Outline

- Forecasting Revisited and the Solution of HW2
  1. Review of the HW2
- Optimize α and β.
  1. Numerical Simulation using Solver
  2. How to find the **best** tuning parameters practically?

[1] Forecasting Revisited and the Solution of HW2

Analytical Solution:

- Now, F is unknown.
- You need to use the estimated version of F, $\hat{F}$.

$$\underline{q^*} = \hat{F}^{-1}(TSL)$$

where

$$\underline{SL} = \frac{Cu}{Cu + Co} = \frac{r - c}{(r - c) + (c - s)} = \frac{r - c}{\underline{r - s}},$$

and $\hat{F}$ is the CDF of demand **based on forecast**.

- Now, we are forecasting demand, so $\hat{F}$ is not equal to F anymore.
- In the case of the additive demand, we can represent the solution as follows:

Safety stock

$$\underline{q^*} = \underline{\text{Forecasted Demand}} + \underline{\text{Forecasting error}}^{-1}(SL)$$

Descriptive Analysis

- Prepare Rstudio and load general packages to use
- Load csv/txt file

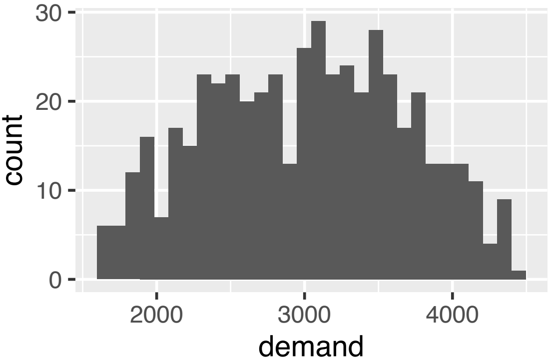
```
df <- read.csv("../data/demand_data_session3.txt", sep = "\t")

# Add time trend
df <- df %>%
  mutate(t = row_number()) %>%
  as_tibble()

# Descriptives
summary(df)
```

```
##        t              demand
##  Min.   :  1.0   Min.   :1606
##  1st Qu.:125.8   1st Qu.:2477
##  Median :250.5   Median :3045
##  Mean   :250.5   Mean   :3012
##  3rd Qu.:375.2   3rd Qu.:3528
##  Max.   :500.0   Max.   :4410
```

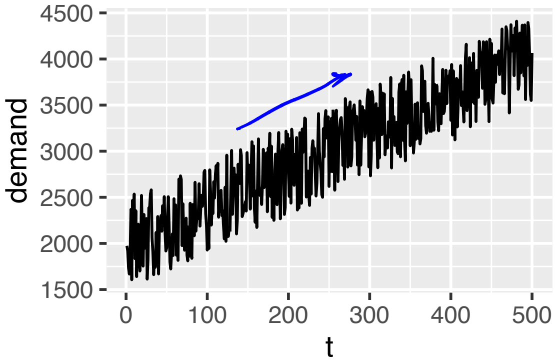
Visualize

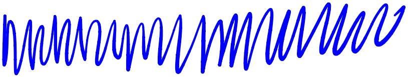
```
df %>%
  ggplot(aes(x = demand)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
df %>%
  ggplot(aes(x = t, y = demand)) +
  geom_line()
```





Setup

Set Exogenously Given Values as Constant

```r
price <- 4
cost <- 0.8
salvage <- 0

# What we know
SL <- (price - cost) / (price - salvage) # 0.8
```

$\frac{r-c}{r-s}$

Set Blank Table

```r
profit <- matrix(NA, nrow = nrow(df), ncol = 5)
colnames(profit) <- c("Simple Newsvendor", "Oracle", "Moving Average", "Simple ES", "Holt")
rownames(profit) <- paste0("day", 1:nrow(df))
head(profit)
```

```
##      Simple Newsvendor Oracle Moving Average Simple ES Holt
## day1                NA     NA             NA        NA   NA
## day2                NA     NA             NA        NA   NA
## day3                NA     NA             NA        NA   NA
## day4                NA     NA             NA        NA   NA
## day5                NA     NA             NA        NA   NA
## day6                NA     NA             NA        NA   NA
```

