`genriesz`: A Python Package for Automatic Debiased Machine Learning with Generalized Riesz Regression

Masahiro Kato*

Data Analytics Department, Mizuho-DL Financial Technology, Co., Ltd.

February 19, 2026

Abstract

Efficient estimation of causal and structural parameters can be automated via the Riesz representation theorem and debiased machine learning (DML). We present `genriesz`, an open-source Python package implementing automatic DML and *generalized Riesz regression*, a unified framework for estimating Riesz representers by minimizing empirical Bregman divergences. This framework includes covariate balancing, nearest-neighbor matching, and calibrated estimation as special cases. A key design principle of the package is *automatic regressor balancing* (ARB): given a Bregman generator g and a representer model class, `genriesz` automatically constructs a compatible link function so that the generalized Riesz regression estimator satisfies balancing (moment-matching) optimality conditions in a user-chosen basis. The package provides a modular interface for specifying (i) the target linear functional through a black-box evaluation oracle, (ii) the representer model through basis functions (polynomial, RKHS approximations, random forest leaf encodings, neural embeddings, and a nearest-neighbor catchment basis), and (iii) the Bregman generator with optional user-supplied derivatives. It returns regression adjustment (RA), Riesz weighting (RW), augmented Riesz weighting (ARW), and TMLE-style estimators with cross-fitting, Wald-type confidence intervals, and p-values. We highlight representative workflows for ATE, ATT, average marginal effects, panel difference-in-differences, and nearest-neighbor matching. The Python package is available at `https://github.com/MasaKat0/genriesz` and on PyPI.

1 Introduction

Many targets in causal inference and econometrics can be expressed as linear functionals of an unknown regression function. Prominent examples include the average treatment effect (ATE), the average treatment effect on the treated (ATT), average marginal effects (AME),

*Email: `mkato-csecon@g.ecc.u-tokyo.ac.jp`

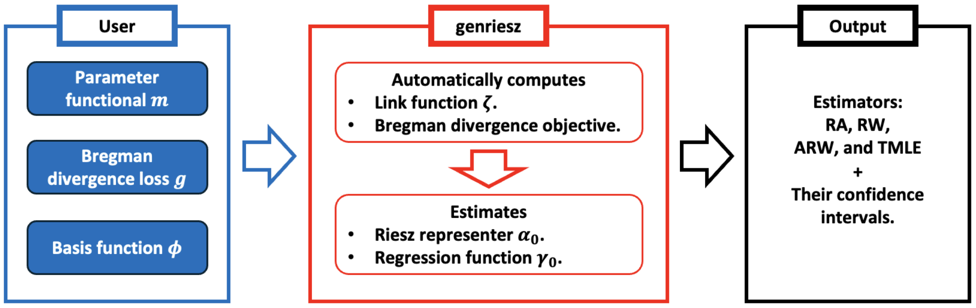
User
Parameter functional m
Bregman divergence loss g
Basis function ϕ
genriesz
Automatically computes
• Link function ζ.
• Bregman divergence objective.
Estimates
• Riesz representer α_0.
• Regression function γ_0.
Output
Estimators: RA, RW, ARW, and TMLE + Their confidence intervals.

Figure 1: The flowchart of `genriesz`

and panel difference-in-differences (DID). Automatic debiased machine learning (ADML) provides general tools for valid inference on such targets (Chernozhukov et al., 2022b).

The ADML workflow separates the problem into two parts. First, one estimates nuisance components, typically a regression function and a Riesz representer. Second, one plugs these estimates into a Neyman orthogonal score and averages the resulting score over the sample. Cross-fitting then yields valid inference under weak conditions, even when the nuisance components are estimated by flexible machine learning methods (Chernozhukov et al., 2018).

The `genriesz` package operationalizes a unifying perspective, *Riesz representer fitting under Bregman divergences* (Kato, 2026). Users specify the estimand through an evaluation oracle and choose a representer model and a Bregman generator. The package then constructs a generator-induced link function to deliver automatic regressor balancing, estimates the representer by convex optimization with ℓ_p regularization, and reports RA, RW, ARW, and TMLE-style estimates with confidence intervals, optionally using cross-fitting.

The flowchart of this automatic procedure is below, where each object will be defined in the subsequent sections (Figure 1):

(i) the user specifies the parameter functional $m(W, \gamma_0)$ for the parameter of interest $\theta_0 := \mathbb{E}[m(W, \gamma_0)]$, the Bregman generator g for Riesz representer estimation, and the basis functions $\phi(X)$.

(ii) the package automatically computes the link function $\zeta\left(X, \phi(X)^\top \beta\right)$ that yields regressor balancing, and estimates the Riesz representer α_0 and, when needed, the regression function γ_0.

(iii) the package outputs RA, RW, ARW, and TMLE-style estimators with standard errors and confidence intervals.

During this process, the user does not need to specify the analytic form of the Riesz representer α_0 or the link function ζ, they are constructed from the chosen basis and generator.

