HR Analytics

```
#Preliminary report
#Abhishek Kumar.

#few assumptions for Logistic Regression:
#  no multicollinearity among the independent variables
#  observations to be independent of each other
#  dependent variable to be binary and ordinal
#  logistic regression requires the dependent variable to be ordinal.
#  linearity of independent variables and log odds
#  requires a large sample size.  A general guideline is that you need
#at minimum of 10 cases with the least frequent outcome for each
#independent variable in your model. For example, if you have 5
#independent variables and the expected probability of your least
#frequent outcome is .10, then you would need a minimum sample size of
#500 (10*5 / .10).

#setting the directory location
setwd("G://analytics//ISB CBA//Residency//Residency3&4//B9MT2SA2//Mini Project  Data sets-20180206//HR A

#loading required library
library(car)
```

```
## Warning: package 'car' was built under R version 3.4.3
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.3
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 2.2.1     v purrr   0.2.4
## v tibble  1.4.2     v dplyr   0.7.4
## v tidyr   0.8.0     v stringr 1.3.0
## v readr   1.1.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.4.1
```

```
## Warning: package 'tibble' was built under R version 3.4.3
```

```
## Warning: package 'tidyr' was built under R version 3.4.3
```

```
## Warning: package 'readr' was built under R version 3.4.2
```

```
## Warning: package 'purrr' was built under R version 3.4.3
```

```
## Warning: package 'dplyr' was built under R version 3.4.3
```

```
## Warning: package 'stringr' was built under R version 3.4.3
```

```
## Warning: package 'forcats' was built under R version 3.4.3
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::recode() masks car::recode()
## x purrr::some()   masks car::some()
```

```r
library(ggplot2)



#Reading the Hr file.
hrdata = read.csv("HR.csv",header = TRUE,stringsAsFactors = FALSE,na.strings = c("NA","N/A"," "))
#View(hrdata)
unique(hrdata$sales)
```

```
##  [1] "sales"       "accounting"  "hr"          "technical"   "support"
##  [6] "management"  "IT"          "product_mng" "marketing"   "RandD"
```

```r
str(hrdata)
```

```
## 'data.frame':    14999 obs. of  10 variables:
##  $ satisfaction_level   : num  0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89 0.42 ...
##  $ last_evaluation      : num  0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1 0.53 ...
##  $ number_project       : int  2 5 7 5 2 2 6 5 5 2 ...
##  $ average_montly_hours : int  157 262 272 223 159 153 247 259 224 142 ...
##  $ time_spend_company   : int  3 6 4 5 3 3 4 5 5 3 ...
##  $ Work_accident        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ left                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ promotion_last_5years: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ sales                : chr  "sales" "sales" "sales" "sales" ...
##  $ salary               : chr  "low" "medium" "medium" "low" ...
```

```r
class(hrdata)
```

```
## [1] "data.frame"
```

```r
summary(hrdata)
```

```
##  satisfaction_level last_evaluation  number_project  average_montly_hours
##  Min.   :0.0900     Min.   :0.3600   Min.   :2.000   Min.   : 96.0
##  1st Qu.:0.4400     1st Qu.:0.5600   1st Qu.:3.000   1st Qu.:156.0
##  Median :0.6400     Median :0.7200   Median :4.000   Median :200.0
##  Mean   :0.6128     Mean   :0.7161   Mean   :3.803   Mean   :201.1
##  3rd Qu.:0.8200     3rd Qu.:0.8700   3rd Qu.:5.000   3rd Qu.:245.0
##  Max.   :1.0000     Max.   :1.0000   Max.   :7.000   Max.   :310.0
##  time_spend_company Work_accident         left
##  Min.   : 2.000     Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 3.000     1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 3.000     Median :0.0000   Median :0.0000
##  Mean   : 3.498     Mean   :0.1446   Mean   :0.2381
##  3rd Qu.: 4.000     3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :10.000     Max.   :1.0000   Max.   :1.0000
##  promotion_last_5years    sales              salary
##  Min.   :0.00000       Length:14999       Length:14999
##  1st Qu.:0.00000       Class :character   Class :character
##  Median :0.00000       Mode  :character   Mode  :character
##  Mean   :0.02127
##  3rd Qu.:0.00000
##  Max.   :1.00000
```

```r
#time_spend_company has value 10 hours which we will check if it is outlier or not.
#WE can also check if we can build another model considering who will get the
#promotion in five years now
```

```r
#we will do further investigation on sales and salary by converting
# them into dummy variables

#checking for missing value if any present in the data
colSums(is.na(hrdata))
```

```
##    satisfaction_level       last_evaluation        number_project
##                     0                     0                     0
##  average_montly_hours    time_spend_company         Work_accident
##                     0                     0                     0
##                  left promotion_last_5years                 sales
##                     0                     0                     0
##                salary
##                     0
```

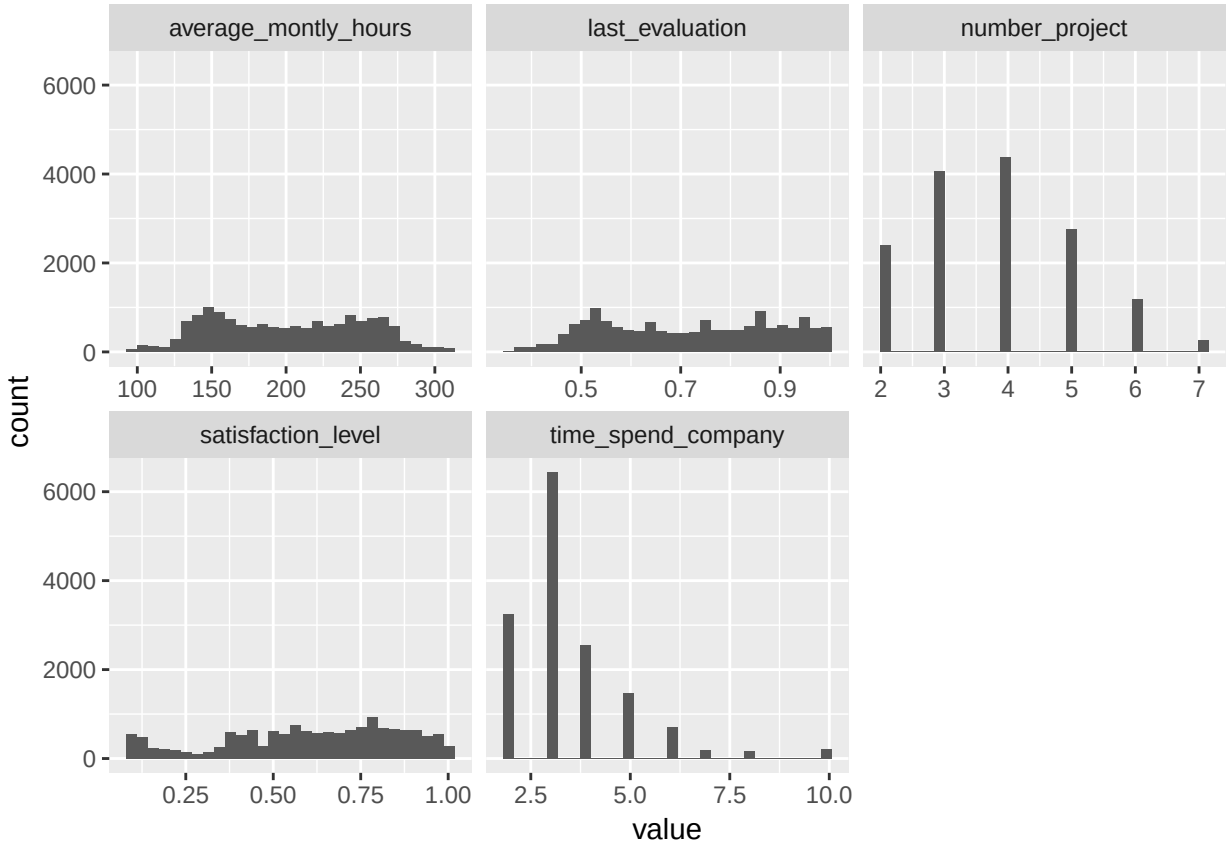
```r
sum(is.na(hrdata)) # no missing value in the data
```

```
## [1] 0
```

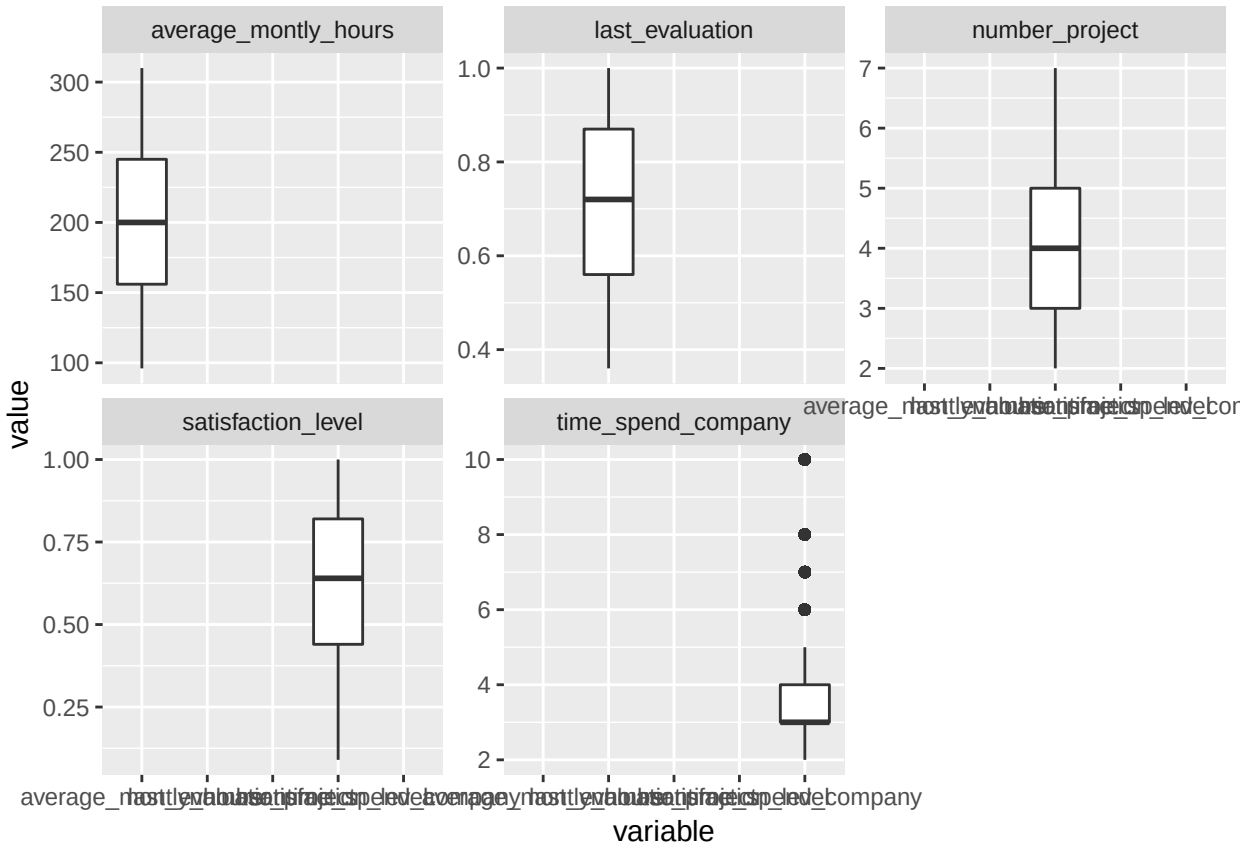
```r
nrow(hrdata)
```

```
## [1] 14999
```

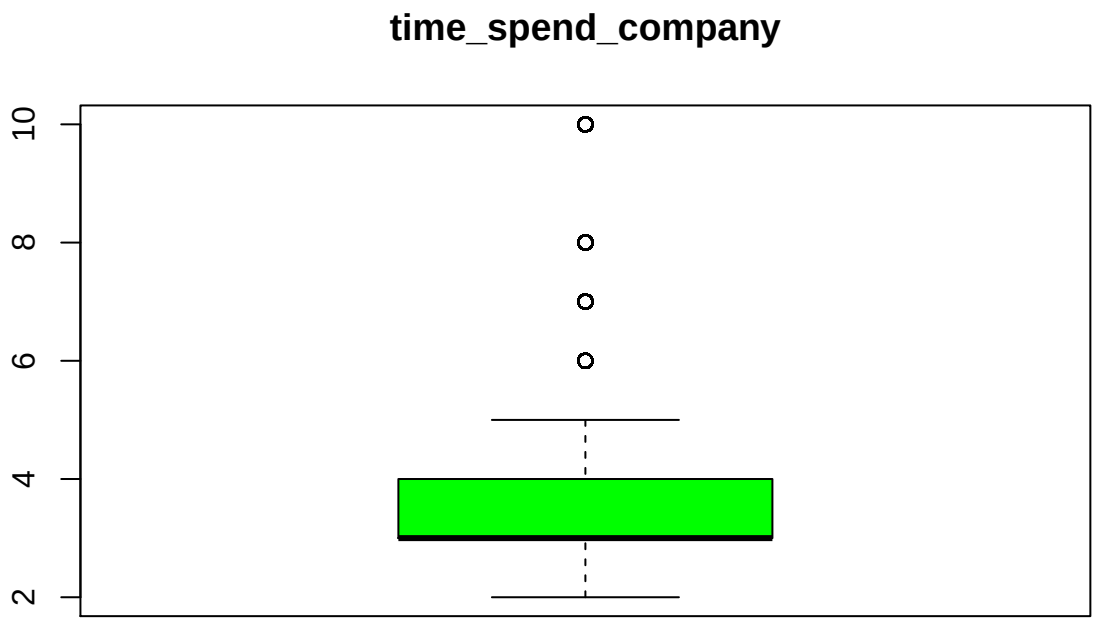
```r
#seperating the quantitative and qualitative data

hrquant = c("satisfaction_level","last_evaluation","number_project"
            ,"average_montly_hours","time_spend_company")
hrquality = c("Work_accident"
              ,"left","promotion_last_5years","sales","salary")

#only quantitative data
hrdataquant =hrdata[,-c(6:10)]
#View(hrdataquant)

#Qualitative analysis


##Univariate analysis on continous data
hrdata %>% gather(satisfaction_level:time_spend_company, key = "variable", value = "value") %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 30) + facet_wrap(~ variable, scales = 'free_x')
```



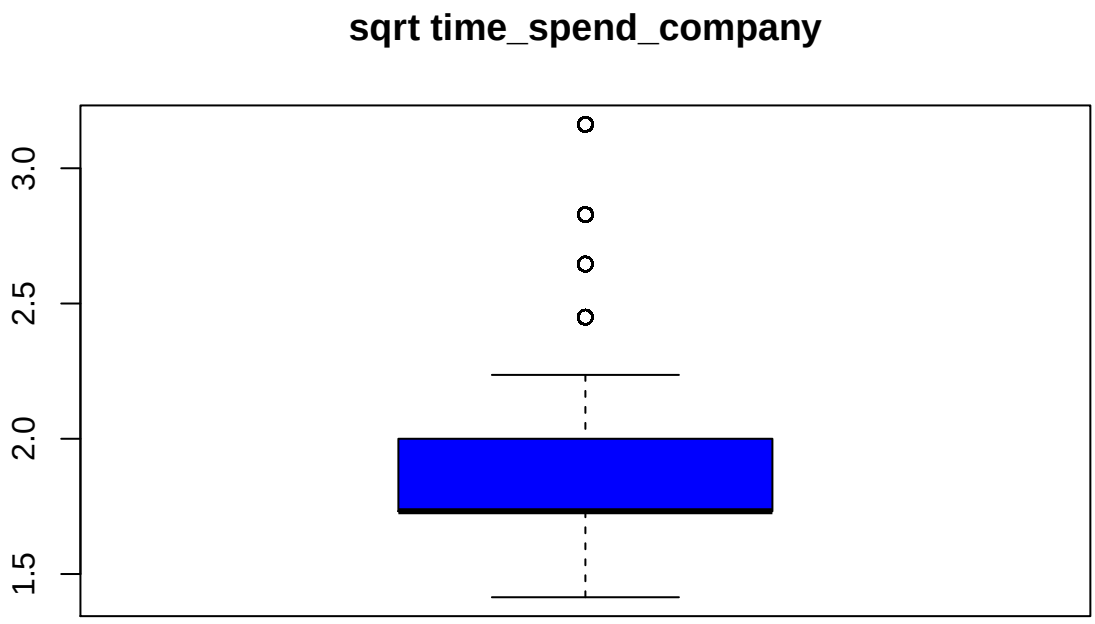

```
hrdata %>% gather(satisfaction_level:time_spend_company, key = "variable", value = "value") %>%
  ggplot(aes(x=variable,y = value)) +
  geom_boxplot() + facet_wrap(~ variable, scales = 'free_y')
```



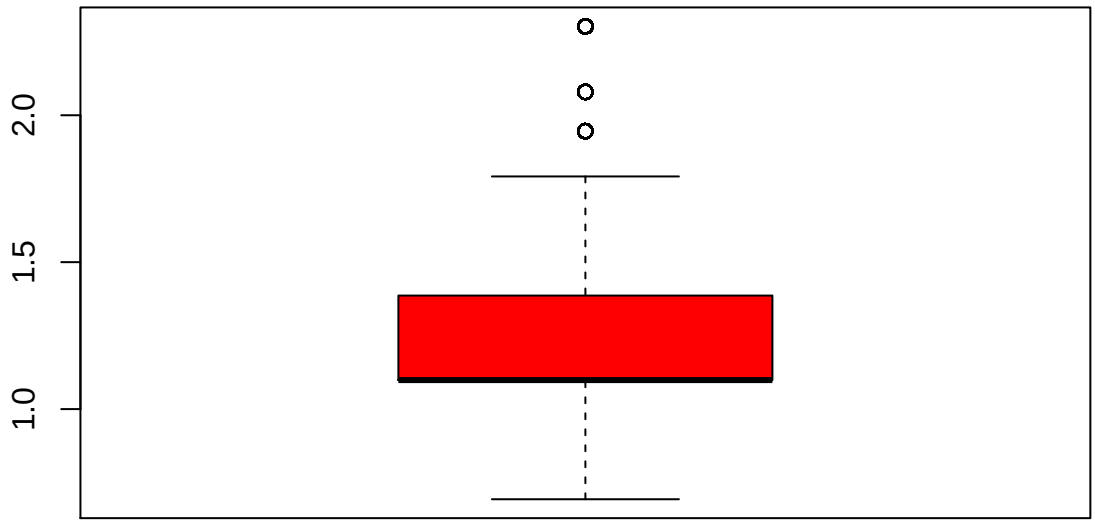

```
#time_spend_company is right skewed with outliers.

boxplot(hrdata$time_spend_company, main="time_spend_company", col ="green")
```




```
boxplot(sqrt(hrdata$time_spend_company), main="sqrt time_spend_company", col ="blue")
```

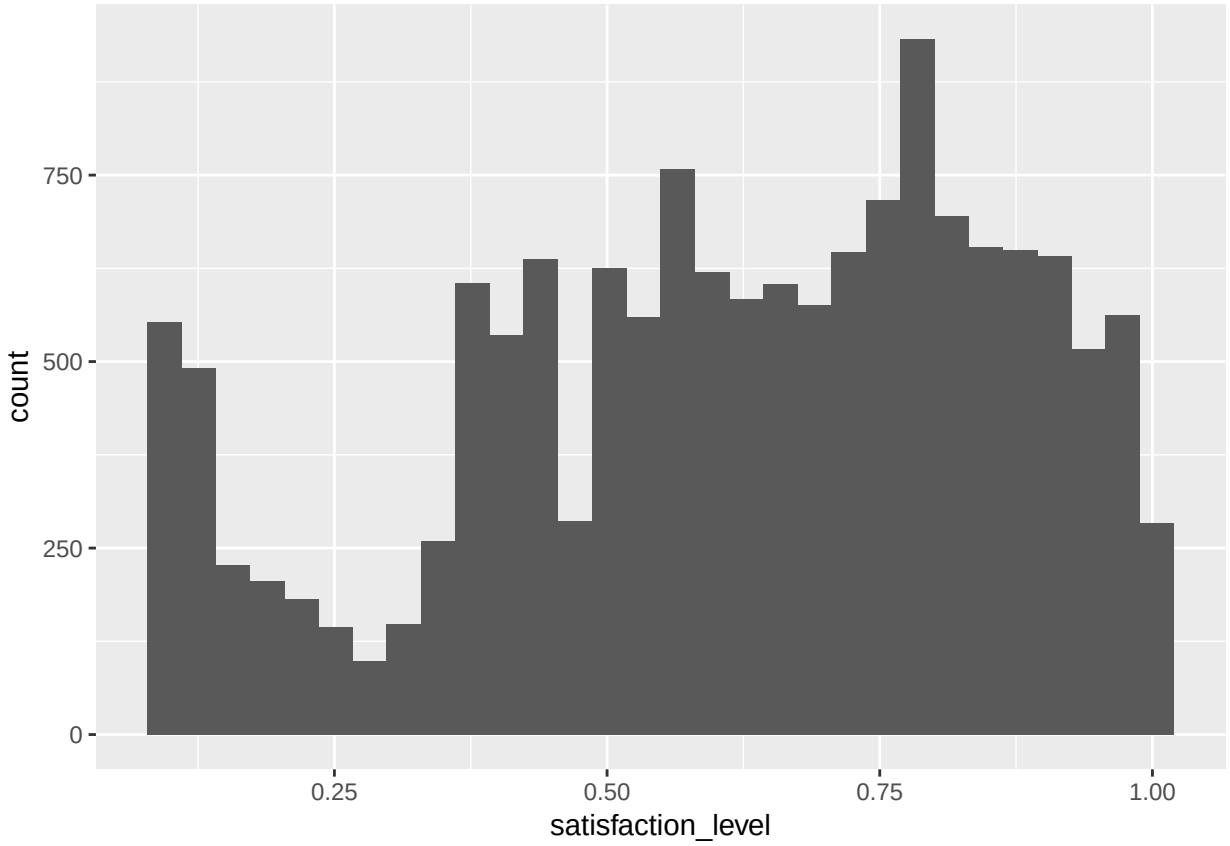



```
boxplot(log(hrdata$time_spend_company), main="log time_spend_company", col ="red")
```

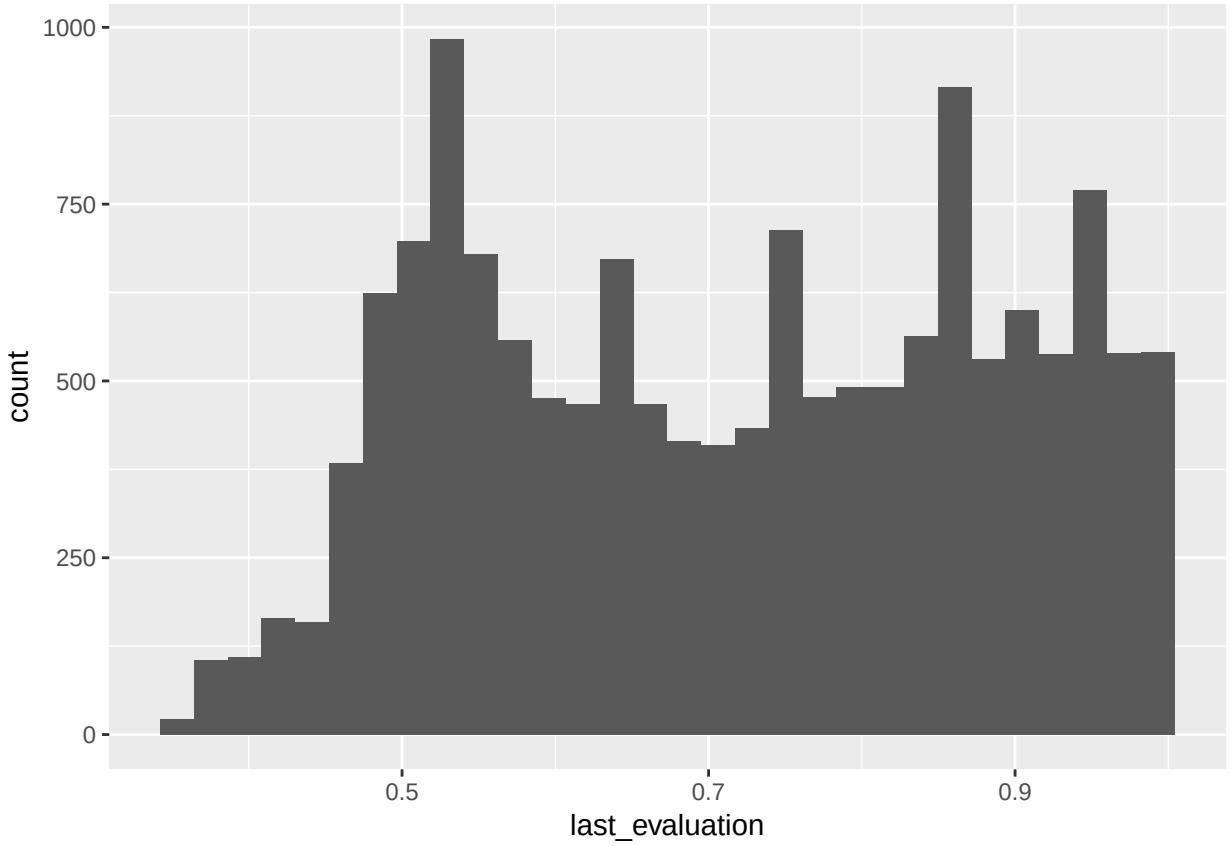



```
#we will see later on if we require the transformed data to build better
# logistic models, as of now going with the original data.

#individual histogram
for(i in 1:ncol(hrdataquant))
{ if(!(is.factor(hrdataquant[,i])))
{
  print(ggplot(hrdataquant,aes(hrdataquant[,i]))+geom_histogram()+labs(x=colnames(hrdataquant)[i]))

}
}
```

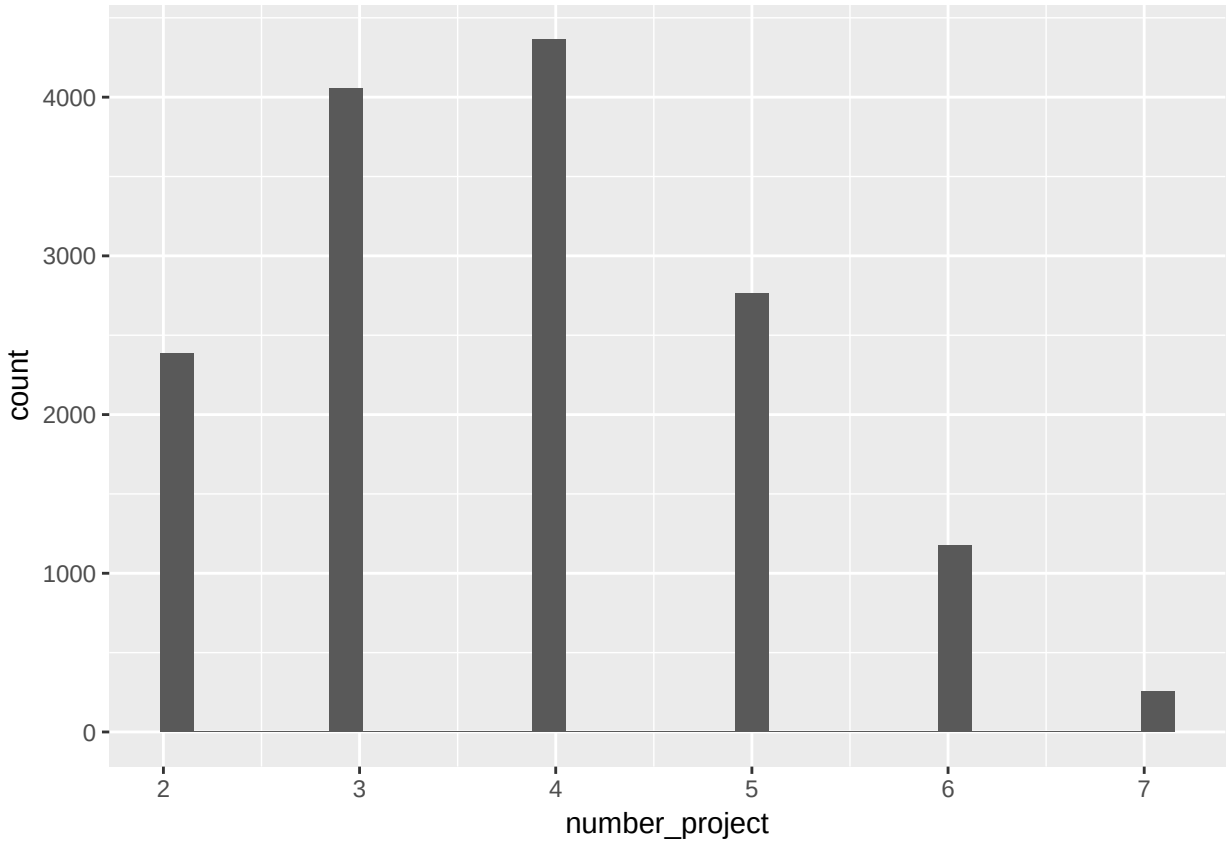
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



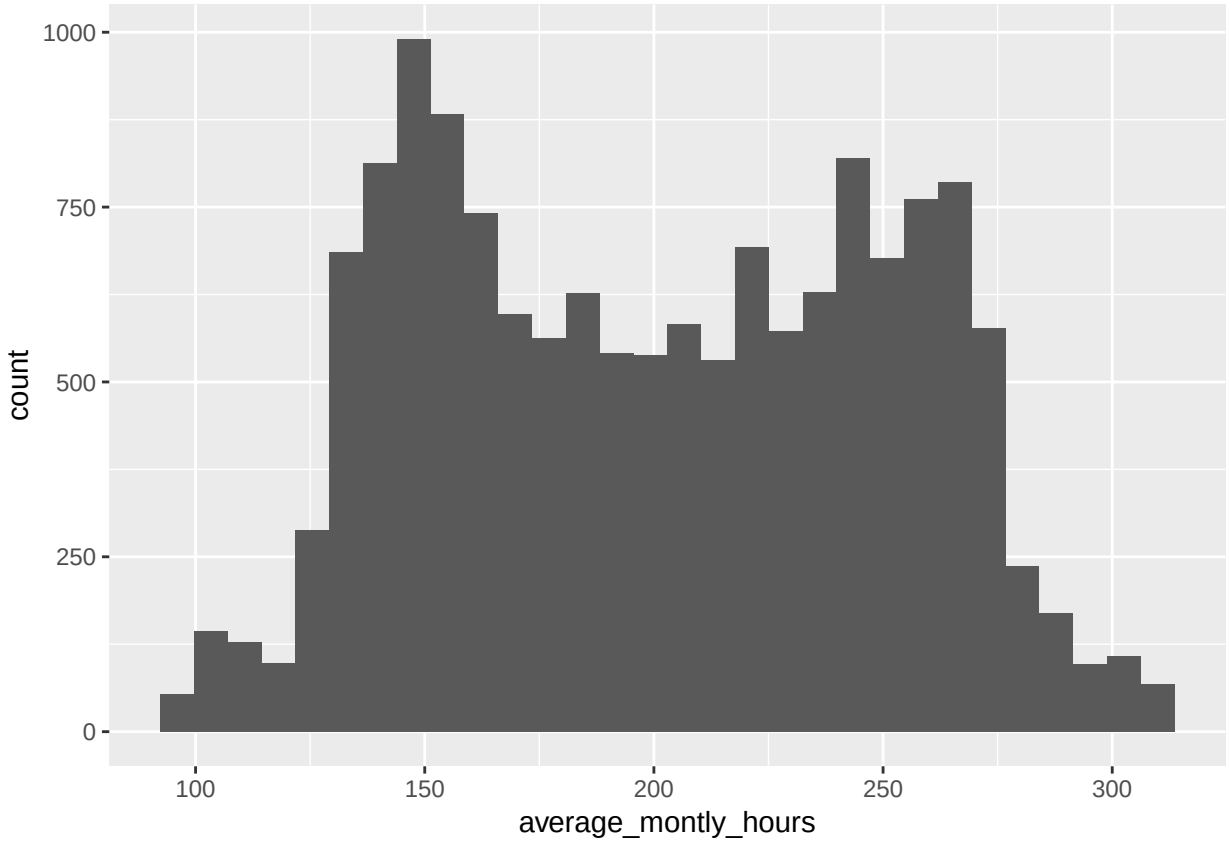

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



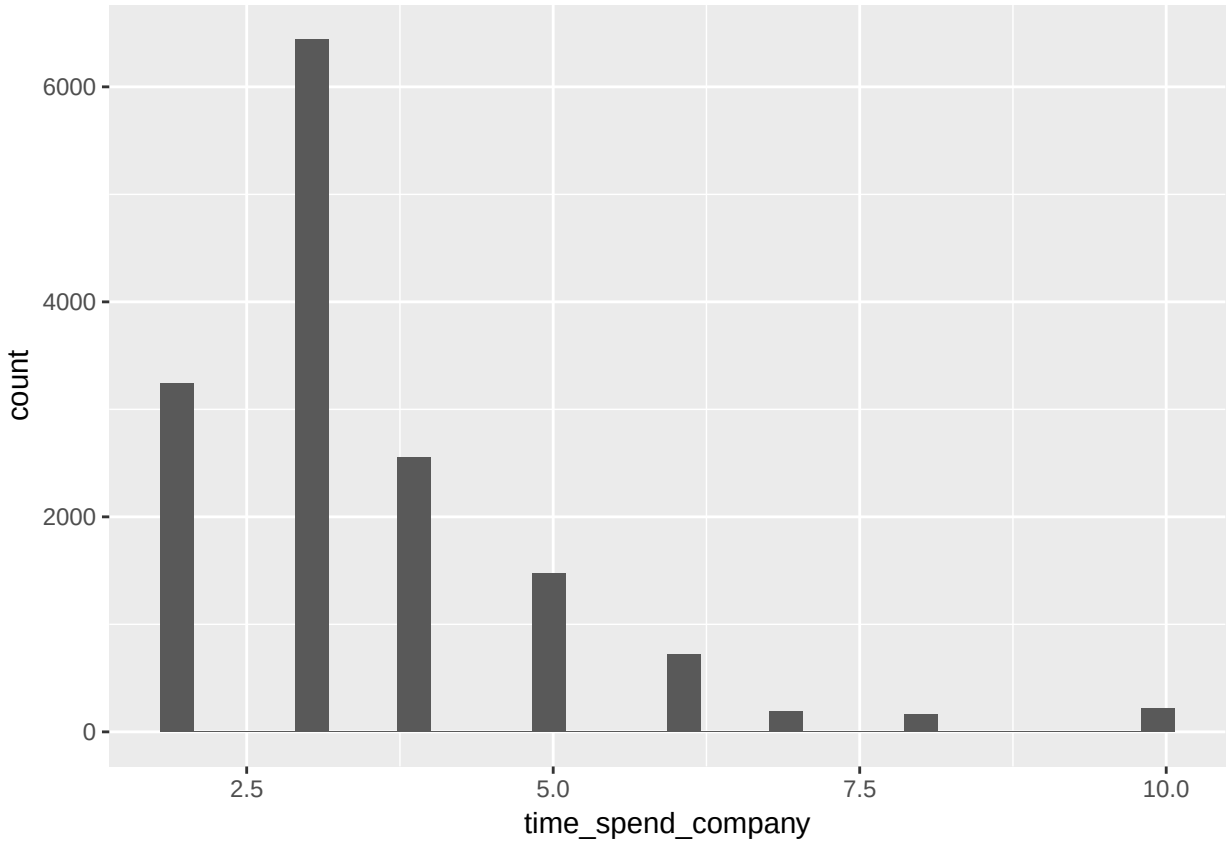

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



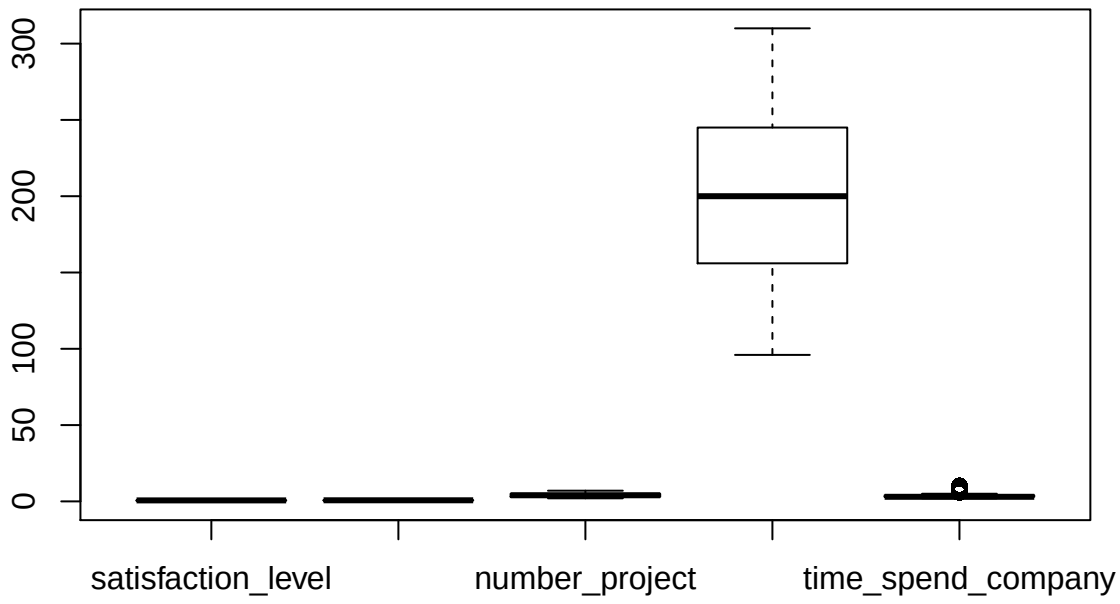

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




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




```
#in one sheet to give better presentation of data
boxplot(hrdataquant)
```




```
## Qualitative variables
# table shows the frequency of categories in the dependent variable
table(hrdata$left)
```

```
##
##     0     1
## 11428  3571
```

```
table(hrdata$promotion_last_5years)
```

```
##
##     0     1
## 14680   319
```

```
table(hrdata$salary)
```

```
##
##   high    low medium
##   1237   7316   6446
```

