1 Pre-analysis simulations

Let $Y_i(0)$ be the sales price if house i is sold in an auction, whereas $Y_i(1)$ is the sales price if it is sold through a real estate agent. Our goal is to estimate the average percentage change in price if treated compared to being non-treated in the sample, which we call θ. For a sample of size n, we have

$$\theta = \frac{1}{n}\sum_{i=1}^{n} \frac{Y_i(1) - Y_i(0)}{Y_i(0)} \tag{1}$$

We will consider two different data generating processes. In the first, the individual treatment effect is proportional to $Y(0)$, i.e.,

$$Y_i(1) = \theta Y_i(0) \Rightarrow \theta = \frac{Y_i(1)}{Y_i(0)} \tag{2}$$

In the second case, the individual treatment effect is additive, i.e.,

$$Y_i(1) = Y_i(0) + \tau, \tag{3}$$

in which case we have

$$\theta = \frac{1}{n}\sum_{i=1}^{n} \frac{\tau}{Y_i(0)} \tag{4}$$

In the power analysis we will consider a number of different estimators. The first is a simple difference-in-means estimator evaluated at the average of the price in the control group. Let W_i be a treatment indicator, taking a value of one if house i is treated and 0 if not and $\widehat{Y(1)} = \frac{1}{n/2}\sum_{i=1}^{n} W_i Y_i(1)$ and $\widehat{Y(0)} = \frac{1}{n/2}\sum_{i=0}^{n}(1 - W_i)Y_i(0)$, the estimator is

$$\hat{\theta}_1 = \frac{1}{n/2}\left(\frac{\sum_{i=1}^{n} W_i Y_i(1) - \sum_{i=0}^{n}(1 - W_i)Y_i(0)}{\sum_{i=0}^{n}(1 - W_i)Y_i(0)}\right) \tag{5}$$

A second alternative is to use the log approximation, i.e.,

$$\hat{\theta}_2 = \frac{1}{n/2}\left(\sum_{i=1}^{n} W_i \log Y_i(1) - (1 - W_i)\log Y_i(0)\right) \tag{6}$$

To estimate the parameter θ, we rely on random assignment of treatment, in which case $Y(0)$ should be the same in both treatment and control group *in expectation.* However, for any given sample, there will always be some imbalance in $Y(0)$. By enforcing balance on observed covariates, more efficient strategies are available. Traditionally, stratified treatmeant assignment has been used to ensure balance on discrete covariates. Recently Morgan and Rubin (2012) proposed rerandomization as a method to ensure balance also on continuous covariates. They show that the variance in the estimation of the sample average treatment effect is inversely proportional to R^2 in the linear projection of $Y(0)$ on the observed covariates if the covariates are perfectly balanced. That means that if observed covariates are strong predictors of $Y(0)$, substantial efficiency gains can be made.

In our case, we expect the estimated market value (estimated before treatment assignment) to be a very strong predictor for sales price. Ideally, we would therefore like to ensure balance on this covariate. However, because houses are assigned to treatment or control one at a time, it is not possible use rerandomization. Instead, we can use regression adjustment to improve efficency after the experiment is carried out. For a recent discussion on regression adjustment in experiments, see, e.g., Lin (2013). Let X_i be the estimated market value of house i (estimated before treatment is assigned), we can simply run the following regression:

$$Y_i^{obs} = \hat{\alpha}_0 + \hat{\alpha}_1 X_i + \hat{\varepsilon}_i, \tag{7}$$

where Y_i^{obs} is $Y_i(1)$ for treated and $Y_i(0)$ for controls. The estimator for θ is

$$\hat{\theta}_3 = \frac{1}{n/2}\left(\frac{\sum_{i=1}^n W_i\hat{\varepsilon}_i - \sum_{i=1}^n (1-W_i)\hat{\varepsilon}_i}{\sum_{i=0}^n (1-W_i)Y_i(0)}\right), \tag{8}$$

where $\hat{\varepsilon}$ are the residuals from the regression in equation (7). An alternative is to include estimated market value and and a treatment indicator in the same regression:

$$Y_i^{obs} = \hat{\beta}_0 + \hat{\beta}_1 X_i + \hat{\beta}_2 W_i + \hat{u}_i, \tag{9}$$

with the corresponding estimator for θ:

$$\hat{\theta}_4 = \frac{\hat{\beta}_2}{\sum_{i=1}^n (1-W_i)Y_i(0)} \tag{10}$$

Finally, Imbens and Wooldridge (2009) suggest running the following regression:

$$Y_i^{obs} = \hat{\delta}_0 + \hat{\delta}_1(X_i - \bar{X}) + \hat{\delta}_2(X_i - \bar{X})W_i + \hat{\delta}_3 W_i + \hat{v}_i. \tag{11}$$

The estimator in our case is

$$\hat{\theta}_5 = \frac{\hat{\delta}_3}{\sum_{i=1}^n (1-W_i)Y_i(0)}. \tag{12}$$

Note that due to the randomization of W, the three estimators $\hat{\theta}_3$, $\hat{\theta}_4$ and $\hat{\theta}_5$ are very similar and should behave identically in large samples (for instance, Lin (2013) suggests that $\hat{\theta}_4$ and $\hat{\theta}_5$ are asymptotically equivalent in forced balanced designs, i.e. when the sample size is the same for treated and control).

