Statistics 1 Unit 5 Team 8

Nikolaos Kornilakis Rodrigo Viale Jakub Trnan Aleksandra Daneva
Luis Diego Pena Monge

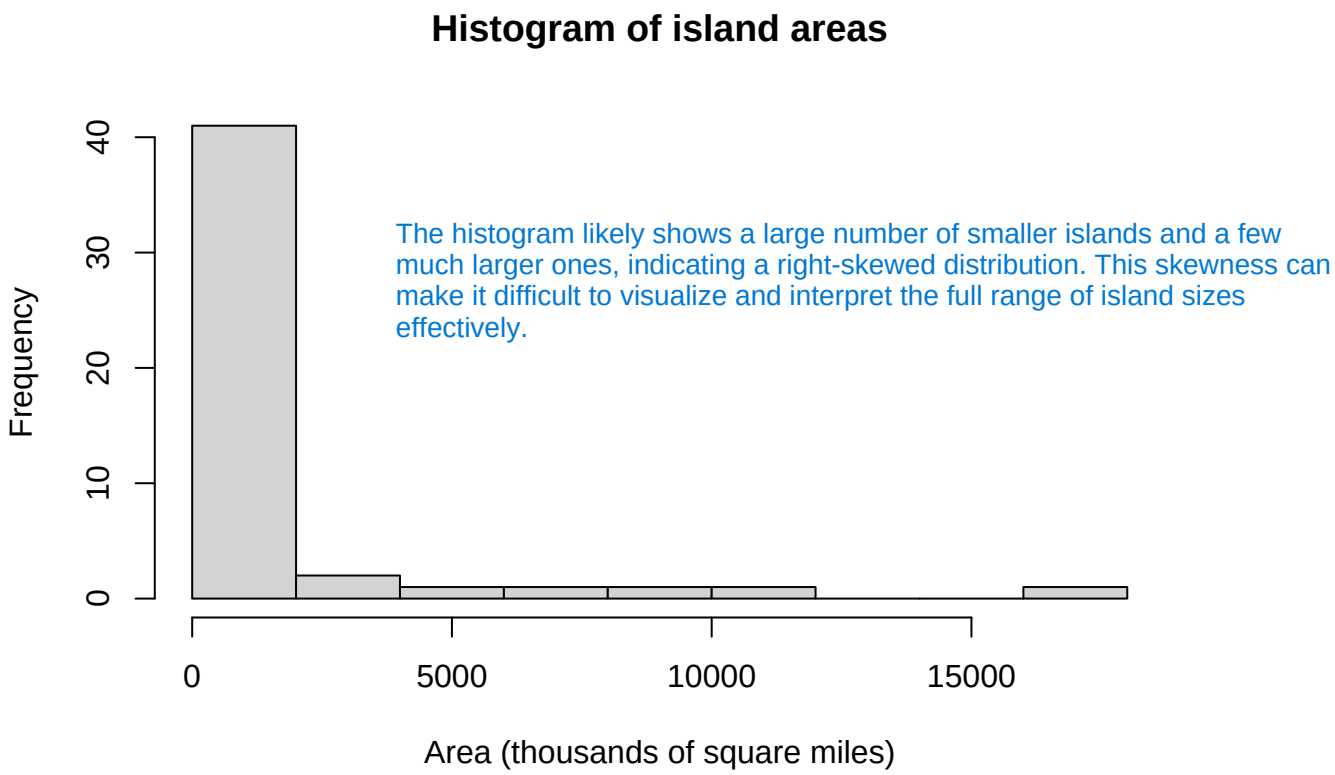
2024-01-17

Exercise 83

a)

From the R documentation description of the dataset, islands contains: "The areas in thousands of square miles of the landmasses which exceed 10,000 square miles."

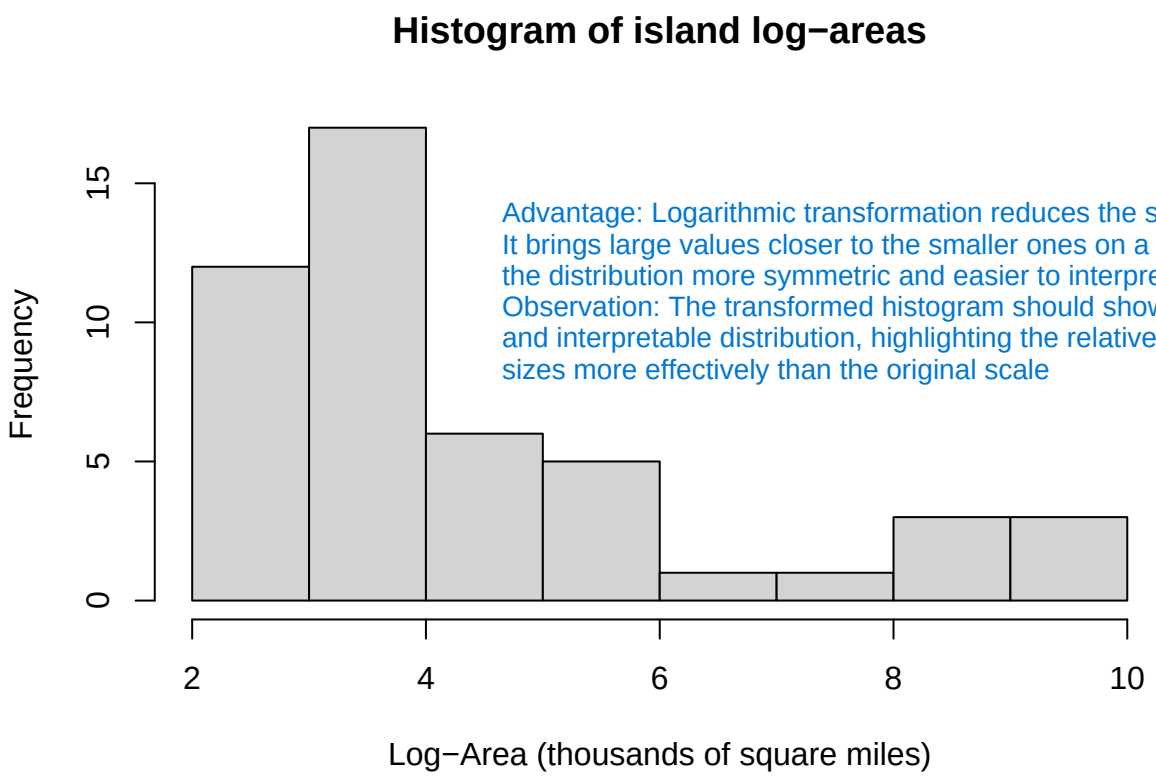
```
hist(islands,
     main = "Histogram of island areas",
     xlab = "Area (thousands of square miles)")
```



It is possible to observe that there is a large amount of islands with a relatively low value of area, while there are some observations that correspond to much higher values. This type of distribution makes it not very useful to visualize the areas in a regular scale.

b)

```
hist(log(islands),
     main = "Histogram of island log-areas",
     xlab = "Log-Area (thousands of square miles)")
```



Advantage: Logarithmic transformation reduces the skewness of the data. It brings large values closer to the smaller ones on a relative scale, making the distribution more symmetric and easier to interpret.
Observation: The transformed histogram should show a more balanced and interpretable distribution, highlighting the relative differences in island sizes more effectively than the original scale

By transforming the areas using logarithms, we are able to reduce the prominent right skew that was observed in point a), and thus get a better visualization of the distribution which was lost due to the significant magnitude changes in the regular scale. However, it is important to take into account this scaling when interpreting results.

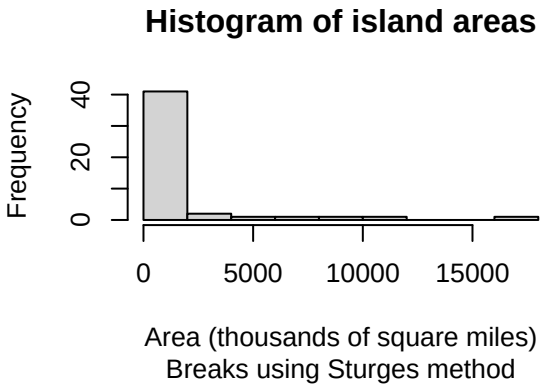
c)

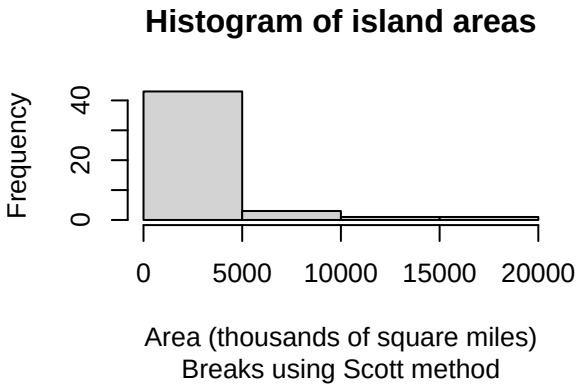
We compare the combinations of regular scale and log scale using Scott and Sturges methods for breaks:

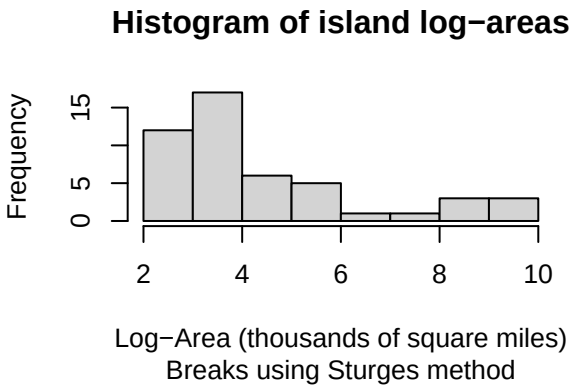
```
par(mfrow = c(2,2))

# Regular scale
hist(islands,
     main = "Histogram of island areas",
     xlab = "Area (thousands of square miles)",
     sub = "Breaks using Sturges method",
     breaks = "Sturges")

hist(islands,
     main = "Histogram of island areas",
     xlab = "Area (thousands of square miles)",
     sub = "Breaks using Scott method",
     breaks = "Scott")
```

