Introduction to Usage of SIHR

Zhenyu (Zach) Wang

April 11, 2024

This vignette presents the usage of **SIHR** package for statistical inference in high-dimensional generalized linear models with continuous and binary outcomes. The package provides tools for constructing confidence intervals and performing hypothesis tests for low-dimensional objectives in both one-sample and two-sample regression settings. If you use the package `SIHR` package in your analysis and publications, please cite:

> Rakshit, P., Wang, Z., Cai, T. T., & Guo, Z. (2021). SIHR: Statistical Inference in High-Dimensional Linear and Logistic Regression Models. arXiv preprint arXiv:2109.03365.

Note that **SIHR** package includes several debiasing methodologies, published in different papers. Please make sure to cite papers associated with the methodologies you used. Citations for those can be found in Table 1.

Installations: The latest version of the package can be installed from CRAN repository mirror:

```r
# Install
install.pacakges("SIHR")
# Load
library(SIHR)
```

Or users can install it through GitHub:

```r
# Install
devtools::install_github("zywang0701/SIHR")
```

In the website `https://github.com/zywang0701/SIHR`, we provide other vignettes, regarding more details about the debiasing methodologies.

Contents

1 Overview of SIHR

Package Features: Figure 1 provides a quick overview of **SIHR** package's features. The following sections will provide more details, as well as sample code on how to perform tasks. The **SIHR** package consists of five main functions `LF()`, `QF()`, `CATE()`, `InnProd()`, and `Dist()` implementing the statistical inferences for five different quantities correspondingly, under the one-sample model or two-sample model regime. And we provide two methods `ci()` and `summary()` to report the inference results after running the main functions.

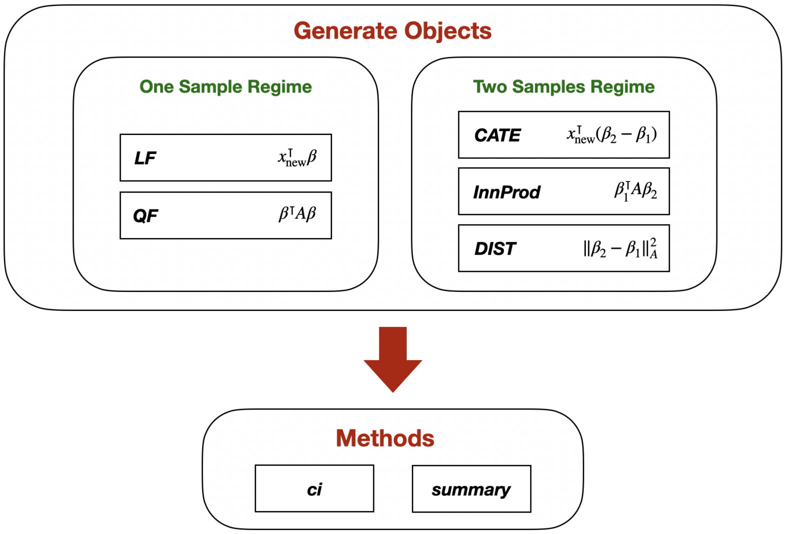
Generate Objects
One Sample Regime
LF $x_{\text{new}}^{\intercal}\beta$
QF $\beta^{\intercal}A\beta$
Two Samples Regime
CATE $x_{\text{new}}^{\intercal}(\beta_2-\beta_1)$
InnProd $\beta_1^{\intercal}A\beta_2$
DIST $\|\beta_2-\beta_1\|_A^2$
Methods
ci
summary

Figure 1: Overview of Package Features

Model Setup We consider the high-dimensional GLMs: for $1 \leq i \leq n$,

$$\mathbb{E}(y_i \mid X_{i\cdot}) = f(X_{i\cdot}^{\intercal}\beta), \quad \text{with } f(z) = \begin{cases} z & \text{for linear model;} \\ \exp(z)/\left[1+\exp(z)\right] & \text{for logistic model;} \end{cases} \tag{1}$$

