Linear Regression

Silva, Eder H N

2022-12-28

Correspondence Between Force and Command Input

Experiments were carried out to characterize the relations between the force generated by the main motor and the command input. For this purpose, a wire rope was used to connect the beam to a load cell a device used to measure the force. The force exerted by the main motor, drove by u, actuated in the load cell through the wire rope the corresponding data were collected and stored.

```
data <- read_csv("CSV/motor0_data1.csv")
```

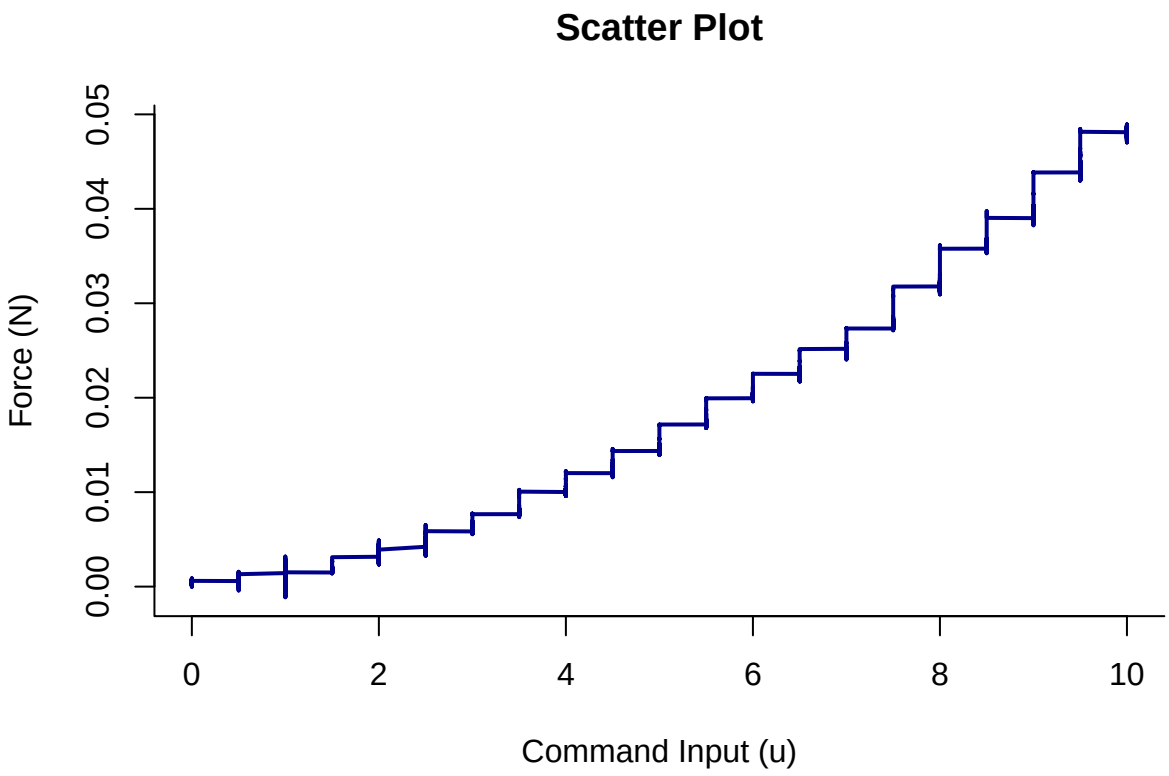
```
## New names:
## Rows: 38700 Columns: 3
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (3): -1...1, -1...2, -1...3
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `-1` -> `-1...1`
## * `-1` -> `-1...2`
## * `-1` -> `-1...3`
```

```
data <- as.data.frame(data)
colnames(data)=c("sample","force","u")
head(data)
```

```
##   sample    force u
## 1 38.699 0.000327 0
## 2 38.698 0.000313 0
## 3 38.697 0.000300 0
## 4 38.696 0.000290 0
## 5 38.695 0.000286 0
## 6 38.694 0.000287 0
```

Motor 0 standardized data

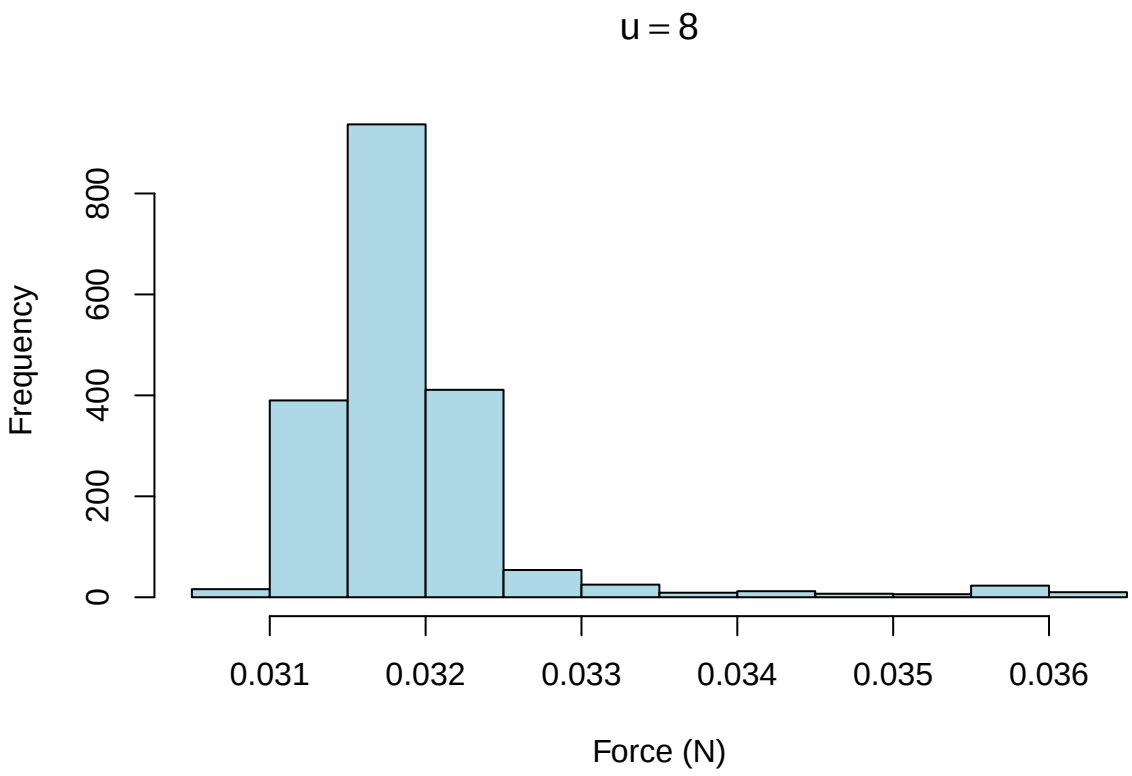
```
plot(x=data[,3],y=data[,2],ylim=c(min(data[,2]),max(data[,2])),type="n",ylab="Force (N)",xlab="Command
par(new=TRUE)
plot(x=data[,3],y=data[,2],type="l",ylim=c(min(data[,2]),max(data[,2])),axes=FALSE,ann=FALSE,col="darkb
```



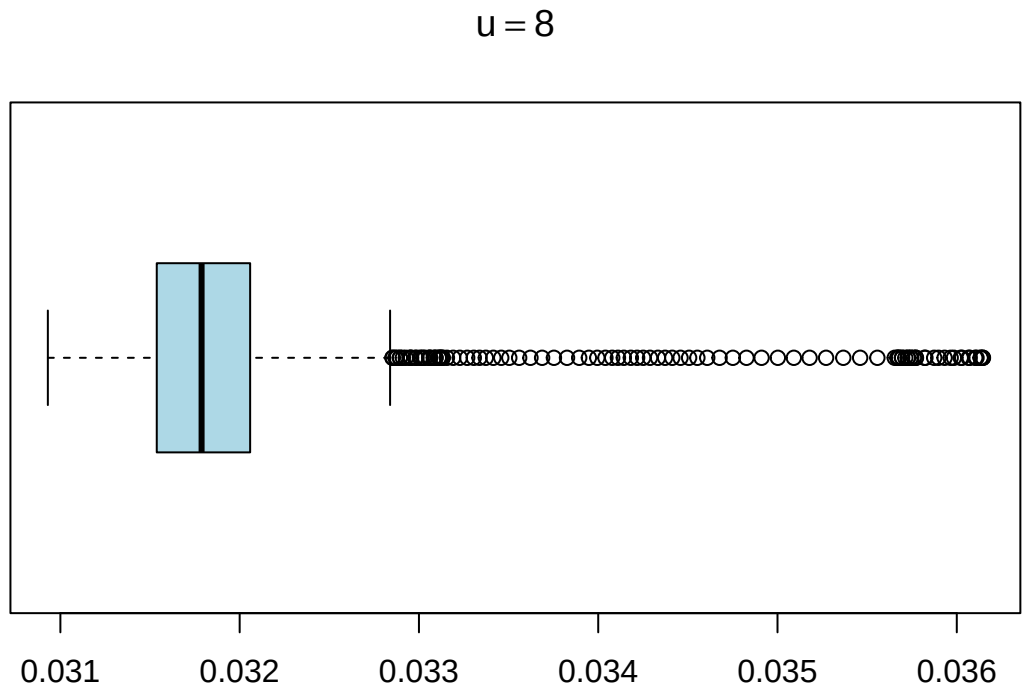
Example sample record of $u = 8$

```
filtered_rows <- data$u == 8
rows <- data[filtered_rows,]

hist(rows[,2],xlab="Force (N)",main=TeX('$u = 8$'),col="#ADD8E6")
```