Relation to existing software. The `genriesz` package complements established DML libraries such as DoubleML (Bach et al., 2022) and EconML (Battocchi et al., 2019), as well as broader causal inference toolkits such as DoWhy (Sharma & Kiciman, 2020) and CausalML (Chen et al., 2020). These libraries offer rich sets of estimators for canonical causal models and heterogeneous treatment effects. In contrast, `genriesz` is *estimand-centric*, the user provides the functional m and the representer model, and the package constructs generalized Riesz representers and corresponding estimators. This focus also makes explicit the connection between representer fitting and classical balancing weights, including stable weights (Zubizarreta, 2015) and entropy balancing (Hainmueller, 2012), as well as density ratio estimation methods that are available in specialized packages such as `densratio` (Makiyama, 2019).

2 Key Ingredients of Generalized Riesz Regression

Generalized Riesz regression connects Riesz representer estimation, covariate balancing, and debiased estimation through Bregman divergence minimization. This section summarizes the methodological and theoretical foundations needed to understand the software design. Full technical details, proofs, and extensions are provided in Kato (2026).

2.1 Linear Functionals, Riesz Representers, and Orthogonal Scores

Let $W := (X, Y) \sim P$, where $X \in \mathbb{R}^d$ is a regressor and $Y \in \mathbb{R}$ is an outcome. Let $\gamma_0(x) := \mathbb{E}\left[Y \mid X = x\right]$. We consider targets of the form

$$\theta_0 := \mathbb{E}\left[m\left(W, \gamma_0\right)\right], \tag{1}$$

where $m\left(W, \gamma\right)$ is linear in γ. In `genriesz`, users supply m as a callable that can evaluate γ at modified inputs, for example, by switching a treatment component.

Under standard conditions, there exists a Riesz representer α_0 such that

$$\mathbb{E}\left[m\left(W, \gamma\right)\right] = \mathbb{E}\left[\alpha_0(X)\gamma(X)\right] \qquad \text{for all suitable } \gamma. \tag{2}$$

The Riesz representer comprises the Neyman orthogonal score

$$\psi\left(W; \theta, \gamma, \alpha\right) := m\left(W, \gamma\right) + \alpha(X)\left(Y - \gamma(X)\right) - \theta, \tag{3}$$

which plays an important role in debiased estimation and valid inference under weak conditions and cross-fitting (Chernozhukov et al., 2018).

2.2 Bregman-Riesz Objectives

Let $g\left(x, \alpha\right)$ be convex in the scalar α for each fixed x. The pointwise Bregman divergence is

$$\mathrm{BD}_g\left(\alpha_0(x) \| \alpha(x)\right) := g\left(x, \alpha_0(x)\right) - g\left(x, \alpha(x)\right) - \partial_\alpha g\left(x, \alpha(x)\right)\left(\alpha_0(x) - \alpha(x)\right). \tag{4}$$

Generalized Riesz regression estimates α_0 by minimizing an empirical Bregman-Riesz objective of the form

$$\widehat{L}\left(\alpha\right) := \frac{1}{n}\sum_{i=1}^{n}\Big(- g\left(X_i, \alpha(X_i)\right) + \partial_\alpha g\left(X_i, \alpha(X_i)\right)\alpha(X_i) - m\left(W_i, \partial_\alpha g\left(\cdot, \alpha\left(\cdot\right)\right)\right)\Big) + \lambda\Omega\left(\alpha\right), \tag{5}$$

Table 1: Correspondence among Bregman divergence losses, density ratio (DR) estimation methods, and Riesz representer (RR) estimation. RR estimation for ATE includes propensity score estimation and covariate balancing weights. In the table, $C \in \mathbb{R}$ denotes a constant that is determined by the problem and the loss function. The package also supports user-defined generators.

$g(\alpha)$	**DR estimation**	**RR estimation**
$(\alpha - C)^2$	LSIF (Kanamori et al., 2009) KuLSIF (Kanamori et al., 2012) Hyvärinen score matching (Hyvärinen, 2005)	SQ-Riesz regression (`genriesz`) Riesz regression (RieszNet and ForestRiesz) (Chernozhukov et al., 2021, 2022a) RieszBoost (Lee & Schuler, 2025) KRRR (Singh, 2024) Nearest neighbor matching (Lin et al., 2023) Causal forest and generalized random forest (Wager & Athey, 2018; Athey et al., 2019)
Dual solution with a linear link function		
	Kernel mean matching (Gretton et al., 2009)	Sieve Riesz representer (Chen & Liao, 2015; Chen & Pouzo, 2015) Stable balancing weights (Zubizarreta, 2015; Bruns-Smith et al., 2025) Approximate residual balancing (Athey et al., 2018) Covariate balancing by SVM (Tarr & Imai, 2025) Distributional balancing (Santra et al., 2026)
$\big((\lvert\alpha\rvert - C)\big)\log\big((\lvert\alpha\rvert - C)\big) - \lvert\alpha\rvert$	UKL divergence minimization (Nguyen et al., 2010)	UKL-Riesz regression (`genriesz`) Tailored loss minimization ($\alpha = \beta = -1$) (Zhao, 2019) Calibrated estimation (Tan, 2019)
Dual solution with a logistic or log link function		
	KLIEP (Sugiyama et al., 2008)	Entropy balancing weights (Hainmueller, 2012)
$(\lvert\alpha\rvert - C)\log\big((\lvert\alpha\rvert - C)\big) - (\lvert\alpha\rvert + C)\log\big((\lvert\alpha\rvert + C)\big)$	BKL divergence minimization (Qin, 1998) TRE (Rhodes et al., 2020)	BKL-Riesz regression (`genriesz`) MLE of the propensity score (Standard approach) Tailored loss minimization ($\alpha = \beta = 0$) (Zhao, 2019)
$\frac{\big((\lvert\alpha\rvert - C)\big)^{1+\omega} - \big((\lvert\alpha\rvert - C)\big)}{\omega} - (\lvert\alpha\rvert - C)$ for some $\omega \in (0, \infty)$	BP divergence minimization (Sugiyama et al., 2011)	BP-Riesz regression (`genriesz`)
$C\log(1-\alpha) + C\alpha(\log(\alpha) - \log(1-\alpha))$ for $\alpha \in (0, 1)$	PU learning (du Plessis et al., 2015) Nonnegative PU learning (Kiryo et al., 2017)	PU-Riesz regression (`genriesz`)
General formulation by Bregman divergence minimization	Density-ratio matching (Sugiyama et al., 2011) D3RE (Kato & Teshima, 2021)	Generalized Riesz regression (`genriesz`)

