IS607 Project 1

Thursday, February 26, 2015

Contents

Load the data set

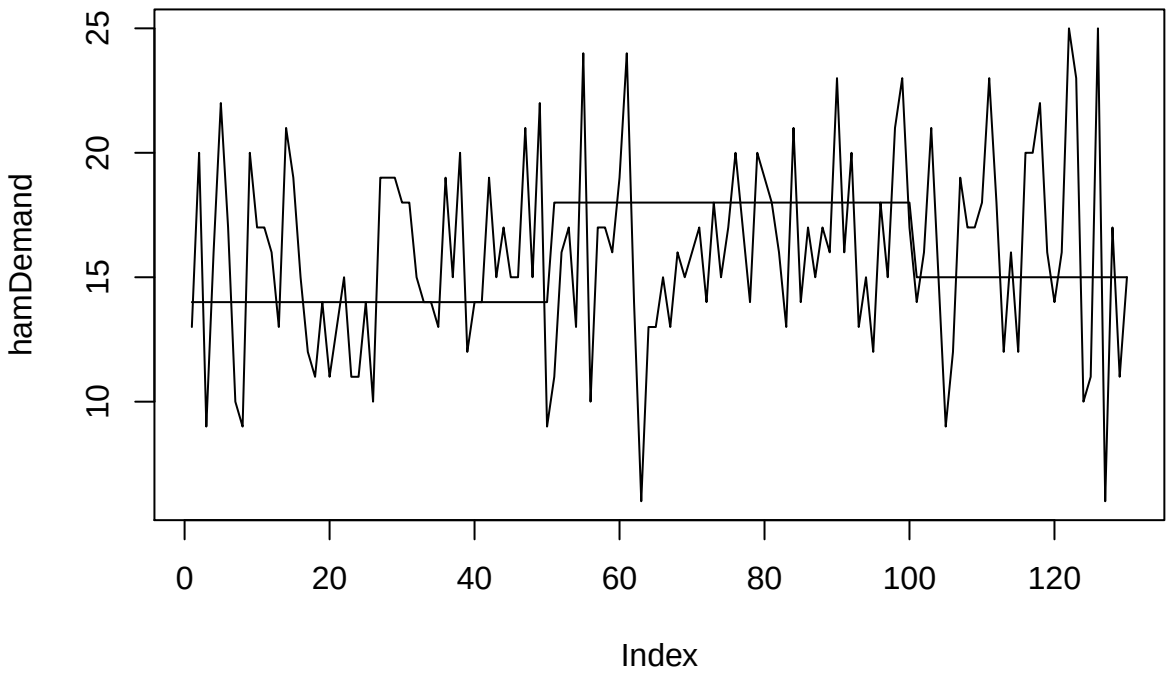
```
# load the data
colClasses=c('POSIXct','numeric','numeric','numeric','numeric','numeric','numeric')
sales <- read.csv("C:/Users/dgn2/Documents/R/IS606/sales.csv", header=TRUE,
                  stringsAsFactors=FALSE,colClasses=colClasses)
details <- read.csv("C:/Users/dgn2/Documents/R/IS606/details.csv", header=TRUE,
                    stringsAsFactors=FALSE)

dimension<-dim(sales)
nRows<-dimension[1]
nCols<-dimension[2]

# extract the price and cost of each product
hamPrice<-details$price[1]
hamCost<-details$cost[1]
turkeyPrice<-details$price[2]
turkeyCost<-details$cost[2]
veggiePrice<-details$price[3]
veggieCost<-details$cost[3]
```

