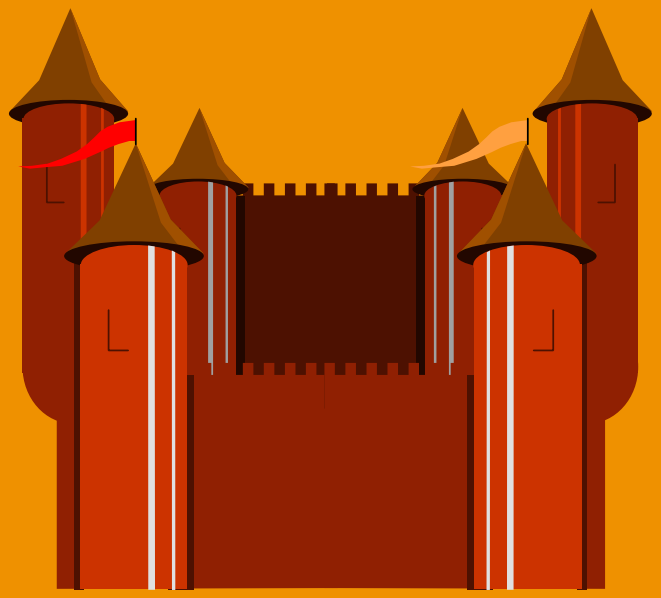
ORF 544

Stochastic Optimization and Learning

Spring, 2019



Warren Powell
Princeton University
http://www.castlelab.princeton.edu

Week 7

Uncertainty modeling

PFAs and policy search

Uncertainty modeling

Types of uncertainty

Types of uncertainty

- There are two information processes that drive the system:
  - » Decisions x_t – This is the *endogenously* controllable information process.
  - » Exogenous information - This comes from the initial state S_0, and the exogenous information process W_t.
- To figure out how to make good decisions, you need:
  - » The system model $S_{t+1} = S^M(S_t, x_t, W_{t+1})$
  - » The initial state S_0 and the exogenous information process W_t
  - » The cost/contribution function $C(S_t, x_t, W_{t+1})$

Types of uncertainty

- The initial state S_0. This contains:
  - » All deterministic parameters needed by the system. This is static data, so it is not modeled as part of the dynamic state S_t.
  - » "State of knowledge" – probabilistic information about uncertain parameters. This information is always represented as a probability distribution of some form:
    - Normally distributed uncertain parameters – This might be:
      - Estimated age of a power transformer
      - Estimated growth rate of a stock
      - The blood sugar of a patient
    - Discrete distributions – Examples include
      - Whether a patient has an infection
      - Probabilities of a discretized distribution of demand

Types of uncertainty

- The exogenous information process W_t which might include:
  - » Passive information – This is information that arrives regardless of any actions we may take. Examples:
    - Purely exogenous – Information that is not influenced by the state of the system or any actions we take. Examples:
      - – Rainfall, traffic
      - – Traffic congestion
      - – Stock prices (if we are a small player)
    - Exogenous distributions are influenced by states and/or actions.
  - » Active information – This is information we choose to collect
    - Running a laboratory experiment
    - Purchasing a report

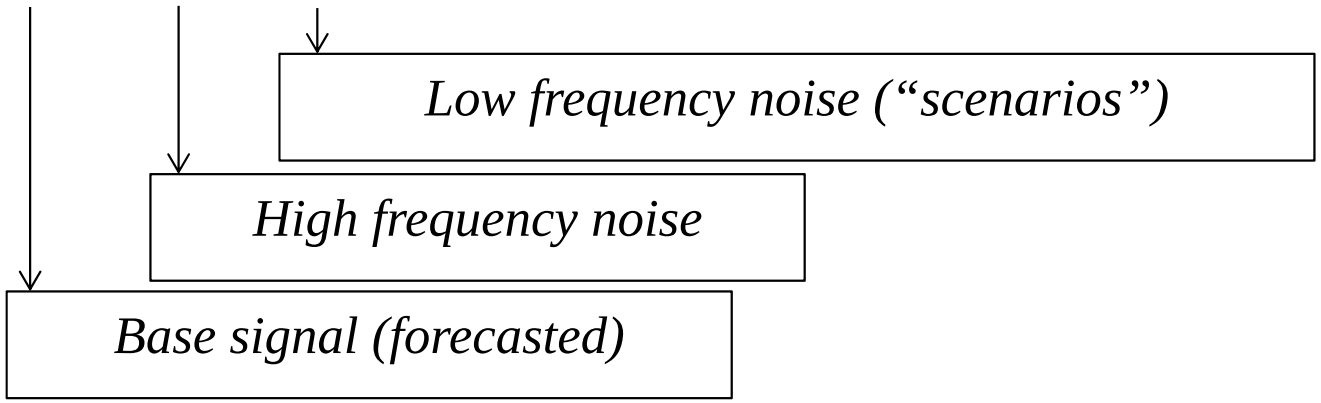
Types of uncertainty

- Observational uncertainty – Errors in our observations of the state of the system:
  - » What is the CO2 content of the atmosphere?
  - » How many people are infected with a disease?
  - » What is the state of cancer cells in a patient?
  - » How many people are employed?
  - » Where is a vehicle on a network?

Types of uncertainty

- Exogenous uncertainty – Uncertainty in the arrival of new information to the system:
  - » New demands for products
  - » Changes in prices
  - » The outcome of an experiment
  - » The results of a medical test (MRI, blood sample)
  - » Uncertainty can arise on different time scales

$$W_t = \mu_t + \varepsilon_t + \psi_t$$



Types of uncertainty

- Prognostic uncertainty – Uncertainty in the forecast of a future event.
  - » Forecasting demands
  - » Forecasting the weather
  - » Forecasting travel times
  - » Anticipating the performance of a drug on a patient.
  - » We think of prognostic uncertainty as the uncertainty in a forecast $f_{tt'}^{W}$ of $W_{t'}$ given what we know at time t.
  - » The exogenous observation of W_t can be thought of as an observation of a change in the forecast $f_{t,t+1}^{W}$ to $f_{t+1,t+1}^{W}$.

Types of uncertainty

- Inferential (or diagnostic) uncertainty – Uncertainty in the estimate of a parameter.
  - » Uncertainty in parameters estimated from observational data
  - » Statistical uncertainty in a model fit from data.
  - » Uncertainty in what disease a patient may be suffering from.
  - » Uncertainty in the state of a transformer (used in the power grid), or a piston (buried in an engine).

Types of uncertainty

- Experimental noise – This is the variability that arises when running repeated experiments (either in a lab or in the field)
  - » Testing the impact of a new flu drug.
  - » Testing the effect of a new material on battery lifetimes
  - » Evaluating the effectiveness of a new drug in laboratory experiments.
  - » Variability in the time required to traverse a path in a transportation network.

Types of uncertainty

- Model uncertainty – This is uncertainty about the "model" itself.
  - » While "model uncertainty" is widely recognized, there is some ambiguity about what is meant by a "model." This can be
    - The transition function, which captures the evolution of the state variable over time.
    - The mathematical model of the uncertainty
    - The cost/contribution function
  - » Model uncertainty can come in two forms:
    - Parameters characterizing the model
    - Uncertainty about the structure of the model:
      - – Linear approximation of a nonlinear model
      - – Different sets of equations describing the climate.

Types of uncertainty

- Transitional uncertainty – We have a model of how a (presumably) deterministic system evolves, but there is still noise.
  - » Modeling the location of an aircraft moving at a certain speed from a known location.
  - » Predicting the time of arrival of a car at a downstream node
  - » We can write this as

$$S_{t+1} = S^M(S_t, x_t) + \varepsilon_{t+1}$$

  - » The controls community will write this as

$$x_{t+1} = f(x_t, u_t) + w_t$$

  - » This is for problems where the transition function is known (e.g. from physics) but where there is exogenous inputs.

Types of uncertainty

- Control uncertainty
  - » You ask for x_t but you get $x_t + \varepsilon_t$
  - » You order 10 items, but only get 8 due to a stockout.
  - » You try to drive at 70 mph, but there is variability due to your inability to hold a speed perfectly, along with the effect of traffic.
  - » Uber sends an invitation to a driver to take a rider, but the driver may decline.
  - » Uber would like 50 drivers on duty, but can only influence their behavior through surge pricing.
  - » Wiley sets a wholesale price of $80, but Amazon sells at some random price above that (limits Wiley's ability to set prices).
  - » You order plastic balls with a diameter of 10mm, but get balls with a diameter of 12mm (or some variance around 10mm).

Types of uncertainty

- Algorithmic uncertainty
  - » Run the same algorithm twice, and you may get different answers (depends on the algorithm and the nature of the compute environment).
  - » PJM modeling
    - Models day ahead, hour ahead and real-time.
    - Introduces truncation errors.

Types of uncertainty

- Goal uncertainty
  - » We are not sure of how a truck dispatcher will trade off between the cost of getting a driver to a load, picking up the load on time, and getting a driver home.
  - » Air Liquide has to purchase large amounts of electricity from the grid. These prices are very volatile. How should they balance average costs and the risk of being exposed to very high prices?

Types of uncertainty

- Thoughts
  - » Modeling uncertainty has not received the attention it deserves in stochastic optimization.
  - » This is part of a community known as "uncertainty quantification."
  - » Stochastic modeling in an optimization setting introduces new twists:
    - How to generate sampled models where each sample contributes to the quality of the solution.
    - Are we properly handling tails and correlations.

Uncertainty modeling

Types of distributions

Types of distributions

- Modeling the flow of exogenous information:
  - » We need to generate sample paths of the sequence
    $$(W_1, W_2, \dots, W_t, \dots)$$
  - » Data driven - Using sample sequences from history
    - Examples
      - Price paths
      - Wind speeds
    - Issues
      - Limited number of sample paths available from history
      - Data reflects history; the future may be different
  - » Mathematical models
    - Requires creating mathematical models of stochastic processes
    - Challenges are to replicate:
      - Errors/volatility at a point in time
      - Correlations over time
      - Correlations over space and other attributes

Types of distributions

- Types of distributions
  - » Probability distributions come in different forms:
    - Classical "thin tailed" distributions –
      - – Exponential family
        - » Normal, exponential, gamma
        - » Uniform
      - – Discrete variants
    - Heavy-tailed distributions
      - – Cauchy distribution (may have infinite variance)
      - – "Jump diffusion" – Sum of low-variance normally distributed error, plus a high-variance error that occurs with low probability
    - Spikes
    - Bursts
    - Rare events

Types of distributions

- Fitting distributions
  - » Empirical distributions
    - Use historical data to fit pdf or cdf.
    - Use this with NORTA transformation.

Types of distributions

- Fitting distributions
  - » Moment matching
    - E.g. use mean and variance from data to match the mean and variance of a distribution.
    - Beta distribution:

$$f(x|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

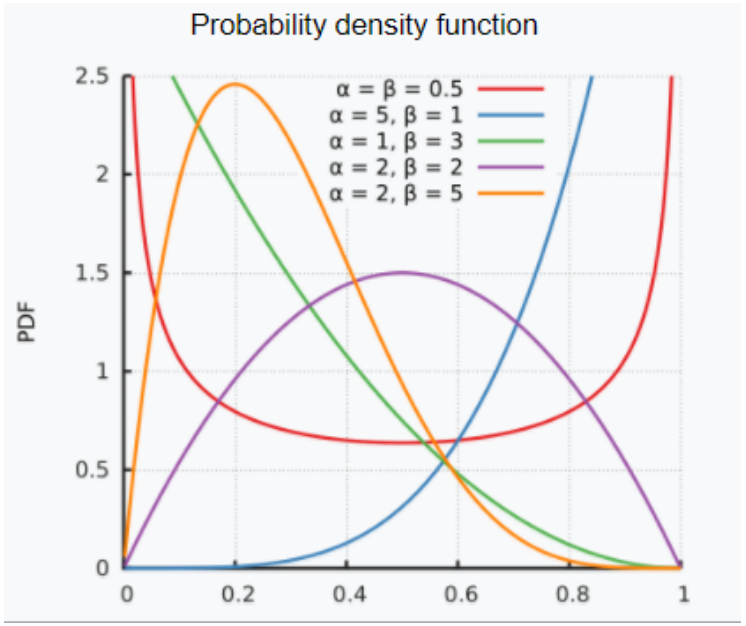
    - Mean:

$$E[X] = \frac{\alpha}{\alpha+\beta}$$

    - Variance

$$Var[X] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

    - Use these formulas to find α and β



Types of distributions

- Simulating random variables using Monte Carlo simulation

10.4.1 Generating uniform $[0, 1]$ random variables

Arguably the most powerful tool in the Monte Carlo toolbox is the ability to use the computer to generate random numbers that are uniformly distributed between 0 and 1. This is so important that most computer languages and computing environments have a built-in tool for generating uniform $[0, 1]$ random variables. While we strongly recommend using these tools, it is useful to understand how they work. It starts with a simple recursion that looks like

$$R^{n+1} \leftarrow (a + bR^n) \bmod (m),$$

where a and b are very large numbers, while m might be a number such as $2^{64} - 1$ (for a 64 bit computer), or perhaps $m = 999,999,999$. For example, we might use

$$R^{n+1} \leftarrow (593845395 + 2817593R^n) \mod (999999999).$$

This process simulates randomness because the arithmetic operation $(a + bR)$ creates a number much larger than m, which means we are taking the low order digits, which move in a very random way.

We have to initialize this with some starting variable R^0 called the *random number seed*. If we fix R^0 to some number (say, 123456), then every sequence $R^1, R^2, \ldots$ will be exactly the same (some computers use an internal clock to keep this from happening, but

Types of distributions

- Simulating random variables using Monte Carlo simulation

sometimes this is a desirable feature). If a and b are chosen carefully, R^n and R^{n+1} will appear (even under careful statistical testing) to be independent.

Due to the mod function, all the values of R^n will be between 0 and 999999999. This is convenient because it means if we divide each of them by 999999999, we get a sequence of numbers between 0 and 1. Thus, let

$$U^n = \frac{R^n}{m}.$$

While this process looks easy, we caution readers to use built-in functions for generating random variables, because they will have been carefully designed to produce the required independence properties. Every programming language comes with this function built in. For example, in Excel, the function `Rand()` will generate a random number between 0 and 1 which is both uniformly distributed over this interval, as well as being independent (a critical feature).

Types of distributions

- Correlated random variables

Now assume that we are given a covariance matrix Σ where $\Sigma_{ij} = Cov(X_i, X_j)$. Just as we use σ above (the square root of the variance σ^2), we are going to take the "square root" of Σ by taking its Cholesky decomposition, which produces an upper right-triangular matrix. In Matlab, this can be done using

$$C = \texttt{chol}(\Sigma).$$

The matrix C satisfies

$$\Sigma = CC^T,$$

which is why it is sometimes viewed as the square root of Σ.

Now assume that we generate a column vector Z of N independent, normally distributed random variables with mean 0 and variance 1. Let μ be a column vector of $\mu_1, \ldots, \mu_N$ which are the means of our vector of random variables. We can now generate a vector of N random variables X with mean μ and covariance matrix Σ using

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_N \end{pmatrix} + C \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_N \end{pmatrix}.$$

Types of distributions

- Generating correlated random variables

Assume our covariance matrix is given by

$$\Sigma = \begin{bmatrix} 9 & 3.31 & 0.1648 \\ 3.31 & 9 & 3.3109 \\ 0.1648 & 3.3109 & 9 \end{bmatrix}.$$

The Cholesky decomposition computed by MATLAB using $C = \texttt{chol}(\Sigma)$ is

$$C = \begin{bmatrix} 3 & 1.1033 & 0.0549 \\ 0 & 3 & 1.1651 \\ 0 & 0 & 3 \end{bmatrix}.$$

Imagine that we generate a vector Z of independent standard normal deviates

$$Z = \begin{bmatrix} 1.1 \\ -0.57 \\ 0.98 \end{bmatrix}.$$

Using this set of sample realizations of Z, a sample realization u would be

$$u = \begin{bmatrix} 10.7249 \\ 2.4318 \\ 7.9400 \end{bmatrix}.$$

Time series modeling

Time series modeling

8.3.1 Time series models

The time series literature is quite rich, so we are just going to illustrate a basic model which represents the price p_{t+1} as a function of the recent history of prices. For illustration, we are going to use the last three time periods, which means that we would write our model as

$$\begin{aligned} p_{t+1} &= \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon_{t+1}, \\ &= \bar{\theta}_t^T \phi_t + \varepsilon_{t+1}, \end{aligned} \tag{8.3}$$

where

$$\phi_t = \begin{pmatrix} p_t \\ p_{t-1} \\ p_{t-2} \end{pmatrix}$$

is our vector of prices. We assume that the noise $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ for a given σ_ε^2.

