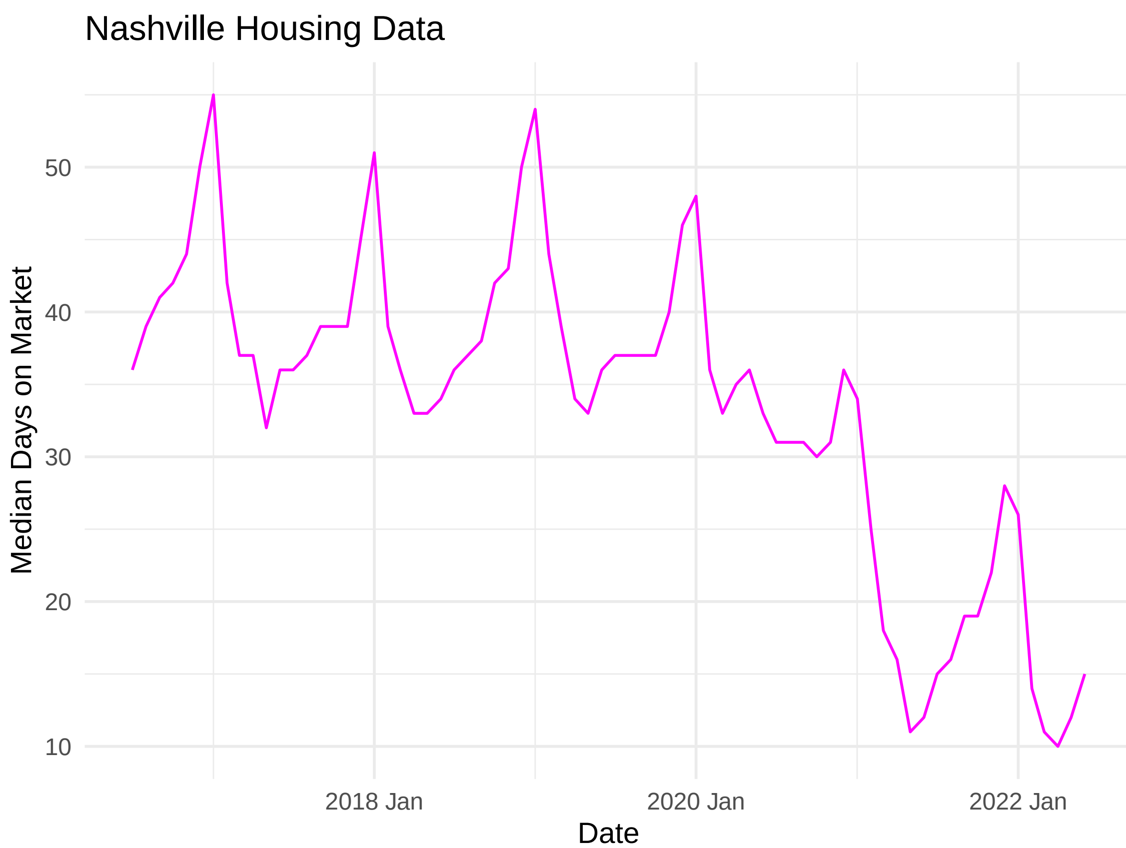
Assignment 5

Rachel Montgomery

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
nashville_housing <- read.csv("nashville_housing.csv")

# Convert date
nashville_housing$date <- yearmonth(nashville_housing$date)

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

# Plot `median_days`
ggplot(housing_ts, aes(x = date, y = median_days)) +
  geom_line(color="magenta") +
  labs(x = "Date", y = "Median Days on Market") +
  labs(title = "Nashville Housing Data")+
  theme_minimal()
```



We can see a downward trend in the housing data, with about yearly seasonality.

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_train)
```

```
[1] "date"             "housing"          "unemployment"     "median_days"
[5] "price_increased" "price_decreased" "pending_listing" "median_price"
```

```
# Fit linear model
fit_tslm <- housing_train %>%
  model(tslm = TSLM(
    median_days ~ housing + unemployment + median_price + price_increased +
```

```r
      price_decreased + pending_listing ))

# 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
housing          7.789e-03  1.666e-03   4.675 2.06e-05 ***
unemployment    -8.188e-01  3.624e-01  -2.260 0.027987 *
median_price     3.861e-05  1.109e-04   0.348 0.729067
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 *
---
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
```

Housing, unemployment, price_decreased, and pending_listing are significant predictors at the 0.05 level.

