Package: MUS (via r-universe)

September 7, 2024

Encoding UTF-8

Type Package

Title Monetary Unit Sampling and Estimation Methods, Widely Used in Auditing

Version 0.1.6

Date 2019-09-15

Author Henning Prömpers, André Guimarães

Maintainer Henning Prömpers <henning@proempers.net>

Description Sampling and evaluation methods to apply Monetary Unit Sampling (or in older literature Dollar Unit Sampling) during an audit of financial statements.

Depends R (>= 3.4.0), stats

Suggests DescTools, pander

License GPL (>= 2)

NeedsCompilation no

BugReports https://github.com/alsguimaraes/MUS

Date/Publication 2019-09-15 11:06:00 UTC

RoxygenNote 6.1.1

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

RemoteUrl https://github.com/alsguimaraes/mus

RemoteRef HEAD

RemoteSha 8f597c29b47831899a997d6522ace803794f4c03

Contents

`MUS-package` *Monetary Unit Sampling and Estimation Methods, Widely Used in Auditing*

Description

Sampling and evaluation methods to apply Monetary Unit Sampling (or in older literature Dollar Unit Sampling) during an audit of financial statements.

Details

Monetary Unit Sampling (MUS), also known as Dollar Unit Sampling (DUS) or Probability-Proportional-to-Size Sampling (PPS), is a sampling approach that is widely used in auditing.

This package was written mainly for a research project. However, it should be possible to use the methods for practical auditing, too. Furthermore, the package comes with ABSOLUTELY NO WARRANTY. Use it at your own risk!

You have to walk through four steps: 1. Plan a sample and determine the sample size, use function: MUS.planning 2. Extract the sample, use function: MUS.extract 3. Audit the extracted sample (e.g. by asking for debtor confirmations). 4. Evaluate the audited sample, use function: MUS.evaluation

Author(s)

Henning PrÃ¶mpers Maintainer: Henning PrÃ¶mpers <henning@proempers.net>

See Also

`MUS.planning` for planning a sample, `MUS.extraction` for extraction of the planned sample and `MUS.evaluation` for evaluation of the extracted and audited sample.

Examples

```
## Simple Example
library(MUS)
# Assume 500 invoices, each between 1 and 1000 monetary units
example.data.1 <- data.frame(book.value=round(runif(n=500, min=1,
max=1000)))
# Plan a sample and cache it
plan.results.simple <- MUS.planning(data=example.data.1,
tolerable.error=100000, expected.error=20000)
# Extract a sample and cache it (no high values exist in this example)
extract.results.simple <- MUS.extraction(plan.results.simple)
# Copy book values into a new column audit values
audited.sample.simple <- extract.results.simple$sample
audited.sample.simple <- cbind(audited.sample.simple,
audit.value=audited.sample.simple$book.value)
# Edit manually (if any audit difference occur)
#audited.sample.simple <- edit(audited.sample.simple)
# Evaluate the sample, cache and print it
evaluation.results.simple <- MUS.evaluation(extract.results.simple,
audited.sample.simple)
print(evaluation.results.simple)
```

MUS.binomial.bound *Calculate a binomial bound for a Monetary Unit Sampling evaluation.*

Description

Calculates a binomial bound for a Monetary Unit Sampling evaluation.

Please treat as experimental.

Usage

```
MUS.binomial.bound(x, scope, as.pct, include.high.values, confidence.level)
```

Arguments

x	A MUS.evaluation.result object (or a tainting vector) used to calculate the binomial bound.
scope	The required scope for the bound ("qty" or "value"). Default is "value".
as.pct	Boolean. Express results as percentage. Default is False.
include.high.values	Boolean. Whether the bound should include high values. Default is "TRUE".
confidence.level	The required confidence level. Default is 95%.

Value

Upper Error Limit calculed using the binomial bound.

Author(s)

Andre Guimaraes <alsguimaraes@gmail.com>

See Also

MUS.evaluation for evaluation of the audited sample.

Examples

```
# Assume 500 invoices, each between 1 and 1000 monetary units
data <- data.frame(book.value=round(runif(n=500, min=1, max=1000)))
# Plan a sample and cache it
plan <- MUS.planning(data=data, tolerable.error=10000, expected.error=2000)
# Extract a sample and cache it (no high values exist in this example)
extract <- MUS.extraction(plan)
# Copy book value into a new column audit values, and inject some error
audited <- extract$sample$book.value*(1-rbinom(nrow(extract$sample), 1, 0.05))
audited <- cbind(extract$sample, audit.value=audited)
# Evaluate the sample, cache and print it
evaluation <- MUS.evaluation(extract, audited)
MUS.binomial.bound(evaluation)
```

MUS.calc.n.conservative

Calculate a conservative sample size.

