Companion to simulations in Scholl and Molo (2024)

Fabio Molo, Raphael Scholl

Introduction

This notebook describes and runs the simulations in Section 5.3 of Scholl Molo (2024) using the R statistical programming language.

Setup

We first load the required libraries.

```
library(ggplot2) # for visualisations
library(scales) # for formatting axes in visualisations
library(parallel) # for parallel processing on multiple CPU cores
library(ggpubr) # for combining multiple ggplot2 graphs with shared legend
library(grid)  # for textGrob() in combined graphs
```

Set parameters

We begin by defining several parameters for our simulated randomized experiments:

```
n <- 200 # n observations
p_max <- 1000 # p confounders
te <- 0 # treatment effect beta_t
coef_size <- 1 # beta_p (taken as constant for all confounders)
r <- c(1, 10, 10000) # number of simulations
distr <- "rnorm" # for standard-normally distributed confounders
```

- We define the number of observations n. In a randomized clinical trial, this would correspond to the total number of patients in the trial. We choose $n = 200$ for all trials. (This roughly corresponds to the median number of enrolled participants of all Phase 3 randomized trials finished in 2023 registered on clinicaltrials.gov, which is 190.) Since we consider trials with two groups of equal sizes, this will simulate a trial in which the control and intervention groups each have 100 participants.

- We define the number of potential confounders p that are active in the system. We will run simulations starting at $p = 1$ and then increasing p gradually by 1 to a maximum of $p_{max} = 1000$.
- We define the effect size β_T of the treatment that the intervention group receives and the control group does not. Whenever we simulate a situation with no treatment effect, we set $\beta_T = 0$.
- We define the effect size of the confounders. For simplicity, we set the same effect size $\beta = 1$ for all confounders: $\beta_1 = \beta_2 = ... = \beta_p = \beta$.
- We define the number of times r the simulation is run for each number of confounders. We perform the simulation three times: first with $r = 1$ to stay close to the scenario outlined by Worrall. Ten we show results for a moderate number of repetitions ($r = 10$), and last we choose a high number of $r = 10000$ to obtain estimates with negligible Monte Carlo error.
- We also need to define the distribution of the simulated confounders, that is, the distribution that the simulated values for the individual observations i are randomly sampled from. We only consider the simplest cases where all confounders are standard normally distributed (i.e. normally distributed with a mean of 0 and a standard deviation of 1).

Generate the true data

Having defined these parameters, we can now generate experimental data as follows:

- We simulate values for p potential confounding variables for each observation i, with $i = 1, \dots, n$, This results in a matrix X of dimensions $n \times p$, where the number of rows corresponds to the number of participants and the number of columns corresponds to the number of potential confounders in the trial. This allows us to indicate whether a confounder j is active for the participant i ($X_{i,j} = 1$) or not ($X_{i,j} = 0$), for the case of binary confounders. For normally distributed confounders, the value of $X_{i,j}$ expresses *to what extent* a confounder is active in the system. We denote confounder number j with $x^{(j)}$. The matrix of counfounders is generated using the following function:

```r
# function to simulate n observations of p confounding variables
# of a given distribution function
generate_confounders <- function(n, p, distr = "rnorm", ...) {
    res <- sapply(1:p, FUN = function(x) {
        do.call(distr, args = list(n = n, ...))
        })
}
```

- We simulate for each observation i whether they receive the treatment ($T_i = 1$) or not ($T_i = 0$). The treatment is allocated randomly.

- We simulate an error term ϵ_i for each observation that corresponds to 'measurement error'. We draw ϵ_i from a standard normal distribution.

With these values set, the true outcome y_i for each observation i is fully determined:

$y_i = \beta_T * T_i + \beta_1 * x_i^{(1)} + ... + \beta_p * x_i^{(p)} + \epsilon_i$

Three notes on this setup:

- β_T represents the *average* treatment effect. We do not consider individual treatment effects per observation ($\beta_{T,i}$).
- The effect of the potential confounders on the outcome are linear additive. We do not consider interactions between confounders or between confounders and the treatment. Also, the confounders are mutually independent, as are the observations i.
- Sampling the sum of p normally distributed variables is equivalent to sampling from one normally distributed variable with mean and variance equal to the sums of means and variances of the individual variables. To stay close to the setup that @Worrall:2002aa is concerned about we still sample each confounder individually.

Conduct an analysis of variance for the treatment

Next we analyse the generated experimental data. Here we conduct an ANOVA of the outcome on the randomized treatment. In R, this is implemented by first fitting a linear regression model and then extracting the relevant values from the ANOVA table. In our case, the relevant values are the variance between groups and the variance within groups. (We also extract the effect size estimate and its confidence interval from each simulation for later analysis below.)

