

# Module 1: Peer Reviewed Assignment

## Outline:

The objectives for this assignment:

1. Learn when and how simulated data is appropriate for statistical analysis.
2. Experiment with the processes involved in simulating linear data.
3. Observe how the variance of data effects the best-fit line, even for the same underlying population.
4. Recognize the effects of standardizing predictors.
5. Interpreting the coefficients of linear models on both original and standardized data scales.

General tips:

1. Read the questions carefully to understand what is being asked.
2. This work will be reviewed by another human, so make sure that you are clear and concise in what your explanations and answers.

## A Quick Note On Peer-Reviewed Assignments

Welcome to your first peer reviewed assignment! These assignments will be a more open form than the auto-graded assignments, and will focus on interpretation and visualization rather than "do you get the right numbers?" These assignments will be graded by your fellow students (except in the specific cases where the work needs to be graded by a proctor) so please make your answers as clear and concise as possible.

```
In [1]: options(warn=1)
```

```
In [2]: # This cell loads the necessary libraries for this assignment
library(tidyverse)
```

— Attaching packages — tidyverse 1.3.0 —

✓ ggplot2 3.3.0	✓ purrr 0.3.4
✓ tibble 3.0.1	✓ dplyr 0.8.5
✓ tidyr 1.0.2	✓ stringr 1.4.0
✓ readr 1.3.1	✓ forcats 0.5.0

— Conflicts — tidyverse\_conflicts() —

✗ dplyr::filter()	masks	stats::filter()
✗ dplyr::lag()	masks	stats::lag()

# Problem 1: Simulating Data

We're going to let you in on a secret. The turtle data from the autograded assignment was simulated...fake data! Gasp! Importantly, simulating data, and applying statistical models to simulated data, are very important tools in data science.

Why do we use simulated data? Real data can be messy, noisy, and we almost never *really* know the underlying process that generated real data. Working with real data is always our ultimate end goal, so we will try to use as many real datasets in this course as possible. However, applying models to simulated data can be very instructive: such applications help us understand how models work in ideal settings, how robust they are to changes in modeling assumptions, and a whole host of other contexts.

And in this problem, you are going to learn how to simulate your own data.

## 1. (a) A Simple Line

Starting out, generate 10 to 20 data points for values along the x-axis. Then generate data points along the y-axis using the equation  $y_i = \beta_0 + \beta_1 x_i$ . Make it a straight line, nothing fancy.

Plot your data (using ggplot!) with your **x** data along the x-axis and your **y** data along the y-axis.

In the *Markdown* cell below the `R` cell, describe what you see in the plot.

**Tip:** You can generate your x-data *deterministically*, e.g., using either `a:b` syntax or the `seq()` function, or *randomly* using something like `runif()` or `rnorm()`. In practice, it won't matter all that much which one you choose.

```
In [3]: # Load necessary Library
library(ggplot2)

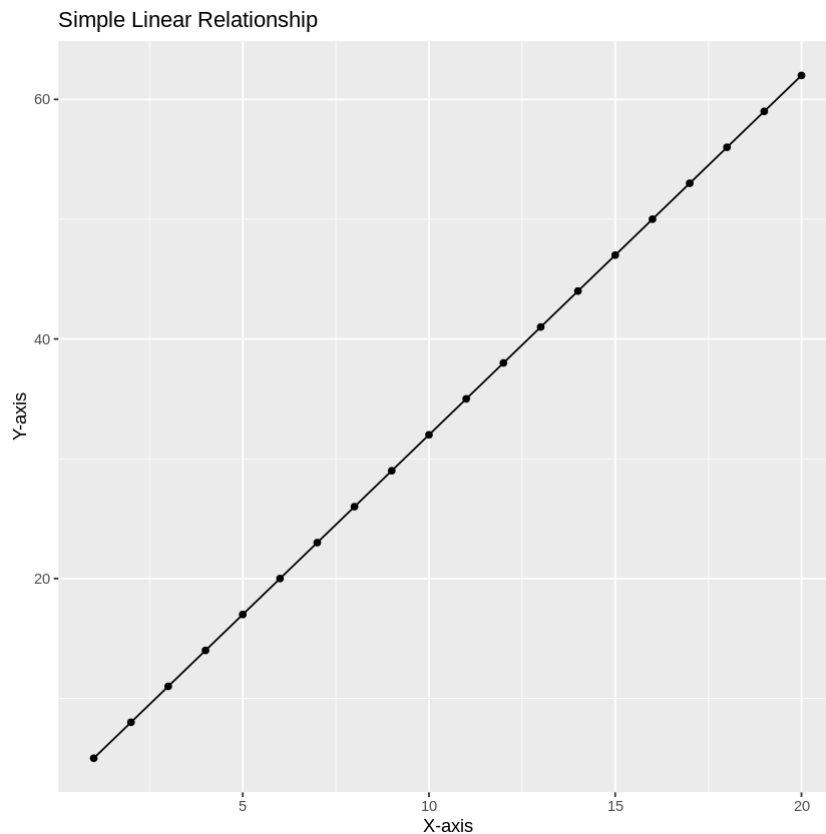
# Generate x values
x <- seq(1, 20, by = 1)

# Parameters for the Line
beta_0 <- 2
beta_1 <- 3

# Generate y values using the Line equation
y <- beta_0 + beta_1 * x

# Create a data frame
data <- data.frame(x = x, y = y)

# Plot the data
ggplot(data, aes(x = x, y = y)) +
  geom_point() +
  geom_line() +
  ggtitle("Simple Linear Relationship") +
  xlab("X-axis") +
  ylab("Y-axis")
```



