# Walmart Store Sales Forecasting Model

*Bumeng Zhuo*

**Abstract**

This is a report for the nonparametric course project. The goal of this project is to build appropriate models based on historical sales data in order to forecast weekly sale of Walmart store accurately. In the report, both parametric and nonparametric models will be constructed and be compared.

## 1 Introduction

To make decisions based on historical retail data is a challenge for supermarkets, especially the effect of holidays on the sales is hard to evaluate. Thus, the goal of this report is to construct a relatively reliable model based on historical sales data and well predict the future sales. Data are all collected from `Kaggle`[1]. There are totally 45 Walmart stores in the dataset, along with many departments and different regions, also including selected holiday markdown events.

In this report, both parametric model and nonparametric model will be constructed. Furthermore, the efficiency and the results of two constructed models will be discussed in detail.

### 1.1 Data description

The web only provides dataset `train`, `store` and `feature`.

- `train` contains weekly sales of 81 departments of 45 Walmart stores from 2010-2-5 to 2012-10-26, also including big sale holidays indicator, i.e. whether this day is or not Thanksgiving, Christmas, Super Bowl and Labar day.
- `store` provides some information for 45 stores, including different size and shape of each store.
- `feature` includes some features about the place where stores locate, e.g. `temperature`, `fuel price`, `unemployment`, for each week from 2010-2-5 to 2013-7-26.

### 1.2 Pre-processing

#### (a) choose particular store

Do pre-processing to `train`. Separate the `date` as `year`, `month`, `mday`, `week`. And compute `weeklabel` which is the number of weeks from first date. Do `xyplot` as shown in figure 1.

[1]data from http://www.kaggle.com

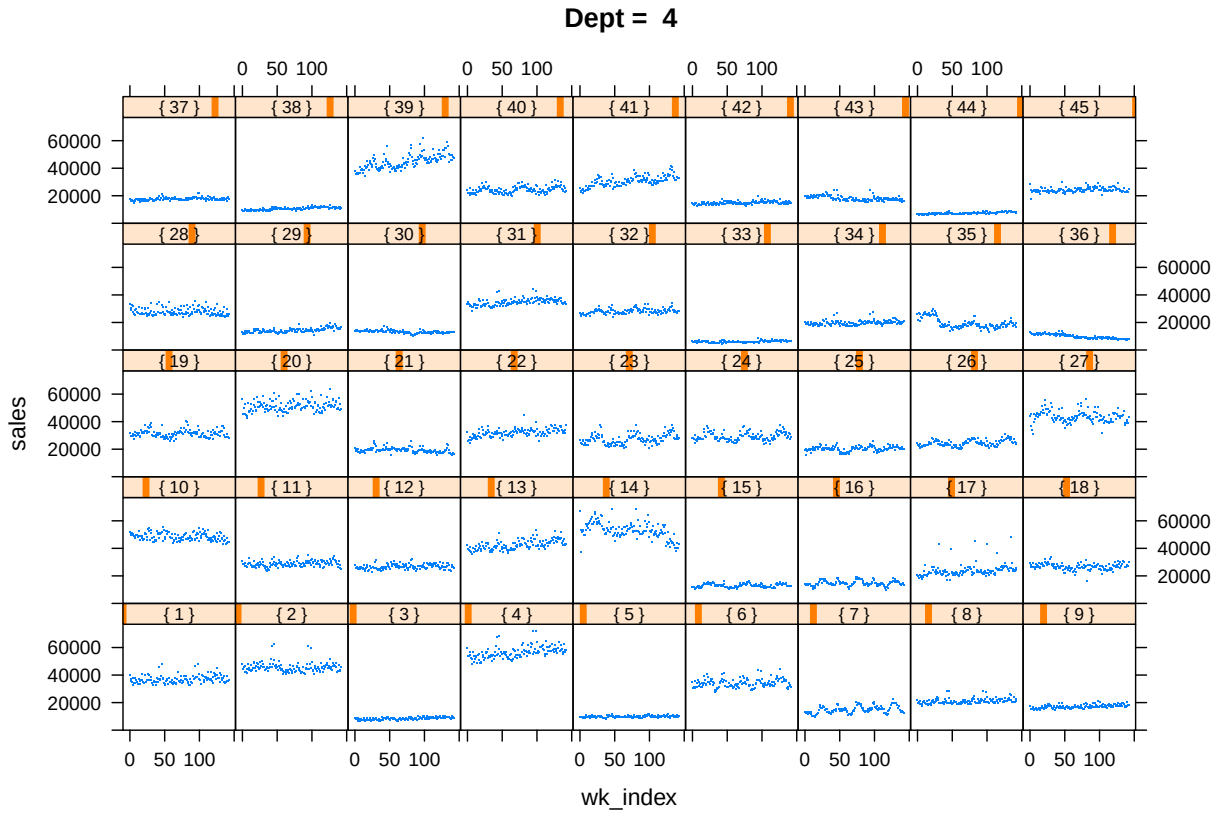


Figure 1: sales of each store in dept 4

Figure 1 is the plot of weekly sales versus number of weeks in each store with department 3. And there are totally 81 different departments, i.e. there are 81 different figures. It's apparent that sales of stores are totally different in some stores, e.g. store #44, #39, #23. It's hard to use same models to explain them. But there are also some well representative models, e.g. #22, #23 and #24 look very similar.

