INFO-F-422 - Statistical foundations of machine learning

Project 2020 - 2021

Members:

Czajka Léo - UCL : 11231701

Hoang Hai Nam - ULB : 443146

Murisier Frederic - ULB: 442283

frederic.murisier@ulb.be

hoang.hai.nam@ulb.be

leo.czajka@gmail.com

DrivenData team name: ULB_CHM_Machine_Learners

DrivenData average score: 0.813

Presentation Link: https://streamable.com/5dgk4h

Warning:

This notebook was validated with R on version : 4.0.5

As such some cells may not work as libraries used might've changed between R version 3 and 4. Please send us emails if you have trouble running cells or with the package.

Cells that take a long time to run was marked with a warning before running them. All cells without warnings may take anywhere between 1s and 2 minutes to run.

1. Data Pre-processing/Feature selection

First we import the data, name our main (train) dataset *df*,

In [1]:

```
train_values <- read.csv("trainingsetvalues.csv")
train_labels <- read.csv("trainingsetlabels.csv")
df <- merge(train_values, train_labels)
test_values <- read.csv("testsetvalues.csv")
```

Then we load each package needed for this part

In [2]:

```
library(Hmisc)
library(tidyverse)
library(dplyr)
```

```
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:base':

    format.pval, units


-- Attaching packages --------------------------------------- tidyverse 1.3.1 --

v tibble  3.1.1     v dplyr   1.0.6
v tidyr   1.1.3     v stringr 1.4.0
v readr   1.4.0     v forcats 0.5.1
v purrr   0.3.4

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter()    masks stats::filter()
x dplyr::lag()       masks stats::lag()
x dplyr::src()       masks Hmisc::src()
x dplyr::summarize() masks Hmisc::summarize()
```

Let's have a general look at all features. This will allow us to quickly classify the different variables of our data set, given their type and structure.

In [3]:

```
describe(df)
```

```
df

 41  Variables      59400  Observations
--------------------------------------------------------------------------------
id
       n  missing distinct     Info     Mean      Gmd      .05      .10
   59400        0    59400        1    37115    24772     3731     7405
     .25      .50      .75      .90      .95
   18520    37062    55657    66863    70564

lowest :     0     1     2     3     4, highest: 74240 74242 74243 74246 74247
--------------------------------------------------------------------------------
amount_tsh
       n  missing distinct     Info     Mean      Gmd      .05      .10
   59400        0       98    0.655    317.7    594.7        0        0
     .25      .50      .75      .90      .95
       0        0       20      500     1200

lowest : 0.00e+00 2.00e-01 2.50e-01 1.00e+00 2.00e+00
highest: 1.38e+05 1.70e+05 2.00e+05 2.50e+05 3.50e+05
--------------------------------------------------------------------------------
date_recorded
       n  missing distinct
   59400        0      356

lowest : 2002-10-14 2004-01-07 2004-03-01 2004-03-06 2004-04-01
highest: 2013-11-02 2013-11-03 2013-12-01 2013-12-02 2013-12-03
--------------------------------------------------------------------------------
funder
```

```
funder
       n  missing distinct 
   55765     3635     1897 

lowest : 0                  A/co Germany        Aar                 Abas Ka            
Abasia            
highest: Zao                Zao Water Spring    Zao Water Spring X  Zinduka            
Zingibali Secondary
--------------------------------------------------------------------------------
gps_height 
       n  missing distinct     Info     Mean      Gmd      .05      .10 
   59400        0     2428    0.959    668.3    761.3        0        0 
     .25      .50      .75      .90      .95 
       0      369     1319     1638     1797 

lowest :  -90  -63  -59  -57  -55, highest: 2623 2626 2627 2628 2770
--------------------------------------------------------------------------------
installer 
       n  missing distinct 
   55745     3655     2145 

lowest : -                  0                   A.D.B               AAR                
Aartisa           
highest: Zao water spring    Zao water spring X  ZINDUKA             Zingibali Secondary
Zuber Mihungo     
--------------------------------------------------------------------------------
longitude 
       n  missing distinct     Info     Mean      Gmd      .05      .10 
   59400        0    57516        1    34.08    4.894    30.04    30.95 
     .25      .50      .75      .90      .95 
   33.09    34.91    37.18    38.78    39.13 

lowest :  0.00000 29.60712 29.60720 29.61032 29.61096
highest: 40.32340 40.32523 40.32524 40.34430 40.34519
--------------------------------------------------------------------------------
latitude 
       n  missing distinct     Info     Mean      Gmd      .05      .10 
   59400        0    57517        1   -5.706    3.371  -10.586   -9.713 
     .25      .50      .75      .90      .95 
  -8.541   -5.022   -3.326   -2.101   -1.409 

lowest : -11.64944018 -11.64837759 -11.58629656 -11.56857679 -11.56680457
highest:  -0.99911702  -0.99901209  -0.99891600  -0.99846435  -0.00000002
--------------------------------------------------------------------------------
wpt_name 
       n  missing distinct 
   59400        0    37400 

lowest : 24                A Kulwa           A Saidi           Abass             Abbas            

highest: Zumbawanu Shuleni Zungu             Zunguni           Zunzuli A Shuleni Zuwena K
indo     
--------------------------------------------------------------------------------
num_private 
       n  missing distinct     Info     Mean      Gmd      .05      .10 
   59400        0       65    0.038   0.4741   0.9443        0        0 
     .25      .50      .75      .90      .95 
       0        0        0        0        0 

lowest :    0    1    2    3    4, highest:  672  698  755 1402 1776
--------------------------------------------------------------------------------
basin 
       n  missing distinct 
   59400        0        9 

lowest : Internal                Lake Nyasa              Lake Rukwa              Lake Ta
nganyika         Lake Victoria          
highest: Lake Victoria           Pangani                 Rufiji                  Ruvuma 
/ Southern Coast Wami / Ruvu            

Internal (7785, 0.131), Lake Nyasa (5085, 0.086), Lake Rukwa (2454, 0.041),
Lake Tanganyika (6432, 0.108), Lake Victoria (10248, 0.173), Pangani (8940,
0.151), Rufiji (7976, 0.134), Ruvuma / Southern Coast (4493, 0.076), Wami /
```

```
0.131), Rufiji (7976, 0.134), Ruvuma / Southern Coast (4493, 0.076), Wami /
Ruvu (5987, 0.101)
--------------------------------------------------------------------------------
subvillage
       n  missing distinct
   59029      371    19287

lowest : 'A' Kati          ##                1                 14Kambalage       18

highest: Zumbawanu Shuleni Zunga             Zunguni           Zunzuli           Zuri

--------------------------------------------------------------------------------
region
       n  missing distinct
   59400        0       21

lowest : Arusha        Dar es Salaam Dodoma        Iringa        Kagera
highest: Ruvuma        Shinyanga     Singida       Tabora        Tanga
--------------------------------------------------------------------------------
region_code
       n  missing distinct     Info     Mean      Gmd      .05      .10
   59400        0       27    0.997     15.3    14.01        2        3
     .25      .50      .75      .90      .95
       5       12       17       20       60

lowest :  1  2  3  4  5, highest: 40 60 80 90 99
--------------------------------------------------------------------------------
district_code
       n  missing distinct     Info     Mean      Gmd      .05      .10
   59400        0       20    0.976     5.63    6.356        1        1
     .25      .50      .75      .90      .95
       2        3        5        7       30

lowest :  0  1  2  3  4, highest: 60 62 63 67 80

Value          0     1     2     3     4     5     6     7     8    13    23
Frequency     23 12203 11173  9998  8999  4356  4074  3343  1043   391   293
Proportion 0.000 0.205 0.188 0.168 0.151 0.073 0.069 0.056 0.018 0.007 0.005

Value         30    33    43    53    60    62    63    67    80
Frequency    995   874   505   745    63   109   195     6    12
Proportion 0.017 0.015 0.009 0.013 0.001 0.002 0.003 0.000 0.000
--------------------------------------------------------------------------------
lga
       n  missing distinct
   59400        0      125

lowest : Arusha Rural Arusha Urban Babati       Bagamoyo     Bahi
highest: Tunduru      Ukerewe      Ulanga       Urambo       Uyui
--------------------------------------------------------------------------------
ward
       n  missing distinct
   59400        0     2092

lowest : Aghondi      Akheri       Arash        Arri         Arusha Chini
highest: Ziwani       Zoissa       Zombo        Zongomera    Zuzu
--------------------------------------------------------------------------------
population
       n  missing distinct     Info     Mean      Gmd      .05      .10
   59400        0     1049    0.952    179.9    276.5        0        0
     .25      .50      .75      .90      .95
       0       25      215      453      680

lowest :     0     1     2     3     4, highest:  9865 10000 11463 15300 30500
--------------------------------------------------------------------------------
public_meeting
       n  missing distinct
   56066     3334        2

Value      False  True
Frequency   5055 51011
Proportion  0.09  0.91
--------------------------------------------------------------------------------
```

```
recorded_by
                    n                 missing                distinct
                59400                       0                       1
                value
GeoData Consultants Ltd

Value      GeoData Consultants Ltd
Frequency                    59400
Proportion                       1
--------------------------------------------------------------------------------
scheme_management
       n  missing distinct
   55523     3877       12

lowest : Company          None             Other            Parastatal       Private ope
rator
highest: VWC              Water authority  Water Board      WUA              WUG


Company (1061, 0.019), None (1, 0.000), Other (766, 0.014), Parastatal (1680,
0.030), Private operator (1063, 0.019), SWC (97, 0.002), Trust (72, 0.001), VWC
(36793, 0.663), Water authority (3153, 0.057), Water Board (2748, 0.049), WUA
(2883, 0.052), WUG (5206, 0.094)
--------------------------------------------------------------------------------
scheme_name
       n  missing distinct
   31234    28166     2696

lowest : 14 Kambarage            A                        ADP                      ADP Sim
bo               ADP Simbu
highest: Ziwani juu water supply Ziwani water supply     Zo                       Zois
Zuzu
--------------------------------------------------------------------------------
permit
       n  missing distinct
   56344     3056        2

Value      False  True
Frequency  17492 38852
Proportion  0.31  0.69
--------------------------------------------------------------------------------
construction_year
       n  missing distinct     Info     Mean      Gmd      .05      .10
   59400        0       55    0.957     1301    912.8        0        0
     .25      .50      .75      .90      .95
       0     1986     2004     2009     2010

lowest :    0 1960 1961 1962 1963, highest: 2009 2010 2011 2012 2013

Value          0  1960  1965  1970  1975  1980  1985  1990  1995  2000  2005
Frequency  20709   153   249  1400  1913  3022  2948  2755  3815  5651  6478
Proportion 0.349 0.003 0.004 0.024 0.032 0.051 0.050 0.046 0.064 0.095 0.109

Value       2010  2015
Frequency  10131   176
Proportion 0.171 0.003

For the frequency table, variable is rounded to the nearest 5
--------------------------------------------------------------------------------
extraction_type
       n  missing distinct
   59400        0       18

lowest : afridev        cemo           climax         gravity        india mark ii
highest: other - swn 81 submersible    swn 80         walimi         windmill

afridev (1770, 0.030), cemo (90, 0.002), climax (32, 0.001), gravity (26780,
0.451), india mark ii (2400, 0.040), india mark iii (98, 0.002), ksb (1415,
0.024), mono (2865, 0.048), nira/tanira (8154, 0.137), other (6430, 0.108),
other - mkulima/shinyanga (2, 0.000), other - play pump (85, 0.001), other -
rope pump (451, 0.008), other - swn 81 (229, 0.004), submersible (4764, 0.080),
swn 80 (3670, 0.062), walimi (48, 0.001), windmill (117, 0.002)
```

```
swn 80 (3670, 0.062), walimi (48, 0.001), windmill (117, 0.002)
--------------------------------------------------------------------------------
extraction_type_group
       n  missing distinct
   59400        0       13

lowest : afridev         gravity         india mark ii   india mark iii  mono
highest: other motorpump rope pump       submersible     swn 80          wind-powered

afridev (1770, 0.030), gravity (26780, 0.451), india mark ii (2400, 0.040),
india mark iii (98, 0.002), mono (2865, 0.048), nira/tanira (8154, 0.137),
other (6430, 0.108), other handpump (364, 0.006), other motorpump (122, 0.002),
rope pump (451, 0.008), submersible (6179, 0.104), swn 80 (3670, 0.062),
wind-powered (117, 0.002)
--------------------------------------------------------------------------------
extraction_type_class
       n  missing distinct
   59400        0        7

lowest : gravity      handpump     motorpump    other        rope pump
highest: motorpump    other        rope pump    submersible  wind-powered

Value            gravity     handpump    motorpump        other    rope pump
Frequency          26780        16456         2987         6430          451
Proportion         0.451        0.277        0.050        0.108        0.008

Value        submersible wind-powered
Frequency           6179          117
Proportion         0.104        0.002
--------------------------------------------------------------------------------
management
       n  missing distinct
   59400        0       12

lowest : company          other            other - school   parastatal       private ope
rator
highest: vwc              water authority  water board      wua              wug


company (685, 0.012), other (844, 0.014), other - school (99, 0.002),
parastatal (1768, 0.030), private operator (1971, 0.033), trust (78, 0.001),
unknown (561, 0.009), vwc (40507, 0.682), water authority (904, 0.015), water
board (2933, 0.049), wua (2535, 0.043), wug (6515, 0.110)
--------------------------------------------------------------------------------
management_group
       n  missing distinct
   59400        0        5

lowest : commercial other      parastatal unknown    user-group
highest: commercial other      parastatal unknown    user-group

Value      commercial      other parastatal    unknown user-group
Frequency        3638        943       1768        561      52490
Proportion      0.061      0.016      0.030      0.009      0.884
--------------------------------------------------------------------------------
payment
       n  missing distinct
   59400        0        7

lowest : never pay             other                 pay annually          pay monthly
pay per bucket
highest: pay annually          pay monthly           pay per bucket        pay when schem
e fails unknown

never pay (25348, 0.427), other (1054, 0.018), pay annually (3642, 0.061), pay
monthly (8300, 0.140), pay per bucket (8985, 0.151), pay when scheme fails
(3914, 0.066), unknown (8157, 0.137)
--------------------------------------------------------------------------------
payment_type
       n  missing distinct
   59400        0        7

lowest : annually   monthly    never pay  on failure other
```

```
lowest : annually   monthly    never pay  on failure other
highest: never pay  on failure other      per bucket unknown

Value        annually    monthly  never pay on failure      other per bucket
Frequency        3642       8300      25348       3914       1054       8985
Proportion      0.061      0.140      0.427      0.066      0.018      0.151

Value         unknown
Frequency        8157
Proportion      0.137
--------------------------------------------------------------------------------
water_quality
       n  missing distinct
   59400        0        8

lowest : coloured           fluoride           fluoride abandoned milky              sal
ty
highest: milky              salty              salty abandoned    soft               unk
nown

coloured (490, 0.008), fluoride (200, 0.003), fluoride abandoned (17, 0.000),
milky (804, 0.014), salty (4856, 0.082), salty abandoned (339, 0.006), soft
(50818, 0.856), unknown (1876, 0.032)
--------------------------------------------------------------------------------
quality_group
       n  missing distinct
   59400        0        6

lowest : colored  fluoride good     milky    salty
highest: fluoride good     milky    salty    unknown

Value       colored fluoride     good    milky    salty  unknown
Frequency       490      217    50818      804     5195     1876
Proportion    0.008    0.004    0.856    0.014    0.087    0.032
--------------------------------------------------------------------------------
quantity
       n  missing distinct
   59400        0        5

lowest : dry          enough       insufficient seasonal     unknown
highest: dry          enough       insufficient seasonal     unknown

Value                 dry       enough insufficient     seasonal      unknown
Frequency            6246        33186        15129         4050          789
Proportion          0.105        0.559        0.255        0.068        0.013
--------------------------------------------------------------------------------
quantity_group
       n  missing distinct
   59400        0        5

lowest : dry          enough       insufficient seasonal     unknown
highest: dry          enough       insufficient seasonal     unknown

Value                 dry       enough insufficient     seasonal      unknown
Frequency            6246        33186        15129         4050          789
Proportion          0.105        0.559        0.255        0.068        0.013
--------------------------------------------------------------------------------
source
       n  missing distinct
   59400        0       10

lowest : dam                  hand dtw             lake                 machine dbh
other
highest: rainwater harvesting river                shallow well         spring
unknown

dam (656, 0.011), hand dtw (874, 0.015), lake (765, 0.013), machine dbh (11075,
0.186), other (212, 0.004), rainwater harvesting (2295, 0.039), river (9612,
0.162), shallow well (16824, 0.283), spring (17021, 0.287), unknown (66, 0.001)
--------------------------------------------------------------------------------
source_type
       n  missing distinct
   59400        0        7
```

```
   59400         0         7

lowest : borehole             dam                  other                rainwater harves
ting river/lake
highest: other                rainwater harvesting river/lake           shallow well
spring

borehole (11949, 0.201), dam (656, 0.011), other (278, 0.005), rainwater
harvesting (2295, 0.039), river/lake (10377, 0.175), shallow well (16824,
0.283), spring (17021, 0.287)
--------------------------------------------------------------------------------
source_class
       n  missing distinct
   59400        0        3

Value      groundwater     surface     unknown
Frequency        45794       13328         278
Proportion       0.771       0.224       0.005
--------------------------------------------------------------------------------
waterpoint_type
       n  missing distinct
   59400        0        7

lowest : cattle trough               communal standpipe          communal standpipe mult
iple dam                        hand pump
highest: communal standpipe multiple dam                         hand pump
improved spring             other

cattle trough (116, 0.002), communal standpipe (28522, 0.480), communal
standpipe multiple (6103, 0.103), dam (7, 0.000), hand pump (17488, 0.294),
improved spring (784, 0.013), other (6380, 0.107)
--------------------------------------------------------------------------------
waterpoint_type_group
       n  missing distinct
   59400        0        6

lowest : cattle trough      communal standpipe dam                 hand pump           imp
roved spring
highest: communal standpipe dam                 hand pump           improved spring    oth
er

Value           cattle trough communal standpipe                dam
Frequency                 116              34625                  7
Proportion              0.002              0.583              0.000

Value               hand pump    improved spring              other
Frequency               17488                784               6380
Proportion              0.294              0.013              0.107
--------------------------------------------------------------------------------
status_group
       n  missing distinct
   59400        0        3

Value                   functional functional needs repair
Frequency                    32259                    4317
Proportion                   0.543                   0.073

Value               non functional
Frequency                    22824
Proportion                   0.384
--------------------------------------------------------------------------------
```

