Homework 4

Context

This assignment reinforces ideas in Module 4: Constrained Optimization. We focus specifically on implementing quantile regression and LASSO.

Due date and submission

Please submit (via Canvas) a PDF containing a link to the web address of the GitHub repo containing your work for this assignment; git commits after the due date will cause the assignment to be considered late. Due date is Wednesday, 4/2 at 10:00AM.

Points

Problem	Points
Problem 0	20
Problem 1	20
Problem 2	30
Problem 3	30

Dataset

The dataset for this homework assignment is in the file `cannabis.rds`. It comes from a study conducted by researchers at the University of Colorado who are working to develop roadside tests for detecting driving impairment due to cannabis use. In this study, researchers measured levels of THC—the main psychoactive ingredient in cannabis—in participants' blood and then collected other biomarkers and had them complete a series of neurocognitive tests. The goal of the study is to understand the relationship between performance on these neurocognitive tests and the concentration of THC metabolites in the blood.

The dataset contains the following variables:

- `id`: subject id
- `t_mmr1`: Metabolite molar ratio—a measure of THC metabolites in the blood. This is the outcome variable.
- `p_*`: variables with the `p_` prefix contain measurements related to pupil response to light.
- `i_*`: variables with the `i_` prefix were collected using an iPad and are derived from neurocognitive tests assessing reaction time, judgment, and short-term memory.
- `h_*`: Variables related to heart rate and blood pressure.

Problem 0

This "problem" focuses on structure of your submission, especially the use git and GitHub for reproducibility, R Projects to organize your work, R Markdown to write reproducible reports, relative paths to load data from local files, and reasonable naming structures for your files.

To that end:

- Create a public GitHub repo + local R Project; I suggest naming this repo / directory bios731_hw4_YourLastName (e.g. bios731_hw4_wrobel for Julia)

- Submit your whole project folder to GitHub
- Submit a PDF knitted from Rmd to Canvas. Your solutions to the problems here should be implemented in your .Rmd file, and your git commit history should reflect the process you used to solve these Problems.

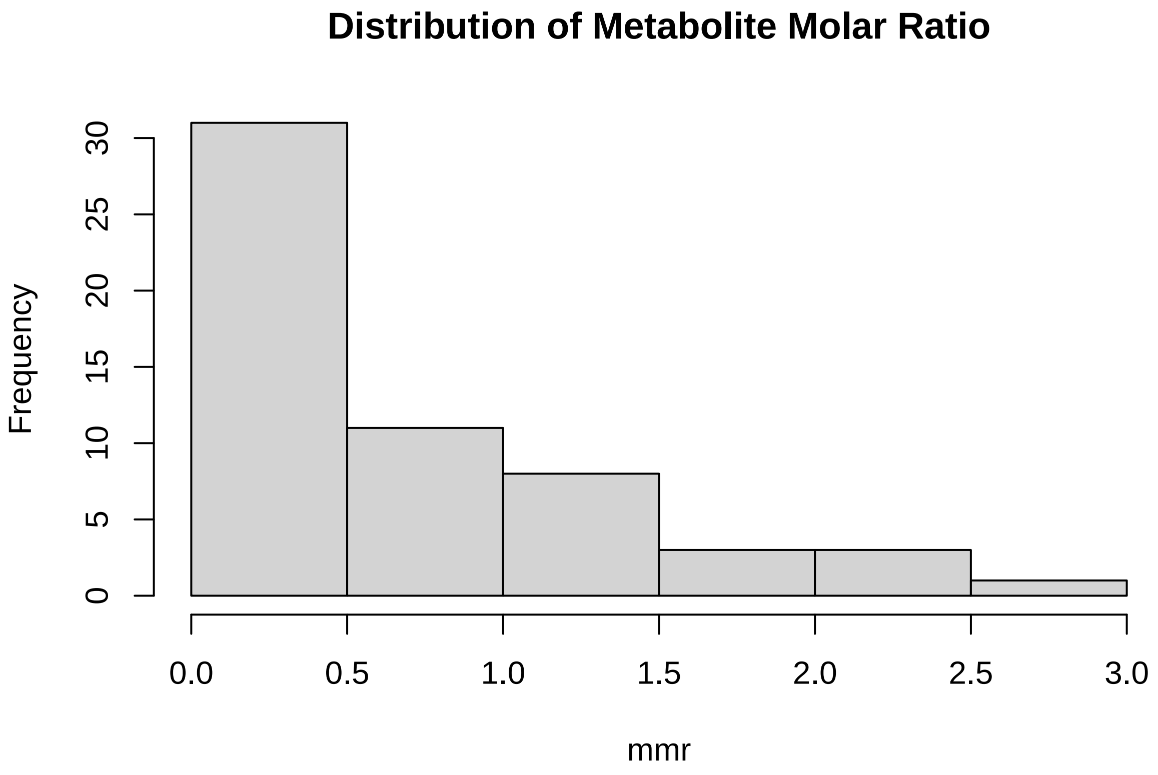
Github repo: https://github.com/ytliu36/bios731_hw4_liu.git

