The Journal of Open Source Software

GPCERF - An R package for implementing Gaussian processes for estimating causal exposure response curves

Naeem Khoshnevis[1¶], **Boyu Ren**[2], **and Danielle Braun**[3]

1 Research Computing, Harvard University, Cambridge, Massachusetts, United States of America **2** McLean Hospital, Belmont, Massachusetts, United States of America **3** Department of Biostatistics, Harvard School of Public Health, Cambridge, Massachusetts, United States of America ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Susan Holmes

Reviewers:

- @nhejazi
- @martinmodrak
- @andrewherren

Submitted: 23 March 2023
Published: unpublished

Summary

We present the GPCERF R package, which employs a novel Bayesian approach based on Gaussian Process (GP) to estimate the causal exposure-response function (CERF) for continuous exposures, along with associated uncertainties. R packages that target causal effects under a binary exposure setting exist (e.g., Ho et al., 2011), as well as in the continuous exposure setting (e.g., Khoshnevis, Wu, et al., 2023). However, they often rely on a separate resampling stage to quantify uncertainty of the estimates. GPCERF provides a two-step end-to-end solution for causal inference with continuous exposures that is equipped with automatic and efficient uncertainty quantification. During the first step (the design phase), the algorithm searches for optimal hyperparameters (using the exposures and covariates) that achieve optimal covariate balance in the induced pseudo-population, i.e., that the correlation between the exposure and each covariate is close to zero. The selected hyperparameters are then used in the second step (the analysis phase) to estimate the CERF on the balanced data set and its associated uncertainty using two different types of GPs: a standard GP and a nearest-neighbor GP (nnGP). The standard GP offers high accuracy in estimating CERF but is also computationally intensive. The nnGP is a computationally efficient approximation of the standard GP and is well-suited for the analysis of large-scale datasets.

Statement of need

In the GPCERF R package we have introduced a novel Bayesian approach. This method utilizes Gaussian Processes (GPs) as a prior for counterfactual outcome surfaces, offering a flexible way to estimate the CERF with automatic uncertainty quantification. Additionally, it can incorporate prior information about the level of smoothness of the underlying causal ERF through specifically designed covariance functions. Popular R packages for estimating causal ERF, such as CausalGPS (Khoshnevis, Wu, et al., 2023), ipw (van der Wal & Geskus, 2011), npcausal (Kennedy, 2020) and CBPS (Fong et al., 2022), are primarily built on frequentist frameworks. To the best of the authors' knowledge, however, Bayesian nonparametric alternatives are relatively scarce. causaldrf (Galagate & Schafer, 2022) uses Bayesian Additive Regression Trees (BART) for flexible causal ERF estimation. The resulting CERF is usually not smooth due to the use of regression trees. Yet, it can achieve high accuracy when the underlying ERF is discontinuous or resembles step functions. BCEE (Talbot et al., 2023) applies a Bayesian model averaging approach for causal ERF estimation. bkmr (Bobb, 2022) employs a kernel-based Bayesian model, which is equivalent to a GP prior, to estimate the effect of multivariate exposure on the outcome of interest. However, since it does not explicitly address confounding in the observational data, the resulting estimate does not have causal interpretation.

While various R packages, like GauPro (Erickson, 2023), mlegp (Dancik, 2022), and GPfit (MacDoanld et al., 2019), offer Gaussian process regression capabilities, we chose not to use them. The primary reason is that these packages rely on traditional techniques for hyper parameter tuning, such as sampling from the hyper-parameters' posterior distributions or maximizing the marginal likelihood function. Our approach, in contrast, aims to achieve optimal covariate balancing. By utilizing the posterior distributions of model parameters, we can automatically assess the uncertainty in our CERF estimates (for further details, see Ren et al., 2021). Since standard GPs are infamous for their scalability issues—particularly due to operations involving the inversion of covariance matrices—we adopt a nearest-neighbor GP (nnGP) prior to ensure computationally efficient inference of the CERF in large-scale datasets. Refer to Figure 3 and Figure 4 for comparisons of the wall clock time between standard GP and nnGP.

