Stat E-139 Project Report

Stephen Camera-Murray, Desiree Koh,
Jennifer Le Hégaret, Elizabeth Wilson-Milne

December 21, 2015

Contents

Overview

Rossmann is a German drugstore chain which operates over 3,000 drug stores in 7 European countries. In order to schedule staff effectively, store managers try to predict their daily sales for six weeks in advance. Rossmann decided to create a Kaggle competition to find a satisfactory corporate-wide prediction model.

For our project, our team joined this contest and created a model based on the historical data provided by Rossmann. We used R for data exploration and manipulation, to create linear regression models, and to validate our results.

Because the Rossmann data violated many of the assumptions that linear regression models are based upon, including the assumption of independence (as each store provided many, correlated data points), we explored a more advanced "mixed effects" modelling technique and rigorously transformed much of the data.

In the end, we achieved a satisfactory model which uses a relatively straightforward formula that relies on easily obtainable data to predict future sales. As you will see, the main drivers of variability in sales from day-to-day are

- The historic level of sales at that particular store
- The time of year
- The day of the week, and
- Whether or not the store is running a promotion

We are confident that Rossmann managers would be able to use such a model, and that it would be of some benefit to them.

The Data

The Rossmann data consisted of two files which, together, provided information about 942 days of sales at 1,115 stores. The first file contained store-level data such as:

- Type of store (in terms of its "model" and its overall selection)
- Information about the closest competitor (distance and longetivity)
- Information about that store's particular promotion cycle

The second file contained daily data for each store for 2013, 2014, and the first part of 2015. This data consisted of items such as:

- Date (also noting that date's exact weekday, holiday status, and school holiday status)
- Whether or not that particular store was open that particular day
- Whether or not the store was running a promotion that day
- Sales amount and number of customers

Our first step was to combine the two datasets into one flat file. We also converted certain variables to strings and factors to prevent R from treating them as numerics. Then, after using all of the data to derive all additional variables, we removed the rows of "closed day" data. The stores consistently posted zero sales when closed, and our model is to predict sales when open.

Derived Data

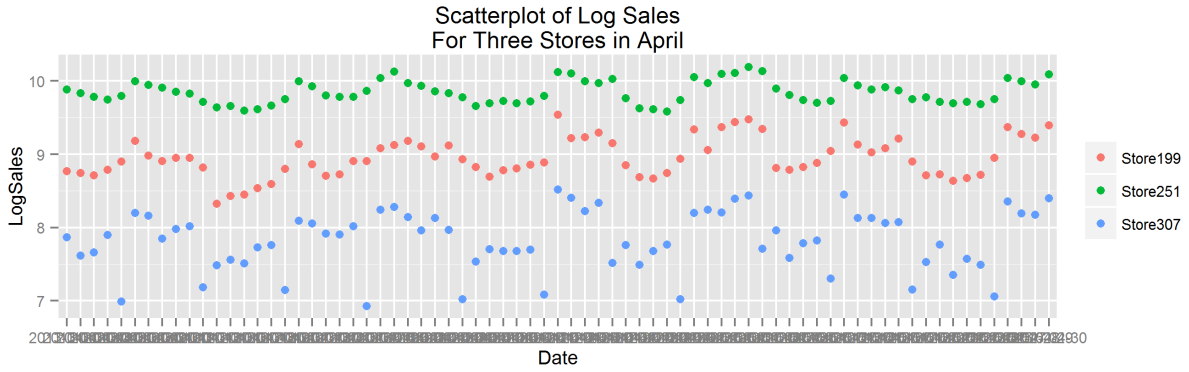
While the Rossmann-provided data included a huge number of records (over one million!), we saw the opportunity to add several interesting columns. For example:

- We converted the rather esoterically-delivered promotion information to determine on exactly which days each store ran exactly how many promotions.
- We analyzed the distribution of distance between stores and their nearest competitors, and derived a *Close Competitor* variable for those with the 10% nearest Competitors.
- We analyzed which stores had summer sales that were at least 150% the level of winter sales, and hypothesized that these might be seasonal, coastally-located stores deserving a special indicator variable we called *Summer Boost*.
- We noticed that some stores were completely closed for long stretches of time within our dataset, and hypothesized that perhaps, after a lengthy renovation, sales would increase due to a "buzz". However, we made our *Reopened* dummy variable even more subtle than that - it applied to a certain number of days after a store reopened, proportional to the amount of time a store had been closed. Therefore, whether it was due to a huge renovation or just a long weekend which forced loyal clients to wait to restock their homes, *Reopened* would capture both possible boosts to sales.
- For the years of data after the first year of data, we added a year-to-year *Prior Daily Average* variable that was a backwards look at each particular store's average daily sales in the prior calendar year.
- Most significantly, we analyzed the dates within our dataset and added several dummy and factor-level variables to distinguish some dates from others. We combined the school calendar information with basic knowledge of the seasons to create a rather specific *Season* variable - some periods of time fall under relatively generic terms such as "Spring", or "Fall Break", while other dates are assigned a factor-level as specific as "Christmas Eve".

(Please see the appendix for the complete list of all variables, both original and derived.)

Checking the Independence Assumption

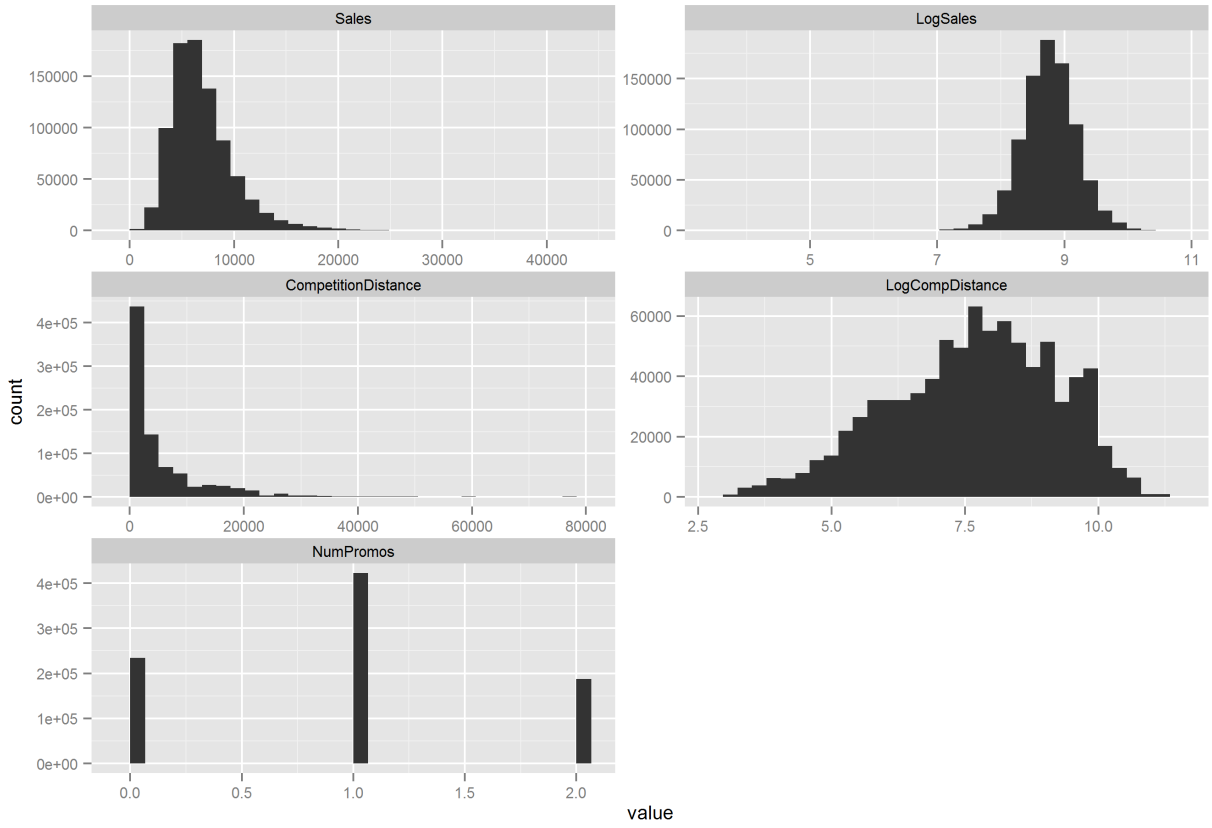
As previously mentioned, Rossmann's data observations cover 942 days of sales for 1,115 stores. As a result, many of the data points are clustered by store:



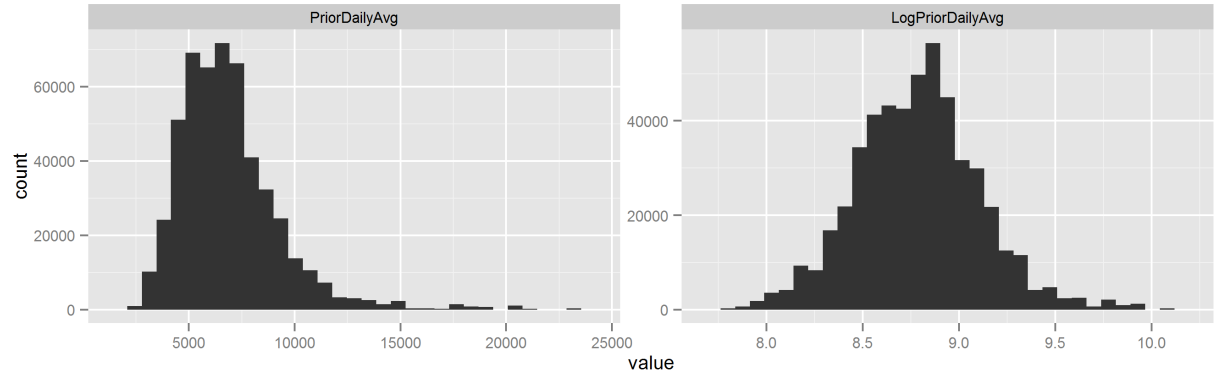
Any linear model not taking this into account would have residuals that were not independent and so be invalid. Therefore, we expanded our toolset to include a "mixed effects" model, as provided by the **lmer** command in R's **lme4** package. Unlike R's more basic **lm** linear model approach, this model allows the regression intercept for each store to vary by implicitly adding an indicator variable for each store, thereby anchoring sales predictions around each store's mean. In fact, it is even possible, with enough processing power, for a mixed effects model to allow the slopes of certain variables to vary by store as well.

Checking the Normality Assumption

Our dataset contained remarkably few quantitative variables - most of the information was about various categorizations. However, those few quantitative variables still required some transformations so that our statistical tools could be accurate. *Sales*, the response variable, and *Distance from Competition* were both right-skewed until a log transformation was applied. However, the total *Number of Promotions* offered by a store at any one time was originally reasonably symmetric.

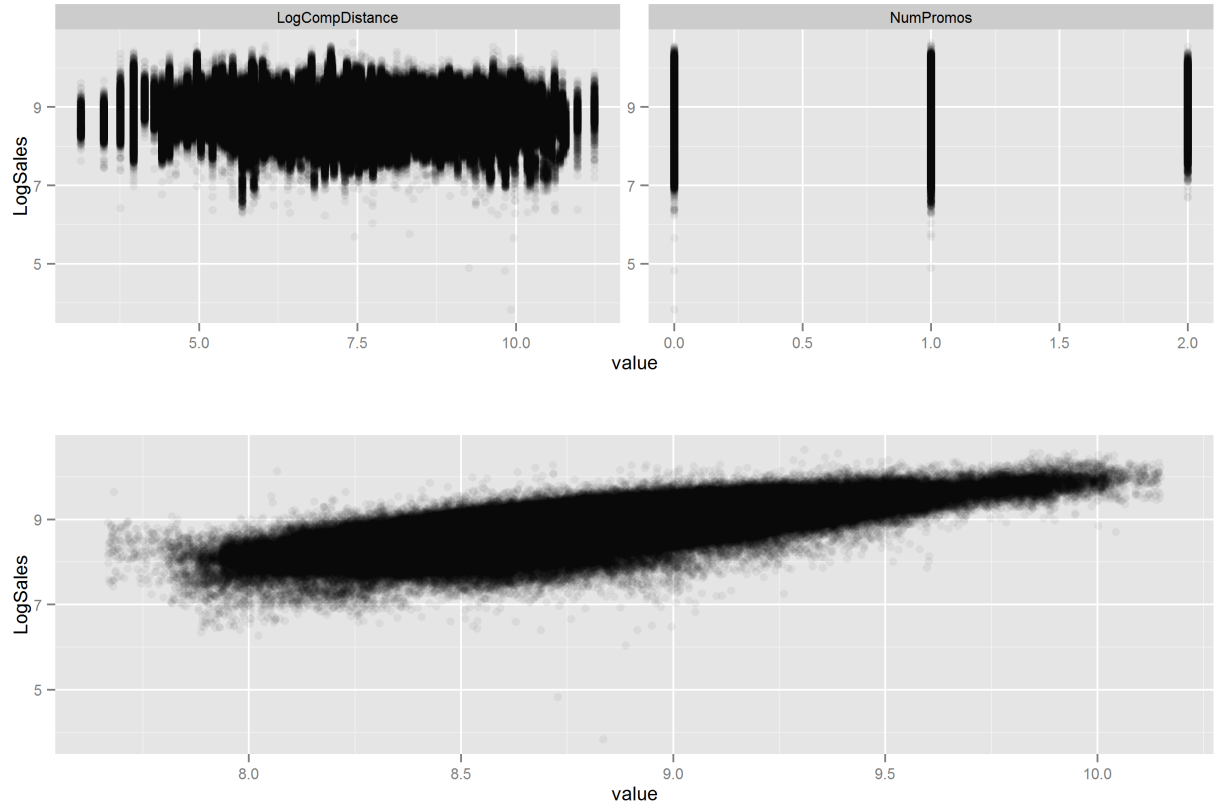


As mentioned before, our *PriorDailyAverage* variable applies only for the second year on of our data, as the first year's data becomes summary information. Therefore, we looked at that subset of data to evaluate this variable's normality, and again decided to perform a log transformation:



Checking the Linearity Assumption

Next, we looked at scatterplots of our response variable, *LogSales*, against each of our quantitative predictor variables: the log of the distance to the closest competitor, the number of promotions at any one time, and the log of the prior year's average daily sales. We did not see any evidence of nonlinearity:



Imagining Interaction Terms

Even though we are not experts in the drugstore business, we knew we needed several interaction variables to allow for different slopes for certain variables under certain conditions. For example, we needed to allow the summer season to have a different magnitude of effect if a store seemed to be a seasonal summer destination, compared to all of the other stores in Germany. We needed to ensure that the distance from the closest competitor was only considered if that competitor was already open for business.

Although we imagined many more interaction terms, we unfortunately ran into the limits of our processing power rather quickly with these models. As a result, we made do with the two mentioned above.

