

# Untitled

## Linear analysis

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-2
##
## Loading required package: rpart
## Loading required package: lattice
## Loading required package: ggplot2
```

Additionally, we considered a standard *linear regression* model involving all predictors, as an alternative means to view the significance of each predictor. Note that in this context, performing *k-fold cross-validation* or *bootstrapping* wouldn't allow us to come up with an "averaged" subset, hence we performed an ordinary 80/20 splitting of the data into the testing and training as shown in the code below:

```
set.seed(1000)
test = sample(nrow(Crime), nrow(Crime)*.2)
excluded = c("crm rte", "crm rte_cat", "region_w_nw")
xs = Crime[-which(names(Crime) %in% excluded)]
ys = Crime[which(names(Crime) %in% c("crm rte"))]
xs_test = xs[test,]
xs_train = xs[-test,]
ys_test = ys[test,]
ys_train = ys[-test,]
p = dim(xs)[2]
```

We then run a simple linear fit with all predictors, in order to analyse the significance levels of the parameters, provided that the linear test itself has a significant  $R^2$  value.

```
lm.fit = lm(crm rte ~ ., data = Crime)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = crm rte ~ ., data = Crime)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.027793 -0.005201 -0.000603  0.004163  0.038831
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.115e-01  2.696e-02   4.136 4.03e-05 ***
## county       5.086e-06  6.285e-06   0.809 0.418738
## year        -1.463e-03  3.674e-04  -3.982 7.67e-05 ***
## prbarr       -3.334e-02  3.323e-03 -10.033 < 2e-16 ***
## prbconv      -2.062e-03  2.847e-04  -7.244 1.34e-12 ***
## prbpris       1.952e-03  4.264e-03   0.458 0.647334
```

```
## avgsen      -1.073e-04  1.355e-04  -0.792  0.428543
## polpc       1.725e+00  2.150e-01   8.021  5.51e-15 ***
## density     7.091e-03  5.542e-04  12.795  < 2e-16 ***
## taxp       1.577e-04  3.932e-05   4.011  6.81e-05 ***
## regionother 4.816e-03  1.010e-03   4.770  2.32e-06 ***
## regionwest -1.411e-03  1.235e-03  -1.143  0.253584
## smsayes     -3.429e-03  2.448e-03  -1.401  0.161817
## pctmin      1.326e-04  3.421e-05   3.876  0.000118 ***
## wcon        -6.811e-07  3.022e-06  -0.225  0.821741
## wtuc        -1.956e-07  1.330e-06  -0.147  0.883171
## wtrd         4.305e-06  4.261e-06   1.010  0.312703
## wfir        -1.014e-05  1.009e-05  -1.005  0.315466
## wser        -4.180e-06  3.529e-06  -1.184  0.236778
## wmf         -1.401e-06  5.970e-06  -0.235  0.814550
## wfed         4.413e-05  9.551e-06   4.620  4.70e-06 ***
## wsta        -6.671e-06  9.998e-06  -0.667  0.504907
## wloc         4.167e-05  1.788e-05   2.331  0.020066 *
## mix          4.689e-03  2.218e-03   2.114  0.034926 *
## pctymle      8.609e-02  1.611e-02   5.344  1.29e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.008666 on 601 degrees of freedom
## Multiple R-squared:  0.7544, Adjusted R-squared:  0.7445
## F-statistic: 76.9 on 24 and 601 DF,  p-value: < 2.2e-16
```

as the  $R^2$  value is sufficiently high (0.754353), we decided to perform `best subset selection` on the set of predictors. Although we are aware of the performance penalties of doing this for  $p = 23$ , the running times were considerably short and hence we decided to stick to this approach:

```
bestsubset=regsubsets(y ~ ., data = data.frame(y = ys_train, x = xs_train), nvmax = p)
```

After getting all best subsets with size  $k = 1 \dots p$ , we analysed both the *training RSE* and *testing RSE*:

```
set.seed(1000)
val.train.errors = rep(NA, p)
val.test.errors = rep(NA, p)
x_cols = colnames(xs, do.NULL = FALSE, prefix = "x.")
colnames(xs) <- paste("x", x_cols, sep = ".")
x_cols = colnames(xs)

for (i in 1:p) {
  coefi = coef(bestsubset, id = i)
  pred = as.matrix(xs_test[, x_cols %in% names(coefi)]) %*% coefi[names(coefi)
                                                                    %in% x_cols]
  val.test.errors[i] = mean((ys_test - pred)^2)
}
which.min(val.test.errors)
```

```
## [1] 8
```

```
val.test.errors[which.min(val.test.errors)]
```

```
## [1] 0.0001756529
```

```
for (i in 1:p) {
  coefi = coef(bestsuSubset, id = i)
  pred = as.matrix(xs_train[, x_cols %in% names(coefi)]) %*% coefi[names(coefi)
                                                                    %in% x_cols]
  val.train.errors[i] = mean((ys_train - pred)^2)
}
which.min(val.train.errors)
```

```
## [1] 8
```

```
val.train.errors[which.min(val.train.errors)]
```

```
## [1] 0.0001748673
```

The ratio between *testing RSE* and *training RSE* is very close to 1 (1.0044925) for the value that minimizes both *RSEs* (8). Consequently, we can conclude that the predictors yielded by the subset generated by *best subset selection* which minimizes both *RSEs* belong to a consistent model and, hence, can be used as a basis for non linear models. Nevertheless, we decided to run a linear fit with these predictors in order to check our conclusions:

```
lm.2.fit = lm(crmrte ~ prbarr + prbconv + polpc + density
              + as.factor(region) + pctmin + wfed + pctymle, data = Crime)
summary(lm.2.fit)
```

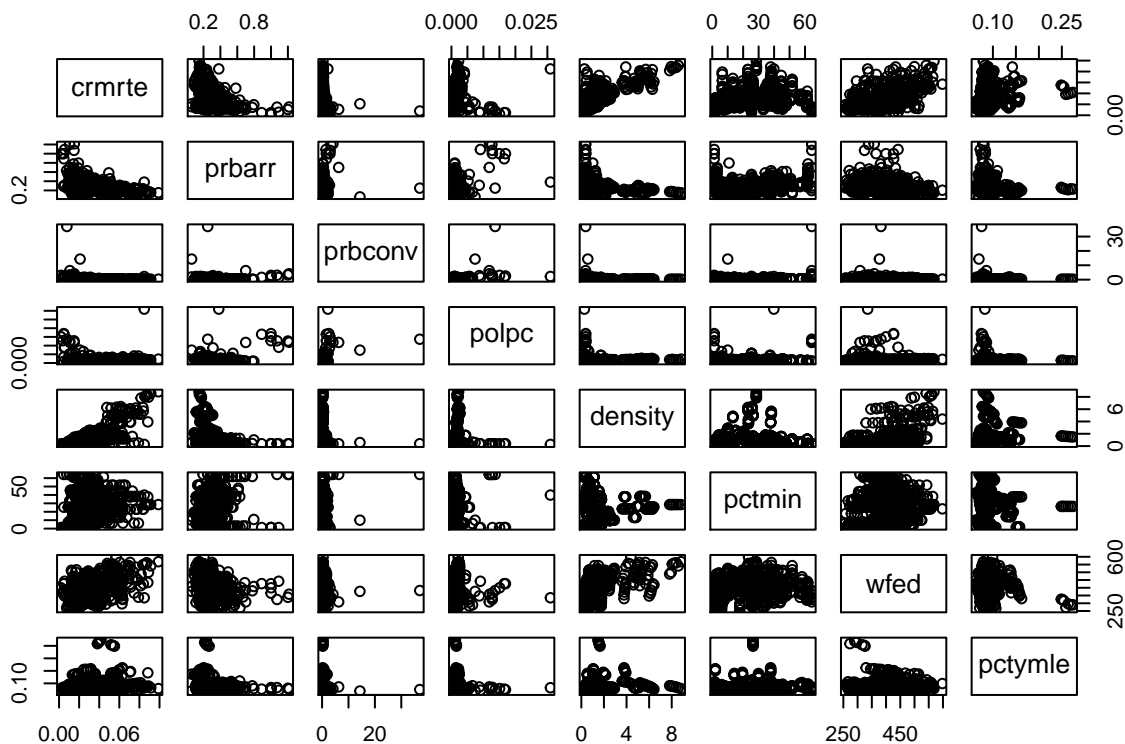
```
##
## Call:
## lm(formula = crmrte ~ prbarr + prbconv + polpc + density + as.factor(region) +
##      pctmin + wfed + pctymle, data = Crime)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.021247 -0.005769 -0.000725  0.004217  0.047783
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.205e-03  3.586e-03   2.009 0.044942 *
## prbarr         -3.202e-02  3.098e-03 -10.336 < 2e-16 ***
## prbconv        -1.807e-03  2.502e-04  -7.222 1.51e-12 ***
## polpc          1.944e+00  2.112e-01   9.203 < 2e-16 ***
## density        7.193e-03  3.171e-04  22.684 < 2e-16 ***
## as.factor(region)other 4.681e-03  9.785e-04   4.784 2.15e-06 ***
## as.factor(region)west -3.471e-03  1.161e-03  -2.990 0.002898 **
## pctmin         1.243e-04  3.220e-05   3.861 0.000125 ***
## wfed           2.641e-05  6.936e-06   3.808 0.000154 ***
## pctymle        7.466e-02  1.546e-02   4.828 1.74e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.00887 on 616 degrees of freedom
## Multiple R-squared:  0.7362, Adjusted R-squared:  0.7324
## F-statistic: 191.1 on 9 and 616 DF,  p-value: < 2.2e-16
```

We can note that all coefficients are significant and the  $R^2$ , as expected, was reduced but only marginally (0.7362415 versus 0.754353), which confirms that the model with this subset is indeed a good model.

Next, we proceeded to graphically analyse any nonlinearities between these predictors and the response, by looking at all pairwise plots:

```
fitnames = c("prbarr" , "prbconv" , "polpc" , "density" ,
             "as.factor(region)" , "pctmin" , "wfed" ,
             "pctymle", "crm rte")
pairs(Crime[names(Crime) %in% fitnames])
```



It can be seen that `prbrarr`, `prbconv` and `polpc` have a peak structure that would benefit from applying a `log` to them in order to shrink those peaks. Additionally, `wfed` has a nonlinear relationship with `wfed`, which makes it suitable as a polynomial regression predictor. Consequently, we run a new, nonlinear model with these modified predictors:

```
lm.3.fit = lm(crmrte ~ log(prbarr) + log(prbconv) +
              log(polpc) + density + as.factor(region) +
              pctmin + poly(wfed,3) + pctymle, data = Crime)
summary(lm.3.fit)
```

```
##
## Call:
## lm(formula = crmrte ~ log(prbarr) + log(prbconv) + log(polpc) +
##      density + as.factor(region) + pctmin + poly(wfed, 3) + pctymle,
##      data = Crime)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.024202 -0.004384 -0.000246  0.003672  0.045724
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.0564773   0.0046897   12.043 < 2e-16 ***
## log(prbarr)      -0.0144423   0.0009070  -15.923 < 2e-16 ***
## log(prbconv)     -0.0113128   0.0006276  -18.024 < 2e-16 ***
## log(polpc)       0.0099603   0.0006892   14.452 < 2e-16 ***
## density         0.0052809   0.0003094   17.068 < 2e-16 ***
## as.factor(region)other 0.0039568   0.0008168    4.844 1.61e-06 ***
## as.factor(region)west -0.0042023   0.0009729   -4.319 1.82e-05 ***
## pctmin          0.0001813   0.0000271    6.691 5.01e-11 ***
## poly(wfed, 3)1      0.0155450   0.0094692    1.642  0.101
## poly(wfed, 3)2      0.0006990   0.0079938    0.087  0.930
## poly(wfed, 3)3     -0.0101414   0.0076411   -1.327  0.185
## pctymle          0.0073072   0.0135778    0.538  0.591
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.007408 on 614 degrees of freedom
## Multiple R-squared:  0.8166, Adjusted R-squared:  0.8133
## F-statistic: 248.6 on 11 and 614 DF, p-value: < 2.2e-16
```

The lack of significance of the polynomials for `wfed` and the increase in  $R^2$  suggests that this model *overfits*. Hence, we decided to remove the polynomials related to `wfed`, but kept the *log* predictors as they have shown to be still very significant.

Next, we analysed all significant interactions between all interaction terms:

```
lm.4.fit = lm(crmrte ~ .*, data = Crime)
summary(lm.4.fit)$r.squared
```

```
## [1] 0.9706801
```

and plugged all interactions to our previous model:

```
lm.5.fit = lm(crmrte ~ log(prbarr) + log(prbconv) +
              log(polpc) + density + as.factor(region) +
              pctmin + poly(wfed,3) + pctymle + .*, data = Crime)
summary(lm.5.fit)$r.squared
```

```
## [1] 0.9725358
```

The number of interactions that we added to the model are:

```
dim(summary(lm.4.fit)$coefficients)
```

```
## [1] 300 4
```

We can see that both models are seriously overfitting ( $R^2$  values are 0.9706801 and 0.9725358 respectively, while the number of predictors has skyrocketed due to all interaction combinations). Even though it is tempting to keep only those interactions with a relevant significance value, since the removal of each of these predictors affects the overall model, we decided instead to choose a final model by using a *Stepwise Algorithm* applying *AIC* to decide. The stepwise procedure is too lengthy and cumbersome to show in text, but the code to generated is displayed below:

```
interaction.fit = stepAIC(lm.5.fit)
```

Once we got the fit, which has the following number of coefficients:

```
length(interaction.fit$coefficients)
```

```
## [1] 190
```

which means a reduction on the number of interactions by 30%, we proceeded to calculate both *training MSE* and *testing MSE*:

```
coefi = coef(interaction.fit)
pred = as.matrix(xs_train[, x_cols %in% names(coefi)]) %*% coefi[names(coefi)
                                                                %in% x_cols]
val.train.errors = mean((ys_train - pred)^2)
val.train.errors
```

```
## [1] 0.001248871
```

```
pred = as.matrix(xs_test[, x_cols %in% names(coefi)]) %*% coefi[names(coefi)
                                                                %in% x_cols]
val.test.errors = mean((ys_test - pred)^2)
val.test.errors
```

```
## [1] 0.001365305
```

Here, we can see that the error rate remains close to one (1.0932317) which still proves that this model holds. Now that we obtained a complex model consisting of linear variables, *log* variables and *interaction* variables, we're gonna perform *Lasso* in order to remove all interaction terms that are not significant, so that we arrive to a model easy to understand.

## Lasso Analysis

With the resulting model from all our previous steps, we performed *k-fold cross validation* using Lasso, in order to obtain the optimum value of  $\lambda$  for our model. The code generating the lasso is the following (note that the first line is a way to manually represent `interaction.fit` as the fit is not recognized by `cv.glmnet`, which requires a formula with a specific formatting):

