Question 1

As always, the first step is to load the data into R.

```
data <- read.csv("LungCancer.csv")
summary(data)
```

```
##       Case            Smoker
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:1.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.4996   Mean   :0.9422
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

Then we make the contingency table

```
contingency_table <- table(data$Smoker, data$Case)
rownames(contingency_table) <- c("Smoker", "Non Smoker")
colnames(contingency_table) <- c("Case", "No Case")
print(contingency_table)
```

```
##
##              Case No Case
##   Smoker       60      22
##   Non Smoker  650     687
```

Then we compute the table of expected outputs

```
row_totals <- rowSums(contingency_table)
col_totals <- colSums(contingency_table)
grand_total <- sum(contingency_table)

expected_counts <- outer(row_totals, col_totals) / grand_total
print(expected_counts)
```

```
##                 Case   No Case
## Smoker      41.02889  40.97111
## Non Smoker 668.97111 668.02889
```

Calculating the test statistic by hand:

```
observed_chi_squared <- sum((contingency_table - expected_counts)^2 / expected_counts)
print(paste("Observed chi-squared value:", observed_chi_squared))
```

```
## [1] "Observed chi-squared value: 18.6329948106806"
```

The distribution of the test statistic follows a chi-squared distribution with (rows - 1) * (columns - 1) degrees of freedom, assuming no association between smoking and lung cancer (i.e that the null hypothesis is true). In this case, with a 2x2 table, it follows a chi-squared distribution with 1 degree of freedom.

We can use the chi squared test to compute a p value for our observations under the null hypothesis.

```
chi_squared_test <- chisq.test(contingency_table)
p_value <- chi_squared_test$p.value
print(paste("Chi-squared value:", chi_squared_test$statistic))
```

```
## [1] "Chi-squared value: 17.6637603325743"
```

```
print(paste("P-value:", p_value))
```

```
## [1] "P-value: 2.63601282605485e-05"
```

In conclusion, our analysis of the data reveals a significant association between smoking and lung cancer. The p-value obtained from the chi-squared test is extremely low (2.64e-05), which indicates that, if there were no association between smoking and lung cancer, it would be incredibly unlikely to observe a test statistic as extreme as the one observed in our data.

Comparing the observed and expected counts in the contingency tables, we can see that smokers have a significantly higher rate of lung cancer, while non-smokers have a lower rate than expected under the null hypothesis. This further supports the presence of a significant association between smoking and lung cancer.

Question 2

We start by loading the data and creating a contingency table:

```
library(knitr)

observed_counts <- matrix(c(7, 7, 7, 13,
                            27, 34, 12, 18,
                            55, 52, 11, 24),
                          nrow = 3,
                          byrow = TRUE)

rownames(observed_counts) <- c("Moderate-advanced", "Minimal", "Not Present")
colnames(observed_counts) <- c("O", "A", "AB", "B")
contingency_table <- as.table(observed_counts)
kable(contingency_table, caption = "Contingency Table")
```

Contingency Table

	O	A	AB	B
Moderate-advanced	7	7	7	13
Minimal	27	34	12	18

	O	A	AB	B
Not Present	55	52	11	24

Next, we can compute a table of expected outputs:

```
row_totals <- rowSums(contingency_table)
col_totals <- colSums(contingency_table)
grand_total <- sum(contingency_table)

expected_counts <- outer(row_totals, col_totals) / grand_total
kable(expected_counts, caption = "Expected Counts Table")
```

Expected Counts Table

	O	A	AB	B
Moderate-advanced	11.33333	11.84270	3.820225	7.003745
Minimal	30.33333	31.69663	10.224719	18.745318
Not Present	47.33333	49.46067	15.955056	29.250936

Then, we can compute the value of the test statistic manually:

```
observed_chi_squared <- sum((contingency_table - expected_counts)^2 / expected_counts)
paste("Observed chi-squared value:", observed_chi_squared)
```

```
## [1] "Observed chi-squared value: 16.1426983845457"
```

The distribution of the test statistic follows a chi-squared distribution with (rows - 1) * (columns - 1) degrees of freedom, assuming no association between disease and blood group. In this case, with a 3x4 table, it follows a chi-squared distribution with (3 - 1) * (4 - 1) = 6 degrees of freedom.

Finally, we compute the p-value using the chi-squared test and provide a conclusion.

