Final Project: Walmart Data Exploration

Stephen Kappel (spk2131), Mayank Misra (mm3557), Mandeep Singh (ms4826)

Due: December 5, 2015

Introduction

For our project, we chose to use sales data provided by Walmart as part of the *Walmart Recruiting - Store Sales Forecasting* competition on Kaggle in 2014. (See https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting.) The dataset contains weekly sales for 45 Walmart stores from 2/5/2010 to 11/1/2012. The sales are broken out by store and by department. Other attributes provide further context to the sales numbers:

- *Economic indicators:* CPI, unemployment rate, fuel prices
- *Store-specific attributes:* store size, store type, markdowns (indicator variables for five types of markdowns)
- *Other:* temperature, holiday (indicator variable)

In our analysis, sales is the outcome variable of primary interest. We aim to understand what factors impact sales. In particular, we try to answer:

1. How do economic factors – over which Walmart has no control – affect Walmart's sales? Are some departments and store types impacted more significantly than others by economic factors?
2. How well can we predict sales? In predicting sales, which predictors are most significant?

Data Preparation

Kaggle provides the data in the form of three CSVs. We create two data.frames that combine the three source CSVs together. `dept.level` contains one row per department per store per week. `store.level` contains one row per store per week.

```r
require(plyr)

get.stores <- function(){
  stores <- read.csv('../data/stores.csv')
  stores$Store <- as.factor(stores$Store)
  return(stores)
}

get.features <- function(){
  features <- read.csv('../data/features.csv')
  features$Store <- as.factor(features$Store)
  features$Date <- as.Date(features$Date)
  return(features)
}

get.train <- function(){
  train <- read.csv('../data/train.csv')
  train$Store <- as.factor(train$Store)
  train$Dept <- as.factor(train$Dept)
  train$Date <- as.Date(train$Date)
  return(train)
}

get.dept.level <- function(){
  # construct a data.frame with all the data with detail down to the department level
  train <- get.train()
  stores <- get.stores()
  features <- get.features()
  master <- merge(train, stores, by='Store')
  master <- merge(master, features, by=c('Store', 'Date'))
  master$IsHoliday.y <- NULL  # IsHoliday is in features and train
  master$Sales_Millions <- master$Weekly_Sales / 10e6
  master <- rename(master, c('Type'='Store_Type', 'Size'='Store_Size',
                             'IsHoliday.x'='IsHoliday', 'Weekly_Sales'='Sales'))
  return(master)
}

get.store.level <- function(){
  # construct a data.frame with detail only down to the store level (no departments)
  train <- get.train()
  train <- ddply(train, c('Store', 'Date'), summarize, Sales=sum(Weekly_Sales))
  stores <- get.stores()
  features <- get.features()
  merged <- merge(train, stores, by='Store')
  merged <- merge(merged, features, by=c('Store', 'Date'))
  merged$Sales_Millions <- merged$Sales / 10e6
  merged <- rename(merged, c('Type'='Store_Type', 'Size'='Store_Size'))
  return(merged)
}

# dept.level <- get.dept.level()
store.level <- get.store.level()
```

As we do our analysis, we often find it helpful to normalize a column within the context of a store. This allows for more meaningful comparisons. Because this is a common operation, we define a function to add a nomalized column to a given data.frame for a specified column/attribute. To start, we add a normalized sales column to the `store.level` data.frame.