Overview

In the context of causal inference for continuous exposures, one of the important targets for inference is the so-called casual exposure response function (CERF), which is defined as the expectation of the counterfactual outcomes over the observed covariates at a range of exposure levels in a given population. If we denote the counterfactual outcome at exposure level w by $Y(w)$, CERF is indeed the function $R(w) = \mathbb{E}(Y(w))$ defined on a set of w of interest. One should be careful when estimating CERF from observational data, which usually contain not only the outcome Y and exposure W, but also potential confounders C. The main reason is that if we do not adjust for the confounders C properly, this may lead to biased estimation of $R(w)$. We choose to follow the approach in Hirano & Imbens (2004), which is based on the generalized propensity score (GPS) to adjust for confounding. GPS, denoted by $s(W, C)$, is defined as the conditional density of W given C. It has been shown that one can obtain an unbiased estimator of the causal effect of W provided the conditional distribution of Y given W and $s(W, C)$ is known (Hirano & Imbens, 2004).

In the GPCERF package, we use a Gaussian process (GP) prior for the conditional distribution of Y given W and $s(W, C)$. This model implicitly performs non-parametric regression of Y on W and $s(W, C)$, and thus we can recover $p(Y|W, s(W, C))$ with high accuracy. We then estimate $E(Y(w))$ using the posterior means of Y at different W and C. See Ren et al. (2021) for more details. We assume that the kernel function of the GP is

$$k((w,c),(w',c')) = \gamma^2 h(\sqrt{\frac{(s(w,c)-(s(w',c'))^2}{\alpha} + \frac{(w-w')^2}{\beta}}),$$

where $h : [0, \infty) \to [0, 1]$ is a non-increasing function; and α and β define the relative importance of GPS and exposure values, respectively. γ indicates the scale of the GP. We call the collection $(h, \alpha, \beta, \gamma)$ the hyper-parameters of the GP.

The primary goal in GPCERF is to find appropriate values for the hyper-parameters. In the context of causal inference, ''appropriate'' values of the hyper-parameters are those that make the estimator of CERF as if it is generated from a study with randomized design. To be more concrete, note that the GP estimates $R(w)$ by creating a pseudo-population that is a weighted version of the original dataset (see more details in Ren et al., 2021). The weight for each sample in the original dataset is a function of the hyperparameters. By tuning the hyperparameters, we can minimize the sample correlations between W and each component of C in this pseudo-population, rendering the pseudo-population to be more balanced on these covariates C. In practice, we minimize the covariate balance, which is a summary of the sample correlations between W and each of C to tune our hyper-parameters. Covariate balance is computed by assessing the correlation between W and C in the pseudo-population using the *wCorr* R package (Bailey & Emad, 2023).

Both GP and nnGP approaches involve two primary steps - tuning and estimation. GPCERF conducts a grid search on the range of provided α, β, and γ/σ. The kernel function is also selected if the user provides multiple candidates. During the tuning step, covariate balance is minimized by choosing the optimal hyperparameters.

The scaling parameter α and β determine how much information the estimation will draw from the two coordinates: GPS score ($s(W, X)$) and exposure level (W). A large scaling parameter suggests that varying the corresponding coordinates is only associated with a minor change in the outcome, that is, this coordinate does not contribute too much to the variation of the outcome. The signal-to-noise ratio parameter γ/σ encodes how different observed data is from pure noise. A large γ/σ indicates strong associations between the outcome and the coordinates of GP while a small γ/σ suggests the observed outcome is likely to be drawn from a random process that is independent of the coordinates. In the setting of observational studies and under the no unobserved confounding assumption, which GPCERF is specifically designed for, both the exposure level and the GPS score encode important information for the estimation of CERF. As a result, the range of their scaling parameter should be comparable and the covariate balance will determine which coordinate is more important (smaller scaling factor). The range should also cover both ends of the importance from extremely important to nearly irrelevant. We choose to achieve this by considering the range on the log_{10} scale with equally spaced candidate values. The range of also follows the same strategy when the prior belief about the strength of causal effect of the exposure is weak.