where $\beta \in \mathbb{R}^p$ denotes the high-dimensional regression vector, $y_i \in \mathbb{R}$ and $X_{i\cdot} \in \mathbb{R}^p$ denote respectively the outcome and the measured covariates of the i-th observation. Throughout the paper, define $\Sigma = \mathbb{E}X_{i\cdot}X_{i\cdot}^{\intercal}$ and assume β to be a sparse vector with its sparsity level denoted as $\|\beta\|_0$. In addition to the one-sample setting, we examine the statistical inference methods for the two-sample regression models. Particularly, we generalize the regression model in (1) and consider:

$$\mathbb{E}(y_i^{(k)} \mid X_{i\cdot}^{(k)}) = f(X_{i\cdot}^{(k)\intercal}\beta^{(k)}) \quad \text{with } k = 1, 2 \text{ and } 1 \leq i \leq n_k, \tag{2}$$

where $f(\cdot)$ is the pre-specified link function defined as (1), $\beta^{(k)} \in \mathbb{R}^p$ denotes the high-dimensional regression vector in k-th sample, $y_i^{(k)} \in \mathbb{R}$ and $X_{i\cdot}^{(k)} \in \mathbb{R}^p$ denote respectively the outcome and the measured covariates in the k-th sample.

Description of Main Functions The **SIHR** package implements a number of published debiasing methodologies. Table 1 gives a short description of each one of the methodologies.

Outline of Remaining Content: In the following sections, we will discuss the implemented methodologies in one-sample and two-sample regimes separately. We will also showcase the code to conduct statistic inference on simulated data.

Regime	Key	Description
One-Sample	`LF`	Abbreviated for linear functional, it implements the inference approach for $x_{\text{new}}^{\intercal}\beta$ proposed in Cai et al. [1, 3], with $x_{\text{new}} \in \mathbb{R}^p$ denoting a loading vector. With $x_{\text{new}} = e_j$ as a special case, `LF()` infers the regression coefficient β_j [9, 7, 10, e.g.]. When x_{new} denotes a future observation's covariates, `LF()` makes inferences for the conditional mean of the outcome for the individual. **Reference:** Cai et al. [1], Cai et al. [3]
	`QF`	Abbreviated for quadratic functional, it makes inferences for $\beta_{\text{G}}^{\intercal}A\beta_{\text{G}}$, following the proposal in Guo et al. [4, 5], Cai and Guo [2]. $A \in \mathbb{R}^{\lvert\text{G}\rvert\times\lvert\text{G}\rvert}$ is either a pre-specified submatrix or the unknown $\Sigma_{\text{G},\text{G}}$ and $\text{G} \in \{1,...,p\}$ denotes the index set of interest; $\beta_{\text{G}}^{\intercal}A\beta_{\text{G}}$ can be viewed as a total measure of all effects of variables in the group G. **Reference:** Guo et al. [4], Guo et al. [5], Cai and Guo [2]
Two-Samples	`CATE`	Abbreviated for conditional average treatment effect, it makes inference for $f(x_{\text{new}}^{\intercal}\beta^{(2)}) - f(x_{\text{new}}^{\intercal}\beta^{(1)})$, see [1] for detailed discussion. This difference measures the discrepancy between conditional means, closely related to the conditional average treatment effect for the new observation with covariates x_{new}. **Reference:** Cai et al. [1]
	`InnProd`	Abbreviated for inner products, it implements the statistical inference for $\beta_{\text{G}}^{(1)\intercal}A\beta_{\text{G}}^{(2)}$ with $A \in \mathbb{R}^{\lvert G\rvert\times\lvert G\rvert}$, which was proposed in [4, 8]. The inner products measure the similarity between the high-dimensional vectors $\beta^{(1)}$ and $\beta^{(2)}$, which is useful in capturing the genetic relatedness in the GWAS applications [4, 8]. **Reference:** Guo et al. [4], Ma et al. [8]
	`Dist`	Short-handed for distance, it makes inferences for the weighted distances $\gamma_{\text{G}}^{\intercal}A\gamma_{\text{G}}$ with $\gamma = \beta^{(2)} - \beta^{(1)}$. The distance measure is useful in comparing different high-dimensional regression vectors and constructing a generalizable model in the multisource learning problem, see Guo et al. [6].

Table 1: Description of the main functions and implemented methodologies.