We can also estimate the corresponding log-versions of these three estimators:

$$\log Y_i^{obs} = \hat{\alpha}_0' + \hat{\alpha}_1' \log X_i + \hat{\varepsilon}_i' \Rightarrow \tag{13}$$

$$\hat{\theta}_6 = \frac{1}{n/2}\left(\sum_{i=1}^{n} W_i \hat{\varepsilon}_i' - \sum_{i=1}^{n}(1-W_i)\hat{\varepsilon}_i'\right), \tag{14}$$

$$\log Y_i^{obs} = \hat{\beta}_0' + \hat{\beta}_1' \log X_i + \hat{\theta}_7 W_i + \hat{u}_i', \tag{15}$$

and

$$\log Y_i^{obs} = \hat{\delta}_0' + \hat{\delta}_1'(\log X_i - \overline{\log X}) + \hat{\delta}_2'(\log X_i - \overline{\log X})W_i + \hat{\theta}_8 W_i + \hat{v}_i'. \tag{16}$$

To perform a power analysis, we make use of data on previous auction sales from Kronofogden during the years 2010–2019. To mimic the proposed experiment we limit the sample to houses with an estimated market value greater than 200,000 SEK. In total, there are 3,151 such sales in our data. The data contain the sales price, the estimated market value and the tax value of each house.

Figure 1 shows the relationship between sales price and market value in this data. We see a strong positive relationship between sales price and market value as expected, and it also seems to be roughly linear. The R^2 from the regression is 0.907 suggesting that there is large scope for efficiency gains to be made by adjusting for market value. We can also note tax value has less predictive power (analysis not shown here, see the replication code) with a R^2 of 0.766. A multiple regression with both these variables leave the R^2 virtually unchanged (0.910) from just using market value. Hence we conclude that we only need to condition on market value.

As discussed above, we also consider regressing log price on log market value. Figure 2 shows this relationship. Once again, the relationship looks linear but also has less outliers. The R^2 is still very high but drops to 0.823.

Not all houses assigned to treatment will actually be sold with a real estate agent, but instead sold at auctions. That is, we will have one-sided noncompliance. Our best guess (although it is hard to know for sure) is that 20% of houses assigned to treatment will be sold at auctions. In simulations, we mimic this situation by letting each house assigned to treatment have a 80% chance of actually being treated. The estimand we use in the simulations will be the intention-to-treat (i.e., in expectation 0.8θ), but in the paper we will, in addition, also estimate the average treatment effect of the treated using IV. We do not expected there to be any large differences in power between the two and we therefore focus on the intention-to-treat here.

We let the treatment effect follow equation (2) or (3) for different values of θ and τ, and we begin by simulating the sampling distribution of the six different estimators by sampling n houses from the 3,151 observations with replacement.

Table 1 shows results from simulations based on 10,000 draws. Results are shown for two sample sizes, 700 and 2,000, for i) no treatment effect, ii) a multiplicative treatment effect of 10% and iii)

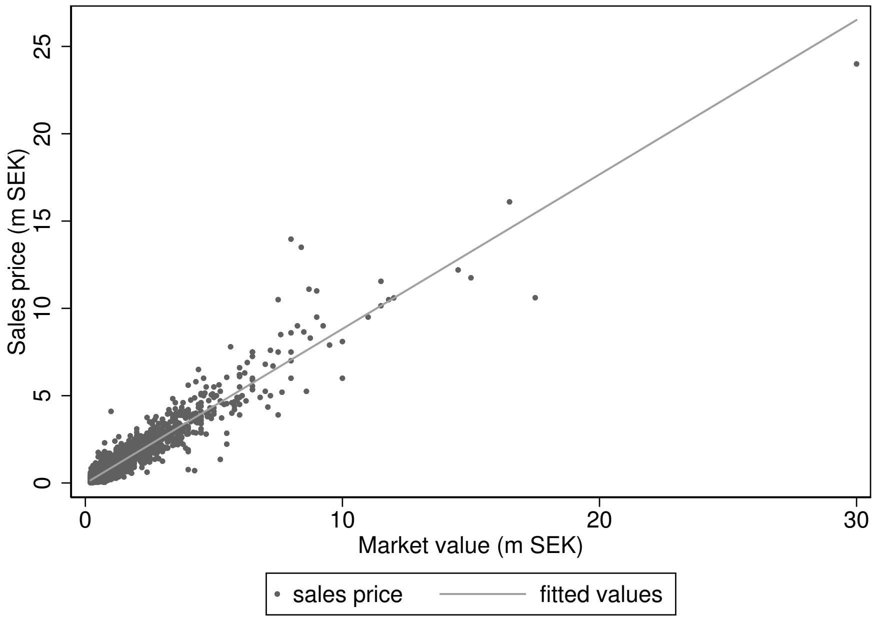


Figure 1: Sales price and market value

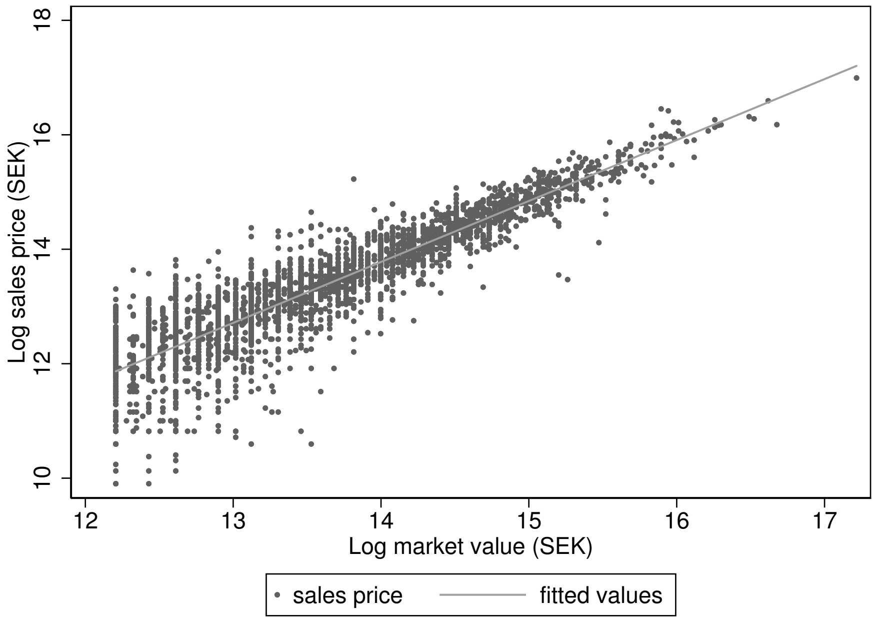


Figure 2: Log sales price and market value

a additive treatment effect of 50,000 SEK (implying a average percentage price increase of 13.8%). We let the treatment and control grous be of the same size.