The vector of coefficients $\bar{\theta}_t = (\bar{\theta}_{t0}, \bar{\theta}_{t1}, \bar{\theta}_{t2})^T$ can be estimated recursively using the methods presented in chapter 3 of StOaL. Assume that we start with an initial estimate $\bar{\theta}_0$ of the vector of coefficients. We are also going to need a three-by-three matrix B^0 that for now we can assume is a scaled identity matrix (we provide a better idea below).

Time series modeling

The vector of coefficients $\bar{\theta}_t = (\bar{\theta}_{t0}, \bar{\theta}_{t1}, \bar{\theta}_{t2})^T$ can be estimated recursively using the methods presented in chapter 3 of StOaL. Assume that we start with an initial estimate $\bar{\theta}_0$ of the vector of coefficients. We are also going to need a three-by-three matrix B^0 that for now we can assume is a scaled identity matrix (we provide a better idea below).

The basic updating equation for $\bar{\theta}_t$ is given by

$$\bar{\theta}_{t+1} = \bar{\theta}_t - H_t \phi_t \varepsilon_{t+1}, \tag{8.4}$$

The error $\hat{\varepsilon}_t$ is computed using

$$\varepsilon_{t+1} = \bar{\theta}_t^T \phi_t - p_{t+1}. \tag{8.5}$$

The three-by-three matrix H_t is computed using

$$H_t = \frac{1}{\gamma_t} B_{t-1}, \tag{8.6}$$

where the matrix B_n is computed recursively using

$$B_t = B_{t-1} - \frac{1}{\gamma_t}(B_{t-1}\phi_t(\phi_t)^T B_{t-1}). \tag{8.7}$$

The variable γ_t is a scalar computed using

$$\gamma_t = 1 + (\phi_t)^T B_{t-1} \phi_t. \tag{8.8}$$

Time series modeling

- Mean reverting models
  - » Basic mean reverting model (Ornstein-Uhlenbeck process)

$$p_{t+1} = p_t + \beta\,(\bar{\mu}_t - p_t) + \varepsilon_{t+1}$$

  - » where

$$\bar{\mu}_{t+1} = (1-\alpha)\bar{\mu}_t + \alpha p_{t+1}$$

β is the rate of mean reversion. Note that if $p_t > \bar{\mu}_t$, then p_{t+1} will trend lower (and vice versa). This is what is meant by "mean reversion"

Time series modeling

- Mean reverting with jump diffusion

$$p_{t+1} = p_t + \beta\,(\bar{\mu}_t - p_t) + \varepsilon_{t+1} + J_{t+1}\varepsilon^J_{t+1}$$

  - where

$$\varepsilon_t \sim N(0,\sigma^2_\varepsilon)$$

$$J_t = \begin{cases} 1 & w.p.\ \eta \\ 0 & w.p.\ 1-\eta \end{cases}$$

$$\varepsilon^J_t \sim \lambda e^{-\lambda x}$$

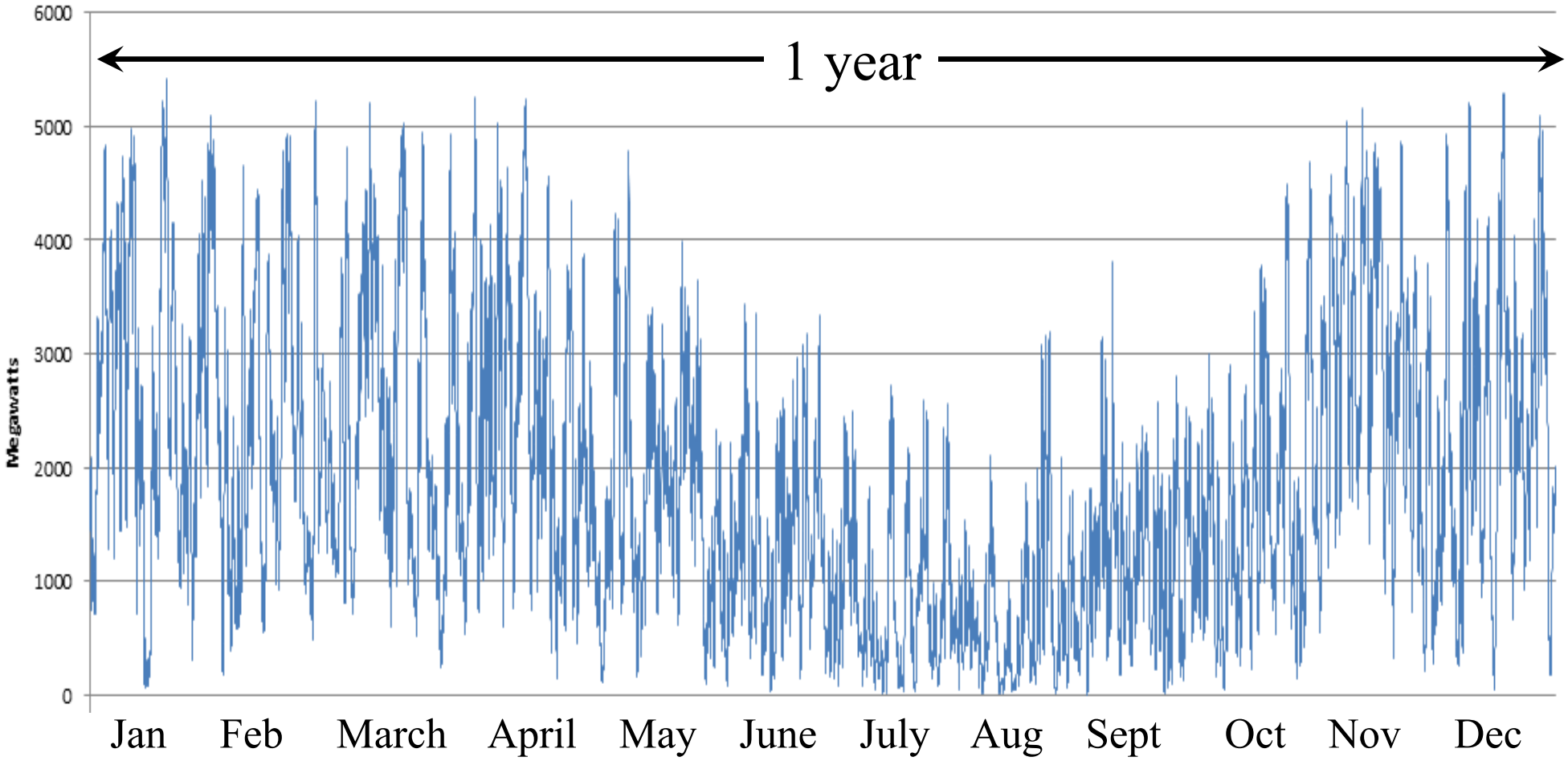
  - To estimate η, pass through data 2-3 times, discarding data that is more than $3\sigma_\varepsilon$ from the mean. Then η is the fraction of data we have discarded.
  - Fit λ so that $\frac{1}{\lambda}$ = variance of the discarded data.

NORTA

Normal to anything

Normal to anything

- Wind power from all PJM wind farms

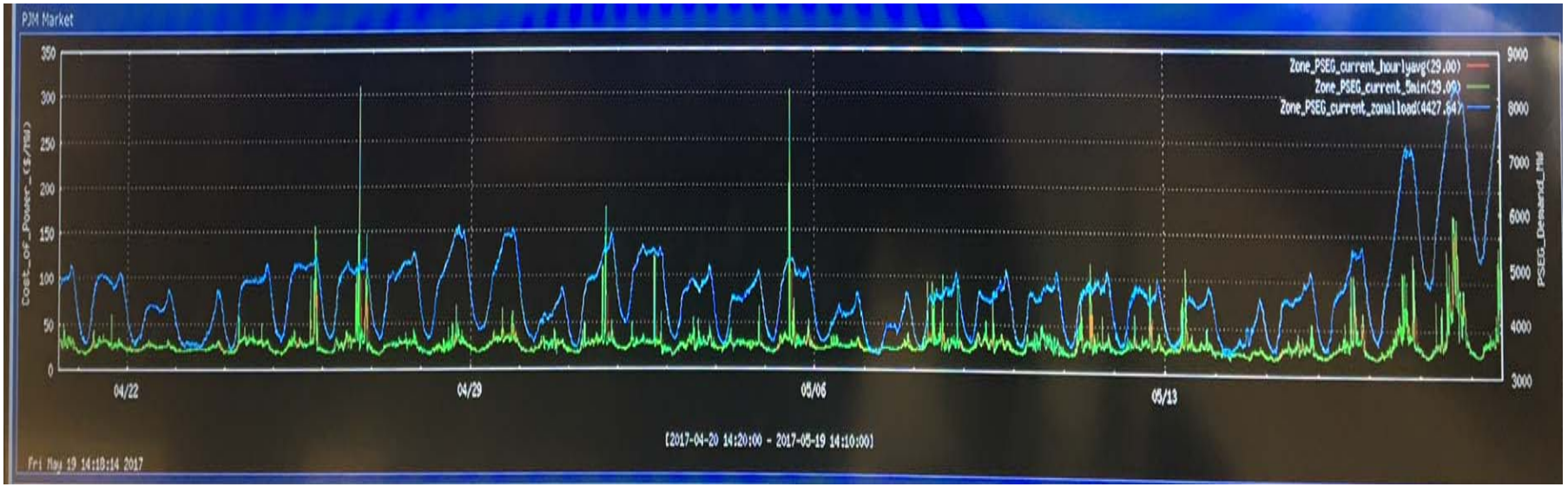


Normal to anything

- Transforming distributions
  - » There are many problems where the distribution may be hard to fit. It can be useful to transform observations into normally distributed random variables, model these transformed variables, and then transform them back.
  - » Let U be a random variable that is uniformly distributed over $[0,1]$, and let X be an arbitrary random variable with cumulative distribution $F_X(x) = Prob[X \leq x]$. Let $F_X^{-1}(u)$ be the inverse of the cumulative distribution. It is possible to show that
    - $F_X(X) \sim U$ and $F_X^{-1}(U) \sim X$

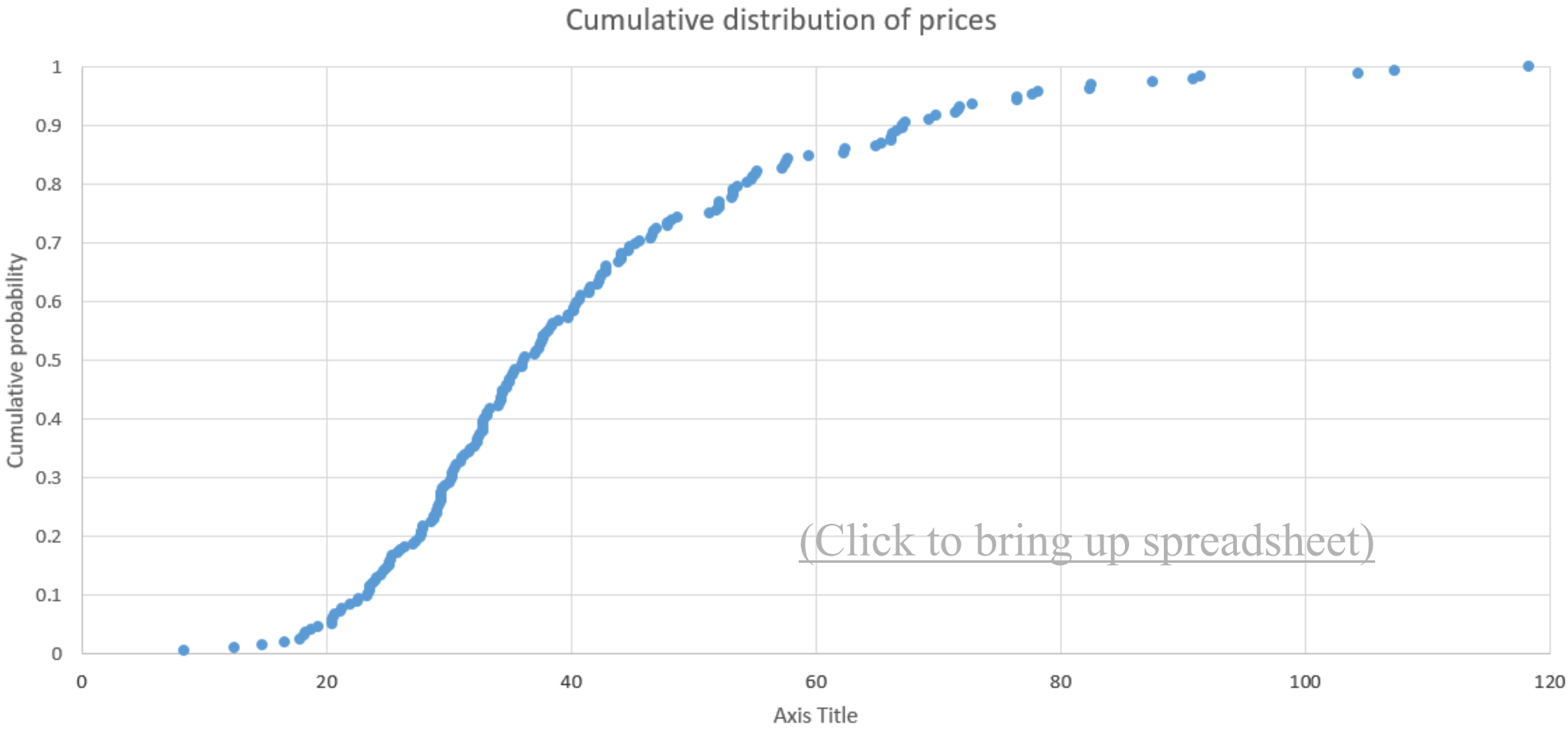
Normal to anything

- Some data can be very heavy tailed
  - » Snapshot of electricity prices for New Jersey:



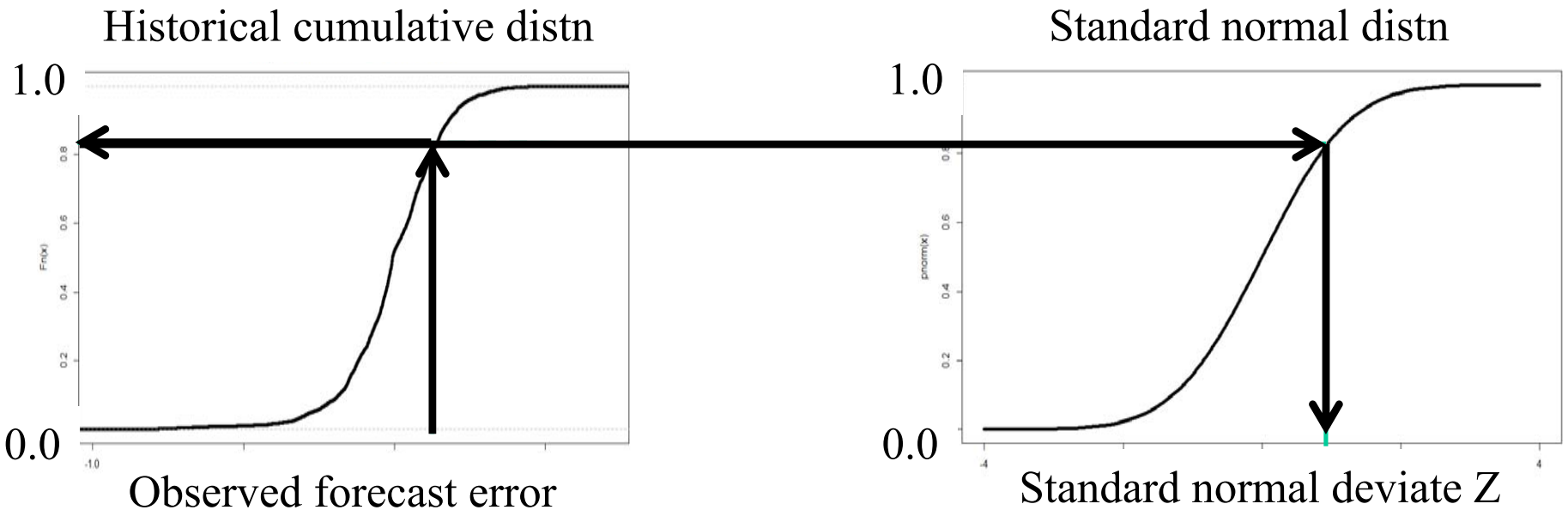
Normal to anything

- An empirical distribution



Normal to anything

- Transforming distributions
  - » Observed errors are transformed to normally distributed errors using quantile mapping:

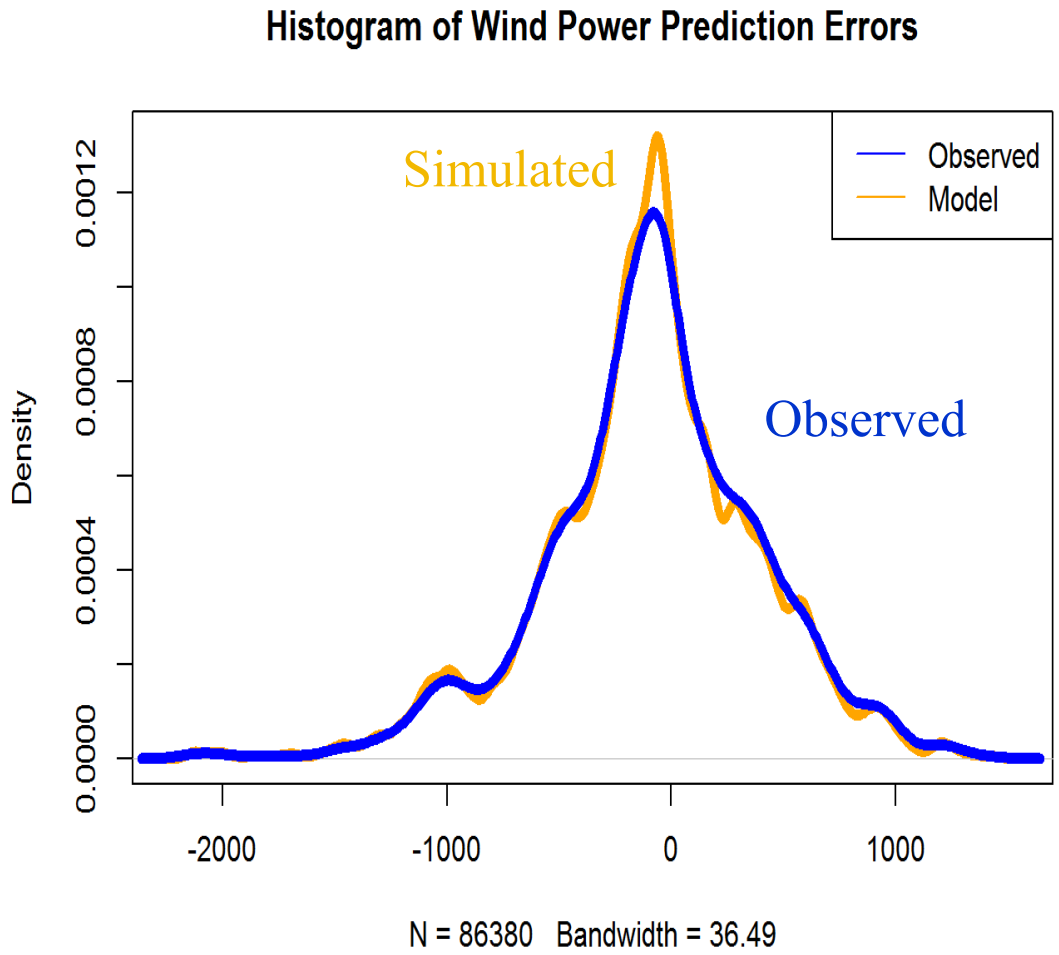


  - » Very useful for representing unusual distributions (e.g. heavy tailed, bimodal, …)