2 One-Sample Regime

2.1 LF

The function `LF()`, shorthanded for Linear Functional, performs inference for $x_{\text{new}}^{\intercal}\beta$ under the high-dimensional model (1). A typical `LF()` code snippet looks like:

```
LF(X, y, loading.mat, model=c("linear","logistic","logistic_alter"),
intercept=TRUE, intercept.loading=FALSE, beta.init=NULL, lambda=NULL, mu=NULL,
prob.filter=0.05, rescale=1.1, alpha=0.05, verbose=FALSE)
```

2.1.1 Arguments

Here we pick some arguments to explain.

- `loading.mat` is the matrix of loading vectors where each column corresponds to a new future observation x_{new}. It is designed to allow for multiple x_{new} simultaneously as input, thereby saving the computational time and labour of running the algorithm multiple times, once for each x_{new}.

- `model` specifies which high-dimensional regression model is to be fit, the choices being `c("linear", "logistic","logistic_alt")`. In particular, this argument specifies the link function and the corresponding weight function to be used. For the details, please check another Vignette. to include

- `intercept` is a logical argument which specifies whether intercept should be fitted while computing the initial estimator $\widehat{\beta}$.

- `intercept.loading` is also logical, specifying whether the intercept term should be included or not for defining the objective $x_{\text{new}}^{\intercal}\beta$.

- `beta.init` allows the user to supply the initial estimator $\widehat{\beta}$ of the regression vector. If `beta.init` is left as `NULL`, the initial estimator $\widehat{\beta}$ is computed using `cv.glmnet`.

- `lambda` denotes the scaled tuning parameter $\lambda_0/\sqrt{n}$ used for computing the initial estimator $\widehat{\beta}$ in (**??**) which can either be pre-specified or can be set to `NULL` whence `LF` uses `cv.glmnet` to compute it.

- `mu` is the parameter related for computing projection direction, to include

- `prob.filter` is specific to logistic model. We need to exclude those observations whose estimated probabilities $\mathbb{P}(y_i \mid X_i)$ are very close to 0 or 1. Those samples for which the estimated probability lies outside $[\texttt{prob.filter}, 1 - \texttt{prob.filter}]$ before proceeding with the algorithm.

- `rescale` is the factor to enlarge the standard error to account for the finite sample bias.

- `alpha` denotes the level of significance α in hypothesis testing.

2.1.2 Code

Example 1 (LF for linear regression) *For $n = 100, p = 120$, the covariates $X_i \sim N(\mathbf{0}_p, \mathbf{I}_p)$. The outcome is generated as $y_i = a_0 + X_{i\cdot}^{\intercal}\beta + \epsilon_i$ with $\epsilon_i \sim N(0,1)$, see the code for the specification of a_0 and β.* *Objective: Given two further observations $x_{\text{new}}^{(1)}, x_{\text{new}}^{(2)}$, we're going to make inference for $x_{\text{new}}^{(1)\intercal}\beta = 1.5$ and $x_{\text{new}}^{(2)\intercal}\beta = -1.25$ simultaneously.*

```
# Generate Data
set.seed(0)
n = 100; p = 120
mu = rep(0,p); Cov = diag(p)
a0 = 0; beta = rep(0,p); beta[c(1,2)] = c(0.5, 1)
X = MASS::mvrnorm(n, mu, Cov)
y = a0 + X %*% beta + rnorm(n)

# Loadings
loading1 = c(1, 1, rep(0, p-2))
loading2 = c(-0.5, -1, rep(0, p-2))
loading.mat = cbind(loading1, loading2)

# Run LF
Est = LF(X, y, loading.mat, model='linear')
```

Having fitted the model, we have two following methods `ci()` and `summary()`.

```
ci(Est)
#>  loading     lower      upper
#>1       1  1.167873  1.8753934
#>2       2 -1.544138 -0.7995375
```

In the above result, we can find the 95% CI for $x_{\text{new}}^{(1)\intercal}\beta$ and $x_{\text{new}}^{(2)\intercal}\beta$. Both true values $x_{\text{new}}^{(1)\intercal}\beta = 1.5$ and $x_{\text{new}}^{(2)\intercal}\beta = -1.25$ lie in the corresponding CIs.

```
summary(Est)
#>Call:
#>Inference for Linear Functional
#>
#>Estimators:
#> loading est.plugin est.debias Std. Error z value  Pr(>|z|)
#>       1      1.268      1.522     0.1805   8.430 0.000e+00 ***
#>       2     -1.033     -1.172     0.1900  -6.169 6.868e-10 ***
```

`summary()` returns a list of the summary statistics, in which we can find the plugin estimator, bias-corrected estimator, and the standard error for the bias-corrected estimator. The bias-corrected estimators are closer to the true values.

