Web-Based Supporting Materials for "RNN-based counterfactual prediction" by Jason Poulos

Table of contents

1 Implementation details **1**

2 Hypothesis testing **2**

2.1 Randomization confidence intervals 2

3 Supporting Figures **3**

1 Implementation details

The networks are implemented with the `Keras` neural network library (Chollet, 2015) in Python on top of a TensorFlow backend. When implementing encoder-decoder networks, the encoder takes the form of a two-layer Long Short-Term Memory (LSTM) network (Schmidhuber and Hochreiter, 1997), each with 128 hidden units, and the decoder is a single-layer Gated Recurrent Unit (GRU) (Chung et al., 2014) also with 128 hidden units. Each recurrent layer uses a linear activation function (f_1) with weights initialized using Xavier initialization (Glorot and Bengio, 2010). The loss function internally computes the predicted outputs as a linear function (f_2) of the log probabilities.

RNN weights are learned with mini-batch gradient descent on the WMSE using `Adam` stochastic optimization with the learning rate set to $5 \cdot 10^{-4}$ (Kingma and Ba, 2014). As a regularization strategy, I apply dropout to the inputs and L2 regularization losses to the network weights. The networks are trained for 1,000 epochs, which takes 10 minutes to run on a laptop CPU. The model is validated on the last 20% of the training set input-out pairs.

The RVAE is implemented similarly, but with the following differences: the encoder takes the form of a single-layer LSTM with 32 hidden units and the decoder is a two-layer LSTM with the number of hidden units equal to 32 and the number of predictors, respectively. The latent space $\boldsymbol{z}$ is implemented as a densely-connected layer with a dimension of 200 units and $f_3(\cdot)$ takes the form of a log-normal distribution. The RVAE is trained with stochastic gradient descent for 5,000 epochs, which takes seven minutes to run on the same CPU.

2 Hypothesis testing

Abadie et al. (2010) propose a randomization inference approach for calculating the exact distribution of placebo effects under the sharp null hypothesis of no effect. Cavallo et al. (2013) extends the placebo-based testing approach to the case of multiple (placebo) treated units by constructing a distribution of *average* placebo effects under the null hypothesis. Firpo and Possebom (2018) derive the conditions under which the randomization inference approach is valid from a finite sample perspective and Hahn and Shi (2017) analyze the approach from a repeated sampling perspective.

Randomization p-values are obtained following these steps:

1. Estimate the observed test static $\hat{\boldsymbol{\phi}}$ from (3). Averaging over the time dimension results in a $\mathrm{T}_\star$-length array of observed average treatment effects.

2. Calculate every possible average placebo treated effect μ by randomly sampling without replacement which $\mathrm{J}-1$ control units are assumed to be treated. There are $\mathcal{Q} = \sum_{\mathrm{g}=1}^{\mathrm{J}-1} \binom{\mathrm{J}}{\mathrm{g}}$ possible average placebo effects. Since calculating $\mathcal{Q}$ can be computationally burdensome for relatively high values of J, I artificially set $\mathcal{Q} = 10,000$ in cases when $\mathrm{J} > 16$. The result is a matrix of dimension $\mathcal{Q} \times \mathrm{T}_\star$

3. Sum over the time dimension the number of μ that are greater than or equal to $\hat{\boldsymbol{\phi}}$.

Each element of the vector obtained from Step 3 is divided by $\mathcal{Q}$ to estimate a $\mathrm{T}_\star$-length vector of exact two-sided p values, $\hat{p}$.

2.1 Randomization confidence intervals

Under the assumption that treatment has a constant additive effect Δ, I construct an interval estimate for Δ by inverting the randomization test. Let δ_Δ be the test statistic calculated by subtracting all possible μ by Δ. I derive a two-sided randomization confidence interval by collecting all values of δ_Δ that yield $\hat{p}$ values greater than or equal to significance level $\alpha = 0.05$. I find the endpoints of the confidence interval by randomly sampling 500 values of Δ.

