Module 1F: Convenience Yield in U.S. Hay and Corn

# Learning Outcomes

# Overview of this Module

In the previous module we used a simple model which predicts a saw-toothed pattern of prices over time due to a recurring pre-harvest stock-out. In real world markets we should not expect to see the the pronounced saw-toothed pattern because harvest typically does not take place at one defined point in time. In a large country with climate variation we typically see early production in the warmer regions, and then full production which gradually tapers off over a month or so. Nevertheless, we should still expect to see a rough saw-toothed pattern in prices assuming that the market stocks out.

A more important discrepancy between the theoretical intertemporal LOP and what we observe in real world markets is that real world markets for fully storable commodities such as corn or wheat never fully stock out. For example, in the U.S. corn market, stocks carried across years have never fallen below 10 percent of an average year production. There is always some carry over from one year to the next, and the level of carry over varies from year to year. Recall from Module 1E that our theory predicts that price must continue to rise as we bridge across years in those situations where stocks carried across years is positive. This means that if the theory is correct the price of commodity must rise indefinitely. This is obviously not possible.

What we tend to observe in real world markets is that years with relatively high stocks being carried over then we observe a relatively small (and sometimes zero) downward adjustment in price with the arrival of new harvest. In normal years with a more moderate amount of year over year carry over, we generally see a moderate downward adjustment in price with the arrival of new harvest. It is only when stocks are very low and only a small amount is being carried over across years do we a large reduction in price with the arrival of new harvest. It is this last case that most closely resembles the saw-toothed pricing pattern which theory predicts happens only when there is a stock out.

This lack of consistency between what the theoretial LOP predicts and what we observe in real markets is an important problem. Economists have “solved” this problem by developing the concept of *convenience yield*. You can think of convenience yield as an implicit negative cost of storage. A merchant who is storing the commodity receives a flow of convenience from having the stocks on hand rather than having to source the commodity in the spot market if an unexpected sales situation arises. The fewer the stocks which are available in the market, the higher the level of convenience from having personal stocks on hand. If the convenience yield is larger than the actual cost of storage then the combined *carrying cost*, which is the sum of the actual cost of storage and the convenience yield, will take on a negative value. With a negative carrying cost the LOP theory predicts that price must decrease from one period to the next.

We cannot observe or estimate convenience yield. For this reason we are able to treat it as a *fudge factor*. Specifically, if observe fall prices and at the same time a positive carry over of stocks from one year to the next, we infer that convenience yield must be larger than the actual cost of storage to ensure a negative carrying charge. The large price drops with the arrival of harvest in years of low carry over are attributed to a relatively high convenience yield and thus a large negative carrying cost. The small price drops in years of high carrying cost are attributed to a smaller convenience yield and thus only a small negative carrying cost.

The purpose of this module is to incorporate convenience yield into our theory of the intertemporal LOP. We then use the enhanced theory to construct a pricing simulation model. The simulation model is calibrated to the U.S. corn market. This market was chosen because the next several modules will utilize the U.S corn market as an on-going case study as we model commodity futures and the relationship between futures prices and spot prices. When studying futures markets in these subsequent modules the concept of convenience yield plays a very important role.

However, before we incorporate convenience yield into our LOP model and build the simulation model we should spend some time examining the saw toothed pricing pattern in real world commodity prices. We need a commodity which has monthly prices and observable stock levels over a large number of years. Observing both price and stocks is important because this will allow us to examine how the saw-toothed pricing pattern is different for normal stock years versus low stock years. The commodity chosen for this exercise is the U.S. hay market. The next section provides a brief introduction to the U.S. hay market and then examines the data.

# Saw-Toothed Pricing in the U.S. Hay Market

## Industry background

## Data

Production, stocks and price data for hay were extracted from the USDA Feed Grains [Database](https://www.ers.usda.gov/data-products/feed-grains-database/). Price data is monthly, production data is yearly and stocks data is bi-yearly (May and December). In this data set hay consists of alfalfa and all other types of hay. The data begins in May of 1950 and ends in July of 2021. With this large number of years it is important to account for inflation. One option is to do nothing, in which case the inflationary trend will end up in the intercept of the model when specified in first differences. The second option is to remove inflation by dividing each monthly hay price by the monthly U.S. consumer price index (CPI). For this analysis where we will use dummies to isolate seasonality, the latter approach is best. This is because the seasonal effects will increase over time due to general inflation and a straight first difference specification is not sufficient to adjust the seasonal effects for inflation.

The monthly CPI used for deflating hay prices was downloaded from the U.S. Federal Reserve ([FRED](https://fred.stlouisfed.org/series/CPIAUCSL)) website. The base year for this CPI data is 1982. This means that the hay price of 1982 is the same as in the original data. CPI adjusted hay prices for the years before (after) 1982 will be higher (lower) than the original prices, To give you a sense of the size of adjustment which was made, the hay price is $22 per ton in the original data and $92.55 per ton in the CPI adjusted data. As well, the hay price is $185 per ton in the original data and $66.06 per ton in the CPI adjusted data.

After performing the usual R preliminaries we can read the data into R in the usual way.

rm(list = ls())   
graphics.off()  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
  
hay\_data <- read.csv(file = "./Data/usda\_hay.csv", header=TRUE, sep=",", stringsAsFactors = FALSE)   
head(hay\_data)

## year month price stckMay stckDec production  
## 1 1950 May 92.55364 14599 67676 103820  
## 2 1950 Jun 87.10218 14599 67676 103820  
## 3 1950 Jul 82.26007 14599 67676 103820  
## 4 1950 Aug 83.47107 14599 67676 103820  
## 5 1950 Sep 83.40181 14599 67676 103820  
## 6 1950 Oct 84.08163 14599 67676 103820

Notice that the annual production data repeats for each of the 12 months. As well, there is a column for May stocks and another column for December stocks, both of which repeat for each of the 12 months.

