Alicia Ngoc Phan

BSTA 477 – Winter 2021

Tutorial 2 - Feb 7th, 2021

Benchmark methods
- Naive method (Corrected)
- Seasonal Naive method (Corrected)
- Drift method

Prediction interval

Ljung box test

Data used: Bike Sharing data set

Alicia Ngoc Phan

Benchmark methods

Naive method (Corrected)

Naive forecasts are values of the last observation.

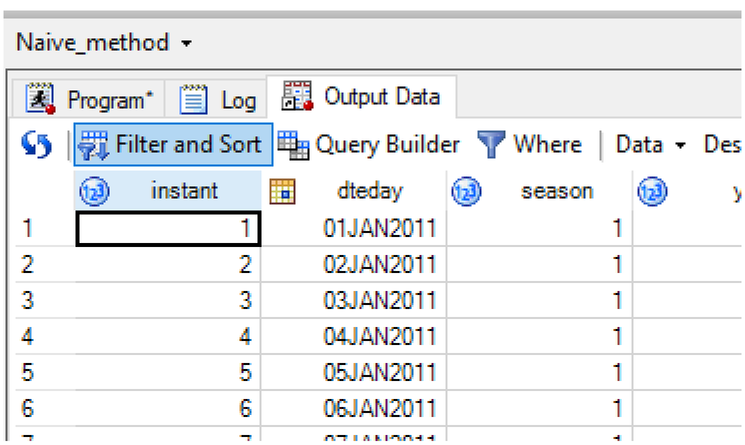
$$\hat{y}_{T+h|T} = y_T.$$

SAS implemented Naive forecast using SAS Lag() operation:

1. Different from other methods, the naive method is applied to the full dataset before partitioning the data into training and validation sets.

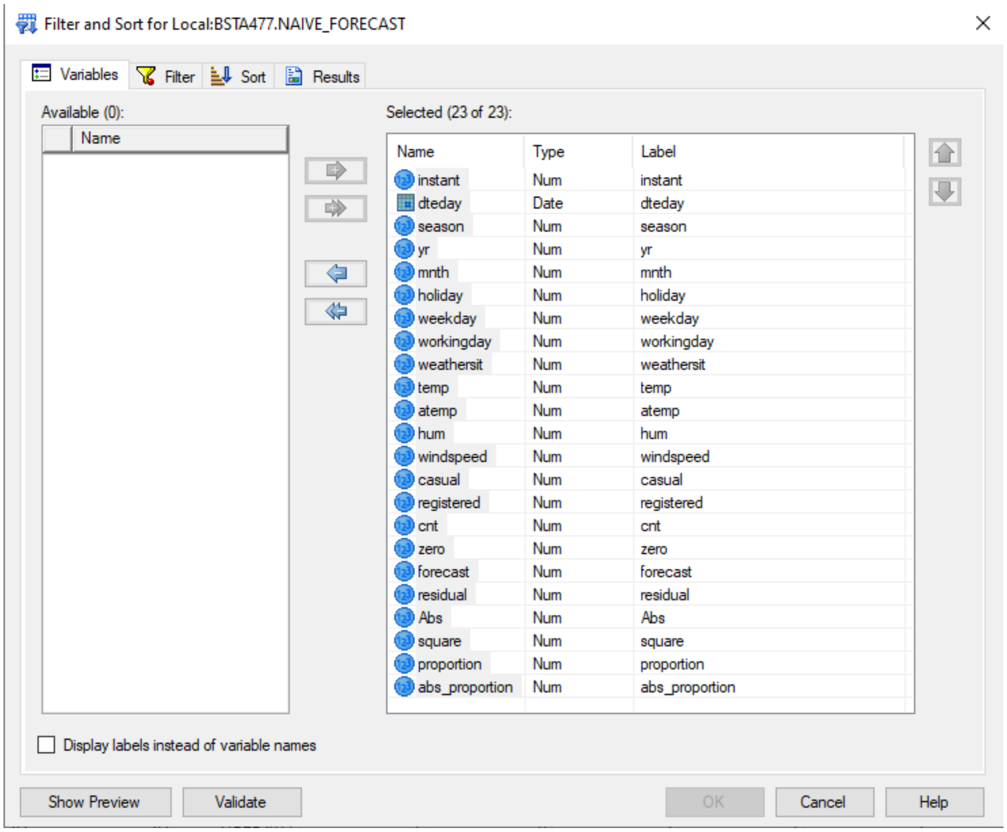
```
*Forecast cnt and calculate residuals;
data bsta477.naive_forecast;
set bsta477.bike_sharing_data_day;
forecast = lag(cnt);
residual = cnt - forecast;
Abs = abs(residual);
square = residual**2;
proportion = residual/cnt;
abs_proportion = abs/cnt;
run;
```

2. After obtaining all the forecasts, we partition the results into training and validation sets to calculate error terms.

