Computational Journal of Mathematical and Statistical Sciences
2(1), 52–79
DOI: 10.21608/cjmss.2023.189834.1002
https://cjmss.journals.ekb.eg/

Research article

A Simple Introduction to Regression Modeling using R

Amr R. Kamel [1, 2], Mohamed R. Abonazel [2 *]

[1]Department of Basic Sciences, Elgazeera High Institute for Computers and Information Systems, Ministry of Higher Education, Cairo, Egypt; amr_ragab@pg.cu.edu.eg.

[2]Department of Applied Statistics and Econometrics, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza, Egypt; mabonazel@cu.edu.eg.

***Correspondence:** mabonazel@cu.edu.eg

Abstract: In statistical modeling, regression analysis is a group of statistical processes used in R programming and statistics to determine the relationship between dataset variables. It is a solid technique for determining the factors that affect an issue of interest. You can confidently establish which elements are most important, which ones can be ignored, and how these factors interact when you do a regression. It can be used to simulate the long-term link between variables and gauge how strongly the relationships between them are related. Regression analysis is typically used to ascertain the relationship between the dataset's dependent and independent variables. Generally, regression analysis is used to determine the relationship between the dependent and independent variables of the dataset. Understanding how dependent variables change when one of the independent variables changes while the other independent variables remain constant is made easier with the use of regression analysis. As a result, it is easier to create a regression model and forecast values in response to changes in one of the independent variables. Based on the categories of dependent variables, the quantity of independent variables, and the contour of the regression line. In this paper, we use the R programming language to present various empirical investigations in statistics and econometrics. We next consider problems involving modeling the relationship between response and explanatory variables for linear and non-liner regression models.

Keywords: Diagnostics; Estimation Method; Graphical Presentation Methods; Model Visualization; Non-linear Regression; Packages; Practical Introduction; Regression Analysis; R Software.

Mathematics Subject Classification: 62J05; 62G08; 62M10.

Received: 26 January 2023; **Revised:** 17 February 2023; **Accepted:** 18 February 2023; **Published:** 20 February 2023.

1. Introduction

Regression analysis is a powerful tool for uncovering the associations between variables observed in data, but cannot easily indicate causation. It has several applications in the fields of business, finance, and economics. It is used for employed to calculate the relationships between a

dependent variable (commonly referred to as the "outcome" or "response" variable) and one or more independent variables (often referred to as "predictors," "covariates," "explanatory variables," or "features"). In linear regression, the most typical type of regression analysis, the line (or a more complicated linear combination) that most closely matches the data in terms of a given mathematical criterion is found. The foundation of many significant statistical models is regression. The outcome we care about in marketing applications is typically the dependent variable (e.g., sales), and the instruments we use to get there are the independent variables (e.g., pricing or advertising). Few other techniques can offer the insights that regression analysis can. The two theoretically separate uses of regression analysis are as follows:

1. Regression analysis is frequently used for forecasting and prediction, where it has a lot in common with machine learning.
2. Regression analysis can be used to infer causal links between the independent and dependent variables in specific circumstances.

Importantly, regressions by themselves only reveal relationships between a dependent variable and a collection of independent variables in a fixed dataset. To use regressions for prediction or to infer causal relationships, respectively, a researcher must carefully justify why existing relationships have predictive power for a new context or why a relationship between two variables has a causal interpretation. The latter is especially important when researchers hope to estimate causal relationships using observational data. Knowing about the effects of independent variables on dependent variables can help market researchers in many different ways. For example, it can help direct spending if we know promotional activities significantly increases sales. Knowing about the relative strength of effects is useful for marketers because it may help answer questions such as whether sales depend more on price or on promotions, see [1].

Regression analysis also allows us to compare the effects of variables measured on different scales such as the effect of price changes (e.g., measured in $) and the number of promotional activities. Regression analysis can also help to make predictions. For example, if we have estimated a regression model using data on sales, prices, and promotional activities, the results from this regression analysis could provide a precise answer to what would happen to sales if prices were to increase by 5% and promotional activities were to increase by 10%. Such precise answers can help (marketing) managers make sound decisions. Furthermore, by providing various scenarios, such as calculating the sales effects of price increases of $5\%, 10\%$, and 15%, managers can evaluate marketing plans and create marketing strategies.

Regression measures whether or not correlations between variables in a data set are statistically significant by capturing their magnitude. Although there are non-linear regression techniques for more complex data and analysis, simple linear regression and multiple linear regression are the two fundamental types of regression. While multiple linear regression employs two or more independent factors to predict the outcome, simple linear regression just uses one independent variable to explain or predict the outcome of the dependent variable y (while holding all others constant).

Professionals in other industries, including banking and investment, might benefit from regression. Regression can also aid in predicting sales for a business based on external factors like the weather, past sales, gross domestic product (GDP) growth, and other variables, see [2,3]. A popular regression model in finance for valuing assets and calculating capital expenses is the capital asset pricing model (CAPM). In addition, Econometrics has occasionally come under fire for depending excessively on the interpretation of regression output without connecting it to economic theory or looking for causal mechanisms. Even if that means creating your own explanation of the underlying processes, it is essential that the findings presented in the data can be effectively explained by a theory.

In this paper, we will review the basics R Programming and we provide R-codes for linear and non-liner regression models with estimation. Also, we will investigate some diagnostic methods for problems of regression analysis. This paper is organized as follows. Section 2 provides an introduction to the R programming language. Section 3 presents regression modeling with examples. In Section 4, the multiple linear regression and estimation have been discussed. While in Section 5, the non-linear regression model will be introduced. Finally, Section 6 offers the concluding remarks.

2. Introduction to R Programming

2.1 R Overview and History

R is a widely used open-source programming language for statistical computing and data analysis. R typically includes a command-line interface. R is publicly accessible under the GNU General Public License, and binary versions that have already been pre-compiled for other operating systems including Linux, Windows, and Mac are also available. The newest cutting-edge technology is the R programming language. The environment for dealing with your data in R is comprehensive. Without building a whole program, you can just use the functions that are built into the environment to process your dataset. Additionally, you can create your own programs to carry out tasks that lack built-in functionalities or to repeatedly complete the same action, for example. R is simpler to code in and understand since it shares many syntactical similarities with other widely used languages. Programs can be written in R software in any of the widely used IDE like R Studio, Rattle, Tinn-R, etc, see [4,5].

Ross Ihaka and Robert Gentleman created R for the first time in 1992 at the University of Auckland in New Zealand. The S language, which was created (mostly) by John Chambers at Bell Laboratories, is a "dialect" of which the R language is a subset. The R Development Core Team, which has more than a dozen members, is presently responsible for maintaining this software. R has received a lot of additional coding contributions since it was first published. R is open source, which means that users may change, copy, and distribute the software or any derivatives as long as the modified source code is made public.

2.2 Finding and installing R

The Comprehensive R Archive Network, or CRAN, is a network of computers that is maintained by the R Development Core Team and houses installation files and documentation for R. You can find it by searching for CRAN R on Google or visiting http://cran.r-project.org/. Windows, Mac, and Unix-like operating systems all support R. By clicking one of the download links at the top, users can get installation files and instructions from the CRAN website. While the R commands vary between systems (if they are available at all), the graphical user interfaces (GUIs) and their menus do not. An interpreted computer language called R, developed by the Development Core Team, supports branching and looping as well as modular programming with the use of functions. For increased efficiency, R enables integration with processes created in C, C++,.Net, Python, or FORTRAN.

2.3 Features of R

As stated earlier, R is a computer language and software environment used for statistical analysis, graphic representation, and reporting. The following are the important features of R:

- R is a well-designed, easy-to-use programming language with input and output capabilities, conditionals, loops, and user-defined recursive functions.
- For calculations on arrays, lists, vectors, and matrices, R offers a number of operators.
- R has a reliable system in place for processing and storing data.
- R offers graphical tools for data analysis and display that may be used on a computer or printed on paper.
- R offers a sizable, well-organized, and comprehensive library of tools for data analysis.

As a conclusion, R is the most popular statistical programming language in the world. Data scientists rank it as their top option, and a robust and brilliant community of contributors backs it up. R is used in mission-critical corporate applications and is taught in universities. You will learn R programming in this lesson using appropriate examples and simple, straightforward techniques. The software is updated frequently, but the changes are typically not substantial, see [6,7,8].

3. Regression Modeling

Linear regression is an approach to model the relationship between a scalar dependent variable y and one or more explanatory variables (independent variables) denoted by x_i , $i = 1,2,\ldots,n$. The steps of regression analysis are as follows:

1. Select the model's goal and the suitable dependent variable to achieve it.
2. Select independent factors.
3. Calculate the regression equation's parameters.
4. Interpret parameters that have been estimated, the goodness of fit, and both qualitative and quantitative evaluations of the parameters.
5. Evaluate whether assumptions are reasonable.
6. Modify and amend the calculated equation if some assumptions are not met.

7. Validate the regression equation that was estimated.

With the understanding that the goal of the model has already been determined and that only the last phases remain, we will look at these procedures.

3.1 Simple Linear Regression

We can better comprehend the relationships between the values of a quantitative explanatory (or predictor) variable and the values of a quantitative outcome (or response) by using linear regression. This method is frequently used to either produce anticipated values or draw conclusions about relationships in the dataset. In certain areas, the predictor is referred to as the independent variable and the outcome as the dependent. Since dependent and independent have so many different connotations in statistics, we refrain from using them in this way. A simple linear regression model for an outcome y as a function of a predictor x takes the form:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \text{for } i = 1, \dots, n, \tag{1}$$

where n represents the number of observations (rows) in the data set. For this model, β_0 is the population parameter corresponding to the intercept (i.e., the predicted value when $= 0$) and β_1 is the true (population) slope coefficient (i.e., the predicted increase in y for a unit increase in x). The errors are represented by the ϵ_i's (which are thought to be random noise with mean 0).

We estimate the population parameters β_0 and β_1 using information from our sample because we hardly ever know their exact values. The "best" coefficients β_0 and β_1 are found using the *lm()* function (in the *MASS* package) when the fitted values (or predicted values) are given by the following formula $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ [9]. What is left over is captured by the residuals$(\epsilon_i = y_i - \hat{y}_i)$. The model almost never fits perfectly - if it did there would be no need for a model.

Typically, an ordinary least square (OLS) criterion that minimizes the sum of the squared residuals is used to identify the regression line that fits the data the best. There is only one least squares regression line, which is determined by the values of $\hat{\beta}_0$ and $\hat{\beta}_1$. The OLS method is one of the oldest estimation methods and is common used in most applications, because it given best linear unbiased estimators (BLUEs). The formula of OLS estimator of the model in Equation (1) is:

$$\hat{\beta}_{\text{OLS}} = (x^T x)^{-1}(x^T y). \tag{2}$$

When the errors have finite variances, the OLS method offers minimum-variance mean-unbiased estimation. OLS is the maximum likelihood estimator when the extra supposition that the errors are normally distributed is held.

