Something that never ceases to amaze (depress) me, is how extremely common it is to see casual claims in RCTs, that are not part of the randomization. For instance, the relationship between treatment adherence and outcome, or between alliance and outcome, are often analyzed but seldom experimentally manipulated. This is basically observational research disguised as experimental, but without DAGs, instrumental variables, propensity scores, or *any* other technique used in observational research. Hence, the effects will *almost surely* be biased in an unknown direction. This is a post I’ve been planning to write for a long time, but put off since the theory and methods are so well covered in, for instance, epidemiology. However, when brms added support for multivariate formulas, I figured I could show how some of the techniques can be implemented in R, and their Bayesian counterparts using brms. So, in this post I do three things:

1. Rant about how quickly clinical psychology researchers forget about **casual inference**, once they start analyzing non-randomized variables, such as the effect of **treatment adherence** (or “compliance”) on treatment outcome in a randomized controlled trial.
2. Show that common regression methods either, makes absolutely no sense, or result in badly biased estimates of the casual dose-response effect, when a third variable influences both adherence and outcome, i.e. when the dose-response relationship is confounded. While **instrumental variable** methods *may* allow for unbiased estimates in these scenarios.
3. Show how **Bayesian intrumental variable** regression can be easily fit using brms new multivariate syntax. And also do a small simulation study to compare the results to lavaan, AER::ivreg and biased lm() models.

**Motivating example**

**The effect of treatment adherence (psychotherapy dose-response effects)**

Let’s assume we’ve performed an RCT and compared some psychotherapy condition to some control condition. Patients in these types of trials seldom complete all scheduled sessions, or adhere 100 % to the treatment. It’s rather common for researchers to try to see if there is some relationship between adherence and treatment outcome. Which is a valid question, if there is no added effect of actually “receiving” more of the treatment, then that *should* be a problem for your theory, or at least some of the components of that theory. However, what usually happens is that researchers somehow forget that the variable “adherence to therapy” is not experimentally manipulated. Unless we have randomized patients to receive 1, 2, …, 10 sessions, or to do 1, 2, …, 10, exposure exercises (and somehow ensured adherence was 100% in all conditions)–then we are now longer in RCT country!

*Never stop to acknowledge you’re doing non-randomized comparisons!*

**A simple model of confounded adherence effects**

A really simple scenario with one know (or unknown) confounder is shown in the DAG below.

Each arrow represent *casual* influence of one variable on another. Y is the observed outcome, adherence is often measured by some proxy, like the number of sessions completed–that we assume is a good indicator of treatment “dose”, it could be any other proxy for treatment adherence. So, what the graph is saying, is that treatment allocation (randomization) influences the outcome (Y) via the amount of treatment received (adherence). Thus, unless something went horribly wrong, randomization should be strongly correlated with S, and only influence Y through the treatment received (S). By definition this makes randomization an *instrumental variable*. However, the problem is that the effect of S on Y is most likely confounded (U). For instance, baseline functioning might be related both to a better prognosis (better outcome) and treatment adherence. It is easy to come up with scenarios were the relationship could be either positive or negative. Higher baseline functioning could be related to a better prognosis, and to better treatment adherence, or, a patient could be too depressed to actually adhere to the treatment. It is also plausible that worse baseline functioning make you more motivated to adhere to your treatment. Whatever the relationship might be, it is clear that the casual effect of treatment adherence will be confounded.

We will deal with three types of scenarios were a variable \(U\) confounds the estimate of treatment adherence:

* **Observed confounder**: \(U\) is an observed variable that is adjusted for.
* **Residual confounding**: our known confounder \(U\) is measured with error.
* **Hidden confounding**: \(U\) is either unknown or unmeasured, and not adjusted for.

**The data generating model**

First, let’s remind ourselves what is commonly estimates in an RCT is the *average treatment effect* (ATE), or some related estimate,

$$  
\Delta = E[Y\_i(C) – Y\_i(Tx)].  
$$

Where \(Y\_i(C)\) is the would-be outcome for patient *i* after participating in the control condition, and \(Y\_i(Tx)\) the would be outcome for patient *i* after participating in the treatment condition. Obviously, we can never observe both \(Y\_i(C)\) and \(Y\_i(Tx)\) for the same patient. But thanks to randomization when can infer the ATE. We also assume that the individual treatment effect \(Y\_i(C) – Y\_i(Tx)\) is the same for all individuals, given the same level of adherence.

