REPLICATION MATERIAL FOR: DEFAULTS AND DONATIONS: EVIDENCE FROM A FIELD EXPERIMENT

ALTMANN, FALK, HEIDHUES, JAYARAMAN, & TEIRLINCK

The following README file describes the data and code required to replicate the empirical results in the article "Defaults and Donations: Evidence from a Field Experiment."

1. DATA

The file *defaults_donations.dta* contains the raw data needed to generate all results reported in the paper. Variable descriptions are provided in the data dictionary in Appendix A of this document.

2. SOFTWARE

Results from the article were produced using Stata/SE version 14.2 and, for the structural estimation, Matlab version R2017b. Additional toolboxes required to replicate the structural estimation exercise in Matlab are the (i) Global Optimization Toolbox, (ii) Optimization Toolbox and (iii) Statistics and Machine Learning Toolbox.

3. CODE FOR TABLES AND FIGURES, NOT INCLUDING SECTION 4

Once you have saved *defaults_donations.dta* to a local folder:

- Executing the file *Figures.do* produces all figures from Section 3 of the article.
- Executing the file *Tables_Tests.do* produces all tables, statistical tests, and further empirical analyses reported in Sections 2, 3, and the appendix of the article.

In order to execute these files, you will need insert your local directory at the top of the respective *.do* file.

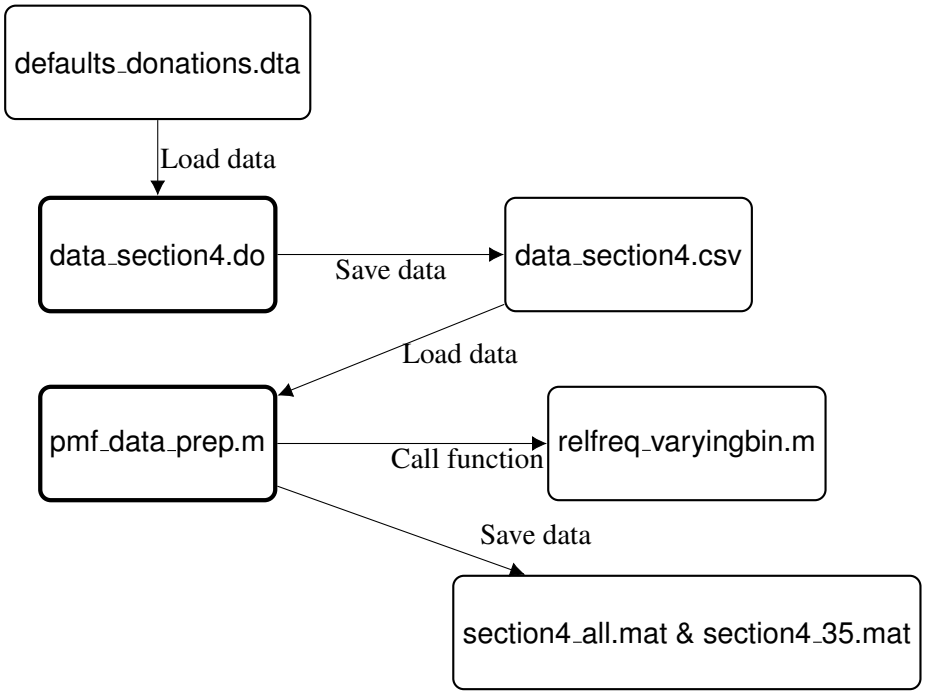
4. CODE FOR STRUCTURAL ESTIMATION IN SECTION 4

This section provides a detailed description of the steps needed, as well as the code, for reproducing the structural estimates in Section 4 of the article. The different code files are listed in order of use when possible. Graphics are included that describe the order and combination of code files to use to reproduce the results in the paper.

In order to execute these files, you will need insert your local directory at the top of the respective *.m* file.

Date: August 2018.

A. Prepare data



4.1. **data_section4.do.** This STATA do file opens the dataset *defaults_donations.dta* and saves the required variables for the structural estimation in a csv file. The variables that are selected are *donated_amount*, which is the donated amount in Euros and *default_donation*, which is the default treatment (AD:0, 10, 20 or 50). These two variables are then saved in the file *data_section4.csv*.

