Own Project Report: Predicting Organic Carbon in Soils

Marcus Schmidt

23/02/2021

(I) INTRODUCTION

(I a) Background, goal & data set

Soils are the basis of all agriculture. Organic carbon and its dynamics plays a large role carbon sequestration and therefore helps regulate our climate. The goal of this project is to predict soil organic carbon from a large set of European soil data. The data set is from the LUCAS initiative 2015 (https://esdac.jrc.ec.europa.eu/projects/lucas) and includes over 20,000 sampling points on land that is used in different ways.

(I b) Data set download

The data set was requested online from the LUCAS initiative and the author got permission to use it for this project. Out of the data set, I created a *.Rdata file to be downloaded here:

```
load(url("https://github.com/ms-soil/ds-submission-soil/raw/main/eusoil.Rdata"))
```

(I c) Data set structure

Let's take a peak at the datset, showing the variables included, the dimension of the data set and a first look at the numbers and their variable type:

```
names(eusoil) # variable names
```

```
##  [1] "Clay"        "Sand"        "Silt"        "pH.CaCl2."   "pH.H2O."
##  [6] "EC"          "OC"          "CaCO3"       "P"           "N"
## [11] "K"           "Elevation"   "Soil_Stones" "LC1"         "LC1_Desc"
```

```
dim(eusoil) # data set dimensions
```

```
## [1] 21859    15
```

```r
as_tibble(eusoil) # overview of data structure
```

```
## # A tibble: 21,859 x 15
##     Clay  Sand  Silt pH.CaCl2. pH.H2O.    EC    OC CaCO3     P     N     K
##    <int> <int> <int>     <dbl>   <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1    NA    NA    NA       3.9    3.91  44.2  25.5     0  42.9   2.8  24.6
##  2    NA    NA    NA       3.1    3.91  46.4 504.      0 165.   19.9 460.
##  3    NA    NA    NA       4.9    5.48  15.8  51.4     0  26.9   4.3 173.
##  4    NA    NA    NA       3      3.76  26.9 470.      0 103.   16.1 313
##  5    10    46    44       3.9    4.04  28.4  43.1     1   6.3   2.3  38.6
##  6    14    36    50       4.2    4.41  41.8  32.4     0   7.5   3.3  48
##  7    18    35    46       4.9    5.13  32    21.1     1  12.4   2.1  36
##  8    14    36    50       4      4.16  72.4  53.2     0  52.1   4.2 158.
##  9    19    48    34       3.7    3.87  11.6  16       1   3.7   1    24.4
## 10     8    71    20       4      3.99  22.2  16       1   8.3   1.1  30
## # ... with 21,849 more rows, and 4 more variables: Elevation <int>,
## #   Soil_Stones <int>, LC1 <chr>, LC1_Desc <chr>
```

(II) METHODS & ANALYSIS

(II a) Data preparation

Analysis was don with R version 4.0.3. Land use in the data set is not straightforward. It is decoded in the LC1-variable, so we detect the sting and use its explanation from the data-set documentation, which can be found at https://github.com/ms-soil/ds-submission-soil (data-info.pdf). After this we remove the LC1-variable. We further check whether there are any observations without organic carbon (OC) and take a look at how many observations we have for each land use. In this step, we finally factorize land use.

```r
eusoil <- eusoil %>% mutate(landuse = case_when(str_detect(LC1, "A") ~ "artif_land",
                                                str_detect(LC1, "B") ~ "cropland",
                                                str_detect(LC1, "C") ~ "woodland",
                                                str_detect(LC1, "D") ~ "shrubland",
                                                str_detect(LC1, "E") ~ "grassland",
                                                str_detect(LC1, "F") ~ "bare_land",
                                                str_detect(LC1, "G") ~ "water",
                                                str_detect(LC1, "H") ~ "wetland")) %>%
  select(-LC1)

table(is.na(eusoil$OC)) # organic carbon data is available in all observations
```

```
##
## FALSE
## 21859
```

```r
# distribution of landuses
table(eusoil$landuse)
```

```
##
## artif_land  bare_land   cropland  grassland  shrubland      water    wetland
##         50        604       8972       4751        846          6         49
##   woodland
##       6581
```

```r
# factorize landuse
eusoil <- eusoil %>% mutate(landuse = as.factor(landuse))
```

