STAT 436 / 536 - Lecture 11: Key

Forecasting and Prediction

- Model selection or model choice focuses on decisions about which covariates and/or model structure should be used.

- **Q**: What are common model selection procedures that you have implemented?

- For this class we we will focus on a prospective, predictive framework where the goal is to predict the next k observations given some observed data.

Decision Theory: Risk and Loss Functions

- A loss function defines a structure for evaluating predictions where a penalty is assigned to a prediction based on the distance from the true value.

- Formally, a loss function is defined as a penalty for a prediction, or decision, a and a true value θ.

- Many predictive modeling scenarios, such as those on Kaggle, have a defined loss function.

- For continuous predictions, two common loss functions are absolute error loss, $L(\theta, a) = |a - \theta|$, and squared error loss $L(\theta, a) = (a - \theta)^2$.

- More exotic loss functions can be devised that account for uncertainty in the prediction or penalize over or under estimates in a more severe manner.

- In a predictive setting, the loss is defined for a single prediction, typically $a = \hat{y}_{t+1}$ and $\theta = y_{t+1}$. The loss function could be used to evaluate a single prediction from two different models, but we still need a more formal way to compare predictions, in general, for two different models.

- A model, or an estimator, can be evaluated using a risk function. A risk function is defined as $R\left(\theta, \delta(y)\right) = E_\theta L(\theta, \delta(y))$, where $\delta(y)$ is an model, or estimator, of θ.

- The interpretation is that the risk function is the average loss incurred by model $\delta(y)$.

- A common use of the a risk function is known as the mean squared error (MSE) loss. MSE is defined as $MSE(\theta) = E_\theta(\delta(y_1, \ldots, y_n) - \theta)^2$.

- Furthermore, MSE can be decomposed as $MSE(\theta) = Var(\delta(y)) + Bias(\delta(y))^2$.

- In practice, risk is evaluated using a hold-out sample or a cross-validation setting.

Holdout Samples and Cross-Validation

- In many predictive modeling scenarios, test and training sets are constructed. Generally, 70 percent of the data is used for training (learning the model) and the evaluated on the remaining thirty percent (the test set).

- **Q:** why does a test set need to be constructed rather than just testing the predictive ability on the training set?

- Cross validation divides the data into chunks, often called folds, and then one fold is set aside for testing and the others are used to train the model. In an iterative fashion each fold, and each data point inside the fold, are used for testing once?

- **Q:** would cross-validation, test-training set, or some other approach be more more appropriate for testing in a time series framework?

- One strategy would be to use data from time 1 to t to make a prediction for $t+$ and then use data up to time $t+1$ to make a prediction for time $t+2\ldots$ This idea is similar to that of the leave-one-out cross validation approach. The key is that it still maintains the time series structure in the data.

- Sketch out pseudocode to implement this procedure.

