Applied Statistical Methods - Solution 7

Peter von Rohr

2022-04-06

Problem 1: Model Selection

Given is a dataset with body weight as a response and different other variables and factors. The columns `Breed` and `BCS` (Body Condition Score) are taken as factors. All other columns are taken as predictor variables. The column `Animal` is not used in any model. Use model selection to find the relevant predictor variables and factors for the best linear fixed effect model. Use the estimated mean square error C_p as a quality measure for a single linear model. The dataset to be analysed can be obtained from

```
## https://charlotte-ngs.github.io/asmss2022/data/asm_bw_mod_sel.csv
```

Your Tasks

- Run a forward selection for the given dataset to find the best model
- Do a backward elemination for the given dataset to find the best model
- Compare the two models whether they are identical with respect to the set of predictor variables and factors that they include.

Solution

Because, we need the residual standard deviation of the full model and backward elimination starts with the full model, we start with backward elimination

Backward Elimination

- Read the data and convert `Breed` and `BCS` to factors

```r
if (params$isonline){
  s_ex07p01_path <- "https://charlotte-ngs.github.io/asmss2022/data/asm_bw_mod_sel.csv"
} else {
  s_ex07p01_path <- file.path(here::here(), "docs", "data", "asm_bw_mod_sel.csv")
}
tbl_ex07p01 <- readr::read_csv(file = s_ex07p01_path)
```

```
## Rows: 30 Columns: 6
```

```
## -- Column specification -----------------------------------------------------------------------
## Delimiter: ","
## chr (1): Breed
## dbl (5): Animal, BC, BW, HEI, BCS
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
tbl_ex07p01$BCS <- as.factor(tbl_ex07p01$BCS)
tbl_ex07p01$Breed <- as.factor(tbl_ex07p01$Breed)
```

- Start with the full model considering all variables

```r
s_resp <- "BW"
vec_cols_to_ignore <- c("Animal")
vec_pred_full <- setdiff(colnames(tbl_ex07p01), c(s_resp, vec_cols_to_ignore))
fmlm_full <- as.formula(paste0(s_resp, " ~ ",
                               paste0(vec_pred_full, collapse = " + "),
                               collapse = ""))
lm_ex07p01_full <- lm(formula = fmlm_full, data = tbl_ex07p01)
smry_ex07p01_full <- summary(lm_ex07p01_full)
n_sd_full <- smry_ex07p01_full$sigma
n_ssqr_full <- crossprod(residuals(lm_ex07p01_full))
n_ssqr_full
```

```
##          [,1]
## [1,] 1237.898
```

- Eliminate the variable that increases the residual sum of squares the least and compute C_p for resulting model

From the full model select one variable at the time, remove that variable, fit a reduced model and compute for that model the residual sum of squares. The model that increases the residual sum of squares the least, is selected and for that model the C_p value is compute.

```r
tbl_belim_res <- NULL
for (p in vec_pred_full){
  fm_update_cur <- as.formula(paste0(". ~ . - ", p, collapse = ""))
  lm_cur <- update(lm_ex07p01_full, fm_update_cur)
  vec_res <- residuals(lm_cur)
  tbl_cur <- tibble::tibble(Variable = p,
                            RSSQ = crossprod(vec_res))
  if (is.null(tbl_belim_res)){
    tbl_belim_res <- tbl_cur
  } else {
    tbl_belim_res <- dplyr::bind_rows(tbl_belim_res, tbl_cur)
  }
}
tbl_belim_res
```

```
## # A tibble: 4 x 2
##   Variable RSSQ[,1]
##   <chr>       <dbl>
## 1 BC          1576.
## 2 HEI         1290.
## 3 BCS         1581.
## 4 Breed       2821.
```

From `tbl_belim_res`, we determine the variable which is excluded

```r
n_idx_var_exclude <- which(tbl_belim_res$RSSQ == min(tbl_belim_res$RSSQ))
s_var_exclude <- tbl_belim_res$Variable[n_idx_var_exclude]
s_var_exclude
```

```
## [1] "HEI"
```