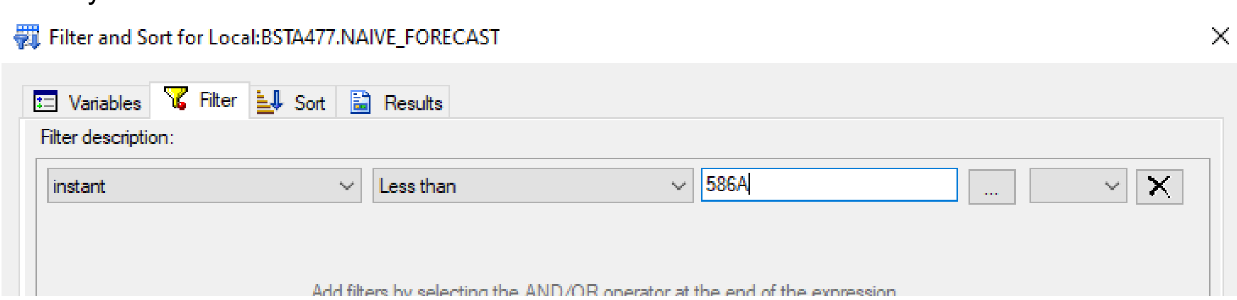


Choose all the variables needed:

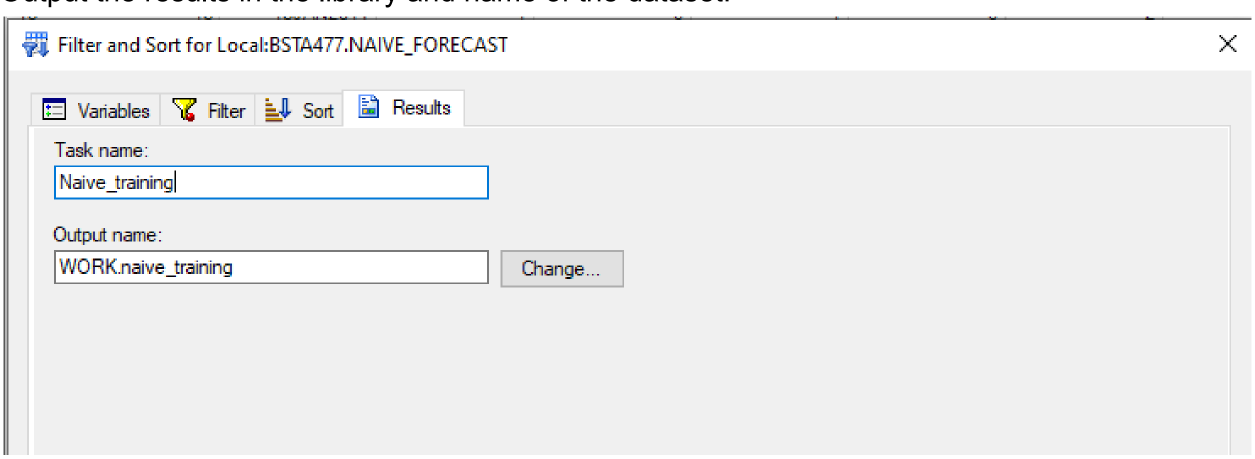
Alicia Ngoc Phan



Filter by time or index:



Output the results in the library and name of the dataset:



3. Do similar steps for the validation set. (Remember to change the data name and task name)

4. Calculating Error terms for evaluation (MAE, MSE, MAPE, MPE):

```
proc means data=work.naive_training;
var abs square proportion abs_proportion;
output out=work.naive_training_result (drop=_TYPE_ _FREQ_)
    Mean(Abs) = Mean_absolute_error
    Mean(square) = Mean_square_error
    Mean(abs_proportion) = Mean_absolute_percentage_error
    Mean(proportion) = Mean_percentage_error;
run;
```

```
proc means data=work.naive_validation;
var abs square proportion abs_proportion;
output out=work.naive_validation_result (drop=_TYPE_ _FREQ_)
    Mean(Abs) = Mean_absolute_error
    Mean(square) = Mean_square_error
    Mean(abs_proportion) = Mean_absolute_percentage_error
    Mean(proportion) = Mean_percentage_error;
run;
```

Note: To calculate RMSE, simply conduct square root on Mean square error.

Seasonal Naive method (Corrected)

Seasonal naive forecasts are values of the last observation for each season.

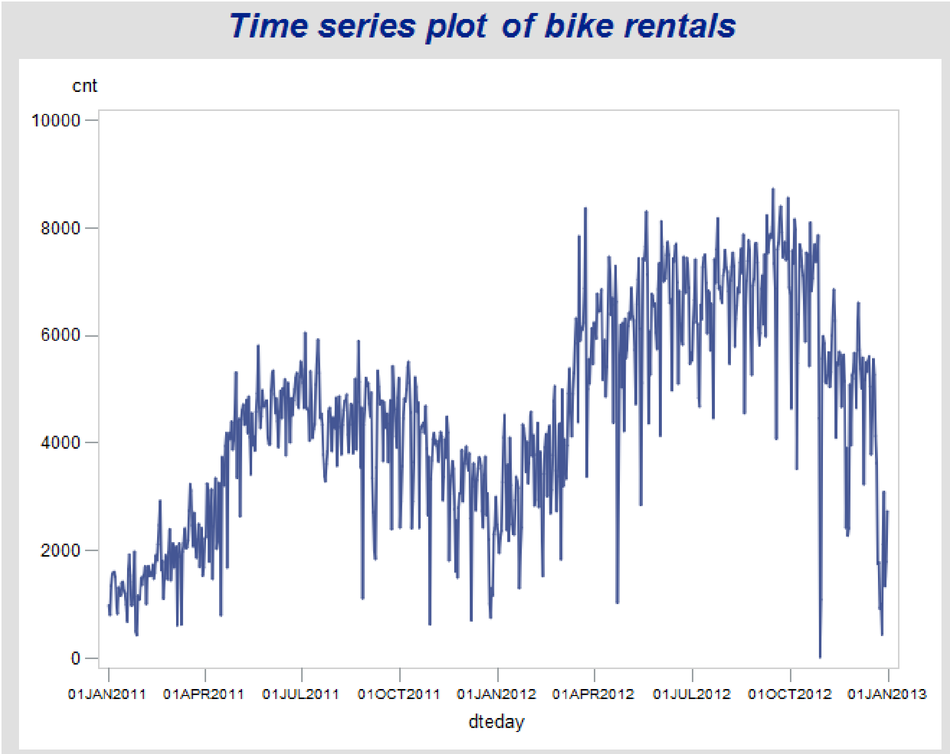
$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$$

