ABSTRACT

KIM, AHNJEONG. The Effect of Transmission Capacity Expansion on Fossil Fuel Generation and Wholesale Electricity Market Efficiency, and Statacons: An SCons-Based Build Tool for Stata.. (Under the direction of Harrison Fell).

As the share of wind energy has increased significantly in recent years to reduce carbon emissions, electricity operators have been planning and constructing more transmission projects. Since wind turbines are geographically limited, increasing transmission capacity is necessary to deliver wind energy from farms to high-demand regions.

This paper evaluates whether transmission expansion can effectively increase wind energy usage and reduce fossil fuel generation in the Midwest. Based on the two-region model of Joskow and Tirole (2005), I show how increased transmission capacity reduces the average cost of fossil fuel generation and empirically estimate the average treatment effect of transmission expansion on fossil generation costs. The results show average costs decreasing from $1.502 to $0.248 per megawatt-hour to the projects, with larger impacts during on-peak hours, in the Central region, and for higher marginal cost fuels like oil and natural gas combined cycle. However, total cost and generation impacts differ across regions and fuel types.

Current competitive wholesale electricity markets exhibit inefficiencies due to financial barriers and physical limitations, so this study also examines how large transmission expansion projects contribute to increased efficiency in the Midcontinent Independent System Operator(MISO) wholesale electricity market. The main findings suggest that as transmission line capacity increases, congestion and electricity price differences across nodes are reduced, decreasing virtual bidding and Financial Transmission Rights(FTR) auction prices. Although forward market premiums increase, the frequency of extreme values decreases. Overall, the findings indicate transmission capacity expansion provides benefits for competitive electricity market efficiency.

The final chapter presents `statacons`, an SCons-based build tool for Stata. Because of the integration of Stata and Python in recent versions of Stata, we are able to adapt SCons for Stata

workflows without the use of an external shell or extensive configuration. We discuss the usefulness of build tools generally, provide examples of the use of `statacons` in Stata workflows, present key elements of the syntax of `statacons`, and discuss extensions, alternatives, and limitations.

The Effect of Transmission Capacity Expansion on Fossil Fuel Generation and Wholesale Electricity Market Efficiency, and Statacons: An SCons-Based Build Tool for Stata.

by
Ahnjeong Kim

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Economics

Raleigh, North Carolina
2024

APPROVED BY:

Raymond Guiteras	Zheng Li
Zachary Brown	Harrison Fell Chair of Advisory Committee

BIOGRAPHY

Ahnjeong Kim was born in Busan and grew up in Seoul, South Korea. She decided to study economics and entered the Department of Social Sciences at Ewha Womans University in Seoul. Ahnjeong pursued an interdisciplinary program, earning a bachelor's degree in economics with a minor in statistics, as well as a master's degree in economics over five years. While studying abroad for a semester at Martin Luther University Wittenberg in Germany, she desired to pursue graduate studies overseas. After obtaining her master's degree in South Korea, Ahnjeong worked full-time as an economics researcher at government research institutions in South Korea for four years. She then moved to Raleigh, North Carolina, to start her doctoral program at North Carolina State University. After completing her dissertation, Ahnjeong desires to continue researching economics and working for the public and her community.

ACKNOWLEDGEMENTS

I express my deepest gratitude to my dissertation chair, Dr. Harrison Fell, for his academic and emotional support throughout this long journey. I sincerely thank him for providing me the opportunity to embark on research in this field. His invaluable guidance on my dissertation, as well as advice and encouragement from my first doctoral course to my final defense, made this achievement possible.

I also wish to thank my exceptional committee members: Dr. Raymond for welcoming me into statacons project and offering invaluable writing guidance; Dr. Li for providing technical insights; and Dr. Brown for your considerable support over the years. Additionally, I thank all the graduate professors at NCSU who helped molded me into a researcher.

Completing this program would have been impossible without my family's love – thank you to my mom, dad, sister Soyeon, brother Jeonghyun, and my beloved dog Ari.

I also extend my gratitude to professional mentors like Dr. Pyo for recommending NCSU and this path, as well as my Masters advisors Dr. Lee and Dr. Song for instilling research skills that prepared me for doctoral success.

To my cherished friends, named and unnamed, from Korea and in Raleigh who walked alongside me – sharing my struggles and celebrating my milestones – I could never have persevered without your encouragement. Specifically, I thank my caring roommate Dasom and friends like Misong, Eunji, Sol, Hana, Dr.Cho, Anjela, Anjelo, Erin, Ewan, Teresa, MK, Yiqing, Shu, Yoonju, Jieun, Ghipbumm, Hyemin, Minjin, Dave, Keyna, Hojun, Jihun, Jieun, another Jieun, Jihyun, Minjae, Hyunji, Sangmi, Eun-a, Sunjae, Inyoung, Jeongeun, and Heewon.

Most importantly, I thank God for these blessings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In energy markets, the importance of transmission lines is growing due to shifts in electricity dynamics: increases in intermittent renewable generation require larger grids for reliable electricity supply, while intensifying climate change volatility necessitates meeting spikes in electricity demand, also calling for expanded transmission infrastructure. In the United States, interstate transmission projects have been planned and constructed since 2010. However, economics research on transmission lines remains limited.

In this dissertation, I discuss how energy markets, especially power generation and the wholesale electricity market, respond as new transmission capacity comes online. The Midwest offers an ideal study area given the recent completion of lengthy expansion efforts. The next two chapters discuss and estimate transmission impacts from an economic point of view.

In the second chapter, "The Effect of Transmission Expansion on Renewable Energy and Fossil Fuel Generation: The Case of MISO," demonstrates that expanded transmission lines, especially interconnections of renewable generation facilities like Midwest wind farms, can reduce fossil generation expenses by displacing higher-marginal cost generation units, such as traditional natural gas turbines, according to economic dispatch priorities. Empirical results show that the effect is more significant in the central region and during peak hours.

In the third chapter, "The Impact of Transmission Capacity Expansion on the Wholesale

Electricity Market," discusses the effect of transmission expansion on the electricity wholesale market. The deregulated electricity market structure is designed complexly to improve its cost efficiency, so the existence of virtual bidders and financial transmission rights (FTRs) makes it difficult to predict the effect of new transmission lines on electricity prices and the market. This study introduces the wholesale electricity market structures of price, convergence bidding, and FTR auctions, then provides empirical evidence that transmission lines reduce congestion and can improve overall market efficiency.

The final chapter presents statacons, a new Stata build tool to aid economists and social scientists working with large, multi-stage dataset projects. Build tools are developed for developers, but Python's accessibility within Stata allows analogous functionality through SCons. This collaboration yielded statacons, enabling automation benefits like reproducibility to be brought to researchers. This chapter introduces the role of build tool and related concepts about target and dependency, and provides simple examples of statacons and its additional capabilities.

Chapter 2

The Effect of Transmission Expansion on Renewable Energy and Fossil Fuel Generation: The Case of MISO

2.1 Introduction

The fossil fuel generation has decreased and has been replaced with renewable energy in recent decades, but fossil fuel still accounts for the largest share of U.S. electricity generation. One of the limitations preventing the increase in the share of renewable energy is its high geographical dependence. In general, rich renewable energy areas are geographically far from locations with high electricity demand. Wind generation, for example, requires at least 13 miles per hour of annual average wind speed for utility-scale turbines and open plains with higher elevation (500-900 feets) than earth surface (EIA 2022). In the U.S., the great plain is suitable for wind generation, but the problem is that it is too far from the region with high-demand of energy such as metropolitan areas. To overcome this limitation, it is necessary to increase the transmission capacity. The transmission line from regions rich in renewable energy sources to regions with high power demand can make it possible to weaken the transmission constraint

and redistribute renewable energy to a wider area.

Many power system operators and electricity utilities have expanded the transmission grid. The investment of electric transmission systems has increased from $9.1 billion (2019 dollars) in 2000 to $40.0 billion in 2019. (EIA 2021) Major interstate transmission projects such as the Gateway Project have been completed or are in progress across the U.S. by Independent System Operators or investor-owned utilities. In the Midwest area, Midwest Independent System Operator (MISO) has been constructing a Multi-Value Project portfolio that consists of 17 transmission projects to provide wind energy from the Great Plains to electricity high-demand region such as Michigan and and Wisconsin and to meet the Renewable Portfolio Standards of each state.

This paper studies the benefit of transmission line expansion especially on fossil fuel generation cost in the Midcontinent Independent System Operator(MISO) area. For this, I develop the previous two-region model from Joskow and Tirole (2005) by including the supply of wind energy. The difference from LaRiviere and Lyu (2022) is that they consider the long run supply curve but I assume a short-run supply curve of wind energy and reasonable marginal price of wind energy. I also combine the two-region model to merit order supply curve and consider the mix of energy supply, the idea comes from Yang (2022). I adopt the situation of the Midwest, and provide the potential benefits of transmission supply under several scenarios of wind generation and transmission capacity, and how it actually reduces the fossil fuel generation from the increase of usage of renewable energy.

In an empirical approach, I estimate the effect of new transmission lines on the average cost of fossil fuel generation by each transmission project. I use the regression discontinuity in time framework reviewed by Hausman and Rapson (2018) instead of the standard regression discontinuity design or the difference-in-difference framework because of the lack of cross-sectional counterfactual in transmission projects but the availability of sufficient high-frequency data sets. As the time is the running variable in the RD in time framework and the starting date of in-service of each new transmission line is the cut-off, I estimate the average treatment effect

of the average cost of fossil fuel generation of 15 transmission projects in MISO north and central region.

The main finding of this study is that the average costs of fossil generation decrease from 0.29 dollars per Megawatts hour to 0.8 dollars per Megawatts hour, which depends on each project. The results are slightly differ from regions and fuel types. The average cost of coal generation consistently decreases, but the average costs of natural gas generation are controversial and not clear because of the high-dependency of its market price. The effects are also different from regions.

This study contributes to the literature on the effect of transmission expansion in the context of a two-node model of transmission capacity. Joskow and Tirole (2005) introduced a simple transmission system with two nodes with congestion and price discrepancy between nodes under insufficient transmission capacity. Borenstein et al. (2000) also developed the model of two geographically distinct electricity markets with transmission. Both models demonstrated that increased transmission capacity would lower the market power in local electricity producers and increase the competitiveness between the distinct regions. This results in a price reduction, less congestion, and the increase in social benefit under a deregulated market system. These theoretical models have been developed as adopting the features of renewable generation. LaRiviere and Lyu (2022) continued the two-region model that includes the intermittent wind supply to explain the transmission project in Texas. Yang (2022) combined the two-region model with the energy mix model from Ambec and Crampes (2012) to evaluate the effect of carbon tax in renewable generation.

This study contributes also to empirical studies of the benefit of transmission expansion on other deregulated electricity markets such as California, Texas, and Alberta in Canada. Davis and Hausman (2016) found that the decrease in transmission capacity due to the sudden closure of a nuclear power plant causes the increase in generation cost in California. In a deregulated electricity market, the transmission expansion also affects by lowering the electricity market clearing prices in the wholesale power market. Using the geographically isolated characteristics

of Alberta peninsula, in Canada. Doucet et al. (2013) developed the method of estimating the value of transmission expansion by determining the transaction cost associated with the electricity flows and measuring the value of new transmission projects. Wolak (2015) demonstrated that the competitiveness benefits of a transmission expansion leads to lower the market clearing price in the Alberta wholesale electricity market. In the context of development economics, Ryan (2021) estimated the effect of transmission expansion on the wholesale power market price and the reliability of electricity supply in India by simulating counterfactual markets.

The role of unconstrained transmission capacity becomes more important as the share of renewable energy increases because of the intermittency and the locational limitation of renewable energy, particularly, wind energy. To deliver more wind energy efficiently, a large transmission expansion was completed in 2013 in Texas, also known as CREZ project and many researches evaluate the market and non-market value of this project. Fell et al. (2021) examine the environmental and social benefits of the CREZ project, and LaRiviere and Lyu (2022) examine the benefit on the increase in renewable energy capacity and its market value.

This paper is organized as follows. Section 2.2 provides the basic knowledge about the electricity market and power generation structure in MISO. Section 2.3 illustrates the two-region model and provides theoretical evidence of the benefits of transmission expansion. Section 2.4 describes the fossil fuel generation and electricity price data set. Section 2.5 and section 2.6 present the empirical strategy and its results. Section 2.7 discusses the findings and concludes.

2.2 Background

2.2.1 The structure of wholesale electricity market

General wholesale electricity markets consist of four stages: generation, transmission, distribution, and consumption.(Kiyak and de Vries (2017)) At generation stage, power plant units

generate electricity from fossil fuel, nuclear, hydro, and renewable energy. At the transmission stage, Transmission system operators (TSO) transmit electricity from power plants to regional electricity distribution operators. TSO coordinates the dispatch of generation to meet the electricity demand across the transmission grid. At the distribution stage, distribution operators(utilities) deliver electricity to the consumers: households and firms.

In the wholesale electricity market, generators sell electricity to the re-seller. Deregulated wholesale electricity markets are operated by an independent system operator(ISO) or Regional Transmission organization (RTO) in the U.S. RTO or ISO generally uses energy markets to decide which power generation units to dispatch[1], or run, and its order. Since the electricity cannot be stored, it is important to dispatch to meet the supply and demand of electricity every time.

The dispatch process occurs in two stages: day-ahead unit commitment in the day-ahead market, and dispatching the system in the real-time market. Since the electricity cannot be stored, it is important to dispatch to meet the supply and demand of electricity every time.

The day-ahead market occurs a day before the operation day in order to allow generators time to prepare for an operation. About 95 percent of energy is traded in the DA market, based on the forecast load for the next day.

RTOs decide which generating units should be committed to being online for each hour, based on the offers from electric suppliers of their power plants generation. To select the most economical generators to commit, operators consider the forecast load for the next day and each generating unit's physical operating characteristics, such as how quickly output can be changed, maximum and minimum output levels and the minimum time a generator must run once it is started. Operators also take into account the cost of each generating unit, such as fuel and non-fuel operating costs and the cost of environmental compliance.

Operators also consider the reliability of electricity. When power cannot flow freely because of transmission constraints[2], that is not possible, RTOs account for congestion and its cost

[1]Dispatch is the change of resource output to balance supply and demand.

[2]Transmission constraint is the situation when the element of the transmission system is limited by how much power it can transfer. When a transmission element reaches a limit, the system becomes congested(constraint is binding).

on transmission lines by allowing prices to differ by location. The reliability of the grid can be affected by forecast load, weather conditions, transmission capacity level, flow direction, and, sometimes, generation and transmission facility outages. If power cannot be delivered reliably, relatively expensive generators may have to be called to replace less-expensive units, which leads to areas with high demand and scarce electric resources having higher prices than those with abundant generation relative to the load.

Operators compile the list of available generators for next-day dispatch and order them from least expensive to most expensive to operate. It allows the market to meet energy demand at the lowest possible price. During periods of high demand, wholesale prices rise accordingly because more high-cost units need to be dispatched to meet the electric load. This is done in advance because some generating units require several hours of starting up before actual power generation. However, actual dispatching can vary from the day-ahead commitment and real-time adjusted dispatch. Day-Ahead markets help reduce the volatility of electricity prices.

The Real-time market works on the operating day, and market participants sell or buy the electricity to balance the differences between real-time actual demand and day-ahead commitments. The remaining energy is traded in the real-time market, which is run once every hour and once every five minutes to balance the real-time load changes and supply. As part of real-time operations, demand, generation, and interchange (imports and exports) must be continually kept in balance to maintain a system frequency of 60 hertz. Real-time markets decide dispatch generally 5 minutes intervals. The transmission operator should transfer electricity from the power plant to the distributor in real time.

The transmission operator should transfer electricity from the power plant to the distributor in real-time. If the transmission grid is a constraint (Limited capacity), it should affect the price of electricity and reliability on demand in peak hours.

2.2.2 Merit order effect

In the electricity market, the electricity demand is inelastic and the supply curve follows the merit order. The merit order means that market operators (RTO or ISO) dispatch plants from the lowest-cost plant. However, various generation technologies have differing variable costs. In a deregulated market, available generators offer their cost to the operator.

The marginal cost of electricity generation differs from fuel types. Due to the zero marginal cost, renewable energy, wind, and solar, are placed at the bottom of the supply curve. In general, conventional fossil fuel power generation (coal-fired, natural gas, petroleum, etc.) has a high marginal cost that depends highly on the fluctuation of fuel price. Nuclear or biomass power has a very low marginal cost. Renewable energy has zero marginal cost.

Coal plants produce approximately one third of electricity in the U.S. Coal plants are used as base load units because they run continuously and are not flexible to control power output. Coal plants require relatively high initial capital cost, but have low marginal costs. Oil fired plants produce only a small amount of the total electricity generation in the U.S. because of its expansive marginal cost and high level of emission. Nuclear plants have very high capital investment costs but low variable costs. Nuclear plants run 18-24 months at once.

Natural gas power plants are divided into three major technologies: steam boiler, combustion turbines, and combined cycle generations. Steam boilers are the oldest technology. It has a long start-up time to become operational and is limited in its flexibility to produce the power output. A combustion turbine is a quick-start unit and has a short operational life. It is relatively inexpensive to build, but expensive to operate because of lower power output for the amount of gas burned and high maintenance costs. Gas turbine generation is used to serve at the highest demand hour during peak periods. A combined cycle power plant is a hybrid of a gas turbine and steam boiler. It has high efficiency and a significantly low emission level. In this paper, I divide the natural gas generation by Combined Cycle (NG CC) and combustion turbine (NG CT) that includes a steamed boiler.

Wind capacity has grown from approximately 11 GW to 82 GW between 2006 and 2016.[3] However, since wind generation depends on weather conditions, the ratio of average generation has been lower than conventional generations. The cost of wind generation is declining due to technology improvements and federal tax credits. However, wind generation is often inversely correlated to demand (seasonally and daily), and it is difficult to forecast its generation accurately. Thus, companion generation like fossil-fuel generation is required to balance wind generation when the wind is not blowing.

2.2.3 Institutional background

Midcontinent Independent System Operator (MISO) is the second largest deregulated wholesale electricity market in the U.S. MISO provides rich data about electricity market prices, node information, generation publicly, and the demand for electricity at the hour level. The midcontinent region still highly depends on their electricity supply on thermal fuel, particularly coal-fired plants.

MISO has gradually increased the wind energy capacity to replace thermal power generation for decades. In the MISO report in 2020, the maximum wind capacity is 20,452 MW, and all of the capacities are concentrated in North and Central (and no wind capacity in South in 2020.) particularly Iowa(10,029MW), Minnesota, and Montana(6,188MW), and Lower Michigan(2,166MW). Most Wind farms are located far from high-demand regions so the transmission line expansion is necessary to transmit the wind power to high-demand regions. Thus, MISO has expanded its transmission grid since 2008 to connect with wind farms and metropolitan areas.

According to U.S. Energy Information Administration reports in 2021, Iowa, Kansas, and Illinois are among the top five wind energy generation states in the United States. (Figure 2.2 panel (a)) The share of wind generation in Iowa is more than 50%, which is higher than even Texas in 2020.

[3] Form EIA-860 (November 9, 2017)

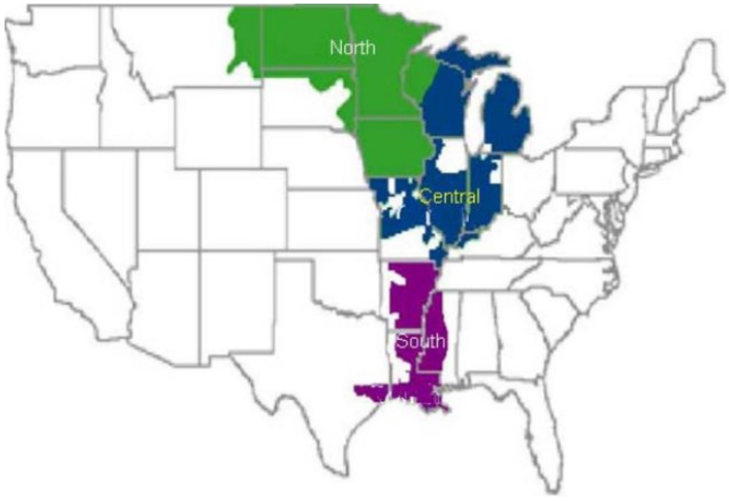
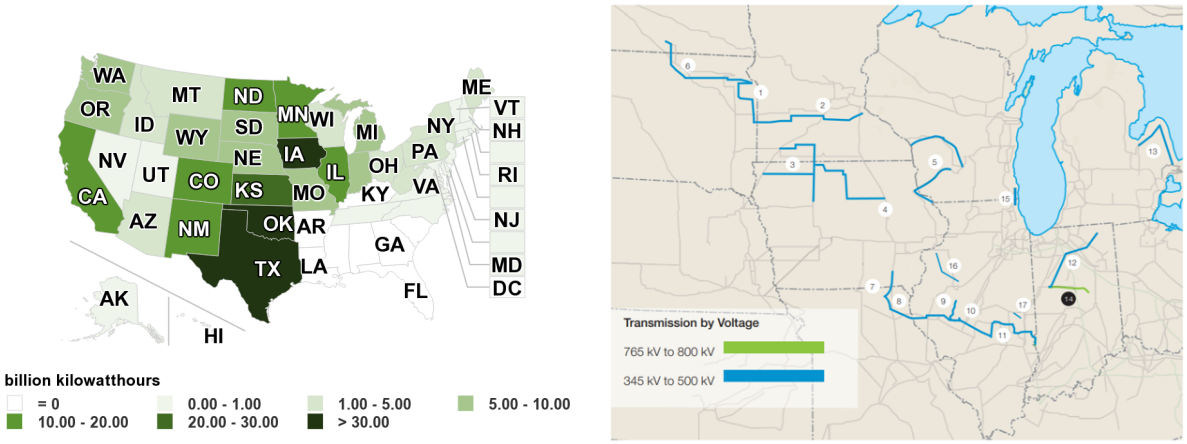


Figure 2.1: MISO Real Time Operation Region

Source: MISO MTEP report



(a) U.S. Wind Generation by States (b) MVP Project Map

Figure 2.2: Wind Farm and MVP Map in Midwest

Source: U.S. Energy Information Administration, MISO

In order to transmit wind power from wind farms to electricity high-demand areas, a number of transmission expansion projects have been completed in the Midwest. The major transmission projects, also known as Multi-Value Projects (MVP), consist of 17 high-voltage(over 345kV) transmission line construction across MISO territory. (See MVP map in Figure 2.2 panel (b).) Most of the projects are now completed but a few lines still remain in a pending status. MISO

expects that all MVPs will be completed in 2024. [4][5]

The Multi-Value Projects(MVP) are the large transmission projects conducted by MISO from 2011 to 2020. MVP consists of 17 projects and is planned to be built in MISO north and central regions because it was planned before combining the south region into MISO territory. Its estimated cost is $5.6 billion.

The main purpose of MVP is to meet the Renewable Portfolio Standard (RPS) of twelve states in MISO territory. To meet RPS, the MISO needs a more regional and reliable transmission system to deliver renewable resources from a remote wind farm to load centers. Before the approval of the MVP portfolio in 2011, the transmission grid in MISO territory(north and central region) was not enough to connect the wind energy from wind farms in the west to the east where electricity demand is high. That is the main region for the MVP portfolio. Although MVPs are not completed until 2023, most of them are completed and not all projects are connected. It is the proper time to evaluate its effect

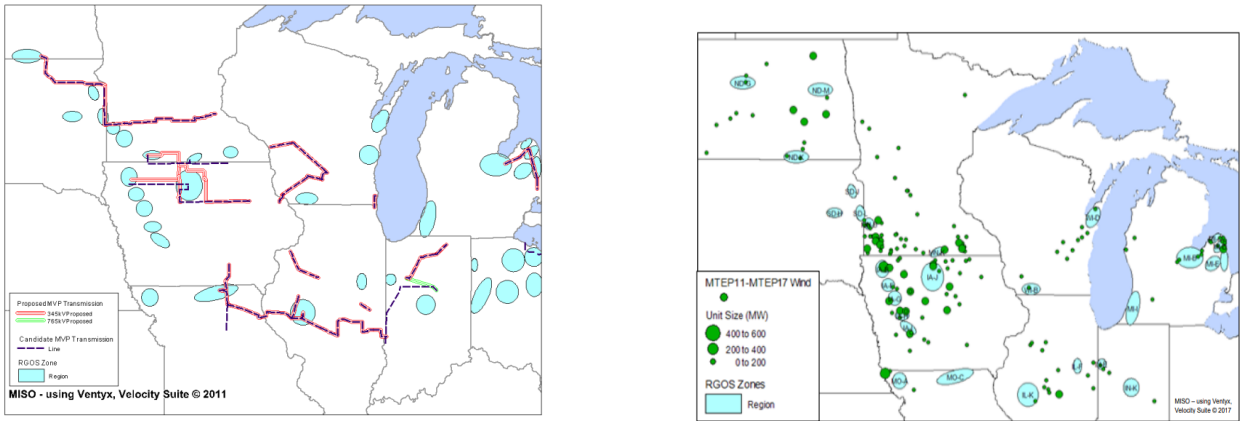
The transmission lines were designed with the consideration of wind capacity and existing infrastructure, such as transmission and natural gas pipelines. (Figure 2.3)

MISO determines wind farm locations based on a low-cost wind siting methodology when transmission and generation capital costs are considered. The place where the wind is the strongest is generally far from the place where the load is high, this strategy depends on a strong regional transmission system to deliver the wind energy. Without this transmission system, wind generation would have to be located near high-demand regions. It requires a larger amount of cost for the construction of wind capacity to produce the renewable energy mandated by public policy. The low-cost wind siting methodology enabled by MVP portfolio will create benefits ranging from a present value of $1.4 to $2.5 billion in 2011 dollars.

In addition, as the geographical distance between wind generations decreases, the correlation in wind output decreases. In other words, a geographically diverse set of wind plants has a

[4]Part of MVP 5 (Cardinal - Hickory Creek Line) is pending in 2022 and expected to complete in 2024. Project 11 has been completed in the late 2020, which is out of the sample periods.

[5]Due to its incomplete status, MISO provides only the expectation of future cost benefits of MVPs. Most recent analytic reports were published in 2017.



(a) MVP Portfolio Lines (b) Wind Turbine Installation Map

MISO North/Central regions, source: MISO MTEP report

Figure 2.3: MISO MVP Portfolio Map and Energy Source

higher output than a closely clustered group of wind plants. The MVP portfolio considers the expansion of the geographic diversity of wind resources to increase the average wind output.

Among 17 MVPs, I choose 5 projects which already have been completed and started its in-service by December 2020. Those 5 MVP projects has been connected from wind farm in MISO territory (Iowa, North Dakota, and Minnesota.) and provide large capacity of transmission after starting their services.

MISO territory is divided into three: North, Central, and Southern. North region operated out of Eagan, MN, which includes MT, ND, SD, MN, and IA. The central region operated out of Carmel, IN, which includes IN, IA, MO, KY, OH, WI and MI. South region operated out of Little Rock, AK, which includes AK, MS, AR, and northeast TX. The Southern region has joined MISO since the end of 2013. In this paper, only the central and north region of MISO territory is considered. I exclude the Southern region because it joined in December 2013 and many of the new transmission projects in MISO territory do not reach the southern regions.

Table 2.1: MVP Portfolio List

Project	Project Name	In Service Date	Max kV
15	Pleasant Prairie-Zion Energy Center 345 kV line	12/6/2013	345
2	Brookings, SD - SE Twin Cities 345 kV	3/26/2015	345
13	Michigan Thumb Wind Zone	12/31/2015	345
17	Sidney to Rising 345 kV line	9/1/2016	345
1	Big Stone South to Brookings	9/8/2017	345
10	Pawnee to Pana - 345 kV Line	10/27/2017	345
9	Maywood -Austin 345 kV Line	12/20/2017	345
16	Fargo-Oak Grove 345 kV Line (Spoon River Project)	2/21/2018	345
14	Reynolds to Greentown 765 kV line	6/25/2018	765
3	Lakefield Jct. - Webster 345 kV line	9/27/2018	345
12	Reynolds to Burr Oak to Hiple 345 kV	9/30/2018	345
6	Ellendale to Big Stone South	2/5/2019	345
7	Zachary - Ottumwa 345	7/1/2019	345
4	Winco to Hazleton 345 kV line	7/18/2019	345
8	Zachary-Maywood 345 kV Line	12/15/2019	345
11	Kansas-Sugar Creek 345 kV Line	12/14/2020	345
5*	Badger Coulee and Cardinal - Hickory Creek Line	Exp in 2024	345

Source: MISO MTEP report

2.3 Conceptual Framework

In this section, I introduce the two-region model with transmission line constrained and related literature. And, I develop and extend the model to apply to the transmission expansion effect in MISO.

2.3.1 Two-region model

? introduce a two-region model to analyze the mitigation of market power in local generators as the transmission capacity increases. This two-region model is based on Cournot duopoly equilibrium. The model assumes that a single monopolistic generator in each region has market power with identical costs, under the ISO system. If there is no connection between the two regions, each firm will produce the single-market monopoly quantity and the price of electricity will be the monopoly price. If there is an uncongested line(unlimited transmission capacity),

the price and the quantity of electricity in two regions are equal to the unconstrained Cournot duopoly equilibrium. Under binding constraints and the transmission capacity is too small, they prove that there is no pure-strategy equilibrium. As the capacity increases, the market power of generators in each region mitigate so that the price will approach the unconstrained Cournot duopoly equilibrium.

Joskow and Tirole (2000) also develop the two-node network model with different approaches in order to analyze the allocation of Financial Transmission Rights (FTR). Under ISO operation, one region (South) has the load(demand) and a monopolistic generator whose marginal cost is high. The other region (north) has no demand and runs a lower marginal cost generator. The north needs to export the electricity to the south through a single transmission line. Under perfect competition assumption, the marginal cost of generation is equal to the market-cleared price of electricity, so the market-cleared price of electricity in the north is cheaper than the price in the south. And, the consumers in the south consume the electricity generated from both the north and the south.

When the transmission capacity is binding, the north generates electricity up to the transmission capacity K, and ISO dispatches the electricity from the north first and meets the rest of the demand by generation from the south. The monopoly generator in the south chooses the monopoly price based on the residual demand (the difference between the total demand and the transmission capacity). And, the cost of transmission becomes the difference in the price. [6][7]

Joskow and Tirole (2005) develop their previous two-node network model to explain the effect of the increase in transmission capacity. The basic assumptions are the same as the model in Joskow and Tirole(2002), and the difference is the role of the transmission capacity K. The difference $\eta = p_S - p_N$ is the shadow price of the transmission capacity constraint and ηK is the congestion rent. Congestion cost is decreasing as K increases. If the transmission

[6]The model assumes the loss of electricity in transmission is zero.

[7]Hogan (1992) suggests that the price of transmission congestion is determined by the difference in nodal price.

capacity K increases sufficiently to reach the uncongested level, η will become zero and there is no price difference between the north and the south.

The common intuition among the three papers is the existence of market power of power generators in a local region caused by the transmission binding constraint. All prove that market power mitigates as transmission capacity increases, which integrates distinct regions, and the price difference among regions disappears.

However, the welfare effect of increased transmission capacity is controversial under these model settings. Under fossil fuel and conventional generation, transmission expansion is not always an attractive decision for generators or operators because of the decrease in monopoly generators' profit and welfare effect if the benefit of transmission expansion does not exceed the cost of construction.

Léautier (2001) finds that the generator's profit in the welfare function decides the welfare outcome of transmission expansion since the mitigation of market power can reduce the generator's profits. Kristiansen and Rosellón (2006) also analyze the incentives for transmission expansion based on merchant mechanisms and FTRs.

Those three models do not consider the environmental externalizes from conventional generations and the intermittency of renewable energy. Recently, the importance of renewable energy is taken into account for transmission expansion issues. Hitaj (2015) finds that transmission congestion can have a significant effect on output from Renewable Power Plants when Renewable Power Plants cluster together in more remote areas of the grid with low electricity demand which may result in greater emission reduction compared to the distributed arrangement of renewable power plants.

LaRiviere and Lyu (2022) include the intermittency of wind energy in the basic model of Joskow and Tirole (2005). They assume the supply curve when there is no wind generation and when wind generation. The cost of meeting a given level of load from fossil fuel generation will decrease. The shadow cost of a transmission constraint increases as wind generation increases since the price of electricity decreases in the wind energy exporting region. As the transmission

capacity increases, the shadow price will decrease.

Yang (2022) extends the model of Joskow and Tirole (2005) to analyze the decision to make an investment in renewable energy under Pgiouvian carbon price. She includes the intermittency of renewable energy in probability and considers how intermittency affects the price of transmission rights by adopting the optimal mix of conventional and renewable energy Ambec and Crampes (2012).

The model supposes two countries: a clean country that produces more renewable energy and a dirty country that has an absolute advantage over thermal production. Both countries have two states of nature with high and low renewable energy output. Two countries are interconnected and can import and export electricity up to a certain transmission capacity limit. She concludes that the shadow value of constrained transmission depends on the intermittent state of renewable generation.

2.3.2 Conceptual model

In this section, I combine the two-region model of Joskow and Tirole (2005) and LaRiviere and Lyu (2022) and apply it to the case of MISO wind farm and transmission expansion problem. Similar to previous literature, I assume two regions: West and East under the same ISO wholesale electricity market. The West is the wind-exporting region and the east is the wind-importing region.

For simplification, I assume The West generates only wind energy and has zero loads. The west should export all wind generation to the East. This assumption is not realistic but reasonable because the load in the location of a wind farm is generally very low. The marginal cost of generation in the West is C_W, and the cost of wind generation is close to zero. Instead, I assume Power Purchase Agreement(PPA) price for wind generation, so the price of wind electricity in the West is imposed at $P_W = a_W$ where $a_W > 0$ at certain periods.

The decision whether to generate energy or not in the West does not depend on the load or market price but depends only on the weather condition. The amount of generation depends

on the wind farm capacity and the transmission capacity K that connects the West and the East. I assume that the total capacity of a wind farm is sufficiently greater than the transmission capacity. Q_W denotes the amount of wind generation in the West.

The East region is assumed to generate only fossil fuel energy and has a high load. I do not assume the monopoly generator in the East, so there is no market power among fossil fuel generators, and the price of the wholesale electricity market is cleared by offers and bidding under ISO. Under perfect competition assumption in the wholesale electricity market, the generators' offers to the market are equal to the marginal cost of power generation. Thus, the supply curve in the East electricity market is upward-sloping. (See 2.4 the right panel.) The total electricity of the East is supplied from both the East and the West, thus, $L_E = Q_W + Q_E$ where Q_E is the total fossil fuel generation in the East.

