FORECASTING AND PREDICTIVE MODELING

LECTURE 10

Lect. PhD. Oneț-Marian Zsuzsanna

Babeș - Bolyai University
Computer Science and Mathematics Faculty

2024 - 2025

In Lecture 9

- Time series regression models

Today

- Exponential smoothing
- ARIMA models

Exponential smoothing

- Exponential smoothing includes methods that consider weighted averages of past observations to produce forecasts, with the weights decaying exponentially as the observations get older.
- It generates reliable forecasts, quickly and for a wide range of time series

Simple exponential smoothing (SES)

- Suitable when there is no trend or season in data.
- For such time series up until now we only considered the naive and the mean benchmark methods.
- If you remember, the naive method assumed that all future values are equal to the last historical value. This is kind of a weighted average, when all weight is given to the last observation.
- Also the mean method considered that all future values are equal to the mean of the historical data. This is also kind of a weighted average where all historical data has the same weight.

- In many cases we want something in between. Consider historical data, but not all, and not equally. This is the idea of simple exponential forecasting:

$$\hat{y}_{T+1|T} = \alpha * y_T + \alpha * (1-\alpha) * y_{T-1} + \alpha * (1-\alpha)^2 * y_{T-2} + ...$$

- where α is between 0 and 1 and is called the *smoothing parameter*.
- The above equation tells us how to estimate $\hat{y}_{T+1|T}$. For other future time steps, the forecast is equal to this value (we have a flat forecast function).

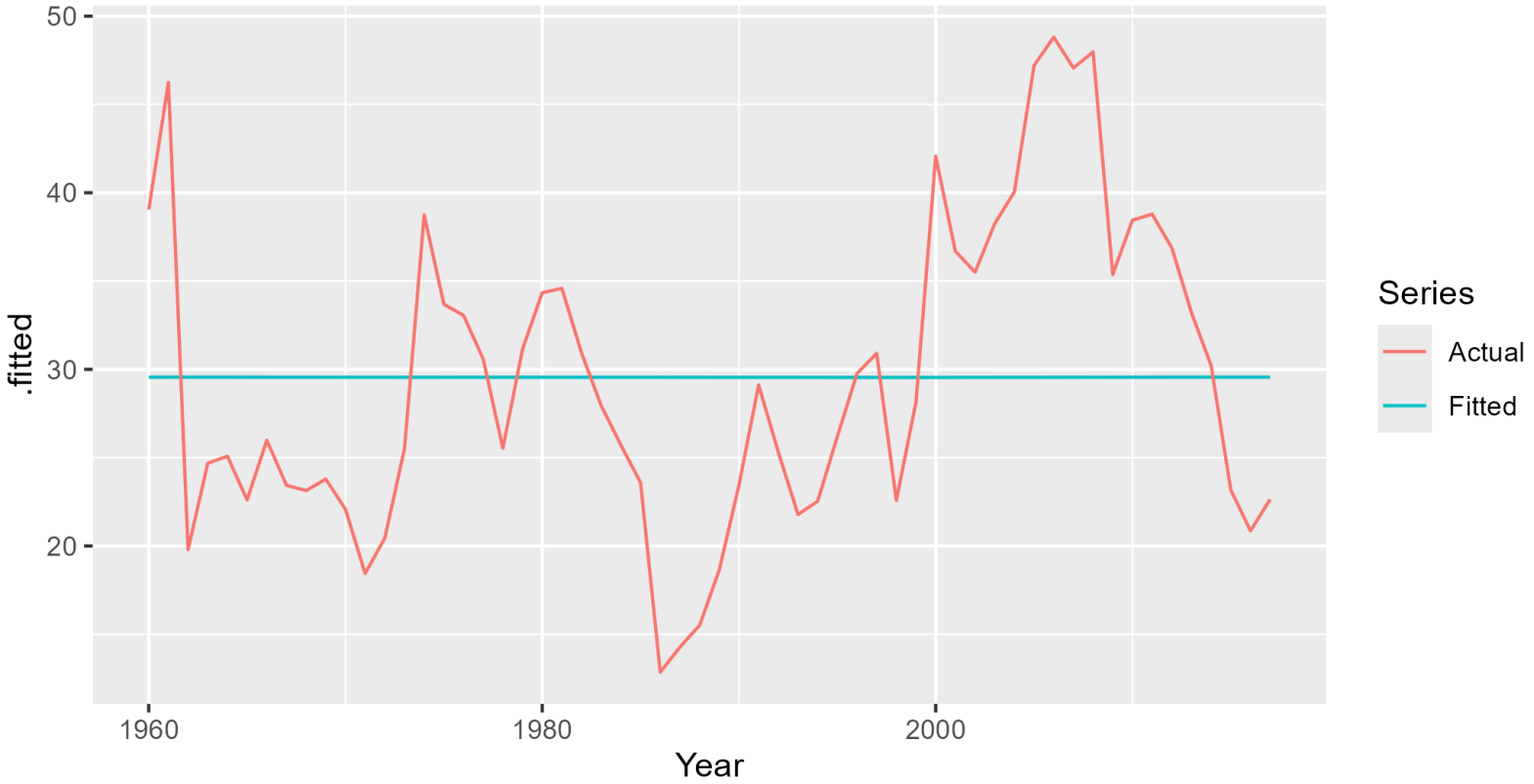
- Let's see a few examples of how the first 6 weights look like, for different values of α

α	0.2	0.4	0.6	0.8
y_t	0.2	0.4	0.6	0.8
y_{t-1}	0.16	0.24	0.14	0.16
y_{t-2}	0.128	0.144	0.096	0.032
y_{t-3}	0.1024	0.0864	0.0384	0.0064
y_{t-4}	0.0819	0.0518	0.0154	0.0013
y_{t-5}	0.0655	0.0311	0.0061	0.0003

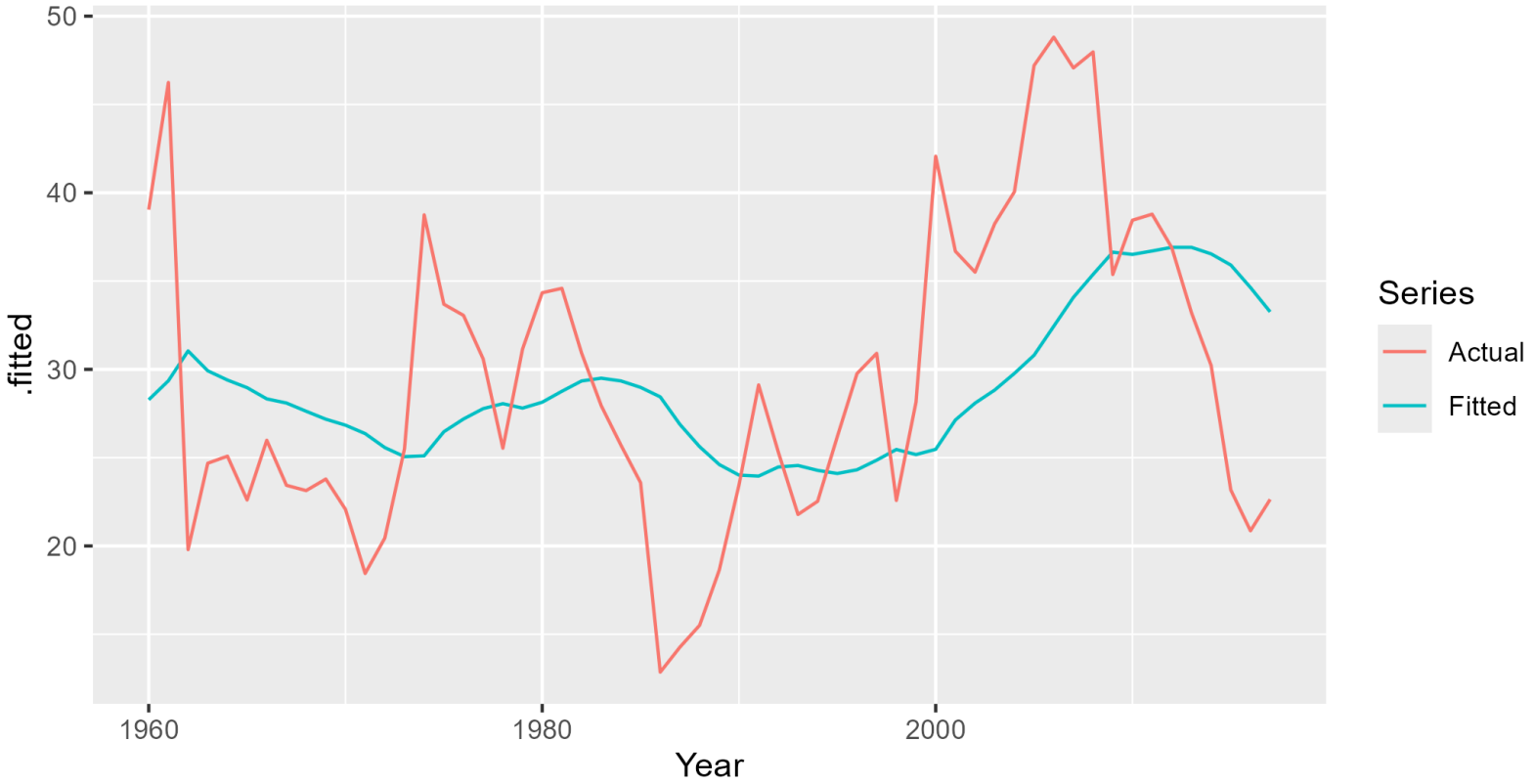
- Obviously, the weights are sensitive of the value of α: when alpha is small, distant observations still get a decent weight, while in case of a larger α the more recent observations get high weights (and the weights quickly decay to close to 0).

- Let us see how the value of α influences the fitted values.

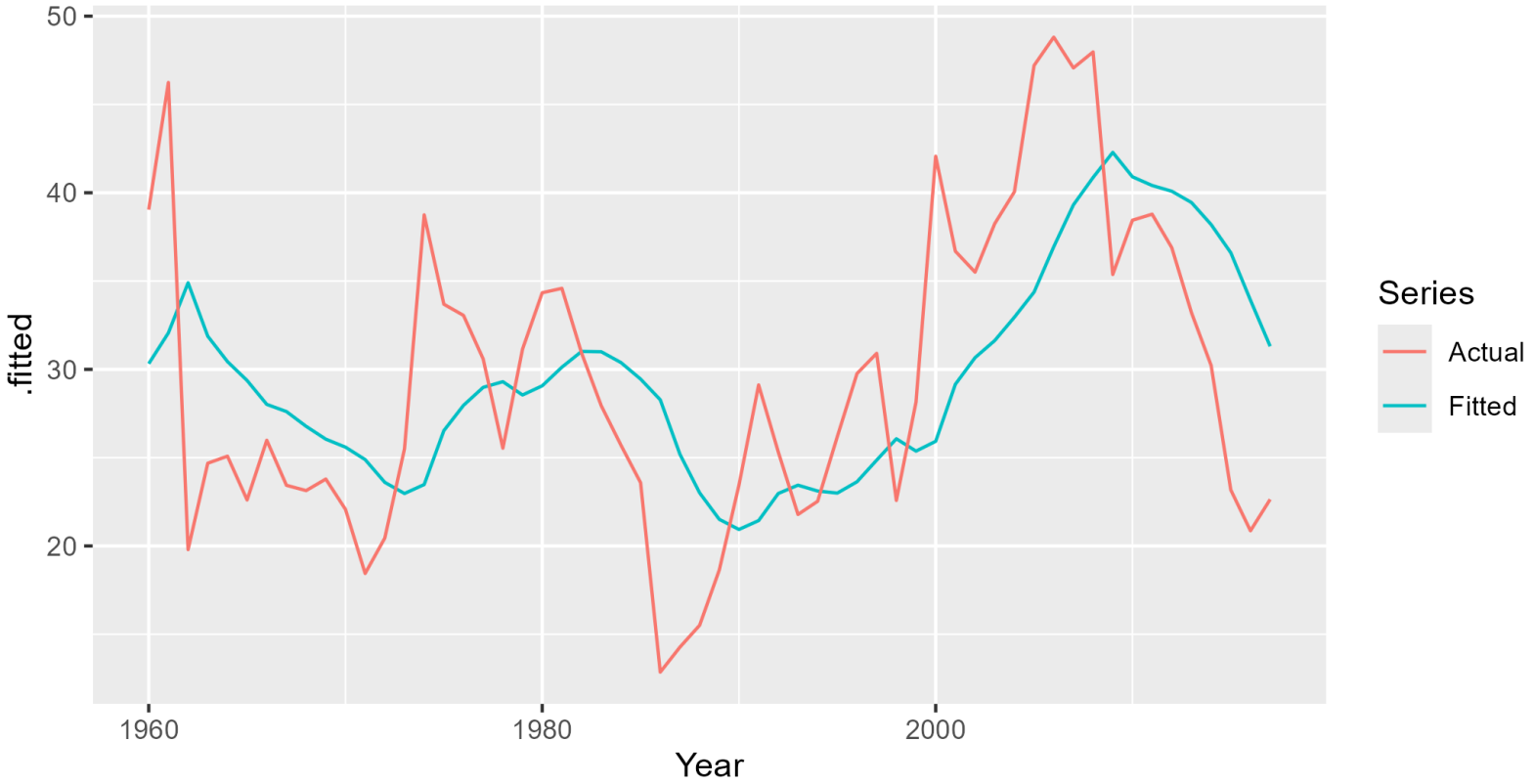
- Actual data and fitted value for $\alpha = 0.0001$



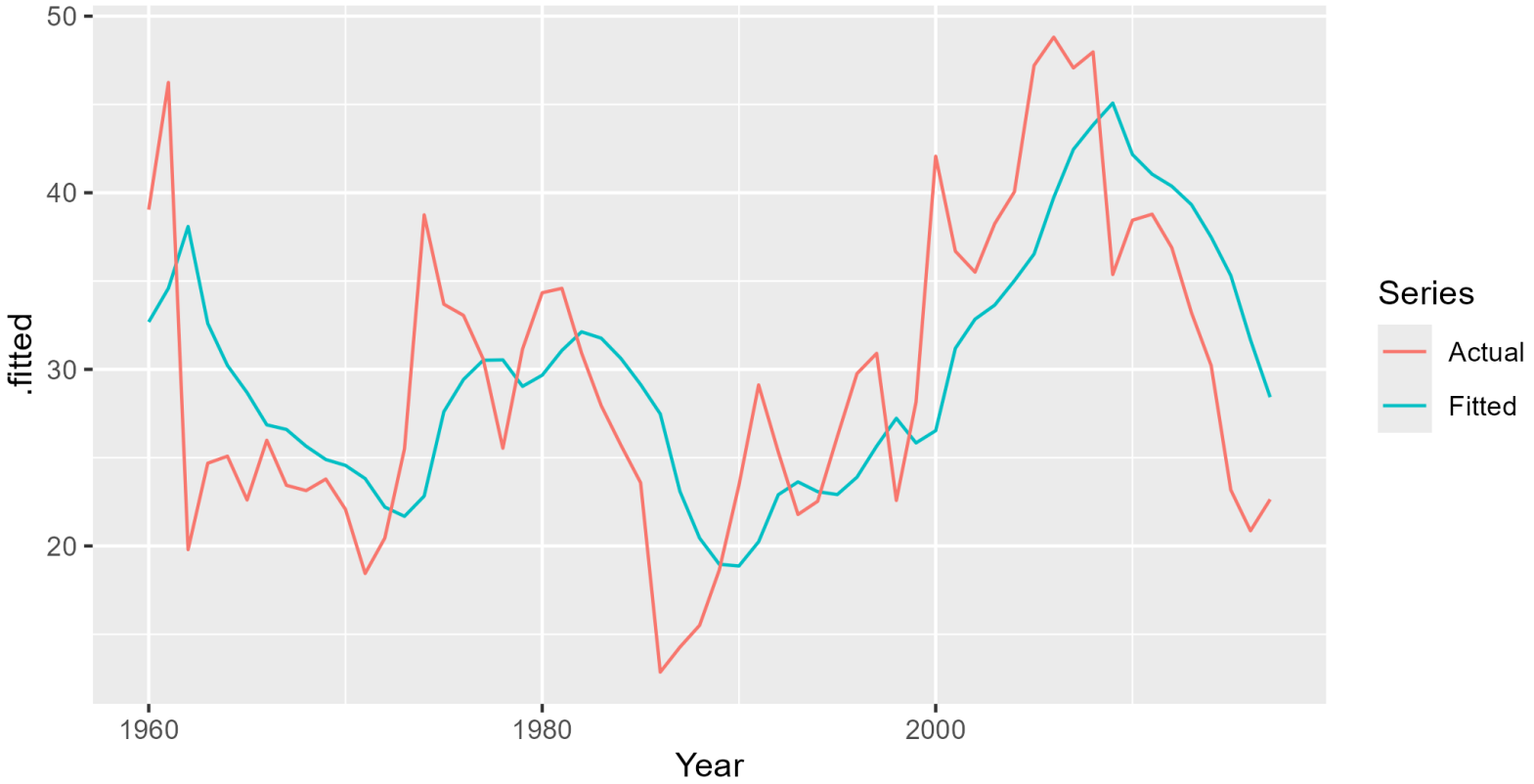
- Actual data and fitted value for $\alpha = 0.1$



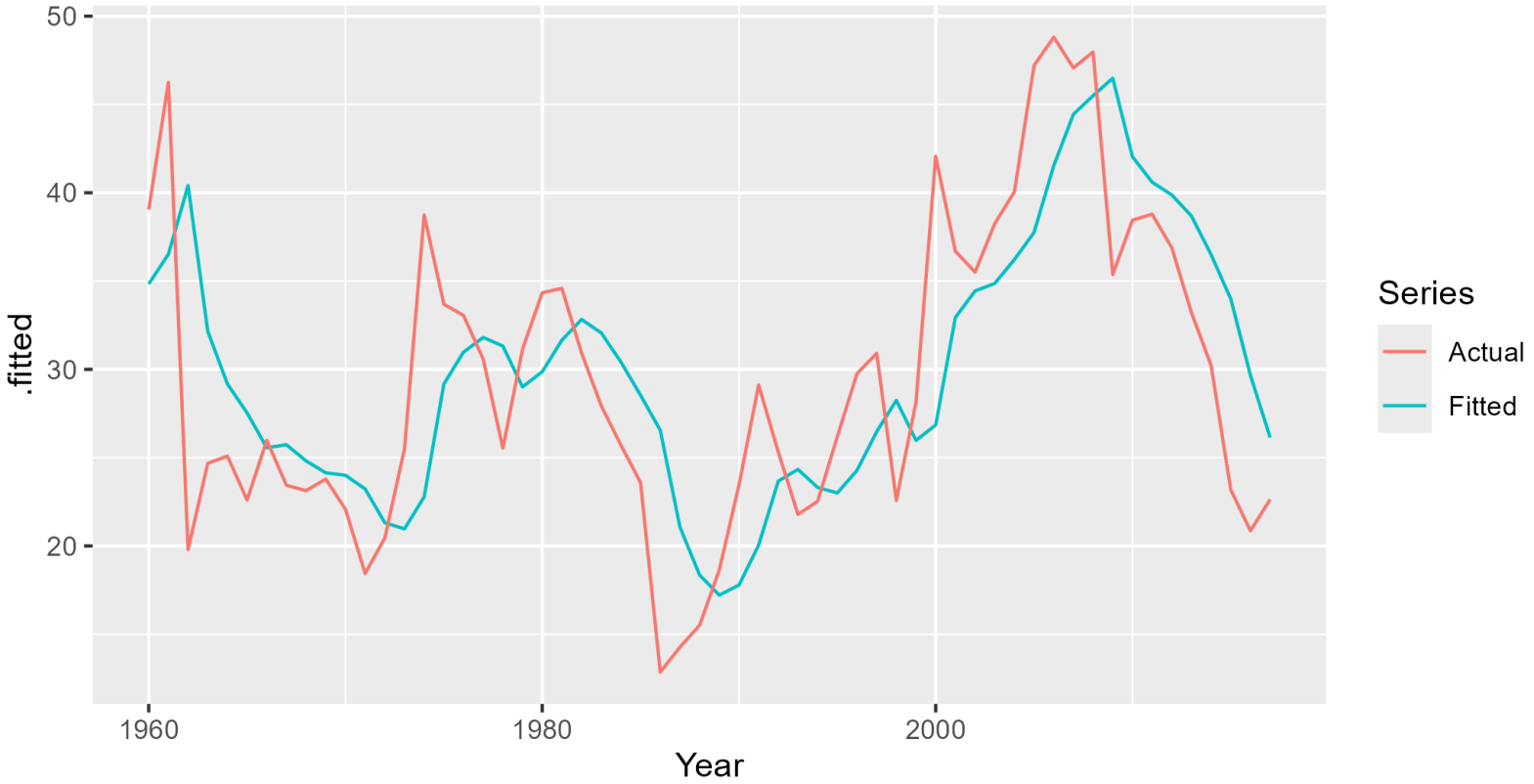
- Actual data and fitted value for $\alpha = 0.2$



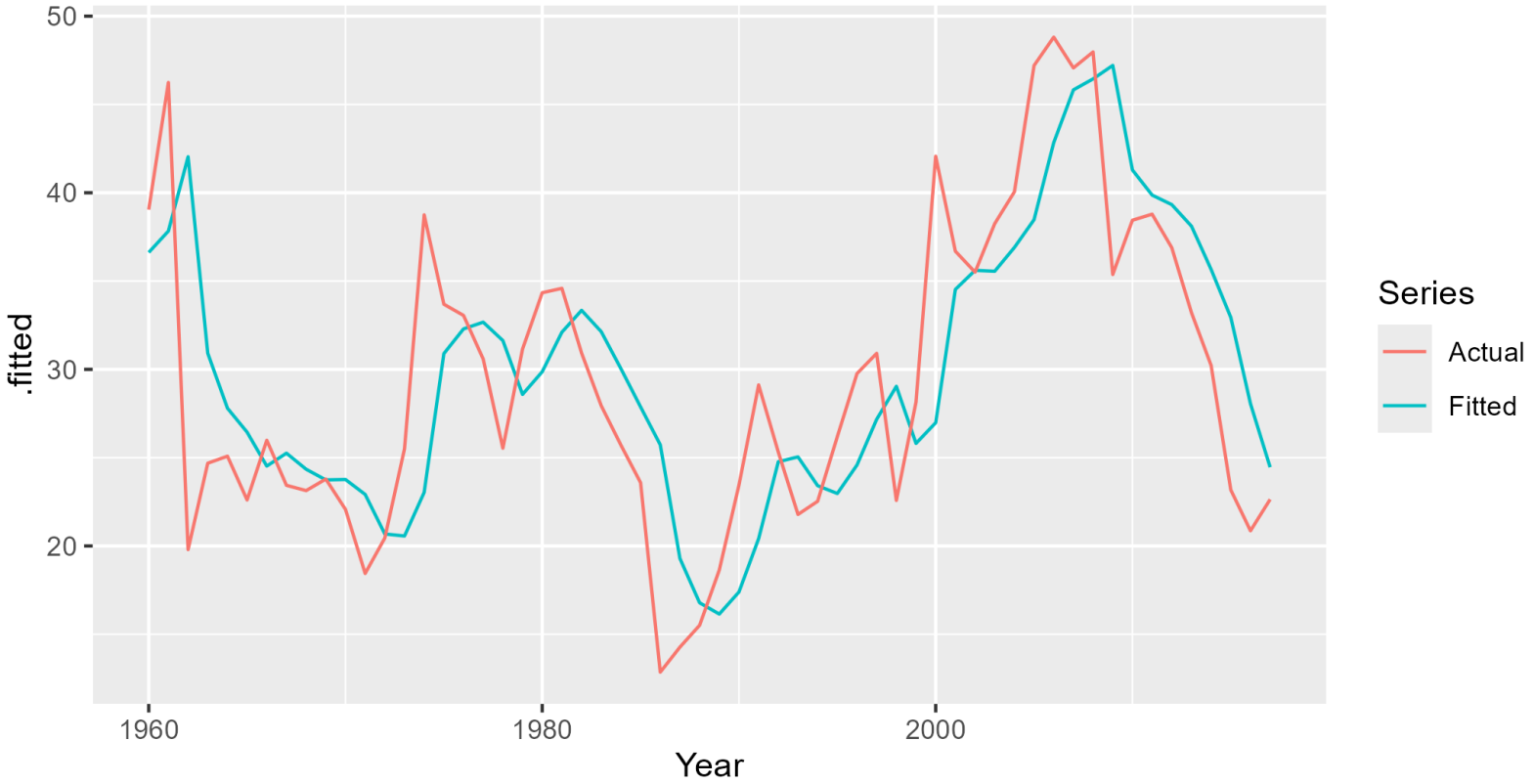
- Actual data and fitted value for $\alpha = 0.3$



