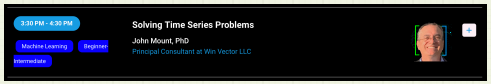
Solving Time Series Problems

John Mount, Ph.D.
Win Vector LLC
jmount@win-vector.com
https://www.win-vector.com

https://odsc.com/california/odsc-west-schedule-2024/
Tutorials | Workshops, 30 Oct Day 2. 3:30 PM
(Hyatt Regency San Francisco Airport, 1333 Old Bayshore Highway, Burlingame, CA 94010)
All slides, code and data are shared here: https://github.com/WinVector/Examples/tree/main/TimeSeries#readme



Motivation

"One of the biggest mistakes in my data science career was not paying enough attention to time series models.

Time series is everywhere—finance, sales, marketing—and it can drastically improve your decision-making process."

– Mark Eltsefon, Staff Scientist at Meta
https://www.linkedin.com/posts/mark-eltsefon_datascience-timeseries-activity-7253363004694491137-94D8?utm_source=share&utm_medium=member_desktop

---

Outline

- Motivation
- Who I am
- What is time series forecasting?
- Families of methodologies
- Our example problem
- The liar's graph and out of sample evaluation
- Solving the problem using Stan
- What is Stan?
- Results/Observations


Luca Cambiaso FightingFigures

We will fill in some details as we go.

Who I am

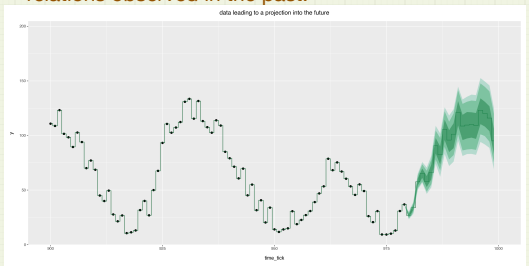
- John Mount, General Partner at Win Vector LLC.
  - Co-author of *Practical Data Science with R*, 2nd edition, Manning, 2020.
  - Co-author of the `vtreat` R package for re-encoding high cardinality explanatory variables.
- Win Vector LLC is a statistics, machine learning, and data science consultancy and training organization.
  - Specialize in solution design, technology evaluation, and prototyping.
  - We help greatly speed up solution and production deployment.
  - Looking for some more engagements.
    - Please contact: jmount@win-vector.com .
- Please follow us on the Win Vector blog: https://win-vector.com/blog-2/

What is Time Series Forecasting?

(a slightly heterodox view)

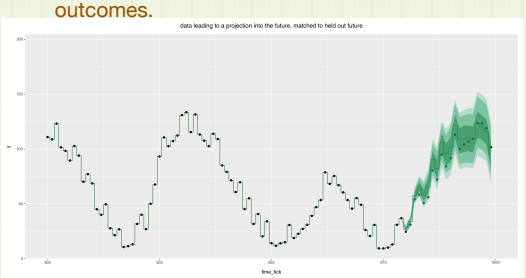
Time Series Forecasting

- Produce an estimate of the future using trends and relations observed in the past.



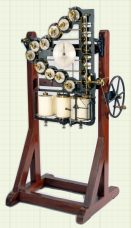
How to Evaluate Forecasts

- Hold out future, and compare prediction to eventual outcomes.



Business Applications

- Predicting future demand
  - Online sales
  - Parking
  - Restaurant
- Predicting future price
  - Expenses
  - Stocks
- Predicting future revenue
  - Financial planning
- Physical systems affecting business
  - Tides
  - Weather

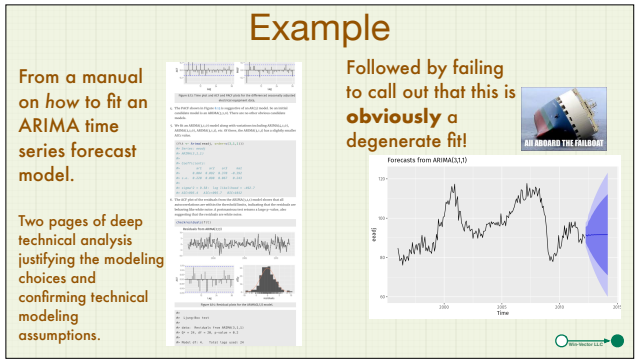


In 1876 A. Légé & Co., 20 Cross Street, Hatton Gardens, London completed the first "tide calculating machine" for William Thomson (later Lord Kelvin)

