Forecasting I

DS-6030 | Spring 2026

forecasting-1.pdf

Table of contents

1 Time Series and Forecasting

1.1 Framing: What Makes Forecasting Different?

All semester we've been building models to predict outcomes. Forecasting is prediction too, but with two key differences:

1. **Extrapolation, not interpolation.** In regression, our test data usually falls within the range of training data. In forecasting, we're predicting *future* values that are beyond anything we've observed. Extrapolating trends is exactly the kind of thing that can go very wrong.
2. **Order matters.** Our observations aren't exchangeable - they have a temporal structure. This constrains how we can split data for training and evaluation. (No more random k-fold CV.)

We can write a forecast as:

$$\hat{y}(t+h) = \hat{f}_h(y_{1:t}, X)$$

where $h > 0$ is the **forecast horizon** - how far ahead we're trying to predict. This is a new dial we haven't had to set before, and it shapes everything: model choice, evaluation, and how wide our prediction intervals get.

- $h > 0$ is the forecast horizon
- $\hat{f}_h$ is the forecasting model (for horizon h)
- $y_{1:t} = y_1, \ldots, y_t$ are the past outcomes
- $X = X_1, \ldots, X_p$ are (optionally) *other* predictive features (available at time t)

1.2 Time Series Data

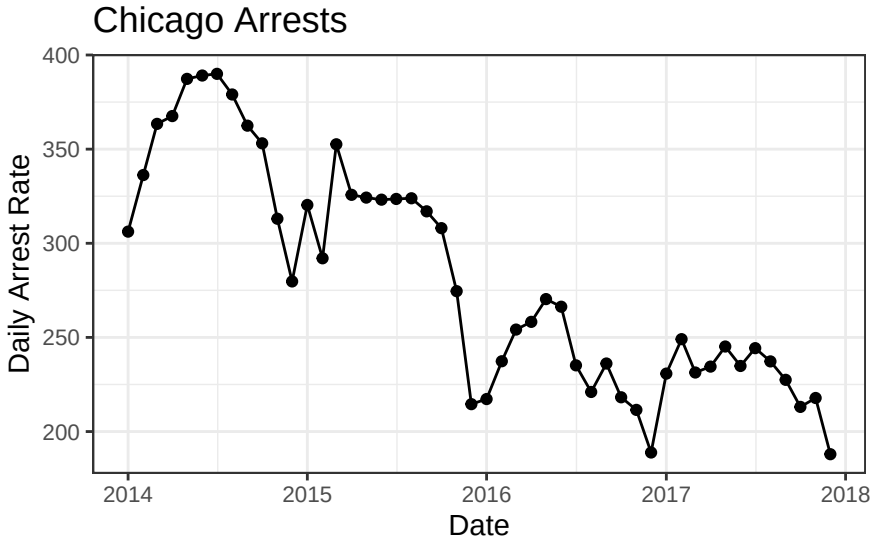
A time series is data that is recorded at sequentially at *regular intervals*[1] of time. We can write the data as $D = \{(t_i, y_i)\}$ where:

- $t_{i-1} < t_i < t_{i+1}$ is the time of the ith observation
- $\delta = t_i - t_{i-1}$ is the fixed interval between times
  - units: years, quarters, months, weeks, days, hours, mins, sec
- The outcome variable y_i can be anything (real valued, integer/count, categorical, graph, image)
- For regular (i.e., fixed interval) time series, it's common to simplify notation by moving time into subscript of y denote the time $\ldots, y_{t-1}, y_t, y_{t+1}, \ldots$

[1] Technically, you can have irregularly spaced time series (e.g., stock market closed weekends and holidays) but need to treat them as sequential observations.

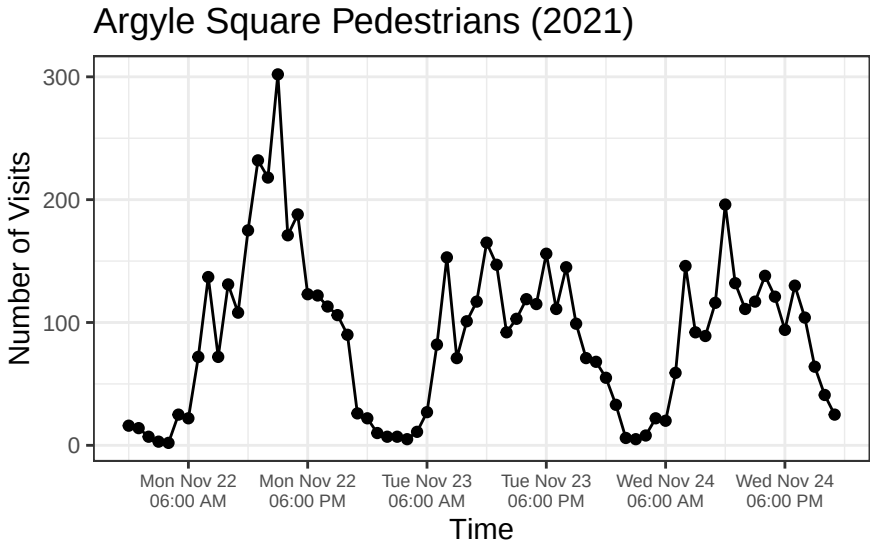
1.2.1 Chicago Arrest Data

The Chicago Police Department published arrest data from 2014–2017. We'll use monthly arrest counts as our running example.



1.3 Traffic Counts

The City of Melbourne Australia has published pedestrian traffic count data in Argyle Square for a few days in 2021.



