference`, `quotient`, and `regression`.
`method`	a character specifying the statistical method. Possible options are `poisson` (default), `binomial`, `hypergeometric`, `stringer.poisson`, `stringer.binomial`, `stringer.hypergeometric`, `stringer.meikle`, `stringer.lta`, `stringer.pvz`,

rohrbach, moment, mpu, pps, direct, difference, quotient, or regression. See the details section for more information.

alternative — a character indicating the alternative hypothesis and the type of confidence / credible interval returned by the function. Possible options are less (default), two.sided, or greater.

conf.level — a numeric value between 0 and 1 specifying the confidence level (i.e., 1 - audit risk / detection risk).

data — a data frame containing a data sample.

values — a character specifying name of a numeric column in data containing the book values of the items.

values.audit — a character specifying name of a numeric column in data containing the audit (i.e., true) values of the items.

strata — a character specifying name of a factor column in data indicating to which stratum the item belongs.

times — a character specifying name of an integer column in data containing the number of times an item should be counted due to (not) being selected (multiple times) for the sample. Items for which this value is 0 will not be included in the evaluation.

x — a numeric value or vector of values equal to or larger than 0 specifying the sum of (proportional) misstatements in the sample or, if this is a vector, the sum of taints in each stratum. If this argument is specified, the input for the data, values and values.audit arguments is discarded and it is assumed that the data come from summary statistics specified by x and n.

n — an integer or vector of integers larger than 0 specifying the sum of (proportional) misstatements in the sample or, if this is a vector, the sum of taints in each stratum. If this argument is specified, the input for the data, values and values.audit arguments is discarded and it is assumed that the data come from summary statistics specified by x and n.

N.units — a numeric value or vector of values than 0 specifying the total number of units in the population or, if this is a vector, the total number of units in each stratum of the population. This argument is strictly required for the hypergeometric, direct, difference, quotient, and regression methods, but is also used in stratification to weigh the estimates of each individual stratum to arrive at the population estimate. If NULL, each stratum is assumed to be equally represented in the population.

N.items — an integer larger than 0 specifying the number of items in the population. Only used for methods direct, difference, quotient, and regression.

pooling — a character specifying the type of model to use when analyzing stratified samples. Possible options are none (default) for no pooling (i.e., no information is shared between strata), complete for complete pooling (i.e., all information is shared between strata) or partial for partial pooling (i.e., some information is shared between strata). The latter two options fit a Bayesian model to the data using a MCMC sampling procedure whose options can be set globally using options("mc.iterations") (otherwise: 2000), options("mc.warmup") (otherwise: 1000), options("mc.chains") (otherwise: 4) and options("mc.cores") (otherwise: 1).

`prior` a logical specifying whether to use a prior distribution, or an object of class `jfaPrior` or `jfaPosterior`. If this argument is specified as `FALSE` (default), the function performs classical evaluation. If this argument is specified as `TRUE` or as a prior from `auditPrior`, this function performs Bayesian evaluation using the specified prior.

Details

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

- `poisson`: Evaluates the sample with the Poisson distribution. If combined with `prior = TRUE`, performs Bayesian evaluation using a *gamma* prior.
- `binomial`: Evaluates the sample with the binomial distribution. If combined with `prior = TRUE`, performs Bayesian evaluation using a *beta* prior.
- `hypergeometric`: Evaluates the sample with the hypergeometric distribution. If combined with `prior = TRUE`, performs Bayesian evaluation using a *beta-binomial* prior.
- `inflated.poisson`: Inflated Poisson model incorporating the explicit probability of misstatement being zero. If `prior = TRUE`, performs Bayesian evaluation using a *beta* prior.
- `hurdle.beta`: Hurdle beta model incorporating the explicit probability of a taint being zero, one, or in between. If `prior = TRUE`, this setup performs Bayesian evaluation using a *beta* prior.
- `stringer.poisson`: Evaluates the sample with the Stringer bound using the Poisson distribution.
- `stringer.binomial`: Evaluates the sample with the Stringer bound using the binomial distribution (Stringer, 1963).
- `stringer.hypergeometric`: Evaluates the sample with the Stringer bound using the hypergeometric distribution.
- `stringer.meikle`: Evaluates the sample using the Stringer bound with Meikle's correction for understatements (Meikle, 1972).
- `stringer.lta`: Evaluates the sample using the Stringer bound with LTA correction for understatements (Leslie, Teitlebaum, and Anderson, 1979).
- `stringer.pvz`: Evaluates the sample using the Stringer bound with Pap and van Zuijlen's correction for understatements (Pap and van Zuijlen, 1996).
- `rohrbach`: Evaluates the sample using Rohrbach's augmented variance bound (Rohrbach, 1993).
- `moment`: Evaluates the sample using the modified moment bound (Dworin and Grimlund, 1984).
- `coxsnell`: Evaluates the sample using the Cox and Snell bound (Cox and Snell, 1979).
- `mpu`: Evaluates the sample with the mean-per-unit estimator using the Normal distribution.
- `pps`: Evaluates the sample with the proportional-to-size estimator using the Student-t distribution.
- `direct`: Evaluates the sample using the direct estimator (Touw and Hoogduin, 2011).
- `difference`: Evaluates the sample using the difference estimator (Touw and Hoogduin, 2011).
- `quotient`: Evaluates the sample using the quotient estimator (Touw and Hoogduin, 2011).
- `regression`: Evaluates the sample using the regression estimator (Touw and Hoogduin, 2011).

