

# Package ‘GPCERF’

March 15, 2023

**Title** Gaussian Processes for Estimating Causal Exposure Response Curves

**Version** 0.2.1

**Maintainer** Naeem Khoshnevis <nkhoshnevis@g.harvard.edu>

**Description** Provides a non-parametric Bayesian framework based on Gaussian process priors for estimating causal effects of a continuous exposure and detecting change points in the causal exposure response curves using observational data. Ren, B., Wu, X., Braun, D., Pillai, N., & Dominici, F.(2021). ``Bayesian modeling for exposure response curve via gaussian processes: Causal effects of exposure to air pollution on health outcomes." arXiv preprint <[arXiv:2105.03454](https://arxiv.org/abs/2105.03454)>.

**License** GPL (>= 3)

**Language** en-US

**URL** <https://github.com/NSAPH-Software/GPCERF>

**BugReports** <https://github.com/NSAPH-Software/GPCERF/issues>

**Copyright** Harvard University

**Imports** parallel, xgboost, stats, MASS, spatstat.geom, logger, Rcpp, RcppArmadillo, ggplot2, cowplot, rlang, Rfast, SuperLearner, wCorr

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**Depends** R (>= 3.5.0)

**Suggests** rmarkdown, knitr, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**LinkingTo** RcppArmadillo, Rcpp

**NeedsCompilation** yes

**Author** Naeem Khoshnevis [aut, cre] (<<https://orcid.org/0000-0003-4315-1426>>, HUIT),  
Boyu Ren [aut] (<<https://orcid.org/0000-0002-5300-1184>>, McLean Hospital),

Tanujit Dey [ctb] (<<https://orcid.org/0000-0001-5559-211X>>, HMS),  
 Danielle Braun [aut] (<<https://orcid.org/0000-0002-5177-8598>>, HSPH)

**Repository** CRAN

**Date/Publication** 2023-03-15 12:50:09 UTC

## R topics documented:

GPCERF-package . . . . .	2
compute_rl_deriv_gp . . . . .	3
compute_rl_deriv_nn . . . . .	4
compute_w_corr . . . . .	5
estimate_cerf_gp . . . . .	6
estimate_cerf_nngp . . . . .	8
estimate_gps . . . . .	10
generate_synthetic_data . . . . .	11
get_logger . . . . .	11
plot.cerf_gp . . . . .	12
plot.cerf_nngp . . . . .	12
print.cerf_gp . . . . .	13
print.cerf_nngp . . . . .	13
set_logger . . . . .	14
summary.cerf_gp . . . . .	15
summary.cerf_nngp . . . . .	15

<b>Index</b>	<b>16</b>
--------------	-----------

---

GPCERF-package	<i>The 'GPCERF' package.</i>
----------------	------------------------------

---

## Description

Provides a non-parametric Bayesian framework based on Gaussian process priors for estimating causal effects of a continuous exposure and detecting change points in the causal exposure response curves using observational data.

## Author(s)

Naeem Khoshnevis  
 Boyu Ren  
 Danielle Braun

## References

Ren, B., Wu, X., Braun, D., Pillai, N. and Dominici, F., 2021. Bayesian modeling for exposure response curve via gaussian processes: Causal effects of exposure to air pollution on health outcomes. arXiv preprint arXiv:2105.03454.

---

compute\_rl\_deriv\_gp    *Detect change-point in standard GP*

---

### Description

Calculates the posterior mean of the difference between left- and right-derivatives at an exposure level for the detection of change points.

### Usage

```
compute_rl_deriv_gp(
  w,
  w_obs,
  y_obs,
  gps_m,
  hyperparam,
  kernel_fn = function(x) exp(-x),
  kernel_deriv_fn = function(x) -exp(-x)
)
```

### Arguments

w	A scalar of exposure level of interest.
w_obs	A vector of observed exposure levels of all samples.
y_obs	A vector of observed outcome values of all samples.
gps_m	An S3 gps object including: gps: A data.frame of GPS vectors. - Column 1: GPS - Column 2: Prediction of exposure for covariate of each data sample (e_gps_pred). - Column 3: Standard deviation of e_gps (e_gps_std) used_params: - dnorm_log: TRUE or FLASE
hyperparam	A vector of hyper-parameters in the GP model.
kernel_fn	The covariance function.
kernel_deriv_fn	The partial derivative of the covariance function.

### Value

A numeric value of the posterior mean of the difference between two one-sided derivatives.

