Predictive Analytics Project: Revenue Prediction

Johnny Chiu

11/12/2017

Table of content

```r
library(dplyr)
library(ggplot2)
```

Read Data

```r
sales_df = read.csv('../_data/catalog sales data.csv')
sales_df$uid = row.names(sales_df)
sales_train = sales_df %>% filter(train==1)
sales_test = sales_df %>% filter(train==0)
```

```
sales_train$datelp6 = as.Date(sales_train$datelp6, "%m/%d/%Y")
sales_train$datead6 = as.Date(sales_train$datead6, "%m/%d/%Y")
```

Exploratory Data Analysis & Data Cleansing

```
distribution_plot <- function(df,col,bin){
  return(ggplot(data=df, aes_string(x=col))+
           geom_histogram(bins=bin)+
           theme_classic()+
           ggtitle(paste("Distribution for feature:",col)))
}
box_plot <- function(df, col){
  return(ggplot(data=df, aes_string(x="''",y=col))+ geom_boxplot())
}
```

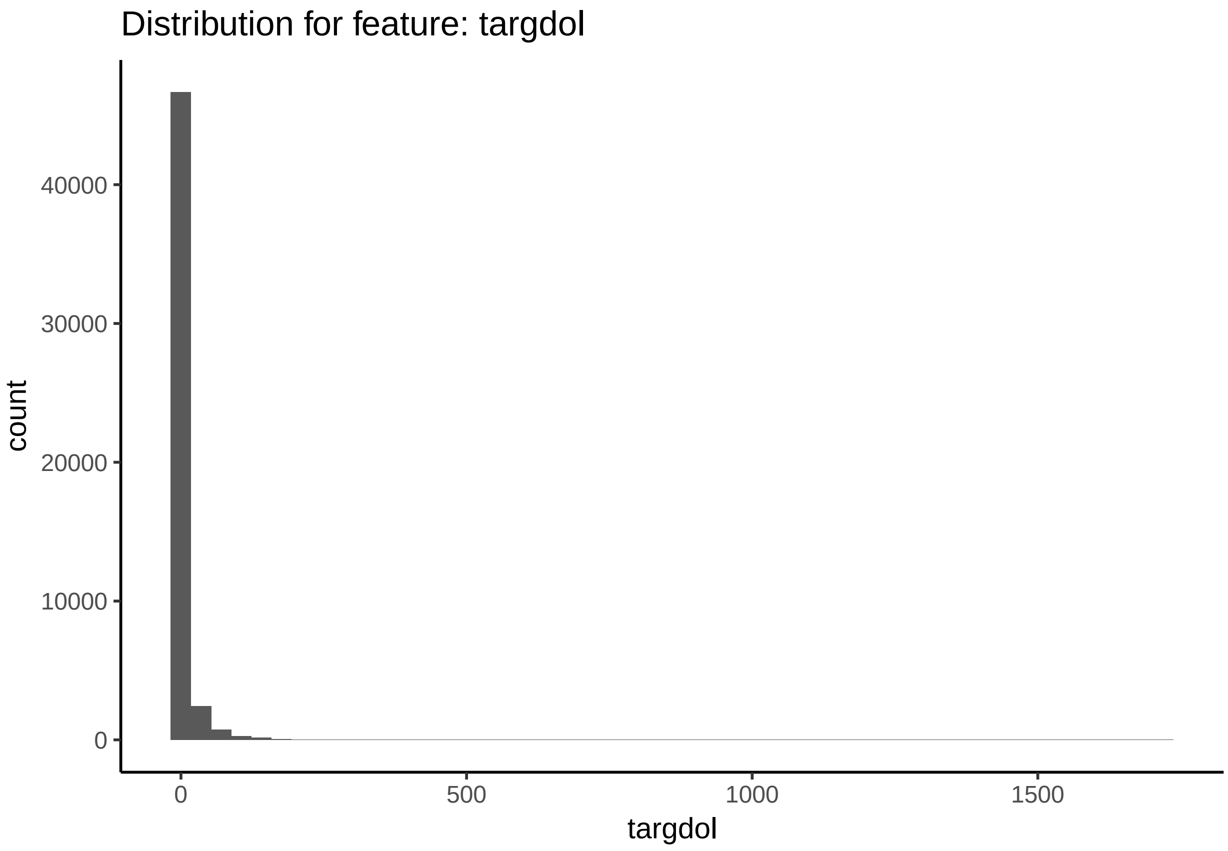
The following EDA will use only the users in the training dataset.

Examine the distribution of each variable

> targdol: dollar purchase resulting from catalog mailing

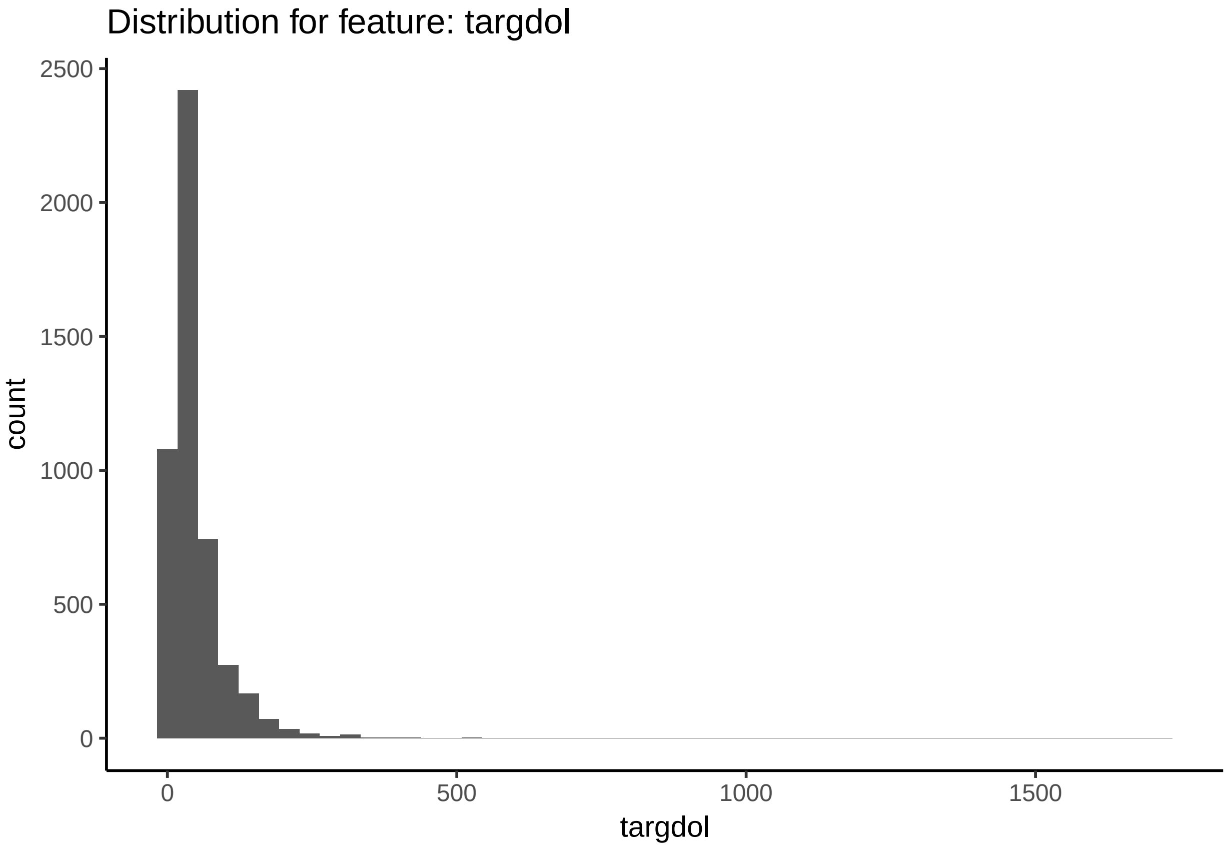
- ***What's the distriution of the dollar purchase resulting from catalog mailing?***

```
distribution_plot(sales_train, "targdol", 50)
```



We can see that the distribution for "targdol" is highly right skewed, where most of the value are 0. We can see that over 0.9039034 of all the values are 0. Let also what the distribution is ignoring all the 0 values

```
targdol_not_zero = sales_train %>% filter(targdol != 0 )
distribution_plot(targdol_not_zero, "targdol", 50)
```