We remove three variables : *num_private* is almost always equal to 0 and there is no documentation about it; *recorded_by* is always equal to the same value, and *wpt_name* is the name of the water point therefore it is different for each water point and as such should not be useful to make any prediction about water points outside our data train set.

In [4]:

```
describe(df$num_private)
describe(df$recorded_by)
```

```
df<- df[, !(colnames(df) %in% c("recorded_by", "num_private", "wpt_name"))]
```

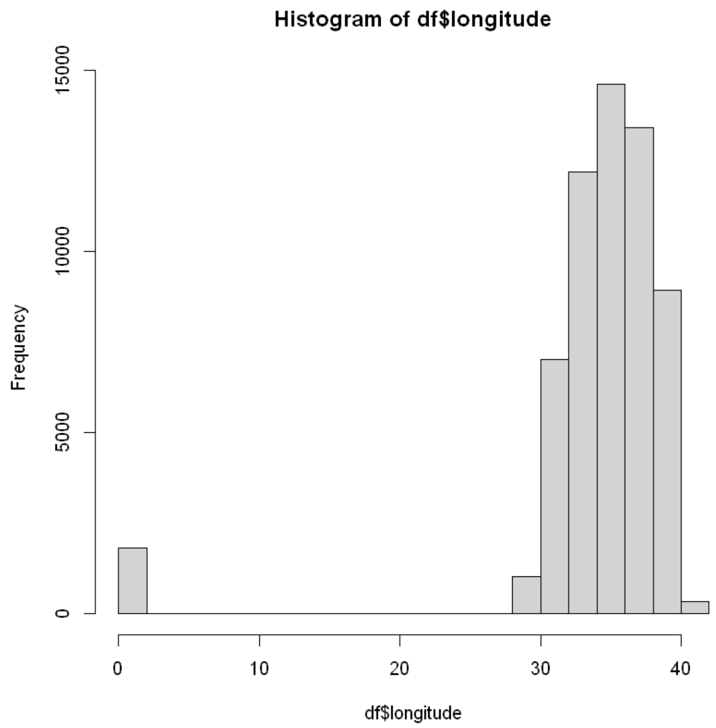
```
df$num_private
       n  missing distinct     Info     Mean      Gmd      .05      .10
   59400        0       65    0.038   0.4741   0.9443        0        0
     .25      .50      .75      .90      .95
       0        0        0        0        0

lowest :    0    1    2    3    4, highest:  672  698  755 1402 1776

df$recorded_by
                      n                 missing                distinct
                  59400                       0                       1
                  value
GeoData Consultants Ltd

Value      GeoData Consultants Ltd
Frequency                    59400
Proportion                       1
```

There are five strictly (we consider dates separately) numerical variables: *latitude*, *longitude*, *gps_height*, *population* and *amount_tsh*. All are relevant to the prediction problem we try to solve here. A closer inspection of the *longitude* reveals that some observations have value 0. Given the geographical situation of Tanzania this is not possible. A thorough correction of the data set would imply infering the *longitude* from categorical geographical variables available in the data set. However this would take time and, as the number of observations for which this occurs is small (3%), we simply decide to remove those observations from the data set. By contrast this is not an issue for the *latitude* which seems to be relatively well distributed. About 1/3 of the *gps_height* on the other hand is equal to 0. This value is of course possible, but it's frequency, together with the fact that zero probably correspond to missing in the longitude's variable, may suggest that, as such, *gps_height* is a relatively badly measured variable.

