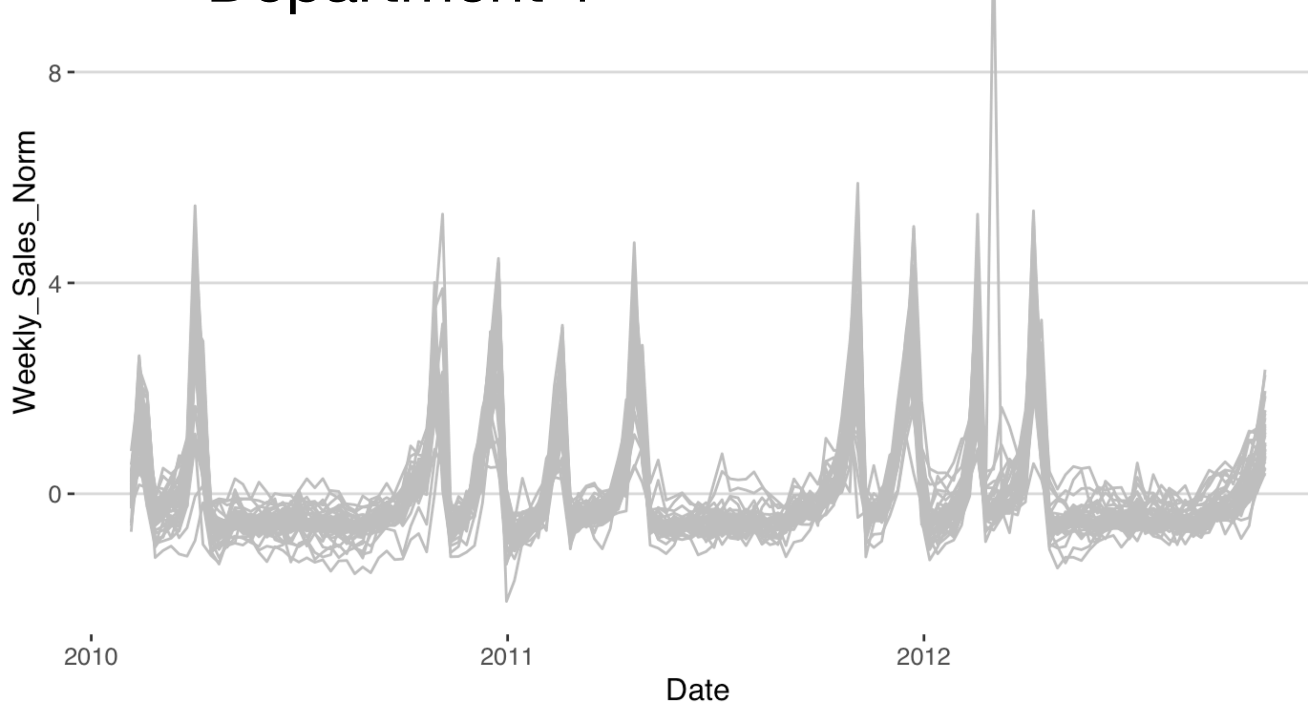
The Walmart Sales Project

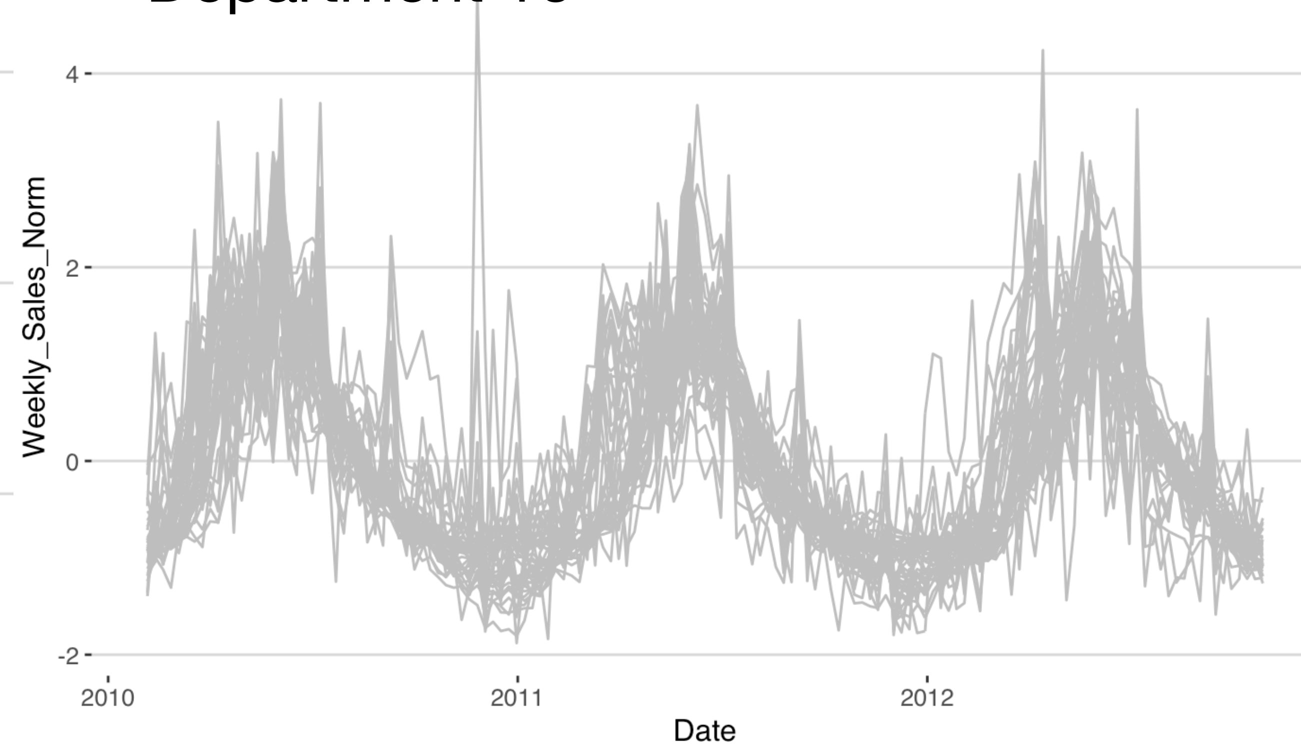
- Goal: Forecast the weekly sales for 99 departments at 45 Walmart stores located in different regions.

- That is 4,455 different time series! However, notice that time series within the same department have similar patterns:

Department 1



Department 16



What is a Time Series

Date	y
2012	123
2013	39
2014	78
2015	110

$$\mathbf{y} = (y_1, y_2, \ldots, y_T) = (123, 39, 78, 110)$$

Frequency of a Time Series

- Frequency: the number of observations before the seasonal pattern repeats. In physics and engineering this is the period.

Data	Frequency
Annual	1
Quarterly	4
Monthly	12
Weekly	52

- Caveat: There are not 52 weeks in a year, but 365.25/7 = 52.18 on average.

Working with Time Series

- Load in R as a ts object

```
> y <- ts(c(123,39,78,110), start=2012, frequency=1)

Time Series:
Start = 1
End = 4
Frequency = 1
[1] 123  39  78 110
```

Working with Multiple Time Series

Notation

- Assume we have *m* different times series of length *T*, e.g. weekly time series per store (within a department). Form a matrix **Y** containing each time-series as follows:

i

i^{th} time series: $\mathbf{y}^{(i)} = (y_1^{(i)}, y_2^{(i)}, \ldots, y_T^{(i)})^T$

$$\mathbf{Y} = \begin{pmatrix} y_1^{(1)} & y_1^{(2)} & \cdots & y_1^{(i)} & \cdots & y_1^{(m)} \\ y_2^{(1)} & y_2^{(2)} & \cdots & y_2^{(i)} & \cdots & y_2^{(m)} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ y_t^{(1)} & y_t^{(2)} & \cdots & y_t^{(i)} & \cdots & y_t^{(m)} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ y_T^{(1)} & y_T^{(2)} & \cdots & y_T^{(i)} & \cdots & y_T^{(m)} \end{pmatrix}$$

Value of each time series at time *t*.

Working with Multiple Time Series

- Recommendation: Loop over departments and construct the matrix **Y**. You can then model column by column or try to combine information across columns, i.e. stores.