Your Turn #1 : Patterns

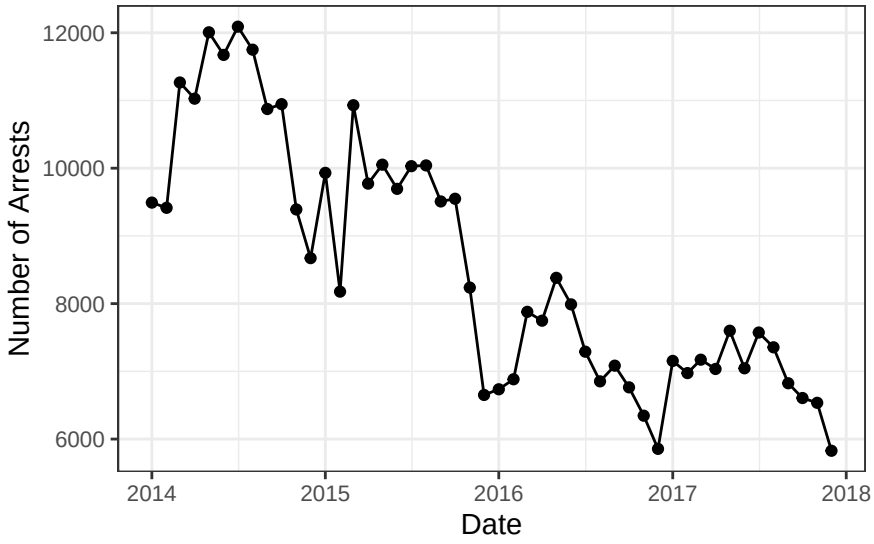
What patterns do you see in the two time series?
Try to identify:

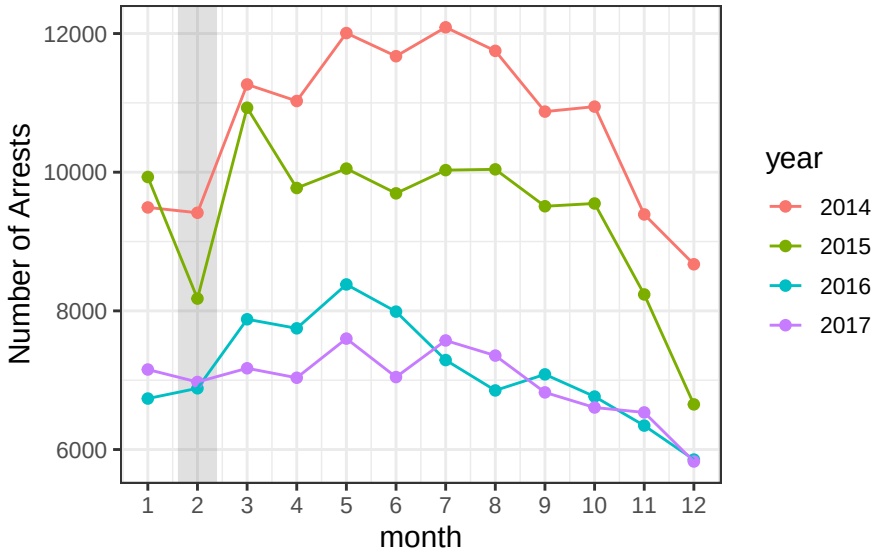
- Is there a trend?
- Is there a repeating seasonal pattern?
- Are there any sudden changes?

1.3.1 Exposure

In modeling count/event data, it is especially important to consider how time influences the opportunity for events.

Consider the Chicago arrest data. Notice that the y-axis is average daily arrests. Why don't we show the arrest count per month?





Your Turn #2

Why are the number of arrests lower in February compared to January and March?

There are two main approaches to dealing with varying exposure:

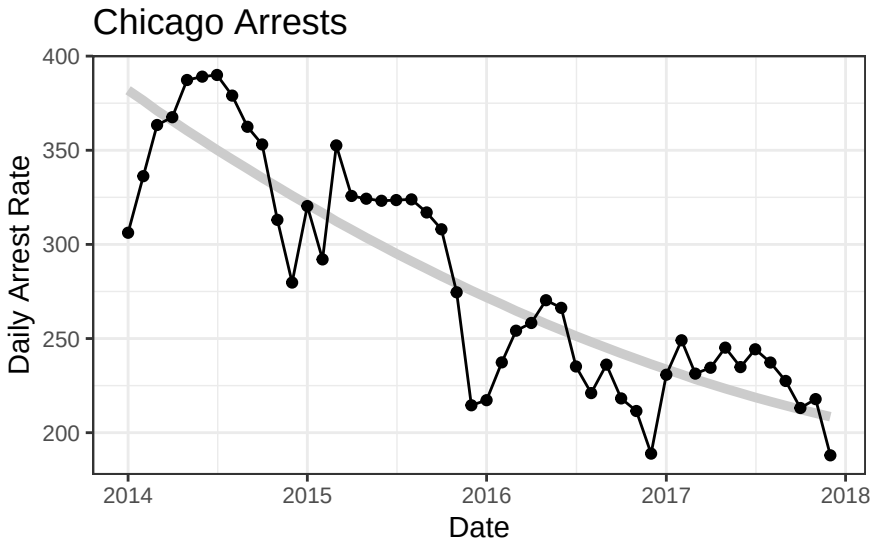
1. Standardize the data (e.g., average arrests per day)
2. Adjust for the exposure when forecasting (e.g., with offsets)

1.4 Patterns in Time Series Data

There are three patterns worth looking for because they represent **exploitable structure** that a model can capture and use to make better forecasts.

