Retail Forecasting Project

ETC3550

Chelaka Paranahewa

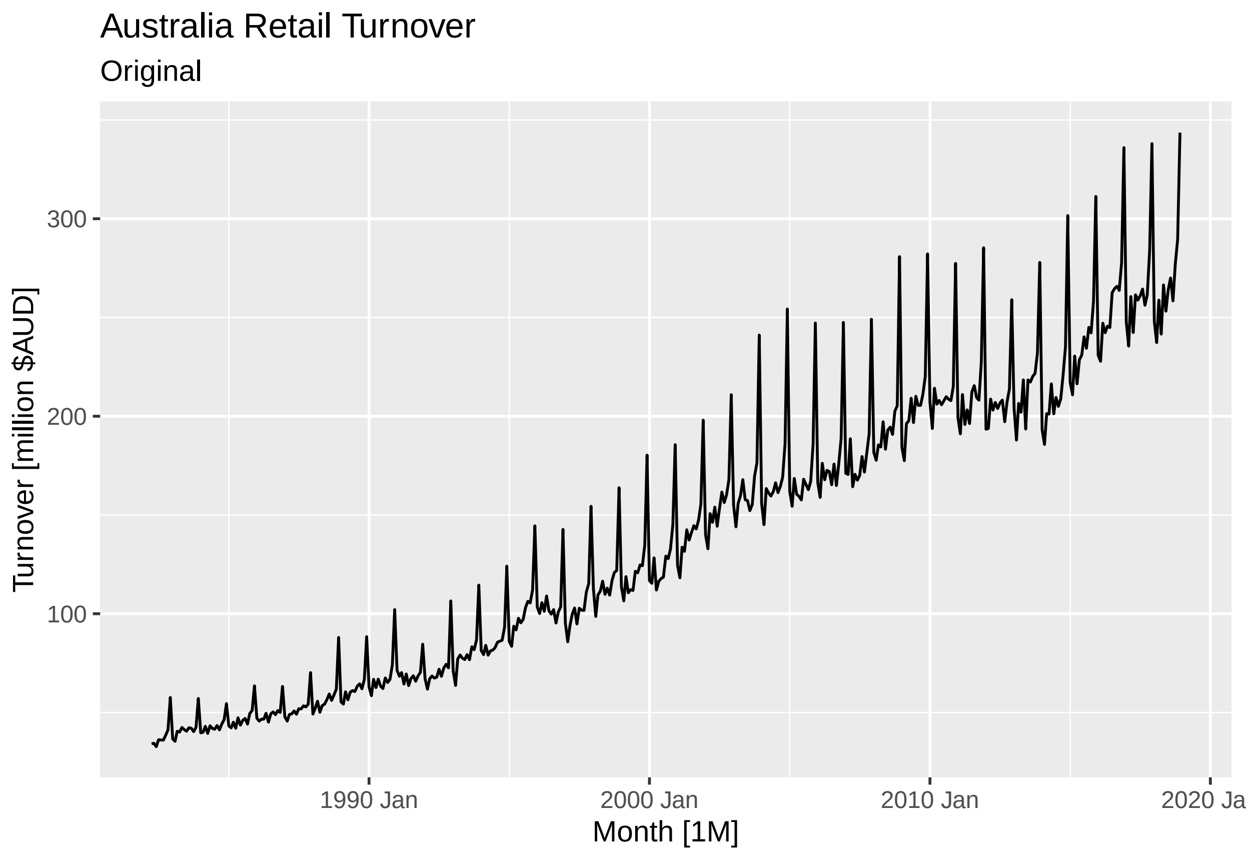
TODO

- Write analysis for Diagnostics

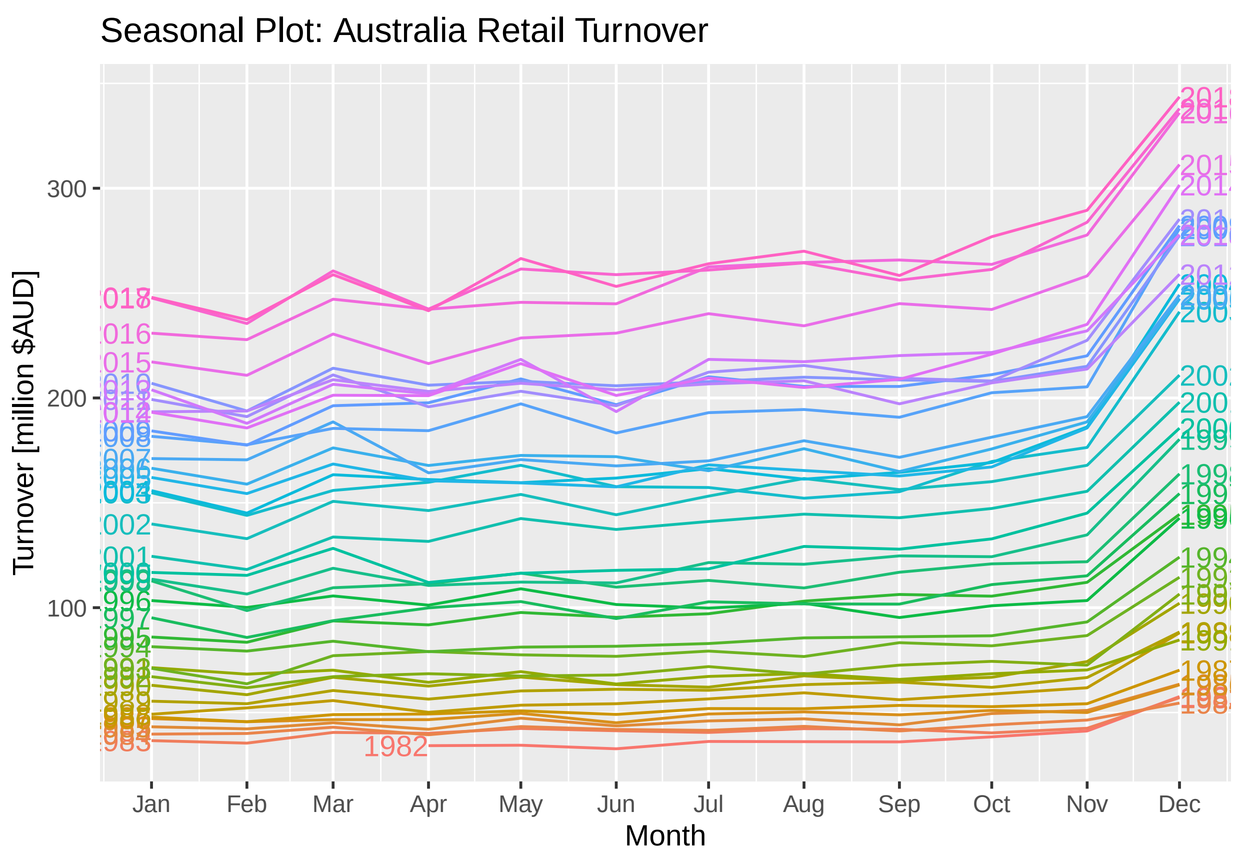
Statistical features of Australian Retail