Looking first at Panel A with no treatment effect, we see that all estimators are centered at 0 as they should be. As expected $\hat{\theta}_1$ and $\hat{\theta}_2$ are much less efficient compared to the other estimators and we also see that $\hat{\theta}_3 \approx \hat{\theta}_4 \approx \theta_5$ and $\hat{\theta}_6 \approx \hat{\theta}_7 \approx \theta_8$. In general, very little distinguishes the last six estimators.

Turning to Panel B we have a multiplicative treatment effect of 10%. We expect $\hat{\theta}_1$ $\hat{\theta}_3$, $\hat{\theta}_4$, $\hat{\theta}_5$ to be unbiased (w.r.t. the intention-to-treat) and they seem to be. The other four should only be approximately unbiased due to the log approximation and they very slightly underestimate the true effect, although the bias is negligible. Other than that, conclusions are very similar to those based on the results in Panel A.

Finally, in Panel C we see results when there is an additive treatment effect of 50,000 SEK, implying an average percentage increase of 13.8% and an intention-to-treat of 11%. We now see that $\hat{\theta}_1$, $\hat{\theta}_3$, $\hat{\theta}_4$ and $\hat{\theta}_5$ are clearly biased. The reason is that they estimate the percentage increase evaluated at the mean of $Y(0)$. Because the percentage effect is much greater for smaller values, and the distribution of houses is skewed to the left (so that the median house price is smaller than the mean) the average percentage price increase is greater than the percentage increase for the average priced house. The other estimators are slightly biased because of the log approximation.

Overall, we find that not controlling for market value is clearly inefficient, as we would expect, so we do not consider the estimators $\hat{\theta}_1$ and $\hat{\theta}_2$ any further. Furthermore, while $\hat{\theta}_3$, $\hat{\theta}_4$, $\hat{\theta}_5$ are good estimators when the true effect is multiplicative, they clearly fail when the effect is additive. While we believe a multiplicative effect is more plausible, we can not rule out that there could be some additive element in the effect, and we therefore conclude that the log estimators are more robust and therefore to be preferred. The difference betweem the three remaining estimators are negligible and they will be discussed in more detail in the power analysis below.

1.1 Power analysis

Here we perform a power analysis by considering the three estimators $\hat{\theta}_6$, $\hat{\theta}_7$ and $\hat{\theta}_8$. For each of the three estimators, we perform standard asymptotic inference (Neyman-Pearson inference, t-tests) as well as Fisher's exact test of the sharp null of no individual treatment effects.

Results are shown in Table 2 for the same treatment effects as in Table 1, for various sample sizes between 200 and 1,000. The first thing to note is that when there is no treatment effect the size of the test is always correct. When it comes to power, for a 10% treatment effect, acceptable power (80%) is achieved around a sample size of a little more than 1,000. For an additive treatment effect of 50,000, a sample size of 700 is more than enough to achieve acceptable power.

We also note that the power from the six different tests are virtually identical. Together with

Table 1: Monte Carlo simulations, 10,000 draws

		$n = 700$			$n = 2000$		
	0.8θ	mean	sd	95% interval	mean	sd	95% interval
Panel A. No treatment effect							
$\hat{\theta}_1$	0	.005	.098	-.174 – .210	.002	.058	-.108 – .123
$\hat{\theta}_2$	0	.000	.079	-.155 – .155	.000	.047	-.092 – .095
$\hat{\theta}_3$	0	.000	.030	-.056 – .061	-.000	.018	-.034 – .036
$\hat{\theta}_4$	0	.000	.030	-.056 – .061	-.000	.018	-.034 – .036
$\hat{\theta}_5$	0	.000	.030	-.057 – .061	-.000	.018	-.034 – .036
$\hat{\theta}_6$	0	-.000	.034	-.065 – .066	-.000	.020	-.040 – .040
$\hat{\theta}_7$	0	-.000	.034	-.065 – .066	-.000	.020	-.040 – .040
$\hat{\theta}_8$	0	-.000	.034	-.065 – .066	-.000	.020	-.040 – .040
Panel B. Multiplicative treatment effect, $\theta = 0.1$							
$\hat{\theta}_1$	.08	.085	.108	-.112 – .308	.083	.064	-.036 – .214
$\hat{\theta}_2$	.08	.076	.080	-.080 – .233	.078	.048	-.015 – .171
$\hat{\theta}_3$	.08	.079	.033	.017 – .145	.080	.019	.043 – .119
$\hat{\theta}_4$	.08	.079	.033	.017 – .146	.080	.019	.043 – .119
$\hat{\theta}_5$	.08	.080	.033	.017 – .147	.080	.019	.043 – .119
$\hat{\theta}_6$	.08	.076	.034	.011 – .143	.076	.020	.037 – .116
$\hat{\theta}_7$	.08	.076	.034	.011 – .143	.076	.020	.037 – .116
$\hat{\theta}_8$	.08	.076	.034	.011 – .143	.076	.020	.037 – .116
Panel C. Additive treatment effect, $\tau = 50,000$							
$\hat{\theta}_1$	.110	.041	.102	-.143 – .255	.037	.060	-.075 – .158
$\hat{\theta}_2$	.110	.096	.078	-.054 – .251	.095	.046	.006 – .186
$\hat{\theta}_3$	.110	.037	.031	-.022 – .099	.036	.018	.001 – .072
$\hat{\theta}_4$	.110	.037	.031	-.022 – .100	.036	.018	.001 – .072
$\hat{\theta}_5$	.110	.037	.031	-.023 – .100	.036	.018	.000 – .072
$\hat{\theta}_6$	.110	.095	.031	.033 – .157	.095	.019	.059 – .132
$\hat{\theta}_7$	.110	.095	.031	.034 – .157	.095	.019	.059 – .132
$\hat{\theta}_8$	.110	.095	.031	.033 – .157	.095	.019	.058 – .132

