

# Package: fdp (via r-universe)

June 10, 2026

**Title** f-Differential Privacy and Gaussian Differential Privacy

**Version** 1.0.0

**Description** Constructs and visualises trade-off functions for f-differential privacy (f-DP) as introduced by Dong et al. (2022) <doi:10.1111/rssb.12454>. Supports Gaussian differential privacy, the f-DP generalisation of (epsilon, delta)-differential privacy, and accepts user-specified optimal type I / type II errors from which the lower convex hull trade-off function is automatically constructed.

**URL** <https://fdp.louisaslett.com/>

**BugReports** <https://github.com/louisaslett/fdp/issues>

**License** GPL (>= 3)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Depends** R (>= 4.1.0)

**Imports** cli, ggplot2, rlang

**Repository** <https://louisaslett.r-universe.dev>

**Date/Publication** 2025-11-03 11:59:38 UTC

**RemoteUrl** <https://github.com/louisaslett/fdp>

**RemoteRef** HEAD

**RemoteSha** 65fdd6c157444999caa7d099ac4bd6d4b702f922

## Contents

+.fdp_plot . . . . .	2
epsdelta . . . . .	3
est_epsdelta . . . . .	5
est_gdp . . . . .	7
fdp . . . . .	9

fdp_attributes . . . . .	13
gdp . . . . .	15
gdp_to_epsdelta . . . . .	17
lap . . . . .	19

<b>Index</b>	<b>22</b>
--------------	-----------

---

+.fdp_plot	<i>Combine fdp plots</i>
------------	--------------------------

---

## Description

Allows combining multiple fdp() plot objects using the + operator.

## Usage

```
## S3 method for class 'fdp_plot'
e1 + e2
```

## Arguments

e1	An fdp_plot object (the result of calling fdp())
e2	Either another fdp_plot object or a ggplot2 layer

## Value

If e2 is an fdp\_plot, returns a new combined fdp\_plot object. If e2 is a ggplot2 layer, returns a modified ggplot2 object.

## Examples

```
# Combine two separate fdp() calls
fdp(gdp(0.5)) + fdp(lap(1))

# Can still add regular ggplot2 layers
fdp(gdp(1)) + ggplot2::ggtitle("My Privacy Plot")

# First legend naming takes precedence
fdp(gdp(0.5), .legend = "First") + fdp(lap(1), .legend = "Second")
# Later .legend arguments apply if none specified in prior calls
fdp(gdp(0.5)) + fdp(lap(1), .legend = "Second")
```

epsdelta

*(epsilon, delta)-differential privacy trade-off function***Description**

Constructs the trade-off function corresponding to the classical  $(\epsilon, \delta)$ -differential privacy guarantee. This is the f-DP representation of the approximate differential privacy definition, which allows a small probability  $\delta$  of privacy breach (if  $\delta > 0$ ) while maintaining  $\epsilon$ -differential privacy with probability  $1 - \delta$ .

The resulting trade-off function is piecewise linear with two segments, reflecting the geometry of  $(\epsilon, \delta)$ -DP in the hypothesis testing framework. The function returned can be called either without arguments to retrieve the underlying data points, or with an `alpha` argument to evaluate the trade-off at specific Type-I error rates.

**Usage**

```
epsdelta(epsilon, delta = 0)
```

**Arguments**

<code>epsilon</code>	Numeric scalar specifying the $\epsilon$ privacy parameter. Must be non-negative.
<code>delta</code>	Numeric scalar specifying the $\delta$ privacy parameter. Must be in $[0, 1]$ . Default is <code>0.0</code> (pure $\epsilon$ -DP).

**Details**

Creates an  $(\epsilon, \delta)$ -differential privacy trade-off function for use in f-DP analysis and visualisation. If you would like a reminder of the formal definition of  $(\epsilon, \delta)$ -DP, please see further down this documentation page in the "Formal definition" Section.

The function returns a closure that stores the  $\epsilon$  and  $\delta$  parameters in its environment. This function can be called with or without arguments supplied, either to obtain the skeleton or particular Type-II error rates for given Type-I errors respectively.

**Value**

A function of class `c("fdp_epsdelta_tradeoff", "function")` which computes the  $(\epsilon, \delta)$ -DP trade-off function.

When called:

- **Without arguments:** Returns a data frame with columns `alpha` and `beta` containing the skeleton points of the piecewise linear trade-off function.
- **With an `alpha` argument:** Returns a data frame with columns `alpha` and `beta` containing the Type-II error values corresponding to the specified Type-I error rates.

### Formal definition

Classical  $(\varepsilon, \delta)$ -differential privacy (Dwork et al., 2006a,b) states that a randomised mechanism  $M$  satisfies  $(\varepsilon, \delta)$ -DP if for all neighbouring datasets  $S$  and  $S'$  that differ in a single observation, and any event  $E$ ,

$$\mathbb{P}(M(S) \in E) \leq e^\varepsilon \mathbb{P}[M(S') \in E] + \delta$$

In the f-DP framework (Dong et al., 2022), this corresponds to a specific trade-off function,

$$f_{\varepsilon, \delta}: [0, 1] \rightarrow [0, 1]$$

which maps Type-I error rates  $\alpha$  to the minimum achievable Type-II error rates  $\beta$  when distinguishing between the output distributions  $M(S)$  and  $M(S')$ .

The special case  $\delta = 0$  corresponds to pure  $\varepsilon$ -differential privacy, where the trade-off function has no fixed disclosure risk.

### References

Dong, J., Roth, A. and Su, W.J. (2022). “Gaussian Differential Privacy”. *Journal of the Royal Statistical Society Series B*, **84**(1), 3–37. doi:10.1111/rssb.12454.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I. and Naor, M. (2006a) “Our Data, Ourselves: Privacy Via Distributed Noise Generation”. In: *Advances in Cryptology - EUROCRYPT 2006*, 486–503. doi:10.1007/11761679\_29.