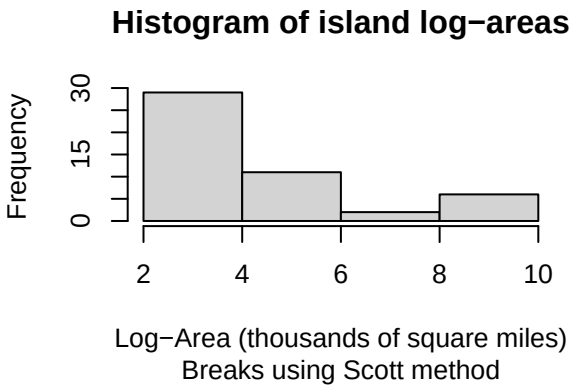
Sturges' Method: Tends to produce fewer bins, which might oversimplify the distribution.
Scott's Method: Often results in more bins, potentially providing a more detailed view of the distribution.
Observation: On both scales, Sturges' method might show a more generalized view, while Scott's method could reveal more nuances in the distribution. The choice between them depends on the level of detail required.

```r
# Log scale
hist(log(islands),
     main = "Histogram of island log-areas",
     xlab = "Log-Area (thousands of square miles)",
     sub = "Breaks using Sturges method",
     breaks = "Sturges")

hist(log(islands),
     main = "Histogram of island log-areas",
     xlab = "Log-Area (thousands of square miles)",
     sub = "Breaks using Scott method",
     breaks = "Scott")
```







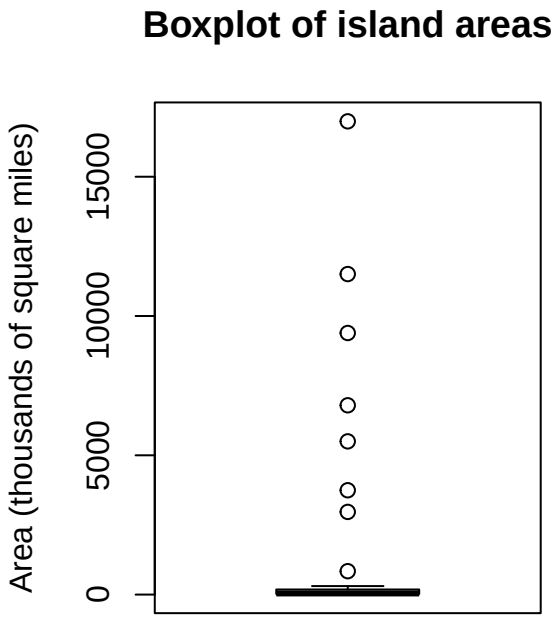


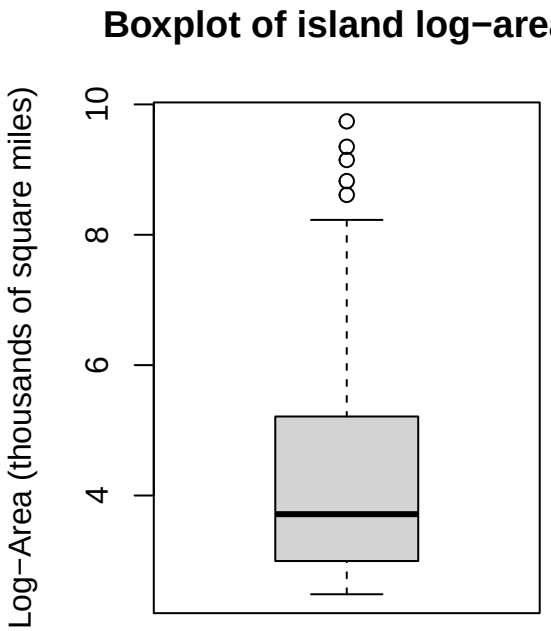
It is possible to see that in both cases Sturges' method results in a higher amount of bins for the histograms. In this particular case, this results in a better visualization of the distribution for both types of scales.

d)

```r
par(mfrow = c(1,2))

boxplot(islands,
        main = "Boxplot of island areas",
        ylab = "Area (thousands of square miles)")
boxplot(log(islands),
        main = "Boxplot of island log-areas",
        ylab = "Log-Area (thousands of square miles)")
```

The boxplot on the original scale might be dominated by outliers (larger islands), making it hard to see the distribution of the smaller islands. The log-scale boxplot should provide a clearer view of the distribution's spread and central tendency, mitigating the effect of outliers.





On the left boxplot (regular scale), we are faced with the same issues as the case of the histogram. Again, we see that plotting using a log-scale improves visualization, for the same reasons stated above. Again, however, it is important to be cautious and take the transformation into account when interpreting results.

e)

In this case we will not arrange the plots in a grid because this would hurt the readability of the island names. Notice that we sort the areas for visualization purposes. Otherwise, the dots look randomly scatterd. We begin plotting the areas in a regular scale:

```
dotchart(sort(islands),
         main = "Dotchart of island areas",
         xlab = "Area (thousands of square miles)",
         cex = 0.5)
```

Normal Distribution in a Histogram
A normal distribution, when represented in a histogram, shows the following characteristics:

Bell Shape: The histogram takes on a bell-shaped curve.
Symmetry: The left and right sides of the histogram are approximately symmetrical around the central peak.
Central Peak: The highest point of the histogram (mode) is at the mean and median of the data.
Tails: The tails on either side of the peak taper off equally and never touch the horizontal axis.
In a perfectly normal distribution, about 68% of the data falls within one standard deviation of the mean, 95% within two standard deviations, and 99.7% within three standard deviations.

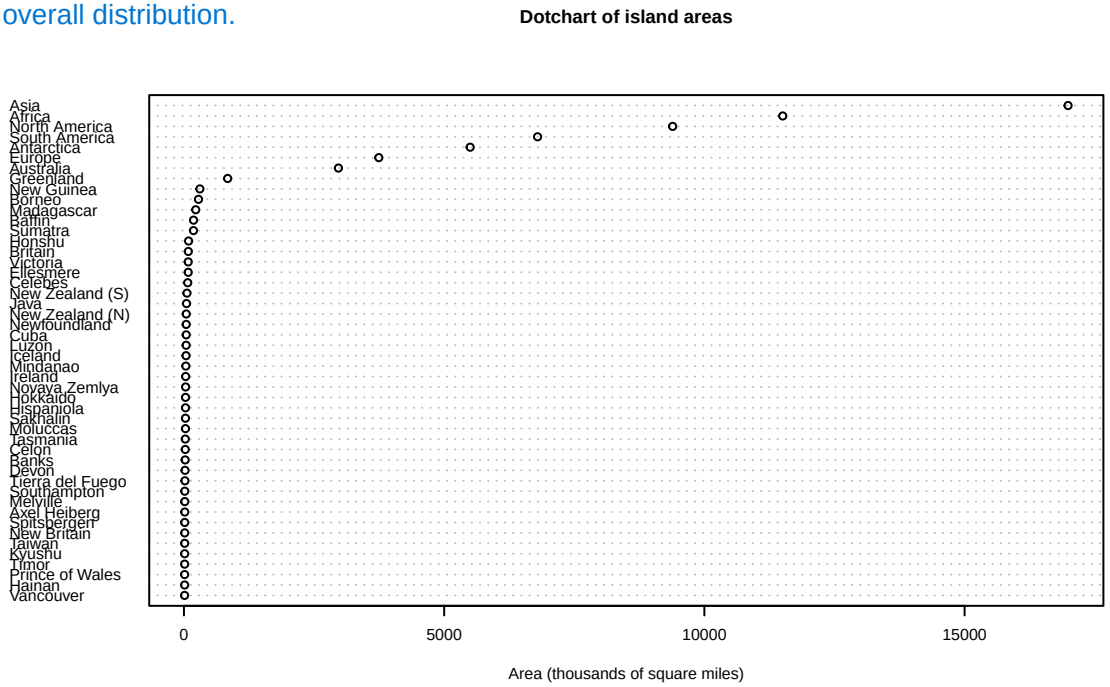
Skewed Distribution in a Histogram
A skewed distribution is one where the tails of the distribution are not symmetrical. There are two types of skewness:

Right-Skewed (Positively Skewed) Distribution:

The tail on the right side (higher values) of the histogram is longer or fatter than the left side.
The mean and median are greater than the mode, and the mean is greater than the median.
It often occurs in situations where the lower bound of data is fixed, but the upper bound is not (e.g., income levels).
Left-Skewed (Negatively Skewed) Distribution:

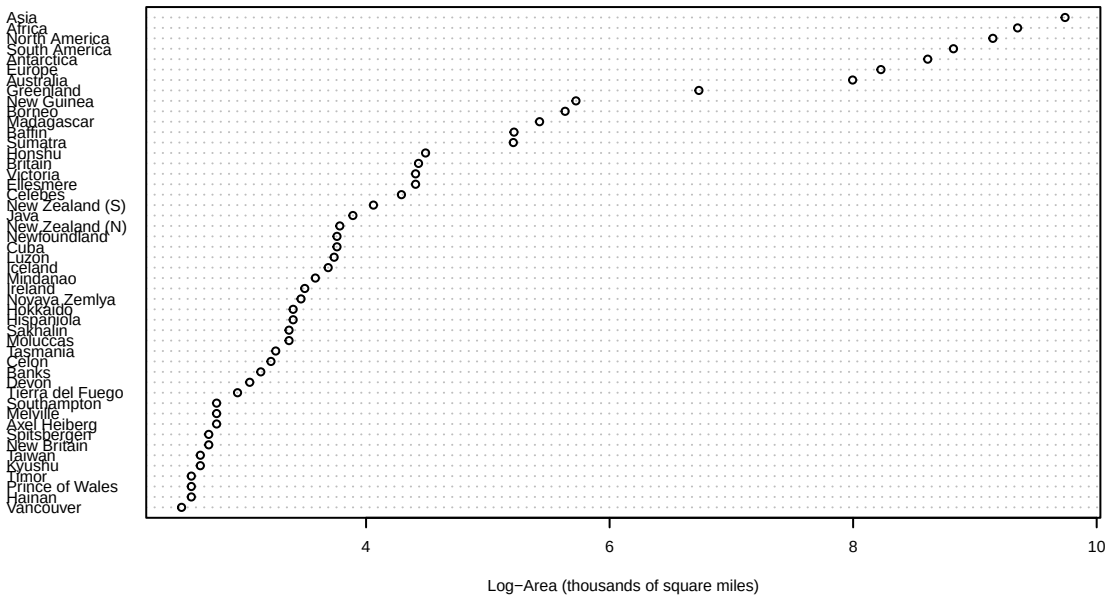
The tail on the left side (lower values) of the histogram is longer or fatter than the right side.
The mean and median are less than the mode, and the mean is less than the median.
It can occur in situations where the upper bound of data is fixed, but the lower bound is not.
What We Derive from These Distributions
Normal Distribution: Indicates that the data is evenly distributed around the mean. It's often an assumption in many statistical methods and models.
Skewed Distribution: Suggests that the data is not evenly distributed and may be influenced by outliers or a natural limit on one side. Analysis might require transformation (e.g., logarithmic) to normalize the data, or special consideration of the skewness in statistical modeling.
In summary, the shape of the histogram provides crucial insights into the underlying distribution of the data, which can guide the choice of statistical methods and interpretations of the data.

The dot-chart provides a clear view of each island's size relative to others, but it might be less effective in conveying the overall distribution.



And follow up plotting using the log-areas:

```
dotchart(sort(log(islands)),
         main = "Dotchart of island log-areas",
         xlab = "Log-Area (thousands of square miles)",
         cex = 0.5)
```



Once again, we are faced with the same visualization issue encountered in the previous sections. Using the raw data has the advantage that large magnitude differences are more evident and clear to see, but we lose interpretation with the smaller (and in this case one could say more typical) observations. Same as before, the log transformation allows us to remove the skew and visualize the data for the smaller-sized islands better.

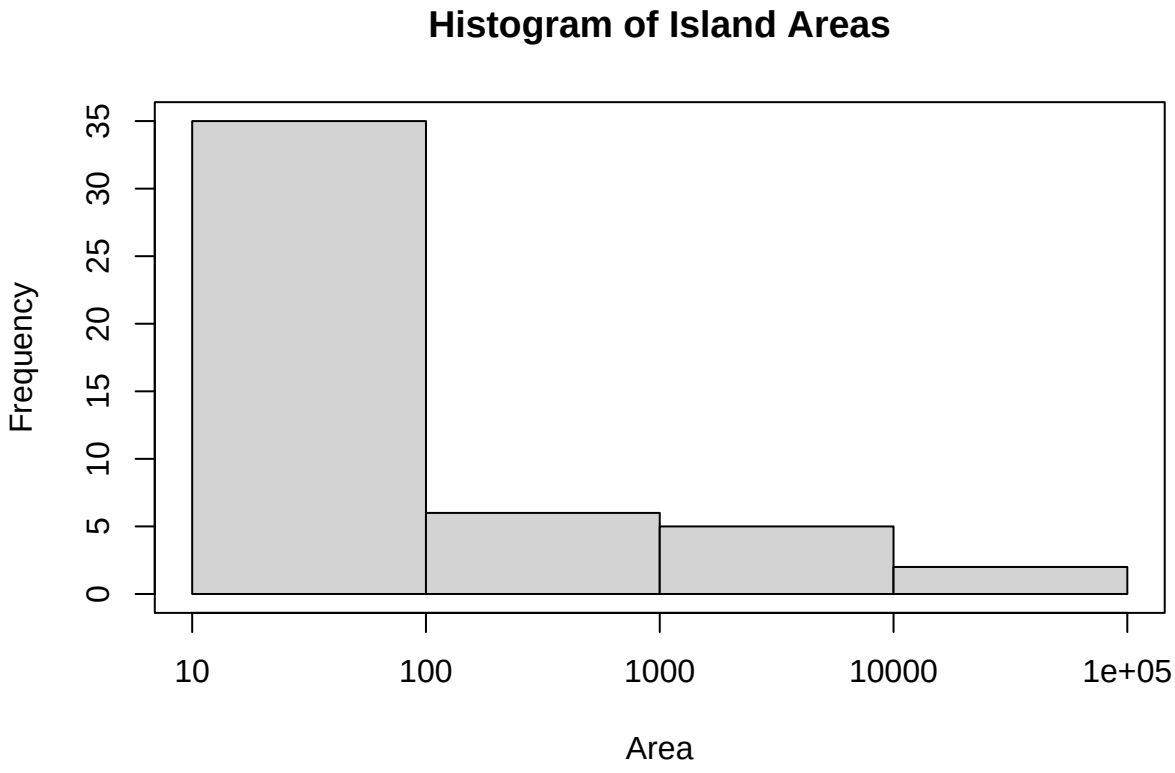
Again, it is important to be cautious and take the transformation into account when interpreting results.

f) For distribution analysis, log-transformed graphs are preferable due to the skewness of the data. For individual comparisons, the dot-chart is effective.

Choosing an adequate visualization relies heavily on the objective behind the data. Depending on what is needed, it might be more useful to look at one kind of plot or another, the same as one kind of scale or another. However, we believe that the dotchart (particularly using the log-transformation, noting again the interpretation disclaimer) is particularly useful for this dataset in a general case (again, this may vary for specific scenarios). This is the case particularly because our dataset is not large, which allows all of the information to be easily readable through this kind of plot. Also, recall the sort of the areas, which makes the information easier to digest. One could argue that if absolute differences are necessary to be visualized, it would in general be best to use the original scale for the values, and we would agree with this in cases with comparable magnitudes. However, as discussed already in this exercise, the magnitude differences make this comparison also difficult when not taking log-scales.

Exercise 84

```
hist(log(islands, 10), breaks = "Scott", axes = FALSE,
     xlab = "Area", main = "Histogram of Island Areas")
axis(1, at = 1 : 5, labels = 10 ^ (1 : 5))
axis(2)
box()
```



a)

The first line creates a histogram of the logarithm of the islands' area, however notice that in this case we do not take the natural logarithm but rather the base 10 logarithm. To assign the breaks for the histogram bins, Scott's method is used. In this same line, we specify axes not to be labelled in the plot yet (however notice

they will be named), as that is done in the following lines, Then, the names for the x-axis and title plot are given, respectively as "Area" and "Histogram of Islands Area".

The second line creates the axis on the first side (hence the 1), which corresponds to the x-axis, specifying the labels to be displayed at each position. Since we are working with base-10-log scale, then the labels are assigned to be the first 5 powers of 10 (excluding 1, the 0-th power), and due to the structure of the histogram, their positions must be from 1 to 5.

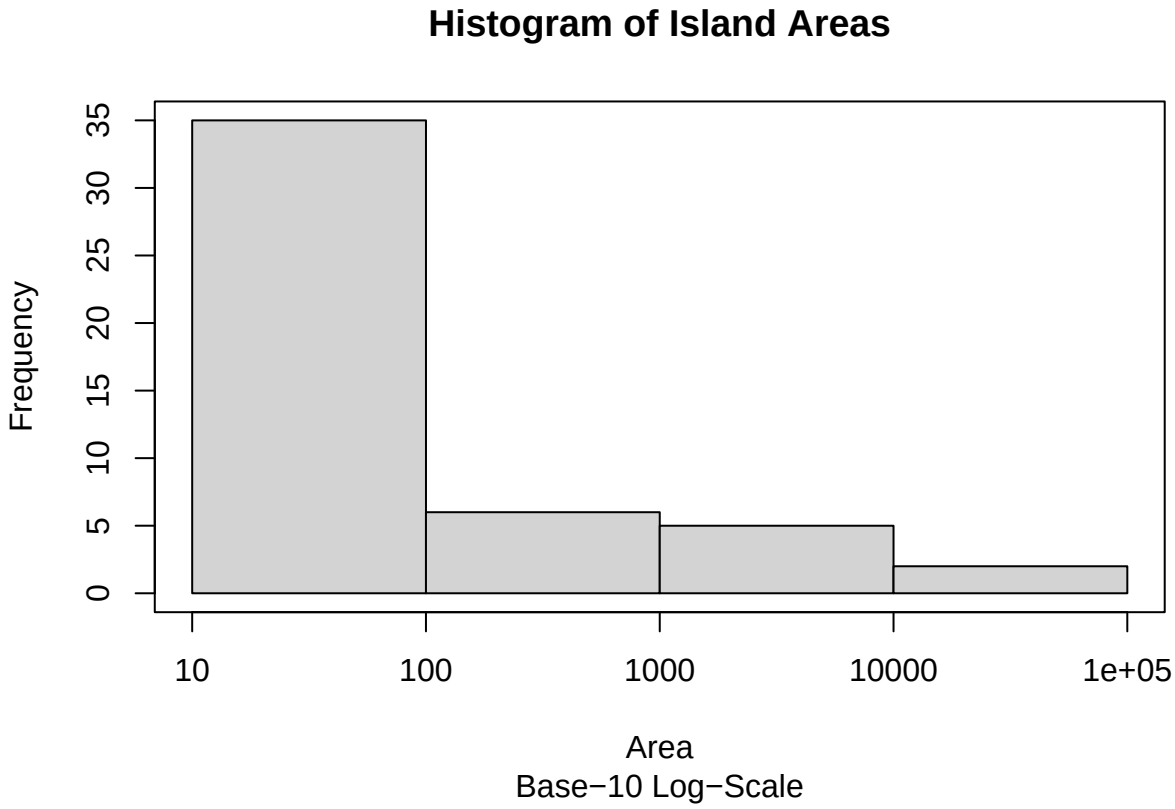
The third line creates the axis on the second side (hence the 2), which corresponds to the y-axis. Notice no other parameter is given, so R creates the default version of this. As stated in the R documentation of the axis function: "When at = NULL, pretty tick mark locations are computed internally. If labels is not specified, the numeric values supplied or calculated for at are converted to character strings as if they were a numeric vector."