- Actual data and fitted value for $\alpha = 0.4$



- Actual data and fitted value for $\alpha = 0.5$

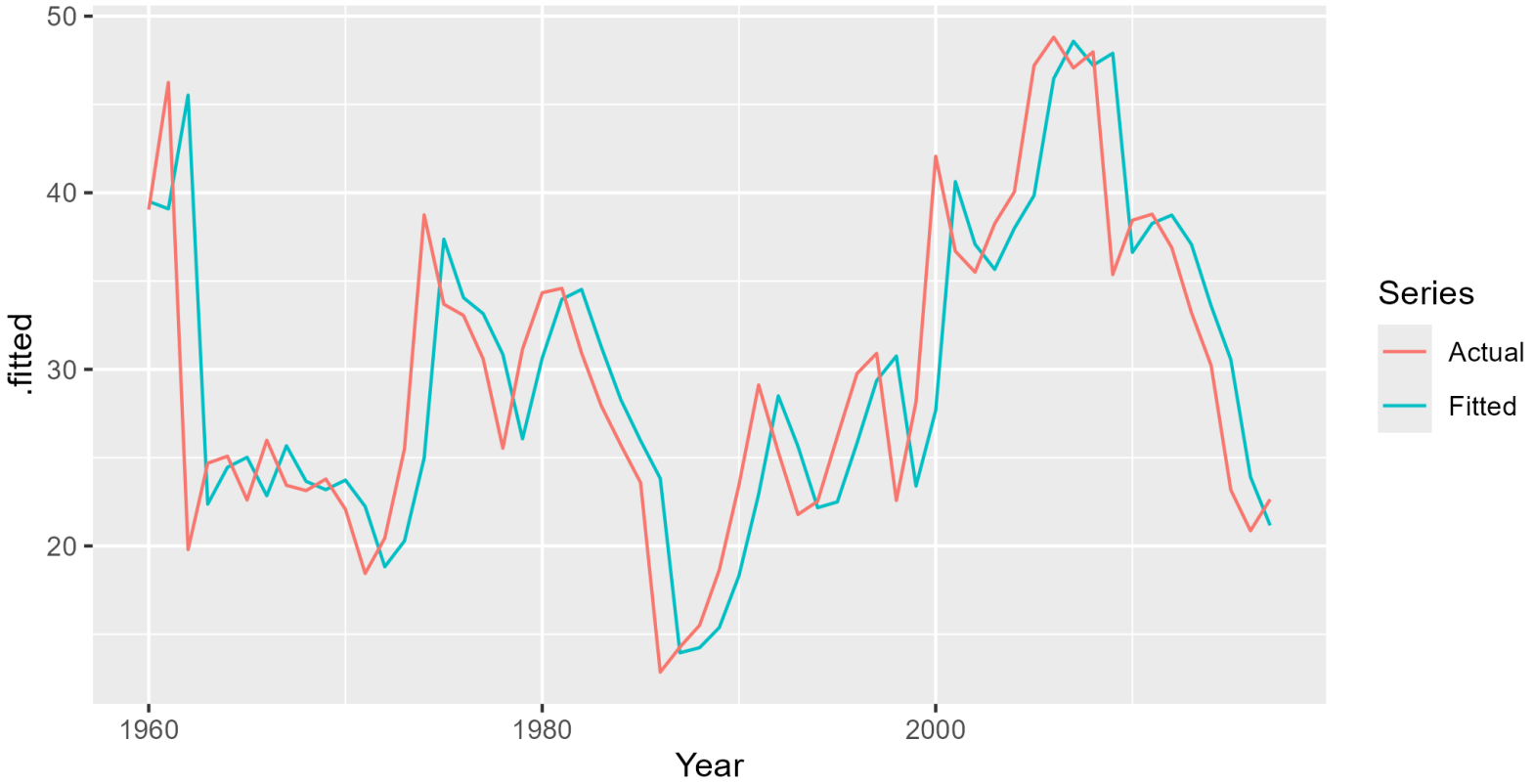


- Actual data and fitted value for $\alpha = 0.6$

- Actual data and fitted value for $\alpha = 0.7$

- Actual data and fitted value for $\alpha = 0.8$

- Actual data and fitted value for $\alpha = 0.9$



- Actual data and fitted value for $\alpha = 0.999$

- We can derive the above expression using the *weighted average form*. According to this form, the forecast at time $T+1$ is the simple weighted average of the most recent observation (y_t) and the previous forecast ($\hat{y}_{T|T-1}$)

$$\hat{y}_{T+1|T} = \alpha * y_T + (1-\alpha) * \hat{y}_{T|T-1}$$

- where α is between 0 and 1.
- If we write the fitted values (one-step forecasts of the training data) accordingly, we will have:

$$\hat{y}_{t+1|t} = \alpha * y_t + (1-\alpha) * \hat{y}_{t|t-1}$$

- for all t between 1 and T.

- The first value that we can forecast with this model is $\hat{y}_{2|1}$, but it needs the fitted value for $t = 0$, which does not exist. We will denote it by l_0 (it is the level of the time series - the average value for a specific time period - and when doing actual computations it will need to be estimated)

$$\hat{y}_{2|1} = \alpha * y_1 + (1 - \alpha) * l_0$$

$$\hat{y}_{3|2} = \alpha * y_2 + (1 - \alpha) * \hat{y}_{2|1}$$

$$\hat{y}_{4|3} = \alpha * y_3 + (1 - \alpha) * \hat{y}_{3|2}$$

...

$$\hat{y}_{T+1|T} = \alpha * y_T + (1 - \alpha) * \hat{y}_{T|T-1}$$

- Now, if we start substituting the equations, we will get:

$$\hat{y}_{3|2} = \alpha * y_2 + (1 - \alpha) * a * y_1 + (1 - \alpha)^2 * l_0$$

$$\hat{y}_{4|3} = \alpha * y_3 + (1-\alpha) * \alpha * y_2 + (1-\alpha)^2 * \alpha * y_1 + (1-\alpha)^3 * l_0$$

...

$$\hat{y}_{T+1|T} = \sum_{j=0}^{T-1} \alpha * (1-\alpha)^j * y_{T-j} + (1-\alpha)^T * l_0$$

- where the last term becomes very small for sufficiently large T.

- Another way of representing the equations is the *component form*, which will be more useful later, when we start adding components for trend and season.
- According to the component form, in case of simple exponential smoothing there is only one component: the level of the time series l_t.
- In this representation the data is described by a *forecast equation* and a *smoothing equation*.
- The *forecast equation* is:

$$\hat{y}_{t+h|t} = l_t$$

- while the *smoothing equation* is:

$$l_t = \alpha * y_t + (1 - \alpha) * l_{t-1}$$

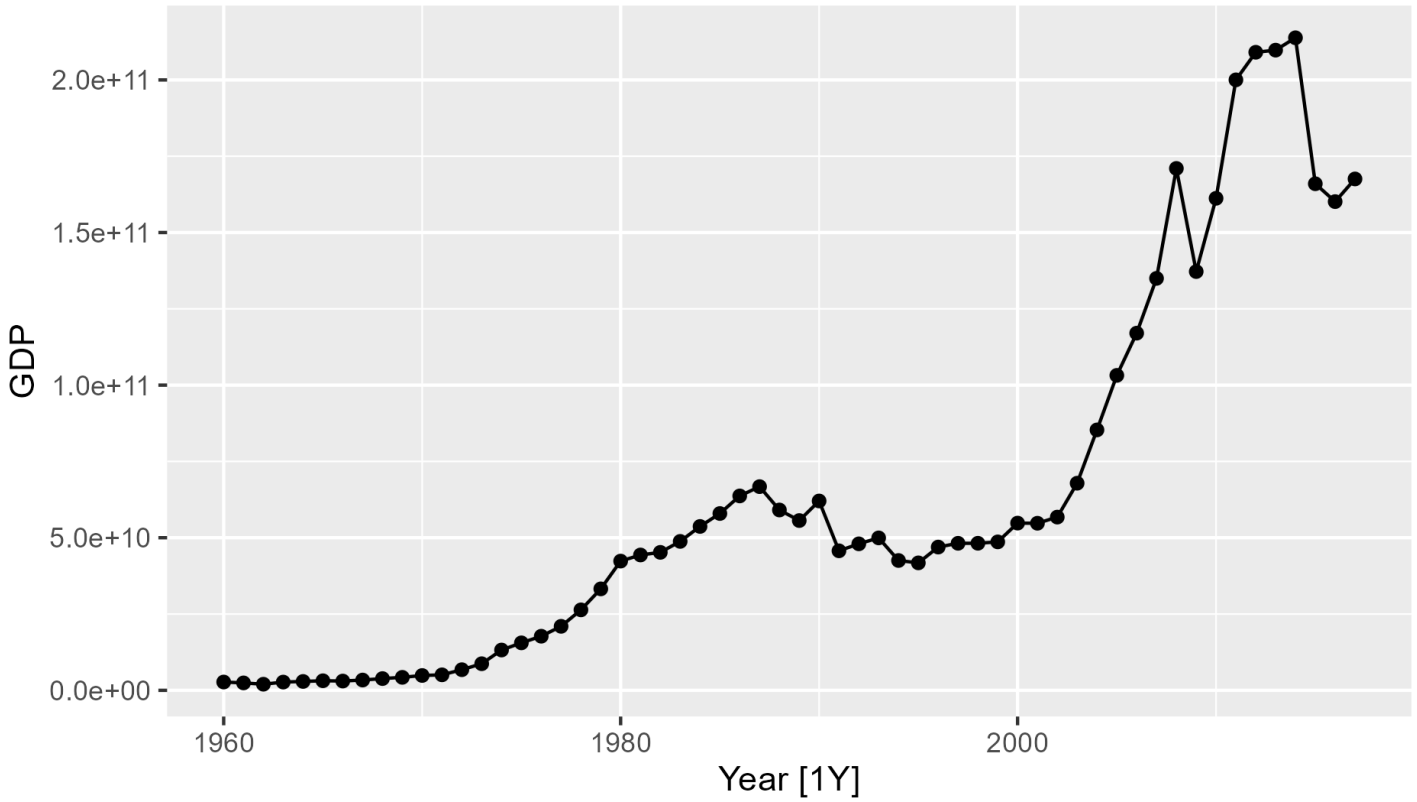
- For simple exponential forecasting there are two parameters that need to be set: the value of α and the value l_0.
- They can be set by the forecaster based on experience, or they can be estimated from the observed data, using the same approach which was used for regression: the minimization of the sum of squared errors.

Example I

- Consider the exports of Algeria from the global economy data set. It seems like there is no season and no trend .

```
algeria_economy <- global_economy |>
  filter(Country == "Algeria")
algeria_economy |> autoplot(Exports)  +
  geom_point()

algeria_model <- algeria_economy |>
  model(ETS(Exports ~ error("A") + trend("N") + season("N")))
algeria_model |>
  report()
```

Example II

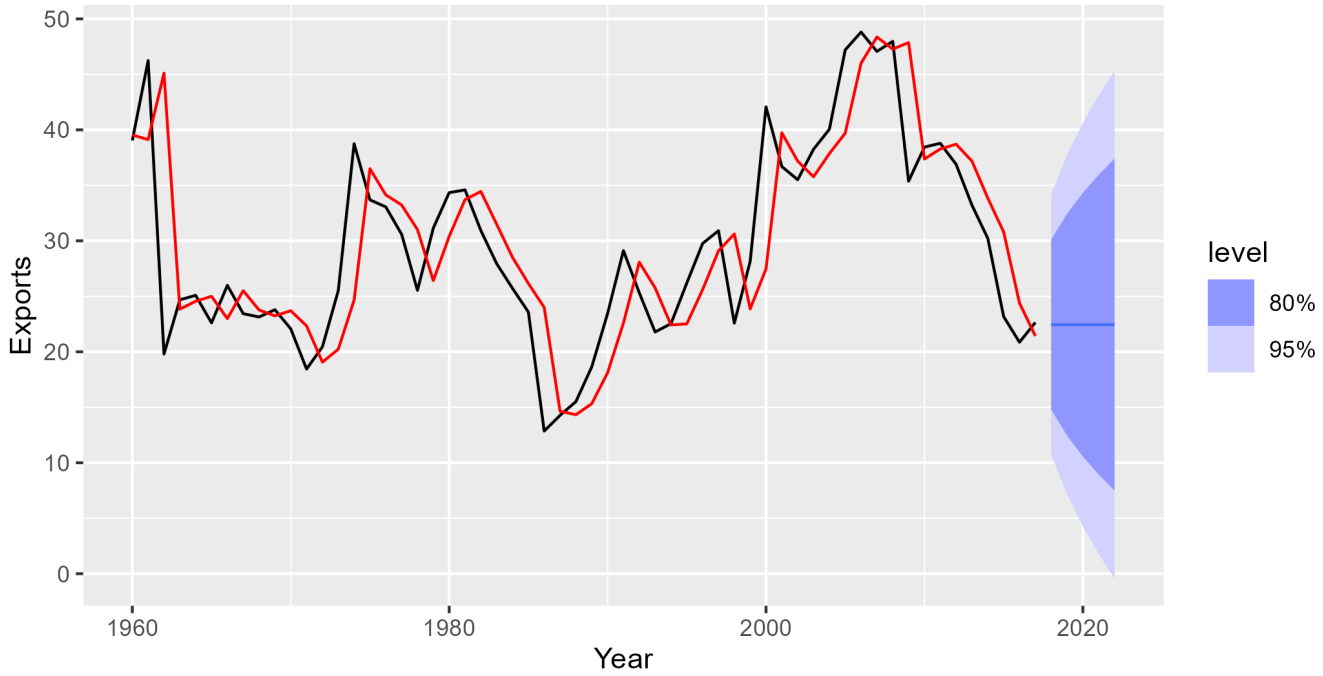


Example III

```
Series: Exports
Model: ETS(A,N,N)
  Smoothing parameters:
    alpha = 0.8399875

  Initial states:
   l[0]
 39.539

  sigma^2:  35.6301

     AIC     AICc      BIC
446.7154 447.1599 452.8968
```

```
algeria_model |>
  tidy() #prints parameters in a nicer manner

algeria_forecast <- algeria_model |>
  forecast(h = 5)
algeria_forecast

algeria_forecast |>
  autoplot(algeria_economy) +
  geom_line(mapping = aes(y =.fitted), data = augment(algeria_model), col = "red
    ")
```

Example IV



- The *report* (or *tidy*) function tells us that the estimated value for α is 0.84 and the value of l_0 is 39.54.
- Using these parameter values, the values of the first few observations (39.043, 46.245, 19.793, ...) and the previous equations we can compute the fitted values:

$$\hat{y}_{2|1} = 0.84 * 39.043 + 0.16 * 39.54 = 39.123$$

$$\hat{y}_{3|2} = 0.84 * 46.245 + 0.16 * 39.12 = 45.105$$

$$\hat{y}_{4|3} = 0.84 * 19.793 + 0.16 * 45.105 = 23.842$$

...

- The estimated value of α is high, meaning that recent observations have higher weight, and this can be seen in the fitted values as well, which follow quite well the historical data. It is not a very smooth line plot.

Holt's linear trend method I

- As its name suggests, this is a method which can be applied if we have trend in data.
- Remember, in SES, we can write the forecast and the level equations in the following way:

$$\text{Forecast equation: } \hat{y}_{t+h|t} = l_t$$

$$\text{Level equation: } l_t = \alpha * y_t + (1 - \alpha) * l_{t-1}$$

- For data with trend, we add another equation, called the *trend equation* and consider it as part of the forecast equation:

$$\text{Forecast equation: } \hat{y}_{t+h|t} = l_t + h * b_t$$

$$\text{Level equation: } l_t = \alpha * y_t + (1 - \alpha) * (l_{t-1} + b_{t-1})$$

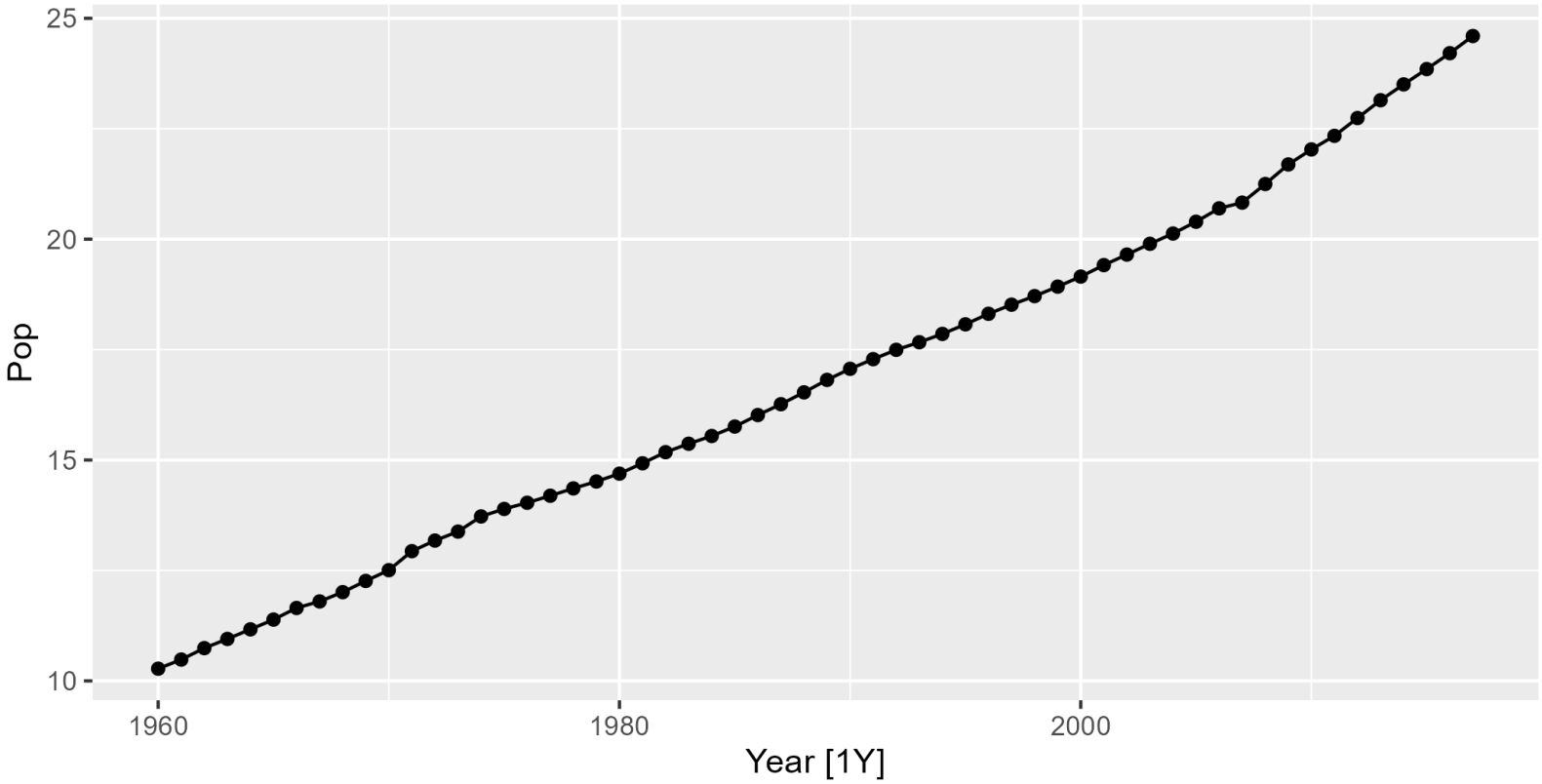
$$\text{Trend equation: } b_t = \beta^*(l_t - l_{t-1}) + (1 - \beta^*)b_{t-1}$$

Holt's linear trend method II

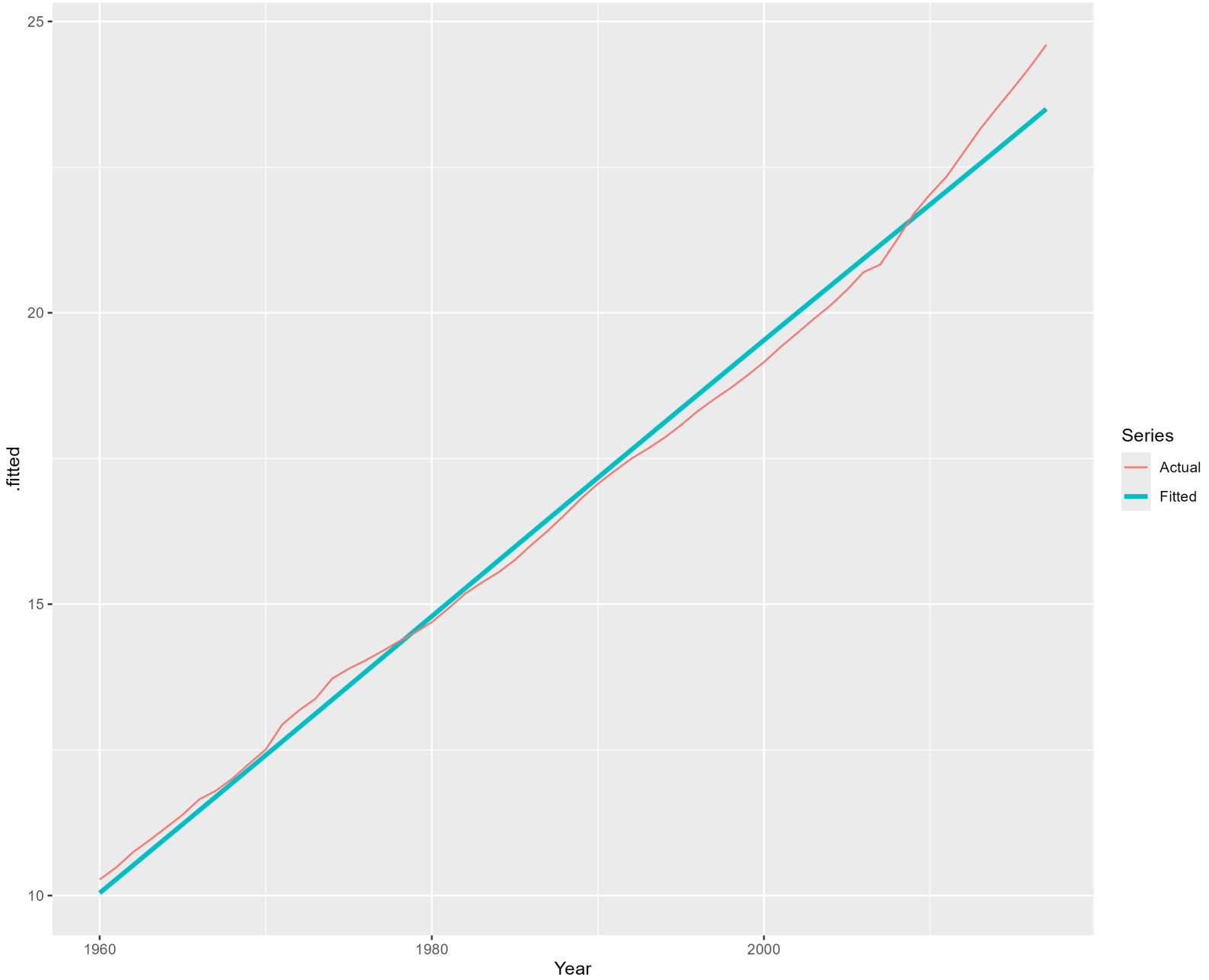
- where b_t denotes the estimate of the trend (slope) of the time series at moment t and β^* is the smoothing parameter for the trend (the other parameters have the same meaning as before).
- Similar to the simple exponential smoothing, the level equation shows that the level l_t is still the weighted average of the actual observation y_t and the one-step ahead (h $= 1$) training forecast for time t.
- The trend equation shows that b_t is the weighted average of the estimated trend at time t and the previous estimate of the trend.
- The forecast function is not longer flat, it will forecast values with a trend.
- The smoothing parameters α, β^* together with the initial level l_0 and initial trend b_0 will be estimated from the training data, using SSE.

