Statistical machine learning with structured data

Daniel J. McDonald

Department of Statistics
Indiana University

http://mypage.iu.edu/~dajmcdon
dajmcdon@indiana.edu

September 22, 2014

Introduction

Machine learning is statistics with a focus on prediction, scalability, and high-dimensional problems.

Regression: predict $Y \in \mathbb{R}$ from X.

- Example: Predict GDP growth Y from stock prices and macroeconomic data X

Classification: predict $Y \in \{0, 1\}$ from X.

- Predict if an email X is real ($Y = 1$) or spam ($Y = 0$).

Finding structure:

- Manifold learning: learn a low-dimensional representation for high-dimensional data.
- Clustering: find meaningful groups of observations.

My research

Focus on developing theory which can justify what practitioners actually do.

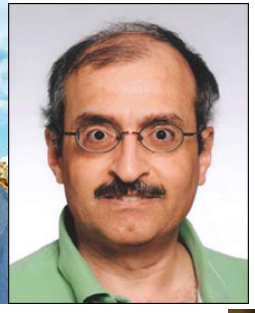
Assume as little as possible about the "truth"

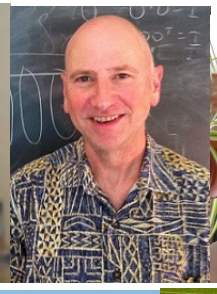
Most of the time, this involves thinking about data which has a certain structure.

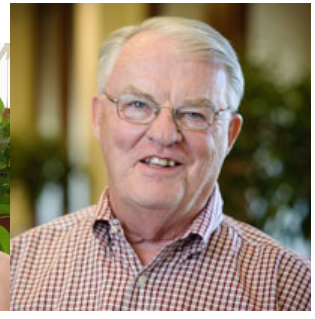
E.g.

- time-series
- sparsity
- functional

Collaborators







Projects

1. High-dimensional statistics for macroeconomic forecasting
2. Tuning parameter selection for regularized linear models
3. Statistical and computational efficiency for massive data sets via approximation-regularization
4. Other interesting stuff

High-dimensional statistics for macroeconomic forecasting

Publications and preprints

- "Time-series forecasting: model evaluation and selection using nonparametric risk bounds." (with Shalizi and Schervish)
- "Estimating β-mixing coefficients." (with Shalizi and Schervish)
- "Estimating β-mixing coefficients via high-dimensional density estimation." (with Shalizi and Schervish)
- "Your favorite DSGE sucks." (with Shalizi and Homrighausen)
- "Sparse additive state-space models" (with Shalizi and Homrighausen)
- "Risk bounds for time series without strong mixing" (with Shalizi and Homrighausen)

This work is supported by the Institute for New Economic Thinking.

Publications and preprints

- "Time-series forecasting: model evaluation and selection using nonparametric risk bounds." (with Shalizi and Schervish)
- "Estimating β-mixing coefficients." (with Shalizi and Schervish)
- "Estimating β-mixing coefficients via high-dimensional density estimation." (with Shalizi and Schervish)
- "Your favorite DSGE sucks." (with Shalizi and Homrighausen)
- "Sparse additive state-space models" (with Shalizi and Homrighausen)
- "Risk bounds for time series without strong mixing" (with Shalizi and Homrighausen)

This work is supported by the Institute for New Economic Thinking.

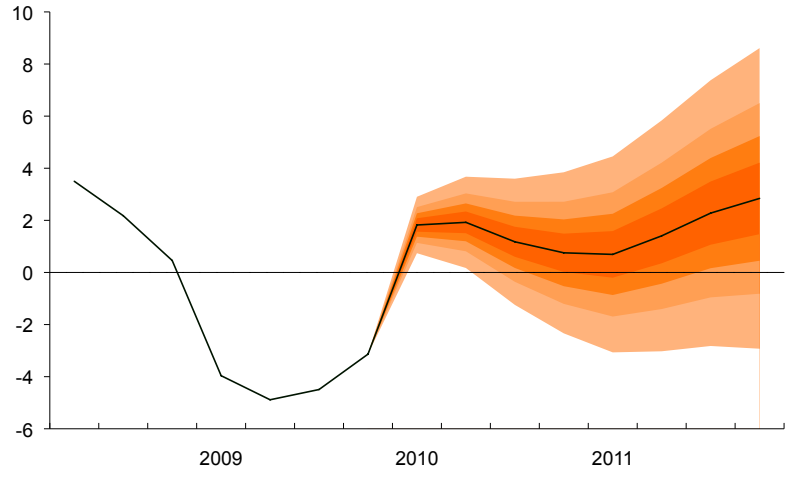
Forecasting

- Given some data

$$y_1, \ldots, y_n \in \mathbb{R}^p$$

- Want to predict the next data point(s)

$$Y_{n+1}, \ldots, Y_{n+k}$$



Source: Czech National Bank

The state of the art

I want to forecast the macroeconomy.

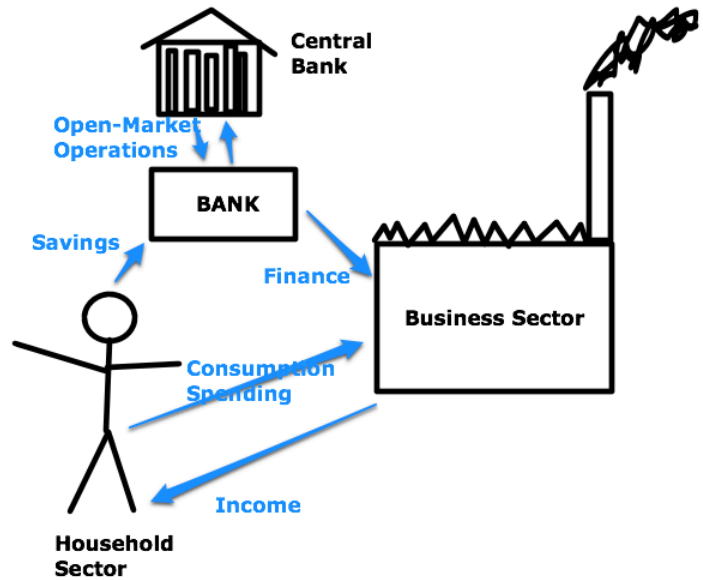
I want my forecasts to derive from economic theory.

I use a Dynamic Stochastic General Equilibrium model.

Are my forecasts any good? Can I make them any better? Can I make inferences about behavior based on my fitted DSGE model?

DSGE models

- Most active area of macroeconomic research in the last 30 years
- Arose in response to the Lucas (1976) critique
- Pioneered by Kydland and Prescott (1982)
- Attempt to incorporate "rational behavior" into forecasting models
- Have come under fire for being unable to forecast the financial collapse of 2008–?



Source: Brad DeLong's realization of Daniel Davies' DSGE model

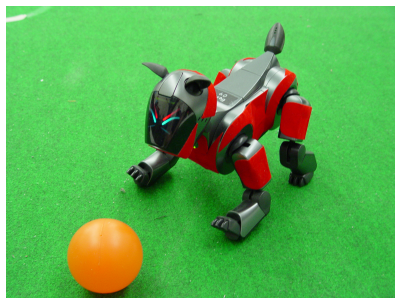
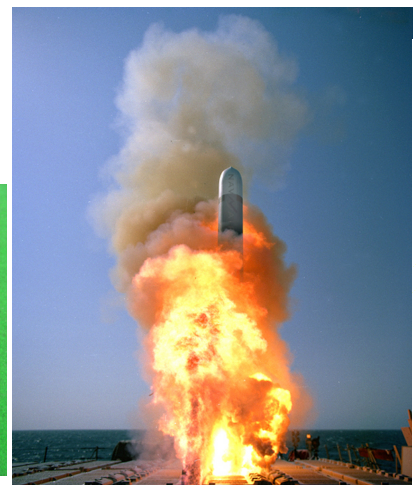
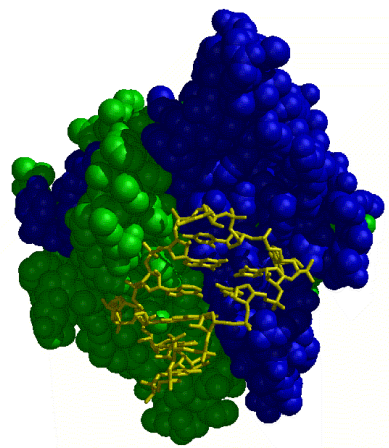
RBC model

- Imagine an infinitely-long-lived individual who faces the following constrained optimization problem:

$$\begin{aligned}
\max_{c_t, l_t} U &= \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t, l_t) \\
y_t &= z_t q(k_t, n_t) \\
1 &= n_t + l_t \\
y_t &= c_t + i_t \\
k_{t+1} &= i_t + (1 - \delta) k_t \\
z_t &\sim \text{AR}(1)
\end{aligned}$$

State space models detour

- Lots of disciplines use state space models
- Sometimes motivated directly by physical relationships

State space models detour

State equation:

$$\alpha_{t+1} = T\alpha_t + \eta_{t+1}$$

Observation equation:

$$y_t = A\alpha_t + \epsilon_t$$

Some assumptions:

$$0 = \mathbb{E}[\eta_t] = \mathbb{E}[\epsilon_t]$$

I get to observe y_t, I never see α_t "hidden state"

Relationship to state space models

DSGE Model

$$\max_{c_t, l_t} U = \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t, l_t)$$

$$y_t = z_t q(k_t, n_t)$$
$$1 = n_t + l_t$$
$$y_t = c_t + i_t$$
$$k_{t+1} = i_t + (1 - \delta) k_t$$
$$z_t \sim \text{AR}(1)$$

