Effect estimation in latent subgroups

Lars Lindhagen

October 26, 2023

1 Introduction

This vignette gives a brief description of the package `latent`, with examples of how to use it. Below follows a short summary of the setting, for details, we refer to [1] and the references therein.

Consider a randomized trial, where patients are randomized to a screening procedure for a certain, treatable, condition. If the condition is detected, appropriate treatment is given; otherwise no action is taken. An example is the HELP-MI trial [2], where myocardial infarction patients are randomized to screening for Helicobacter pylori (HP) with subsequent eradication to reduce the risk of gastro-intestinal bleedings due to anticoagulant treatment.

A natural scientific question in such a trial is the treatment effect among the patients who could actually receive the treatment in question. In the HELP-MI trial, this would be the HP-infected patients. We shall refer to this subgroup as the *treatable* subgroup.

The problem is that this subgroup is not identified in the control arm, it is latent. It is, however, perfectly possible to perform maximum likelihood (ML) estimation of this effect, and the current package is intened to help the user carry out such estimation in a variety of settings.

1.1 Notation

Let G be an indicator for membership of the latent subgroup: $G = 1$ for treatable patients, and $G = 0$ for untreatable ones. Moreover, R is the randomization indicator: $R = 1$ for the screening arm and $R = 0$ for controls. Finally, x is a vector of pre-randomization covariates, whereas Y denotes the outcome, recorded after randomization.

1.2 The framework

The methods of the package rests on three statistical ("plug-in") models, specified by the user. The core of the package then performs ML estimation of the treatment effect in the latent subgroup, relying on the EM algorithm. The three plug-in models to be specified are the following:

1. A subgroup model, modeling membership of the latent subgroup.
2. An outcome model for untreatable patients ($G = 0$).

3. An outcome model for treatable patients ($G = 1$).

The subgroup model has a dichotomous outcome G, whereas the outcome for the other models is whatever outcome Y is recorded in the trial It can for example be numerical, dichotmous or a time to event outcome. All models can use any number of covariates as predictors of treatability or outcome to increase the precision. However, the model for treatable patients will always contain randomization, which the model for untreatable patients will not.

2 How to use the package

We shall illustrate the usage of the package by simulating and analyzing data in a variety of settings. All simulations will use correctly specified models. In other words, we will generate data from the same models as we later use to analyze them. Some example of the performance of the framework under misspecified models can be found in [1].

2.1 A basic example

As a first basic example, consider a numerical outcome without any covariates. We model subgroup membership using logistic regression, and the outcomes using linear regression. In the absence of covariates, these models simply read

$$\begin{cases} \operatorname{logit} P(G = 1) = \alpha_S & \\ \qquad\qquad Y = \alpha_0 + \epsilon, & G = 0, \\ \qquad\qquad Y = \alpha_1 + \psi R + \epsilon, & G = 1, \end{cases} \tag{1}$$

where $\operatorname{logit} p = p/(1-p)$, and ϵ are centred normal residuals with standard deviations σ_0 and σ_1. (Since the two outcome models are completely separate, we allow for different residual standard deviations.) The sought treatment effect is ψ.

As for the parameter values, we follow one of the simulations in [1]. Thus, we shall generate data on $n = 3000$ patients, with a treatability prevalence of 25%. Hence, $\alpha_S = \operatorname{logit} 0.25 = -1.099$. Moreover, we let $\alpha_0 = 100$, $\alpha_1 = 100.4$, $\sigma_0 = 2$, and $\sigma_1 = 2.5$. Finally, the treatmant effect is $\psi = -0.4$, making the treatable patients similar to the untreatable ones provided that they receive treatment.