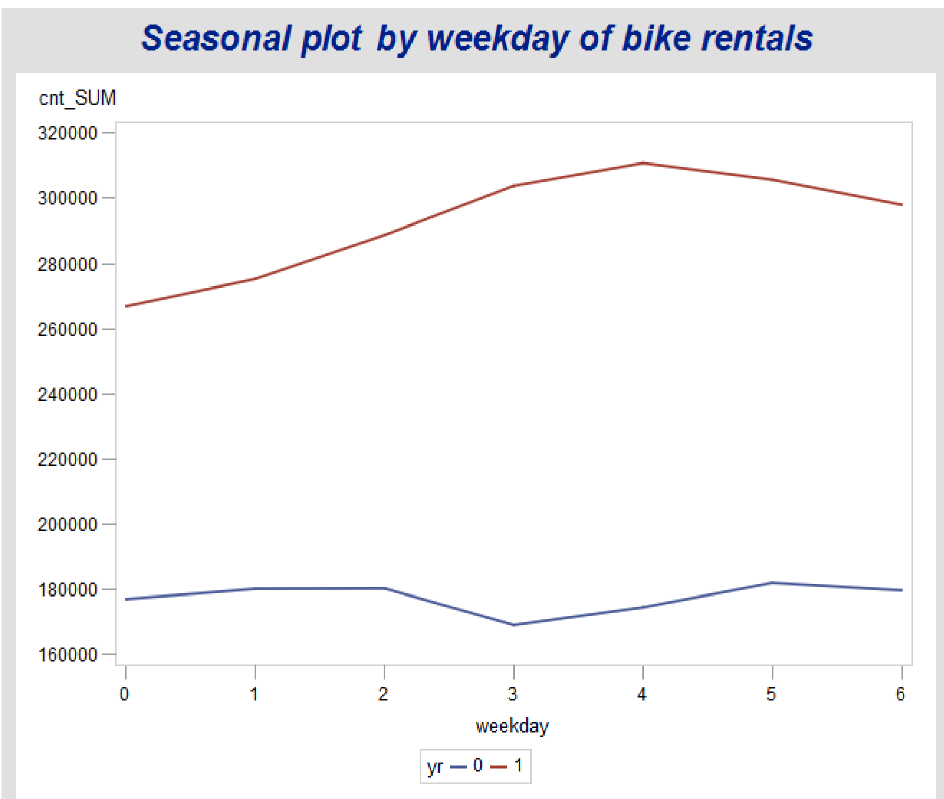
Before applying a seasonal naive method, use time series decomposition or seasonal plot to know the seasonal length of the data. (Try with different interval options). Based on the theory section, seasonality is used to describe patterns within a year, whereas, cycle is used to describe patterns in spread in multiple years.

Important note that you can apply the seasonal naive method if the data has seasonality. Otherwise, if the data has annual frequency or only cyclical patterns, then the seasonal naive method cannot be applied.

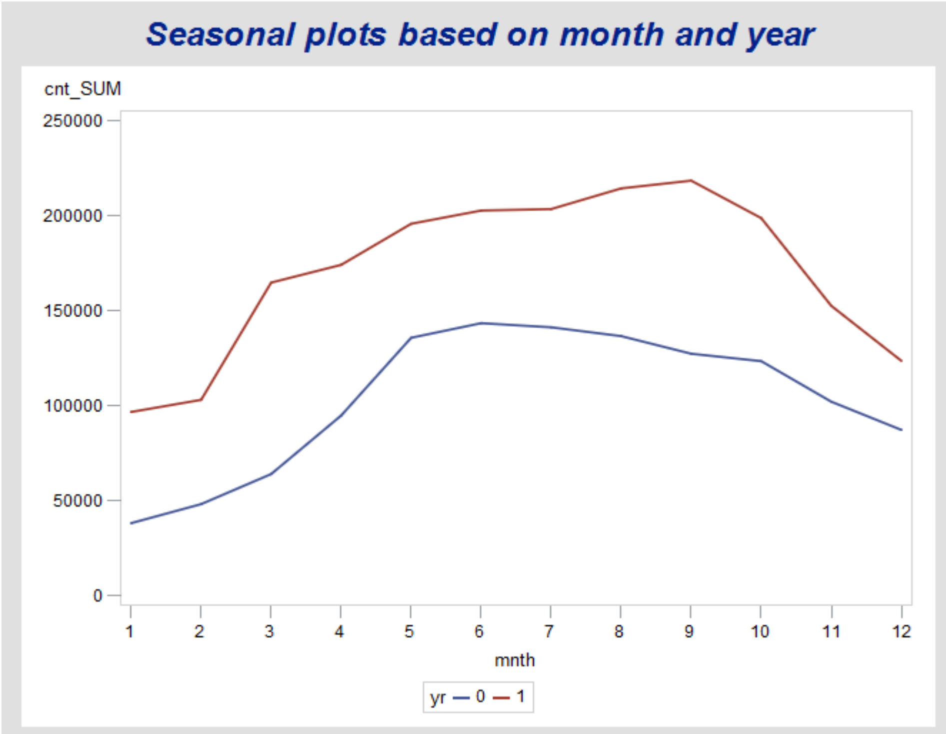
We explore the seasonality from the bike sharing time series and seasonal plots (monthly, weekly, etc.):

Alicia Ngoc Phan





Alicia Ngoc Phan



=> Based on the data from the Bike sharing dataset, we can see that there are some patterns such as :