Dwork, C., McSherry, F., Nissim, K. and Smith, A. (2006b) “Calibrating Noise to Sensitivity in Private Data Analysis”. In: *Theory of Cryptography*, 265–284. doi:10.1007/11681878\_14.

### See Also

`fdp()` for plotting trade-off functions, `est_epsdelta()` for finding the choice of  $\varepsilon$  and  $\delta$  that lower bounds a collection of trade-off functions.

Additional trade-off functions can be found in `gdp()` for Gaussian differential privacy, and `lap()` for Laplace differential privacy.

### Examples

```
# Pure epsilon-differential privacy with epsilon = 1
pure_dp <- epsdelta(1.0)
pure_dp
pure_dp() # View the skeleton points

# Approximate DP with epsilon = 1 and delta = 0.01
approx_dp <- epsdelta(1.0, 0.01)
approx_dp

# Evaluate at specific Type-I error rates
approx_dp(c(0.05, 0.1, 0.25, 0.5))

# Plot and compare different (epsilon, delta) configurations
fdp(epsdelta(0.5),
    epsdelta(1.0),
```

```

epsdelta(1.0, 0.01))

# Compare with Gaussian DP
fdp(epsdelta(1.0),
    epsdelta(1.0, 0.01),
    gdp(1.0),
    .legend = "Privacy Mechanism")

```

---

est_epsdelta	<i>(epsilon, delta)-differential privacy parameters lower bounding empirical trade-off points</i>
--------------	---

---

## Description

Estimates the  $(\epsilon, \delta)$ -differential privacy parameters that lower bound a given set of empirical trade-off points. This function uses numerical optimisation to identify the tightest  $(\epsilon, \delta)$ -DP guarantee consistent with observed Type-I/Type-II error trade-offs, holding either  $\epsilon$  or  $\delta$  fixed whilst optimising over the other parameter. **Note:** due to the numerical optimisation involved, this is only an approximation.

## Usage

```
est_epsdelta(x, epsilon = NULL, delta = NULL, dp = 2L)
```

## Arguments

x	<p>One or more f-DP trade-off specifications to be lower bounded. Accepts the same flexible input types as <code>fdp()</code>:</p> <ul style="list-style-type: none"> <li>• A function (user-defined or built-in, e.g., <code>gdp()</code>) that when called with a numeric vector <code>alpha</code> returns a data frame with columns <code>alpha</code> and <code>beta</code>;</li> <li>• A data frame with columns <code>alpha</code> and <code>beta</code> containing empirical trade-off points;</li> <li>• A numeric vector of length 101 (interpreted as <code>beta</code> values on the canonical grid <code>alpha = seq(0, 1, by = 0.01)</code>).</li> </ul> <p>The function extracts all Type-I/Type-II error coordinates and finds the minimal <math>(\epsilon, \delta)</math>-DP parameters lower bounding them.</p>
epsilon	Optional numeric scalar specifying a fixed value of $\epsilon \geq 0$ . If supplied, the function searches for the minimal $\delta \in [0, 1]$ such that the $(\epsilon, \delta)$ -DP trade-off lower bounds <code>x</code> . Exactly one of <code>epsilon</code> or <code>delta</code> must be specified. Default is <code>NULL</code> .
delta	Optional numeric scalar specifying a fixed value of $\delta \in [0, 1]$ . If supplied, the function searches for the minimal $\epsilon \geq 0$ such that the $(\epsilon, \delta)$ -DP trade-off lower bounds <code>x</code> . Exactly one of <code>epsilon</code> or <code>delta</code> must be specified. Default is <code>NULL</code> .
dp	Integer scalar specifying the number of decimal places of precision for the result (with careful rounding employed to ensure the bound holds). Must be a non-negative integer. Default is <code>2L</code> .

## Details

This function numerically solves an inverse problem in the  $f$ -differential privacy framework: given empirical trade-off points  $\{(\alpha_i, \beta_i)\}_{i=1}^n$  characterising the distinguishability between output distributions of a randomised mechanism on neighbouring datasets, find the minimal classical  $(\varepsilon, \delta)$ -DP parameters such that the  $(\varepsilon, \delta)$ -DP trade-off function lower bounds all observed points.

**Warning:** since this is a numerical optimisation on a finite set of trade-off points, there is no mathematical guarantee of correctness. As such, the  $(\varepsilon, \delta)$  found ought best to be viewed as an approximate lower bound on the true values, since there could be intermediate trade-off points that are not supplied which cause the true values to be larger. For example, consider:

```
est_epsdelta(gdp(0.5)( ), delta = 0)
```

This code will return  $\varepsilon = 1.45$ , yet Corollary 1, p.16, in Dong et al. (2022) means the exact answer here is  $(\varepsilon = 1.45, \delta = 0.000544\dots)$  and that indeed there does not in general exist any finite  $\varepsilon$  solution for  $\delta = 0$ .

**Note:** for lower bounding  $\mu$ -Gaussian Differential Privacy one should use `gdp_to_epsdelta()` instead, which employs exact theoretical results from the literature!

This function may be useful for post-hoc privacy auditing, privacy budget allocation, or mechanism comparison.

### Mathematical formulation:

The  $(\varepsilon, \delta)$ -DP trade-off function  $f_{\varepsilon, \delta}: [0, 1] \rightarrow [0, 1]$  is piecewise linear (see `epsdelta()`). This function seeks parameters  $(\varepsilon, \delta)$  such that

$$f_{\varepsilon, \delta}(\alpha_i) \leq \beta_i \quad \text{for all } i = 1, \dots, n$$

whilst minimising either  $\varepsilon$  (if `delta` is fixed) or  $\delta$  (if `epsilon` is fixed).