Math games

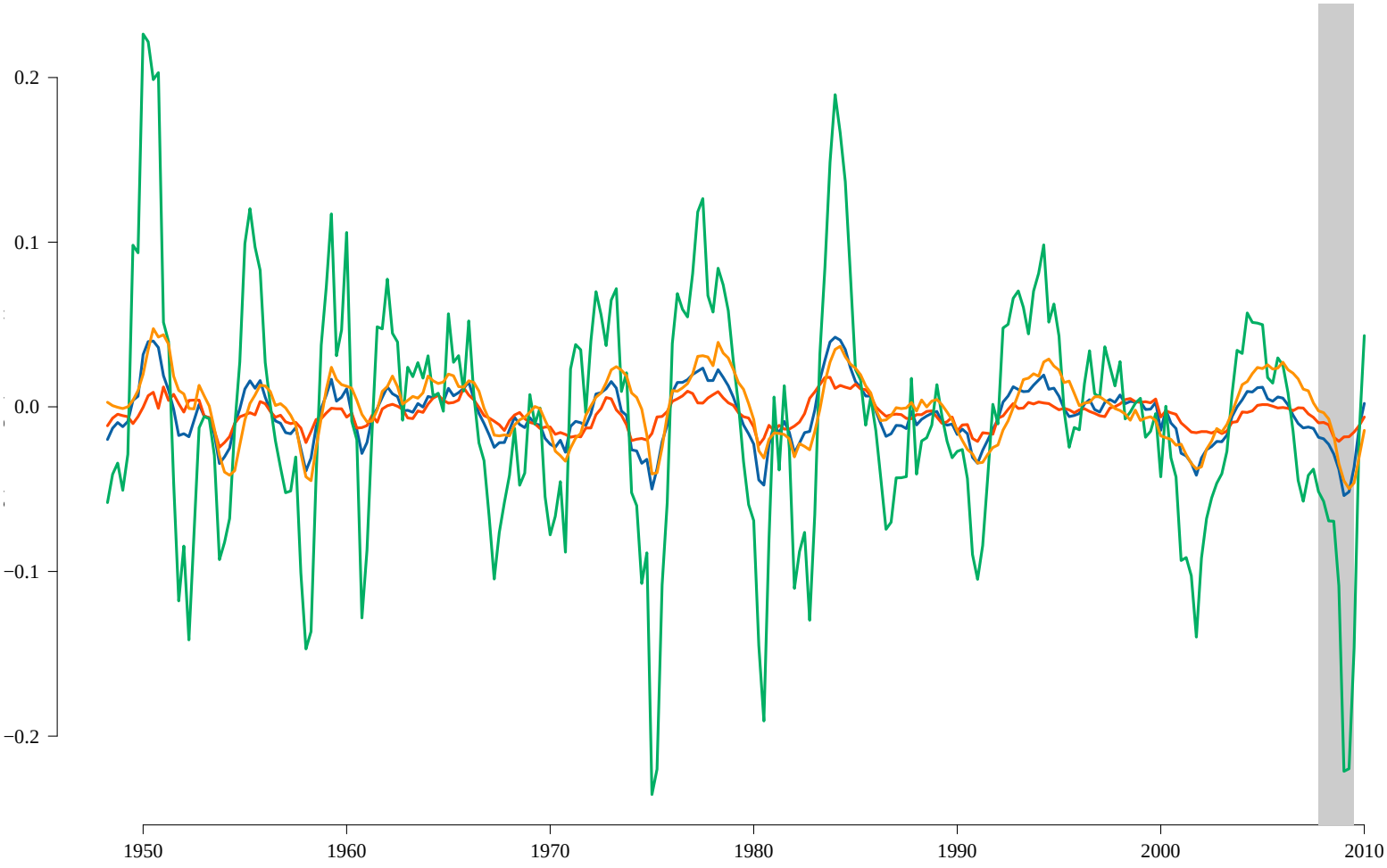
State Space Model

$$\mathbf{x}_t = g(\boldsymbol{\alpha}_t, \epsilon_t)$$
$$\boldsymbol{\alpha}_{t+1} = h(\boldsymbol{\alpha}_t, \eta_{t+1})$$
$$\boldsymbol{\alpha}_1 \sim F$$

DSGE as prior

- The DSGE works as a prior on the underlying state-space model
- The quality of the forecasts depends on the tightness and location of the regularization
- Altering the DSGE (adding restrictions, assumptions, sticky prices, etc) changes both
- IF this makes the DSGE more realistic, it corresponds to decreasing bias and increasing variance
- It is not clear that this is a good idea.

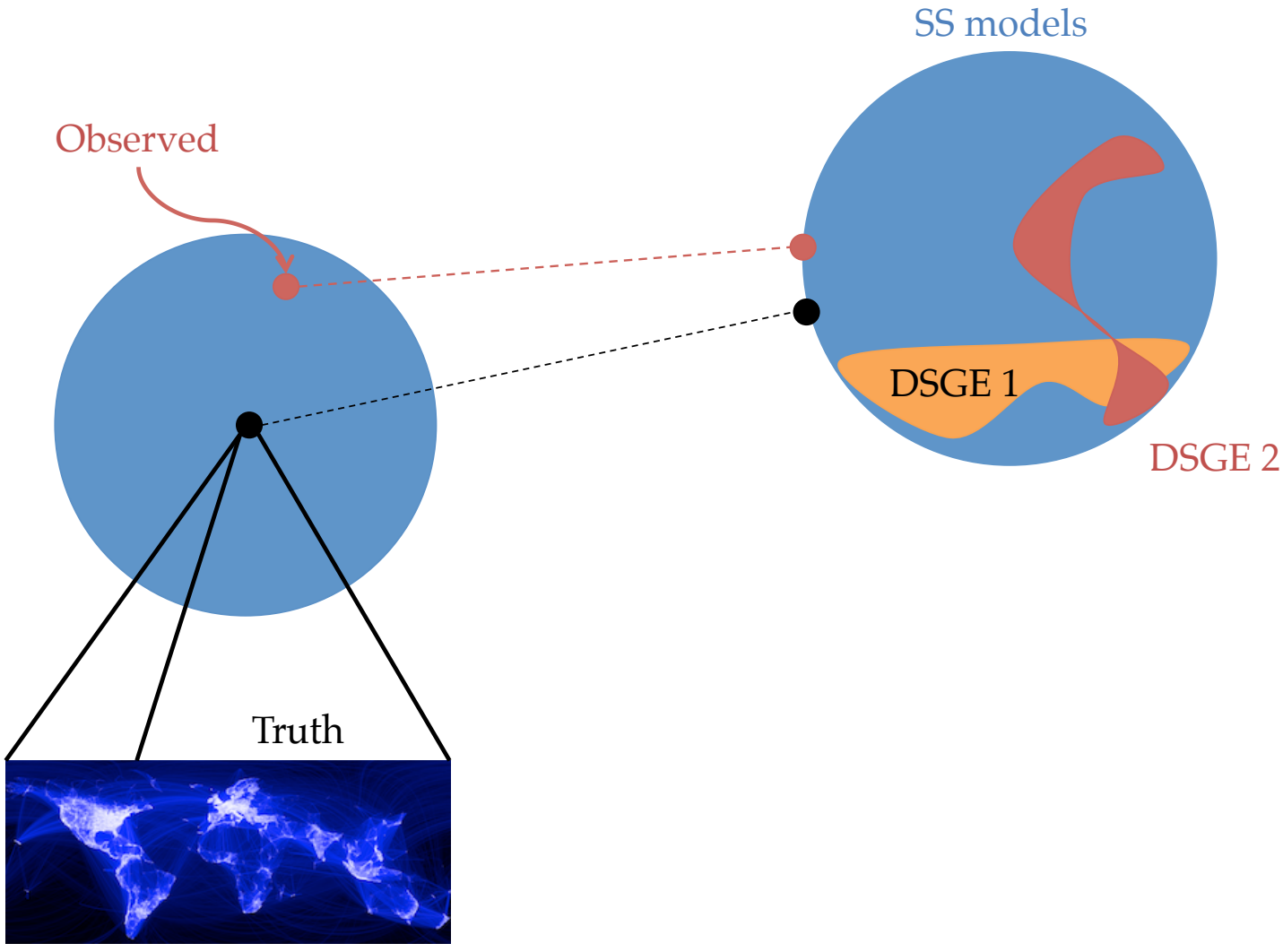
The problem



Income Consumption Investment Hours worked

MSE: Non-recession .89 During recession 2.11

The world economy. According to Facebook.


SS models
Observed
DSGE 1
DSGE 2
Truth

The issues

- DSGEs do not and cannot describe the real economy
- Neither can state-space models or anything else
- We might still be able to predict well
- Why didn't DSGEs at least predict well?

Why did it fail?

- First of all, it actually did much worse than it looks.
- Those were residuals after detrending. A lot of the recession got considered "trend", and economic models are for deviations about the trend.
- It missed the deviations too, but that was $< 1\%$ of the downturn.
- The US Congress held hearings on this issue.

