BS755 / MA575 Lab 4: Ordinary Least Squares

Shariq Mohammed

Friday, September 29, 2023

Contents

Data Processing and Visualization

In this lab, we will explore Ordinary Least Squares (OLS) regression using an air quality dataset. We'll start by setting up our environment, loading necessary packages, and preparing the data for analysis.

Before we begin, let's set the working directory to where our data is stored.

```
# Set the working directory (e.g., change to the directory where ...
# the data is stored on your computer)
# setwd("your/directory/path")
setwd('~/OneDrive - Boston University/Fall 2023/Teaching/Shariq BS755 MA575/Labs/Lab 04/')
```

We'll need the 'ggplot2' package for data visualization.

```
# Install and load required packages if not already installed
# install.packages("ggplot2")
library(ggplot2)
```

Let's read the data from a CSV file, replacing missing values with `NA` as they are represented as '-200' in the dataset.

```
# Read data from a CSV file using semicolon as delimiter
# Use '..' to move up one directory level to access the 'Lab 03' directory ...
# ... and read the 'AirQualityData.csv' file.
aq_data <- read.csv("../Lab 03/AirQualityData.csv", header=TRUE, as.is=TRUE, sep=';')

# In this data set missing values are stored as -200
# Replace missing data (-200) with NA
aq_data[aq_data == -200] <- NA
# Note that this replaces -200 to NA across all columns in aq_data
```

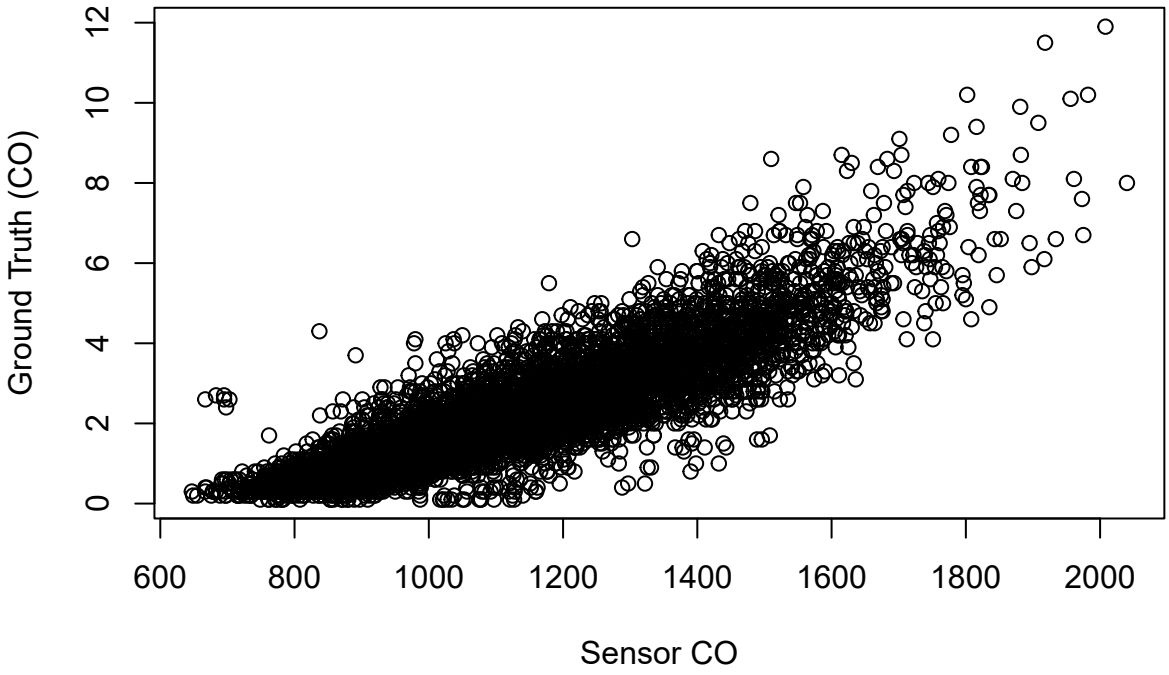
Now, we'll create a new dataset `Dataset` by selecting specific columns of interest from 'aq_data'. These columns will be used for our analysis.

```
# Create a new data frame 'tempdataset' by selecting specific columns from 'aq_data'
# These columns are named in 'tempdataset' as ...
# ... 'Temperature', 'RelativeHumidity', 'SensorCO', and 'GroundCO'
tempdataset <- data.frame(Temperature = aq_data$Temperature,
                          RelativeHumidity = aq_data$RelativeHumidity,
                          SensorCO = aq_data$PT08.S1.CO.,
                          GroundCO = aq_data$CO.GT.)

# Remove rows with missing values (NA)
# Use 'complete.cases' to exclude those rows in 'tempdataset' where ...
# ... there are missing values in atleast one of the four columns, ...
# resulting in a complete dataset without NA values.
Dataset <- tempdataset[complete.cases(tempdataset), ]

# Remove the 'tempdataset' data frame from the R environment
rm(tempdataset)

str(Dataset)
```