Description

Calculate a conservative sample size (AICPA, 2012). Based on Technical Notes on the AICPA Audit Guide Audit Sampling, Trevor Stewart, AICPA, 2012.

Usage

```
MUS.calc.n.conservative(confidence.level, tolerable.error, expected.error, book.value)
```

Arguments

confidence.level	dito.
tolerable.error	Tolerable error in monetary units.
expected.error	Expected error in monetary units.
book.value	Book value in monetary units.

Value

Returns the (conservative) sample size.

Author(s)

Andre Guimaraes <alsguimaraes@gmail.com>

Examples

```
MUS.calc.n.conservative(0.95, 100000, 50000, 10000000)
```

`MUS.combine` *Combine MUS objects (joining strata into a full set).*

Description

Combine a list of MUS objects into a single object. Typical use case is to group multiple strata into a single object. Works with MUS.planning.result, MUS.extraction.result and MUS.evaluation.result objects.

Usage

```
MUS.combine(object.list)
```

Arguments

`object.list`	A list of MUS.planning.result, MUS.extraction.result and MUS.evaluation.result objects.

Value

An object of the same type of the first item in the list is returned containing an aggregation of the objects in the list.

Author(s)

Andre Guimaraes <alsguimaraes@gmail.com>

Examples

```
## Simple Example
# Assume 500 invoices, each between 1 and 1000 monetary units
stratum.1 <- data.frame(book.value=round(runif(n=500, min=1, max=1000)))
plan.1 <- MUS.planning(data=stratum.1, tolerable.error=100000, expected.error=20000)

stratum.2 <- data.frame(book.value=round(runif(n=500, min=1, max=1000)))
plan.2 <- MUS.planning(data=stratum.2, tolerable.error=100000, expected.error=20000)

plan.combined <- MUS.combine(list(plan.1, plan.2))

print(plan.combined)
```

`MUS.combined.high.error.rate` *Calculate a high error rate bound for a combined Monetary Unit Sampling evaluation.*

Description

Calculate a high error rate bound for a combined Monetary Unit Sampling evaluation.

Please treat as experimental.

Usage

```
MUS.combined.high.error.rate(evaluation, interval.type)
```

Arguments

`evaluation`	A MUS.evaluation.result object used to calculate the combined bound.
`interval.type`	Optional. Interval type for high error rate evaluation. Default is "one-sided".

Value

Upper Error Limit calculed using high error rate evaluation for a combined sample.

Author(s)

Andre Guimaraes <alsguimaraes@gmail.com>

See Also

`MUS.evaluation` for evaluation of the audited sample. `MUS.combine` for combining multiple evaluations.

Examples

```
# Assume 500 invoices, each between 1 and 1000 monetary units
data1 <- data.frame(book.value=round(runif(n=500, min=1, max=1000)))
# Plan a sample and cache it
plan1 <- MUS.planning(data=data1, tolerable.error=10000, expected.error=2000)
# Extract a sample and cache it (no high values exist in this example)
extract1 <- MUS.extraction(plan1)
# Copy book value into a new column audit values, and inject some error
audited1 <- extract1$sample$book.value*(1-rbinom(nrow(extract1$sample), 1, 0.05))
audited1 <- cbind(extract1$sample, audit.value=audited1)
# Evaluate the sample, cache and print it
evaluation1 <- MUS.evaluation(extract1, audited1)

# Assume 500 invoices, each between 1 and 1000 monetary units
data2 <- data.frame(book.value=round(runif(n=500, min=1, max=1000)))
```

```
# Plan a sample and cache it
plan2 <- MUS.planning(data=data2, tolerable.error=10000, expected.error=2000)
# Extract a sample and cache it (no high values exist in this example)
extract2 <- MUS.extraction(plan2)
# Copy book value into a new column audit values, and inject some error
audited2 <- extract2$sample$book.value*(1-rbinom(nrow(extract2$sample), 1, 0.05))
audited2 <- cbind(extract2$sample, audit.value=audited2)
# Evaluate the sample, cache and print it
evaluation2 <- MUS.evaluation(extract2, audited2)

combined <- MUS.combine(list(evaluation1, evaluation2))
MUS.combined.high.error.rate(combined)
```

`MUS.evaluation` *Evaluate a sample using Monetary Unit Sampling.*

Description

Evaluate a sample using Monetary Unit Sampling. At the end of the evaluation step, you get to know the audit conclusion for the population. To conduct the evaluation step it is required that you audited the sample and high values before. You can use print() for a comprehensive output.