The load of the East is L_E. The total load is perfectly inelastic in this model because consumers face the fixed rate retail price instead of the wholesale electricity price, so the consumption of electricity tends not to be changed by the price. When wind generation is not available, the total load becomes the same as the fossil fuel generation, $L_E = Q_E$. When wind generation is available in the West, the total load becomes the sum of the generation from the East and the West $L_E = Q_E + Q_W$. The wind generation Q_W should be imported from the West regardless of the amount of fossil fuel generation because the wind is a must-taken energy and dispatched first due to its lower marginal cost. And, the Q_E also cannot be zero because some base-load generation such as nuclear power plants should be working for the long-term. I assume that L_E is large enough so that the fossil fuel generation will not be zero regardless of wind generation. Additionally, wind generation and transmission capacity cannot reach the total load in this model. $0 < Q_E \leq L_E$ and $0 \leq Q_W < L_E$

As follows the model of Joskow and Tirole (2005) and LaRiviere and Lyu (2022), the net demand function for the East is $P_E = a_E + b_E(L_E - Q_E)$ where $b_E > 0$ and $a_E > 0$. And the net supply from the West follows: $P_W = a_W$. And, I assume $P_W < P_E$ since the marginal cost of fossil fuel generation is higher than the marginal cost and contract price of wind generation.

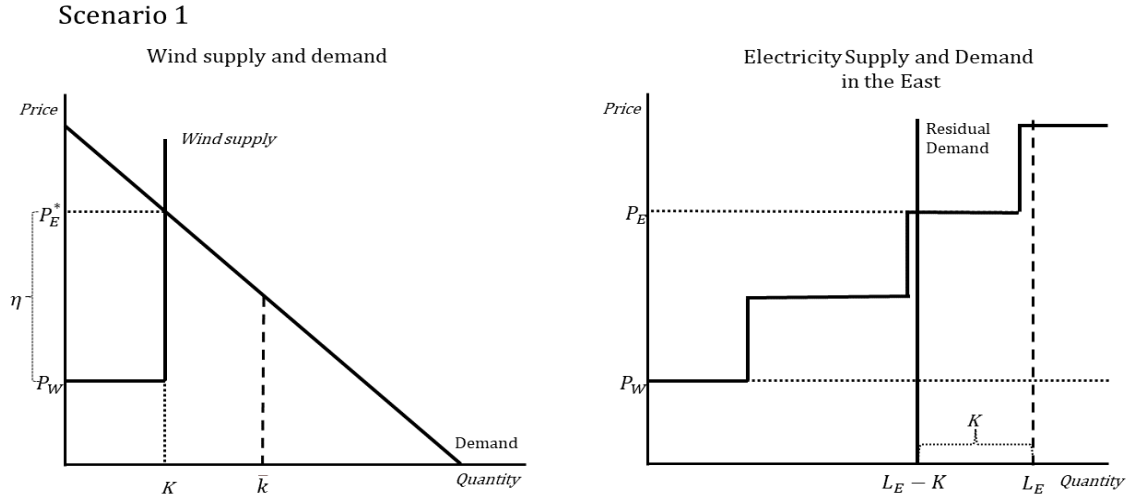


Figure 2.4: Scenario 1
Note: Wind generation is available and transmission line is binding constraint.

First, I assume the case of transmission is a binding constraint. Then, the total capacity of wind generation is Q_W^c but the actual wind farm generation becomes $Q_W^* = K$ at the transmission capacity. If then,the net demand for the East: $P_E = a_E + b_E(L_E - K)$, and the net supply from the West: $P_W = a_W$.

When wind turbine generates power, the shadow price of transmission constraint is the difference of the price between two regions, thus, $\eta = P_E - P_W = a_E + b_E(L_E - K) - a_W$ where and $K > 0$ and $K < L_E$. The East imports the electricity from the West at the price of $P_W + \eta$, which is equal to the price in the East, P_E. The power generation in the East is $Q_E^* = L_E - K$ and $P_E^1 = a_E + b_E(L_E - K)$. This is illustrated in figure 2.4.

For the next, I assume the transmission line is extended but still congested at some limits, and wind generation is available. The wind farm in the West produces electricity increases, and P_E falls because of the decrease in residual demand in the East. and, if I also assume there is no congestion in transmission and the wind capacity is sufficiently large so that it reaches $\bar{K}$, P_E is equal to P_W, the lowest marginal cost of electricity in both regions. η becomes zero because no one needs to pay for the congestion. (See figure 2.5)

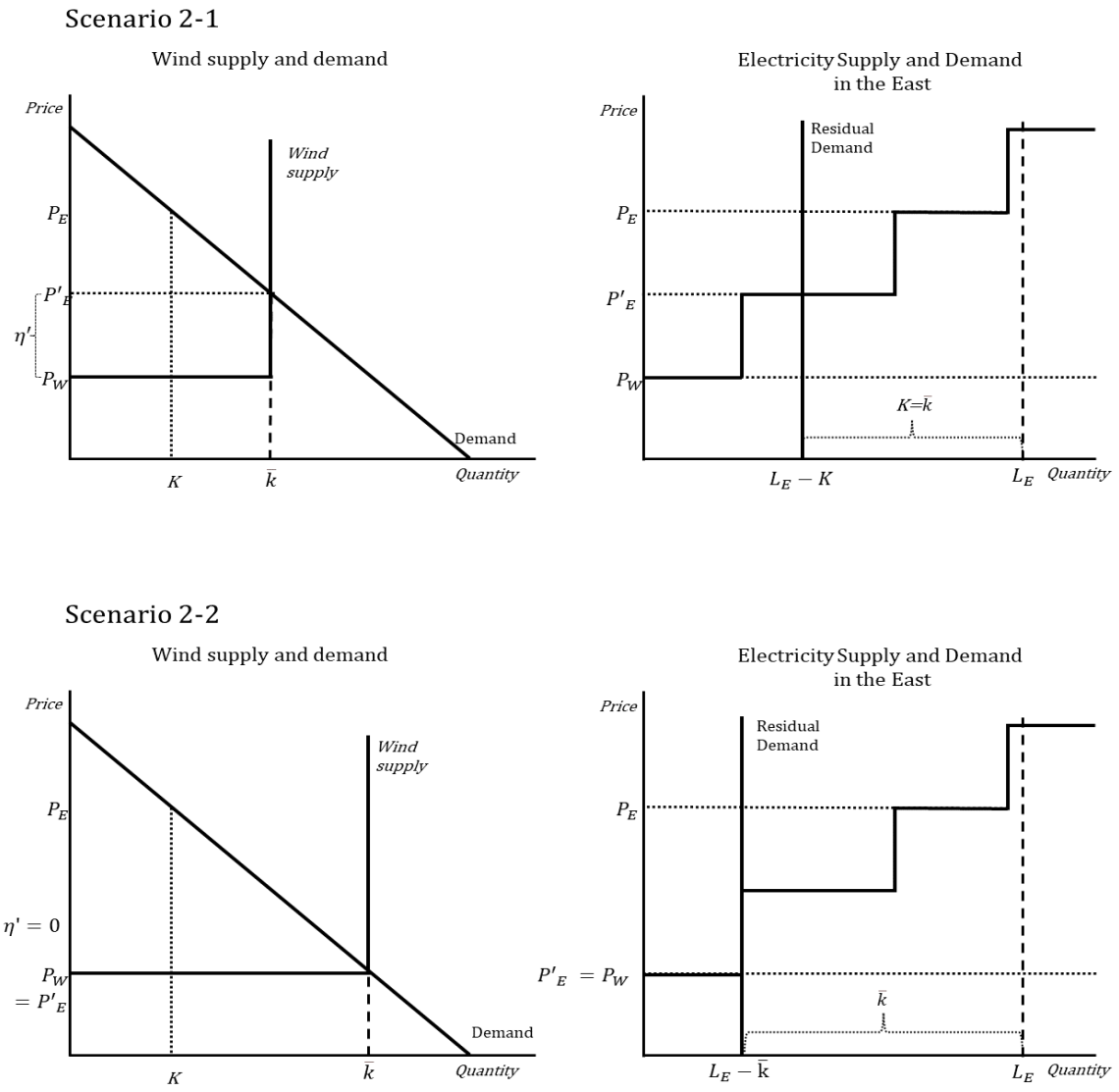


Figure 2.5: Scenario 2

Note: Wind generation is available when transmission line is extended but still binding constraint(Scenario 1), and when transmission line is fully extended (Scenario 2).

From scenarios 2-1, and 2-2, the shadow price of transmission decreases as the transmission capacity K increases: $\frac{\partial \eta}{\partial K} = -b_E < 0$, and the electricity price in the East P_E also decreases $\frac{\partial P_E^{wind}}{\partial K} = -b_E < 0$.

When wind generation is not available in the West because of weather conditions, the transmission is not binding regardless of its capacity because no electricity is supplied from

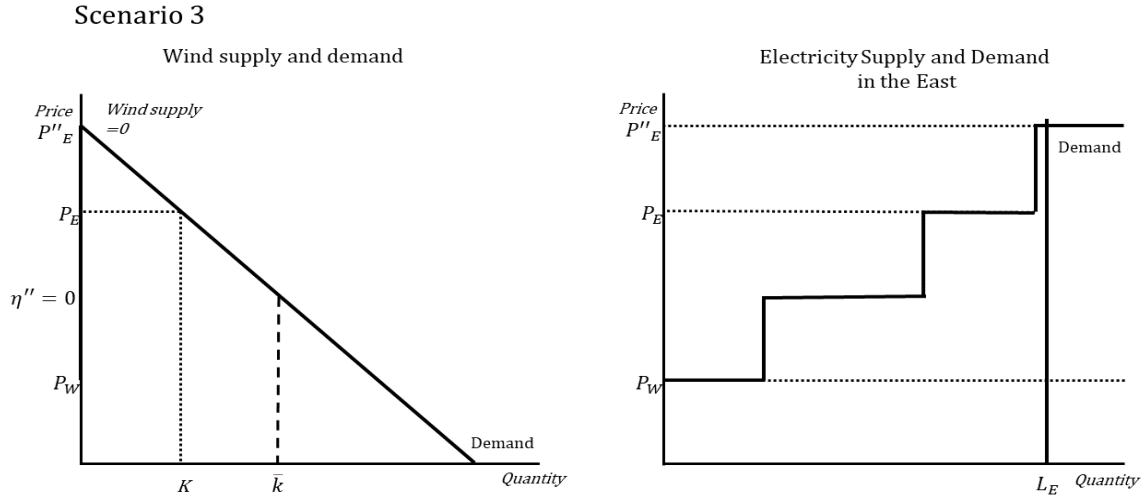


Figure 2.6: Scenario 3
Note: Wind generation is not available and transmission line is uncongested

the West. Then, $\eta = 0$ and $P_E = a_E + b_E L_E$. (See figure 2.6)

2.4 Data

In this section, I introduce the structure of power generation and fuel cost data, and I explain how the data set is merged for analysis. I also present the descriptive statistics of sample data.

2.4.1 Generation cost

Power generation data comes from Clean Air Market Division(CAMD)'s Air Markets Program Data (AMPD) of the Environmental Protection Agency(EPA), which is publicly accessible. AMPD includes the actual load of plant (unit) level power generation, the amount of hourly emission of CO_2, SO_2 and NOx, and the amount of input heat-rates (MMBtu[8]). AMPD also provides the

[8]Million British thermal unit. A British thermal unit is the quantity of heat required to raise the temperature of 1 pound of liquid water by 1 degree Fahrenheit at the temperature at which water has its greatest density (approximately 39 degrees Fahrenheit) at hourly level. In the United States, natural gas can be priced in units of dollars per therm, dollars per MMBtu, or dollars per cubic feet. Source:EIA

facility attributes that include power plants and unit specific information.

The Energy Information Administration (EIA) publishes the Annual Electric Power Industry Report (Form EIA 861) that provides the information about the geographic location and the type of electricity utility plants. After integrating AMPD and EIA data at hour and each plant-unit level, I choose fossil fuel plants located only in the Central and North MISO.[9]

Power plants of electricity utilities are major electricity supplier, and I also include the power generation of Independent Power Producers of non-cogeneration (IPP-Non CHP[10]) using fossil fuel because the IPPs also participate in some part of the power generation, transmission and distribution.

Most frequently used fossil fuels for power generation in the U.S are coal, natural gas and diesel oil(petroleum). Residual Oil, Petroleum Coke, and Wood are also used but do not include in the data set due to their small share of generation and the difficulties of cost estimation.[11]

Figure 2.7 illustrates the changes of the number of power plants by each fuel type in MISO during the sample periods(2013-2020). Plants have several units and each unit uses the same or different fossil fuel. There were 207 fossil fuel plants reported in 2013, but it decreases by 128 in 2020. In contrast, the number of natural gas plants(both combined cycle and combustion turbine) increased from 2013 to 2020. Overall, the number of total fossil fuel plants has been decreased in 2020.

I estimate the aggregate total and the average generation cost as a proxy for the cost of fossil fuel generation. Since the fixed cost of generation, for instance, Operation and Maintenance cost, is relatively low and constant in fossil fuel generation, I use only fuel cost as the marginal cost of fossil fuel generation. Since the variable cost accounts for most of the total power generation cost, typically in fossil fuel power generation. The unit price (dollar per MMBtu) of Natural gas, Petroleum(Oil), and Coal for electricity generation is obtained from the EIA database of the US delivered Fuel price at Monthly-State/Regional level.

[9]Central and North region covers MN, ND, MT, SD, IA, MI, IN, part of KY, and part of MO.

[10]Combined Heat and Power plant

[11]Some power plants and their units use the combination of fossil fuels to generate electricity, and I only use the primary fuel of each plant.

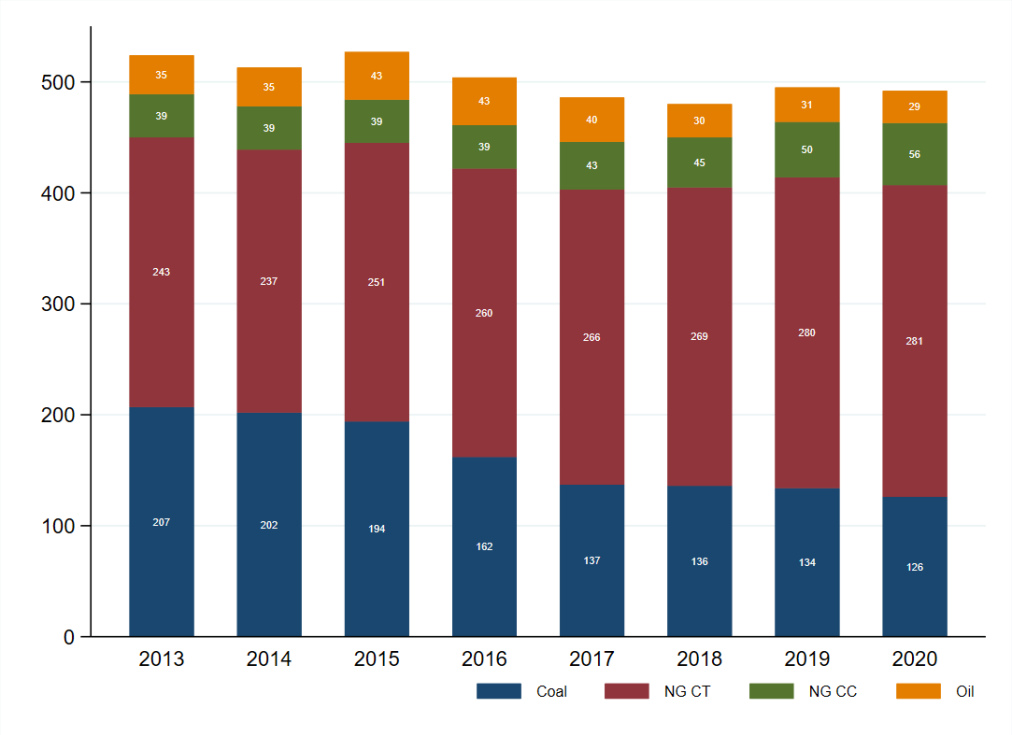


Figure 2.7: Changes of the Number of Plant-Unit in MISO Region

Note: The graph illustrates the number of each plant unit-level in MISO Central and North Region during sample periods. NG CC represents Combined Cycle turbine Natural Gas power and NG CT indicates all other types of natural gas plant including combustion turbine plant. Others include other gas, wood and petroleum coke plant. Source: AMPD

Equation 2.1 represents the basic estimation equation for the average fuel cost.

$$\text{Estimated average fuel cost(\$/MWh)} = \frac{\sum_h P_h^{fuel} \times HR_h \times \text{Generation}_h}{\sum_h \text{Total generation}} \tag{2.1}$$

2.4.2 Aggregation

I aggregate data at a daily-level instead of the original hour level. One of the reasons for aggregation is to avoid potential bias caused by hourly changed demand within a day. The exact time of transmission in-service is unknown to the public. Additionally, in general, the electricity demand and supply change hourly level within a day. Even though the new transmission line decreases the actual fossil fuel generation and its cost, if data is hourly level and electricity demand goes to peak hour, power generation must increase regardless of the effect of a new line.

This might cause bias in estimation. Fortunately, daily level aggregated electricity supply and demand are relatively stable, so it can capture the daily effect of the increase in transmission capacity. Although a daily level generation depends on the day of the week and holiday effects, it should be controlled in estimation. The other reason is that the fuel price is determined daily rather than hourly in energy markets.

I also combine all generations of plant-units in all MISO Central and North regions a day to avoid the heterogeneity among power plants. This research assumes that most coal plants will decrease their generation after the starting of a new transmission line due to the increase in supply of wind energy. However, some plants located far from the new transmission line might not be affected by the new transmission services, so they continue or even increase their fossil fuel generation for some unobserved reason. These heterogeneous behavior of power plants can affect the estimation of the regression at plant-unit level. The aggregated data can avoid this issue and, additionally, also remove the outlier problems.

2.4.3 Outliers

Sometimes natural gas and diesel oil plants (and their units) report 1-10 MWh generation with a high level of heat inputs. Since the estimation of generation cost depends on fuel price and the hourly amount of heat input, it could affect extremely high values when estimating the average cost and the total cost. Large heat rate but very small generation prevent measuring exact average generation cost because the denominator becomes too small in equation 2.1. To address this, I compare AMPD's gross generation and EIA 923's net generation in plant level and omit some erroneous observations.

In other cases, some plants report a very large amount of generation and it leads to a very high level of generation costs. However, It is not an erroneous record but a general situation when fossil fuel plants start their generation initially. Thus, these observations should be considered as part of the cost of power generation and included in the final data set.

2.4.4 Descriptive statistics

The number of coal plants has decreased from 2013 in most states in Midwest.(Figure 2.7) Coal generation was 71.01% of total in 2013, however, its share has dropped to 49.04% in 2020(See figure 2.8). Coal plants have been retired gradually for decades, however, it still has a great share of the total power generation in MISO Central and North region. The number of natural gas plants including combined cycle turbine generation and combustion turbine increases, and the ratio of power generation also increases over time. It is because of relatively lower emission of carbon from natural gas generation and the low and stable price of natural gas in recent years.

The share of renewable energy power plants, especially wind, has increased since 2013. The percentage of wind generation has been increased from 2.57 to 17.48. (Figure 2.8).[12]

The entire power generation in MISO north and central regions has no big changes from 2013 to 2020. It is natural because the demand for electricity is relatively stable, and the supply of electricity should meet the demand, the power generation should be stable regardless of the change of composition of fuel. Increased wind generation has replaced fossil fuel generation. The impact of COVID19 on electricity demand in the Midwest is not clearly shown in total load.

The average cost of fossil fuel generation in MISO was decreasing from 2013 to 2020. (Figure 2.10 panel (b)) The decrease in the average cost of coal generation, which has the largest share in fossil fuel generation, drives this trend. The average cost of natural gas combined cycle generation is relatively lower than other types of natural gas generation. The stable price of natural gas for a couple of years leads to the increase in share of natural gas generation and of coal power plants, which also affects the long-term decrease of the average cost of fossil fuel generation.

The total cost of fossil fuel generation is constantly decreasing over time (Figure 2.10 panel (a)), which is led by the decrease in the average cost of coal generation (2.10 panel (b)) and the amount of fossil fuel generation itself (Figure 2.8). The average cost of fossil fuel generation

[12]MISO starts to adopt solar energy in 2021.

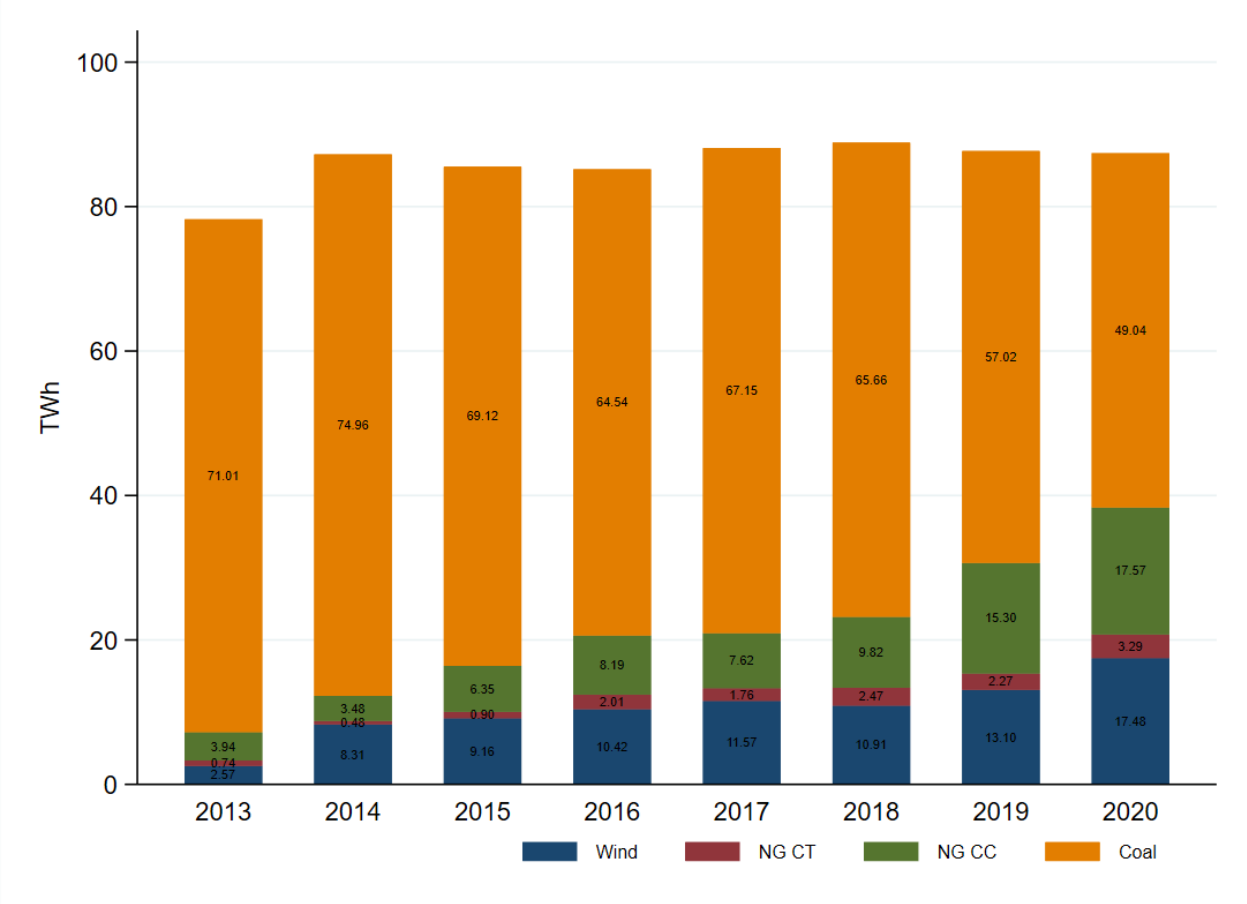
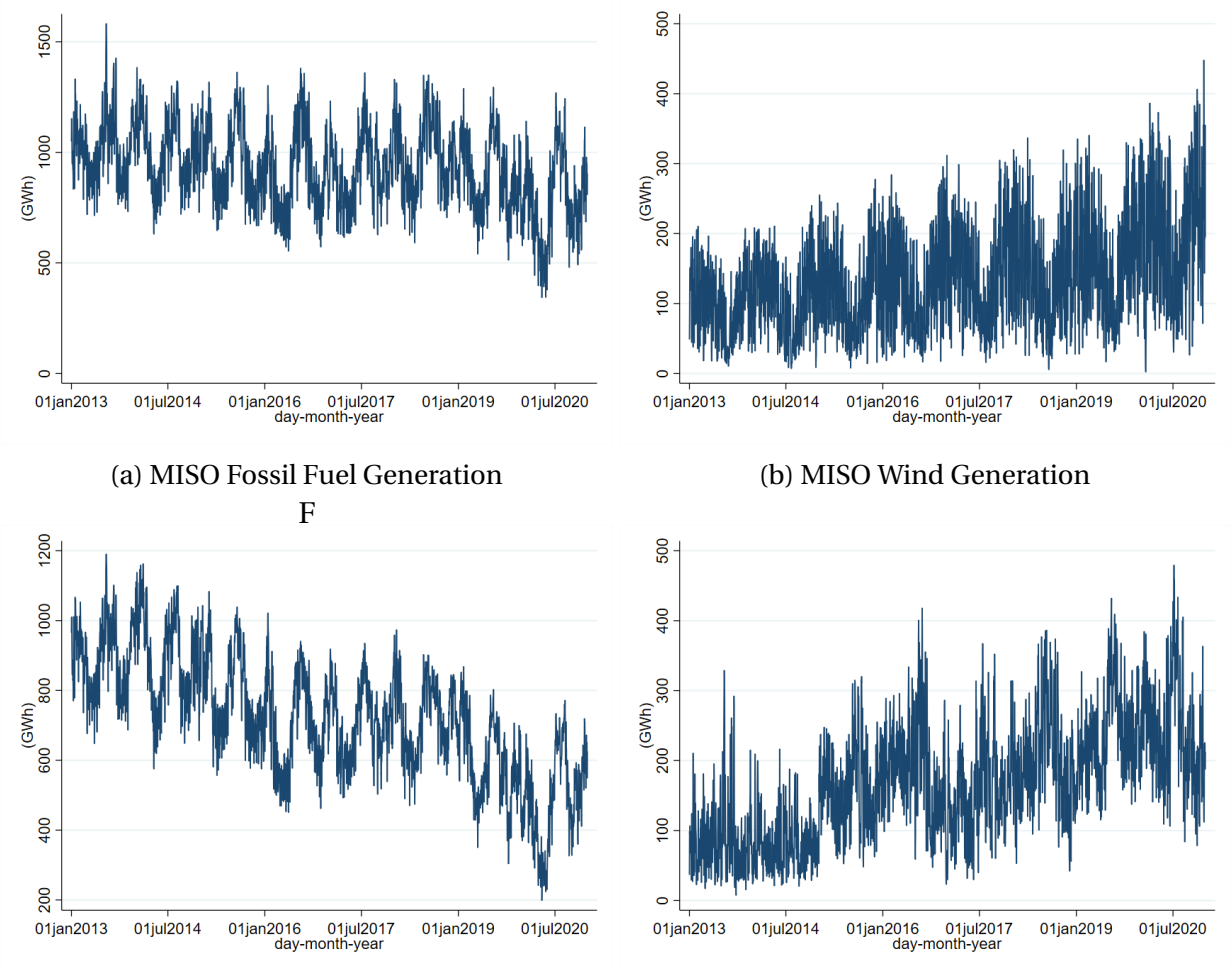


Figure 2.8: Coal, Natural gas, and Wind generation

Note: Each bar indicates the amount of power generated each year. The share of other generation such as biomass, nuclear, and hydro power generation are less than 1%, and they are omitted in this figure. NG CC indicates the combined cycle natural gas generation, and NG CT indicates the other types of natural gas generation including combustion turbines.

decreases over time, from 22.4$/MWh to 17.8$/Mwh, which is led by the decrease in coal prices and natural gas prices. (Figure 2.10 panel (b)) The average cost is not affected by the entire power generation and indicates the efficiency of generation increases.

Table 2.2 shows descriptive statistics during the sample periods (2013-2020). The average cost of coal generation is 20.61 dollars per MWh and that of combined cycle natural gas generation is 24.9 dollars per MWh. The average cost of a combustion turbine is 40.31 dollars per MWh. These indicate the inefficiency of the power generation of combustion turbine natural gas.



(a) MISO Fossil Fuel Generation

F

(b) MISO Wind Generation

(c) MISO Coal Generation

(d) MISO Natural Gas Generation

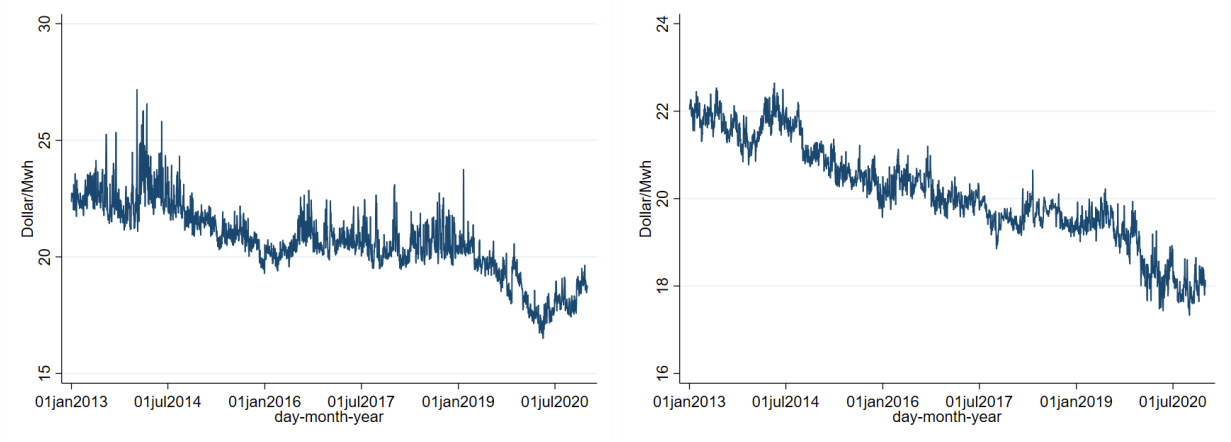
Figure 2.9: The Trend of Power Generation in 2013-2020

Source: MISO, Natural gas generation includes combined cycle and combustion turbine generation.

2.5 Empirical Strategy

2.5.1 Identification problem

Ideal identification strategy of this study is to measure the true treatment effect by comparing outcomes in treated area and non-treated area after transmission expansion. However, cross-sectional analysis such as a difference-in-difference is not available in this case. It is impossible to define treated area and non-treated area accurately due to the characteristics of the electricity grid. All the electricity grids are connected within ISO territory and also adjacent ISOs import



(a) The Average Cost of Fossil Fuel Generation (b) The Average Cost of Coal Generation

Figure 2.10: The Change of Generation Cost 2013-2020

Source: AMPD

and export electricity. The effect of large transmission expansion would not be limited in certain treated areas so that the non-treated area after the treatment cannot be observed correctly. Despite this geographical problem, the time of the treatment is clear and constant. Once a new transmission line starts its service, its effect continues and all areas are treated.

The main approach for this study to measure the treatment effect of a new transmission line is Regression Discontinuity Design in Time, reviewed by Hausman and Rapson (2018). RD in Time is different from standard cross-sectional RD framework in several aspects, but it is a proper empirical strategy where the time is the running variable and the treatment begins at a particular threshold in time. I focus on identifying the local impact of treatment using RD in a time framework with ambient time series data of power generation and electricity market price data.

2.5.2 Regression discontinuity design

Regression Discontinuity (RD) design is one of the widely used nonexperimental methods for causal inference in social science. The regression discontinuity (RD) design was introduced by Thistlewaite and Campbell (1960) and The RD design has become one of the most credible non-

Table 2.2: Summary Statistics

Panel (a): Generation (GWh)						
	Obs.	Sum(TWh)	Mean	Min	Max	SD
Fossil fuel total	2,922	2,731	934.62	343.27	1,569.92	180.13
Coal	2,922	2,351	804.57	230.38	1,270.51	181.92
Natural Gas (Combined Cycle)	2,903	318	109.71	0.00	285.02	63.24
Natural Gas (Combustion Turbine)	2,846	61	21.57	0.00	188.99	29.05
Panel (b): Total Generation Cost (Million dollars)						
	Obs.	Sum	Mean	Min	Max	SD
Fossil fuel total	2,922	56,726	19.41	5.84	39.52	4.65
Coal	2,922	47,634	16.30	4.22	27.65	4.25
Natural Gas (Combined Cycle)	2,903	6,992	2.41	0.00	6.53	1.12
Natural Gas (Combustion Turbine)	2,851	2,071	0.73	0.00	6.62	0.95
Panel (c): Average Cost of Generation (Dollars/MWh)						
	Obs.	Sum	Mean	Min	Max	SD
Fossil fuel total	2,922	60,219	20.61	16.41	26.97	1.56
Coal	2,922	58,729	20.10	17.33	22.58	1.13
Natural Gas (Combined Cycle)	2,903	72,246	24.89	13.67	86.98	9.16
Natural Gas (Combustion Turbine)	2,846	114,710	40.31	5.67	291.22	19.69
Panel (d): Other variables						
	Obs.	Sum	Mean	Min	Max	SD
Wind generation (GWh)	2,922	369,561	126.48	2.24	447.90	78.78
Real-Time Actual Load (GWh)	2,771	3,755,728	1,354.90	0.00	1,930.68	163.90
NG Spot Price (Dollars)	2,922	8,765	3.00	1.33	8.15	0.85

Note: 1 Gigawatt hour (GWh) is 1,000 Megawatt hour (MWh). Natural gas(Combustion turbine) includes the other single turbine generation of natural gas. The hourly level data (generation and actual load) are aggregated to daily level. Sample periods are from January 1st in 2013 to December 31th in 2020. All the plant, utility, generation, and load data are limited to only MISO Central and North territory.

experimental methods for causal inference and program evaluation. (Cattaneo and Titiunik (2020)) The advantage of RD design is that it can be employed when the randomized treatment assignment is impossible. Another advantage is that it requires mild assumptions than other non-experimental approaches such as the difference-in-difference or IV.

RD design consists of three key components: a score(running variable), a cutoff(threshold), and a discontinuous treatment assignment. Every individual or observation should be assigned a value of a running variable, and the treatment is assigned only to the individual whose running variable exceeds the cutoff. The individuals below the cutoff are used as a control(counterfactual) group and outcomes of each individual might have discontinuity above

the cutoff. In this section, I bring the definition of the canonical RD design from Cattaneo and Titiunik (2020) that the score and the treatment are one-dimensional, and the treatment is binary.

