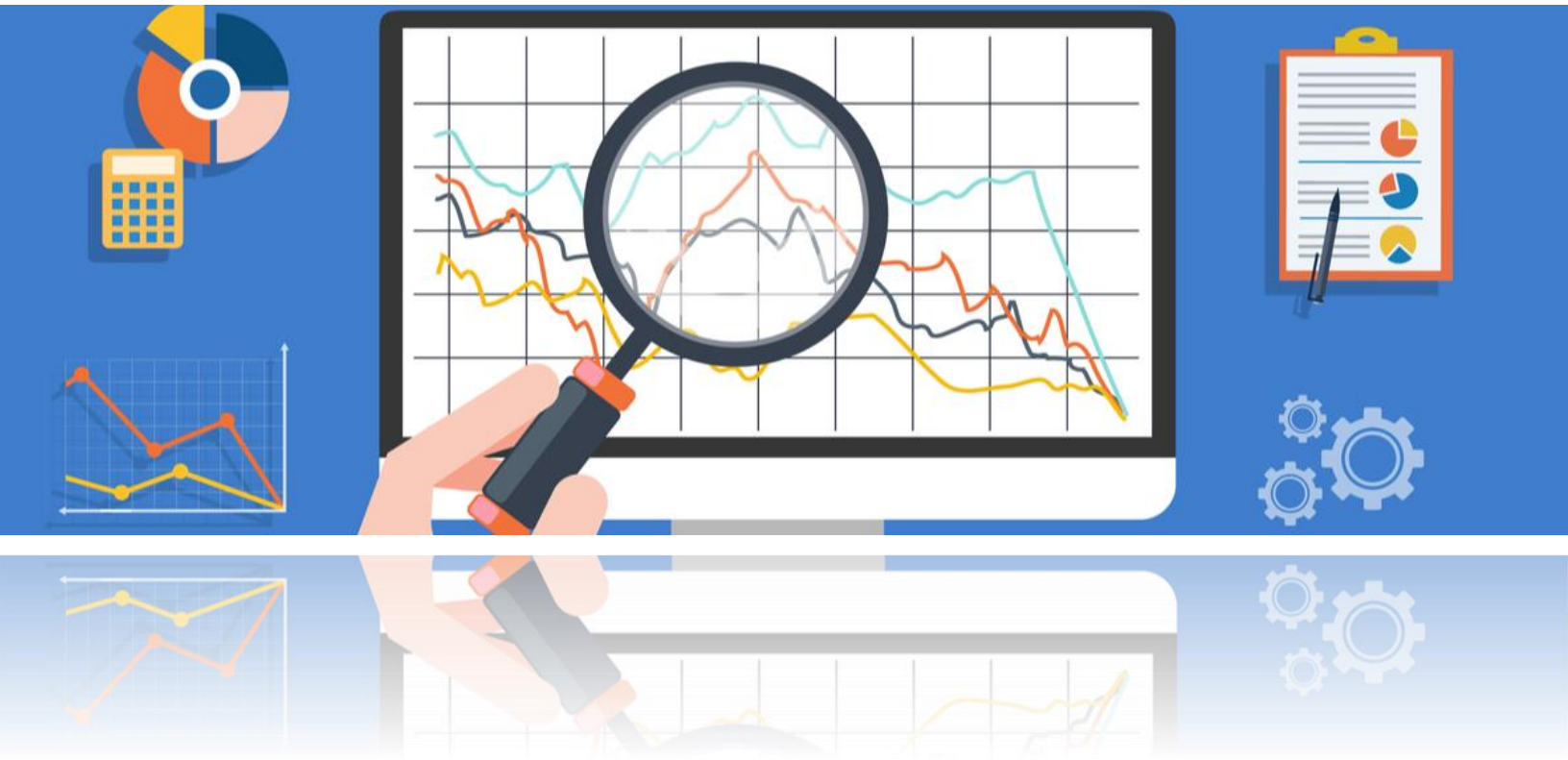


STAT902 Forecasting Competition

By

Name: Jinyang Li
ID: 20788352
Date: 2019-04-24

TABLE OF CONTENTS

I. SCENARIO 1: HYDROLOGICAL FORECAST

1. Model Specification

For this problem, we use $SARIMA\ (1,0,2) \times (1,1,2)_{12}$ to forecast the 1-month to 24-month ahead forecast of the monthly resolution of the level of body water. The definition for the SARIMA model is as below:

- SARIMA Model:

  x_t is said to follow an SARIMA (Seasonal Autoregressive Integrated Moving Average) model of orders p, d, q, P, D, Q and seasonal periods s if

$$\Phi_P(B^s)\phi(B)(1-B^s)^D(1-B)^d x_t = \Theta(B^s)\theta(B)w_t$$

  This is abbreviated as x_t follows $SARIMA\ (p,d,q) \times (P,D,Q)_s$

2. Model Selection and Estimation

The problem is this scenario is to forecast the level change of body water in monthly frequency. Intuitively, we may guess to use an $ARIMA$ model as it can captures a suite of different standard temporal structures in time series data. We know that the human body may have periodic adjustment on a yearly basis, we may further guess to use a $SARIMA$ model for better fitting and prediction.

Fig 1.1 shows the monthly resolution of the level of body water. It shows some extend of non-stationarity. By the ADF test in the notebook for scenario 1, we get the p-value of 0.079825, which indicating a non-stationarity in significance level of both 1% and 5%. Therefore, it confirms our use of the differencing in $SARIMA$ model.

Fig 1.2 shows the ACF of the data, in which we notice a seasonal effect of lag 12. This confirms out guess on seasonal differencing. The autocorrelation that remains in the residuals of the seasonally differenced data is then modeled using $ARMA\ (p,q)$ models.

We have tried to fit the data with $SARIMA$ model using multiple parameter combinations (as below) and chosen the best by the lowest AIC/BIC.

