Day 8 - Holt-Winters Filtering

Introduction

Today, we add complexity to the exponential smoothing model that allows us to obtain forecasts for series that show signs of seasonality and trend. To guide our discussion, we will use a data set describing the amount of PM2.5 in the air in Bozeman, Montana during September of 2020. It may be helpful to know that PM2.5 is defined as small particulate matter in the air measuring 2.5 micrometers or less in diameter, and that there was a significant fire immediately outside Bozeman that began on 2020-09-04. You can see more about the fire in this YouTube video.

> **!** Important
>
> The raw data had 11 hours that were missing measurements. For this set of notes, I have imputed these values using time series techniques that we will cover later in the semester.

```
# packages
rm(list = ls())
library(tidyverse)

# read in the .rds file that I created
mt_pm_sept2020 <- readRDS("mt_pm25_sept2020.rds")

# clean the data and construct a ts object
mt_pm_clean <- mt_pm_sept2020 %>%
  mutate(dt = ymd_hms(datetime)) %>%
  dplyr::select(dt, rawvalue, everything()) %>%
  arrange(dt)

# create two ts
boz_ts_full <- ts(
  mt_pm_clean$rawvalue,
  start = c(1, 5),
  end = c(30, 5),
  freq = 24
)

boz_ts_red <- window(
  boz_ts_full,
```

```
  start = c(1, 5),
  end = c(29, 5)
)
```

