Statistical Methods - HW1

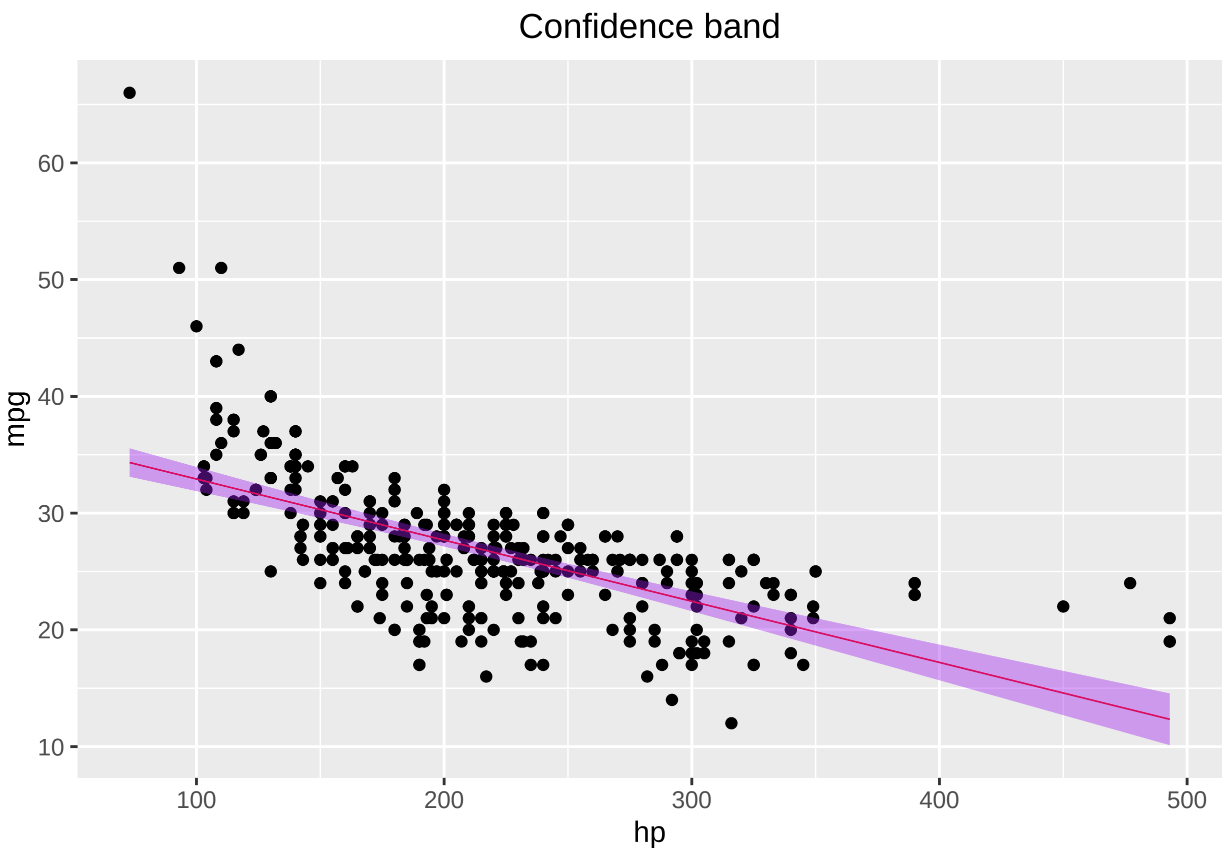
2023-01-20

Problem 1

Write a function $confBand(x, y, conf = 0.95)$ taking in a predictor vector $(x_1, ..., x_n)$ and a response vector $y = (y_1, ..., y_n)$ and return a plot with the points $(x_1, y_1), ..., (x_n, y_n)$, the least squares line, and the confidence band at level conf. Apply your function to hp and mpg from the 04cars dataset.

```r
# Loading the 04cars dataset
load("04cars.rda")
# Loading ggplot library
library(ggplot2)

#' Creating a data frame "df" with 2 columns from the 04cars dataset
#' $Horsepower (13) and $Highway_MPG (15)
df = dat[,c(13,15)]
# Setting the names for the columns of the data frame "df"
names(df) = c("hp", "mpg")
# Using complete.cases() on "df" to remove the missing values
df = df[complete.cases(df),]

#' Defining a function confBand that has 2 parameters that needs to passed
#' when calling the function (predictor vector (hp) and response vector (mpg))
#' and a default argument conf
confBand <- function(x, y, conf = 0.95) {
  # Using lm() function to fit the linear model
  fit = lm(y ~ x)
  std_err = predict(fit, interval = "confidence", level = conf, se.fit = TRUE)$se.fit