Usage

```
MUS.evaluation(extract, filled.sample, filled.high.values,
col.name.audit.values, col.name.riskweights,
interval.type, print.advice, tainting.order,
experimental, combined)
```

Arguments

`extract`	A MUS.extraction.result object that you got by executing the function MUS.extraction.
`filled.sample`	A data frame or matrix with the sample from the extraction routine that have an additional column with the audit values.
`filled.high.values`	A data frame or matrix with the high value items from the extraction routine that have an additional column with the audit values.
`col.name.audit.values`	Single character with the name of the column containing the audit value in filled.sample respectively filled.high.values. Default is "audit.value".
`col.name.riskweights`	Single character with the name of the column containing the risk weights in filled.sample respectively filled.high.values. Default is NULL, then no risk weights are included in the calcualations (the ordinary MUS case).
`interval.type`	Interval type for high error rate evaluation. Default is "one-sided".
`print.advice`	Boolean. Prints recommendations only if TRUE. Default is "TRUE".

`tainting.order` Calculates UEL with different tainting orders (increasing, absolute, random). Default is "decreasing".

`experimental` Boolean. Calculates other bounds, such as momentum, binomial, multinomial. Not ready for production. Default is "FALSE".

`combined` Boolean. Marks the dataset as a combination of multiple strata. Default is "FALSE".

Value

An object MUS.evaluation.result is returned which is a list containing the following elements:

`MUS.extraction.result elements`
All elements that are contained in MUS.extraction.result object. For auditing acceptability and for further steps all inputs are also returned.

`filled.sample` dito.

`filled.high.values`
dito.

`col.name.audit.values`
dito.

`Overstatements.Result.Details`
Detail table for overstatements found in the sample.

`Understatements.Result.Details`
Detail table for understatements found in the sample.

`Results.Sample` Comprehensive results of sample evaluation.

`Results.High.values`
Comprehensive results of individually significant item evaluation.

`Results.Total` Comprehensive results of both evaluations (sample and individual significant items).

`acceptable` Boolean, if population is acceptable given results, confidence level and materiality.

Author(s)

Henning Prömpers <henning@proempers.net>

Examples

```
## Simple Example
# Assume 500 invoices, each between 1 and 1000 monetary units
example.data.1 <- data.frame(book.value=round(runif(n=500, min=1,
max=1000)))
# Plan a sample and cache it
plan.results.simple <- MUS.planning(data=example.data.1,
tolerable.error=100000, expected.error=20000)
# Extract a sample and cache it (no high values exist in this example)
extract.results.simple <- MUS.extraction(plan.results.simple)
# Copy book value into a new column audit values
audited.sample.simple <- extract.results.simple$sample
```

```
audited.sample.simple <- cbind(audited.sample.simple,
audit.value=audited.sample.simple$book.value)
# Edit manually (if any audit difference occur)
#audited.sample.simple <- edit(audited.sample.simple)
# Evaluate the sample, cache and print it
evaluation.results.simple <- MUS.evaluation(extract.results.simple,
audited.sample.simple)
print(evaluation.results.simple)

## Advanced Example
example.data.2 <- data.frame(own.name.of.book.values=round(runif(n=500,
min=1, max=1000)))
plan.results.advanced <- MUS.planning(data=example.data.2,
col.name.book.values="own.name.of.book.values", confidence.level=.70,
tolerable.error=100000, expected.error=20000, n.min=3)
extract.results.advanced <- MUS.extraction(plan.results.advanced,
start.point=5, seed=1, obey.n.as.min=TRUE)
extract.results.advanced <- MUS.extraction(plan.results.advanced)
audited.sample.advanced <- extract.results.advanced$sample
audited.sample.advanced <- cbind(audited.sample.advanced,
own.name.of.audit.values=audited.sample.advanced$own.name.of.book.values)
#audited.sample.advanced <- edit(audited.sample.advanced)
evaluation.results.advanced <- MUS.evaluation(extract.results.advanced,
audited.sample.advanced,
col.name.audit.values="own.name.of.audit.values")
print(evaluation.results.advanced)
```

`MUS.extend` *Extend a MUS sample.*

Description

Extends a sample that requires further evidence. Works with MUS.extraction.result.