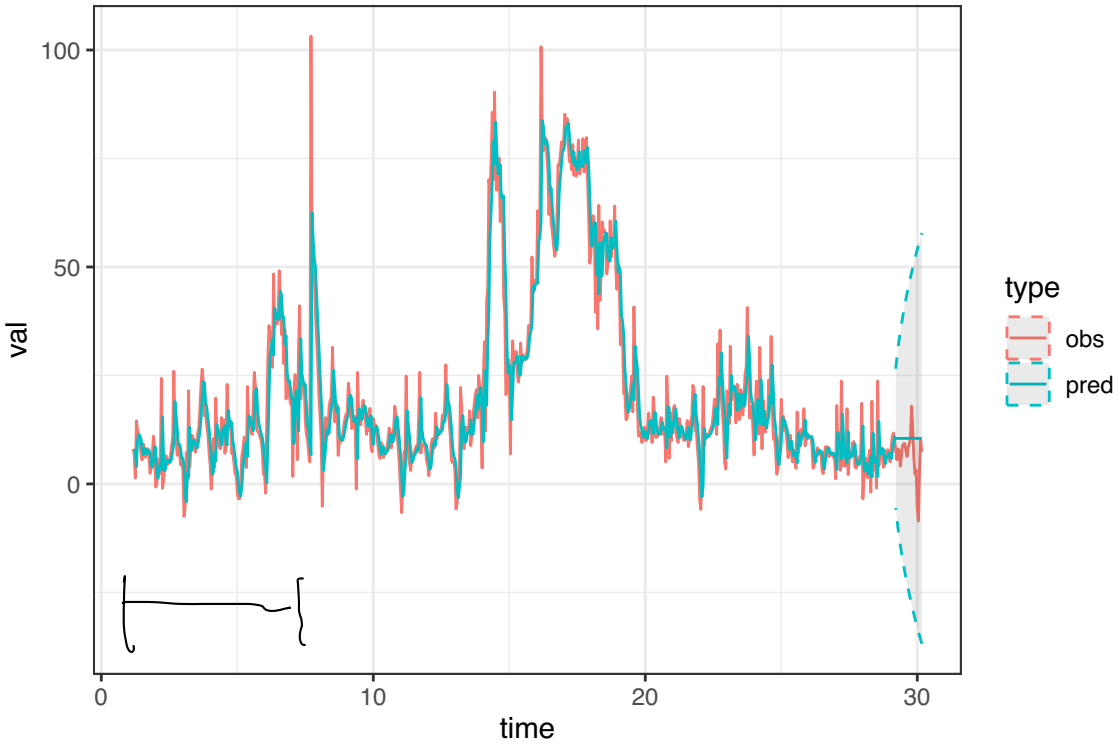
Review

Fit an exponential smoothing model to the `boz_ts_red` series, and use the model to forecast 24 hours ahead, and plot the observed time series, fitted time series, and forecasted time series on the same plot. How did the model do?

```
hw <- HoltWinters(
  boz_ts_red,
  beta = FALSE,
  gamma = FALSE
)
hw_pred <- predict(hw, n.ahead = 24, prediction.interval = TRUE)

# plot
bind_rows(
  tibble(
    time = time(boz_ts_full),
    val = c(boz_ts_full)
  ) %>% mutate(type = "obs"),
  bind_rows(
    tibble(
      time = time(hw$fitted[,1]),
      val = c(hw$fitted[,1])
    ),
    tibble(
      time = time(hw_pred),
      val = c(hw_pred[,1]),
      lwr = c(hw_pred[,3]),
      upr = c(hw_pred[,2])
    )
  )  %>% mutate(type = "pred")
) %>%
  ggplot(aes(x = time, y = val, col = type)) +
  geom_line() +
  geom_ribbon(
    aes(ymin = lwr, ymax = upr),
    linetype = 2,
```

```
  alpha = 0.1
) +
theme_bw()
```



The fitted values from the period we observe look pretty good. The prediction is really bad.

Holt-Winters filtering

Note

Holt-Winters Filtering extends the exponential smoothing model to accommodate trend terms and seasonal effects. For a series $\{x_t\}$ with additive seasonal effects and period p. The model remains

$$x_t = \mu_t + w_t$$

with a_t denoting our estimate of μ_t. However, the way we obtain a_t is a bit more complicated:

$$\begin{aligned} a_t &= \alpha(x_t - s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \\ b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(x_t - a_t) + (1-\gamma)s_{t-p} \end{aligned}$$

The forecasting equation is

$$\hat{x}_{n+k|n} = a_n + kb_n + s_{n+k-p}$$

for $k \leq p$.

seasonal period
$n = 12$, dec 2024
$k = 1$, jan 2025
$n+k-p = 12+1-12 = 1$: jan 2024

- a_t : level
- b_t : trend
- s_t : seasonal component

Note

For multiplicative seasonal effects, the estimates and forecasting equation are slightly different:

$$a_t = \alpha\left(\frac{x_t}{s_{t-p}}\right) + (1-\alpha)(a_{t-1} + b_{t-1})$$
$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$
$$s_t = \gamma\left(\frac{x_n}{a_n}\right) + (1-\gamma)s_{t-p}$$

The forecasting equation is

$$\hat{x}_{n+k|n} = (a_n + kb_n)s_{n+k-p}$$

for $k \leq p$. To fit the Holt-Winters model in R, we use the `HoltWinters` function. By default, R estimates the smoothing parameters by again minimizing the sum of square one-step ahead prediction errors.