In the estimation step, the optimal parameters are used to estimate the posterior mean and standard deviation of $R(w)$ at a set of exposure values of interest. The outcome data is not used during the tuning step, separating the design and analysis phases. Ren et al. (2021) discusses the implemented approaches in detail. In the following we provide an example for running the package for each implemented models.

Example 1: Standard GP models

To compute the causal exposure response function, one can use the `etimate_cerf_gp()` function. In this example, we generated a synthetic dataset of 500 observations and six covariates. We considered the estimation of $R(w)$ for w that are between 5- and 95-percentiles of the observed exposure levels. We imposed this restriction to make sure that the positivity assumption, which is required for the identifiability of $R(w)$ from the observed data, is not likely to be violated (Ren et al., 2021). We then developed a wrapper function to modify the number of threads in the SuperLearner package (Polley et al., 2021). We estimated the GPS values using these wrapper functions. One can read more details by running `?GPCERF::estimate_gps` in R. To compute the posterior mean and standard deviation of $R(w)$, we need to provide the range of exposure values of interest and a range of hyperparameters that will be examined as additional input and parameters. The function outputs an `S3` object, which can be further investigated using generic functions, as shown below.

```
library(GPCERF)

set.seed(781)
# Generate synthetic data with 500 data samples.
sim_data <- generate_synthetic_data(sample_size = 500,
                                    gps_spec = 1)

# SuperLearner internal libraries' wrapper. (Optional)

m_xgboost <- function(nthread = 12, ...) {
  SuperLearner::SL.xgboost(nthread = nthread, ...)
```

```r
}
m_ranger <- function(num.threads = 12, ...){
  SuperLearner::SL.ranger(num.threads = num.threads, ...)
}

# Estimate GPS function
gps_m <- estimate_gps(cov_mt = sim_data[, paste0("cf", seq(1,6))],
                      w_all = sim_data$treat,
                      sl_lib = c("m_xgboost", "m_ranger"),
                      dnorm_log = TRUE)

# exposure values of interest
# We trim the exposure level to satisfy positivity assumption to avoid including
# extreme exposure values.
q1 <- stats::quantile(sim_data$treat, 0.05)
q2 <- stats::quantile(sim_data$treat, 0.95)
w_all <- seq(q1, q2, 1)

# Hyperparameters' range for grid search to find optimal hyperparameters
params_lst <- list(alpha = 10 ^ seq(-2, 2, length.out = 10),
                   beta = 10 ^ seq(-2, 2, length.out = 10),
                   g_sigma = c(0.1, 1, 10),
                   tune_app = "all")

# Estimate exposure response function
cerf_gp_obj <- estimate_cerf_gp(sim_data,
                                w_all,
                                gps_m,
                                params = params_lst,
                                nthread = 12)
```

The customized summary function provides the following:

```r
summary(cerf_gp_obj)
```

```
GPCERF standard Gaussian grocess exposure response function object

Optimal hyper parameters(#trial: 300):
  alpha = 12.9154966501488   beta = 12.9154966501488   g_sigma = 0.1

Optimal covariate balance:
  cf1 = 0.072
  cf2 = 0.082
  cf3 = 0.062
  cf4 = 0.068
  cf5 = 0.056
  cf6 = 0.082

Original covariate balance:
  cf1 = 0.222
  cf2 = 0.112
  cf3 = 0.175
  cf4 = 0.318
  cf5 = 0.198
  cf6 = 0.257
            ----***----
```

As one can see, as part of the grid search, 300 different combination of hyper parameters have been tried. Figure 1 shows the causal exposure response function and achieved covariate balance in this simulated example.

```
plot(cerf_gp_obj)
```