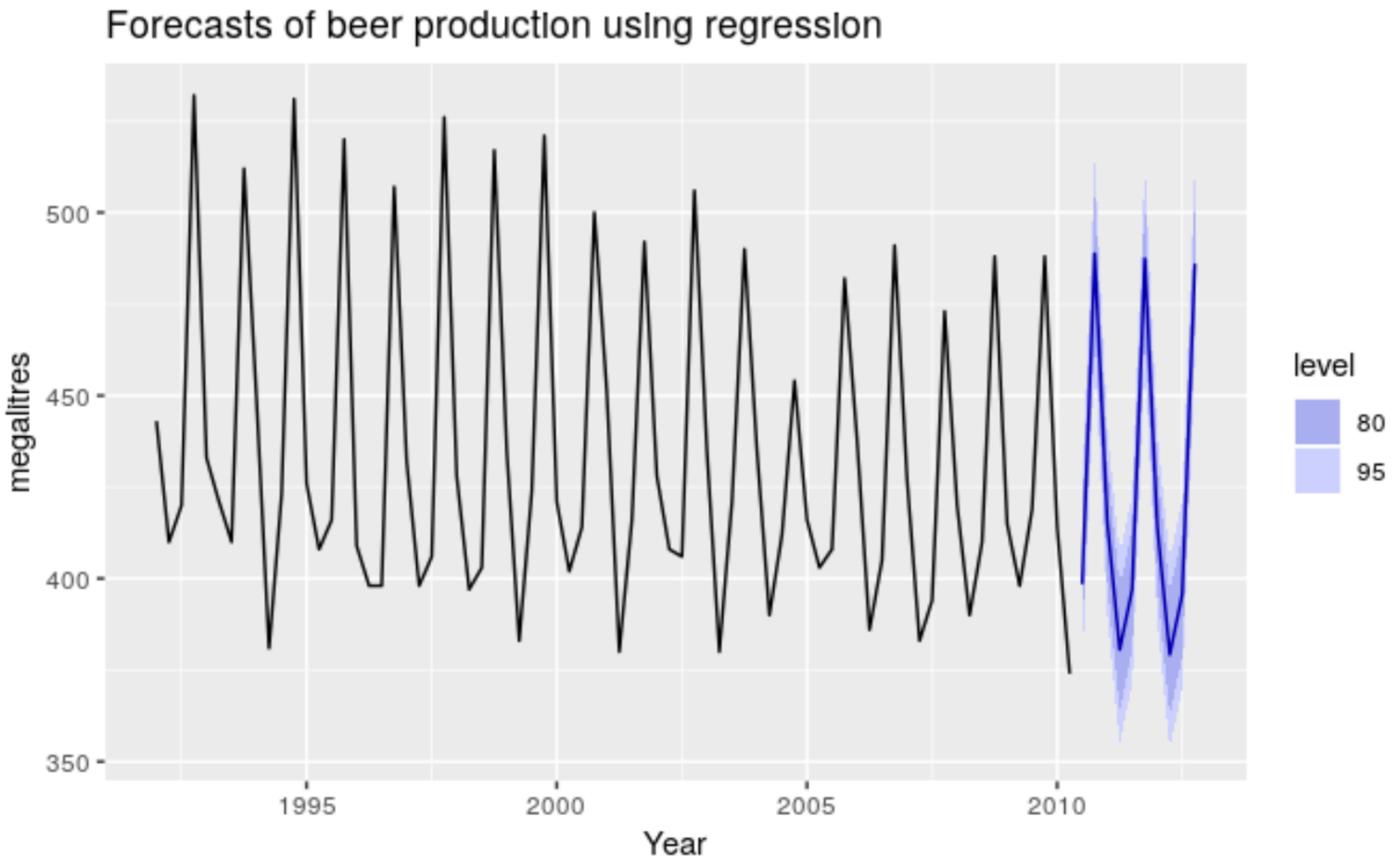
```
library(tidyverse)

> dept_tbl <- all_stores %>%
>     filter(Dept == num_dept) %>%
>     select('Date', 'Store', 'Weekly_Sales') %>%
>     spread(Store, Weekly_Sales)

> store_one_ts <- ts(dept_tbl[, 2], frequency = 52)
```

```
# A tibble: 143 x 46
   Date          `1`    `2`    `3`    `4`    `5`    `6`   `7`    `8`    `9`   `10`   `11`   `12`   `13`   `14`
   <date>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
 1 2010-02-05 10218.  9675.  3702.  7438.  4546. 11819. 2774.  2956.  3860. 25694.  5894.  4566.  3333. 10433.
 2 2010-02-12 11874.  9988.  4451.  8881.  6757. 11119. 4072.  4953.  3341. 24556.  6933.  3859.  4341  16731.
 3 2010-02-19 13856. 11496.  6473. 10693.  5245. 18669. 3734.  5912.  6438. 33322.  9789.  6936.  4882. 11945.
 4 2010-02-26 12881. 12558.  6322. 11943.  5150. 16651. 5188.  5946.  7241. 27774. 12764.  5895.  6392.  8287.
 5 2010-03-05 17130. 21957.  7769. 15409.  6922. 30437. 5025.  8506.  9652. 35228. 15792.  8287.  8843.  8047.
 6 2010-03-12 23767. 25827. 14296. 16169.  9407. 38650. 4289. 12776. 10164. 34864. 18317.  7352. 11771. 18217.
 7 2010-03-19 41742. 45311. 14103. 23152. 14971. 51103. 6680. 17947. 17654. 48984. 29762. 10249. 15575. 23750.
 8 2010-03-26 26680. 22084. 11458. 22741.  9350. 37527. 5238. 12889. 15085. 49448. 19302. 12934. 18668. 39182.
 9 2010-04-02 46061. 48042. 15256. 35582. 14933. 60800. 7798. 26715. 28892. 47481. 29953. 12760. 19276. 39508.
10 2010-04-09 52977. 58887. 17276. 38717. 15053. 56726. 6077. 34847. 23307. 53134. 30274. 13026. 18038. 61478.
# ... with 133 more rows, and 31 more variables: `15` <dbl>, `16` <dbl>, `17` <dbl>, `18` <dbl>, `19` <dbl>, …
```

Introduction to Forecasting

- Forecasting: the prediction of data at future times using observations collected in the past.

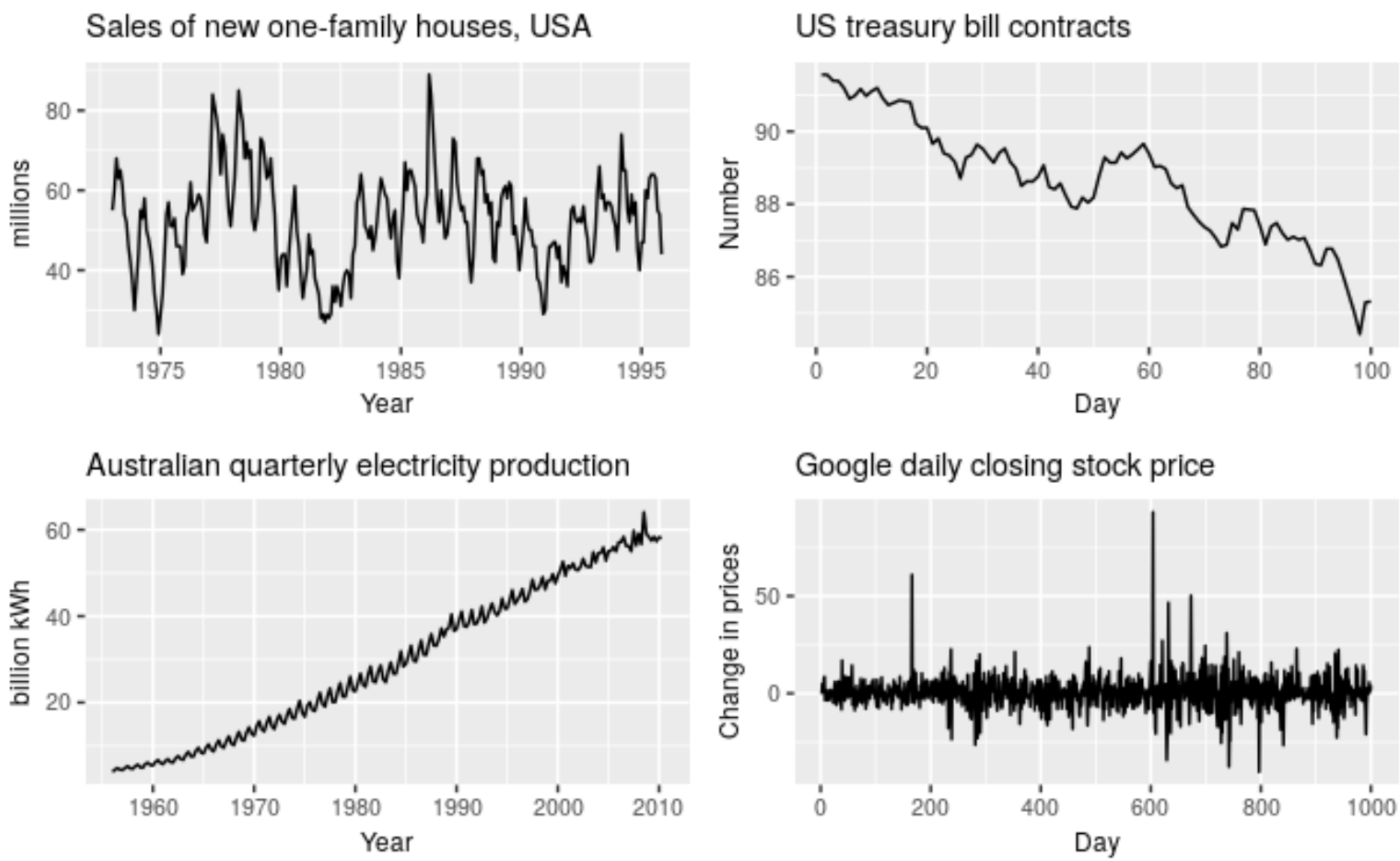
- Forecast horizon: How many time steps in the future a model will predict. I denote this value by h in the example code.