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMA: (0, 0, 1) x (0, 0, 1, 12)
SARIMA: (0, 0, 1) x (0, 0, 2, 12)
SARIMA: (0, 0, 2) x (0, 1, 0, 12)
SARIMA: (0, 0, 2) x (0, 1, 1, 12)
```

- In Python, the best parameter combination is $SARIMA\ (1,0,2) \times (1,1,2)_{12}$
- In R, by using the automatically selection function, the best parameter combination is $SARIMA\ (2,0,0) \times (1,1,0)_{12}$

In this problem, we fit the data by $SARIMA\ (1,0,2) \times (1,1,2)_{12}$ model.

Table 1.1 shows the summary of our fitted $SARIMA\ (1,0,2) \times (1,1,2)_{12}$ model.

3. Model Diagnostics

Standard model diagnostics can be found in **Fig 1.3**.

- The standardized residual looks stationary.
- By the histogram and Q-Q plot, the residual roughly follows normal distribution.
- By the ACF of residuals, they are uncorrelated.

Therefore, the residual is white noises and our time series model fit the data quite well.

4. Model Forecast

In this scenario, we want to forecast the level of body water in the future 24 months, as well as the 95% prediction and confidence bands.

Fig 1.5 is the visualization of the forecast. The right end of the whole data and the forecast is plotted for closer view. We can see a wider prediction interval than the confidence band. The reason from STAT850 is summarized as follow:

The difference between a prediction interval and a confidence interval is the standard error.

Confidence intervals

- tell you about how well you have determined the mean. Assume that the data really are randomly sampled from a pre-determined distribution. If you repeat many times and calculate a confidence interval of the mean from each sample, you'd expect about 95 % of those intervals to include the true value of the population mean. The key point is that the confidence interval tells you about the likely location of the true population parameter.
- The standard error for a confidence interval on the mean takes into account the uncertainty due to sampling.

Prediction intervals

- tell you where you can expect to see the next data point sampled. Assume that the data really are randomly sampled from a Gaussian distribution. Collect a sample of data and calculate a prediction interval. Then sample one more value from the population. If you do repeat many times, you'd expect that next value to lie within that prediction interval in 95% of the samples. The key point is that the prediction interval tells you about the distribution of values, not the uncertainty in determining the population mean.
- The standard error for a prediction interval on an individual observation takes into account the uncertainty due to sampling like above, but also takes into account the variability of the individuals around the predicted mean. The standard error for the prediction interval will be wider than for the confidence interval and hence the prediction interval will be wider than the confidence interval.

II. SCENARIO 2: FINANCIAL RISK FORECAST

1. Model Specification

For this scenario, we need to forecast (lower) 15% quantiles 10 steps ahead for each stock price series. An example of stock price process is shown in **Fig 2.1**.

By financial market knowledge, we know the stock price is uncorrelated (**Fig 2.2**) but may still be serially dependent due to a dynamic conditional variance process. A time series exhibiting conditional heteroscedasticity, or autocorrelation in the squared series, is said to have *autoregressive conditional heteroscedastic* (ARCH) effects. As indicated by **Fig 2.1**, one can observe volatility clustering in some extend.