- (Monthly seasonal plot) an increasing trend in January up until the peak around June or September, and decreasing trend after June or in September. From this we have a choice of m=12 (monthly) indicating the trend of each month or m=2 for this case, it's semi-annually. However, the dataset used only has observations of 2 years, thus only 4 observations if semi-annual seasonality, or 24 months/observations if monthly seasonality. Therefore, there would not be enough observations to forecast on.
- (Daily seasonal plot) An increasing trend from Monday to Wednesday (0-2) and from Thursday to Saturday (3-5), decreasing trend on Sunday. There is inconsistency in trends from Wednesday to Thursday comparing 2011 to 2012. With enough observations and patterns seen above, we can choose m=7 (days of week)

From the above analysis, we chose m=7 daily seasonality. Taking into consideration, our dataset has daily frequency, we don't have to aggregate the data

1. Similar to naive method, SAS implemented seasonal naive forecast before the data partition task. Seasonal naive method applies SAS lag operation: lagm() with m: seasonality on the **full dataset**.

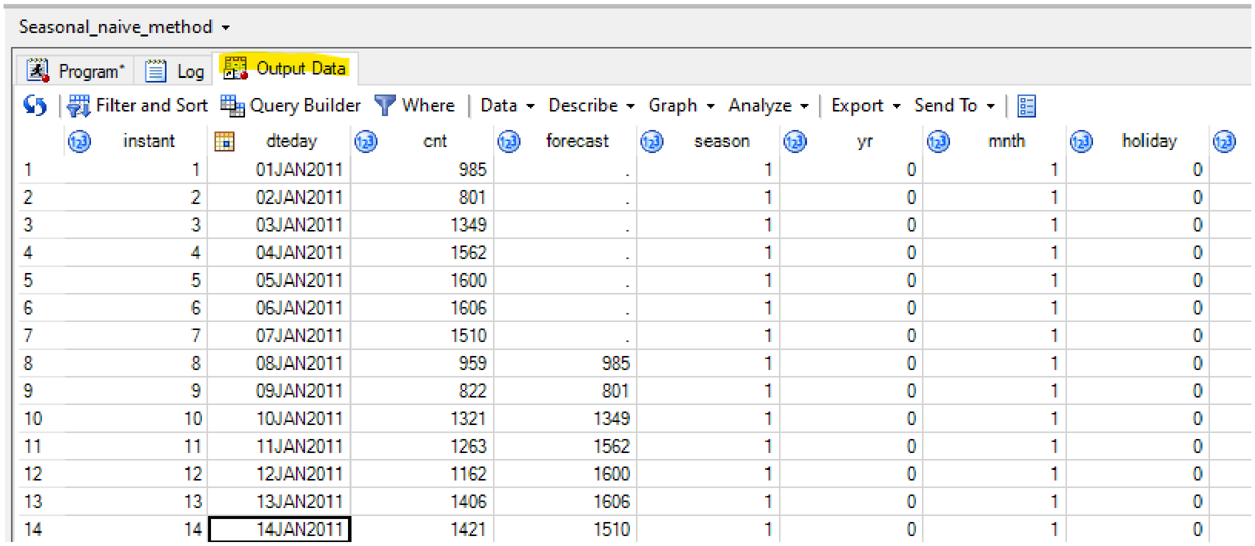
```
ods graphics on;
data work.seasonal_naive_forecasts;
set bsta477.bike_sharing_data_day;
forecast = lag7(cnt);
residual = cnt - forecast;
Abs = abs(residual);
square = residual**2;
proportion = residual/cnt;
abs_proportion = abs/cnt;
run;
```

2. We can see that all the days have forecasts except the first 7 days.

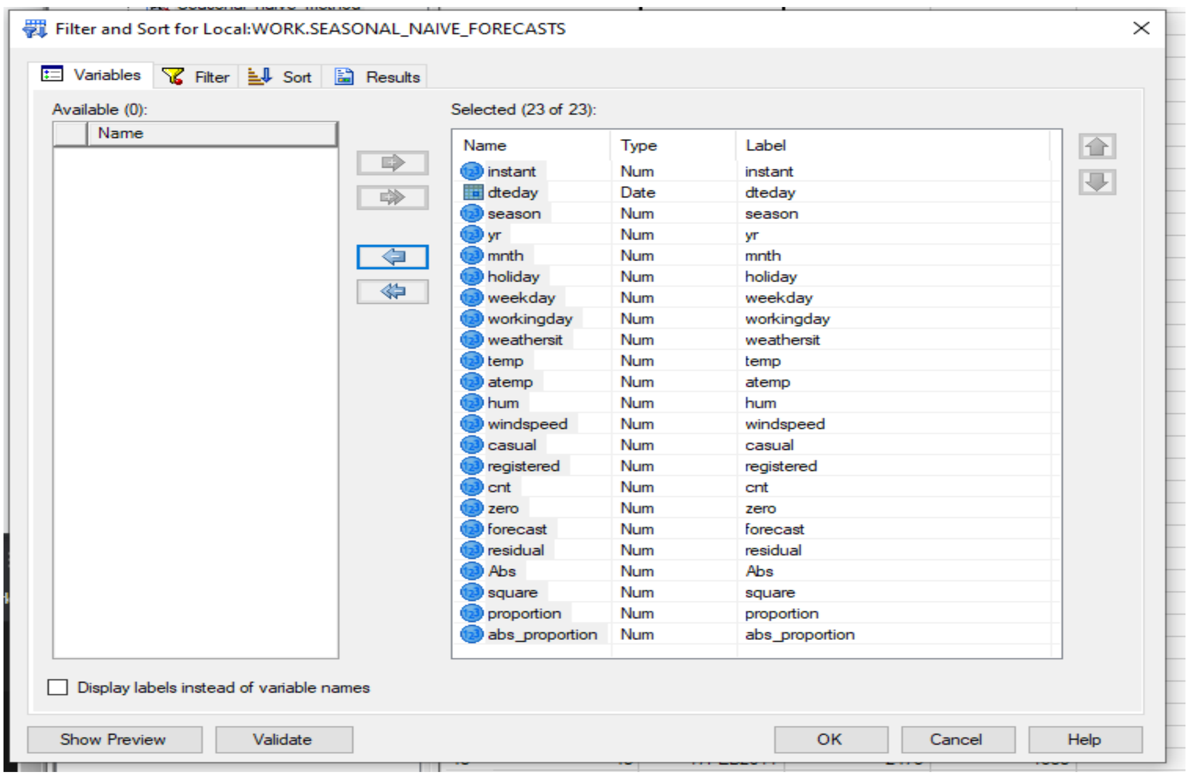
dteday	cnt	forecast
01JAN2011	985	.
02JAN2011	801	.
03JAN2011	1349	.
04JAN2011	1562	.
05JAN2011	1600	.
06JAN2011	1606	.
07JAN2011	1510	.
08JAN2011	959	985
09JAN2011	822	801
10JAN2011	1321	1349
11JAN2011	1263	1562
12JAN2011	1162	1600
13JAN2011	1406	1606
14JAN2011	1421	1510
15JAN2011	1248	959
16JAN2011	1204	822
17JAN2011	1000	1321
18JAN2011	683	1263
19JAN2011	1650	1162