```
chi_squared_test <- chisq.test(contingency_table)
```

```
## Warning in chisq.test(contingency_table): Chi-squared approximation may be
## incorrect
```

```
p_value <- chi_squared_test$p.value
paste("Chi-squared value:", chi_squared_test$statistic)
```

```
## [1] "Chi-squared value: 16.1426983845457"
```

```
paste("P-value:", p_value)
```

```
## [1] "P-value: 0.0130081997704131"
```

In conclusion, our analysis of the data reveals a significant association between tuberculosis disease severity and blood group within the ABO system. The p-value obtained from the chi-squared test is relatively low (0.013), which indicates that, if there were no association between the disease and blood group, it would be quite unlikely to observe a test statistic as extreme as the one observed in our data (16.14). So it would be reasonable to reject the null hypothesis, that is: the assumption that there is no relation between the severity of tuberculosis and the blood type.

Comparing the observed and expected counts in the tables, we can see that the counts do differ considerably from what we would expect under the null hypothesis. For example, the number of cases with moderate-advanced severity is notably lower for blood group O and higher for blood group B than expected. This supports the presence of a significant association between tuberculosis disease severity and blood group within the ABO system.

Question 3

Let’s load the data and generate the scatterplot:

```
library(ggplot2)
library(gridExtra)
```

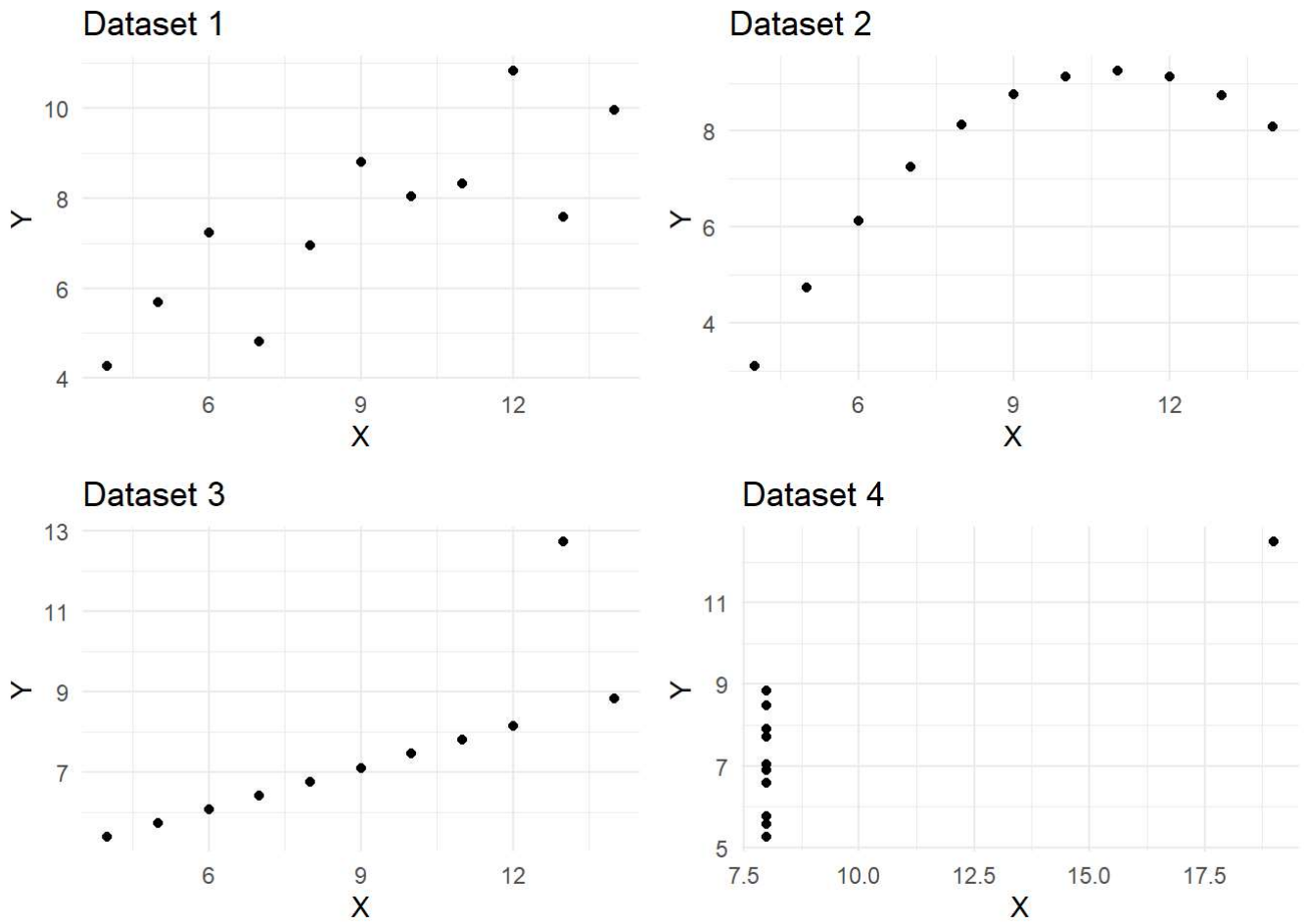
```
## Warning: package 'gridExtra' was built under R version 4.2.3
```