Air Passengers example

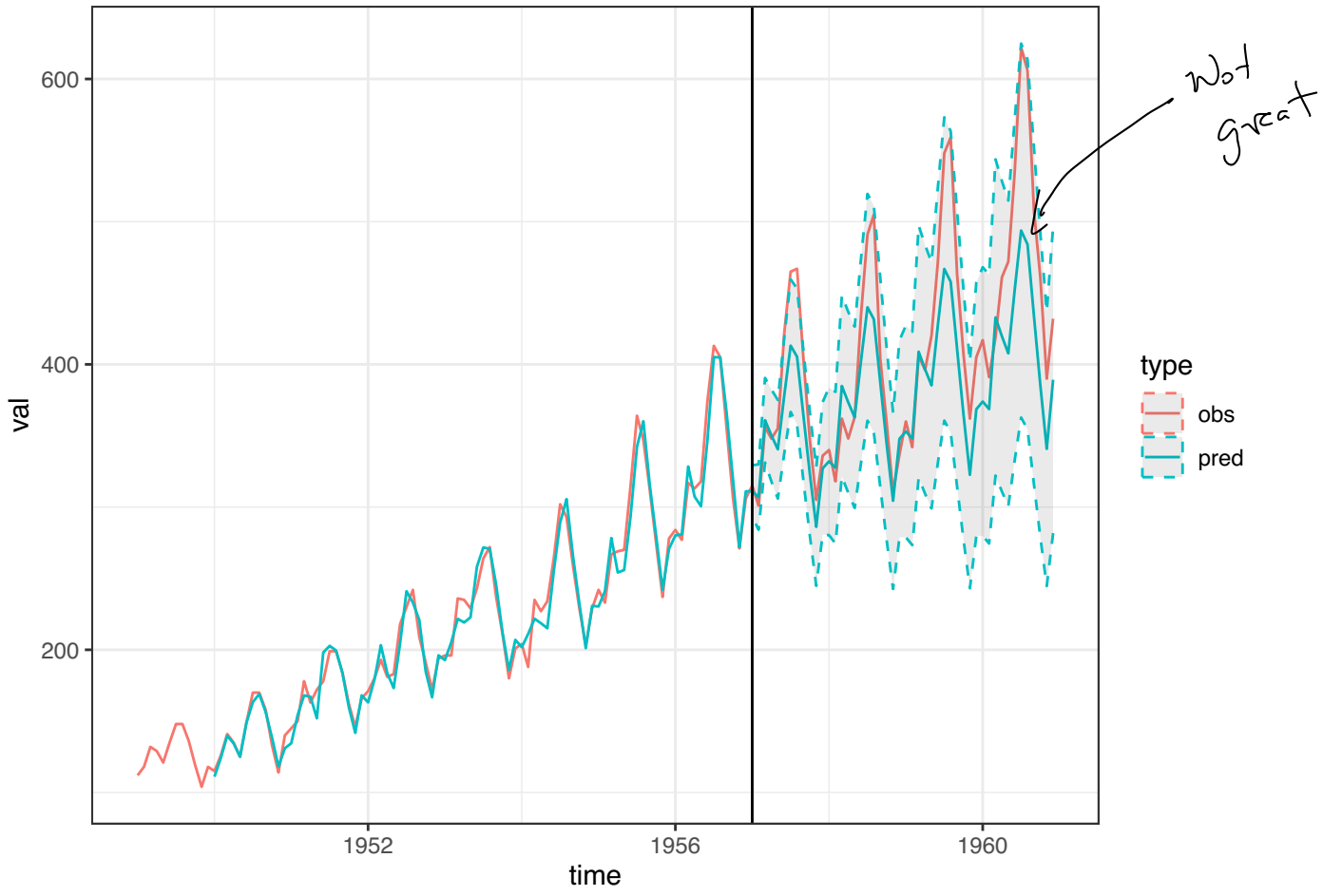
The code below creates a time series that excludes the last four years of the `AirPassengers` data. Fit a multiplicative Holt-Winters model to the reduced series and forecast four years into the future. Compare this forecast to the actual data from those four years.

```
# load data
ap_full <- AirPassengers
ap_red <- window(
  ap_full,
  start = c(1949, 1),
  end = c(1956, 12)
)

# fit model
hw <- HoltWinters(
  ap_red,
  seasonal = "mult"
)
hw_pred <- predict(hw, n.ahead = 4*12, prediction.interval = TRUE)

# plot
bind_rows(
  tibble(
    time = time(ap_full),
    val = c(ap_full)
  ) %>% mutate(type = "obs"),
  bind_rows(
    tibble(
      time = time(hw$fitted[,1]),
      val = c(hw$fitted[,1])
    ),
    tibble(
      time = time(hw_pred),
      val = c(hw_pred[,1]),
      lwr = c(hw_pred[,3]),
      upr = c(hw_pred[,2])
    )
  )  %>% mutate(type = "pred")
) %>%
  ggplot(aes(x = time, y = val, col = type)) +
  geom_line() +
  geom_ribbon(
    aes(ymin = lwr, ymax = upr),
```

```
    linetype = 2,
    alpha = 0.1
  ) +
  theme_bw() +
  geom_vline(aes(xintercept = 1957))
```



Function of forecasting so far ahead & the smoothing parameters!

> **Note**
>
> In the previous example, we pretended as if we did not observe the last four years of the time series, fit a model to rest of the data, and used that model to predict the data that we had omitted. This process is called **cross-validation**, and provides a way to assess the quality of the forecasts obtained from a model.