The model after this first round of elimination corresponds to the model that results when taking away the variable HEI from the full model.

```
vec_pred_cur <- setdiff(vec_pred_full, s_var_exclude)
fm_cur <- as.formula(paste0(s_resp, " ~ ",
                              paste0(vec_pred_cur, collapse = " + "),
                              collapse = ""))
lm_cur <- lm(formula = fm_cur, data = tbl_ex07p01)
summary(lm_cur)
```

```
##
## Call:
## lm(formula = fm_cur, data = tbl_ex07p01)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.3039  -3.4462  -0.8291   3.6745  14.4923
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -18.8251   210.7066  -0.089  0.92962
## BC               2.7650     1.1849   2.333  0.02916 *
## BCS2             6.8468     5.4061   1.267  0.21858
## BCS3            -0.2888     4.7594  -0.061  0.95216
## BCS4             6.9643     4.9052   1.420  0.16969
## BCS5             1.4942     5.3126   0.281  0.78114
## BreedLimousin   29.7294     5.9290   5.014 5.09e-05 ***
## BreedSimmental  13.2744     4.6980   2.826  0.00985 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.656 on 22 degrees of freedom
## Multiple R-squared:  0.8818, Adjusted R-squared:  0.8442
## F-statistic: 23.45 on 7 and 22 DF,  p-value: 8.216e-09
```

For the current model, we have to compute the C_p value

```
n_nr_obs <- nrow(tbl_ex07p01)
n_rssq <- crossprod(residuals(lm_cur))
# model size is the number of predictors plus the intercept
n_model_size <- length(vec_pred_cur) + 1
n_cp_cur <- n_rssq / (n_sd_full^2) - n_nr_obs + 2 * n_model_size
n_cp_cur
```

```
##            [,1]
## [1,] -0.1244789
```

Verify, according to https://search.r-project.org/CRAN/refmans/olsrr/html/ols_mallows_cp.html

```
olsrr::ols_mallows_cp(lm_cur, lm_ex07p01_full)
```

```
## [1] -0.1244789
```

- Repeat above step until all variables and factors are elminiated. The repetition could be done sequentially, but it is more efficient to do it in a loop. Inside of this loop, we have to perform several steps. For a better overview, we encapsulate these step in functions. The first function takes a model and returns a submodel with the one predictor variable or factor less such that the residual standard error increases the least. The second function is going to compute the C_p value for a given sub-model and a full model.

```r
get_subm_back <- function(plm_cur_model){
  # minimal value for RSSQ
  n_rssq_min <- NULL
  lm_result_sub <- NULL
  # obtain the vector of predictor variables and factors
  vec_pred_cur <- attr(terms(plm_cur_model), "term.labels")
  # loop over vector of predictors and compute RSSQ for each sub-model
  for (p in vec_pred_cur){
    # remove p from predictors
    fm_cur_subm <- as.formula(paste0(". ~ . - ", p, collapse = ""))
    lm_cur_subm <- update(plm_cur_model, fm_cur_subm)
    vec_res_subm <- residuals(lm_cur_subm)
    n_rssq_subm <- crossprod(vec_res_subm)
    # check whether n_rssq_sub is minimal
    if (is.null(n_rssq_min)){
      n_rssq_min <- n_rssq_subm
      lm_result_sub <- lm_cur_subm
    } else {
      if (n_rssq_subm < n_rssq_min){
        n_rssq_min <- n_rssq_subm
        lm_result_sub <- lm_cur_subm
      }
    }
  }
  # return model with minimal rssq
  return(lm_result_sub)
}
```

The function `get_subm_back()` can be verified by a call with the full model. Then the sub-model with `HEI` eliminated should result.

```r
lm_ex07p01_first_subm <- get_subm_back(plm_cur_model = lm_ex07p01_full)
lm_ex07p01_first_subm
```

```
##
## Call:
## lm(formula = BW ~ BC + BCS + Breed, data = tbl_ex07p01)
##
## Coefficients:
##    (Intercept)              BC            BCS2             BCS3             BCS4             BCS5   Bre
##       -18.8251          2.7650          6.8468          -0.2888           6.9643           1.4942
```

The second function computes the C_p value for the obtained sub-model.

```r
compute_cp_value <- function(pn_res_sd_full_model, pn_nr_obs, plm_cur_model){
  n_rssq <- crossprod(residuals(plm_cur_model))
  # model size is the number of predictors plus the intercept
  vec_pred_cur <- attr(terms(plm_cur_model), "term.labels")
  n_model_size <- length(vec_pred_cur) + 1
  n_cp_cur <- n_rssq / (pn_res_sd_full_model^2) - pn_nr_obs + 2 * n_model_size
  return(n_cp_cur)
}
```

For the first submodel, we get

```r
compute_cp_value(pn_res_sd_full_model = n_sd_full,
                 pn_nr_obs = nrow(tbl_ex07p01),
                 plm_cur_model = lm_ex07p01_first_subm)
```

```
##             [,1]
## [1,] -0.1244789
```