3 Supporting Figures

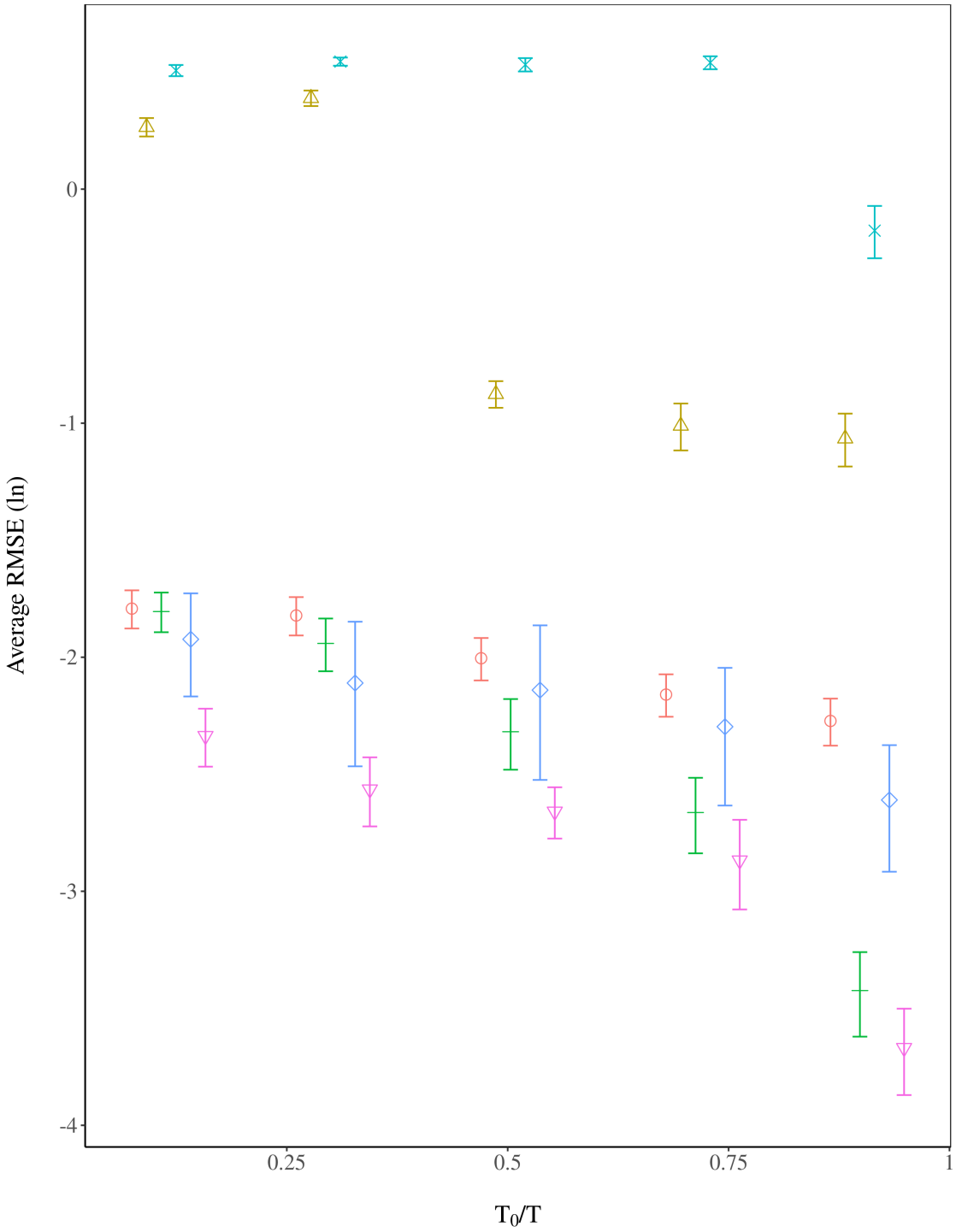


Figure 1: Placebo tests on Basque Country terrorism data: ⊖, DID; △, ED; +, MC-NNM; ×, RVAE; ◇, SCM; ▽, VT-EN.