Note: The table shows the average point estimate, the standard deviation of the empirical sampling distribution, as well as a 95% interval of the empirical sampling distribution (2.5th percentile to 97.5th percentile).

Table 2: Monte Carlo power analysis, 10,000 draws

	$n = 200$	$n = 300$	$n = 400$	$n = 500$	$n = 700$	$n = 1,000$
Panel A. No treatment effect						
$\hat{\theta}_6$ (Fisher)	.052	.049	.049	.050	.048	.051
$\hat{\theta}_7$ (Fisher)	.052	.049	.049	.050	.048	.051
$\hat{\theta}_8$ (Fisher)	.052	.049	.049	.050	.048	.051
$\hat{\theta}_6$ (Neyman-Pearson)	.051	.049	.050	.050	.048	.050
$\hat{\theta}_7$ (Neyman-Pearson)	.051	.049	.050	.050	.047	.050
$\hat{\theta}_8$ (Neyman-Pearson)	.051	.050	.050	.050	.048	.050
Panel B. Multiplicative treatment effect, $\theta = .1$						
$\hat{\theta}_6$ (Fisher)	.226	.308	.402	.476	.615	.771
$\hat{\theta}_7$ (Fisher)	.226	.308	.402	.476	.615	.771
$\hat{\theta}_8$ (Fisher)	.226	.308	.402	.476	.615	.771
$\hat{\theta}_6$ (Neyman-Pearson)	.228	.310	.404	.476	.616	.773
$\hat{\theta}_7$ (Neyman-Pearson)	.228	.310	.404	.477	.615	.773
$\hat{\theta}_8$ (Neyman-Pearson)	.228	.309	.403	.476	.615	.773
Panel C. Additive treatment effect, $\tau = 50,000$						
$\hat{\theta}_6$ (Fisher)	.343	.507	.613	.705	.848	.948
$\hat{\theta}_7$ (Fisher)	.343	.507	.613	.705	.848	.948
$\hat{\theta}_8$ (Fisher)	.343	.507	.613	.705	.848	.948
$\hat{\theta}_6$ (Neyman-Pearson)	.342	.507	.610	.706	.849	.949
$\hat{\theta}_7$ (Neyman-Pearson)	.342	.507	.610	.706	.850	.949
$\hat{\theta}_8$ (Neyman-Pearson)	.345	.508	.612	.707	.850	.949

Note: Tests are performed at 5% significance level. For the Fisher test of the sharp null, the randomization distribution has been approximated with a draw of 1,000 randomly selected treatment assignments from one side of the lexicographic ordering (thereby implicitly including mirror allocations).

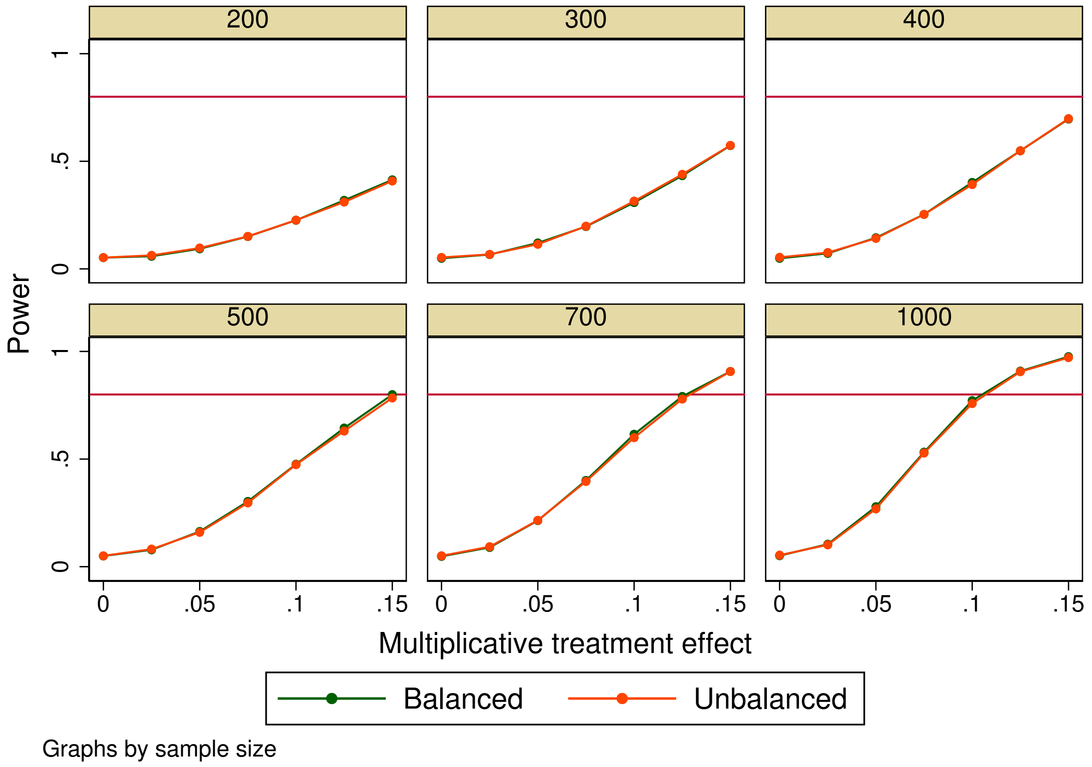
the results from Table 1, this fact suggests that it does not matter for the results which of the three estimators that are used and whether asymptotic or exact inference is used. For the rest of the analysis, we therefore restrict attention to one estimator ($\hat{\theta}_6$) using Fisher's exact test, though the results carry over to any of the other five tests.

