Sound Sensor Correlation Analysis for 2 open sensors

KRIISH HATE

2025-03-17

```
knitr::opts_chunk$set(echo = TRUE, message = TRUE, warning = FALSE)
# ================================
# R Script: Sound Sensor Analysis
# ================================
# This script reads sound sensor data from a TXT file,
# converts raw ADC values to dB, plots a correlation graph with a best-fit
line,
# and computes linear regression parameters, displaying them on the graph.

# ------------------------------
# Load necessary libraries
# ------------------------------
# The ggplot2 library is used to create visualizations
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
# ------------------------------
# Read the data from the TXT file
# ------------------------------
# Define the file path for the dataset
file_path <- "LOG.TXT"  # Ensure this file is in the working directory

# Read the file into a data frame
# read.table() reads a comma-separated file without a header
# The resulting data frame will have three columns: Timestamp, Sensor1, and
Sensor2
data <- read.table(file_path, header = FALSE, sep = ",", col.names =
c("Timestamp", "Sensor1", "Sensor2"))

# ------------------------------
# Define constants
# ------------------------------
# Set the reference voltage used for ADC conversion
VREF <- 6.14

# ------------------------------
# Convert raw ADC values to dB
# ------------------------------
# Convert raw ADC values to voltage
```

```
# Formula: ( (read value) / (ADC max value) ) * voltage reference
data$Sensor1_Voltage <- ((data$Sensor1) / (32768.0)) * VREF
data$Sensor2_Voltage <- ((data$Sensor2) / (32768.0)) * VREF

# ------------------------------
# Convert voltage to decibels (dB)
# ------------------------------
# Formula: dB = Voltage * 50 (Scaling factor)
data$Sensor1_dB <- data$Sensor1_Voltage * 50.0
data$Sensor2_dB <- data$Sensor2_Voltage * 50.0
```

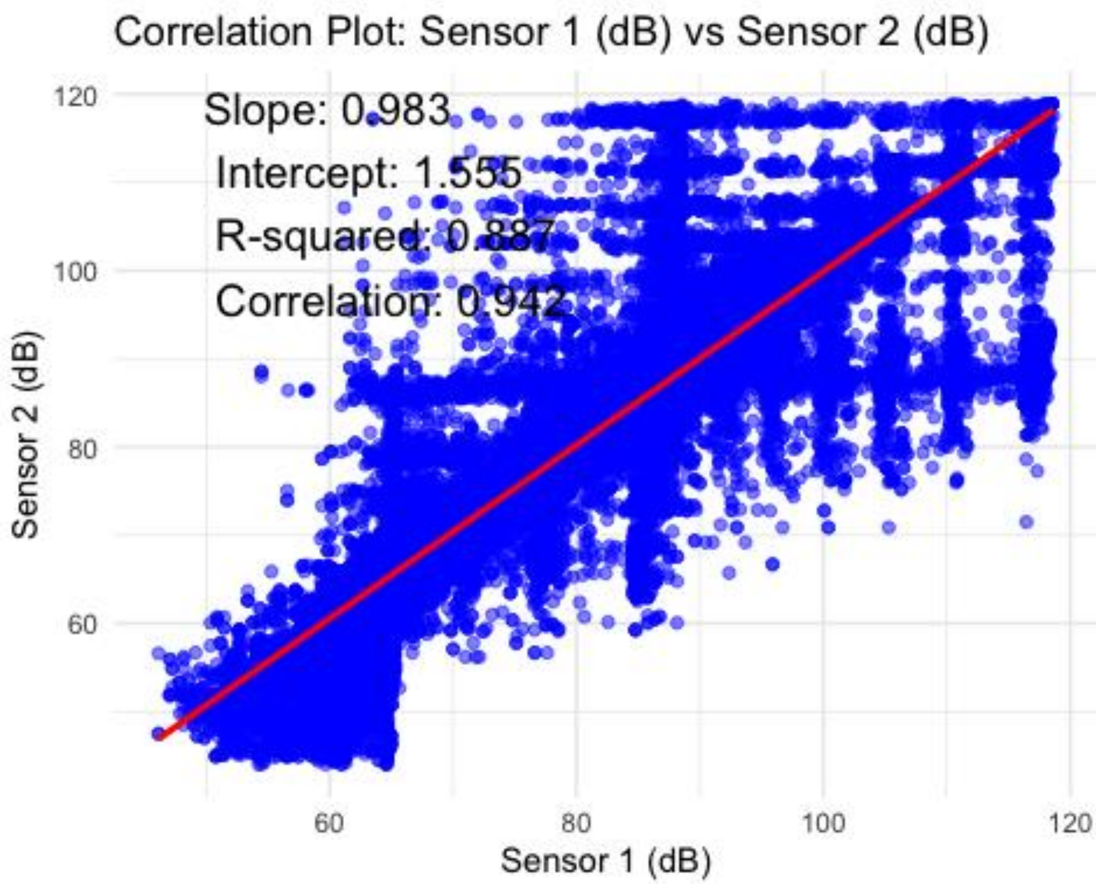
```
knitr::opts_chunk$set(echo = TRUE, message = TRUE, warning = FALSE)
# ===============================
# Plot Sensor 1 vs. Sensor 2 dB Values with Statistics
# ===============================
# ------------------------------
# Compute linear regression
# ------------------------------
# Fit a linear regression model where:
# Sensor2_dB = slope * Sensor1_dB + intercept
model <- lm(Sensor2_dB ~ Sensor1_dB, data = data)

# Display the regression summary in the console
summary(model)
```

```
