Estimating and Testing Direct Effects in Directed Acyclic Graphs using Estimating Equations

Stefan Konigorski

November 16, 2017

Overview

In any association study, it is important to distinguish direct and indirect effects in order to build truly functional models. For this purpose, we consider a directed acyclic graph (DAG) setting with an exposure variable, primary and intermediate outcome variables, and confounding factors. In order to make valid statistical inference on the direct effect of the exposure on the primary outcome, it is necessary to consider all potential effects in the graph, and we propose to use the estimating equation method with robust Huber-White sandwich standard errors. Then, a large-sample Wald-type test statistic is computed for testing the absence of the direct effect. In this package, the proposed causal inference method based on estimating equations (CIEE) is implemented for both the analysis of continuous and time-to-event primary outcomes subject to censoring for the model in Figure 1. Additionally, standard multiple regression, regression of residuals, and the structural equation approach are implemented for fitting the same model.

Results from simulation studies (Konigorski et al., 2017) showed that CIEE successfully removes the effect of intermediate outcomes from the primary outcome and is robust against measured and unmeasured confounding of the indirect effect through observed factors. Also, an application in a genetic association study in the same study showed that CIEE can identify genetic variants that would be missed by traditional regression methods. Both multiple regression methods and the structural equation method fail in some scenarios where their corresponding test statistics lead to inflated type I errors. An alternative approach for the analysis of continuous traits is the sequential G-estimation method (Vansteelandt et al., 2009).

In this package, CIEE is implemented for the model described in the DAG in Figure 1, which includes the direct effect α_{XY} of an exposure X on the primary outcome Y and an indirect effect of X on Y through a secondary outcome K. The model further includes measured and unmeasured factors L and U, respectively, which potentially confound the effect of K on Y. CIEE can also be applied to different models with different error distributions. The goal is to estimate and test the direct effect α_{XY}, while removing the indirect effect of X on Y through K, and with robustness against effects of L and U. Without restriction of generality, it is assumed that there aren't any factors affecting X and that any such factors are included as covariates in the analysis or have been dealt with using other approaches. Also, we generally assume that either $\alpha_{LY} = 0$ (L is a factor influencing K) or $\alpha_{XL} = 0$ (L is a measured confounder of $K \to Y$). Otherwise, the effect of L as intermediate outcome could be removed from Y in the analysis analogously to K.

Alternative approaches

Two traditional methods for the aim to estimate and test α_{XY} are (i) to include the intermediate outcomes and factors as covariates in a multiple regression (MR) model of the primary outcome on the exposure, or (ii) to first regress the primary outcome on the intermediate outcome and factors, and then regress the extracted residuals on the exposure (regression of residuals, RR). In more detail, estimates of α_{XY} are obtained from fitting the following models in the quantitative outcome setting for a normally-distributed Y (GLM setting):

MR: Obtain the least squares (LS) estimate of α_{XY} by fitting

$$Y_i = \alpha_0 + \alpha_{XY} x_i + \alpha_1 k_i + \alpha_2 l_i + \varepsilon_i,\ \varepsilon_i \sim N(0, \sigma_1^2)$$

.

