Coding_Sample

2023-10-14

We will load a dataset that includes 50 observations for 16 variables. Each observation is indexed to a year. Observations track indicators for the Tour de France cycling race. There are 10 control variables that measure race statistics (e.g., winner time), 5 variables of interest that are all dummy variables that measure whether an anti-doping policy was in place, and 1 response variable that measures the percentage of total cyclists that tested positive for performance enhancing drugs.

```r
# loading dataset and packages
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
library(ggplot2)
library(sandwich)
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(reshape2)
library(regclass)
```

```
## Loading required package: bestglm
```

```
## Loading required package: leaps
```

```
## Loading required package: VGAM

## Loading required package: stats4

## Loading required package: splines

##
## Attaching package: 'VGAM'

## The following object is masked from 'package:lmtest':
##
##     lrtest

## Loading required package: rpart

## Loading required package: randomForest

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.
```

```r
library(stargazer)
```

```
##
## Please cite as:

##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```r
tdf <- read.csv("/Users/martrinmunoz/Desktop/EconPredoc/Writing Samples/TdF/tdf_cleaned.csv")
```

```r
# let's look at the dataset
summary(tdf)
```

```
##       year         ped_tot        gen_ad_test   amph_test       epo_test
##  Min.   :1968   Min.   :0.1060   Min.   :1    Min.   :0.00   Min.   :0.00
##  1st Qu.:1980   1st Qu.:0.3145   1st Qu.:1    1st Qu.:1.00   1st Qu.:0.00
##  Median :1992   Median :0.4125   Median :1    Median :1.00   Median :0.00
##  Mean   :1992   Mean   :0.3729   Mean   :1    Mean   :0.88   Mean   :0.34
##  3rd Qu.:2005   3rd Qu.:0.4490   3rd Qu.:1    3rd Qu.:1.00   3rd Qu.:1.00
##  Max.   :2017   Max.   :0.5400   Max.   :1    Max.   :1.00   Max.   :1.00
##   bio_passport   night_test        ooct         num_stages      tot_length
##  Min.   :0.0   Min.   :0.00   Min.   :0.00   Min.   :20.50   Min.   :3278
##  1st Qu.:0.0   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:20.50   1st Qu.:3529
##  Median :0.0   Median :0.00   Median :0.00   Median :21.50   Median :3734
##  Mean   :0.2   Mean   :0.06   Mean   :0.24   Mean   :21.65   Mean   :3754
##  3rd Qu.:0.0   3rd Qu.:0.00   3rd Qu.:0.00   3rd Qu.:22.50   3rd Qu.:3982
```

```
##  Max.   :1.0   Max.   :1.00   Max.   :1.00   Max.   :25.50   Max.   :4492
##    avg_speed      num_entrants   num_finishers   first_prize_money
##  Min.   :33.41   Min.   :100.0   Min.   : 53.0   Min.   : 18006
##  1st Qu.:36.23   1st Qu.:150.0   1st Qu.: 97.0   1st Qu.: 49364
##  Median :38.93   Median :209.0   Median :135.5   Median :433754
##  Mean   :38.21   Mean   :186.1   Mean   :127.1   Mean   :290396
##  3rd Qu.:39.91   3rd Qu.:219.0   3rd Qu.:152.5   3rd Qu.:468621
##  Max.   :41.65   Max.   :229.0   Max.   :174.0   Max.   :520725
##  tot_prize_money   tot_time_winner  second_lag_time
##  Min.   : 482781   Min.   : 82.09   Min.   :0.00200
##  1st Qu.: 701118   1st Qu.: 87.70   1st Qu.:0.02800
##  Median :2104928   Median : 92.79   Median :0.07150
##  Mean   :1991232   Mean   : 98.55   Mean   :0.08872
##  3rd Qu.:3043916   3rd Qu.:109.07   3rd Qu.:0.12075
##  Max.   :3702938   Max.   :133.83   Max.   :0.29800
```