3.2 Inspirational Example

In our example, for ninety days the Pioneer Valley Planning Commission (PVPC) in Florence, Massachusetts, gathered information north of Chestnut Street. When a rail-trail user passed the data collecting point, a laser sensor put up by data collectors recorded the event (in *mosaicData* package).

On the other hand, Monte Carlo simulation (MCS) techniques were employed to produce data that was identical to data gathered from real phenomena and complied with our model's parameters. The most common application of MCS approaches is to empirically investigate the characteristics of theoretical models because of their special suitability for doing so, for more details about MCS using R, see [10,11,12].

The PVPC is trying to figure out how daily ridership the number of people who utilize the bike path each day relates to a range of explanatory factors like temperature, precipitation, cloud cover, and day of the week. In order to obtain summary statistics for each variable in the data frame we will be used;

```
##=== Prepare the R console
rm(list = ls(all = TRUE)) # Remove all objects in R console
set.seed(09061982) # Set the seed for reproducible results
########################## Installing the packages ########################

## install.packages("Name")
PackageNames <- c("stats4",
                  "Metrics",
                  "graphics",
                  "ggplot2",
                  " car",
                  "MASS",
                  "MVTests",
                  " mosaic",
                  " mosaicData",
                    "vtable"
                    )
for(i in PackageNames){
  if(!require(i, character.only = T)){
    install.packages(i, dependencies = T)
    require(i, character.only = T)
  }
}

glimpse(RailTrail)    ## Load data. View structure
Rows: 90
Columns: 11
$ hightemp   <int> 83, 73, 74, 95, 44, 69, 66, 66, 80, 79, 78, 65, 41, ~
$ lowtemp    <int> 50, 49, 52, 61, 52, 54, 39, 38, 55, 45, 55, 48, 49, ~
$ avgtemp    <dbl> 66.5, 61.0, 63.0, 78.0, 48.0, 61.5, 52.5, 52.0, 67.5~
$ spring     <int> 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1~
$ summer     <int> 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0~
$ fall       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0~
$ cloudcover <dbl> 7.6, 6.3, 7.5, 2.6, 10.0, 6.6, 2.4, 0.0, 3.8, 4.1, 8~
$ precip     <dbl> 0.00, 0.29, 0.32, 0.00, 0.14, 0.02, 0.00, 0.00, 0.00~
$ volume     <int> 501, 419, 397, 385, 200, 375, 417, 629, 533, 547, 43~
$ weekday    <lgl> TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE, FA~
$ dayType    <chr> "weekday", "weekday", "weekday", "weekend", "weekday~
```

Table 1, presents some descriptive statistics for the variables (dependent and independent variables). Moreover, when the dataset are contaminated with a single or few outliers, the problem of identifying such observations is a serious problem. We note that in most cases datasets contain more outliers or a group of influential observations.

```
### Descriptive Statistics for RailTrail Dataset ###

library("vtable")
st(RailTrail)  # in table or using;
summary(RailTrail)
```

Table1: Descriptive statistics for rail-trail dataset.

sumtable {vtable} Summary Statistics

Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
hightemp	90	68.833	13.023	41	59.25	77.75	97
lowtemp	90	46.033	11.845	19	38	53.75	72
avgtemp	90	57.433	11.33	33	48.625	64.5	84
spring	90	0.589	0.495	0	0	1	1
summer	90	0.278	0.45	0	0	1	1
fall	90	0.133	0.342	0	0	0	1
cloudcover	90	5.807	3.226	0	3.65	8.475	10
precip	90	0.093	0.263	0	0	0.02	1.49
volume	90	375.4	127.463	129	291.5	451.25	736
weekday	90						
... No	28	31.1%					
... Yes	62	68.9%					
dayType	90						
... weekday	62	68.9%					
... weekend	28	31.1%					

Moreover, the process of identifying or diagnosing outliers in regression analysis is crucial; hence some techniques for doing so will be demonstrated. These techniques involve statistics that concentrate on observations that have an impact on the OLS estimate. The OLS estimator is extremely vulnerable to outliers value, hence the model needs a robust estimator that is unaffected by outliers in the dataset in order to produce an accurate estimation. In various regression models, many papers discuss a variety of robust estimators; see e.g. [13-18].

On the other hand, the OLS estimator is best in the class of linear unbiased estimators when the errors are homoscedastic and serially uncorrelated and consistent when the regressors are exogenous and there is no multicollinearity. When the explanatory factors are highly correlated, this issue

occurs. The correlation matrix and variance inflation factor (VIF), it is used to diagnostics of the multicollinearity problem, see Figure 1. The distinct effects of each of the explanatory variables on the response variable are then challenging to separate. As a result, the computed regression parameters could have unexpectedly divergent signs or be statistically insignificant. Thus, it would be challenging for the researcher to draw a significant statistical inference. For information on how to handle and resolve this issue in other regression models; see e.g. [19-22]. On the other hand, Figure 2 displays the histogram and boxplot for dependent variable in rail-trail dataset.

```
### Correlation Analysis ###

library("corrplot")
RailTrail_data <- RailTrail[,-c(10,11)]
## Correlation Matrix between all variables
Correlation_Matrix <- cor(RailTrail_data)
print(Correlation_Matrix)
##             hightemp    lowtemp    avgtemp      spring     summer        fall  cloudcover
## hightemp    1.00000000  0.6598839  0.9196439 -0.33333833  0.6669179 -0.39625396 -0.09557041
## lowtemp     0.65988392  1.0000000  0.9019603 -0.38873326  0.7374661 -0.40902843  0.36599773
## avgtemp     0.91964390  0.9019603  1.0000000 -0.39477095  0.7687716 -0.44153789  0.13638819
## spring     -0.33333833 -0.3887333 -0.3947710  1.00000000 -0.7422503 -0.46944033 -0.10242904
## summer      0.66691793  0.7374661  0.7687716 -0.74225033  1.0000000 -0.24325213  0.17035425
## fall       -0.39625396 -0.4090284 -0.4415379 -0.46944033 -0.2432521  1.00000000 -0.07620144
## cloudcover -0.09557041  0.3659977  0.1363882 -0.10242904  0.1703542 -0.07620144  1.00000000
## precip      0.13431718  0.3737956  0.2725832 -0.24646475  0.3409780 -0.09253473  0.36914883
## volume      0.58257188  0.1760858  0.4268535 -0.03531086  0.2274170 -0.24853781 -0.37456168
##              precip      volume
## hightemp    0.13431718  0.58257188
## lowtemp     0.37379561  0.17608580
## avgtemp     0.27258316  0.42685354
## spring     -0.24646475 -0.03531086
## summer      0.34097796  0.22741700
## fall       -0.09253473 -0.24853781
## cloudcover  0.36914883 -0.37456168
## precip      1.00000000 -0.23238396
## volume     -0.23238396  1.00000000
### plot Correlation Matrix
corrplot.mixed(Correlation_Matrix, bg = "black")

hist(volume) # Histogram for dependent variable

boxplot(volume) # Boxplot for dependent variable
```

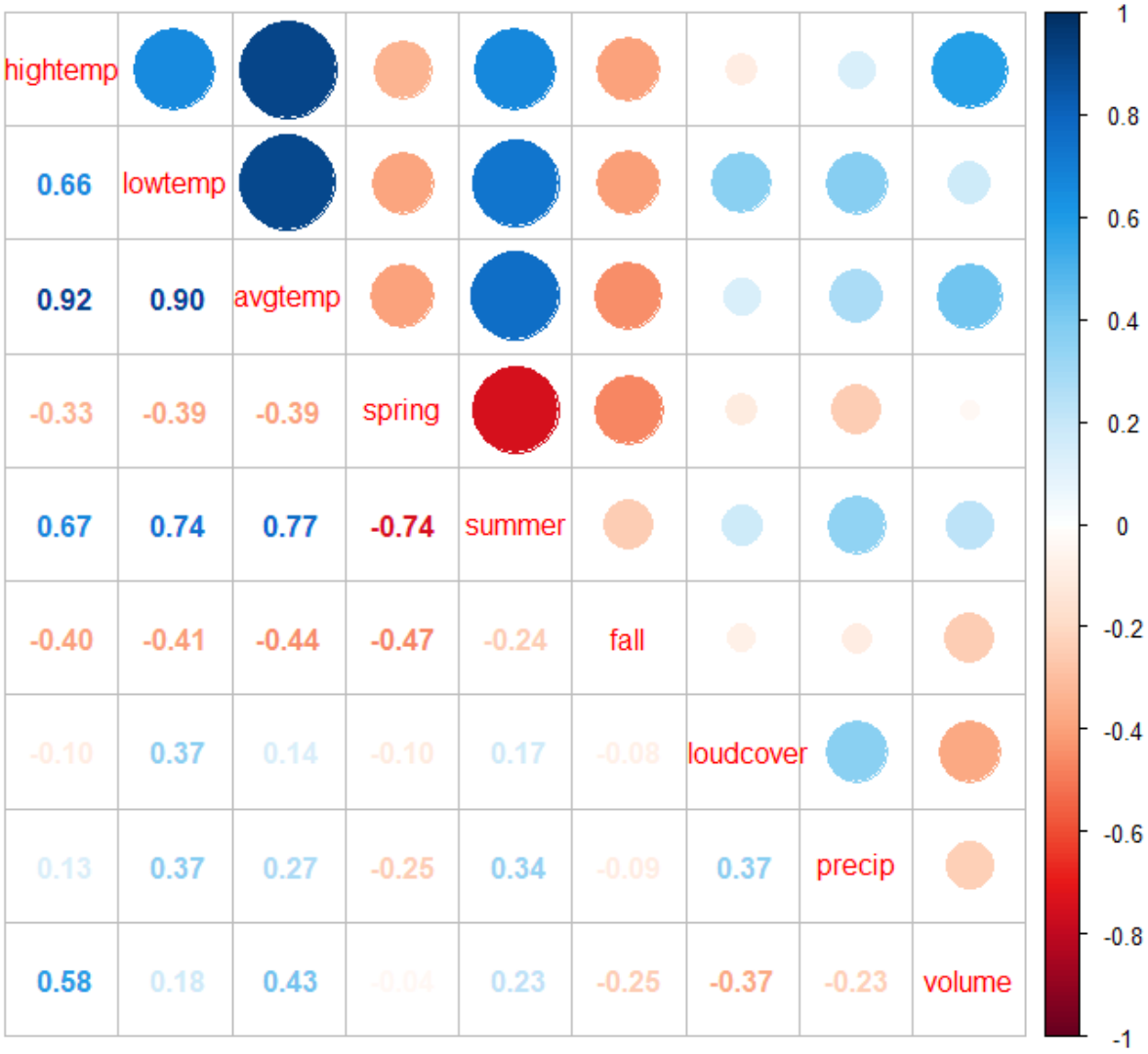

Figure 1: Correlation matrix for rail-trail dataset.