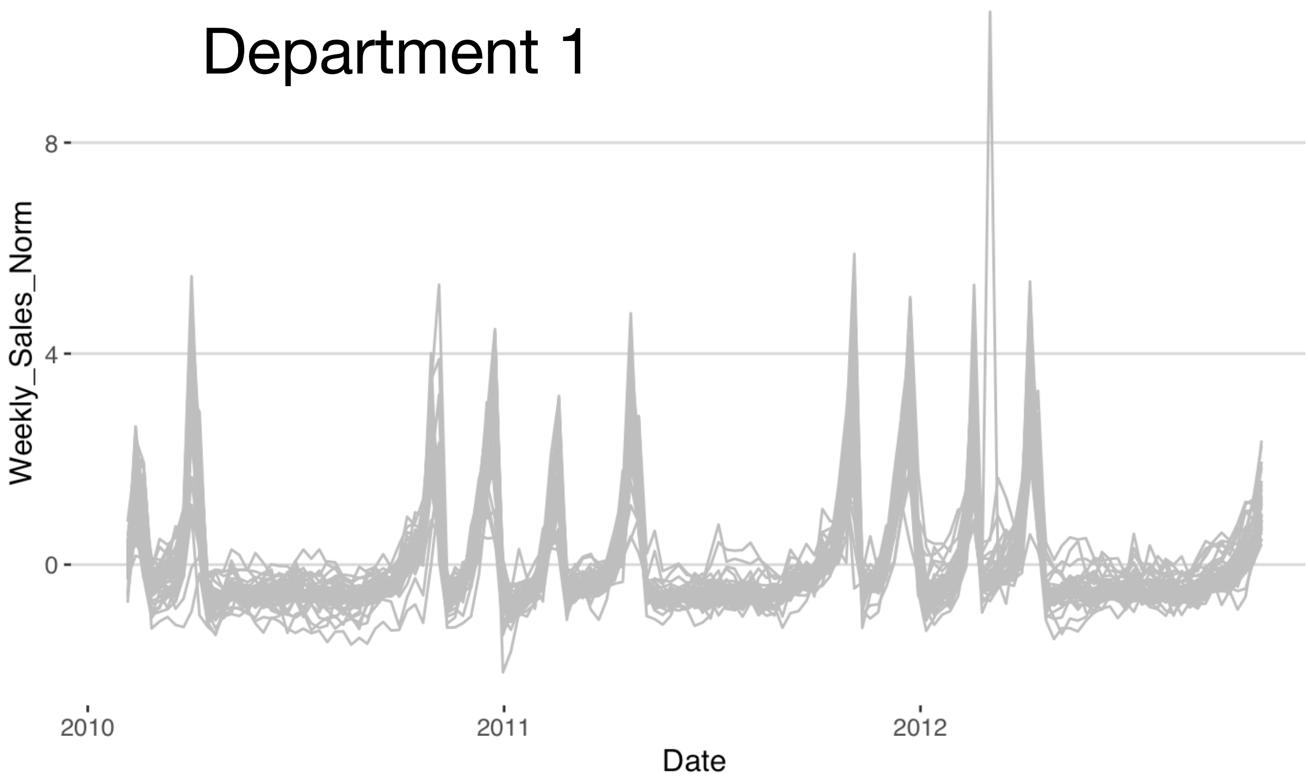
Time Series Patterns

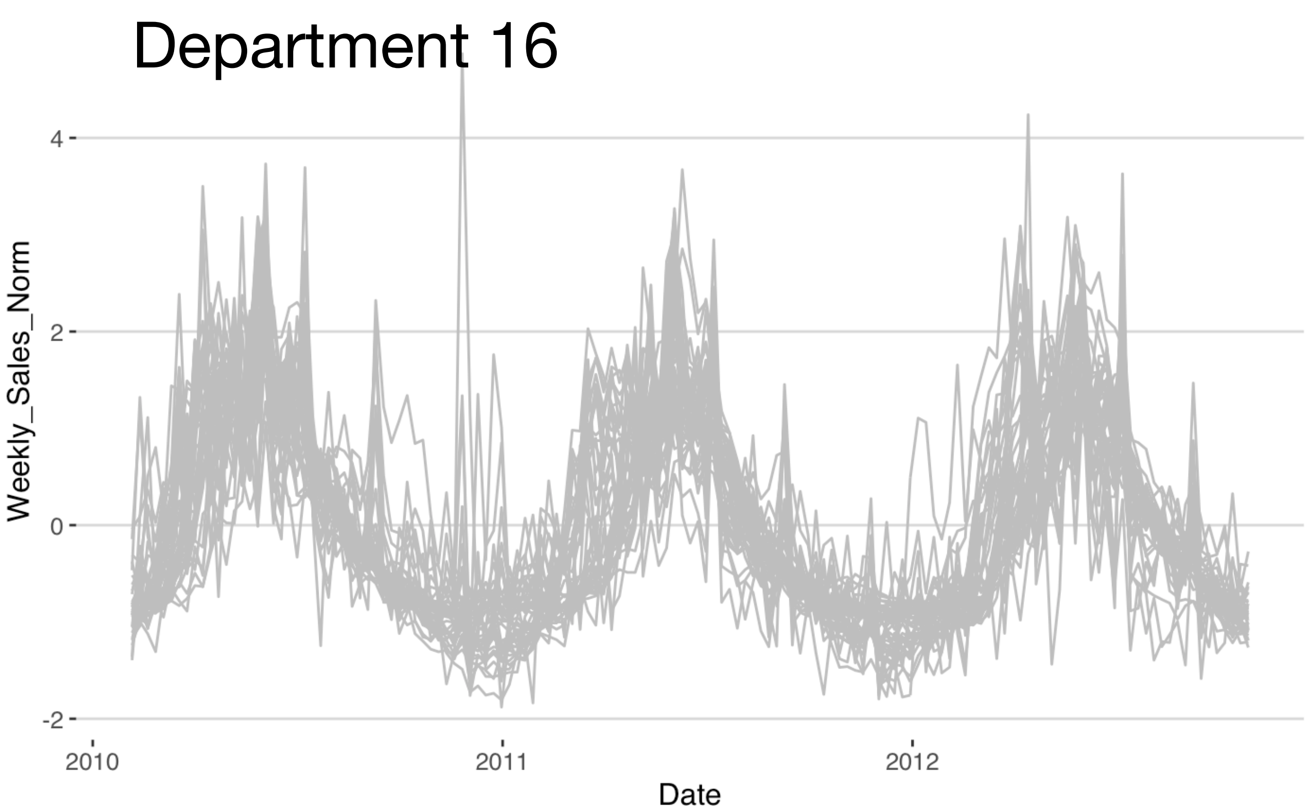
- Trend: A long-term increase or decrease in the data. Does not have to be linear!
- Seasonal: When a time series is affected by seasonal factors such as the time of the year or day of the week. Seasonality is always of a fixed and known frequency.
- Cyclic: When a time series exhibits rises and falls that are not of a fixed frequency.
- Random: Everything else after the Trend/Seasonal/Cyclic nature of the times series is removed.