```r
# notice that the 'gen_ad' variable is constant throughout. This will create
# problems for our analysis, so let's drop it. Note that 'gen_ad' was not
# included in my description of the dataset above.
vars_to_remove <- c('gen_ad_test')
tdf <- tdf[, !(colnames(tdf) %in% vars_to_remove)]
```

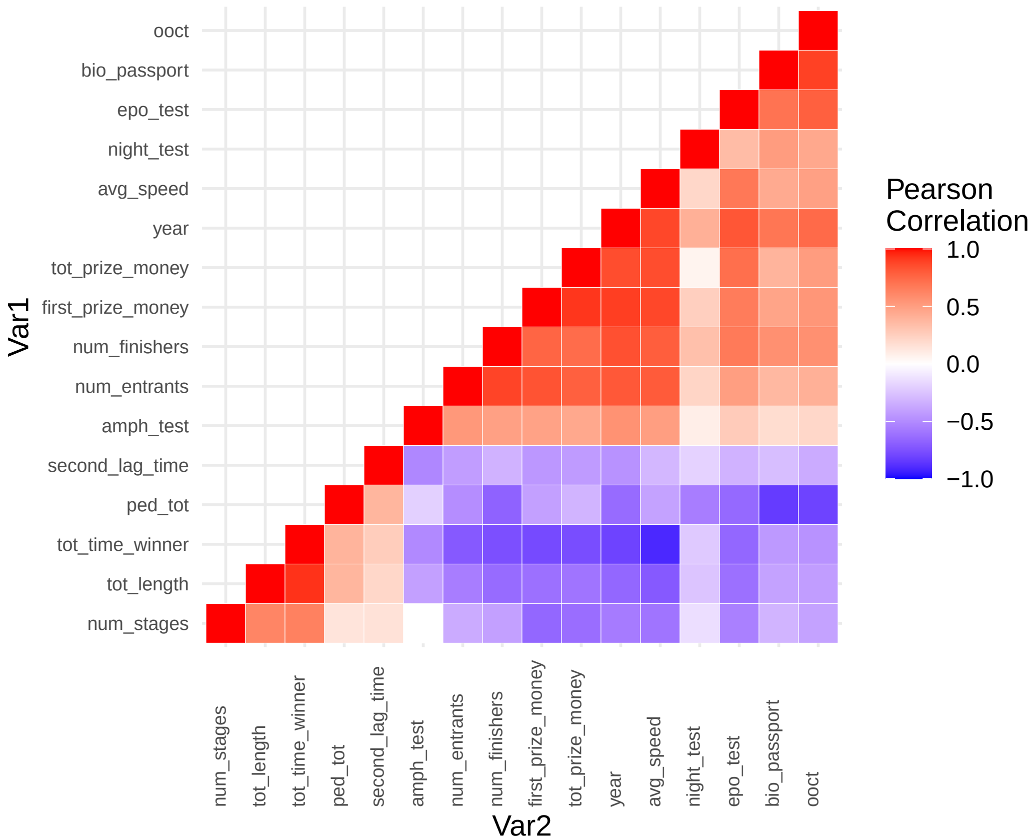
```r
# let's run a linear regression of ped_tot on all the predictor variables
model1 <- lm(ped_tot ~., data = tdf)
summary(model1)
```

```
##
## Call:
## lm(formula = ped_tot ~ ., data = tdf)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.073605 -0.022833  0.001806  0.020037  0.057763
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -2.114e+00  5.243e+00  -0.403  0.68929
