

Package ‘diffpriv’

July 18, 2017

Type Package

Title Easy Differential Privacy

Version 0.4.2

Date 2017-07-16

Description An implementation of major general-purpose mechanisms for privatizing statistics, models, and machine learners, within the framework of differential privacy of Dwork et al. (2006) <doi:10.1007/11681878_14>. Example mechanisms include the Laplace mechanism for releasing numeric aggregates, and the exponential mechanism for releasing set elements. A sensitivity sampler (Rubinstein & Alda, 2017) <arXiv:1706.02562> permits sampling target non-private function sensitivity; combined with the generic mechanisms, it permits turn-key privatization of arbitrary programs.

License MIT + file LICENSE

LazyData TRUE

Depends R (>= 3.4.0)

Imports gsl, methods, stats

URL <https://github.com/brubinstein/diffpriv>,
<http://brubinstein.github.io/diffpriv>

BugReports <https://github.com/brubinstein/diffpriv/issues>

RoxygenNote 6.0.1

VignetteBuilder knitr

Encoding UTF-8

Suggests randomNames, testthat, knitr, rmarkdown

Collate 'utils.R' 'bernstein_polynomials.R' 'privacy_params.R'
'mechanisms.R' 'bernstein_mechanism.R' 'diffpriv.R'
'exponential_mechanism.R' 'numeric_mechanism.R'
'gaussian_mechanism.R' 'laplace_mechanism.R'
'sensitivity_sampler.R'

NeedsCompilation no

Author Benjamin Rubinstein [aut, cre],
 Francesco Aldà [aut]

Maintainer Benjamin Rubinstein <brubinstein@unimelb.edu.au>

Repository CRAN

Date/Publication 2017-07-18 11:42:21 UTC

R topics documented:

bernstein	2
diffpriv	3
DPMech-class	5
DPMechBernstein-class	5
DPMechExponential-class	7
DPMechGaussian-class	8
DPMechLaplace-class	9
DPMechNumeric-class	10
DPParamsDel-class	11
DPParamsEps-class	12
DPParamsGam-class	13
predict.bernstein	14
releaseResponse	15
sensitivityNorm	15
sensitivitySampler	16
sensitivitySampler,DPMech,function,numeric-method	17
sensitivitySamplerManual	18
sensitivitySamplerManual,DPMech,function,numeric,numeric,numeric-method	19
sensitivitySamplerManual,DPMechNumeric,function,numeric,numeric,numeric-method	20
setDelta<-	21
setEpsilon<-	22
setGamma<-	22
Index	23

bernstein	<i>Fit a Bernstein polynomial approximation.</i>
-----------	--

Description

Fits the basis of Bernstein polynomial functions to given real-valued function f of $[0, 1]^d$ where $d = \text{dims}$, against a regular lattice of $k + 1$ points in each dimension for given k . Note the approximation is not the iterated variant.

Usage

```
bernstein(f, dims, k = 10)
```

Arguments

f	a function to be approximated.
dims	the function f's domain dimension.
k	the lattice resolution of approximation.

Value

an S3 object of class `bernstein`.

References

Francesco Aldà and Benjamin I. P. Rubinstein. "The Bernstein Mechanism: Function Release under Differential Privacy", in Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'2017), pp. 1705-1711, Feb 2017.

See Also

[predict.bernstein](#) for subsequent evaluation.

Examples

```
f <- function(x) x * sin(x*10)
b <- bernstein(f, dims = 1)
xs <- seq(from=0, to=1, length=50)
mean((f(xs) - predict(b,xs))^2)
```

diffpriv

diffpriv: *practical differential privacy in R*.

Description

The `diffpriv` package is a collection of generic tools for privacy-aware data science, under the formal framework of differential privacy. A differentially-private mechanism can release responses to untrusted third parties, models fit on privacy-sensitive data. Due to the formal worst-case nature of the framework, however, mechanism development typically requires theoretical analysis. `diffpriv` offers a turn-key approach to differential privacy.

General-purpose mechanisms

Differential privacy's popularity is owed in part to a number of generic mechanisms for privatizing non-private target functions. Virtual S4 class `DPMech-class` captures common features of these mechanisms and is superclass to:

- `DPMechLaplace`: the Laplace mechanism of Dwork et al. (2006) for releasing numeric vectors;

- [DPMechExponential](#): the exponential mechanism of McSherry and Talwar (2007) for releasing solutions to optimizations, over numeric or non-numeric sets; and
- More mechanisms coming soon. Users can also develop new mechanisms by subclassing [DPMech-class](#).