Finally, the fourth line creates a box around the plotting area.

b)

We add a subtitle to the previous plot:

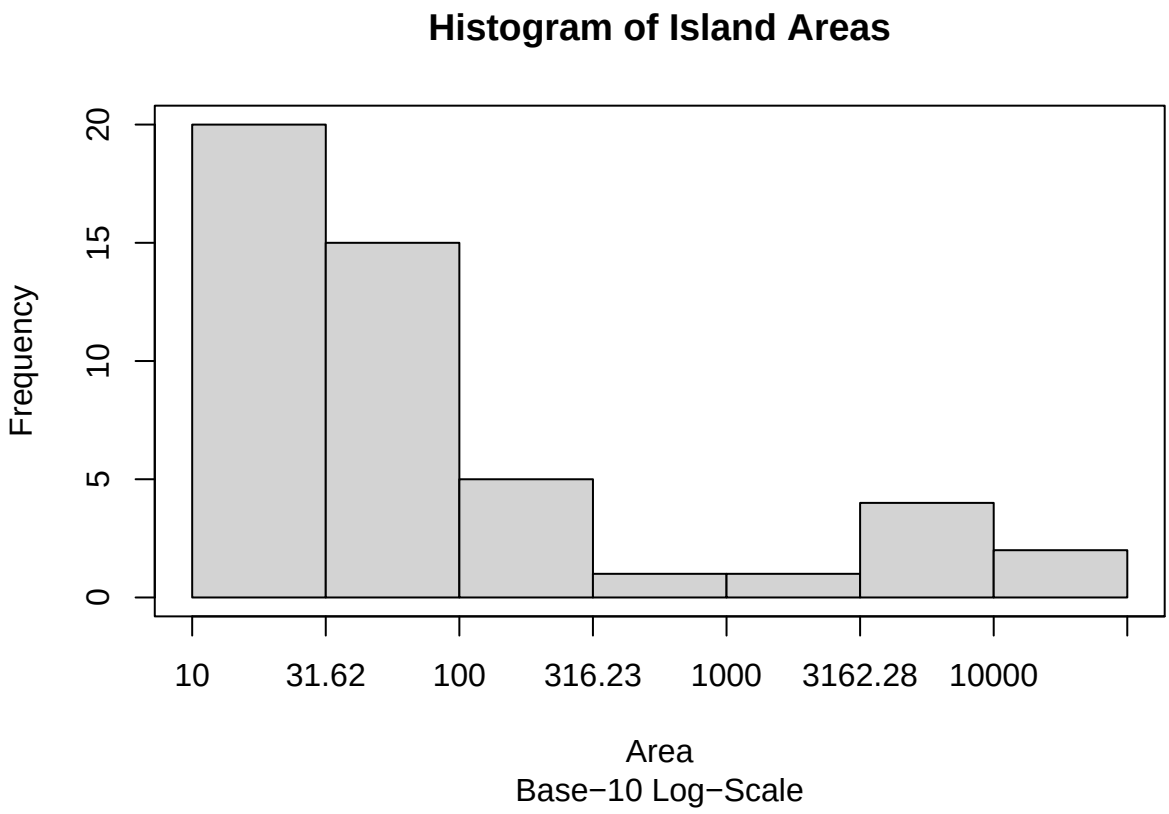
```
hist(log(islands, 10), breaks = "Scott", axes = FALSE,
     xlab = "Area", main = "Histogram of Island Areas",
     sub = "Base-10 Log-Scale")
axis(1, at = 1 : 5, labels = 10 ^ (1 : 5))
axis(2)
box()
```



c)

We modify the code to use Sturges' rule, and adjust for the bin breaks differences:

```r
hist(log(islands, 10), breaks = "Sturges", axes = FALSE,
     xlab = "Area", main = "Histogram of Island Areas",
     sub = "Base-10 Log-Scale")
axis(1, at = seq(1, 5, by = 0.5), labels = round(10 ^ (seq(1, 5, by = 0.5)) , 2 ))
axis(2)
box()
```

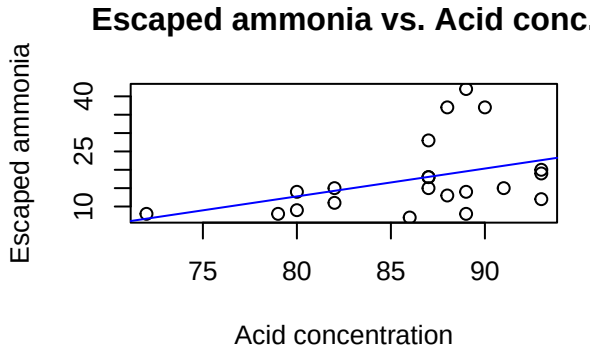


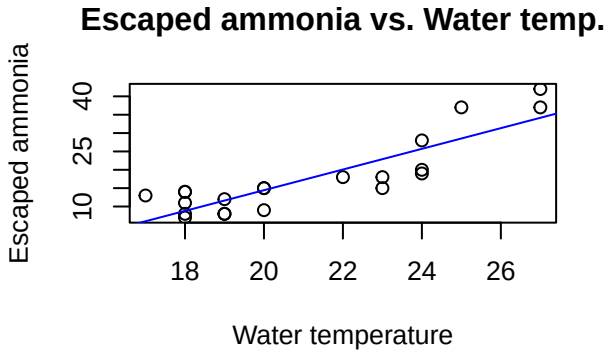
Exercise 85

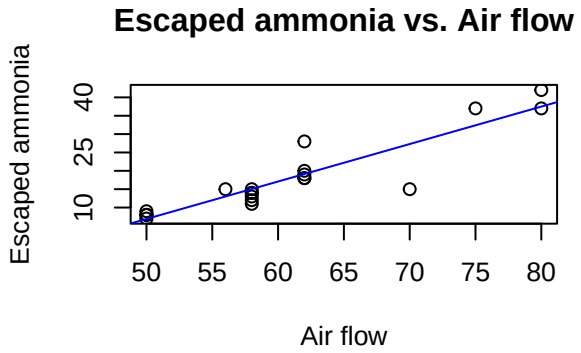
a)

```r
par(mfrow = c(2,2))

with(stackloss, plot(Acid.Conc., stack.loss,
                     main = "Escaped ammonia vs. Acid conc.",
                     xlab = "Acid concentration",
                     ylab = "Escaped ammonia"))
abline(with(stackloss, lm(stack.loss ~ Acid.Conc.)), col = "blue")

with(stackloss, plot(Water.Temp, stack.loss,
                     main = "Escaped ammonia vs. Water temp.",
                     xlab = "Water temperature",
                     ylab = "Escaped ammonia"))
abline(with(stackloss, lm(stack.loss ~ Water.Temp)), col = "blue")

with(stackloss, plot(Air.Flow, stack.loss,
                     main = "Escaped ammonia vs. Air flow",
```

```
                    xlab = "Air flow",
                    ylab = "Escaped ammonia"))
abline(with(stackloss, lm(stack.loss ~ Air.Flow)), col = "blue")
```





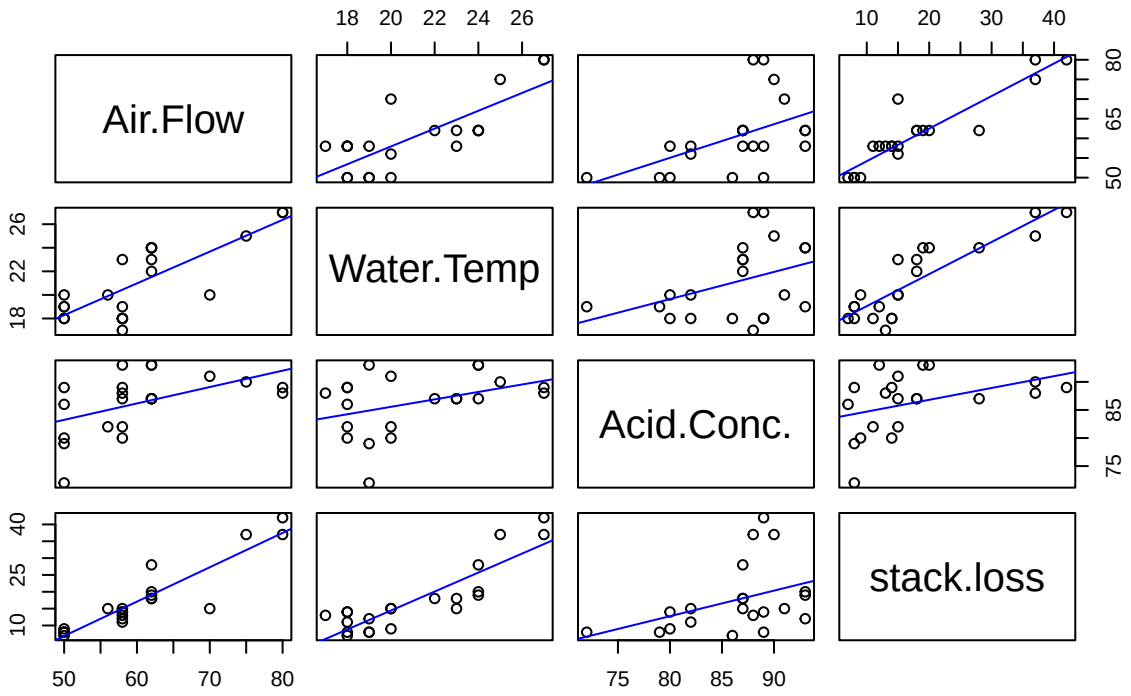


We observe in the previous plots that the amount of escaped ammonia (stack loss) seems to have a linear relation with airflow and a weaker but still somewhat apparent linear relation with water temperature. However it is not the case with acid concentration.