Let's check the summary value of targdol

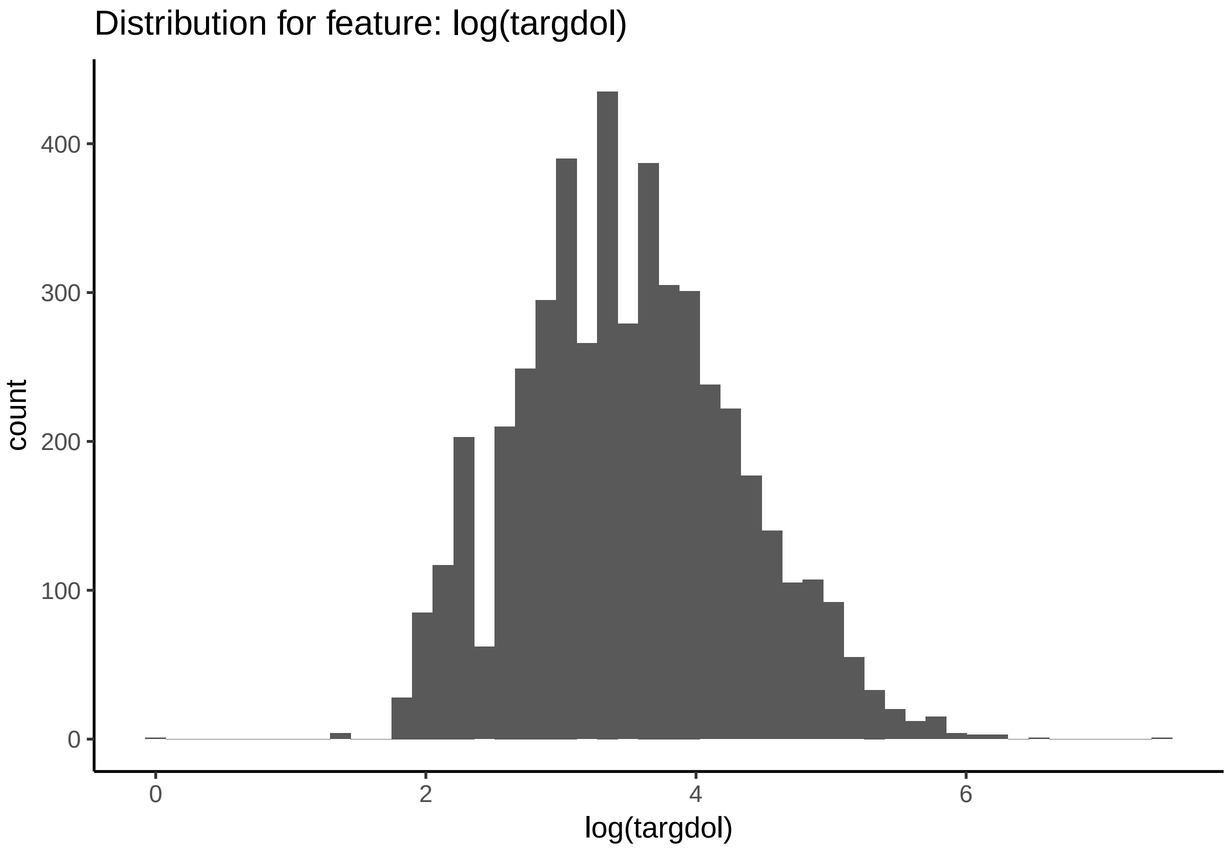
```
summary(targdol_not_zero$targdol)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   18.95   32.20   47.49   56.85 1720.00
```

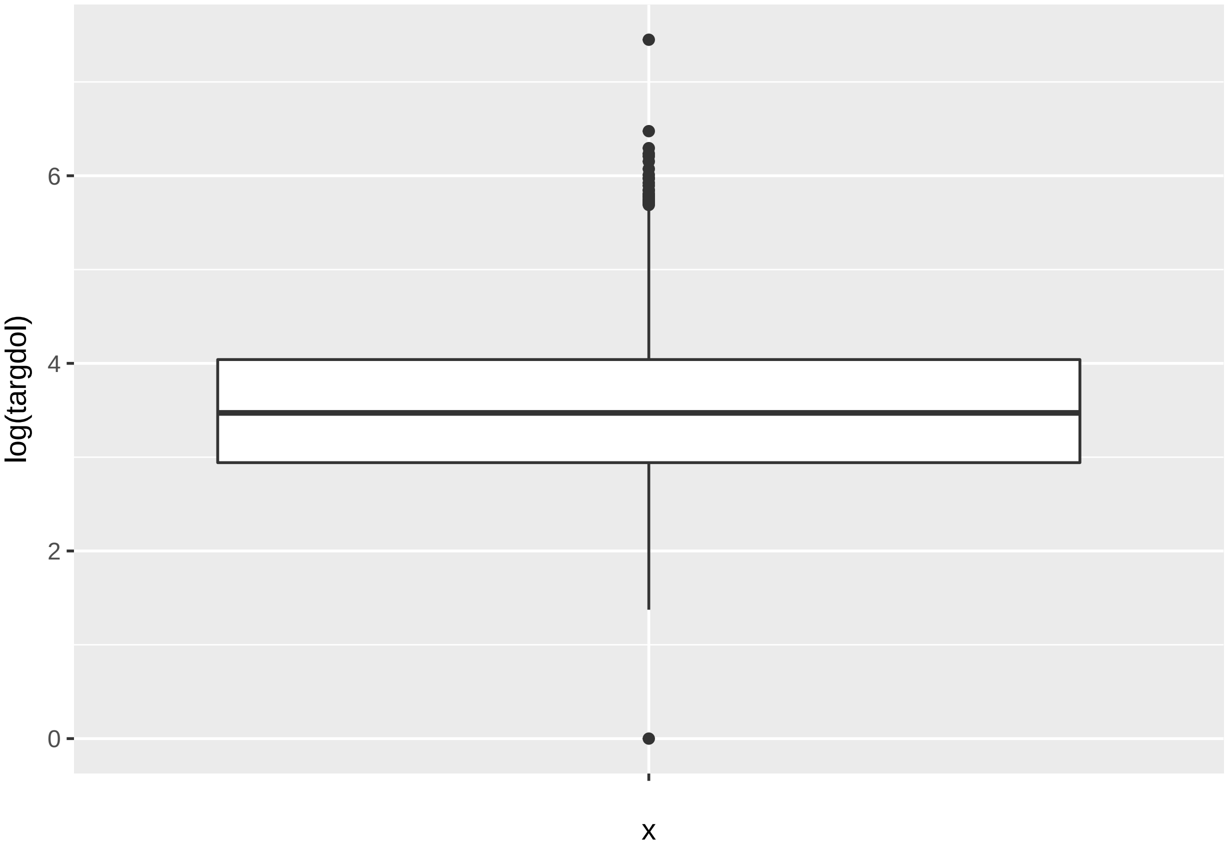
We see that the variance of targdol is 2882.7455301, which is very high.

Since the distribution is higher right skewed, we can use log transformation to normalize it. Let's check how it will be like after taking log

```
distribution_plot(targdol_not_zero, "log(targdol)",50)
```



```r
box_plot(targdol_not_zero, "log(targdol)")
```



We can use log(targdol) as our response variable for the regression model.

> datead6: date added to file

- ***What's the date added to file with the highest frequency?***

```
head(sort(table(sales_train$datead6),decreasing = TRUE),10)/50418
```

```
##
##  1993-09-01  1992-09-01  1991-09-01  2010-11-23  2007-11-10  2010-10-23
## 0.017513586 0.004680868 0.003391646 0.003312309 0.003153636 0.002955294
##  1994-09-01  2009-10-17  2007-10-13  2004-10-09
## 0.002915625 0.002915625 0.002895791 0.002697449
```

1.75% of all the users is added to file file on the date *Sep 1, 1993*.

- ***What's the date added to file with the highest frequency for people who purchase vesus who don't purchase?***

```
sales_train_purchase = sales_train %>% filter(targdol!=0)
sales_train_non_purchase = sales_train %>% filter(targdol==0)


head(sort(table(sales_train_purchase$datead6),decreasing = TRUE),10)/dim(sales_train_purchase)[1]
```

```
##
##  1993-09-01  1992-09-01  1994-09-01  1995-10-01  1991-09-01  1995-08-01
## 0.039422085 0.010319917 0.006604747 0.004127967 0.003921569 0.003921569
##  1995-09-01  2004-10-09  2010-11-23  2009-10-17
## 0.003921569 0.003715170 0.003715170 0.003508772
```

```
head(sort(table(sales_train_non_purchase$datead6),decreasing = TRUE),10)/dim(sales_train_non_purchase)[
```

```
##
##  1993-09-01  1992-09-01  1991-09-01  2010-11-23  2007-11-10  2010-10-23
## 0.015184429 0.004081364 0.003335308 0.003269480 0.003203651 0.002984223
##  2007-10-13  2009-10-17  2004-10-09  1994-09-01
## 0.002918395 0.002852566 0.002589252 0.002523424
```

The highest frequency date for both group is *Sep 1, 1993*.

> datelp6: date of last purchase

- ***What's the date of last purchase with the highest frequency?***

```
head(sort(table(sales_train$datelp6),decreasing = TRUE),10)/dim(sales_train)[1]
```

```
##
##  2012-03-01  2011-11-15  2010-11-15  2009-11-15  2011-03-01  2008-11-15
## 0.032309889 0.023900194 0.010789797 0.007497322 0.007060970 0.005216391
##  2010-11-23  2010-10-23  2011-10-08  2009-10-17
## 0.004125511 0.003927169 0.003193304 0.002955294
```

- ***What's the date of last purchase with the highest frequency for people who purchase vesus who don't purchase?***

```
head(sort(table(sales_train_purchase$datelp6),decreasing = TRUE),10)/dim(sales_train_purchase)[1]
```

```
##
## 2012-03-01 2011-11-15 2010-11-15 2009-11-15 2011-03-01 2008-11-15
## 0.32198142 0.22579979 0.09989680 0.06563467 0.06026832 0.04066047
## 2010-03-01 2007-11-15 2009-03-01 2005-11-15
## 0.02414861 0.02063983 0.01609907 0.01506708
```