We begin with some setup, including loading the `latent` package:

```
library(latent)
set.seed(12345) # For repeatability.
n <- 3000 # Number of patients.
```

Next, we create the "true" data, containing information on the latent subgroup.

```r
df_true <- data.frame(
    g = rbinom(n, size = 1, prob = 0.25), # Latent subgroup (still observed).
    rand = rbinom(n, size = 1, prob = 0.5)) # Randomization.
# Outcome.
df_true$y <- ifelse(df_true$g == 0,
    rnorm(n, mean = 100, sd = 2),
    rnorm(n, mean = 100.4 - 0.4 * df_true$rand, sd = 2.5))
head(df_true)
```

```
##   g rand         y
## 1 0    0 101.01978
## 2 1    1  97.43575
## 3 1    0  98.13143
## 4 1    1 101.93652
## 5 0    1 102.20176
## 6 0    1 101.91475
```

In reality, the subgroup G is only observed in the screening arm ($R = 1$). So we remove the information on it in the control arm, giving us the analysis database that we would actually observe in the trial:

```r
df_obs <- df_true
df_obs$g <- ifelse(df_obs$rand == 1, df_obs$g, NA)
head(df_obs)
```

```
##    g rand         y
## 1 NA    0 101.01978
## 2  1    1  97.43575
## 3 NA    0  98.13143
## 4  1    1 101.93652
## 5  0    1 102.20176
## 6  0    1 101.91475
```

The subgroup is now missing whenever $R = 0$.

We now set up the three plug-in models required by the framework (1). We begin with the logistic regression model for subgroup membership:

```r
modS <- latent_glm(
    formula = (g ~ 1),
    family_name = "binomial",
    link = "logit",
    coef_names = "alphaS")
```

Note that the outcome of this model is membership of the subgroup G, coded as the variable g.

We have not added any covariates, hence the trivial analysis formula `g ~ 1`. We conveniently choose to denote the single regression coefficient α_S by `alphaS`. The residual variance σ^2 will automatically be named `sigma2`.

Next, we add the two linear models for the outcome.

```
mod0 <- latent_linear(
    formula = (y ~ 1),
    coef_names = "alpha0")
mod1 <- latent_linear(
    formula = (y ~ rand),
    coef_names = c("alpha1", "psi"))
```

The first model again has a trivial analysis formula with a sole intercept, but the outcome is now the study outcome `y`. In the second model, however, the outcome depends on randomization, resulting in an additional coefficient `psi`, corresponding to the treatment effect ψ.

We are now ready to perform the analysis. This done by simply passing the three models, together with the data, to the package main function:

```
res <- latent_main(
    data = df_obs,
    modelS = modS,
    model0 = mod0,
    model1 = mod1)
res
```

```
## $theta
## thetaS.alphaS theta0.alpha0 theta0.sigma2 theta1.alpha1    theta1.psi
##    -1.0979198    99.9997091     3.8525897   100.5389571    -0.5709809
## theta1.sigma2
##     6.9481484
##
## $info
##              thetaS     theta0              theta1
## thetaS 2.926947e+02  18.928767 16.914480 10.459217 7.140556e-16  3.457487e+00
## theta0 1.892877e+01 480.011470 -4.281616 56.147593 0.000000e+00 -3.707370e+00
##        1.691448e+01  -4.281616 56.429811  1.057308 0.000000e+00  5.275002e+00
## theta1 1.045922e+01  56.147593  1.057308 76.053344 5.296375e+01  1.789607e+00
##        7.140556e-16   0.000000  0.000000 52.963751 5.296375e+01  2.036024e-12
##        3.457487e+00  -3.707370  5.275002  1.789607 2.036024e-12  5.889538e+00
```

The result is a list with two elements: `theta` and `info`. The former is the vector $\theta = (\vartheta_S, \vartheta_0, \vartheta_1)$ of all model parameters of the three statistical models, and the latter is the corresponding Fisher information matrix.

A word on the model parameters: The first one, ϑ_S (denoted `thetaS` by the package) consists of all parameters of the subgroup model. In the current example, this is just the intercept α_S. Similarly, ϑ_0 and ϑ_1 are the model parameters of the two outcome models. The former has two elements: $\vartheta_0 = (\alpha_0, \sigma_0^2)$, whereas the latter has three: $\vartheta_1 = (\alpha_1, \psi, \sigma_1^2)$. This gives a totality of six model parameters in θ.