Normal to anything

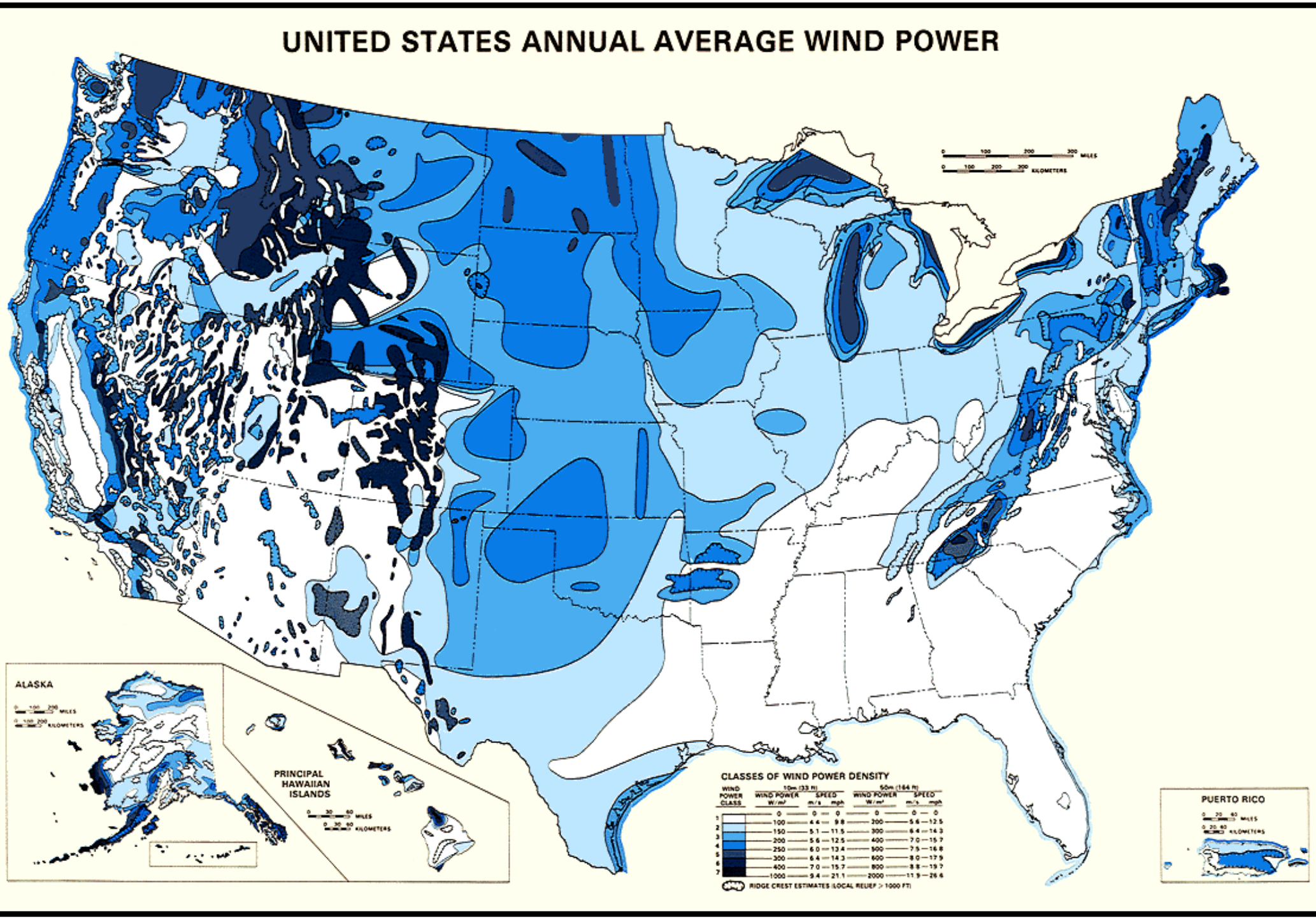
- Modeling errors

Error distribution



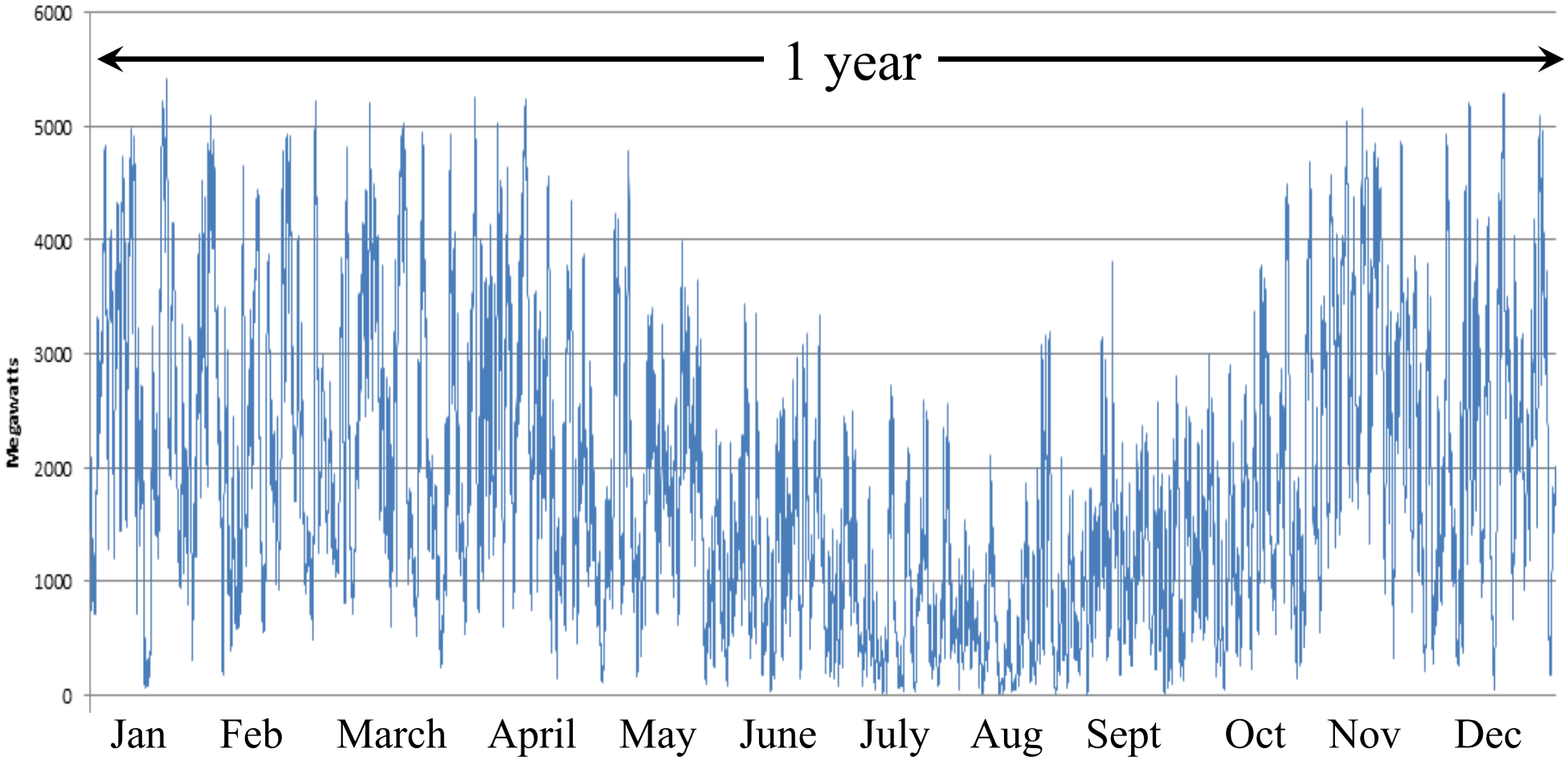
Uncertainty modeling

Case study: Modeling forecast errors for wind



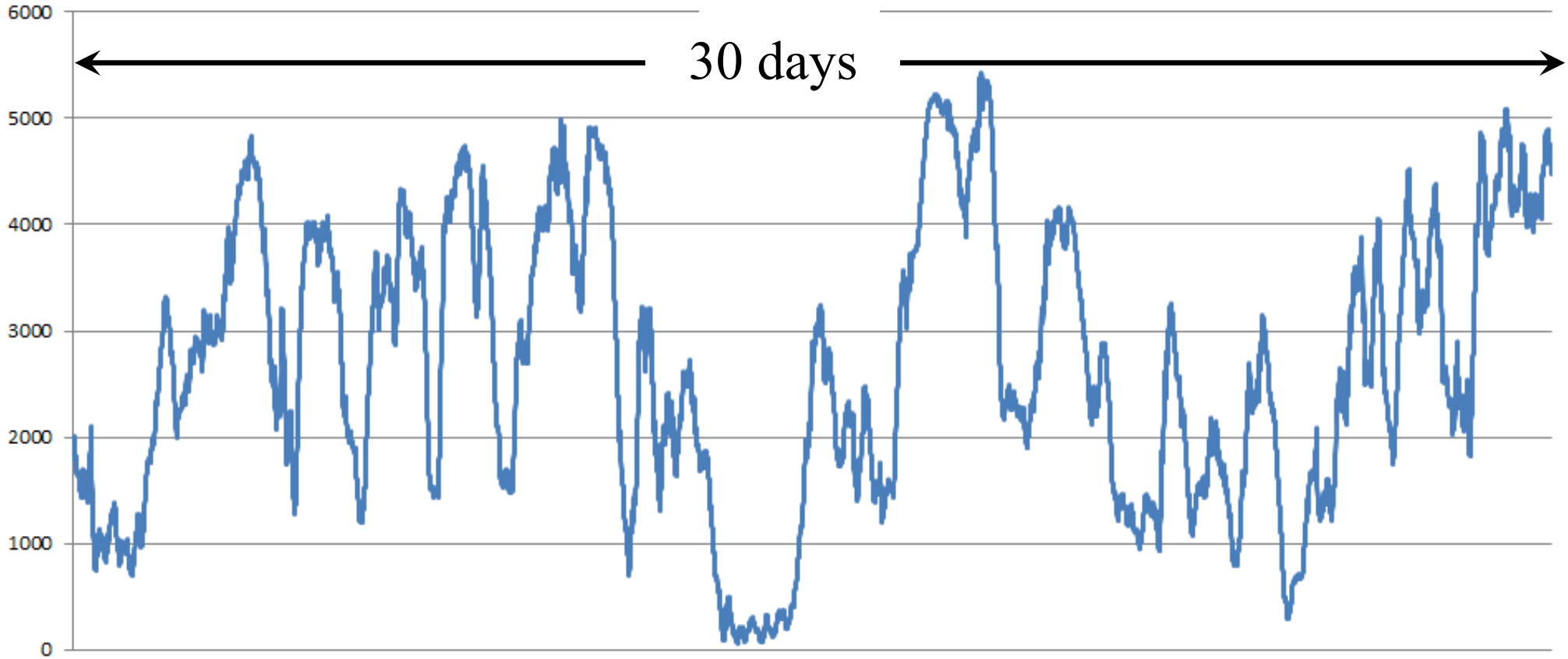
Energy from wind

- Wind power from all PJM wind farms



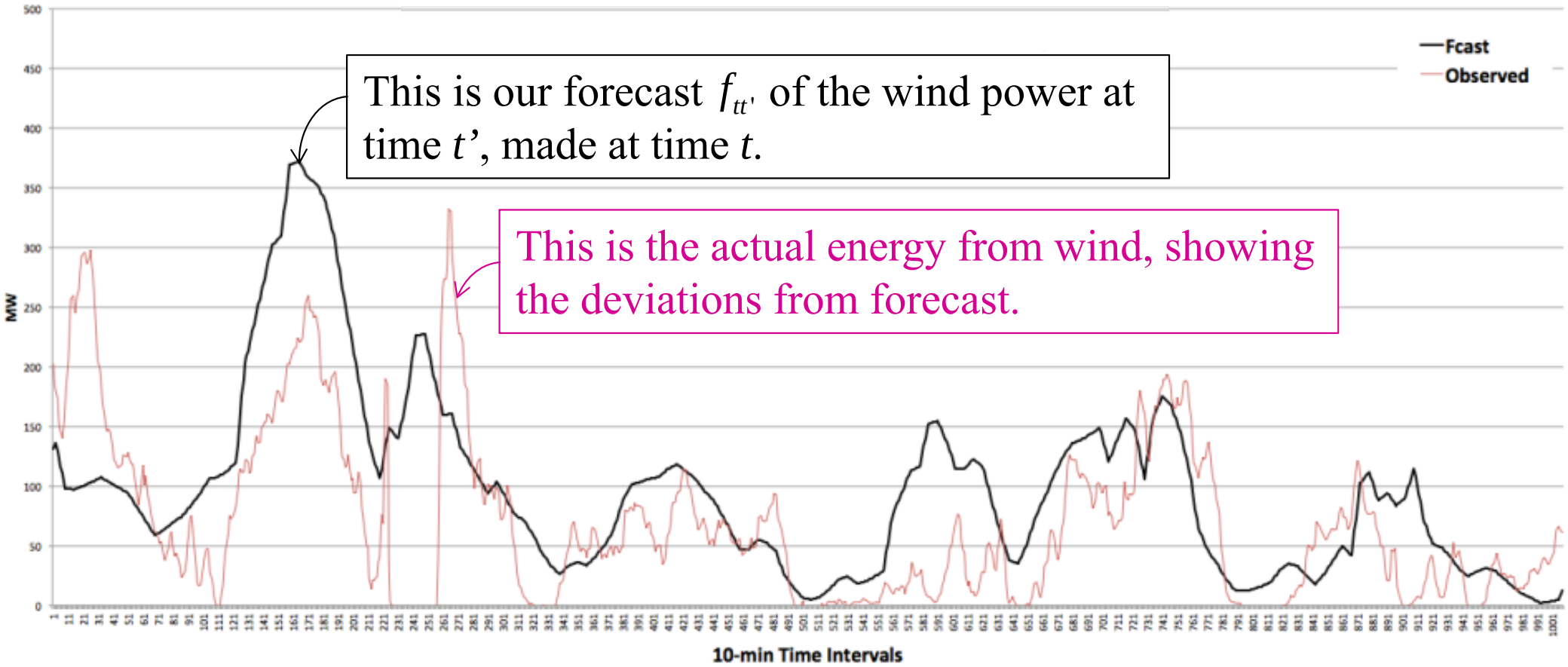
Energy from wind

- Wind power from all PJM wind farms



Energy from wind

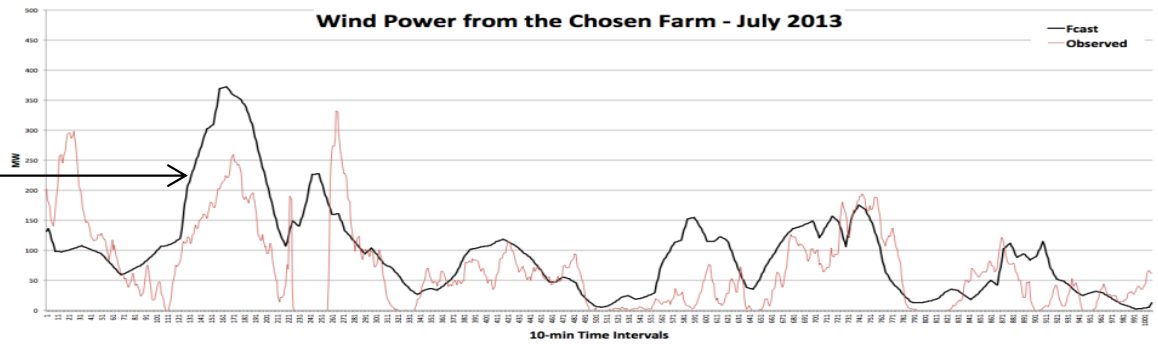
- Illustration of forecasted wind power and actual
  - » The forecast (black line) is deterministic (at time t, when the forecast was made). The actuals are stochastic.



Energy from wind

- Two types of uncertainty arise in forecasting:
  - » At time t, we have forecasts for different times into the future:

$$f_{tt'} =$$



    - The forecast is an imperfect estimate of the actual load at time t':

$$L_{t'} = f_{t,t'} + \varepsilon^{L}_{t'}$$

The actual load at time t' is "stochastic" at time t.

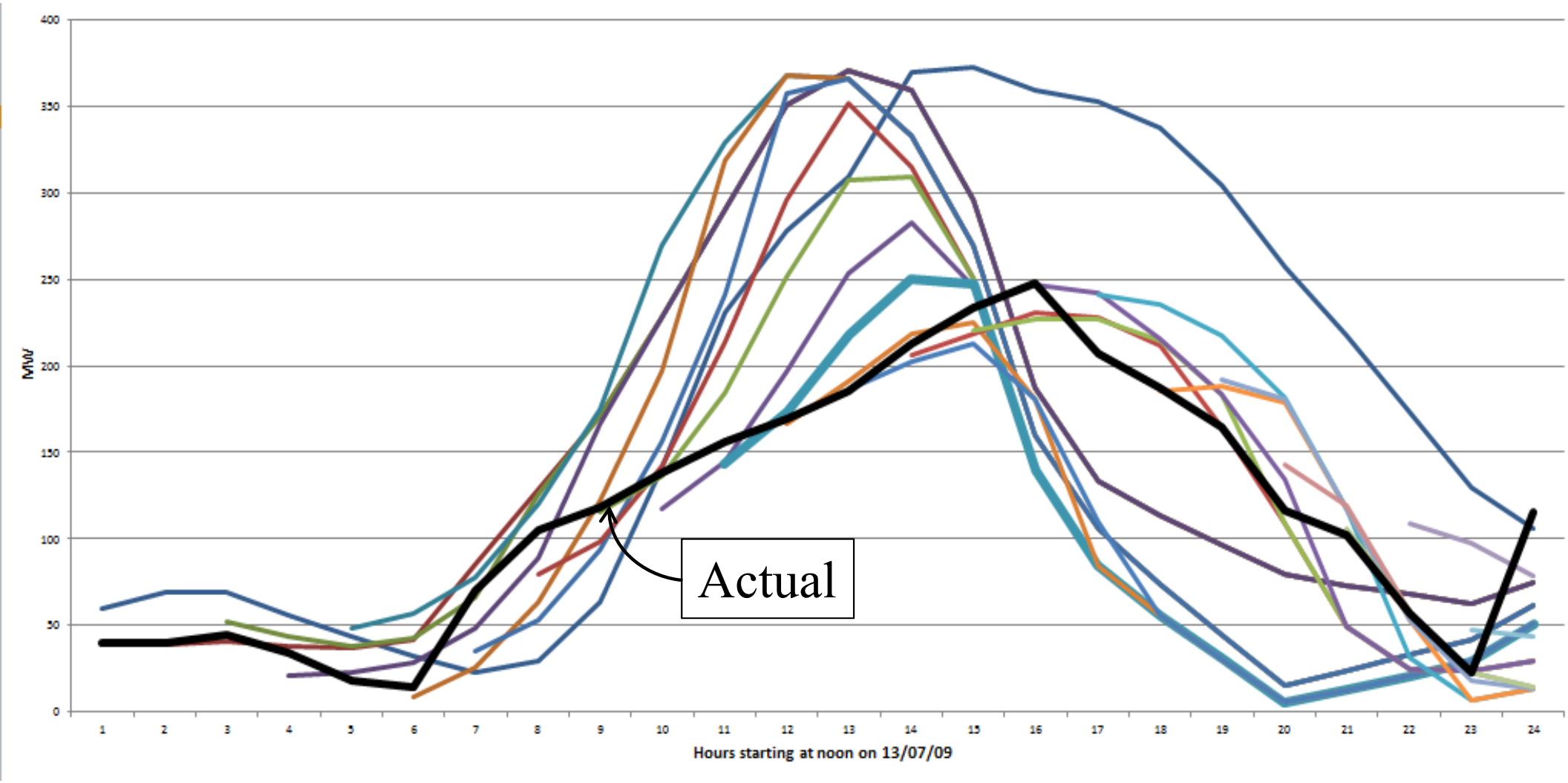
  - » As new information arrives, the forecasts themselves change from time t to t+1:

$$f_{t+1,t'} = f_{t,t'} + \varepsilon^{f}_{t+1,t'}$$

This change in the forecast is "stochastic" at time t.

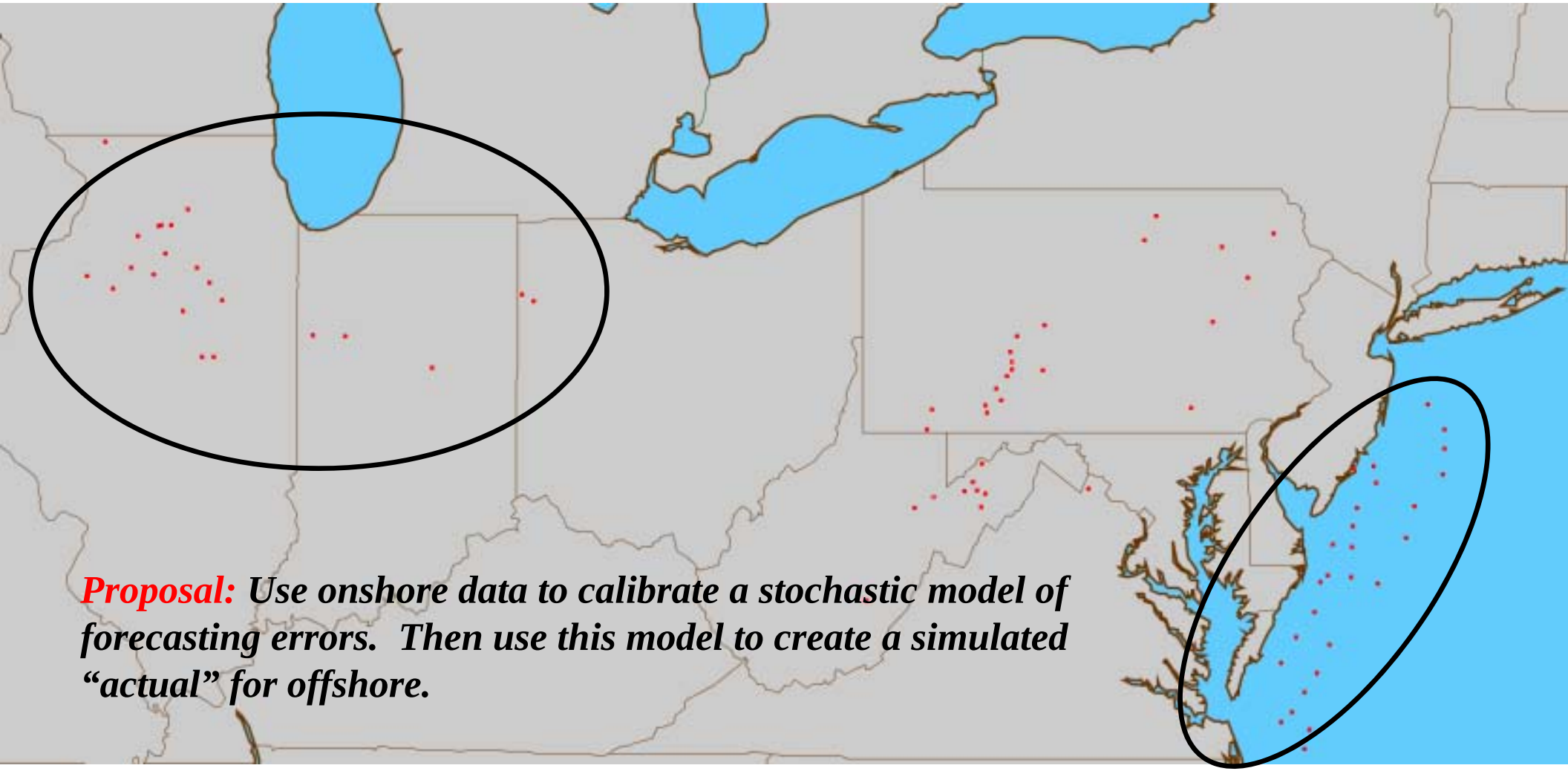
Forecasting wind

- Rolling 24-hour forecast of PJM wind farms



Onshore & offshore wind farms

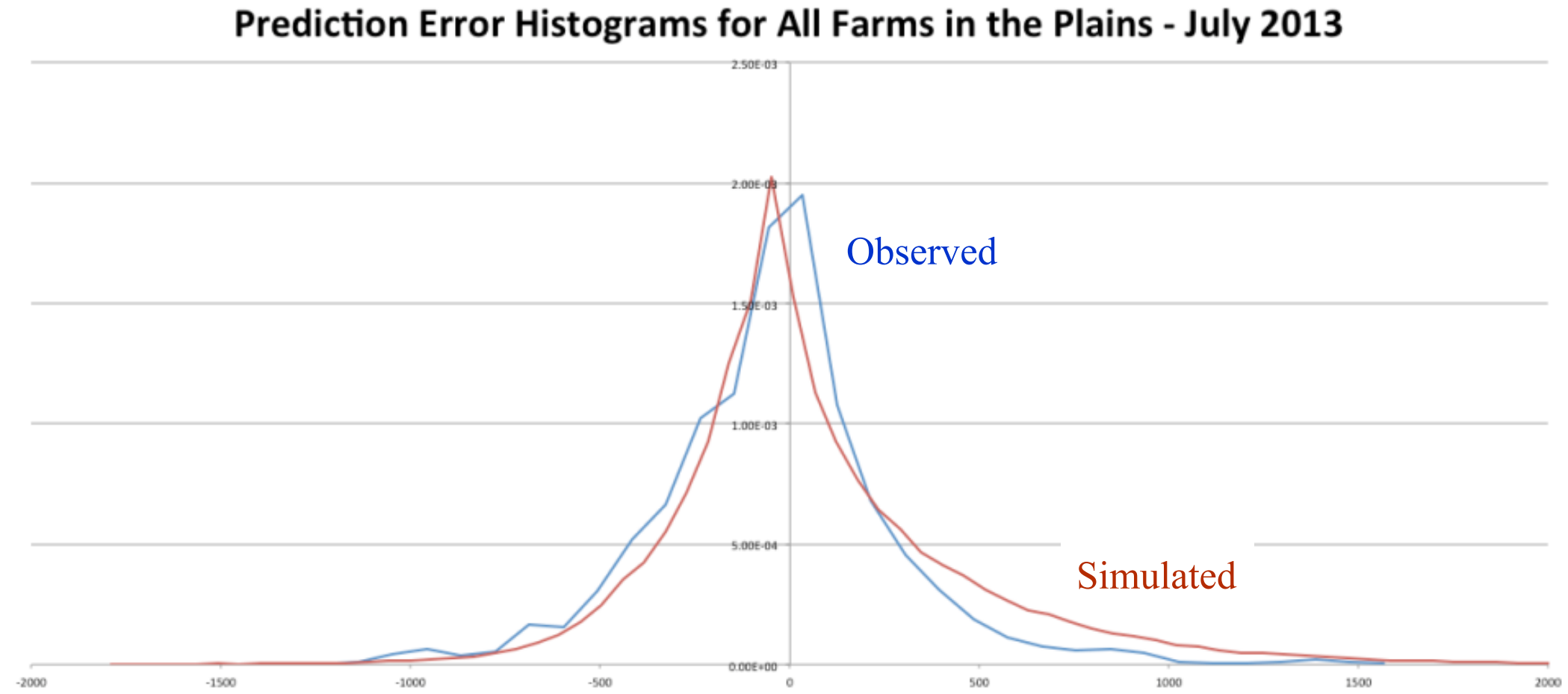
- We were given access to data on the wind power generated by onshore wind farms within PJM