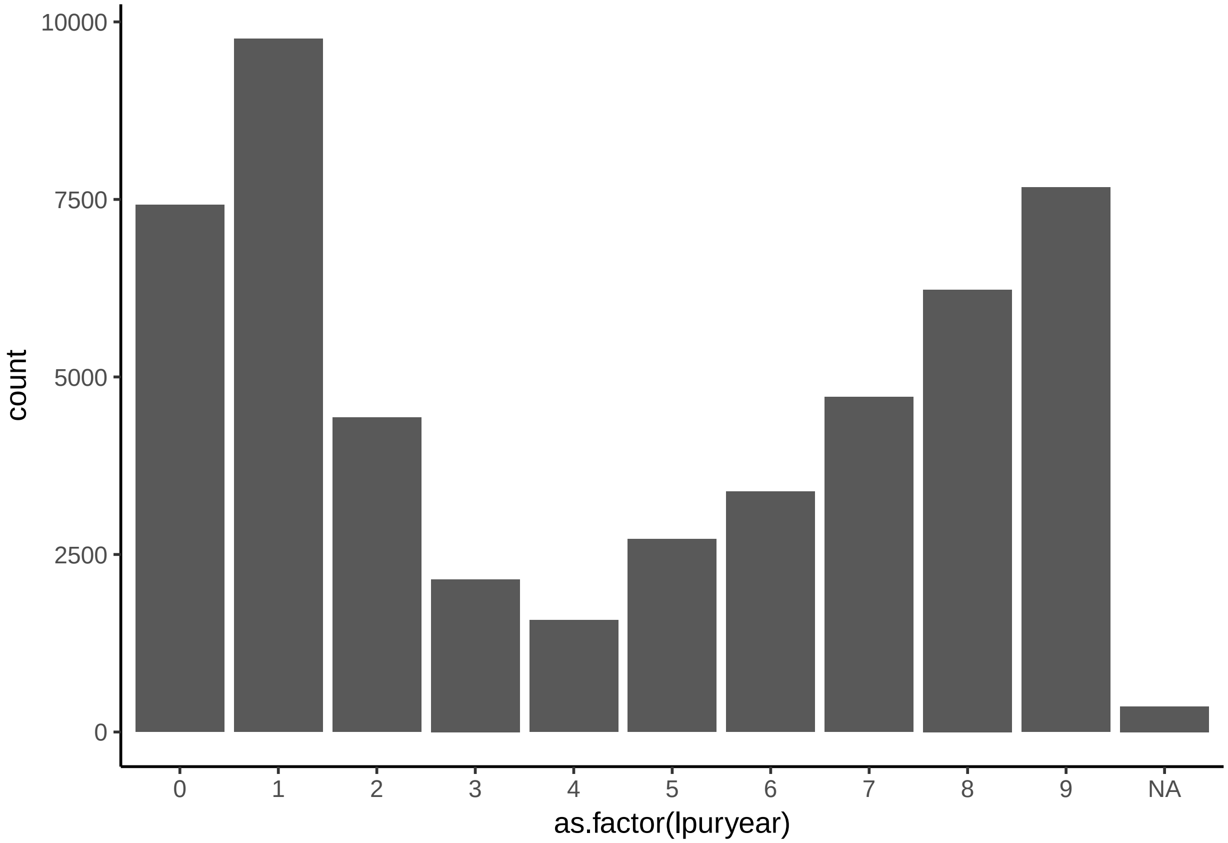
```
head(sort(table(sales_train_non_purchase$datelp6),decreasing = TRUE),10)/dim(sales_train_non_purchase)[
```

```
##
##  2010-11-23  2010-10-23  2011-10-08  2009-10-17  2010-10-26  2010-11-21
## 0.004564106 0.004344678 0.003532794 0.003269480 0.003203651 0.003115880
##  2009-11-29  2011-11-22  2010-11-30  2007-11-10
## 0.003028109 0.003028109 0.002852566 0.002698966
```

One thing worth mentioning about the distribution of the people who have purchased is that the top 2 date accounts for more than 50% of all the customers. It is not usual that more than 50% of the consumers' last purchase is on either *Mar 1, 2012* or *Nov 15, 2011*. We have to keep this in mind.

> lpuryear: latest purchase year

```
ggplot(data=sales_train, aes(x=as.factor(lpuryear))) + geom_bar() + theme_classic()
```

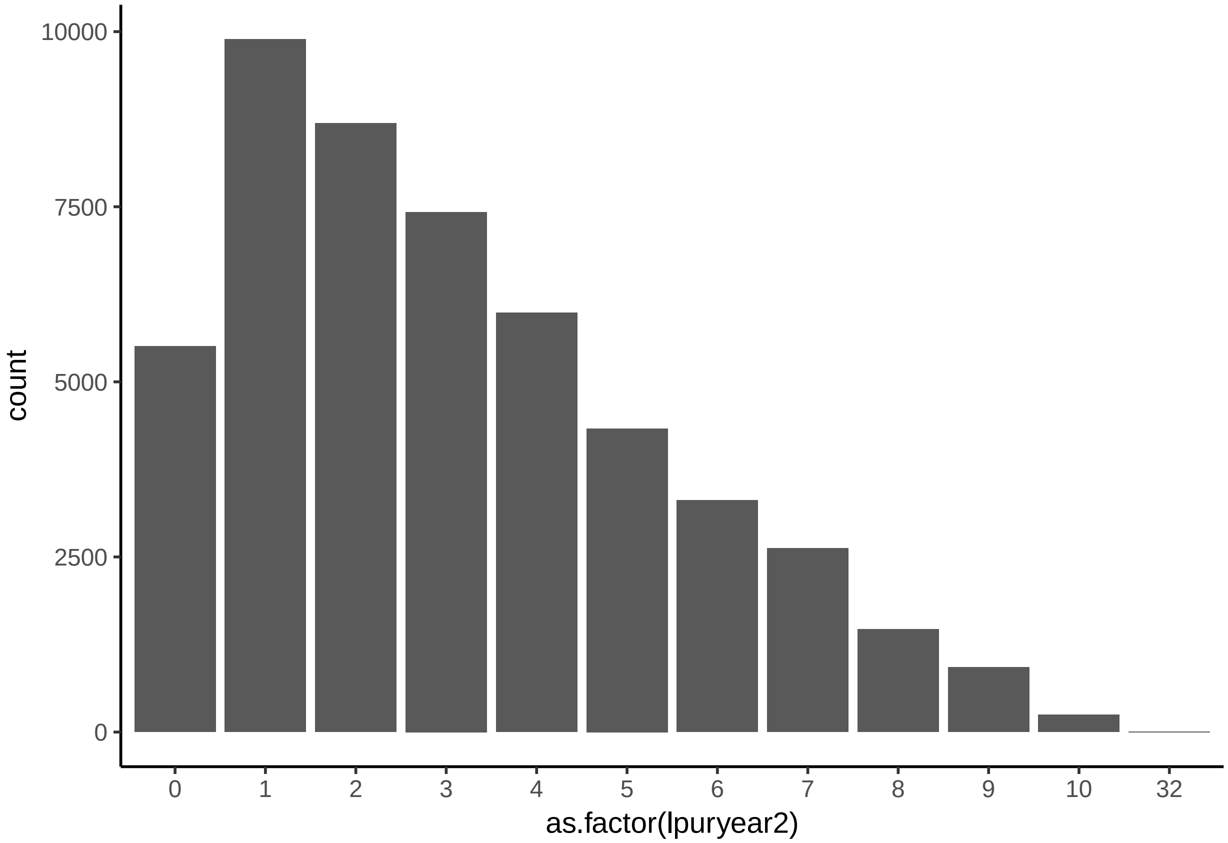


There are 361 NA's in this feature. Since we also have the feature "datelp6", we can try to recreate this feature using that feature.

```
sales_train$lpuryear2 = floor(as.numeric(difftime(as.Date("2012-12-01"), sales_train$datelp6, unit="wee
```

How the distribution look like using the recreated "lpuryear2"?

```
ggplot(data=sales_train, aes(x=as.factor(lpuryear2))) + geom_bar() + theme_classic()
```



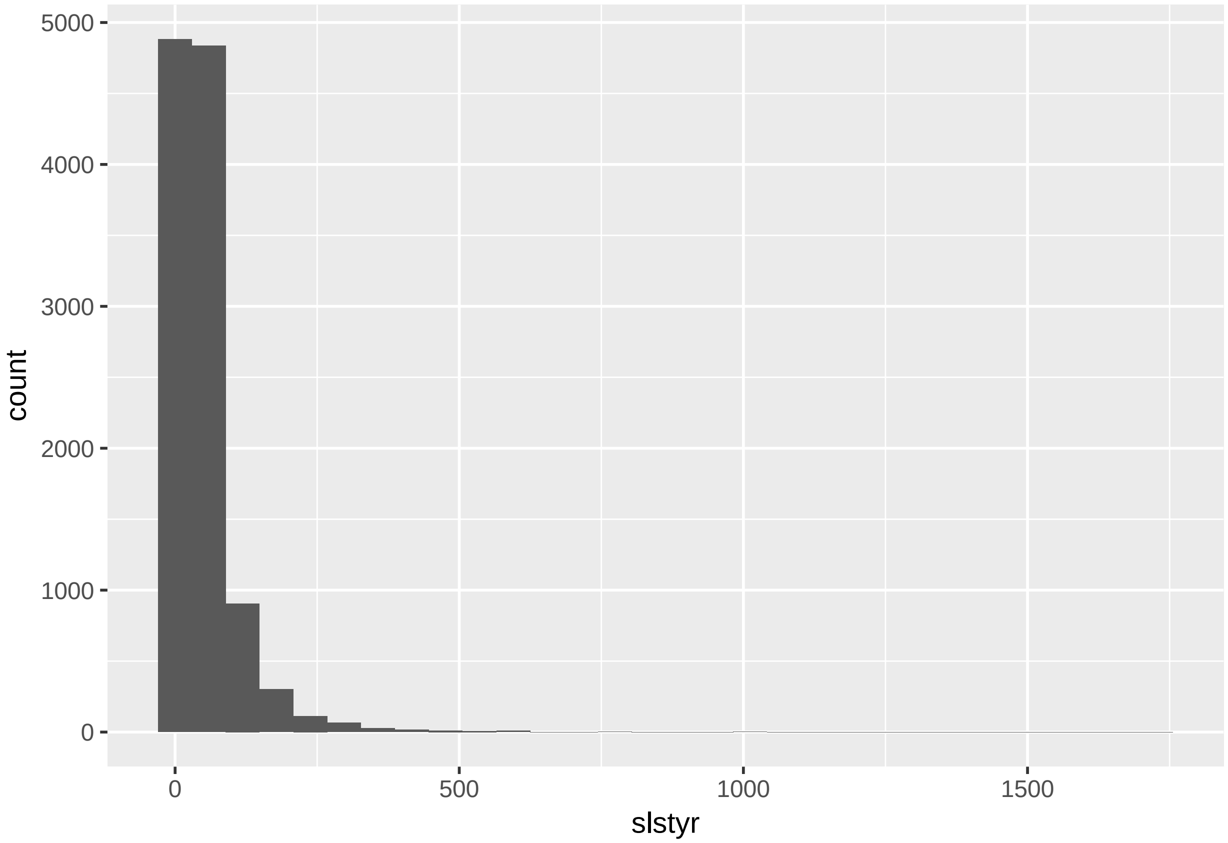
It seems that the lpuryear can not be re-created using the feature "datelp6". We will need to figure how to fill the missing value later. Maybe we can fill it by the current lpuryear distribution.