Exactly one of `epsilon` or `delta` must be specified by the user; the function then searches for the minimal value of the unspecified parameter. The optimisation first verifies whether any solution exists within reasonable bounds ( $\varepsilon < 30$  or  $\delta < 1$ ), then constructs an objective measuring the signed vertical distance between the empirical points and the candidate  $(\varepsilon, \delta)$ -DP curve. A numerical root finder then seeks the parameter value where this crosses zero, with the solution rounded up to the specified decimal precision (`dp`). There are then checks that the rounded bound holds numerically, with incremental adjustment if necessary to guarantee  $f_{\varepsilon, \delta}(\alpha_i) \leq \beta_i$  for all  $i$  within machine precision.

## Value

A  $(\varepsilon, \delta)$ -DP trade-off function object (see `epsdelta()`) of class `c("fdp_epsdelta_tradeoff", "function")`. This represents the tightest  $(\varepsilon, \delta)$ -DP trade-off function that lower bounds the input `x`.

## References

Dong, J., Roth, A. and Su, W.J. (2022). “Gaussian Differential Privacy”. *Journal of the Royal Statistical Society Series B*, **84**(1), 3–37. doi:10.1111/rssb.12454.

**See Also**

`epsdelta()` for constructing  $(\epsilon, \delta)$ -DP trade-off functions with known parameters, `est_gdp()` for the analogous estimation problem in the Gaussian DP framework, `fdp()` for plotting and comparing trade-off functions.

For lower bounding  $\mu$ -Gaussian Differential Privacy, see `gdp_to_epsdelta()` which uses exact theoretical results from the literature.

**Examples**

```
# Estimate epsilon given fixed delta for empirical trade-off points
# Note: unrealistically small set of points, in practice this would be a
#       collection of potentially thousands of points representing multiple
#       trade-off functions, the collection of which should be lower bounded.
empirical <- data.frame(
  alpha = c(0.00, 0.05, 0.10, 0.25, 0.50, 1.00),
  beta = c(1.00, 0.92, 0.85, 0.70, 0.45, 0.00)
)
result <- est_epsdelta(empirical, delta = 0.01)
result # Print the estimated parameters

# Estimate delta given fixed epsilon
result2 <- est_epsdelta(empirical, epsilon = 1.0)
result2

# Visualise the fit
fdp(empirical, result, .legend = "Trade-off")

# Find epsilon bounding a Gaussian DP mechanism with delta = 0.1 and compare
# with the exactly computed values
gdp_mechanism <- gdp(1.1)
approx_dp <- est_epsdelta(gdp_mechanism, delta = 0.1)
dp <- gdp_to_epsdelta(1.1, environment(approx_dp)$epsilon)
fdp(gdp_mechanism, approx_dp, dp,
    .legend = "Mechanism")

# Compare precision levels
result_2dp <- est_epsdelta(empirical, delta = 0.01, dp = 2L)
result_4dp <- est_epsdelta(empirical, delta = 0.01, dp = 4L)
fdp(empirical, result_2dp, result_4dp)
```

---

est\_gdp

*Gaussian differential privacy parameters lower bounding empirical trade-off points*


---

**Description**

Estimates the minimal Gaussian differential privacy (GDP) parameter  $\mu$  that provides a valid lower bound for a collection of empirical or analytically-derived trade-off points. **Note:** due to the numerical optimisation involved, this is only an approximation.

**Usage**

```
est_gdp(x, dp = 2L)
```

**Arguments**

- `x` One or more f-DP trade-off specifications to be lower bounded. Accepts the same flexible input types as `fdp()`:
- A function (user-defined or built-in, e.g., `lap()`) that when called with a numeric vector `alpha` returns a data frame with columns `alpha` and `beta`;
  - A data frame with columns `alpha` and `beta` containing empirical trade-off points;
  - A numeric vector of length 101 (interpreted as `beta` values on the canonical grid `alpha = seq(0, 1, by = 0.01)`).
- The function extracts all Type-I/Type-II error coordinates and finds the minimal  $(\epsilon, \delta)$ -DP parameters lower bounding them.
- `dp` Integer scalar specifying the number of decimal places of precision for the result (with careful rounding employed to ensure the bound holds). Must be a non-negative integer. Default is 2L.

**Details**

Given a set of trade-off points  $\{(\alpha_i, \beta_i)\}_{i=1}^n$  representing Type-I and Type-II error rates, this function numerically solves for the smallest  $\mu \geq 0$  such that the  $\mu$ -GDP trade-off function

$$G_\mu(\alpha) = \Phi(\Phi^{-1}(1 - \alpha) - \mu)$$

satisfies  $G_\mu(\alpha_i) \leq \beta_i$  for all  $i = 1, \dots, n$ , where  $\Phi$  denotes the standard normal cumulative distribution function.

**Warning:** since this is a numerical optimisation on a finite set of trade-off points, there is no mathematical guarantee of correctness. As such, the  $\mu$  found ought best to be viewed as an approximate lower bound on the true values, since there could be intermediate trade-off points that are not supplied which cause the true values to be larger.

This function may be useful for post-hoc privacy auditing, privacy budget allocation, or mechanism comparison.

**Value**

A GDP trade-off function object (see `gdp()`) with class `c("fdp_gdp_tradeoff", "function")`. This represents the tightest  $\mu$ -GDP trade-off function that lower bounds the input `x`.

**References**

Dong, J., Roth, A. and Su, W.J. (2022). “Gaussian Differential Privacy”. *Journal of the Royal Statistical Society Series B*, **84**(1), 3–37. doi:10.1111/rssb.12454.