Please treat as experimental.

Usage

```
MUS.extend(extract, new_plan=NULL, additional.n=NULL)
```

Arguments

`extract`	An object of the type MUS.extraction.result to be extended.
`new_plan`	Provide a new MUS plan. If null, you must provide the qty of items to extend the sample.
`additional.n`	Ignored if new_plan is provided, otherwise sample will be extended by additional.n items

Value

Returns an extended MUS.extraction.result object.

Author(s)

Andre Guimaraes <alsguimaraes@gmail.com>

Examples

```
## Simple Example
# Assume 500 invoices
mydata <- data.frame(book.value=
  round(c(runif(n=480, min=10,max=20000),
  runif(n=20, min=15000,max=50000)))
)

# Plan a sample and cache it
plan <- MUS.planning(data=mydata,
tolerable.error=50000, expected.error=3000)

# Extract a sample and cache it
extract <- MUS.extraction(plan, obey.n.as.min=TRUE)

# Create a new plan
new_plan <- MUS.planning(data=mydata,
tolerable.error=50000, expected.error=5000)

# extends the sample using the new plan
extended <- MUS.extend(extract, new_plan)

# extends the sample by 20 itens using the original plan
extended20 <- MUS.extend(extract, additional.n=20)
```

`MUS.extraction` *Extract a sample using Monetary Unit Sampling.*

Description

Extract a sample using Monetary Unit Sampling. At the end of the extraction step, you get to know the items that you have to audit.

Usage

```
MUS.extraction(plan, start.point, seed, obey.n.as.min, combined)
```

Arguments

plan	A MUS.planning.result object that you got by executing the function MUS.planning.
start.point	The extraction method uses fixed interval sampling. The monetary unit specified by start.point will be drawn in each interval. Default is NULL, in this case a random number is drawn.
seed	A seed number which will be used to initialise the random number generator. Default is NULL which means that no new random number generator is initialised. This argument is mainly used for simulations or if you want to be able to regenerate the sample on another computer.
obey.n.as.min	Boolean. If set to TRUE, the sample interval will be exactly recalculated and thus the sample size will be exactly the planned sample size. Default is FALSE which is what most commercial statistical software do. In this case the drawn sample size might be slightly smaller than specified.
combined	Boolean. Marks the dataset as a combination of multiple strata. Default is "FALSE".

Value

An object MUS.extraction.result is returned which is a list containing the following elements:

MUS.planning.result elements	All elements that are contained in MUS.planning.result object. For auditing acceptability and for further steps all inputs are also returned.
start.point	dito.
seed	dito.
obey.n.as.min	dito.
high.values	The part of the population that is classified as individually significant items. All of them have to be audited.
sample.population	The part of the population that is not in the high-values-subpopulation.
sampling.interval	The reassessed sampling interval that have to be used for evaluation.
sample	The extracted sample. All elements have to be audited.

Author(s)

Henning Prömpers <henning@proempers.net>

See Also

MUS.planning for planning a sample and MUS.evaluation for evaluation of the extracted and audited sample.

Examples

```
## Simple Example
# Assume 500 invoices, each between 1 and 1000 monetary units
example.data.1 <- data.frame(book.value=round(runif(n=500, min=1,
max=1000)))
# Plan a sample and cache it
plan.results.simple <- MUS.planning(data=example.data.1,
tolerable.error=100000, expected.error=20000)
# Extract a sample and cache it
extract.results.simple <- MUS.extraction(plan.results.simple)

## Advanced Example
example.data.2 <- data.frame(own.name.of.book.values=round(runif(n=500,
min=1, max=1000)))
plan.results.advanced <- MUS.planning(data=example.data.2,
col.name.book.values="own.name.of.book.values", confidence.level=.70,
tolerable.error=100000, expected.error=20000, n.min=3)
extract.results.advanced <- MUS.extraction(plan.results.advanced,
start.point=5, seed=0, obey.n.as.min=TRUE)
```

`MUS.factor` *Calculate MUS Factor.*

Description

Calculate MUS Factor (AICPA, 2012). Based on Technical Notes on the AICPA Audit Guide Audit Sampling, Trevor Stewart, AICPA, 2012.

Usage

```
MUS.factor(confidence.level, pct.ratio)
```

Arguments

`confidence.level`	dito.
`pct.ratio`	Expected.error by tolerable.error.

Value

Returns the MUS factor.