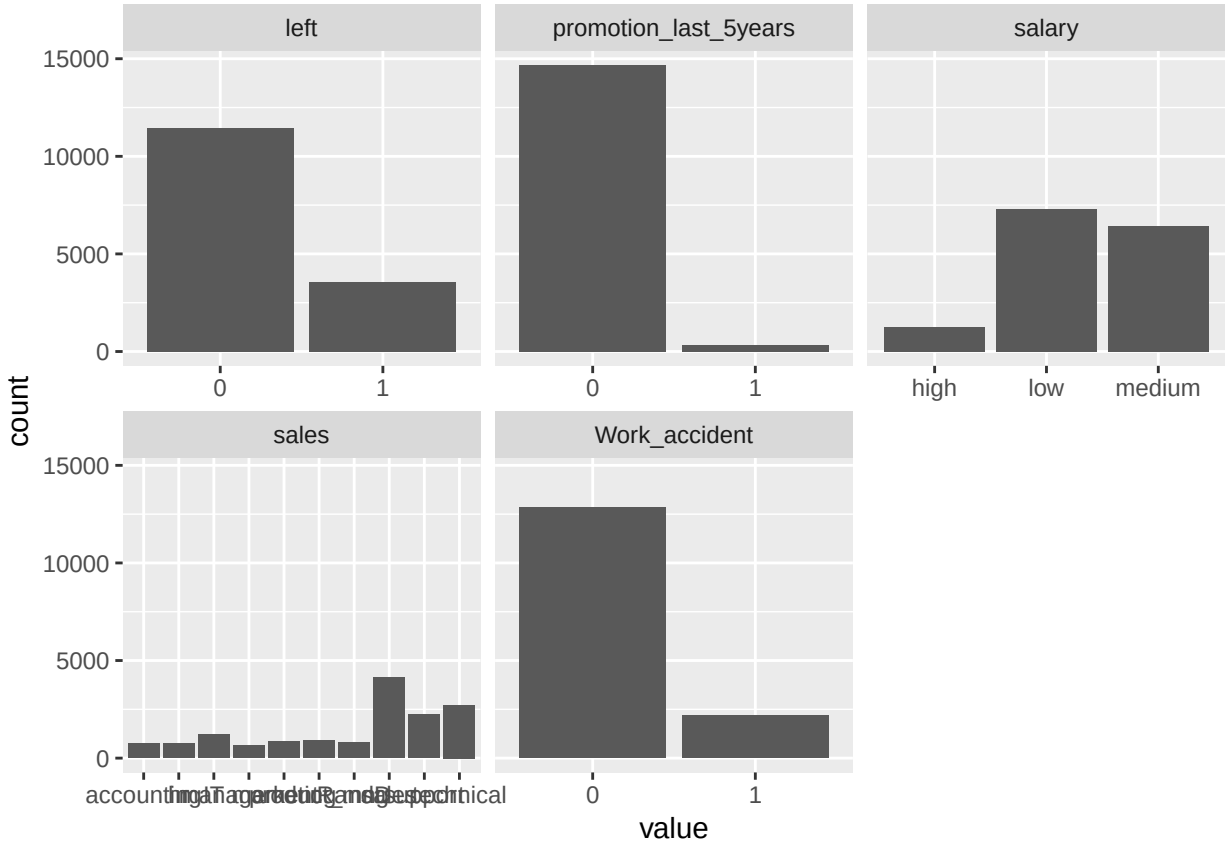
```
table(hrdata$sales)
```

```
##
##  accounting          hr          IT  management   marketing product_mng
##         767         739        1227         630         858         902
##       RandD       sales     support   technical
##         787        4140        2229        2720
```

```
table(hrdata$Work_accident)
```

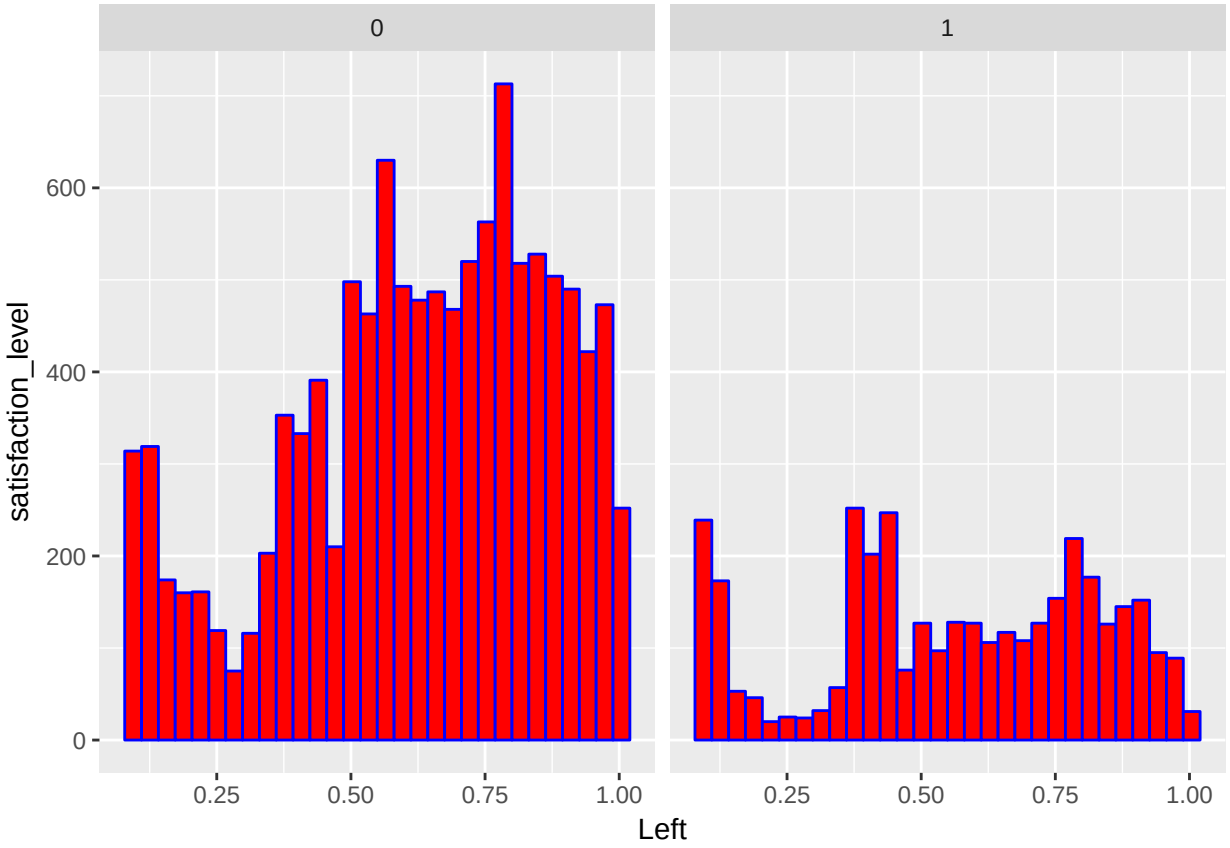
```
##
##     0     1
## 12830  2169
```

```
hrdata %>% gather(Work_accident:salary, key = "variable", value = "value") %>% ggplot(aes(x = value)) +
  geom_bar()+ facet_wrap(~variable, scales = 'free_x')
```

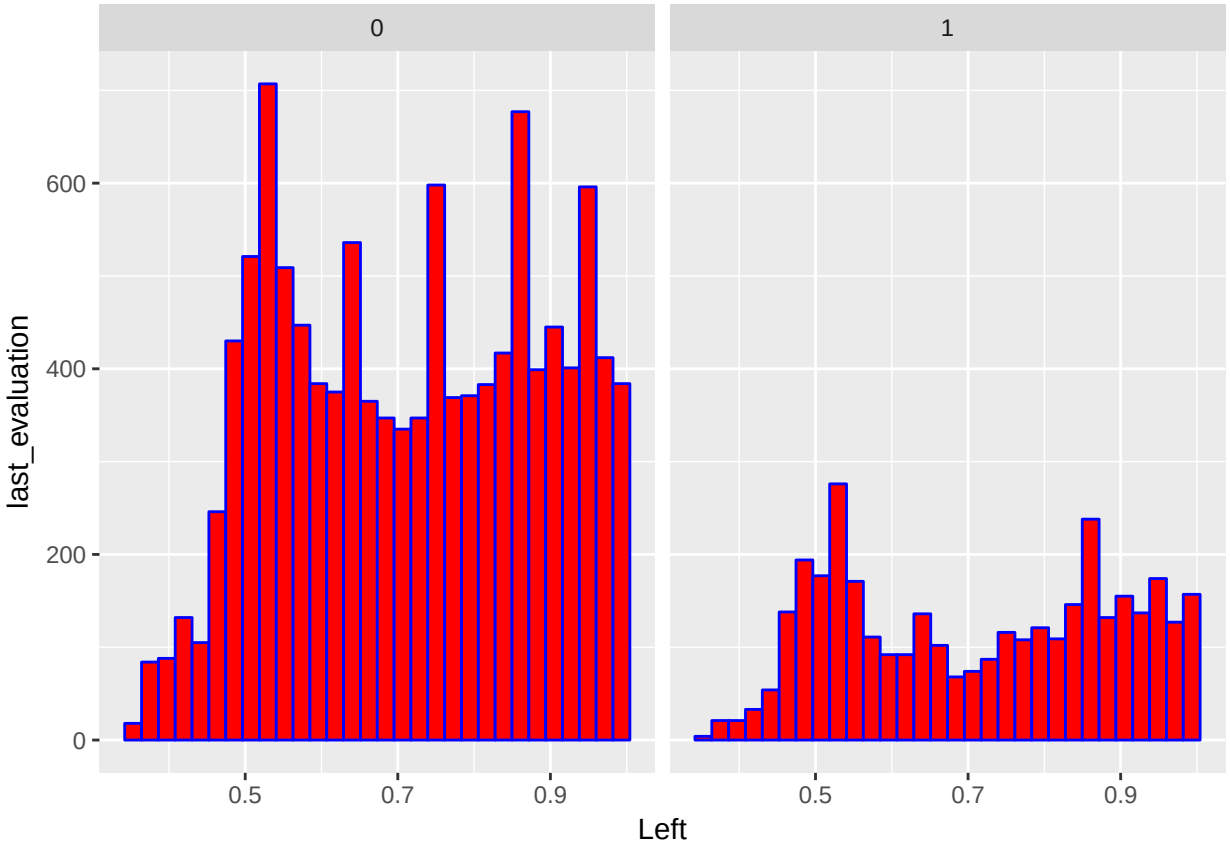



```
for(i in 1:ncol(hrdataquant))
{ if(!(is.factor(hrdataquant[,i])))
{
print(qplot(x=hrdataquant[,i],data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~left)+labs(x="
}
}
```

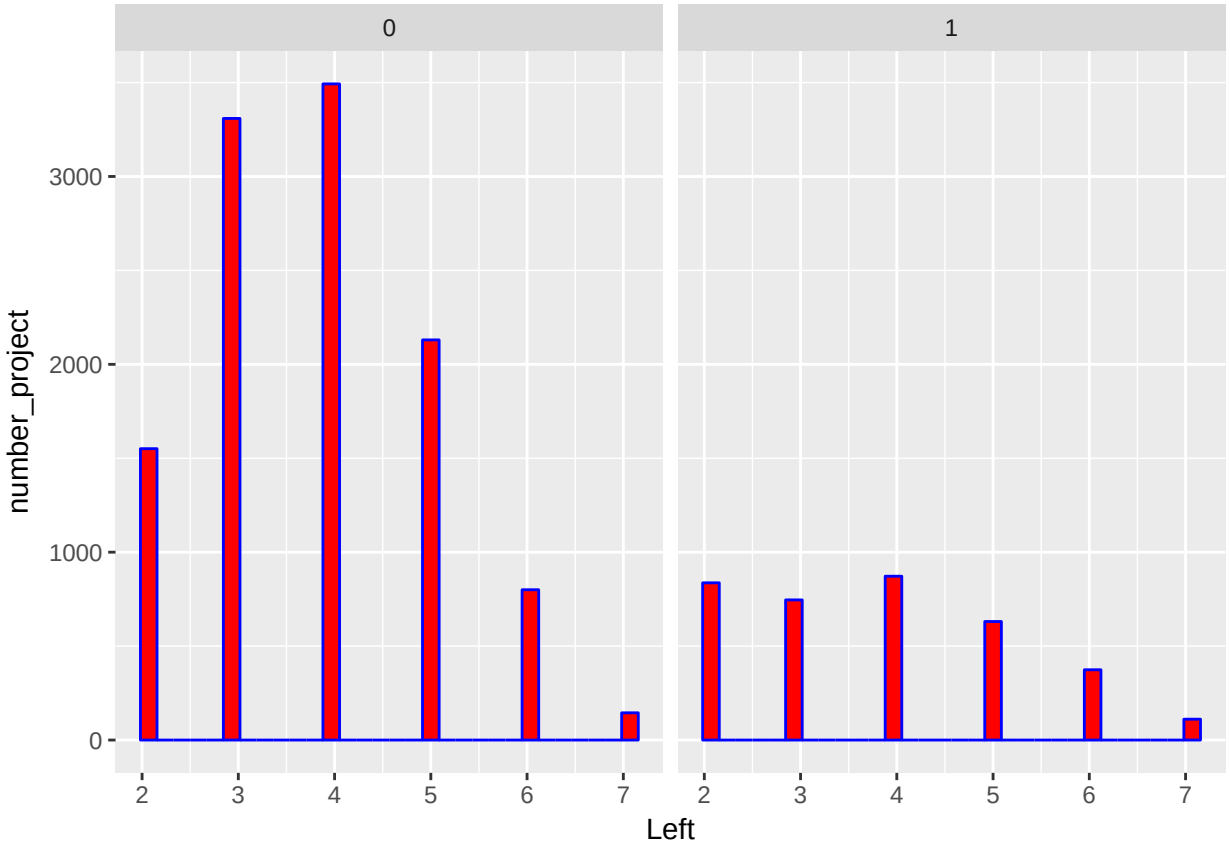
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



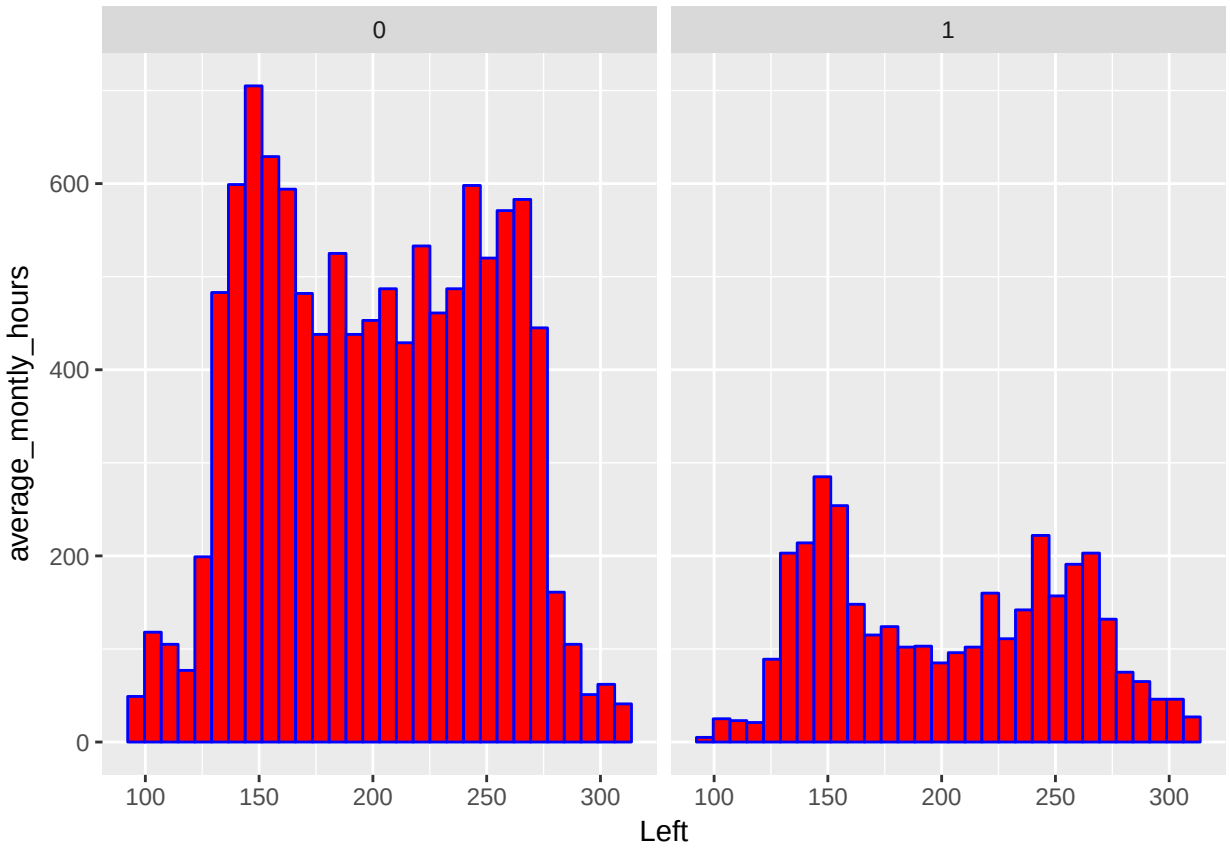

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



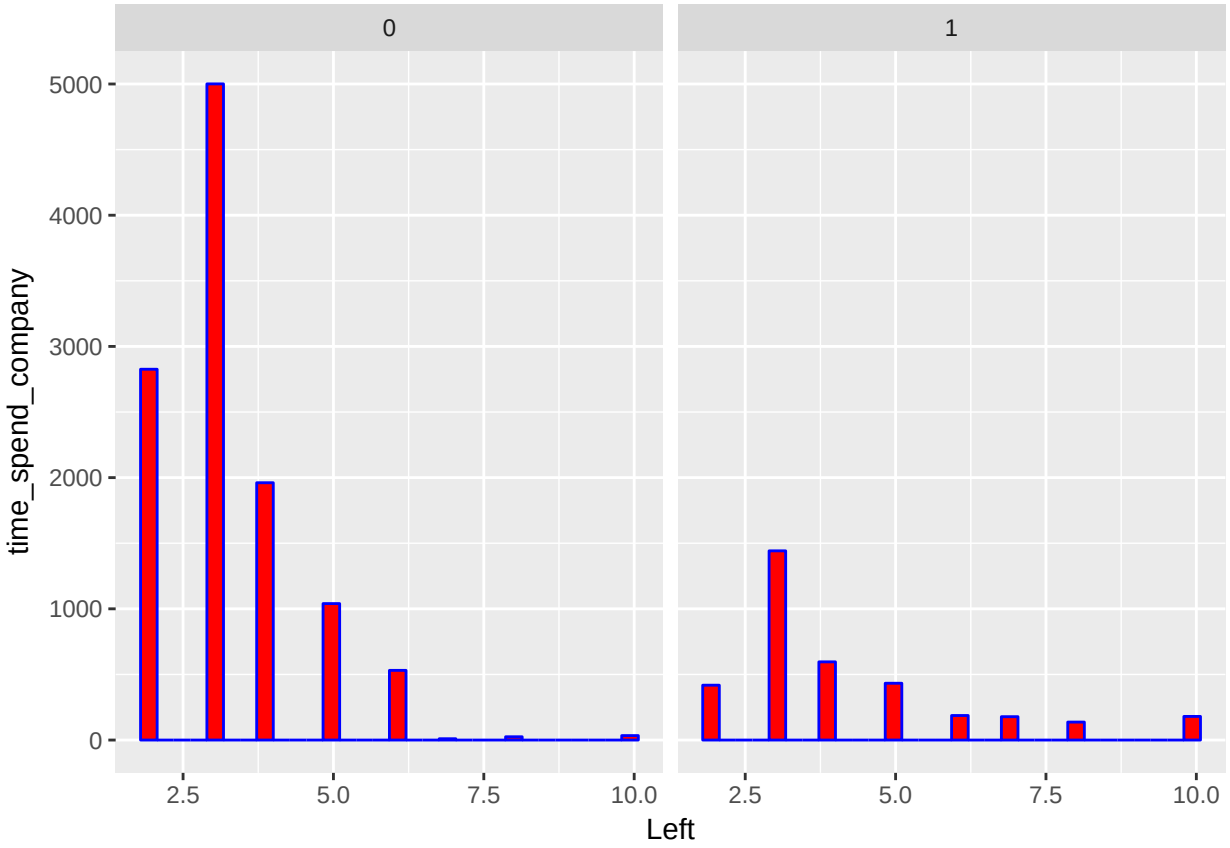

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




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

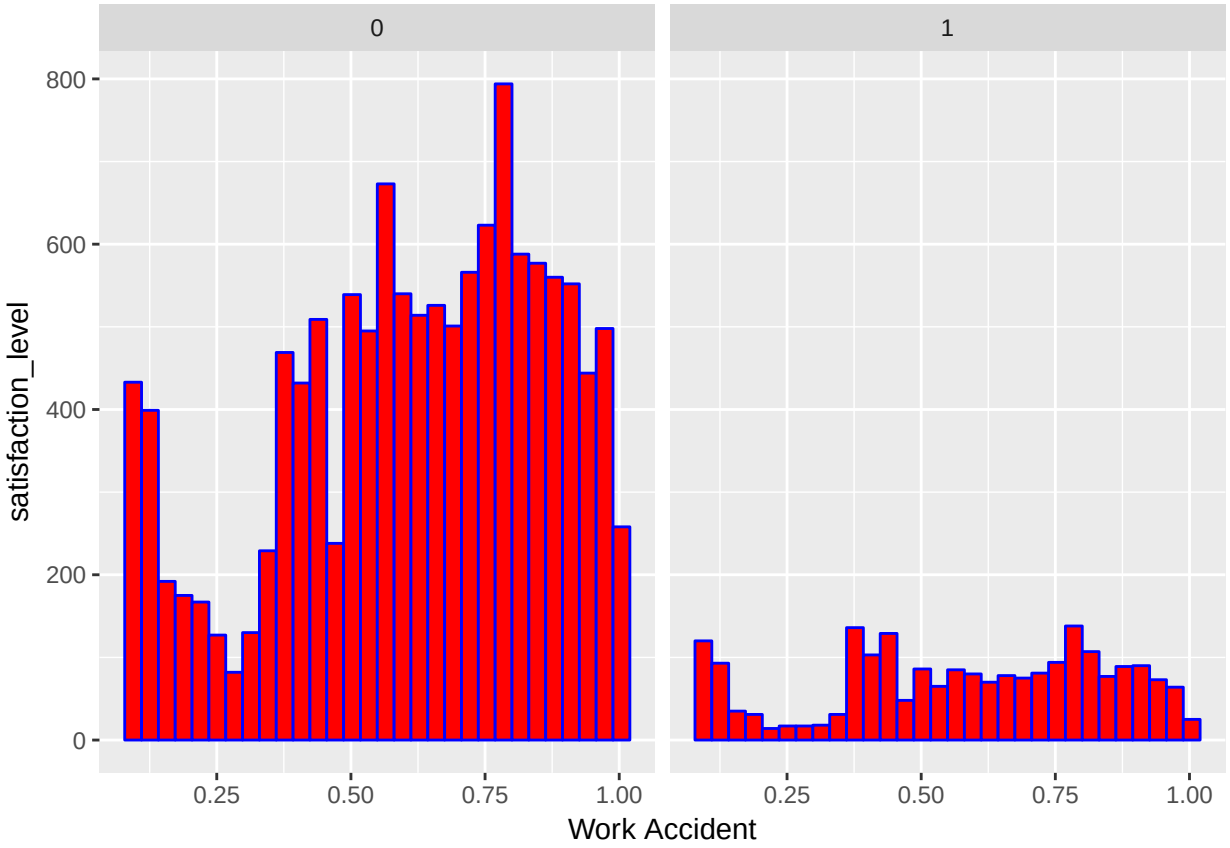



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



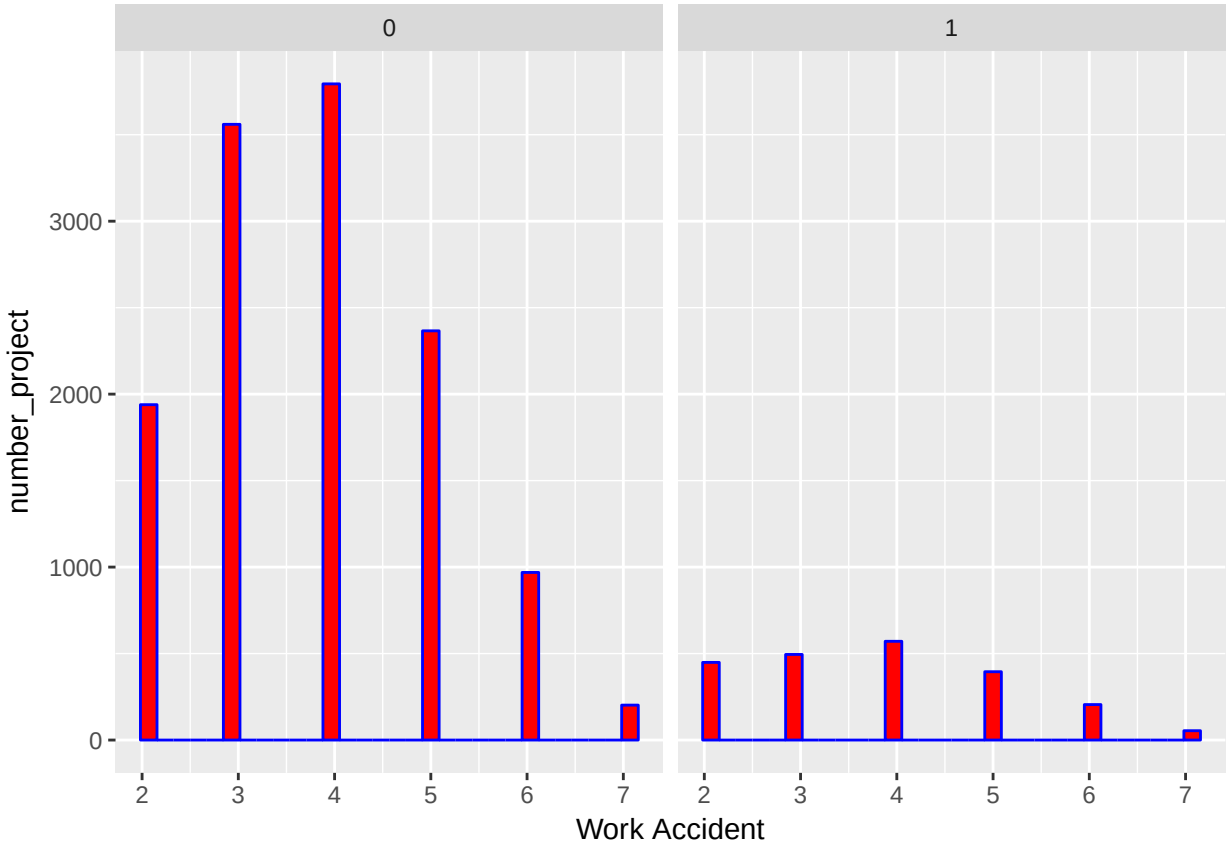

```
for(i in 1:ncol(hrdataquant))
{ if(!(is.factor(hrdataquant[,i])))
{
  print(qplot(x=hrdataquant[,i],data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~hrdata$Work_a
}
}
```