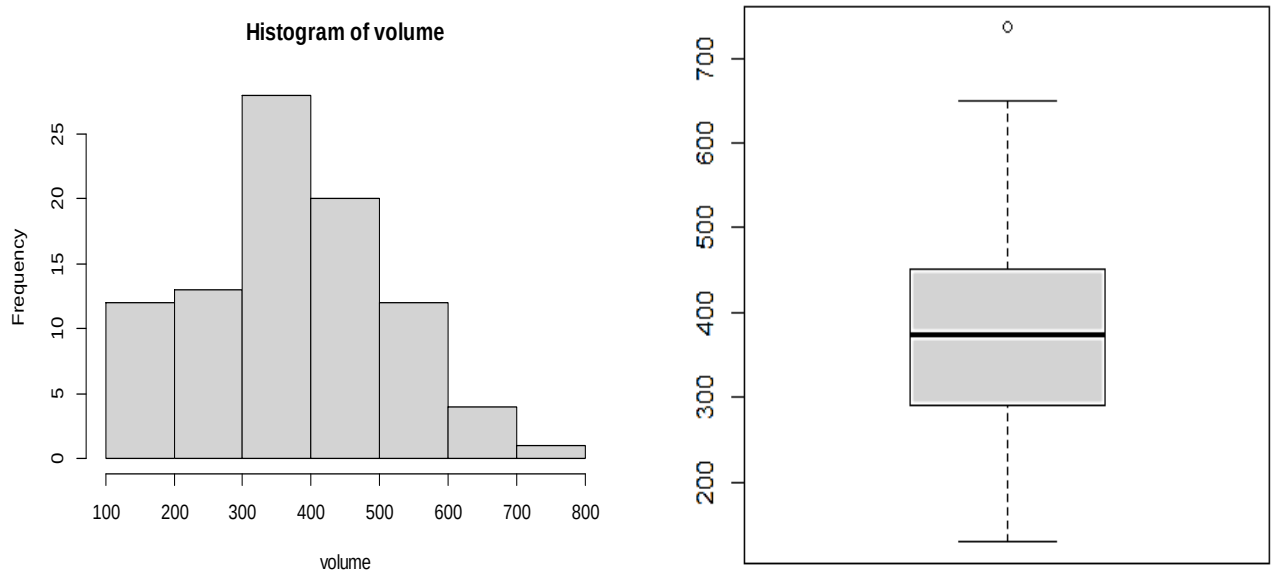

Figure 2: Histogram and Boxplot for dependent variable in rail-trail dataset.

The main regression function in R used for modelling linear regression is *lm()*. R also includes with a wealth of tools for modelling that is more complicated, including *gam()* for generalized additive models and *glm()* for generalized linear models, se [23,24]. By applying the following formula to the *lm(dependent variable ~ Independent variable)*, the fitted coefficient are displayed below;

```
### Simple Linear Regression ###

library("MASS")
simple_Model <- lm(volume ~ hightemp, data = RailTrail)
print(simple_Model)

## Call:
## lm(formula = volume ~ hightemp, data = RailTrail)
##
## Coefficients:
## (Intercept)     hightemp
##    -17.079        5.702
```

The estimated parameters coefficients are extracted as follows;

```
coeffs <- coefficients(simple_Model); coeffs

## (Intercept)     hightemp
##    -17.079        5.702
```

The following plot, Figure 3 displays a scatterplot of ridership (volume) versus high temperature (hightemp), with a simple linear regression line superimposed.

```
library("mosaic")
plotModel(simple_Model, system = "ggplot2")
```

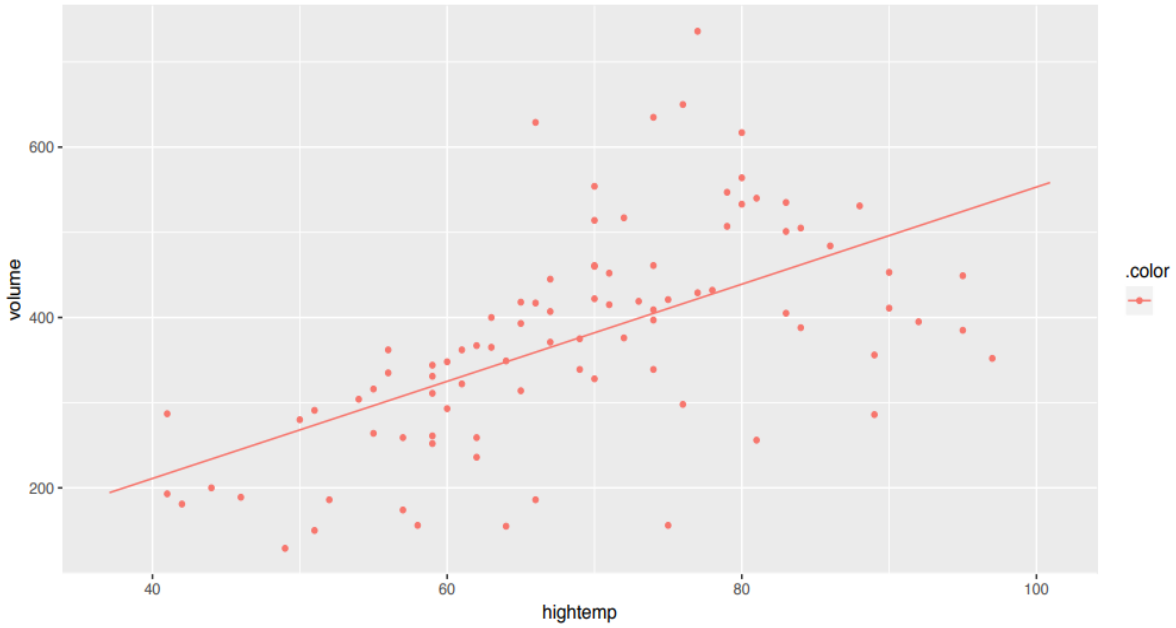

Figure 3: Scatterplot for simple linear regression.

3.3 Measuring the Strength of Fit

To get more information about the fitted model, *summary* (the model's name) can be used to get details about Residuals, Coefficients, Residual standard error, R^2 , Adjusted R^2 . Moreover, the function *summary.anova* (the model's name) used to get ANOVA table.

```
## To get more information about the fitted model;
summary(simple_Model) # same as summary.lm(simple_Model)
## Call:
## lm(formula = volume ~ hightemp, data = RailTrail)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
##  -254.562  -57.800    8.737   57.352  314.035
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.079     59.395  -0.288    0.774
## hightemp       5.702      0.848   6.724 1.71e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 104.2 on 88 degrees of freedom
## Multiple R-squared:  0.3394,    Adjusted R-squared:  0.3319
## F-statistic: 45.21 on 1 and 88 DF,  p-value: 1.705e-09
```

```
summary.aov(simple_Model) # ANOVA Table

##             Df Sum Sq Mean Sq F value   Pr(>F)
## hightemp     1 490744  490744   45.21 1.71e-09 ***
## Residuals   88 955214   10855
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The definition of R^2 is given by:

$$\begin{aligned} R^2 &= 1 - \frac{SSE}{SST} = \frac{SSM}{SST} \\ &= 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \\ &= 1 - \frac{SSE}{(n-1)\text{Var}\,(y)}, \end{aligned}$$

where SST is the total sum of squares, SSM is the sum of squares ascribed to the model, and SSE is the sum of squared residuals. For the rail trail dataset, let's determine these values;

```
n <- nrow(RailTrail)
SST <- var(~volume, data = RailTrail) * (n - 1)
SSE <- var(residuals(simple_Model)) * (n - 1)
1 - SSE / SST
## [1] 0.33939

rsquared(simple_Model)  # using function rsquared
## [1] 0.33939
```

On the other hand, the R^2 of the regression model on the right is 0.3394. We say that the regression model based on average daily temperature explained about 33% of the variation in daily ridership.

3.4 Confidence interval and predicted confidence interval

To estimate the confidence interval for our model; we run the below function;

```
newRailTrail = data.frame(hightemp=4.5)
predict(simple_Model, newRailTrail, interval = 'confidence')
##        fit       lwr      upr
##   1 8.579171 -102.0126 119.1709
```

For estimating a predicted confidence interval for the estimated model, we can use predict instead of confidence.

```
predict(simple_Model, newRailTrail, interval = 'predict')
##       fit       lwr      upr
## 1 8.579171 -226.1531 243.3115
```

3.5 Diagnostics: Assessing the Regression Model Fit

An important part of any statistical analysis is assessment of how well the chosen model fits the data. In regression, estimation of the linear slope ($\hat{\beta}$) is not sufficient to understand whether a linear model is appropriate. Six aspects of the model should be assessed;

(i) Independence;
(ii) Normality;
(iii) Linearity;
(iv) Constant variance;
(v) Presence of outliers; and
(vi) Need for additional predictor variables.

For diagnosing the model, the graphical plots can be used in Figures 4, 5 and 6. Diagnostic plots provide checks for heteroscedasticity, normality, influential observations, and post fit model examination.

