Restaurant log-Analysis

Joaquim Assun????o

August 26, 2016

Required Packages

```
require(reshape)
require(ggplot2)
require(forecast)
require(markovchain)
require(zoo)
```

—– Instructions ——
1) Use the file *"restaurant_tot_tidy.R"* to read *"sample.txt"* and generate the file *"DF_rest_tot.csv"*.
2) Change you directory to the current one with the *"setwwd()"* function.

---

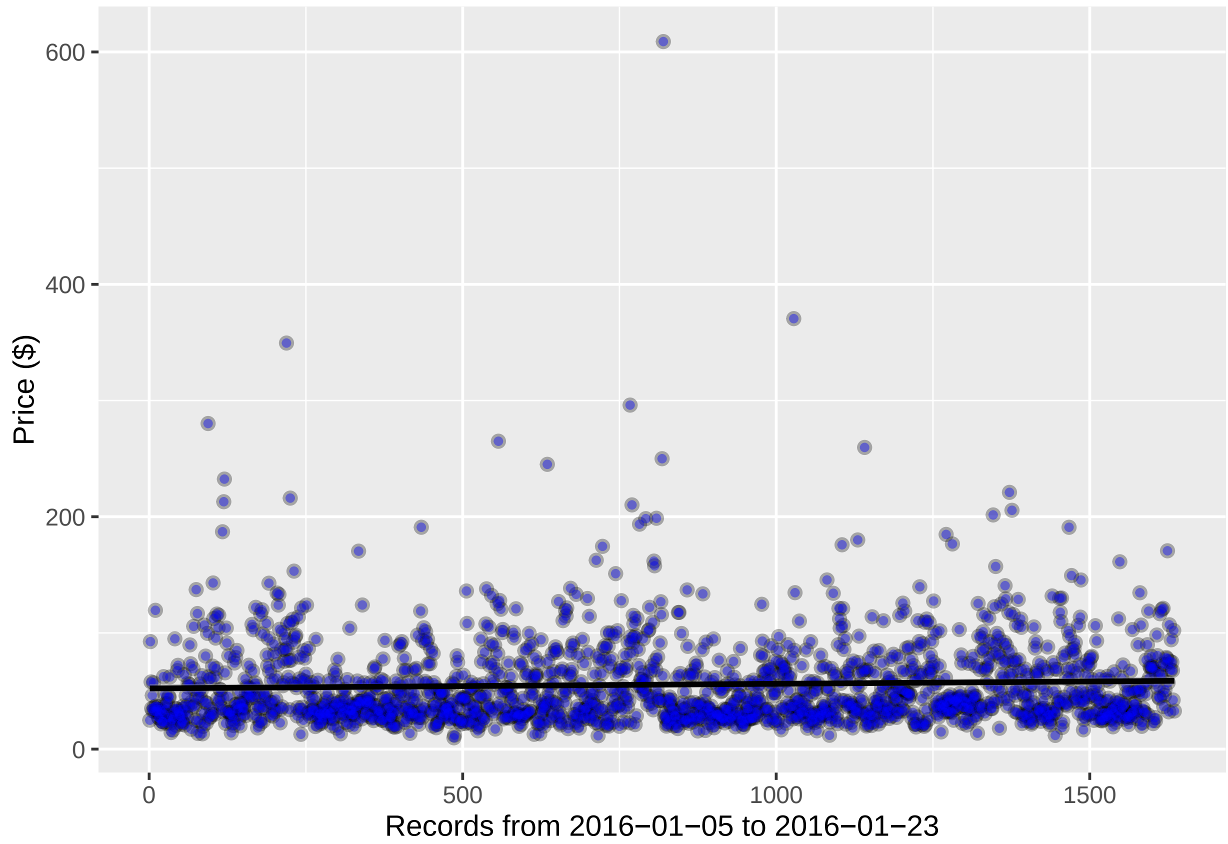
1) Data Loading and Basic Analysis

```
# Load DF_rest_tot into DF_rest_tot DF
DF_rest_tot <- read.csv("DF_rest_tot.csv")

# First lets know some basic information of our data
summary(DF_rest_tot$valorTotal)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.74   30.98   43.26   55.50   69.23  608.90
```

```
# Now, lets see our data
g = ggplot(DF_rest_tot, aes(x = 1:nrow(DF_rest_tot), y = valorTotal))
g = g + xlab("Records from 2016-01-05 to 2016-01-23")
g = g + ylab("Price ($)")
g = g + geom_point(size = 2, colour = "black", alpha=0.3)
g = g + geom_point(size = 1, colour = "blue", alpha=0.4)
g = g + geom_smooth(method = "lm", colour = "black")
g
```