Creating the Model

Process

Once we had all of our variables in place, we created a simple **baseline model** that used only prior sales as a predictor. Then, we created a **final model** which incorporated contributed variables and interaction terms. Lastly, we compared the two in terms in terms of prediction and reliability.

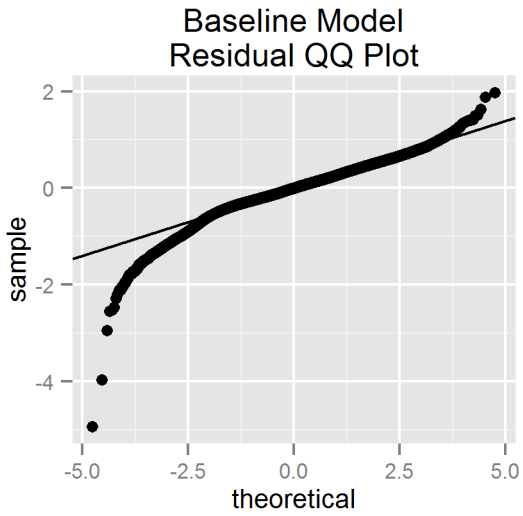
Baseline Model

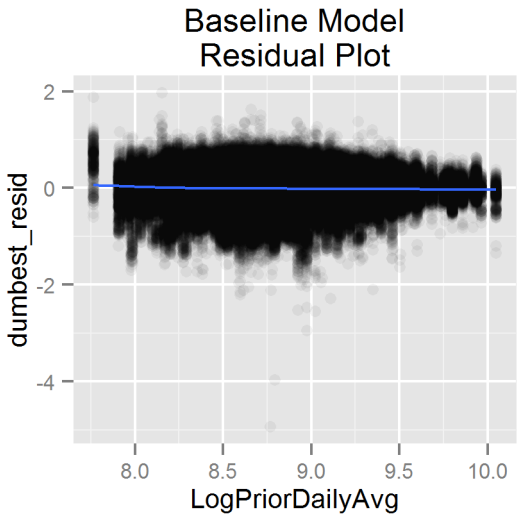
The simplest baseline model we could imagine is that each store continues to have sales at the same rate as in the prior year, as based on the prior year's daily average. Given our data, of course, we deal with log values for these variables:

$$LogSales = LogPriorDailyAverage_i$$

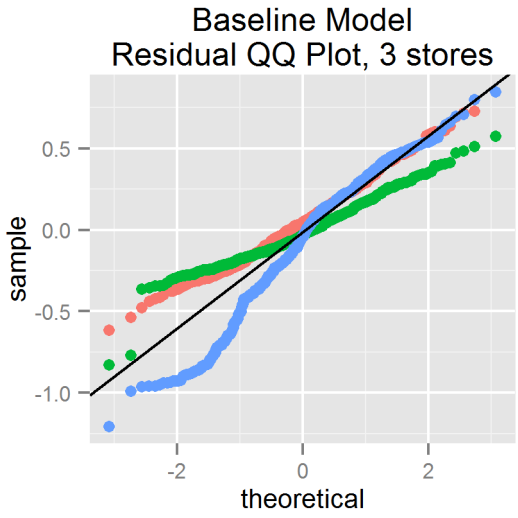
When we run this model against our data for (only) the years 2014 and 2015, we find that the standard error of the residuals is 0.2789 and our residuals range from -4.9384 to 1.9728.

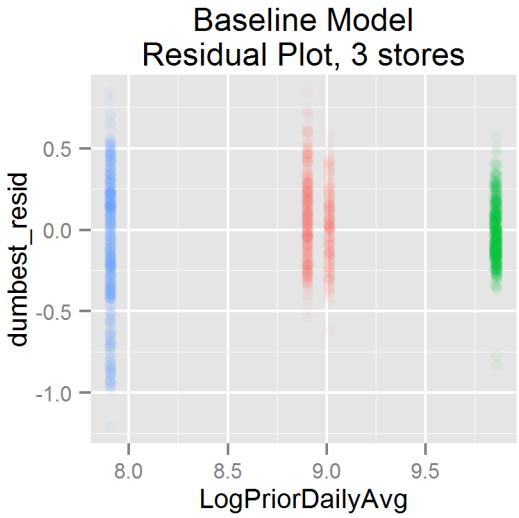
If our model is valid, then we should observe normality of the residuals, constant variance of the residuals, and linearity between the residuals and the fitted values, and these assumptions should hold true both overall and at the store-specific level.





The above plots show our overall residuals for the entire model. The below plots highlight store-specific residuals for three representative stores:





Overall, our residuals are reasonably close to the normal distribution, even though the tails are thicker than is ideal. Looking at the residual plot, the residuals show constant variance within a nicely even random cloud formation. As the cloud follows a perfectly flat line, we see no evidence of non-linearity.

The QQ Plot of the small subset for three specific stores shows that each store has its own particular curve around the normality line, but, even individually, each curve is still quite close to normal. The Residual Plot is interesting: each store does seem to have its own level of variance around the mean, and one store seems to tend to be underpredicted while another store tends to be overpredicted.

These assumption checks tell us that this model is a reasonably valid one.

Final Model

To create our mixed effects model, we needed to identify the most relevant variables that drive sales. Due to our limited processing power, we did this by first running a simple **lm** linear model. Within this model, we included the average prior daily average (as used in our baseline model) and many other variables:

- Factor variables for the year, the day of the week, and our extensively-specific *Season* variable,
- Factor variables about the type of store and the type of selection of goods it offers,
- Indicator variables for summer seasonality, stores that were regularly open on Sundays, stores that had recently reopened after a relatively lengthy period of time,
- Indicator and quantitative variables about how many promotions the store ran each day, and the type of promotion, and
- An interaction term about the existence of competition combined with the distance from that competition, and an indicator term about whether or not this store was "extra close" to its competition.

Reviewing the basic linear model's reported p-values helped us to weed out a few promotion-oriented variables as well as *Reopened*, as we kept only variables for which the p-value was multiplied by a factor of $\frac{1}{10^{16}}$ or less. *(For the complete coefficients table for this model, please see the appendix.)*

The code for our final model was thus

```
final <-  lmer ( LogSales ~ YearFactor + Season + DayOfWeek_Named + Interact_SummerBoost +
                    SundayOpen + StoreType + Assortment + Promo + Interact_Competitor +
                    CloseCompetitor + LogPriorDailyAvg + (1|StoreFactor), data = full )
```

which results in the following equation for expected sales on days when a store is open:

Log Sales = 5.02 + *Store Specific Value* + .43 *Log Prior Avg Sales* - .10 *Open Sunday* + .05 *Close Competitor* - .01 *Competitor Distance* + .43 *Summer Boost* + .34 *Promo* + .01 $Year_{2015}$ + ...

Store Type	*Assortment*	*Day of Week*	*Season*
a: 0.00	a: 0.00	Monday: 0.11	Easter Holiday: 0.28
b: 0.44	b: -0.09	Tuesday: -0.01	Easter Break: -0.07
c: 0.01	c: 0.09	Wednesday: -0.06	Spring: -0.15
d: 0.02		Thursday: -0.06	Summer: -0.14
		Friday: 0.00	Summer Break: -0.17
		Saturday: -0.05	Fall: -0.15
		Sunday: -0.11	Fall Break: -0.20
			Holiday Shopping: 0.05
			Christmas Break: 0.00
			Last Minute Holiday Shopping: 0.54
			Christmas Eve: -0.27
			New Year's Eve: -0.45
			Public Holiday: -0.21

(On days when the store is closed, expected sales are zero.)