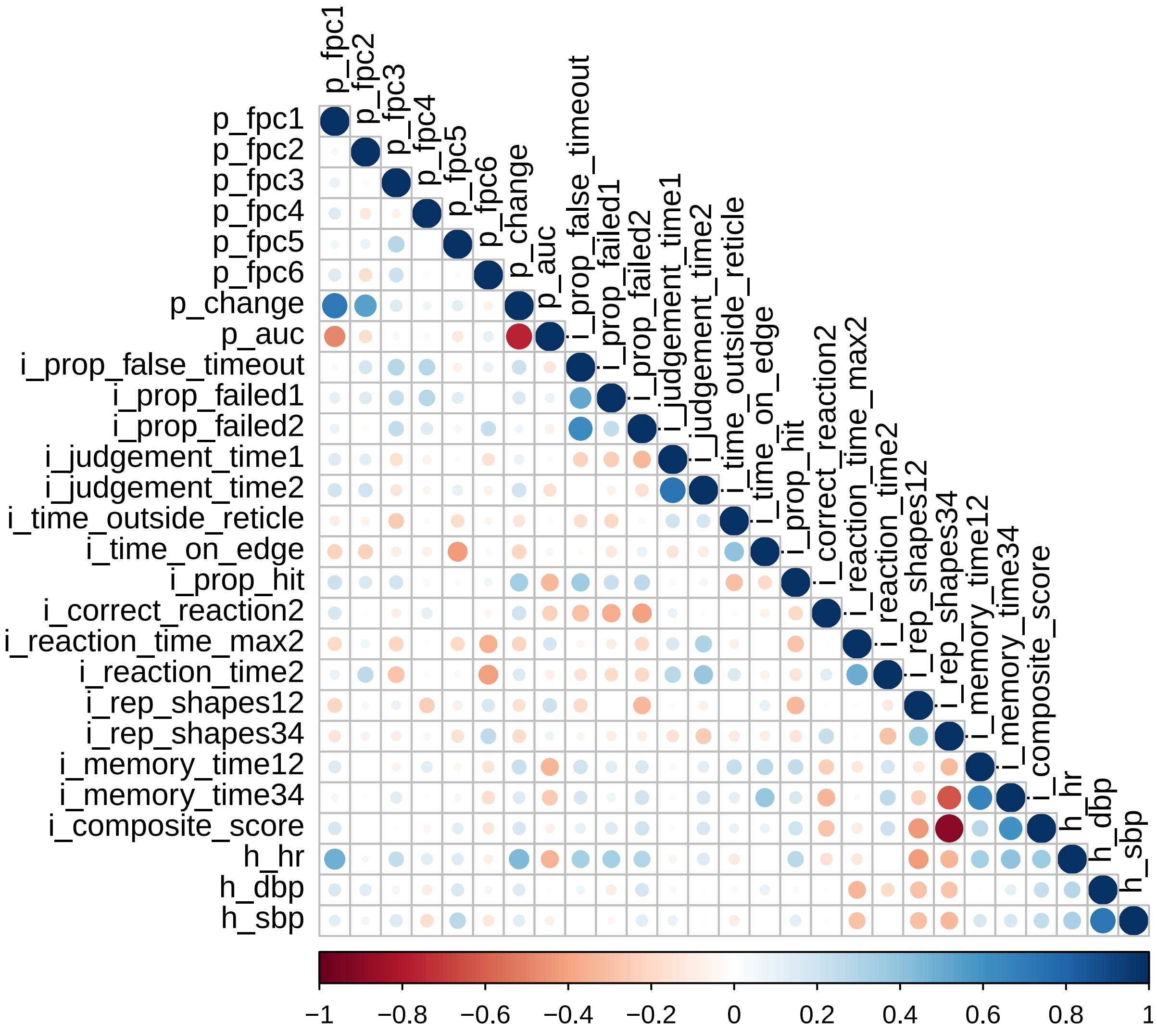
Problem 1: Exploratory data analysis

Perform some EDA for this data. Your EDA should explore the following questions:

- What are n and p for this data?
- What is the distribution of the outcome?
- How correlated are variables in the dataset?