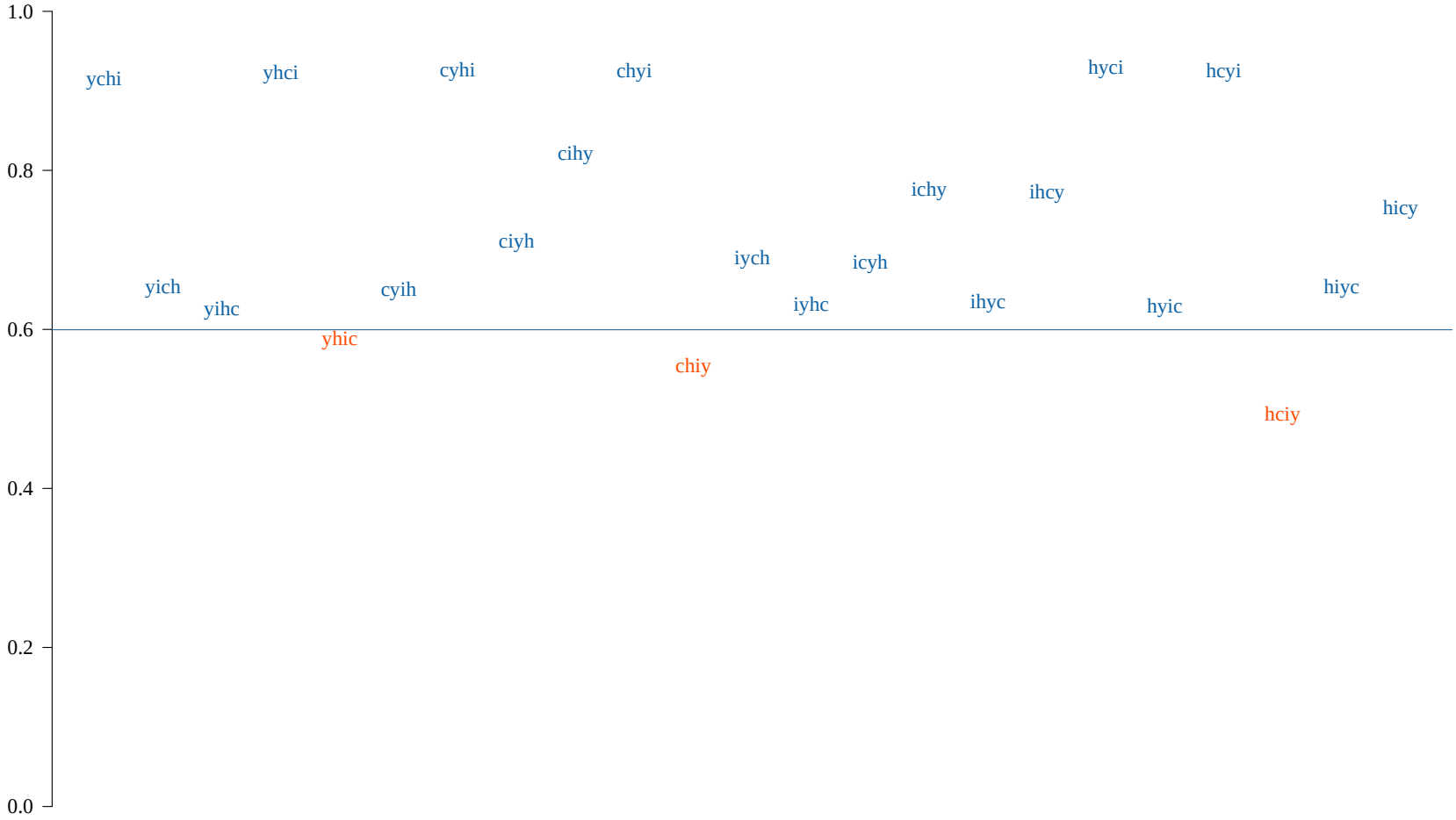
Are DSGEs good for forecasting?

- Requires separate model for the trend
- Puts strange nonlinear constraints on the space of models
- Don't know if these constraints help or hurt: the suspicion is hurt (next slide)
- Any state space model is wrong. A DSGE is a SS model. Therefore it is wrong.
- Can it forecast better than an unconstrained SS model?
- What determines its forecasting abilities?

Do DSGEs really impose "economically relevant" constraints?

1. The point of a DSGE is that it adds economic theory that is otherwise absent from the statistical model
2. If this model fits the data well, then the DSGE is a good approximation to the real economy
3. What does it mean to "fit the data" without overfitting?
4. Try an experiment: mess up the data, see if the DSGE notices
5. Mislabel the time series. Call Income $\rightarrow$ Consumption, etc.

% MSE remaining (relative to predicting w/ 0)

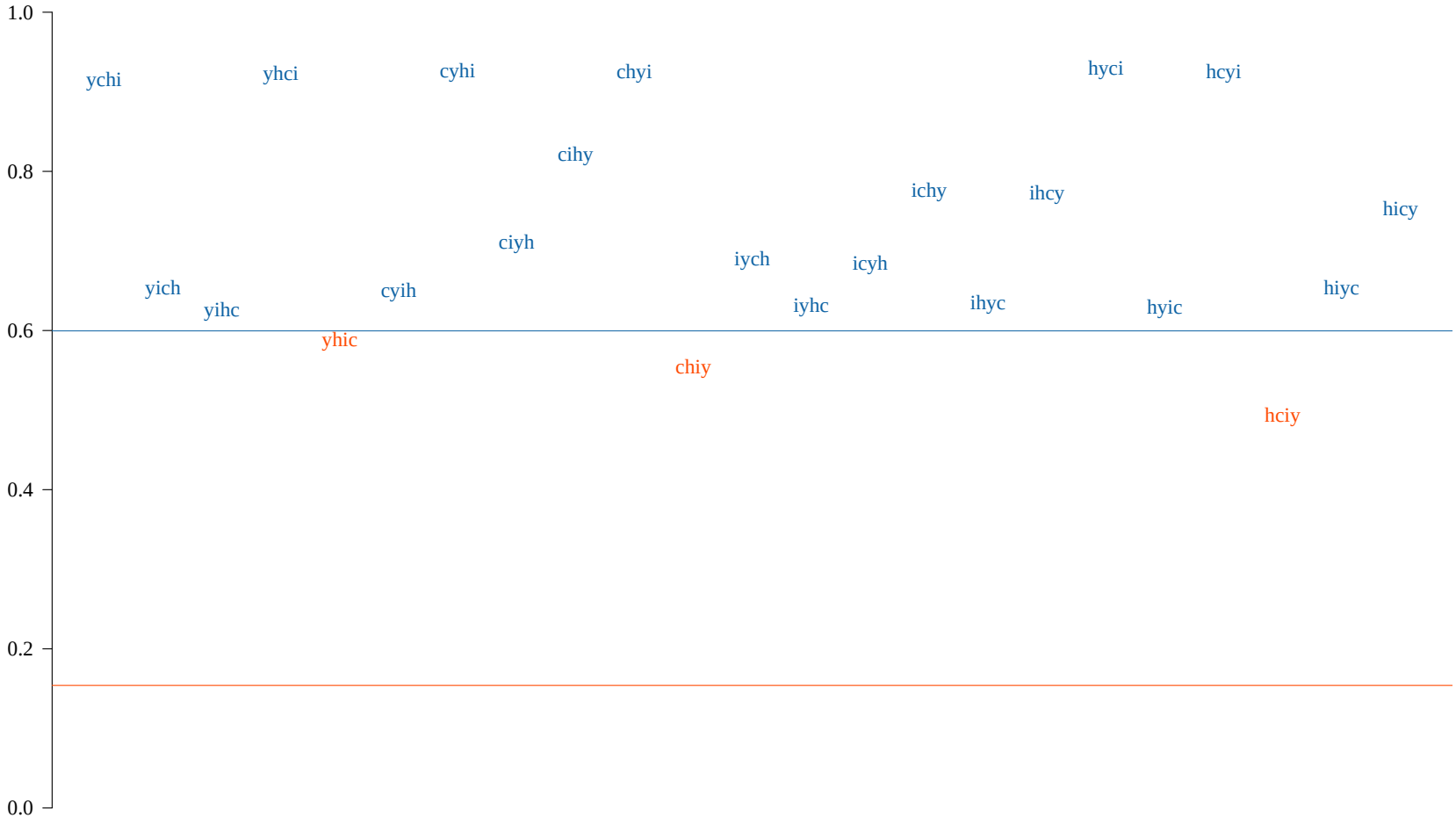


Horizontal line: DSGE w/ real data

The alternative

- We use a simple regularized statistical model
- The model knows nothing about recent economic events
- It took about 10 lines of code and less than 1 second to run
- The DSGE takes about 1000 lines of code and 30 minutes to run

% MSE remaining



DSGE w/ real data Alternative

The moral

Carefully constructed theoretical models are good.

If they match the data well (without overfitting), we can imagine using them for predictions and inference.

Priors and regularization are good. They help us avoid overfitting.

But bad priors can give bad results.

When the prior can't distinguish the data, something is wrong.

Simple models with carefully chosen tuning parameters work better.

Tuning parameter selection for regularized linear models

Motivation (extrinsic):

References for our results

- "Leave-one-out cross-validation is risk consistent for lasso." (with Homrighausen)
- "Risk consistency of cross-validation for lasso-type procedures." (with Homrighausen)
- "The lasso, persistence, and cross-validation." (with Homrighausen)

References for our results

- "Leave-one-out cross-validation is risk consistent for lasso." (with Homrighausen)
- "Risk consistency of cross-validation for lasso-type procedures." (with Homrighausen)
- "The lasso, persistence, and cross-validation." (with Homrighausen)

The general setup

Suppose we have data

$$\mathcal{D}_n = \left\{ Z_1 = (Y_1,\ X_1^\top), \ldots, Z_n = (Y_n,\ X_n^\top) \right\}$$

where

- $X_i = (X_{i1}, \ldots, X_{ip})^\top \in \mathbb{R}^p$ are the features
- $Y_i \in \mathbb{R}$ are the responses (we don't always have a response)
- $Z = (Y,\ X^\top)$ is an iid draw from $\mathcal{D}_n$

We use $\mathcal{D}_n$ to find a function $\widehat{f}$ that can predict Y from X.

Risk

Define $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$ to be a (prediction) loss function

We use ℓ to measure the quality of predictions.

The (prediction) risk is defined to be

$$R(\widehat{f}) = \mathbb{E}_Z \left[\ell(\widehat{f}(X),\ Y) \right] = \mathbb{E}_{Z|\mathcal{D}_n} \left[\ell(\widehat{f}(X),\ Y) \right]$$

Good predictors

The Bayes rule, that is, the best predictor, is

$$m(X) = \operatorname*{argmin}_{\text{measurable } f} R(f)$$

We will generally restrict attention to some class $\mathcal{F}$ of functions.

Then the oracle is

$$f^*(X) = \operatorname*{argmin}_{f \in \mathcal{F}} R(f)$$

Parameterizing this Relationship

Suppose we want the best linear approximation of $m(X)$.