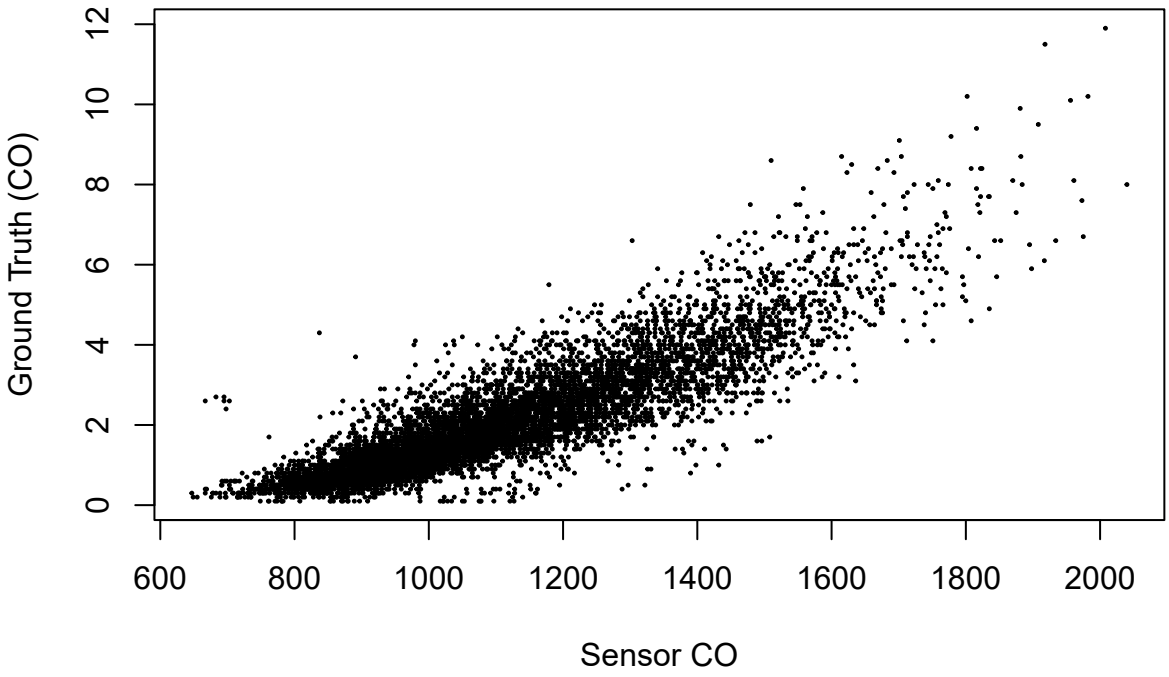
```
## 'data.frame':    7344 obs. of  4 variables:
##  $ Temperature     : num  13.6 13.3 11.9 11 11.2 11.2 11.3 10.7 10.7 10.3 ...
##  $ RelativeHumidity: num  48.9 47.7 54 60 59.6 59.2 56.8 60 59.7 60.2 ...
##  $ SensorCO        : int  1360 1292 1402 1376 1272 1197 1185 1136 1094 1010 ...
##  $ GroundCO        : num  2.6 2 2.2 2.2 1.6 1.2 1.2 1 0.9 0.6 ...
```

We'll create a scatterplot to visualize the relationship between 'SensorCO' and 'GroundCO'.

```
# Create a scatterplot to visualize the relationship between ...
# ... 'SensorCO' and 'GroundCO' variables. 'SensorCO' is plotted ...
# ... on the x-axis, and 'GroundCO' is plotted on the y-axis.
# - 'ylab' sets the label for the y-axis to "Ground Truth (CO)"
# - 'xlab' sets the label for the x-axis to "Sensor CO"
plot(x = Dataset$SensorCO, y = Dataset$GroundCO,
     ylab="Ground Truth (CO)", xlab="Sensor CO")
```



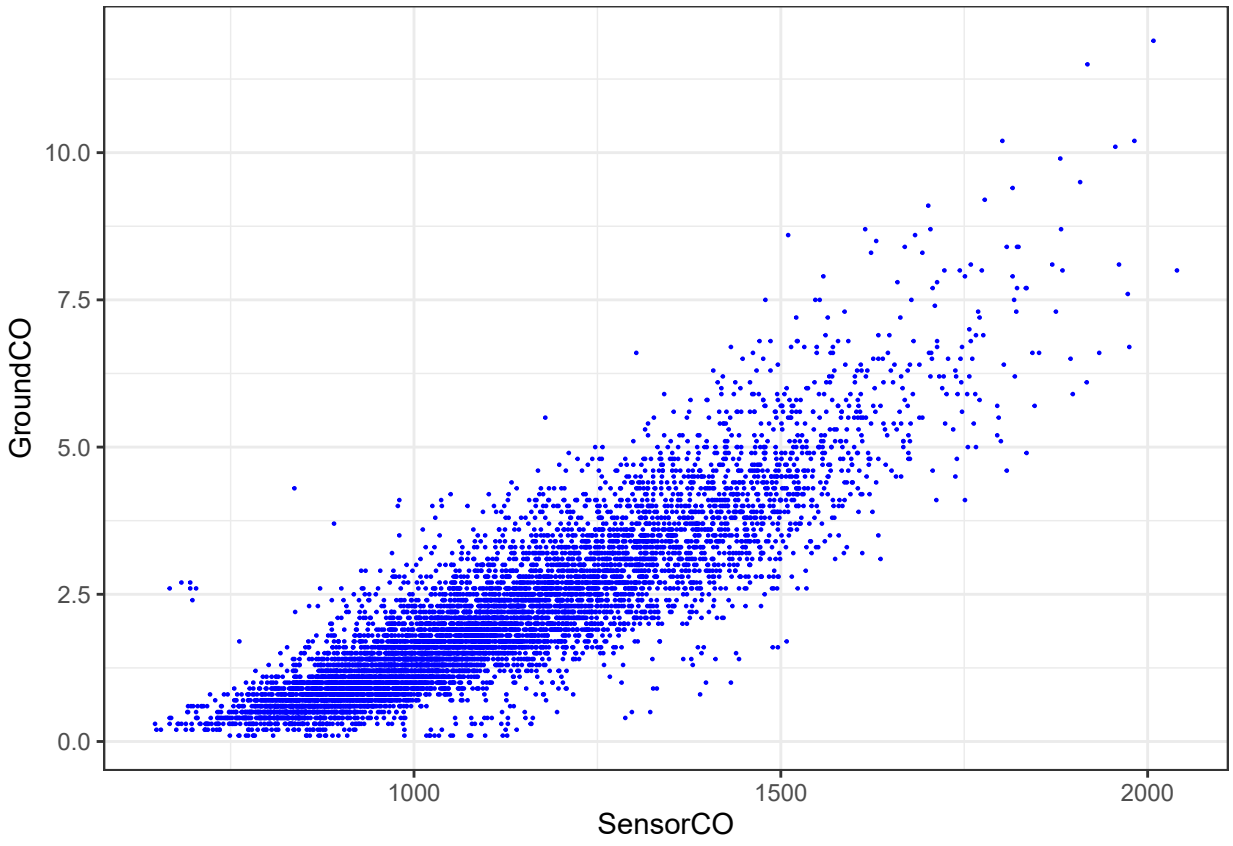

```
# - 'pch' specifies the marker type for points (19 represents filled circles)
# - 'cex' controls the size of the marker (0.2 reduces the size)
# Try different values for these input arguments to see how the plot changes
plot(x = Dataset$SensorCO, y = Dataset$GroundCO,
     ylab="Ground Truth (CO)", xlab="Sensor CO",
     pch=19, cex=0.2)
```