We use the Engle's LM test to test the ARCH effect in each stock series, which is a Lagrange multiplier test to assess the significance of ARCH effects. By the Engle's LM test, the p-values of all the series are less than 1%, therefore the null hypothesis that there is no ARCH effect can be rejected at 1% significance level, meaning that the ARCH affects are quite significant in the daily log returns.

Therefore, we may guess using GARCH type of model to fit the data. The definition of a GARCH model is as following:

Let w_t be a unit variance strong white noise process. x_t is said to follow a (strong) Generalized Autoregressive Conditionally Heteroscedastic model of orders p and q ($GARCH(p,q)$) if

$$x_t = \sigma_t w_t$$
$$\sigma_t^2 = \omega + \sum_{j=1}^{p} \alpha_j x_{t-j}^2 + \sum_{l=1}^{q} \beta_j \sigma_{t-l}^2$$

2. Model Selection and Diagnostics

- Use $GARCH(1,1)$

  We first fit the data using simple $GARCH(1,1)$ with normal distribution to see if any model adjustment is needed. **Table 2.1** and **Fig 2.4** show the summary and diagnostics of the fitted $GARCH(1,1)$ model for stock 17. The fitted models for other series are similar so we only discuss this one for simplification.

  From **Table 2.1** we can see:

  - The Ljung–Box test for standardized residuals looks good, but there is some evidence of serial correlation in standardized squared residuals (**Fig 2.3**).
  - The ARCH LM test shows that we have eliminated the ARCH effect by $GARCH(1,1)$
  - Nyblom test, which tests for coefficient stability (structural change), shows no evidence for unstable parameters.
  - Sign Bias test, which examines the leverage effects, shows no or weak evidence of asymmetric effects.
  - Adjusted Pearson Goodness-of-fit test, which tests for distribution goodness-of-fit, shows that the normal distribution in the model cannot be rejected.

The **Fig 2.4** also validates the statements above in a perspective of data visualization.

By the above model diagnostics, fitting the stock series by $GARCH(1,1)$ is a good choice.

- Use $EGARCH(1,1)$

  We know that in the financial market, the stock is mostly fat-tailed distributed and there may exist leverage effect. So sometimes the simple GARCH model may not be sufficient to capture all the features in stock time series.

  As a result, the data is also fitted using $EGARCH(1,1)$ with t distribution whose result can be found in **Table 2.2**. Some other asymmetric univariate GARCH model are also fitted for the stock series, like GJR-GARCH, APARCH. The result is similar to the EGARCH one.

  However, by model diagnostics, we find out that the leverage effect is not significant, and the tail distribution follows normal distribution quite well.

After comparing the model summaries for different models, we choose $GARCH(1,1)$ with normal distribution as the final choice. The $GARCH(1,1)$ model is simple but already have the full capacity to fit the data.

3. Model Forecast

Multi-period forecasts can be produced for GARCH-type models using forward recursion. Some models, like EGARCH, that are non-linear in the sense that they do not normally have analytically tractable multi-period forecasts available.

There are three methods for forecasting using ARCH packages in Python:

- Analytical: multi-step analytical forecasts are only available for model which are linear in the square of the residual, such as GARCH or HARCH.
- Simulation: simulation-based forecasts are always available for any horizon, and is used mostly for horizons lager than 1 since the first out-of-sample forecast from an ARCH-type model is always fixed.