Actual P&L

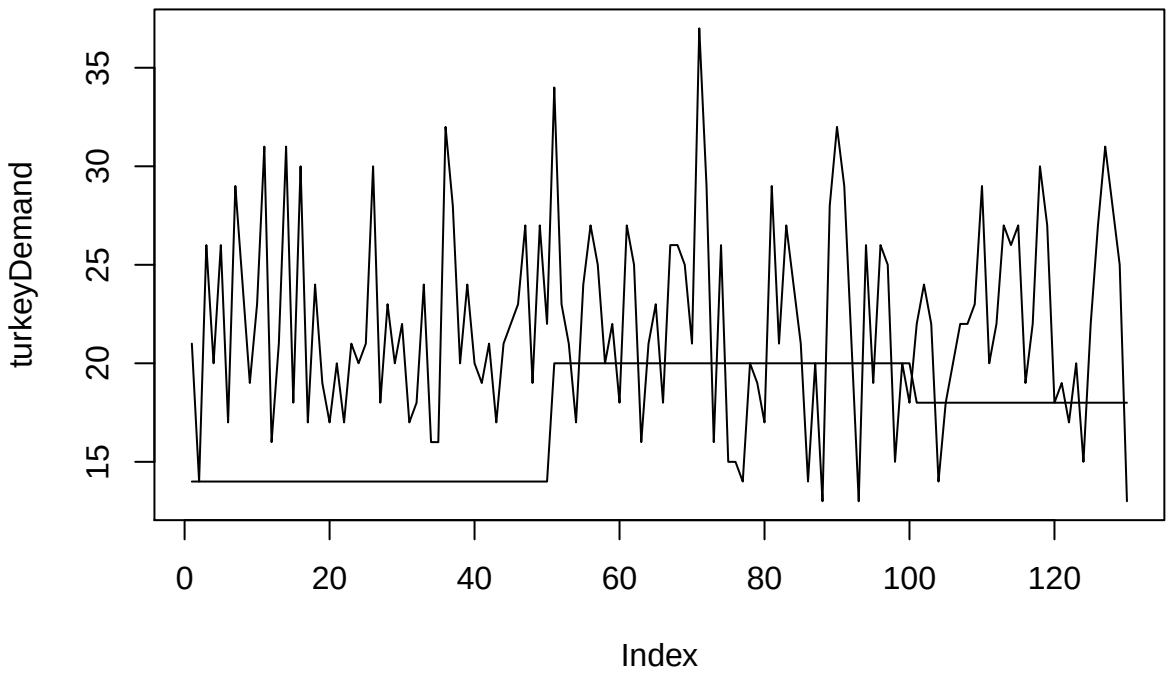
In computing the profit and loss associated with this venture, we assume the products that remain unsold out of inventory go to waste each day.

```
# define the function to compute revenue, expense, and P&L
pnlUnderScenario<- function(demand,supply,pricePerUnit,costPerUnit){
  expense<-supply*costPerUnit
  unitsSold<-demand
  flag<-supply-demand<0
  unitsSold[flag]<-supply[flag]
  revenue<-unitsSold*pricePerUnit
  pnl<-sum(revenue-expense)
  }
```

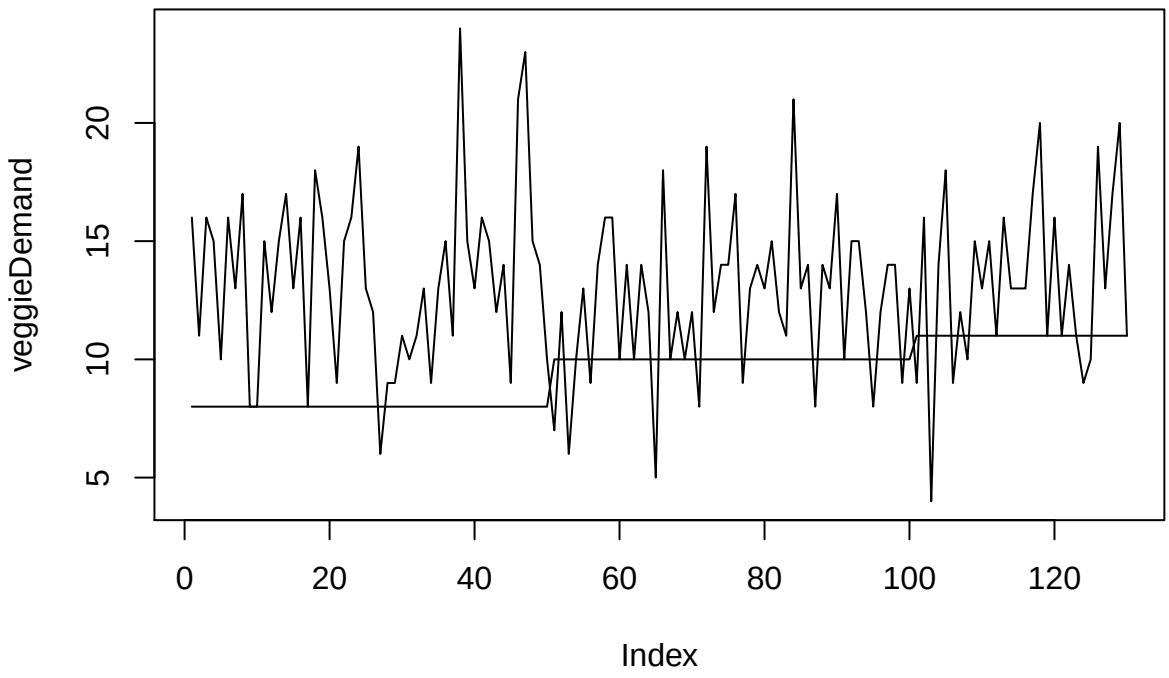
```
# ham revenue, expense, and P&L
hamDemand<-sales[,2]
hamSupply<-sales[,5]
hamExpense<-hamSupply*hamCost
hamUnitsSold<-hamDemand
hamFlag<-hamSupply-hamDemand<0
hamUnitsSold[hamFlag]<-hamSupply[hamFlag]
hamRevenue<-hamUnitsSold*hamPrice
hamPnl<-hamRevenue-hamExpense
plot(hamDemand,type='l')
lines(hamSupply,type='l')
```



