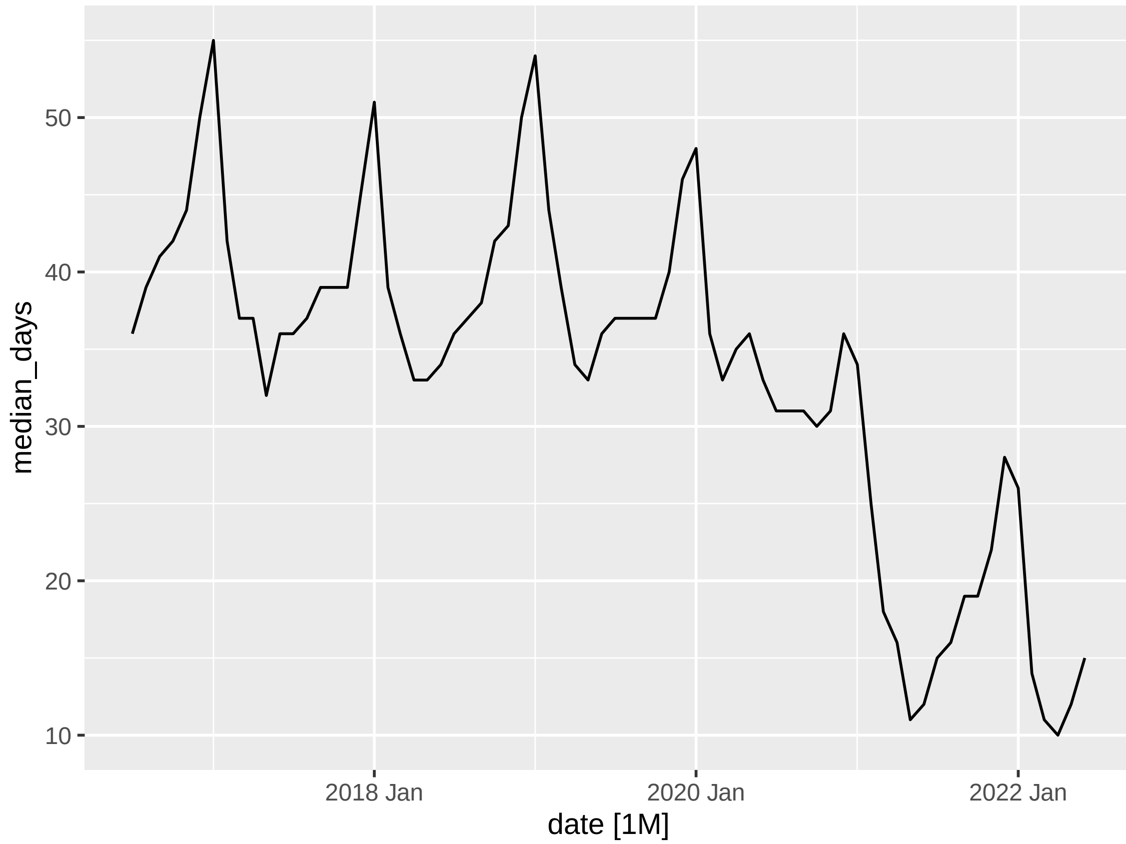
Assignment 5

Jaewoo Cho

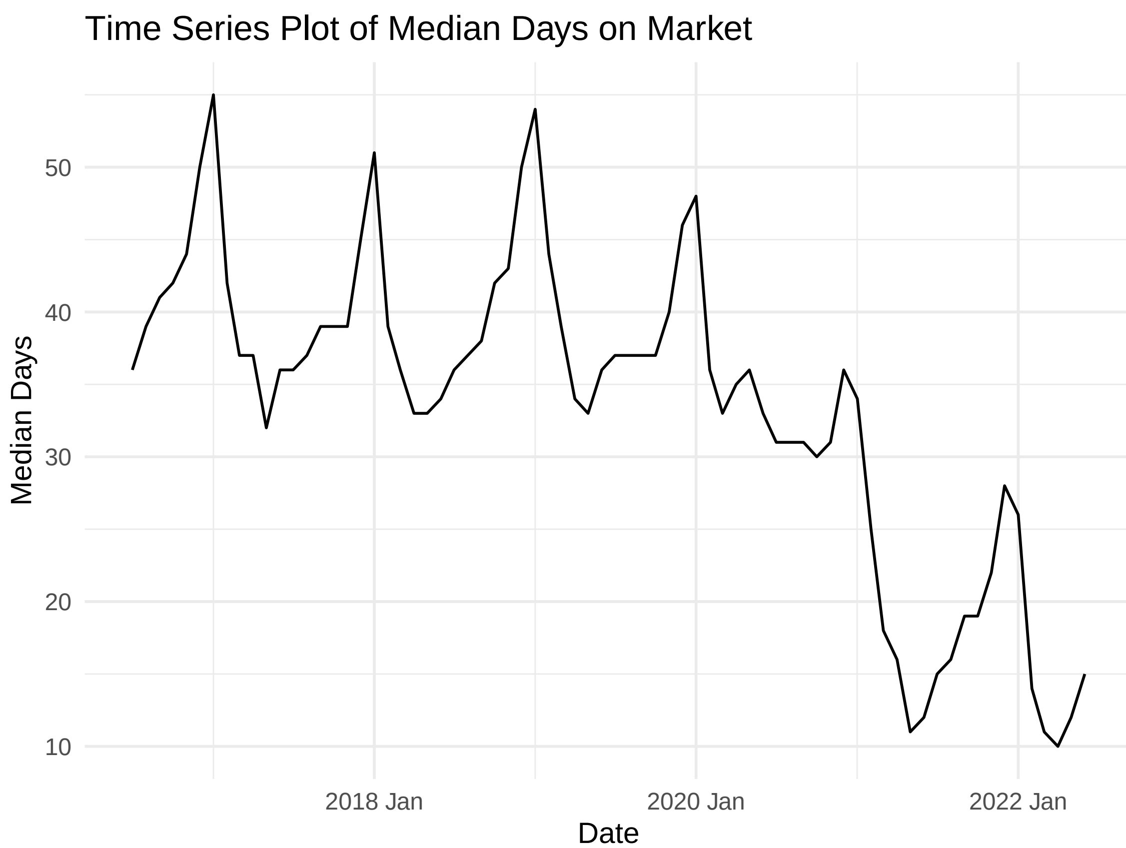
Build and forecast `median_days` houses are on the market in the Nashville area