where Ω is a regularizer, for example an ℓ_p penalty on a coefficient vector. Squared-loss generators recover Riesz regression (primal, Chernozhukov et al., 2021) and series Riesz representer (dual, Chen & Liao, 2015), while KL-type generators recover tailored-loss, calibrated estimation formulations (primal, Zhao, 2019; Tan, 2019) and entropy balancing (dual, Hainmueller, 2012). The package provides built-in generators, squared distance (SQ), unnormalized KL (UKL) divergence, binary KL (BKL) divergence, Basu's power (BP) divergence, and PU families, and it also supports user-defined generators.

Dual coordinate and conjugate objective. A key identity behind the implementation is the conjugacy relation

$$g^*(x,v) := \sup_{\alpha\in\mathcal{A}(x)} \{\alpha v - g(x,\alpha)\}, \qquad \alpha^*(x,v) := \arg\max_{\alpha\in\mathcal{A}(x)} \{\alpha v - g(x,\alpha)\}, \tag{6}$$

where $\mathcal{A}(x)$ is the domain of α given x. For differentiable generators, $\alpha^*(x,v) = (\partial_\alpha g)^{-1}(x,v)$. Using $v(x) := \partial_\alpha g(x,\alpha(x))$ and $g^*(x,v(x)) = \alpha(x)v(x) - g(x,\alpha(x))$, the objective (5) is equivalent, up to constants, to

$$\widehat{L}^*(v) := \frac{1}{n}\sum_{i=1}^{n}\Big(g^*(X_i, v(X_i)) - m(W_i, v)\Big) + \lambda\Omega^*(v). \tag{7}$$

This dual view is convenient for optimization and for deriving the balancing optimality conditions.

2.3 Automatic Regressor Balancing via Generator-Induced Links

To fit α_0 in a model class, `genriesz` focuses on GLM-style parameterizations

$$\alpha_\beta(x) = \zeta^{-1}(x, f_\beta(x)), \qquad f_\beta(x) = \phi(x)^\top\beta, \tag{8}$$

where ϕ is a user-chosen basis and ζ is a link. A key choice is to set the link to the derivative of the generator,

$$\zeta(x,\alpha) = \partial_\alpha g(x,\alpha), \qquad \zeta^{-1}(x,\cdot) = (\partial_\alpha g(x,\cdot))^{-1}, \tag{9}$$

so that the dual coordinate $v(x) = \partial_\alpha g(x,\alpha(x))$ is linear in β. This is the package's notion of ARB.

In `genriesz`, the model is estimated by solving a convex program in β:

$$\widehat{\beta} := \arg\min_{\beta\in\mathbb{R}^p}\left\{\frac{1}{n}\sum_{i=1}^{n}\Big(g^*(X_i, f_\beta(X_i)) - m(W_i, f_\beta)\Big) + \lambda\Omega(\beta)\right\}. \tag{10}$$

After estimating $\widehat{\beta}$, the fitted Riesz representer is $\widehat{\alpha}(x) = \zeta^{-1}\Big(x, f_{\widehat{\beta}}(x)\Big)$.

Proposition 2.1 (ARB implies balancing optimality conditions)**.** *Consider the model (8)–(9). Fix $q \in [1,\infty]$ and set $\Omega(\beta) = \frac{1}{q}\|\beta\|_q^q$. Let $\widehat{\beta}$ be any empirical minimizer of (10) and define $\widehat{\alpha}(x) = \alpha_{\widehat{\beta}}(x)$. Under mild regularity conditions, the KKT conditions imply that there exist scalars $s_1,\dots,s_p$ such that*

$$\frac{1}{n}\sum_{i=1}^{n}\Big(\widehat{\alpha}(X_i)\phi_j(X_i) - m(W_i,\phi_j)\Big) + \lambda s_j = 0, \qquad s_j \in \partial\Big(\frac{1}{q}|\beta_j|^q\Big)\Big(\big|_{\beta_j=\widehat{\beta}_j}\Big), \qquad j = 1,\dots,p. \tag{11}$$

Consequently, the implied balancing condition takes the following explicit form:

- *If $q = 1$, then $|s_j| \le 1$ and hence*

$$\left| \frac{1}{n} \sum_{i=1}^{n} \left(\widehat{\alpha}(X_i)\phi_j(X_i) - m\left(W_i, \phi_j\right) \right) \right| \le \lambda, \qquad j = 1, \dots, p. \tag{12}$$

- *If $q > 1$, then $s_j = \operatorname{sign}\left(\widehat{\beta}_j\right) \left|\widehat{\beta}_j\right|^{q-1}$ and hence*

$$\left| \frac{1}{n} \sum_{i=1}^{n} \left(\widehat{\alpha}(X_i)\phi_j(X_i) - m\left(W_i, \phi_j\right) \right) \right| = \lambda \left|\widehat{\beta}_j\right|^{q-1}, \qquad j = 1, \dots, p. \tag{13}$$

In particular, when $\lambda = 0$ and the constraints are feasible, (11) yields exact sample balancing.

