Lab 4: Inference after variable selection in regression

Grading: Turn in your first attempt at the tasks for a binary “fair attempt or not” grade on Canvas. That is, your first attempt need not be neat or correct.

Done early? Come discuss your work with me and I’ll give you one suggestion on how to improve what you’ve got in the remaining time. Iterate if necessary.

Scientific Context

Suppose that you are working with a collaborator who is interested in identifying risk factors associated with a continuous outcome. The design of the study recruits 100 patients and measures 25 continuous covariates. They tell you that they broadly plan to analyze the study data by:

- Using forward stepwise to select variables of interest
- Fitting linear regression to the selected variables and reporting the p-values.

Today you will investigate the selective type I error rate and coverage of this approach, to help you advise your collaborator on their data analysis plan.

Set-up functions

Data generation

I’ve generated an arbitrary fixed, continuous design matrix X by sampling from a multivariate normal distribution. **Do not at any point repeat this sampling step in your simulation! That will make it random-X rather than fixed-X regression.**

```
n <- 100
p <- 25

library(dplyr)

rho <- 0.3
Sigma <- (1-rho)*diag(p) + rho*matrix(1, p, p)

set.seed(123)
X <-  MASS::mvrnorm(n, rep(0, p), Sigma) %>%
    as_tibble(.name_repair = \(x) stringr::str_c("X", 1:p))
```

The following is a function that randomly generates Y with $\mu_i = \mathbb{E}[Y_i] = 0.5[X_1]_i + 0.2[X_2]_i - 0.3[X_{10}]_i$.

```
generate_model_and_data <- function(X, scale_err_var) {
        response_and_mean <- X %>% rowwise() %>%
            mutate(mu = 0.5*X1 +0.2*X3 - 0.3*X10,
                         y = mu + sqrt(1/scale_err_var)*rt(1, df=5)) %>% relocate(y, ever

        return(list(data = response_and_mean %>% select(-mu),
                                mu = response_and_mean %>% pull(mu)))
}

set.seed(1)
(model_and_data <- generate_model_and_data(X, 2.5))
```

```
$data
# A tibble: 100 × 26
# Rowwise:
        y      X1      X2     X3      X4     X5      X6      X7     X8      X9
    <dbl>   <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
 1 -0.364  0.185   0.569   0.587 -0.0984  1.63   0.647   0.906   0.578  1.02
 2 -0.734 -0.472  -0.494   0.741  0.585  -0.442 -0.0976  0.658   0.483  0.670
 3 -0.999 -2.42   -0.999  -1.95  -0.0703 -0.296  0.119  -0.408   0.485 -0.513
 4  0.251  1.17    0.214   0.814  0.374   0.383 -0.175  -2.25    0.568 -0.519
 5  0.114 -0.0527  0.638   1.95  -0.377   0.999 -0.903  -0.695  -0.338 -0.567
 6 -0.273 -1.43   -0.809  -1.95  -1.20   -1.25  -0.584  -0.372  -1.12  -0.349
 7 -0.239  0.713  -0.0712 -0.459 -0.135   0.583 -0.671   0.0880  1.11   1.11
 8  1.65   2.02    1.40    1.76   1.94    1.16   2.07    0.636  -0.113  1.44
 9  0.614  1.01    0.743   0.619  1.54   -1.98   0.899  -0.578  -0.186  0.459
10  1.07   1.11   -0.249   0.468 -0.365  -0.495  0.618   1.02   -0.919 -0.0930
# ℹ 90 more rows
# ℹ 16 more variables: X10 <dbl>, X11 <dbl>, X12 <dbl>, X13 <dbl>, X14 <dbl>,
#   X15 <dbl>, X16 <dbl>, X17 <dbl>, X18 <dbl>, X19 <dbl>, X20 <dbl>,
#   X21 <dbl>, X22 <dbl>, X23 <dbl>, X24 <dbl>, X25 <dbl>

$mu
  [1]  0.052399455 -0.355985512 -1.327314021  0.459753620  0.255122010
  [6] -0.626341035  0.008397927  1.657647311  0.134182147  0.558131372
 [11]  0.178888573  1.316449825 -0.656787716 -0.225103828  0.735819620
 [16] -0.201399227 -0.131997939 -0.475905296  0.559317076  0.264597606
 [21]  0.749159054 -0.204619284  0.364029808  0.202756251 -0.374000248
 [26]  0.011932554 -0.569395064 -0.112151421  0.202686846 -0.512098505
 [31]  0.740300170 -0.981828364 -0.080327360  0.087032378 -0.981286701
 [36] -1.190499625  0.092552871 -0.207400879 -0.045971322  0.709265062
 [41]  0.065782447  0.161891009  0.561270063  0.219402357 -0.328564197
 [46] -0.686582472 -0.240279897 -0.179212696  0.530471224  0.388582517
 [51]  0.395041154 -0.029297052  0.011277827 -0.350340093  0.153584093
 [56]  0.393955624  0.501289977  0.164334842 -0.417378462  0.722130015
 [61] -0.344451081 -0.167391437  0.857295412  0.299595423  1.081325107
 [66] -0.416577375  0.128700689  0.479208157  0.039293696 -0.340064901
 [71] -0.074233196  0.128201213  0.275195509  0.092138492 -0.579774857
```

```
 [76] -0.066391747  0.120546625  0.402204569 -0.439319601 -0.438566343
 [81]  0.209688630  0.119820390  0.463459233 -0.365785812 -0.004582268
 [86] -0.298064904 -0.453282429  0.311043705 -0.358510397 -0.111968330
 [91]  0.150223254 -0.825398043  0.770529402  0.158870350 -0.253914427
 [96]  0.080447086 -0.453091341 -0.296727021  0.379704619 -0.331818066
```

