Lab 07: Rent Due, Lights Due, Mountain Dew – SOLUTIONS

PSTAT 100, Summer Session A 2025 with Ethan P. Marzban

MEMBER 1 (NetID 1) MEMBER 2 (NetID 2)
MEMBER 3 (NetID 3)

July 17, 2025

Required Packages

```
library(ottr)         # for checking test cases (i.e. autograding)
library(pander)       # for nicer-looking formatting of dataframe outputs
library(tidyverse)    # for graphs, data wrangling, etc.
```

Logistical Details

Logistical Details

- This lab is due by **11:59pm on Friday, July 18, 2025**.
- Collaboration is allowed, and encouraged!
  - If you work in groups, list ALL of your group members' names and NetIDs (not Perm Numbers) in the appropriate spaces in the YAML header above.
  - Please delete any "MEMBER X" lines in the YAML header that are not needed.
  - No more than 3 people in a group, please.
- Ensure your Lab properly renders to a `.pdf`; non-`.pdf` submissions will not be graded and will receive a score of 0.
- Ensure all test cases pass (test cases that have passed will display a message stating `"All tests passed!"`)

Lab Overview and Objectives

Welcome to another PSTAT 100 Lab! In this lab, we will cover the following:

- Linear Regression (both simple and multiple)
- Multicollinearity

Part I: A Mini-Project

In this portion of the lab, we'll undergo a sort of mini-project to explore the implementation of **linear regression** in `R`.

Recap: Statistical Models

Given data $\mathcal{D} := \{\vec{\boldsymbol{x}}_i, y_i\}_{i=1}^n$ consisting of observations y_i of a **response variable** `y` and observations $\vec{\boldsymbol{x}} := (x_{i1}, \cdots, x_{ip})$ on p **explanatory** or **predictor variables** $\texttt{x}_1$ through $\texttt{x}_\text{p}$, a **statistical model** assumes the relationship between y_i and $\vec{\boldsymbol{x}}_i$ to be

$$y_i = f(\vec{\boldsymbol{x}}) + \varepsilon_i$$

for some **noise** term ε_i.

A **linear regression** model assumes:

1) A linear signal function; i.e. $f(\vec{\boldsymbol{x}}_i) = \beta_0 + \sum_{j=1}^p x_{ij}$
2) Numerical response values (i.e. `y` is assumed to be a numerical variable as opposed to a categorical one).

The matrix representation of a linear regression model is:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{:=\vec{\boldsymbol{y}}} = \underbrace{\begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}}_{:=\mathbf{X}} \underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}}_{:=\vec{\boldsymbol{\beta}}} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}}_{:=\vec{\boldsymbol{\varepsilon}}}$$

It is typical to assume i.i.d. Gaussian errors:

$$\varepsilon_i \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \sigma^2) \iff \vec{\boldsymbol{\varepsilon}} \sim \mathcal{N}_n\left(\vec{\mathbf{0}}\ ,\ \sigma^2\mathbf{I}\right)$$

The **ordinary least squares** (OLS) fit to the data seeks to find estimators $\widehat{\boldsymbol{\beta}}$ that solve the following minimization problem:

$$\widehat{\boldsymbol{\beta}} = \arg\ \min_{\vec{\boldsymbol{b}}} \left\{ \left\| \vec{\boldsymbol{y}} - \mathbf{X}\vec{\boldsymbol{b}} \right\|^2 \right\}$$

which, under certain conditions, admits the following solution:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \vec{\boldsymbol{y}}$$

In `R`, the function `lm()` performs model fitting of an MLR model to the data assuming L_2 loss.

Introduction to the Dataset

In this part of the lab, we consider a simulated dataset containing information on 302 different rental properties. For each rental property, we have observations on:

- **`ppm`**: the price per month (i.e. rental costs), in thousands of dollars
- **`num_bedrooms`**: the number of bedrooms (a value of `0` indicates a studio apartment)
- **`dist_to_highway`**: the distance (in miles) to the nearest highway
- **`dist_to_town_center`**: the distance (in miles) to the nearest town center

The data is stored in the `rental.csv` file, located in the `data/` subfolder.

Our ultimate goal is to model the rent (`ppm`) as a function of the other three variables included in the dataset.

Tip

If `df` is the name of an `R` dataframe and `y` is the name of one of the columns in `df`, running

```
lm(y ~ ., data = df)
```

is a shorthand to regress `y` onto *all* other columns present in `df`.

Question 1

Regress `ppm` onto all of the other three variables present in the dataset (`num_bedrooms`, `dist_to_highway`, and `dist_to_town_center`). Assign this model to an `R` object called `full_mod`.