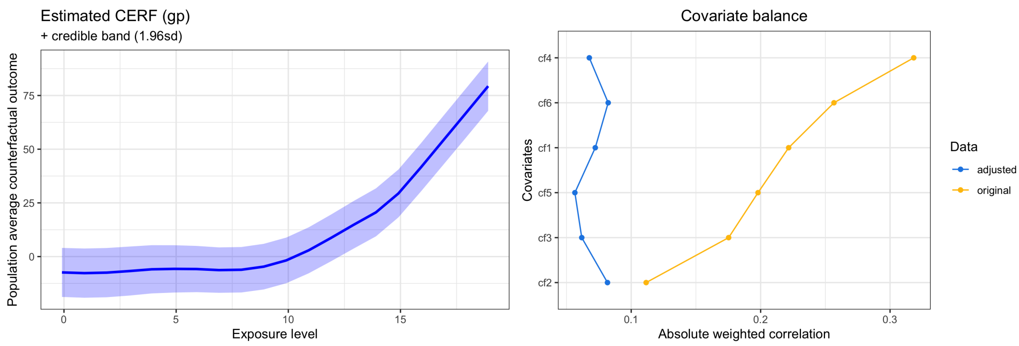

Figure 1: Plot of GP models S3 object. Left: Estimated CERF with credible band. Right: Covariate balance of confounders before and after weighting with GP approach.

The discussion on acceptable covariate balance for causal inference analyses is not within the scope of this paper. However, in the literature, a mean covariate balance upper limit of 0.1 is generally considered acceptable (Wu et al., 2020). It is possible to expand the hyperparameters' search domain to achieve a lower covariate balance.

Example 2: Nearest neighbor GP models

As previously mentioned, GP models are limited in scalability. To address this limitation, the `estimate_cerf_nngp()` function can be used to implement nearest neighbor GP models. While most of the parameters for this model are similar to those used in the GP model, there are two additional hyperparameters specific to the nnGP model that we will discuss in the following.

```
set.seed(781)
# Generate synthetic data with 500 data samples.
sim_data <- generate_synthetic_data(sample_size = 5000, gps_spec = 1)

# SuperLearner internal libraries' wrapper.
m_xgboost <- function(nthread = 12, ...) {
  SuperLearner::SL.xgboost(nthread = nthread, ...)
}

m_ranger <- function(num.threads = 12, ...){
  SuperLearner::SL.ranger(num.threads = num.threads, ...)
}

# Estimate GPS function
gps_m <- estimate_gps(cov_mt = sim_data[, paste0("cf", seq(1,6))],
                      w_all = sim_data$treat,
                      sl_lib = c("m_xgboost", "m_ranger"),
                      dnorm_log = TRUE)

# exposure values of interest
# We trim the exposure level to satisfy positivity assumption to avoid including
# extreme exposure values.
q1 <- stats::quantile(sim_data$treat, 0.05)
```

```r
q2 <- stats::quantile(sim_data$treat, 0.95)

w_all <- seq(q1, q2, 1)

# Hyperparameters' range for grid search to find optimal hyperparameters
params_lst <- list(alpha = 10 ^ seq(-2, 2, length.out = 10),
                   beta = 10 ^ seq(-2, 2, length.out = 10),
                   g_sigma = c(0.1, 1, 10),
                   tune_app = "all",
                   n_neighbor = 50,
                   block_size = 1e3)

# Estimate exposure response function
cerf_nngp_obj <- estimate_cerf_nngp(sim_data,
                                    w_all,
                                    gps_m,
                                    params = params_lst,
                                    nthread = 12)
```

The nearest neighbor GP model contains two controlling parameters: `n_neighbor`, which indicates the size of the neighbor set, and `block_size`, which determines the size of the computational chunks. The choice of `block_size` is primarily used to balance the trade-off between speed and memory requirements, where a larger `block_size` leads to faster computation but also requires more memory. It is worth noting that changing `n_neighbor` may lead to different outcomes due to its approximate nature. However, the outcome values remain unaffected by changes to `block_size`, which serves as an internal optimization parameter.