The code below calculates the sum of squared differences between the actual series and the forecasted series between 1957 and 1960. The resulting sum of squared forecast error may not be meaningful on its own, but it provides a mechanism to compare the forecasting capabilities of multiple models.

```
diffs <- window(
  ap_full,
  start = c(1957, 1),
  end = c(1960, 12)
) - hw_pred[,1]
sum(diffs^2)
```

```
[1] 111693.6
```

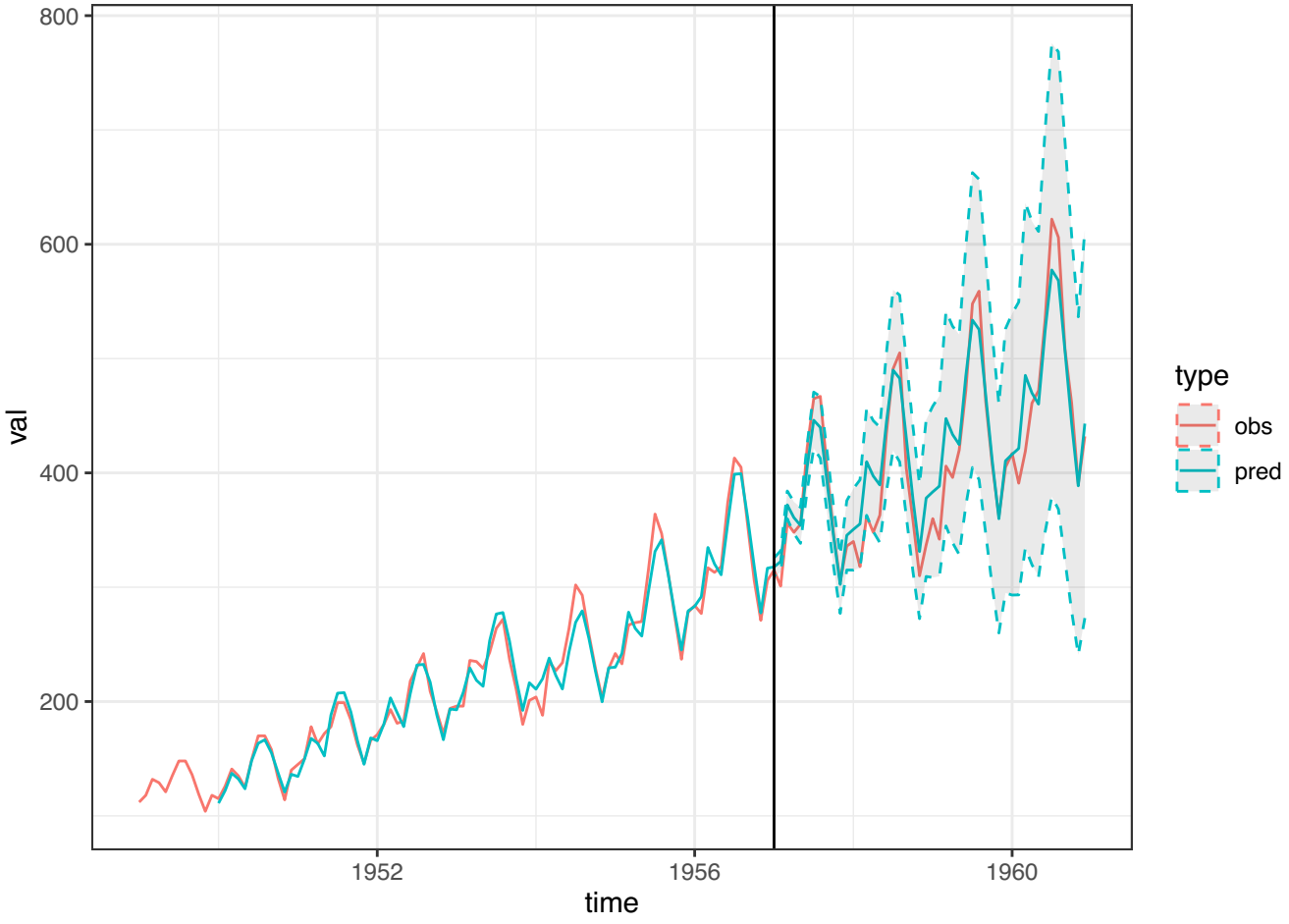
Fit another Holt-Winters model to `ap_red` time series, this time forcing α, β, and γ to all be 0.2. Plot the results and calculate the sum of squared forecast errors (SSFE). Compare this result to the SSFE for the first Holt-Winters fit. Which model resulted in better forecasts?

```
# fit model
hw2 <- HoltWinters(
  ap_red,
  seasonal = "mult",
  alpha = .2,
  beta = .2,
  gamma = .2
)
hw_pred2 <- predict(hw2, n.ahead = 4*12, prediction.interval = TRUE)

# plot
bind_rows(
  tibble(
    time = time(ap_full),
    val = c(ap_full)
  ) %>% mutate(type = "obs"),
  bind_rows(
    tibble(
      time = time(hw2$fitted[,1]),
      val = c(hw2$fitted[,1])
```