- Bootstrap: bootstrap-based forecasts are similar to simulation-based forecasts except that they make use of standardized residuals from the actual data used in estimation rather than assuming a specific distribution.

In the Notebook for scenario 2, we have implemented all the above three method for forecasting under $GARCH(1,1)$ and $EGARCH(1,1)$. Non-convergence issue exists when fitting some asymmetric GARCH models. We multiply the log-returns by 100 first and scale back after fitting. The situation is resolved for most cases but still remain in some specific stocks when fitting $EGARCH(1,1)$.

For the output of our 10 steps ahead forecast, we fit the data under the $GARCH(1,1)$ model with normal distribution and applying the Bootstrap method to simulate the forecasts taking the advantage of the non-parametric distribution of the actual data.

III. SCENARIO 3&4: IMPUTATION AND MULTIVARIATE TIME SERIES FORECASTING

As for scenario 3&4, we have data for monthly beer production, car production, steel production, gas consumption and electricity consumption. The imputation and forecasting will forecast on the month beer production.

1. Imputation

The scenario 3 asks for imputing (predicting) the missing values. There are 30 missing values in beer production from 1972-09 to 1975-02.

Below is a short summary about the data.

	Beer	*Car*	*Steel*	*Gas*	*Electricity*	*Temperature*
Start	1956-01	1961-07	1956-01	1956-01	1956-01	1943-11
End	1992-03	1992-03	1992-03	1992-03	1992-03	1992-03
# of values	435	369	435	435	435	581
# of missing values	30	NA	NA	NA	NA	NA
Total	435	369	435	435	435	581

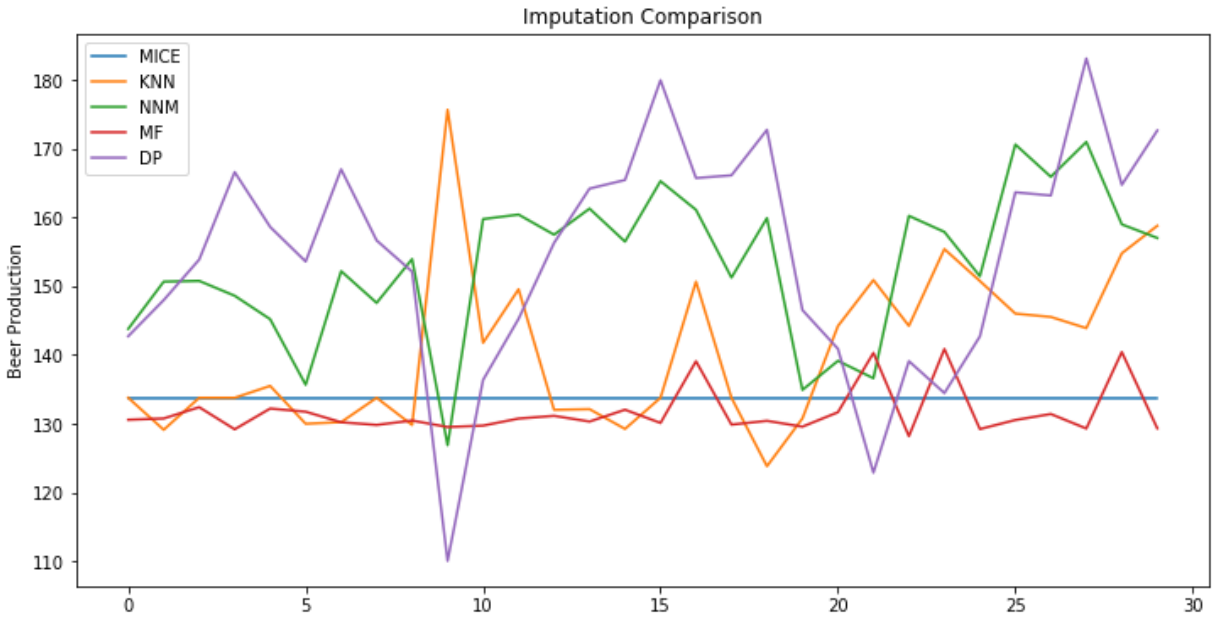
We merged the 6 categories using left join on time index. The final dataset merged is of shape 435×6.

Some basic imputation methods are as below:

- SoftImpute9: This method uses matrix completion via iterative soft-thresholded Singular Value Decomposition (SVD) to impute missing values.
- **KNN**: This method uses k-nearest neighbor to find similar samples and imputed unobserved data by weighted average of similar observations.
- Cubic Spline: This method uses cubic spline to interpolate each feature at different time steps.
- **MICE**: The Multiple Imputation by Chained Equations (MICE) method is widely used in practice, which uses chain equations to create multiple imputations for variables of different types.
- **MF**: Using matrix factorization (MF) to fill the missing items in the incomplete matrix by factorizing the matrix into two low-rank matrices.
- PCA: Imputing the missing values with the principal component analysis (PCA) model.
- MissForest: This is a non-parametric imputation method which uses random forests trained on the observed values to predict the missing values.
- **NNM**: Using Nuclear Norm Minimization (NNM) to implement Exact Matrix Completion via Convex Optimization by Emmanuel Candes and Benjamin Recht.

In Notebook for scenario 3, we have implemented the KNN, MICE, MF and NNM methods, as well as a deep learning method called ‘Datawig’ which our final imputation output is based on.

A plot of the forecast of the above five methods is as below:



Datawig trains machine learning models to impute missing values in tables. It has the advantages of fully making use of the information in the data and learns all parameters of the entire imputation pipeline automatically in an end-to-end fashion. Details on the underlying model can be found in Biessmann, Salinas et al. 2018.

By using Datawig, we imputed the missing value for the beer production as well as the car production value from 1956-01 to 1961-06 for the forecasting in scenario 4.