> slstyr: sales this year; slslyr: sales last year, sls2ago: sales 2 years ago, sls3ago: sales 3 years ago

- ***What's the distribution for all these sale ignoring people who didn't make any purchase?***

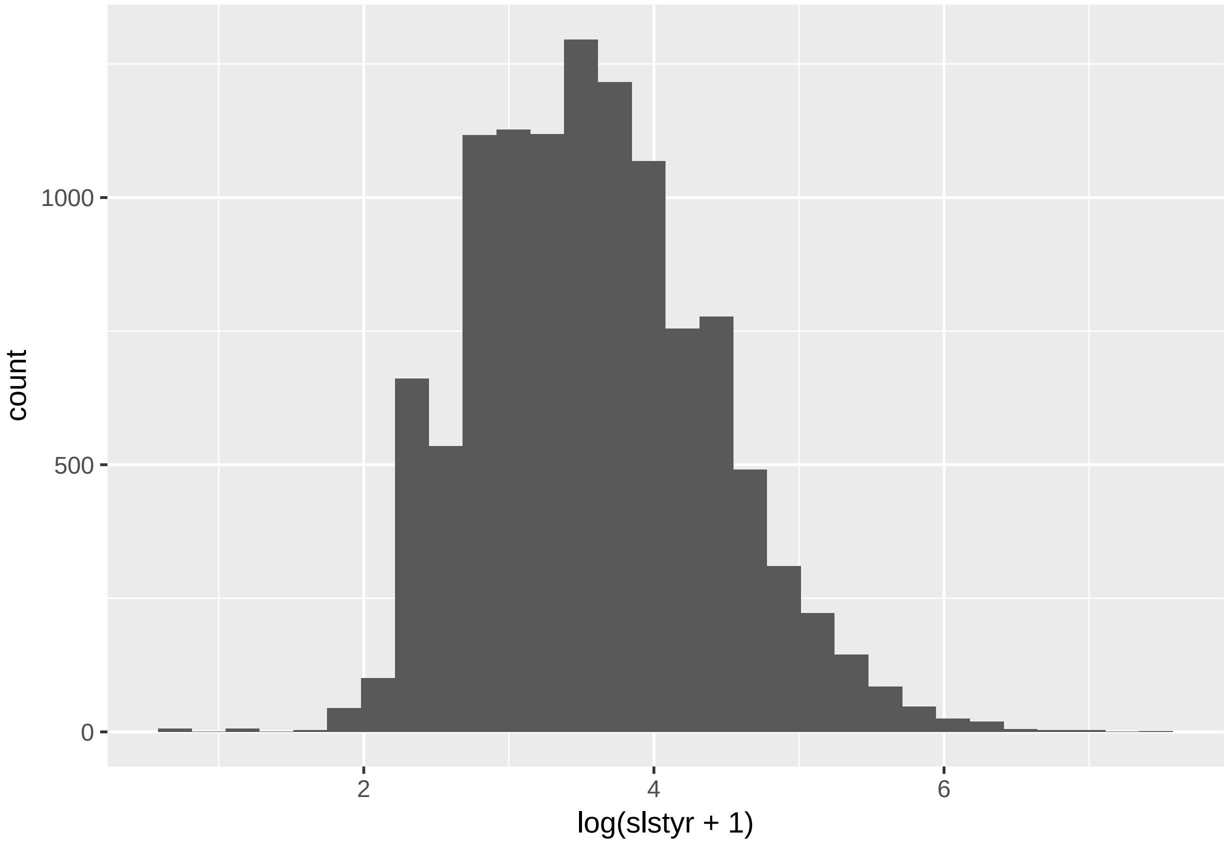
```
ggplot(data=sales_train[sales_train['slstyr']!=0,], aes(x=slstyr))+
        geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

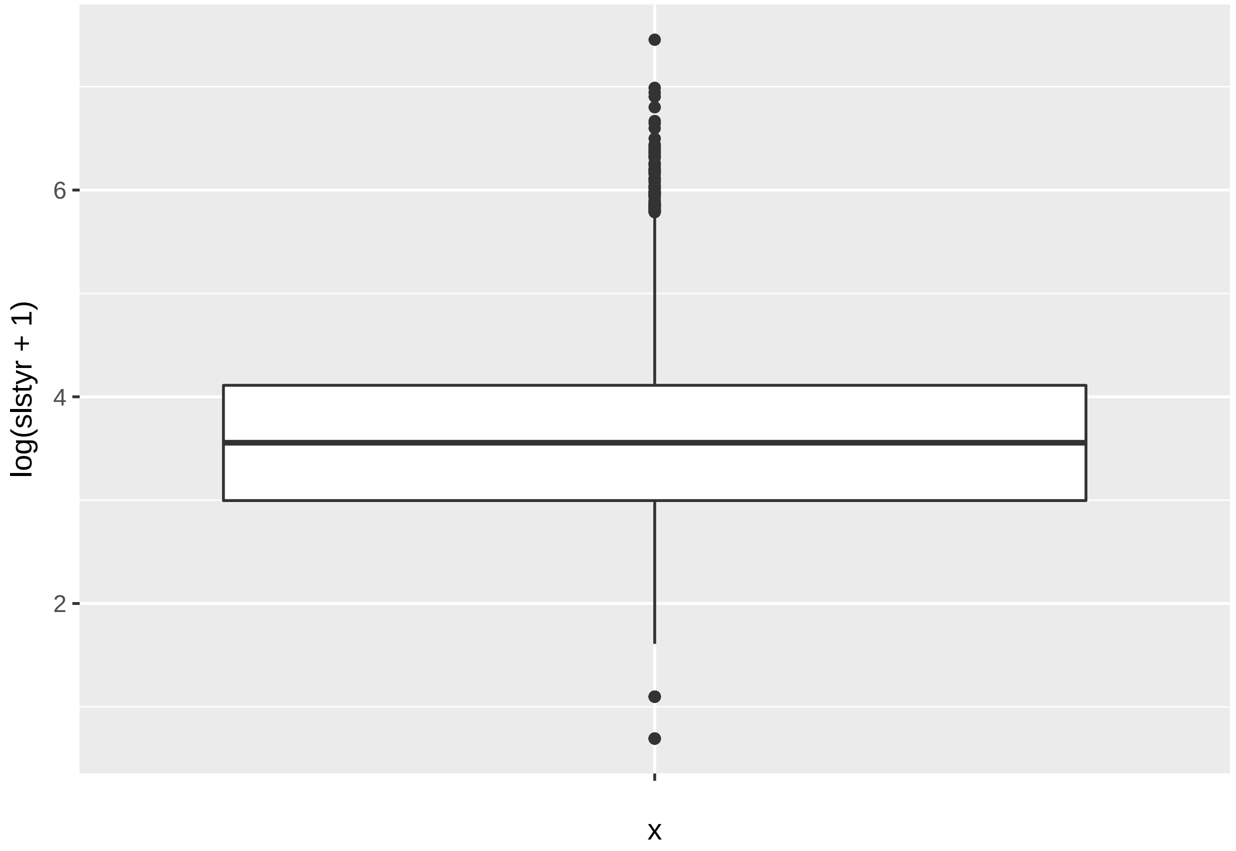


```
ggplot(data=sales_train[sales_train['slstyr']!=0,], aes(x=log(slstyr+1)))+ geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

