# Context

## Introduction

Regression models describe the relationship between variables by fitting a line to the observed data. A straight line is used in linear regression models, whereas a curved line is used in logistic and nonlinear regression models. You may use regression to predict how a dependent variable will change when the independent variable(s) change. However, the technique must be used to understand the assumptions in each type of regression analysis, their limitations, and the potential error that can occur when models are applied to a larger population. This project will provide an overview of regression analysis techniques: how they work, what they assume, and how they could go wrong if essential assumptions aren't true. Also, a real-life example with R-code will be provided to make the discussion concrete.

## Simple Linear Regression

Simple linear regression is the most commonly used analysis approach for analyzing the connection between a quantitative result (respond) and a single explanatory variable. The "simple" component indicates that we are only looking at one explanatory variable. When you need to know the following, you may use simple linear regression:

- The strength of the relationship between two variables
- The value of the dependent variable is at a certain value of an independent variable.

*Use multiple linear regression instead if you have more than one independent variable.

## Simple Linear Regression Assumptions

We normally have many distinct explanatory variable values in linear regression, and we usually believe that values between the observed values of the explanatory variables are also viable explanatory variable values. These are the assumptions:

- **Equal Variance Assumption (homoscedasticity).**

For all X's values, the variance of the $\varepsilon_i$'s is constant. The X's or residual plots of $\varepsilon_i$ against y can be used to detect this. We can assume constant variance if the residual plots have a rectangular shape. Non-constant variance exists and must be addressed if a residual plot reveals an increasing or decreasing wedge or bowtie shape.

- **Linearity Assumption**

The relationship between Y and Xs is linear (straight-line). Any curvilinear connection is not taken into account. Scatter plots are the easiest way to assess this early in your investigation. Residual plots can reveal nonlinear patterns.

- **Normality Assumption**

We assume the $\varepsilon_i$' s are normally distributed when hypothesis tests and confidence limits are to be used.

Techniques to Use: Shapiro-Wilk Test, Cramer-von Mises Test, Anderson-Darling Test, and Kolmogorov Test

- **Independence Assumption**

The $\varepsilon_i$' s are assumed to be uncorrelated with one another, which implies that the Y's are also uncorrelated. This assumption can be violated in two ways: model misspecification or time-sequenced data.

Techniques to Use: Durbin-Watson Test, QQ-plot

If the data does not match the homoscedasticity or normality assumptions, you may be able to apply a nonparametric test like the Spearman rank test instead.

**Simple Linear Regression Model**

We assume that the population means of the result and the value of the explanatory variable have a linear relationship. We may represent the structural model using the equation if $y_i$ is some outcome and $x_i$ is some explanatory variable.

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_{i,}\ 1 \le i \le n, \varepsilon_{i,} \overset{uncorr}{\sim} (0, \sigma^2)$$

Where $\beta_0$, read "beta zero", is the intercept parameter; and $\beta_1$, read "beta one", is the slope parameter. According to the structural model, the population means of Y (across all individuals with that specific value "$x_i$" for their explanatory variable) may be determined using the simple linear equation $\beta_0 + \beta_1 x_i$ for each value of $x_i$.

The error model we utilize assumes that each x has a Gaussian distribution around the population mean with a spread of $\sigma^2$, meaning that any value of x has the same value (and the corresponding population means of y).

Not only does the error model include the "Normality" and "Equal variance" assumptions, but it also includes the "fixed-x" assumption (the explanatory variable is measured without error).

**Statistical Hypothesis**

$$H_0: \beta_1 = 0 \quad v.s \quad H_1: \beta_1 \neq 0$$

The main null hypothesis in basic linear regression is $H_0: \beta_1 = 0$, whereas the alternative hypothesis is $H_1: \beta_1 \neq 0$. If the null hypothesis is correct, we can see from $E(x) = \beta_0 + \beta_1 x$ that the population means of $Y$ is 0 for any $x$ value, indicating that $x$ does not influence $Y$. Changes in x are linked to changes in $Y$, as an alternative, or changes in $x$ cause changes in $Y$ in a randomized experiment.

**Least Squares Estimation**

The Least Square Estimation is used when one seeks for the line to minimize the sum of square loss function between observed $y_i$ and fitted $(\beta_0 + \beta_1 x_i)$. The deviation of $Y_i$ from its expected value is

$$\varepsilon_i = Y_i - (\beta_0 + \beta_1 x_{1i})$$

Since $\beta_0$, $\beta_1$ are unknown, "Good" estimators of $\beta_0$, $\beta_1$, denoted by $b_0$, and $b_1$, should minimize the overall deviations,

$$Q = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (Y_i - b_0 - b_1 x_i)^2$$

called the (ordinary) least squares estimation (LSE, or OLS)

By calculus, we have

$$\frac{\partial Q}{\partial b_0} = -2 \sum_{i=1}^{n} (Y_i - b_0 - b_1 X_i) = 0$$

$$\frac{\partial Q}{\partial b_1} = -2 \sum_{i=1}^{n} X_i (Y_i - b_0 - X_i) = 0$$

Finally, we have the LSE,

$$b_1 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

$$b_0 = \frac{1}{n}\{\sum_{i=1}^{n} Y_i - b_1 \sum_{i=1}^{n} x_i\} = \hat{Y} - b_1\hat{X}$$

**Regression Calculation**

To get estimates of the parameters $\beta_0$, $\beta_1$, and $\sigma^2$, the basic regression analysis uses very simple formulas. These figures may be calculated using one of two methods, both of which provide the same results.

$$\widehat{\beta_1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}$$

$$\widehat{\beta_0} = \bar{y} - \widehat{\beta_1}\bar{x}$$

$$s^2 = \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{n-2}$$

The least squares method states that the line that minimizes the sum of the squared residuals, where the residuals are the vertical distances between individual points and the best-fit "regression" line, should be chosen as the best-fit line.

When we ask a computer to conduct basic linear regression, it employs these equations to determine the best fit line and then displays the parameter estimations.

**Simple Regression R-Code**

Lm(respondent variable ~ explanatory variable, data = data_name)

The code takes the data you have collected "data = data_name" and calculates the effect that the explanatory variable has on the respondent variable using the equation for the linear model: lm(). Also, the output table provides standard error, t-value, p-value, regression coefficient, etc...

```
Call:
lm(formula = time ~ speed, data = uav)

Residuals:
    Min      1Q  Median      3Q     Max
-62.406 -19.224   0.504  24.121  60.448

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  252.161      9.692  26.016  < 2e-16 ***
speed         -8.611      1.147  -7.505 1.19e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 29.9 on 73 degrees of freedom
Multiple R-squared:  0.4355,    Adjusted R-squared:  0.4278
F-statistic: 56.32 on 1 and 73 DF,  p-value: 1.194e-10
```

This output table repeats the formula that generated the results, then describes the model residuals ('Residuals'), which indicate how well the model matches the real data. The 'Coefficients' table follows next. The first row shows the y-intercept estimations, while the second row shows the model's regression coefficient. The first row of the table is labeled (Intercept). With a value of 252.161, this is the y-intercept of the regression equation of Time and Speed.

The standard error of the estimate is displayed in the **Std. Error** column. This figure depicts the degree of variability in our estimate of the relationship between the explanatory variable and response variable.

The test statistic is displayed in the **t value** column. The t-value from a two-sided t-test is utilized as the test statistic in linear regression unless you indicate otherwise. The greater the test statistic, the less likely our findings are to occur by chance.

The p-value is displayed in the **Pr(>| t |)** column. If the null hypothesis of no impact were true, this figure informs us how likely we are to witness the estimated influence of the explanatory variable on the respondent variable.

In simple regression, the strength of the relationship is determined by how near the simple correlation of $x$ and y is to 1 or -1. The $R^2$ value, often known as the multiple correlation coefficient, is a metric we utilize. $R^2$ is equivalent to the square of the simple correlation when used in simple regression.

$$R^2 = \frac{SS_{tot} - SS_{res}}{SS_{tot}} = \frac{explanied\ variability}{total\ variability}$$

$R^2$ is the percentage of the total variance in $Y$ that can be explained by utilizing $x$ data in regression. $R^2$ is always in the range of 0 to 1. An $R^2$ of 0 indicates that $x$ does not reveal anything about $Y$. An $R^2$ of 1 indicates that using $x$ data allows for flawless $y$ prediction, with every point on the scatterplot falling perfectly on the

regression line.

However, we need a new technique to quantify the overall strength of the regression in multiple regression since there are many associations (one for each $x$).

**Multiple Linear Regression**

Multiple regression is very similar to Simple Linear Regression but there are p explanatory variables in multiple linear regression, and the relationship between the dependent variable and the explanatory variables is represented by the equation:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \varepsilon_{i,}$$

A multiple regression formula has one y-intercept and several slopes (one for each variable). It is understood similarly to a basic linear regression formula, with the exception that there are several factors that all influence the slope of the relationship. It is difficult to get the coefficients of the variables without using a computer.

**Multiple Linear Regression Assumptions**

The following assumptions must be considered when using multiple regression analysis: Constance Variance, linearity, normality, independence, and Multicollinearity (the first four assumptions are mentioned in the Simple Regression Section).

**Multicollinearity**

The occurrence of near-linear connections among the set of independent variables is referred to as collinearity or multicollinearity. Multicollinearity can lead to erroneous regression coefficient estimates, inflated regression coefficient standard errors, deflated partial t-tests for regression coefficients, misleading nonsignificant p-values, and decreased model predictability.

**Collinearity Detection**

1) Consider the variance inflation factors (VIF). Large VIF's flag collinear variables
2) Look for near-perfect relationship in the pairwise scatter plots of pairs of independent variables
3) Focus on small eigenvalues of the correlation matrix of the independent variables.

The answers will differ depending on the source of collinearity. If the data collection generated the collinearity, then gather more data over a larger $x$ -subspace. If the linear model you chose accentuated the collinearity, use variable selection techniques to reduce the model. If the collinearity was caused by one or more observations, eliminate them and proceed as needed. Above all, be cautious while choosing variables from the start.

**Centering in Regression**

The process of establishing a reference value for each predictor and coding the data based on that reference value is known as centering, and it ensures that each regression coefficient generated and evaluated is relevant to the research issue. When using non-centered data in regression analysis, which refers to the widespread practice of inputting predictors in their original score format, the results are sometimes inconsistent and misleading. Unnecessary centering is inexpensive, but the expense of not centering when it is required might be substantial. As a result, it is usually preferable to concentrate on regression analyses.

**Stepwise Regression**

All variables are examined at the same time in ordinary least squares regression. Stepwise regression differs from other types of regression in that variables are added or removed from the model one at a time. We begin with the entire model and delete the least significant variable.

- **Backward stepwise regression**: the model is re-fitted to this group of variables, and the new least significant variable is removed from the equation. This procedure is repeated until all non-significant variables in the dataset have been eliminated.

- **Forward stepwise regression**: start with no variables in the model and add the most significant variable. The next most important variable is chosen from the remaining variables and added to the model. This process is continued until a new variable does not enhance the model's fit sufficiently to warrant its inclusion.

**Regularization**

To avoid overfitting, regularization penalizes high-valued regression coefficients. It compresses (simplifies) the model by lowering the number of parameters. This more straightforward model will almost probably do better in terms of predicting. More sophisticated models are penalized by regularization, which ranks them from least to greatest overfitting; the model with the lowest "overfitting" score is usually the best choice for predictive power.

Because least squares regression approaches, which minimize the residual sum of squares, can be unstable, regularization is required. This is true if the model has multicollinearity.