Data generation and analysis are executed using the following R function:

```r
simul <- function(p, r, n, te, coef_size, distr) {

    # initiate empty vectors to store values
    est <- numeric(r)
    ci_lower <- numeric(r)
    ci_upper <- numeric(r)
    var_between <- numeric(r)
    var_within <- numeric(r)

    for (i in 1:r) {

        # generate potential confounders
        X <- generate_confounders(n = n, p = p, distr = distr)

        # set effect of counfounders and random noise
        beta <- rep(coef_size, p)
        epsilon <- rnorm(n, mean = 0, sd = 1)

        # randomize treatment
        treat <- sample(rep(0:1, n / 2))

        # compute the "true" outcome y
        y <- te * treat + X %*% beta + epsilon

        # fit linear regression model for treatment
        fit_lm <- lm(y ~ treat)

        # conduct analysis of variance
        fit <- anova(fit_lm)

        # record relevant values for each simulation
        est[i] <- coef(fit_lm)["treat"]
        ci_lower[i] <- confint(fit_lm)["treat", 1]
        ci_upper[i] <- confint(fit_lm)["treat", 2]
        var_between[i] <- fit["treat", "Mean Sq"]
        var_within[i] <- fit["Residuals", "Mean Sq"]
    }
    print(paste("Finished simulation for p =", p))
    print(Sys.time())
    return(list(
        est = est,
        ci_lower = ci_lower,
        ci_upper = ci_upper,
        var_between = var_between,
        var_within = var_within
    ))
}
```

Run the simulation

Then we run the simulation 1, 10, 10000 times for each number of potential confounders p. We set a 'seed' to be able to always reproduce the same pseudorandom data. The results of all simulations are stored in a list and saved on disk.

```
RNGkind("L'Ecuyer-CMRG") # needs to be set for reproducibility in parallel processing
set.seed(0)

# using parameters set above
# three times for three different specified numbers of simulations r

simres1 <- lapply(
```

```r
    1:p_max, simul, r = r[1], n = n, te = te,
    coef_size = coef_size, distr = distr
)

# parallelize the simulation across multiple CPU cores (for speed)
# this takes long to run
# (consider smaller p and r values to save time)
simres2 <- mclapply(
    1:p_max, simul, r = r[2], n = n, te = te,
    coef_size = coef_size, distr = distr,
    mc.cores = 6
)

# this takes long to run
# (consider smaller p and r values to save time)
simres3 <- mclapply(
    1:p_max, simul, r = r[3], n = n, te = te,
    coef_size = coef_size, distr = distr,
    mc.cores = 6
)

# to reproduce the simulations without parallelization
# (which is probably necessary on Windows):
# simres <- lapply(
#     1:p_max, simul, r = r, n = n, te = te,
#     coef_size = coef_size, distr = distr
#     )

saveRDS(simres1, "simres_r1.rds")
saveRDS(simres2, "simres_r2.rds")
saveRDS(simres3, "simres_r3.rds")

# alternatively we can load the results of previously run simulations from disk
# to save time
simres1 <- readRDS("simres_r1.rds")
simres2 <- readRDS("simres_r2.rds")
simres3 <- readRDS("simres_r3.rds")
```

Visualize the results

We can now extract the average variances within and between groups across all simulations for each p to obtain Figure 2 from [cite_preprint]:

```r
plot_df1 <- data.frame(
    p = 1:p_max,
    var_between = sapply(simres1, function(x) mean(x$var_between)),
    var_within = sapply(simres1, function(x) mean(x$var_within))
    )

# Find the critical value for the ratio F for n observations
# and for significance level 0.05
alpha <- 0.05  # Significance level
df1 <- 1  # Between-group degrees of freedom
df2 <- n-2  # Within-group degrees of freedom
critical_f_value <- qf(1 - alpha, df1, df2)
sig_threshold <- plot_df1$var_within * critical_f_value

pointsize <- 0.8

fig2_panel1 <- ggplot(data = plot_df1, aes(x = p)) +
        # shaded area in first layer indicates values for var_between where F > Fcrit
        geom_ribbon(
          aes(ymin = sig_threshold, ymax = Inf),
          fill = "grey80",
          alpha = .5
          ) +
        geom_point(
          aes(y = var_between, colour = "between groups"),
          size = pointsize
          ) +
        geom_point(
          aes(y = var_within, colour = "within groups"),
          size = pointsize,
          alpha = 0.8
          ) +
        scale_colour_manual(
            name = "Variance",
            values = c(
                "between groups" = "#00608b", # blue
                "within groups" = "#ffa600" # yellow-orange
                )
            ) +
        xlab("Number of potential confounders") +
        ylab("Variance") +
        theme_minimal() +
        theme(
            axis.title.x = element_text(size = 11),
            axis.text.x = element_text(size = 10, color = "black"),
            panel.grid.minor.x = element_blank(),
            panel.grid.minor.y = element_blank(),
            axis.title.y = element_text(size = 10)
            )
        #coord_fixed(ratio = 0.025, ylim = c(0, 9000))
```

```
fig2_panel1
```