## year               1.704e-03  2.698e-03   0.632  0.53174
## amph_test          5.976e-02  3.986e-02   1.499  0.14306
## epo_test          -7.410e-02  3.656e-02  -2.027  0.05057 .
## bio_passport      -9.260e-02  3.451e-02  -2.683  0.01118 *
## night_test        -1.458e-02  3.430e-02  -0.425  0.67356
## ooct              -8.750e-02  3.213e-02  -2.723  0.01014 *
## num_stages        -2.193e-02  1.007e-02  -2.178  0.03645 *
## tot_length         9.843e-05  1.175e-04   0.838  0.40798
## avg_speed         -6.890e-03  1.543e-02  -0.446  0.65808
## num_entrants      -7.241e-04  5.570e-04  -1.300  0.20235
## num_finishers     -1.750e-03  5.068e-04  -3.454  0.00150 **
## first_prize_money -1.095e-07  1.019e-07  -1.075  0.29011
## tot_prize_money    6.076e-08  1.759e-08   3.454  0.00150 **
## tot_time_winner   -2.933e-03  4.411e-03  -0.665  0.51059
## second_lag_time    3.339e-01  1.050e-01   3.180  0.00313 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.03522 on 34 degrees of freedom
## Multiple R-squared:  0.9408, Adjusted R-squared:  0.9147
## F-statistic: 36.04 on 15 and 34 DF,  p-value: < 2.2e-16
```

There are two potential issues with the linear regression. Firstly, there may be multicollinearity between predictor variables and secondly there may be too many variables for the number of observations which could lead to overfitting. So let's examine whether there is multicollinearity. If there is, we may be able to drop some variables to prevent overfitting.

```r
# let's create a correlation matrix
heatmap <- function(df, vars) {
  cormat <- round(cor(na.omit(df)),2)
  # set up hierarchical clustering
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat <- cormat[hc$order, hc$order]
  # remove redundant information
  cormat[lower.tri(cormat)] <- NA
  # melt cormat
  melted_cormat <- melt(cormat, na.rm = TRUE)
  # create matrix
  heat_plot <- ggplot(melted_cormat, aes(Var2, Var1, fill = value)) +
    geom_tile(color = "white") +
    scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                         midpoint = 0, limit = c(-1,1), space = "Lab",
                         name = "Pearson\nCorrelation") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, vjust = 0, size = 7,
                                     hjust = 0),
    axis.text.y = element_text(size = 7)) +
    coord_fixed()
    colnames(melted_cormat) <- c('Var1', 'Var2', 'correlation')
    melted_cormat <- melted_cormat[order(melted_cormat$Var1),]
    return(list(heat_plot = heat_plot, cormat = melted_cormat))
}
# create cormat for tdf dataset
heatmap(df = tdf, vars = colnames(tdf))$heat_plot
```




```
# Now let's look at the Variable Inflation Factor for the unrestricted model
VIF(model1)
```

```
##             year         amph_test          epo_test      bio_passport
##        61.087819          6.763028         12.088262          7.680833
##       night_test              ooct        num_stages        tot_length
##         2.675428          7.591460          5.543018         44.738136
##        avg_speed      num_entrants     num_finishers first_prize_money
##        45.418148         18.704689          9.910155         18.211458
##  tot_prize_money   tot_time_winner   second_lag_time
##        16.225575        126.913025          2.220194
```

```
# looks like some of these independent variables have severe VIF (i.e., > 10),
# so let's see if we can drop any. First, notice that (i) total prize amount
# (tot_prize_amount) and first prize amount (first_prize_amount) and
# (ii) number of entrants (num_entrants) and number of finishers (num_finishers)
# are pairs of variables that are likely very highly correlated such that one of
# the pair can be dropped. Let's confirm this by printing their correlations
# below
cor(tdf$tot_prize_money, tdf$first_prize_money)
```

```
## [1] 0.9178642
```

```
cor(tdf$num_entrants, tdf$num_finishers)
```

```
## [1] 0.8825201
```