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

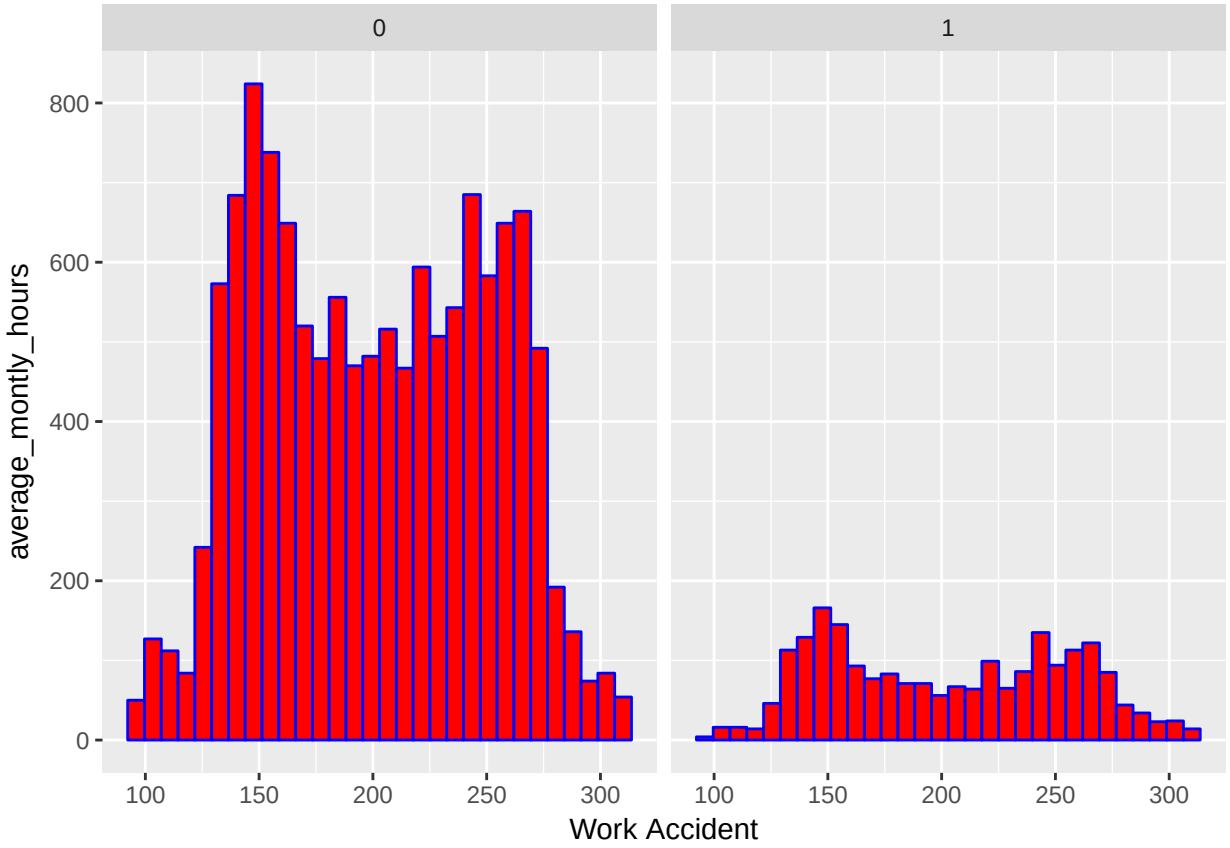



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

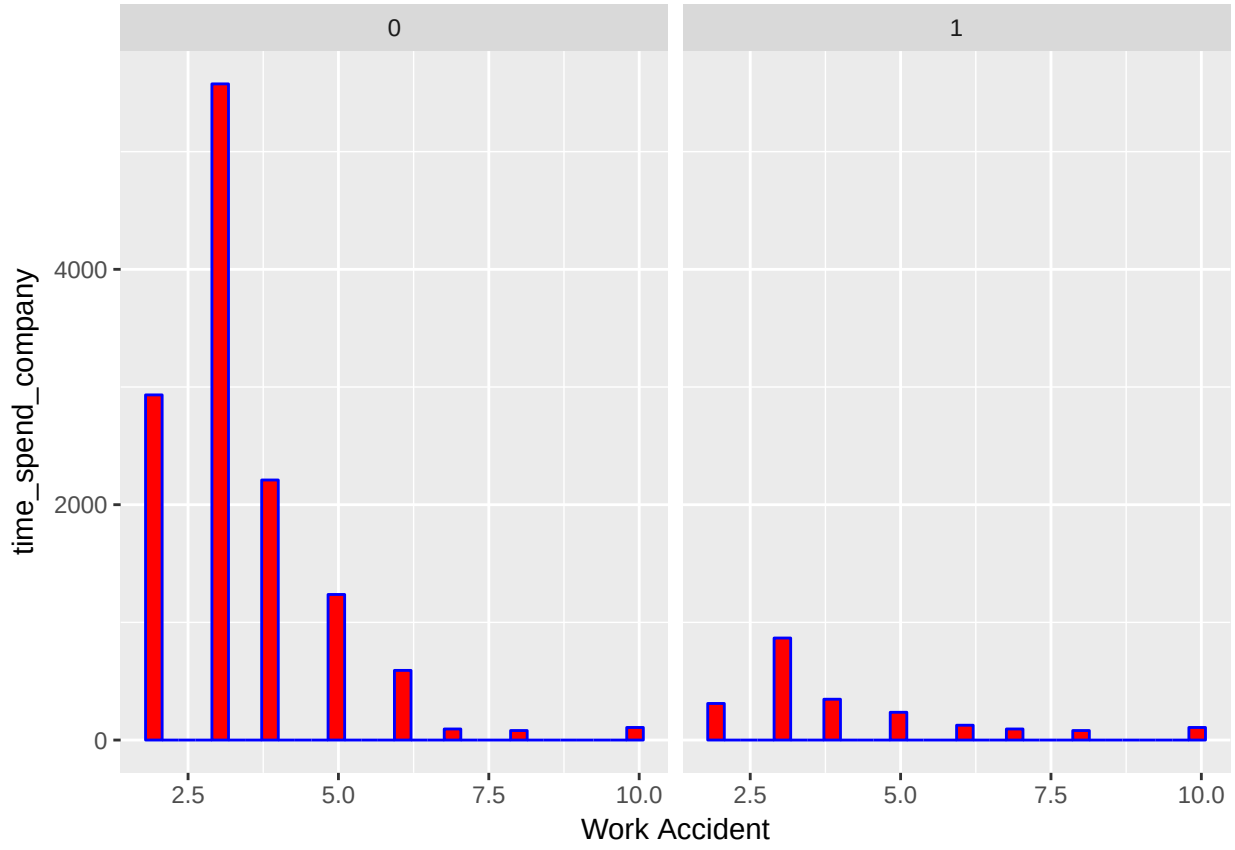

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




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

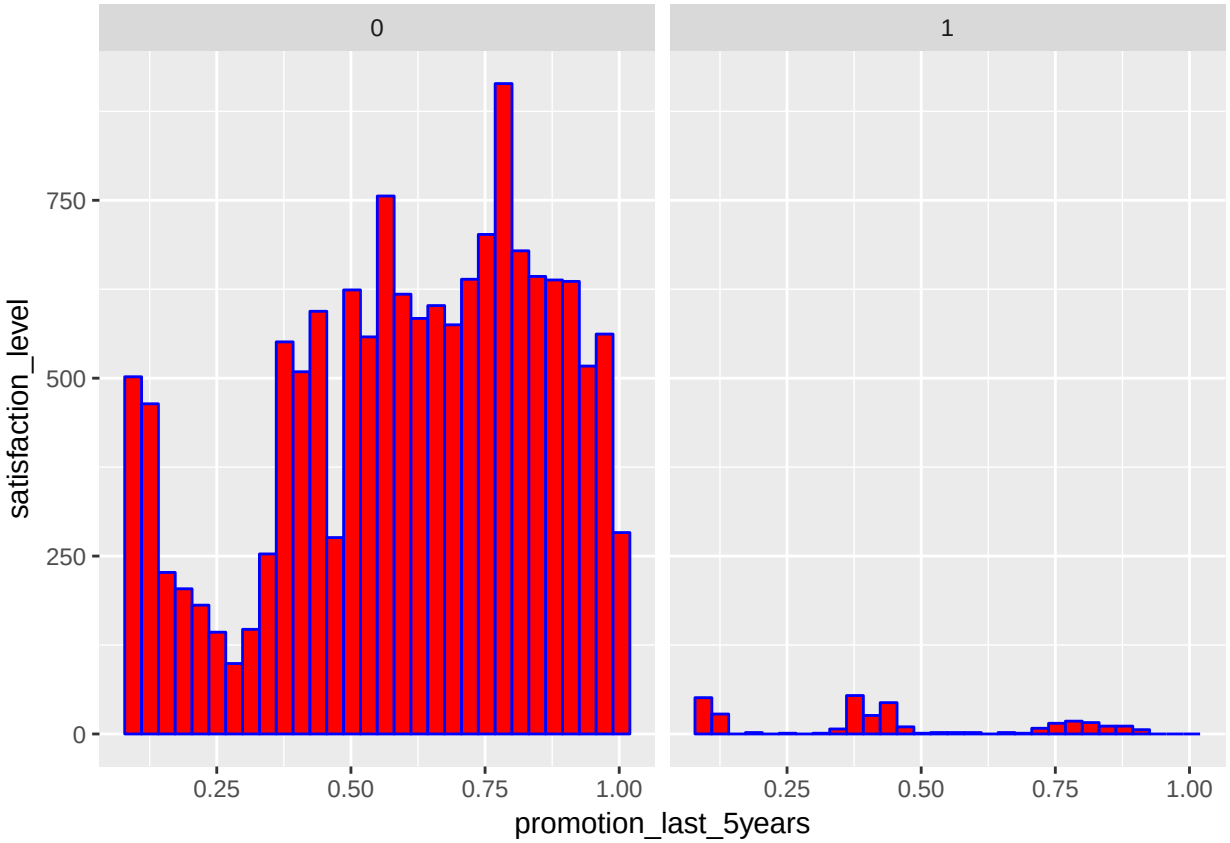



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

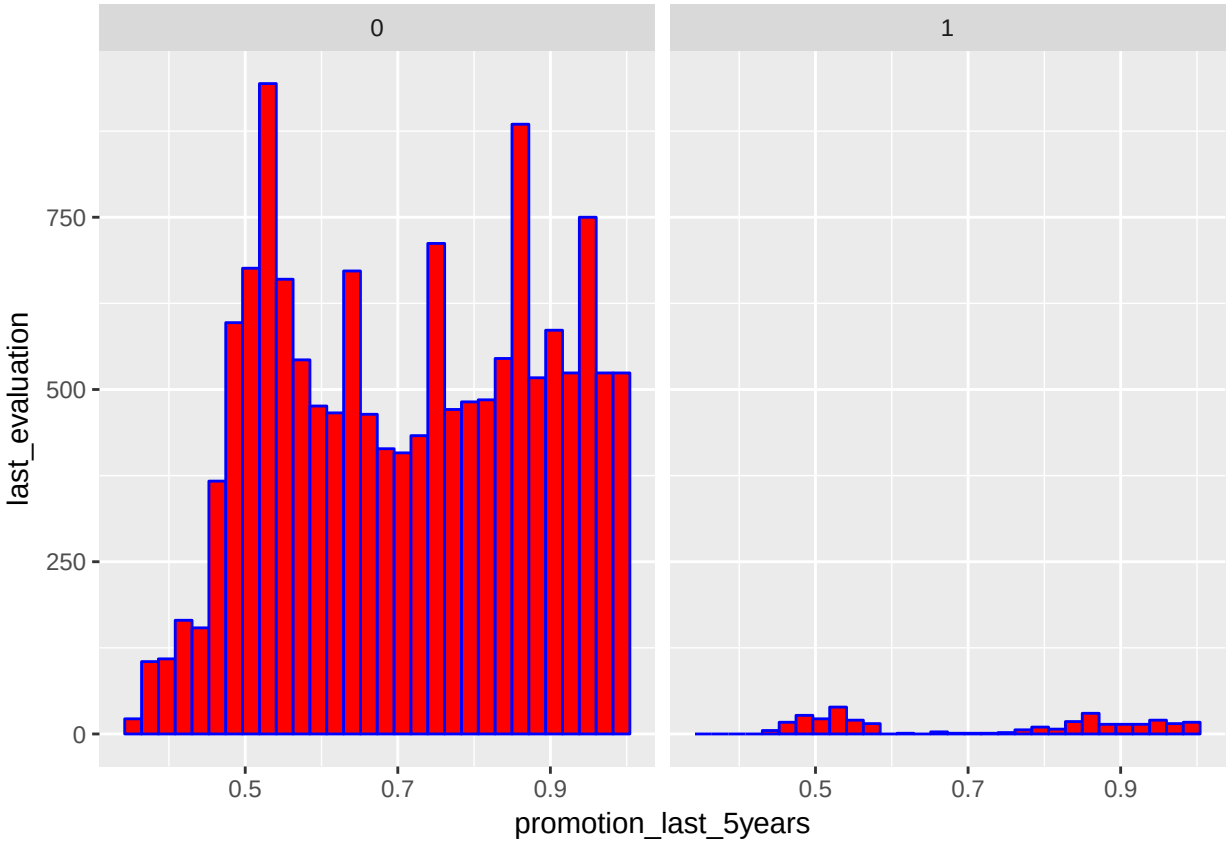



```
for(i in 1:ncol(hrdataquant))
{ if(!(is.factor(hrdataquant[,i])))
{
  print(qplot(x=hrdataquant[,i],data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~hrdata$promo
}
}
```

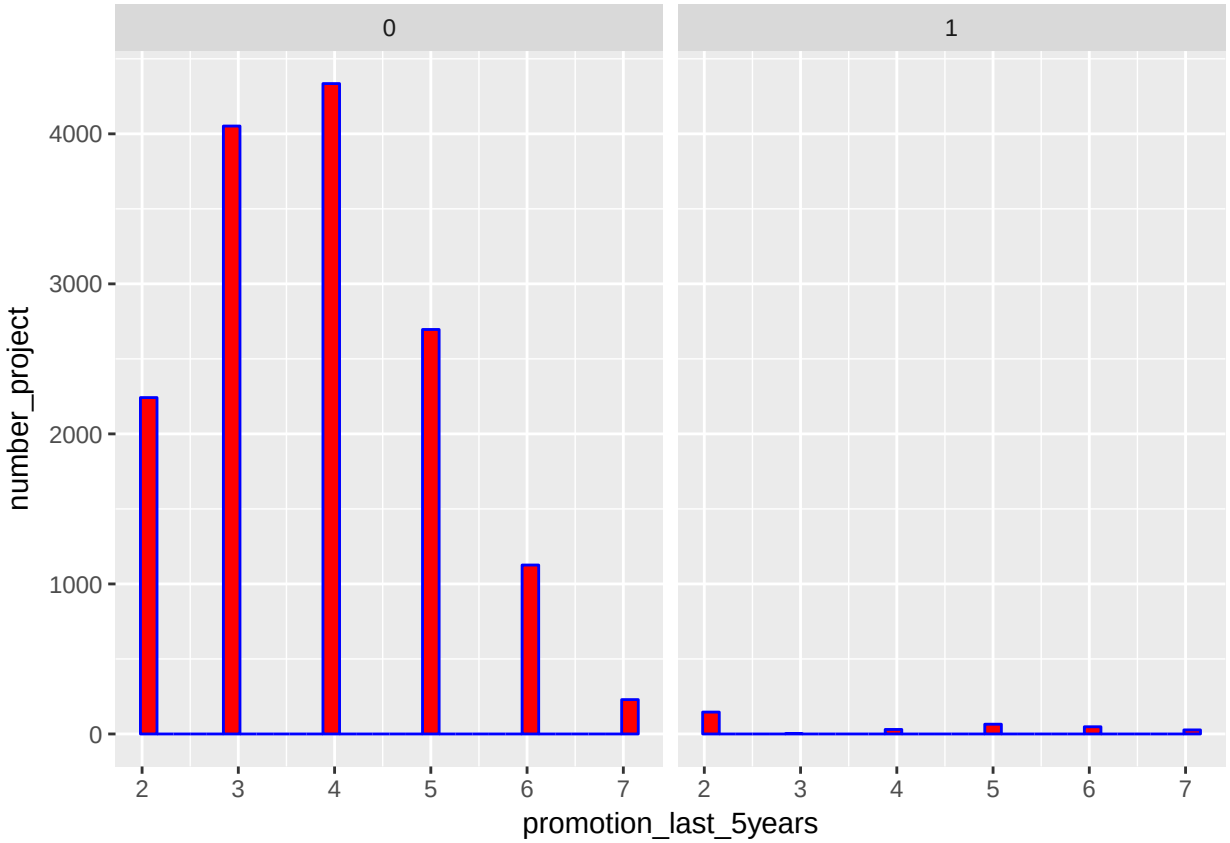
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



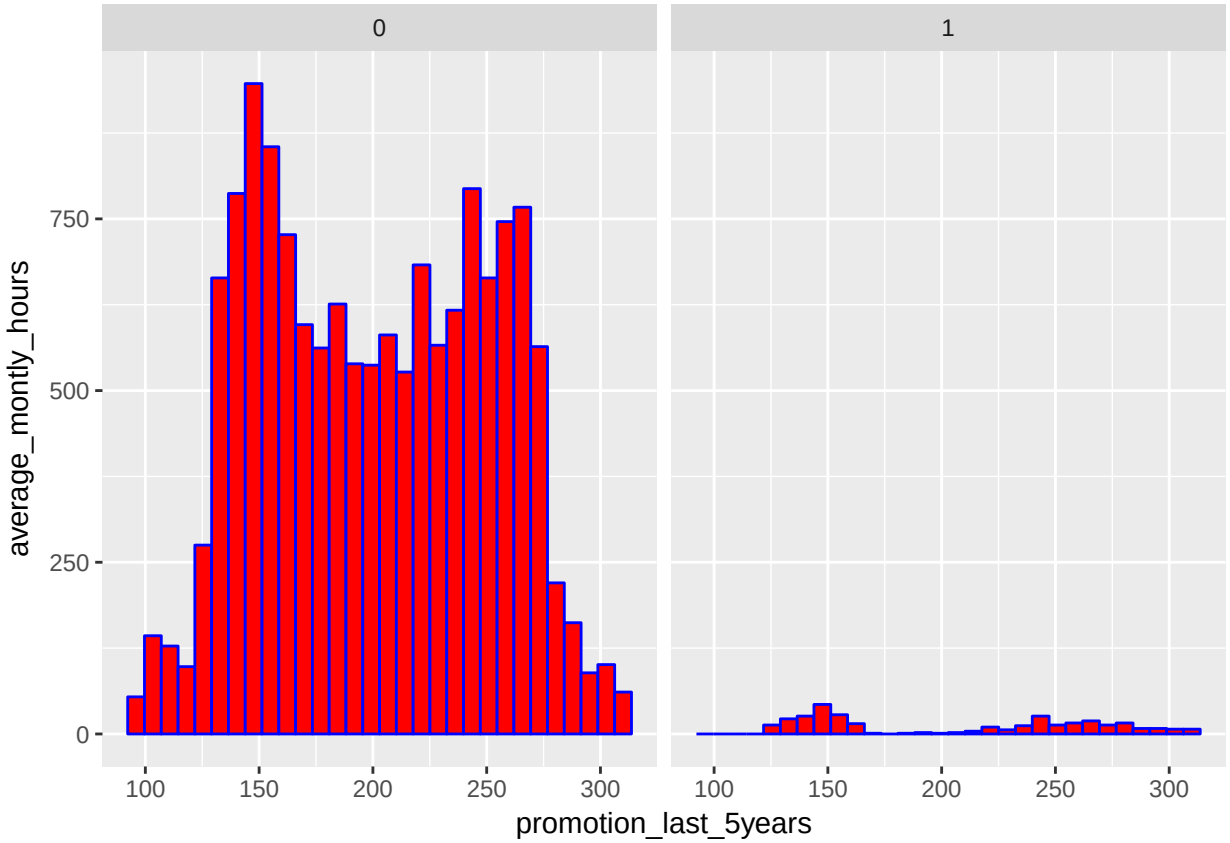

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



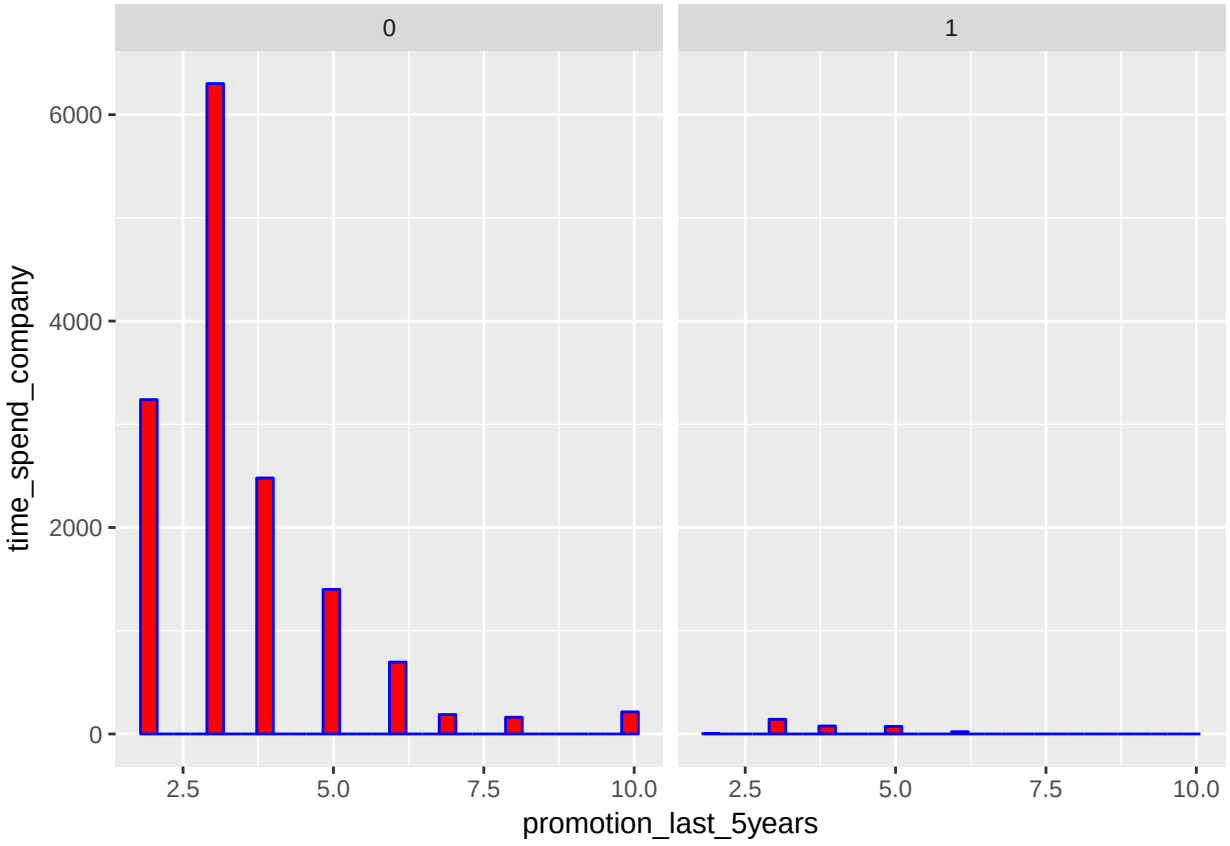

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




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

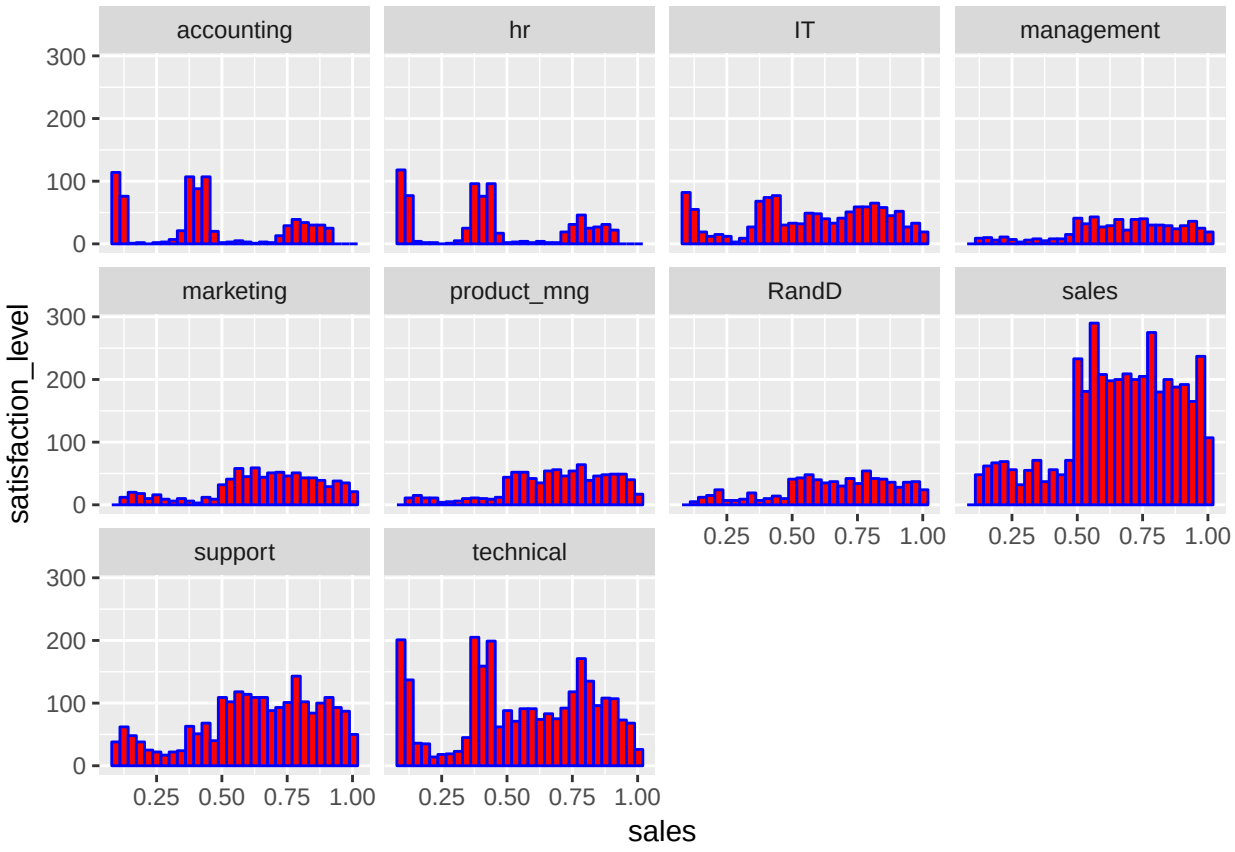



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



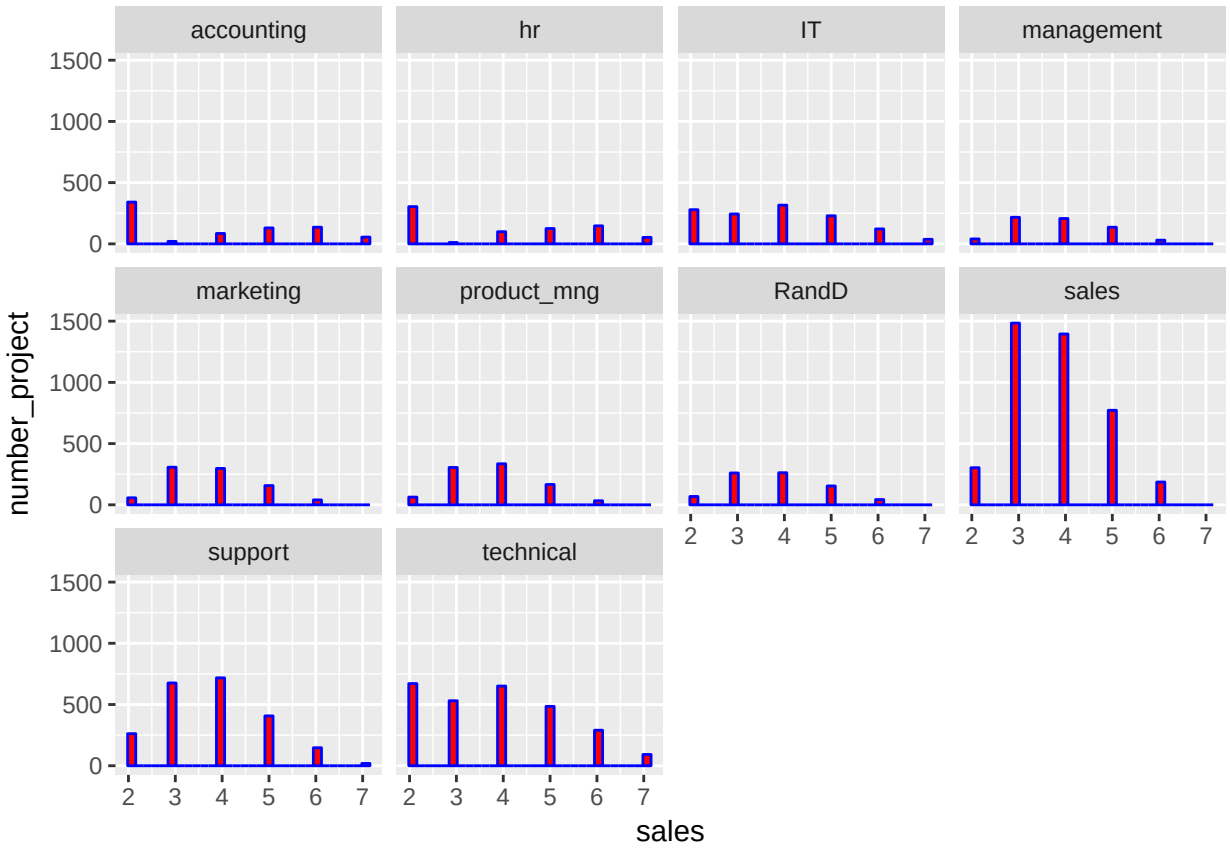

```
for(i in 1:ncol(hrdataquant))
{ if(!(is.factor(hrdataquant[,i])))
{
  print(qplot(x=hrdataquant[,i],data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~hrdata$sales)
}
}
```

