Udemy_Project

```
library(rmarkdown)
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(Hmisc)
```

```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula
```

```
## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:plyr':
##
##     is.discrete, summarize

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

#reading data

```
df <- read_excel('Data_Extract_From_Enterprise_Surveys.xlsx')
```

#changing column names

```
colnames(df)[1] <- "country"
colnames(df)[2] <- "code"
colnames(df)[3] <- "serie"
colnames(df)[4] <- "seirsCode"
```

#turing some columns into rows

```
library(reshape2)
df <- melt(df,id.vars=c('country','code','serie'), measure.vars =c(colnames(df)[5:length(colnames(df))]
```

#changing column name

```
colnames(df)[4] <- "year"
```

#eliminating data where value column is nan

```
df <- subset(df, subset = is.na(df$value) == F)
```

#using a function to give nan values to ".."

```
fillP <- function(x){
  if (x=="..") {
    return(NA)
  }
  else {
    return(as.numeric(x))
  }
  }
df$value <- sapply(df$value, fillP)
```

#pivoting the data

```
df_p <- dcast(df, country + code + year ~ serie, value.var="value", fun.aggregate=sum)
```

#creating a new dataframe by taking 'Percent of firms that spend on R&D' as reference

```
df_m <- subset(df_p, subset = is.na(df_p['Percent of firms that spend on R&D'])==FALSE)
```

#number of unique countries

```
length(unique(df_m$country))
```

```
## [1] 128
```

#elimination those countries having more than one oberservation

```
for (i in unique(df_m$country)){
  d <- subset(df_m, subset = df_m$country == i)
  if (nrow(d) > 1){
    d <- tail(d, 1)
    df_m <- subset(df_m, subset = df_m$country != i)
    df_m <- rbind(df_m, d)
  }
}
```

#dropping a column bcz of missing values

```
df_m['Percent of firms using e-mail to interact with clients/suppliers'] <- NULL
```

#variable list for the model based on corr table

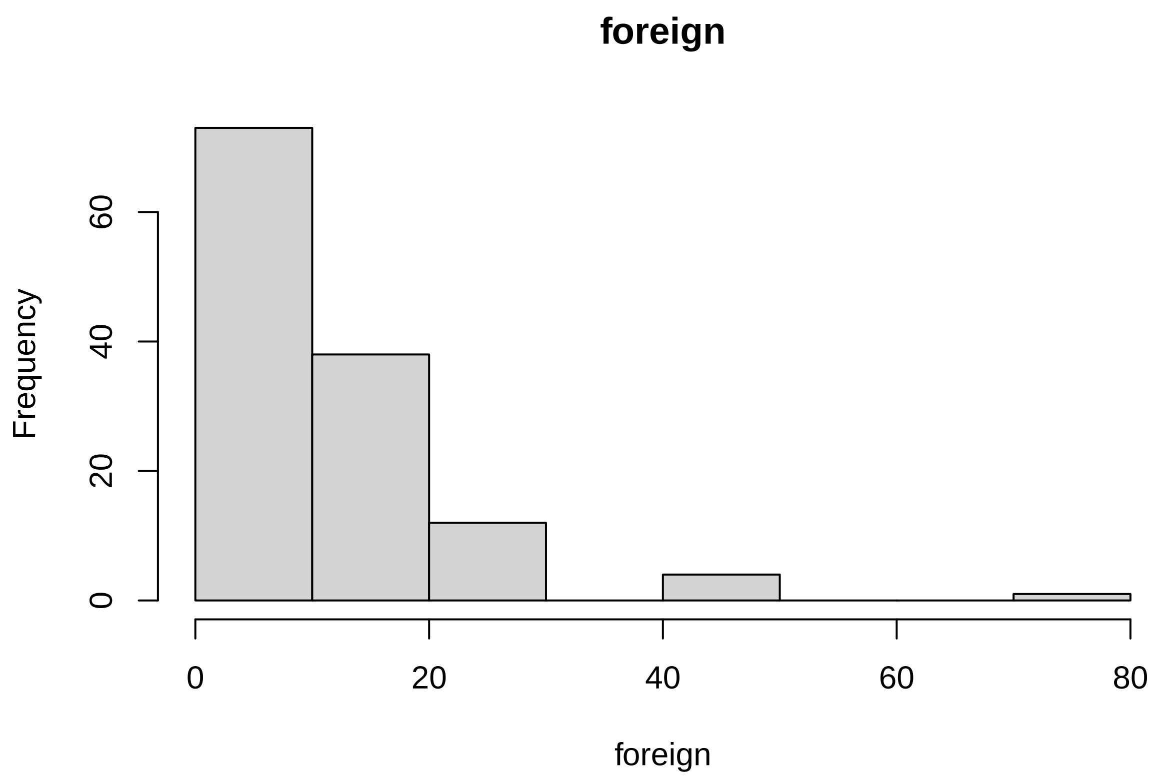
```
model.list <- c("country","code",
                "Percent of firms with at least 10% of foreign ownership",
                "Percent of firms with an internationally-recognized quality certification",
                "Percent of firms with female participation in ownership",
                "Percent of firms having their own Web site",
                "Percent of firms offering formal training",
                "Proportion of investment financed by banks (%)",
                "Proportion of private domestic ownership in a firm (%)",
                "Percent of firms expected to give gifts in meetings with tax officials",
                "Percent of firms competing against unregistered or informal firms",
                "Percent of firms that spend on R&D")
```

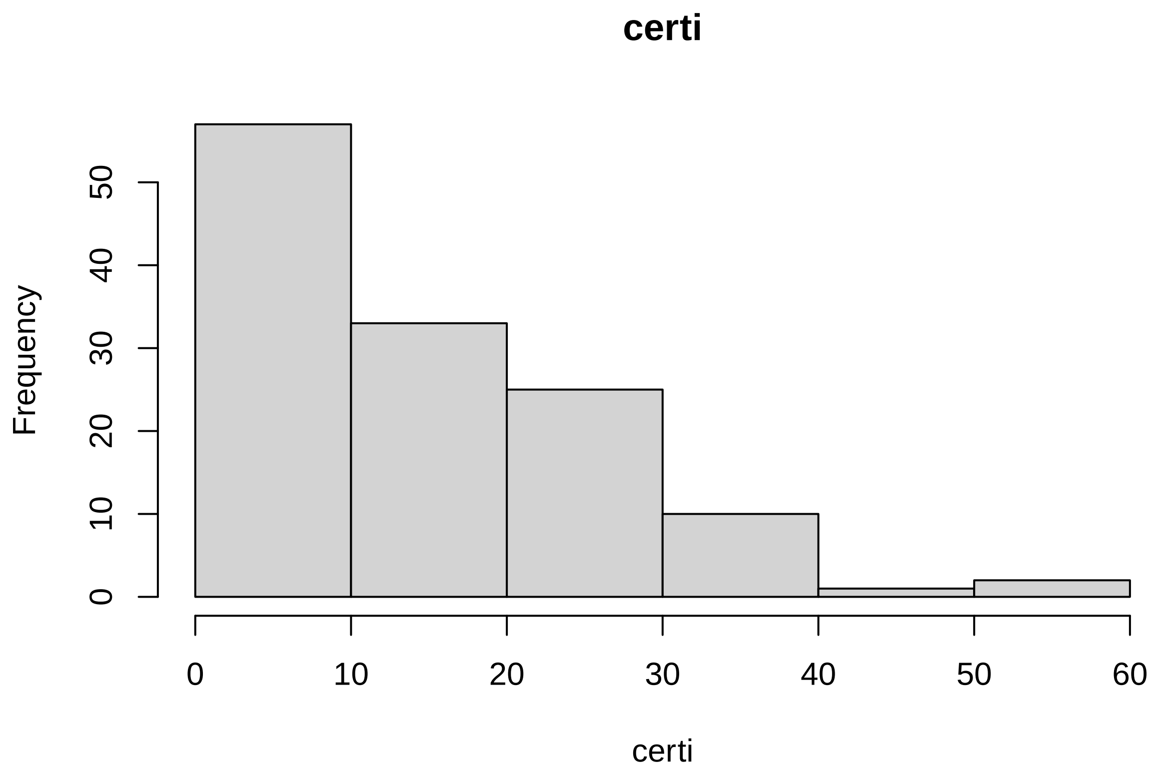
#model dataframe

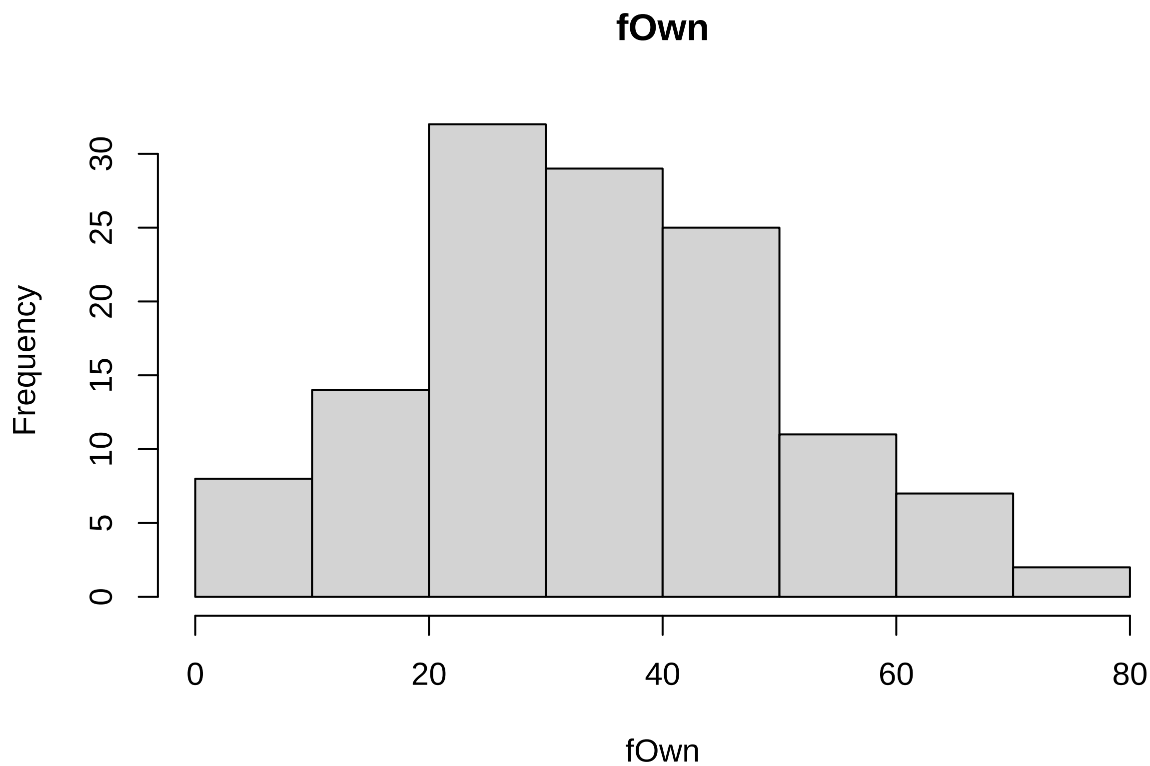
```
df_m <- df_m[model.list]
colnames(df_m)[3:length(colnames(df_m))] <- c("foreign","certi","fOwn","web","train", "bank","dOwn","ta
```

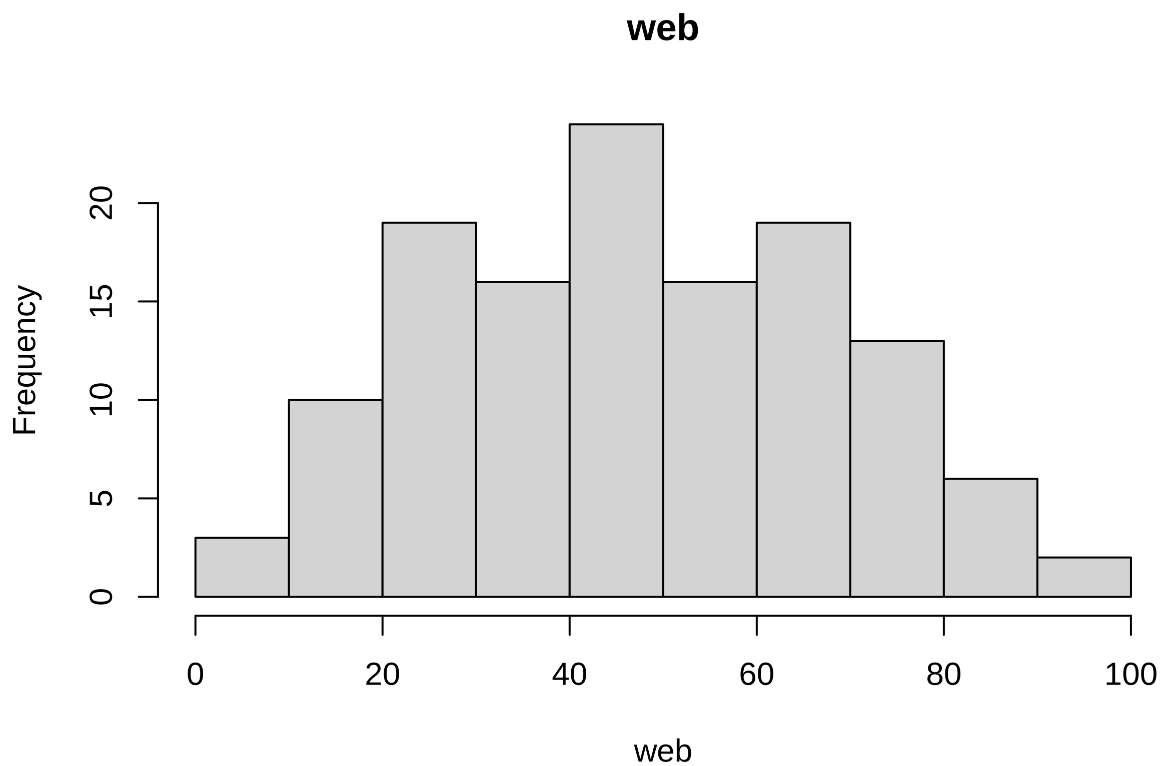
#historgrams for each variables

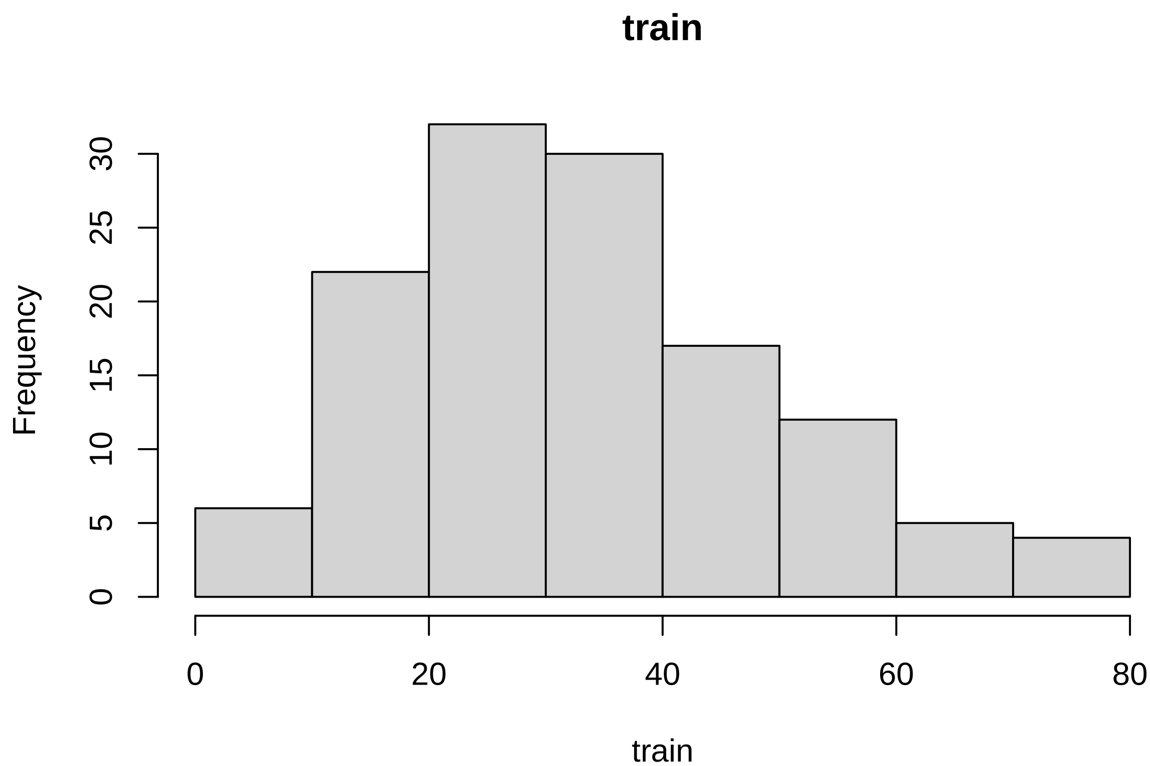
```r
for (i in colnames(df_m)[3:length(colnames(df_m))]){
  hist(df_m[,i], main = i, xlab = i)