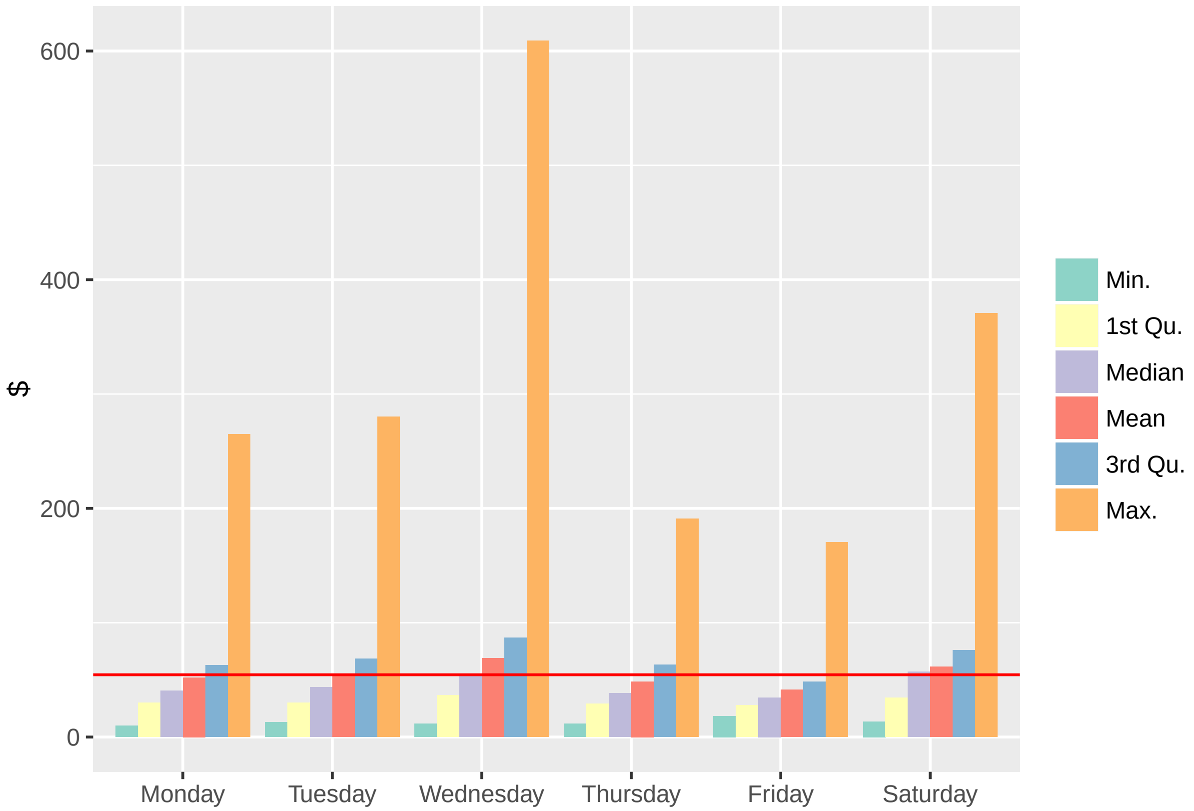



```
# Considering the week days, are the values too different?
DF_statistics2 <- data.frame()
DF_statistics2 <- rbind(DF_statistics2,
                        summary(DF_rest_tot$valorTotal[DF_rest_tot$week_day == "Monday"]))
DF_statistics2 <- rbind(DF_statistics2,
                        summary(DF_rest_tot$valorTotal[DF_rest_tot$week_day == "Tuesday"]))
DF_statistics2 <- rbind(DF_statistics2,
                        summary(DF_rest_tot$valorTotal[DF_rest_tot$week_day == "Wednesday"]))
DF_statistics2 <- rbind(DF_statistics2,
                        summary(DF_rest_tot$valorTotal[DF_rest_tot$week_day == "Thursday"]))
DF_statistics2 <- rbind(DF_statistics2,
                        summary(DF_rest_tot$valorTotal[DF_rest_tot$week_day == "Friday"]))
DF_statistics2 <- rbind(DF_statistics2,
                        summary(DF_rest_tot$valorTotal[DF_rest_tot$week_day == "Saturday"]))
DF_statistics2$week_day <- c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
names(DF_statistics2)[1:6] <- names(
  summary(DF_rest_tot$valorTotal[DF_rest_tot$week_day == "Monday"]))

DF_statistics2.m <- melt(DF_statistics2, id.vars=7)
DF_statistics2.m$week_day <- factor(DF_statistics2.m$week_day,
                                    levels = c("Monday","Tuesday","Wednesday","Thursday","Friday","Satu


ggplot(DF_statistics2.m, aes(x=week_day, value)) +
  geom_bar(aes(fill = variable), position = "dodge", stat="identity") +
  scale_fill_brewer(palette="Set3") +
```

```
  geom_hline(yintercept = mean(DF_statistics2$Mean),colour="red") +
  xlab("") +
  ylab("$") +
  guides(fill=guide_legend(title=NULL))
```