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

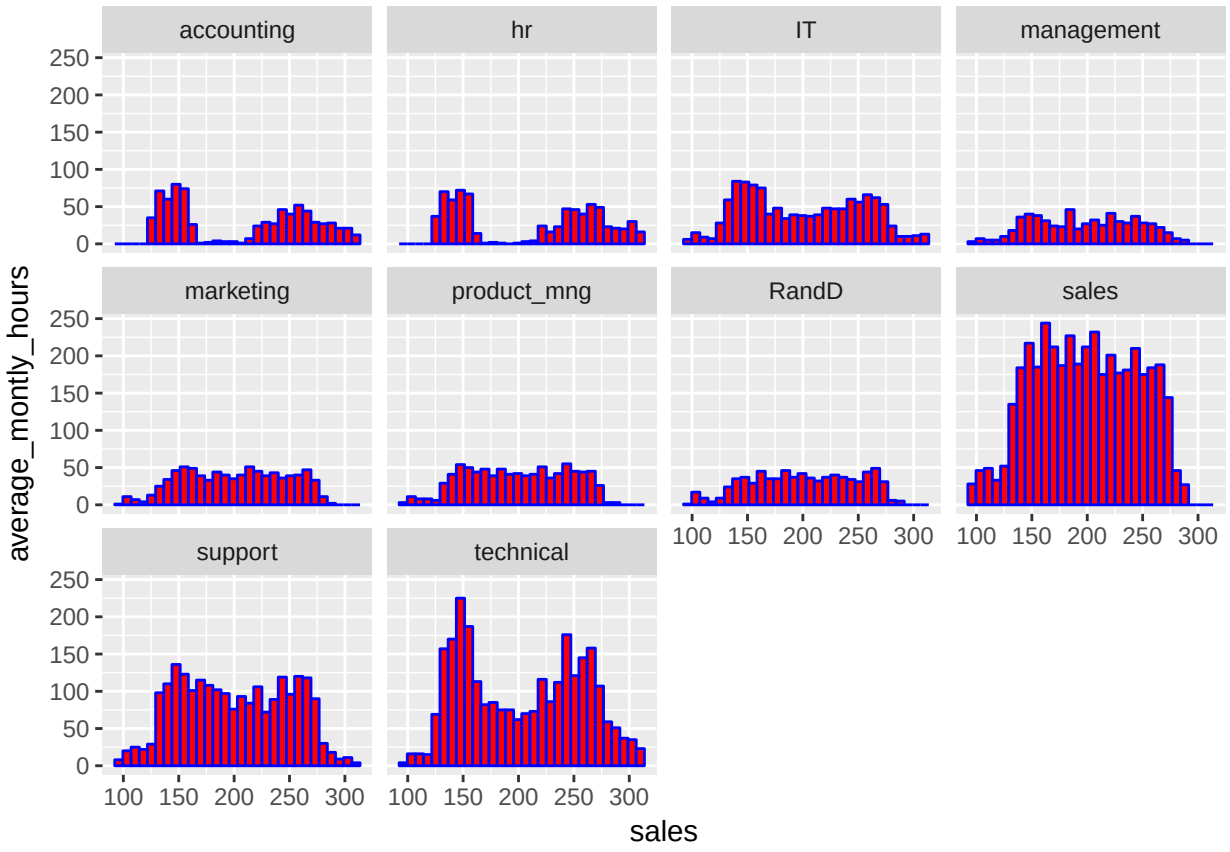



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

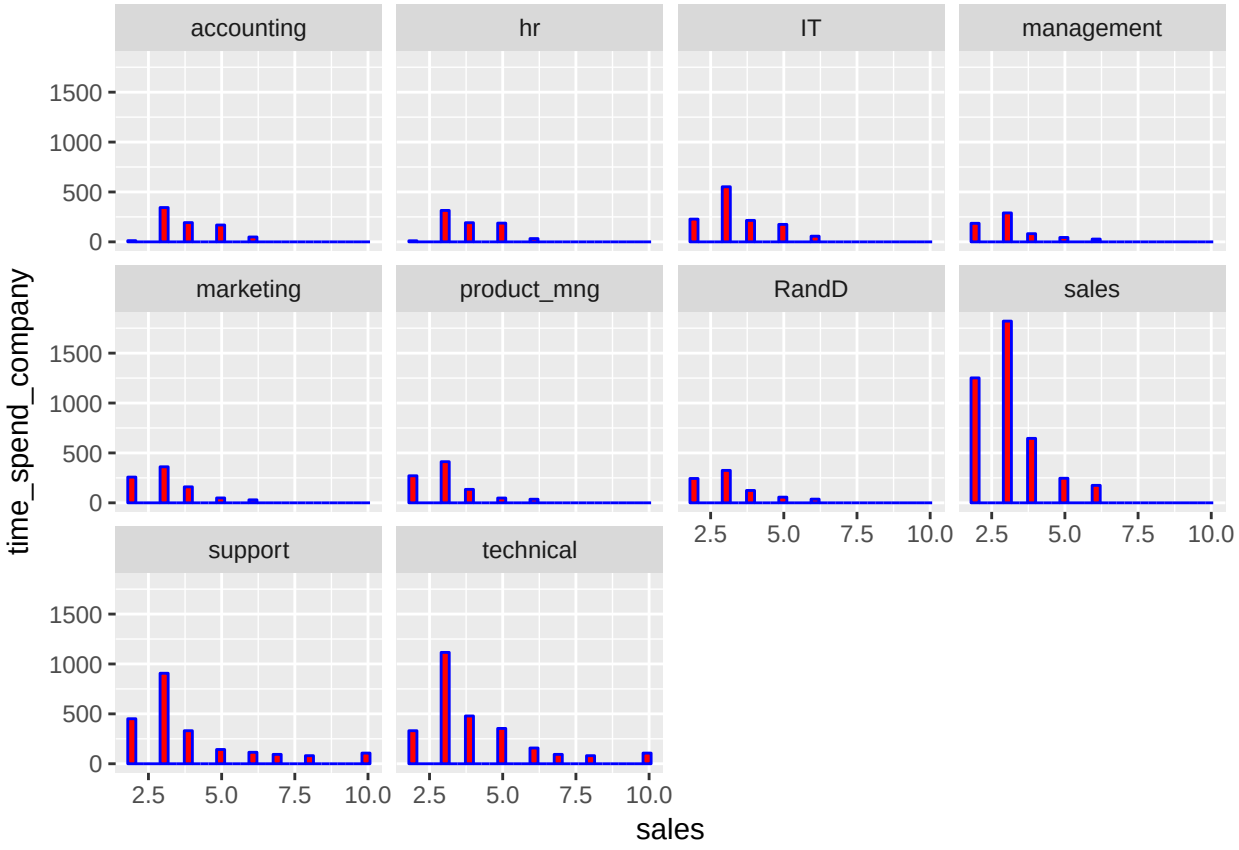

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



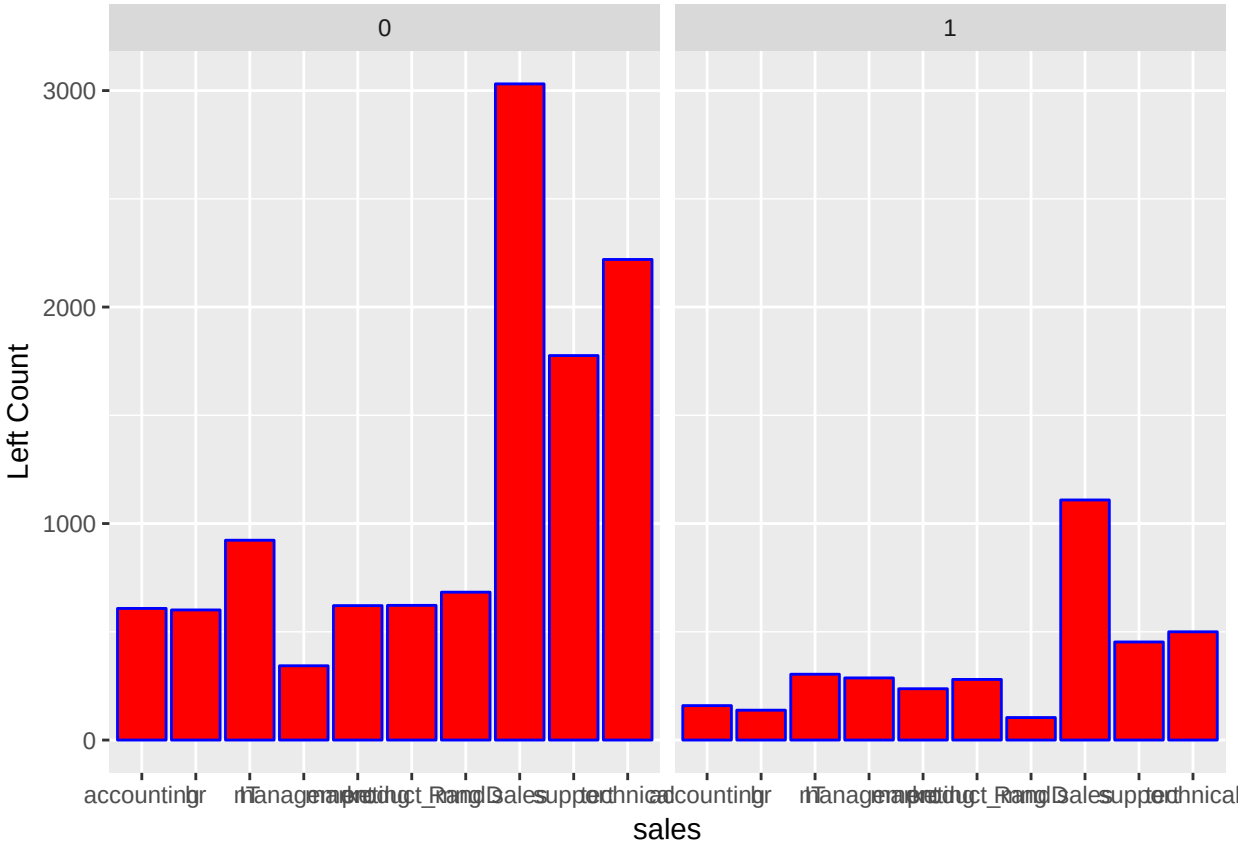

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



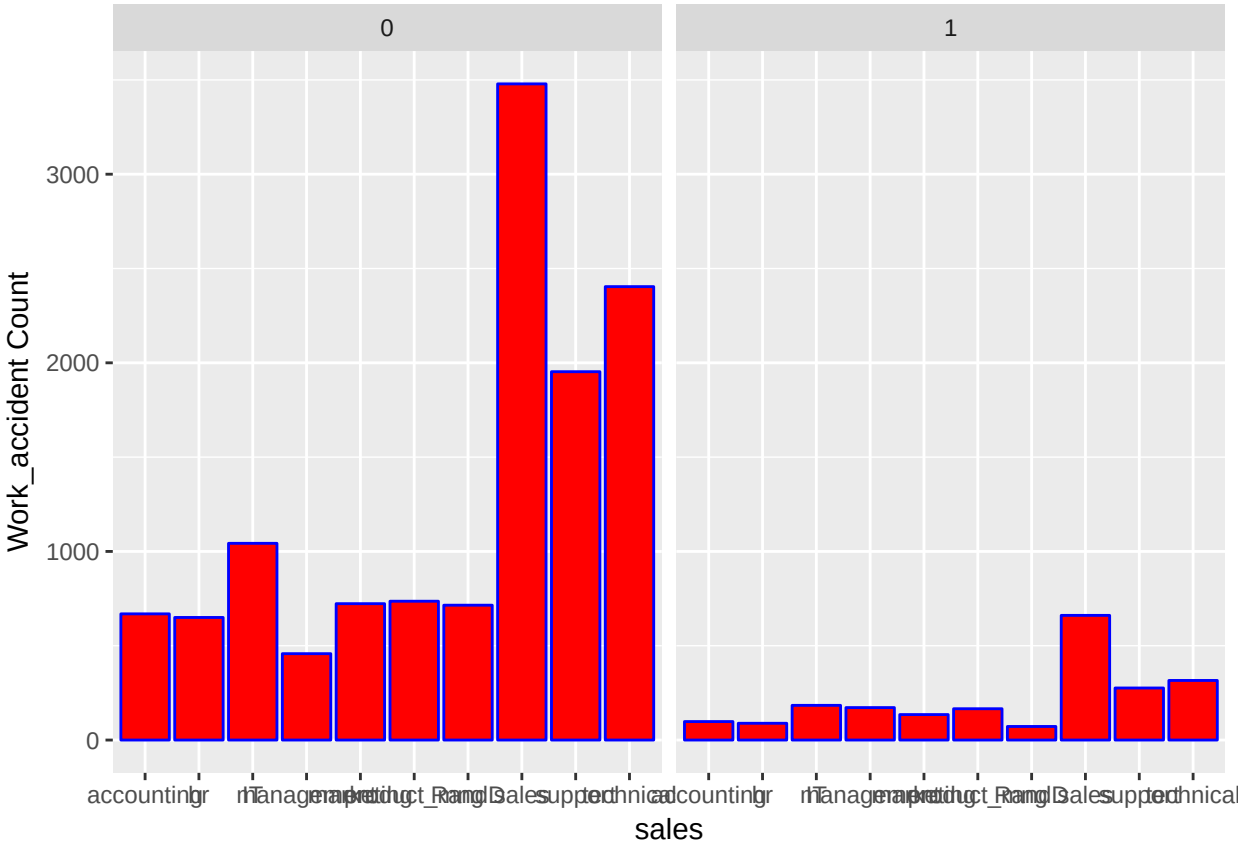

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



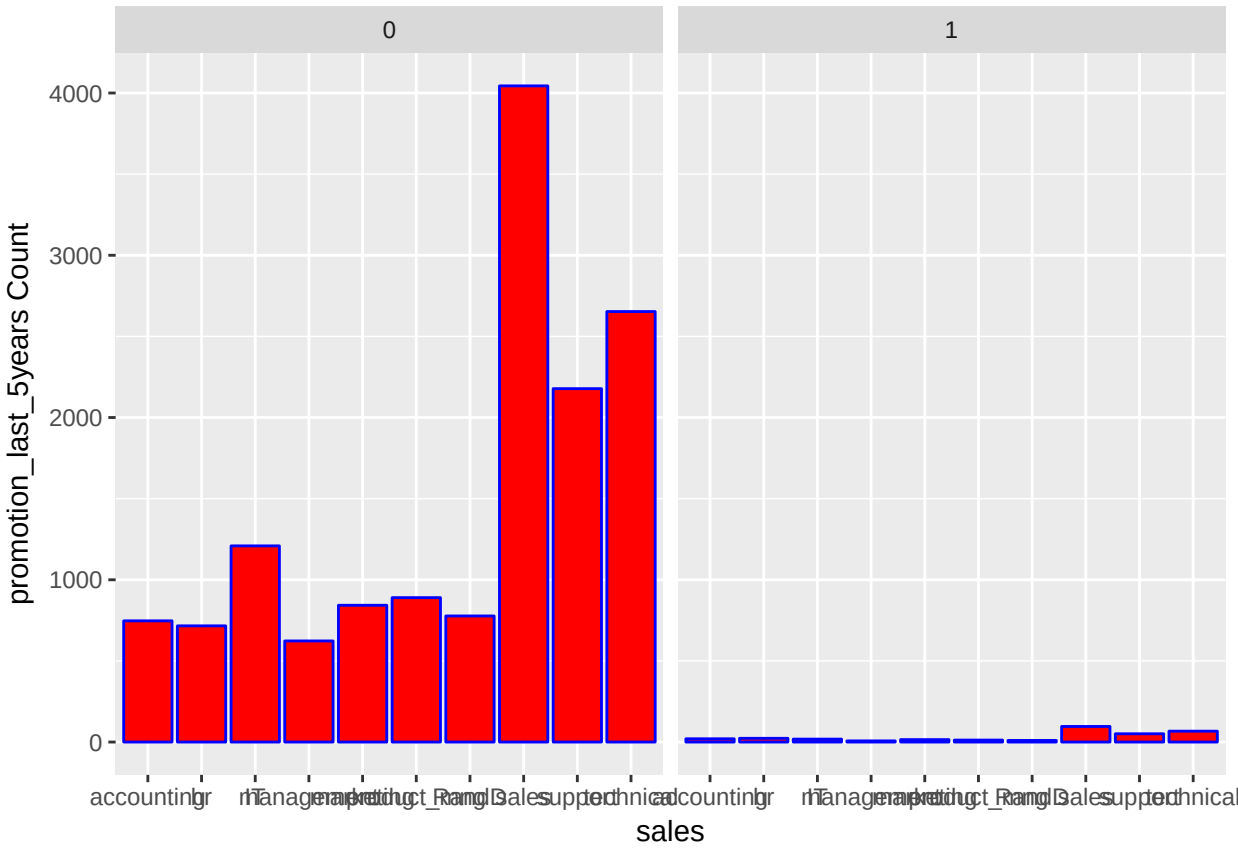

```
#below graph shows that there are people leaving the orgranisation have majority of
#people having satisfaction level above 50%...which is strange.
qplot(x=hrdata$sales,data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~hrdata$left)+labs(x="sa
```



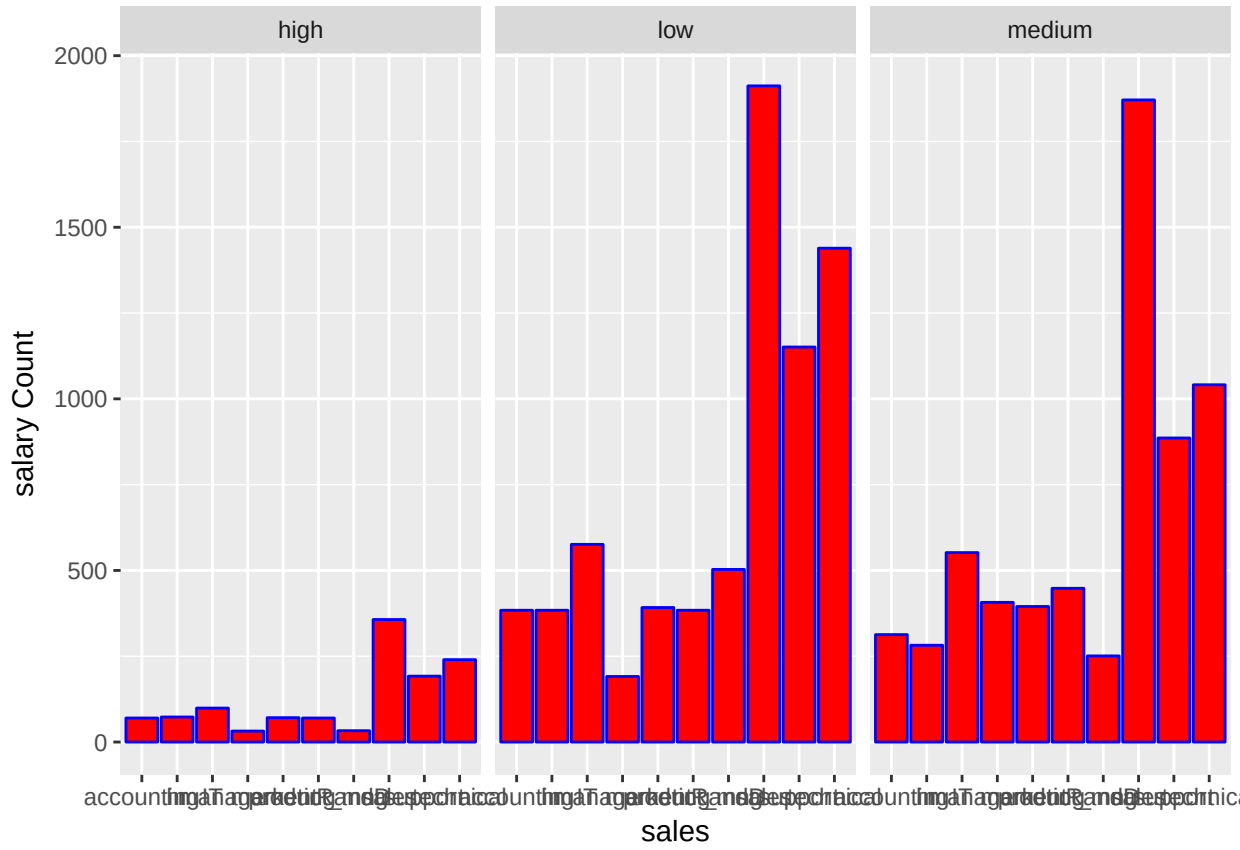

```
qplot(x=hrdata$sales,data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~hrdata$Work_accident)+la
```




```
qplot(x=hrdata$sales,data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~hrdata$promotion_last_5y
```

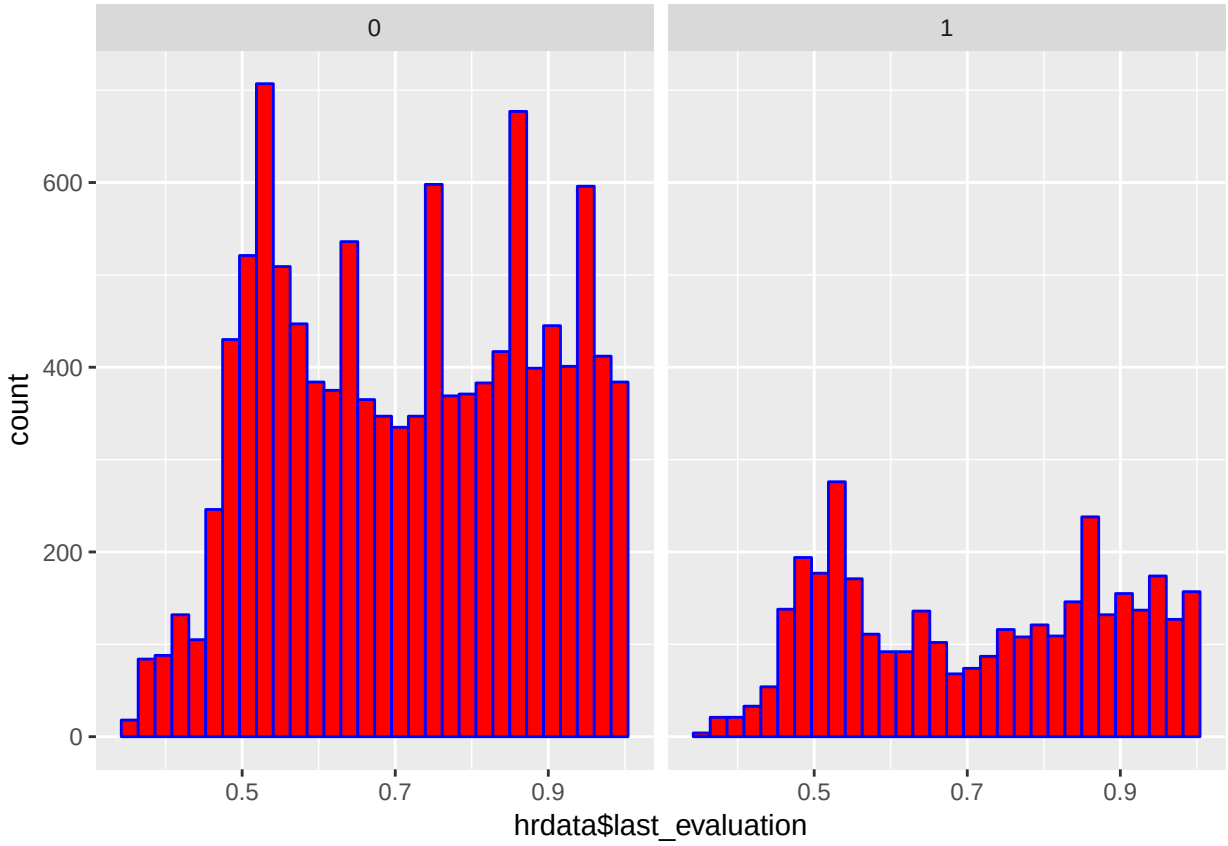



```
qplot(x=hrdata$sales,data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~hrdata$salary)+labs(x="
```

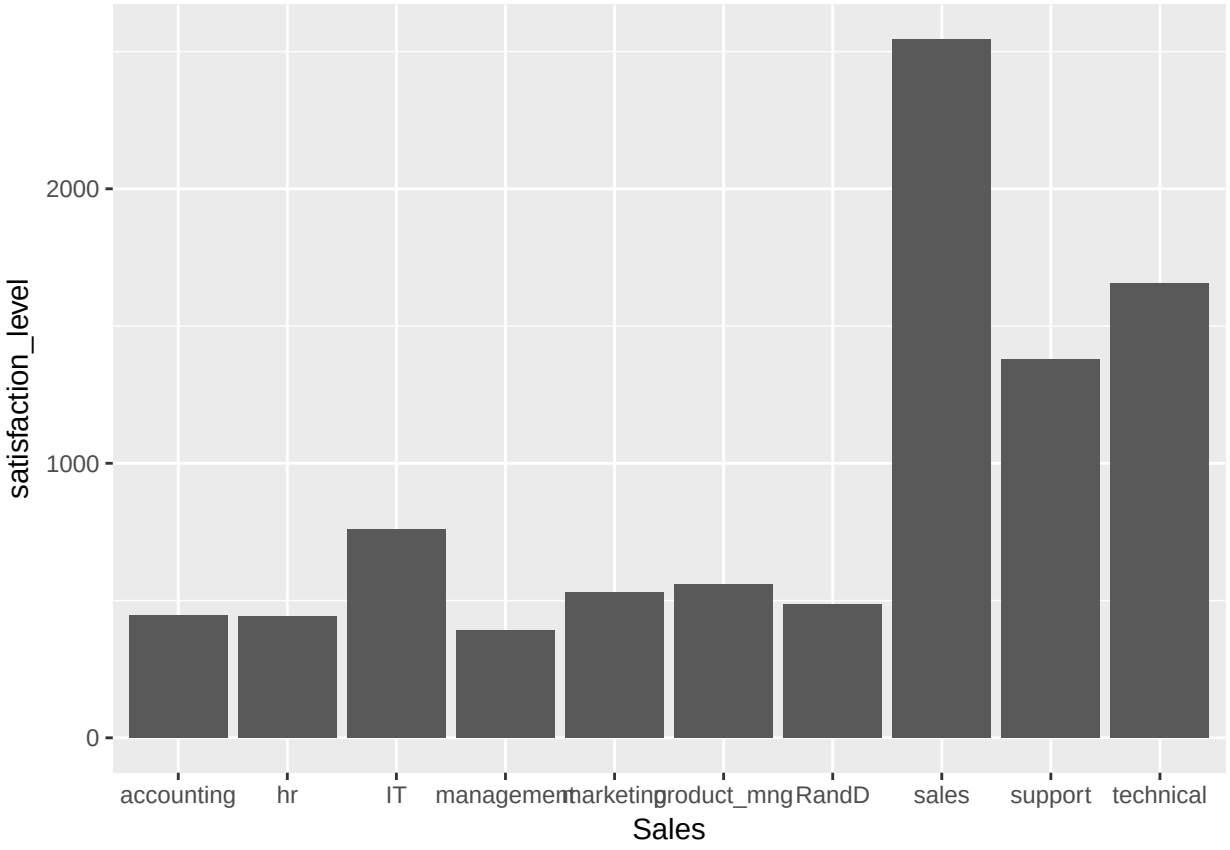


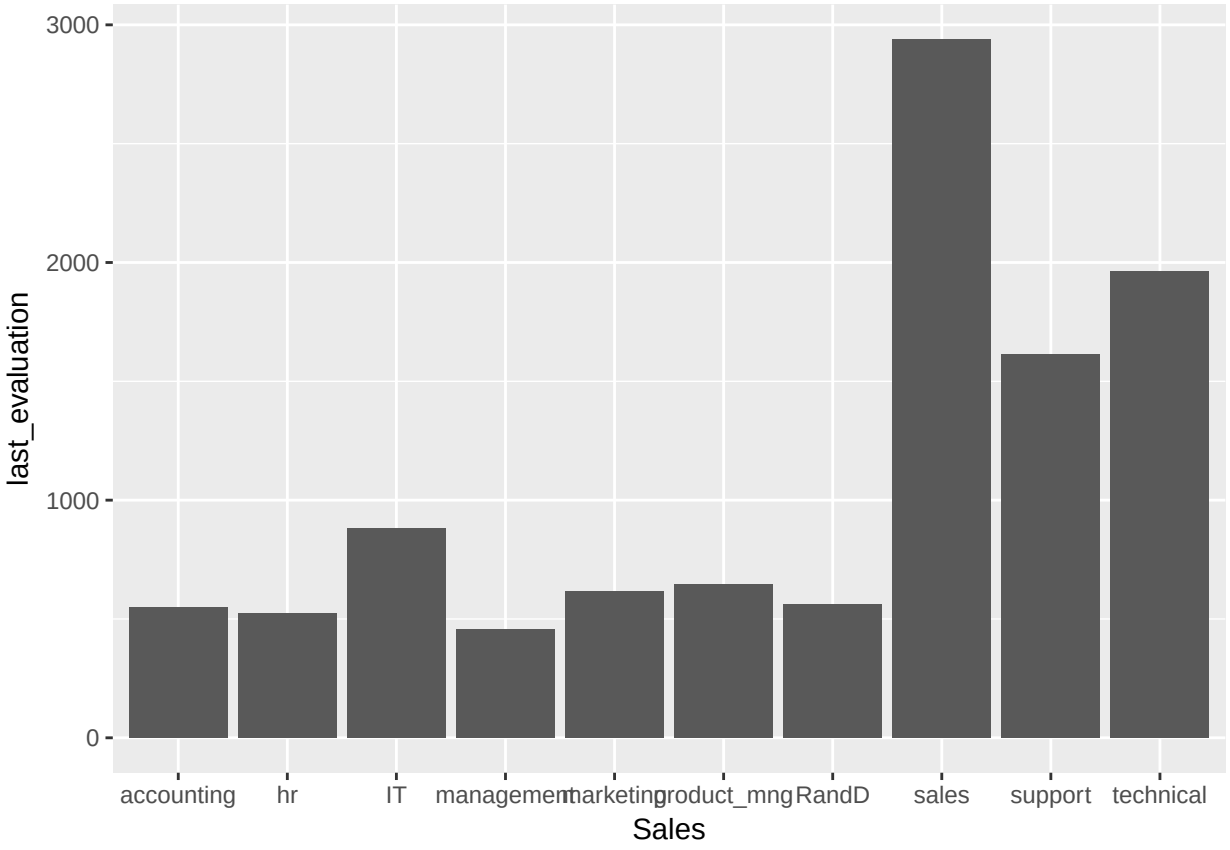

```
qplot(x=hrdata$last_evaluation,data=hrdata,col = I("blue"),fill = I("red")) + facet_wrap(~hrdata$left)
```

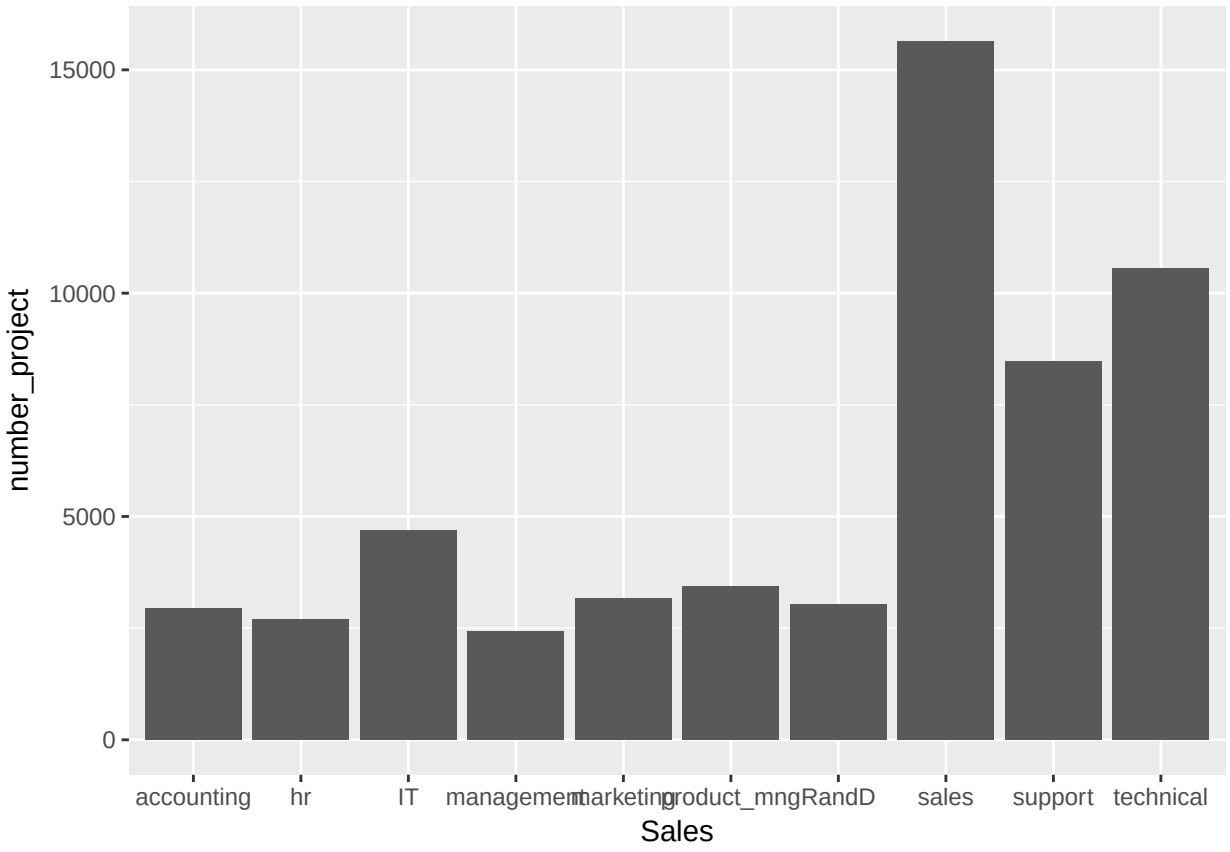
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

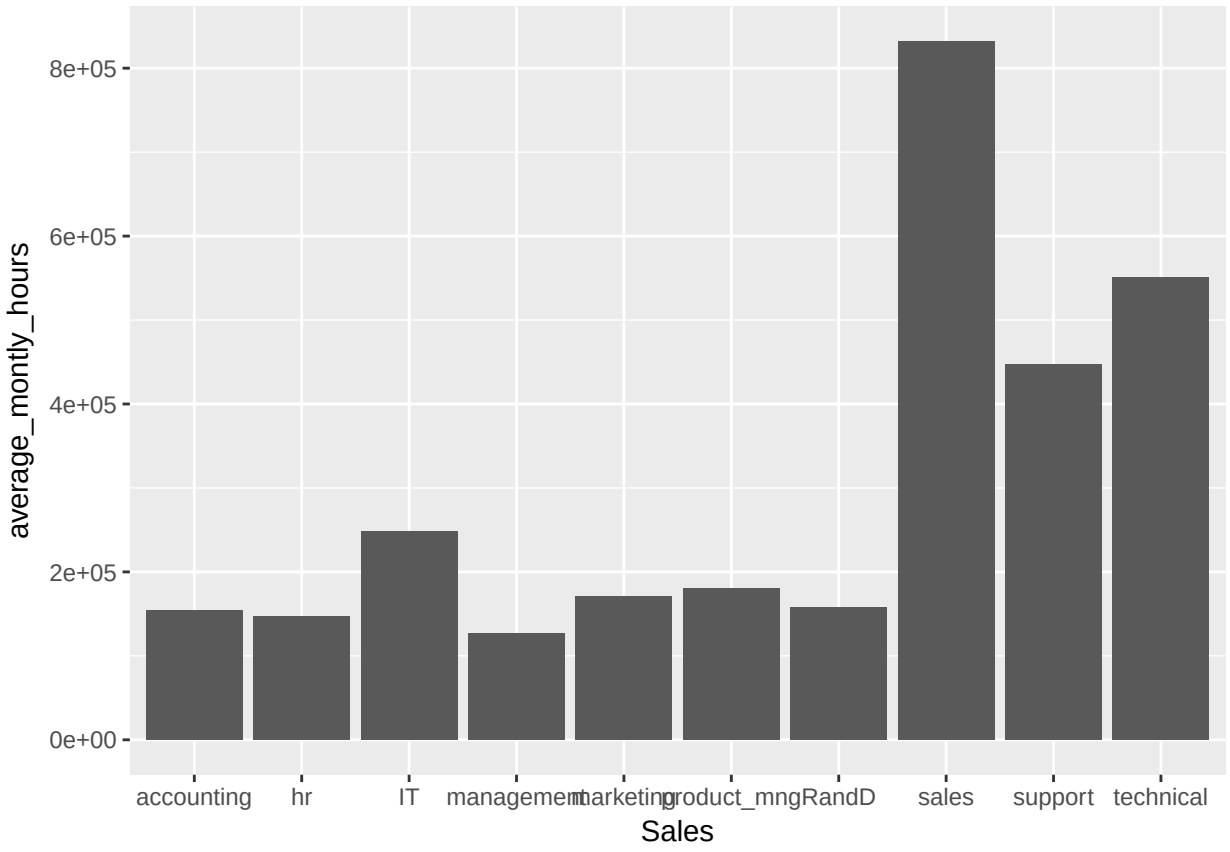


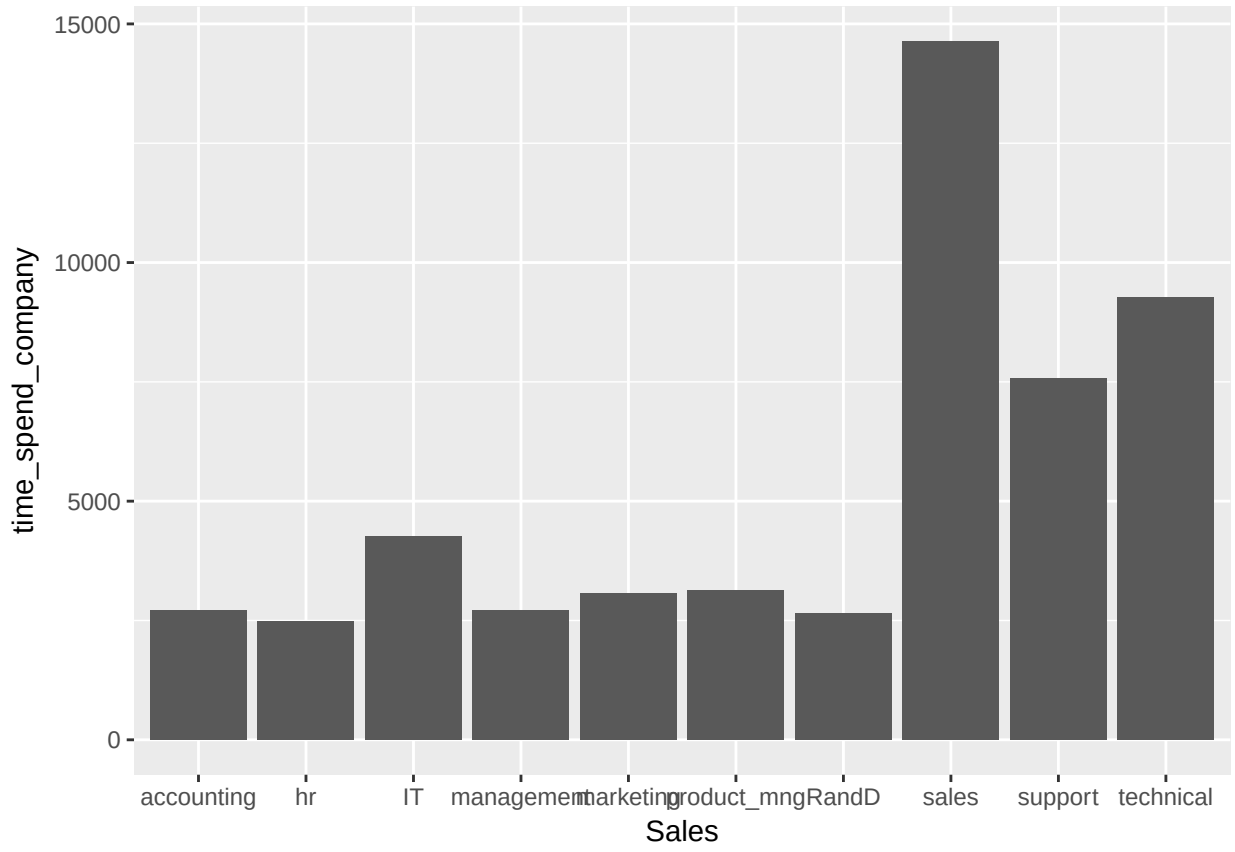

```
for(i in 1:ncol(hrdataquant))
{ if(!(is.factor(hrdataquant[,i])))
{
  print(ggplot(hrdata,aes(x = sales , y =hrdataquant[,i]))+geom_bar(stat="identity")+labs(x="Sales",y=c

}
}
```


satisfaction_level
2000
1000
0
accounting
hr
IT
management
marketing
product_mng
RandD
sales
support
technical
Sales