- Let us see how the value of β^* influences the results. We will work with Australia's Population and an ETS model with $\alpha = 0.001$ and different values for β^*.

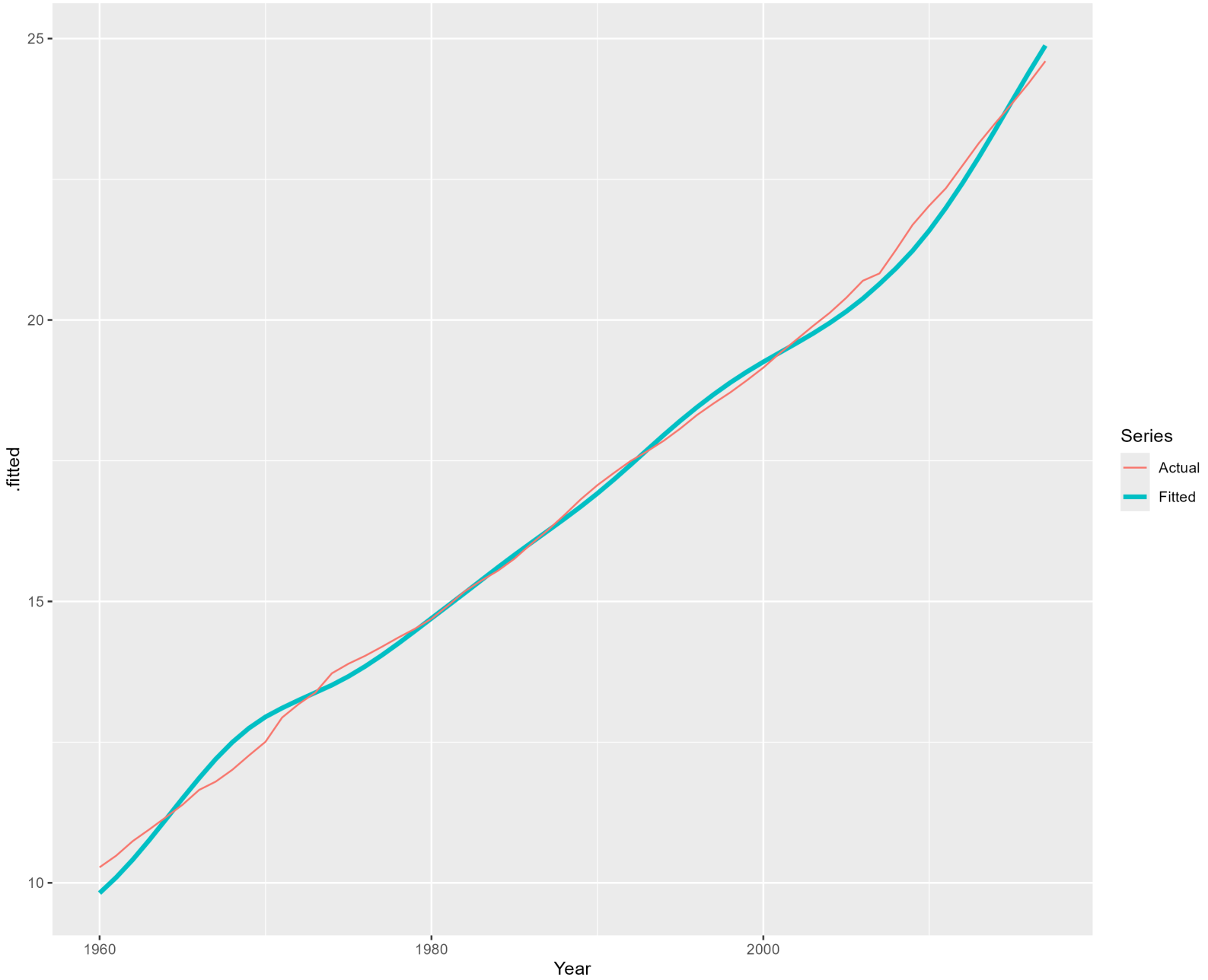
```
aus_economy <- global_economy |>
  filter(Code == "AUS") |>
  mutate(Pop = Population / 1e6)

aus_economy |> autoplot(Pop) +
    geom_point()
```


25
20
15
10
Pop
1960
1980
2000
Year [1Y]

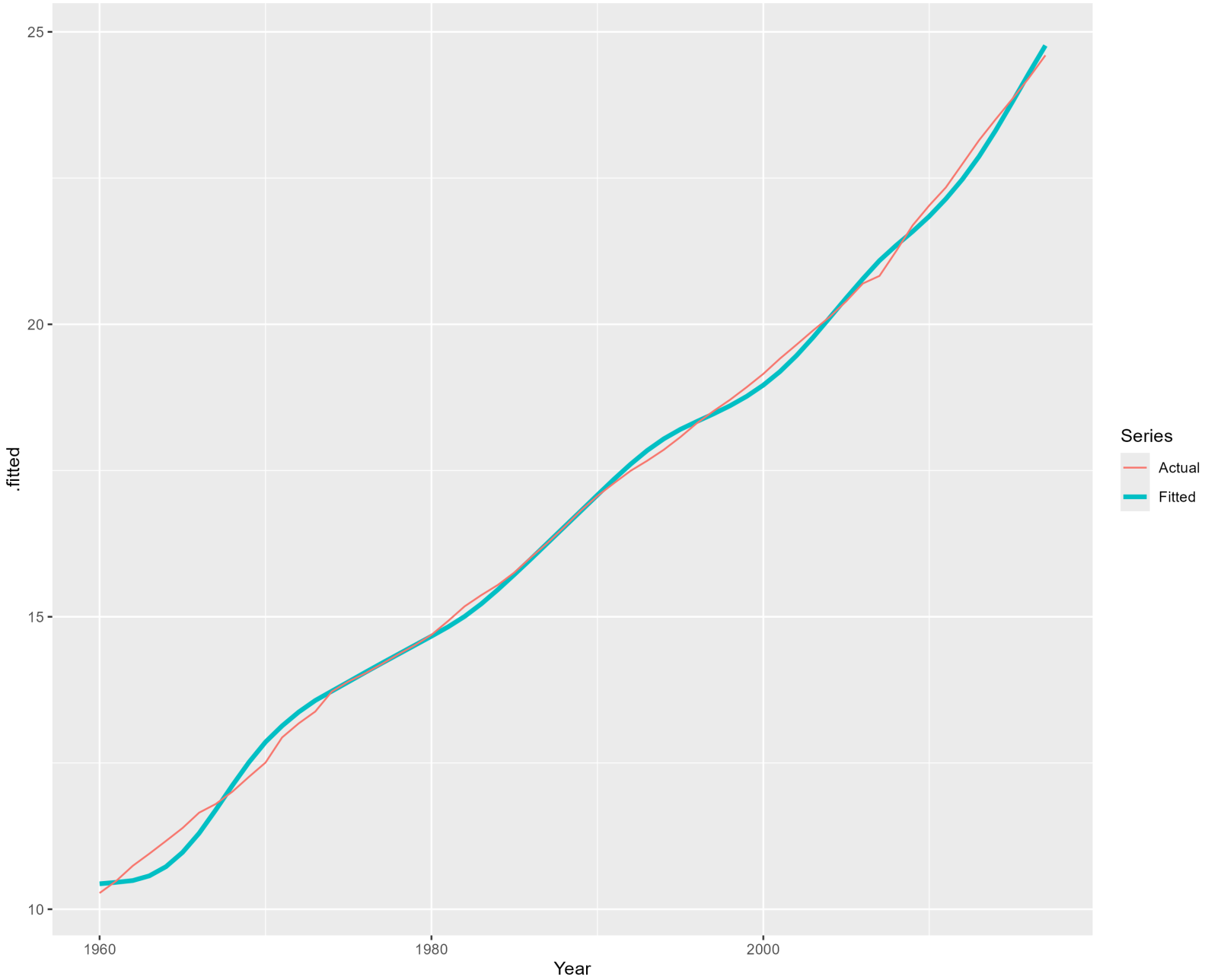
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.001$



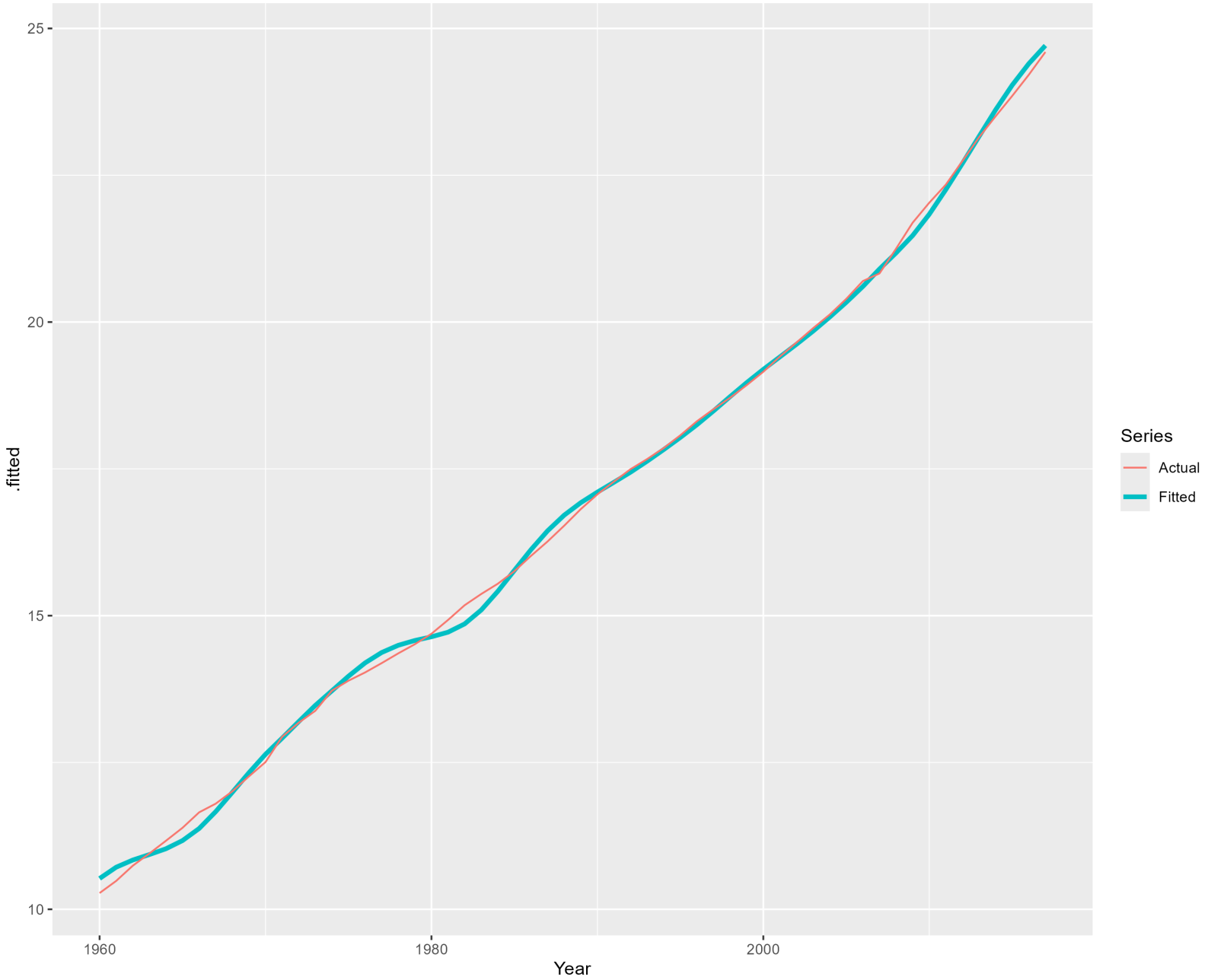
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.1$



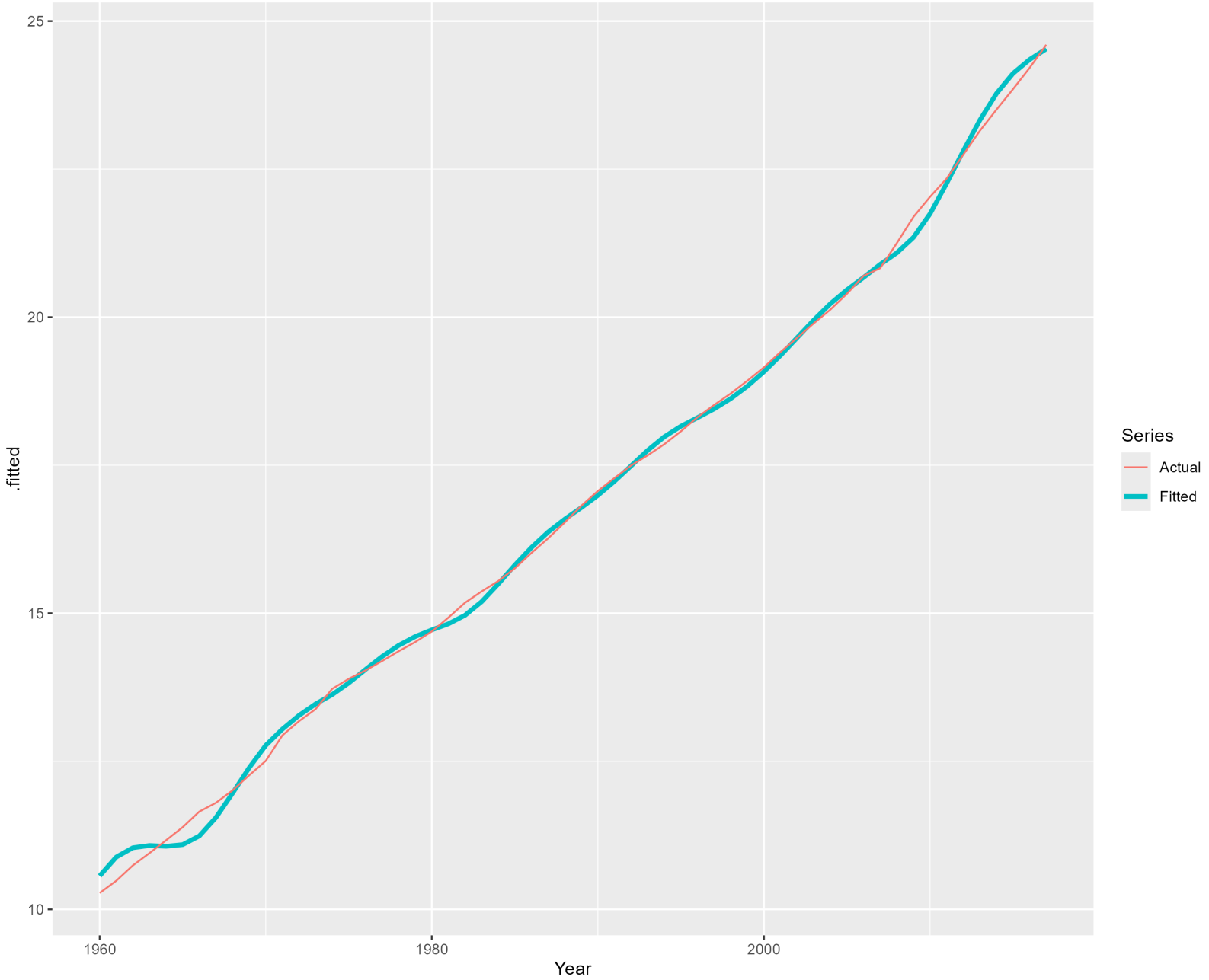
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.2$



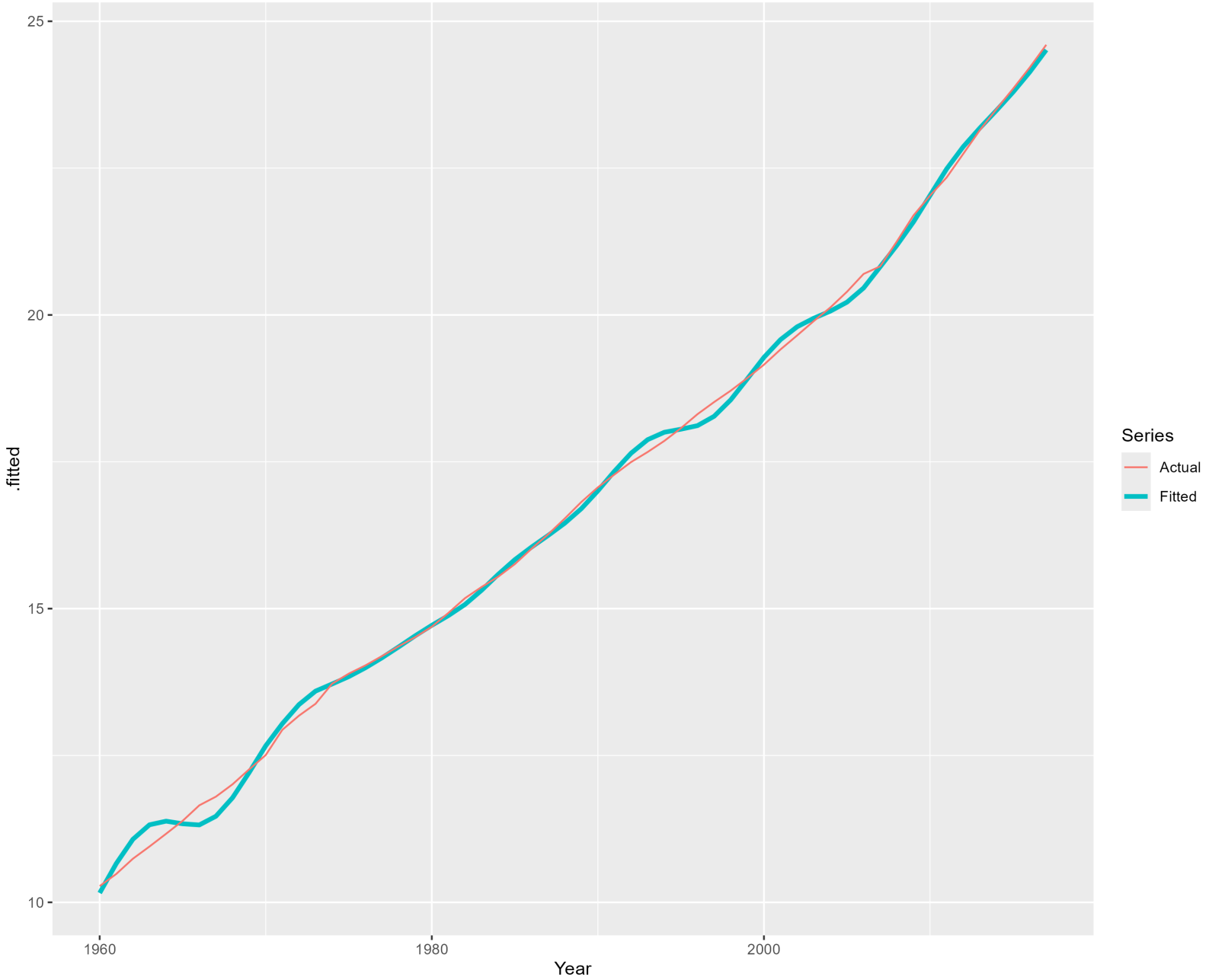
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.3$