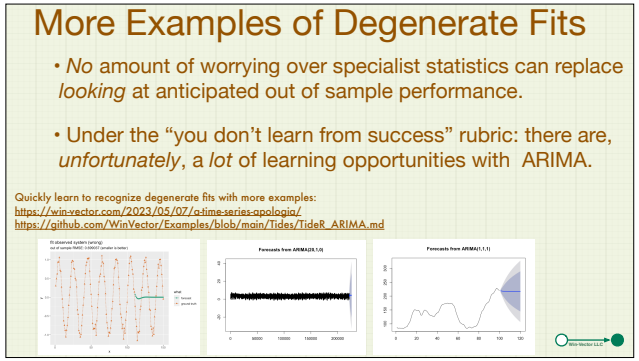
de-Motivation

"We have found that choosing the wrong model or parameters can often yield poor results, and it is unlikely that even experienced analysts can choose the correct model and parameters efficiently given this array of choices."

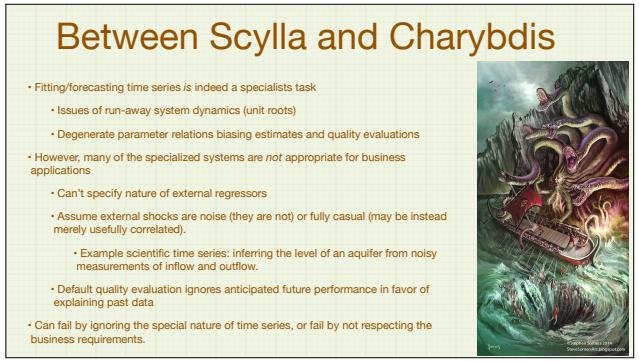
– Facebook research announcement of Prophet, 2017
https://research.facebook.com/blog/2017/2/prophet-forecasting-at-scale/



At no point in the training period was the curve flat. But that is precisely the predicted expected value. Yes, models of expected tend to have less variance than actuals https://win-vector.com/2024/06/15/good-models-do-not-match-variance/ . However, this is a degenerate fit. Most references avoid making the obvious by the simple expedient of never showing the final graph. Can't over emphasize the importance of insisting on method-agnostic hold out tests.



Each of these is a degenerate fit. Some are from "auto ARIMA" which claims to pick the right parameters. Once of these is once again an example from package help, not marked as a failure!



Families of Methodologies

Methodology Families

- ARIMA derived methods
- Decomposition or attribution methods
- Brute force methods
- Hidden state inference

ARIMA derived methods

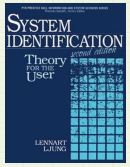
- "Auto regression integrated moving average"
- Fit for a conserved relation between observations
  - AR example $\hat{y}_t = \hat{\mathrm{E}}[y_t] = 1.975y_{t-1} - y_{t-2}$ for many t
  - Push that forward as our prediction

$$\hat{y}_t = 1.975y_{t-1} - y_{t-2}$$
$$\hat{y}_{t+1} = 1.975\hat{y}_t - y_{t-1}$$
$$\hat{y}_{t+2} = 1.975\hat{y}_{t+1} - \hat{y}_t$$
$$\cdots$$
$$\hat{y}_{t+k} = 1.975\hat{y}_{t+k-1} - \hat{y}_{t+k-2}$$

  - Can run away!
- Regression with ARIMA residuals:

$$(1 - \phi_1 B - \phi_2 B^2)(y_t - \beta_0 - \beta_1 x_t - \beta_2 z_t) = c + (1 - \theta_1 B - \theta_2 B^2)\epsilon_t$$

- Very powerful
  - Can dominate other solution method when *only* measuring loss or fit quality.
- Highly technical (avoiding issues such as "unit roots").



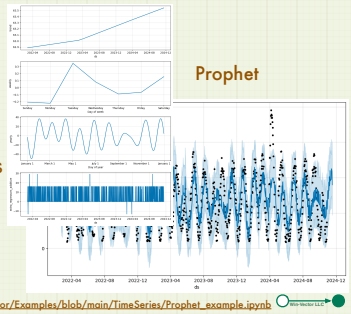
https://github.com/WinVector/Examples/blob/main/TimeSeries/ts_example.md
https://github.com/WinVector/Examples/blob/main/TimeSeries/sm_example.ipynb

Among the most powerful methods. The "get k steps out by running a 1 step out process k times" is very powerful, but also very sensitive.