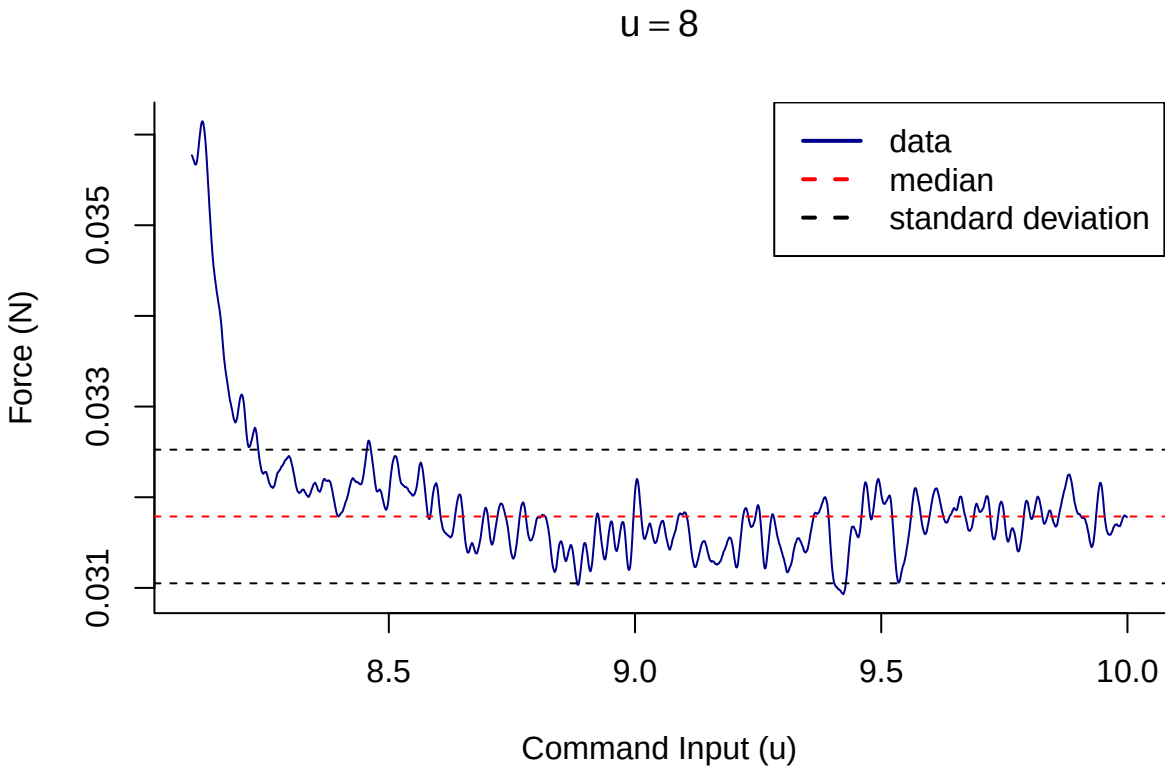

```
boxplot(rows[,2],main=TeX('$u = 8$'),horizontal=TRUE,col="#ADD8E6")
```




```
shapiro.test(rows[,2])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rows[, 2]
## W = 0.65565, p-value < 2.2e-16
```

```
plot(rows[,1],rows[,2],type="l",col="darkblue",lwd=1,ylab="Force (N)",xlab="Command Input (u)",main=TeX
abline(h=median(rows[,2]),lty=2,col="red")
abline(h=(median(rows[,2])+sd(rows[,2])),lty=2,col="black")
abline(h=(median(rows[,2])-sd(rows[,2])),lty=2,col="black")
legend("topright",legend = c('data','median','standard deviation'),col = c("darkblue","red","black"),lty
```



According to the previous figure, there are indications the condition of normality is not met. Therefore, the mean may not be a good representation of the data, which is why the median was used.

```
uni <- unique(data[,3])
modelagem = matrix(data=0, ncol=4, nrow=length(uni))
colnames(modelagem)=c("u","Mean","Median","Length")

for (i in 1:length(uni)) {

  filtered_rows <- data$u == uni[i]
  modelagem[i,1] <- uni[i]
  rows <- data[filtered_rows,]
  modelagem[i,2]  <- mean(rows[,2])
  modelagem[i,3] <- median(rows[,2])
  modelagem[i,4] <- length(rows[,2])
}
head(modelagem)
```

```
##        u         Mean    Median Length
## [1,] 0.0 0.0004251545 0.0004280   1800
## [2,] 0.5 0.0004659592 0.0004700   1800
## [3,] 1.0 0.0009030337 0.0009270   1700
## [4,] 1.5 0.0017865271 0.0017500   1700
## [5,] 2.0 0.0029095326 0.0028575   1900
## [6,] 2.5 0.0043809567 0.0044010   1800
```

It can be seen that the samples do not have the same size, but as many samples as possible were used, seeking

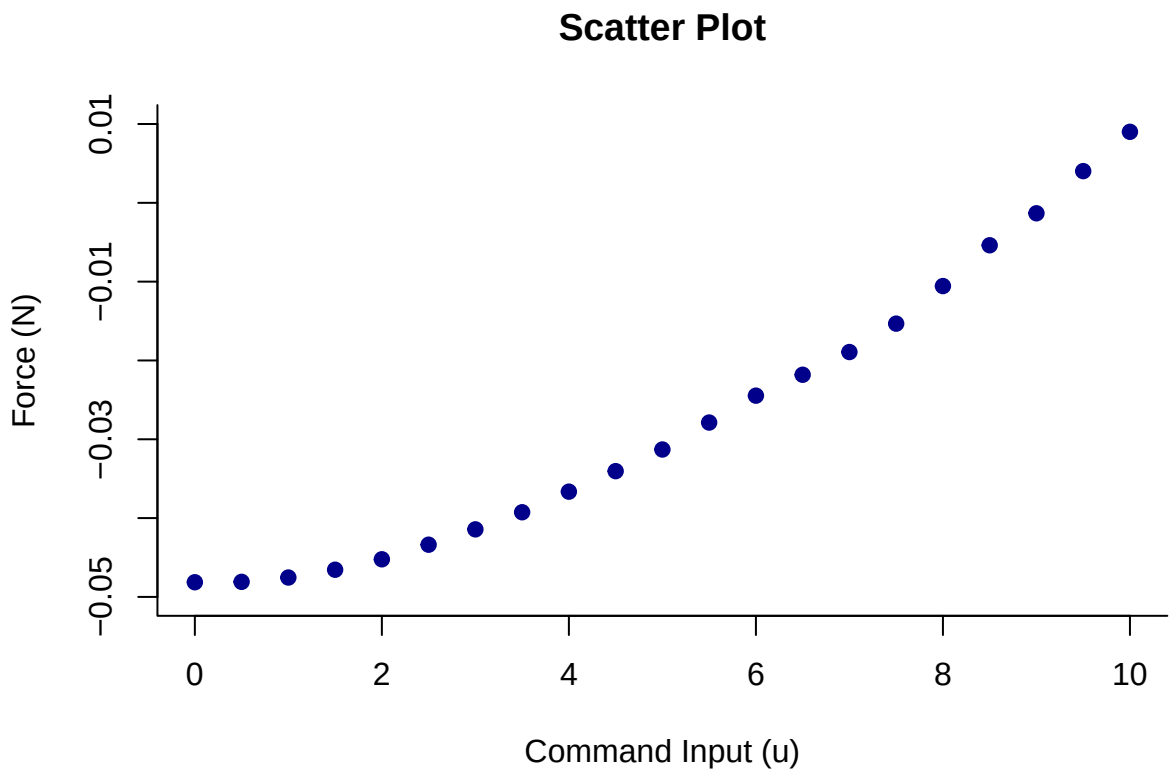
a good representation of the data.