There is too much data to plot monthly prices and so a data frame will be constructed which has annual prices, annual production and annual values for the May and December stocks. The next chunk of code deletes the *month* column and then creates the annual averages of the remaining columns.

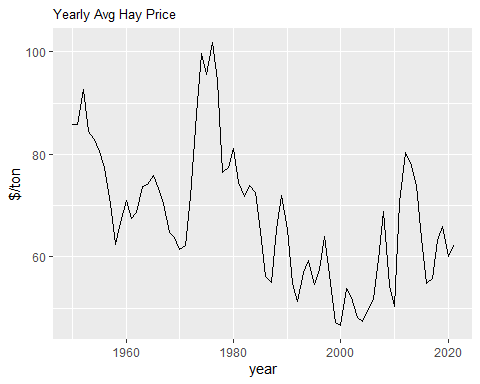
hay\_data2 <-select(hay\_data, -"month")  
  
mean\_data <-hay\_data2 %>%  
 group\_by(year) %>%   
 summarise\_each(funs(mean))

## Warning: `summarise\_each\_()` was deprecated in dplyr 0.7.0.  
## Please use `across()` instead.

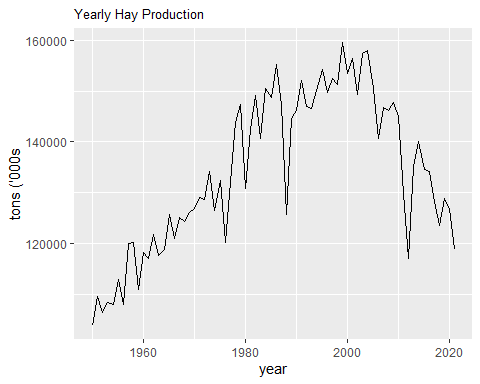
## Warning: `funs()` was deprecated in dplyr 0.8.0.  
## Please use a list of either functions or lambdas:   
##   
## # Simple named list:   
## list(mean = mean, median = median)  
##   
## # Auto named with `tibble::lst()`:   
## tibble::lst(mean, median)  
##   
## # Using lambdas  
## list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))

We can now plot average annual price, annual production, annual May stocks and Annual December stocks.

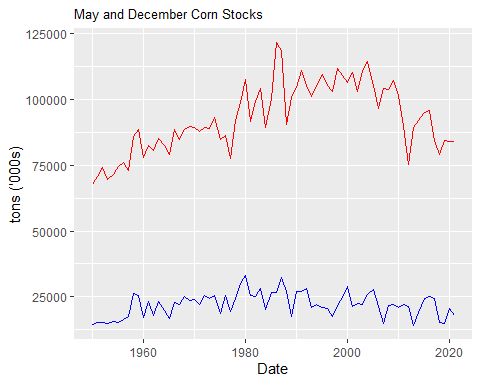
plot\_price <- ggplot(mean\_data, aes(x = year, y = price)) +   
 geom\_line() +   
 labs(title = "Yearly Avg Hay Price", y= "$/ton") +   
 theme(plot.title = element\_text(size=10))  
plot\_price



plot\_production <- ggplot(mean\_data, aes(x = year, y = production)) +   
 geom\_line() +   
 labs(title = "Yearly Hay Production", y= "tons ('000s") +   
 theme(plot.title = element\_text(size=10))  
plot\_production



plot\_stocks <- ggplot(mean\_data, aes(x = year)) +   
 geom\_line(aes(y = stckMay), color = "blue") +   
 geom\_line(aes(y = stckDec), color = "red") +   
 labs(title = "May and December Corn Stocks", y = "tons ('000s)", x = "Date") +  
 theme(plot.title = element\_text(size=10))   
plot\_stocks



The first graph show the average annual inflated-adjusted (real) price of hay beginning in 1950. The longer term pricing pattern is similar to the major crops in that prices rose steadily to duing the inflationary period of the 1970s, and then began a long term decline throughout the 1980s and 1990s. For the past two decades prices have continually strengthened.

The second graph shows annual production. This graph is rather surprising because it shows steady growth up until about the year 2000 and then a fairly sharp downturn in production for the last 20 years. More research would be required to understand the reason for this abrupt change. If it was a change in the USDA definition of hay we would expect to see a one time drop rather than a gradual decline. The large reduction in production in the late 1980s is a weather shock (i.e., drought) rather than a response to a price change. The same is true for the sharp reduction in production in 2012. These extreme droughts can be verified on the associated [Wikipedia](https://en.wikipedia.org/wiki/Droughts_in_the_United_States) page.

The third graph shows the annual May and December hay stocks. The top schedule shows December stocks, which corresponds to the beginning of the cattle feeding season. The bottom schedule shows May stocks, which corresponds to the end of the cattle feeding season. The rising December stocks up until the year 2000 reflects the rising production levels up until the year 2000. Conversely, the falling December stocks reflect the falling production levels after 2000.