Decomposition or Attribution

- Fit past to things we know future values of:
  - Month of year …
- Use the combination of the future values of these things as our future prediction.

Prophet



https://github.com/WinVector/Examples/blob/main/TimeSeries/Prophet_example.ipynb

Fit the past to things we know the future of: such as seasonal averages. Use this fit to project into the future. Very useful in attributing or explaining past effects. Tends to under-perform ARIMA.

Brute force

- Treat as a standard machine learning problem
  - Model $\hat{y}_{t+k} = \hat{\mathrm{E}}[y_{t+k}] = f_k(y_t, \cdots y_{t-w})$ for $k = 1 \ldots u$
  - Fit for all the $f_k()$
    - Linear methods
      - https://win-vector.com/2023/05/07/a-time-series-apologia/
    - Neural net methods such as LTSM

https://github.com/WinVector/Examples/blob/main/TimeSeries/nested_model_example.ipynb

Not actually a bad idea. Kind of “not allowed” until deep learning had great success with it.

Hidden State Inference

- Includes methods such as EM and particle filters
- Guess hidden state and parameters β, ζ

Estimate $\hat{\beta} = \hat{\mathrm{E}}[\beta] = \dfrac{\int_{\beta,\zeta} \mathrm{d}_{\beta,\zeta} \mathrm{P}[\text{training data} \mid \beta, \zeta]\beta}{\int_{\beta,\zeta} \mathrm{d}_{\beta,\zeta} \mathrm{P}[\text{training data} \mid \beta, \zeta]}$

https://github.com/WinVector/Examples/blob/main/TimeSeries/nested_model_example.ipynb

Computationally expensive, but very versatile. This is the method we will pursue in this session.

Opinion

- Good time series and system identification solutions combine a few ideas from a few of these methodological themes.
- Many of the methodologies simplify if you delegate the solution to a systematic solver.
- Most of the solving methods are *excessively* technical and restrictive.
  - There *are* in fact issues with time series forecasting require expert packages.
  - You want the solution correctness *fully* delegated to the solver, and *not* something the user is expected to do or check.
  - If the package wasn't built for your application, you *may not* be able to adapt it to your application. You are instead forced to adapt your goals to the package.
    - Example: a Kalman filter package designed to de-noise measuring the height of water in damn isn't going to be your best bet in predicting future restaurant visits.
- A case where there are *more* packages in Python, but more *good* packages in R.

Notice that none of the methods are “just a linear regression”

Our Example Problem

External Regressors

- For many business applications we need to model future values as a function of past values *and* additional facts called “external regressors.”
- This is where it all goes *wrong*.
  - ARMAX, ARIMAX, SARIMAX, transfer functions, regression with ARIMA residuals all *claim* to solve this.
  - No two of them seem to be solving the same problem (if you can even find a description of what problem they claim to solve).
    - https://win-vector.com/2024/07/15/armax-offerings-remain-a-muddle/
    - The ARIMAX model muddle https://robjhyndman.com/hyndsight/arimax/
- To the extent time series modeling is magic, it obscures out the effects of external regressors.

There is a fiction that ARMAX is how time series researches work. That appears to be wishful thinking. Time series research moved on to transfer functions and conformal adapters (don’t ask).

Our Example Problem

- We are modeling a restaurant
  - We imagine number of diners on a given day is related to number of guests the most recent same day of week.
- Improve prediction *and* utility of prediction by bringing in information
  - We have transient external regressors such as events being near our restaurant.
  - We have durable external regressors such as price changes and restaurant reviews.
- Can also model this as two sub-populations of guests: loyal and transient
  - The issue is: these subpopulations are not labeled in our data!

Win-Vector LLC

As Equations

$$\text{loyal\_guests}_{date} = \beta_{loyal} + \beta_7 \, \text{loyal\_guests}_{date-7} + \beta_{14} \, \text{loyal\_guests}_{date-14} + \beta_{review} \, x_{review,date-7}$$
$$\text{transient\_guests}_{date} = \beta_{imp} + \beta_{events} \, x_{events,date}$$
$$\text{total\_guests}_{date} = \text{loyal\_guests}_{date} + \text{transient\_guests}_{date}$$

- These equations are our specification of the problem. Under different assumptions we would write different equations.
  - Subtle point: we are modeling realized values, not the traditional expected values.
  - We only observe total_guests (not loyal_guests or transient_guests).
  - In practice we benefit from a *lot* more external regressors- such as reservations by date.
- Without controllable external regressors forecasting devolves into Cassandra's curse.

