The role of irrelevant variables in linear models

This notebook demonstrates overfitting and importance to remove irrelevant variables from a statistical model.

Artificial data

We will generate artificial data first. The dataset contains 100 independent variables and one dependent variable, all continuous. All the variables follow the normal distribution with zero mean and unit variance. Notice that there is **no relationship between dependent and independent variables**. Consequently, all further findings must be false positives (Type I error).

```
set.seed(3) # you can change the seed or remove the line to see more runs
d<-data.frame(matrix(rnorm(200*100,0,1),nrow=200))
d$out<-rnorm(200,0,1)
```

Learn a linear model

Next, we will create a linear regression model. There is no feature selection in this step. Check the number of seemingly relevant independent variables, their number should be around alpha*nvars=5. You may need to generate data multiple times to get an “illustrative” case.

```
lm.all<-lm(out ~ .,data=d)
lm.all.par<-coefficients(summary(lm.all))
sum(lm.all.par[,"Pr(>|t|)"]<0.05)
```

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

```
summary(lm.all) # print out the number of significant independent variables
```

```
##
## Call:
## lm(formula = out ~ ., data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.71750 -0.55166 -0.04587  0.59639  2.31520
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.351e-02  1.113e-01  -0.121   0.9036
## X1          -1.787e-01  1.082e-01  -1.652   0.1018
## X2           2.623e-01  1.124e-01   2.334   0.0216 *
## X3           2.221e-01  1.048e-01   2.119   0.0366 *
## X4          -1.251e-01  1.211e-01  -1.033   0.3041
## X5          -1.493e-01  1.080e-01  -1.382   0.1700
## X6          -1.037e-01  1.137e-01  -0.911   0.3643
## X7           1.914e-01  1.195e-01   1.601   0.1125
## X8           1.617e-01  1.205e-01   1.342   0.1827
## X9          -6.043e-02  1.084e-01  -0.558   0.5783
## X10          3.401e-02  1.060e-01   0.321   0.7489
## X11         -1.676e-01  1.172e-01  -1.429   0.1561
## X12          9.697e-02  1.115e-01   0.870   0.3866
## X13          3.607e-02  1.063e-01   0.339   0.7351
## X14          3.719e-02  1.090e-01   0.341   0.7336
## X15         -1.039e-01  1.183e-01  -0.878   0.3823
## X16          2.849e-02  1.249e-01   0.228   0.8201
## X17         -8.893e-02  1.117e-01  -0.796   0.4277
## X18         -7.911e-02  1.165e-01  -0.679   0.4985
## X19          4.472e-03  1.103e-01   0.041   0.9677
## X20         -7.417e-02  1.171e-01  -0.634   0.5278
## X21         -5.447e-02  1.087e-01  -0.501   0.6173
## X22          5.220e-02  1.205e-01   0.433   0.6658
## X23          1.120e-02  1.123e-01   0.100   0.9208
## X24          1.539e-01  1.062e-01   1.450   0.1503
## X25         -3.345e-02  1.146e-01  -0.292   0.7711
## X26         -1.089e-01  1.228e-01  -0.887   0.3773
## X27          7.796e-02  1.127e-01   0.692   0.4909
## X28         -7.585e-02  1.049e-01  -0.723   0.4714
## X29         -3.022e-02  1.212e-01  -0.249   0.8037
## X30         -7.516e-02  1.326e-01  -0.567   0.5722
## X31          1.348e-01  1.153e-01   1.170   0.2449
## X32         -1.215e-01  1.138e-01  -1.067   0.2885
## X33          1.274e-01  1.105e-01   1.154   0.2514
## X34         -1.031e-02  1.121e-01  -0.092   0.9269
## X35         -1.118e-01  1.112e-01  -1.006   0.3171
## X36          6.059e-03  1.140e-01   0.053   0.9577
## X37          6.237e-02  1.174e-01   0.531   0.5965
## X38         -1.013e-01  1.188e-01  -0.853   0.3958
## X39          