### Examples

```
set.seed(847)
data <- generate_synthetic_data(sample_size = 100)
gps_m <- estimate_gps(cov_mt = data[, -(1:2)],
  w_all = data$treat,
  sl_lib = c("SL.xgboost"),
```

```

dnorm_log = FALSE)

wi <- 8.6

val <- compute_rl_deriv_gp(w = wi,
                           w_obs = data$treat,
                           y_obs = data$Y,
                           gps_m = gps_m,
                           hyperparam = c(1,1,2))

```

---

compute\_rl\_deriv\_nn     *Calculate right minus left derivatives for change-point detection in nnGP*

---

### Description

Calculates the posterior mean of the difference between left- and right-derivatives at an exposure level for the detection of change points. nnGP approximation is used.

### Usage

```

compute_rl_deriv_nn(
  w,
  w_obs,
  gps_m,
  y_obs,
  hyperparam,
  n_neighbor,
  block_size,
  kernel_fn = function(x) exp(-x),
  kernel_deriv_fn = function(x) -exp(-x)
)

```

### Arguments

w	A scalar of exposure level of interest.
w_obs	A vector of observed exposure levels of all samples.
gps_m	An S3 gps object including: gps: A data.frame of GPS vectors. - Column 1: GPS - Column 2: Prediction of exposure for covariate of each data sample (e_gps_pred). - Column 3: Standard deviation of e_gps (e_gps_std) used_params: - dnorm_log: TRUE or FLASE
y_obs	A vector of observed outcome values.
hyperparam	A vector of hyper-parameters in the GP model.
n_neighbor	The number of nearest neighbors on one side.

block_size	The number of samples included in a computation block. Mainly used to balance the speed and memory requirement. Larger block_size is faster, but requires more memory.
kernel_fn	The covariance function. The input is the square of Euclidean distance.
kernel_deriv_fn	The partial derivative of the covariance function. The input is the square of Euclidean distance.

**Value**

A numeric value of the posterior mean of the difference between two one-sided derivatives.

**Examples**

```
set.seed(325)
data <- generate_synthetic_data(sample_size = 200)
gps_m <- estimate_gps(cov_mt = data[, -(1:2)],
                     w_all = data$treat,
                     sl_lib = c("SL.xgboost"),
                     dnorm_log = FALSE)

wi <- 12.2

deriv_val <- compute_rl_deriv_nn(w = wi,
                                w_obs = data$treat,
                                gps_m = gps_m,
                                y_obs = data$Y,
                                hyperparam = c(0.2, 0.4, 1.2),
                                n_neighbor = 20,
                                block_size = 10)
```

---

compute_w_corr	<i>Compute weighted covariate balance</i>
----------------	---

---

**Description**

Computes weighted covariate balance for given data sets.

**Usage**

```
compute_w_corr(w, covariate, weight)
```

**Arguments**

w	A vector of observed continuous exposure variable.
covariate	A data frame of observed covariates variable.
weight	A vector of weights.

**Value**

The function returns a list saved the measure related to covariate balance `absolute_corr`: the absolute correlations for each pre-exposure covairates; `mean_absolute_corr`: the average absolute correlations for all pre-exposure covairates.

**Examples**

```
set.seed(639)
n <- 100
mydata <- generate_synthetic_data(sample_size=100)
year <- sample(x=c("2001", "2002", "2003", "2004", "2005"), size = n,
  replace = TRUE)
region <- sample(x=c("North", "South", "East", "West"), size = n,
  replace = TRUE)
mydata$year <- as.factor(year)
mydata$region <- as.factor(region)
mydata$cf5 <- as.factor(mydata$cf5)
cor_val <- compute_w_corr(mydata[,2],
  mydata[, 3:length(mydata)],
  runif(n))

print(cor_val$mean_absolute_corr)
```

---

estimate_cerf_gp	<i>Estimate the conditional exposure response function using Gaussian process</i>
------------------	---

---

**Description**

Estimates the conditional exposure response function (cerf) using Gaussian Process (gp). The function tune the best match (the lowest covariate balance) for the provided set of hyperparameters.