We continue the power analysis by considering two different ways of splitting a sample into treatment and control. In the first, we continue with a balanced design with equal number houses in the treatment and control groups. In the second, we instead let $n/1.8$ houses go to treatment and $0.8n/1.8$ go to control, which would imply that the number of actually treated is the same as the number in the control group in expectation.

Figure 3 shows the result. The top panel shows results for multiplicative treatment effects, while the bottom panel shows for additive treatment effects. Each graph corresponds to a different sample size. The horizontal line is at 0.8 to indicate sufficient power. The first thing to note is that the balanced and unbalanced designs give virtually identical power in all situations. Therefore, the experiment will use a balanced design as it is the simplest to implement in the field. Second, to achieve sufficient power, a sample size of 500 is needed to detect a 15% price increase, a sample size of 700 is needed to detect a 12.5% price increase while a little more than 1,000 observations are needed for a 10% price increase. When the treatment effect is additive, a sample size of 700 is sufficient to detect a 50,000 SEK price increase.

2 Conclusions

When choosing the sample size to use, we need to weigh the need to get sufficient power with the cost and time of running the experiment. Balancing these two factors, we have chosen to run the experiment on a sample size of 700 which, in our estimation, means the experiment will run for approximately two years. We will use the balanced design with half the sample being assigned to treatment and half to control in the manner outlined in the registration form under the headline "Experimental Design". The analysis above suggests that the three estimators $\hat{\theta}_6$, $\hat{\theta}_7$ and $\hat{\theta}_8$ are virtually identical and that it does not matter whether Fisher or Neyman-Pearson inference is used. To test whether the treatment has any causal effect, we will use an exact test of the sharp null, as it does not rely on any asymptotic arguments. We will use the simplest estimator, $\hat{\theta}_6$, in that case. To be able to construct confidence interval we will also use regression to estimate $\hat{\theta}_8$, which is the most flexible estimator, and use standard asymptotic (Neyman-Pearson) inference.



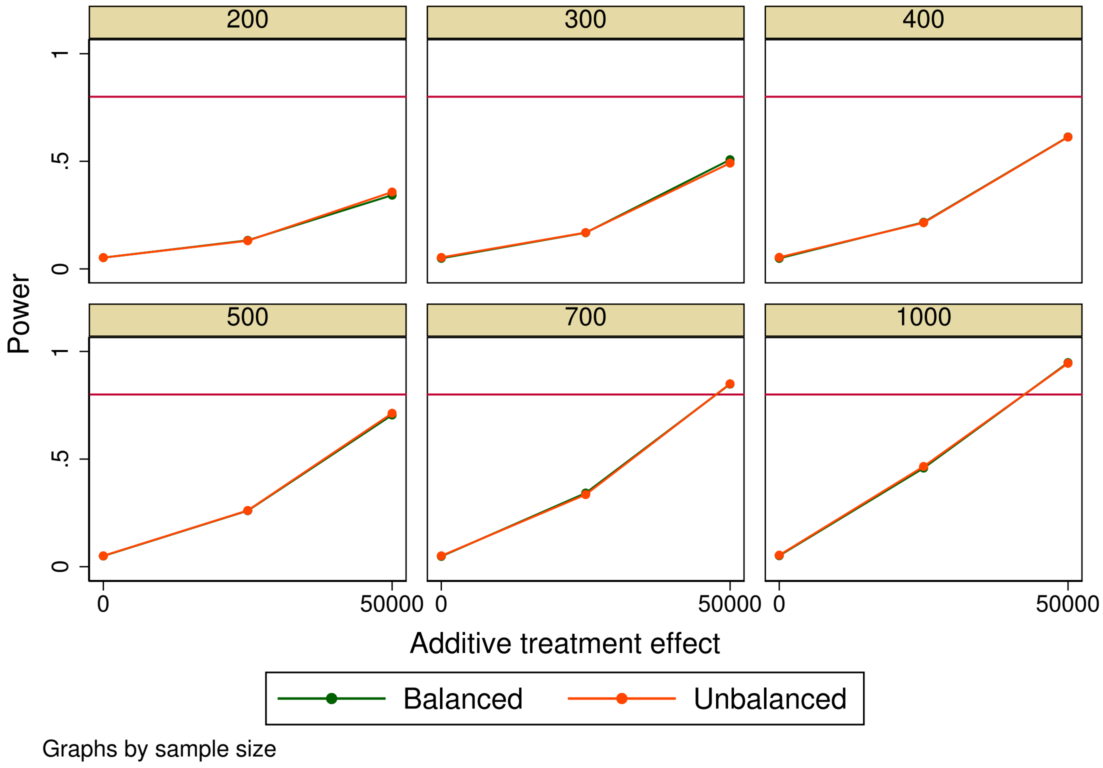


Figure 3: Power analysis for different sample sizes and treatment effects

References

Imbens, G. W. and Wooldridge, J. M. (2009). Recent developments in the econometrics of program evaluation. *Journal of Economic Literature*, 47(1):5–86.