Solution:

```
## replace this line with your code
rental <- read.csv("data/rental.csv")
full_mod <- lm(ppm ~ . , data = rental)
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q1.R")})
```

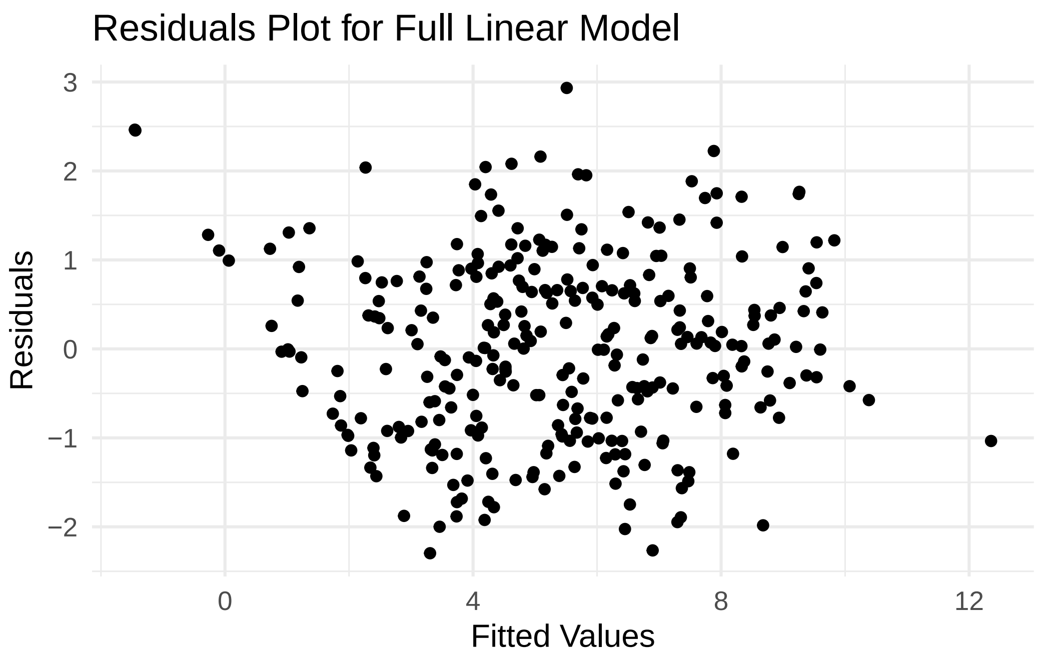
```
All tests passed!
```

❗ Question 2

Produce a residuals plot of the fit from Question 1, and comment on the fit. You may want to look up the help file for `lm()` to see how to extract the residuals and fitted values.

Solution:

```r
## replace this line with your code
data.frame(
  x = full_mod$fitted.values,
  y = full_mod$residuals
) %>% ggplot(aes(x = x, y = y)) +
  geom_point() +
  xlab("Fitted Values") + ylab("Residuals") +
  ggtitle("Residuals Plot for Full Linear Model") +
  theme_minimal(base_size = 12)
```



There are no obvious signs of trend or heteroskedasticity; hence, this is indicative of a model that is fitting relatively well.

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

❗ Question 3

Part (a)
Produce a regression table from the `full_mod` fit, and interpret the estimated coefficient values in the context of the problem. For example, what average change in rent is expected from a

one-unit change in the number of bedrooms, holding all else constant? What about a one-unit change in the distance to the nearest town center? etc.

Solution:

```r
## replace this line with your code
full_mod %>% summary()
```

```

Call:
lm(formula = ppm ~ ., data = rental)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.29678 -0.78228  0.01765  0.71668  2.93300 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)        0.25100    0.28441   0.883    0.378    
num_bedrooms       1.48200    0.05039  29.410   <2e-16 ***
dist_to_highway    1.46437    0.06192  23.649   <2e-16 ***
dist_to_town_cent -0.98341    0.05708 -17.228   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.023 on 298 degrees of freedom
Multiple R-squared:  0.8379,    Adjusted R-squared:  0.8363 
F-statistic: 513.6 on 3 and 298 DF,  p-value: < 2.2e-16
```

A one-unit increase in the number of bedrooms corresponds to a predicted 1.48 thousand dollar increase in monthly rent. A one-unit increase in the distance to the nearest highway corresponds to a predicted 1.46 thousand dollar increase in monthly rent. A one-unit increase in the distance to the nearest town center to a predicted 0.98 thousand dollar decrease in monthly rent.

Perhaps this makes intuitive sense; houses closer to highways are likely less expensive than those closer to the highway, and houses closer to the town center are likely more expensive than those farther away.

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Part (b)
Based on the regression table from part (a), is there statistical evidence to suggest that any of the coefficients might be nonzero? Use a 5% level of significance, and justify your answer.

Solution:
Replace this line with your answer.
The p-values are given by the final column in the regression table. We see that all but the intercept coefficients have p-values that are practically zero; hence, at a 5% level of significance, the only null hypothesis that we would fail to reject is that the intercept is zero.