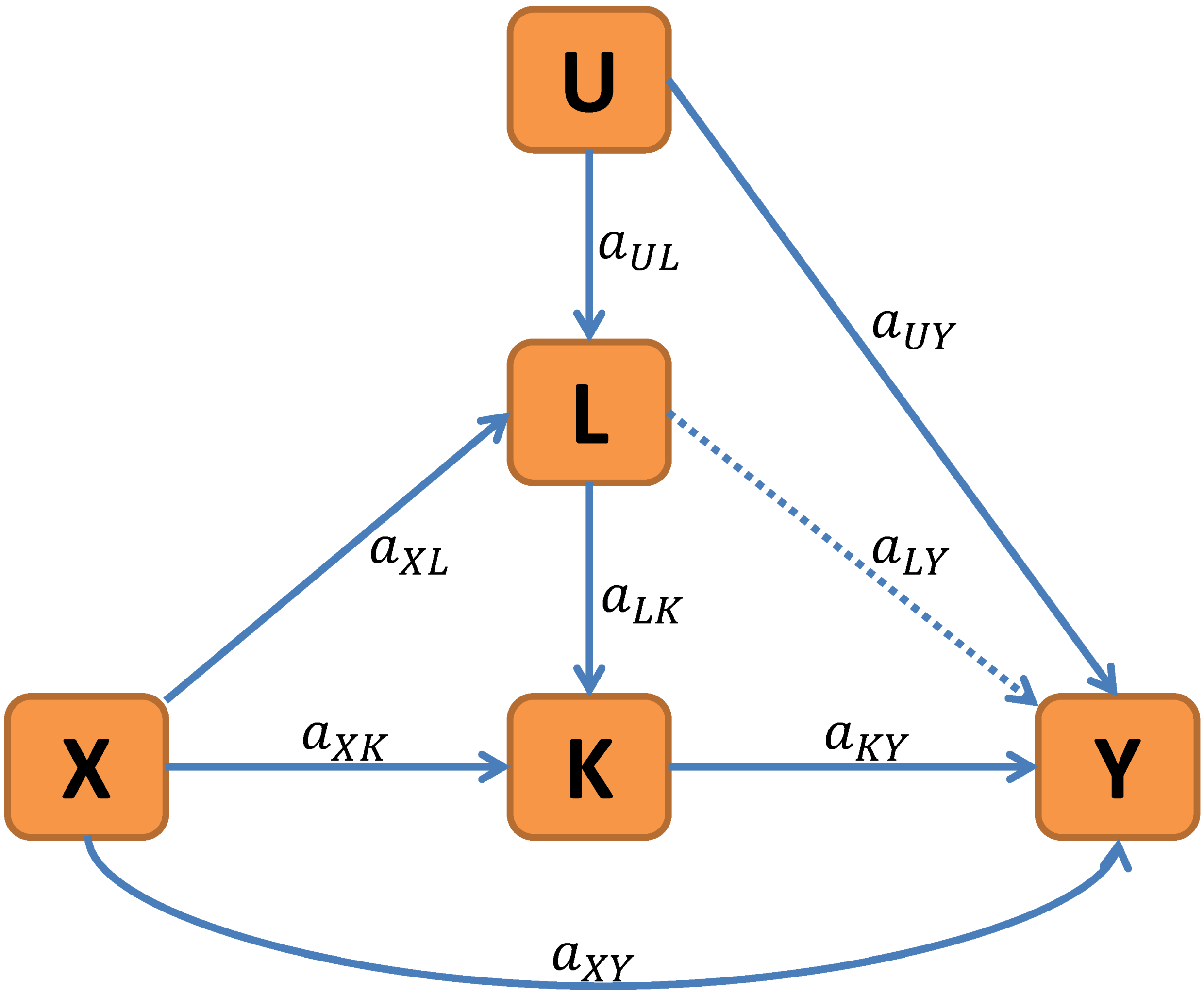

Figure 1: Overview of the underlying directed acyclic graph considered in this study. It is assumed that $\alpha_{LY} = 0$ so that L is a measured predictive factor of K, however, CIEE is also valid if L is a measured confounder of $K \rightarrow Y$ (i.e., $\alpha_{LY} \neq 0$ and $\alpha_{XL} = 0$).

RR: First, obtain residuals $\hat{\varepsilon}_{1i} = y_i - \hat{\alpha}_0 - \hat{\alpha}_1 k_i - \hat{\alpha}_2 l_i$ by fitting

$$Y_i = \alpha_0 + \alpha_1 k_i + \alpha_2 l_i + \varepsilon_{1i},\ \varepsilon_{1i} \sim N(0, \sigma_1^2)$$

using the LS estimation. Second, obtain the LS estimate of α_{XY} by fitting

$$\hat{\varepsilon}_{1i} = \alpha_3 + \alpha_{XY} x_i + \varepsilon_{2i},\ \varepsilon_{2i} \sim N(0, \sigma_2^2).$$

Then, $H_0 : \alpha_{XY} = 0$ versus $H_A : \alpha_{XY} \neq 0$ is tested using the default t-test in the `lm()` function in R. For the analysis of a censored time-to-event primary trait Y (accelerated failure time setting; AFT), only the MR approach is implemented. Here, the equivalent censored log-linear regression model is fitted using the `survreg()` function in the `survival` R package to obtain the maximum likelihood estimate of α_{XY}, and a Wald-type test is performed for testing the null hypothesis $H_0 : \alpha_{XY} = 0$. Both approaches are implemented in the functions `mult_reg()` and `res_reg()` and can be used as follows. For this illustration, data is first generated using the `generate_data()` function, which generates data for the quantitative outcomes Y and K, a genetic marker X (single nucleotide polymorphism, SNP, taking values 0, 1, 2) as exposure, and observed as well as unobserved confounders L, U.

```
dat <- generate_data(setting="GLM", n = 1000, maf = 0.2, cens = 0.3, a = NULL, b = NULL,
                     aUL = 0, aXL = 0, aXK = 0.2, aLK = 0, aUY = 0, aKY = 0.3, aXY = 0.1,
                     aLY = 0, mu_U = 0, sd_U = 1, X_orth_U = TRUE, mu_X = NULL,
                     sd_X = NULL, mu_L = 0, sd_L = 1, mu_K = 0, sd_K = 1, mu_Y = 0,
                     sd_Y = 1)
head(dat)
```

```
##            Y           K X          L           U
## 1  0.1553496 -0.97958391 0  0.6563416  0.37644631
## 2 -0.7246851 -1.54779872 1 -1.2680501  0.05540095
## 3  0.4126110  1.89214546 1  0.8662948 -1.24724658
## 4  0.2323169 -0.09882776 0 -0.5631225  0.82403408
## 5  0.8687194  0.75719547 0 -0.4151525 -0.47959382
## 6  0.6663823  0.72782520 0 -0.8464409 -0.72093759
```

```
mult_reg(setting = "GLM", Y = dat$Y, X = dat$X, K = dat$K, L = dat$L)
```

```
## $point_estimates
##     alpha_0     alpha_1    alpha_XY     alpha_2
## 0.005877705 0.348813792 0.147878962 0.006556433
##
## $SE_estimates
##    alpha_0    alpha_1   alpha_XY    alpha_2
## 0.03979066 0.03394673 0.05732815 0.03182939
##
## $pvalues
##      alpha_0      alpha_1     alpha_XY      alpha_2
## 8.825970e-01 1.307963e-23 1.003622e-02 8.368433e-01
```

```
res_reg(Y = dat$Y, X = dat$X, K = dat$K, L = dat$L)
```

```
## $point_estimates
##      alpha_0      alpha_1      alpha_2      alpha_3     alpha_XY
##  0.065253943  0.360133657  0.003673109 -0.058959447  0.145220313
##
## $SE_estimates
##    alpha_0    alpha_1    alpha_2    alpha_3   alpha_XY
## 0.03254883 0.03375720 0.03189982 0.03972985 0.05675693
##
```

```
## $pvalues
##      alpha_0      alpha_1      alpha_2      alpha_3     alpha_XY
## 4.525425e-02 3.096414e-25 9.083533e-01 1.381223e-01 1.065498e-02
```