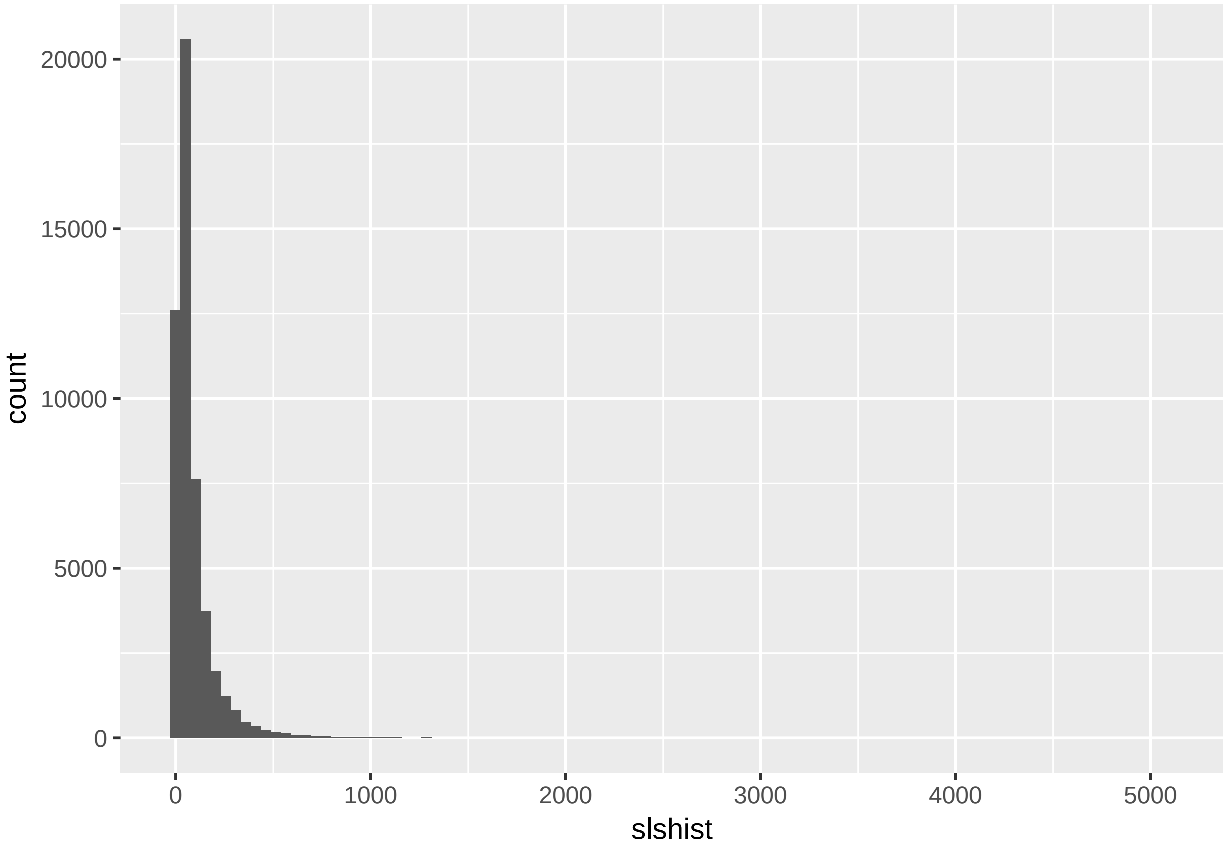


```
ggplot(data=sales_train[sales_train['slstyr']!=0,], aes(x='',y=log(slstyr+1)))+ geom_boxplot()
```



> **slshist: LTD dollars**

```
ggplot(data=sales_train, aes(x=slshist))+
        geom_histogram(bins=100)
```



