Vanderbilt Case Study Technical Appendix: Time Series Methods

IDS.506 - Health Info. Mgmt. Analytics

Robert Duc Bui - mbui7 - 660809303

Environment

Our feature engineering and modeling process will primarily be based on the `tidyverse` ecosystem and its time series focused sister ecosystem `tidyverts`. Specifically, we will be using `tsibble` to handle time series objects, `feasts` to perform STL decomposition of the time series, and `fable` to perform ARIMA modeling on the time series and its regressors.

Miscellaneous beautification and presentation will be done using `patchwork`, `knitr`, and `kableExtra`. This notebook was created in R 4.1.2, RStudio version 2021.09.2, running on x64 Windows 11. Knitting to PDF is performed using `TinyTeX`, a lightweight distribution of `LaTeX` built for RStudio.

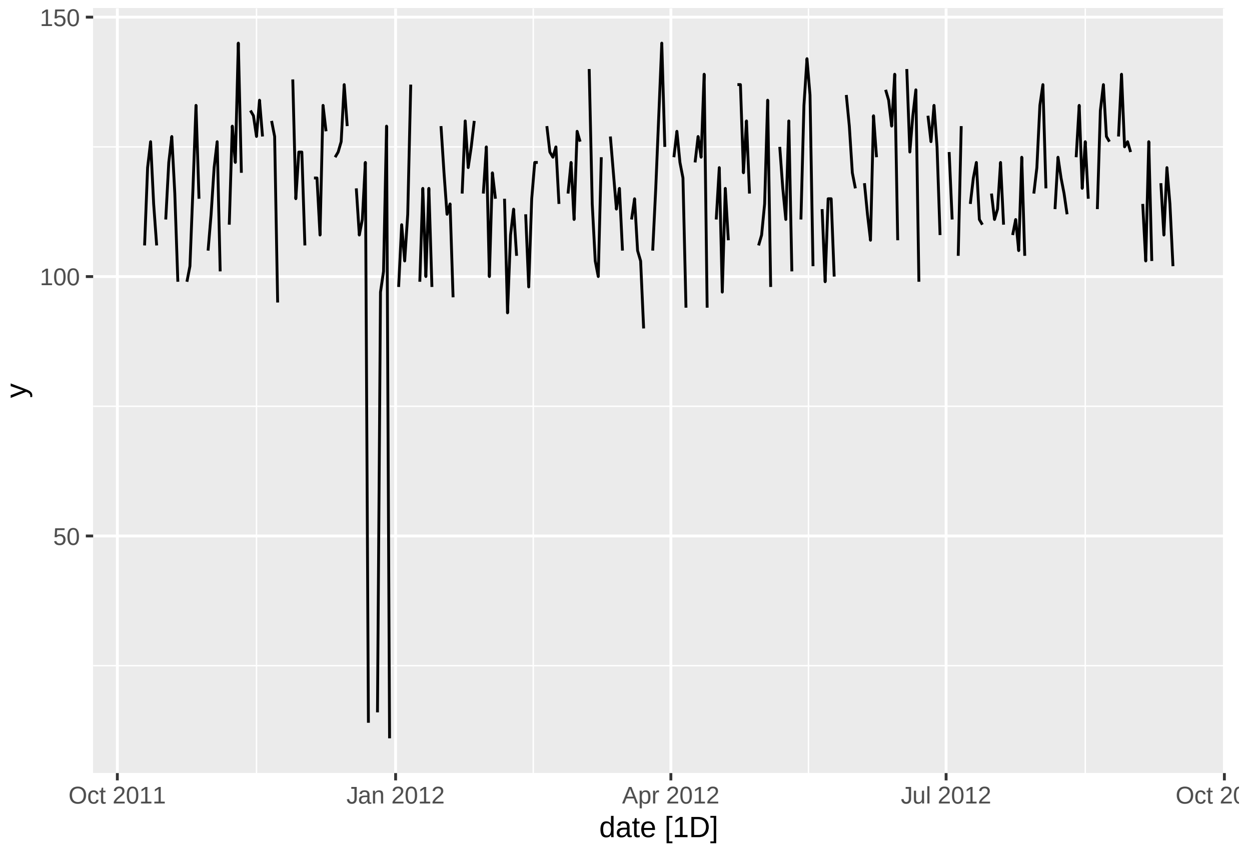
```r
# General purpose & data import/manipulation
library(tidyverse)
library(stringr)
library(readxl)

# Time-series analysis
library(tsibble)
library(fable)
library(feasts)

# Misc.
library(patchwork)
library(knitr)
library(kableExtra)
```

Importing & Observing time series

After reading the `.xlsx` file into R and performing some basic variable name cleaning, we can use `autoplot` to quickly visualise the time series of actual surgeries. Note that gaps are present in the data, representing weekends where no surgeries are carried out.