Table 1: A few rows of the dataset, Other Retailing in South Australia

State	Industry	Series ID	Month	Turnover
South Australia	Other retailing	A3349433W	1982 Apr	34.2
South Australia	Other retailing	A3349433W	1982 May	34.4
South Australia	Other retailing	A3349433W	1982 Jun	32.7
South Australia	Other retailing	A3349433W	1982 Jul	36.2
South Australia	Other retailing	A3349433W	1982 Aug	36.1
South Australia	Other retailing	A3349433W	1982 Sep	36.0



Plotting the original dataset, we can see a general trend upwards. The dataset also has a seasonal pattern which during the early 1980 - 1990, was relatively small compared to later years where the spike grows to large proportions.



From the seasonal plot, there is a clear increase in retail sales during the month December which suggests that there is strong seasonality with the data.

Transformations and Differencings

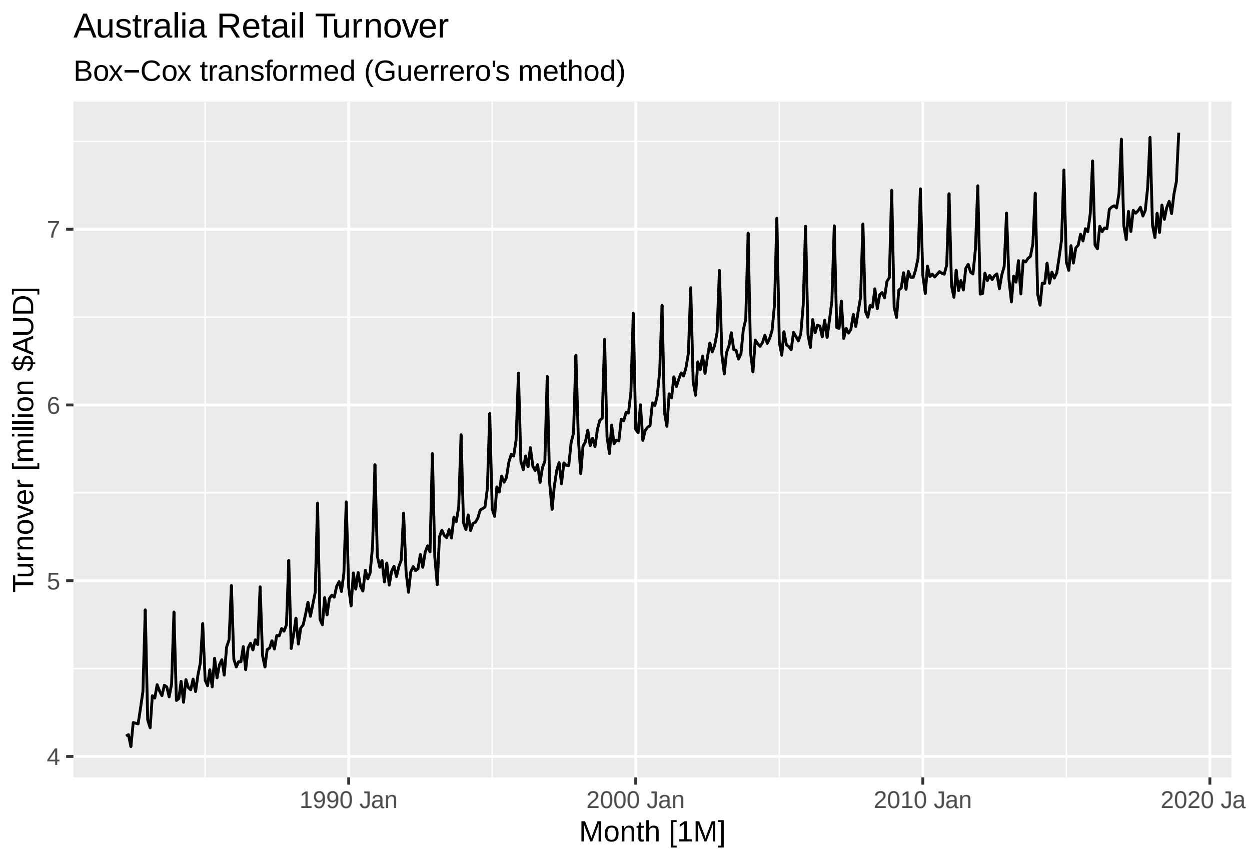
Transformations

To prepare the dataset for ARIMA modelling, the data needs to be tranformed so that the variance across the dataset remains relatively constant. As mentioned before the original dataset does not have a constant seasonal variation, since the beginning has small peaks in the seasonal variation which gradually grow over the course of approximately two decades.