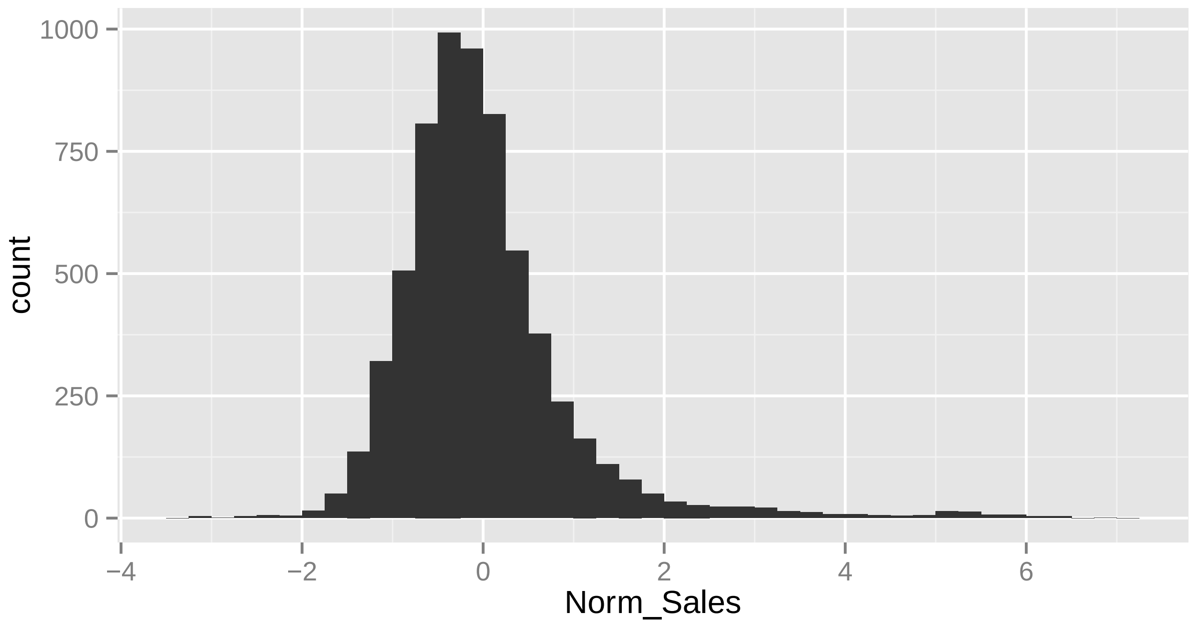
```
add.normalized.col <- function(df, col.name){
  df$temp <- df[, col.name]
  store.ply <- ddply(df, c('Store'), summarize, Mean_X=mean(temp),
                     SD_X=sd(temp))
  df <- merge(df, store.ply, by=c('Store'))
  df$Norm_X <- (df[,col.name] - df$Mean_X) / df$SD_X
  df$Mean_X <- NULL
  df$SD_X <- NULL
  df$temp <- NULL
  df <- rename(df, c('Norm_X'=paste('Norm_', col.name, sep='')))
  return(df)
}

store.level <- add.normalized.col(store.level, 'Sales')
```

We also commonly want to get sales aggregates after grouping and/or filtering by some attribute, so we define a function for this operation.

```r
get.aggr.sales <- function(df, aggr.dims, filter=NULL){
  # df: master data.frame
  # aggr.dims: a list of column names to aggregate on
  # filter: NULL or c(column name to filter, value to filter to)
  if(!is.null(filter)){
    sub.df <- subset(df, df[,filter[1]] == filter[2])
  } else{
    sub.df <- df
  }
  grouped <- ddply(sub.df, aggr.dims, summarize, Sales=sum(Sales),
                   Sales_Millions=sum(Sales_Millions))
  return(grouped)
}
```

Let's take a look at the normalized distribution of store-level weekly sales. We hope this is roughly normal as it will allow us to apply some statistical tests with more confidence in the coming analysis.

```r
require(ggplot2)
ggplot(data=store.level, aes(x=Norm_Sales)) + geom_histogram(binwidth=0.25)
```



The distribution appears roughly symmetric and normal, although there is a light long tail on the positive side.

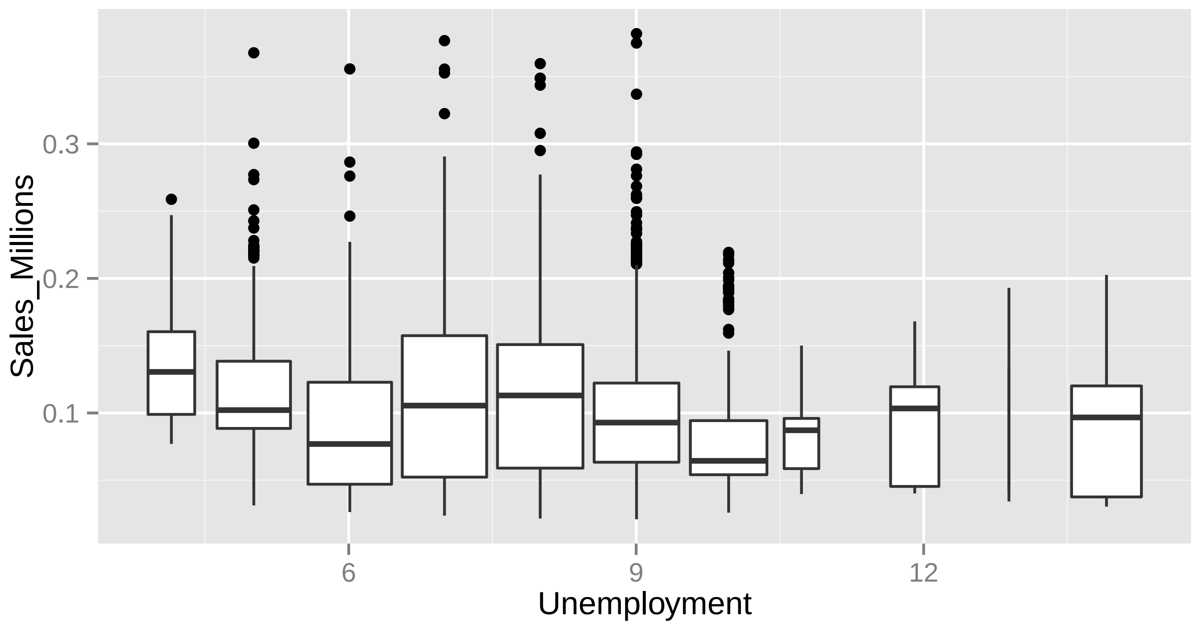
Impact of economic factors

In this section we explore how economic factors (unemployment, CPI, and fuel price) relate to Walmart's sales. It's not immediately obvious how we should expect Walmart's sales to vary with economic conditions.