Since the goal of project is to contruct models in order to analyse each store with each department, I particularly choose store 23 to construct models for its sale. And in latter part, all analyses are based on store 23, since it's well representative and fluctuant.

**(b) variables explanation**

Merge all information of different table to one table. And the total predictors are shown as below:

Table 1: variable explanation table

| predictor | meanings |
|---|---|
| `year`,`month`,`mday` | date |
| `yday`,`week` | days and weeks in that year |
| `weeklabel` | total weeks from first date |
| `isholiday`,`holiday` | whether it is holiday, which holiday it is |
| `dept` | department associated with store 23 |
| `temp`,`fuel`,`cpi`,`unemp` | temperature, fuel-price, CPI, unemployment |

**(c) log-scale of weekly sales**

Since the range of data is large, in order to easily visualize the data, log-scale of weekly sales is preferred to use. Because the minimum value of sales is -298, `logsales` is computed based on

$$\log(sales) = \log(sales + 299)$$

**(d) outlier analysis**

In the dataset, store #23 has totally 77 departments. When fixed date, the other feature variables are all the same for all departments. However, some departments only appears very few while most of them appeares 143 times.

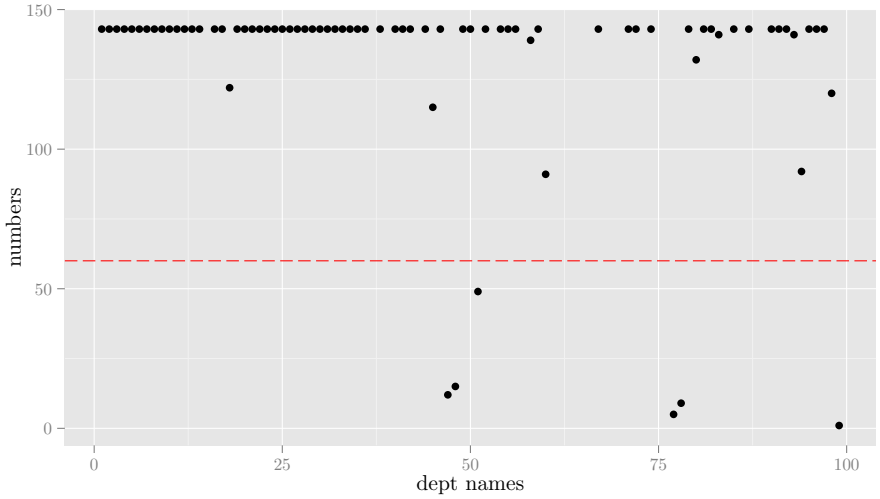


Figure 2: number of departments

As we can see, these small number department data would cause a large bias toward other data. Therefore, I delete these data from the dataset which are smaller than 60 (under red line), and only construct model for totally 71 departments for store #23.

**(e) determine the trainset and testset**

In order to compare the efficiency of models, we need to choose some data as testset. In this case, since all data have been categorized based on department, it's better to randomly choose 1/10 data of each department to form `testset`, and what's left is `trainset`.

**(f) test of accuracy: RMSLE**

Since we have `testset` and true values of each data, in order to compare the efficiency and accuracy of different model, predictions are evaluated by root mean squared logarithmic error (RMSLE), computed as:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(m_i - \hat{m}_i))^2},$$

where $m_i$ is the true value on log-scale, $\hat{m}_i$ is the estimate on log-scale, and $n$ is the number of entries to be evaluated.

Thus, in the end of pre-processing, we have two dataset. `trainset` is used to construct models and `testset` is utilized to test the accuracy of models. And in each set, we have 13 predictors and 1 responce `logsales`.

# 2 Parametric method

Since we've got all the predictors and corresponding responce, the most straightforward way is to fit a linear regression.

## 2.1 Linear regression assumption

In this case, regression model is multivariate. Thus we are doing estimate based on:

- the true value of responce could be linearly explained by all predictors, i.e. predictors are all linear form in the regression model;
- the errors between observation values and true values are normally distributed;
- all errors i.i.d complies with $N(0, \sigma^2)$.