To stabalise the seasonal variation, a Box-Cox transformation can be use. A Box-Cox transformation needs a value for λ to transformed the date. A $\lambda = 0$ will perform a logrithmic tranformation to the data, whereas $\lambda \neq 0$ will perform a exponential transformation.

```
lambda <- myseries %>%
    features(Turnover, features = guerrero) %>%
    pull(lambda_guerrero)

myseries %>% autoplot(box_cox(Turnover, lambda)) +
    labs(
        title = "Australia Retail Turnover",
        subtitle = "Box-Cox transformed (Guerrero's method)"
    ) +
    ylab("Turnover [million $AUD]")
```



Using a λ value close to 0 such as 0.0845031 gives the best transformation that keeps the seasonal variations constant through out the time series. This value was determined by using Guerreor's method and then manually checking values around it to verify its the best value for λ.

Differencing

To objectively determine if the dataset needs differencing we will use unitroot_kpss & unitroot_nsdiffs.

```
myseries %>%
    features(Turnover, unitroot_kpss) %>%
    kable(caption = "Kwiatkowski-Phillips-Schmidt-Shin Test")
```

Table 2: Kwiatkowski-Phillips-Schmidt-Shin Test

State	Industry	kpss_stat	kpss_pvalue
South Australia	Other retailing	7.376257	0.01

```
myseries %>%
    features(difference(Turnover, 12), unitroot_kpss) %>%
    kable(caption = "Kwiatkowski-Phillips-Schmidt-Shin Test")
```

Table 3: Kwiatkowski-Phillips-Schmidt-Shin Test

State	Industry	kpss_stat	kpss_pvalue
South Australia	Other retailing	0.3540707	0.0969523

The KPSS testis performing a hypothesis test to verify that the data is stationary. However, the null hypothesis is rejected since the kpss_pvalue is less than 0.05, hence indicating that the data is not stationary and needs differencing to make it stationary.

After a first order differencing, the hypothesis test gives a p value of 0.10 which is greater than 0.05 which means the null hypothesis is not rejected. Thus making the first difference of the dataset, stationary.

Based on Kwiatkowski-Phillips-Schmidt-Shin's Test, we have determined that the data needs differencing to make it stationary, while we tested with a first stage differencing we need to verify this using the unitroot_ndiffs which reveals the number of differencing that is needed.

```r
myseries %>%
    features(box_cox(Turnover, lambda), unitroot_ndiffs) %>%
    kable(caption = "Number of differences required for a stationary series")
```

Table 4: Number of differences required for a stationary series

State	Industry	ndiffs
South Australia	Other retailing	1

According to unitroot_ndiffs the data only needs to perform a first stage differencing.

```r
myseries %>%
    features(box_cox(Turnover, lambda), unitroot_nsdiffs) %>%
    kable(
        caption =
            "Number of seasonal differences required for a stationary series"
    )
```

Table 5: Number of seasonal differences required for a stationary series

State	Industry	nsdiffs
South Australia	Other retailing	1

According to unitroot_ndiffs the data only needs to perform a first stage seasonal differencing.

Modelling ARIMA and ETS models

ETS Modelling

For the ETS models, the best method to short list possible candidate models is to use the AIC values that are outputed after models are trained. AIC, otherwise known as Akaike's Information Criterion, is defined as $AIC = T\log(\frac{SSE}{T}) + 2(k+2)$. Using it to find a model by minimising the AIC will result in a model that is good at forecasting.

To find the model with the lowest AIC value, we need to check all the different models which can be done by `model(ETS(Turnover))` or manually testing all combinations.

```r
progressr::with_progress(
    trained_etsmodel <- training %>%
        model(
            # Additive
            ANN =  ETS(Turnover ~ error("A") + trend("N") + season("N")),
            ANA =  ETS(Turnover ~ error("A") + trend("N") + season("A")),
            ANM =  ETS(Turnover ~ error("A") + trend("N") + season("M")),
            AAN =  ETS(Turnover ~ error("A") + trend("A") + season("N")),
            AAA =  ETS(Turnover ~ error("A") + trend("A") + season("A")),
            AAM =  ETS(Turnover ~ error("A") + trend("A") + season("M")),
            AAdN = ETS(Turnover ~ error("A") + trend("Ad") + season("N")),
            AAdA = ETS(Turnover ~ error("A") + trend("Ad") + season("A")),
            AAdM = ETS(Turnover ~ error("A") + trend("Ad") + season("M")),

            # Multiplicative
            MNN =  ETS(Turnover ~ error("M") + trend("N") + season("N")),
            MNA =  ETS(Turnover ~ error("M") + trend("N") + season("A")),
            MNM =  ETS(Turnover ~ error("M") + trend("N") + season("M")),
            MAN =  ETS(Turnover ~ error("M") + trend("A") + season("N")),
            MAA =  ETS(Turnover ~ error("M") + trend("A") + season("A")),
            MAM =  ETS(Turnover ~ error("M") + trend("A") + season("M")), #*
            MAdN = ETS(Turnover ~ error("M") + trend("Ad") + season("N")),
            MAdA = ETS(Turnover ~ error("M") + trend("Ad") + season("A")),
            MAdM = ETS(Turnover ~ error("M") + trend("Ad") + season("M"))
        )
)

trained_etsmodel %>%
    glance() %>%
    select(.model, AIC, AICc) %>%
    arrange(AIC) %>%
    kable(caption = "ETS Models ranked based on AIC")
```