In sharp RD design, the probability of receiving treatment is 1 if the outcome is above the cutoff, and the probability of treatment is 0 if the outcome is below the cutoff. The assignment of treatment is different from the receiving or complying with treatment. In RD design, every individual should be assigned the treatment if their score is above the cutoff, but they might not receive the treatment in some cases. It is called Fuzzy RD, and it will be discussed in the next section.

For valid RD design, RD design requires key assumptions.(Lee and Lemieux (2010)) If individuals can precisely manipulate the score, the RD design is invalid. Since individuals cannot precisely manipulate the score, the outcome of an individual near the cutoff is randomly assigned like a randomized experiment. In other words, the probability that the score is placed above or below the cutoff should be random. Another key assumption is the continuity of the score near the cutoff. If the running variable is discrete, it should be considered in a different approach.

In general causal effect inference, the random assignment of treatment should be assumed, and it is a quite strong assumption in quasi or nonexperimental research in social science. Thus, the continuity assumption is required to avoid the overlap condition. Since $T_i = 1$ is always true at $X \geq c$ and $T_i = 0$ at $X < c$, T cannot be correlated with any other factor, and one observation cannot be both $T_i = 0$ and $T_i = 1$, the assumption of overlap is cannot be violated in RD design under continuity assumption.

The average treatment effect

Assume that X_i denotes the score variable, D_i is the observed treatment assignment, and Y_i is the outcome. And, the cutoff point denotes c. Since the treatment is binary, $T_i = 1\,(X_i \geq c)$ treatment assignment, and $T_i = 0\,(X_i \leq c)$. There are two potential outcomes $Y_i(0)$ (outcomes

without the treatment $T_i = 0$) and $Y_i(1)$: outcomes with the treatment $T_i = 1$. In the sharp RD design, the probability of treatment assignment changes at the cutoff from zero to one. $Pr[T_i = 1|X_i = x] = 0$ if $x < c$ and $Pr[T_i = 1|X_i = x] = 1$ if $x > c$.

And, $Y_i = Y_i(0)$ for individual with $T_i = 0$ with $X_i < c$, and $Y_i = Y_i(1)$ for individual with $T_i = 1$ with $X_i > c$.

The ultimate goal is to capture the causal effect by comparing the outcome with an individual who would receive treatment and an individual who would not receive treatment, $Y_i(1) - Y_i(0)$. However, the problem is that it is not possible that an individual receives the treatment and not receive the treatment simultaneously. Instead, RD design can capture the observations that are close to the cutoff and compare the outcomes.

If an individual is at $X = c$, the causal effect of that individual can be captured. However, no observation is available because of continuous assumptions. However, a researcher can use observation near discontinuity. Thus, the RD estimates can use observations near the cutoff. It can capture the causal effect of treatment. The average outcome of individuals whose score is right below the cutoff and does not receive the treatment is a counterfactual.

To specify, RD design focuses on the average treatment effect of $Y_i(1) - Y_i(0)$ of individuals whose running variable is near the cutoff, represented by $E[Y_i(1) - Y_i(0)|X = c]$. The average treatment effect in sharp RD design τ:

$$\tau = E[Y_i(1) - Y_i(0)|X = c] = \lim_{x \downarrow c} E[Y_i|X_i = x] - \lim_{x \uparrow c} E[Y_i|X_i = x] \tag{2.2}$$

For local polynomial estimation, after the decisions of a polynomial order, a kernel function, and a bandwidth, τ in equation 2.2 is estimated by running regression below and above the cutoff separately. The regression model on the left-hand side of the cutoff ($X < c$) and the right-hand side of the cutoff ($X \geq c$) are,

$$Y = \alpha_l + f_l(X - c) + \varepsilon$$

$$Y = \alpha_r + f_r(X - c) + \varepsilon$$

where $f_l(\cdot)$ and $f_r(\cdot)$ are functional forms. The treatment effect is the difference between the two regression intercepts on both sides of the cutoff, α_r and α_l. $\tau = \alpha_r - \alpha_l$.

To estimate the treatment effect directly, one runs the pooled regression model:

$$Y = \alpha_l + \tau T + f_l(X - c) + T[f_r(X - c) - f_l(X - c)] + \varepsilon \tag{2.3}$$

More specifically, I can apply the linear regression model, then the pooled equation 2.3 can be represent by

$$Y = \alpha_l + \tau T + \beta_l(X - c) + (\beta_r - \beta_l)T(X - c)] + \varepsilon \tag{2.4}$$

where $c - h \geq X \geq c + h$.

2.5.3 Regression discontinuity in time model

In this section, I introduce Regression Discontinuity in Time model, which is reviewed by Hausman and Rapson (2018). This approach is useful when there is no cross-sectional variation in policy implementation but sufficient observations are available at a daily or hourly frequency over a long time horizon.

RD in time uses time as the running variable and a certain particular date is a cutoff. It is used in time series data of daily or hourly observations, for a given geographic region. Specifically, the date c, the cutoff, of a policy change is known and not generally randomly assigned. The cutoff date is chosen by policymakers and announced in advance. The treatment effect is constant above the cutoff. For all dates $t > c$, observations are treated, and for all dates $t < c$, observations are not treated. And, once the treatment is assigned, its effect is constant over time. As Davis(2008) points out, an advantage of RD in Time is the ability to include flexible controls driven by high-frequency data.

The standard RD design is considered a local randomized experiment (Lee and Lemieux

(2010)) and one of Randomized Control Trial(RCT) for testing causal inference since whether the running variable is above or below the cutoff is as good as random. In contrast, the RD in Time approach has focused more on the discontinuity at the cutoff in RD design as emphasized by Hahn et al. (2001). Time cannot be thought of as randomly assigned near the cutoff, which is close to the quasi-experimental framework.

The main difference between RD in Time from the standard RD is that RD in time uses time as the running variable and a certain particular date is a cutoff. However, in contrast to the time series approach, it is RD design that time is the running variable but the framework approaches as a random cross-sectional RD.

Another difference is the sample size. The standard RD has sufficient observations near below and above the cutoff, but RD in Time has insufficient or no cross-sectional variation, which leads to a small sample size for estimation as the bandwidth narrows around the cutoff. It makes estimation depend also on observations far from the cutoff, which is different from the standard RD.

RD in Time expands the time horizon to obtain enough power on estimation or to absorb the seasonality and weather effects. However, any potential confounders such as other policy changes or weather events during the time must be controlled or absorbed by the global polynomial approximation. Of course, it can lead to bias resulting from observable confounders and time-series properties of the data-generating process. Instead, increasing the frequency of the data to increase the sample size and the power of estimation cannot work if the data are serially correlated.

The inclusion of control variables is also different. In standard RD, few controls are needed since treatment is already randomly assigned within a narrow bandwidth. In contrast, RD in time, high-frequency data is highly variable so control variables are important for absorbing noise and preventing bias in RD in Time framework. Previous RD in Time literature, for example, includes discontinuous controls such as day-of-week or weekend effects.

Additionally, the assumptions for inference are different due to its long time horizon. The

errors are likely to exhibit persistence so the serial correlation should be controlled. Some dependent variables in the RD in Time framework might contain unit roots[13], so it requires different procedures for inference.

This time-series variation is one of the challenges in RD in time. The time-series nature of the underlying data-generating process such as the auto-regressive and serial correlation of the dependent variables should be considered when implementing RD in Time.

If there is serial dependence in the error term[14], standard errors must account for it. Much literature on RD in Time addresses this by using clustered standard errors.

High-frequency data may be more likely to exhibit qualitatively autoregression[15]. If autoregression is found in the outcome variable, the lagged dependent variable in regression can be a solution. However, if only an error term has a serial correlation, the lagged dependent variable would be misspecification.

Another drawback of RD in Time is the density test issue. It is not applicable to test the discontinuities in other covariates and the outcome variables at the cutoff by testing the conditional density of the running variable since the time is uniformly distributed. The causal treatment effect and any unobserved effects that may exist but cannot be tested for in RD in Time. However, the absence of these effects is a necessary but insufficient condition for identification.

In this paper, the treatment effect is the transmission expansion. The main outcome variable is the average cost of fossil fuel power generation. The running variable is one dimension of the time variable-daily level without omitted value. The score is continuous at the cutoff (the in-service date). The cutoff date is randomly assigned, it is chosen based on the process of line construction, weather condition, and other considerations. The cutoff date is not dependent

[13]Statistically, the existence of a unit root in time series can cause problems in inference. If a time series is stationary, its statistical properties or moments (mean and variance) do not vary over time. The existence of a unit root in a time series means that a series is not stationary and some analytical tools such as ordinary least squares cannot be applied. The unit root is easily tested by using the Augmented Dickey-Fuller test.

[14]the correlation of the error term means that the error in one period is correlated with the errors in other periods. $\varepsilon_t = \rho\varepsilon_{t-1} + u_t$ $(-1 < \rho < 1)$. Serial correlation causes a biased variance in estimation.

[15]Autoregression is when time series value in one period is regressed on previous value from same time series. For example, AR(1) process follows $y_t = \alpha y_{t-1} + \beta + \varepsilon_t$.

on the outcome variables or running variable, so the treatment effect (transmission expansion and its new service) does not affect the running variable. And, once the new line energizes, the treatment effect goes constantly and all the regions within the grid are assigned and receive the treatment effect. Of course, before the cutoff date, the treatment cannot be assigned in any regions or nodes.

RD in time approach can be applied properly in this framework because data has a sufficiently long time horizon (from 2013 to 2020) at high frequency (daily level), and the most important thing is that there is no cross-sectional variation. Since the effect of the transmission grid is, directly and indirectly, spread throughout the entire region, it does not make sense to compare the trend of the two region's outcomes after the cutoff. In the RD in time framework, I can focus more on discontinuity at the cutoff.

Doshi and Du (2021) also uses the RD approach to measure the effect of transmission expansion on financial transmission rights(FTR) in TEXAS after CREZ projects. They use the cutoff as the completion of the CREZ project and the running variable is also time, at a monthly level because FTR is a monthly auction in ERCOT. They use all the FTR prices at each node without aggregation and sample periods are from 2011 to 2016.

2.5.4 Empirical strategy

In the RD in time framework, observations are highly noisy due to their high-frequency, which may cause bias in the estimation. To address this, the control variables are necessary to observe noise and decrease potential bias. Also, other time-series-specific treats such as serial correlation in error terms or autocorrelation between outcome variables should be controlled. The serial correlation can be controlled by the clustered standard error, and the autocorrelation can be addressed by using a lagged variable if it exists. To test this, I need to conduct a unit-root test in advance.

To absorb seasonality, I need a very long bandwidth. At least one year of the time window is required to capture seasonality in daily level frequency. This long window is general in RD

in time estimation, but it may cause bias under RD design. Additionally, there were several additional small and trivial transmission construction during the window, so I have to consider and deal with such confounders. One of the solutions is that the focused transmission project is reduced to MVP long projects only, and to expand timeline, and other transmission projects built adjacent areas or trivial projects turn to dummy variables to control it.

I use the local linear estimation model (the order of one) as a default because the regression uses a very large window (about a year on both sides) so it is hard for the higher-order polynomial model to capture the true model, and too high-order polynomial model is not recommended. I additionally test the quadratic model to check the misspecification.

About kernel selection, I can choose the triangular kernel with MSE optimal bandwidth choice. However, in RD in a time framework with large bandwidth, it might not be proper because of a long time. If the observation that is far from the cutoff may have zero weight, it is not preferred in this model. Thus, in RD in time design, I need to use a uniform kernel instead.

Since this framework requires the covariates necessarily, the related issues also should be considered. The day-of-week and month dummies satisfy the requirements of covariates because those are independent of the treatment effect and time has uniform distribution below and above the cutoff date.

Wind generation is exogenous from any other variables because wind generation depends only on the weather condition. The natural gas spot price is also not affected by the treatment effect, because it is decided by the supply and demand of natural gas. The real-time total load is also not affected because consumers usually pay the fixed rate of the retail price and the electricity consumption is not changed by the electricity or congestion price in the wholesale market.

Final estimation model is as follows:

$$y_t = \alpha + \tau \cdot D_t + \theta_1 \cdot Trend_t + \theta_2 \cdot D_t \cdot Trend_t + \beta_1 \cdot \boldsymbol{X}_t + \beta_2 \cdot \boldsymbol{X}_t \cdot Trend_t + \varepsilon_t \tag{2}$$

Equation 2 represents the linear Regression Discontinuity in Time. y_t represents the average cost of power generation ($/MWh) in a given region. t denotes time (day). The term D denotes the treatment effect. The value of D is 1 if t is greater than c (cut off date, the day the new transmission line starts its service), 0 otherwise. $Trend$ is the linear time trend. τ is a treatment effect coefficient which is the coefficient of interest. $\boldsymbol{X}_t$ is a vector of control variables, which includes Real time actual load, Henry Hub Natural Gas Spot price (daily) [16], Real-Time wind generation in across Central and North MISO region. Real time load and wind generation are exogenous to fossil fuel input prices (LaRiviere and Lyu (2022))) so that it controls the potential bias on fossil fuel generation caused by the increase in demand and wind generation. X_t includes time fixed effects such as day-of-week, month, and year dummies to control unobserved time specific shocks of each day. Finally, ε represents other unobserved shocks.

Local linear regression estimates τ on the in-service date of the new transmission line. Basically I use two types of windows. First, a fixed window (90 days) and window decided by a Mean squared Error(MSE)-optimal choice calculation. The standard errors are clustered by the day of weeks to account for possible serial correlation within days.

2.6 Empirical Results

The most interesting outcome from transmission expansion is the effect on the average cost of fossil fuel generation and how it affects the merit order effect. To begin with, I look at the change of power generation first. And, I show the estimation result of the average cost including the different effect by region and specific fuel type.

[16]The observations are daily level and only exist on weekdays. The observation on weekends and holidays are missing. Since the demand and supply of electricity exist regardless of holidays, I interpolate the missing values of Natural Gas spot price.

2.6.1 The effect on the average cost

I hypothesize that the coefficient of the average treatment effect will have a negative sign. If a new large transmission line starts service, it should deliver additional wind energy from farms to high-demand areas. This would displace some higher marginal cost fossil fuel generation, as the dispatch order prioritizes lower cost wind. Consequently, bringing the new transmission online leads to decreasing the overall average cost of power generation.

Figure 2.11 shows Regression Discontinuity plots of average costs for fossil fuel generation—including coal, combined cycle natural gas, and combustion turbine—before and after the in-service date of each transmission project. Optimal bandwidth is selected by minimizing the mean squared error (MSE). Bins are evenly spaced using a mimicking variance estimator with a uniform kernel. Some projects exhibit immediate average cost decreases after coming online. Others display higher initial intercepts but downward-sloping post-start cost trends.

Table 2.3 displays local linear regression discontinuity estimates for the regression coefficient (τ) from equation 2. Bandwidths are selected to minimize the mean squared error (MSE) using a triangular kernel. Each element of table 2.3 represents the coefficient of the Regression Discontinuity regression. Standard errors are clustered by day of week. Column (1) has no control variables. Column (2) includes wind generation, daily load, and natural gas spot price as control variables. Column (3) includes fixed effects of the day of week and month. Column (4) includes a event dummies.

After controlling for additional factors, all projects display decreases in the average fossil fuel generation cost following new transmission lines, ranging from \$0.248/MWh to \$1.502/MWh. Controlling for the reduction of natural gas price and the change of load, the average cost of fossil fuel generation still decreased after Multi-Value Portfolio projects finishing construction that year. Expanded transmission infrastructure appears to directly reduce fossil fuel electricity expenses in practice.

Table 2.4 displays estimated average cost changes for MISO North and Central regions. After controlling for additional factors, the Central region shows consistent, statistically significant

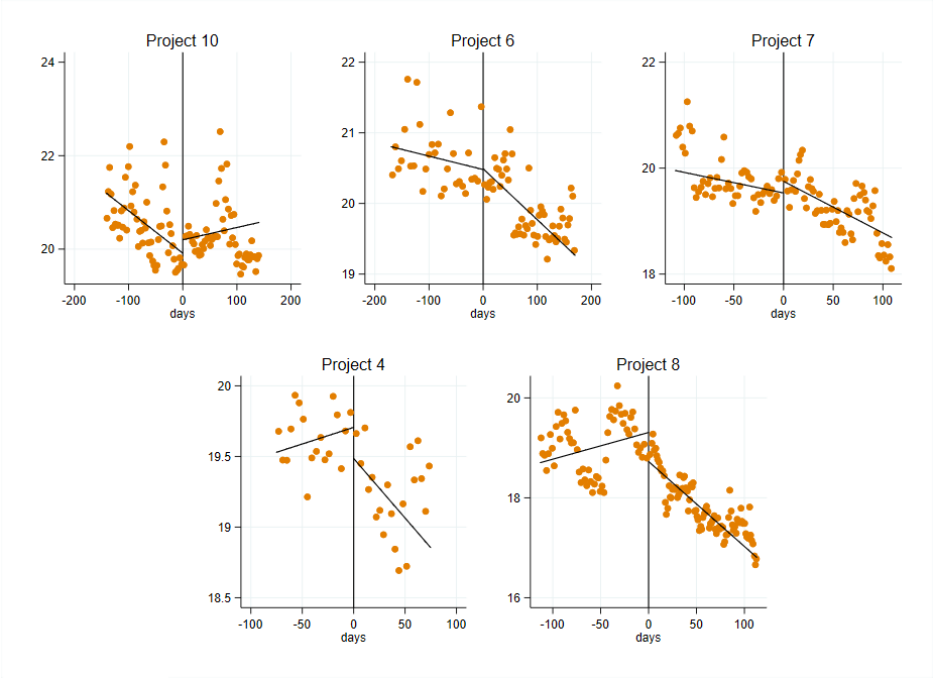


Figure 2.11: RD Plots of the Average Cost

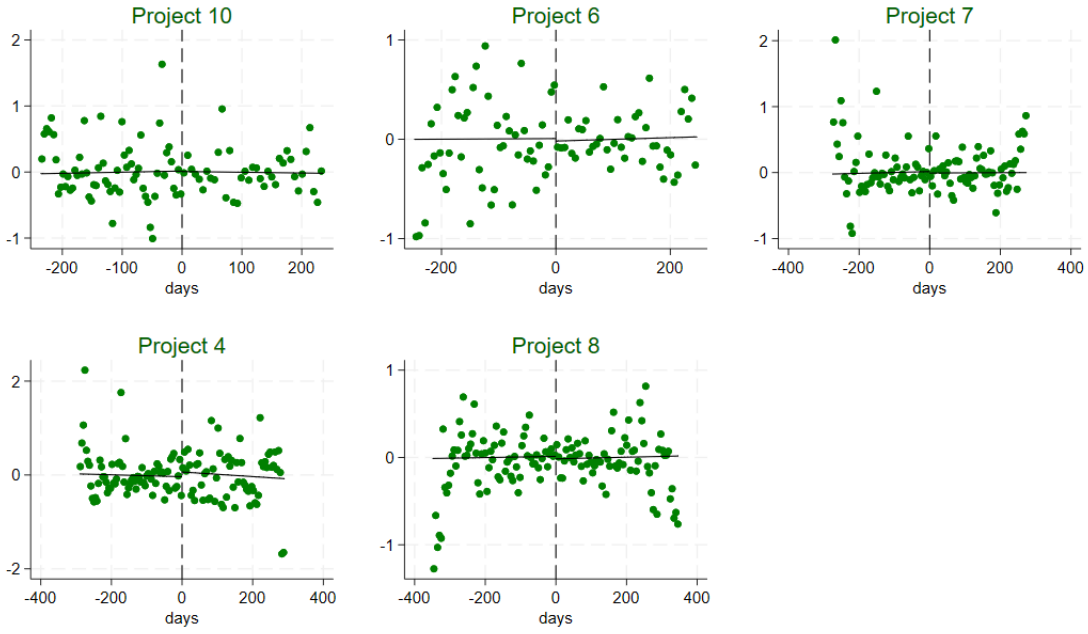


Figure 2.12: RD Plots of the Average Cost(Residuals)

Note: Regression Discontinuity plots of average fossil fuel generation cost, including coal, combined cycle natural gas, and combustion turbine natural gas. Optimal bandwidth is selected to minimize the mean squared error (MSE). Bins are evenly spaced using a mimicking variance estimator with a uniform kernel. A linear order polynomial (order 1) is used in the model.

Table 2.3: Local Linear Regression of the Average Cost: Auto Bandwidth

	Dependent variable: the average cost($/MWh)			
Project Number	(1)	(2)	(3)	(4)
Project 10	0.307***	-0.311*	-0.364***	-0.147
	(0.09)	(0.14)	(0.09)	(0.09)
N	274	180	264	246
Project 6	0.057	-0.438**	-0.741**	-0.813***
	(0.15)	(0.14)	(0.26)	(0.22)
N	370	226	108	116
Project 7	0.183*	-0.087	-1.502***	-1.470***
	(0.08)	(0.05)	(0.05)	(0.05)
N	200	166	256	248
Project 4	-0.112	0.188**	-0.522***	0.001
	(0.13)	(0.07)	(0.08)	(0.07)
N	158	134	296	224
Project 8	-0.176	0.522***	-0.248***	-0.338***
	(0.09)	(0.13)	(0.04)	(0.03)
N	120	54	308	334
Controls		x	x	x
Fixed effects			x	x
Event dummies				x

Note: * p < 0.05, ** p < 0.01, *** p < 0.001. Standard errors are in parentheses and clustered by the day of week. Bandwidths are selected to minimize the mean squared error (MSE) using a triangular kernel. Each element represents the coefficient of the RD regression. Column (1) has no control variables. Column (2) includes wind generation, daily load, and natural gas spot price as control variables. Column (3) includes fixed effects of the day of week and month. Column (4) includes a event dummies.

decreases. In the North region, only Project 8 increased average costs, but coefficients of other projects are statistically insignificant but negative. Reductions appear driven primarily by the Central region.

Table 2.5 shows the average cost changes by fuel type, controlling for fixed effects and additional factors. Except Project 10, the average cost of coal generation decrease across projects. However, substantial total reductions derive from falling combined cycle and combustion turbine natural gas generation, the highest and lowest marginal cost major fuels respectively. Meanwhile, small-share oil generation cost changes do not seem to correlate with total shifts

Table 2.4: Local Linear Regression of the Average Cost by Region

	Dependent variable: the average cost($/MWh)					
	MISO		North		Central	
Project Number		Control		Control		Control
Project10	0.307***	-0.364***	0.943***	-0.015	-0.120	-0.686***
	(0.09)	(0.09)	(0.13)	(0.11)	(0.08)	(0.09)
N	274	264	342	252	254	240
Project6	0.057	-0.741**	0.557**	-0.507	-0.190	-0.921**
	(0.15)	(0.26)	(0.20)	(0.27)	(0.17)	(0.29)
N	370	108	216	116	322	106
Project7	0.183*	-1.502***	0.334**	-0.947***	0.034	-1.839***
	(0.08)	(0.05)	(0.12)	(0.09)	(0.08)	(0.04)
N	200	256	184	218	100	294
Project4	-0.112	-0.522***	-0.029	-0.071	-0.076	-0.882***
	(0.13)	(0.08)	(0.19)	(0.09)	(0.13)	(0.06)
N	158	296	130	258	138	326
Project8	-0.176	-0.248***	0.276*	0.155**	0.550**	-0.343***
	(0.09)	(0.04)	(0.12)	(0.05)	(0.19)	(0.05)
N	120	308	58	334	68	254
Controls, FE		Yes		Yes		Yes

Note: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Standard errors are in parentheses and clustered by the day of week. Bandwidths are selected to minimize the mean squared error (MSE) using a triangular kernel. Each element represents the coefficient of the RD regression. Columns (2), (4), and (6) control for variables including wind generation, real-time load, natural gas market price, as well as fixed effects for the day and month. All variables are limited to MISO Central and North region generation.

of the average cost.

I also examine the effect during on-peak hour. The definitions of on-peak and off-peak hours follow MISO's operations. [17] Table 2.6 displays the impact on on-peak average costs. Columns (3) and (4) present controlled estimates. As in Table 2.3, average costs consistently fall. Larger coefficients compared to all hours indicate the effect of reductions are more significant during peak periods.

[17] Peak hours are weekdays, Monday through Friday, from Hour Ending 7:00 AM to 10:00 PM. Off-peak includes weekends all hours (1:00 AM - 12:00 AM), holidays, and weekdays from Hour Ending 1:00 AM-6:00 AM and 11:00 PM-12:00 AM. Holidays includes New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving Day, Christmas Day The Off-Peak day will occur on the Monday immediately following a holiday that occurs on a Sunday. Source: https://help.misoenergy.org/knowledgebase/article/KA-01093/en-us

Table 2.5: Local Linear Regression of the Average Cost by Fuel Type

	Dependent variable: the average cost($/MWh)				
	Total	Coal	NG CC	NG CT	Oil
Project10	-0.364***	0.060*	-0.377*	-3.800	-20.913
	(0.09)	(0.03)	(0.17)	(2.23)	(23.42)
N	264	426	360	584	256
Project6	-0.741**	-0.277***	-0.158	-6.923***	-68.913
	(0.26)	(0.04)	(0.19)	(1.23)	(49.71)
N	108	258	142	222	296
Project7	-1.502***	-0.663***	-3.497***	-4.563***	150.922**
	(0.05)	(0.04)	(0.04)	(0.33)	(50.25)
N	256	140	220	158	160
Project4	-0.522***	-0.110**	-0.536***	-0.371	-153.117**
	(0.08)	(0.04)	(0.06)	(0.41)	(49.49)
N	296	146	220	204	264
Project8	-0.248***	-0.231***	0.041	0.597	14.080
	(0.04)	(0.05)	(0.07)	(1.49)	(42.31)
N	308	230	344	292	426
Controls, FE	Yes	Yes	Yes	Yes	Yes

Note: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Standard errors are in parentheses and clustered by the day of week. Bandwidths are selected to minimize the mean squared error (MSE) using a triangular kernel. Each element represents the coefficient of the RD regression. Columns (2), (4), and (6) control for variables including wind generation, real-time load, natural gas market price, as well as fixed effects for the day and month. All are limited to MISO Central and North region generation.

2.6.2 The effect on power generation and total cost

I expect that fossil fuel generation is affected negatively after the large transmission capacity changes. On the other hand, the renewable generation will increase as the transmission capacity increases, but it should be a long-run effect. The window of estimation shows only one-year time span.

Table 2.7 shows the change of power generation by fuel type in MISO territory before and after each transmission project. NG CC indicates Combined-cycle natural gas generation, and NG CT indicates all other types of natural gas generation including combustion turbines. The demand of electricity(load) and natural gas spot price, and the day of week, month and year

Table 2.6: Local Linear Regression of the Average Cost: Peak Hour

	Dependent variable: the average cost($/MWh)			
Project Number	(1)	(2)	(3)	(4)
Project 10	0.221	-0.495***	-0.511***	-0.494***
	(0.21)	(0.13)	(0.08)	(0.09)
N	226	270	270	238
Project 6	-0.049	-0.200	-0.779***	-0.767***
	(0.23)	(0.16)	(0.16)	(0.16)
N	268	324	254	244
Project 7	0.394**	-0.049	-1.253***	-1.409***
	(0.12)	(0.07)	(0.07)	(0.08)
N	192	214	266	264
Project 4	-0.384*	0.144*	-0.361***	-0.356***
	(0.19)	(0.07)	(0.09)	(0.10)
N	144	196	266	274
Project 8	-0.453***	-0.003	-0.203*	-0.229***
	(0.11)	(0.18)	(0.09)	(0.07)
N	156	112	110	116
Controls		x	x	x
Fixed effects			x	x
Event dummies				x

Note: * p < 0.05, ** p < 0.01, *** p < 0.001. Standard errors are in parentheses and clustered by the day of week. Bandwidths are selected to minimize the mean squared error (MSE) using a triangular kernel. Each element represents the coefficient of the RD regression. Column (2) includes wind generation, daily load, and natural gas spot price as control variables. Column (3) includes fixed effects of the day of week and month. Column (4) includes a event dummies. All are limited to MISO Central and North region generation.

fixed effects are controlled at all estimations.

Column (1) shows the changes in coal generation following the in-service dates of various transmission projects. The empirical results do not show consistent impacts on power generation across projects. Some projects display significant coal generation increases (Projects 10 and 7), while others show decreases (Projects 6 and 8).

However, the result of total fossil fuel generation reveals (Column (1)) the coal generation increases for Projects 10 and 7 relate to the increase in the total fossil fuel generation. For Project 10 and 6, natural gas generations(particularly NG CT) with higher marginal cost fell

Table 2.7: Local Linear Regression of Power Generation

	Dependent variable: Power generation (GWh)				
Project Number	Total	Coal	NG CC	NG CT	Oil
Project10	-29.363**	57.835***	-33.726***	-19.571***	-0.112**
	(8.95)	(5.06)	(2.81)	(3.66)	(0.04)
N	246	320	294	242	238
Project6	-4.360	13.131	21.150**	-29.099***	-0.319*
	(14.52)	(11.42)	(7.26)	(6.40)	(0.13)
N	234	270	142	148	514
Project7	172.911***	53.716***	6.751	46.850***	-0.031
	(6.67)	(5.09)	(4.31)	(4.72)	(0.02)
N	328	254	222	218	222
Project4	-3.430	-17.390**	-11.705**	20.200**	-0.056*
	(6.41)	(6.13)	(3.99)	(6.25)	(0.03)
N	168	228	206	150	180
Project8	0.962	-30.531**	42.574***	26.492***	0.017
	(15.32)	(10.81)	(3.25)	(4.92)	(0.01)
N	368	368	322	340	382
Controls, FE	Yes	Yes	Yes	Yes	Yes

Note: p<0.05, ** p<0.01, *** p<0.001. Standard errors are in parentheses and clustered by the day of week. Bandwidths are selected to minimize the mean squared error (MSE) using a triangular kernel. Each element represents the coefficient of the RD regression. Estimates control for natural gas spot price, day of week, month and year fixed effects, and minor transmission project dummies. NG CC: Natural gas combined cycle generation, NG CT: Other natural gas types including single cycle and combustion turbine. All are limited to MISO Central and North region generation.

and lead the decrease in total fossil fuel generation instead of coal generation. Otherwise, for project 8 and 4, falls in coal generation lead the overall decrease instead of natural gas generation because of very lower natural gas prices and their lower marginal costs. Across all projects, the highest marginal cost fuel, oil generation, declined, except in Project 8. Overall, the conclusion is that changes in total fossil fuel generation lead to changes in either natural gas or coal generation, depending on the relative marginal costs and merit order of the fuels.

Table 2.8 shows estimated fossil fuel total cost changes by region. Column (1) displays overall MISO territory results after controlling for fixed effects. Projects 10, 6, and 4 reduced Central region total costs, while Project 7 increased costs. The results suggest regional impacts,

Table 2.8: Local Linear Regression of Total Generation Cost

	Dependent variable: total cost($/MWh)		
	MISO	North	Central
Project10	-1,060.532***	198.682*	-1,180.997***
	(291.63)	(94.87)	(204.64)
N	234	308	214
Project6	-855.339	395.663*	-1,229.160***
	(484.20)	(180.16)	(281.45)
N	158	136	204
Project7	279.541*	0.819	761.645***
	(116.93)	(82.27)	(79.60)
N	348	274	328
Project4	-26.145	15.442	-129.878
	(129.97)	(95.63)	(83.18)
N	206	190	254
Project8	339.659	111.615	244.046
	(305.32)	(120.05)	(214.77)
N	358	374	332
Controls, FE	Yes	Yes	Yes

Note: $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Standard errors are in parentheses and clustered by the day of week. Bandwidths are selected to minimize the mean squared error (MSE) using a triangular kernel. Each element represents the coefficient of the RD regression. Estimates control for natural gas spot price, day of week, month and year fixed effects. All are limited to MISO Central and North region generation.

especially Central region-driven effects, exist on total expenses.

Table 2.9 displays the impact on total cost of power generation by fuel type after controlling fixed effects and other variables. The highest marginal cost fuel, oil generation, shows significant total cost reductions. However, combined cycle natural gas and coal changes remain inconsistent. Impacts on natural gas combined cycle and coal generation remain unclear and heterogeneous across projects.

Table 2.9: Local Linear Regression of Total Generation Cost by Fuel Type

	Dependent variable: total cost($/MWh)				
	Total	Coal	NG CC	NG CT	Oil
Project 10	-1,060.5***	1,226.9***	-772.408***	-757.165***	-24.895***
	(291.63)	(83.30)	(80.86)	(143.89)	(7.29)
N	234	370	290	244	246
Project 6	-855.339	108.697	422.513*	-1,132.6***	-53.852
	(484.20)	(205.58)	(169.73)	(323.70)	(27.55)
N	158	286	130	126	442
Project 7	279.541*	489.926***	-541.314***	1,002.8***	-11.050***
	(116.93)	(129.98)	(75.17)	(132.68)	(3.12)
N	348	192	242	252	238
Project 4	-26.145	-267.359*	-218.212**	648.803***	-11.252**
	(129.97)	(116.35)	(72.51)	(164.64)	(4.35)
N	206	164	264	170	192
Project 8	339.659	-700.589***	635.110***	537.437***	-1.357
	(305.32)	(204.67)	(75.17)	(140.38)	(1.76)
N	358	386	308	288	418
Controls, FE	Yes	Yes	Yes	Yes	Yes

Note: p < 0.05, ** p < 0.01, *** p < 0.001. standard error is in parentheses and robust. Autobandwidth. Fixed effects include the day of week, month, and year fixed effect. Hole (dropping the observations in in-service week due to the ambiguous in-service date reports(7 observations) exist in all estimation.