Now that we have the two functions ready, we can do the repetition of the elimination process of variables from a model. To make it a little bit easier, we start again with the full model.

```r
n_nr_obs <- nrow(tbl_ex07p01)
lm_current <- lm_ex07p01_full
n_sd_current <- summary(lm_current)$sigma
vec_pred_current <- attr(terms(lm_current), "term.labels")
# initialise a result dataframe
tbl_elim_result <- NULL
# loop as long as, there are variables in vec_pred_current
while (length(vec_pred_current) > 0){
  # get variables and C_p of current model
  tbl_elim_current <- tibble::tibble(`Current Model` = as.character(formula(lm_current))[3],
                                     Cp = compute_cp_value(pn_res_sd_full_model = n_sd_current,
                                                           pn_nr_obs = n_nr_obs,
                                                           plm_cur_model = lm_current))
  # store variables and C_p value of current model in result
  if (is.null(tbl_elim_result)) {
    tbl_elim_result <- tbl_elim_current
  } else {
    tbl_elim_result <- dplyr::bind_rows(tbl_elim_result, tbl_elim_current)
  }
  # get new submodel
  lm_current <- get_subm_back(plm_cur_model = lm_current)
  vec_pred_current <- attr(terms(lm_current), "term.labels")
}
tbl_elim_result
```

```
## # A tibble: 4 x 2
##   `Current Model`        Cp[,1]
##   <chr>                   <dbl>
## 1 BC + HEI + BCS + Breed  1.00
## 2 BC + BCS + Breed       -0.124
## 3 BC + Breed              3.07
## 4 Breed                   5.69
```

In the above shown result dataframe, the model which only fits an intercept is missing. Hence, we add that model to the results

```r
lm_inter <- lm(BW ~ 1, data = tbl_ex07p01)
tbl_elim_inter <- tibble::tibble(`Current Model` = as.character(formula(lm_inter))[3],
                                 Cp = compute_cp_value(pn_res_sd_full_model = n_sd_current,
                                                       pn_nr_obs = n_nr_obs,
                                                       plm_cur_model = lm_inter))
tbl_elim_result <- dplyr::bind_rows(tbl_elim_result, tbl_elim_inter)
tbl_elim_result
```

```
## # A tibble: 5 x 2
##   `Current Model`         Cp[,1]
##   <chr>                       <dbl>
## 1 BC + HEI + BCS + Breed      1.00
## 2 BC + BCS + Breed           -0.124
## 3 BC + Breed                  3.07
## 4 Breed                       5.69
## 5 1                         157.
```

- Select the model with the smallest C_p value. The model with the smallest C_p value

```
n_model_idx <- which(tbl_elim_result$Cp == min(tbl_elim_result$Cp))
tbl_elim_result[n_model_idx,]
```

```
## # A tibble: 1 x 2
##   `Current Model`  Cp[,1]
##   <chr>             <dbl>
## 1 BC + BCS + Breed -0.124
```

Forward Selection In forward selection, we start with the smallest model with only an intercept. Based on the preparation for backward selection, we can start with the iteration after an initialisation of the current model with the smallest model. The major difference between forward selection and backward selection is the way how subsequent submodels are generated. In forward selection, predictor variables or factors are added. This is done in a function called `get_subm_forward()`.

```
get_subm_forward <- function(plm_cur_model, pvec_pred_full){
  # minimal value for RSSQ
  n_rssq_min <- NULL
  lm_result_sub <- NULL
  # loop over vector of predictors and compute RSSQ for each sub-model
  for (p in pvec_pred_full){
    # remove p from predictors
    fm_cur_subm <- as.formula(paste0(". ~ . + ", p, collapse = ""))
    lm_cur_subm <- update(plm_cur_model, fm_cur_subm)
    vec_res_subm <- residuals(lm_cur_subm)
    n_rssq_subm <- crossprod(vec_res_subm)
    # check whether n_rssq_sub is minimal
    if (is.null(n_rssq_min)){
      n_rssq_min <- n_rssq_subm
      lm_result_sub <- lm_cur_subm
    } else {
      if (n_rssq_subm < n_rssq_min){
        n_rssq_min <- n_rssq_subm
        lm_result_sub <- lm_cur_subm
      }
    }
  }
  # return model with minimal rssq
  return(lm_result_sub)
}
```