1. Plot `median_days` and comment on any patterns in the time series

```r
# Load data
library(readr)
nashville_housing <- read_csv("~/Desktop/DS Fall 2023/DS adv stats/data/nashville_housing.csv")

# Convert date (provided for you)
nashville_housing$date <- yearmonth(nashville_housing$date)

# Convert to `tsibble` (provided for you)
housing_ts <- nashville_housing %>% as_tsibble(index = date)

# Plot `median_days`
housing_ts %>% autoplot(median_days)
```




```r
# Create a time series plot for `median_days`
ggplot(data = housing_ts, aes(x = date, y = median_days)) +
  geom_line() +
  labs(title = "Time Series Plot of Median Days on Market",
       x = "Date",
       y = "Median Days") +
  theme_minimal()
```



Answer: "Comment on any patterns in the time series" - The time series has a pattern that looks like a downward trend that has a decrease in median days as the dates of time passes.

2. Fit `TSLM` on `housing_train` data with all predictors. Report significant predictors and interpret `Multiple R-squared`.

```
# Set up training and testing indices
train <- 1:which(as.character(housing_ts$date) == "2021 Jun")

# Initialize training and testing data
housing_train <- housing_ts[train,]
housing_test <- housing_ts[-train,]

# Fit TSLM with all predictors
# Hint: use `colnames()` to see all variables

#colnames(housing_ts)

fit_tslm <- housing_train %>%
  model(tslm = TSLM(
    median_days ~ unemployment + housing + price_increased +
      price_decreased + pending_listing + median_price))
```

```r
# Report fit
report(fit_tslm)
```

```
Series: median_days
Model: TSLM

Residuals:
     Min       1Q   Median       3Q      Max
-10.4435  -3.1927   0.3305   2.3224  12.7707

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -1.395e+01  4.738e+01  -0.294 0.769594
unemployment    -8.188e-01  3.624e-01  -2.260 0.027987 *
housing          7.789e-03  1.666e-03   4.675 2.06e-05 ***
price_increased  2.515e-03  7.847e-03   0.321 0.749829
price_decreased -1.287e-02  3.119e-03  -4.126 0.000131 ***
pending_listing  4.545e-03  1.795e-03   2.532 0.014356 *
median_price     3.861e-05  1.109e-04   0.348 0.729067
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.31 on 53 degrees of freedom
Multiple R-squared: 0.6484, Adjusted R-squared: 0.6086
F-statistic: 16.29 on 6 and 53 DF, p-value: 1.6e-10
```

> Answer: "Report significant predictors and interpret `Multiple R-squared`." The significant predictors are - unemployment with a coefficient estimate of approximately -0.8188 and a p-value of 0.027987. - housing with a coefficient estimate of approximately 0.0078 and a very low p-value of 2.06e-05. - price_decreased with a coefficient estimate of approximately -0.0129 and a very low p-value of 0.000131. - pending_listing with a coefficient estimate of approximately 0.0045 and a p-value of 0.014356. - These variables are statistically significant predictors because their p-values are below the commonly used significance level of 0.05. They have a meaningful impact on the dependent variable median_days in the regression model. - The Multiple R-squared value, which is 0.6484 in the model, represents the proportion of the variance in the dependent variable (median_days) that is explained by the combination of all the predictor variables included in the model. In other words, approximately 64.84% of the variability in median_days can be accounted for by the variables unemployment, housing, price_increased, price_decreased, pending_listing, and median_price.A higher Multiple R-squared value indicates that a larger portion of the variability in the dependent variable is explained by the independent variables. A Multiple R-squared of 0.6484 suggests that the model has a reasonably good fit, as it captures a substantial portion of the variation in median_days.

3. Check multicolinearity using `lm` and `VIF` functions. Report which predictors have `VIF` > 10 and keep *only* one variable.

```r
# Fit model with `lm`
# Multicolinearity?
fit <- lm(median_days ~ unemployment + housing + price_increased + price_decreased + pending_listing + m

# Check for multicolinearity using `VIF`
```

```r
# had to use car instead of regclass due to error
#regclass::VIF(fit)
car::vif(fit)
```

```
   unemployment         housing price_increased price_decreased pending_listing
       1.410145       21.507180        1.496580       10.460191        7.143705
   median_price
      13.305073
```

```r
# You may need to install the {regclass} package
round(coefficients(fit), 5)[c("pending_listing", "median_price")]
```

```
pending_listing    median_price
        0.00454         0.00004
```

> Answer: "Report which predictors have `VIF` > 10 and say which variable you are deciding to keep." Based on these VIF values, it appears that housing, median_price, and price_decreased have VIF values greater than 10, indicating a high degree of multicollinearity with other predictor variables. High VIF values suggest that these variables are highly correlated with other predictors in the model.Based on the restuls, I am going to remove housing and median_price, but I am going to keep price_decreased as it really close to the VIF > 10.

4. Re-fit `lm` and check for whether multicolinarity remains after keeping *only* one of the multicolinear variables. Are any `VIF` > 10?

```r
# Re-fit model with `lm`
fit <- lm(median_days ~ unemployment + price_increased + price_decreased + pending_listing , data = hous

# Check for multicolinearity using `VIF`
# had to use car instead of regclass due to error
#regclass::VIF(fit)
car::vif(fit)
```

```
   unemployment price_increased price_decreased pending_listing
       1.275576        1.396013        1.283824        1.260736
```

```r
# You may need to install the {regclass} package
round(coefficients(fit), 5)[c("pending_listing")]
```

```
pending_listing
       -0.00446
```

> Answer: "Are any `VIF` > 10?" - There are no VIF > 10 as the results all have really low VIF values that are even lower compared to the previous code.