As another approach for modeling DAGs, the structural equation modeling method (SEM; Bollen, 1989) can be used. Among others, it is implemented in the `sem()` function of the `lavaan` package (Rosseel, 2012). For a comparison of the results, the function `sem_appl()` applies the SEM method to the DAG in Figure 1 based on the following model equations:

$$L_i = \alpha_0 + \alpha_1 x_i + \varepsilon_{1i},\ \varepsilon_{1i} \sim N(0, \sigma_1^2)$$
$$K_i = \alpha_2 + \alpha_3 x_i + \alpha_2 l_i + \varepsilon_{2i},\ \varepsilon_{2i} \sim N(0, \sigma_2^2)$$
$$Y_i = \alpha_5 + \alpha_6 k_i + \alpha_{XY} x_i + \varepsilon_{3i},\ \varepsilon_{3i} \sim N(0, \sigma_3^2)$$

```
sem_appl(Y = dat$Y, X = dat$X, K = dat$K, L = dat$L)
```

```
## $point_estimates
##      alpha_1      alpha_3      alpha_4      alpha_6     alpha_XY
## -0.064117539  0.218310593 -0.003486751  0.348787788  0.147464274
##
## $SE_estimates
##    alpha_1    alpha_3    alpha_4    alpha_6   alpha_XY
## 0.05644210 0.05295543 0.02965018 0.03387925 0.05717930
##
## $pvalues
##      alpha_1      alpha_3      alpha_4      alpha_6     alpha_XY
## 2.559616e-01 3.747265e-05 9.063876e-01 7.419979e-25 9.909256e-03
```

Further proposed approaches for fitting the model in Figure 1 include the two-stage sequential G-estimation method (Vansteelandt et al., 2009). It first removes the effect of K from the primary outcome Y, and then tests the association of X with the adjusted primary outcome.

In more detail, for the analyis of a quantitative outcome Y with n independent observations, in the first stage, the effect of K on Y, α_1, is estimated and $\hat{\alpha}_1$ is obtained using the LS estimation method under the model

$$Y_i = \alpha_0 + \alpha_1 k_i + \alpha_2 x_i + \alpha_3 l_i + \varepsilon_i,\ \varepsilon_i \sim N(0, \sigma_1^2), \quad i = 1, \ldots, n \tag{1}$$

Then, to block all indirect paths from X to Y, the adjusted outcome $\tilde{Y}$ is obtained by removing the effect of K on Y with

$$\tilde{y}_i = y_i - \bar{y} - \hat{\alpha}_1 (k_i - \bar{k}) \tag{2}$$

where $\bar{y} = \sum_{i=1}^n y_i$ and $\bar{k} = \sum_{i=1}^n k_i$. In the second stage, the significance of the direct effect of X on Y, α_{XY}, is tested under the model

$$\tilde{Y}_i = \alpha_4 + \alpha_{XY} x_i + \varepsilon_i,\ \varepsilon_i \sim N(0, \sigma_2^2) \tag{3}$$

using the proposed test statistic in Vansteelandt and colleagues (2009). The approach is implemented in the `CGene` package, which can be obtained from cran.r-project.org/src/contrib/Archive/CGene/.

Causal inference using estimating equations (CIEE)

CIEE follows the general idea of the two-stage sequential G-estimation method with the major difference that the approach is one-stage and obtains coefficient estimates of all parameters simultaneously by solving estimating equations. This also allows building on existing asymptotic properties of the estimator and obtaining robust sandwich standard error estimates considering the additional variability of the estimates from the outcome adjustment.