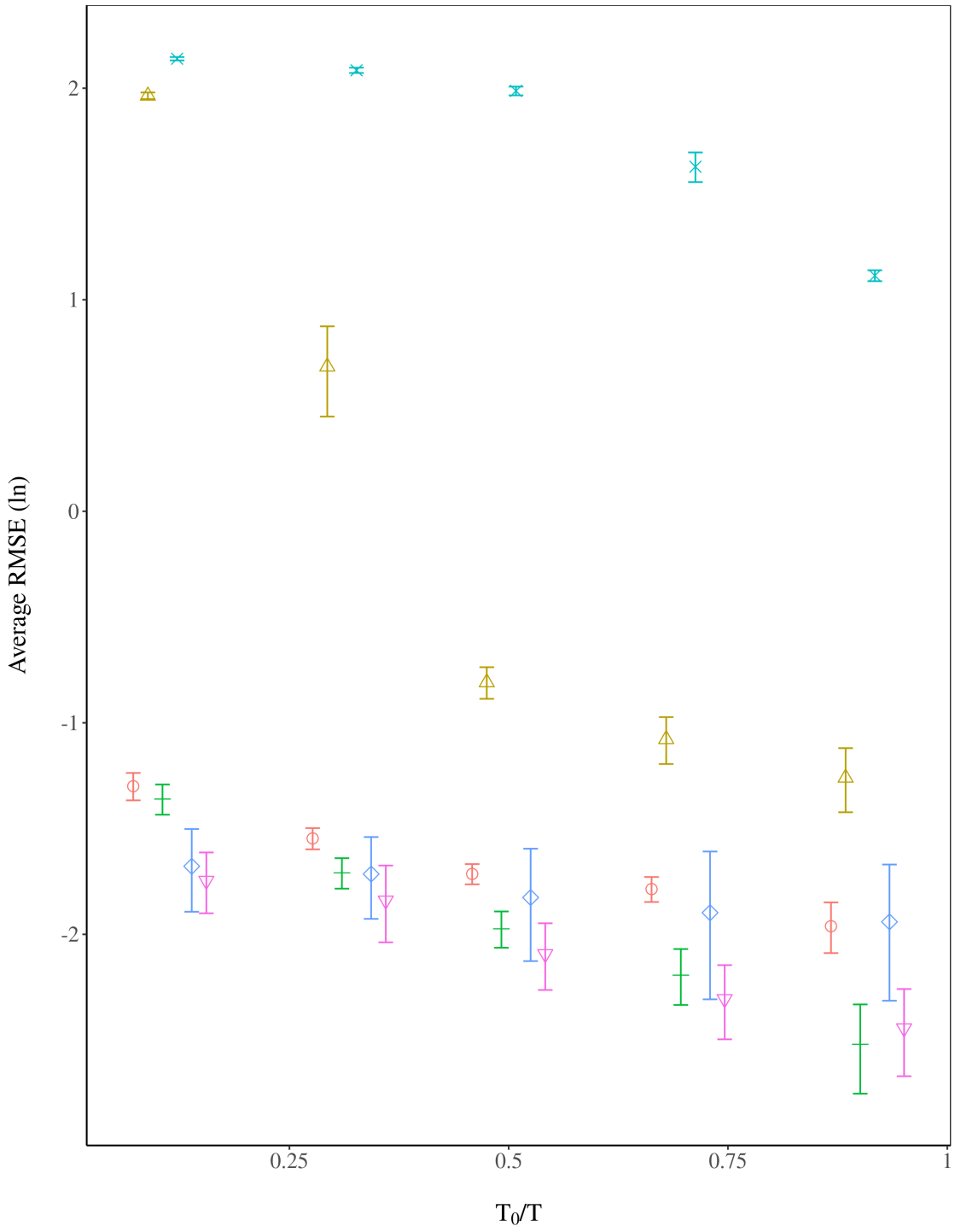