## 2.2 Linear model operation

After trying several kinds of models, we gets the final linear model:

$$\begin{aligned} lm: \ logsales \sim & \ temp + fuel + cpi + unemp + \\ & factor(dept) + factor(year) + factor(month) + factor(holiday) + \\ & temp : as.factor(dept) + fuel : factor(dept) + cpi : factor(dept) + unemp : factor(dept) + \\ & temp : factor(year) + fuel : factor(year) + cpi : factor(year) + unemp : as.factor(year) + \\ & temp : factor(month) + fuel : factor(month) + cpi : factor(month) + unemp : factor(month) + \\ & temp : factor(holiday) \end{aligned}$$

In this model, many interaction terms are also included in order to well explain the responce.And the results of linear model is great, shown as below:

- Residual standard error: 0.325 on 8554 degrees of freedom;
- Multiple R-squared: 0.9487, Adjusted R-squared: 0.946
- F-statistic: 370.8 on 427 and 8554 DF, $p$-value: $<$ 2.2e-16

Based on the results, we can see that adjusted R-squared has been 0.946, which means 94.6% variant of y could be explained by this linear model. And the $p$-value is significantly small.

## 2.3 Parametric model evaluation

### (a) RMSLE score

Use this linear model to predict the `logsales` in `testset`, and we can easily get the RMSLE score computed as

$$RMSLE.lm = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(logsale_i - \hat{logsale_i})\right)^2} \approx 0.3457.$$

It shows that the linear model is actually accurate and it can well predict the true value in `testset`.

### (b) CI of parameters

Since this model has so many parameters, we can just take the head of coefficient table as example.

Table 2: coef. table of linear model

| Parameters | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | -4.1141 | 65.7567 | -0.0626 | 0.9501 |
| temp | 0.0319 | 0.0301 | 1.0581 | 0.2900 |
| fuel | -2.2483 | 0.9223 | -2.4377 | 0.0148 |
| cpi | 0.1972 | 0.4387 | 0.4496 | 0.6530 |
| unemp | -0.9407 | 3.0623 | -0.3072 | 0.7587 |
| as.factor(dept)2 | 2.5915 | 12.4645 | 0.2079 | 0.8353 |

Since each $\hat{\beta}$ is unbiased estimate of $\beta$. At $\alpha$ significance level, we can easily compute the confidence interval for parameters:

$$[\hat{\beta} \pm t_{2/\alpha}(n-p)].$$

And for example, the CI of $\beta_{fuel}$ is $[-4.0562, \ -0.4404]$.

### (c) predictive interval for fitted values

Using linear model, it's easy to compute the predictive interval for parameters, and so is confidence band (here I'm using pointwise predictive band).

Since there are so many departments, just take *dept 1* as example. Plot `logsales` versus `weeklabel` as figure 3.

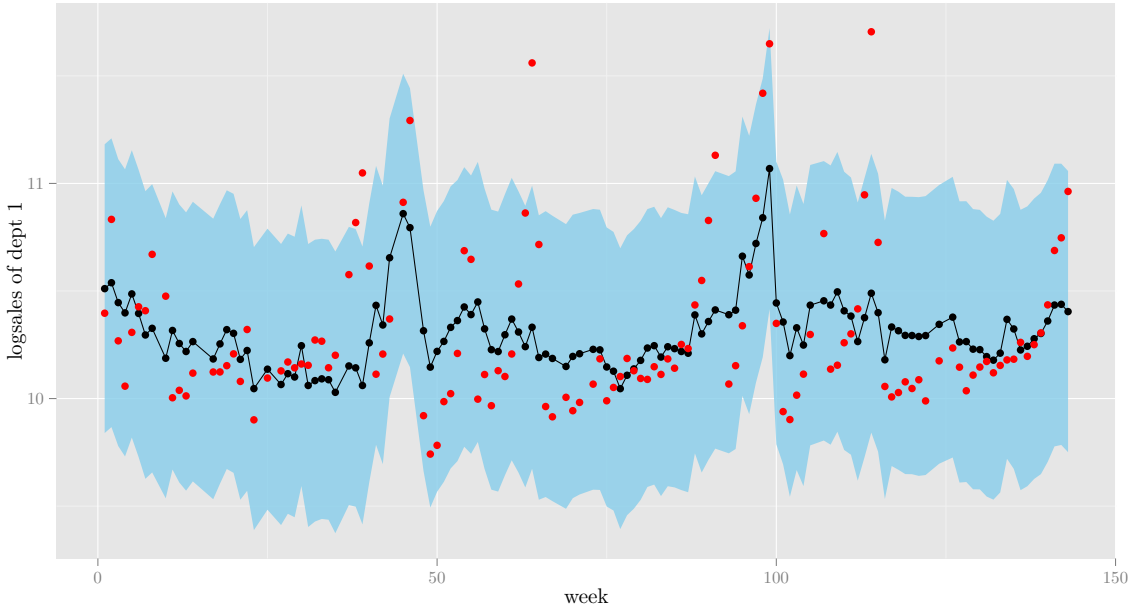


Figure 3: pointwisde predictive interval for fitted value

As we can see in figure 3, the blue background are the predictive band constructed by predicted value. Black points are predicted value and red points are observation value. According to the figure, the red points almost all fall on the blue background, which means this simple linear model is actually accurate.