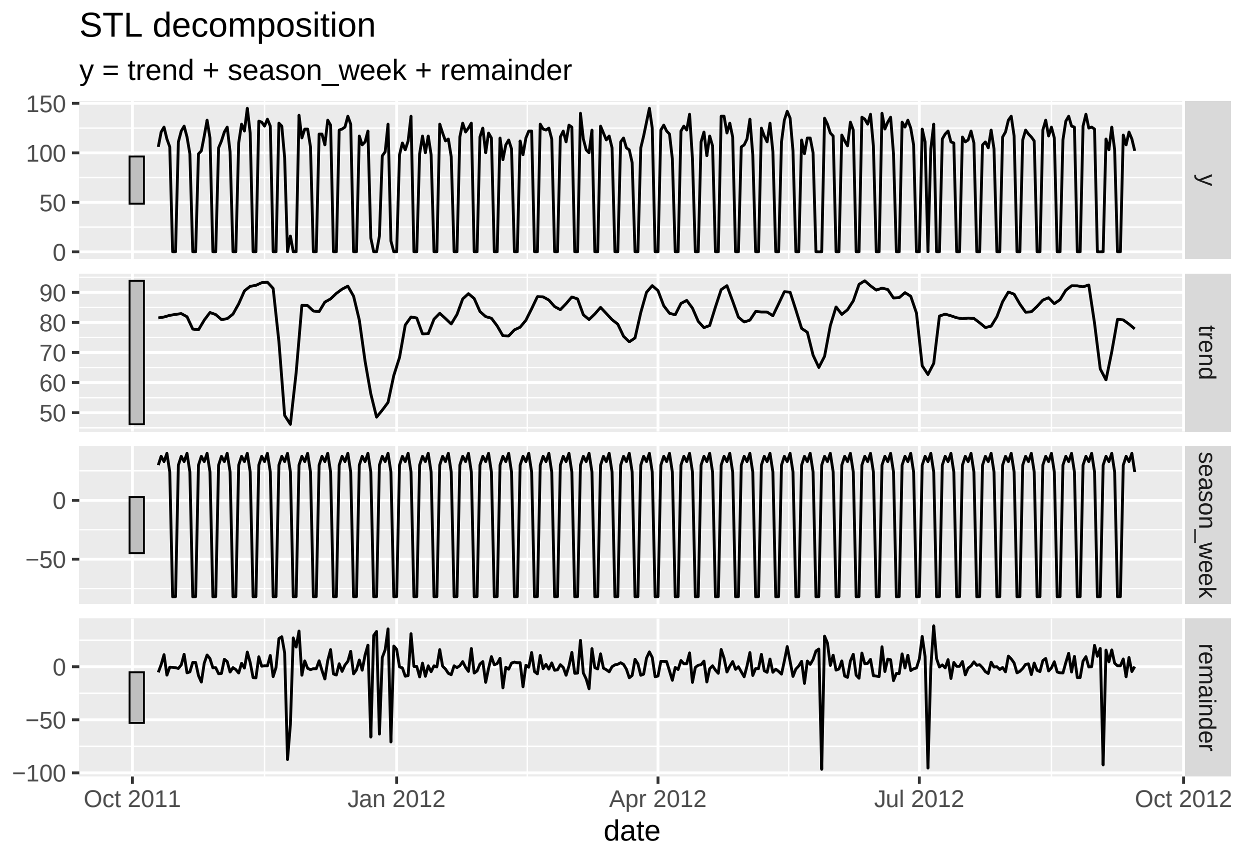


Aside from three precipitous dips in scheduled surgeries, which we surmise to be special outlier events, there is little evidence indicating heteroskedasticity. We will not be using any transformations (eg. Box-Cox) on this dataset.

STL Decomposition

To ascertain whether common time series methods can be used to analyse this dataset, we can perform a STL decomposition to observe its constituent components. Seasonality here is assumed to be weekly, and the result of our decomposition can be seen below:

```
dcmp_stl <- ts %>% replace(is.na(.),0) %>% model(STL(y~season(window=Inf)))
components(dcmp_stl) %>% autoplot()
```



As the decomposition plot above shows, there is only a limited seasonality effect in our dataset - most of the undulations of our target variable are decomposed into the Trend component and random error. Seasonality effect is not non-existent however, so we can still perform standard methods such as AutoRegressive Integrated Moving Average (ARIMA) models.