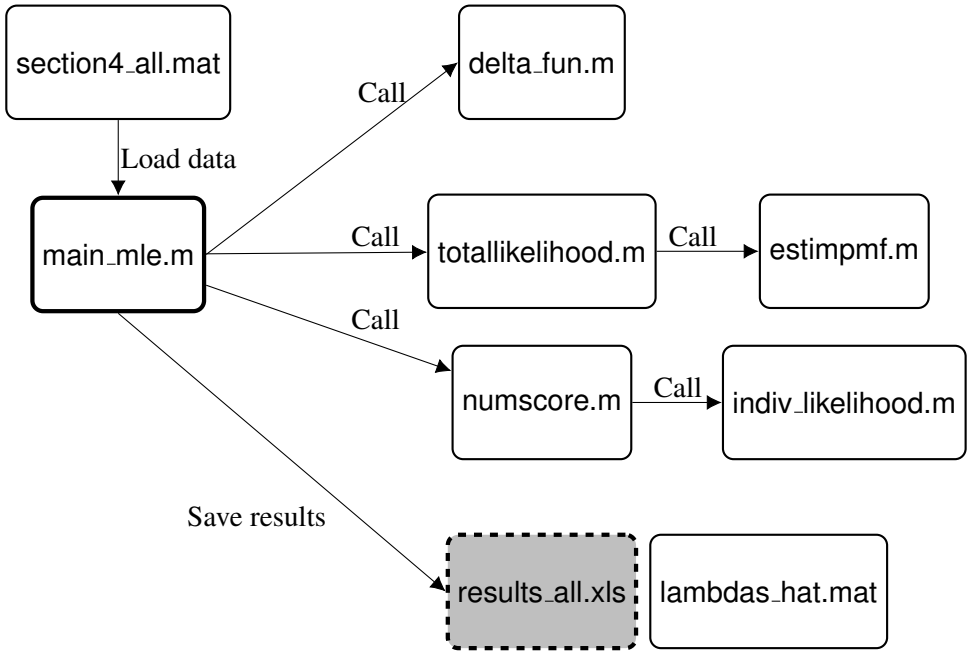
4.2. **pmf_data_prep.m.**

- Opens the dataset *data_section4.csv* using the matlab function csvread.
- Saves two datasets, one with all observations included (section4_all.mat), and one with all nonzero donations and including up to 3.5% of zero contributions (section4_35.mat).
- Most of the donations take on integers values, but because there are few exception, the function round is used to round all donations to integer values (and thus are assigned to integer bins).
- Calls the function *relfreq_varyingbin.m* to identify the probability mass function of rho. It constructs the relative frequency with varying bandwidth for observations of donations up to €300.
- Save matlab dataset *section4_all.mat* and *section4_35.mat*, containing donation, default and relfreq. Donation is the amount donated in integer values. Default is the default treatment corresponding to the observation (AD:0, 10, 20, 50) and relfreq is the probability mass for integer bins from 0 up to 300 (the first element corresponds to the integer bin 0).

4.3. **relfreq_varyingbin.m.**

- Function to identify probability mass function of the donations data.
- Input: donations under AD treatment.
- Output: relfreq- the relative frequency at each integer value; count- count at each integer value; relcount- count per bandwidth interval, divided by bandwidth; total- total number of observations; bandwidth- width of each bin.
- Donations are rounded to nearest integer value; relative frequency from €0 to €300 donations.
- The probability mass function needs to be nonzero at each bin (due to the log likelihood function). There are two characteristics of the distribution of donation under the AD treatment that the probability mass function should capture: 1) The data is sparse in the upper tail. 2) there are specific patterns in the data, such as mass in factors of ten and (to a lesser extent) five. The varying bandwidth takes this into account in order to portray the patterns of donation in the AD group without putting too much emphasis on arbitrary numbers between factors of five and ten. Because of sparsity at higher donations, the bandwidth increases, while preserving a unit bandwidth at (some) factors of ten.
- The function assigns a bandwidth of 1: up to a value of 30
- The function assigns a bandwidth of 4: for values between 30 and 100 that are not a fivefold integer. For Fivefold integers a bandwidth of 1 is assigned.
- The function assigns a bandwidth of 9: for values between 100 and 200 that are not factors of ten, and for factors of ten, a bandwidth of 1 is assigned.
- The function assigns a bandwidth of 10-10-10-9 from 201 up till 249 and 251 up till 299 and a bandwidth of 1 for 200, 250 and 300.