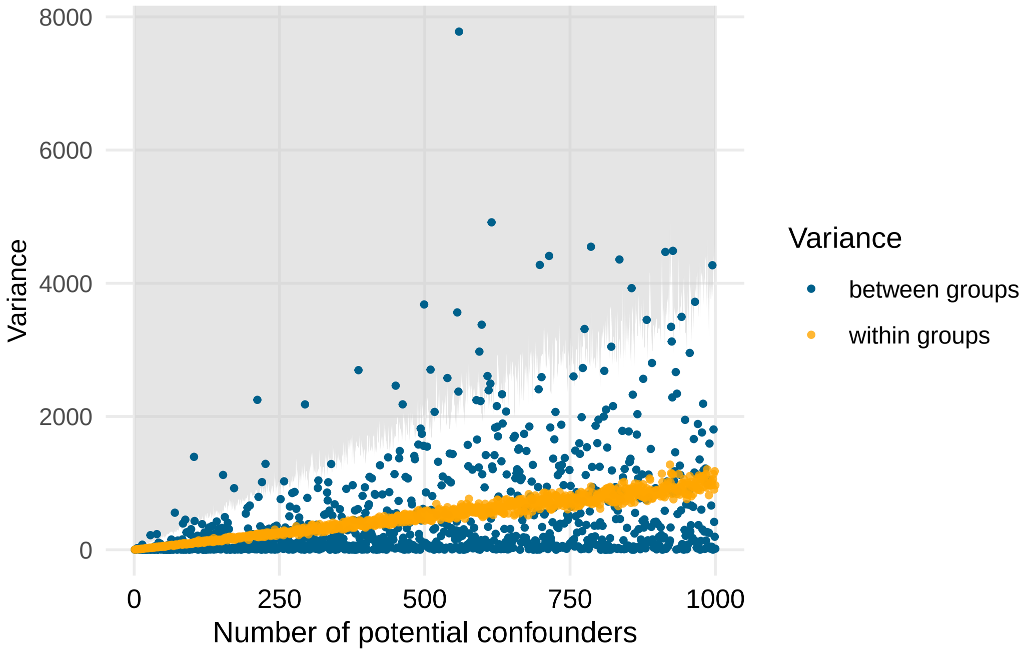


Figure 1

```
plot_df2 <- data.frame(
    p = 1:p_max,
    var_between = sapply(simres2, function(x) mean(x$var_between)),
    var_within = sapply(simres2, function(x) mean(x$var_within))
    )

fig2_panel2 <- ggplot(data = plot_df2, aes(x = p)) +
        geom_point(
          aes(y = var_between, colour = "between groups"),
          size = pointsize
          ) +
        geom_point(
          aes(y = var_within, colour = "within groups"),
          size = pointsize,
          alpha = 0.8
          ) +
        scale_colour_manual(
            name = "Variance",
            values = c(
                "between groups" = "#00608b", # blue
                "within groups" = "#ffa600" # yellow-orange
                )
            ) +
        xlab("Number of potential confounders") +
        ylab("Variance") +
        theme_minimal() +
        theme(
            axis.title.x = element_text(size = 11),
            axis.text.x = element_text(size = 10, color = "black"),
            panel.grid.minor.x = element_blank(),
            panel.grid.minor.y = element_blank(),
            axis.title.y = element_text(size = 10)
            )
        #coord_fixed(ratio = 0.1, ylim = c(0, 2250))

fig2_panel2
```