```
                    ## For regression diagnostics ##
Aov <- aov (volume ~ hightemp)
## Call:
##    aov(formula = volume ~ hightemp)
## Terms:
##
##                 hightemp Residuals
## Sum of Squares  490743.5  955214.1
## Deg. of Freedom        1        88
## Residual standard error: 104.1859
## Estimated effects may be unbalanced
```

```
residuals <-residuals(Aov)       # to find residual values
volume.hat<- fitted.values(Aov) # to find fitted values of dependent var.
par(mfrow = c(2,2))             # plot Normality of Residuals
plot(simple_Model)
```

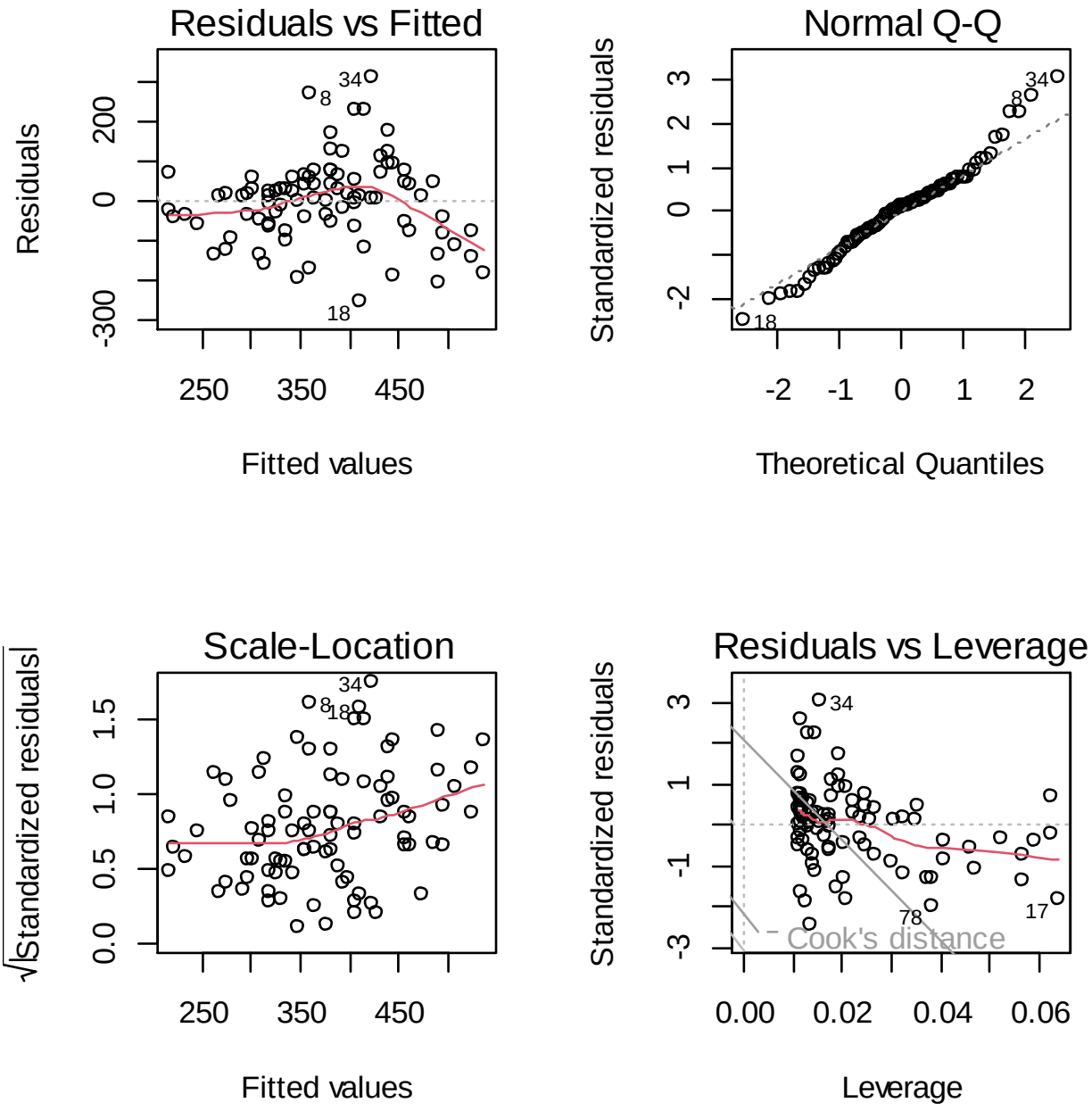

Figure 4: Residuals of OLS estimation results.

```
library("car")
library("lmtest")
residualPlots(simple_Model)            # residual plots

##              Test stat   Pr(>|Test stat|)
## hightemp      -3.2581          0.001601 **
## Tukey test    -3.2581          0.001122 **
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

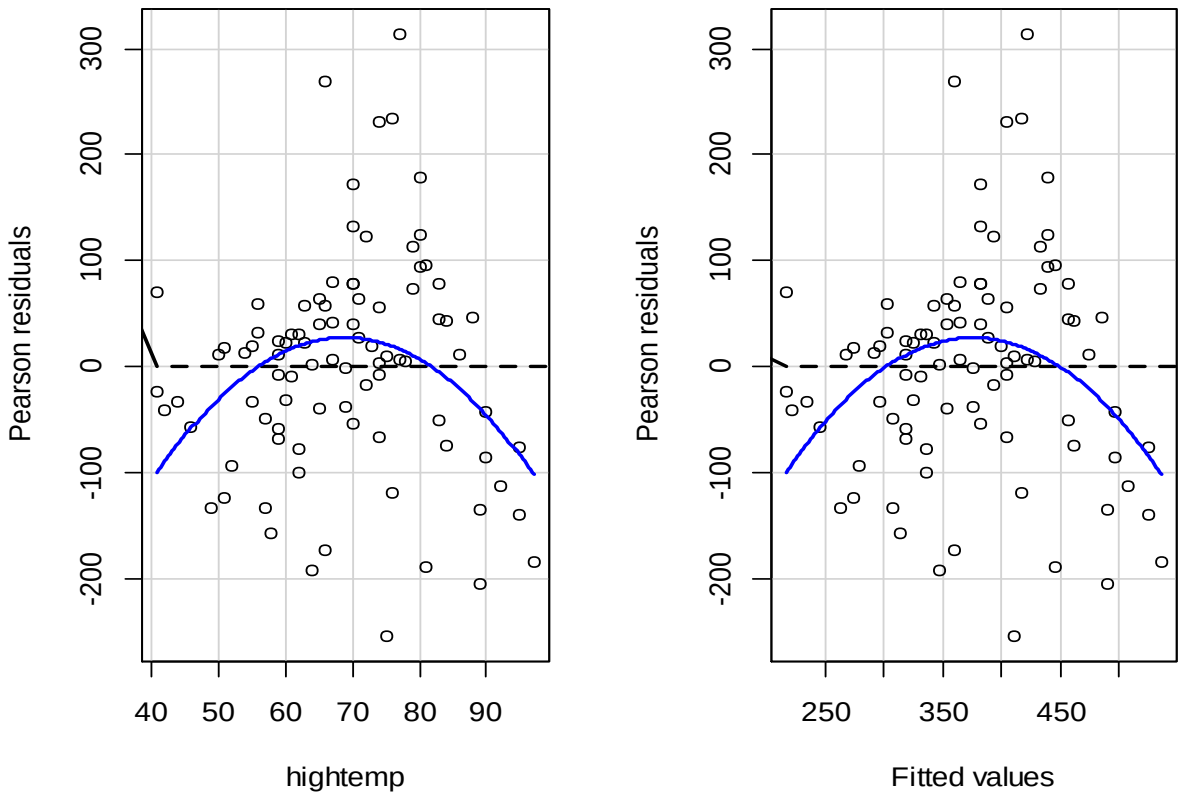

Figure 5: Residuals and fitted values for simple linear regression.

```
library("lmtest")
ncvTest(simple_Model)              # Testing for heteroskedasticity

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 6.425507, Df = 1, p = 0.011249

outlierTest(simple_Model)          # Outliers-Bonferonni test

## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##    rstudent unadjusted p-value Bonferroni p
## 34 3.192585          0.0019631      0.17668

dwtest(simple_Model)              # Durban Watson test for autocorrelation

## Durbin-Watson test
##
## data:  simple_Model
## DW = 1.8916, p-value = 0.3011
## alternative hypothesis: true autocorrelation is greater than 0.

# Test Shapiro Wilk Test of residuals (Null: distribution is normal)
shapiro.test(simple_Model$residuals)

## Shapiro-Wilk normality test
## data:  simple_Model$residuals
## W = 0.97838, p-value = 0.139
```

```
library("lmtest")
ncvTest(simple_Model)              # Testing for heteroscedasticity

## Post Fit Model Examination
volume.Hat <- predict(simple_Model, interval="prediction")
temp_df <- cbind(newRailTrail, volume.Hat)
gg <- ggplot(temp_df, aes(x=volume, y=hightemp))
gg <- gg + geom_line(aes(y=lwr), color = "red", linetype = "dashed")
gg <- gg + geom_line(aes(y=upr), color = "red", linetype = "dashed")
gg <- gg + geom_smooth(method=lm, level=.95, se=TRUE)
gg <- gg + geom_point()
gg <- gg + xlab("volume") + ylab("hightemp")
gg <- gg + ggtitle("Simple Linear Fit w 95% CI's of Mean and Individual
Predictions")
plot_CIboth <- gg + theme_bw()
print(plot_CIboth)
```