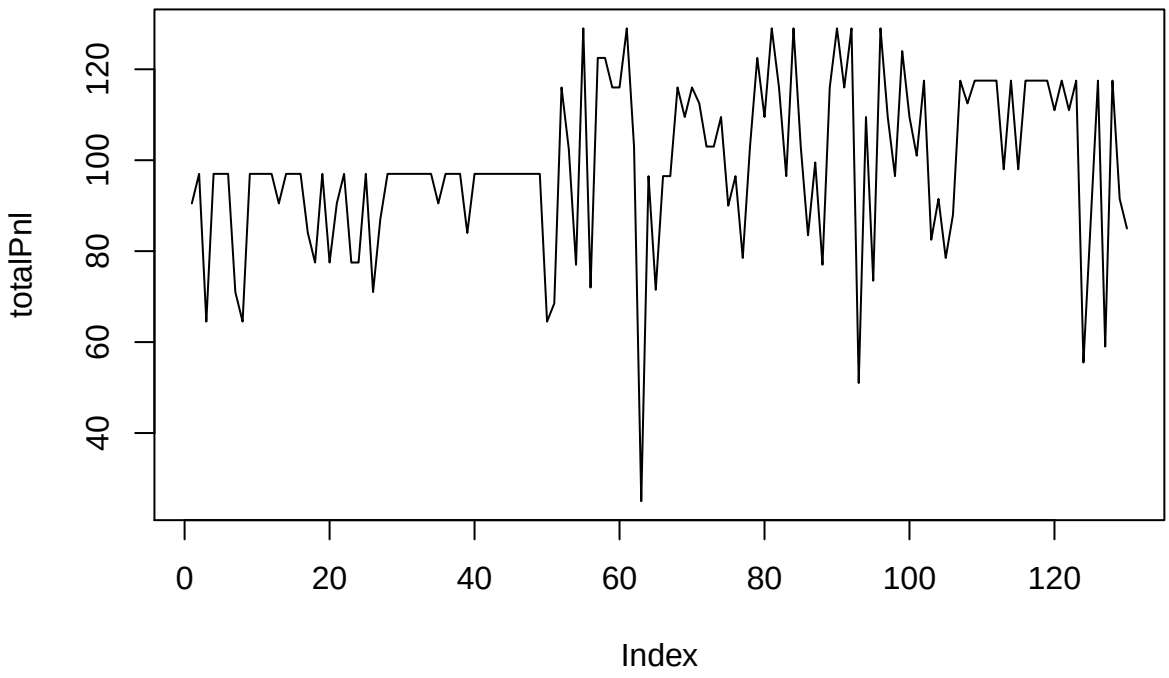

```
# turkey revenue, expense, and P&L
turkeyDemand<-sales[,3]
turkeySupply<-sales[,6]
turkeyExpense<-turkeySupply*turkeyCost
turkeyUnitsSold<-turkeyDemand
turkeyFlag<-turkeySupply-turkeyDemand<0
turkeyUnitsSold[turkeyFlag]<-turkeySupply[turkeyFlag]
turkeyRevenue<-turkeyUnitsSold*turkeyPrice
turkeyPnl<-turkeyRevenue-turkeyExpense
plot(turkeyDemand,type='l')
lines(turkeySupply,type='l')
```




```
# turkey revenue, expense, and P&L
veggieDemand<-sales[,4]
veggieSupply<-sales[,7]
veggieExpense<-veggieSupply*veggieCost
veggieUnitsSold<-veggieDemand
veggieFlag<-veggieSupply-veggieDemand<0
veggieUnitsSold[veggieFlag]<-veggieSupply[veggieFlag]
veggieRevenue<-veggieUnitsSold*veggiePrice
veggiePnl<-veggieRevenue-veggieExpense
plot(veggieDemand,type='l')
lines(veggieSupply,type='l')
```




```
# total revenue, expense, and P&L
totalExpense<-hamExpense+turkeyExpense+veggieExpense
totalRevenue<-hamRevenue+turkeyRevenue+veggieRevenue
totalPnl<-totalRevenue-totalExpense
actualPnL<-sum(totalPnl)
actualPnLPerDay<-actualPnL/nRows
plot(totalPnl,type='l')
```