Regularization works by skewing data to a specific set of values (such as small values near zero). A tuning parameter is added to promote certain values, resulting in the bias:

- An L1 penalty equal to the absolute value of the magnitude of coefficients is added during L1 regularization. To put it another way, it sets a restriction on the size of the coefficients. L1 can produce sparse models (i.e., models with few coefficients), with certain coefficients becoming zero and being deleted. This approach is used in **Lasso regression**.
- L2 regularization imposes an L2 penalty equal to the square of the coefficient magnitude. L2 does not produce sparse models, and all coefficients are reduced by the same amount (none are eliminated). This approach is used in **Ridge regression** and SVMs.

**All Possible Regression**

All subset regression tests all possible subsets of the set of potential independent variables. There are $2^k$ unique subsets of independent variables to be examined if there are K possible independent variables.

When employing an all-possible-regressions process, you are usually given the option of ranking the models based on many numerical factors. Adjusted R-squared and the Mallows "Cp" statistic are the two most usually utilized. The latter statistic is similar to adjusted R-squared, but it penalizes you more for having more independent variables. Cp does not have a 0-to-1 scale.

Its values are usually positive and larger than 1, with lower values being preferable. The models that provide the best (lowest) values of Cp are likely to be comparable to those that produce the best (highest) values of adjusted R-squared, but the actual rankings may change slightly. When all other factors are equal, the Cp criteria favors models with fewer parameters, which makes it less likely to overfit the data.

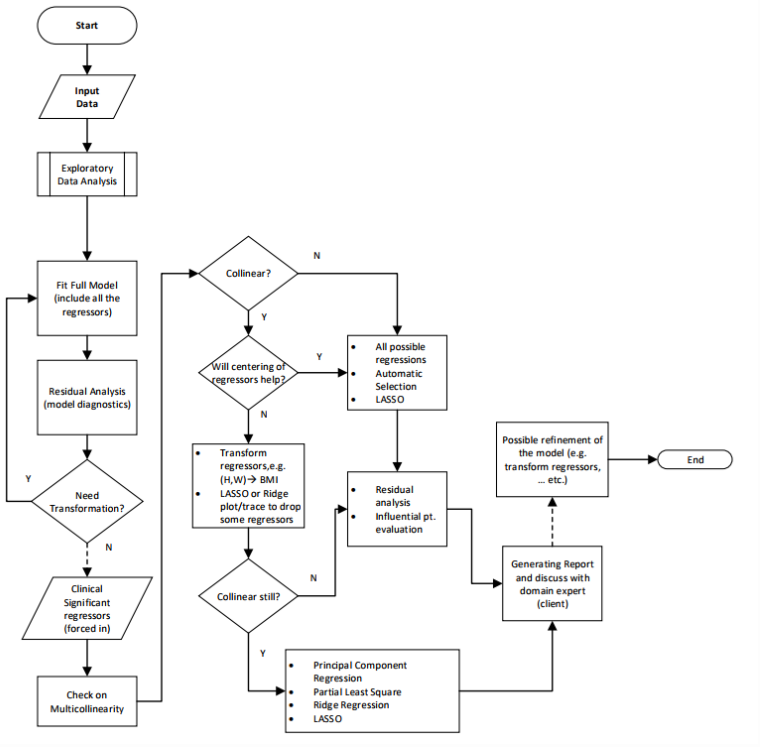
**Cross-validation**

Cross-validation is a more effective method of testing a model than relying just on residuals. Residual evaluation does not indicate how well a model can make new predictions on cases it has not already seen. When developing a model, cross-validation strategies tend to focus on not utilizing the complete data set.

- Before the data is modeled, certain examples are deleted; these cases are referred to as the testing set.
- After the model has been developed using the remaining examples (commonly referred to as the training set), the cases that were deleted (testing set) can be used to evaluate the model's performance on "unseen" data (i.e. the testing set).

**Implementation**

For the implementation, we will examine the relationship between explanatory variables and response variables in the Data: UAV using the R and the flow chart listed below.




R-Code and analysis are included in the Appendix part.

**Logistic regression**

In many cases, such as regression, we are interested in input-output interactions, but the output variable is discrete rather than continuous. There are numerous situations in which we have binary outcomes. We have several input variables, which may or may not be continuous, in addition to the binary output. How could such data be modeled and analyzed?

One of the most used methods for fitting models to categorical data, particularly binary answer data, is logistic regression. It belongs to a class of statistical models known as generalized linear models, and it is the most significant and possibly most commonly used among them in data science. Unlike linear regression, logistic

regression can predict probability values inside the (0,1) interval directly; moreover, such probabilities are well-calibrated when compared to probabilities predicted by other classifiers like Naive Bayes. The marginal probabilities of the training data are preserved using logistic regression. The model's coefficients also give some insight into the relative relevance of each input variable through the odds ratio.

**Logistic Regression Model**

Supply the value of the linear equation by logit to obtain the formula for logistic regression, and since it operates on a probability model, we must offer the value of the linear equation by logit (log odds).

The log odds of "success" in the X model for a single predictor are as follows

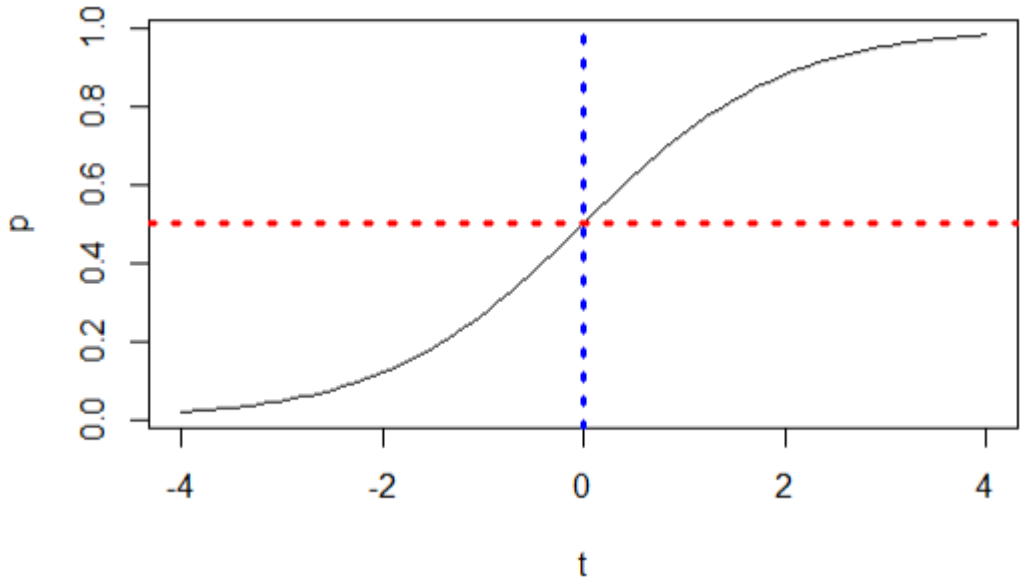
$$\log\left(\frac{\boldsymbol{p}}{1-\boldsymbol{p}}\right) = \boldsymbol{b_0} + \boldsymbol{b_1 x}$$

$p$ is expressed as

$$p = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

where $p$ is the probability of an event occurring. As $p$ grows or decreases with X, depending on the sign of $\beta_1$, and follows a "sigmoidal" pattern. If $\beta_1 = 1$, then $p$ is independent of X.

To minimize the misclassification rate, we should predict Y = 1 when $p \geq 0.5$ and Y = 0 when $p < 0.5$. This means guessing 1 whenever $\beta_0 + \beta_1 X$ is non-negative, and 0 otherwise. As a result, logistic regression gives us a linear classifier.




However, unlike linear regression where the regression coefficients can be estimated easily by Least Square Estimation (LSE), the score equation involves the nonlinear form of coefficient and voids the use of LSE. The common approach for estimating the coefficients in binary regression is the maximum likelihood estimation. Though the maximum likelihood estimator does not have a close form, it can be estimated by numerical algorithm, for example, the Newton-Raphson algorithm or Fisher's scoring method.

**Likelihood Function for Logistic Regression**

We can use likelihood to fit logistic regression since it predicts probabilities rather than classes. We have a vector of features, $x_i$, and an observed class, $y_i$, for each training data point. Since $y_i \overset{iid}{\sim} \text{Bernoulli}(p_i)$, the likelihood is

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i)^{1-y_i} \text{ where } p(x_i) = \sum_{i=0}^{n} \frac{1}{1 + e^{\beta_0 + x_i \beta_1}} e^{\beta_0 + x_i \beta_1} x_i$$

The log-likelihood turns products into sums:

$$\ell(\beta_0, \beta) = \sum_{i=1}^{n} y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i))$$

$$= \sum_{i=1}^{n} \log(1 - p(x_i)) + \sum_{i=1}^{n} y_i \log \frac{p(x_i)}{1 - p(x_i)}$$

$$= \sum_{i=1}^{n} \log(1 - p(x_i)) + \sum_{i=1}^{n} y_i (\beta_0 + x_i \beta_1)$$

$$= \sum_{i=1}^{n} -\log(1 + \exp(\beta_0 + x_i\beta_1)) + \sum_{i=1}^{n} y_i (\beta_0 + x_i\beta_1)$$

**Maximum likelihood estimates for Logistic Regression**

To find the maximum likelihood estimates we'd differentiate the log-likelihood with respect to the parameters, set the derivatives equal to zero, and solve. To start that, take the derivative with respect to $\beta_0$, and $\beta_1$

$$\frac{\partial \ell}{\partial \beta_0} = -\sum_{i=0}^{n} \frac{1}{1 + e^{\beta_0 + x_i \beta_1}} e^{\beta_0 + x_i \beta_1} x_{i0} + \sum_{i=0}^{n} y_i x_i$$

$$= \sum_{i=0}^{n} (y_i - p(x_i; \beta_0, \beta_1)) x_{i0})$$

$$\frac{\partial \ell}{\partial \beta_1} = -\sum_{i=0}^{n} \frac{1}{1 + e^{\beta_0 + x_i \beta_1}} e^{\beta_0 + x_i \beta_1} x_{i1} + \sum_{i=0}^{n} y_i x_i$$

$$= \sum_{i=0}^{n} (y_i - p(x_i; \beta_0, \beta_1)) x_{i1})$$

Since this is a transcendental equation and there is no closed-form solution, as a result, we are not going to be able to set this to zero and solve it so taking second derivative and use Newton-Raphson method for solving equation numerically:

$$\frac{\partial^2 \ell}{\partial {\beta_0}^2} = -\sum_{i=0}^{n} x_{i0} x_i p(x_i; \beta_0, \beta_1)(1 - p(x_i; \beta_0, \beta_1))$$

$$\frac{\partial^2 \ell}{\partial {\beta_1}^2} = -\sum_{i=0}^{n} x_i x_{i1} p(x_i;\beta_0,\beta_1)\big(1 - p(x_i;\beta_0,\beta_1)\big)$$

$$\frac{\partial^2 \ell}{\partial {\beta_0}^2 \partial {\beta_1}^2} = -\sum_{i=0}^{n} x_{i0} x_{i1} p(x_i;\beta_0,\beta_1)\big(1 - p(x_i;\beta_0,\beta_1)\big)$$

**Newton - Raphson Method**

Newton-Raphson technique is an iterative approach that uses Taylor's expansion to determine the root of the graph. A root is a place where a graph crosses the x-axis and is our coefficient. The Newton Raphson technique can be used to estimate the coefficients:

Let's begin with the simple case of maximizing a function of one scalar variable, such as $f(\beta)$. We want to find the location of the global minimum, $\beta$. We suppose that f is smooth, and that $\beta$ is a regular interior minimum, meaning that the derivative at $\beta$ is zero and the second derivative is positive. Near the maximum we could make a Taylor expansion around $\beta^*$

$$f'(\beta) \approx f'(\beta^*) + \frac{f''(\beta^*)}{2}(\beta - \beta^*)^2$$

$f(\beta^*)$is close to quadratic near the maximum when the second derivative has to be negative to ensure $f(\beta) < f(\beta)$. Newton's method utilizes this fact to maximize a quadratic approximation to the function we are looking for. If the initial point $\beta^*$ is close to the maximum, take a second-order Taylor expansion around $\beta^*$.

$$0 = f'(\beta^*) + \frac{1}{2} f''(\beta^*) 2 (f'(\beta - \beta^*)$$

$$\beta = \beta^* - \frac{f'(\beta^*)}{f''\beta^*)}$$

The value $\beta_1$ should be a better guess at the minimum $\beta^*$ than the initial one $\beta_0$ was. So if we use it to make a quadratic approximation to $f$, we'll get a better approximation, and so we can iterate this procedure, minimizing one approximation and then using that to get a new approximation

$$\beta_{i+1} = \beta_i - \frac{f'(\beta_i)}{f''(\beta_i)}$$

$\beta_{i+1}$ is the estimated value of one of the coefficients, and $\beta_i$ is the value of the same coefficient determined in the previous iteration. The above formula is repeated until a sufficiently precise value is obtained.