This plot shows a perfect linear relationship between  $x$  and  $y$ . Each point lies exactly on the line  $y = \beta_0 + \beta_1 x$ .

## 1. (b) The Error Component

That is a perfect set of data points, but that is a problem in itself. In almost any real life situation, when we measure data, there will be some error in those measurements. Recall that our simple linear model is of the form:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2)$$

Add an error term to your y-data following the formula above. Plot at least three different plots (using ggplot!) with the different values of  $\sigma^2$ .

How does the value of  $\sigma^2$  affect the final data points? Type your answer in the *Markdown* cell below the R cell.

**Tip:** To randomly sample from a normal distribution, check out the `rnorm()` function.

```
In [4]: # Set seed for reproducibility
set.seed(123)

# Define sigma values
sigma_values <- c(1, 3, 5)

# Create plots for different sigma values
plots <- lapply(sigma_values, function(sigma) {
  # Add random error to y
  y_error <- y + rnorm(length(x), mean = 0, sd = sigma)

  # Create a data frame
  data_error <- data.frame(x = x, y = y_error)

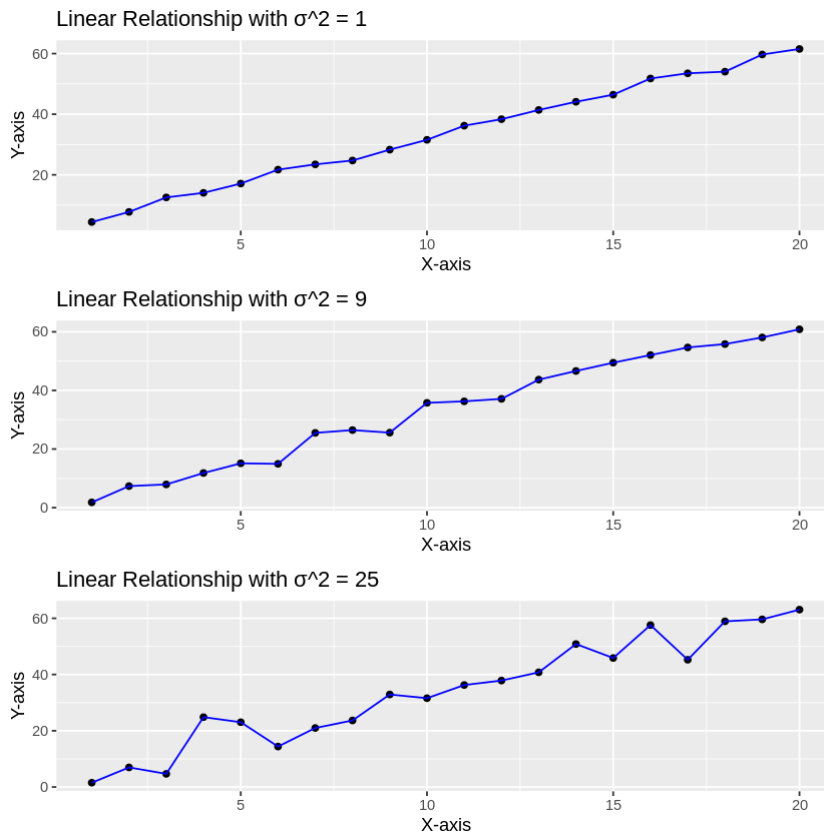
  # Plot with error
  ggplot(data_error, aes(x = x, y = y)) +
    geom_point() +
    geom_line(aes(y = y), color = "blue") +
    ggtitle(paste("Linear Relationship with  $\sigma^2$  =", sigma^2)) +
    xlab("X-axis") +
    ylab("Y-axis")
})

# Print plots
library(gridExtra)
grid.arrange(grobs = plots, ncol = 1)
```

Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

combine



As the value of  $\sigma^2$  increases, the points become more dispersed around the line, indicating higher variability in the data. This simulates real-world scenarios where measurements have inherent variability, affecting the precision of our observations.