##
## Call:
## lm(formula = Sensor2_dB ~ Sensor1_dB, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.568  -0.480   0.021   0.472  53.177
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.5548143  0.0487270   31.91   <2e-16 ***
## Sensor1_dB  0.9833464  0.0005877 1673.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.708 on 356718 degrees of freedom
## Multiple R-squared:  0.887,	Adjusted R-squared:  0.887
## F-statistic: 2.8e+06 on 1 and 356718 DF,  p-value: < 2.2e-16
```

```
# Extract regression parameters
# Extract the slope (coefficient for Sensor1_dB)
slope <- coef(model)[2]
# Extract the y-intercept
intercept <- coef(model)[1]
# Compute the R-squared value, indicating model fit
```

```r
r_squared <- summary(model)$r.squared
# Compute correlation coefficient as square root of R-squared
correlation_coefficient <- sqrt(r_squared)

# ------------------------------
# Create text annotation for the plot
# ------------------------------
# Format the regression parameters as a text string to display on the plot
regression_text <- paste(
  "Slope:", round(slope, 3), "\n",
  "Intercept:", round(intercept, 3), "\n",
  "R-squared:", round(r_squared, 3), "\n",
  "Correlation:", round(correlation_coefficient, 3)
)

# ------------------------------
# Determine annotation position on the plot
# ------------------------------
# Set the x-position for annotation as 5% from the left side
text_x_pos <- min(data$Sensor1_dB) + (max(data$Sensor1_dB) -
min(data$Sensor1_dB)) * 0.05
# Set the y-position for annotation as 85% from the bottom
text_y_pos <- min(data$Sensor2_dB) + (max(data$Sensor2_dB) -
min(data$Sensor2_dB)) * 0.85

# ------------------------------
# Create a scatter plot with best-fit line
# ------------------------------
# Generate a scatter plot using ggplot2
plot <- ggplot(data, aes(x = Sensor1_dB, y = Sensor2_dB)) +
  # Add scatter points in blue with 50% transparency
  geom_point(alpha = 0.5, color = "blue") +
  # Add a linear regression line (best-fit line) in red
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  # Annotate the plot with regression parameters
  annotate("text", x = text_x_pos, y = text_y_pos, label = regression_text,
size = 5, hjust = 0, color = "black") +
  # Set the plot title
  ggtitle("Correlation Plot: Sensor 1 (dB) vs Sensor 2 (dB)") +
  # Set x-axis label
  xlab("Sensor 1 (dB)") +
  # Set y-axis label
  ylab("Sensor 2 (dB)") +
  # Use a minimal theme for cleaner visualization
  theme_minimal()

# ------------------------------
# Display the plot
# ------------------------------
```