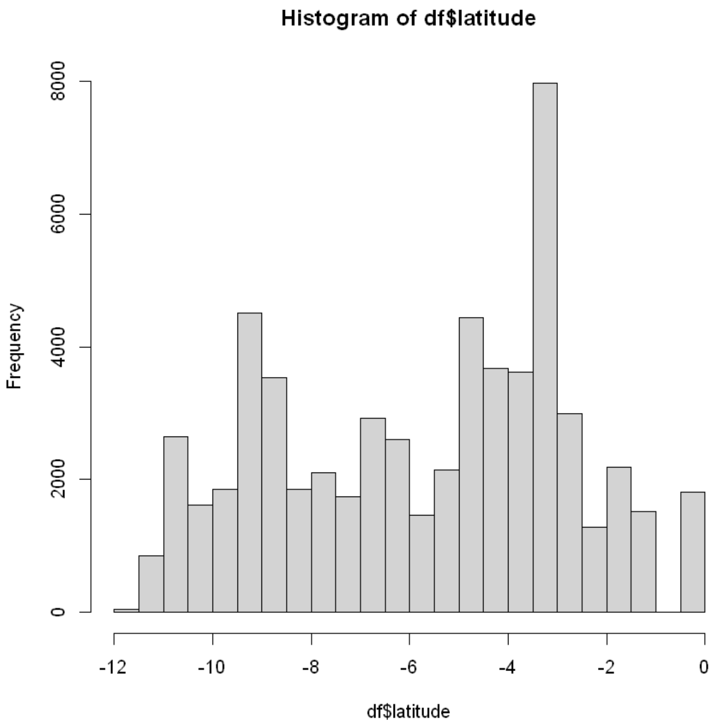
In [5]:

```
hist(df$longitude)
sum(df$longitude==0)/nrow(df)
hist(df$latitude)
sum(df$latitude==0)
hist(df$gps_height)
sum(df$gps_height==0)
# df <- df %>% filter(df$longitude!=0)
```

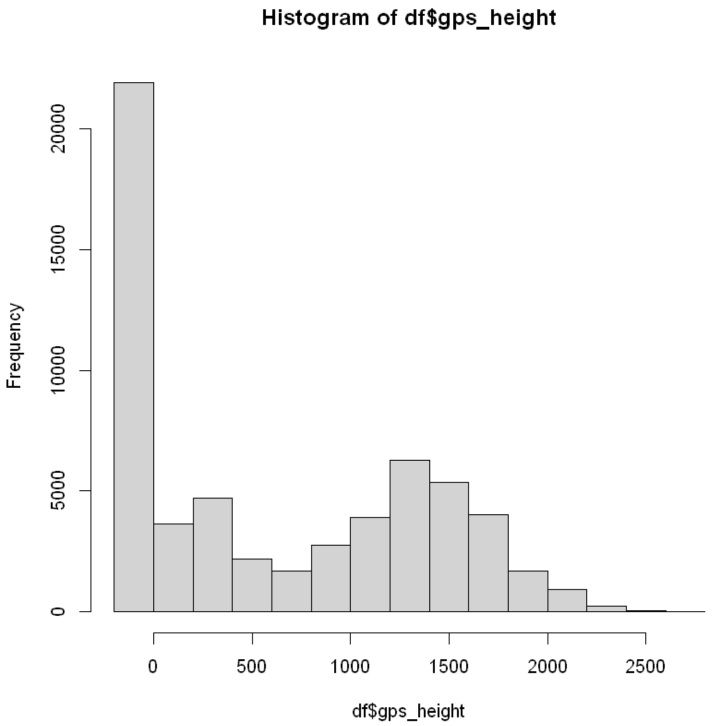
0.0305050505050505



0



20438



The variables *population* and *amount_tsh* also seem to have been badly measured. 36% of observations have a population equal to 0, and an additional 12% have a population equal to 1. Both values could be correct. But water pumps should not be too far from people benefiting from it, and as such we would have expected such values to be less frequent. Also, the relative weight of these values is in stark contrast with the rest of the distribution of the *population* variable. About 70% of the *amount_tsh* variable is equal to 0, meaning that there was no water available to waterpoint. This figure is puzzling given that more than half of the pump are qualified as functional, we would expect indeed that some water needs to be available at the water point for the pump to be tried and ultimately assessed as functional. A closer inspection of the distribution of non-zero values further illustrate how irregular this variable is.

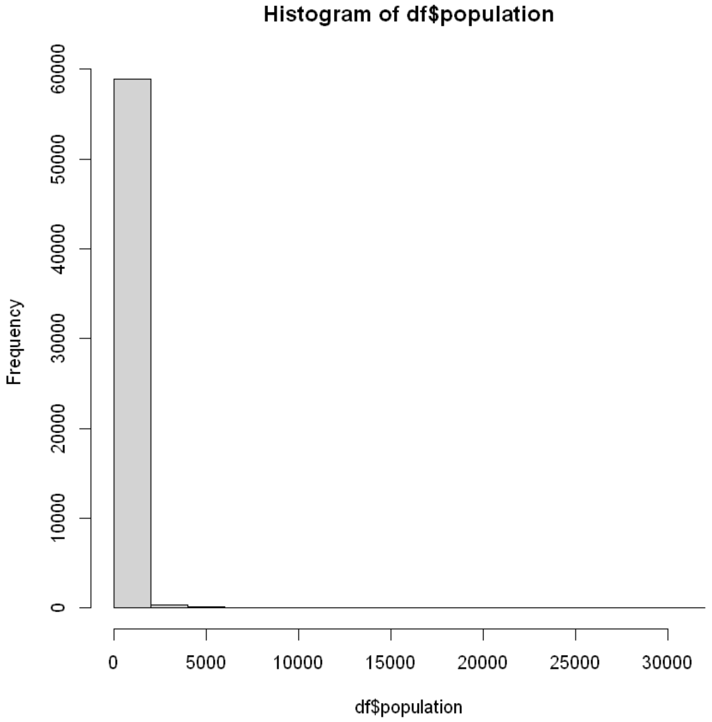
In [6]:

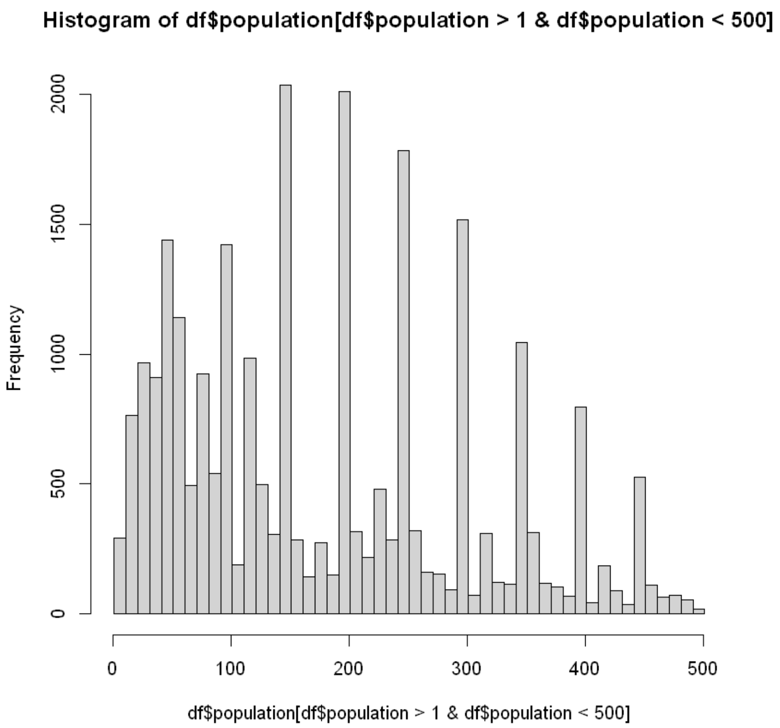
```
sum(df$population==0)/nrow(df)
sum(df$population==1)/nrow(df)
hist(df$population)
```

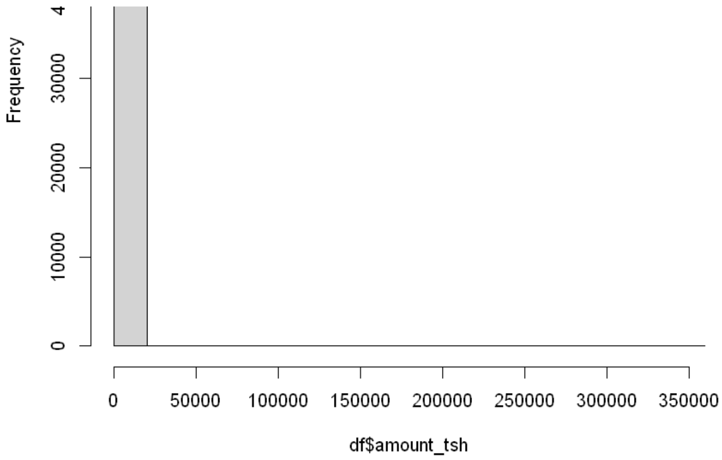
```r
hist(df$population[df$population>1 & df$population <500], breaks = seq(from=1, to=501, by=10))
hist(df$amount_tsh)
hist(df$amount_tsh[df$amount_tsh>0 & df$amount_ts <1000])
sum(df$amount_tsh==0)/nrow(df)
table(df$status_group)
```

0.359949494949495

0.118265993265993

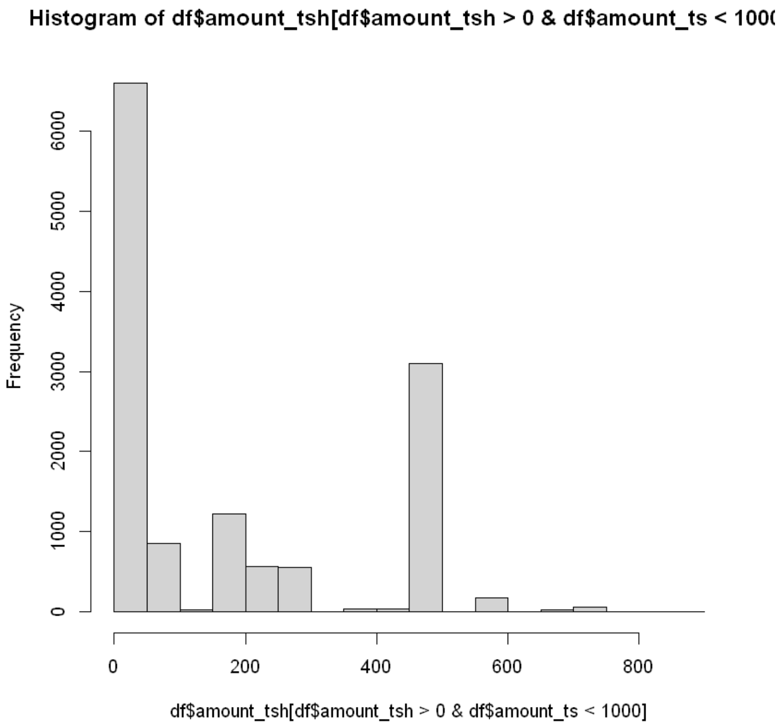






0.700993265993266

```
            functional functional needs repair         non functional 
                 32259                    4317                  22824 
```



The data set contains two information about time: one about the year when the pump was constructed (*construction_year*), the other about the exact date when the pump was tested (*date_recorded*). *construction_year* equals zero for about 35% of the observation. This is obviously a measurement error. We set these observations to the median. *date_recorded* could be used as a factor variable with 356 different values and capturing daily fixed effect. However given that this contributes to considerably increase dimensions of the problem and therefore computing-time for several machine learning procedures, we decide to exploit information from this variable differently. First we transform it into a number to capture time trained at daily rate. We delete 31 observations with strictly implausible values. Then we create 3 categorical variables capturing the year, month and day of the week, during which the measure was taken. Finally we generate the variable *age* equal to the difference between the year of the observation and the construction year.