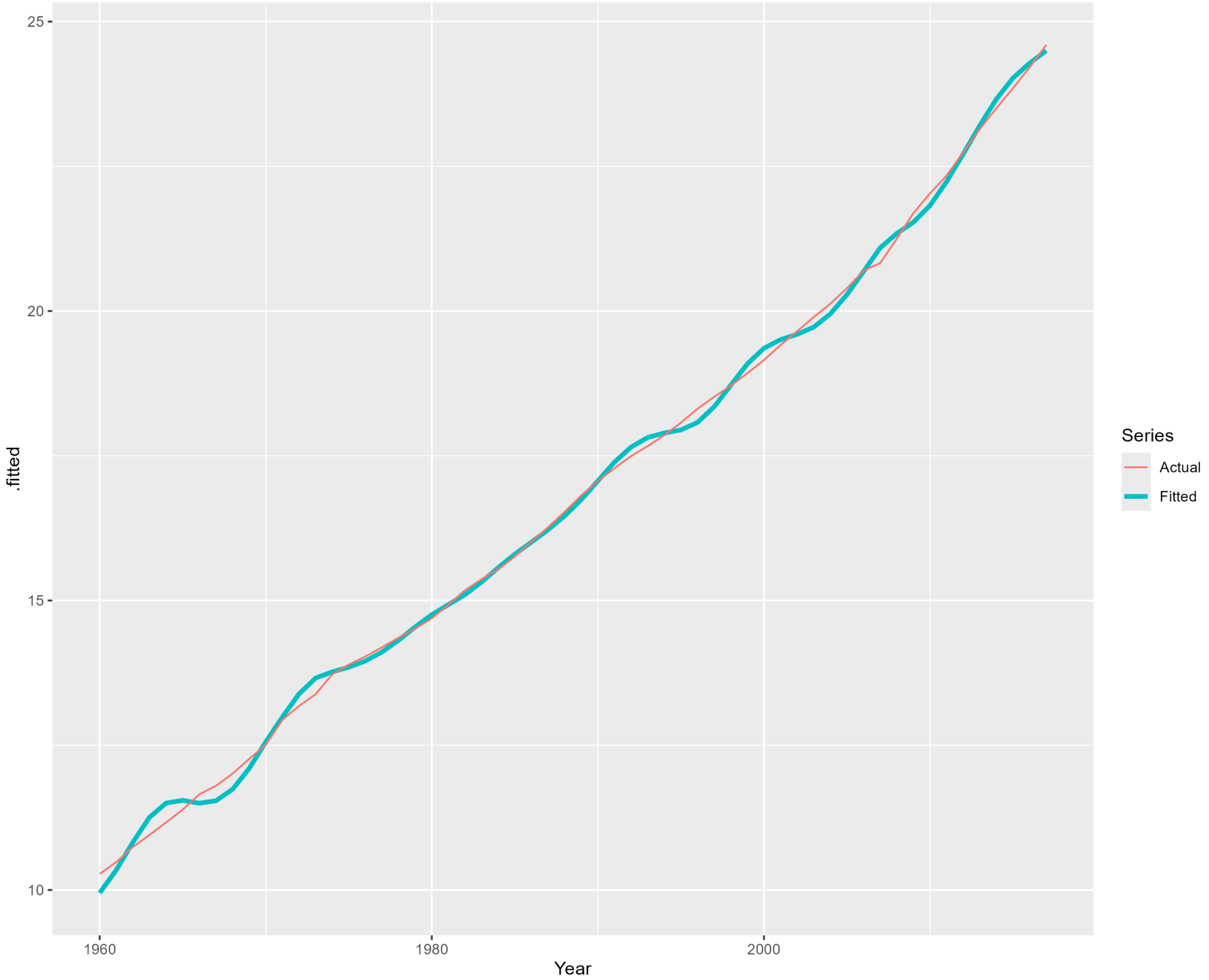
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.4$



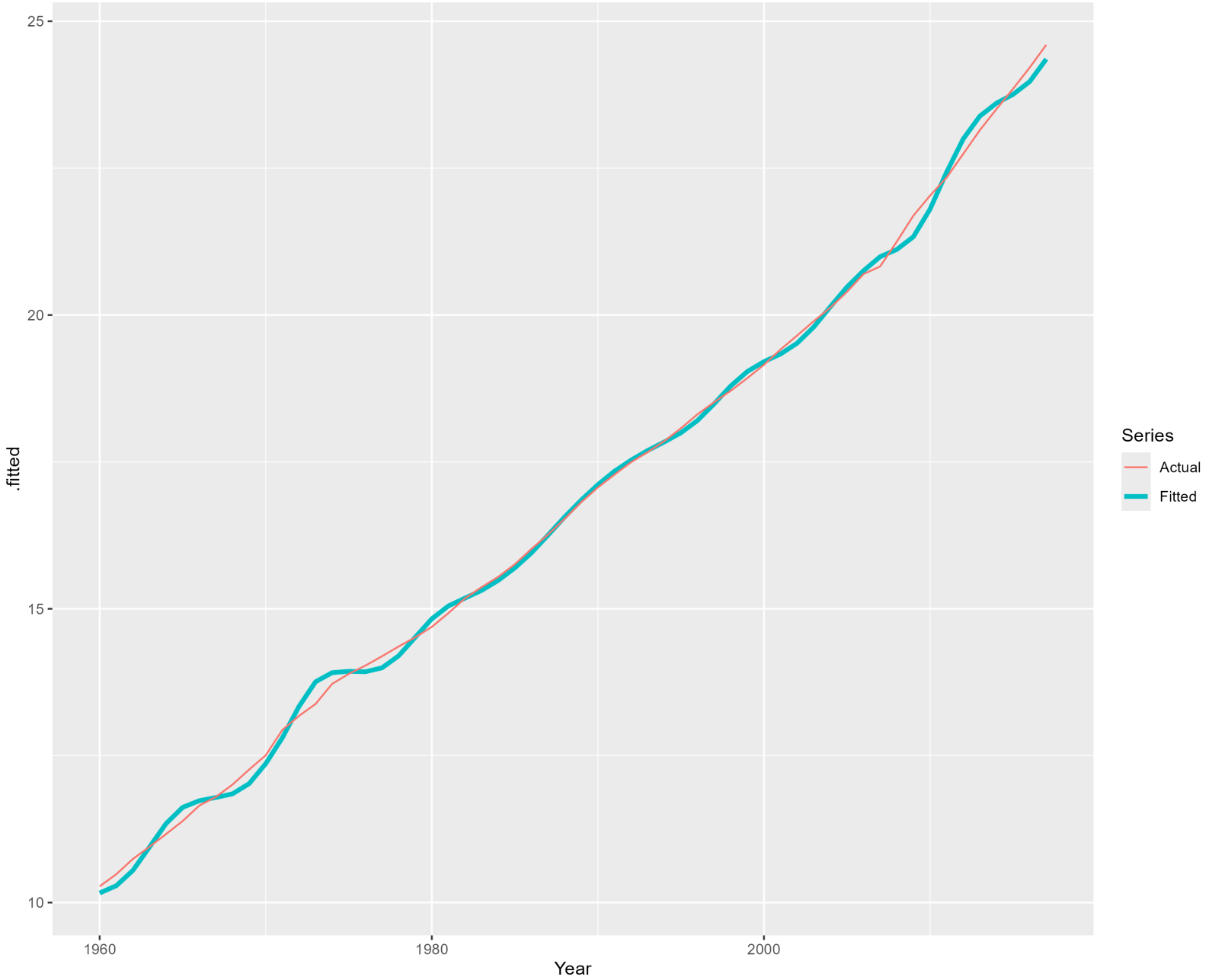
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.5$



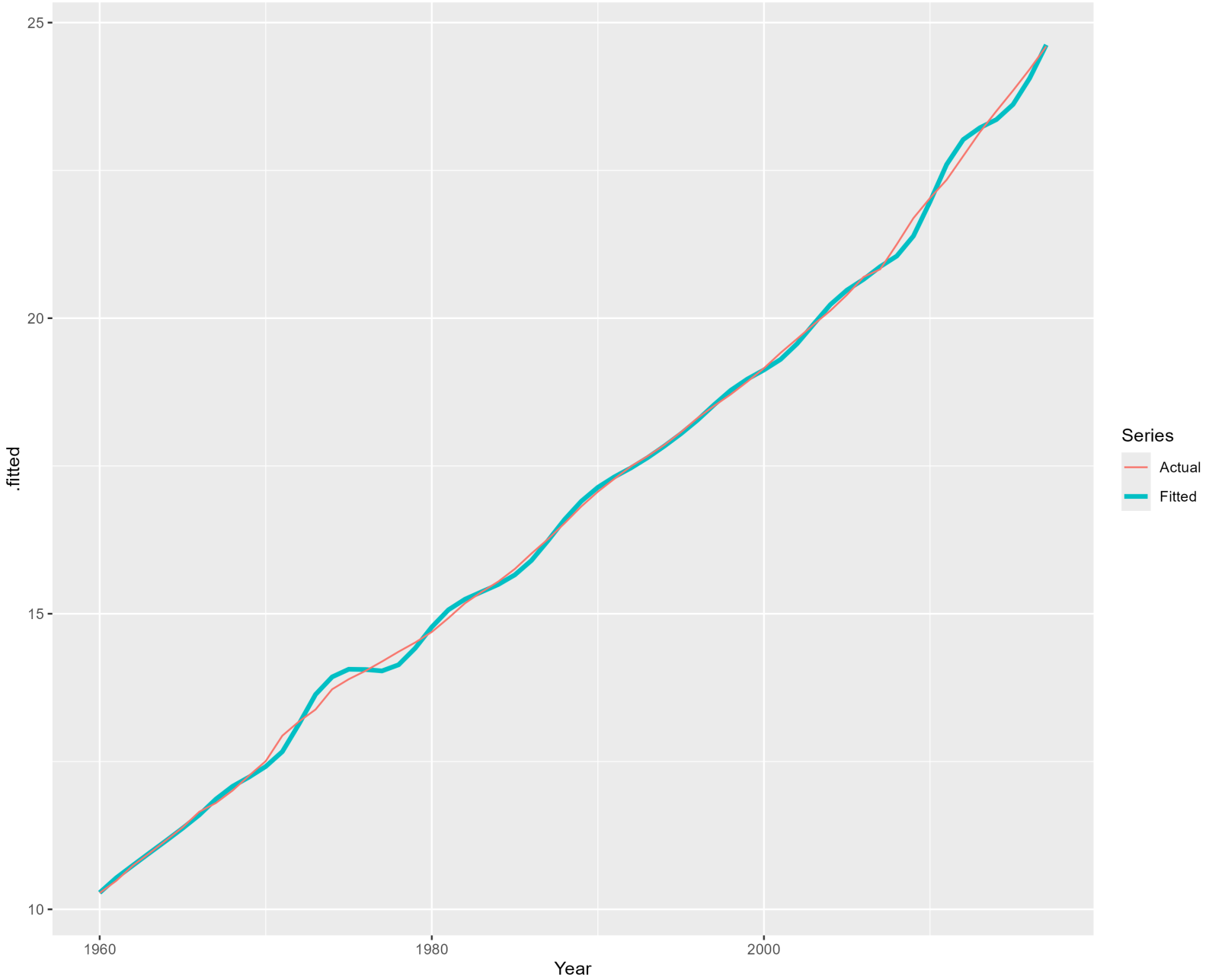
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.6$



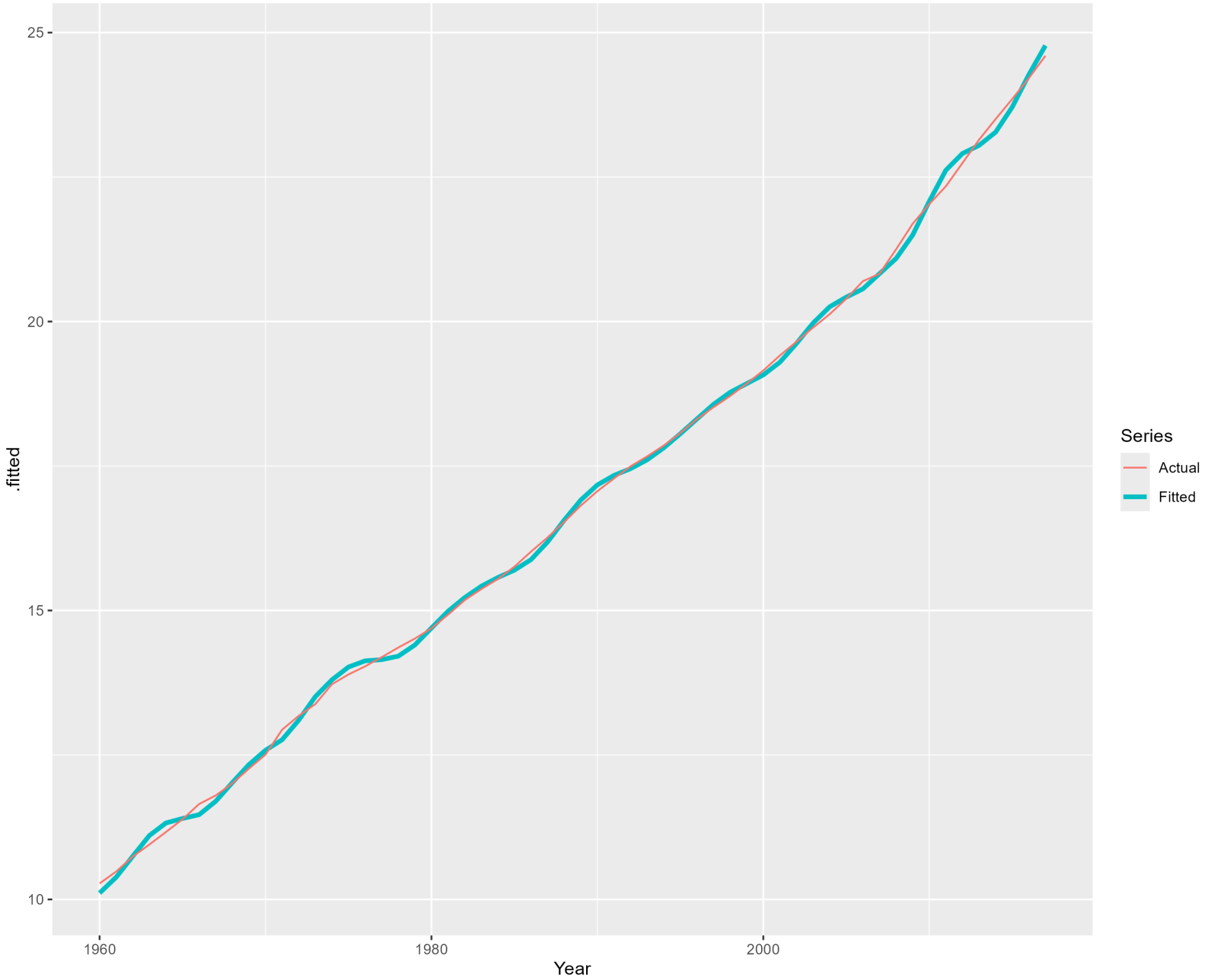
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.7$



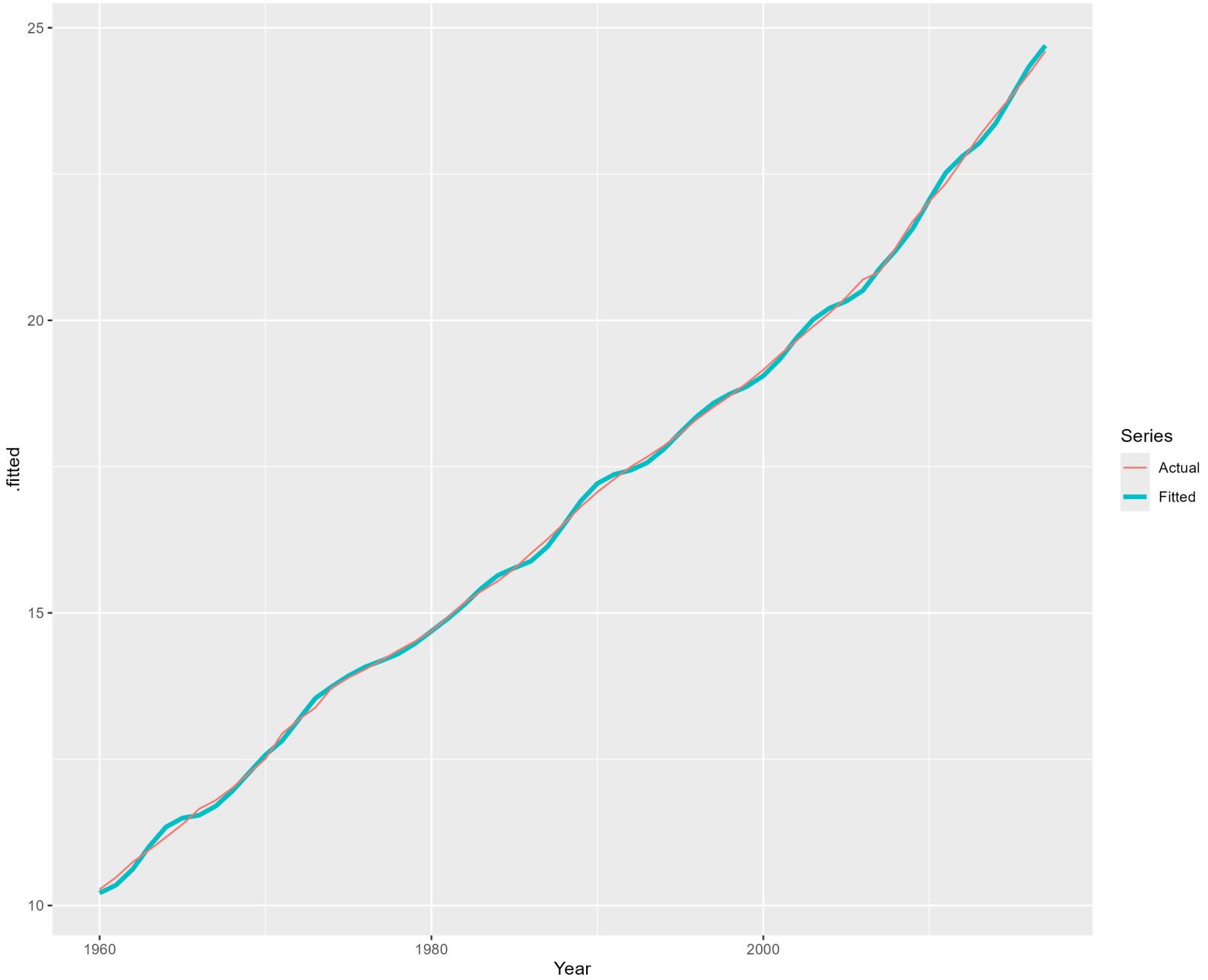
- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.8$



- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 0.9$



- Actual data and fitted value for $\alpha = 0.001$ and $\beta^* = 1$

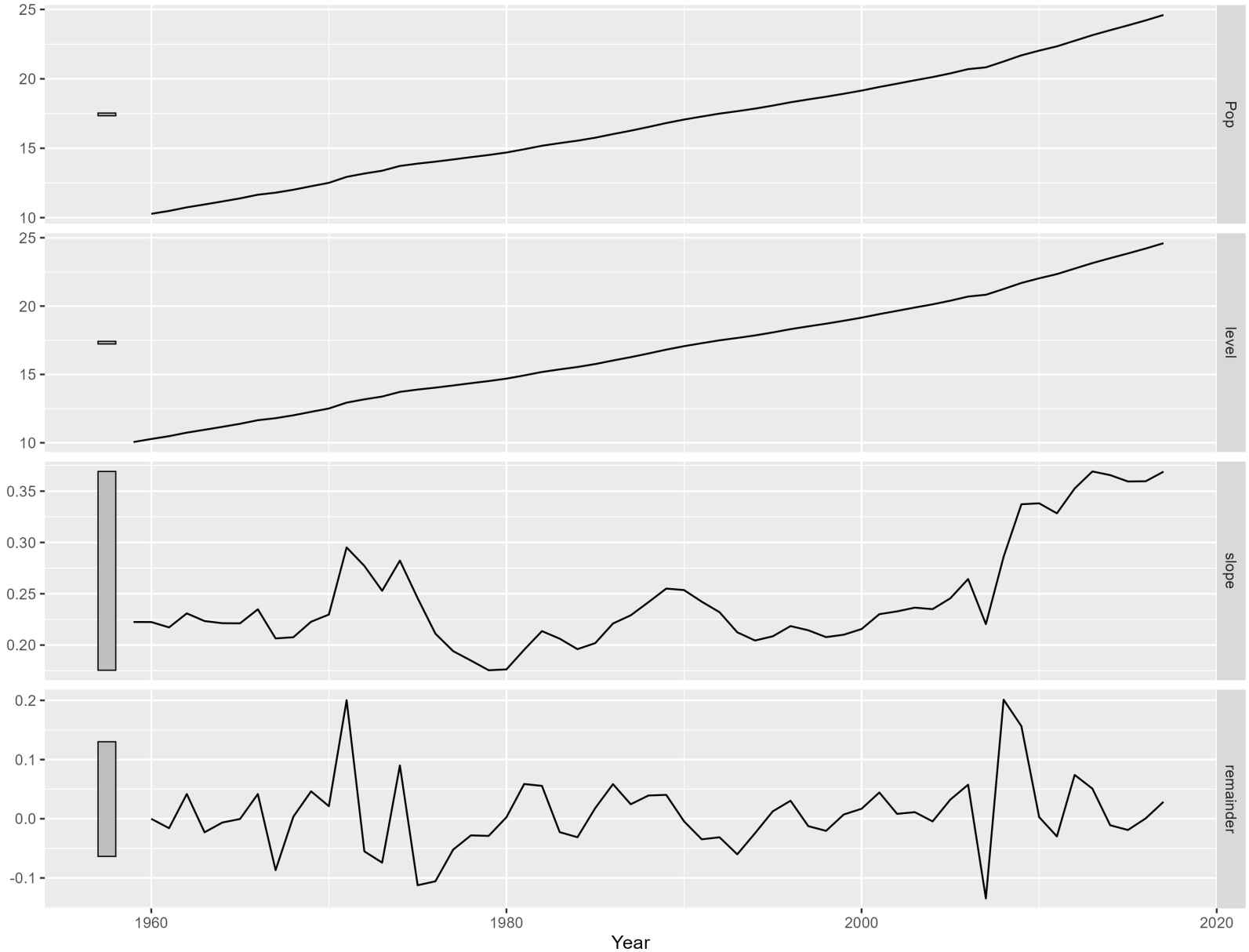


Example

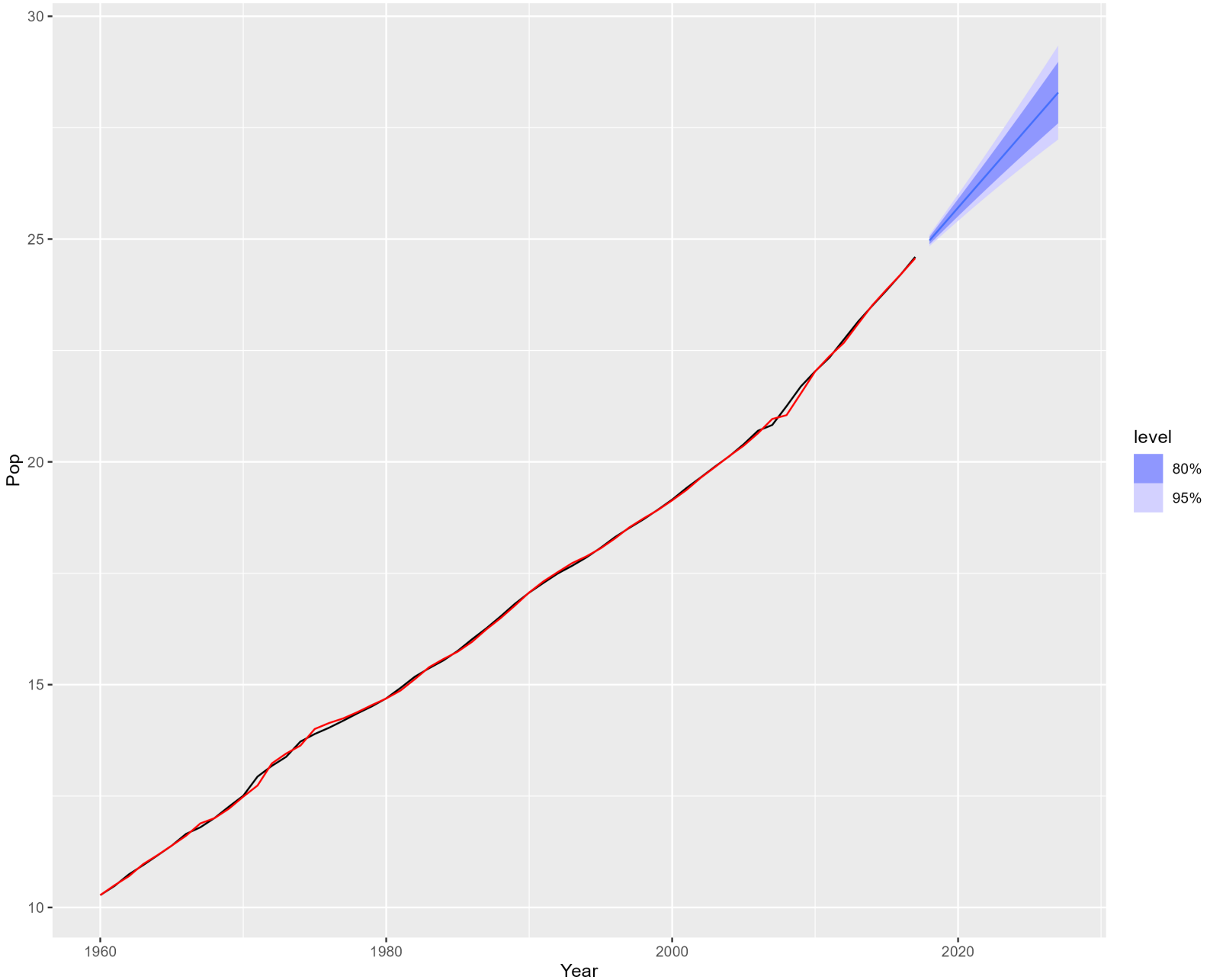
- Let us now see what values the ETS model will estimate for the parameters.

```
fit <- aus_economy |>
  model(AAN = ETS(Pop ~ error("A") + trend("A") + season("N")))
fit |> report()

fit |> components()  |>
  autoplot()
```

```
Series: Pop
Model: ETS(A,A,N)
  Smoothing parameters:
    alpha = 0.9999
    beta  = 0.3266366

  Initial states:
     l[0]      b[0]
 10.05414 0.2224818

  sigma^2:  0.0041

      AIC      AICc       BIC
-76.98569 -75.83184 -66.68347
```