3.014e-02  1.107e-01   0.272   0.7859
## X40         -7.317e-03  1.054e-01  -0.069   0.9448
## X41          9.551e-02  1.167e-01   0.819   0.4149
## X42          3.429e-02  1.107e-01   0.310   0.7573
## X43         -3.275e-02  1.051e-01  -0.312   0.7560
## X44         -2.283e-02  1.092e-01  -0.209   0.8348
## X45          6.326e-02  1.213e-01   0.522   0.6031
## X46         -8.991e-02  1.100e-01  -0.817   0.4158
## X47         -1.291e-01  1.025e-01  -1.260   0.2107
## X48         -4.107e-02  1.288e-01  -0.319   0.7505
## X49         -4.745e-02  1.159e-01  -0.409   0.6831
## X50          2.556e-02  1.123e-01   0.228   0.8205
## X51          1.105e-01  1.166e-01   0.948   0.3454
## X52         -1.401e-01  9.693e-02  -1.445   0.1515
## X53          1.219e-01  1.223e-01   0.997   0.3212
## X54         -3.604e-02  1.181e-01  -0.305   0.7608
## X55          1.390e-01  1.259e-01   1.104   0.2721
## X56          8.914e-02  1.061e-01   0.840   0.4027
## X57          1.874e-01  1.132e-01   1.655   0.1011
## X58          1.133e-01  1.121e-01   1.011   0.3146
## X59         -7.545e-02  1.133e-01  -0.666   0.5069
## X60          4.617e-02  1.122e-01   0.412   0.6815
## X61         -1.438e-01  1.205e-01  -1.193   0.2357
## X62         -5.427e-02  1.102e-01  -0.493   0.6234
## X63         -8.474e-02  1.059e-01  -0.800   0.4257
## X64          7.870e-02  1.005e-01   0.783   0.4354
## X65         -5.987e-05  1.103e-01  -0.001   0.9996
## X66         -2.079e-01  1.101e-01  -1.889   0.0618 .
## X67          5.217e-02  1.280e-01   0.408   0.6845
## X68         -1.035e-01  1.055e-01  -0.981   0.3289
## X69          8.384e-02  1.048e-01   0.800   0.4255
## X70          7.474e-02  1.058e-01   0.707   0.4814
## X71          4.206e-02  1.144e-01   0.367   0.7140
## X72         -1.247e-01  1.241e-01  -1.005   0.3174
## X73          1.530e-01  1.225e-01   1.249   0.2145
## X74         -8.651e-02  1.262e-01  -0.685   0.4947
## X75         -1.015e-01  1.167e-01  -0.870   0.3864
## X76          1.989e-01  1.097e-01   1.812   0.0730 .
## X77         -8.960e-02  1.124e-01  -0.797   0.4274
## X78         -2.961e-02  1.075e-01  -0.275   0.7835
## X79          3.957e-02  1.115e-01   0.355   0.7234
## X80          5.116e-02  1.172e-01   0.436   0.6634
## X81          1.629e-01  1.116e-01   1.460   0.1476
## X82          6.212e-02  1.202e-01   0.517   0.6065
## X83         -1.704e-01  1.020e-01  -1.670   0.0980 .
## X84         -2.316e-01  1.023e-01  -2.264   0.0258 *
## X85          1.029e-02  1.088e-01   0.095   0.9249
## X86          4.341e-02  1.152e-01   0.377   0.7070
## X87         -5.847e-02  1.057e-01  -0.553   0.5814
## X88          4.877e-02  1.204e-01   0.405   0.6863
## X89         -8.241e-03  1.044e-01  -0.079   0.9372
## X90         -1.692e-01  1.366e-01  -1.239   0.2183
## X91          1.923e-02  1.047e-01   0.184   0.8546
## X92          7.270e-02  1.153e-01   0.630   0.5299
## X93          8.263e-02  1.073e-01   0.770   0.4429
## X94          1.165e-01  1.117e-01   1.043   0.2996
## X95         -2.243e-01  1.044e-01  -2.148   0.0341 *
## X96          1.262e-01  1.152e-01   1.096   0.2759
## X97          8.587e-02  1.244e-01   0.690   0.4916
## X98         -2.049e-01  1.126e-01  -1.820   0.0718 .
## X99          9.602e-03  1.093e-01   0.088   0.9302
## X100         5.274e-02  1.160e-01   0.455   0.6503
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.144 on 99 degrees of freedom
## Multiple R-squared:  0.464,  Adjusted R-squared:  -0.0774
## F-statistic: 0.857 on 100 and 99 DF,  p-value: 0.7786
```