Value

An object of class `jfaEvaluation` containing:

`conf.level`	a numeric value between 0 and 1 giving the confidence level.
`mle`	a numeric value between 0 and 1 giving the most likely misstatement in the population as a fraction.
`ub`	a numeric value between 0 and 1 giving the upper bound for the misstatement in the population.
`lb`	a numeric value between 0 and 1 giving the lower bound for the misstatement in the population.
`precision`	a numeric value between 0 and 1 giving the difference between the most likely misstatement and the bound relative to `alternative`.
`p.value`	for classical tests, a numeric value giving the p-value.
`x`	an integer larger than, or equal to, 0 giving the number of misstatements in the sample.
`t`	a value larger than, or equal to, 0, giving the sum of proportional misstatements in the sample.
`n`	an integer larger than 0 giving the sample size.
`materiality`	if `materiality` is specified, a numeric value between 0 and 1 giving the performance materiality as a fraction.
`alternative`	a character indicating the alternative hypothesis.
`method`	a character the method used.
`N.units`	if `N.units` is specified, in integer larger than 0 indicating the number of units in the population
`N.items`	if `N.items` is specified, in integer larger than 0 indicating the number of items in the population.
`K`	if `method = 'hypergeometric'`, an integer indicating the assumed total errors in the population.
`prior`	an object of class `jfaPrior` that contains the prior distribution.
`posterior`	an object of class `jfaPosterior` that contains the posterior distribution.
`data`	a data frame containing the relevant columns from the `data`.
`strata`	a data frame containing the relevant statistical results for the strata.
`data.name`	a character giving the name of the data.

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. doi:10.1093/biomet/66.1.125.

Derks, K., de Swart, J., van Batenburg, P., Wagenmakers, E.-J., & Wetzels, R. (2021). Priors in a Bayesian audit: How integration of existing information into the prior distribution can improve audit transparency and efficiency. *International Journal of Auditing*, 25(3), 621-636. doi:10.1111/ijau.12240

Derks, K., de Swart, J., Wagenmakers, E.-J., Wille, J., & Wetzels, R. (2021). JASP for audit: Bayesian tools for the auditing practice. *Journal of Open Source Software*, *6*(68), 2733. doi:10.21105/joss.02733

Derks, K., de Swart, J., Wagenmakers, E.-J., & Wetzels, R. (2024). The Bayesian approach to audit evidence: Quantifying statistical evidence using the Bayes factor. *Auditing: A Journal of Practice & Theory*. doi:10.2308/AJPT2021086

Derks, K., de Swart, J., Wagenmakers, E.-J., & Wetzels, R. (2022). An impartial Bayesian hypothesis test for audit sampling. *PsyArXiv*. doi:10.31234/osf.io/8nf3e

Derks, K., de Swart, J., Wagenmakers, E.-J., & Wetzels, R. (2022). Bayesian generalized linear modeling for audit sampling: How to incorporate audit information into the statistical model. *PsyArXiv*. doi:10.31234/osf.io/byj2a

Dworin, L. D. and Grimlund, R. A. (1984). Dollar-unit sampling for accounts receivable and inventory. *The Accounting Review*, 59(2), 218-241. https://www.jstor.org/stable/247296

Leslie, D. A., Teitlebaum, A. D., & Anderson, R. J. (1979). *Dollar-unit Sampling: A Practical Guide for Auditors*. Copp Clark Pitman; Belmont, CA. ISBN: 9780773042780.

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

Pap, G., and van Zuijlen, M. C. (1996). On the asymptotic behavior of the Stringer bound. *Statistica Neerlandica*, 50(3), 367-389. doi:10.1111/j.14679574.1996.tb01503.x.

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

```
# Using summary statistics
evaluation(materiality = 0.05, x = 0, n = 100) # Non-stratified
evaluation(materiality = 0.05, x = c(2, 1, 0), n = c(50, 70, 40)) # Stratified

# Using data
data("BuildIt")
```

```
BuildIt$inSample <- c(rep(1, 100), rep(0, 3400))
levs <- c("low", "medium", "high")
BuildIt$stratum <- factor(c(levs[3], levs[2], rep(levs, times = 1166)))
sample <- subset(BuildIt, BuildIt$inSample == 1)

# Non-stratified evaluation
evaluation(
  materiality = 0.05, data = sample,
  values = "bookValue", values.audit = "auditValue"
)
# Stratified evaluation
evaluation(
  materiality = 0.05, data = sample, values = "bookValue",
  values.audit = "auditValue", strata = "stratum"
)
```

`fairness_selection` *Algorithm Auditing: Fairness Measures Selection*

Description

This function aims to provide a fairness measure tailored to a specific context and dataset by answering the questions in the developed decision-making workflow. The questions within the decision-making workflow are based on observable data characteristics, the properties of fairness measures and the information required for their calculation. However, these questions are posed to the user in an easily understandable manner, requiring no statistical background or in-depth knowledge of the fairness measures. Obtainable fairness measures include Disparate Impact, Equalized Odds, Predictive Rate Parity, Equal Opportunity, Specificity Parity, Negative Predictive Rate Parity, Accuracy Parity, False Positive Rate Parity and False Negative Rate Parity. The function returns an object of class `jfaFairnessSelection` that can be used with associated `print()` and `plot()` methods.

Usage

```
fairness_selection(
  q1 = NULL,
  q2 = NULL,
  q3 = NULL,
  q4 = NULL
)
```

Arguments

q1	a character indicating the answer to the first question of the decision-making workflow ('Is the information on the true values of the classification relevant in your context?'). If `NULL` (the default) the user is presented with the first question of the decision-making corresponding to their desired answer. Possible options are `NULL` (default), `1` (to indicate 'Yes'), or `2` (to indicate 'No').

q2 a character indicating the answer to the second question of the decision-making workflow ('In what type of classification are you interested?'). If NULL (the default) the user is presented with the second question of the decision-making workflow and can respond interactively by selecting the numerical value corresponding to their desired answer. Possible options are NULL (default), 1 (to indicate 'Correct classification'), 2 (to indicate 'Incorrect classification'), or 3 (to indicate 'Correct and incorrect classification').

q3 a character indicating the answer to the third question of the decision-making workflow ('What is more important: a correct classification of the positive class, a correct classification of the negative class, or both?'). If NULL (the default) the user is presented with the third question of the decision-making workflow and can respond interactively by selecting the numerical value corresponding to their desired answer. Possible options are NULL (default), 1 (to indicate 'Correct classificvation of the positive class'), 2 (to indicate 'Correct classificvation of the negative class') or 3 (to indicate 'Both a correct classification of the positive and of the negative class').

q4 a character indicating the answer to the fourth question of the decision-making workflow ('What are the errors with the highest cost?'). If NULL (the default) the user is presented with the fourth question of the decision-making workflow and can respond interactively by selecting the numerical value corresponding to their desired answer. Possible options are NULL (default), 1 (to indicate 'False Positive') or 2 (to indicate 'False Negative').