A linear predictor specifies a $\beta \in \mathbb{R}^p$ and forms

$$\widehat{f}(X) = X_1\widehat{\beta}_1 + \ldots + X_p\widehat{\beta}_p = X^\top \widehat{\beta}$$

Thus $\mathcal{F} = \{\beta \in \mathbb{R} : f(X) = X^\top \beta\}$.

Goal: Find $\widehat{f}$ that approximates the predictive performance of

$$f^* = \underset{\beta \in \mathcal{B}}{\operatorname{argmin}}\, \mathbb{E}\left[\ell(X^\top \beta,\ Y)\right]$$

Important: This does not assume that m is linear in X!

We need to find a good estimator of f^*.

ℓ_1-REGULARIZED REGRESSION

Of course, for large p, small n, we need to regularize

Known as

- 'lasso'
- 'basis pursuit'

We use the following estimator of β (and hence f)

$$\widehat{\beta}_t = \underset{\beta}{\text{argmin}} \, ||\mathbb{Y} - \mathbb{X}\beta||_2^2 \text{ subject to } ||\beta||_1 \leq t$$

Alternatively:

$$\widehat{\beta}_\lambda = \underset{\beta}{\text{argmin}} \, ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_1$$

High dimensions, low assumptions

What can be done without linearity/Gaussianity/coherence when $p \gg n \ldots$?

A procedure is persistent (relative to the oracle) if

$$\mathcal{E}(\widehat{\beta}_t, \beta_t^*) = R(\widehat{\beta}_t) - R(\beta_t^*) \xrightarrow{\mathrm{P}} 0$$

If $\ell(\beta^\top X, Y) = (\beta^\top X - Y)^2$, then

- $t^4 = o\left(\frac{n}{\log p}\right)$ implies persistence
  Greenshtein and Ritov (2004)
- $t^2 = o\left(\frac{n}{\log p}\right)$ implies persistence
  Bartlett et al. (2012), ignoring additional log terms

Methods for choosing t

The tuning parameter can be selected by

- degrees of freedom based methods (GIC)

  Zou et al. (2007), Tibshirani, Taylor (2012)

- scaled, sparse linear regression (SSR)

  Sun, Zhang (2012)

- Consistent cross-validation (CCV)

  Feng, Yu (2013)

However...

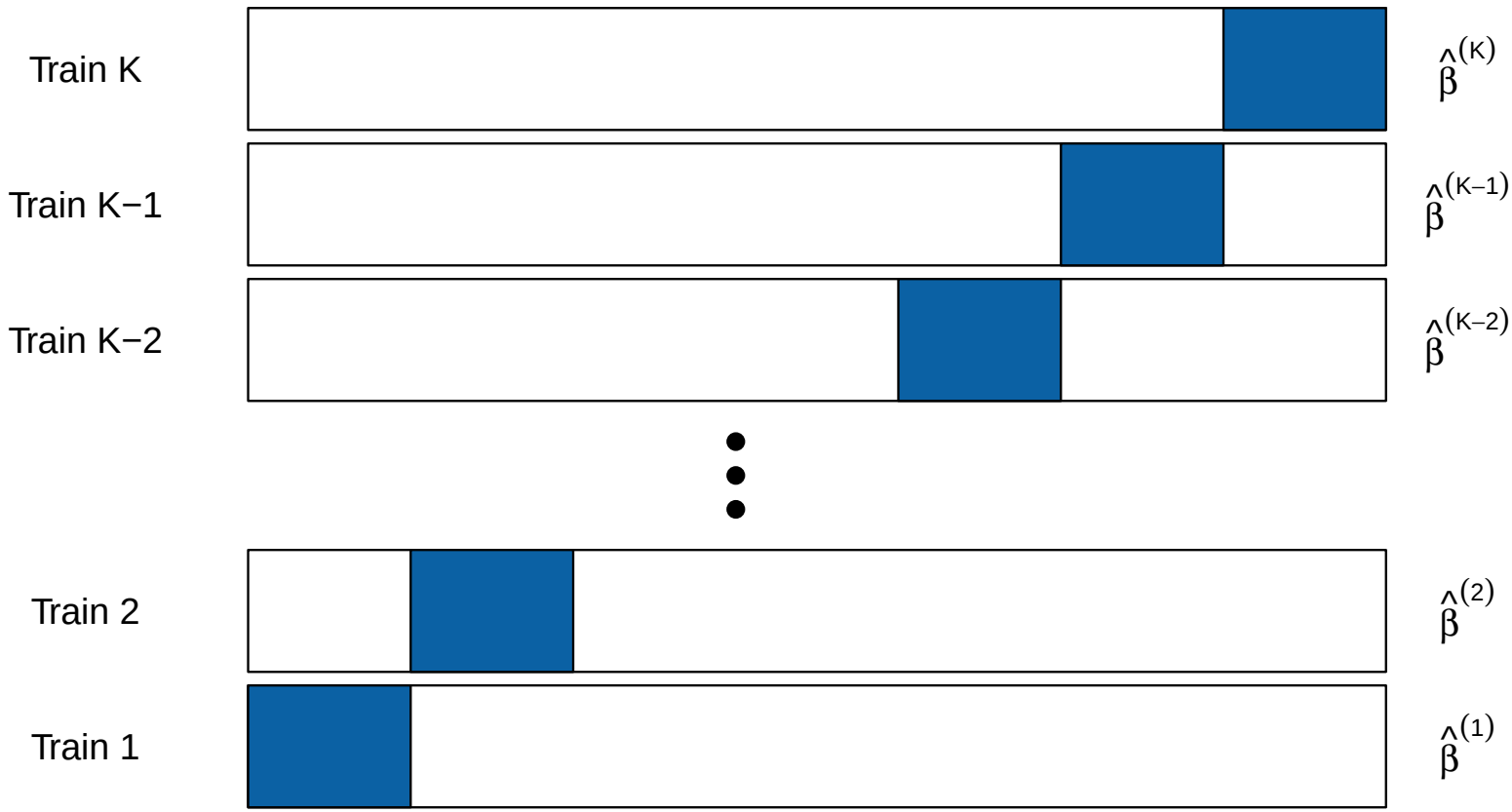
Many papers recommend cross-validation

[It is also the default method in the `R` package `glmnet`. See Zou, Hastie, and Tibshirani (2010)]

Does cross-validation work?

- *In practice, we do not know what is the right value for [t]. Thus, we might want to use cross-validation in order to try various points...* [Greenshtein and Ritov (2004)].
- *One often uses a cross-validation scheme... to select a reasonable tuning parameter t minimizing the cross-validated squared error risk.* [Bühlmann and van de Geer (2011)].
- *Regarding the choice of [t], we typically use* $[\widehat{t}]$ *from cross-validation. 'Luckily', empirical indications... support [good performance]* [Bühlmann's discussion of Tibshirani (2011)]

What is cross-validation?



Cross-validation

Define

- $V_n = \{v_1, \ldots, v_K\}$ to be a set of validation sets
- $\widehat{\beta}_t^{(v)}$ lasso estimator computed on observations not in $v \subset \{1, \ldots, n\}$

The cross-validation estimator of the risk is

$$\widehat{R}_{V_n}(t) = \widehat{R}_{V_n}\left(\widehat{\beta}_t^{(v_1)}, \ldots, \widehat{\beta}_t^{(v_K)}\right)$$
$$:= \frac{1}{K} \sum_{v \in V_n} \frac{1}{|v|} \sum_{r \in v} \left(Y_r - X_r^\top \widehat{\beta}_t^{(v)}\right)^2$$

Define

$$\widehat{t} := \underset{t \in T_n}{\operatorname{argmin}} \widehat{R}_{V_n}(t)$$

Cross-validation

Define

- $V_n = \{v_1, \ldots, v_K\}$ to be a set of validation sets
- $\widehat{\beta}_t^{(v)}$ lasso estimator computed on observations not in $v \subset \{1, \ldots, n\}$

The cross-validation estimator of the risk is

$$\widehat{R}_{V_n}(t) = \widehat{R}_{V_n}\left(\widehat{\beta}_t^{(v_1)}, \ldots, \widehat{\beta}_t^{(v_K)}\right)$$
$$:= \frac{1}{K} \sum_{v \in V_n} \frac{1}{|v|} \sum_{r \in v} \left(Y_r - X_r^\top \widehat{\beta}_t^{(v)}\right)^2$$

Define

$$\widehat{t} := \underset{t \in T_n}{\operatorname{argmin}} \widehat{R}_{V_n}(t)$$

Choosing T_n

In practice, the optimization set $T_n = [0, t_{\max}]$ needs to be specified

However, if $t_{\max}$ is too small, good solutions might be excluded

We choose

$$t_{\max} := \frac{||Y||_2^2}{a_n}$$

See Bühlmann, van de Geer (2011) for motivation

This turns out to "work"

Under some conditions (not very restrictive ones),

Theorem

$$\mathbb{P}\left(\mathcal{E}(\widehat{t}, t_n) > \delta\right) \leq \frac{C}{m_n^2 \delta}\left(1 + \sqrt{\frac{b_n}{n}}\right) + 2e^{-n/2} + e^{-n/8}.$$

In particular, $\mathbb{P}\left(\mathcal{E}(\widehat{t}, t_n) > \delta\right) \to 0.$

In words, if you use Cross Validation, you can do (essentially) as well as if you had used the optimal tuning parameter.

Punch line

Theory papers on Lasso specify a rate known up to unknown constants that guarantee good results.

We say that you can use the data to choose t and do just as well.

Moreover, we allow you to search for t in a random interval.

Justifies standard practice under minimal assumptions.

Statistical and computational efficiency for massive data sets via approximation-regularization

Publications and preprints

- "Approximate principal components analysis of large data sets via the Nyström and column-sampling methods." (with Homrighausen)
- "Regularized PCA for non-i.i.d. survey data." (with Homrighausen and Loewenstein)
- "The Nyström extension for supervised PCA." (with Homrighausen)
- "Preconditioning for least-squares regression." (with Homrighausen)

This work is supported by the National Science Foundation.