While poor economic conditions hurt most retailers, Walmart is known for low prices. Could bad economic times drive more people to shop at Walmart?

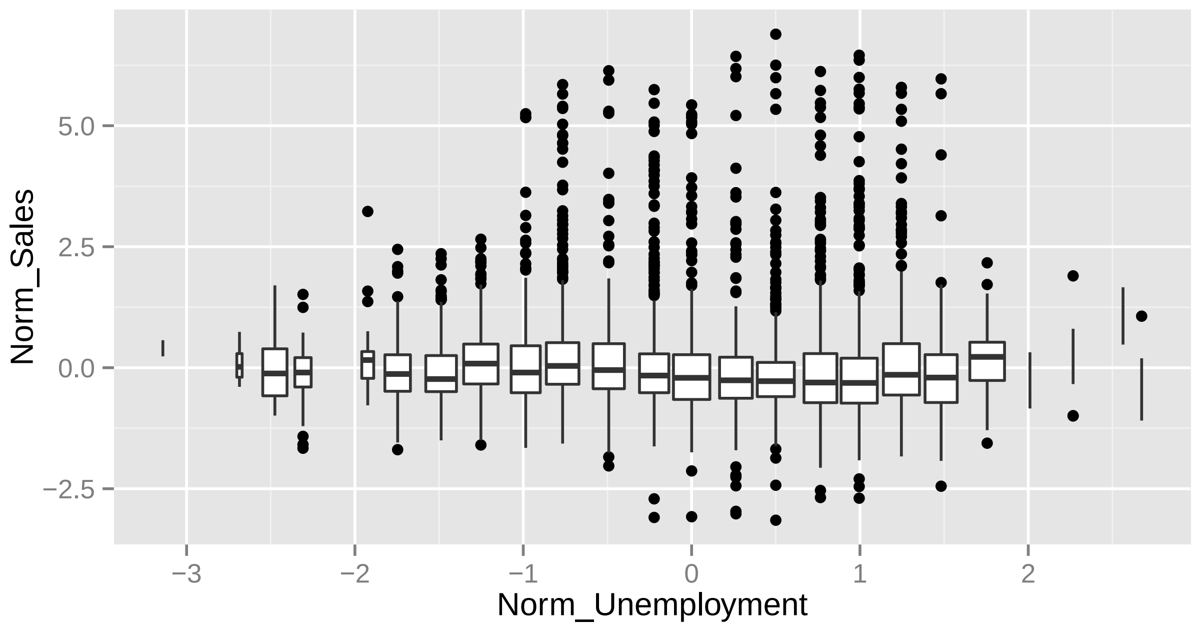
We start by creating a boxplot of sales by unemployment rate.

```r
ggplot(data=store.level, aes(x=Unemployment, y=Sales_Millions,
  group=round_any(Unemployment, 1.0))) + geom_boxplot()
```



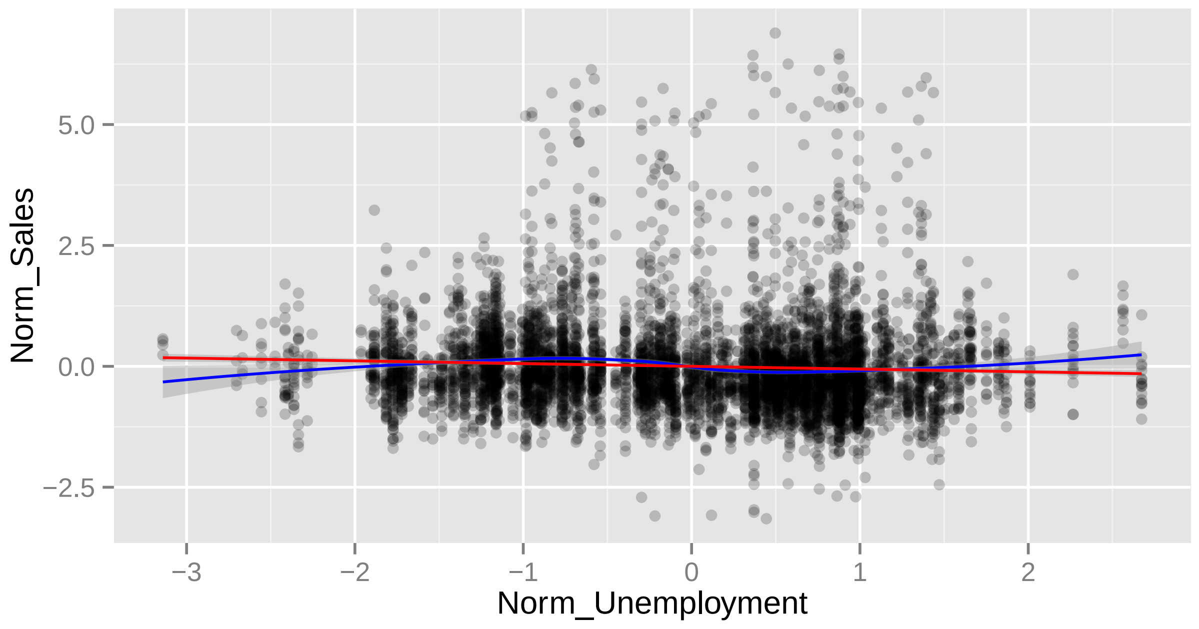
This plot don't show much of a relationship, but we haven't done any normalization, so this probably isn't a meaningful representation of the relationship. We normalize sales and unemployment rate by store and create an update boxplot.

```r
store.level <- add.normalized.col(store.level, 'Unemployment')
ggplot(data=store.level, aes(x=Norm_Unemployment, y=Norm_Sales,
  group=round_any(Norm_Unemployment, 0.25))) + geom_boxplot()
```