The Fisher information matrix is the negative Hessian of the log-likelihood with respect to θ. In our example, this is a 6×6 matrix. The inverse of this is the covariance matrix, whose diagonal elements are the squared standard errors of the parameters in θ.

The parameter names in R consist of two parts. The first part tells us to which model the parameter refers (e.g. `thetaS` for the subgroup model with parameters ϑ_S). This is selected automatically by the package. The second part (e.g. `alphaS`) was chosen by ourselves when we created the model objects.

From the output, we can read that the estimate of α_S is -1.098 (`thetaS.alphaS`), corresponding to a subgroup prevalece of $e^{-1.098}/(1 + e^{-1.098}) = 0.250$. This is more or less identical to the true prevalance of 25%, which is not too surprising, considering that the prevalence is directly observable in the screening arm. Adding the control arm, where it is not, only improves things somewhat. Similarly, we can see that the residual variance in the control arm (`theta0.sigma2`) is estimated to 3.853, corresponding to a standard deviation of $\sqrt{3.853} = 1.963$, close to the true value $\sigma_0 = 2$.

But the most important parameter, representing the treatment effect in the latent subgroup, is ψ in ϑ_1, denoted `theta1.psi`. We see that the point estimate of this is $\hat{\psi} = -0.571$ (recall that the true value is -0.4). An asymptotic 95% Wald confidence interval for this can be computed as $\hat{\psi} \pm 1.96 \times \text{SE}$, where the standard error SE is found by inverting the Fisher information matrix:

```
se <- sqrt(diag(solve(res$info)))
se

##     thetaS     theta0                theta1
## 0.05945467 0.05486988 0.14002378 0.25396760 0.28875661 0.44247332
```

So the standard error of $\hat{\psi}$ is 0.289 and the confidence interval is $-0.571 \pm 1.96 \times 0.289 = (-1.137, -0.005)$ A p-value for the null hypothesis of no treatment effect ($\psi = 0$) can be computed using a normal approximation for the test statistic $z = \hat{\psi}/\text{SE} = -1.98$, giving $p = 0.048$.

2.2 Other types of outcome

The framework of the package is very flexible. For example, different outcomes can be handled by modifying the outcome models. We shall illustrate this possibility by analyzing dichotomous, ordinal, and time to event data. We shall, however, model subgroup membership in the same way. Therefore, the subgroup model need not be modified.

2.2.1 Dichotomous outcomes

To exemplify dichotomous outcomes, we shall use logistic regression:

$$\begin{cases} \operatorname{logit} P(Y=1) = \alpha_0, & G=0, \\ \operatorname{logit} P(Y=1) = \alpha_1 + \psi R, & G=1. \end{cases}$$

We set $\alpha_0 = \operatorname{logit} 0.2$ and $\alpha_1 = \alpha_0 + 0.4$, giving event rates of 20% and 27.16% for $G = 0$ and $G = 1$, respectively. With $\psi = -0.4$, this again means that treatable patients become similar to untreatable ones if given treatment.

Data can now be generated as follows:

```
set.seed(12345)
df_true <- data.frame(
    g = rbinom(n, size = 1, prob = 0.25),
    rand = rbinom(n = n, size = 1, prob = 0.5))
df_true$y <- ifelse((df_true$g == 0) | (df_true$rand == 1),
    rbinom(n, size = 1, prob = 0.2),
    rbinom(n, size = 1, prob = 0.2716446))
# Make subgroup latent.
df_obs <- df_true
df_obs$g <- ifelse(df_obs$rand == 1, df_obs$g, NA)
head(df_obs)

##    g rand y
## 1 NA    0 0
## 2  1    1 0
## 3 NA    0 0
## 4  1    1 0
## 5  0    1 0
## 6  0    1 1
```