2.6.3 Robustness check

Results in table 2.3) uses around 300 samples on average for estimations. Larger samples guarantee the power of estimation, however, which is the trade-off between the increase of the probability of bias and the increase of power of estimation. Generally, Regression Discontinuity in Time framework uses much wider time windows, for instance, several months or even years. However, such a long time window is difficult to be applied in this research framework. If there are several minor transmission projects completed in MISO territory within a long time window, it is hard to figure out the completion date of each transmission project, so these unobserved completion might affect the outcome unexpectedly. Another reason is that the power generation is highly affected by seasonal demand differences, and weather conditions

Table 2.10: Local Linear Regression of the Average Cost across Different Bandwidths

	Dependent variable: the average cost($/MWh)					
	365 days		180 days		90 days	
Project Number		Control		Control		Control
Project10	0.339***	-0.098	0.262**	-0.114	1.230***	-0.108
	(0.07)	(0.06)	(0.09)	(0.10)	(0.11)	(0.12)
Project6	0.087	-0.347***	-0.165	-0.527***	-1.269**	-0.799*
	(0.14)	(0.09)	(0.20)	(0.14)	(0.48)	(0.37)
Project7	0.208***	-3.507***	0.473***	-2.944***	0.415**	-1.007***
	(0.05)	(0.05)	(0.08)	(0.07)	(0.15)	(0.09)
Project4	0.011	0.025	-0.174**	-0.068	-0.143	0.018
	(0.06)	(0.05)	(0.07)	(0.07)	(0.12)	(0.08)
Project8	-0.688***	0.017	-0.616***	-0.003	-0.137	-0.095
	(0.05)	(0.05)	(0.10)	(0.08)	(0.10)	(0.09)
N	724	723	354	353	174	173
Controls, FE		Yes		Yes		Yes

Note: * p < 0.05, ** p < 0.01, *** p < 0.001. Standard error are in parentheses and clustered by the day of week. Fixed effects control the day of week, and month. Column (2),(4), and (6) have control variables, fixed effect, linear time trend, and interaction term. Each element represents the coefficient of the RD regression. All are limited to MISO Central and North region generation.

are also different by year. It is difficult to assume that the environment and conditions below and above the in-service date are exactly the same and comparable. It affects the external validity of identification, and the increase in time window will increase the probability of bias.

I tested the sensitivity of bandwidth(sample size) on the results. In regression discontinuity analysis, the bandwidth size can impact results. To demonstrate, I examine bandwidth effects on the estimates. Table 2.10 displays local linear regression outputs using one-year, 180-day, and 90-day symmetrical bandwidths around project start dates. Controlling for additional factors, the results consistently show the decrease in the average cost for Projects 6 and 7 across bandwidths and does not show statistically significant changes for other projects.

Another concern of estimation is the linearity of the model. Table A.1 shows the result of estimation with a squared trend and cubic trend. Column (1), and (2) include a squared time trend term, and column (3) and (4) include a cubic time trend term. Column (1) and (3)

have no control variables and no fixed effects. Column (2) and (4) have control variables. The coefficients of squared term and cubic terms are close to zero, which implies that the linear model is proper to explain this model.

2.7 Discussion and Conclusion

In this paper, I analyze the impact of the increase in transmission capacity on power generation, focusing on how fossil fuel generation is affected by wind generation. I provide an estimation of the average treatment effect of the changes in transmission capacity. The main findings show the heterogeneous and unclear effect of the increase in transmission capacity. Some projects show expected negative effects on cost, however, most projects have no significant effect or even positive effect. There are several limitations to explaining these unclear findings.

A remaining question is why the results are heterogeneous and unclear. One answer is the possibility that the increased capacity affects only power generation in peak-hours or local areas. This research assumes major transmission projects would affect the extensive region and use the average over the entire MISO generation cost and price date. However, if the benefits of transmission lines have only a small local impact or only certain hours, for example, the reduction of average generation cost appears only in peak hours or congestion price decreases at a few nodes, the aggregated level data may attenuate the effect and it will not be shown clearly in estimation. To address this, the exact location information is needed, but it is impossible to obtain.

Second answer can be the fractional construction of transmission lines. Such large transmission projects do not start their service at once, usually divide their projects into several parts, and are constructed separately. Some parts start their service earlier and the other parts start several months or years later. For example, Multi-Value Project 5 consists of two parts, Badger-Coulee Project and Cardinal-Hickory Creek project. The former one has been completed and delivered electricity since 2018, the latter one is pending construction and expected

to be completed by 2023. Several similar cases exist in other projects. Thus, even if one project is totally completed and starts its service, it does not mean that its capacity increases dramatically because part of the line might have already worked. Thus, it is difficult to say that transmission in service that occurred in other regions at the same time had an effect.

As addressing these concerns, further study will extend to the effect of congestion level after the transmission expansion. The major goal of new transmission construction is to reduce the congestion of electricity which can potentially increase the price of electricity regardless of generation costs in a deregulated electricity market system.

Transmission investment has been increased and the importance of transmission capacity has been growing as the share of renewable energy increases. Not only for the transmission line owner but also policymakers, the accurate evaluation of the benefit is necessary to decide transmission investment because of its astronomical amount of costs, long-term construction and the difficulties in legal procedures. The results of this paper suggest that the investment on the expansion of the transmission line will be realized by the economic benefits.

Chapter 3

The Impact of Transmission Capacity Expansion on the Wholesale Electricity Market

3.1 Introduction

The wholesale electricity markets in the United States have transitioned from traditional regulated structures to competitive (deregulated) markets in order to improve the efficiency of electricity generation and operations. However, current deregulated electricity market systems still exhibit structural inefficiencies.

One issue is the difference in electricity prices between the day-ahead and real-time markets. If both markets were efficient and perfectly competitive, the price difference should converge to zero because all supply and demand is cleared in the day-ahead market. However, in most deregulated electricity markets, a forward market premium persists, where day-ahead prices remain higher than real-time prices on average(Jha and Wolak (2019), Birge et al. (2018)).

The introduction of virtual bidders, who do not actually sell or buy electricity, was expected to reduce the forward market premium in the wholesale electricity market by arbitraging the

electricity price between the day-ahead and real-time markets. However, some firms have market power in the wholesale electricity market due to barriers to entry such as high transaction costs. With market power, participants have an incentive to buy more demand than their actual needs to increase day-ahead prices and artificially congest nodes, thus increasing their profits from Financial Transmission Rights (FTRs) (Birge et al. (2018)).

However, very large transmission expansions from huge projects can reduce FTR prices (Doshi and Du (2021)), increase the number of market participants, and eventually decrease the market power of certain large firms in the market. Borenstein et al. (2000) analyzed the Cournot duopoly equilibrium quantity between two electricity nodes as transmission capacity varies, in the context of a two-node model. The model explains how small transmission capacity leads to nodal price differences under nodal pricing systems. If line capacity is low, quantity and price diverge between the two regions due to line congestion. Transmission expansions increase capacity between lines, equalizing prices across nodes and reducing the market power of local generators. Although this model does not consider sequential financial markets or bidding strategies, the main conclusions can apply to current electricity market systems.

Transmission line expansion is fundamentally expected to reduce congestion and decrease Financial Transmission Rights (FTR) prices. Doshi and Du (2021) demonstrates that FTR prices in Texas decreased after the CREZ projects.[1]

The goal of this study is to empirically evaluate the effect of large transmission expansions on competitive wholesale electricity markets, particularly in the Midcontinent Independent System Operator(MISO) territory. Large transmission lines connecting wind farms to high demand areas would reduce the market power of local generators near metropolitan areas. Markets would become more competitive, which should decrease forward market premiums, reduce electricity price differences across nodes, and improve overall market efficiency.

This study contributes to the literature evaluating transmission line expansion impacts on

[1]The Competitive Renewable Energy Zones (CREZ) projects constructed 3,500 miles of transmission lines capable of carrying 18,500 MW of electricity across Texas to deliver wind energy from the east to west regions of the Electric Reliability Council of Texas (ERCOT) territory.

wholesale electricity markets within the context of energy economics (Ryan (2021), LaRiviere and Lyu (2022) Joskow and Tirole (2005), **?**). It also builds on previous studies of electricity market efficiencies (Birge et al. (2018), Ito and Reguant (2016), Jha and Wolak (2019), Hopkins (2020), Mercadal (2022)) by examining how large transmission projects influence market power and financial participant behavior in electricity markets.

The Midcontinent Independent System Operator (MISO) provides open access data including hourly day-ahead and real-time electricity prices (LMPs), virtual bidding offers and cleared bids, and monthly Financial Transmission Rights (FTR) auction data. Additionally, the MISO wholesale electricity market structure has not seen significant changes since the integration of the southern region in late 2013. Birge et al. (2018) proved that forward market premiums persisted in the MISO electricity market from 2013-2016. Several major transmission expansion projects have been completed since 2015, concurrently with a significant increase in renewable energy from wind. This study focuses on the 2015-2019 period to demonstrate the major impact of increased transmission capacity on the deregulated wholesale market.

The main findings suggest that wholesale electricity market efficiency improves as transmission capacity increases. The empirical results show that congestion levels and electricity price differences across nodes are reduced by the large transmission line expansion from the MVP project in the MISO territory. These physical changes also affect electricity pricing and other market participant behaviors. Further empirical results demonstrate that although forward market premiums increase, the frequency of extremely high premiums decreases with greater transmission capacity. Virtual bidding prices and volumes decrease, implying reduced incentives for virtual bidding as transmission capacity and congestion fall. Since FTR auctions are also highly correlated with congestion, the findings indicate decreasing FTR auction prices as participants opt for less forward flow and more counterflow positions in response to reduced congestion. Overall, the findings indicate large transmission capacity expansion provides benefits for deregulated electricity market efficiency.

This paper is organized as follows: Section 3.2 describes the structure of the wholesale

electricity market and FTR market. Section **??** provides the data and methodology. Section **??** presents the results of the empirical tests and analysis, and proceeds with further analysis about market efficiency. Section **??** concludes.

3.2 The Structure of Electricity Market

In this section, I introduce the structure of the electricity market, how it operates, and how it is affected by transmission capacity.

The deregulated electricity market is operated by Independent System Operators (ISOs) or Regional Transmission Organizations (RTOs). Most deregulated electricity markets in the United States consist of two sequential markets: the day-ahead market and the real-time market. This two-market structure makes electricity supply more reliable.

The day-ahead (DA) market occurs one day before the operation day in order to allow generators time to prepare for operation. About 95 percent of energy is traded in the day-ahead market, based on the forecasted load for the next day. The remaining energy is traded in the real-time (RT) market, which runs every hour and every five minutes to balance real-time load changes and supply.

In the day-ahead market, generators submit their supply bids for the next day based on their marginal cost of generation and operation. Operators (ISOs or RTOs) decide which generators should be committed for each hour. Operators consider the forecasted load for the next day and compile a list of available generators for next-day dispatch, ordering them from least expensive to most expensive to operate. This allows the market to meet energy demand at the lowest possible price. During periods of high demand, wholesale prices rise accordingly because more high-cost units need to be dispatched to meet the electric load. This is done in advance because some generating units require several hours to start up before actual power generation begins.

The day-ahead markets provide hedging and scheduling activities and forecast real-time

markets. Day-ahead markets provide a level of price certainty and protect against electricity price volatility. Also, the day-ahead markets allow generators to prepare for future operations (Hogan (2014)).

The real-time market operates every 5 minutes on a given day. Market participants buy or sell electricity in the real-time market to balance the differences between actual real-time demand and day-ahead commitments, in order to avoid outages.

3.2.1 The price of electricity

Under a competitive electricity market, the dispatch and pricing mechanism follows nodal or zonal pricing. All U.S. deregulated wholesale electricity markets have followed nodal pricing since 2010. The Locational Marginal Price (LMP) represents the unique value of energy at each location (commercial pricing node) on the electricity network.

LMP consists of three components: the System Marginal Price, the Marginal Congestion Component (MCC), and the Marginal Loss Component (MLC). These three components are calculated in both the day-ahead and real-time markets. The System Marginal Price is the incremental price of energy for an electric system and is decided by the current dispatch. The Marginal Congestion Component (MCC) represents the price of congestion. The MCC can be positive or negative depending on whether electricity flows relieve or worsen congestion, and the level of congestion is represented by the absolute value of the MCC.

If the electricity market is efficient, the LMP of the day-ahead market and real-time market should converge at the same level for each given hour, because the day-ahead market forecasts electricity supply and demand based on the assumption of inelastic electricity demand. However, in most actual electricity markets, prices are not equal between the day-ahead and real-time markets.

3.2.2 Virtual bidder and forward market premium

The price difference between the day-ahead and real-time markets is called the forward market premium (or day-ahead market premium), which represents inefficiency in the electricity market. There are several reasons for the market inefficiency and forward market premium.

The main reason is the existence of market power in the market. In a normal competitive electricity market, market participants have incentives to arbitrage the prices of the day-ahead and real-time markets, which leads the forward premium to converge to zero. However, Ito and Reguant (2016) demonstrates the strategy whereby certain large generators with market power have incentives to commit lower quantities of supply in the day-ahead market to increase day-ahead prices, and then increase their quantity in the real-time market to lower the market price, thereby maximizing their profit.[2]

If only physical generators and retailers existed in the wholesale market, some of them would exercise their market power. To prevent this situation, all ISOs in the United States allow virtual bidders (or financial traders) to participate in the electricity market. Virtual bidders do not have physical assets, so they do not purchase or sell physical electricity. Instead, virtual bidders buy some amount of megawatts (MW) virtually in the day-ahead market at demand bidding prices and sell the exact amount of MW in the real-time market at supply bidding prices in order to earn arbitrage profits. These activities can reduce the forward market premium.

Ledgerwood and Pfeifenberger (2013) illustrates how virtual bidders arbitrage and reduce the forward market premium in the electricity market. If there is a positive forward premium ($P_{DA} - P_{RT} > 0$) in the market, virtual bidders sell a certain amount of MW at the day-ahead market price and buy the same amount of MW at the real-time market price to gain positive profits equal to MW $\times (P_{RT} - P_{DA})$. In the same way, other virtual bidders would bid at a lower price in the day-ahead market and buy at a higher price in the real-time market. Market participants have an incentive to continue this activity until the difference between the day-

[2]With this strategy, generators can sell more in the day-ahead market at better prices. This strategy makes the real-time market price lower than the day-ahead market price.

ahead and real-time price converges to zero[3]. As the number of market participants increases, the market becomes more competitive, and forward market premiums decreases.

However, the introduction of financial traders does not completely remove the forward market premium. One reason is the high transaction costs to participate in the market. For example, MISO requires demonstrating total assets of $5,000,000 for energy market trading participation. [4]

When the system operator reduces implicit and explicit transaction costs, financial trading becomes more effective. Jha and Wolak (2019) shows that reducing transaction costs increased power planning quality and reduced emissions and generation costs in the California electricity market (CAISO). However, forward market premiums still exist. Birge et al. (2018) shows the forward market premium persisted in the MISO market even after reducing the cost of virtual transactions through the Revenue Sufficiency Guarantee (RSG) charges in 2011.

3.2.3 Financial transmission rights

Financial Transmission Rights (FTRs) offer the opportunity to hedge price risk from congestion by providing their owners with financial revenue that can offset congestion charges (Hogan (1992)). FTRs are purely financial instruments, so they do not represent physical rights to transmission lines or energy delivery. Only market participants can hold FTRs.

The system operator (for example, MISO) sells the rights to some portion of each transmission line's capacity to market participants through auctions held monthly, quarterly, or annually. Each auction sells a different part of the lines' capacity. A second round exists, but most transactions occur in the first round. In the day-ahead market, participants bid congestion prices for the transmission lines. The FTR payout is calculated by subtracting the source node's (e.g. generation node) FTR price from the sink node's (e.g. demand node or load zone

[3] or to the minimum level of transaction costs, if they exist.

[4] To participate in the MISO energy market, a company must demonstrate total assets of $5,000,000 or a tangible net worth of $500,000. Companies unable to provide audited financial statements at the required level must post a $50,000 financial security, of which $25,000 is restricted. (Source: MISO Minimum Participation Requirements)

node) FTR price.

For example, if the congestion is from A(a source node) to B(a sink node), the nodal price at B is higher than the nodal price at A. The Forward FTR holder is paid from ISO by the difference between the congestion price at B and the congestion price at A. The Counterflow FTR holder must pay MISO the difference between the congestion price at B and the congestion price at A.

3.2.4 Transmission expansion and market power

In deregulated electricity markets, generators have an incentive to create forward market premiums when they have market power (Ito and Reguant (2016), Jha and Wolak (2019), Birge et al. (2018)). Joskow and Tirole (2000) point out that generators holding FTRs have an incentive to exercise their market power by reducing generation, since profits from Financial Transmission Rights (FTRs) can exceed those from generation. Borenstein et al. (2008) also finds that forward market premiums persist when trading is limited to physical participants because of generator and consumer monopsony power. Ito and Reguant (2016) shows firms have an incentive to not arbitrage day-ahead and real-time prices in order to restrict entry and maintain imperfect competition.

Much literature points out that the main source of market power in electricity markets is barriers to entry and high transaction costs. Jha and Wolak (2019) and Birge et al. (2018) show empirical results that lowering transaction costs can reduce market power and decrease forward market premiums in the CAISO and MISO markets.

Another contributor to market power in electricity markets is limited transmission capacity. Congestion limits electricity delivery to other regions, providing opportunities for local generators and participants to exercise market power in the day-ahead market. Transmission expansion can distribute electricity over wider areas, reducing the market power of local generators.

Borenstein et al. (2000) demonstrates that limited transmission capacity incentives firms to exercise market power, increasing congestion and price inefficiencies to maximize profits.

Firms are motivated to restrict output to congest transmission into their areas of dominance. To remove this inefficiency, transmission capacity between two markets should be large enough for firms to compete over the expanded area. Ryan (2021) finds reducing transmission constraints can increase competitiveness in the Indian electricity market. Mercadal (2022) shows congestion allows firms more market power in local nodal markets, mainly in high wind areas with limited transmission.

3.2.5 Electricity market example

In this section, let me explain the structure of the deregulated electricity market, the role of virtual bidding and the financial transmission rights(FTR)s.

The example of pricing and congestion

I start from the basic example of two-node model from Joskow and Tirole (2000) Under the perfectly competitive market, there are two electricity nodes connected by one transmission line. The Node West generates power with lower marginal cost(20 dollars per mega-watt hour), and Node East generates electricity with higher marginal cost.(30 dollars per mega-watt hour). The demand of electricity(Load) is assumed perfectly inelastic and is equal to 100MW at each node. The capacity of power generation at Node West is 200 mega-watt. Both regions are operated by Independent System Operator(ISO).

When the transmission capacity is 80MW, the prices of electricity at two nodes are illustrated in the Figure 3.1. The Locational Marginal Price(LMP) at the West is 20 $/MWh, which is same as the marginal cost of electricity because all the demands can be supplied by the lowest marginal cost.[5] On the other hand, the LMP at the East is $30/MWh. Since the supply of electricity follows the merit of order, the West exports the electricity by 80MW at $20/MWh, and the remaining 20MW is supplied by the power plant at the East with the marginal cost of $30/MWh. In this case, due to the lack of transmission capacity, Node west can export only 80MW to Node East even

[5] I assume that marginal congestion cost and marginal loss component are zero for simplicity.

though their generation capacity is large enough.(The situation of constrained.) It is also lead the difference of the electricity price between two nodes and create congestion. The marginal congestion cost between two nodes are calculated by \$30/MWh-\$20/MWh=\$10/Mwh.

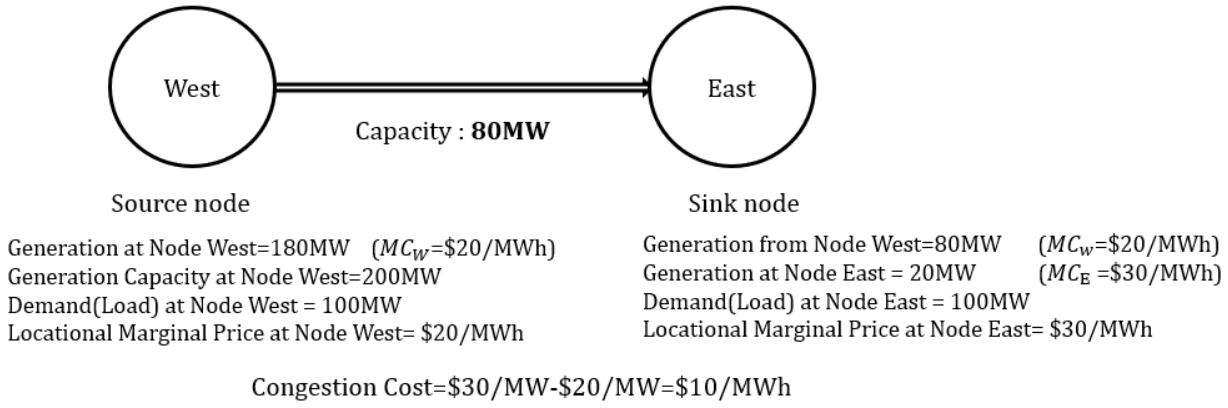


Figure 3.1: The Case of Constrained

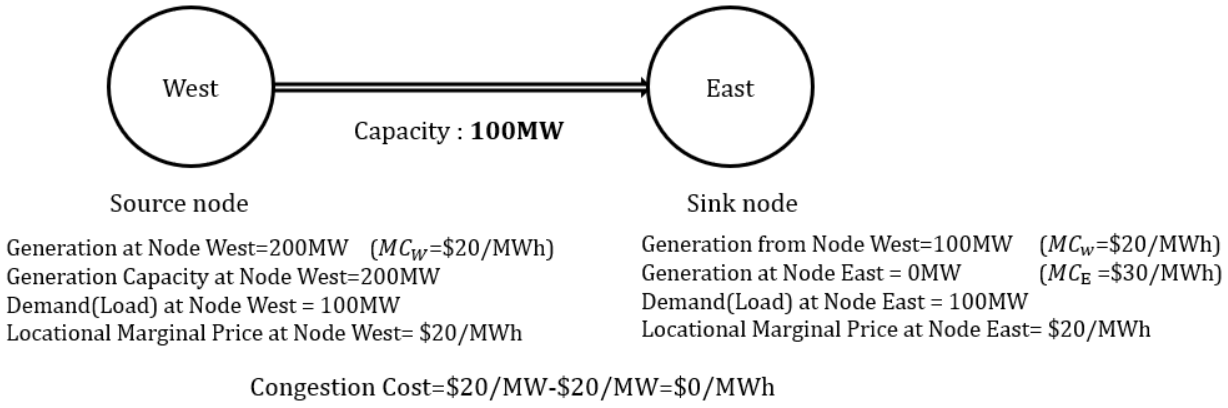


Figure 3.2: The Case of Unconstrained

Assuming that the transmission capacity is expanded by 100MW now. The West and the East still consume 100MW, but power generation become different. The West generates electricity by 200MW and export 100MW to the East by $20/MWh, and the East do not need to produce power at $30/MWh. Hence, the price of electricity at the East becomes $20/MWh, which is equal to the price of power at the West. The price difference across nodes disappears and marginal congestion cost is zero.

The example of virtual bidding

What happens if the virtual bidding is allowed in this market? and how virtual bidding can reduce the forward market premium?

Assume that there is a forward market premium in the previous example. (In the previous example, I do not consider the day-ahead market and real-time market.) The day-ahead LMP is $30/MWh(The case of Constrained), but virtual bidder expect that the real-market LMP will be realized at $35/MWh. Virtual bidders have incentive to buy electricity at sink node(Node East) at the day-ahead market to earn profit from arbitrage. If virtual bidders buy the demand bid by 20MW at Node East, the demand of electricity at day-ahead market increases by 120MW and the price of LMP at Node East at day-ahead market is cleared to $33/MWh to meet the increased power demand. At real-time market, virtual bidders should sell their power at sink node because they cannot buy physical electricity. The price of electricity at real-time price is realized at $35/MWh as expected. The virtual bidder buy the electricity by 20MW at $33/MWh and sell at $35MWh, so their profit is 20MW $\times$ ($35/MWh-$33/MWh) = $40 from this bidding. The virtual bidders repeat this virtual trade until the day-ahead market price converges to the real-time(forward) market price. On other words, as the size of forward market premium larger, the incentive to participate the virtual bidding also increases.[6]

Virtual bidding is not affected directly by the change of transmission capacity, however, the price of electricity is affected because the marginal congestion component is the part of

[6]This is the example of demand bid. The supply bid works in the opposite way. The supply bidder sell their power at the day-ahead market and buy their bid at the real-time market.

price of electricity. If the line is not congested, the real-market LMP can be $33/MWh or lower, which would affect the behavior of virtual bidders.

The example of FTR market

I give the example when the financial Transmission Rights(FTRs) is considered. Assume that one virtual bidder holds forward flow FTR of the transmission line between Node West and Node East by 10MW. The total capacity of transmission line is 80MW.(The case of constrained.) Assume the first example that LMP of Node West is $20/MWh and LMP of Node East is $30/MWh at day-ahead market, so the congestion cost of this line is $10/MWh. The revenue of forward flow FTR holder is 10MW × ($30/MWh-$20/MWh) = $100. If a holder buy this FTR at $50, the profit of FTR holder is $100-$50=$50. [7] [8]

When the capacity of transmission line increases, the price and revenue of FTR are changed. Although market participants do not hold the FTR of the new line, other lines become less congested due to the new line. Assume the example of unconstrained transmission line between Node West and Node East. Since the larger capacity of transmission line(100MW), the congestion cost is zero and the revenue of forward flow FTR holder become zero. This holder is not willing to buy FTR for this line again next month, which lead to the decrease in the price of FTR at the auction next month. Or, the holder will buy the counterflow FTR as expecting the revenue from reverse congestion.

3.3 Data and Methodology

This section introduces the data utilized in this study. Firstly, the measurement of transmission line capacity is described. Secondly, the specific bid and FTR price data of the wholesale

[7]The system operator receives this revenue from power generator at Node West and load node at Node East and pays to forward flow FTR holder.

[8]The counterflow FTR holder get paid from system operators when the line is not congested. For example, the LMP at Node West is $30/MWh and the LMP at Node East is $20/MWh, the congestion cost of the line from Node West to Node East is -$10/MWh, and the revenue of the counterflow FTR holder is the amount of FTR holds×$10.

electricity market are introduced for this research along with their basic statistics to understand their properties. Finally, the calculation of the forward market premium data is illustrated.

3.3.1 Transmission projects and capacity

MISO is the second largest electricity market in the United States, including 7 major hubs and 15 states in the Midcontinent region. MISO also operates sequential day-ahead and real-time electricity markets along with FTR auctions.

Although 17 large transmission projects, known as the Multi-Value Portfolio (MVP), have been completed since 2011, most were finished and entered service between 2017 and 2019. Additionally, the MISO wholesale electricity market has not seen significant structural changes since merging with the southern region in 2013. From 2015 to 2019, natural gas prices remained stable, coal plants gradually retired, electricity demand continued, and wind generation share increased incrementally.

Since 2013, MISO has completed several transmission expansion projects, including the Multi-Value Project (MVP) portfolio. The MVP consists of 17 projects, but I exclude Projects 5 and 11 which were completed in 2020, outside the sample period. The other 15 projects were completed between 2013 and 2019. As shown in Table 3.1, most of the new transmission lines were constructed between 2015 and 2019.

To approximate changes in transmission line capacity, I set line capacity at 0 before MVP projects began. As each project was completed, I accumulated the product of line length and voltage capacity (kV) to measure the increase. Figure 3.3 illustrates the change in line capacity as MVP projects were completed, showing the large increase under MISO's North and Central regions.

3.3.2 Electricity market data

MISO provides public data on day-ahead cleared bids, including virtual and physical bids, FTR monthly and annual bid offers, as well as historical day-ahead and real-time LMPs for each

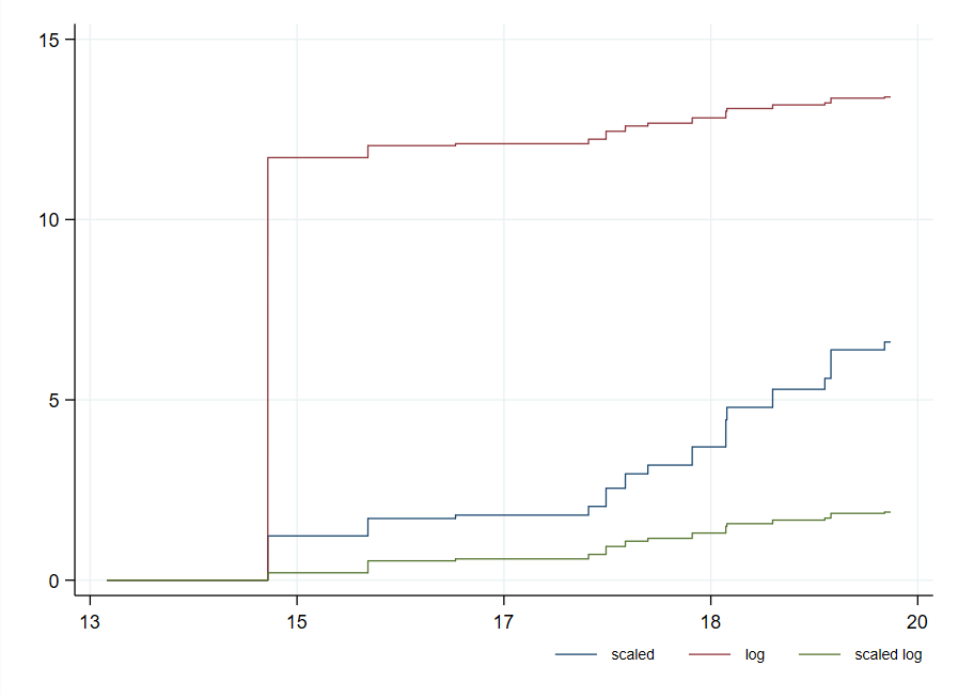


Figure 3.3: The Trend of Transmission Line Capacity

Note: The line capacity is assumed to be 0 before the start of the MVP projects. For each completed project, the increase in transmission capacity is calculated by adding the product of the new line's length and its voltage capacity in kV.

node.

The virtual bidding data includes cleared bid prices and energy amounts (MW) hourly for each market participant at each node. I use the price and amount of cleared bids in megawatts (MW). Bidder names and participants are masked, so individual bid information cannot be matched to FTR owners. Bidding node locations are also marked with only regional information available (North, Central, South). Thus, I only use cleared bids from the North and Central regions.

Virtual bids have two types: Decrease and Increase. Decrease bids, also called Demand bids, are treated as positive virtual bids. Increase bids, also called Supply bids, are treated as negative virtual bids. In MISO, virtual bidding is only allowed for demand bids, so Increase bids represent selling while Decrease bids represent buying.[10]

Bid volumes represent the amount of cleared virtual bidding in megawatts (MW) at each

[10]Most RTOs use the terms DEC and INC bids instead of Demand and Supply bids.

Table 3.1: MVP Portfolio List

Project No.	Project Name	In Service Date	Max kV	Line (Miles)
MVP 15	Pleasant Prairie to Zion Energy Center	5/31/2013	345	5.3
MVP 2	Brookings, SD to SE Twin Cities	3/26/2015	345	356
MVP 13	Michigan Thumb Wind Zone	12/31/2015	345	140
MVP 17	Sidney to Rising	9/1/2016	345	27
MVP 1	Big Stone South to Brookings	9/8/2017	345	70
MVP 10	Pawnee to Mt. Zion to Sugar Creek	10/27/2017	345	146
MVP 9	Palmyra Tap to Pawnee	12/20/2017	345	116
MVP 16	Oak Grove to Fargo (Spoon River Project)	2/21/2018	345	70
MVP 14	Reynolds to Greentown	6/25/2018	765	66
MVP 3	Lakefield Jct. to Webster	9/27/2018	345	218
MVP 12	Reynolds to Burr Oak to Hiple	9/30/2018	345	100
MVP 6	Ellendale to Big Stone	2/5/2019	345	145
MVP 7	Ottumwa to Adair	7/1/2019	345	88
MVP 4	Winco to Hazleton	7/18/2019	345	229
MVP 8	Adair to Palmyra Tap	12/15/2019	345	63
MVP 11	Pawnee to Sugar Creek	12/14/2020	345	146
MVP 5	Dubuque and North LaCrosse to Cardinal	Exp in 2024	345	157[9]

Source: MISO MTEP Report

hour and node. The statistics are shown in the third panel of Table 3.2.

Wholesale electricity market price data is available hourly, including Locational Marginal Prices (LMPs) and Marginal Congestion Costs (MCCs), for both Day-Ahead (DA) and Real-Time (RT) markets. RT LMPs are provided as the hourly average of 5-minute LMPs. The price data includes node information but cannot be matched to bidding data. Nodes are divided into four types: Generation, Load Zone, Hub, and Interface. I use Generation, Load Zone, and Hub nodes located in the North and Central regions of the MISO territory for this study.

In the first panel of Table 3.2, the average Day-Ahead Locational Marginal Price (LMP) is slightly higher than the Real-Time LMP, and the standard deviation of Day-Ahead LMP across hubs is lower than for Real-Time LMP. This is expected since Real-Time LMPs generally reflect more volatile conditions, as also shown in the difference in statistics between Day-Ahead and Real-Time Marginal Congestion Component(MCC).

MISO facilitates annual and Multi-Period Monthly Auctions (MPMA) for Financial Transmission Rights (FTRs). The annual FTR auction is conducted before each planning year in three

rounds. Some allocations from the first or second rounds may be re-offered in later rounds. 25% of the total feasible FTR capability is traded in the annual auction. Multi-Period Monthly Auctions trade residual capability after the annual FTR auctions, with seasonal (three month) contract terms and single round bidding. Monthly FTR auctions have one month contract terms and are also single round. MISO allows market participants to re-sell FTRs they hold in the second or third auction rounds.

In this study, I only use the amount and price of monthly FTR auctions to observe immediate changes in FTR market and market participants following the in-service date of new transmission lines. It is difficult to directly reflect short-term changes in annual and seasonal auctions. I exclude FTR selling offers from the monthly FTR auctions.

MISO has two types of FTR pricing: forward flow and counterflow, for price hedging by transmission customers. For example, the owner of a forward flow FTR is paid by MISO when congestion occurs on the line, based on the price difference between the two nodes. On the other hand, counterflow FTR owners must pay MISO when their line is congested, but get paid when uncongested, also based on the nodal price difference. Forward FTR prices are positive, while counterflow FTR prices are negative.

The sample period starts on January 1st, 2015 and ends December 31st, 2019. Control variables include wind generation and hourly load to capture changes in the share of wind generation and electricity demand over time. [11]

3.3.3 Forward market premium

Table 3.2 includes the statistics of the forward market premium. The forward market premium is calculated as the simple average day-ahead LMP minus the simple average real-time LMP across load zone nodes. On average, the forward market premium is positive during sample periods.

[11] Solar generation was not included in MISO's generation mix until 2021.