ETS(A,A,N) decomposition
Pop = lag(level, 1) + lag(slope, 1) + remainder
Pop
level
slope
remainder
Year
1960
1980
2000
2020

```
fit |> tidy()

fit_forecast <- fit |>
  forecast(h = 10)

fit_augmented <- fit |> augment()

fit_forecast |> autoplot() +
  autolayer(aus_economy, Pop) +
  autolayer(fit_augmented, .fitted, color = "red")
```


30
25
20
15
10
Pop
1960
1980
2000
2020
Year
level
80%
95%

- We can see that the following parameter values were estimated for the linear trend model:
  - $\alpha = 0.999$
  - $\beta = 0.327$
  - $l_0 = 10.05$
  - $b_0 = 0.222$
- If we join the fitted values with the components, we can follow how the equations work.

```
fit |> components() |>
 left_join(fitted(fit), by= c("Country", ".model", "Year"))
```

```
 A dable: 59 x 8 [1Y]
# Key:     Country, .model [1]
# :        Pop = lag(level, 1) + lag(slope, 1) + remainder
   Country   .model  Year   Pop level slope remainder .fitted
   <fct>     <chr>  <dbl> <dbl> <dbl> <dbl>     <dbl>   <dbl>
 1 Australia AAN     1959  NA    10.1 0.222 NA           NA
 2 Australia AAN     1960  10.3  10.3 0.222 -0.000145    10.3
 3 Australia AAN     1961  10.5  10.5 0.217 -0.0159      10.5
 4 Australia AAN     1962  10.7  10.7 0.231  0.0418      10.7
 5 Australia AAN     1963  11.0  11.0 0.223 -0.0229      11.0
 6 Australia AAN     1964  11.2  11.2 0.221 -0.00641     11.2
 7 Australia AAN     1965  11.4  11.4 0.221 -0.000314    11.4
 8 Australia AAN     1966  11.7  11.7 0.235  0.0418      11.6
 9 Australia AAN     1967  11.8  11.8 0.206 -0.0869      11.9
```

Damped trend models

- The forecasts of Holt's methods contain a constant trend (b_t) which is multiplied by the forecast horizon (h). This means that all future forecasts have the same trend. While this is accurate on short term forecasts, on the long term it tends to over-estimate the actual value.
- In order to avoid this over-estimation, *damped* trends models were introduced, where, with the use of an extra parameter ϕ, the trend line is damped to a flat line in the future.
- The value of ϕ determines how fast the trend becomes linear (smaller value of ϕ means the line will be linear sooner). In general it is suggested to use values between 0.8 and 0.98.

- Using the damped model, the previous equations are slightly changed:

$$\hat{y}_{t+h|t} = l_t + (\phi + \phi^2 + ... + \phi^h) * b_t$$

$$l_t = \alpha * y_t + (1 - \alpha) * (l_{t-1} + \phi * b_{t-1})$$

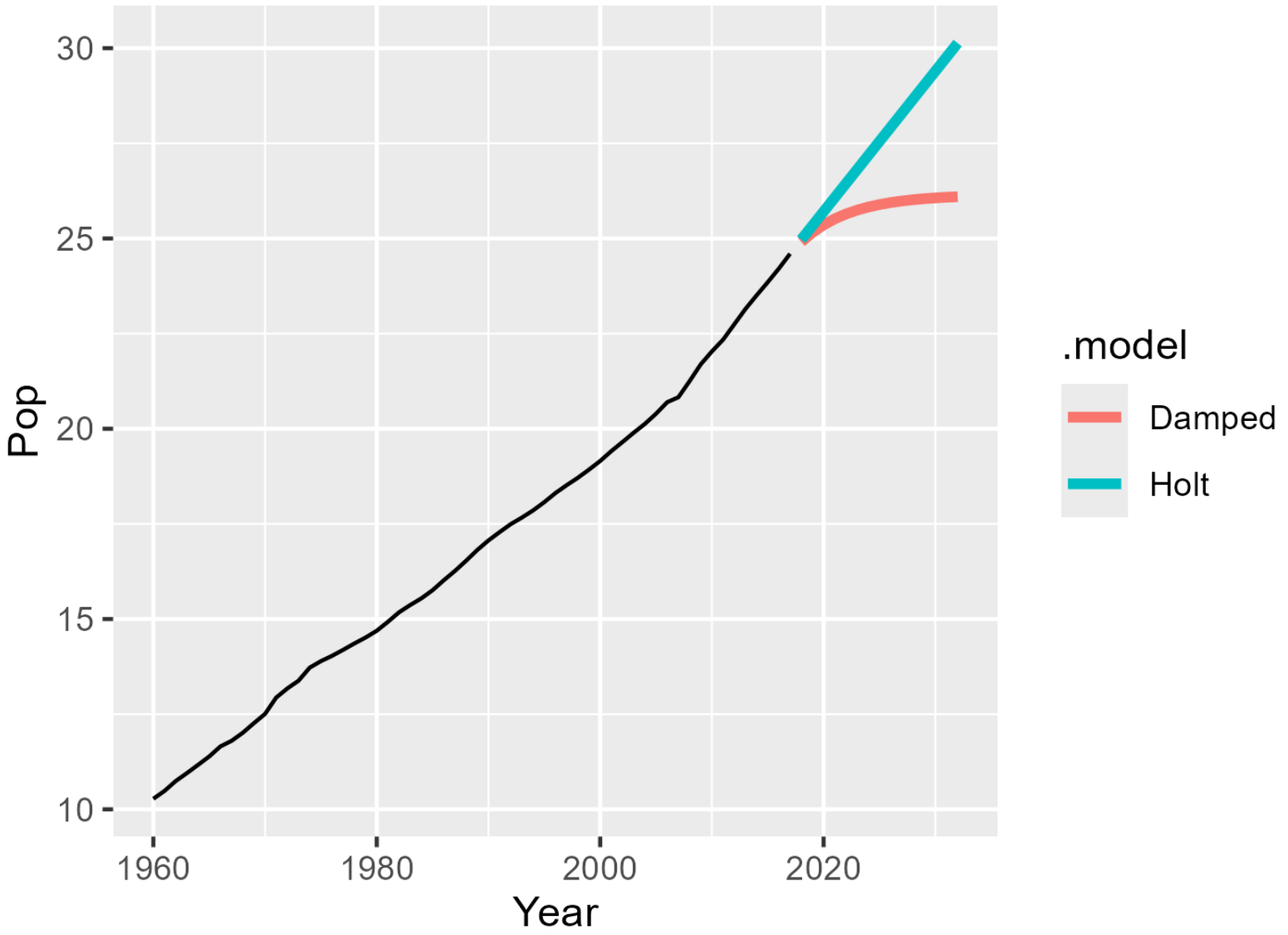
$$b_t = \beta^* * (l_t - l_{t-1}) + (1 - \beta^*) * \phi * b_{t-1}$$

```
aus_economy_damped <-aus_economy |>
  model(Holt = ETS(Pop ~ error("A") + trend("A") + season("N")),
    Damped = ETS(Pop ~ error("A") + trend("Ad", phi = 0.8) + season("N"))) #AD -
      additive damped
aus_economy_damped |> tidy() #function report does not work for multiple models
```

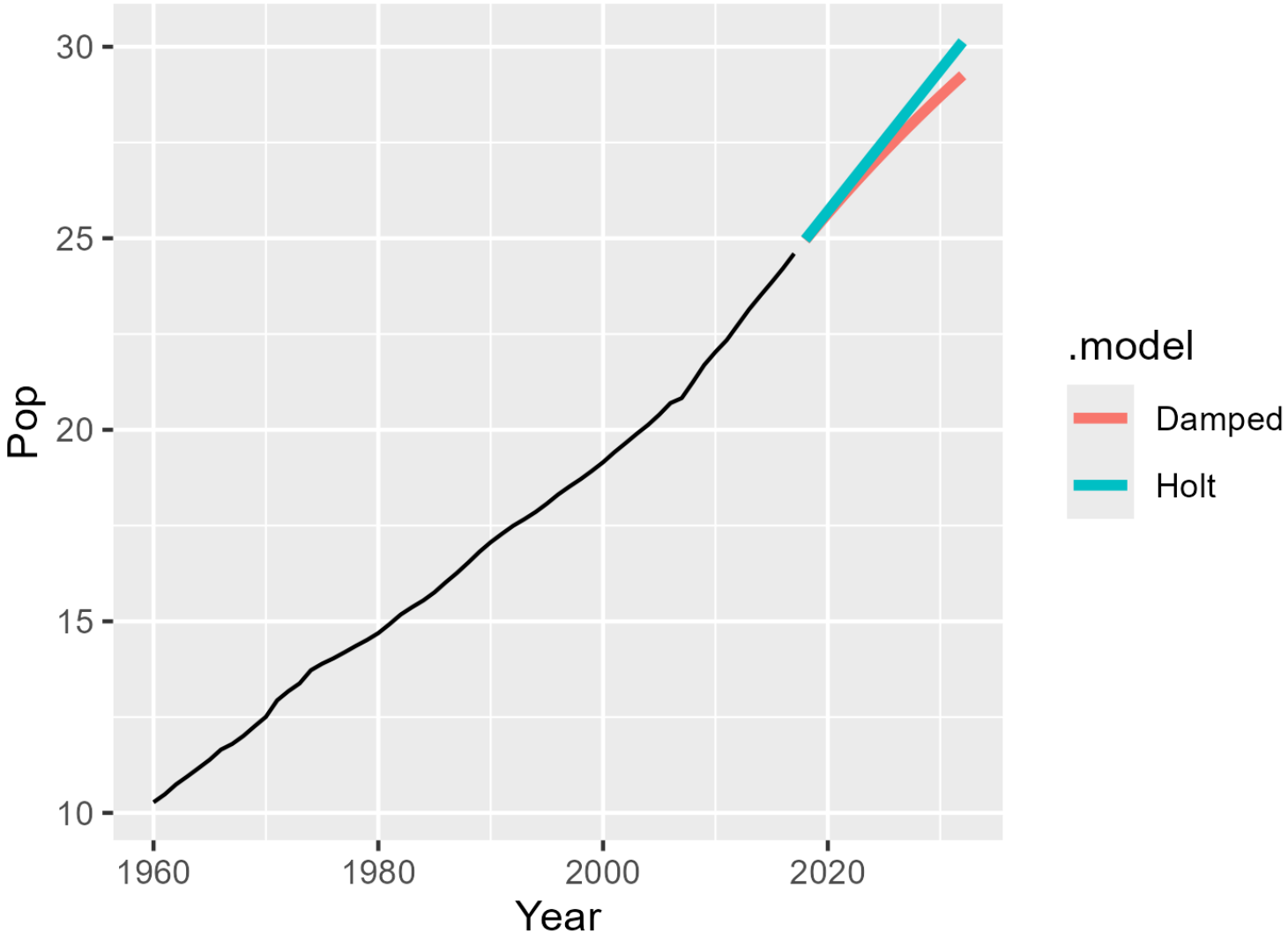
```
   .model   term  estimate
   <chr>    <chr>    <dbl>
 1 Holt     alpha    1.00
 2 Holt     beta     0.327
 3 Holt     l[0]    10.1
 4 Holt     b[0]     0.222
 5 Dampened alpha    1.00
 6 Dampened beta     1.00
 7 Dampened phi      0.8
 8 Dampened l[0]    10.0
 9 Dampened b[0]     0.315
```

```
aus_economy_damped |> augment() |> view()

aus_economy_damped_forecast <- aus_economy_damped |> forecast(h = 15)

aus_economy_damped_forecast |> autoplot(aus_economy, level = NULL, linewidth
    =1.3)

%
```

- Regular model and damped model with $\phi = 0.8$



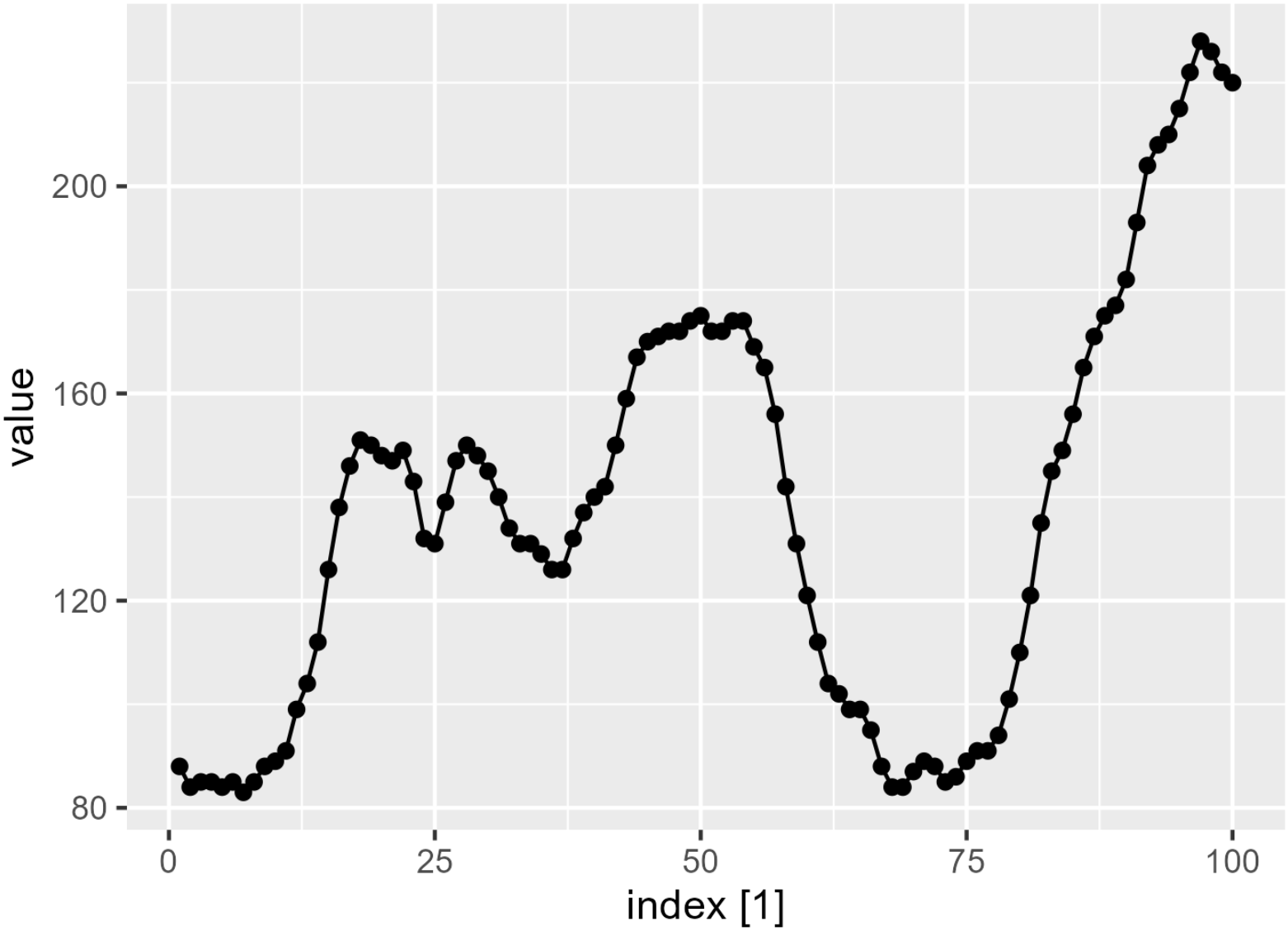
- Regular model and damped model with $\phi = 0.98$ (same code as above, but with ϕ set to 0.98.



Example

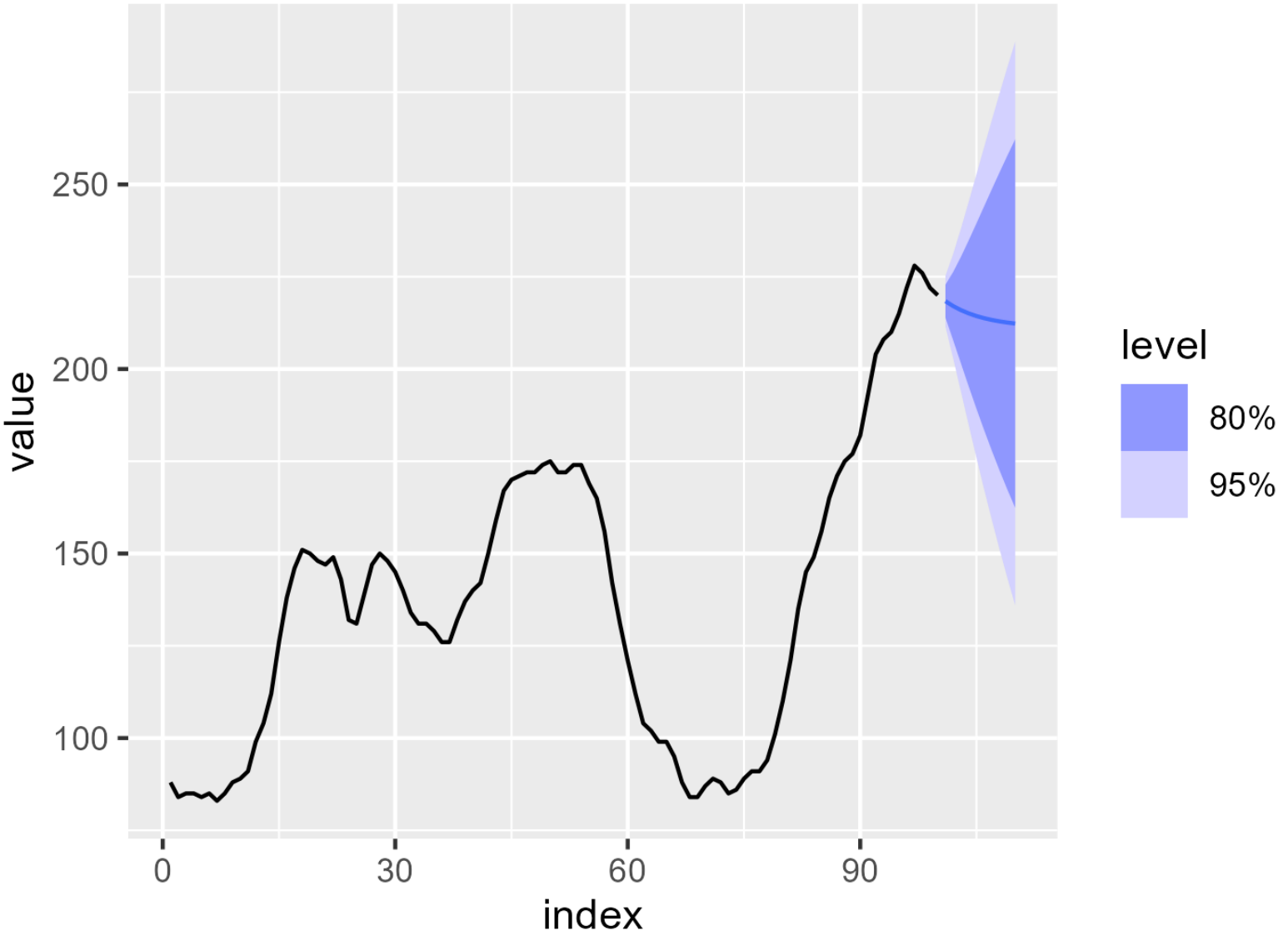
- Let's now compare the performance of the three methods discussed up until now: SES, Holt's method and the damped model.
- We will use a data set containing the number of users connected to the internet via a server. There is an observation in every minute and we have a total of 100 minutes.
- Also, in order to do a proper comparison, we will use time series cross validation for computing the evaluation measures.

```
www_usage <- WWWusage |> as_tsibble()
www_usage_cv <- www_usage |>
  stretch_tsibble(.init = 10)

www_usage |> autoplot() +
  geom_point()

www_usage_model <- www_usage_cv |>
  model(SES = ETS(value ~ error("A") + trend("N") + season("N")),
    Holt = ETS(value ~ error("A") + trend("A") + season("N")),
    Damped = ETS(value ~ error("A") + trend("Ad") + season("N")))

www_usage_model |>
  forecast(h = 1) |>
  accuracy(www_usage)
```


200
160
120
80
value
0
25
50
75
100
index [1]

```
.model .type     ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
 <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Damped Test  0.288   3.69  3.00 0.347  2.26 0.663 0.636 0.336
2 Holt   Test  0.0610  3.87  3.17 0.244  2.38 0.701 0.668 0.296
3 SES    Test  1.46    6.05  4.81 0.904  3.55 1.06  1.04  0.803
```

```r
#let's do real forecast for the winner
www_usage |>
  model(ETS(value ~ error("A") + trend("Ad") + season("N"))) |>
  forecast(h = 10) |>
  autoplot(www_usage)
```