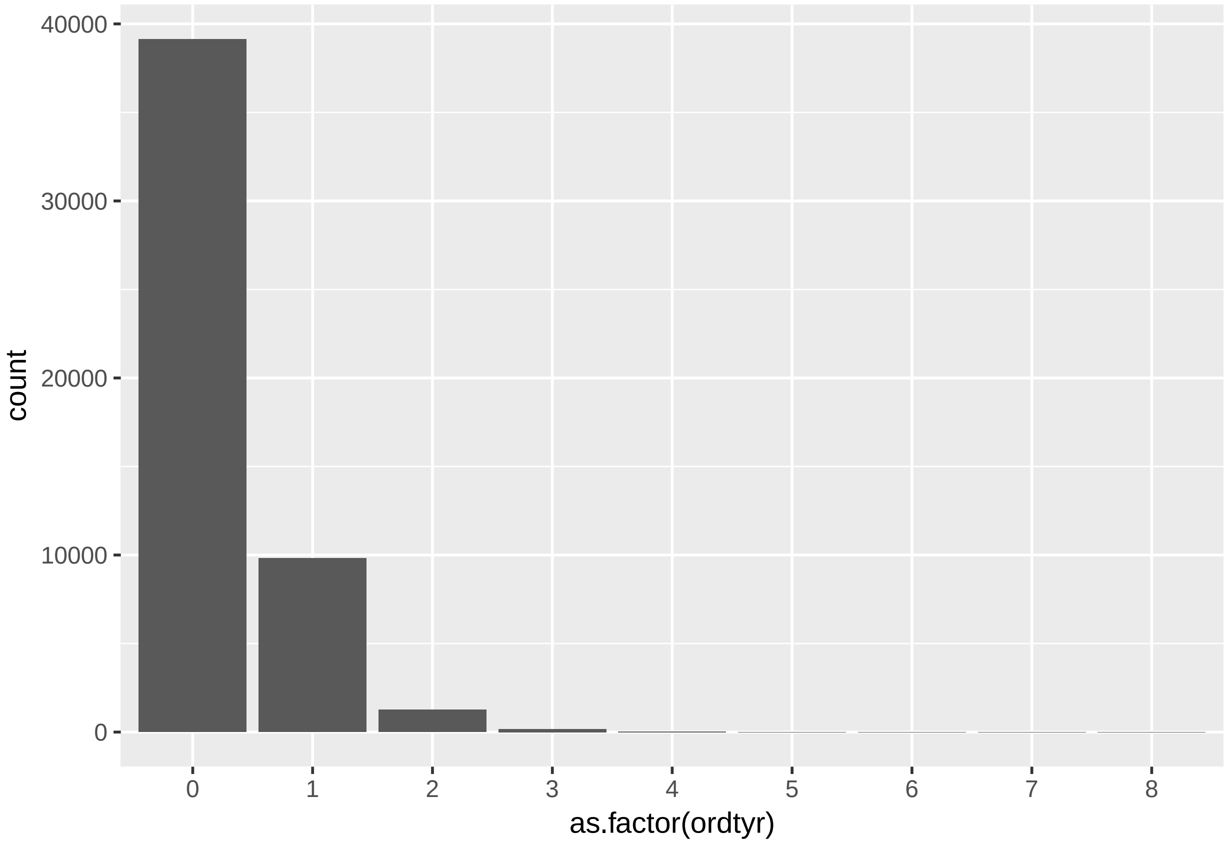
> ordtyr: orders this year

> ordlyr: orders last year

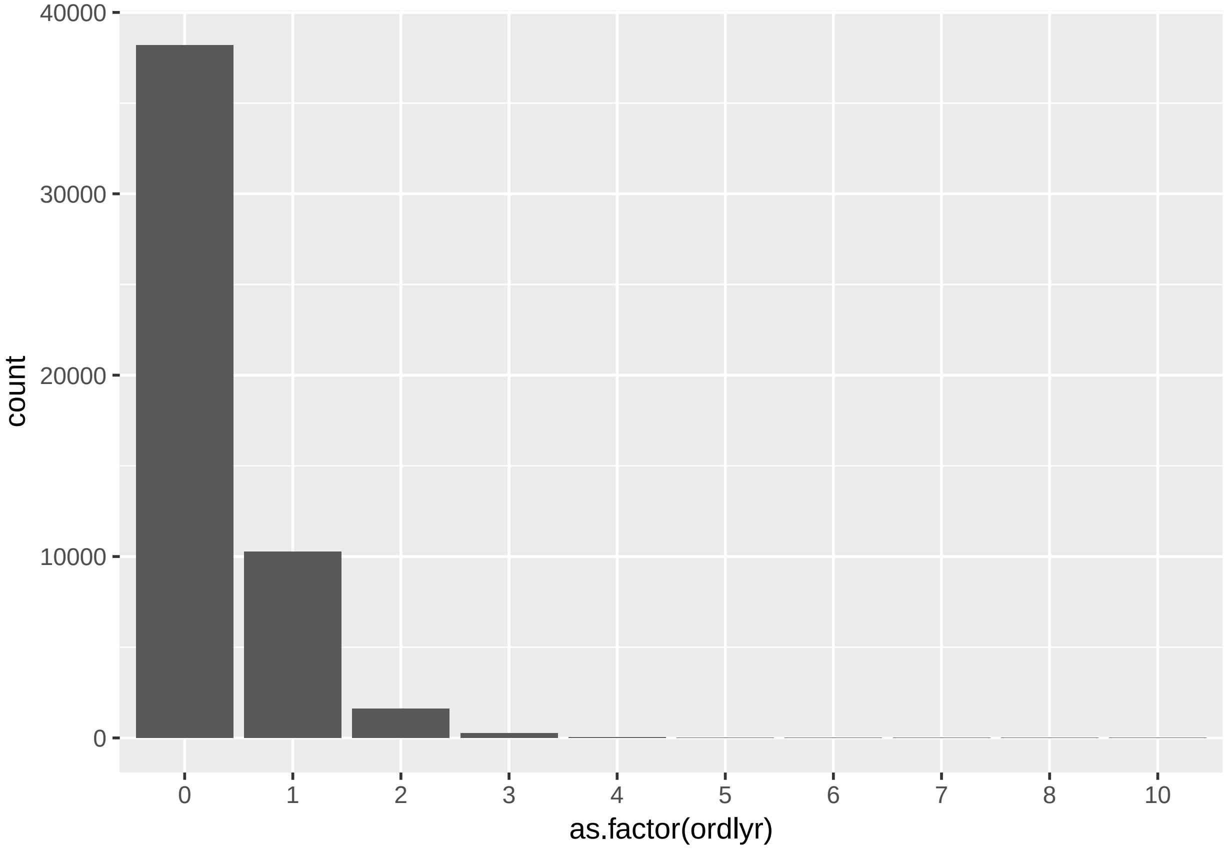
> ord2ago: orders 2 years ago

> ord3ago: orders 3 years ago

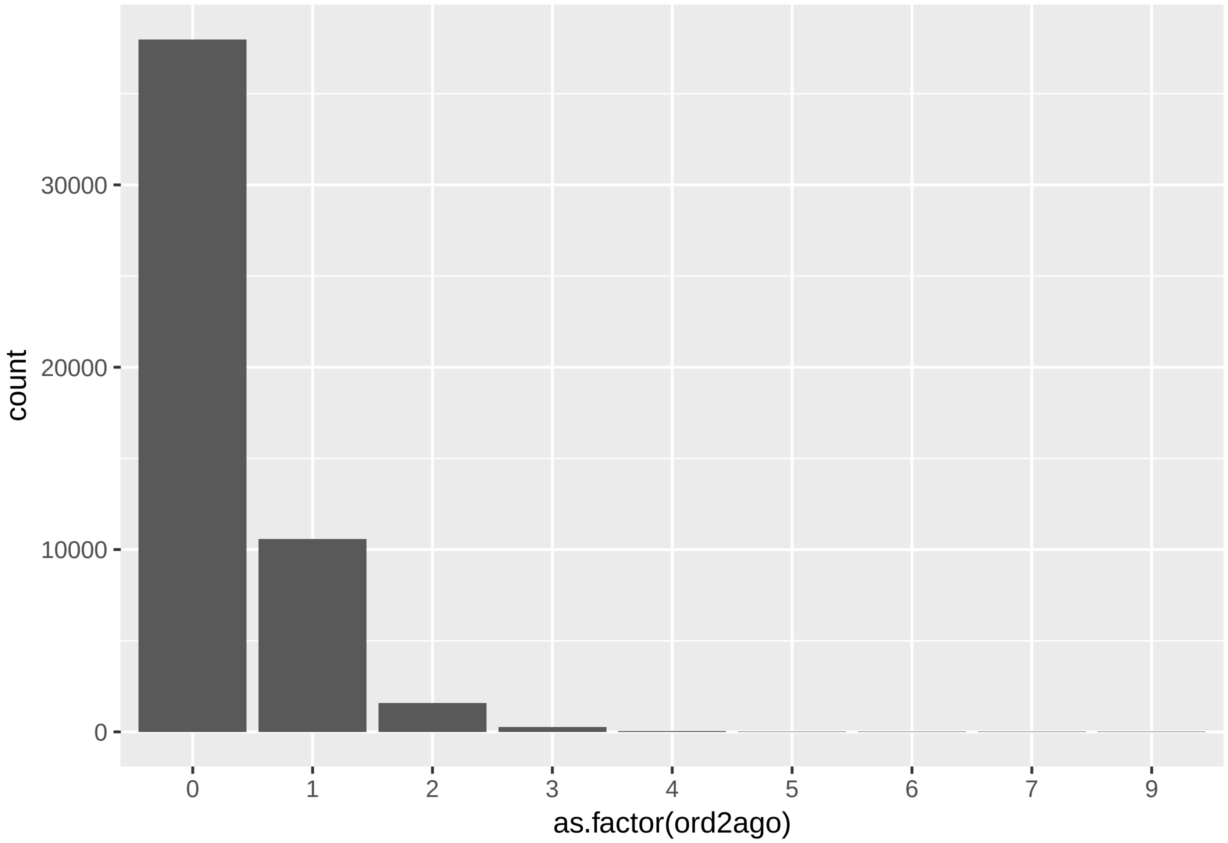
```
ggplot(data=sales_train, aes(x=as.factor(ordtyr)))+
        geom_bar()
```



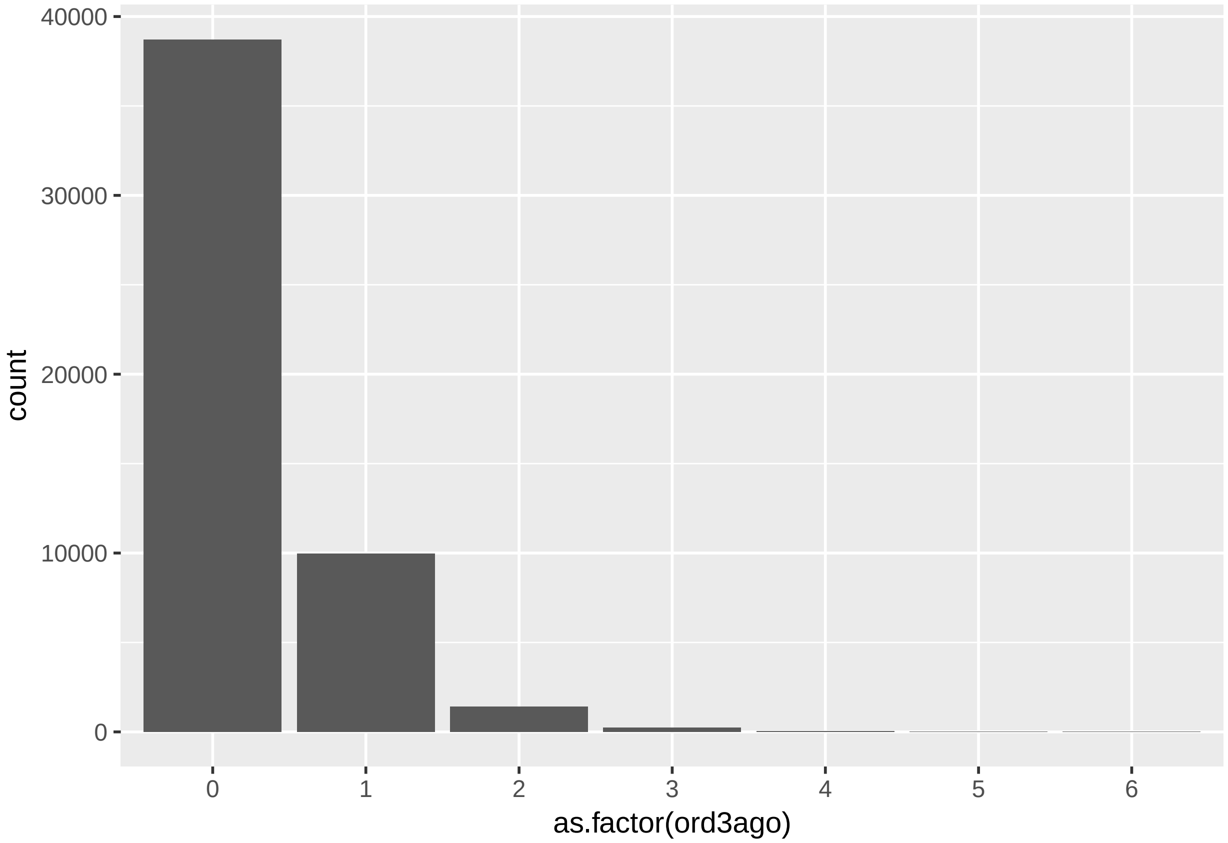
```
ggplot(data=sales_train, aes(x=as.factor(ordlyr)))+
        geom_bar()
```



```
ggplot(data=sales_train, aes(x=as.factor(ord2ago)))+
        geom_bar()
```

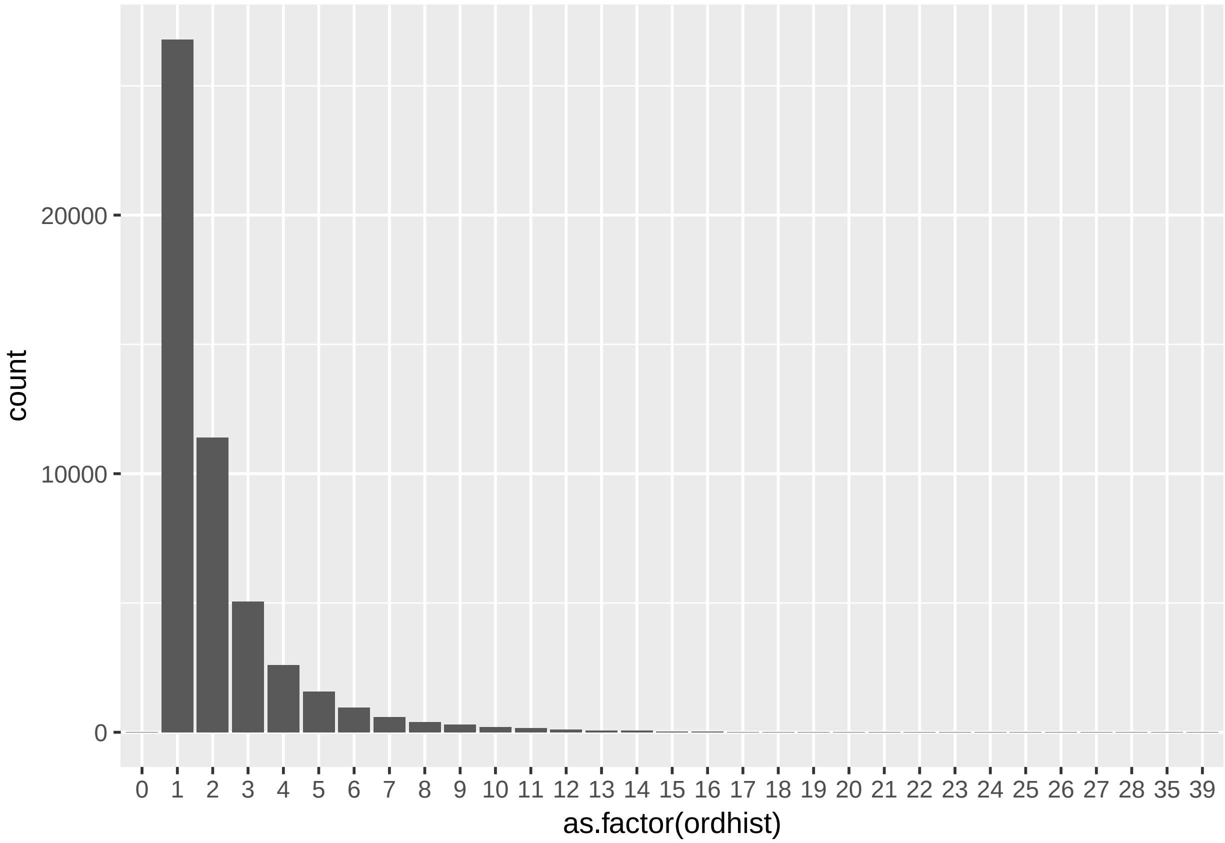


```
ggplot(data=sales_train, aes(x=as.factor(ord3ago)))+
        geom_bar()
```