**(d) compared with other alternative model**

Linear model is the simplest model in parametric regression. Besides that, we can also use LASSO and Ridge regression, etc. Since there are many parameters in the model, in this case $p = 428$, using LASSO or Ridge regression model could help reduce many parameters to 0 due to the penalty of $\|\beta\|$. However, in this project, the goal of regression models is to predict store's sale. Thus it would be more accurate to just use linear regression model.

# 3 Nonparametric method

Since parametric methods are based on too strong hypothesis, sometimes the true function in reality cannot meet such strong demands. Therefore here comes nonparametric method to construct the model. And in this project I choose additive model to explain the responce `logsales`.

## 3.1 Nonparametric regression assumption

Unlike linear regresion model, nonparametric regression is based on much weaker assumption. For kernel regression, local linear regression and additive model, the requirement is that function should have *two bounded derivatives*.

## 3.2 Additive regression model

Here I'm using additive regression to construct models for response. Also aftering comparing several additive models, the following model gets the best result:

$$\begin{aligned} gam: \ logsales \sim\ & s(temp) + s(fuel) + s(cpi) + s(unemp) + as.factor(dept) + \\ & as.factor(month) + as.factor(wk) + s(temp) : as.factor(dept) + \\ & s(fuel) : as.factor(dept) + s(cpi) : as.factor(dept) + s(unemp) : as.factor(dept) + \\ & s(fuel) : as.factor(month) + s(cpi) : as.factor(month) + s(unemp) : as.factor(month) \end{aligned}$$

Take a quick look at the formula of additive regression model and linear regression model, it's easy to find that they are quite similar. And both two models include some interaction terms, which means these terms well explain the variant of response `logsales`.

And the result of numeric terms in additive model is shown as below:

Table 3: Approximate significance of smooth terms

| predictor | edf | Ref.df | F | p-value |
|---|---|---|---|---|
| s(temp) | 5.511 | 6.712 | 1.667 | 0.1350 |

| predictor | edf | Ref.df | F | p-value |
|---|---|---|---|---|
| s(fuel) | 1.008 | 1.016 | 3.074 | 0.0804 . |
| s(cpi) | 5.347 | 6.329 | 1.538 | 0.1290 |
| s(unemp) | 2.154 | 2.528 | 0.472 | 0.6558 |

As we can see, the estimated degree of freedom for numeric terms are all more than 1 to make model as smooth as possible. And the results of whole model is shown below:

- R-sq.(adj) = 0.946 Deviance explained = 94.9%
- GCV = 0.11067 Scale est. = 0.10504 n = 8982

The additive model could explain almost 94.6% variants of response, which is quite similar to linear regression.

### 3.3 Additive model evaluation

#### (a) RMSLE scores

Use this additive model to predict the `logsales` in `testset`, and we can easily get the RMSLE score computed as

$$RMSLE.gam = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(logsale_i - \hat{logsale_i}\right)^2} \approx 0.3450,$$

which is also quite close to the result of linear regression model but just a littler smaller. It shows that the results of two models are quite similar.

#### (b) Confidence interval for RMSLE score

Do repeat B times bootstrap in `R`. In this case, $B = 200$. Get the RMSLE score each time, and plot histogram as follows.

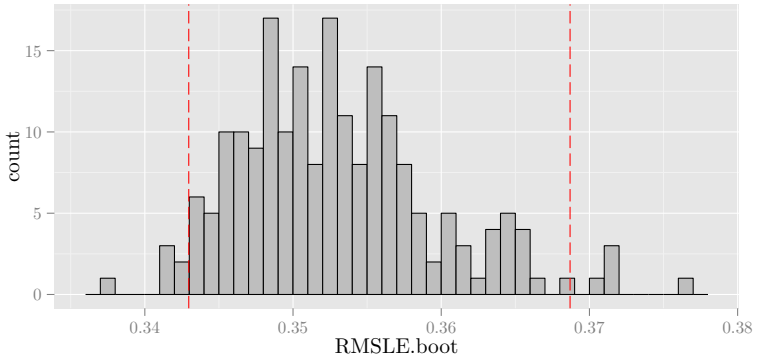


Figure 4: histogram of RMSLE scores by bootstrap

In the figure 4, red dash line is the 0.025 and 0.975 quantile of RMSLE scores. And as we can see in the histogram, lots of scores range from around 0.343 to 0.368, which shows the result is quite accurate using additive model.

### (c) Confidence Band of predicted value

In order to compute 95% confidence band for predicted value, here I'm using Bonferroni method to compute the confidence interval for each point. Although it is a little too conservative, it's good to make confidence band wider and contain as much real points as possible.

Since $\alpha = 0.05$, the significance level for each point is $\alpha = 0.05/n$, where $n$ is the number of predict points. Here is the confidence band using additive model toward *dept 1* data.