Cassandra by Evelyn De Morgan (1898, London)

Win-Vector LLC

Definitely need some care in accounting for where the past ends and the future starts.

The Equations in Detail

$$\text{loyal\_guests}_{date} = \beta_{loyal} + \beta_7 \, \text{loyal\_guests}_{date-7} + \beta_{14} \, \text{loyal\_guests}_{date-14} + \beta_{review} \, x_{review,date-7}$$
$$\text{total\_guests}_{date} = \text{loyal\_guests}_{date} + \beta_{imp} + \beta_{events} \, x_{events,date}$$

- $x_{review,date-7}, x_{events,date}$ features engineered by the data scientist.
  - Notice we only know about reviews in the past, but claim to know about events in the future.
- $\text{total\_guests}_{date}$ observed
- $\text{loyal\_guests}_{date}$ unobserved (to be inferred)
  - Classic Bayesian set up. Our population is a mixture of unlabeled sub-populations with different behaviors. Could divide into even more sub-populations if we cared.
- $\beta_{loyal}, \beta_{imp}, \beta_7, \beta_{14}, \beta_{review}, \beta_{events}$ the linear model coefficients to be inferred
  - We call $\beta_{loyal}, \beta_7, \beta_{14}$ the "durable" auto-regressive portion of the model.
  - We call β_{review} a "durable" effect, associated with a durable external regressor.
  - We call β_{events} a "transient" or "impermanent" effect, associated with a transient external regressor.

Win-Vector LLC

How can we know events in the future?

- Buy the information from somebody that curates records about future events.

PredictHQ

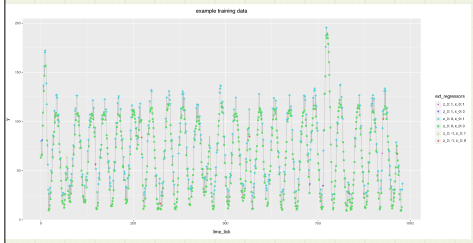
(Not an endorsement, I just used them for a couple of clients.
Some notes of mine on problems in using *past* predictions of the future:
https://win-vector.com/2024/09/09/please-version-data/)

Can Standard Packages Solve This?

$$\text{loyal\_guests}_{date} = \beta_{loyal} + \beta_7\ \text{loyal\_guests}_{date-7} + \beta_{14}\ \text{loyal\_guests}_{date-14} + \beta_{review}\ x_{review,date-7}$$
$$\text{total\_guests}_{date} = \text{loyal\_guests}_{date} + \beta_{imp} + \beta_{events}\ x_{events,date}$$

- If we force $\beta_{review} = 0$, then this is "regression with ARIMA residuals" variation of ARIMAX.
- If we force $\beta_{events} = 0$, then this is the sort of system social scientists want to study policy changes (such as tariffs or seatbelt laws). Also called an ARMAX variation (may or may not in fact be).
- None of the ARIMA style packages we surveyed offered structural *choice or control*. The equations the package solves either match yours, or do not.

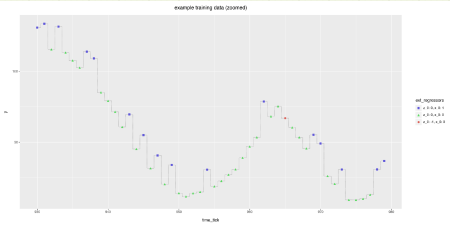
An Artificial Example



- Observed total count y
- Unobserved loyal customers y_{auto}
- External regressor z adds or removes loyal customers
  - modeling price changes, reviews, etc.
- External regressor x adds impermanent or transient customers
  - models people in the area for an event
- Problem more regular, and less noisy than most real world applications.

Time series methods tend to be really good and modeling sin(x). So if they are not good and noised up copies of sin(x), what good are they going to be on brutal real world data?