[DPMech-class](#)-derived objects are initialized with a problem-specific non-private target function. Subsequently, the [releaseResponse](#) method can privatize responses of target on input datasets. The level of corresponding privatization depends on given privacy parameters [DPParamsEps](#) or derived parameters object.

Privatize anything with sensitivity measurement

diffpriv mechanisms have in common a reliance on the 'sensitivity' of target function to small changes to input datasets. This sensitivity must be provably bounded for an application's target in order for differential privacy to be proved, and is used to calibrate privacy-preserving randomization. Unfortunately bounding sensitivity is often prohibitively complex, for example if target is an arbitrary computer program. All [DPMech-class](#) mechanisms offer a [sensitivitySampler](#) method due to Rubinstein and Aldà (2017) that repeatedly probes target to estimate sensitivity automatically. Mechanisms with estimated sensitivities achieve a slightly weaker form of random differential privacy due to Hall et al. (2013), but without any theoretical analysis necessary.

References

- Benjamin I. P. Rubinstein and Francesco Aldà. "Pain-Free Random Differential Privacy with Sensitivity Sampling", accepted into the 34th International Conference on Machine Learning (ICML'2017), May 2017.
- Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis." In Theory of Cryptography Conference, pp. 265-284. Springer Berlin Heidelberg, 2006.
- Frank McSherry and Kunal Talwar. "Mechanism design via differential privacy." In the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), pp. 94-103. IEEE, 2007.
- Rob Hall, Alessandro Rinaldo, and Larry Wasserman. "Random Differential Privacy." Journal of Privacy and Confidentiality, 4(2), pp. 43-59, 2012.

Examples

```
## Not run:
## for full examples see the diffpriv vignette
vignette("diffpriv")

## End(Not run)
```

DPMech-class *An S4 class for differentially-private mechanisms.*

Description

A base class for representing output-perturbing mechanisms in differential privacy. As this class is VIRTUAL it cannot be instantiated, but it can be subclassed.

Slots

`sensitivity` non-negative scalar numeric target sensitivity. Defaults to Inf for use with `sensitivitySampler()`.
`target` the target non-private function to be privatized, takes lists. Defaults to a constant function.
`gammaSensitivity` NA_real_ if inactive, or scalar in [0,1) indicating that responses must be RDP with specific confidence.

References

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis." In Theory of Cryptography Conference, pp. 265-284. Springer Berlin Heidelberg, 2006.

See Also

[DPMechLaplace](#) subclass for the Laplace mechanism.

DPMechBernstein-class *An S4 class for the Bernstein mechanism of differential privacy.*

Description

A class that implements the Bernstein mechanism (not iterated version) of differential privacy, for privatizing release of real-valued functions on $[0, 1]^l$ based on arbitrary datasets. Approximates the target on a lattice.

Usage

```
## S4 method for signature 'DPMechBernstein'
show(object)

## S4 method for signature 'DPMechBernstein,DPPParamsEps'
releaseResponse(mechanism,
  privacyParams, X)

## S4 method for signature 'DPMechBernstein'
sensitivityNorm(mechanism, X1, X2)
```

Arguments

object	an instance of class DPMech.
mechanism	an object of class <code>DPMechBernstein</code> .
privacyParams	an object of class <code>DPParamsEps</code> .
X	a privacy-sensitive dataset, if using sensitivity sampler a: list, matrix, data frame, numeric/character vector.
X1	a privacy-sensitive dataset.
X2	a privacy-sensitive dataset.

Value

list with slots per argument, actual privacy parameter and response: mechanism response with length of target release: privacyParams, sensitivity, latticeK, dims, target, response. scalar numeric norm of non-private target on datasets. The L_∞ of the functions on a lattice.

Methods (by generic)

- show: automatically prints the object.
- releaseResponse: releases Bernstein mechanism responses.
- sensitivityNorm: measures target sensitivity.

Slots

sensitivity non-negative scalar numeric maximum absolute target sensitivity maximized over the lattice. Defaults to Inf for use with `sensitivitySampler()`.

target might be a closure that takes arbitrary dataset and returns a real-valued function on $[0, 1]^l$.

gammaSensitivity NA_real_ if inactive, or scalar in $[0, 1)$ indicating that responses must be RDP with specific confidence.

latticeK positive scalar integer-valued numeric specifying the lattice resolution. Defaults to (invalid) NA_integer_.

dims positive scalar integer-valued numeric specifying the dimension of released function domain. Defaults to (invalid) NA_integer_.