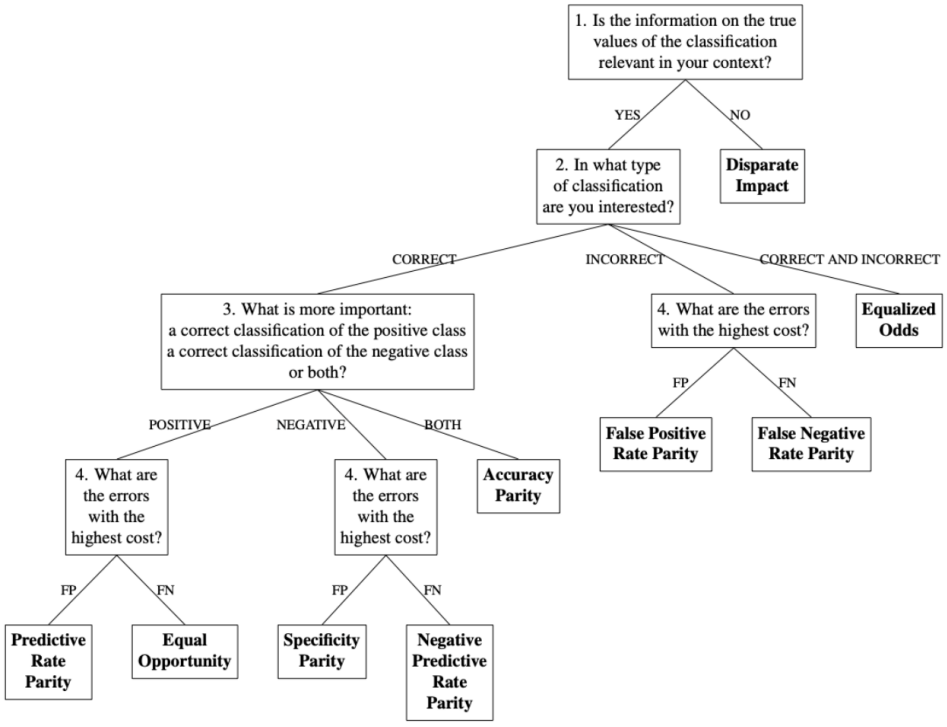
Details

Several fairness measures can be used to assess the fairness of AI-predicted classifications. These include:

- Disparate Impact. See Friedler et al. (2019), Feldman et al. (2015), Castelnovo et al. (2022) for a more detailed explanation of this measure.
- Equalized Odds. See Hardt et al. (2016), Verma et al. (2018) and Picogna et al. (2025) for a more detailed explanation of this measure.
- False Positive Rate Parity. See Castelnovo et al. (2022) (under the name Predictive Equality), Verma et al. (2018) and Picogna et al. (2025) for a more detailed explanation of this measure.
- False Negative Rate Parity. See Castelnovo et al. (2022) (under the name Equality of Opportunity), Verma et al. (2018) and and Picogna et al. (2025) for a more detailed explanation of this measure.
- Predictive Rate Parity. See Castelnovo et al. (2022) (under the name Predictive Parity) and Picogna et al. (2025) for a more detailed explanation of this measure.
- Equal Opportunity. See Hardt et al. (2016), Friedler et al. (2019), Verma et al. (2018) and Picogna et al. (2025) for a more detailed explanation of this measure.
- Specificity Parity. See Friedler et al. (2019), Verma et al. (2018) and Picogna et al. (2025) for a more detailed explanation of this measure.
- Negative Predictive Value Parity. See Verma et al. (2018) and Picogna et al. (2025) for a more detailed explanation of this measure.

- Accuracy Parity. See Friedler et al. (2019) and Picogna et al. (2025) for a more detailed explanation of this measure.

The fairness decision-making workflow below aids in choosing which fairness measure is appropriate for the situation at hand (Picogna et al., 2025).



Value

An object of class `jfaFairnessSelection` containing:

`measure`	The abbreviation for the selected fairness measure's name.
`name`	The name of the selected fairness measure.

Author(s)

Federica Picogna, `<f.picogna@nyenrode.nl>`

References

Castelnovo, A., Crupi, R., Greco, G. et al. (2022). A clarification of the nuances in the fairness metrics landscape. In *Sci Rep 12, 4209*. doi:10.1038/s41598022079391

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., & Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. doi:10.1145/2783258.2783311

Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P., & Roth, D. (2019). A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. doi:10.1145/3287560.3287589

Hardt M. , Price E., Srebro N. (2016). Equality of opportunity in supervised learning. In *Advances in neural information processing systems, 29*. doi:10.48550/arXiv.1610.02413

Picogna, F., de Swart, J., Kaya, H., & Wetzels, R. (2025). How to choose a fairness measure: A decision-making workflow for auditors. doi:10.31219/osf.io/cpxmf_v1

Verma S., Rubin J. (2018). Fairness definitions explained. In *Proceedings of the international workshop on software fairness, 1–7*. doi:10.1145/3194770.3194776

Examples

```
# Workflow leading to predictive rate parity
fairness_selection(q1 = 1, q2 = 1, q3 = 1, q4 = 1)
```

`jfa-methods` *Methods for jfa objects*

Description

Methods defined for objects returned from the `auditPrior`, `planning`, `selection`, and `evaluation` functions.

Usage

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

## S3 method for class 'summary.jfaPrior'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'jfaPrior'
summary(object, digits = getOption("digits"), ...)

## S3 method for class 'jfaPrior'
predict(object, n, cumulative = FALSE, ...)

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

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

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

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

## S3 method for class 'summary.jfaPosterior'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'jfaPosterior'
summary(object, digits = getOption("digits"), ...)