```
# So we will drop one of the variables from each pair. I will drop the one with
# the higher VIF. Let's take a look at some other variables with very high VIF.
# Notice that year has a very high VIF. It will also be a problem if year is
```

```r
# correlated with the variables of interest because then we will imprecisely
# estimate the coefficients on the variable of interest. Let's check if year is
# correlated with variables of interest.
cor(tdf[-1], tdf$year)
```

```
##                         [,1]
## ped_tot           -0.6362180
## amph_test          0.5629625
## epo_test           0.8206518
## bio_passport       0.6929589
## night_test         0.4114216
## ooct               0.7398777
## num_stages        -0.5672752
## tot_length        -0.6625301
## avg_speed          0.8743058
## num_entrants       0.8145740
## num_finishers      0.8447573
## first_prize_money  0.8985264
## tot_prize_money    0.8531097
## tot_time_winner   -0.8095467
## second_lag_time   -0.4696109
```

```r
# so year is severely correlated with epo_test and ooct which are variables of
# interest. Given that year also has one of the highest VIFs we will drop that
# too. Finally, I will also drop avg_speed  since it was unclear how this
# was measured and it also has a high VIF. Let's remove the variables now
model1_remove <- c('avg_speed', 'num_entrants', 'first_prize_money',
                   'tot_time_winner', 'year')
tdf <- tdf[, !(colnames(tdf) %in% model1_remove)]
```

```r
# let's run a linear regression on the restricted model
model2 <- lm(ped_tot ~., data = tdf)
summary(model2)
```

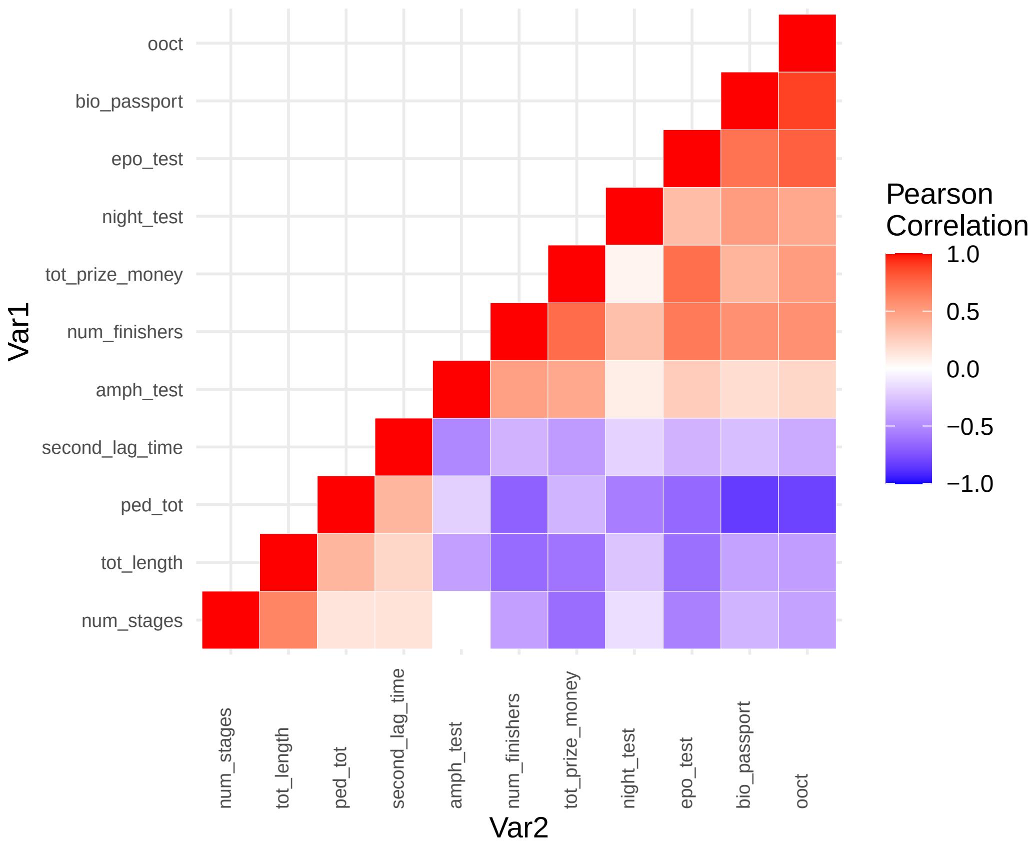
```
##
## Call:
## lm(formula = ped_tot ~ ., data = tdf)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.07017 -0.02248 -0.00172  0.02308  0.07161
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       9.470e-01  1.476e-01   6.417 1.37e-07 ***