(II b) Variable selection

We want to use a complete data set for our predictions, so we exclude variables where we don't have all observation. We can see it with this code. It yields that we are missing the texture variables in over 80% of cases.

```r
# print availability for every variable
res <- for (i in 1:length(names(eusoil))) {
  print(table(!is.na(eusoil[,i])))
  print(names(eusoil[i]))
}
# texture (the silt, sand and clay variables) are only available in a fraction of cases
# see which fraction of the dataset misses texture
condition <- !is.na(eusoil$Clay)
table(condition)[[1]]/(table(condition)[[1]] + table(condition)[[2]])
# texture is missing in 80.5% of cases
```

Here is a bit of the result:

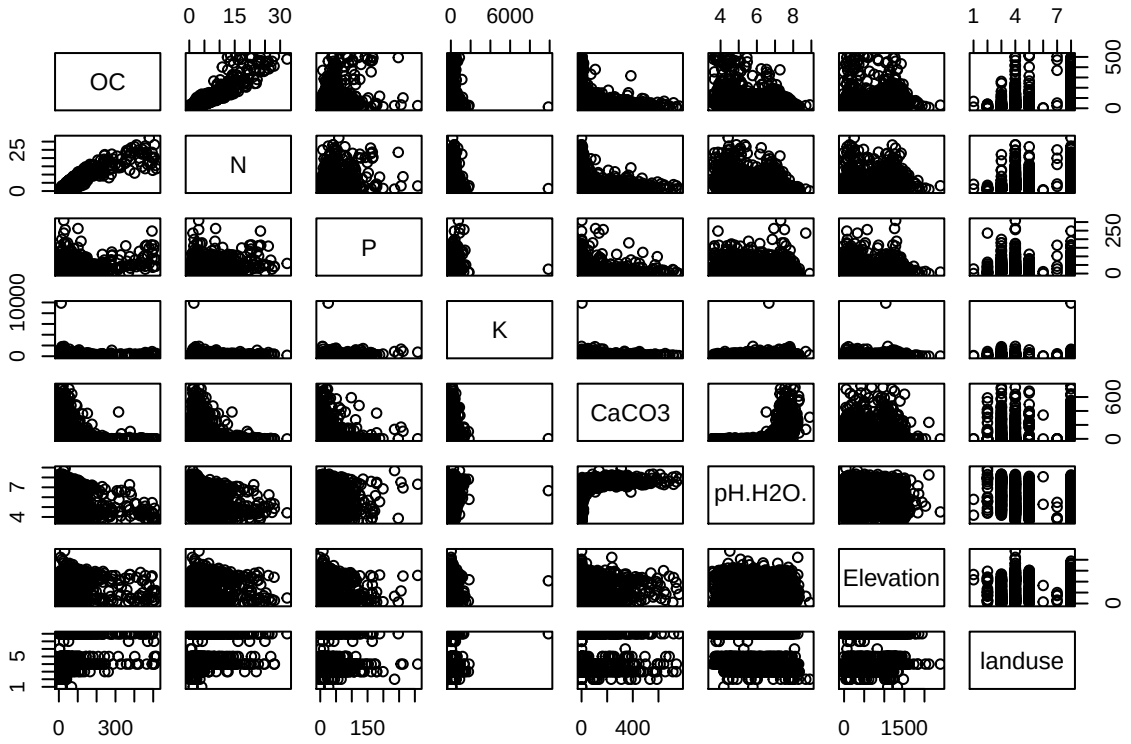
```r
# FALSE  TRUE
# 17599  4260
# [1] "Clay"
#
# FALSE  TRUE
# 17599  4260
# [1] "Sand"
#
# FALSE  TRUE
# 17599  4260
# [1] "Silt"
```