```

formula = "~ log(prbconv) + log(polpc) + density + pctmin + poly(wfed, 3) +
pctymle + county + year + prbarr + prbconv + prbpris + avgsgen + polpc +
taxpc + region + smsa + wcon + wtuc + wtrd + wfir + wser + wmfg + wfed +
wsta + wloc + mix + county:year + county:avgsgen + county:polpc +
density:county + pctmin:county + county:wcon + county:wtuc + county:wtrd +
county:wfir + county:wmfg + county:wsta + county:wloc + pctymle:county +
year:prbconv + year:prbpris + year:polpc + year:region + year:smsa + pctmin:year +
year:wtrd + year:wfir + year:wmfg + year:wsta + year:mix + pctymle:year +
prbarr:prbpris + prbarr:polpc + density:prbarr + prbarr:region + pctmin:prbarr +
prbarr:wcon + prbarr:wtuc + prbarr:wtrd + prbarr:wfir + prbarr:wfed +
prbconv:prbpris + prbconv:polpc + prbconv:smsa + pctmin:prbconv + prbconv:wcon +
prbconv:wfir + prbconv:wser + prbconv:wmfg + prbconv:mix + density:prbpris +
prbpris:taxpc + prbpris:region + pctmin:prbpris + prbpris:wcon + prbpris:wtrd +
prbpris:wmfg + prbpris:wfed + prbpris:wsta + prbpris:wloc + density:avgsgen +
avgsgen:taxpc + avgsgen:region + avgsgen:smsa + pctmin:avgsgen + avgsgen:wcon +
avgsgen:wtrd + avgsgen:wfir + avgsgen:wser + avgsgen:wfed + avgsgen:mix + pctymle:avgsgen +
polpc:region + polpc:smsa + pctmin:polpc + polpc:wtuc + polpc:wtrd + polpc:wser +
polpc:wmfg + polpc:wfed + density:region + density:smsa + density:pctmin +
density:wcon + density:wtuc + density:wtrd + density:wsta + density:mix +
density:pctymle + taxpc:region + taxpc:smsa + pctmin:taxpc +
taxpc:wmfg + taxpc:wfed + taxpc:wsta + taxpc:wloc + region:smsa +
pctmin:region + region:wcon + region:wtuc + region:wtrd + region:wfir +
region:wser + region:wmfg + region:wfed + region:wsta + region:wloc +
region:mix + pctymle:region + pctmin:smsa + smsa:wtuc + smsa:wtrd +
smsa:wser + smsa:wmfg + smsa:wfed + smsa:wsta + smsa:mix + pctymle:smsa +
pctmin:wtrd + pctmin:wfir + pctmin:wser + pctmin:wmfg + pctmin:wfed +
pctmin:wsta + pctmin:wloc + pctmin:mix + pctmin:pctymle + wcon:wtuc +
wcon:wfir + wcon:wmfg + wcon:wfed + wcon:wsta + wcon:mix + pctymle:wcon +
wtuc:wsta + wtuc:wloc + wtrd:wfir + wtrd:wmfg + wtrd:wsta + wtrd:mix +
wfir:wmfg + wfir:wfed + wfir:wsta + wfir:wloc + wfir:mix + pctymle:wfir +
wser:wmfg + wser:wfed + wser:wloc + wser:mix + wmfg:wfed + pctymle:wmfg +
wfed:wsta + wfed:wloc + wfed:mix + wsta:wloc + wsta:mix + pctymle:wsta
+ pctymle:mix"
xs_lasso = model.matrix(as.formula(formula), Crime)
xs_lasso_train = xs_lasso[-test,]
xs_lasso_test = xs_lasso[test,]
grid=10^seq(2,-4,length=100)
lasso.mod=cv.glmnet(xs_lasso[-test ,],ys_train,alpha=1,lambda=grid)

```

Now that we have computed all possible values for  $\lambda$ , we can create the plot showing the *training RSE* and  $\lambda$  increases:

*#PLOT HERE*

We then chose a value of  $\lambda$  within 1 standard deviation from the optimum value, as this is a commonly established good practice. The fit with that value is performed in the code below:

```

lasso.mod.3=glmnet(xs_lasso[-test ,],ys_train,alpha=1,lambda=lasso.mod$lambda.1se)
lasso_vars = names(coef(lasso.mod.3)[,1][coef(lasso.mod.3)[,1]!=0])[-1]
length(lasso_vars)

```

```
## [1] 16
```

Lasso actually yielded 16 variables with nonzero values, a reduction of 92% with respect to the *stepwise AIC*. The *testing RSE* for this model is

```
ys_lasso_pred = predict(lasso.mod.3, xs_lasso_test, s = lasso.mod$lambda.1se)
lasso_rse = mean((ys_test - ys_lasso_pred)^2)
```

and the error rate w.r.t. to our original model using *best subset selection* is:

```
(lasso_rse / val.test.errors[which.min(val.test.errors)]) * 100
```

```
## [1] 4.240786
```

Finally, we analyse the relative average *L1* distance between our estimators and the true model:

```
mean(abs((ys_test - ys_lasso_pred)/ ys_test))
```

```
## [1] 0.2080632
```

which yields a more than reasonable value, since it's below 30%. Consequently, the final set obtained in this section, after performing *linear*, *best subset selection*, *log*, *polynomial*, *stepwise AIC* and *Lasso* yielded a model that will be used in the following sections for more complex fits that will derive in our final model.