Publications and preprints

- "Approximate principal components analysis of large data sets via the Nyström and column-sampling methods." (with Homrighausen)
- "Regularized PCA for non-i.i.d. survey data." (with Homrighausen and Loewenstein)
- "The Nyström extension for supervised PCA." (with Homrighausen)
- "Preconditioning for least-squares regression." (with Homrighausen)

This work is supported by the National Science Foundation.

A big problem

- PCA requires computing an SVD
- So do most other data reduction methods
- Datasets are large: a recent ImageNet contest had $n \approx 10^6$ and $p \approx 65000$[1]
- That was only about 8% of the data set
- SVD requires $O(np^2 + n^3)$ computations
- And you need to store the entire matrix in fast memory
- This is bad.

[1] see e.g. Krizhevsky, Sutskever, and Hinton. *NIPS* 2013

A big problem

- PCA requires computing an SVD
- So do most other data reduction methods
- Datasets are large: a recent ImageNet contest had $n \approx 10^6$ and $p \approx 65000$[1]
- That was only about 8% of the data set
- SVD requires $O(np^2 + n^3)$ computations
- And you need to store the entire matrix in fast memory
- This is bad.

[1] see e.g. Krizhevsky, Sutskever, and Hinton. *NIPS* 2013

Approximations

Can't take the SVD of $\mathbb{X}$

What about approximating it?

Focus on two methods of "approximate SVD"

1. Nyström extension
2. Column sampling

Not our methods.

A quick "sketch" of the intuition

- Suppose we want to approximate $\mathbf{A} \in \mathbb{R}^{q \times q}$
- Assume $\mathbf{A}$ is symmetric and positive semi-definite
- Choose $l \ll q$ and form a "sketching" matrix $\Phi \in \mathbb{R}^{q \times l}$
- Then write $\mathbf{A} \approx (\mathbf{A}\Phi)(\Phi^\top \mathbf{A}\Phi)^\dagger (\mathbf{A}\Phi)^\top$.
- Different Φ yield different approximations
- For Nyström and column sampling use

$$\Phi = \pi\tau$$

Where π is a permutation of $\mathbf{I}_q$ and $\tau = \begin{bmatrix} \mathbf{I}_l & \mathbf{0} \end{bmatrix}^\top$.

Notes on sketching

- Essentially, let

$$\mathbf{S} = \frac{1}{n}\mathbb{X}^\top\mathbb{X} \qquad \text{and} \qquad \mathbf{Q} = \mathbb{X}\mathbb{X}^\top$$

- Both of these are symmetric, positive semi-definite
- Randomly choose l entries in $\{1, \ldots, n\}$ and $\{1, \ldots, p\}$
- Then partition the matrix so the selected portion is $\mathbf{S}_{11}$ and $\mathbf{Q}_{11}$

$$\mathbf{S} = \mathbf{V}\Lambda^2\mathbf{V}^\top = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} \quad \mathbf{Q} = \mathbf{U}\Lambda^2\mathbf{U}^\top = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix}$$

Approximating things we don't care about

If we want to approximate $\mathbf{S}$ (or $\mathbf{Q}$), we have for example

Nyström

$$\mathbf{S} \approx \begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix} \mathbf{S}_{11}^{\dagger} \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \end{bmatrix}$$

Column sampling

$$\mathbf{S} \approx U\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right) \Lambda\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right) U\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right)^{\top}$$

Previous theoretical results have focused on the accuracy of these approximations and the way to randomly select the indices

What we actually want...

We really want $\mathbf{U}$, $\mathbf{V}$, and Λ. Then we can get the principal components, principal coordinates, and the amount of variance explained.

It turns out that there are quite a few ways to use these two methods to get the things we want.

Let

$$L(\mathbf{S}) = \begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix} \qquad L(\mathbf{Q}) = \begin{bmatrix} \mathbf{Q}_{11} \\ \mathbf{Q}_{21} \end{bmatrix}$$

Lots of approximations

After some reasonable algebra...

Quantity of interest	Label	Approximations
$\mathbf{V}$	$\mathbf{V}_{nys}$	$L(\mathbf{S})\mathbf{V}(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})^{\dagger}$
	$\mathbf{V}_{cs}$	$\mathbf{U}(L(\mathbf{S}))$
$\mathbf{U}$	$\mathbf{U}_{nys}$	$L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger}$
	$\mathbf{U}_{cs}$ $\mathbf{U}(L(\mathbf{Q}))$	
	$\widehat{\mathbf{U}}_{nys}$	$\mathbb{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2}$
	$\widehat{\mathbf{U}}_{cs}$	$\mathbb{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2}$
	$\widehat{\mathbf{U}}$	$\mathbf{U}(\mathbf{x}_1)$

So what do we use?

Well...

It depends. We did some theory which gives a way to calculate how far off your approximation might be. You could use these bounds if you like to use your data and make a choice.

Did some simulations too.

Conclusions

- For computing $\mathbf{V}$, CS beats Nyström in terms of accuracy, but is much slower for similar choices of the approximation parameter and d large.
- For computing $\mathbf{U}$, the naïve methods are bad, better to approximate $\mathbf{V}$ and multiply, so see above
- $\widehat{\mathbf{U}}$ really stinks. This is used for supervised PCA. Future research.
- Other choices of Φ (also future research)

Other fun things

Publications and preprints

- "A low-dimensional representation of music" (with Raphael)
- "Support vector machines for spatial data" (with Huang and Li)
- "Sparse logistic regression for structured prediction with an application to U.S. recessions" (with Guo)

Publications and preprints

- "A low-dimensional representation of music" (with Raphael)
- "Support vector machines for spatial data" (with Huang and Li)
- "Sparse logistic regression for structured prediction with an application to U.S. recessions" (with Guo)

The question:

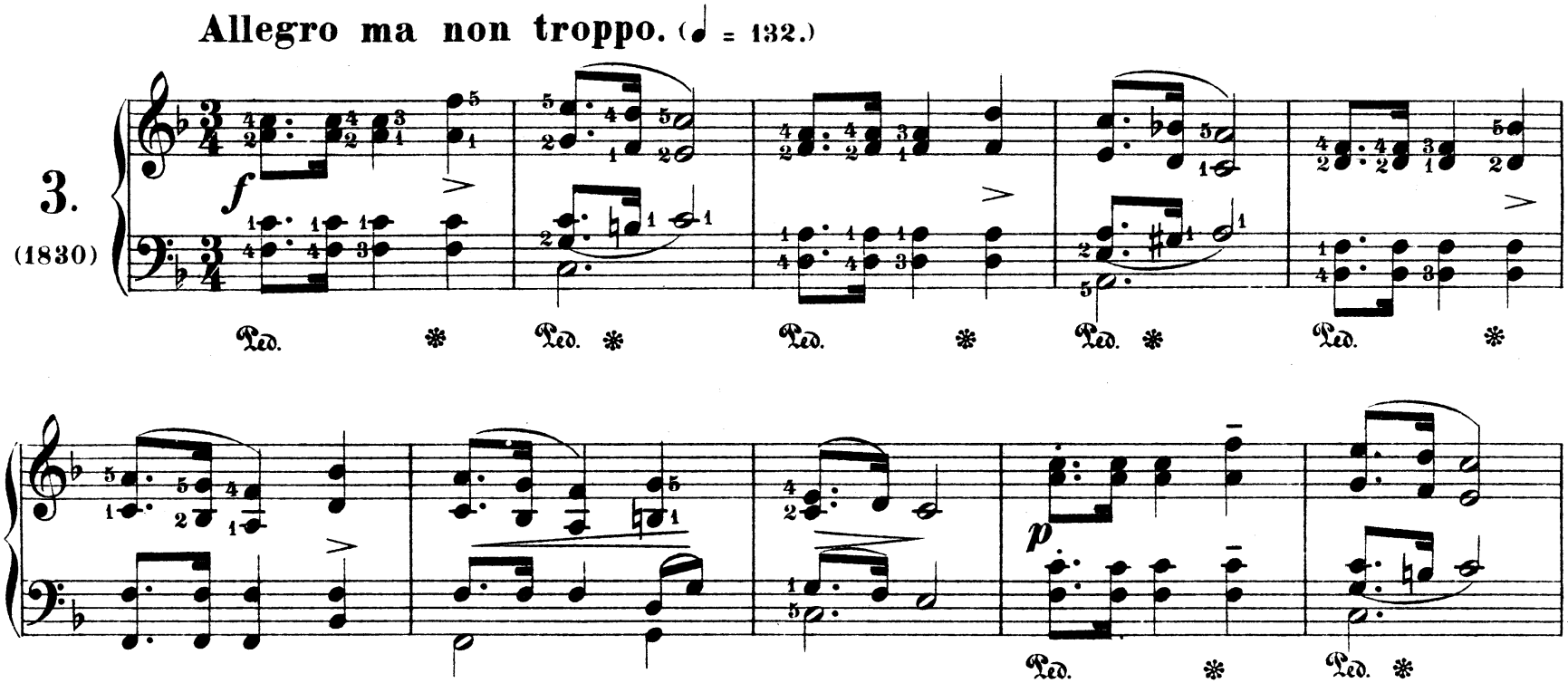
- Easy to describe musical characteristics you like: "up-tempo", "strong beat", "good lyrics", "jazzy", etc.

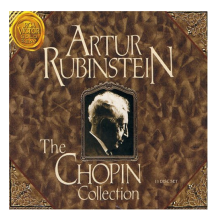
- Harder to describe characteristics of a performance that you like.

- In classical music, there are hundreds or thousands of recordings of the same piece.

- Why do we like some better than others?

Example


Allegro ma non troppo. (♩ = 132.)
3.
(1830)
f
p
Ped.