The data generating model (DGM) is, thus, that adherence depends on \(U\_i\), the confounder variable,

$$ S\_i = 7 – f(U\_i) + e\_{1i}.$$

Where \(f()\) is just a function that centers and scales the confounding variable, so that the mean adherence is 7, and \(e\_i\) is just random variation that influence the relationship between \(U\_i\) and \(S\_i\). Moreover, the *observed* measure of adherence could be noisy measure of a patient’s actual adherence, \(S\_i\), e.g. sessions completed,

$$ \text{Sessions}\_i = S\_i + e\_{2i}.$$

To generate the outcome, we first simulate \(Y\_i(C)\), and then calculate \(Y\_i(Tx)\),

$$ Y(Tx)\_i = Y(C)\_i – \beta\_S S\_i.$$

So that \(\beta\_S\) is the casual effect of treatment adherence,

$$ E(\Delta\_i | S\_i) = \beta\_s S\_i.$$

Lastly, the outcome is generated as,

$$ Y\_i = Y\_i(C) [1-R\_i(Tx)] + Y\_i(Tx) R\_i(Tx).$$

Where \(R\_i(Tx)\) is 1 for participants allocated to the treatment condition and 0 otherwise.

**Different ways to answer the same question**

Now, that we have a DGM, let’s generate some fake data and fit different models. We’ll start with a standard OLS regression, the first model would make sense in the absence of any hidden or residual confounding (extremely unlikely). The second model (“naïve OLS”) makes little sense, since it only analyze the treatment group, and as we’ll see, further confounds prognosis with the casual effect of treatment adherence. Sadly, this way of analyzing psychotherapy trials is really common… Anyway, lets generate some fake data. I’ll use a two-level random intercept-only model as a starting point, and assume that the confounder (baseline functioning), is measured without error.

library(powerlmm)

# Skeleton to base data on

sp <- study\_parameters(n1 = 11,

n2 = 100,

T\_end = 10,

fixed\_intercept = 10,

fixed\_slope = -.5,

icc\_pre\_subject = 0.8,

sigma\_error = 3,

cohend = 0)

## Some fake data, code at the end of post

dpp <- data\_generator(update(sp, n2 = 5000),

confounder = "intercept\_subject",

label = "example",

beta\_s = 1,

c\_scale = 3,

hidden\_confounder\_sd = 0,

measure\_error\_sd = 0)()

**OLS**

First, we fit the two standard OLS models where we are adjusting for pretest scores.

# lm

## Assume no confounding

OLS <- lm(y ~ session + pre, data = dpp)

summary(OLS)

## Naïve OLS (tx only)

naive <- dpp %>%

filter(treatment == 1) %>%

lm(y ~ session + pre, data = .)

summary(naive)

**Instrumental variables (IV)**

Now, let’s try to use treatment assignment as an *instrument* to better uncover the casual effect of adherence. Two-stage least-squares (2SLS) is a common method to estimate an IV model. The point of this post is not to explain IV regression, it’s explained well in standard econometric textbooks ([Wooldridge, 2010](https://www.amazon.com/gp/product/0262232588/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0262232588&linkCode=as2&tag=rpsyc-20&linkId=8b0629e282068c90e6ecc6f093bcf1b4); [Wooldridge, 2015](https://www.amazon.com/gp/product/130527010X/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=130527010X&linkCode=as2&tag=rpsyc-20&linkId=4941b5cb6ce034f04eed61345e7a656a)). The *very* short explanation, is that 2SLS is basically to regress session on the instruments, then regress y on the predicted sessions from Step 1. This can be done using lm() to get point estimates, but the standard errors will be highly suspicious.