For more advanced visualization, we can use `ggplot2` package.

```
# Create a scatterplot using the ggplot2 package
# - 'Dataset' is the data frame containing the data to be plotted.
```

```
# - aes() specifies the aesthetics of the plot, including the x and y variables.
#   - 'x=SensorCO' sets the x-axis to the 'SensorCO' variable.
#   - 'y=GroundCO' sets the y-axis to the 'GroundCO' variable.
# - geom_point() adds points to the plot to represent the data points.
#   - 'size=0.15' sets the size of the points to 0.15 (adjust as needed).
#   - 'col='blue'' sets the color of the points to blue.
# - theme_minimal() applies a black-and-white theme to the plot background.
ggplot(Dataset, aes(x=SensorCO, y=GroundCO)) +
  geom_point(size=0.15, col='blue') +
  theme_bw()
```



Linear Regression

Now, we'll perform Ordinary Least Squares (OLS) linear regression to model 'GroundCO' as a function of 'SensorCO'.

```
# Perform Ordinary Least Squares (OLS) linear regression.
# Model GroundCO as a function of SensorCO using the lm() function.
# Store the regression model in the variable 'm.ols'.
# The 'data' argument specifies the data frame 'Dataset' where ...
# ... the variables are located.
m.ols <- lm(GroundCO ~ SensorCO, data = Dataset)
```

Summary and Confidence Intervals

The summary output provides information about the model, including coefficients, residuals, R-squared, and more.

```
# Generate a summary of the Ordinary Least Squares (OLS) model
summary(m.ols)
```

```
##
## Call:
## lm(formula = GroundCO ~ SensorCO, data = Dataset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9862 -0.3917 -0.0342  0.3206  4.7067
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.285e+00  4.133e-02  -103.7   <2e-16 ***
## SensorCO     5.776e-03  3.651e-05   158.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6842 on 7342 degrees of freedom
## Multiple R-squared:  0.7731, Adjusted R-squared:  0.7731
## F-statistic: 2.502e+04 on 1 and 7342 DF,  p-value: < 2.2e-16
```

```
# Summary Output Information:
# - Call: Exact model formula used, including response and predictors.
# - Residuals: Statistics on model residuals (differences between ...
#   ... observed and predicted values).
# - Coefficients: Estimated values of coefficients, standard errors, ...
#   ... t-values, p-values, and the significance codes.
# - Residual Standard Error: Quantifies the average prediction error.
# - R-squared: Measures the proportion of response variable variance ...
#   ... explained by the model. Includes adjusted R-squared.
# - F-statistic: F-statistics, degrees of freedom p-value.
```

We'll calculate 95% confidence intervals for the model coefficients.

```
# Calculates 95% confidence intervals for the model coefficients.
confint(m.ols, level=0.95)
```

```
##                    2.5 %       97.5 %
## (Intercept) -4.365864616 -4.203829785
## SensorCO     0.005704283  0.005847436
```

```
# Provides lower and upper bounds at the 95% confidence level.
# Width of the interval indicates the level of uncertainty in ...
# ... the coefficient's estimate. That is, narrow intervals indicate ...
# ... more precise coefficient estimates, while wider intervals ...
# ... imply greater uncertainty.

# Round numerical values to a specified number of decimal digits.
round(confint(m.ols, level=0.95), digits=4)
```

```
##               2.5 %  97.5 %
```

```
## (Intercept) -4.3659 -4.2038
## SensorCO     0.0057  0.0058
```

Variance-Covariance Matrix

The variance-covariance matrix is essential for assessing the uncertainty and relationships among coefficients.

```
# Variance-covariance matrix for the coefficient estimates in the model.
vcov(m.ols)
```

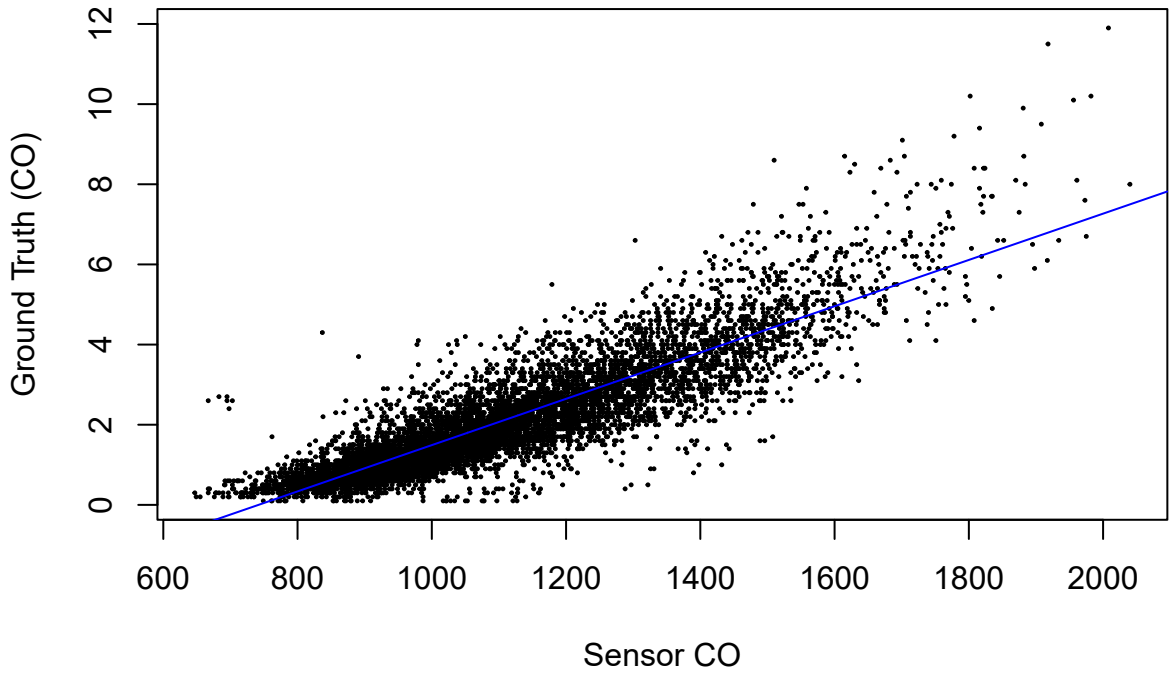
```
##                (Intercept)      SensorCO
## (Intercept)  1.708116e-03 -1.480638e-06
## SensorCO    -1.480638e-06  1.333211e-09
```

```
# Diagonal elements of the variance-covariance matrix represent the ...
# ... variance of each coefficient estimate. Off-diagonal elements represent ...
# ... covariances between coefficient pairs. This matrix is crucial for ...
# ... assessing the uncertainty and relationships among the coefficients, ...
# ... which is vital in hypothesis testing and making inferences about the model.
# Note: For example, the 'e-09' part indicates that the number should be ...
# ... multiplied by 10 to the power of -9.
```

Scatterplot with OLS Regression Line

Let's visualize the OLS regression line on the scatterplot using base R.