Using scatter plots to explore relationships

Before looking ahead to predicting a value of force by using a value of u, first establish the legitimate reason to using a linear function to make that prediction will actually work well.

In order to achieve both of these important steps, first plot the data in a pairwise fashion so you can visually look for a relationship; then need to somehow quantify that relationship in terms of how well those points follow a linear function.

```
plot(x=data[,3],y=data[,2],ylim=c(-.05,.01),type="n",ylab="Force (N)",xlab="Command Input (u)",main="Sca
par(new=TRUE)
plot(x=modelagem[,1],y=modelagem[,3],pch=19,ylim=c(min(data[,2]),max(data[,2])),axes=FALSE,ann=FALSE,col
```



Each circle represents the statistical median of more than one-thousand points.

After displaying the data using a scatter plot, the next step is to find a statistic that quantifies the relationship somehow. The correlation coefficient (also known as Pearson's correlation coefficient) measures the strength and direction of the linear relationship between two quantitative variables u and y.

```
cor.test(x=modelagem[,1],y=modelagem[,3])
```

```
##
##  Pearson's product-moment correlation
##
## data:  modelagem[, 1] and modelagem[, 3]
## t = 19.897, df = 19, p-value = 3.497e-14
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9426521 0.9907399
## sample estimates:
##      cor
## 0.976835
```

Building a Simple Linear Regression Model

$$y_i = \beta_0 + \beta_1 u_i + \beta_2 u_i^2 + \beta_3 u_i^3 + \varepsilon_i$$

```
u <- modelagem[,1]
y <- modelagem[,3]

X <- u
XSQ <- u**2
XCUB <- u**3

model=lm(y~1+X+XSQ+XCUB)
anova(model)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df    Sum Sq   Mean Sq    F value Pr(>F)
## X          1 0.0044551 0.0044551 22676.9971 <2e-16 ***
## XSQ        1 0.0002105 0.0002105  1071.2191 <2e-16 ***
## XCUB       1 0.0000000 0.0000000     0.0748 0.7877
## Residuals 17 0.0000033 0.0000002
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model)
```

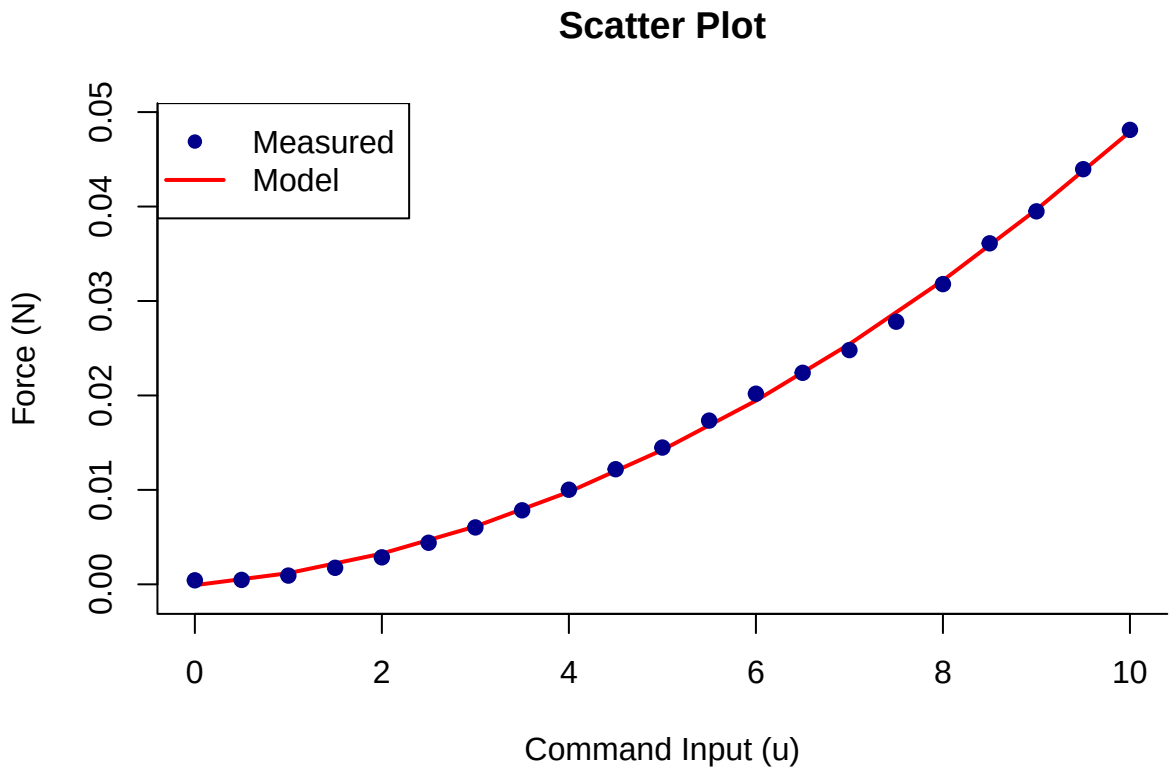
```
##
## Call:
## lm(formula = y ~ 1 + X + XSQ + XCUB)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -8.994e-04 -2.544e-04  2.414e-05  2.534e-04  7.376e-04
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.770e-05  3.269e-04  -0.268  0.79173
## X            8.645e-04  2.902e-04   2.979  0.00843 **
## XSQ          4.059e-04  6.842e-05   5.932 1.64e-05 ***
## XCUB        -1.229e-06  4.492e-06  -0.274  0.78773
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0004432 on 17 degrees of freedom
## Multiple R-squared:  0.9993, Adjusted R-squared:  0.9992
## F-statistic:  7916 on 3 and 17 DF,  p-value: < 2.2e-16
```

```
summ=summary(model)
func <- model$coefficients
print(func)
```

```
##   (Intercept)             X           XSQ          XCUB
## -8.770158e-05  8.644903e-04  4.058642e-04 -1.228902e-06
```

```
u=0:10
y=as.numeric(func[1])+as.numeric(func[2])*u+as.numeric(func[3])*u**2+as.numeric(func[4])*u**3

plot(x=data[,3],y=data[,2],ylim=c(min(data[,2]),max(data[,2])),type="n",ylab="Force (N)",xlab="Command
par(new=TRUE)
plot(u,y,type="l",ylim=c(min(data[,2]),max(data[,2])),axes=FALSE,ann=FALSE,col="red",lwd=2)
par(new=TRUE)
plot(x=modelagem[,1],y=modelagem[,3],pch=19,ylim=c(min(data[,2]),max(data[,2])),axes=FALSE,ann=FALSE,co
legend("topleft",legend = c('Measured','Model'),col = c("darkblue","red"),lty = c(0, 1), lwd=2, pch =c(
```



Using r^2 to measure model fit

One important way to assess how well the model fits is to measure the value of r^2, where r is the correlation coefficient. Statisticians measure how well a model fits by looking at what percentage of the variability in $\hat{y}$ is explained by the model.

```
summ$r.squared
```

```
## [1] 0.9992847
```

Finding and exploring the residuals

After you've established a relationship between u and y and have come up with an equation of a linear that represents that relationship, the job is not done. (Many researchers erringly stop here, so I'm depending on you to break the cycle on this!)

But the most important job remains to be completed: checking to be sure that the conditions of the model are truly met and that the model fits well in more specific ways than the scatter plot and correlation measure. This section presents methods for defining and assessing the fit of a simple linear regression model.