# IV

## Two-step manual

step1 <- lm(session ~ treatment + pre:treatment, data = dpp)

dpp$session\_hat <- predict(step1)

IV\_ols <- lm(y ~ session\_hat + pre, data = dpp)

summary(IV\_ols)

However, 2SLS is easier done with ivreg() from the package AER. The same structural model is fit using this code.

library(AER)

## ivreg (2SLS)

IV <- ivreg(y ~ session + pre | treatment + treatment:pre, data = dpp)

summary(IV)

The model can also be thought of as a special case of a structural equation model, and fit with e.g. lavaan.

library(lavaan)

l\_model <- ' session ~ treatment + treatment:pre

y ~ session + pre

session ~~ y '

IV\_lavaan <- sem(l\_model, data = dpp)

summary(IV\_lavaan)

Lastly, using the same approach, a Bayesian IV model can be fit using brms (>2.0.0).

library(brms)

f1 <- bf(session ~ treatment + treatment:pre)

f2 <- bf(y ~ session + pre)

IV\_brm <- brm(f1 + f2, data = dpp, cores = 4)

summary(IV\_brm)

This is allows extending the model a lot using brms highly flexible modelling syntax. Moreover, once you’ve exhausted brms capabilities you can get the stancode() and further customize the model yourself.

**A small simulation study**

Let’s also do a small simulation study, to better show the impact of observed confounding, residual confounding, and hidden confounding. We’ll use the OLS and IV models shown above. For each of the three scenarios 5000 data sets were generated, with 100 patients per condition. The correlations between the variables are shown in the figure below, the actual values are arbitrary chosen, and the resulting bias, and its direction, is not meant to reflect real data, instead the values are meant to highlight the different types of confounding.

In the figure, *s* is the latent adherence variable, *session* the observed adherence, *intercept\_subject* is the baseline confounder, that is measured with or without error by *pre*, and *y* is the outcome.

**Relative bias**

The relative bias of the estimated adherence effect is shown in the figure below.

In the figure, we see for the OLS models, that when there’s:

* **Hidden confounding**:
  + Adjusting for the baseline variable does not help, which is as expected, since it is not related to *adherence*.
* **Observed confounder**:
  + The estimates are biased when the confounder is omitted.
  + However, the effect is attenuated since *session* is a noisy proxy of adherence.
* **Residual confounding**:
  + Controlling for the confounder reduces bias, but due to residual confounding estimates are still biased.

And the overall pattern is that:

* the naïve OLS is as bad idea, and is the most affected by bias in all scenarios. We use control groups for a reason, don’t ignore the control group!
* IV estimates are unbiased.

We see the same results if look at the coverage probabilities of the confidence intervals.

**Coverage probabilities of the CIs**

**When less is more–collider bias**

Let’s throw in another example of how things can go wrong, this time by adjusting for a collider. We extend the first DAG, and add another covariate *therapeutic alliance*. A researcher might think that by controlling for X (alliance), the dose-response relationship will be more clearly revealed. Or they might just do a multivariable regression and think *“hey, both therapeutic alliance and adherence probably predict treatment outcome, let’s throw both in and see what p-value we get”*. However, the problem is that alliance might neither be a predictor of outcome, nor a confounder of the adherence and outcome relationship, instead it could be a common effect of adherence and a hidden confounder (i.e., a collider). Obviously, this example is made up and *alliance* is just a placeholder–there’s so much going on during a psychological treatment that it would be easy to come up with other colliders. In the DAG bellow, conditioning on alliance opens up a path between adherence and the hidden confounder, that bias the relationship between adherence and outcome *even* more.

If we also simulate this model, and assume that \(U\_1\) is measured with error, and use the same IV regressions, but add an OLS model that also includes the collider we get these results.

What we see is that:

* The OLS models that control for the baseline confounder is still biased due to residual confounding, and the naïve OLS is badly biased.
* By adjusting for the collider the bias increase, it would even have been better to not adjust for anything.
* The IV models are unbiased, and as you see below, their CIs have nominal coverage probabilities as well.