Summarize key findings from your EDA in one paragraph and 2-3 figures or tables.





Summary: There are 57 subjects and 27 predictors in the dataset. The outcome, metabolite molar ratio is right-skewed. Based on the corrlation matrix, there's not many highly correlated predictors, only the correlation between 8 pairs, including: p_fpc1 and p_change (r = 0.72), p_change and p_auc (r = -0.76), i_prop_false_timeout and i_prop_failed2 (r = 0.64), i_judgement_time1 and i_judgement_time2 (r = 0.74), i_rep_shapes34 and i_memory_time34 (r = -0.62), i_rep_shapes34 and i_composite_score (r = -0.90), i_memory_time12 and i_memory_time34 (r = 0.66), h_dbp and h_sbp (r = 0.72).

Problem 2: Quantile regression

Use linear programming to estimate the coefficients for a quantile regression. You need to write a function named `my_rq`, which takes a response vector y, a covariate matrix X and quantile τ , and returns the estimated coefficients. Existing linear programming functions can be used directly to solve the LP problem (for example, `simplex` function in the `boot` package, or `lp` function in the `lpSolve` package).

- Use your function to model `t_mmr1` from the cannabis data using `p_change` (percent change in pupil diameter in response to light), `h_hr` (heart rate), and `i_composite_score` (a composite score of the ipad variables) as variables.
- Compare your results with though estimated using the `rq` function in R at quantiles $\tau \in \{0.25, 0.5, 0.75\}$.
- Compare with mean obtain using linear regression
- Summarize findings

```r
library(lpSolve)

my_rq <- function(y, X, tau) {
  n <- length(y)
  p <- ncol(X)

  # Modify design matrix to allow negative betas
  X_new <- cbind(X, -X)  # Create + and - (splitting each   into positive and negative parts)

  # Construct LP problem
  f.obj <- c(rep(0, 2 * p), rep(tau, n), rep(1 - tau, n))  # Objective function
  f.con <- cbind(X_new, diag(n), -diag(n))  # Constraint matrix
  f.rhs <- y  # Right-hand side
  f.dir <- rep("=", n)  # Constraints are equality constraints

  # Solve LP
  lp_result <- lp("min", f.obj, f.con, f.dir, f.rhs, all.int = FALSE)

  # Extract beta coefficients
  beta_hat <- lp_result$solution[1:p] - lp_result$solution[(p + 1):(2 * p)]  #   = + - -
  return(beta_hat)
}
```

```r
library(quantreg)
```

```
## Loading required package: SparseM
```

```r
# Prepare variables
y <- dat$t_mmr1
X <- cbind(1, dat$p_change, dat$h_hr, dat$i_composite_score)  # Add intercept

# Estimate at quantiles 0.25, 0.5, and 0.75
tau_vals <- c(0.25, 0.5, 0.75)
my_rq_results <- lapply(tau_vals, function(tau) my_rq(y, X, tau))
names(my_rq_results) <- paste0("tau_", tau_vals)

# Compare with rq()
rq_results <- lapply(tau_vals, function(tau) coef(rq(t_mmr1 ~ p_change + h_hr + i_composite_score, tau
names(rq_results) <- paste0("tau_", tau_vals)

# Compare with OLS
ols_result <- coef(lm(t_mmr1 ~ p_change + h_hr + i_composite_score, data = dat))

# Print results
list("My Quantile Regression" = my_rq_results, "RQ Function" = rq_results, "OLS" = ols_result)
```

```
## $`My Quantile Regression`
## $`My Quantile Regression`$tau_0.25
## [1] -0.150050841  0.011418255  0.008176194  0.384063034
##
## $`My Quantile Regression`$tau_0.5
## [1] -0.28342611  0.01219383  0.01357267  0.19819454
##
## $`My Quantile Regression`$tau_0.75
## [1] -1.114045873  0.004178877  0.027280412 -0.580906640
```

```
##
##
## $`RQ Function`
## $`RQ Function`$tau_0.25
##       (Intercept)          p_change             h_hr i_composite_score
##      -0.150050841       0.011418255      0.008176194       0.384063034
##
## $`RQ Function`$tau_0.5
##       (Intercept)          p_change             h_hr i_composite_score
##       -0.28342611        0.01219383       0.01357267        0.19819454
##
## $`RQ Function`$tau_0.75
##       (Intercept)          p_change             h_hr i_composite_score
##      -1.114045873       0.004178877      0.027280412      -0.580906640
##
##
## $OLS
##       (Intercept)          p_change             h_hr i_composite_score
##       -0.17879290        0.00766679       0.01435096       -0.23822747
```