Table 3.2: Descriptive Statistics

Variables	Statistics				
	N	Mean	Std. Dev.	Min	Max
Real-Time LMP	15,057,408	26.218	18.309	-1,082.920	1,999.130
Real-Time MCC	15,057,408	-0.672	11.502	-1,102.920	1,969.450
Day-Ahead LMP	15,057,408	26.689	10.368	-499.000	2,088.920
Day-Ahead MCC	15,049,104	-0.607	4.717	-533.530	2,070.570
Forward Market Premium	43,824	0.473	11.828	-542.237	113.684
Demand Bid Volume	73,796,366	1.932	10.127	0	2,909.400
Supply Bid Volume	109,399,446	1.366	6.212	0	2,088.920
Demand Bid Price	73,796,366	27.363	13.575	-601.22	2,088.920
Supply Bid Price	109,399,446	26.508	12.016	-601.22	2,088.920
Forward FTR Volume(mw)	811,802	3.620	5.867	0.100	361.300
Counterflow FTR Volume(mw)	1,025,342	4.661	8.271	0.100	799.500
Forward flow FTR Price	811,802	-264.108	466.739	-19,495.000	-0.010
Counterflow FTR Price	1,025,342	241.390	394.956	0.000	13,677.490
Transmission line Capacity(Scaled)	43,824	2.829	1.793	0.000	6.604

Note: DA = Day-Ahead, RT = Real-Time, LMP = Locational Marginal Price of electricity, $/MW, MCC = Marginal Congestion Cost, $/MW, Demand Bid = Positive virtual bid, Supply Bid = Negative virtual bid. Average Forward Market Premium is calculated as the simple average DA LMP minus simple average RT LMP across load zones or generation nodes for each hour. Nodes and FTRs are limited to the MISO North and Central regions.
Sample is hourly from 2015/01/01 01:00 to 2019/12/31 24:00. On-Peak and Off-Peak hours follow MISO's definitions.
Sources: MISO Market Data, MISO Multi-Period Monthly Auction FTR,

3.4 Empirical Results

In this section, I present the empirical results estimating the effect of transmission line expansion on the electricity market. First, I present results on congestion levels and price differences across regions to evaluate changes in the physical efficiency of the wholesale market. Next, I show the effects on the forward market premium, virtual bidding prices and volumes, and monthly FTR auctions, examining how market participant behavior changes with transmission expansion.

3.4.1 Empirical strategy

Transmission expansion can affect electricity markets in two key ways. First, by lowering the market power of local generators, which is expected to increase market efficiency and

decrease forward market premiums. Second, by increasing transmission capacity, which lowers congestion and reduces artificial congestion from speculative market participants. This would reduce MCC and LMP at each node, also expected to lower forward market premiums.

I estimate the effect of transmission expansion on the electricity market using a linear regression model.

$$y_t = \beta_1 + \beta_2 \text{Line Capacity}_t + \theta X_t + \gamma_t + \varepsilon_t \tag{3.1}$$

I use the scaled down line capacity by 100,000 as the independent variable (Figure 3.3). X represents control variables including actual day-ahead load (GW) and day-ahead wind generation (GW). γ represents fixed effects for time t to capture time-varying changes in the dependent variables, including year, month, day of week, and hour fixed effects. The dependent variables are congestion levels, bid prices and volumes, and FTR prices and volumes.

Theoretically, increasing transmission capacity is expected to reduce congestion in the electricity market. Larger line capacity should lower congestion levels. Congestion levels directly affect electricity prices (LMPs) since LMPs contain marginal congestion cost factors. Additionally, the congestion levels of transmission lines are highly correlated with the amount and price of Financial Transmission Rights (FTRs).

3.4.2 Congestion

To estimate the impact on congestion levels from new transmission lines, I use the absolute value of MCC to measure congestion regardless of electricity flow direction. A higher absolute MCC indicates higher congestion. The MCC sign, positive or negative, depends on the flow direction. Since examining all MISO nodes is infeasible, I analyze changes at individual hub nodes and use the simple average across load zone and generation nodes.

Table 3.3 shows the estimated effect of transmission expansion on day-ahead and real-time congestion levels by node type. Day-ahead congestion levels decrease significantly as

Table 3.3: OLS Regressions - Congestion Level All Nodes

	Dependent variable : Absolute value of MCC					
	Generation		Load zone		Hubs	
	Day-Ahead	Real-Time	Day-Ahead	Real-Time	Day-Ahead	Real-Time
	(1)	(2)	(3)	(4)	(5)	(6)
Trans capacity(Scaled.)	-0.347***	-0.260**	-0.435***	-0.299**	-1.853***	-1.782***
	(0.03)	(0.09)	(0.03)	(0.09)	(0.12)	(0.38)
Wind Gen(Gw)	0.306***	0.427***	0.412***	0.535***	0.993***	1.335***
	(0.01)	(0.01)	(0.01)	(0.01)	(0.02)	(0.05)
Load (Gw)	0.103***	0.193***	0.130***	0.230***	0.426***	0.844***
	(0.00)	(0.01)	(0.00)	(0.01)	(0.01)	(0.04)
Constant	-4.583***	-10.471***	-5.874***	-12.385***	-18.549***	-43.307***
	(0.15)	(0.45)	(0.19)	(0.54)	(0.72)	(2.26)
Fixed effects	Yes	Yes	Yes	Yes	Yes	Yes
R-squared	0.217	0.099	0.274	0.114	0.185	0.080
Observation	43,824	43,824	43,824	43,824	43,824	43,824

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Variables are simple average of absolute MCC across node types. Nodes limited to MISO North and Central regions. Hub nodes: ILLINOIS, INDIANA, MINN, MICHIGAN. Fixed effects: year, month, day of week, hour. Sample period: 01/01/2015 - 12/31/2019.

transmission capacity increases by 100,000 - by 0.435 on average across load zone nodes, 0.347 across generation nodes, and 0.853 across hub nodes. The decrease is larger in the day-ahead versus real-time market. Congestion across generation nodes decreases by 0.347 overall as line capacity increases.

3.4.3 Price differences across regions

According to Joskow and Tirole (2005), increasing transmission line capacity can reduce electricity price differences across nodes by lowering congestion. In 3.1, I include the standard deviation of LMP across nodes as the dependent variable to estimate changes in price discrepancies across regions. The results are shown in Tables 3.4 and 3.5.

The results are consistent with the previous congestion estimates and theory. In Table 3.4, electricity price differences across nodes decrease significantly, from 0.493 to 0.815. The tendency to decrease is greater in the real-time market, indicating transmission expansion effectively reduces real-time congestion. Table 3.4 shows the difference in congestion across nodes, measured by the standard deviation of marginal congestion components. Similar to

LMP, congestion differences across nodes also decrease.

Combining the regression results of the average congestion level and its standard deviation, the physical efficiency of wholesale electricity market seem to be improved as the large transmission line capacity. These empirical results are consistent with the hypothesis from theoretical analysis as well.

Combining the regression results on the average congestion level and its standard deviation, the physical efficiency of the wholesale electricity market appears to improve with the large increase in transmission line capacity. These empirical results are consistent with the hypotheses from theoretical analysis as well.

Table 3.4: OLS Regressions - Price Difference of LMP

Dependent variable : Standard Deviation of LMP Across Nodes				
	Generation Node		Load zone Node	
	Day-Ahead	Real-Time	Day-Ahead	Real-Time
	(1)	(2)	(3)	(4)
Trans capacity(Scaled.)	-0.614***	-0.767***	-0.493***	-0.815***
	(0.04)	(0.14)	(0.03)	(0.11)
Load (Gw)	0.178***	0.403***	0.150***	0.309***
	(0.00)	(0.01)	(0.00)	(0.01)
Wind Gen(Gw)	0.567***	0.895***	0.417***	0.630***
	(0.01)	(0.02)	(0.00)	(0.02)
Constant	-7.196***	-20.360***	-6.484***	-16.182***
	(0.24)	(0.78)	(0.21)	(0.63)
Fixed effects	Yes	Yes	Yes	Yes
R-squared	0.403	0.138	0.332	0.104
Observation	43,824	43,824	43,824	43,824

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Variables are the standard deviation of LMP across nodes. Nodes are limited to MISO North and Central regions. Fixed effects include year, month, day of week, hour. Sample period: 01/01/2015 - 12/31/2019.

Table 3.5: OLS Regressions - Price Difference of MCC

Dependent variable : Standard Deviation of Congestion Across Nodes				
	Generation Node		Load zone Node	
	Day-Ahead	Real-Time	Day-Ahead	Real-Time
	(1)	(2)	(3)	(4)
Trans capacity(Scaled.)	-0.635***	-0.792***	-0.466***	-0.804***
	(0.04)	(0.14)	(0.03)	(0.11)
Load (Gw)	0.162***	0.389***	0.131***	0.290***
	(0.00)	(0.01)	(0.00)	(0.01)
Wind Gen(Gw)	0.475***	0.870***	0.361***	0.624***
	(0.01)	(0.02)	(0.00)	(0.02)
Constant	-6.462***	-20.150***	-5.811***	-15.875***
	(0.22)	(0.75)	(0.19)	(0.59)
Fixed effects	Yes	Yes	Yes	Yes
R-squared	0.361	0.135	0.290	0.100
Observation	43,800	43,824	43,800	43,824

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001

Table 3.6: Descriptive Statistics of Standard Deviation of LMP and MCC

Variables	Statistics				
Standard Deviation of	N	Mean	Std. Dev.	Min	Max
Day-Ahead LMP	43,824	3.951	2.874	0.349	112.317
Real-Time LMP	43,824	5.729	9.447	0.004	455.803
Day-Ahead MCC	43,800	3.090	2.715	0.000	112.622
Real-Time MCC	43,824	4.880	9.273	0.000	454.455

Note: The standard deviation of variables across nodes at each hour.

3.4.4 Forward market premium

Convergence of electricity prices between the day-ahead and real-time markets is also important for measuring market efficiency. Generators, retailers, and system operators can reduce costs by forecasting day-ahead prices closer to real-time prices. The forward market premium (FMP) is defined as:

$$FMP = LMP_{DA} - LMP_{RT}$$

Where:

LMP_{DA} = Day-ahead locational marginal price

Table 3.7: OLS Regressions - Forward Market Premium Variables

Dependent variable: Simple Averaged Forward Market Premium					
	(1)	(2)	(3)	(4)	(5)
Line capacity(Scaled)	0.441**	0.029***	-0.013***	-0.282***	1.303**
	(0.16)	(0.01)	(0.00)	(0.07)	(0.45)
Wind Gen(Gw)	0.038	0.001	-0.004***	-0.050***	0.173***
	(0.02)	(0.00)	(0.00)	(0.01)	(0.05)
Load (Gw)	-0.053***	-0.002***	0.008***	0.276***	-0.763***
	(0.02)	(0.00)	(0.00)	(0.01)	(0.05)
Constant	3.472***	0.727***	-0.367***	-11.908***	36.598***
	(0.02)	(0.01)	(0.54)	(2.52)	(0.82)
Fixed effects	Yes	Yes	Yes	Yes	Yes
R-squared	0.003	0.038	0.092	0.253	0.111
Observation	43,824	43,824	43,824	29,421	14,403

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Column (1) shows the regression on the simple averaged forward market premium across loadzone nodes. Column (2) shows the results from a linear probability regression estimating the probability of a positive forward market premium. The dependent variable is a binary dummy equal to 1 if the premium is positive, and 0 otherwise. Column (3) shows results from a linear probability regression estimating the probability of the real-time price being above 2 standard deviations. Column (4) shows the change in forward market premium when its value is positive. Column (5) shows the change in forward market premium when its value is negative.

LMP_{RT} = Real-time locational marginal price

However, unlike congestion levels and regional price differences, the forward market premium persists in the market. In the first column of Table B.3, the average forward market premium during the sample period is still positive, 0.441 statistically significantly, indicating day-ahead LMPs exceed real-time LMPs.

Table B.3 shows additional estimation to analyze the positive forward market premium. Column (2) shows a linear probability regression estimating the probability of an extreme premium above 40 or below -40. The -0.004 coefficient indicates extreme premiums decrease with transmission expansion. Column (3) shows a similar regression estimating the probability of extreme real-time LMPs above 1.5 standard deviations, also with a negative coefficient, -0.013, implying fewer extremes with more transmission. Column (4) presents an OLS regression of positive premium values, while Column (5) shows negative values. The coefficients demon-

Table 3.8: OLS Regressions - Bid Variables (All Nodes)

	Bid Price		Bid Volume		The number of bids	
	Demand	Supply	Demand	Supply	Demand	Supply
	(1)	(2)	(3)	(4)	(5)	(6)
Trans. Line capacity	-1.097***	-1.085***	-151.893***	-83.442***	5.596	107.261***
	(0.08)	(0.08)	(10.09)	(11.00)	(4.24)	(7.37)
Wind Gen(Gw)	-0.827***	-0.950***	123.592***	132.363***	-23.873***	35.477***
	(0.01)	(0.01)	(1.39)	(1.53)	(0.56)	(0.96)
Load (Gw)	0.756***	0.731***	20.781***	24.854***	11.957***	1.039*
	(0.02)	(0.02)	(0.65)	(0.65)	(0.30)	(0.42)
Constant	-13.107***	-12.084***	402.355***	192.003***	-3.026	1463.618***
	(0.86)	(0.84)	(39.07)	(41.11)	(18.06)	(27.80)
Fixed effects	Yes	Yes	Yes	Yes	Yes	Yes
R-squared	0.632	0.654	0.547	0.491	0.654	0.694
Observation	43,824	43,824	43,824	43,824	43,824	43,824

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Virtual bidding only. Bid volume is average of cleared MW across nodes for each hour. Bid price is average cleared price across nodes for each hour. Number of bids is average cleared bids across nodes for each hour. Sample period: 01/01/2015 - 12/31/2019.

strate positive premiums decrease by 0.282 but negatives increase by 1.303 with transmission expansion.

Positive premiums signify higher day-ahead prices, potentially indicating virtual bidder manipulation and failure to arbitrage. However, negative premiums caused by unexpected events like weather are inevitable. The results show events with positive premiums decrease as transmission expands, indicating improved market efficiency despite premium persistence.

3.4.5 Virtual bidders

In this section, I analyze financial electricity market aspects potentially affected by virtual bidding and Financial Transmission Rights(FTR) auctions.

The estimation results in Table 3.8 show clearer changes. Bid prices decrease with transmission capacity expansion, by -1.097 for demand bids (Column 1) and -0.085 for supply bids (Column 2). The effect on bid volumes is more pronounced. Total hourly demand bid volumes decrease by 151.893 (Column 3), a greater absolute reduction than the decrease in total supply bid volumes (Column 4).

This implies virtual market participants reduce both demand and supply bid offers as transmission capacity rises, with reduced congestion making the market seemingly less profitable. Column (6) in Table 3.8 shows supply bid numbers clearly increase, but the change in demand bid numbers is positive but small and insignificant. This suggests some bidders shift strategy from demand to supply bidding.

To analyze participant motivations and incentives, I examine Financial Transmission Rights (FTR) market changes in the next section.

3.4.6 FTR market

Figure B.5 shows that the volumes of both forward and counterflow FTRs increased (Panel b). This is expected, as more transmission capacity leads to greater FTR availability and volumes. Forward flow FTR prices decreased while counterflow prices increased gradually over time (Panel a). It appears both are converging to certain levels. This aligns with **?**, who found new ERCOT transmission lines decreased monthly FTR prices.

Table 3.9 provides clearer evidence on changes in monthly auctioned FTR variables as transmission capacity rises.[12] I use the absolute value of counterflow FTR prices since they are negative. Table 3.9 shows counterflow FTR prices increase by 21.358 while forward flow prices decrease 2.518 with more transmission. This aligns with reduced congestion. As lines become less congested, forward flow FTRs are less profitable while counterflow FTRs become more so. With less congestion, market participants likely have greater incentive to buy counterflow rather than forward flow FTRs. Consistent with this, counterflow FTR volumes increase 0.11 while forward flow volumes decrease 0.12.

These results demonstrate that increased transmission capacity reduces congestion levels, changing participant behaviors and strategies in the wholesale market.

[12] Seasonal (3+ month) and annual FTR auctions excluded given less effect from transmission changes. Monthly auctions comprise about 50% of MISO FTRs. Re-sale auctions also excluded.

Table 3.9: OLS Regressions - FTR Price and Volume

	FTR Price		FTR Volume	
	Forward	Counter	Forward	Counter
Trans. Line capacity	-2.518***	21.358***	-0.120***	0.110***
	(0.40)	(0.80)	(0.00)	(0.01)
Wind Gen(Gw)	-0.063	1.006***	0.008***	0.004***
	(0.06)	(0.09)	(0.00)	(0.00)
Load(Gw)	0.346***	-0.485***	0.002***	0.001*
	(0.03)	(0.04)	(0.00)	(0.00)
Constant	334.455***	-288.587***	5.564***	4.907***
	(1.80)	(2.70)	(0.02)	(0.02)
Fixed effects	Yes	Yes	Yes	Yes
R-squared	0.700	0.546	0.832	0.795
Observation	43,824	43,824	43,824	43,824

Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. FTR price is the average monthly FTR price across nodes. FTR volume is the average monthly FTR volume in megawatts (MW) across nodes. Includes only monthly auctioned FTR transactions, buy-type FTRs only. Forward flow FTRs pay owners when the line is congested. Counterflow FTRs require owners to pay MISO when the line is congested.

3.5 Conclusion

This study strongly demonstrates that large transmission capacity expansions positively impact electricity market efficiency, both physically and financially. Physically, the overall congestion level and associated regional electricity price differences decrease. The large expansion also affects electricity financial markets—decreasing day-ahead convergence bidding prices and volumes, while increasing financial transmission rights (FTR) volumes. Notably, it arises a shift in FTR auction volumes from forward flow FTRs to counterflow FTRs.

This study has some limitations. First, it observes only aggregated market price and volume changes, instead of individual changes by node and participant. Bidding node information is masked, precluding analysis on how electricity prices at specific nodes change in relation to transmission expansions and congestion levels. Second, this research does not analyze the effect of individual bidder behavior on markets because virtual bidding participant identities are also masked. Participant information is available for FTR auctions but cannot be matched

to masked bidding data.

With weather-driven volatility increasing reliability risks, this study shows well-planned transmission expansion can strengthen the reliability and efficiency of deregulated electricity market. Since new large-scale transmission requires astronomical investments, the benefits of expanded capacity should be studied thoroughly. Further research could evaluate changes in market power, competitiveness, and social welfare.

Chapter 4

Statacons: An SCons-Based Build Tool for Stata

4.1 Build Tools

Transparency and reproducibility are increasingly important norms in empirical science (Wilson et al. 2014; Stodden et al. 2018; Christensen et al. 2019; Orozco et al. 2020). Public posting of data and code, use of version control software, and literate programming are all becoming more common, both in general and with Stata users specifically.

One set of tools that have not received as much attention in Stata are *build tools* (Mokhov et al. 2018). Build tools track *dependencies* throughout a project: which inputs and programs are used to create which outputs, which outputs are subsequently used or combined to create final outputs, etc. A project with a workflow encoded in a build script can be replicated from beginning to end with one command. Just as importantly, when inputs or code change, a build tool can decide exactly which outputs need to be rebuilt, avoiding both errors of omission – failing to update outputs when inputs change – and errors of inclusion – unnecessarily re-running potentially time-consuming code when no relevant inputs have changed.

Stata users already often instinctively create a rudimentary build tool: a "master" do-file that

lists all of a project's do-files in order. This is adequate for simple projects, but has limitations in complex workflows. First, a list of do-files provides no information on what produces what, and what the relationships among different inputs and outputs are.[1] This makes it difficult to trace outputs back to their source or debug problems. Second, a master do-file does not handle the tradeoff between errors of omission and errors of inclusion well. Running a master do-file from beginning to end should avoid errors of omission, but is inefficient if some sections of the project take a long time to run. It would not make sense to re-run a computationally intensive estimation or bootstrapping procedure just to adjust the formatting of some graphs. Again, users often intuitively handle these problems by commenting out lines of the master do-file or setting up switches to skip sections of code, and this is usually fine for simple projects. Furthermore, after an initial setup, virtually all the effort in using a build tool consists in encoding inputs and outputs, so on the margin there is little additional effort involved in using a build tool relative to providing comments in a master do-file.

However, as projects become more complex, an approach like a master do-file can be inadequate. For example, many projects have some or all of the following characteristics: merging multiple input datasets and producing several inter-related datasets for analysis; results of some analyses being used as inputs for others (e.g., estimation results used in subsequent simulations); multiple collaborators working on different aspects of the project simultaneously, using version control software and file-sharing platforms to share code and files; combining Stata tasks with non-Stata tasks, e.g., compiling text, tables and figures into a PDF. In such cases, it is easy to lose track of which parts of a project need to be rebuilt. Similarly, in projects with multiple collaborators (or a single researcher collaborating with her past self), one collaborator may not know about the dependence of one part of the project on another.

Software developers face similar problems, and have developed sophisticated build tools to deal with them. Two ideas are central: first, the relationships between inputs, code and outputs

[1] Comments in a master do-file can provide some documentation of inputs and outputs, but drift between comments and the true content of code is a notorious source of confusion and error (Gentzkow and Shapiro 2014).

are encoded in a *build script*; second, the build software uses this script and information it collects on the state of the project to manage the build, deciding what is up to date and what needs to be built or rebuilt. The build script itself provides living documentation of the full structure of the project, and the build tool can provide the user with the status of of any particular component, or of all components.

The use of build tools does not appear to be common among Stata users. We suspect the main barriers to adoption are: first, lack of awareness of the existence and value of build tools; second, the startup cost of configuring the tool, system and Stata so that they can work together; third, the startup cost of learning how to use these tools.

In this paper and with `statacons`, we attempt to address all three of these barriers. First, we will show in our examples that `statacons`can be used to manage both simple and complex workflows efficiently. Second, because of the integration of Python and Stata in recent versions (16+), as well as our own adaptations, we are able to make SCons, a popular Python-based build tool, accessible in Stata through our command `statacons`. With a minimal amount of configuration, Stata users can use `statacons` directly in Stata and write the build scripts, `SConstructs`, directly in Stata's do-file editor, taking advantage of the editor's Python syntax highlighting.[2] Third, our examples in this paper, as well as additional examples in our companion web tutorial and project Wiki, can easily be adapted to users' own work.

More specifically, `statacons`offers the following benefits: (a) not requiring changes to existing Stata code or the structure of existing workflows; (b) rebuilding exactly those parts of a project that need to be rebuilt at any given moment, with neither errors of inclusion nor omission; (c) complementing existing workflow practices, including literate programming and use of version control software and file-sharing platforms; (d) allowing easy extensiblity; and (e) operating fully within Stata.

Our paper proceeds as follows: in Section 4.2, we use a simple example of managing a Stata workflow with `statacons`and an `SConstruct` to introduce basic concepts, vocabu-

[2] `SConstruct` files written for `statacons`are standard `SConstructs` and will work without modification in standard SCons, so users who prefer the terminal can continue to work there.

lary and mechanics; in Section 4.3, we provide the syntax of the `stataconscommand` and `SConstructs`; in Section 4.4, we present an applied example of using `statacons`to manage the complex workflow of an empirical project (Shumway and Wilson 2021); in Section 4.5, we describe some of the technical details of our adaptation of SCons to Stata; in Section 4.6, we discuss extensions, alternative approaches, and limitations; Section 4.7 concludes. We include a detailed installation guide at the end of the paper. Online Appendices discuss parallel builds, illustrate some additional advanced features, and provide examples of the use of `statacons` in collaborative projects using version control software and shared files. Our project web page[3] provides installation instructions and hosts a companion web tutorial based on Software Carpentry's "Automation and Make" tutorial (Jackson 2016) with several additional worked examples.[4] The `statacons` package is available at our project repository[5], where we have posted complete replication code and data for the Introductory Example and key code from the Applied Example, as well as a project Wiki with examples of additional tools.

4.2 Introduction by Example

In this section, we use a simple example to introduce the basic concepts, terminology and mechanics of managing a workflow with `statacons`. Replication code and data for this example are provided in `stataconsIntro.zip`, posted to our repository.[6]

```
  // dataprep.do
version 16.1
use "inputs/auto-original.dta", clear
generate mpg_sqd = mpg{\caret}2
label variable mpg_sqd "Mileage (mpg) squared"
save "outputs/auto-modified.dta", replace
```

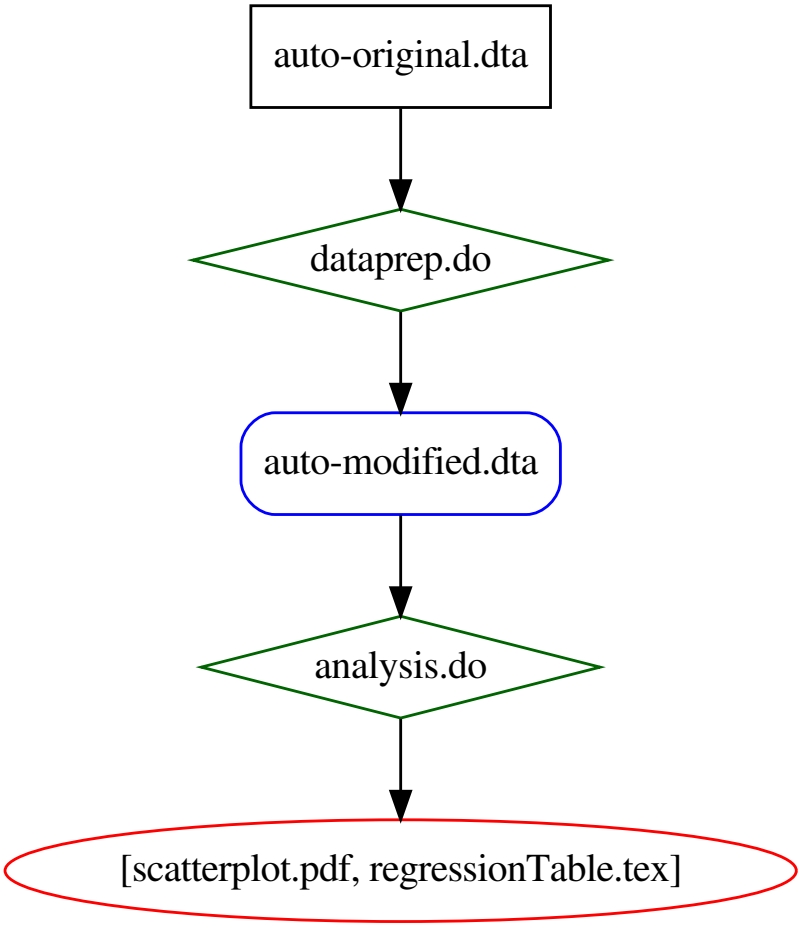
[3] `https://bquistorff.github.io/statacons`
[4] `https://bquistorff.github.io/statacons/swc`
[5] `https://github.com/bquistorff/statacons`
[6] `https://github.com/bquistorff/statacons/raw/main/examples/stataconsIntro.zip`

```
// analysis.do
version 16.1
{\smallskip}
use "outputs/auto-modified.dta", clear
{\smallskip}
twoway scatter price mpg
graph export "outputs/scatterplot.pdf", replace
{\smallskip}
regress price mpg
eststo linear
{\smallskip}
regress price mpg mpg_sqd
eststo quadratic
{\smallskip}
esttab linear quadratic using "outputs/regressionTable.tex", ///
  se r2 replace
```

Listing 4.1: File Structure and Do-Files for Introductory Example

Our project has two steps, using the two do-files in Listing 4.1. In the first step, the do-file `dataprep.do` takes an input file, `auto-original.dta`, creates a new variable `mpg_sqd`, which is the square of `mpg`, and saves the resulting dataset as `auto-modified.dta`. In the second step, the do-file `analysis.do` uses `auto-modified.dta` to create a scatterplot of `price` against `mpg`, saved as `scatterplot.pdf`, regresses `price` on `mpg` and `mpg_sqd`, and exports the regression results to `regressionTable.tex` using `eststo` and `esttab` (Jann 2007). This workflow is represented visually in Figure 4.1 and can easily be managed by the master do-file at the bottom of that figure.

When using a build tool, it is useful to think about the workflow differently. Rather than think about a sequence of do-files in a particular order, we think about the *targets* that we want to build and the *dependencies* that we use to create these targets. In this example, the ultimate targets are `scatterplot.pdf` and `regressionTable.tex`. The dependencies for these targets are the *source* code `analysis.do` and the dataset `auto-modified.dta`. The




```
// master.do
version 16.1
do "code/dataprep.do"
do "code/analysis.do"
exit
```

Figure 4.1: Linear Workflow and Master Do-File for Introductory Example

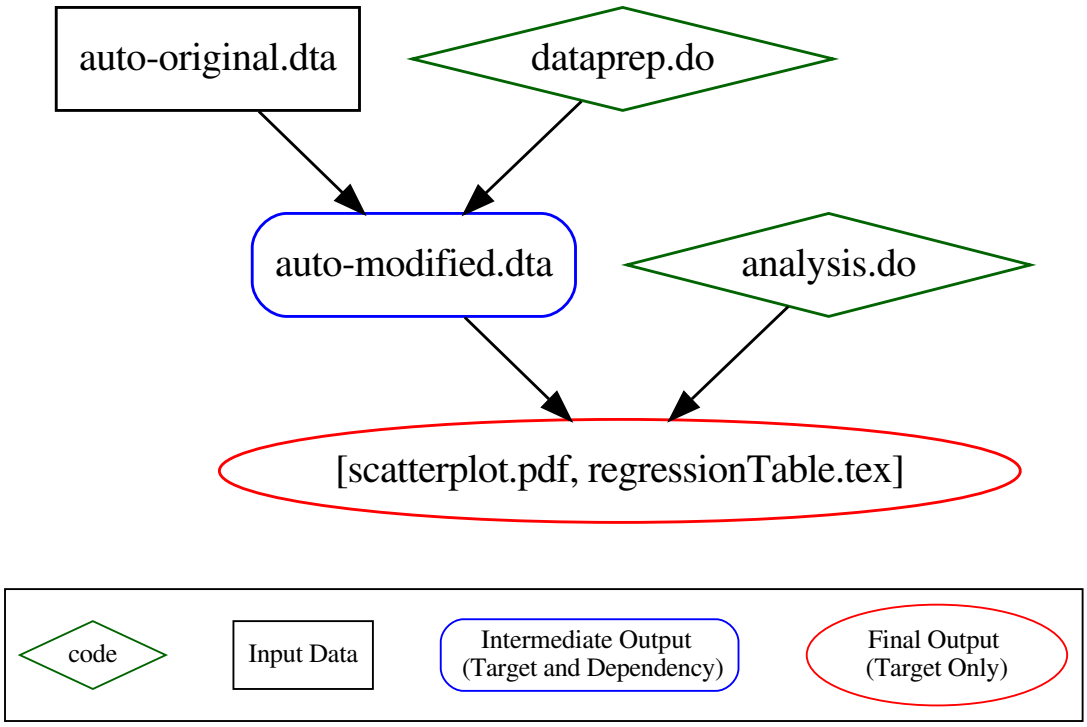


Figure 4.2: Target-Build Workflow for Introductory Example

dataset `auto-modified.dta` is itself a target, with dependencies `dataprep.do` and `auto-original.dta`.

This workflow is depicted visually in Figure 4.2. To use `statacons`, we encode these relationships in an `Sconstruct` file. The `Sconstruct` file used in this example is shown in Listing 4.2.

```
# **** Setup from pystatacons package *****
import pystatacons
env = pystatacons.init_env()
# **** Substance begins  *****
# analysis
cmd_analysis = env.StataBuild(
    target = ['outputs/scatterplot.pdf',
              'outputs/regressionTable.tex'],
    source = 'code/analysis.do'
)
Depends(cmd_analysis, ['outputs/auto-modified.dta'])
# dataprep
cmd_dataprep = env.StataBuild(
```

```
        target = ['outputs/auto-modified.dta'],
        source = 'code/dataprep.do'
    )
    Depends(cmd_dataprep, ['inputs/auto-original.dta'])
```

Listing 4.2: SConstruct File for Introductory Example

Let's consider first the analysis task. We have given this task the name `cmd_analysis`.[7] The *targets* are `scatterplot.pdf` and `regressionTable.tex`. The *source* file for this task is `analysis.do`. The source file is the code we will run to build these targets. The `Depends()` line lists any *dependencies* other than the source file.[8] In this case, this is `auto-modified.dta`.

Finally, `env.StataBuild` tells `scons` that this is a task for Stata, to be handled in the way defined by the *Builder method* `StataBuild`. The essence of `StataBuild` is that `scons` will build the targets (`scatterplot.pdf` and `regressionTable.tex`) by running the source file (`analysis.do`) in Stata's batch mode.[9]

The dataprep task, `cmd_dataprep`, is similar and we leave it as an exercise.

We will now use `statacons` to build our outputs. We will run `statacons`with the option `debug(explain)`, which adds some helpful screen output:

```
. statacons,  debug(explain)
scons: Reading SConscript files ...
Using 'LabelsFormatsOnly' custom_datasignature.
Calculates timestamp-independent checksum of dataset,
  including variable formats, variable labels and value labels.
Edit use_custom_datasignature in config_project.ini to change.
  (other options are Strict, DataOnly, False)
scons: done reading SConscript files.
scons: Building targets ...
scons: building `outputs\\auto-modified.dta' because it doesn't
```

[7] More precisely, `cmd_analysis` is a *NodeList*, where the nodes are the targets. Conveniently, you can use the NodeList elsewhere in the SConstruct to refer to the targets. That is, if we use `cmd_analysis` elsewhere in this SConstruct, scons will understand this to mean the targets `scatterplot.pdf` and `regressionTable.tex`.

[8] While not strictly necessary, it is useful to split the source from the other dependencies so there is no ambiguity about what is the code that will be used in the StataBuild environment.

[9] Batch mode does not require user input and is often used for automating Stata tasks. See the 'Stata batch mode' section of the relevant Getting Started With Stata manual.

```
> exist
stata_run(["outputs\\auto-modified.dta"], ["code\\dataprep.do"])
Running: "C:\\Program Files\\Stata16\\StataMP-64.exe" /e do
> "code\\dataprep.do".
  Starting in hidden desktop (pid=38584).
scons: building ‘outputs\\scatterplot.pdf’ because it doesn’t exist
stata_run(["outputs\\scatterplot.pdf", "outputs\\regression
> Table.tex"], ["code\\analysis.do"])
Running: "C:\\Program Files\\Stata16\\StataMP-64.exe" /e do
> "code\\analysis.do".
  Starting in hidden desktop (pid=34176).
scons: done building targets.
```

There are a few key lines from the screen output that we will highlight:

```
scons: building ‘outputs\\auto-modified.dta’ because it doesn’t exist
```

This tells us that `scons` has noticed that `auto-modified.dta` does not exist, and must be created. The way it is created is given in the next two lines:

```
stata_run(["outputs\\auto-modified.dta"], ["code\\dataprep.do"])
Running: "C:\\Program Files\\Stata16\\StataMP-64.exe" /e do
> "code\\dataprep.do"
```

The first of these is what is sent to the Stata builder in `scons`: in words, “build the target `auto-modified.dta` using the source `dataprep.do` in the way defined in the Stata builder.” The second is what the builder tells the computer to do: run the do-file `dataprep.do` in batch mode.[10]

Having created `auto-modified.dta`, SCons now notices that its next target, `scatterplot.pdf`, does not exist, and needs to be built, which it does by running `analysis.do` in batch mode. SCons then concludes that all targets are up to date, and exits.