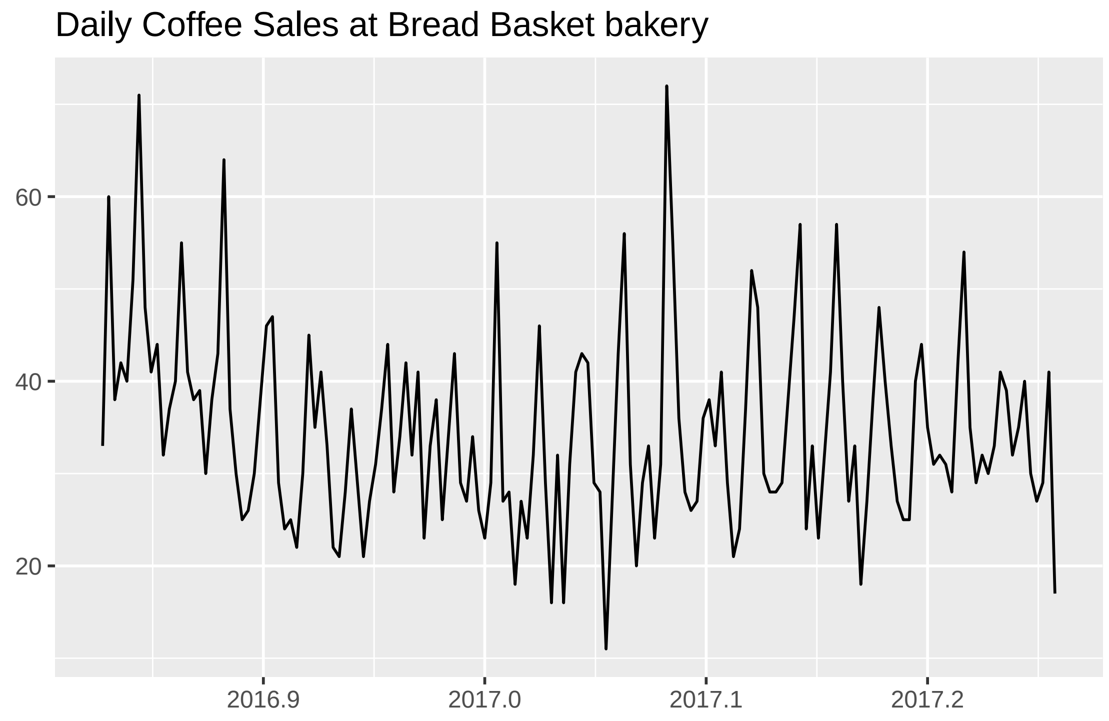
```
## Initialize Step
# 1. decide on loss function
# 2. determine t, the first point to make predictions
# 3. create objects to store / evaluate predictions across models to compare

## Loop
# 4. Loop to fit models on 1:k
# 5. Store predictions for k+1

## Evaluation
# 6. Compare predictions from t+1 to T
```

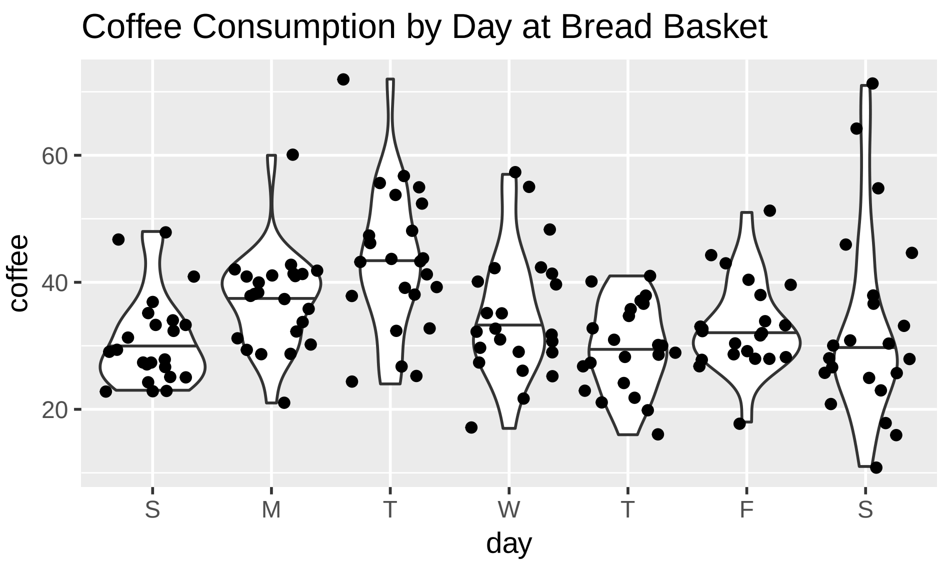
- Consider a dataset on bakery sales from a bakery in Edinburgh.

```r
bakery_sales <- read_csv('http://math.montana.edu/ahoegh/teaching/timeseries/data/BreadBasket.csv')
bakery_sales %>% filter(Item == 'Coffee') %>% group_by(Date) %>% tally() %>% select(n) %>%
  ts(frequency = 365, start=(c(2016,303))) %>% autoplot() +
  ggtitle('Daily Coffee Sales at Bread Basket bakery')
```



