R Notebook

Code

Hide

```
library(e1071)
library(MASS)
df <- read.csv("CO2_cleaned_restructured.csv")
str(df)
```

```
'data.frame':   48509 obs. of  14 variables:
 $ X                   : int  0 1 2 3 4 5 6 7 8 9 ...
 $ Country             : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
 $ Code                : chr  "AF" "AF" "AF" "AF" ...
 $ Calling.Code        : chr  "93" "93" "93" "93" ...
 $ Year                : int  1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 ...
 $ Population.2022.    : int  41128771 41128771 41128771 41128771 41128771 41128771 41128771 4112
8771 41128771 41128771 ...
 $ Area                : int  652230 652230 652230 652230 652230 652230 652230 652230 652230 6522
30 ...
 $ X..of.World         : chr  "0.40%" "0.40%" "0.40%" "0.40%" ...
 $ Density.km2.        : chr  "63/km²" "63/km²" "63/km²" "63/km²" ...
 $ CO2.emission..Tons.: num  0 0 0 0 0 0 0 0 0 0 ...
 $ Value               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Class               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Class.Modifier      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Rounded.Class       : int  0 0 0 0 0 0 0 0 0 0 ...
```

Remove columns.

Hide

```
df <- df[,c("Country","Year","Population.2022.","Area","Density.km2.","CO2.emission..Tons.")]
str(df)
```

```
'data.frame':   48509 obs. of  6 variables:
 $ Country             : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
 $ Year                : int  1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 ...
 $ Population.2022.    : int  41128771 41128771 41128771 41128771 41128771 41128771 41128771 4112
8771 41128771 41128771 ...
 $ Area                : int  652230 652230 652230 652230 652230 652230 652230 652230 652230 6522
30 ...
 $ Density.km2.        : chr  "63/km²" "63/km²" "63/km²" "63/km²" ...
 $ CO2.emission..Tons.: num  0 0 0 0 0 0 0 0 0 0 ...
```

Format columns.

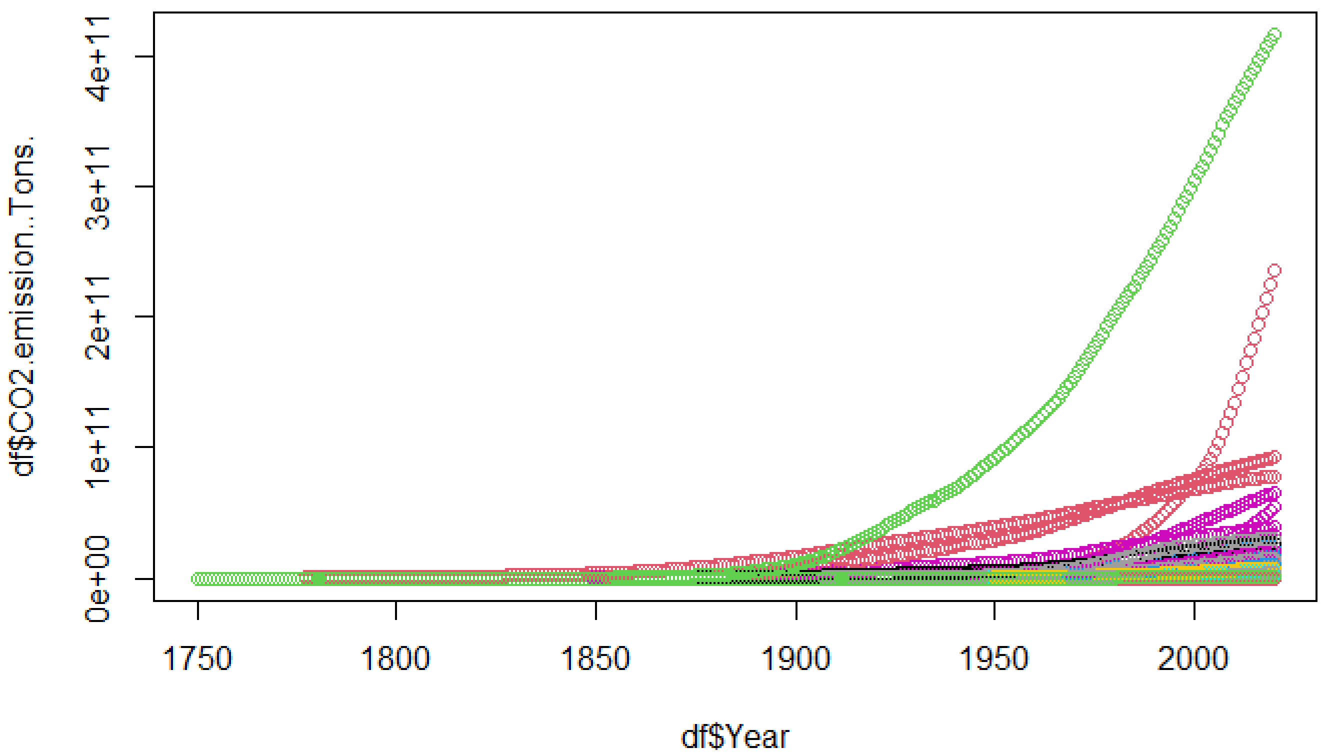
Hide

```
df$Density.km2. <- as.numeric(gsub("([0-9]+).*$", "\\1", df$Density.km2.))
df$Country <- factor(df$Country)
```