**Usage**

```
estimate_cerf_gp(
  data,
  w,
  gps_m,
  params,
  nthread = 1,
  kernel_fn = function(x) exp(-x^2)
)
```

**Arguments**

data	A data.frame of observation data. <ul style="list-style-type: none"> <li>• Column 1: Outcome (Y)</li> </ul>
------	---



```

        beta=0.2,
        g_sigma = 1,
        tune_app = "all"),
nthread = 1)

```

---

estimate_cerf_nngp	<i>Estimate the conditional exposure response function using nearest neighbor Gaussian process</i>
--------------------	--

---

### Description

Estimates the conditional exposure response function (cerf) using the nearest neighbor (nn) Gaussian Process (gp). The function tune the best match (the lowest covariate balance) for the provided set of hyperparameters.

### Usage

```

estimate_cerf_nngp(
  data,
  w,
  gps_m,
  params,
  kernel_fn = function(x) exp(-x^2),
  nthread = 1
)

```

### Arguments

data	<p>A data.frame of observation data.</p> <ul style="list-style-type: none"> <li>• Column 1: Outcome (Y)</li> <li>• Column 2: Exposure or treatment (w)</li> <li>• Column 3~m: Confounders (C)</li> </ul>
w	A vector of exposure level to compute CERF.
gps_m	<p>An S3 gps object including: gps: A data.frame of GPS vectors. - Column 1: GPS - Column 2: Prediction of exposure for covariate of each data sample (e_gps_pred). - Column 3: Standard deviation of e_gps (e_gps_std) used_params: - dnorm_log: TRUE or FLASE</p>
params	<p>A list of parameters that is required to run the process. These parameters include:</p> <ul style="list-style-type: none"> <li>• alpha: A scaling factor for the GPS value.</li> <li>• beta: A scaling factor for the exposure value.</li> <li>• g_sigma: A scaling factor for kernel function (gamma/sigma).</li> <li>• tune_app: A tuning approach. Available approaches:</li> </ul>



- all: try all combinations of hyperparameters.
- n\_neighbor: Number of nearest neighbors on one side.
- block\_size: Number of samples included in a computation block. Mainly used to balance the speed and memory requirement. Larger block\_size is faster, but requires more memory. alpha, beta, and g\_sigma can be a vector of parameters.

kernel\_fn      A kernel function. A default value is a Gaussian Kernel.

nthread        An integer value that represents the number of threads to be used by internal packages.

## Value

A cerf\_nngp object that includes the following values:

- w, the vector of exposure levels.
- pst\_mean, the computed mean for the w vector.
- pst\_sd, the computed credible interval for the w vector.

## Examples

```
set.seed(19)
data <- generate_synthetic_data(sample_size = 120, gps_spec = 3)
# Estimate GPS function
gps_m <- estimate_gps(cov_mt = data[,-(1:2)],
                     w_all = data$treat,
                     sl_lib = c("SL.xgboost"),
                     dnorm_log = FALSE)
# exposure values
w.all <- seq(0,20,2)
cerf_nngp_obj <- estimate_cerf_nngp(data,
                                   w.all,
                                   gps_m,
                                   params = list(alpha = c(0.1),
                                                beta = 0.2,
                                                g_sigma = 1,
                                                tune_app = "all",
                                                n_neighbor = 20,
                                                block_size = 1e4),
                                   nthread = 1)
```

---

`estimate_gps`*Estimate a model for generalized propensity score*

---

**Description**

Estimates a model for generalized propensity score (GPS) using parametric approach.

**Usage**

```
estimate_gps(cov_mt, w_all, sl_lib, dnorm_log)
```

**Arguments**

<code>cov_mt</code>	A covariate matrix containing all covariates. Each row is a data sample and each column is a covariate.
<code>w_all</code>	A vector of observed exposure levels.
<code>sl_lib</code>	A vector of SuperLearner's package libraries.
<code>dnorm_log</code>	Logical, if TRUE, probabilities $p$ are given as $\log(p)$ .

**Value**

A data.frame that includes:

- a vector of estimated GPS at the observed exposure levels;
- a vector of estimated conditional means of exposure levels when the covariates are fixed at the observed values;
- estimated standard deviation of exposure levels
- a vector of observed exposure levels.

**Examples**

```
data <- generate_synthetic_data(sample_size = 200)
gps_m <- estimate_gps(cov_mt = data[, -(1:2)],
                     w_all = data$treat,
                     sl_lib = c("SL.xgboost"),
                     dnorm_log = FALSE)
```

---

`generate_synthetic_data`*Generate synthetic data for the GPCERF package*

---

**Description**

Generates synthetic data set based on different GPS models and covariates.