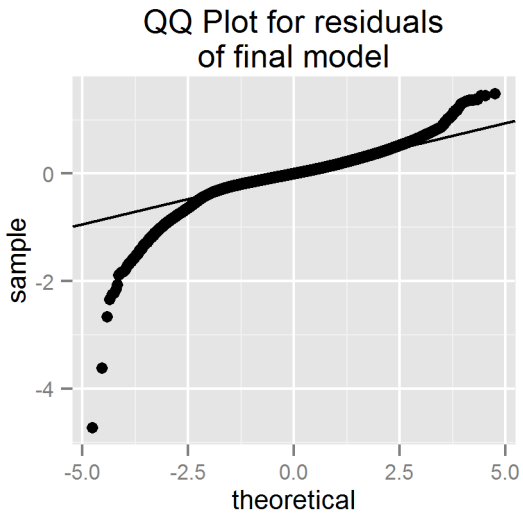
These coefficients make intuitive sense. Some specific observations:

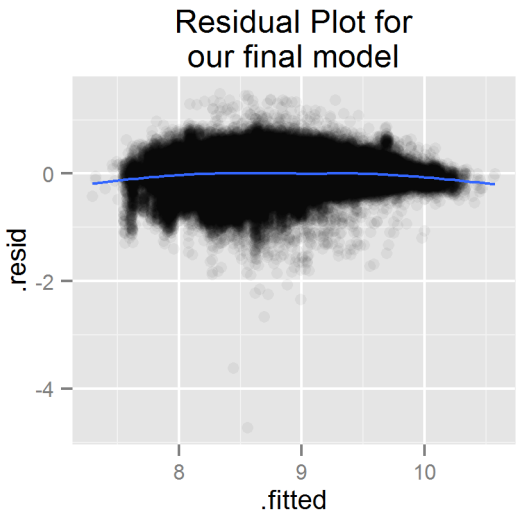
- It is interesting to see what a huge magnitude of an effect solely the day of the week and the season have on a store's sales - competition, the assortment of goods the store offers, and the type of store it is (most of the time) hardly matter in comparison.
- December and the resulting school vacation, apart from New Year's Eve, are extremely huge drivers of the entire year's sales. Other than that, only the actual observed Easter holiday (and not the school's Easter-time break) manages to tax the store's capacity.
- Shoppers do respond to the store's primary promotion!
- Being open on Sundays does not benefit the bottom line by much - if a store is open on Sundays, all of its other days are slightly depressed as an effect, and, on top of that, the Sunday itself is relatively quiet.
- Our hypothesis that close competition might actually boost sales holds - we had posited that clusters of similar stores can attract more shoppers, and we see that having a close competitor does indeed boost sales. The coefficient on *Competitor Distance* is slightly ambiguous - do sales go down the farther away the competitor is, or do sales decline because this store does actually have a competitor?

(For the complete coefficients table for this model, please see the appendix.)

Assumptions Check

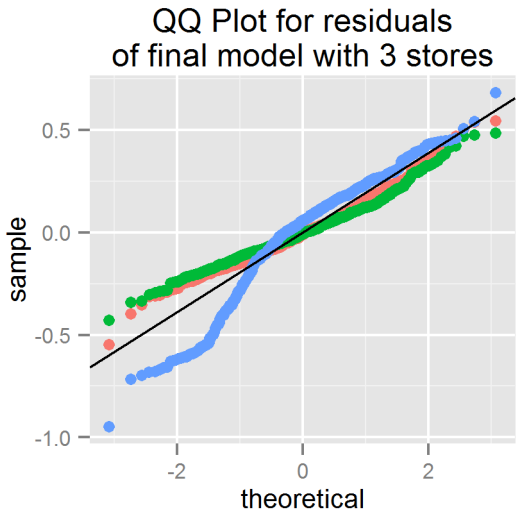
First, we will check the overall residuals of the model as a whole:

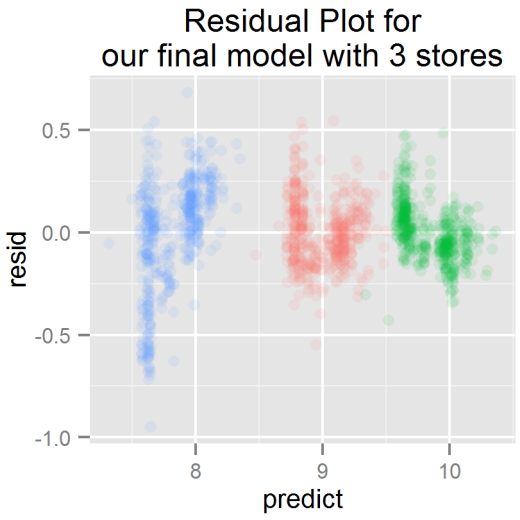




Unfortunately, even though the QQ Plot confirms that a huge proportion of residuals are normally distributed, it also shows us that we still have relatively thick tails, including three clear outliers. We also see a suspicious curve within the Residual Plot - while the overall shape is a cloud that sufficiently indicates equal variance, and the line is quite close to zero, there is a slight arc to it that indicates nonlinearity. We are not confident that the necessary assumptions are sufficiently met.

Checking the same three stores we had used before as a representative sample, we see a suspicious pattern:

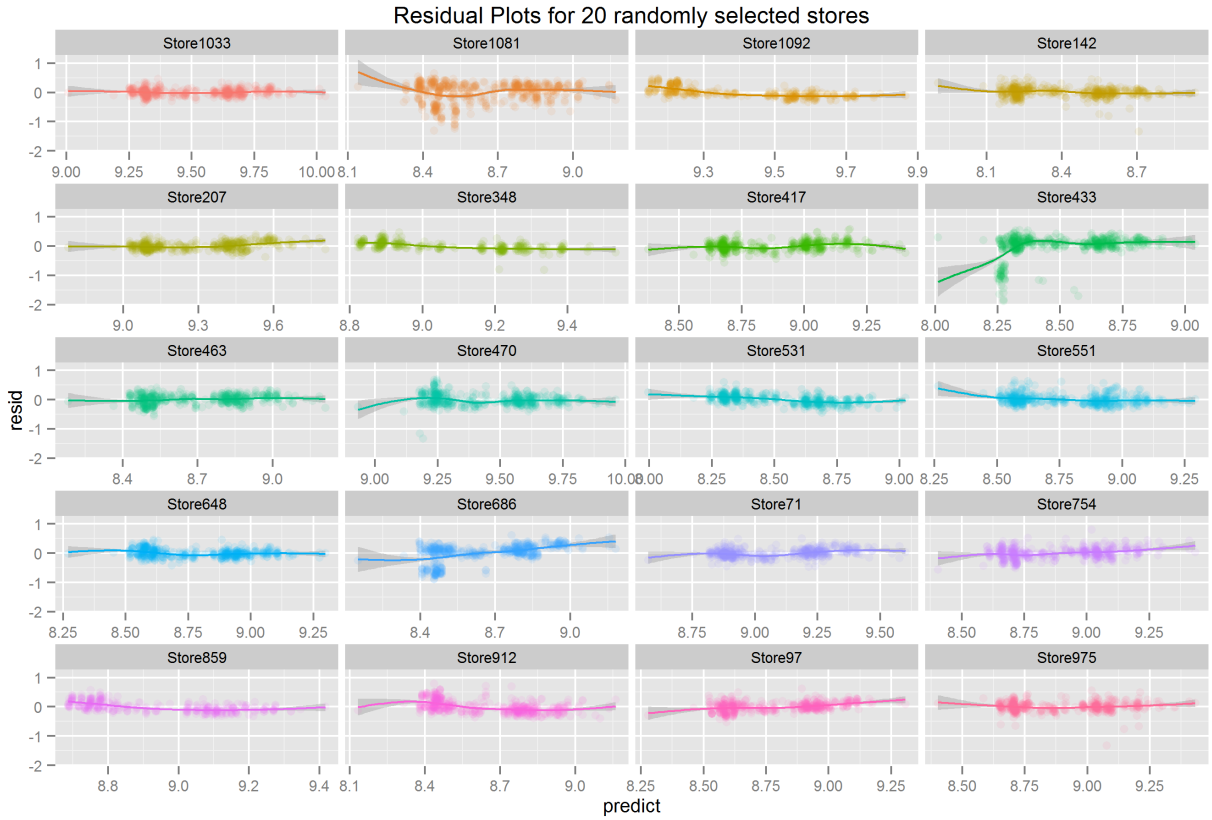




While the residuals seem to have somewhat improved normality, as compared to the baseline model, they seem highly nonlinear, as they have a per-store "U-Shape". What lies behind this? Remembering our regression model, there are many indicator values contained within it, and sales vary dramatically depending on the day. Perhaps this is why predicted values per store seem grouped at two different discrete amounts, with not as many predictions in the middle.

What about the variance? For these three stores, it seems that the variance of the lower predictions is usually greater than the variance of higher predictions, which is a dangerous indication of possibly unequal variance.

Are these issues common for all stores? We randomly sampled twenty more stores to see:



Here, we see that every store is indeed different when it comes to the distribution of its residuals and that, overall, the stores' residuals suggest a reasonably even variance around zero.

Taking the two views together, we conclude that the assumptions of linear regression hold reasonably well for the great majority of stores and days within our data set. However, we also see that a fuller mixed effects model may be truly useful - some of the stores' density lines do have slight slopes up or down, so that different store-specific slopes for at least some of the variables in our model might be appropriate. Also, perhaps more store-specificity would address the outliers we see in these graphs.

Performance as Compared to Baseline

We compared our final model to our baseline in a number of ways:

Test	Baseline	Final
Residual Standard Error	.279	.189
Adjusted R-Squared	.558	.798
AIC	-1293257	-1689237

Our mixed effects model consistently outperformed the baseline model.

Summary

In the end, we feel we achieved a satisfactory model, given our processing power and time constraints. As-is, this model explains about 80% of the variability presented in the data, and it does so using very straight-forward, easily-obtainable inputs:

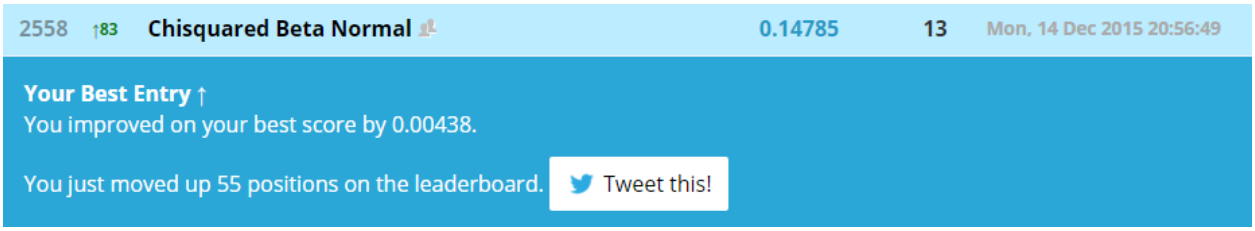
- The historic level of sales at that particular store
- The time of year
- The day of the week, and
- Whether or not the store is running a promotion

We do see, however, how much better our model could be. There are many clear opportunities for improvement:

- With better processing power, we could create store-specific coefficients for each of the predictor variables
- With better processing power, we could explore more two-way interactions (and maybe even three!) between our predictor variables
- With more time, we could explore additional outliers to understand what may lie behind them. For example, we noticed consistently large residuals for December 23rd and 24th - the December 23rd frenzy was underpredicted, while the half-day December 24th was overpredicted. As a result, we added those specific days to our *Season* factor variable, and they proved highly significant. We are sure that there are other robust patterns we could identify, given time (and perhaps some industry experience), and accounting for those patterns may help further normalize our residual errors.

Still, we learned a lot, gaining experience with data wrangling, R code, assumptions checking, interpreting QQ plots, and expanding our statistical toolkit even further via mixed effects modelling. We look forward to applying these techniques to future projects.

A Fun Aside: The Kaggle Process



The Rossmann Store Sales Kaggle competition closed on Dec. 14 with around 3,300 teams worldwide. Our team entered the competition when it was around two thirds done. At this point, the top teams had their models complete and were making slight adjustments to improve their predictions. Scoring is based on root mean square percentage error (RMSPE) on an unknown portion of a test dataset. The upper half of the field was fairly crowded with a RMSPRE range of around 0.13 to 0.09.

Before we learned about linear mixed models, our first submission was a straightforward multiple linear regression with transformed quantitative variables. It scored below the Kaggle Median DayOfWeek Benchmark. At this point, we knew something was wrong. Learning about the linear mixed model boosted our ranking dramatically. After we added several derived variables and interaction terms, our final submission attained an RMSPRE of 0.1479, raising our ranking hundreds of positions to the 75% percentile. Judging by the publicly available scripts, most of the teams still ranking above us used random forest models rather than linear regression.

Appendix

Variables

Original Variables Provided Within the Store File

Name	Description
Store	Unique ID for each store
StoreType	Differentiates between 4 different store models: a, b, c, d
Assortment	Describes an assortment level: a = basic, b = extra, c = extended
CompetitionDistance	Distance in meters to nearest competitor store
CompetitionOpenSince[Mon/Yr]	Approximate year and month of time nearest competitor was opened
Promo2	Special Promotion: 0 = store is not participating, 1 = store is participating
Promo2Since[Year/Week]	Year and calendar week when store started participating in Promo2
PromoInterval	When Promo2 is started, e.g. "Feb,May,Aug,Nov"

Original Variables Provided Within the Daily Sales File

Name	Description
Store	Unique ID for each store
DayOfWeek	Day of week for historical store sales, 1-7 (1 = Monday)
Date	Date for historical store sales
Sales	Sales for given day (original target prediction)
Open	Dummy variable
Promo	Dummy variable about basic promotion offer
StateHoliday	a = public holiday, b = Easter holiday, c = Christmas, 0 = None
SchoolHoliday	Dummy variable

Derived Data

Name	Description
LogSales	Sales transformed
LogCompDistance	CompetitionDistance transformed
PriorSales	Total sales for same store, same month in previous year
LogPriorSales	PriorSales transformed
DailyAvg_2013	Store's average daily sales, calendar year of 2013
DailyAvg_2014	Store's average daily sales, calendar year of 2014
LogDailyAvg_2013	Log transformed

Name	Description
LogDailyAvg_2014	Log transformed
PriorDailyAvg	Correct average for the store on that day depending on year
LogPriorDailyAvg	Log of correct average
AvgSummerSales	Average monthly sales level during summer
AvgWinterSales	Average monthly sales level during winter
SeasonSalesRatio	Ratio of average monthly summer to average monthly winter sales.
NumPromos	Total number of promos running on sales date in that store
StoreFactor	Store ID converted to a factor
YearFactor	Sales year converted to a factor
Month	Sales month converted to a factor
Season	Factor specifying 14 different time periods within the year
DayOfWeek_Named	DayOfWeek converted to a factor
MonFri	Dummy variable
StateHolidayDummy	Dummy variable
SummerMonthDummy	Dummy variable
WinterMonthDummy	Dummy variable
SummerBoost	Dummy variable if store has high sales in summer
SundayOpen	Dummy variable
SundayClosed	Dummy variable
Reopened	Dummy variable, store just reopened after lengthy closure.
CompetitionOpen	Dummy variable
CompetitionNONE	Dummy variable
CloseCompetitor	Dummy variable for smallest 10% of distances between stores