```
library(readr)
```

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

```
# Read the CSV file line by line
lines <- readLines("anscombe.csv")

# Find the starting and ending line numbers for each dataset
start_lines <- grep("^DATASET", lines)
end_lines <- c(start_lines[-1] - 1, length(lines))

# Read each dataset into a data frame
dataset1 <- read.csv(text = paste(lines[(start_lines[1] + 1):(end_lines[1] - 1)], collapse =
"\n"), header = TRUE)
dataset2 <- read.csv(text = paste(lines[(start_lines[2] + 1):(end_lines[2] - 1)], collapse =
"\n"), header = TRUE)
dataset3 <- read.csv(text = paste(lines[(start_lines[3] + 1):(end_lines[3] - 1)], collapse =
"\n"), header = TRUE)
dataset4 <- read.csv(text = paste(lines[(start_lines[4] + 1):end_lines[4]], collapse = "\n"), he
ader = TRUE)

plot1 <- ggplot(dataset1, aes(x = Set1x, y = Set1y)) + geom_point() +
  labs(title = "Dataset 1", x = "X", y = "Y") + theme_minimal()

plot2 <- ggplot(dataset2, aes(x = Set2x, y = Set2y)) + geom_point() +
  labs(title = "Dataset 2", x = "X", y = "Y") + theme_minimal()

plot3 <- ggplot(dataset3, aes(x = Set3x, y = Set3y)) + geom_point() +
  labs(title = "Dataset 3", x = "X", y = "Y") + theme_minimal()

plot4 <- ggplot(dataset4, aes(x = Set4x, y = Set4y)) + geom_point() +
  labs(title = "Dataset 4", x = "X", y = "Y") + theme_minimal()

grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



Dataset 1 exhibits some variability at smaller scales, yet a discernible linear trend is present across the entire range.

Dataset 2 displays an initial linear increase followed by a more parabolic pattern in the second half, characterized by a distinct non-linear bend and decrease.

Dataset 3 largely follows a linear pattern, with a notable outlier around $x \approx 13$.

Dataset 4 features data points clustered around the same x value, suggesting that the chosen explanatory variable may be inadequate or that the data is not suitable for linear regression analysis.