3. When we obtain all the forecasts, we will **partition the data into training and validation** sets to calculate error terms. Click on the output data tab in the program you just ran to conduct the data partition.

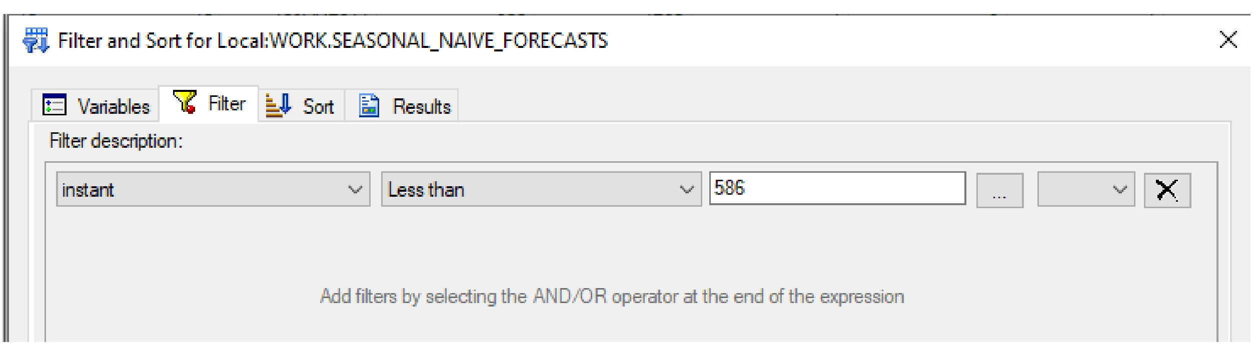
Alicia Ngoc Phan



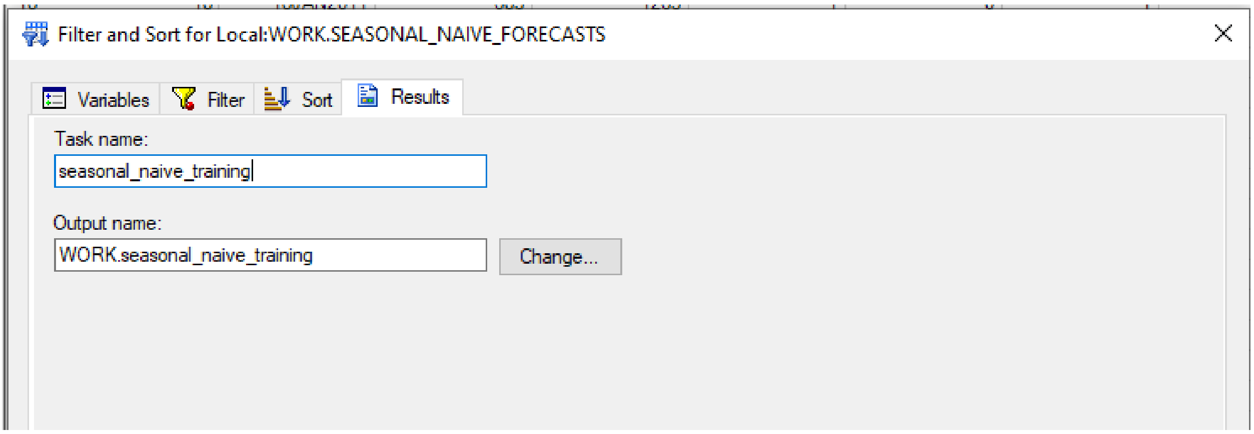
	instant	dteday	cnt	forecast	season	yr	mnth	holiday
1	1	01JAN2011	985	.	1	0	1	0
2	2	02JAN2011	801	.	1	0	1	0
3	3	03JAN2011	1349	.	1	0	1	0
4	4	04JAN2011	1562	.	1	0	1	0
5	5	05JAN2011	1600	.	1	0	1	0
6	6	06JAN2011	1606	.	1	0	1	0
7	7	07JAN2011	1510	.	1	0	1	0
8	8	08JAN2011	959	985	1	0	1	0
9	9	09JAN2011	822	801	1	0	1	0
10	10	10JAN2011	1321	1349	1	0	1	0
11	11	11JAN2011	1263	1562	1	0	1	0
12	12	12JAN2011	1162	1600	1	0	1	0
13	13	13JAN2011	1406	1606	1	0	1	0
14	14	14JAN2011	1421	1510	1	0	1	0