## S3 method for class 'jfaPosterior'
predict(object, n, cumulative = FALSE, ...)

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

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

## S3 method for class 'summary.jfaPlanning'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'jfaPlanning'
summary(object, digits = getOption("digits"), ...)

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

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

## S3 method for class 'summary.jfaSelection'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'jfaSelection'
summary(object, digits = getOption("digits"), ...)

## S3 method for class 'jfaEvaluation'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'summary.jfaEvaluation'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'jfaEvaluation'
summary(object, digits = getOption("digits"), ...)

## S3 method for class 'jfaEvaluation'
plot(x, type = c("estimates", "posterior", "sequential"), ...)
```

```
## S3 method for class 'jfaDistr'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'summary.jfaDistr'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'jfaDistr'
summary(object, digits = getOption("digits"), ...)

## S3 method for class 'jfaDistr'
plot(x, type = c("estimates", "robustness", "sequential"), ...)

## S3 method for class 'jfaRv'
print(x, digits = getOption("digits"), ...)

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

## S3 method for class 'jfaFairness'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'summary.jfaFairness'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'jfaFairness'
summary(object, digits = getOption("digits"), ...)

## S3 method for class 'jfaFairness'
plot(x, type = c("estimates", "posterior", "robustness", "sequential"), ...)

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

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

Arguments

`...`	further arguments, currently ignored.
`digits`	an integer specifying the number of digits to which output should be rounded. Used in `summary`.
`object, x`	an object of class `jfaPrior`, `jfaPosterior`, `jfaPlanning`, `jfaSelection`, `jfaEvaluation`, `jfaDistr`, `jfaRv`, or `jfaFairness`.
`n`	used in `predict`. Specifies the sample size for which predictions should be made.
`cumulative`	used in `predict`. Specifies whether cumulative probabilities should be shown.
`type`	used in `plot`. Specifies the type of plot to produce.

Value

The summary methods return a data.frame which contains the input and output.

The print methods simply print and return nothing.

---

model_fairness *Algorithm Auditing: Fairness Metrics and Parity*

Description

This function aims to assess fairness in algorithmic decision-making systems by computing and testing the equality of one of several model-agnostic fairness metrics between protected classes. The metrics are computed based on a set of true labels and the predictions of an algorithm. The ratio of these metrics between any unprivileged protected class and the privileged protected class is called parity. This measure can quantify potential fairness or discrimination in the algorithms predictions. Available parity metrics include predictive rate parity, proportional parity, accuracy parity, false negative rate parity, false positive rate parity, true positive rate parity, negative predictive value parity, specificity parity, and demographic parity. The function returns an object of class jfaFairness that can be used with associated summary() and plot() methods.

Usage

```
model_fairness(
  data,
  protected,
  target,
  predictions,
  privileged = NULL,
  positive = NULL,
  metric = c(
    "prp", "pp", "ap", "fnrp", "fprp",
    "tprp", "npvp", "sp", "dp", "eo"
  ),
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  prior = FALSE
)
```

Arguments

data	a data frame containing the input data.
protected	a character specifying the column name in data containing the protected classes (i.e., the sensitive attribute).
target	a character specifying the column name in data containing the true labels of the target (i.e., to be predicted) variable.
predictions	a character specifying the column name in data containing the predicted labels of the target variable.

privileged	a character specifying the factor level of the column protected to be used as the privileged group. If NULL (the default), the first factor level of the protected column is used.
positive	a character specifying the factor level positive class of the column target to be used as the positive class. If NULL (the default), the first factor level of the target column is used.
metric	a character indicating the fairness metrics to compute. This can also be an object of class jfaFairnessSelection as returned by the fairness_selection function. See the Details section below for more information on possible fairness metrics.
alternative	a character indicating the alternative hypothesis and the type of confidence / credible interval used in the individual comparisons to the privileged group. Possible options are two.sided (default), less, or greater. The alternative hypothesis relating to the overall equality is always two sided.
conf.level	a numeric value between 0 and 1 specifying the confidence level (i.e., 1 - audit risk / detection risk).
prior	a logical specifying whether to use a prior distribution, or a numeric value equal to or larger than 1 specifying the prior concentration parameter. If this argument is specified as FALSE (default), classical estimation is performed and if it is TRUE, Bayesian estimation using a default prior with concentration parameter 1 is performed.

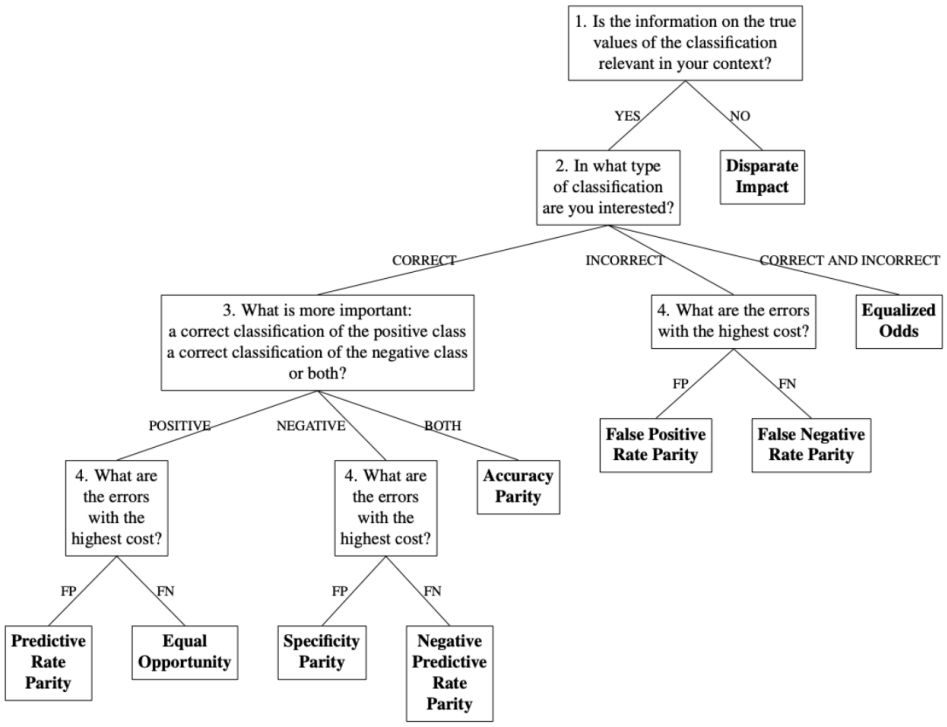
Details

The following model-agnostic fairness metrics are computed based on the confusion matrix for each protected class, using the true positives (TP), false positives (FP), true negative (TN) and false negatives (FN). See Pessach & Shmueli (2022) for a more detailed explanation of the individual metrics. The equality of metrics across groups is done according to the methodology described in Fisher (1970) and Jamil et al. (2017).