The above function can be used in the iterative process of forward selection

```
# initialise current model
lm_current_forward <- lm(BW ~ 1, data = tbl_ex07p01)
n_sd_current_forward <- summary(lm_current_forward)$sigma
vec_pred_current_forward <- attr(terms(lm_current_forward), "term.labels")
n_nr_pred_fact <- length(vec_pred_full)
```

```r
tbl_result_forward <- NULL
# start iteration
while (length(vec_pred_current_forward) < n_nr_pred_fact){
  # results for current model
  tbl_cur_forward <- tibble::tibble(`Current Model` = as.character(formula(lm_current_forward))[3],
                                    Cp = compute_cp_value(pn_res_sd_full_model = n_sd_current,
                                                          pn_nr_obs = n_nr_obs,
                                                          plm_cur_model = lm_current_forward))
  # collect result
  if (is.null(tbl_result_forward)){
    tbl_result_forward <- tbl_cur_forward
  } else {
    tbl_result_forward <- dplyr::bind_rows(tbl_result_forward, tbl_cur_forward)
  }
  # update current model
  lm_current_forward <- get_subm_forward(plm_cur_model = lm_current_forward, pvec_pred_full = vec_pred_
  vec_pred_current_forward <- attr(terms(lm_current_forward), "term.labels")

}
# add full model
tbl_cur_forward <- tibble::tibble(`Current Model` = as.character(formula(lm_current_forward))[3],
                                  Cp = compute_cp_value(pn_res_sd_full_model = n_sd_current,
                                                        pn_nr_obs = n_nr_obs,
                                                        plm_cur_model = lm_current_forward))
tbl_result_forward <- dplyr::bind_rows(tbl_result_forward, tbl_cur_forward)
tbl_result_forward
```

```
## # A tibble: 5 x 2
##   `Current Model`         Cp[,1]
##   <chr>                    <dbl>
## 1 1                       157.
## 2 Breed                     5.69
## 3 Breed + BC                3.07
## 4 Breed + BC + BCS         -0.124
## 5 Breed + BC + BCS + HEI    1.00
```

The model with the lowest C_p value is

```r
n_model_idx <- which(tbl_result_forward$Cp == min(tbl_result_forward$Cp))
tbl_result_forward[n_model_idx,]
```

```
## # A tibble: 1 x 2
##   `Current Model`  Cp[,1]
##   <chr>             <dbl>
## 1 Breed + BC + BCS -0.124
```

Because the C_p values for backward selection and forward selection are negative, they cannot be used as estimates for the mean square error (MSE), because MSE must be positive. This indicates that C_p is not a good model selection criterion. Often, people just ignore the models with the negative C_p values and take the one that has the smallest positive C_p value besides the full model. In our case, this results in the model

```r
tbl_result_modified <- tbl_result_forward[tbl_result_forward$Cp > 1, ]
tbl_result_modified[tbl_result_modified$Cp == min(tbl_result_modified$Cp),]
```

```
## # A tibble: 1 x 2
##   `Current Model` Cp[,1]
##   <chr>             <dbl>
## 1 Breed + BC         3.07
```

It might be worth while to use AIC or BIC as alternative criteria. In R the function `MASS::stepAIC()` can be used to do model selection based on AIC.

```
MASS::stepAIC(lm_ex07p01_full)
```

```
## Start:  AIC=129.6
## BW ~ BC + HEI + BCS + Breed
## 
##         Df Sum of Sq    RSS    AIC
## - HEI    1     51.61 1289.5 128.82
## - BCS    4    343.53 1581.4 128.95
## <none>               1237.9 129.60
## - BC     1    338.00 1575.9 134.84
## - Breed  2   1583.34 2821.2 150.31
## 
## Step:  AIC=128.82
## BW ~ BC + BCS + Breed
## 
##         Df Sum of Sq    RSS    AIC
## - BCS    4    306.43 1595.9 127.22
## <none>               1289.5 128.82
## - BC     1    319.16 1608.7 133.46
## - Breed  2   1576.02 2865.5 148.78
## 
## Step:  AIC=127.22
## BW ~ BC + Breed
## 
##         Df Sum of Sq    RSS    AIC
## <none>               1595.9 127.22
## - BC     1    271.97 1867.9 129.94
## - Breed  2   2138.17 3734.1 148.72

## 
## Call:
## lm(formula = BW ~ BC + Breed, data = tbl_ex07p01)
## 
## Coefficients:
##    (Intercept)              BC   BreedLimousin  BreedSimmental  
##         23.036           2.542          32.675          14.414
```