B. Maximum likelihood estimation



4.4. **main_mle.m.**

- Main file for running maximum likelihood estimation.
- Call function *delta_fun*: to calculate and store the gain from opting out.
- Exclude donations under AD from estimation.
- Subsample of donations up to 300.
- Specify starting values and options for the minimization with Fmincon with global search.
- Use the function minimizer Fmincon with global search to find the values for lambda that minimize the output of the function *totallikelihood*.
- The Hessian is estimated using the BHHH [1] approach (outer product of gradients). [2]
- Call on the function *numscore* to calculate the score vector numerically. The BHHH approximation to the Hessian is the the outer product of the score vector. This can then be used to obtain the standard errors. Note that using the Hessian approximated by Fmincon is not recommended.
- Save estimated lambdas, their standard deviation and log-likelihood in results_all.xls and lambdas_hat.mat.
- Repeat same procedure, but with rescaled parameters. Similar results are obtained.[3]

[1]Berndt, Ernst R, Bronwyn H Hall, Robert E Hall, and Jerry A Hausman. 1974. Estimation and Inference in Nonlinear Structural Models. In Annals of Economic and Social Measurement, Volume 3, number 4. NBER, 653-665.

[2]For a description of the BHHH procedure see Kenneth Train (2003), Discrete Choice Methods with Simulation, SUNY-Oswego, Department of Economics, https://EconPapers.repec.org/RePEc:oet:tbooks:emetr2 p.220

[3]Similar results are also obtained when estimating the lambdas with 3.5% of observations, including a large bulk of the zeros.

4.5. **totallikelihood.m.**

- Log-likelihood function. A detailed description of the derivation of this function can be found in the Online Appendix, section D.1.
- Inputs: lambda, donations under 10, 20 and 50 defaults, vector with corresponding defaults, Delta, relfreq, c1 and c2 (rescaling parameters).
- Outputs: negative of the total log-likelihood (to be minimized).
- Calls function *estimpmf* to match each observation to the corresponding probability mass estimated at that integer bandwidth.

4.6. **numscore.m.**

- Function that numerically approximates the score vector. The score vector contains the score of each observation, which is the derivative of that observation's log-likelihood with respect to the parameters.
- Inputs:
  1. lambda_0 the vector with the estimated parameters.
  2. The vectors needed to calculate the log-likelihood, namely: rho, Deltas, defaults, the probability mass function *relfreq*, and the rescaling parameters c1 and c2.
  3. param indicates the parameter for which the score is calculated (1, 2 or 3).
  4. dx a small step to approximate the derivative.
- *Numscore* then calls the function *indiv_likelihood.m* which calculates the vector with the individual log-likelihoods at the specified parameters.
- The score for each individual is approximated by $\frac{(f1-f0)}{dx}$.

4.7. **indiv_likelihood.m.**

- The function that calculates the vector with the individual log-likelihoods.
- Similar to *totallikelihood.m* except that the individual likelihoods are not summed.

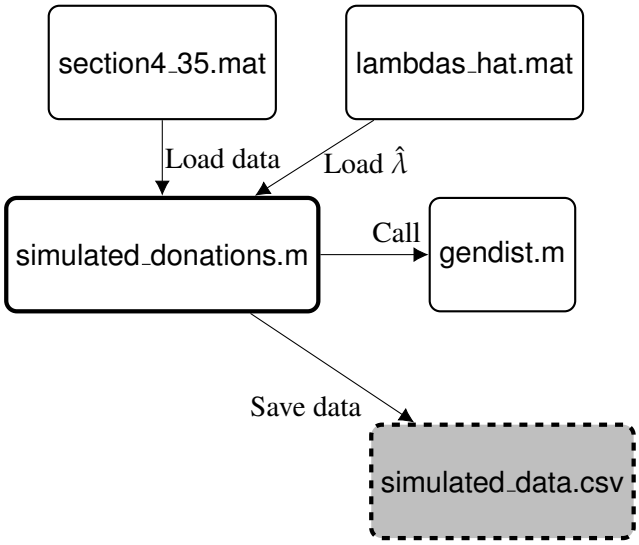
4.8. **delta_fun.m.**

- Calculates the gain from deviating from the default given the optimal $x^* = \rho$.
- Set values to missing for AD group (NaN).