}
```

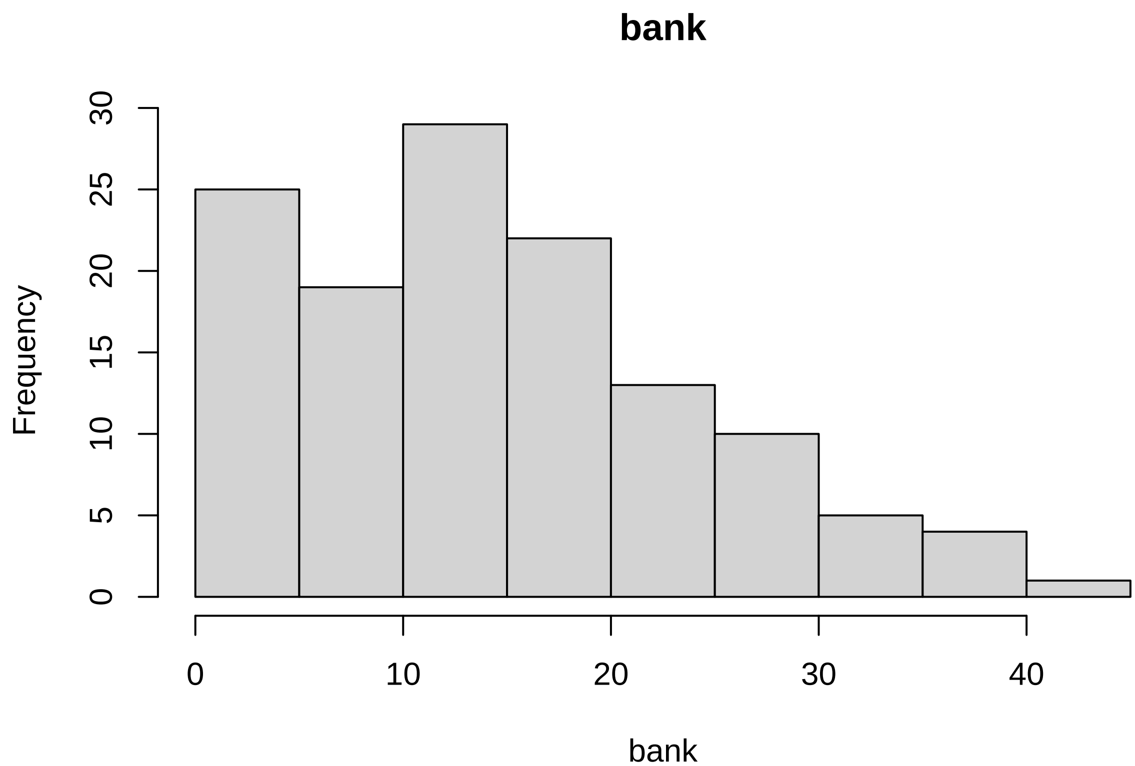


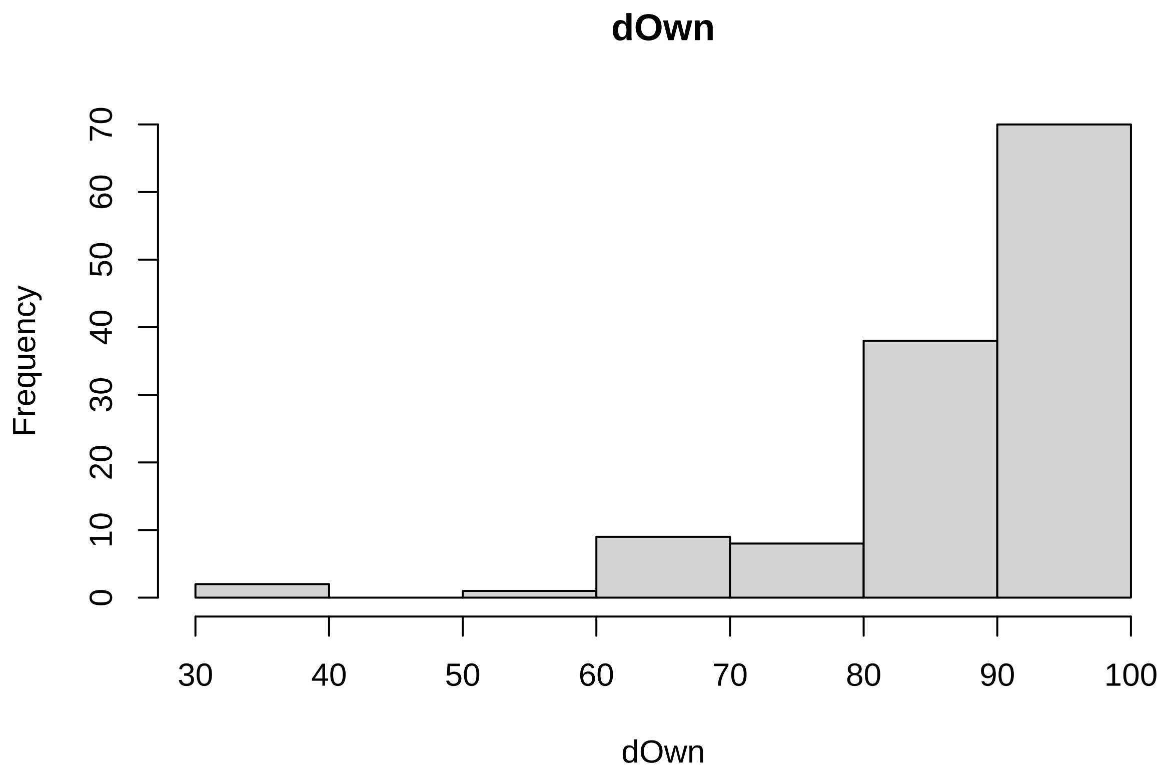

certi
Frequency
0
10
20
30
40
50
0
10
20
30
40
50
60
certi

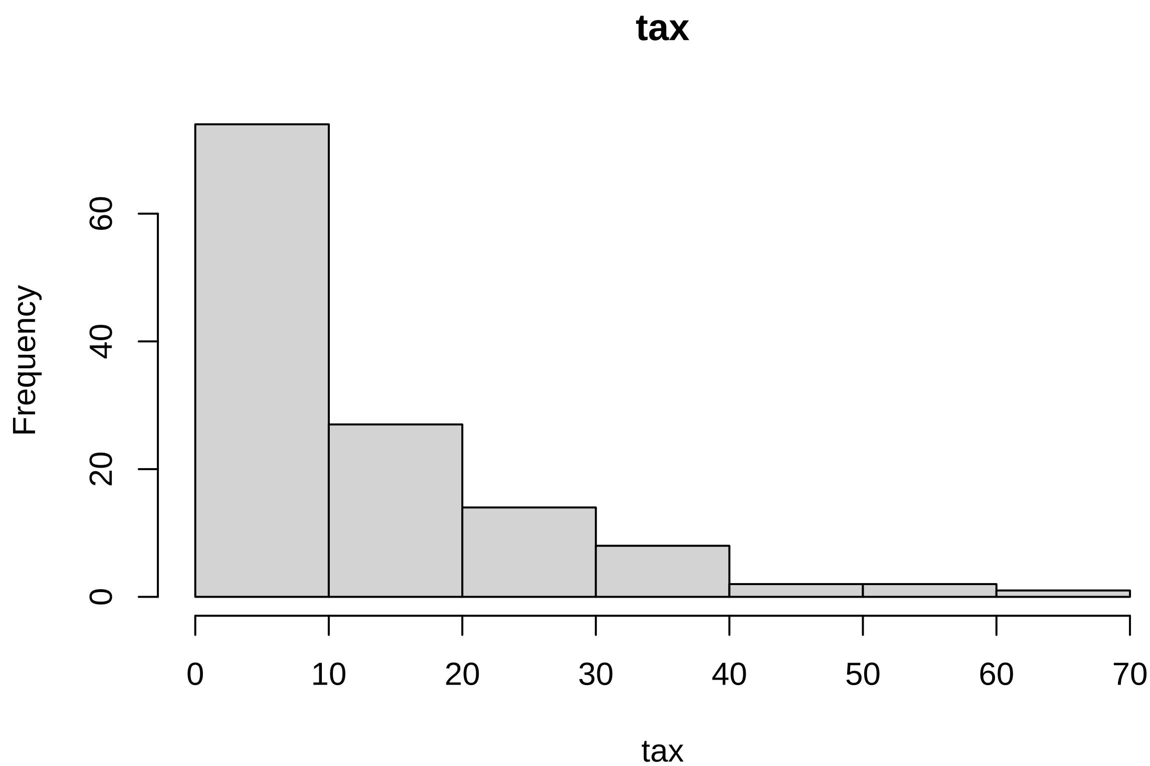

fOwn
Frequency
0
5
10
15
20
25
30
0
20
40
60
80
fOwn

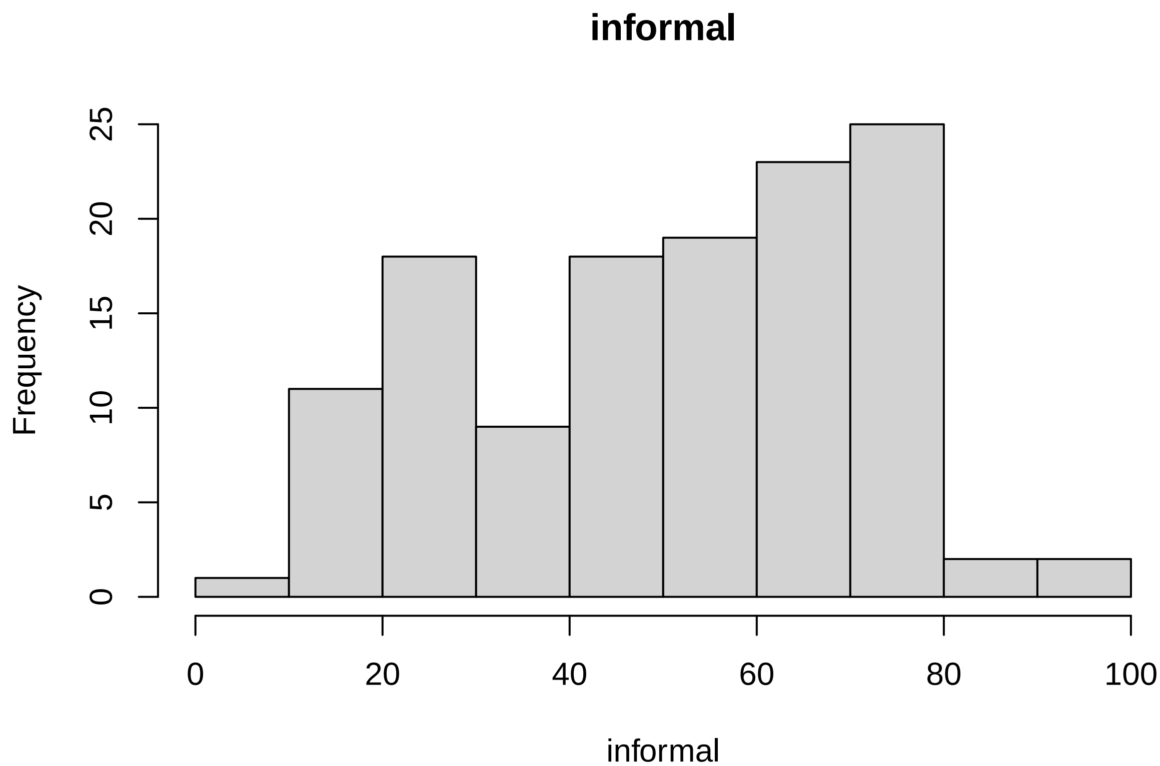

web
Frequency
0
5
10
15
20
0
20
40
60
80
100
web

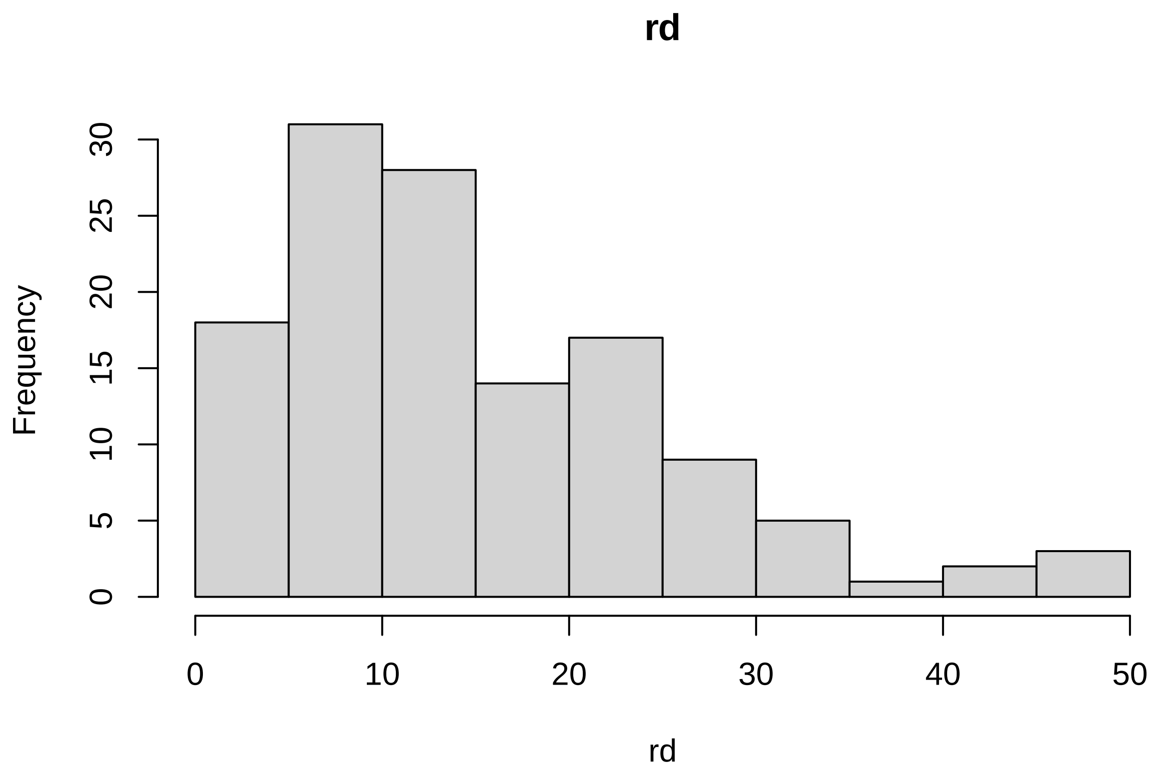

train
Frequency
0
5
10
15
20
25
30
0
20
40
60
80
train


bank
Frequency
0
5
10
15
20
25
30
0
10
20
30
40
bank


dOwn
Frequency
0
10
20
30