Coefficient Table, LM Model

```
##
## Call:
## lm(formula = full)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.711 -0.108 -0.001  0.112  1.692
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   2.82e-01   8.21e-03   34.35  < 2e-16 ***
## YearFactor2015               -4.57e-03   6.21e-04   -7.36  1.8e-13 ***
## SeasonChristmas_Eve          -2.73e-01   6.81e-03  -40.17  < 2e-16 ***
## SeasonEaster_Break           -6.62e-02   2.47e-03  -26.77  < 2e-16 ***
## SeasonEaster_Holiday          2.99e-01   9.88e-02    3.02   0.0025 **
## SeasonFall                   -1.58e-01   2.17e-03  -72.67  < 2e-16 ***
## SeasonFall_Break             -2.08e-01   2.94e-03  -70.78  < 2e-16 ***
## SeasonHoliday_Shopping        5.02e-02   2.51e-03   19.99  < 2e-16 ***
## SeasonHolidayShopping_LastMin 5.31e-01   6.79e-03   78.10  < 2e-16 ***
## SeasonNYE                    -4.55e-01   6.81e-03  -66.84  < 2e-16 ***
## SeasonPublic_Holiday         -2.21e-01   1.02e-02  -21.68  < 2e-16 ***
## SeasonSpring                 -1.46e-01   2.08e-03  -70.49  < 2e-16 ***
## SeasonSummer                 -1.41e-01   2.11e-03  -66.61  < 2e-16 ***
## SeasonSummer_Break           -1.73e-01   2.22e-03  -77.79  < 2e-16 ***
## SeasonWinter                 -1.99e-01   2.10e-03  -94.54  < 2e-16 ***
## DayOfWeek_NamedMon            1.07e-01   9.75e-04  109.44  < 2e-16 ***
## DayOfWeek_NamedSat           -5.08e-02   1.03e-03  -49.54  < 2e-16 ***
## DayOfWeek_NamedSun           -1.14e-01   4.57e-03  -24.94  < 2e-16 ***
## DayOfWeek_NamedThu           -6.05e-02   9.78e-04  -61.85  < 2e-16 ***
## DayOfWeek_NamedTue           -1.39e-02   9.66e-04  -14.37  < 2e-16 ***
## DayOfWeek_NamedWed           -5.65e-02   9.69e-04  -58.28  < 2e-16 ***
## Interact_SummerBoost          3.34e-01   8.33e-03   40.05  < 2e-16 ***
## SundayOpenTRUE               -5.33e-02   2.30e-03  -23.17  < 2e-16 ***
## Reopened                     -1.12e-02   1.32e-02   -0.85   0.3979
## StoreTypeb                    1.22e-01   3.73e-03   32.66  < 2e-16 ***
## StoreTypec                    8.59e-03   8.49e-04   10.12  < 2e-16 ***
## StoreTyped                    2.10e-02   6.63e-04   31.67  < 2e-16 ***
## Assortmentb                   1.12e-01   4.05e-03   27.73  < 2e-16 ***
## Assortmentc                   3.65e-02   5.91e-04   61.76  < 2e-16 ***
## Promo                         3.31e-01   8.86e-04  373.32  < 2e-16 ***
## Promo2Active                 -6.69e-03   8.46e-04   -7.91  2.5e-15 ***
## NumPromos                     4.54e-03   6.35e-04    7.15  8.7e-13 ***
## Interact_Competitor          -1.58e-03   7.35e-05  -21.50  < 2e-16 ***
## CloseCompetitorTRUE          -1.84e-02   9.68e-04  -19.05  < 2e-16 ***
## LogPriorDailyAvg              9.66e-01   9.05e-04 1067.21  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.197 on 506379 degrees of freedom
## Multiple R-squared:  0.778,  Adjusted R-squared:  0.778
## F-statistic: 5.23e+04 on 34 and 506379 DF,  p-value: <2e-16
```

Coefficient Table, Mixed Effects Model

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## LogSales ~ YearFactor + Season + DayOfWeek_Named + Interact_SummerBoost +
##     SundayOpen + StoreType + Assortment + Promo + Interact_Competitor +
##     CloseCompetitor + LogPriorDailyAvg + (1 | StoreFactor)
##    Data: full
##
## REML criterion at convergence: -245134
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -25.064  -0.560  -0.020   0.565   7.890
##
## Random effects:
##  Groups      Name        Variance Std.Dev.
##  StoreFactor (Intercept) 0.0322   0.180
##  Residual                0.0356   0.189
## Number of obs: 506414, groups:  StoreFactor, 1115
##
## Fixed effects:
##                                Estimate Std. Error t value
## (Intercept)                    5.022424   0.062890      80
## YearFactor2015                 0.013807   0.000642      21
## SeasonChristmas_Eve           -0.268753   0.006503     -41
## SeasonEaster_Break            -0.070351   0.002366     -30
## SeasonEaster_Holiday           0.277070   0.094675       3
## SeasonFall                    -0.153724   0.002079     -74
## SeasonFall_Break              -0.203615   0.002809     -72
## SeasonHoliday_Shopping         0.054750   0.002400      23
## SeasonHolidayShopping_LastMin  0.535134   0.006491      82
## SeasonNYE                     -0.450309   0.006503     -69
## SeasonPublic_Holiday          -0.210110   0.009762     -22
## SeasonSpring                  -0.147198   0.001986     -74
## SeasonSummer                  -0.142050   0.002021     -70
## SeasonSummer_Break            -0.169496   0.002127     -80
## SeasonWinter                  -0.200597   0.002011    -100
## DayOfWeek_NamedMon             0.106931   0.000931     115
## DayOfWeek_NamedSat            -0.050362   0.000979     -51
## DayOfWeek_NamedSun            -0.114430   0.004365     -26
## DayOfWeek_NamedThu            -0.060355   0.000935     -65
## DayOfWeek_NamedTue            -0.013627   0.000922     -15
## DayOfWeek_NamedWed            -0.056351   0.000926     -61
## Interact_SummerBoost           0.430664   0.009035      48
## SundayOpenTRUE                -0.095195   0.045775      -2
## StoreTypeb                     0.441600   0.078483       6
## StoreTypec                     0.008443   0.016580       1
## StoreTyped                     0.017464   0.012702       1
## Assortmentb                   -0.087628   0.087406      -1
## Assortmentc                    0.089002   0.011308       8
## Promo                          0.335284   0.000593     566
## Interact_Competitor           -0.008331   0.000272     -31
## CloseCompetitorTRUE            0.052187   0.018490       3
```

```
## LogPriorDailyAvg               0.425436   0.007141      60

##
## Correlation matrix not shown by default, as p = 32 > 20.
## Use print(x, correlation=TRUE)  or
##   vcov(x)     if you need it
```

Code

The following module contains the code we used to transform our data:

```
##########################################

# This module provides the functions we need to
#    obtain our data,
#    transform some variables
#    add derived data
#    and create a new data source file

getFullData <- function(pwd)
{
    # Read the files
    print("Reading files")
    dataStores <- read.csv(paste(pwd, "store.csv", sep=""), header=T)
    dataSales <- read.csv(paste(pwd, "train.csv", sep=""), header=T)