We also create the outcome models:

```
mod0 <- latent_glm(
    formula = (y ~ 1),
    family_name = "binomial",
    link = "logit",
    coef_names = "alpha0")
mod1 <- latent_glm(
    formula = (y ~ rand),
    family_name = "binomial",
    link = "logit",
    coef_names = c("alpha1", "psi"))
```

In the absence of residual variances, there are now only four model parameters. The analysis is done just like in the former example:

```
res <- latent_main(
    data = df_obs,
    modelS = modS,
    model0 = mod0,
    model1 = mod1)
res
```

```
## $theta
## thetaS.alphaS theta0.alpha0 theta1.alpha1    theta1.psi
##    -1.1174488    -1.3205251    -1.2205341    -0.3002382
##
## $info
##              thetaS     theta0    theta1
## thetaS 2.773845e+02   3.570927  1.216976 1.570966e-14
## theta0 3.570927e+00 327.377585 48.543045 0.000000e+00
## theta1 1.216976e+00  48.543045 70.949175 5.416304e+01
##        1.570966e-14   0.000000 54.163043 5.416304e+01
```

From this, we can infer that the point estimate is $\hat{\psi} = -0.300$ with standard error 0.350. Confidence intervals and p-values can now be computed as before.

2.2.2 Ordinal outcomes

We now consider ordinal outcome data, analyzed using proportional odds models, for technical details, see [1]. We generate outcome data with four levels. The intercepts $(\alpha_1, \alpha_2, \alpha_3)$ of the proportional odds model are chosen so that the outcome probabilities for untreated patients are $(0.2, 0.4, 0.3, 0.1)$ in both subgroups. To this we add a an offset of $+0.4$ on the logit scale for treatable patients, together with a treatment effect of $\psi = -0.4$. This means that treatable patients have larger outcomes than untreatable ones unless they are treated, in which case their outcomes are similar.

```r
set.seed(12345)
df_true <- data.frame(
    g = rbinom(n, size = 1, prob = 0.25),
    rand = rbinom(n = n, size = 1, prob = 0.5))
## Generate outcome.
logit <- function(p) log(p / (1 - p)) # Classical logit function.
alpha <- -logit(c(0.2, 0.6, 0.9)) # Outcome probabilities (0.2, 0.4, 0.3, 0.1).
# Treatable patients have larger outcomes, unless treated.
lp <- ifelse((df_true$g == 0) | (df_true$rand == 0), 0, -0.4)
u <- runif(n) # A bunch of random numbers.
y <- rep(NA, n) # The outcome, defined below.
for (k in 1:3) {
    p.le.k = 1 / (1 + exp(alpha[k] + lp)) # P(y <= k).
    y <- ifelse(is.na(y) & (u < p.le.k), k, y)
}
y <- ifelse(is.na(y), 4, y) # If y is still NA, it has to be 4.
df_true$y <- factor(y, ordered = T)
# Make subgroup latent.
df_obs <- df_true
df_obs$g <- ifelse(df_obs$rand == 1, df_obs$g, NA)
head(df_obs)
```

```
##    g rand y
## 1 NA    0 3
## 2  1    1 1
## 3 NA    0 2
## 4  1    1 2
## 5  0    1 2
## 6  0    1 4
```

We also create the outcome models:

```r
mod0 <- latent_ordinal(
    formula = (y ~ 1),
    K = 4,
    coef_names = NULL)
mod1 <- latent_ordinal(
    formula = (y ~ rand),
    K = 4,
    coef_names = "psi")
```