- **Q**: Do you anticipate seasonality in coffee sales? If so, on what scale and set up a model to evaluate this.

```r
weekly.ts <- bakery_sales %>% filter(Item == 'Coffee') %>% group_by(Date) %>% tally() %>%
  select(n) %>% ts(frequency = 7, start=(c(1,1)))
coffee.df <- data.frame(coffee = as.numeric(weekly.ts), day = as.factor(cycle(weekly.ts)))
ggplot(data=coffee.df, aes(y=coffee, x=day)) + geom_violin(draw_quantiles=.5) +
  scale_x_discrete(labels = c('S','M','T', 'W','T','F','S')) + geom_jitter() +
  ggtitle('Coffee Consumption by Day at Bread Basket')
```



- What would be a reasonable loss function in this situation? Lets just focus on the point estimate for now.

```r
abs.loss <- function(pred, actual){
  ##################################################################
  # DESCRIPTION: function to return absolute difference
  # ARGS:      - pred
  #            - actual
  # RETURNS:  - abs.difference
  ##################################################################
  return(abs(pred - actual))
}
abs.loss(5,3)
```

```
## [1] 2
```

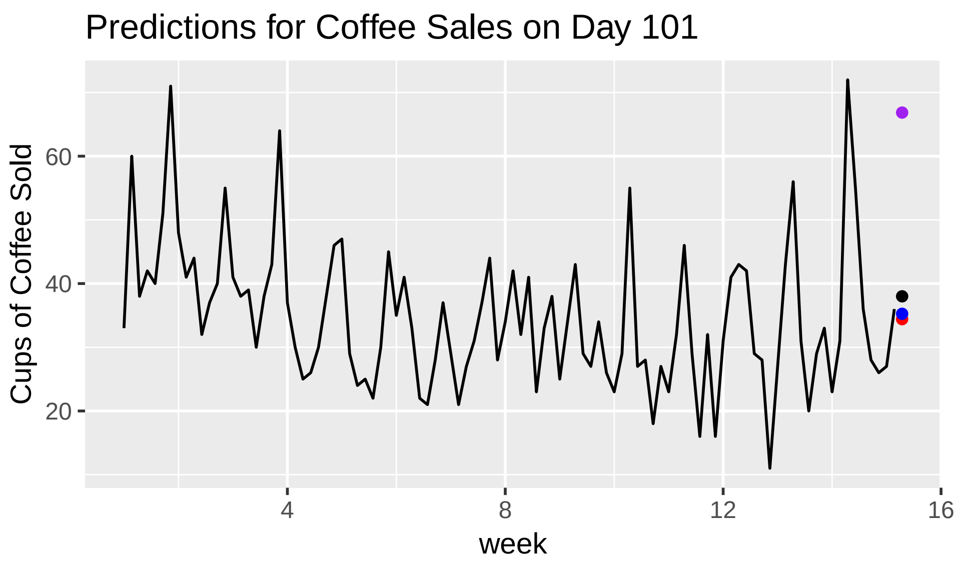


Figure 1: Black is the observed point, purple is 3-parameter HW, blue is 2-parameter HW, red is 1-parameter HW

- There are 158 days in the time series dataset. Lets make our first prediction on the 101st time point.

```
num.days <- nrow(coffee.df)
coffee.df <- coffee.df %>% mutate(day.numb = 1:num.days)
tmp <- coffee.df %>% filter(day.numb < 101) %>% select(coffee) %>% ts(frequency = 7)
hw.pred1 <- HoltWinters(tmp, gamma=FALSE, beta=FALSE)
hw.pred2 <- HoltWinters(tmp, gamma=FALSE)
hw.pred3 <- HoltWinters(tmp)
autoplot(tmp) + annotate('point',x=15 +2/7, y=predict(hw.pred1) %>% as.numeric, color='red') +
  annotate('point',x=15 +2/7, y=predict(hw.pred2) %>% as.numeric, color='blue') +
  annotate('point',x=15 +2/7, y=predict(hw.pred3) %>% as.numeric, color='purple') +
  annotate('point',x=15 +2/7, y=coffee.df[101,'coffee'], color='black') +
  ggtitle('Predictions for Coffee Sales on Day 101') + xlab('week') + ylab('Cups of Coffee Sold')
```

- We can evaluate each of the predictions from the three models

```
abs1 <- abs.loss(predict(hw.pred1) %>% as.numeric,coffee.df[101,'coffee']); abs1
```

```
## [1] 3.584554
```

```
abs2 <- abs.loss(predict(hw.pred2) %>% as.numeric,coffee.df[101,'coffee']); abs2
```

```
## [1] 2.745026
```

```
abs3 <- abs.loss(predict(hw.pred3) %>% as.numeric,coffee.df[101,'coffee']); abs3
```

```
## [1] 28.8451
```

- **Q**: Is this enough information to make a decision on the models?