Proposition 2.1 is a software-relevant consequence of the duality theory in Kato (2026). ARB ensures that the solver automatically enforces the correct balancing equations for the user-specified basis, even when the primal model (8) is nonlinear in β, for example under KL-type links.

Remark (Automatic link construction in `genriesz`). *Users can supply analytic derivatives $\partial_\alpha g$ and $(\partial_\alpha g)^{-1}$. If they do not, `genriesz` approximates $\partial_\alpha g$ by finite differences and computes $(\partial_\alpha g)^{-1}$ by root finding. This allows rapid prototyping of new Bregman generators, at the cost of additional numerical care, such as domain constraints and branch selection.*

Remark. *ℓ_1 penalty and sparsity The ℓ_1 choice is useful because (12) directly controls the maximum absolute moment imbalance by the single tuning parameter λ. In `genriesz`, this role is primarily about feasibility relaxation and stability of representer fitting, rather than recovering a sparse coefficient vector. In particular, using ℓ_1 here should be understood as imposing an interpretable slack on balancing equations, not as a sparsity assumption on the true representer model.*

2.4 Special Cases and Connections to Balancing Weights

The ARB construction (9) makes explicit how generalized Riesz regression recovers classical balancing-weight estimators as dual solutions. Intuitively, the dual variable associated with the linear moment conditions corresponds to per-sample weights, and different generators g induce different weight regularizers. Table 1 summarizes common choices implemented in `genriesz`, see Kato (2026) for precise statements.

Domain constraints and branch selection. Some generators require constraints such as $|\alpha| > C$ and $\alpha \in (0, 1)$. `genriesz` exposes a `branch_fn` interface that selects a valid branch, for example treated versus control, so that the fitted representer respects the generator domain by construction.

2.5 Estimators: RA, RW, ARW, and TMLE

Given nuisance estimates $\widehat{\gamma}$ and $\widehat{\alpha}$, optionally cross-fitted, `genriesz` outputs four plug-in estimators, regression adjustment (RA), Riesz weighting (RW), augmented Riesz weighting (ARW), and a TMLE-style estimator, defined as follows:

$$\widehat{\theta}^{\mathrm{RA}} := \frac{1}{n}\sum_{i=1}^{n} m\left(W_i, \widehat{\gamma}\right), \tag{14}$$

$$\widehat{\theta}^{\mathrm{RW}} := \frac{1}{n}\sum_{i=1}^{n} \widehat{\alpha}(X_i)Y_i, \tag{15}$$

$$\widehat{\theta}^{\mathrm{ARW}} := \frac{1}{n}\sum_{i=1}^{n} \Big(\widehat{\alpha}(X_i)\left(Y_i - \widehat{\gamma}(X_i)\right) + m\left(W_i, \widehat{\gamma}\right)\Big), \tag{16}$$

$$\widehat{\theta}^{\mathrm{TMLE}} := \frac{1}{n}\sum_{i=1}^{n} m\left(W_i, \widehat{\gamma}^{(1)}\right), \tag{17}$$

In TMLE, $\widehat{\gamma}^{(1)}$ is a one-dimensional fluctuation update of $\widehat{\gamma}$ along the direction $\widehat{\alpha}$. We consider two likelihood choices, Gaussian and Bernoulli with a logit link, which lead to different update maps for $\widehat{\gamma}^{(1)}$. See Appendix B.

Proposition 2.2 (Asymptotic normality)**.** *Suppose $\widehat{\alpha}$ and $\widehat{\gamma}$ are obtained by cross-fitting and satisfy mean-square-error rate conditions such as $\|\widehat{\alpha} - \alpha_0\|_{L_2(P)}\|\widehat{\gamma} - \gamma_0\|_{L_2(P)} = o_p\left(n^{-1/2}\right)$ along with mild moment conditions. Then the ARW estimator in (17) is asymptotically linear with influence function $\psi\left(W; \theta_0, \gamma_0, \alpha_0\right)$ in (3), so that*

$$\sqrt{n}\left(\widehat{\theta}^{\mathrm{ARW}} - \theta_0\right) \xrightarrow{\mathrm{d}} \mathcal{N}\left(0, \mathbb{V}\left[\psi\left(W; \theta_0, \gamma_0, \alpha_0\right)\right]\right).$$

Analogous statements hold for the TMLE-style estimator.

Inference. For each estimator, `genriesz` computes Wald-type standard errors from the empirical variance of the corresponding estimated influence-function scores. Cross-fitting is recommended in high-capacity settings (Chernozhukov et al., 2018).