In more detail, for the analyis of a quantitative outcome Y, we formulate unbiased estimating equations $U(\theta) = 0$ for a consistent estimation of the unknown parameter vector $\theta = (\alpha_0, \alpha_1, \alpha_2, \alpha_3, \sigma_1^2, \alpha_4, \alpha_{XY}, \sigma_2^2)^T$ where

$$U(\theta) = \left(\frac{\partial l_1(\theta)}{\partial \alpha_0}, \frac{\partial l_1(\theta)}{\partial \alpha_1}, \frac{\partial l_1(\theta)}{\partial \alpha_2}, \frac{\partial l_1(\theta)}{\partial \alpha_3}, \frac{\partial l_1(\theta)}{\partial \sigma_1^2}, \frac{\partial l_1(\theta)}{\partial \alpha_4}, \frac{\partial l_1(\theta)}{\partial \alpha_{XY}}, \frac{\partial l_1(\theta)}{\partial \sigma_2^2} \right)^T$$

with

$$l_1(\theta) = \sum_{i=1}^{n} \left[-log(\sigma_1) + log\left(\phi\left(\frac{y_i - \alpha_0 - \alpha_1 k_i - \alpha_2 x_i - \alpha_3 l_i}{\sigma_1} \right) \right) \right]$$

and

$$l_2(\theta) = \sum_{i=1}^{n} \left[-log(\sigma_2) + log\left(\phi\left(\frac{y_i - \bar{y} - \alpha_1 (k_i - \bar{k}) - \alpha_4 - \alpha_{XY} x_i}{\sigma_2} \right) \right) \right],$$

where ϕ is the probability density function of the standard normal distribution.

By solving the first five estimating equations based on $l_1(\theta)$, we are hence obtaining estimates of $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \sigma_1^2$. Analogously, solving the last three estimating equations based on $l_2(\theta)$ yields estimates of $\alpha_4, \alpha_{XY}, \sigma_2^2$. To give an intuition on how these estimating equations are obtained, $l_1(\theta)$ is the log-likelihood function under the model in (1) and $l_2(\theta)$ is the log-likelihood function under the model in (3) given that α_1 is known. All parameters in θ are estimated simultaneously and the additional variability obtained in the outcome adjustment in (2) is considered by using the robust Huber-White sandwich estimator of the standard error of $\hat{\theta}$. Then, the large sample Wald-type test statistic $\hat{\theta}/\widehat{SE}(\hat{\theta})$, which has an asymptotic standard normal distribution, is computed to test the absence of the exposure effect α_{XY}.

For the analysis of a censored time-to-event primary outcome Y, the estimating equations can be constructed as described above for a quantitative primary phenotype (for the same parameters except for σ_1 instead of σ_1^2), but in order to remove the effect of K from Y, the true underlying log survival times Y_{est} need to be estimated for censored survival times. For uncensored survival times, Y_{est} equals the observed log-survival time Y. Then, the estimating equations are constructed accordingly, the robust Huber-White sandwich estimator of the standard error is obtained and the large-sample Wald-type tests are computed. For more statistical details, see Konigorski et al. (2017).

In the implementation of CIEE in this package, the `est_funct_expr()` function contains the estimating equations as an expression.

```
estfunct <- est_funct_expr(setting="GLM")
estfunct
```

```
## $logL1
## expression(log((1/sqrt(sigma1sq)) * dnorm((y_i - alpha0 - alpha1 *
##     k_i - alpha2 * x_i - alpha3 * l_i)/sqrt(sigma1sq), mean = 0,
##     sd = 1)))
##
## $logL2
```

```
## expression(log((1/sqrt(sigma2sq)) * dnorm((y_i - y_bar - alpha1 *
##     (k_i - k_bar) - alpha4 - alphaXY * x_i)/sqrt(sigma2sq), mean = 0,
##     sd = 1)))
```

```
est_funct_expr(setting = "AFT")
```

```
## $logL1
## expression(-c_i * log(sigma1) + c_i * log(dnorm((y_i - alpha0 -
##     alpha1 * k_i - alpha2 * x_i - alpha3 * l_i)/sigma1, mean = 0,
##     sd = 1)) + (1 - c_i) * log(1 - pnorm((y_i - alpha0 - alpha1 *
##     k_i - alpha2 * x_i - alpha3 * l_i)/sigma1, mean = 0, sd = 1)))
##
## $logL2
## expression(log((1/sqrt(sigma2sq)) * dnorm(((c_i * y_i + (1 -
##     c_i) * ((alpha0 + alpha1 * k_i + alpha2 * x_i + alpha3 *
##     l_i) + (sigma1 * dnorm((y_i - alpha0 - alpha1 * k_i - alpha2 *
##     x_i - alpha3 * l_i)/sigma1, mean = 0, sd = 1)/(1 - pnorm((y_i -
##     alpha0 - alpha1 * k_i - alpha2 * x_i - alpha3 * l_i)/sigma1,
##     mean = 0, sd = 1))))) - y_adj_bar - alpha1 * (k_i - k_bar) -
##     alpha4 - alphaXY * x_i)/sqrt(sigma2sq), mean = 0, sd = 1)))
```