The texture variables clay, sand and silt are not often measured in this data set so we exclude them, as well as some other variables that are not commonly measured by soil scientists or not clearly defined:

```r
eusoil <- eusoil %>% select(-Clay, -Sand, -Silt, -EC, -pH.CaCl2., -Soil_Stones, -LC1_Desc) %>%
  select(OC, N, P, K, CaCO3, pH.H2O., Elevation, landuse)
```

What variables are left and how do they relate. We want to draw some conclusions of what to include by looking at a correlation plot.

```r
#### variable selection III: influential variables ####
names(eusoil)
## a) numerical variables that correlate with organic carbon but not with each other
relationships <- plot(eusoil[1:2000,]) # plotting the first 2000 observations
```

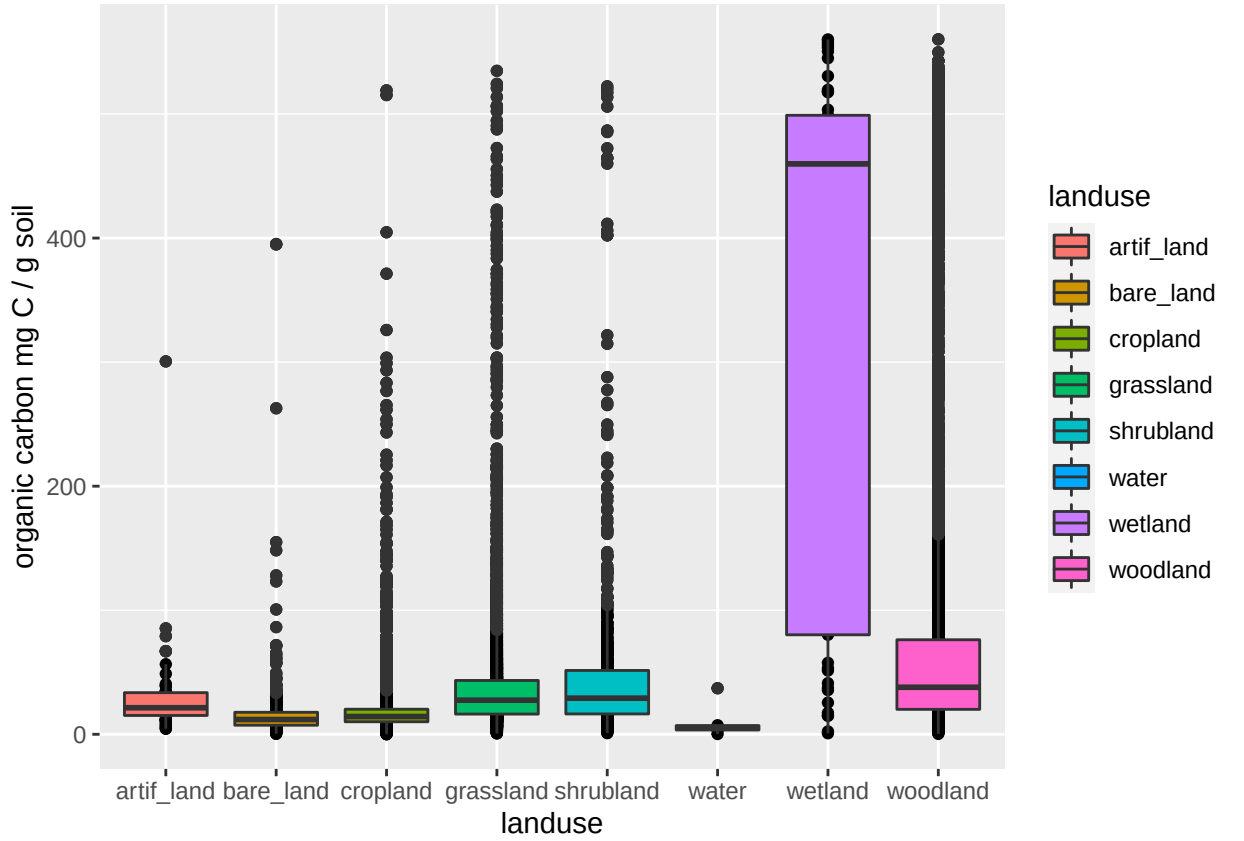



```
relationships
```

P (phosphorus), K (potassium) show no clear relationship to organic carbon so we will exclude them soon. CaCO3 and pH are co-dependend so it will be enough to include one of them. What remains at this point is N (Nitrogen), pH and elevation.