4.9. **estimpmf.m.**

- Assigns to each observation the corresponding probability mass computed at that integer bin.
- Input: y- computed probability mass function for integer donation values on the interval [0, 300]; x- donation data.
- Output: vector of length(x) with assigned probability mass corresponding to each donation observation.

C. Simulation



4.10. **simulated_donations.m.**

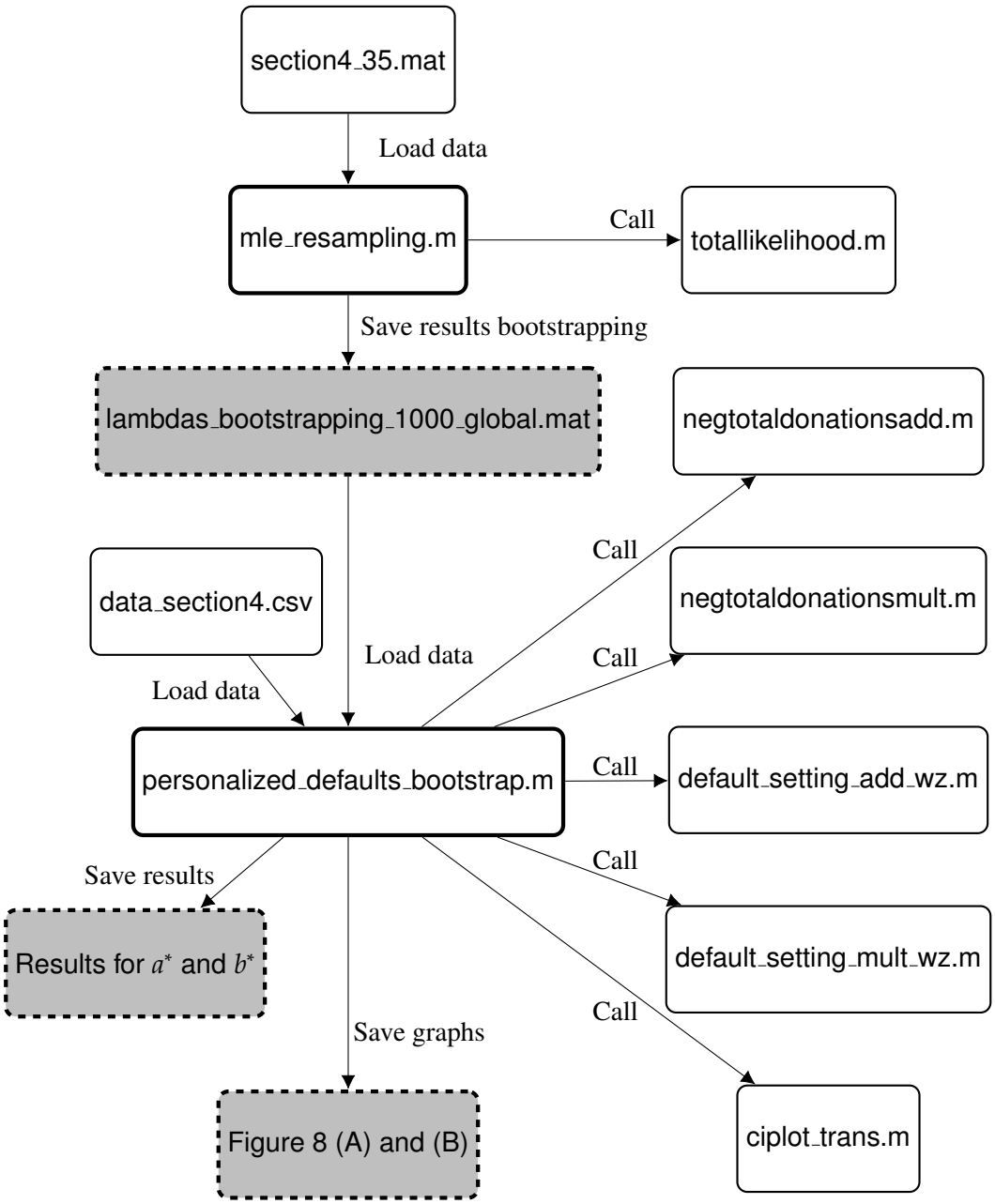
- Load *section4_35.mat*. Use dataset with all positive donations plus a share of the zero contributions (in total 3.5% of observations are included, this is done for illustrative reasons. Note that the simulation results will not change as the ungenerous agents with $\rho = 0$ do not change their behavior). Keep only donations from AD treatment.
- Simulate donations from the *relfreq* probability mass function by using the function *gendist.m*. We draw a random sample with N=500,000.
- Load the estimated lambdas. Set the lambda parameters to the estimated lambdas.
- Simulate the share of agents that do not face any deviation costs. For a random share 1-λ_1 of agents the indicator z equals 1, implying that these agents do face deviation costs.
- For the share 1-λ_1 (with $z = 1$), a deviation cost δ is randomly drawn from the exponential distribution with parameter λ_2 . Similarly, an opt-out cost α is randomly drawn from the exponential distribution with parameter λ_3.
- For each simulated agent, the utility for the three option is calculated. U_x= utility for picking ρ. U_d utility from donating the default and U_0 utility from opting out completely.
- Save simulated rho, delta, alpha, z, and the utility at different choices for each agent in *data_for_simulation.m*. Then simulate choices when default= 10, 20 and 50; After each simulation with the different defaults, the simulated data is appended to and saved in *simulated_data.mat*
- The simulated data is then saved in *simulated_data.csv* by using csvwrite.