40
50
60
70
30
40
50
60
70
80
90
100
dOwn

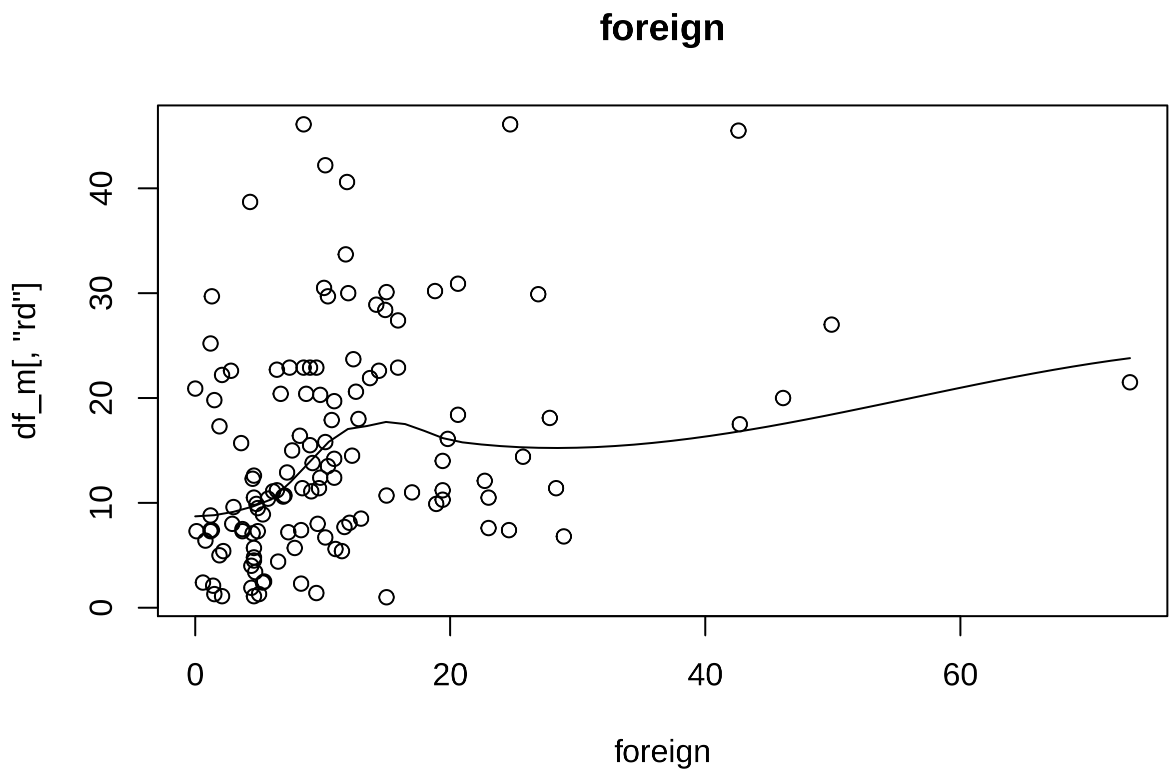

tax
Frequency
0
20
40
60
0
10
20
30
40
50
60
70
tax

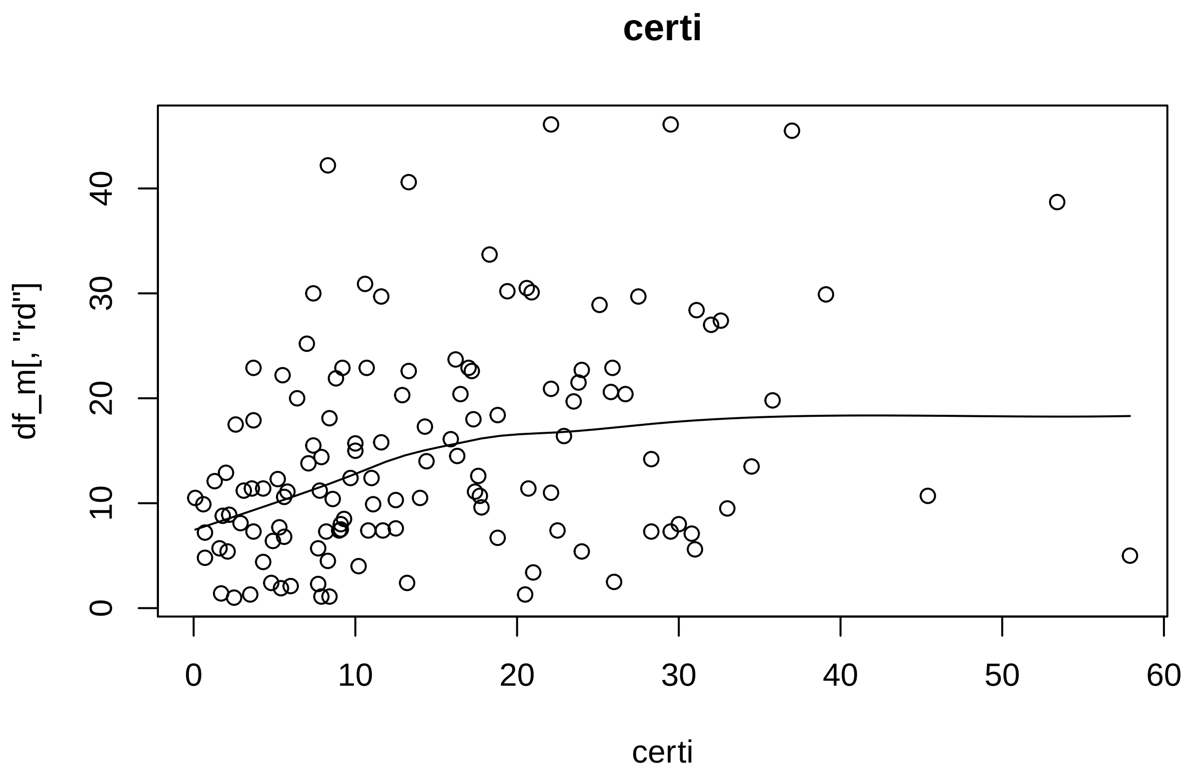

informal
Frequency
0
5
10
15
20
25
0
20
40
60
80
100
informal

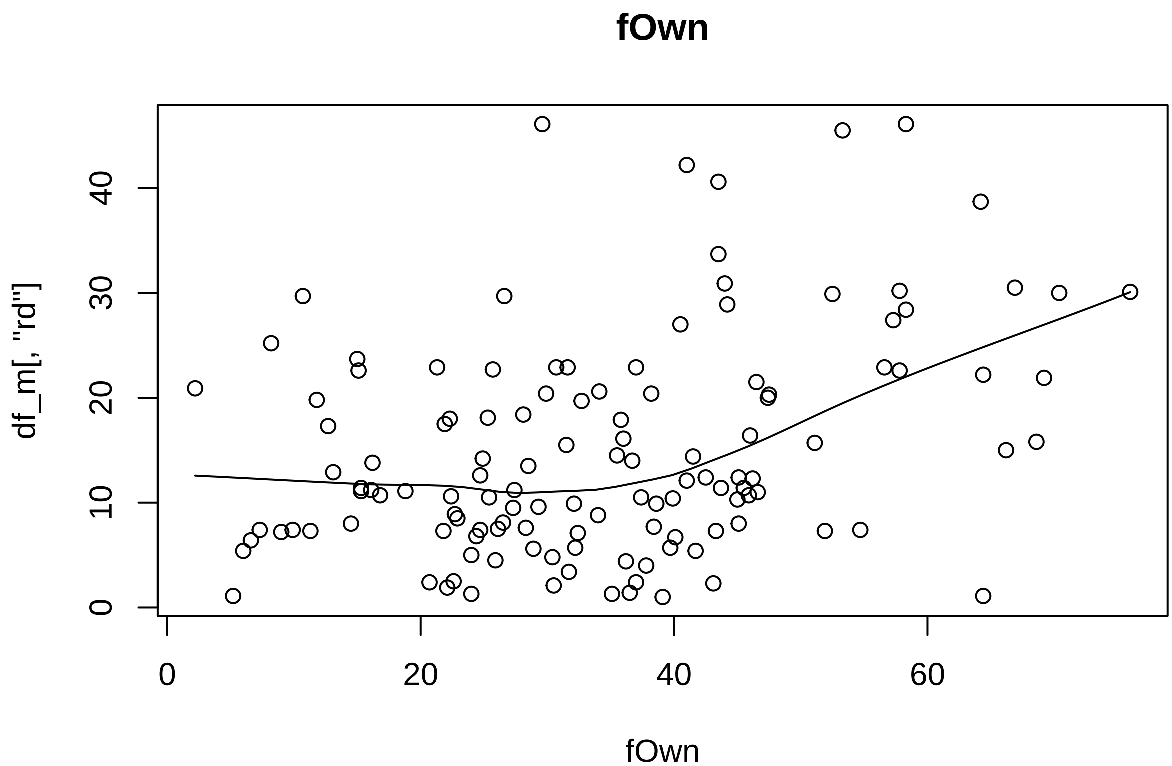


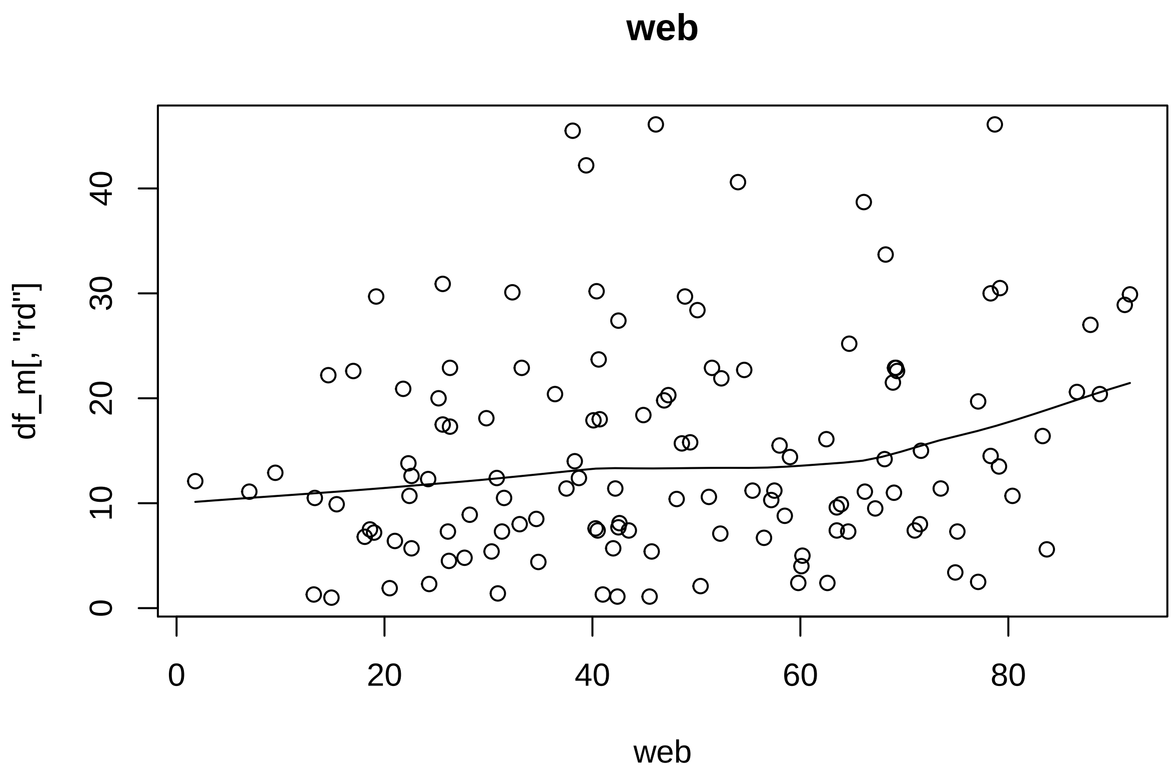
#scatter plot for each variables

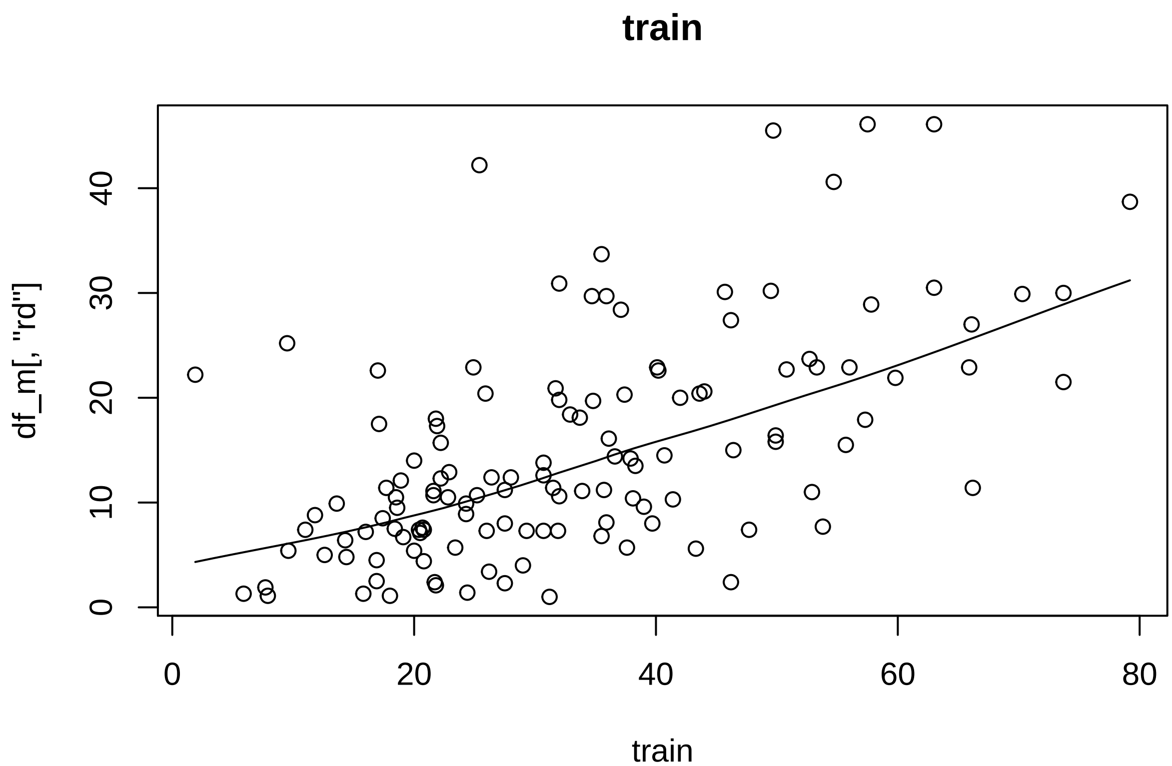
```
for (i in colnames(df_m)[3:length(colnames(df_m))]){
  scatter.smooth(df_m[,i], df_m[,"rd"], main = i, xlab = i)