In [7]:

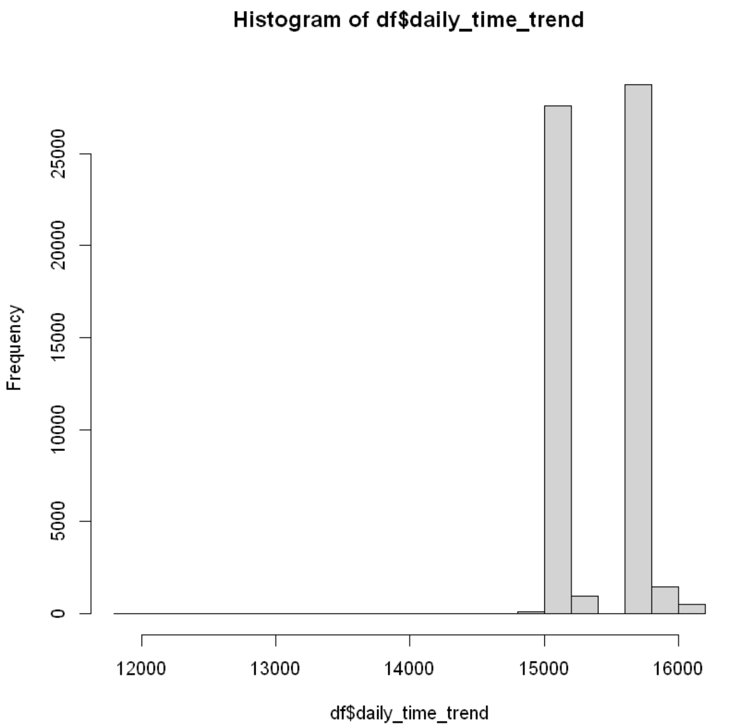
```
sum(df$construction_year==0)/nrow(df)
describe(df$construction_year[df$construction_year!=0])
df$construction_year[df$construction_year==0] <- 2000

df$m_date <- as.Date(df$date_recorded)
df$daily_time_trend <- as.numeric(df$m_date)
hist(df$daily_time_trend)
table(df$daily_time_trend)
hist(df$daily_time_trend[df$daily_time_trend>14942], breaks = seq(from=14942, to=16042,
```

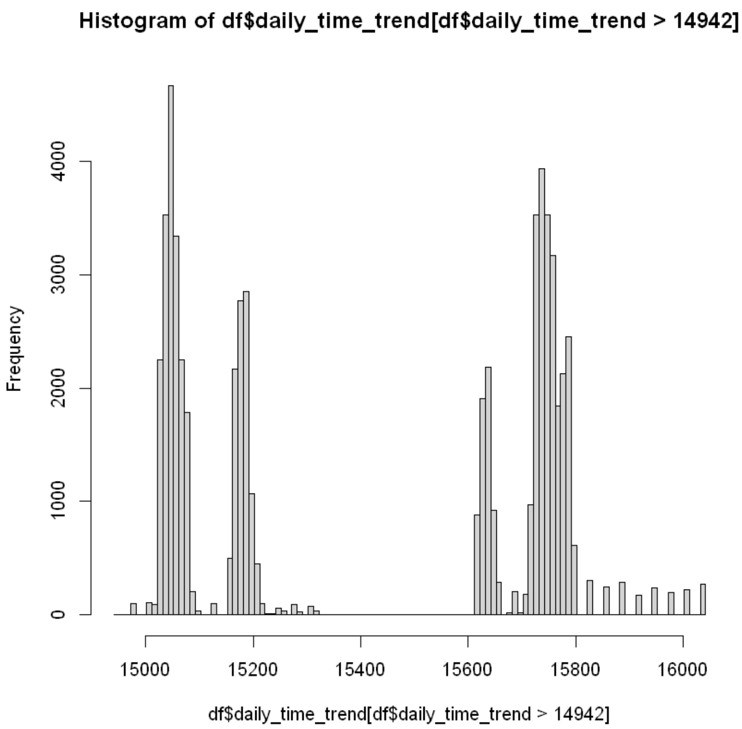
```r
by=10))

df$m_months <- as.factor(as.numeric(format(df$m_date,'%m')))
df$m_day <- as.factor(weekdays(df$m_date))

df$m_year <- as.numeric(format(df$m_date,'%Y'))
df$age <- df$m_year - df$construction_year

df$age <- as.numeric(df$m_year) - df$construction_year

 df<- df[, !(colnames(df) %in% c("date_recorded"))]
```

0.348636363636364

```
df$construction_year[df$construction_year != 0]
       n  missing distinct     Info     Mean      Gmd      .05      .10
   38691        0       54    0.998     1997    13.87     1973     1978
     .25      .50      .75      .90      .95
    1987     2000     2008     2010     2011

lowest : 1960 1961 1962 1963 1964, highest: 2009 2010 2011 2012 2013

11974 12424 12478 12483 12509 12513 12539 12570 12600 12631 12662 12753 14977
    1     1     4     1     1     1     2     1     1     2     2    14    30
14978 14982 15006 15007 15008 15009 15019 15020 15021 15022 15023 15024 15025
   22    47     7    17    53    33    13    23    36    18   130   117   154
15026 15027 15028 15029 15030 15031 15032 15033 15034 15035 15036 15037 15038
  189   175   323   263   254   351   294   341   276   232   378   458   434
15039 15040 15041 15042 15043 15044 15045 15046 15047 15048 15049 15050 15051
  327   333   328   416   364   426   379   373   520   572   513   558   497
15052 15053 15054 15055 15056 15057 15058 15059 15060 15061 15062 15063 15064
  466   262   351   364   417   381   294   342   298   337   296   391   213
15065 15066 15067 15068 15069 15070 15071 15072 15073 15074 15075 15076 15077
  252   264   267   254   114   145   185   167   161   159   157   160   154
15078 15079 15080 15081 15082 15083 15084 15085 15086 15087 15097 15098 15101
  181   204   133   337   135    59    74    27    31    11    11    14     5
15128 15129 15132 15158 15159 15160 15161 15162 15163 15164 15165 15166 15167
   19    43    35    17    61    99   144   180   200   120   183   226   228
15168 15169 15170 15171 15172 15173 15174 15175 15176 15177 15178 15179 15180
  240   197   272   265   234   320   338   332   253   262   276   223   181
15181 15182 15183 15184 15185 15186 15187 15188 15189 15190 15191 15192 15193
  215   373   320   304   345   235   191   294   330   326   274   230   216
15194 15195 15196 15197 15198 15199 15200 15201 15202 15203 15204 15205 15206
  167    82   115   136   102    81    64    56    48    79    50    26    25
15207 15208 15209 15210 15211 15212 15213 15214 15216 15217 15218 15220 15221
   28    45    71    26    32    64    26     2     1     1     1    16    48
15222 15223 15225 15226 15228 15229 15230 15231 15232 15233 15234 15235 15236
    1     1     1     1     1     1     1     1     1     1     1     1     1
15237 15238 15240 15242 15243 15244 15245 15250 15251 15254 15281 15282 15285
    1     1     1     1     1     1     1    13    45    35    65    23    25
15311 15315 15360 15364 15614 15615 15616 15617 15618 15619 15620 15621 15622
   71    37     1     1    18    55   108   123   118   109   107   101   140
15623 15624 15625 15626 15627 15628 15629 15630 15631 15632 15633 15634 15635
  146   179   215   211   150   153   166   224   244   216   193   250   353
15636 15637 15638 15639 15640 15641 15642 15643 15644 15645 15646 15647 15648
  265   251   179   200   202   135   152   105   111    71    73   155    90
15649 15650 15651 15652 15653 15654 15655 15656 15657 15658 15659 15663 15673
  121   108    17    72    30    23    25    75    73    19    45     4     8
15674 15684 15685 15686 15687 15688 15689 15690 15691 15692 15695 15697 15698
    9     5    12     5    53    42    37    27    10    16     6     5     7
15706 15708 15709 15711 15712 15713 15714 15715 15716 15717 15718 15719 15720
    1    86    71     1    18    15    25    11    41    30    24    23   145
15721 15722 15723 15724 15725 15726 15727 15728 15729 15730 15731 15732 15733
  342   312   409   368   331   315   364   379   435   338   251   337   376
15734 15735 15736 15737 15738 15739 15740 15741 15742 15743 15744 15745 15746
  459   421   324   338   370   546   464   324   315   306   363   295   281
15747 15748 15749 15750 15751 15752 15753 15754 15755 15756 15757 15758 15759
  272   337   380   444   429   418   290   369   371   333   283   299   305
15760 15761 15762 15763 15764 15765 15766 15767 15768 15769 15770 15771 15772
  251   277   391   340   220   139   122   213   347    86   132   123   118
15773 15774 15775 15776 15777 15778 15779 15780 15781 15782 15783 15784 15785
   32    45    28   102   346   283   428   319   187   360   381   336   313
15786 15787 15788 15789 15790 15791 15792 15793 15794 15797 15798 15799 15827
```

```
15786 15787 15788 15789 15790 15791 15792 15793 15794 15797 15798 15799 15827
  248   241   179   204   212   181   157   130   143    13   153   171    17
15828 15829 15859 15860 15888 15889 15890 15919 15920 15950 15951 15980 15981
  185   102   218    30    14   237    33    42   132    32   209    55   138
16011 16012 16040 16041 16042
   24   194     1    33   240
```





The remaining variables are all categorical. One subgroup provide information on the geographical location (*basin subvillage region region_code district_code lga ward*). The rest provide diverse information about the pump. The geographical variable can be divided into two groups depending on the number of different categories: few (*basin region region_code district_code lga ward*) or many (*subvillage ward*). We remove from our data set those with too many levels.