Hay is storable but the cost of storing it for more than one season is relatively high because it must be covered to prevent excessive deterioration from rain. It is for this reason that the May stocks do not have noticable up and down trends. Even though there is not a large amount of variation in May stocks there is enough to get a sense of the relative scarcity of hay during the previous spring and winter season. For this reason it is useful to examine the summary statistics for May hay stocks.

summary(mean\_data$stckMay)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 14156 18384 22081 22071 25358 33192

We see that the average level of stocks is about 22 million tons (keep in mind that the units of measure for the Q variables are thousands of tons). The production chart shows that in recent years production was about 120 million tonnes. This means that about 18 percent of annual production is carried over. This carry over fraction is similar to those of the major grains in the U.S.. The table shows that the first quartile is about 18 million tons. This means that 25 percent of the observations were below 18 million. In the analysis below 18 million is used as the threshold for defining a “low stock” year. Even though 18 million is not much below the average of 22 million, we will see that there is a large difference in the seasonal pattern of prices for a normal year versus a low stock year.

## Dummy Variables with Interaction Terms

In our study of the U.S. potato CPI in Module 1E we used a set of 11 monthly dummies (December was omitted and serves as the reference point) for measuring seasonality in potato prices. We will do the same thing here except we will use conditional monthly dummies in additional to the regular unconditional dummies. Specifically, we will create an indicator variable which takes on a value of one if May stocks are low (i.e., less than the first quartile, which is approximately 18 million tons) and a value of zero otherwise, We will then create a new set of 11 dummies, where each dummy is the product of the regular monthly dummy and the indicator variable. This will become clearer with a formal specification of the enhanced dummy variable model.

The enhanced dummy variable model can be expressed as

In a normal year and the model reduces to the standard dummy variable model:

It is important to understand that even though this equation is estimated in first difference format, we can continue to interpret as the long run average difference between the December price and the price in month . Let’s use the much simplier quarterly dummy variable model to ensure that this is true.

The quarterly dummy variable model can be expressed as

When switching from Q4 to Q1 we know takes on a value of 1 and the other two dummies take on a value of 0. Thus, is our estimate of the average price difference between Q1 and Q4. Formally, where the “bar” denotes a long term average.

When switching from Q1 to Q2 we know takes on a value of -1, takes on a value of 1 and takes on a value of 0. Thus, the average price difference between Q1 and Q2 is given by . Formally, we have . Now substitute into this equation and solve for to get . The same logic can be used to show that . The proof is similar for our current model with monthly prices.

When we are in a low stock year then and the dummy model becomes

This equation can be written more compactly as

This tells us that is a measure of the difference in price for month in a low stock year and December in a normal year. We can therefore conclude that is the how the month is seasonal effect is impacted by the low stock year. If is positive (negative) then the month price is higher (lower) than in a normal year due to the low stocks. One important case is when but . In this situation the low stocks has caused the month price to be lower rather than higher than the December price.

## Building the Data Set

Our first task is to create the indicator variable. We need to first identify the first quartile of the May stock variable and then create a column within which the indicator takes on a value of one if the actual value for May stocks is below the first quartile and zero otherwise. Keep in mind that we are working with annual data at this point.

(quant\_cut <- unname(quantile(mean\_data$stckMay, 0.25)))

## [1] 18384

mean\_data <- mean\_data %>%   
 mutate(stckDum = ifelse(stckMay<quant\_cut,1,0) )

We now need to create a new data frame that has the top row deleted and only contains the year and month columns. We will later add the first differenced price series to this data frame.

diff\_data <- select(hay\_data, c("year","month"))  
 diff\_data <- diff\_data[-1,]

We can now use the original monthly data stored in the *hat\_data* data frame to calculate the first difference of price and add it to the previously-created *diff\_data* data frame.

diff\_data <- diff\_data %>%   
 mutate(price\_diff = diff(hay\_data$price) )  
 head(diff\_data)

## year month price\_diff  
## 2 1950 Jun -5.45146149  
## 3 1950 Jul -4.84210277  
## 4 1950 Aug 1.21099960  
## 5 1950 Sep -0.06926666  
## 6 1950 Oct 0.67982493  
## 7 1950 Nov 2.09722914

We now have a data set consisting of monthly price first differences. The next step is to convert the annual low stock indicator price series into a monthly format. For example, if 1976 is a low stock year, then in the *diff\_data* data frame each of the 12 months for the 1976 entry should take on a value of 1. The easiest way to do this is to merge the two data sets, using *year* as the common variable.

full\_data <- merge(x = diff\_data, y = mean\_data[,c("year","stckDum")], by = "year", all.x = TRUE)  
  
head(full\_data)

## year month price\_diff stckDum  
## 1 1950 Jun -5.45146149 1  
## 2 1950 Jul -4.84210277 1  
## 3 1950 Aug 1.21099960 1  
## 4 1950 Sep -0.06926666 1  
## 5 1950 Oct 0.67982493 1  
## 6 1950 Nov 2.09722914 1

The next step in the construction of the data set is to create the monthly dummy variables and add them to the *full\_data* data frame.

D <- hay\_data$month  
dummies <- model.matrix(~D+0)  
  
col\_order <- c("DJan","DFeb","DMar","DApr","DMay","DJun","DJul","DAug","DSep","DOct","DNov")  
dummies <- dummies[, col\_order]

After we difference these dummies we can add them to our *full\_data* data frame.

dummies\_diff <- diff(dummies)  
full\_data <- cbind(full\_data,dummies\_diff)  
head(full\_data, 4)

## year month price\_diff stckDum DJan DFeb DMar DApr DMay DJun DJul DAug DSep  
## 2 1950 Jun -5.45146149 1 0 0 0 0 -1 1 0 0 0  
## 3 1950 Jul -4.84210277 1 0 0 0 0 0 -1 1 0 0  
## 4 1950 Aug 1.21099960 1 0 0 0 0 0 0 -1 1 0  
## 5 1950 Sep -0.06926666 1 0 0 0 0 0 0 0 -1 1  
## DOct DNov  
## 2 0 0  
## 3 0 0  
## 4 0 0  
## 5 0 0