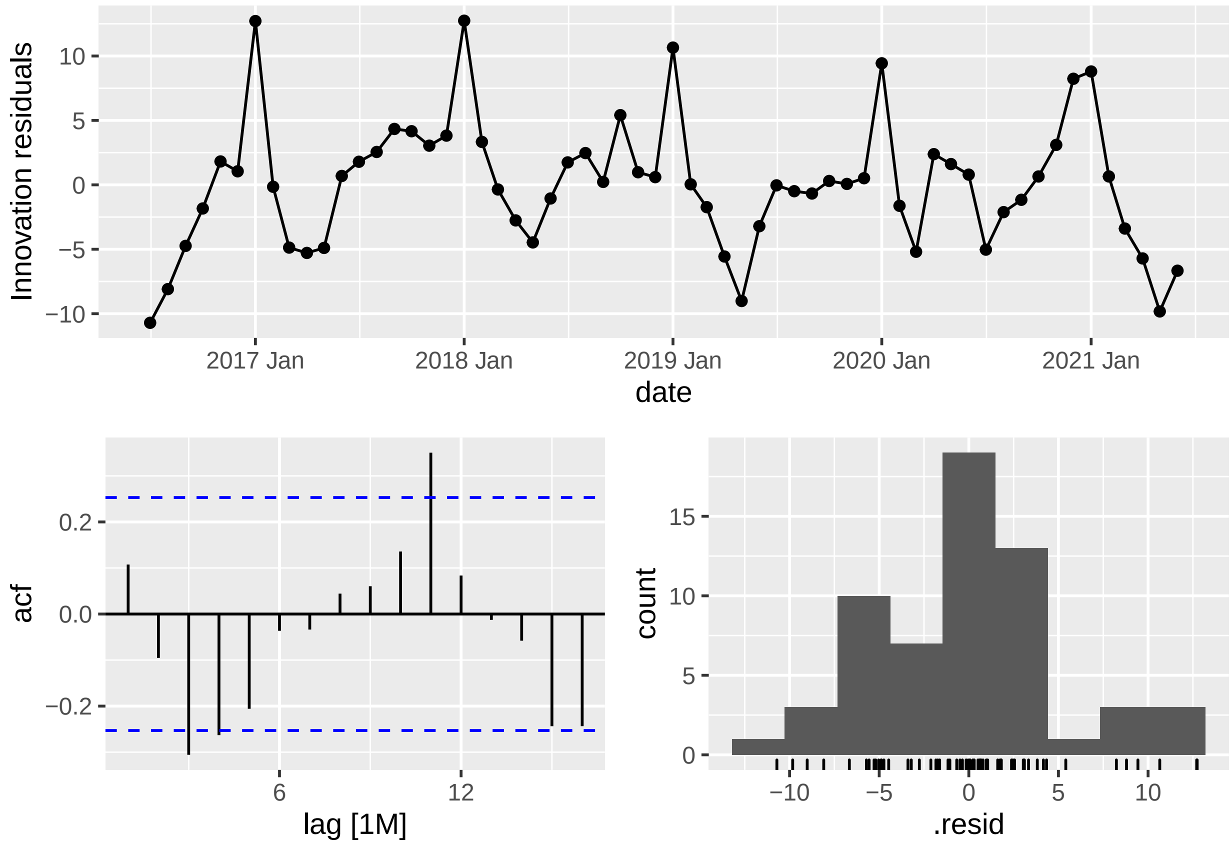
5. Re-fit `TSLM` with significant predictors only

```r
# Re-fit `TSLM` with significant predictors only

# Report fit
fit_tslm_sig <- housing_train %>%
                model(tslm = TSLM(median_days ~ unemployment+ housing + price_decreased + pending_listi
# Report fit
report(fit_tslm_sig)
```

```
Series: median_days
Model: TSLM

Residuals:
     Min       1Q   Median       3Q      Max
-10.7076  -3.2565   0.1487   2.4032  12.7388

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      2.945843   5.793381   0.508  0.61315
unemployment    -0.790912   0.341492  -2.316  0.02431 *
housing          0.007371   0.001113   6.621 1.57e-08 ***
price_decreased -0.012608   0.002875  -4.386 5.26e-05 ***
pending_listing  0.004883   0.001565   3.120  0.00288 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.221 on 55 degrees of freedom
Multiple R-squared: 0.6472, Adjusted R-squared: 0.6216
F-statistic: 25.23 on 4 and 55 DF, p-value: 6.7677e-12
```

6. Plot residuals and perform Ljung-Box test. Are the residuals significantly different from white noise?

```r
# plotting residuals
gg_tsresiduals(fit_tslm_sig)
```




```
# Report fit
fit_tslm_sig %>%
  augment() %>%
  features(.innov, ljung_box, lag = 12, dof = 5)
```

```
# A tibble: 1 x 3
  .model lb_stat   lb_pvalue
  <chr>    <dbl>       <dbl>
1 tslm      40.8 0.000000902
```

Answer: "Are the residuals significantly different from white noise?" Yes, the residuals are significantly different from white noise, as indicated by a very low p-value in the Ljung-Box test. - Ljung-Box Statistic (lb_stat): 40.75451 - lb_stat (Ljung-Box Statistic): This statistic measures the presence of autocorrelation in the residuals. A higher value suggests stronger evidence of autocorrelation. - p-value (lb_pvalue): 9.023594e-07 (a very small p-value) - lb_pvalue (p-value): This is the associated p-value for the Ljung-Box test statistic. The p-value is very small (9.023594e-07), which indicates that the residuals are significantly different from white noise.