## Problem 2: The Effects of Variance on Linear Models

Once you've completed **Problem 1**, you should have three different "datasets" from the same underlying data function but with different variances. Let's see how those variance affect a best fit line.

Use the `lm()` function to fit a best-fit line to each of those three datasets. Add that best fit line to each of the plots and report the slopes of each of these lines.

Do the slopes of the best-fit lines change as  $\sigma^2$  changes? Type your answer in the *Markdown* cell below the R cell.

**Tip:** The `lm()` function requires the syntax `lm(y~x)`.

```

In [5]: # Load necessary Libraries
library(ggplot2)
library(gridExtra)

# Set seed for reproducibility
set.seed(123)

# Define sigma values
sigma_values <- c(1, 3, 5)

# Create a list to store the plots and slopes
plots <- list()
slopes <- numeric(length(sigma_values))

# Loop through each sigma value
for (i in 1:length(sigma_values)) {
  sigma <- sigma_values[i]

  # Add random error to y
  y_error <- y + rnorm(length(x), mean = 0, sd = sigma)

  # Create a data frame
  data_error <- data.frame(x = x, y = y_error)

  # Fit a linear model
  model <- lm(y ~ x, data = data_error)

  # Store the slope
  slopes[i] <- coef(model)[2]

  # Plot with error and best-fit line
  plots[[i]] <- ggplot(data_error, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, color = "red") +
    geom_line(aes(y = y), color = "blue", linetype = "dashed") +
    ggtitle(paste("Linear Relationship with  $\sigma^2 =$ ", sigma^2)) +
    xlab("X-axis") +
    ylab("Y-axis")
}

# Display plots
grid.arrange(grobs = plots, ncol = 1)

# Report slopes
slopes

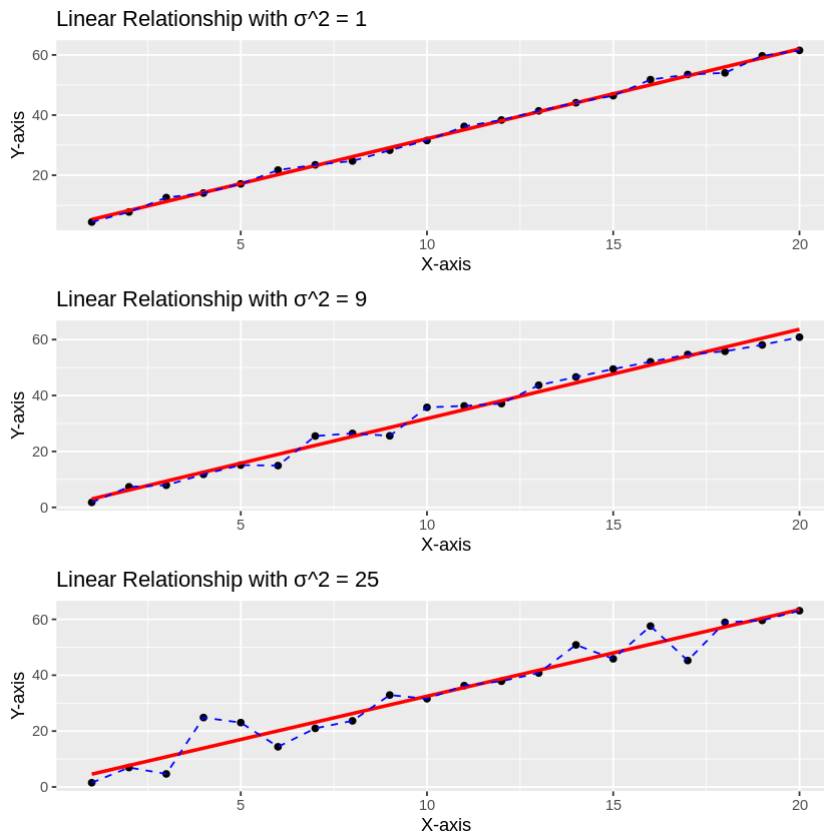
```

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

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

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

2.9839553611847 · 3.19267622696729 · 3.09873118463354



The larger the value of  $\sigma^2$ , the less closely the points follow a line. The  $\sigma^2 = 25$  plot, looks as if there's no linear relationship between  $x$  and  $y$ .