Time Series Patterns



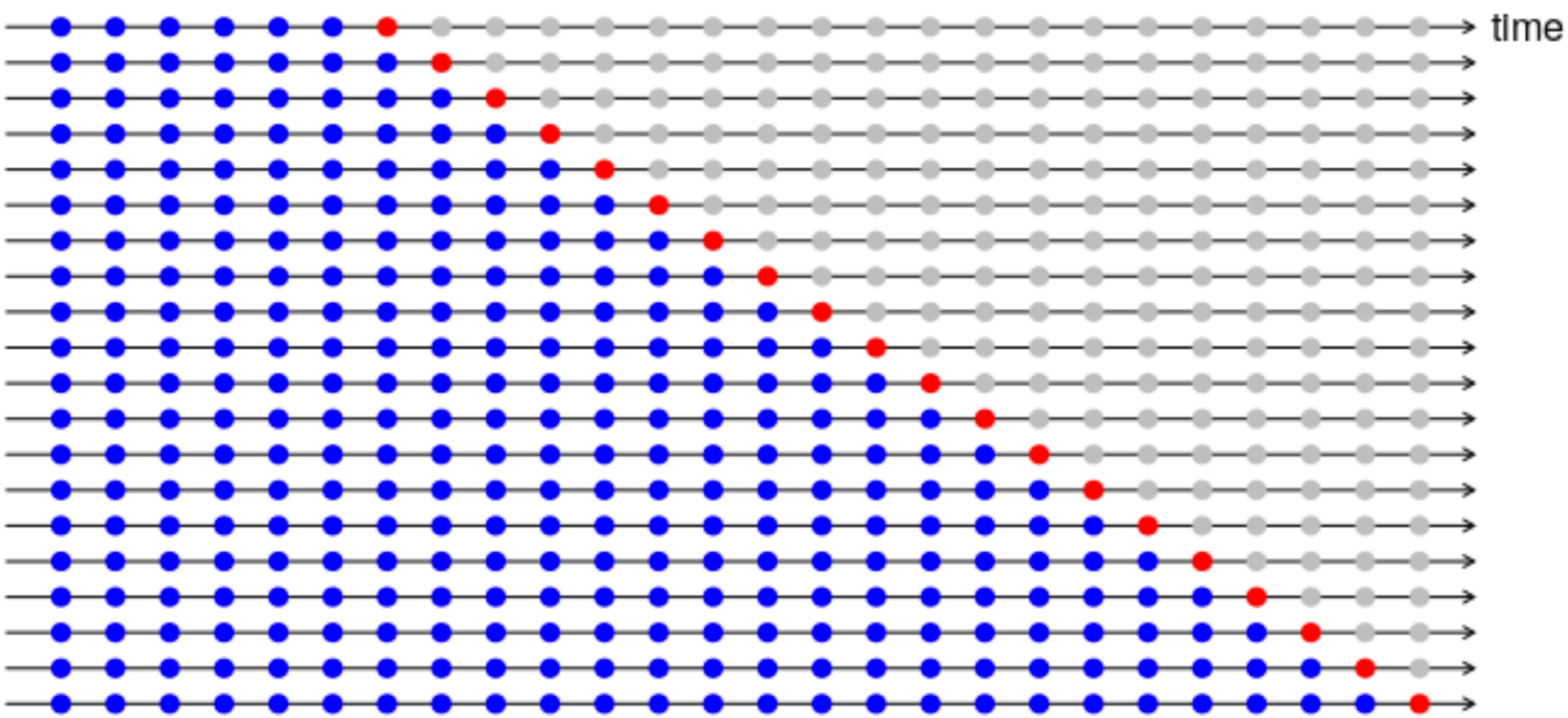
Time Series Patterns





How to Evaluate a Forecast

- K-fold cross-validation does not work. Why?
- Instead use training sets that occur prior to the test data. Training sets are grown until they contain all of the data.
- This respects the dependency structure of the time series.



Baseline Forecasting Methods

Naive

- Predict all future forecasts to be the value of the last observation:

$$\hat{y}_{T+h|T} = y_T$$

```
library(forecast)
> naive(y, h)
```

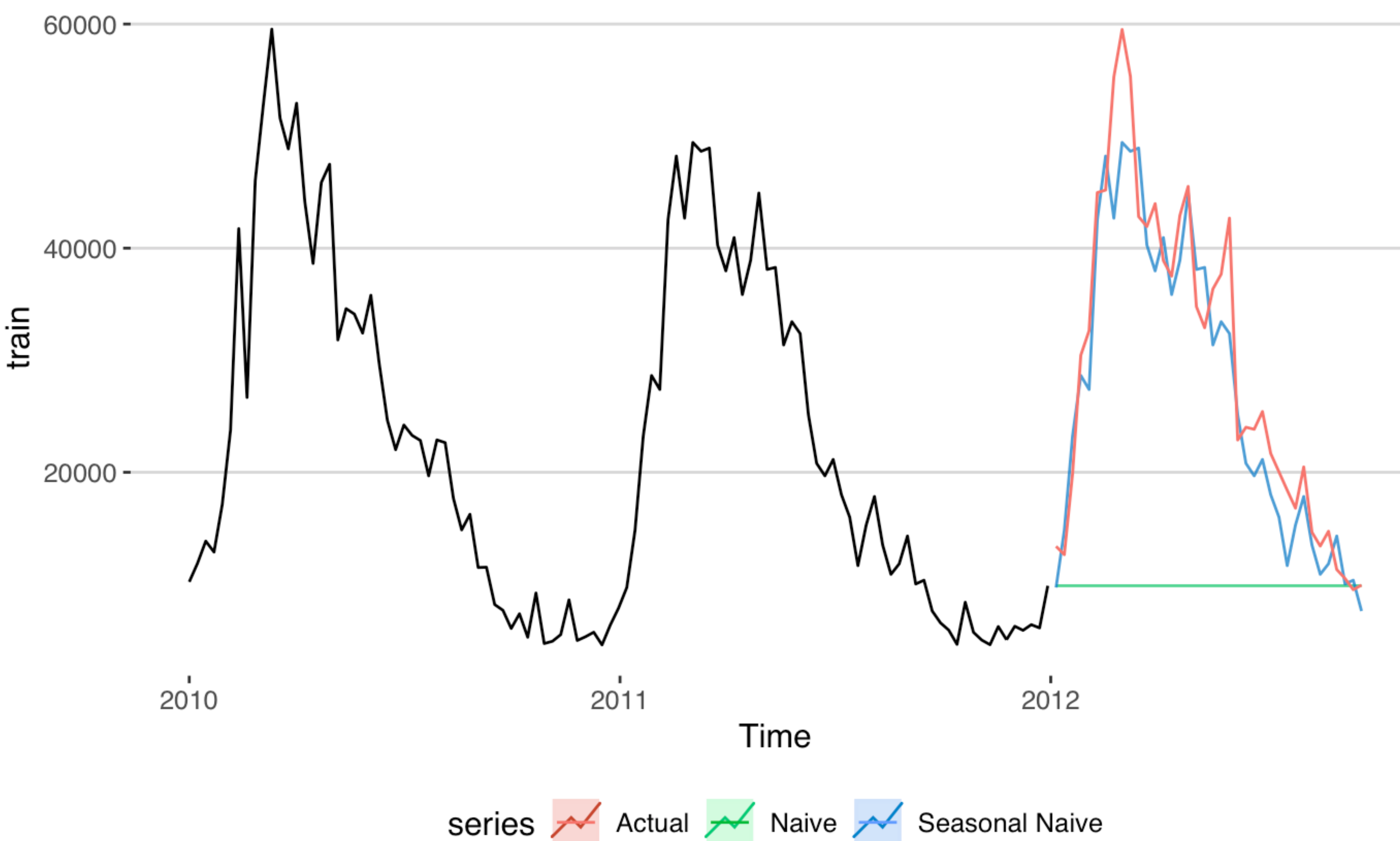
Seasonal Naive

- Predict all future forecasts to be equal to the last observed value from the same season of the year (e.g. the same week of the previous year)

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$$

```
library(forecast)
> snaive(y, h)
```

Baseline Forecasting Methods



Time Series Regression Models

- Through clever feature engineering many machine learning models can learn time series patterns.

- How do we model Trend?

- How do we model Seasonality?