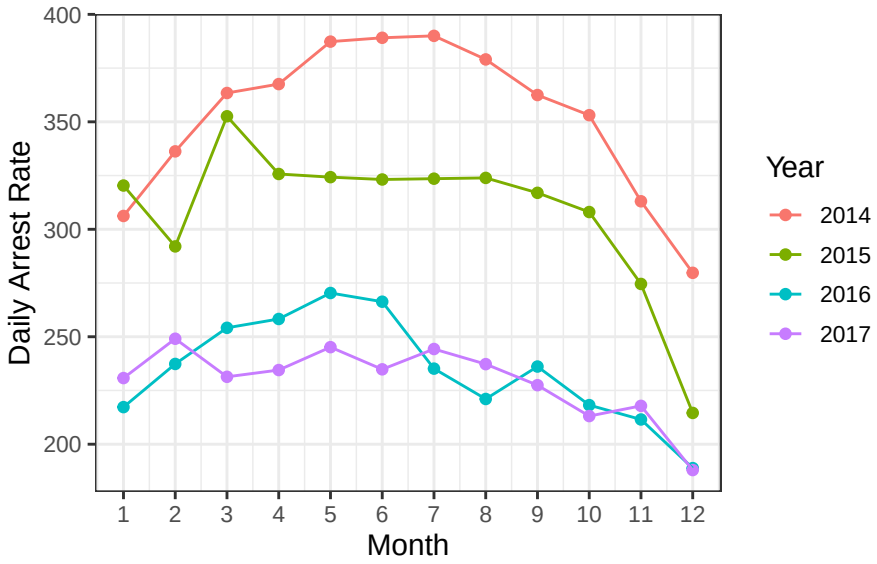
1. Trends

Observations change systematically over time. We see that arrests are declining over the four-year period.



2. Seasonality

Patterns that repeat at regular intervals. Monthly arrest rates peak in warmer months and dip in winter. This pattern is driven by human behavior.

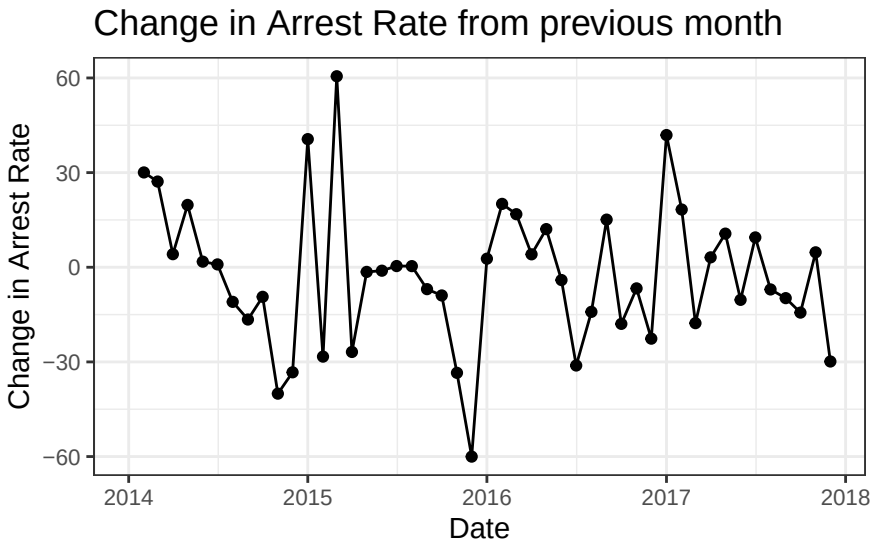


3. Shocks and Changepoints

Shocks (temporary) and changepoints (persistent) are significance changes to the data generating system

- The data generating system may experience shocks or structural changes (changepoints) that temporarily or persistently change the process
- E.g., large weather events, change in political leaders
- These can sometimes be captured in *trends* and *special times*, but there is a field of time series analysis called *intervention analysis* that explicitly models such changes in the presence of temporally correlated errors.

There appears to be a notable drop in arrests in late 2015[2].



Other Patterns

Other structures exist:

- exogenous predictors (e.g., number of officers on duty),
- cycles with non-fixed periods (e.g., economic cycles)
- special times (e.g., holidays, events)

But trend, seasonality, and changepoints are the big three.

[2] In November 2015, dashcam footage of Laquan McDonald being shot by a Chicago police officer was released. The video sparked major public protests and intense scrutiny of the Chicago Police Department. The U.S. Department of Justice opened a civil rights investigation into CPD practices.