This deep learning model is evaluated using mean square error and r2 score. It has a MSE of 198.71 and r2 score of 0.84, which suggest a good regression on our data. One should notice that **we don't have prediction intervals for imputation** as the result of applying deep learning method.

2. Forecasting

Intuitively, we may think that the beer production should be related to the production of car, steel, the consumption of gas, electricity as well as the local temperature. Those are factors affecting the raw materials to produce beer, the transportation for sales and trade, the energy to supply boiling, fermentation and filtration in the brewery. The above conjecture is also verified by investigating into the CCF plot between Beer and other features as shown in **Fig 4.1**.

There are many ways for multivariate time series forecasting, like the extension of traditional univariate time series models. Except than that, deep learning techniques are prevailing as a result of computation power and cutting-edge algorithms. As for the forecasting of scenario 4, we decide to deploy the Long Short-Term Memory (LSTM) recurrent neural networks in favor of its flexibility and accuracy rather than traditional methods.

LSTM

Long Short-Term Memory networks are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, including multivariate time series modeling.

One can find more information about LSTM from wiki or Colah's blog.

REASON WE USE LSTM

- vs traditional time series models
  ARMAs and ARIMAs are particularly simple models which are essentially linear update models plus some noise thrown in. With nonlinear activation functions, neural networks are approximations to nonlinear functions. LSTMs are thus essentially a nonlinear timeseries model, where the nonlinearity is learned from the data.
- vs other machine learning algorithms
  LSTM recurrent neural networks are capable of automatically learning features from sequence data, support multiple-variate data, and can output a variable length sequences that can be used for multi-step forecasting.

i. Data Preprocessing

Below is a snapshot of the dataset we got from the imputation step.

Date	Beer	Car	Steel	Gas	Electricity	Temp
1956-01-01	93.2	12700	196.9	1709	1254	25.1
1956-02-01	96.0	12574	192.1	1646	1290	25.3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1992-02-01	133.0	21181	517.0	38690	12380	25.0
1992-03-01	163.0	25467	558.0	39792	13023	25.4

STATIOANRITY TRANSFORMATION

The Dickey Fuller test is one of the most popular statistical tests. It can be used to determine the presence of unit root in the series, and hence help us understand if the series is stationary or not. The null and alternate hypothesis of this test are:

- Null Hypothesis: The series has a unit root (value of a =1)
- Alternate Hypothesis: The series has no unit root.

If we fail to reject the null hypothesis, we can say that the series is non-stationary. This means that the series can be linear or difference stationary (we will understand more about difference stationary in the next section).

By **Fig 4.2, Fig 4.3** and Dickey Fuller test in the Notebook for scenario 4, some of our variables are non-stationary, namely 'Steel', 'Gas' and 'Electricity'. Then it is important for us to make all

the series stationary as the forecasting is based on the assumption that the model's statistics don't change over time.

Therefore, the transformation to stationarity is needed for those non-stationary variables. Typical technologies for stationarity transformation include differencing and log transformation. Details can be found in the Notebook for scenario 4.

DATA NORMALIZATION