Name	Type	Label
instant	Num	instant
dteday	Date	dteday
season	Num	season
yr	Num	yr
mnth	Num	mnth
holiday	Num	holiday
weekday	Num	weekday
workingday	Num	workingday
weathersit	Num	weathersit
temp	Num	temp
atemp	Num	atemp
hum	Num	hum
windspeed	Num	windspeed
casual	Num	casual
registered	Num	registered
cnt	Num	cnt
zero	Num	zero
forecast	Num	forecast
residual	Num	residual
Abs	Num	Abs
square	Num	square
proportion	Num	proportion
abs_proportion	Num	abs_proportion



Alicia Ngoc Phan

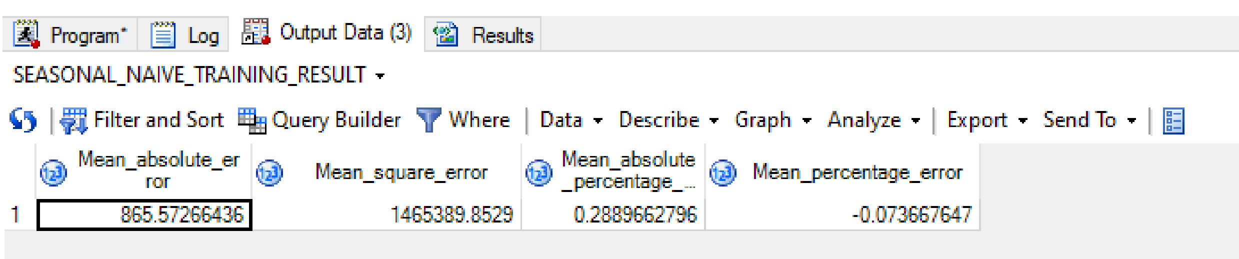


And same for validation set partition operations (changes made in the filter tab of the correct observations and output results name).

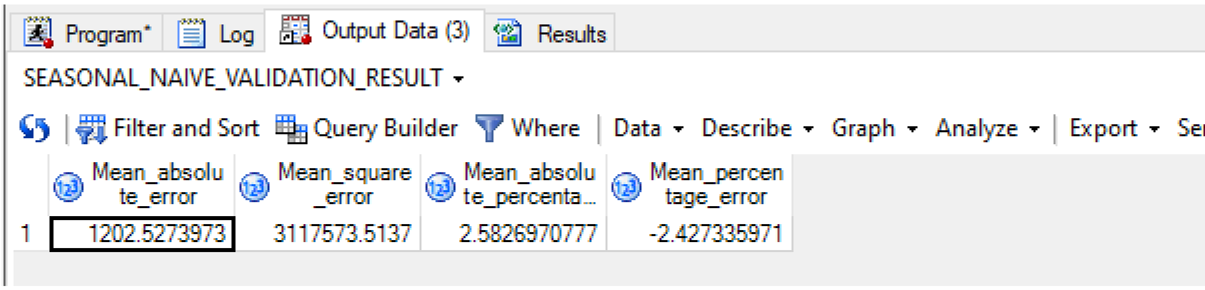
4. Then, we calculate the error terms for the training and validation set.

```
proc means data=work.seasonal_naive_training;
var abs square proportion abs_proportion;
output out=work.seasonal_naive_training_result (drop=_TYPE_ _FREQ_)
    Mean(Abs) = Mean_absolute_error
    Mean(square) = Mean_square_error
    Mean(abs_proportion) = Mean_absolute_percentage_error
    Mean(proportion) = Mean_percentage_error;
run;

proc means data=work.seasonal_naive_validation;
var abs square proportion abs_proportion;
output out=work.seasonal_naive_validation_result (drop=_TYPE_ _FREQ_)
    Mean(Abs) = Mean_absolute_error
    Mean(square) = Mean_square_error
    Mean(abs_proportion) = Mean_absolute_percentage_error
    Mean(proportion) = Mean_percentage_error;
run;
```



Alicia Ngoc Phan



	Mean_absolute_error	Mean_square_error	Mean_absolute_percenta...	Mean_percentage_error
1	1202.5273973	3117573.5137	2.5826970777	-2.427335971

Drift method

Drift method forecasts are based on the last observation and the average change of the time series.

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right).$$

=> Forecast 1 step ahead, 5th period
Y(4+1) = Y4 + 1*(Y4 - Y1)/(NoObs - 1)

=> Forecast 2 steps ahead, 5th period
Y(3+2) = y3 + 2*(Y3 -Y1)/(NoObs -1)
Always remember to partition data into training and validation set

Example code: Forecast 1 step ahead

- Create **instant** variable in your data set **before data partitioning**. Instant variable is the index variable of observations. Code: instant = _N_
- Partition data into training and validation sets.