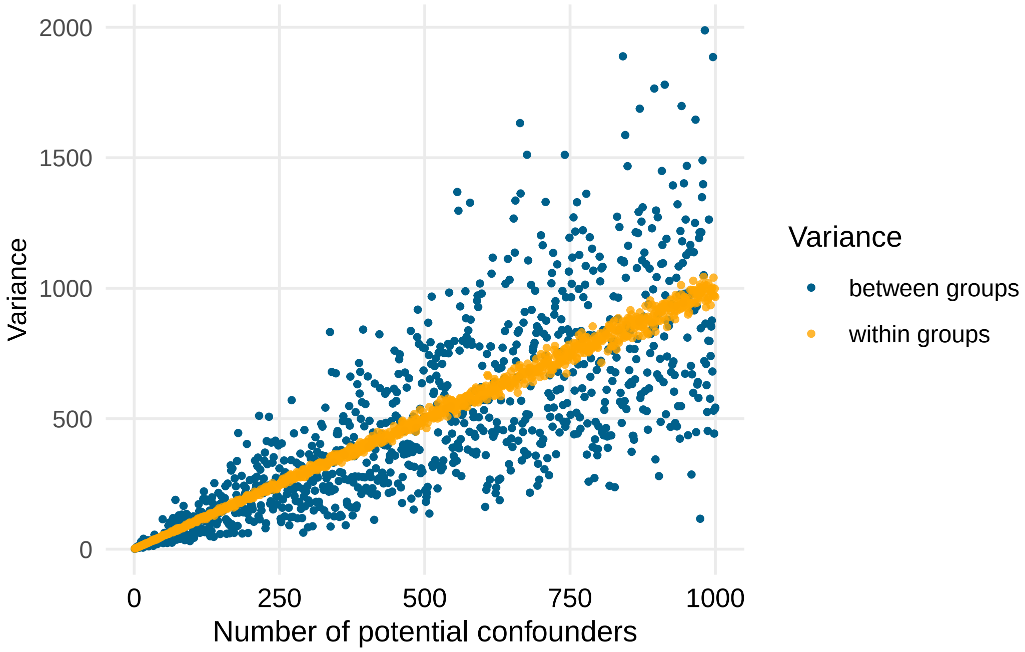


Figure 2

```
plot_df3 <- data.frame(
    p = 1:p_max,
    var_between = sapply(simres3, function(x) mean(x$var_between)),
    var_within = sapply(simres3, function(x) mean(x$var_within))
    )

fig2_panel3 <- ggplot(data = plot_df3, aes(x = p)) +
        geom_point(
          aes(y = var_between, colour = "between groups"),
          size = pointsize
          ) +
        geom_point(aes(y = var_within, colour = "within groups"),
                   size = pointsize/1.5) +
        scale_colour_manual(
            name = "Variance",
            values = c(
                "between groups" = "#00608b", # blue
                "within groups" = "#ffa600" # yellow-orange
                )
            ) +
        xlab("Number of potential confounders") +
        ylab("Variance") +
        theme_minimal() +
        theme(
            axis.title.x = element_text(size = 11),
            axis.text.x = element_text(size = 10, color = "black"),
            panel.grid.minor.x = element_blank(),
            panel.grid.minor.y = element_blank(),
            axis.title.y = element_text(size = 10)
            )
```

```
fig2_panel3
```