References

Francesco Aldà and Benjamin I. P. Rubinstein. "The Bernstein Mechanism: Function Release under Differential Privacy", in Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'2017), pp. 1705-1711, Feb 2017.

Examples

```
## See the bernstein vignette
```

DPMechExponential-class

An S4 class for the exponential mechanism of differential privacy.

Description

A class that implements the exponential mechanism of differential privacy, for privatizing releases from sets (not necessarily numeric as required by [DPMechLaplace](#)). Currently limited to responses from a finite sets - the most widely used case - as these induce easily computed sampling distributions from a uniform base measure.

Usage

```
## S4 method for signature 'DPMechExponential'
show(object)

## S4 method for signature 'DPMechExponential,DPPParamsEps'
releaseResponse(mechanism,
  privacyParams, X)

## S4 method for signature 'DPMechExponential'
sensitivityNorm(mechanism, X1, X2)
```

Arguments

object	an instance of class DPMech.
mechanism	an object of class DPMechExponential .
privacyParams	an object of class DPPParamsEps .
X	a privacy-sensitive dataset, if using sensitivity sampler a: list, matrix, data frame, numeric/character vector.
X1	a privacy-sensitive dataset.
X2	a privacy-sensitive dataset.

Value

list with slots per argument, actual privacy parameter and response: mechanism response with length of target release: privacyParams, sensitivity, responseSet, target, response.
 scalar numeric norm of non-private target on datasets.

Methods (by generic)

- show: automatically prints the object.
- releaseResponse: releases exponential mechanism responses.
- sensitivityNorm: measures target quality score sensitivity.

Slots

`sensitivity` non-negative scalar numeric quality function sensitivity. Defaults to `Inf` for use with `sensitivitySampler()`.

`target` the quality score function mapping dataset to a function on responses (elements of `responseSet`).

`gammaSensitivity` `NA_real_` if inactive, or scalar in $[0,1]$ indicating that responses must be RDP with specific confidence.

`responseSet` a list of possible responses of the mechanism.

References

Frank McSherry and Kunal Talwar. "Mechanism design via differential privacy." In the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), pp. 94-103. IEEE, 2007.

Examples

```
## Sensitive data are strings of length at most 5.
## Task is to release most frequent character present, hence quality function
## is a closure that counts character frequencies for given candidate char.
## Global sensitivity is max string length.
qualF <- function(X) { function(r) sum(r == unlist(strsplit(X, ""))) }
rs <- as.list(letters)
m <- DPMechExponential(sensitivity = 5, target = qualF, responseSet = rs)
X <- strsplit("the quick brown fox jumps over the lazy dog", " ")[[1]]
p <- DPParamsEps(epsilon = 1)
releaseResponse(m, p, X)
```

DPMechGaussian-class *An S4 class for the Gaussian mechanism of differential privacy.*

Description

A class that implements the Gaussian mechanism of differential privacy, for privatizing numeric vector releases.

Usage

```
## S4 method for signature 'DPMechGaussian'
show(object)
```

Arguments

`object` an instance of class `DPMech`.

Methods (by generic)

- `show`: automatically prints the object.

Slots

`sensitivity` non-negative scalar numeric L2 target sensitivity. Defaults to `Inf` for use with `sensitivitySampler()`.

`target` the target non-private function to be privatized, takes lists. Defaults to a constant function. Gaussian mechanism assumes functions that release numeric vectors of fixed dimension `dims`.

`gammaSensitivity` `NA_real_` if inactive, or scalar in $[0,1)$ indicating that responses must be RDP with specific confidence.

`dims` positive scalar numeric dimension of responses. Defaults to `NA_integer_` for use with `sensitivitySampler()` which can probe `target` to determine dimension.

References

Cynthia Dwork and Aaron Roth. "Algorithmic Foundations of Differential Privacy" Foundations and Trends in Theoretical Computer Science. Now Publishers, 2014.

DPMechLaplace-class *An S4 class for the Laplace mechanism of differential privacy.*

Description

A class that implements the basic Laplace mechanism of differential privacy, for privatizing numeric vector releases.

Usage

```
## S4 method for signature 'DPMechLaplace'
show(object)
```

Arguments

`object` an instance of class `DPMech`.