Plot emission as a function of year colored by country.

Hide

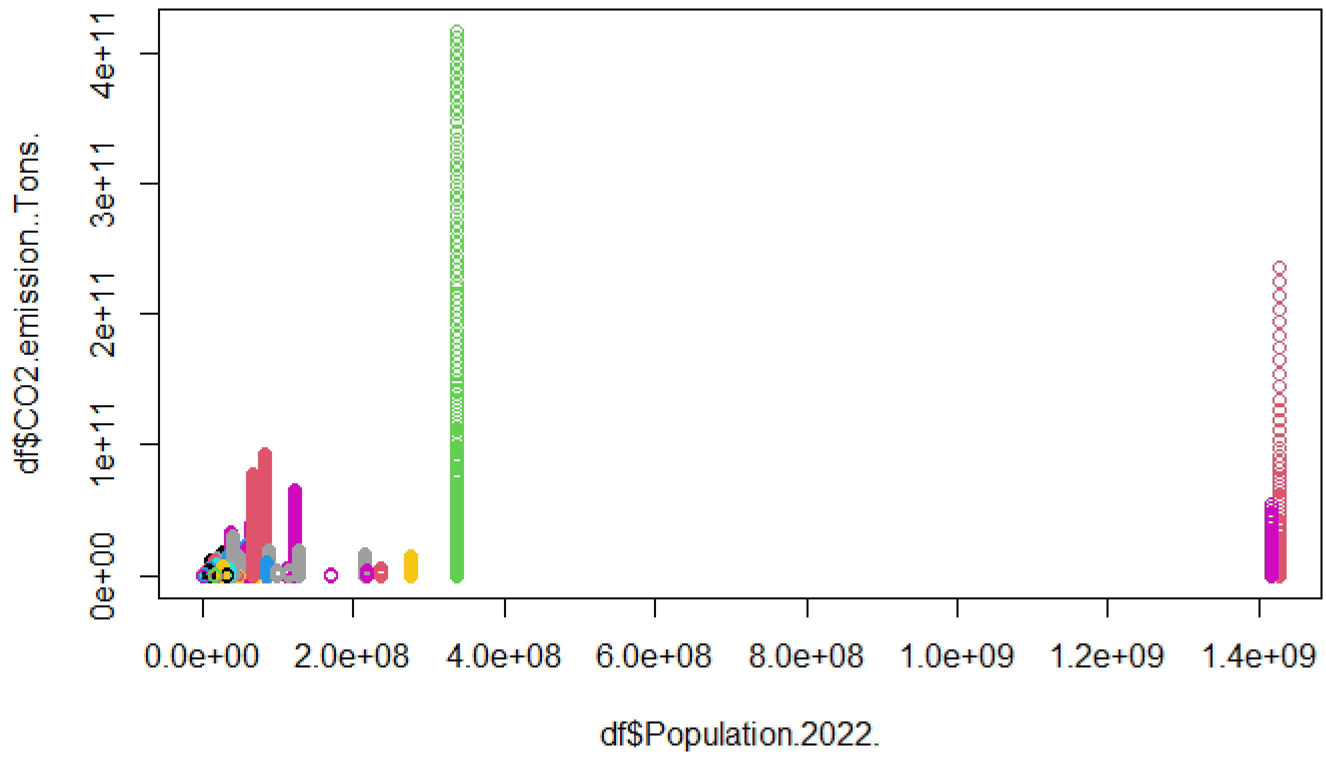
```
plot(df$Year,df$CO2.emission..Tons.,col=df$Country)
```



Plot emission as a function of population.

Hide

```
plot(df$Population.2022.,df$CO2.emission..Tons.,col=df$Country)
```



Training and testing data.