We don't have to give names to the intercepts, they will automatically be named `alpha1`, `alpha2` etc. We can now run the analysis as usual:

```
res <- latent_main(
    data = df_obs,
    modelS = modS,
    model0 = mod0,
    model1 = mod1)
res
```

```
## $theta
## thetaS.alphaS theta0.alpha1 theta0.alpha2 theta0.alpha3 theta1.alpha1
##    -1.1180473     1.4310474    -0.4362784    -2.2560592     1.4059998
## theta1.alpha2 theta1.alpha3    theta1.psi
##    -0.4141761    -2.2517289    -0.5422899
##
## $info
##               thetaS      theta0                              theta1
## thetaS  2.773148e+02    2.177551   -1.455619   -0.3823406  -4.092583   4.100836
## theta0  2.177551e+00  365.627335 -177.018801    0.0000000  51.249215 -26.257789
##        -1.455619e+00 -177.018801  649.531034 -136.2089509 -26.539051  92.893581
##        -3.823406e-01    0.000000 -136.208951  203.3413877   0.000000 -18.961688
## theta1 -4.092583e+00   51.249215  -26.539051    0.0000000 109.630493 -43.549568
##         4.100836e+00  -26.257789   92.893581  -18.9616876 -43.549568 131.894325
##         8.618780e-02    0.000000  -18.940421   28.1490683   0.000000 -24.912890
##        -9.730064e-15    0.000000    0.000000    0.0000000  54.865515  46.924524
##
## thetaS   0.0861878 -9.730064e-15
## theta0   0.0000000  0.000000e+00
##        -18.9404208  0.000000e+00
##         28.1490683  0.000000e+00
## theta1   0.0000000  5.486552e+01
##        -24.9128903  4.692452e+01
##         33.8024049  5.704118e+00
##          5.7041175  1.074942e+02
```

Since each outcome model now has three intercepts, there are now $1 + 3 + 4 = 8$ model parameters altogether. Apart form this, everything works as usual. The estimated treatment effect is $\hat{\psi} = -0.542$ with standard error 0.246.

2.2.3 Time to event outcomes

Finally, we shall give an example of time to event data, analyzed using the flexible spline proportional hazards model of Royston and Parmar. Again, we refer to [1] for details.

We shall use three knots for the spline, placed at times $t = 0.2$, 0.6, and 1. Since the Royston–Parmar model uses a spline in $\log t$, the actual knots will reside at the logarithm of this. Data are then generated by choosing spline coefficients γ_j, $j = 0, 1, 2$, that yield a 3-knot approximatation of a Gompertz distribution with shape 1.5 and rate 1. Offset for treatable

patients and treatment effect ($\psi = -0.4$) are handled similarly to Section 2.2.2 for ordinal outcomes. Finally, event times are randomly censored according to the same distribution, meaning that half of the data points will be censored.

```
set.seed(12345)
df_true <- data.frame(
    g = rbinom(n, size = 1, prob = 0.25),
    rand = rbinom(n = n, size = 1, prob = 0.5))
## Generate outcome.
knots <- log(c(0.2, 0.6, 1))
gamma <- c(-0.08425288,  0.93050973, -0.89444548) # Approximate Gompertz.
lp <- ifelse((df_true$g == 0) | (df_true$rand == 0), 0, 0.4)
# Treatment modifies gamma0 (constant spline term).
gamma_mat <- matrix(rep(gamma, times = n), nrow = n, byrow = T)
gamma_mat[, 1] <- gamma_mat[, 1] + lp
# Simulate outcome times y and censoring times from the same distribution.
y_time_true <- flexsurv::rsurvspline(n = n,
    gamma = gamma_mat, knots = knots,
    scale = "hazard", timescale = "log")
cns_time <- flexsurv::rsurvspline(n = n,
    gamma = gamma_mat, knots = knots,
    scale = "hazard", timescale = "log")
# Censor data.
y_time_obs <- pmin(y_time_true, cns_time)
y_obs <- (y_time_true < cns_time)
df_true$y <- survival::Surv(y_time_obs, y_obs)
# Make subgroup latent.
df_obs <- df_true
df_obs$g <- ifelse(df_obs$rand == 1, df_obs$g, NA)
head(df_obs)

##    g rand           y
## 1 NA    0 0.695138641+
## 2  1    1  0.037484632
## 3 NA    0 0.008685517+
## 4  1    1 0.049473907+
## 5  0    1  0.392054106
## 6  0    1 0.187019222+
```