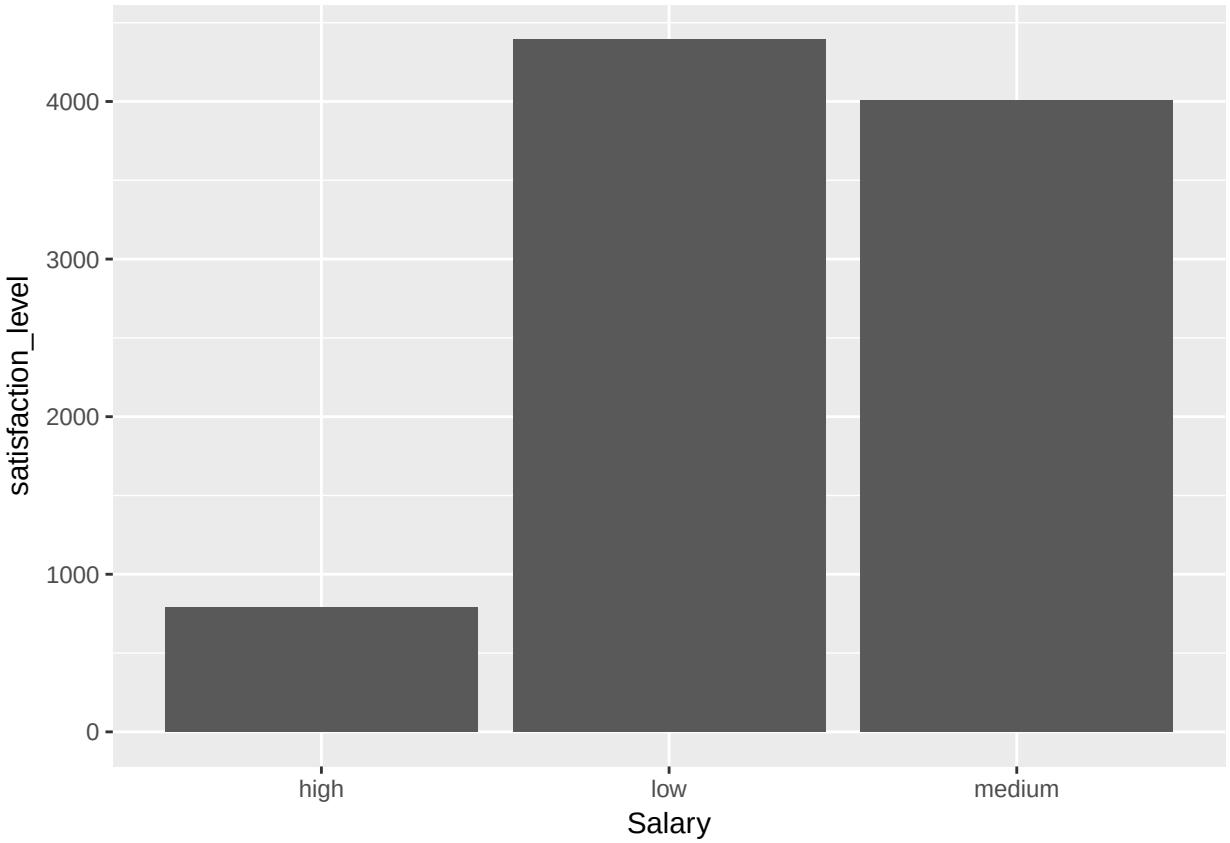

3000
2000
1000
0
last_evaluation
accounting
hr
IT
management
marketing
product_mng
RandD
sales
support
technical
Sales

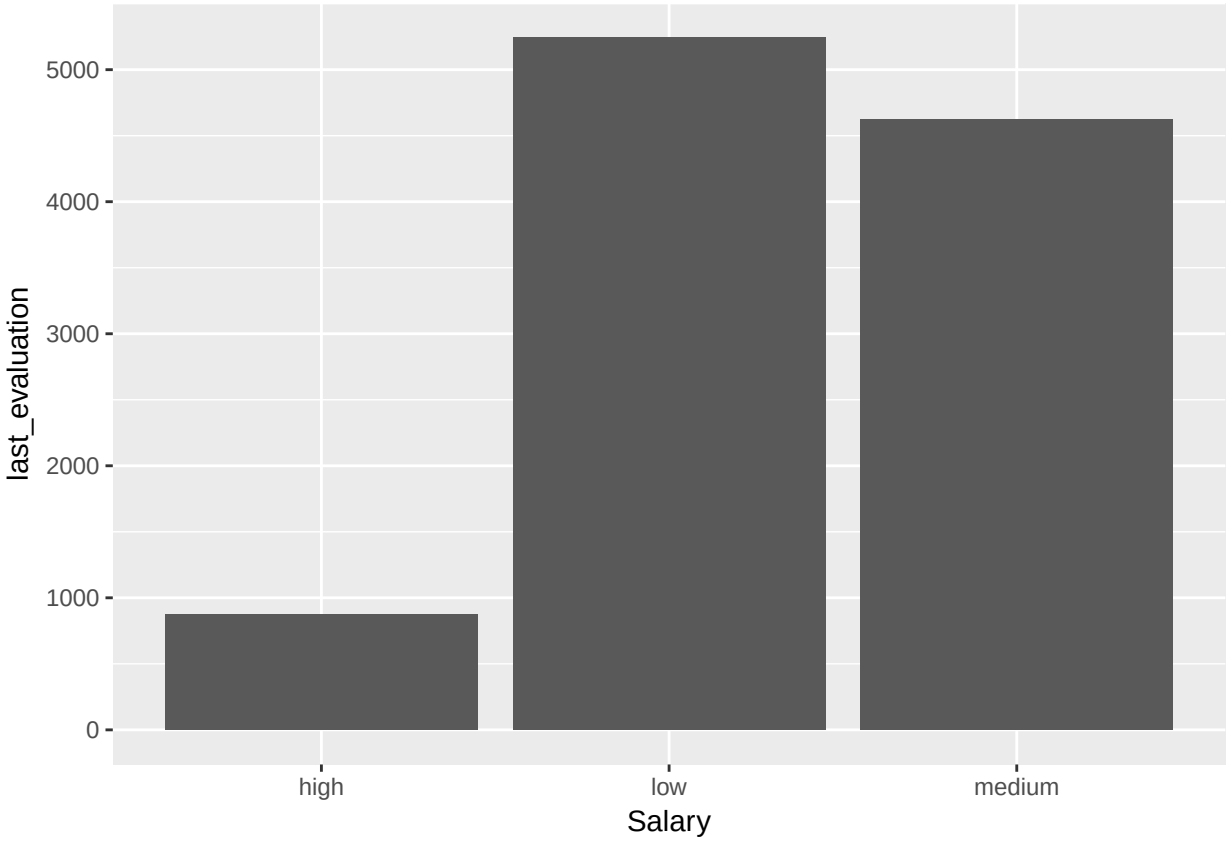

15000
10000
5000
0
number_project
accounting
hr
IT
management
marketing
product_mng
RandD
sales
support
technical
Sales

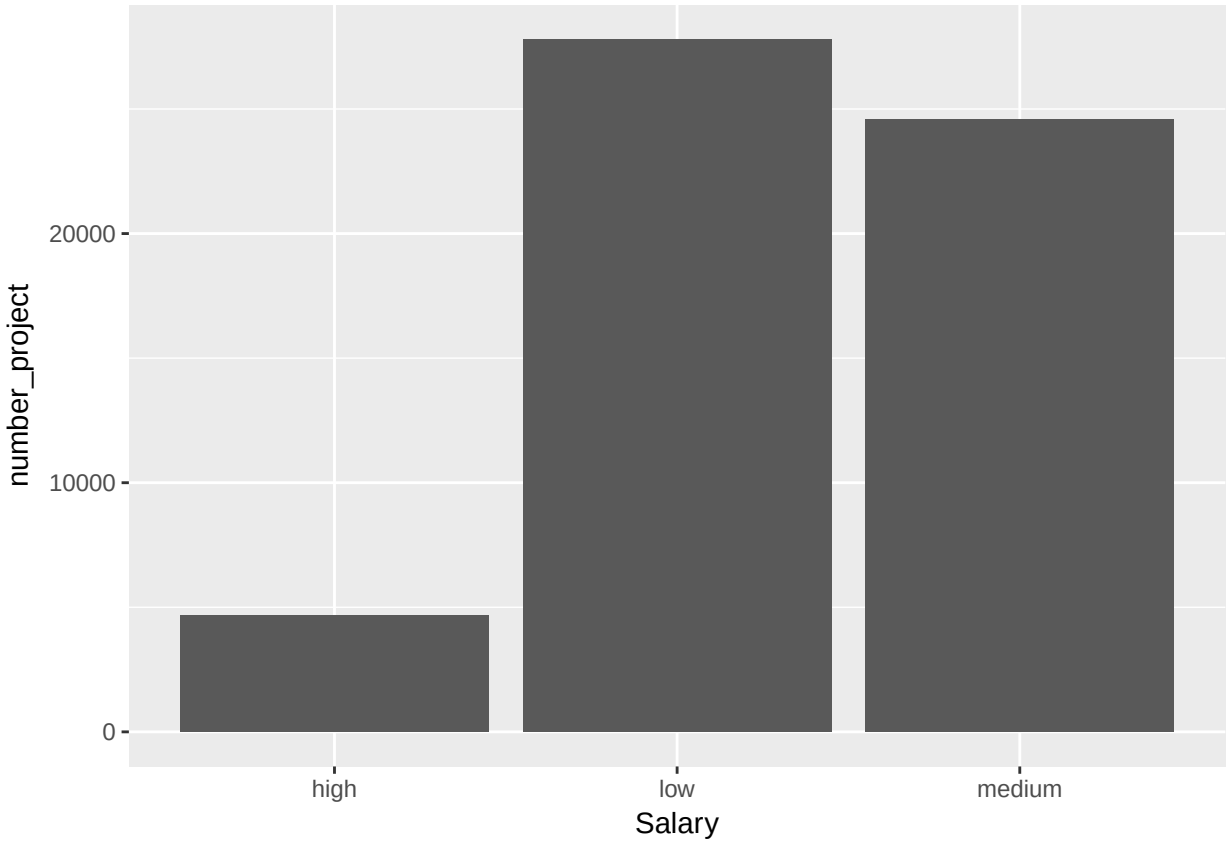

8e+05
6e+05
4e+05
2e+05
0e+00
average_montly_hours
accounting
hr
IT
management
marketing
product_mng
RandD
sales
support
technical
Sales

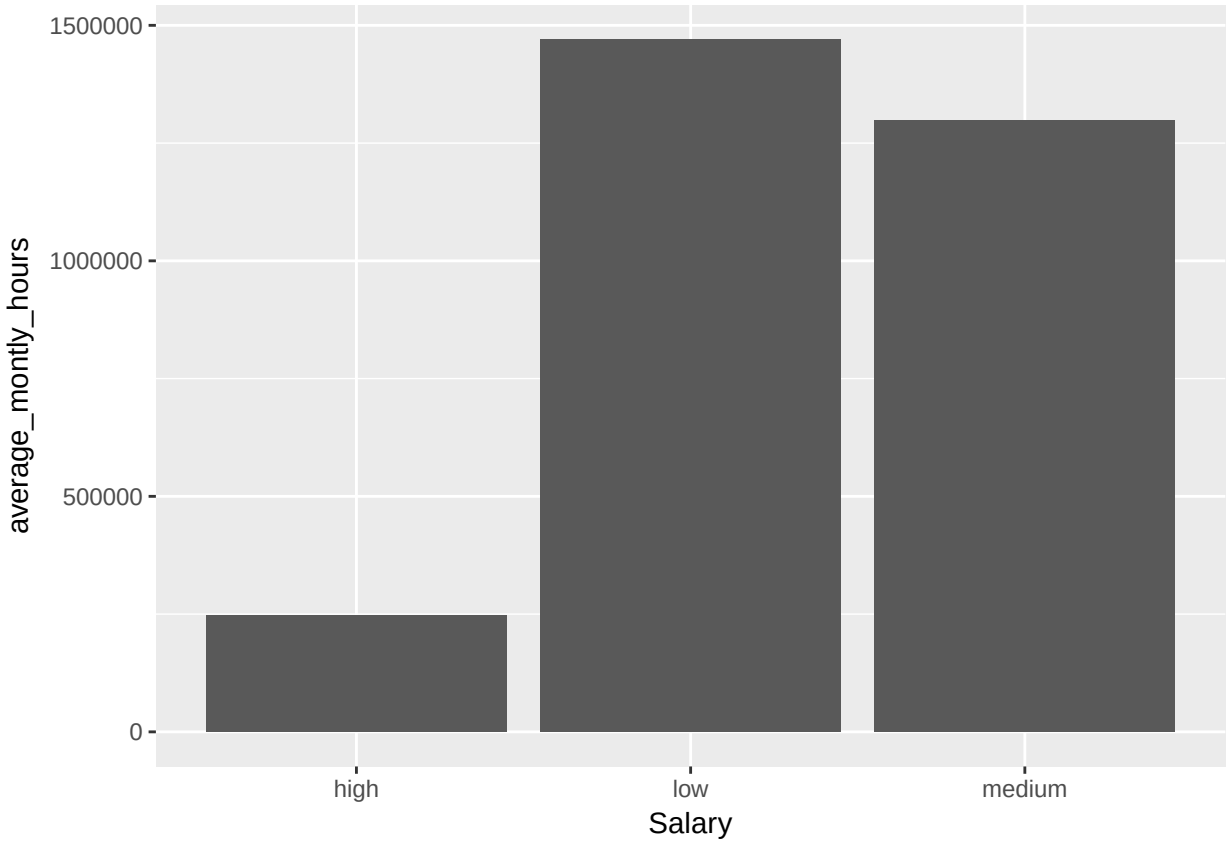


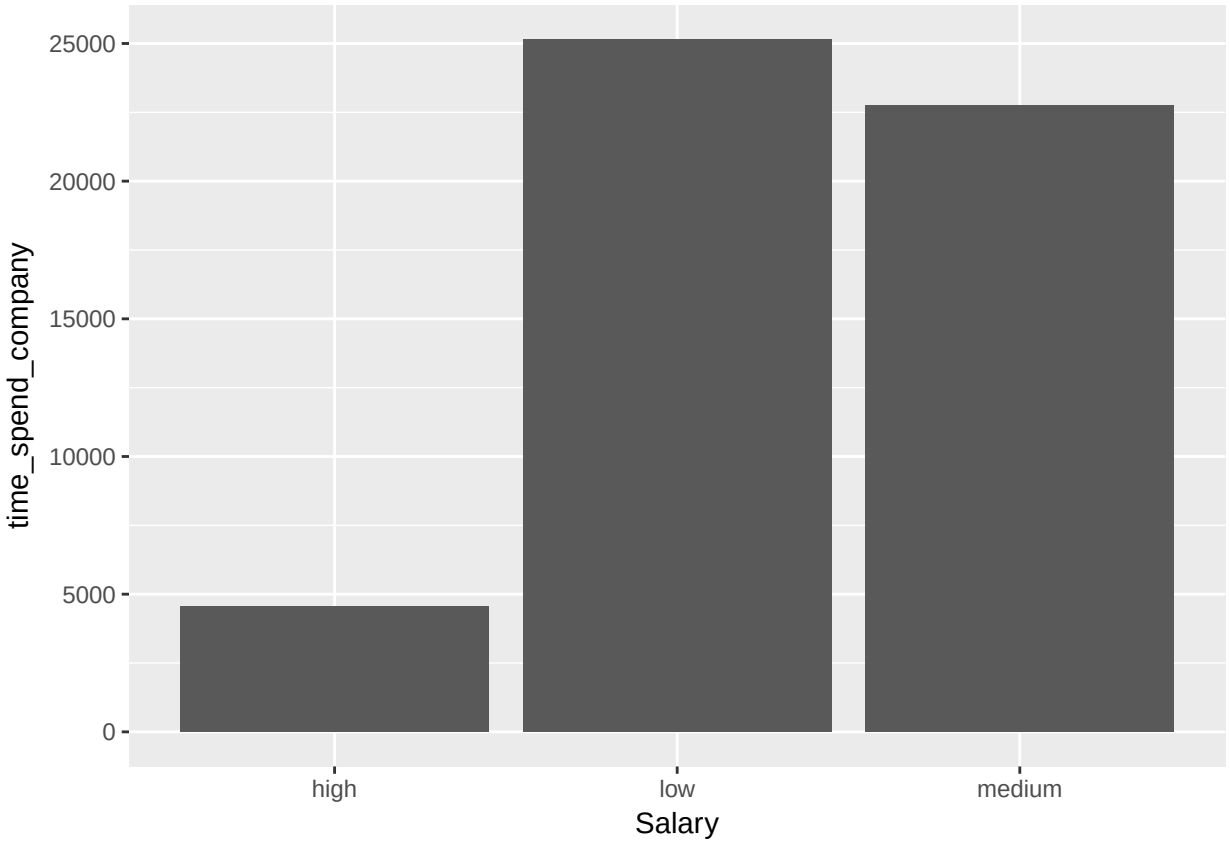

```
#more number of people are from sales, support and technical department
#employees from sales, support and technical department have recieved the last evaluations
#major contribution of project is from sales, support and technical departement
#avg monthy hours and time spend in the company are also more for these three departments
#accident in work is also more in these department.
#and they are one who leaves the company more, sales being the highest.
#sales, management and marketing are the ones who have contributes in getting
#promotion in last 5 years

for(i in 1:ncol(hrdataquant))
{ if(!(is.factor(hrdataquant[,i])))
{
  print(ggplot(hrdata,aes(x = salary , y =hrdataquant[,i]))+geom_bar(stat="identity")+labs(x="Salary",y=

}
}
```