Methods (by generic)

- `show`: automatically prints the object.

Slots

`sensitivity` non-negative scalar numeric L1 target sensitivity. Defaults to `Inf` for use with `sensitivitySampler()`.

`target` the target non-private function to be privatized, takes lists. Defaults to a constant function. Laplace mechanism assumes functions that release numeric vectors of fixed dimension `dims`.

`gammaSensitivity` `NA_real_` if inactive, or scalar in $[0,1)$ indicating that responses must be RDP with specific confidence.

`dims` positive scalar numeric dimension of responses. Defaults to `NA_integer_` for use with `sensitivitySampler()` which can probe `target` to determine dimension.

References

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis." In Theory of Cryptography Conference, pp. 265-284. Springer Berlin Heidelberg, 2006.

DPMechNumeric-class *A virtual S4 class for differentially-private numeric mechanisms.*

Description

A virtual class that implements common features of Laplace, Gaussian mechanisms from differential privacy, for privatizing numeric vector releases.

Usage

```
## S4 method for signature 'DPMechNumeric'
show(object)

## S4 method for signature 'DPMechNumeric'
sensitivityNorm(mechanism, X1, X2)

## S4 method for signature 'DPMechNumeric,DPPParamsEps'
releaseResponse(mechanism, privacyParams,
                X)
```

Arguments

object	an instance of class DPMech.
mechanism	an object of class DPMechNumeric-class.
X1	a privacy-sensitive dataset.
X2	a privacy-sensitive dataset.
privacyParams	an object of class DPPParamsEps .
X	a privacy-sensitive dataset, if using sensitivity sampler a: list, matrix, data frame, numeric/character vector.

Value

scalar numeric norm of non-private target on datasets.
 list with slots per argument, actual privacy parameter; mechanism response with length of target
 release: privacyParams, sensitivity, dims, target, response.

Methods (by generic)

- show: automatically prints the object.
- sensitivityNorm: measures sensitivity of non-private target.
- releaseResponse: releases mechanism responses.

Slots

`sensitivity` non-negative scalar numeric target sensitivity. Defaults to `Inf` for use with `sensitivitySampler()`.

`target` the target non-private function to be privatized, takes lists. Defaults to a constant function.
Laplace mechanism assumes functions that release numeric vectors of fixed dimension `dims`.

`gammaSensitivity` `NA_real_` if deactive, or scalar in `[0,1)` indicating that responses must be RDP with specific confidence.

`dims` positive scalar numeric dimension of responses. Defaults to `NA_integer_` for use with `sensitivitySampler()` which can probe `target` to determine dimension.

Examples

```
f <- function(xs) mean(xs)
n <- 100
m <- DPMechLaplace(sensitivity = 1/n, target = f, dims = 1)
X1 <- runif(n)
X2 <- runif(n)
sensitivityNorm(m, X1, X2)
f <- function(xs) mean(xs)
n <- 100
m <- DPMechLaplace(sensitivity = 1/n, target = f, dims = 1)
X <- runif(n)
p <- DPPParamsEps(epsilon = 1)
releaseResponse(m, p, X)
```

DPPParamsDel-class *An S4 class for relaxed differential privacy parameters.*

Description

An S4 base class representing the privacy parameters in (ϵ, δ) -differential privacy.

Usage

```
## S4 method for signature 'DPPParamsDel'
show(object)

## S4 method for signature 'DPPParamsDel'
getDelta(object)

## S4 replacement method for signature 'DPPParamsDel'
setDelta(object) <- value

## S4 method for signature 'DPPParamsDel,numeric'
toGamma(object, gamma)
```

Arguments

object	an object of class DPPParamsDel .
value	a scalar numeric δ .
gamma	a scalar numeric γ .

Methods (by generic)

- show: automatically prints the object.
- getDelta: getter for slot delta.
- setDelta<-: setter for slot delta.
- toGamma: returns object to corresponding instance of subclass [DPPParamsGam](#).

Slots

epsilon positive scalar numeric privacy level.
 delta a scalar numeric privacy level in interval [0,1).

See Also

[DPPParamsEps](#) superclass, [DPPParamsGam](#) subclass for random relaxation.

DPPParamsEps-class *An S4 class for basic differential privacy parameters.*

Description

An S4 base class representing the basic privacy parameter ϵ in differential privacy.