Approximately 64.84% of the variation in median number of days houses are on the market in Nashville is explained by the predictors in the model.

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

```r
# Fit model with `lm`
fit <- lm(median_days ~ housing + unemployment + median_price + price_increased +
      price_decreased + pending_listing, data=housing_train)

# Check for multicolinearity using `VIF`

# VIF
regclass::VIF(fit)
```

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

```r
# Coefficients
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." Housing, median_price, and price_decreased all have VIF > 10. Out of the three, I'm keeping price_decreased because it has the lowest VIF out them, and it was significant at the 0.001 level.

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

```r
# Re-fit model with `lm`

fit2 <- lm(median_days ~ unemployment  + price_increased +
      price_decreased + pending_listing, data=housing_train)

# Check for multicolinearity using `VIF`
regclass::VIF(fit2)
```

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

There is no longer any variables with `VIF` > 10?.

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

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

fit_tslm2 <- housing_train %>%
  model(tslm = TSLM( median_days ~ housing + unemployment +
                    price_decreased + pending_listing))

## report fit
report(fit_tslm2)
```

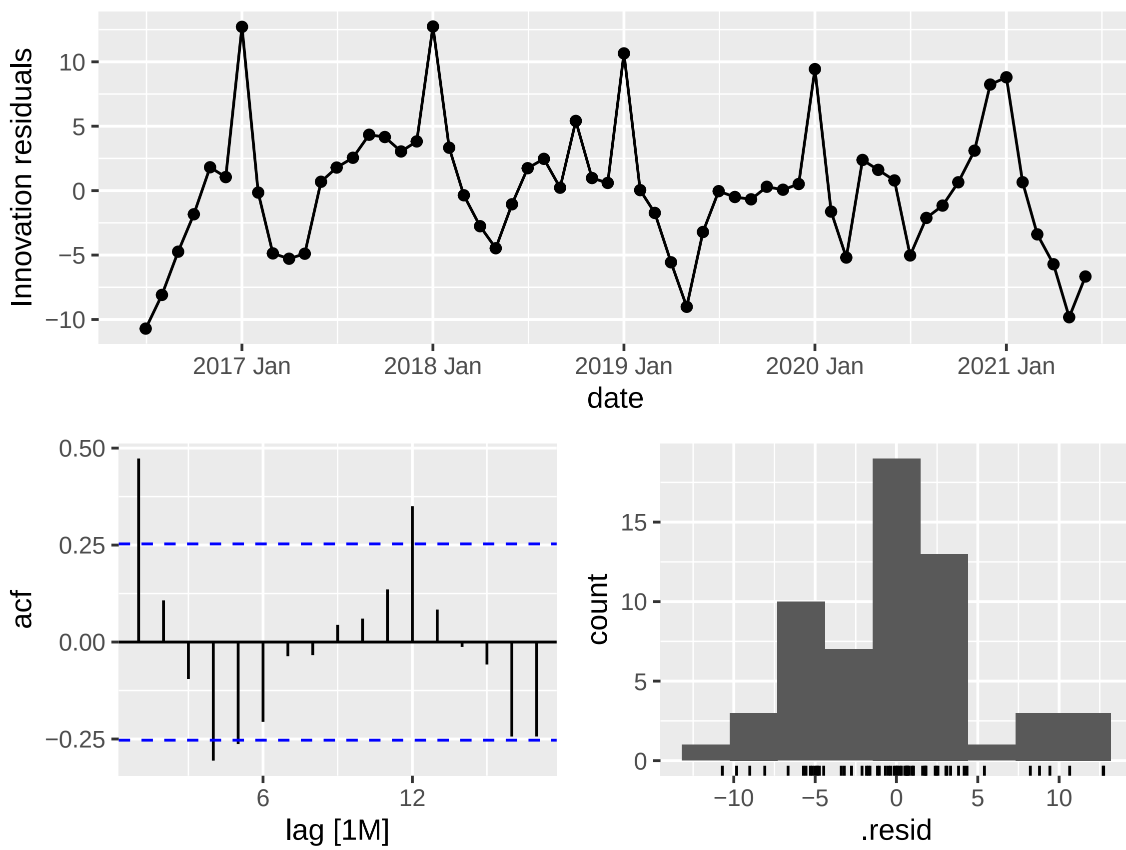
```
Series: median_days
Model: TSLM

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
```

```
housing          0.007371   0.001113   6.621 1.57e-08 ***
unemployment    -0.790912   0.341492  -2.316  0.02431 *
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
```

As we can see in the output, all of the predictors are significant at the 0.05 level.