Notes

This analysis shows that Wednesday is the most profitable day.
It also shows the basic statistics about this data (Median, Mean, 1st. Qu. *etc.*).
An appropriated analysis would involve some algorithm based on machine learning, perhaps a classifier such as RandomForests, or a rule associator such as Apriori. It would be easily implemented withing a few hours, plus an analysis time. Furthermore, to keep this document compact, I decided to not perform statistical tests and hypothesis.

1) Model

Now we fit a Markovian model aiming to forecast sales.

```
Xo <- floor(DF_rest_tot$valorTotal)
# Training with 61%
X <- Xo[1:1000]

# Should reduce its dimension here. But there is no need for a such small sample.
myFit<-markovchainFit(X)
```

```r
# fp stands to "Forward probabilities"
fp <- myFit$estimate[]
drawNames <- as.integer(colnames(fp))
fp <- data.frame(fp)
colnames(fp) <- drawNames
fp <- fp[order(as.numeric(rownames(fp))),order(as.numeric(colnames(fp)))]
```

1.1) Verification

```r
#Log-likelihood
myFit$logLikelihood
```

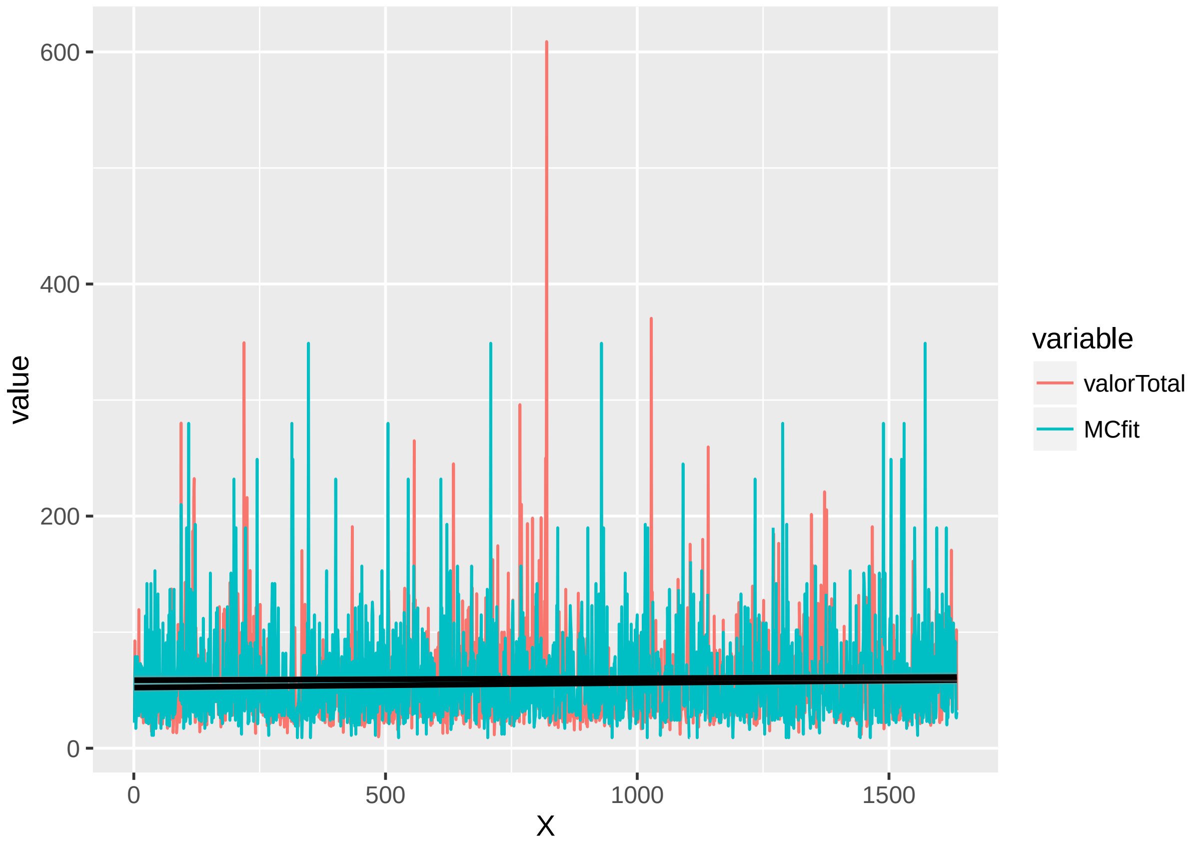
```
## [1] -2251.295
```

```r
result <- NULL
cumList <- sort(unique(X))
for( i in 2:(length(Xo)+1)){
  probVec <- fp[(which(drawNames==Xo[i-1])),]
  probVec <- as.numeric(probVec)
  cProbVec <- cumsum(probVec)
  rand <- runif(1)-0.01
  whichVec <- which(rand < cProbVec)[1]
  result <- c(result,cumList[whichVec])
}

# Creates a new field with the resultant simulation
DF_rest_tot$MCfit <- result

# Save tidy data ".m stands for melted"
DF_rest_tot.m <- melt(DF_rest_tot, id.vars=c(-2,-10))

g = ggplot(DF_rest_tot.m, aes(x=X, y=value, group=variable))
g = g + geom_line(aes(colour = variable))
g = g + geom_smooth(method = "lm", colour = "black")
g
```




```