4.11. **gendist.m.**

- Function to generate random numbers from a discrete distribution.

– This function is written by Tristan Ursell (2012) and can be obtained through the following link: https://mathworks.com/matlabcentral/fileexchange/34101-random-numbers-from-a-discrete-distribution

D. Personalized defaults



4.12. **mle_resampling.m.**

- Load *section4_35.mat*. We use the dataset that excludes a large share of the zeros contributions to speed up the bootstrapping procedure. Note that the lambda estimates are similar when using the 3.5% sample (that contains all nonzero donations plus zero contributions up to 3.5% of the sample) as when using the whole sample. This is because identification of the parameters comes from the nonzero donations and extensive margin movement, which are both still captured in the 3.5% sample.
- Bootstrapping: draw samples with replacement from the treatment groups with default being 10, 20 and 50. With these bootstrap samples, re-estimate the lambdas using global search with Fmincon. The function *totallikelihood.m* is minimized. Repeat this 1000 times and save the estimated lambdas in *lambdas_bootstrapping_1000_global.mat*.

4.13. **personalized_defaults_bootstrap.m.**

- Calculate optimal default, based on rho. Set default= $\rho + a$ and $\rho * b$. Find a and b that maximize total donations.
- Load the estimated lambdas obtained by bootstrapping (with 1000 bootstrap samples).
- Take a random sample from the AD treatment, N=100,000.
- For each set of estimated lambdas, minimize the *negtotaldonationsadd.m*. This function returns the negative of total donations under the policy $\rho + a$. Find the value of a that maximizes total donations. The estimate of a^* is then the average of these estimated a's obtained with the lambda bootstrap estimates.
- For Figure 8 (A), the plot of the average donations with confidence interval under the policy $\rho + a$, we draw a large sample of ρ's for simulation to derive a good approximation of the α and δ distributions.
- Then for each set of estimated lambdas obtained by bootstrapping we simulate donations when the personalized defaults are set to $\rho + a$, a ranging from 0 to 80, using the function *default_setting_add_wz.m*, and store the means. These are then used to construct the plot and confidence interval. By using the lambdas obtained by bootstrapping, the confidence interval takes into account the uncertainty that comes from using estimated parameters.
- For the policy rho*b, repeat this procedure. For each of the set of estimated lambdas, minimize the function *negtotaldonationsmult.m*. This function returns the negative of the total of donations under the policy $\rho * b$ Find the value of b that maximizes total donations.
