more:

<https://en.wikipedia.org/wiki/Forecasting#Average_approach>

**注意：在时间序列预测中，经常会出现均值模型预测精度优于其他模型！如根据2009年-2015年的月度数据预测2016年1-12月，则计算2009年-2015年该指标的平均值（12个平均值），训练集则2009年-2015年（7年）重复该指标7次(rep(12个平均值,7))，测试集则重复该指标1次。将训练集平均值作为测试集的预测值，可用于比较不同模型的预测精度。**

#### simple model

In forecasting, you *very* often find that extremely simple methods, like

* the overall mean
* the naive random walk (i.e., the last observation used as a forecast)
* a seasonal random walk (i.e., the observation from one year back)
* Single Exponential Smoothing

outperform more complex methods. That is why you should always test your methods against these very simple benchmarks.

"complex" is a close relative of "overfitted."

I would aconsider ARIMA as complex method and Mean forecast as simple methods. There is ample evidence that simple methods like Mean forecast outperform complex methods like ARIMA.

##### Average approach

In this approach, the predictions of all future values are equal to the mean of the past data. This approach can be used with any sort of data where past data is available. In time series notation:



where {\displaystyle y\_{1},...,y\_{T}} is the past data.

Although the time series notation has been used here, the average approach can also be used for cross-sectional data (when we are predicting unobserved values; values that are not included in the data set). Then, the prediction for unobserved values is the average of the observed values.

##### Naïve approach

Naïve forecasts are the most cost-effective forecasting model, and provide a **benchmark** against which more sophisticated models can be compared. This forecasting method is only suitable for time series [data](https://en.wikipedia.org/wiki/Data).[[3]](https://en.wikipedia.org/wiki/Forecasting#cite_note-otexts.org-3) Using the naïve approach, forecasts are produced that are equal to the last observed value. This method works quite well for economic and financial time series, which often have patterns that are difficult to reliably and accurately predict.[[3]](https://en.wikipedia.org/wiki/Forecasting#cite_note-otexts.org-3) If the time series is believed to have seasonality, **seasonal naïve approach** may be more appropriate where the forecasts are equal to the value from last season. The naïve method may also use a **drift**, which will take the last observation plus the average change from the first observation to the last observation.[[3]](https://en.wikipedia.org/wiki/Forecasting#cite_note-otexts.org-3) In time series notation:



##### Drift method

A variation on the naïve method is to allow the forecasts to increase or decrease over time, where the amount of change over time (called the [drift](https://en.wikipedia.org/wiki/Stochastic_drift)) is set to be the average change seen in the historical data. So the forecast for time {\displaystyle T+h} is given by



This is equivalent to drawing a line between the first and last observation, and extrapolating it into the future.

##### Seasonal naïve approach

The seasonal naïve method accounts for seasonality by setting each prediction to be equal to the last observed value of the same season. For example, the prediction value for all subsequent months of April will be equal to the previous value observed for April. The forecast for time {\displaystyle T+h} is:[[3]](https://en.wikipedia.org/wiki/Forecasting#cite_note-otexts.org-3)

{\displaystyle {\hat {y}}\_{T+h|T}=y\_{T+h-km}}

where {\displaystyle m}= seasonal period and {\displaystyle k} is the smallest integer greater than {\displaystyle (h-1)/m}.

The seasonal naïve method is particularly useful for data that has a very high level of seasonality.

#### case

There is a chapter called Extrapolation for Time-Series and Cross-Sectional Data which also available free in the same [website](http://www.forecastingprinciples.com/paperpdf/extrapolation.pdf). The following is the quote from the chapter

"For example, in the real-time M2-competition, which examined 29 monthly series, Box-Jenkins proved to be one of the least-accurate methods and its overall median error was 17% greater than that for **a naive forecast**"

There is also another paper and an ongoing study by Greene and Armstrong entitled "[Simple Forecasting: Avoid Tears Before Bedtime](http://www.kestencgreen.com/simplefor.pdf)" in the same website. The authors of the paper summarize as follows:

In total we identified 29 papers incorporating 94 formal comparisons of the accuracy of forecasts from complex methods with those from simple—but not in all cases sophisticatedly simple—methods. Eighty-three percent of the comparisons found that forecasts from simple methods were more accurate than, or similarly accurate to, those from complex methods. On average, the errors of forecasts from complex methods were about 32 percent greater than the errors of forecasts from simple methods in the 21 studies that provide comparisons of errors

* Makridakis (Pioneered Empirical competition on Forecasting called M, M2 and M3, and paved way for evidence based methods in forecasting)
* Armstrong (Provides valuable insights in the form of books/articles on Forecasting Practice)
* Gardner (Invented Damped Trend exponential smoothing another simple method which works surprisingly well vs. ARIMA)

All of the above researchers advocate, simplicity (methods like your mean forecast) vs. Complex methods like ARIMA. So you should feel comfortable that your forecasts are good and always favor simplicity over complexity based on empirical evidence.

<http://stats.stackexchange.com/questions/124955/is-it-unusual-for-the-mean-to-outperform-arima>

#### case

This wasn't one of my selected entries in this competition, but it is a good example of how sometimes very simple models can punch far above their weight. The model just groups the training data by city and source (reduced to 3 levels: remote\_api\_created, city\_initiated and everything else), takes a mean (in logspace) and applies those values as predictions, which are then sent back to raw space. Using the last 4 weeks of the data, this gets 0.31499 against the private leaderboard, which would rank in the high 70's, easily inside the top 25%. A refactored, turnkey version of it is attached, but the gist of it is here:

mean\_vals = train.groupby(['city', 'src']).mean()  
test = test.merge(mean\_vals,  
                  how = 'left',   
                  left\_on = ['city', 'src'],  
                  right\_index = ['city','src'],  
                  sort = False,  
                  copy = False)

This just uses python/pandas, with no real algorithm other than grouping and aggregation.

<https://www.kaggle.com/c/see-click-predict-fix/discussion/6466#35490>