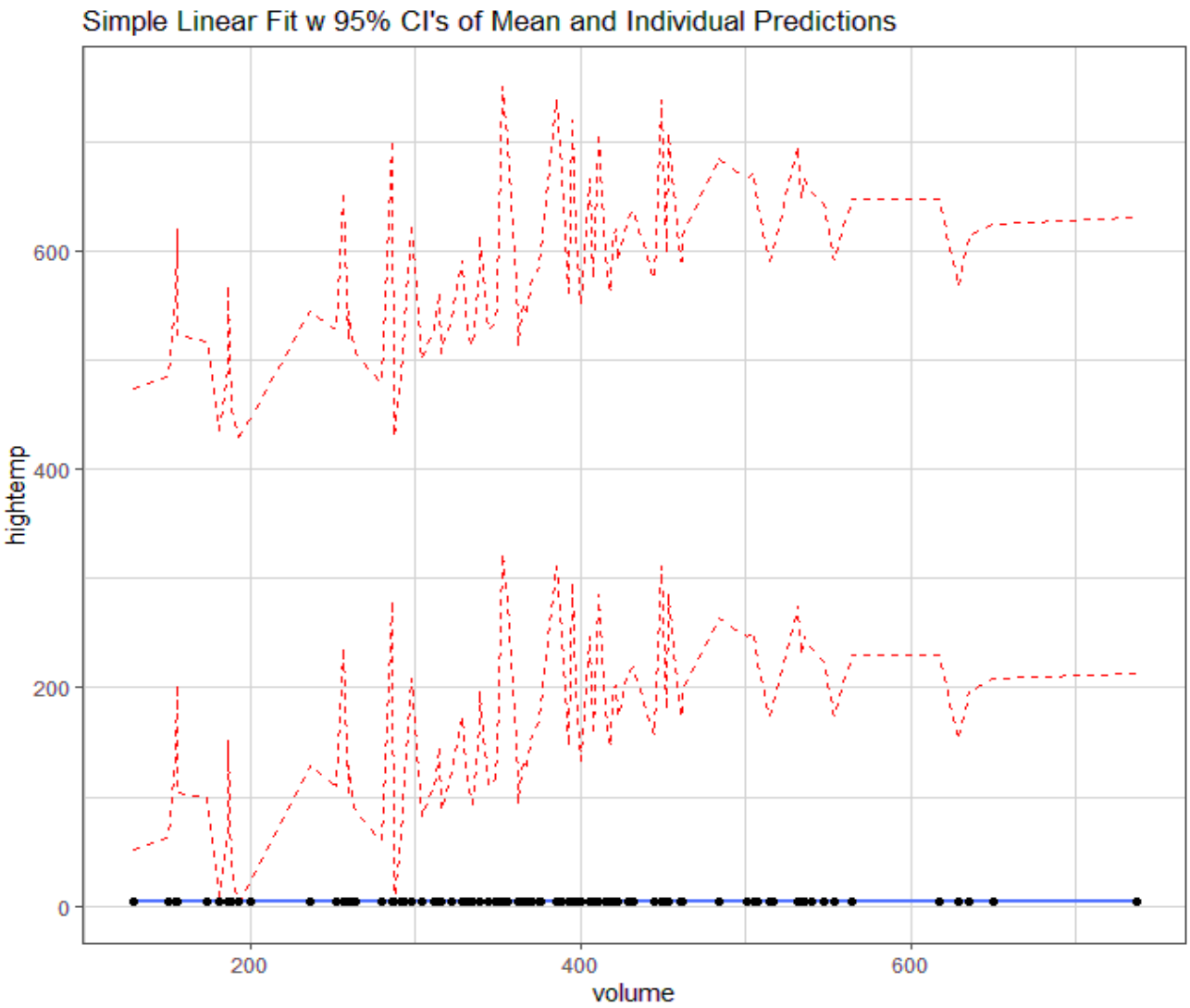

Figure 6: Post fit for simple linear regression examination.

4. Multiple Linear Regression

Multiple regression is a natural extension of simple linear regression that incorporates multiple explanatory (or predictor) variables. It has the general form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, \sigma_\epsilon). \quad (3)$$

The estimated coefficients (i.e., $\hat{\beta}_i$'s) are now interpreted as "conditional on" the other variables - each β_i reflects the predicted change in y associated with a one-unit increase in x_i, conditional upon the rest of the x_i's. This type of model can help to disentangle more complex relationships between three or more variables [25]. The value of R^2 from a multiple regression model has the same interpretation as before: the proportion of variability explained by the model.

In the case of multiple linear regression; we use the same function *lm()*, but her it takes more than one dependent variable *lm(dependent variable ~ independent* $Var_1 + Var_2 + \ldots + Var_n$*)* as follows;

```
## Multiple Linear Regression
Multiple_Model <- lm(volume ~ hightemp + lowtemp + cloudcover+ precip
                     , data = RailTrail)
summary(Multiple_Model)
```

```
## Call:
## lm(formula = volume ~ hightemp + lowtemp + cloudcover + precip,
##     data = RailTrail)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -269.447  -37.449    4.186   41.178  299.266
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   35.308     59.796   0.590   0.5564
## hightemp       6.571      1.153   5.699  1.7e-07 ***
## lowtemp       -1.290      1.387  -0.930   0.3551
## cloudcover    -7.501      3.851  -1.948   0.0547 .
## precip      -100.616     42.064  -2.392   0.0190 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 93.2 on 85 degrees of freedom
## Multiple R-squared:  0.4894, Adjusted R-squared:  0.4654
## F-statistic: 20.37 on 4 and 85 DF,  p-value: 8.537e-12
```

```
summary.aov(Multiple_Model)  # testing the significance of predictors
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## hightemp     1 490744  490744  56.500 5.26e-11 ***
## lowtemp      1 111176  111176  12.800 0.000575 ***
## cloudcover   1  56061   56061   6.454 0.012885 *
## precip       1  49695   49695   5.721 0.018964 *
## Residuals   85 738282    8686
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the case of multiple regression with a categorical variable, consider first the case where x_2 is an indicator variable that can only be "true or false" 0 or 1 (e.g., weekday). Then;

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2. \tag{4}$$

In the case where x_1 is quantitative but x_2 is an indicator variable, we have:

$$\text{For weekends, } \hat{y}|_{x_1, x_2=0} = \hat{\beta}_0 + \hat{\beta}_1 x_1$$

$$\text{For weekdays, } \hat{y}|_{x_1, x_2=1} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2(1) = \left(\hat{\beta}_0 + \hat{\beta}_2\right) + \hat{\beta}_1 x_1.$$

This is called a parallel slopes model since the predicted values of the model take the geometric shape of two parallel lines with slope $\hat{\beta}_1$: one with y-intercept $\hat{\beta}_0$ for weekends, and another with y-intercept $\hat{\beta}_0 + \hat{\beta}_2$ for weekdays, see Figure 7.

```
Multiple_Model2 <- lm(volume ~ hightemp + weekday, data = RailTrail)
summary(Multiple_Model2)
```

```
## Call:
## lm(formula = volume ~ hightemp + weekday, data = RailTrail)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -236.34  -59.86   12.38   60.88  281.41
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  42.8066    64.3444   0.665   0.5076
## hightemp      5.3478     0.8463   6.319 1.09e-08 ***
## weekdayTRUE -51.5535    23.6744  -2.178   0.0321 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 102 on 87 degrees of freedom
## Multiple R-squared:  0.3735,  Adjusted R-squared:  0.3591
## F-statistic: 25.94 on 2 and 87 DF,  p-value: 1.462e-09
```

```
plotModel(Multiple_Model2, system = "ggplot2")
```

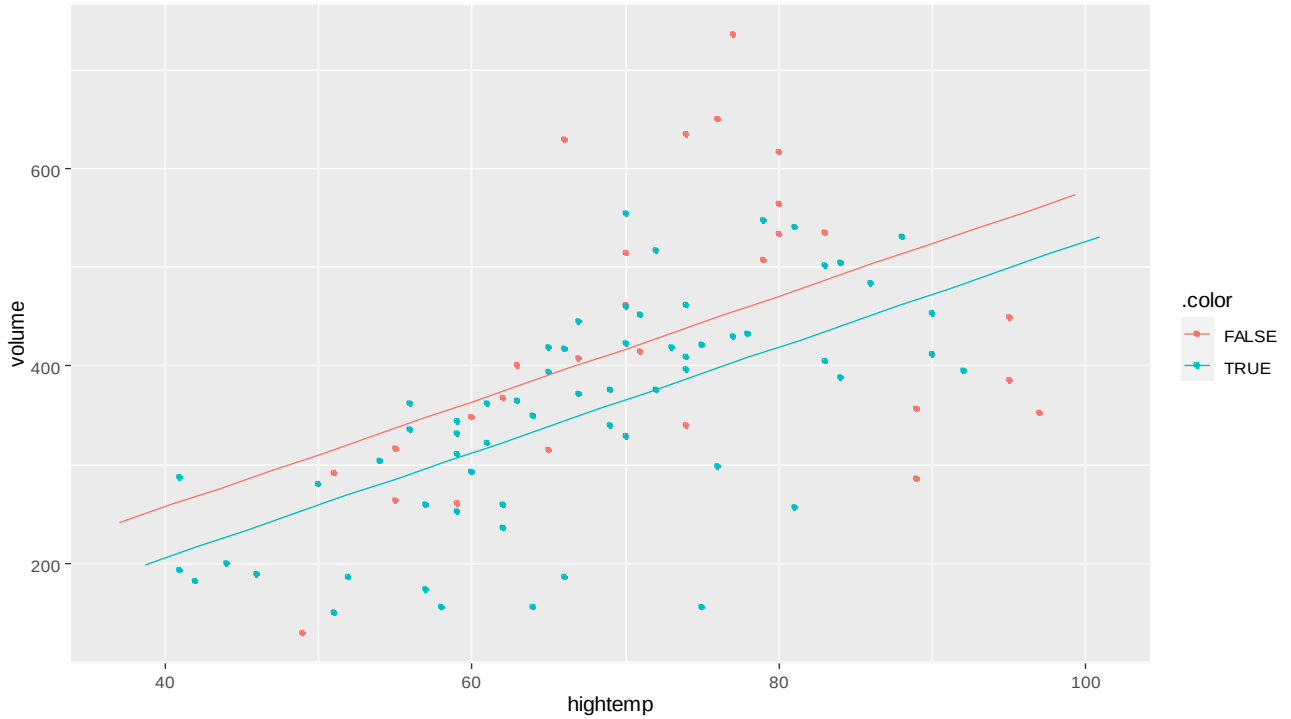

Figure 7: Scatterplot for multiple regression with a categorical variable.

```
# Testing for multicolinearity, A vif > 5 suggests collinearity

vif(Multiple_Model)

## hightemp    lowtemp cloudcover     precip
## 2.310808   2.765642   1.581822   1.254962
```

5. Non-linear Regression Model

We consider standard non-linear regression models of the following form:

$$y = f(\theta, x) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{5}$$

with y being the response (the dependent variable), x the (possibly multivariate) independent variable, which is often controlled by the experimenter, θ the vector of model parameters characterizing the relationship between x and y through the function f, and ϵ the residual error term that is assumed to be normally distributed, centered around 0 and with unknown variance (σ^2). Furthermore, we assume that the residual error terms are mutually independent as is usually assumed for standard non-linear regression analysis. In R, this non-linear regression model may be fitted using *nls ()* in the standard R installation (the package *stats*). Parameter estimation is based on an iterative procedure that involves a linearization approximation leading to a least-squares problem at each step.

Note that functions *gnls ()* and *nlme()* in *nlme* allow fitting of non-linear regression models for several curves corresponding to different covariate configurations (such as different treatments) and thus necessitating the use of correlation structures (e.g., random effects). However, for building these

more complex models (i.e., obtaining model fits that converge), *nls ()* is often used initially to produce fits of individual curves, which may then subsequently be combined and supplied to enable fitting more complex non-linear regression models (e.g., through the use of the wrapper *nlsList()*).

5.1 About nlstools package

The package *nlstools* provides a number of tools to facilitate fitting standard non-linear regression models (Equation (5)) and is specifically designed to work directly with *nls()*. The package contains functions and graphical tools that will help users to create *nls()* objects and carry out various diagnostic tests. More specifically, the *nlstools* toolbox will assist users in:

- Fitting non-linear models using function *nls()* by means of graphical tools.
- Getting a summary of parameter estimates, confidence intervals, residual standard error and sum of squares, and correlation matrix of the estimates
- Visualizing the fitted curve superimposed on the observations.
- Checking the validity of the error model by carrying out tests and graphical checks of residuals.
- Inspecting the contours of the residual sum of squares (likelihood contours) to detect possible structural correlations between parameters and the presence of potential local minimum.
- Visualizing the projection of confidence regions and investigate the nature of correlations.
- Using resampling techniques in order to detect influential observations and obtain non-parametric confidence intervals of the parameter estimates.

We will elaborate on these features in the next section, using a concrete data example from pulmonary medicine.

5.2 Non-linear Model Fitting in R

As mentioned in the introduction, fitting non-linear regression models requires the provision of starting values for model parameters. A poor choice of starting values may cause non-convergence or convergence to an unwanted local (rather than global) minimum when trying to minimize the least-squares criterion. Biologically interpretable parameter often allows the user to guess adequate starting values by assessing (often graphically) a set of plausible candidate model parameter values. For this purpose, *nlstools* provides the graphical function *preview()*, which can be used to assess the suitability of the chosen starting values, prior to fitting the model. This graphical approach for assessing candidate starting values is also used by Ritz and Streibig [26], but it was not wrapped up in a single function. Below is an example of usage. First, you should specify the model equation to be used in the non-linear regression as a formula in R. Use this formula as first argument of the function *preview()*, then supply the name of your dataset as second argument, and finally provide a list of values for the model parameters as third argument. An additional argument variable can be used to specify which independent variable is plotted against the dependent variable (column index of the original dataset; default is 1) when more than one independent variable is modeled.

