Homework 4

GitHub Link: https://github.com/rw417/bios731_hw4_wan

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.

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.

ANSWER:
The data has 29 columns and 57 rows. The first two columns are participant ID and the response variable. So there are 27 predictor variables and 57 observations. In Figure 1 below, we see that the response variable, `t_mmr1` is zero for 29 of the 57 observations. After applying log-tranformation, the non-zero values of `t_mmr1` are approximately normally distributed. In Figure 2, we see that most variables are not highly correlated. The top 10 most correlated variables are shown in Table 1 below.

Figure 1: Distribution of Metabolite Molar Ratio

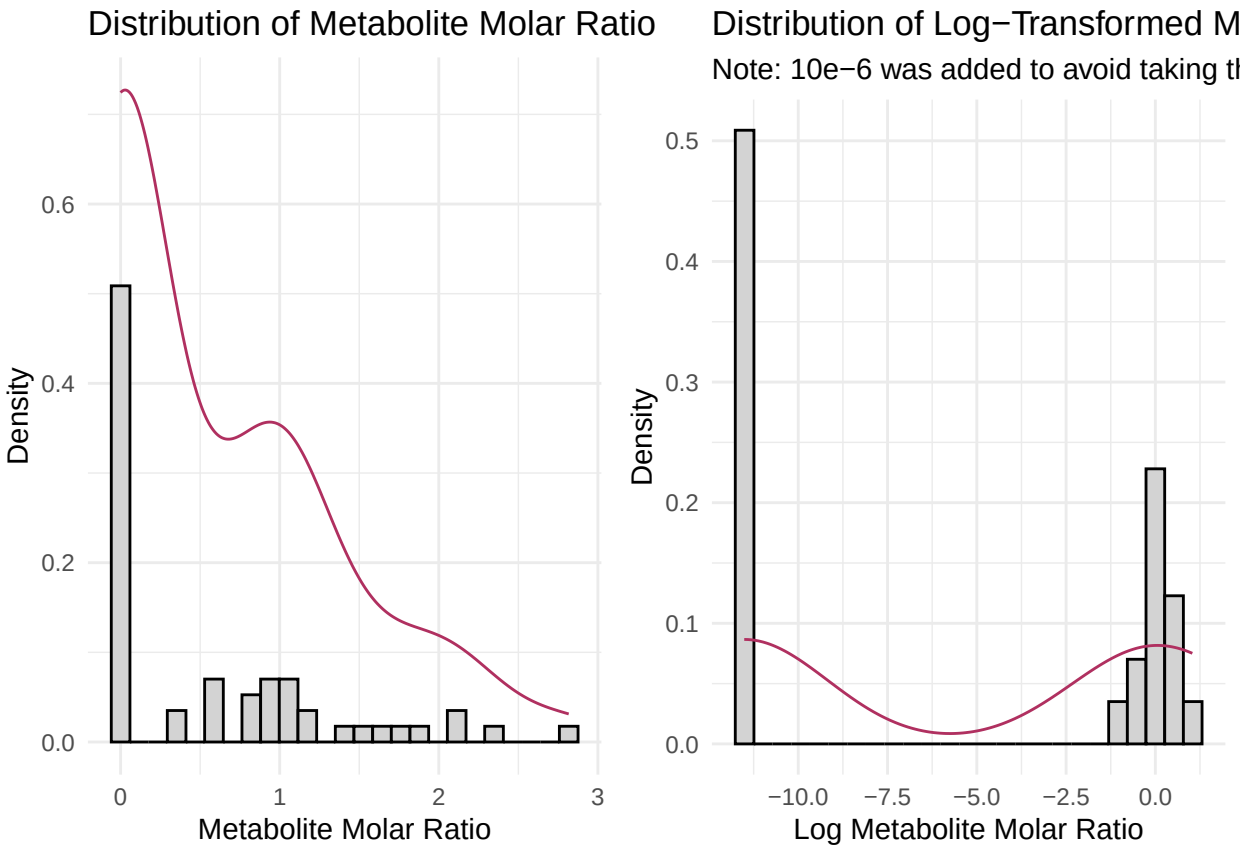

Figure 2: Correlation Plot of Variables

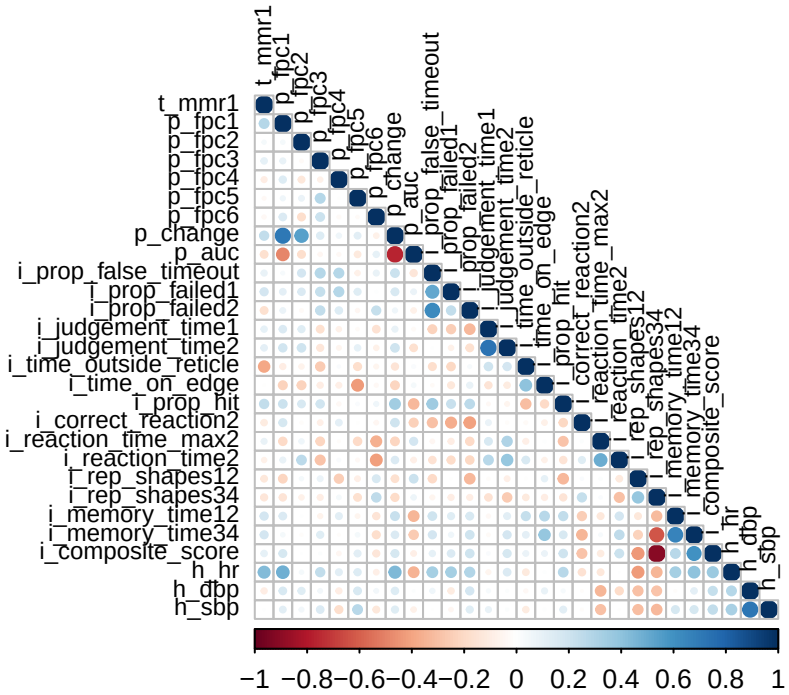

Table 1: Top 10 Correlated Variable Pairs

Variable 1	Variable 2	Correlation
i_rep_shapes34	i_composite_score	-0.901
p_change	p_auc	-0.761
i_judgement_time1	i_judgement_time2	0.740
h_dbp	h_sbp	0.722
p_fpc1	p_change	0.717
i_memory_time12	i_memory_time34	0.663
i_prop_false_timeout	i_prop_failed2	0.640
i_rep_shapes34	i_memory_time34	-0.621
i_memory_time34	i_composite_score	0.603
p_fpc2	p_change	0.549

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

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

ANSWER:
My `my_rq()` function uses the simplex method to solve the linear programming problem for quantile regression. The coefficients it computed are exactly the same as those from the `rq()` function in the `quantreg` package. Table 2 compares model coefficients of quantile regressions on the 0.25, 0.5, and 0.75 quantiles of $t_m mr1$ against linear regression, which is on the mean of $t_m mr1$. The coefficients of the quantile regressions are very different because $t_m mr1$ is heavily zero-inflated. Quantile regression is more robust against skewness in the data so it is the preferred method for this data set.

Table 2: Comparison of Coefficients from Quantile Regression and Linear Regression

Method	Tau	Intercept	p_change	h_hr	i_composite_score
my_rq	0.25	-0.150	0.011	0.008	0.384
my_rq	0.5	-0.283	0.012	0.014	0.198
my_rq	0.75	-1.114	0.004	0.027	-0.581
rq	0.25	-0.150	0.011	0.008	0.384
rq	0.5	-0.283	0.012	0.014	0.198
rq	0.75	-1.114	0.004	0.027	-0.581
lm	NA	-0.179	0.008	0.014	-0.238

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.

ANSWER:

$$
\begin{aligned}