fix all 3 - try forcing $\alpha = .2$, letting R opt β, γ!

```
    ),
    tibble(
      time = time(hw_pred2),
      val = c(hw_pred2[,1]),
      lwr = c(hw_pred2[,3]),
      upr = c(hw_pred2[,2])
    )
  )  %>% mutate(type = "pred")
) %>%
  ggplot(aes(x = time, y = val, col = type)) +
  geom_line() +
  geom_ribbon(
    aes(ymin = lwr, ymax = upr),
    linetype = 2,
    alpha = 0.1
  ) +
  theme_bw() +
  geom_vline(aes(xintercept = 1957))
```



```r
diffs <- window(
  ap_full,
  start = c(1957, 1),
  end = c(1960, 12)
) - hw_pred2[,1]
sum(diffs^2)
```

```
[1] 30127.87
```

Avocados example

The code below creates a complete and reduced time series of conventional avocado prices in San Francisco. Fit two Holt-Winters models to the reduced time series: one that allows R to estimate the smoothing parameters, and one that sets the parameters to .2. Plot the results and comment on the quality of the forecasts from both models.

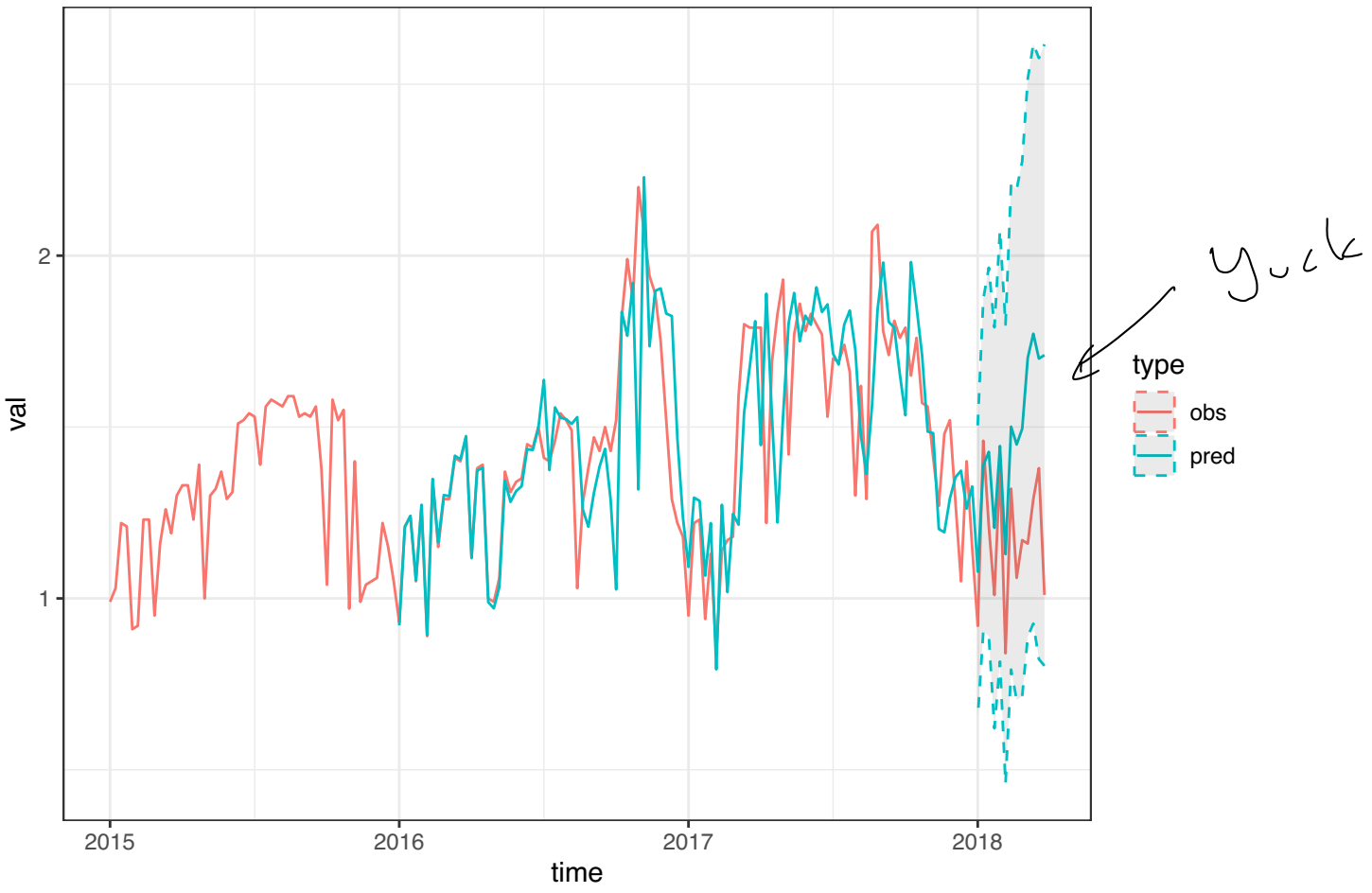
```
# load the data, sort by date and filter to conventional sales in SF
avocado <- readr::read_csv("avocado.csv") %>%