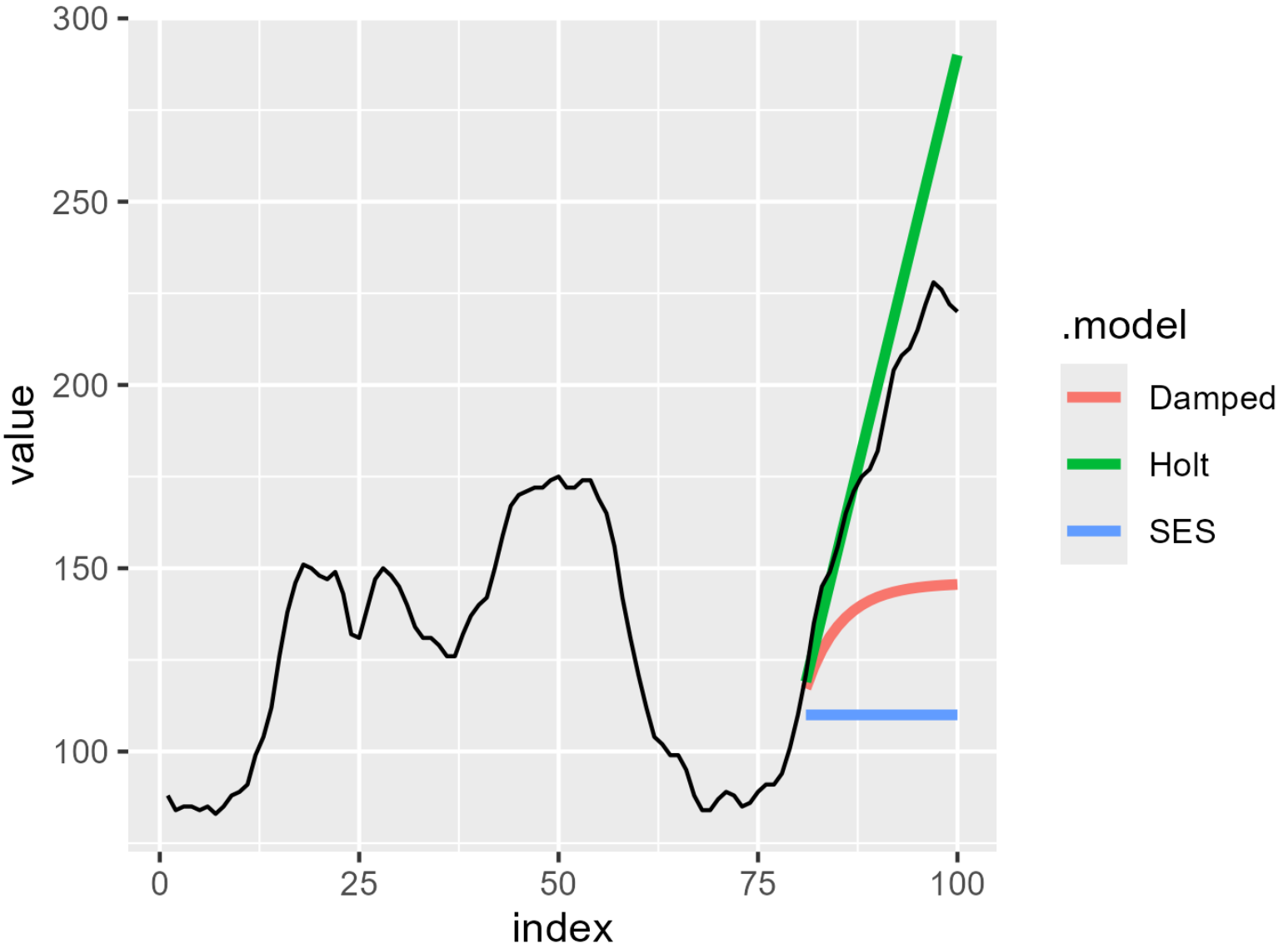

250
200
150
100
value
0
30
60
90
index
level
80%
95%

- Alternatively, we could try a train-test split as well for *proper* evaluation (proper in the sense that the testing and evaluation is done on data which was not used for training the model, like in case of fitted values and residuals).

```
www_usage_train <- www_usage |>
  head(n = 80)
www_usage_test <- www_usage |>
  tail(n= 20)

www_usage_model2 <- www_usage_train |>
  model(SES = ETS(value ~ error("A") + trend("N") + season("N")),
        Holt = ETS(value ~ error("A") + trend("A") + season("N")),
        Damped = ETS(value ~ error("A") + trend("Ad", phi = 0.8) + season("N")))
       #try phi = 0.8, 0.9. 0.98

www_usage_model2 |>
  forecast(h = 20) |>
  accuracy(www_usage_test)
```

.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1 Damped	Test	47.3	53.6	47.3	23.7	23.7	NaN	NaN	0.874
2 Holt	Test	-18.3	28.1	20.5	-8.33	9.89	NaN	NaN	0.794
3 SES	Test	76.2	82.8	76.2	38.9	38.9	NaN	NaN	0.850

```
www_usage_model2 |>
  forecast(h = 20) |>
  autoplot(www_usage, level = NULL)
```


300
250
200
150
100
value
0
25
50
75
100
index
.model
Damped
Holt
SES

Methods with seasonality

- It is also called *Holt-Winters'* method and it is made of a forecast equation and 3 smoothing equations: one for level l_t, one for trend b_t and one for season s_t each having its own smoothing parameter α, β^* and γ.
- We will denote the seasonal period by m (the number of observations in a season).
- There are two variations for this method:
  - *additive method*, which is suitable if the seasonal component has approximately constant seasonal variation. In this case the seasonal component is expressed in absolute terms and the sum of the components for a year will be approximately 0.
  - *multiplicative method*, which is suitable if the seasonal component changes proportionally to the level of the series. In this case the seasonal component is expressed as percentage and the sum for a year will be approximately m.

Holt-Winters' additive method

- The component form for this method is:

$$\text{Forecast equation: } \hat{y}_{t+h|t} = l_t + h * b_t + s_{t+h-m(k+1)}$$

$$\text{Level equation: } l_t = \alpha * (y_t - s_{t-m}) + (1 - \alpha) * (l_{t-1} + b_{t-1})$$

$$\text{Trend equation: } b_t = \beta^* * (l_t - l_{t-1}) + (1 - \beta^*) * b_{t-1}$$

$$\text{Season equation: } s_t = \gamma * (y_t - l_{t-1} - b_{t-1}) + (1 - \gamma) * s_{t-m}$$

- where k is the integer part of $\frac{h-1}{m}$ (we use it to make sure that we consider values for the last season)

Holt-Winters' multiplicative method

- The component form for the multiplicative method is:

$$\text{Forecast equation: } \hat{y}_{t+h|t} = (l_t + h * b_t) * s_{t+h-m(k+1)}$$

$$\text{Level equation: } l_t = \alpha * \frac{y_t}{s_{t-m}} + (1-\alpha) * (l_{t-1} + b_{t-1})$$

$$\text{Trend equation: } b_t = \beta^* * (l_t - l_t - 1) + (1-\beta^*) * b_{t-1}$$

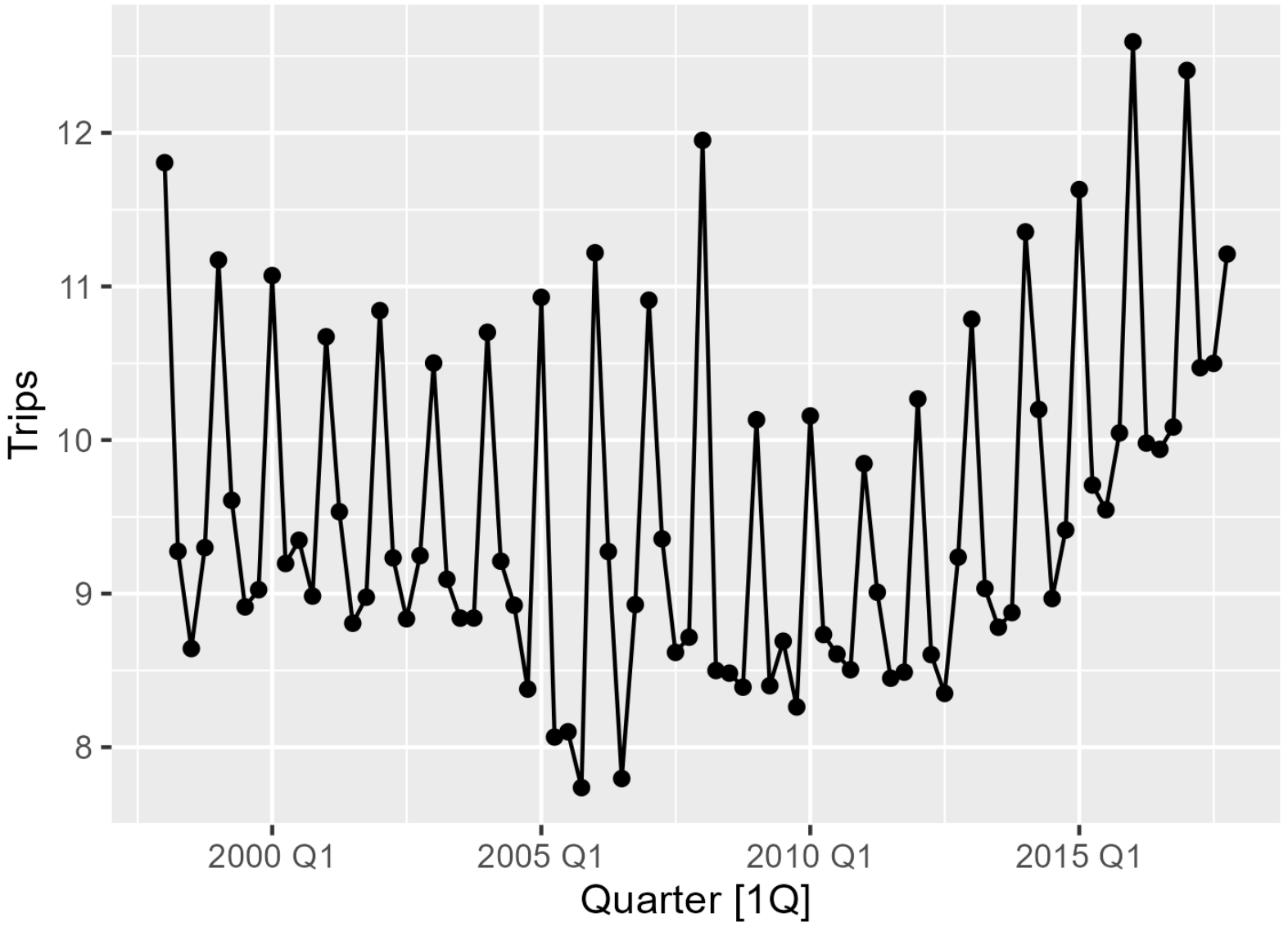
$$\text{Season equation: } s_t = \gamma * \frac{y_t}{(l_{t-1} + b_{t-1})} + (1-\gamma) * s_{t-m}$$

Example I

- Let's see how can the two models forecast quarterly visitor nights in Australia. We will use the *tourism* data set that we have used previously and consider only holiday type trips.

```
aus_holidays <- tourism |>
  filter(Purpose == "Holiday") |>
  summarise(Trips = sum(Trips) / 1e3)

aus_holidays |> autoplot() +
  geom_point()

aus_holidays_model <- aus_holidays |>
  model(additive = ETS(Trips ~ error("A") + trend("A") + season("A")),
        multiplicative = ETS(Trips ~ error("A") + trend("A") + season("M")))

aus_holidays_model |> tidy()
```

Example II



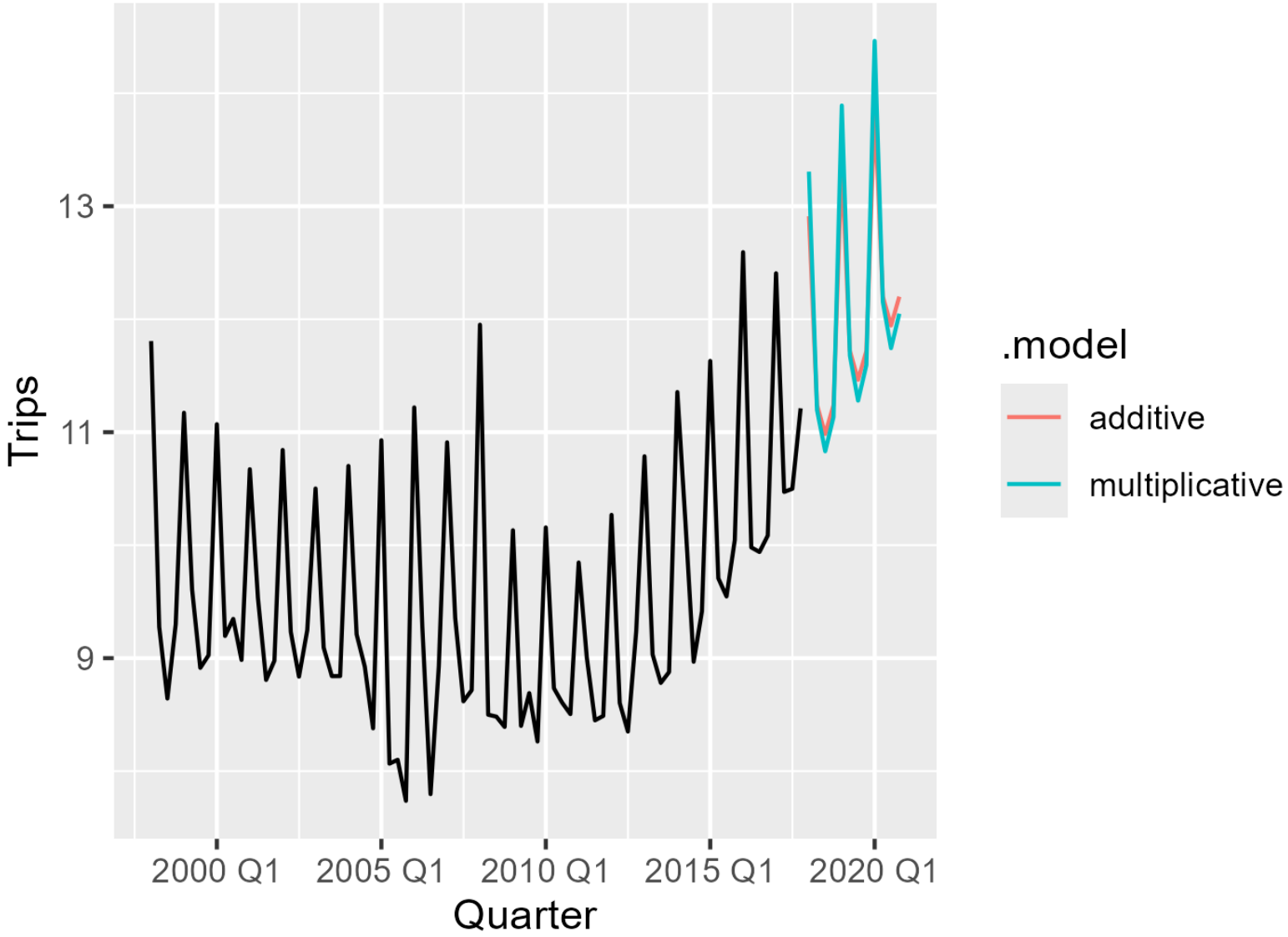
Example III

	.model	term	estimate
	<chr>	<chr>	<dbl>
1	additive	alpha	0.262
2	additive	beta	0.0431
3	additive	gamma	0.000100
4	additive	l[0]	9.79
5	additive	b[0]	0.0211
6	additive	s[0]	-0.534
7	additive	s[-1]	-0.670
8	additive	s[-2]	-0.294
9	additive	s[-3]	1.50
10	multiplicative	alpha	0.371
11	multiplicative	gamma	0.000100
12	multiplicative	l[0]	9.85
13	multiplicative	s[0]	0.939
14	multiplicative	s[-1]	0.931
15	multiplicative	s[-2]	0.967
16	multiplicative	s[-3]	1.16

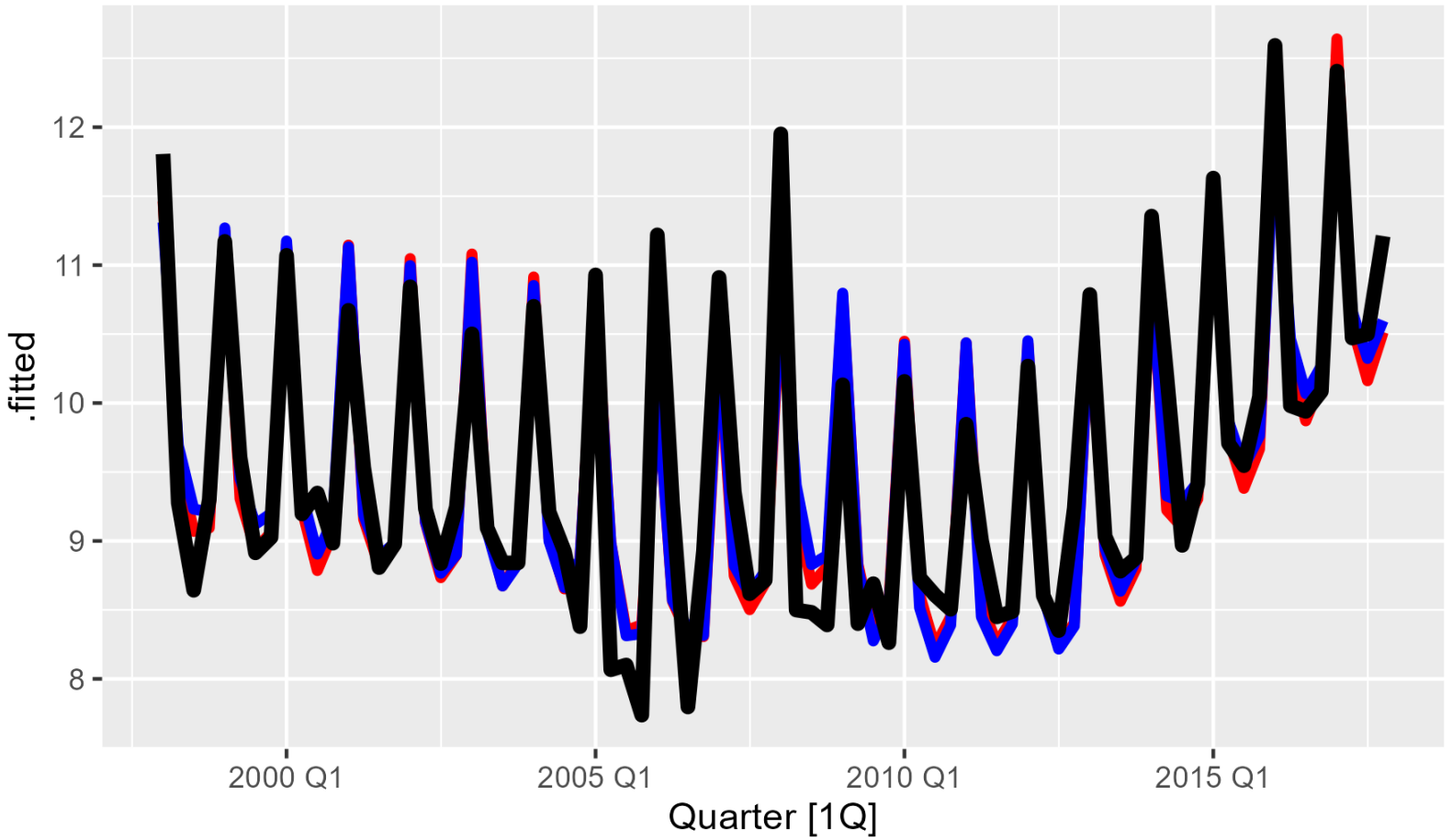
Example IV

```
aus_holidays_forecast <- aus_holidays_model |>
  forecast(h = "3 years")

aus_holidays_forecast |>
  autoplot(aus_holidays, level = NULL)

  aus_holidays_model[1] |> augment() |> autoplot(.fitted, color = "red",
      linewidth = 1.5) +
  autolayer(aus_holidays, color = "black", linewidth =2) +
  autolayer(aus_holidays_model[2] |> augment(), .fitted, color = "blue",
      linewidth = 1.5)
```

Example V


13
11
9
Trips
2000 Q1
2005 Q1
2010 Q1
2015 Q1
2020 Q1
Quarter
.model
additive
multiplicative

Example VI



- The *tidy* function gives us the parameters that were estimated for the model. For the additive model we have:
  - $\alpha = 0.262$
  - $\beta^* = 0.0431$
  - $\gamma = 0.0001$
  - $l_0 = 9.79$
  - $b_0 = 0.0211$
  - $s_0 =$ -0.534
  - $s_{-1} =$ -0.670
  - $s_{-2} =$ -0.294
  - $s_{-3} = 1.5$
- For the seasonal part, we need estimates for an entire season (this is why we have values up until s_{-3}.
- We can see how the sum of the initial values is approximately 0.

- Let's see how the computation for one-step forecasts of the additive model (h = 1) happens.

$$\text{Forecast equation: } \hat{y}_{t+1|t} = l_t + b_t + s_{t+1-m}$$

$$\text{Level equation: } l_t = \alpha * (y_t - s_{t-m}) + (1 - \alpha) * (l_{t-1} + b_{t-1})$$

$$\text{Trend equation: } b_t = \beta^* * (l_t - l_{t-1}) + (1 - \beta^*) b_{t-1}$$

$$\text{Season equation: } s_t = \gamma * (y_t - l_{t-1} - b_{t-1}) + (1 - \gamma) * s_{t-m}$$

time t	y_t	l_t	b_t	s_t	$\hat{y}_{t+1\|t}$
0	-	9.79	0.021	1.5, -0.294, -0.670, -0.534	-
1	11.8	9.939	0.027	1.5	9.672
2	9.28	9.863	0.022	-0.294	9.215
3	8.64	9.735	0.016	-0.670	9.216
4	9.3	9.772	0.017	-0.534	11.289

- The *tidy* function gives us the parameters that were estimated for the model. For the multiplicative model we have:
  - $\alpha = 0.223$
  - $\beta^* = 0.0311$
  - $\gamma = 0.0001$
  - $l_0 = 9.88$
  - $b_0 = -0.0279$
  - $s_0 = 0.942$
  - $s_{-1} = 0.926$
  - $s_{-2} = 0.968$
  - $s_{-3} = 1.16$
- For the seasonal part, we need estimates for an entire season (this is why we have values up until s_{-3}.
- We can see how the sum of the initial values is approximately m.