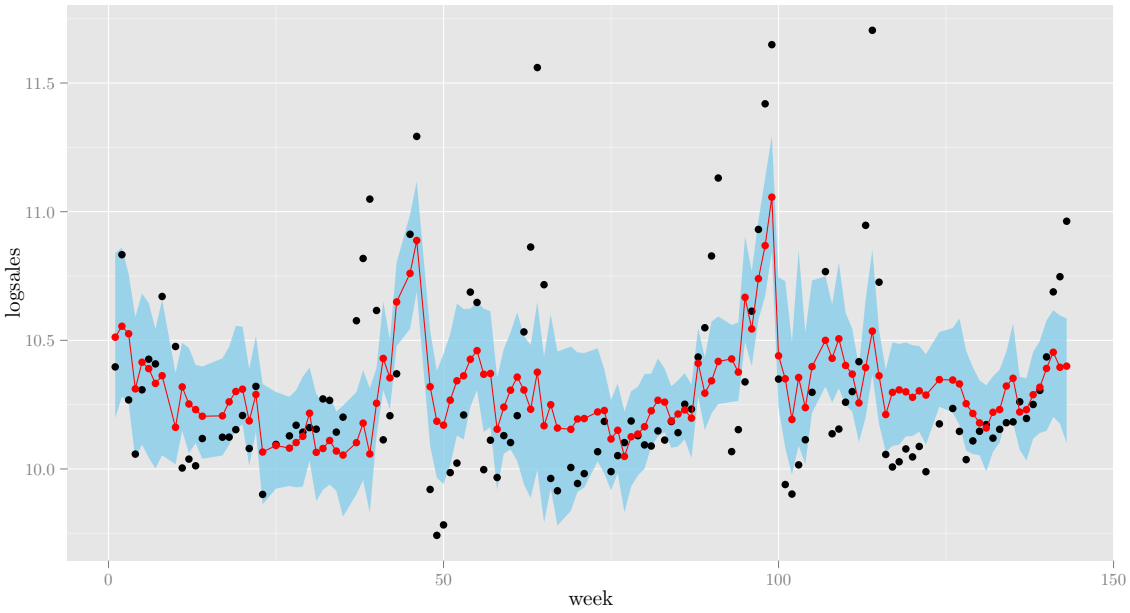


Figure 5: confidence band for predicted value on dept 1

In figure 5, blue area is the constructed confidence band, black points are the true values of data, and red points are the predicted value. As we can see, the confidence band constructed by bootstrap using Baferroni method is narrower than pointwise predictive interval in linear regression. However, also most of black points fall on blue area, which shows additive model is proper for prediction in this case.

### (d) compared with other alternative methods

In regression problem, I've tryed using kernel smoother and local linear regression that are mainly used for numeric dataset. In order to apply these two methods, I define the distance of `factor` predictors as "the squared Euclidean distance sums the squared differences between these two categorical vectors.[2]".

However, the result of kernel regression is not that efficient, and I got result as $RMSLE = 1.1424 \gg 0.3450$ in additive model. I think this may be because `factor` predictors play an important role in prediction in this dataset, and kernel regressions don't perform so well in categorical regression case. And thus I choose additive model in this project.

[2]reference from this website: http://www.econ.upf.edu/ michael/stanford/maeb4.pdf

# 4 Discussion

- Although linear regression is the simplest model and it poses strong assumptions on the true model, it performs very well in this case. And it's easy to read model in detail and analyse the result.
- In nonparametric model, since there are many categorical predictors and those predictors are mainly important in the model, kernel regressions don't perform very well. And thus in this report I choose additive model to compare linear model.
- Since additive model smoothes the numeric predictors, the good way to see the smooth result is to plot additive model as follow:

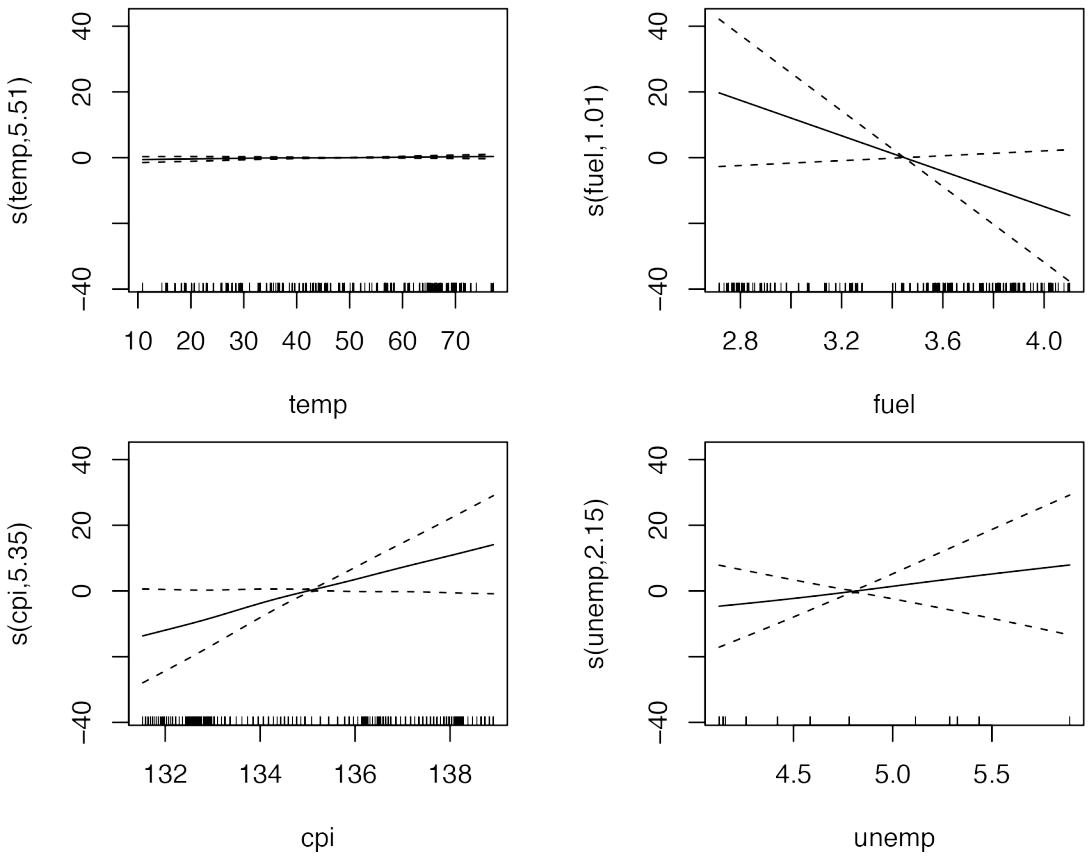


Figure 6: additive model smooth of numeric predictors

As we can see, although additive regression model do some smoothes to the variables, the real plot of function of numeric predictor is almost linear fit, which means linear fit has been adequate. This may be because too many factors are added to the model and so does many interaction terms with numeric predictors.

Thus, as a result, additive model is just slightly better than linear model in this case, although it is based on much weaker assumption.