Simple Newsvendor

```r
for (i in 301:nrow(df)) {

  stocking_optimal_adaptive <- as.numeric(quantile(df$demand[1:(i-1)], SL))
  profit[i, "Simple Newsvendor"] <-
  α price * min(stocking_optimal_adaptive, df$demand[i]) -
  α cost * stocking_optimal_adaptive
}                                                     S*
```

Oracle

```r
for (i in 301:nrow(df)) {

  profit[i, "Oracle"] <- price * df$demand[i] - cost * df$demand[i]
}
```

Moving average

```
df <- df %>%
  dplyr::mutate(forecast_MA4 =
                  (dplyr::lag(demand, 4) +
                     dplyr::lag(demand, 3) +
                     dplyr::lag(demand, 2) +
                     dplyr::lag(demand, 1)) / 4)
df <- df %>%
  dplyr::mutate(forecast_error_MA4 = demand - forecast_MA4)

for (i in 301:nrow(df)) {

  safety_stock <- as.numeric(quantile(df$forecast_error_MA4[1:(i-1)], SL, na.rm = TRUE))
  stocking_optimal_adaptive <- df$forecast_MA4[i] + safety_stock
  profit[i, "Moving Average"] <-
    price * min(stocking_optimal_adaptive, df$demand[i]) -
    cost * stocking_optimal_adaptive
}
```

[2] Optimizing tuning parameters

- Optimizing forecast based on the previous data (say, day 101-300)
- Use the optimized α to cauculate the optimal stocking and average profit.
- Compare it with the naive one where we fixed α.

MSE

$$\frac{1}{n}\sqrt{\Sigma(x_i - \hat{x}_i)^2}$$

Demand

forecasted

$$\begin{array}{ccc} 1 & -1 & / \\ 1 & +1 & / \\ \hline 2 & & 0 \end{array}$$

Simple Exponential Smoothing (untuned)

```
L0 <- 2000
alpha <- 0.2
Lt <- c(); Ft <- c()
for (i in 1:nrow(df)) {
  if (i == 1) {
    Lt[i] <- alpha * df$demand[i] + (1 - alpha) * L0
    Ft[i] <- L0
  } else {
    Lt[i] <- alpha * df$demand[i] + (1 - alpha) * Lt[i-1]
    Ft[i] <- Lt[i-1]
  }
}
df <- df %>%
  dplyr::mutate(forecast_Simple_ES = Ft,
                forecast_error_Simple_ES = demand - forecast_Simple_ES)

for (i in 301:nrow(df)) {

  safety_stock <- as.numeric(quantile(df$forecast_error_Simple_ES[1:(i-1)], SL, na.rm = TRUE))
  stocking_optimal_adaptive <- df$forecast_Simple_ES[i] + safety_stock
  profit[i, "Simple ES"] <-
    price * min(stocking_optimal_adaptive, df$demand[i]) -
    cost * stocking_optimal_adaptive
}
```

```r
# Compare average profit by forecating method
apply(profit[301:500, ], 2, mean)
```

```
## Simple Newsvendor            Oracle     Moving Average          Simple ES
##          10583.16         11620.86           11272.74           11298.89
##              Holt
##                NA
```

Construct MSE function for Simple Exponential Smoothing

```
mse_simple_ES <- function(alpha, predict_from, predict_to, df) {

  df_predict <- df %>%
    dplyr::slice(predict_from:predict_to)
  df_true <- df %>%
    dplyr::slice(predict_from:predict_to)

  L0 <- 2000
  Lt <- c(); Ft <- c()
  for (i in 1:(predict_to - predict_from + 1)) {
    if (i == 1) {
      Lt[i] <- alpha * df_predict$demand[i] + (1 - alpha) * L0
      Ft[i] <- L0
    } else {
      Lt[i] <- alpha * df_predict$demand[i] + (1 - alpha) * Lt[i-1]
      Ft[i] <- Lt[i-1]
    }
  }

  mse <- sum((df_true$demand - Ft)^2) / (predict_to - predict_from + 1)
  return(mse)
}
```