- To create Figure 8 (B), the plot of the average donation under the policy $\rho * b$ with 95% confidence interval, a large sample of rho's is drawn for simulation to approximate well the alpha and delta distributions.
- For each set of estimated lambdas obtained by bootstrapping we simulate donations when the personalized defaults are set to $\rho * b$, b between [1:0.014], using the function *default_setting_mult_wz.m* and store the means. These results are then used to construct the plot and confidence interval. By using the lambdas obtained by bootstrapping, the confidence interval takes into account the uncertainty in the estimated parameters.
- Use *ciplot_trans.m* (based on ciplot.m from Raymond Reynolds, 24/11/06) to plot the mean and confidence interval and save the plot.

4.14. **negtotaldonationadd.m.**

- Simulate donations with default $\rho + a$.
- Calculates negative of the sum of donations.
- Inputs: rho, lambda1, lambda2, lambda3, a.
- Output: tot, which is the negative sum of donations.

4.15. **negtotaldonationmult.m.**

- Simulate donations with default $\rho * b$.
- Calculates negative of the sum of donations.
- Inputs: rho, lambda1, lambda2, lambda3, b.
- Output: tot, which is the negative sum of donations.

4.16. **default_setting_add_wz.m.**

- Function that simulates donations under default setting policy = rho+j-1.
- Calculates the mean and total of simulated donations under each rho+a policy.
- Inputs:
  - rho: simulated nonzero rho's from donations distribution under AD treatment.
  - alpha: simulated alpha costs.
  - delta: simulated delta costs.
  - addz: zeros added to represent ungenerous types.
  - z: indicator for agents that faces deviation costs.
  - n: sample size.
  - j: set default = rho+j-1.
- Output: mean and total.

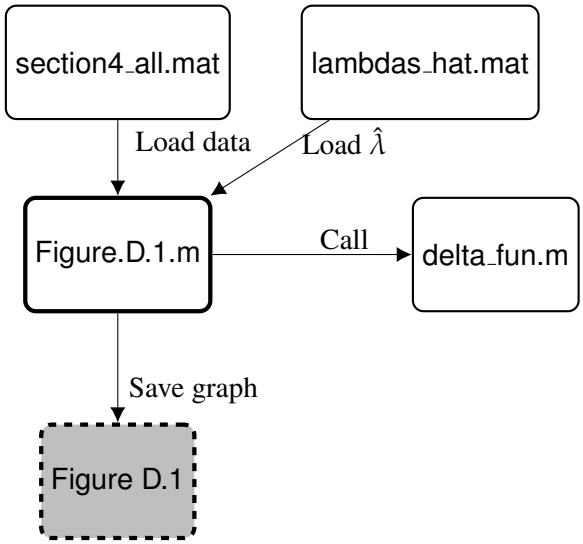
4.17. **default_setting_mult_wz.m.**

- Function that simulates donations under default setting policy = rho*j.
- Calculates the mean and total of simulated donations under each rho*b policy.
- Inputs:
  - rho: simulated nonzero rho's from donations distribution under AD treatment.
  - alpha: simulated alpha costs.
  - delta: simulated delta costs.
  - addz: zeros added to represent ungenerous types.
  - z: indicator for agents that face deviation costs.
  - n: sample size.
  - j: set default = rho*j.
- Output: mean and total.

4.18. **ciplot_trans.m.**

- Function to plot with confidence intervals.
- Confidence interval transparent.
- Based on function *ciplot.m* written by Raymond Reynolds, 24/11/06.