The function `get_estimates()` obtains estimates of the parameters in the models (1)-(3) by using the `lm()` and `survreg()` functions for computational purposes. The estimates are identical to estimates obtained by solving the estimating equations.

```
estimates <- get_estimates(setting = "GLM", Y = dat$Y, X = dat$X, K = dat$K, L = dat$L)
estimates
```

```
##      alpha_0      alpha_1      alpha_2      alpha_3   sigma_1_sq
##  0.005877705  0.348813792  0.147878962  0.006556433  1.045246179
##      alpha_4     alpha_XY   sigma_2_sq
## -0.059868183  0.147458580  1.045290708
```

In order to compute the robust Huber-White sandwich estimator of the parameters, in a first step, the `deriv_obj()` function computes the expression of all first and second derivatives for the 8 parameters $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \sigma_1^2, \alpha_4, \alpha_{XY}, \sigma_2^2$ by using the expressions from the `est_funct_expr()` function as input. Then, the numerical values of all first and second derivatives are obtained for the observed data and parameter point estimates for all observed individuals, using the `scores()` and `hessian()` functions.

```
derivobj <- deriv_obj(setting = "GLM", logL1 = estfunct$logL1, logL2 = estfunct$logL2,
                      Y = dat$Y, X = dat$X, K = dat$K, L = dat$L, estimates = estimates)
names(derivobj)
```

```
## [1] "logL1_deriv" "logL2_deriv"
```

```
head(derivobj$logL1_deriv$gradient)
```

```
##          alpha0      alpha1     alpha2     alpha3   sigma1sq alpha4
## [1,]  0.4657860 -0.45627643  0.0000000  0.3057147 -0.3698779      0
## [2,] -0.3159393  0.48901039 -0.3159393  0.4006268 -0.4284474      0
## [3,] -0.3892211 -0.73646292 -0.3892211 -0.3371802 -0.4026097      0
## [4,]  0.2531497 -0.02501822  0.0000000 -0.1425543 -0.4463138      0
## [5,]  0.5754084  0.43569660  0.0000000 -0.2388822 -0.3128088      0
## [6,]  0.3943365  0.28700806  0.0000000 -0.3337825 -0.4006056      0
##      alphaXY sigma2sq
## [1,]       0        0
## [2,]       0        0
## [3,]       0        0
```

```
## [4,]       0        0
## [5,]       0        0
## [6,]       0        0
```

```r
score_matrix <- scores(derivobj)
head(score_matrix)
```

```
##          alpha0      alpha1     alpha2     alpha3   sigma1sq     alpha4
## [1,]  0.4657860 -0.45627643  0.0000000  0.3057147 -0.3698779  0.4699406
## [2,] -0.3159393  0.48901039 -0.3159393  0.4006268 -0.4284474 -0.3234196
## [3,] -0.3892211 -0.73646292 -0.3892211 -0.3371802 -0.4026097 -0.3833109
## [4,]  0.2531497 -0.02501822  0.0000000 -0.1425543 -0.4463138  0.2496645
## [5,]  0.5754084  0.43569660  0.0000000 -0.2388822 -0.3128088  0.5728376
## [6,]  0.3943365  0.28700806  0.0000000 -0.3337825 -0.4006056  0.3890683
##         alphaXY   sigma2sq
## [1,]  0.0000000 -0.3679137
## [2,] -0.3234196 -0.4260357
## [3,] -0.3833109 -0.4048722
## [4,]  0.0000000 -0.4471696
## [5,]  0.0000000 -0.3142644
## [6,]  0.0000000 -0.4026488
```

```r
hessian_matrix <- hessian(derivobj)
str(hessian_matrix)
```

```
##  num [1:1000, 1:8, 1:8] -0.957 -0.957 -0.957 -0.957 -0.957 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ : NULL
##   ..$ : chr [1:8] "alpha0" "alpha1" "alpha2" "alpha3" ...
##   ..$ : chr [1:8] "alpha0" "alpha1" "alpha2" "alpha3" ...
```

The robust Huber-White sandwich estimator of the standard error can then be obtained using the `sandwich_se()` function:

```r
sandwich_se(scores = score_matrix, hessian = hessian_matrix)
```

```
##    alpha_0    alpha_1    alpha_2    alpha_3 sigma_1_sq    alpha_4
## 0.03983255 0.03343622 0.05997271 0.03237697 0.04962314 0.03991663
##   alpha_XY sigma_2_sq
## 0.05989090 0.04964436
```

Alternatively, bootstrap standard error estimates can be computed using the `bootstrap_se()` function. Also, for comparison, the function `naive_se()` computes naive standard error estimates of the parameter estimates of $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_{XY}$ without accounting for the additional variability due to the two stages in the model in (1)-(3):

```r
bootstrap_se(setting = "GLM", BS_rep = 1000, Y = dat$Y, X = dat$X, K = dat$K, L = dat$L)
```

```
##    alpha_0    alpha_1    alpha_2    alpha_3 sigma_1_sq    alpha_4
## 0.04018567 0.03442461 0.06191941 0.03255589 0.04795089 0.02519520
##   alpha_XY sigma_2_sq
## 0.06171368 0.04803157
```

```r
naive_se(setting = "GLM", Y = dat$Y, X = dat$X, K = dat$K, L = dat$L)
```

```
##    alpha_0    alpha_1    alpha_2    alpha_3 sigma_1_sq    alpha_4
## 0.03979066 0.03394673 0.05732815 0.03182939         NA 0.03972831
##   alpha_XY sigma_2_sq
## 0.05675473         NA
```