- Predictive rate parity (prp): calculated as TP / (TP + FP), its ratio quantifies whether the predictive rate is equal across protected classes.
- Proportional parity (pp): calculated as (TP + FP) / (TP + FP + TN + FN), its ratio quantifies whether the positive prediction rate is equal across protected classes.
- Accuracy parity (ap): calculated as (TP + TN) / (TP + FP + TN + FN), quantifies whether the accuracy is the same across groups.
- False negative rate parity (fnrp): calculated as FN / (TP + FN), quantifies whether the false negative rate is the same across groups.
- False positive rate parity (fprp): calculated as FP / (TN + FP), quantifies whether the false positive rate is the same across groups.
- True positive rate parity (tprp, also known as equal opportunity): calculated as TP / (TP + FN), quantifies whether the true positive rate is the same across groups.
- Negative predictive value parity (npvp): calculated as TN / (TN + FN), quantifies whether the negative predictive value is equal across groups.
- Specificity parity (sp): calculated as TN / (TN + FP), quantifies whether the true positive rate is the same across groups.

- Demographic parity (`dp`): calculated as TP + FP, quantifies whether the positive predictions are equal across groups.
- Equalized odds (`eo`): calculated as a combination of the true positive rate and the false positive rate, quantifies whether the true positive rate and, simultaneously, the false positive rate are the same across groups

Note that, in an audit context, not all fairness measures are equally appropriate in all situations. The fairness tree below aids in choosing which fairness measure is appropriate for the situation at hand (Picogna et al., 2025).



Value

An object of class `jfaFairness` containing:

`data`	the specified data.
`conf.level`	a numeric value between 0 and 1 giving the confidence level.
`privileged`	The privileged group for computing the fairness metrics.
`unprivileged`	The unprivileged group(s).
`target`	The target variable used in computing the fairness metrics.
`predictions`	The predictions used to compute the fairness metrics.
`protected`	The variable indicating the protected classes.
`positive`	The positive class used in computing the fairness metrics.

`negative`	The negative class used in computing the fairness metrics.
`alternative`	The type of confidence interval.
`confusion.matrix`	A list of confusion matrices for each group.
`performance`	A data frame containing performance metrics for each group, including accuracy, precision, recall, and F1 score.
`metric`	A data frame containing, for each group, the estimates of the fairness metric along with the associated confidence / credible interval.
`parity`	A data frame containing, for each unprivileged group, the parity and associated confidence / credible interval when compared to the privileged group.
`odds.ratio`	A data frame containing, for each unprivileged group, the odds ratio of the fairness metric and its associated confidence/credible interval, along with inferential measures such as uncorrected p-values or Bayes factors.
`measure`	The abbreviation of the selected fairness metric.
`prior`	a logical indicating whether a prior distribution was used.
`data.name`	The name of the input data object.

Author(s)

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

References

Calders, T., & Verwer, S. (2010). Three naive Bayes approaches for discrimination-free classification. In *Data Mining and Knowledge Discovery*. Springer Science and Business Media LLC. doi:10.1007/s106180100190x

Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. In *Big Data*. Mary Ann Liebert Inc. doi:10.1089/big.2016.0047

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., & Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. doi:10.1145/2783258.2783311

Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P., & Roth, D. (2019). A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. doi:10.1145/3287560.3287589

Fisher, R. A. (1970). *Statistical Methods for Research Workers*. Oliver & Boyd.

Jamil, T., Ly, A., Morey, R. D., Love, J., Marsman, M., & Wagenmakers, E. J. (2017). Default "Gunel and Dickey" Bayes factors for contingency tables. *Behavior Research Methods*, 49, 638-652. doi:10.3758/s1342801607398

Pessach, D. & Shmueli, E. (2022). A review on fairness in machine learning. *ACM Computing Surveys*, 55(3), 1-44. doi:10.1145/3494672

Picogna, F., de Swart, J., Kaya, H., & Wetzels, R. (2025). How to choose a fairness measure: A decision-making workflow for auditors. doi:10.31219/osf.io/cpxmf_v1

Zafar, M. B., Valera, I., Gomez Rodriguez, M., & Gummadi, K. P. (2017). Fairness beyond disparate treatment & disparate impact. In *Proceedings of the 26th International Conference on World Wide Web*. doi:10.1145/3038912.3052660

Examples

```
# Frequentist test of specificy parity
model_fairness(
  data = compas,
  protected = "Gender",
  target = "TwoYrRecidivism",
  predictions = "Predicted",
  privileged = "Male",
  positive = "yes",
  metric = "sp"
)
```

---

planning *Audit Sampling: Planning*

Description

`planning()` is used to calculate a minimum sample size for audit samples. It allows specification of statistical requirements for the sample with respect to the performance materiality or the precision. The function returns an object of class `jfaPlanning` that can be used with associated `summary()` and `plot()` methods.