Optimize alpha

- There are many ways/solvers to do this.
- To impose the restriction $\alpha \in [0, 1]$, we used **L-BFGS-B** here.
- Other solver such as **BFGS**, **Nelder-Mead** should work well even without any range restriction in this simple data.

```r
res <- optim(0.2, mse_simple_ES, method = "L-BFGS-B",
             predict_from = 1,
             predict_to = 300,
             df = df,
             lower = c(0),
             upper = c(1))
res
```

```
## $par
## [1] 0.08900983
##
## $value
## [1] 94325.41
##
## $counts
## function gradient
##       49       49
##
## $convergence
## [1] 52
##
## $message
## [1] "ERROR: ABNORMAL_TERMINATION_IN_LNSRCH"
```

Use the tuned values to calculate the optimal profit

```r
profit <- cbind(profit, rep(NA, nrow(df)))
colnames(profit)[6] <- "Simple ES (tuned)"

alpha <- res$par

L0 <- 2000
Lt <- c(); Ft <- c()
for (i in 1:nrow(df)) {
  if (i == 1) {
    Lt[i] <- alpha * df$demand[i] + (1 - alpha) * L0
    Ft[i] <- L0
  } else {
    Lt[i] <- alpha * df$demand[i] + (1 - alpha) * Lt[i-1]
    Ft[i] <- Lt[i-1]
  }
}
df <- df %>%
  dplyr::mutate(forecast_Simple_ES = Ft,
                forecast_error_Simple_ES = demand - forecast_Simple_ES)

for (i in 301:nrow(df)) {

  safety_stock <- as.numeric(quantile(df$forecast_error_Simple_ES[1:(i-1)], SL, na.rm = TRUE))
  stocking_optimal_adaptive <- df$forecast_Simple_ES[i] + safety_stock
  profit[i, "Simple ES (tuned)"] <-
    price * min(stocking_optimal_adaptive, df$demand[i]) -
    cost * stocking_optimal_adaptive
}
```

Comparison Table

```
# Compare average profit by forecating method
apply(profit[301:500, ], 2, mean)
```

```
## Simple Newsvendor            Oracle    Moving Average         Simple ES
##          10583.16          11620.86          11272.74          11298.89
##              Holt Simple ES (tuned)
##                NA          11311.68
```

Solution: Holt's

```
# Load data
df <- read.csv("../data/demand_data_session2.txt")

# Add time trend
df <- df %>%
  mutate(t = row_number()) %>%
  as_tibble()

# Setting
price <- 4
cost <- 0.8
salvage <- 0

# What we know
optimal_order_quantity <- as.integer(quantile(df$demand, (price - cost) / (price - salvage) ))
```

SL

```
# Make table for summary
res_table <- data.frame(
  method = c("Oracle",
             "holt_wo_tuning",
             "holt_w_tuning"),
  AvgProf = rep(NA, 3)
)
```

Oracle

```r
# Oracle
res_table$AvgProf[1] <- (price - cost) * mean(df$demand[301:500])
```

1. Estimate initial values

```
res_regression <- lm(data = filter(df, t <= 300),
                     formula = demand ~ t)
b <- as.numeric(res_regression$coefficients[1])
a <- as.numeric(res_regression$coefficients[2])
```