3 API Design and Software Architecture

The design goal of `genriesz` is to separate *statistical intent*, the functional m and the representer class, from *numerical implementation*, basis matrices, generator derivatives, solvers, and cross-fitting.

3.1 Core Abstractions

The main entry point is `grr_functional`:

```
from genriesz import grr_functional
res = grr_functional(
    X=X, Y=Y,
    m=m, # Functional object
    basis=basis, # Feature map phi(x)
    generator=gen, # BregmanGenerator (or g=..., grad_g=..., inv_grad_g=...,
        grad2_g=...)
    cross_fit=True, folds=5,
    estimators=("ra","rw","arw","tmle"),
)
print(res.summary_text())
```

The returned result object stores point estimates, standard errors, confidence intervals, p-values, and optional out-of-fold nuisance predictions.

Estimators, cross-fitting, and outcome models. The `estimators` argument selects which plug-in estimators to report. The built-in names follow the `genriesz` naming convention, `"ra"` for regression adjustment, `"rw"` for Riesz weighting, `"arw"` for augmented Riesz weighting, and `"tmle"` for the TMLE-style update. Cross-fitting is enabled by `cross_fit=True` with `folds` controlling the number of folds.

For RA, ARW, and TMLE, the package needs an outcome regression model $\widehat{\gamma}$. Users can control its construction by `outcome_models`, for example, `"shared"` fits a linear model using the same basis and regularization interface as the Riesz model, while `"separate"` fits the same outcome regression on a user-supplied outcome basis `outcome_basis`. Setting `outcome_models="none"` skips outcome modeling, then only RW is available.

3.2 Built-In Functionals and Wrappers

For common causal estimands with a binary treatment indicator D stored in a column of X, the package provides convenience wrappers that call `grr_functional` with predefined m:

- `grr_ate`: ATE for $X = [D, Z]$ with $m(W, \gamma) = \gamma(1, Z) - \gamma(0, Z)$.
- `grr_att`: ATT for $X = [D, Z]$. One convenient linear functional is $m(W, \gamma) = \frac{D}{\pi_1}(\gamma(1, Z) - \gamma(0, Z))$ with $\pi_1 := \mathbb{E}[D]$. In the wrapper, π_1 is estimated by $\widehat{\pi}_1 := \frac{1}{n}\sum_{i=1}^{n} D_i$.
- `grr_did`: panel DID implemented as ATT on $\Delta Y := Y_1 - Y_0$, where Y_0 and Y_1 are pre and post outcomes for the same units.
- `grr_ame`: AME via derivatives, requiring a basis that implements $\frac{\partial \phi(x)}{\partial x_k}$.

These wrappers reduce boilerplate for standard workflows while still exposing basis and generator choices.

Algorithm 1 Simplified workflow implemented by `grr_functional`

```
Require: Data $(X_i, Y_i)_{i=1}^n$, functional $m$, basis $\phi$, generator $g$, folds $K$.
1: for $k = 1$ to $K$ (if cross-fitting; else $K = 1$) do
2:     Fit representer model $\widehat{\alpha}^{(-k)}$ on training fold via generalized Riesz regression.
3:     Fit outcome model $\widehat{\gamma}^{(-k)}$ on training fold if RA, ARW, or TMLE is requested.
4:     Predict $\widehat{\alpha}_i$ and $\widehat{\gamma}_i$ on held-out fold.
5: end for
6: Compute RA, RW, ARW, and TMLE estimators.
7: Compute standard errors and confidence intervals from influence-function scores.
8: return Estimates and inference summary.
```

3.3 Basis Functions and Extensibility

The `genriesz` package treats the representer model as linear in a user-specified feature map $\phi(x)$. The core module includes polynomial features and treatment-interaction features for ATE and ATT workflows, as well as RKHS-style approximations via random Fourier features and Nyström features.

Two optional modules expand the basis library: (i) `genriesz.sklearn_basis` provides a random forest leaf one-hot basis that turns a fitted ensemble into a sparse feature map, and (ii) `genriesz.torch_basis` wraps a PyTorch embedding network as a frozen feature map.

Finally, `genriesz` provides a kNN catchment basis `KNNCatchmentBasis` and matching utilities in `genriesz.matching`, which connect nearest-neighbor matching to squared-loss Riesz regression (Kato, 2025; Lin et al., 2023).

Why feature maps. Keeping the solver linear in β in the dual coordinate preserves convexity and the exact ARB optimality conditions in Proposition 2.1. This suggests a practical pattern for neural pipelines, learn an embedding, freeze it, and then fit the representer in the induced feature space.

3.4 Optimization and Regularization

The GLM solver supports ℓ_p penalties for any $p \geq 1$.

Penalty interface. For the Riesz model, set `riesz_penalty="l2"` for ridge, `riesz_penalty="l1"` for lasso, and `riesz_penalty="lp"` with `riesz_p_norm=p` for general $p \geq 1$. A shorthand such as `riesz_penalty="l1.5"` is also supported. When using the default linear outcome regression, the same interface is available via `outcome_penalty` and `outcome_p_norm`.

For $p \geq 1$, `genriesz` uses L-BFGS-B via SciPy, with a smooth approximation of the ℓ_1 subgradient when $p = 1$. The solver exposes iteration limits and tolerances, but defaults aim to work out-of-the-box for moderate feature dimensions.