There were 4 independent variables looking significant. On the other hand, both Adjusted R-squared and F-statistic suggest that the full model should not be used and none of its independent variables is truly relevant.

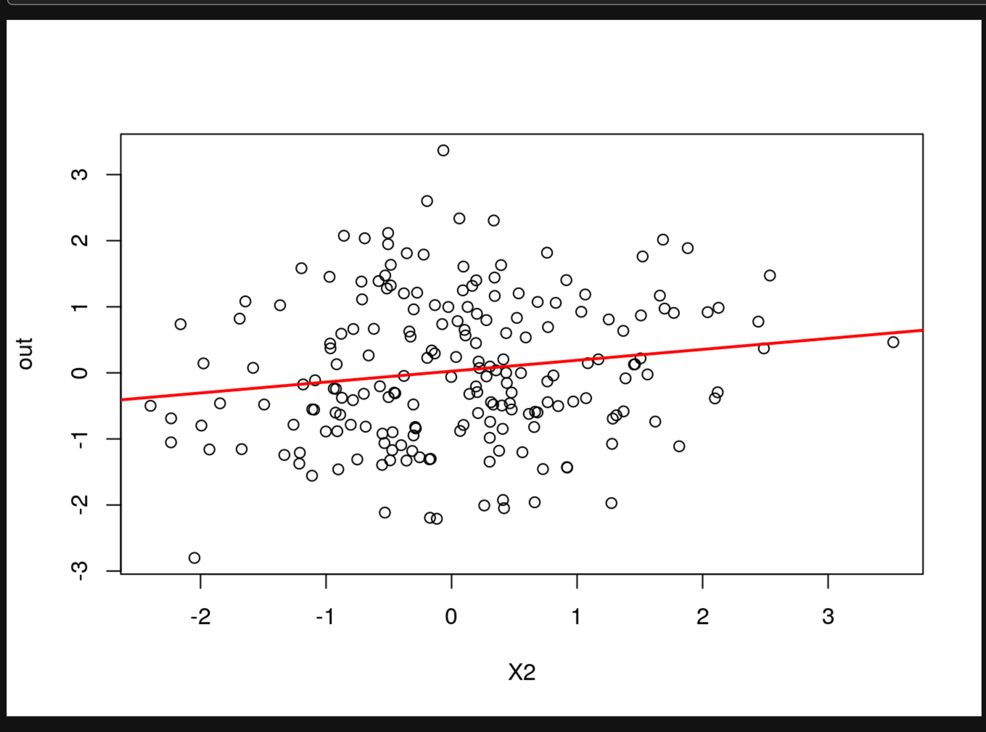
See the best predictor

Let us pick the best predictor/independent variable and visualise its relationship with the dependent variable. Both correlation and visual fit look significant, still, it is a false positive due to **selection bias**. The significance is only the outcome of selection of the best variable out of a large pool of irrelevant variables.

```
pred<-rownames(lm.all.par)[which(lm.all.par[,"Pr(>|t|)"]==min(lm.all.par[,"Pr(>|t|)"]))] # find the most significant predict
or
cor.test(d[[pred]],d$out)
```

```
##
##  Pearson's product-moment correlation
##
## data:  d[[pred]] and d$out
## t = 2.1997, df = 198, p-value = 0.02899
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.01605274 0.28703994
## sample estimates:
##       cor
## 0.1544498
```

```
plot(d[[pred]], d$out,xlab=pred,ylab="out") # visualize the relationship between indep and dep variable
abline(lm(as.formula(paste("out~",pred)), d),col="red", lw=2) # visualize the fit
```



The pairwise correlation between the best predictor and the outcome is significant. Note that the correlation coefficient and its p-value may differ from the coefficient and p-value of the given variable observed in the multivariate model above (paired vs multivariate analysis).