The final step for constructing the data set is to create interaction variables. Recall that these interaction variables are created by multipying the set of 11 differenced dummy variables by the indicator variable which indicates whether or not the current year is a low stock year. The interaction variables are created directly within the *full\_data* data frame as follows:

full\_data <- full\_data %>%   
 mutate(IJan = DJan\*stckDum,  
 IFeb = DFeb\*stckDum,  
 IMar = DMar\*stckDum,  
 IApr = DApr\*stckDum,  
 IMay = DMay\*stckDum,  
 IJun = DJun\*stckDum,  
 IJul = DJul\*stckDum,  
 IAug = DAug\*stckDum,  
 ISep = DSep\*stckDum,  
 IOct = DOct\*stckDum,  
 INov = DNov\*stckDum)  
  
head(full\_data, 4)

## year month price\_diff stckDum DJan DFeb DMar DApr DMay DJun DJul DAug DSep  
## 2 1950 Jun -5.45146149 1 0 0 0 0 -1 1 0 0 0  
## 3 1950 Jul -4.84210277 1 0 0 0 0 0 -1 1 0 0  
## 4 1950 Aug 1.21099960 1 0 0 0 0 0 0 -1 1 0  
## 5 1950 Sep -0.06926666 1 0 0 0 0 0 0 0 -1 1  
## DOct DNov IJan IFeb IMar IApr IMay IJun IJul IAug ISep IOct INov  
## 2 0 0 0 0 0 0 -1 1 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 -1 1 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 -1 1 0 0 0  
## 5 0 0 0 0 0 0 0 0 0 -1 1 0 0

## Estimated Model Without Interaction Variables

We will begin by estimating the simple version of the model, which means the one without the interaction variables. The coefficient estimates should be stored for the purpose of graphical analysis.

regP\_diff <- lm(price\_diff ~ DJan + DFeb + DMar + DApr + DMay + DJun + DJul + DAug + DSep + DOct + DNov + 0, data = full\_data)  
summary(regP\_diff)

##   
## Call:  
## lm(formula = price\_diff ~ DJan + DFeb + DMar + DApr + DMay +   
## DJun + DJul + DAug + DSep + DOct + DNov + 0, data = full\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -9.1557 -1.3362 -0.0353 1.1572 18.7633   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## DJan 0.5385 0.2880 1.870 0.06183 .   
## DFeb 1.1569 0.3883 2.980 0.00297 \*\*   
## DMar 1.1586 0.4511 2.568 0.01039 \*   
## DApr 2.4256 0.4911 4.940 9.44e-07 \*\*\*  
## DMay 4.8592 0.5135 9.463 < 2e-16 \*\*\*  
## DJun 0.2507 0.5207 0.481 0.63036   
## DJul -1.6049 0.5135 -3.126 0.00184 \*\*   
## DAug -1.4315 0.4911 -2.915 0.00365 \*\*   
## DSep -0.8339 0.4511 -1.849 0.06487 .   
## DOct -0.1768 0.3883 -0.455 0.64904   
## DNov -0.5174 0.2880 -1.797 0.07274 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.535 on 842 degrees of freedom  
## Multiple R-squared: 0.311, Adjusted R-squared: 0.302   
## F-statistic: 34.55 on 11 and 842 DF, p-value: < 2.2e-16

matrix\_coef1 <- summary(regP\_diff)$coefficients  
coeff1 <- as.data.frame(matrix\_coef1[,1])  
colnames(coeff1) <- "dum"  
coeff1

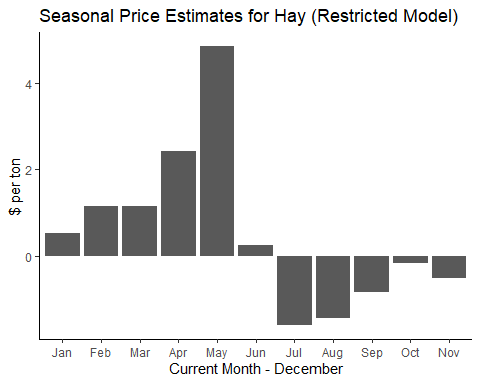
## dum  
## DJan 0.5385143  
## DFeb 1.1569003  
## DMar 1.1586057  
## DApr 2.4256484  
## DMay 4.8591714  
## DJun 0.2506579  
## DJul -1.6049427  
## DAug -1.4315036  
## DSep -0.8339005  
## DOct -0.1767678  
## DNov -0.5174031

Lets graph the estimated coefficients for the 11 dummy variables. To do this we need to first create a set of X axis labels and bind them to our set of coefficient estimates.

label <- factor(c("Jan","Feb","Mar", "Apr","May","Jun","Jul","Aug","Sep","Oct","Nov"),  
 levels = c("Jan","Feb","Mar", "Apr","May","Jun","Jul","Aug","Sep","Oct","Nov"))  
  
coeff1 <- cbind(coeff1, label)

Now we can build the column graph.

plot1 <- ggplot(coeff1, aes(x=label, y=dum)) +  
 geom\_bar(stat = "identity") +  
 theme\_classic() +  
 labs(title = "Seasonal Price Estimates for Hay (Restricted Model)",  
 x = "Current Month - December",  
 y = "$ per ton")  
  
plot1



Discuss the results …

## Estimated Model with Interaction Terms

To conclude this first section of the module we will estimate the dummy variable with the interaction terms. After estimating the model we will graph the estimated coefficients on the dummies and interaction variables separately so we can more effectively see how low stocks affects the seasonality in hay prices.