Proposal: ***Use onshore data to calibrate a stochastic model of forecasting errors. Then use this model to create a simulated "actual" for offshore.***

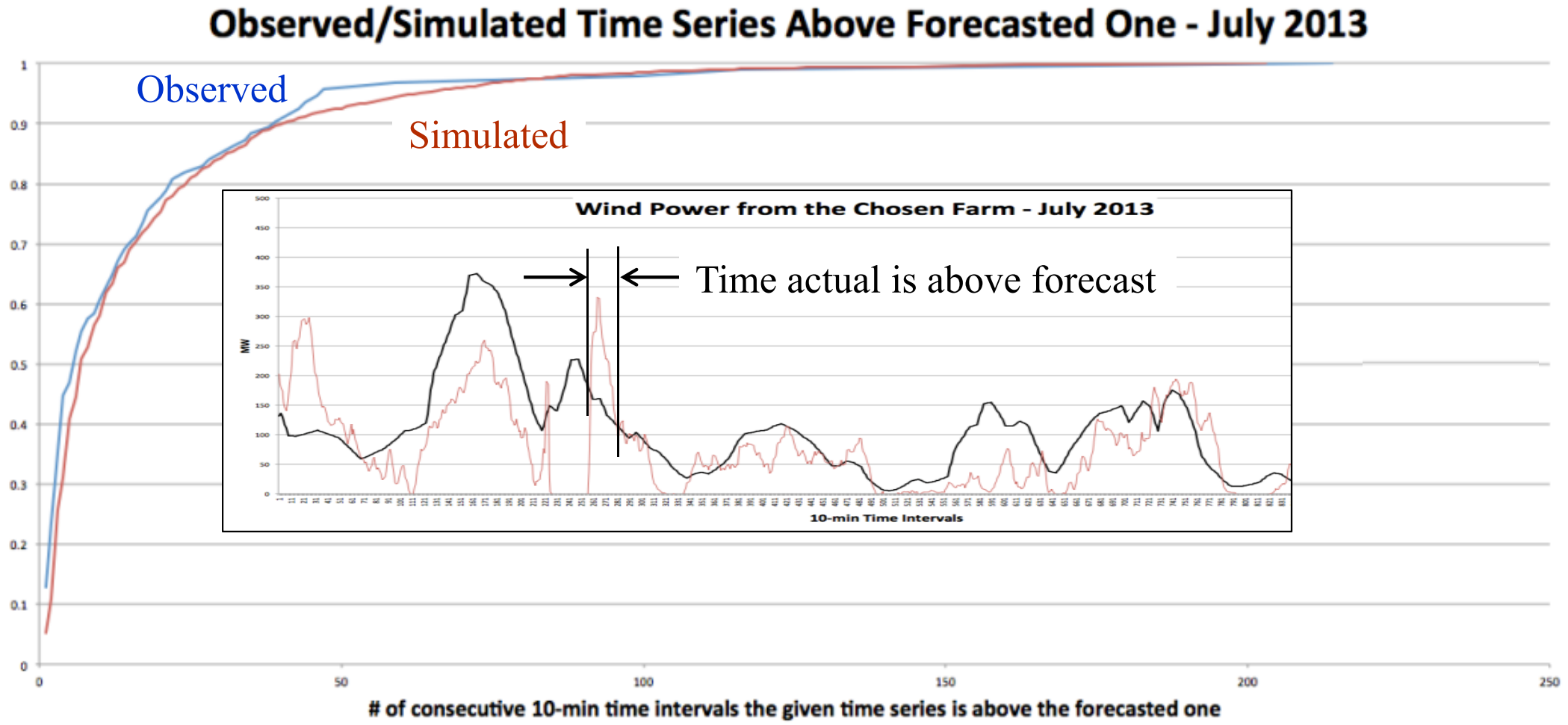
Simulating onshore wind

- Distribution of forecast errors
  - » Uses adjusted spatial correlations to improve fit.



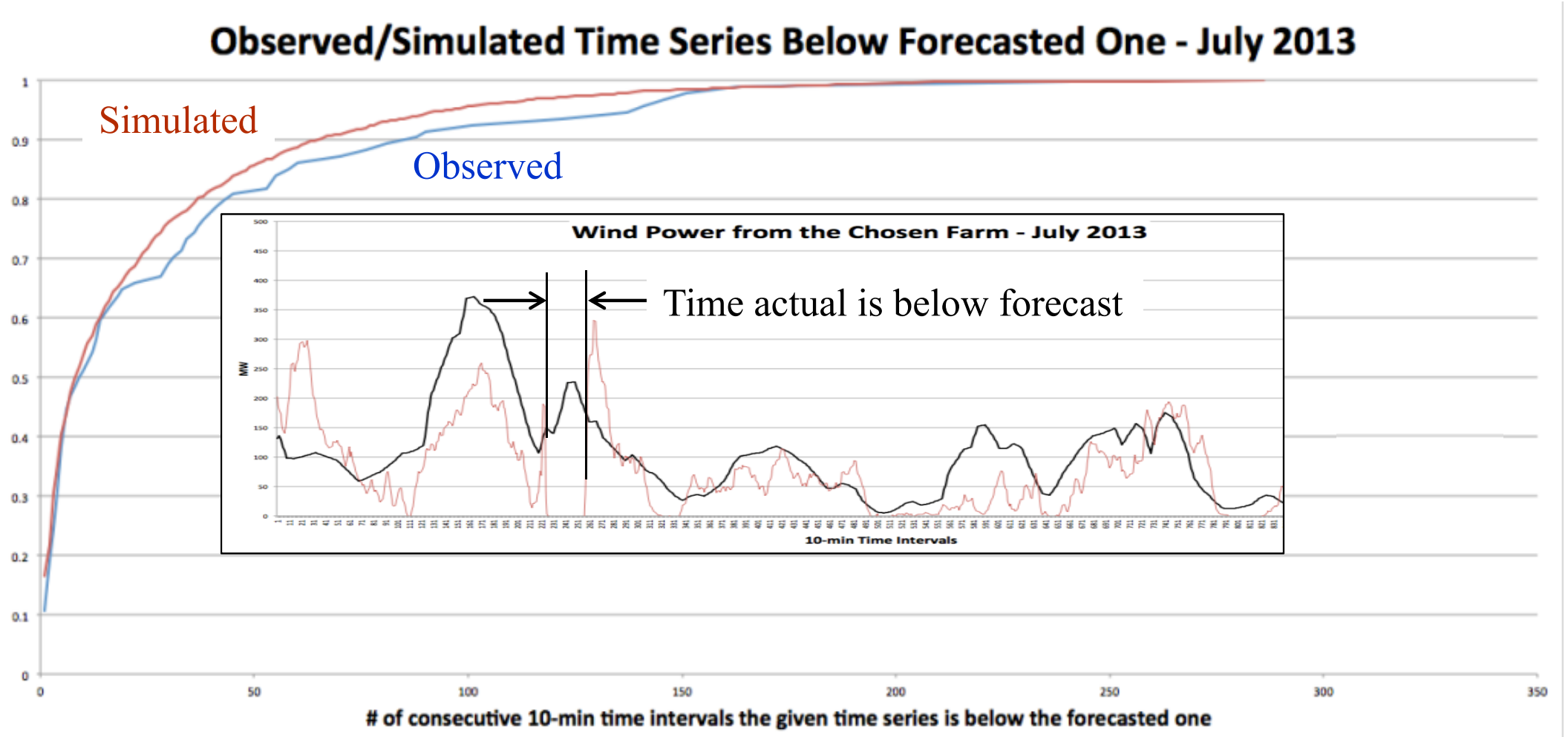
Simulating onshore wind

- Cumulative histogram of the # of consecutive time intervals the observed/simulated time series is above the forecasted one (chosen farm only):



Simulating onshore wind

- Cumulative histogram of the # of consecutive time intervals the observed/simulated time series is below the forecasted one (chosen farm only):



Simulating onshore wind

- Challenges
  - » We would like to replicate:
    - Error distribution
    - Upcrossing distribution
    - Downcrossing distribution
  - » We would like to replicate performance at two levels:
    - Each wind farm
    - All wind farms when aggregated together
- What has *not* worked:
  - » ARMA, ARIMA, GARCH, even when using quantile transformation to handle non-normality of wind errors
    - Times series models struggle to capture the longer-term behavior

Simulating onshore wind

- What seems to be working:
  - » We use a hybrid Markov chain model with two stage variables:
    - The crossing state S_t^C:

$$S_t^C = \begin{cases} A\,|\,(\text{S/M/L}) & \text{If we are in an "above the forecast" state } |\text{ S/M/L} \\ B\,|\,(\text{S/M/L}) & \text{If we are in a "below the forecast" state } |\text{ S/M/L} \end{cases}$$

A/B means above or below
S/M/L means a short, medium or long crossing distribution
$\mathbb{P}(S_{t+1}^C \,|\, S_t^C)$ is estimated from historical data.

    - The wind speed W_t given the crossing state:

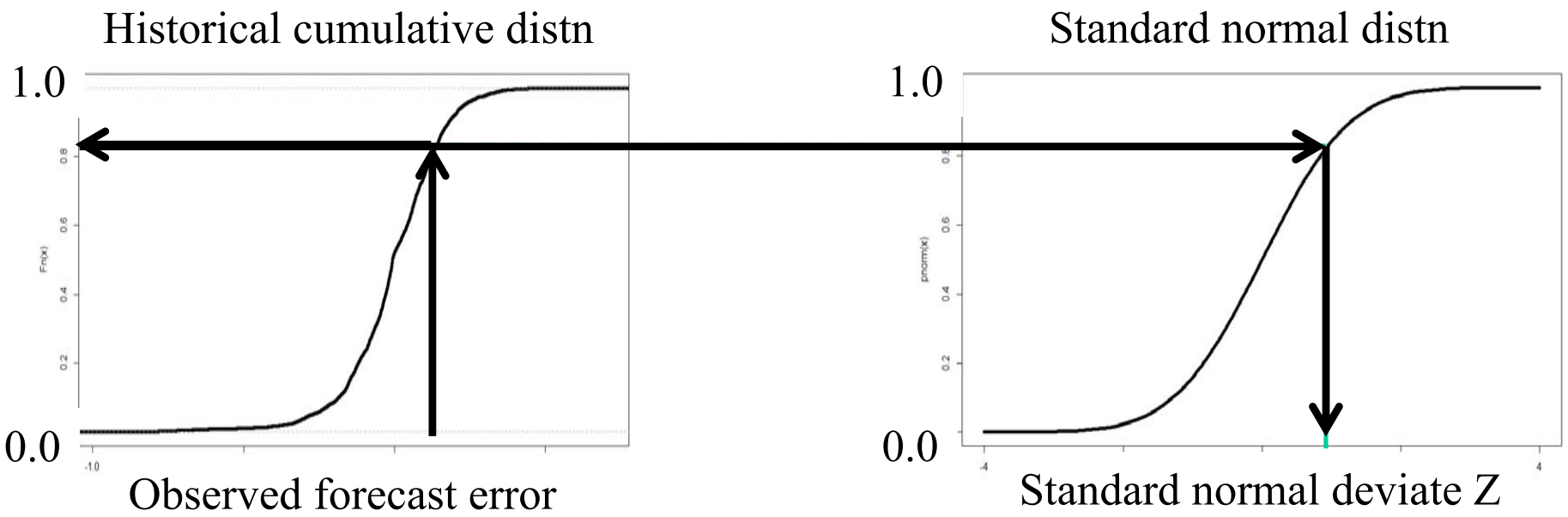
W_t = Wind speed at time t.

W_t^g = Wind speed aggregated into 5 ranges.

$\mathbb{P}(W_{t+1} \,|\, W_t^g, S_t^C)$ = Density of W_{t+1} given W_t^g and S_t^C

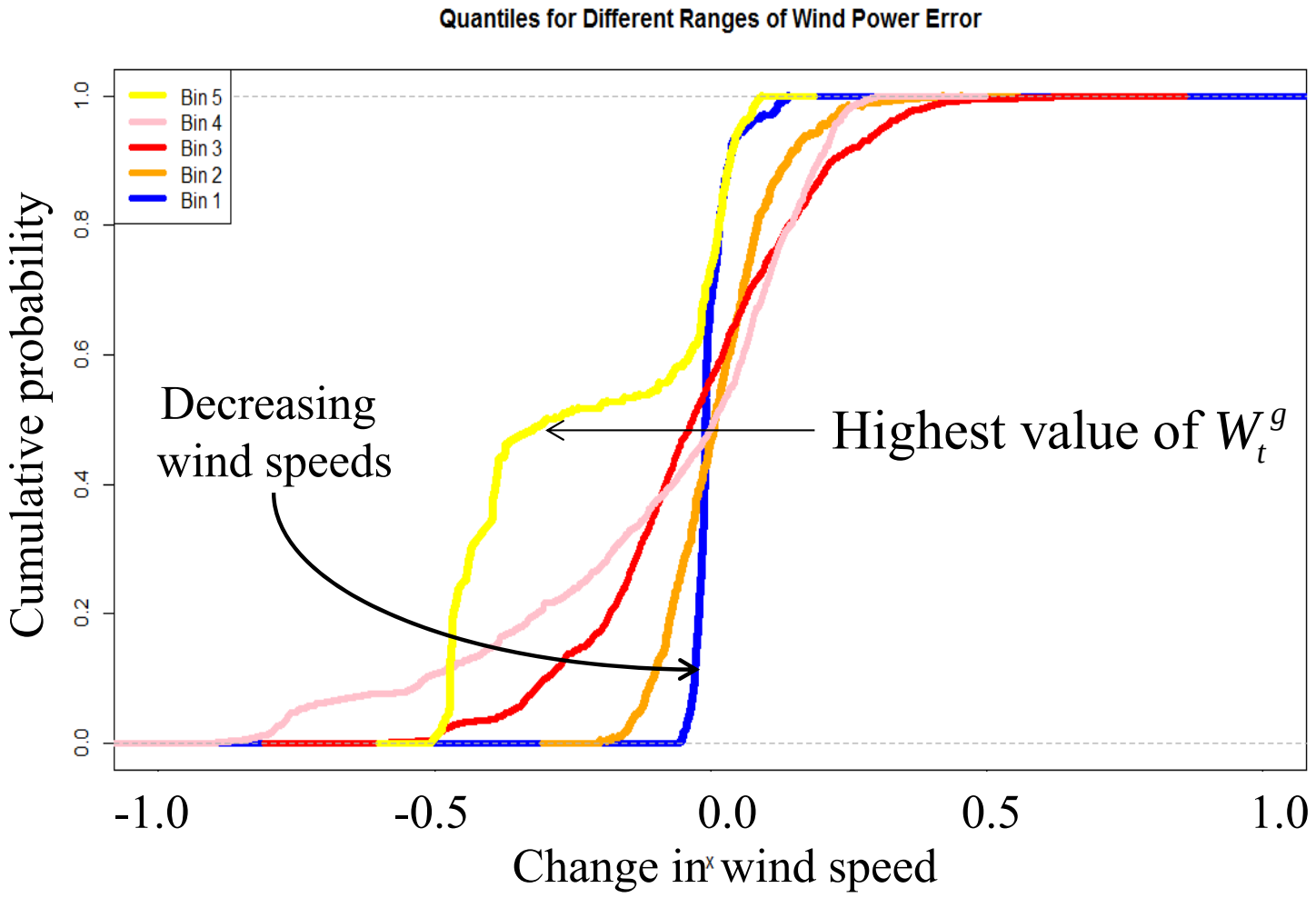
ARIMA Data Transformation

- Data transformation
  - » Observed errors are transformed to normally distributed errors using quantile mapping:



Simulating onshore wind

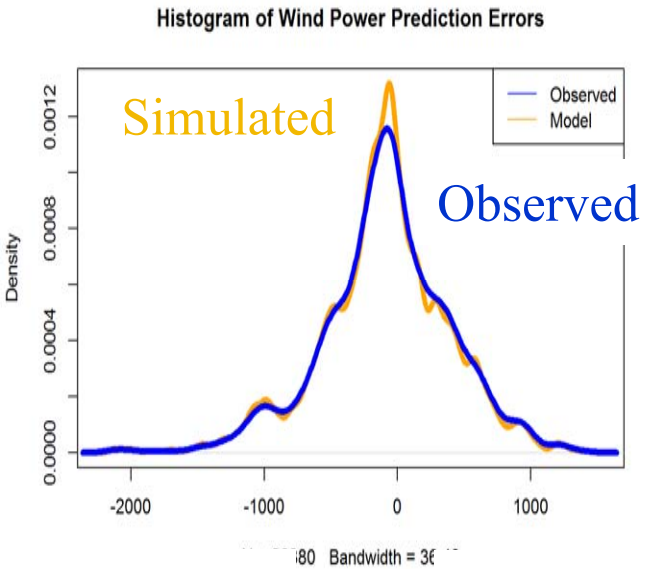
- Conditional cumulative distribution of wind error
  - » Conditioned on the aggregated wind state W_t^g



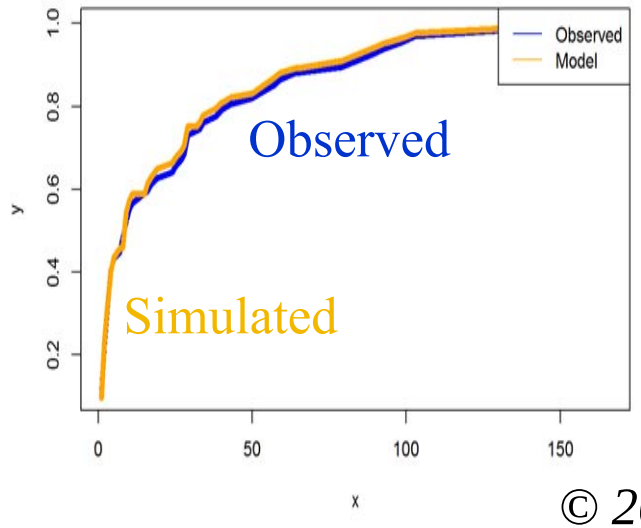
Simulating onshore wind

- Modeling a single, aggregated time series

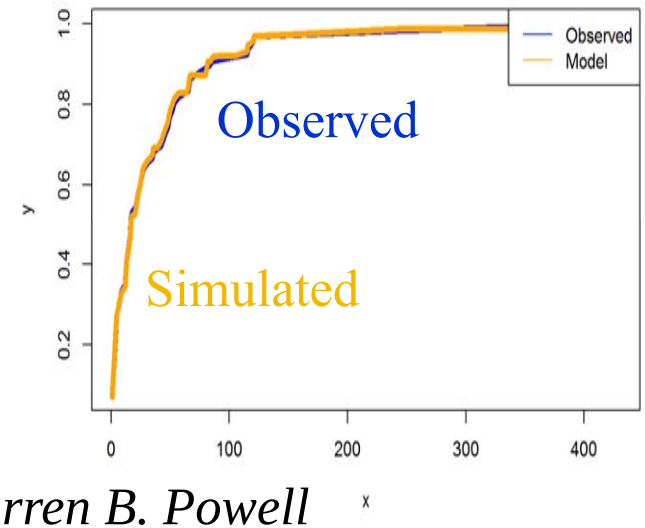
Error distribution



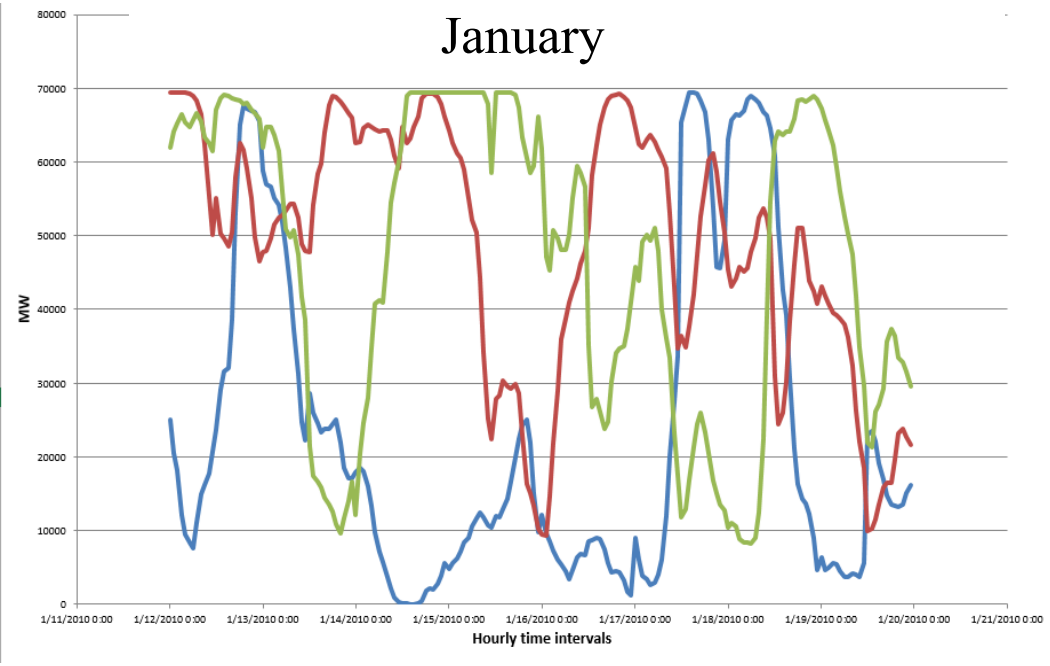
Upcrossing distribution

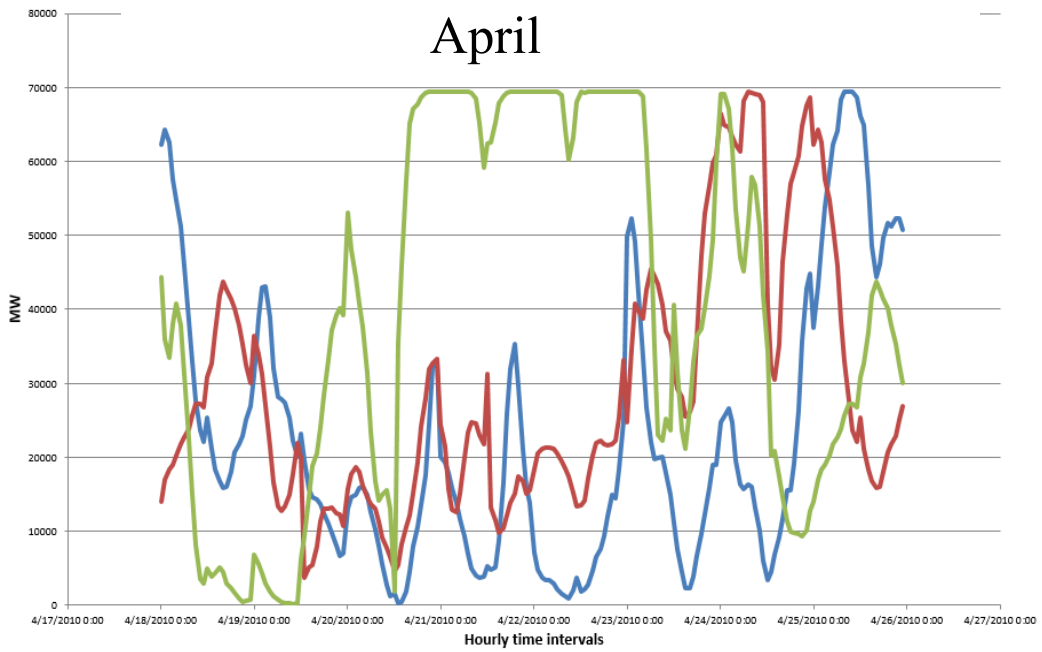


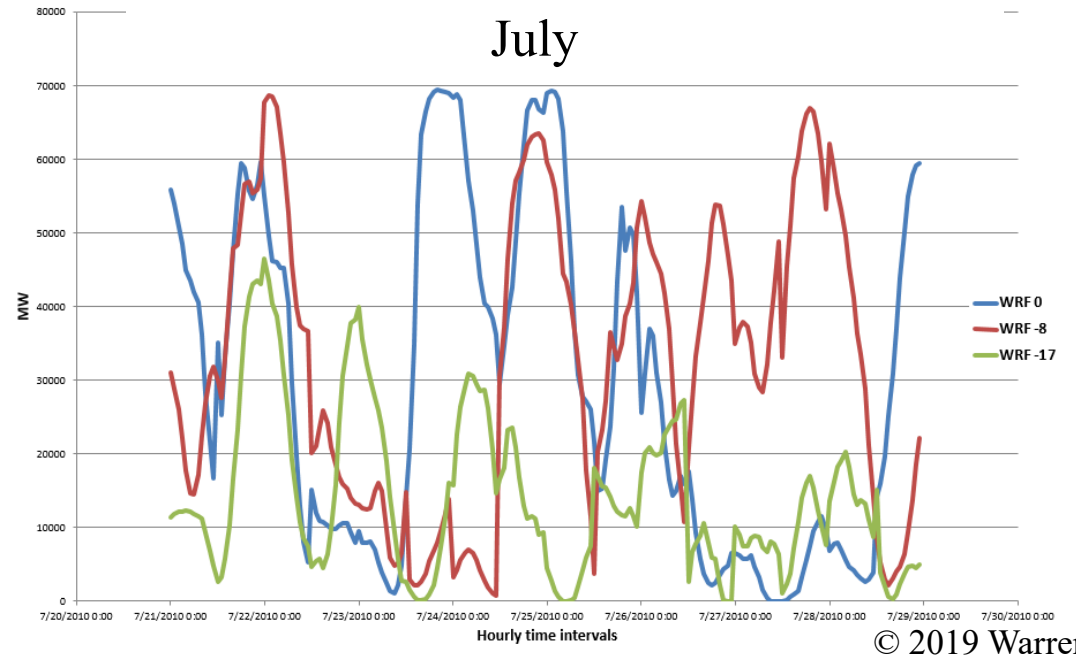
Downcrossing distribution

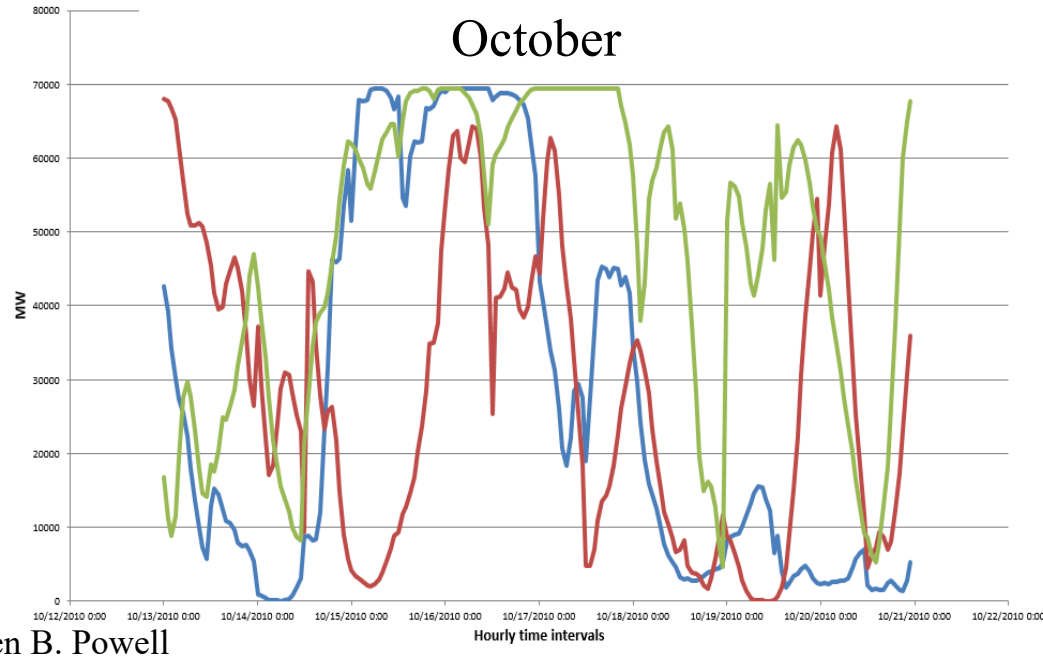


WRF simulations for offshore wind



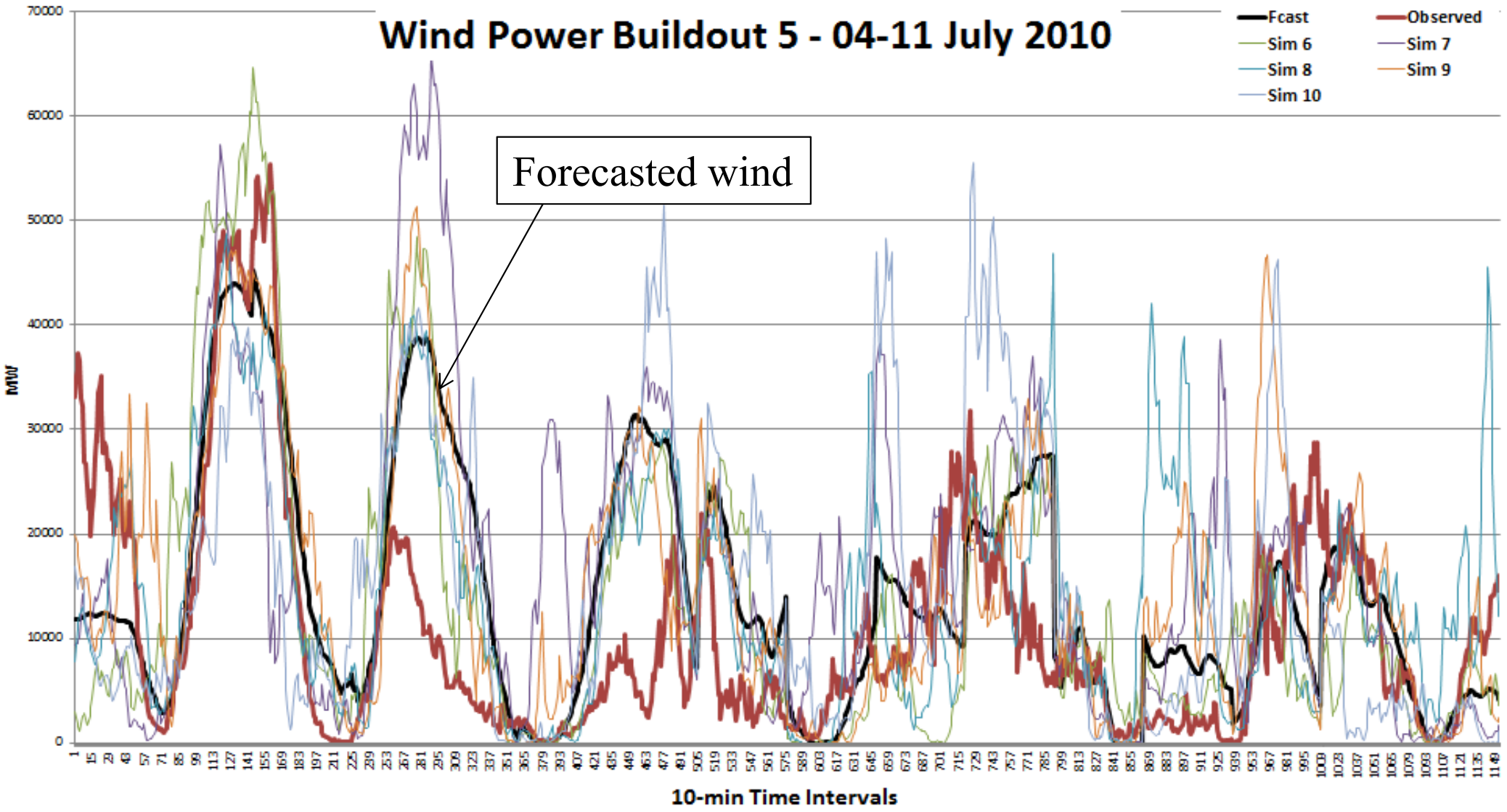






Simulating offshore wind

- Offshore wind – Buildout level 5 – ARMA error model



Designing policies

Designing policies

- We have to start by describing what we mean by a policy.
  - » Definition:

    A policy is a mapping from a state to an action.

