# Water Pumps

*Steven Gusenius, Zuber Saiyed, Margarita Linets*

## About the Dataset

Using data from Taarifa and the Tanzanian Ministry of Water, we set out to predict what could impact the state of water pumps. A smart understanding of which waterpoints will fail can improve maintenance operations and ensure that clean, potable water is available to communities across Tanzania. More information about the challenge and the dataset can be found here - https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/page/23/

### MORE METADATA TO FOLLOW

```
#Libraries
library_list = list('ggplot2','glmnet','ggmap','reshape2', 'tree', 'stringr')
lapply(library_list, require, character.only = TRUE)
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
##
## [[6]]
## [1] TRUE
```

## Import Data

```
# Define train_values_url
train_values_url <- "http://s3.amazonaws.com/drivendata/data/7/public/4910797b-ee55-40a7-8668-10efd5c1b
WaterPumps_value = read.csv(train_values_url, header=TRUE, stringsAsFactors = FALSE)
# Define train_labels_url
train_labels_url <- "http://s3.amazonaws.com/drivendata/data/7/public/0bf8bc6e-30d0-4c50-956a-603fc693d
WaterPumps_label = read.csv(train_labels_url, header=TRUE, stringsAsFactors = TRUE)
# Define test_values_url
test_values_url <- "http://s3.amazonaws.com/drivendata/data/7/public/702ddfc5-68cd-4d1d-a0de-f5f566f76d
```

## Data Cleaning:

```
#Merge into a single datset
WaterPumps = merge(WaterPumps_value, WaterPumps_label, by='id')
```

```r
#Convert all character variables to upper case and remove non alpha numeric variables
charcols = which(sapply(WaterPumps,is.character))
WaterPumps[,charcols] <- sapply(WaterPumps[,charcols], toupper)
WaterPumps[,charcols] <- sapply(WaterPumps[, charcols], function(x) gsub("[^[:alnum:]=\\.]",'',x))

#Drop id variables - it is not a meanigful predict, just a record counter
WaterPumps = WaterPumps[, -which(names(WaterPumps) %in% c('id','recorded_by'))]
#Create New Variables:
WaterPumps$WellAge = as.numeric(str_sub(WaterPumps$date_recorded,1,4))-WaterPumps$construction_year
# Remove all WellAges that are negative or where Construction Year is missing
WaterPumps$WellAge = ifelse(WaterPumps$WellAge>100|WaterPumps$WellAge<0, NA, WaterPumps$WellAge)
#Flag all instances where installer differs from funder
WaterPumps$funderinstaller <- ifelse(WaterPumps$funder==WaterPumps$installer, TRUE,FALSE)
```

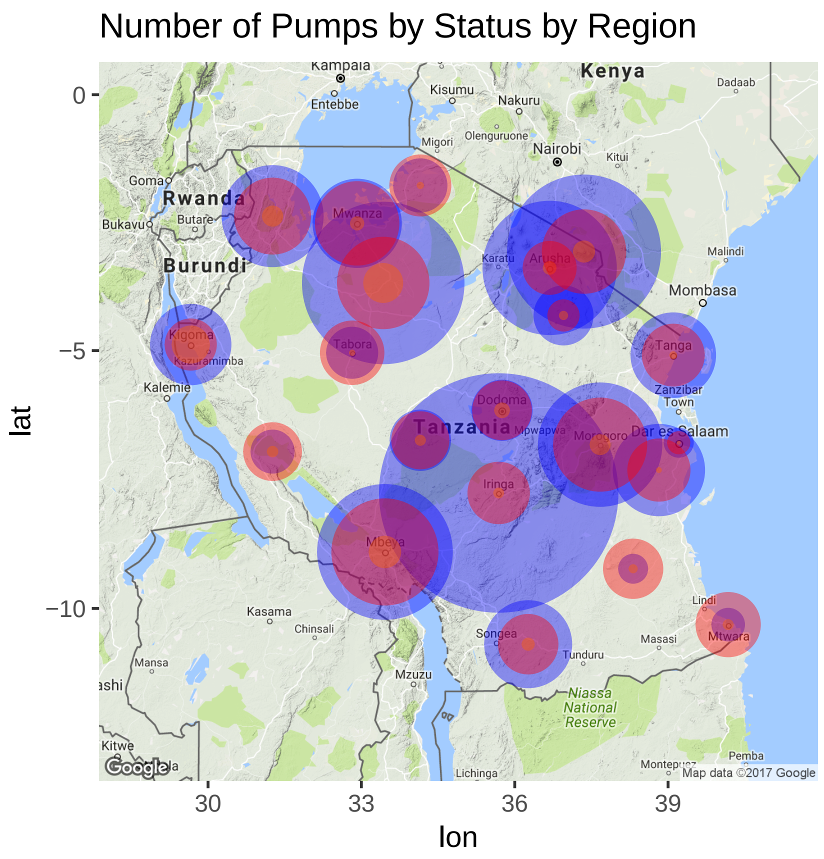
```r
# Clean Data: Filter out variables with more than 25 factors - model matrix becomes too large
KeepVars = setNames(data.frame(sapply(WaterPumps[,which(sapply(WaterPumps, is.character))], function(x)
                    c('factorlevels'))
KeepVars$vars = rownames(KeepVars)
WaterPumps = WaterPumps[,-which(names(WaterPumps) %in% KeepVars$vars[which(KeepVars$factorlevels==FALSE
```