Finding the residuals

A residual is the difference between the observed value of force and the predicted value of $\hat{y}$. Specifically, for any data point, takes observed y (from the data) and subtract the expected $\hat{y}$. If the residual is large, the linear function doesn't fit well in that spot. If the residual is small, the line fits well in that spot.

```
error=resid(model)
```

The conditions for regression concentrate on the error terms, or residuals. The residuals are the amount that's left over after the model has been fit. They represent the difference between the actual value of y observed in the data set and the estimated value of $\hat{y}$ based on the model. The conditions of the regression model are the following:

Noting the conditions

- **The residuals have a normal distribution with mean zero $\varepsilon_i \backsim N(0, \sigma^2)$.**

```
summary(error)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -8.994e-04 -2.544e-04  2.414e-05  0.000e+00  2.534e-04  7.376e-04
```

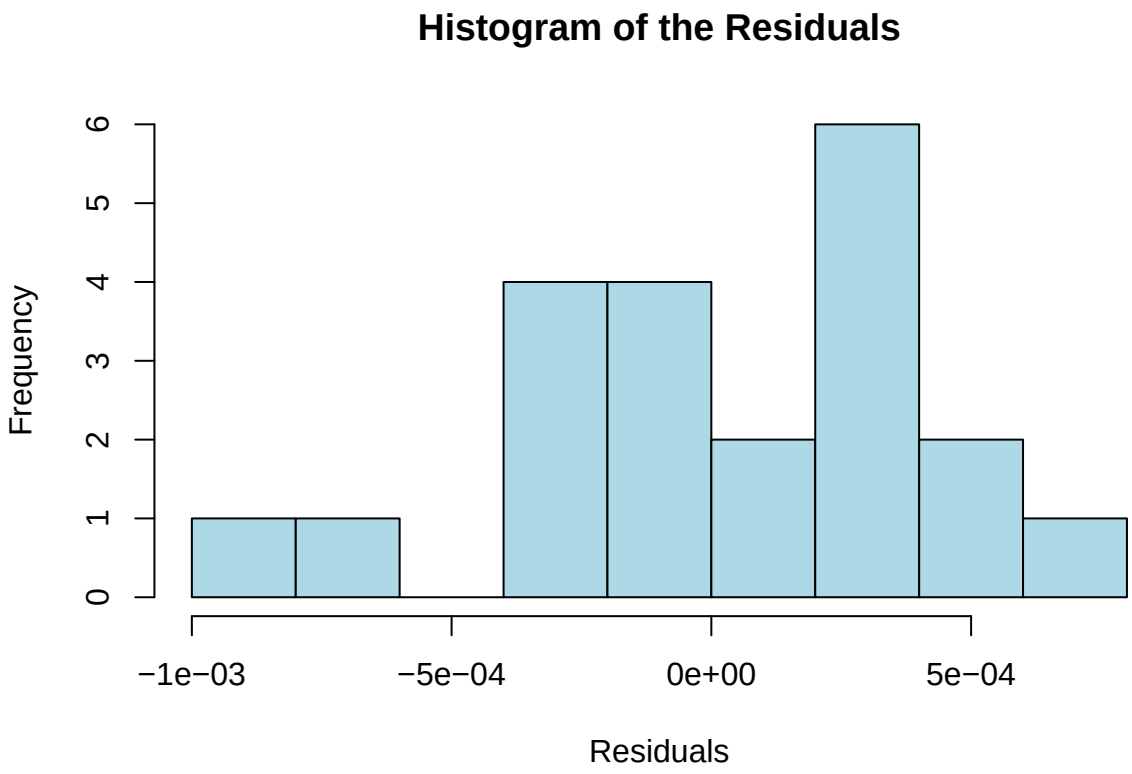
```
sd(error)
```

```
## [1] 0.0004086437
```

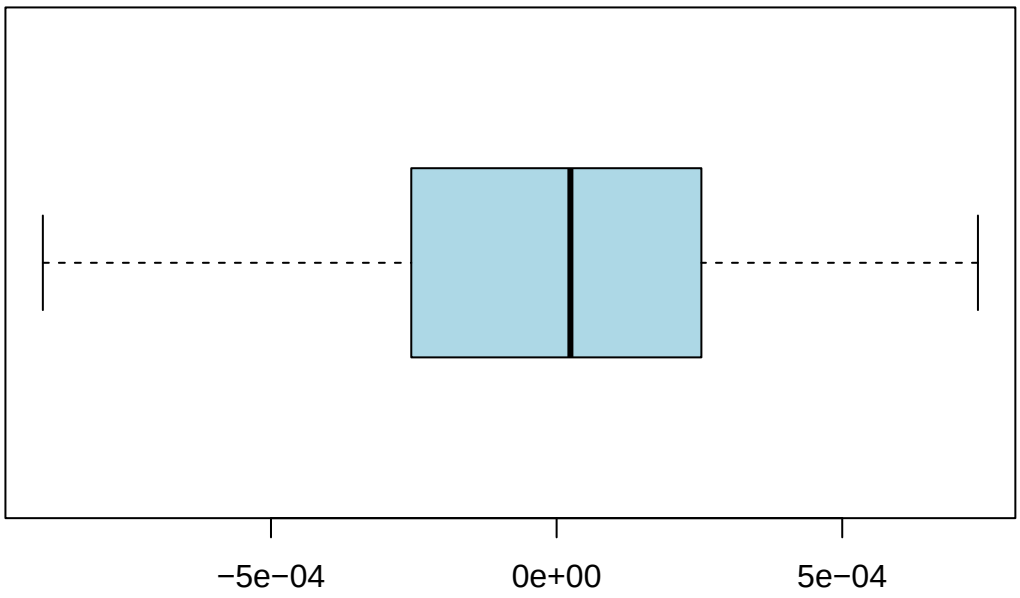
```
shapiro.test(error)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  error
## W = 0.98078, p-value = 0.936
```

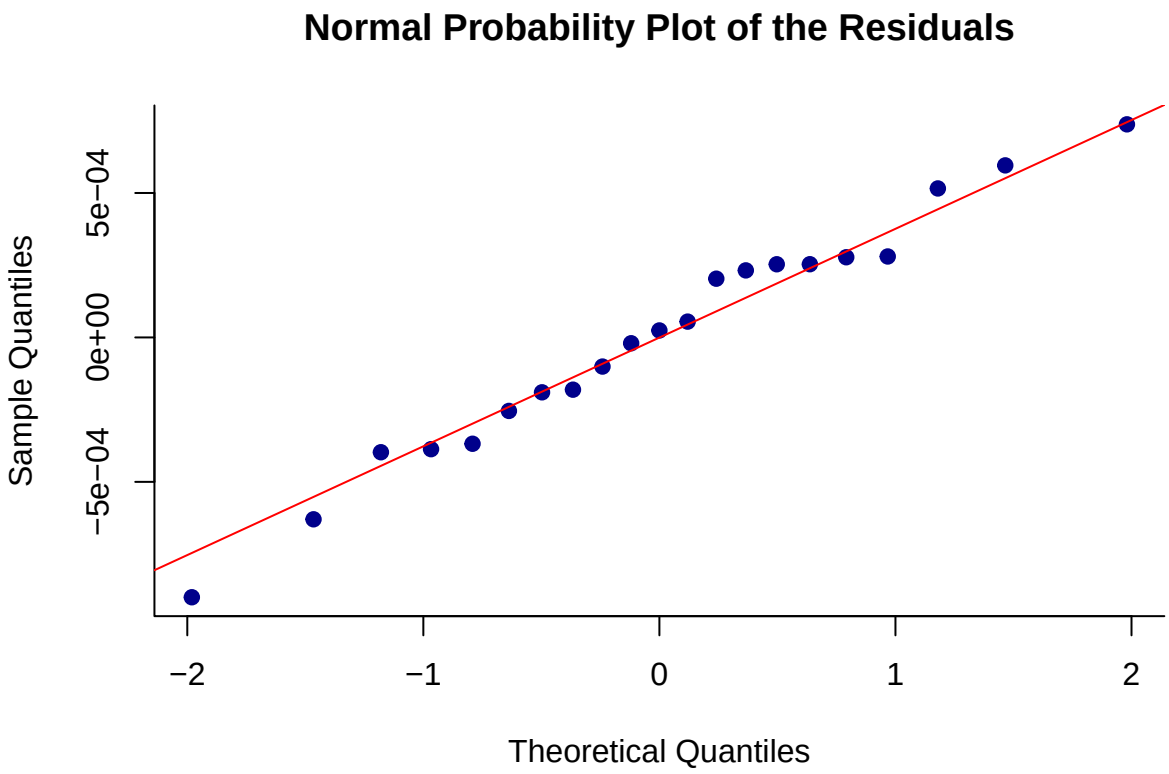
```
hist(error,col="#ADD8E6",main="Histogram of the Residuals",xlab="Residuals")
```




```
boxplot(error,main="Residuals",col="#ADD8E6",horizontal=TRUE)
```




```
qqnorm(error,pch=19,col="darkblue",main="Normal Probability Plot of the Residuals",bty = "l")
qqline(error,col="red")
```



As the condition of normality is met, then residual plot lots of residuals close to zero. The residuals also occur at random some above the line, some below the line.