Artur Rubinstein
The Chopin Collection

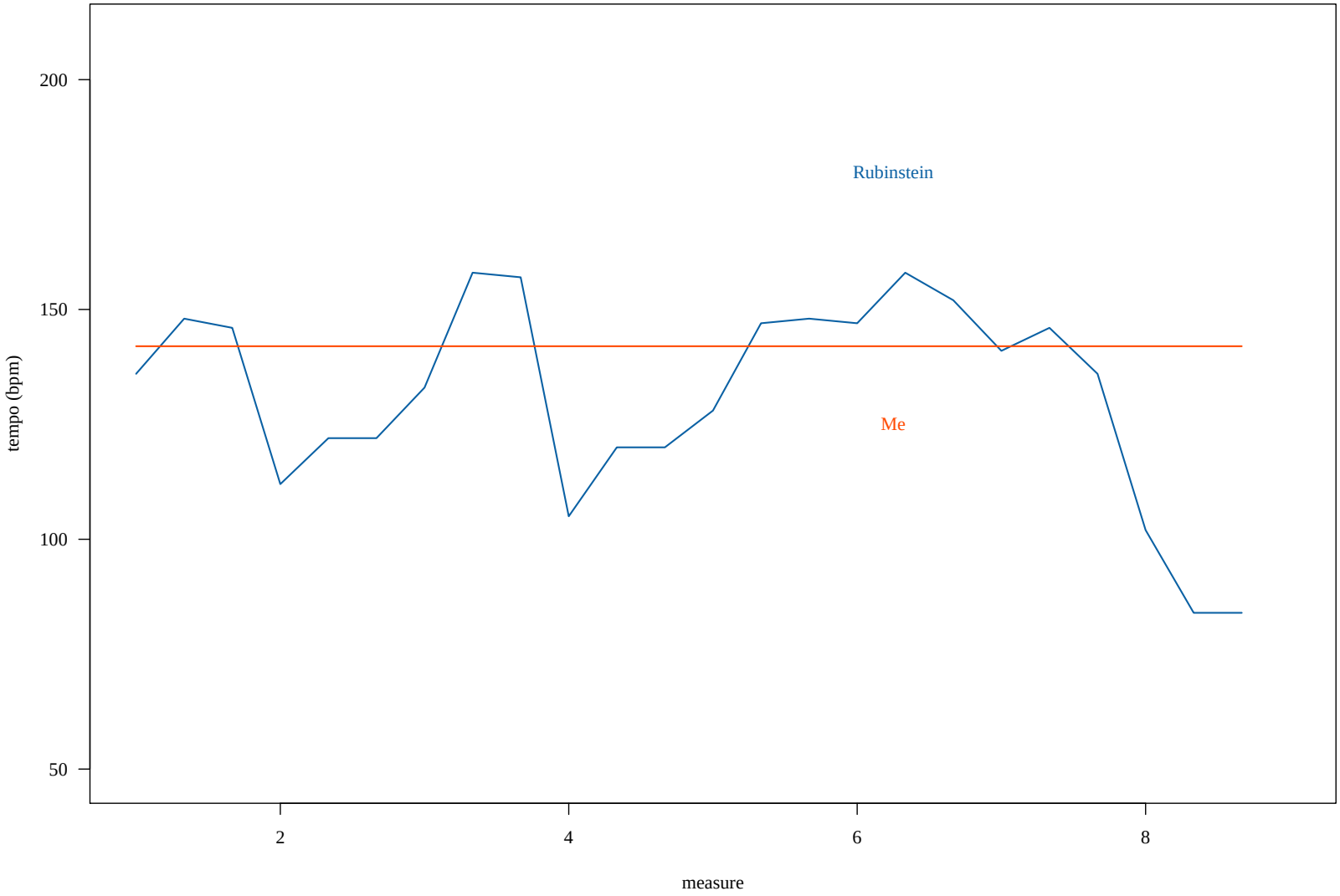
What's different?

1. Mistakes
2. Extraneous noise
3. Recording quality
4. Articulation/Legato/Bowing/Breathing
5. Dynamics
6. Rubato/Tempo

The first three are mostly uninteresting, but the rest are about interpretation.

We like performances with "better" interpretations.

Performances



Data

- CHARM Mazurka Project
- Focus on timing and dynamics
- 46 recordings: Chopin Mazurka Op. 68 No. 3
- Recorded between 1939 and 2006
- 41 different performers

Define terms

Notes All those little black dots

Beat Strongly felt impetus

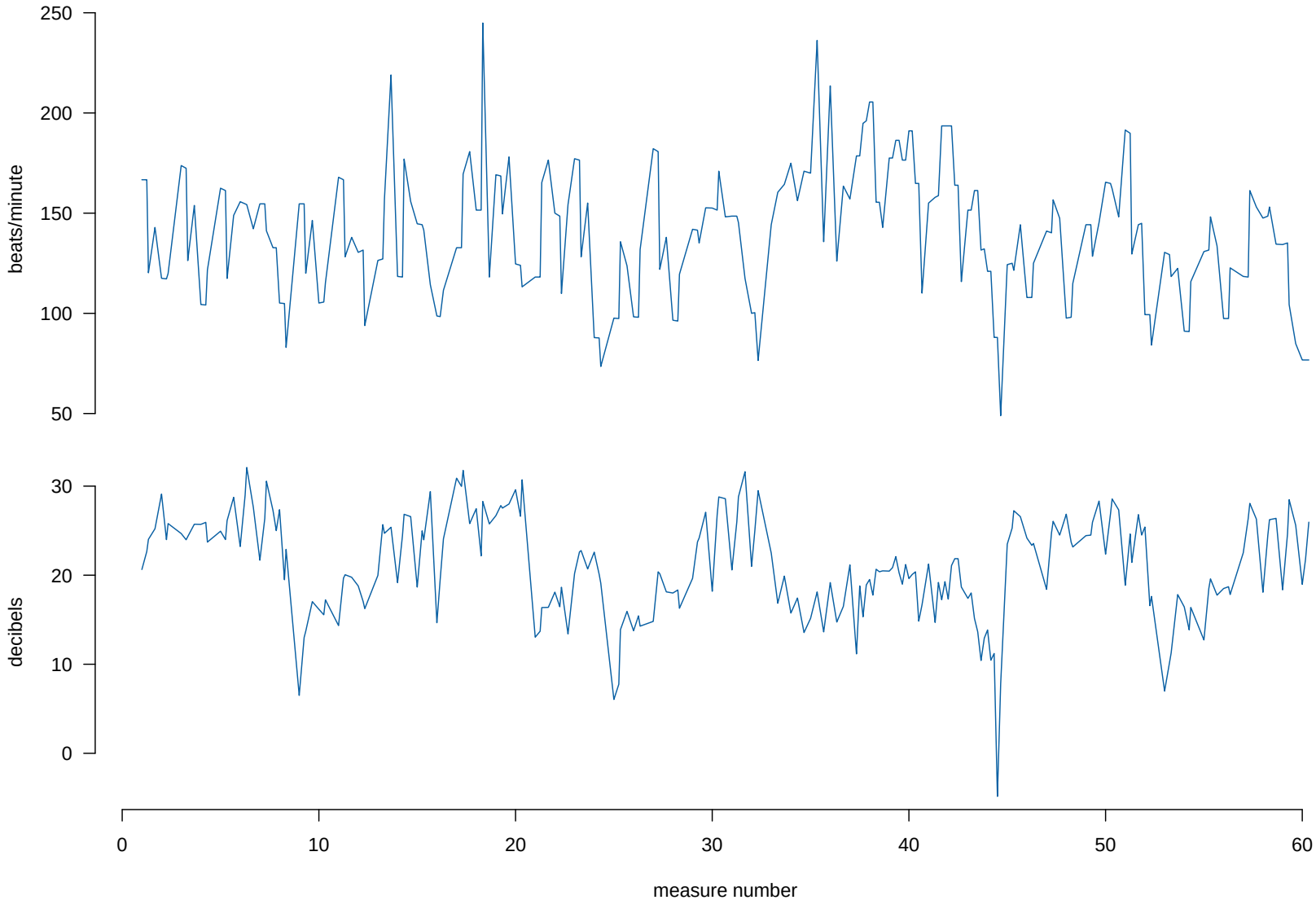
Tempo The speed of the piece, usually measured in bpm

Measure Short collections delimited by vertical lines

Time signature The number of beats per measure

Dynamics The loudness of the notes

Bivariate functional data



Analyzing functional data

- Functional data $X(t)$ is described by a sequence of observations indexed by t in some set $\mathcal{T}$
- Thus, we view $X(t)$ as a function of t
- Think of $X(t)$ as a continuous process and t indexing time
- We can't observe the continuous function, but we get observations at a discrete set $t_1, \ldots, t_p$
- We likely have prior information about how X depends on t (eg. smoothness, periodicity, etc.)
- Treating $(X(t_1), \ldots, X(t_p))$ simply as a vector of random variables ignores this information (the correlation structure)

Basis projection

Given functional data, we can often use our prior information to compress the data into more manageable representations.

These representations may be more interpretable or simply more useful.

Given a some functions $\phi_1, \ldots, \phi_k$, we can find coefficients $\theta_1, \ldots, \theta_k$ and write

$$X(t) = \sum_{i=1}^{k} \theta_i \phi_i(t) = \Theta^\top \boldsymbol{\phi}(t)$$

if $X \in \mathcal{X}$ and $\boldsymbol{\phi}$ spans $\mathcal{X}$.

Of course, we only have some prior information about $\mathcal{X}$, so we want to choose a collection wisely in order to approximate X well with it's projection onto span($\boldsymbol{\phi}$).

Basis projection

Given functional data, we can often use our prior information to compress the data into more manageable representations.

These representations may be more interpretable or simply more useful.