4 Assumptions on Input Data

genriesz is designed around a simple data interface: X is a NumPy array of shape (n, d) and Y is a vector of shape $(n,)$. The meaning of columns of X is left to the user and to the functional object.

Binary treatment conventions. For wrappers for ATE, ATT, and DID estimation, the treatment indicator D is assumed to be binary and stored at a known column index of X. The user can freely choose the remaining covariates Z.

Panel DID. For grr_did, the package expects two outcome vectors Y0 and Y1 representing pre and post outcomes for the same units and computes $\Delta Y := Y_1 - Y_0$ internally.

Sampling and inference. The default inference uses an i.i.d. approximation and Wald-type confidence intervals. As with standard DML software, clustered or dependent data require user-supplied adaptations, for example cluster-robust variance estimators, which are not yet part of the core release.

5 Examples and Reproducibility

The repository includes runnable scripts and Jupyter notebooks illustrating the workflows below.

5.1 ATE with Polynomial Features and UKL Generator

Let $X = [D, Z]$ and choose a polynomial basis with treatment interactions:

```
from genriesz import (
    grr_ate,
    PolynomialBasis, TreatmentInteractionBasis,
    UKLGenerator
)

psi = PolynomialBasis(degree=2, include_bias=True)
phi = TreatmentInteractionBasis(base_basis=psi)
gen = UKLGenerator(C=1.0, branch_fn=lambda x: int(x[0] == 1.0)).as_generator()

res = grr_ate(
    X=X, Y=Y,
    basis=phi,
    generator=gen,
    cross_fit=True, folds=5,
    estimators=("ra","rw","arw","tmle"),
    riesz_penalty="l2", riesz_lam=1e-3,
)
print(res.summary_text())
```

5.2 ATT and Panel DID

ATT and panel DID are available via wrapper functions:

```
from genriesz import (
    grr_att, grr_did,
    PolynomialBasis, TreatmentInteractionBasis,
    SquaredGenerator
)

psi = PolynomialBasis(degree=2, include_bias=True)
phi = TreatmentInteractionBasis(base_basis=psi)
gen = SquaredGenerator().as_generator()

res_att = grr_att(
    X=X, Y=Y,
    basis=phi,
    generator=gen,
    cross_fit=True, folds=5,
    estimators=("arw","tmle"),
)

res_did = grr_did(
    X=X, Y0=Y0, Y1=Y1,
    basis=phi,
    generator=gen,
    cross_fit=True, folds=5,
    estimators=("arw","tmle"),
)
```

5.3 Average Marginal Effects

Average marginal effects can be computed when the chosen basis implements derivatives:

```
from genriesz import grr_ame, PolynomialBasis, SquaredGenerator

phi = PolynomialBasis(degree=2, include_bias=True)
res_ame = grr_ame(
    X=X, Y=Y,
    coordinate=2,
    basis=phi,
    generator=SquaredGenerator().as_generator(),
    cross_fit=True, folds=5,
    estimators=("ra","rw","arw","tmle"),
)
```

5.4 Nearest-Neighbor Matching

Nearest-neighbor matching weights can be computed using the built-in matching Riesz method:

```
from genriesz import grr_ate, PolynomialBasis

# A basis is required by the API. When outcome_models="none", it is not used.
basis = PolynomialBasis(degree=1, include_bias=True)

res_match = grr_ate(
    X=X, Y=Y,
    basis=basis,
    riesz_method="nn_matching",
    M=1,
    cross_fit=False,
    outcome_models="none",
    estimators=("rw",),
)
```

6 Project Development and Dependencies

Availability and installation. The package is available on PyPI and can be installed by `pip install genriesz`. The package is also available on GitHub `https://github.com/MasaKat0/genriesz`. Optional extras enable scikit-learn-based and PyTorch-based bases. The documentation is hosted at `https://genriesz.readthedocs.io`.

Dependencies. The core package depends only on NumPy and SciPy (Harris et al., 2020; Virtanen et al., 2020). Optional extras provide scikit-learn-based tree feature maps and PyTorch-based neural feature maps (Pedregosa et al., 2011; Buitinck et al., 2013; Paszke et al., 2019).

Quality control. The project includes unit tests, type hints, and continuous integration to catch regressions. The software is released under the GNU General Public License v3.0 (GPL-3.0).

7 Comparison to Related Software

Debiased ML and causal ML libraries. `DoubleML` (Bach et al., 2022, 2024) and `EconML` (Battocchi et al., 2019) provide production-quality implementations of canonical DML estimators, with a focus on treatment effect estimation and, in `EconML`, heterogeneous effects. `DoWhy` (Sharma & Kiciman, 2020) provides an end-to-end causal inference interface centered on causal graphs. `CausalML` (Chen et al., 2020) offers a broad suite of uplift modeling and heterogeneous effect estimators.

`genriesz` is complementary, it targets low-dimensional linear functionals with valid inference via orthogonalization and emphasizes representer estimation and balancing. This estimand-centric API makes it convenient to prototype new targets by defining only the oracle m, while keeping the representer fitting problem explicit.