- Let's see how the computation for one-step forecasts of the multiplicative model (h = 1) happens.

$$\text{Forecast equation: } \hat{y}_{t+1|t} = (l_t + b_t) * s_{t+1-m}$$

$$\text{Level equation: } l_t = \alpha * \frac{y_t}{s_{t-m}} + (1-\alpha) * (l_{t-1} + b_{t-1})$$

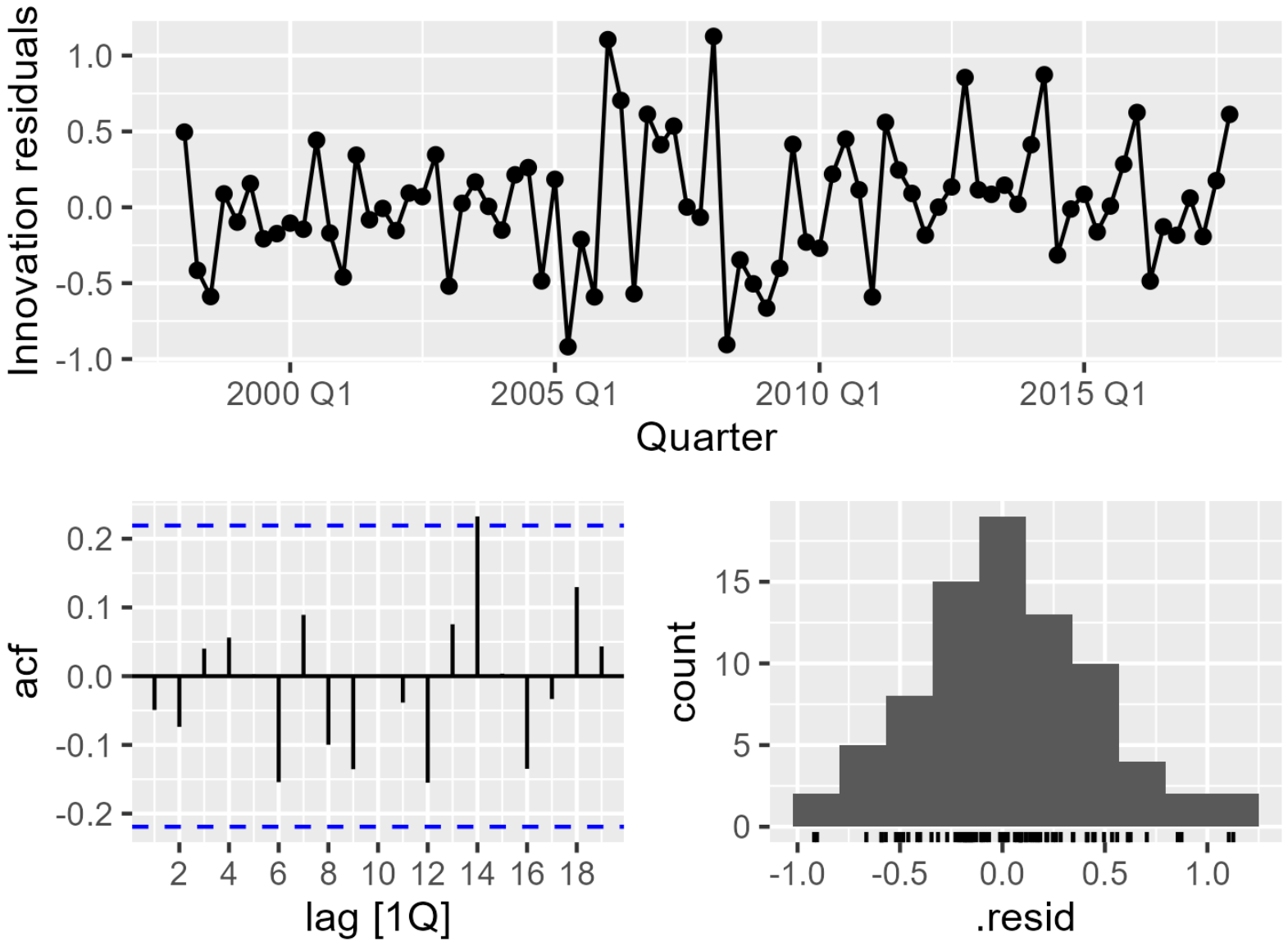
$$\text{Trend equation: } b_t = \beta^* * (l_t - l_{t-1}) + (1-\beta^*) b_{t-1}$$

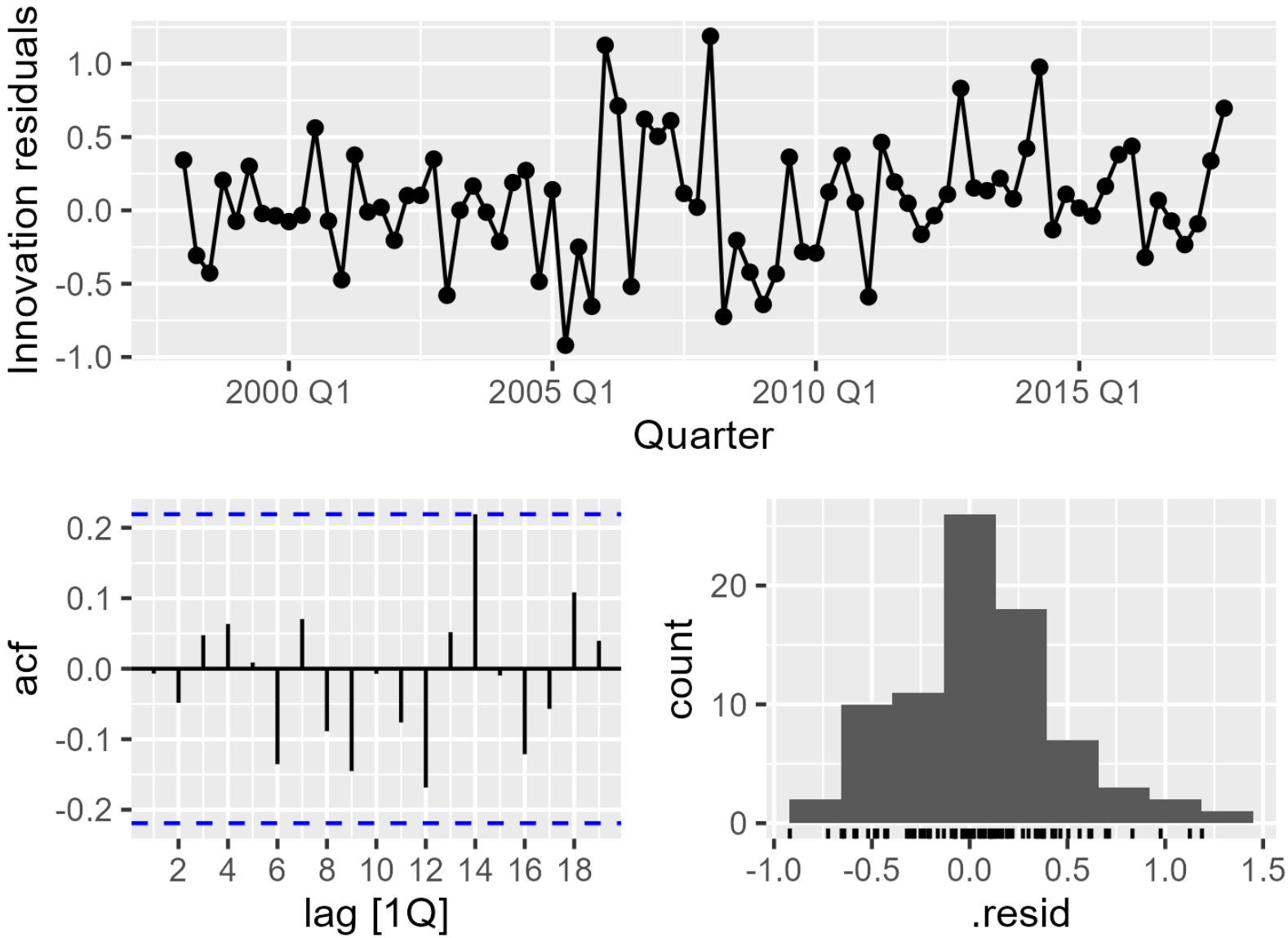
$$\text{Season equation: } s_t = \gamma * \frac{y_t}{l_{t-1} + b_{t-1}} + (1-\gamma) * s_{t-m}$$

time t	y_t	l_t	b_t	s_t	$\hat{y}_{t+1\|t}$
0	-	9.88	-0.028	1.16, 0.968, 0.926, 0.942	-
1	11.8	9.924	-0.026	1.160	9.581
2	9.28	9.828	-0.028	0.968	9.075
3	8.64	9.696	-0.031	0.926	9.104
4	9.3	9.711	-0.030	0.942	11.230

- Using the *components* function (that we have used previously for visualizing decompositions) we can look at the value of the level, trend and season for the models.
- We can see that in both cases the seasonal components look almost identical. This is due to the very small value of γ.
- The trend does not change much either (it does seem like it changes a lot, but look at the scales, compared to the level).

```
aus_holidays_model[1] |> gg_tsresiduals() #additive

aus_holidays_model[2] |> gg_tsresiduals() #multiplicative
```

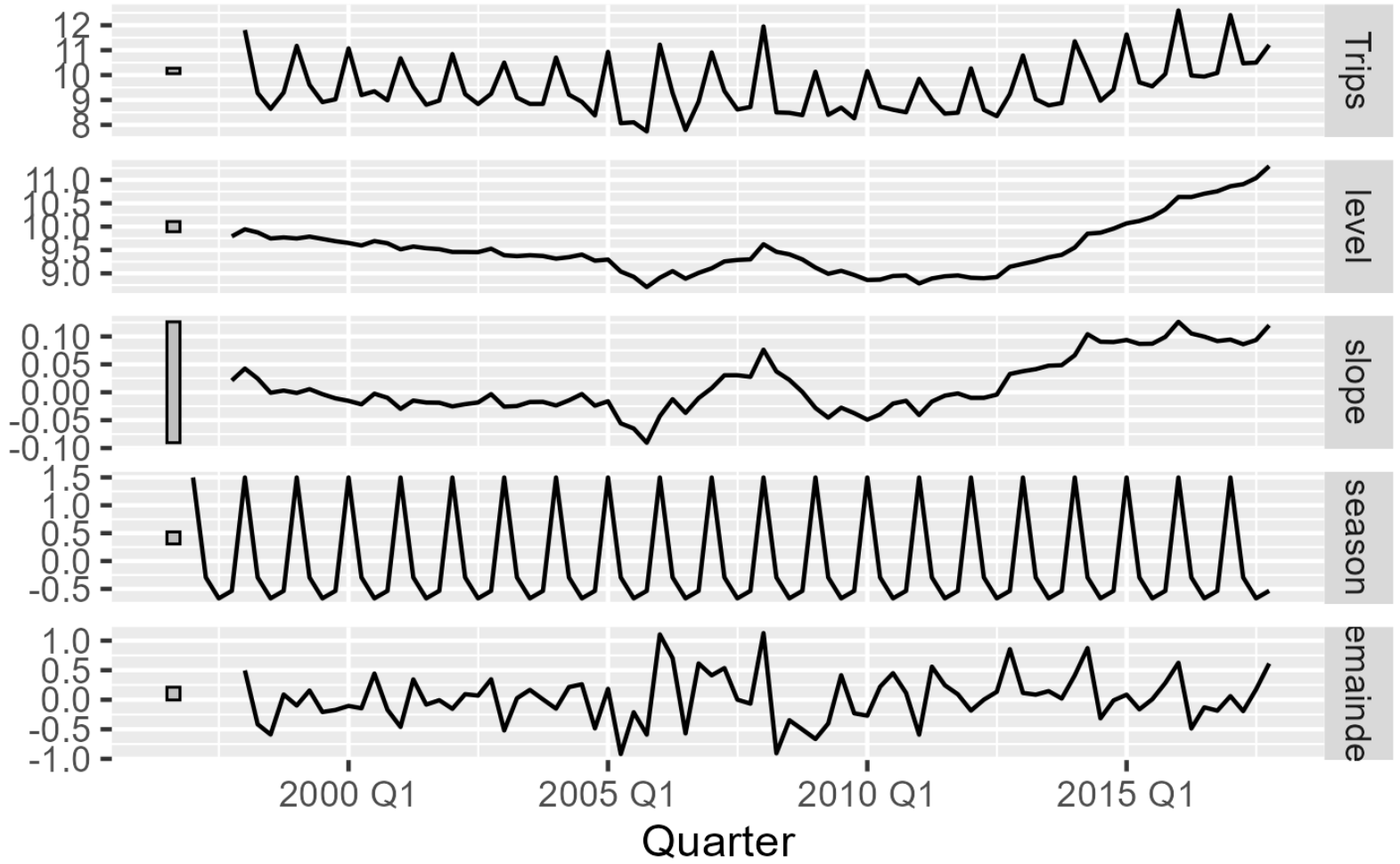

Innovation residuals
1.0
0.5
0.0
-0.5
-1.0
2000 Q1
2005 Q1
2010 Q1
2015 Q1
Quarter
acf
0.2
0.1
0.0
-0.1
-0.2
2 4 6 8 10 12 14 16 18
lag [1Q]
count
15
10
5
0
-1.0 -0.5 0.0 0.5 1.0
.resid


Innovation residuals
1.0
0.5
0.0
-0.5
-1.0
2000 Q1
2005 Q1
2010 Q1
2015 Q1
Quarter
acf
0.2
0.1
0.0
-0.1
-0.2
2 4 6 8 10 12 14 16 18
lag [1Q]
count
20
10
0
-1.0 -0.5 0.0 0.5 1.0 1.5
.resid

```
aus_holidays_model[1] |>
  components() |> autoplot()

comp <- aus_holidays_model[1] |> components()
view(comp)
```

ETS(A,A,A) decomposition

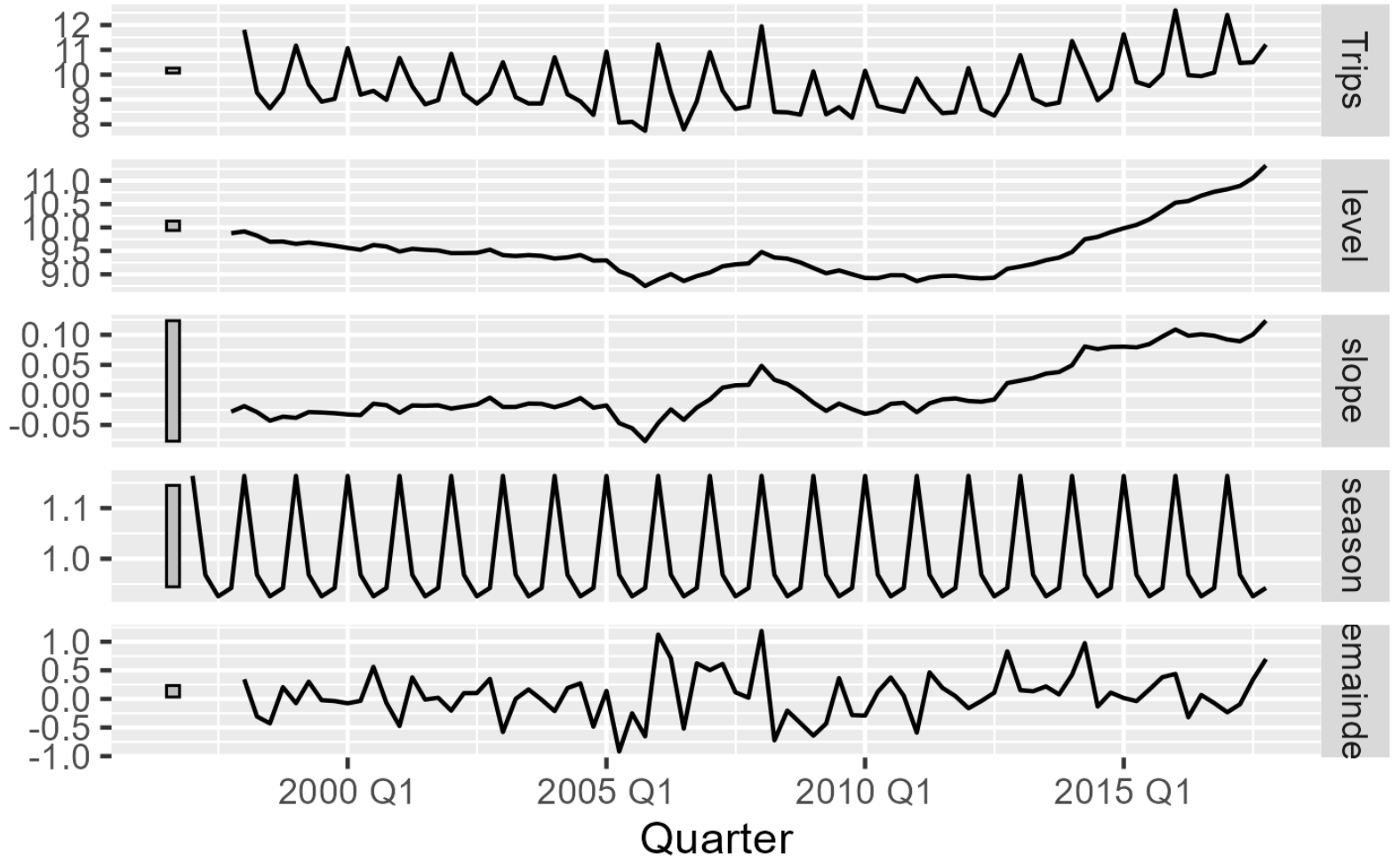
Trips = lag(level, 1) + lag(slope, 1) + lag(season, 4) + remainder



```
aus_holidays_model[2] |>
  components() |>
  autoplot()

accuracy(aus_holidays_model) # accuracy computed on training data
```

ETS(A,A,M) decomposition

Trips = (lag(level, 1) + lag(slope, 1)) * lag(season, 4) + remainder



	.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	additive	Training	0.0287	0.417	0.321	0.132	3.38	0.774	0.772	-0.0492
2	multiplicative	Training	0.0473	0.426	0.324	0.301	3.38	0.781	0.790	-0.0681

Holt-Winters' damped method

- Even if we have added a seasonal component, the trend component is still constant and will continue increasing (decreasing) forever.
- To avoid this, we can use damping with both additive and multiplicative Holt-Winters models.
- One often used version is damping with the multiplicative model, which has the following equations:

Forecast equation: $\hat{y}_{t+h|t} = [l_t + (\phi + \phi^2 + ... + \phi^h) * b_t] * s_{t+1-m(k+1)}$

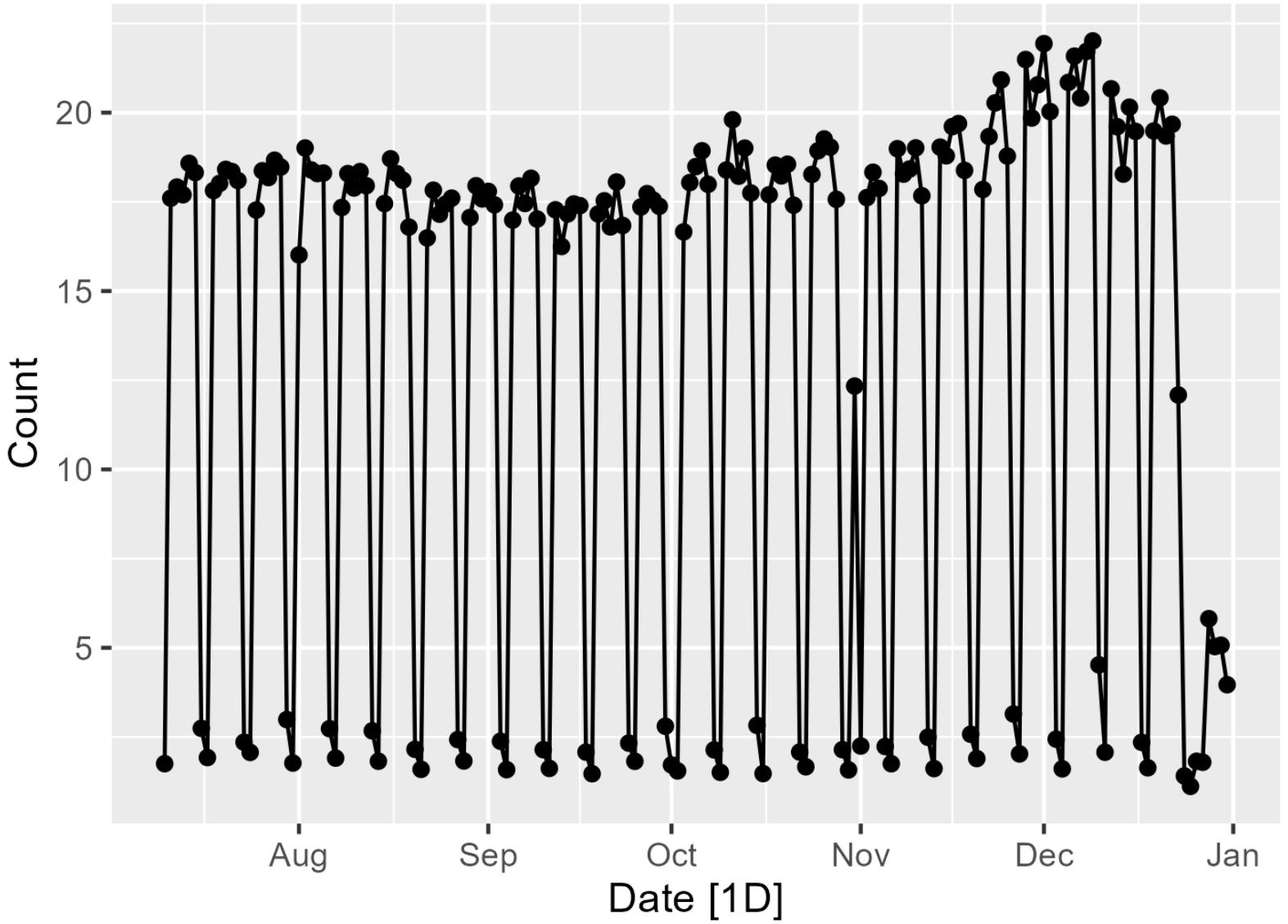
$$\text{Level equation: } \alpha * \frac{y_t}{s_{t-m}} + (1-\alpha) * (l_{t-1} + \phi * b_{t-1})$$