**See Also**

`gdp()` for constructing GDP trade-off functions with known  $\mu$ , `fdp()` for visualising and comparing trade-off functions, `gdp_to_epsdelta()` for converting from GDP to classical  $(\epsilon, \delta)$ -DP, `est_epsdelta()` for estimating classical DP parameters from trade-off points.

**Examples**

```
# Estimate GDP from manually specified empirical trade-off points
# These could come from empirical measurements or privacy audits
empirical_points <- data.frame(
  alpha = c(0.00, 0.05, 0.10, 0.25, 0.50, 1.00),
  beta  = c(1.00, 0.93, 0.87, 0.72, 0.43, 0.00)
)
result <- est_gdp(empirical_points)
result

# Visualise how well the GDP bound fits the empirical points
fdp(empirical_points, result)

# Find the GDP lower bound for a Laplace mechanism.
lap_mechanism <- lap(1.5)
gdp_bound <- est_gdp(lap_mechanism)
gdp_bound

# Compare the Laplace mechanism with its GDP lower bound
fdp(lap_mechanism, gdp_bound)

# Control precision with the dp parameter
result_1dp <- est_gdp(empirical_points, dp = 1L)
result_3dp <- est_gdp(empirical_points, dp = 3L)
# Higher precision gives tighter bounds
fdp(empirical_points, result_1dp, result_3dp)
```

fdp

*Plot f-differential privacy trade-off functions***Description**

Produce a comparative plot of one or more (analytic or empirical) f-differential privacy trade-off functions.

**Usage**

```
fdp(..., .legend = NULL, .tol = sqrt(.Machine$double.eps))
```

## Arguments

...	<p>One or more f-DP trade-off specifications. Each argument can be a:</p> <ul style="list-style-type: none"> <li>• function (user-defined or built-in, e.g. <code>gdp()</code>, <code>epsdelta()</code>, <code>lap()</code>, etc) that when called with a numeric vector <code>alpha</code> returns a data frame with columns <code>alpha</code> and <code>beta</code>;</li> <li>• data frame with columns <code>alpha</code> and <code>beta</code>;</li> <li>• numeric vector of length equal to the internal <code>alpha</code> grid (interpreted as <code>beta</code>).</li> </ul> <p>Arguments may be named to control legend labels. See Details for full explanation of different ways to pass these arguments.</p>
<code>.legend</code>	Character string giving the legend title. Use NULL (default) for no title.
<code>.tol</code>	<p>Numeric tolerance used when:</p> <ul style="list-style-type: none"> <li>• Validating <math>\beta</math>, <code>beta &lt;= 1 - alpha + .tol</code>.</li> <li>• Checking convexity for objects forced to draw as lines.</li> </ul>

## Details

This is the main plotting function in the package, which produces plots of f-differential privacy (f-DP) trade-off functions in the style shown in the original f-DP paper (Dong et al., 2022). If you would like a reminder of the formal definition of f-DP, please see further down this documentation page in the "Formal definition" Section.

The ... arguments define the trade-off functions to be plotted and can be:

- Built-in analytic trade-off function generators such as `gdp()`, `epsdelta()`, `lap()`.
- User-defined functions defining trade-off functions.
- Data frames containing an `alpha` and `beta` column.
- Numeric vectors interpreted as a sequence of `beta` values over a canonical grid of Type-I error rates `alpha = seq(0, 1, by = 0.01)`.

We cover each of these cases in more detail in the subsequent sub-sections. After that is a discussion of the two main approaches to modifying the legend labels.

### Built-in analytic trade-off function generators:

Most built-in trade-off function generators will take one or more arguments specifying the level of differential privacy, for example, `gdp(0.5)` corresponding to  $\mu = 0.5$ -Gaussian differential privacy.

These function calls can be passed directly, eg `fdp(gdp(0.5))`, and will automatically provide suitable legend names in the plot, including the detail of any argument specification. So the example `fdp(gdp(0.5))` results in a legend label "0.5-GDP".

### User-defined trade-off functions:

Custom trade-off functions should accept a vector of Type-I error values,  $\alpha$ , and return the corresponding vector of Type-II error values,  $\beta$ . In the simplest case, the user defined function will accept a single argument, so in the (unrealistic) perfect privacy setting:

```
my_fdp <- function(a) {
  1 - a
}
```

This can then be plotted by calling `fdp(my_fdp)`.

However, often there will be a need to pass additional arguments. This is supported using the direct calling mechanism, so assume an axis offset is required for the above unrealistic example:

```
my_fdp <- function(a, off) {
  pmax(0, 1 - a - off)
}
```

This is now called by using the dummy variable `alpha` (which need not be defined in your calling environment), `fdp(my_fdp(alpha, 0.1))`, which will produce the trade-off function curve with offset 0.1.

### Data frames:

One need not define a trade-off function explicitly, it can be implicitly defined by giving a set of coordinates  $\{(\alpha_i, \beta_i)\}_{i=1}^n$  in a two-column data frame with columns named `alpha` and `beta`. These coordinates will be linearly interpolated to produce the trade-off function curve. For example

```
my_fdp <- data.frame(alpha = c(0, 0.25, 1), beta = c(1, 0.25, 0))
```

Can be used to produce the f-DP curve corresponding to  $\epsilon \approx 1.09861$ -differential privacy by then calling `fdp(my_fdp)`. Of course, that particular example is more easily produced using the built-in analytic trade-off function generator `epsdelta()` by calling `fdp(epsdelta(1.09861))`.

### Numeric vectors:

Finally, it is possible to simply provide a vector of  $\beta$  values at the grid of  $\alpha$  values that `fdp()` uses internally for plotting — that is, at the values `seq(0.0, 1.0, by = 0.01)`. For example,

```
a <- seq(0.0, 1.0, by = 0.01)
my_fdp <- 1 - a
```

would then produce the (unrealistic) perfect f-DP privacy curve by calling `fdp(my_fdp)`.