Bootstrapping

```
# set the parameters for resampling

nPaths<-1000

# create the data resampling index
resampleIndex<-sample(1:nRows,nRows*nPaths,replace=TRUE,
                              prob=NULL)

# create the resampled data for each column
resampledData<-sales[resampleIndex,]
```

```
# reshape the data (nRows x nPaths)
hamDemandPaths<-data.frame(matrix(resampledData[,2],nrow=nRows,ncol=nPaths))
turkeyDemandPaths<-data.frame(matrix(resampledData[,3],nrow=nRows,ncol=nPaths))
veggieDemandPaths<-data.frame(matrix(resampledData[,4],nrow=nRows,ncol=nPaths))
hamSupplyPaths<-data.frame(matrix(resampledData[,5],nrow=nRows,ncol=nPaths))
turkeySupplyPaths<-data.frame(matrix(resampledData[,6],nrow=nRows,ncol=nPaths))
veggieSupplyPaths<-data.frame(matrix(resampledData[,7],nrow=nRows,ncol=nPaths))
```

We find the distribution of P&L by resampling the original supply and demand.

```
# find the distribution of P&L using the original supply and demand
hamPnLUnderScenarios<-0
turkeyPnLUnderScenarios<-0
veggiePnLUnderScenarios<-0
totalPnLUnderScenarios<-0
for (pathIndex in 1:nPaths){
  hamPnLUnderScenarios[pathIndex]<-pnlUnderScenario(
    hamDemandPaths[,pathIndex],hamSupplyPaths[,pathIndex],
    hamPrice,hamCost)
  turkeyPnLUnderScenarios[pathIndex]<-pnlUnderScenario(
    turkeyDemandPaths[,pathIndex],turkeySupplyPaths[,pathIndex],
    turkeyPrice,turkeyCost)
  veggiePnLUnderScenarios[pathIndex]<-pnlUnderScenario(
    veggieDemandPaths[,pathIndex],veggieSupplyPaths[,pathIndex],
    veggiePrice,veggieCost)
  }
totalPnLUnderScenarios=hamPnLUnderScenarios+turkeyPnLUnderScenarios+
  totalPnLUnderScenarios
```