Usage

```
planning(
  materiality = NULL,
  min.precision = NULL,
  expected = 0,
  likelihood = c("poisson", "binomial", "hypergeometric"),
  conf.level = 0.95,
  N.units = NULL,
  by = 1,
  max = 5000,
  prior = FALSE
)
```

Arguments

`materiality`	a numeric value between 0 and 1 specifying the performance materiality (i.e., the maximum tolerable misstatement in the population) as a fraction. Can be `NULL`, but `min.precision` should be specified in that case.
`min.precision`	a numeric value between 0 and 1 specifying the minimum precision (i.e., the estimated upper bound minus the estimated most likely error) as a fraction. Can be `NULL`, but `materiality` should be specified in that case.
`expected`	a numeric value between 0 and 1 specifying the expected (tolerable) misstatements in the sample relative to the total sample size, or a number (>= 1) specifying the expected (tolerable) number of misstatements in the sample. It is advised

to set this value conservatively to minimize the probability of the observed misstatements in the sample exceeding the expected misstatements, which would imply that insufficient work has been done in the end and that additional samples are required. This argument also facilitates sequential sampling plans since it can also be a vector (e.g., `c(1, 0)`) of tolerable misstatements in each stage of the audit sample. Hence, `c(1, 0)` gives the sample size for a sequential sampling plan in which the auditor can stop after seeing 0 misstatements in the first sample, but can extend the sample if more than 0 misstatements are found.

`likelihood` a character specifying the likelihood of the data. Possible options are `poisson` (default) for the Poisson likelihood, `binomial` for the binomial likelihood, or `hypergeometric` for the hypergeometric likelihood. See the details section for more information about the possible likelihoods.

`conf.level` a numeric value between 0 and 1 specifying the confidence level (i.e., 1 - audit risk / detection risk).

`N.units` a numeric value larger than 0 specifying the total number of units in the population. Required for the `hypergeometric` likelihood.

`by` an integer larger than 0 specifying the increment between acceptable sample sizes (e.g., `increment = 5` considers only sample sizes of 5, 10, 15, ...).

`max` an integer larger than 0 specifying the sample size at which the algorithm terminates (e.g., `max = 100` will terminate the algorithm at $n = 100$).

`prior` a logical specifying whether to use a prior distribution, or an object of class `jfaPrior` or `jfaPosterior`. If this argument is specified as `FALSE` (default), the function performs classical planning. If this argument is specified as `TRUE` or as a prior from `auditPrior`, this function performs Bayesian planning using a prior that is conjugate to the specified `likelihood`.

Details

This section elaborates on the available input options for the `likelihood` argument and the corresponding conjugate prior distributions used by `jfa`.

- `poisson`: The Poisson distribution is an approximation of the binomial distribution. The Poisson distribution is defined as:

$$f(\theta, n) = \frac{\lambda^{\theta} e^{-\lambda}}{\theta!}$$

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

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

.

- `binomial`: The binomial distribution is an approximation of the hypergeometric distribution. The binomial distribution is defined as:

$$f(\theta, n, x) = \binom{n}{x} \theta^{x} (1-\theta)^{n-x}$$

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

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

.

- `hypergeometric`: The hypergeometric distribution is defined as:

$$f(x, n, K, N) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}$$

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

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

.

Value

An object of class `jfaPlanning` containing:

`conf.level`	a numeric value between 0 and 1 giving the confidence level.
`x`	a numeric value larger than, or equal to, 0 giving (the proportional sum of) the tolerable errors in the sample.
`n`	an integer larger than 0 giving the minimum sample size.
`n_staged`	in the case of a multi-stage sampling plan, an integer larger than 0 giving the minimum sample size per stage.
`ub`	a numeric value between 0 and 1 giving the expected upper bound.
`precision`	a numeric value between 0 and 1 giving the expected precision.
`p.value`	a numeric value giving the expected one-sided p-value.
`K`	if `likelihood = 'hypergeometric'`, an integer larger than 0 giving the assumed population errors.
`N.units`	an integer larger than 0 giving the number of units in the population (only returned if `N.units` is specified).
`materiality`	a numeric value between 0 and 1 giving the performance materiality if specified.
`min.precision`	a numeric value between 0 and 1 giving the minimum precision if specified.
`expected`	a numeric value larger than, or equal to, 0 giving the expected misstatement input.
`likelihood`	a character indicating the likelihood.
`errorType`	a character indicating the expected misstatements input.
`iterations`	an integer giving the number of iterations of the algorithm.
`prior`	if a prior distribution is specified, an object of class `jfaPrior` that contains information about the prior distribution.
`posterior`	if a prior distribution is specified, an object of class `jfaPosterior` that contains information about the expected posterior distribution.

Author(s)

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

References

Derks, K., de Swart, J., van Batenburg, P., Wagenmakers, E.-J., & Wetzels, R. (2021). Priors in a Bayesian audit: How integration of existing information into the prior distribution can improve audit transparency and efficiency. *International Journal of Auditing*, 25(3), 621-636. doi:10.1111/ijau.12240

Derks, K., de Swart, J., Wagenmakers, E.-J., Wille, J., & Wetzels, R. (2021). JASP for audit: Bayesian tools for the auditing practice. *Journal of Open Source Software*, *6*(68), 2733. doi:10.21105/joss.02733

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. doi:10.1080/03610929308831139

See Also

`auditPrior` `selection` `evaluation`

Examples

```
# Classical planning
planning(materiality = 0.03, expected = 0)

# Classical two-stage planning
planning(materiality = 0.03, expected = c(1, 0))

# Bayesian planning using a default prior
planning(materiality = 0.03, prior = TRUE)

# Bayesian planning using a custom prior
prior <- auditPrior(method = "impartial", materiality = 0.05)
planning(materiality = 0.05, prior = prior)
```

---

repeated_test *Data Auditing: Repeated Values Test*

Description

This function analyzes the frequency with which values get repeated within a set of numbers. Unlike Benford's law, and its generalizations, this approach examines the entire number at once, not only the first or last digit(s).

Usage

```
repeated_test(
  x,
  check = c("last", "lasttwo", "all"),
  method = c("af", "entropy"),
  samples = 2000
)
```

Arguments

x	a numeric vector of values from which the digits should be analyzed.
check	which digits to shuffle during the procedure. Can be last or lasttwo.
method	which statistics is used. Defaults to af for average frequency, but can also be entropy for entropy.
samples	how many samples to use in the bootstraping procedure.