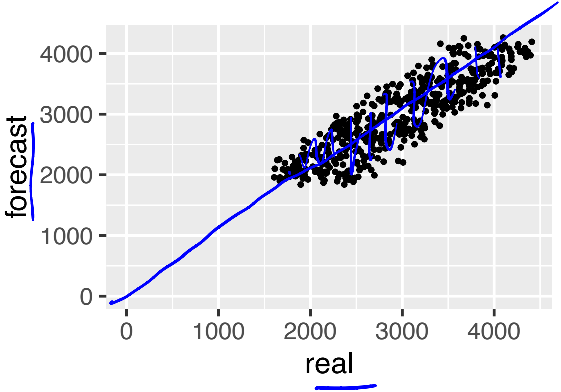
T_0

L_0

2 Forecast by Holt's method

```
L_0 <- b
T_0 <- a
alpha <- 0.2
beta <- 0.2

# use function make code clean, and easy to debug
forecast_holt_trend <- function(alpha, beta, starting_forecast, L_0, T_0, demand,
                                forecast_start_day, forecast_end_day) {
  forecasted <- c()
  L_t <- c()
  T_t <- c()
  for (i in 1:(forecast_end_day-forecast_start_day)) {
    if (i == 1){
      # ex-ante
      forecasted[1] <- starting_forecast
      # ex-post (after observing the demand)
      L_t[1] <- alpha * demand[forecast_start_day+i] + (1 - alpha) * forecasted[1]
      T_t[1] <- beta * (L_t[1] - L_0) + (1 - beta) * T_0
    } else {
      # ex-ante: update the forecast with alpha and beta
      forecasted[i] <- L_t[i-1] + T_t[i-1]
      # ex-post: update the L_t and T_t
      L_t[i] <- alpha * demand[forecast_start_day+i] + (1 - alpha) * forecasted[i]#L_t[i-1]
      T_t[i] <- beta * (L_t[i] - L_t[i-1]) + (1 - beta) * T_t[i-1]
    }
  }

  output <- list(forecasted = forecasted,
                 L_t = L_t,
                 T_t = T_t)

  return(output)
}
```

```r
forecast_1_500 <- forecast_holt_trend(alpha = alpha,
                                      beta = beta,
                                      starting_forecast = b + 1 * a,
                                      L_0 = L_0,
                                      T_0 = T_0,
                                      demand = df$demand,
                                      forecast_start_day = 0,
                                      forecast_end_day = 500)
tibble(real = df$demand, forecast = forecast_1_500$forecasted) %>%
  ggplot(aes(x = real, y = forecast)) +
  geom_point(size = 0.5) +
  xlim(c(0, NA)) +
  ylim(c(0, NA))
```



Check: Example 7.3

- Observed demand (in thousands) has been $D_1 = 8,415$, $D_2 = 8,732$, $D_3 = 9,014$, $D_4 = 9,808$, $D_5 = 10,413$, and $D_6 = 11,961$.

F_7

- Forecast demand for Period 7 using trend-corrected exponential smoothing with $\alpha = 0.1, \beta = 0.2$.
- The first step is to obtain initial estimates of level and trend using linear regression.
  - For the MP3 player data, we obtain

$$L_0 = 7,367 \text{ and } T_0 = 673$$

- The forecast for Period 1 is thus given by

$$F_1 = L_0 + T_0 = 7,367 + 673 = 8,040$$

- The observed demand for Period 1 is $D_1 = 8,415$.
- So, the error for Period 1 is thus given by

- With $\alpha = 0.1, \beta = 0.2$, the revised estimate of level and trend for Period 1:

$$L_1 = \alpha D_1 + (1-\alpha)(L_0 + T_0)$$
$$= (0.1 \times 8,415) + (0.9 \times 8,040) = 8,078$$
$$T_1 = \beta(L_1 - L_0) + (1-\beta)$$
$$T_1 = [0.2 \times (8,078 - 7,367)] + (0.8 \times 673) = 681$$

- We thus obtain the following forecast for Period 2:

$$F_2 = L_1 + T_1 = 8,078 + 681 = 8,759$$

- Continuing in this manner, we obtain
  - $L_2 = 8,755$, $T_2 = 680$, $L_3 = 9,393$, $T_3 = 672$, $L_4 = 10,039$, $T_4 = 666$, $L_5 = 10,676$, $T_5 = 661$, $L_6 = 11,399$, $T_6 = 673$.