2 Decomposition

2.1 Additive Decomposition

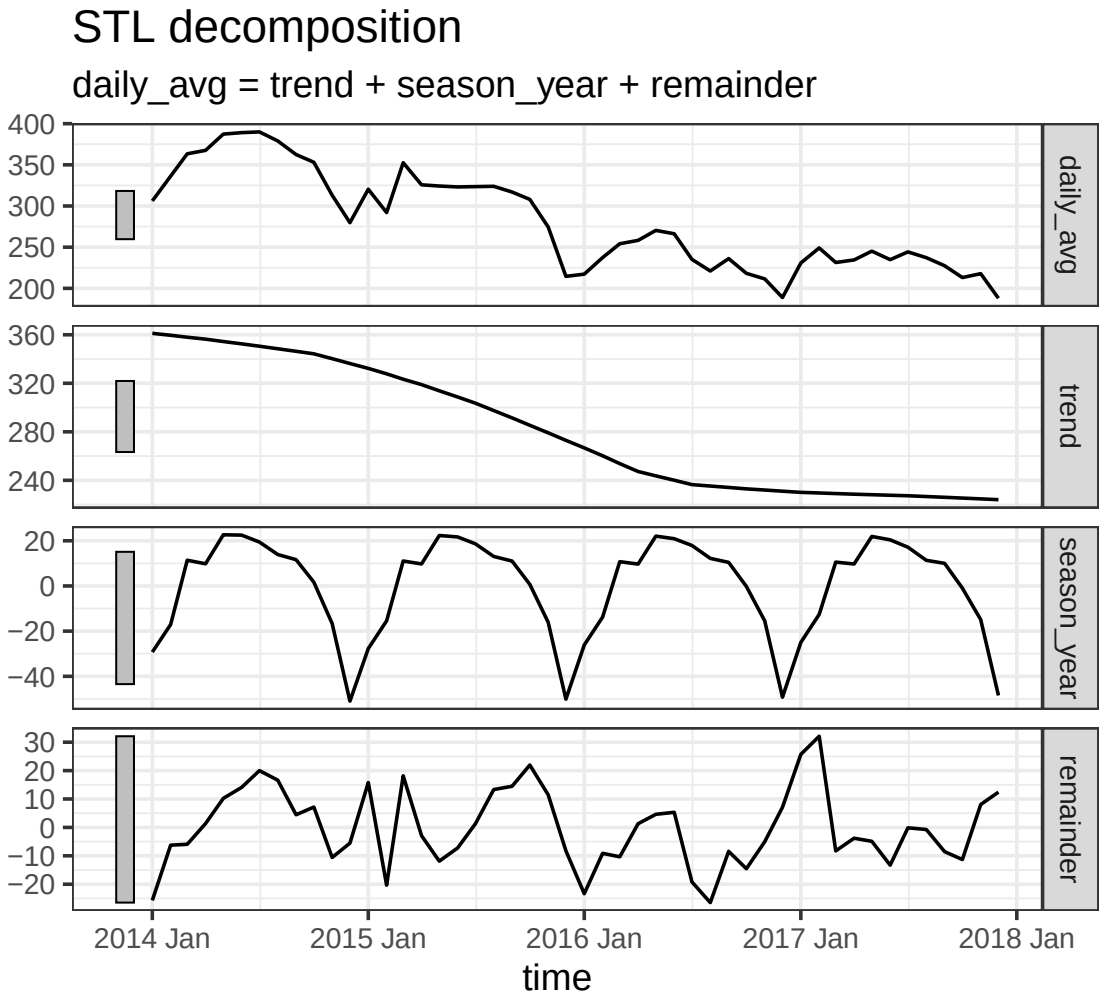
Consider the **additive decomposition** model:

$$y_t = T_t + S_t + R_t$$

- T_t is the **trend** (long-term direction)
- S_t is the **seasonal** component (repeating pattern)
- R_t is the **remainder** - what's left after removing trend and seasonality

The goal of decomposition is to **identify what's predictable vs. what isn't.** Trend and seasonality are exploitable structure. The remainder is what's hardest to forecast as ideally it looks like noise.

We'll use STL (*Seasonal and Trend decomposition using Loess*[3]) to estimate these components.



[3] Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition. J. Off. Stat, 6(1), 3-73.

2.2 Multiplicative Decomposition

It's also common to consider a *multiplicative* structure:

$$y_t = T_t \cdot S_t \cdot R_t$$

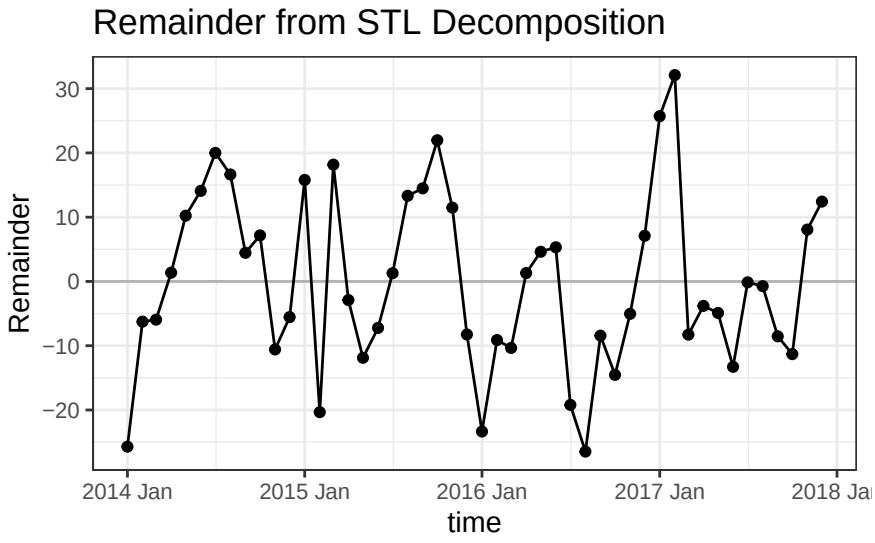
This is useful when the seasonal swings grow proportionally with the level (e.g., retail sales). Taking logs converts multiplicative to additive:

$$\log y_t = \log T_t + \log S_t + \log R_t$$
$$y'_t = T'_t + S'_t + R'_t$$

For the arrest data, the additive form looks reasonable - the seasonal amplitude doesn't change much across years.