ARIMA and `fable`: An Overview

The main model that we will be constructing is the aforementioned ARIMA model. This is a time-series specific form of regression analysis that predicts future series by examining differences between past values in the series. An ARIMA model has three main components (which very roughly correspond to the traditional STL conceptualisation of a time series):

- Autoregression (AR): predicts future values based on past values as regressors. Configured through the p parameter, denoting the number of lag observation autoregressive terms in the model.
- Integration (I): represents the differencing of raw observations to allow for the time series to become stationary. Configured through the d parameter, denoting the number of times that raw observations are differenced.
- Moving Average (MA): incorporates the dependency between an observed value and residual error from a moving average model. Configured through the q parameter, denoting the window size of the moving average.

Our modeling process will be making use of the `fable` package, which trains a model through a search for optimal (p, d, q) sets. Specifically, it uses a variation of the Hyndman-Khandakar algorithm, which obtains an ARIMA model through stepwise AICc optimisation (initialising random sets of (p, d, q), then permuting each parameter by 1, moving in the direction of best AICc).

Note that we will mostly build ARIMAX models, which is an extension of ARIMA that allows for exogenous regressors. Here the regressors are the past schedules provided in the dataset.

Building candidate models

We start modeling by performing one final check: to see if our seasonality is correctly set to a weekly frequency. Note that this frequency is defined not by *business week* intervals of 5 days, but by normal week intervals of 7 days.

```r
# Sanity check: verifying seasonality being used as 7 (weekly seasonality)
guess_frequency(ts$date)
```

```
## [1] 7
```

Having verified that the data is set to a weekly seasonality in `fable`'s backend, we continue to call `model` and train the following set of models:

- `snaive`: a Seasonal Naive model that forecasts based on last week's value. Fundamentally a random walk model, used here as a benchmark for all other models;
- `arima`: a tunedARIMA model with no regressors;
- `arima.all`: a tuned ARIMAX model with `xreg` regressors being all lagged schedule data points.;
- `arima.10`: a tuned ARIMAX model with regressors from T-10 to T-1;
- `arima.5`: a tuned ARIMAX model with regressors from T-5 to T-1;
- `arima.1`: a tuned ARIMAX model with only T-1 as regressor;
- `arima.wk`: a tuned ARIMAX model with only T-5 as regressor, essentially predicting using the schedule from a week out;
- `arima.mth`: a tuned ARIMAX model with only T-28 as regressor, essentially predicting using the schedule a month out;
- `arima000.1`: an empty ARIMA model set to (0,0,0) with only T-1 as regressor, essentially a pseudo-autoregression on T-1 with random error terms;
- `arima000.5`: an empty ARIMA model set to (0,0,0) with T-5 to T-1 as regressor, essentially a pseudo-autoregression on week-of scheduling data with random error terms;
- `arima000.10`: an empty ARIMA model set to (0,0,0) with T-10 to T-6 as regressor, essentially a pseudo-autoregression on 1-week-out scheduling data with random error terms.
- `arima000.wk`: an empty ARIMA model set to (0,0,0) with T-28 to T-6 as regressor, essentially a pseudo-autoregression on all available 1-week-out scheduling data with random error terms.
- `arima500.wk`: an ARIMAX model set to (5,0,0) with T-28 to T-5 as regressor, using lagged time series with `xreg` from on all available 1-week-out scheduling data.