The customized summary function provides the following:

```r
summary(cerf_nngp_obj)
```

```
GPCERF nearest neighbor Gaussian process exposure response function object summary

Optimal hyper parameters(#trial: 300):
  alpha = 0.0278255940220712   beta = 0.215443469003188   g_sigma = 0.1

Optimal covariate balance:
  cf1 = 0.058
  cf2 = 0.071
  cf3 = 0.087
  cf4 = 0.066
  cf5 = 0.076
  cf6 = 0.088

Original covariate balance:
  cf1 = 0.115
  cf2 = 0.137
  cf3 = 0.145
  cf4 = 0.296
  cf5 = 0.208
  cf6 = 0.225
            ----***----
```

Figure 2 shows the result of `plot(cerf_nngp_obj)` function.

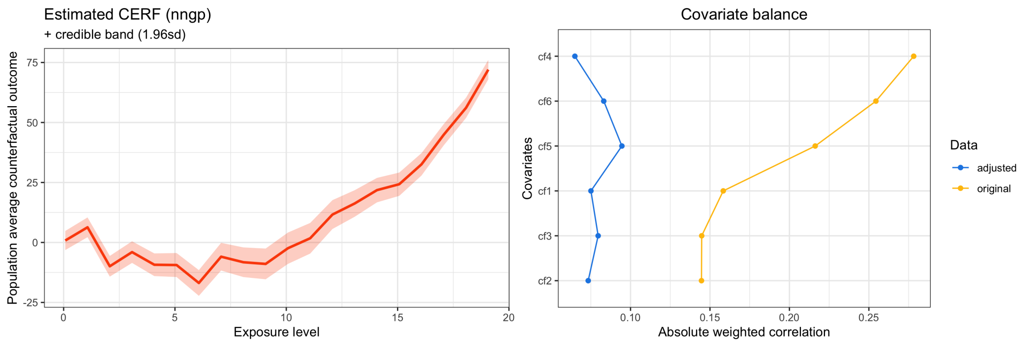

Figure 2: Plot of nnGP models S3 object. Left: Estimated CERF with credible band. Right: Covariate balance of confounders before and after weighting with nnGP approach.

Performance analyses of standard and nearest neighbor GP models

The time complexity of the standard Gaussian Process (GP) model is $O(n^3)$, while for the nearest neighbor GP (nnGP) model, it is $O(n * m^3)$, where m is the number of neighbors. An in-depth discussion on achieving these complexities is outside the scope of this paper. Readers interested in further details can refer to Ren et al. (2021). This section focuses on comparing the wall clock time of standard GP and nnGP models in calculating the Conditional Exposure Response Function (CERF) at a specific exposure level, w. We set the hyper-parameters to values at $\alpha = \beta = \gamma/\sigma = 1$. Figure 3 shows the comparison of standard GP model with nnGP utilizing 50 nearest neighbors. Due to the differing parallelization architectures of the standard GP and nnGP in our package, we conducted this benchmark on a single core. The sample size was varied from 3,000 to 10,000, a range where nnGP begins to demonstrate notable efficiency over the standard GP. We repeat the process 20 times with different seed values. We plotted wall clock time against sample size for both methods. To enhance the visualization of the increasing rate of wall clock time, we applied a log transformation to both axes. For this specific set of analyses the estimated slope of 3.09 (ideally 3) for standard GP aligns with its $O(n^3)$ time complexity. According to the results, a sample size of 10,000 data samples is not large enough to establish a meaningful relationship for the time complexity of the nnGP model effectively.