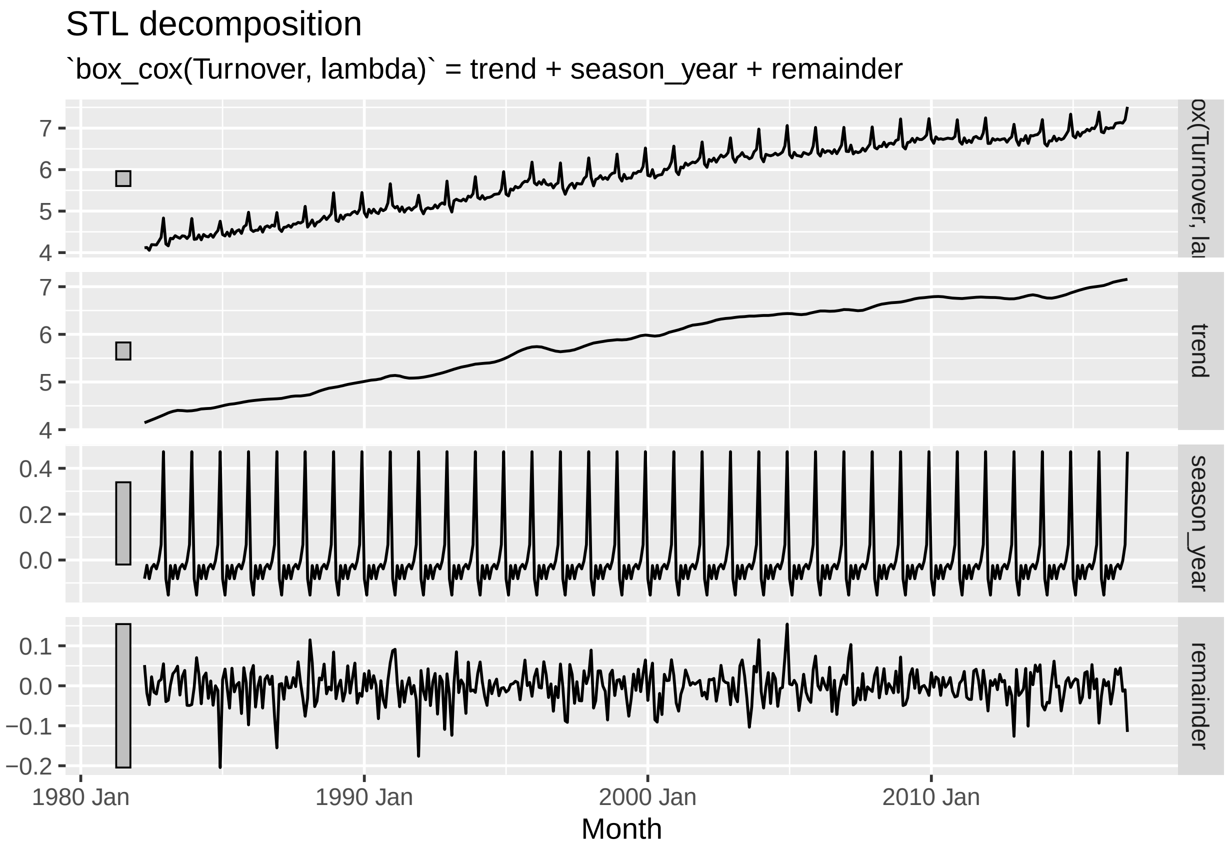
Table 6: ETS Models ranked based on AIC

.model	AIC	AICc
MAM	3708.183	3709.717
MAdM	3720.894	3722.612
MNM	3837.702	3838.899
AAM	3904.291	3905.825
AAdM	3906.502	3908.221
ANM	3943.362	3944.559
ANA	4034.415	4035.612
MAA	4038.320	4039.854
MNA	4046.030	4047.227
AAA	4058.046	4059.579
MAdA	4066.866	4068.585
AAdA	4079.408	4081.127
MAN	4737.305	4737.451
MAdN	4748.376	4748.580
MNN	4758.211	4758.269
AAdN	4930.541	4930.746
AAN	4931.342	4931.488

.model	AIC	AICc
ANN	4950.265	4950.323

By testing all combinations we see that the top 6 ETS models are: MAM, MAdM, MNM, AAM, AAdM, ANM. The list of ETS models can be further reduced by plotting a time series decomposition which shows the relationship between the seasonality and error components.

```
training %>%
    model(STL(
        box_cox(Turnover, lambda) ~
            trend(window = 13) + season(window = "periodic"),
        robust = TRUE
    )) %>%
    components() %>%
    autoplot()
```



The decomposition shown in the graph indicates that the time series is multiplicative. Since the transformation done to the dataset was a box_cox transformation using a lambda value of 0.0845031 which is closer to 0. And according to the box_cox transofrmation lambda value being zero is a log transformation. Hence the multiplicative decomposition.

So it can be deduced that the time series has a multiplicative relationship both the seasonality and error components which means the final short list of ETS models are: MNM, MAM, MAdM.

After short listing, the best ETS model according to the AIC values is a MAM with a AIC value of 3709.7172205

```
bind_rows(
    trained_etsmodel %>%
        accuracy(),
    trained_etsmodel %>%
        forecast(h = "2 years") %>%
        accuracy(myseries)
) %>%
    arrange(MASE) %>%
    select(-State, -Industry, -ME, -MPE, -ACF1) %>%
    kable(caption = "Shortlist of the best ETS Models ranked on MASE")
```