Alternative regression models

There are four alternative linear models to consider: 1. lm.all … the full model, use all 100 independent variables, 2. lm.sel … use only the (seemingly) relevant variables, 3. lm.simple … use only the intercept, 4. lm.zero … always predict zero.

Knowing the origin of data, it is obvious that it holds lm.all<lm.sel<lm.simple<lm.zero when considering model performance on unseen data. The best prediction is to always predict zero. The other linear models will be constructed as follows:

```
sel.form<-as.formula(paste("out~",paste(rownames(lm.all.par)[lm.all.par[,"Pr(>|t|)"]<0.05],collapse="+")))
lm.sel<- lm(sel.form, data=d) # obviously, it is better to employ lasso or stepwise regression, this is just to illustrate
lm.simple<-lm(out ~ 1,data=d)
lm.zero<-lm(out ~ 0,data=d)
```

Compare the alternative models, find the best one

Large samples, holdout method

Let us suppose the common situation of not knowing the true relationships among variables. The goal is to find the best model, i.e., the model that represents a trade-off between sufficient flexibility and minimal overfitting. In this case, we deal with artificial data and we can easily generate previously unseen test samples. **The holdout method** makes the simplest testing choice here:

```
d_test<-data.frame(matrix(rnorm(1000*100,0,1),nrow=1000))
d_test$out<-rnorm(1000,0,1)

myRMSE(predict(lm.all,d_test),d_test$out)
```

```
## [1] 1.46135
```

```
myRMSE(predict(lm.sel,d_test),d_test$out)
```

```
## [1] 1.040769
```

```
myRMSE(predict(lm.simple,d_test),d_test$out)
```

```
## [1] 0.994938
```

```
myRMSE(predict(lm.zero,d_test),d_test$out)
```

```
## [1] 0.9946782
```

The zero model shows the smallest root mean squared error. At the same time, the model has the smallest variance in this error (not shown). The holdout method ranks the linear models correctly. The absolute value of calculated RMSE for the zero model matches with the amount of irreducible error (the value of 1 as we deal with the outcome that has zero mean and unit variance).

Small samples, ANOVA and cross-validation

Often, the sample set is small and new observations cannot be easily reached. Under such circumstances, ANOVA and cross-validation are available to compare the models. Let us start with ANOVA:

```
anova(lm.zero,lm.simple,lm.sel,lm.all)
```

	Res.Df <dbl>	RSS <dbl>	Df <dbl>	Sum of Sq <dbl>	F <dbl>	Pr(>F) <dbl>
1	200	241.7399	NA	NA	NA	NA
2	199	241.5229	1	0.2170007	0.1659505	0.6846156
3	195	224.5144	4	17.0085789	3.2518133	0.0149781
4	99	129.4546	96	95.0597110	0.7572562	0.9138584