The Journal of Open Source Software

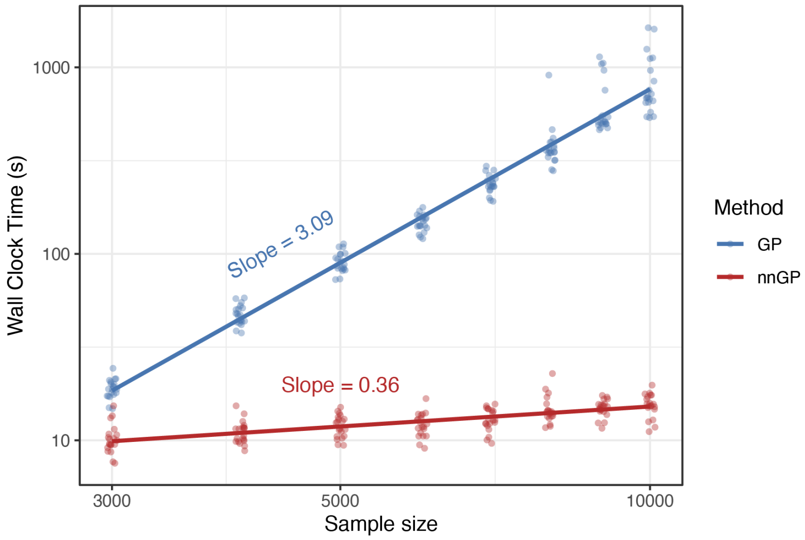

Figure 3: Representation of Wall Clock Time (s) vs. Data Samples for Standard GP and nnGP Models. All computations are conducted with $w = 1$ and $\alpha = \beta = \gamma/\sigma = 1$. The process is repeated 20 times using various seed values to ensure robustness. A jitter effect is applied to enhance the visibility of data points. Both axes are displayed on log10 scales. The solid lines represent the linear regression modeled as $lm(log10(WC) \sim log10(n))$.

Figure 4 compares the performance of the nnGP model across three nearest neighbor categories: 50, 100, and 200, using a data sample sequence ranging from 5,000 to 100,000 with intervals of 5,000. For each category, different sets of runs demonstrate a linear relationship, consistent with an $O(n)$ time complexity, assuming that m^3 remains constant for varying sample sizes within each category.

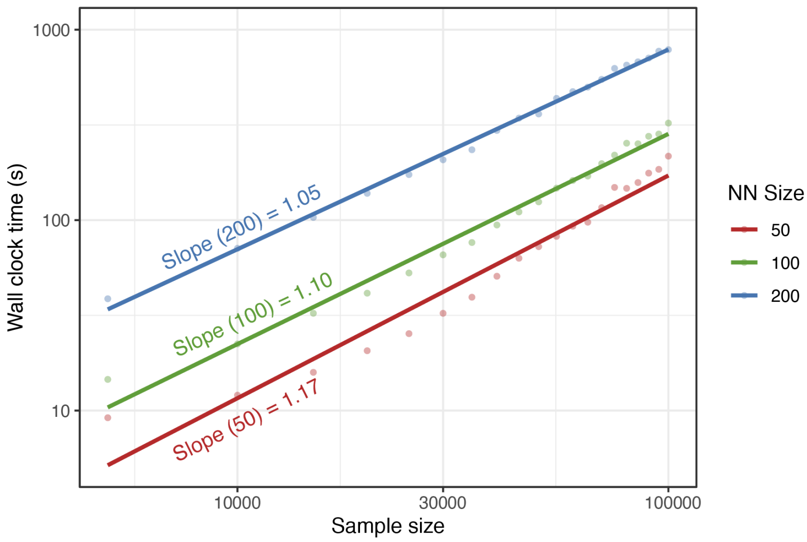

Figure 4: Representation of Wall Clock Time (s) vs. Data Samples of the nnGP model across different nearest neighbor categories (50, 100, 200) over a range of data sample sizes from 5,000 to 100,000 in 5,000 increments. . All computations are conducted with $w = 1$ and $\alpha = \beta = \gamma/\sigma = 1$. Both axes are displayed on log10 scales. The solid lines represent the linear regression modeled as $lm(log10(WC) \sim log10(n))$.

The Journal of Open Source Software

Software related features

We have implemented several features to enhance the package performance and usability. By utilizing an internal `parallel` package, the software is capable of scaling up in a shared memory system. Additionally, we have implemented a logging infrastructure that tracks the software's internal progress and provides users and developers with detailed information on processed runs (Daróczi, 2021). We have also activated continuous integration (CI) through GitHub actions, which runs unit tests and checks the code quality for any submitted pull request. The majority of the codebase is tested at least once. To ensure efficient development, we follow a successful git branching model (Driessen, 2010) and use the tidyverse styling guide. The software is available on CRAN (Khoshnevis, Ren, et al., 2023) and is primarily written in R (R Core Team, 2022). However, some of the core computations are written in C++ using the Rcpp package (Eddelbuettel, 2013; Eddelbuettel & Balamuta, 2018; Eddelbuettel & François, 2011).