  arrange(Date) %>%
  filter(
    region == "SanFrancisco",
    type == "conventional"
  )

# complete ts
avo_ts <- ts(
  avocado$AveragePrice,
  start = with(avocado, c(year(Date[1]), week(Date[1]))),
  freq = 52
)

# reduced to test the model
avo_ts_red <- window(
  avo_ts,
  start = c(2015, 1),
  end = c(2017, 52)
)
```

```
# first model
avo_hw1 <- HoltWinters(
  avo_ts_red
)
avo_hw1_pred <- predict(
  avo_hw1,
  n.ahead = 13,
  prediction.interval = TRUE
)
bind_rows(
  tibble(
    time = time(avo_ts),
    val = c(avo_ts)
  ) %>% mutate(type = "obs"),
  bind_rows(
```

```
    tibble(
      time = time(avo_hw1$fitted[,1]),
      val = c(avo_hw1$fitted[,1])
    ),
    tibble(
      time = time(avo_hw1_pred),
      val = c(avo_hw1_pred[,1]),
      lwr = c(avo_hw1_pred[,3]),
      upr = c(avo_hw1_pred[,2])
    )
  )  %>% mutate(type = "pred")
) %>%
  ggplot(aes(x = time, y = val, col = type)) +
  geom_line() +
  geom_ribbon(
    aes(ymin = lwr, ymax = upr),
    linetype = 2,
    alpha = 0.1
  ) +
  theme_bw()
```