To apply the Newton-Raphson algorithm, one would raise the dimension of parameter and replace function *f* by the log-likelihood function. Specifically,

$$l' = \mathbb{X}^T(y - p)$$

$$l'' = -\mathbb{X}^T \mathbb{V} \mathbb{X}^T = \begin{pmatrix} 1 & x \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} p_1(1-p_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & p_n(1-p_n) \end{pmatrix} \begin{pmatrix} 1 & \cdots & 1 \\ x & \cdots & x_n \end{pmatrix}$$

Equations $l' and\ l''$ represent the vector of initial approximation for each $\beta_i$.

Using the terms defined above, the Newton-Raphson algorithm can be written in a very similar form.

$$\widehat{\overrightarrow{\beta_{i+1}}} = \widehat{\overrightarrow{\beta_i}} - [l'']^{-1} \cdot l'$$

**Implementation**

Using the UAV data set as an example, the following R code shows how to implement the Newton-Raphson procedure. Note that the update procedure terminates when the distance between the new and old coefficient vector is less than 0.0001.

1) Example using single X

```
mylogit <- glm(time_over_200 ~ speed, data = mydata, family = binomial(link="logit"))
summary(mylogit)

#Store the data in matrix form
mydata <- as.matrix(mydata)
N <- nrow(mydata)
Y <- mydata[,10]
x0 <- matrix(1,N,1)
X <- cbind(x0,mydata[,7])

#refers to y_bar,keep Y==1 as success
BETA <- matrix(c(length(which(Y==1))/N,0),ncol=1);

DIFF <- 1;

while (DIFF > 0.0001) {
  # Define the p(x),calculating the vector of probabilities
  p <- 1/(exp(-X %*% BETA)+1)
  # Element wise multiplication
  V <- diag(c(p*(1-p)))
  # First derivative(Score vector)
  First_deriv <- t(X)%*%(Y-p)
  # Second derivative(Hessian matrix)
  Second_deriv <- -t(X)%*%V%*%X
  # Multiplying Score and hessian matrix
  step <- -solve(Second_deriv)%*%First_deriv
  # Each time updated by step wise
  NewBETA <- BETA + step
  # Keep looping going
  DIFF <- max(abs(step))
  # Reset
  BETA <- NewBETA
}
```

```
Call:
glm(formula = time_over_200 ~ speed, family = binomial(link = "logit"),
    data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7167  -0.6893  -0.2057   0.7214   2.7808

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   3.7427     0.9712   3.854 0.000116 ***
speed        -0.6323     0.1454  -4.348 1.38e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 95.477  on 74  degrees of freedom
Residual deviance: 61.955  on 73  degrees of freedom
AIC: 65.955

Number of Fisher Scoring iterations: 5

           [,1]
[1,]  3.7426637
[2,] -0.6323381
```

2) Example using more than one X

```
mylogit <- glm(time_over_200 ~ wind_speed+wind_angle+battery_voltage+battery_current+speed + payload, data = mydata, family = binomial(link="logit"))
summary(mylogit)

#Store the data in matrix form
uav <- as.matrix(uav)
N <- nrow(uav)
Y <- uav[,10]
x0 <- matrix(1,N,1)
X <- cbind(x0,uav[,3:8])

#refers to y_bar,keep Y==1 as success
BETA <- matrix(c(length(which(Y==1))/N,0,0,0,0,0,0),ncol=1);

DIFF <- 1;

while (DIFF > 0.0001) {
  # Define the p(x),calculating the vector of probabilities
  p <- 1/(exp(-X %*% BETA)+1)
  # Element wise multiplication
  V <- diag(c(p*(1-p)))
  # First derivative(Score Vector)
  First_deriv <- t(X)%*%(Y-p)
  # Second derivative(Hessian matrix)
  Second_deriv <- -t(X)%*%V%*%X
  # Multiplying Score and hessian matrix
  step <- -solve(Second_deriv)%*%First_deriv
  # Each time updated by step wise
  NewBETA <- BETA + step
  # Keep looping going
  DIFF <- max(abs(step))
  # Reset
  BETA <- NewBETA
}

print(BETA)
```

```
Call:
glm(formula = time_over_200 ~ wind_speed + wind_angle + battery_voltage +
    battery_current + speed + payload, family = binomial(link = "logit"),
    data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7707  -0.5133  -0.1412   0.5021   2.9655

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      6.727e+00  1.326e+01   0.507 0.611877
wind_speed       3.259e-01  4.803e-01   0.679 0.497403
wind_angle       8.818e-06  1.709e-04   0.052 0.958863
battery_voltage  1.658e-01  5.169e-01   0.321 0.748424
battery_current -4.251e-01  2.867e-01  -1.483 0.138188
speed           -9.035e-01  2.377e-01  -3.801 0.000144 ***
payload          5.712e-03  3.179e-03   1.797 0.072335 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 95.477  on 74  degrees of freedom
Residual deviance: 51.144  on 68  degrees of freedom
AIC: 65.144

Number of Fisher Scoring iterations: 12

                        [,1]
                6.726813e+00
wind_speed      3.259156e-01
wind_angle      6.905822e-06
battery_voltage 1.657646e-01
battery_current -4.250904e-01
speed           -9.035364e-01
payload          5.712343e-03
```

As you can see from the results of the examples, the implementation remains the same for both examples and only the $\mathbb{X}$ matrix varies.

**Summary**

In this project, a simple and multiple regression model was used to derive the relationship between flying time and wind speed, wind angle, battery voltage, battery current, speed, and payload of the UAV. The simple linear regression was focused on utilizing the model to explain how linear regression works and how to derive the summary table. Then the multiple regression analysis was focused on finding the explanatory variables that most affect the flying time of UAVs. By using the methods listed in the paper, the final model for the UAV was derived and we see that the wind angle, battery current, speed, and payload have the most effects on the flying time. To have a long flying time for a UAV, consider this information:

1) The wind should blow from the back of the UAV
2) Keep the Battery current as low as possible
3) Fly speed must not be so high, it will lead to the use of more Battery Current
4) Fly with Low weight or no payload.

Professionals in a variety of sectors use logistic regression to categorize data for a variety of objectives. Many individuals will benefit from using logistic regression to better understand their data and forecast patterns among their constituents. It enables us to make predictions about future data before it becomes accessible. It is based on the majority rule and will not accurately forecast outcomes for all goods, persons, or issues taken into account. Nonetheless, it is highly successful in forecasting a high probability of correctness for a large portion of the subject group under consideration.

# Appendix

# Math 374 Project

```
pkglst <- c("ISLR","latex2exp","olsrr","car")

tba_pkg <- pkglst[!(pkglst %in% installed.packages())]

if (length(tba_pkg)) {
  install.packages(tba_pkg,dependencies=TRUE,repos="https://cloud.r-project.org")
}

for (i in c(1:length(pkglst)))
{
library(pkglst[[i]],character.only=TRUE)
  }
```

```
## Warning: package 'ISLR' was built under R version 4.1.3
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```
## Loading required package: carData
```

```
ipar=par()
panel.hist <- function(x, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}
```

## 1. Load data and view the description of variables

```
uav = read.csv("UAV.csv")
print(sprintf("the number of cases with missing values is %s", sum(!complete.cases(uav))))
```

```
## [1] "the number of cases with missing values is 0"
```

```
dim(uav)
```

```
## [1] 75 10
```

```
names(uav)
```

```
##  [1] "flight"          "time"            "wind_speed"      "wind_angle"
##  [5] "battery_voltage" "battery_current" "speed"           "payload"
##  [9] "altitude"        "time_over_200"
```

There are **75** observations and 9 variables, indicating the performance Unmanned Aerial Vehicle of 75 flights.

We see that there are no missing values. In order to find which variable can influence the flight time we will use these features:

“time”,“wind_speed”,“wind_angle”,“battery_current”,“speed”,“payload”

#Variable List

Flight: an integer that represents the code of the flight performed. A flight is defined as the data set recorded from the take-off to landing in a predefined route

Time: the time of UAV fly in the air

Wind_speed: Average Airspeed provided by the anemometer in meters per second (m/s).

Wind_angle: Average angle in degrees (deg) of the air flowing through the anemometer with respect to the north.

Battery_voltage: Average system voltage in Volts (V) measured immediately after the battery

Battery_current: Average system current in Ampere (A) measured immediately after the battery.

Speed: Average programmed horizontal ground speed during cruise in meters per second (m/s)

Payload: mass of the payload attached to aircraft in grams (g). The payload used was confined in a standard USPS Small Flat Rate Box.

Altitude: predefined altitude in meters (m). The aircraft takes off vertically until it reaches the preset altitude.

Time_over_200 : whether the UAV flying time is over 200 seconds

```
demo = uav[complete.cases(uav),c("time","wind_speed","wind_angle","battery_current","speed","payload")]

dim(demo)
```

```
## [1] 75  6
```

```
head(demo)
```

```
##     time wind_speed wind_angle battery_current speed payload
## 1 200.70   3.898058   133.5661        17.93040     4       0
## 2 271.20   3.522941   174.0111        15.99376     4       0
## 3 180.10   4.581182   188.1649        15.75132     6       0
## 4 171.00   4.596319   164.9439        14.49325     8       0
## 5 217.00   3.339100   172.2958        15.30548     4       0
## 6 204.49   4.027639   193.0638        17.16973     4       0
```

```
tail(demo)
```

```
##      time wind_speed wind_angle battery_current speed payload
## 70 188.30   6.670192   209.1627        22.10957    12     500
## 71 155.60   5.585645   208.0347        19.16656    10     500
## 72 250.21   5.831995   256.1861        17.39911     4       0
## 73 160.69   4.563220   189.2933        18.52801    12     500
## 74 196.30   3.867380   187.1001        19.91028     4     250
## 75 177.20   3.756720   122.7608        14.91345     8     250
```

```
summary(demo)
```

```
##       time         wind_speed      wind_angle        battery_current
##  Min.   :106.1   Min.   :2.914   Min.   :    107.6   Min.   :14.10
##  1st Qu.:156.1   1st Qu.:4.183   1st Qu.:    172.5   1st Qu.:16.74
##  Median :179.6   Median :5.045   Median :    196.2   Median :18.73
##  Mean   :184.2   Mean   :4.929   Mean   :  21609.9   Mean   :18.84
##  3rd Qu.:205.3   3rd Qu.:5.708   3rd Qu.:    214.5   3rd Qu.:20.93
##  Max.   :271.2   Max.   :6.670   Max.   :1606737.0   Max.   :23.14
##      speed           payload
##  Min.   : 4.000   Min.   :  0.0
##  1st Qu.: 5.000   1st Qu.:  0.0
##  Median : 8.000   Median :250.0
##  Mean   : 7.893   Mean   :233.3
##  3rd Qu.:10.000   3rd Qu.:500.0
##  Max.   :12.000   Max.   :750.0
```

We then will be using the data set demo that contains six features (to regress Time on five covariates) for 75 cases/flights.