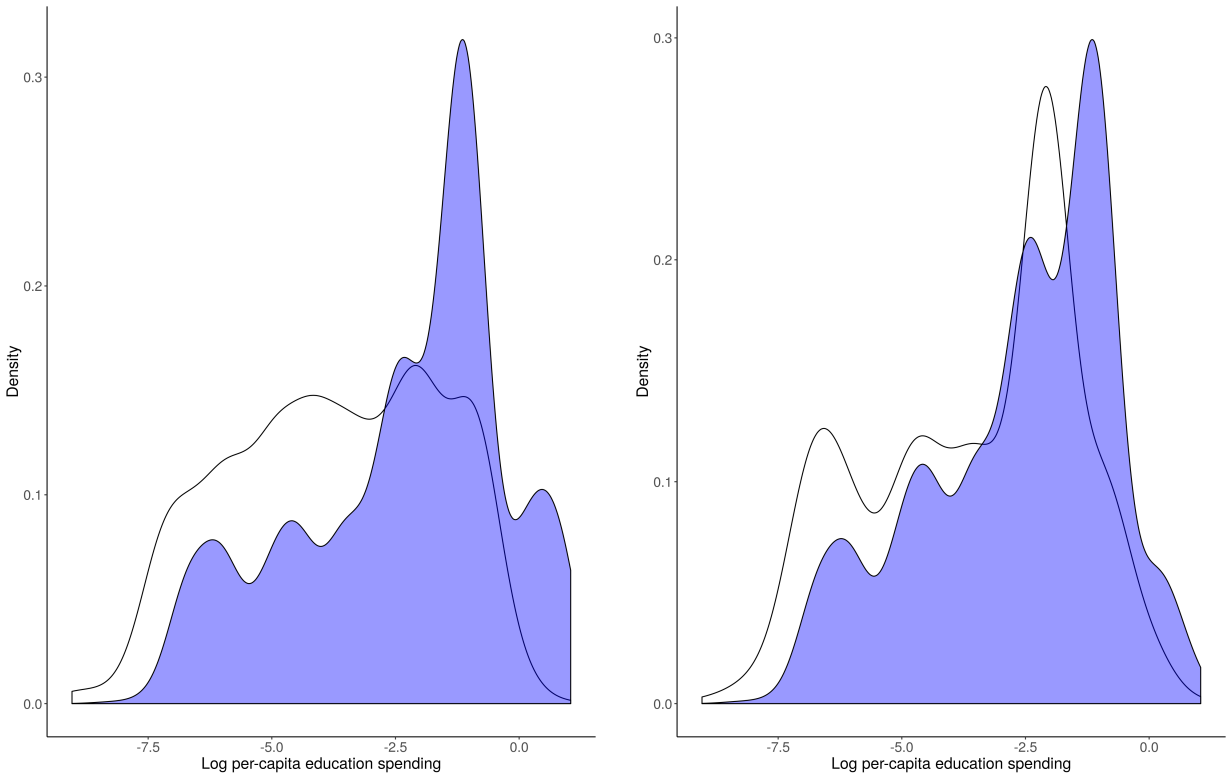