  y_pred = fit$fitted.values
  lower <- y_pred - sqrt(2 * qf(p = conf, 2, 98))*std_err
  upper <- y_pred + sqrt(2 * qf(p = conf, 2, 98))*std_err
  # Plotting the points with point shape = 16
  # plot(y ~ x, pch = 16)
  #title(main = "Least square line and Confidence Intervals")
  # Plotting the least squares regression line using abline() function
  #abline(fit, col = 'red')
  # Plotting the confidence intervals
  #lines(x, lower, col = 'blue')
  #lines(x, upper, col = 'blue')

  # Plotting the confidence band using ggplot library and geom_ribbon()
  ggplot(data = df, aes(x = hp, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red", linewidth = 0.3) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "purple", alpha = 0.4) +
  ggtitle("Confidence band") +
  theme(plot.title = element_text(hjust = 0.5))
}

#' Calling confBand function by passing hp and mpg columns
#' from the dataframe "df" as arguments
confBand(df$hp, df$mpg)
```

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



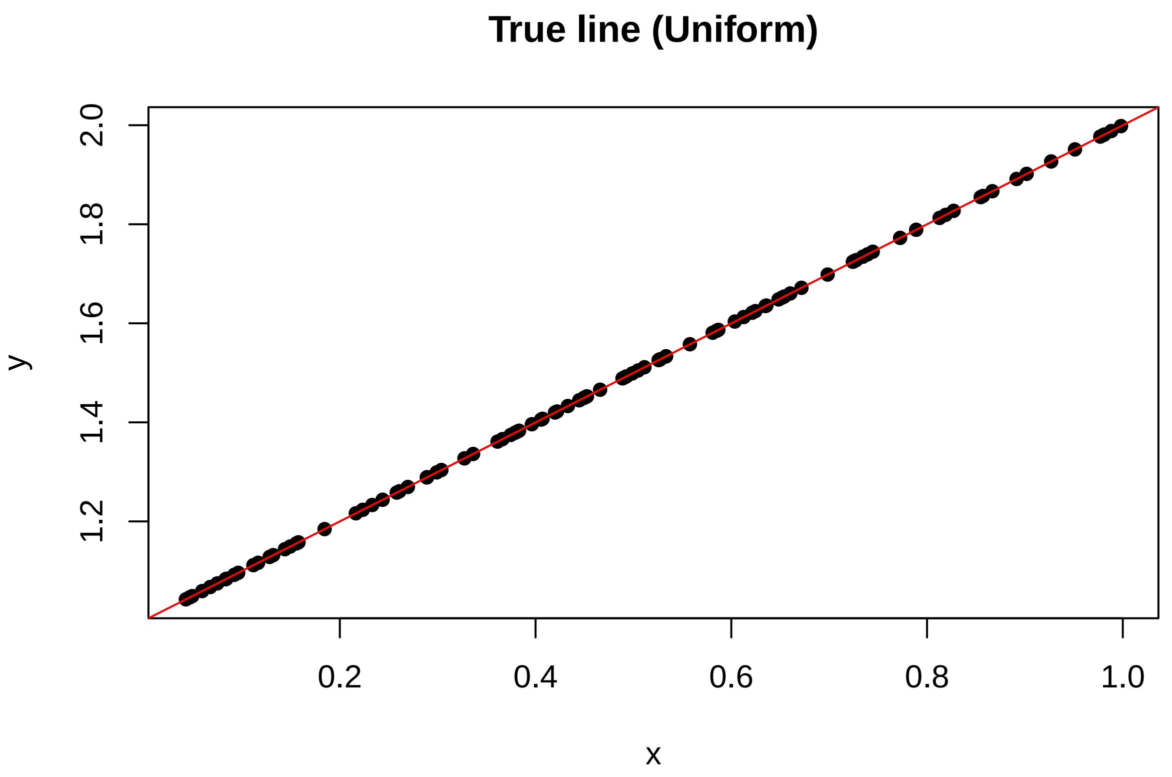
Problem 2

Let n = 100 and draw $x_1, ..., x_n$ belongs to $Unif(0, 1)$, which stay fixed in what follows. Repeat the following experiment $N = 1000$ times.