Table 7: Shortlist of the best ETS Models ranked on MASE

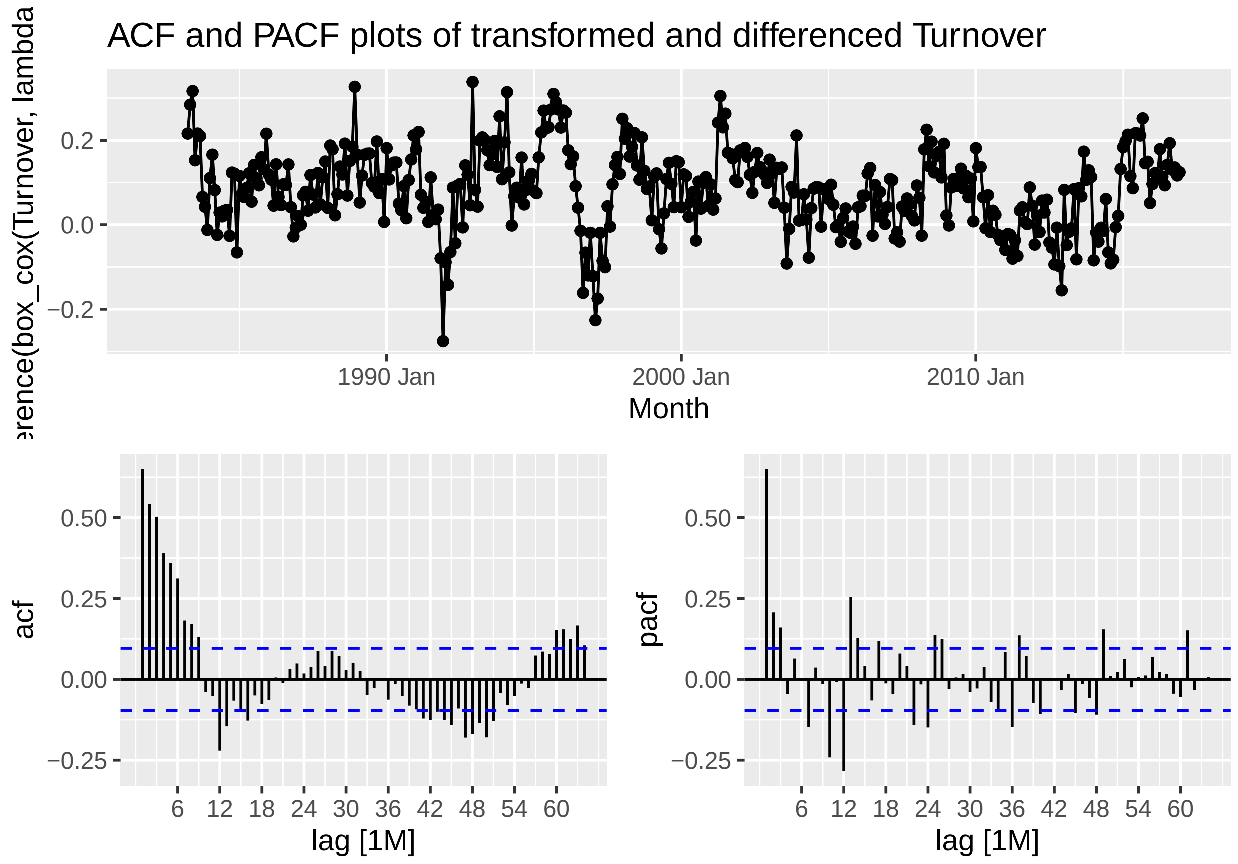
.model	.type	RMSE	MAE	MAPE	MASE	RMSSE
MAM	Training	5.119750	3.631144	2.852607	0.4308353	0.4713354
MAdM	Training	5.210748	3.683136	2.884265	0.4370042	0.4797128
MNM	Training	5.559450	4.072462	3.294554	0.4831977	0.5118151
MNM	Test	6.722146	4.839242	1.812264	0.5741762	0.6188555
MAM	Test	8.277969	6.249165	2.226650	0.7414636	0.7620879
MAdM	Test	8.230045	6.369881	2.279294	0.7557866	0.7576759

After testing the models with the last 2 years of the dataset, the MNM model proves to be the best across RMSE, MAPE and MASE. The MAM and MAdM are also just as good with both tied for second place since their RMSE, MAPE and MASE values are marginally different.

ARIMA Modelling

From section Transformations and Differencings, we know that the dataset needs to transformed and differenced once to make the data stationary. So we can directly plot the ACF and PACF plots of the stationary dataset.

```
training %>% gg_tsdisplay(
    difference(box_cox(Turnover, lambda), 12),
    plot_type = "partial", lag = 64
) + labs(title = "ACF and PACF plots of transformed and differenced Turnover")
```



The ACF plot shows a decaying sinusoidal pattern and by looking at the PACF the last siginificant lag is at lag 3 which may suggest a non-seasonal AR(3).

There is exponential decay in the seasonal lags of the PACF which could indicate a seasonal MA(3) component.

From the ACF and PACF plots we have a possible ARIMA model to start from ARIMA(box_cox(Turnover, lambda) ~ 1 + pdq(3, 0, 0) + PDQ(0, 1, 3))

To get better values we could make the p, P, q & Q values drift by 1 value and also checking with the addition of a constant.

```
progressr::with_progress(
    trained_arimamodel <- training %>% model(
        # search = ARIMA(box_cox(Turnover, lambda), stepwise = FALSE),
        # stepwise = ARIMA(box_cox(Turnover, lambda)),
        ARIMA1300013 = ARIMA(box_cox(Turnover, lambda) ~ 1 + pdq(3, 0, 0) + PDQ(0, 1, 3)),
        ARIMA0300013 = ARIMA(box_cox(Turnover, lambda) ~ 0 + pdq(3, 0, 0) + PDQ(0, 1, 3)),
        ARIMA1200013 = ARIMA(box_cox(Turnover, lambda) ~ 1 + pdq(2, 0, 0) + PDQ(0, 1, 3)),
        ARIMA0200013 = ARIMA(box_cox(Turnover, lambda) ~ 0 + pdq(2, 0, 0) + PDQ(0, 1, 3)),
        ARIMA1201013 = ARIMA(box_cox(Turnover, lambda) ~ 1 + pdq(2, 0, 1) + PDQ(0, 1, 3)),
        ARIMA0201013 = ARIMA(box_cox(Turnover, lambda) ~ 0 + pdq(2, 0, 1) + PDQ(0, 1, 3)),
        ARIMA1300112 = ARIMA(box_cox(Turnover, lambda) ~ 1 + pdq(3, 0, 0) + PDQ(1, 1, 2)),
        ARIMA0300112 = ARIMA(box_cox(Turnover, lambda) ~ 0 + pdq(3, 0, 0) + PDQ(1, 1, 2)),
        ARIMA1200112 = ARIMA(box_cox(Turnover, lambda) ~ 1 + pdq(2, 0, 0) + PDQ(1, 1, 2)),
        ARIMA0200112 = ARIMA(box_cox(Turnover, lambda) ~ 0 + pdq(2, 0, 0) + PDQ(1, 1, 2))
    )
)
```

```r
trained_arimamodel %>%
    glance() %>%
    select(.model, AIC, AICc) %>%
    arrange(AIC) %>%
    kable(caption = "ARIMA Models ranked based on AIC")
```