}
```

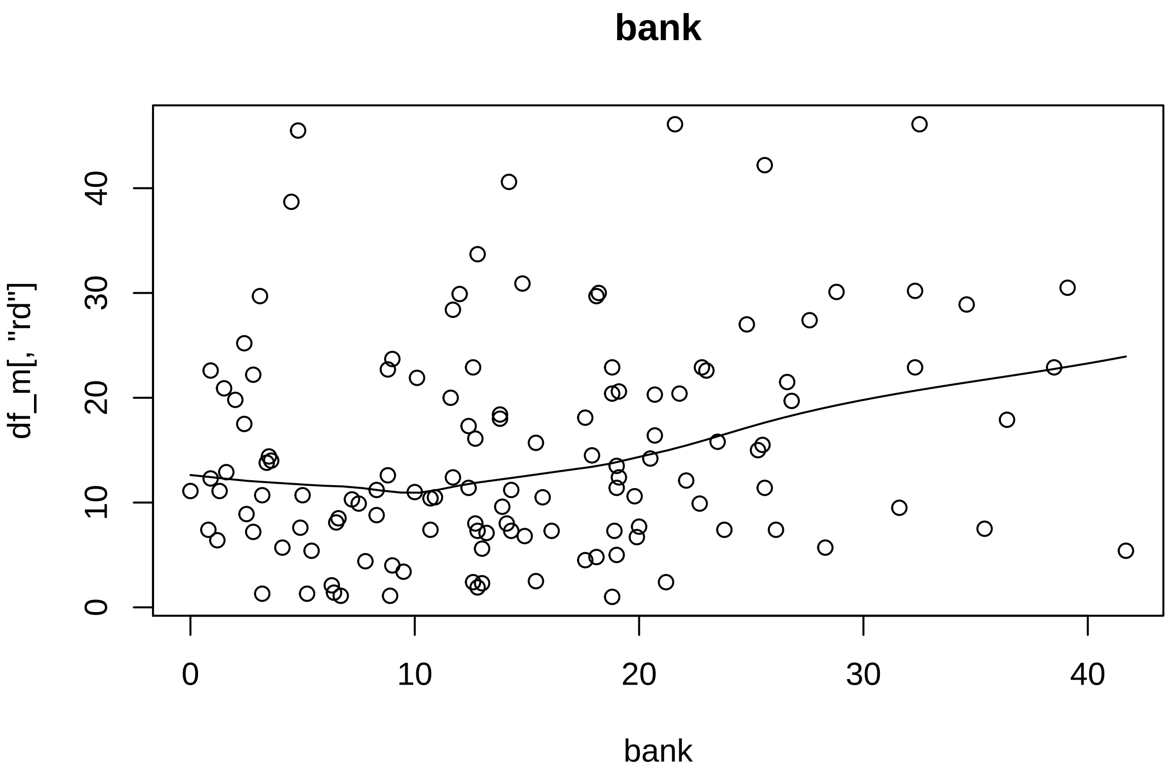

foreign
df_m[, "rd"]
0
10
20
30
40
0
20
40
60
foreign

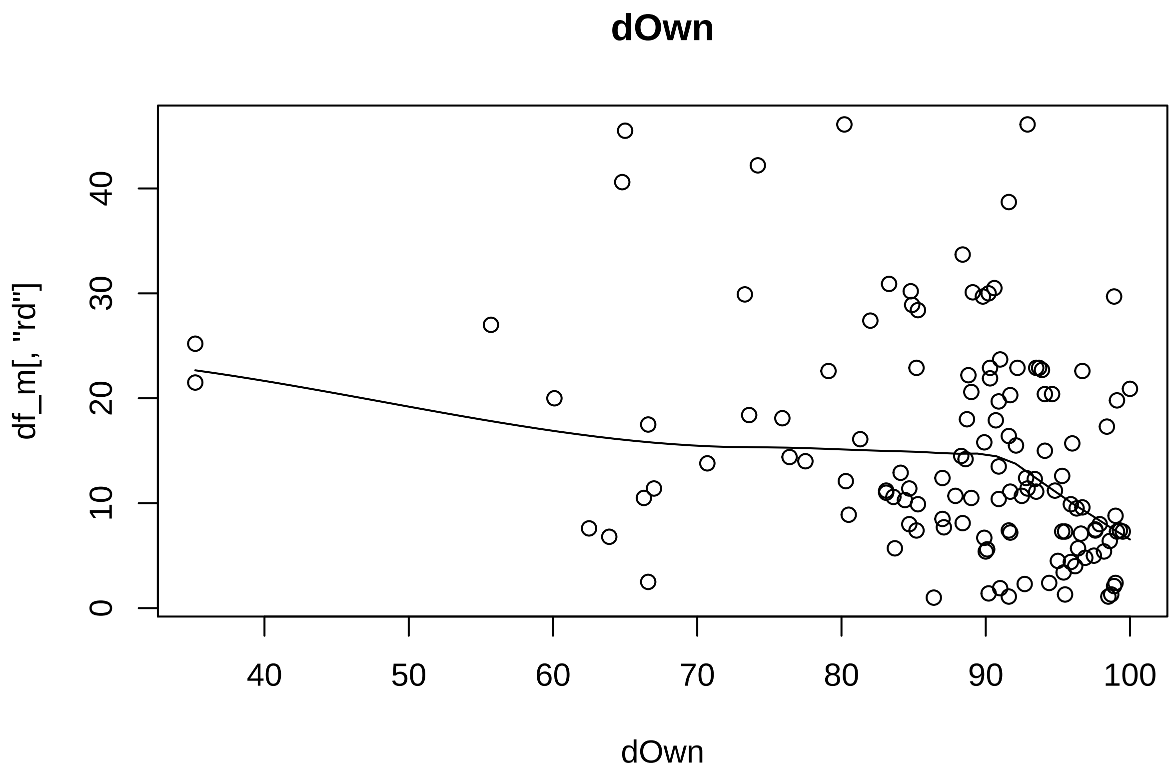

certi
df_m[, "rd"]
0
10
20
30
40
0
10
20
30
40
50
60
certi

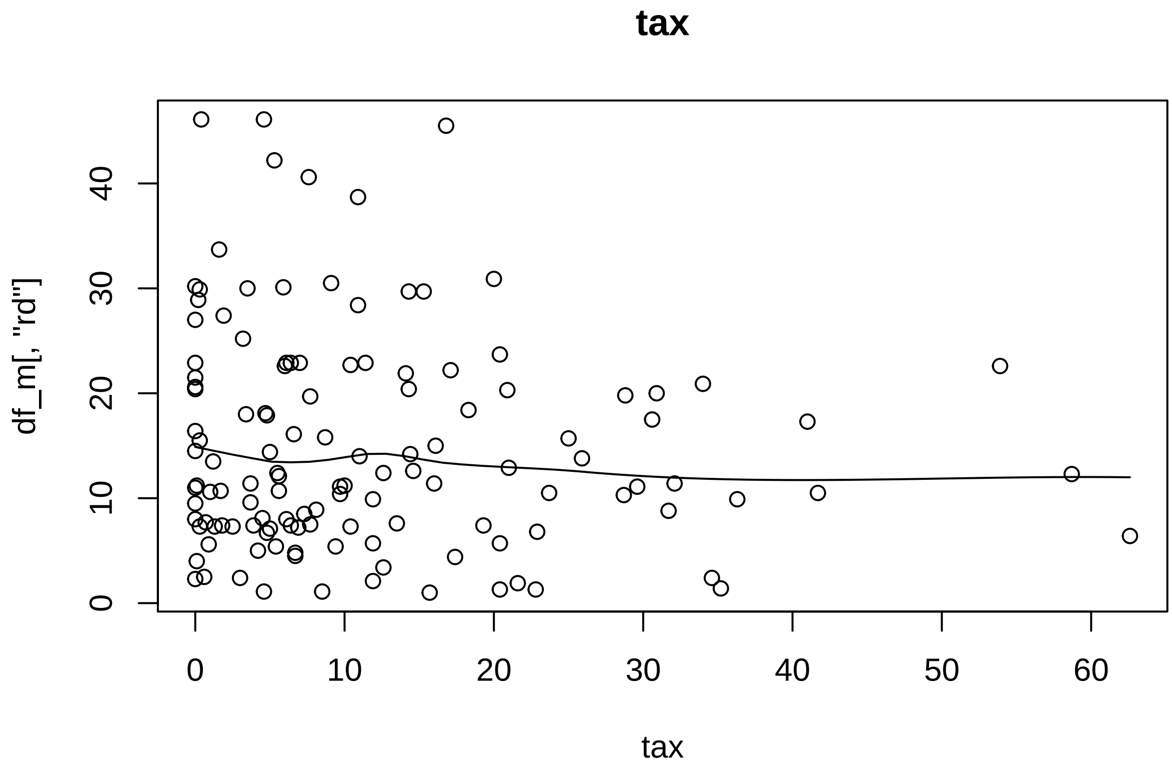

fOwn
df_m[, "rd"]
0
10
20
30
40
0
20
40
60
fOwn

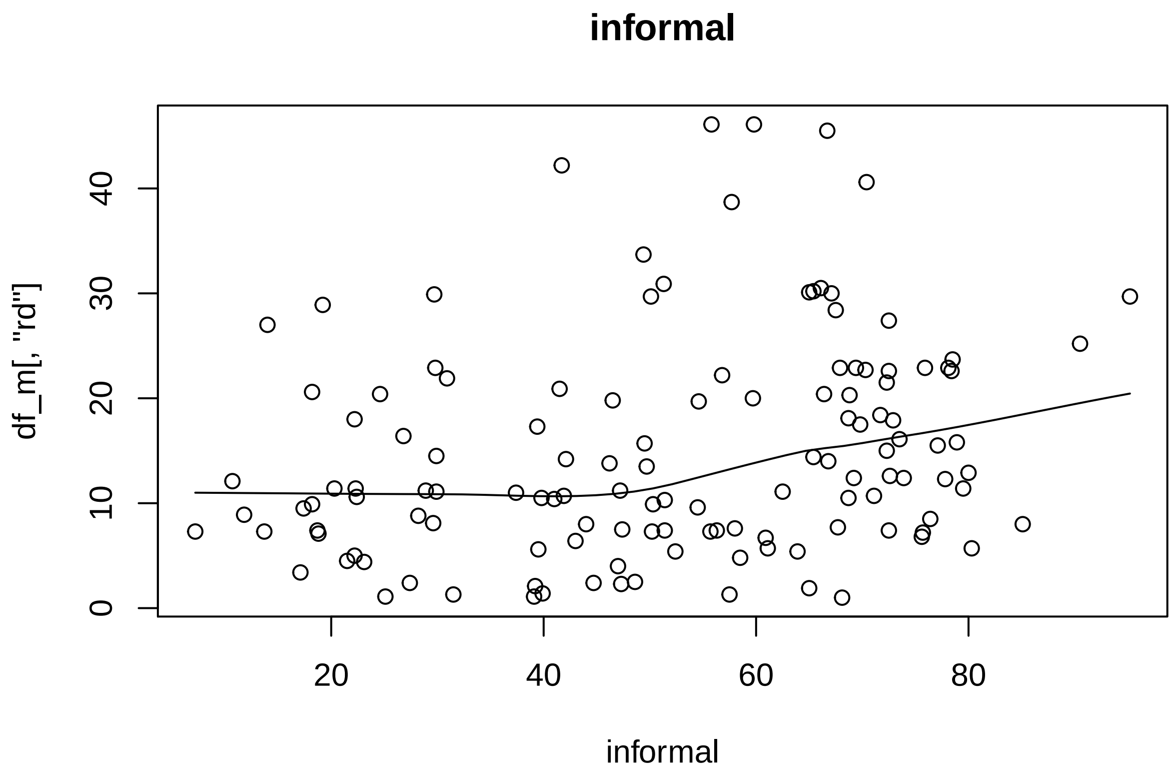

web
df_m[, "rd"]
0
10
20
30
40
0
20
40
60
80
web


train
df_m[, "rd"]
0
10
20
30
40
0
20
40
60
80
train

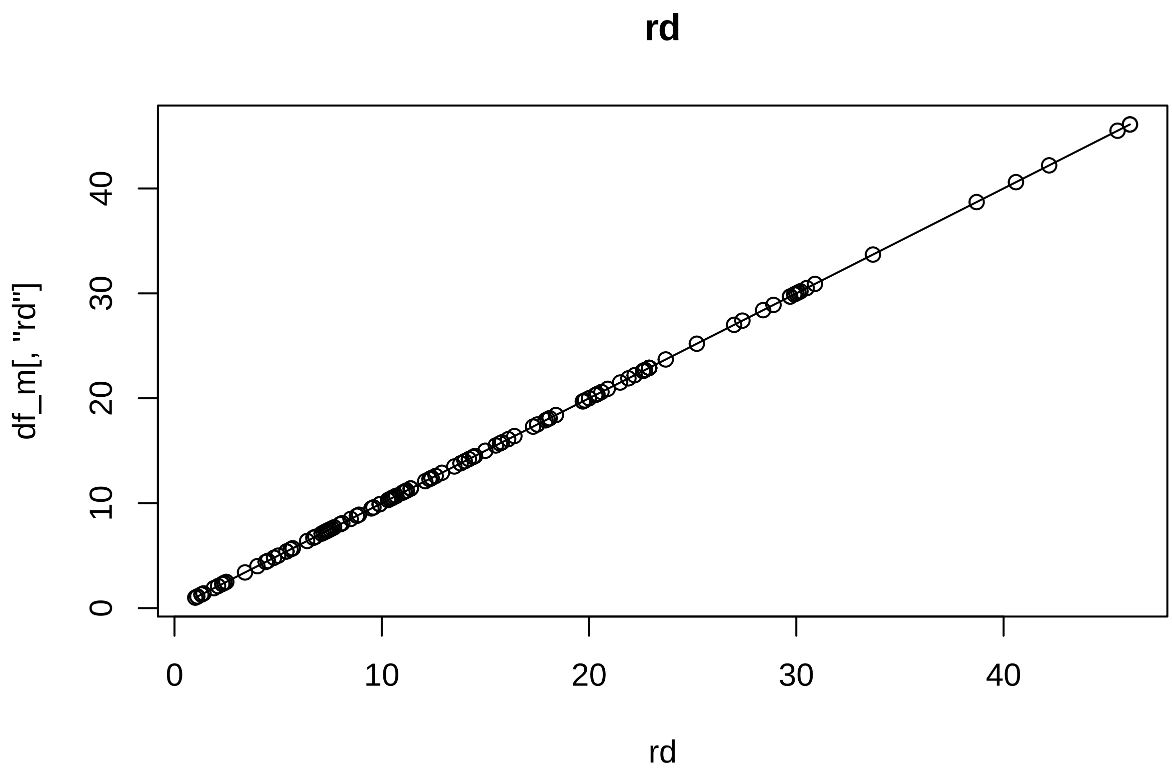

bank
df_m[, "rd"]
0
10
20
30
40
0
10
20
30
40
bank


dOwn
df_m[, "rd"]
0
10
20
30
40
40
50
60
70
80
90
100
dOwn


tax
df_m[, "rd"]
0
10
20
30
40
0
10
20
30
40
50
60
tax



informal
df_m[, "rd"]
0
10
20
30
40
20
40
60
80
informal

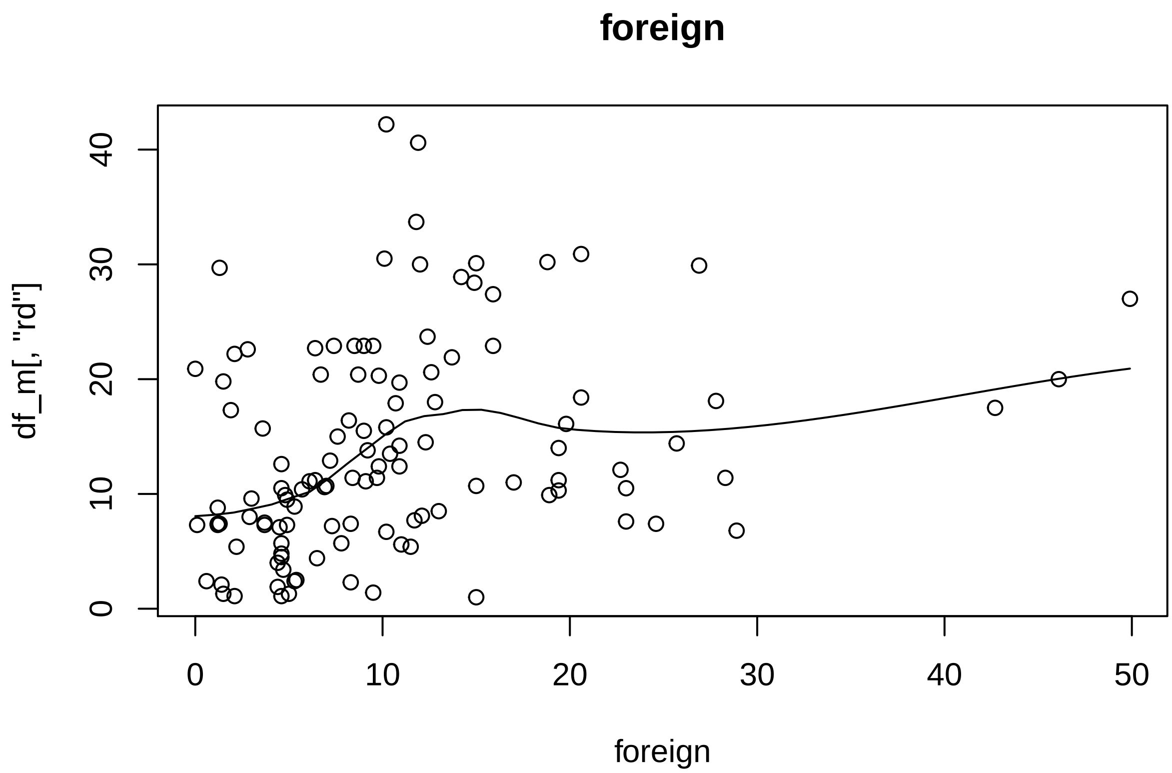


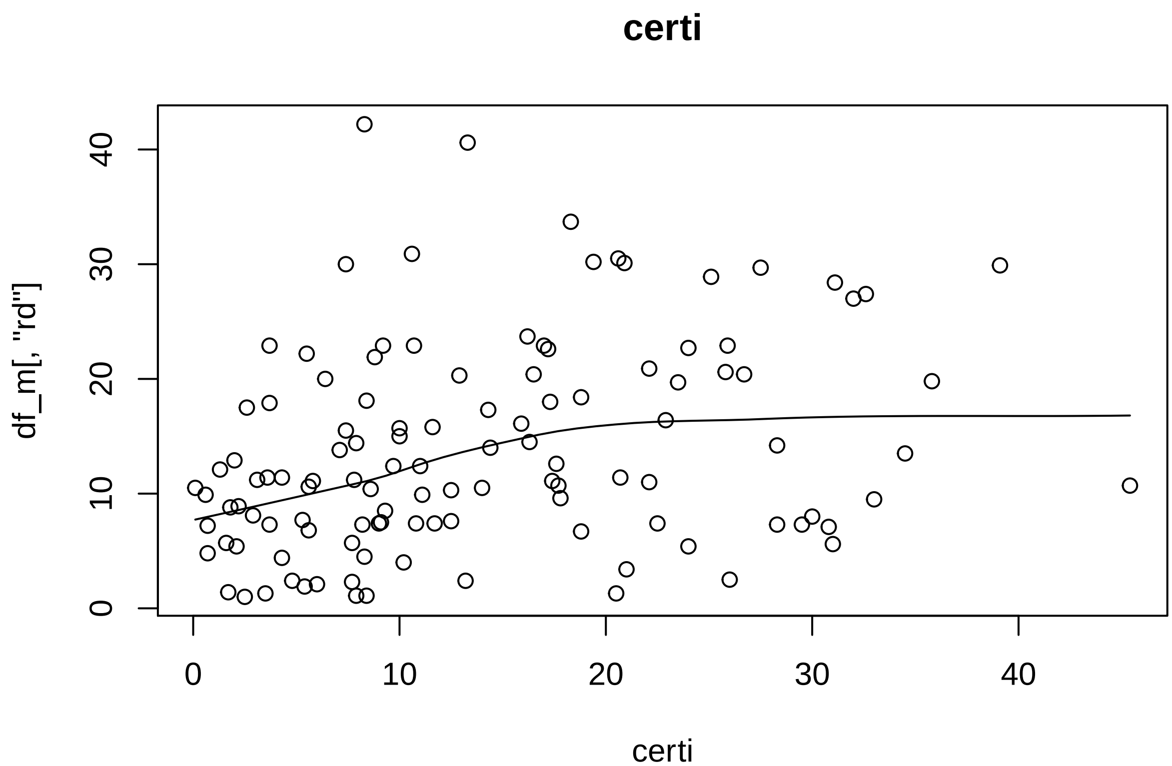
#dealing with outliers

