Penalized Statistical Models: A Help for Managing Your Overparameterized Model

Statistical models: linear regression

We will introduce a glimpse of the theory through linear regression. However, the key concepts are easily extended to more complex contexts, without substantial changes.

We introduce the notation:

- We observe a sample of n observations;
- For each unit i, we observe a response y_i and some covariates $x_{i1}, x_{i2}, \dots, x_{ip}$;
- The response can be written as a vector $n \times 1$, the covariates as a matrix $n \times p$ or as a sequence of vectors $X_1, X_2, \dots, X_p$.

The standard linear regression model as the form

$$Y = X_1\beta_1 + X_2\beta_2 + \cdots + X_p\beta_p + \varepsilon$$

or in compact form

$$Y = X\beta + \varepsilon.$$

Whether you use the likelihood approach or the OLS approach (+ normal errors for statistical inference) the loss function to be minimize is

$$\min_{\beta_1,\dots,\beta_p} \sum_{i=1}^{n} (y_i - x_{i1}\beta_1 - \cdots - x_{ip}\beta_p)^2$$

which, in compact math formula, can be written as

$$\min_{\beta} \sum_{i=1}^{n} (y_i - x_i^T\beta)^2$$

or (this is called euclidean norm)

$$\min_{\beta} \|Y - X\beta\|^2.$$

Penalized statistical models

Penalized statistical models introduce a penalization on some or all the coefficients. Penalization "reduces" the dimension of the statistical model adopted in a "non-trivial" way. We can find (at least) three motivations to think about the use of penalization:

- when you have too many covariates (even $p > n$) it can be helpful to do variable selection;
- when you have too many covariates (or they become too many with the interactions) it can be helpful to have the convergence of the model and not inflated standard error;
- when you have cluster effects it can be helpful to have the convergence of the model.

Later we'll talk about the limits.

The minimization problem now has the form

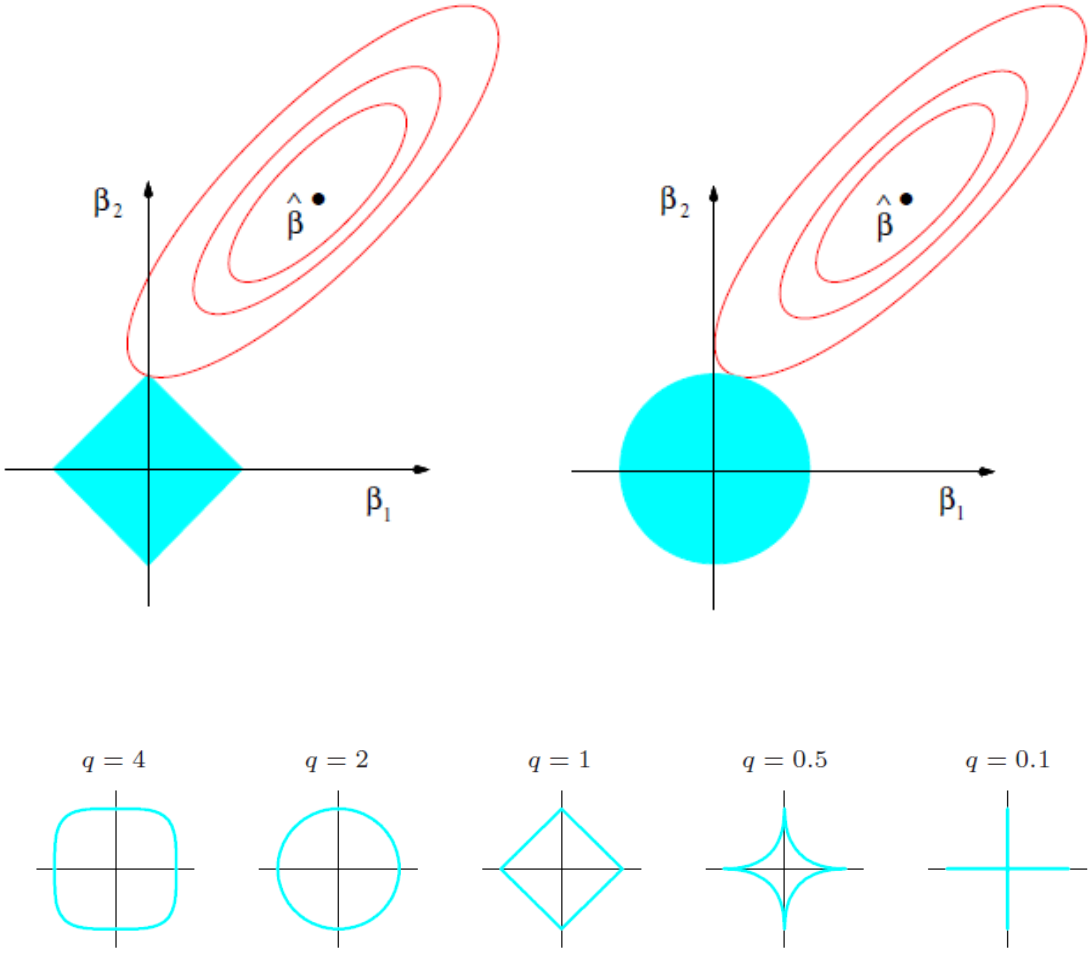
$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} X_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \right\}$$

or

$$\hat{\beta} = \arg\min_{\beta} \left(\|y - X\beta\|_2^2 + \lambda \|\beta\|_q^q \right)$$