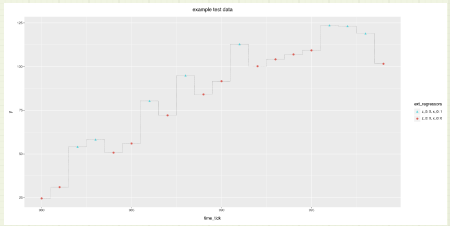
Closer In



Problem specification

```
{'b_auto_0': 1.28,
 'b_auto': (1.975, -1),
 'b_z': (14.2,),
 'b_x': (16.1,),
 'generating_lags': (1, 2),
 'modeling_lags': (1, 2)}
```


Held Out Test Data

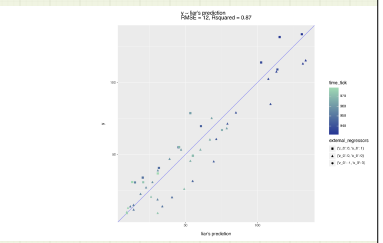


- To cross-validate it is traditional (in finance) to design the system to be able to hold out arbitrary large contiguous blocks of time.
    - I.e. train the model in the future and see if it correctly works in the past.

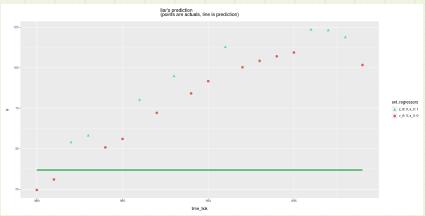
The Liar's Graph and Out of Sample Evaluation

The Liar's Graph

- If you take away one thing today, please take away this.
  - Most common scatter plots of "time series performance" plot how well $\hat{y}_t \sim y_t$.
  - This is often "one tick out performance" even if your application requires predicting many time periods out!
- Do NOT get conned by the complicated diagnostics and the scatter plot. Insist on seeing the model applied.



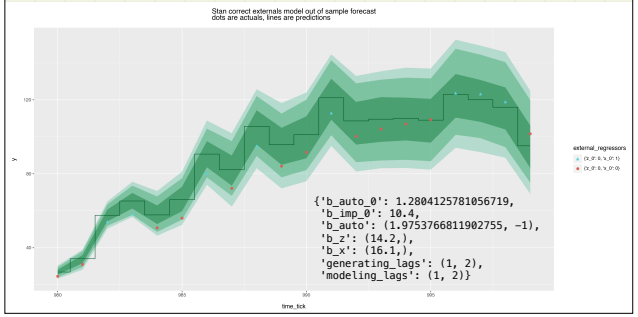
Look at Out of Sample Predictions!



- This and the scatter plot on the previous slide are plotting the performance of the model $\hat{y}_t = y_{t-1}$!
- Once you get out of the training region this is just the horizontal line.
- It is a "good model", just not implantable; as eventually you do not know y_{t-1}

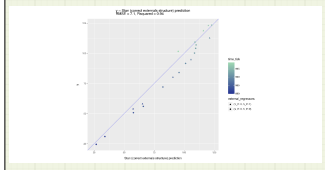
Solving the Problem

Stan Result

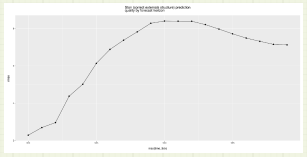


In this, all predictions are made with only information available at time = 979.

Quantitative Metrics



Not the liar's graph, the predictions are all 20 points of the test period predicted *prior* to entering the test period. So the k^{th} prediction is k steps into the future.



How we did that

- Copy our equations into "Stan"
- Have Stan solve for or infer both plausible hidden state and parameters.

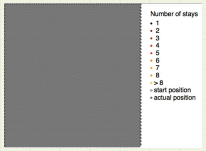
```
y_auto[3:(N_y_observed + N_y_future)] ~ normal(
  b_auto_0
   + b_auto[1] * y_auto[2:(N_y_observed + N_y_future - 1)]
   + b_auto[2] * y_auto[1:(N_y_observed + N_y_future - 2)]
   + b_x_dur[1] * x_dur_1[3:(N_y_observed + N_y_future)],
  b_var_y_auto);
y ~ normal(
  b_imp_0 + y_auto
   + b_x_imp[1] * x_imp_1,
  b_var_y);
```

Complete workbook: https://github.com/WinVector/Examples/blob/main/TimeSeries/Stan_soln.ipynb

What is Stan?