Lin, W. (2013). Agnostic notes on regression adjustments to experimental data: Reexamining freedman's critique. *The Annals of Applied Statistics*, 7(1):295–318.

Morgan, K. L. and Rubin, D. B. (2012). Rerandomization to improve covariate balance in experiments. *The Annals of Statistics*, 40(2):1263–1282.

Appendix: code and data

We include code and data to replicate the above analysis. The data file is called `sim_data.csv` which include sales price, market value and tax value for Kronofogden sales of houses with estimated market value above 200,000 SEK during the period 2010–2019. We include three scripts: two Julia scripts and one Stata do-file script, all included in this appendix. Julia (v1.0.3) was used for the simulations and Stata (version 15) was used to generate figures:

1. `replication_code_table1.jl` reads `prep_data.csv` and produces `table1_sim.csv` which contains all the data for Table 1.

2. `replication_code_table2.jl` reads `prep_data.csv` and produces `table2_sim.csv` which contains all the data for Table 2 and Figure 3 (see below). Note that this script takes quite some time (roughly 20 hours) to run on a standard desktop computer.

3. `replication_code_figures.do` run regressions to get R^2 and generates Figures 1, 2 and 3. This script reads `table2_sim.csv`, so it requires `replication_code_table2.jl` to have run first to generate the data used for Figure 3.

replication_code_table1.jl

```
using CSV
using Random
using Statistics
using Distributions
using DataFrames

function get_estimates(y0, y1, x, n0, n1, share_never_takers)
    n = n0 + n1
    treat_index = collect(1:n1)
    control_index = collect(n1+1:n0+n1)
    yobs = gen_yobs(treat_index, control_index, n0, n1, share_never_takers,
                    y0, y1)

    theta1_hat = ((mean(yobs[treat_index]) - mean(yobs[control_index])) /
                   mean(yobs[control_index]))

    lnyobs = log.(yobs)
    theta2_hat = mean(lnyobs[treat_index]) - mean(lnyobs[control_index])

    x_t = hcat(x, ones(n))
    res = yobs .- x_t*inv(transpose(x_t)*x_t)*transpose(x_t)*yobs
    theta3_hat = ((mean(res[treat_index]) - mean(res[control_index])) /
                   mean(yobs[control_index]))

    b_est_sep = sep_reg(yobs, x_t, treat_index, control_index, n)
    theta4_hat = b_est_sep / mean(yobs[control_index])

    x_t_norm = copy(x_t)
    x_t_norm[:, 1] = x_t_norm[:, 1] .- mean(x_t_norm[:, 1])
    b_est_inter = inter_reg(yobs, x_t_norm, treat_index, control_index, n)
    theta5_hat = b_est_inter / mean(yobs[control_index])

    x_t_log = hcat(log.(x), ones(n))
    res_log = (lnyobs .- x_t_log*inv(transpose(x_t_log)*x_t_log) *
               transpose(x_t_log)*lnyobs)

    theta6_hat = mean(res_log[treat_index]) - mean(res_log[control_index])
    theta7_hat = sep_reg(lnyobs, x_t_log, treat_index, control_index, n)

    x_t_log_norm = copy(x_t_log)
    x_t_log_norm[:, 1] = x_t_log_norm[:, 1] .- mean(x_t_log_norm[:, 1])
    theta8_hat = inter_reg(lnyobs, x_t_log_norm, treat_index, control_index, n)

    return [theta1_hat, theta2_hat, theta3_hat, theta4_hat,
            theta5_hat, theta6_hat, theta7_hat, theta8_hat]
```

```
end

function gen_yobs(treat_index, control_index, n0, n1,
                  share_never_takers, y0, y1)
    nr_never_takers = rand(Binomial(n1, share_never_takers), 1)[1]
    never_takers_ind = sample(treat_index, nr_never_takers, replace=false)
    yobs = copy(y0)
    treated_index = setdiff(treat_index, never_takers_ind)
    yobs[treated_index] = y1[treated_index]
    return yobs
end

function sep_reg(Y, X, treat_index, control_index, n)
    t = zeros(n)
    t[treat_index] .+= 1
    x = hcat(copy(X), t)
    beta = inv(transpose(x)*x)*transpose(x)*Y
    return beta[3]
end

function inter_reg(Y, X, treat_index, control_index, n)
    t = zeros(n)
    t[treat_index] .+= 1
    x = hcat(copy(X), t)
    x = hcat(x, x[:, 1] .* x[:, 3])
    beta = inv(transpose(x)*x)*transpose(x)*Y
    return beta[3]
end

function sim_estimates(y, x, n0, n1, rep, N_pop, te_type, te, s_nt)
    n = n0 + n1
    y0 = copy(y)
    if te_type == "additive"
        y1 = copy(y) .+ te
    elseif te_type == "multiplicative"
        y1 = copy(y) .* (te + 1)
    end
    out = Array{Any, 2}(undef, rep, 8)
    for r = 1:rep
        samp = sample(1:N_pop, n)
        y1_s = y1[samp]
        y0_s = y0[samp]
        x_s = X[samp, :]
        out[r, :] = get_estimates(y0_s, y1_s, x_s, n0, n1, s_nt)
    end
    return out
end

function get_theta(y, te, te_type)
    if te_type == "multiplicative"
        return te
    elseif te_type == "additive"
        y0 = copy(y)
        y1 = y0 .+ te
        return mean((y1 .- y0) ./ y0)
    end
end

input_filepath = "sim_data.csv"
output_filepath = "table1_sim.csv"
df = CSV.read(input_filepath)

Y = Float64.(convert(Matrix, df)[:, 1])
X = Float64.(convert(Matrix, df)[:, 2])
N_POP = size(df)[1]

Random.seed!(12345)

REP = 10000
SHARE_NEVER_TAKERS = 0.2
TE_TYPE_LIST = ["multiplicative", "multiplicative", "additive"]
TE_LIST = [0, 0.1, 50000]
N0_LIST = [350, 1000]

for N0 = N0_LIST
    N1 = copy(N0)
    for i = 1:length(TE_LIST)
        TE = TE_LIST[i]
        TE_TYPE = TE_TYPE_LIST[i]
        estimates = sim_estimates(Y, X, N0, N1, REP, N_POP, TE_TYPE,
                                  TE, SHARE_NEVER_TAKERS)

        theta = get_theta(Y, TE, TE_TYPE)
        itt = (1 - SHARE_NEVER_TAKERS) * theta
        out = Array{Any, 2}(undef, 8, 11)
        for i = 1:8
            out[i, :] = [mean(estimates[:, i]), std(estimates[:, i]),
                         quantile(estimates[:, i], 0.025),
                         quantile(estimates[:, i], 0.975),
                         i, theta, itt, N0, N1, TE, TE_TYPE]
        end
        df_out = DataFrame(out)
        names!(df_out, Symbol.(["theta_hat", "sd_theta_hat", "lb_ci_theta_hat",
                                "ub_ci_theta_hat", "estimator_nr", "theta",
                                "itt", "n0", "n1", "treatment_effect",
                                "treatment_type"]))
        if isfile(output_filepath) == false
            CSV.write(output_filepath, df_out)
        else
            CSV.write(output_filepath, df_out, append=true)
        end
    end
end
```