```r
# Print the scatter plot with best-fit line
print(plot)

## `geom_smooth()` using formula = 'y ~ x'
```




```r
# -------------------------------
# Save the plot as a JPEG file
# -------------------------------
ggsave("Sensor_Correlation_dB.jpeg", plot = plot, width = 6, height = 4, dpi
= 300)

## `geom_smooth()` using formula = 'y ~ x'

# -------------------------------
# Print regression results to the console
# -------------------------------
# Display regression results with explanatory text
cat("\n--- Regression Results ---\n")

##
## --- Regression Results ---

cat("Slope:", slope, "\n")

## Slope: 0.9833464
```

```r
cat("Intercept:", intercept, "\n")
```

```
## Intercept: 1.554814
```

```r
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.8869885
```

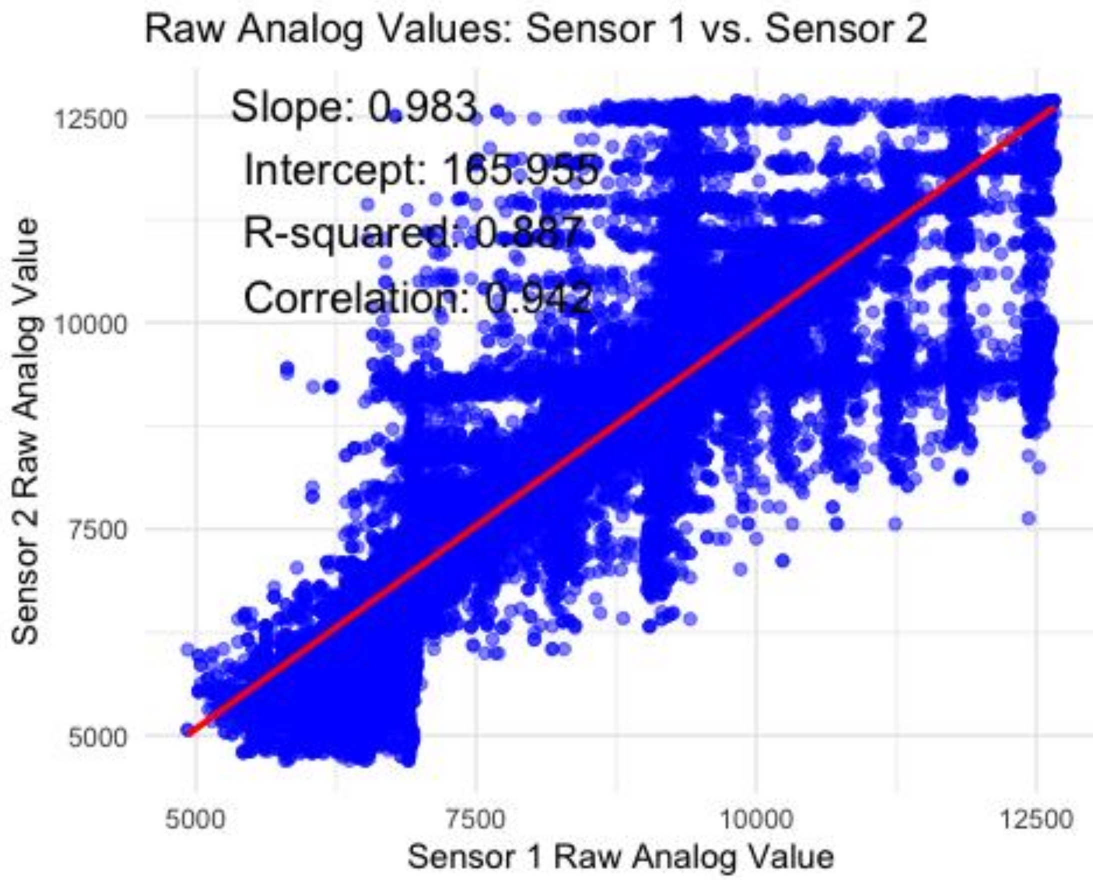
```r
cat("Correlation Coefficient:", correlation_coefficient, "\n")
```

```
## Correlation Coefficient: 0.9418007
```

```r
knitr::opts_chunk$set(echo = TRUE, message = TRUE, warning = FALSE)
# ================================
# Plot Raw Sensor 1 vs. Sensor 2 Analog Values with Statistics
# ================================
# -------------------------------
# Compute Linear Regression
# -------------------------------
# Fit a linear regression model where:
# Sensor2 = slope * Sensor1 + intercept
model <- lm(Sensor2 ~ Sensor1, data = data)

# Extract regression parameters
slope <- coef(model)[2]  # The slope of the regression line
intercept <- coef(model)[1]  # The y-intercept of the regression line
r_squared <- summary(model)$r.squared  # R-squared value, indicating model
fit
correlation_coefficient <- sqrt(r_squared)  # Square root of R-squared gives
correlation coefficient

# -------------------------------
# Create text annotation for the plot
# -------------------------------
# Format the regression parameters as a text string to display on the plot
regression_text <- paste(
  "Slope:", round(slope, 3), "\n",
  "Intercept:", round(intercept, 3), "\n",
  "R-squared:", round(r_squared, 3), "\n",
  "Correlation:", round(correlation_coefficient, 3)
)

# Determine annotation position for displaying text on the plot
text_x_pos <- min(data$Sensor1) + (max(data$Sensor1) - min(data$Sensor1)) *
0.05  # 5% from the left
text_y_pos <- min(data$Sensor2) + (max(data$Sensor2) - min(data$Sensor2)) *
0.85  # 85% from the bottom

# -------------------------------
# Create a scatter plot of raw analog values with best-fit line
# -------------------------------
```