**Summary**

Lastly, to summarize this post:

* Don’t do naïve regressions on treatment adherence dose-response effects. It makes no sense even in the absence of confounding.
* Measurement error *will* bias your regression estimates even if you adjust for confounders.
* Confounders and colliders are everywhere, make your assumptions explicit. We don’t need more zombie studies claiming XYZ is related to treatment outcome.
* IV regression does not automagically solve all problems. But if you think hard about your data it can be a very useful tool in treatment studies.
* brms new multivariate syntax make it extremely easy to fit Bayesian IV models, that can be extended to much more complicated scenarios than what I covered in this article.

**Code**

**Simulation functions**

library(dplyr)

library(tidyr)

library(ggplot2)

library(parallel)

library(lavaan)

library(AER)

library(brms)

#' Extract results from lm and ivreg

#' @param x object

#' @param model character; name of model

get\_res <- function(x, model) {

est <- coef(x)

ind <- names(est) == "session"

est <- est[ind]

se <- sqrt(diag(vcov(x)))[ind]

data.frame(model = model,

para = names(est),

est = est,

se = se,

lwr = est - 1.96 \* se,

upr = est + 1.96 \* se)

}

#' brms: fit and extract results

#' @param dpp new pre-post data to fit

#' @param fit\_brm brms-object to update

get\_brm <- function(dpp, fit\_brm) {

fit <- suppressMessages(update(fit\_brm, newdata = dpp, iter = 2000))

est <- fixef(fit)

est <- est[rownames(est) == "y\_session", ]

data.frame(model ="IV (brms)",

para = "session",

est = est[1],

se = est[2],

lwr = est[3],

upr = est[4])

}

#' lavaan: creates function that fit and extract results

#' @param model character; lavaan model syntax

#' @return function; accepts new data via "dpp"

make\_lavaan\_func <- function(model) {

function(dpp) {

fit\_lavaan <- tryCatch({sem(model, data=dpp)},

error = function(e) {NA},

warning = function(w) {NA})

if(isS4(fit\_lavaan)) {

se <- sqrt(diag(vcov(fit\_lavaan)))

ind <- names(se) == "y~session"

est <- coef(fit\_lavaan)

est <- est[ind]

se <- se[ind]

} else if ([is.na](http://is.na)(fit\_lavaan)) {

se <- NA

est <- NA

}

data.frame(model = "IV (lavaan)",

para = "session",

est = est,

se = se,

lwr = est - 1.96 \* se,

upr = est + 1.96 \* se)

}

}

#' add confounding to pre-post data

#' @param x prepost data

#' @param confounder charater; name of column in x to use as confounder

#' @param n2 number of total subjects

#' @beta\_s dose-response parameter

#' @c\_scale scale confounding distribution

add\_confounding <- function(x, confounder, n2, beta\_s, c\_scale) {

x$confounder <- x[, confounder]

x <- x %>%

mutate(s = (7 - (confounder - mean(confounder))/sd(confounder) \* c\_scale) - hidden\_confounder\*0.5 + rnorm(n2, sd = 1),

s = ifelse(s < 1, 1, s) \* treatment,

session = s + rnorm(n2, sd = 2),

session = ifelse(session < 1, 1, session) \* treatment,

yt = yc - beta\_s \* s,

y = yt \* treatment + yc \* (1-treatment),

id = 1:n2)

x

}

add\_collider <- function(x, confounder, n2, beta\_s, c\_scale) {

x$confounder <- x[, confounder]

x <- x %>%

mutate(s = (7 - (confounder - mean(confounder))/sd(confounder) \* c\_scale) + rnorm(n2, sd = 1),

s = ifelse(s < 1, 1, s) \* treatment,

session = s + rnorm(n2, sd = 0),

session = ifelse(session < 1, 1, session) \* treatment,

cred = (session + hidden\_confounder) \* treatment,

yt = yc - beta\_s \* s,

y = yt \* treatment + yc \* (1-treatment),

id = 1:n2)

x

}

#' Create data generator

#'

#' @param pp powerlmm::study\_parameters-object

#' @param confounder character; passed to 'add\_confounding'

#' @param beta\_s passed to 'add\_confounding'