- This gives us a forecast for Period 7 of

$$F_7 = L_6 + T_6 = 11,399 + 673 = 12,072$$

Debug if any: Check the Match!

```
demand_ex <- c(8415, 8732, 9014, 9808, 10413, 11961)
alpha_ex <- 0.1
beta_ex <- 0.2

L_0_ex <- 7367; T_0_ex <- 673
F1 <- L_0_ex + T_0_ex

forecast_ex <- forecast_holt_trend(alpha = alpha_ex,
                                   beta = beta_ex,
                                   # corresponds to F1 above
                                   starting_forecast = L_0_ex + 1 * T_0_ex,
                                   L_0 = L_0_ex,
                                   T_0 = T_0_ex,
                                   demand = demand_ex,
                                   forecast_start_day = 0,
                                   forecast_end_day = 7)
tibble(t = 1:7, forecast = forecast_ex$forecasted)
```

```
## # A tibble: 7 x 2
##       t forecast
##   <int>    <dbl>
## 1     1     8040
## 2     2     8758
## 3     3     9435.
## 4     4    10065.
## 5     5    10706.
## 6     6    11337.
## 7     7    12072.
```

No bugs, you are good to go.

3. Optimal Stocking at week 301

```
forecast_error_holt <- df$demand - forecast_1_500$forecasted
safety_stock <- quantile(forecast_error_holt[1:300], 0.80)
stocking <- forecast_1_500$forecasted[300] + safety_stock
```

SL $\quad \frac{r-c}{r-s}$

4. Ave.Profit in Holt's method?

```
forecast_error_holt <- df$demand - forecast_1_500$forecasted
safety_stock <- quantile(forecast_error_holt[1:300], 0.80)
stocking <- forecast_1_500$forecasted[301:500] + safety_stock

# Ave. profit
res_table[2, "AvgProf"] <-
  mean(price * pmin(stocking, df$demand[301:500]) +
         salvage * pmax(stocking - df$demand[301:500], 0) -
         cost * stocking)
res_table
```

General Formula

```
##           method  AvgProf
## 1         Oracle 11620.86
## 2 holt_wo_tuning 11283.66
## 3  holt_w_tuning       NA
```

5. optimization

```
# Let's find optimal alpha and beta
mse_holt_2 <- function(par, starting_forecast, L_0, T_0, demand,
                       forecast_start_day, forecast_end_day) {
  alpha <- par[1]
  beta <- par[2]

  forecasted <- forecast_holt_trend(alpha = alpha,
                                    beta = beta,
                                    starting_forecast = starting_forecast,#b + 100 * a,
                                    L_0 = L_0,#b + 100*a,
                                    T_0 = T_0,#a,
                                    demand = demand,
                                    forecast_start_day = forecast_start_day,
                                    forecast_end_day = forecast_end_day)[["forecasted"]]

  mse <- sqrt(sum((demand[(forecast_start_day+1):(forecast_end_day)] - forecasted)^2) / length(forecasted)
  return(mse)
}
```

```
res <- optim(par = c(0.2, 0.2), mse_holt_2, method = "L-BFGS-B",
             lower = c(0, 0),
             upper = c(1, 1),
             starting_forecast = b + 1 * a,
             L_0 = b + 1*a,
             T_0 = a,
             demand = df$demand,
             forecast_start_day = 0,
             forecast_end_day = 300)
res
```

```
## $par
## [1] 0 0
##
## $value
## [1] 294.4498
##
## $counts
## function gradient
##        2        2
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: NORM OF PROJECTED GRADIENT <= PGTOL"
```