Training set

```
Data bsta477.drift_forecast_training;
 set bsta477.training_set_bike_sharing;
 forecast = lag1(cnt) + 1*((lag1(cnt) - 985)/(instant-2));
 residual = cnt - forecast;
 Abs = abs(residual);
 square = residual**2;
 proportion = residual/cnt;
 abs_proportion = abs/cnt;
 run;
```

Note:

- 985 is Y1 of the variable cnt of the data set. And the instant has to subtract by 2 in order to calculate the forecast from previous values.

```
proc means data=bsta477.drift_forecast_training mean;
 var abs square proportion abs_proportion;
 output out=bsta477.drift_training_eva (drop=_TYPE_ _FREQ_)
     Mean(Abs) = Mean_absolute_error
     Mean(square) = Mean_square_error
     Mean(abs_proportion) = Mean_absolute_percentage_error
     Mean(proportion) = Mean_percentage_error;
 run;
```

Note: Applying drift method to validation set uses the same concept. However, the fixed forecast point would be from the training set. Therefore, Yt = last observation of the training set, Y1 = first observation of the training set, h = forecasting period in validation set.

Validation set

```
data bsta477.drift_forecast_validation;
 set bsta477.validation_set_bike_sharing;
 trend = _N_;
 forecast = 7273 + trend*((7273-985)/(instant-2));
 residual = cnt - forecast;
 Abs = abs(residual);
 square = residual**2;
 proportion = residual/cnt;
 abs_proportion = abs/cnt;
 run;
```

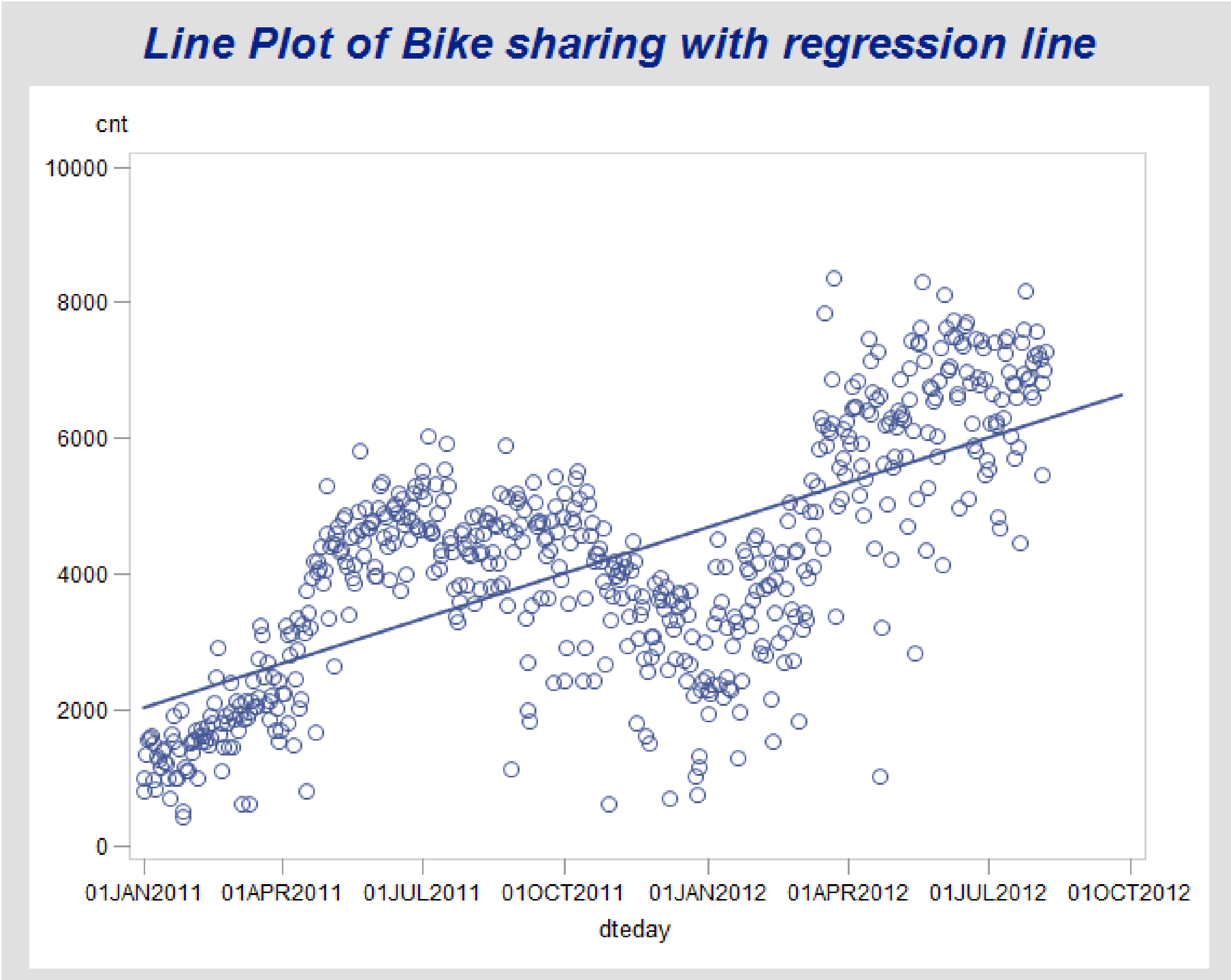
Note: 7273 is the last observation of cnt variable in the training set, 985 is the first observation of the training set, trend is the ordered forecasting period.

```
proc means data=bsta477.drift_forecast_validation mean;
 var abs square proportion abs_proportion;
 output out=bsta477.drift_validation_eva (drop=_TYPE_ _FREQ_)
     Mean(Abs) = Mean_absolute_error
     Mean(square) = Mean_square_error
     Mean(abs_proportion) = Mean_absolute_percentage_error
     Mean(proportion) = Mean_percentage_error;
 run;
```

Prediction interval

The scatter plot of Bike sharing data set with regression line. (Y hat = Bo + B1*X)

Alicia Ngoc Phan



Using Y hat = Bo + B1*X to predict Y, this Y value is the point forecast which is the mean of all possible values. Assume that the forecast errors are normally distributed, prediction intervals present the interval for the possible values based on the % prediction interval.