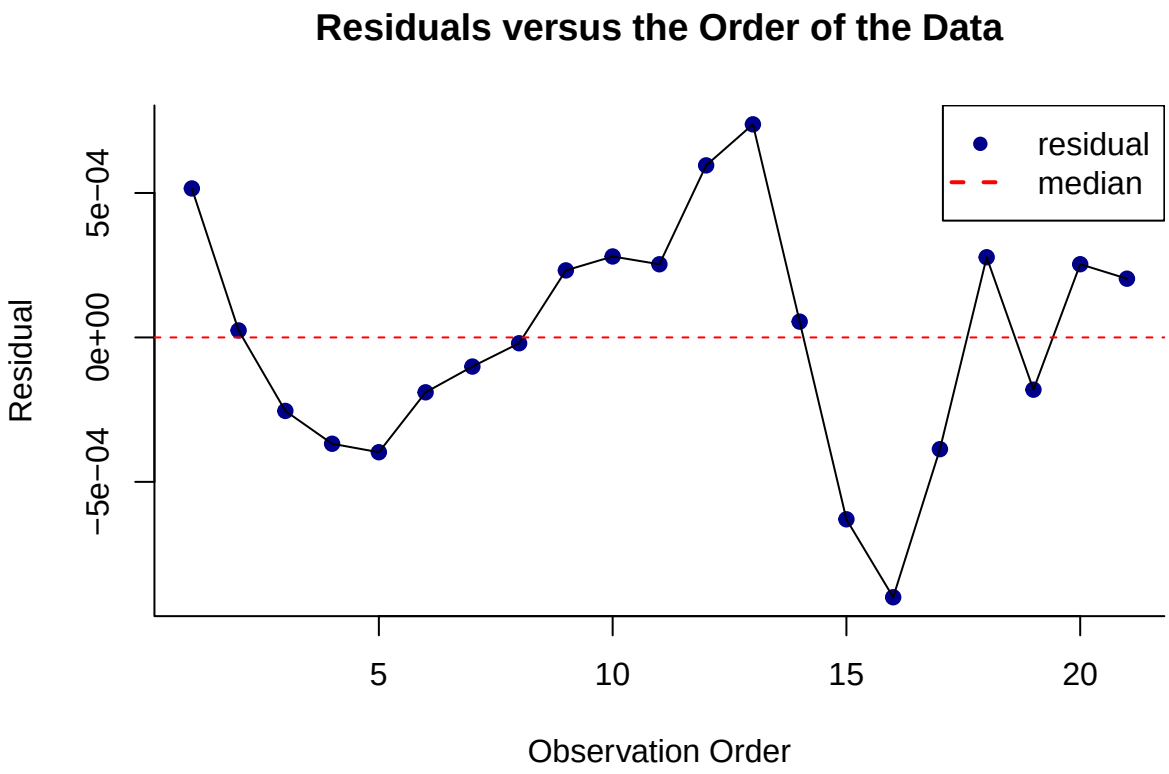
- **The residuals have the same variance for each fitted (predicted) value of $\hat{y}$, $Var(\varepsilon_i) = \sigma^2$.**

```
var(error)
```

```
## [1] 1.669897e-07
```

- **The residuals are independent (don't affect each other).**

```
plot(error,pch=19,col="darkblue",main="Residuals versus the Order of the Data",xlab="Observation Order"
par(new=TRUE)
plot(error,type="l",axes=FALSE,ann=FALSE)
abline(h=0,lty=2,col="red")
legend("topright",legend = c('residual','median'),col = c("darkblue","red"),lty = c(0,2), lwd=2, pch =c
```



Writing into JSON file

```
# creating the list
list1 <- vector(mode="list", length=2)
list1[[1]] <- c("Intercept", "u", "u^2", "u^3")
list1[[2]] <- c(as.numeric(func[1]), as.numeric(func[2]),as.numeric(func[3]), as.numeric(func[4]))

# creating the data for JSON file
jsonData <- toJSON(list1)

# writing into JSON file
write(jsonData, "JSON/motor0.json")

# Give the created file name to the function
motor0 <- fromJSON(file = "JSON/motor0.json")

# Print the result
print(motor0)
```

```
## [[1]]
## [1] "Intercept" "u"         "u^2"       "u^3"
##
## [[2]]
## [1] -8.770158e-05  8.644903e-04  4.058642e-04 -1.228902e-06
```

```
list2 <- vector(mode="list", length=1)
list2[[1]] <- error
jsonData <- toJSON(list2)
write(jsonData, "JSON/motor0_resi.json")

list3 <- vector(mode="list", length=1)
list3[[1]] <- modelagem[,3]
jsonData <- toJSON(list3)
write(jsonData, "JSON/motor0_median.json")
```

Motor 1 standardized data

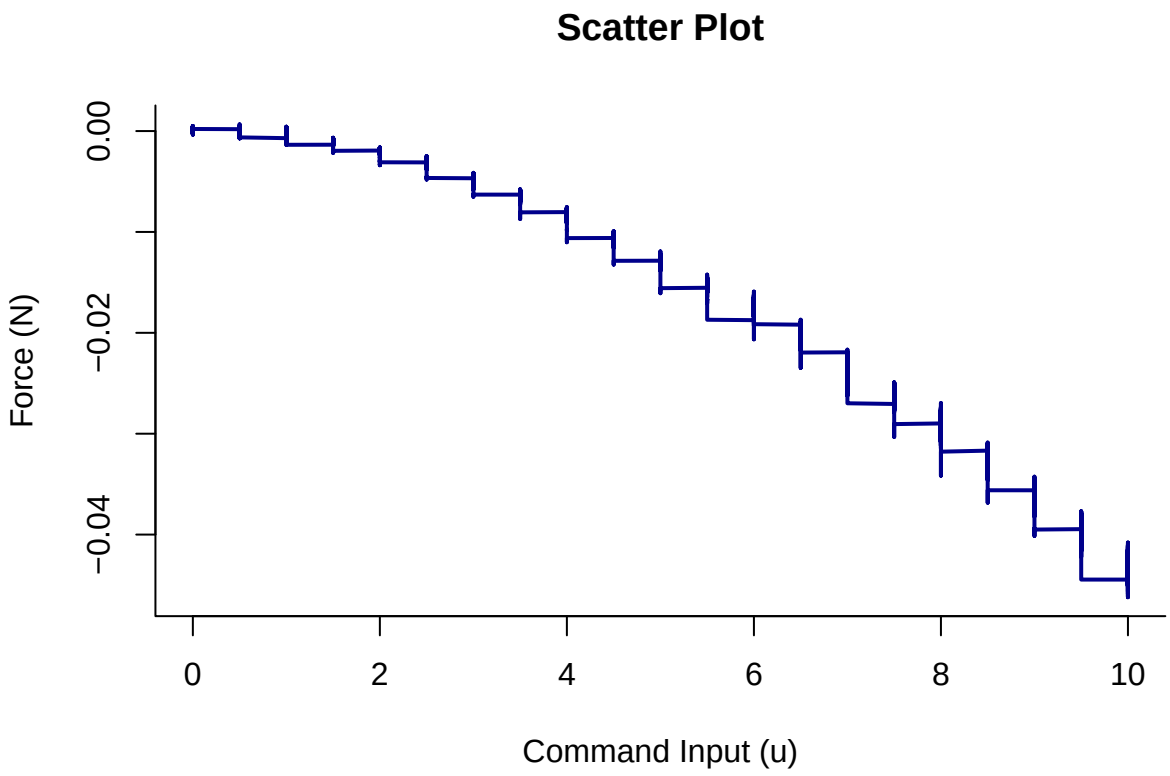
```
data <- read_csv("CSV/motor1_data1.csv")
```

```
## New names:
## Rows: 47700 Columns: 3
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (3): -1...1, -1...2, -1...3
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `-1` -> `-1...1`
## * `-1` -> `-1...2`
## * `-1` -> `-1...3`
```

```
data <- as.data.frame(data)
colnames(data)=c("sample","force","u")
head(data)
```

```
##   sample    force u
## 1 47.699 0.000113 0
## 2 47.698 0.000114 0
## 3 47.697 0.000115 0
## 4 47.696 0.000115 0
## 5 47.695 0.000112 0
## 6 47.694 0.000106 0
```

```
plot(x=data[,3],y=data[,2],ylim=c(min(data[,2]),max(data[,2])),type="n",ylab="Force (N)",xlab="Command
par(new=TRUE)
plot(x=data[,3],y=data[,2],type="l",ylim=c(min(data[,2]),max(data[,2])),axes=FALSE,ann=FALSE,col="darkb
```