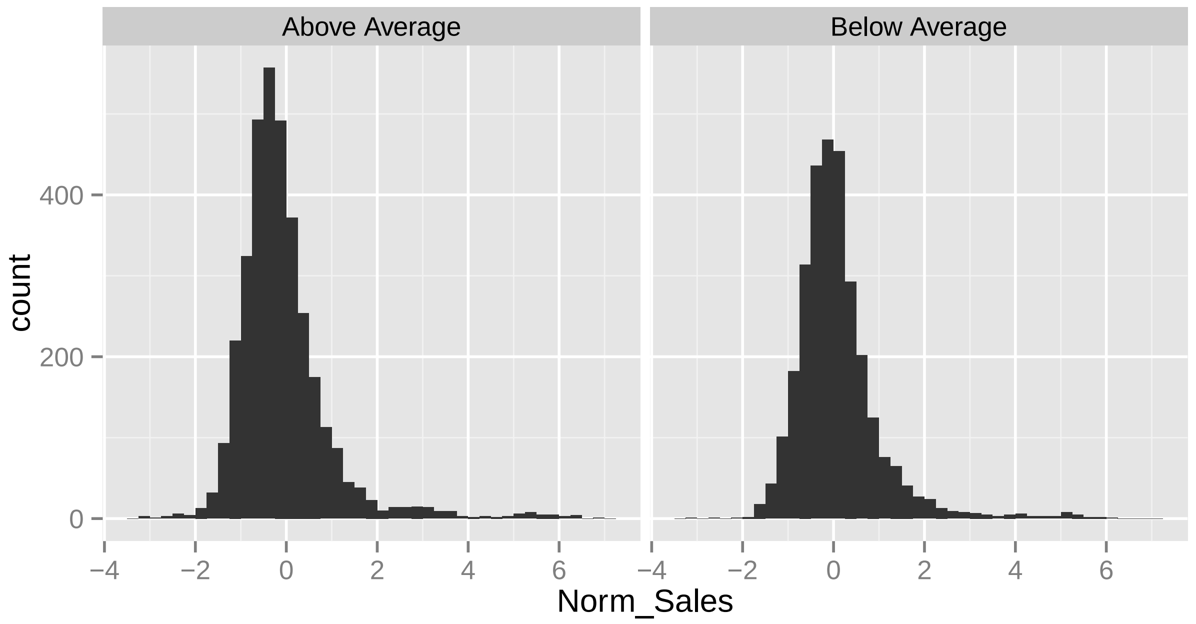
The same data in a scatter plot with LOESS (blue) and GAM (red) curves overlaid:

```r
ggplot(data=store.level, aes(x=Norm_Unemployment, y=Norm_Sales)) +
  geom_point(alpha = 1/5) + geom_smooth(method='loess', color='blue') +
  geom_smooth(method='gam', color='red')
```



Within the -1 to 1 normalized unemployment range, a patter seems to have emerged. There appears to be below-average sales when there is above-average unemployment. Let's create a pair of histograms showing number of weeks by normalized sales - one histogram for weeks with below average unemployment and on histogram for weeks with above average unemployment. We want to see if these distributions are noticably different. We also want to see the shape of these distributions to see if the data is normally distributed and a t-test would be reasonable.

```
store.level$Unemployment_Category <- ifelse(
  store.level$Norm_Unemployment < 0, 'Below Average', 'Above Average')
ggplot(data=store.level, aes(x=Norm_Sales)) + geom_histogram(binwidth=0.25) +
  facet_wrap(~Unemployment_Category)
```