# 2. Simple Linear Regression

```
S_regression = lm(time ~ speed, data = uav)
summary(S_regression)
```

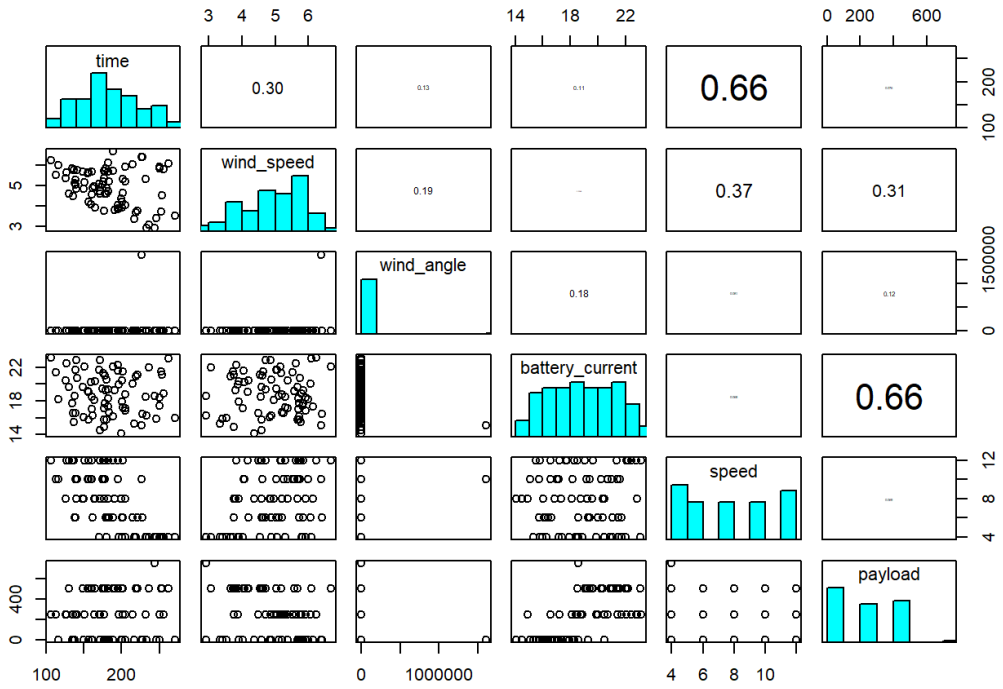
```
##
## Call:
## lm(formula = time ~ speed, data = uav)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.406 -19.224   0.504  24.121  60.448
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  252.161      9.692  26.016  < 2e-16 ***
## speed         -8.611      1.147  -7.505 1.19e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29.9 on 73 degrees of freedom
## Multiple R-squared:  0.4355, Adjusted R-squared:  0.4278
## F-statistic: 56.32 on 1 and 73 DF,  p-value: 1.194e-10
```

# 3. Explore the data

First, let's have a look on scatter matrix and correlation matrix. The intention is to find the evidence of

1. correlation between $y$ and $x_i$'s.
2. correlation among $x_i$'s

```
pairs(demo, upper.panel = panel.cor, diag.panel = panel.hist)
```




The time is not strongly correlated with any of the covariates except the Speed.Overall some covariates are considered weak correlated. The former does not necessarily imply a minor regression impact, but the later raises questions about collinearity.

# 4. Multiple Regression Model

$$Time = \beta_0 + \beta_1 WindSpeed + \beta_2 WindAngle + \beta_3 BatteryCurrent + \beta_4 Speed + \beta_5 Payload + \epsilon$$

```
full = lm(time ~., data = demo)
summary(full)
```

```
##
## Call:
## lm(formula = time ~ ., data = demo)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.520 -19.689  -3.984  22.607  55.880
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.111e+02  3.400e+01   9.148 1.61e-13 ***
## wind_speed       7.267e-01  4.431e+00   0.164   0.8702
## wind_angle       3.689e-05  1.849e-05   1.995   0.0500 *
## battery_current -3.912e+00  1.959e+00  -1.997   0.0498 *
## speed           -8.940e+00  1.203e+00  -7.432 2.18e-10 ***
## payload          5.564e-02  2.277e-02   2.444   0.0171 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.39 on 69 degrees of freedom
## Multiple R-squared:  0.5189, Adjusted R-squared:  0.4841
## F-statistic: 14.89 on 5 and 69 DF,  p-value: 6.793e-10
```

$$\mathrm{H}_0 : \beta_1 = \beta_2 = \cdots = \beta_5 = 0, \ \text{vs.} \mathrm{H}_1 : \beta_1^2 + \beta_2^2 + \cdots + \beta_5^2 \neq 0$$

We sees that the $F$-test on regression effect is significant with $p$-value at <6.793e-10, indicating at least one of the regression coefficient is significantly different from 0. With significant regression effect, we then perform model diagnostics to ensure the appropriateness for further inferential work. Things to watch/look for:

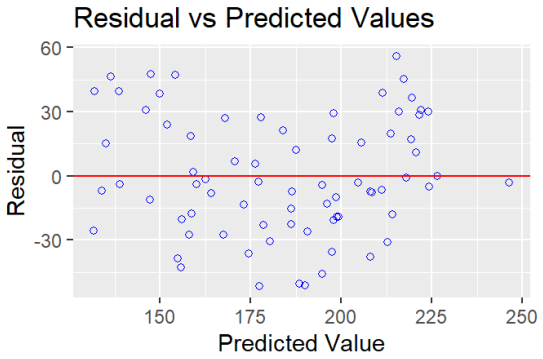
1. Violation against Homoscedasticity.
2. Violation against Normality
3. Violation against Independence

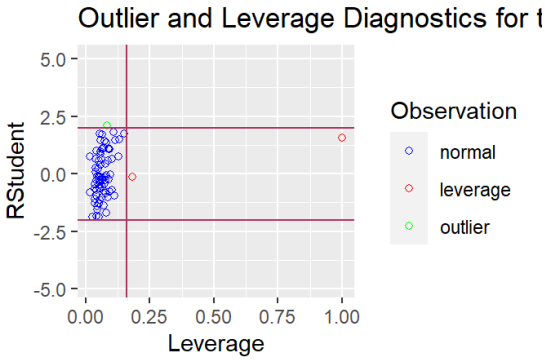
# 4-1. Model diagnostics

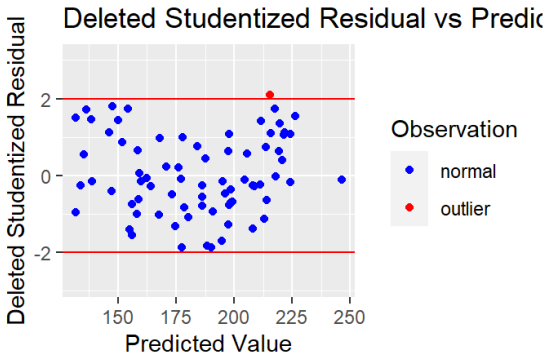
Function "*ols_plot_diagnostics*" in **olsrr** provides a set of diagnostic plots.

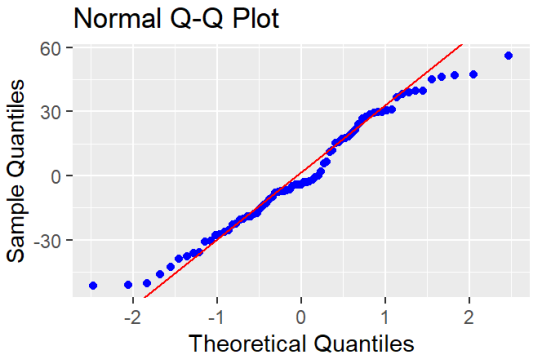
```
ols_plot_diagnostics(full, print_plot = TRUE)
```

page 1 of 3

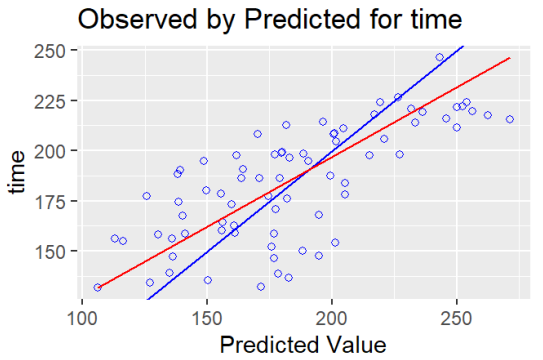


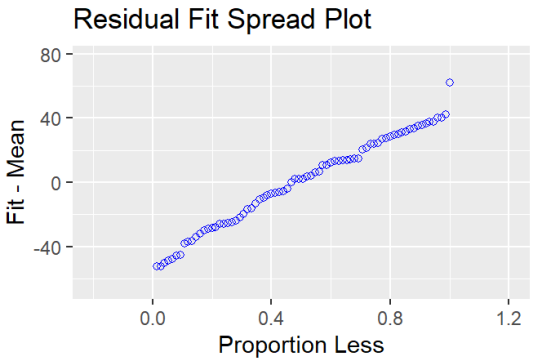



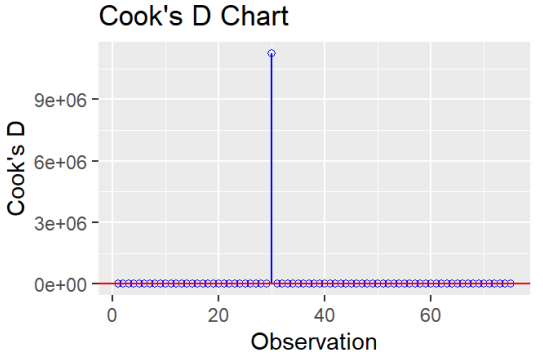



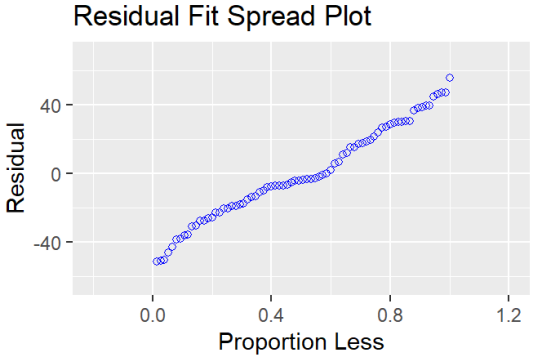




page 2 of 3

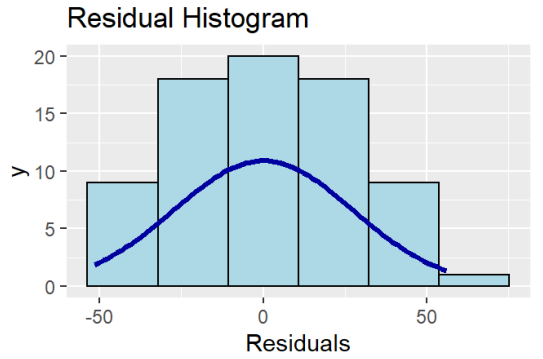


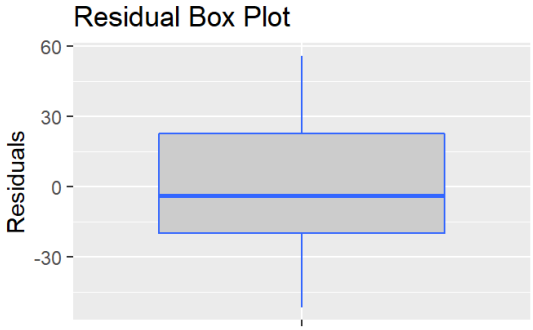




page 3 of 3




The Q-Q plot shows the




residuals are mostly along the diagonal line, but it deviates a little near the top and the bottom. Generally, the model has relatively normally distributed model, so we can trust the regression model results without much concern

```
durbinWatsonTest(full,max.lag=1,alternative="positive")
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.1734908      1.644378   0.047
##  Alternative hypothesis: rho > 0
```

```
durbinWatsonTest(full,max.lag=1,alternative="negative")
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.1734908      1.644378   0.958
##  Alternative hypothesis: rho < 0
```