Usage

```
## S4 method for signature 'DPPParamsEps'
show(object)

## S4 method for signature 'DPPParamsEps'
getEpsilon(object)

## S4 replacement method for signature 'DPPParamsEps'
setEpsilon(object) <- value

## S4 method for signature 'DPPParamsEps,numeric'
toGamma(object, gamma)
```

Arguments

object	an object of class DPParamsEps .
value	a scalar numeric ϵ .
gamma	a scalar numeric γ .

Methods (by generic)

- show: automatically prints the object.
- getEpsilon: getter for slot epsilon.
- setEpsilon<-: setter for slot epsilon.
- toGamma: returns object to corresponding instance of subclass [DPParamsGam](#).

Slots

epsilon positive scalar numeric privacy level.

See Also

[DPParamsDel](#) subclass for (ϵ, δ) relaxation, [DPParamsGam](#) subclass for random relaxation.

DPParamsGam-class *An S4 class for random differential privacy parameters.*

Description

An S4 base class representing the privacy parameters in $(\epsilon, \delta, \gamma)$ -random differential privacy.

Usage

```
## S4 method for signature 'DPParamsGam'
show(object)

## S4 method for signature 'DPParamsGam'
getGamma(object)

## S4 replacement method for signature 'DPParamsGam'
setGamma(object) <- value

## S4 method for signature 'DPParamsGam,numeric'
toGamma(object, gamma)
```

Arguments

object	an object of class DPParamsGam .
value	a scalar numeric γ .
gamma	scalar numeric γ .

Methods (by generic)

- show: automatically prints the object.
- getGamma: getter for slot gamma.
- setGamma<=: setter for slot gamma.
- toGamma: returns object with set gamma; generic for use with superclasses [DPParamsEps](#) and [DPParamsDel](#).

Slots

epsilon positive scalar numeric privacy level.

delta a scalar numeric privacy level in interval [0,1).

gamma a scalar numeric privacy level in [0, 1).

See Also

[DPParamsEps](#), [DPParamsDel](#) superclasses.

predict.bernstein *Evaluate Bernstein approximations on data.*

Description

Evaluates a given S3 object of type bernstein on given data D.

Usage

```
## S3 method for class 'bernstein'
predict(object, D, ...)
```

Arguments

object	an S3 object of type bernstein.
D	either a numeric vector or matrix, all values in $[0, 1]$. If numeric then length should be <code>object\$dim</code> unless the latter is 1 in which case the length can be arbitrary. If a matrix then the number of columns should match <code>object\$dim</code> .
...	additional arguments.

Value

a numeric vector of scalar real evaluations.

References

Francesco Aldà and Benjamin I. P. Rubinstein. "The Bernstein Mechanism: Function Release under Differential Privacy", in Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'2017), pp. 1705-1711, Feb 2017.

Examples

```
f <- function(x) x * sin(x*10)
b <- bernstein(f, dims = 1)
xs <- seq(from=0, to=1, length=50)
mean((f(xs) - predict(b,xs))^2)
```

releaseResponse	DPMech <i>private release method.</i>
-----------------	---------------------------------------

Description

Runs the differentially-private mechanism on given data.

Usage

```
releaseResponse(mechanism, privacyParams, X)
```

Arguments

mechanism	an object of class DPMech-class .
privacyParams	an object of class DPParamsEps or subclass.
X	a privacy-sensitive dataset, if using sensitivity sampler a: list, matrix, data frame, numeric/character vector.

Value

list with slots per argument, including at least: actual privacy parameters `privacyParams`, and response `response`.

sensitivityNorm	DPMech <i>sensitivity-inducing norm.</i>
-----------------	--

Description

Norm of a [DPMech-class](#)'s non-private target function evaluated on two given databases X1, X2.

Usage

```
sensitivityNorm(mechanism, X1, X2)
```

Arguments

mechanism	an object of class DPMech-class .
X1	a privacy-sensitive dataset.
X2	a privacy-sensitive dataset.

sensitivitySampler *Sensitivity sampler for DPMech-class's.*

Description

Given a constructed `DPMech-class`, complete with target function and `sensitivityNorm`, and an oracle for producing records, samples the sensitivity of the target function to set the mechanism's sensitivity.

Usage

```
sensitivitySampler(object, oracle, n, m = NA_integer_, gamma = NA_real_)
```

Arguments

object	an object of class <code>DPMech-class</code> .
oracle	a source of random databases. A function returning: list, matrix/data.frame (data in rows), numeric/character vector of records if given desired length > 1; or single record given length 1, respectively a list element, a row/named row, a single numeric/character. Whichever type is used should be expected by <code>object@target</code> .
n	database size scalar positive numeric, integer-valued.
m	sensitivity sample size scalar positive numeric, integer-valued.
gamma	RDP privacy confidence level.