Time Series Regression Models

Modeling Trend

- Include t = 1, … , T as a predictor to the model. For linear regression:

$$y_t = \beta_0 + \beta_1 t + \epsilon_t$$

```
library(forecast)
> f_tslm <- tslm(y ~ trend)
> forecast(f_tslm, h)
```

- Can include higher order terms (t^2, t^3, …) to model non-linear trends.

- Caveat: Does not work for tree based methods. Why?

Time Series Regression Models

Modeling Seasonality

- Include seasonal dummy variables, e.g. for quarterly data the categorical variable has four levels. One for each quarter.

$$y_t = \beta_0 + \beta_1 d_{1,t} + \beta_2 d_{2,t} + \beta_3 d_{3,t} + \epsilon_t$$

```
library(forecast)
> f_tslm <- tslm(y ~ season)
> forecast(f_tslm, h)
```

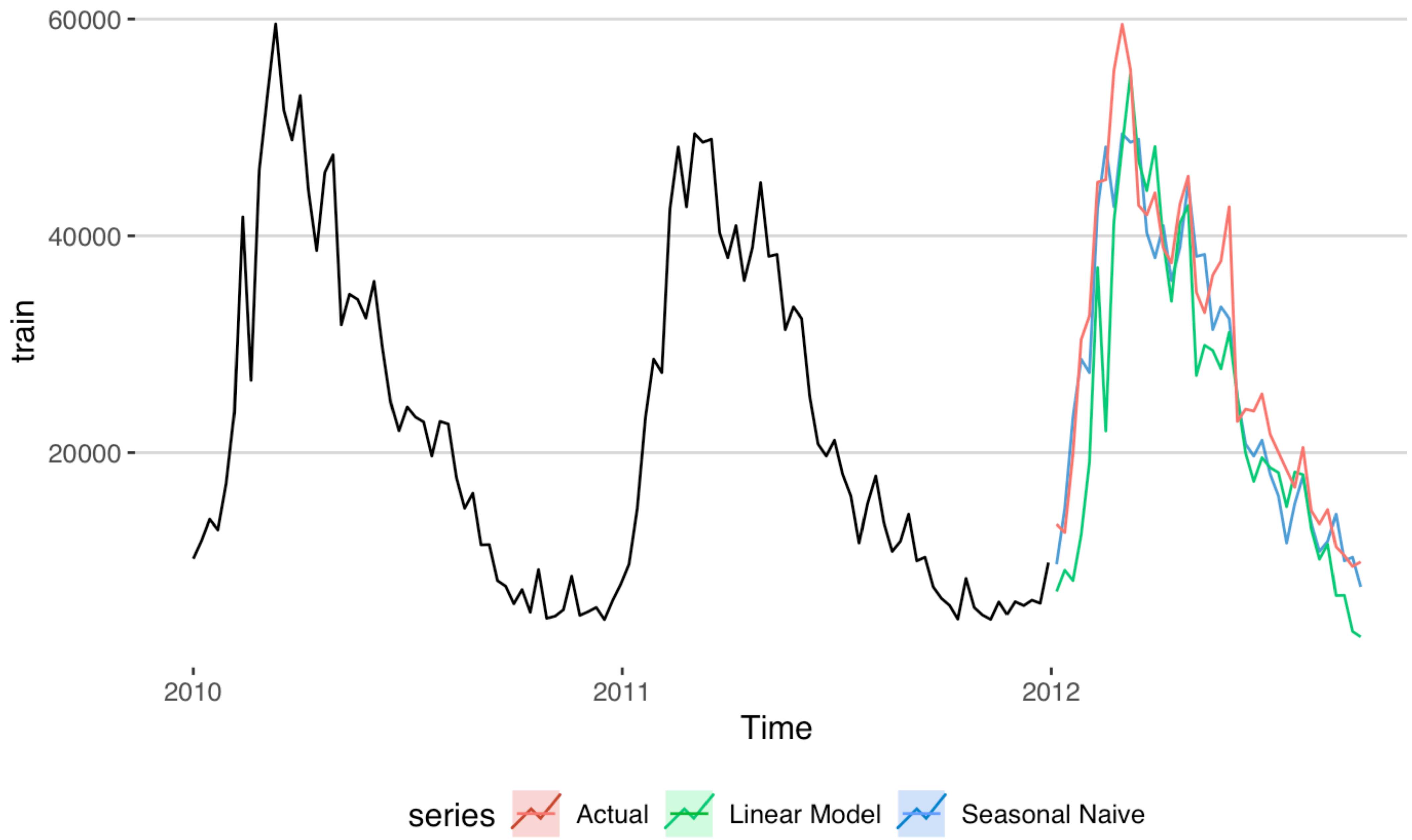
Trend + Seasonality

- Of course you can use both:

$$y_t = \beta_0 + \beta_1 t + \beta_2 d_{1,t} + \beta_3 d_{2,t} + \beta_4 d_{3,t} + \epsilon_t$$

```
library(forecast)
> f_tslm <- tslm(y ~ trend + season)
> forecast(f_tslm, h)
```

Time Series Regression Models



Time Series Regression Models

Harmonic Regression (Fourier Terms) for Seasonality

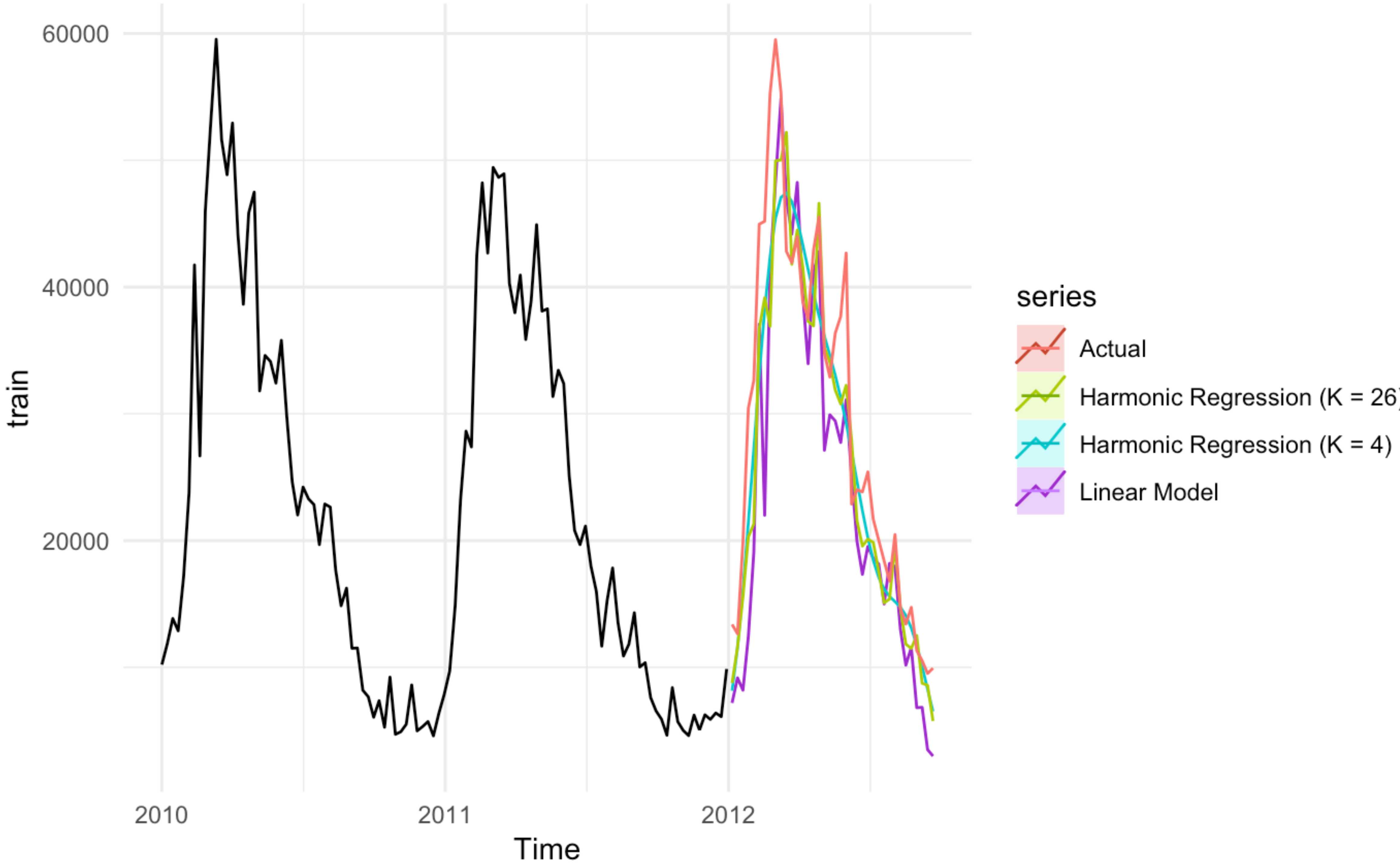
- Use Fourier terms as features. Alternative to seasonal dummy variables. Useful if there are too many categories. If m is the seasonal period, then the predictors are:

$$x_{1,t} = \sin\left(\frac{2\pi t}{m}\right), \quad x_{2,t} = \cos\left(\frac{2\pi t}{m}\right) \quad x_{3,t} = \sin\left(\frac{4\pi t}{m}\right), \quad x_{4,t} = \cos\left(\frac{4\pi t}{m}\right), \ldots$$

```
library(forecast)

> f_tslm <- tslm(y ~ trend + fourier(y, K = 4))
> forecast(f_tslm, newdata = fourier(y, K = 4, h))
```

- What happens when we have m/2 terms?