For supervise learning, data needs to be normalized before feeding into the network, which is a critical part of data preprocessing for training. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges which is the case in scenario 4.

Because the different features in our scenario do not have similar ranges of values, the gradients may end up taking a long time and can oscillate back and forth and take a long time before it can finally find its way to the global/local minimum. To overcome the model learning problem, we normalize the data. We make sure that the different features take on similar ranges of values so that gradient descents can converge more quickly.

ii. Model Specification

Hyperparameter tuning is important before training. Some examples of hyperparameters include learning rate, loss function, optimizer, etc. A good set of hyperparameters will help improve the model performance. After exploring a range of possibilities manually, our architecture of LSTM network is decided as below:

The preprocessed data is then fed into the LSTM network with 50 neurons in the first hidden layer and 24 neurons in the output layer for predicting 24 steps ahead. The input shape will be 24 time-steps with 6 features. A batch size of 16 is used and a total number of 200 epochs is iterated.

We use the Mean Absolute Error (MAE) loss function and the efficient Adam version of stochastic gradient descent.

iii. Model Evaluation

we keep track of both the training and test loss during. At the end of the run both the training and test loss are plotted as **Fig 4.4.** We can observe a quick drop of both training and test losses indicating a proper hyperparameters without overfitting problem.

PREDICTION INTERVAL

The 95% **prediction interval is not applicable** for the LSTM network.

Instead we just use a univariate $GARCH(1,1)$ model and a vector AR model to plot a prediction interval for illustration purpose as shown in **Fig 4.5** and **Fig 4.6**.

IV. FIGURE

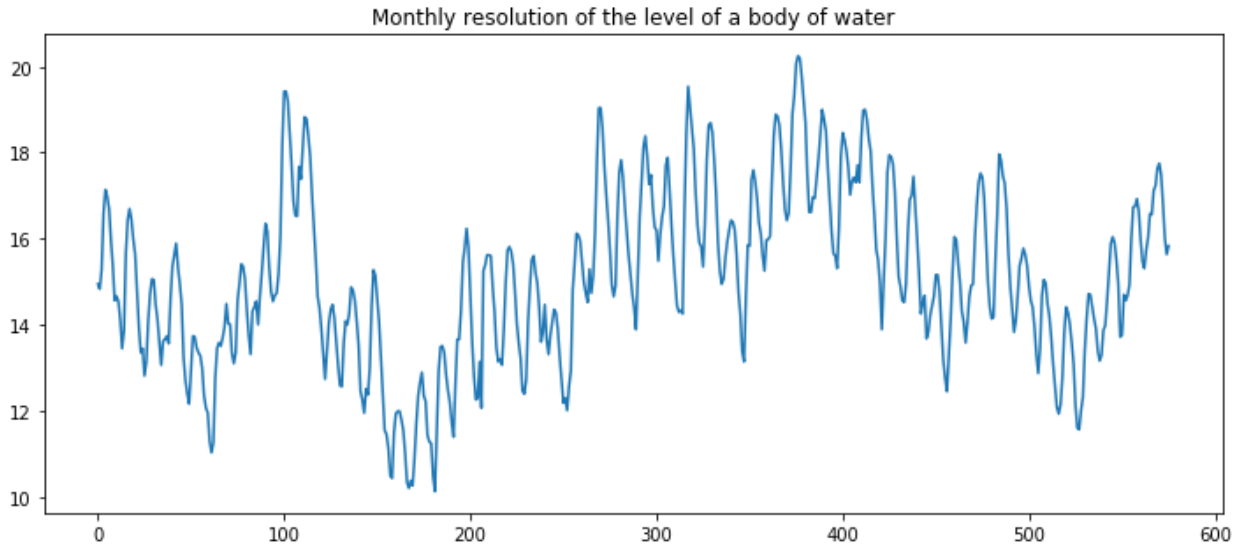


Fig. 1.1

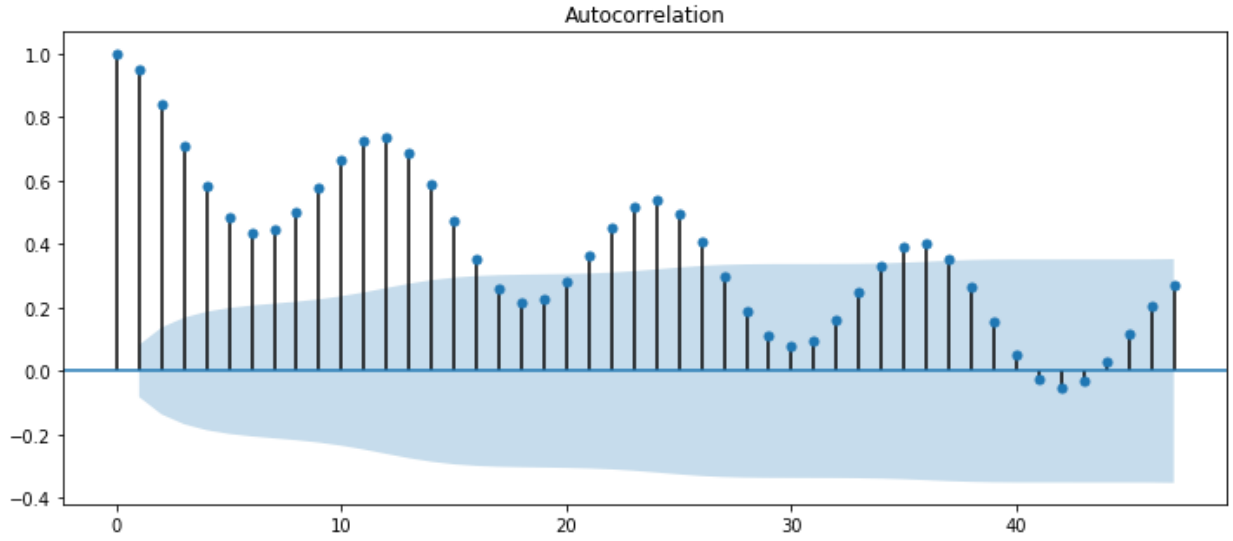


Fig. 1.2

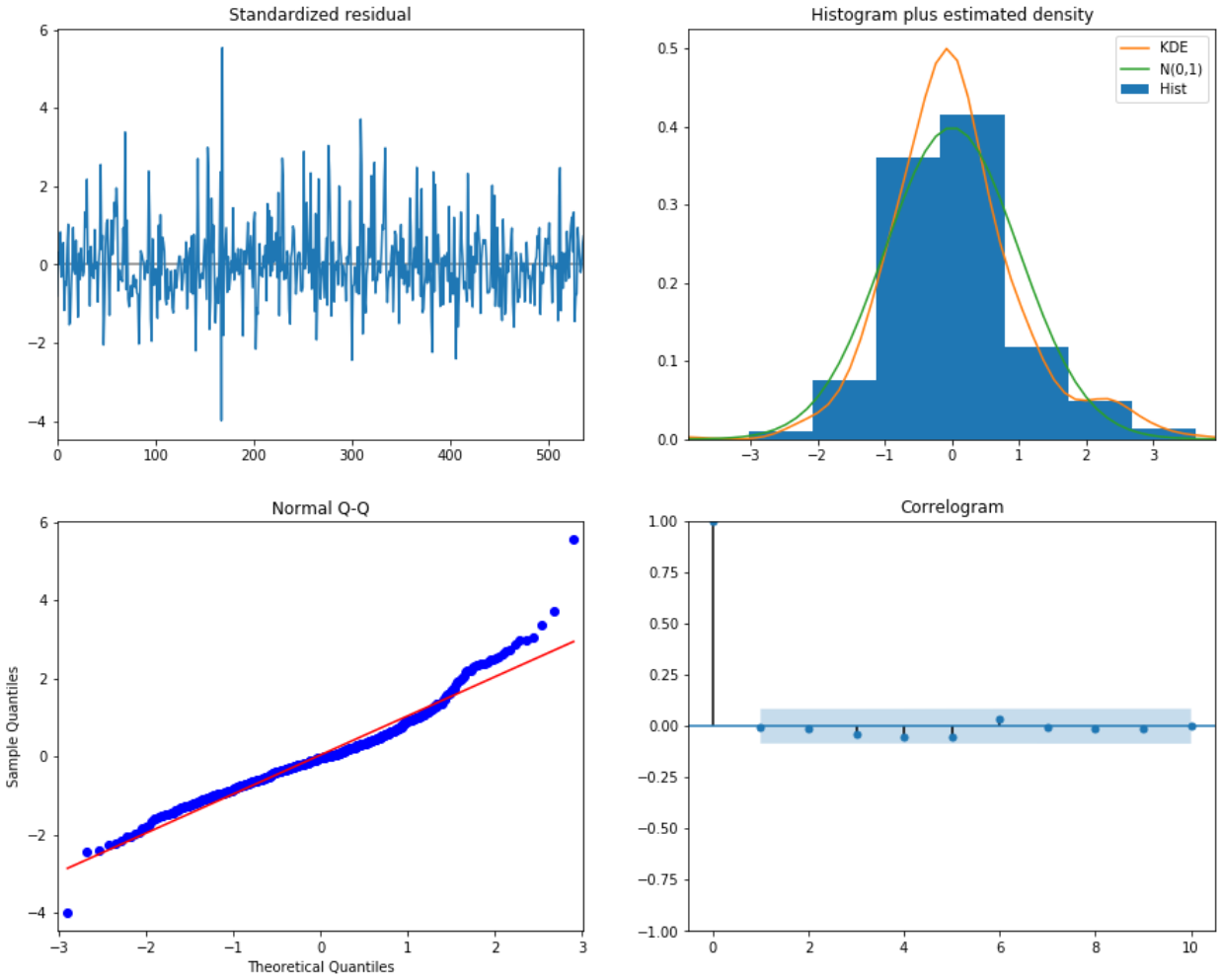

Standardized residual
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

Fig. 1.3

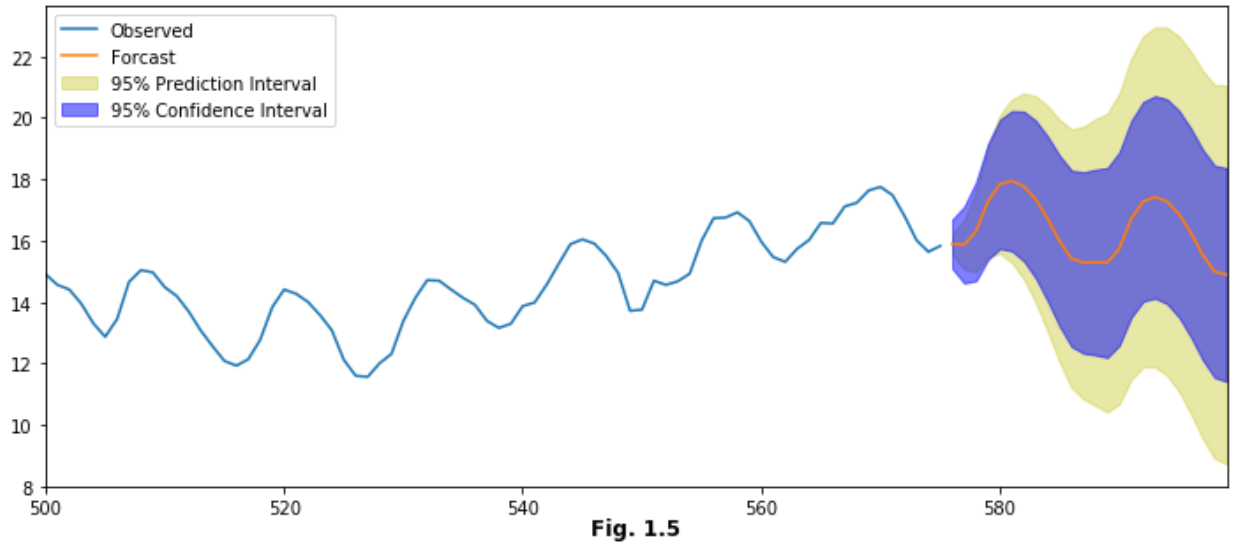

Observed
Forcast
95% Prediction Interval
95% Confidence Interval

Fig. 1.5

Fig. 2.1

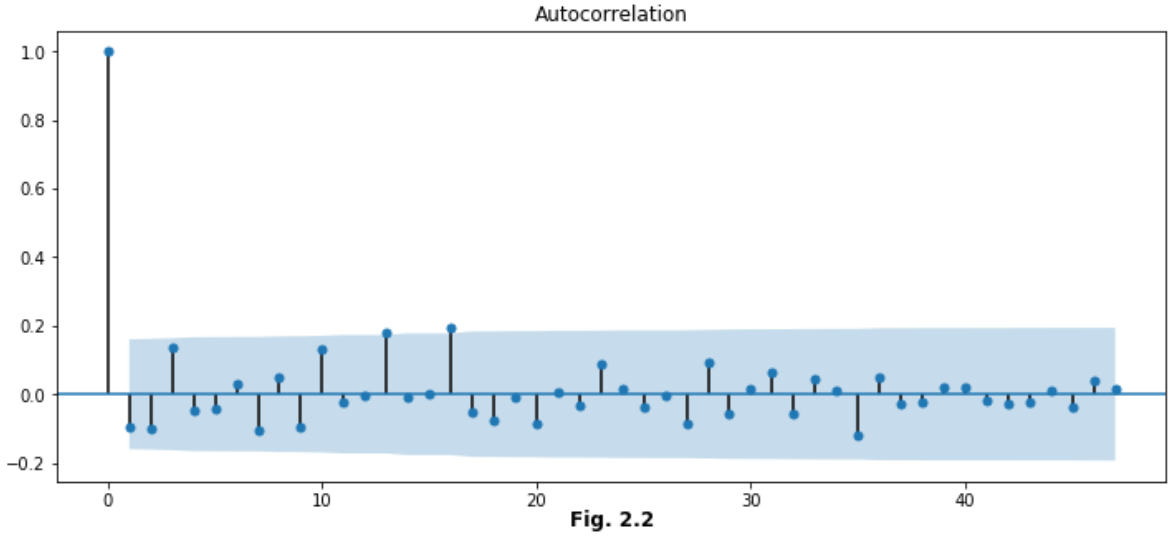


Fig. 2.2

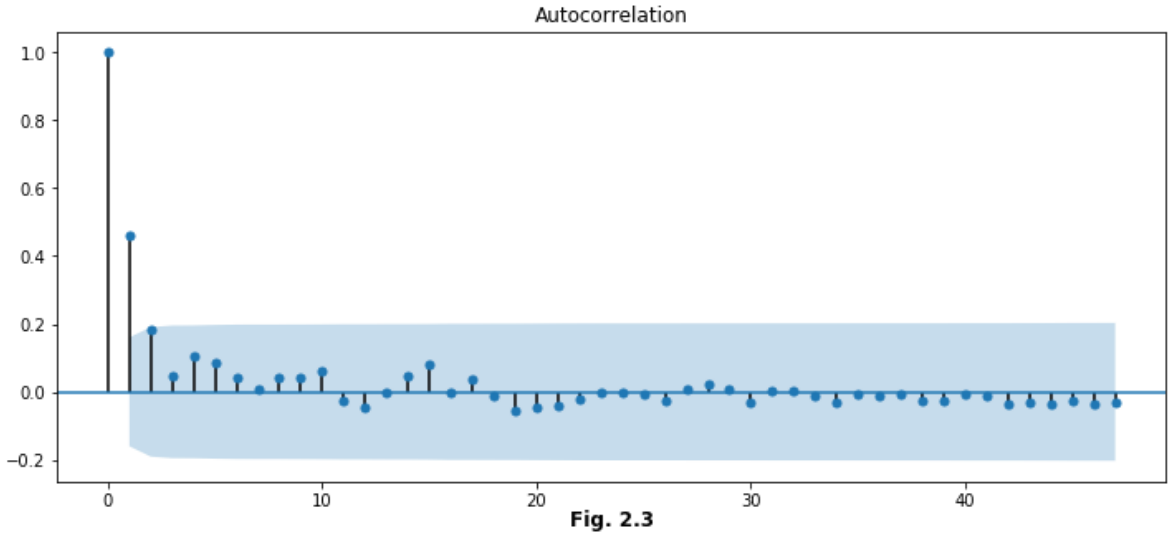


Fig. 2.3

Stock 17 Model Diagnostics – $GARCH(1,1)$

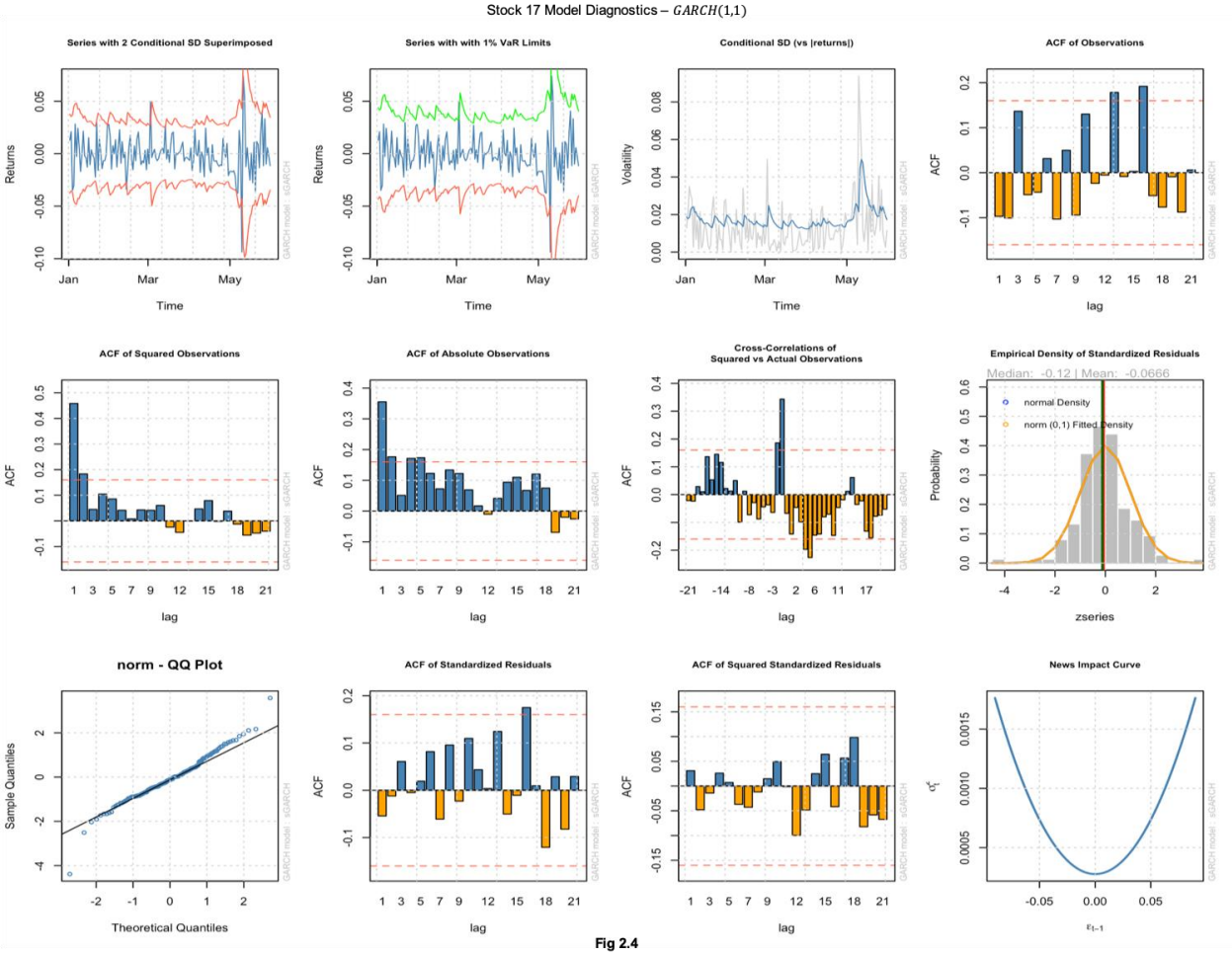


Fig 2.4

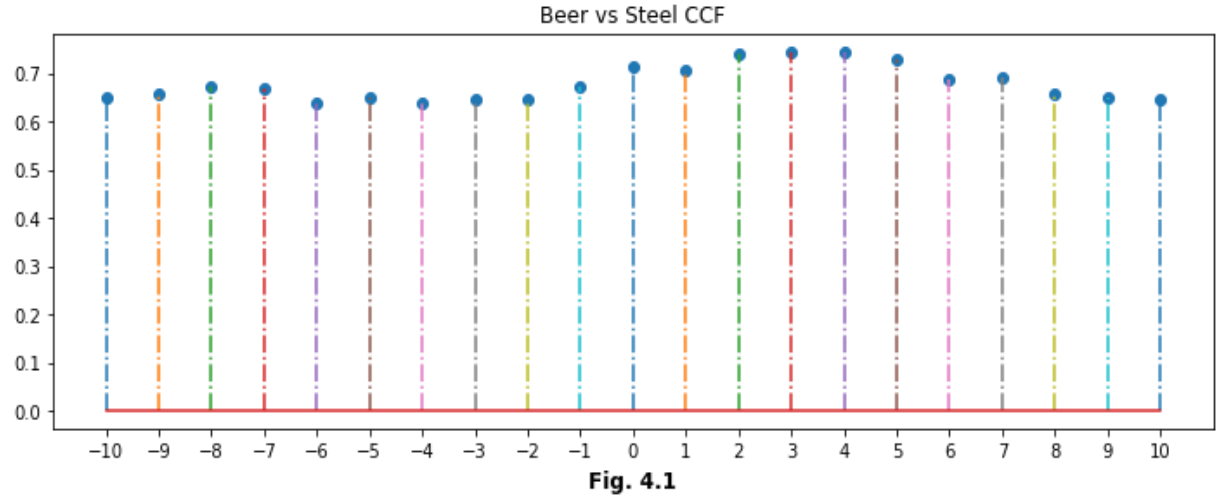


Fig. 4.1

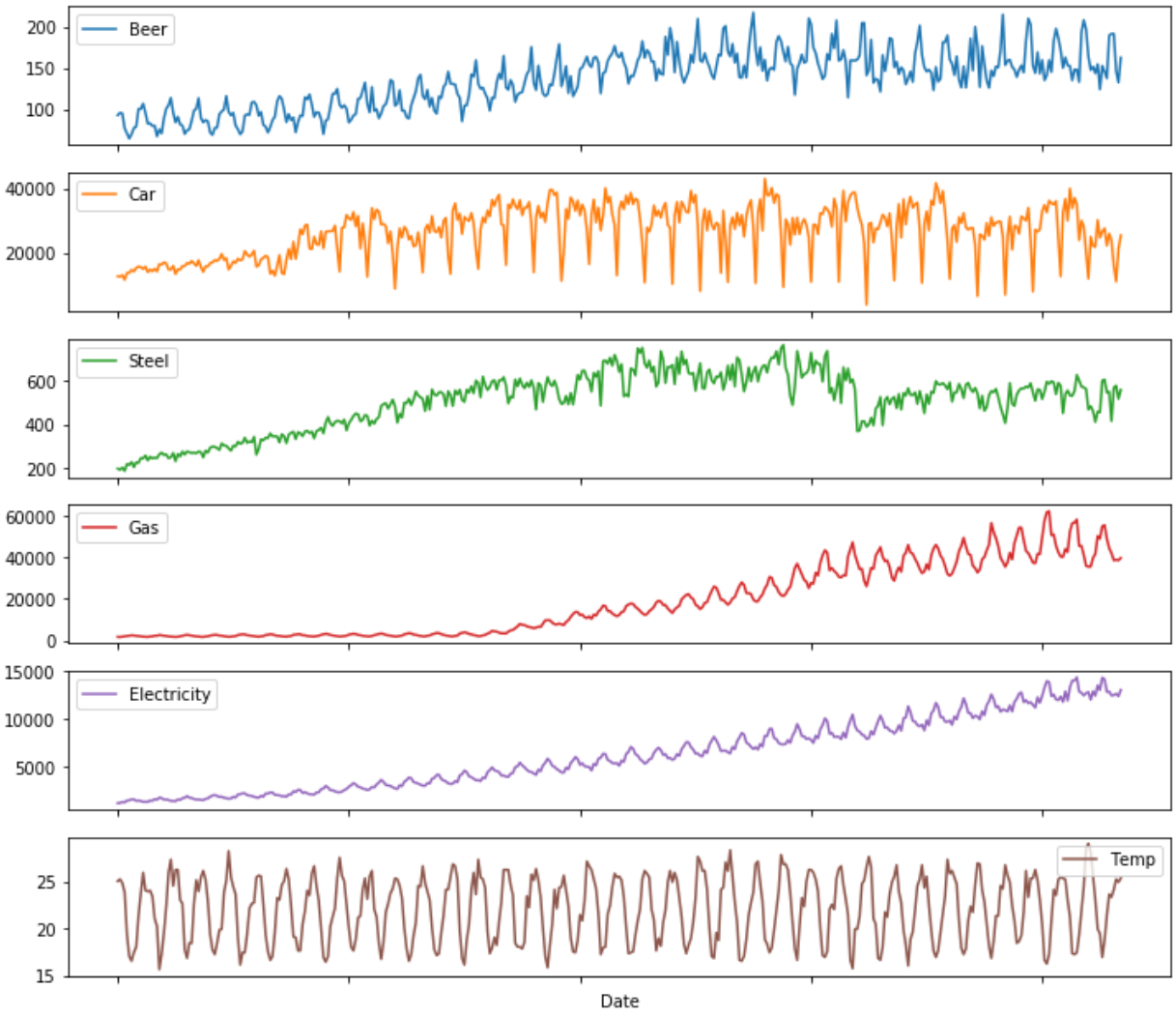

Beer
Car
Steel
Gas
Electricity
Temp
Date

Fig. 4.2

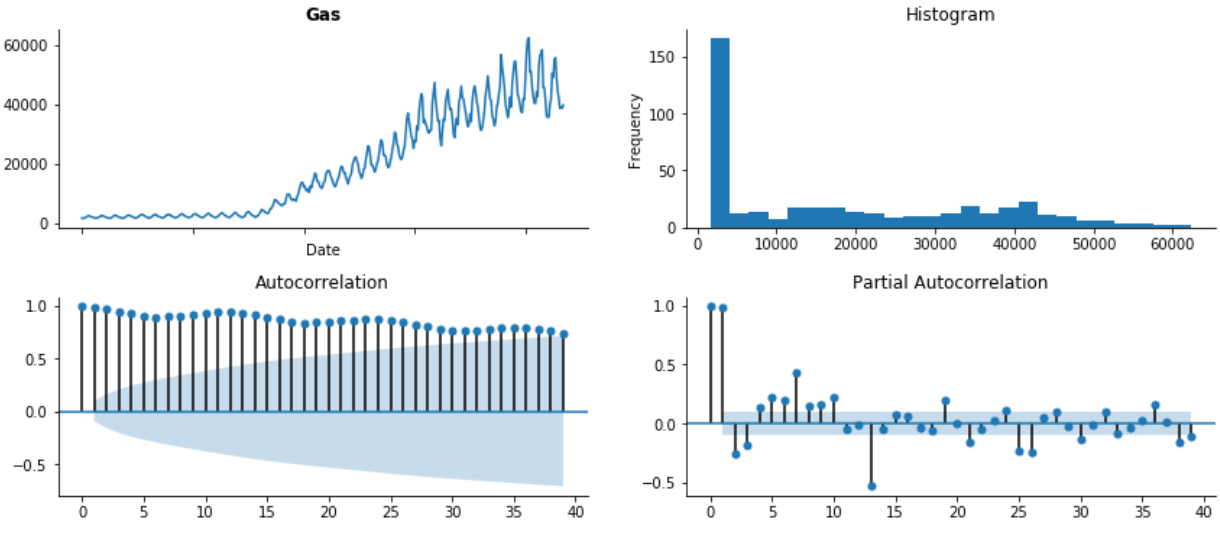

Gas
Date
Histogram
Frequency
Autocorrelation
Partial Autocorrelation

Fig. 4.3

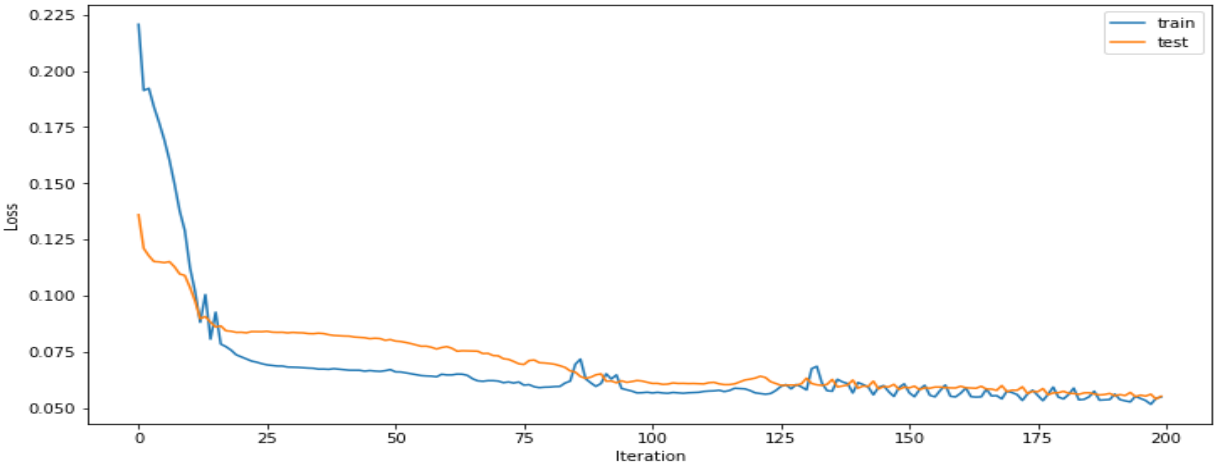


Fig. 4.4

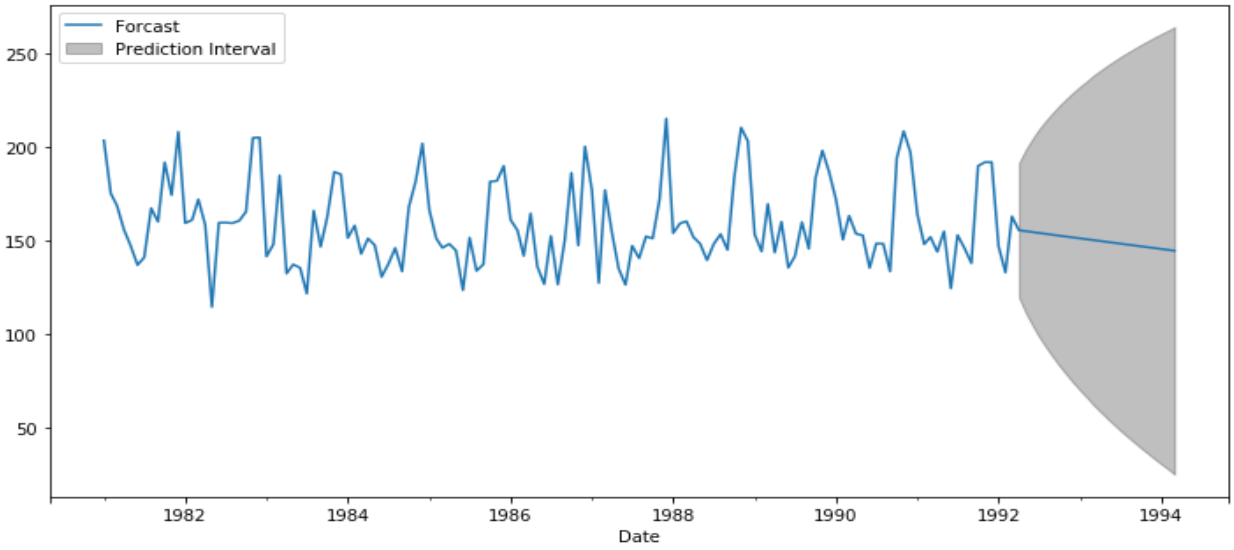


Fig. 4.5

Beer

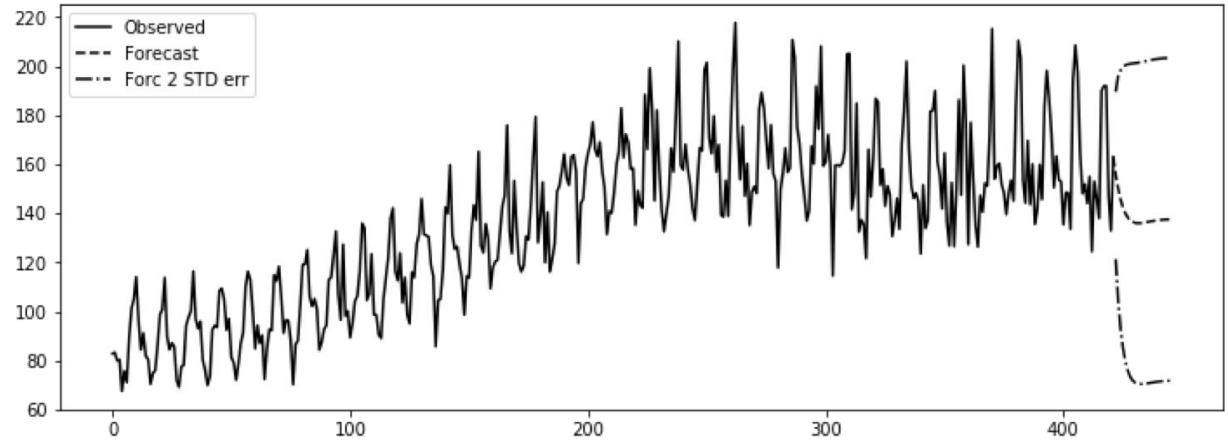


Fig 4.6

V. TABLE

Statespace Model Results

Dep. Variable:	Monthly Resolution	No. Observations:	576
Model:	SARIMAX(1, 0, 2)x(1, 1, 2, 12)	Log Likelihood	-291.171
Date:	Sun, 21 Apr 2019	AIC	596.342
Time:	23:53:03	BIC	626.344
Sample:	0	HQIC	608.079
	- 576		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9550	0.017	57.066	0.000	0.922	0.988
ma.L1	0.2364	0.035	6.725	0.000	0.167	0.305
ma.L2	0.1352	0.042	3.205	0.001	0.053	0.218
ar.S.L12	-0.6150	0.287	-2.146	0.032	-1.177	-0.053
ma.S.L12	-0.4665	0.297	-1.571	0.116	-1.048	0.115
ma.S.L24	-0.6171	0.310	-1.988	0.047	-1.225	-0.009
sigma2	0.1512	0.010	15.901	0.000	0.133	0.170

Ljung-Box (Q):	51.15	Jarque-Bera (JB):	218.79
Prob(Q):	0.11	Prob(JB):	0.00
Heteroskedasticity (H):	0.67	Skew:	0.79
Prob(H) (two-sided):	0.01	Kurtosis:	5.70

Table 1.1