We also test for correlation by Durbin Watson tests (one for positive correlation, one for negative correlation). Values below 2.0 mean there is positive autocorrelation and above 2.0 indicates negative autocorrelation. which indicate there is no significant evidence against zero correlation.

```
ols_test_normality(full)
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.9736         0.1185
## Kolmogorov-Smirnov        0.0867         0.5945
## Cramer-von Mises          6.8039         0.0000
## Anderson-Darling          0.565          0.1388
## -----------------------------------------------
```

One core assumption of linear regression analysis is that the residuals of the regression are normally distributed.

The p-value for all the tests (except for the Kolmogorov-Smirnov, which is just on the border) is greater than 0.05, all the normality tests are significant, meaning normality is not violated

When the normality assumption is violated, interpretation and inferences may not be reliable or not at all valid.

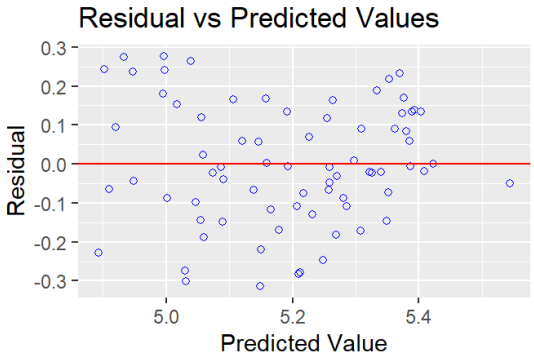
Also, looking at the histogram of residuals along with heteroscedasticity (bell shaped scatter but perfectly not fitted), logarithm transformation is not needed but we will attempt it for this data.

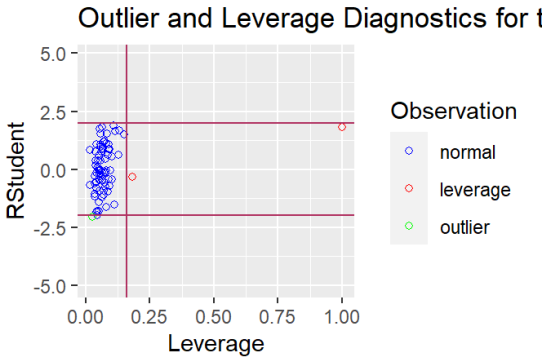
# 4-2. Variable transformation

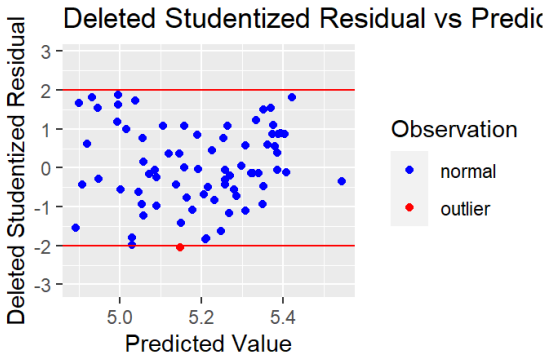
```
demo$time = log(demo$time)
lnfull = lm(time~.,data = demo)
summary(lnfull)
```

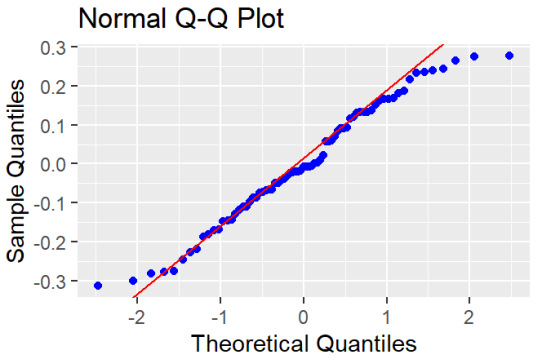
```
##
## Call:
## lm(formula = time ~ ., data = demo)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.313703 -0.103372 -0.008019  0.131985  0.276484
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.956e+00  1.907e-01  31.223  < 2e-16 ***
## wind_speed      -4.181e-04  2.486e-02  -0.017   0.9866
## wind_angle       1.971e-07  1.037e-07   1.900   0.0616 .
## battery_current -2.471e-02  1.099e-02  -2.248   0.0277 *
## speed           -4.735e-02  6.747e-03  -7.017 1.24e-09 ***
## payload          3.180e-04  1.277e-04   2.490   0.0152 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1593 on 69 degrees of freedom
## Multiple R-squared:  0.503,  Adjusted R-squared:  0.467
## F-statistic: 13.97 on 5 and 69 DF,  p-value: 1.994e-09
```

```
ols_plot_diagnostics(lnfull, print_plot = TRUE)
```

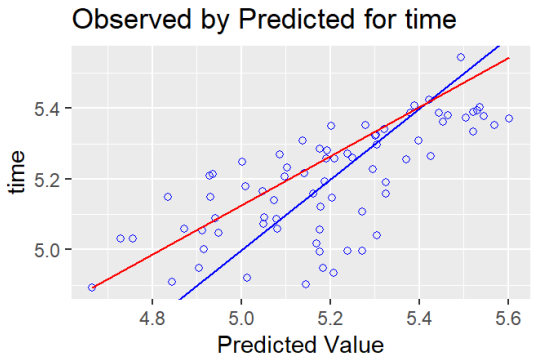

Residual vs Predicted Values
Residual
Predicted Value

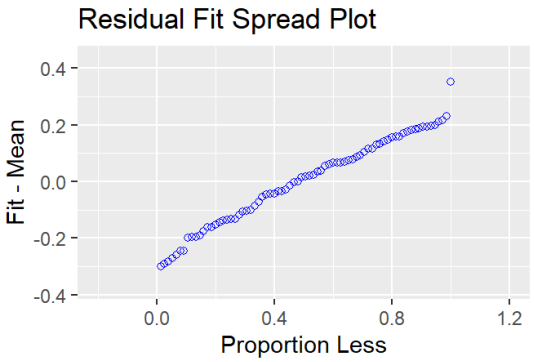

Outlier and Leverage Diagnostics for t
RStudent
Leverage
Observation
normal
leverage
outlier

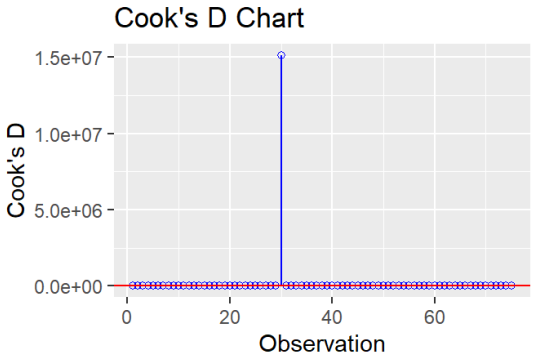

Deleted Studentized Residual vs Predic
Deleted Studentized Residual
Predicted Value
Observation
normal
outlier

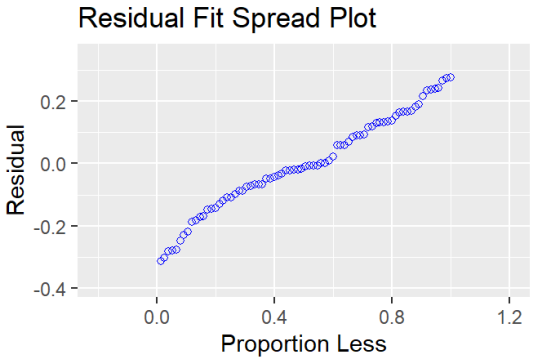

Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles



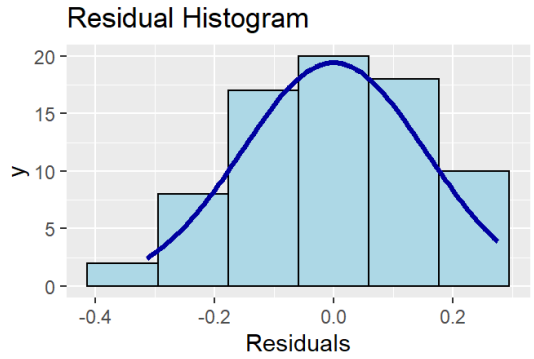

Observed by Predicted for time
time
Predicted Value

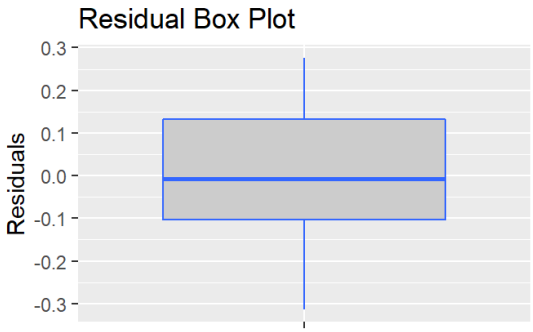

Residual Fit Spread Plot
Fit - Mean
Proportion Less


Cook's D Chart
Cook's D
Observation


Residual Fit Spread Plot
Residual
Proportion Less




Residual Histogram
y
Residuals




```
durbinWatsonTest(lnfull,max.lag=1,alternative="positive")
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.2670207      1.458946   0.004
##  Alternative hypothesis: rho > 0
```

```
durbinWatsonTest(lnfull,max.lag=1,alternative="negative")
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.2670207      1.458946   0.995
##  Alternative hypothesis: rho < 0
```

```
ols_test_normality(lnfull)
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.9762         0.1704
## Kolmogorov-Smirnov        0.0696         0.8363
## Cramer-von Mises         17.8788         0.0000
## Anderson-Darling          0.3857         0.3830
## -----------------------------------------------
```

normality, heteroscedasticity, correlation assumption still meets the condition.