Balancing-weight and density ratio toolkits. Several toolkits implement specific balancing-weight estimators, such as entropy balancing or stable weights, for ATE-like problems. Density ratio estimation packages such as `densratio` (Makiyama, 2019) implement methods such as uLSIF, RuLSIF, and KLIEP. `genriesz` connects these approaches to Riesz representer estimation and DML-style inference through a single interface.

8 Conclusion

The `genriesz` package provides an estimand-centric implementation of automatic debiased machine learning via generalized Riesz regression under Bregman divergences. By unifying representer fitting, automatic regressor balancing, and debiased estimation behind modular abstractions, the package aims to make automatic debiasing practical for a wide range of causal and structural parameter estimation problems.

References

Susan Athey, Guido W. Imbens, and Stefan Wager. Approximate residual balancing: debiased inference of average treatment effects in high dimensions. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 80(4):597–623, 2018. 4

Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *The Annals of Statistics*, 47(2):1148 – 1178, 2019. 4

Philipp Bach, Victor Chernozhukov, Malte S. Kurz, and Martin Spindler. Doubleml - an object-oriented implementation of double machine learning in python. *Journal of Machine Learning Research*, 23(53):1–6, 2022. 2, 12

Philipp Bach, Malte S. Kurz, Victor Chernozhukov, Martin Spindler, and Sven Klaassen. Doubleml: An object-oriented implementation of double machine learning in r. *Journal of Statistical Software*, 108(3):1–56, 2024. 12

Keith Battocchi, Eleanor Dillon, Maggie Hei, Greg Lewis, Paul Oka, Miruna Oprescu, and Vasilis Syrgkanis. EconML: A Python package for ML-based heterogeneous treatment effects estimation. `https://github.com/py-why/EconML`, 2019. 2, 12

David Bruns-Smith, Oliver Dukes, Avi Feller, and Elizabeth L Ogburn. Augmented balancing weights as linear regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 04 2025. 4

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013. 12

Huigang Chen, Totte Harinen, Jeong-Yoon Lee, Mike Yung, and Zhenyu Zhao. CausalML: Python package for causal machine learning, 2020. 2, 12

Xiaohong Chen and Zhipeng Liao. Sieve semiparametric two-step gmm under weak dependence. *Journal of Econometrics*, 189(1):163–186, 2015. 4

Xiaohong Chen and Demian Pouzo. Sieve wald and qlr inferences on semi/nonparametric conditional moment models. *Econometrica*, 83(3):1013–1079, 2015. 4

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 2018. 2, 3, 7

Victor Chernozhukov, Whitney K. Newey, Victor Quintas-Martinez, and Vasilis Syrgkanis. Automatic debiased machine learning via riesz regression, 2021. arXiv:2104.14737. 4

Victor Chernozhukov, Whitney Newey, Víctor M Quintas-Martínez, and Vasilis Syrgkanis. RieszNet and ForestRiesz: Automatic debiased machine learning with neural nets and random forests. In *International Conference on Machine Learning (ICML)*, 2022a. 4

Victor Chernozhukov, Whitney K. Newey, and Rahul Singh. Automatic debiased machine learning of causal and structural effects. *Econometrica*, 90(3):967–1027, 2022b. 2

Marthinus Christoffel du Plessis, Gang. Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *International Conference on Machine Learning (ICML)*, 2015. 4

A. Gretton, A. J. Smola, J. Huang, Marcel Schmittfull, K. M. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. *Dataset Shift in Machine Learning, 131-160 (2009)*, 01 2009. 4

Jens Hainmueller. Entropy balancing for causal effects: A multivariate reweighting method to produce balanced samples in observational studies. *Political Analysis*, 20(1):25–46, 2012. 3, 4

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, 2020. 12

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. 4

Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul.):1391–1445, 2009. 4

Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. Statistical analysis of kernel-based least-squares density-ratio estimation. *Mach. Learn.*, 86(3):335–367, March 2012. ISSN 0885-6125. 4

Masahiro Kato. Nearest neighbor matching as least squares density ratio estimation and riesz regression, 2025. arXiv: 2510.24433. 9

Masahiro Kato. A unified framework for debiased machine learning: Riesz representer fitting under bregman divergence, 2026. arXiv: 2601.07752. 2, 3, 6

Masahiro Kato and Takeshi Teshima. Non-negative bregman divergence minimization for deep direct density ratio estimation. In *International Conference on Machine Learning (ICML)*, 2021. 4

Ryuichi Kiryo, Gang Niu, Marthinus Christoffel du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 4

Kaitlyn J. Lee and Alejandro Schuler. Rieszboost: Gradient boosting for riesz regression, 2025. arXiv: 2501.04871. 4

Zhexiao Lin, Peng Ding, and Fang Han. Estimation based on nearest neighbor matching: from density ratio to average treatment effect. *Econometrica*, 91(6):2187–2217, 2023. 4, 9

Koji Makiyama. densratio: Density ratio estimation. `https://CRAN.R-project.org/package=densratio`, 2019. R package version 0.2.1. 3, 13

XuanLong Nguyen, Martin Wainwright, and Michael Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE*, 2010. 4

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: an imperative style, high-performance deep learning library. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 12

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 12

Jing Qin. Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, 85(3):619–630, 1998. 4

B. Rhodes, K. Xu, and M.U. Gutmann. Telescoping density-ratio estimation. In *NeurIPS*, 2020. 4

Diptanil Santra, Guanhua Chen, and Chan Park. Distributional balancing for causal inference: A unified framework via characteristic function distance, 2026. arXiv: 2601.15449. 4