7. Fit the same `TSLM` model but now with `ARIMA` (i.e., fit a dynamic regression model). Comment on whether any differencing was used.

```
# Fit model with `ARIMA`
# Set pandemic
```

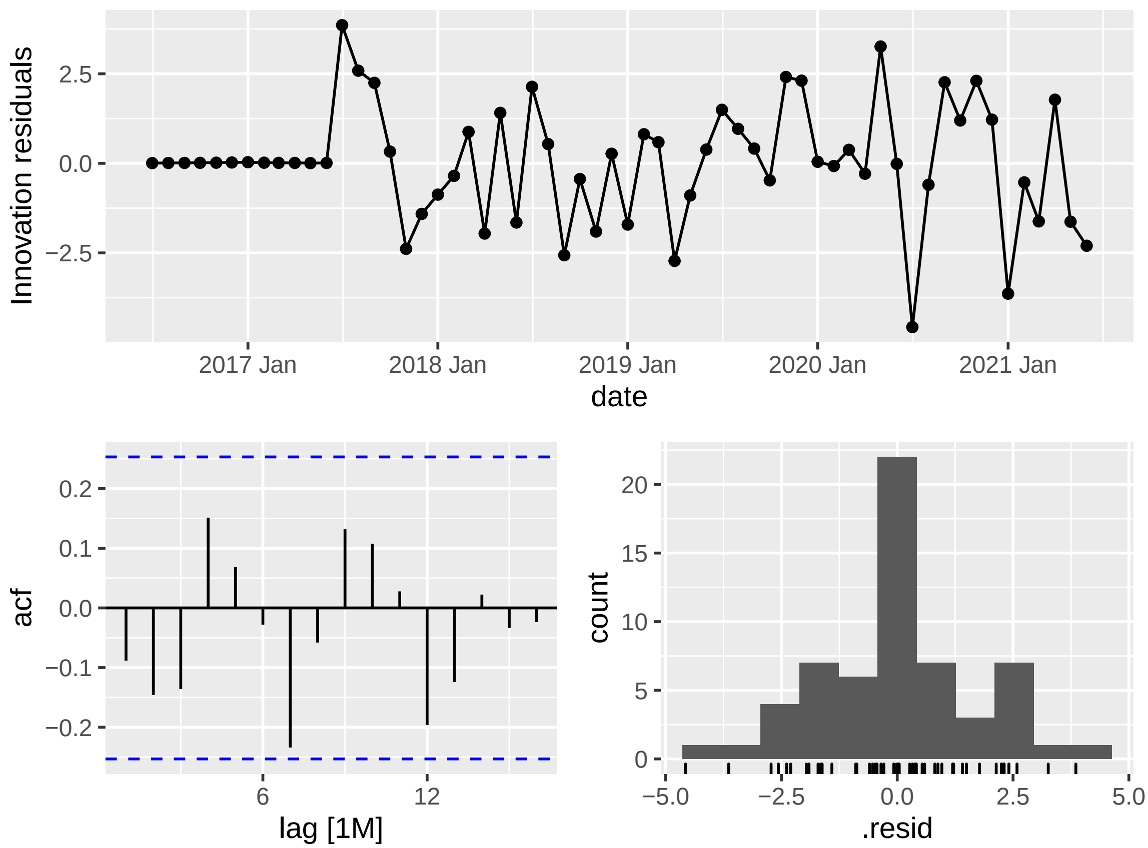
```r
tslm_arima <- housing_train %>%
  model(dynamic = ARIMA(median_days ~ unemployment+ housing + price_decreased + pending_listing))
# Report fit
report(tslm_arima)
```

```
Series: median_days
Model: LM w/ ARIMA(0,0,0)(1,1,0)[12] errors

Coefficients:
         sar1  unemployment  housing  price_decreased  pending_listing
      -0.6520        0.4923   0.0034          -0.0016           0.0032
s.e.   0.1212        0.1360   0.0007           0.0014           0.0008
      intercept
        -2.2258
s.e.     0.3198

sigma^2 estimated as 3.8:  log likelihood=-100.27
AIC=214.54   AICc=217.34   BIC=227.64
```

> Answer: "Comment on whether any differencing was used." The model is specified as "LM w/ ARIMA(0,0,0)(1,1,0)[12] errors." The ARIMA portion of the model is ARIMA(0,0,0)(1,1,0)[12], which includes a seasonal differencing of order 1 and a seasonal moving average term.This indicates that differencing was indeed used in the model, specifically seasonal differencing with a lag of 12 (indicating monthly data). The (1,1,0) part of the ARIMA model represents the seasonal differencing (D=1) and the seasonal order (S=12), respectively. This differencing helps make the time series stationary and remove any trend and seasonality before applying linear regression (LM) to the differenced series.

8. Plot residuals from the dynamic regression model and perform Ljung-Box test. Are the residuals significantly different from white noise?

```r
# Plot residuals
# Residuals
gg_tsresiduals(tslm_arima)
```




```
# Perform Ljung-Box test
# Set lag based on seasonal lag in from `ARIMA` fit
# (remember to adjust dof = number of coefficients)
# Ljung-Box
tslm_arima %>% augment() %>%
  features(.innov, ljung_box, lag = 12, dof = 6)
```

```
# A tibble: 1 x 3
  .model  lb_stat lb_pvalue
  <chr>     <dbl>     <dbl>
1 dynamic    12.6    0.0493
```

Answer: "Are the residuals significantly different from white noise?" Yes, the residuals are marginally different from white noise, as indicated by a p-value of 0.04934696 in the Ljung-Box test. Ljung-Box Statistic (lb_stat): 12.62754, p-value (lb_pvalue): 0.04934696 The p-value is 0.04934696, which is less than the commonly used significance level of 0.05. This indicates that the residuals are marginally different from white noise at a 5% significance level.