When explaining your results, be sure to explain what LP method you used for estimating quantile regression.

Answer: As for the quantile regression, it minimize $\sum_{i=1}^{n} \rho_\tau(y_i - x_i\beta)$, which is equivalent to minimize $\sum_{i=1}^{n}(\tau u_i + (1-\tau)v_i)$ when define $u_i = [y_i - x_i\beta]_+ \geq 0$ and $v_i = [y_i - x_i\beta]_- \geq 0$ and $y_i = x_i\beta + u_i - v_i$. The above linear programming is solved by `lp()` in `lpSolve`.

`my_rq()` and `rq()` give the same result at each quantile given, and the association between predictors and outcomes vary at each quantile, p_change and t_mmr1 is positively correlated, but with highest correlation at $\tau = 0.5$ compared to the other 2. h_hr is also positively correlated with t_mmr1 and correlation become stronger with higher τ. At smaller τ, i_composite_score has positive correlation at $\tau = 0.25$ and 0.5, but become negative when τ=0.75. OLS is able to detect the positive correlation of p_change and h_hr, but failed to identify the positive correlation of i_composite_score at lower quantiles.

Problem 3: Implementation of LASSO

As illustrated in class, a LASSO problem can be rewritten as a quadratic programming problem.

1. Many widely used QP solvers require that the matrix in the quadratic function for the second order term to be positive definite (such as `solve.QP` in the `quadprog` package). Rewrite the quadratic programming problem for LASSO in matrix form and show that the matrix is not positive definite, thus QP solvers like `solve.QP` cannot be used.
2. The `LowRankQP` function in the `LowRankQP` package can handle the non positive definite situation. Use the matrix format you derived above and `LowRankQP` to write your own function `my_lasso()` to estimate the coefficients for a LASSO problem. Your function needs to take three parameters: Y (response), X (predictor), and *lambda* (tuning parameter), and return the estimated coefficients.

- Use your function to model `log(t_mmr1)` from the cannabis data using all other variables as potential covariates in the model
- Compare your results with those estimated using the `cv.glmnet` function in R from the `glmnet` package
- Summarize findings

```
library(LowRankQP)
```

```
## LowRankQP 1.0 loaded
## Copyright J.T. Ormerod & M. P. Wand 2023
```

```r
my_lasso <- function(Y, X, lambda) {
  n <- nrow(X)
  p <- ncol(X)

  mat<-cbind(diag(p), -diag(p))
  # Construct quadratic term
  Dmat <- t(mat)%*%crossprod(X)%*%mat # X'X (2p x 2p)-alpha estimated is c(u,v)
  dvec <- -t(mat)%*%crossprod(X, Y) # -X'Y (2p x 1)

  # Constraints:sum u+v = lambda
  Amat <- matrix(rep(1, 2 * p), ncol = 2*p)  # (2p x 1)
  bvec <-c(lambda)  # (2p x 1)

  # Solve QP problem
  sol <- LowRankQP(Dmat, dvec, Amat, bvec, u = rep(100, 2*p))

  # Extract beta coefficients
  beta_hat <- sol$alpha[1:p]-sol$alpha[(p+1):(2*p)]
  return(beta_hat)
}
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:SparseM':
##
##     det

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

```r
# Prepare X and Y
X <- as.matrix(dat[, -c(1, 2)])
X <- scale(X)
Y <- log(dat$t_mmr1+1e-4)  # Log-transform outcome