As a second example, we consider the logistic regression where the argument `model` is set as `"logistic"` or `"logistic_alter"`.

Example 2 (LF for logistic regression) *For $n = 130, p = 120$, the covariates $X_i \sim N(\mathbf{0}_p, \mathbf{I}_p)$. The regression vector $\beta \in \mathbb{R}^p$ is generated as shown in the following code. We generate the outcome $Y_i \sim$* Bernoulli $(f(a_0 + X_i^\intercal \beta))$ *with $f(z) = \exp(z)/[1+\exp(z)]$. Objective: The target parameter values are $x_{\rm new}^{(1)\intercal}\beta = 2$ and $x_{\rm new}^{(2)\intercal}\beta = -2.5$ corresponding to the two loadings $x_{new}^{(1)}$ and $x_{new}^{(2)}$.*

```
# Generate Data
n = 300; p = 120
mu = rep(0,p); Cov = diag(p)
a0 = 0
beta = rep(0,p); beta[c(1,2)] = c(1, 1)
X = MASS::mvrnorm(n, mu, Cov)
val = a0 + X %*% beta
y = rbinom(n, 1, exp(val)/(1+exp(val)))

# Loadings
loading1 = c(1, 1, rep(0, p-2))
loading2 = c(-0.5, -2, rep(0, p-2))
loading.mat = cbind(loading1, loading2)
```

To boost computation efficiency, we may specify the argument `beta.init` as the initial coefficients estimators. For example, here we use `cv.glmnet` to construct $\widehat{\beta}$ and pass them on to the argument `beta.init` in `LF`.

```
cv.fit = glmnet::cv.glmnet(X, y, family='binomial', alpha=1, standardize=TRUE)
beta.init = as.vector(coef(cv.fit, s=cv.fit[['lambda.min']]))
Est = LF(X, y, loading.mat, model='logistic', beta.init=beta.init)
```

The corresponding CIs and summary statistics are given below:

```
ci(Est)
#>  loading       lower       upper
#>1       1  1.257559  2.492327
#>2       2 -3.186513 -1.605671
```

Consequently, we have two objective values $x_{\rm new}^{(1)\intercal}\beta = 2$ and $x_{\rm new}^{(2)\intercal}\beta = -2.5$. Both of these values lie within their corresponding 95% CIs.

```
summary(Est)
#> Call:
#> Inference for Linear Functional
#>
#> Estimators:
#> loading est.plugin est.debias Std. Error z value  Pr(>|z|)
#>       1      1.340      1.875     0.3150   5.952 2.645e-09 ***
#>       2     -1.741     -2.396     0.4033  -5.941 2.825e-09 ***
```

Note that the plugin estimators $x_{\rm new}^{(1)\intercal}\widehat{\beta}$ and $x_{\rm new}^{(2)\intercal}\widehat{\beta}$ are severely biased in such setting, the proposed bias-correction approach significantly saves the bias with $\widehat{x_{\rm new}^{(1)\intercal}\beta}$ and $\widehat{x_{\rm new}^{(2)\intercal}\beta}$.

2.2 QF

The function `QF()`, abbreviated for Quadratic Functional, conducts inference for $\beta_{\rm G}^\intercal A \beta_{\rm G}$ if A is the submatrix pre-specified or $\beta_{\rm G}^\intercal \Sigma_{\rm G,G} \beta_{\rm G}$ under the high-dimensional regression model (1). The function `QF()` can be called with the following arguments.

```
QF(X, y, G, A=NULL, model=c("linear","logistic","logistic_alter"), intercept=TRUE,
beta.init=NULL, split=TRUE, lambda=NULL, mu=NULL, prob.filter=0.05,
rescale=1.1,tau=c(0.25, 0.5, 1), alpha=0.05, verbose=FALSE)
```

2.2.1 Arguments

We only explain new arguments, as all other arguments for `QF` are defined similarly as the function `LF`.