The code for this modeling process is included below.

```r
# Setting seed for replicability
set.seed(8675309)

# Training models
fc <- ts %>%
  model(
    # Seasonal naive random walk - taking last week's value
    snaive = SNAIVE(y),

    # ARIMA search only, no regressors
    arima.0     = ARIMA(y),

    # ARIMAX search + all regressors
    arima.all   = ARIMA(y~t28+t21+t14+t13+t12+t11+t10+t9+t8+t7+t6+t5+t4+t3+t2+t1),
    # ARIMAX search + last 10 days regressors
    arima.10    = ARIMA(y~t10+t9+t8+t7+t6+t5+t4+t3+t2+t1),
    # ARIMAX search + last 5 days regressors
    arima.5     = ARIMA(y~t5+t4+t3+t2+t1),
    # ARIMAX search + last day regressor
    arima.1     = ARIMA(y~t1),

    # ARIMAX search with t-5 lag (a week out)
    arima.wk    = ARIMA(y~t5),
    # ARIMAX search with t-28 lag (a month out)
    arima.mth   = ARIMA(y~t28),

    # ARIMA(0,0,0) or ARMA(0,0), with last day t-1 regressor
    arima000.1 = ARIMA(y~t1 + pdq(0,0,0)),
    # ARIMA(0,0,0) or ARMA(0,0), with t-4 (week of) regressors
    arima000.5 = ARIMA(y~t4+t3+t2+t1 + pdq(0,0,0)),
    # ARIMA(0,0,0) or ARMA(0,0), with t-10 to t-5 (a week out) regressors
    arima000.10= ARIMA(y~t10+t9+t8+t7+t6+t5 + pdq(0,0,0)),
    # ARIMA(0,0,0) or ARMA(0,0), with t-28 to t-5
    #(all available regressors, a week out) regressors
    arima000.wk= ARIMA(y~t28+t21+t14+t13+t12+t11+t10+t9+t8+t7+t6+t5 + pdq(0,0,0)),
    # ARIMA(5,0,0) or ARMA(5,0), with t-28 to t-5
    #(all available regressors, a week out) regressors
    arima500.wk= ARIMA(y~t28+t21+t14+t13+t12+t11+t10+t9+t8+t7+t6+t5 + pdq(5,0,0))
  )
```

Evaluating candidate models

Below we can see the performance of our models, measured in terms of RMSE, MAPE, and AICc.

- RMSE: Root-mean-square error, a classic in statistical learning.
- MAPE: Mean absolute percentage error, measuring the forecasting error as a percentage of the true value.
- AICc: the second-order term for the Akaike Information Criterion, defined as:

$$2(k - \ln(\hat{L}) + \frac{k^2 + k}{n - k - 1})$$

which penalises models with large k parameters, low n sample size, and low $\hat{L}$ maximum likelihood. To put it simply, the smaller the AICc, the better.

Model performance here is arranged in ascending order of MAPE. We use MAPE as the main benchmark because a simple percentage value is easily understandable by both analysts and non-technical management personnel.

Table 1: TS Model Performance

.model	.type	RMSE	MAPE	AICc
arima.all	Training	4.401909	3.251324	1451.933
arima.10	Training	4.522224	3.361579	1442.145
arima.5	Training	4.558495	3.441014	1440.565
arima000.5	Training	4.608516	3.458619	1432.631
arima000.1	Training	4.631349	3.480340	1428.832
arima.1	Training	4.625476	3.494169	1432.562
arima000.wk	Training	7.054323	6.481219	1654.872
arima000.10	Training	7.245671	6.680365	1654.917
arima.wk	Training	7.564974	7.103524	1665.341
arima.mth	Training	13.649232	15.152928	1954.547
snaive	Training	22.102803	19.939612	NA
arima.0	Training	17.350197	20.074839	2067.499
arima500.wk	Training	NaN	NaN	NA

As of current build, our `arima500.wk` model is not returning its in-sample accuracy. In a real-world scenario and with more time, we could examine why this is the case, as we surmise it might give us some insight into the specific quirks of this particular dataset and model. For the time being, we set it aside and consider the remaining models available to us.

Conclusion

Judging from the above table, we can see that any ARIMAX model that uses the available regressors are substantially more performant than the random walk baseline, or the no-regressor ARIMA model. This is consistent with our previous guess that the low seasonality and high randomness of the time series itself means that any model that Vanderbilt uses must make more use of regressors, here being the available schedule.

The top three models all make use of a mix of the ARIMA method and the `xreg` regressor terms, with `arima.all` having the lowest MAPE. Of course, this model has higher AICc due to the fact that it makes use of all regressors on top of the ARIMA (p, d, q) terms. Switching to the second-best model `arima.10` which makes use of the last ten days of schedule drops our model complexity while only trading off about .11 in terms of MAPE.

Should Vanderbilt wish to build a model that can forecast from longer out, `arima000.wk` is a middle-of-the-road solution: by setting $(p, d, q) = (0, 0, 0)$, we effectively ignore the actual time series (which results in a moderate drop in accuracy) and only use regressors T-28 to T-5 (which results in a model that can forecast a week in advance). Granted, MAPE rises from ~3% (really good) to ~6.5% (middling), but this is offset by much higher actual actionability stemming from the fact that it can predict from a week out instead of on the day before.

In conclusion, this technical report has highlighted both the advantages and shortcomings of time-series specific methods (ARIMA and ARIMAX) in Vanderbilt's OR case. Absent alternative non-timeseries-specific models, we would recommend an ARMA(0,0) model with added regressors being the T-28 to T-6 schedule, which allows for forecasting from a week out.