    … any mapping.

Designing policies

- "Policies" and the English language

Behavior	Habit	Procedure
Belief	Laws/bylaws	Process
Bias	Manner	Protocols
Commandment	Method	Recipe
Conduct	Mode	Ritual
Convention	Mores	Rule
Culture	Patterns	Style
Customs	Plans	Technique
Dogma	Policies	Tenet
Etiquette	Practice	Tradition
Fashion	Prejudice	Way of life
Formula	Principle	

Designing policies

- Two fundamental strategies:

1) Policy search – Search over a class of functions for making decisions to optimize some metric.

$$\max_{\pi=(f\in F,\theta^f\in\Theta^f)} E\left\{\sum_{t=0}^{T} C\left(S_t, X_t^{\pi}(S_t\,|\,\theta)\right)|\, S_0\right\}$$

2) Lookahead approximations – Approximate the impact of a decision now on the future.

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\max_{\pi\in\Pi}\left\{\mathbb{E}\sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'}))\,|\,S_{t+1}\right\}|\,S_t,x_t\right\}\right)$$

Designing policies

- Policy search:

1a) Policy function approximations (PFAs) $x_t = X^{PFA}(S_t \mid \theta)$

- Lookup tables
  - "when in this state, take this action"
- Parametric functions
  - Order-up-to policies: if inventory is less than s, order up to S.
  - Affine policies - $x_t = X^{PFA}(S_t \mid \theta) = \sum_{f \in F} \theta_f \phi_f(S_t)$
  - Neural networks
- Locally/semi/non parametric
  - Requires optimizing over local regions

1b) Cost function approximations (CFAs)

- Optimizing a deterministic model modified to handle uncertainty (buffer stocks, schedule slack)

$$X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$$

Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  - » An optimal policy (based on looking ahead):

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$

2a) Approximating the value of being in a downstream state using machine learning ("value function approximations")

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ V_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

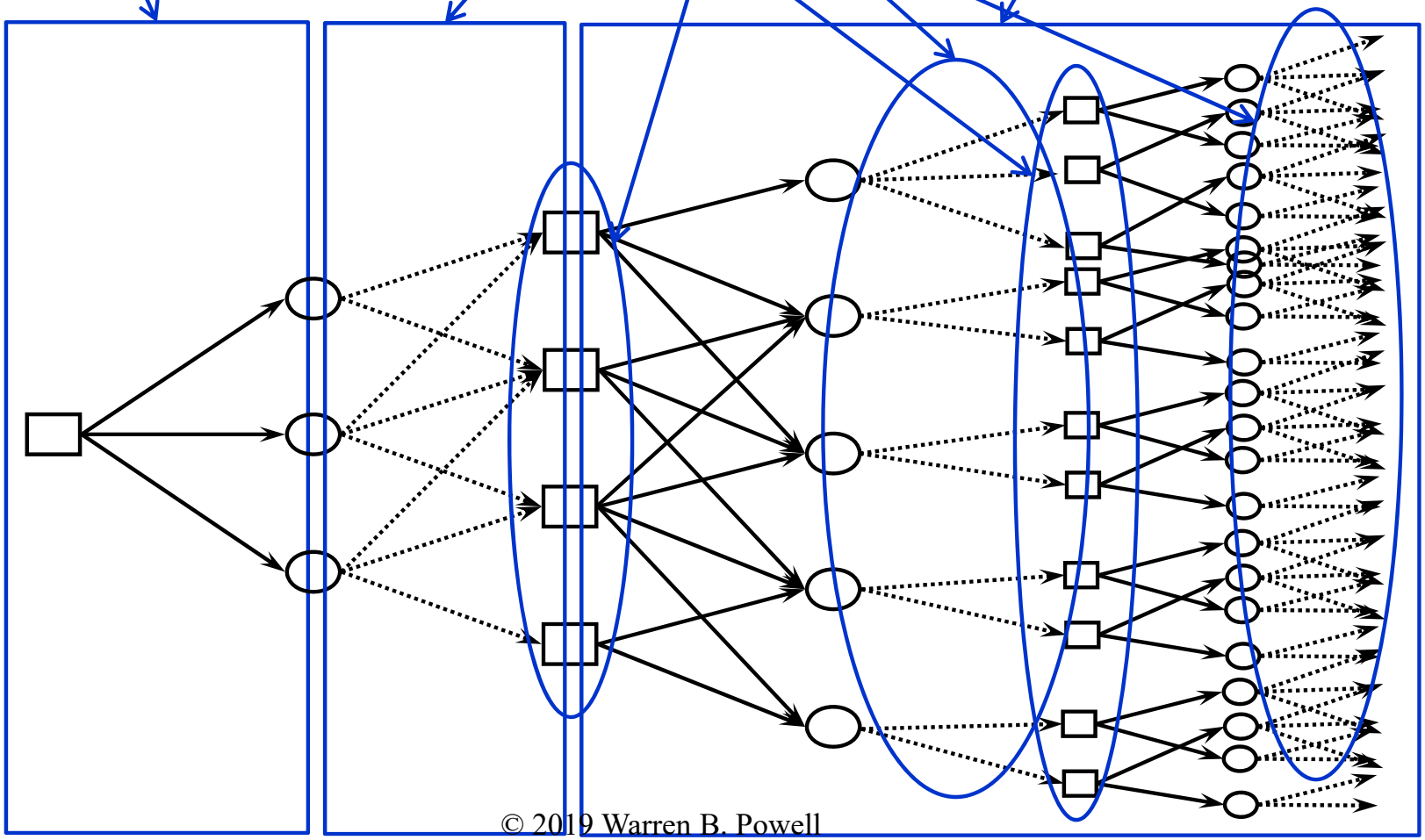
$$X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \bar{V}_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

$$= \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^x(S_t^x) \right)$$

Designing policies

- The ultimate lookahead policy is optimal

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$



Designing policies

- The ultimate lookahead policy is optimal

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\max_{\pi\in\Pi}\left\{\mathbb{E}\sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'}))\,|\,S_{t+1}\right\}|\,S_t,x_t\right\}\right)$$

» 2b) Instead, we have to solve an approximation called the *lookahead model:*

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \tilde{\mathbb{E}}\left\{\max_{\tilde{\pi}\in\tilde{\Pi}}\left\{\tilde{\mathbb{E}}\sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'}, \tilde{X}_{tt'}^{\pi}(\tilde{S}_{tt'}))\,|\,\tilde{S}_{t,t+1}\right\}|\,\tilde{S}_{tt},x_t\right\}\right)$$

» *A lookahead policy* works by approximating the *lookahead model*.

Designing policies

- Types of lookahead approximations
  - » One-step lookahead – Widely used in pure learning policies:
    - Point estimate (easiest)

$$X^{Greedy}(S^n) = \arg\max_x F(x, \mathbb{E}W)$$

    - Bayes greedy/naïve Bayes (harder)

$$X^{Bayes}(S^n) = \arg\max_x \mathbb{E}F(x, W)$$

    - Thompson sampling

$$X^{TS}(S^n) = \arg\max_x \hat{\mu}_x \quad \text{where } \hat{\mu}_x \sim N(\bar{\mu}_x^n, \beta_x^n)$$

    - Value of information (knowledge gradient)

$$X^{KG}(S^n) = \arg\max_x \hat{\mu}_x \quad \text{where } \hat{\mu}_x \sim N(\bar{\mu}_x^n, \beta_x^n)$$

Designing policies

- Creating a lookahead model
  - » We still need a policy $\tilde{X}_t^{\pi}(\tilde{S}_{tt'})$ in our lookahead model. Assume we use a linear decision rule:

$$\tilde{X}_t^{Lin}(\tilde{S}_{tt'}|\theta_t) = \theta_{t0} + \theta_{t1}\phi_1(\tilde{S}_{tt'}) + \theta_{t2}\phi_2(\tilde{S}_{tt'}),$$

  - » Our lookahead policy might then be

$$X_t^{LA-Stoch}(S_t) = \arg\max_{x_t} \Bigg(C(S_t, x_t) + \tilde{E}\left\{ \max_{\tilde{\theta}_t} \tilde{E}\left\{ \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{X}_t^{Lin}(\tilde{S}_{tt'}|\tilde{\theta}_t)) | \tilde{S}_{t,t+1} \right\} | S_t, x_t \right\} \Bigg)$$

  - » But this means that we have to optimize $\tilde{\theta}_t(S_t)$ for each time period, and given the state S_t that we are in!

Designing policies

- Creating a lookahead model
  - » A more practical idea is to replace

$$X_t^{LA-Stoch}(S_t) = \arg\max_{x_t} \Bigg(C(S_t, x_t) + \tilde{E}\left\{ \max_{\tilde{\theta}_t} \tilde{E}\left\{ \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{X}_t^{Lin}(\tilde{S}_{tt'}|\tilde{\theta}_t)) | \tilde{S}_{t,t+1} \right\} | S_t, x_t \right\} \Bigg)$$

  - » with

$$X_t^{LA-Stoch}(S_t|\theta) = \arg\max_{x_t} \Bigg(C(S_t, x_t) + \tilde{E}\left\{ \tilde{E}\left\{ \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{X}_t^{Lin}(\tilde{S}_{tt'}|\theta)) | \tilde{S}_{t,t+1} \right\} | S_t, x_t \right\} \Bigg).$$

  - » Now we have to tune θ as we would any tunable parameter.

Designing policies

- Types of lookahead approximations
  - » Multi-step lookahead
    - Deterministic lookahead, also known as model predictive control, rolling horizon procedure
    - Stochastic lookahead:
      - – Two-stage (widely used in stochastic linear programming)
      - – Multistage
        - » Monte carlo tree search (MCTS) for discrete action spaces
        - » Multistage scenario trees (stochastic linear programming) – typically not tractable.

    - We cover these at the end of the course.

Four (meta)classes of policies

Policy search

1) **Policy function approximations (PFAs)**
   - » Lookup tables, rules, parametric/nonparametric functions
2) **Cost function approximation (CFAs)**
   - » $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

Lookahead approximations

3) **Policies based on value function approximations (VFAs)**
   - » $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$
4) **Direct lookahead policies (DLAs)**
   - » *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg \max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

   - » *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

   - » *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg \max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

   - » *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg \max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Four (meta)classes of policies

Function approx.

1) **Policy function approximations (PFAs)**
   - » Lookup tables, rules, parametric/nonparametric functions

2) **Cost function approximation (CFAs)**
   - » $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

3) **Policies based on value function approximations (VFAs)**
   - » $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$

4) **Direct lookahead policies (DLAs)**
   - » *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

   - » *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

   - » *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

   - » *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Four (meta)classes of policies

1) **Policy function approximations (PFAs)**

» Lookup tables, rules, parametric/nonparametric functions

2) **Cost function approximation (CFAs)**

» $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

3) **Policies based on value function approximations (VFAs)**

» $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$

4) **Direct lookahead policies (DLAs)**

» *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

» *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

» *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

» *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Imbedded optimization

Four (meta)classes of policies

1) **Policy function approximations (PFAs)**
   - » Lookup tables, rules, parametric functions

2) **Cost function approximation (CFAs)**
   - » $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

3) **Policies based on value function approximations (VFAs)**
   - » $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$

4) **Direct lookahead policies (DLAs)**
   - » *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg \max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

   - » *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

   - » *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

   - » *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg \max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Designing policies

- Finding the best policy
  - » We have to first articulate our classes of policies

$$f \in \mathcal{F} = \{PFAs, CFAs, VFAs, DLAs\}$$

$$\theta \in \Theta^f = \text{Parameters that characterize each family.}$$

  - » So minimizing over $\pi \in \Pi$ means:

$$\Pi = \left\{ f \in \mathcal{F}, \theta \in \Theta^f \right\}$$

  - » We then have to pick an objective such as

$$\max_\pi \mathbb{E} C\left(S_T, X_T^\pi\right) \quad \text{or} \quad \mathbb{E} F(X^{\pi,N}, W)$$

or

$$\max_\pi \mathbb{E} \sum_{t=0}^{T} C\left(S_t, X^\pi(S_t \mid \theta)\right) \quad \text{or} \quad \mathbb{E} \sum_{n=0}^{N-1} F\left(X^\pi(S^n \mid \theta), W^{n+1}\right)$$

Designing policies

- Evaluating a policy
  - » We simulate a policy N times and take an average:

$$\bar{F}^{\pi} = \frac{1}{N}\sum_{n=1}^{N} F^{\pi}(\omega^n)$$

  - » If we simulate policies π_1 and π_2, we would like to conclude that π_1 is better than π_2 if

$$\bar{F}^{\pi_1} > \bar{F}^{\pi_2}$$

  - » There are some technical issues we deal with later:
    - How large should N be?
    - How do we deal with the fact that this is at best a statistically noisy measurement?
      - – Need to compute confidence intervals
    - How do we search over different policies?
      - – Stochastic search (or "optimal learning")

Designing policies

- There are two ways to evaluate a policy:

» **Computer simulation**

- We use a computer model to simulate a policy.
- Provides controlled testing environment.
- Requires living with model assumptions.
- Can quickly compare different policies.