replication_code_table2.jl

```
using Distributions
using CSV
using Random
using Statistics
using DataFrames

function fisher_p_reg(y0, y1, x, n0, n1, set_size, share_never_takers)
    n = n0 + n1
    treat_index = collect(1:n1)
    control_index = collect(n1+1:n0+n1)
    yobs = gen_yobs(treat_index, control_index, n0, n1,
                    share_never_takers, y0, y1)
    lnyobs = log.(yobs)
    lnx = copy(x)
    lnx[:, 1] = log.(x[:, 1])

    res = lnyobs .- lnx*inv(transpose(lnx)*lnx)*transpose(lnx)*lnyobs

    res1 = sum((res[treat_index] .- mean(res[treat_index])).^2)
    res0 = sum((res[control_index] .- mean(res[control_index])).^2)
    THETA6_HAT = Array{Float64}(undef, set_size)
    THETA7_HAT = Array{Float64}(undef, set_size)
    THETA8_HAT = Array{Float64}(undef, set_size)
    se7 = 0
    se8 = 0
    for i = 1:set_size
        if i == 1
            THETA7_HAT[i], se7 = sep_reg(lnyobs, lnx, treat_index,
                                         control_index, n, 1)
            THETA8_HAT[i], se8 = inter_reg(lnyobs, lnx, treat_index,
                                           control_index, n, 1)
        end
        THETA6_HAT[i] = mean(res[treat_index]) - mean(res[control_index])
        THETA7_HAT[i] = sep_reg(lnyobs, lnx, treat_index, control_index, n, 0)
        THETA8_HAT[i] = inter_reg(lnyobs, lnx, treat_index, control_index, n, 0)
        ti = sample(2:n, n1 - 1, replace=false)
        treat_index = append!([1], ti)
        control_index = setdiff(1:n, treat_index)
    end
    THETA6_HAT_ABS = abs.(THETA6_HAT)
    THETA7_HAT_ABS = abs.(THETA7_HAT)
    THETA8_HAT_ABS = abs.(THETA8_HAT)

    se6 = sqrt((res1 + res0) / (n0 + n1 - 2)) * sqrt(1 / n0 + 1 / n1)
    t6 = abs(THETA6_HAT_ABS[1]  / se6)
    rank_sort6 = sortperm(THETA6_HAT_ABS, rev=true)
    rank_sort7 = sortperm(THETA7_HAT_ABS, rev=true)
    rank_sort8 = sortperm(THETA8_HAT_ABS, rev=true)
    p_val6f = findall(x->x==1, rank_sort6)[1] / set_size
    p_val7f = findall(x->x==1, rank_sort6)[1] / set_size
    p_val8f = findall(x->x==1, rank_sort6)[1] / set_size
    p_val6np = ccdf(TDist(n0 + n1 - 3), t6) * 2
    p_val7np = ccdf(TDist(n0 + n1 - 3), THETA7_HAT_ABS[1]  / se7) * 2
    p_val8np = ccdf(TDist(n0 + n1 - 4), THETA8_HAT_ABS[1]  / se8) * 2
    power6f = signif(p_val6f)
    power7f = signif(p_val7f)
    power8f = signif(p_val8f)
    power6np = signif(p_val6np)
    power7np = signif(p_val7np)
    power8np = signif(p_val8np)

    return [power6f, power7f, power8f, power6np, power7np, power8np]
end

function gen_yobs(treat_index, control_index, n0, n1,
                  share_never_takers, y0, y1)
    nr_never_takers = rand(Binomial(n1, share_never_takers), 1)[1]
    never_takers_ind = sample(treat_index, nr_never_takers, replace=false)
    yobs = copy(y0)
    treated_index = setdiff(treat_index, never_takers_ind)
    yobs[treated_index] = y1[treated_index]
    return yobs
end

function signif(p_val)
    if p_val <= 0.05
        sign = 1
    else
        sign = 0
    end
    return sign
end

function inter_reg(Y, X, treat_index, control_index, n, se_ind)
    t = zeros(n)
    t[treat_index] .+= 1
    x = copy(X)
    x[:, 1] = x[:, 1] .- mean(x[:, 1])
    x = hcat(x, t)
    x = hcat(x, x[:, 1] .* x[:, 3])
    beta = inv(transpose(x)*x)*transpose(x)*Y
    if se_ind == 1
        mse = sum((Y .- x * beta).^2) / (n - 4)
        se_est = sqrt(mse * inv(transpose(x) * x)[3, 3])
        return beta[3], se_est
    else
        return beta[3]
    end
end

function sep_reg(Y, X, treat_index, control_index, n, se_ind)
    t = zeros(n)
    t[treat_index] .+= 1
    x = hcat(copy(X), t)
    beta = inv(transpose(x)*x)*transpose(x)*Y
    if se_ind == 1
        mse = sum((Y .- x * beta).^2) / (n - 3)
        se_est = sqrt(mse * inv(transpose(x) * x)[3, 3])
        return beta[3], se_est
    else
        return beta[3]
    end
end

function sim_p(Y, X, n0, n1, set_size, rep, N_pop, te_type, te, s_nt)
    n = n0 + n1
    Y0 = copy(Y)
    if te_type == "additive"
        Y1 = copy(Y) .+ te
    elseif te_type == "multiplicative"
        Y1 = copy(Y) .* (te + 1)
    end
    out = Array{Any, 2}(undef, rep, 6)
    for r = 1:rep
        samp = sample(1:N_pop, n)
        y1 = Y1[samp]
        y0 = Y0[samp]
        x = X[samp, :]
        out[r, :] = fisher_p_reg(y0, y1, x, n0, n1, set_size, s_nt)
    end
    return out
end

input_filepath = "sim_data.csv"
output_filepath = "table2_sim.csv"
df = CSV.read(input_filepath)
```

```
Y = Float64.(convert(Matrix, df)[:, 1])
X = Float64.(hcat(convert(Matrix, df)[:, 2], ones(length(Y))))
N_POP = size(df)[1]

Random.seed!(12345)

SET_SIZE = 1000
REP = 10000
SHARE_NEVER_TAKERS = 0.2
TE_TYPE_LIST = vcat(fill.(["multiplicative", "additive"], [7, 2])...)
TE_LIST = [0, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 25000, 50000]
N_LIST = [200, 300, 400, 500, 700, 1000]
N1_LIST = round.(Int, 0.5 .* N_LIST)
N1_LIST = append!(N1_LIST, round.(Int, 1 / 1.8 .* N_LIST))
N0_LIST = append!(N_LIST, N_LIST) .- N1_LIST

for j = 1:length(N1_LIST)
    N0 = N0_LIST[j]
    N1 = N1_LIST[j]
    for i = 1:length(TE_LIST)
        TE = TE_LIST[i]
        TE_TYPE = TE_TYPE_LIST[i]
        power = sim_p(Y, X, N0, N1, SET_SIZE, REP, N_POP,
                      TE_TYPE, TE, SHARE_NEVER_TAKERS)
        out = [mean(power[:, 1]), mean(power[:, 2]), mean(power[:, 3]),
               mean(power[:, 4]), mean(power[:, 5]), mean(power[:, 6]),
               N0, N1, TE, TE_TYPE]

        df_out = DataFrame(reshape(out, 1, 10))
        names!(df_out, Symbol.(["power6f", "power7f", "power8f",
                                "power6np", "power7np", "power8np",
                                "n0", "n1", "treatment_effect",
                                "treatment_type"]))
        if isfile(output_filepath) == false
            CSV.write(output_filepath, df_out)
        else
            CSV.write(output_filepath, df_out, append=true)
        end
    end
end
```