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

❗ Question 4

Use the `model.matrix()` function to extract the model matrix **X** from the `full_mod` relationship. Then, using only basic linear algebra operations in `R` (e.g. `%*%` for matrix multiplication, `t()` for transpose, and `solve()` for matrix inversion), recompute the OLS estimates for the coefficients of the full model. Check that these agree with the regression table from Question 3. Store your resulting estimates in a vector called `beta_est`.

Solution:

```
## replace this line with your code
X <- model.matrix(full_mod)
beta_est <- solve(t(X) %*% X) %*% t(X) %*% rental$ppm
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q4.R")})
```

```
All tests passed!
```

Caution

When specifying the `newdata` argument in the `predict()` function as applied to a model fitted by `lm()`, the new data must:

1) be a data frame
2) have the same variable names as the original data frame

❗ Question 5

Part (a)
A new house has gone on the market. It has 3 bedrooms, lies 1.24 miles from the nearest highway, and lies 0.45 miles from the nearest town center. Use the `full_mod` model to predict the rental price of this apartment, in thousands of dollars per month. Assign this value to a

variable called `pred_ppm1`.

Solution:

```
## replace this line with your code
pred_ppm1 <- predict(
  full_mod,
  newdata = data.frame(num_bedrooms = 3,
                       dist_to_highway = 1.25,
                       dist_to_town_cent = 0.45)
)
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q5a.R")})
```

```
Test q5a failed:

round(pred_ppm1, 3) == 6.07 is not TRUE

  `actual`:   FALSE
  `expected`: TRUE
```

Part (a)
Another new house has gone on the market. It has 3 bedrooms, lies 10.1 miles from the nearest highway, and lies 0.45 miles from the nearest town center. Explain why it is **not** a good idea to use the `full_mod` model to predict the rental price of this apartment. There is a specific word/term I'm looking for here; also, you should justify your answer using code output. **Hint:** `summary()`.

Solution:

```
## replace this line with your code
summary(rental)
```

```
      ppm          num_bedrooms   dist_to_highway dist_to_town_cent
 Min.   : 0.774   Min.   :0.00   Min.   :0.200   Min.   :0.1193
 1st Qu.: 3.398   1st Qu.:2.00   1st Qu.:2.473   1st Qu.:2.3371
 Median : 5.307   Median :2.00   Median :2.965   Median :3.0507
 Mean   : 5.359   Mean   :2.48   Mean   :3.000   Mean   :3.0111
 3rd Qu.: 7.204   3rd Qu.:3.00   3rd Qu.:3.627   3rd Qu.:3.6649
 Max.   :11.318   Max.   :5.00   Max.   :5.730   Max.   :5.8873
```

The range of distance-from-highway values is 0.2 to around 5.7. A distance-from-highway value of 10.1 falls very far outside this range - hence, trying to use our

fitted model to predict the rental price of this new house would be in danger of performing extrapolation.

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Part II: Multicollinearity

In this part of the lab, we explore the effects of multicollinearity through simulation.

First, we generate two independent draws of size 1000 from a standard normal distribution:

```r
n <- 1000
set.seed(100)
x1 <- rnorm(n); x2 <- rnorm(n)
```

Next, we construct a new variable that is simply the sum of `x1` and `x2`:

```r
x3 <- x1 + x2
```

What this means is that `x1`, `x2`, and `x3` are highly collinear! Nevertheless, let's construct a response value `y` as a linear combination of `x1`, `x2`, and `x3`, perturbed by zero-mean and homoskedastic normal noise:

```r
y <- x1 + x2 + x3 + rnorm(n)
```

Now, we construct a data frame using `y`, `x1`, `x2`, and `x3`:

```r
df1 <- data.frame(y, x1, x2, x3)
```

Here is the correlation matrix of `df1`:

```r
cor(df1)
```

```
           y         x1         x2        x3
y  1.0000000 0.69027267 0.64147554 0.9356903
x1 0.6902727 1.00000000 0.01409256 0.7289175
x2 0.6414755 0.01409256 1.00000000 0.6948059
x3 0.9356903 0.72891747 0.69480590 1.0000000
```

As expected, it reveals that `x1` and `x2` are not highly correlated between themselves, but are both highly correlated with `x3`.

`lm()` still produces an optimal linear fit:

```r
(lm1 <- lm(y ~ . , data = df1))
```

```

Call:
lm(formula = y ~ ., data = df1)

Coefficients:
(Intercept)           x1           x2           x3
    -0.0139       1.9618       1.9112           NA
```

but now `R` is unable to produce an estimate for the parameter associated with `x3`.