### Legend labels:

As discussed above, built-in analytic trade-off function generators will provide automatic legend labels that make sense for their particular trade-off function. In all other cases, the default will be for the legend label to equal the function, data frame, or numeric vector variable name used when calling `fdp()`. Thus, in all the examples above where `my_fdp` was used as the name of the function/data frame/vector the default legend label will be simply "my\_fdp".

This default can be overridden in two ways:

1. by using an argument name. For example, to set the legend label to "hello" in the user-defined function with offset, one would call `fdp(hello = my_fdp(alpha, 0.1))`. This also works with spaces or special characters by using backtick quoted argument names, for example `fdp(`So cool!` = my_fdp(alpha, 0.1))`.
2. by modifying the object passed with `fdp_name()` in advance. See the help file for that function for further details.

**Drawing method and validation:**

By default, built-in and user-defined function arguments will be plotted as a trade-off function curve. This means that they will first be checked to ensure the rendered line is indeed a valid trade-off function: that is, convex, non-increasing and less than  $1 - \alpha$  (however, technically continuity cannot be checked with a finite number of calls to a black-box function). If a problem is detected an error will be thrown. **Note** that due to the finite precision nature of computers, it might be that these validity checks throw a false alarm, in which case you may use the `.tol` argument to increase the tolerance within which these validity checks must pass.

In contrast, data frame/vector arguments are plotted differently depending on their size. If there are at least 100 rows/elements then these will be treated in the same way as built-in and user-defined function arguments, with trade-off function validity checks. However, if there are fewer rows/elements, then these will be treated as merely a collection of points, the only check being that they all lie below the  $\beta = 1 - \alpha$  line. Those points will then be plotted, together with the lower convex hull which corresponds to the lower bounding trade-off function for that collection of points.

This default behaviour of validating and drawing a line versus computing lower convex hull and plotting points can be controlled with the `fdp_point()` and `fdp_line()` functions. See those help files for further details.

A final performance note: all function type arguments are evaluated on a uniform grid `alpha = seq(0, 1, 0.01)`. To use a custom resolution, supply an explicit data frame instead of a function.

**Value**

A `ggplot2` object of class `c("fdp_plot", "gg", "ggplot")` displaying the supplied trade-off functions (and points, if applicable). It can be further modified with additional `ggplot2` layers or combined with other `fdp_plot` objects using `+`.

**Formal definition (Dong et al., 2022)**

For any two probability distributions  $P$  and  $Q$  on the same space, the trade-off function

$$T(P, Q): [0, 1] \rightarrow [0, 1]$$

characterises the optimal relationship between Type I and Type II errors in a hypothesis test distinguishing between them. It is defined as:

$$T(P, Q)(\alpha) = \inf \{ \beta_\phi : \alpha_\phi \leq \alpha \}$$

where the infimum is taken over all measurable rejection rules  $\phi$ . The terms  $\alpha_\phi = \mathbb{E}_P[\phi]$  and  $\beta_\phi = 1 - \mathbb{E}_Q[\phi]$  represent the Type I and Type II errors, respectively.

A function  $f: [0, 1] \rightarrow [0, 1]$  is a trade-off function if and only if it is convex, continuous, non-increasing, and satisfies  $f(x) \leq 1 - x$  for all  $x \in [0, 1]$ .

In the context of differential privacy, we are interested in the distributions of the output of a randomised algorithm when run on two neighbouring datasets (datasets that differ in a single record),  $S$  and  $S'$ . Let  $M$  be a randomised algorithm which has output probability distribution denoted  $M(S)$  when applied to dataset  $S$ . Then, each pair of neighbouring datasets generate a specific trade-off function  $T(M(S), M(S'))$  which characterises how hard it is to distinguish between whether dataset  $S$  or  $S'$  has been used to produce the released output. Considering all possible neighbouring

datasets leads to a family of trade-off functions, the lower bound of which determines the privacy of the randomised algorithm.

More formally, let  $f$  be a trade-off function. A randomised algorithm  $M$  is said to be  $f$ -differentially private (f-DP) if for any pair of neighbouring datasets  $S$  and  $S'$ , the following condition holds:

$$T(M(S), M(S')) \geq f$$

This definition means that the task of distinguishing whether the mechanism was run on dataset  $S$  or its neighbour  $S'$  is at least as difficult as distinguishing between two canonical distributions whose trade-off function is  $f$ .

Therefore, this function is concerned with plotting  $T(P, Q): [0, 1] \rightarrow [0, 1]$  or  $f: [0, 1] \rightarrow [0, 1]$ . That is, plotting a function which returns the smallest type-II error for a specified type-I error rate.

## References

- Andrew, A. M. (1979). “Another efficient algorithm for convex hulls in two dimensions”. *Information Processing Letters*, **9**(5), 216–219. doi:10.1016/00200190(79)900723.
- Dong, J., Roth, A. and Su, W.J. (2022). “Gaussian Differential Privacy”. *Journal of the Royal Statistical Society Series B*, **84**(1), 3–37. doi:10.1111/rssb.12454.

## Examples

```
# Plotting mu=1 Gaussian differential privacy
fdp(gdp(1))

# Plotting the f_(epsilon,delta) curve corresponding to (1, 0.1)-differential privacy
fdp(epsdelta(1, 0.1))

# These can be plotted together for comparison
fdp(gdp(1), epsdelta(1, 0.1))

# The same curves custom labels and a custom legend header
fdp("Gaussian DP" = gdp(1),
    "Classical DP" = epsdelta(1, 0.1),
    .legend = "Methods")

# Alternatively, combine separate fdp() calls using +
fdp(gdp(1)) + fdp(epsdelta(1, 0.1))
```

---

fdp\_attributes

*Control rendering of f-DP trade-off functions*

---

## Description

These functions attach attributes to f-DP objects that control their visualization:

- `fdp_line()` forces the object to be rendered as a continuous trade-off function curve. The function validates that the resulting curve is convex (a requirement for valid trade-off functions). Use this for analytic trade-off functions or when you want to ensure convexity is checked.