There is also a categorical variable, land use and want to see whether it influences organic carbon:

```
qplot(landuse, OC, data = eusoil, fill = landuse) + geom_boxplot() +
  ylab("organic carbon mg C / g soil")
```



Landuse plays a role so we keep it. Finally, we give out some easier-to-type variable names and select what we found important

```
eusoil <- eusoil %>% mutate(pH = pH.H2O., elev = Elevation) %>%
  select(OC, N, P, pH, elev, landuse)
```

(II c) Data set division into validation, train and test set

First, from the created data, a validation set of 10% is taken so we have a large proportion of the data for model training. The training set will be 80% and the test set 20% of the remaining data. We should choose a split that allows lots of training but still a representative test set. For reproducible results in drawing, we use the set.seed() function.

```
set.seed(1, sample.kind = "Rounding")
valindex <- createDataPartition(eusoil$OC, p = 0.1, list = F)
temp <- eusoil[-valindex,]
validation <- eusoil[valindex,]

set.seed(1, sample.kind = "Rounding")
testindex <- createDataPartition(temp$OC, p = 0.2, list = F)
trainset <- temp[-testindex,]
testset <- temp[testindex,]

# see how many observations are in each
nrow(trainset); nrow(testset); nrow(validation)
```

```
## [1] 15735
```

```
## [1] 3937
```

```
## [1] 2187
```

Over 2000 obervations for final validation set should be plenty.

(II d) RMSE function

A rooted mean square error (RMSE) function is used to evaluate the performance of the models that will be set up.

```
RMSE <- function(observed_values, predicted_values){
  sqrt(mean((observed_values - predicted_values)^2))
}
```

(II e) Reference model

A reference model can be useful to evaluate, how different the models perform from simply guessing, or taking the average of the observed carbon values.

```
#### guessing model ####
mu <- mean(trainset$OC)

rmse_mu <- RMSE(testset$OC, mu)
rmse_mu
```

```
## [1] 77.78776
```

```
# create an RMSE table to always add the RMSEs
rmses <- data.frame(model = "mean", rmse = rmse_mu)
rmses
```

```
##   model     rmse
## 1  mean 77.78776
```

So if we guess, we are typically ~78 mg C / g soil off. We will improve this prediction, first with linear models but also with the KNN and random forest algorithms.