## Exploratory Analysis: Visualizations *ADD VISUALIZATIONS YOU LIKE TO THIS SECTION - WE WILL SELECT THE ONES WE LIKE AT THE END*

```r
## Aggregate waterpump count by region
regional_geocodes = cbind.data.frame(data.frame("region" = unique(WaterPumps$region)),
                                setNames(data.frame(t(data.frame(lapply(lapply(unique(WaterPumps$region
                                 function(x) as.numeric(geocode(x)))))), c('longitude','latitude')))

#Drop Row Names
rownames(regional_geocodes) <- c()
#Aggregate Pump Kinds by Region
PumpsByRegion=dcast(WaterPumps, region~status_group, fun=length, value.var = 'status_group')
PumpsByRegion = merge(PumpsByRegion, regional_geocodes, id='region')
colnames(PumpsByRegion)=c("region", "functional","needsrepair", "nonfunctional","longitude","latitude" 

Tanzania = as.numeric(geocode("Tanzania"))
Tanzania_Map = ggmap(get_googlemap(center=Tanzania, scale=2, zoom=6))+
  geom_point(aes(x=longitude, y=latitude), data=PumpsByRegion, col='blue', alpha=0.4,
             size=PumpsByRegion$functional*0.01)+
  geom_point(aes(x=longitude, y=latitude), data=PumpsByRegion, col='orange', alpha=0.4,
             size=PumpsByRegion$needsrepair*0.01)+
  geom_point(aes(x=longitude, y=latitude), data=PumpsByRegion, col='red', alpha=0.4,
             size=PumpsByRegion$nonfunctional*0.01)+
  ggtitle('Number of Pumps by Status by Region')
Tanzania_Map
```