The relationship between unemployment and sales isn't too clear from this plot, but the populations do appear to be roughly normal. We now run an F-test to check if the populations have equal variances (as assumed by a two-sample t-test).

```
above <- subset(store.level, Unemployment_Category == 'Above Average')
below <- subset(store.level, Unemployment_Category == 'Below Average')
var.test(above$Norm_Sales, below$Norm_Sales)
```

```
##
##  F test to compare two variances
##
## data:  above$Norm_Sales and below$Norm_Sales
## F = 1.2265, num df = 3477, denom df = 2956, p-value = 9.068e-09
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  1.144162 1.314409
## sample estimates:
## ratio of variances
##           1.226456
```

We reject the null hypothesis that the populations have equal variance, so we proceed with Welsh t-test for populations with unequal variances. The null hypothesis of this t-test is that the mean normalized sales during weeks with above average unemployment rates is equal to the mean normalized sales during weeks with below average umemployment rates.

```r
t.test(above$Norm_Sales, below$Norm_Sales, var.equal=FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  above$Norm_Sales and below$Norm_Sales
## t = -6.831, df = 6409.719, p-value = 9.209e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2166014 -0.1200037
## sample estimates:
##   mean of x   mean of y
## -0.07733810  0.09096447
```

We can confidently reject the null hypothesis. Now, we drop our assumption of normality, and perform a non-parametric test to see determine if sales for weeks with above-average unemployment are drawn from a different distribution than the sales for weeks with below-average unemployment. We apply the Mann-Whitney-Wilcoxon procedure below.

```r
wilcox.test(Norm_Sales ~ Unemployment_Category, data=store.level)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Norm_Sales by Unemployment_Category
## W = 4330099, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

From all the results above, we can conclude that unemployment rate is not independent of Walmart sales.

We repeat the same procedures for CPI and fuel price. We hide the code and output to save space, but we summarize the p-values from the tests in the table below:

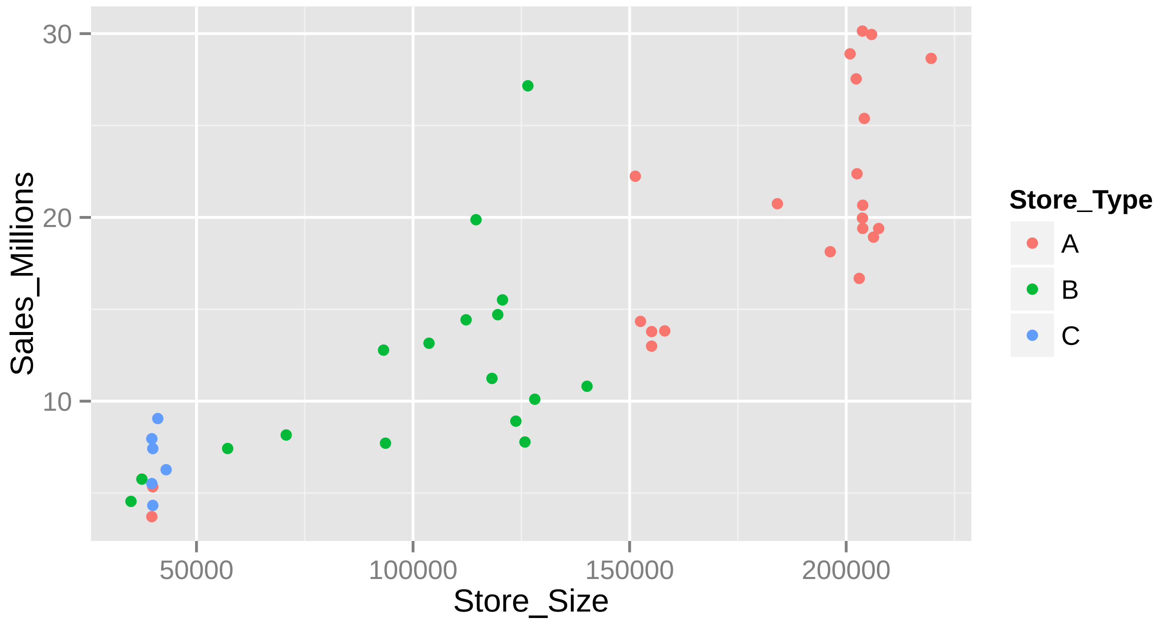
Economic Factor	F-test 95% CI	t-test 95% CI	Mann-Whitney-Wilcoxon p-value
Unemployment	(1.14, 1.31)	(-0.22, -0.12)	2.2e-16
CPI	(0.83, 0.95)	(0.11, 0.21)	2.2e-16
Fuel price	(0.34, 0.39)	(-0.15, -0.04)	3.9e-4

All three variables tell a similar story. When external economic factors are unfavorable (high unemployment, low CPI, and high fuel prices), Walmart sales are not as strong as they are during times when external economic factors are favorable.

Sales predictions

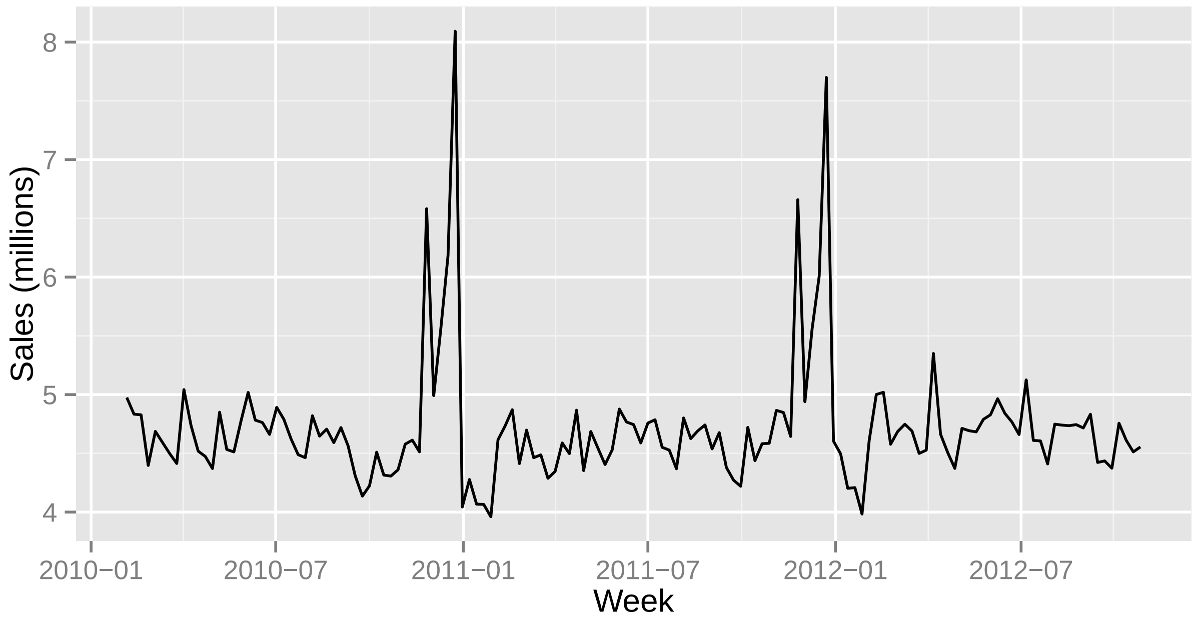
In order to make good predictions about sales, we need to consider much more than just the economic factors we've considered so far. For example, there is roughly a linear relationship between a store's sales and its size.