Figure 2: Placebo tests on West German reunification data: ⊖, DID; △, ED; +, MC-NNM; ×, RVAE; ◇, SCM; ▽, VT-EN.



(A) Unweighted (B) Weighted by propensity score

Figure 3: Pre-period densities of log per-capita state government education spending by treatment status: ⊟, Control; ■, Treated

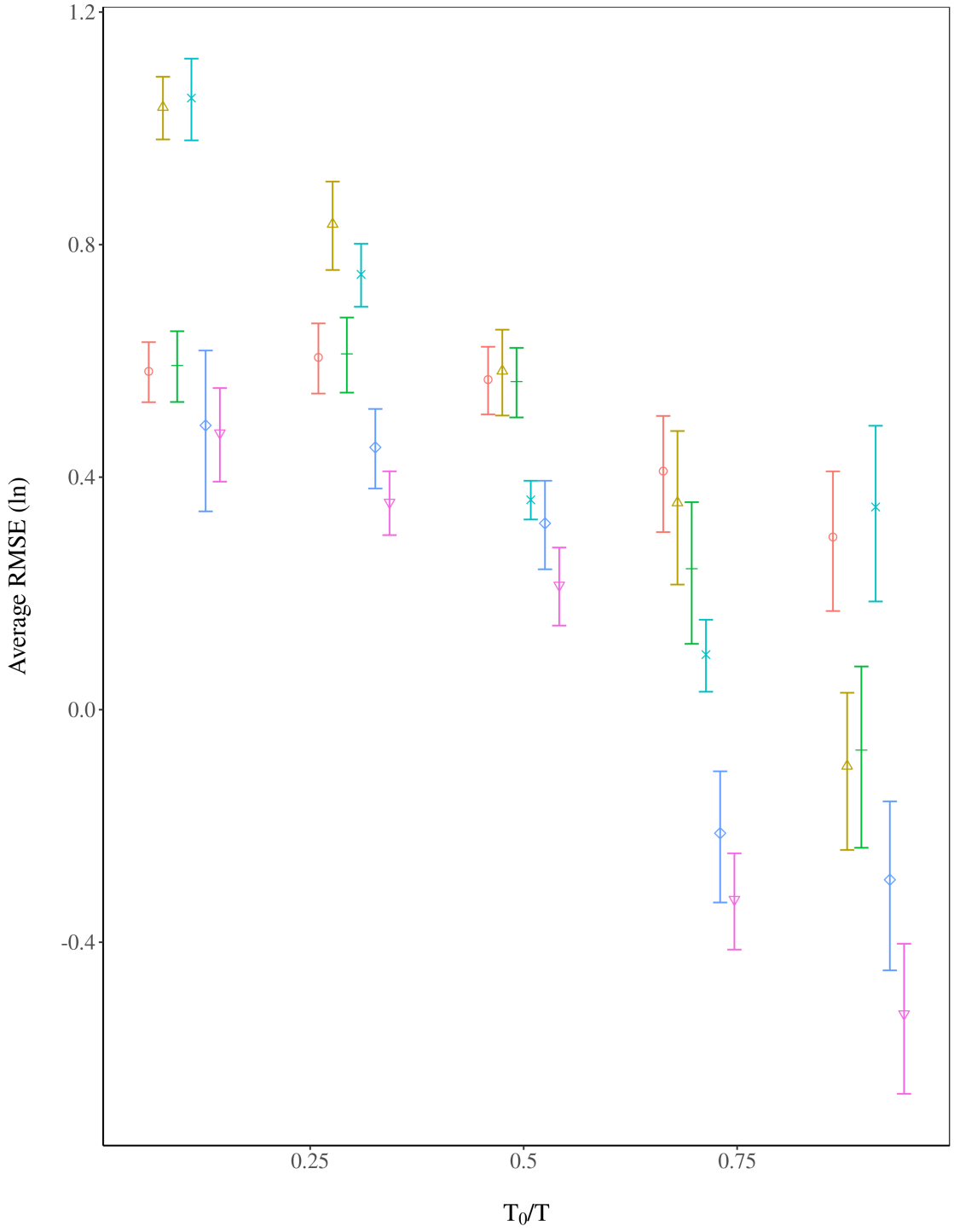


Figure 4: Placebo tests on education spending data: ⊖, DID; △, ED; +, MC-NNM; ×, RVAE; ◇, SCM; ▽, VT-EN.

References

Abadie, A., Diamond, A. and Hainmueller, J. (2010) Synthetic control methods for comparative case studies: Estimating the effect of California's tobacco control program. *Journal of the American Statistical Association*, **105**, 493–505.

Cavallo, E., Galiani, S., Noy, I. and Pantano, J. (2013) Catastrophic natural disasters and economic growth. *Review of Economics and Statistics*, **95**, 1549–1561.

Chollet, F. (2015) Keras. Available at: `https://github.com/fchollet/keras`.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2014) Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv e-prints*, arXiv:1412.3555.

Firpo, S. and Possebom, V. (2018) Synthetic control method: Inference, sensitivity analysis and confidence sets. *Journal of Causal Inference*, **6**.

Glorot, X. and Bengio, Y. (2010) Understanding the difficulty of training deep feedforward neural networks. In *Artificial Intelligence and Statistics*, vol. 9, 249–256.

Hahn, J. and Shi, R. (2017) Synthetic control and inference. *Econometrics*, **5**, 52.

Kingma, D. P. and Ba, J. (2014) Adam: A Method for Stochastic Optimization. *arXiv e-prints*, arXiv:1412.6980.

Schmidhuber, J. and Hochreiter, S. (1997) Long short-term memory. *Neural Computation*, **9**, 1735–1780.