Optimizing Expected P&L

We can determine the distribution of P&L for a range of supply scenarios, then find the supply that maximizes expected P&L for each product.

```
# maximize veggie P&L for each price scenario
hamSupplyRange<-min(hamDemand):max(hamDemand)
nScenarios<-length(hamSupplyRange)
hamPnLUnderSupplyScenarios<-matrix(rep(0,nPaths*nScenarios),
                                   nrow=nPaths,ncol=nScenarios)

for (scenarioIndex in 1:nScenarios){

  hamSupplyScenario<-rep(hamSupplyRange[scenarioIndex],nRows,
                         nrow=nRows,ncol=1)

  for (pathIndex in 1:nPaths){
    hamPnLUnderSupplyScenarios[pathIndex,scenarioIndex]<-pnlUnderScenario(
      hamDemandPaths[,pathIndex],hamSupplyScenario,
      hamPrice,hamCost)
  }
}
#
hamExpectedPnLUnderScenario<-colMeans(hamPnLUnderSupplyScenarios)
# add code to compute the percentile bounds

hamMaxExpectedPnLUnderScenario<-max(hamExpectedPnLUnderScenario)
#
hamMaxIndex<-hamExpectedPnLUnderScenario==hamMaxExpectedPnLUnderScenario
#
hamOptimalSupply<-hamSupplyRange[hamMaxIndex]
```

```
# maximize veggie P&L for each price scenario
turkeySupplyRange<-min(turkeyDemand):max(turkeyDemand)
nScenarios<-length(turkeySupplyRange)
turkeyPnLUnderSupplyScenarios<-matrix(rep(0,nPaths*nScenarios),
                                      nrow=nPaths,ncol=nScenarios)

for (scenarioIndex in 1:nScenarios){

  turkeySupplyScenario<-rep(turkeySupplyRange[scenarioIndex],nRows,
                            nrow=nRows,ncol=1)

  for (pathIndex in 1:nPaths){
    turkeyPnLUnderSupplyScenarios[pathIndex,scenarioIndex]<-pnlUnderScenario(
      turkeyDemandPaths[,pathIndex],turkeySupplyScenario,
      turkeyPrice,turkeyCost)
  }
}
turkeyExpectedPnLUnderScenario<-colMeans(turkeyPnLUnderSupplyScenarios)
# add code to compute the percentile bounds

#
turkeyMaxExpectedPnLUnderScenario<-max(turkeyExpectedPnLUnderScenario)
#
turkeyMaxIndex<-turkeyExpectedPnLUnderScenario==turkeyMaxExpectedPnLUnderScenario
#
turkeyOptimalSupply<-turkeySupplyRange[turkeyMaxIndex]
```