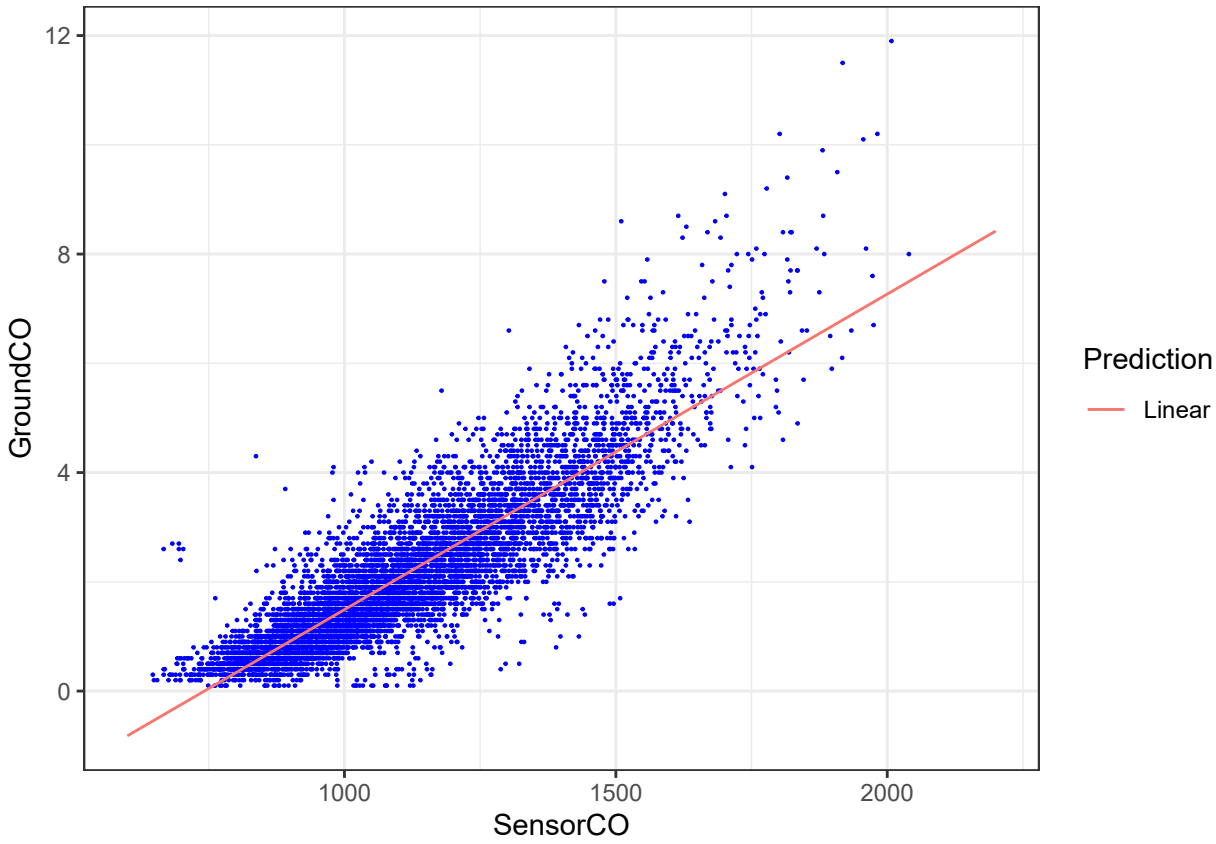
```
# Scatterplot of SensorCO vs. GroundCO with the OLS regression line.
# - 'abline()' is used to add a line to the current plot.
# - 'lsfit()' performs a least-squares regression fit line for the specified variables.
plot(Dataset$SensorCO, Dataset$GroundCO,
     ylab="Ground Truth (CO)", xlab="Sensor CO",
     pch=19, cex=0.2)
abline(lsfit(x=Dataset$SensorCO, y=Dataset$GroundCO), col="blue")
```



```
# - A straight line (OLS fit) is overlaid on the scatterplot. This line ...
# ... visually illustrates the relationship between the two variables.
```

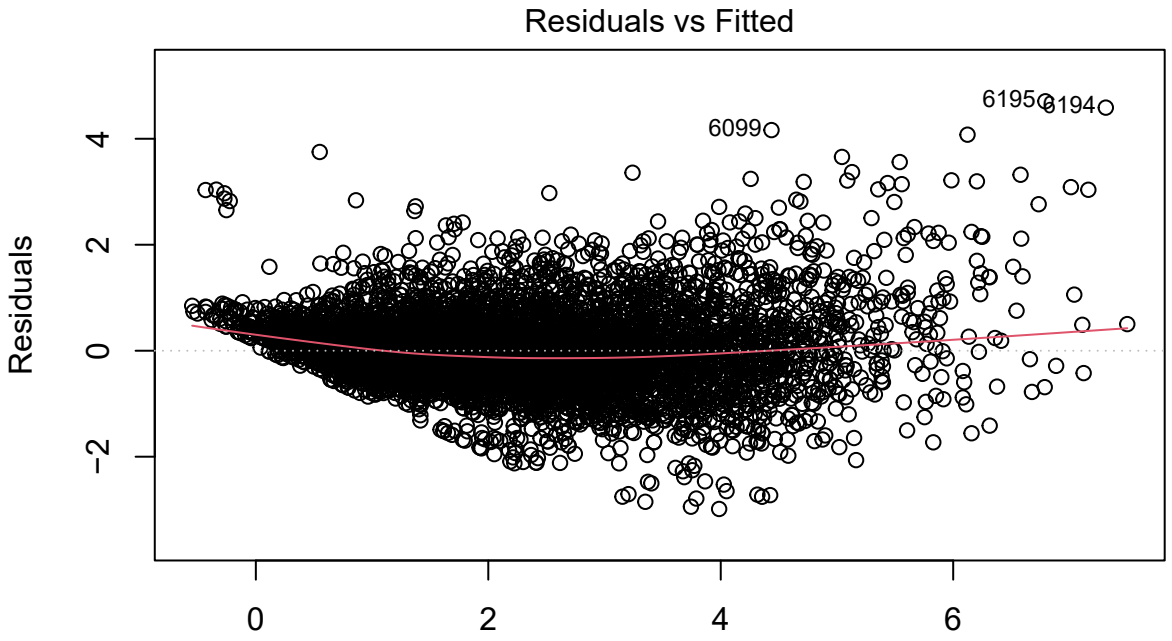
Alternative visualization using ggplot2 package:

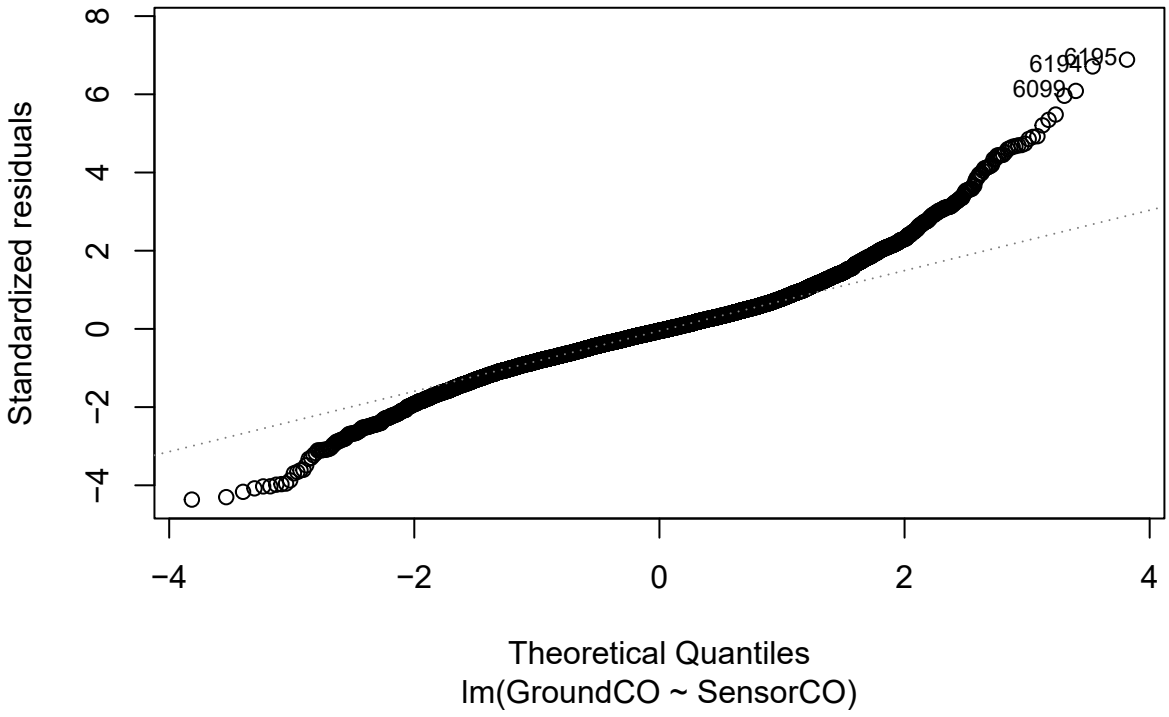
```
# Generate sequences of numbers 'from' a starting value 'to' an ...
# ... ending value. 'length.out' specifies the length of the sequence.
# Here length.out is set to the length of the 'SensorCO' column ...
# ... in the 'Dataset' data frame.
SensorCONew <- seq(from=600, to=2200, length.out=length(Dataset$SensorCO))

# Generate predictions/fitted-values from the model.
# - 'newdata' specifies the data frame containing the predictor ...
#   ... variable 'SensorCO' for which we want to make predictions.
predLinear <- predict(m.ols, newdata=data.frame(SensorCO=SensorCONew))
# 'predLinear' contains the predicted values of the response variable ...
# ... (in this case, 'GroundCO') based on the provided 'SensorCO' values ...
# ... and the estimated coefficients of the OLS model.

# Create the scatterplot with OLS regression line using ggplot2
# - 'geom_line()' adds a line plot layer using 'SensorCONew' and ...
#   ... 'predLinear' for x and y, respectively.
# - 'scale_color_discrete(name="Prediction", labels=c("Linear"))' ...
# ... sets the color scale for the plot, naming it "Prediction" ...
# ... with a single label "Linear."
ggplot(Dataset, aes(x=SensorCO, y=GroundCO)) +
  geom_point(size=0.15, col='blue') +
  geom_line(aes(x=SensorCONew, y=predLinear, color="blue")) +
  scale_color_discrete(name="Prediction", labels=c("Linear")) +
  theme_bw()
```