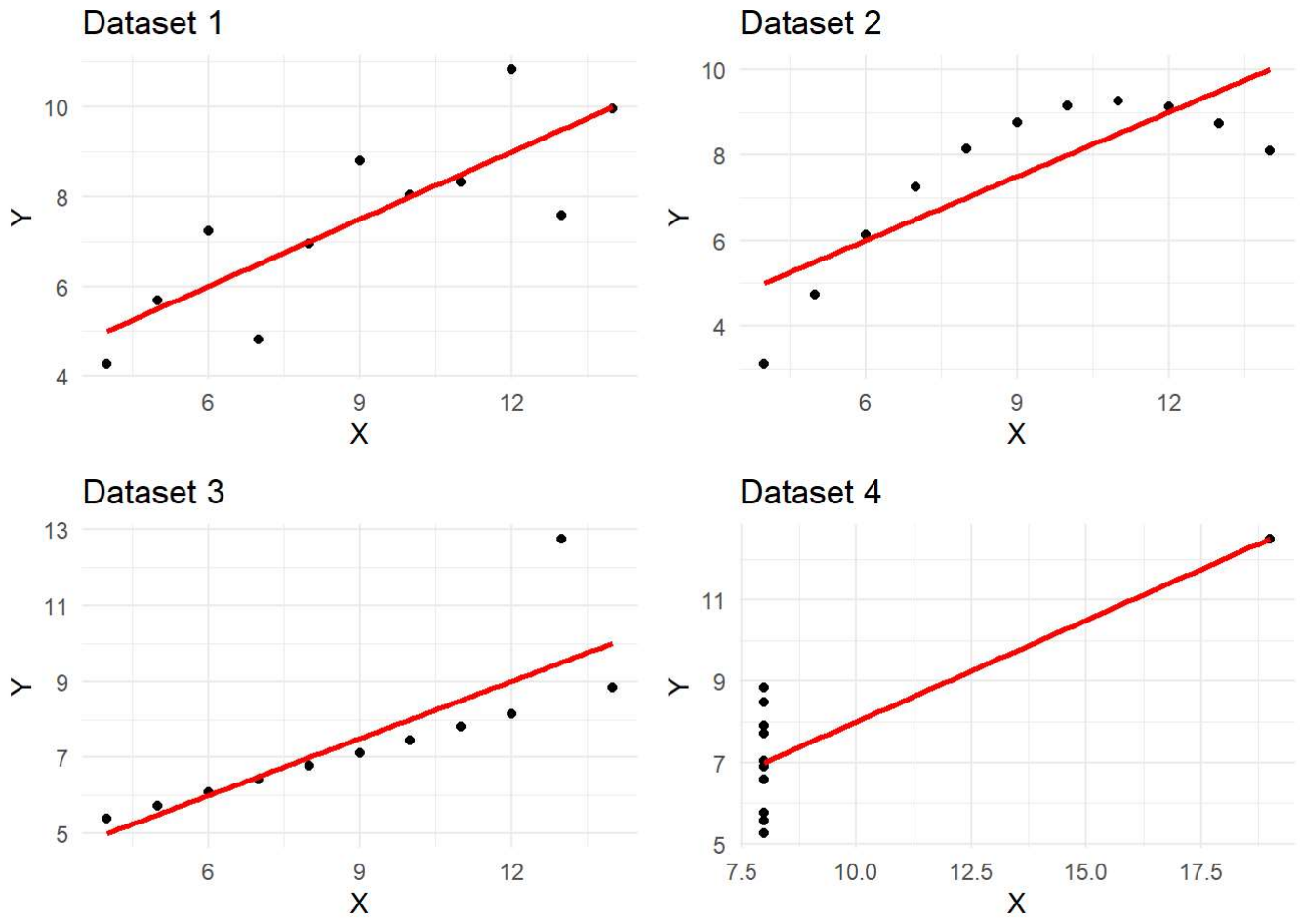
```
# Perform simple linear regression for each dataset
lm1 <- lm(Set1y ~ Set1x, data = dataset1)
lm2 <- lm(Set2y ~ Set2x, data = dataset2)
lm3 <- lm(Set3y ~ Set3x, data = dataset3)
lm4 <- lm(Set4y ~ Set4x, data = dataset4)

# Add regression lines to the scatter plots
plot1 <- plot1 + geom_smooth(method = "lm", se = FALSE, color = "red")
plot2 <- plot2 + geom_smooth(method = "lm", se = FALSE, color = "red")
plot3 <- plot3 + geom_smooth(method = "lm", se = FALSE, color = "red")
plot4 <- plot4 + geom_smooth(method = "lm", se = FALSE, color = "red")

# Display the scatter plots with regression lines
grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)
```