Forward stepwise and p-value calculation

Here is code that does 4 steps of forward stepwise on data, then does “agnostic linear regression” as discussed in class to produce p-values and confidence intervals for the regression of the outcome on the four (or fewer) selected variables.

```
fit_fs_4steps <- function(data) {
    empty_model <- lm(y ~ 1, data = data)
    best_after_fs <- step(empty_model, direction="forward",
                                            scope = formula(lm(y~., data=data)),
                                            steps=4, trace=0)

    results <- best_after_fs %>% broom::tidy()

    results$std.error <-  sqrt(diag(sandwich::vcovHC(best_after_fs)))

    results %>% filter(term != "(Intercept)") %>% mutate(vars = as.integer(stringr::str_r
                     p.value = 2*pnorm(-abs(statistic)),
                     ci.lower = estimate - qnorm(0.975)*std.error,
                     ci.upper = estimate + qnorm(0.975)*std.error)      %>% select(vars,

}

model_and_data %>% .[["data"]] %>% fit_fs_4steps()
```

```
# A tibble: 4 × 5
   vars estimate  p.value ci.lower ci.upper
  <int>    <dbl>    <dbl>    <dbl>    <dbl>
1     1    0.467 5.59e-12   0.334    0.600
2     3    0.154 2.93e- 2   0.0155   0.293
3    10   -0.254 2.71e- 3  -0.421   -0.0881
4    18   -0.143 9.42e- 2  -0.311    0.0245
```

```
(fs_results <- fit_fs_4steps(model_and_data$data))
```

```
# A tibble: 4 × 5
   vars estimate  p.value ci.lower ci.upper
  <int>    <dbl>    <dbl>    <dbl>    <dbl>
1     1    0.467 5.59e-12   0.334    0.600
2     3    0.154 2.93e- 2   0.0155   0.293
3    10   -0.254 2.71e- 3  -0.421   -0.0881
4    18   -0.143 9.42e- 2  -0.311    0.0245
```

Task 1

In task 2, you will be asked to set `scale_err_var` to 1, and calculate the selective type I error rate for $H_0 : [\beta^*_{\{1,3,7,10\}}]_4 = 0$ (i.e. the slope coefficient for variable 10) and the selective coverage for each of the three slope components of $\beta^*_{\{1,3,10\}}$ (i.e. the slope coefficient for variables 1 3 and 10).

Write some code that will take the results of the forward stepwise analysis on a data set that has mean response vector μ and output a table containing all the components you will need per simulation in order to calculate selective type I error and coverage.

Here's the skeleton of my function:

```
get_evaluations <- function(results, X, mu) {
    # YOUR CODE HERE
}

get_evaluations(fs_results, X, model_and_data$mu)
```

Task 2

Using the code that will be provided after the completion of Task 1 (or writing your own code if you prefer), calculate with `scale_err_var = 1`:

- the selective type I error rate for $H_0 : [\beta^*_{\{1,3,7,10\}}]_3 = 0$ (i.e. the slope coefficient for variable 7)
- the selective coverage for each of the three slope components of $\beta^*_{\{1,3,10\}}$ (i.e. the slope coefficient for variables 1 3 and 10)
- The probability of selecting each combination of 4 or fewer variables in $X_1, \ldots, X_{25}$.

Repeat with `scale_err_var = 100`.

Think about and comment on your findings.