Finally, the functions `ciee()` and `ciee_loop()` allow an easy integrated use of all above functions and a simultaneous computation of the estimating equations approach using either standard error computation, the traditional regression-based approaches, and the SEM method. `ciee()` fits the model in equations (1)-(3) (e.g. the model in Figure 1) and yields parameter estimates, standard error estimates, and p-values for all parameters. `ciee_loop()` provides an extension of `ciee()` and allows the input of multiple exposure variables (e.g. multiple SNPs) which are tested sequentially. In the output of `ciee_loop()`, only the coefficient estimates, standard error estimates, and p-values with respect to the direct effect α_{XY} are provided.

```
results_ciee <- ciee(setting = "GLM", Y = dat$Y, X = dat$X, K = dat$K, L = dat$L,
                     estimates = c("ee", "mult_reg", "res_reg", "sem"),
                     ee_se = "sandwich")
results_ciee
```

```
## $results_ee
## $results_ee$point_estimates
##      alpha_0      alpha_1      alpha_2      alpha_3   sigma_1_sq
##  0.005877705  0.348813792  0.147878962  0.006556433  1.045246179
##      alpha_4     alpha_XY   sigma_2_sq
## -0.059868183  0.147458580  1.045290708
##
## $results_ee$SE_estimates
##    alpha_0    alpha_1    alpha_2    alpha_3 sigma_1_sq    alpha_4
## 0.03983255 0.03343622 0.05997271 0.03237697 0.04962314 0.03991663
##   alpha_XY sigma_2_sq
## 0.05989090 0.04964436
##
## $results_ee$wald_test_stat
##    alpha_0    alpha_1    alpha_2    alpha_3 sigma_1_sq    alpha_4
##  0.1475604 10.4322133  2.4657710  0.2025030 21.0636836 -1.4998306
##   alpha_XY sigma_2_sq
##  2.4621200 21.0555800
##
## $results_ee$pvalues
##      alpha_0      alpha_1      alpha_2      alpha_3   sigma_1_sq
## 8.826897e-01 1.767231e-25 1.367187e-02 8.395235e-01 1.713215e-98
##      alpha_4     alpha_XY   sigma_2_sq
## 1.336583e-01 1.381184e-02 2.032800e-98
##
##
## $results_mult_reg
## $results_mult_reg$point_estimates
##     alpha_0     alpha_1    alpha_XY     alpha_2
## 0.005877705 0.348813792 0.147878962 0.006556433
##
## $results_mult_reg$SE_estimates
##    alpha_0    alpha_1   alpha_XY    alpha_2
## 0.03979066 0.03394673 0.05732815 0.03182939
##
## $results_mult_reg$pvalues
##      alpha_0      alpha_1     alpha_XY      alpha_2
## 8.825970e-01 1.307963e-23 1.003622e-02 8.368433e-01
##
##
## $results_res_reg
## $results_res_reg$point_estimates
```

```
##      alpha_0      alpha_1      alpha_2      alpha_3     alpha_XY
##  0.065253943  0.360133657  0.003673109 -0.058959447  0.145220313
##
## $results_res_reg$SE_estimates
##    alpha_0    alpha_1    alpha_2    alpha_3   alpha_XY
## 0.03254883 0.03375720 0.03189982 0.03972985 0.05675693
##
## $results_res_reg$pvalues
##      alpha_0      alpha_1      alpha_2      alpha_3     alpha_XY
## 4.525425e-02 3.096414e-25 9.083533e-01 1.381223e-01 1.065498e-02
##
##
## $results_sem
## $results_sem$point_estimates
##      alpha_1      alpha_3      alpha_4      alpha_6     alpha_XY
## -0.064117539  0.218310593 -0.003486751  0.348787788  0.147464274
##
## $results_sem$SE_estimates
##    alpha_1    alpha_3    alpha_4    alpha_6   alpha_XY
## 0.05644210 0.05295543 0.02965018 0.03387925 0.05717930
##
## $results_sem$pvalues
##      alpha_1      alpha_3      alpha_4      alpha_6     alpha_XY
## 2.559616e-01 3.747265e-05 9.063876e-01 7.419979e-25 9.909256e-03
##
##
## attr(,"class")
## [1] "ciee"
```

```
maf <- 0.2
n <- 1000
dat <- generate_data(n = n, maf = maf)
datX <- data.frame(X = dat$X)
names(datX)[1] <- "X1"
for(i in 2:10){
 X <- rbinom(n, size = 2, prob = maf)
 datX$X <- X
 names(datX)[i] <- paste("X", i, sep="")
}
results_ciee_loop <- ciee_loop(setting = "GLM", Y = dat$Y, X = datX, K = dat$K, L = dat$L)
results_ciee_loop