- **Q**: Why do you think the purple prediction is so large?

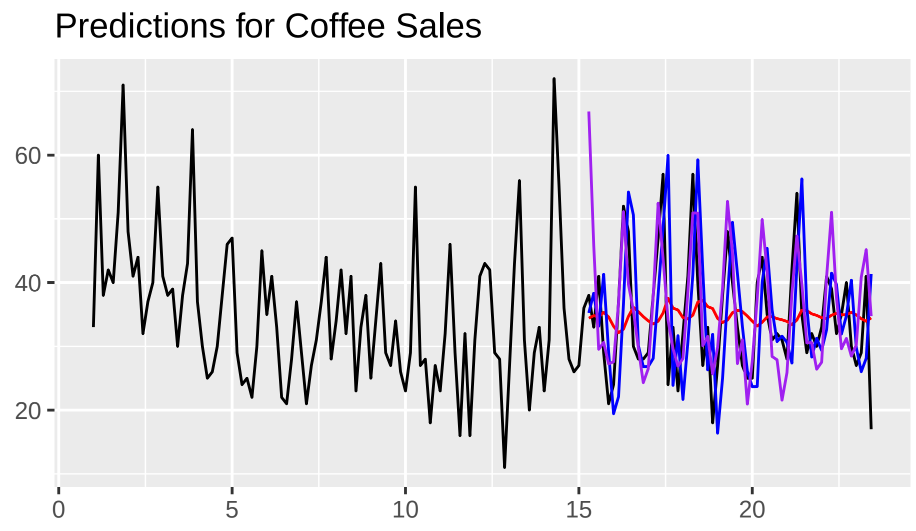
- Now modify the code to make predictions at the remaining time points. Include a visualization of the three predictions and a comparison of the absolute loss for the three models.

```
num.days <- nrow(coffee.df)
coffee.df <- coffee.df %>% mutate(day.numb = 1:num.days)
first.pred <- 101
num.pred <- num.days - first.pred + 1
preds1 <- preds2 <- preds3 <- week.time <- rep(0, num.days - first.pred + 1)

for (time.pts in 1:num.pred){
  tmp <- coffee.df %>% filter(day.numb < (time.pts + first.pred - 1)) %>% select(coffee) %>% ts(frequen
  hw.pred1 <- HoltWinters(tmp, gamma=FALSE, beta=FALSE)
  hw.pred2 <- HoltWinters(tmp, gamma=FALSE)
  hw.pred3 <- HoltWinters(tmp)
  preds1[time.pts] <- predict(hw.pred1)
  preds2[time.pts] <- predict(hw.pred2)
  preds3[time.pts] <- predict(hw.pred3)
  week.time[time.pts] <- 15 + (1 + time.pts)/7
}

df1 <- data.frame(preds=preds1, time = week.time)


coffee.df %>% select(coffee) %>% ts(frequency = 7) %>%
autoplot() + geom_line(data=data.frame(y=preds1, x=week.time),aes(x=x, y=y), color='red')  +
  geom_line(data=data.frame(y=preds2, x=week.time),aes(x=x, y=y), color='blue')  +
  geom_line(data=data.frame(y=preds3, x=week.time),aes(x=x, y=y), color='purple')  +
  ggtitle('Predictions for Coffee Sales')
```



```
mean(abs.loss(preds1, coffee.df %>% filter(day.numb > 100) %>% select(coffee) %>% pull()))
```

```
## [1] 7.550384
```

```
mean(abs.loss(preds2, coffee.df %>% filter(day.numb > 100) %>% select(coffee) %>% pull()))
```

```
## [1] 8.951779
```

```
mean(abs.loss(preds3, coffee.df %>% filter(day.numb > 100) %>% select(coffee) %>% pull()))
```

```
## [1] 5.518699
```