Stan

- A Markov Chain Monte Carlo sampler
- Can be used for complicated Bayesian inference
  - Guesses likely values of parameters, and nuisance variables, *and unobserved intermediate state* conditioned on observed data.
  - Returns distributional answers.
  - Subsumes the implementation portion of many other systems/ ideas (such as "particle filters" and "importance sampling").
- Fairly high dependency (requires C++ compiler and linker)
  - May eventually see competition from Torch based alternatives



What we Did

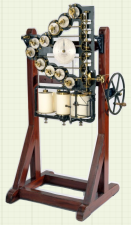
- Asked Stan to use hidden state methods to solve an ARIMAX *style* formulation of the problem.
- We directly controlled how different regressors last over time.
- We directly specified two sub-populations of customers (with different behavior) that we only observe the sum of.
  - We asked Stan to "un-stir" the mixture.

Why it all Works

- We want to infer parameters β and hidden state ζ given data. That is pick β, ζ such that P[$\beta, \zeta \mid data$] is maximal.
- By Bayes' Law P[$\beta, \zeta \mid data$] = P[β, ζ]P[$data \mid \beta, \zeta$]/P[$data$].
  - Can ignore P[$data$] as it is free of our parameter estimates.
  - Therefore: picking β, ζ such that P[β, ζ]P[$data \mid \beta, \zeta$] is large is the same as picking such that P[$\beta, \zeta \mid data$] is large.
- Structural mis-specification *much* more risky than "wrong priors" when we have a lot of training data.

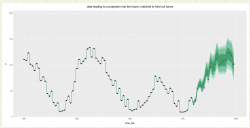
(skip slide)

"No Math"



- *We* don't do the math, Stan and the Stan authors do.
- We specified the two equations of how we think the customer counts behave over time.
  - Ignored that we only have access to the sum of the counts, and not to the sub-populations.
- We then sent these specifying equations and past data to Stan and let our computer "turn the crank."

What are the Bands?



- The bands are not frequentist confidence intervals
  - They do not represent the sensitivity of a deterministic prediction with respect to plausible re-draws of the training data. I.e. they are not estimating sampling noise or sampling sensitivity.
- They are the distribution of alternate sampled predictions from a single model specification
  - Closest to Bayesian credible intervals or conformal intervals (though may require some calibration to have desired properties).

Stan is very expressive, and one can use it to address issues that are normally insurmountable.

The Stan Solution in Detail

https://github.com/WinVector/Examples/blob/main/TimeSeries/Stan_soln.ipynb

What we get from Stan

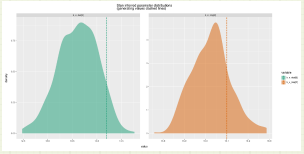
	lp__	accept_stat__	stepsize__	treedepth__	n_leapfrog__	divergent__	energy__	b_auto_0	b_imp_0	b_auto[0]	...	y[990]	y[991]	y[992]	y
0	2722.44	0.979102	0.000706	10.0	1023.0	0.0	-2231.00	1.70910	6.71948	1.94931	...	100.3790	121.7440	109.6260	112
1	2748.47	0.871088	0.000706	10.0	1023.0	0.0	-2225.93	1.74729	5.52749	1.93935	...	108.3290	125.1170	106.9760	104
2	2732.40	0.823397	0.000706	10.0	1023.0	0.0	-2209.06	1.64855	5.59184	1.93849	...	108.5620	127.6830	113.5490	114
3	2726.63	0.786470	0.000706	10.0	1023.0	0.0	-2252.23	1.75269	5.57718	1.93583	...	119.6260	135.2040	117.8630	119
4	2695.50	0.966396	0.000706	10.0	1023.0	0.0	-2225.77	1.65860	5.87227	1.94142	...	119.7290	140.2880	125.0790	125
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	2746.19	0.937626	0.020688	8.0	255.0	0.0	-2250.90	1.65820	5.25812	1.93254	...	78.4539	96.6416	82.7909	84
396	2725.57	0.522581	0.020688	8.0	255.0	0.0	-2214.98	1.68432	5.23821	1.93332	...	76.1461	94.2906	83.5327	85
397	2721.35	0.467874	0.020688	8.0	255.0	0.0	-2228.27	1.75156	5.16328	1.94212	...	73.0259	95.1352	81.2215	86
398	2693.27	0.765531	0.020688	8.0	255.0	0.0	-2190.17	1.60223	5.54163	1.93545	...	85.9863	107.9670	95.5999	101
399	2703.47	0.971172	0.020688	8.0	255.0	0.0	-2204.56	1.78176	5.75108	1.93982	...	108.5280	131.6890	120.1180	12