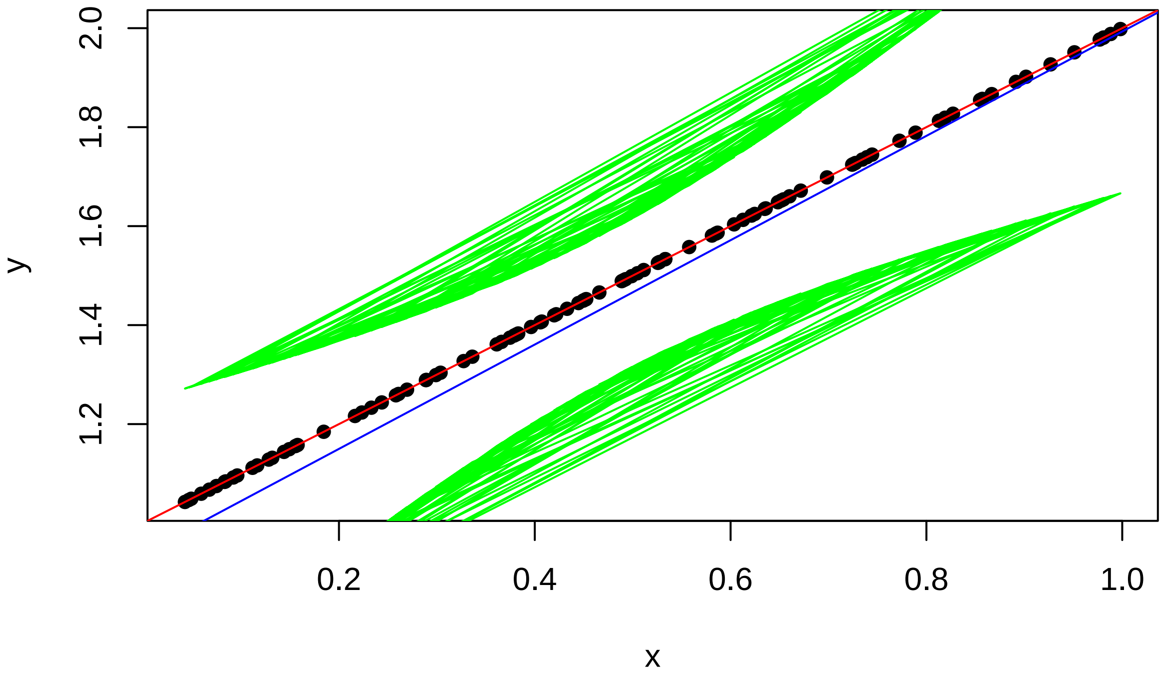
- Generate $y_i = 1 + x_i + e_i$, with e_i i.i.d. $N(0, 0.2)$.
- Compute the 99% confidence band and record whether it contains the true line, or not.

Summarize the result of this numerical experiment by returning the proportion of times (out of N) that the confidence band contained the true line.

```
# Generating n=100 uniform numbers from 0 to 1
x = runif(n = 100, 0, 1)
y = 1 + x
# Plotting the points (x, y) with point shape = 16
plot(x, y, pch = 16)
title(main = "True line (Uniform)")
# Using lm() function to fit the model
fitunif = lm(y ~ x)
# Plotting the true line using abline() function
abline(fitunif, col = 'red')
```



```
#' Counter variable to count the number of times the confidence band
#' contains the true line
cnt_contains_trueline = 0


# "for" loop for repeating the experiment N = 1000 times
N = 1000
for(i in 1:N) {
  #' simulating random variates n = 100 times having normal distribution
  #' with mean = 0 and variance = 0.2
  eps = rnorm(n = 100, 0, sqrt(0.2))
  # Regression equation
  y_i = 1 + x + eps
  # Using lm() function to fit the model
  fit = lm(y_i ~ x)
  # Plotting the lines using abline() function
  #abline(fit, col = 'blue')
  #abline(fitunif, col = 'red')

  #Using f-value to find the simultaneous confidence bands
  std_err = predict(fit, interval = "confidence", level = 0.99, se.fit = TRUE)$se.fit
  y_pred = fit$fitted.values
  lower <- y_pred - sqrt(2 * qf(p = 0.99, 2, 98))*std_err
  upper <- y_pred + sqrt(2 * qf(p = 0.99, 2, 98))*std_err
  #' Counting the number of times the confidence band
  #' contains the true line (based on the coeff of true line)
  if ((all (y >= lower[] & y <= upper[]))) {
    cnt_contains_trueline = cnt_contains_trueline + 1
  }
  if(i==1){
    plot(x, y, pch = 16)
    abline(fit, col = 'blue')
    abline(fitunif, col = 'red')
    lines(x,upper,col='green')
    lines(x,lower,col='green')
  }
}
```