Table 8: ARIMA Models ranked based on AIC

.model	AIC	AICc
ARIMA1300013	-1173.091	-1172.728
ARIMA1300112	-1172.961	-1172.597
ARIMA1201013	-1169.124	-1168.760
ARIMA0300013	-1168.133	-1167.851
ARIMA0300112	-1168.082	-1167.800
ARIMA0201013	-1165.399	-1165.116
ARIMA1200013	-1141.190	-1140.908
ARIMA1200112	-1141.095	-1140.813
ARIMA0200013	-1131.407	-1131.196
ARIMA0200112	-1131.134	-1130.923

By checking slightly different variations of the initially obtained ARIMA model we can tell that they are fit the date well since their AIC values are close together.

```r
bind_rows(
    trained_arimamodel %>%
        accuracy(),
    trained_arimamodel %>%
        forecast(h = "2 years") %>%
        accuracy(myseries)
) %>%
    arrange(MASE) %>%
    select(-State, -Industry, -ME, -MPE, -ACF1) %>%
    kable(caption = "Shortlist of the best ARIMA Models ranked on MASE")
```

Table 9: Shortlist of the best ARIMA Models ranked on MASE

.model	.type	RMSE	MAE	MAPE	MASE	RMSSE
ARIMA1201013	Training	4.952779	3.563100	2.793952	0.4227619	0.4559636
ARIMA1200112	Training	5.038566	3.580408	2.847238	0.4248155	0.4638614
ARIMA1200013	Training	5.037117	3.580471	2.848343	0.4248230	0.4637279
ARIMA1300013	Training	5.048520	3.582733	2.765884	0.4250913	0.4647777
ARIMA1300112	Training	5.051571	3.584964	2.766082	0.4253561	0.4650586
ARIMA0201013	Training	4.992002	3.600041	2.823828	0.4271449	0.4595746
ARIMA0300112	Training	5.080727	3.640988	2.812820	0.4320033	0.4677428
ARIMA0300013	Training	5.086617	3.646139	2.817608	0.4326145	0.4682851
ARIMA0200013	Training	5.064571	3.652992	2.916282	0.4334276	0.4662554
ARIMA0200112	Training	5.091195	3.678640	2.937793	0.4364707	0.4687065
ARIMA0200112	Test	10.199216	7.534512	2.753414	0.8939701	0.9389621
ARIMA0200013	Test	11.490382	8.632952	3.156135	1.0242999	1.0578296
ARIMA0300112	Test	13.848926	10.772215	3.953826	1.2781236	1.2749624
ARIMA0300013	Test	13.822333	10.772470	3.955113	1.2781539	1.2725141
ARIMA0201013	Test	16.329307	13.299378	4.896372	1.5779715	1.5033116

.model	.type	RMSE	MAE	MAPE	MASE	RMSSE
ARIMA1201013	Test	20.765357	17.286884	6.366140	2.0510893	1.9117041
ARIMA1300112	Test	22.132087	18.125307	6.659770	2.1505682	2.0375282
ARIMA1300013	Test	22.341353	18.350703	6.745479	2.1773114	2.0567937
ARIMA1200112	Test	28.183447	23.443429	8.613578	2.7815634	2.5946296
ARIMA1200013	Test	28.354257	23.663139	8.697970	2.8076321	2.6103548

After testing the models with the last 2 years of the dataset, the manualXv2 models performed the best during the testing phase.

Best Models for Forecasting

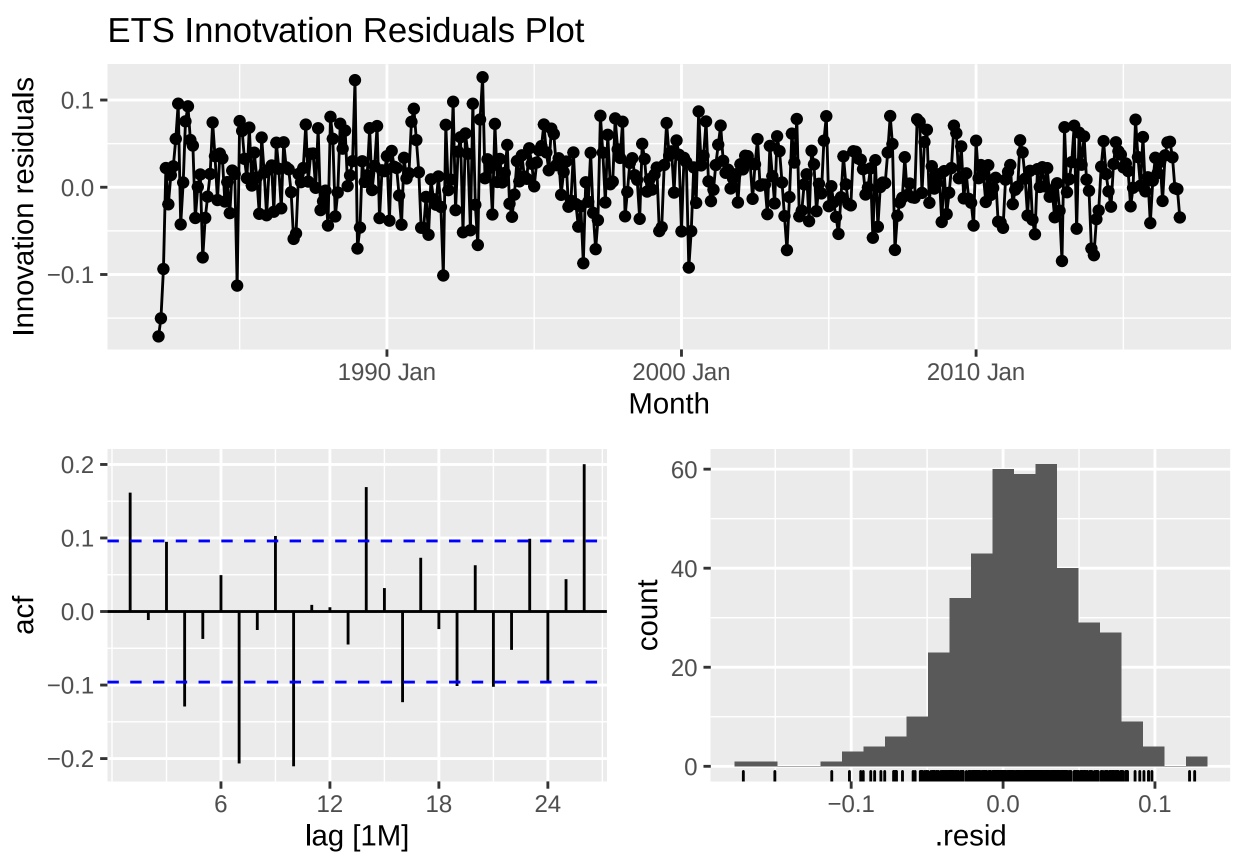
Best ETS Model

```
MNM <- trained_etsmodel %>%
    select(MNM)


MNM %>%
    report()
```

```
## Series: Turnover
## Model: ETS(M,N,M)
##   Smoothing parameters:
##     alpha = 0.3953603
##     gamma = 0.2146401
##
##   Initial states:
##       l[0]      s[0]      s[-1]      s[-2]     s[-3]    s[-4]    s[-5]     s[-6]
##  43.82605 0.9583036 0.8979279 0.9585486 1.367883 1.038136 1.006751 0.9584283
##       s[-7]     s[-8]     s[-9]    s[-10]    s[-11]
##  0.9862604 0.9569759 0.9387507 0.9906923 0.9413432
##
##   sigma^2:  0.0018
##
##      AIC     AICc      BIC
## 3837.702 3838.899 3898.198
```

```
MNM %>%
    gg_tsresiduals() + labs(title = "ETS Innotvation Residuals Plot")
```



```
MNM %>%
    augment() %>%
    features(.innov, ljung_box, lag = 24, dof = 14)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 MNM       109.         0
```

```
MNM_fc <- MNM %>%
    forecast(h = "2 years")

# MNM %>%
    # kable(caption = "")

MNM_fc %>%
    hilo(level = c(80, 95)) %>%
    select(-.model, -Turnover) %>%
    kable(caption = "ETS Point Forecast and Point Interval (80% & 95%)")
```