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

```
# Plot residuals
fit_tslm2 %>%
  gg_tsresiduals()
```




```
# Perform Ljung-Box test
# Set `lag = 12` (notice seasonal pattern in ACF)
# (remember to adjust dof = number of coefficients)
```

```r
fit_tslm2 %>% augment() %>% features(.innov, ljung_box, lag = 12, dof = 4)
```

```
# A tibble: 1 x 3
  .model lb_stat  lb_pvalue
  <chr>    <dbl>      <dbl>
1 tslm      40.8 0.00000232
```

Answer: "Are the residuals significantly different from white noise?"

Using the results of the Ljung-Box test, because our p-value is less than 0.05, we can conclude 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.

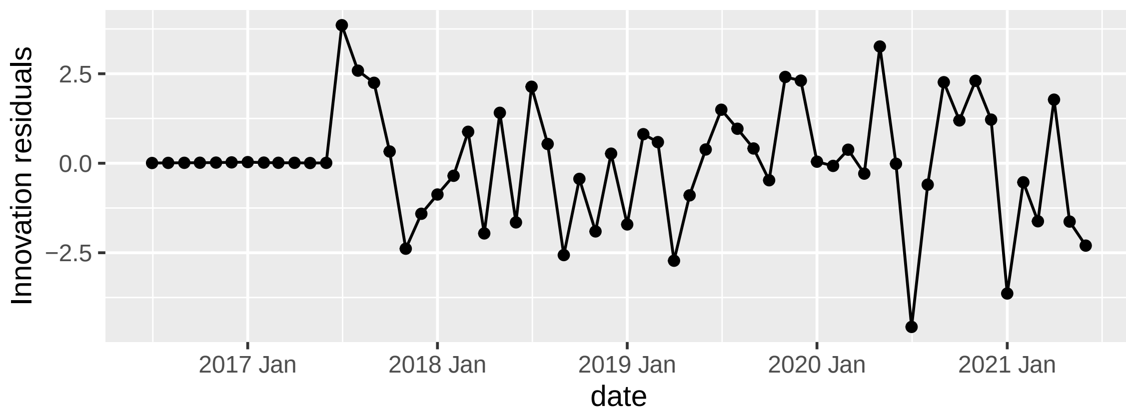
```r
# Fit TSLM with ARIMA errors

fit_dynamic <- housing_train %>%
  model(dynamic = ARIMA(median_days ~ housing + unemployment +
                    price_decreased + pending_listing ))

# Report fit
report(fit_dynamic)
```

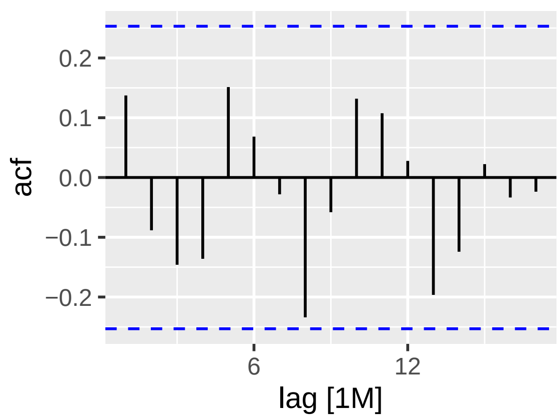
```
Series: median_days
Model: LM w/ ARIMA(0,0,0)(1,1,0)[12] errors

Coefficients:
         sar1  housing  unemployment  price_decreased  pending_listing
      -0.6520   0.0034        0.4923          -0.0016           0.0032
s.e.   0.1212   0.0007        0.1360           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
```