**Usage**

```
generate_synthetic_data(  
  sample_size = 1000,  
  outcome_sd = 10,  
  gps_spec = 1,  
  cova_spec = 1  
)
```

**Arguments**

<code>sample_size</code>	A number of data samples.
<code>outcome_sd</code>	Standard deviation used to generate the outcome in the synthetic data set.
<code>gps_spec</code>	A numeric value (1-6) that indicates the GPS model used to generate the continuous exposure.
<code>cova_spec</code>	A numeric value (1-2) to modify the covariates.

**Value**

A data frame of the synthetic data. Outcome is labeled as Y, exposure as w, and covariates cf1-6.

**Examples**

```
set.seed(351)  
data <- generate_synthetic_data(sample_size = 200)
```

---

`get_logger`*Get logger settings*

---

**Description**

Returns current logger settings.

**Usage**

```
get_logger()
```

**Value**

Returns a list that includes **logger\_file\_path** and **logger\_level**.

**Examples**

```
set_logger("mylogger.log", "INFO")
log_meta <- get_logger()
```

---

plot.cerf_gp	<i>Extend generic plot functions for cerf_gp class</i>
--------------	--

---

**Description**

A wrapper function to extend generic plot functions for cerf\_gp class.

**Usage**

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

**Arguments**

x	A cerf_gp object.
...	Additional arguments passed to customize the plot.

**Value**

Returns a ggplot2 object, invisibly. This function is called for side effects.

---

plot.cerf_nngp	<i>Extend generic plot functions for cerf_nngp class</i>
----------------	--

---

**Description**

A wrapper function to extend generic plot functions for cerf\_nngp class.

**Usage**

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

**Arguments**

x                    A cerf\_nngp object.  
 ...                  Additional arguments passed to customize the plot.

**Value**

Returns a ggplot2 object, invisibly. This function is called for side effects.

---

print.cerf\_gp                    *Extend print function for cerf\_gp object*

---

**Description**

Extend print function for cerf\_gp object

**Usage**

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

**Arguments**

x                    A cerf\_gp object.  
 ...                  Additional arguments passed to customize the results.

**Value**

No return value. This function is called for side effects.

---

print.cerf\_nngp                    *Extend print function for cerf\_nngp object*

---

**Description**

Extend print function for cerf\_nngp object

**Usage**

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

**Arguments**

x                    A cerf\_nngp object.  
 ...                  Additional arguments passed to customize the results.

**Value**

No return value. This function is called for side effects.

---

set_logger	<i>Set logger settings</i>
------------	----------------------------

---

**Description**

Updates logger settings, including log level and location of the file.

**Usage**

```
set_logger(logger_file_path = "GPCERF.log", logger_level = "INFO")
```

**Arguments**

`logger_file_path` A path (including file name) to log the messages. (Default: GPCERF.log)

`logger_level` The log level. Available levels include:

- TRACE
- DEBUG
- INFO (Default)
- SUCCESS
- WARN
- ERROR
- FATAL

**Value**

No return value. This function is called for side effects.

**Examples**

```
set_logger("mylogger.log", "INFO")
```

---

summary.cerf\_gp      *print summary of cerf\_gp object*

---

**Description**

print summary of cerf\_gp object

**Usage**

```
## S3 method for class 'cerf_gp'  
summary(object, ...)
```

**Arguments**

object            A cerf\_gp object.  
...                Additional arguments passed to customize the results.

**Value**

Returns summary of data

---

summary.cerf\_nngp      *print summary of cerf\_nngp object*

---

**Description**

print summary of cerf\_nngp object

**Usage**

```
## S3 method for class 'cerf_nngp'  
summary(object, ...)
```

**Arguments**

object            A cerf\_nngp object.  
...                Additional arguments passed to customize the results.

**Value**

Returns summary of data.

# Index

`compute_rl_deriv_gp`, 3  
`compute_rl_deriv_nn`, 4  
`compute_w_corr`, 5

`estimate_cerf_gp`, 6  
`estimate_cerf_nngp`, 8  
`estimate_gps`, 10

`generate_synthetic_data`, 11  
`get_logger`, 11  
GPCERF (GPCERF-package), 2  
GPCERF-package, 2

`plot.cerf_gp`, 12  
`plot.cerf_nngp`, 12  
`print.cerf_gp`, 13  
`print.cerf_nngp`, 13

`set_logger`, 14  
`summary.cerf_gp`, 15  
`summary.cerf_nngp`, 15