b)

To look at the potential linear relations, we first create a function to plot both the points (i.e. the scatter plot) as well as the linear regression line for each pair of variables, to be fed into the pairs function.

```
panel.lm <- function(x,y){
  points(x,y)
  abline(lm(y~x), col = "blue")
}

pairs(stackloss, panel = panel.lm,
      main = "Scatterplots of all pairwise variable combinations")
```



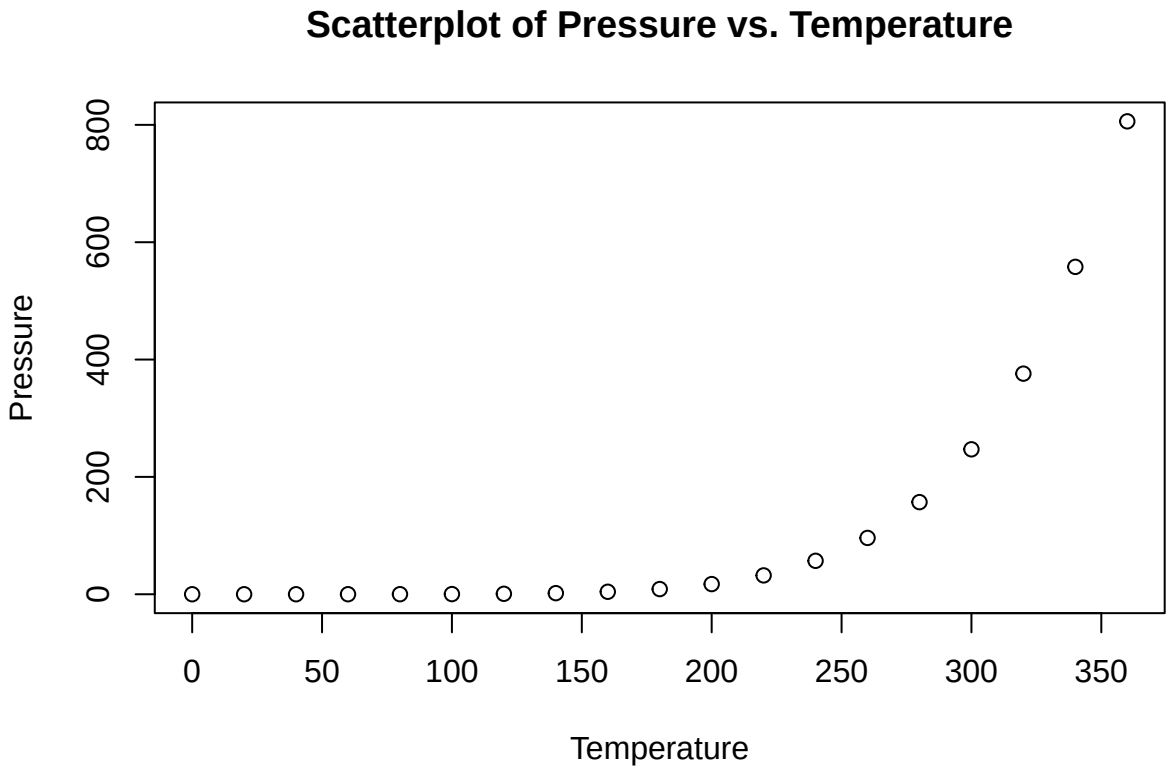
Two pairs of variables exhibit a particularly apparent potential linear relationship: air flow with stack loss and water temperature with stack loss (as discussed in the previous point). Water temperature with air loss seems to also exhibit a somewhat linear relationship, but not as strongly as the first two mentioned pairs. It is hard to discern a linear pattern for the other pairs.

Exercise 86

a)

```
with(pressure, plot(temperature, pressure,
                    main = "Scatterplot of Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Pressure"))
```

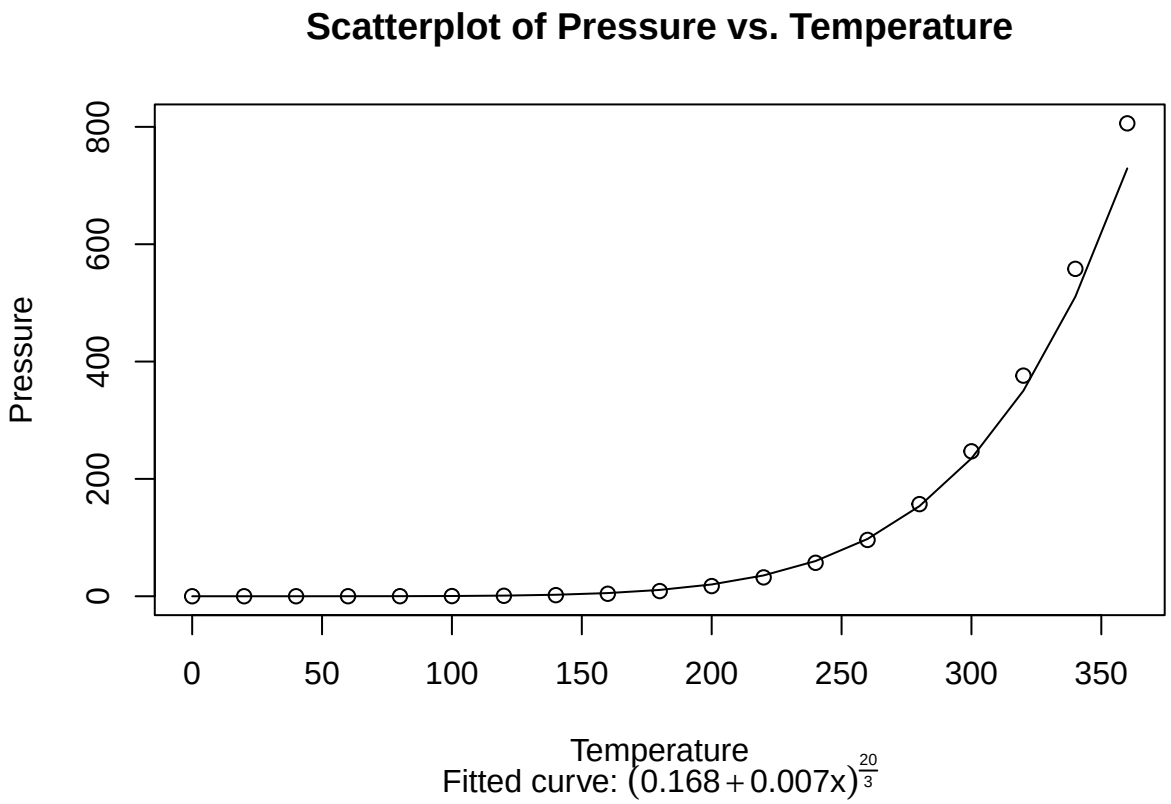
Summary and Conclusions
Non-linear Relationship: The original scatterplot indicates a non-linear relationship between temperature and pressure.
Effectiveness of Transformation: The power transformation makes the relationship more linear, as seen in the scatterplot of transformed data.
Normality of Residuals: The residuals from the original data do not appear to be normally distributed, but the residuals from the transformed data fit better to a normal distribution, though not perfectly.
Choice of Graphs: The use of scatterplots, Q-Q plots, and transformations provides a comprehensive understanding of the relationship between temperature and pressure and the distribution of residuals in the dataset.
This exercise demonstrates the importance of transformations in linearizing relationships and the use of residual analysis to check the assumptions of linear models.



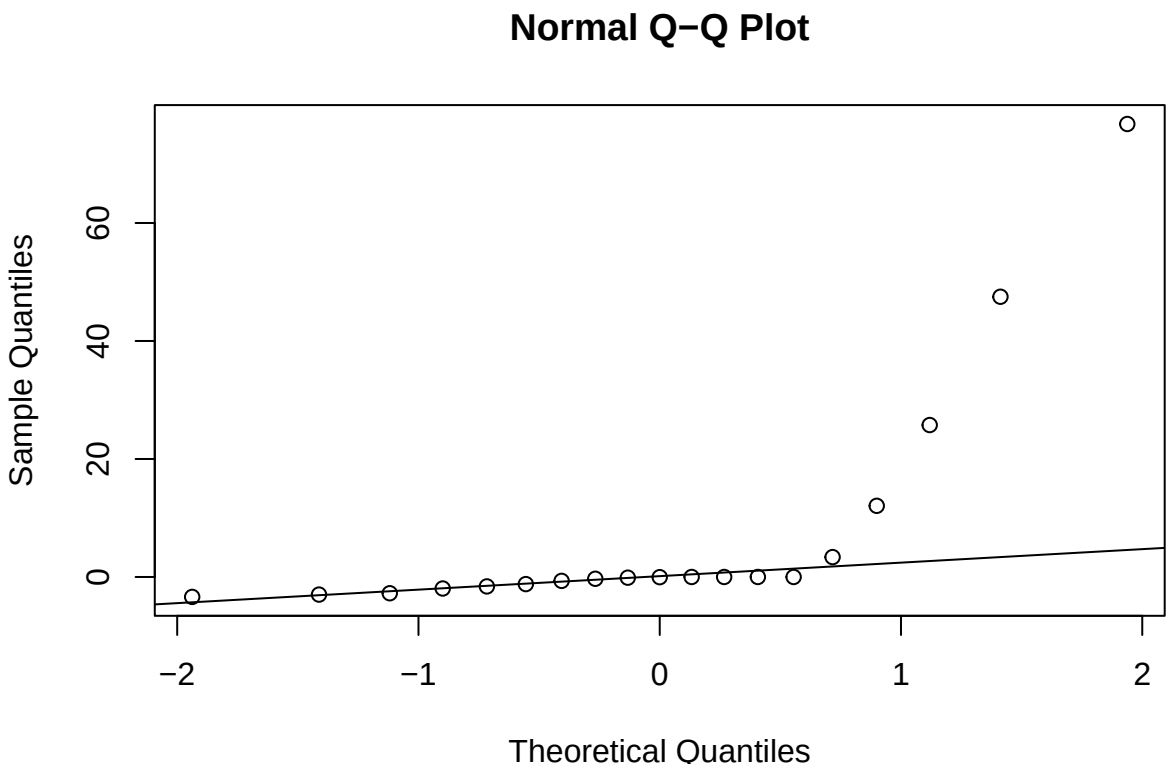
It is visually evident that the relationship between this variables is non-linear, but rather seems exponential.