We begin by estimating the unrestricted model and saving the estimated coefficients.

regP\_diff2 <- lm(price\_diff ~ DJan + DFeb + DMar + DApr + DMay + DJun + DJul + DAug + DSep + DOct + DNov + IJan + IFeb + IMar + IApr + IMay + IJun + IJul + IAug + ISep + IOct + INov + 0, data = full\_data)  
summary(regP\_diff2)

##   
## Call:  
## lm(formula = price\_diff ~ DJan + DFeb + DMar + DApr + DMay +   
## DJun + DJul + DAug + DSep + DOct + DNov + IJan + IFeb + IMar +   
## IApr + IMay + IJun + IJul + IAug + ISep + IOct + INov + 0,   
## data = full\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -9.1695 -1.2475 -0.0492 1.1833 17.7122   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## DJan 0.2338 0.3204 0.730 0.46566   
## DFeb 0.9085 0.4320 2.103 0.03577 \*   
## DMar 0.8536 0.5019 1.701 0.08939 .   
## DApr 2.2551 0.5465 4.127 4.05e-05 \*\*\*  
## DMay 5.7397 0.5715 10.043 < 2e-16 \*\*\*  
## DJun 1.1451 0.5796 1.976 0.04852 \*   
## DJul -0.3972 0.5715 -0.695 0.48719   
## DAug -0.5134 0.5465 -0.940 0.34771   
## DSep -0.1443 0.5019 -0.287 0.77385   
## DOct 0.4735 0.4320 1.096 0.27333   
## DNov -0.3213 0.3204 -1.003 0.31626   
## IJan 1.2722 0.6546 1.943 0.05231 .   
## IFeb 1.0371 0.8826 1.175 0.24029   
## IMar 1.2732 1.0252 1.242 0.21462   
## IApr 0.7116 1.1158 0.638 0.52380   
## IMay -3.6787 1.1664 -3.154 0.00167 \*\*   
## IJun -3.7344 1.1824 -3.158 0.00164 \*\*   
## IJul -5.0429 1.1664 -4.323 1.72e-05 \*\*\*  
## IAug -3.8335 1.1158 -3.436 0.00062 \*\*\*  
## ISep -2.8796 1.0252 -2.809 0.00509 \*\*   
## IOct -2.7156 0.8826 -3.077 0.00216 \*\*   
## INov -0.8189 0.6546 -1.251 0.21131   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.459 on 831 degrees of freedom  
## Multiple R-squared: 0.3599, Adjusted R-squared: 0.343   
## F-statistic: 21.24 on 22 and 831 DF, p-value: < 2.2e-16

matrix\_coef2 <- summary(regP\_diff2)$coefficients  
coeff2 <- as.data.frame(matrix\_coef2[,1])  
coeff2

## matrix\_coef2[, 1]  
## DJan 0.2338463  
## DFeb 0.9084823  
## DMar 0.8536051  
## DApr 2.2550683  
## DMay 5.7397381  
## DJun 1.1451147  
## DJul -0.3972435  
## DAug -0.5134314  
## DSep -0.1442806  
## DOct 0.4735457  
## DNov -0.3212744  
## IJan 1.2722316  
## IFeb 1.0371002  
## IMar 1.2732106  
## IApr 0.7116016  
## IMay -3.6786872  
## IJun -3.7344412  
## IJul -5.0428943  
## IAug -3.8334804  
## ISep -2.8795615  
## IOct -2.7156044  
## INov -0.8189205

We would like the estimated coefficients for the 11 interaction variables in separate column rather than being part of the larger 22 coefficient vector. We can filter out the estimated dummy coefficients and the interaction variable coefficients as follows:

coeff2\_A <- coeff2 %>% slice(1:11)  
coeff2\_A <- coeff2\_A   
colnames(coeff2\_A) <- "dum"  
  
coeff2\_B <- coeff2 %>% slice(12:22)  
colnames(coeff2\_B) <- "inter"

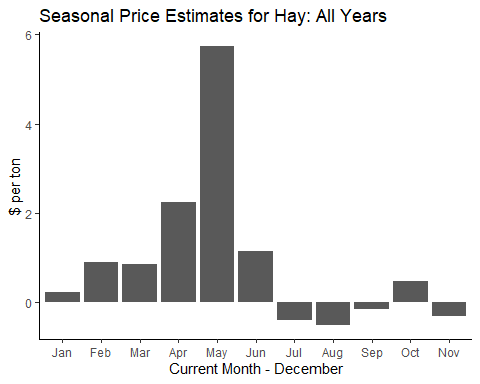
Now we can bind the two sets of estimates into a new data frame. Within this new data frame we will create a new column called *dum\_plus\_inter* which is the sum of the dummy variable and the interaction term (i.e., )

coeff2 <- cbind(coeff2\_A,coeff2\_B)  
  
coeff2 <- coeff2 %>%   
 mutate(dum\_plus\_inter = dum + inter )   
coeff2

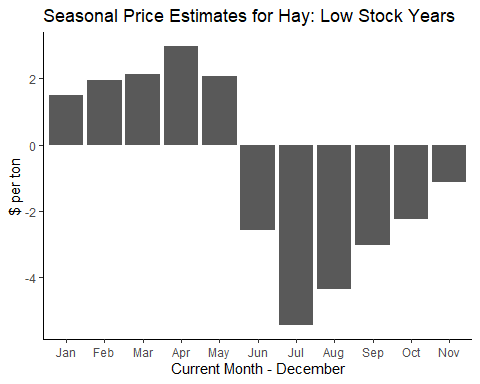
## dum inter dum\_plus\_inter  
## DJan 0.2338463 1.2722316 1.506078  
## DFeb 0.9084823 1.0371002 1.945582  
## DMar 0.8536051 1.2732106 2.126816  
## DApr 2.2550683 0.7116016 2.966670  
## DMay 5.7397381 -3.6786872 2.061051  
## DJun 1.1451147 -3.7344412 -2.589327  
## DJul -0.3972435 -5.0428943 -5.440138  
## DAug -0.5134314 -3.8334804 -4.346912  
## DSep -0.1442806 -2.8795615 -3.023842  
## DOct 0.4735457 -2.7156044 -2.242059  
## DNov -0.3212744 -0.8189205 -1.140195