    # Mark which dates are reopening dates before we delete the zero data
    print("Adding reopened data")
    dataSales$Reopened <- addReopenedFlag(dataSales)

    # Only keep rows for open stores with positive sales
    dataSales <- dataSales [ dataSales$Sales > 0 & dataSales$Open != 0, ]

    # Add store-level derived data
    print("Adding store-level derived data")
    dataStores <- addStoreDerived(dataStores, dataSales)

    # Merge the data
    print("Merging the datasets")
    dataTraining <- merge ( dataSales, dataStores, by = "Store" )

    # add date-level derived data
    print("Adding date-level derived data")
    dataTraining <- addDateDerived(dataTraining, pwd)
    dataTraining$LogSales <- log(dataTraining$Sales) # we already removed 0 values

    # and more
    dataTraining$ID <- paste(dataTraining$Date, dataTraining$StoreFactor, sep="") # for graphs
    dataTraining$PriorDailyAvg <- NA
    dataTraining$PriorDailyAvg[dataTraining$YearFactor == 2014]
        <- dataTraining$DailyAvg_2013[dataTraining$YearFactor == 2014]
    dataTraining$PriorDailyAvg[dataTraining$YearFactor == 2015]
        <- dataTraining$DailyAvg_2014[dataTraining$YearFactor == 2015]
    dataTraining$LogPriorDailyAvg <- NA
    dataTraining$LogPriorDailyAvg[dataTraining$YearFactor == 2014]
        <- dataTraining$LogDailyAvg_2013[dataTraining$YearFactor == 2014]
    dataTraining$LogPriorDailyAvg[dataTraining$YearFactor == 2015]
        <- dataTraining$LogDailyAvg_2014[dataTraining$YearFactor == 2015]

    # export new files by given names
```

```
    print("Exporting full Stores data")
    write.csv(dataStores, paste(pwd, "dataStores_full.csv", sep=""))

    print("Exporting full Training data")
    write.csv(dataTraining, paste(pwd, "dataTraining_full.csv", sep=""))

    print("Done!")
}

addStoreDerived <- function(dataStores, dataSales)
{
    # fix Store factor
    dataStores$StoreFactor <- paste("Store", as.factor(dataStores$Store), sep="")

    # A little data supplementation, to override NA values of competitor distance with the average
    dataStores$CompetitionDistance[dataStores$Store %in% c(291, 622, 879)] <- 5458

    # Mark if the store is open on Sundays
    dataSales$OpenSundayCheck <- dataSales$Open == 1 & dataSales$DayOfWeek == 7
    dataStores$SundayOpen
        <- dataStores$Store %in% unique(dataSales$Store[dataSales$OpenSundayCheck == 1])
    dataStores$SundayClosed <- abs(dataStores$SundayOpen - 1)

    # Mark which stores are in the closest 10% to their competitors
    dataStores$CloseCompetitor
        <- dataStores$CompetitionDistance < quantile(dataStores$CompetitionDistance, c(.1),
                                                     na.rm = TRUE)

    # Average sales by store
    dataStores$DailyAvg_2013 = tapply ( dataSales$Sales[year(as.Date(dataSales$Date)) == 2013],
            dataSales$Store[year(as.Date(dataSales$Date)) == 2013], FUN = mean )
    dataStores$LogDailyAvg_2013 <- log( dataStores$DailyAvg_2013 + 3)
    dataStores$DailyAvg_2014 = tapply ( dataSales$Sales[year(as.Date(dataSales$Date)) == 2014],
            dataSales$Store[year(as.Date(dataSales$Date)) == 2014], FUN = mean )
    dataStores$LogDailyAvg_2014 <- log( dataStores$DailyAvg_2014 + 3)

    # Determine which stores are seasonal stores (high summer sales)

    # Average summer/winter sales by store
    dataSales$month = month(as.Date(dataSales$Date))
    dataSales$SummerMonthDummy <- (dataSales$month== 6)|(dataSales$month==7)|(dataSales$month==8)
    dataSales$WinterMonthDummy <- (dataSales$month==12)|(dataSales$month==1)|(dataSales$month==2)
    dataStores$AvgSummerSales = tapply ( dataSales$Sales[dataSales$SummerMonthDummy==1],
                                         dataSales$Store[dataSales$SummerMonthDummy==1], FUN = mean )
    dataStores$AvgWinterSales = tapply ( dataSales$Sales[dataSales$WinterMonthDummy==1],
                                         dataSales$Store[dataSales$WinterMonthDummy==1], FUN = mean )
    dataStores$SeasonSalesRatio <- dataStores$AvgSummerSales/dataStores$AvgWinterSales
    dataStores$SummerBoost <- (dataStores$SeasonSalesRatio > 1.5)

    return(dataStores)
}

addDateDerived <- function(dataTraining, pwd)
```

```
{
    print("Adding quick calculations")
    dataTraining$Month = months(as.Date(dataTraining$Date))
    dataTraining$YearFactor <- as.factor(format(as.Date(dataTraining$Date),'%Y'))
    dataTraining$DayOfWeekDummy <- as.character(dataTraining$DayOfWeek)
    dataTraining$DayOfWeek_Named <- format(as.Date(dataTraining$Date),"%a")
    dataTraining$MonFri <- dataTraining$DayOfWeek_Named == "Mon"
        | dataTraining$DayOfWeek_Named == "Fri"
    dataTraining$StateHolidayDummy <- dataTraining$StateHoliday != 0

    print("Figuring out prior sales")
    dataTraining <- addPriorSales(dataTraining, pwd)

    print("Adding competition info")
    dataTraining <- addCompetitionFlag(dataTraining) #$CompetitionOpen
    dataTraining$CompetitionNONE <- abs(dataTraining$CompetitionOpen - 1)
    dataTraining$CompetitionDistance <- dataTraining$CompetitionDistance + 3
    dataTraining$LogCompDistance <- log(dataTraining$CompetitionDistance)

    print("Adding promotion info")
    dataTraining <- addPromo2Flag(dataTraining) #$Promo2
    dataTraining$NumPromos <- dataTraining$Promo + dataTraining$Promo2

    print("Adding season data")
    dataTraining <- addSeason(dataTraining) # $Season

    return(dataTraining)
}

addCompetitionFlag <- function(dataTraining)
{
    dataTraining$CompetitionOpen
        <- as.integer ( as.Date ( paste ( dataTraining$CompetitionOpenSinceYear,
            dataTraining$CompetitionOpenSinceMonth, "1", sep = "-" ), "%Y-%m-%d" )
            <= as.Date ( dataTraining$Date ) )
    dataTraining$CompetitionOpen [ is.na ( dataTraining$CompetitionOpen ) ] <- 0

    return(dataTraining)
}

addPromo2Flag <- function(dataTraining)
{
    # Reorder factors for math to determine if we're in the promo2 month
    # Note: the factor order may be different in the test dataset

    dataTraining$PromoInterval <- factor ( dataTraining$PromoInterval,
                                           levels ( dataTraining$PromoInterval )[c(1,4,3,2)] )
    # Flag to determine eligibility for promo 2
    dataTraining$Promo2Active <- as.integer (
        ( ( as.integer ( format ( as.Date ( dataTraining$Date ), "%Y%U" ) ) >
                ( ( dataTraining$Promo2SinceYear * 100 ) + dataTraining$Promo2SinceWeek ) )
          & ( ( as.integer ( format ( as.Date ( dataTraining$Date ), "%m" ) ) %% 3 ) ==
                  ( as.integer ( dataTraining$PromoInterval ) - 2 ) ) )
```