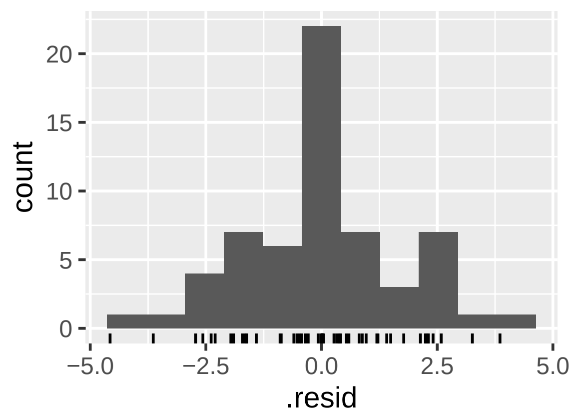
Answer: "Comment on whether any differencing was used." Yes, a differencing of 1 was used.

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

```r
# Plot residuals
fit_dynamic %>%
  gg_tsresiduals()
```








```
# Perform Ljung-Box test
# Set lag based on seasonal lag in from `ARIMA` fit
# (remember to adjust dof = number of coefficients)

fit_dynamic %>%
  augment() %>%
  features(.innov, ljung_box, lag = 12, dof = 4) ## is dof=4 correct?
```

```
# A tibble: 1 x 3
  .model  lb_stat lb_pvalue
  <chr>     <dbl>     <dbl>
1 dynamic    12.6     0.125
```

Answer: "Are the residuals significantly different from white noise?"

Using the results of the Ljung-Box test, because our p-value is greater than 0.05, we can conclude that the residuals are not significantly different from white noise, which is what we are hoping for. If we are able to show that the residual errors of the fitted model are white noise, it means the model has done a great job of explaining the variance in the dependent variable.

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

```r
# Fit model with `ETS`
fit_ets <- housing_train %>%
  model(ETS(median_days))

# 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
```

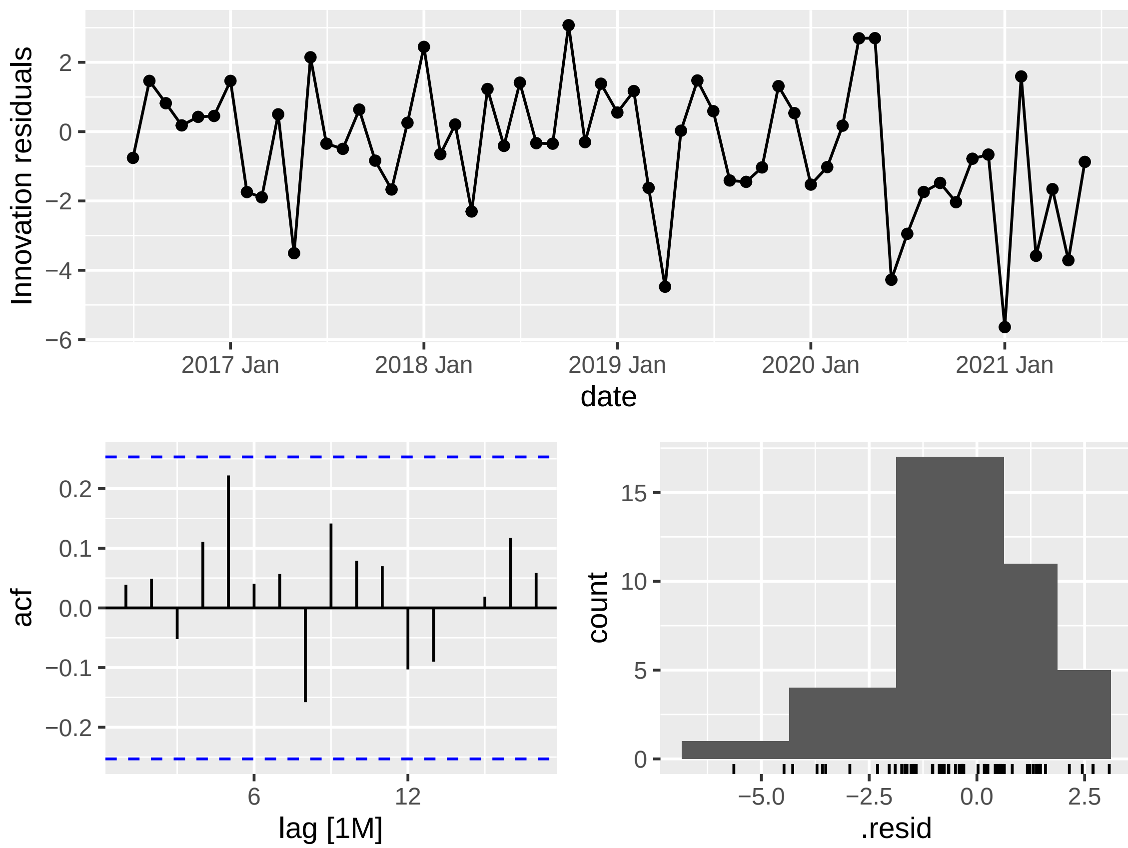
Answer: "Interpret the `alpha` and `gamma` parameters."