Details can be found in the book *The elements of statistical learning* by Hastie *et al.*, Section 3.4. The most famous choices of q are

- $q = 1$ for the Lasso regression;
- $q = 2$ for the Ridge regression.



What do you do in practice? You shrink the overall magnitude of the coefficients. And, importantly, you can apply the penalization over a subset of all coefficients.

Problems:

- Choice of λ, which is a tuning parameter. There is no optimal way of selecting it. The most common option is cross-validation;
- Regression coefficients are biased (towards zero);
- Standard errors and p-values are meaningless.

How can we solve that?

- We can avoid shrinking the parameters of interest;

- Use of lasso on all or some of the coefficients. You can fit first the lasso, then refit the usual linear model with only the covariates with non-zero coefficients. Standard errors and p-values are hopefully decent;
- Use of any penalization on nuisance parameters: refit them in the second step as offsets. Standard errors and p-values must be computed with the bootstrap (degrees of freedom are not computed correctly);
- More complex modeling (*e.g.* Debiased lasso).

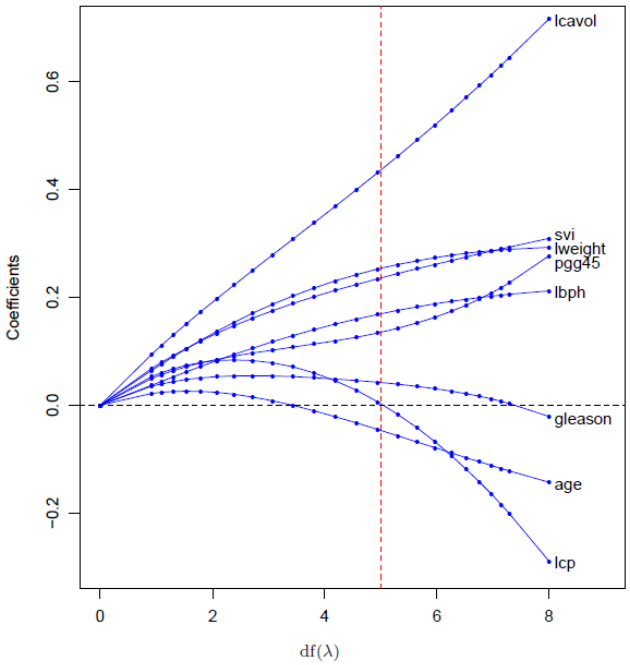
Example (taken from the book of Hastie et al.)

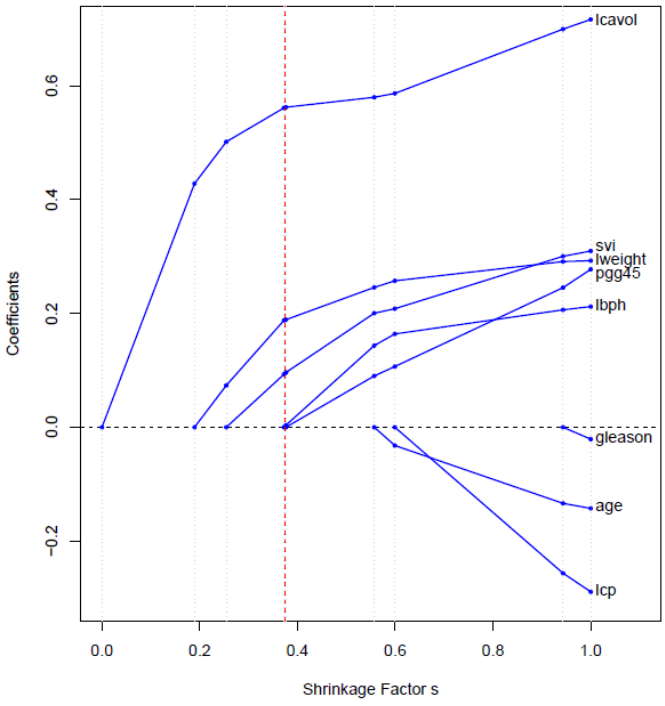
3.2.1 Example: Prostate Cancer

The data for this example come from a study by Stamey et al. (1989). They examined the correlation between the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy. The variables are log cancer volume (`lcavol`), log prostate weight (`lweight`), `age`, log of the amount of benign prostatic hyperplasia (`lbph`), seminal vesicle invasion (`svi`), log of capsular penetration (`lcp`), Gleason score (`gleason`), and percent of Gleason scores 4 or 5 (`pgg45`).