```
    )
    return(dataTraining)
}
addSeason <- function(dataTraining)
{
    dataTraining$Season <- ""

    # specific holiday times
    dataTraining$Season[dataTraining$SchoolHoliday == 1 & (month(as.Date(dataTraining$Date)) == 12
            | month(as.Date(dataTraining$Date)) == 1)] <- "Christmas_Break"
    dataTraining$Season[dataTraining$SchoolHoliday == 1 & (month(as.Date(dataTraining$Date)) == 3
            | month(as.Date(dataTraining$Date)) == 4)] <- "Easter_Break"
    dataTraining$Season[dataTraining$SchoolHoliday == 1 & (month(as.Date(dataTraining$Date)) == 7
            | month(as.Date(dataTraining$Date)) == 8 | month(as.Date(dataTraining$Date)) == 9)]
       <- "Summer_Break"
    dataTraining$Season[dataTraining$SchoolHoliday == 1 & month(as.Date(dataTraining$Date)) == 10 ]
      <- "Fall_Break"

    # super-specific dates
    dataTraining[dataTraining$Date=="2013-12-23",]$Season="HolidayShopping_LastMin"
    dataTraining[dataTraining$Date=="2014-12-23",]$Season="HolidayShopping_LastMin"
    dataTraining[dataTraining$Date=="2013-12-24",]$Season="Christmas_Eve"
    dataTraining[dataTraining$Date=="2014-12-24",]$Season="Christmas_Eve"
    dataTraining[dataTraining$Date=="2013-12-31",]$Season="NYE"
    dataTraining[dataTraining$Date=="2014-12-31",]$Season="NYE"

    # state holidays
    dataTraining$Season[dataTraining$Season == "" & dataTraining$StateHoliday == "a"]
        <- "Public_Holiday"
    dataTraining$Season[dataTraining$Season == "" & dataTraining$StateHoliday == "b"]
        <- "Easter_Holiday"

    # filling in what's left
    dataTraining$Season[dataTraining$Season == "" & month(as.Date(dataTraining$Date)) == 12]
        <- "Holiday_Shopping"
    dataTraining$Season[dataTraining$Season == "" & (month(as.Date(dataTraining$Date)) == 1
        | month(as.Date(dataTraining$Date)) == 2)] <- "Winter"
    dataTraining$Season[dataTraining$Season == "" & (month(as.Date(dataTraining$Date)) == 3
         | month(as.Date(dataTraining$Date)) == 4 | month(as.Date(dataTraining$Date)) == 5)]
        <- "Spring"
    dataTraining$Season[dataTraining$Season == "" & (month(as.Date(dataTraining$Date)) == 6
        | month(as.Date(dataTraining$Date)) == 7 | month(as.Date(dataTraining$Date)) == 8)]
        <- "Summer"
    dataTraining$Season[dataTraining$Season == "" & (month(as.Date(dataTraining$Date)) == 9
        | month(as.Date(dataTraining$Date)) == 10 | month(as.Date(dataTraining$Date)) == 11)]
         <- "Fall"

    dataTraining$SummerMonthDummy <- 0
    dataTraining$SummerMonthDummy[ dataTraining$Season == "Summer"
                                 | dataTraining$Season == "Summer_Break" ] <- 1
    dataTraining$WinterMonthDummy <- 0
    dataTraining$WinterMonthDummy[ dataTraining$Season == "Winter"
                                 | dataTraining$Season == "Christmas_Break" ] <- 1
```

```
    return(dataTraining)
}

addReopenedFlag <- function(dset)
{

    #print("Sorting table")
    dset <- dset[order(dset$Store, dset$Date),]

    # the vector to return
    Reopened <- rep(0, length(dset$Open))

    # make a vector of true/false according to if there is a change or not
    diffs <- dset$Open[-1L] != dset$Open[-length(dset$Open)]; #diffs

    # make a vector of the last indexes of each particular run
    idx <- c(which(diffs), length(diffs)); #idx

    # this vector counts how long each run lasted
    runs <- diff(c(1, idx)); #runs

    # find index values of the last day of each long run and length of each run
    # NOTE: IF HAVE PROBLEMS, MAYBE JUST FLAG 14 DAYS AND UP HERE RATHER THAN 2
    poss <- idx[runs > 2]; #poss
    lengths <- runs[runs > 2]; #lengths

    # check each long run to see if it is a run of closures
    numRuns <- length(poss); #numRuns
    #print("Checking runs")
    for(i in 1:numRuns)
    {
        if(dset$Open[poss[i]] == 0) # this run was of closed days
        {
            # mark the first days of REOPENING as "bump"
            currStore <- dset$Store[poss[i]]

            #print(paste("Found one for Store #", currStore, "at index # ", poss[i]))

            if(lengths[i] < 5) # long weekend, maybe just one day's bump
            {
                cap <- 1
            }
            else if(lengths[i] < 31) # a month or less closed gives it several days
            {
                cap <- 5
            }
            else # after longer than a month, give the excitement two weeks
            {
                cap <- 14
            }
            # don't go past our dataset, though
            cap <- min(cap, length(dset$Open) - poss[i])
            #print(paste("For a run of", lengths[i], "cap is", cap))
```

```
            j <- 1
            while(j <= cap & dset$Store[poss[i] + j] == currStore)
            {
                #print(paste("Put a 1 in", (poss[i] + j)))
                Reopened[poss[i] + j] <- 1
                j <- j + 1
            }
        }
    }
    #print(paste("done.  Reopened has values of"))
    #print(unique(Reopened))
    return(Reopened)
}

addPriorSales <- function(dset, pwd)
{
    #dset <- dataTraining

    # When necessary, we make a file of 2013 and 2014 by store, by month numbers
    if(!file.exists(paste(pwd, "data_PriorSales.csv", sep="")))
    {
        print("Making data file")
        makeSalesFile(pwd)
    }
    print("Reading in data file")
    Prior <- read.csv(paste(pwd, "data_PriorSales.csv", sep=""), header=T)
    Prior <- Prior[, -1] # drop column of row numbers

    # Then, for every month and store in the dataset we are given,
    # we find the prior year's sales information and put that in the dataset
    dset$PriorSalesKey <- paste(dset$Store, months(as.Date(dset$Date)),
                                (year(as.Date(dset$Date)) - 1), sep="")
    dset <- merge ( Prior, dset, by = "PriorSalesKey", all.y = TRUE )
    dset$PriorSales[is.na(dset$PriorSales) | is.nan(dset$PriorSales)] <- 0
    dset$PriorSales <- dset$PriorSales + 1
    dset$LogPriorSales <- log(dset$PriorSales)
    return(dset)
}

makeSalesFile <- function(pwd)
{
    Sales <- read.csv(paste(pwd, "train.csv", sep=""), header=T)
    Sales <- Sales[year(as.Date(Sales$Date)) < 2015,]
    Sales$MonthYear <- paste(months(as.Date(Sales$Date)), year(as.Date(Sales$Date)), sep="")
    Sales <- Sales[order(Sales$Store, Sales$MonthYear),]
    SumData <- tapply(Sales$Sales, list(Sales$Store, Sales$MonthYear), sum)

    Melted <- melt(SumData)
    colnames(Melted) <- c("Store", "MonthYear", "PriorSales")
    Melted$PriorSalesKey <- paste(Melted$Store, Melted$MonthYear, sep="")

    write.csv(Melted[ ,c("PriorSalesKey", "PriorSales")], paste(pwd, "data_PriorSales.csv", sep=""))
}
```