- `G` is the set of indices of interest.
- If `A` is specified, it will conduct inference for $\beta_{\rm G}^{\intercal} A\beta_{\rm G}$; otherwise, it will turn to $\beta_{\rm G}^{\intercal}\Sigma_{\rm G,G}\beta_{\rm G}$.
- `split` indicates whether we conduct the sample splitting for computing the initial estimator of regression coefficients. When `split=TRUE`, the initial estimator of regression coefficients is computed using half of the available observations while the remaining half is used for bias correction. The option of using sampling splitting might require a larger sample size.
- `tau.vec` allows the user to supply a vector of possible values to enlarge the variance estimator for some technical reasons. to include

2.2.2 Code

Example 3 (QF for linear regression) *For $n = 200, p = 150$, the covariates $X_{i\cdot}$ is generated from multivariate normal distribution with mean $\mu = 0_p$ and covariance $\Sigma \in \mathbb{R}^{p\times p}$, where $\Sigma_{j,k} = 0.5^{|j-k|}$. See the following code for the vector of regression coefficients β. The outcome is generated by $y_i = X_{i\cdot}\beta + \epsilon_i$ with standard normal distributed noise. Objective: We're going to make inference for $\beta_{\rm G}^{\intercal}\Sigma_{\rm G,G}\beta_{\rm G}$ with $G = \{40, \ldots, 60\}$*

```
# Generate Data
n = 200; p = 150
mu = rep(0,p)
Cov = matrix(0, p, p); for(j in 1:p) for(k in 1:p) Cov[j,k] = 0.5^{abs(j-k)}
beta = rep(0, p); beta[25:50] = 0.2
X = MASS::mvrnorm(n,mu,Cov)
y = X%*%beta + rnorm(n)

# Specify set G
test.set =c(40:60)

# Run QF
Est = QF(X, y, G = test.set, A = NULL, model = "linear", split=FALSE)
```

Continuing running two functions `ci()` and `summary()`:

```
ci(Est)
#>   tau     lower    upper
#>1 0.25 0.8118792 1.466422
#>2 0.50 0.8046235 1.473677
#>3 1.00 0.7905648 1.487736
```

With the default $\tau = c(0.25, 0.5, 1)$, we obtain three different CIs for $\beta_{\rm G}^{\intercal}\Sigma_{\rm G,G}\beta_{\rm G}$. Note that the true value $\beta_{\rm G}^{\intercal}\Sigma_{\rm G,G}\beta_{\rm G} = 1.16$ belongs to all of these constructed CIs.

```
summary(Est)
#> Call:
#> Inference for Quadratic Functional
#>
#>  tau est.plugin est.debias Std. Error z value  Pr(>|z|)
#> 0.25      0.904      1.139     0.1670   6.822 8.969e-12 ***
#> 0.50      0.904      1.139     0.1707   6.674 2.486e-11 ***
#> 1.00      0.904      1.139     0.1779   6.405 1.504e-10 ***
```

For different τ value, the plugin estimator and bias-corrected estimator of $\beta_{\rm G}^{\intercal}\Sigma_{\rm G,G}\beta_{\rm G}$ remain the same; whereas bigger τ value yields a larger standard error, leading to a wider CI. Similarly to the `LF()` case, our proposed bias-corrected estimator is effective in correcting the bias of plugin estimator.

3 Two-Samples Regime

3.1 CATE

The function `CATE()`, shorthanded for Conditional Average Treatment Effect, conducts inference for $\Delta(x_{\text{new}}) = f(x_{\text{new}}^{\intercal}\beta^{(2)}) - f(x_{\text{new}}^{\intercal}\beta^{(1)})$ under the high-dimensional regression model (2).

```
CATE(X1, y1, X2, y2, loading.mat, model=c("linear","logistic","logistic_alter"),
intercept=TRUE, intercept.loading=FALSE, beta.init1=NULL, beta.init2=NULL,
lambda=NULL, mu=NULL, prob.filter=0.05, rescale=1.1, alpha=0.05, verbose=FALSE)
```

3.1.1 Arguments

Here, `X1` and `y1` denote the design matrix and the response vector for the first sample of data respectively, while `X2` and `y2` denote those for the second sample of data. `beta.init1` and `beta.init2` are the initial estimator of the regression vector for the first and second samples. All other arguments are similarly defined as for the function `LF()`.