TABLE 3.2. *Linear model fit to the prostate cancer data. The Z score is the coefficient divided by its standard error (3.12). Roughly a Z score larger than two in absolute value is significantly nonzero at the $p = 0.05$ level.*

Term	Coefficient	Std. Error	Z Score
`Intercept`	2.46	0.09	27.60
`lcavol`	0.68	0.13	5.37
`lweight`	0.26	0.10	2.75
`age`	−0.14	0.10	−1.40
`lbph`	0.21	0.10	2.06
`svi`	0.31	0.12	2.47
`lcp`	−0.29	0.15	−1.87
`gleason`	−0.02	0.15	−0.15
`pgg45`	0.27	0.15	1.74





We will now see three examples with simple synthetic data and one example with a more

complex dataset.

Example 1:

I have $n = 50$ observations and $p = 30$ confounders, with some non-zero correlations. I want to fit a model with all the confounders. A simple linear model will have inflated variances. A two-step approach can be done in ***R*** with

```
#Data generation
library(MASS)

set.seed(1)

corxx <- 0.5 #equi-correlation for all the covariates
n <- 50 #sample size
npar <- 30 #number of parameters
mu.vect <- c(rep(0, npar)) #mean vector of covariates
sigma.mat <- matrix(corxx, npar, npar ) #covariance matrix of covariates
diag(sigma.mat) <- 1 #marginal variances of covariates
X <- mvrnorm(n, mu.vect, sigma.mat) #generate covariates
y <- rnorm(n, 0.5*X[,1]+0.1*X[,2],1) #generate response
dat <- data.frame(y, X)

head(dat, 3)
```

```
           y         X1         X2        X3        X4         X5           X6
1  0.9445473  0.0851827  0.5191248 0.6408947 0.8436839  1.0428215  0.005321089
2 -1.5444795 -1.0901368 -0.7409806 0.1330469 0.5770252 -0.8958749 -0.087232360
3  1.3806814  0.7954234  0.8938855 1.3955671 2.4273029  0.4899190  1.631827613
         X7         X8         X9       X10        X11        X12        X13
1 0.9663489  0.6990478  0.1727148 1.3254987  1.9022230  1.5021431  0.9036577
2 0.5513086 -0.9696532  0.5195927 1.8343541 -0.7691069 -0.1287008 -0.6208245
3 1.1730006 -0.3756451 -0.2260493 0.4080592  0.6375067  0.8196479  0.1547507
         X14        X15        X16        X17        X18       X19         X20
1 -0.6806188 -0.2766938  0.6866682  0.1362971 -0.6082909 0.6006193  0.08025368
2 -0.3942242 -0.1329056 -0.1242716 -0.7913948  1.1426524 0.1823005 -0.12448575
3  0.2601741  0.8894793 -0.1990132  0.6750766 -0.5307901 1.3682903  0.74227354
          X21        X22        X23        X24        X25        X26        X27
1  0.38523550  0.9155030 0.42280171 -1.0018276  0.1914517  0.7531948  0.1792432
2 -0.53180167  0.3695072 0.24126505 -0.6319879 -0.1022301 -0.9702015  0.5085255
3 -0.08358785 -0.1782449 0.08851921  1.7238312  0.1940070  1.3230480 -0.3856783
```

```
        X28        X29        X30
1  1.087733 -0.8287921  0.8573205
2 -1.341724 -0.3708307  0.7989308
3  1.743460  0.4086203 -0.2452837
```

```r
#Linear regression model

mod <- lm(y ~ . , data=dat)
head(summary(mod)$coefficients, 10)
```

```
             Estimate Std. Error    t value   Pr(>|t|)
(Intercept)  0.3648778  0.2208680  1.6520176 0.11496153
X1           0.7518655  0.2799468  2.6857445 0.01463208
X2          -0.3416268  0.3651739 -0.9355183 0.36126033
X3          -0.1916749  0.2607915 -0.7349738 0.47133063
X4           0.5347888  0.2962070  1.8054563 0.08687443
X5           0.2695380  0.3121936  0.8633681 0.39870300
X6          -0.5895717  0.3506146 -1.6815378 0.10902270
X7           0.2143118  0.3023581  0.7088011 0.48705454
X8           0.3062780  0.2980983  1.0274395 0.31711692
X9           0.7469112  0.3349418  2.2299729 0.03800732
```

```r
#Penalized regression model (glm allowed)
#install.packages(glmnet)
library(glmnet)

# The function glmnet gives the estimates for a grid of values for lambdas
# The function cv.glmnet makes the choice of lambda for you by cross-validation

fit <- cv.glmnet(x = model.matrix(mod)[,-1], #design matrix
                 y = y, #response vector
                 alpha = 1, #1 for lasso, 0 for ridge, intermediate value for mixture
                 penalty.factor = rep(1, 30)) #change to zero (in the correct indexes) to av

head(coef(fit, s = "lambda.min"), 10) #s is the lambda (shrinkage) parameter. Options are -)
```

```
10 x 1 sparse Matrix of class "dgCMatrix"
            lambda.min
(Intercept)  0.1078511
X1           0.4687275
X2           .
```

```
X3           .
X4           0.2043753
X5           0.2354020
X6          -0.1244027
X7           .
X8           .
X9           .
```