9. Fit an `ETS` model on `median_days` and report fit. Interpret the `alpha` and `gamma` parameters.

```
# Fit model with `ETS`

fit_ets <- housing_train %>% model(ETS(median_days))
```

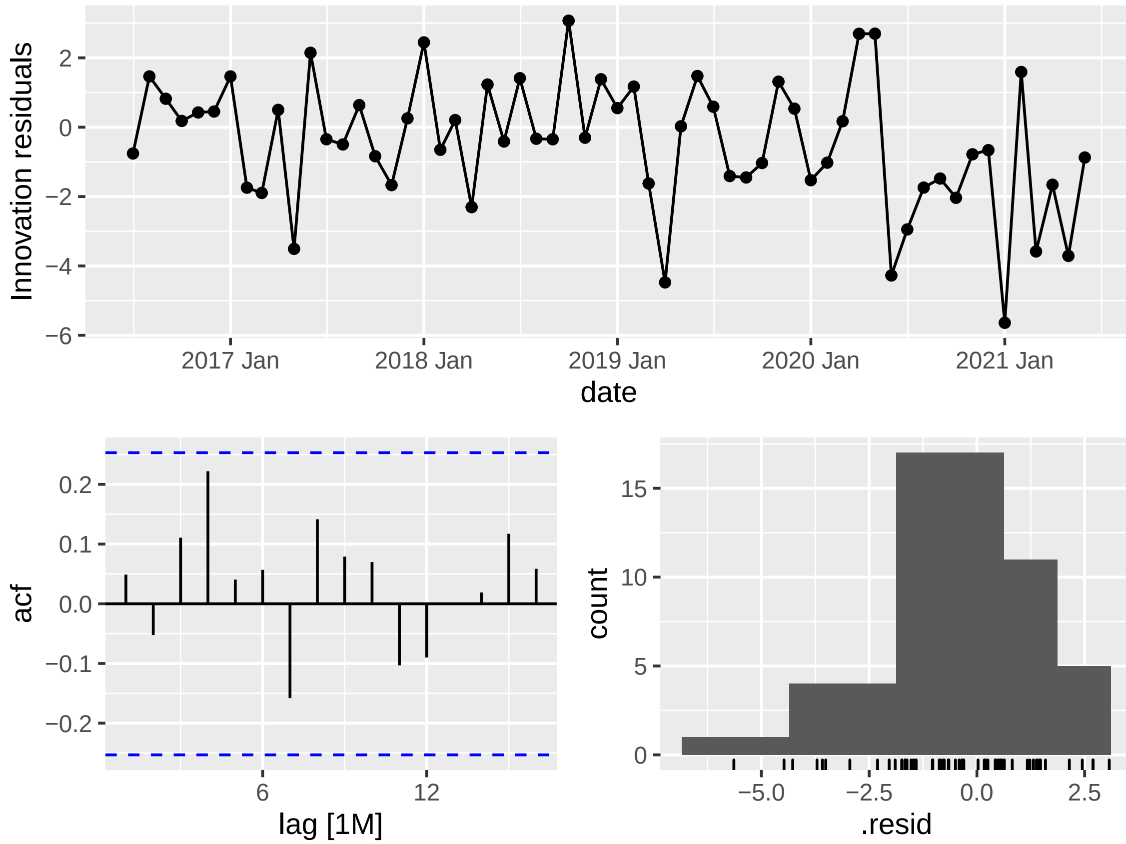
```r
# Report fit
report(fit_ets)
```

```
Series: median_days
Model: ETS(A,N,A)
  Smoothing parameters:
    alpha = 0.9063522
    gamma = 0.0001264875

  Initial states:
     l[0]      s[0]     s[-1]     s[-2]     s[-3]     s[-4]   s[-5]   s[-6]
 39.97425 -3.764648 -5.290642 -3.845123 -3.168479 0.0992651 11.2187 7.64077
    s[-7]     s[-8]      s[-9]    s[-10]    s[-11]
 2.052448 0.4614416 -0.4352389 -1.751995 -3.216495

  sigma^2:  4.6734

     AIC     AICc      BIC
352.2311 363.1402 383.6463
```

> Answer: "Interpret the `alpha` and `gamma` parameters." - Alpha represents the smoothing parameter for the level component of the time series. In the model, alpha is approximately 0.9063522. The level component represents the underlying or average value of the time series data. A higher alpha value gives more weight to recent observations when estimating the level, making it more responsive to recent changes in the data. A high alpha indicates that the model is giving significant weight to recent observations when forecasting the median_days series. - Gamma represents the smoothing parameter for the seasonal component of the time series. In the model, gamma is approximately 0.0001264875. The seasonal component captures regular, repeating patterns in the data, often related to seasonality or cycles. A small gamma value suggests that the seasonal component is not changing rapidly and is relatively stable over time. The seasonal component changes slowly, indicating that the seasonality of the median_days series is not highly volatile.

10. Plot residuals from the `ETS` model and perform Ljung-Box test. Are the residuals significantly different from white noise?

```r
fit <- housing_train %>% model(ANA = ETS(median_days ~ error("A") + trend("N") + season("A")))
# Plot residuals
gg_tsresiduals(fit_ets)
```




```
# Perform Ljung-Box test
# Set lag based on seasonal lag in from `ETS` fit
# Set `dof = 12`
fit_ets %>% augment() %>%
  features(.innov, ljung_box, lag = 12, dof = 12)
```

```
# A tibble: 1 x 3
  .model           lb_stat lb_pvalue
  <chr>              <dbl>     <dbl>
1 ETS(median_days)    9.82         0
```

> Answer: "Are the residuals significantly different from white noise?" Yes, the residuals are significantly different from white noise, as indicated by a Ljung-Box test p-value of 0, suggesting the presence of autocorrelation in the residuals.The p-value associated with the Ljung-Box test statistic measures the significance of the test result.A small p-value (typically below a significance level, such as 0.05) suggests that there is significant autocorrelation in the residuals.The p-value is 0, which means the test has found strong evidence against the null hypothesis of no autocorrelation.