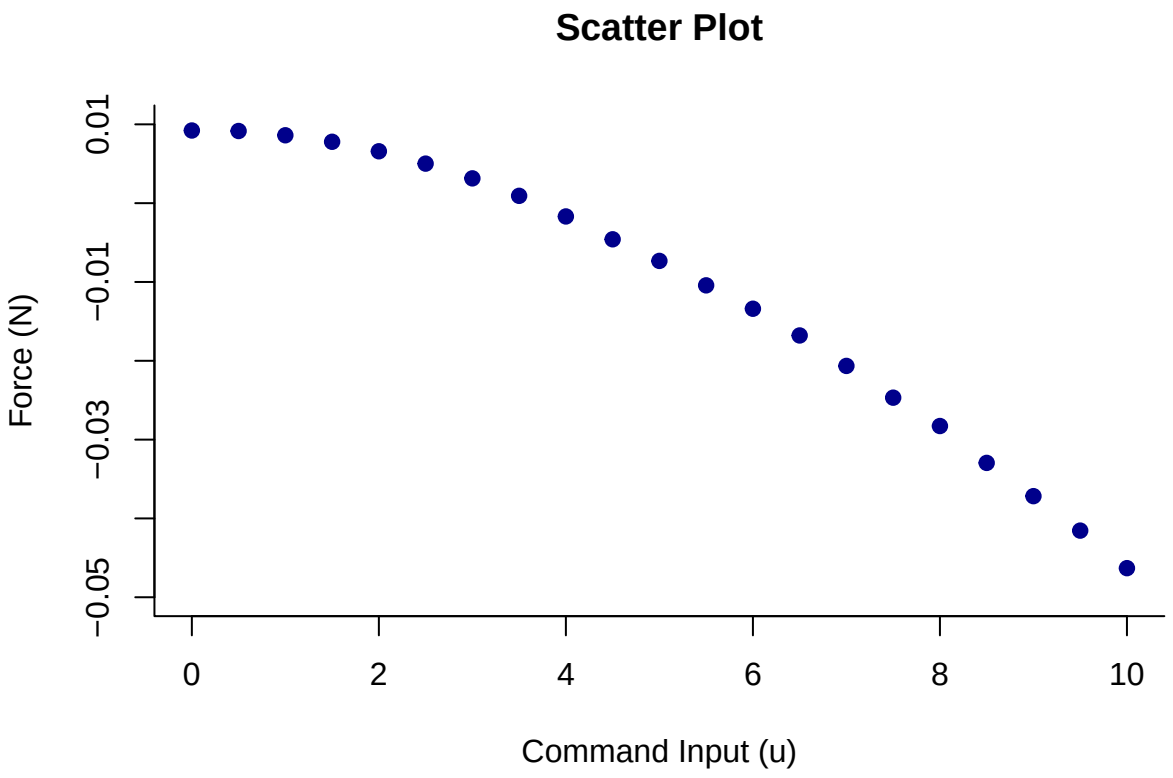

```
uni <- unique(data[,3])
modelagem = matrix(data=0, ncol=4, nrow=length(uni))
colnames(modelagem)=c("u","Mean","Median","Length")

for (i in 1:length(uni)) {

  filtered_rows <- data$u == uni[i]
  modelagem[i,1] <- uni[i]
  rows <- data[filtered_rows,]
  modelagem[i,2]  <- mean(rows[,2])
  modelagem[i,3] <- median(rows[,2])
  modelagem[i,4] <- length(rows[,2])
}
head(modelagem)
```

```
##        u          Mean        Median Length
## [1,] 0.0  6.068558e-05  5.708280e-05   2800
## [2,] 0.5  1.264576e-05  7.283717e-06   3000
## [3,] 1.0 -3.952270e-04 -4.165000e-04   2200
## [4,] 1.5 -1.088907e-03 -1.063000e-03   2100
## [5,] 2.0 -2.042202e-03 -2.002500e-03   1800
## [6,] 2.5 -3.266025e-03 -3.227000e-03   2100
```

```
plot(x=data[,3],y=data[,2],ylim=c(-.05,.01),type="n",ylab="Force (N)",xlab="Command Input (u)",main="Sca
par(new=TRUE)
plot(x=modelagem[,1],y=modelagem[,3],pch=19,ylim=c(min(data[,2]),max(data[,2])),axes=FALSE,ann=FALSE,col
```




```
cor.test(x=modelagem[,1],y=modelagem[,3])
```

```
##
##  Pearson's product-moment correlation
##
## data:  modelagem[, 1] and modelagem[, 3]
## t = -20.266, df = 19, p-value = 2.504e-14
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.9910651 -0.9446189
## sample estimates:
##        cor
## -0.9776429
```

```
u <- modelagem[,1]
y <- modelagem[,3]

X <- u
XSQ <- u**2
XCUB <- u**3

model=lm(y~1+X+XSQ+XCUB)
anova(model)
```

```
## Analysis of Variance Table
##
## Response: y
```

```
##           Df    Sum Sq   Mean Sq    F value    Pr(>F)
## X          1 0.0038311 0.0038311 142765.956 < 2.2e-16 ***
## XSQ        1 0.0001749 0.0001749   6515.892 < 2.2e-16 ***
## XCUB       1 0.0000019 0.0000019     71.405 1.719e-07 ***
## Residuals 17 0.0000005 0.0000000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model)
```

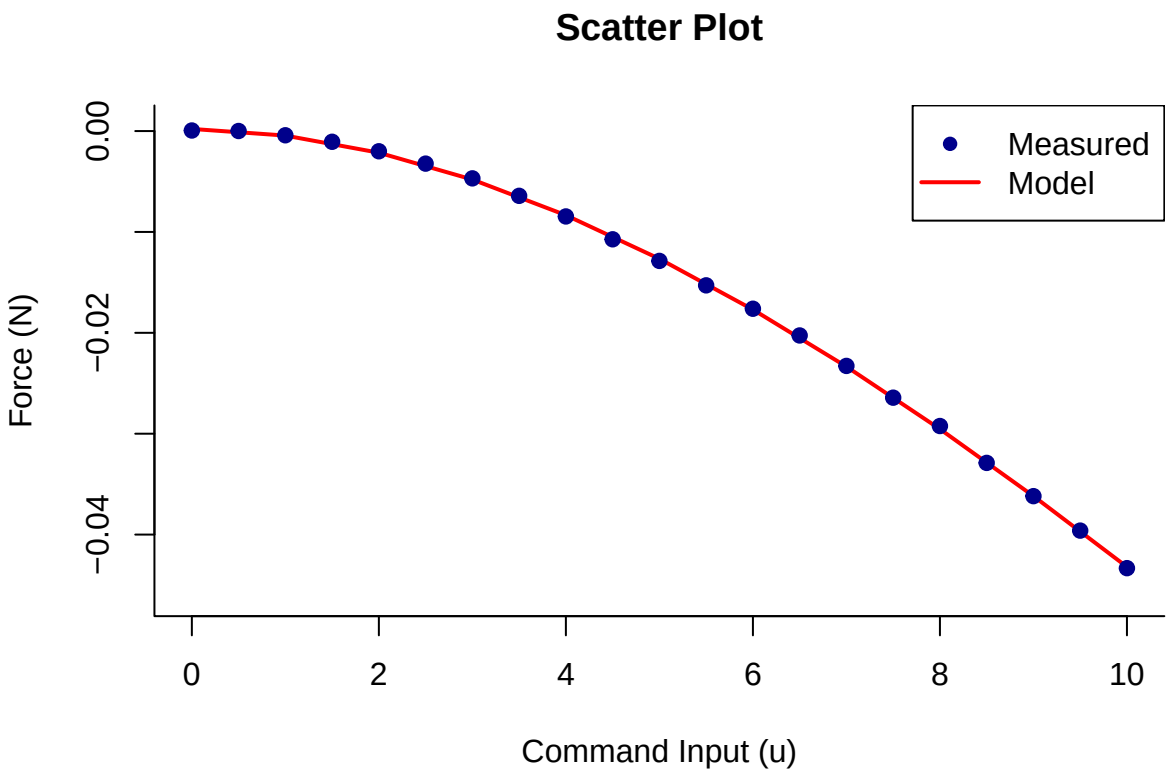
```
##
## Call:
## lm(formula = y ~ 1 + X + XSQ + XCUB)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -3.208e-04 -1.221e-04  2.916e-05  1.079e-04  3.011e-04
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.121e-04  1.208e-04   1.756   0.0972 .
## X           -1.082e-04  1.073e-04  -1.009   0.3272
## XSQ         -5.636e-04  2.529e-05 -22.289 5.06e-14 ***
## XCUB         1.403e-05  1.660e-06   8.450 1.72e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0001638 on 17 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 4.978e+04 on 3 and 17 DF,  p-value: < 2.2e-16
```

```
summ=summary(model)
func <- model$coefficients
print(func)
```

```
##   (Intercept)             X           XSQ          XCUB
##  2.121191e-04 -1.082047e-04 -5.636027e-04  1.403026e-05
```

```
u=0:10
y=as.numeric(func[1])+as.numeric(func[2])*u+as.numeric(func[3])*u**2+as.numeric(func[4])*u**3

plot(x=data[,3],y=data[,2],ylim=c(min(data[,2]),max(data[,2])),type="n",ylab="Force (N)",xlab="Command
par(new=TRUE)
plot(u,y,type="l",ylim=c(min(data[,2]),max(data[,2])),axes=FALSE,ann=FALSE,col="red",lwd=2)
par(new=TRUE)
plot(x=modelagem[,1],y=modelagem[,3],pch=19,ylim=c(min(data[,2]),max(data[,2])),axes=FALSE,ann=FALSE,co
legend("topright",legend = c('Measured','Model'),col = c("darkblue","red"),lty = c(0, 1), lwd=2, pch =c
```




```
error=resid(model)
```