Time Series Regression Models



Time Series Decomposition

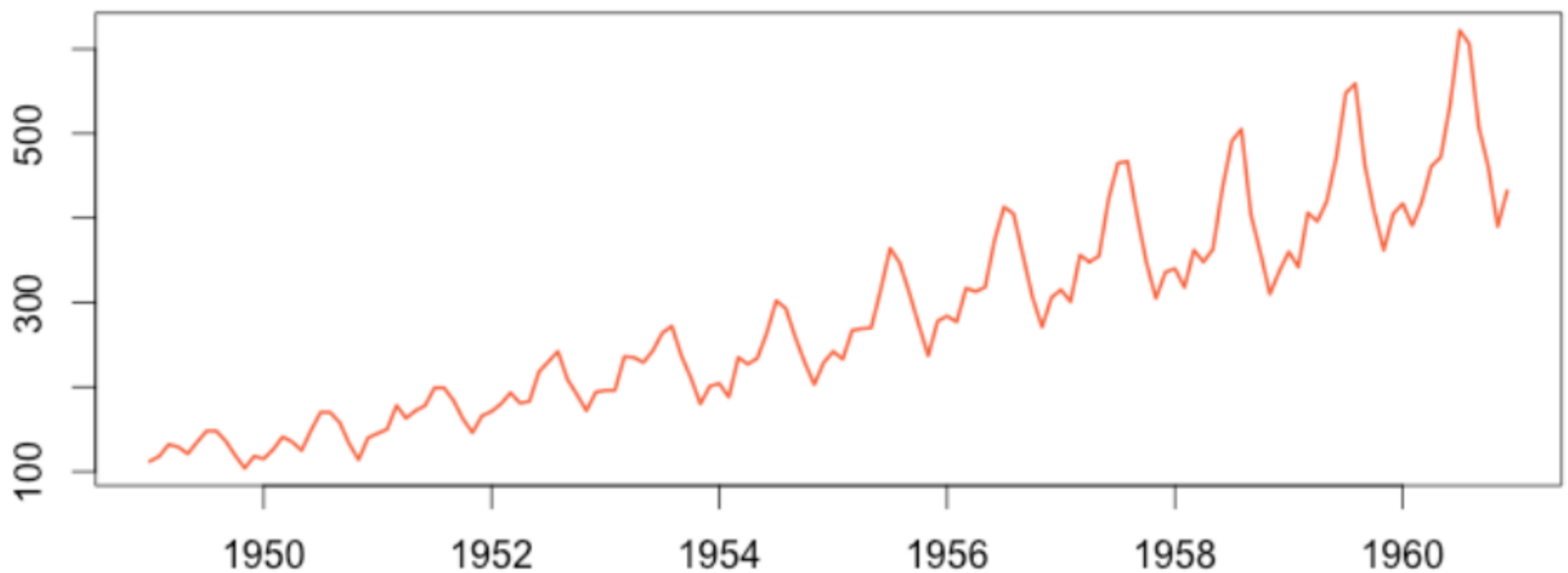
- Idea: Use smoothing functions (splines, local regression) to decompose a time series into a Seasonal, Trend, and Random component.

$$y_t = f(S_t, T_t, R_t) = S_t + T_t + R_t$$

Time Series Decomposition

- Note: Sometimes the decomposition is multiplicative, e.g. magnitude of seasonality or variation in the series increases with the trend. In this case take a log transform of y.

$$y_t = S_t \times T_t \times R_t \implies \log(y_t) = \log(S_t) + \log(T_t) + \log(R_t)$$



- Decompositions in-between multiplicative and additive can be modeled by applying a Box-Cox transformation.

STL: Seasonal Trend decomposition using LOESS

Iteratively re-weighted LOESS estimates of $\hat{S}_t$, $\hat{T}_t$ and $\hat{R}_t$.

At each iteration k, let $\hat{S}_t^k$, $\hat{T}_t^k$ and $\hat{R}_t^k$ be the current estimates of the various components and ρ_t^k be the sample weights. For the first iteration each component is set to zero and the weights are set to one. At each iteration:

1. Estimate the seasonal component $\hat{S}_t^k$ by LOESS smoothing cyclical sub-series (e.g. January, February, March) of the de-trended series $y_t - \hat{T}_t^k$. At this point $\hat{S}_t^k$ may still contain a trend, so a low-pass filter is applied to de-trend each sub-series.

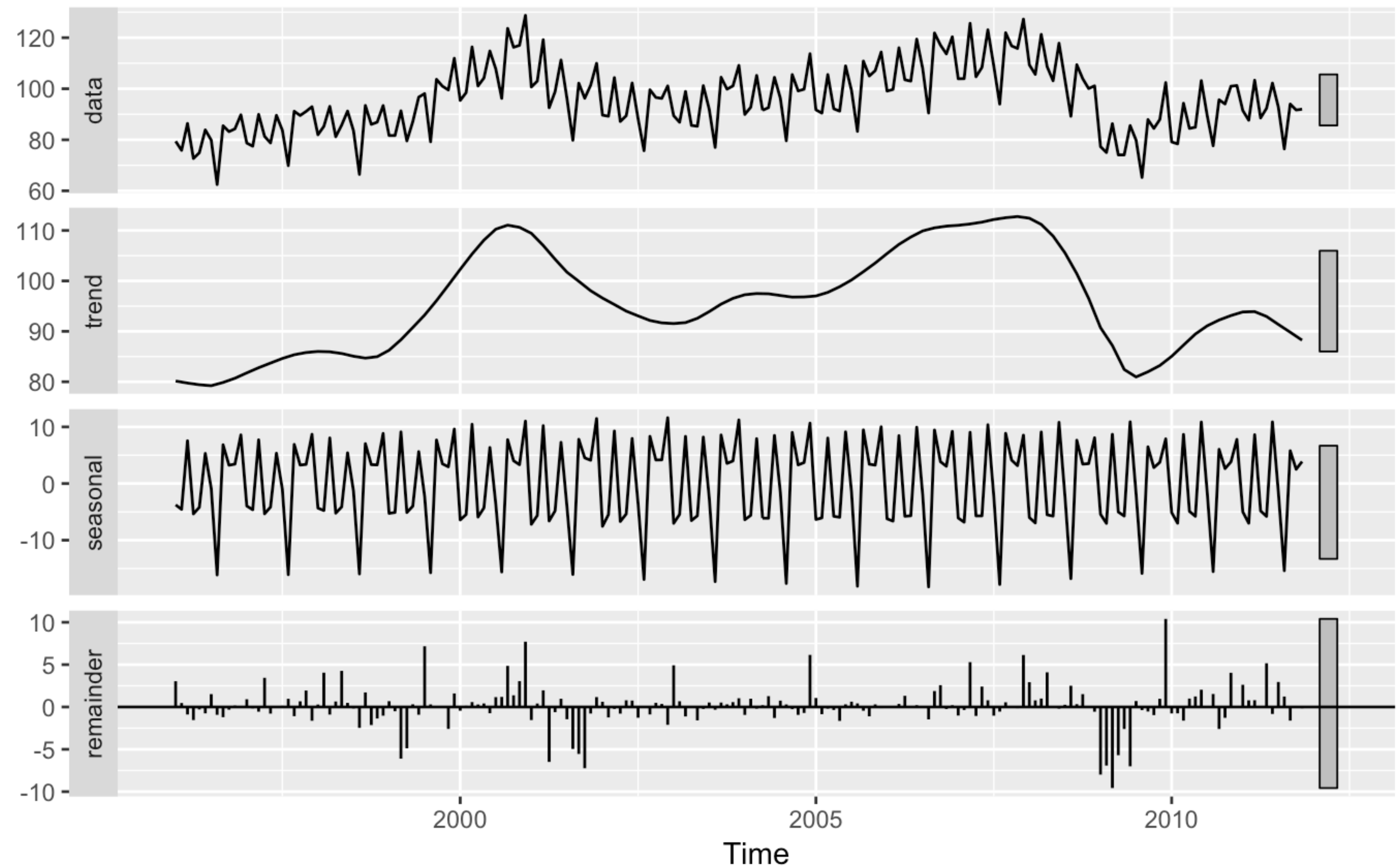
2. Estimate the trend component $\hat{T}_t^k$ by LOESS smoothing the de-seasonalized series: $y_t - \hat{S}_t^k$.