# 4-3. Evaluation of Multicollinearity

```
ols_coll_diag(lnfull)
```

```
## Tolerance and Variance Inflation Factor
## ---------------------------------------
##         Variables Tolerance      VIF
## 1      wind_speed 0.6595075 1.516283
## 2      wind_angle 0.9260737 1.079828
## 3 battery_current 0.4917670 2.033484
## 4           speed 0.8204002 1.218917
## 5         payload 0.4389948 2.277932
##
##
## Eigenvalue and Condition Index
## ------------------------------
##    Eigenvalue Condition Index    intercept   wind_speed   wind_angle
## 1 4.496722972        1.000000 4.464256e-04 1.024823e-03 0.0008653353
## 2 1.002878132        2.117503 6.676063e-07 2.375797e-05 0.8809818394
## 3 0.392951695        3.382817 8.316492e-04 4.872641e-03 0.0641975370
## 4 0.088091621        7.144647 1.190263e-02 1.009761e-02 0.0042406756
## 5 0.014696426       17.492110 1.309205e-01 9.730304e-01 0.0268517986
## 6 0.004659155       31.066656 8.558982e-01 1.095074e-02 0.0228628141
##   battery_current        speed     payload
## 1    3.775739e-04 4.720511e-03 0.006613138
## 2    1.299018e-05 4.452041e-05 0.006010296
## 3    9.907642e-05 1.190597e-02 0.414810574
## 4    9.537624e-03 8.696945e-01 0.004418885
## 5    5.650525e-02 1.009438e-01 0.209494906
## 6    9.334675e-01 1.269063e-02 0.358652201
```

# 4-4.Variable Selection

We will try following schemes for variable selection:

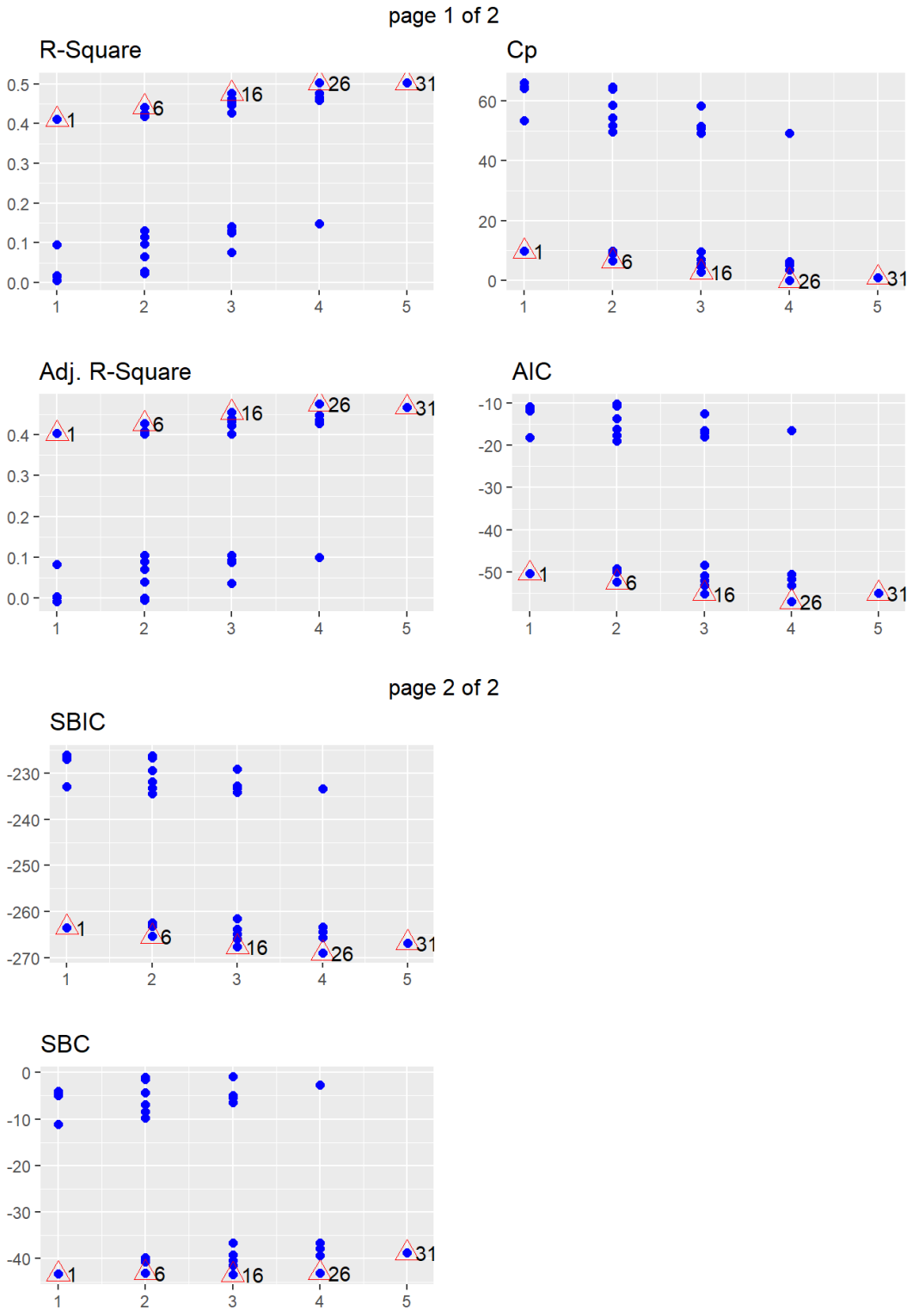
1. All possible regression model under *Mallow's* $Cp$, $R^2_{\text{Adj}}$, *AIC*, *SBIC*, *SBC*.
2. Best subset
3. Forward, Backward, Stepwise(both)
4. LASSO

# 4-4-1.All Possible Regression Model

All subset regression tests all possible subsets of the set of potential independent variables. If there are K potential independent variables (besides the constant), then there are $2^k$ distinct subsets of them to be tested.

This shows the panel of fit criteria for all possible regression methods

```
select01 = ols_step_all_possible(lnfull)
plot(select01)
```




In each of the criterion, the red triangles mark the best model among ones of the same complexity level~(number of predictors)

```
(Mod_cp=select01$predictors[which.min(select01$cp)])
```

```
## [1] "wind_angle battery_current speed payload"
```

```
(Mod_adjr=select01$predictors[which.max(select01$adjr)])
```

```
## [1] "wind_angle battery_current speed payload"
```

```
(Mod_aic=select01$predictors[which.min(select01$aic)])
```

```
## [1] "wind_angle battery_current speed payload"
```

```
(Mod_sbic=select01$predictors[which.min(select01$sbic)])
```

```
## [1] "wind_angle battery_current speed payload"
```

```
(Mod_sbc=select01$predictors[which.min(select01$sbc)])
```

```
## [1] "battery_current speed payload"
```

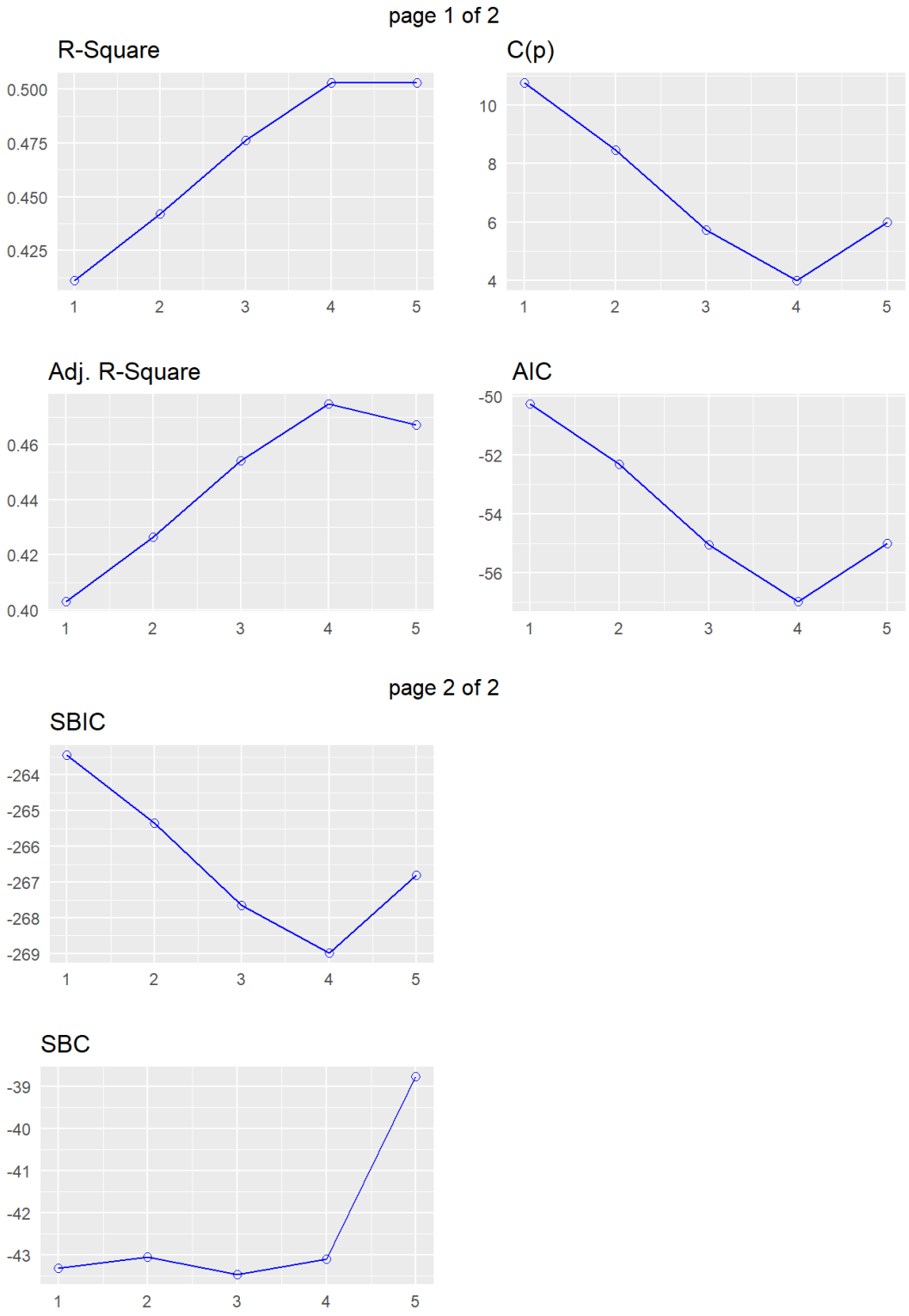
Extracting the best models under each of the five criteria (excluding $R^2$). One sees that those criteria land on the models that contain : “wind_angle battery_current speed payload”.

# 4-4-2.Best Subset

Select the subset of predictors that do the best at meeting some well-defined objective criterion, such as having the largest R2 value or the smallest MSE, Mallow’s Cp or AIC.

This shows the panel of fit criteria for best subset regression methods.

```
select02 = ols_step_best_subset(lnfull)
plot(select02)
```




```
(Mod_best=select02$predictors[4])
```

```
## [1] "wind_angle battery_current speed payload"
```

Using the Best subset under set of criteria, the best of the best models is “battery_current speed payload” again.