```
# Non-linear Model fitting in R
library("nlstools")

formulaExp <- as.formula(VO2 ~ (t <= 5.883) * VO2rest + (t > 5.883) *
                           (VO2rest + (VO2peak - VO2rest) *
                              (1 - exp(-(t - 5.883) / mu))))
preview(formulaExp, data = O2K,
        start = list(VO2rest = 400, VO2peak = 1600, mu = 1))
```

Figure 7, shows good agreement between the data and the theoretical model based on the provided set of starting values. Judged by the figure, the chosen starting values seem to be suitable for initializing *nls()*. Note that next to the plot, the residual sum of squares measuring the discrepancy between the model (based on the chosen starting values) and the observed data is provided. This value gives an idea of the magnitude of the residual sum of squares to expect from the model fit based on *nls()*.

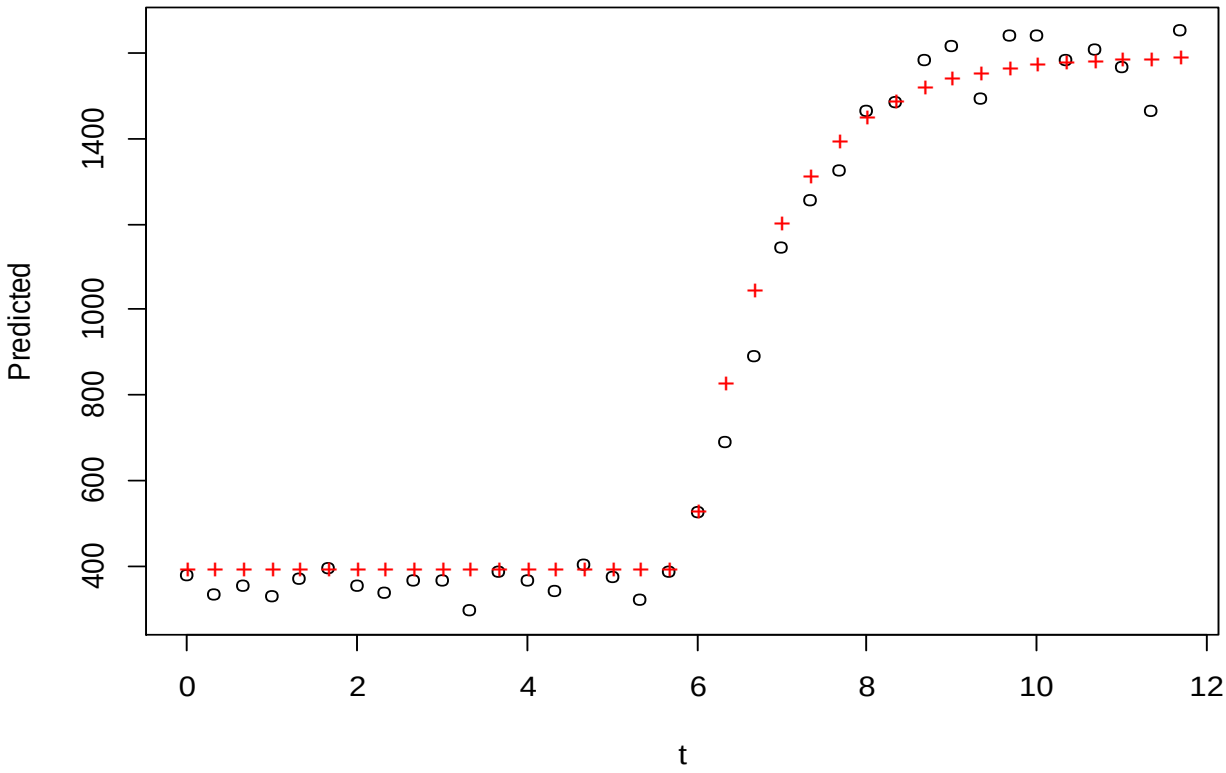

Figure 7: Graphically assessing the starting values prior the fit of a non-linear model.

Once suitable starting values are obtained, the model may be fitted using *nls()* and then the function *overview()* in *nlstools* may be used for providing a single display with all relevant pieces of information about the model fit.

```
O2K.nls1 <- nls(formulaExp, start = list(VO2rest = 400, VO2peak = 1600,
                                         mu = 1), data = O2K)
overview(O2K.nls1)
```

```
## Formula: VO2 ~ (t <= 5.883) * VO2rest + (t > 5.883) * (VO2rest +
## (VO2peak -VO2rest) * (1 - exp(-(t - 5.883)/mu)))
##
## Parameters:
##         Estimate Std. Error t value Pr(>|t|)
## VO2rest 3.568e+02  1.141e+01   31.26   <2e-16 ***
## VO2peak 1.631e+03  2.149e+01   75.88   <2e-16 ***
## mu      1.186e+00  7.661e-02   15.48   <2e-16 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 49.59 on 33 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 7.598e-06
## ------
## Residual sum of squares: 81200
## ------
## t-based confidence interval:
##                2.5%       97.5%
## VO2rest  333.537401  379.980302
## VO2peak 1587.155300 1674.611703
## mu         1.030255    1.342002
## ------
## Correlation matrix:
##           VO2rest    VO2peak        mu
## VO2rest 1.00000000 0.07907046 0.1995377
## VO2peak 0.07907046 1.00000000 0.7554924
## mu      0.19953773 0.75549241 1.0000000
```

```
plotfit(O2K.nls1, smooth = TRUE)
```

In order to facilitate the visualization of the model fit together with the data, *nlstools* provides the function *plotfit()*, which offers functionality similar to *abline()* with a simple linear regression model fit as argument. Thus *plotfit()* avoids manual definition of a grid of values for the independent variable, subsequent prediction, and use of *lines()*.The function superimposes the fitted curve on top of the plot of the data in Figure 8. Be aware that the fitted regression curve is represented more smoothly when the option *smooth = TRUE* is used. Only when a single (one-dimensional) independent variable is involved is this option accessible. The input variable in the function *plotfit()*

must be specified in order to select which independent variable will be used for the x axis for plots of a model fit containing multiple independent variables (for an example, see the worked example *michaelis* in *nlstools*). As it would also depend on the other independent variables in this scenario, smoothing is not an option, hence *smooth = FALSE*.

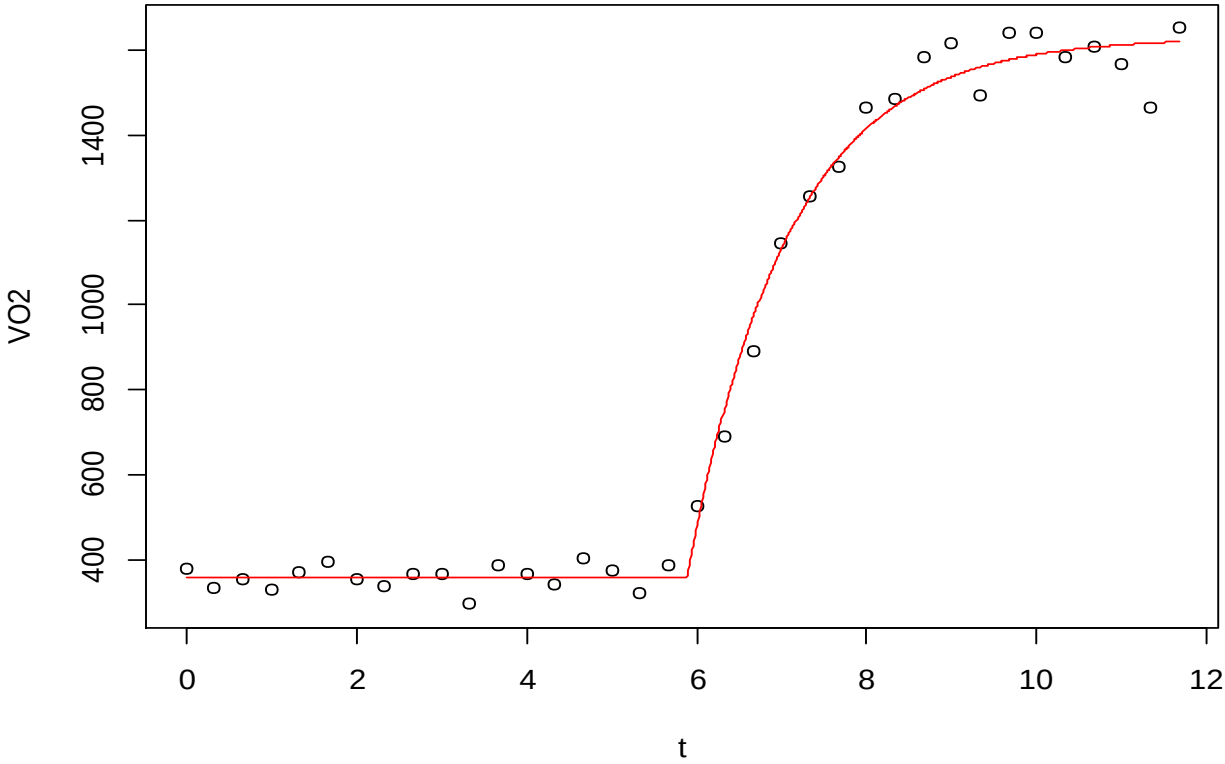

Figure 8: Plot of the data (dependent vs. independent variable) with the fitted model superimposed.

5.3 Assessing the goodness of fit through the residuals

An examination of the quality of the obtained non-linear regression model fit may be based on the residuals calculated from the fit as follows:

$$\hat{\epsilon} = y - f(\hat{\theta}, x). \tag{6}$$

Standardized residuals are obtained by dividing the centered residuals by the residual standard error. *nlstools* provides the function *nlsResiduals ()*, which extracts the residuals from an *nls* object. The Ritz and Streibig [26] diagnostic plots can be conveniently displayed using the related *plot()* method. For the model fit *O2K.nls1* the resulting plots are shown in Figure 9.

The top left panel of this figure displays the fitted values vs the raw residuals. The standardized residuals (with a mean of 0 and a standard deviation of 1) are displayed alongside the fitted values in the top right panel. The autocorrelation plot is displayed in the bottom left panel, and the QQ plot of the standardized residuals is displayed in the bottom right panel.

The residuals appear to be about normally distributed (a clear alignment along the diagonal in the QQ plot), and there is no sign of autocorrelation or heteroscedastic variance in the figure, which means there are no issues with the model assumptions.