b)

```
y <- function(x) (0.168 + 0.007*x)^(20/3)

with(pressure, plot(temperature, pressure,
                    main = "Scatterplot of Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Pressure",
                    sub = expression(paste( "Fitted curve: ", y = (0.168 + 0.007*x)^frac(20,3)))))
with(pressure, lines(temperature, y(temperature)))
```



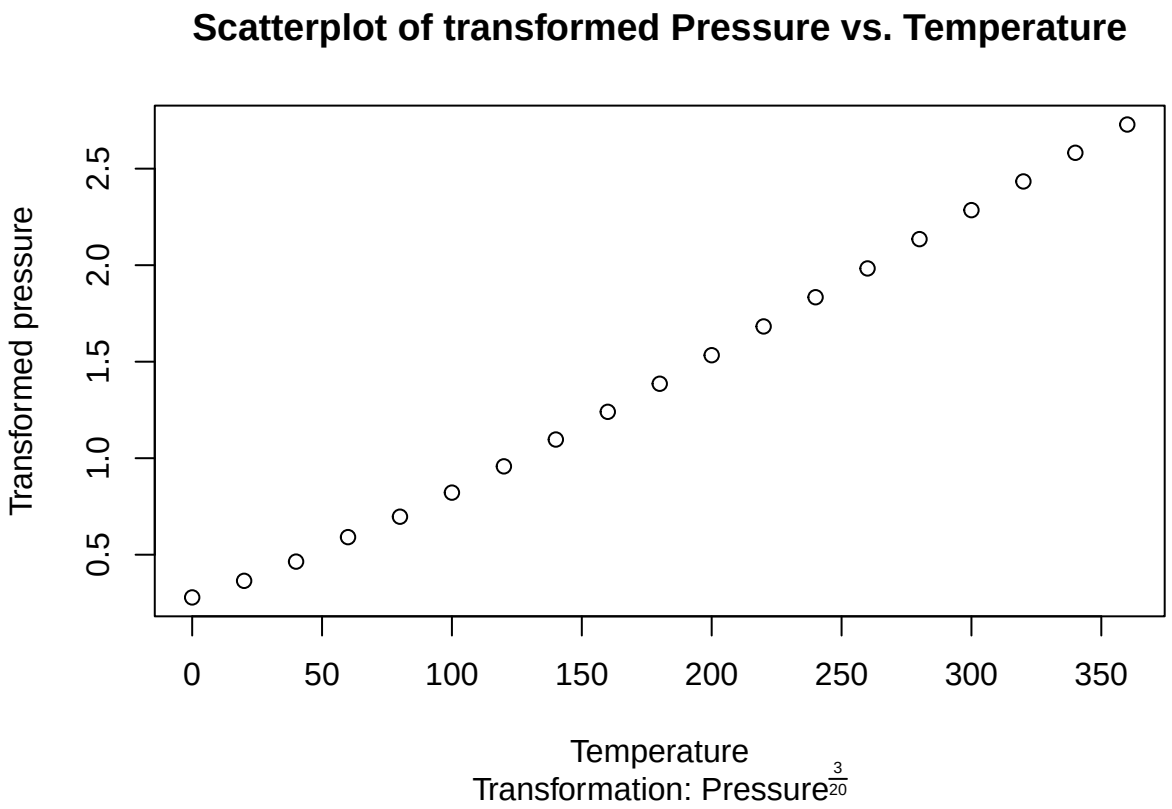

```r
residuals <- with(pressure, pressure - (0.168 + 0.007 * temperature)^(20/3))

qqnorm(residuals)
qqline(residuals)
```



The Q-Q plot indicates the residuals do not follow a normal distribution. As seen in class, if they did, the points in the plot should exhibit a more linear nature, which they clearly don't, particularly in the right tail.

c)

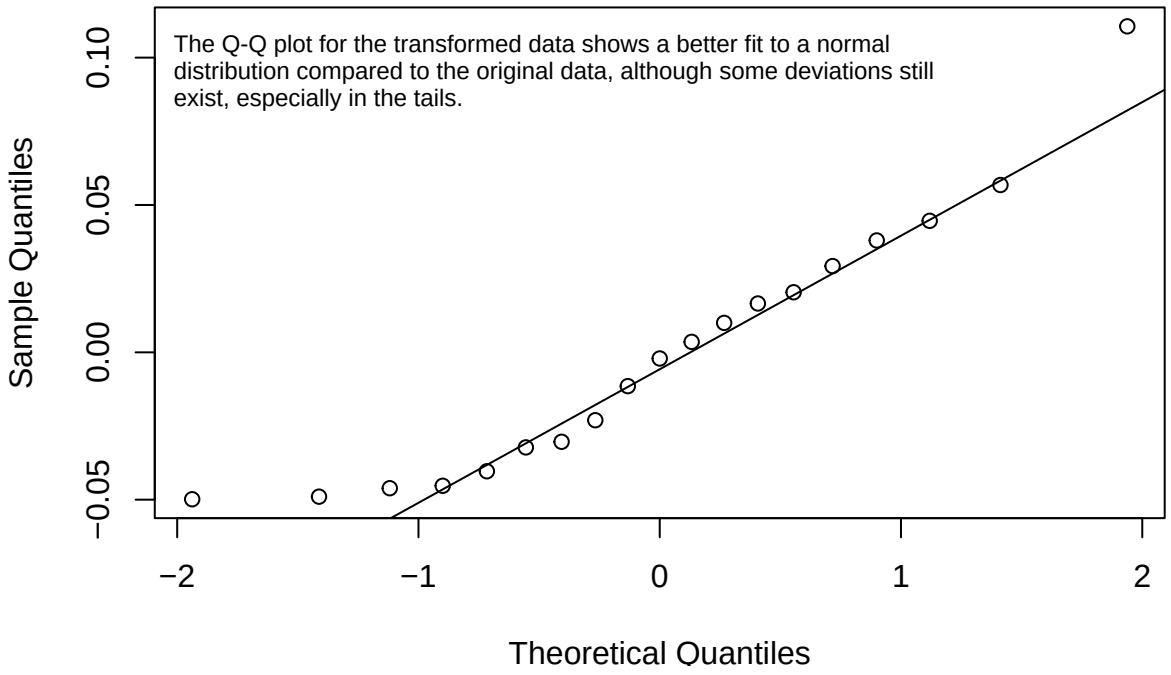
```
with(pressure, plot(temperature, pressure^(3/20),
                    main = "Scatterplot of transformed Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Transformed pressure",
                    sub = expression(paste("Transformation: ", Pressure^frac(3,20)))))
```



We see that when applying the transformation, a much more linear relationship is evident between the variables.

d)

```
# Line prediction
line_p <- lm(pressure^(3/20) ~ temperature, data = pressure)$coefficients

residuals <- with(pressure, pressure^(3/20) - (line_p[1] + line_p[2] * temperature))

qqnorm(residuals)
qqline(residuals)
```



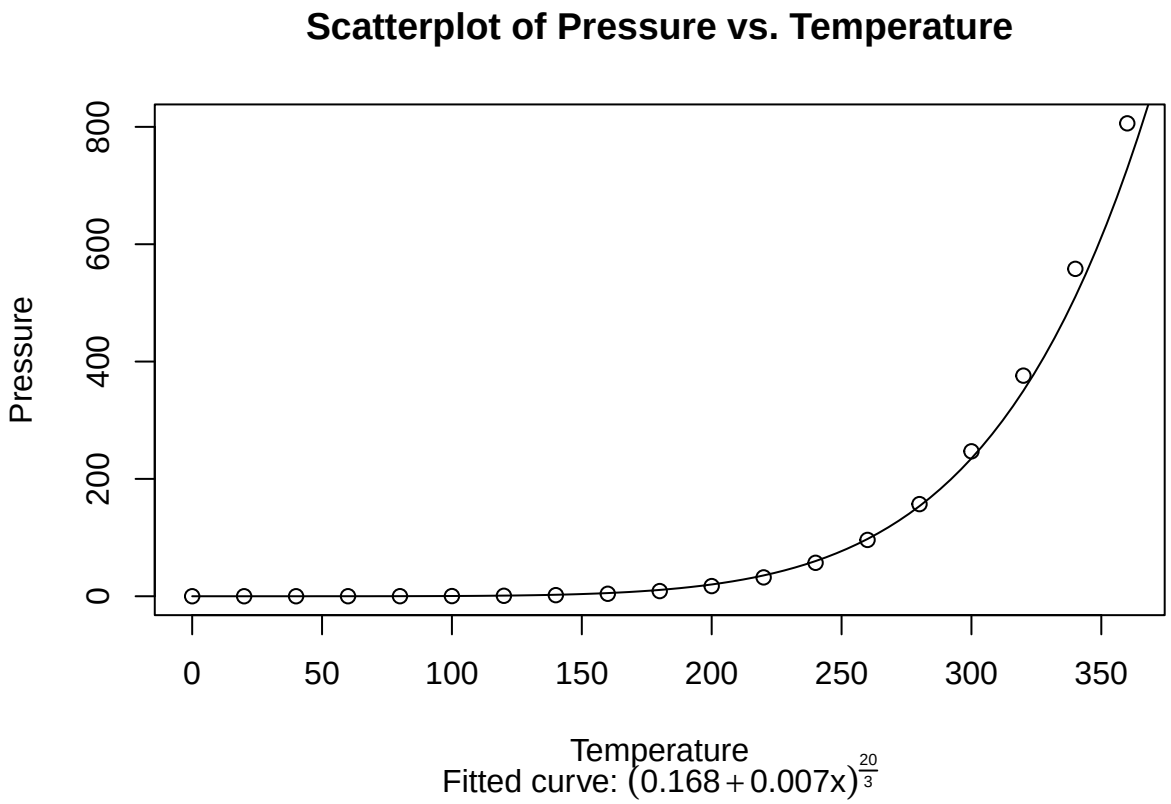
When looking at the residuals using the transformation from last point, the Q-Q plot seems to indicate a better fit to a normal distribution than before. However, we still encounter some deviations at the tails, but these are less prominent than in the non-transformed case.