## Problem 3: Interpreting the Linear Model

Choose one of the above three models and write out the actual equation of that model. Then in words, in the *Markdown* cell below the R cell, describe how a 1 unit increase in your predictor affects your response. Does this relationship make sense?

```
In [6]: # Select the model with  $\sigma^2 = 3$ 
sigma <- 3
y_error <- y + rnorm(length(x), mean = 0, sd = sigma)
data_error <- data.frame(x = x, y = y_error)

# Fit the linear model
model <- lm(y ~ x, data = data_error)

# Display the coefficients
coef(model)
```

**(Intercept):** 1.72148863691626 **x:** 2.99226287326635

For every 1 unit increase in the predictor  $x$ , the response  $y$  increases by  $\beta_1$  units. That is 2.992 units. This relationship indicates a positive correlation between the predictor and the response, which makes sense given the linear trend in the data.

## Problem 4: The Effects of Standardizing Data

We spent some time standardizing data in the autograded assignment. Let's do that again with your simulated data.

Using the same model from **Problem 3**, standardize your simulated predictor. Then, using the `lm()` function, fit a best fit line to the standardized data. Using `ggplot`, create a scatter plot of the standardized data and add the best fit line to that figure.

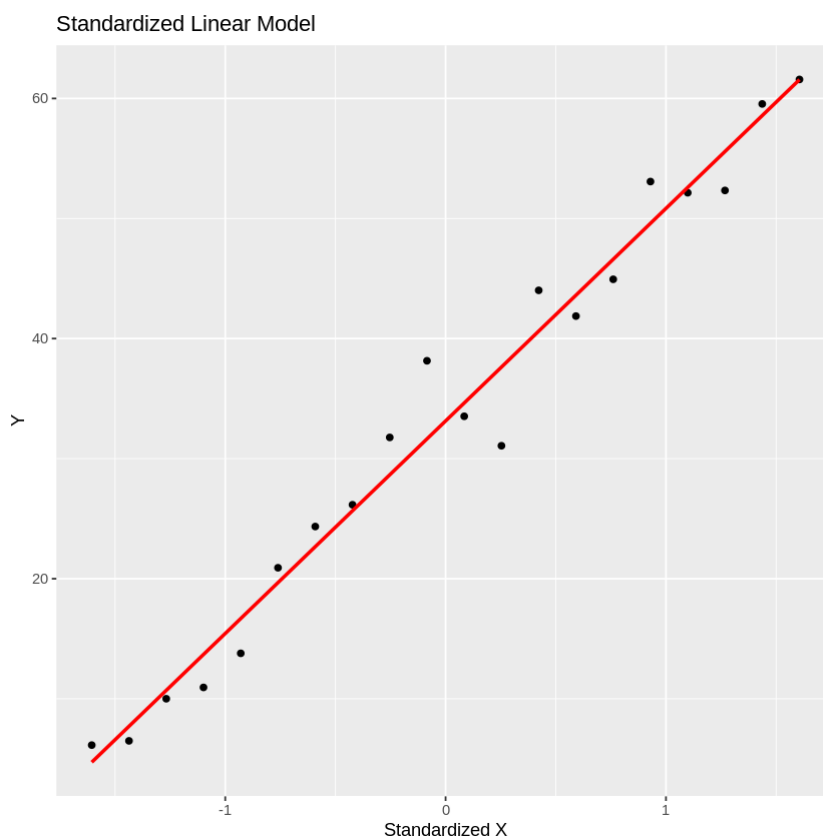
```
In [7]: # Standardize the predictor
x_standardized <- scale(x)

# Fit the model with standardized predictor
model_standardized <- lm(y_error ~ x_standardized)

# Create a data frame for plotting
data_standardized <- data.frame(x_standardized = x_standardized, y = y_error)

# Plot the standardized data with the best-fit line
ggplot(data_standardized, aes(x = x_standardized, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  ggtitle("Standardized Linear Model") +
  xlab("Standardized X") +
  ylab("Y")
```

`geom\_smooth()` using formula 'y ~ x'





# Problem 5: Interpreting the Standardized Model

Write out the expression for your standardized model. In words, in the *Markdown* cell below the `R` cell, describe how a 1 unit increase in your standardized predictor affects the response. Is this value different from the original model? If yes, then what can you conclude about interpretation of standardized predictors vs. unstandardized predictors.

The equation of the standardized model is:  $y = \beta_0 + \beta_1 x * \textit{standardized}$

## Interpretation

For every 1 unit increase in the standardized predictor, the response increases by  $\beta_1$  units. This value represents the change in  $y$  per standard deviation increase in  $x$ .

## Comparison to Original Model

The slope of the standardized model represents the effect size in terms of standard deviations, making it easier to compare across different datasets or predictors. The original model slope indicates the effect size in the original units of the predictor. Thus, standardizing provides a dimensionless interpretation that can facilitate comparisons.