```
O2K.res1 <- nlsResiduals(O2K.nls1)
plot(O2K.res1)
```

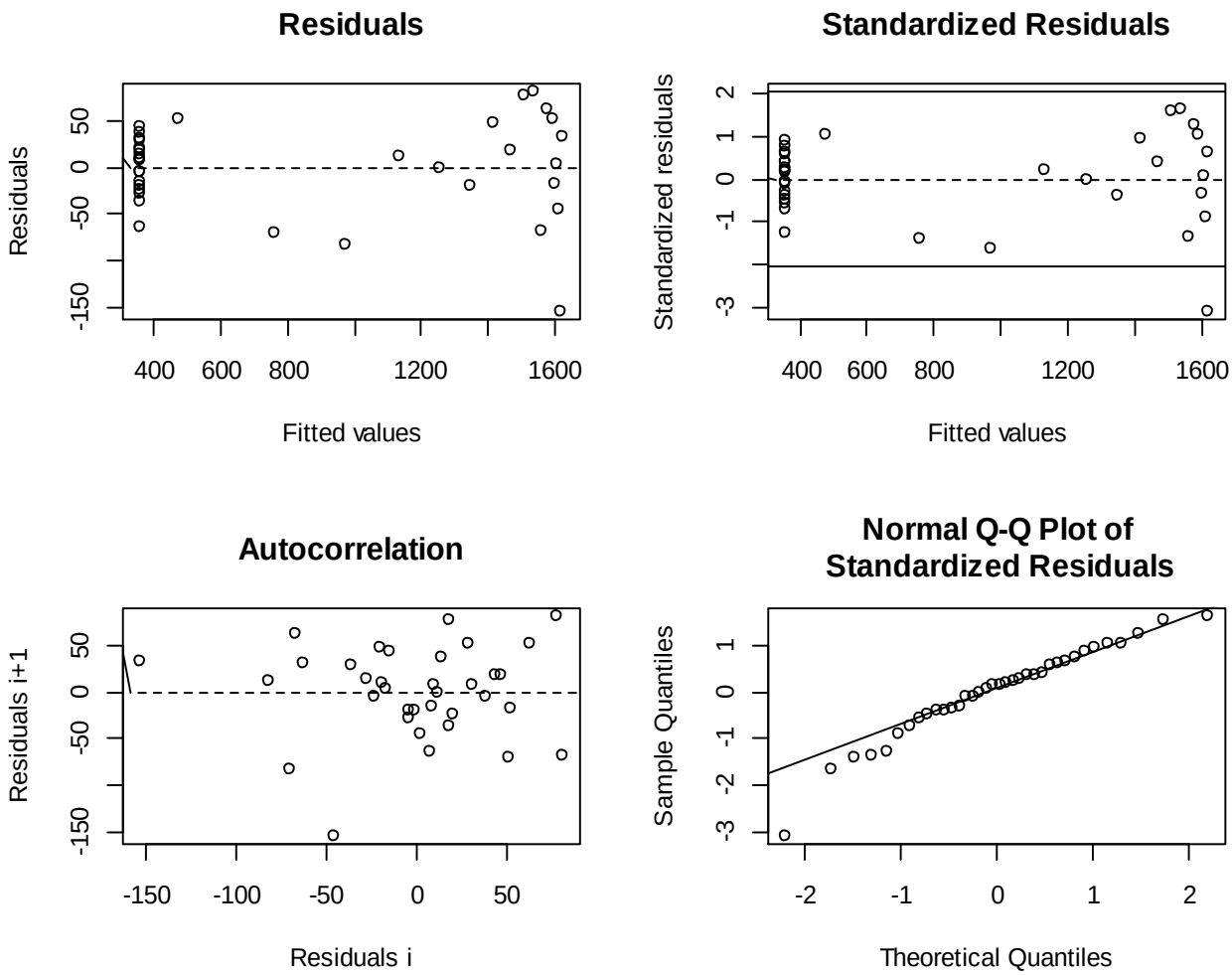

Figure 9: Residuals of non-linear model.

Moreover, the Shapiro-Wilk test can be used to assess the normality of residuals in addition to visually assessing the model assumptions. One of the more effective tests for normalcy is this one (the function *shapiro.test()* is included in the *stats* package in the default R installation). Similar to the runs test, the *runs.test()* function in the *tseries* package can be used to determine whether the residuals lack autocorrelation. But keep in mind that this is not a particularly reliable test since it effectively simply makes use of the residuals signals rather than their real values. We view these tests as complements to the standard visual evaluation of the model assumptions that are occasionally helpful. The *test.nlsResiduals()* function makes both tests available.

```
test.nlsResiduals(O2K.res1)          # normality and autocorrelation test
## ------
##   Shapiro-Wilk normality test
##
## data:  stdres
## W = 0.95205, p-value = 0.1214
## ------
##    Runs Test
## data:  as.factor(run)
## Standard Normal = 0.76123, p-value = 0.4465
## alternative hypothesis: two.sided
```

In our example, the null hypothesis (H_0) of normal distribution could not be rejected (since Shapiro-Wilk test: $p = 0.12$) and there was also no indication of autocorrelation (since runs test: $p = 0.45$).

5.4 Confidence regions

We use the following inequality to establish the $1-\alpha$ joint confidence region for the model parameters. If the matching residual sum of squares (RSS) falls within the margin specified in the following Equation (7) (often known as Beale's criterion), then the supplied set of parameters is said to be inside the confidence region, see [27].

$$RSS(\theta) < RSS_{\min}\left[1 + \frac{p}{n-p}F_{1-\alpha}(p, n-p)\right], \tag{7}$$

with $F_{1-\alpha}$ the appropriate quantile of the F-distribution with $(p, n-p)$ degrees of freedom, and RSS_{min} the minimum residual sum of squares obtained from the least-squares estimation (previously defined for the function *overview ()*), where n is the number of observations and p is the number of model parameters in f. The joint confidence zone specified in Equation (7) is visualized using two functions in *nlstools*: one that displays projections and the other that displays contours.

```
O2K.cont1 <- nlsContourRSS(O2K.nls1)
plot(O2K.cont1, col = FALSE, nlev = 5)
```

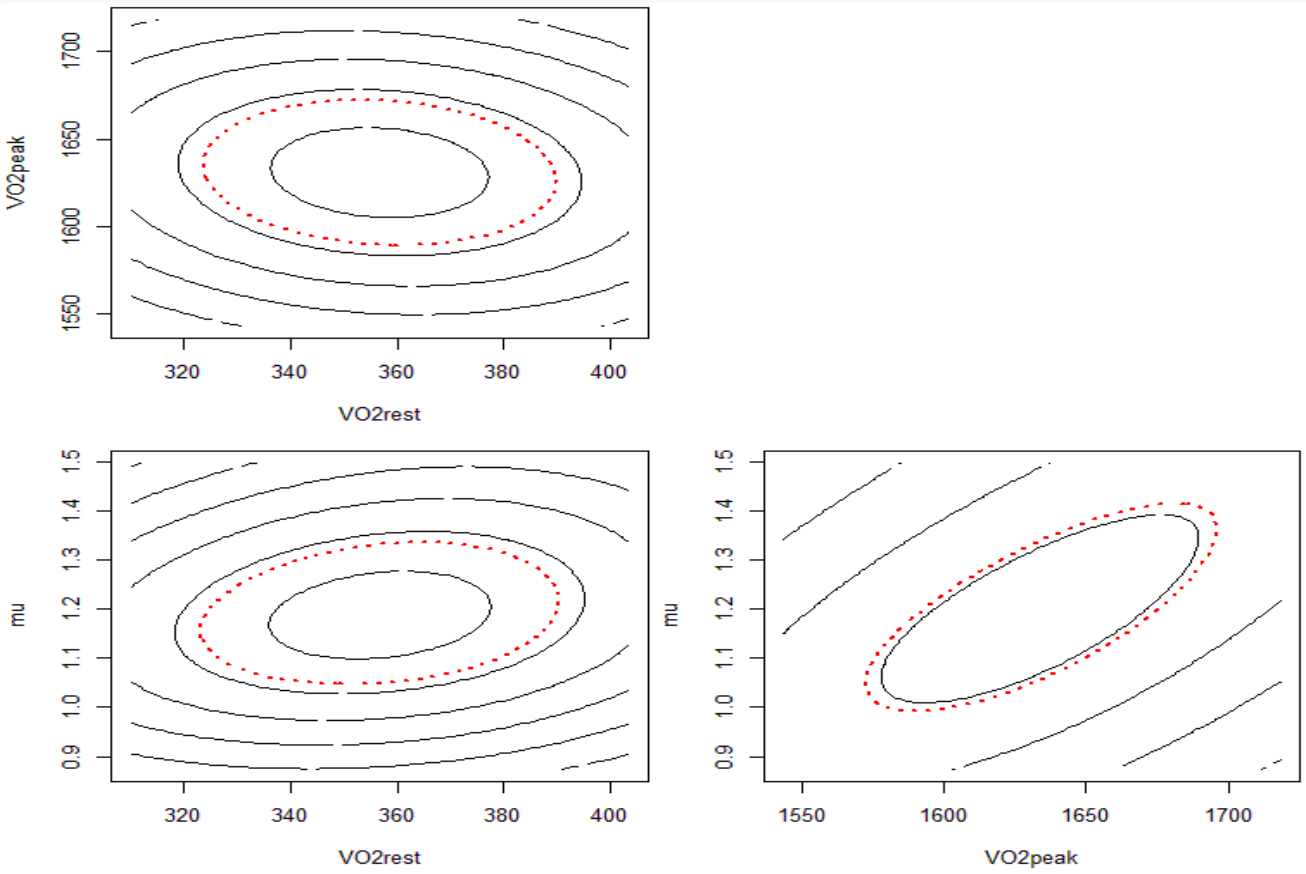

Figure 10: The panel contours based on the residual sum of squares.