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

```
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```

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



```
# Calculate R-squared values
r_squared_1 <- summary(lm1)$r.squared
r_squared_2 <- summary(lm2)$r.squared
r_squared_3 <- summary(lm3)$r.squared
r_squared_4 <- summary(lm4)$r.squared

# Create a table showing the R-squared values for each dataset
r_squared_table <- data.frame(
  Dataset = c("Dataset 1", "Dataset 2", "Dataset 3", "Dataset 4"),
  R_Squared = c(r_squared_1, r_squared_2, r_squared_3, r_squared_4)
)

# Print the table
library(knitr)
kable(r_squared_table, caption = "R-squared values for each dataset")
```

R-squared values for each dataset

Dataset	R_Squared
Dataset 1	0.6665425
Dataset 2	0.6662420
Dataset 3	0.6663240
Dataset 4	0.6667073

The results show that despite the different shapes of the datasets, they all have approximately the same R-squared value. This is due to the clever arrangement of data points that results in the same R-squared value when fitting a linear regression. Dataset 1 and 3 show a decent linear fit, while dataset 2 somewhat fits the data but fails to capture the parabolic trend. Dataset 4 is not suitable for linear regression modeling, as the fitted trend is nonsensical. The R-squared value, being the square of the sample correlation coefficient, is not always the best indicator of a model's fit, as demonstrated by these datasets.

Question 4

First we load the data

```
growth_data <- read.csv('growth.txt')


head(growth_data)
```

```
##     t.height
## 1      1 0.2
## 2 2 0.550972
## 3 3 3.110788
## 4 4 5.035953
## 5 5 5.310012
## 6 6 3.434758
```

Now we can create our 3 models:

```
# Load required packages
library(readr)
library(ggplot2)