# 4-4-3. Automatic Model Selection

Build regression model from a set of candidate predictor variables by entering predictors based on p values, in a stepwise manner until there is no variable left to enter any more. The model should include all the candidate predictor variables. If details is set to TRUE, each step is displayed

```
# Forward
select03=ols_step_forward_p(lnfull)
(Mod_fwd=select03$predictors)
```

```
## [1] "speed"           "wind_angle"      "payload"         "battery_current"
```

Build regression model from a set of candidate predictor variables by removing predictors based on p values, in a stepwise manner until there is no variable left to remove any more. The model should include all the candidate predictor variables. If details is set to TRUE, each step is displayed.

```
# Backward
select04=ols_step_backward_p(lnfull)
(Mod_bwd=select04$indvar[!(select04$indvar %in% select04$removed)])
```

```
## [1] "wind_angle"      "battery_current" "speed"           "payload"
```

```
# Both
select05=ols_step_both_p(lnfull)
(Mod_both=select05$predictors)
```

```
## [1] "speed"      "wind_angle"
```

Using automatic model selection, Two out Three searches agree on model with regressors, “wind_angle battery_current speed payload”. Next, we try LASSO (Least Absolute Shrinkage and Selection Operator).

# 4-4-4. LASSO

Function *glmnet* in package **glmnet** will be used. There are parameters, $\lambda$ and $\alpha$, that are essential to the use of the function. We will see the key objective function that *glmnet* try to minimize.

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{N}\sum_{i=1}^{N} w_i l(y_i, \beta_0+\boldsymbol{\beta}^t \boldsymbol{x}_i) + \lambda\left[(1-\alpha)||\boldsymbol{\beta}||_2^2/2 + \alpha||\boldsymbol{\beta}||_1\right]$$

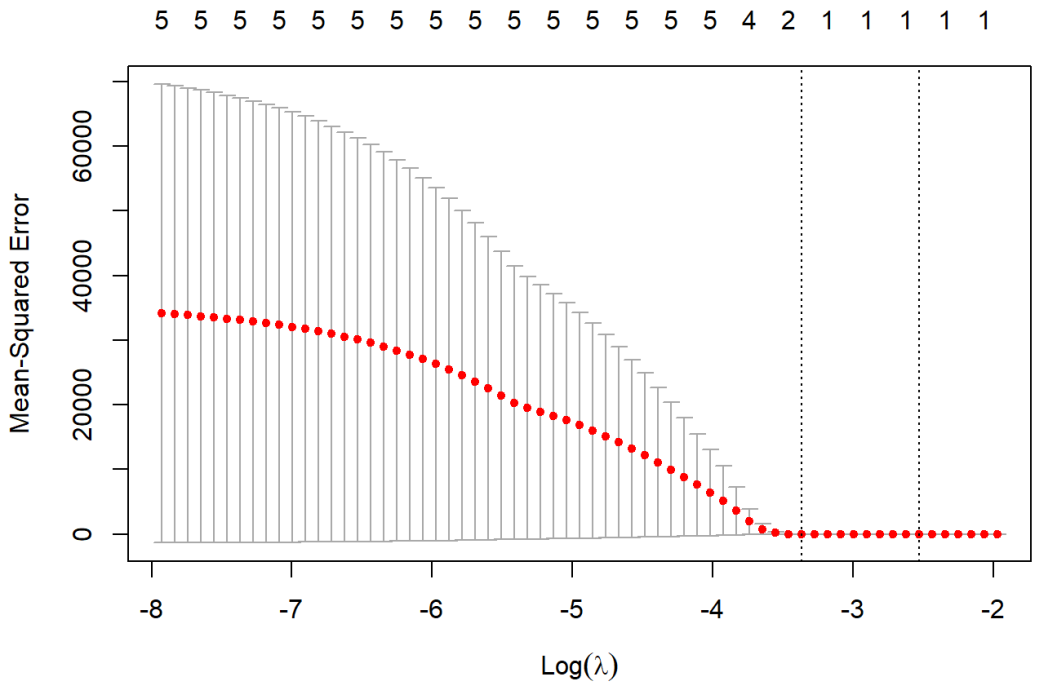
When $\alpha = 0$, this will lead the solution to be the Ridge regression estimate. When $\alpha = 1$, it will leads to the LASSO solution. So, the choice of $\alpha$ depends solely on the method attempted. The other parameter, $\lambda$, needs to be tuned on the other hand. Cross-Validation~(CV) will be used to select the optimal value of $\lambda$. CV usually runs through a sequence of candidate values on $\lambda$, which depends on the search space specified. We will simply use cv.glmnet to perform the task.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-3
```

```
set.seed(1)
cv.out=cv.glmnet(model.matrix(lnfull)[,-1],demo$time,alpha=1)
plot(cv.out)
```




```
(bestlam=cv.out$lambda.min)
```

```
## [1] 0.03441493
```

```
out=glmnet(model.matrix(lnfull)[,-1],demo$time,alpha=1,lambda=bestlam)
lasso.coef=coef(out)
lasso.coef[lasso.coef!=0]
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1]  5.467104e+00  4.415352e-09 -3.475504e-02
```

The final model found by LASSO is the full model, “battery_current speed payload”. So, after summarizing from all different selection schemes, we narrowed down to the final two models.

1. “battery_current speed payload”
2. “battery_current speed payload Wind_angle”

To further evaluate the superiority between these two models, cross-validation will be conducted in two different fashion:

1. Leave-one-out cross validation
2. $k$-fold cross validation

# 4-5. Leave-one-out cross validation

```
# Cross Validation - LOOCV
M01=paste("time~", paste(unlist(strsplit(Mod_cp,split=" ")), collapse = " + "))
M02=lnfull$call[[2]]
n=nrow(demo)
loocv_err=matrix(NA,n,2,dimnames=list(NULL, c("M01","M02")))
for (i in (1:n))
{
  fit01=lm(M01,data=demo[-i,])
  fit02=lm(M02,data=demo[-i,])
  loocv_err[i,]=(c(predict(fit01,demo[i,]),predict(fit02,demo[i,]))-demo$time[i])**2
}
colMeans(loocv_err)
```

```
##      M01      M02
## 25699.89 30621.98
```

```
which.min(colMeans(loocv_err))
```

```
## M01
##   1
```

```
print(M01)
```

```
## [1] "time~ wind_angle + battery_current + speed + payload"
```

# 4-6. $k$-fold cross validation

```
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
  }

k=20
set.seed(1)
folds=sample(1:k,nrow(demo),replace=TRUE)
cv.errors=matrix(NA,k,2, dimnames=list(NULL, paste(1:2)))
for(j in 1:k){
  for(i in 1:2){
    pred=predict(eval(parse(text=sprintf("lm(M0%s,data=demo[folds!=%s,])",i,j))),demo[folds==j,],id=i)
    cv.errors[j,i]=mean((demo$time[folds==j]-pred)^2)
    }
  }
mean.cv.errors=apply(cv.errors,2,mean,na.rm=TRUE)
mean.cv.errors
```

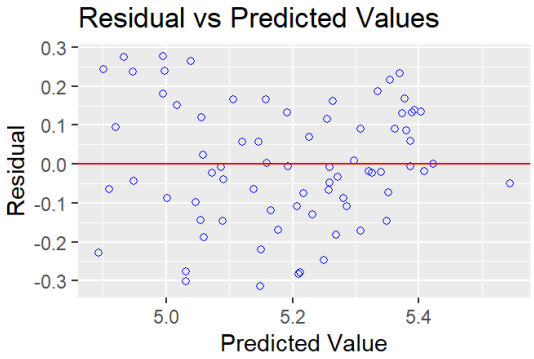
```
##        1        2
## 10581.11 13745.80
```

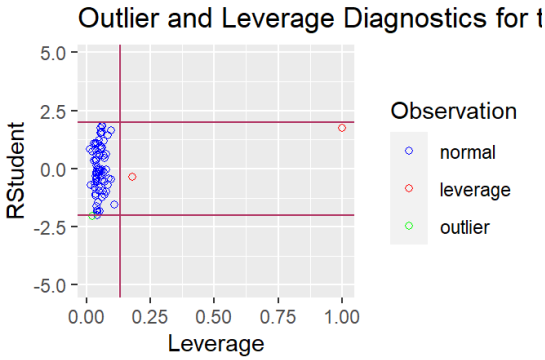
Based on those two scenarios, the best model contains “wind_angle battery_current speed payload”.

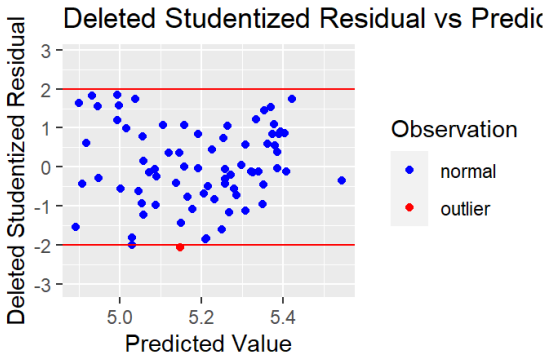
```
final=lm(M01,data=demo)
summary(final)
```

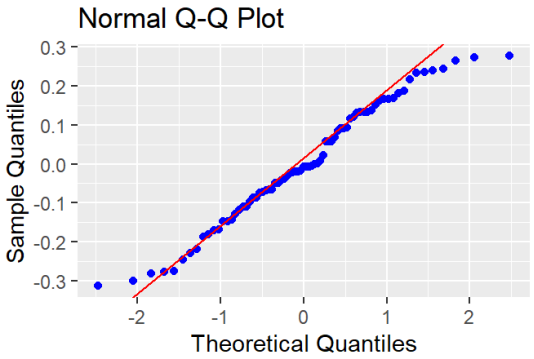
```
##
## Call:
## lm(formula = M01, data = demo)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.313812 -0.103334 -0.007793  0.131930  0.277065
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.955e+00  1.835e-01  32.445  < 2e-16 ***
## wind_angle       1.967e-07  1.012e-07   1.944  0.05597 .
## battery_current -2.477e-02  1.032e-02  -2.400  0.01906 *
## speed           -4.739e-02  6.113e-03  -7.753  5.2e-11 ***
## payload          3.190e-04  1.123e-04   2.841  0.00588 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1581 on 70 degrees of freedom
## Multiple R-squared:  0.503,  Adjusted R-squared:  0.4746
## F-statistic: 17.71 on 4 and 70 DF,  p-value: 4.376e-10
```