Using r^2 to measure model fit

One important way to assess how well the model fits is to measure the value of r^2, where r is the correlation coefficient.

```
summ$r.squared
```

```
## [1] 0.9998862
```

Noting the conditions

- **The residuals have a normal distribution with mean zero $\varepsilon_i \backsim N(0, \sigma^2)$.**

```
summary(error)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -3.208e-04 -1.221e-04  2.916e-05  0.000e+00  1.079e-04  3.011e-04
```

```
sd(error)
```

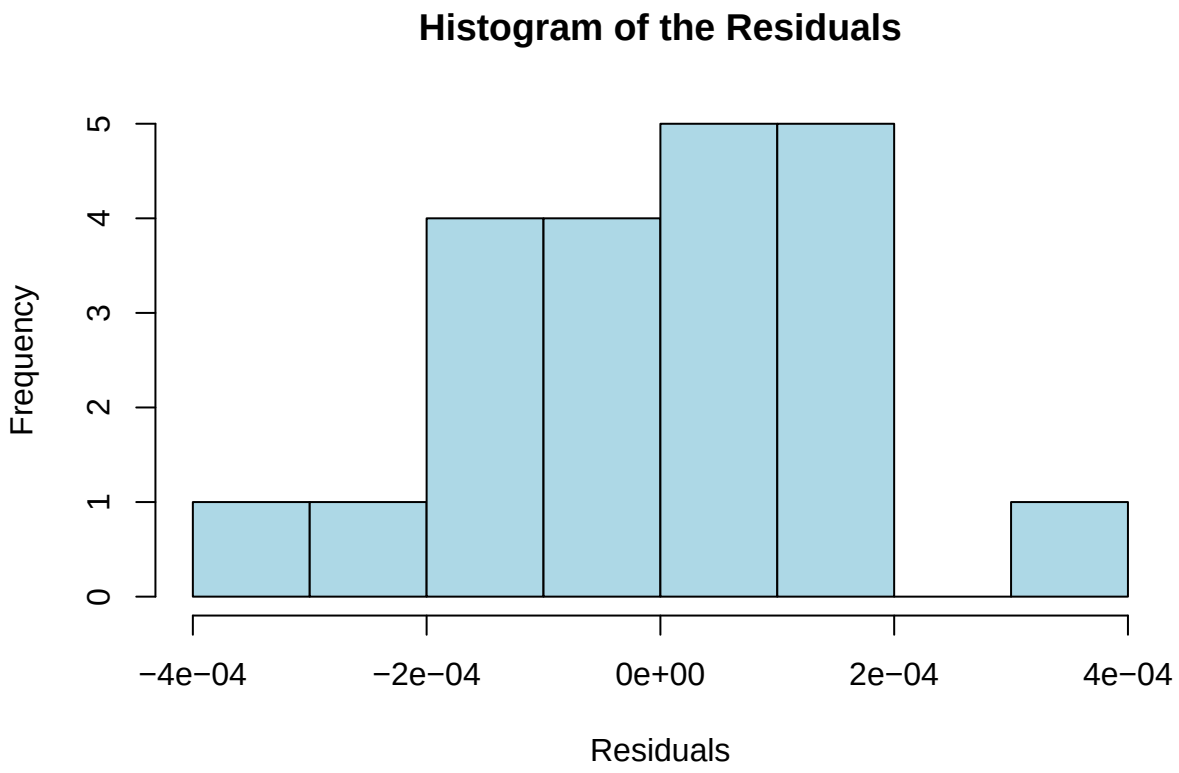
```
## [1] 0.0001510291
```

```
shapiro.test(error)
```

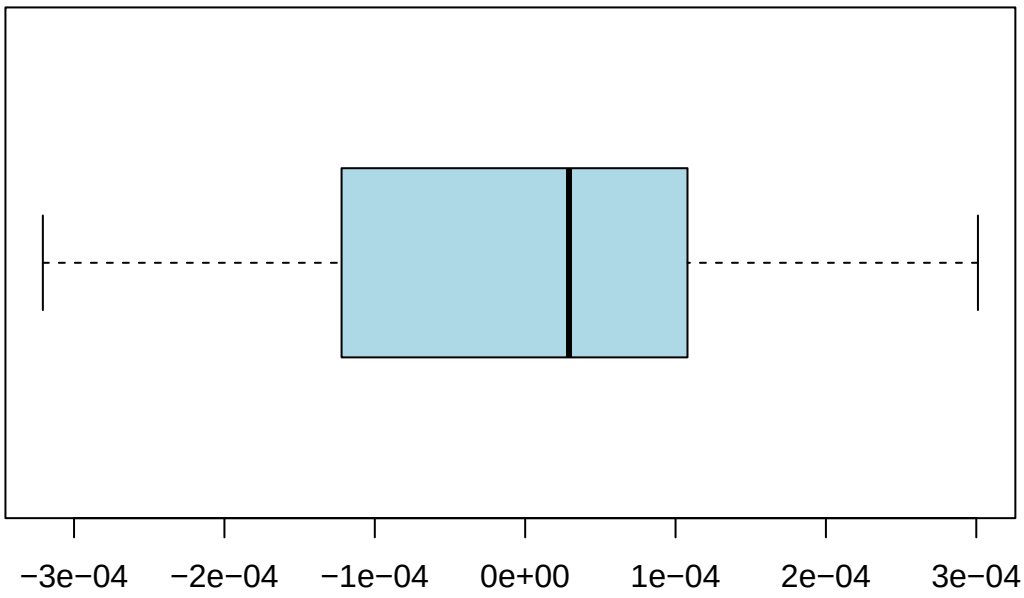
```
##
##  Shapiro-Wilk normality test
##
## data:  error
```

```
## W = 0.98504, p-value = 0.9787
```

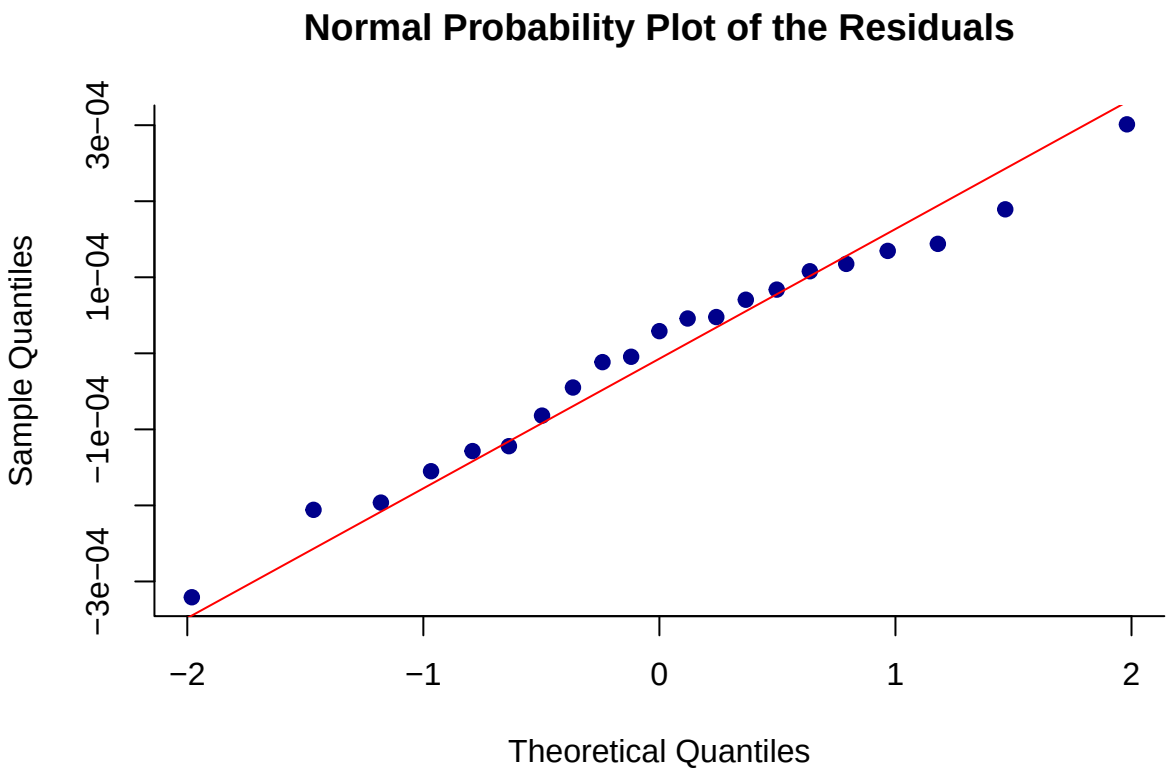
```
hist(error,col="#ADD8E6",main="Histogram of the Residuals",xlab="Residuals")
```




```
boxplot(error,main="Residuals",col="#ADD8E6",horizontal=TRUE)
```




```
qqnorm(error,pch=19,col="darkblue",main="Normal Probability Plot of the Residuals",bty = "l")
qqline(error,col="red")
```



As the condition of normality is met, then residual plot lots of residuals close to zero. The residuals also occur at random some above the line, some below the line.