#' @param c\_scale passed to 'add\_confounding'

#'

#' @return function with no arguments that generate data.frame

data\_generator <- function(pp, confounder, label, beta\_s, c\_scale, hidden\_confounder\_sd = 0, measure\_error\_sd = 0, collider = FALSE) {

function() {

d <- simulate\_data(pp) # powerlmm::simulate\_data

dpp <- d %>%

filter(time %in% c(0, 10),

treatment == 0) %>%

mutate(time = factor(time, labels = c("pre", "post")),

treatment = rep(0:1, each = pp$n2)) %>%

spread(time, y) %>%

mutate(hidden\_confounder = rnorm(pp$n2, sd = hidden\_confounder\_sd),

yc = post + hidden\_confounder)

if(collider) {

dpp <- add\_collider(dpp, confounder = confounder, pp$n2, beta\_s, c\_scale)

} else {

dpp <- add\_confounding(dpp, confounder = confounder, pp$n2, beta\_s, c\_scale)

}

dpp <- dpp %>%

mutate(pre = intercept\_subject + rnorm(pp$n2, sd = measure\_error\_sd),

label = label)

dpp

}

}

#' Simulation function

#'

#' @param i i:th simulation

#' @param data\_gen data\_generator

#' @param fit\_brm prefit brms-object

sim <- function(i, data\_gen, fit\_brm) {

dpp <- data\_gen()

d\_t <- dpp %>%

filter(treatment == 1)

OLS <- lm(y ~ session, data = dpp)

OLS\_pre <- lm(y ~ pre + session, data = dpp)

OLS\_tx <- lm(y ~ session, data = d\_t)

OLS\_pre\_tx <- lm(y ~ pre + session, data = d\_t)

IV\_ivreg <- ivreg(y ~ session + pre | treatment + treatment:pre, data = dpp)

IV\_brm <- get\_brm(dpp, fit\_brm)

IV\_lavaan <- get\_lavaan(dpp)

res <- mapply(get\_res,

list(OLS, OLS\_pre, OLS\_tx, OLS\_pre\_tx, IV\_ivreg),

model = c("OLS s", "OLS s + pre", "OLS s (tx)", "OLS s + pre (tx)", "IV (ivreg)"),

SIMPLIFY = FALSE)

res <- c(res, list(IV\_brm), list(IV\_lavaan))

res <- do.call(rbind, res)

res$sim <- i

res

}

sim\_collider <- function(i, data\_gen, fit\_brm) {

dpp <- data\_gen()

d\_t <- dpp %>%

filter(treatment == 1)

OLS <- lm(y ~ session, data = dpp)

OLS\_pre <- lm(y ~ pre + session, data = dpp)

OLS\_pre\_cred <- lm(y ~ pre + session + cred, data = dpp)

OLS\_tx <- lm(y ~ session, data = d\_t)

OLS\_pre\_tx <- lm(y ~ pre + session, data = d\_t)

OLS\_pre\_tx\_cred <- lm(y ~ pre + session + cred, data = d\_t)

IV\_ivreg <- ivreg(y ~ session + pre | treatment + treatment:pre, data = dpp)

IV\_brm <- get\_brm(dpp, fit\_brm)

IV\_lavaan <- get\_lavaan(dpp)

res <- mapply(get\_res,

list(OLS, OLS\_pre, OLS\_pre\_cred, OLS\_tx, OLS\_pre\_tx, OLS\_pre\_tx\_cred, IV\_ivreg),

model = c("OLS s", "OLS s + pre","OLS s + pre + collider",

"OLS s (tx)", "OLS s + pre (tx)", "OLS s + pre + collider (tx)",

"IV (ivreg)"),

SIMPLIFY = FALSE)

res <- c(res, list(IV\_brm), list(IV\_lavaan))

res <- do.call(rbind, res)

res$sim <- i

res

}

#' Summarise simulation results

#'

#' @param x Sim results from sim()