In [8]:

```
table(df$basin)
table(df$region, df$region_code)
table(df$district_code)
table(df$region_code)
```

```r
table(df$lga)
describe(df$ward)
describe(df$subvillage)
 df<- df[, !(colnames(df) %in% c("ward", "subvillage"))]
```

```
               Internal                Lake Nyasa                Lake Rukwa
                   7785                      5085                      2454
        Lake Tanganyika             Lake Victoria                   Pangani
                   6432                     10248                      8940
                 Rufiji Ruvuma / Southern Coast               Wami / Ruvu
                   7976                      4493                      5987


                    1    2    3    4    5    6    7    8    9   10   11   12
  Arusha            0 3024    0    0    0    0    0    0    0    0    0    0
  Dar es Salaam     0    0    0    0    0    0  805    0    0    0    0    0
  Dodoma         2201    0    0    0    0    0    0    0    0    0    0    0
  Iringa            0    0    0    0    0    0    0    0    0    0 5294    0
  Kagera            0    0    0    0    0    0    0    0    0    0    0    0
  Kigoma            0    0    0    0    0    0    0    0    0    0    0    0
  Kilimanjaro       0    0 4379    0    0    0    0    0    0    0    0    0
  Lindi             0    0    0    0    0    0    0  300    0    0    0    0
  Manyara           0    0    0    0    0    0    0    0    0    0    0    0
  Mara              0    0    0    0    0    0    0    0    0    0    0    0
  Mbeya             0    0    0    0    0    0    0    0    0    0    0 4639
  Morogoro          0    0    0    0 4006    0    0    0    0    0    0    0
  Mtwara            0    0    0    0    0    0    0    0  390    0    0    0
  Mwanza            0    0    0    0    0    0    0    0    0    0    0    0
  Pwani             0    0    0    0    0 1609    0    0    0    0    0    0
  Rukwa             0    0    0    0    0    0    0    0    0    0    0    0
  Ruvuma            0    0    0    0    0    0    0    0    0 2640    0    0
  Shinyanga         0    0    0    0    0    0    0    0    0    0    6    0
  Singida           0    0    0    0    0    0    0    0    0    0    0    0
  Tabora            0    0    0    0    0    0    0    0    0    0    0    0
  Tanga             0    0    0 2513   34    0    0    0    0    0    0    0

                   13   14   15   16   17   18   19   20   21   24   40   60
  Arusha            0    0    0    0    0    0    0    0    0  326    0    0
  Dar es Salaam     0    0    0    0    0    0    0    0    0    0    0    0
  Dodoma            0    0    0    0    0    0    0    0    0    0    0    0
  Iringa            0    0    0    0    0    0    0    0    0    0    0    0
  Kagera            0    0    0    0    0 3316    0    0    0    0    0    0
  Kigoma            0    0    0 2816    0    0    0    0    0    0    0    0
  Kilimanjaro       0    0    0    0    0    0    0    0    0    0    0    0
  Lindi             0    0    0    0    0    8    0    0    0    0    0    0
  Manyara           0    0    0    0    0    0    0    0 1583    0    0    0
  Mara              0    0    0    0    0    0    0 1969    0    0    0    0
  Mbeya             0    0    0    0    0    0    0    0    0    0    0    0
  Morogoro          0    0    0    0    0    0    0    0    0    0    0    0
  Mtwara            0    0    0    0    0    0    0    0    0    0    0    0
  Mwanza            0    0    0    0   55    0 3047    0    0    0    0    0
  Pwani             0    0    0    0    0    0    0    0    0    0    1 1025
  Rukwa             0    0 1808    0    0    0    0    0    0    0    0    0
  Ruvuma            0    0    0    0    0    0    0    0    0    0    0    0
  Shinyanga         0   20    0    0 4956    0    0    0    0    0    0    0
  Singida        2093    0    0    0    0    0    0    0    0    0    0    0
  Tabora            0 1959    0    0    0    0    0    0    0    0    0    0
  Tanga             0    0    0    0    0    0    0    0    0    0    0    0

                   80   90   99
  Arusha            0    0    0
  Dar es Salaam     0    0    0
  Dodoma            0    0    0
  Iringa            0    0    0
  Kagera            0    0    0
  Kigoma            0    0    0
  Kilimanjaro       0    0    0
  Lindi          1238    0    0
  Manyara           0    0    0
  Mara              0    0    0
  Mbeya             0    0    0
  Morogoro          0    0    0
  Mtwara            0  917  423
```

```
  Mtwara            0  917  423
  Mwanza            0    0    0
  Pwani             0    0    0
  Rukwa             0    0    0
  Ruvuma            0    0    0
  Shinyanga         0    0    0
  Singida           0    0    0
  Tabora            0    0    0
  Tanga             0    0    0

    0     1     2     3     4     5     6     7     8    13    23    30    33 
   23 12203 11173  9998  8999  4356  4074  3343  1043   391   293   995   874 
   43    53    60    62    63    67    80 
  505   745    63   109   195     6    12 

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16 
2201 3024 4379 2513 4040 1609  805  300  390 2640 5300 4639 2093 1979 1808 2816 
  17   18   19   20   21   24   40   60   80   90   99 
5011 3324 3047 1969 1583  326    1 1025 1238  917  423 

    Arusha Rural     Arusha Urban           Babati         Bagamoyo 
            1252               63              511              997 
            Bahi          Bariadi       Biharamulo     Bukoba Rural 
             224             1177              403              487 
    Bukoba Urban          Bukombe            Bunda         Chamwino 
              88              514              438              347 
           Chato           Chunya    Dodoma Urban            Geita 
             236              298              358              488 
             Hai           Hanang          Handeni           Igunga 
             625              274              254              338 
           Ilala            Ileje          Ilemela           Iramba 
             497              231              142              544 
    Iringa Rural           Kahama          Karagwe           Karatu 
             728              836              771              326 
          Kasulu           Kibaha          Kibondo     Kigoma Rural 
            1047              269              874              824 
    Kigoma Urban          Kilindi           Kilolo        Kilombero 
              71              161              349              959 
          Kilosa            Kilwa        Kinondoni         Kisarawe 
            1094              392               93              223 
         Kishapu           Kiteto           Kondoa           Kongwa 
             399              193              523              361 
         Korogwe           Kwimba            Kyela      Lindi Rural 
             412              627              859              388 
     Lindi Urban           Liwale          Longido           Ludewa 
              21              154              310              564 
         Lushoto            Mafia             Magu           Makete 
             694              132              824              630 
         Manyoni           Masasi            Maswa          Mbarali 
             377              528              809              626 
     Mbeya Rural           Mbinga           Mbozi            Mbulu 
             485              750             1034              297 
           Meatu             Meru         Misenyi        Missungwi 
             468             1009              260              348 
          Mkinga         Mkuranga          Monduli   Morogoro Rural 
             288              560              189              521 
  Morogoro Urban      Moshi Rural      Moshi Urban           Mpanda 
              96             1251               79              679 
         Mpwapwa     Mtwara Rural     Mtwara Urban          Mufindi 
             388              423              124              520 
          Muheza           Muleba     Musoma Rural          Mvomero 
             334              402              396              671 
          Mwanga       Nachingwea         Namtumbo          Nanyumbu 
             519              300              694              158 
          Newala            Ngara       Ngorongoro           Njombe 
             231              669              201             2503 
           Nkasi        Nyamagana            Nzega          Pangani 
             428                1              575              305 
           Rombo            Rorya          Ruangwa           Rufiji 
             594              210              291              454 
          Rungwe             Same        Sengerema        Serengeti 
            1106              877              331              716 
 Shinyanga Rural  Shinyanga Urban             Siha          Sikonge 
```

```
           588                191                434                170
     Simanjiro     Singida Rural     Singida Urban      Songea Rural
           308                995                177                693
  Songea Urban Sumbawanga Rural  Sumbawanga Urban      Tabora Urban
            80                521                180                155
    Tandahimba              Tanga             Tarime            Temeke
           266                 99                209                215
       Tunduru            Ukerewe             Ulanga            Urambo
           423                341                665                382
          Uyui
           339

df$ward
       n  missing distinct
   59400        0     2092

lowest : Aghondi      Akheri       Arash        Arri         Arusha Chini
highest: Ziwani       Zoissa       Zombo        Zongomera    Zuzu

df$subvillage
       n  missing distinct
   59029      371    19287

lowest : 'A' Kati          ##                 1                  14Kambalage        18

highest: Zumbawanu Shuleni Zunga              Zunguni            Zunzuli            Zuri
```

Next comes the group of variable giving information about the pump with different levels of details (*extraction_type extraction_type_group extraction_type_class management management_group payment payment_type water_quality quality_group quantity quantity_group source source_type source_class waterpoint_type waterpoint_type_group*). For instance *extraction_type* is strictly more precise than *extraction_type_group*, which is itself strictly more precise than *extraction_type_class*. As computational time varies a lot across machine learning algorithm depending on the number of features, we decide to keep both precise and general versions of these features, thus is allowing us to pick one or the other, depending on the model at use.

In [9]:

```
table(df$extraction_type, df$extraction_type_group)
  table(df$extraction_type_group, df$extraction_type_class)
  table(df$management, df$management_group )
  table(df$payment,df$payment_type )
  table(df$water_quality,df$quality_group )
  table(df$quantity, df$quantity_group)
  table(df$source, df$source_type )
  table(df$source_type,df$source_class )
  table(df$waterpoint_type,df$waterpoint_type_group )
```

```
                            afridev gravity india mark ii india mark iii  mono
  afridev                      1770       0             0              0     0
  cemo                            0       0             0              0     0
  climax                          0       0             0              0     0
  gravity                         0   26780             0              0     0
  india mark ii                   0       0          2400              0     0
  india mark iii                  0       0             0             98     0
  ksb                             0       0             0              0     0
  mono                            0       0             0              0  2865
  nira/tanira                     0       0             0              0     0
  other                           0       0             0              0     0
  other - mkulima/shinyanga       0       0             0              0     0
  other - play pump               0       0             0              0     0
  other - rope pump               0       0             0              0     0
  other - swn 81                  0       0             0              0     0
  submersible                     0       0             0              0     0
  swn 80                          0       0             0              0     0
  walimi                          0       0             0              0     0
  windmill                        0       0             0              0     0

                            nira/tanira other other handpump other motorpump
```

afridev	0	0	0	0
cemo	0	0	0	90
climax	0	0	0	32
gravity	0	0	0	0
india mark ii	0	0	0	0
india mark iii	0	0	0	0
ksb	0	0	0	0
mono	0	0	0	0
nira/tanira	8154	0	0	0
other	0	6430	0	0
other - mkulima/shinyanga	0	0	2	0
other - play pump	0	0	85	0
other - rope pump	0	0	0	0
other - swn 81	0	0	229	0
submersible	0	0	0	0
swn 80	0	0	0	0
walimi	0	0	48	0
windmill	0	0	0	0