3.1.2 Code

We consider the logistic regression case to illustrate `CATE()` with the argument `model=’logistic_alter’`.

Example 4 (CATE for logistic regression) *In the first group of data, the covariates $X_{i\cdot}^{(1)}$, for $1 \leq i \leq n_1$ with $n_1 = 100$, follows multivariate normal distribution with $\mu = 0_p$ and covariance $\Sigma = \mathbf{I}_p$; in the second group of data, the covariates $X_{i\cdot}^{(2)}$, for $1 \leq i \leq n_2$ with $n_2 = 180$, follows multivariate normal distribution with $\mu = 0_p$ and covariance $\Sigma \in \mathbb{R}^{p\times p}$ with $p = 120$ and $\Sigma_{j,k} = 0.5^{|j-k|}$. We generate following the model $y_i^{(k)} \sim$ Bernoulli$(f(X_{i\cdot}^{(k)\intercal}\beta^{(k)})$ with $f(z) = \exp(z)/[1+\exp(z)]$ for $k = 1, 2$. See the following code for details of $\beta^{(1)}$, $\beta^{(2)}$. Objective: We perform inference for $\Delta(x_{\text{new}})$, where the loading x_{new} is generated in the following line of code.*

```r
# Generate Data
n1 = 100; n2 = 180; p = 120
mu1 = mu2 = rep(0,p)
Cov1 = diag(p)
Cov2 = matrix(0, p, p); for(j in 1:p) for(k in 1:p) Cov2[j,k] = 0.5^{abs(j-k)}
beta1 = rep(0, p); beta1[c(1,2)] = c(0.5, 0.5)
beta2 = rep(0, p); beta2[c(1,2)] = c(1.8, 1.8)
X1 = MASS::mvrnorm(n1,mu1,Cov1); val1 = X1%*%beta1
X2 = MASS::mvrnorm(n2,mu2,Cov2); val2 = X2%*%beta2
y1 = rbinom(n1, 1, exp(val1)/(1+exp(val1)))
y2 = rbinom(n2, 1, exp(val2)/(1+exp(val2)))

# Loading
loading.mat = c(1, 1, rep(0, p-2))

# Run CATE
Est = CATE(X1, y1, X2, y2,loading.mat, model="logistic_alter")
```

Having fitted the model, it allows for method `ci()` and `summary()` as `LF()` does.

```r
ci(Est)
#>  loading    lower    upper
#>1       1 1.614269 4.514703
```

The true value $x_{\text{new}}^{\intercal}(\beta^{(2)} - \beta^{(1)}) = 2.6$ is included in the above 95% CI.

```r
ci(Est, probability = TRUE)
#>  loading     lower     upper
#>1       1 0.1531872 0.5086421
```

If we specify `probability` as `TRUE`, for the logistic regression, `ci()` yields the CI for $f(x_{\text{new}}^{\intercal}\beta^{(2)}) - f(x_{\text{new}}^{\intercal}\beta^{(1)})$ whose true value is 0.2423.

3.2 InnProd

The function `InnProd()`, shorthanded for Inner Product, conducts inference for $\beta_{\mathrm{G}}^{(1)\intercal} A \beta_{\mathrm{G}}^{(2)}$ if A is the submatrix pre-specified or $\beta_{\mathrm{G}}^{(1)\intercal} \Sigma_{\mathrm{G},\mathrm{G}} \beta_{\mathrm{G}}^{(2)}$ under the high-dimensional regression models. All arguments are similarly defined as previous functions.

```
InnProd(X1, y1, X2, y2, G, A = NULL, model=c("linear","logistic","logistic_alter"),
intercept=TRUE, beta.init1=NULL, beta.init2=NULL, split = TRUE,lambda=NULL,
mu=NULL, prob.filter=0.05, rescale=1.1, tau = c(0.25,0.5,1), alpha=0.05,
verbose=FALSE)
```