Example 1 - Simulation

Imagine the setting before with $5,000$ replies. A simulation study will investigate how $\beta_1 = 0.5$ and $\beta_2 = 0.1$ are estimated and tested over repeated samples.

```
df <- readRDS("Sim2.rds") #upload simulation output

#average mean without penalization
round(apply(df[[3]],2,mean),3)
```

```
[1] 0.507 0.102
```

```
#average mean with penalization
round(apply(df[[4]],2,mean),3)
```

```
[1] 0.342 0.048
```

```
#proportion of rejection without penalization (alpha=0.01)
round(apply(df[[1]],2,function(x)mean(x < 0.01)),3)
```

```
[1] 0.142 0.013
```

```
#proportion of rejection with penalization (alpha=0.01)
round(apply(df[[2]],2,function(x)mean(x < 0.01)),3)
```

```
[1] 0.315 0.020
```

```
#proportion of rejection without penalization (alpha=0.05)
round(apply(df[[1]],2,function(x)mean(x < 0.05)),3)
```

```
[1] 0.334 0.057
```

```r
#proportion of rejection with penalization (alpha=0.05)
round(apply(df[[2]],2,function(x)mean(x < 0.05)),3)
```

```
[1] 0.475 0.050
```

```r
#proportion of rejection without penalization (alpha=0.10)
round(apply(df[[1]],2,function(x)mean(x < 0.10)),3)
```

```
[1] 0.462 0.113
```

```r
#proportion of rejection with penalization (alpha=0.10)
round(apply(df[[2]],2,function(x)mean(x < 0.10)),3)
```

```
[1] 0.558 0.073
```

Example 2:

I have $n = 50$ observations and $p = 4$ confounders, with some non-zero correlations. I want to fit a model with all the interactions, but I am really interested only in marginal effects. I include the interactions to avoid (or at least reduce) possible biases.

```r
#Generate data
set.seed(2)

corxx <- 0.5
n <- 50
npar <- 4
mu.vect <- c(rep(0, npar))
sigma.mat <- matrix(corxx, npar, npar)
diag(sigma.mat) <- 1
X <- mvrnorm(n, mu.vect, sigma.mat)
y <- rnorm(n, 0.5*X[,3]+0.5*X[,4]+0.5*X[,3]*X[,4],1)
dat <- data.frame(y, X)
mod <- lm(y ~ X1*X2*X3*X4, data=dat)

head(summary(mod)$coefficients, 10)
```

```
               Estimate Std. Error     t value   Pr(>|t|)
(Intercept)  0.019539005  0.2071385  0.09432821 0.92540216
X1          -0.178155209  0.2062088 -0.86395549 0.39366664
X2           0.098043902  0.2571784  0.38122916 0.70540519
X3           0.548110051  0.2872324  1.90824615 0.06483242
X4           0.402183724  0.2350559  1.71101341 0.09618914
X1:X2        0.193697821  0.2021930  0.95798458 0.34483349
X1:X3       -0.088648844  0.3016178 -0.29391115 0.77061261
X2:X3       -0.169162466  0.4420505 -0.38267682 0.70434135
X1:X4        0.481172703  0.2779745  1.73099584 0.09251971
X2:X4        0.004587679  0.4187686  0.01095516 0.99132325
```

```
#Penalization only on random effects (see penalty.factor below)
fit <- cv.glmnet(x = model.matrix(mod)[,-1], y = y, alpha = 1, penalty.factor=c(rep(0,4),rep
head(coef(fit, s = "lambda.min"), 10)
```

```
10 x 1 sparse Matrix of class "dgCMatrix"
             lambda.min
(Intercept)  0.17989429
X1          -0.02405605
X2           0.10546593
X3           0.62050076
X4           0.29656730
X1:X2        0.02177734
X1:X3        .
X2:X3        .
X1:X4        0.28048677
X2:X4        .
```