2.3 Residual Check

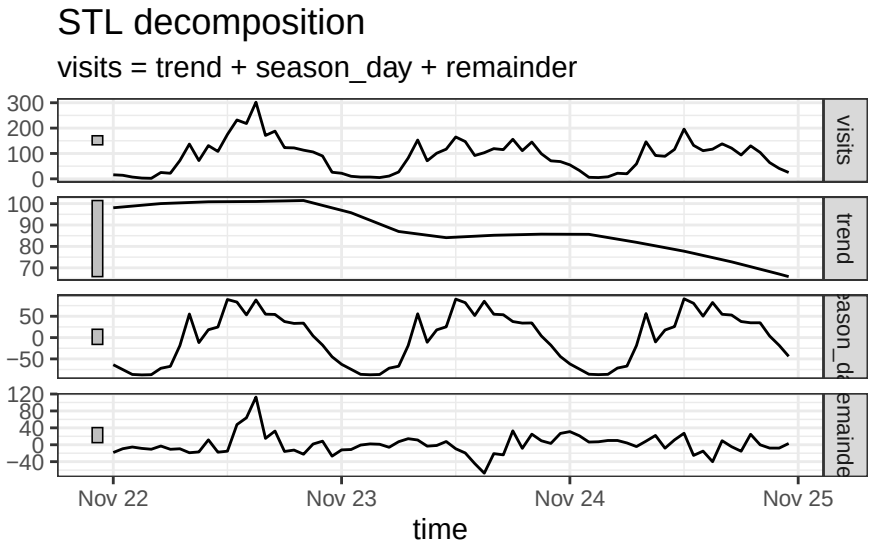
After decomposition, always look at the remainder. **Does it look like noise, or is there structure we missed?**



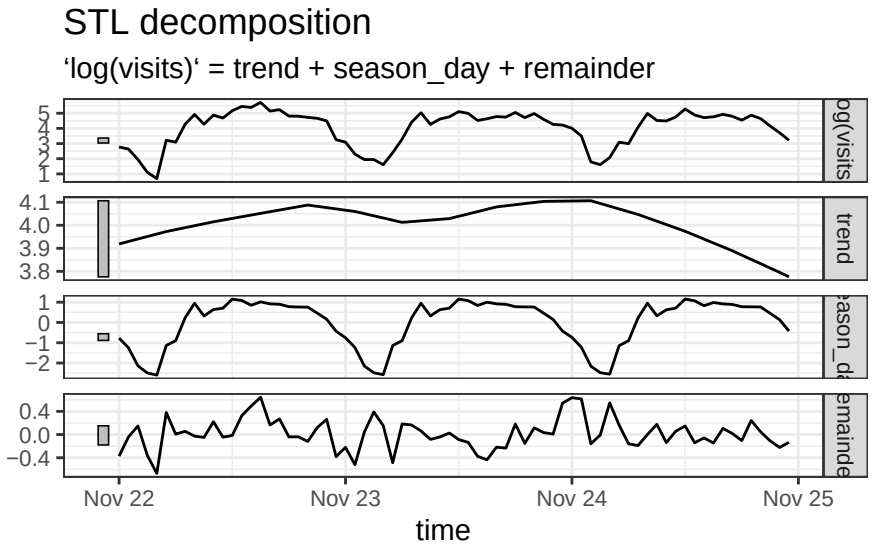
This is the same diagnostic thinking from our calibration and residual analysis modules. If you see patterns in what's left over, the model is missing something.

2.4 Decomposing Traffic Counts

Additive



Multiplicative using the log transformation



3 Simple Baseline Models

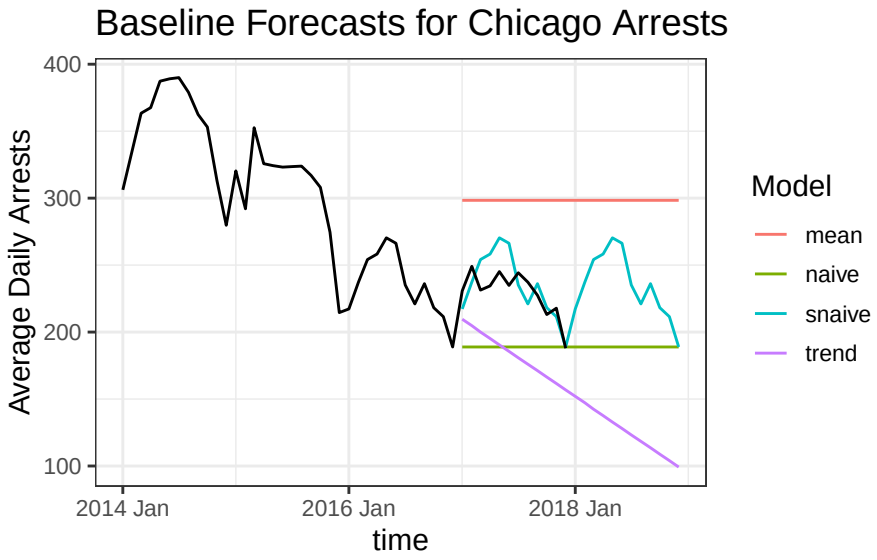
Before building anything complex, we need to establish the baseline performance. Three simple baselines give us benchmarks:

- **Naive:** forecast = last observed value.
- **Seasonal Naive:** forecast = value from the same season last year.
- **Mean:** forecast = historical average.
- **Linear Trend:** forecast = linear regression.