```

```
## $results_ee
## $results_ee$point_estimates
##           X1           X2           X3           X4           X5
##  0.108620032 -0.060209184  0.010169822 -0.088623977 -0.042123830
##           X6           X7           X8           X9          X10
##  0.008428589  0.056204387  0.059120855 -0.012902566  0.029948568
##
## $results_ee$SE_estimates
##         X1         X2         X3         X4         X5         X6
## 0.05686331 0.05263118 0.05357397 0.05332807 0.05612258 0.05491737
##         X7         X8         X9        X10
## 0.05548012 0.05396784 0.05575429 0.05500532
##
```

```
## $results_ee$wald_test_stat
##         X1         X2         X3         X4         X5         X6
##  1.9101954 -1.1439831  0.1898277 -1.6618635 -0.7505683  0.1534777
##         X7         X8         X9        X10
##  1.0130546  1.0954830 -0.2314183  0.5444668
##
## $results_ee$pvalues
##         X1         X2         X3         X4         X5         X6
## 0.05610805 0.25263065 0.84944419 0.09654015 0.45291249 0.87802161
##         X7         X8         X9        X10
## 0.31103409 0.27330508 0.81698982 0.58612032
##
##
## $results_mult_reg
## $results_mult_reg$point_estimates
##           X1           X2           X3           X4           X5
##  0.099126535 -0.052483926  0.016487340 -0.097757749 -0.038825636
##           X6           X7           X8           X9          X10
##  0.007790898  0.067052006  0.062702918 -0.011320810  0.030489092
##
## $results_mult_reg$SE_estimates
##         X1         X2         X3         X4         X5         X6
## 0.05681687 0.05453296 0.05533832 0.05453254 0.05503983 0.05552789
##         X7         X8         X9        X10
## 0.05807012 0.05614157 0.05690424 0.05649471
##
## $results_mult_reg$pvalues
##         X1         X2         X3         X4         X5         X6
## 0.08135124 0.33606934 0.76581328 0.07333233 0.48072018 0.88844656
##         X7         X8         X9        X10
## 0.24850129 0.26431861 0.84234644 0.58953805
##
##
## $results_res_reg
## $results_res_reg$point_estimates
##           X1           X2           X3           X4           X5
##  0.098222692 -0.052302980  0.016455241 -0.097374065 -0.038773602
##           X6           X7           X8           X9          X10
##  0.007776953  0.066610759  0.062634996 -0.011313192  0.030488208
##
## $results_res_reg$SE_estimates
##         X1         X2         X3         X4         X5         X6
## 0.05650133 0.05438438 0.05522901 0.05437121 0.05494781 0.05542256
##         X7         X8         X9        X10
## 0.05782097 0.05605494 0.05682807 0.05643726
##
## $results_res_reg$pvalues
##         X1         X2         X3         X4         X5         X6
## 0.08244635 0.33641924 0.76580668 0.07361032 0.48057415 0.88843462
##         X7         X8         X9        X10
## 0.24958983 0.26409829 0.84224264 0.58916985
##
##
## $results_sem
```

```
## $results_sem$point_estimates
##           X1           X2           X3           X4           X5
##  0.108717027 -0.060196981  0.010170457 -0.088628956 -0.042102686
##           X6           X7           X8           X9          X10
##  0.008465442  0.056223524  0.059138097 -0.012923965  0.029951931
##
## $results_sem$SE_estimates
##         X1         X2         X3         X4         X5         X6
## 0.05673975 0.05451092 0.05535891 0.05451431 0.05509520 0.05560043
##         X7         X8         X9        X10
## 0.05801453 0.05620300 0.05697556 0.05656920
##
## $results_sem$pvalues
##         X1         X2         X3         X4         X5         X6
## 0.05535688 0.26945846 0.85423429 0.10399384 0.44475962 0.87898583
##         X7         X8         X9        X10
## 0.33248114 0.29269715 0.82055324 0.59647652
##
##
## attr(,"class")
## [1] "ciee"
```