- `fdp_point()` forces the object to be rendered as individual Type I/II error coordinates, with the lower convex hull automatically computed and drawn. Use this for empirical estimates or small datasets where individual points should be visible.
- `fdp_name()` sets or retrieves the legend label for the object. When called with `nm`, it sets the label; when called without `nm`, it returns the current label.
- `fdp_attributes()` retrieves all f-DP related attributes attached to an object.

By default, `fdp()` automatically determines the rendering method: data frames or vectors with  $\geq 100$  elements are treated as lines (with convexity validation), while those with  $< 100$  elements are treated as points (with lower hull computation).

### Usage

```
fdp_attributes(x)

fdp_line(x)

fdp_point(x, hide = FALSE)

fdp_name(x, nm)
```

### Arguments

<code>x</code>	An f-DP object (function, data frame, or vector) to which attributes are added or retrieved.
<code>hide</code>	Logical; if TRUE, individual points are not drawn (only their lower convex hull is shown).
<code>nm</code>	Character string specifying the legend label. If missing, returns the current label.

### Details

Functions to control how f-differential privacy trade-off functions and empirical Type I/II error points are rendered by `fdp()`.

### Value

For `fdp_line()`, `fdp_point()`, and `fdp_name()` (when setting): the input object `x` with modified attributes (returned invisibly).

For `fdp_name()` (when getting) and `fdp_attributes()`: the requested attribute value(s) or NULL.

### See Also

`fdp()` for the main plotting function.

**Examples**

```

# Force a small dataset to be drawn as a line (with convexity check)
df <- data.frame(alpha = c(0, 0.5, 1), beta = c(1, 0.4, 0))
fdp(fdp_line(df))

# Draw points but hide them (only show the lower hull)
fdp(fdp_point(df, hide = TRUE))

# Conversely, the following points if interpolated do not define a convex
# trade-off function, so fdp_line would fail
df2 <- data.frame(alpha = c(0, 0.5, 0.51, 1), beta = c(1, 0.4, 0.34, 0))
#fdp(fdp_line(df2)) # Not run, would error
# But the following is ok, since we will compute lower convex hull due to
# small number of points
fdp(df2)

# If you have a large number of points which will not interpolate to give
# convexity, then fdp_point can force that behaviour
df3 <- gdp(0.5)()
df3$beta <- pmin(df3$beta * rnorm(101, 0.95, sd=0.025), 1.0)
#fdp(df3) # Not run, would error
# But wrapping in fdp_point forces plotting points and lower convex hull
fdp(fdp_point(df3))

# Set a custom legend label programmatically, rather than via argument in
# call to fdp ... eg alternative is fdp(`my label` = my_gdp)
my_gdp <- gdp(1)
my_gdp <- fdp_name(my_gdp, "Custom GDP Label")
fdp(my_gdp)

```

gdp

*Gaussian differential privacy trade-off function***Description**

Constructs the trade-off function corresponding to  $\mu$ -Gaussian differential privacy (GDP). This framework, introduced by Dong et al. (2022), provides a natural privacy guarantee for mechanisms based on Gaussian noise, typically offering tighter composition properties and a better privacy-utility trade-off than classical  $(\epsilon, \delta)$ -differential privacy.

**Usage**

```
gdp(mu = 1)
```

**Arguments**

**mu** Numeric scalar specifying the  $\mu$  privacy parameter. Must be non-negative.

## Details

Creates a  $\mu$ -Gaussian differential privacy trade-off function for use in f-DP analysis and visualisation. If you would like a reminder of the formal definition of  $\mu$ -GDP, please see further down this documentation page in the "Formal definition" Section.

The function returns a closure that stores the  $\mu$  parameter in its environment. This function can be called with or without argument supplied, either to obtain points on a canonical grid or particular Type-II error rates for given Type-I errors respectively.

## Value

A function of class `c("fdp_gdp_tradeoff", "function")` that computes the  $\mu$ -GDP trade-off function.

When called:

- **Without arguments:** Returns a data frame with columns `alpha` and `beta` containing points on a canonical grid (`alpha = seq(0, 1, by = 0.01)`) of the trade-off function.
- **With an `alpha` argument:** Returns a data frame with columns `alpha` and `beta` containing the Type-II error values corresponding to the specified Type-I error rates.

## Formal definition

Gaussian differential privacy (Dong et al., 2022) arises as the trade-off function corresponding to distinguishing between two Normal distributions with unit variance and means differing by  $\mu$ . Without loss of generality, the trade-off function is therefore,

$$G_\mu := T(N(0, 1), N(\mu, 1)) \quad \text{for } \mu \geq 0.$$

This leads to,

$$G_\mu(\alpha) = \Phi(\Phi^{-1}(1 - \alpha) - \mu)$$

where  $\Phi$  is the standard Normal cumulative distribution function.

The most natural way to satisfy  $\mu$ -GDP is by adding Gaussian noise to construct the randomised algorithm. Theorem 1 in Dong et al. (2022) identifies the correct variance of that noise for a given sensitivity of the statistic to be released. Let  $\theta(S)$  be the statistic of the data  $S$  which is to be released. Then the *Gaussian mechanism* is defined to be

$$M(S) := \theta(S) + \eta$$

where  $\eta \sim N(0, \Delta(\theta)^2/\mu^2)$  and,

$$\Delta(\theta) := \sup_{S, S'} |\theta(S) - \theta(S')|$$

the supremum being taken over neighbouring data sets. The randomised algorithm  $M(\cdot)$  is then a  $\mu$ -GDP release of  $\theta(S)$ .