	rope pump	submersible	swn 80	wind-powered
afridev	0	0	0	0
cemo	0	0	0	0
climax	0	0	0	0
gravity	0	0	0	0
india mark ii	0	0	0	0
india mark iii	0	0	0	0
ksb	0	1415	0	0
mono	0	0	0	0
nira/tanira	0	0	0	0
other	0	0	0	0
other - mkulima/shinyanga	0	0	0	0
other - play pump	0	0	0	0
other - rope pump	451	0	0	0
other - swn 81	0	0	0	0
submersible	0	4764	0	0
swn 80	0	0	3670	0
walimi	0	0	0	0
windmill	0	0	0	117

	gravity	handpump	motorpump	other	rope pump	submersible
afridev	0	1770	0	0	0	0
gravity	26780	0	0	0	0	0
india mark ii	0	2400	0	0	0	0
india mark iii	0	98	0	0	0	0
mono	0	0	2865	0	0	0
nira/tanira	0	8154	0	0	0	0
other	0	0	0	6430	0	0
other handpump	0	364	0	0	0	0
other motorpump	0	0	122	0	0	0
rope pump	0	0	0	0	451	0
submersible	0	0	0	0	0	6179
swn 80	0	3670	0	0	0	0
wind-powered	0	0	0	0	0	0

	wind-powered
afridev	0
gravity	0
india mark ii	0
india mark iii	0
mono	0
nira/tanira	0
other	0
other handpump	0
other motorpump	0
rope pump	0
submersible	0
swn 80	0
wind-powered	117

	commercial	other	parastatal	unknown	user-group
company	685	0	0	0	0

other	0	844	0	0	0
other - school	0	99	0	0	0
parastatal	0	0	1768	0	0
private operator	1971	0	0	0	0
trust	78	0	0	0	0
unknown	0	0	0	561	0
vwc	0	0	0	0	40507
water authority	904	0	0	0	0
water board	0	0	0	0	2933
wua	0	0	0	0	2535
wug	0	0	0	0	6515

	annually	monthly	never pay	on failure	other	per bucket
never pay	0	0	25348	0	0	0
other	0	0	0	0	1054	0
pay annually	3642	0	0	0	0	0
pay monthly	0	8300	0	0	0	0
pay per bucket	0	0	0	0	0	8985
pay when scheme fails	0	0	0	3914	0	0
unknown	0	0	0	0	0	0

	unknown
never pay	0
other	0
pay annually	0
pay monthly	0
pay per bucket	0
pay when scheme fails	0
unknown	8157

	colored	fluoride	good	milky	salty	unknown
coloured	490	0	0	0	0	0
fluoride	0	200	0	0	0	0
fluoride abandoned	0	17	0	0	0	0
milky	0	0	0	804	0	0
salty	0	0	0	0	4856	0
salty abandoned	0	0	0	0	339	0
soft	0	0	50818	0	0	0
unknown	0	0	0	0	0	1876

	dry	enough	insufficient	seasonal	unknown
dry	6246	0	0	0	0
enough	0	33186	0	0	0
insufficient	0	0	15129	0	0
seasonal	0	0	0	4050	0
unknown	0	0	0	0	789

	borehole	dam	other	rainwater harvesting	river/lake
dam	0	656	0	0	0
hand dtw	874	0	0	0	0
lake	0	0	0	0	765
machine dbh	11075	0	0	0	0
other	0	0	212	0	0
rainwater harvesting	0	0	0	2295	0
river	0	0	0	0	9612
shallow well	0	0	0	0	0
spring	0	0	0	0	0
unknown	0	0	66	0	0

	shallow well	spring
dam	0	0
hand dtw	0	0
lake	0	0
machine dbh	0	0
other	0	0
rainwater harvesting	0	0
river	0	0
shallow well	16824	0
spring	0	17021

```
spring                        0   17021
unknown                       0       0
```

```
                     groundwater surface unknown
borehole                   11949       0       0
dam                            0     656       0
other                          0       0     278
rainwater harvesting           0    2295       0
river/lake                     0   10377       0
shallow well               16824       0       0
spring                     17021       0       0
```

```
                            cattle trough communal standpipe   dam hand pump
cattle trough                         116                  0     0         0
communal standpipe                      0              28522     0         0
communal standpipe multiple             0               6103     0         0
dam                                     0                  0     7         0
hand pump                               0                  0     0     17488
improved spring                         0                  0     0         0
other                                   0                  0     0         0

                            improved spring other
cattle trough                             0     0
communal standpipe                        0     0
communal standpipe multiple               0     0
dam                                       0     0
hand pump                                 0     0
improved spring                         784     0
other                                     0  6380
```

Finally we create a dummy variable equal to 1 whenever the *funder* is the same as the *installer* and remove these two features for they contain too many features.

In [10]:

```
df$installer <- tolower(df$installer)
df$funder <- tolower(df$funder)
df$funder_is_installer <- df$funder == df$installer & df$installer!=""
df$funder_is_installer[df$installer=="" | df$funder==""] <- "MISSING"
table(df$funder_is_installer)

df<- df[, !(colnames(df) %in% c("funder", "installer"))]
```

```
FALSE MISSING    TRUE
36762    3708   18930
```

Last we run a chi-test for each of the categorical variable with relatively few levels against our dependent variable. All p-values are below significant levels therefore if any discrimination can be made between this categorical variable it should be on the basis of the magnitude of the coefficient. By decreasing order the most relevant categorical variable are the following : *quantity quantity_group waterpoint_type extraction_type extraction_type_group extraction_type_class waterpoint_type_group region_code region payment_type payment source source water_quality quality_group management scheme_management basin source_type district_code source_class public_meeting management_group permit*.

In [11]:

```
chisq.test(df$status_group, df$basin)
chisq.test(df$status_group, df$region)
chisq.test(df$status_group, df$region_code)
chisq.test(df$status_group, df$district_code)
chisq.test(df$status_group, df$public_meeting)
chisq.test(df$status_group, df$scheme_management)
chisq.test(df$status_group, df$permit)
chisq.test(df$status_group, df$extraction_type)
chisq.test(df$status_group, df$extraction_type_group)
chisq.test(df$status_group, df$extraction_type_class)
chisq.test(df$status_group, df$management)
chisq.test(df$status_group, df$management_group)
```

```
chisq.test(df$status_group, df$payment)
chisq.test(df$status_group, df$payment_type)
chisq.test(df$status_group, df$water_quality)
chisq.test(df$status_group, df$quality_group)
chisq.test(df$status_group, df$quantity)
chisq.test(df$status_group, df$quantity_group)
chisq.test(df$status_group, df$source)
chisq.test(df$status_group, df$source_type)
chisq.test(df$status_group, df$source_class)
chisq.test(df$status_group, df$waterpoint_type)
chisq.test(df$status_group, df$waterpoint_type_group)
```

```
	Pearson's Chi-squared test

data:  df$status_group and df$basin
X-squared = 1921, df = 16, p-value < 2.2e-16

	Pearson's Chi-squared test

data:  df$status_group and df$region
X-squared = 4794.6, df = 40, p-value < 2.2e-16
```

```
Warning message in chisq.test(df$status_group, df$region_code):
"Chi-squared approximation may be incorrect"
```

```
	Pearson's Chi-squared test

data:  df$status_group and df$region_code
X-squared = 5157.4, df = 52, p-value < 2.2e-16
```

```
Warning message in chisq.test(df$status_group, df$district_code):
"Chi-squared approximation may be incorrect"
```

```
	Pearson's Chi-squared test

data:  df$status_group and df$district_code
X-squared = 1673.5, df = 38, p-value < 2.2e-16

	Pearson's Chi-squared test

data:  df$status_group and df$public_meeting
X-squared = 384, df = 4, p-value < 2.2e-16
```

```
Warning message in chisq.test(df$status_group, df$scheme_management):
"Chi-squared approximation may be incorrect"
```

```
	Pearson's Chi-squared test

data:  df$status_group and df$scheme_management
X-squared = 1991.1, df = 24, p-value < 2.2e-16

	Pearson's Chi-squared test

data:  df$status_group and df$permit
X-squared = 104.18, df = 4, p-value < 2.2e-16
```

```
Warning message in chisq.test(df$status_group, df$extraction_type):
"Chi-squared approximation may be incorrect"
```

```
	Pearson's Chi-squared test

data:  df$status_group and df$extraction_type
X-squared = 7365.6, df = 34, p-value < 2.2e-16

	Pearson's Chi-squared test

data:  df$status_group and df$extraction_type_group
X-squared = 7265.8, df = 24, p-value < 2.2e-16

	Pearson's Chi-squared test

data:  df$status_group and df$extraction_type_class
X-squared = 6931.2, df = 12, p-value < 2.2e-16

	Pearson's Chi-squared test
```

```
data:  df$status_group and df$management
X-squared = 2081.1, df = 22, p-value < 2.2e-16

 Pearson's Chi-squared test

data:  df$status_group and df$management_group
X-squared = 287.65, df = 8, p-value < 2.2e-16

 Pearson's Chi-squared test

data:  df$status_group and df$payment
X-squared = 3965.6, df = 12, p-value < 2.2e-16

 Pearson's Chi-squared test

data:  df$status_group and df$payment_type
X-squared = 3965.6, df = 12, p-value < 2.2e-16

Warning message in chisq.test(df$status_group, df$water_quality):
"Chi-squared approximation may be incorrect"

 Pearson's Chi-squared test

data:  df$status_group and df$water_quality
X-squared = 2277.4, df = 14, p-value < 2.2e-16

 Pearson's Chi-squared test

data:  df$status_group and df$quality_group
X-squared = 2100.1, df = 10, p-value < 2.2e-16

 Pearson's Chi-squared test

data:  df$status_group and df$quantity
X-squared = 11361, df = 8, p-value < 2.2e-16

 Pearson's Chi-squared test

data:  df$status_group and df$quantity_group
X-squared = 11361, df = 8, p-value < 2.2e-16

Warning message in chisq.test(df$status_group, df$source):
"Chi-squared approximation may be incorrect"

 Pearson's Chi-squared test

data:  df$status_group and df$source
X-squared = 2624, df = 18, p-value < 2.2e-16

 Pearson's Chi-squared test

data:  df$status_group and df$source_type
X-squared = 1906.8, df = 12, p-value < 2.2e-16

 Pearson's Chi-squared test

data:  df$status_group and df$source_class
X-squared = 590.26, df = 4, p-value < 2.2e-16

Warning message in chisq.test(df$status_group, df$waterpoint_type):
"Chi-squared approximation may be incorrect"

 Pearson's Chi-squared test

data:  df$status_group and df$waterpoint_type
X-squared = 7450.3, df = 12, p-value < 2.2e-16

Warning message in chisq.test(df$status_group, df$waterpoint_type_group):
"Chi-squared approximation may be incorrect"

 Pearson's Chi-squared test

data:  df$status_group and df$waterpoint_type_group
X-squared = 6114.8, df = 10, p-value < 2.2e-16
```