```r
plot <- ggplot(data, aes(x = Sensor1, y = Sensor2)) +
  # Add scatter points in blue with 50% transparency
  geom_point(alpha = 0.5, color = "blue") +
  # Add a linear regression line (best-fit line) in red
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  # Annotate the plot with regression parameters
  annotate("text", x = text_x_pos, y = text_y_pos, label = regression_text,
size = 5, hjust = 0, color = "black") +
  # Set the plot title
  ggtitle("Raw Analog Values: Sensor 1 vs. Sensor 2 ") +
  # Set x-axis label
  xlab("Sensor 1 Raw Analog Value") +
  # Set y-axis label
  ylab("Sensor 2 Raw Analog Value") +
  # Use a minimal theme for a clean layout
  theme_minimal()

# ------------------------------
# Display the plot
# ------------------------------
# Print the scatter plot with best-fit line
print(plot)
```

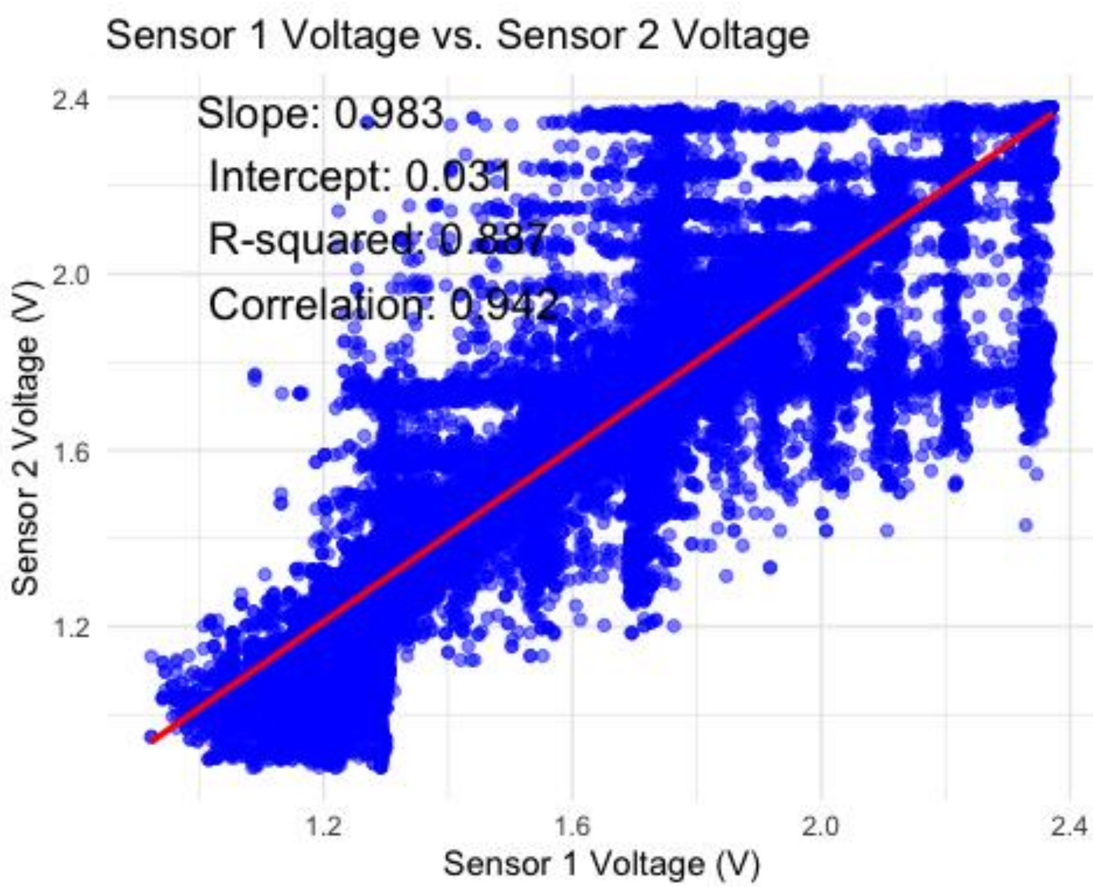
```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
# -------------------------------
# Save the plot as a JPEG file
# -------------------------------
ggsave("Sensor_Correlation_rawlongvalue.jpeg", plot = plot, width = 6, height
= 4, dpi = 300)

## `geom_smooth()` using formula = 'y ~ x'

# -------------------------------
# Print regression results to the console
# -------------------------------
# Display regression results with explanatory text
cat("\n--- Regression Results ---\n")

##
## --- Regression Results ---

cat("Slope:", slope, "\n")

## Slope: 0.9833464

cat("Intercept:", intercept, "\n")

## Intercept: 165.9549
```

```r
cat("R-squared:", r_squared, "\n")

## R-squared: 0.8869885

cat("Correlation Coefficient:", correlation_coefficient, "\n")

## Correlation Coefficient: 0.9418007

knitr::opts_chunk$set(echo = TRUE, message = TRUE, warning = FALSE)
# ===============================
# Plot Sensor 1 Voltage vs. Sensor 2 Voltage with Statistics
# ===============================
# ------------------------------
# Compute Linear Regression
# ------------------------------
# Fit a linear regression model where:
# Sensor2 Voltage = slope * Sensor1 Voltage + intercept
model <- lm(Sensor2_Voltage ~ Sensor1_Voltage, data = data)

# Extract regression parameters
slope <- coef(model)[2]  # The slope of the regression line
intercept <- coef(model)[1]  # The y-intercept of the regression line
r_squared <- summary(model)$r.squared  # R-squared value, indicating model
fit
correlation_coefficient <- sqrt(r_squared)  # Square root of R-squared gives
