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

```
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: 50 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_ex07p01$BCS <- as.factor(tbl_ex07p01$BCS)
tbl_ex07p01$Breed <- as.factor(tbl_ex07p01$Breed)
```

- Start with the full model considering all variables

```
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,] 3293.537
```

- 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.

```
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          5448.
## 2 HEI         3317.
## 3 BCS         3538.
## 4 Breed       6956.
```

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

```
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
## -18.657  -4.370   1.399   4.807  18.849
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -322.5198   146.9749  -2.194  0.03379 *
## BC                4.4441     0.8265   5.377 3.11e-06 ***
## BCS2              0.8595     3.7447   0.230  0.81957
## BCS3              0.6574     4.1381   0.159  0.87454
## BCS4              4.9026     4.1095   1.193  0.23957
## BCS5              5.2990     4.5029   1.177  0.24590
## BreedLimousin    32.2864     4.8362   6.676 4.23e-08 ***
## BreedSimmental   11.8968     3.7201   3.198  0.00263 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.887 on 42 degrees of freedom
## Multiple R-squared:  0.8924, Adjusted R-squared:  0.8745
## F-statistic: 49.78 on 7 and 42 DF,  p-value: < 2.2e-16
```

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,] -0.7023619
```

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] -0.7023619
```

- 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.

```
get_subm_elim <- 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_elim()` can be verified by a call with the full model. Then the sub-model with `HEI` eliminated should result.

```
lm_ex07p01_first_subm <- get_subm_elim(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
##      -322.5198          4.4441          0.8595          0.6574          4.9026          5.2990
```

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

```
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

```
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,] -0.7023619
```

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.

```
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_elim(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
## 2 BC + BCS + Breed       -0.702
## 3 BC + Breed              0.212
## 4 Breed                  34.2
```

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

```
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
## 2 BC + BCS + Breed        -0.702
## 3 BC + Breed               0.212
## 4 Breed                   34.2
## 5 1                      336.
```

- 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 -0.702
```

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.

Solution