4000
3000
2000
1000
0
satisfaction_level
high
low
medium
Salary

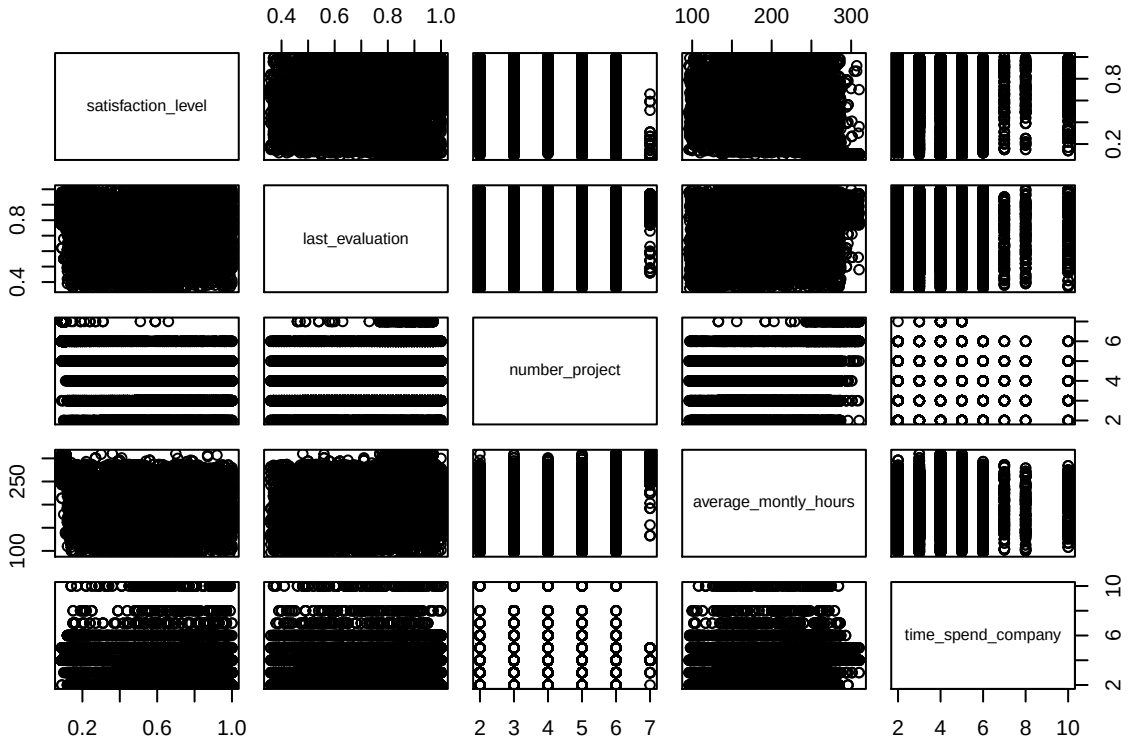

5000
4000
3000
2000
1000
0
last_evaluation
high
low
medium
Salary


number_project
20000
10000
0
high
low
medium
Salary

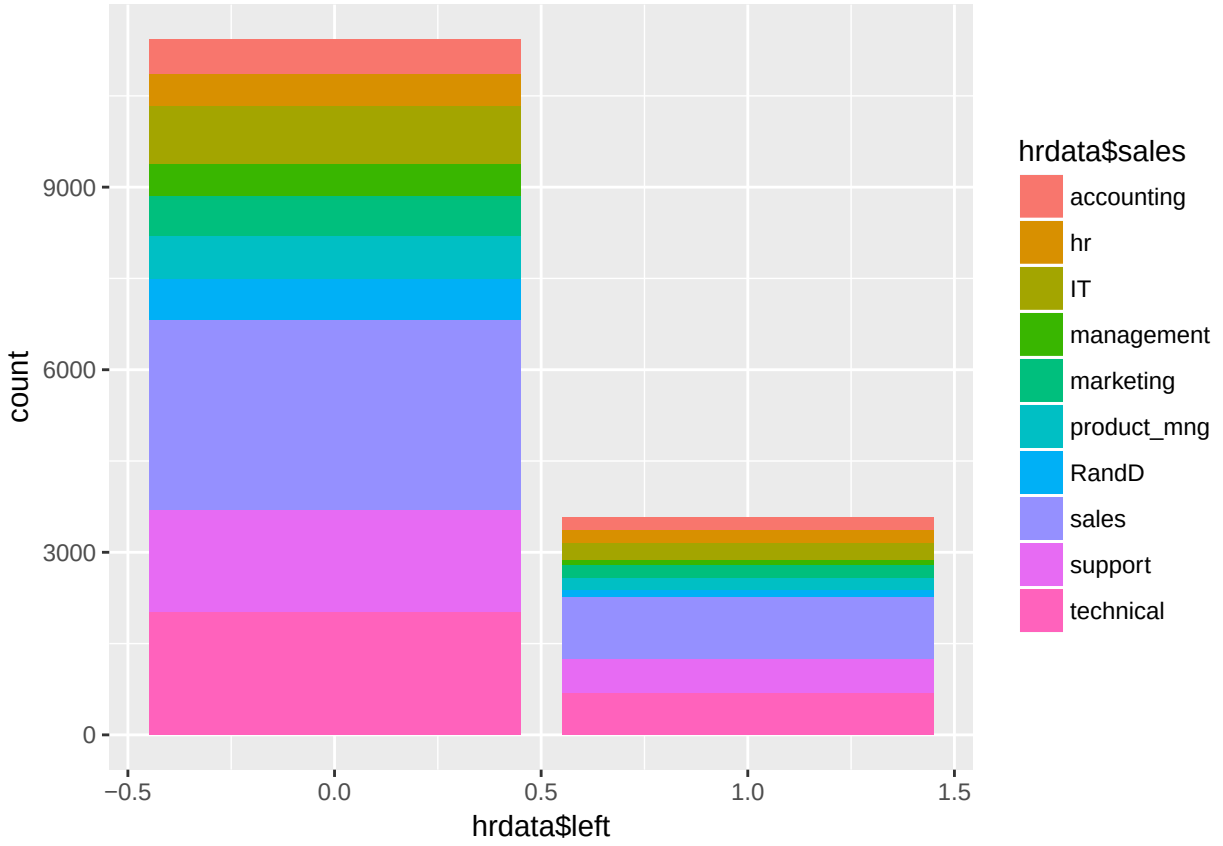

1500000
1000000
500000
0
average_montly_hours
high
low
medium
Salary




```
#clear
#boxplot(formula=retail_raw[,i]~retail_raw$Sale.Made,data=retail_raw, ylab=colnames(retail_raw)[i])+lab

plot(hrdata[,hrquant])
```




```
#cor(hrdata[,hrquant])
#there is no need to check the collinearlity between x and y
library(car)

#scatterplot.matrix(hrdataquant) #taking too much time
#library(corrgram)
#library(corrplot)
#corrgram(hrdataquant)
#corrplot(cor(hrdataquant),method="number")
#correlation does show some negative relation between left and word_accident,also
#some positive relation between left and time_spend, though the relations are not
#strong enough to comment as of now. we will dive deep going forward.

ggplot(data = hrdata, aes(x = hrdata$left, fill = hrdata$sales)) +
  geom_bar()
```




```
# ggplot(data = TTM, aes(x = Type.of.Behavior, y = Sample.Size, fill = Stage.of.Change)) +
#   geom_bar() + coord_flip()
#
# df %>%
#   gather(variable, value, F1:F3) %>%
#   ggplot(aes(x = Rank, y = value, fill = variable)) +
#   geom_bar(stat = "identity")


# Base on above analysis we will build logistic model to see which employee will
# leave the organization.

#check if response variable is of factor type or not
class(hrdata$left)
```

```
## [1] "integer"
```

```
#it is integer, lets convert it into factor.

###############################################################################
############################Model Building#####################################

#Start building the basic naive model for the benchmark.
set.seed(777)
# we will subset based on categories - after basic model
rownumbers <- sample(1:nrow(hrdata),0.75*nrow(hrdata))
hrtrain <- hrdata[rownumbers,]
```

```
hrtest <- hrdata[-rownumbers,]

nrow(hrtrain)
```

```
## [1] 11249
```

```
table(hrtrain$left)
```

```
##
##    0    1
## 8616 2633
```

```
nrow(hrtest)
```

```
## [1] 3750
```

```
table(hrtest$left)
```

```
##
##    0    1
## 2812  938
```

```
#will calculate percentage ratio of 0 and 1

glmmodel <- glm(formula = left~.,family = "binomial",data = hrtrain)
summary(glmmodel)
```

```
##
## Call:
## glm(formula = left ~ ., family = "binomial", data = hrtrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1646  -0.6582  -0.4009  -0.1329   3.0108
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.3962973  0.2263106  -6.170 6.84e-10 ***
## satisfaction_level    -4.2471786  0.1145470 -37.078  < 2e-16 ***
## last_evaluation        0.7884126  0.1731555   4.553 5.28e-06 ***
## number_project        -0.3067882  0.0246311 -12.455  < 2e-16 ***
## average_montly_hours   0.0039395  0.0005934   6.639 3.15e-11 ***
## time_spend_company     0.2487930  0.0179574  13.855  < 2e-16 ***
## Work_accident         -1.4673914  0.1020736 -14.376  < 2e-16 ***
## promotion_last_5years -1.2055708  0.2849986  -4.230 2.34e-05 ***
## saleshr                0.3431734  0.1525328   2.250  0.02446 *
## salesIT               -0.0667481  0.1423662  -0.469  0.63918
## salesmanagement       -0.2545218  0.1843702  -1.380  0.16743
## salesmarketing        -0.0342590  0.1552657  -0.221  0.82537
## salesproduct_mng      -0.0367359  0.1521276  -0.241  0.80918
## salesRandD            -0.4725524  0.1689924  -2.796  0.00517 **
## salessales             0.0220734  0.1203826   0.183  0.85452
## salessupport           0.0851803  0.1280735   0.665  0.50599
## salestechnical         0.1318381  0.1249850   1.055  0.29150
## salarylow              1.9260461  0.1493075  12.900  < 2e-16 ***
## salarymedium           1.4092432  0.1501477   9.386  < 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: 12242.1  on 11248  degrees of freedom
## Residual deviance:  9552.9  on 11230  degrees of freedom
## AIC: 9590.9
##
## Number of Fisher Scoring iterations: 5
```

```
#Some of the department are not significant as per the GLM model
#Deviance is a measure of goodness of fit of a generalized linear model
#Or rather, it's a measure of badness of fit-higher numbers indicate worse fit.
#he null deviance shows how well the response variable is
#predicted by a model that includes only the intercept (grand mean)
#There is not a good gap between Null Deviance and Residual Deviance,
# thus this is not a good fit, though we will do validation further.


#Checking the multicollinearity in the data
library(usdm)
```

```
## Warning: package 'usdm' was built under R version 3.4.1
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 3.4.3
```

```
## Loading required package: raster
```

```
## Warning: package 'raster' was built under R version 3.4.3
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
##
## Attaching package: 'usdm'
```

```
## The following object is masked from 'package:car':
##
##     vif
```

```
vifstep(hrtrain[,-c(9,10)],th=2)
```

```
## No variable from the 8 input variables has collinearity problem.
##
## The linear correlation coefficients ranges between:
## min correlation ( left ~ last_evaluation ):  0.0007530489
## max correlation ( average_montly_hours ~ number_project ):  0.4000562
##
## ---------- VIFs of the remained variables --------
##               Variables      VIF
```

```
## 1    satisfaction_level 1.247208
## 2       last_evaluation 1.223596
## 3        number_project 1.342240
## 4  average_montly_hours 1.257949
## 5    time_spend_company 1.079428
## 6         Work_accident 1.024341
## 7                  left 1.233451
## 8 promotion_last_5years 1.013378
```

```
#there is no issue of multicollinearty in the data.


#lets see if we get something better from AIC
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.4.3
```

```
##
## Attaching package: 'MASS'
```

```
## The following objects are masked from 'package:raster':
##
##     area, select
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
glmaicmodel <- stepAIC(glmmodel,direction = "both")
```

```
## Start:  AIC=9590.94
## left ~ satisfaction_level + last_evaluation + number_project +
##     average_montly_hours + time_spend_company + Work_accident +
##     promotion_last_5years + sales + salary
##
##                         Df Deviance     AIC
## <none>                       9552.9  9590.9
## - sales                  9   9587.6  9607.6
## - last_evaluation        1   9573.8  9609.8
## - promotion_last_5years  1   9575.6  9611.6
## - average_montly_hours   1   9597.4  9633.4
## - number_project         1   9714.6  9750.6
## - time_spend_company     1   9740.4  9776.4
## - Work_accident          1   9821.2  9857.2
## - salary                 2   9833.1  9867.1
## - satisfaction_level     1  11248.0 11284.0
```

```
glmaicmodel$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## left ~ satisfaction_level + last_evaluation + number_project +
##     average_montly_hours + time_spend_company + Work_accident +
##     promotion_last_5years + sales + salary
##
## Final Model:
```

```
## left ~ satisfaction_level + last_evaluation + number_project +
##     average_montly_hours + time_spend_company + Work_accident +
##     promotion_last_5years + sales + salary
##
##
##   Step Df Deviance Resid. Df Resid. Dev      AIC
## 1                     11230   9552.943 9590.943
```

```
#It gives the same model as earlier model- no change
summary(glmmodel)
```

```
##
## Call:
## glm(formula = left ~ ., family = "binomial", data = hrtrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1646  -0.6582  -0.4009  -0.1329   3.0108
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.3962973  0.2263106  -6.170 6.84e-10 ***
## satisfaction_level    -4.2471786  0.1145470 -37.078  < 2e-16 ***
## last_evaluation        0.7884126  0.1731555   4.553 5.28e-06 ***
## number_project        -0.3067882  0.0246311 -12.455  < 2e-16 ***
## average_montly_hours   0.0039395  0.0005934   6.639 3.15e-11 ***
## time_spend_company     0.2487930  0.0179574  13.855  < 2e-16 ***
## Work_accident         -1.4673914  0.1020736 -14.376  < 2e-16 ***
## promotion_last_5years -1.2055708  0.2849986  -4.230 2.34e-05 ***
## saleshr                0.3431734  0.1525328   2.250  0.02446 *
## salesIT               -0.0667481  0.1423662  -0.469  0.63918
## salesmanagement       -0.2545218  0.1843702  -1.380  0.16743
## salesmarketing        -0.0342590  0.1552657  -0.221  0.82537
## salesproduct_mng      -0.0367359  0.1521276  -0.241  0.80918
## salesRandD            -0.4725524  0.1689924  -2.796  0.00517 **
## salessales             0.0220734  0.1203826   0.183  0.85452
## salessupport           0.0851803  0.1280735   0.665  0.50599
## salestechnical         0.1318381  0.1249850   1.055  0.29150
## salarylow              1.9260461  0.1493075  12.900  < 2e-16 ***
## salarymedium           1.4092432  0.1501477   9.386  < 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: 12242.1  on 11248  degrees of freedom
## Residual deviance:  9552.9  on 11230  degrees of freedom
## AIC: 9590.9
##
## Number of Fisher Scoring iterations: 5
```

```
# The logistic regression coefficients give the change in the log odds of the
# outcome for a unit increase in the predictor variable.
# The indicator variables for Sales Departments have a slightly different interpretation.
# For example, having Employee from HR Department versus the Employee from
```

```
# Accounting(baseline) Department , changes the log odds of leaving the company
# by 0.343, i.e changes the odds of leaving the company by 1.41 , so the
# probability of HR employee leaving the company is 58.5%  as compared to Accounting employee
##Note that while R produces it, the odds ratio for the intercept is not generally interpreted.


#calculate the odds ratios for each preditor.
exp((glmmodel$coefficients))
```

```
##           (Intercept)    satisfaction_level       last_evaluation
##             0.24751173             0.01430454             2.19990144
##        number_project  average_montly_hours    time_spend_company
##             0.73580643             1.00394723             1.28247651
##         Work_accident promotion_last_5years              saleshr
##             0.23052605             0.29952100             1.40941313
##                salesIT       salesmanagement        salesmarketing
##             0.93543076             0.77528713             0.96632119
##      salesproduct_mng            salesRandD            salessales
##             0.96393067             0.62340903             1.02231882
##          salessupport        salestechnical             salarylow
##             1.08891343             1.14092359             6.86232349
##          salarymedium
##             4.09285675
```

```
perunitpercentchange<-100/(1+exp(-glmmodel$coefficients))
perunitpercentchange
```

```
##           (Intercept)    satisfaction_level       last_evaluation
##               19.84043                1.41028               68.74904
##        number_project  average_montly_hours    time_spend_company
##               42.38989               50.09849               56.18794
##         Work_accident promotion_last_5years              saleshr
##               18.73394               23.04857               58.49612
##                salesIT       salesmanagement        salesmarketing
##               48.33192               43.67108               49.14361
##      salesproduct_mng            salesRandD            salessales
##               49.08171               38.40123               50.55181
##          salessupport        salestechnical             salarylow
##               52.12822               53.29119               87.28111
##          salarymedium
##               80.36465
```

```
#satisfaction_level - there will be decrease of 1.41% of the employee
# leaving the company with every unit increase of satisfaction level.
#last_evaluation - Employee leaving the company by 68.74% with every
# unit of increase in last_evaluation.

## odds ratios and 95% CI
exp(cbind(OR=glmmodel$coefficients,confint(glmmodel)))
```

```
## Waiting for profiling to be done...
```

```
##                              OR      2.5 %     97.5 %
## (Intercept)          0.24751173 0.15806419 0.38399717
## satisfaction_level   0.01430454 0.01141034 0.01787833
## last_evaluation      2.19990144 1.56739067 3.09027958
```

```
## number_project        0.73580643 0.70099250 0.77205948
## average_montly_hours  1.00394723 1.00278222 1.00511766
## time_spend_company    1.28247651 1.23811962 1.32844022
## Work_accident         0.23052605 0.18797975 0.28054561
## promotion_last_5years 0.29952100 0.16539778 0.50869940
## saleshr               1.40941313 1.04555126 1.90163345
## salesIT               0.93543076 0.70810150 1.23755352
## salesmanagement       0.77528713 0.53850001 1.11000420
## salesmarketing        0.96632119 0.71264850 1.31015825
## salesproduct_mng      0.96393067 0.71535627 1.29905524
## salesRandD            0.62340903 0.44671751 0.86680956
## salessales            1.02231882 0.80894684 1.29706057
## salessupport          1.08891343 0.84845422 1.40202766
## salestechnical        1.14092359 0.89452793 1.46037693
## salarylow             6.86232349 5.16240991 9.27724125
## salarymedium          4.09285675 3.07324544 5.54114539
```

```
# We can use the confint function to obtain confidence intervals for the
# coefficient estimates. Note that for logistic models, confidence intervals
# are based on the profiled log-likelihood function. We can also get CIs based
# on just the standard errors by using the default method.

## CIs using profiled log-likelihood
confint(glmmodel)
```

```
## Waiting for profiling to be done...
```

```
##                              2.5 %       97.5 %
## (Intercept)           -1.84475403 -0.957120105
## satisfaction_level    -4.47323506 -4.024166026
## last_evaluation        0.44941224  1.128261566
## number_project        -0.35525809 -0.258693679
## average_montly_hours   0.00277836  0.005104614
## time_spend_company     0.21359379  0.284005487
## Work_accident         -1.67142106 -1.271018950
## promotion_last_5years -1.79940190 -0.675898000
## saleshr                0.04454427  0.642713227
## salesIT               -0.34516783  0.213136461
## salesmanagement       -0.61896777  0.104363795
## salesmarketing        -0.33876696  0.270147933
## salesproduct_mng      -0.33497459  0.261637260
## salesRandD            -0.80582884 -0.142935976
## salessales            -0.21202207  0.260100601
## salessupport          -0.16433915  0.337919517
## salestechnical        -0.11145915  0.378694575
## salarylow              1.64140351  2.227564223
## salarymedium           1.12273415  1.712201228
```

```
## CIs using standard errors
confint.default(glmmodel)
```

```
##                               2.5 %       97.5 %
## (Intercept)           -1.839857929 -0.952736675
## satisfaction_level    -4.471686500 -4.022670648
## last_evaluation        0.449034100  1.127791022
## number_project        -0.355064216 -0.258512186
```

```
## average_montly_hours   0.002776494  0.005102428
## time_spend_company     0.213597064  0.283988899
## Work_accident         -1.667451919 -1.267330857
## promotion_last_5years -1.764157789 -0.646983715
## saleshr                0.044214662  0.642132127
## salesIT               -0.345780842  0.212284545
## salesmanagement       -0.615880826  0.106837176
## salesmarketing        -0.338574207  0.270056200
## salesproduct_mng      -0.334900539  0.261428735
## salesRandD            -0.803771407 -0.141333441
## salessales            -0.213872170  0.258018963
## salessupport          -0.165839157  0.336199841
## salestechnical        -0.113128038  0.376804239
## salarylow              1.633408830  2.218683342
## salarymedium           1.114959081  1.703527317
```

```
#model2 <- train(left~.,data = hrtrain,method = "glm",family="binomial")
#model2$finalModel
#model1$coefficients
# Prediction/Evaluation of the glmmodel

##1. Accuracy of the glmmodel
glmpredict <- predict(object = glmmodel,newdata = hrtest, type = "response")
glmclass <- ifelse(glmpredict>0.5,1,0)
table(glmclass,hrtest$left)
```

```
##
## glmclass    0    1
##        0 2602  600
##        1  210  338
```

```
cat("Accuracy of glm model with 0.5 as cutoff")
```

```
## Accuracy of glm model with 0.5 as cutoff
```

```
sum(diag(table(hrtest$left,glmclass)))/nrow(hrtest)
```

```
## [1] 0.784
```

```
# cat("Miss classification Error")
# misClassError(hrtest$left, glmpredict)
# #quite high misclassificaiton

library(InformationValue)
```

```
## Warning: package 'InformationValue' was built under R version 3.4.4
```

```
optCutOff<- optimalCutoff(hrtest$left, glmpredict)[1]
optCutOff
```

```
## [1] 0.4239428
```

```
glmclass <- ifelse(glmpredict>optCutOff,1,0)
table(glmclass,hrtest$left)
```

```
##
## glmclass    0    1
##        0 2530  453
```

```
##        1  282  485
```

```
cat("Accuracy of glm model with optCutOff as cutoff")
```

```
## Accuracy of glm model with optCutOff as cutoff
```

```
sum(diag(table(hrtest$left,glmclass)))/nrow(hrtest)
```

```
## [1] 0.804
```

```
cat("Miss classification Error with optimal cutoff")
```

```
## Miss classification Error with optimal cutoff
```

```
misClassError(hrtest$left, glmpredict, threshold = optCutOff)
```

```
## [1] 0.196
```

```
#quite high misclassificaiton

library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:InformationValue':
##
##     confusionMatrix, precision, sensitivity, specificity
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
confusionMatrix(table(glmclass,hrtest$left,dnn=list("Predicted","Actual")))
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted    0    1
##         0 2530  453
##         1  282  485
##
##                Accuracy : 0.804
##                  95% CI : (0.7909, 0.8166)
##     No Information Rate : 0.7499
##     P-Value [Acc > NIR] : 2.277e-15
##
##                   Kappa : 0.4437
##  Mcnemar's Test P-Value : 3.598e-10
##
##             Sensitivity : 0.8997
##             Specificity : 0.5171
##          Pos Pred Value : 0.8481
##          Neg Pred Value : 0.6323
##              Prevalence : 0.7499
##          Detection Rate : 0.6747
```

```
##    Detection Prevalence : 0.7955
##       Balanced Accuracy : 0.7084
##
##        'Positive' Class : 0
##
```

```
#So the accuracy of our glm model is 80.4%, as we have calculated from
# different method above.

##2. Goodness of Fit - Hosmer Lemeshow Test


library(ResourceSelection)
```

```
## Warning: package 'ResourceSelection' was built under R version 3.4.4
```

```
## ResourceSelection 0.3-2   2017-02-28
```

```
hoslem.test(hrtrain$left,fitted(glmmodel),g=10)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  hrtrain$left, fitted(glmmodel)
## X-squared = 241.01, df = 8, p-value < 2.2e-16
```

```
#since p-value is less than 0.05, so we reject the null hypothesis, i.e
# the current model is not a good fit as shown by Hosmer & Lemeshow test.

##3. evaluate the statistical significance of each coffecient in the model
# - Wald test
# h0: - coffecients are zeros/insignificant
# h1 :- coffecients of the independent variabe in the model is significantly
# different from 0.
#if the test fails to reject the null  hypothesis ,this suggests that
# removing the variable from the model  will not  substantially harm the
# fit of the model.


library(survey)
```

```
## Warning: package 'survey' was built under R version 3.4.3
```

```
## Loading required package: grid
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
```

```
##
##     cluster
```

```
##
## Attaching package: 'survey'
```

```
## The following object is masked from 'package:raster':
##
##     cv
```

```
## The following object is masked from 'package:graphics':
##
##     dotchart
```

```
#regTermTest(glmmodel,"satisfaction_level")

for (i in c(1:ncol(hrtrain[,-7]))){
  if (i==7){
    #cat("Significance of Response Variable is not required")
  }else{
  print (regTermTest(glmmodel,colnames(hrtrain)[i]))
  }
}
```

```
## Wald test for satisfaction_level
##  in glm(formula = left ~ ., family = "binomial", data = hrtrain)
## F =  1374.782  on  1  and  11230  df: p= < 2.22e-16
## Wald test for last_evaluation
##  in glm(formula = left ~ ., family = "binomial", data = hrtrain)
## F =  20.73169  on  1  and  11230  df: p= 5.339e-06
## Wald test for number_project
##  in glm(formula = left ~ ., family = "binomial", data = hrtrain)
## F =  155.1353  on  1  and  11230  df: p= < 2.22e-16
## Wald test for average_montly_hours
##  in glm(formula = left ~ ., family = "binomial", data = hrtrain)
## F =  44.07933  on  1  and  11230  df: p= 3.2987e-11
## Wald test for time_spend_company
##  in glm(formula = left ~ ., family = "binomial", data = hrtrain)
## F =  191.9499  on  1  and  11230  df: p= < 2.22e-16
## Wald test for Work_accident
##  in glm(formula = left ~ ., family = "binomial", data = hrtrain)
## F =  206.6642  on  1  and  11230  df: p= < 2.22e-16
## Wald test for promotion_last_5years
##  in glm(formula = left ~ ., family = "binomial", data = hrtrain)
## F =  17.89369  on  1  and  11230  df: p= 2.3545e-05
## Wald test for sales
##  in glm(formula = left ~ ., family = "binomial", data = hrtrain)
## F =  3.728433  on  9  and  11230  df: p= 0.00010854
```

```
#All the predictors in the model are significant as per the wald test.
#As P-value is less than 0.05 thus we reject null hypothesis, i.e removing
# any predictor from the model will harm the fit of the model.


## we will use aod package to perform wald test to check overall effect of sales
## and salary predictors..
#install.packages("aod")
```

```
library(aod)
```

```
## Warning: package 'aod' was built under R version 3.4.4
```

```
##
## Attaching package: 'aod'
```

```
## The following object is masked from 'package:survival':
##
##     rats
```

```
wald.test(b=coef(glmmodel),Sigma = vcov(glmmodel), Terms = 9:17)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 33.6, df = 9, P(> X2) = 0.00011
```

```
# The chi-squared test statistic of 33.6, with 9 degrees of freedom is
# associated with a p-value of 0.00011 indicating that the overall effect of
# Sales is statistically significant.

wald.test(b=coef(glmmodel),Sigma = vcov(glmmodel), Terms = 18:19)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 226.6, df = 2, P(> X2) = 0.0
```

```
# The chi-squared test statistic of 226.6, with 2 degrees of freedom is
# associated with a p-value of 0.0 indicating that the overall effect of
# Salary is statistically significant.

## we can also test the difference in the coefficient for the different level
## of Sales, like HR and IT

hrit <- cbind(0,0,0,0,0,0,0,0,1,-1,0,0,0,0,0,0,0,0,0) # deifnes the vector for which we want to test.
# To contrast these two terms, we multiply one of them by 1, and the other by
# -1. The other terms in the model are not involved in the test, so they are
# multiplied by 0. The second line of code below uses L=hrit to tell R that we
# wish to base the test on the vector hrit (rather than using the Terms option as
# we did above).
wald.test(b=coef(glmmodel),Sigma = vcov(glmmodel),L=hrit)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 8.7, df = 1, P(> X2) = 0.0031
```

```
# The chi-squared test statistic of 8.7 with 1 degree of freedom is associated
# with a p-value of 0.0031, indicating that the difference between the coefficient
# for Sales=HR and the coefficient for Sales=IT is statistically significant.
#This way we can test the difference between anyother coefficients.
```

```
##4. MC Fadden's R2 Test.
#unlike linear regression, there is no R2 statistic which
# explains the proportion of variation in the dependent
# variable that is explained by the predictors.
# for this we use Mc Fadden's R2

#install.packages("pscl")
library(pscl)
```

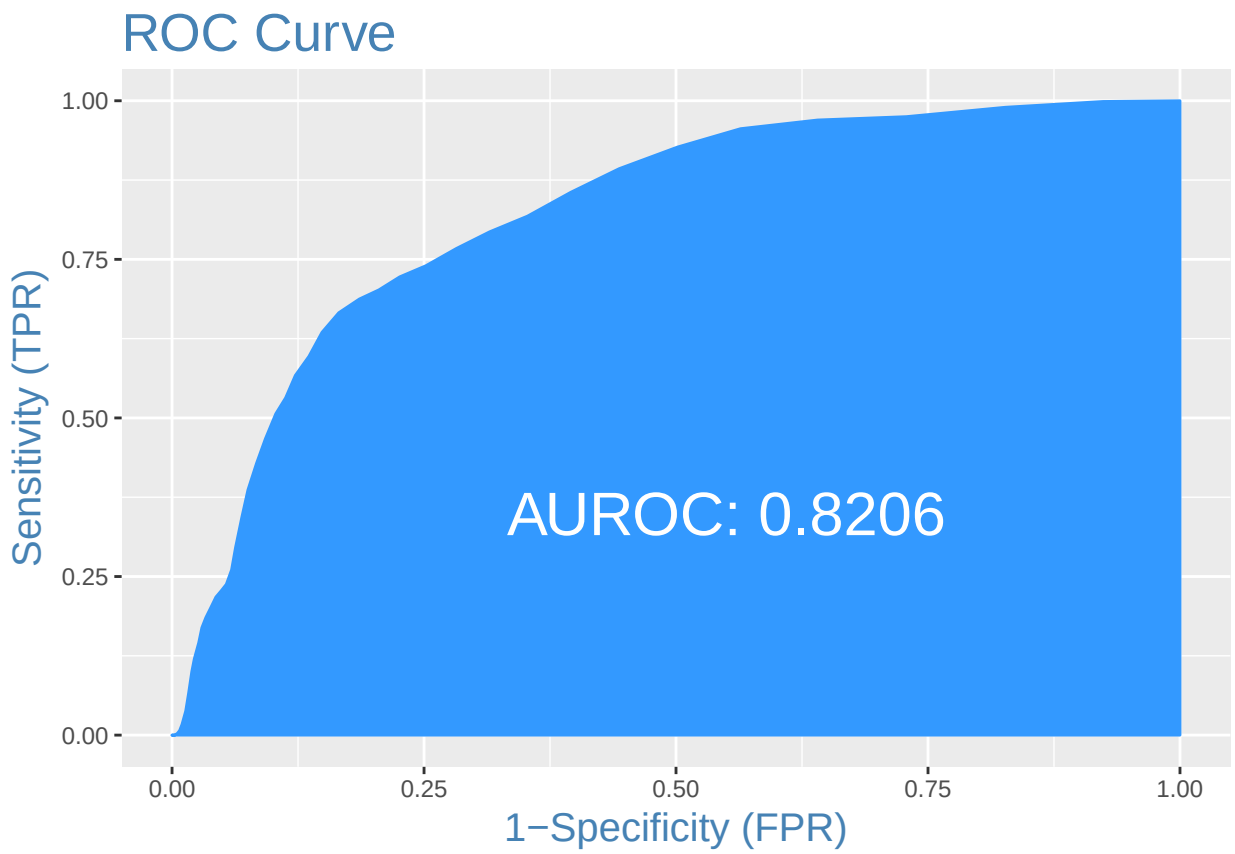
```
## Warning: package 'pscl' was built under R version 3.4.4
```

```
## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

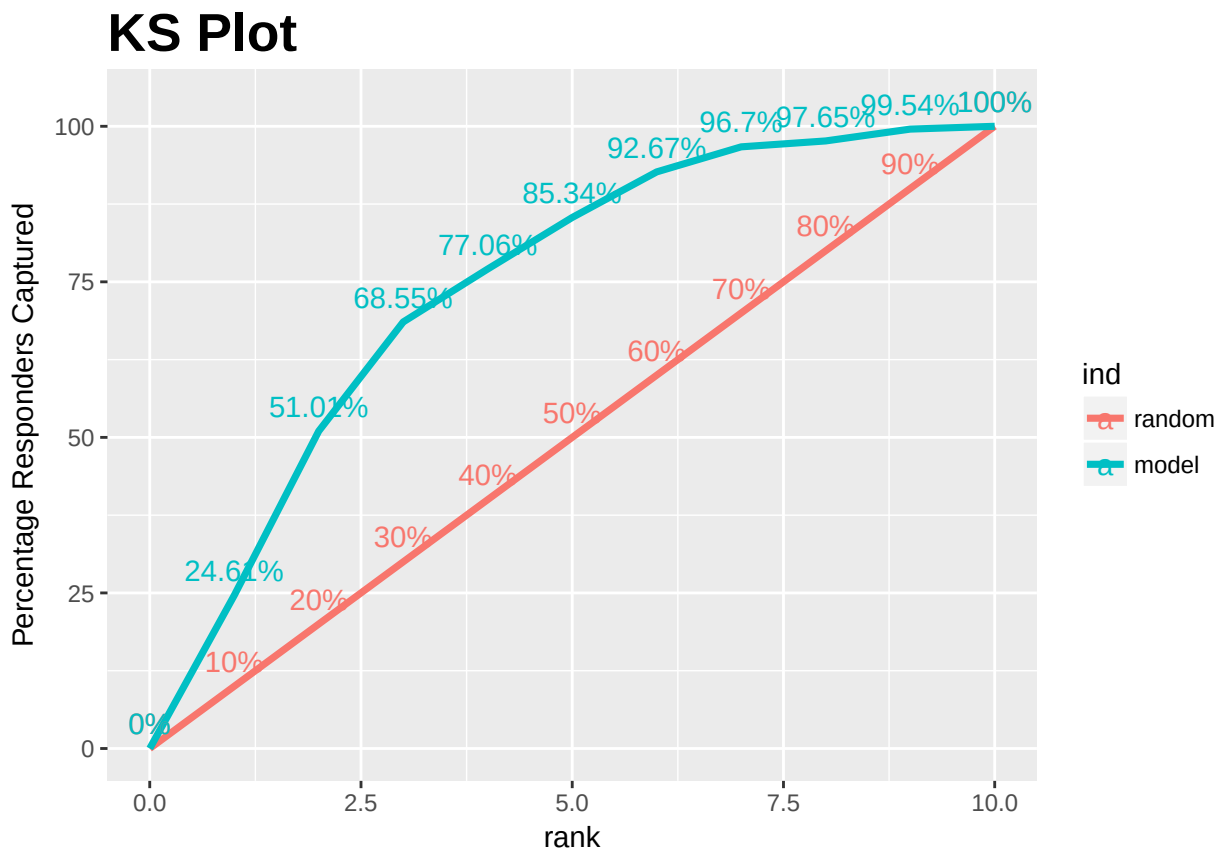
```
pR2(glmmodel)
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -4776.4714983 -6121.0529285  2689.1628603     0.2196651     0.2126307
##          r2CU
##     0.3206104
```

```
#McFadden = .2196
#The predictors in our model explain just the 21.96%
# variation in the data.
# This suggests that one or more variables are missing
# in our model

#5. ROC
#install.packages("InformationValue")
library(InformationValue)
plotROC(actuals = hrtrain$left,predictedScores = as.numeric(fitted(glmmodel)))
```




```
#since the area under curve is 82.06, we can say  the correctly identification
# ability of our model is good.

#6. Kolmogrov Smirnov Chart
#measure the performance of classification models.
# It is a measure of degree of seperation between left(1) and not-left(0)
# Distance between left and not-left should be as large as possible
#The random line in the chart corresponds to the case  of capturing the responders(1)
# by random selection,i.e  when you dont have any model at disposal.
#The model line represent the case of capturing the responders if you go by the model
# generated probability scores where you begin by targeting datapoint with
# highest probability scores.

ks_plot(actuals = hrtrain$left,predictedScores = as.numeric(fitted(glmmodel)))
```




```
# as the measure between left and not-left is not so large, we can say that
# performance of the model is not good.
# Kolmogoru Smirnov statistic is the maximum  difference between the cumulative
# true positive and cumulative false positive rate
# the hihger the value of kolmogrov statistic,  the more efficient is the
# model at capturing the responders(1).
ks_stat(actuals = hrtrain$left,predictedScores = as.numeric(fitted(glmmodel)))
```

```
## [1] 0.5033
```

```
#As the kolmogrov Smirnov statistic is .5033 which is small, we can say
# that the model is not that efficient in capturing  the responders(1).

#7. F1 score - accuracy test.
hrtr <- hrtrain
hrtr$fitted_label <- glmmodel$fitted.values
#hrtrain$fitted_label <- glmmodel$fitted.values
hrtr$fitted_label <- ifelse(hrtr$fitted_label>optCutOff,1,0)
optCutOff
```

```
## [1] 0.4239428
```

```
tp = sum((hrtr$left==1)&(hrtr$fitted_label==1))
tp
```

```
## [1] 1317
```

```r
fp = sum((hrtr$left==0)&(hrtr$fitted_label==1))
fp
```

```
## [1] 869
```

```r
prec = tp/(tp+fp)
cat("The precision is: ")
```

```
## The precision is:
```

```r
prec
```

```
## [1] 0.6024703
```

```r
#tp = sum((hrtrain1$left==1)&(hrtrain1$fitted_label==1))
fn = sum((hrtr$left==1)&(hrtr$fitted_label==0))
rec = tp/(tp+fn)
cat("The recall is: ")
```

```
## The recall is:
```

```r
rec
```

```
## [1] 0.5001899
```

```r
F1 = 2*(prec*rec)/(prec+rec)
cat("The F1 score is: ")
```

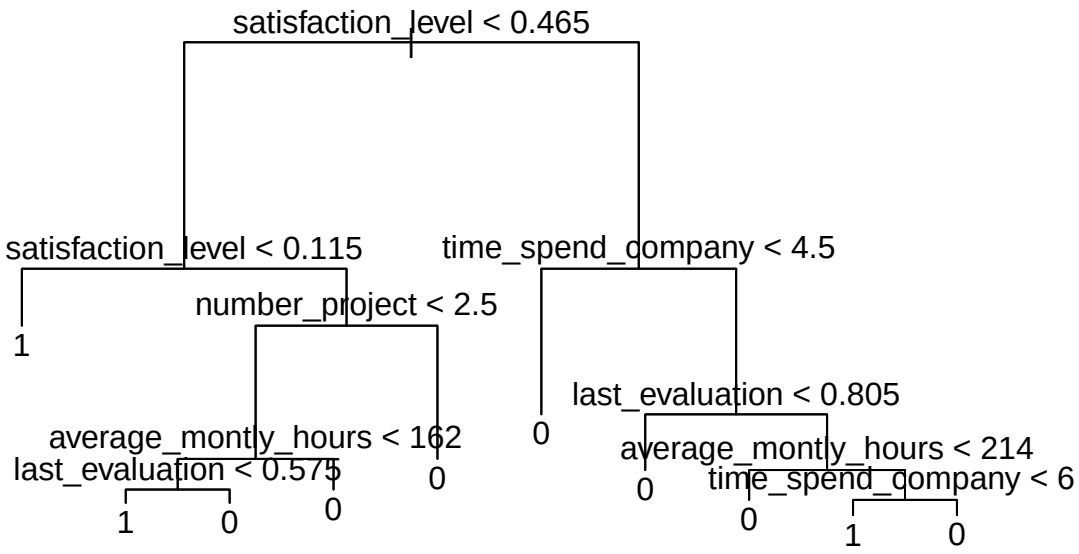
```
## The F1 score is:
```

```r
F1
```

```
## [1] 0.5465864
```

```r
#F1 score (also F-score or F-measure) is a measure of a test's accuracy.
# It considers both the precision p and the recall r of the test to compute
# the score: p is the number of correct positive results divided by the number
# of all positive results returned by the classifier, and r is the number of
# correct positive results divided by the number of all relevant samples
# (all samples that should have been identified as positive). The F1 score
# is the harmonic average of the precision and recall, where an F1 score
# reaches its best value at 1 (perfect precision and recall) and worst at 0.
#This also suggest the test's accuracy of the model is not good.

################################################################################
############################2. Decision Tree#####################################
#Decision Tree using tree()
#install.packages("tree")
library(tree)
hrtrain$salary <- as.factor(hrtrain$salary)
hrtrain$sales <- as.factor(hrtrain$sales)
hrtrain$left <- as.factor(hrtrain$left)
hrtest$left <- as.factor(hrtest$left)
hrtest$salary <- as.factor(hrtest$salary)
hrtest$sales <- as.factor(hrtest$sales)
#Build tree on train data

DT_Model = tree(left~.,data=hrtrain)
#summary(DT_Model)
#predict on test data
```

```
tree.pred = predict(DT_Model,newdata=hrtest,type="class")

#plot the tree to check how the tree model is

plot(DT_Model)
text(DT_Model,pretty=1)
```



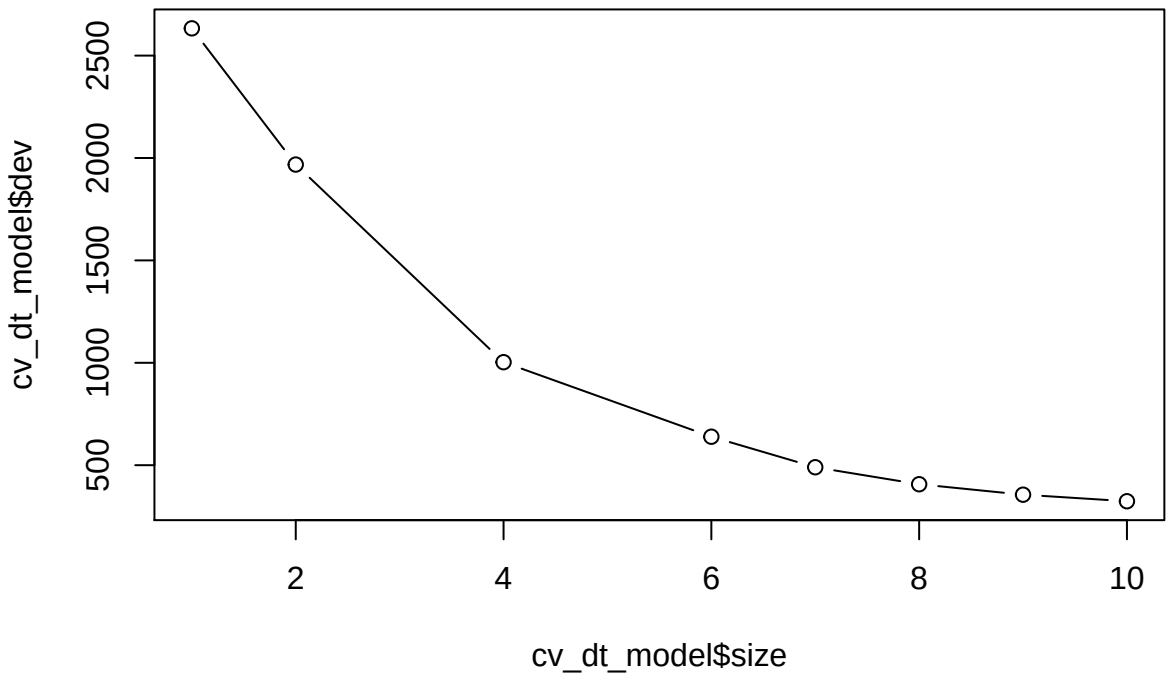

```
#confusion matrix

table(tree.pred,hrtest$left)
```

```
##
## tree.pred    0    1
##         0 2783   80
##         1   29  858
```

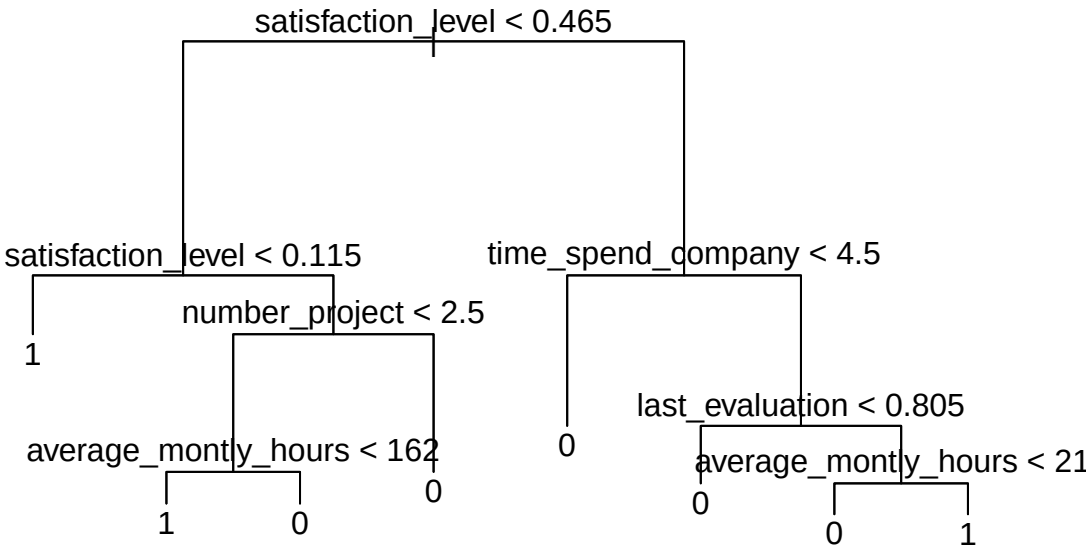
```
sum(diag(table(hrtest$left ,tree.pred)))/nrow(hrtest)
```

```
## [1] 0.9709333
```

```
#prune a tree using cross validation
set.seed(1234)
cv_dt_model=cv.tree(DT_Model,FUN=prune.misclass)

plot(cv_dt_model$size,cv_dt_model$dev,type="b")
```




```
DT_Pruned=prune.misclass(DT_Model,best=8)
plot(DT_Pruned)
text(DT_Pruned,pretty = 8)
```




```
#rpart
library(rpart)

library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.4.4
```

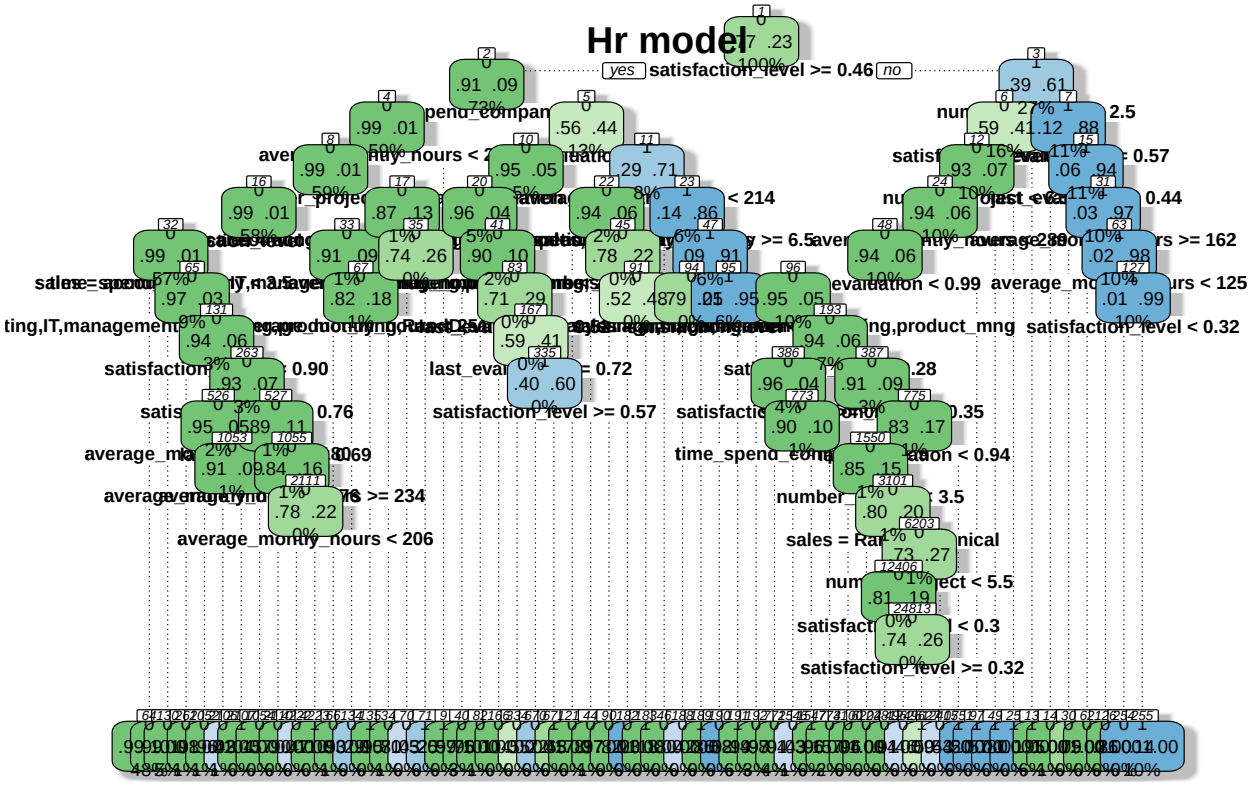
```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
#install.packages("rpart.plot")
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

```
rpartmodel=rpart(formula=left~.,data=hrtrain,method="class",
                 control = rpart.control(cp = 0.0))
#summary(rpartmodel)
windows()
fancyRpartPlot(model=rpartmodel,main="Hr model",cex=0.6)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

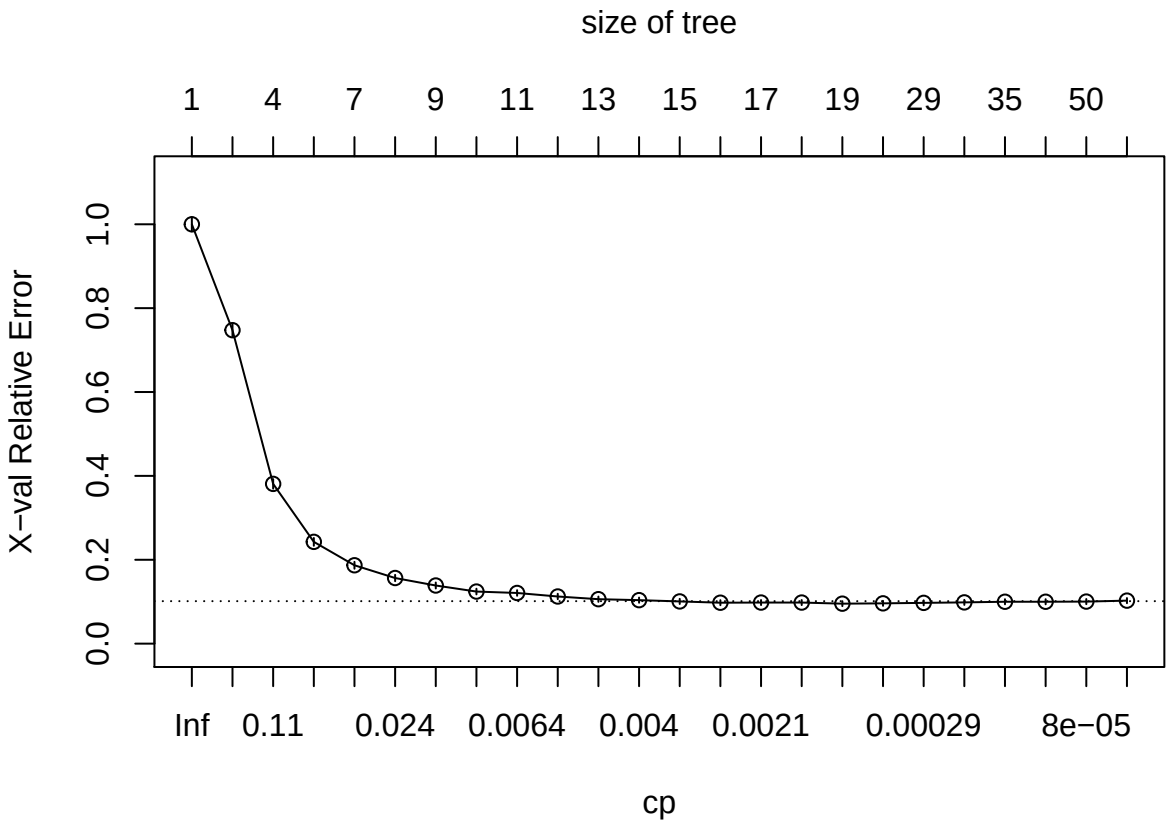