```
sales.by.size <- get.aggr.sales(store.level, c('Store', 'Store_Type', 'Store_Size'))
ggplot(data=sales.by.size, aes(x=Store_Size, y=Sales_Millions, color=Store_Type)) +
  geom_point()
```

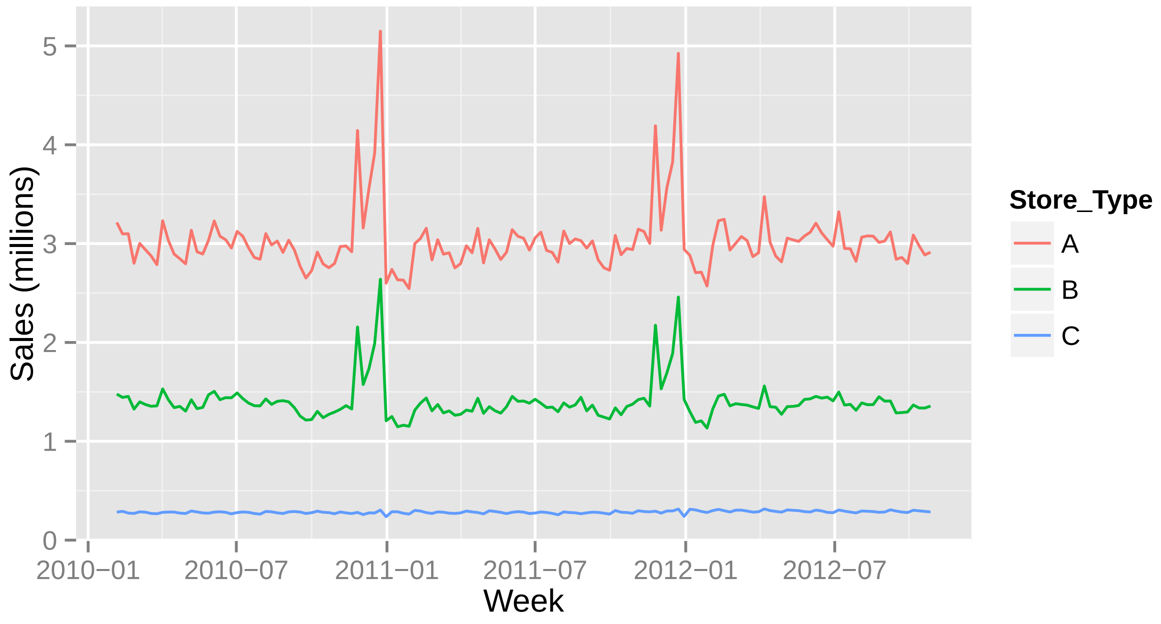


We also notice in the plot above that store type is closely related to both store sales and store size. The next couple plots show that dates will also be important in our prediction, as we observe seasonality and trending patters in the timeseries.

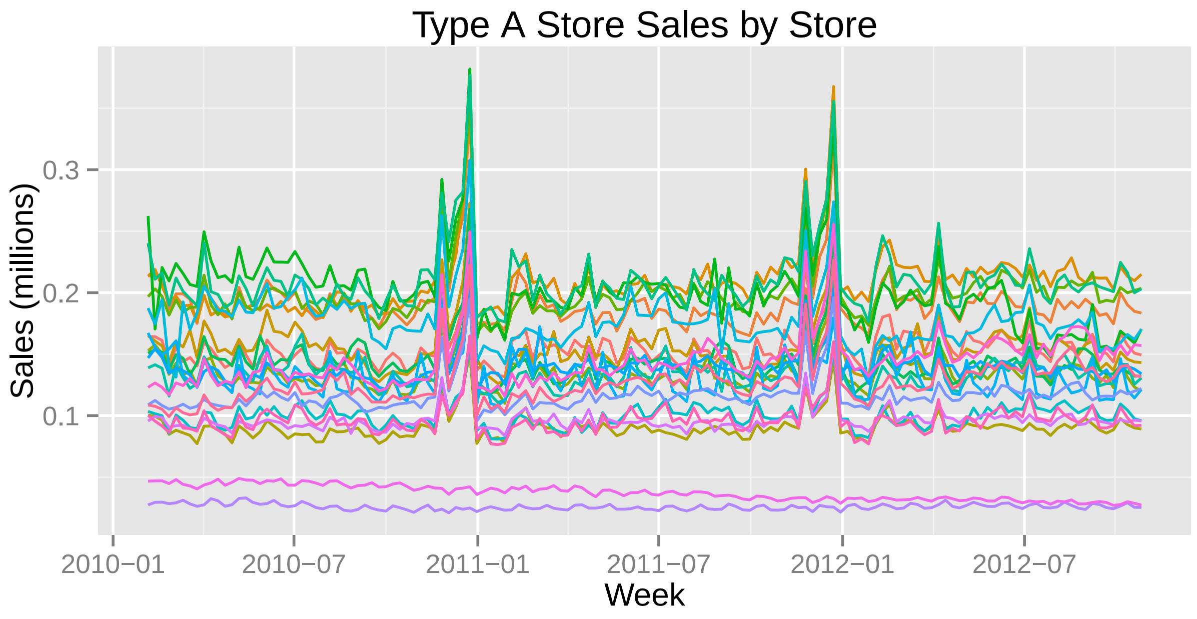
```
# plot sales by week
sales.by.week <- get.aggr.sales(store.level, c('Date'))
labels <- labs(x='Week', y='Sales (millions)')
ggplot(data=sales.by.week, aes(x=Date, y=Sales_Millions)) + geom_line() + labels
```



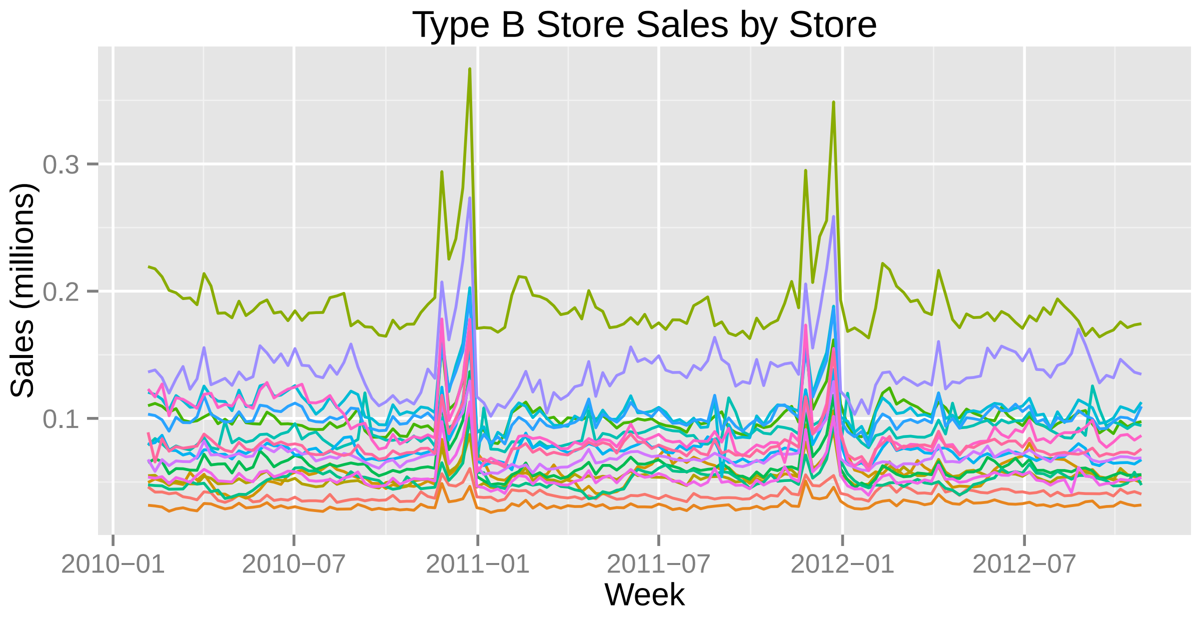
```
# plot sales by week by store type
sales.by.type <- get.aggr.sales(store.level, c('Store_Type','Date'))
ggplot(data=sales.by.type, aes(x=Date, y=Sales_Millions, group=Store_Type, color=Store_Type)) + geom_li
```



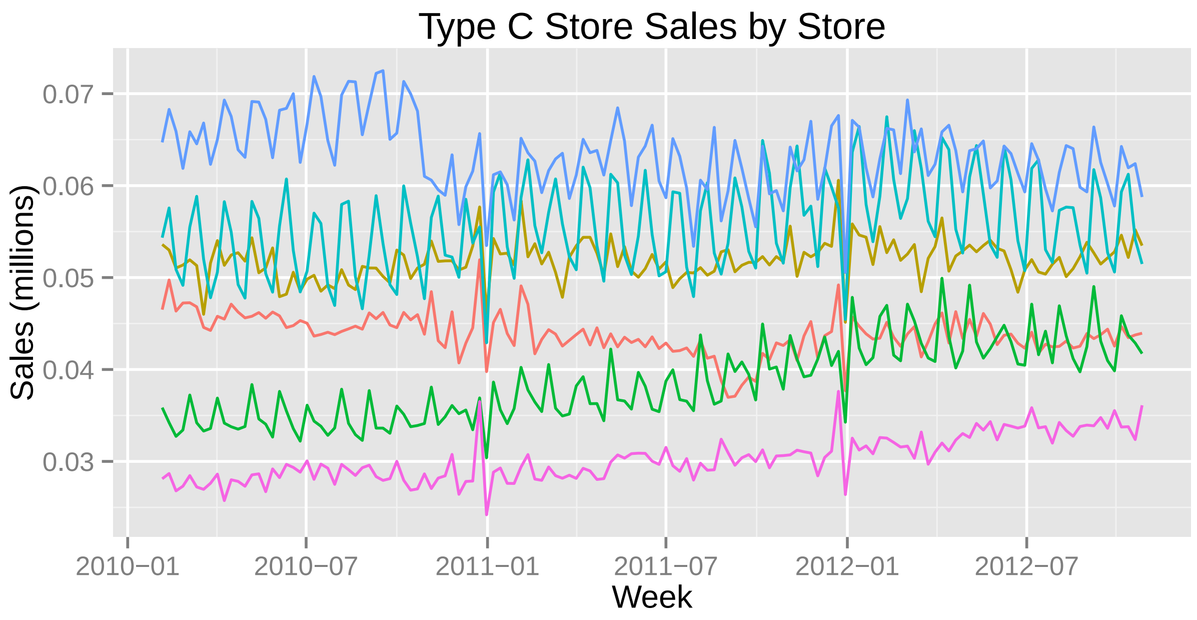
```
# by store (by type)
par(mfrow=c(2,2))
for (store.type in c('A', 'B', 'C')){
  sales.data <- get.aggr.sales(store.level, c('Store', 'Date'), c('Store_Type', store.type))
  print(ggplot(data=sales.data, aes(x=Date, y=Sales_Millions, group=Store, color=Store)) + geom_line() +
}
```



Type A Store Sales by Store
Sales (millions)
0.1
0.2
0.3
2010−01
2010−07
2011−01
2011−07
2012−01
2012−07
Week



Type B Store Sales by Store
Sales (millions)
0.1
0.2
0.3
2010−01
2010−07
2011−01
2011−07
2012−01
2012−07
Week



formula <- Norm_Weekly_Sales ~ Norm_Unemployment + IsHoliday*Store_Type + Store_Size

TODO: incorporate dates into model - month indicators, lag variables, autoregressive varaibles, trend variables

TODO: linear regression model; fit a model per store? note that a multi-level model is probably the right answer here, but we probably don't want to go to that level of complexity

TODO: stepwise selection to find most significant predictors

TODO: tree-based models

TODO: KNN

TODO: compare results from different model types