```
ols_plot_diagnostics(final, print_plot = TRUE)
```

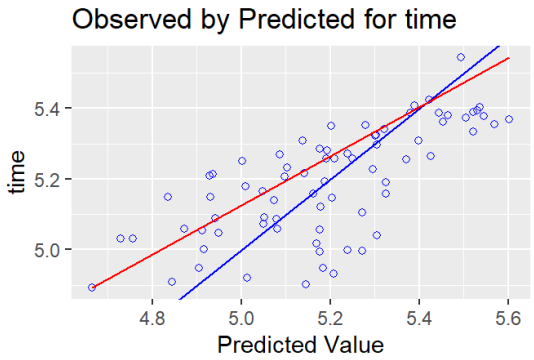

Residual vs Predicted Values
Residual
Predicted Value

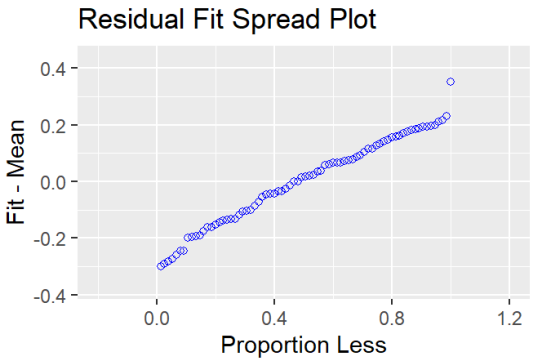

Outlier and Leverage Diagnostics for t
RStudent
Leverage
Observation
normal
leverage
outlier

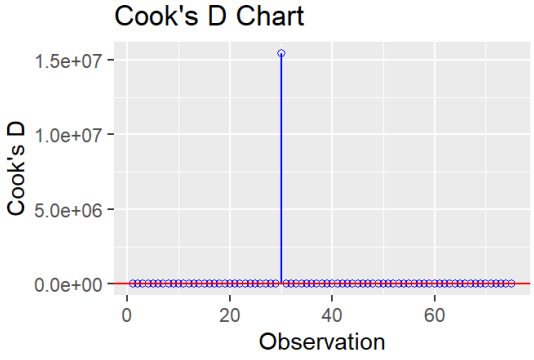

Deleted Studentized Residual vs Predic
Deleted Studentized Residual
Predicted Value
Observation
normal
outlier

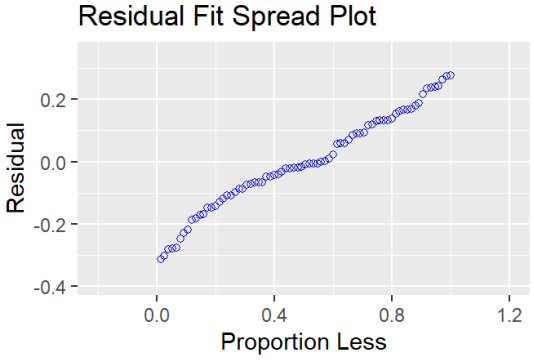

Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles



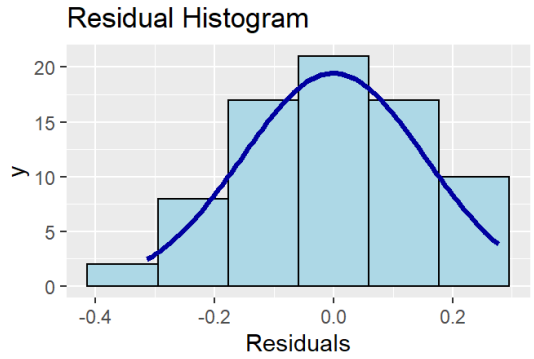

Observed by Predicted for time
time
Predicted Value

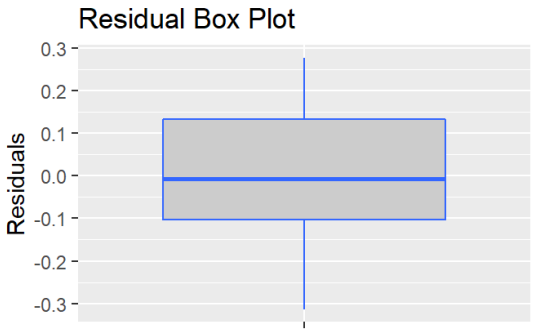

Residual Fit Spread Plot
Fit - Mean
Proportion Less


Cook's D Chart
Cook's D
Observation


Residual Fit Spread Plot
Residual
Proportion Less




Residual Histogram
y
Residuals




```
durbinWatsonTest(final,max.lag=1,alternative="positive")
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.2665602      1.459906   0.005
##  Alternative hypothesis: rho > 0
```

```
durbinWatsonTest(final,max.lag=1,alternative="negative")
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.2665602      1.459906   0.995
##  Alternative hypothesis: rho < 0
```

```
ols_test_normality(final)
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.9763         0.1726
## Kolmogorov-Smirnov         0.07          0.8306
## Cramer-von Mises         17.8805         0.0000
## Anderson-Darling          0.385          0.3845
## -----------------------------------------------
```

# 5. Summary

The final model is

$$\hat{\text{Time}} = 5.955 + (1.967e-07) \cdot \text{Wind\_Angle} + (-0.02477) \cdot \text{Battery\_Current} + (-0.04739) \cdot \text{Speed} + (0.000319) \cdot \text{Payload}.$$

The $R^2_{\text{Adj}}$ is 0.4746, and there is no serious violation on model assumptions.

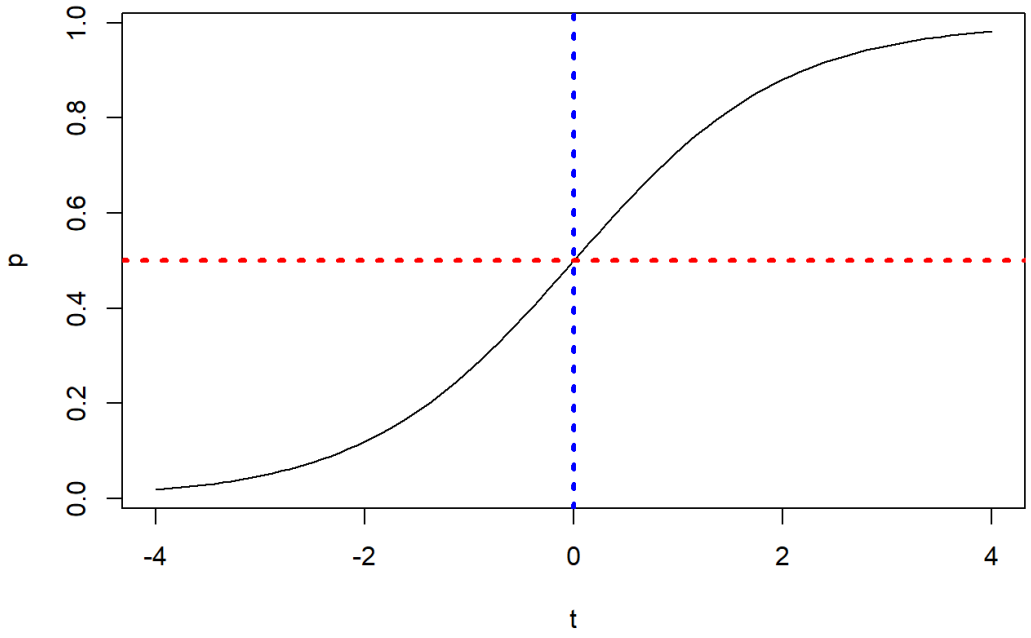
In order to have a long flying time for UAV, consider these information: 1) The wind should blow from the back of the UAV 2) Keep the Battery current as low as possible 3) Fly speed must not to be so high, it will lead to use more Battery Current 4) Fly with Low weight or no payload.

# 6. Logistic Regression

```
f1=function(x){
exp(x)/(1+exp(x))
}

curve(f1,-4,4, xlab = "t", ylab = "p")

abline(v=c(0),lwd = 3,lty=3,col='blue')
abline(h=0.5,lwd = 3,lty=3,col='red')
```




```
mydata = read.csv("UAV.csv")
```

```
mylogit <- glm(time_over_200 ~ speed, data = mydata, family = binomial(link="logit"))
summary(mylogit)
```

```
##
## Call:
## glm(formula = time_over_200 ~ speed, family = binomial(link = "logit"),
##     data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7167  -0.6893  -0.2057   0.7214   2.7808
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.7427     0.9712   3.854 0.000116 ***
## speed        -0.6323     0.1454  -4.348 1.38e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.477  on 74  degrees of freedom
## Residual deviance: 61.955  on 73  degrees of freedom
## AIC: 65.955
##
## Number of Fisher Scoring iterations: 5
```

```
#Store the data in matrix form
mydata <- as.matrix(mydata)
N <- nrow(mydata)
Y <- mydata[,10]
x0 <- matrix(1,N,1)
X <- cbind(x0,mydata[,7])

#refers to y_bar,keep Y==1 as success
BETA <- matrix(c(length(which(Y==1))/N,0),ncol=1);

DIFF <- 1;

while (DIFF > 0.0001) {
  # Define the p(x),calculating the vector of probabilities
  p <- 1/(exp(-X %*% BETA)+1)
  # Element wise multiplication
  V <- diag(c(p*(1-p)))
  # First derivative(Score Vector)
  First_deriv <- t(X)%*%(Y-p)
  # Second derivative(Hessian matrix)
  Second_deriv <- -t(X)%*%V%*%X
  # Multiplying Score and hessian matrix
  step <- -solve(Second_deriv)%*%First_deriv
  # Each time updated by step wise
  NewBETA <- BETA + step
  # Keep looping going
  DIFF <- max(abs(step))
  # Reset
  BETA <- NewBETA
}

print(BETA)
```

```
##            [,1]
## [1,]  3.7426637
## [2,] -0.6323381
```

```
mydata = read.csv("UAV.csv")
```

```
mylogit <- glm(time_over_200 ~ wind_speed+wind_angle+battery_voltage+battery_current+speed + payload, data = mydata, famil
y = binomial(link="logit"))
summary(mylogit)
```

```
##
## Call:
## glm(formula = time_over_200 ~ wind_speed + wind_angle + battery_voltage +
##     battery_current + speed + payload, family = binomial(link = "logit"),
##     data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7707  -0.5133  -0.1412   0.5021   2.9655
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      6.727e+00  1.326e+01   0.507 0.611877
## wind_speed       3.259e-01  4.803e-01   0.679 0.497403
## wind_angle       8.818e-06  1.709e-04   0.052 0.958863
## battery_voltage  1.658e-01  5.169e-01   0.321 0.748424
## battery_current -4.251e-01  2.867e-01  -1.483 0.138188
## speed           -9.035e-01  2.377e-01  -3.801 0.000144 ***
## payload          5.712e-03  3.179e-03   1.797 0.072335 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 95.477  on 74  degrees of freedom
## Residual deviance: 51.144  on 68  degrees of freedom
## AIC: 65.144
##
## Number of Fisher Scoring iterations: 12
```

```
#Store the data in matrix form
uav <- as.matrix(uav)
N <- nrow(uav)
Y <- uav[,10]
x0 <- matrix(1,N,1)
X <- cbind(x0,uav[,3:8])

#refers to y_bar,keep Y==1 as success
BETA <- matrix(c(length(which(Y==1))/N,0,0,0,0,0,0),ncol=1);

DIFF <- 1;

while (DIFF > 0.0001) {
  # Define the p(x),calculating the vector of probabilities
  p <- 1/(exp(-X %*% BETA)+1)
  # Element wise multiplication
  V <- diag(c(p*(1-p)))
  # First derivative(Score Vector)
  First_deriv <- t(X)%*%(Y-p)
  # Second derivative(Hessian matrix)
  Second_deriv <- -t(X)%*%V%*%X
  # Multiplying Score and hessian matrix
  step <- -solve(Second_deriv)%*%First_deriv
  # Each time updated by step wise
  NewBETA <- BETA + step
  # Keep looping going
  DIFF <- max(abs(step))
  # Reset
  BETA <- NewBETA
}

print(BETA)
```

```
##                         [,1]
##                  6.726813e+00
## wind_speed       3.259156e-01
## wind_angle       6.905822e-06
## battery_voltage  1.657646e-01
## battery_current -4.250904e-01
## speed           -9.035364e-01
## payload          5.712343e-03
```

**Reference**

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, 2013, *Introduction to Statistical Learning,* Springer, New York

Wu, P. S (2022) Math338-SLR02.pdf

Wu, P. S (2022) Math312Sp20_Multicollinearity.pdf

Wu, P. S (2022) Math338_GLM.pdf

Wu, P. S (2022) Math338SP22_LogisticReg.pdf

Wu, P. S (2022) Math338SP22-VarTransformation.pdf

Wu, P. S (2022) Math338SP2022_Regression LSE.pdf

Wu, P. S (2022) RegressionFlow.pdf

Wu, P. S (2022) Lab04-MRNoCollin.html

Wu, P. S (2022) LogReg.pdf