We can now create the pair of graphs for the unrestricted model. The first graph will show the estimated coefficients for the dummy variables, which reflects seasonality in a normal year. The second graph will show the sum of the estimated coefficients for the dummy variables and the interaction variables. This latter graph reflects seasonality in a low stock year.

plot2A <- ggplot(coeff2, aes(x=label, y=dum)) +  
 geom\_bar(stat = "identity") +  
 theme\_classic() +  
 labs(title = "Seasonal Price Estimates for Hay: All Years",  
 x = "Current Month - December",  
 y = "$ per ton")  
  
plot2A



plot2B <- ggplot(coeff2, aes(x=label, y=dum\_plus\_inter)) +  
 geom\_bar(stat = "identity") +  
 theme\_classic() +  
 labs(title = "Seasonal Price Estimates for Hay: Low Stock Years",  
 x = "Current Month - December",  
 y = "$ per ton")  
  
plot2B



## Convenience Yield

Imagine in the days prior to debit and credit cards how you would decide how much money you would withdraw from the bank in order to pay for your purchases for the coming week. It is likely that you would take out money than what you expect to use in case you need to make an unexpected purchase. You are willing to give up the interest earnings on the extra amount that you withdraw in order to have the convenience of cash on hand when an unexpected purchase is necessary. The more inconvenient it is to obtain more cash, the greater the convenience yield you will obtain from having the extra cash on hand.

The situation is similar for merchants and processors. They will want to keep more inventory on hand than what they expect to use. The implicit benefit that merchants and processors receive from having inventory on hand rather than having to buy it in the spot market is called *convenience yield*. Convenience yield, which is equivalent to a negative cost of storage, is larger when stocks are scarce and smaller when stocks are plentiful. This makes sense because when stocks are scare it is more costly to search and find stocks in the spot market when an unexpected order comes along.

## Combining Convience Yield and Storage Costs

Our goal is to create an integrated model of storage costs and convenience yield. In the previous module the marginal physical cost of a unit of the stored commodity was assumed to be constant at level , and the capital cost of storage was assumed to equal where is the rate of interest and is the commodity’s price. The rate of interest is currently very low and so we will simplify by assuming . However, the assumption that the unit cost of storage does not depend on aggregate stocks in the market is not very realistic. We expect a higher unit cost storage with higher stocks because congestion will require high-cost storage options to be utilized.

With these additional assumptions, the marginal cost of storage in period can be expressed as where is a measure of aggregate stocks in the market. The marginal convenience yield can be expressed as . The negative slope of this function shows that marginal convenience yield is lower when stocks are larger. The net cost of storage is given by . We subtract because it is a benefit, which is equivalent to a negative cost. We call this net cost of storage the commodity’s *carrying cost*

Similar to the previous module, the intertemporal LOP tells us that when storage is positive then the price increase must equal the carrying cost. A merchant is indifferent between selling the commodity immediately and receiving price , or storing the commodity for one period and then selling it for a net price of . Indifference implies . We can rearrange to obtain a revised LOP expression:

Let and . If we substitute this pair of expressions together with and we obtain the final LOP expression:

## Pricing Dynamics The goal is to generate a pricing pattern similar to that shown in the diagram above. In the Q4 summer quarter the price is falling, which means that . We know that because storage costs are higher and convenience yield is lower when stocks increase. Thus, it must be the case that . In Q1 when stocks are largest and convenience yield is smallest, we see that takes on its largest value, and thus the price increases are the largest. As stocks diminish as we progress from Q1 to Q2 to Q3 the value of remains positive but it is getting smaller and smaller. Thus, the price increases are weakening. By the time we hit Q4 the convenience yield is larger than the storage costs, which means that and the price decreases instead of increases.

The LOP price equation is one of three key equations which governing how prices change over time. The general stock adjustment equation is where is the level of new production (i.e., harvest) in period and is the level of consumption in period . For the case of quarterly data, the full stock adjustment equation can be written as

In the above equation represents starting stocks, represents the level of stocks which are typically carried over from year to year (i.e., long-run carry over) and is a measure of the short run demand for year 2 stocks to be carried into year 3 net of . These three variables, , and , together with the two harvest variables, and , are exogenous in the model. An exogenous variable means that we must attach values to these variables from outside the model rather than calculating equilbrium values from within the model.

The final equation for simulating prices over the eight quarters is the inverse demand for the commodity by the processor. We will use the standard linear demand schedule, . The model has seven LOP equations, eight stock adjustment equations and eight demand equations for a total of 23 equations. There are also 23 variables to be solved for: eight quarterly prices (), eight quarterly consumption levels () and seven ending stocks (). There are only seven stock variables because as you can see above the value of is determined by . After assigning values to the model parameters and the exogenous variables we can solve this system of equations and then recover the equilibrium prices, which we are interested in analyzing.