Given a some functions $\phi_1, \ldots, \phi_k$, we can find coefficients $\theta_1, \ldots, \theta_k$ and write

$$X(t) = \sum_{i=1}^{k} \theta_i \phi_i(t) = \Theta^\top \boldsymbol{\phi}(t)$$

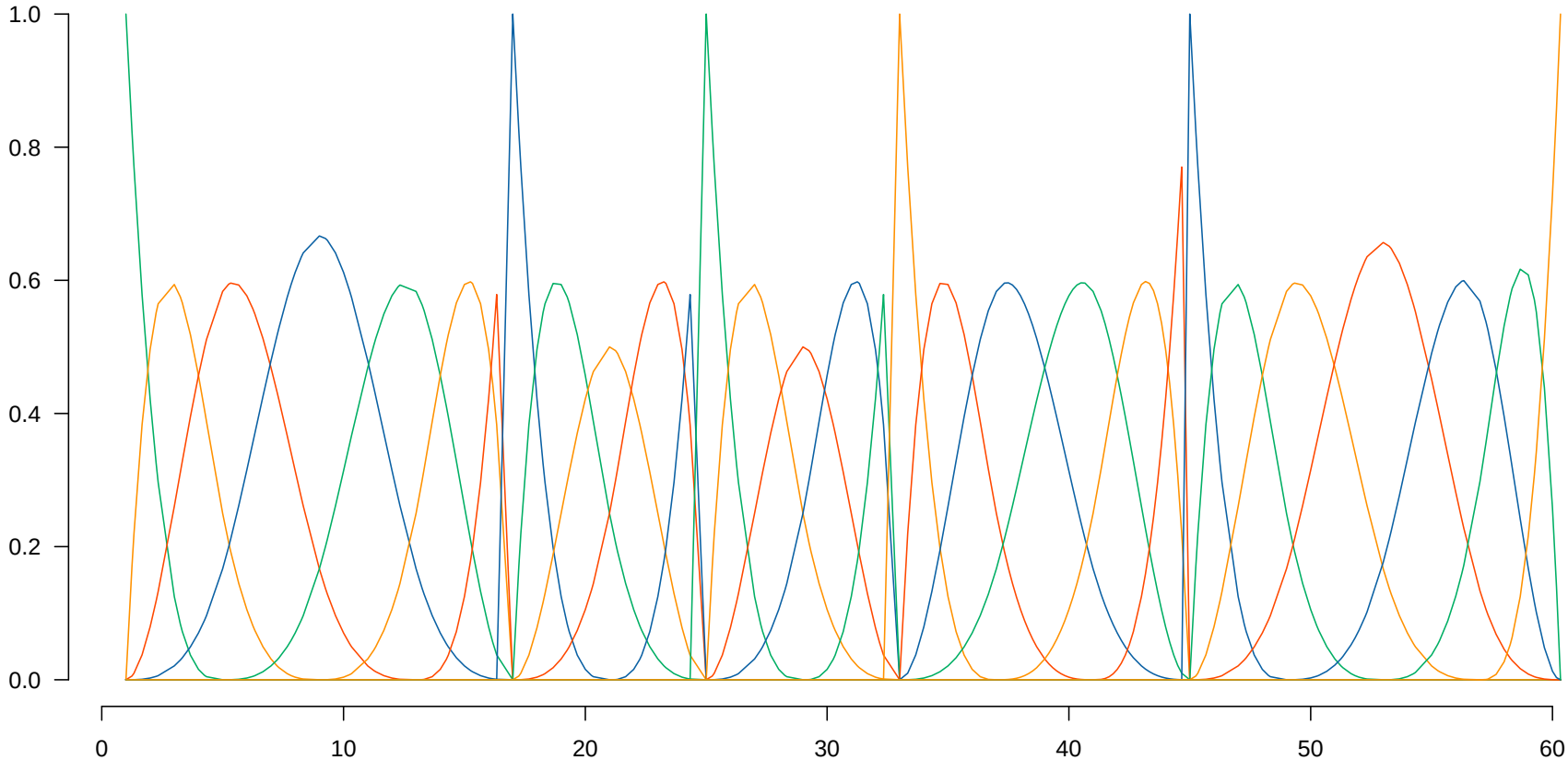
if $X \in \mathcal{X}$ and $\boldsymbol{\phi}$ spans $\mathcal{X}$.

Of course, we only have some prior information about $\mathcal{X}$, so we want to choose a collection wisely in order to approximate X well with it's projection onto span($\boldsymbol{\phi}$).

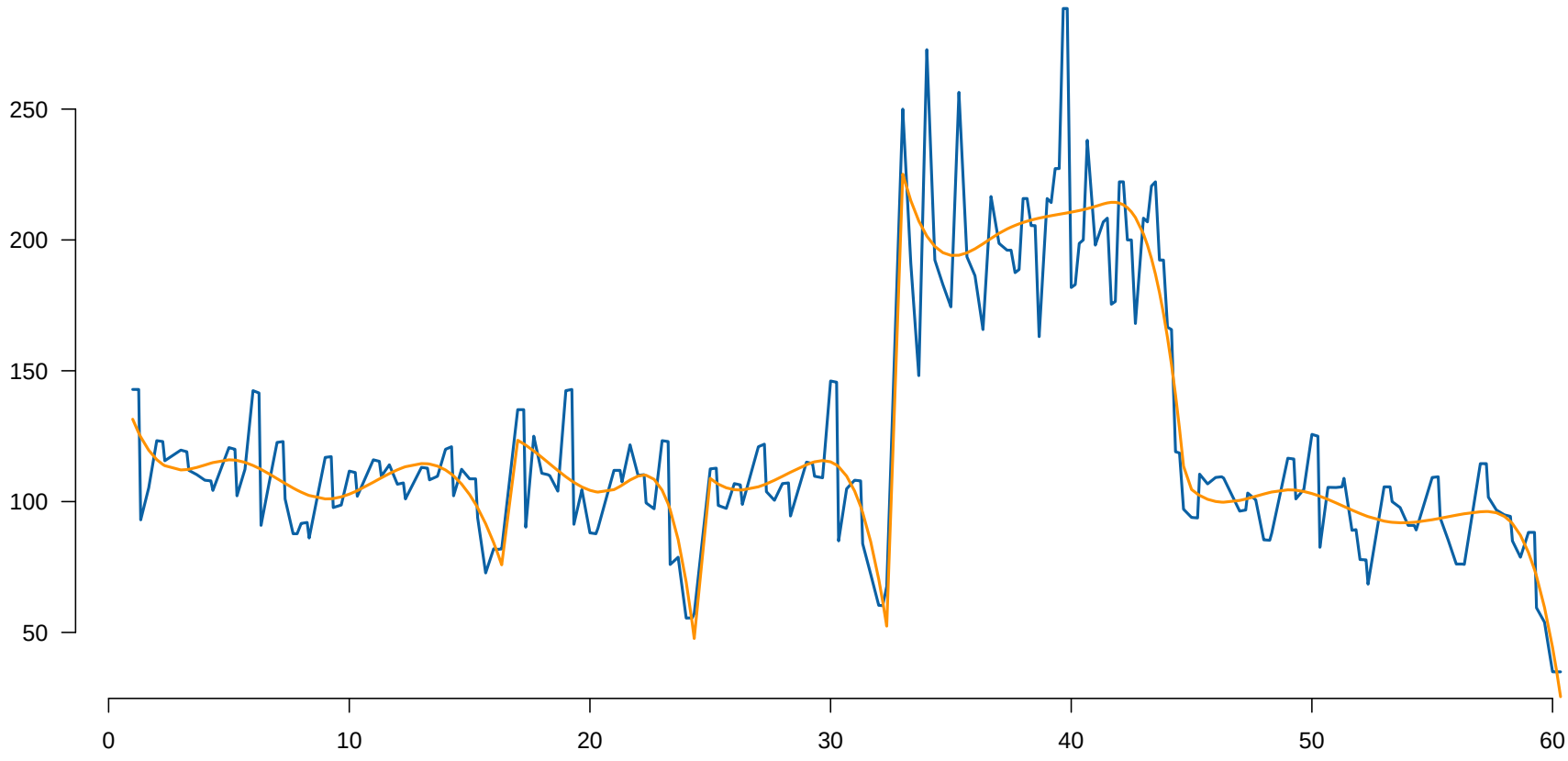
An ideal basis

1. Smooth over small portions of the musical "space"
2. Able to express dramatic, instantaneous changes
3. Able to handle irregularly spaced observations
4. Computationally simple
5. Hierarchical or "multi-scale"