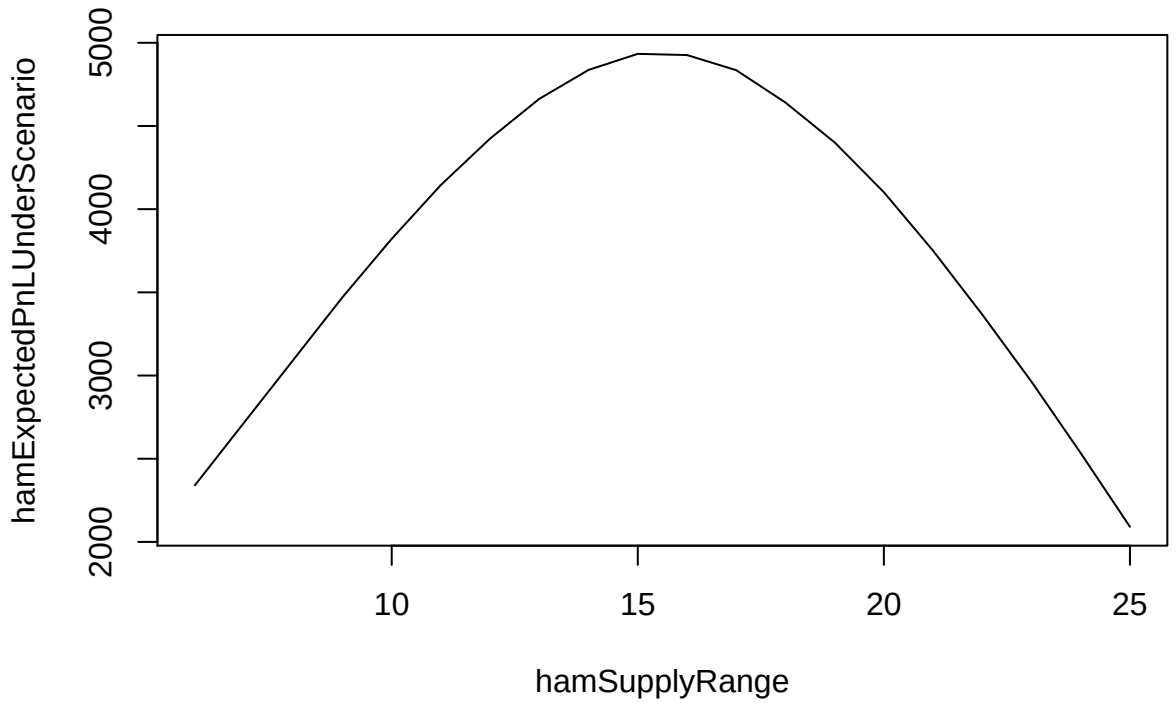
```
# maximize veggie P&L for each price scenario
veggieSupplyRange<-min(veggieDemand):max(veggieDemand)
nScenarios<-length(veggieSupplyRange)
veggiePnLUnderSupplyScenarios<-matrix(rep(0,nPaths*nScenarios),
                                      nrow=nPaths,ncol=nScenarios)

for (scenarioIndex in 1:nScenarios){

  veggieSupplyScenario<-rep(veggieSupplyRange[scenarioIndex],nRows,
                            nrow=nRows,ncol=1)

  for (pathIndex in 1:nPaths){
    veggiePnLUnderSupplyScenarios[pathIndex,scenarioIndex]<-pnlUnderScenario(
      veggieDemandPaths[,pathIndex],veggieSupplyScenario,
      veggiePrice,veggieCost)
  }
}
#
veggieExpectedPnLUnderScenario<-colMeans(veggiePnLUnderSupplyScenarios)
# add code to compute the percentile bounds

#
veggieMaxExpectedPnLUnderScenario<-max(veggieExpectedPnLUnderScenario)
#
veggieMaxIndex<-veggieExpectedPnLUnderScenario==veggieMaxExpectedPnLUnderScenario
#
```

```
veggieOptimalSupply<-veggieSupplyRange[veggieMaxIndex]
```

```
totalMaxExpectedPnLUnderScenario<-hamMaxExpectedPnLUnderScenario+
  turkeyMaxExpectedPnLUnderScenario+veggieMaxExpectedPnLUnderScenario
```

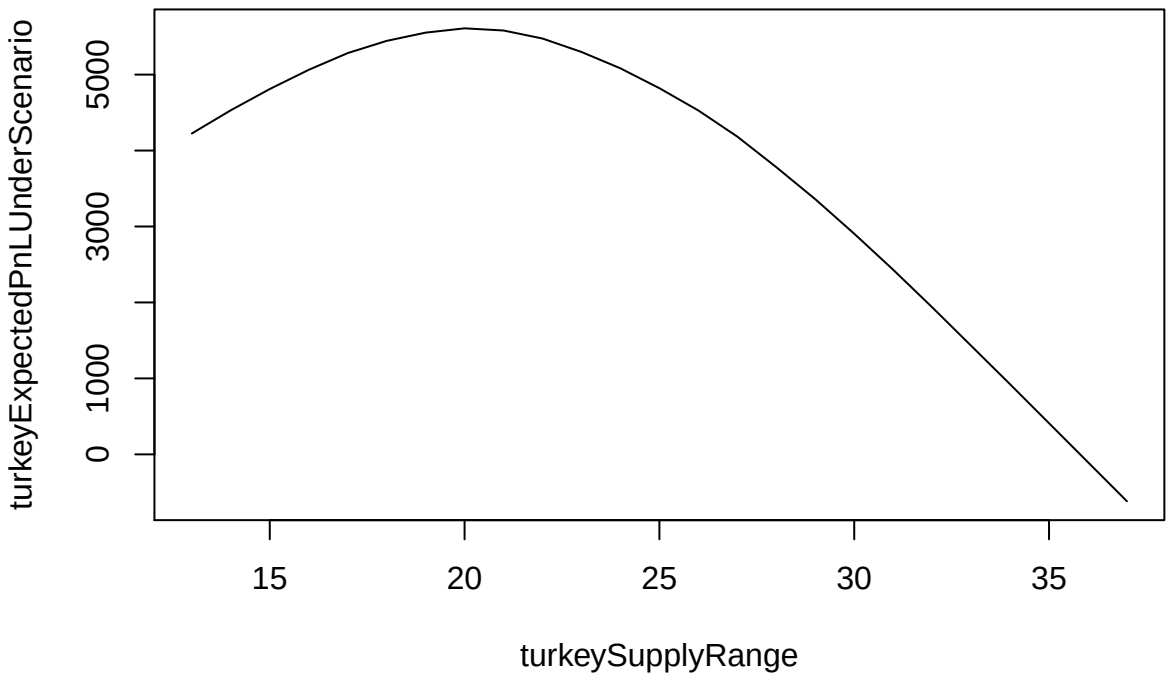
The expected P&L of 37.94995 by day for ham is maximized by supplying 15 units each day.

```
#
plot(hamSupplyRange,hamExpectedPnLUnderScenario,type='l')
```