```
df_m <- subset(df_m, subset = df_m$rd < 45)
df_m <- subset(df_m, subset = df_m$foreign < 60)
df_m <- subset(df_m, subset = df_m$certi < 50)
df_m <- subset(df_m, subset = df_m$dOwn > 50)
df_m <- subset(df_m, subset = df_m$tax < 50)
df_m <- subset(df_m, subset = df_m$informal < 85)
```

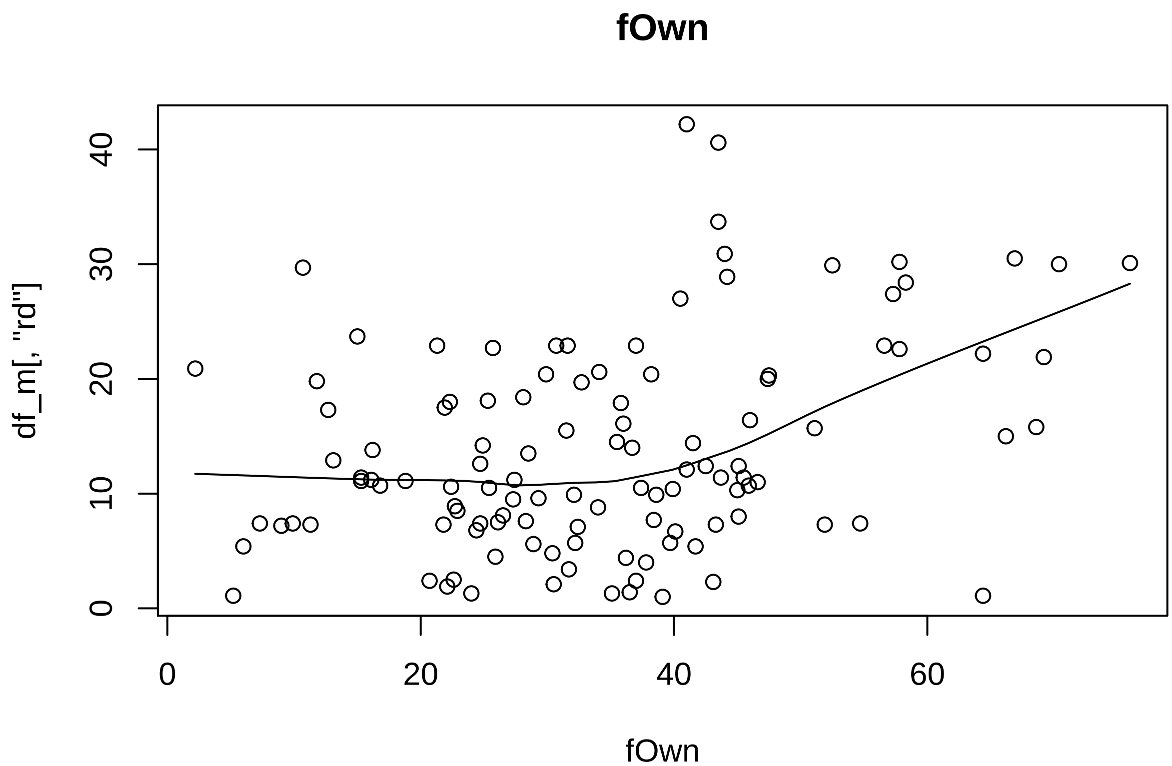
#scatter plot for each variables

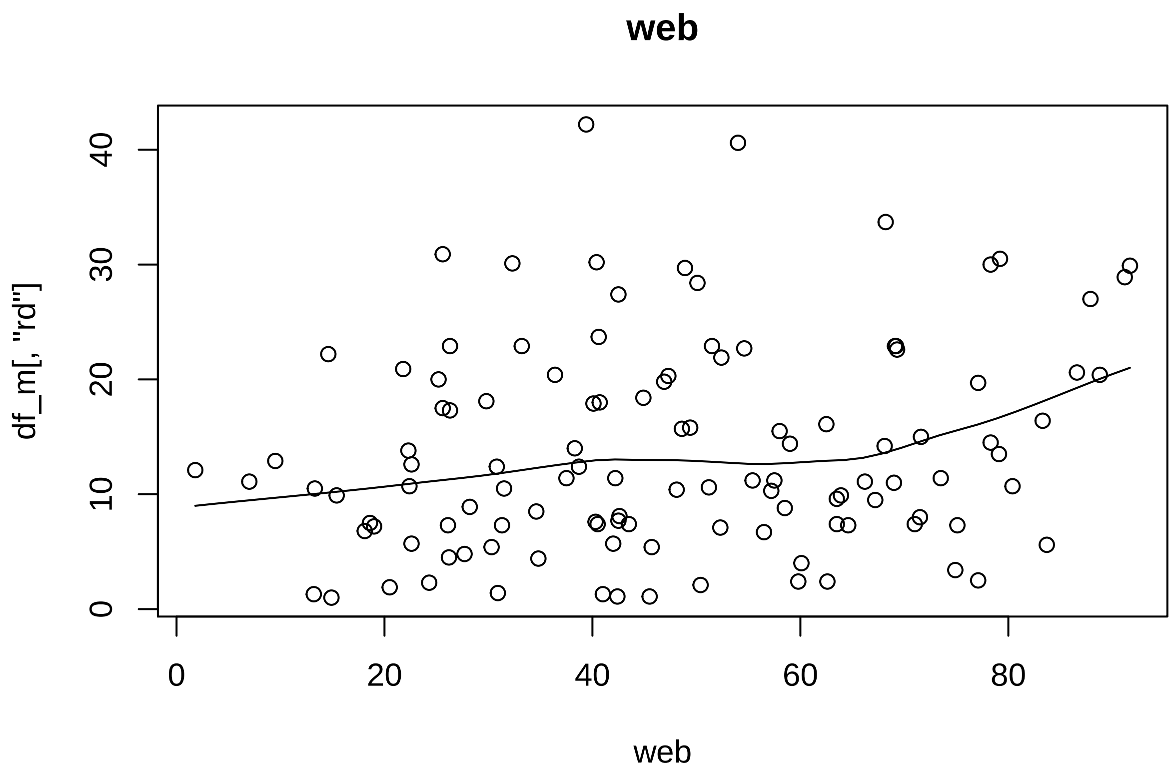
```
for (i in colnames(df_m)[3:length(colnames(df_m))]){
  scatter.smooth(df_m[,i], df_m[,"rd"], main = i, xlab = i)
}
```

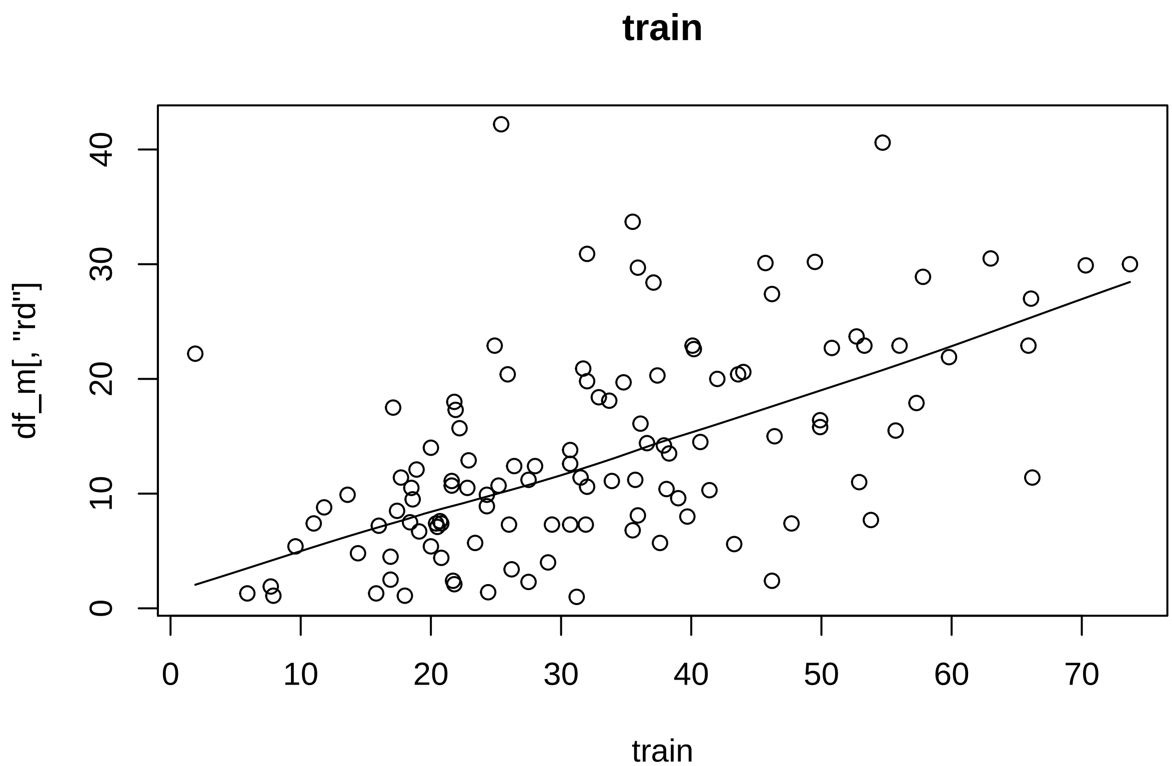

foreign
df_m[, "rd"]
foreign
0
10
20
30
40
50
0
10
20
30
40

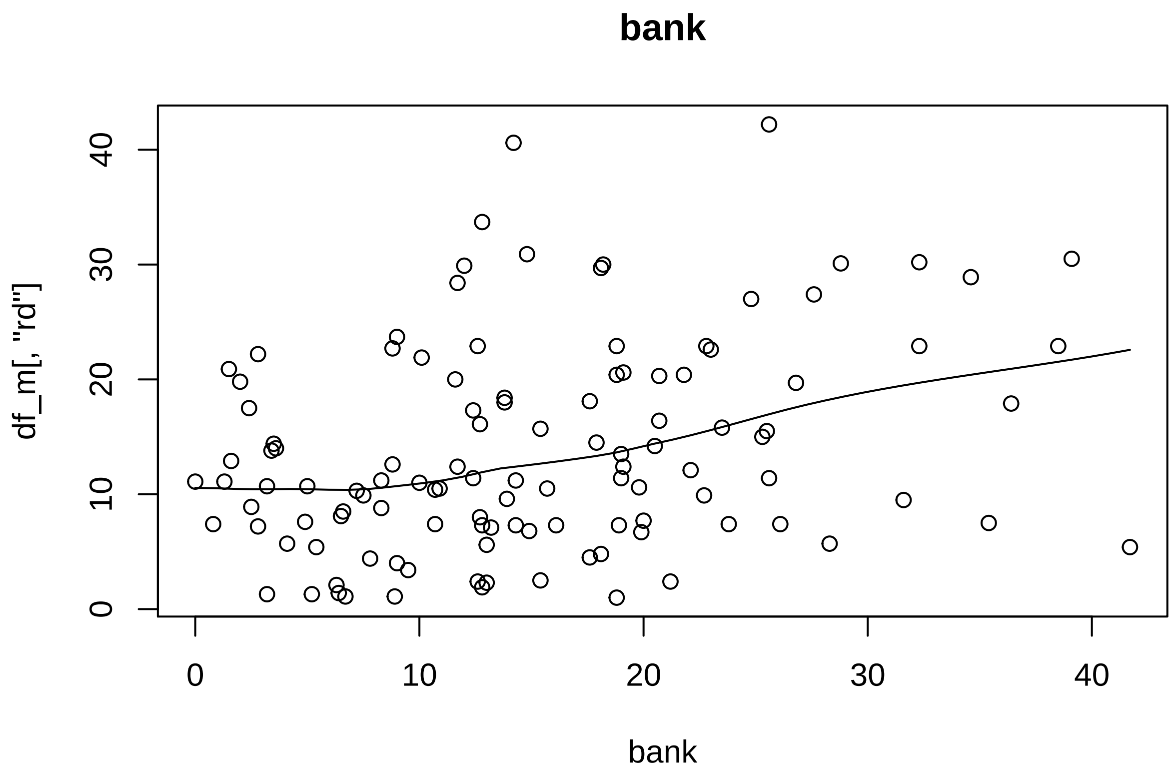


certi
40
30
20
10
0
df_m[, "rd"]
0
10
20
30
40
certi

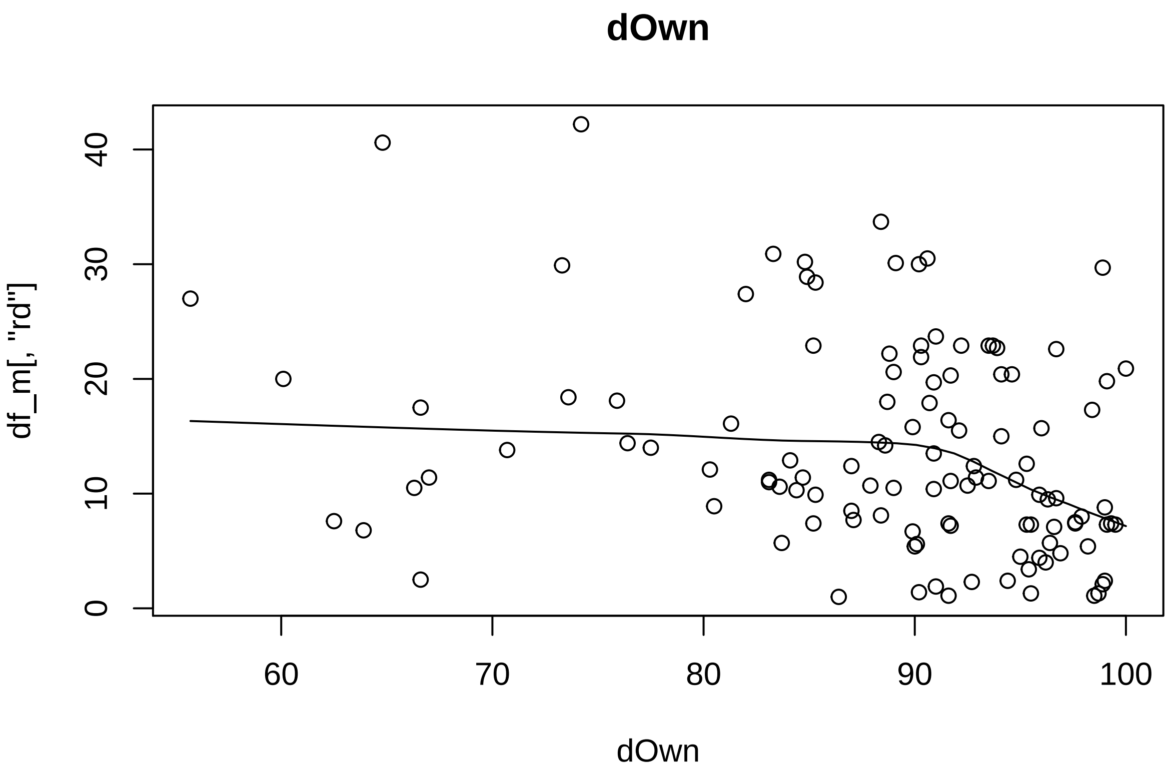

fOwn
df_m[, "rd"]
0
10
20
30
40
0
20
40
60
fOwn


web
df_m[, "rd"]
0
10
20
30
40
0
20
40
60
80
web

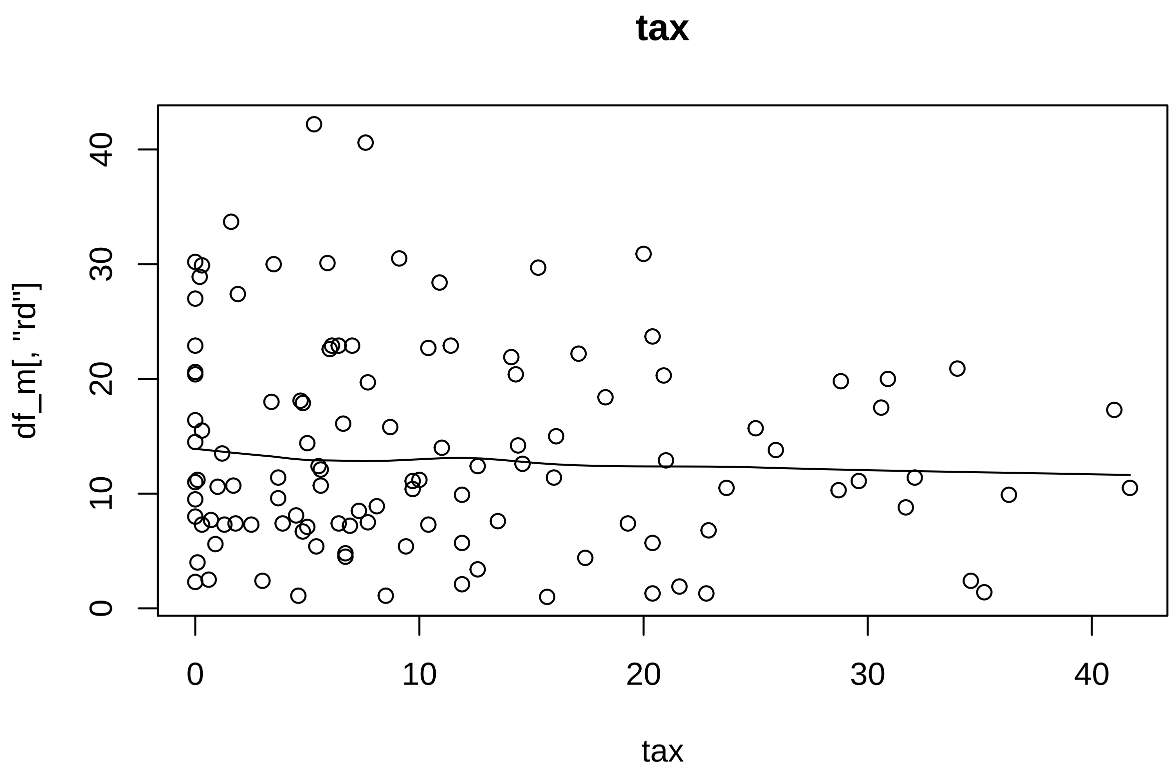

train
df_m[, "rd"]
0
10
20
30
40
0
10
20
30
40
50
60
70
train

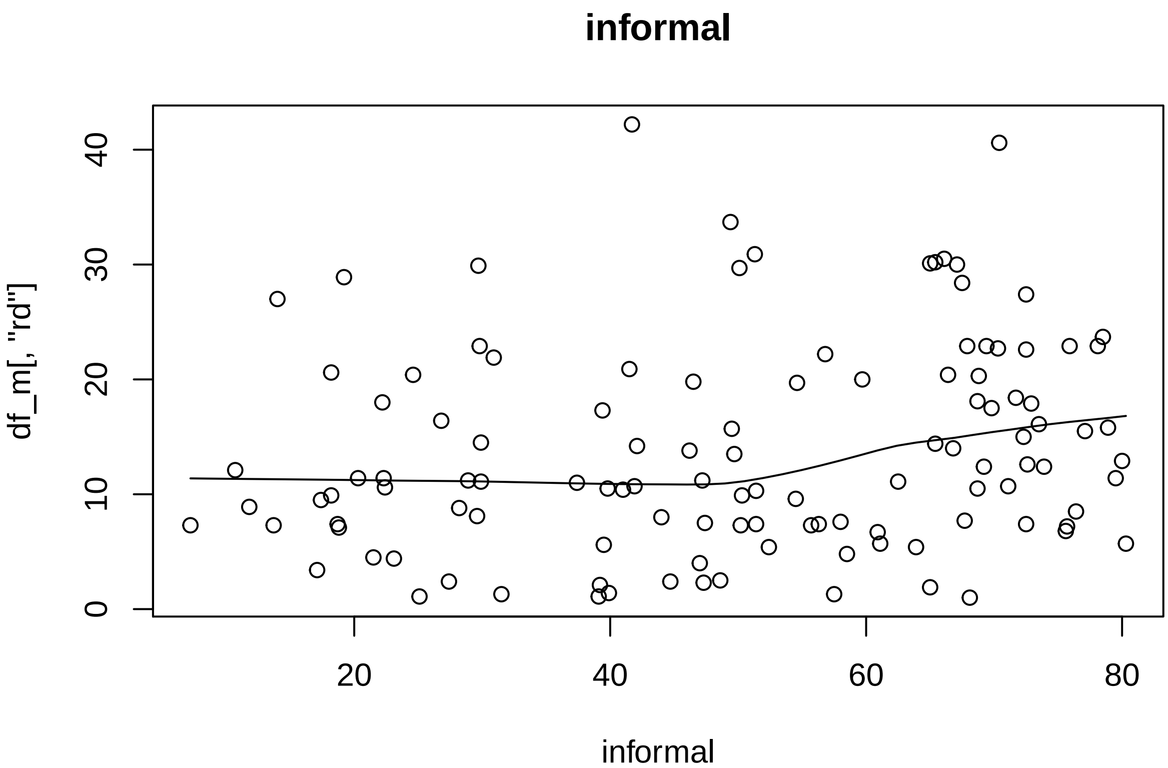


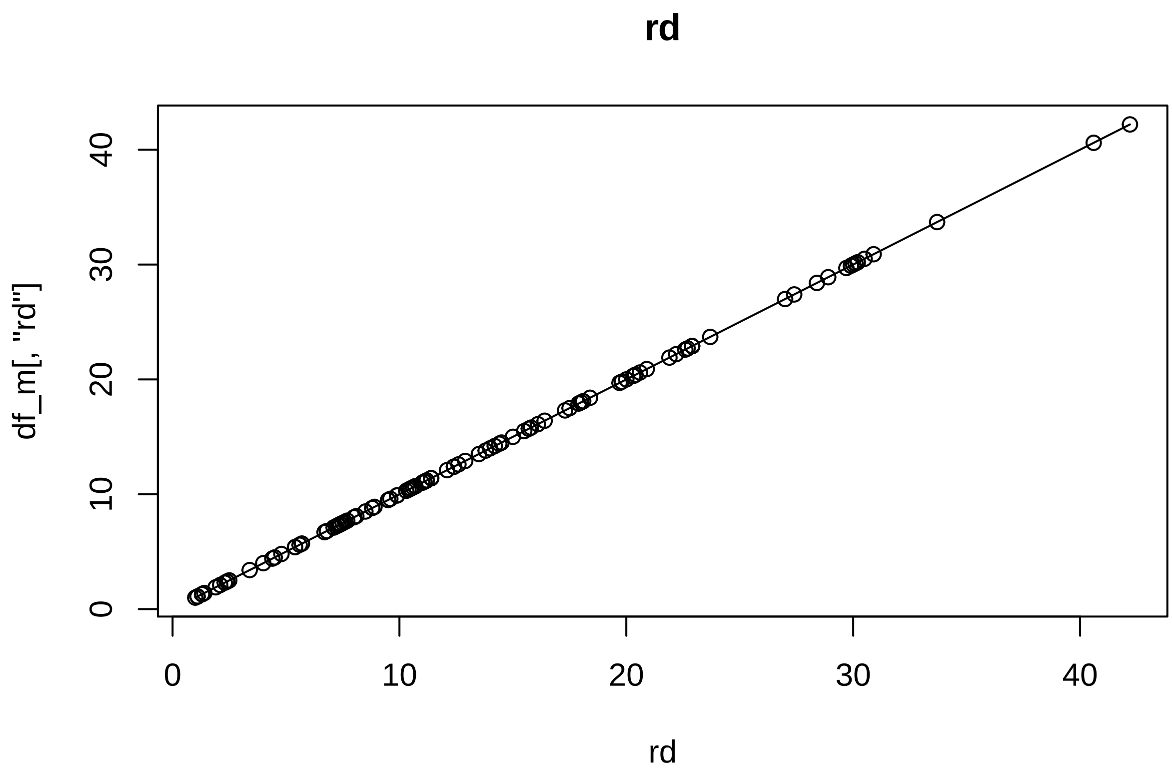
bank
df_m[, "rd"]
0
10
20
30
40
0
10
20
30
40
bank



dOwn
40
30
20
10
0
df_m[, "rd"]
60
70
80
90
100
dOwn


tax
df_m[, "rd"]
0
10
20
30
40
0
10
20
30
40
tax


informal
df_m[, "rd"]
0
10
20
30
40
20
40
60
80
informal



#showing there is no any missing values