3. Estimate the random component (residuals) as $\hat{R}_t^k = y_t - \hat{T}_t^k - \hat{S}_t^k$. Based on these residuals calculate robustness weights, ρ_t^k. These weights are used in subsequent passes of LOESS.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. J. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, *6*(1), 3–73. http://www.jos.nu/Articles/abstract.asp?article=613

STL: Seasonal Trend decomposition using LOESS

```
library(forecast)
library(fpp2)

stl(fpp2::elecequip, t.window=13, s.window=7)
```



STL: Seasonal Trend decomposition using LOESS

Tuning Parameters

- s.window: the span (in lags) of the loess window used to estimate the seasonal component. Should be an odd number. Higher values result in smoother estimates.

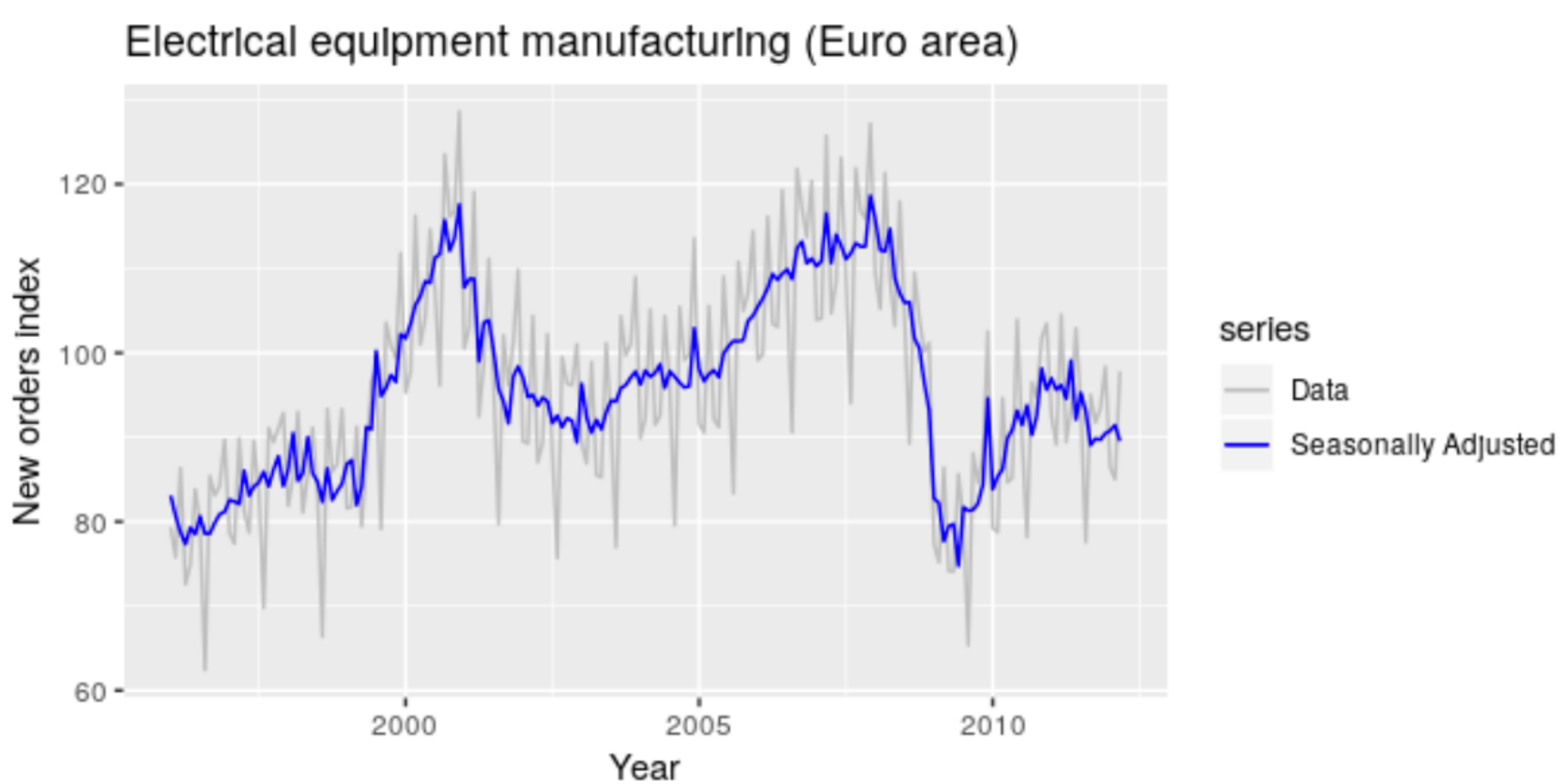
- t.window: The span (in lags) of the loess window used to estimate the trend component. Should be an odd number. Higher values result in smoother estimates.

Seasonal Adjusted Forecasts

- STL **is not** a forecasting method. To produce a forecast we use the following decomposition:

$$y_t = \hat{S}_t + \hat{A}_t \qquad \hat{A}_t = \hat{T}_t + \hat{R}_t$$

- Â is the seasonal adjusted component, i.e. the time series with the seasonality removed:



Seasonal Adjusted Forecasts

- To forecast a decomposed time series, we forecast the seasonal component and the seasonal adjusted component separately.

- Seasonal Component: Seasonality usually does not change much across periods. Typically a seasonal naive model is applied to the estimated seasonal component.

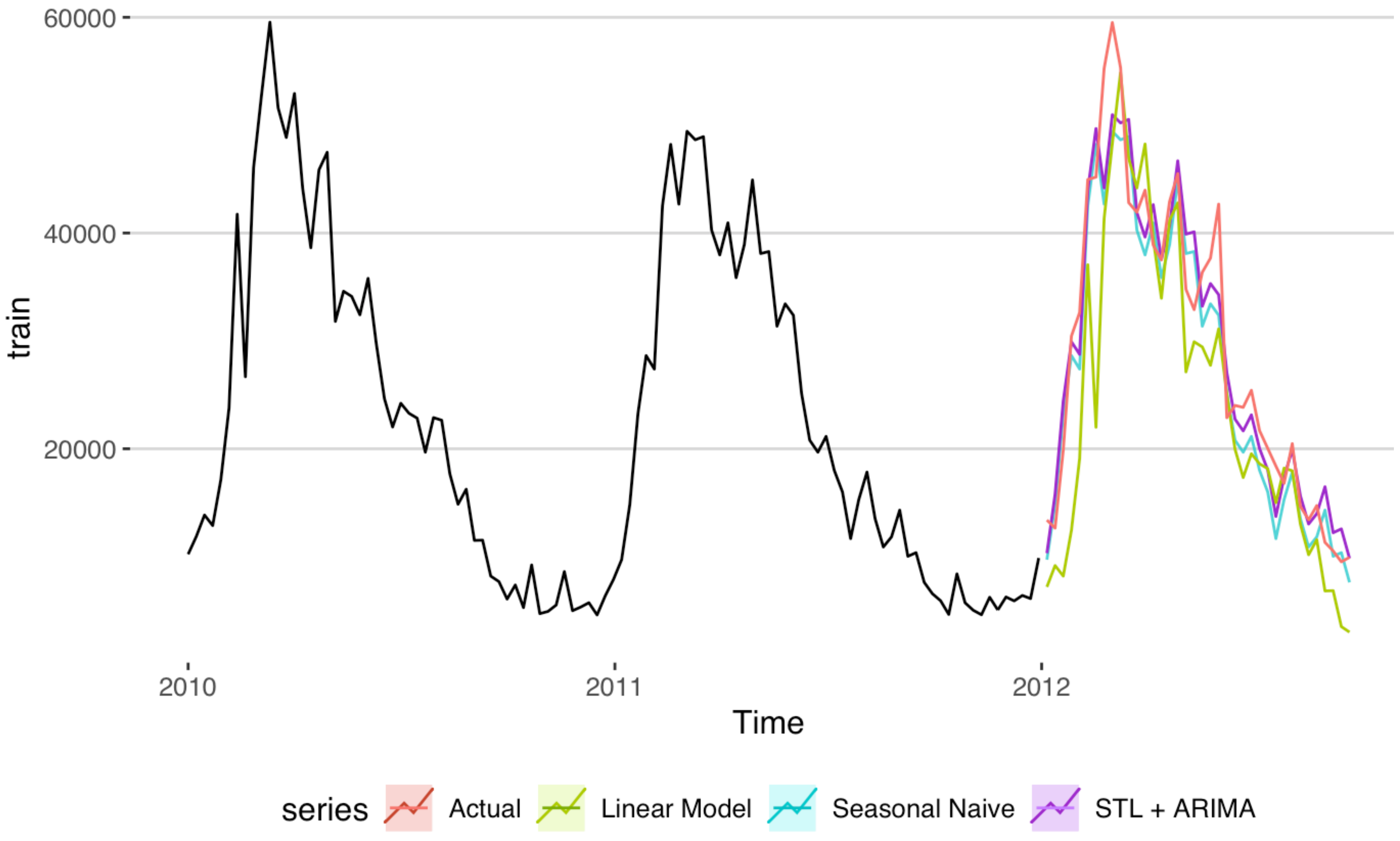
- Seasonal Adjusted Component: Any non-seasonal forecasting method may be used. Good choices are ETS or ARIMA models.