3.2.1 Code

Example 5 (InnProd for linear regression) *In the first group of data, the covariates $X_{i\cdot}^{(1)}$, for $1 \leq i \leq n_1$ with $n_1 = 200$, follows multivariate normal distribution with $\mu = 0_p$ and covariance $\Sigma = \mathbf{I}_p$; in the second group of data, the covariates $X_{i\cdot}^{(2)}$, for $1 \leq i \leq n_2$ with $n_2 = 260$, follows multivariate normal distribution with $\mu = 0_p$ and covariance $\Sigma \in \mathbb{R}^{p \times p}$ with $p = 120$ and $\Sigma_{j,k} = 0.5^{|j-k|}$. See the following code to see how the sparse $\beta^{(1)}$, $\beta^{(2)}$ are generated. We generate following the model $y_i^{(k)} = X_{i\cdot}^{(k)\intercal} \beta^{(k)} + \epsilon_i^{(k)}$ with standard normal error $\epsilon^{(k)}$ for $k = 1, 2$. Objective: We perform inference for $\beta_G^{(1)\intercal} \beta_G^{(2)} = \beta_G^{(1)\intercal} I_G \beta_G^{(2)}$ with $G = \{1, 2, \ldots, 20\}$ where I_G denotes the identity matrix of order $|G| \times |G|$.*

```r
# Generate Data
n1 = 200; n2 = 260; p = 120
mu1 = mu2 = rep(0,p)
Cov1 = diag(p)
Cov2 = matrix(0, p, p); for(j in 1:p) for(k in 1:p) Cov2[j,k] = 0.5^{abs(j-k)}
beta1 = rep(0, p); beta1[1:10] = 0.5
beta2 = rep(0, p); beta2[3:12] = 0.4
X1 = MASS::mvrnorm(n1,mu1,Cov1)
X2 = MASS::mvrnorm(n2,mu2,Cov2)
y1 = X1%*%beta1 + rnorm(n1)
y2 = X2%*%beta2 + rnorm(n2)

# Specify set G and matrix A
test.set = c(1:20)
A = diag(length(test.set))

# Run InnProd
Est = InnProd(X1, y1, X2, y2, G=test.set, A, model="linear")
```

Having fitted the model, it allows for method `ci()` and `summary()` as `QF()` does.

```r
ci(Est)
#>    tau     lower    upper
#> 1 0.25 0.7432061 2.490451
#> 2 0.50 0.7128181 2.520839
#> 3 1.00 0.6520422 2.581615
```

The true value $\beta_G^{(1)\intercal} \beta_G^{(2)} = 1.6$ is included in the above CIs with all default τ values.

3.3 Dist

The function `Dist()`, shorthanded for Distance, conducts inference for $\gamma_{\mathrm{G}}^{\intercal} A \gamma_{\mathrm{G}}$, where $\gamma = \beta^{(2)} - \beta^{(1)}$, if A is the submatrix pre-specified or $\gamma_{\mathrm{G}}^{\intercal} \Sigma_{\mathrm{G},\mathrm{G}} \gamma_{\mathrm{G}}$ under the high-dimensional regression models. All argument are similarly defined previously.

```
Dist(X1, y1, X2, y2, G, A = NULL, model=c("linear","logistic","logistic_alter"),
intercept=TRUE, beta.init1=NULL, beta.init2=NULL, split = TRUE, lambda=NULL,
mu=NULL, prob.filter=0.05, rescale=1.1, tau = c(0.25,0.50,1), alpha=0.05,
verbose=FALSE)
```