We also create the outcome models:

```r
mod0 <- latent_flexsurv_ph(
    formula = (y ~ 1),
    knots = knots,
    coef_names = NULL)
mod1 <- latent_flexsurv_ph(
    formula = (y ~ rand),
    knots = knots,
    coef_names = "psi")
```

The spline coeffocients will automatically be named `gamma0`, `gamma1` etc. Finally, we run the analysis:

```r
res <- latent_main(
    data = df_obs,
    modelS = modS,
    model0 = mod0,
    model1 = mod1)
res
```

```
## $theta
## thetaS.alphaS theta0.gamma0 theta0.gamma1 theta0.gamma2 theta1.gamma0
##   -1.11741953   -0.06698531    0.96835055   -0.83820060   -0.11532605
## theta1.gamma1 theta1.gamma2    theta1.psi
##    0.98452724   -0.84530417    0.64004792
##
## $info
##               thetaS      theta0                              theta1
## thetaS  2.773777e+02   -5.632161    5.577087    1.230293   -1.764022
## theta0 -5.632161e+00  957.946961 -620.851540 -438.303018  133.983668
##         5.577087e+00 -620.851540 1349.727401  -16.910061  -80.862581
##         1.230293e+00 -438.303018  -16.910061  459.508963  -67.822130
## theta1 -1.764022e+00  133.983668  -80.862581  -67.822130  233.786407
##         1.458428e+00  -84.121025  185.891637   -1.701377 -213.759925
##         1.311862e+00  -66.116354   -3.235776   70.941731  -65.142633
##        -7.775898e-16    0.000000    0.000000    0.000000  191.002926
##
## thetaS    1.458428   1.311862 -7.775898e-16
## theta0  -84.121025 -66.116354  0.000000e+00
##         185.891637  -3.235776  0.000000e+00
##          -1.701377  70.941731  0.000000e+00
## theta1 -213.759925 -65.142633  1.910029e+02
##         462.233903   1.826727 -1.870293e+02
##           1.826727  56.905762 -4.689905e+01
##        -187.029333 -46.899052  1.910029e+02
```

Again, there are $1 + 3 + 4 = 8$ parameters, this time owing to the fact that we chose three spline knots. The estimated treatment effect is $\hat{\psi} = 0.640$ with standard error 0.218.

2.3 Adding covariates

An important property of our framework is that each of the models can be modified independently of the others. For example, we may add covariates in order to increase the power. This idea is well-known for the outcome of a randomized trial, but is also very natural for the subgroup. For example, it might be that men are more often treatable than women. By adding sex to the subgroup model, we are less ignorant about the subgroup status of the patients, and hence take a step in the direction of identifying the subgroup, which should increase power.

We shall illustrate these ideas by returning to the basic example of Section 2.1 with a numerical outcome. Assume that two covariates x_g and x_y have been identified, that are predictive of subgroup membership and the outcome, respectively. We can then modify the models (1):

$$\begin{cases} \operatorname{logit} P(G = 1) = \alpha_S + \beta_g x_g & \\ \qquad\qquad Y = \alpha_0 + \beta_0 x_y + \epsilon, & G = 0, \\ \qquad\qquad Y = \alpha_1 + \beta_1 x_y + \psi R + \epsilon, & G = 1. \end{cases}$$

For the simulation, we let x_g and x_y be independent standard normal and set $\beta_g = \beta_0 = \beta_1 = 0.5$. Due to non-collapsibility, α_S needs to be modified to keep the prevalence at 25%. It turns out that the correct value is -1.158 rather than $\operatorname{logit} 0.25 = -1.099$ as in Section 2.1.