Amit Sharma and Emre Kiciman. Dowhy: An end-to-end library for causal inference, 2020. 2, 12

Rahul Singh. Kernel ridge riesz representers: Generalization, mis-specification, and the counterfactual effective dimension, 2024. arXiv: 2102.11076. 4

Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008. 4

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density ratio matching under the bregman divergence: A unified framework of density ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64, 10 2011. 4

Zhiqiang Tan. Regularized calibrated estimation of propensity scores with model misspecification and high-dimensional data. *Biometrika*, 107(1):137–158, 2019. 4

Alexander Tarr and Kosuke Imai. Estimating average treatment effects with support vector machines. *Statistics in Medicine*, 44(5), 2025. 4

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 12

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018. 4

Qingyuan Zhao. Covariate balancing propensity score by tailored loss functions. *The Annals of Statistics*, 47(2):965 – 993, 2019. 4

José R. Zubizarreta. Stable weights that balance covariates for estimation with incomplete outcome data. *Journal of the American Statistical Association*, 110(511):910–922, 2015. 3, 4

A Built-in Bregman generators

This appendix lists the built-in Bregman generators and their induced links.

Squared distance generator. `SquaredGenerator` uses

$$g(\alpha) = (\alpha - C)^2 .$$

The link is $\zeta(x, \alpha) = \partial_\alpha g(\alpha) = 2(\alpha - C)$.

Unnormalized KL divergence generator. `UKLGenerator` uses, for $|\alpha| > C$,

$$g(\alpha) = (|\alpha| - C)\log(|\alpha| - C) - |\alpha| .$$

The link is $\zeta(x, \alpha) = \operatorname{sign}(\alpha)\log(|\alpha| - C)$.

Binary KL divergence generator. `BKLGenerator` uses, for $|\alpha| > C$,

$$g(\alpha) = (|\alpha| - C)\log(|\alpha| - C) - (|\alpha| + C)\log(|\alpha| + C) .$$

The corresponding link is $\zeta(x, \alpha) = \operatorname{sign}(\alpha)\Big(\log(|\alpha| - C) - \log(|\alpha| + C)\Big)$.

Basu's power divergence generator. `BPGenerator` uses, for some $\omega > 0$ and $|\alpha| > C$,

$$g(\alpha) = \frac{(|\alpha| - C)^{1+\omega} - (|\alpha| - C)}{\omega} - (|\alpha| - C) .$$

The link is $\zeta(x, \alpha) = \operatorname{sign}(\alpha)\frac{1+\omega}{\omega}\left((|\alpha| - C)^{\omega} - 1\right)$.

PU learning loss generator. `PUGenerator` uses, for $|\alpha| \in (0, 1)$,

$$g(\alpha) = C\Big(|\alpha|\log(|\alpha|) + (1 - |\alpha|)\log(1 - |\alpha|)\Big).$$

The link is $\zeta(x, \alpha) = \operatorname{sign}(\alpha)\, C\Big(\log(|\alpha|) - \log(1 - |\alpha|)\Big)$.

B Likelihoods in TMLE

Gaussian likelihood. When Y is continuous, we use the additive fluctuation

$$\widehat{\gamma}^{(1)}(x) := \widehat{\gamma}(x) + \widehat{\epsilon}\widehat{\alpha}(x), \qquad \widehat{\epsilon} := \frac{\frac{1}{n}\sum_{i=1}^{n}\widehat{\alpha}(X_i)\left(Y_i - \widehat{\gamma}(X_i)\right)}{\frac{1}{n}\sum_{i=1}^{n}\widehat{\alpha}(X_i)^2}. \tag{18}$$

Since the functional $\gamma \mapsto m(W, \gamma)$ is linear, (17) can be written as

$$\widehat{\theta}^{\mathrm{TMLE}} = \widehat{\theta}^{\mathrm{RA}} + \widehat{\epsilon}\frac{1}{n}\sum_{i=1}^{n} m\left(W_i, \widehat{\alpha}\right). \tag{19}$$

Bernoulli likelihood. When $Y \in \{0,1\}$, we use a logistic fluctuation that updates $\widehat{\gamma}$ on the logit scale. Define $\Lambda(t) \coloneqq 1/\left(1+\exp\left(-t\right)\right)$ and $\operatorname{logit}(p) \coloneqq \log\left(\frac{p}{1-p}\right)$. We set

$$\widehat{\gamma}^{(1)}(x) \coloneqq \Lambda\left(\operatorname{logit}\left(\widehat{\gamma}(x)\right) + \widehat{\epsilon}\widehat{\alpha}(x)\right), \tag{20}$$

where $\widehat{\epsilon}$ is defined as a solution to the one-dimensional score equation

$$\frac{1}{n}\sum_{i=1}^{n} \widehat{\alpha}(X_i)\left(Y_i - \widehat{\gamma}^{(1)}(X_i)\right) = 0. \tag{21}$$

The TMLE-style estimator is then computed by plugging $\widehat{\gamma}^{(1)}$ into (17). In contrast to the Gaussian case, the representation (19) does not generally hold because the map $\epsilon \mapsto \widehat{\gamma}^{(1)}$ is nonlinear under the logit fluctuation.