Details

To determine whether the data show an excessive amount of bunching, the null hypothesis that x does not contain an unexpected amount of repeated values is tested against the alternative hypothesis that x has more repeated values than expected. The statistic can either be the average frequency ($AF = sum(f_i^2)/sum(f_i)$) of the data or the entropy ($E = -sum(p_i * log(p_i))$, with $p_i = f_i/n$) of the data. Average frequency and entropy are highly correlated, but the average frequency is often more interpretable. For example, an average frequency of 2.5 means that, on average, your observations contain a value that appears 2.5 times in the data set.To quantify what is expected, this test requires the assumption that the integer portions of the numbers are not associated with their decimal portions.

Value

An object of class jfaRv containing:

x	input data.
frequencies	frequencies of observations in x.
samples	vector of simulated samples.
integers	counts for extracted integers.
decimals	counts for extracted decimals.
n	the number of observations in x.
statistic	the value the average frequency or entropy statistic.
p.value	the p-value for the test.
cor.test	correlation test for the integer portions of the number versus the decimals portions of the number.
method	method used.
check	checked digits.
data.name	a character string giving the name(s) of the data.

Author(s)

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

References

Simohnsohn, U. (2019, May 25). Number-Bunching: A New Tool for Forensic Data Analysis. Retrieved from https://datacolada.org/77.

See Also

digit_test

Examples

```
set.seed(1)
x <- rnorm(50)

# Repeated values analysis shuffling last digit
repeated_test(x, check = "last", method = "af", samples = 2000)
```

---

retailer *Retailer Group Audit*

Description

Sample outcomes summarized per branch from an audit of a retail company consisting of 20 branches.

Usage

```
data(retailer)
```

Format

A data frame with 20 rows and 5 variables.

stratum branch/stratum number.

items total number of items in each branch.

samples number of items in sample per branch.

errors number of errors in sample per branch.

Source

Derks, K., de Swart, J., & Wetzels, R. (2022). Een Bayesiaanse blik op gestratificeerde steekproeven heeft voordelen voor de auditor. *Maandblad voor Accountancy en Bedrijfseconomie*, 96(1/2), 37-46. doi:10.5117/mab.96.78836

Examples

```
data(retailer)
```

`sanitizer` *Factory Workers' use of Hand Sanitizer*

Description

Data from a study on factory workers' use of hand sanitizer. Sanitizer use was measured to a 100th of a gram.

Usage

```
data(sanitizer)
```

Format

A data frame with 1600 rows and 1 variable.

Source

http://datacolada.org/appendix/74/

References

[Retracted] Li, M., Sun, Y., & Chen, H. (2019). The decoy effect as a nudge: Boosting hand hygiene with a worse option. Psychological Science, 30, 139–149.

Examples

```
data(sanitizer)
```

`selection` *Audit Sampling: Selection*

Description

`selection()` is used to perform statistical selection of audit samples. It offers flexible implementations of the most common audit sampling algorithms for attributes sampling and monetary unit sampling. The function returns an object of class `jfaSelection` that can be used with the associated `summary()` method.

Usage

```
selection(
  data,
  size,
  units = c("items", "values"),
  method = c("interval", "cell", "random", "sieve"),
  values = NULL,
  order = NULL,
  decreasing = FALSE,
  randomize = FALSE,
  replace = FALSE,
  start = 1
)
```

Arguments

`data`	a data frame containing the population data.
`size`	an integer larger than 0 specifying the number of units to select. Can also be an object of class `jfaPlanning`.
`units`	a character specifying the type of sampling units. Possible options are `items` (default) for selection on the level of items (rows) or `values` for selection on the level of monetary units.
`method`	a character specifying the sampling algorithm. Possible options are `interval` (default) for fixed interval sampling, `cell` for cell sampling, `random` for random sampling, or `sieve` for modified sieve sampling.
`values`	a character specifying the name of a column in `data` containing the book values of the items.
`order`	a character specifying the name of a column in `data` containing the ranks of the items. The items in the `data` are ordered according to these values in the order indicated by `decreasing`.
`decreasing`	a logical specifying whether to order the items from smallest to largest. Only used if `order` is specified.
`randomize`	a logical specifying if items should be randomly shuffled prior to selection. Note that `randomize = TRUE` overrules `order`.
`replace`	a logical specifying if sampling units should be selected with replacement. Only used for method `random` when selecting items.
`start`	an integer larger than 0 specifying index of the unit that should be selected. Only used for method `interval`.

Details

This section elaborates on the possible options for the `units` argument:

- `items`: In attributes sampling each item in the population is a sampling unit. An item with a book value of $5000 is therefore equally likely to be selected as an item with a book value of $500.

- values: In monetary unit sampling each monetary unit in the population is a sampling unit. An item with a book value of $5000 is therefore ten times more likely to be selected as an item with a book value of $500.

This section elaborates on the possible options for the method argument:

- interval: In fixed interval sampling the sampling units are divided into a number of equally large intervals. In each interval, a single sampling unit is selected according to a fixed starting point (specified by start).
- cell: In cell sampling the sampling units in the population are divided into a number (equal to the sample size) of equally large intervals. In each interval, a single sampling unit is selected randomly.
- random: In random sampling all sampling units are drawn with equal probability.
- sieve: In modified sieve sampling items are selected with the largest sieve ratio (Hoogduin, Hall, & Tsay, 2010).

Value

An object of class jfaSelection containing:

data	a data frame containing the population data.
sample	a data frame containing the selected data sample.
n.req	an integer giving the requested sample size.
n.units	an integer giving the number of obtained sampling units.
n.items	an integer giving the number of obtained sample items.
N.units	an integer giving the number of sampling units in the population data.
N.items	an integer giving the number of items in the population data.
interval	if method = "interval", a numeric value giving the size of the selection interval.
units	a character indicating the type of sampling units.
method	a character indicating the sampling algorithm.
values	if values is specified, a character indicating the book value column.
start	if method = "interval", an integer giving the index of the selected unit in each interval.
data.name	a character indicating the name of the population data.