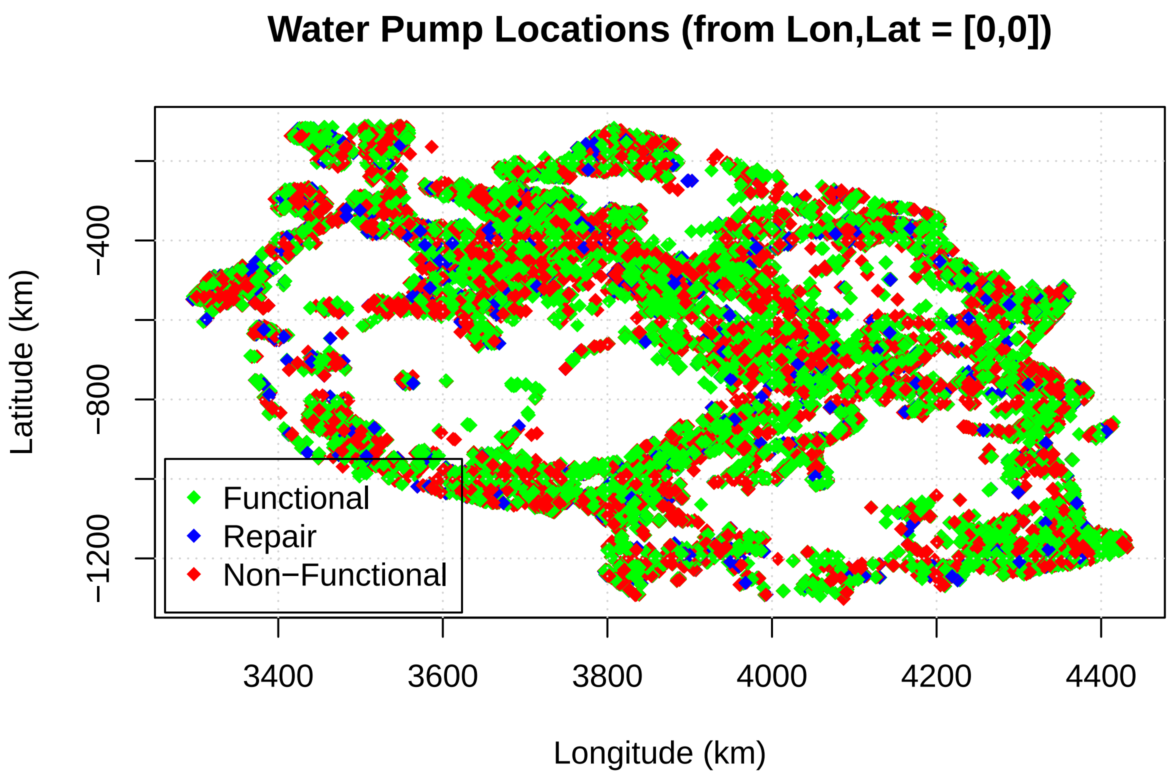
```
# Water Pump Position Plot

# Convert Latitude/Longitude to kilometers.
#  - Use 1862 meters per nautical mile (nmi).
#  - Uses 60 nmi per degree of latitude
#  - Uses 60 nmi per degree of longitude (at the equator)
#  - Assumes spherical earth (for logitudes above and below the equator)

# Convert lat long to xy
x = 1.862 * 60 * cos(WaterPumps$latitude*pi/180)*(WaterPumps$longitude)
y = 1.862 * 60 *(WaterPumps$latitude)

# Get Water Pump Operational State
z = as.factor(WaterPumps_label[,2])

# Plot Water pump Locations, color coded by OPerational state.  Use [x>10] indexing to elimminate bogus
plot(x[x>10],y[x>10],col=c("green","blue","red")[z[x>10]], xlab = 'Longitude (km)', ylab = 'Latitude (k
grid()
legend('bottomleft',legend = c('Functional','Repair','Non-Functional'), pch=18, col = c('green','blue',
```



## Variable Selection

```
# Take Only Non-Missing Values
WaterPumps <- WaterPumps[complete.cases(WaterPumps),]

# Create Model Matrix
modelmatrix = model.matrix(status_group~., WaterPumps)[,-1]

# Train and Validation Split
set.seed(500)
size=round(nrow(WaterPumps)*0.7)
train = sample(nrow(WaterPumps), size=size,replace = FALSE)
valid = which(!seq(1, nrow(WaterPumps), 1) %in% train)

train.x = modelmatrix[train,]
valid.x = modelmatrix[valid,]
train.y = WaterPumps$status_group[train]
valid.y = WaterPumps$status_group[valid]
```

```
# LASSO
grid = 10^seq(10,-2, length=100)
set.seed(500)
lasso.mod = glmnet(train.x, train.y, alpha=1, lambda=grid, family = 'multinomial')

# Cross-Validate to Find Best Lambda
```

```r
lasso.cv.out = cv.glmnet(train.x,train.y,alpha=1,lambda=grid, family = 'multinomial')
minlambda = lasso.cv.out$lambda.min

# Random Forest
```

## Modeling

## Evaluation & Selection