Audit sampling with jfa:: CHEAT SHEET



Basics

ifa is an R package that is developed to facilitate planning, selection, and evaluation of statistical audit samples in both its Bayesian and classical manifestations.

The package provides five main functions that can be used in order to facilitate an efficient audit sampling workflow.

Installation

Installing the package can be done via: install.packages("jfa")

Loading the package can be done via: library(jfa)



Create a prior probability distribution (optional)

jfa::auditPrior()

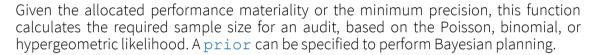
This function creates a prior distribution for Bayesian audit sampling in which several different types of audit information can be incorporated.

• likelihood: Specifies the family of the prior probability distribution.



Calculate the required sample size

jfa::planning()



• expectedError: A fraction specifying the expected errors in the sample.

planning (materiality = 0.05, expectedError = 0.01, prior = FALSE)

auditPrior(materiality = 0.05,

method = "none")

Select the required transactions from the population

jfa::selection()

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.

selection (population = dataset, sampleSize = 100)



Evaluate the audited transactions

jfa::evaluation()



This function takes a data frame (using sample, bookValues, and auditValues) or summary statistics (using nSumstats and kSumstats) and calculates the most likely error and upper confidence bound on the misstatement according to the specified method.

• prior: An object returned by the auditPrior() function that specifies the prior.

evaluation(sample = sample, bookValues = "ist", auditValues = "soll", method = "stringer", materiality = 0.05)



Create a report of the results

jfa::report()



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.

```
report (object = result,
file = "report.html")
```