# 5 Appendix

```
##################
## Preprocessiong
##################

library(lubridate)  # to deal with Dates in R
library(lattice)
setwd("~/Desktop/374 project")

save("ptrain",file="ptrain.Rdata")
dept.names = sort(unique(mytrain$dept))
k=4; # plot the dept with name dept.names[k], k=1, .., 81.

pdf(file="dept=4, each store.pdf",height=6,width=9)
day1="2010-02-05"
tmp=subset(mytrain, dept==dept.names[k])
tmp$wk_index = as.numeric(difftime(tmp$date, day1, units="weeks"))
xyplot(sales ~ wk_index | store, tmp, par.strip.text=list(cex=0.75), pch=".",
       strip = strip.custom(strip.names = FALSE, strip.levels = TRUE),
       main=paste("Dept = ", dept.names[k]))
dev.off()

zbmtrain = mytrain[mytrain$store==23,]
zbmtrain$weeklabel = as.numeric(round(difftime(zbmtrain$date,zbmtrain$date[1],units="week"))+1)
ptrain = zbmtrain[,!colnames(zbmtrain)%in%c("store","size","type")]
save("ptrain",file="ptrain.Rdata")
ptrain$logsales = log(ptrain$sales+299)

############
## outliers
############

i = 2
plot(ptrain[ptrain$date==ptrain$date[i],"logsales"]~ptrain[ptrain$date==ptrain$date[i],"dept"])
dept3 = ptrain[ptrain$dept==3,]
salesbar = aggregate(ptrain$logsales,by=list(date=ptrain$date),mean)

######################
### test and training
######################

pdept.names = unique(ptrain$dept)
dept.len = c()
for(i in 1:length(pdept.names)){
  dept.len = c(dept.len, nrow(ptrain[ptrain$dept==pdept.names[i],]))
}
```

```
outlier = cbind(pdept.names,dept.len)
write.csv(outlier,file="outlier.txt",row.names=F)

ptrain = ptrain[! ptrain$dept %in% pdept.names[which(dept.len <= 60)],]

save("ptrain",file="ptrain.Rdata") #delete outliers

pdept.names = unique(ptrain$dept)
dept.len = c()
for(i in 1:length(pdept.names)){
  dept.len = c(dept.len, nrow(ptrain[ptrain$dept==pdept.names[i],]))
}
dept.len

#####################################################
#### randomly take 1/10 data to form test dataset
#####################################################

trainset = c()
testset = c()
for(i in 1:length(pdept.names)){
  x = ptrain[ptrain$dept==pdept.names[i],]
  nx = nrow(x)
  t = sample(1:nx,size=round(nx/10),replace=F)
  trainset = rbind(trainset, x[-t,])
  testset = rbind(testset, x[t,])
}

save(trainset,file="trainset.Rdata")
save(testset,file="testset.Rdata")
load("testset.Rdata")
load("trainset.Rdata")

##################
## additive model
##################

library(gam)
require(mgcv)

gamtrain1 = gam(logsales ~ s(temp) + s(fuel) + s(cpi) + s(unemp) + as.factor(dept) +
                  as.factor(month) + as.factor(wk) + (temp):as.factor(dept) +
                  (fuel):as.factor(dept) + (cpi):as.factor(dept) + (unemp):as.factor(dept) +
                  (fuel):as.factor(month) + (cpi):as.factor(month) + (unemp):as.factor(month),
                data = trainset)

gamtest = predict(gamtrain1,testset)
```

```r
error.gam = gamtest - testset$logsales
RMSLE.gam = sqrt(t(error.gam)%*%error.gam/length(error.gam))
cat("RMSLE.gam =",RMSLE.gam)

############################
### local linear regression
############################

library(locfit)
loctrain = locfit(logsales~(temp)+(unemp)+(cpi)+(dept),deg=1,data=trainset,alpha=c(0.1,0.1,0.1
loctest = predict(loctrain,testset)
error.loc = loctest - testset$logsales
RMSLE.loc = sqrt(t(error.loc)%*%error.loc/length(error.loc))
cat("RMSLE.loc =",RMSLE.loc)

##################
## linear model
##################

trainlm = lm(logsales~temp + fuel + cpi + unemp + as.factor(dept) +as.factor(year) + as.factor
          as.factor(holiday)+
          temp*as.factor(dept)+fuel*as.factor(dept) + cpi*as.factor(dept) + unemp*as.factor(dep
          temp*as.factor(year) + fuel*as.factor(year) + cpi*as.factor(year) + unemp*as.factor(y
          temp*as.factor(month) + fuel*as.factor(month) + cpi*as.factor(month) + unemp*as.facto
          temp*as.factor(holiday)
        ,  data=trainset )

lmtest = predict(trainlm,testset)
error.lm = lmtest - testset$logsales
RMSLE.lm = sqrt(t(error.lm)%*%error.lm/length(error.lm))
cat("RMSLE.lm =",RMSLE.lm)

train.lm2 = lm(formula = logsales ~ temp + fuel + cpi + unemp + as.factor(dept) +
                 as.factor(month) + as.factor(wk) + temp:as.factor(dept) +
                 fuel:as.factor(dept) + cpi:as.factor(dept) + unemp:as.factor(dept) +
                 fuel:as.factor(month) + cpi:as.factor(month) + unemp:as.factor(month),
               data = trainset)
lmtest2 = predict(train.lm2,testset)
error.lm2 = lmtest2 - testset$logsales
RMSLE.lm2 = sqrt(t(error.lm2)%*%error.lm2/length(error.lm2))
cat("RMSLE.lm2 =",RMSLE.lm2)

###################################
## confidence band of linear model
###################################

a=predict(train.lm2,trainset[which(trainset$dept==1),],interval="prediction",level=0.95)