```
printcp(rpartmodel)
```

```
##
## Classification tree:
## rpart(formula = left ~ ., data = hrtrain, method = "class", control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] average_montly_hours last_evaluation      number_project
## [4] salary               sales                satisfaction_level
## [7] time_spend_company
##
## Root node error: 2633/11249 = 0.23407
##
## n= 11249
##
##            CP nsplit rel error   xerror      xstd
## 1  2.5256e-01      0  1.000000 1.000000 0.0170557
## 2  1.8325e-01      1  0.747436 0.747436 0.0153039
## 3  7.0072e-02      3  0.380934 0.380934 0.0114794
## 4  5.6210e-02      5  0.240790 0.242689 0.0093240
## 5  3.2662e-02      6  0.184580 0.186859 0.0082380
## 6  1.7091e-02      7  0.151918 0.156476 0.0075665
## 7  1.2533e-02      8  0.134827 0.138625 0.0071373
## 8  7.2161e-03      9  0.122294 0.124193 0.0067673
## 9  5.6969e-03     10  0.115078 0.120775 0.0066763
## 10 4.9373e-03     11  0.109381 0.112419 0.0064477
```

```
## 11 4.1777e-03     12  0.104444 0.105963 0.0062647
## 12 3.7979e-03     13  0.100266 0.103684 0.0061986
## 13 3.0384e-03     14  0.096468 0.100646 0.0061094
## 14 2.2788e-03     15  0.093430 0.097607 0.0060186
## 15 1.8990e-03     16  0.091151 0.097987 0.0060300
## 16 1.5192e-03     17  0.089252 0.097987 0.0060300
## 17 7.5959e-04     18  0.087733 0.095329 0.0059496
## 18 3.0384e-04     22  0.084694 0.096088 0.0059727
## 19 2.8485e-04     28  0.082795 0.097227 0.0060072
## 20 1.8990e-04     32  0.081656 0.098367 0.0060414
## 21 1.2660e-04     34  0.081276 0.099886 0.0060868
## 22 8.4399e-05     40  0.080517 0.099886 0.0060868
## 23 7.5959e-05     49  0.079757 0.100266 0.0060981
## 24 0.0000e+00     54  0.079377 0.102545 0.0061653
```

```
plotcp(rpartmodel)
```




```
rpartmodel_test=predict(object=rpartmodel,newdata=hrtest,type="class")
#mean(rpartmodel_test==hrtest$left)
#confusion matrix

table(hrtest$left ,rpartmodel_test)
```

```
##    rpartmodel_test
##        0    1
##   0 2772   40
##   1   59  879
```

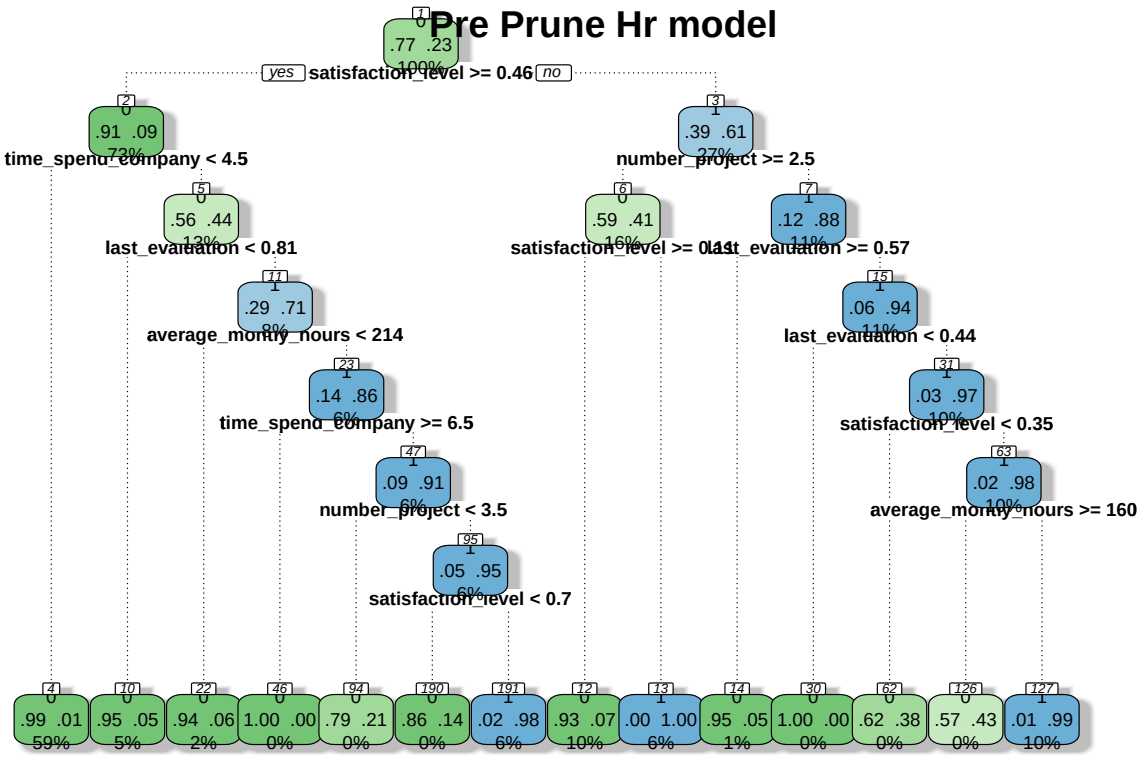
```
#model accuracy on test data
cat("Accuracy of the Decision Tree model is:")
```

```
## Accuracy of the Decision Tree model is:
```

```
baseaccuracy<-sum(diag(table(hrtest$left,rpartmodel_test)))/nrow(hrtest)
baseaccuracy
```

```
## [1] 0.9736
```

```
##Now we will prune tree to stop the tree at optimum
##Pre-prune the tree
rpartmodelpre=rpart(formula=left~.,data=hrtrain,method="class",
                    control = rpart.control(cp = 0.0,maxdepth = 9,minsplit = 100,minbucket = 20))
#summary(rpartmodelpre)
windows()
fancyRpartPlot(model=rpartmodelpre,main="Pre Prune Hr model",cex=0.6)
```

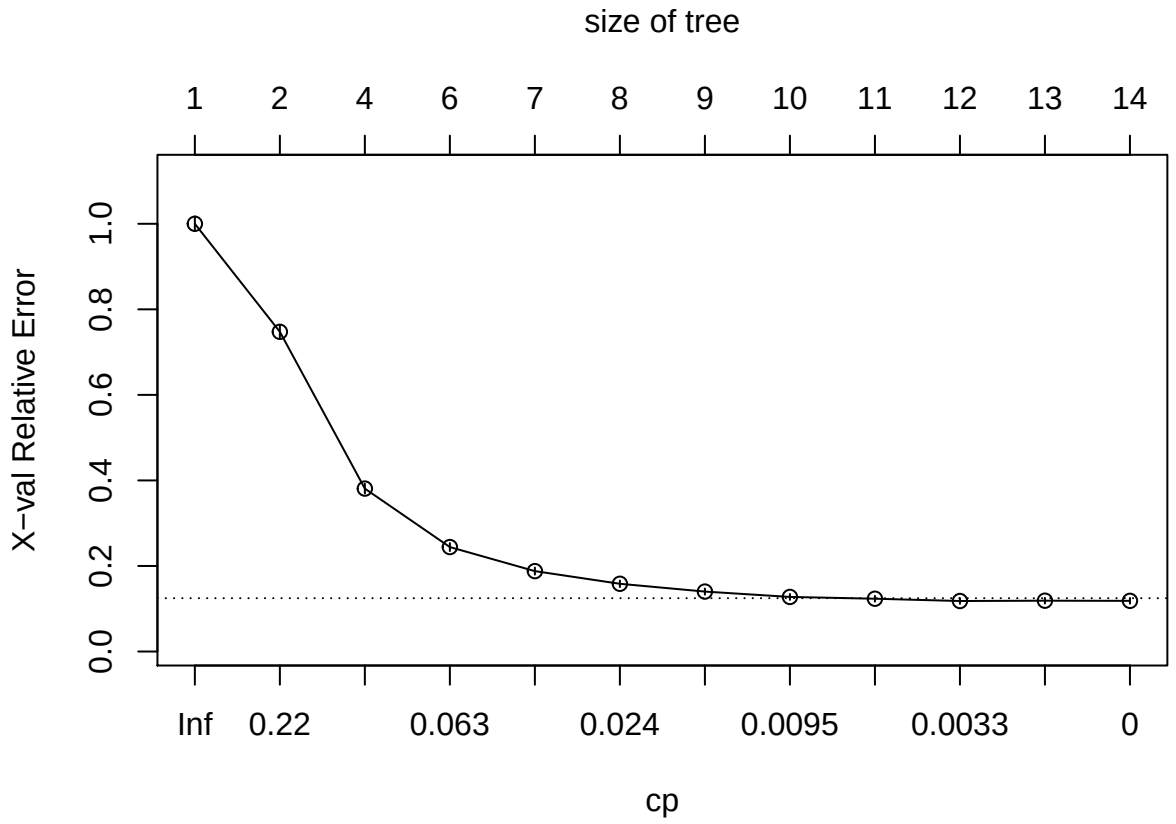



```
printcp(rpartmodelpre)
```

```
##
## Classification tree:
## rpart(formula = left ~ ., data = hrtrain, method = "class", control = rpart.control(cp = 0,
##     maxdepth = 9, minsplit = 100, minbucket = 20))
##
## Variables actually used in tree construction:
## [1] average_montly_hours last_evaluation      number_project
## [4] satisfaction_level   time_spend_company
```

```
##
## Root node error: 2633/11249 = 0.23407
##
## n= 11249
##
##           CP nsplit rel error  xerror      xstd
## 1  0.2525636      0   1.00000 1.00000 0.0170557
## 2  0.1832510      1   0.74744 0.74744 0.0153039
## 3  0.0700722      3   0.38093 0.38093 0.0114794
## 4  0.0562096      5   0.24079 0.24421 0.0093513
## 5  0.0326624      6   0.18458 0.18800 0.0082619
## 6  0.0170908      7   0.15192 0.15837 0.0076105
## 7  0.0125332      8   0.13483 0.14014 0.0071750
## 8  0.0072161      9   0.12229 0.12761 0.0068570
## 9  0.0056969     10   0.11508 0.12343 0.0067472
## 10 0.0018990     11   0.10938 0.11812 0.0066045
## 11 0.0011394     12   0.10748 0.11888 0.0066251
## 12 0.0000000     13   0.10634 0.11850 0.0066148
```

```
plotcp(rpartmodelpre)
```




```
rpartmodelpre_test=predict(object=rpartmodelpre,newdata=hrtest,type="class")
#mean(rpartmodel_test==hrtest$left)
#confusion matrix

table(hrtest$left ,rpartmodelpre_test)
```

```
##    rpartmodelpre_test
##        0    1
##   0 2801   11
##   1   85  853
```

```
#model accuracy on test data
cat("Accuracy of the Pre Prune Decision Tree model is:")
```

```
## Accuracy of the Pre Prune Decision Tree model is:
```

```
preaccuracy<-sum(diag(table(hrtest$left,rpartmodelpre_test)))/nrow(hrtest)
preaccuracy
```

```
## [1] 0.9744
```

```
##Post-Prune the tree
rpartmodelpost <- prune(rpartmodel,cp = 0.0045 )
rpartmodelpost_test=predict(object=rpartmodelpost,newdata=hrtest,type="class")
cat("Accuracy of the Post Prune Decision Tree model is:")
```

```
## Accuracy of the Post Prune Decision Tree model is:
```

```
postaccuracy<-sum(diag(table(hrtest$left,rpartmodelpost_test)))/nrow(hrtest)
postaccuracy
```

```
## [1] 0.9733333
```

```
data.frame(baseaccuracy,preaccuracy,postaccuracy)
```

```
##   baseaccuracy preaccuracy postaccuracy
## 1       0.9736      0.9744    0.9733333
```

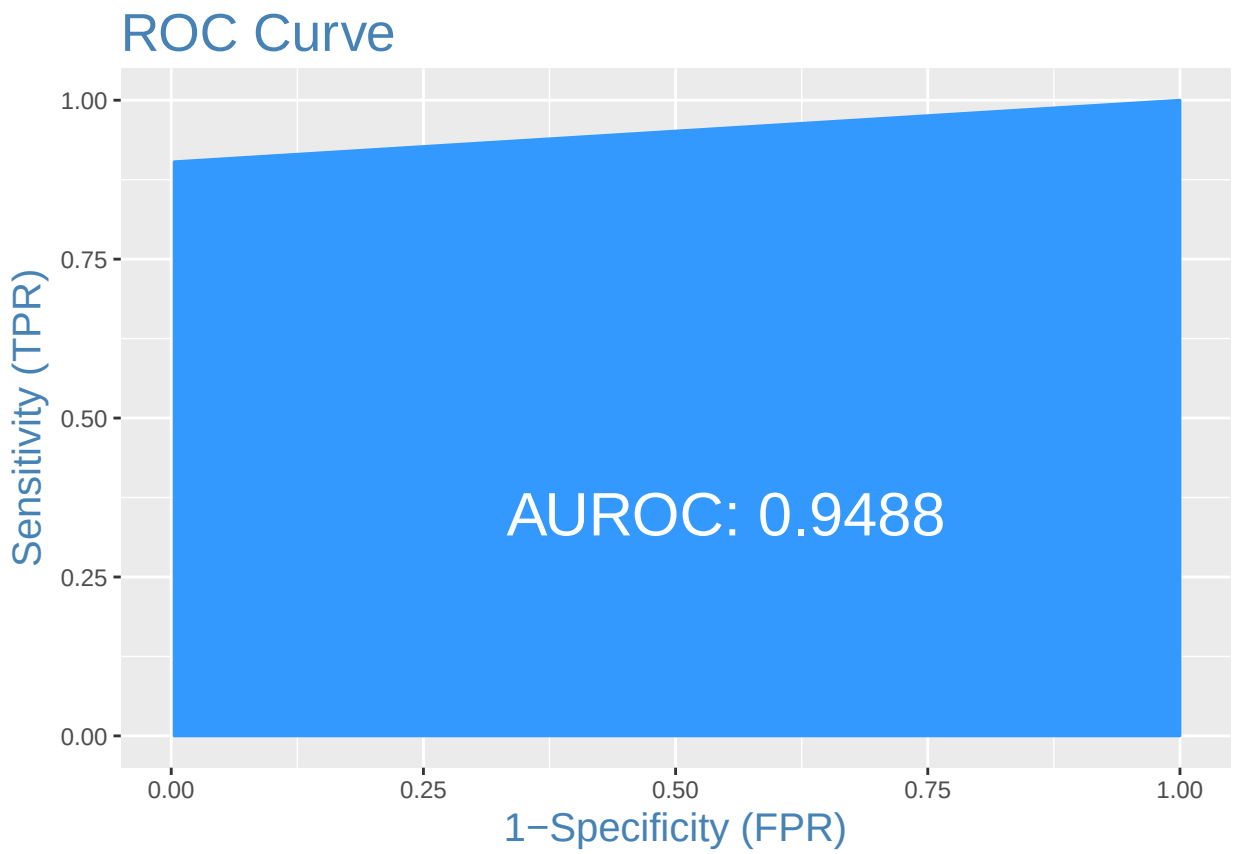
```
#so the best model based on decision tree is the pre-pruned Model.

##1. Accuracy of the rpartmodelpre
cat("Accuracy of the Pre Prune Decision Tree model is:")
```

```
## Accuracy of the Pre Prune Decision Tree model is:
```

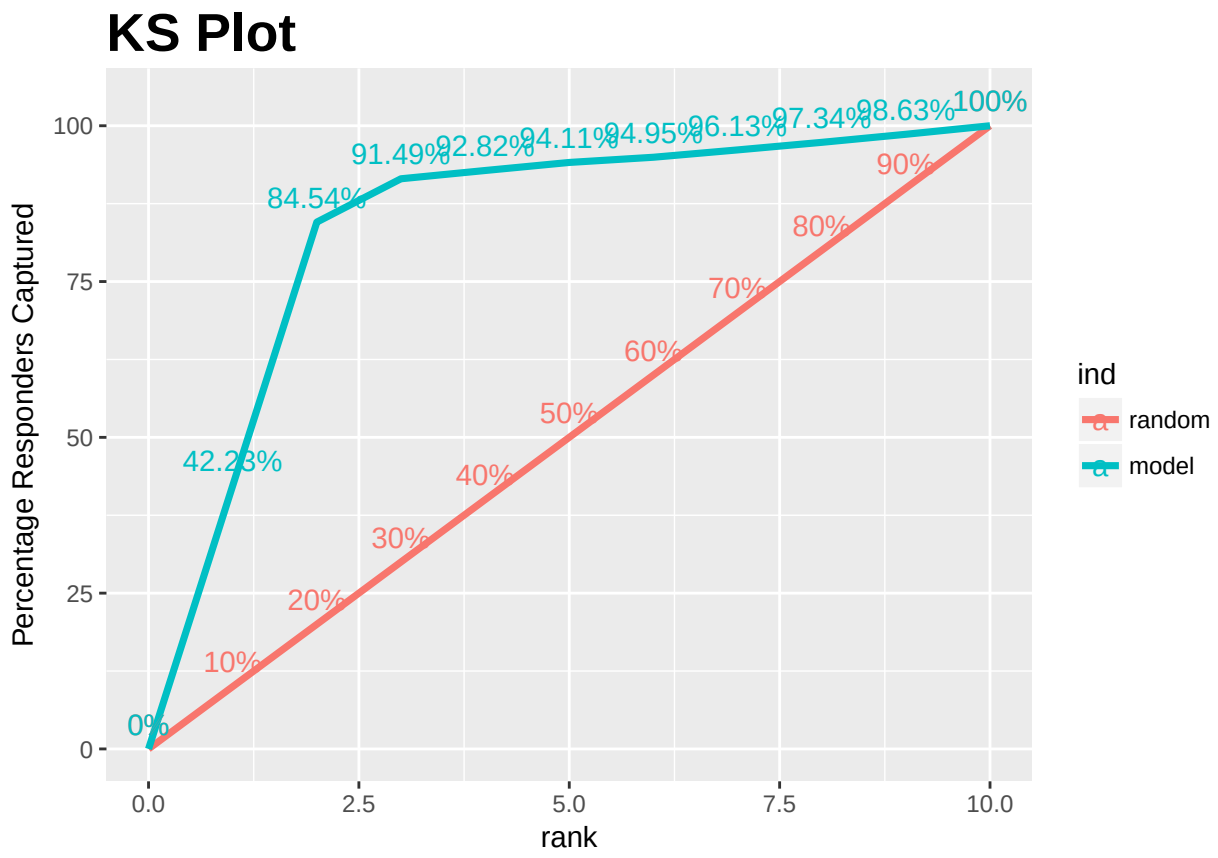
```
preaccuracy<-sum(diag(table(hrtest$left,rpartmodelpre_test)))/nrow(hrtest)
preaccuracy
```

```
## [1] 0.9744
```

```
#So the accuracy of our Pre-Prune Decision Tree is 97.44%, as we have calculated from
# different method above.

#2. ROC
predicthr <- predict(object=rpartmodelpre,newdata=hrtrain,type="class")
plotROC(actuals = hrtrain$left,predictedScores = as.numeric(predicthr))
```




```
#since the area under curve is 94.88, we can say  the discrimination  ability
# of our model is good.

#3. Kolmogrov Smirnov Chart
#measure the performance of classification models.

ks_plot(actuals = hrtrain$left,predictedScores = as.numeric(predicthr))
```




```
# as the measure between left and not-left is large, we can say that
# performance of the model is good.
ks_stat(actuals = hrtrain$left,predictedScores = as.numeric(predicthr))
```

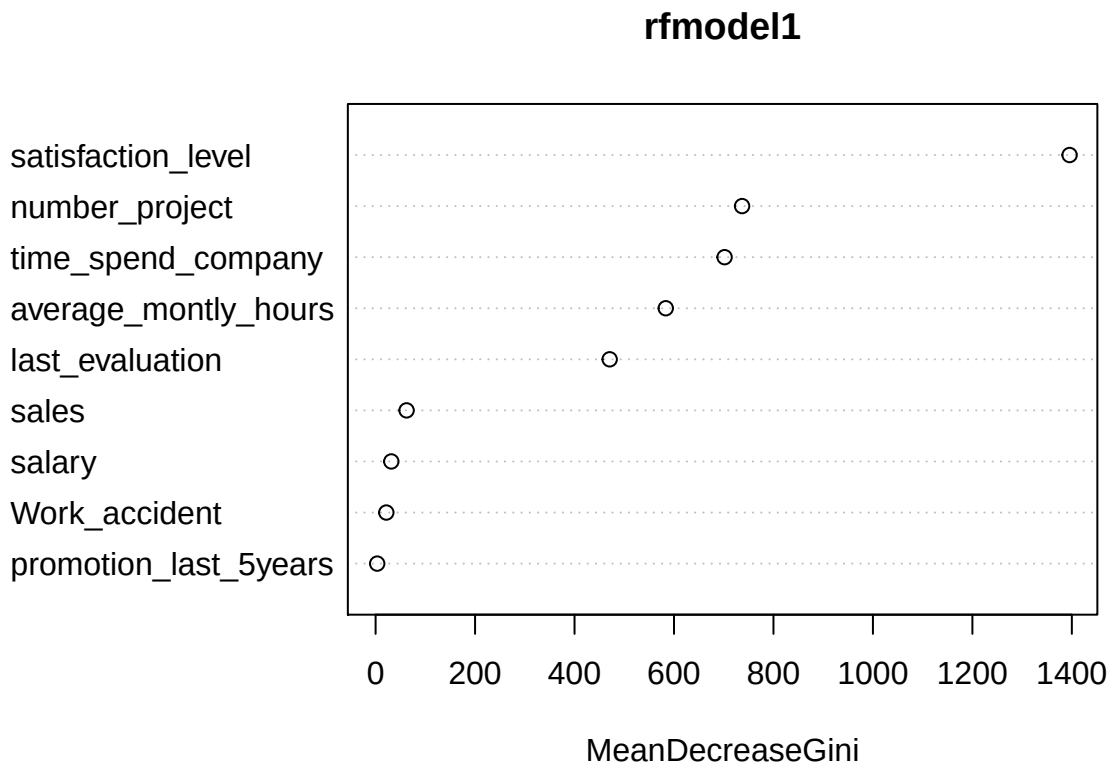
```
## [1] 0.8426
```

```
#As the kolmogrov Smirnov statistic is 0.8426 which is good, we can say
# that the model is efficient in capturing  the responders(1).

################################################################################
############################Random Forest#######################################

library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:rattle':
##
##     importance
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
#Random Forest Model 1: default settings
set.seed(1234)

rfmodel1=randomForest(left~.,data=hrtrain)

  #importance of each variable
#the more the decrease in GIni, the more the importance
#of the independednt variable in the prediction the dependent varioable
varImpPlot(rfmodel1)
```




```
rf_predict=predict(object=rfmodel1,newdata=hrtest,type="response")

table(hrtest$left,rf_predict)
```

```
##    rf_predict
##        0    1
##   0 2806    6
##   1   27  911
```

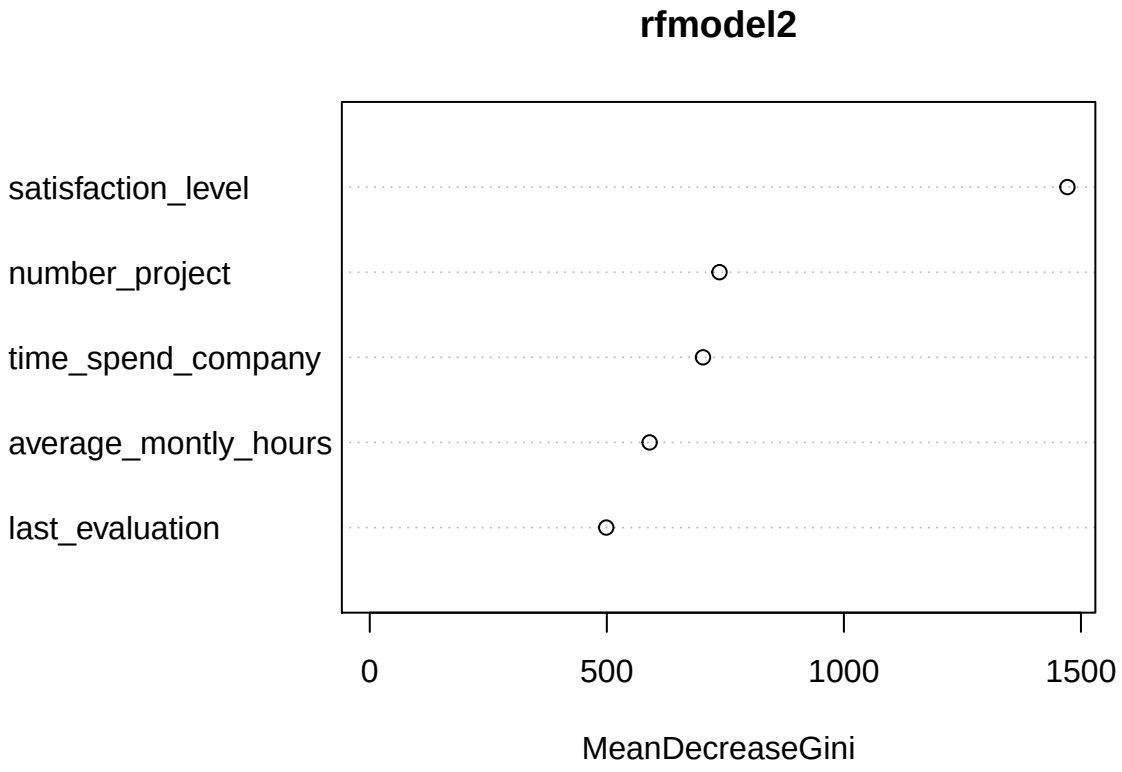
```
rfbaseaccuracy <-sum(diag(table(hrtest$left,rf_predict)))/nrow(hrtest)
cat("Accuracy of the model is" )
```

```
## Accuracy of the model is
```

```
rfbaseaccuracy
```

```
## [1] 0.9912
```

```
#remove some insignificant variables and build random forest model again

set.seed(1000)

rfmodel2=randomForest(formula=left~satisfaction_level+number_project
                      +time_spend_company+average_montly_hours+last_evaluation
                      ,data=hrtrain)
varImpPlot(rfmodel2)
```




```
rfpredict2=predict(object=rfmodel2,newdata=hrtest,type="response")

table(hrtest$left,rfpredict2)
```

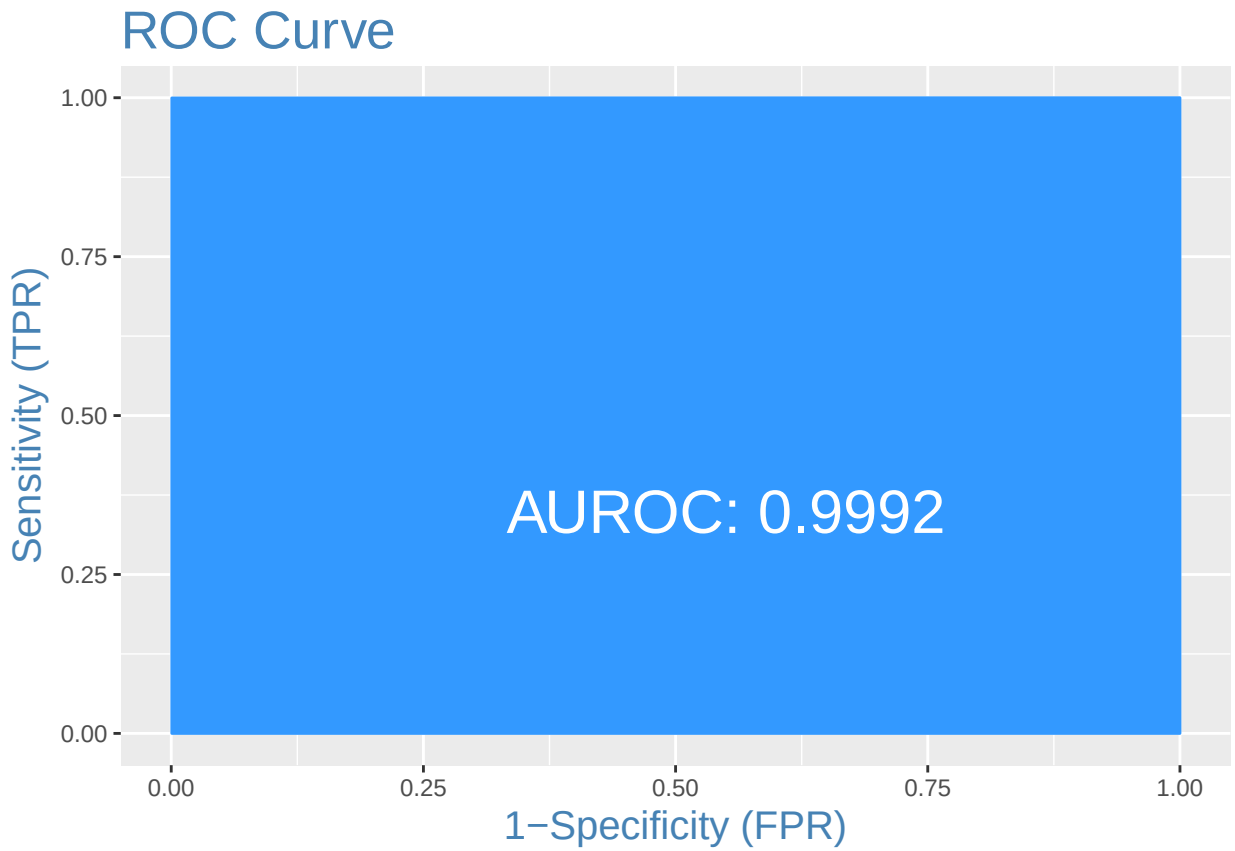
```
##    rfpredict2
##        0    1
##   0 2805    7
##   1   27  911
```

```
sum(diag(table(hrtest$left,rfpredict2)))/nrow(hrtest) #99.09%
```

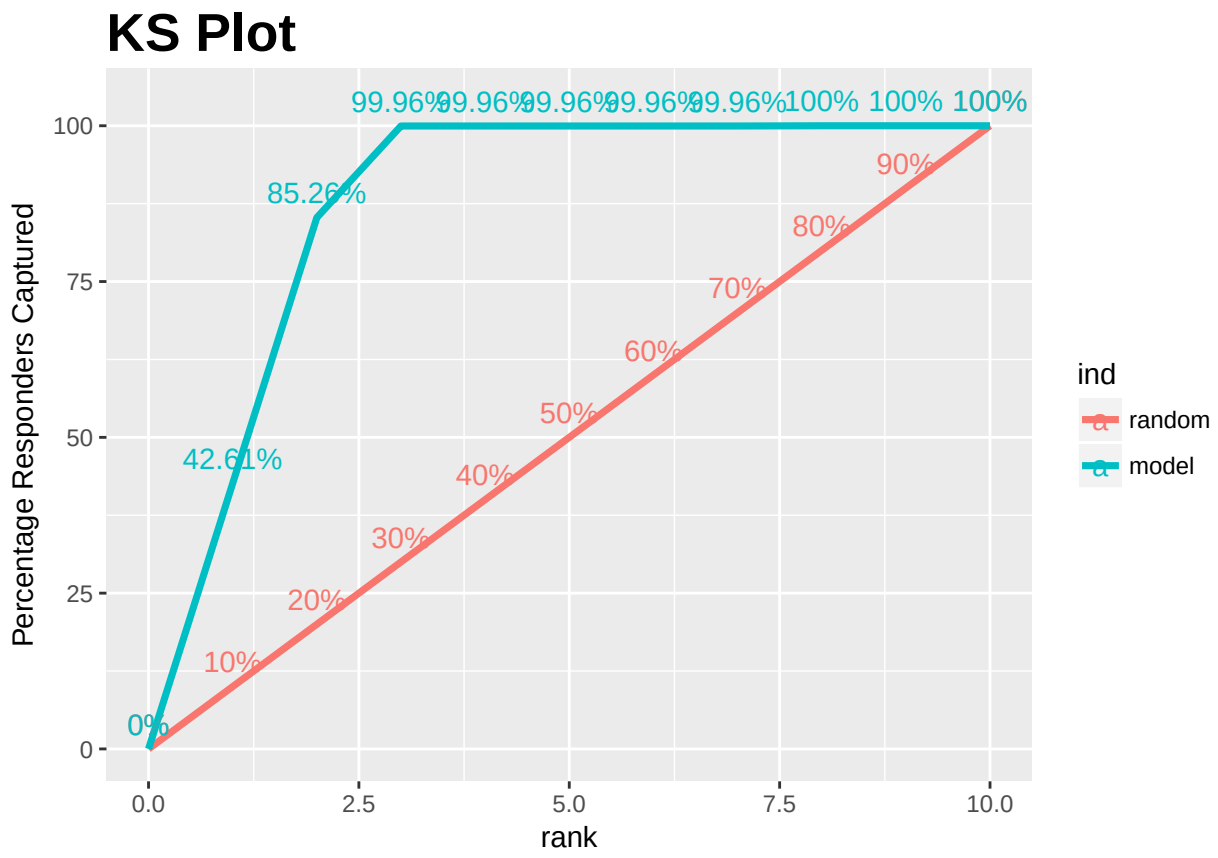
```
## [1] 0.9909333
```

```
#2. ROC
predicttrainrf <- predict(object=rfmodel2,newdata=hrtrain,type="class")
```

```
plotROC(actuals = hrtrain$left,predictedScores = as.numeric(predicttrainrf))
```




```
#since the area under curve is 99.87, we can say  the discrimination  ability
# of our model is good.