## amph_test         8.394e-02  2.335e-02   3.595 0.000899 ***
## epo_test         -4.166e-02  2.334e-02  -1.785 0.082084 .
## bio_passport     -8.214e-02  2.981e-02  -2.756 0.008860 **
## night_test       -2.456e-02  2.656e-02  -0.925 0.360856
## ooct             -8.229e-02  3.068e-02  -2.682 0.010675 *
## num_stages       -2.436e-02  7.314e-03  -3.331 0.001902 **
## tot_length        2.890e-05  3.042e-05   0.950 0.347940
## num_finishers    -2.339e-03  3.054e-04  -7.659 2.72e-09 ***
## tot_prize_money   4.199e-08  1.030e-08   4.078 0.000217 ***
## second_lag_time   4.110e-01  9.215e-02   4.460 6.75e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0354 on 39 degrees of freedom
## Multiple R-squared:  0.9314, Adjusted R-squared:  0.9139
## F-statistic: 52.99 on 10 and 39 DF,  p-value: < 2.2e-16
```

```r
# let's look at VIF of the restricted model
VIF(model2)
```

```
##       amph_test        epo_test    bio_passport      night_test            ooct
##        2.297097        4.879392        5.673003        1.587879        6.852144
##      num_stages      tot_length   num_finishers tot_prize_money second_lag_time
##        2.894875        2.970654        3.562655        5.503620        1.693550
```

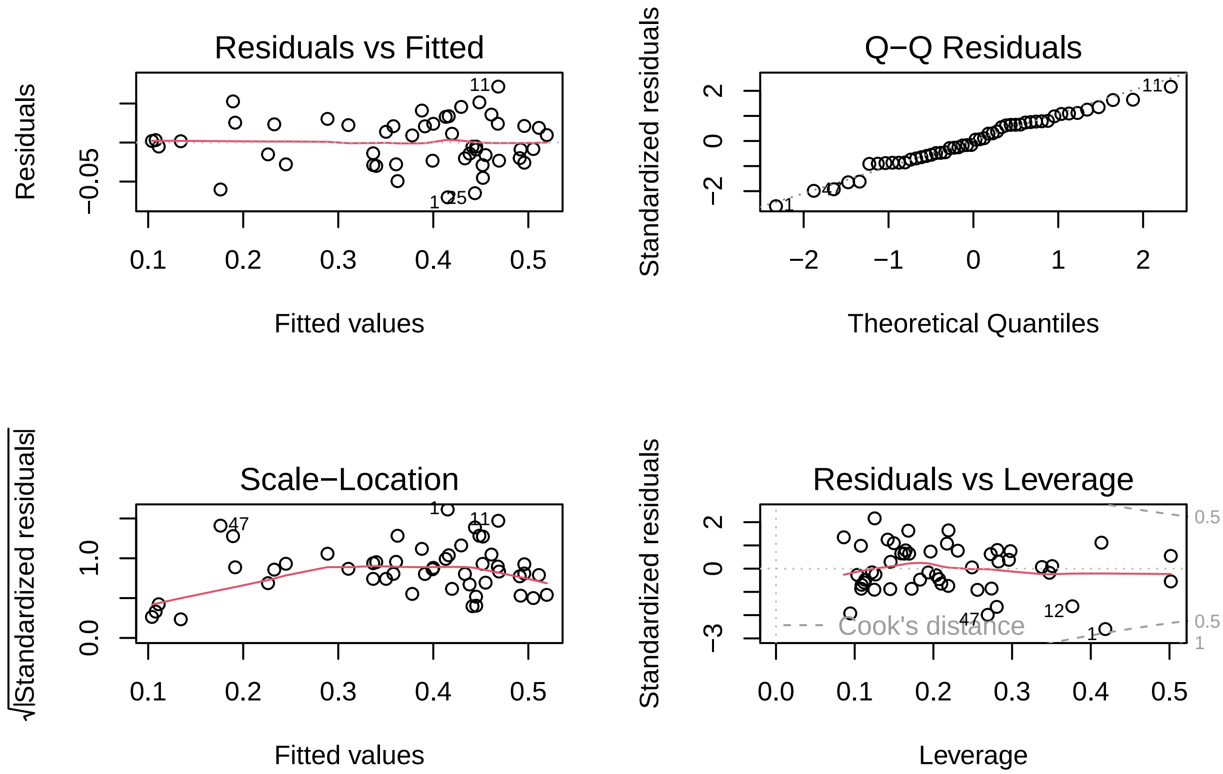
```r
# the VIF values look much better. They are all under 10 now. Let's create a
# correlation matrix again and then list all the correlations between variables
# that are higher than |0.7|
heatmap(df = tdf, vars = colnames(tdf))$heat_plot
```




```r
tdf_cor <- heatmap(df = tdf, vars = colnames(tdf))$cormat
for(i in 1:nrow(tdf_cor)) {
  if (abs(tdf_cor[i, 'correlation']) > 0.7 &
      tdf_cor[i, 'Var1'] != tdf_cor[i, 'Var2']) {
    print(as.name(paste(' ', tdf_cor[i, 'Var1'], 'and', tdf_cor[i, 'Var2'],
                        'have correlation', tdf_cor[i, 'correlation'])))
  }
}
```

```
## `  ped_tot and bio_passport have correlation -0.84`
## `  ped_tot and ooct have correlation -0.81`
## `  num_finishers and tot_prize_money have correlation 0.73`
## `  tot_prize_money and epo_test have correlation 0.72`
## `  epo_test and ooct have correlation 0.78`
## `  bio_passport and ooct have correlation 0.89`
```