Author(s)

Andre Guimaraes <alsguimaraes@gmail.com>

Examples

```
MUS.factor(0.95, 0.5)
```

`MUS.moment.bound` *Calculate the moment bound for a Monetary Unit Sampling evaluation.*

Description

Calculates the moment bound (Dworkin & Grimlund, 1984) for a Monetary Unit Sampling evaluation.

Please treat as experimental.

Usage

```
MUS.moment.bound(x, confidence.level, as.pct, include.high.values)
```

Arguments

`x`	A MUS.evaluation.result object (or a tainting vector) used to calculate the moment bound.
`confidence.level`	The required confidence level. Default is 95%.
`as.pct`	Boolean. Express results as percentage. Default is False.
`include.high.values`	Boolean. Whether the bound should include high values. Default is "TRUE".

Value

Upper Error Limit calculed using the moment bound.

Author(s)

Andre Guimaraes <alsguimaraes@gmail.com>

See Also

`MUS.evaluation` for evaluation of the audited sample.

Examples

```
sample = c(rep(0, 96), -.16, .04, .18, .47)
MUS.moment.bound(sample)

# Assume 500 invoices, each between 1 and 1000 monetary units
data <- data.frame(book.value=round(runif(n=500, min=1, max=1000)))
# Plan a sample and cache it
plan <- MUS.planning(data=data, tolerable.error=10000, expected.error=2000)
# Extract a sample and cache it (no high values exist in this example)
extract <- MUS.extraction(plan)
# Copy book value into a new column audit values, and inject some error
```

```
audited <- extract$sample$book.value*(1-rbinom(nrow(extract$sample), 1, 0.05))
audited <- cbind(extract$sample, audit.value=audited)
# Evaluate the sample, cache and print it
evaluation <- MUS.evaluation(extract, audited)
MUS.moment.bound(evaluation)
```

`MUS.multinomial.bound` *Calculate a multinomial bound for a Monetary Unit Sampling evaluation.*

Description

Calculates a multinomial bound for a Monetary Unit Sampling evaluation.

Please treat as experimental.

Usage

```
MUS.multinomial.bound(x, as.pct, include.high.values)
```

Arguments

`x`	A MUS.evaluation.result object used to calculate the multinomial bound.
`as.pct`	Boolean. Express results as percentage. Default is False.
`include.high.values`	Boolean. Whether the bound should include high values. Default is "TRUE".

Value

Upper Error Limit calculed using the multinomial bound.

Author(s)

Andre Guimaraes <alsguimaraes@gmail.com>

See Also

`MUS.evaluation` for evaluation of the audited sample.

Examples

```
# Assume 500 invoices, each between 1 and 1000 monetary units
data <- data.frame(book.value=round(runif(n=500, min=1, max=1000)))
# Plan a sample and cache it
plan <- MUS.planning(data=data, tolerable.error=10000, expected.error=2000)
# Extract a sample and cache it (no high values exist in this example)
extract <- MUS.extraction(plan)
# Copy book value into a new column audit values, and inject some error
```

```
audited <- extract$sample$book.value*(1-rbinom(nrow(extract$sample), 1, 0.05))
audited <- cbind(extract$sample, audit.value=audited)
# Evaluate the sample, cache and print it
evaluation <- MUS.evaluation(extract, audited)
MUS.multinomial.bound(evaluation)
```

`MUS.planning` *Plan a sample using Monetary Unit Sampling.*

Description

Plan a sample for Monetary Unit Sampling. At the end of this planning step, you get to know the sample size.

Be aware that this MUS routines cannot calculate with decimals. Furthermore, you must provide book values etc. as Euro-Cent so that no decimals occur.

Usage

```
MUS.planning(data, col.name.book.values, confidence.level,
tolerable.error, expected.error, n.min, errors.as.pct, conservative, combined)
```

Arguments

`data`	A data frame or matrix which contains at least one column with the book values.
`col.name.book.values`	The name of the column that contains the book values. Default is "book.value".
`confidence.level`	The required confidence level. Default is 95%.
`tolerable.error`	The tolerable error (materiality) in Monetary Units.
`expected.error`	The expected error which is contained in the population in Monetary Units.
`n.min`	Minimum sample size that should be used. Default is 0.
`errors.as.pct`	Boolean. Tolerable and Expected error informed as percentages. Default is False.
`conservative`	Boolean. If true, use greater sample size between normal calculation and conservative algorithm (i.e., gamma-based, AICPA compatible).
`combined`	Boolean. Marks the dataset as a combination of multiple strata. Default is "FALSE".