More generally, *any* mechanism  $M(\cdot)$  satisfies  $\mu$ -GDP if,

$$T(M(S), M(S')) \geq G_\mu$$

for all neighbouring data sets  $S, S'$ . In particular, one can seek the minimal  $\mu$  for a collection of trade-off functions using `est_gdp()`.

## References

Dong, J., Roth, A. and Su, W.J. (2022). “Gaussian Differential Privacy”. *Journal of the Royal Statistical Society Series B*, **84**(1), 3–37. doi:10.1111/rssb.12454.

## See Also

`fdp()` for plotting trade-off functions, `est_gdp()` for finding the choice of  $\mu$  that lower bounds a collection of trade-off functions.

Additional trade-off functions can be found in `epsdelta()` for classical  $(\epsilon, \delta)$ -differential privacy, and `lap()` for Laplace differential privacy.

## Examples

```
# Gaussian DP with mu = 1
gdp_1 <- gdp(1.0)
gdp_1
gdp_1() # View points on the canonical grid

# Stronger privacy with mu = 0.5
gdp_strong <- gdp(0.5)
gdp_strong

# Evaluate at specific Type-I error rates
gdp_1(c(0.05, 0.1, 0.25, 0.5))

# Plot and compare different mu values
fdp(gdp(0.5),
    gdp(1.0),
    gdp(2.0))

# Compare Gaussian DP with classical (epsilon, delta)-DP
fdp(gdp(1.0),
    epsdelta(1.0),
    epsdelta(1.0, 0.01),
    .legend = "Privacy Mechanism")
```

---

gdp_to_epsdelta	<i>Convert Gaussian differential privacy to classical (epsilon, delta)-differential privacy</i>
-----------------	---

---

## Description

Computes the exact  $(\epsilon, \delta)$ -differential privacy guarantee corresponding to a given  $\mu$ -Gaussian differential privacy (GDP) mechanism for a specified  $\epsilon$  value. This conversion is based on the closed-form relationship established in Corollary 1 (p.16) of Dong et al. (2022), which provides the tightest possible  $\delta$  for any given  $\epsilon$  and  $\mu$ .

**Usage**

```
gdp_to_epsdelta(mu = 0.5, epsilon = 1, dp = NULL)
```

**Arguments**

mu	Numeric scalar specifying the $\mu$ parameter of the Gaussian differential privacy mechanism. Must be non-negative.
epsilon	Numeric scalar specifying the target $\varepsilon$ privacy parameter. Must be non-negative. The function computes the minimal $\delta$ such that $\mu$ -GDP implies $(\varepsilon, \delta)$ -DP.
dp	Optional integer specifying the number of decimal places for rounding the computed $\delta$ value. If provided, $\delta$ is rounded <i>up</i> to ensure the privacy guarantee remains valid. If NULL (default), the exact value is returned without rounding. Must be a positive integer if specified.

**Details**

While GDP provides a complete characterisation of privacy through the trade-off function, classical  $(\varepsilon, \delta)$ -differential privacy remains the most widely recognised privacy definition in both theoretical and applied research. This function enables practitioners to translate GDP guarantees into the more familiar  $(\varepsilon, \delta)$ -DP language.

For a mechanism satisfying  $\mu$ -GDP, the exact  $(\varepsilon, \delta)$ -DP guarantee is given by Corollary 1 of Dong et al. (2022):

$$\delta(\varepsilon, \mu) = \Phi\left(-\frac{\varepsilon}{\mu} + \frac{\mu}{2}\right) - e^{\varepsilon} \Phi\left(-\frac{\varepsilon}{\mu} - \frac{\mu}{2}\right)$$

where  $\Phi$  denotes the cumulative distribution function of the standard Normal distribution. This was a result originally proved in Balle and Wang (2018).

**Value**

A  $(\varepsilon, \delta)$ -DP trade-off function object (see `epsdelta()`) of class `c("fdp_epsdelta_tradeoff", "function")`.

**References**

Balle, B. and Wang, Y-X. (2018). “Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising”. *Proceedings of the 35th International Conference on Machine Learning*, **80**, 394–403. Available at: <https://proceedings.mlr.press/v80/balle18a.html>.

Dong, J., Roth, A. and Su, W.J. (2022). “Gaussian Differential Privacy”. *Journal of the Royal Statistical Society Series B*, **84**(1), 3–37. doi:10.1111/rssb.12454.

**See Also**

`gdp()` for constructing Gaussian differential privacy trade-off functions, `epsdelta()` for directly constructing  $(\varepsilon, \delta)$ -DP trade-off functions, `est_gdp()` for estimating  $\mu$  from empirical trade-off functions, `est_epsdelta()` for estimating  $(\varepsilon, \delta)$  from empirical trade-off functions, `fdp()` for plotting and comparing trade-off functions.

## Examples

```
# Convert mu = 1 GDP to (epsilon, delta)-DP with epsilon = 1
dp_guarantee <- gdp_to_epsdelta(mu = 1.0, epsilon = 1.0)
dp_guarantee

# Round delta to 6 decimal places for reporting
dp_rounded <- gdp_to_epsdelta(mu = 1.0, epsilon = 1.0, dp = 6)
dp_rounded

# Compare the original GDP with its (epsilon, delta)-DP representation
fdp(gdp(1.0),
    gdp_to_epsdelta(mu = 1.0, epsilon = 1.0),
    .legend = "Privacy Mechanism")

# Explore how delta varies with epsilon for a fixed mu
mu_fixed <- 1.0
epsilons <- c(0.1, 0.5, 1.0, 2.0)

res <- fdp(gdp(mu_fixed))
for (eps in epsilons) {
  res <- res+fdp(gdp_to_epsdelta(mu = mu_fixed, epsilon = eps))
}
res
```

---

lap

*Laplace differential privacy trade-off function*


---

## Description

Constructs the trade-off function corresponding to  $\mu$ -Laplace differential privacy. This corresponds to a randomised algorithm based on Laplace (double exponential) noise, which is the canonical mechanism in the original differential privacy framework (Dwork et al., 2006).