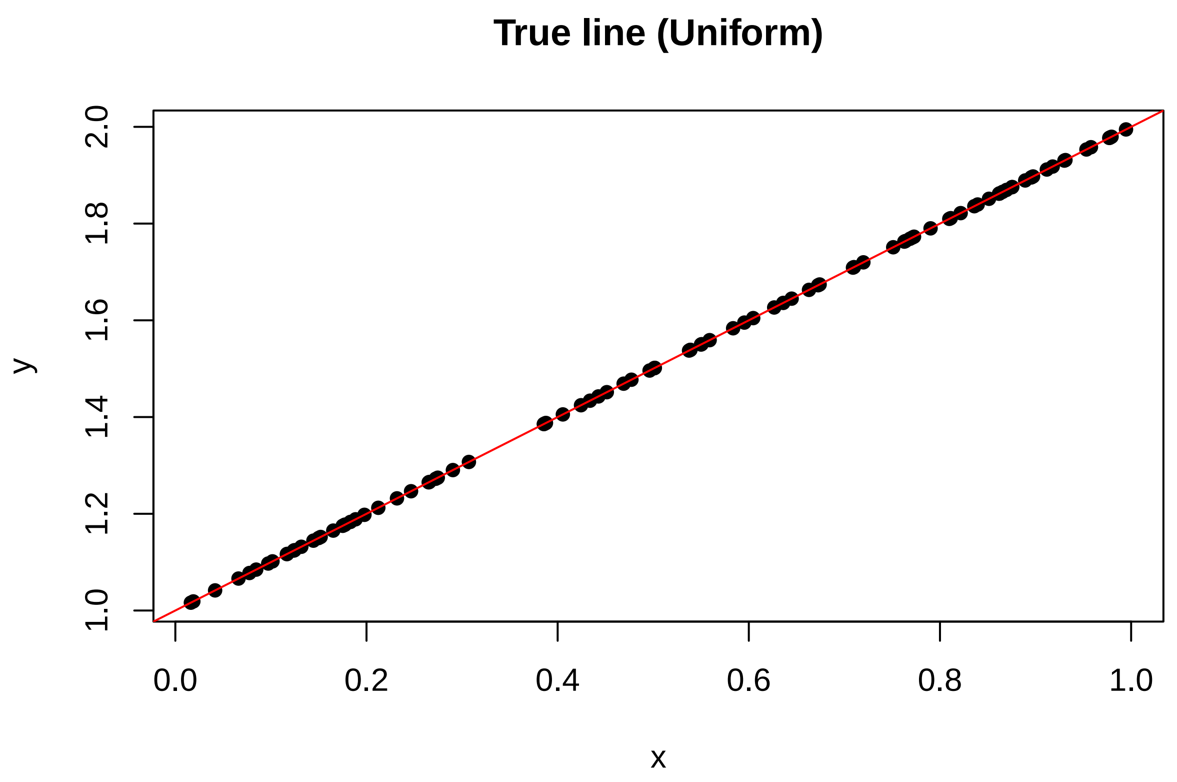
```
#' Calculating the proportion of times (out of N)
#' the confidence band contained the true line
proportion_contains_trueline = cnt_contains_trueline / N
sprintf("Proportion of times the confidence band contained the true line: %f", proportion_contains_truel
```

```
## [1] "Proportion of times the confidence band contained the true line: 0.991000"
```