## Data

We will simulate eight quarterly prices, which means two full years beginning with Q1 in the fall of year 1 and ending with Q8 in the summer of year 2. Data from the USDA Feed Grains Database reveals that average corn harvested acres, yield per harvested acre and beginning stocks for the most recent five crop years (2015/16 - 2019/20) was 82.91 million acres, 173.4 bushels per acre and 2.015 billion bushels, respectively.12 Multiplying the five year average acreage and yield gives five year average production of 14.38 billion bushels. If these estimates are used as the long term average it follows that for the base case. Similarly, if the five year average level of stocks is viewed as normal pipeline stocks it follows that for the base case.

The two parameters from the demand equation are set to and . These parameters ensure that the average price across all eight quarters is $3.628/bu, which is very close to the $3.648/bu average farm gate price for 2016 - 2020. Moreover, the demand elasticity, which is calculated with the simulated $3.628/bu average quarterly price and the simulated 3.595 billion bushel average quarterly consumption, is equal to -0.288. This simulated elasticity is reasonably close to the -0.2 corn demand elasticity estimate which was reported by Moschini et al. [2017].

Data on storage costs and convenience yields are not available and so it is not possible to directly estimate values for m0 and m1. The chosen values, and , are those which achieve a reasonably close match between the seasonal pattern of the simulated prices and real-world prices. More is said about this below.

Let’s formally assign these values to the model parameters and exogenous variables.

a <- 16.21  
b <- 3.50  
m0 <- -0.22  
m1 <- 0.03  
S0 <- 2.015  
H1 <- 14.38  
S\_bar <- 2.015  
v <- c(a, b, m0, m1, S0, H1, S\_bar)

## Simulation

We could tell R to solve the systems of 23 equations and 23 variables and then recover the eight equilibrium quarterly prices. However, to tie in better with the next module it is useful to specify the equilibrium prices as linear functions of and . These two variables are chosen because it is natural to view (i.e., year 2 harvest) as a random variable in Q1 through Q4, and to view (i.e., long term net demand for stocks) as a random variable in all eight quarters. The USDA provides a forecast for both of these variables and the properties of these forecasts will be used when using the model to generate random prices.

The desired equation is

In this equation the (tilde) on the and variables indicate that they are random as described above.

There are eight values for each of , and , which means that we need 24 values to generate the set of eight quarterly prices. The complex set of equations which are required to obtain the 24 values are contained in a R function titled “get\_delta” (this function is stored in file titled “price\_function.R”). We can call this function into our program, generate the 24 delta values and store these values in a matrix titled “del” as follows:

source("./Code/price\_function.R")  
del <- get\_delta(v)  
del

## del0 del1 del2  
## [1,] 9.545454 -0.4212615 0.4005162  
## [2,] 9.760180 -0.4248723 0.4039492  
## [3,] 9.919621 -0.4321249 0.4108446  
## [4,] 10.025144 -0.4430814 0.4212615  
## [5,] 10.077655 -0.4578358 0.4352893  
## [6,] 10.077603 -0.4465144 0.4530481  
## [7,] 10.024987 -0.4390203 0.4746902  
## [8,] 9.919357 -0.4352893 0.5004010

The next step is to use these 24 values together with to generate the eight quarterly prices. The following function does the necessary calculations:

get\_price <- function(H5,D) {  
 Price <- del[,1] + del[,2]\*H5 + del[,3]\*D   
}

If we assume and , which implies that year 2 harvest and long term net demand are both normal, then the prices which are generated will equal the base case prices. We first assign these values to and and then call the above pricing function:

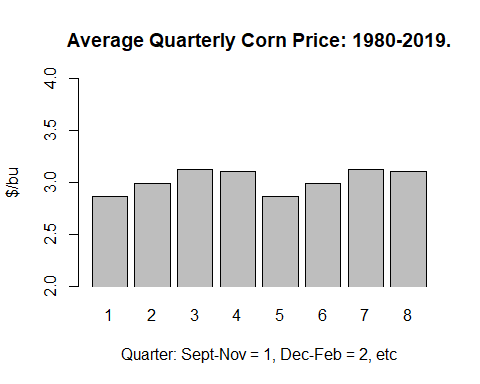
H5 <- 14.38  
D <- 0  
P <- get\_price(H5,D)  
P

## [1] 3.487714 3.650515 3.705664 3.653634 3.493977 3.656725 3.711874 3.659897

## Comparing Simulated and Real-World Corn Prices

It is of interest to compare the simulate quarterly prices of corn to real world long term average (1980 - 2019) spot prices of corn. The following long term quarterly corn prices are plotted as follows:

historic <- c(2.866, 2.993, 3.123, 3.105, 2.866, 2.993, 3.123, 3.105)  
barplot(historic,names.arg = c(1,2,3,4,5,6,7,8), main="Average Quarterly Corn Price: 1980-2019.",  
 xlab="Quarter: Sept-Nov = 1, Dec-Feb = 2, etc", ylab="$/bu",  
 beside=TRUE, ylim=c(2, 4), xpd = FALSE)

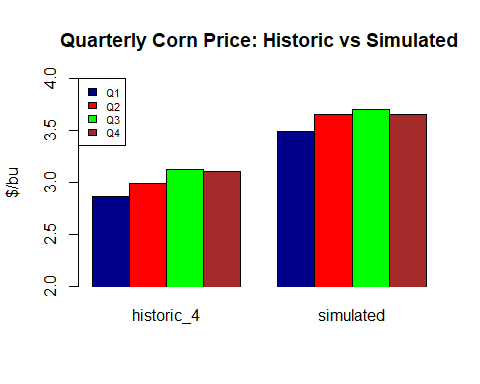


The next chart compares the simulated prices for Q1 through Q4 with the average quarterly corn prices (as shown above). The similarity of the seasonal pattern in the simulated and real-world prices suggests that despite its simplicity the calibrated model is well suited to analyzing the forward curve for corn.