Example 2 - Simulation

Imagine the setting above with $5,000$ replies. A simulation study will investigate how one of the main effects ($\beta_a = 0.5$) is estimated. As above, we have a nuisance effect equal to 0.5 of the interaction among the target covariate X_a and another X_b .

```
df <- readRDS("Sim1.rds")  #upload simulation output

#average mean without penalization
round(mean(df[[3]]),3)
```

```
[1] 0.5
```

```r
#average mean with penalization
round(mean(df[[4]]),3)
```

```
[1] 0.496
```

```r
#proportion of rejection without penalization (alpha=0.01)
round(mean(df[[1]] < 0.01),3)
```

```
[1] 0.229
```

```r
#proportion of rejection with penalization (alpha=0.01)
round(mean(df[[2]] < 0.01),3)
```

```
[1] 0.45
```

```r
#proportion of rejection without penalization (alpha=0.05)
round(mean(df[[1]] < 0.05),3)
```

```
[1] 0.452
```

```r
#proportion of rejection with penalization (alpha=0.05)
round(mean(df[[2]] < 0.05),3)
```

```
[1] 0.672
```

```r
#proportion of rejection without penalization (alpha=0.10)
round(mean(df[[1]] < 0.10),3)
```

```
[1] 0.579
```

```r
#proportion of rejection with penalization (alpha=0.10)
round(mean(df[[2]] < 0.10),3)
```

```
[1] 0.77
```

Example 3

Imagine we have 30 observations, on each we observe 4 repeated measures. A cluster effects model seems appropriate. Code is AI generated :)

```
library(lme4)
library(plm)
library(lmerTest)

set.seed(1)

#Data generation
n_units <- 30  #number of units
n_times <- 4   #number of times each subject is observed
n_total <- n_units * n_times
ID <- rep(1:n_units, each = n_times)
Time <- rep(0:(n_times - 1), times = n_units)

beta_0 <- 1      #intercept
beta_time <- 0    #beta referred to time
p_x1 <- 0.3      #first fixed effect
p_x2 <- 0       #second fixed effect

X1 <- rnorm(n_total, mean = 0, sd = 1) #first covariate
X2 <- rnorm(n_total, mean = 0, sd = 1) #second covariate

u_i_unique <- rnorm(n_units, mean = 0, sd = 3) #random effect
u_i <- rep(u_i_unique, each = n_times)
epsilon <- rnorm(n_total, mean = 0, sd = 1)

Y <- beta_0 + (beta_time * Time) +
  (p_x1 * X1) + (p_x2 * X2) + u_i + epsilon #response

df <- data.frame(ID, Time, X1, X2, Y)

#Random effects model
mod_random <- lmer(Y ~ Time + X1 + X2 + (1 | ID), data = df)
summary(mod_random)$coefficients
```

```
             Estimate Std. Error       df   t value     Pr(>|t|)
(Intercept) 1.06254081 0.52487800 32.31302 2.0243577 5.126491e-02
Time        0.02741646 0.08056636 87.01775 0.3402966 7.344533e-01
```

```
X1          0.47563555 0.11647739 88.62946 4.0835011 9.699916e-05
X2          0.10010697 0.09957250 88.67032 1.0053676 3.174556e-01
```

```r
#Fixed effects model
mod_fixed <- plm(Y ~ Time + X1 + X2,
                 data = df,
                 index = c("ID"),
                 model = "within")

summary(mod_fixed)$coefficients
```

```
       Estimate Std. Error   t-value     Pr(>|t|)
Time 0.02775882 0.08057047 0.3445285 0.7312802915
X1   0.47111556 0.11702661 4.0257133 0.0001209637
X2   0.09010798 0.10005387 0.9005947 0.3702901170
```

```r
#Penalized fixed effects model + bootstrap for p-values
X_mat_full <- model.matrix(Y ~ factor(ID) + Time + X1 + X2 - 1, data = df)
cv_fit_full <- cv.glmnet(X_mat_full,
                         df$Y,
                         penalty.factor = c(rep(1, n_units), rep(0, 3)),
                         alpha = 1)

id_coefs_full <- as.matrix(coef(cv_fit_full,
                                s = "lambda.min"))[paste0("factor(ID)",
                                                          1:n_units), ]
df$offset_lasso <- id_coefs_full[df$ID]
mod_full <- lm(Y ~ Time + X1 + X2, data = df, offset = offset_lasso)

#Point estimates
coef(mod_full)
```

```
(Intercept)        Time          X1          X2
 1.27590199  0.02762981  0.47315525  0.09351499
```