We apply all modification to the *test_value* dataset

In [12]:

```
test_values<- test_values[, !(colnames(test_values) %in% c("recorded_by", "num_private",
"wpt_name"))]

# test_values <- test_values %>% filter(test_values$longitude!=0)
test_values$construction_year[test_values$construction_year==0] <- 2000
test_values$m_date <- as.Date(test_values$date_recorded)
test_values$daily_time_trend <- as.numeric(test_values$m_date)
# test_values <- test_values %>% filter(test_values$daily_time_trend>=14977)

test_values$m_months <- as.factor(as.numeric(format(test_values$m_date,'%m')))
test_values$m_day <- as.factor(weekdays(test_values$m_date))

test_values$m_year <- as.numeric(format(test_values$m_date,'%Y'))
test_values$age <- test_values$m_year - test_values$construction_year

test_values<- test_values[, !(colnames(test_values) %in% c("ward", "subvillage"))]

test_values$installer <- tolower(test_values$installer)
test_values$funder <- tolower(test_values$funder)
test_values$funder_is_installer <- test_values$funder == test_values$installer & test_va
lues$installer!=""
test_values$funder_is_installer[test_values$installer=="" | test_values$funder==""] <- "
MISSING"
```

2. Model selection

For this part we decided on 3 different learning methods: Decision trees using the package rpart, Support Vector Machine from e1071 and random forest from randomForest

For the learning method and model assessment we use cross-validation to choose the best performing learning methods, to see which method is less susceptible to bias and overfitting to the training dataset. This tells us less about which learning methods is best at predicting its own training dataset and more about how the models will perform in test datasets that are disconnected from its training.

This helps us identify which method works best to avoid overfitting the data to the training set so that we can maximise the score of the prediction of DrivenData's test set. For each method, the features are the same, all parameters are set to their default. We assume that this would be fair, as we could not afford spending an absurd amount of computational time hyperparameter tuning each method.

For each of the learning methods, we sample a subset of data for a sub-training dataset, and sample another subset for the testing datasets. We calculated the differences of predictions of the test set and averaged out the errors.

Random Forest

In [12]:

```
# testing k-fold cross validation here
df$random <- runif(nrow(df), min=1, max=60000)
df$subset <-  ntile(df$random, 10)

library(randomForest)

rf_model_1 <- function(some_number) {
    #sampling subsets for training and testing dfs
  df_train <- df %>% filter(df$subset == some_number)
  df_test <- df %>% filter(df$subset != some_number)
# train a model
```

```
  model_forest <- randomForest(as.factor(status_group) ~
                                + gps_height
                               + longitude + latitude
                               + extraction_type_group + quantity + source ,
                                data = df_train,)
  # predicting the test set
  df_test$y_pred <- predict(model_forest, df_test)
# calculating the error
  error <- sum(df_test$y_pred!=df_test$status_group)/nrow(df_test)

  results <- list("df_test" = df_test , "model_forest" = model_forest, "error" = error)

  return(c(results, model_forest))
}

allerrors=c()

list_results <- lapply(1:10, rf_model_1) #Generate data


for (i in 1:10){
  allerrors <- c(allerrors, list_results[[i]][3]$error)
}
print("Average random forest error rate")
print(mean(allerrors))
```

```
randomForest 4.6-14

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


Attaching package: 'randomForest'


The following object is masked from 'package:dplyr':

    combine


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

    margin
```

```
[1] "Average random forest error rate"
[1] 0.264407
```

Decision Trees

In [13]:

```
library(rpart)
df$random <- runif(nrow(df), min=1, max=60000)
df$subset <-  ntile(df$random, 10)

tr_model_1 <- function(some_number) {
  df_train <- df %>% filter(df$subset == some_number)
  df_test <- df %>% filter(df$subset != some_number)

  model_rpart <- rpart(as.factor(status_group) ~
                                  + gps_height
                               + longitude + latitude + management
                               + extraction_type_group + quantity + source,
                               data = df_train)


  df_test$y_pred <- predict(model_rpart, df_test, type = "class")
```

```
  error <- sum(df_test$y_pred!=df_test$status_group)/nrow(df_test)

  results <- list("df_test" = df_test , "model_rpart" = model_rpart, "error" = error)

  return(results)
}

list_results <- lapply(1:10, tr_model_1) #Generate data

allerrors=c()
for (i in 1:10){
  allerrors <- c(allerrors, list_results[[i]][3]$error)
}

print("Average decision tree error rate")
print(mean(allerrors))
```

```
[1] "Average decision tree error rate"
[1] 0.3032941
```

Support Vector Machine

Note: This cell may take up to 5m to complete

In [14]:

```
library(e1071)
df$random <- runif(nrow(df), min=1, max=60000)
df$subset <-  ntile(df$random, 10)

svm_model_1 <- function(some_number) {
  df_train <- df %>% filter(df$subset == some_number)
  df_test <- df %>% filter(df$subset != some_number)

  model_svm <- svm(as.factor(status_group) ~
                                + gps_height
                              + longitude + latitude
                              + extraction_type_group + quantity + source,
                              data = df_train)



  df_test$y_pred <- predict(model_svm, df_test, type = "class")


  error <- sum(df_test$y_pred!=df_test$status_group)/nrow(df_test)

  results <- list("df_test" = df_test , "model_svm" = model_svm, "error" = error)

  return(results)
}

list_results <- lapply(1:10, svm_model_1) #Generate data

allerrors=c()
for (i in 1:10){
  allerrors <- c(allerrors, list_results[[i]][3]$error)
}
print("Average SVM error rate")
print(mean(allerrors))
```

```
Attaching package: 'e1071'


The following object is masked from 'package:Hmisc':

    impute
```

```
[1] "Average SVM error rate"
[1] 0.3053311
```

Based on these results we found that on average, models derived from the randomForest approach seem to be performing the best, it is the learning method that's the least prone to overfitting.

We will further validate the model by using the model made from the dataset to predict its own training model, see how well it performs.

In [15]:

```
model_forest <- randomForest(as.factor(status_group) ~
                               + gps_height + amount_tsh
                               + longitude + latitude
                               + water_quality + quantity
                               + construction_year + district_code
                                + population + scheme_management + source,
                               data = df, importance=TRUE,
                               ntree = 82, nodesize = 5)

# predict training set with own model
forest_pred_train <- predict(model_forest, df)
frerror <- sum(forest_pred_train!=df$status_group)/nrow(df)
print(frerror)
```

```
[1] 0.1083333
```

As we can see, it's pretty good at predicting its own training dataset, with an accuracy of around 90%. However, the problem with this is that we can always keep increasing this accuracy by removing features that decrease accuracy(low MeanDecreaseAccuracy):

In [16]:

```
# get model statistics
importance(model_forest)
```

A matrix: 11 × 5 of type dbl

	functional	functional needs repair	non functional	MeanDecreaseAccuracy	MeanDecreaseGini
gps_height	34.39114	24.10932	29.16074	42.88251	2316.4981
amount_tsh	43.82203	25.21307	26.04596	54.23750	1347.5128
longitude	46.58416	35.91711	39.91826	73.96162	3921.1147
latitude	56.13789	34.59848	45.99041	83.71714	3800.2329
water_quality	21.25659	14.14418	18.57473	24.41223	548.4099
quantity	92.52433	38.38091	119.69305	132.28225	4438.5402
construction_year	27.00938	28.38941	35.14898	42.67699	2195.5471
district_code	37.59373	25.29965	40.95607	56.84924	1048.2598
population	22.15101	14.81803	19.30993	29.46698	1513.3235
scheme_management	34.89092	10.80960	27.02708	37.81756	812.2543
source	45.22860	29.05990	38.33517	51.23014	1113.4281

Doing this excessively will undoubtedly increases the accuracy and improve the model's ability to predict the training dataset, however, we will hit a point where our average prediction accuracy of the test set decreases. This is an overfitting issue, this is why we intentionally tried to not let the feature statistic importance influence too much to our feature selection process. This is the same reasoning why we chose our learning procedure using Cross Validation errors.

Hyperparameter Tuning

To further improve our randomForest model we will perform some hyperparameter tuning, specifically ntree and nodesize. Using the same cross-validation method to compare the models.

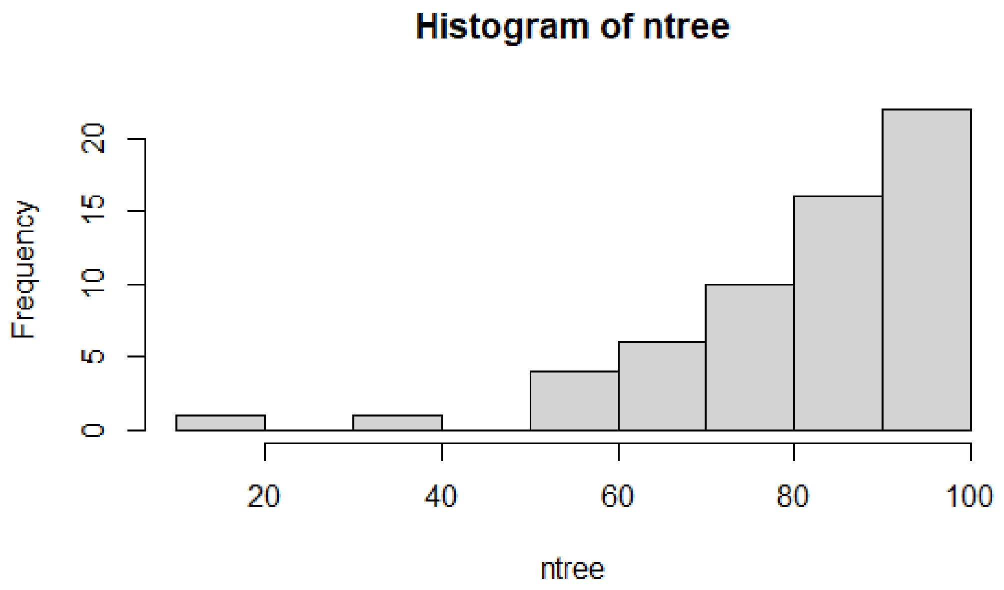
Note: The data collection loops below has been disabled with # as it takes around 2 hours to complete.