Value

object with updated `gammaSensitivity` slot.

References

Benjamin I. P. Rubinstein and Francesco Aldà. "Pain-Free Random Differential Privacy with Sensitivity Sampling", accepted into the 34th International Conference on Machine Learning (ICML'2017), May 2017.

Examples

```
## Simple example with unbounded data hence no global sensitivity.
f <- function(xs) mean(xs)
m <- DPMechLaplace(target = f, dims = 1)
m@sensitivity ## Inf
m@gammaSensitivity ## NA as Laplace is naturally eps-DP
P <- function(n) rnorm(n)
m <- sensitivitySampler(m, oracle = P, n = 100, gamma = 0.33)
m@sensitivity ## small like 0.03...
m@gammaSensitivity ## 0.33 as directed, now m is (eps,gam)-DP.
```

sensitivitySampler, DPMech, function, numeric-method
Sensitivity sampler for DPMech-class's.

Description

Given a constructed [DPMech-class](#), complete with target function and sensitivityNorm, and an oracle for producing records, samples the sensitivity of the target function to set the mechanism's sensitivity.

Usage

```
## S4 method for signature 'DPMech`,`function`,`numeric`
sensitivitySampler(object, oracle, n,
  m = NA_integer_, gamma = NA_real_)
```

Arguments

object	an object of class DPMech-class .
oracle	a source of random databases. A function returning: list, matrix/data.frame (data in rows), numeric/character vector of records if given desired length > 1; or single record given length 1, respectively a list element, a row/named row, a single numeric/character. Whichever type is used should be expected by object@target.
n	database size scalar positive numeric, integer-valued.
m	sensitivity sample size scalar positive numeric, integer-valued.
gamma	RDP privacy confidence level.

Value

object with updated gammaSensitivity slot.

References

Benjamin I. P. Rubinstein and Francesco Aldà. "Pain-Free Random Differential Privacy with Sensitivity Sampling", accepted into the 34th International Conference on Machine Learning (ICML'2017), May 2017.

Examples

```
## Simple example with unbounded data hence no global sensitivity.
f <- function(xs) mean(xs)
m <- DPMechLaplace(target = f, dims = 1)
m@sensitivity ## Inf
m@gammaSensitivity ## NA as Laplace is naturally eps-DP
P <- function(n) rnorm(n)
m <- sensitivitySampler(m, oracle = P, n = 100, gamma = 0.33)
```

```
m@sensitivity ## small like 0.03...
m@gammaSensitivity ## 0.33 as directed, now m is (eps,gam)-DP.
```

sensitivitySamplerManual

Sensitivity sampler for [DPMech-class](#).

Description

Given a constructed [DPMech-class](#), complete with target function and sensitivityNorm, and an oracle for producing records, samples the sensitivity of the target function to set the mechanism's sensitivity. Typically the method [sensitivitySampler](#) should be used instead; NOTE this method does not properly set the gammaSensitivity slot of [DPMech-class](#) unlike the preferred method.

Usage

```
sensitivitySamplerManual(object, oracle, n, m, k)
```

Arguments

object	an object of class DPMech-class .
oracle	a source of random databases. A function returning: list, matrix/data.frame (data in rows), numeric/character vector of records if given desired length > 1; or single record given length 1, respectively a list element, a row/named row, a single numeric/character. Whichever type is used should be expected by object@target.
n	database size scalar positive numeric, integer-valued.
m	sensitivity sample size scalar positive numeric, integer-valued.
k	order statistic index in 1,...,m.

Value

object with updated sensitivity parameter.

References

Benjamin I. P. Rubinstein and Francesco Aldà. "Pain-Free Random Differential Privacy with Sensitivity Sampling", accepted into the 34th International Conference on Machine Learning (ICML'2017), May 2017.

See Also

[sensitivitySampler](#) preferred method for sensitivity sampling.

Examples

```
## Simple example with unbounded data hence no global sensitivity.
f <- function(xs) mean(xs)
m <- DPMechLaplace(target = f, dims = 1)
P <- function(n) rnorm(n)
m <- sensitivitySamplerManual(m, oracle = P, n = 100, m = 10, k = 10)
m@sensitivity
```

sensitivitySamplerManual,DPMech,function,numeric,numeric,numeric-method
Sensitivity sampler for DPMech-class.