3.3.1 Code

Example 6 (Dist for linear regression) *In the first group of data, the covariates $X_{i\cdot}^{(1)}$, for $1 \leq i \leq n_1$ with $n_1 = 220$, follows multivariate normal distribution with $\mu = 0_p$ and covariance $\Sigma = \mathbf{I}_p$; in the second group of data, the covariates $X_{i\cdot}^{(2)}$, for $1 \leq i \leq n_2$ with $n_2 = 180$, follows multivariate normal distribution with $\mu = 0_p$ and covariance $\Sigma \in \mathbb{R}^{p\times p}$ with $p = 100$ and $\Sigma_{j,k} = 0.5^{|j-k|}$. We generate a sparse $\beta^{(1)}$ as $\beta_j^{(1)} = 0.2$ for $j = 1,2$ and $\beta_j^{(1)} = 0$ otherwise, whereas $\beta^{(2)}$ is generated as a relatively dense vector of coefficients : $\beta_1^{(2)} = 0.3, \beta_2^{(2)} = 1.5$ and $\beta_j^{(2)} = 0.08$ if $3 \leq j \leq 10$ and $\beta_j^{(2)} = 0$. otherwise. We generate following the model $y_i^{(k)} = X_{i\cdot}^{(k)\intercal}\beta^{(k)} + \epsilon_i^{(k)}$ with standard normal error $\epsilon^{(k)}$ for $k = 1,2$. Objective: we perform inference for $\gamma_G^{\intercal}\Sigma_{G,G}\gamma_G$, where $\gamma = \beta^{(2)} - \beta^{(1)}$ and $G = \{1,...,10\}$.*

```
# Generate Data
n1 = 220; n2 = 180; p = 100
mu = rep(0,p); Cov = diag(p)
beta1 = rep(0, p); beta1[1:2] = c(0.5, 1)
beta2 = rep(0, p); beta2[1:10] = c(0.3, 1.5, rep(0.08, 8))
X1 = MASS::mvrnorm(n1,mu,Cov)
X2 = MASS::mvrnorm(n2,mu,Cov)
y1 = X1%*%beta1 + rnorm(n1)
y2 = X2%*%beta2 + rnorm(n2)

# Specify set G
test.set = c(1:10)

# Run Dist
Est = Dist(X1, y1, X2, y2, G=test.set, A=NULL, model="linear", split=FALSE)
```

Having fitted the model, it allows for method `ci()` and `summary()` as `LF()` does.

```
ci(Est)
#>   tau    lower     upper
#>1 0.25 0.028202 0.6831165
#>2 0.50 0.000000 0.7196383
#>3 1.00 0.000000 0.7926819
```

```
summary(Est)
#> Call:
#> Inference for Distance
#>
#>  tau est.plugin est.debias Std. Error z value Pr(>|z|)
#> 0.25      0.4265     0.3557     0.1671   2.129  0.03327 *
#> 0.50      0.4265     0.3557     0.1857   1.915  0.05547 .
#> 1.00      0.4265     0.3557     0.2230   1.595  0.11070
```

The true value $\gamma_{\mathrm{G}}^{\intercal}\Sigma_{\mathrm{G,G}}\gamma_{\mathrm{G}} = 0.3412$. Similar to the previous instances, we note that the bias-corrected estimator effectively correct the bias of the plugin estimator. Depending on the τ values, we obtain various CIs, all of which encompass the true value. It is important to mention that in case of negative lower boundaries, they will be truncated at 0 for $\tau = 0.5$ and $\tau = 1$.

References

[1] T. Cai, T. Tony Cai, and Z. Guo. Optimal statistical inference for individualized treatment effects in high-dimensional models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 83(4):669–719, 2021.

[2] T. T. Cai and Z. Guo. Semisupervised inference for explained variance in high dimensional linear regression and its applications. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(2):391–419, 2020.

[3] T. T. Cai, Z. Guo, and R. Ma. Statistical inference for high-dimensional generalized linear models with binary outcomes. *Journal of the American Statistical Association*, pages 1–14, 2021.

[4] Z. Guo, W. Wang, T. T. Cai, and H. Li. Optimal estimation of genetic relatedness in high-dimensional linear models. *Journal of the American Statistical Association*, 114:358–369, 2019.

[5] Z. Guo, C. Renaux, P. Bühlmann, and T. Cai. Group inference in high dimensions with applications to hierarchical testing. *Electronic Journal of Statistics*, 15(2):6633–6676, 2021.

[6] Z. Guo, X. Li, L. Han, and T. Cai. Robust inference for federated meta-learning. *arXiv preprint arXiv:2301.00718*, 2023.

[7] A. Javanmard and A. Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *The Journal of Machine Learning Research*, 15(1):2869–2909, 2014.

[8] R. Ma, Z. Guo, T. T. Cai, and H. Li. Statistical inference for genetic relatedness based on high-dimensional logistic regression. *arXiv preprint arXiv:2202.10007*, 2022.

[9] S. van de Geer, P. Bühlmann, Y. Ritov, and R. Dezeure. On asymptotically optimal confidence regions and tests for high-dimensional models. *The Annals of Statistics*, 42:1166–1202, 2014.

[10] C.-H. Zhang and S. S. Zhang. Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1):217–242, 2014.