```
nrow(na.omit(df_m))
```

```
## [1] 116
```

```
nrow((df_m))
```

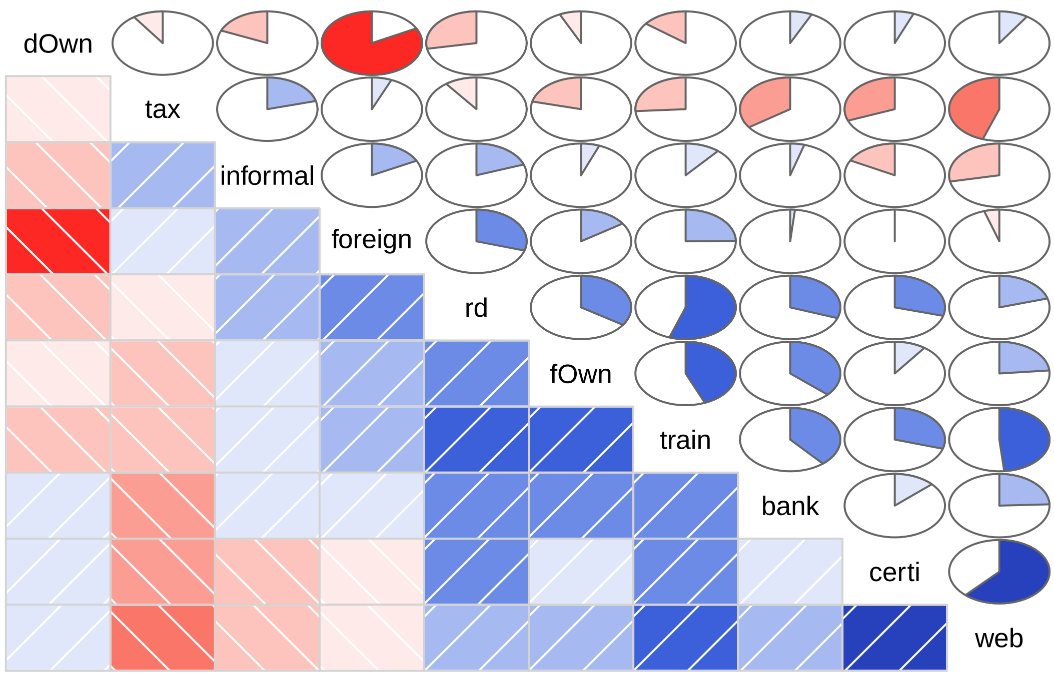
```
## [1] 116
```

#corr matrix

```
library(corrgram)
```

```
##
## Attaching package: 'corrgram'

## The following object is masked from 'package:lattice':
##
##     panel.fill

## The following object is masked from 'package:plyr':
##
##     baseball
```

```r
corrgram(df_m,order=TRUE, lower.panel=panel.shade,
  upper.panel=panel.pie, text.panel=panel.txt)
```



#corr matrix

```r
nums <- unlist(lapply(df_m, is.numeric))
mydata.rcorr <- rcorr(as.matrix(df_m[,nums]))
c <- as.data.frame(mydata.rcorr$r)
c
```

```
##               foreign        certi        fOwn         web      train
## foreign   1.000000000 -0.006577648  0.15533839 -0.04888926  0.2477403
## certi    -0.006577648  1.000000000  0.10050674  0.61787549  0.2967015
## fOwn      0.155338393  0.100506740  1.00000000  0.23479274  0.4374069
## web      -0.048889258  0.617875490  0.23479274  1.00000000  0.4845754
## train     0.247740340  0.296701513  0.43740693  0.48457537  1.0000000
## bank      0.015239674  0.132223004  0.36630985  0.24342718  0.3871571
## dOwn     -0.826937933  0.061317712 -0.06979685  0.09261499 -0.1452550
## tax       0.063597072 -0.309035283 -0.21122667 -0.44650793 -0.2598369
## informal  0.174420413 -0.172736947  0.05919286 -0.28175223  0.1114369
## rd        0.297193978  0.292824998  0.34267880  0.20447037  0.5524317
##                 bank        dOwn         tax    informal        rd
## foreign   0.01523967 -0.82693793  0.06359707  0.17442041 0.2971940
## certi     0.13222300  0.06131771 -0.30903528 -0.17273695 0.2928250
## fOwn      0.36630985 -0.06979685 -0.21122667  0.05919286 0.3426788
## web       0.24342718  0.09261499 -0.44650793 -0.28175223 0.2044704
```

```
## train     0.38715715 -0.14525505 -0.25983688  0.11143693  0.5524317
## bank      1.00000000  0.07035361 -0.34680709  0.04547343  0.3049863
## dOwn      0.07035361  1.00000000 -0.09671298 -0.18528855 -0.2785461
## tax      -0.34680709 -0.09671298  1.00000000  0.20844496 -0.1029323
## informal  0.04547343 -0.18528855  0.20844496  1.00000000  0.1949148
## rd        0.30498635 -0.27854609 -0.10293234  0.19491484  1.0000000
```

#there are high correaltion between foreing & dOwn, certi & web.

#simple regression for each variables with Percent of firms that spend on R&D

```r
for (i in colnames(df_m)[3:(length(colnames(df_m))-1)]){
  model <- lm(rd ~ ., df_m[c("rd",i)])
  print(summary(model))
}
```

```
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.080  -6.148  -2.071   6.171  28.559
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.58188    1.26068   8.394 1.46e-13 ***
## foreign      0.29987    0.09023   3.323   0.0012 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.671 on 114 degrees of freedom
## Multiple R-squared:  0.08832,    Adjusted R-squared:  0.08033
## F-statistic: 11.04 on 1 and 114 DF,  p-value: 0.001196
##
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.4564  -5.7194  -0.7811   5.8237  29.9088
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.10358    1.38998   7.269 4.87e-11 ***
## certi        0.26357    0.08061   3.270  0.00142 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.683 on 114 degrees of freedom
## Multiple R-squared:  0.08575,    Adjusted R-squared:  0.07773
## F-statistic: 10.69 on 1 and 114 DF,  p-value: 0.001423
##
```

```
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.634  -5.414  -1.439   5.942  27.115
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.93873    1.93302   3.590 0.000490 ***
## fOwn         0.19869    0.05102   3.895 0.000166 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.531 on 114 degrees of freedom
## Multiple R-squared:  0.1174, Adjusted R-squared:  0.1097
## F-statistic: 15.17 on 1 and 114 DF,  p-value: 0.0001662
##
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.841  -6.006  -1.731   5.881  29.098
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.71681    2.01071   4.833 4.24e-06 ***
## web          0.08591    0.03852   2.230   0.0277 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.889 on 114 degrees of freedom
## Multiple R-squared:  0.04181,    Adjusted R-squared:  0.0334
## F-statistic: 4.974 on 1 and 114 DF,  p-value: 0.02769
##
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.8087  -4.6459  -0.8223   3.2174  30.8266
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.02634    1.67769   1.804   0.0739 .
## train        0.32862    0.04644   7.076 1.28e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.569 on 114 degrees of freedom
```

```
## Multiple R-squared:  0.3052, Adjusted R-squared:  0.2991
## F-statistic: 50.07 on 1 and 114 DF,  p-value: 1.285e-10
##
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.206  -6.378  -1.384   4.904  27.084
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.33866    1.53355   6.090 1.57e-08 ***
## bank         0.29417    0.08603   3.419 0.000872 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.648 on 114 degrees of freedom
## Multiple R-squared:  0.09302,    Adjusted R-squared:  0.08506
## F-statistic: 11.69 on 1 and 114 DF,  p-value: 0.0008719
##
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.029  -5.886  -2.137   6.643  24.663
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.99278    7.53144   4.912 3.04e-06 ***
## dOwn        -0.26221    0.08468  -3.097  0.00246 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.721 on 114 degrees of freedom
## Multiple R-squared:  0.07759,    Adjusted R-squared:  0.0695
## F-statistic: 9.589 on 1 and 114 DF,  p-value: 0.002464
##
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.281  -7.030  -1.926   6.322  27.881
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.78764    1.22174  12.104   <2e-16 ***
## tax         -0.08841    0.08001  -1.105    0.272
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.033 on 114 degrees of freedom
## Multiple R-squared:  0.0106, Adjusted R-squared:  0.001916
## F-statistic: 1.221 on 1 and 114 DF,  p-value: 0.2715
##
##
## Call:
## lm(formula = rd ~ ., data = df_m[c("rd", i)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.382  -6.919  -1.543   5.689  29.107
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.47767    2.20118   4.306 3.54e-05 ***
## informal     0.08670    0.04086   2.122    0.036 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.907 on 114 degrees of freedom
## Multiple R-squared:  0.03799,    Adjusted R-squared:  0.02955
## F-statistic: 4.502 on 1 and 114 DF,  p-value: 0.03602
```

#lineral model with all variables

```r
model <- lm(rd ~ ., subset(df_m, select = -c(country, code)))
summary(model)
```

```
##
## Call:
## lm(formula = rd ~ ., data = subset(df_m, select = -c(country,
##     code)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.6912  -3.5780  -0.8568   4.2476  26.7149
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.49882   12.53097   1.556  0.12268
## foreign     -0.05806    0.13656  -0.425  0.67160
## certi        0.26084    0.08445   3.089  0.00257 **
## fOwn         0.06659    0.04905   1.358  0.17745
## web         -0.07355    0.04718  -1.559  0.12201
## train        0.25721    0.05917   4.347 3.17e-05 ***
## bank         0.12738    0.08268   1.541  0.12636
## dOwn        -0.23035    0.12506  -1.842  0.06828 .
## tax          0.04503    0.07416   0.607  0.54497
## informal     0.03933    0.03613   1.089  0.27882
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 7.107 on 106 degrees of freedom
## Multiple R-squared:  0.4305, Adjusted R-squared:  0.3821
## F-statistic: 8.902 on 9 and 106 DF,  p-value: 6.785e-10
```

#linear model by dropping web dOwn variables based on corr table

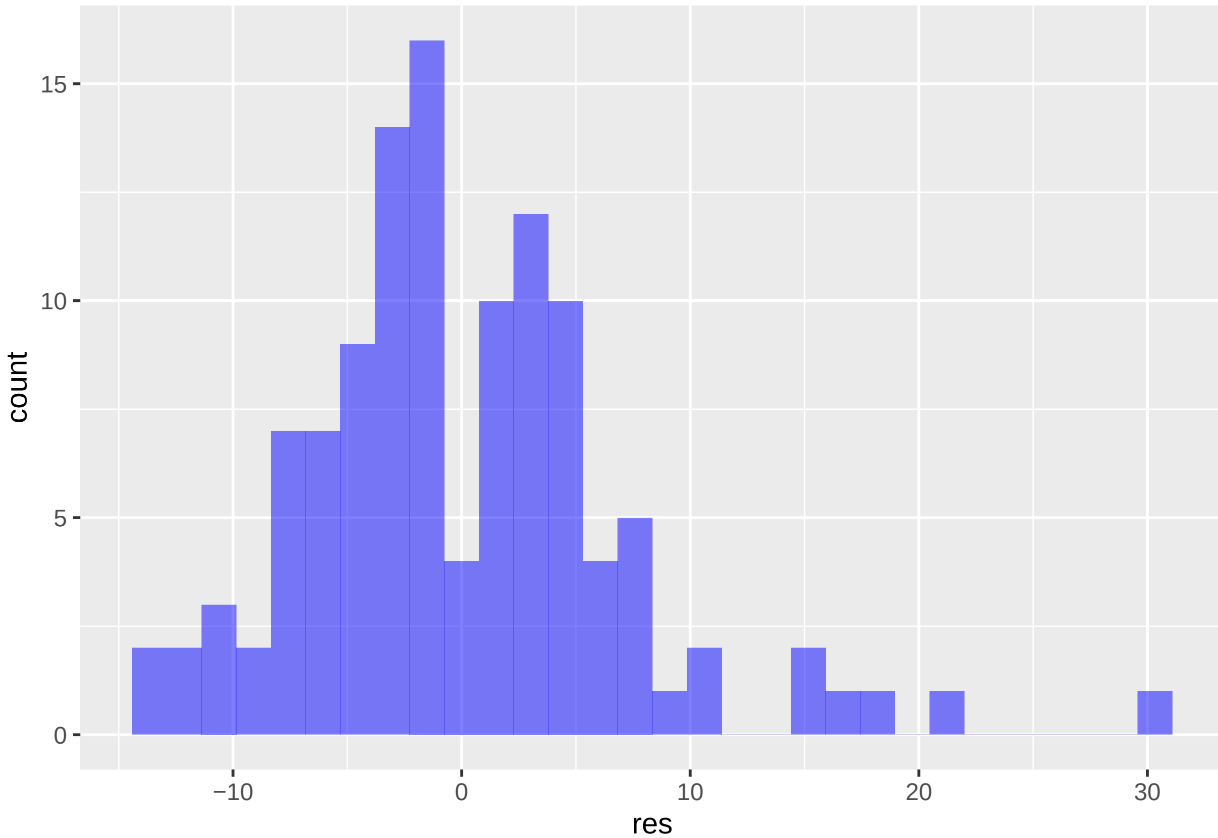
```
model <- lm(rd ~ ., subset(df_m, select = -c(country, code, web, dOwn)))
summary(model)
```