Hide

```
set.seed(1234)
spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(df),
                nrow(df)*cumsum(c(0,spec)), labels=names(spec)))
train <- df[i=="train",]
test <- df[i=="test",]
vald <- df[i=="validate",]
```

Run linear regression for baseline.

Hide

```
lm1 <- lm(CO2.emission..Tons.~.,data=train)
pred <- predict(lm1, newdata=test)
```

```
Warning: prediction from a rank-deficient fit may be misleading
```

Hide

```
cor_lm1 <- cor(pred,test$CO2.emission..Tons.)
mse_lm1 <- mean((pred-test$CO2.emission..Tons.)^2)
print(paste("Correlation: ", cor_lm1))
```

```
[1] "Correlation:  0.539501910954775"
```

Hide

```
print(paste("MSE: ", mse_lm1))
```

```
[1] "MSE:  97277678612296318986"
```

Run svm.

Hide

```
svm1 <- svm(CO2.emission..Tons.~., data=train, kernel="linear", cost=10, scale=TRUE)
summary(svm1)
```

```


Call:
svm(formula = CO2.emission..Tons. ~ ., data = train, kernel = "linear", cost = 10, scale = TRUE)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.005464481
    epsilon:  0.1


Number of Support Vectors:  3501
```

Evaluate svm.

Hide

```
pred <- predict(svm1, newdata=test)
cor_svm1 <- cor(pred, test$CO2.emission..Tons.)
mse_svm1 <- mean((pred-test$CO2.emission..Tons.)^2)
print(paste("Correlation: ", cor_svm1))
```

```
[1] "Correlation:  0.291806441457665"
```

Hide

```
print(paste("MSE: ", mse_svm1))
```

```
[1] "MSE:  1.30297323309213e+20"
```

Try different costs.

Hide

```
tune_svm1 <- tune(svm, CO2.emission..Tons.~., data=vald, kernel="linear", ranges=list(cost=c(0.0
01,0.01,0.1,1,5,10,100)))
```

```
WARNING: reaching max number of iterations

WARNING: reaching max number of iterations
```

Hide

```
summary(tune_svm1)
```

```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
```

cost <dbl>
100

1 row

```
- best performance: 1.057287e+20

- Detailed performance results:
```

cost <dbl>	error <dbl>	dispersion <dbl>
1e-03	1.118102e+20	6.510760e+19
1e-02	1.092300e+20	6.472186e+19
1e-01	1.069315e+20	6.454090e+19
1e+00	1.058173e+20	6.456601e+19
5e+00	1.057293e+20	6.459578e+19
1e+01	1.057299e+20	6.459608e+19
1e+02	1.057287e+20	6.459504e+19

7 rows

```
NA
```

Try with polynomial.

Hide

```
tune_svm2 <- tune(svm, CO2.emission..Tons.~., data=vald, kernel="polynomial", ranges=list(cost=c
(0.001,0.01,0.1,1,5,10,100)))
summary(tune_svm2)
```

```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
```

cost <dbl>
100

1 row

```
- best performance: 1.013922e+20

- Detailed performance results:
```

cost <dbl>	error <dbl>	dispersion <dbl>
1e-03	1.134375e+20	5.913644e+19
1e-02	1.134176e+20	5.912814e+19
1e-01	1.132162e+20	5.902890e+19
1e+00	1.121074e+20	5.852101e+19
5e+00	1.100234e+20	5.752605e+19
1e+01	1.088480e+20	5.703783e+19
1e+02	1.013922e+20	5.365820e+19

7 rows

```
NA
```

Try with radial.

Hide

```
tune_svm3 <- tune(svm, CO2.emission..Tons.~., data=vald, kernel="radial", ranges=list(cost=c(0.0
01,0.01,0.1,1,5,10,100)))
summary(tune_svm3)
```

```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
```

cost <dbl>
100

1 row

```
- best performance: 8.359985e+19

- Detailed performance results:
```

cost <dbl>	error <dbl>	dispersion <dbl>
1e-03	1.133624e+20	6.589259e+19
1e-02	1.129030e+20	6.577121e+19
1e-01	1.109067e+20	6.521043e+19
1e+00	1.060252e+20	6.344614e+19
5e+00	1.018952e+20	6.187804e+19
1e+01	9.951527e+19	6.095884e+19
1e+02	8.359985e+19	5.405124e+19

7 rows

```
NA
```

These algorithms are fairly slow. Each model takes about 10 minutes to calculate. This makes it difficult to make minor adjustments to the parameters for testing. It seems that radial with higher cost is the best. That is the only time the error dropped below 1e20, but only by a small amount. Making the cost even higher may reduce the error even more.