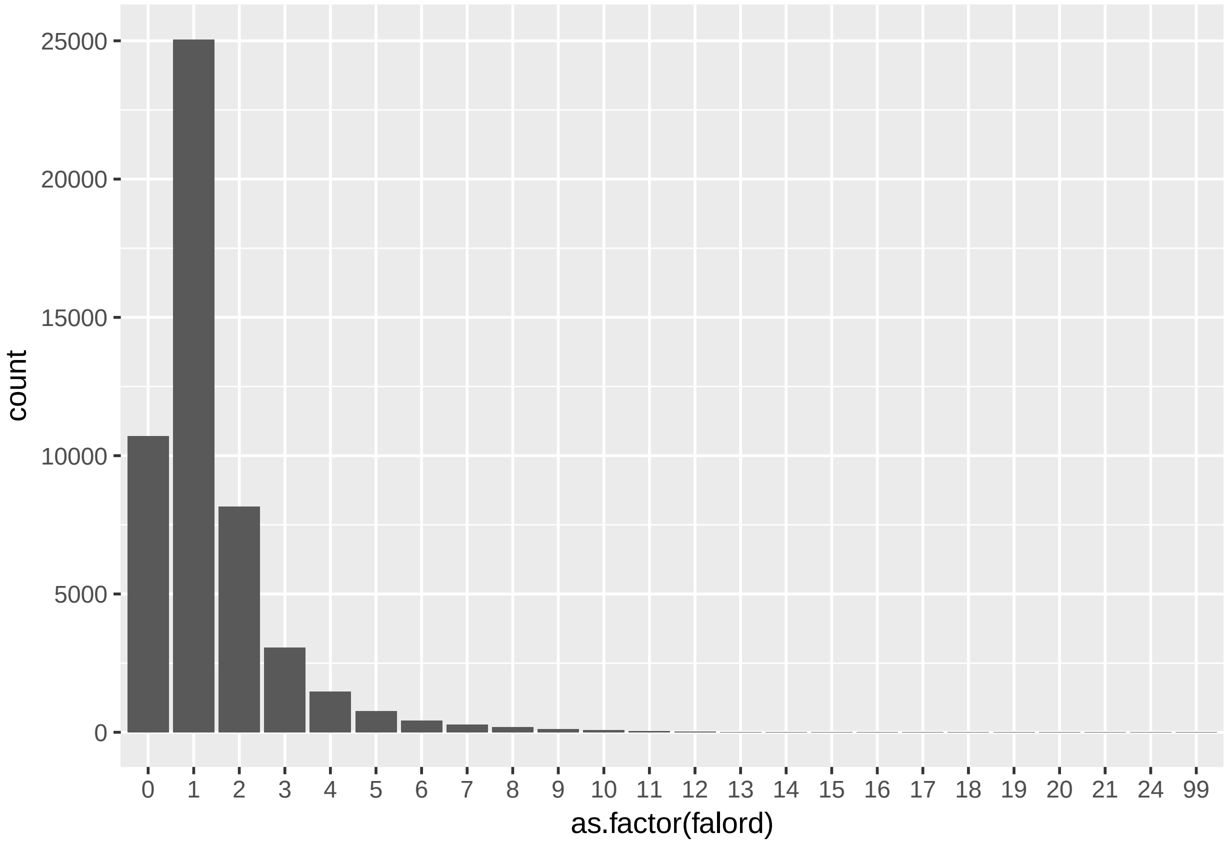
> ordhist: LTD orders

```
ggplot(data=sales_train, aes(x=as.factor(ordhist)))+
        geom_bar()
```



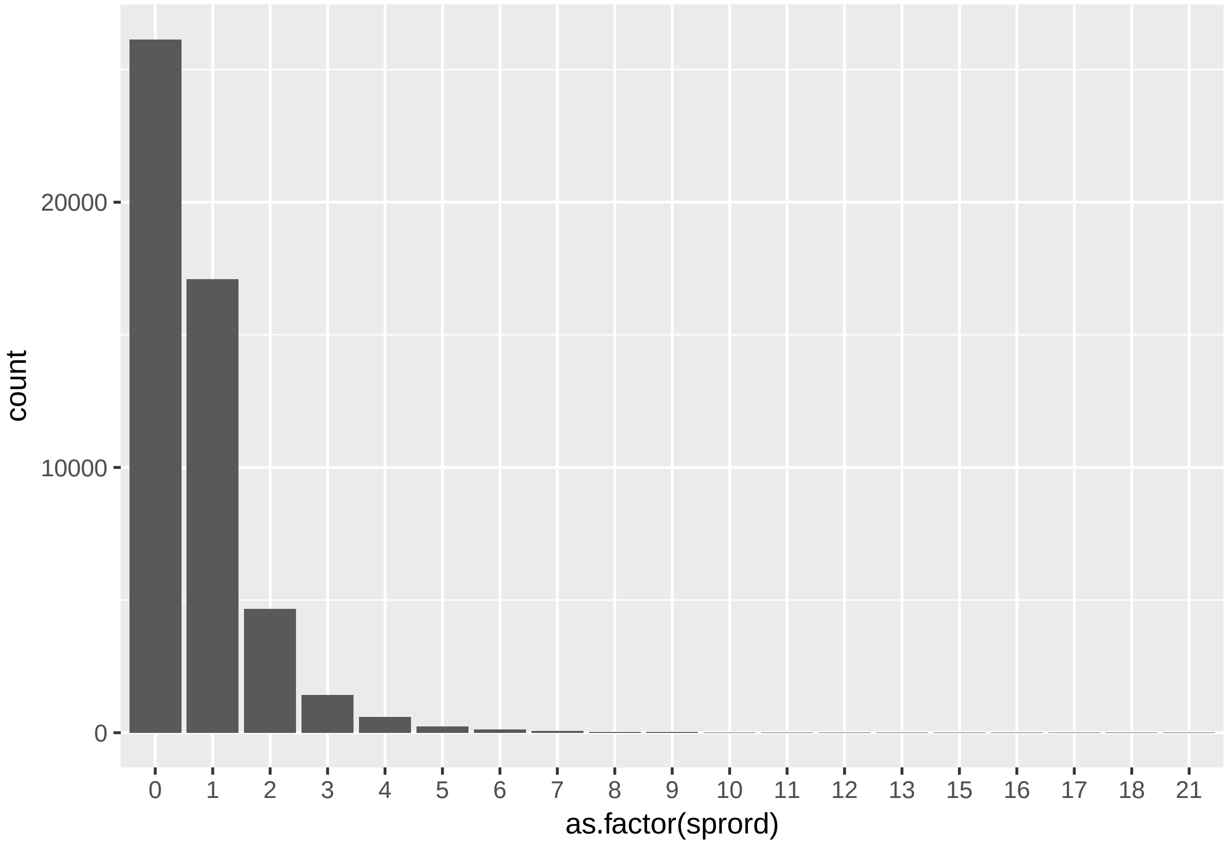
##### > **falord**: LTD fall orders

```
ggplot(data=sales_train, aes(x=as.factor(falord)))+
        geom_bar()
```



##### > **sprord**: LTD spring orders

```
ggplot(data=sales_train, aes(x=as.factor(sprord)))+
        geom_bar()
```



Examine the nature of relationships

```
# plot(sales_train)
```

Check Missing value

```
na_count = data.frame(na_count = colSums(is.na(sales_train)))
na_count$name=row.names(na_count)
na_count[na_count$na_count !=0,]$name
```

```
## [1] "lpuryear"
```

We can see that "lpuryear" is the only column that with NAs.

Detect Outliers

Linearize and Normalize Transformations

```
data_manipulate <- function(sales_train){
  sales_train$log_targdol = log(sales_train$targdol+1)

  sales_train$log_slstyr = log(sales_train$slstyr+1)
  sales_train$log_slslyr = log(sales_train$slslyr+1)
  sales_train$log_sls2ago = log(sales_train$sls2ago+1)
  sales_train$log_sls3ago = log(sales_train$sls3ago+1)
  sales_train$targdol_bol = ifelse(sales_train$targdol!=0, 1, 0)
  return(sales_train)
}

sales_train_2 = data_manipulate(sales_train)
```

Strategy for Building the Prediction Model

1. Based on preliminary analyses, transform the data and include any interactions as appropriate.
2. First develop a binary logistic regression model for targdol > 0. Use this model to estimate the probabilities of being responders for the test set.
3. Next develop a multiple regression model using data with targdol > 0 only.
4. For each observation (including targdol = 0) calculate E(targdol) by multiplying the predicted targdol from the multiple regression model by P(targdol > 0) from the logistic regression model by using the formula E(y) = E(y|y > 0)P(y > 0).