- Alpha: 0.9063522 (Level Smoothing Parameter)
  - Alpha indicates the rate at which the ETS model updates its estimate of the level (the long-term average) of the time series.
  - Because alpha is close to 1, we can conclude that the ETS model assigns relatively high importance to recent observations when estimating the level of the median_days time series.
- Gamma : 0.0001264875 (Seasonal Smoothing Parameter)
  - Gamma indicates the rate at which the ETS model updates its estimate of the seasonal component of the time series.
  - Because Gamma is close to 0, we can conclude that the ETS model assumes that the seasonal pattern in the median_days time series is relatively stable and does not change rapidly from one season to the next.

In summary, the ETS model has a high alpha value, indicating that it is giving more weight to recent data for estimating the level, and a very small gamma value, suggesting a stable seasonal pattern.

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

```r
# Plot residuals
fit_ets %>%
  gg_tsresiduals()
```




```r
# 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?"

Using the results of the Ljung-Box test, because our p-value is greater than 0.05, we can conclude that the residuals are not significantly different from white noise, which is what we are hoping for.

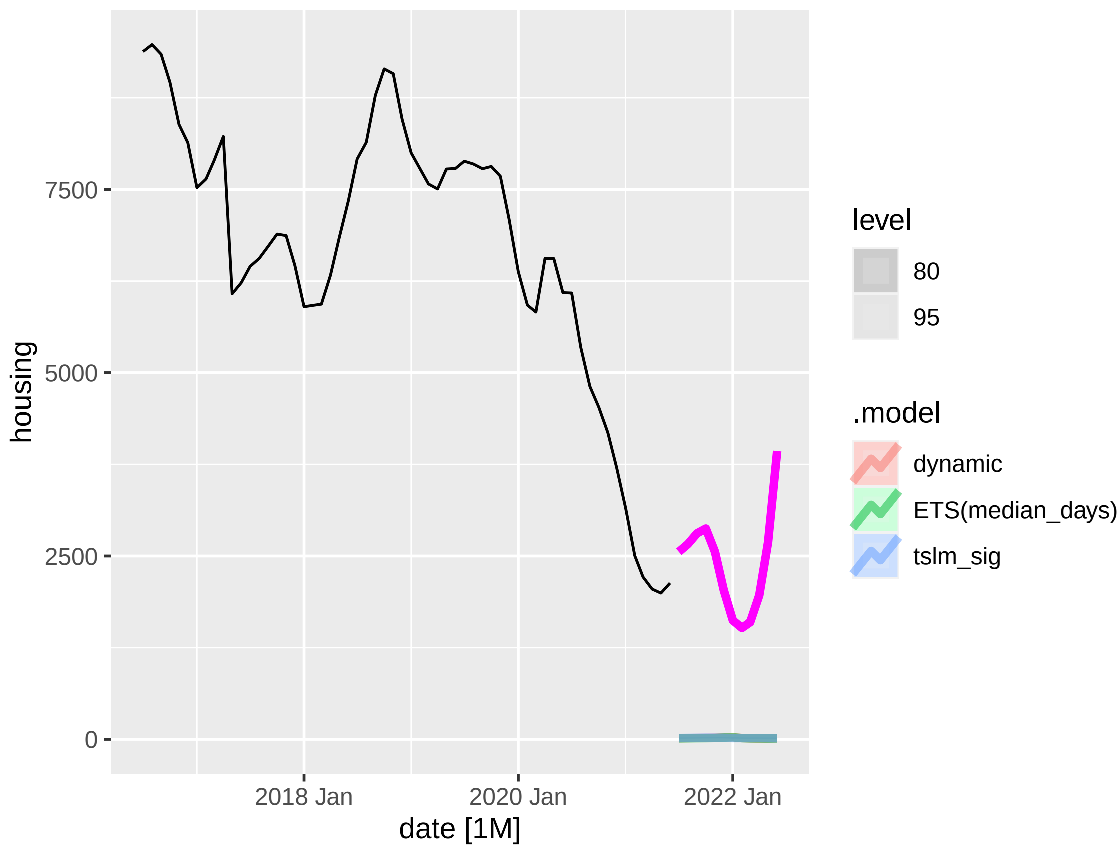
11. Combine all models and forecast using `housing_test` data