[10]Exactly where the Stata executable is on your machine will differ from user to user. We have written `statacons`to find it automatically for most standard setups, and allow users to adjust the default in configuration files. We discuss configuration files, by default named `config_project.ini` and `config_local.ini`, in Section 4.3.4.

Incidentally, note that the order in which the tasks appear in the SConstruct is not important. Even though the SConstruct lists the analysis task first, the logic of the build requires that SCons perform the dataprep task first.[11]

Now if we ask `statacons`to rebuild, it will check whether the dependencies of any target have changed since the previous build.

```
. statacons, debug(explain)
scons: Reading SConscript files ...
Using 'LabelsFormatsOnly' custom_datasignature.
Calculates timestamp-independent checksum of dataset,
  including variable formats, variable labels and value labels.
Edit use_custom_datasignature in config_project.ini to change.
  (other options are Strict, DataOnly, False)
scons: done reading SConscript files.
scons: Building targets ...
scons: `.' is up to date.
scons: done building targets.
```

`statacons` does this by calculating file *signatures* – a compact string that will not change if the file contents do not change, and almost certainly will change if the file contents change – for each target and dependency, and comparing these signatures to those from the previous build, which it has saved in a database called `.sconsign.dblite`.[12] The `stataconsign` command (a Stata call to the SCons command `sconsign`) will print the content of the `.sconsign.dblite`.

As an example, here is the the current state of the `.sconsign.dblite` database for this Introductory Example:

```
. stataconsign
=== .:
SConstruct: None 1651866262 581
=== code:
```

[11] Of course, the way the SConstruct is written affects how easy it is for a *human* reader to understand it. In some cases, it may be easier to follow if the main outputs are listed first, as we have done here. In other cases, a "chronological" ordering may be preferable.

[12] Technically there are different types of functions: "hashes", which provide some cryptographic guarantees, and "checksums", which do not. In the text, we use the more generic term "signature" for all these functions.

```
analysis.do: af3fe510ee8586a10401fe8e9b4fef49 1670255364 317
dataprep.do: 57d511b57ba3f9d41a1c488dab10b3ff 1670255364 192
=== inputs:
auto-original.dta: 74:12(71728):3831085005:1395876116:17340132
    1640899314 12765
=== outputs:
auto-modified.dta: 74:13(15616):2430311699:721316426:3189221131
> 1670255367 13701
        code/dataprep.do: 57d511b57ba3f9d41a1c488dab10b3ff
> 1670255364 192
        inputs/auto-original.dta: 74:12(71728):3831085005:1395876116
> :17340132 16
> 40899314 12765 947c32d25892d9cb39480bf147f78ed3
> [stata_run(target, source, env)]
regressionTable.tex: bb4d5242836537f8cb562435b9e17cb8 1670255371 931
        code/analysis.do: af3fe510ee8586a10401fe8e9b4fef49
> 1670255364 317
        outputs/auto-modified.dta: 74:13(15616):2430311699:721316426
>  :3189221131 1670255367 13701 947c32d25892d9cb39480bf147f78ed3
> [stata_run(target, source, env)]
scatterplot.pdf: 5e43c38cd77e29c7235c2866a448742f 1670255371 65452
        code/analysis.do: af3fe510ee8586a10401fe8e9b4fef49
> 1670255364 317
> outputs/auto-modified.dta: 74:13(15616):2430311699:
> 721316426:3189221131 1670255367 13701
    947c32d25892d9cb39480bf147f78ed3
> [stata_run(target, source, env)]
```

Examining the database provides insight into how SCons and `statacons` work. For the code (`analysis.do` and `dataprep.do`), the database is straightforward. Each is listed with three elements: a signature, a timestamp, and the file length. Similarly, for the input dataset `auto-original.dta`, the dataset entry consists of the same three items, although the signature is calculated differently, as we discuss in Section 4.5.

For *generated* files, i.e., files that are built by `statacons`, the database entries are more

complex. As an example, consider the line for `auto-modified.dta`. Recall that this is the output of the dataprep task. The first line in the database entry for `auto-modified.dta` has the same structure as those of the code and input data: a signature, a timestamp and the file length. What is different is that the entry for `auto-modified.dta` has three additional lines: one for each of the two dependencies (`auto-original.dta` and `dataprep.do`) and one for the build action `stata_run`. The lines for the dependencies just replicate their own database entries. The final line is a signature for the build action. These are the three items that SCons needs to consider when deciding whether to rebuild `auto-modified.dta`. If any of the dependencies or if the build action have changed, SCons will decide to update `auto-modified.dta`. If none have changed, SCons will know that it does not need to rebuild. The remaining entries are all for derived files and all have the same structure: one line for the file itself, one line for each dependency and one for the build action.

In this case, `statacons` determines that no dependencies have changed since the last build, so nothing needs to be done.

Now, let's see what `statacons` does when we make a small modification to `analysis.do` only. We add a title to the scatterplot, changing

```
twoway scatter price mpg
```

to

```
twoway scatter price mpg, title("Price vs. MPG")
```

What happens when we run `statacons`?

```
. statacons, debug(explain)
scons: Reading SConscript files ...
Using 'LabelsFormatsOnly' custom_datasignature.
Calculates timestamp-independent checksum of dataset,
  including variable formats, variable labels and value labels.
Edit use_custom_datasignature in config_project.ini to change.
  (other options are Strict, DataOnly, False)
scons: done reading SConscript files.
scons: Building targets ...
```

```
scons: rebuilding ‘outputs\\auto-modified.dta’ because
> ‘code\\dataprep.do’
changed stata_run(["outputs\\auto-modified.dta"],
> ["code\\dataprep.do"])
Running: "C:\\Program Files\\Stata16\\StataMP-64.exe" /e do
> "code\\dataprep.do".
  Starting in hidden desktop (pid=6792).
scons: rebuilding ‘outputs\\scatterplot.pdf’ because
> ‘outputs\\auto-modified.dta’
> changed stata_run(["outputs\\scatterplot.pdf",
> "outputs\\regressionTable.tex"], ["code\\analysis.do"])
Running: "C:\\Program Files\\Stata16\\StataMP-64.exe" /e do
> "code\\analysis.do".
  Starting in hidden desktop (pid=35432).
scons: done building targets.
```

`statacons` sees that we need to rebuild our target `scatterplot.pdf`, because its source file `analysis.do` has changed. `statacons`also sees that there have been no changes to the dependencies for `auto-modified.dta`, so it does not need to rebuild that.

Next, let's explore what happens when we make an edit to `dataprep.do`. We add a variable `mpg_cub`, the cube of `mpg`:

```
generate mpg_cub = mpg^3
label variable mpg_cub "Mileage (mpg) cubed"
```

And we rebuild using `statacons`:

```
. statacons, debug(explain)
scons: Reading SConscript files ...
Using ’LabelsFormatsOnly’ custom_datasignature.
Calculates timestamp-independent checksum of dataset,
  including variable formats, variable labels and value labels.
Edit use_custom_datasignature in config_project.ini to change.
  (other options are Strict, DataOnly, False)
scons: done reading SConscript files.
scons: Building targets ...
```

```
scons: rebuilding ‘outputs\\auto-modified.dta’ because
> ‘code\\dataprep.do’  changed stata_run(["outputs\\auto
> -modified.dta"], ["code\\dataprep.do"])
Running: "C:\\Program Files\\Stata16\\StataMP-64.exe" /e do
> "code\\dataprep.do".
  Starting in hidden desktop (pid=6792).
scons: rebuilding ‘outputs\\scatterplot.pdf’ because
> ‘outputs\\auto-modified.dta’
> changed stata_run(["outputs\\scatterplot.pdf",
> "outputs\\regressionTable.tex"], ["code\\analysis.do"])
Running: "C:\\Program Files\\Stata16\\StataMP-64.exe" /e do
> "code\\analysis.do".
  Starting in hidden desktop (pid=35432).
scons: done building targets.
```

First, `statacons` sees that `dataprep.do`, the source for `auto_modified.dta`, has changed, so `auto_modified.dta` must be rebuilt. Then, after rebuilding `auto_modified.dta`, `statacons` sees that it has changed, and since `auto_modified.dta` is a dependency for our ultimate targets `scatterplot.pdf` and `regressionTable.tex`, these targets must be re-built.[13]

Finally, let's make a small edit to `dataprep.do` that does not change the content of `auto-modified.dta`:

```
// this is an edit to dataprep.do that does not change auto-modified.dta
```

The result of rebuilding with `statacons` may be surprising:

```
. statacons , debug(explain)
scons: Reading SConscript files ...
Using ’LabelsFormatsOnly’ custom_datasignature.
Calculates timestamp-independent checksum of dataset,
  including variable formats, variable labels and value labels.
Edit use_custom_datasignature in config_project.ini to change.
```

[13] Careful readers of the underlying do-files might note that `analysis.do` does not actually use the variable `mpg_cub`. However, `stataconsdoes` not know that – it only sees that a dependency has changed and, since the targets potentially could change, they must be rebuilt. See Section 4.4 for a strategy to avoid this sort of quasi-false positive.

```
  (other options are Strict, DataOnly, False)
scons: done reading SConscript files.
scons: Building targets ...
scons: rebuilding ‘outputs\\auto-modified.dta’ because
> ‘code\\dataprep.do’
changed stata_run(["outputs\\auto-modified.dta"],
> ["code\\dataprep.do"])
Running: "C:\\Program Files\\Stata16\\StataMP-64.exe" /e do
> "code\\dataprep.do".
  Starting in hidden desktop (pid=14608).
scons: done building targets.
```

This highlights an important way in which `scons`, and therefore `statacons`, differs from the classic and widely used build tool `make` and many of its derivatives.[14] As we would expect, `statacons`sees that `dataprep.do` has changed, so `auto-modified.dta` must be rebuilt. However, notice that `statacons`does *not* rebuild `scatterplot.pdf` and `regressionTable.tex`. This is because, while `auto-modified.dta` has been re-built, nothing about it has *changed* other than its timestamp. `statacons`will re-build a target only if a dependency has changed. When deciding whether to re-build `scatterplot.pdf` and `regressionTable.tex`, `statacons`computes a signature of `auto-modified.dta` that depends only on the content of the dataset, not on its timestamp. Then `statacons`will check whether this signature has changed since the previous time the target was built. Since none of the content of `auto-modified.dta` has changed, its signature is unchanged, and `statacons` sees that there is no need to rebuild any more targets.

4.3 Syntax

The syntax of `statacons` follows that of `scons`. For the most commonly used options, we have adapted SCons syntax into a style more familiar to Stata users. These options are listed below.

[14] For example, GNU Make: `https://www.gnu.org/software/make/`.

We also allow users to use the standard SCons syntax, although not a mix of the two in the same command.[15]

The full syntax of `scons` is too long to replicate here, but can be found in the *SCons manual*[16] Here, we present the basic syntax and several of the most useful options, which we have made available in Stata syntax. For options not listed here, the user will need to use standard SCons syntax. More in-depth and comprehensive discussion can be found in the *SCons user guide* [17]

The syntax of `statacons` is:

```
statacons [targets][, options(values)]
```

The SCons equivalent is:

```
statacons [option[=value]][targets]
```

By default, `statacons` will look in the current directory for an `SConstruct` file, consider all targets described by that file, and then build or re-build targets when needed.

4.3.1 Selecting targets

The user can select a single target or a subset of targets by specifying them on the command line. In the example from the previous section, to rebuild only `auto-modified.dta` but not the other targets, we would use:

```
. statacons outputs/auto-modified.dta
```

The SCons equivalent is identical.

Alternatively, the user can edit the `SConstruct` file to specify the default targets using the `Default()` function. See the SConstruct syntax subsection below.

If no targets are specified on the command line and no `Default()` function given in the `SConstruct`, then `statacons` will consider all targets in the `SConstruct` in the current directory or sub-directories.

[15] Invoking SCons from a terminal prompt requires SCons syntax.
[16] https://scons.org/doc/4.3.0/PDF/scons-man.pdf
[17] https://scons.org/doc/4.3.0/PDF/scons-user.pdf

4.3.2 Options

Standard `scons` options

We discuss several of the most commonly used options. For a complete list of options, see the SCons manual.

- `clean` will *clean*, i.e., delete, specified targets. Use this option with care, since by default it will remove all targets defined in the `SConstruct`.

  - SCons equivalent: `-c`, `-clean`, `-remove`

- `debug(type [,type...])` will print additional information to the screen to help *debug* the build process. What information is printed depends on the `type` or `types` specified. The most useful `type` is `explain`, which will lead `statacons` to print an explanation for each target it selects to build or re-build – typically, which dependency has changed. Note that explanations are not given for targets that are not built.

  - SCons equivalent: `-debug=type[,type...]`

- `directory(directory)` change directory to `directory` before starting build.

  - SCons equivalent: `-C directory`, `-directory=directory`
  - Note that the change of directory will occur only within the SCons process; the user's working directory will not change in Stata itself.

- `dry_run` will put SCons into *no execute* mode. SCons will attempt to determine what targets it would build and print the commands it would execute, but will not actually execute these commands.

  - SCons equivalent: `-n`, `-no-exec`, `-dry-run`, `-just-print`
  - It is important to note that because SCons by default rebuilds targets only when dependencies have changed, SCons may not be able to determine whether it will need to rebuild "downstream" targets. For example, suppose `A.dta` is a dependency of `B.dta` and `B.dta` is a dependency of `C.dta`. When `A.dta` changes, SCons knows it must rebuild `B.dta`. However, since in *no execute* mode SCons does not actually rebuild `B.dta`, it does not know whether `B.dta` will *change*. As a result, it does not know whether it will have to rebuild `C.dta` and will not print any commands for rebuilding `C.dta`. In general, we can think of the default set of commands printed in *no execute* mode as the smallest set that will be executed.

- `file(file)` or `sconstruct(file)` allows the user to specify *file* as the initial SConstruct file for `statacons` to read. By default, `statacons` will look for a file named `SConstruct` in the current directory.

  - SCons equivalent: `-f file`, `-file=file`, `-sconstruct=file`

- `help` prints the full SCons help message to the screen. The user can define an additional help message in the `SConstruct` using the `Help` function, see the *Scons Manual* for details. (`statacons, help` is distinct from `help statacons`, as the latter gives the help file for the Stata command, not the SCons program.)

  - SCons equivalent: `-h`, `-help`

- `q` do not print SCons status messages ("quiet")

  – SCons equivalent: `-Q`

- `silent` do not print messages about SCons actions nor SCons status messages ("silent"). Implies `q`.

  – SCons equivalent: `-s, -silent, -quiet`

- `tree(type[,type...])` prints a *dependency tree*, which is useful for debugging or checking on the status of a build, or just for visualizing and understanding the workflow. The part of the tree printed and the format depend on the `type` or `types` specified:

  – `all`: print the entire tree

  – `derived`: print only "derived" (i.e., target) files. Omit files that are not targets, e.g., code or input data.

  – `linedraw`: uses unicode characters to draw the tree rather than the default ascii. Some users may find this easier to read on the screen.

  – `status`: prints the current status of each item on the tree. This is useful for understanding the status of a build, since it will tell you which files are current ("`C`"), which files exist ("`E`"), and several other status categories.

  – `prune`: makes the tree easier to read by not repeating dependencies for files that have already been described by the tree. Without this option, `tree` prints a lot of redundant information. With the `prune` option invoked, `tree` will indicate that a file's dependencies have already been listed by enclosing the file name in [square brackets].

– SCons equivalent: `-tree=type[,type...]`

– We most commonly invoke `tree` as `tree(status,prune)`. For our introductory example, this produces the output shown in Listing 4.3.

– Exporting the output of `tree` to a text file[18] will produce output similar to that of the BITSS / Social Science Reproduction Platform Diagram Builder (BITSS 2020).

```
. statacons, tree(status,prune)
 E         = exists
  R        = exists in repository only
   b       = implicit builder
   B       = explicit builder
    S      = side effect
     P     = precious
      A    = always build
       C   = current
        N  = no clean
         H = no cache
[E b   C  ]+-.
[E b   C  ]  +-code
[E     C  ]  | +-code\\analysis.do
[E     C  ]  | +-code\\dataprep.do
[E b   C  ]  +-inputs
[E     C  ]  | +-inputs\\auto-original.dta
[E b   C  ]  +-outputs
[E B P C  ]  | +-outputs\\auto-modified.dta
[E     C  ]  | | +-code\\dataprep.do
[E     C  ]  | | +-inputs\\auto-original.dta
[E B P C  ]  | +-outputs\\regressionTable.tex
[E     C  ]  | | +-code\\analysis.do
[E B P C  ]  | | +-[outputs\\auto-modified.dta]
[E B P C  ]  | +-outputs\\scatterplot.pdf
[E     C  ]  |   +-code\\analysis.do
[E B P C  ]  |   +-[outputs\\auto-modified.dta]
[E     C  ]  +-SConstruct
```

[18] For example, in a Unix-like terminal, enter `scons -ns -tree=prune > treeOutput.txt`

Listing 4.3: SCons Tree from our Introductory Example

Custom `statacons` options

All of the options above are standard to SCons. We have written several options specifically for `statacons`.[19]

- `assume_built("target")` will instruct the Stata builder to skip a task if all of its targets are listed and then to mark those targets as up-to-date. This can be a colon-separated list of file patterns.
  - SCons syntax: `-assume-built="target"`
- `assume_done("filename.do")` will instruct the Stata builder to skip the given do-file in the current build, but mark the associated target(s) as up-to-date. This is useful if we know that a target is up to date but SCons does not recognize it as such, for example: (1) if we have just built `target.tex` by running `filename.do` directly in Stata rather than through `scons` or `statacons`; (2) if we have only made edits that we know will not result in any changes to `target.tex`, such as adding comments in `filename.do`. Works even if the task for this do-file includes parameters or a non-`do` file command. See discussion in Section 4.6.3, "Limitations."
  - SCons syntax: `-assume-done="filename.do"`
  - `assume_done("file name1.do:file name2.do [ : ... ]")` will split by the colon and skip each file. Each component can be a file pattern, as seen in the next example.
  - SCons syntax: `-assume-done="file name1.do:file name2.do[: ... ]"`
  - `assume_done(*)` will skip all do-files in the current build and mark all their targets as up-to-date.
  - SCons syntax: `-assume-done=*`
- `config_file(CONFIG_FILE)`. By default, `statacons` will look for `config_project.ini` and `config_local.ini` in the same directory as the `SConstruct`. If these files

[19] Standard SCons (i.e., invoked as `scons` from the command line) will process these commands as long as you have installed the Python package `pystatacons` and your `SConstruct` imports `pystatacons`. SCons-style syntax will be required.

do not exist or they do not specify several core parameters, such as the location of the Stata executable, defaults based on typical configurations for the user's operating system, version and edition of Stata, will be used. Setting `config_file(CONFIG_FILE)` will instruct the Stata builder to use CONFIG_FILE as the configuration file. See Section 4.3.4 for more on default settings and configuration files. The user can specify multiple files as `FILE1:FILE2:...:FILEN`. In case of conflicts between files, the file that appears later in `FILE1:FILE2:...:FILEN` will take precedence.

  - SCons syntax: `-config-file=CONFIG_FILE`

- `show_config` will print the current values of all configuration options, whether chosen by default or specified in a configuration file. `statacons, show_config` activates the `dry_run` and `silent` options, so `statacons` will not build any files or report anything about the state of the build.

  - SCons syntax: `-show-config`

4.3.3 SConstruct syntax

Basic SConstruct recipe

The basic recipe for encoding a target built by Stata into an `SConstruct` file is:

```
task_name = env.StataBuild(
      target = ['path/to/target1.ext', 'path/to/target2.ext'],
      source = 'path/to/dofile.do'
)
Depends(task_name, ['path/to/dependency1.ext',
                    'path/to/dependency2.ext']
)
```

In Section 4.2, we described each of the elements: *builder* (here, `StataBuild`), *target*, *source*, and *dependencies*. `target` and `source` are both required.[20]

[20] For managing tasks that do not build specific targets, see "**??**" below.

Additional options for SConstruct recipe

We have added some options to the basic recipe:

```
task_name = env.StataBuild(
      target = ['path/to/target1.ext', 'path/to/target2.ext'],
      source = 'path/to/source.ext',
      file_cmd = "command",
      params = 'arguments or options',
      depends = ['path/to/dependency1.ext',
                 'path/to/dependency2.ext']
)
```

The additional options are as follows:

- `file_cmd`: the command that SCons should pass to Stata's batch mode. The default is `do`, but the user can specify anything that Stata can accept as a command, e.g., `dyndoc`, `markdown`, etc.
- `params`: arguments or options that should follow the `source` in the call to Stata batch mode. For example, a call to `markdown` might specify

  ```
  params = ', saving(myfile.html) replace'
  ```

- `depends`: an alternative to using the `Depends()` function

We include several additional examples of these options in our web tutorial.[21]

Defining lists of files in SConstructs

As projects grow in size, it may become unwieldy to write out long lists of dependencies. Furthermore, if different targets have overlapping sets of dependencies, writing the same dependencies in different places is inefficient and can lead to errors. We can do better by defining variables in our SConscript files using standard Python syntax and some special SCons

[21] `https://bquistorff.github.io/statacons/swc.html`, especially the "Parameters" lesson.

functions. As a preview of the Applied Example in Section 4.4 below, we illustrate two of these methods.

One of the targets, `ctyQtrWages.dta`, uses as inputs five Excel files named `wagedata1.xlsx`, ..., `wagedata5.xlsx`. We could write these out explicitly in the `Depends()` line, but instead we will demonstrate two alternatives. The first is the SCons `Glob()` function:

```
wageDataFiles = Glob("inputs/wagedata[1-5].xlsx")
Depends(wage, wageDataFiles)
```

where `[1-5]` means "any element of the sequence 1, 2, ..., 5."[22] SCons will search the directory `inputs` for files matching the pattern and will define `wageDataFiles` to be the set of files that match.

A more flexible alternative is to use Python's "list comprehension" method as follows:[23]

```
wageDataFiles = ["inputs/wagedata"+e+".xlsx"
                 for e in ['1','2','3','4','5']]
Depends(wage, wageDataFiles)
```

While this method is not as concise as `Glob()`, it has a few advantages. First, it allows for more complex structures and patterns. For example, a sequence of ISO country codes would begin `'AFG','ALA','ALB'`. Second, it can define lists of targets that may not yet exist, while `Glob()` is limited to matching existing files. Third, it is more precise. As a concrete example, suppose the file `wagedata1.xlsx` is missing from the `dataprep/inputs` directory. If we write out all our dependencies explicitly or use the list comprehension method above, SCons will return an error, which is useful. If we use `Glob("inputs/wagedata[1-5].xlsx")`, SCons will think that the dependencies are `wagedata2.xlsx`, ..., `wagedata5.xlsx`, i.e., the files that are present, and will proceed with the build.[24]

[22] `Glob()` allows unix shell pattern matching, i.e., `*` (match everything), `?` (match single character), `[seq]` (match any singe character in the sequence `seq`), and `!seq` (match any single character not in the sequence `seq`). See the SCons User Guide for more details.

[23] See `https://docs.python.org/3/tutorial/datastructures.html` for more information on list comprehension and other Python data structures.

[24] Despite our concerns about `Glob()`, we have used it in our Applied Example for the sake of providing an example. See the definition of the `industryCompFiles` variable in `dataprep/SConstruct`.

See the companion web tutorial[25] for more examples of using Python coding to write more concise `SConstructs`.

SConstruct functions

We discuss only a few of the more useful functions available in SConstruct files. See the *Scons manual* or the *SCons User Guide* for a comprehensive list and more details on implementation. We also note a few additional options that the `pystatacons` package provides.

Note that some of these functions (e.g., `SetOption()`), will require the use of *nix-like standard SCons syntax rather than our adaptation to Stata syntax.

- `AlwaysBuild(target[,target...])`: instructs SCons that the given targets should always be rebuilt when they are specified, even if they are up to date. Note that a target must be specified either in the command line or as part of the default set – that is, `AlwaysBuild()` neither implies nor is implied by `Default()`.

- `Default()`: allows the user to specify which targets SCons will examine by default. In our introductory example, we would set a default of rebuilding only `auto-modified.dta` but not the other targets by adding `Default('outputs/auto-modified.dta')` to the `SConstruct`. If `Default()` is not specified, SCons will examine all targets in current directory or subdirectories unless otherwise specified on the command line.

- `Decider(function)`: determine how SCons will decide whether a target is up to date.

- `"content"`: the default. For each dependency of a target, SCons will calculate a signature and compare the result against the last time the target was built.[26] This means that a target will be rebuilt only if one if its dependencies have *changed*. So, suppose `A.dta` is a dependency of `B.dta` and `B.dta` is a dependency of `C.dta`. Now, suppose that `A.dta`

[25] `https://bquistorff.github.io/statacons/swc.html`, especially the "Variables" lesson.

[26] Rather than recalculating the signature, SCons will save time by using a cached value of the signature if (1) a cached value is available from previous builds, (2) the modification timestamp has not changed, and (3) the timestamp is at least as old in seconds as the `max-drift` parameter (default is two days, but is configurable).

changes, but when `B.dta` is rebuilt, it does not change. Even though `B.dta` is *newer* than `C.dta`, SCons will not rebuild `C.dta`.

- `"timestamp-newer"`: SCons will rebuild a target if any of its dependencies are newer than the target itself (similar to the classic build tool `make`). Suppose `A.dta` is a dependency of `B.dta` and `B.dta` is a dependency of `C.dta`. Now, suppose that `A.dta` changes, but when `B.dta` is rebuilt, it does not change. Even though `B.dta` has not changed, it is newer than `C.dta`, so SCons will rebuild `C.dta`. Because it does take time for SCons to calculate signatures, `timestamp-newer` may be faster despite some unnecessary rebuilds if there are relatively many input files (and therefore many signatures to calculate) and relatively low computation time.

- `"timestamp-match"`: SCons will rebuild a target if any of its dependencies have a timestamp different (newer or older) than the last time the target was built. If all dependencies' timestamps are the same as the last time the target was built, the target will be considered up to date and will not be rebuilt.

- `"content-timestamp"`: rebuild a target only if a dependency's timestamp *and* signature have changed since the last time the target was rebuilt, that dependency is considered up to date. This is similar to `content`, except it skips checking signatures for dependencies whose timestamps are unchanged and therefore may run faster.

- `"content-timestamp-newer"`: rebuild a target only if a dependency's signature has changed since the last time the target was rebuilt *and* any of its dependencies are newer than the target itself. Similar to the `assume_done` and `assume_built` command-line options defined above, this means that SCons will not re-run Stata code that has already been run outside of SCons. Unlike `assume_done` and `assume_built`, it only needs to be specified once in the SConstruct, not specified at the command line each time. `"content-timestamp-newer"` is defined by `pystatacons`. To use it, include

`Decider(pystatacons.decider_str_lookup["content-timestamp-newer"])` in your SConstruct after `pystatacons` has been imported.

- `Ignore(target, dependency)`: tell SCons to ignore the file *dependency* when deciding whether to rebuild *target*.
- `Ignore(directory, target)`: remove *target* from the set of default targets. The first argument of `Ignore()` must be the *directory* where *target* would be rebuilt. SCons will still build *target* if (1) *target* is specified on the command line or (2) *target* is a dependency of another target and needs to be rebuilt.
- `Requires(target, prerequisite)`: specify that `target` should be built after `prerequisite` even if `prerequisite` is not a dependency for `target`. This is useful if the user wishes to impose some order on the build even if that order is not required by the build logic. For example, the user may want to move longer-running tasks later in the build so she can inspect the output of shorter-running tasks while the others run.
- `SetOption(name, value)`: allows some command-line options to be set in the SConstruct. For example, `SetOption('silent',1)` is equivalent to `silent` on the command line. See the User Manual for the options that can be set by `SetOption()`. The command line over-rides values set through `SetOption()`.

4.3.4 Default settings and configuration files

We have incorporated what we think are sensible defaults into `statacons`. These include:

- finding the Stata executable automatically by searching the user's `PATH` variable and other default locations;
- using our custom `complete_datasignature.ado` to compute signatures for `.dta` that depend only on their content, not their embedded timestamp. (See discussion in Section 4.5);

- deleting the batch-mode generated log-files after a do-file is successfully executed in batch mode, but retaining it in the default directory if the do-file fails.

By setting these defaults, we reduce the need for users to spend time figuring out how to configure the tool, which we suspect has been a barrier to adoption of build tools previously. However, some users may wish to choose other options or have unusual setups. Therefore, we allow users to change their configurations in two files, `config_project.ini` (for settings common to all users in a given project) and `config_local.ini` (for user-specific settings). We provide templates for both with suggested values in the project files installed with `statacons`. In case of conflicts between `config_project.ini` and `config_local.ini`, `config_local.ini` will take precedence.

These configuration files follow a simplified INI format readable by Python's standard library[27]. They are simple text files that are easily read and edited. Each file stores key-value pairs organized under one or more headers.

```
[header1]
key1: value 1
key2: value 2

[header2]
key3: value 3
```

As an example, the user may wish to retain the log-files created by batch-mode Stata in a folder called `logs` and detail which specific Stata executable to use, which can be done in `config_local.ini` with

```
[SCons]
success_batch_log_dir: ./logs/

[Programs]
stata_exe: "C:/Program Files/Stata17/StataMP-64.exe"
```

[27] See `https://docs.python.org/3/library/configparser.html` for full details

As a second example, by default SCons operates relative to the directory where the `SConstruct` is found, so this is the directory from which Stata will run in batch mode. This is a reasonable default, but some users may wish for Stata to start in a different directory. We allow this through a `stata_chdir` option in `config_project.ini`. See the `config_project.ini` file provided with this package for the available options.

We provide some additional examples of the use of configuration files elsewhere in this paper, such as telling SCons where to find shared folders in Dropbox. Users can specify other configuration files using the `config_file()` option listed in the Custom `statacons` options section above. As mentioned there, in the case of conflicts between config files, the file listed later in `config_file()` will take precedence.

4.3.5 Other advanced features

We conclude this section by briefly mentioning a few advanced features of SCons. We have additional discussion and examples in the Appendices. This is not a complete list nor a comprehensive guide; rather, we aim to familiarize readers with some concepts and vocabulary they may encounter while using the tool.

The first is the `SConscript`, which allows users to split a long `SConstruct` into more manageable parts, i.e., a *hierarchical build*.

This second is SCons's ability to manage *parallel builds*, conducting tasks simultaneously rather than sequentially when the logic of the workflow permits. SCons parallel builds are well-suited for a relatively small number of relatively slow tasks. For tasks with a relatively high number of relatively quick tasks, like bootstrapping or simulation, the approach of `parallel` (Vega Yon and Quistorff 2019) is preferable. We provide an application of parallel builds to our Applied Example in Appendix A.2. Parallel builds are currently only available in `scons` (i.e., from a terminal), not in `statacons`.

The third is the SCons *cache*, which allows storing and sharing of derived files.

4.4 Applied Example

This section describes the use of `statacons` in an empirical project (Shumway and Wilson 2021). Key build scripts and do-files are provided in `appliedExample.zip`, posted to our repository.[28]

For our purposes, this project has two main parts: dataprep, in which raw data are cleaned and merged, and new variables are created for analysis; and analysis, in which the analysis of these data is carried out. The value of a build tool is apparent from Figures 4.3, 4.4, 4.5, and 4.6: rather than trying to remember all these interdependencies, we encode them in SConstructs. Consulting an SConstruct and the output of `tree()` is a more reliable way to document interdependencies, to orient a new collaborator, remind the user's future self of how a project works after some time away, trace errors, etc., than depending on memory. The SConstructs and output of `tree` for both the dataprep and analysis tasks are provided in `appliedExample.zip` on the project repository.

We have separated the two main parts of the project, data cleaning and data analysis, into two directories, each with its own `SConstruct`. These two directories are linked by a third, which we have called `transferDataprep`. The phony target `post_outputs` in the dataprep SConstruct copies the main outputs from `dataprep/outputs` to `transferDataprep`, while the phony target `pull_inputs` in the analysis SConstruct copies these from `transferDataprep` to `analysis/inputs`. See Section 4.6.1 for more details and an extension to shared files.