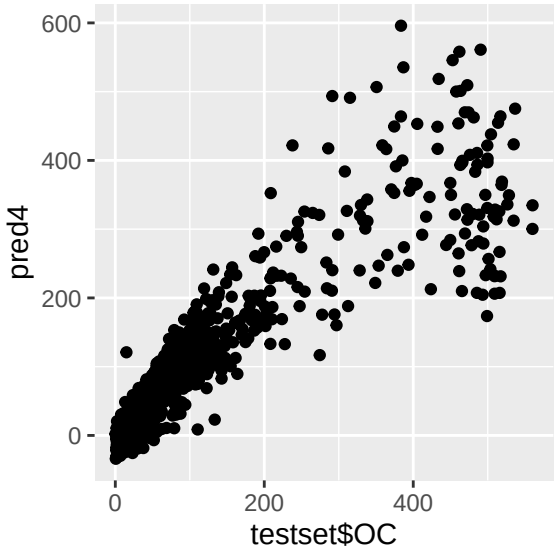
(II f) Linear models

In the following, I try different linear models, first with N (nitrogen), then adding pH, elevation and landuse. The full code is present in the R script, to not overwhelm the report, I show the model with all these variables and the table of how RMSE impoved step by step.

```
#### lm with N, pH, elevation, landuse ####

m_all <- lm(OC ~ ., data = trainset)

base <- data.frame(testset)

pred4 <- predict(m_all, base)
head(pred4)
```

```
##         7        12        16        28        29        38
##  12.67697  72.41147  55.78709 188.79738  66.74575 168.52116
```

```
qplot(testset$OC, pred4)
```




```
rmse4 <- RMSE(testset$OC, pred4)
rmse4
```

```
## [1] 30.81398
```

```
rmses <- rbind(rmses,
               data.frame(model = "lm with all variables", rmse = rmse4))
rmses
```

```
##                    model     rmse
## 1                   mean 77.78776
## 2             lm with N 33.24694
## 3         lm with N + pH 32.60644
## 4 lm with N + pH + elev. 32.54013
## 5  lm with all variables 30.81398
```

(II g) KNN models

Since all variables appeared to improve the model, I also include these in a KNN model, which looks multidimensionally at the nearest neighbors of an observation and predicts from there. A first try including elevation yields a less-than-optimal result (see RMSE table below), but excluding elevation worked well. I could not immediately detect the reason for this, but my conclusion is that not every variable improves every type of model in the same way. Generally, knn performed well after improving on the number of neighbors with a sapply() function.

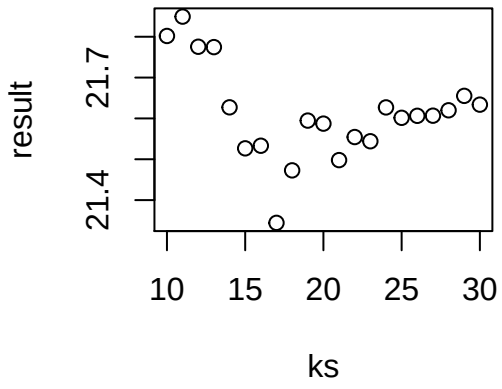
```
#### KNN [caret] all variables / excl. elevation ####
sqrt(nrow(trainset))
```

```
## [1] 125.4392
```

```
ks <- seq(10,30,1)
ks
```

```
##  [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

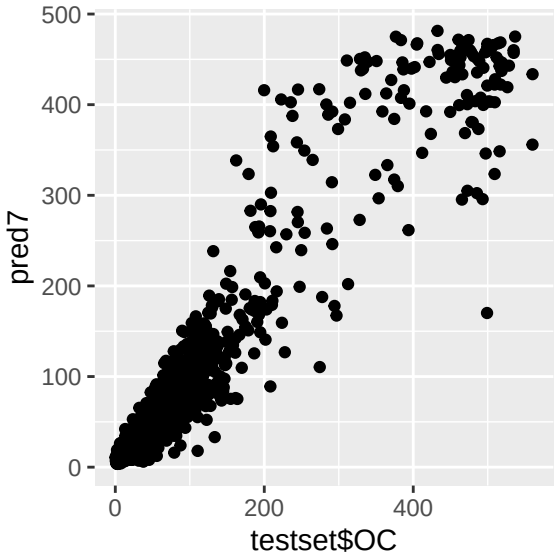
```
# checking which is the optimal k
result <- lapply(ks, function(i){
  fit_knn2 <- knnreg(OC ~ N + pH + landuse, data = trainset, k = i)
  pred <-predict(fit_knn2, base)
  RMSE(testset$OC, pred)
})

plot(ks, result)
```




```
best_k <- ks[which.min(data.frame(result)[1,])] #31
best_k # this is 17
```

```
## [1] 17
```

```
## using best k

fit_knn2 <- knnreg(OC ~ N + pH + landuse, data = trainset, k = best_k)
pred7 <-predict(fit_knn2, base)

qplot(testset$OC, pred7)
```




```
rmse7 <- RMSE(testset$OC, pred7)
rmse7
```

```
## [1] 21.34432
```

```
rmses <- rbind(rmses,
               data.frame(model = "knn excluding elev.", rmse = rmse7))
rmses
```