If your model can't regularly beat one of these, it's not very useful.

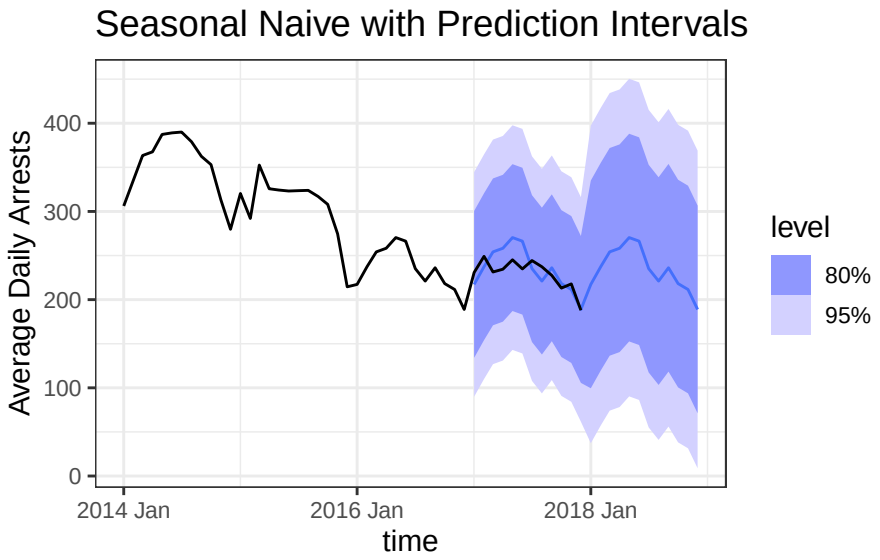
Let's split the arrest data into training and test sets. We'll train on 2014–2016 and hold out 2017 for evaluation.

Forecast into the test period:



3.1 Prediction Intervals on Baselines

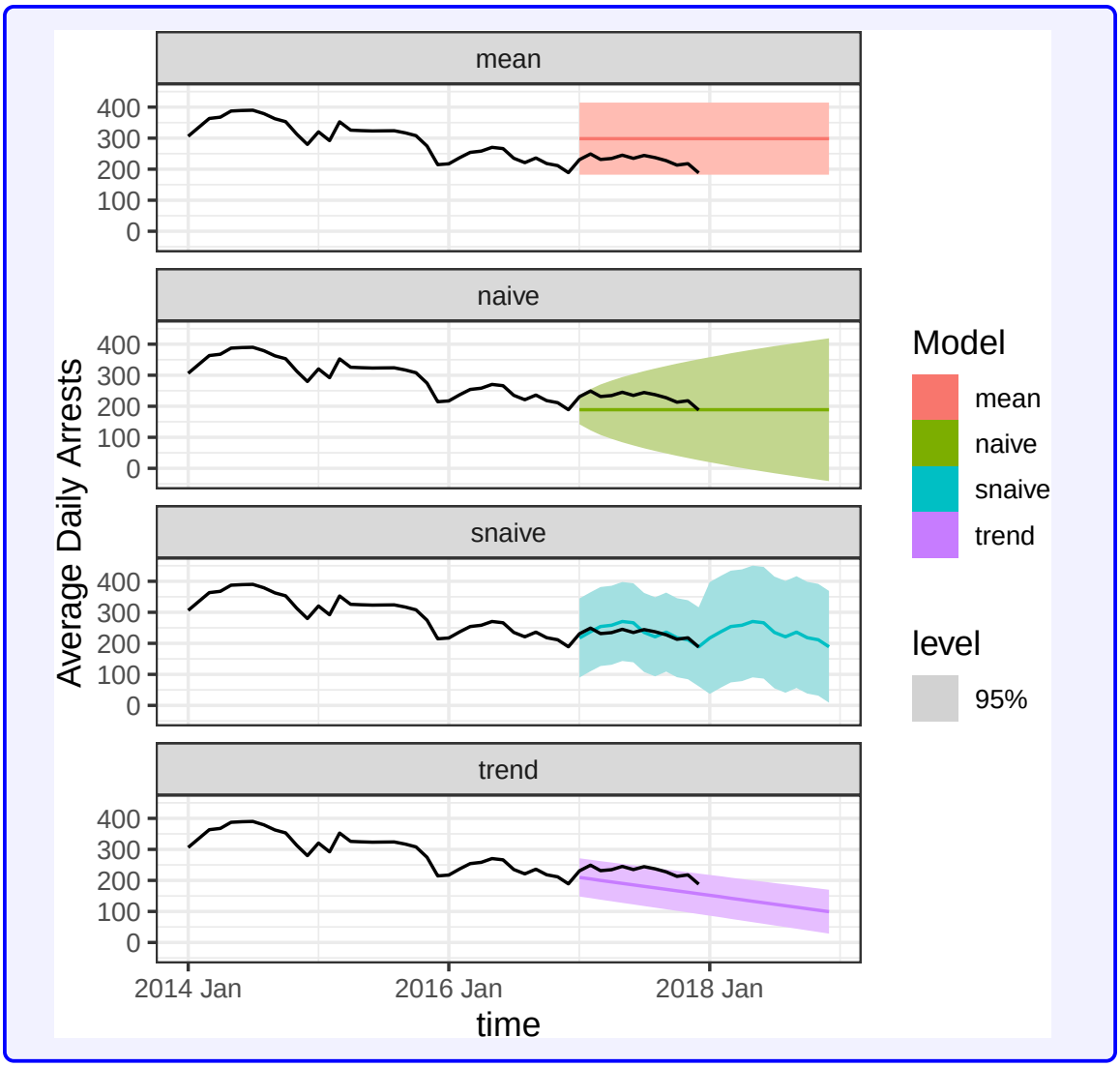
Even a simple forecast isn't a single number. Here are the seasonal naive forecasts with 80% and 95% prediction intervals:



Notice what happens as the forecast horizon h increases: **the intervals get wider.** At $h = 1$ (one month ahead), we're fairly confident. At $h = 12$ (a full year out), the uncertainty is much larger. This is a fundamental feature of forecasting - the further ahead you predict, the less certain you are.