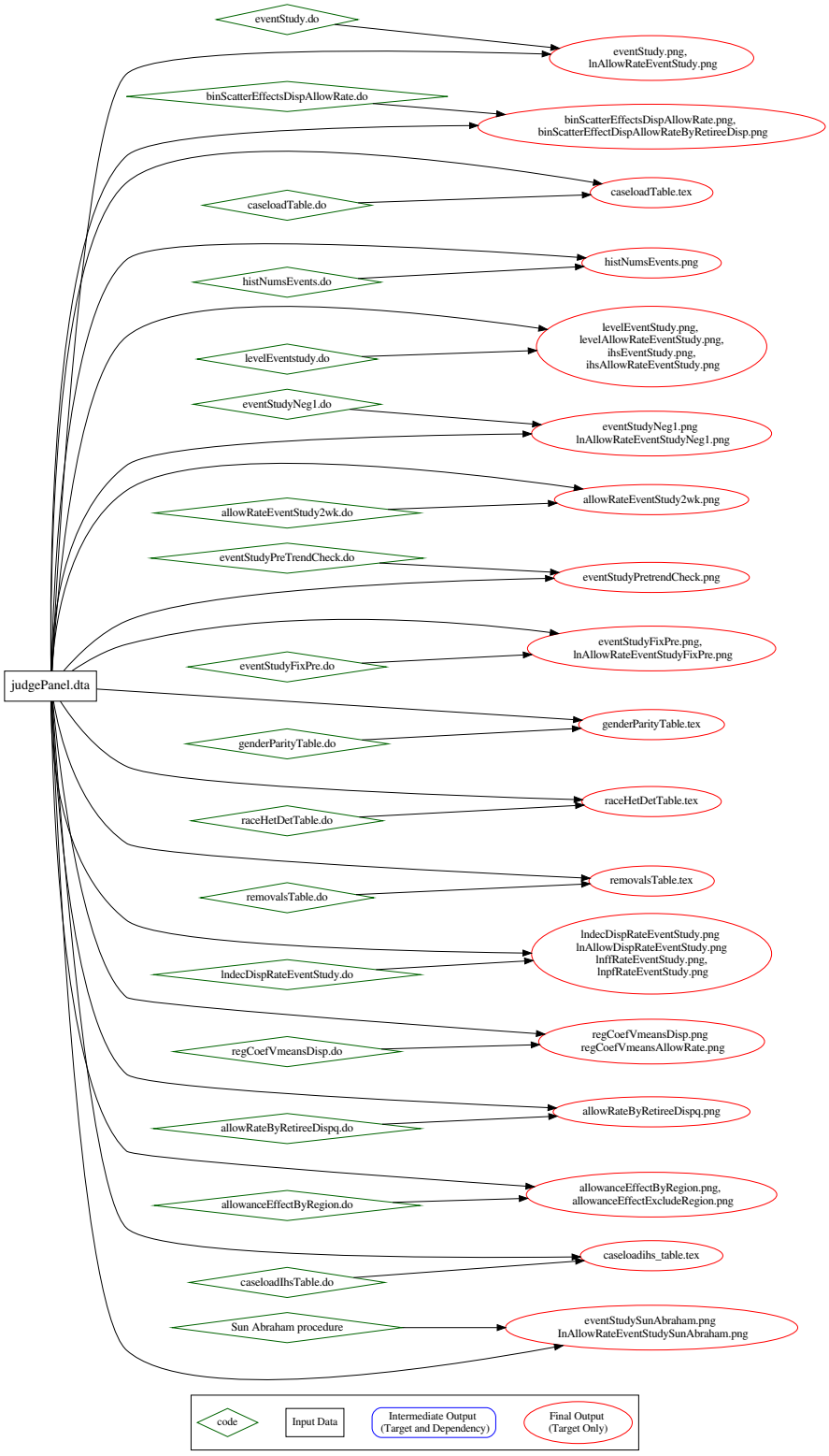
One of the estimation procedures, the dynamic treatment effects estimator of Sun and Abraham (2021), takes over two hours to run even on a fast desktop. We have written our code and SConstruct to avoid repeating this estimation when it is not necessary. The relevant section of the SConstruct is shown in Figure 4.7 along with a diagram.

```
# Sun-Abraham event study estimation methodology
sunAbrahamSelect = env.StataBuild(
```

[28] `https://github.com/bquistorff/statacons/raw/main/examples/appliedExample.zip`

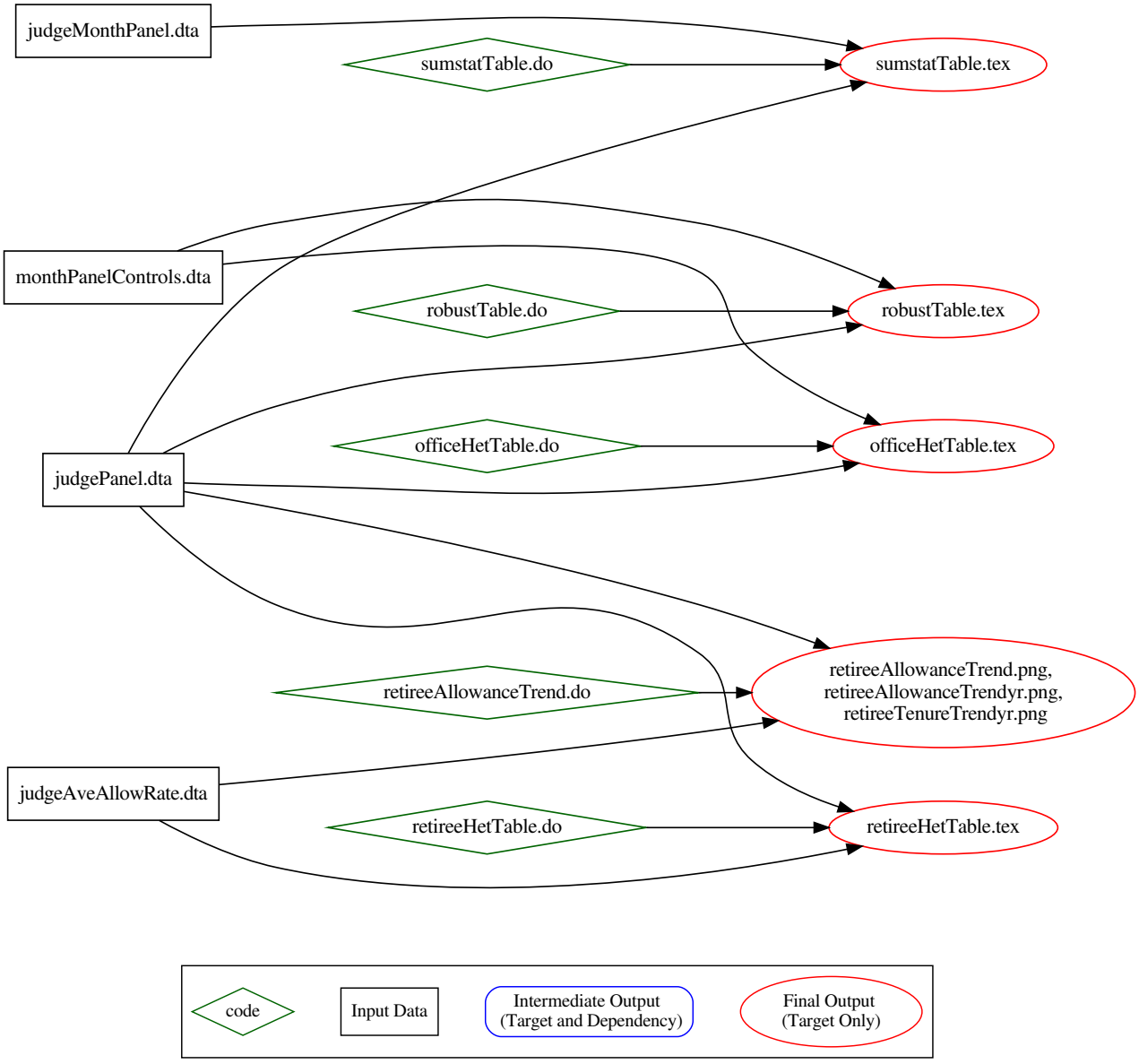
Notes: brackets represent a series of similarly-named files, e.g., `wagedata[N].xlsx` represents `wagedata1.xlsx`, ..., `wagedata5.xlsx`. We have omitted file paths for brevity.

Figure 4.3: Workflow for Applied Example: Dataprep



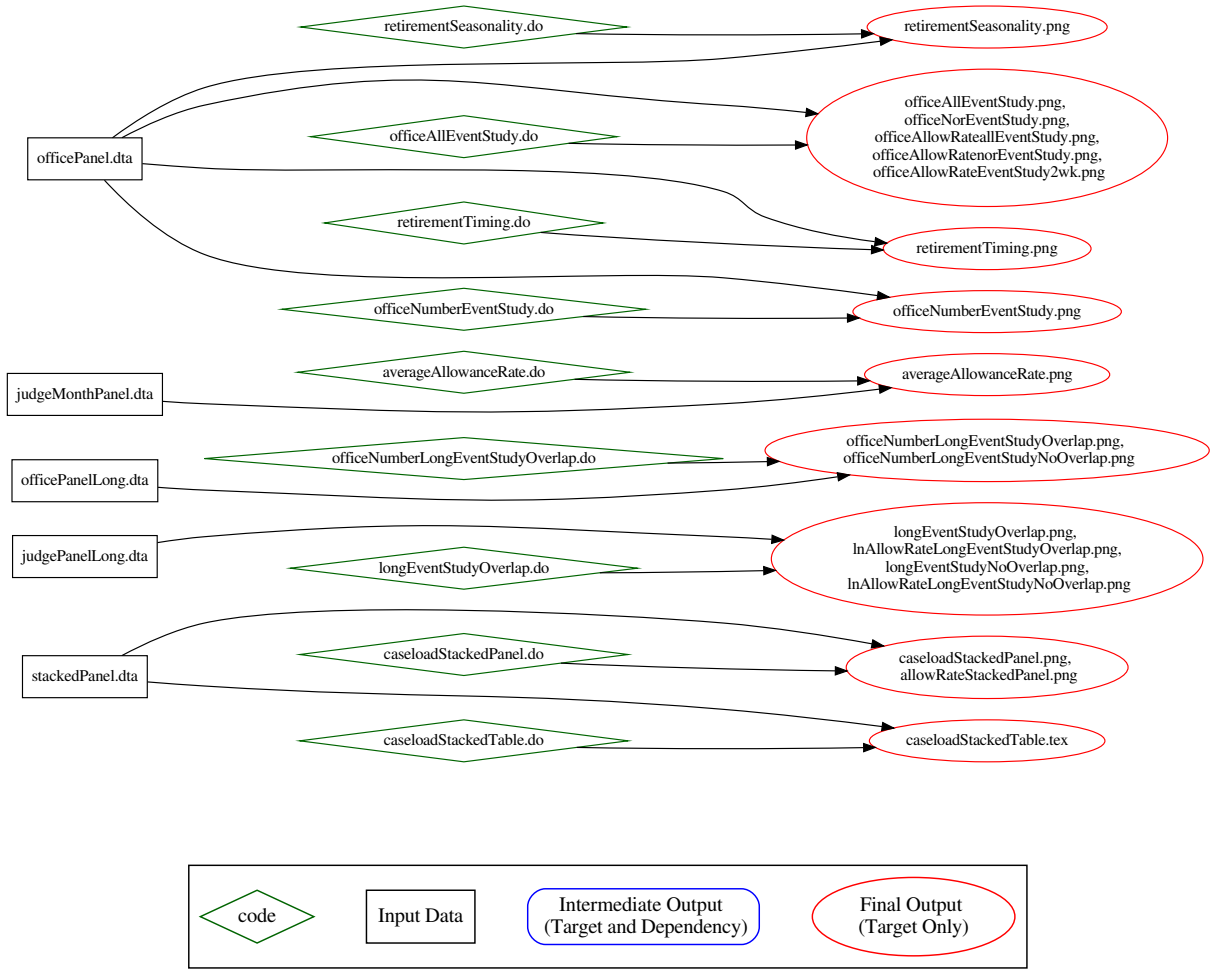
Notes: We have omitted file paths for brevity. The workflow for the Sun-Abraham event study procedure is simplified, see Figure 4.7 for the complete workflow.

Figure 4.4: Workflow for Applied Example: Analysis (Part 1)



Notes: We have omitted file paths for brevity.

Figure 4.5: Workflow for Applied Example: Analysis (Part 2)



Notes: We have omitted file paths for brevity.

Figure 4.6: Workflow for Applied Example: Analysis (Part 3)

```
    source = "code/sunAbrahamSelect.do",
     target = ['outputs/dta/sunAbrahamEstimationSample.dta']
)
Depends(sunAbrahamSelect, ['inputs/judgePanel.dta'])
# do estimation
sunAbrahamEst = env.StataBuild(
    source = "code/sunAbrahamEstimation.do",
    target = ['outputs/dta/sunAbrahamResults.dta']
)
Depends(sunAbrahamEst,
        ['outputs/dta/sunAbrahamEstimationSample.dta'])
# plot results
sunAbrahamFigs = env.StataBuild(
    source = "code/sunAbrahamGraph.do",
    target = ['outputs/figures/eventStudySunAbraham.png',
    'outputs/figures/lnAllowRateEventStudySunAbraham.png']
)
Depends(sunAbrahamFigs, ['outputs/dta/sunAbrahamResults.dta'])
```

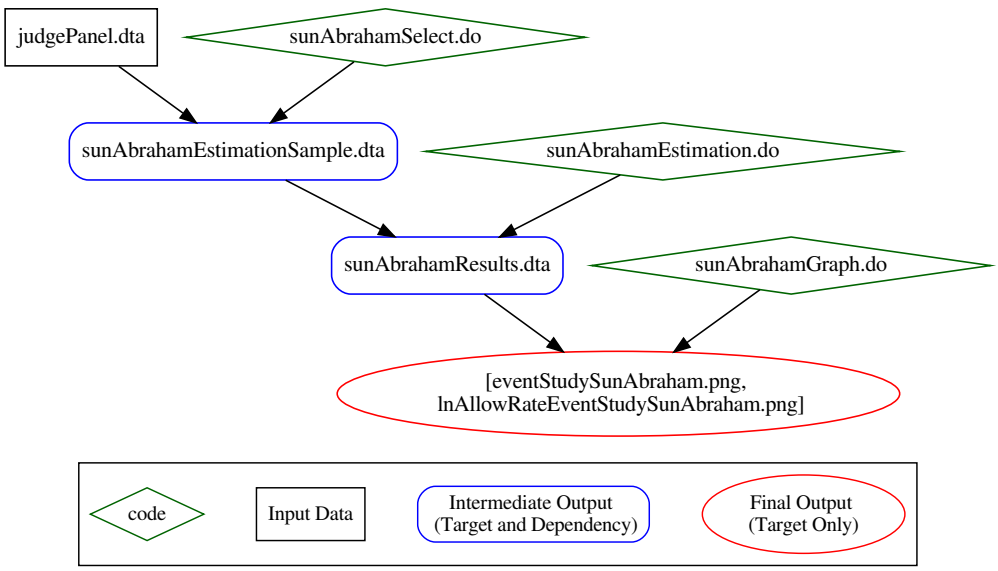


Figure 4.7: Separation of Concerns in Applied Example

First, this task uses only a subset of the variables and observations in `judgePanel.dta`. While we do need to re-run the estimation if the data used have changed, we would not want to re-run the estimation if the only variables or observations to change were not used in this task. The first task, `sunAbrahamSelect`, creates `sunAbrahamEstimationSample.dta`, which keeps only the variables and observations of `judgePanel.dta` needed for this task. So, suppose `judgePanel.dta` has changed. SCons will rebuild `sunAbrahamEstimationSample.dta`. If `sunAbrahamEstimationSample.dta` changes, then SCons will go on to the estimation step in `sunAbrahamEst`, since `sunAbrahamEstimationSample.dta` is the dependency for that task. However, if `sunAbrahamEstimationSample.dta` has not changed, then SCons will not repeat `sunAbrahamEst`.

Similarly, the ultimate outputs are two graphs of estimated treatment effects. Suppose we wanted to change some aspect of the graphs, e.g., the axis titles. We would not want to have to repeat the full estimation. On the other hand, we do want to re-create these graphs whenever the underlying estimates change. We handle this by separating the estimation and the creation of graphs into distinct steps. The task `sunAbrahamEst` handles the estimation, saving the estimation results in `sunAbrahamResults.dta`.[29] Then the task `sunAbrahamFigs` handles producing the graphs. If we edit `sunAbrahamGraphs.do` to change some aspect of the graphs, SCons will not see any need to rebuild `sunAbrahamEst`. On the other hand, if something about the estimation changes such that `sunAbrahamResults.dta` changes, SCons will automatically rebuild the graphs.

This division of tasks is known as *separation of concerns* in computer science and is in our view a key distinction between working with a build tool and working in the literate programming paradigm described in Section 4.6.2. The virtue of literate programming is that everything is in one piece of code, which is certainly convenient and simple. A build tool workflow in

[29] Here we have saved estimation results into a `.dta` dataset. An alternative is to use Stata's `estimates save` command to save results to a `.ster` file. The user-written `estwrite` and `estread` (Jann 2005) improve on `estimates save` by allowing users to store multiple estimation results in a single `.sters` file and then refer to individual estimates by model name. Note that as of version `1.2.5`, `estwrite` has an option `reproducible` that will eliminate the timestamps previously embedded in `.sters` files. As of this writing, `.ster` files produced by `estimates save` contain an embedded timestamp.

which discrete tasks are separated will save computation time. Of course, there is a cost to creating separate do-files, so this tradeoff may not be worthwhile for simple, quick tasks.

Finally, we can further reduce computation time by using SCons's ability to manage *parallel builds*. We discuss this in Section A.2 of the Appendix.

4.5 Technical Details

We note here some technical details of how our package works.

First, as we have discussed above, SCons's default behavior is to look for changes in the content of dependencies when deciding whether to rebuild a target. SCons does this by computing a *signature* of each dependency. However, because Stata embeds a timestamp in `.dta` files, `.dta` files created at two different times will have different signatures even if the contents are identical. This will lead to unnecessary rebuilds of target files. Stata's `datasignature` command can create signatures that depend only on the values of the data in the `.dta` file, not the timestamp, but `datasignature` signatures do not depend on variable labels or value labels.

We create an ado program `complete_datasignature.ado` that augments Stata's datasignature with a signature of the dataset's metadata (variable formats, variable and value labels, notes and characteristics) but not the timestamp. We then patch SCons so that it uses our `complete_datasignature.ado` instead of its default signature algorithm.

In the `config_project.ini` file, the user has several options:

- `Strict`: include data and all metadata in the signature (variable formats, variable and value labels, notes and characteristics)
- `LabelsFormatsOnly`: include data, variable formats, and variable and value labels in the signature (do not include dataset labels, notes, characteristics)
- `DataOnly`: use Stata's standard `datasignature`, which depends on data only (does not

depend on any metadata or the embedded timestamp)

- `False`: use SCons's default signature, which will depend on the embedded timestamp

In addition, we provide some control over the use of `datasignature`'s `fast` mode. The fast mode speeds up the calculation of the file signature, but is not *machine-independent*: users on different machines may produce different signatures. This is not so important for solo users working on a single machine, but for collaborative workflows (or a single user working on different machines) that save time by sharing built files in a common directory, using the `fast` option may lead `statacons` to incorrectly believe that a build is out of date. We make `fast` the default unless the `CacheDir()` option is specified (indicating a collaborative workflow, see Appendix B), in which case the machine-independent but slower method is the default. Users can override these defaults, for example to use the slow option for a solo project, through the `dta_sig_mode` option in the `config_project.ini` configuration file.

Our next set of issues deals with SCons directing Stata to run in batch mode. First, SCons needs to know if the do file finished successfully to know if targets should be considered built. This would typically be communicated via the program's exit status, but Stata always returns a normal exit even if the script it runs encountered an error. Our Stata builder therefore parses the end of the Stata batch-mode log file to see if there was error and communicate this back to SCons. This requires that batch-mode Stata produce plain-text log-files, so we do not allow the user to change the options with which batch-mode Stata is called.

Second, Stata typically uses the do-file name as the name of the batch-mode log-file, but we may need to change this for a variety of reasons. If two commands would result in the same log file name (e.g., we call the same do-file with different parameters or we have multiple non-`do` commands), then they can't be run in parallel because Stata will fail to start when the second instance can't open the log file. Therefore, for all commands aside from plain do commands (with no parameters), we have Stata run a generated do-file that then runs the user's command. We name this generated file so that its resulting log file will be unique (we take the original command's typical log-file name and append a short hash of the text of the command).

Finally, although SCons is a Python tool, and Stata 16+ has a Python environment, several hurdles need to be overcome to use SCons from inside Stata. First, Stata sets a handler for several program "signals" for Python at startup. Later, SCons temporarily sets new signal handlers. When SCons is done, though, it tries to reinstate the original handlers in a way that does not deal correctly with the handlers Stata set from outside Python. We therefore patch the SCons function that reinstates the signal handlers to deal with this case and avoid a crash. Second, Stata changes the `stdout` and `stderr` communication channels for the Python process at startup. Later, Scons temporarily changes these, but does not restore them correctly when finished. We patch this restore function so that output can continue to be displayed in the Stata window. Third, Stata's Python environment is kept intact between Python script calls. SCons, however, is typically only able to be called from a clean environment. We therefore remove references to SCons modules before starting another run.

4.6 Extensions, Alternative Approaches and Limitations

4.6.1 Extensions

There are several ways to add to the basic functions we have described to this point. In increasing order of difficulty, these are: use additional build tools already supplied with SCons; use built-in SCons commands; execute Python commands and utilities; add user-written SCons build tools; write a new builder. We provide a few examples of the first three, and provide references for those interested in the last two.

Additional Built-in Build Tools: Compiling LaTeX with SCons

SCons comes with a builder for LaTeX documents. As long as the user has a TeX distribution (e.g., TexLive, MiKTeX, etc.) installed and known to the system, this will work more or less out of the box. Conveniently, SCons will automatically scan the `.tex` file for dependencies, so it is not necessary to code dependencies explicitly into the SConstruct. By default, these implicit

dependencies include `.tex` files that have been `\include{}`ed or `\imported{}`ed, graphics files (`.pdf`, `.eps`, etc.) included through `\includegraphics{}`, `.bib` files, among others. The user may need to adjust some construction variables to get the document to compile in the desired way. In Appendix C, we add compiling a `.tex` file into a `pdf` to our Introductory Example.

All tools built into SCons are listed in the *Scons User Guide*

Built-in SCons Commands: "Install" (Copy) Built Files Across Directories

SCons has several built-in utilities that can be convenient tools. See Sections 11 and 12 of the *SCons User Guide* for a full list. One simple and especially useful tool, `Install()`, copies files across directories. We use `Install()` extensively in the code for this paper.[30]

First, in our Introductory Example, we use `Install()` to post our table, graph and compiled PDF to Overleaf for use in this paper. We need to tell statacons where it should copy these files, i.e., the Overleaf directory.

In general, Overleaf files will reside in `Dropbox/apps/Overleaf/ProjectName`, where in this case our ProjectName is `SJ-BuildTool`. If there is only one user, or if all users have the same path to Dropbox,[31] we can define this directly in the SConstruct. For a Windows user, this would be:

```
# define Overleaf directory
overleaf_dir = 'C:/Users/UserName/Dropbox/apps/Overleaf/SJ-BuildTool'
```

If there are multiple users with different paths to Dropbox, each path must[32] be defined in the user's own `config_local.ini` configuration file:

```
# define Overleaf directory
overleaf_dir: C:/Users/UserName/Dropbox/apps/Overleaf/SJ-BuildTool
```

[30] All code is available on our project website, and the project files included with statacons contain templates for the configuration files discussed below, with extensive examples.

[31] For example through a symbolic link.

[32] Adept Python coders may be able to program their way around this in the SConstruct by accessing the operating system environment. As we note in the Conclusion, we encourage users who develop enhancements to `statacons` to make them available publicly, and we provide a project Wiki to host.

and then, in the SConstruct, we read in this variable:

```
# read in overleaf directory from config_local.ini
overleaf_dir = env['CONFIG']['Project']['overleaf_dir']
```

Having obtained the target location, we can add our `Install()` command to the SConstruct:

```
# transfer files to overleaf
env.Install(overleaf_dir, [cmd_analysis, pdf_output])
env.Alias('post_to_overleaf', overleaf_dir)
```

`Install()` takes two arguments: the first is the directory to which the files should be copied; the second is the list of files to be copied. Here, the first argument is the path to the Overleaf folder, as defined in the `overleaf_dir` variable. The second argument, `[cmd_analysis, pdf_output]`, tells SCons to copy the targets of the tasks `cmd_analysis` and `pdf_output`. Finally, in the last line, we use `Alias()` to give this task a convenient name. Now we can post our output files to overleaf with

```
statacons post_to_overleaf
```

Our second application of `Install()` is in our Applied Example.[33] There are two components to this project, dataprep and analysis. We want the outputs of dataprep to become inputs for analysis. A single user working alone could simply write code in analysis to use output files from dataprep directly, or could write a single `Install()` task in the dataprep SConstruct to push from `dataprep/outputs` to `analysis/inputs`, or in the analysis SConstruct to pull from `dataprep/outputs` to `analysis/inputs`. However, these strategies may not work well in a collaborative project, since a user in analysis would not want datasets to be changed without warning, and a user in dataprep would not want work in progress to be pulled into analysis. At the cost of slightly increasing complication, we set up a more deliberate workflow.

First, we add a `transferDataprep` directory on the same level as the `dataprep` and `analysis` directories.

[33] Code posted at `https://github.com/bquistorff/statacons/raw/main/examples/appliedExample.zip`

Second, we add an `Install()` task to the dataprep SConstruct that the user can select to push finished outputs from `dataprep/outputs` to `transferDataprep`.

```
# post outputs to ../transferDataPrep folder
env.Install('../transferDataPrep', outputs)
env.Alias('postOutputs', '../transferDataPrep')
```

To avoid transferring incomplete work, we want the `postOutputs` task to run only when it is explicitly called (`statacons postOutputs` at the Stata prompt or `scons postOutputs` in a terminal). In this case, `postOutputs` does not run automatically because its "target", the `transferDataPrep` directory, is not in the `dataprep` directory – by default, SCons will look for targets only inside the SConstruct's directory and subdirectories.

However, if the target directory is in `dataprep` or its subdirectories, we will need to remove `postOutputs` from the default targets. There are two ways to do this. The first is to set the `Default()` function explicitly and not include the transfer task among the defaults. This may be cumbersome if there are many targets. The second way is to use the `Ignore()` function in the SConstruct to tell SCons not to build `postOutputs` unless explicitly called from the command line. The code in the SConstruct would be

```
# post outputs to ./transferDataPrep folder
env.Install('./transferDataPrep', outputs)
env.Alias('postOutputs', './transferDataPrep')
# since ./transferDataPrep is a subdirectory, need
#  to Ignore postoutputs so it is not built by default
Ignore(postOutputs,'./transferDataPrep')
```

Third and finally, we set up a similar task in the analysis SConstruct to pull data from `transferDataprep` into `analysis/inputs`. In this example, we demonstrate the use of Python utilities, here `shutil.copytree`:[34]

```
# get inputs from folder "judgesExample/transferDataPrep"
# relative path from judgesExample/analysis is ../transferDataPrep
```

[34]The `dirs_exist_ok` option of `shutil.copytree` requires Python 3.8 or higher.

```
# need to specify alias pullInputs at command line
def copy_inputs(src = "../transferDataPrep", dst="inputs",
                **kwargs):
    import shutil
    shutil.copytree(src, dst, dirs_exist_ok=1)
env.AlwaysBuild(env.Alias("pullInputs", action=copy_inputs))
```

While we have made the `transferDataprep` directory local in this example, it could be a shared folder such as on Dropbox or Google Drive. Each user would just have to specify the right path in their `config_local.ini` configuration file. See Appendix B on collaborative workflows for additional comments.

Python Commands – Download and Unzip Archives

It is straightforward to use Python utilities in SCons. As an example, the data for our Applied Example are stored on Dropbox in a zip file. In `dataprep/SConstruct`, we include instructions to download and unzip the `.zip` file using Python utilities.

```
def url_download(fname = "judgesExampleInputs.zip",
    url="https://dl.dropboxusercontent.com/s/abc123/
    judgesExampleInputs.zip",
      **kwargs):
    import urllib.request
    urllib.request.urlretrieve(url, fname)
env.AlwaysBuild(env.Alias("dl_orig_data", action=url_download))

env.AlwaysBuild(env.Alias("unzip_inputs",
    action="python -m zipfile -e judgeExampleInputs.zip .")
)
```

Since these actions do not have targets, their aliases must be specified in the call to `statacons`, e.g.,

```
statacons dl_original_data
statacons unzip_inputs
```

Other types of extensions

There are many other ways to add to SCons. We list a few resources here. First, there are several user-written Tools implemented as Python packages.[35] Second, the user can write Commands, Builders or Tools. See the *SCons User guide* for documentation.[36] A nice step-by-step example is provided by Wichman (2021).

4.6.2 Alternative approaches

Alternative build tools

We chose to work in SCons for two main reasons. First, because it is a Python tool we could integrate it into Stata without requiring the use of a shell. Second, it provides the fundamental desirable features of build systems, as articulated in Mokhov et al. (2018): it is "minimal", in that it only rebuilds when dependencies have changed, and only runs each task once, and it provides "early-cutoff optimization," knowing when a newly built object has not changed and then ceasing to rebuild downstream targets. There are many alternatives to SCons; in this section we list several and discuss what we see as the advantages and disadvantages relative to SCons.

The best-known build tool is `make`. SCons has a few advantages relative to `make`. First, `make` rebuilds targets when any dependency's timestamp is newer than the target, which leads to unnecessary rebuilds if a dependency's content has not changed.[37] Second, `make` uses a custom syntax that is typically difficult for beginners (it uses many special characters and is unforgiving of rules for which whitespace characters to use). Third, `make` requires the use of a Unix shell (or emulator like Terminal on Macintosh or Windows Subsystem for Linux) to be used interactively (using `shell make` in Stata will not show you the output). One advantage of `make` over Scons is that it is somewhat easier to code straightforward commands into a script

[35] See `https://github.com/SCons/scons/wiki/ToolsIndex` for a list.

[36] As of Version 4.3.0 (December 2021), the relevant sections are 17, "Extending SCons: Writing Your Own Builder", and 18, "Not Writing a Builder: the `Command` Builder."

[37] SCons can replicate this behavior if desired, see the `Decider()` function mentioned in Section 4.3.3.

without having to create a build environment. A second advantage is the `makefile2graph` utility, which produces graphs of dependency trees. We are not aware of a similar utility for SCons.

There are many other build tools, such as `cmake`, some of which have support for using content signatures instead of file timestamps, but most are not in Python and so can not be used interactively inside of Stata. Among Python-based build tools, we chose to work in `scons` because its widespread use means there are active discussion and support communities.

The user-written `project` command in Stata (Picard 2013) is an ingenious pure Stata approach: dependencies are encoded into a project's do-files and the `project` program maintains a database of signatures to decide which outputs to rebuild. We prefer working in SCons for a few reasons, including integration with other tools, configurability, and access to advanced options such as the cache and parallel jobs, and that it does not require making changes to existing do-files.

Literate programming

A more popular approach to reproducible research in Stata has followed the *literate programming* paradigm, in which text and code are combined in a single document. There are several excellent options in this paradigm, including built-in Stata commands such as `dyndoc` and `putdocx`, "notebooks" like the Jupyter notebooks popular in Python and now accessible in Stata 17, and user-written commands like `texdoc` (Jann 2016), `webdoc` (Jann 2017), `markdoc` (Haghish 2016), and `markstat` (Rodríguez 2017).

The literate programming approach is appealing and is certainly useful in many cases.[38] A literate programming script does provide a degree of self-documentation since input and output file names are present in the same file as the text. However, literate programming has limitations similar to those of a a master do-file: as projects grow more complex and computationally intensive, either all analyses must be repeated every time the document is

[38] For example, we used `texdoc` with `statacons` to produce Section 4.2 of this paper, `dyndoc` with `statacons` to produce the companion tutorial website, and `markdoc` to produce our help files.

produced, or the user must make some manual decisions about which parts to refresh and which to skip. See our discussion of *separation of concerns* in Section 4.4. Fortunately, a build tool such as `statacons` can complement a literate programming tool by automating some of these decisions. For example, our web tutorial consists of several web pages, each produced by a corresponding `dyndoc` file. We have written an SConstruct to manage this process, so that we rebuild a web page if and only if the inputs have changed. This code is available on our Wiki page.[39]

4.6.3 Limitations

The main limitation we have found in working with SCons is that SCons only recognizes builds it has performed itself. That is, SCons determines which targets need to be rebuilt by comparing the current state of the project to its state after the last time a project was built *by SCons,* as recorded in the `sconsign.dblite` database.[40]

As an example, suppose that in our Introductory Example, the user has edited `analysis.do` to change an axis title in the target `scatterplot.pdf`. The user runs her code from the Stata command window in the usual way (`do code/analysis.do`) and is satisfied with the result. In reality, the targets `scatterplot.pdf` and `regressionTable.tex` are now up to date. However, SCons will not realize this – SCons will compute a signature of `analysis.do`, see that it has changed since its signature was last recorded in `sconsign.dblite`, and believe that the targets need to be rebuilt.

In `make`, if the user knows that a build is up-to-date, the `--touch` option will change targets' timestamps to the present. Since `make` rebuilds only when dependencies are *newer* than the target, `make` will no longer rebuild the target.

SCons does not have a similar option, so we have added the `assume_built` and `assume_done` options to `statacons`, with which the user can designate certain targets that do not

[39] `https://github.com/bquistorff/statacons/wiki/Statacons-and-literate-programming`.
[40] This applies for all settings of the `Decider()` function, including `timestamp-newer`.

need to be rebuilt or do-files that do not need to be re-run. If the already built targets all have newer timestamps, then our provided "content-timestamp-newer" `Decider()` can also be used. See the respective entries in Section 4.3.2.

A second, related limitation is that setting up a collaborative project, for example keeping code under version control (e.g., Git) and sharing files through a service like Dropbox, requires a bit of care. Our preferred approach is a shared SCons *cache*. This is a powerful tool provided by SCons but there is a learning curve.

Finally, as mentioned in Section 4.6.2, it is easier to implement one-off tasks with `make` than with SCons.[41] We try to reduce this limitation by providing a few examples that users can adapt, but some familiarity with Python coding and some trial-and-error will likely be required to extend these examples to other tasks.

4.7 Conclusion

In this paper, we have provided introductions to build tools and `statacons`, a build tool for Stata we have adapted from SCons. `statacons` runs directly in Stata without an external shell, can be used with a minimum of initial configuration, and does not require modification of existing code. While there is a startup cost to integrating `statacons` into a project, and it requires some discipline to keep SConstructs up to date, the benefits – reliable reproducibility and saved computation time – are considerable.

We recommend that users start small: while retrofitting a large, existing project may prove frustrating for new users, using `statacons` and a build-tool mindset from the beginning of a new project can promote better code by encouraging breaking down tasks into parts, simplifying the scope of individual pieces of code, and separation of concerns. In debugging problems, starting small is also advised – first, run code in Stata to see if the problem is with the code rather than the build tool, then build one or a few targets at a time until the problem

[41] At least, it is easier if the user is already comfortable using a shell and shell commands.

is isolated, examine the information provided by the `debug`, `tree` and `show_config` options and the `stataconsign` function.[42]

It is our hope that this paper will encourage the use of build tools in Stata. One barrier is that there are few examples that are simple enough for the novice to understand easily but also show some of the power of the tool. We provide a few such examples and the accompanying material. We hope that other users will build on this work by providing additional annotated examples and by extending the scope of tasks that can be performed. We encourage users to share these advances publicly, and we provide a project Wiki to host such contributions.

[42]The do-file `debugging-checklist.do` included with the `statacons` package may be useful.

Chapter 5

Conclusion

In this dissertation, I analyze the impact of the increase in transmission capacity on power generation and wholesale electricity market. And, I introduce the concept of build tools and , a build tool for Stata we have adapted from SCons.

I provide an estimation of the average treatment effect of the changes in transmission capacity in the second chapter. The main findings show the heterogeneous and unclear effect of the increase in transmission capacity. However, there are several limitations to explaining these unclear findings. The third chapter strongly demonstrates that large transmission capacity expansions positively impact electricity market efficiency. The overall congestion level and associated regional electricity price differences decrease. The large expansion also affects electricity financial markets—decreasing day-ahead convergence bidding prices and volumes, while increasing financial transmission rights (FTR) volumes. Notably, it arises a shift in FTR auction volumes from forward flow FTRs to counterflow FTRs.