# Read data
file_path <- "growth.txt"
growth_data <- read_table(file_path, col_names = c("time", "height"))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   time = col_character(),
##   height = col_character()
## )
```

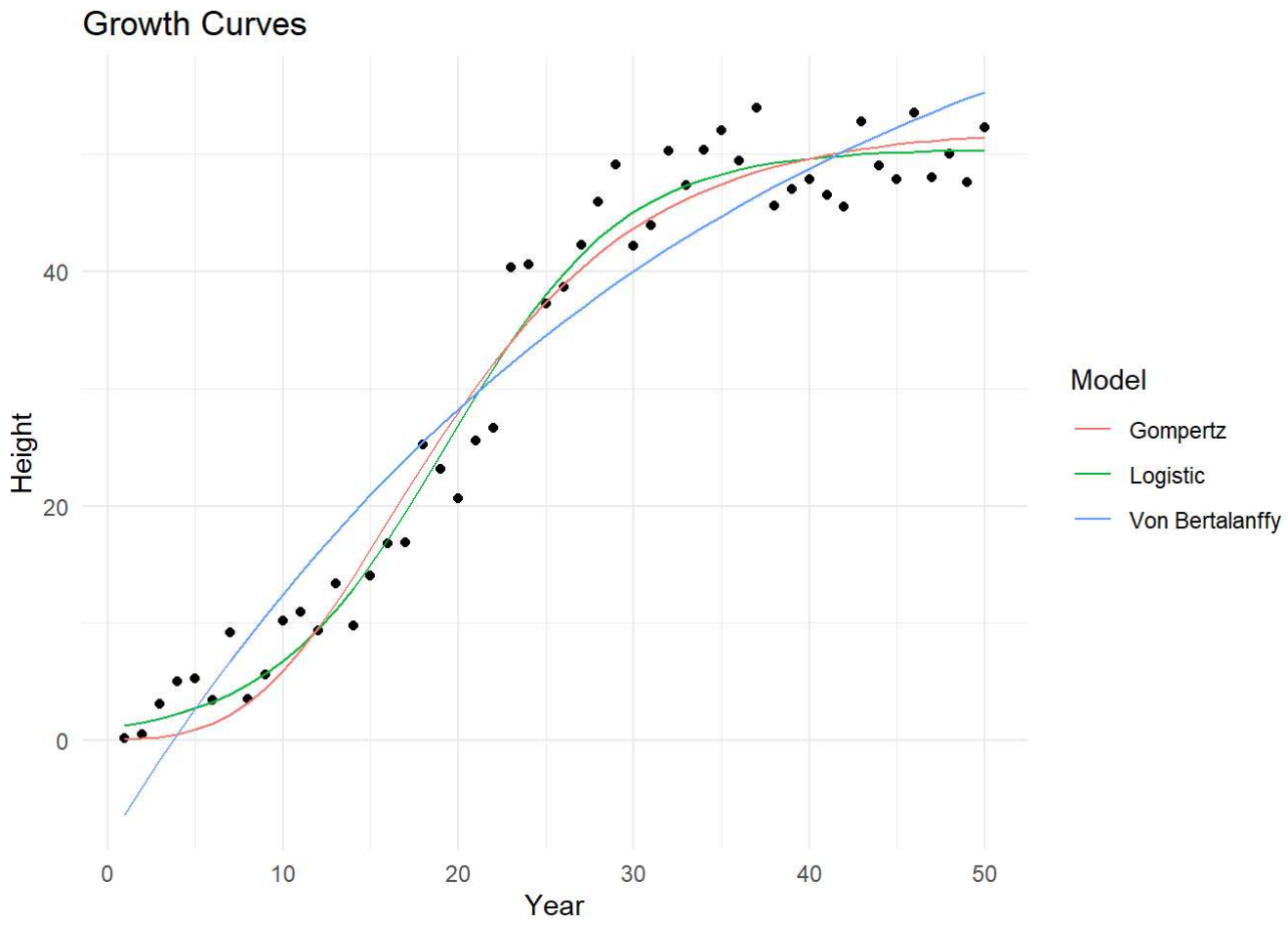
```
# Convert 'time' and 'height' to numeric
growth_data$time <- as.numeric(growth_data$time)
```

```
## Warning: NAs introduced by coercion
```

```
growth_data$height <- as.numeric(growth_data$height)
```

```
## Warning: NAs introduced by coercion
```

```
# Remove any rows with NA values
growth_data <- na.omit(growth_data)

# Fit the models
logistic_fit <- nls(height ~ SSlogis(time, a, b, c), data = growth_data)
gompertz_fit <- nls(height ~ SSgompertz(time, a, b, c), data = growth_data)

von_bertalanffy <- function(time, a, b, c) {
  a - a * exp(-b * (time + c))
}
# Objective function for the Von Bertalanffy model
objective_von_bert <- function(pars, time, height) {
  a <- pars[1]
  b <- pars[2]
  c <- pars[3]
  sum((height - von_bertalanffy(time, a, b, c))^2)
}

# Estimate starting values using optim()
start_vals_von_bert <- optim(c(max(growth_data$height), 0.1, 0), objective_von_bert, time = grow
th_data$time, height = growth_data$height)$par

# Fit the Von Bertalanffy model
von_bert_fit <- nls(height ~ a - a * exp(-b * (time + c)), data = growth_data, start = list(a =
start_vals_von_bert[1], b = start_vals_von_bert[2], c = start_vals_von_bert[3]))

# Create a new data frame for predictions
prediction_data <- data.frame(time = growth_data$time)
prediction_data$logistic <- predict(logistic_fit, prediction_data)
prediction_data$gompertz <- predict(gompertz_fit, prediction_data)
prediction_data$von_bert <- predict(von_bert_fit, prediction_data)