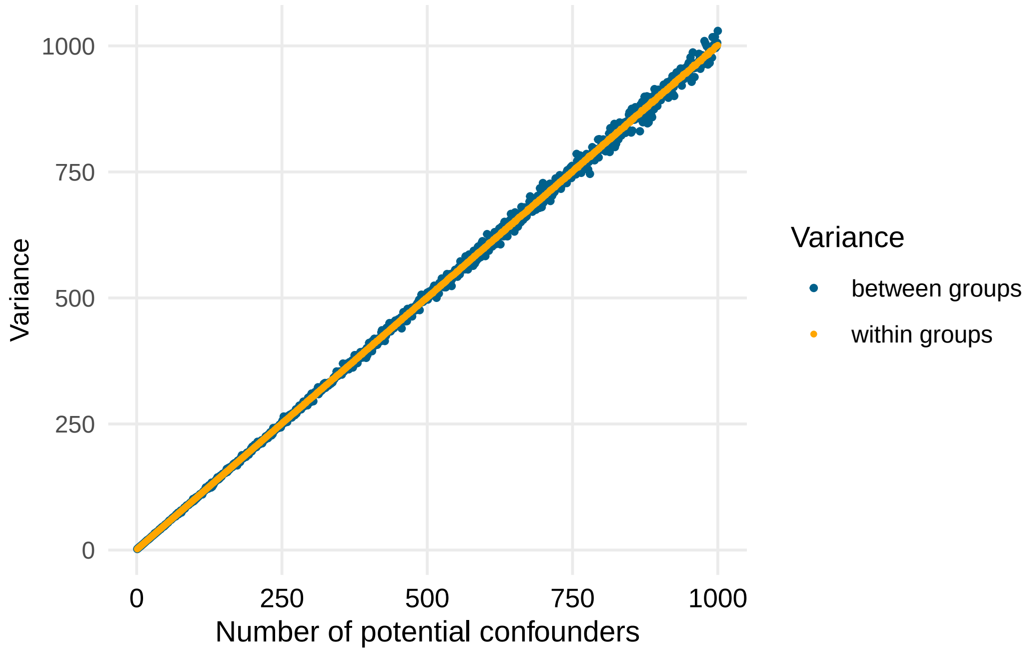


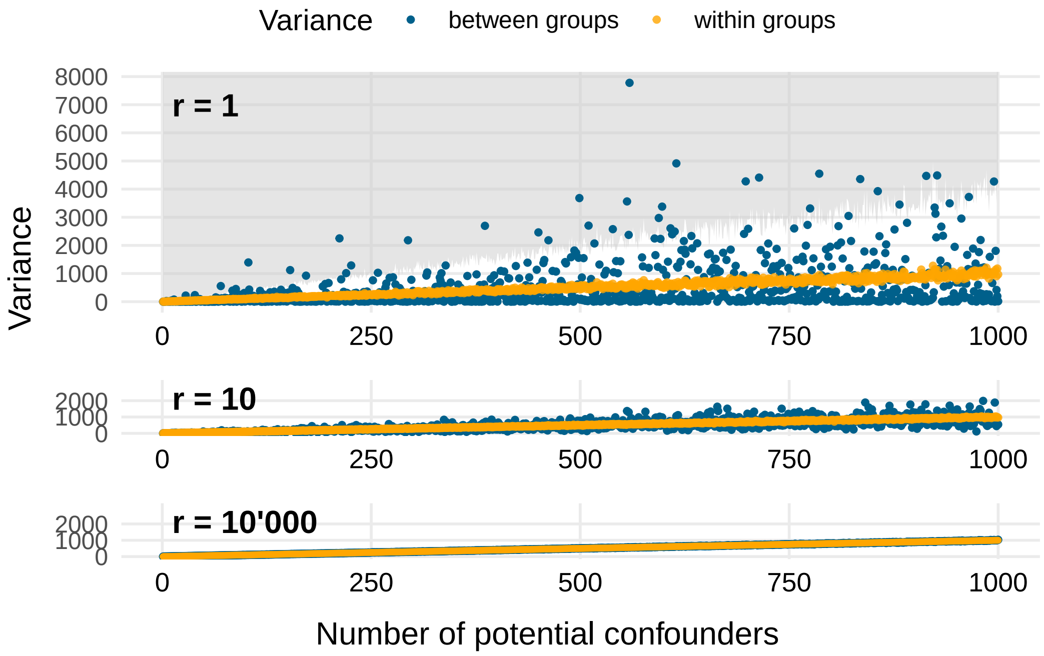
Figure 3

Combined Figure 2

```
# combine the three previously generated graphs in three panels, using ggarrange
fig2 <- ggarrange(
    fig2_panel1 + rremove("xlab") + rremove("ylab") + scale_y_continuous(
      limits = c(0, max(plot_df1$var_between)),
      breaks = seq(from = 0, to = 9000, by = 1000)),
    fig2_panel2 + rremove("ylab") + rremove("xlab") + scale_y_continuous(
        limits = c(0, max(plot_df1$var_between)/2.5),
        breaks = c(0, 1000, 2000)),
    fig2_panel3 + rremove("xlab") + rremove("ylab") + scale_y_continuous(
        limits = c(0, max(plot_df1$var_between)/2.5),
        breaks = c(0, 1000, 2000)),
    ncol = 1,
    heights = c(1, 1/2.5, 1/2.5),
    common.legend = TRUE,
    legend = "top",
    labels = c("r = 1", "r = 10", "r = 10'000"),
    label.x = 0.13,
    label.y = 0.92,
    hjust = 0,
    font.label = list(family = "Helvetica", face = "bold", size = 12)
    )

# add common axis labels
fig2_annotated <- annotate_figure(
  fig2,
  left = textGrob(
    "Variance",
    rot = 90, vjust = .5, hjust=0.1,
    gp = gpar(cex = 1, fontfamily = "Helvetica")),
  bottom = textGrob(
    "Number of potential confounders",
    gp = gpar(cex= 1, fontfamily = "Helvetica"))
  )

fig2_annotated
```



```
ggsave("figures/figure2_combined.pdf", fig2_annotated, height = 8, width = 7)
```