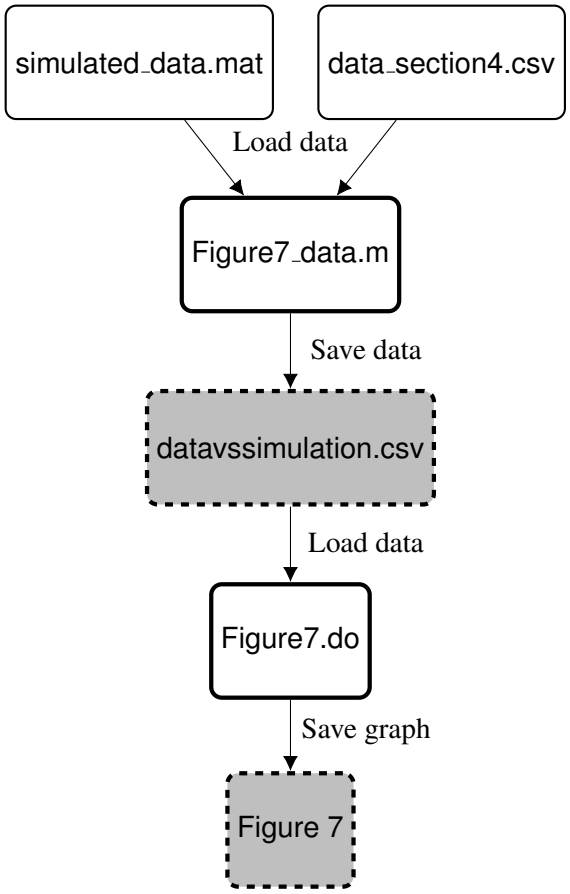
E Figure D.1



4.19. **FigureD.1.m.**

- Load *section4_all.mat*.
- Load *lambdas_hat.mat* for the estimated value of lambda2.
- Use *Delta_fun.m* to calculate delta (gain from opting out).
- Plot Delta for different defaults (10, 20 and 50) and the median of the delta costs.
- Save graph as *Delta.pdf*

F. Figure 7 Fitted versus Actual Donation Distribution



4.20. **Figure7_data.m.**

- Save actual data and simulated data in one csv file, *datavssimulation.csv*, to make plot graphs with STATA.

4.21. **Figure7.do.**

- Import datavssimulation.csv to STATA.
- Plots histograms of simulated data versus actual data of relative frequency, calculated on the interval of 0 to 300, plotted on interval 0 to 100.
- Save Figure7.pdf.

APPENDIX A. DATA DICTIONARY

Variable Name	**Description**
session_id	Website session ID
default_donation	Default donation amount assigned to the user
default_codonation	Default codonation level (in percent) assigned to the user
treat	Default treatment assignment (labeled "*(D,C)*" where D = default donation level and C = default donation level assigned to the user)
receiver_type	"Type" of donation selected by the user (project, project element, or fundraising event)
tax_deductible	Dummy = 1 if contributions to the chosen charity are tax deductible
donate	Dummy = 1 if user made a donation
codonate	Dummy = 1 if user made a codonation
donated_amount	Amount donated in EUR
codonated_amount	Amount codonated in EUR
codonated_pct	Codonation in percent of (main) donation
don_followdefault	Dummy = 1 if *donated_amount* = *default_donation*
codon_followdefault	Dummy = 1 if *codonated_pct* = *default_codonation*
both_follow	Dummy = 1 if *codonated_pct* = *default_codonation* & *donated_amount* = *default_donation*
num_donations	Number of donations within a given session
donation_sum	Total amount donated within a given session (sum of *donated_amount* per *session_id*)
codonation_sum	Total amount codonated within a given session (sum of *codonated_amount* per *session_id*)
q99_8	Dummy = 1 if *donation_sum* belongs to the top 0.2% of *donation_ sum*
D0, D10, D20, D50	Dummies for treatment assignment in donation dimension (e.g. D50=1 if *default_donation*=50)
C5, C10, C15	Dummies for treatment assignment in codonation dimension (e.g. C10=1 if *default_codonation*=10)
T05, T105, T205, T505, T010, T1010, T2010, T5010, T015, T1015, T2015, T5015	Treatment Dummies for the 12 individual treatment cells (e.g. T105 = 1 if *default_donation*=10 and *default_codonation*=5)