Exercise 87

a)

```
with(pressure, plot(temperature, pressure,
                    main = "Scatterplot of Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Pressure",
                    sub = expression(paste( "Fitted curve: ", y = (0.168 + 0.007*x)^frac(20,3)))))
curve((0.168 + 0.007 * x)^(20/3), from = 0, to = 400, add = TRUE)
```

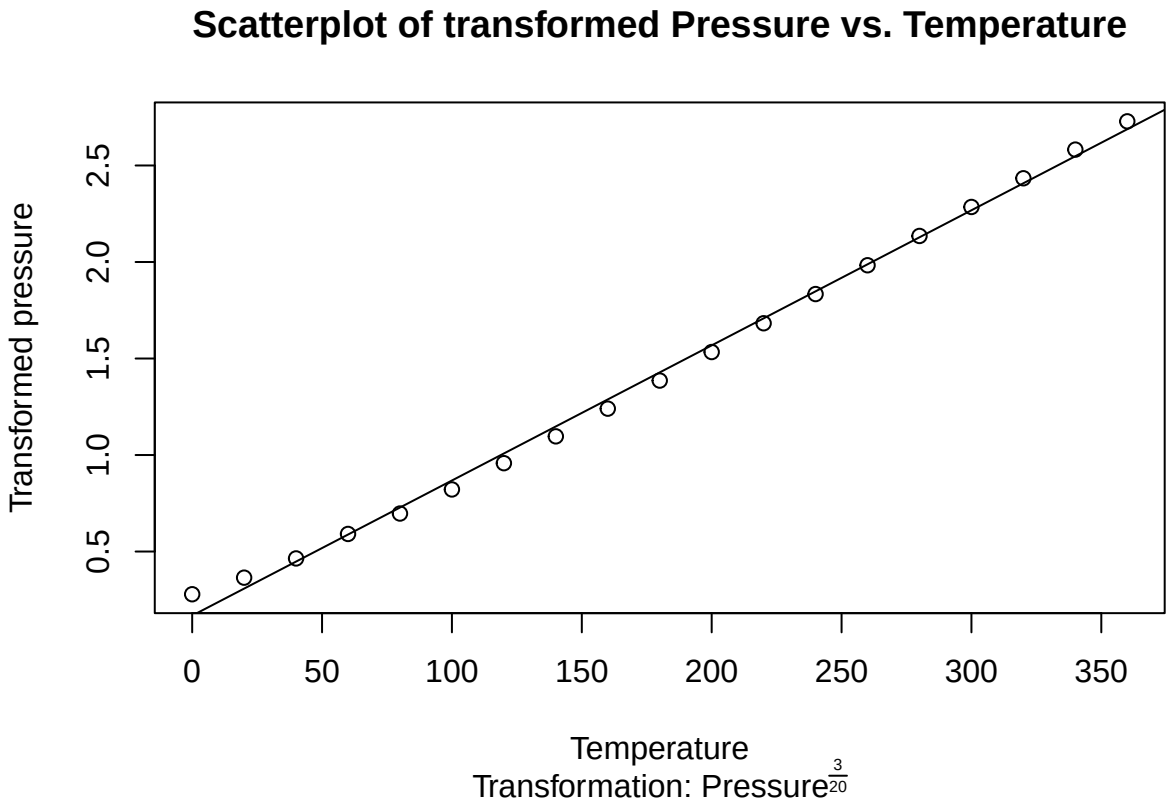
Relationship Between Temperature and Pressure
Non-Linear Relationship: The equation =(0.168+0.007)203y=(0.168+0.007x) 320 suggests a non-linear relationship between temperature (x) and pressure (y). This is because the relationship is not a straight line but rather a curve.
Curve Fit: The curve function is used to fit this non-linear equation to the data points in the scatterplot. The fit of this curve to the data points indicates how well this particular non-linear model describes the relationship between temperature and pressure.
Physical Interpretation: In many physical systems, the relationship between temperature and pressure is indeed non-linear. For example, in ideal gas behavior and other thermodynamic processes, temperature and pressure often have a non-linear relationship.



b)

As seen in the previous exercise, using the transformation we observe a much more linear relationship between the variables. We use the line $y = 0.168 + 0.007x$ to fit the new points, as suggested in point a):

```
with(pressure, plot(temperature, pressure^(3/20),
                    main = "Scatterplot of transformed Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Transformed pressure",
                    sub = expression(paste("Transformation: ", Pressure^frac(3,20)))))
abline(a = 0.168, b = 0.007)
```



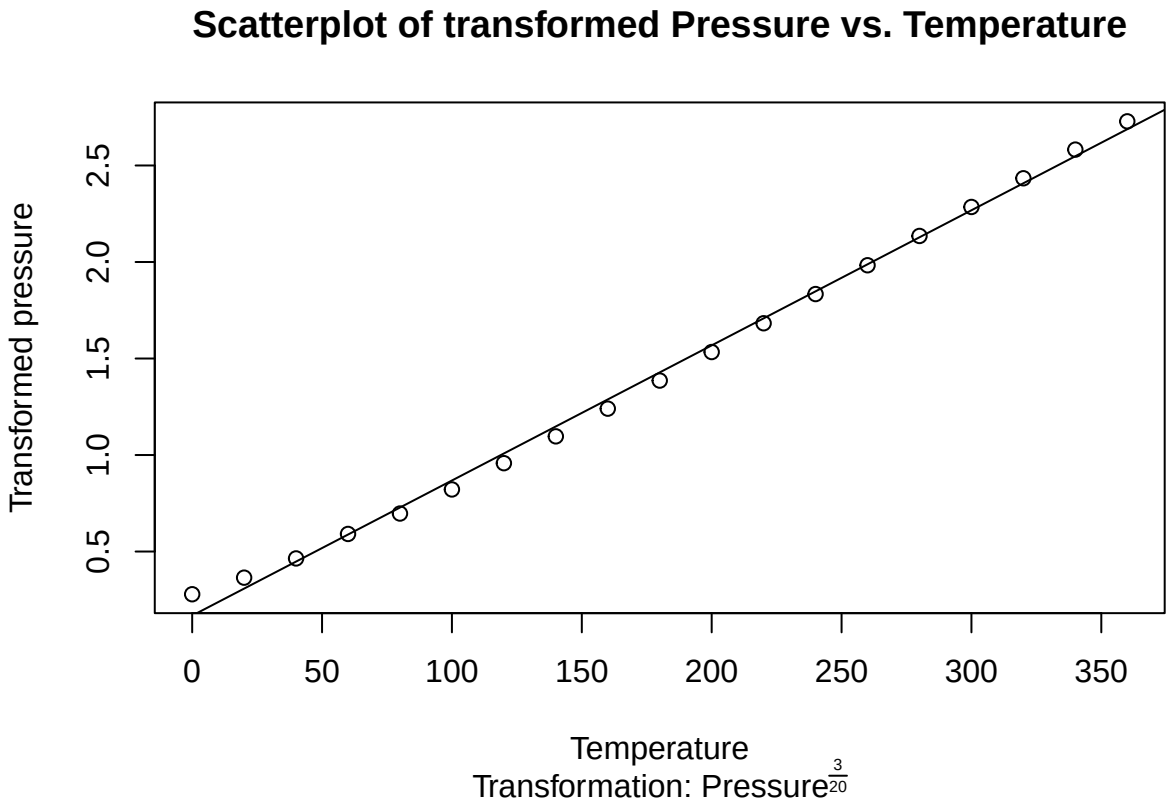
c)

We display the plot again, which in the previous points already contained axis names and annotations, including a title:

```
with(pressure, plot(temperature, pressure^(3/20),
                    main = "Scatterplot of transformed Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Transformed pressure",
                    sub = expression(paste("Transformation: ", Pressure^frac(3,20)))))
abline(a = 0.168, b = 0.007)
```

Conclusion

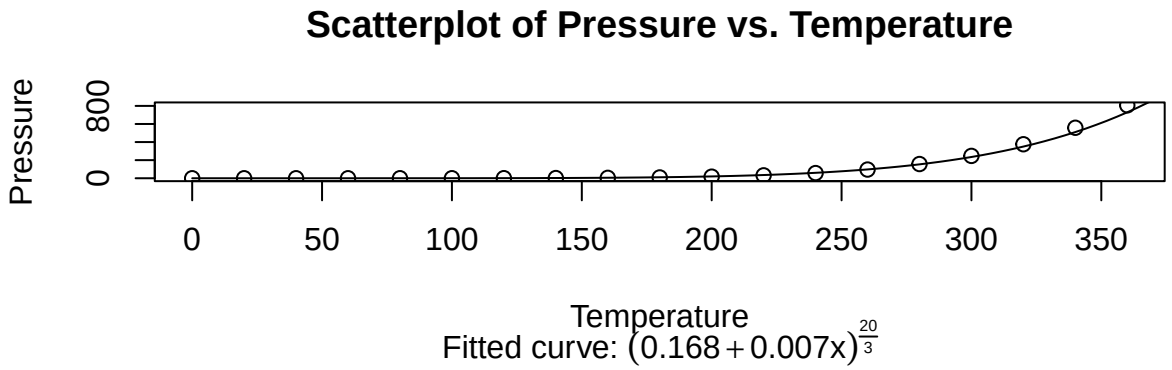
The scatterplot of pressure vs. temperature, along with the fitted non-linear curve, suggests a non-linear relationship between these two variables. This type of analysis is crucial in fields like physics and engineering, where understanding the relationship between different variables is key to modeling and predicting system behavior. The use of scatterplots and curve fitting provides a visual and analytical method to explore and confirm the nature of the relationship between variables.