We use a simple version of the bootstrap to compute the p-values: 1) we fit each time the two-step model on the bootstrapped sample; 2) For each parameter, we have *n_boot* estimates of each parameter; 3) we compute, for each parameter, how many replies are above and below 0 (tested null hypothesis); 4) the minimum (multiplied by 2) among the two proportions gives the p-value; 5) note that a small correction is applied to avoid a p-value of zero.

```r
#bootstrap part (helped by AI)
n_boot <- 200    #number of replicates used for bootstrap p-values

unique_ids <- unique(df$ID)
boot_coefs <- matrix(NA, nrow = n_boot, ncol = 2)
colnames(boot_coefs) <- c("X1", "X2")

for(b in 1:n_boot) {
  resampled_ids <- sample(unique_ids, n_units, replace = TRUE)
  df_boot <- do.call(rbind, lapply(1:n_units, function(j) {
    curr <- df[df$ID == resampled_ids[j], ]
    curr$ID <- j
    return(curr)
    }))

  X_mat_boot <- model.matrix(Y ~ factor(ID) + Time + X1 + X2 - 1,
                             data = df_boot)
  p_factors <- c(rep(1, n_units), rep(0, 3))

  try({cv_fit_boot <- cv.glmnet(X_mat_boot,
                                df_boot$Y,
                                penalty.factor = p_factors,
                                alpha = 1)
      id_coefs <- as.matrix(coef(cv_fit_boot,
                                 s = "lambda.min"))[paste0("factor(ID)",
                                                           1:n_units), ]
      df_boot$offset_lasso <- id_coefs[df_boot$ID]

      mod_boot <- lm(Y ~ Time + X1 + X2, data = df_boot, offset = offset_lasso)

      boot_coefs[b, ] <- coef(mod_boot)[c("X1", "X2")]
    }, silent = TRUE)
  }

  boot_coefs <- na.omit(boot_coefs)
  n_success <- nrow(boot_coefs)

  calc_p_val <- function(boot_dist) {
    p_up <- (sum(boot_dist > 0) + 1) / (n_success + 1)
    p_down <- (sum(boot_dist < 0) + 1) / (n_success + 1)

    return(2 * min(p_up, p_down))
```

```
  }

#Bootstrap p-values
calc_p_val(boot_coefs[, "X1"]);calc_p_val(boot_coefs[, "X2"])
```

```
[1] 0.009950249
```

```
[1] 0.3084577
```

Example 3 - Simulation

Imagine the setting before with 500 replies (bootstrap simulations require time). A simulation study will investigate how $\beta_1 = 0.3$ and $\beta_2 = 0$ are estimated and tested over repeated samples.

```
df <- readRDS("Sim3.rds")  #upload simulation output

#average mean first coefficient
rbind(c("Random", "Fixed", "Penalized"),
round(apply(df[[3]], 2, function(x)mean(x)),3))
```

```
     [,1]     [,2]    [,3]
[1,] "Random" "Fixed" "Penalized"
[2,] "0.307"  "0.307" "0.306"
```

```
#average mean second coefficient
rbind(c("Random", "Fixed", "Penalized"),
round(apply(df[[4]], 2, function(x)mean(x)),3))
```

```
     [,1]     [,2]    [,3]
[1,] "Random" "Fixed" "Penalized"
[2,] "0.005"  "0.005" "0.005"
```

```
#proportion of rejection first coefficient (alpha=0.01)
rbind(c("Random", "Fixed", "Penalized"),
round(apply(df[[1]], 2, function(x)mean(x < 0.01)),3))
```

```
     [,1]     [,2]    [,3]
[1,] "Random" "Fixed" "Penalized"
[2,] "0.594"  "0.586" "0.556"
```

```
#proportion of rejection second coefficient (alpha=0.01)
rbind(c("Random", "Fixed", "Penalized"),
round(apply(df[[2]], 2, function(x)mean(x < 0.01)),3))
```

```
     [,1]     [,2]    [,3]
[1,] "Random" "Fixed" "Penalized"
[2,] "0.008"  "0.008" "0.012"
```

```
#proportion of rejection first coefficient (alpha=0.05)
rbind(c("Random", "Fixed", "Penalized"),
round(apply(df[[1]], 2, function(x)mean(x < 0.05)),3))
```

```
     [,1]     [,2]    [,3]
[1,] "Random" "Fixed" "Penalized"
[2,] "0.82"   "0.81"  "0.808"
```

```
#proportion of rejection second coefficient (alpha=0.05)
rbind(c("Random", "Fixed", "Penalized"),
round(apply(df[[2]], 2, function(x)mean(x < 0.05)),3))
```

```
     [,1]     [,2]    [,3]
[1,] "Random" "Fixed" "Penalized"
[2,] "0.044"  "0.048" "0.06"
```

```
#proportion of rejection first coefficient (alpha=0.10)
rbind(c("Random", "Fixed", "Penalized"),
round(apply(df[[1]], 2, function(x)mean(x < 0.10)),3))
```

```
     [,1]     [,2]    [,3]
[1,] "Random" "Fixed" "Penalized"
[2,] "0.9"    "0.892" "0.89"
```

```
#proportion of rejection second coefficient (alpha=0.10)
rbind(c("Random", "Fixed", "Penalized"),
round(apply(df[[2]], 2, function(x)mean(x < 0.10)),3))
```

```
     [,1]     [,2]    [,3]
[1,] "Random" "Fixed" "Penalized"
[2,] "0.1"    "0.102" "0.12"
```

Real data (in italiano)

I nostri dati reali riguardano uno studio che durava 10 giorni lavorativi (2 settimane) in cui diverse variabili venivano misurate in specifici momenti della giornata + altre variabili misurate solo una volta con il questionario preliminare. La variabile target è il workaholism ("dipendenza da lavoro").