11. Combine all models and forecast using `housing_test` data

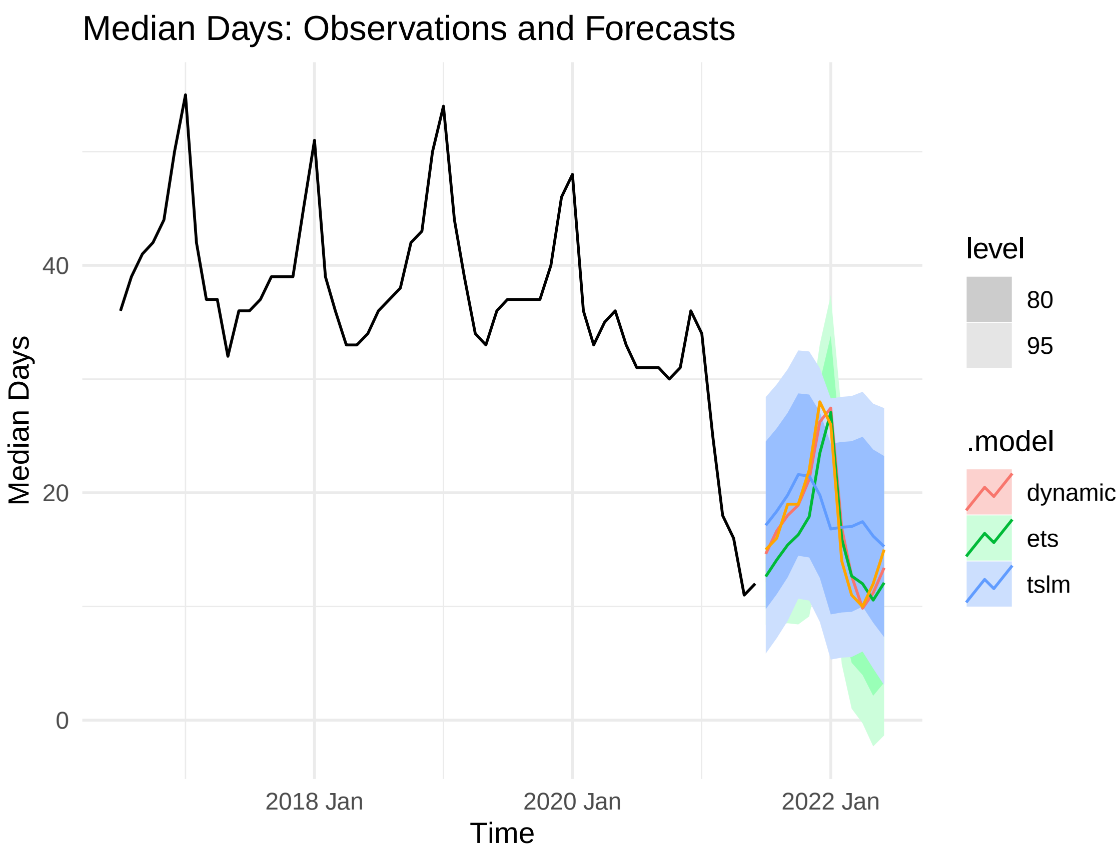
```
# Update test data with outlier and pandemic
# Combine all models
```

```r
all_models <- housing_train %>%
  model(tslm = TSLM(
    median_days ~ unemployment+ housing + price_decreased + pending_listing),
    ets = ETS(median_days),
    dynamic = ARIMA(
    median_days ~ unemployment+ housing + price_decreased + pending_listing))

# Forecast using `housing_test`
fc <- all_models %>% forecast(new_data = housing_test)
fc
```

```
# A fable: 36 x 10 [1M]
# Key:     .model [3]
   .model     date median_days .mean housing unemploym~1 price~2 price~3 pendi~4
   <chr>     <mth>      <dist> <dbl>   <dbl>       <dbl>   <dbl>   <dbl>   <dbl>
 1 tslm   2021 Jul   N(17, 33)  17.1    2557         4.9     164     988    2391
 2 tslm   2021 Aug   N(18, 32)  18.4    2663         4.4     222     986    2398
 3 tslm   2021 Sep   N(20, 32)  19.8    2809         3.8     232     984    2367
 4 tslm   2021 Oct   N(22, 31)  21.6    2874         3.5     236     916    2414
 5 tslm   2021 Nov   N(21, 31)  21.5    2558         3.2     188     764    2424
 6 tslm   2021 Dec   N(20, 32)  19.8    2035         3.1     166     500    2174
 7 tslm   2022 Jan   N(17, 34)  16.8    1622         3.7     180     332    1848
 8 tslm   2022 Feb   N(17, 34)  17.0    1519         3.5     224     292    1900
 9 tslm   2022 Mar   N(17, 34)  17.0    1597         3.2     162     366    1937
10 tslm   2022 Apr   N(17, 34)  17.5    1965         3.1     140     584    2016
# ... with 26 more rows, 1 more variable: median_price <dbl>, and abbreviated
#   variable names 1: unemployment, 2: price_increased, 3: price_decreased,
#   4: pending_listing
```