```
##
## Call:
## lm(formula = rd ~ ., data = subset(df_m, select = -c(country,
##     code, web, dOwn)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.483  -4.385  -1.195   3.647  30.498
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.23029    2.87158  -1.821 0.071315 .
## foreign      0.16306    0.07945   2.052 0.042563 *
## certi        0.18654    0.07380   2.528 0.012931 *
## fOwn         0.05940    0.04978   1.193 0.235397
## train        0.21813    0.05531   3.944 0.000143 ***
## bank         0.11338    0.08363   1.356 0.177987
## tax          0.06932    0.07344   0.944 0.347331
## informal     0.05935    0.03525   1.684 0.095151 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.227 on 108 degrees of freedom
## Multiple R-squared:  0.3999, Adjusted R-squared:  0.361
## F-statistic: 10.28 on 7 and 108 DF,  p-value: 7.966e-10
```

#histogram for residuals

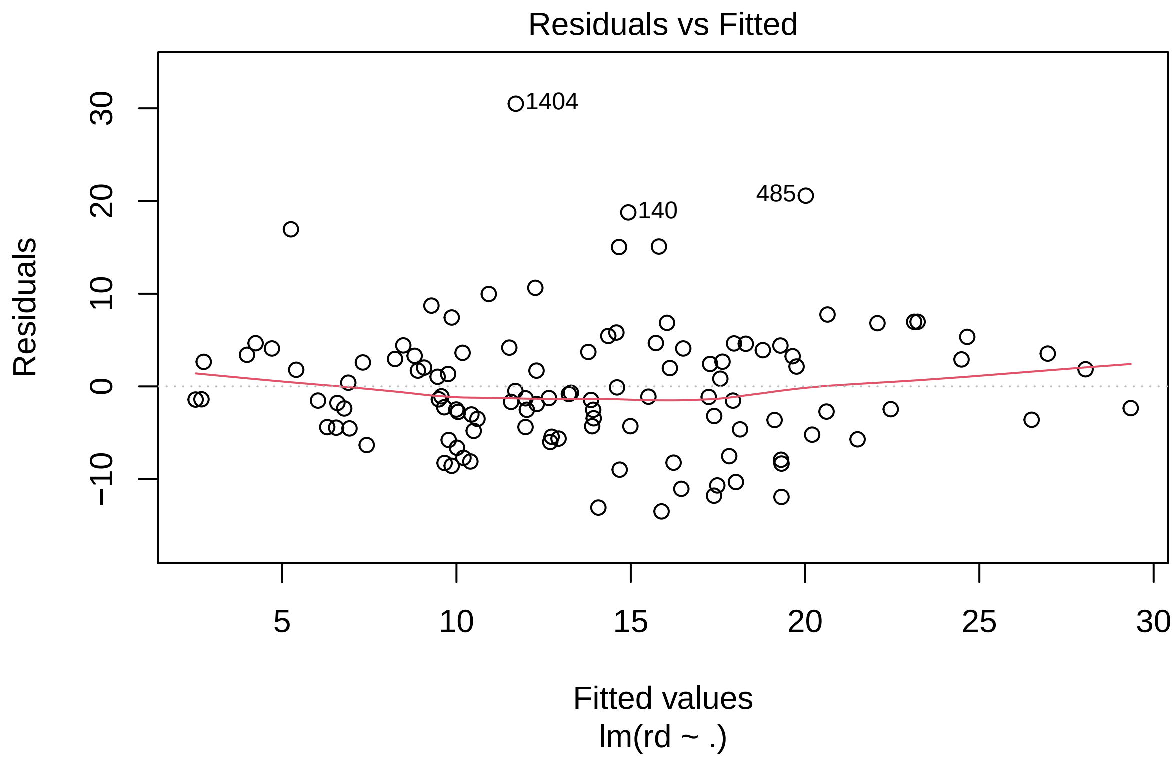
```
res <- residuals(model)


res <- as.data.frame(res)


ggplot(res,aes(res)) +  geom_histogram(fill='blue',alpha=0.5)
```

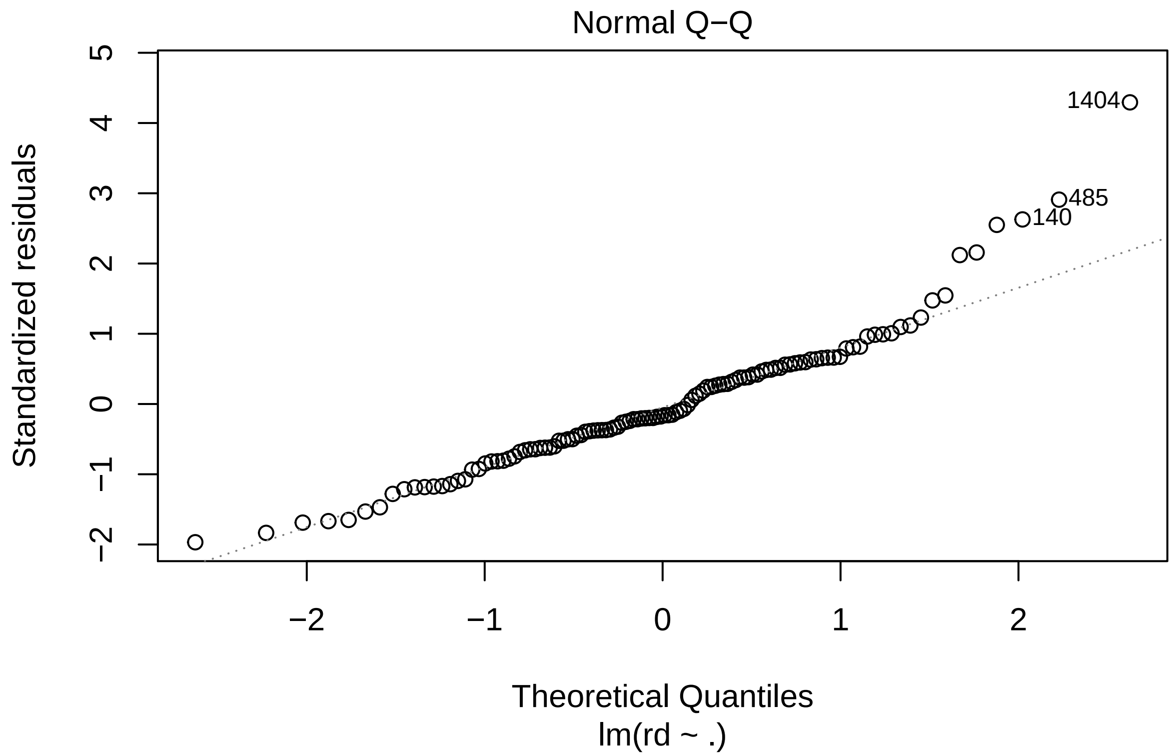
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

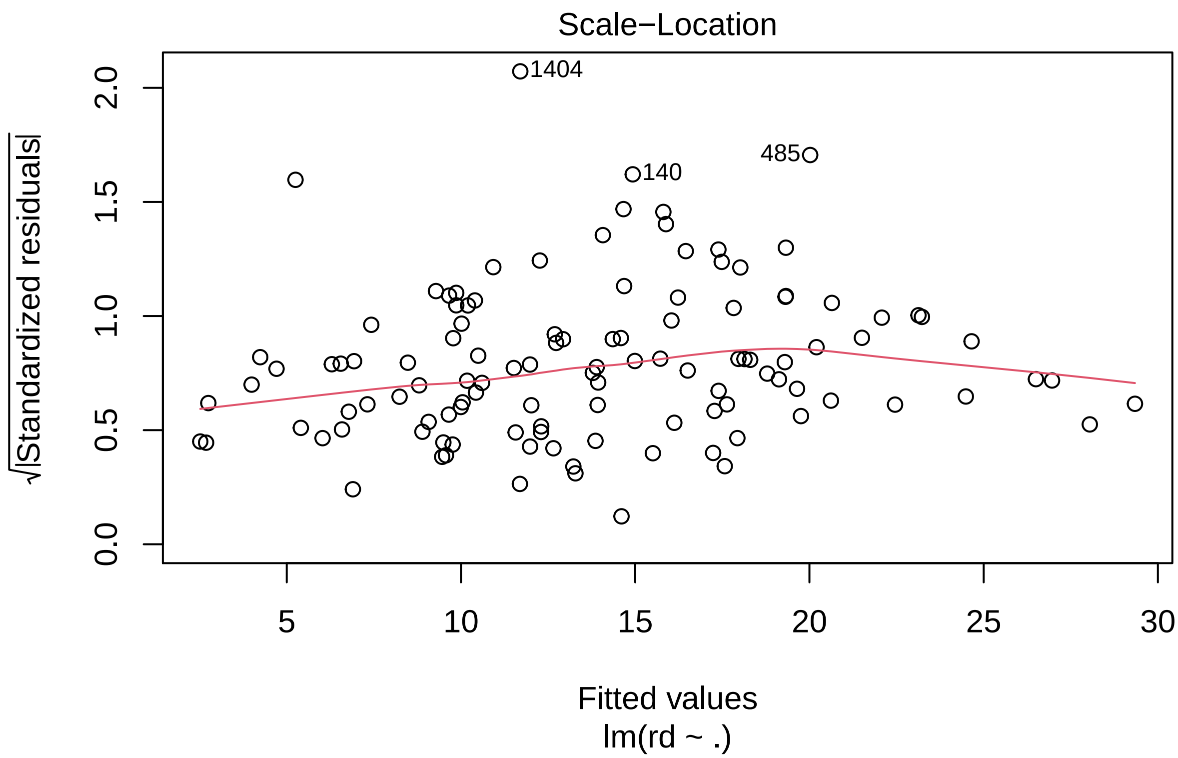


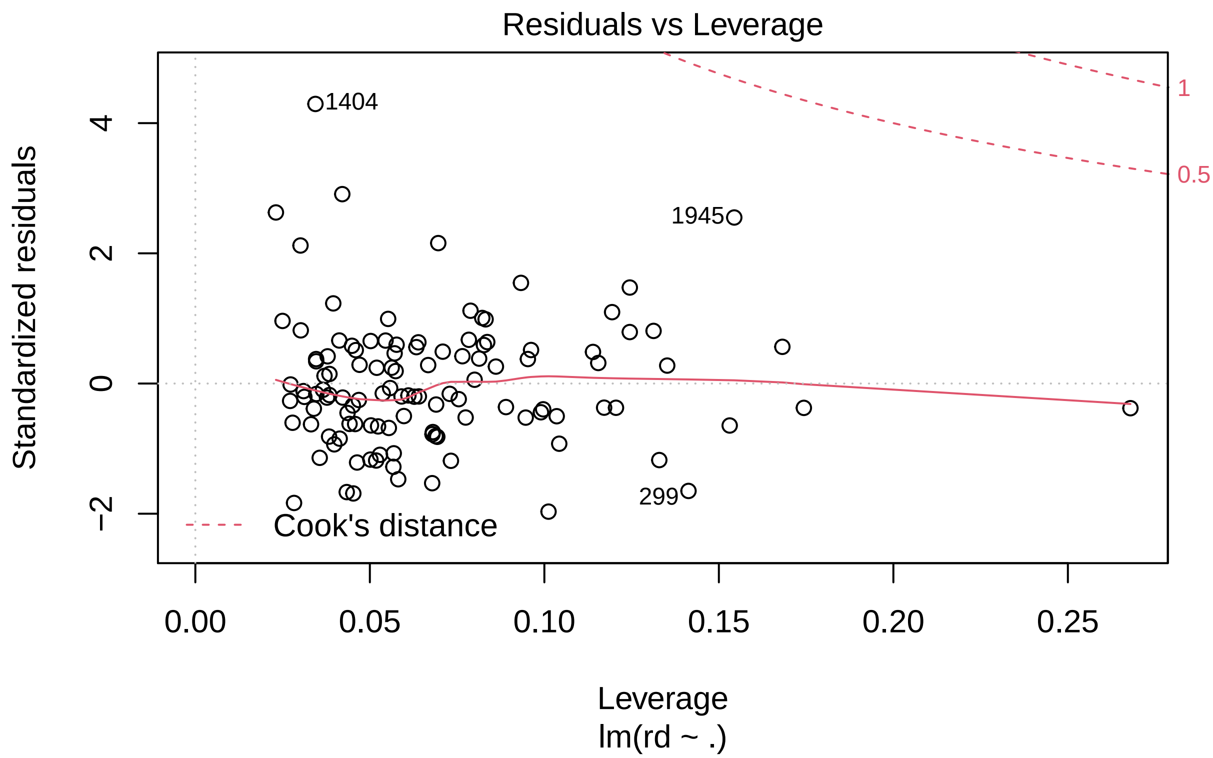
plots for residuals

```
plot(model)
```


Residuals vs Fitted
Residuals
30
20
10
0
−10
1404
140
485
5
10
15
20
25
30
Fitted values
lm(rd ~ .)


Normal Q−Q
Standardized residuals
5
4
3
2
1
0
−1
−2
1404
485
140
−2
−1
0
1
2
Theoretical Quantiles
lm(rd ~ .)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
1404
485
140
5
10
15
20
25
30
Fitted values
lm(rd ~ .)



#splitting data into train test datas

```
#
library(caTools)

set.seed(101)


sample <- sample.split(df_m$rd, SplitRatio = 0.85)

train = subset(subset(df_m, select = -c(country, code, web, d0wn)), sample == TRUE)

test = subset(subset(df_m, select =-c(country, code, web, d0wn)), sample == FALSE)
```

#training linear model

```
model <- lm(rd ~ .,train)
r.predictions <- predict(model,test)
```

#predicting besed on test datas

```
results <- cbind(r.predictions,test$rd)
colnames(results) <- c('pred','real')
results <- as.data.frame(results)
```

#dealing with zero values

```
to_zero <- function(x){
    if  (x < 0){
        return(0)
    }else{
        return(x)
    }
}
results$pred <- sapply(results$pred,to_zero)
```

#calculatinng mean squarted error

```
mse <- mean((results$real-results$pred)^2)
print(mse)
```

```
## [1] 47.05415
```

#calculatinng coefficient of determination

```
SSE = sum((results$pred - results$real)^2)
SST = sum( (mean(df_m$rd) - results$real)^2)

R2 = 1 - SSE/SST
R2
```

```
## [1] 0.4271447
```

#appling random forest model

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

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

```
model <- randomForest(rd ~ .,train,mportance = TRUE, na.action = na.omit) ##training  model
model
```

```
##
## Call:
##  randomForest(formula = rd ~ ., data = train, mportance = TRUE,      na.action = na.omit)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 57.57264
##                     % Var explained: 28.49
```

```
r.predictions <- predict(model,test)  #predicting besed on test datas
results <- cbind(r.predictions,test$rd)
colnames(results) <- c('pred','real')
results <- as.data.frame(results)
to_zero <- function(x){   #dealing with zero values
    if  (x < 0){
        return(0)
    }else{
        return(x)
    }
}
results$pred <- sapply(results$pred,to_zero)
mse <- mean((results$real-results$pred)^2)    #calculatinng mean squarted error
print(mse)
```

```
## [1] 53.51444
```

```
SSE = sum((results$pred - results$real)^2)   #calculatinng coefficient of determination
SST = sum( (mean(df_m$rd) - results$real)^2)

R2 = 1 - SSE/SST
R2
```

```
## [1] 0.3484946
```

#showing which variable is important based on random forest model.

```
importance(model)
```

```
##          IncNodePurity
## foreign      1352.6656
## certi         942.5003
## fOwn         1284.3216
## train        1668.3132
## bank          723.7881
## tax           508.6516
## informal      772.4876
```

#clustring based on explanatory variables (5 cluster)

```
km_5 <- kmeans( subset(df_m, select = -c(country, code, web, dOwn)), 5,nstart = 10)
```

#clustring based on explanatory variables (7 cluster)

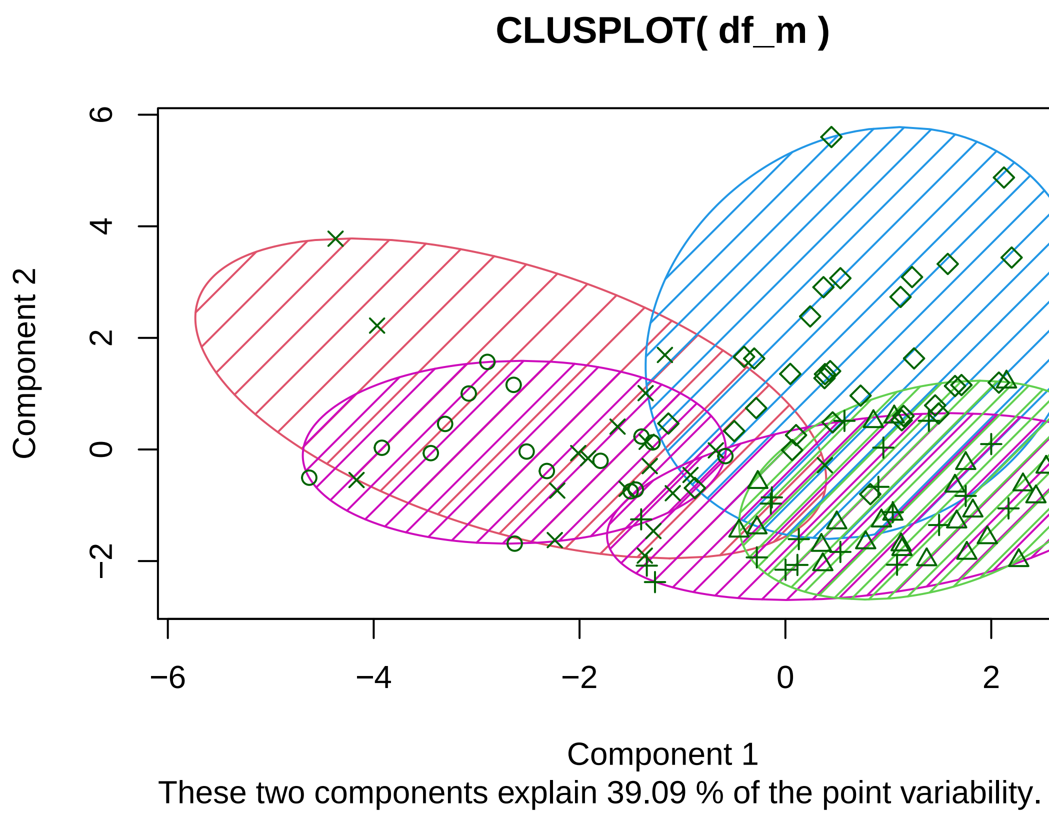
```
km_7 <- kmeans( subset(df_m, select = -c(country, code, web, dOwn)), 7, nstart = 10)
```

#adding predicted cluster to dataframe

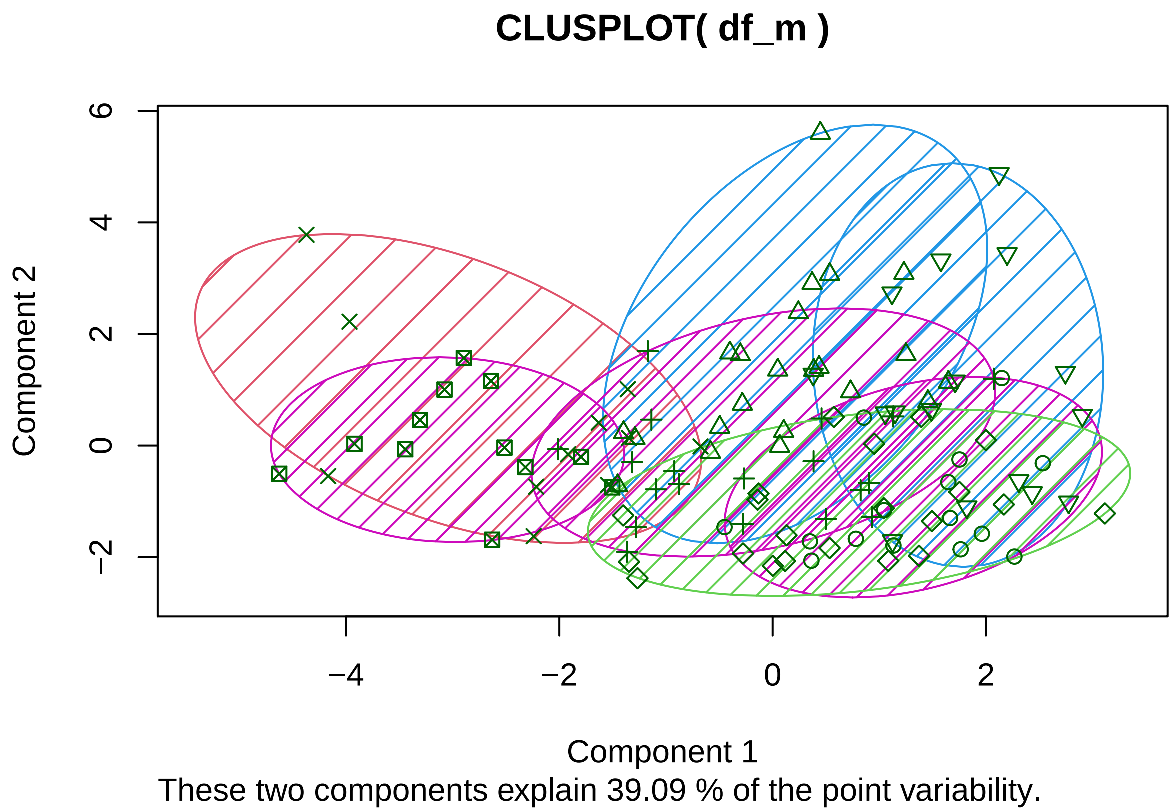
```
df_m["k5"] <- km_5$cluster
df_m["k7"] <- km_7$cluster
```

#visualization of clusters