correlation coefficient

# ------------------------------
# Create text annotation for the plot
# ------------------------------
# Format the regression parameters as a text string to display on the plot
regression_text <- paste(
  "Slope:", round(slope, 3), "\n",
  "Intercept:", round(intercept, 3), "\n",
  "R-squared:", round(r_squared, 3), "\n",
  "Correlation:", round(correlation_coefficient, 3)
)

# Determine annotation position for displaying text on the plot
text_x_pos <- min(data$Sensor1_Voltage) + (max(data$Sensor1_Voltage) -
min(data$Sensor1_Voltage)) * 0.05  # 5% from the left
text_y_pos <- min(data$Sensor2_Voltage) + (max(data$Sensor2_Voltage) -
min(data$Sensor2_Voltage)) * 0.85  # 85% from the bottom

# ==============================
# plot of Sensor 1 Voltage vs. Sensor 2 Voltage with best-fit line
# ==============================
plot <- ggplot(data, aes(x = Sensor1_Voltage, y = Sensor2_Voltage)) +
  # Add scatter points in blue with 50% transparency
  geom_point(alpha = 0.5, color = "blue") +
```

```r
  # Add a linear regression line (best-fit line) in red
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  # Annotate the plot with regression parameters
  annotate("text", x = text_x_pos, y = text_y_pos, label = regression_text,
size = 5, hjust = 0, color = "black") +
  # Set the plot title
  ggtitle("Sensor 1 Voltage vs. Sensor 2 Voltage") +
  # Set x-axis label
  xlab("Sensor 1 Voltage (V)") +
  # Set y-axis label
  ylab("Sensor 2 Voltage (V)") +
  # Use a minimal theme for a clean layout
  theme_minimal()

# ------------------------------
# Display the plot
# ------------------------------
# Print the scatter plot with best-fit line
print(plot)

## `geom_smooth()` using formula = 'y ~ x'
```



```r
# ------------------------------
# Save the plot as a JPEG file
# ------------------------------
ggsave("Sensor_Correlation_voltage.jpeg", plot = plot, width = 6, height = 4,
dpi = 300)

## `geom_smooth()` using formula = 'y ~ x'

# ------------------------------
# Print regression results to the console
# ------------------------------
# Display regression results with explanatory text
cat("\n--- Regression Results ---\n")

##
## --- Regression Results ---

cat("Slope:", slope, "\n")

## Slope: 0.9833464

cat("Intercept:", intercept, "\n")

## Intercept: 0.03109629

cat("R-squared:", r_squared, "\n")

## R-squared: 0.8869885

cat("Correlation Coefficient:", correlation_coefficient, "\n")

## Correlation Coefficient: 0.9418007
```