Seasonal Adjusted Forecasts

```
library(forecast)

> stlf(y, h = horizon, t.window = 13 s.window = 7,
       method = 'arima', ic = 'bic')

> stlf(y, h = horizon, t.window = 13 s.window = 7,
       method = 'ets', ic = 'aic', opt.crit = 'mae')
```

- method: How the seasonal adjusted component is modeled.
- ic: Information criterion. Both ARIMA and ETS have hyper-paramters that need to be chosen. Selects the best model based on either AIC or BIC.
- opt.crit: Only for ETS. Optimization criterion used to estimate the model's parameters.

Seasonal Adjusted Forecasts



De-noising Multiple Time Series

- Recall: We have T measurements on m time series, $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(m)}$ which are the columns of the matrix **Y**.
- Idea: **Y** is a noisy version of some "ground truth" signal that is approximately low rank (once we remove the noise). Using a low rank approximation to **Y** might increase the signal to noise ratio.
- From a previous lecture we know that the top k principle components are the best rank k approximation of the original dataset.
- We can de-noise a collection of correlated time series by applying PCA to **Y** and choosing the top k PCs.

De-noising Multiple Time Series

- An algorithm for PCA is to center and scale each feature and then run SVD. Let **Y*** denote the centered and scaled version of **Y**.

- The de-noising algorithm is as follows:

  1. Compute the SVD decomposition of **Y***

  $$Y^* = U\Sigma V$$

  2. Use the truncated decomposition with only *k* components for modeling:

  $$\tilde{Y} = U_k \Sigma_k V_k$$

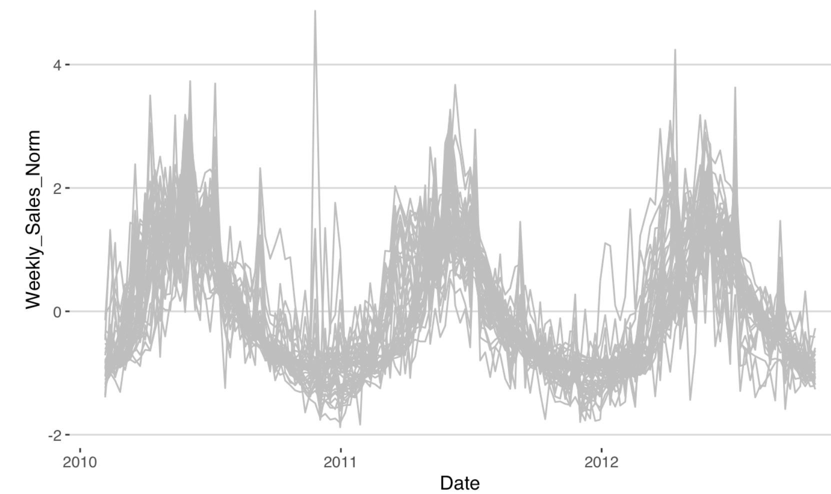
De-noising Multiple Time Series

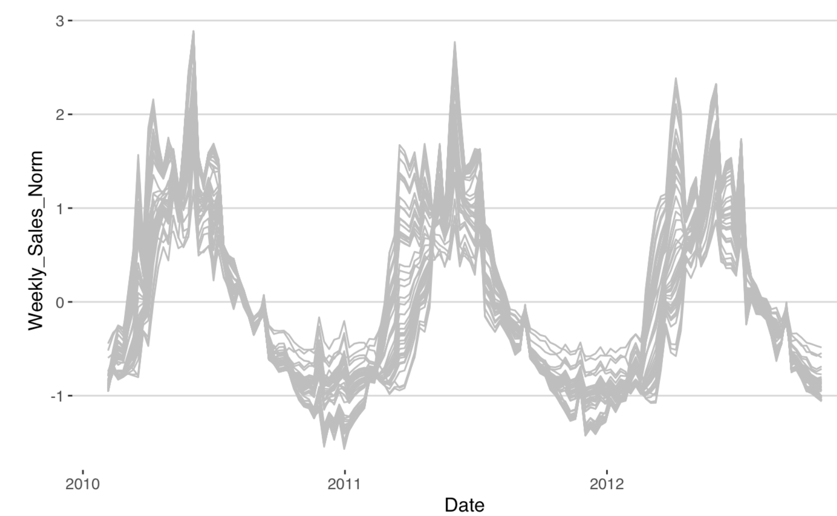
Rank 2 Approximation

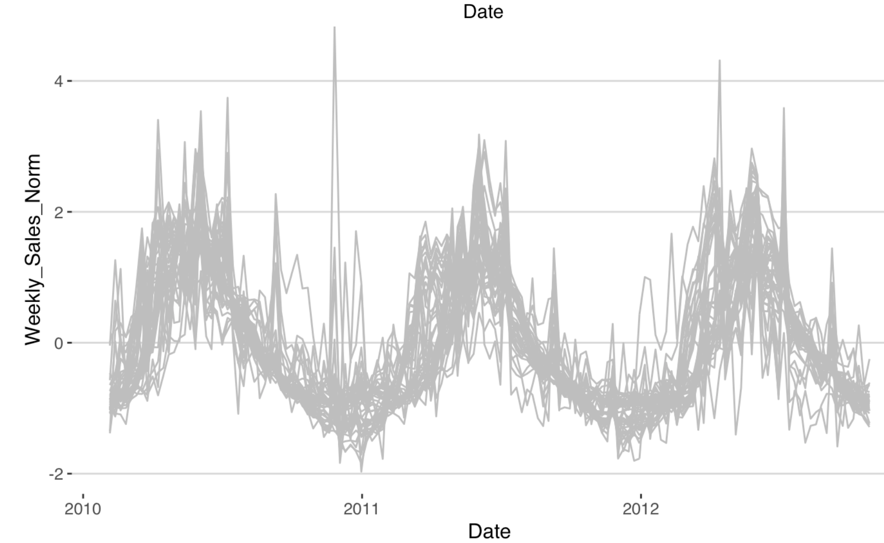
$$\tilde{Y} = U_2 \Sigma_2 V_2$$

Rank 12 Approximation

$$\tilde{Y} = U_{12} \Sigma_{12} V_{12}$$







Lagged Features

- Lagged Features: Use the past p values of the time series to predict the current value, e.g.

$$y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t$$

```
> y <- c(1, 2, 3)
> cbind(y, lag(y, 1), lag(y, 2))
     y
[1,] 1 NA NA
[2,] 2  1 NA
[3,] 3  2  1
```

- Seasonal Lagged Features: Include the value of the time series at the previous season, e.g. last 12 months.

- Note: ARIMA and ETS models build lags into the model.

Further Resources

- *Forecasting: Principles and Practice* by Rob J Hyndman and George Athanasopoulos.
  - Free online at https://otexts.org/fpp2/

- Winner's code for the Walmart Challenge.
  - https://github.com/davidthaler/Walmart_competition_code