4 rows

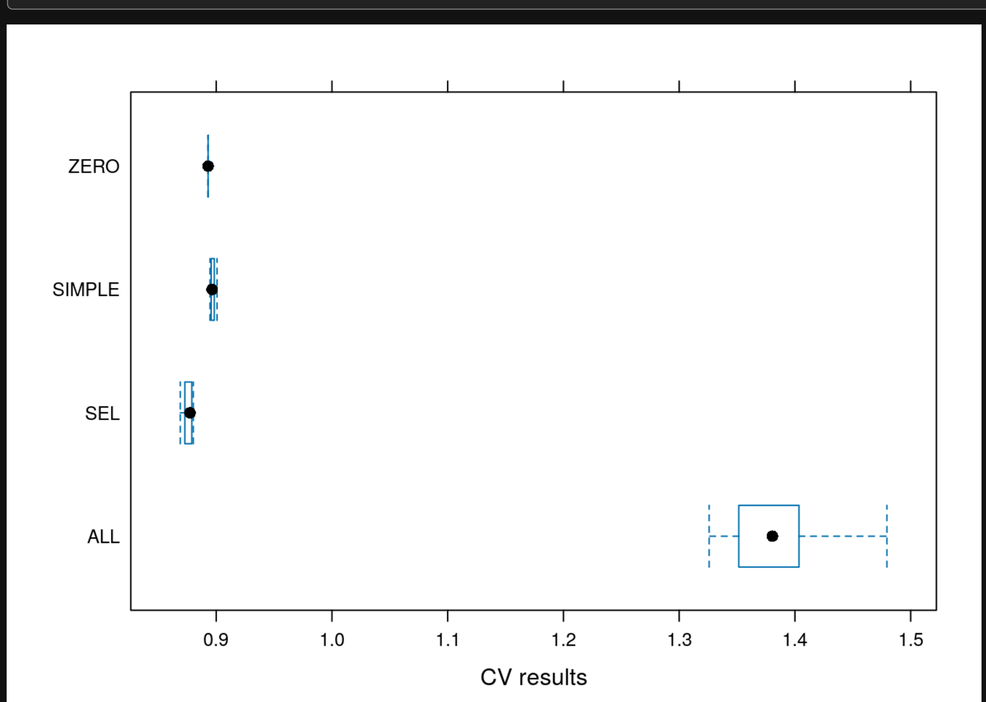
```
anova(lm.zero,lm.all)
```

	Res.Df <dbl>	RSS <dbl>	Df <dbl>	Sum of Sq <dbl>	F <dbl>	Pr(>F) <dbl>
1	200	241.7399	NA	NA	NA	NA
2	99	129.4546	101	112.2853	0.850196	0.7909779

2 rows

ANOVA may often recommend lm.sel, which is not correct. The method may suffer from conceptual simplicity of the test (no need for test data, the decision made on train data only). However, ANOVA does not prefer lm.all over lm.zero. In general, it is a legitimate option. Let us employ cross-validation now:

```
folds <- cvFolds(nrow(d), K = 10, R = 10) # 10 times repeated 10-fold CV
cvFitLmAll <- cvLm(lm.all, cost = rtmspe, folds = folds, trim = 0.1)
cvFitLmSel <- cvLm(lm.sel, cost = rtmspe, folds = folds, trim = 0.1)
cvFitSimple <- cvLm(lm.simple, cost = rtmspe, folds = folds, trim = 0.1)
cvFitZero <- cvLm(lm.zero, cost = rtmspe, folds = folds, trim = 0.1)
cvFits <- cvSelect(ALL = cvFitLmAll, SEL = cvFitLmSel, SIMPLE = cvFitSimple, ZERO=cvFitZero)
bwplot(cvFits) # rmse is the default cost function
```



The recommendation is misleading again, sel is best, full model worst. The problem was methodological in this case, **we cannot do feature selection on all the data**, we must proceed independently in all the folds! We will not implement the correct way of cross-validation here. However, remember that correct CV implementation could be non-trivial when doing learning, data preprocessing and hyperparameter tuning at the same time.

Summary

Irrelevant features cause overfitting and make our models work worse on unseen data. If having a finite/limited sample set the learning algorithm finds spurious relationships which increases variance and thus error. Removal of irrelevant features is crucial namely when dealing with a large number of them. (Proper) testing on unseen data (hold-out method, cross-validation) can help to detect overfitting and find out the optimal complexity of the model. ANOVA helps to decide the complexity from one run of the model only.

Further questions and tasks:

1. Show how the previously learned feature selection methods (p-values, stepwise selection, shrinkage) work in this case. Clearly demonstrate whether they work well/fail and explain why. Play with several different random generator seeds.
2. Describe step by step the correct way of model comparison procedure through cross-validation that was incorrectly implemented above. You can also implement the procedure, however, the implementation is optional only.