In [17]:

```
tree_tuner <- function(some_number) {
  df_train <- train %>% filter(train$subset == some_number)
  df_test <- train %>% filter(train$subset != some_number)
  best_score<- 1
  for (i in 1:100){
  model_forest <- randomForest(as.factor(status_group) ~
                               + gps_height  + date_recorded
                               + longitude + latitude + management
                               + extraction_type_group
                               + water_quality + quantity + source
                               + waterpoint_type ,
                               data = df_train,
                               ntree = i, nodesize = 2)
  if (sum(predict(model_forest, df_test)!=df_test$status_group)/nrow(df_test)<best_score
){
    best_score <- sum(predict(model_forest, df_test)!=df_test$status_group)/nrow(df_test
)
    best_para <- i
  }}
  return(best_para)
}

node_tuner <- function(some_number) {
  df_train <- train %>% filter(train$subset == some_number)
  df_test <- train %>% filter(train$subset != some_number)
  best_score<- 1
  for (i in 1:10){
    model_forest <- randomForest(as.factor(status_group) ~
                                 + gps_height  + date_recorded
                               + longitude + latitude + management
                               + extraction_type_group
                               + water_quality + quantity + source
                               + waterpoint_type ,
                               data = df_train,
                               ntree = 80, nodesize = i)
    if (sum(predict(model_forest, df_test)!=df_test$status_group)/nrow(df_test)<best_sco
re){
      best_score <- sum(predict(model_forest, df_test)!=df_test$status_group)/nrow(df_te
st)
      best_para <- i
    }}
  return(best_para)
}

#list_ntree <- c()
#for (x in 1:6){
  #curlist<- lapply(1:10, tree_tuner)
  #list_ntree <- c(list_ntree, curlist)
#}

#list_node<- c()
#for (x in 1:6){
  #curlist<- lapply(1:10, tree_tuner)
  #list_node <- c(list_node, curlist)
#}
```

For this process we applied the same cross-validation technique earlier, for each sub-sample, we run the model and assess it 100 times varying the ntree param from 1-100, each sub-sample(for which there're 10), we obtain 1 best-performing n-tree parameter, and we do this for 6 trials which gives us 60 values of best n-tree parameters. For node-size we also obtained 60 values however, we vary the parameter from 1-10.

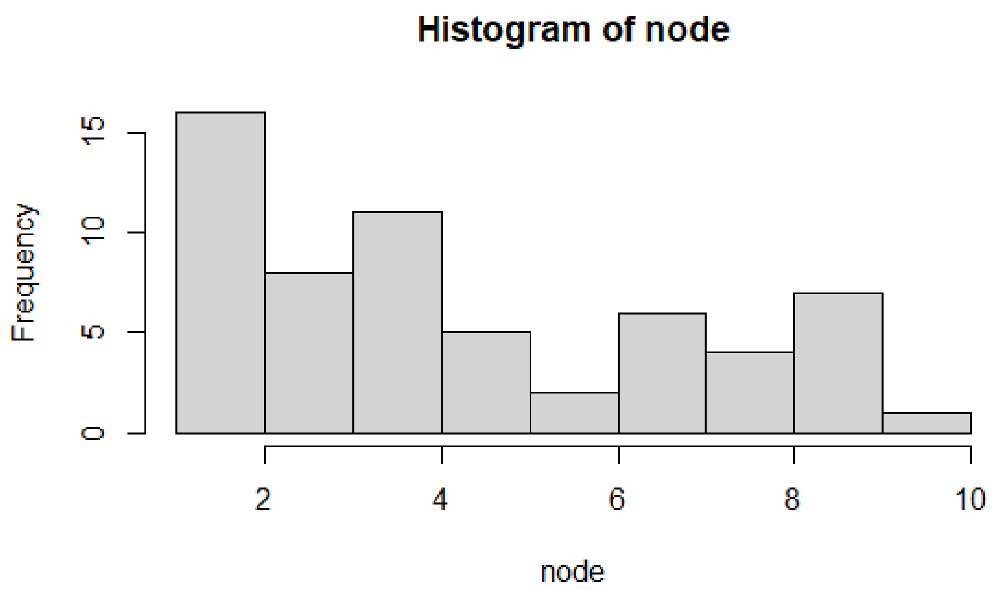
Top-performing ntree values



In [18]:

```
#>summary(ntree)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 19.00   73.00   85.50   82.18   95.25  100.00
```

Top-performing nodesize values



In [19]:

```
#>summary(node)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   1.000   2.000   4.000   4.633   7.000  10.000
```

From the mean values we determined the best ntree parameter to be 82, and the best nodesize to be 5.

Revisiting feature selection

We also revisited the feature selection section. Using Cross-Validation(CV) we "played around" with the feature set and see how they would improve our CV accuracy. Due to the randomness of randomForest, the change in accuracy after removing or adding features to the set is not always apparent, as such this was a long and time-consuming section, as well as being rather vague, some features had a consistent improvement while others seem to not change or slightly decrease the accuracy. We did this section somewhat manually, as we wanted our feature set to also make sense intuitively. As well as the fact that we decided against evaluating each feature alone and picking the best performers, as some features have synergy together(having both increases the accuracy) and some other combinations have antagonistic behavior(having both decreases the accuracy)

Below you can find the code that we used, this would normally in another loop and we would do this 6-10 times and average out the averages. This code is very similar to the ones we used early, to vary the features we manually added and subtracted from the function.

In [20]:

```
# testing k-fold cross validation here
df$random <- runif(nrow(df), min=1, max=60000)
df$subset <-  ntile(df$random, 10)

library(randomForest)

rf_model_1 <- function(some_number) {
  #sampling subsets for training and testing dfs
  df_train <- df %>% filter(df$subset == some_number)
  df_test <- df %>% filter(df$subset != some_number)

  model_forest <- randomForest(as.factor(status_group) ~
                                 + amount_tsh + gps_height
                                + longitude + latitude
                               + lga + population + construction_year
                               + management + extraction_type
                               + quality_group + quantity + source
                               + waterpoint_type+ m_year + payment,
                               data = df_train,
                               ntree = 82, nodesize = 5)

  df_test$y_pred <- predict(model_forest, df_test)
  error <- sum(df_test$y_pred!=df_test$status_group)/nrow(df_test)

  results <- list("df_test" = df_test , "model_forest" = model_forest, "error" = error)

  return(c(results, model_forest))
}

allerrors=c()

list_results <- lapply(1:10, rf_model_1) #Generate data


for (i in 1:10){
  allerrors <- c(allerrors, list_results[[i]][3]$error)
}
print("Average random forest error rate")
print(mean(allerrors))
```

```
[1] "Average random forest error rate"
[1] 0.2415881
```

For our final pick of model, we will use Random Forest on the whole dataset with the following features: ***amount_tsh, gps_height, longitude, latitude, lga, population, construction_year, management, extraction_type, quality_group, quantity, source, waterpoint_type, m_year, payment.***

3. Alternate learning procedure

We chose here to implement a Gradient Boosting Tree. The Gradient Boosting Tree is a technique used to produce a prediction model with decisions tree. They are an ensemble of decision tree models, which mean that

the gradient boosting tree is a set of decisions trees that perform the prediction together. The gradient boosting tree consist on the fact that each one of the tree will learn the difference from the prediction of the other previous tree and from the real value. Which mean that the final prediction will be the addition of the prediction from all the other trees. As the running time of the gradient boosting tree is very long we need to, as we did for the previous model, sample a subset of data for a sub-training dataset, and sample another subset for the testing datasets. We calculated the differences of predictions of the test set and averaged out the errors.

Note: The training code below takes a very long time(60 minutes at least) Please do not run it if you don't want to wait.

In [13]:

```
library(Hmisc)
library(tidyverse)
library(dplyr)
library(caret)
library(xgboost)
# running time is extremely slow so we will restrict our sample keep var with
# common mistakes : "factor quantity has new levels unknown"

df_gbt <- df %>% filter(df$construction_year!=0)
df_gbt <- df_gbt %>% filter(df_gbt$population>1)
df_gbt <- df_gbt %>% filter(df_gbt$amount_tsh>0)

# creating a new variable in the gbt data : random_100
df_gbt$random_100 <- runif(nrow(df_gbt), min=1, max=nrow(df_gbt))
# creating a new variable from the random_100 in the gbt data : subset_100
df_gbt$subset_100 <-  ntile(df_gbt$random_100, 100)

# filtering the variable subset_100 to keep only a percentage of it
df_gbt <- df_gbt %>% filter(df_gbt$subset_100 < 6)
nrow(df_gbt)

# creating a new variable in the gbt data : random
df_gbt$random <- runif(nrow(df_gbt), min=1, max=nrow(df_gbt))
# creating a new variable from the random in the gbt data : subset_100
#  which will be a part of the random  variable
df_gbt$subset <-  ntile(df_gbt$random, 10)

table(df_gbt$subset)

gbt_model_1 <- function(some_number) {
  df_train <- df_gbt %>% filter(df_gbt$subset == some_number)
  df_test <- df_gbt %>% filter(df_gbt$subset != some_number)

  model_gbt <- train(as.factor(status_group) ~
                      + gps_height + population
                     + construction_year + longitude + latitude,
                     data = df_train,
                     method = "xgbTree")

  df_test$y_pred <- predict(model_gbt, df_test)

  error <- sum(df_test$y_pred!=df_test$status_group)/nrow(df_test)

  results <- list("df_test" = df_test , "model_gbt" = model_gbt, "error" = error)

  return(results)
}

# generation of the data
list_results_gbt1 <- lapply(1:10, gbt_model_1)
# printing the result and keeping each result in a vector to compute the mean
results_v = c()
for (i in 1:10){
  results_v[i] <- list_results_gbt1[[i]][3]$error
}
```

```
error_mean_gbtmodel1 = mean(results_v)


print('Average of gradient boosting tre model error rate : ')
print(error_mean_gbtmodel1)
```

```
Attaching package: 'caret'


The following object is masked from 'package:purrr':

    lift


The following object is masked from 'package:survival':

    cluster



Attaching package: 'xgboost'


The following object is masked from 'package:dplyr':

    slice
```

735

```
 1  2  3  4  5  6  7  8  9 10
74 74 74 74 74 73 73 73 73 73
```

```
[1] "Average of gradient boosting tre model error rate : "
[1] 0.3318058
```

Based on the CV error score, as well as the absurdly long relative training time, we won't use this learning method over randomForest, as its default error is still worse than randomForest's default.

4. Final Drivendata Prediction

Final Model

In [29]:

```
final_forest <- randomForest(as.factor(status_group) ~
                                 + amount_tsh + gps_height
                                + longitude + latitude
                               + lga + population + construction_year
                               + management + extraction_type
                               + quality_group + quantity + source
                               + waterpoint_type+ m_year + payment,
                               data = df,
                               ntree = 82, nodesize = 5)
```

In [30]:

```
# predicting test set with model
forest_pred_test <- predict(final_forest, test_values)

# create submission data frame
submission <- data.frame(test_values$id)
submission$status_group <- forest_pred_test
names(submission)[1] <- "id"

# printing submission to csv
write.csv(submission, file = "submission.csv", row.names = FALSE)
```

Due to the random nature of random Forest, we submitted 3 times. The scores we got are the following:

0.8122	hhoang	2021-05-18 19:20:23 UTC
0.8143	hhoang	2021-05-18 19:20:54 UTC
0.8130	hhoang	2021-05-18 19:21:31 UTC

The submission csv files are included with this project as submission1.csv, submission2.csv, and submission3.csv

In [ ]:

```

```