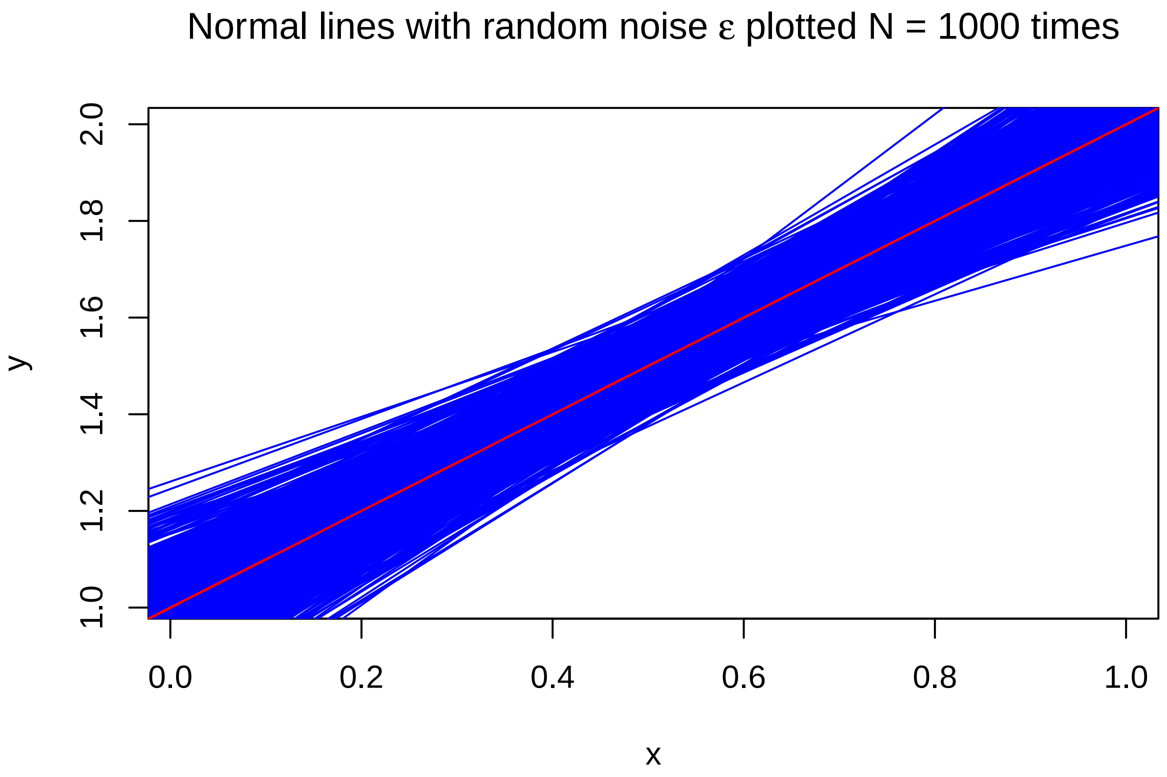
Alternative solution:

Using confint() (for Pointwise confidence bands)

```
# Generating n=100 uniform numbers from 0 to 1
x = runif(n = 100, 0, 1)
y = 1 + x
# Plotting the points (x, y) with point shape = 16
plot(x, y, pch = 16)
title(main = "True line (Uniform)")
# Using lm() function to fit the model
fitunif = lm(y ~ x)
# Plotting the true line using abline() function
abline(fitunif, col = 'red')
```



```r
# Finding coefficients of true line
intercept <- coef(fitunif)[[1]]
slope <- coef(fitunif)[[2]]

#' Counter variable to count the number of times the confidence band
#' contains the true line
cnt_contains_trueline = 0

plot(x, y, pch = 16)
title(expression(paste("Normal lines with random noise ", epsilon, " plotted N = 1000 times")))

# "for" loop for repeating the experiment N = 1000 times
N = 1000
for(i in 1:N) {
  #' simulating random variates n = 100 times having normal distribution
  #' with mean = 0 and variance = 0.2
  eps = rnorm(n = 100, 0, sqrt(0.2))
  # Regression equation
  y_i = 1 + x + eps
  # Using lm() function to fit the model
  fitnorm = lm(y_i ~ x)
  # Plotting the lines using abline() function
  abline(fitnorm, col = 'blue')
  abline(fitunif, col = 'red')
  #' Calculating the confidence interval for the parameters
  #' in a fitted regression model
  lwr_intercept <- confint(fitnorm, level = 0.99)[1,1]
  upper_intercept <- confint(fitnorm, level = 0.99)[1,2]
  lwr_x <- confint(fitnorm, level = 0.99)[2,1]
  upper_x <- confint(fitnorm, level = 0.99)[2,2]
  #' Counting the number of times the confidence band
  #' contains the true line (based on the coeff of true line)
  if (intercept >= lwr_intercept && intercept <= upper_intercept && slope >= lwr_x && slope <= upper_x)
    cnt_contains_trueline = cnt_contains_trueline + 1
  }
}
```



```
#' Calculating the proportion of times (out of N)
#' the confidence band contained the true line
proportion_contains_trueline = cnt_contains_trueline / N
sprintf("Proportion of times the confidence band contained the true line: %f", proportion_contains_truel
```

```
## [1] "Proportion of times the confidence band contained the true line: 0.994000"
```

Contribution Statement:

1. Swetha Arunraj (PID: A59019948):

- Coded Problem 1
- Added comments to the code of Problem 1
- Contributed to embed the codes in R Markdown and create the final PDF

2. Harin Raja Radha Krishnan (PID: A59019874):

- Coded Problem 2
- Added comments to the code of Problem 2
- Contributed to embed the codes in R Markdown and create the final PDF