# Now, using the model to test the last data (total income considering the last 7 days)
# We have log from 18 days
# ...1635 records 1635/18
# Lets roughly define 90.8 records per day: 635 records
# if NA, use the last value
sum(na.locf(DF_rest_tot$MCfit[1000:1635]))
```

```
## [1] 38379
```

```
# Now compare with the original log
sum(DF_rest_tot$valorTotal[1000:1635])
```

```
## [1] 36643.25
```

```
# The difference in %
(sum(DF_rest_tot$valorTotal[1000:1635])) / (sum(na.locf(DF_rest_tot$MCfit[1000:1635])))
```

```
## [1] 0.9547734
```

1.2) Forecasting

```r
# Convert my fitted model to a pure matrix
myMarkov <- matrix(myFit$estimate[,],ncol = dim(myFit$estimate[,]))
rownames(myMarkov) <- rownames(myFit$estimate[,])
colnames(myMarkov) <- colnames(myFit$estimate[,])

# Now I use this function to simulate the next weekion
DTMCsimulate <- function(mc,N) {
  walking <- function(char,mc) {
    sample(colnames(mc),1,prob=mc[char,])
  }
  sim <- character(N)
  sim[1] <- sample(colnames(mc),1)
  for (i in 2:N) {
    sim[i] <- walking(sim[i-1],mc)
  }
  sim
}

# Simulate 7 days, 635 records
result_next_week <- as.numeric(DTMCsimulate(myMarkov, 635))
# Voila
 print (paste("Next week should retrieve an average of",sum(result_next_week),"Dollars"))
```

```
## [1] "Next week should retrieve an average of 34855 Dollars"
```

```r
# Compared to the previous week...
  sum(result_next_week) / (sum(DF_rest_tot$valorTotal[1000:1635]))
```

```
## [1] 0.9511984
```

Model notes

Here I built a simple Discrete Time Markov Chain using the state of the art in this formalism. However, for a more robust analysis, it should integrate more variables (fields from our data). A Hidden Markov Model would improve the accuracy by using a non-direct inference. A Stochastic Automata Network would be able to model the system as a whole (Not only the sold value) by grouping data into a robust model. Unfortunately, it would need many pages to explain what is going on under the hood.