Description

Given a constructed [DPMech-class](#), complete with target function and sensitivityNorm, and an oracle for producing records, samples the sensitivity of the target function to set the mechanism's sensitivity. Typically the method [sensitivitySampler](#) should be used instead; NOTE this method does not properly set the gammaSensitivity slot of [DPMech-class](#) unlike the preferred method.

Usage

```
## S4 method for signature 'DPMech`,`function`,`numeric`,`numeric`,`numeric`'
sensitivitySamplerManual(object,
  oracle, n, m, k)
```

Arguments

object	an object of class DPMech-class .
oracle	a source of random databases. A function returning: list, matrix/data.frame (data in rows), numeric/character vector of records if given desired length > 1; or single record given length 1, respectively a list element, a row/named row, a single numeric/character. Whichever type is used should be expected by object@target.
n	database size scalar positive numeric, integer-valued.
m	sensitivity sample size scalar positive numeric, integer-valued.
k	order statistic index in 1,...,m.

Value

object with updated sensitivity parameter.

References

Benjamin I. P. Rubinstein and Francesco Aldà. "Pain-Free Random Differential Privacy with Sensitivity Sampling", accepted into the 34th International Conference on Machine Learning (ICML'2017), May 2017.

See Also

[sensitivitySampler](#) preferred method for sensitivity sampling.

Examples

```
## Simple example with unbounded data hence no global sensitivity.
f <- function(xs) mean(xs)
m <- DPMechLaplace(target = f, dims = 1)
P <- function(n) rnorm(n)
m <- sensitivitySamplerManual(m, oracle = P, n = 100, m = 10, k = 10)
m@sensitivity
```

sensitivitySamplerManual,DPMechNumeric,function,numeric,numeric,numeric-method
Sensitivity sampler for DPMechNumeric-class.

Description

Given a constructed [DPMechNumeric-class](#), complete with target function and sensitivityNorm, and an oracle for producing records, samples the sensitivity of the target function to set the mechanism's sensitivity. Typically the method [sensitivitySampler](#) should be used instead; NOTE this method does not properly set the gammaSensitivity slot of [DPMech-class](#) unlike the preferred method. This method can probe target to determine response dimension when the corresponding object@dims is NA.

Usage

```
## S4 method for signature 'DPMechNumeric`,`function`,`numeric,numeric,numeric`
sensitivitySamplerManual(object,
  oracle, n, m, k)
```

Arguments

object	an object of class DPMechNumeric-class .
oracle	a source of random databases. A function returning: list, matrix/data.frame (data in rows), numeric/character vector of records if given desired length > 1; or single record given length 1, respectively a list element, a row/named row, a single numeric/character. Whichever type is used should be expected by object@target.

n database size scalar positive numeric, integer-valued.
 m sensitivity sample size scalar positive numeric, integer-valued.
 k order statistic index in 1,...,m.

Value

object with updated sensitivity parameter, and (potentially) dims.

References

Benjamin I. P. Rubinstein and Francesco Aldà. "Pain-Free Random Differential Privacy with Sensitivity Sampling", accepted into the 34th International Conference on Machine Learning (ICML'2017), May 2017.

See Also

[sensitivitySampler](#) preferred method for sensitivity sampling.

Examples

```
## Simple example with unbounded data hence no global sensitivity.
f <- function(xs) mean(xs)
m <- DPMechLaplace(target = f, dims = 1)
P <- function(n) rnorm(n)
m <- sensitivitySamplerManual(m, oracle = P, n = 100, m = 10, k = 10)
m@sensitivity
```

setDelta<- *Setter for slot delta.*

Description

Use this method instead of slot delta.

Usage

```
setDelta(object) <- value
```

Arguments

object the instance of DPPParamsDel.
 value positive numeric δ value.

See Also

[DPPParamsDel](#).

setEpsilon<- *Setter for slot epsilon.*

Description

Use this method instead of slot epsilon.

Usage

```
setEpsilon(object) <- value
```

Arguments

object	the instance of DPParamsEps.
value	positive numeric ϵ value.

See Also

[DPParamsEps.](#)

setGamma<- *Setter for slot gamma.*

Description

Use this method instead of slot gamma.

Usage

```
setGamma(object) <- value
```

Arguments

object	the instance of DPParamsGam.
value	positive numeric γ value.