Value

An object MUS.planning.result is returned which is a list containing the following elements:

`data` For auditing acceptability and for further steps all inputs are also returned.

`col.name.book.values` dito.

`confidence.level` dito.

`tolerable.error` dito.

`expected.error` dito.

`book.value` The calculated gross book value of the population. Negative values are ignored.

`n` The calculated sample size based on the input parameters which is greater or egal than the provided minimum sample size.

`High.value.threshold` Whenever a book value of an element is above the threshold, the element will be considered individually significant. Individual significant items will be audited completely, no sample extrapolation will be necessary.

`tolerable.taintings` The number of taintings in the sample that will be acceptable at maximum.

Author(s)

Henning Prömpers <henning@proempers.net>

See Also

`MUS.extraction` for extraction of the planned sample and `MUS.evaluation` for evaluation of the extracted and audited sample.

Examples

```
## Simple Example
# Assume 500 invoices, each between 1 and 1000 monetary units
example.data.1 <- data.frame(book.value=round(runif(n=500, min=1,
max=1000)))
# Plan a sample and cache it
plan.results.simple <- MUS.planning(data=example.data.1,
tolerable.error=100000, expected.error=20000)

## Advanced Example
example.data.2 <- data.frame(own.name.of.book.values=round(runif(n=500,
min=1, max=1000)))
plan.results.advanced <- MUS.planning(data=example.data.2,
col.name.book.values="own.name.of.book.values", confidence.level=.70,
tolerable.error=100000, expected.error=20000, n.min=3)
```

print.MUS.evaluation.result *Pretty and comprehensive printing of MUS evaluation results*

Description

Pretty and comprehensive printing of MUS evaluation results that can be used for working papers.

Usage

```
## S3 method for class 'MUS.evaluation.result'
print(x, error.rate, print.misstatements,
  print.planning, print.extraction, print.error.as.pct, print.advice,
 style, use.pander, ...)
```

Arguments

x	A MUS.evaluation.result object that you got by executing the function MUS.evaluation.
error.rate	Selects type of error rate calculation (i.e., "high", "low", "both" or "auto"). Defaults to "auto".
print.misstatements	Boolean. Should misstatements table be printed? Defaults to TRUE.
print.planning	Boolean. Should planning parameters be printed? Defaults to FALSE.
print.extraction	Boolean. Should extraction parameters be printed? Defaults to FALSE.
print.error.as.pct	Boolean. Should errors as percentage be printed? Defaults to TRUE.
print.advice	Boolean. Should recommendations be printed? Defaults to TRUE.
style	Two options: "report" or "default". Report uses an alternative layout. Defaults to "default".
use.pander	Boolean. Uses pander to generate rmarkdown report. Defaults to FALSE.
...	Further arguments, currently ignored.

Author(s)

Henning Prömpers <henning@proempers.net>

See Also

MUS.evaluation for evaluation of the extracted and audited sample.

print.MUS.extraction.result

Pretty and comprehensive printing of MUS extraction results

Description

Pretty and comprehensive printing of MUS extraction results that can be used for working papers.

Usage

```
## S3 method for class 'MUS.extraction.result'
print(x, print.title,
  print.planning, style, use.pander, ...)
```

Arguments

x	A MUS.evaluation.result object that you got by executing the function MUS.evaluation.
print.title	Boolean. Should title be printed? Defaults to TRUE.
print.planning	Boolean. Should planning parameters be printed? Defaults to FALSE.
style	Two options: "report" or "default". Report uses an alternative layout. Defaults to "default".
use.pander	Boolean. Uses pander to generate rmarkdown report. Defaults to FALSE.
...	Further arguments, currently ignored.

Author(s)

Henning Prömpers <henning@proempers.net>

See Also

MUS.extraction for extraction of the audit sample.

print.MUS.planning.result

Pretty and comprehensive printing of MUS planning results

Description

Pretty and comprehensive printing of MUS planning results that can be used for working papers.

Usage

```
## S3 method for class 'MUS.planning.result'
print(x, print.title,
  style, use.pander, ...)
```

Arguments

x	A MUS.evaluation.result object that you got by executing the function MUS.evaluation.
print.title	Boolean. Should title be printed? Defaults to TRUE.
style	Two options: "report" or "default". Report uses an alternative layout. Defaults to "default".
use.pander	Boolean. Uses pander to generate rmarkdown report. Defaults to FALSE.
...	Further arguments, currently ignored.

Author(s)

Henning Prömpers <henning@proempers.net>

See Also

MUS.planning for planning of the audit sample.

Index