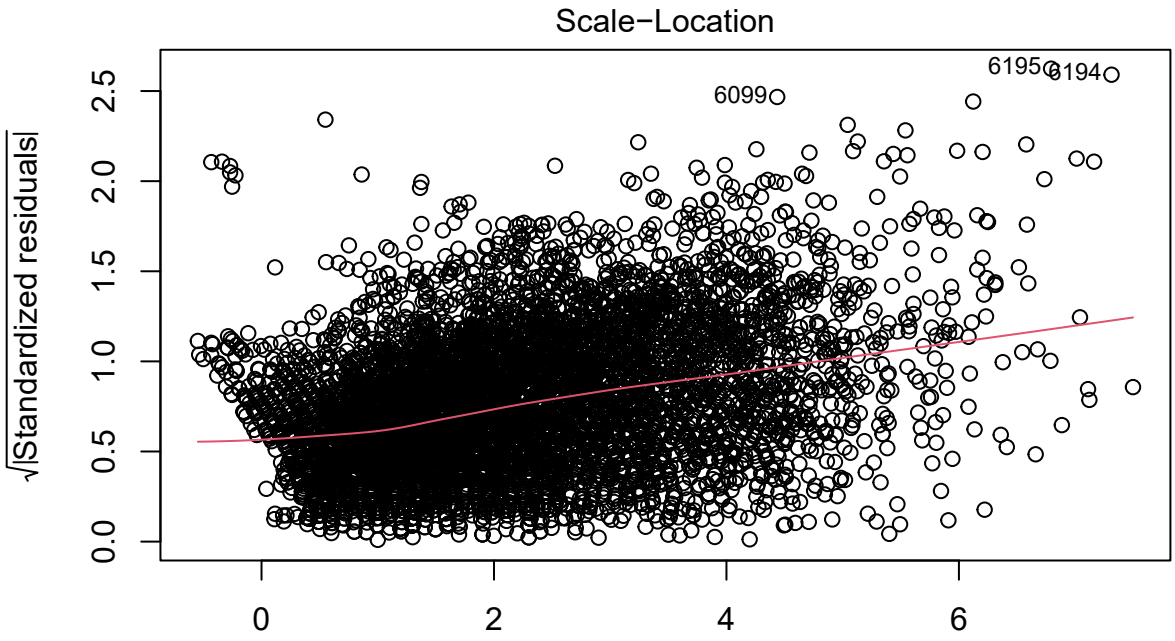


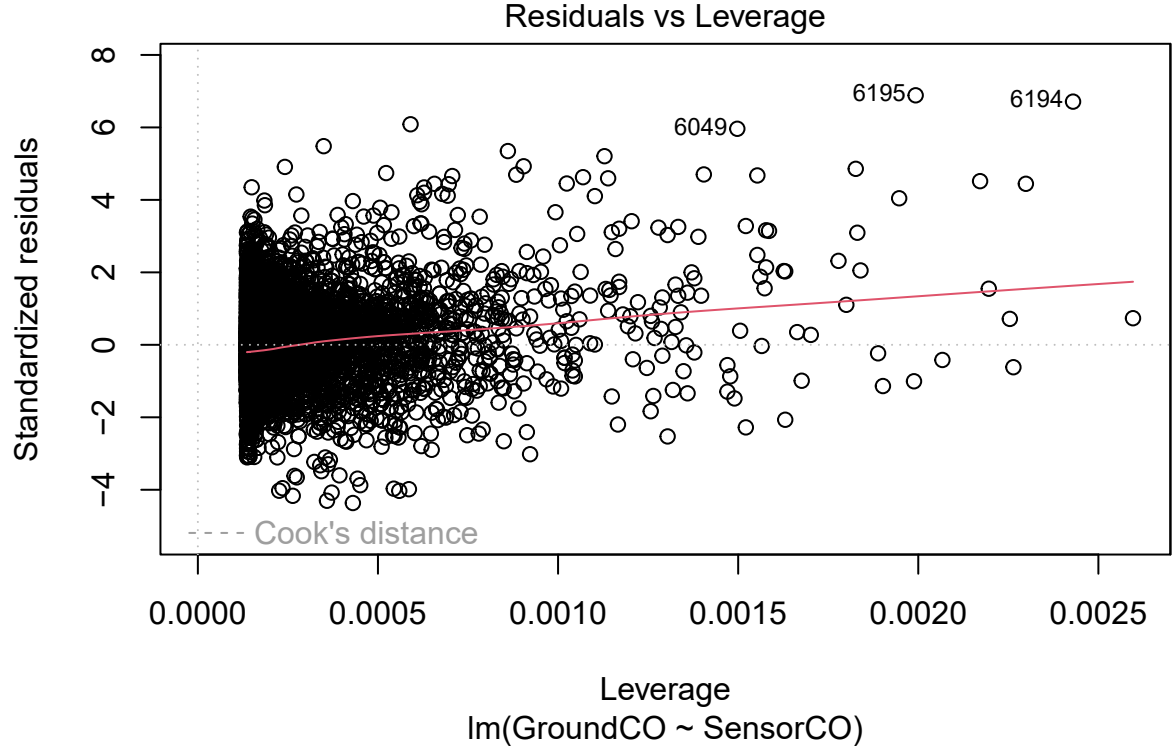
Model Diagnostics

```
# Perform model diagnostics
# 'plot' function returns a set of diagnostic plots to assess the assumptions ...
# ... and goodness of fit of a linear model fit using 'lm' function.
plot(m.ols)
```


Residuals vs Fitted
Residuals
Fitted values
lm(GroundCO ~ SensorCO)
6099
6195
6194


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(GroundCO ~ SensorCO)
6194
6195
6099