## Usage

```
lap(mu = 1)
```

## Arguments

**mu** Numeric scalar specifying the  $\mu$  privacy parameter. Must be non-negative.

## Details

Creates a  $\mu$ -Laplace differential privacy trade-off function for use in f-DP analysis and visualisation. If you would like a reminder of the formal definition of  $\mu$ -Laplace DP, please see further down this documentation page in the "Formal definition" Section.

The function returns a closure that stores the  $\mu$  parameter in its environment. This function can be called with or without argument supplied, either to obtain points on a canonical grid or particular Type-II error rates for given Type-I errors respectively.

**Value**

A function of class `c("fdp_lap_tradeoff", "function")` that computes the  $\mu$ -Laplace DP trade-off function.

When called:

- **Without arguments:** Returns a data frame with columns `alpha` and `beta` containing the skeleton points of the trade-off function.
- **With an `alpha` argument:** Returns a data frame with columns `alpha` and `beta` containing the Type-II error values corresponding to the specified Type-I error rates.

**Formal definition**

Laplace differential privacy arises as the trade-off function corresponding to distinguishing between two Laplace distributions with unit scale parameter and locations differing by  $\mu$ . Without loss of generality, the trade-off function is therefore,

$$L_\mu := T(\text{Lap}(0, 1), \text{Lap}(\mu, 1)) \quad \text{for } \mu \geq 0.$$

The most natural way to satisfy  $\mu$ -Laplace DP is by adding Laplace noise to construct the randomised algorithm. This is the canonical noise mechanism used in classical  $\varepsilon$ -differential privacy. Let  $\theta(S)$  be the statistic of the data  $S$  which is to be released. Then the *Laplace mechanism* is defined to be

$$M(S) := \theta(S) + \eta$$

where  $\eta \sim \text{Lap}(0, \Delta(\theta)/\mu)$  and,

$$\Delta(\theta) := \sup_{S, S'} |\theta(S) - \theta(S')|$$

the supremum being taken over neighbouring data sets. The randomised algorithm  $M(\cdot)$  is then a  $\mu$ -Laplace DP release of  $\theta(S)$ . In the classical regime, this corresponds to the Laplace mechanism which satisfies  $(\varepsilon = \mu)$ -differential privacy (Dwork et al., 2006).

More generally, *any* mechanism  $M(\cdot)$  satisfies  $\mu$ -Laplace DP if,

$$T(M(S), M(S')) \geq L_\mu$$

for all neighbouring data sets  $S, S'$ .

In the  $f$ -differential privacy framework, the canonical noise mechanism is Gaussian (see `gdp()`), but  $\mu$ -Laplace DP does arise as the trade-off function in the limit of the group privacy of  $\varepsilon$ -DP as the group size goes to infinity (see Proposition 7, Dong et al., 2022).

**References**

- Dong, J., Roth, A. and Su, W.J. (2022). “Gaussian Differential Privacy”. *Journal of the Royal Statistical Society Series B*, **84**(1), 3–37. doi:10.1111/rssb.12454.
- Dwork, C., McSherry, F., Nissim, K. and Smith, A. (2006) “Calibrating Noise to Sensitivity in Private Data Analysis”. In: *Theory of Cryptography*, 265–284. doi:10.1007/11681878\_14.

**See Also**

[fdp\(\)](#) for plotting trade-off functions.

Additional trade-off functions can be found in [gdp\(\)](#) for Gaussian differential privacy, and in [epsdelta\(\)](#) for classical  $(\epsilon, \delta)$ -differential privacy.

**Examples**

```
# Laplace DP with mu = 1
lap_1 <- lap(1.0)
lap_1
lap_1() # View points on the canonical grid

# Plot and compare different mu values
fdp(lap(0.5),
    lap(1.0),
    lap(2.0))

# Notice that (epsilon=1)-differential privacy is indeed 1-Laplace DP
# The gap between the lines is the inefficiency in the privacy
# characterisation of classical differential privacy
fdp(lap(1),
    epsdelta(1))

# Compare Laplace DP with Gaussian DP and classical (epsilon, delta)-DP
fdp(lap(1.0),
    gdp(1.0),
    epsdelta(1.0),
    .legend = "Privacy Mechanism")
```

# Index

`+.fdp_plot`, 2

`epsdelta`, 3

`epsdelta()`, 6, 7, 10, 11, 17, 18, 21

`est_epsdelta`, 5

`est_epsdelta()`, 4, 9, 18

`est_gdp`, 7

`est_gdp()`, 7, 16–18

`fdp`, 9

`fdp()`, 4, 5, 7–9, 14, 17, 18, 21

`fdp_attributes`, 13

`fdp_line (fdp_attributes)`, 13

`fdp_line()`, 12

`fdp_name (fdp_attributes)`, 13

`fdp_name()`, 11

`fdp_point (fdp_attributes)`, 13

`fdp_point()`, 12

`gdp`, 15

`gdp()`, 4, 5, 8–10, 18, 20, 21

`gdp_to_epsdelta`, 17

`gdp_to_epsdelta()`, 6, 7, 9

`lap`, 19

`lap()`, 4, 8, 10, 17