```
*----------------------------------*
*          GARCH Model Fit         *
*----------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,1)
Mean Model      : ARFIMA(0,0,0)
Distribution    : norm
```

Optimal Parameters

	Estimate	Std. Error	t value	Pr(>|t|)
mu	0.001493	0.001064	1.4027	0.160696
omega	0.000071	0.000025	2.8144	0.004887
alpha1	0.654661	0.215102	3.0435	0.002338
beta1	0.254466	0.127638	1.9937	0.046190

Robust Standard Errors:

	Estimate	Std. Error	t value	Pr(>|t|)
mu	0.001493	0.000986	1.5133	0.130215
omega	0.000071	0.000020	3.6190	0.000296
alpha1	0.654661	0.354124	1.8487	0.064504
beta1	0.254466	0.128760	1.9763	0.048123

LogLikelihood : 417.2

Information Criteria

Akaike	-5.5092
Bayes	-5.4289
Shibata	-5.5106
Hannan-Quinn	-5.4766

Weighted LB Test on Standardized Residuals

	statistic	p-value
Lag[1]	0.2503	0.6169
Lag[2*(p+q)+(p+q)-1][2]	1.8531	0.2884
Lag[4*(p+q)+(p+q)-1][5]	3.8189	0.2775

d.o.f=0
H0 : No serial correlation

Weighted LB Test on Standardized Squared Residuals

	statistic	p-value
Lag[1]	0.2642	0.60727
Lag[2*(p+q)+(p+q)-1][5]	7.1950	0.04650
Lag[4*(p+q)+(p+q)-1][9]	8.9281	0.08409

d.o.f=2

Weighted ARCH LM Tests

	Statistic	Shape	Scale	P-Value
ARCH Lag[3]	0.2525	0.500	2.000	0.6153
ARCH Lag[5]	0.9880	1.440	1.667	0.7365
ARCH Lag[7]	1.3057	2.315	1.543	0.8594

Nyblom stability test

Joint Statistic: 1.001
Individual Statistics:

mu	0.17790
omega	0.09422
alpha1	0.42092
beta1	0.10533

Asymptotic Critical Values (10% 5% 1%)

Joint Statistic:	1.07	1.24	1.6
Individual Statistic:	0.35	0.47	0.75

Sign Bias Test

	t-value	prob	sig
Sign Bias	0.6900	0.4913	
Negative Sign Bias	0.9523	0.3425	
Positive Sign Bias	1.2794	0.2028	
Joint Effect	3.2931	0.3486	

Adjusted Pearson Goodness-of-Fit Test:

	group	statistic	p-value(g-1)
1	20	13.73	0.7990
2	30	16.80	0.9652
3	40	27.07	0.9255
4	50	28.00	0.9931

Table 2.1