Your Turn #3

Compare the prediction intervals across the three baselines. Which model produces the widest intervals? Why does that make sense?


mean
naive
snaive
trend
400
300
200
100
0
Average Daily Arrests
2014 Jan
2016 Jan
2018 Jan
time
Model
mean
naive
snaive
trend
level
95%

4 Feature Engineering for Time Series

Here's the bridge to what you already know: once you create time-based features, you can use *any* model in your toolkit: linear models, random forests, boosted trees, etc.

Three types of features:

4.1 Lag Features

Use past values as predictors: y_{t-1}, y_{t-2}, y_{t-12} (same month last year), etc.

date	daily_avg	lag_1	lag_2	lag_12
2014-01-01	306.2	NA	NA	NA
2014-02-01	336.2	306.2	NA	NA
2014-03-01	363.4	336.2	306.2	NA
2014-04-01	367.6	363.4	336.2	NA
2014-05-01	387.3	367.6	363.4	NA
2014-06-01	389.1	387.3	367.6	NA
2014-07-01	390.0	389.1	387.3	NA
2014-08-01	379.0	390.0	389.1	NA
2014-09-01	362.5	379.0	390.0	NA
2014-10-01	353.1	362.5	379.0	NA
2014-11-01	313.0	353.1	362.5	NA
2014-12-01	279.7	313.0	353.1	NA
2015-01-01	320.4	279.7	313.0	306.2
2015-02-01	292.0	320.4	279.7	336.2
2015-03-01	352.6	292.0	320.4	363.4

4.2 Calendar Features

Use time indicators as predictors: month of year, day of week, year, holiday flags.

date	daily_avg	month	index
2014-01-01	306.2	Jan	1
2014-02-01	336.2	Feb	2
2014-03-01	363.4	Mar	3
2014-04-01	367.6	Apr	4
2014-05-01	387.3	May	5

- For monthly data with yearly seasonality, month indicators capture the seasonal pattern directly.
- For daily data, you might also add day-of-week, holiday flags, etc.

4.3 Fourier Terms

A smoother alternative to calendar dummies for capturing seasonality (i.e., instead of a separate parameter for each month), *Fourier terms* use pairs of sin and cos functions. These are especially useful when the seasonal period is long (e.g., 365 days for daily data). Details are in the Appendix for those who want to go deeper.

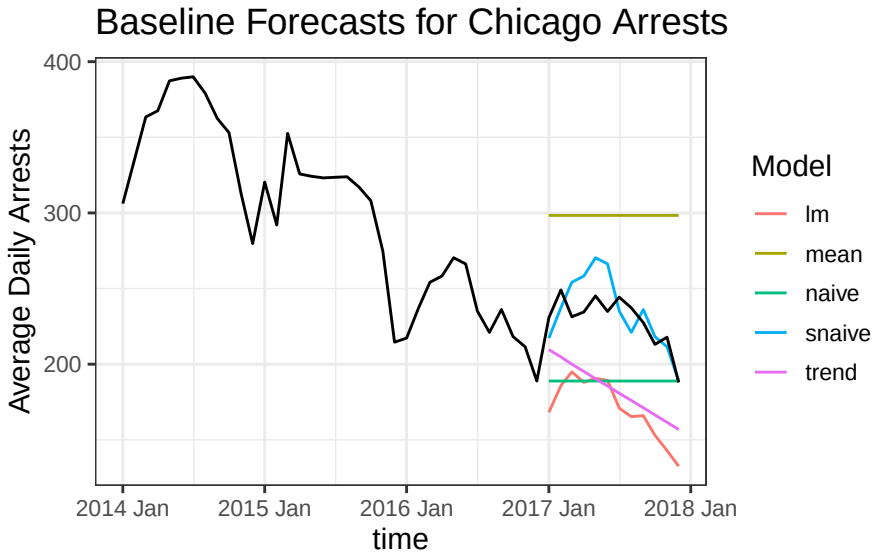
4.4 Regression with Time Series Features

Once we have features, this is just a regression problem:

```
#>
#> Call:
#> lm(formula = daily_avg ~ index + month + lag_1 + lag_12, data = arrest_model_data)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -31.52  -8.14   0.43   8.50  34.10
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  262.479    124.744    2.10   0.0476 *
#> index         -2.566      1.500   -1.71   0.1019
#> monthFeb      -9.656     16.910   -0.57   0.5740
#> monthMar      21.174     21.979    0.96   0.3463
#> monthApr       2.910     24.136    0.12   0.9052
#> monthMay      19.698     26.184    0.75   0.4602
#> monthJun      12.334     27.993    0.44   0.6640
#> monthJul       8.033     26.995    0.30   0.7690
#> monthAug       5.623     26.081    0.22   0.8314
#> monthSep      11.142     26.075    0.43   0.6735
#> monthOct      -3.334     25.299   -0.13   0.8964
#> monthNov     -12.021     21.210   -0.57   0.5769
#> monthDec     -51.198     17.637   -2.90   0.0085 **
#> lag_1          0.625      0.174    3.60   0.0017 **
#> lag_12        -0.301      0.228   -1.32   0.2018
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 18.8 on 21 degrees of freedom
#> Multiple R-squared:  0.894,  Adjusted R-squared:  0.824
#> F-statistic: 12.7 on 14 and 21 DF,  p-value: 3.13e-07
```

The key insight: forecasting with regression is just prediction with carefully constructed features.