Table 10: ETS Point Forecast and Point Interval (80% & 95%)

Month	.mean	80%	95%
2017 Jan	248.9254	[235.2131, 262.6378]80	[227.9542, 269.8967]95
2017 Feb	239.9734	[225.7568, 254.1901]80	[218.2309, 261.7159]95
2017 Mar	260.3434	[243.9087, 276.7781]80	[235.2087, 285.4781]95

Month	.mean	80%	95%
2017 Apr	251.1777	[234.4023, 267.9530]80	[225.5219, 276.8334]95
2017 May	260.6503	[242.3377, 278.9629]80	[232.6436, 288.6571]95
2017 Jun	251.1366	[232.6617, 269.6116]80	[222.8817, 279.3916]95
2017 Jul	262.4349	[242.2979, 282.5720]80	[231.6380, 293.2319]95
2017 Aug	259.1267	[238.4553, 279.7980]80	[227.5126, 290.7408]95
2017 Sep	258.4658	[237.0897, 279.8419]80	[225.7739, 291.1577]95
2017 Oct	259.9801	[237.7428, 282.2175]80	[225.9710, 293.9892]95
2017 Nov	274.1267	[249.9279, 298.3255]80	[237.1179, 311.1356]95
2017 Dec	340.7373	[309.7523, 371.7223]80	[293.3498, 388.1248]95
2018 Jan	248.9645	[224.8216, 273.1073]80	[212.0412, 285.8877]95
2018 Feb	240.0110	[216.1534, 263.8687]80	[203.5240, 276.4981]95
2018 Mar	260.3842	[233.8838, 286.8846]80	[219.8554, 300.9130]95
2018 Apr	251.2170	[225.0672, 277.3669]80	[211.2243, 291.2098]95
2018 May	260.6912	[232.9637, 288.4187]80	[218.2857, 303.0967]95
2018 Jun	251.1760	[223.9025, 278.4495]80	[209.4648, 292.8872]95
2018 Jul	262.4761	[233.4039, 291.5482]80	[218.0140, 306.9381]95
2018 Aug	259.1673	[229.9079, 288.4267]80	[214.4189, 303.9157]95
2018 Sep	258.5063	[228.7793, 288.2333]80	[213.0428, 303.9699]95
2018 Oct	260.0209	[229.5840, 290.4578]80	[213.4716, 306.5702]95
2018 Nov	274.1697	[241.5213, 306.8182]80	[224.2382, 324.1012]95
2018 Dec	340.7907	[299.5301, 382.0514]80	[277.6880, 403.8934]95