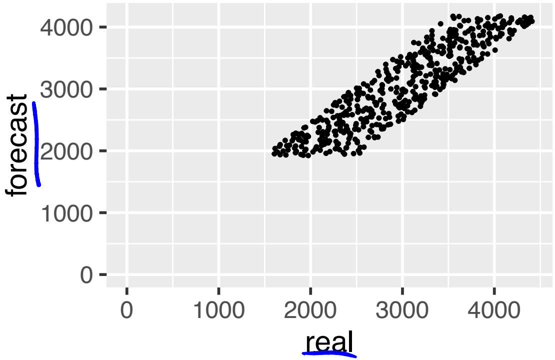
```
alpha <- res$par[1]
beta <- res$par[2]
```

Alternative: Grid-Search

```r
# You can also do the grid-search
tb_mse <- expand_grid(alpha = seq(from = 0, to = 1, length.out = 100),
                      beta = seq(from = 0, to = 1, length.out = 100)) %>%
  rowwise() %>%
  mutate(mse = mse_holt_2(par = c(alpha, beta),
                          starting_forecast = b + 1 * a,
                          L_0 = b + 1*a,
                          T_0 = a,
                          demand = df$demand,
                          forecast_start_day = 0,
                          forecast_end_day = 300))
grid_1 <- unique(tb_mse$alpha)
grid_2 <- unique(tb_mse$beta)
obj_spread <- matrix(0, nrow = length(grid_1), ncol = length(grid_2))
for (i in 1:length(grid_1)) {
  for (j in 1:length(grid_2)) {
    temp <- tb_mse %>%
      filter(alpha == grid_1[i] & beta == grid_2[j])
    obj_spread[i,j] <- temp[["mse"]]
    }
  }
saveRDS(obj_spread, file = "../code/grid_search.rds")

obj_spread <- readRDS("../code/grid_search.rds")
library(plotly)
library(htmlwidgets)
p <- plot_ly(z = obj_spread, type = "surface")
htmlwidgets::saveWidget(p, "../code/obj_spread.html", selfcontained = F, libdir = "lib")
```

6. Use the tuned parameters to calculate Avg Profit

```
forecast_1_500 <- forecast_holt_trend(alpha = alpha,
                                      beta = beta,
                                      starting_forecast = b + 1 * a,
                                      L_0 = L_0,
                                      T_0 = T_0,
                                      demand = df$demand,
                                      forecast_start_day = 0,
                                      forecast_end_day = 500)
tibble(real = df$demand, forecast = forecast_1_500$forecasted) %>%
  ggplot(aes(x = real, y = forecast)) +
  geom_point(size = 0.3) +
  xlim(c(0, NA)) +
  ylim(c(0, NA))
```



```r
forecast_error_holt <- df$demand - forecast_1_500$forecasted
safety_stock <- quantile(forecast_error_holt[1:300], 0.80)
stocking <- forecast_1_500$forecasted[301:500] + safety_stock

# Ave. profit
res_table[3, "AvgProf"] <-
  mean(price * pmin(stocking, df$demand[301:500]) +
         salvage * pmax(stocking - df$demand[301:500], 0) -
         cost * stocking)
res_table
```

```
##           method  AvgProf
## 1         Oracle 11620.86
## 2 holt_wo_tuning 11283.66
## 3  holt_w_tuning 11288.88
```

Some common Mistakes

- Mistake in forecasting
  - Inconsistent initial values, setup, parameters.
- Mistake in MSE
  - Using ES's MSE, which is wrong.
  - Some bugs inside MSE function.
- Mistake in optimal stocking
  - You cannot use the future value to decide optimal stocking.
- Mistake in optimization
  - Not optimizing two parameters.
  - Not feeding appropriate local objects.

I highly recommend that you review HWs as this might share the large portion in final exam too.

Comment

- In grading, we allow you to have different values if that is not tremendously different.
- Those “small” differences might come from the following difference.
  - Adaptive or Fixing error distribution?
  - Slight difference in tuned parameters.
  - The way to initialize the Holt’s method.
- Debug!
  - Check if you get the “exact” same result when you apply your function to some small scale setup.