replication_code_figures.do

```
import delimited using "sim_data.csv", clear

* Run regressions to get $R^2$
reg price market_value
reg price tax_value
reg price market_value tax_value

gen lnprice = ln(price)
gen lnmarket_value = ln(market_value)
reg lnprice lnmarket_value

* Generate figures
twoway || scatter price market_value, msize(vsmall) ///
       || lfit price market_value ||, ///
       scheme(s1mono) legend(order(1 "sales price" 2 "fitted values")) ///
       xtitle("Market value (m SEK)") ytitle("Sales price (m SEK)")

twoway || scatter lnprice lnmarket_value, msize(vsmall) ///
       || lfit lnprice lnmarket_value ||, ///
       scheme(s1mono) legend(order(1 "sales price" 2 "fitted values")) ///
       xtitle("Log market value (SEK)") ytitle("Log sales price (SEK)")


* Read file generated by Julia script "replication_code_table2.jl"
import delimited using "table2_sim.csv", clear

gen sample_size = n0 + n1
label variable sample_size "sample size"

twoway || connected power6f treatment_effect if n0 == n1, msize(small) ///
       || connected power6f treatment_effect if n0 != n1, msize(small) ///
       || if treatment_type == "multiplicative" , scheme(s1color) ///
          by(sample_size) ytitle("Power") yline(0.8) ///
          xtitle("Multiplicative treatment effect") ///
          legend(order(1 "Balanced" 2 "Unbalanced"))

replace treatment_type = "additive" if treatment_effect == 0

twoway || connected power6f treatment_effect if n0 == n1, msize(small) ///
       || connected power6f treatment_effect if n0 != n1, msize(small) ///
       || if treatment_type == "additive" , scheme(s1color) ///
          by(sample_size) ytitle("Power") yline(0.8) ///
          xtitle("Additive treatment effect") ///
          legend(order(1 "Balanced" 2 "Unbalanced"))
```