```
# looks like we have 0.78 cor between epo_test and ooct and 0.89 between
# bio_passport and ooct, which means that these coefficients could be
# imprecisely estimated. I will ignore this potential issue for now.
# Let's do model diagnostics for the restricted model
par(mfrow = c(2, 2))
plot(model2)
```




```
# it appears that there is heteroskedasticity (non-horizontal line on bottom
# left plot), so let's get heteroskedastic robust standard errors
coeftest(model2, vcov = vcovHC(model2, type = "HC1"))
```

```
##
## t test of coefficients:
##
##                   Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)      9.4703e-01  1.4637e-01  6.4703 1.153e-07 ***
## amph_test        8.3936e-02  2.8394e-02  2.9561 0.0052647 **
## epo_test        -4.1661e-02  2.1046e-02 -1.9796 0.0548426 .
## bio_passport    -8.2138e-02  1.9471e-02 -4.2184 0.0001418 ***
## night_test      -2.4559e-02  1.8981e-02 -1.2939 0.2033168
## ooct            -8.2289e-02  1.9509e-02 -4.2181 0.0001419 ***
## num_stages      -2.4362e-02  7.8334e-03 -3.1100 0.0034891 **
## tot_length       2.8905e-05  3.5738e-05  0.8088 0.4235304
```

```
## num_finishers   -2.3389e-03  3.0353e-04 -7.7055 2.350e-09 ***
## tot_prize_money  4.1992e-08  9.4934e-09  4.4233 7.574e-05 ***
## second_lag_time  4.1104e-01  1.1631e-01  3.5341 0.0010712 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# One issue with the OLS regression is that the dependent variable is bounded
# between 0 and 1, which means the OLS regression might imprecisely estimate the
# standard errors of coefficients and give predictions of the dependent variable
# that are above 1 or below 0. This is less of an issue if most of the data from
# the dependent variable is not close to the boundary, which is the case with
# ped_tot.
summary(tdf$ped_tot)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1060  0.3145  0.4125  0.3729  0.4490  0.5400
```