Create Possible Features

```
classificaiton_selected_features = c("log_slstyr", "log_slslyr","log_sls2ago","log_sls3ago",
                      "ordtyr","ordlyr","ord2ago","ord3ago",
                      "ordhist","falord","sprord",
                      "targdol_bol")
```

Classification

Model Fitting

```
fit = glm(targdol_bol ~ ., family=binomial, data=sales_train_2[classificaiton_selected_features])
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fit)
```

```
##
## Call:
## glm(formula = targdol_bol ~ ., family = binomial, data = sales_train_2[classificaiton_selected_featu
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.4083  -0.3171  -0.2742   4.4217
##
## Coefficients:
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -3.48247    0.02963 -117.517  < 2e-16 ***
## log_slstyr   0.26054    0.01954   13.334  < 2e-16 ***
## log_slslyr   0.18160    0.02022    8.983  < 2e-16 ***
## log_sls2ago  0.04156    0.02141    1.941   0.0522 .
## log_sls3ago  0.09248    0.02295    4.030 5.57e-05 ***
## ordtyr       0.06314    0.05620    1.124   0.2612
## ordlyr      -0.02933    0.05569   -0.527   0.5984
## ord2ago      0.13757    0.05830    2.359   0.0183 *
## ord3ago     -0.06990    0.06408   -1.091   0.2754    
## ordhist     -1.66698    0.05012  -33.259  < 2e-16 ***
## falord       1.88769    0.05187   36.392  < 2e-16 ***
## sprord       1.85056    0.04961   37.300  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 31907  on 50417  degrees of freedom
## Residual deviance: 25952  on 50406  degrees of freedom
## AIC: 25976
## 
## Number of Fisher Scoring iterations: 7
```

```
predict1 = predict(fit, newdata=sales_train_2[classificaiton_selected_features], type="response")
predict_response = rep(0,dim(sales_train_2)[1])
predict_response[predict1>0.5]=1

real_response = sales_train_2$targdol_bol

print(table(actual = real_response, prediction = predict_response))
```

```
##       prediction
## actual     0     1
##      0 45228   345
##      1  3947   898
```

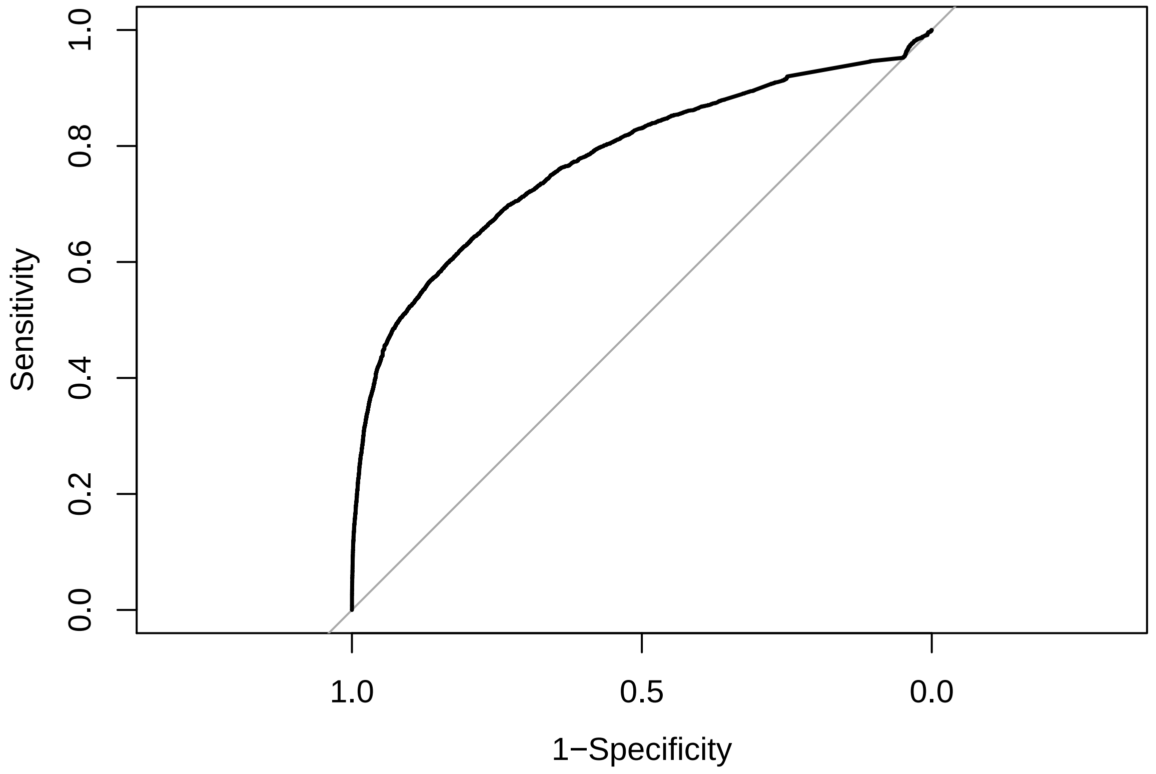
```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
## 
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
## 
##     cov, smooth, var
```

```
plot.roc(real_response, fit$fitted.values, xlab="1-Specificity")
```



```
my_auc = auc(real_response, fit$fitted.values)
```

The AUC for this logistic regression is 0.7763324

Model Diagnostics

How to do it for logistic regression?

Model Selection

Regression

For regression, we will only use the data with targdol > 0 as our training data

```
sales_train_reg = sales_train_2 %>% filter(targdol != 0 )
```

Model Fitting

```
regression_selected_features = c("log_slstyr", "log_slslyr","log_sls2ago","log_sls3ago",
                      "ordtyr","ordlyr","ord2ago","ord3ago",
                      "ordhist","falord","sprord",
                      "log_targdol")
```

> Multiple Linear Regression

```
fit_multiple = lm(log_targdol~.,data=sales_train_reg[regression_selected_features])
summary(fit_multiple)
```

```
##
## Call:
## lm(formula = log_targdol ~ ., data = sales_train_reg[regression_selected_features])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8399 -0.5521 -0.0349  0.5021  3.9431
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.33420    0.02010 165.917  < 2e-16 ***
## log_slstyr   0.09789    0.01191   8.217 2.65e-16 ***
## log_slslyr   0.06458    0.01188   5.434 5.77e-08 ***
## log_sls2ago  0.06061    0.01320   4.593 4.48e-06 ***
## log_sls3ago  0.06933    0.01391   4.985 6.43e-07 ***
## ordtyr      -0.18846    0.03219  -5.855 5.10e-09 ***
## ordlyr      -0.12594    0.03060  -4.115 3.93e-05 ***
## ord2ago     -0.12876    0.03413  -3.772 0.000164 ***
## ord3ago     -0.15988    0.03715  -4.304 1.71e-05 ***
## ordhist     -0.11507    0.01463  -7.865 4.51e-15 ***
## falord       0.13044    0.01576   8.278  < 2e-16 ***
## sprord       0.13330    0.01413   9.435  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7642 on 4833 degrees of freedom
## Multiple R-squared:  0.05259,    Adjusted R-squared:  0.05043
## F-statistic: 24.39 on 11 and 4833 DF,  p-value: < 2.2e-16
```

> Stepwise Regression

> Ridge Regression

> Lasso Regression

Model Diagnostics

> Checking Normality

> Checking Homoscedasticity

> Checking Independence

> Checking Outliers: Deleted residual

> Checking Influential Observations

```
*  Leverage
*  Cook's distance
```

> Checking Multicollinearity: VIF (>10)

Model Selection

Model Validation

```r
model_validation <- function(sales_test, fit_classification, fit_regression, classificaiton_selected_fea
  ## predict sales_test_2 using fit_classification
  sales_test_2 = data_manipulate(sales_test)
  predict_classification_final = predict(fit, newdata=sales_test_2[classificaiton_selected_features], ty

  ## keep the data with targdol prob > 0.5, save as sales_test_reg
  sales_test_2$targdol_bol_predict = predict_classification_final
  sales_test_2$uid = row.names(sales_test_2)
  sales_test_reg = sales_test_2 %>% filter(targdol_bol_predict>0.5)

  ## predict sales_test_reg using fit_regression, and take exp(log_targdon) to recover back to real meas
  predict_regression_log_final = predict(fit_multiple, newdata=sales_test_reg[regression_selected_featu
  predict_regression_final = exp(predict_regression_log_final)
  sales_test_reg$targdol_predict = predict_regression_final

  ## generate a data frame with original and predicted value, save as sales_test_final
  sales_test_final = merge(sales_test_2 %>% select(uid, targdol), sales_test_reg %>% select(uid, targdol
  sales_test_final[is.na(sales_test_final)]=0

  ## calculate MSPE & top_1000
  calculate_MSPE <- function(actual, predicted){
    return(mean((actual-predicted) ^ 2))
  }

  calculate_top1000 <- function(actual, predicted, by_predicted=TRUE){
    df = data.frame(actual=actual, predicted=predicted)
    if (by_predicted){
      df = df %>% arrange(desc(predicted))
    }else{
      df = df %>% arrange(desc(actual))
    }
    return(sum(df[1:1000,]$actual))
  }

  mspe = calculate_MSPE(sales_test_final$targdol, sales_test_final$targdol_predict)
  top1000 = calculate_top1000(sales_test_final$targdol, sales_test_final$targdol_predict)
  actual_top1000 = calculate_top1000(sales_test_final$targdol, sales_test_final$targdol_predict, by_pred

  return(list(mspe=mspe, top1000=top1000, actual_top1000=actual_top1000))
}
```

```
model_validation(sales_test, fit_classification, fit_regression, classificaiton_selected_features, regre
```

```
## $mspe
## [1] 5650733624
##
## $top1000
## [1] 49609.43
##
## $actual_top1000
## [1] 120252.4
```

1. Cover page (Title, names of group members)
2. Executive Summary: Give a non-technical summary of your findings mentioning the key predictors of responders vs. non-responders and of the amount of sales. This summary should not include any equations and as few statistics as possible. (About 1/2 page)
3. Introduction: Describe your overall approach and any a priori hypotheses. Give a brief outline of the other sections of the report. (About 2 pages)
4. Model Fitting: This is the core of the report. Divide this into two parts: (i) classification model, (ii) multiple regression model. Explain the steps used in model fitting including exploratory analysis of data to assess the nature of relationships, detection of outliers and influential observations, linearizing and normalizing transformations etc.; different models fitted and methods used to fit them (e.g., stepwise regression); model diagnostics. The final model including residual analyses and other diagnostics resulting to data transformations. (10 pages)
5. Model Validation: Explain how you validated the model against the test data set. Report the results about how well the model predicted the test set sales values and how well your top 1,000 predicted customers from the test set performed in terms of actual sales. (2 pages)
6. Conclusions: Draw conclusions about significant predictors, any key missing predictors which would have improved the model, etc. (1 page)
7. References