```
library(cluster)
clusplot(df_m, df_m$k5, color=TRUE, shade=TRUE, labels=0,lines=0, )
```




```
clusplot(df_m, df_m$k7, color=TRUE, shade=TRUE, labels=0,lines=0, )
```




```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:reshape2':
##
##     smiths
```

```
library(grid)
library(rworldmap)
```

```
## Loading required package: sp

## ### Welcome to rworldmap ###

## For a short introduction type :   vignette('rworldmap')
```

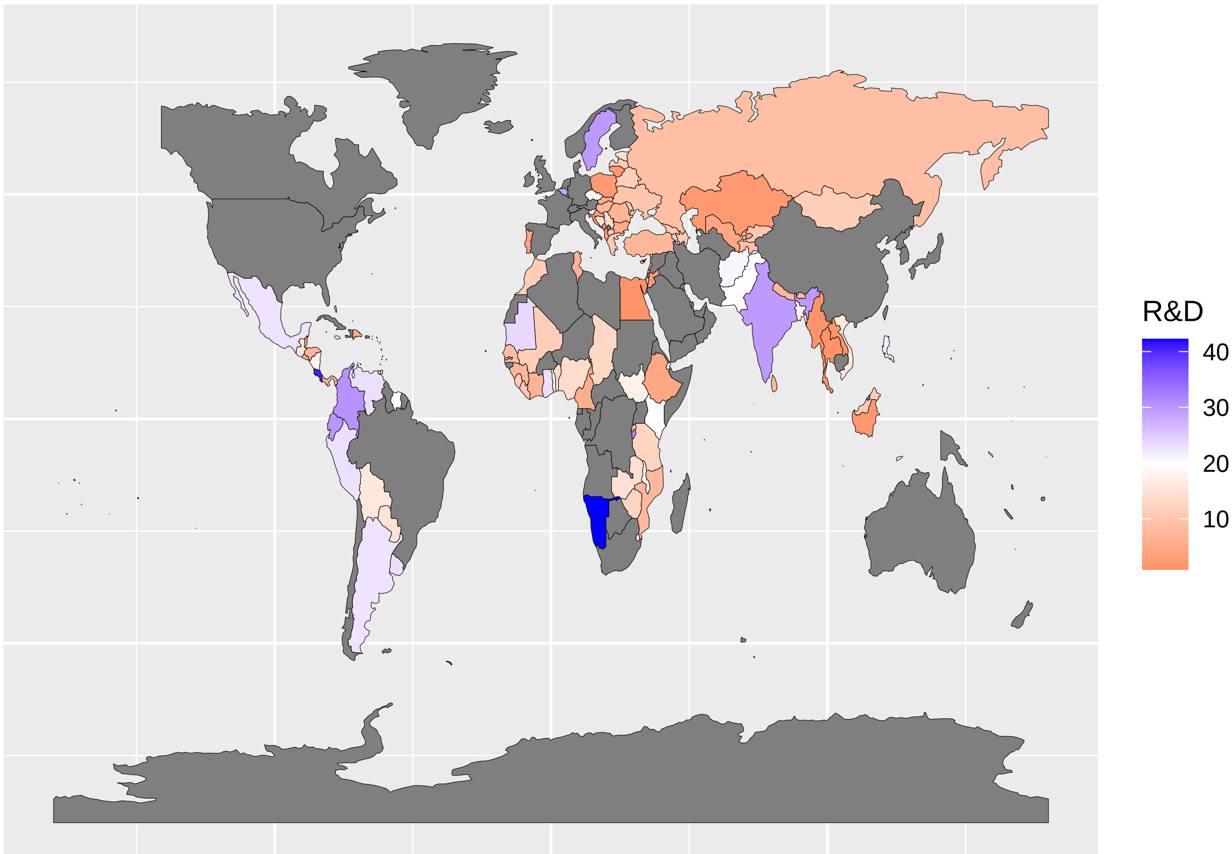
```
library(mapproj)
```

```
## Loading required package: maps

##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:cluster':
##
##     votes.repub

## The following object is masked from 'package:plyr':
##
##     ozone
```

#world map based on Percent of firms that spend on R&D variable

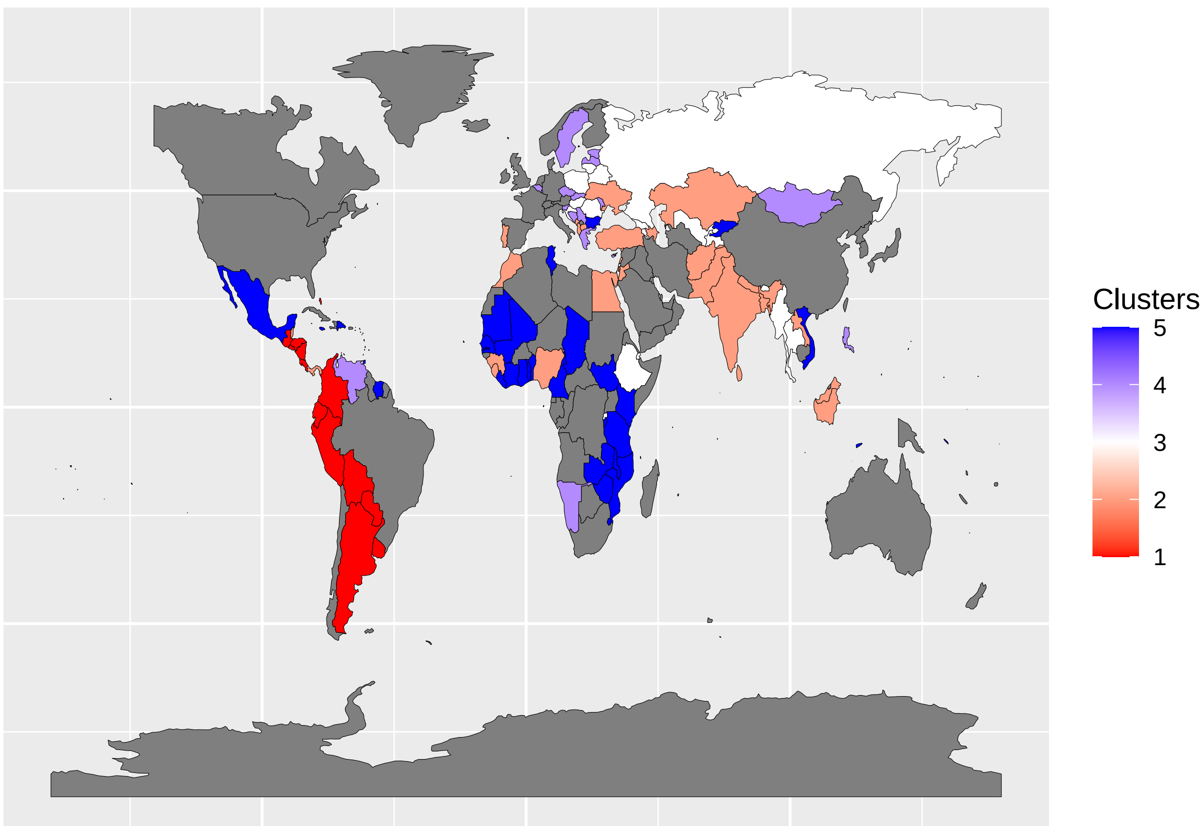
```
worldMap<-getMap() # worldmap laden

mf <- merge(df_m, as.data.frame(worldMap$ISO_A3),  by.x = 'code', by.y = "worldMap$ISO_A3", sort = TRUE
m <-  which(worldMap$ISO_A3%in%mf$code)

Coords <- lapply(m, function(i){
  f <- data.frame(worldMap@polygons[[i]]@Polygons[[1]]@coords)
  f$region =as.character(worldMap$ISO_A3[i])
  colnames(f) <- list("long", "lat", "region")
  return(f)
})

Coords <- do.call("rbind", Coords)

tw <- data.frame(country = mf$code, value = mf$rd)
Coords$value2014 <- tw$value[match(Coords$region,tw$country)]

mp <- ggplot() + geom_polygon(data = Coords, aes(x = long, y = lat, group = region, fill = value2014),
                             colour = "black", size = 0.1)
  #coord_map(xlim = c(-13, 35),  ylim = c(32, 71))


mp <- mp + scale_fill_gradient2(name = "R&D", low = "coral", mid="white", high = "blue", midpoint=20, sp


mp <- mp + theme(#panel.grid.minor = element_line(colour = NA), panel.grid.minor = element_line(colour
               #panel.background = element_rect(fill = NA, colour = NA),
               axis.text.x = element_blank(),
               axis.text.y = element_blank(), axis.ticks.x = element_blank(),
               axis.ticks.y = element_blank(), axis.title = element_blank(),
               #rect = element_blank(),
               plot.margin = unit(0 * c(-1.5, -1.5, -1.5, -1.5), "lines"))
mp
```



#world map based on clusters

```r
worldMap<-getMap() # worldmap laden

mf <- merge(df_m, as.data.frame(worldMap$ISO_A3),  by.x = 'code', by.y = "worldMap$ISO_A3", sort = TRUE
m <-  which(worldMap$ISO_A3%in%mf$code)

Coords <- lapply(m, function(i){
  f <- data.frame(worldMap@polygons[[i]]@Polygons[[1]]@coords)
  f$region =as.character(worldMap$ISO_A3[i])
  colnames(f) <- list("long", "lat", "region")
  return(f)
})

Coords <- do.call("rbind", Coords)

tw <- data.frame(country = mf$code, value = mf$k5)
Coords$value <- tw$value[match(Coords$region,tw$country)]

mp <- ggplot() + geom_polygon(data = Coords, aes(x = long, y = lat, group = region, fill = value),
                              colour = "black", size = 0.1)
  #coord_map(xlim = c(-13, 35),  ylim = c(32, 71))


mp <- mp + scale_fill_gradient2(name = "Clusters", low = "red", mid="white", high = "blue", space="Lab"
```

```
mp <- mp + theme(#panel.grid.minor = element_line(colour = NA), panel.grid.minor = element_line(colour
              #panel.background = element_rect(fill = NA, colour = NA),
              axis.text.x = element_blank(),
              axis.text.y = element_blank(), axis.ticks.x = element_blank(),
              axis.ticks.y = element_blank(), axis.title = element_blank(),
              #rect = element_blank(),
              #plot.margin = unit(0 * c(-1.5, -1.5, -1.5, -1.5), "lines")
              )
mp
```



#making a new dataframe for neural network

```
df_m <- na.omit(df_m)
df_n <- subset(df_m, select = -c(country, code, k5,k7, dOwn, web))
```

#making formula based on column names for the model

```
n <- names(df_n)
 as.formula(paste("rd ~", paste(n[!n %in% "rd"], collapse = " + ")))
```

```
## rd ~ foreign + certi + fOwn + train + bank + tax + informal
```

#finding max and mins for each cloumn

```
maxs <- apply(df_n, 2, max)
mins <- apply(df_n, 2, min)
```

#scaling columns

```
scaled <- as.data.frame(scale(df_n, center = mins, scale = maxs - mins))
```

#splitting dataset

```
split = sample.split(scaled$rd, SplitRatio = 0.90)

train = subset(scaled, split == TRUE)
test = subset(scaled, split == FALSE)
```

#training model

```
library(neuralnet)
```

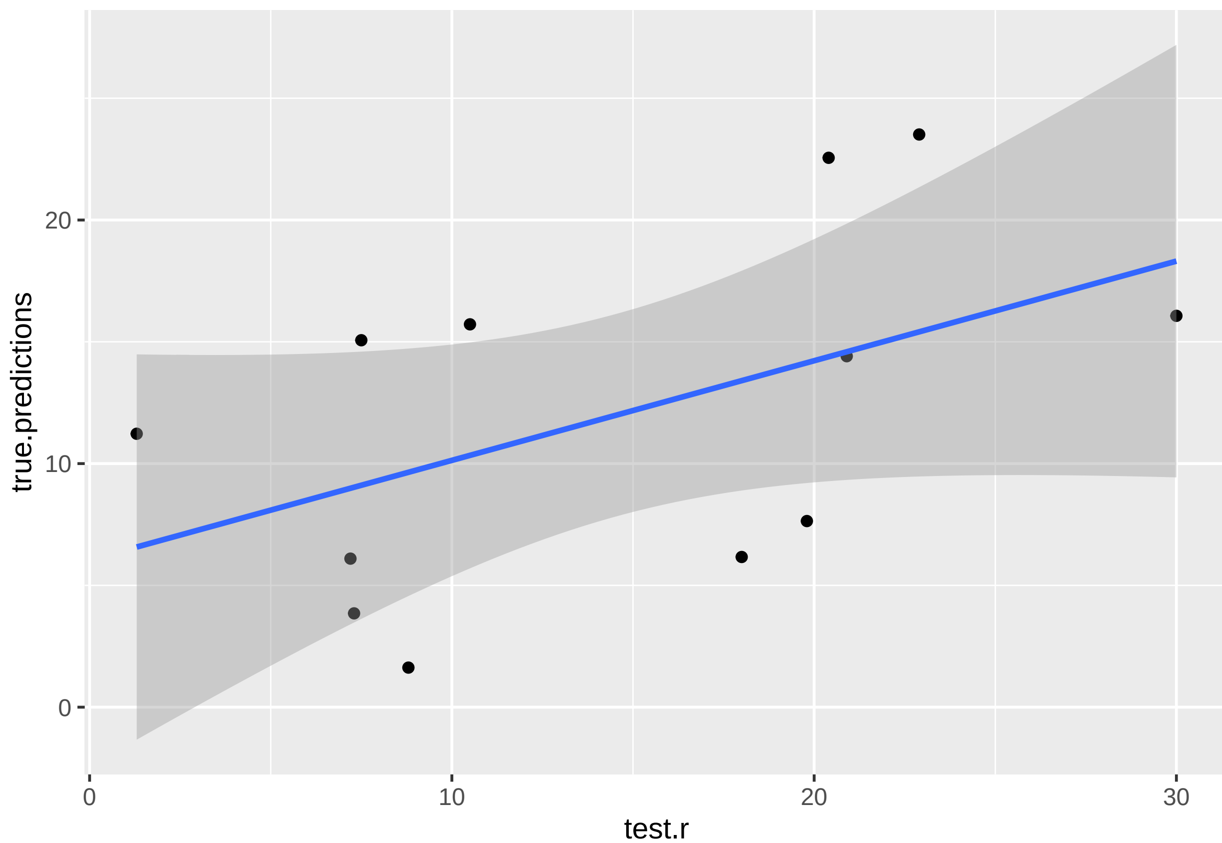
```
##
## Attaching package: 'neuralnet'

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

```
n <- names(train)
f <- as.formula(paste("rd ~", paste(n[!n %in% "rd"], collapse = " + ")))
model <- neuralnet(f,data=train,hidden=c(5,3),linear.output=TRUE)
model <- neuralnet(f,data=train,hidden=c(5,3),linear.output=TRUE)
```

#predicting and plotting predicted and real values

```
predicted.nn.values <- compute(model, test[1:(length(test)-1)])
true.predictions <- predicted.nn.values$net.result*(max(df_n$rd)-min(df_n$rd))+min(df_n$rd)
test.r <- (test$rd)*(max(df_n$rd)-min(df_n$rd))+min(df_n$rd)
error.df <- data.frame(test.r,true.predictions)
ggplot(error.df,aes(x=test.r,y=true.predictions)) + geom_point() + stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



20
10
0
true.predictions
0
10
20
30
test.r