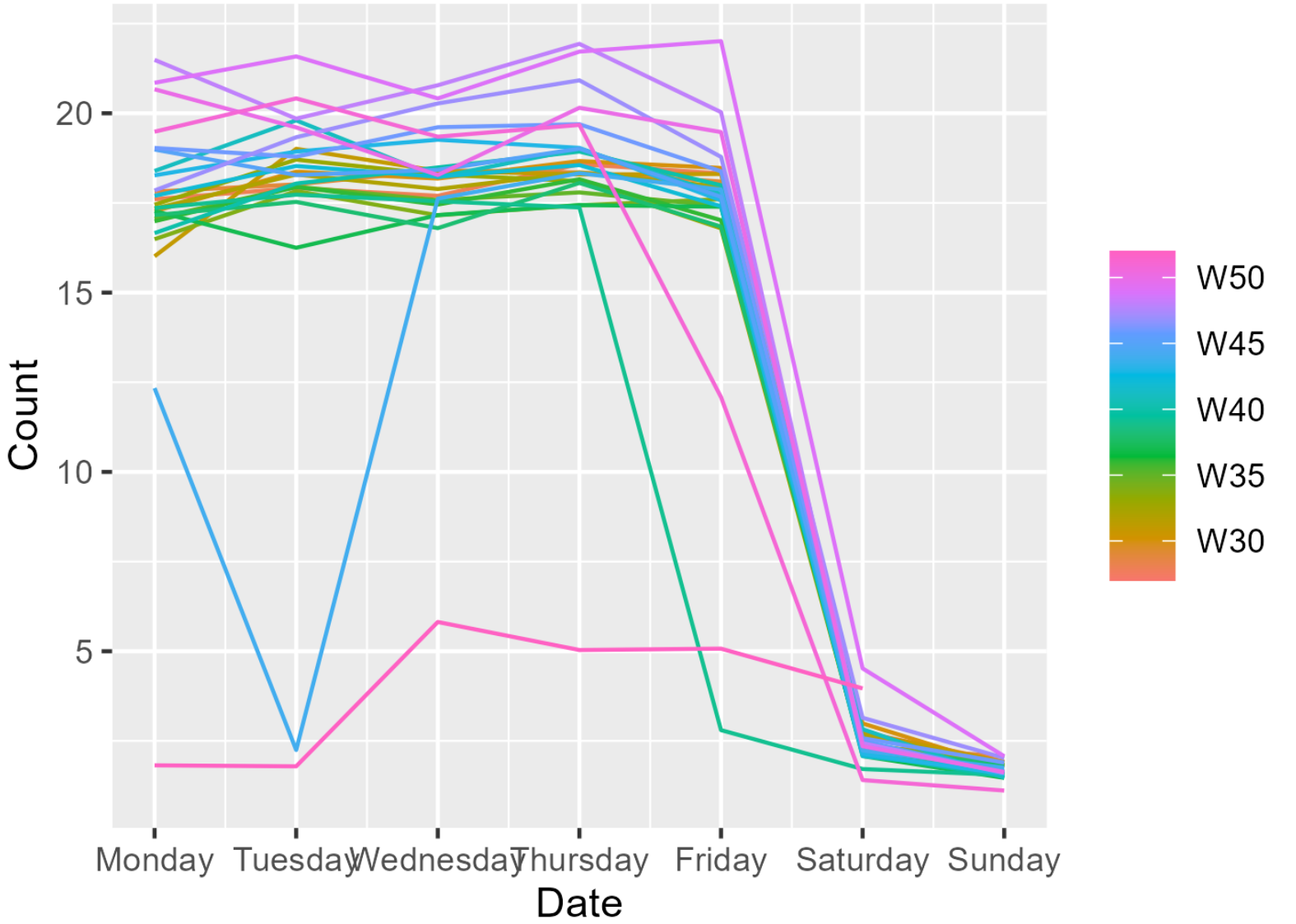
$$\text{Trend equation: } b_t = \beta^* * (l_t - l_{t-1}) + (1-\beta^*) * \phi * b_{t-1}$$

$$\text{Seasonal equation: } s_t = \gamma * \frac{y_t}{l_{t-1} + \phi * b_{t-1}} + (1-\gamma) * s_{t-m}$$

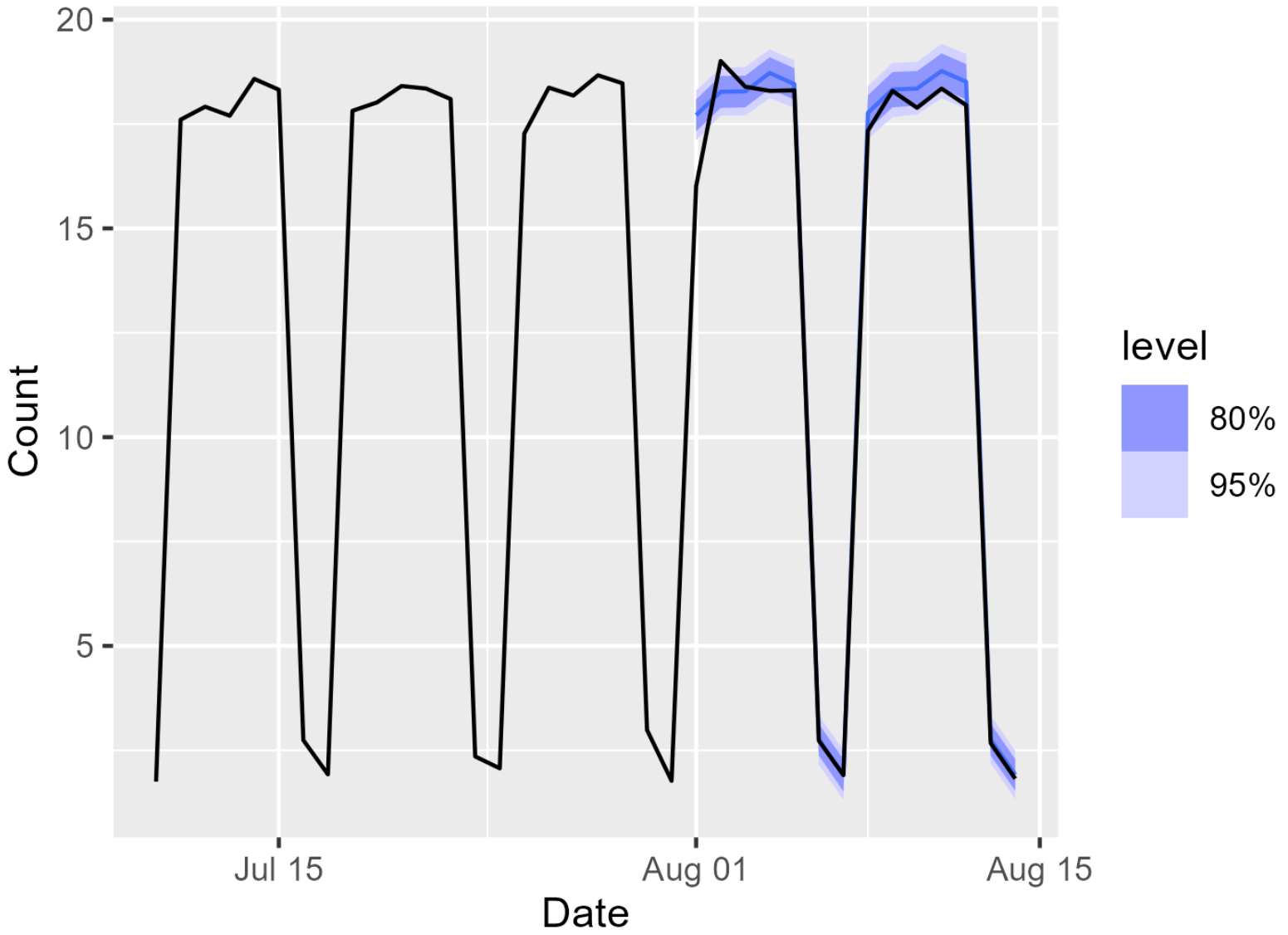
- Let's take as an example the data about pedestrian traffic. It contains hourly data for 2015 and 2016, but we will aggregate it to be daily (so we will consider a season of 7 days) and work with 2016 July and part of August.

```
pedestrian #two years, hourly data

#pick second half of 2016 and aggregate into daily
pedestrian_short <- pedestrian |>
  filter(Date >= "2016-07-10", Sensor == "Southern Cross Station") |>
  index_by (Date) |>
  summarize(Count = sum(Count)/ 1000)

pedestrian_short  |> autoplot(Count) + geom_point()

pedestrian_short |> gg_season(period = "week")  #remember seasonal plot?

pedestrian_train <- pedestrian_short |>
  filter(Date <= "2016-07-31")
pedestrian_model <- pedestrian_train |>
  model(ETS(Count ~ error("A") + trend("Ad") + season("M"))) #Experiment with M
     and A
pedestrian_forecast <- pedestrian_model |>
  forecast(h = "2 weeks")
```


20
15
10
5
Count
Aug
Sep
Oct
Nov
Dec
Jan
Date [1D]


Count
20
15
10
5
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Date
W50
W45
W40
W35
W30

```
pedestrian_forecast |> autoplot(pedestrian_short |>
  filter(Date <= "2016-08-14"))

pedestrian_test <-pedestrian_short |>
  filter(Date >= "2016-08-01", Date <= "2016-08-14")

accuracy(pedestrian_forecast, data = pedestrian_test)
```


20
15
10
5
Count
Jul 15
Aug 01
Aug 15
Date
level
80%
95%

```
  .model                                                                .type  RMSE   MAE  MAPE  MASE
  <chr>                                                                 <chr> <dbl> <dbl> <dbl> <dbl>
1 "ETS(Count ~ error(\"A\") + trend(\"Ad\") + season(\"M\"))" Test   0.568 0.371  2.72   NaN
```

Taxonomy of methods

- We have seen that both the trend and the season special can be of different types (Additive, Multiplicative, etc.). Considering all possibilities, there are 9 combinations:

Trend Component	Seasonal Component		
	N (None)	A (Additive)	M (Multiplicative)
N (None)	(N, N)	(N, A)	(N, M)
A (Additive)	(A, N)	(A, A)	(A, M)
Ad (Additive damped)	(A_d, N)	(A_d, A)	(A_d, M)

- Some of them were discussed previously and have special names:
  - (N, N) - simple exponential smoothing
  - (A, N) - Holt's linear method
  - (A_d, N) - Additive damped trend methods
  - (A, A) - Additive Holt-Winters' method
  - (A, M) - Multiplicative Holt-Winters' method
  - (A_d, M) - Holt-Winters' damped method

State space model I

- We have talked about exponential smoothing methods, which generate forecasts. Each has an underlying statistical model, which generates the forecast and the prediction interval as well.
- Each model consists of measurement equations, which describe the observed data, and state equations which describe the unobserved states (level, trend, season). This is why they are called *state space models*.
- We have talked about how the trend can be None, additive, or damped and how the season can be multiplicative and additive and none, but we did not talk about the error, which can be Additive or Multiplicative as well.
- For each method, we can have either Additive or Multiplicative errors. They generate the same point forecast, but different prediction intervals.

State space model II

- To distinguish between the two types of errors, we will introduce a third letter to the previous notation, and label each state space model with $ETS(\cdot, \cdot, \cdot)$, for (Error, Trend, Season).
- For example, ETS(A, N, N) is the simple exponential smoothing with additive errors. Let's look at this model in more detail.

State space model III

- Remember, the component form for SES was

$$\hat{y}_{t+1|t} = l_t$$

$$l_t = \alpha * y_t + (1 - \alpha) * l_{t-1}$$

- Let's reorganize the second equation a little:

$$l_t = \alpha y_t + l_{t-1} - \alpha * l_{t-1} \Rightarrow l_t = l_{t-1} + \alpha(y_t - l_{t-1})$$

- Now, l_{t-1} can be seen as $\hat{y}_{t|t-1}$, which means that $y_t - l_{t-1}$ is actually the residual e_t. So we have

$$l_t = l_{t-1} + \alpha * e_t$$

- This means that the training data error is used for adjusting the estimated level through the smoothing process. The higher the value of α the bigger the adjustment, and the rougher the estimate of the level.

State space model IV

- We can also consider that:

$$y_t = l_{t-1} + e_t$$

- By specifying the probability distribution for e_t, the previous equation becomes a state space model. For additive errors, we assume that the residuals are normally distributed white noise with mean 0 and variance σ^2, denoted by $e_t = \epsilon_t \sim NID(0, \sigma^2)$ (normally and independently distributed).
- Then we will have:

$$\text{measurement/observation equation:} y_t = l_{t-1} + \epsilon_t$$

$$\text{state/transition equation:} l_t = l_{t-1} + \alpha * \epsilon_t$$

- These two equations and the statistical distribution of the errors form the fully specified statistical model.

State space model V

- The measurement equation shows the relation between the observed value and the unobserved state. In this case it is a linear relation.
- The state equation shows how the state evolves through time.
- Similarly, the state space models can be defined for the other methods.

State space model VI

- For the ETS (M, N, N) model (simple exponential smoothing with multiplicative error) the state space model is:

$$y_t = l_{t-1} * (1 - \epsilon_t)$$

$$l_t = l_{t-1} * (1 + \alpha * \epsilon_t)$$

- For the ETS (A, A, N) model (Holt's linear method with additive errors) the state space model is:

$$y_t = l_{t-1} + b_{t-1} + \epsilon_t$$

$$l_t = l_{t-1} + b_{t-1} + \alpha * \epsilon_t$$

$$b_t = b_{t-1} + \beta * \epsilon_t$$

- where $\beta = \alpha * \beta^*$
- and so on...

Forecasting with ETS models

- When we want to forecast, we can do it by setting all errors, ϵ_t to be equal to 0.
- For example, the model for ETS(A, A, N) was:

$$y_{T+1} = l_T + b_T + \epsilon_{T+1} \Rightarrow$$

$$\hat{y}_{T+1|T} = l_T + b_T$$

- Similarly,

$$y_{T+2} = l_{T+1} + b_{T+1} + \epsilon_{T+2} =$$

$$= l_T + b_T + \alpha * \epsilon_{T+1} + b_T + \beta * \epsilon_{T+1} + \epsilon_{T+2}$$

- Therefore,

$$\hat{y}_{T+2|T} = l_T + 2 * b_T$$

- This forecast is the same as the one we had at Holt's linear method.

Prediction intervals

- The statistical models will give us prediction intervals as well. For most ETS models, the prediction interval is:

$$\hat{y}_{T+h|T} \pm c * \sigma_h$$

- where c is a constant which depends on the coverage probability and σ_h^2 is the forecast variance.
- For most additive error models the prediction intervals can be computed by some formula, for example:

$$\text{(A, N, N) model: } \sigma_h^2 = \sigma^2[2 + \alpha^2 * (h-1)]$$

$$\text{(A, A, N) model: } \sigma_h^2 = \sigma^2[1+(h-1)\{\alpha^2+\alpha\beta h+\frac{1}{6}\beta^2 h(2h-1)\}]$$

- where σ is the residual variance.

Selecting the best model

- While not every model fits any type of time series (for example, you should not use Holt-Winters' if you have no seasonality), in general more than one possible model can be used and we might need to select the best of them.
- We can use information criteria for comparison: AIC, AIC_c and BIC (discussed in more detail at regression models).
- The *report* function used for a model will provide these values.
- However, if we use the ETS function, without specifying what components to use, it will automatically select the best model.

Example I

- Let's take a look at some time series that we have used previously:
  - aus_holidays
  - pedestrians
  - aus_economy

```
aus_holidays <- tourism |>
  filter(Purpose == "Holiday") |>
  summarize(Trips = sum(Trips)/1e3)

aus_holidays_model <- aus_holidays |>
  model(ETS(Trips))
report(aus_holidays_model)
```

Example II

```
Series: Trips
Model: ETS(M,N,A)
  Smoothing parameters:
    alpha = 0.3484054
    gamma = 0.0001000018

  Initial states:
     l[0]       s[0]      s[-1]      s[-2]    s[-3]
 9.727072 -0.5376106 -0.6884343 -0.2933663 1.519411

  sigma^2:  0.0022

     AIC     AICc      BIC
226.2289 227.7845 242.9031
```

- We can see that the best model is ETS(M, N, A), it has AIC 226, AIC_c 228 and BIC 243.
- We have used this time series as an example at Holt-Winters' and we built an additive and a multiplicative model, with the following evaluation values:
  - Additive: AIC 229, AIC_c 231 and BIC 250.
  - Multiplicative: AIC 228, AIC_c 230 and BIC 245.
- Since all three measures should be minimized, we can see that the ETS(M, N, A) model is indeed a better fit.

```r
pedestrian_short <- pedestrian |>
  filter(Date >= "2016-07-10", Sensor == "Southern Cross Station") |>
  index_by (Date) |>
  summarize(Count = sum(Count)/ 1000)
pedestrian_model <- pedestrian_short |>
  model(ETS(Count))
pedestrian_model |> report()
```

```
Series: Count
Model: ETS(M,N,M)
  Smoothing parameters:
    alpha = 0.0441892
    gamma = 0.0001000137

  Initial states:
     l[0]      s[0]   s[-1]    s[-2]    s[-3]    s[-4]    s[-5]     s[-6]
 13.83834 0.1970596 1.30524 1.368108 1.348718 1.349004 1.305631 0.1262381

  sigma^2:  0.0436

     AIC     AICc      BIC
1160.842 1162.183 1192.489
```

- The best model is ETS(M, N, M) with AIC 1161, AIC_c 1162 and BIC 1192.

- We have used this data previously when talking about damped Holt-Winters', the ETS(A, Ad, M) has AIC 1196, AIC_c 1198 and BIC 1237, while ETS(M, Ad, M) has AIC 1165, AIC_c 1167 and BIC 1206.

```r
aus_economy <- global_economy |>
  filter(Code == "AUS") |>
  select(Population) |>
  mutate(Pop = Population / 1e6) #just to work with smaller values

aus_economy_model <- aus_economy |>
  model(ETS(Pop))
aus_economy_model |> report()
```

```
Series: Pop
Model: ETS(A,A,N)
  Smoothing parameters:
    alpha = 0.9999
    beta  = 0.3266366

  Initial states:
     l[0]      b[0]
 10.05414 0.2224818

  sigma^2:  0.0041

      AIC      AICc       BIC
-76.98569 -75.83184 -66.68347
```

- The best model is ETS(A, A, N), which is exactly the one we have used previously.

Observations about model selection

- There are some combinations for the model that can lead to instabilities, due to division by values which are potentially close to 0. These are: ETS(A, N, M), ETS(A, A, M) and ETS(A, Ad, M). These should not be considered when selecting a model.
- Models with multiplicative error are stable only if the data contains only strictly positive points.

ARIMA models

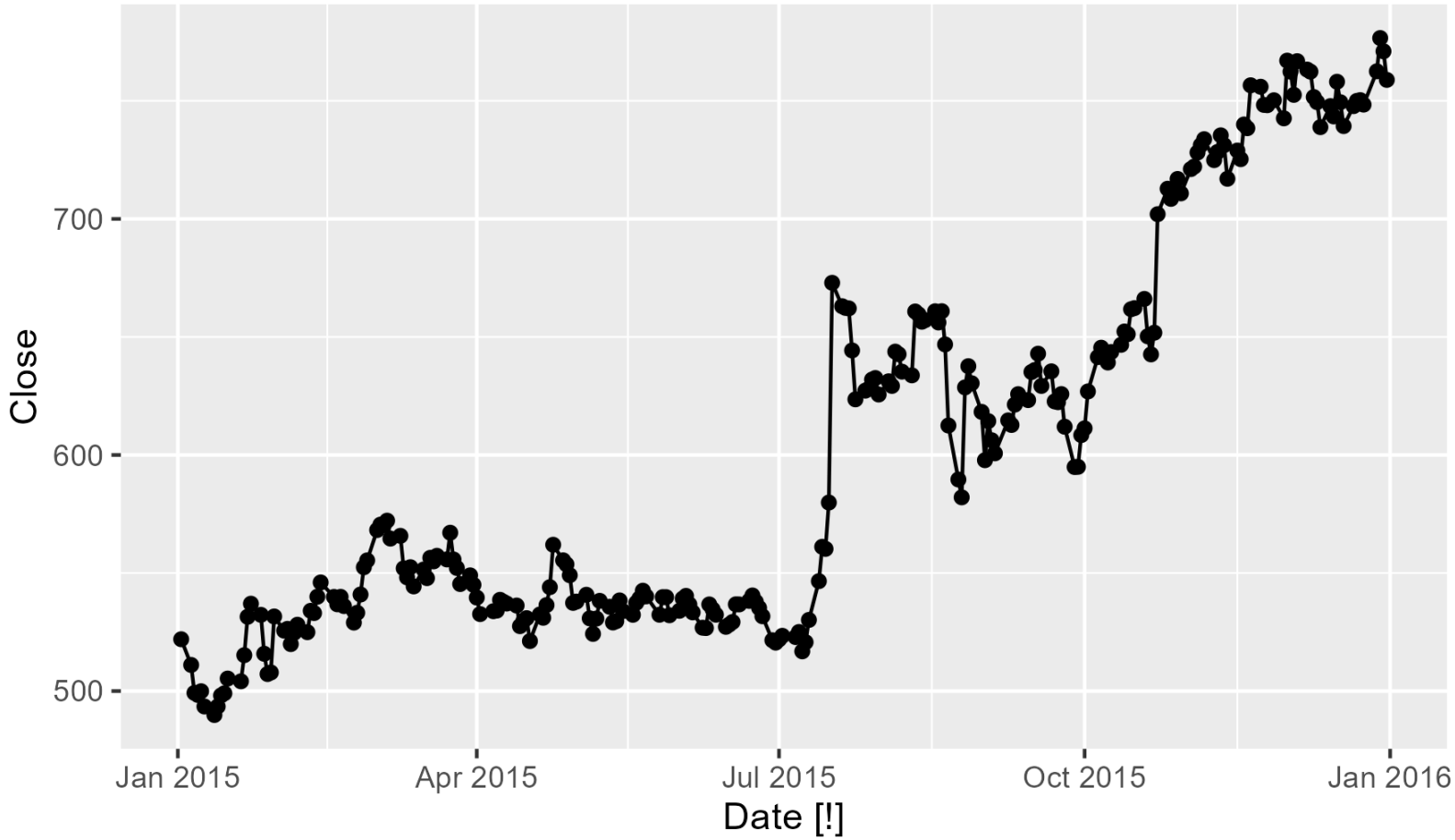
- Besides Exponential Smoothing, ARIMA models are the most widely used approaches for forecasting.
- While Exponential Smoothing focuses on describing the trend and seasonality in data, ARIMA models try to describe the autocorrelation in the data.

Stationarity and differencing - recap

- A time series is stationary if its statistical properties (mean, variance, etc.) do not depend on the time t when it is observed.
- This means that time series with trend and season are not stationary, but white noise is.
- Interestingly, cyclic time series are stationary, since the lengths of the cycles are not equal.
- In general, a stationary time series has a plot which is approximately horizontal and has constant variance (except in case of cycles).
- Besides the time series plot, you can recognize non-stationary time series by their ACF plot as well: the ACF of non-stationary time series decreases slowly (and r_1 is often positive and large), while for stationary time series we quickly get to near zero autocorrelation.

- One way to make non-stationary time series to be stationary is to *difference* it, take the differences of consecutive observations.
- Differencing helps to stabilize the mean of the time series.
- In *R* we can use the *difference* function to difference a time series.
- Another transformation, taking the logarithm, could help in stabilizing the variance of the time series.

```
google_2015 <- gafa_stock |>
  filter(Symbol == "GOOG", year(Date) == 2015)

google_2015 |> autoplot(Close) + geom_point()

google_2015 |>
  ACF(Close) |>
  autoplot()

google_2015 |> autoplot(difference(Close))

google_2015 |>
  ACF(difference(Close)) |>
  autoplot()

#is the differenced data really white noise?
google_2015 |>
  mutate(diff = difference(Close)) |>
  features(diff, ljung_box, lag = 10)
#p = 0.637 => yes, it is.
```



	Symbol	lb_stat	lb_pvalue
	<chr>	<dbl>	<dbl>
1	GOOG	7.91	0.637

```
#check the non-differenced data
google_2015 |>
  features(Close, ljung_box, lag = 10) # this is not white noise
```

```
  Symbol lb_stat lb_pvalue
  <chr>    <dbl>     <dbl>
1 GOOG     2083.         0
```

- Differencing can be repeated, if we consider that the result is not stationary yet. We can have second-order differencing.
- We can also have seasonal differencing, where we take the difference between an observation and the previous observation from the same season.
- The *unitroot_kpss* feature will perform the KPSS statistical test to determine if the time series is stationary.
- The *unitroot_ndiffs* and *unitroot_nsdiffs* functions will tell you how many differencing and seasonal differencing is required.
- Getting the right number of differencing is important, doing it too many times might introduce false dynamics and autocorrelations.