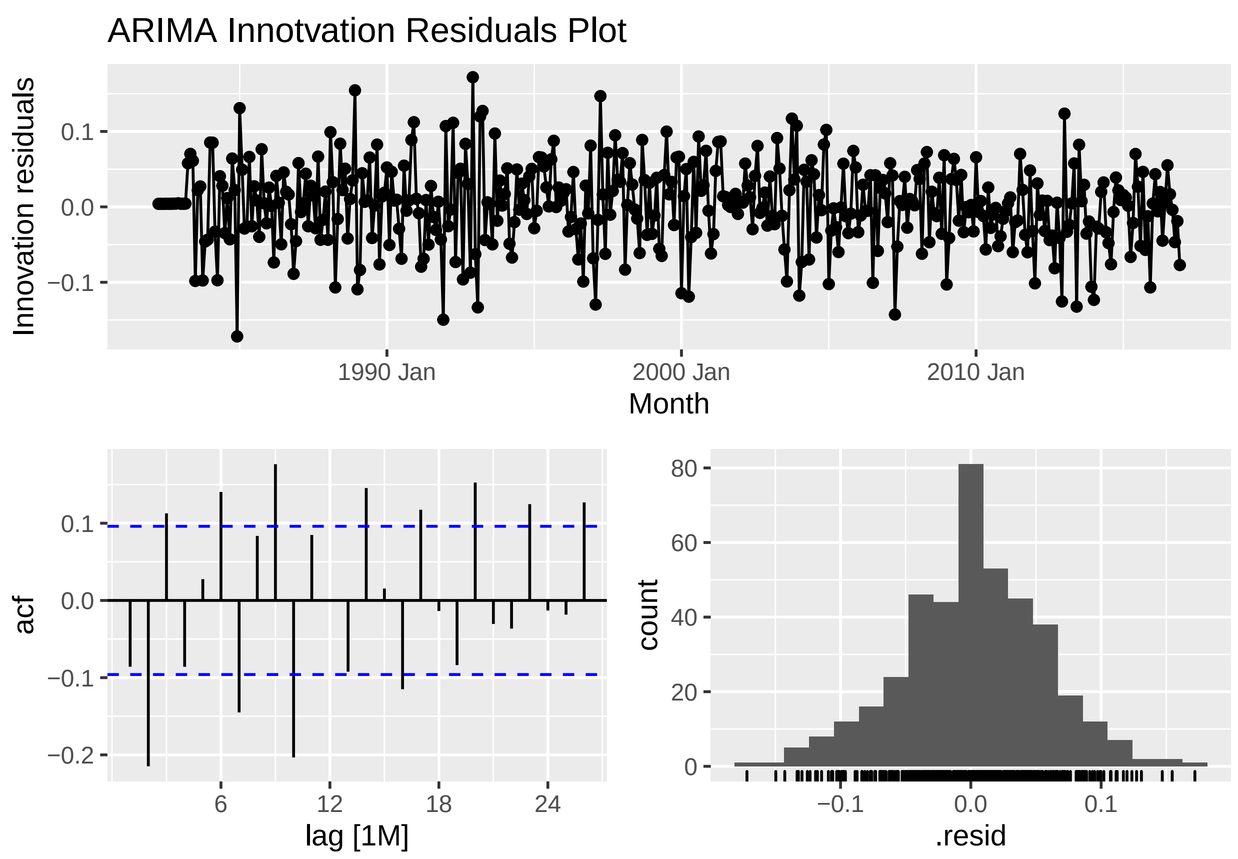
Best ARIMA Model

```
ARIMA1200013 <- trained_arimamodel %>%
    select(ARIMA1200013)

ARIMA1200013 %>%
    report()
```

```
## Series: Turnover
## Model: ARIMA(2,0,0)(0,1,3)[12] w/ drift
## Transformation: box_cox(Turnover, lambda)
##
## Coefficients:
##          ar1     ar2     sma1     sma2     sma3  constant
##       0.6168  0.3072  -0.7805  -0.1928  -0.0267    0.0064
## s.e.  0.0476  0.0480   0.1003   0.0636   0.0599    0.0003
##
## sigma^2 estimated as 0.003118:  log likelihood=577.6
## AIC=-1141.19   AICc=-1140.91   BIC=-1113.16
```

```
ARIMA1200013 %>%
    gg_tsresiduals() + labs(title = "ARIMA Innotvation Residuals Plot")
```



```
ARIMA1200013 %>%
    augment() %>%
    features(.innov, ljung_box, lag = 12, dof = 5)
```

```
## # A tibble: 1 x 3
##   .model       lb_stat lb_pvalue
##   <chr>          <dbl>     <dbl>
## 1 ARIMA1200013    85.8  8.88e-16
```

```
ARIMA1200013_fc <- ARIMA1200013 %>%
    forecast(h = "2 years")

# ARIMA1200013_fc %>% kable(caption = "")

ARIMA1200013_fc %>%
    hilo(level = c(80, 95)) %>%
    select(-.model, -Turnover) %>%
    kable(caption = "ARIMA Point Forecast and Point Interval (80% & 95%)")
```

Table 11: ARIMA Point Forecast and Point Interval (80% & 95%)

Month	.mean	80%	95%
2017 Jan	248.5252	[237.3024, 259.9325]80	[231.6224, 266.2443]95
2017 Feb	240.0030	[227.2531, 253.0002]80	[220.8522, 260.2481]95
2017 Mar	262.9021	[246.8712, 279.2912]80	[238.8872, 288.5013]95
2017 Apr	256.4513	[239.2871, 274.0373]80	[230.7907, 283.9780]95

Month	.mean	80%	95%
2017 May	266.5255	[247.4292, 286.1251]80	[238.0223, 297.2555]95
2017 Jun	260.5440	[240.7839, 280.8564]80	[231.0921, 292.4392]95
2017 Jul	272.8736	[251.3235, 295.0517]80	[240.7880, 307.7374]95
2017 Aug	277.1969	[254.5282, 300.5504]80	[243.4780, 313.9454]95
2017 Sep	277.0420	[253.6938, 301.1176]80	[242.3420, 314.9609]95
2017 Oct	283.4963	[259.0412, 308.7319]80	[247.1759, 323.2705]95
2017 Nov	298.2578	[272.0853, 325.2807]80	[259.4063, 340.8717]95
2017 Dec	377.7600	[344.6584, 411.9356]80	[328.6206, 431.6510]95
2018 Jan	274.5012	[248.7236, 301.1774]80	[236.3163, 316.6619]95
2018 Feb	264.5089	[238.9338, 291.0028]80	[226.6599, 306.4239]95
2018 Mar	288.9749	[260.5486, 318.4411]80	[246.9304, 335.6206]95
2018 Apr	280.8603	[252.6088, 310.1696]80	[239.1061, 327.2951]95
2018 May	292.9437	[263.0641, 323.9584]80	[248.8045, 342.1055]95
2018 Jun	284.6634	[255.1236, 315.3457]80	[241.0525, 333.3298]95
2018 Jul	295.7456	[264.7571, 327.9449]80	[250.0117, 346.8369]95
2018 Aug	300.1420	[268.3803, 333.1576]80	[253.2836, 352.5485]95
2018 Sep	299.3583	[267.3655, 332.6271]80	[252.1760, 352.1870]95
2018 Oct	307.8634	[274.7610, 342.2945]80	[259.0555, 362.5508]95
2018 Nov	323.3181	[288.4399, 359.6010]80	[271.8981, 380.9541]95
2018 Dec	412.4577	[368.5696, 458.0884]80	[347.7221, 484.9039]95