We can now generated data:

```r
set.seed(12345)
df_true <- data.frame(
    xg = rnorm(n),
    xy = rnorm(n),
    rand = rbinom(n = n, size = 1, prob = 0.5))
# Subgroup
expit <- function(x) exp(x) / (1 + exp(x))
df_true$g <- rbinom(n, size = 1, prob = expit(-1.158 + 0.25 * df_true$xg))
# Outcome.
df_true$y <- ifelse(df_true$g == 0,
    rnorm(n, mean = 100 + 0.5 * df_true$xy, sd = 2),
    rnorm(n, mean = 100.4 + 0.5 * df_true$xy - 0.4 * df_true$rand, sd = 2.5))
df_obs <- df_true
df_obs$g <- ifelse(df_obs$rand == 1, df_obs$g, NA)
head(df_obs)
```

```
##           xg          xy rand  g         y
## 1  0.5855288  0.50989212    0 NA  98.61569
## 2  0.7094660 -0.71536427    1  0 102.39400
## 3 -0.1093033 -0.40552154    0 NA  97.45578
## 4 -0.4534972 -0.01252119    1  0 100.40468
## 5  0.6058875  1.10088104    0 NA  99.17082
## 6 -1.8179560  0.95737579    0 NA  97.96304
```

Next, we create the models:

```r
modS <- latent_glm(
    formula = (g ~ xg),
    family_name = "binomial",
    link = "logit",
    coef_names = c("alphaS", "betaG"))
mod0 <- latent_linear(
    formula = (y ~ xy),
    coef_names = c("alpha0", "beta0"))
mod1 <- latent_linear(
    formula = (y ~ xy + rand),
    coef_names = c("alpha1", "beta1", "psi"))
```

Note that there are now more parameters to be named. The rest of the analysis follows the by now well-trodden path:

```
res <- latent_main(
    data = df_obs,
    modelS = modS,
    model0 = mod0,
    model1 = mod1)
res
```

```
## $theta
## thetaS.alphaS  thetaS.betaG theta0.alpha0  theta0.beta0 theta0.sigma2
##    -1.2617121     0.2090331    99.9987333     0.5934325     3.8760717
## theta1.alpha1  theta1.beta1    theta1.psi theta1.sigma2
##   100.3717693     0.5453931    -0.2782398     6.2581235
##
## $info
##               thetaS                   theta0                          theta1
## thetaS  2.642029e+02  3.023013e+01  13.048189  -1.123301 11.4037010  6.921198
##         3.023013e+01  2.647747e+02   7.211382  -4.128046  4.1492069 -2.778527
## theta0  1.304819e+01  7.211382e+00 514.097994  -1.505705 -3.7469562 53.471103
##        -1.123301e+00 -4.128046e+00  -1.505705 510.599480 -1.3644082 -1.947406
##         1.140370e+01  4.149207e+00  -3.746956  -1.364408 62.9460528  1.628247
## theta1  6.921198e+00 -2.778527e+00  53.471103  -1.947406  1.6282469 73.162325
##        -1.469449e+00  1.114794e+00  -1.947406  52.411665  0.9972904  0.341885
##         1.513546e-16  6.991478e-16   0.000000   0.000000  0.0000000 53.051047
##         3.294659e+00 -5.807958e-01  -2.188131   1.084044  5.1707249  1.163845
##
## thetaS -1.4694490  1.513546e-16  3.294659e+00
##         1.1147939  6.991478e-16 -5.807958e-01
## theta0 -1.9474057  0.000000e+00 -2.188131e+00
##        52.4116654  0.000000e+00  1.084044e+00
##         0.9972904  0.000000e+00  5.170725e+00
## theta1  0.3418850  5.305105e+01  1.163845e+00
##        72.4578778  3.239158e-01 -6.651135e-01
##         0.3239158  5.305105e+01 -2.028488e-12
##        -0.6651135 -2.028488e-12  6.361142e+00
```

Naturally, adding covariates results in more model parameters (9 of them this time), but otherwise things are the same. The estimated treatment effect is $\hat{\psi} = -0.278$ with standard error 0.300.

References

[1] L. Lindhagen, H. Garmo, and O. Östlund, A modular framework for treatment effect estimation in latent subgroups, 2023, manuscript.

[2] R. Hofmann, HELicobacter Pylori Screening to Prevent Gastrointestinal Bleeding in MI Patients (HELP-MI), https://clinicaltrials.gov/show/NCT05024864, Accessed 2022–06–09.