» **Real world**

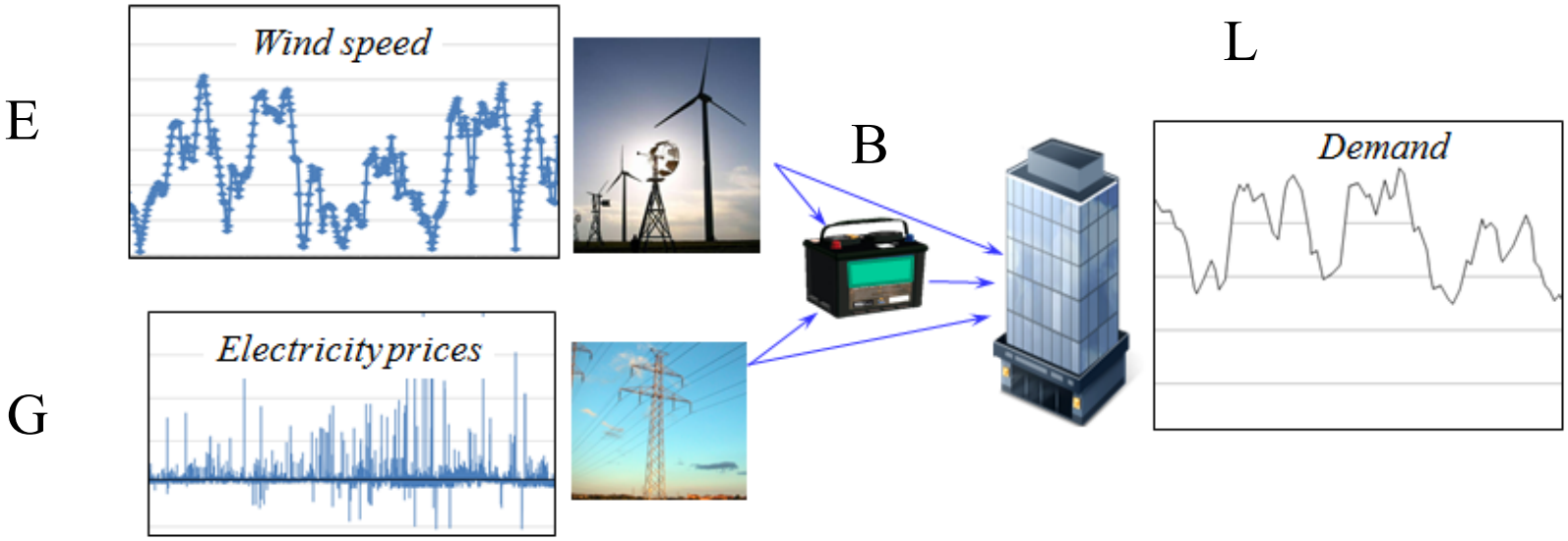
- We observe the performance of a policy in the field.
- Have to live with what the real world offers.
- No control over test environment.
- Policy evaluations are quite slow.

Four classes of policies

Energy storage example

An energy storage problem

- Designing policies



» We can illustrate this problem with each of the four classes of policies

An energy storage problem

- Policy function approximation

11.8.1 Policy function approximation

Our policy function approximation is given by

$$X_t^{PFA}(S_t|\theta) = \begin{cases} x_t^{EL} &= \min\{L_t, E_t\}, \\ x_t^{BL} &= \begin{cases} h_t & \text{If } p_t > \theta^U \\ 0 & \text{If } p_t < \theta^U \end{cases} \\ x_t^{GL} &= L_t - x_t^{EL} - x_t^{BL}, \\ x_t^{EB} &= \min\{E_t - x_t^{EL}, \rho^{chrg}\}, \\ x_t^{GB} &= \begin{cases} \rho^{chrg} - x_t^{EB} & \text{If } p_t < \theta^L \\ 0 & \text{If } p_t > \theta^L \end{cases} \end{cases}$$

where $h_t = \min\{L_t - x_t^{EL}, \min\{R_t, \rho^{chrg}\}\}$. This policy is parameterized by (θ^L, θ^U) which determine the price points at which we charge or discharge.

An energy storage problem

- Cost function approximation

11.8.2 Cost function approximation

The cost function approximation minimizes a one-period cost plus a tunable error correction term:

$$X^{CFA-EC}(S_t|\theta) = \arg\min_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \theta(x_t^{GB} + x_t^{EB} + x_t^{BL})\right), \quad (11.25)$$

where $\mathcal{X}_t$ is defined by (9.21)-(9.25). We use a linear correction term for simplicity which is parameterized by the scalar θ.

» Notes:

- This is a very simple "cost function correction term" with a scalar parameter θ

An energy storage problem

- Value function approximation

11.8.3 Value function approximation

Our VFA policy uses an approximate value function approximation, which we write as

$$X^{VFA}(S_t) = \arg\min_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \overline{V}_t^x(R_t^x)\right), \tag{11.26}$$

where $\overline{V}_t^x(R_t^x)$ is a piecewise linear function approximating the marginal value of the post-decision resource state. We use methods described in chapter 19 to compute the value function approximation which exploits the natural convexity of the problem. For now, we simply note that the approximation is quite good.

» We will describe methods for estimating $\bar{V}_t^x(R_t^x)$ later in the course.

An energy storage problem

- Direct lookahead

11.8.4 Deterministic lookahead

The next policy is a deterministic lookahead over a horizon H which has access to a forecast of wind energy.

$$X_t^{LA-DET}(S_t|H) = \underset{(x_t,\tilde{x}_{t+1,t},\ldots,\tilde{x}_{t,t+H})}{\arg\min} \left(C(S_t,x_t) + \sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'},\tilde{x}_{tt'}) \right) \quad (11.27)$$

subject to, for $t' = t,\ldots,T$:

$$\tilde{x}_{tt'}^{EL} + \tilde{x}_{tt'}^{EB} \leq f_{tt'}^{E}, \quad (11.28)$$

$$f_{tt'}^{\eta}(\tilde{x}_{tt'}^{GL} + \tilde{x}_{tt'}^{EL} + \tilde{x}_{tt'}^{BL}) = f_{tt'}^{L}, \quad (11.29)$$

$$\tilde{x}_{tt'}^{BL} \leq \tilde{R}_{tt'}, \quad (11.30)$$

$$\tilde{R}_{t,t'+1} - (\tilde{R}_{tt'} + f_{t,t'+1}^{\eta}(\tilde{x}_{tt'}^{GB} + \tilde{x}_{tt'}^{EB}) - \tilde{x}_{tt'}^{BL}) = f_{t,t'+1}^{R}, \quad (11.31)$$

$$\tilde{x}_{tt'} \geq 0. \quad (11.32)$$

» This is a classic rolling horizon procedure.

An energy storage problem

- We can create distinct flavors of this problem:
  - » Problem class 1 – Best for PFAs
    - Highly stochastic (heavy tailed) electricity prices
    - Stationary data
  - » Problem class 2 – Best for CFAs
    - Stochastic prices and wind (but not heavy tailed)
    - Stationary data
  - » Problem class 3 - Best for VFAs
    - Stochastic wind and prices (but not too random)
    - Time varying loads, but inaccurate wind forecasts
  - » Problem class 4 – Best for deterministic lookaheads
    - Relatively low noise problem with accurate forecasts
  - » Problem class 5 – A hybrid policy worked best here
    - Stochastic prices and wind, nonstationary data, noisy forecasts.

An energy storage problem

- The policies
  - » The PFA:
    - Charge battery when price is below p1
    - Discharge when price is above p2
  - » The CFA
    - Optimize over a horizon H; maintain upper and lower bounds (u, l) for every time period except the first (note that this is a hybrid with a lookahead).
  - » The VFA
    - Piecewise linear, concave value function in terms of energy, indexed by time.
  - » The lookahead (deterministic)
    - Optimize over a horizon H (only tunable parameter) using forecasts of demand, prices and wind energy
  - » The lookahead CFA
    - Use a lookahead policy (deterministic), but with a tunable parameter that improves robustness.

An energy storage problem

- Each policy is best on certain problems
  - » Results are percent of *posterior* optimal solution

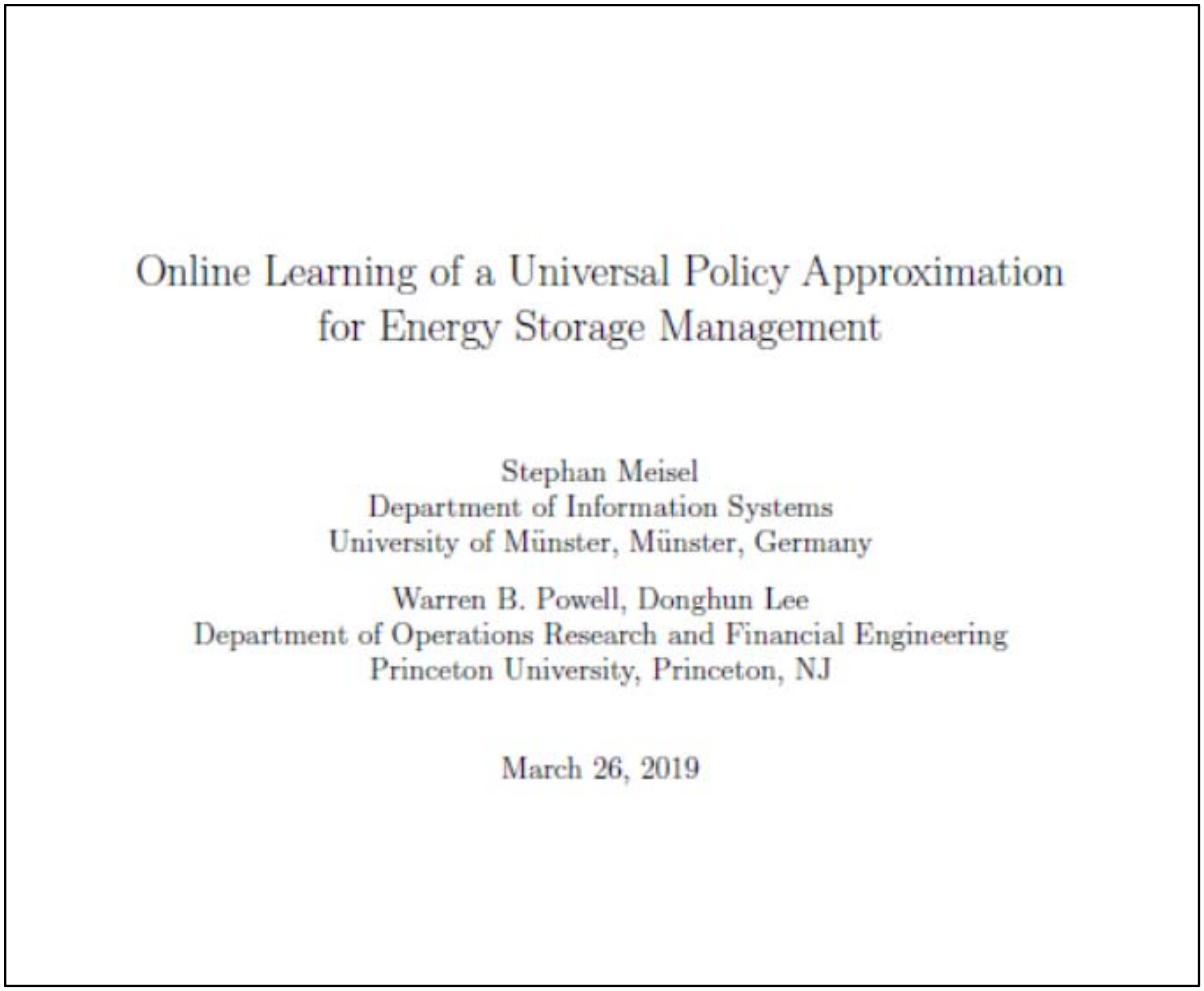
Problem:	Problem description	PFA	CFA Error correction	VFA	Determ. Lookahead	CFA Lookahead
A	A stationary problem with heavy-tailed prices, relatively low noise, moderately accurate forecasts.	**0.959**	**0.839**	**0.936**	**0.887**	**0.887**
B	A time-dependent problem with daily load patterns, no seasonalities in energy and price, relatively low noise, less accurate forecasts.	**0.714**	**0.752**	**0.712**	**0.746**	**0.746**
C	A time-dependent problem with daily load, energy and price patterns, relatively high noise, forecast errors increase over horizon.	**0.865**	**0.590**	**0.914**	**0.886**	**0.886**
D	A time-dependent problem, relatively low noise, very accurate forecasts.	**0.962**	**0.749**	**0.971**	**0.997**	**0.997**
E	Same as (C), but the forecast errors are stationary over the planning horizon.	**0.865**	**0.590**	**0.914**	**0.922**	**0.934**

  - » … any policy might be best depending on the data.

Joint research with Prof. Stephan Meisel, University of Muenster, Germany.

An energy storage problem

- The next step – the Universal Policy



Online Learning of a Universal Policy Approximation
for Energy Storage Management

Stephan Meisel
Department of Information Systems
University of Münster, Münster, Germany

Warren B. Powell, Donghun Lee
Department of Operations Research and Financial Engineering
Princeton University, Princeton, NJ

March 26, 2019

Evaluating policies

Evaluating policies

Class 1) State-independent, final reward:

$$\begin{aligned}\max_{\pi} F^{\pi} &= \mathbb{E}\{F(x^{\pi,N}, \widehat{W})|S^0\} \\ &= \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\widehat{W}|S^0}F(x^{\pi,N}, \widehat{W}).\end{aligned}$$

We then simulate F^{π} using

$$F^{\pi}(\theta|\omega,\psi) = F(x^{\pi,N}(\theta|\omega), \widehat{W}(\psi)).$$

Finally we approximate the expectation by averaging using

$$\overline{F}^{\pi}(\theta) = \frac{1}{K}\frac{1}{L}\sum_{k=1}^{K}\sum_{\ell=1}^{L} F^{\pi}(\theta|\omega^k,\psi^{\ell}).$$

Evaluating policies

Class 2) State-independent, cumulative reward reward:

$$\begin{aligned}\max_{\pi} F^{\pi} &= \mathbb{E}\left\{\sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1})|S^0\right\} \\ &= \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1}).\end{aligned}$$

We then simulate F^{π} using

$$F^{\pi}(\theta|\omega) = \sum_{n=0}^{N-1} F(X^{\pi}(S^n(\omega)|\theta), W^{n+1}(\omega)).$$

Finally we approximate the expectation by averaging using

$$\overline{F}^{\pi}(\theta) = \frac{1}{K}\sum_{k=1}^{K} F^{\pi}(\theta|\omega^k).$$

Evaluating policies

Class 3) State-dependent, cumulative reward:

$$\begin{aligned}\max_{\pi} F^{\pi} &= \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t), W_{t+1})|S_0\right\} \\ &= \mathbb{E}_{S_0}\mathbb{E}_{W_1,\ldots,W_T|S_0}\left\{\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t), W_{t+1})|S_0\right\}.\end{aligned}$$

We then simulate F^{π} using

$$F^{\pi}(\theta|\omega) = \sum_{t=0}^{T-1} C(S_t(\omega), X^{\pi}(S_t(\omega)|\theta), W_{t+1}(\omega)).$$

We then average over sample paths to obtain

$$\overline{F}^{\pi}(\theta) = \frac{1}{K}\sum_{k=1}^{K} F^{\pi}(\theta|\omega^k).$$

Evaluating policies

Class 4) State-dependent, final reward:

$$\begin{aligned}\max_{\pi^{lrn}} F^{\pi^{lrn}} &= \mathbb{E}\{C(S, X^{\pi^{impl}}(S|\theta^{imp}), \widehat{W})|S^0\} \\ &= \mathbb{E}_{S^0}\mathbb{E}^{\pi^{imp}}_{((W^n_t)^T_{t=0})^N_{n=0}|S^0}\left(\mathbb{E}^{\pi^{imp}}_{(\widehat{W}_t)^T_{t=0}|S^0}\frac{1}{T}\sum_{t=0}^{T-1} C(S_t, X^{\pi^{impl}}(S_t|\theta^{imp}), \widehat{W}_{t+1})\right).\end{aligned}$$

We then simulate F^π using

$$F^\pi(\theta^{lrn}|\omega, \psi) = \sum_{t=0}^{T} C(S_t(\omega), X^{\pi^{imp}}(S_t(\omega)|\theta^{imp}), \widehat{W}_{t+1}(\psi)).$$

We then average over sample paths to obtain

$$\overline{F}^\pi(\theta^{lrn}) = \frac{1}{K}\frac{1}{L}\sum_{k=1}^{K}\sum_{\ell=1}^{L} F^\pi(\theta^{lrn}|\omega^k, \psi^\ell).$$