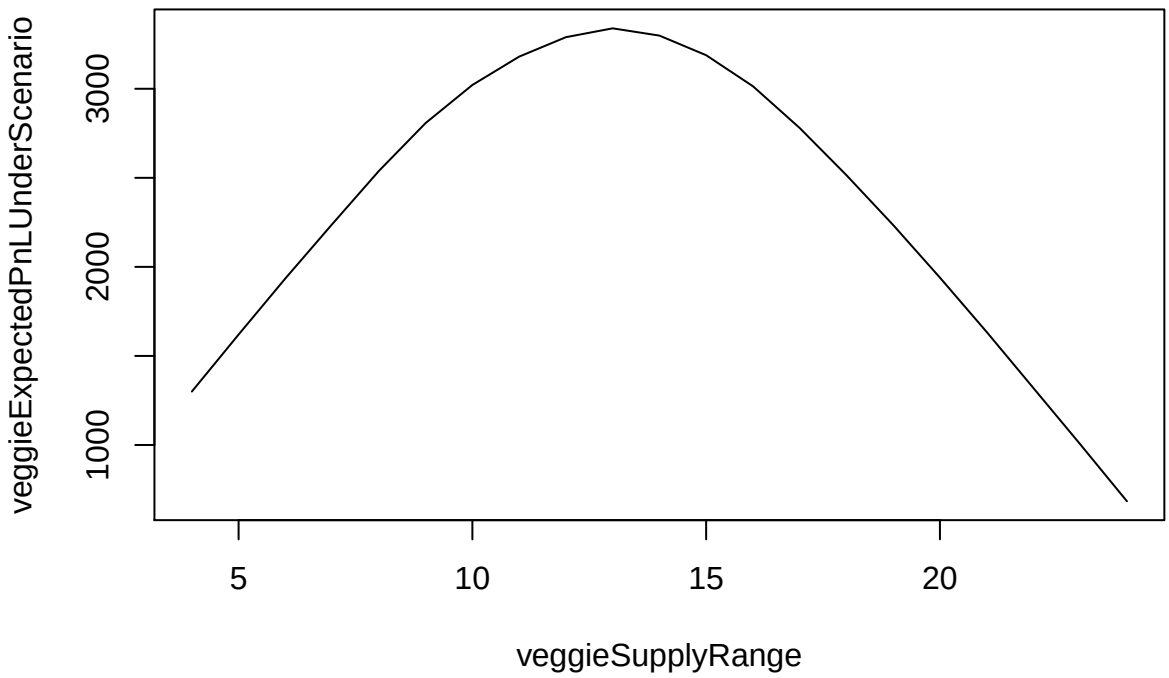
The expected P&L of 43.1465 by day for turkey is maximized by supplying 20 units each day.

```
#
plot(turkeySupplyRange,turkeyExpectedPnLUnderScenario,type='l')
```



The expected P&L of 25.6869615 by day for veggie is maximized by supplying 13 units each day.

```
#
plot(veggieSupplyRange,veggieExpectedPnLUnderScenario,type='l')
```



The expected P&L for the entire venture is 106.7834115 per day (up from 98.6769231).