# Plot the observed data and the estimated curves for all three models
ggplot(growth_data, aes(x = time, y = height)) +
  geom_point() +
  geom_line(data = prediction_data, aes(y = logistic, color = "Logistic")) +
  geom_line(data = prediction_data, aes(y = gompertz, color = "Gompertz")) +
  geom_line(data = prediction_data, aes(y = von_bert, color = "Von Bertalanffy")) +
  labs(title = "Growth Curves", x = "Year", y = "Height", color = "Model") +
  theme_minimal()
```



For the logistic and Gompertz models, the built-in self-starting functions SSlogis() and SSgompertz() were used to automatically estimate reasonable starting values for the parameters. These functions simplify the process and reduce the risk of convergence issues when using nls fitting.

For the Von Bertalanffy model, starting values were estimated using the optim() function, which performs a general-purpose optimization. An objective function was created to calculate the sum of squared residuals between observed data and model predictions. The optim() function searched for parameter values that minimized this objective function, with an initial guess based on the maximum observed height in the data and small positive values for the other two parameters. The resulting values were used as starting values for the nls() function when fitting the model.

To provide a 95% confidence interval for lim(t→∞)f(t) for each of the three models, we will first need to extract the parameter estimates and their standard errors from each model. The lim(t→∞)f(t) for each model is as follows:

```
# Confidence intervals for each model
confint_logistic <- confint(logistic_fit)
```

```
## Waiting for profiling to be done...
```

```
confint_gompertz <- confint(gompertz_fit)
```

```
## Waiting for profiling to be done...
```

```
confint_von_bert <- confint(von_bert_fit)
```

```
## Waiting for profiling to be done...
```

```
# 95% confidence intervals for lim(t→∞)f(t) for each model
logistic_CI <- confint_logistic["a",]
gompertz_CI <- confint_gompertz["a",]
von_bert_CI <- confint_von_bert["a",]

logistic_CI
```

```
##     2.5%    97.5%
## 48.80814 52.13473
```

```
gompertz_CI
```

```
##     2.5%    97.5%
## 49.79354 55.09659
```

```
von_bert_CI
```

```
##      2.5%     97.5%
##  61.53431 103.51205
```

These confidence intervals represent the range within which we can be 95% confident that the true value of lim(t→∞)f(t) lies. In the context of this problem, this represents the height the tree would reach as time goes to infinity, or in other words, the maximum height the tree can grow to.

Now let's select the best model:

```
# Calculate AIC values
AIC_logistic <- AIC(logistic_fit)
AIC_gompertz <- AIC(gompertz_fit)
AIC_von_bert <- AIC(von_bert_fit)