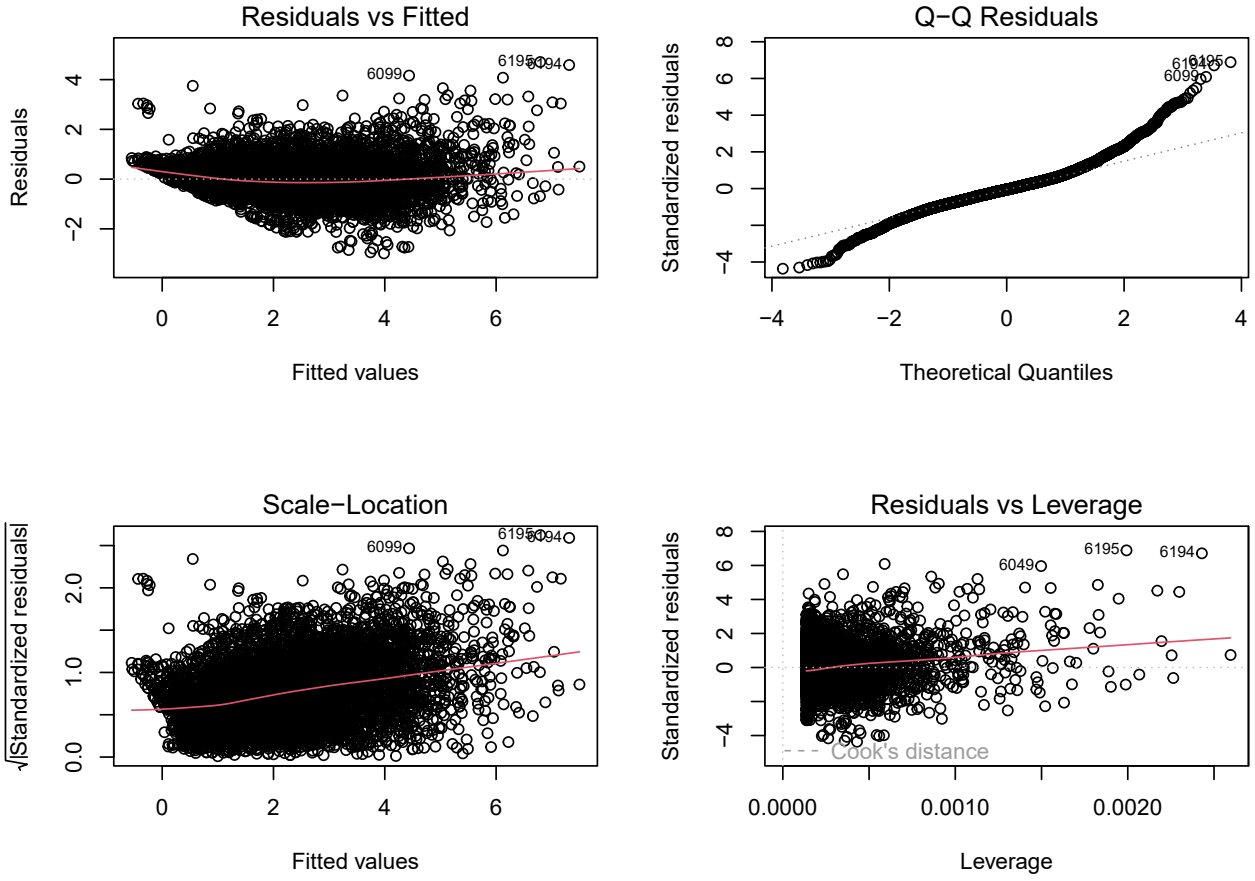



```
# Residuals vs. Fitted Values Plot: Shows relationship between the ...
# ... observed residuals (differences between the actual and ...
# ... fitted values) and the fitted values from the linear model. It ...
# ... helps to check for heteroscedasticity (whether the spread of ...
# ... residuals changes with fitted values) and to identify outliers.
# Normal Q-Q Plot: Quantile-quantile (Q-Q) plot that assesses whether ...
```

```
# ... the residuals follow a normal distribution. Points along a ...
# ... straight line suggests that the residuals are approximately ...
# ... normally distributed.
# Scale-Location (Spread-Location) Plot: To check for homoscedasticity ...
# ... (constant variance of residuals), linearity and outliers similar to ...
# ... residuals vs fitted values plot.
# Residuals vs. Leverage Plot: To detect potential outliers/influential observations.
```

Plotting the four diagnostic plots into one figure. You can also specify the figure height and width for the plot in a chunk.

```
# Set the layout of the plotting device to a 2x2 grid
# 'mfrow' parameter specifies the number of rows and columns ...
# ... in which to arrange multiple plots
par(mfrow = c(2, 2))
# Create a plot of the object 'm.ols' (presumably the result of a linear regression)
plot(m.ols)
```




```
# Reset the layout of the plotting device to the default (1x1)
par(mfrow = c(1, 1))
```

Are the model assumptions satisfied? Is this a good model?

Quadratic Regression

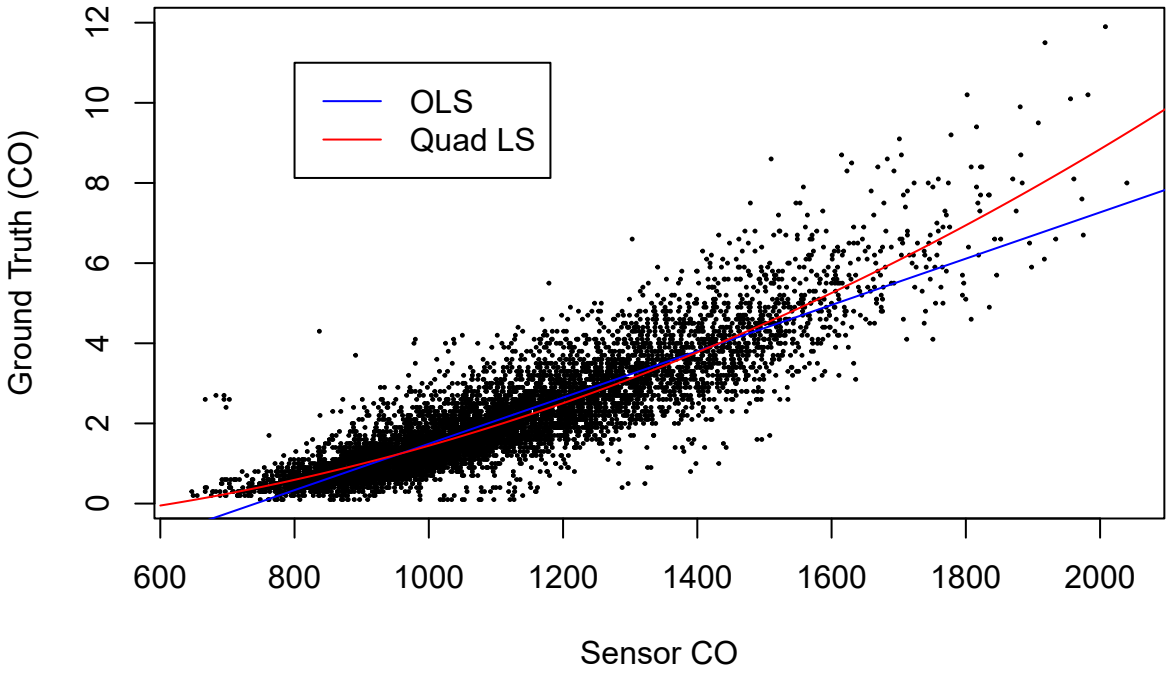
We'll also explore Quadratic Least Squares (Quad LS) regression to allow for a curved fit.

```r
# Perform Quadratic Least Squares (Quad LS) regression
# - 'GroundCO ~ SensorCO + I(SensorCO^2)' indicates that we are modeling ...
#   ... 'GroundCO' as a function of 'SensorCO' and 'SensorCO^2'.
# - This model represents a quadratic relationship between 'GroundCO' ...
#   ... and 'SensorCO', allowing for a curved fit. It estimates the ...
#   ... coefficients of the quadratic equation that best fits the data.
m.quadls <- lm(GroundCO ~ SensorCO + I(SensorCO^2), data = Dataset)

# Generate a summary of the Quadratic Least Squares (Quad LS) model
summary(m.quadls)
```

```
##
## Call:
## lm(formula = GroundCO ~ SensorCO + I(SensorCO^2), data = Dataset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9977 -0.3653 -0.0478  0.2939  4.0370
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -7.159e-01  1.709e-01  -4.189 2.84e-05 ***
## SensorCO      -4.603e-04  2.925e-04  -1.574    0.116
## I(SensorCO^2)  2.620e-06  1.220e-07  21.482  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6637 on 7341 degrees of freedom
## Multiple R-squared:  0.7866, Adjusted R-squared:  0.7865
## F-statistic: 1.353e+04 on 2 and 7341 DF,  p-value: < 2.2e-16
```