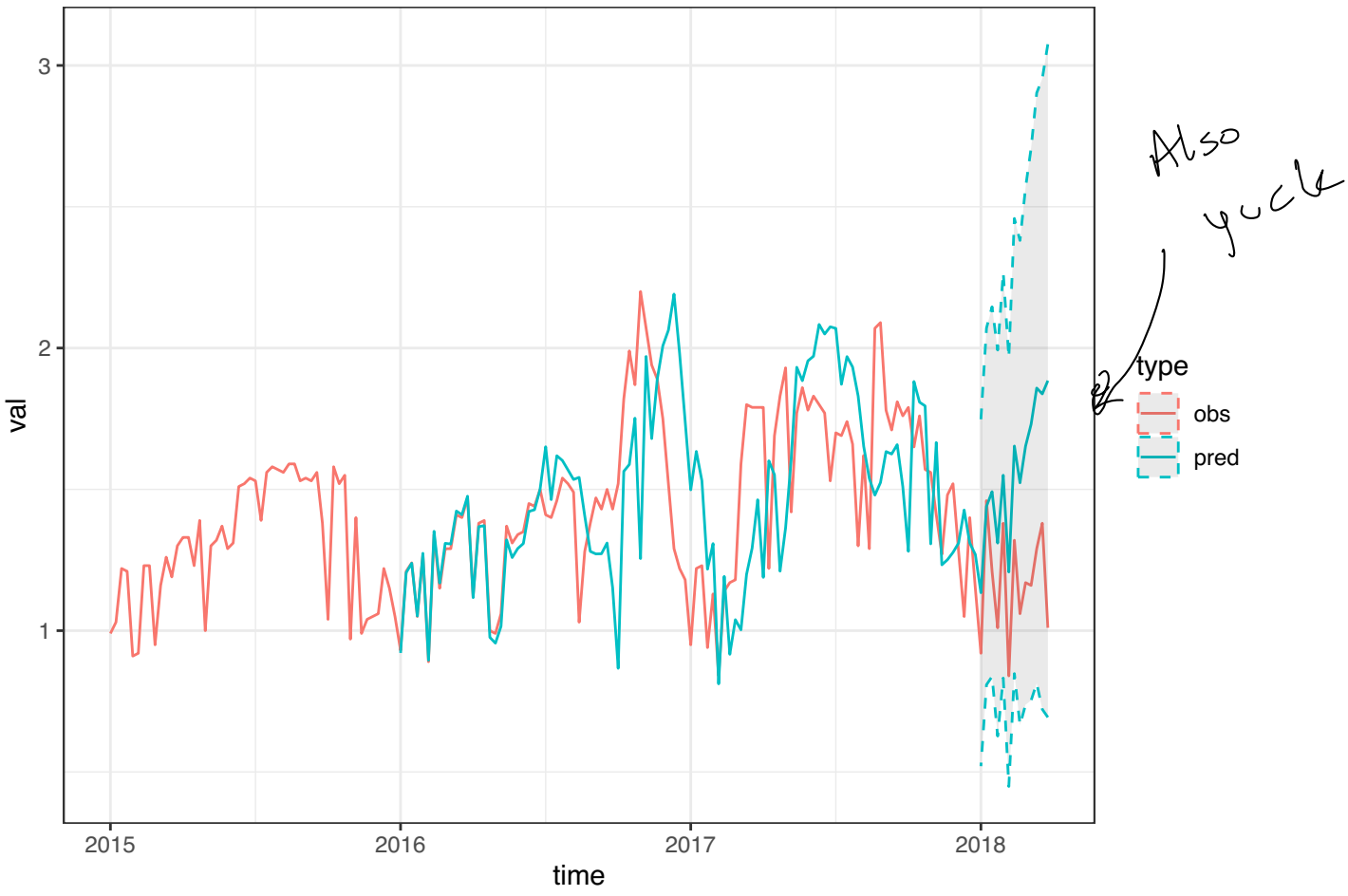



```r
# new model
avo_hw2 <- HoltWinters(
  avo_ts_red,
  alpha = .2,
  beta = .2,
  gamma = .2
)

# prediction
avo_hw2_pred <- predict(
  avo_hw2,
  n.ahead = 13,
  prediction.interval = TRUE
)

# plot
bind_rows(
  tibble(
```

```
    time = time(avo_ts),
    val = c(avo_ts)
  ) %>% mutate(type = "obs"),
  bind_rows(
    tibble(
      time = time(avo_hw2$fitted[,1]),
      val = c(avo_hw2$fitted[,1])
    ),
    tibble(
      time = time(avo_hw2_pred),
      val = c(avo_hw2_pred[,1]),
      lwr = c(avo_hw2_pred[,3]),
      upr = c(avo_hw2_pred[,2])
    )
  )  %>% mutate(type = "pred")
) %>%
  ggplot(aes(x = time, y = val, col = type)) +
  geom_line() +
  geom_ribbon(
    aes(ymin = lwr, ymax = upr),
    linetype = 2,
    alpha = 0.1
  ) +
  theme_bw()
```



Key takeaways

- The Holt-Winters method is far more flexible than exponential smoothing, allowing for forecasting with series that contain trends and seasonality.
- Holt-Winters, and other exponential smoothing procedures, can be slow to respond to changes in a series, particularly if the series are long.
- The way that R estimates the smoothing parameters for the Holt-Winters method can sometimes result in **overfitting** of the model to an observed time series, particularly if the series is longer and contains clear trend and seasonality. Setting the smoothing parameters to fixed values can *sometimes* result in better forecasting results, and 0.2 often performs well in particular.
- Cross-validation offers a technique to assess the quality of the forecasts from a model, and allows for comparison between models.
- Forecasting techniques assume that the observed trends and seasonal components continue through time. As a result, they only represent sensible predictions if those trends continue. Unforeseen future events can lead to poor forecasts, and we should be aware that forecasting necessarily requires us to extrapolate beyond the observed time period.