```
O2K.conf1 <- nlsConfRegions(O2K.nls1, exp = 2, length = 2000)
plot(O2K.conf1, bounds = TRUE)
```

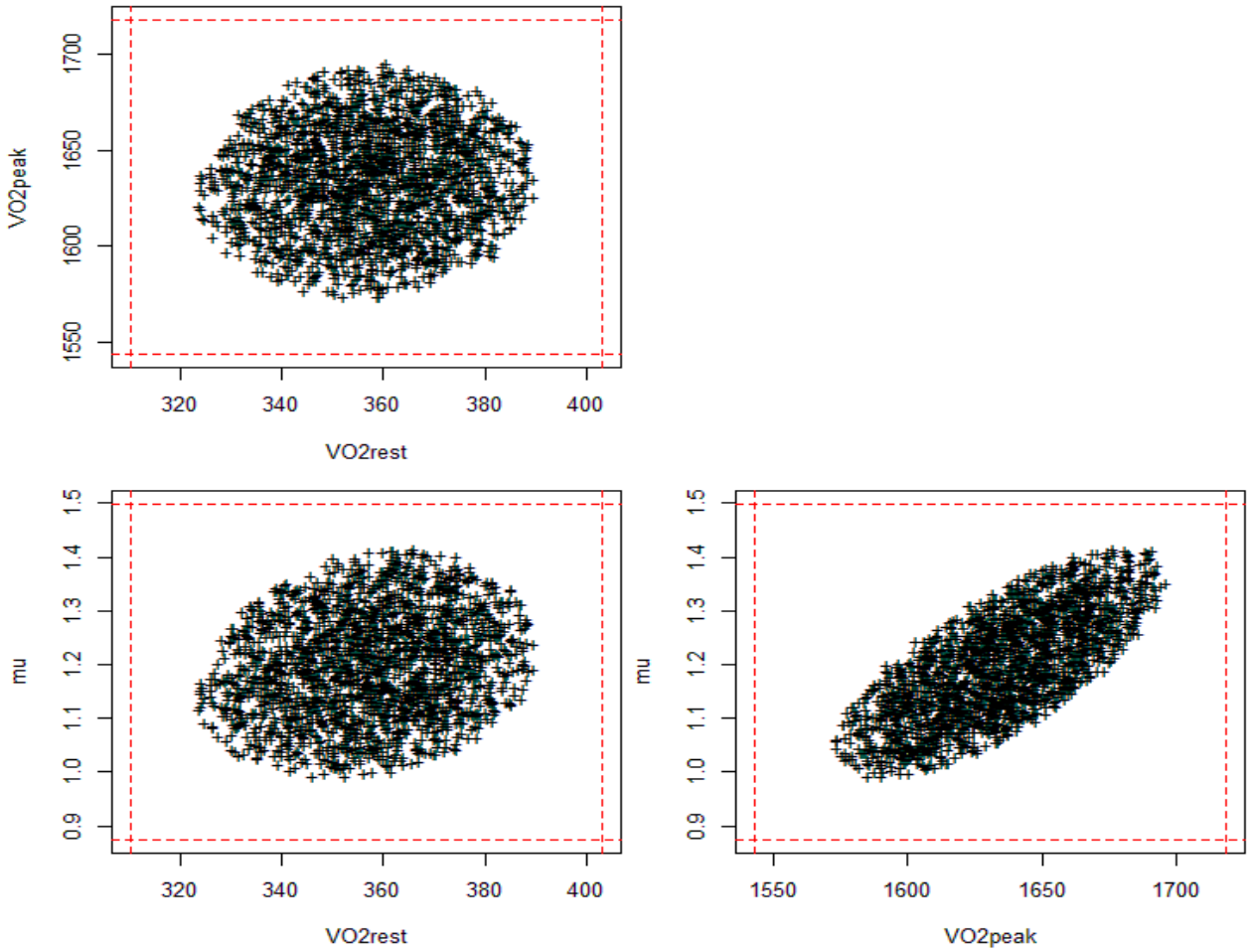

Figure 11: The projections of the confidence region according to the Beale's criterion.

The contours based on the residual sum of squares are shown in Figure 10. The Beale's 95% confidence zone is depicted by the contours shown by a red dotted line. According to Beale's criterion, the projections of the confidence region are displayed in Figure 11. The boundaries of the sample regions are indicated by the dashed red frames surrounding the confidence regions. Users can plot a different illustration of the Beale's confidence region, also known as the joint parameter likelihood region, using the function *nlsConfRegions()*, see [28, 29].

With a global minimum in the middle and excellent elliptical outlines in our case, the non-linear regression model fits the data well. The comparatively high connection between these two measures is reflected by the thinner elliptic shape of Beale's confidence zone between *VO2peak* and μ.

6. Conclusions

R is a programming language and an analytics tool; it is extensively used by Software Programmers, Statisticians, Data Scientists, and Data Miners. It is one of the most popular Data

analytics tools used in Data Analytics and Business Analytics. It has numerous applications in domains like healthcare, academics, consulting, finance, media, and many more. Its vast applicability in Statistics, Data Visualization, and Machine Learning have given rise to the demand for certified trained professionals in R. Therefore, in this paper, we studied the basics of R Programming language and we provide R-codes for linear and non-linear regression models with estimation. Practical evidence was provided to researchers to diagnose some problems in regression with R and test all assumptions.

Conflict of interest

The authors state that they have no financial or other conflicts of interest to disclose with connection to this paper.

References

1. Chatterjee, S., & Hadi, A. S. (2006). *Regression analysis by example*. John Wiley & Sons.
2. Abonazel, M., & Rabie, A. (2019). The impact of using robust estimations in regression models: An application on the Egyptian economy. *Journal of Advanced Research in Applied Mathematics and Statistics*, *4*(2), 8-16.
3. Abonazel, M. R., & Abd-Elftah, A. I. (2019). Forecasting Egyptian GDP using ARIMA models. *Reports on Economics and Finance*, *5*(1), 35-47.
4. Crawley, M. J. (2012). *The R book*. John Wiley & Sons.
5. Venables, W. N., Smith, D. M., & R Development Core Team. (2009). *An introduction to R*. John Wiley & Sons.
6. Winter, B. (2019). *Statistics for linguists: An introduction using R*. Routledge.
7. Chambers, J. (2008). *Software for Data Analysis: Programming with R*. New York: Springer.
8. Dalgaard, P., & Dalgaard, P. (2008). Advanced data handling. *Introductory Statistics with R*, 163-184.
9. Faraway, J. 2005. *Linear Models with R*. New York: Chapman and Hall/CRC.
10. Abonazel, M. R. (2018). A practical guide for creating Monte Carlo simulation studies using R. *International Journal of Mathematics and Computational Science*, *4*(1), 18-33.
11. Robert, C. P., Casella, G., & Casella, G. (2010). *Introducing monte carlo methods with r* (Vol. 18). New York: Springer.
12. Abonazel, M. R. (2015). How to create a Monte Carlo simulation study using R: with applications on econometric models. *In annual conference on statistics, computer sciences and operations research*. Faculty of Graduate Studies for Statistical Research, Cairo University, Egypt (Vol. 50).
13. Youssef, A. H., Kamel, A. R., & Abonazel, M. R. (2021). Robust SURE estimates of profitability in the Egyptian insurance market. *Statistical journal of the IAOS*, *37*(4), 1275-1287.

14. Abonazel, M.R., & Gad, A. E. (2020). Robust partial residuals estimation in semiparametric partially linear model. *Communications in Statistics-Simulation and Computation*, *49*(5), 1223-1236.
15. Abonazel, M. R., & Dawoud, I. (2022). Developing robust ridge estimators for Poisson regression model. *Concurrency and Computation: Practice and Experience*, *34*(15), e6979.
16. Youssef, A. H., Abonazel, M. R., & Kamel, A. R. (2022). Efficiency comparisons of robust and non-robust estimators for seemingly unrelated regressions model. *WSEAS Transactions on Mathematics*, *21*, 218-244.
17. Kamel, A. R. (2021). *Handling outliers in seemingly unrelated regression equations model*, MSc thesis, Faculty of graduate studies for statistical research (FGSSR), Cairo University, Egypt.
18. Kamel, A. R., & Alqarni, A. A. (2022). A New Approach for Model Selection with Two Qualitative Regressors. Computational Journal of Mathematical and Statistical Sciences, 1(1), 63-79.
19. Abonazel, M. R. (2019). New ridge estimators of SUR model when the errors are serially correlated. *International Journal of Mathematical Archive*, *10*(7), 53-62.
20. Abonazel, M. R., & Taha, I. M. (2021). Beta ridge regression estimators: simulation and application. *Communications in Statistics-Simulation and Computation*, 1-13.
21. Algamal, Z. Y., Lukman, A. F., Abonazel, M. R., & Awwad, F. A. (2022). Performance of the Ridge and Liu Estimators in the zero-inflated Bell Regression Model. *Journal of Mathematics*, 2022.
22. Dawoud, I., Abonazel, M. R., Awwad, F., & Tag Eldin, E. (2022). A New Tobit Ridge-Type Estimator of the Censored Regression Model with Multicollinearity Problem. *Frontiers in Applied Mathematics and Statistics*, 68.
23. Murrell, P. 2005. *R Graphics*. London: CRC Press.
24. Wickham, H., & Grolemund, G. (2016). *R for data science: import, tidy, transform, visualize, and model data*. " O'Reilly Media, Inc.".
25. Roback, P., & Legler, J. (2021). *Beyond multiple linear regression: applied generalized linear models and multilevel models in R*. Chapman and Hall/CRC.
26. Ritz, C., & Streibig, J. C. (2008). *Non-linear regression with R*. New York, NY: Springer New York.
27. Alharbi, Y. S., & Kamel, A. (2022). Fuzzy System Reliability Analysis for Kumaraswamy Distribution: Bayesian and Non-Bayesian Estimation with Simulation and an Application on Cancer Data Set. *WSEAS Transactions on Biology and Biomedicine*, *19*, 118-139.
28. Bates, D., & Watts, D. G. (1988). *Non-linear Regression Analysis and Its Applications*. John Wiley & Sons, Chichester, UK.

29. Hamdy, A., & Almetwally, E. M. (2023). Bayesian and Non-Bayesian Inference for The Generalized Power Akshaya Distribution with Application in Medical. Computational Journal of Mathematical and Statistical Sciences, 2(1), 31-51.