Scatterplot with Regression Lines

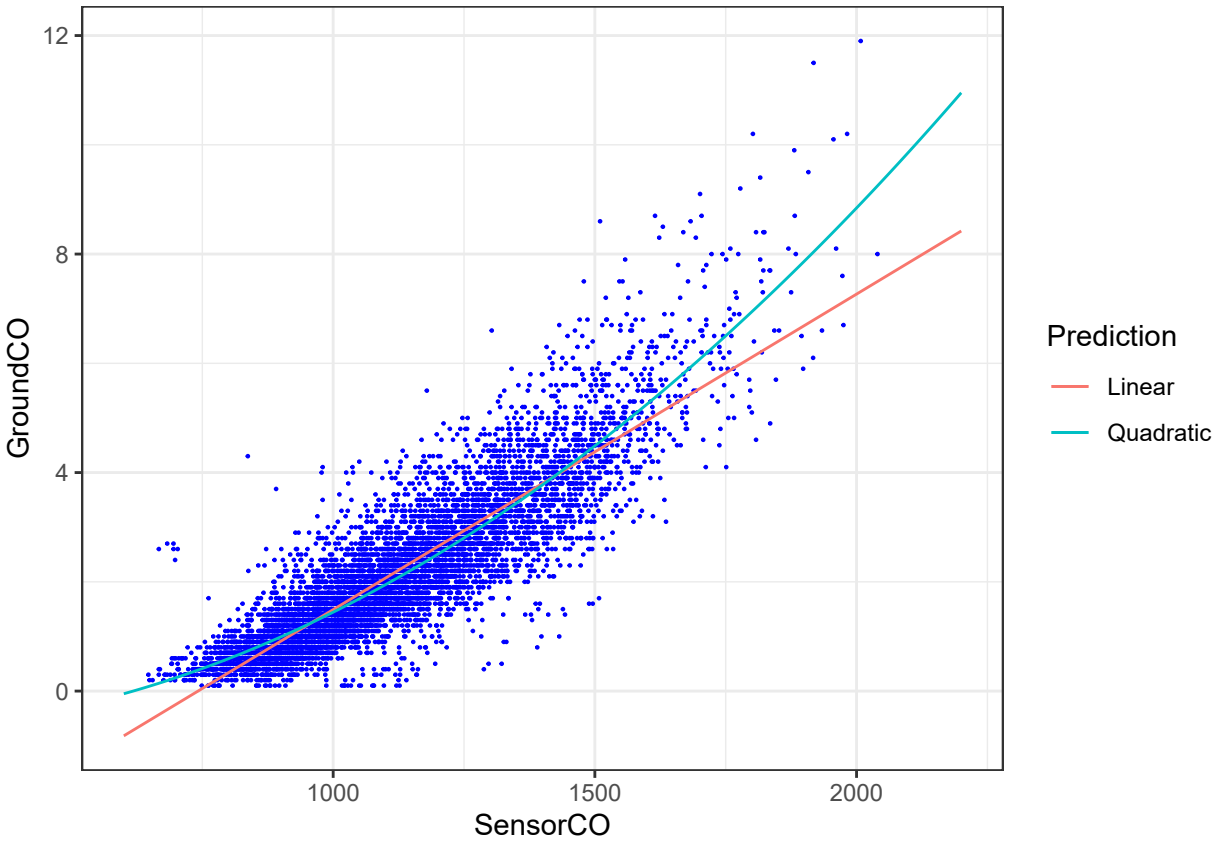
```r
# Overlay OLS and Quad LS regression line on the scatterplot
# - 'lines()' is a function in R used to add lines to a plot.
# - 'SensorCONew' is the x-axis values.
# - 'predict(m.ols, newdata=data.frame(SensorCO=SensorCONew))' calculates ...
#   ... the predicted values of the response variable (in this case, 'GroundCO') ...
#   ... based on the provided 'SensorCO' values and the coefficients ...
# ... of the linear regression model ('m.ols').
# - The line connects the predicted values of 'GroundCO' corresponding ...
#   ... to the 'SensorCO' values in 'SensorCONew'.
plot(Dataset$SensorCO, Dataset$GroundCO,
     ylab="Ground Truth (CO)", xlab="Sensor CO",
     pch=19, cex=0.2)
lines(x=SensorCONew,
      y=predict(m.ols, newdata=data.frame(SensorCO=SensorCONew)),
      col="blue")
lines(x=SensorCONew,
      y=predict(m.quadls, newdata=data.frame(SensorCO=SensorCONew)),
      col="red")
legend(x=800, y=11, legend=c("OLS", "Quad LS"), col=c("blue", "red"), lty=1, cex=1)
```




```
# - 'legend()' is a function used to add a legend to a plot in R.
# - The first two arguments, '800' and '11', specify the x and y coordinates ...
#   ... where the legend will be placed on the plot.
# - 'legend=c("OLS", "Quad LS")' provides the labels for the legend one ...
#   ... for each plot element.
# - 'col' specifies the colors associated with each label in the legend.
```

Alternative visualization using `ggplot2` package:

```
# Create the same plot using ggplot2
predQuad <- predict(m.quadls, newdata=data.frame(SensorCO=SensorCONew))
ggplot(Dataset, aes(x=SensorCO, y=GroundCO)) +
  geom_point(size=0.15, col='blue') +
  geom_line(aes(x=SensorCONew, y=predLinear, color="blue")) +
  geom_line(aes(x=SensorCONew, y=predQuad, color="red")) +
  scale_color_discrete(name="Prediction", labels=c("Linear", "Quadratic")) +
  theme_bw()
```



In this lab, we explored Ordinary Least Squares (OLS) and Quadratic Least Squares (Quad LS) regression techniques to analyze the air quality dataset. We visualized the relationships and assessed the model fit, providing valuable insights for further analysis.

As an exercise, you can perform linear regression on your project data, selecting a response and covariate of interest.