```r
# Combine all models


all_models <- housing_train %>%
```

```r
  model(
    tslm_sig = TSLM( median_days ~ housing + unemployment +
                    price_decreased + pending_listing),

     ETS(median_days),

    dynamic = ARIMA(median_days ~ housing + unemployment +
                    price_decreased + pending_listing)
  )

# Forecast models
fc_all_models <- all_models %>%
  forecast(new_data = housing_test)
```

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

```r
# Plot forecasts
housing_train %>%
  autoplot(housing) +
  autolayer(fc_all_models, alpha = 0.5, size = 1.5) +
  geom_line(
    data = housing_test,
    aes(y = housing),
    color = "magenta",
    size = 1.5)
```




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

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

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

```
# A tibble: 3 x 4
  .model           .type winkler  crps
  <chr>            <chr>   <dbl> <dbl>
1 ETS(median_days) Test    19.1  1.79
2 dynamic          Test     7.64 0.797
3 tslm_sig         Test    22.7  2.81
```

Answer: "Which model would you use to forecast `median_days`?"

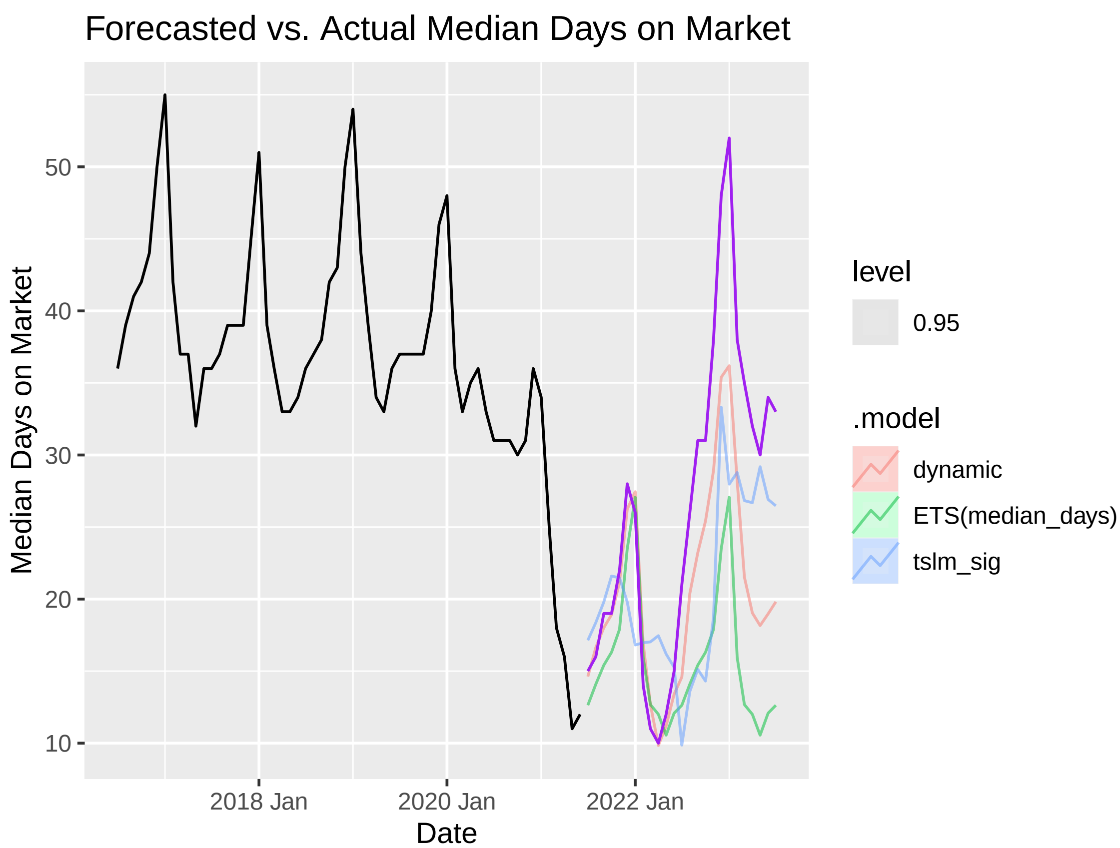
Based on these performance metrics, the “Dynamic” model appears to be the best choice for forecasting ‘median_days’.

It has the lowest RMSE, a close-to-zero ME, the lowest MAE,andthe lowest values for both “winkler” and “crps” metrics, indicating better overall forecasting accuracy.

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
# Load in the data
housing_validation <- read_csv("housing_validation.csv")

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

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

# Create new tsibble (hint: you'll need to use `append_row` and populate the new rows)
#combined_data <- append_row(housing_test, housing_validation_ts) ##append didn't work
combined_data <- bind_rows(housing_test, housing_validation_ts)

# Fit an ETS model to the combined data
fit <- combined_data %>%
  model(ETS(median_days))

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

#Plot forecasts
# forecasted values with the actual values
housing_train %>%