&-\sum_{i}^{n}\left(y_i-\sum_j \beta_j^+ x_{ij}+\sum_j \beta_j^- x_{ij}\right)^2 \\
&=-(Y-X\beta^+ + X\beta^-)^T(Y-X\beta^+ + X\beta^-) \\
&=-(Y^TY - Y^TX\beta^+ + Y^TX\beta^- - \beta^{+T}X^TY + \beta^{+T}X^TX\beta^+ - \beta^{+T}X^TX\beta^- + \beta^{-T}X^TY - \beta^{-T}X^TX\beta^+ + \beta^{-T}X^TX\beta^-) \\
&=-(Y^TY - 2\beta^{+T}X^TY + 2\beta^{-T}X^TY + \beta^{+T}X^TX\beta^+ - 2\beta^{+T}X^TX\beta^- + \beta^{-T}X^TX\beta^-) \\
&=-(Y^TY - 2Y^TX(\beta^+ - \beta^-) + (\beta^+ - \beta^-)^TX^TX(\beta^+ - \beta^-))
\end{aligned}
$$

To see why X^TX is positive semidefinite, let X not have full rank. Then there exists some non-zero vector v such that $Xv = 0$. Thus $v^TX^TXv = (Xv)^TXv = 0$. Thus X^TX is positive semidefinite.

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

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

ANSWER:
The `glmnet` and `cv.glmnet` functions use penalties instead of L1 constraints to shrink the coefficients. To make my LASSO function comparable with `glmnet`, I first ran `cv.glmnet` over a default set of penalty values. I then calculated the corresponding L1 constraints for those penalty values. I then used the calculated constraints to run my LASSO function. Table 3 below reports the models with the smallest cross-validated mean squared error from `glmnet` and my LASSO function.

Both `glmnet` and my LASSO picked similar variables, with `glmnet` picking one more variable, `i_prop_failed2`. The model in `glmnet` with the smallest CV error has an L1 constraint of 5.586. On the other hand, the model in my LASSO with the smallest CV error has an L1 constraint of 0.181. Nevertheless, the CV MSEs are similar across both models.

Table 3: Comparison of of `glmnet` and `my_lasso`

	glmnet	my_lasso
Intercept	-18.141	-19.253
p_fpc1	0.001	0.015
p_fpc2	0.000	0.000
p_fpc3	0.000	0.000
p_fpc4	0.000	0.000
p_fpc5	0.000	0.000
p_fpc6	0.000	0.000
p_change	0.000	0.000
p_auc	0.000	0.000
i_prop_false_timeout	0.000	0.000
i_prop_failed1	0.000	0.000
i_prop_failed2	-4.457	0.000
i_judgement_time1	0.000	0.000
i_judgement_time2	0.000	0.000
i_time_outside_reticle	-0.041	-0.008
i_time_on_edge	0.000	0.000
i_prop_hit	0.000	0.000
i_correct_reaction2	0.000	0.000
i_reaction_time_max2	0.000	0.000
i_reaction_time2	0.000	0.000
i_rep_shapes12	0.000	0.000
i_rep_shapes34	0.000	0.000
i_memory_time12	0.000	0.000
i_memory_time34	0.000	0.000
i_composite_score	0.000	0.000
h_hr	0.185	0.158
h_dbp	0.000	0.000
h_sbp	0.000	0.000
L1 Constraint	4.684	0.181
MSE	27.096	27.148