```
# Nevertheless, let's run a fractional logistic regression to cover our bases.
# We will run the regression on the same set of variables that we used in the
# previous OLS regression.
logistic1 <- glm(ped_tot~., data = tdf, family = quasibinomial('logit'))
# Let's also get standard error estimates that are heteroskedastic robust
se_glm_robust_quasi = coeftest(logistic1, vcov = vcovHC(logistic1, type="HC1"))
# Results are below
se_glm_robust_quasi
```

```
##
## z test of coefficients:
##
##                    Estimate  Std. Error z value  Pr(>|z|)
## (Intercept)      1.8546e+00  6.3142e-01  2.9372 0.0033119 **
## amph_test        3.3785e-01  1.2307e-01  2.7452 0.0060476 **
## epo_test        -1.8576e-01  9.0680e-02 -2.0486 0.0405052 *
## bio_passport    -4.7733e-01  1.0543e-01 -4.5274 5.971e-06 ***
## night_test      -4.1971e-01  1.2220e-01 -3.4347 0.0005932 ***
## ooct            -3.5161e-01  8.4494e-02 -4.1614 3.164e-05 ***
## num_stages      -9.5820e-02  3.5335e-02 -2.7117 0.0066932 **
## tot_length       1.0004e-04  1.6064e-04  0.6228 0.5334435
## num_finishers   -1.0208e-02  1.4165e-03 -7.2063 5.748e-13 ***
## tot_prize_money  1.9310e-07  4.8640e-08  3.9700 7.189e-05 ***
## second_lag_time  1.7350e+00  4.9970e-01  3.4722 0.0005162 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now let's take a different approach. Instead of choosing what predictor variables to include in our model, let's try an automatic method that selects predictor variables for us based on an algorithm.

We will use LASSO to do this.

```
# First we load the dataset again
tdf <- read.csv("/Users/martrinmunoz/Desktop/EconPredoc/Writing Samples/TdF/tdf_cleaned.csv")
#Let's remove the 'gen_ad' variable again
vars_to_remove <- c('gen_ad_test')
tdf <- tdf[, !(colnames(tdf) %in% vars_to_remove)]
```

```r
# Now we create a training set
X_train <- model.matrix(ped_tot~., data = tdf)[,-1]
Y_train <- tdf$ped_tot
# We create a list that will store the mean-squared errors from cross-fold
# validation
MSEs <- NULL
# Now we run LASSO using 5-fold cross-validation and we use a for loop to repeat
# this algorithm 100 times to try and guard against the stochastic nature of the
# algorithm, which is especially a problem when the dataset is small as it is
# here. Since the dataset is small we also don't separate out a training and
# testing set. We just use the whole dataset to train the model. Alpha=1
# indicates that this is a LASSO regression.
# Each time the loop is run, we append MSEs to the MSE list. The LASSO algorithm
# estimates the lambda parameter that results in the lowest MSE. The lambda
# parameter is a penalty term that is used in generating the model.
for (i in 1:100){
  cv <- cv.glmnet(x = X_train, y = Y_train, alpha=1, nfolds=5,
                  standardize = TRUE)
  MSEs <- cbind(MSEs, cv$cvm)
}
# we name the rows of the MSE list based on the lambda estimated through the
# LASSO model
rownames(MSEs) <- cv$lambda
# finally, we display the model based on the lambda
# that is the minimum lambda plus 1 standard error
model3 <- coef(cv, s = cv$lambda.1se)
model3
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)        8.805105e-01
## year               .
## amph_test          2.936671e-02
## epo_test          -5.708355e-03
## bio_passport      -1.018389e-01
## night_test        -4.529744e-02
## ooct              -7.987883e-02
## num_stages        -1.540839e-02
## tot_length         .
## avg_speed          .
## num_entrants       .
## num_finishers     -1.623525e-03
## first_prize_money  .
## tot_prize_money    1.731768e-08
## tot_time_winner    .
## second_lag_time    1.830929e-01
```