  autoplot(median_days, color="black") +
  autolayer(forecasted_values, alpha = 0.5, level= 0.95) +
  geom_line(
    data = combined_data, aes(y = median_days),
    color = "purple") +
  labs(
  title = "Forecasted vs. Actual Median Days on Market",
  x = "Date",
  y = "Median Days on Market")
```




```r
# Point estimates
fc_all_models %>%
  accuracy(combined_data)
```

```
# A tibble: 3 x 10
  .model           .type        ME  RMSE   MAE     MPE  MAPE  MASE RMSSE  ACF1
  <chr>            <chr>     <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
1 ETS(median_days) Test    1.40     2.72  2.52   6.15  15.0    NaN   NaN 0.576
2 dynamic          Test   -0.00508  1.35  1.11  -0.696  6.90   NaN   NaN 0.383
3 tslm_sig         Test   -0.909    4.90  3.89 -13.9   25.1    NaN   NaN 0.528
```

```r
# Distributional estimates
fc_all_models %>% accuracy(
  combined_data,
  list(winkler = winkler_score, crps = CRPS))
```

```
# A tibble: 3 x 4
  .model           .type winkler  crps
  <chr>            <chr>   <dbl> <dbl>
1 ETS(median_days) Test    19.1  1.79
2 dynamic          Test     7.64 0.797
3 tslm_sig         Test    22.7  2.81
```

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

```r
# Compute point accuracy estimates

accuracy_metrics <- accuracy(forecasted_values, housing_validation_ts)

# Print the accuracy metrics
print(accuracy_metrics)
```

```
# A tibble: 3 x 10
  .model          .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
  <chr>           <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 ETS(median_days) Test   18.9  19.5  18.9  54.7  54.7   NaN   NaN 0.637
2 dynamic         Test   10.7  11.2  10.7  31.1  31.1   NaN   NaN 0.557
3 tslm_sig        Test   11.6  13.1  11.6  33.9  33.9   NaN   NaN 0.545
```

```r
# Compute CRPS estimates
forecasted_values %>%
  accuracy(housing_validation_ts,list(crps = CRPS))
```

```
# A tibble: 3 x 3
  .model          .type  crps
  <chr>           <chr> <dbl>
1 ETS(median_days) Test  14.3
2 dynamic         Test   9.52
3 tslm_sig        Test   8.69
```

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

I still choose dynamic because it consistently has the lowest values, which is what we are looking for.