```

```
FF=as.vector(a[,1])
LL=as.vector(a[,2])
UU=as.vector(a[,3])
x=trainset$weeklabel[trainset$dept==1]
y=trainset$logsales[trainset$dep==1]

plot(x,FF,ylim=c(min(LL),max(UU)),"n")
polygon(c(x,rev(x)),c(LL,rev(UU)),col="grey")
lines(x,FF,ylim=c(min(LL),max(UU)),"l",col="darkred",lwd=2)

A=cbind(x,y,FF,LL,UU)
save(A,file="data/A.Rdata")
load("data/A.Rdata")
p2=ggplot(data.frame(cbind(x,FF)),aes(x=x,y=FF))+
  geom_ribbon(data=data.frame(x), aes(x, ymin=LL, ymax=UU),fill="skyblue",inherit.aes=F, alpha=
  geom_point(aes(x=x,y=y),color="red")+ xlab("week")+ylab("logsales")
p2

################################
##### test data confidence band
################################

b = predict(trainlm,testset[which(testset$dept==1),],interval="prediction",level=0.95)
FFb = as.vector(b[,1])
LLb = as.vector(b[,2])
UUb = as.vector(b[,3])
xb = testset$weeklabel[testset$dept==1]
yb = testset$logsales[testset$dep==1]
p2=ggplot(data.frame(cbind(xb,FFb)),aes(x=xb,y=FFb))+
  geom_ribbon(data=data.frame(xb), aes(xb, ymin=LLb, ymax=UUb),fill="skyblue",inherit.aes=F, al
  geom_point(aes(x=xb,y=yb),color="red")+ xlab("week")+ylab("logsales")
p2

################################
##### CI of additive model
################################

B = 200
lendep1 = length(which(trainset$dept==1))
yhat.train = matrix(0, nrow=lendep1, ncol=B)
RMSLE.boot = numeric(B)
for(b in 95:B){
  nn = sample(1:nrow(trainset),replace=TRUE)
  X = trainset[nn,]
  gamboot = gam(logsales ~ s(temp) + s(fuel) + s(cpi) + s(unemp) + as.factor(dept) +
        as.factor(month) + as.factor(wk) + (temp):as.factor(dept) +
        (fuel):as.factor(dept) + (cpi):as.factor(dept) + (unemp):as.factor(dept) +
```

```
        (fuel):as.factor(month) + (cpi):as.factor(month) + (unemp):as.factor(month),
      data = X)
  yhat.train[,b] = predict(gamboot, trainset[which(trainset$dept==1),])
  yhat.test = predict(gamboot,testset)
  error.boot = yhat.test - testset$logsales
  RMSLE.boot[b] = sqrt(t(error.boot)%*%error.boot/length(error.boot))
}

save(yhat.train,file="data/yhat.train.Rdata")
save(RMSLE.boot,file="data/RMSLE.boot.Rdata")
load("data/yhat.train.Rdata")
load("data/RMSLE.boot.Rdata")


library(ggplot2)
p = ggplot(NULL,aes(RMSLE.boot)) + geom_histogram(aes(RMSLE.boot),fill="grey",color="black",bi
p

nn = length(which(trainset$dept==1))
ylow = apply(yhat.train,1,quantile,(0.025/nn))
yup = apply(yhat.train,1,quantile,(1-0.025/nn))
xx = trainset[trainset$dept==1,]$weeklabel
yy = trainset[trainset$dept==1,]$logsales
yp = predict(gamtrain1,trainset)[trainset$dept==1]
B = data.frame(xx,yy,yp,ylow,yup)
save(B,file="data/B.Rdata")

pci = ggplot(B,aes(x=xx,y=yy))+
  geom_ribbon(aes(xx, ymin=ylow, ymax=yup),fill="skyblue",inherit.aes=F, alpha=0.8)+
  geom_point( )+geom_line(aes(x=xx,y=yp),color="red")+
  geom_point(aes(x=xx,y=yp),color="red")+ xlab("week")+ylab("logsales")
pci


##### kernel regression

# factor distance 0 or 1

fhat = function(X1,x1,X2,x2,y,h){
  dist1sq = rowSums((sweep(as.matrix(X1),2,as.matrix(x1)))^2)
  dist2sq = rowSums(2*(sweep(as.matrix(X2),2,as.matrix(x2))!=0))
  dist = sqrt(dist1sq + dist2sq)
  up = sum(dnorm(dist/h)*y)
  down = sum(dnorm(dist/h))
  return(up/down)
}
```

```
colnames(trainset)
X1 = trainset[,c("temp","fuel","cpi","unemp","weeklabel")]
X2 = cbind(as.numeric(trainset$dept),as.numeric(trainset$year),as.numeric(trainset$month),
     as.numeric(trainset$wk),as.numeric(trainset$holiday),as.numeric(trainset$isholiday),
     as.numeric(trainset$yday),as.numeric(trainset$mday))
y = trainset[,"logsales"]

h = 0.04

cv = c()
for(k in 1:length(h)){
  cvscore = c()
  for(i in 1:nrow(trainset)){
    x1 = trainset[i,c("temp","fuel","cpi","unemp","weeklabel")]
    x2 = cbind(as.numeric(trainset$dept[i]),as.numeric(trainset$year[i]),as.numeric(trainset$mo
               as.numeric(trainset$wk[i]),as.numeric(trainset$holiday[i]),as.numeric(trainset$i
               as.numeric(trainset$yday[i]),as.numeric(trainset$mday)[i])
    X1i = (as.matrix(X1))[-i,]
    X2i = (as.matrix(X2))[-i,]
    yi = y[-i]
    cvscore = c(cvscore, (y[i]-fhat(X1i,x1,X2i,x2,yi,h[k])^2))
  }
  cv = c(cv,mean(cvscore))
}

yhat = c()
for(i in 1:nrow(testset)){
  x1 = testset[i,c("temp","fuel","cpi","unemp","weeklabel")]
  x2 = cbind(as.numeric(testset$dept[i]),as.numeric(testset$year[i]),as.numeric(testset$month[
            as.numeric(testset$wk[i]),as.numeric(testset$holiday[i]),as.numeric(testset$isholi
            as.numeric(testset$yday[i]),as.numeric(testset$mday)[i])
  yhat = c(yhat, fhat(X1,x1,X2,x2,y,h))
}
error.ker =  testset$logsales - yhat
RMSLE.ker = sqrt(t(error.ker)%*%error.ker/length(error.ker))
cat("RMSLE.ker =",RMSLE.ker)

pdf("data/coef_additive.pdf",height=6,width=6)
par(mfrow=c(2,2))
plot(gamtrain1)
dev.off()
```