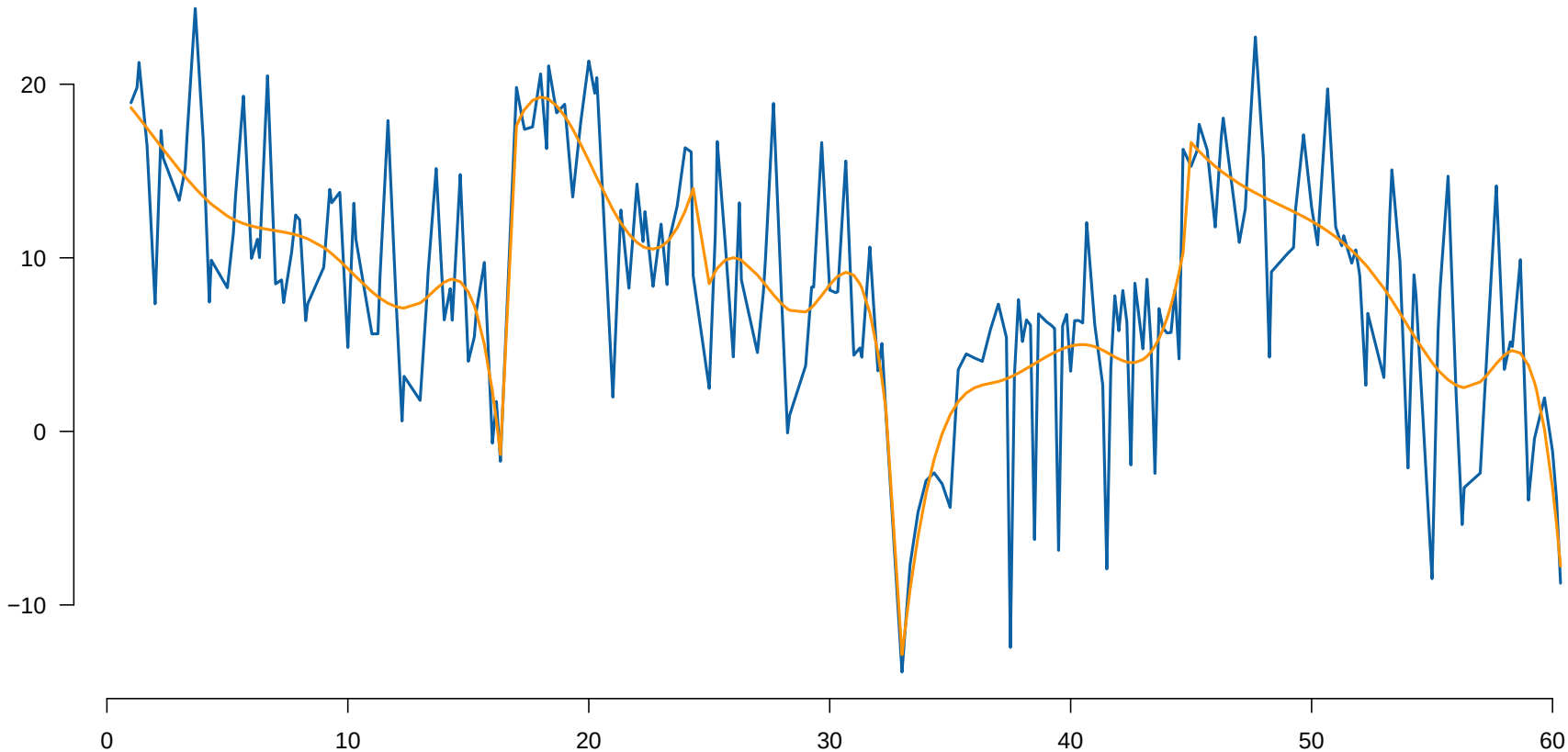
Use musical structure to choose a basis



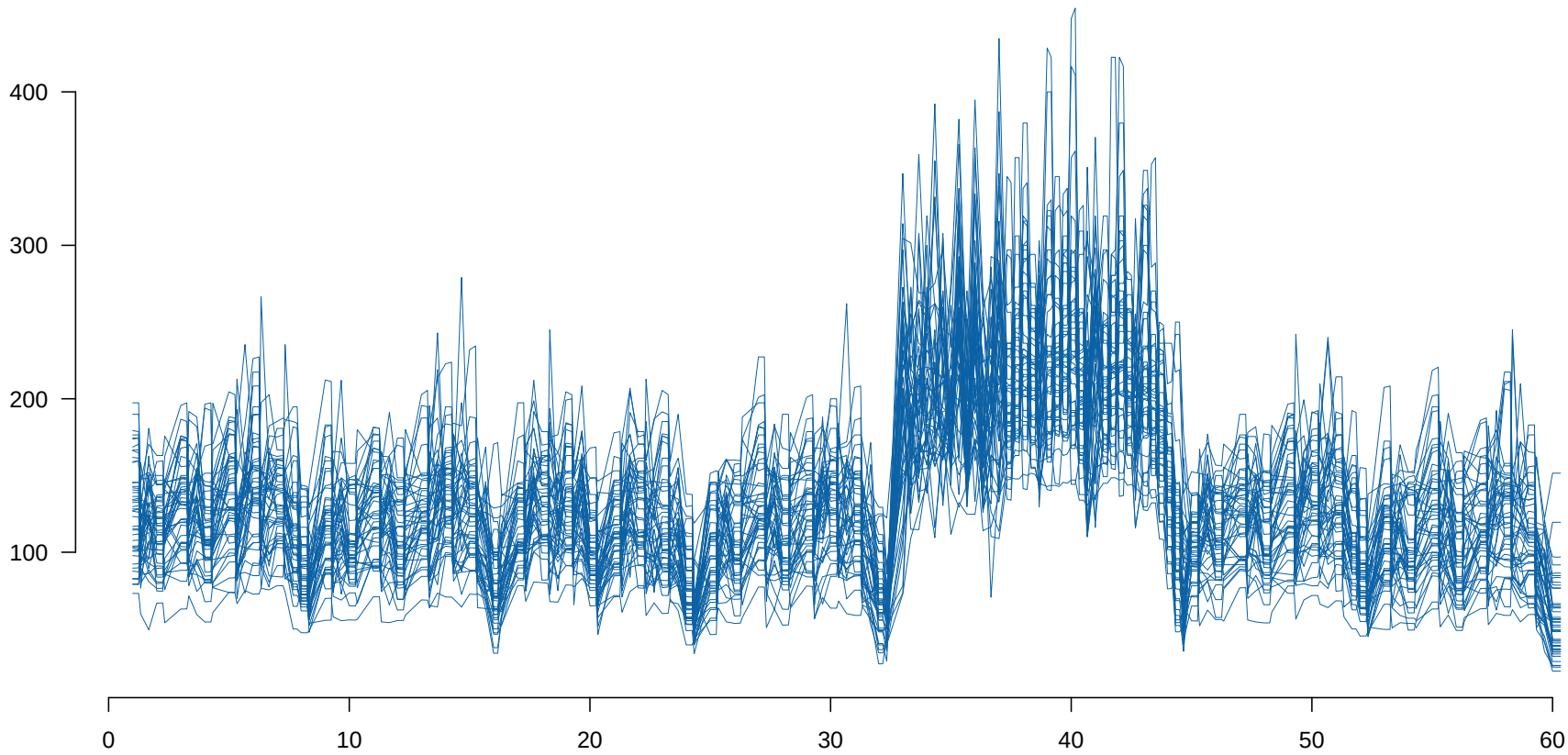
Reconstruction (Joyce Hatto 1993)



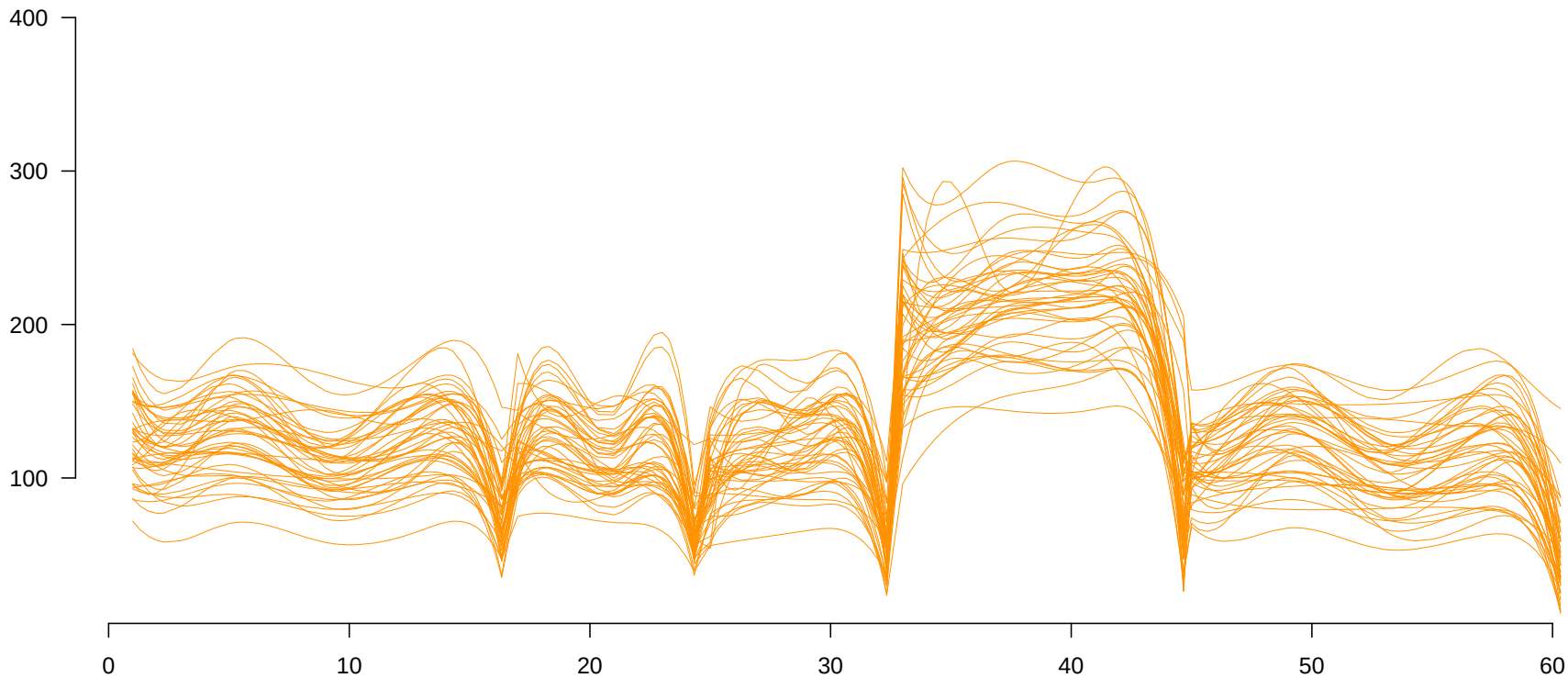
Reconstruction (Joyce Hatto 1993)



Reconstructed curves



Reconstructed curves



Did lots of other stuff

- Used functional data analysis and sparse clustering to try to find "similar" performances
- The clustering algorithm performs ok, but the solutions aren't as "clean" as I would like (PCA preprocessing not much better, down-weights dynamics)
- Would be interesting to dynamically the basis
- Perhaps cluster on both knot position and magnitude (each cluster would have common positions)
- Given a clustering, ask some musicians to listen, validate, critique, provide preferences

Wrap up

To summarize

There's lots of interesting data out there that requires good machine learning methods.

Very often, statisticians who collaborate with applied scientists can make giant leaps relative to current practice.

Machine learning without a clear handle on the application is not super fun to me.

Solving interesting problems is fun. See me if you have some ideas or want to run simulations.

The End.