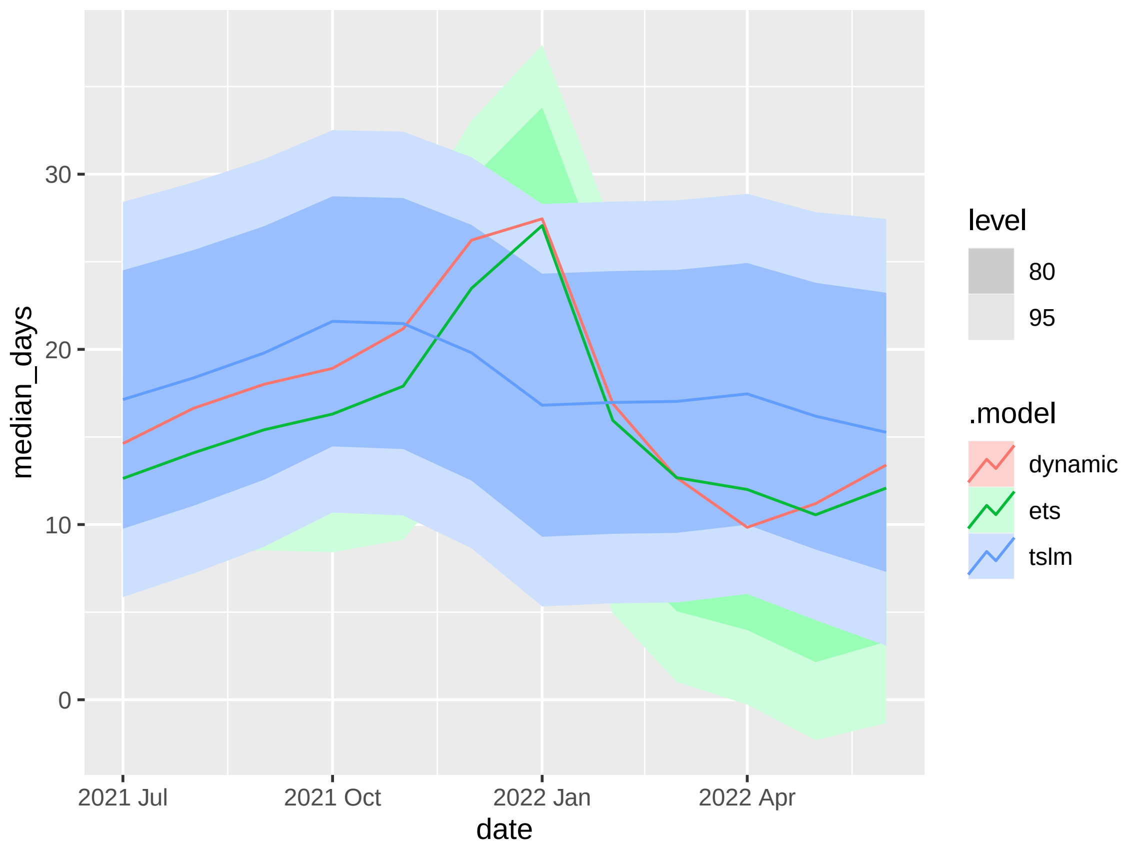
12. Plot forecasts, compute point and distributional accuracy estimates. Which model would you use to forecast `median_days`?

```r
# Plot forecasts
autoplot(housing_train, median_days) +
  autolayer(fc, .mean, series = "Forecast") +
  autolayer(housing_test,
            median_days,
            series = "Test Data",
            colour = "orange") +
  labs(title = "Median Days: Observations and Forecasts",
       x = "Time",
       y = "Median Days") +
  theme_minimal()
```



Median Days: Observations and Forecasts
Median Days
40
20
0
2018 Jan
2020 Jan
2022 Jan
Time
level
80
95
.model
dynamic
ets
tslm


```
autoplot(fc)
```




```r
# Compute point accuracy estimates
fc %>% accuracy(housing_test) %>%
  select(.model, RMSE, ME, MAE)
```

```
# A tibble: 3 x 4
  .model   RMSE       ME   MAE
  <chr>   <dbl>    <dbl> <dbl>
1 dynamic  1.35 -0.00508  1.11
2 ets      2.72  1.40     2.52
3 tslm     4.90 -0.909    3.89
```

```r
# Compute distributional accuracy estimates
fc %>% accuracy(
  housing_test,
  list(crps = CRPS)
)
```

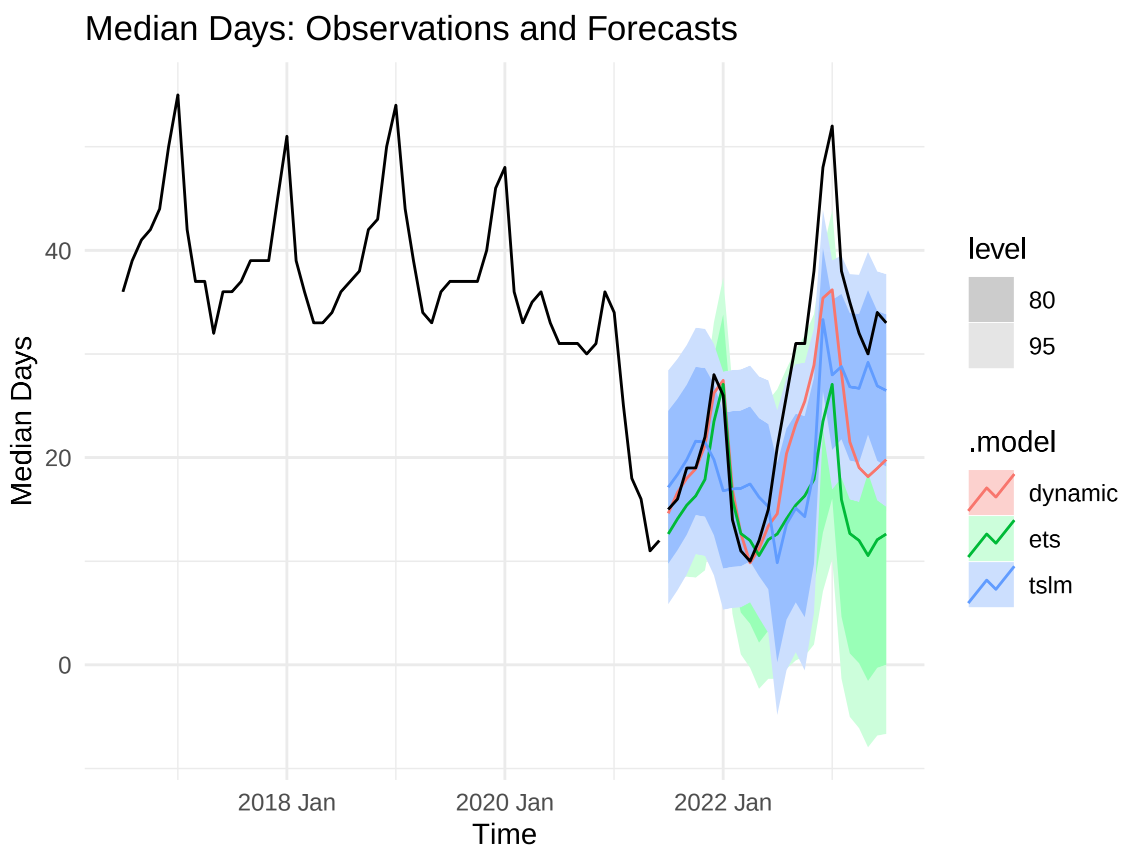
```
# A tibble: 3 x 3
  .model  .type  crps
  <chr>   <chr> <dbl>
1 dynamic Test  0.797
2 ets     Test  1.79
3 tslm    Test  2.81
```