historic\_4 <- historic[1:4]  
simulated <- P[1:4]  
all\_price <- cbind(historic\_4,simulated)  
rownames(all\_price) <- c("Q1","Q2","Q3","Q4")  
all\_price

## historic\_4 simulated  
## Q1 2.866 3.487714  
## Q2 2.993 3.650515  
## Q3 3.123 3.705664  
## Q4 3.105 3.653634

barplot(all\_price, main="Quarterly Corn Price: Historic vs Simulated",ylab="$/bu",  
 col=c("darkblue","red", "green", "brown"),  
 legend = rownames(all\_price), args.legend = list(x = "topleft", cex = .7), beside=TRUE, ylim=c(2, 4), xpd = FALSE)



It should be obvious from this chart that the simple model we are using does a reasonably good job capturing the seasonality in real-world corn prices. This is especially important when when we examine hedging in a future module.

## Verifying the LOP

It is a good idea to verify that the model is working as intended. First, we should check that the LOP equation, , holds for all eight quarters. Second, we should check that consumption of the inventory across all eight quarters is equal to . In other words, all stocks are fully accounted for. Both of these verifications are conducted in the Appendix of this module.

## What If Analysis

Now that we have the model built we can do “what if” analysis. An obvious “what if” is how does the set of 8 prices respond to a change in the size of the year 2 harvest (e.g., a smaller value for due to fewer planted acres)? For example, suppose , which is below the level of year 1 harvest, . Generating a new set of eight quarterly prices requires using the pricing function, “get\_price”, with the revised value for .

H5 <- 13  
D <- 0  
P\_rev <- get\_price(H5,D)  
P\_rev

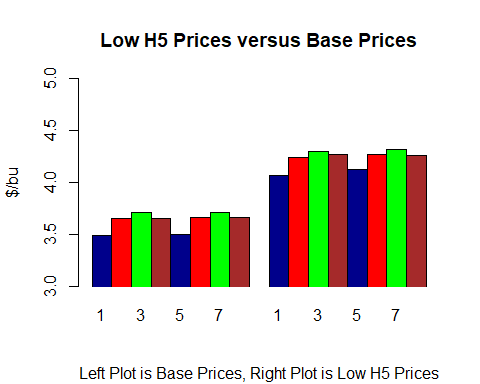
## [1] 4.069054 4.236839 4.301997 4.265086 4.125790 4.272915 4.317722 4.260596

To graph this revised set of prices together with the base set of prices we first combine the two price series into a single matrix and then generate a side-by-side bar chart similar to what what shown in the previous section.

all\_price2 <- cbind(P,P\_rev)  
all\_price2

## P P\_rev  
## [1,] 3.487714 4.069054  
## [2,] 3.650515 4.236839  
## [3,] 3.705664 4.301997  
## [4,] 3.653634 4.265086  
## [5,] 3.493977 4.125790  
## [6,] 3.656725 4.272915  
## [7,] 3.711874 4.317722  
## [8,] 3.659897 4.260596

barplot(all\_price2, main="Low H5 Prices versus Base Prices",ylab="$/bu", sub="Left Plot is Base Prices, Right Plot is Low H5 Prices", names.arg=c(1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8),  
 col=c("darkblue","red", "green", "brown"),  
 beside=TRUE, ylim=c(3, 5), xpd = FALSE)



The chart shows that the lower year 2 harvest raises all eight prices by roughly the same amount. This is an important pricing property of storable commodities. A supply or demand shock in the current year will impact prices in future years by roughly the same amount. This is because merchants are continually shifting the level of stocks being carried through time in order to maximize the average selling price of the commodity.

When prices adjust in response to a supply or demand shock such as the one implied by the previous chart the impact on the eight prices is similar but not identical. This is because the shock will typically change the level of stocks, which in turn affects storage costs and convenient yield. The pricing impacts are calculated as follows:

impacts <- P\_rev - P  
impacts

## [1] 0.5813409 0.5863238 0.5963324 0.6114524 0.6318134 0.6161899 0.6058481  
## [8] 0.6006992

## Appendix

The purpose of this Appendix is to demonstrate the LOP and the stock adjustment equation hold for the base case set of eight similated prices. The verification begins by substituting the equilibrium price into the inverse demand schedule to obtain the quarterly consumption, , Then substitute quarterly consumption into the stock adjustment equation to derive the quarterly stock levels, . Finally, substitute the derived values of and the equilibrium prices into the LOP equation, , to verify that the equation holds.

H5 <- 14.38  
D <- 0  
X <- a/b - 1/b\*P  
S <- rep(0, times = 8)  
S[1] <- S0 + H1 - X[1]  
S[2] <- S[1] - X[2]  
S[3] <- S[2] - X[3]  
S[4] <- S[3] - X[4]  
S[5] <- S[4] + H5 - X[5]  
S[6] <- S[5] - X[6]  
S[7] <- S[6] - X[7]  
S[8] <- S[7] - X[8]  
S

## [1] 12.760061 9.171637 5.598970 2.011436 12.758287 9.171637 5.600744  
## [8] 2.015000

lop <- rep(0, times = 8)  
for(i in 1:7)  
{  
 lop[i+1] <- P[i+1]-P[i]-(m0+m1\*S[i])  
}  
lop

## [1] 0.000000e+00 -3.247402e-15 -1.193490e-15 3.635980e-15 -4.996004e-15  
## [6] 8.049117e-16 5.828671e-16 2.137179e-15