```
##                     model     rmse
## 1                    mean 77.78776
## 2              lm with N 33.24694
## 3          lm with N + pH 32.60644
## 4 lm with N + pH + elev. 32.54013
## 5  lm with all variables 30.81398
## 6 knn with all variables 50.09455
## 7    knn excluding elev. 21.34432
```

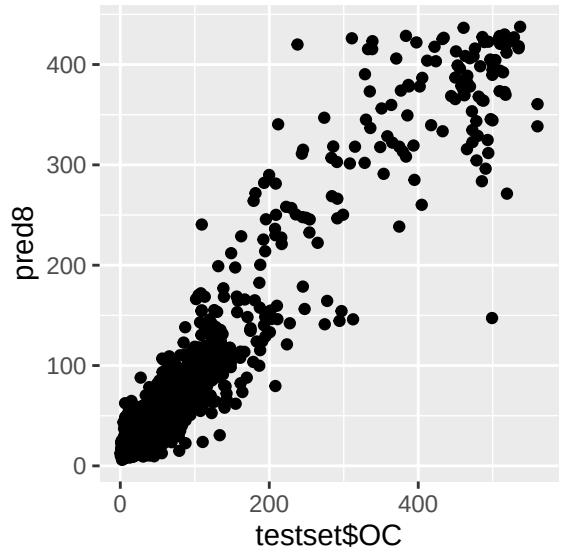
(II h) Random forest model

A random forest model consists of many models (trees) that are combined (to a forest) so the number of trees plays a role. The more trees the more exact the model usually is, but it then also takes more time to calculate. Here, I started with 10 trees, then tried 100, then 250 and then 500. The number of trees (ntree = x) of 250 appeard to yield a good compromise between computing time and performance. The best model also includes all variables, unlike KNN did.

```r
set.seed(1)

fit_rf <- randomForest(
  OC ~ ., data = trainset,
  ntree = 250
)

# going from 10 to 100 trees in the only-N model improved the RMSE by ~0.4
# and another 0.2 for 250 trees, hardly then anymore for 500 trees

pred8 <- predict(fit_rf, base)
head(pred8)
```

```
##         7        12        16        28        29        38
##  21.22866  50.31972  65.58160 165.00924  46.67700 148.49321
```

```r
qplot(testset$OC, pred8)
```




```r
rmse8 <- RMSE(testset$OC, pred8)
rmse8
```

```
## [1] 23.14774
```

```r
rmses <- rbind(rmses,
               data.frame(model = "rf with all variables", rmse = rmse8))
rmses
```

```
##                    model     rmse
## 1                   mean 77.78776
## 2              lm with N 33.24694
## 3          lm with N + pH 32.60644
## 4 lm with N + pH + elev. 32.54013
```

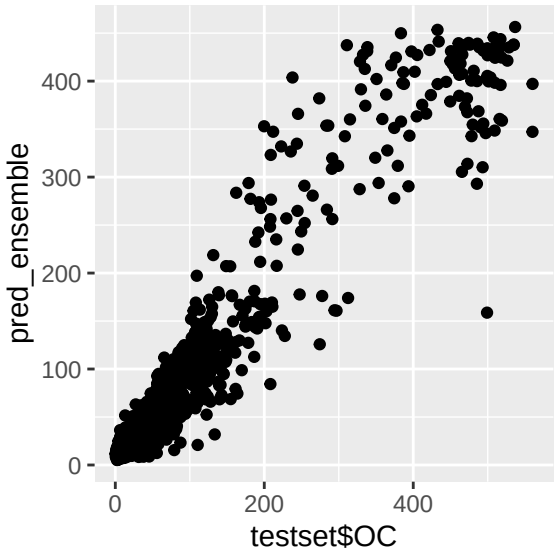
```
## 5  lm with all variables 30.81398
## 6 knn with all variables 50.09455
## 7    knn excluding elev. 21.34432
## 8  rf with all variables 23.14774
```