! Question 6

Recall that the variance-covariance matrix of the OLS estimators is given by $\sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$ Because our data was simulated, we know to take $\sigma^2 = 1$. Compute the variance-covariance matrix of the OLS estimators in this simulation, and assign this to a matrix called `Sigma1`.

Solution:

```r
## replace this line with your code
X <- model.matrix(lm1)
(Sigma1 <- solve(t(X) %*% X))
```

```
              (Intercept)            x1            x2            x3
(Intercept)  1.000282e-03 -9.630134e-06  1.947416e-06 -5.755388e-06
x1          -1.027670e-05 -2.761503e+11 -2.761503e+11  2.761503e+11
x2           1.427270e-06 -2.761503e+11 -2.761503e+11  2.761503e+11
x3          -5.506753e-06  2.761503e+11  2.761503e+11 -2.761503e+11
```

Answer Check:

```r
# DO NOT EDIT THIS LINE
invisible({check("tests/q6.R")})
```

```
All tests passed!
```

What you should see is that the variances in this matrix **make no sense**! This is one of the consequences of multicollinearity - the OLS estimators become highly *unstable*.

Also, returning to the output of `lm()` above: note that the estimated coefficients for `x1` and `x2` are nearly twice what they should be (recall that, when we simulated the data, we set the parameters associated with these two variables to be 1).

This also makes sense, if we think about the model `lm()` is trying to fit.

$$\mathtt{y} = \beta_0 + \beta_1 \mathtt{x}_1 + \beta_2 \mathtt{x}_2 + \beta_3 \mathtt{x}_3 + \mathtt{noise}$$

Since we know $\mathtt{x}_3 = \mathtt{x}_1 + \mathtt{x}_2$ in this case, our model is actually:

$$\mathtt{y} = \beta_0 + \beta_1 \mathtt{x}_1 + \beta_2 \mathtt{x}_2 + \beta_3 (\mathtt{x}_1 + \mathtt{x}_2) + \mathtt{noise}$$

which can be rearranged as

$$\mathtt{y} = \beta_0 + (\beta_1 + \beta_3) \mathtt{x}_1 + (\beta_2 + \beta_3) \mathtt{x}_2 + \mathtt{noise}$$

Indeed, `lm()` has estimated $(\beta_1 + \beta_3)$ and $(\beta_2 + \beta_3)$ quite well! The problem is, it is now near impossible to estimate β_1 and β_2 *separately*, as we can only obtain estimates for $(\beta_1 + \beta_3)$ and $(\beta_2 + \beta_3)$. So this is another consequence of multicollinearity - even the parameters we *are* able to estimate have their effects intertwined with the variable(s) that are multicollinear.

In practice, one of the easiest things to do when we detect multicollinearity is to remove the offending variable.

! Question 7

Generate a new fit, called `lm2`, that regresses `y` onto only `x1` and `x2`. Then, display the regression table and compare this to the regression table for `lm1`. Are they at all similar?

Solution:

```r
## replace this line with your code
lm2 <- lm(y ~ x1 + x2)
lm2 %>% summary()
```

```

Call:
lm(formula = y ~ x1 + x2)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.1553 -0.6838  0.0151  0.6733  3.3025 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.01390    0.03312   -0.42    0.675    
x1           1.96179    0.03216   61.01   <2e-16 ***
x2           1.91122    0.03378   56.58   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.047 on 997 degrees of freedom
Multiple R-squared:  0.8757,	Adjusted R-squared:  0.8754 
F-statistic:  3511 on 2 and 997 DF,  p-value: < 2.2e-16
```

```r
lm1 %>% summary()
```

```

Call:
lm(formula = y ~ ., data = df1)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.1553 -0.6838  0.0151  0.6733  3.3025 

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.01390    0.03312   -0.42    0.675    
x1           1.96179    0.03216   61.01   <2e-16 ***
x2           1.91122    0.03378   56.58   <2e-16 ***
x3                NA         NA      NA       NA    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.047 on 997 degrees of freedom
Multiple R-squared:  0.8757,	Adjusted R-squared:  0.8754 
F-statistic:  3511 on 2 and 997 DF,  p-value: < 2.2e-16
```

The two regression tables are the same, aside from the missing portion of the table for lm1. In other words, it seems that, behind the scenes, the way R obtained the non-missing portion of the regression table for lm1 was to simply remove the offending variable.

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Submission Details

Congrats on finishing this PSTAT 100 lab! Please carry out the following steps:

Submission Details

1) Check that all of your tables, plots, and code outputs are rendering correctly in your final `.pdf`.

2) Check that you passed all of the test cases (on questions that have autograders). You'll know that you passed all tests for a particular problem when you get the message "All tests passed!".

3) Submit **ONLY** your `.pdf` to Gradescope. Make sure to **match ALL pages to the ONE question on Gradescope**; failure to do so will incur a penalty of 0.1 points.