Formulate prediction interval:

$$\hat{y}_{T+h|T} \pm c\hat{\sigma}_h$$

Ex: Formulate 95% prediction interval

$$\hat{y}_{T+h|T} \pm 1.96\hat{\sigma}_h,$$

Multipliers to be used for prediction intervals

Alicia Ngoc Phan

Percentage	Multiplier
50	0.67
55	0.76
60	0.84
65	0.93
70	1.04
75	1.15
80	1.28
85	1.44
90	1.64
95	1.96
96	2.05
97	2.17
98	2.33
99	2.58

=> Formulate 95% prediction interval using RMSE (Root Mean Squared Error):

Y hat +/- 1.96*RMSE

<u>Example: Calculate the prediction interval for bike rentals for 2Jan2011:</u>

	dteday	cnt	instant	forecast	residual	Abs	square	proportion	abs_proportion
1	01JAN2011	985	1	.	.	.	.	.	.
2	02JAN2011	801	2	985	-184	184	33856	-0.229712859	0.2297128589

Y hat = 985

Calculate RMSE:

```
data bsta477.naive_training_evaluation;
 set bsta477.naive_training_evaluation;
 RMSE = sqrt(Mean_Square_Error);
 run;
```

	Mean_absolute_error	Mean_square_error	Mean_absolute_percenta...	Mean_percentage_error	RMSE
1	691.68835616	1002741.8904	0.2348341596	-0.063429558	1001.3700067

=> Prediction interval = 985 +/- 1011.37*1.96 => [0, 2,967.28]

Alicia Ngoc Phan

Ljung box test

Reference:

- A more flexible Ljung Box Test in SAS
- Testing adequacy of ARIMA models using weighted Portmanteau test on the Residual autocorrelations.
- Chi-square table

Ho: The residual is white noise series (The model does not show lack of fit)
Ha: The residual has autocorrelation. (The model shows lack of fit)

Decision matrix:

- If Q > Chi-square OR p < significant level => Reject Ho, the model shows lack of fit
- If Q < Chi-square OR p > significant level => Accept Ho, the model does not show lack of fit.

To find Chi-square critical (**alpha**, degree of freedom)

Ex: Test the residuals of drift training method, taking significant level = alpha = 0.05

```
ods graphics on;
proc arima data=bsta477.drift_forecast_training;
identify var=residual whitenoise=ignoremiss;
run;
```

Result:

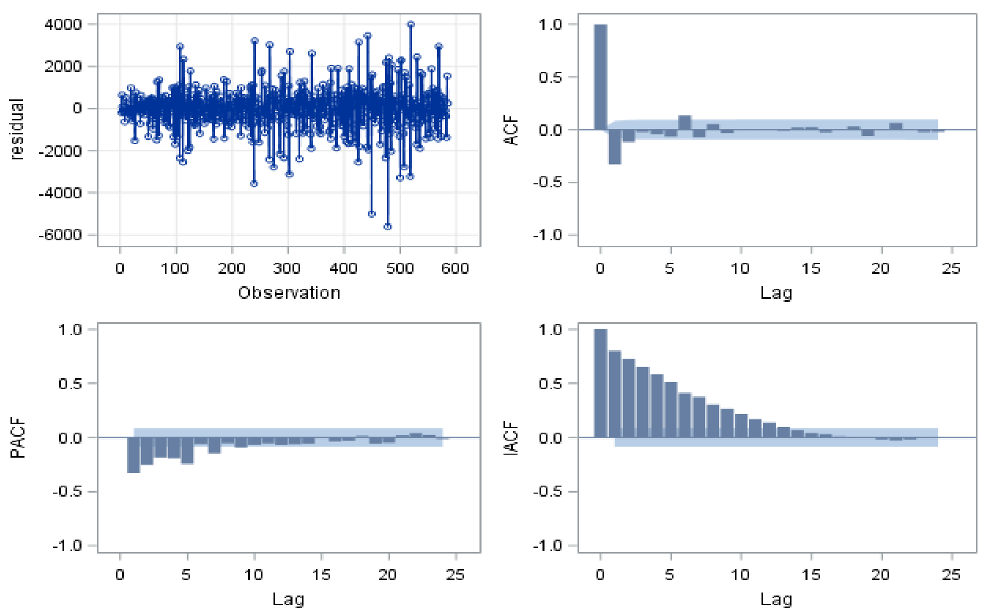
Alicia Ngoc Phan

The ARIMA Procedure

Name of Variable = residual	
Mean of Working Series	-2.6666
Standard Deviation	1003.582
Number of Observations	584

Autocorrelation Check for White Noise									
To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	86.00	6	<.0001	-0.328	-0.117	-0.024	-0.045	-0.064	0.136
12	91.27	12	<.0001	-0.072	0.050	-0.033	0.001	-0.003	0.001
18	92.90	18	<.0001	-0.012	0.020	0.023	-0.027	-0.004	0.030
24	98.15	24	<.0001	-0.058	0.008	0.064	0.001	-0.025	-0.023

Trend and Correlation Analysis for residual



Page Break

=> Chi-square critical (0.05, DF)

1. Check Chi-square value for lag 6 : Chi-Square critical = 1.635
   => Chi-Square > Chi-Square critical
   => Therefore, we conclude that there is not enough evidence to accept the null hypothesis, and conclude that the model shows lack of fit. (Beside p-value, you can also see there is high autocorrelation in the residuals in the ACF and PACF plots of the residuals above).
2. Check p-value and significance level. We can see that p for all lags < 0.05. Therefore, there is not enough evidence to accept the null hypothesis.