```
## in this case all variables improve the model
```

(II i) Ensemble model (KNN and random forest)

The best two models were combined into an ensemble, where the mean is taken from the prediction of both. This works well as it improves the RMSE again:

```
pred_ensemble <- (pred7 + pred8)/2 # knn & random forest

qplot(testset$OC, pred_ensemble)
```




```
rmse_ens1 <- RMSE(testset$OC, pred_ensemble)
rmse_ens1
```

```
## [1] 20.86007
```

```
rmses <- rbind(rmses,
               data.frame(model = "ensemble of knn & rf", rmse = rmse_ens1))
```

(III) RESULTS

(III a) RMSEs of tested models

As a result, we have an order of performance from best to least performing model that is Ensemble > KNN > Random forest > linear model.

Here's the overview table:

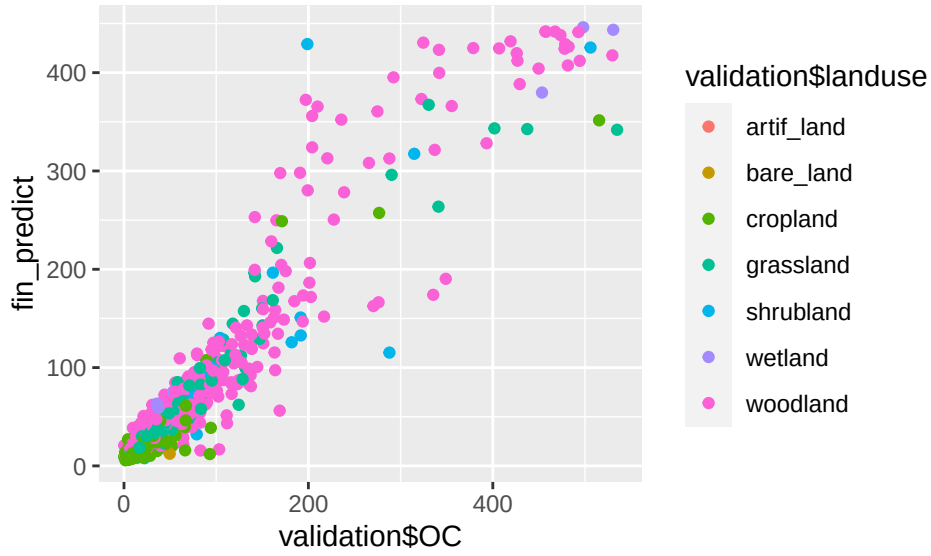
```
rmses
```

```
##                     model     rmse
## 1                    mean 77.78776
## 2              lm with N 33.24694
## 3          lm with N + pH 32.60644
## 4 lm with N + pH + elev. 32.54013
## 5  lm with all variables 30.81398
## 6 knn with all variables 50.09455
## 7    knn excluding elev. 21.34432
## 8  rf with all variables 23.14774
## 9   ensemble of knn & rf 20.86007
```

(III b) Evaluation of best model on validation set

I now predict organic Carbon in the validation set with the best two models: KNN without elevation and Random forest with all variables.

```
# incorporate the mean prediction of knn and rf since they are the best ones

fin_predict_a <- predict(fit_knn2, validation)
fin_predict_b <- predict(fit_rf, validation)

fin_predict <- (fin_predict_a + fin_predict_b) / 2

qplot(validation$OC, fin_predict, col = validation$landuse)
```



```r
fin_rmse <- RMSE(validation$OC, fin_predict)

rmses <- rbind(rmses,
               data.frame(model = "FINAL RMSE OVERALL", rmse = fin_rmse))
```

```r
fin_rmse
```

This yields the final RMSE, our typical prediction error, which is:

```
## [1] 19.01878
```

(III b) Model performance in the most common range of organic carbon

(IV) CONCLUSION

(IV a) Lessons learned

(IV b) Limitations

(IV c) Final thoughts