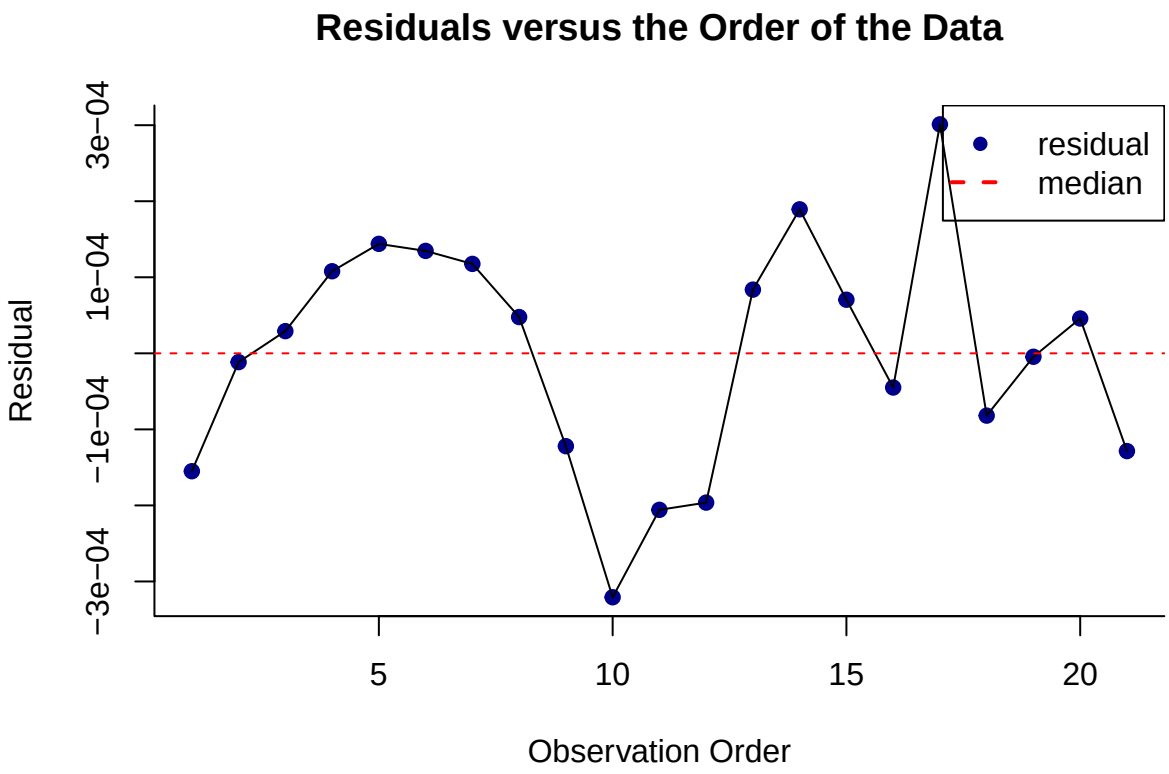
- **The residuals have the same variance for each fitted (predicted) value of $\hat{y}$, $Var(\varepsilon_i) = \sigma^2$.**

```
var(error)
```

```
## [1] 2.28098e-08
```

- **The residuals are independent (don't affect each other).**

```
plot(error,pch=19,col="darkblue",main="Residuals versus the Order of the Data",xlab="Observation Order"
par(new=TRUE)
plot(error,type="l",axes=FALSE,ann=FALSE)
abline(h=0,lty=2,col="red")
legend("topright",legend = c('residual','median'),col = c("darkblue","red"),lty = c(0,2), lwd=2, pch =c
```



Writing into JSON file

```
# creating the list
list1 <- vector(mode="list", length=2)
list1[[1]] <- c("Intercept", "u", "u^2", "u^3")
list1[[2]] <- c(as.numeric(func[1]), as.numeric(func[2]),as.numeric(func[3]), as.numeric(func[4]))

# creating the data for JSON file
jsonData <- toJSON(list1)

# writing into JSON file
write(jsonData, "JSON/motor1.json")

# Give the created file name to the function
motor1 <- fromJSON(file = "JSON/motor1.json")

# Print the result
print(motor1)
```

```
## [[1]]
## [1] "Intercept" "u"         "u^2"       "u^3"
##
## [[2]]
## [1]  2.121191e-04 -1.082047e-04 -5.636027e-04  1.403026e-05
```

```
list2 <- vector(mode="list", length=1)
list2[[1]] <- error
jsonData <- toJSON(list2)
write(jsonData, "JSON/motor1_resi.json")

list3 <- vector(mode="list", length=1)
list3[[1]] <- modelagem[,3]
jsonData <- toJSON(list3)
write(jsonData, "JSON/motor1_median.json")
```

Reference

```
#Citing
citation()
```

```
##
## To cite R in publications use:
##
##   R Core Team (2021). R: A language and environment for statistical
##   computing. R Foundation for Statistical Computing, Vienna, Austria.
##   URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2021},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it
## when using it for data analysis. See also 'citation("pkgname")' for
## citing R packages.
```

```
citation("readr")
```

```
##
## To cite package 'readr' in publications use:
##
##   Hadley Wickham, Jim Hester and Jennifer Bryan (2022). readr: Read
##   Rectangular Text Data. R package version 2.1.3.
##   https://CRAN.R-project.org/package=readr
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {readr: Read Rectangular Text Data},
##     author = {Hadley Wickham and Jim Hester and Jennifer Bryan},
##     year = {2022},
##     note = {R package version 2.1.3},
```

```
##     url = {https://CRAN.R-project.org/package=readr},
##   }
```

```
citation("rjson")
```

```
##
## To cite package 'rjson' in publications use:
##
##   Alex Couture-Beil (2022). rjson: JSON for R. R package version
##   0.2.21. https://CRAN.R-project.org/package=rjson
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {rjson: JSON for R},
##     author = {Alex Couture-Beil},
##     year = {2022},
##     note = {R package version 0.2.21},
##     url = {https://CRAN.R-project.org/package=rjson},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

```
citation("latex2exp")
```

```
##
## To cite package 'latex2exp' in publications use:
##
##   Stefano Meschiari (2022). latex2exp: Use LaTeX Expressions in Plots.
##   R package version 0.9.6. https://CRAN.R-project.org/package=latex2exp
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {latex2exp: Use LaTeX Expressions in Plots},
##     author = {Stefano Meschiari},
##     year = {2022},
##     note = {R package version 0.9.6},
##     url = {https://CRAN.R-project.org/package=latex2exp},
##   }
```

```
### lm {stats}  R Documentation Fitting Linear Models
#Author(s)
#The design was inspired by the S function of the same name described in Chambers (1992). The implement

### References
## Chambers, J. M. (1992) Linear models. Chapter 4 of Statistical Models in S eds J. M. Chambers and T.
##@article{chambers1992linear,
##  title={Linear models. Chapter 4 of statistical models in S},
##  author={Chambers, JM and Hastie, TJ},
##  journal={Wadsworth \& Brooks/Cole},
##  volume={1992},
##  year={1992}
##}
```

```
## Wilkinson, G. N. and Rogers, C. E. (1973). Symbolic descriptions of factorial models for analysis of
##@article{wilkinson1973symbolic,
##  title={Symbolic description of factorial models for analysis of variance},
##  author={Wilkinson, GN and Rogers, CE},
##  journal={Journal of the Royal Statistical Society: Series C (Applied Statistics)},
##  volume={22},
##  number={3},
##  pages={392--399},
##  year={1973},
##  publisher={Wiley Online Library}
##}

## Intermediate Statistics For Dummies - 2007
##@book{rumsey2007intermediate,
##  title={Intermediate statistics for dummies},
##  author={Rumsey, Deborah J},
##  year={2007},
##  publisher={John Wiley \& Sons}
##}
```