Answer: "Which model would you use to forecast `median_days`?" Among the models provided, the "dynamic" model has the lowest RMSE of approximately 1.351507. The "ets" model has an RMSE of approximately 2.721791, and the "tslm" model has the highest RMSE of approximately 4.897561. Lower RMSE values indicate better accuracy in point forecasting.Among the models, the "dynamic" model has the lowest CRPS of approximately 0.7969838. The "ets" model has a CRPS of approximately 1.7911959, and the "tslm" model has a CRPS of approximately 2.8073965. Lower CRPS values indicate better probabilistic forecasting performance.Based on both the RMSE and CRPS metrics, the "dynamic" model appears to be the best choice for forecasting median_days, as it has the lowest values for both metrics, indicating better accuracy and probabilistic forecasting performance.

13. Load the `housing_validation.csv` file and plot the actual data over the `housing_train` and `housing_test` data. Use the color `"purple"` for the line

You'll need to combine the `housing_test` and `housing_validation` datasets (hint: first create `housing_validation` as a `tsibble`)

```r
library(tsibble)
# Load in the data
housing_validation <- read_csv("~/Desktop/DS Fall 2023/DS adv stats/data/housing_validation.csv")


# Set year and month for validation
housing_validation$date <- yearmonth(housing_validation$date)

# Create tsibble
housing_valid_ts <- housing_validation %>%
  as_tsibble(index = date)

# Create new tsibble (hint: you'll need to use `append_row` and populate the new rows)
combined_ts <-bind_rows(housing_test, housing_valid_ts)

# Forecast using the new combined `housing_test` and `housing_validation` data
fc_combined <- all_models %>% forecast(new_data = combined_ts)

# Plot forecasts
autoplot(housing_train, median_days) +
  autolayer(fc_combined, .mean, series = "Forecast") +
  autolayer(combined_ts,
            median_days,
            series = "Test Data",
            colour = "black") +
  labs(title = "Median Days: Observations and Forecasts",
       x = "Time",
       y = "Median Days") +
  theme_minimal()
```




```
# Compute point accuracy estimates
fc_combined %>% accuracy(combined_ts) %>%
  select(.model, RMSE, ME, MAE)
```

```
# A tibble: 3 x 4
  .model   RMSE    ME   MAE
  <chr>   <dbl> <dbl> <dbl>
1 dynamic  8.16  5.57  6.10
2 ets     14.2  10.5  11.1
3 tslm    10.1   5.61  7.92
```

```
# Compute distributional accuracy estimates
fc_combined %>% accuracy(
  combined_ts,
  list(crps = CRPS)
)
```

```
# A tibble: 3 x 3
  .model  .type  crps
  <chr>   <chr> <dbl>
1 dynamic Test   5.33
2 ets     Test   8.28
3 tslm    Test   5.86
```

14. Using *only* the `housing_validation` data (use your `tsibble`), check the accuracy of your forecasts

```
# Compute point accuracy estimates
fc_combined %>%
  accuracy(housing_valid_ts) %>%
  select(.model, RMSE, ME, MAE)
```

```
# A tibble: 3 x 4
  .model   RMSE    ME   MAE
  <chr>   <dbl> <dbl> <dbl>
1 dynamic  11.2  10.7  10.7
2 ets      19.5  18.9  18.9
3 tslm     13.1  11.6  11.6
```

```
# Compute distributional accuracy estimates
fc_combined %>% accuracy(housing_valid_ts,list(crps = CRPS))
```

```
# A tibble: 3 x 3
  .model  .type  crps
  <chr>   <chr> <dbl>
1 dynamic Test   9.52
2 ets     Test  14.3
3 tslm    Test   8.69
```

15. Based on the updated accuracies, does your choice of model change? Why or why not?

> Answer Based on the updated accuracies, the choice of model will be the tslm model instead of the dynamic model. The "dynamic" model has the lowest RMSE of approximately 11.23714. The "tslm" model has an RMSE of approximately 13.14495, and the "ets" model has the highest RMSE of approximately 19.52407. Lower RMSE values indicate better accuracy in point forecasting.The "dynamic" model has the lowest CRPS of approximately 9.522384. The "tslm" model has a CRPS of approximately 8.687121, and the "ets" model has a CRPS of approximately 14.275650. Lower CRPS values indicate better probabilistic forecasting performance.Based on the updated accuracy metrics, the "dynamic" model is no longer the best choice. The "tslm" model has the lowest RMSE and the lowest CRPS, indicating that it performs better in terms of both point forecasting accuracy and probabilistic forecasting performance. Therefore, the "tslm" model would be the preferred choice for forecasting in this scenario.