Acknowledgement

This work was partially funded by the following grants: NIH: R01s R01ES028033, R01ES030616, R01AG066793, R01ES029950, RF1AG071024, RF1AG074372, R01MD016054, R01ES034373, R01ES028033-03S1, R01AG066793-02S1, and Sloan Foundation: G-2020-13946.

References

Bailey, P., & Emad, A. (2023). *wCorr: Weighted correlations*. https://CRAN.R-project.org/package=wCorr

Bobb, J. F. (2022). *Bkmr: Bayesian kernel machine regression*. https://CRAN.R-project.org/package=bkmr

Dancik, G. M. (2022). *Mlegp: Maximum likelihood estimates of gaussian processes*. https://CRAN.R-project.org/package=mlegp

Daróczi, G. (2021). *Logger: A lightweight, modern and flexible logging utility*. https://CRAN.R-project.org/package=logger

Driessen, V. (2010). *A successful git branching model*. https://nvie.com/posts/a-successful-git-branching-model/.

Eddelbuettel, D. (2013). *Seamless R and C++ integration with Rcpp*. Springer. https://doi.org/10.1007/978-1-4614-6868-4

Eddelbuettel, D., & Balamuta, J. J. (2018). Extending R with C++: A Brief Introduction to Rcpp. *The American Statistician*, *72*(1), 28–36. https://doi.org/10.1080/00031305.2017.1375990

Eddelbuettel, D., & François, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*(8), 1–18. https://doi.org/10.18637/jss.v040.i08

Erickson, C. (2023). *GauPro: Gaussian process fitting*. https://CRAN.R-project.org/package=GauPro

Fong, C., Ratkovic, M., & Imai, K. (2022). *CBPS: Covariate balancing propensity score*. https://CRAN.R-project.org/package=CBPS

Galagate, D., & Schafer, J. (2022). *Causaldrf: Estimating causal dose response functions*. https://CRAN.R-project.org/package=causaldrf

Hirano, K., & Imbens, G. W. (2004). The propensity score with continuous treatments. *Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives*, *226164*, 73–84.

Ho, D. E., Imai, K., King, G., & Stuart, E. A. (2011). MatchIt: Nonparametric preprocessing for parametric causal inference. *Journal of Statistical Software*, *42*(8), 1–28. https://doi.org/10.18637/jss.v042.i08

Kennedy, E. (2020). *npcausal [R package]* (Version 0.1.0). https://github.com/ehkennedy/npcausal

Khoshnevis, N., Ren, B., & Braun, D. (2023). *GPCERF: Gaussian processes for estimating causal exposure response curves*. https://CRAN.R-project.org/package=GPCERF

Khoshnevis, N., Wu, X., & Braun, D. (2023). *CausalGPS: An R package for causal inference with continuous exposures*. https://arxiv.org/abs/2310.00561

MacDoanld, B., Chipman, H., & Ranjan, P. (2019). *GPfit: Gaussian processes modeling*. https://CRAN.R-project.org/package=GPfit

Polley, E., LeDell, E., Kennedy, C., & van der Laan, M. (2021). *SuperLearner: Super learner prediction*. https://CRAN.R-project.org/package=SuperLearner

R Core Team. (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

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*.

Talbot, D., Lefebvre, G., Atherton, J., & Chiu, Y. (2023). *BCEE: The bayesian causal effect estimation algorithm*. https://CRAN.R-project.org/package=BCEE

van der Wal, W. M., & Geskus, R. B. (2011). ipw: An R package for inverse probability weighting. *Journal of Statistical Software*, *43*(13), 1–23. https://doi.org/10.18637/jss.v043.i13

Wu, X., Braun, D., Schwartz, J., Kioumourtzoglou, M.-A., & Dominici, F. (2020). Evaluating the impact of long-term exposure to fine particulate matter on mortality among the elderly. *Science Advances*, *6*(29), eaba5692. https://doi.org/10.1126/sciadv.aba5692