Le variabili within sono le seguenti (tutte misurate da 1 a 7):

- **jd** = job demands (carico di lavoro)
- **jc** = job control (autonomia decisionale)
- **sw** = supervisor overwork (es. oggi il mio capo lavorava eccessivamente)
- **wh** = workaholism (è l'outcome che vogliamo predire).

Le variabili between sono queste:

- **gender** (F/M)
- **age** (years)
- **job.tenure** (years)
- **mw** = meaningful work (punteggio aggregato 1-5)
- **oe** = overwork endorsement (punteggio aggregato 1-5)
- **ji** = job insecurity (punteggio aggregato 1-5)

In totale sono presenti 134 soggetti; il numero di misurazioni per soggetto è variabile (da 1 a 10). In totale si hanno 869 misurazioni.

Fittiamo i modelli elencati sotto, con le seguenti considerazioni: 1) le variabili within vengono prima centrate sulle rispettive medie; 2) dove viene fatta penalizzazione i p-values sono ottenuti con bootstrap con 1000 ripetizioni (più lento – 5-10 minuti a modello – ma più preciso); 3) nel secondo step di refit, gli effetti di cluster entrano come offsets, mentre quando le interazioni sono penalizzate quelle che "si salvano" vengono stimate nuovamente insieme agli effetti marginali.

I modelli considerati sono:

- Modello ad effetti casuali con solo effetti marginali *Rand Univ*;
- Modello ad effetti casuali con interazioni a due delle covariate *Rand Int*;

- Modello con effetti di gruppo fissi ma penalizzati, con solo effetti marginali *Pen Univ*;
- Modello con effetti di gruppo fissi ma penalizzati, con interazioni a due delle covariate *Pen Int*;
- Modello con effetti di gruppo fissi ma penalizzati, con interazioni a due delle covariate penalizzate *Pen Int 2*.

```
ddf <- readRDS("RealData.rds")
```

Risultati coefficienti effetti univariati

```
ddf$Est1[,-1]
```

```
            Rand_Univ Rand_Int Pen_Univ Pen_Int Pen_Int_2
(Intercept)     3.727    4.681    3.362   7.966     3.376
jd.cmc          0.291   -0.102    0.291  -0.102     0.193
jc.cmc         -0.047   -0.639   -0.047  -0.642    -0.102
sw.cmc          0.041    0.505    0.041   0.502     0.250
genderM         0.176    0.198    0.300  -0.214     0.300
age            -0.053   -0.015   -0.038  -0.135    -0.039
job.tenure      0.073    0.424    0.070   0.711     0.069
mw              0.020    0.719    0.067   0.976     0.067
oe              0.230    0.229    0.162   0.598     0.163
ji              0.103   -2.213    0.031  -4.517     0.033
```

P-values coefficienti effetti univariati

```
ddf$Pval1[,-1]
```

```
            Rand_Univ Rand_Int Pen_Univ Pen_Int Pen_Int_2
(Intercept)     0.001    0.562    0.038   0.418     0.521
jd.cmc          0.000    0.749    0.002   0.717     0.008
jc.cmc          0.048    0.041    0.068   0.040     0.026
sw.cmc          0.050    0.044    0.058   0.054     0.018
genderM         0.338    0.938    0.228   0.957     0.729
age             0.115    0.958    0.376   0.637     0.651
job.tenure      0.067    0.499    0.208   0.641     0.687
mw              0.829    0.521    0.731   0.917     0.973
oe              0.007    0.843    0.114   0.713     0.731
ji              0.519    0.322    0.719   0.204     0.843
```