Both `ciee()` and `ciee_loop()` return `ciee` objects as output, and the implemented `summary.ciee()` function can be used through the generic `summary()` to provide a reader-friendly formatted output of the results.

```r
summary(results_ciee)
```

```
## [1] "Results based on estimating equations."
##                 point_estimates SE_estimates wald_test_stat      pvalues
## CIEE_alpha_0        0.005877705   0.03983255      0.1475604 8.826897e-01
## CIEE_alpha_1        0.348813792   0.03343622     10.4322133 1.767231e-25
## CIEE_alpha_2        0.147878962   0.05997271      2.4657710 1.367187e-02
## CIEE_alpha_3        0.006556433   0.03237697      0.2025030 8.395235e-01
## CIEE_sigma_1_sq     1.045246179   0.04962314     21.0636836 1.713215e-98
## CIEE_alpha_4       -0.059868183   0.03991663     -1.4998306 1.336583e-01
## CIEE_alpha_XY       0.147458580   0.05989090      2.4621200 1.381184e-02
## CIEE_sigma_2_sq     1.045290708   0.04964436     21.0555800 2.032800e-98
## [1] "Results based on traditional multiple regression."
##             point_estimates SE_estimates      pvalues
## MR_alpha_0      0.005877705   0.03979066 8.825970e-01
## MR_alpha_1      0.348813792   0.03394673 1.307963e-23
## MR_alpha_XY     0.147878962   0.05732815 1.003622e-02
## MR_alpha_2      0.006556433   0.03182939 8.368433e-01
## [1] "Results based on traditional regression of residuals."
##             point_estimates SE_estimates      pvalues
## RR_alpha_0      0.065253943   0.03254883 4.525425e-02
## RR_alpha_1      0.360133657   0.03375720 3.096414e-25
## RR_alpha_2      0.003673109   0.03189982 9.083533e-01
## RR_alpha_3     -0.058959447   0.03972985 1.381223e-01
## RR_alpha_XY     0.145220313   0.05675693 1.065498e-02
## [1] "Results based on structural equation modeling."
##              point_estimates SE_estimates      pvalues
## SEM_alpha_1     -0.064117539   0.05644210 2.559616e-01
## SEM_alpha_3      0.218310593   0.05295543 3.747265e-05
## SEM_alpha_4     -0.003486751   0.02965018 9.063876e-01
```

```
## SEM_alpha_6      0.348787788   0.03387925 7.419979e-25
## SEM_alpha_XY     0.147464274   0.05717930 9.909256e-03
```

```r
summary(results_ciee_loop)
```

```
## [1] "Results based on estimating equations."
##           point_estimates SE_estimates wald_test_stat    pvalues
## CIEE_X1       0.108620032   0.05686331      1.9101954 0.05610805
## CIEE_X2      -0.060209184   0.05263118     -1.1439831 0.25263065
## CIEE_X3       0.010169822   0.05357397      0.1898277 0.84944419
## CIEE_X4      -0.088623977   0.05332807     -1.6618635 0.09654015
## CIEE_X5      -0.042123830   0.05612258     -0.7505683 0.45291249
## CIEE_X6       0.008428589   0.05491737      0.1534777 0.87802161
## CIEE_X7       0.056204387   0.05548012      1.0130546 0.31103409
## CIEE_X8       0.059120855   0.05396784      1.0954830 0.27330508
## CIEE_X9      -0.012902566   0.05575429     -0.2314183 0.81698982
## CIEE_X10      0.029948568   0.05500532      0.5444668 0.58612032
## [1] "Results based on traditional multiple regression."
##         point_estimates SE_estimates    pvalues
## MR_X1       0.099126535   0.05681687 0.08135124
## MR_X2      -0.052483926   0.05453296 0.33606934
## MR_X3       0.016487340   0.05533832 0.76581328
## MR_X4      -0.097757749   0.05453254 0.07333233
## MR_X5      -0.038825636   0.05503983 0.48072018
## MR_X6       0.007790898   0.05552789 0.88844656
## MR_X7       0.067052006   0.05807012 0.24850129
## MR_X8       0.062702918   0.05614157 0.26431861
## MR_X9      -0.011320810   0.05690424 0.84234644
## MR_X10      0.030489092   0.05649471 0.58953805
## [1] "Results based on traditional regression of residuals."
##         point_estimates SE_estimates    pvalues
## RR_X1       0.098222692   0.05650133 0.08244635
## RR_X2      -0.052302980   0.05438438 0.33641924
## RR_X3       0.016455241   0.05522901 0.76580668
## RR_X4      -0.097374065   0.05437121 0.07361032
## RR_X5      -0.038773602   0.05494781 0.48057415
## RR_X6       0.007776953   0.05542256 0.88843462
## RR_X7       0.066610759   0.05782097 0.24958983
## RR_X8       0.062634996   0.05605494 0.26409829
## RR_X9      -0.011313192   0.05682807 0.84224264
## RR_X10      0.030488208   0.05643726 0.58916985
## [1] "Results based on structural equation modeling."
##          point_estimates SE_estimates    pvalues
## SEM_X1       0.108717027   0.05673975 0.05535688
## SEM_X2      -0.060196981   0.05451092 0.26945846
## SEM_X3       0.010170457   0.05535891 0.85423429
## SEM_X4      -0.088628956   0.05451431 0.10399384
## SEM_X5      -0.042102686   0.05509520 0.44475962
## SEM_X6       0.008465442   0.05560043 0.87898583
## SEM_X7       0.056223524   0.05801453 0.33248114
## SEM_X8       0.059138097   0.05620300 0.29269715
## SEM_X9      -0.012923965   0.05697556 0.82055324
## SEM_X10      0.029951931   0.05656920 0.59647652
```

References

Bollen KA (1989). Structural equations with latent variables. New York: John Wiley & Sons.

Konigorski S, Wang Y, Cigsar C, Yilmaz YE (2017). Estimating and testing direct genetic effects in directed acyclic graphs using estimating equations. Genetic Epidemiology, in press.

Rosseel Y (2012). lavaan: an R package for structural equation modeling. Journal of Statistical Software, 48(2), 1–36.

Vansteelandt S, Goetgeluk S, Lutz S, et al. (2009). On the adjustment for covariates in genetic association analysis: a novel, simple principle to infer direct causal effects. Genetic Epidemiology, 33, 394-405.