The major goal of new transmission construction is to reduce the congestion of electricity which can potentially increase the price of electricity regardless of generation costs in a deregulated electricity market system. Also, With weather-driven volatility increasing reliability risks, well-planned transmission expansion can strengthen the reliability and efficiency of deregulated electricity market.

Transmission investment has been increased and the importance of transmission capacity has been growing as the share of renewable energy increases. Not only for the transmission line owner but also policymakers, the accurate evaluation of the benefit is necessary to decide transmission investment because of its astronomical amount of costs, long-term construction and the difficulties in legal procedures. The results of this paper suggest that the investment on the expansion of the transmission line will be realized by the economic benefits.

About statacons,it will encourage the use of build tools in Stata. One barrier is that there are few worked examples that are simple enough for the novice to understand easily but also show some of the power of the tool. I believe that other users will build on this work by providing additional annotated examples and by extending the scope of tasks that can be performed.

REFERENCES

Ambec, S. and Crampes, C. (2012). Electricity provision with intermittent sources of energy. *Resource and Energy Economics*, 34(3):319–336.

Birge, J. R., Hortaçsu, A., Mercadal, I., and Pavlin, J. M. (2018). Limits to arbitrage in electricity markets: A case study of miso. *Energy Economics*, 75:518–533.

BITSS (2020). Guide for Advancing Computation Reproducibility in the Social Sciences. Berkeley Initiative for Transparency in the Social Sciences.

Borenstein, S., Bushnell, J., Knittel, C. R., and Wolfram, C. (2008). Inefficiencies and market power in financial arbitrage: A study of california's electricity markets. *The Journal of Industrial Economics*, 56(2):347–378.

Borenstein, S., Bushnell, J., and Stoft, S. (2000). The competitive effects of transmission capacity in a deregulated electricity industry. *The RAND Journal of Economics*, 31(2):294.

Cattaneo, M. D. and Titiunik, R. (2020). Regression discontinuity designs. page 34.

Christensen, G., Wang, Z., Paluck, E. L., Swanson, N., Birke, D. J., Miguel, E., and Littman, R. (2019). Open Science Practices are on the Rise: The State of Social Science (3S) Survey. preprint, MetaArXiv.

Davis, L. and Hausman, C. (2016). Market impacts of a nuclear power plant closure. *American Economic Journal: Applied Economics*, 8(2):92–122.

Doshi, G. and Du, X. (2021). Transmission integration and the market for congestion revenue rights. *The Energy Journal*, 42(01).

Doucet, J., Kleit, A., and Fikirdanis, S. (2013). Valuing electricity transmission: The case of alberta. *Energy Economics*, 36:396–404.

Fell, H., Kaffine, D. T., and Novan, K. (2021). Emissions, transmission, and the environmental value of renewable energy. *American Economic Journal: Economic Policy*, 13(2):241–272.

Gentzkow, M. and Shapiro, J. M. (2014). Code and Data for the Social Sciences: A Practitioner's Guide. University of Chicago mimeo.

Haghish, E. F. (2016). Markdoc: Literate Programming in Stata. *The Stata Journal*, 16(4):964–988.

Hahn, J., Todd, P., and van der Klaauw, W. (2001). Identification and estimation of treatment effects with a regression-discontinuity design. *Econometrica*, 69:201–209.

Hausman, C. and Rapson, D. S. (2018). Regression discontinuity in time: Considerations for empirical applications. *Annual Review of Economics*, page 23.

Hitaj, C. (2015). Location matters: The impact of renewable power on transmission congestion and emissions. *Energy Policy*, 86:1–16.

Hogan, W. W. (1992). Contract networks for electric power transmission. *Systems*, page 4.

Hogan, W. W. (2014). Electricity market design and efficient pricing: Applications for new england and beyond. *The Electricity Journal*, 27(7):23–49.

Hopkins, C. A. (2020). Convergence bids and market manipulation in the california electricity market. *Energy Economics*, 89:104818.

Ito, K. and Reguant, M. (2016). Sequential markets, market power, and arbitrage. *American Economic Review*, 106(7):1921–1957.

Jackson, M. (2016). Software Carpentry: Automation and Make.

Jann, B. (2005). ESTWRITE: Stata module to store estimation results on disk. Statistical Software Components, Boston College Department of Economics.

Jann, B. (2007). Making Regression Tables Simplified. *The Stata Journal*, 7(2):227–244.

Jann, B. (2016). Creating LaTeX Documents from within Stata using Texdoc. *The Stata Journal*, 16(2):245–263.

Jann, B. (2017). Creating HTML or Markdown Documents from within Stata using Webdoc. *The Stata Journal*, 17(1):3–38.

Jha, A. and Wolak, F. (2019). Can financial participants improve price discovery and efficiency in multi-settlement markets with trading costs? *NBER Working Paper*, (w25851).

Joskow, P. and Tirole, J. (2000). Transmission rights and market power on electric power networks. *The RAND Journal of Economics*, 31(3):450.

Joskow, P. and Tirole, J. (2005). Merchant transmission investment. *Journal of Industrial Economics*, 53(2):233–264.

Kiyak, C. and de Vries, A. (2017). Electricity market mechanism regarding the operational flexibility of power plants. *Modern Economy*, 08(04):567–589.

Kristiansen, T. and Rosellón, J. (2006). A merchant mechanism for electricity transmission expansion. *Journal of Regulatory Economics*, 29(2):167–193.

LaRiviere, J. and Lyu, X. (2022). Transmission constraints, intermittent renewables and welfare. *Journal of Environmental Economics and Management*, 112:102618.

Léautier, T. (2001). Transmission constraints and imperfect markets for power. *Journal of Regulatory Economics*, pages 27–54.

Ledgerwood, S. D. and Pfeifenberger, J. P. (2013). Using virtual bids to manipulate the value of financial transmission rights. *The Electricity Journal*, 26(9):9–25.

Lee, D. S. and Lemieux, T. (2010). Regression discontinuity designs in economics. *Journal of Economic Literature*, 48(2):281–355.

Mercadal, I. (2022). Dynamic competition and arbitrage in electricity markets: The role of financial players. *American Economic Journal: Microeconomics*, 14(3):665–99.

Mokhov, A., Mitchell, N., and Peyton Jones, S. (2018). Build systems à la carte. *Proceedings of the ACM on Programming Languages*, 2(ICFP):1–29.

Orozco, V., Bontemps, C., Maigné, E., Piguet, V., Hofstetter, A., Lacroix, A., Levert, F., and Rousselle, J.-M. (2020). How to Make a Pie: Reproducible Research for Empirical Economics and Econometrics. *Journal of Economic Surveys*, 34(5):1134–1169.

Picard, R. (2013). PROJECT: Stata module providing a set of tools to build and manage a Stata project. Statistical Software Components, Boston College Department of Economics.

Rodríguez, G. (2017). Literate Data Analysis with Stata and Markdown. *The Stata Journal*, 17(3):600–618.

Ryan, N. (2021). The competitive effects of transmission infrastructure in the indian electricity market. *American Economic Journal: Microeconomics*, 13(2):202–242.

Shumway, C. and Wilson, R. (2021). Workplace Disruptions, Judge Caseloads, and Judge Decisions: Evidence from SSA Judicial Corps Retirements. Working Paper, forthcoming, Journal of Public Economics.

Stodden, V., Seiler, J., and Ma, Z. (2018). An empirical analysis of journal policy effectiveness for computational reproducibility. *Proceedings of the National Academy of Sciences*, 115(11):2584–2589.

Sun, L. and Abraham, S. (2021). Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. *Journal of Econometrics*, 225(2):175–199.

Thistlewaite, D. L. and Campbell, D. T. (1960). Regression-discontinuity analysis: An alternative to the ex-post facto experiment. *Journal of Educational Psychology*, 51:309–317.

Vega Yon, G. G. and Quistorff, B. (2019). parallel: A command for parallel computing. *The Stata Journal*, 19(3):667–684.

Wichman, M. (2021). ToolsForFools.

Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., Haddock, S. H. D., Huff, K. D., Mitchell, I. M., Plumbley, M. D., Waugh, B., White, E. P., and Wilson, P. (2014). Best Practices for Scientific Computing. *PLoS Biol*, 12(1):e1001745.

Wolak, F. A. (2015). Measuring the competitiveness benefits of a transmission investment policy: The case of the alberta electricity market. *Energy Policy*, 85:426–444.

Yang, Y. (2022). Electricity interconnection with intermittent renewables. *Journal of Environmental Economics and Management*, 113:102–653.

APPENDICES

Appendix A

Appendix

Table A.1: Local Linear Estimation of the Average Cost: Higher Order

	Dependent variable: the average cost($/MWh)			
In-service date	(1)	(2)	(3)	(4)
10/27/2017	0.193**	0.279***	0.562***	0.313***
	(0.03)	(0.03)	(0.03)	(0.04)
D × trend	0.009***	-0.006*	0.009***	0.002
	(0.00)	(0.00)	(0.00)	(0.00)
trend^2	-0.000***	0.000	-0.000***	-0.000
	(0.00)	(0.00)	(0.00)	(0.00)
D × trend^2	0.000***	0.000***	0.000***	0.000
	(0.00)	(0.00)	(0.00)	(0.00)
trend^3			-0.000***	-0.000
			(0.00)	(0.00)
D × trend^3			-0.000	0.000***
			(0.00)	(0.00)
2/5/2019	0.159***	-0.164	0.043	-0.155
	(0.02)	(0.09)	(0.05)	(0.08)
D × trend	0.008***	0.009**	-0.000	0.001
	(0.00)	(0.00)	(0.00)	(0.00)
trend^2	-0.000***	-0.000**	0.000**	-0.000
	(0.00)	(0.00)	(0.00)	(0.00)
D × trend^2	-0.000***	-0.000***	-0.000*	0.000*
	(0.00)	(0.00)	(0.00)	(0.00)
trend^3			0.000**	0.000
			(0.00)	(0.00)
D × trend^3			-0.000***	-0.000***
			(0.00)	(0.00)
N	731	731	731	731

Note: * p < 0.05, ** p < 0.01, *** p < 0.001. Standard error is in parentheses and clustered by the day of week. Window is ± 365 days. Fixed effects control the day of week, and month. Column (1) and (2) include a squared time trend term, and column (3) and (4) include a cubic time trend term. Column (2) and (4) have control variables fixed effects.

Local Linear Estimation of the Average Cost: Higher Order 2

	Dependent variable: the average cost($/MWh)			
In-service date	(1)	(2)	(3)	(4)
7/1/2019	-0.100**	-1.373***	-0.468***	-1.299***
	(0.02)	(0.18)	(0.03)	(0.19)
D × trend	-0.011***	-0.009*	0.001	0.000
	(0.00)	(0.00)	(0.00)	(0.00)
$trend^2$	0.000***	0.000	0.000*	0.000**
	(0.00)	(0.00)	(0.00)	(0.00)
D × $trend^2$	-0.000***	0.000**	-0.000***	-0.000***
	(0.00)	(0.00)	(0.00)	(0.00)
$trend^3$			-0.000	0.000**
			(0.00)	(0.00)
D × $trend^3$			0.000***	0.000**
			(0.00)	(0.00)
7/18/2019	-0.158**	-0.089*	-0.406***	-0.265**
	(0.03)	(0.03)	(0.05)	(0.06)
D × trend	-0.011***	-0.009*	0.003*	-0.005
	(0.00)	(0.00)	(0.00)	(0.00)
$trend^2$	0.000***	0.000	-0.000	0.000**
	(0.00)	(0.00)	(0.00)	(0.00)
D × $trend^2$	-0.000**	0.000**	-0.000***	-0.000***
	(0.00)	(0.00)	(0.00)	(0.00)
$trend^3$			-0.000**	0.000**
			(0.00)	(0.00)
D × $trend^3$			0.000***	0.000*
			(0.00)	(0.00)
N	731	731	731	731

Note: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Standard error is in parentheses and clustered by the day of week. Window is ± 365 days. Fixed effects control the day of week, and month. Column (1) and (2) include a squared time trend term, and column (3) and (4) include a cubic time trend term. Column (2) and (4) have control variables fixed effects.

Local Linear Estimation of the Average Cost: Higher Order 3

	Dependent variable: the average cost($/MWh)			
In-service date	(1)	(2)	(3)	(4)
12/15/2019	-0.366***	-0.422***	-0.556***	-0.286**
	(0.04)	(0.03)	(0.06)	(0.05)
D × trend	-0.001	0.001	-0.006**	-0.007*
	(0.00)	(0.00)	(0.00)	(0.00)
$trend^2$	-0.000***	-0.000*	0.000***	-0.000
	(0.00)	(0.00)	(0.00)	(0.00)
D × $trend^2$	0.000***	0.000***	-0.000*	0.000*
	(0.00)	(0.00)	(0.00)	(0.00)
$trend^3$			0.000***	-0.000
			(0.00)	(0.00)
D × $trend^3$			-0.000**	-0.000***
			(0.00)	(0.00)
N	731	731	731	731

Note: * p < 0.05, ** p < 0.01, *** p < 0.001. Standard error is in parentheses and clustered by the day of week. Window is ± 365 days. Fixed effects control the day of week, and month. Column (1) and (2) include a squared time trend term, and column (3) and (4) include a cubic time trend term. Column (2) and (4) have control variables fixed effects.

Appendix B

Appendix

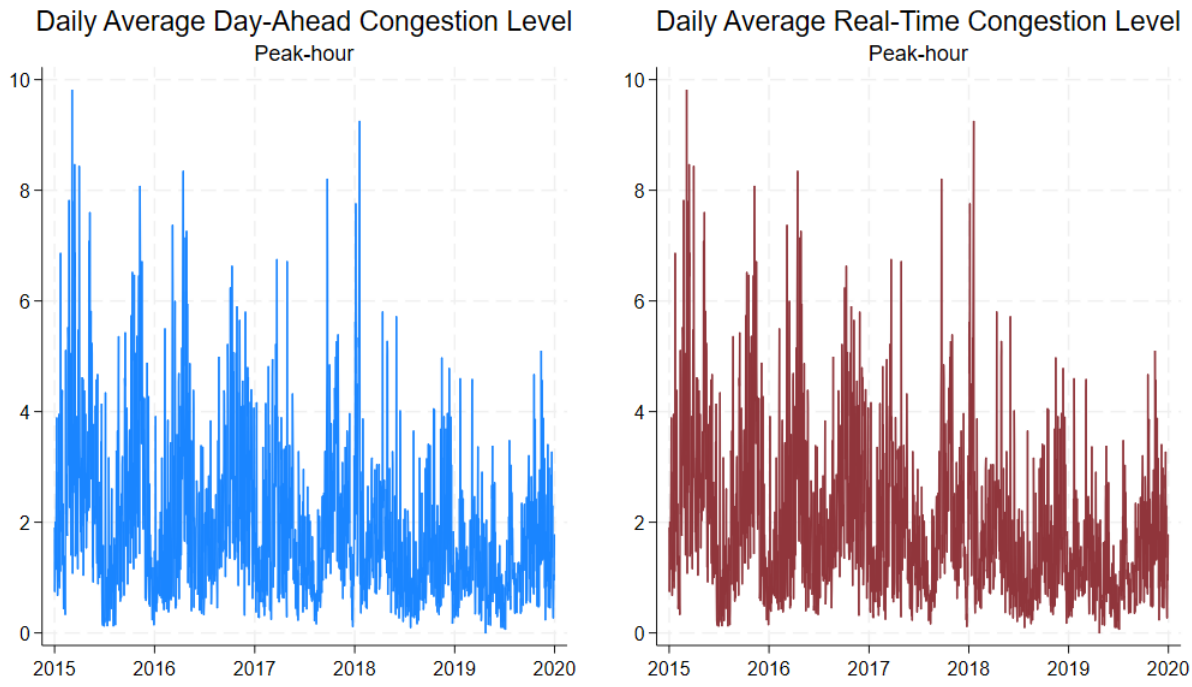
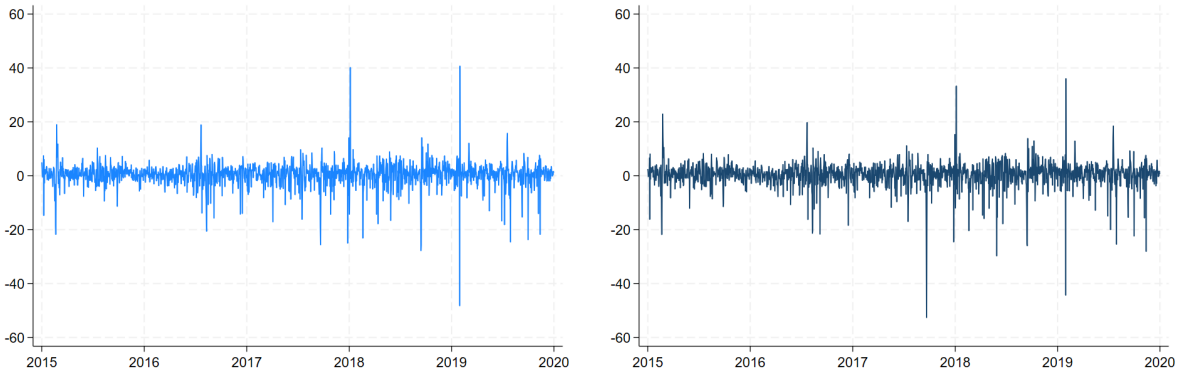


Figure B.1: The Trend of Daily Average Congestion Level at On-Peak-hour

Note: The daily averaged absolute value of marginal congestion component(MCC) across loadzone nodes in MISO North and Central region. The MCC across generation nodes show similar trends.



(a) Loadzone nodes

(b) Hub nodes

Figure B.2: The Trend of Forward Market Premium

Note: The daily simple average of forward market premium across loadzone nodes in MISO North and Central Region.

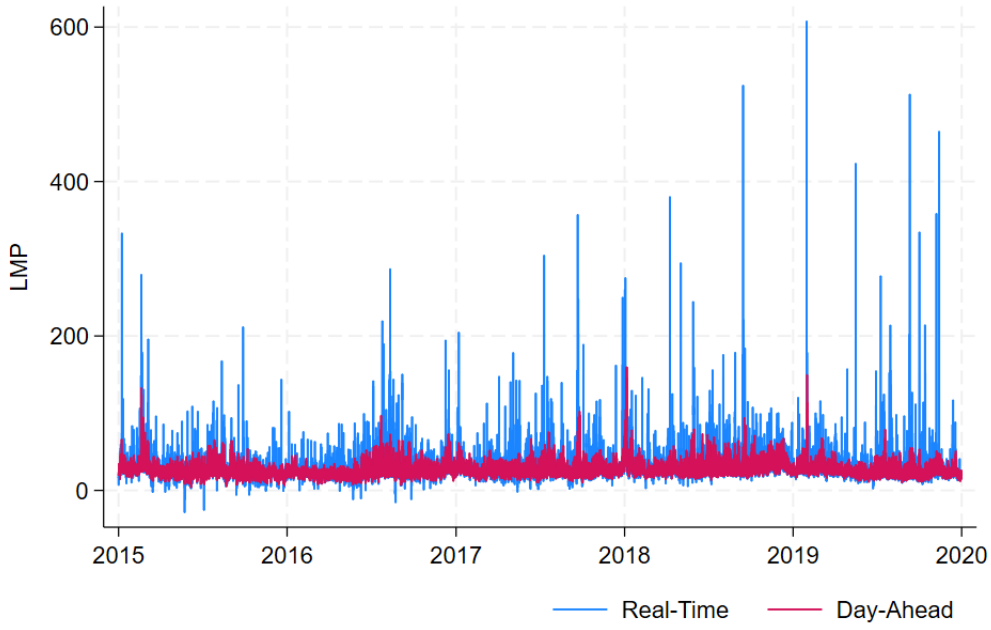
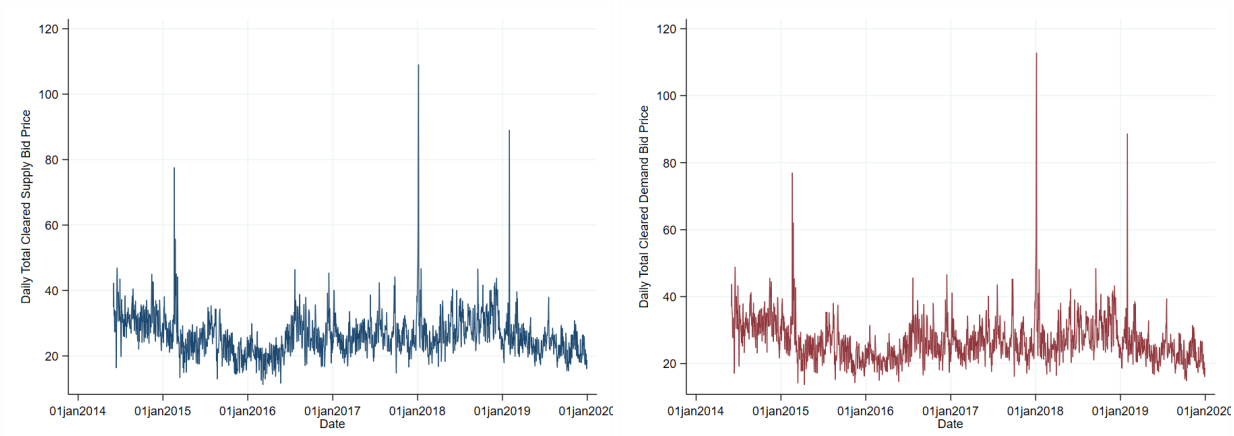


Figure B.3: Average LMP across Loadzone nodes

Note: The simple average of Day-Ahead and Real-Time Locational Marginal Price(LMP) across loadzone nodes in MISO North and Central Region.

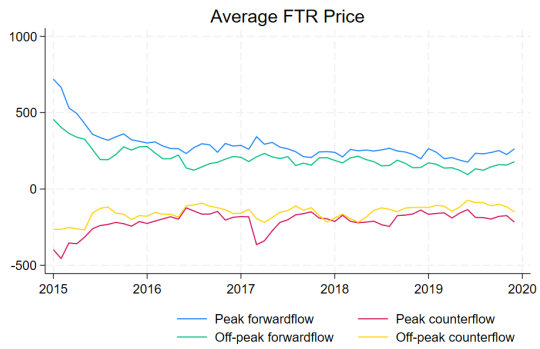


(a) INC Bids

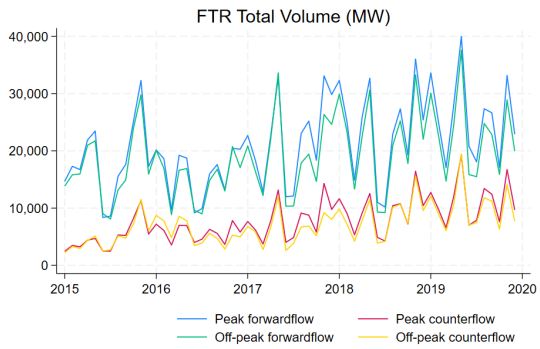
(b) DEC Bisd

Figure B.4: Daily Average Bids Cleared Price

Note: Bid price represents the market clearing price of virtual bidding at each loadzone node and each hour in the MISO Market. INC bid indicates the supply (negative) virtual bid and DEC bid indicates the (positive) demand virtual bid in virtual bid market in MISO.



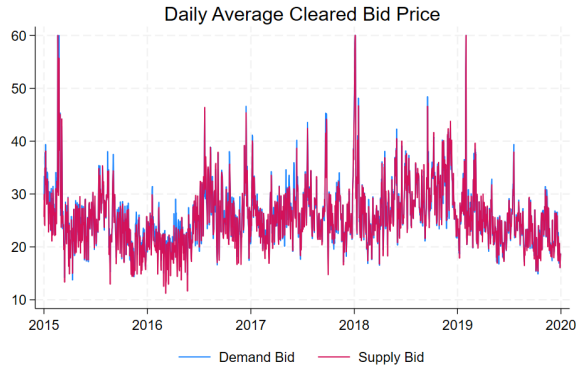
(a) FTR Price



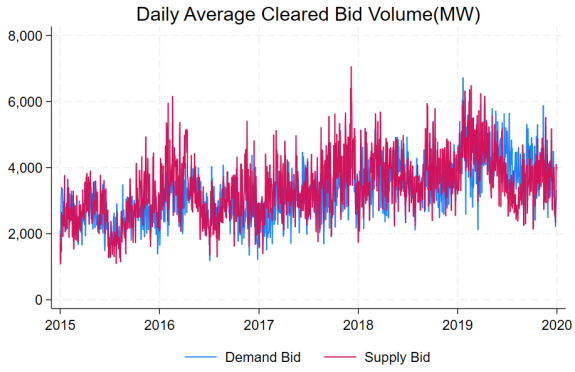
(b) FTR Volume

Figure B.5: The Trend of Monthly FTR

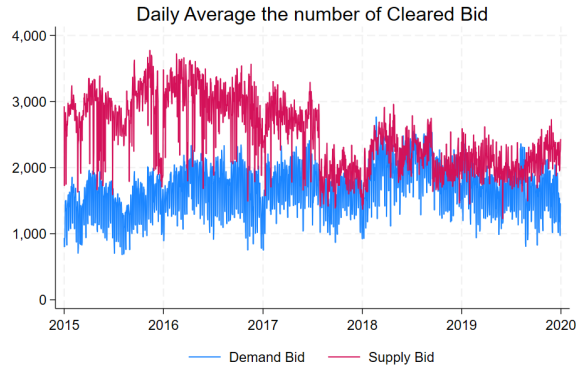
Note: Graph (a) and (b) show the average cleared price and cleared volume of monthly FTR in the MISO North and South regions. FTR auctions are divided into two-hour types: Peak-hour and Off-peak hour, which are followed by on-peak hour definition in MISO. FTR pricing has two types: Forward flow and Counterflow. Forward flow charges fees for congestion in the positive direction, while counter flow imposes charges for congestion in the negative direction.



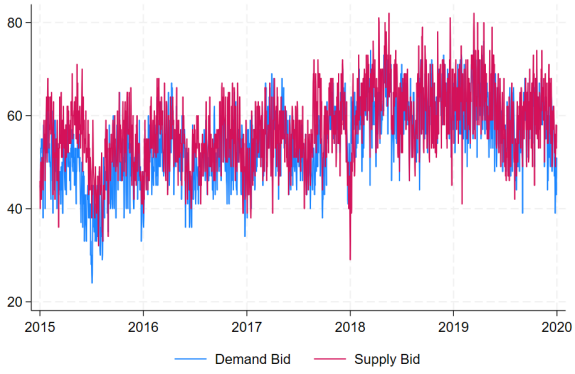
(a) Cleared Bid Price



(b) Cleared Bid Volume



(c) the number of Cleared Bid



(d) the number of Market Participants

Figure B.6: The Trend of Daily Averaged Virtual Bid

Note: Graphs indicate the daily averaged cleared price and cleared volumes of virtual bidding in the MISO North and Central regions. The red line indicates the demand bid, while the blue line indicates the supply bid.

Table B.1: OLS Regressions - Congestion Level Day-Ahead Hub Nodes

Dependent variable : Absolute value of Day-Ahead MCC					
	Hubs Ave.	INDIANA HUB	MINN HUB	MICH HUB	ILLINOIS HUB
	(1)	(2)	(3)	(4)	(5)
Trans capacity(Scaled)	-1.853***	-0.691***	-0.936***	-0.403***	0.169***
	(0.12)	(0.05)	(0.05)	(0.04)	(0.03)
Load (Gw)	0.426***	0.108***	0.139***	0.101***	0.078***
	(0.01)	(0.00)	(0.01)	(0.00)	(0.00)
Wind Gen(Gw)	0.993***	0.177***	0.549***	0.108***	0.159***
	(0.02)	(0.01)	(0.01)	(0.01)	(0.01)
Constant	-18.549***	-4.802***	-7.036***	-3.807***	-2.894***
	(0.72)	(0.24)	(0.28)	(0.21)	(0.14)
Fixed effects	Yes	Yes	Yes	Yes	Yes
R-squared	0.185	0.131	0.241	0.087	0.108
Observation	43,824	43,800	43,800	43,800	43,800

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Hub prices are the simple average of the LMPs across connected Nodes. MISO has four hub nodes in North and Central region, INDIANA, MINN, MICHIGAN, and ILLINOIS Hubs. Column (1), Hub Ave., indicates the simple average of the absolute value of day-ahead marginal congestion cost across four hubs. Fixed effects: Year, month, day of week, hour dummies, Load: Electricity demand in MISO North and Central (GWh), Wind Gen: Wind generation (GWh), Sample period: 01/01/2015 - 12/31/2019.

Table B.2: OLS Regressions - Congestion Level Real-time Hub nodes

Dependent variable : Absolute value of Real-Time MCC					
	Hubs Total	INDIANA HUB	MINN HUB	MICH HUB	ILLINOIS HUB
	(1)	(2)	(3)	(4)	(5)
Trans capacity(Scaled)	-1.782***	-1.250***	-0.688***	-0.118	0.273*
	(0.38)	(0.12)	(0.11)	(0.14)	(0.11)
Load (Gw)	0.844***	0.242***	0.213***	0.209***	0.180***
	(0.04)	(0.02)	(0.01)	(0.01)	(0.01)
Wind Gen(Gw)	1.335***	0.242***	0.571***	0.155***	0.367***
	(0.05)	(0.02)	(0.02)	(0.02)	(0.02)
Constant	-43.307***	-12.394***	-11.880***	-10.722***	-8.312***
	(2.26)	(0.83)	(0.62)	(0.74)	(0.52)
Fixed effects	Yes	Yes	Yes	Yes	Yes
R-squared	0.080	0.062	0.084	0.036	0.057
Observation	43,824	43,824	43,824	43,824	43,824

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Hub prices are the simple average of the LMPs across connected Nodes. MISO has four hub nodes in North and Central region, INDIANA, MINN, MICHIGAN, and ILLINOIS Hubs. Column (1), Hub Ave., indicates the simple average of the absolute value of day-ahead marginal congestion cost across four hubs. Fixed effects: Year, month, day of week, hour dummies, Load: Electricity demand in MISO North and Central (GWh), Wind Gen: Wind generation (GWh), Sample period: 01/01/2015 - 12/31/2019.

Table B.3: OLS Regressions - Forward Market Premium All Nodes

Dependent variable: Simple Averaged Forward Market Premium			
	Generation	Load zone	Hub nodes
	(1)	(2)	(3)
Trans capacity(Scaled)	0.440**	0.441**	0.580**
	(0.16)	(0.16)	(0.18)
Wind Gen(Gw)	0.037	0.038	0.056*
	(0.02)	(0.02)	(0.02)
Load (Gw)	-0.052**	-0.053***	-0.083***
	(0.02)	(0.02)	(0.02)
Constant	3.488***	3.472***	4.765***
	(0.85)	(0.82)	(0.96)
Fixed effects	Yes	Yes	Yes
R-squared	0.003	0.003	0.004
Observation	43,824	43,824	43,824

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Generation represents the simple average forward market premium across all generation nodes, and Load zone represents the simple average across all load zone nodes in MISO North and Central regions. Hub nodes represents the simple average forward market premium across four MISO hub nodes. Fixed effects: Year, month, day of week, hour dummies, Load: Electricity demand in MISO North and Central (GWh), Wind Gen: Wind generation (GWh), Sample period: 01/01/2015 - 12/31/2019.

Table B.4: OLS Regressions - Forward Market Premium of each Hub

Dependent variable: Forward Market Premium						
	Simple	Weighted	INDIANA	MINN	MICHIGAN	ILLINOIS
	(1)	(2)	(3)	(4)	(5)	(6)
Trans. capacity (Scaled)	0.410**	0.580**	0.804***	0.946***	0.145	0.284
	(0.14)	(0.18)	(0.21)	(0.17)	(0.24)	(0.19)
Load (Gw)	-0.024	-0.083***	-0.120***	-0.020	-0.154***	-0.002
	(0.01)	(0.02)	(0.03)	(0.02)	(0.02)	(0.02)
Wind Gen(Gw)	0.062***	0.056*	-0.005	-0.015	0.102***	0.161***
	(0.02)	(0.02)	(0.03)	(0.02)	(0.03)	(0.02)
Constant	2.121**	4.765***	6.714***	1.497*	8.912***	1.108
	(0.70)	(0.96)	(1.39)	(0.74)	(1.37)	(1.03)
Fixed effects	Yes	Yes	Yes	Yes	Yes	Yes
R-squared	0.005	0.004	0.005	0.002	0.006	0.003
Observation	43,824	43,824	43,824	43,824	43,824	43,824

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Column 1: Simple average forward market premium, Column 2: Weighted average forward market premium, Columns 3-6: Forward market premium for INDIANA, MINN, MICHIGAN, and ILLINOIS Hubs Fixed effects: Year, month, day of week, hour dummies, Load: Electricity demand in MISO North and Central (GWh), Wind Gen: Wind generation (GWh), Sample period: 01/01/2015 - 12/31/2019.

Table B.5: OLS Regressions - Forward Market Premium Generation Nodes

	(1)	(2)	(3)	(4)	(5)	(6)
Trans. line capacity	0.024***	-0.004**	-0.013***	-0.295***	1.344**	0.183*
	(0.01)	(0.00)	(0.00)	(0.07)	(0.43)	(0.08)
Load (Gw)	-0.001	0.002***	0.008***	0.260***	-0.709***	0.085***
	(0.00)	(0.00)	(0.00)	(0.01)	(0.05)	(0.01)
Wind Gen(Gw)	-0.000	-0.001***	-0.005***	-0.063***	0.233***	0.013
	(0.00)	(0.00)	(0.00)	(0.01)	(0.05)	(0.01)
Constant	0.653***	-0.088***	-0.349***	-10.815***	33.613***	-3.681***
	(0.02)	(0.01)	(0.01)	(0.52)	(2.35)	(0.42)
Fixed effects	Yes	Yes	Yes	Yes	Yes	Yes
R-squared	0.041	0.022	0.090	0.227	0.109	0.024
Observation	43,824	43,824	43,824	28,971	14,853	43,422

Note: Standard Errors in parentheses. * p<0.05, ** p<0.01, *** p<0.001. Column (1) shows results from a linear probability regression estimating the probability of a positive forward market premium. The dependent variable is a binary dummy equal to 1 if the forward market premium is positive, and 0 otherwise. Column (2) shows results from a linear probability regression estimating the probability of an extreme forward market price above 40 or below -40. Column (3) shows results from a linear probability regression estimating the probability of the real-time price being above 2 standard deviations. Column (4) shows the change in forward market premium when its value is positive, and Column (5) shows the change in forward market premium when its value is negative. Column (6) shows the change in forward market premium when its value is not extreme (above 40 or below -40).