Risultati coefficienti con interazioni a due

```
ddf$Est2
```

	Names	Rand_Int	Pen_Int	Pen_Int_2
1	(Intercept)	4.681	7.966	3.376
2	jd.cmc	-0.102	-0.102	0.193
3	jc.cmc	-0.639	-0.642	-0.102
4	sw.cmc	0.505	0.502	0.25
5	genderM	0.198	-0.214	0.3
6	age	-0.015	-0.135	-0.039
7	job.tenure	0.424	0.711	0.069
8	mw	0.719	0.976	0.067
9	oe	0.229	0.598	0.163
10	ji	-2.213	-4.517	0.033
11	jd.cmc:sw.cmc	-0.014	-0.013	-0.012
12	jd.cmc:genderM	-0.028	-0.028	-0.02
13	jd.cmc:job.tenure	-0.018	-0.018	-0.008
14	jd.cmc:oe	0.047	0.047	0.046
15	jc.cmc:genderM	0.062	0.062	0.082
16	jc.cmc:job.tenure	0.000	0.000	0.012
17	sw.cmc:genderM	-0.073	-0.073	-0.059
18	sw.cmc:job.tenure	-0.014	-0.014	-0.015
19	sw.cmc:mw	-0.045	-0.044	-0.04
20	jd.cmc:age	0.013	0.013	.
21	job.tenure:mw	0.031	0.035	.
22	jd.cmc:mw	-0.004	-0.004	.
23	sw.cmc:oe	-0.006	-0.006	.
24	jd.cmc:ji	-0.011	-0.010	.
25	jc.cmc:sw.cmc	-0.012	-0.010	.
26	genderM:job.tenure	-0.113	-0.023	.
27	jc.cmc:age	0.020	0.020	.
28	genderM:oe	-0.037	0.146	.
29	jc.cmc:mw	0.015	0.015	.
30	jd.cmc:jc.cmc	-0.006	-0.003	.
31	jc.cmc:ji	-0.021	-0.020	.
32	age:oe	-0.007	-0.019	.
33	sw.cmc:age	-0.001	-0.001	.
34	genderM:age	0.019	-0.031	.
35	job.tenure:oe	0.044	0.025	.
36	job.tenure:ji	-0.006	-0.062	.

```
37          sw.cmc:ji   -0.073  -0.073         .
38         genderM:ji    0.068   0.282         .
39              oe:ji    0.081   0.153         .
40         genderM:mw   -0.091   0.087         .
41             age:ji    0.055   0.133         .
42          jc.cmc:oe    0.007   0.008         .
43             age:mw   -0.037  -0.040         .
44              mw:oe   -0.038  -0.107         .
45              mw:ji    0.138   0.127         .
46     age:job.tenure   -0.017  -0.022         .
```

P-values coefficienti con interazioni a due

```
ddf$Pval2
```

```
                Names Rand_Int Pen_Int Pen_Int_2
1         (Intercept)    0.562   0.418     0.521
2              jd.cmc    0.749   0.717     0.008
3              jc.cmc    0.041   0.040     0.026
4              sw.cmc    0.044   0.054     0.018
5             genderM    0.938   0.957     0.729
6                 age    0.958   0.637     0.651
7          job.tenure    0.499   0.641     0.687
8                  mw    0.521   0.917     0.973
9                  oe    0.843   0.713     0.731
10                 ji    0.322   0.204     0.843
11      jd.cmc:sw.cmc    0.509   0.587     0.911
12     jd.cmc:genderM    0.604   0.751     0.735
13  jd.cmc:job.tenure    0.169   0.204     0.513
14          jd.cmc:oe    0.066   0.054     0.048
15     jc.cmc:genderM    0.245   0.228      0.14
16  jc.cmc:job.tenure    0.998   0.881     0.372
17     sw.cmc:genderM    0.112   0.082     0.158
18  sw.cmc:job.tenure    0.204   0.166      0.12
19          sw.cmc:mw    0.066   0.082     0.102
20         jd.cmc:age    0.193   0.172         .
21      job.tenure:mw    0.647   0.799         .
22          jd.cmc:mw    0.874   0.949         .
23          sw.cmc:oe    0.755   0.729         .
24          jd.cmc:ji    0.810   0.957         .
```

```
25      jc.cmc:sw.cmc    0.529   0.639         .
26 genderM:job.tenure    0.283   0.619         .
27         jc.cmc:age    0.038   0.042         .
28         genderM:oe    0.852   0.833         .
29          jc.cmc:mw    0.574   0.551         .
30      jd.cmc:jc.cmc    0.772   0.879         .
31          jc.cmc:ji    0.647   0.835         .
32             age:oe    0.855   0.605         .
33         sw.cmc:age    0.879   0.979         .
34        genderM:age    0.812   0.879         .
35      job.tenure:oe    0.390   0.569         .
36      job.tenure:ji    0.951   0.597         .
37          sw.cmc:ji    0.062   0.130         .
38         genderM:ji    0.856   0.785         .
39              oe:ji    0.603   0.551         .
40         genderM:mw    0.704   0.935         .
41             age:ji    0.475   0.246         .
42          jc.cmc:oe    0.775   0.683         .
43             age:mw    0.383   0.831         .
44              mw:oe    0.678   0.641         .
45              mw:ji    0.427   0.599         .
46     age:job.tenure    0.188   0.569         .
```