#' @param type character; label to use for type of confounding

#' @param theta true value of session dose-response coef

summarise\_sim <- function(x, type, theta = -1) {

x <- x %>% group\_by(model) %>%

summarise(est\_M = mean(est, na.rm = TRUE),

est\_SE = mean(se, na.rm = TRUE),

est\_SD = sd(est, na.rm = TRUE),

cover = mean(lwr < theta & theta < upr, na.rm = TRUE)) %>%

mutate(est\_RB = ((est\_M - theta)/theta) \* 100,

se\_RB = (est\_SE - est\_SD)/est\_SD)

x$type <- type

x

}

**Setup and run the simulation.**

source("functions.R")

library(methods)

# Sim setup

nsim <- 5000

cores <- 16

sp <- study\_parameters(n1 = 11,

n2 = 100,

T\_end = 10,

fixed\_intercept = 10,

fixed\_slope = -.5,

icc\_pre\_subject = 0.8,

sigma\_error = 3,

cohend = 0)

# start clusters

cl <- makeCluster(cores)

clusterEvalQ(cl, {

library(powerlmm)

library(dplyr)

library(tidyr)

library(AER)

library(brms)

library(lavaan)

library(methods)

})

# brms setup

# precompile model

if(!exists("fit\_brm")) {

dpp <- data\_generator(p\_pre, "intercept\_subject", label = "setup", beta\_s = 1, c\_scale = 3)()

f1 <- bf(session ~ treatment + treatment:pre)

f2 <- bf(y ~ session + pre)

fit\_brm <- brm(f1 + f2, data = dpp, chains = 1, cores = 1, iter = 1)

}

# lavaan setup

l\_model <- ' session ~ 1 + treatment + treatment:pre

y ~ 1 + session + pre

session ~~ y'

get\_lavaan <- make\_lavaan\_func(l\_model)

clusterExport(cl, c("sp",

"get\_res",

"add\_confounding",

"add\_collider",

"fit\_brm",

"get\_brm",

"get\_lavaan",

"l\_model"))

# run sim

ptm <- proc.time()

# Observed

res1 <- parLapply(cl,

1:nsim,

sim,

data\_gen = data\_generator(sp, "intercept\_subject",

label = "Observed confounder",

beta\_s = 1,

c\_scale = 3,

hidden\_confounder\_sd = 0,

measure\_error\_sd = 0),

fit\_brm = fit\_brm)

res1 <- do.call(rbind, res1)

# Residual confound

res2 <- parLapply(cl,

1:nsim,

sim, data\_gen = data\_generator(sp,

"intercept\_subject",

label = "Residual confounding",

beta\_s = 1,

c\_scale = 3,

hidden\_confounder\_sd = 0,

measure\_error\_sd = 6),

fit\_brm = fit\_brm)

res2 <- do.call(rbind, res2)

# Hidden confound

res3 <- parLapply(cl,

1:nsim,

sim,

data\_gen = data\_generator(sp, "intercept\_subject",

label = "Hidden confounding",

beta\_s = 1,

c\_scale = 0,

hidden\_confounder\_sd = 6,

measure\_error\_sd = 0),

fit\_brm = fit\_brm)

res3 <- do.call(rbind, res3)

x <- mapply(summarise\_sim,

list(res1, res2, res3),

type = list("Known confounder", "Residual confounder", "Residual + hidden confounder"),

SIMPLIFY = FALSE)

x <- do.call(rbind, x)

# Collider ----------------------------------------------------------------

res\_c <- parLapply(cl,

1:nsim,

sim\_collider,

data\_gen = data\_generator(sp, "intercept\_subject",

label = "Collider",

beta\_s = 1,

c\_scale = 3,

hidden\_confounder\_sd = 5,

measure\_error\_sd = 4,

collider = TRUE),

fit\_brm = fit\_brm)

res\_c <- do.call(rbind, res\_c)

stopCluster(cl)

(et <- proc.time() - ptm)

x\_c <- summarise\_sim(res\_c, type = "collider", theta = -1)

saveRDS(list("res1" = res1,

"res2" = res2,

"res3" = res3,

"res\_c" = res\_c,

"x" = x,

"x\_c" = x\_c,

"et" = et),

file = "sim\_res.rds")