#3. Kolmogrov Smirnov Chart
#measure the performance of classification models.

ks_plot(actuals = hrtrain$left,predictedScores = as.numeric(predicttrainrf))
```




```
# as the measure between left and not-left is large, we can say that
# performance of the model is good.
ks_stat(actuals = hrtrain$left,predictedScores = as.numeric(predicttrainrf))
```

```
## [1] 0.9134
```

```
#As the kolmogrov Smirnov statistic is 0.9134 which is good, we can say
# that the model is efficient in capturing  the responders(1).

#so the best algo which provides the correctd prediction is from random forest which is
#rfmodel2 with few predictors.


################################################################################
############UnRegularized Logistic Regression with optimization################
################################################################################

#create dummy variable for categorical variables

hrtrain1 <- hrtrain
hrtest1 <- hrtest
hrtrain1$left <- ifelse(hrtrain1$left==1,0,1)
hrtest1$left <- ifelse(hrtest1$left==1,0,1)
# table(hrtrain$left)
# table(hrtrain1$left)
# table(hrtest1$left)
for(level in unique(hrtrain1$sales)){
```

```
  hrtrain1[paste("dummy", level, sep = "_")] <- ifelse(hrtrain1$sales == level, 1, 0)
}

for(level in unique(hrtrain1$salary)){
  hrtrain1[paste("dummy", level, sep = "_")] <- ifelse(hrtrain1$salary == level, 1, 0)
}
#View(hrtrain1)

for(level in unique(hrtest1$sales)){
  hrtest1[paste("dummy", level, sep = "_")] <- ifelse(hrtest1$sales == level, 1, 0)
}

for(level in unique(hrtest1$salary)){
  hrtest1[paste("dummy", level, sep = "_")] <- ifelse(hrtest1$salary == level, 1, 0)
}
#View(hrtest1)

hrtrain1_dummy <- hrtrain1[,-c(9,10)]
hrtest1_dummy <- hrtest1[,-c(9,10)]

#create sigmoid function
sigmoid <- function(z){
  1/(1+exp(-z))
}

#check whether our sigmoid is correct or not
sigmoid(0)#correct for 0
```

```
## [1] 0.5
```

```
sigmoid(10000)#correct for large positive value
```

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

```
sigmoid(-10000)#correct for large negative value
```

```
## [1] 0
```

```
#cost function and gradient
# h(x) = g(z)---sigmoid function
# cost function j(beta) = (sum(i,-y(i)*log(h_beta(x(i)))-(1-y(i))*log(1-h_beta(x(i))))*1/nrow(x(i)) --
# gradient dj(beta)/d(beta(k)) = (sum(i,h_beta(x(i))-y(i)x(k)))*1/nrow(x(i)) --> k is no of variables

#now we will create X vectors
# In X vector we will add 1 in each row for intercept
# hrtest1$left<- as.integer(hrtest1$left)
# hrtrain1$left<-as.numeric(hrtrain1$left)
# table(hrtrain1$left)
# table(hrtest1$left)
y <- (hrtrain1$left)
ytest <- (hrtest1$left)
class(hrtest1$left)
```

```
## [1] "numeric"
```

```
X <- cbind(1,hrtrain1[,-c(7,9,10)])
head(X)
```

```
##       1 satisfaction_level last_evaluation number_project
## 10318 1               0.22            0.70              4
## 7382  1               0.71            0.56              3
## 5176  1               0.62            0.92              5
## 14922 1               0.10            0.86              6
## 10426 1               0.63            0.54              4
## 161   1               0.10            0.95              7
##       average_montly_hours time_spend_company Work_accident
## 10318                  225                  4             0
## 7382                   177                  4             0
## 5176                   149                  3             0
## 14922                  283                  4             0
## 10426                  147                  4             0
## 161                    301                  4             0
##       promotion_last_5years dummy_sales dummy_support dummy_IT
## 10318                     0           1             0        0
## 7382                      0           0             1        0
## 5176                      0           0             0        1
## 14922                     0           0             1        0
## 10426                     0           0             0        0
## 161                       0           0             0        0
##       dummy_technical dummy_marketing dummy_management dummy_RandD
## 10318               0               0                0           0
## 7382                0               0                0           0
## 5176                0               0                0           0
## 14922               0               0                0           0
## 10426               1               0                0           0
## 161                 0               1                0           0
##       dummy_product_mng dummy_accounting dummy_hr dummy_medium dummy_high
## 10318                 0                0        0            1          0
## 7382                  0                0        0            0          1
## 5176                  0                0        0            0          0
## 14922                 0                0        0            0          0
## 10426                 0                0        0            0          0
## 161                   0                0        0            0          0
##       dummy_low
## 10318         0
## 7382          0
## 5176          1
## 14922         1
## 10426         1
## 161           1
```

```
class(X)
```

```
## [1] "data.frame"
```

```
Xtest <- cbind(1,hrtest1[,-c(7,9,10)])
head(Xtest)
```

```
##    1 satisfaction_level last_evaluation number_project
## 1  1               0.38            0.53              2
```

```
## 4  1               0.72            0.87             5
## 5  1               0.37            0.52             2
## 6  1               0.41            0.50             2
## 10 1               0.42            0.53             2
## 13 1               0.84            0.92             4
##    average_montly_hours time_spend_company Work_accident
## 1                   157                  3             0
## 4                   223                  5             0
## 5                   159                  3             0
## 6                   153                  3             0
## 10                  142                  3             0
## 13                  234                  5             0
##    promotion_last_5years dummy_sales dummy_technical dummy_product_mng
## 1                      0           1               0                 0
## 4                      0           1               0                 0
## 5                      0           1               0                 0
## 6                      0           1               0                 0
## 10                     0           1               0                 0
## 13                     0           1               0                 0
##    dummy_IT dummy_marketing dummy_hr dummy_support dummy_management
## 1         0               0        0             0                 0
## 4         0               0        0             0                 0
## 5         0               0        0             0                 0
## 6         0               0        0             0                 0
## 10        0               0        0             0                 0
## 13        0               0        0             0                 0
##    dummy_accounting dummy_RandD dummy_low dummy_medium dummy_high
## 1                 0           0         1            0          0
## 4                 0           0         1            0          0
## 5                 0           0         1            0          0
## 6                 0           0         1            0          0
## 10                0           0         1            0          0
## 13                0           0         1            0          0
```

```
#Initialize fitting parameters
initial_beta = matrix(rep(0,ncol(X)))
initial_beta
```

```
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
```

```
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0
## [21,]    0
```

```
#create cost function
costj <- function(beta, X, y){
  cgradient <- list()
  h_beta <- sigmoid(as.matrix(X)%*%beta)
  cost = 1/nrow(X)*sum(-y*log(h_beta)-(1-y)*log(1-h_beta))
  return(cost)
}

#check cost of initial parameters
costj(initial_beta,X,y)
```

```
## [1] 0.6931472
```

```
#create gradient function

gradientd <- function(beta, X, y){
  h_beta <- sigmoid(as.matrix(X)%*%beta)
  gradient <- 1/nrow(X)*(sigmoid(t(beta)%*%t(as.matrix(X)))-t(y))%*%as.matrix(X)
  return(gradient)
}

#check gradient for intial parameters
gradientd(initial_beta,X,y)
```

```
##               1 satisfaction_level last_evaluation number_project
## [1,] -0.2659347         -0.2050453       -0.189924     -0.9992444
##      average_montly_hours time_spend_company Work_accident
## [1,]            -52.02925          -0.846253   -0.06151658
##      promotion_last_5years dummy_sales dummy_support    dummy_IT
## [1,]          -0.008489644 -0.07125078   -0.03862566 -0.02297982
##      dummy_technical dummy_marketing dummy_management dummy_RandD
## [1,]     -0.04533736     -0.01586808      -0.01457907 -0.01813495
##      dummy_product_mng dummy_accounting    dummy_hr dummy_medium
## [1,]       -0.01649035       -0.0124011 -0.01026758    -0.128367
##       dummy_high  dummy_low
## [1,] -0.03560316 -0.1019646
```

```
#now we will optimize the cost to the get the optimum parameters

opt_para <- optim(par = initial_beta,X=X,y=y,fn=costj,gr=gradientd)
names(opt_para)
```

```
## [1] "par"         "value"       "counts"      "convergence" "message"
```

```
cat("optimized parameters are: ")
```

```
## optimized parameters are:
```

```
round(opt_para$par,3)
```

```
##         [,1]
##  [1,]  0.429
```

```
##  [2,]  2.617
##  [3,] -1.279
##  [4,]  0.405
##  [5,] -0.002
##  [6,] -0.228
##  [7,]  0.413
##  [8,] -0.529
##  [9,]  0.073
## [10,]  0.148
## [11,] -0.165
## [12,]  0.093
## [13,]  0.174
## [14,]  0.108
## [15,]  0.248
## [16,] -0.923
## [17,] -0.566
## [18,] -0.478
## [19,] -0.083
## [20,]  0.342
## [21,] -0.501
```

```
cat("cost at optimized parameters is: ")
```

```
## cost at optimized parameters is:
```

```
round(opt_para$value,3)
```

```
## [1] 0.462
```

```
hrtrain1$fitted_prob <- sigmoid(as.matrix(X)%*%opt_para$par)
hrtrain1$fitted_label <- ifelse(hrtrain1$fitted_prob>=0.5,1,0)

accuracy = sum(hrtrain1$left==hrtrain1$fitted_label)/nrow(hrtrain1)
accuracy
```

```
## [1] 0.7446884
```

```
tp = sum((hrtrain1$left==1)&(hrtrain1$fitted_label==1))
tp
```

```
## [1] 8086
```

```
fp = sum((hrtrain1$left==0)&(hrtrain1$fitted_label==1))
fp
```

```
## [1] 2342
```

```
prec = tp/(tp+fp)
cat("The precision is: ")
```

```
## The precision is:
```

```
prec
```

```
## [1] 0.7754124
```

```
#tp = sum((hrtrain1$left==1)&(hrtrain1$fitted_label==1))
fn = sum((hrtrain1$left==1)&(hrtrain1$fitted_label==0))
rec = tp/(tp+fn)
cat("The recall is: ")
```

```
## The recall is:
```

```
rec
```

```
## [1] 0.9384865
```

```
F1 = 2*(prec*rec)/(prec+rec)
cat("The F1 score is: ")
```

```
## The F1 score is:
```

```
F1
```

```
## [1] 0.8491913
```

```
################################################################################
############Regularized Logistic Regression with optimization##################
################################################################################

#Initialize fitting parameters
initial_beta = matrix(rep(0,ncol(X)))
initial_beta
```

```
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0
## [21,]    0
```

```
#create cost function for regularization
costr <- function(beta, X, y,lambda){
  cgradient <- list()
  h_beta <- sigmoid(as.matrix(X)%*%beta)
  cost = 1/nrow(X)*sum(-y*log(h_beta)-(1-y)*log(1-h_beta))+lambda/(2*nrow(X))*sum(beta^2)
  return(cost)
}

#check cost of initial parameters
costr(initial_beta,X,y,lambda = 10)
```

```
## [1] 0.6931472
```

```r
#create gradient function

gradientr <- function(beta, X, y,lambda){
  h_beta <- sigmoid(as.matrix(X)%*%beta)
  gradient <- 1/nrow(X)*(sigmoid(t(beta)%*%t(as.matrix(X)))-t(y))%*%as.matrix(X)+(lambda/(nrow(X)))*c(0
  return(gradient)
}

#check gradient for intial parameters
gradientr(initial_beta,X,y,lambda = 10)
```

```
##               1 satisfaction_level last_evaluation number_project
## [1,] -0.2659347         -0.2050453       -0.189924     -0.9992444
##      average_montly_hours time_spend_company Work_accident
## [1,]            -52.02925          -0.846253   -0.06151658
##      promotion_last_5years dummy_sales dummy_support    dummy_IT
## [1,]          -0.008489644 -0.07125078   -0.03862566 -0.02297982
##      dummy_technical dummy_marketing dummy_management dummy_RandD
## [1,]     -0.04533736     -0.01586808      -0.01457907 -0.01813495
##      dummy_product_mng dummy_accounting    dummy_hr dummy_medium
## [1,]       -0.01649035       -0.0124011 -0.01026758    -0.128367
##       dummy_high  dummy_low
## [1,] -0.03560316 -0.1019646
```

```r
#now we will optimize the cost to the get the optimum parameters

opt_rpara <- optim(par = initial_beta,X=X,y=y,lambda=1,fn=costr,gr=gradientr,method = "BFGS")
names(opt_rpara)
```

```
## [1] "par"         "value"       "counts"      "convergence" "message"
```

```r
cat("optimized parameters are: ")
```

```
## optimized parameters are:
```

```r
round(opt_rpara$par,3)
```

```
##         [,1]
##  [1,]  0.248
##  [2,]  4.187
##  [3,] -0.770
##  [4,]  0.302
##  [5,] -0.004
##  [6,] -0.248
##  [7,]  1.444
##  [8,]  1.082
##  [9,] -0.027
## [10,] -0.091
## [11,]  0.059
## [12,] -0.140
## [13,]  0.028
## [14,]  0.254
## [15,]  0.456
## [16,]  0.028
## [17,] -0.009
## [18,] -0.346
```

```
## [19,] -0.221
## [20,]  1.169
## [21,] -0.735
```

```
cat("cost at optimized parameters is: ")
```

```
## cost at optimized parameters is:
```

```
round(opt_rpara$value,3)
```

```
## [1] 0.426
```

```
# predict <- sigmoid(as.matrix(Xtest)%*%opt_para$par)
# predicttest <- ifelse(predict>=0.5,1,0)
# View(predict)
# joinpredictactual <- cbind(ytest,predicttest)
# View(joinpredictactual)

hrtrain1$fitted_prob <- sigmoid(as.matrix(X)%*%opt_rpara$par)
hrtrain1$fitted_label <- ifelse(hrtrain1$fitted_prob>=0.5,1,0)
head(hrtrain1)
```

```
##       satisfaction_level last_evaluation number_project
## 10318               0.22            0.70              4
## 7382                0.71            0.56              3
## 5176                0.62            0.92              5
## 14922               0.10            0.86              6
## 10426               0.63            0.54              4
## 161                 0.10            0.95              7
##       average_montly_hours time_spend_company Work_accident left
## 10318                  225                  4             0    1
## 7382                   177                  4             0    1
## 5176                   149                  3             0    1
## 14922                  283                  4             0    0
## 10426                  147                  4             0    1
## 161                    301                  4             0    0
##       promotion_last_5years     sales salary dummy_sales dummy_support
## 10318                     0     sales medium           1             0
## 7382                      0   support   high           0             1
## 5176                      0        IT    low           0             0
## 14922                     0   support    low           0             1
## 10426                     0 technical    low           0             0
## 161                       0 marketing    low           0             0
##       dummy_IT dummy_technical dummy_marketing dummy_management
## 10318        0               0               0                0
## 7382         0               0               0                0
## 5176         1               0               0                0
## 14922        0               0               0                0
## 10426        0               1               0                0
## 161          0               0               1                0
##       dummy_RandD dummy_product_mng dummy_accounting dummy_hr dummy_medium
## 10318           0                 0                0        0            1
## 7382            0                 0                0        0            0
## 5176            0                 0                0        0            0
## 14922           0                 0                0        0            0
## 10426           0                 0                0        0            0
```

```
## 161             0                 0                0        0            0
##       dummy_high dummy_low fitted_prob fitted_label
## 10318          0         0   0.4291378            0
## 7382           1         0   0.9563341            1
## 5176           0         1   0.8375623            1
## 14922          0         1   0.2474131            0
## 10426          0         1   0.7743076            1
## 161            0         1   0.3034366            0
```

```
table(hrtrain$left)
```

```
##
##    0    1
## 8616 2633
```

```
table(hrtrain1$left,hrtrain1$fitted_label)
```

```
##
##        0    1
##   0  881 1752
##   1  587 8029
```

```
accuracy = sum(hrtrain1$left==hrtrain1$fitted_label)/nrow(hrtrain1)
accuracy
```

```
## [1] 0.7920704
```

```
# though the accuracy has been improved but it does not seem good fit classification.

tp = sum((hrtrain1$left==1)&(hrtrain1$fitted_label==1))
tp
```

```
## [1] 8029
```

```
fp = sum((hrtrain1$left==0)&(hrtrain1$fitted_label==1))
fp
```

```
## [1] 1752
```

```
prec = tp/(tp+fp)
cat("The precision is: ")
```

```
## The precision is:
```

```
prec
```

```
## [1] 0.8208772
```

```
#tp = sum((hrtrain1$left==1)&(hrtrain1$fitted_label==1))
fn = sum((hrtrain1$left==1)&(hrtrain1$fitted_label==0))
rec = tp/(tp+fn)
cat("The recall is: ")
```

```
## The recall is:
```

```
rec
```

```
## [1] 0.9318709
```

```
F1 = 2*(prec*rec)/(prec+rec)
cat("The F1 score is: ")
```

```
## The F1 score is:
```

```
F1
```

```
## [1] 0.8728597
```

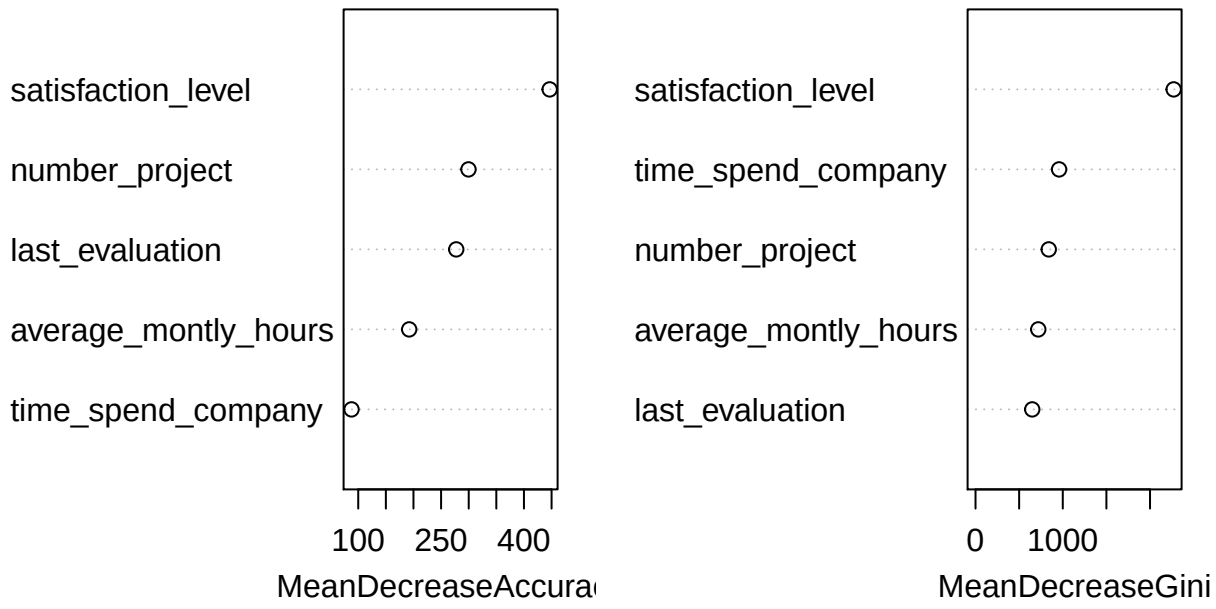
```
#F1 score (also F-score or F-measure) is a measure of a test's accuracy.
# It considers both the precision p and the recall r of the test to compute
# the score: p is the number of correct positive results divided by the number
# of all positive results returned by the classifier, and r is the number of
# correct positive results divided by the number of all relevant samples
# (all samples that should have been identified as positive). The F1 score
# is the harmonic average of the precision and recall, where an F1 score
# reaches its best value at 1 (perfect precision and recall) and worst at 0.

# So regularization in this case does not give the best model.
##############################################################################
############################Best Model Description############################
# #so the best algo which provides the correctd prediction is from random forest which is
#rfmodel2 with few predictors.
#So we will create final model using Random forest with few predictors
# as suggested by the fit and validation.

set.seed(1000)
hrdata$left<-as.factor(hrdata$left)
class(hrdata$sales)
```

```
## [1] "character"
```

```
hrdata$sales<-as.factor(hrdata$sales)
hrdata$salary<-as.factor(hrdata$salary)
Best_Model=randomForest(formula=left~satisfaction_level+number_project
                    +time_spend_company+average_montly_hours+last_evaluation
                    ,data=hrdata,importance = TRUE,mtry=3,do.trace=100)
```

```
## ntree      OOB      1      2
##   100:   0.83%  0.32%  2.46%
##   200:   0.77%  0.27%  2.38%
##   300:   0.78%  0.28%  2.38%
##   400:   0.78%  0.28%  2.38%
##   500:   0.79%  0.29%  2.38%
```

```
varImpPlot(Best_Model)
```




```
print(Best_Model)
```

```
##
## Call:
##  randomForest(formula = left ~ satisfaction_level + number_project +      time_spend_company + avera
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0.79%
## Confusion matrix:
##       0    1 class.error
## 0 11395   33 0.002887644
## 1    85 3486 0.023802856
```

```
Best_Model$confusion
```

```
##       0    1 class.error
## 0 11395   33 0.002887644
## 1    85 3486 0.023802856
```

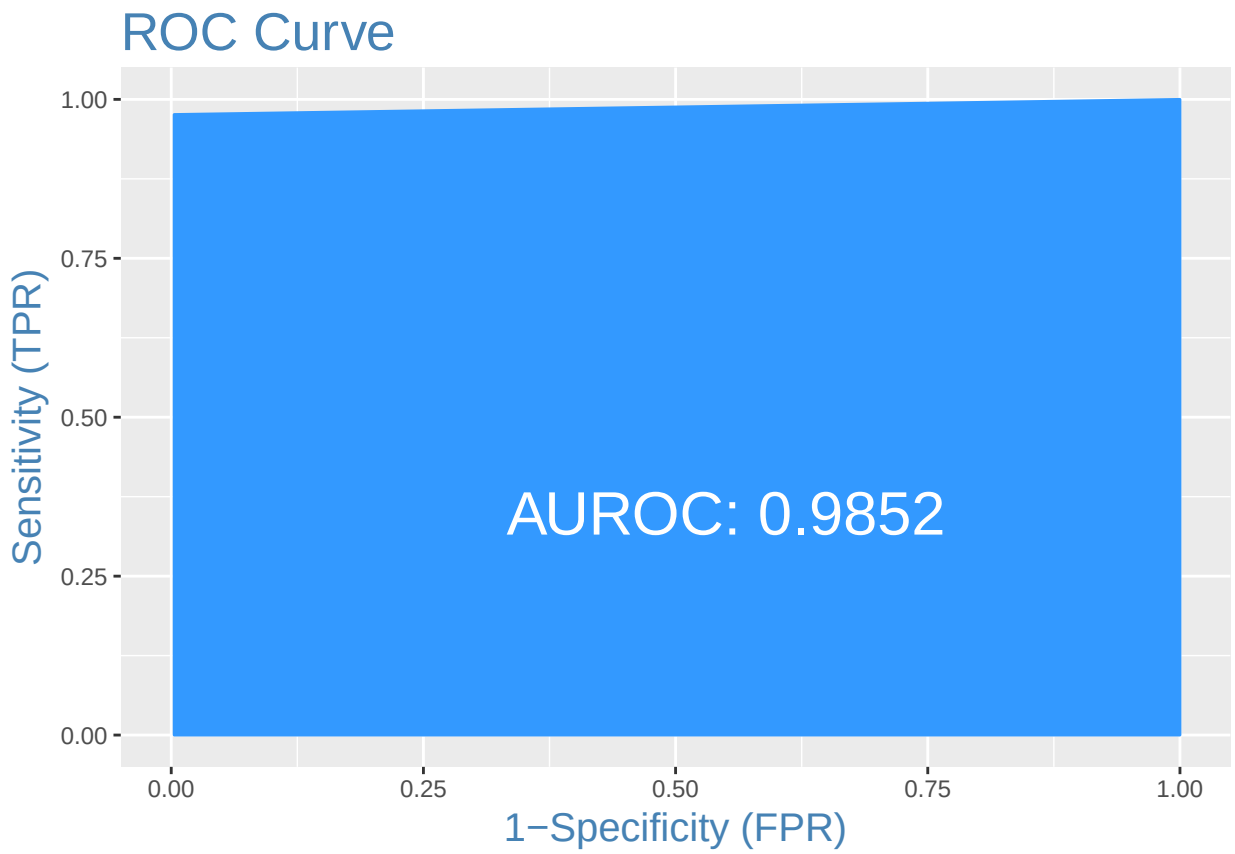
```
# Best_Model$predicted
Best_fitted <- Best_Model$predicted

#Best_fitted=predict(object=Best_Model,newdata=hrdata,type="response")

table(hrdata$left,Best_fitted)
```

```
##    Best_fitted
##          0     1
##   0 11395    33
##   1    85  3486
```

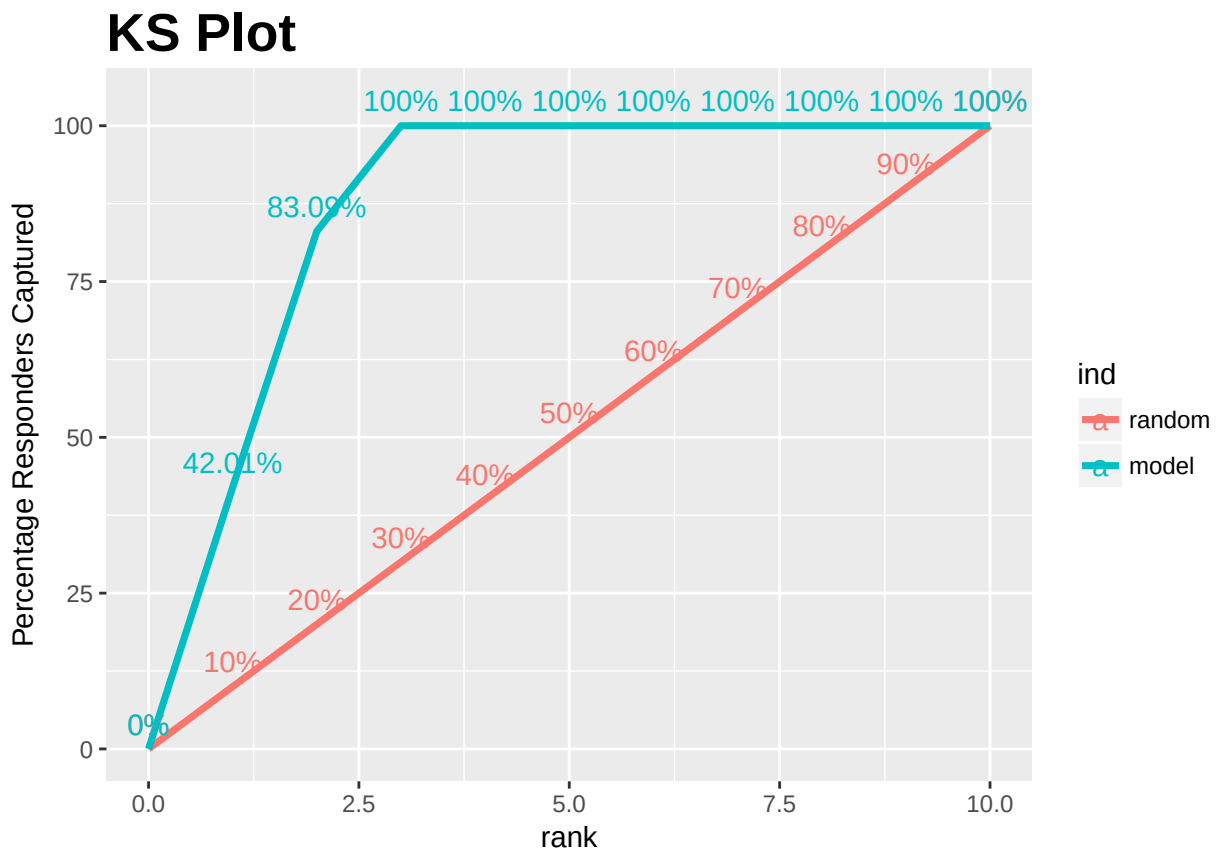
```
sum(diag(table(hrdata$left,Best_fitted)))/nrow(hrdata) #99.25%
```

```
## [1] 0.9921328
```

```
#2. ROC

plotROC(actuals = hrdata$left,predictedScores = as.numeric(Best_fitted))
```




```
#since the area under curve is 98.55, we can say  the discrimination  ability
# of our model is good.

#3. Kolmogrov Smirnov Chart
#measure the performance of classification models.

ks_plot(actuals = hrdata$left,predictedScores = as.numeric(Best_fitted))
```




```
# as the measure between left and not-left is large, we can say that
# performance of the Best model is good.
ks_stat(actuals = hrdata$left,predictedScores = as.numeric(Best_fitted))
```

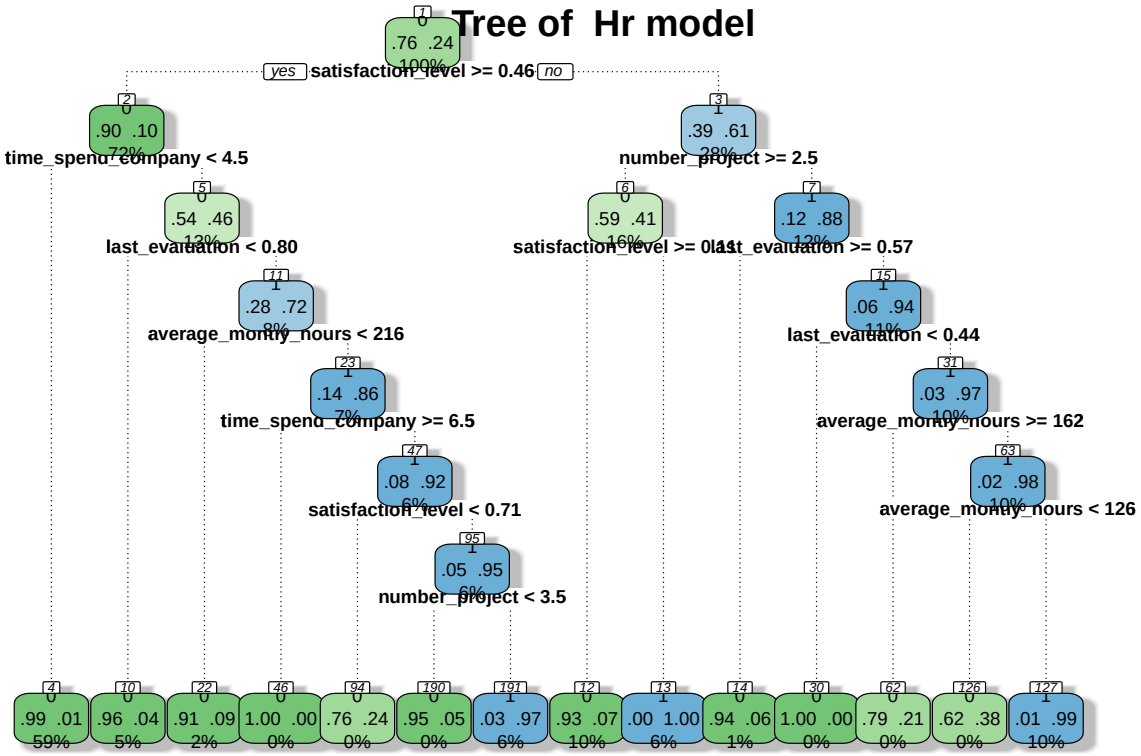
```
## [1] 0.9187
```

```
#As the kolmogrov Smirnov statistic is 0.91.8 which is good, we can say
# that the Best model is efficient in capturing  the responders(1).

Best_Model$importance
```

```
##                                0         1 MeanDecreaseAccuracy
## satisfaction_level   0.067029231 0.6432054            0.2040259
## number_project       0.021960111 0.4855721            0.1321926
## time_spend_company   0.012237787 0.3166658            0.0845776
## average_montly_hours 0.021578796 0.4310924            0.1189480
## last_evaluation      0.004883856 0.4685621            0.1151471
##                      MeanDecreaseGini
## satisfaction_level           2270.8914
## number_project                839.2562
## time_spend_company            957.8192
## average_montly_hours          719.0021
## last_evaluation               649.8613
```

```
#As we have also best model from pre-prune decision tree and also random
# forest creates multiple decision tree so we have considered the plot
# of one of decision tree for explaning the business.
dctree4plot=rpart(formula=left~satisfaction_level+number_project
```

```
                    +time_spend_company+average_montly_hours+last_evaluation
                    ,data=hrdata,method="class",
                    control = rpart.control(cp = 0.0,maxdepth = 9,minsplit = 100,minbucket = 20))
#summary(rpartmodelpre)
windows()
fancyRpartPlot(model=dctree4plot,main="Tree of  Hr model",cex=0.6)
```




```
# So the conclusion is:
#1. The critical attributes on which the business has to focus on are:
#a.satisfaction_level
#b.number_project
#c.time_spend_company
#d.average_montly_hours
#e.last_evaluation
#2.As the satisfaction level increases mostly people tend to leave the
# Organization.
#3. More people tend to leave if they have less than 3 projects, so
# business has to focus on employee and the project they should be alligned to.
#4.People who spends daily on an average 6.5 hours in the office, they
# do not tend to leave the organization, so policy need to brought in to
# strictly follow the business hours.
#5.Employees who spend less than 126 hours on monthly basis they are the
# ones who are leaving the organization in bigger number.
#6.Employess who has last_evaluation less than .44 they are not leaving
# the organization, it mean employees need to evaluated frequently so
# they can understand their goals and do not leave the organization.
```

```
#7. Other attributes are not much impacting the attrition rate.
#8.This does not sums up the conclusion of HR Analytics Model, as the time
# progress we have to evolve this model and improve the model and remove
#the nuances.
```