```
*----------------------------------*
*          GARCH Model Fit         *
*----------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : eGARCH(1,1)
Mean Model      : ARFIMA(0,0,0)
Distribution    : std
```

Optimal Parameters

	Estimate	Std. Error	t value	Pr(>|t|)
mu	-0.001522	0.000001	-2344.4	0
omega	-0.250034	0.000124	-2014.3	0
alpha1	-0.189924	0.000113	-1680.4	0
beta1	0.967394	0.000408	2369.3	0
gamma1	-0.175202	0.000101	-1728.5	0
shape	7.549996	0.004866	1551.5	0

LogLikelihood : 414.9

Robust Standard Errors:

	Estimate	Std. Error	t value	Pr(>|t|)
mu	-0.001522	0.000044	-34.278	0
omega	-0.250034	0.005819	-42.969	0
alpha1	-0.189924	0.002264	-83.875	0
beta1	0.967394	0.025697	37.647	0
gamma1	-0.175202	0.004702	-37.260	0
shape	7.549996	0.035720	211.369	0

Information Criteria

Akaike	-5.4515
Bayes	-5.3310
Shibata	-5.4545
Hannan-Quinn	-5.4025

Weighted LB Test on Standardized Residuals

	statistic	p-value
Lag[1]	1.632	0.2014
Lag[2*(p+q)+(p+q)-1][2]	1.946	0.2719
Lag[4*(p+q)+(p+q)-1][5]	2.379	0.5319

d.o.f=0
H0 : No serial correlation

Weighted LB Test on Standardized Squared Residuals

	statistic	p-value
Lag[1]	1.422	0.2330
Lag[2*(p+q)+(p+q)-1][5]	2.399	0.5276
Lag[4*(p+q)+(p+q)-1][9]	2.864	0.7812

d.o.f=2

Weighted ARCH LM Tests

	Statistic	Shape	Scale	P-Value
ARCH Lag[3]	0.7148	0.500	2.000	0.3979
ARCH Lag[5]	1.3229	1.440	1.667	0.6401
ARCH Lag[7]	1.4517	2.315	1.543	0.8310

Nyblom stability test

Joint Statistic: 3.427
Individual Statistics:

mu	0.07197
omega	0.07216
alpha1	0.07541
beta1	0.04062
gamma1	0.07064
shape	0.07642

Asymptotic Critical Values (10% 5% 1%)

Joint Statistic:	1.49	1.68	2.12
Individual Statistic:	0.35	0.47	0.75

Sign Bias Test

	t-value	prob	sig
Sign Bias	1.3048	0.1940	
Negative Sign Bias	1.6545	0.1002	
Positive Sign Bias	0.3969	0.6921	
Joint Effect	3.9587	0.2660	

Adjusted Pearson Goodness-of-Fit Test:

	group	statistic	p-value(g-1)
1	20	34.0	0.01838
2	30	37.2	0.14118
3	40	54.8	0.04791
4	50	62.0	0.10057

Table 2.2

VI. REFERENCE

1. Biessmann, F., Salinas, D., Schelter, S., Schmidt, P., & Lange, D. (2018, October). Deep Learning for Missing Value Imputationin Tables with Non-Numerical Data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (pp. 2017-2025). ACM.
2. Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, *8*(1), 6085.
3. Long short-term memory. In Wikipedia. Retrieved April 22, 2019, from https://en.wikipedia.org/wiki/Long_short-term_memory
4. Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.
5. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
6. White, I. R., Royston, P., & Wood, A. M. (2011). Multiple imputation using chained equations: issues and guidance for practice. Statistics in medicine, 30(4), 377-399.
7. Yong, Z., Youwen, L., & Shixiong, X. (2009). An improved KNN text classification algorithm based on clustering. Journal of computers, 4(3), 230-237.
8. Engle, R. (2001). GARCH 101: The use of ARCH/GARCH models in applied econometrics. Journal of economic perspectives, 15(4), 157-168.
9. Ho, S. L., & Xie, M. (1998). The use of ARIMA models for reliability forecasting and analysis. Computers & industrial engineering, 35(1-2), 213-216.
10. Surgailis, D., & Viano, M. C. (2002). Long memory properties and covariance structure of the EGARCH model. ESAIM: Probability and Statistics, 6, 311-329.
11. Breusch, T. S., & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. Econometrica: Journal of the Econometric Society, 1287-1294.
12. Candès, E. J., & Recht, B. (2009). Exact matrix completion via convex optimization. Foundations of Computational mathematics, 9(6), 717.
13. Christie, A. A. (1982). The stochastic behavior of common stock variances: Value, leverage and interest rate effects. Journal of financial Economics, 10(4), 407-432.

VII. APPENDIX

One can find all the files of our 4 scenarios in my GitHub. Specific code for each scenario is listed below:

1. Notebook for scenario 1
2. Notebook for scenario 2
3. Notebook for scenario 3
4. Notebook for scenario 4