See Also

[DPParamsGam.](#)

Index

- bernstein, [2](#)
- diffpriv, [3](#)
- diffpriv-package (diffpriv), [3](#)
- DPMech-class, [5](#), [16–19](#)
- DPMechBernstein, [6](#)
- DPMechBernstein
 - (DPMechBernstein-class), [5](#)
- DPMechBernstein-class, [5](#)
- DPMechExponential, [4](#), [7](#)
- DPMechExponential
 - (DPMechExponential-class), [7](#)
- DPMechExponential-class, [7](#)
- DPMechGaussian (DPMechGaussian-class), [8](#)
- DPMechGaussian-class, [8](#)
- DPMechLaplace, [3](#), [5](#), [7](#)
- DPMechLaplace (DPMechLaplace-class), [9](#)
- DPMechLaplace-class, [9](#)
- DPMechNumeric (DPMechNumeric-class), [10](#)
- DPMechNumeric-class, [10](#), [20](#)
- DPPParamsDel, [12–14](#), [21](#)
- DPPParamsDel (DPPParamsDel-class), [11](#)
- DPPParamsDel-class, [11](#)
- DPPParamsEps, [4](#), [6](#), [7](#), [10](#), [12–15](#), [22](#)
- DPPParamsEps (DPPParamsEps-class), [12](#)
- DPPParamsEps-class, [12](#)
- DPPParamsGam, [12](#), [13](#), [22](#)
- DPPParamsGam (DPPParamsGam-class), [13](#)
- DPPParamsGam-class, [13](#)
- getDelta, DPPParamsDel-method
 - (DPPParamsDel-class), [11](#)
- getEpsilon, DPPParamsEps-method
 - (DPPParamsEps-class), [12](#)
- getGamma, DPPParamsGam-method
 - (DPPParamsGam-class), [13](#)
- predict.bernstein, [3](#), [14](#)
- releaseResponse, [4](#), [15](#)
- releaseResponse, DPMechBernstein, DPPParamsEps-method
 - (DPMechBernstein-class), [5](#)
- releaseResponse, DPMechExponential, DPPParamsEps-method
 - (DPMechExponential-class), [7](#)
- releaseResponse, DPMechNumeric, DPPParamsEps-method
 - (DPMechNumeric-class), [10](#)
- sensitivityNorm, [15](#)
- sensitivityNorm, DPMechBernstein-method
 - (DPMechBernstein-class), [5](#)
- sensitivityNorm, DPMechExponential-method
 - (DPMechExponential-class), [7](#)
- sensitivityNorm, DPMechNumeric-method
 - (DPMechNumeric-class), [10](#)
- sensitivitySampler, [4](#), [16](#), [18–21](#)
- sensitivitySampler, DPMech, function, numeric-method,
 - [17](#)
- sensitivitySamplerManual, [18](#)
- sensitivitySamplerManual, DPMech, function, numeric, numeric-method,
 - [19](#)
- sensitivitySamplerManual, DPMechNumeric, function, numeric, numeric-method,
 - [20](#)
- setDelta<-, [21](#)
- setDelta<-, DPPParamsDel-method
 - (DPPParamsDel-class), [11](#)
- setEpsilon<-, [22](#)
- setEpsilon<-, DPPParamsEps-method
 - (DPPParamsEps-class), [12](#)
- setGamma<-, [22](#)
- setGamma<-, DPPParamsGam-method
 - (DPPParamsGam-class), [13](#)
- show, DPMechBernstein-method
 - (DPMechBernstein-class), [5](#)
- show, DPMechExponential-method
 - (DPMechExponential-class), [7](#)
- show, DPMechGaussian-method
 - (DPMechGaussian-class), [8](#)
- show, DPMechLaplace-method
 - (DPMechLaplace-class), [9](#)

show,DPMechNumeric-method
(DPMechNumeric-class), [10](#)

show,DPPParamsDel-method
(DPPParamsDel-class), [11](#)

show,DPPParamsEps-method
(DPPParamsEps-class), [12](#)

show,DPPParamsGam-method
(DPPParamsGam-class), [13](#)

toGamma,DPPParamsDel,numeric-method
(DPPParamsDel-class), [11](#)

toGamma,DPPParamsEps,numeric-method
(DPPParamsEps-class), [12](#)

toGamma,DPPParamsGam,numeric-method
(DPPParamsGam-class), [13](#)