400 rows × 2035 columns

Each row is a sample. The columns we are interested in for a given row are the parameters (“b_*”) and hidden state (“y[*”, “y_auto[*”, “y_future[*”). One thing to keep in mind: the estimated state and parameters in a given row are mutually consistent as they were picked with the plausibility (exp(lp__) / Z , for some Z) high. It is a domain question if and when we can average these across rows.

This gives us

- The green-ribbon future predictions we showed earlier.
- Distributional estimates of the model parameters.
- Distributional estimates of the breakdown between loyal customers (“auto regressive”) and total customers (total or auto + transient).



```
{'b_auto_0': 1.2804125781056719,
 'b_imp_0': 10.4,
 'b_auto': (1.9753766811902755, -1),
 'b_z': (14.2,),
 'b_x': (16.1,),
 'generating_lags': (1, 2),
 'modeling_lags': (1, 2)}
```

Note: the sampling distribution being to one side of the actual generative value is not in itself evidence of inference bias. These distributions are the estimate family given by a single draw of the data. The method should only be considered biased in the usual frequentist sense if the average of these inferences over multiple re-runs over different data draws is to one side.

Prototyping with Stan

- Stan isolates model solution from model specification
  - Can make Stan slower than optimized methods
  - Makes Stan much more flexible
  - Can turn features on and off and see if model performance degrades
    - All of this in terms of domain or business structure!
    - Not going on and on about characteristic functions, unit roots and such

Our Thesis

In business forecasting: ability to specify actionable structure beats technique in forecasting.

By actionable structure I mean properly structured external regressors that are under our control.

A Few Things From Experience

- At first it feels like forecasting software binding fitting and prediction together is a mistake.
  - Violates the *very* useful `sklearn` separate `.fit()` and `.predict()` API by forcing a `.fit_predict()` pattern.
  - However, part of forecast inference is to also estimate un-observed details of past state. So fitting is in fact not separate from forecasting.
  - Our own `vtreat` high cardinality variable re-encoding package uses the same restriction to prevent over fitting.
- Stan great for prototyping
  - Not forced to use full Stan methodology in production.
- Can, in many cases, export inferences from Stan for use by simpler implementations.
  - https://win-vector.com/2019/07/03/replicating-a-linear-model/

Follow up resources

- This talk
  - https://github.com/WinVector/Examples/tree/main/TimeSeries#readme
- Some of the Perils of Time Series Forecasting
  - https://win-vector.com/2023/05/25/some-of-the-perils-of-time-series-forecasting/
- A Time Series Apologia
  - https://win-vector.com/2023/05/07/a-time-series-apologia/
- Please Version Data
  - https://win-vector.com/2024/09/09/please-version-data/
- The `vtreat` data preparation system
  - https://github.com/WinVector/pyvtreat
  - https://github.com/WinVector/vtreat

Thank you

All materials: https://github.com/WinVector/Examples/tree/main/TimeSeries#readme

Part of Our Using Stan to Solve Problems Training Offering

- The series currently includes:
  - Dealing with range censored data, or tobit style regression.
  - Learning rank preferences from observed actions.
  - Time series with external explanatory variables.
- Please contact jmount@win-vector.com for custom data science research, training and consulting.

All materials: https://github.com/WinVector/Examples/tree/main/TimeSeries#readme

(stop on this slide)

Appendices

All materials: https://github.com/WinVector/Examples/tree/main/TimeSeries#readme

ARIMAX Solutions

- https://github.com/WinVector/Examples/blob/main/TimeSeries/ts_example.md
- https://github.com/WinVector/Examples/blob/main/TimeSeries/sm_example.ipynb

Just going to show the R versions. Or skip.

Ad-Hoc Nested Regression Solution

- https://github.com/WinVector/Examples/blob/main/TimeSeries/nested_model_example.ipynb
- https://win-vector.com/2023/05/07/a-time-series-apologia/

(possibly skip)