# Run my LASSO function
lambda_val <- 0.7
beta_lasso <- my_lasso(Y, X, lambda_val)
beta_lasso
```

```
##  [1]  8.046376e-14 -1.344271e-02  5.791218e-14 -1.346945e-13 -5.051048e-14
##  [6]  1.627779e-14 -7.066545e-13  3.860789e-09 -2.700533e-14  6.945947e-14
## [11] -2.027924e-01 -9.195394e-15 -5.851263e-14 -7.409229e-02  3.264022e-14
## [16]  2.326846e-14  3.480940e-14  1.496787e-13 -1.199432e-02  1.050277e-14
## [21]  2.015300e-13 -9.572836e-14 -5.082425e-14 -1.133612e-13  3.866133e-01
## [26] -1.106506e-02 -3.665752e-13
```

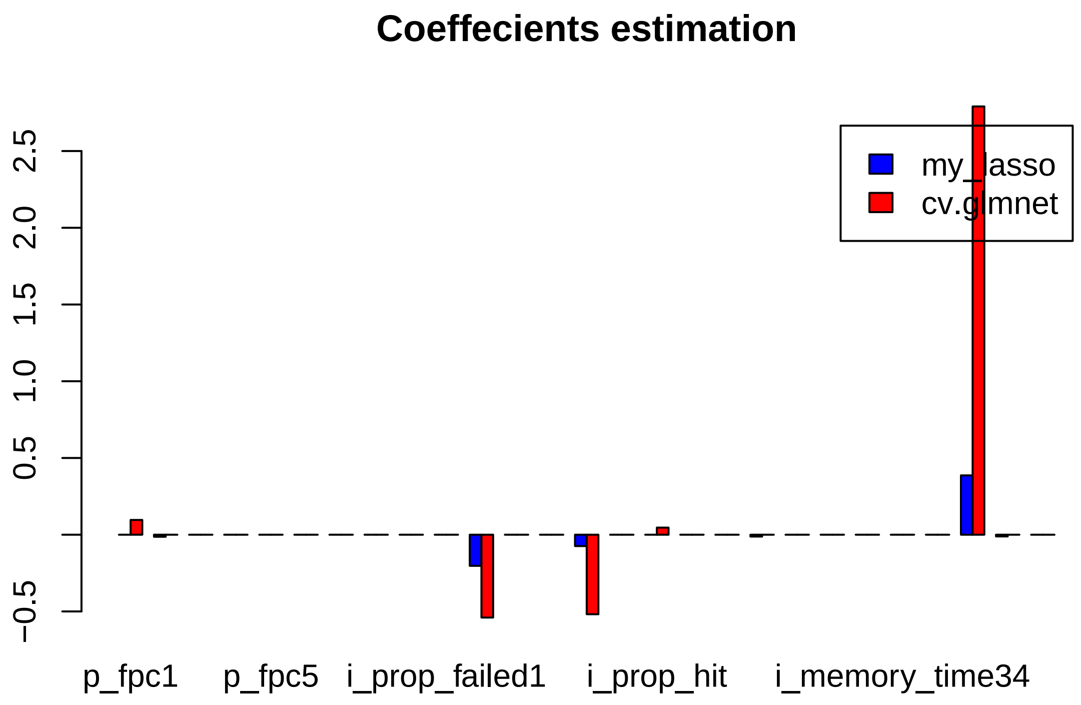
```r
# Compare with glmnet
cv_fit <- cv.glmnet(X, Y)
```

```r
beta_glmnet <- coef(cv_fit, s = "lambda.min")[-1]
beta_glmnet
```

```
##  [1]  0.09617113  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
##  [7]  0.00000000  0.00000000  0.00000000  0.00000000 -0.53978110  0.00000000
## [13]  0.00000000 -0.51809916  0.00000000  0.04623226  0.00000000  0.00000000
## [19]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## [25]  2.79066318  0.00000000  0.00000000
```

```r
beta_matrix <- rbind(beta_lasso, beta_glmnet)

barplot(beta_matrix, beside = TRUE, names.arg = colnames(X),
        col = c("blue", "red"), legend.text = c("my_lasso", "cv.glmnet"),
        main = "Coeffecients estimation")
```



The results will not be exactly the same because the estimation procedures are different, but trends (which variables are selected) should be similar.

Summary: From the plot above we can see at lambda = 0.7, although the absolute value of my_lasso estimation is relatively small, the predictors selected and the sign of estimated coefficients are mostly the same.