The result of `MASS::stepAIC()` also shows the variable `BC` and the factor `Breed` to be important. This means the following model would be the best model that is selected from the data.

```
lm_ex07p01_best <- lm(BW ~ BC + Breed, data = tbl_ex07p01)
(smry_ex07p01_best <- summary(lm_ex07p01_best))
```

```
## 
## Call:
## lm(formula = BW ~ BC + Breed, data = tbl_ex07p01)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -15.9567  -5.2240  -0.6697   4.9009  17.2184 
## 
```

```
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       23.036    214.647   0.107  0.91536
## BC                 2.542      1.208   2.105  0.04511 *
## BreedLimousin     32.675      5.846   5.589 7.15e-06 ***
## BreedSimmental    14.414      4.756   3.031  0.00546 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.835 on 26 degrees of freedom
## Multiple R-squared:  0.8537, Adjusted R-squared:  0.8368
## F-statistic: 50.58 on 3 and 26 DF,  p-value: 5.459e-11
```

The model that was used to generate the data is the model with only `BC` and an intercept. Hence the true model is

```
lm_ex07p01_true <- lm(BW ~ BC, data = tbl_ex07p01)
summary(lm_ex07p01_true)
```

```
##
## Call:
## lm(formula = BW ~ BC, data = tbl_ex07p01)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -30.7130  -5.7404  -0.5809   5.1293  22.7049
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -891.240    189.128  -4.712 6.09e-05 ***
## BC             7.712      1.051   7.336 5.48e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.55 on 28 degrees of freedom
## Multiple R-squared:  0.6578, Adjusted R-squared:  0.6455
## F-statistic: 53.81 on 1 and 28 DF,  p-value: 5.482e-08
```

This shows the difficulty in the analysis of data when there are correlated variables and factors. As it seams they make it almost impossible to find the true model. But in any case there is no problem with the best model, it is a very good model and it is able to explain 83.7 percent of the variation in the response.

Problem 2: Verification of Model Selection Results

Use the R-package `olsrr` to verify the results of Problem 1. Have a look at the documentation of `olsrr` at https://github.com/rsquaredacademy/olsrr. In a first step, we are going to read the data from

```
## https://charlotte-ngs.github.io/asmss2022/data/asm_bw_mod_sel.csv
```

Solution

- Reading the data and convert factor columns to factor data-types

```
tbl_ex07p02 <- readr::read_csv(file = s_ex07p02_path)
```

```
## Rows: 30 Columns: 6
```

```
## -- Column specification --------------------------------------------------------------------------
## Delimiter: ","
## chr (1): Breed
## dbl (5): Animal, BC, BW, HEI, BCS

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
tbl_ex07p02$BCS <- as.factor(tbl_ex07p02$BCS)
tbl_ex07p02$Breed <- as.factor(tbl_ex07p02$Breed)
tbl_ex07p02
```

```
## # A tibble: 30 x 6
##    Animal    BC    BW   HEI BCS   Breed
##     <dbl> <dbl> <dbl> <dbl> <fct> <fct>
##  1      1  179.  475.  132. 1     Angus
##  2      2  177.  479.  130. 3     Angus
##  3      3  177.  469.  127. 1     Angus
##  4      4  181.  516.  132. 2     Limousin
##  5      5  180.  490.  130. 1     Simmental
##  6      6  184.  522.  129. 3     Limousin
##  7      7  182.  504.  130. 2     Simmental
##  8      8  178.  484.  130. 5     Angus
##  9      9  182.  485.  129. 3     Simmental
## 10     10  180.  494.  129. 4     Simmental
## # ... with 20 more rows
```

- Fitting the full model

```
lm_ex07p02_full <- lm(BW ~ BC + HEI + BCS + Breed, data = tbl_ex07p02)
```

- Run the model selection

```
olsrr::ols_step_best_subset(lm_ex07p02_full)
```

```
##     Best Subsets Regression
## -------------------------------
## Model Index    Predictors
## -------------------------------
##      1         Breed
##      2         BC Breed
##      3         BC BCS Breed
##      4         BC HEI BCS Breed
## -------------------------------
##
##                                                      Subsets Regression Summary
## ----------------------------------------------------------------------------------------------------
##                        Adj.        Pred
## Model    R-Square    R-Square    R-Square     C(p)        AIC         SBIC         SBC         MSEP
## ----------------------------------------------------------------------------------------------------
##   1        0.8288      0.8161      0.7886     5.6876     217.0776    129.7226    222.6824    2001.656
##   2        0.8537      0.8368      0.8003     3.0738     214.3568    127.8456    221.3628    1775.988
##   3        0.8818      0.8442      0.7894    -0.1245     215.9609    125.5199    228.5716    1492.390
##   4        0.8865      0.8433      0.7848     1.0000     216.7355    127.5176    230.7475    1492.355
## ----------------------------------------------------------------------------------------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

The results of `olsrr::ols_step_best_subset()` are consistent with our calculation of Problem 1.