Author(s)

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

References

Derks, K., de Swart, J., Wagenmakers, E.-J., Wille, J., & Wetzels, R. (2021). JASP for audit: Bayesian tools for the auditing practice. *Journal of Open Source Software*, *6*(68), 2733. doi:10.21105/joss.02733

Hoogduin, L. A., Hall, T. W., & Tsay, J. J. (2010). Modified sieve sampling: A method for single- and multi-stage probability-proportional-to-size sampling. *Auditing: A Journal of Practice & Theory*, 29(1), 125-148. doi:10.2308/aud.2010.29.1.125

Leslie, D. A., Teitlebaum, A. D., & Anderson, R. J. (1979). *Dollar-unit Sampling: A Practical Guide for Auditors*. Copp Clark Pitman; Belmont, CA. ISBN: 9780773042780.

See Also

`auditPrior` `planning` `evaluation`

Examples

```
data("BuildIt")

# Select 100 items using random sampling
set.seed(1)
selection(data = BuildIt, size = 100, method = "random")

# Select 150 monetary units using fixed interval sampling
selection(
  data = BuildIt, size = 150, units = "values",
  method = "interval", values = "bookValue"
)
```

`sinoForest` *Sino Forest Corporation's Financial Statements.*

Description

Financial statement numbers of Sino Forest Corporation's 2010 Report.

Usage

```
data(sinoForest)
```

Format

A data frame with 772 rows and 1 variable.

Source

https://cran.r-project.org/package=benford.analysis

References

Nigrini, M. J. (2012). Benford's Law: Application for Forensic Accounting, Auditing and Fraud Detection. Wiley and Sons: New Jersey.

Examples

```
data(sinoForest)
```

Index

∗ **Bayesian**
digit_test, 13
∗ **Benford**
digit_test, 13
∗ **algorithm**
fairness_selection, 21
model_fairness, 27
∗ **audit**
auditPrior, 6
digit_test, 13
evaluation, 16
fairness_selection, 21
model_fairness, 27
planning, 31
repeated_test, 34
selection, 37
∗ **bias**
fairness_selection, 21
model_fairness, 27
∗ **datasets**
accounts, 4
allowances, 5
benchmark, 10
BuildIt, 11
carrier, 12
compas, 13
retailer, 36
sanitizer, 37
sinoForest, 40
∗ **digits**
digit_test, 13
∗ **distribution**
digit_test, 13
∗ **evaluation**
auditPrior, 6
evaluation, 16
planning, 31
∗ **fairness**
fairness_selection, 21
model_fairness, 27
∗ **items**
selection, 37
∗ **jfa**
jfa-package, 3
∗ **model**
model_fairness, 27
∗ **mus**
selection, 37
∗ **package**
jfa-package, 3
∗ **performance**
model_fairness, 27
∗ **planning**
auditPrior, 6
planning, 31
∗ **prior**
auditPrior, 6
evaluation, 16
planning, 31
∗ **repeated**
repeated_test, 34
∗ **selection**
selection, 37
∗ **values**
repeated_test, 34
∗ **workflow**
fairness_selection, 21

accounts, 4
allowances, 5
auditPrior, 6, *20*, *24*, *34*, *40*

benchmark, 10
BuildIt, 11

carrier, 12
compas, 13

digit_test, 13, *36*

evaluation, *10*, 16, *24*, *34*, *40*

fairness_selection, 21

jfa *(*jfa-package*)*, 3
jfa-methods, 24
jfa-package, 3

model_fairness, 27

planning, *10*, *20*, *24*, 31, *40*
plot.jfaDistr *(*jfa-methods*)*, 24
plot.jfaEvaluation *(*jfa-methods*)*, 24
plot.jfaFairness *(*jfa-methods*)*, 24
plot.jfaFairnessSelection
*(*jfa-methods*)*, 24
plot.jfaPlanning *(*jfa-methods*)*, 24
plot.jfaPosterior *(*jfa-methods*)*, 24
plot.jfaPredict *(*jfa-methods*)*, 24
plot.jfaPrior *(*jfa-methods*)*, 24
plot.jfaRv *(*jfa-methods*)*, 24
predict.jfaPosterior *(*jfa-methods*)*, 24
predict.jfaPrior *(*jfa-methods*)*, 24
print.jfaDistr *(*jfa-methods*)*, 24
print.jfaEvaluation *(*jfa-methods*)*, 24
print.jfaFairness *(*jfa-methods*)*, 24
print.jfaFairnessSelection
*(*jfa-methods*)*, 24
print.jfaPlanning *(*jfa-methods*)*, 24
print.jfaPosterior *(*jfa-methods*)*, 24
print.jfaPredict *(*jfa-methods*)*, 24
print.jfaPrior *(*jfa-methods*)*, 24
print.jfaRv *(*jfa-methods*)*, 24
print.jfaSelection *(*jfa-methods*)*, 24
print.summary.jfaDistr *(*jfa-methods*)*, 24
print.summary.jfaEvaluation
*(*jfa-methods*)*, 24
print.summary.jfaFairness
*(*jfa-methods*)*, 24
print.summary.jfaPlanning
*(*jfa-methods*)*, 24
print.summary.jfaPosterior
*(*jfa-methods*)*, 24
print.summary.jfaPrior *(*jfa-methods*)*, 24
print.summary.jfaSelection
*(*jfa-methods*)*, 24

repeated_test, *15*, 34
retailer, 36

sanitizer, 37
selection, *10*, *20*, *24*, *34*, 37
sinoForest, 40
summary.jfaDistr *(*jfa-methods*)*, 24
summary.jfaEvaluation *(*jfa-methods*)*, 24
summary.jfaFairness *(*jfa-methods*)*, 24
summary.jfaPlanning *(*jfa-methods*)*, 24
summary.jfaPosterior *(*jfa-methods*)*, 24
summary.jfaPrior *(*jfa-methods*)*, 24
summary.jfaSelection *(*jfa-methods*)*, 24