# Select the best model
best_model <- which.min(c(AIC_gompertz, AIC_logistic, AIC_von_bert))

# Print best model
best_model
```

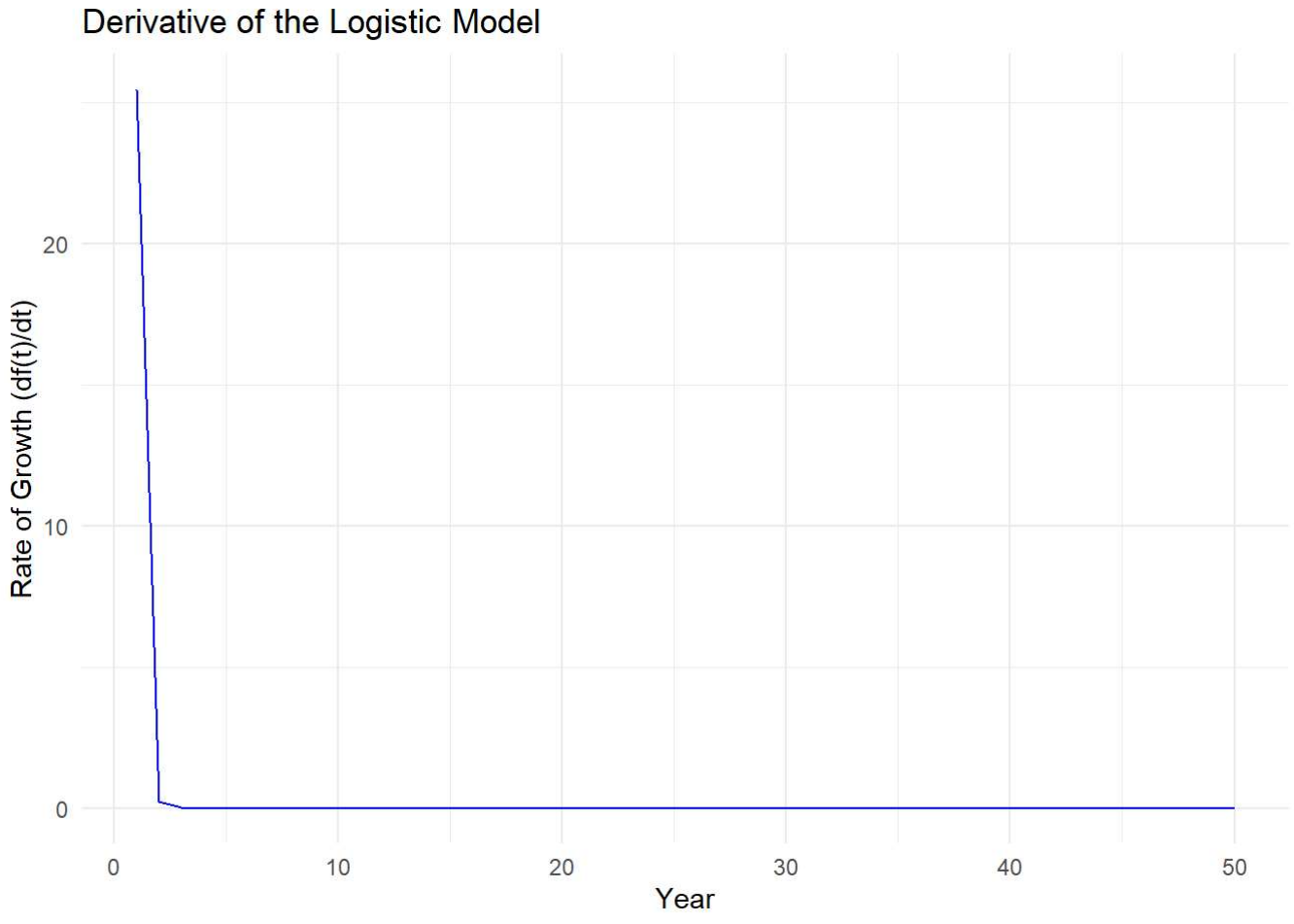
```
## [1] 2
```

In this analysis, the best model is selected based on the Akaike Information Criterion (AIC) values. The AIC is a measure of the relative quality of a statistical model, considering both the goodness of fit and the complexity of the model. Lower AIC values indicate a better balance between fitting the data well and not overfitting, which can lead

to poor predictions for new data points.

Now that we have identified the best model (logistic), let's plot an estimate of the derivative df(t)/dt, which represents the rate of growth over time:

```
# Extract parameters from the logistic model
logistic_params <- coef(logistic_fit)
a <- logistic_params["a"]
b <- logistic_params["b"]
c <- logistic_params["c"]

# Create a function for the derivative of the logistic model
logistic_derivative <- function(t, a, b, c) {
  (a * b * c * exp(-c * t)) / (1 + b * exp(-c * t))^2
}

# Calculate the derivative for the time values
prediction_data$logistic_derivative <- logistic_derivative(prediction_data$time, a, b, c)

# Plot the derivative of the logistic model
ggplot(prediction_data, aes(x = time, y = logistic_derivative)) +
  geom_line(color = "blue") +
  labs(title = "Derivative of the Logistic Model", x = "Year", y = "Rate of Growth (df(t)/dt)")
+
  theme_minimal()
```



Most of the change is bunched up at the start, so I'll also plot it with the time on a log scale.

```
# Plot the derivative of the logistic model with log scale on X-axis
ggplot(prediction_data, aes(x = time, y = logistic_derivative)) +
  geom_line(color = "blue") +
  labs(title = "Derivative of the Logistic Model (Log-scaled X-axis)", x = "Year (log scale)", y
= "Rate of Growth (df(t)/dt)") +
  scale_x_log10() +
  theme_minimal()
```