- Once you create time-based features (lags, seasonality, trends), you can use the same models you already know: linear regression, elastic net, boosted trees, etc.
- This makes feature engineering the most transferable skill: the modeling step is familiar, the challenge is representing time appropriately.
- Many "specialized" forecasting methods (ARIMA, ETS, Prophet) are doing something very similar under the hood: capturing patterns like trend and seasonality through structured features and fitting relatively simple models (often linear with constraints or regularization).



Your Turn #4

1. Suppose you have *daily* data on website traffic with yearly seasonality and a day-of-week effect. What features would you create?

2. What complications arise when you try to forecast h steps ahead using lag features? (Hint: at $h = 2$, do you have y_{t+1} available to use as a lag?)

5 Summary

Today we covered the first half of the forecasting workflow:

> **Look at data → identify structure → build baselines** → choose model → evaluate with rolling CV → communicate uncertainty

Key takeaways:

- Forecasting is prediction under extrapolation: order matters, horizon matters.
- Decomposition separates the predictable (trend, seasonality) from the remainder.
- Always compare against simple baselines: seasonal naive is usually the benchmark to beat.
- Prediction intervals widen as horizon increases; even simple models have uncertainty.
- Feature engineering (lags, calendar indicators) turns forecasting into a regression problem.

Next time: how to properly evaluate forecasting models with time series CV, a survey of modeling approaches, and deeper look at how uncertainty grows with horizon.

6 Appendix: Fourier Basis Functions for Seasonality

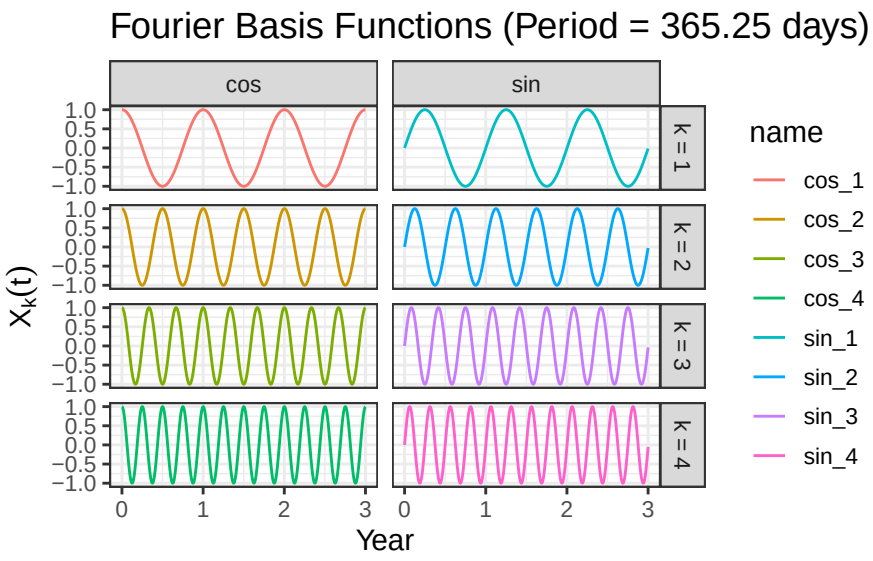
Calendar features (e.g., month indicators) work well, but they estimate a separate parameter for each level — 12 parameters for monthly seasonality, 365 for daily. **Fourier basis functions** offer a smoother, lower-dimensional alternative.

A Fourier pair at order k with period P is:

$$X_{kc}(t) = \cos\left(\frac{2\pi kt}{P}\right)$$
$$X_{ks}(t) = \sin\left(\frac{2\pi kt}{P}\right)$$

Using K orders gives $2K$ predictor variables. Low K captures broad seasonal shape; high K captures finer detail.

Here is what the Fourier basis functions look like over 3 years of daily data:



For the Chicago arrest data (monthly, period = 12), the first few Fourier features look like:

time	cos_1	cos_2	cos_3	sin_1	sin_2	sin_3
2014 Jan	0.866	0.5	0	0.500	0.866	1
2014 Feb	0.500	-0.5	-1	0.866	0.866	0
2014 Mar	0.000	-1.0	0	1.000	0.000	-1
2014 Apr	-0.500	-0.5	1	0.866	-0.866	0
2014 May	-0.866	0.5	0	0.500	-0.866	1
2014 Jun	-1.000	1.0	-1	0.000	0.000	0
2014 Jul	-0.866	0.5	0	-0.500	0.866	-1
2014 Aug	-0.500	-0.5	1	-0.866	0.866	0
2014 Sep	0.000	-1.0	0	-1.000	0.000	1
2014 Oct	0.500	-0.5	-1	-0.866	-0.866	0
2014 Nov	0.866	0.5	0	-0.500	-0.866	-1
2014 Dec	1.000	1.0	1	0.000	0.000	0