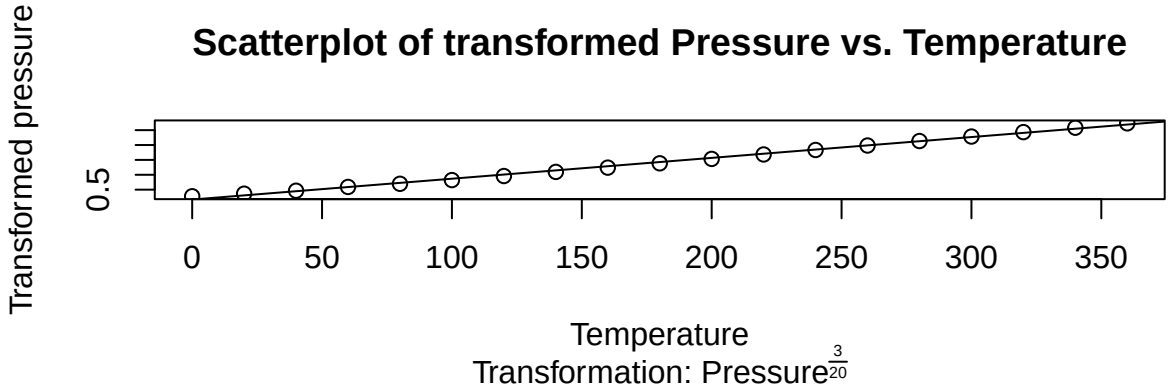


d)

We begin with the 2×1 layout:

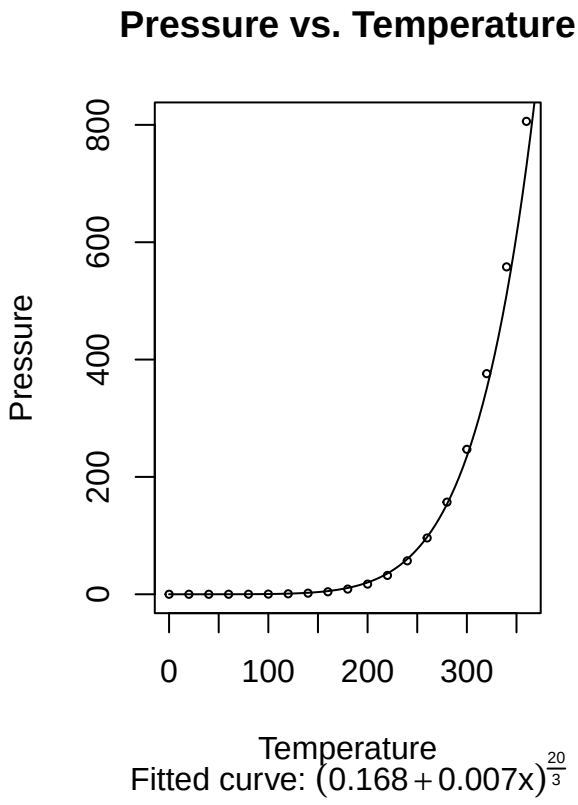
```
par(mfrow = c(2,1))

with(pressure, plot(temperature, pressure,
                    main = "Scatterplot of Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Pressure",
                    sub = expression(paste( "Fitted curve: ", y = (0.168 + 0.007*x)^frac(20,3)))))
curve((0.168 + 0.007 * x)^(20/3), from = 0, to = 400, add = TRUE)

with(pressure, plot(temperature, pressure^(3/20),
                    main = "Scatterplot of transformed Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Transformed pressure",
                    sub = expression(paste("Transformation: ", Pressure^frac(3,20)))))
abline(a = 0.168, b = 0.007)
```

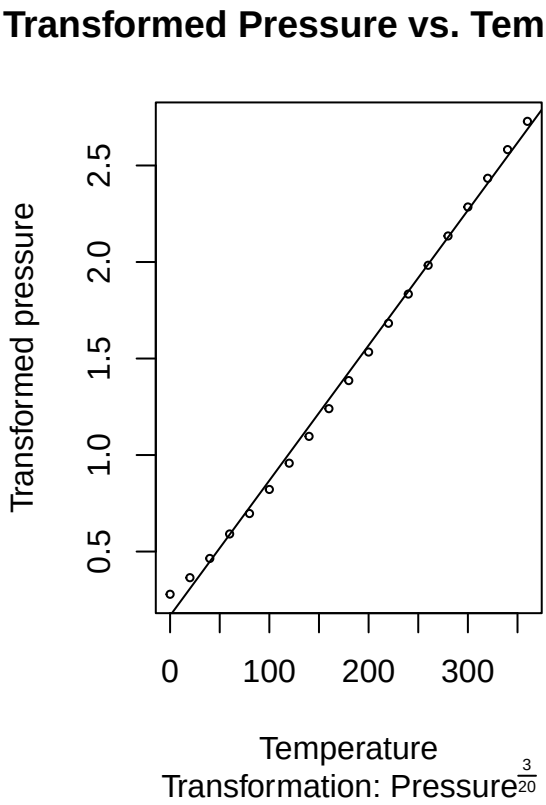




We now do the 1×2 layout:

```
par(mfrow = c(1,2))

with(pressure, plot(temperature, pressure,
                    main = "Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Pressure",
                    sub = expression(paste( "Fitted curve: ", y = (0.168 + 0.007*x)^frac(20,3))),
                    cex = 0.5))
curve((0.168 + 0.007 * x)^(20/3), from = 0, to = 400, add = TRUE)

with(pressure, plot(temperature, pressure^(3/20),
                    main = "Transformed Pressure vs. Temperature",
                    xlab = "Temperature",
                    ylab = "Transformed pressure",
                    sub = expression(paste("Transformation: ", Pressure^frac(3,20))),
                    cex = 0.5))
abline(a = 0.168, b = 0.007)
```





Exercise 88

To solve this exercise we need first to solve the first part of exercise 79, which is a π exercise. First we simulate the times of the claims, using a Poisson process as given by prof. Hornik in class:

```r
set.seed(1111)

PoissonProcess <- function(t, lambda) {
  sort(runif(rpois(1, t * lambda), max = t))
}

claim_times <- PoissonProcess(t = 1,
                              lambda = 100)

claim_amounts <- rgamma(length(claim_times),
                        shape = 2,
                        rate = 2)

premium_t <- 105*claim_times

total_amount_before_claim <- premium_t - cumsum(c(0,claim_amounts[-length(claim_times)]))
total_amount_after_claim <- total_amount_before_claim - claim_amounts

# We add t=0 for visualization and interpretation purposes
if(claim_times[1] != 0){
  claim_times <- c(0, claim_times)
  claim_amounts <- c(0, claim_amounts)
  premium_t <- c(0, premium_t)
```

Claim times refer to the specific moments when insurance claims occur within a given time frame.

Claim amounts are the monetary values associated with each insurance claim. Each claim amount represents the cost that the insurance company needs to pay for that particular claim.

In the simulation, premiums are modeled in a simplified manner to represent the continuous inflow of funds to the insurance company from its policyholders. This model helps in understanding the financial dynamics of an insurance operation, particularly how the company's financial status is affected by the interplay of regular premium collection and irregular claim payouts.

Explanation: In the simulation, the premium is calculated as a function of time. The code assumes that the premium is collected continuously over time at a constant rate. Here, the rate is implied to be 105 units of money per unit of time.
Interpretation: The variable premium_t represents the cumulative premium collected up to each claim time. It's calculated by multiplying this constant rate (105) by the time elapsed (claim_times). This approach simplifies the real-world scenario where premiums are typically collected at discrete intervals (e.g., monthly).
Role of Premiums in the Simulation
Financial Tracking: The simulation tracks the total amount of money the insurance company has over time, considering both the premiums collected and the claims paid out.
Impact on Company's Finances: Premiums increase the total amount of money the company has, while claim payouts decrease it. The simulation visualizes this dynamic, showing how the company's financial status fluctuates over time.

```
  total_amount_before_claim <- c(0, total_amount_before_claim)
  total_amount_after_claim <- c(0, total_amount_after_claim)
}

claim_sim <- data.frame(t = claim_times,
                        Premium = premium_t,
                        Claim = claim_amounts,
                        PreClaim_Total = total_amount_before_claim,
                        PostClaim_Total = total_amount_after_claim)

head(claim_sim, 15)
```

```
##             t   Premium     Claim PreClaim_Total PostClaim_Total
## 1  0.00000000  0.000000 0.0000000       0.000000       0.0000000
## 2  0.00169020  0.177471 0.8945953       0.177471      -0.7171243
## 3  0.01833906  1.925601 0.4089391       1.031006       0.6220665
## 4  0.02617067  2.747920 0.1591923       1.444385       1.2851932
## 5  0.03460843  3.633885 0.4715241       2.171159       1.6996345
## 6  0.03976418  4.175239 0.4493722       2.240988       1.7916158
## 7  0.06044078  6.346281 0.5084475       3.962658       3.4542109
## 8  0.06248020  6.560421 0.2258738       3.668351       3.4424768
## 9  0.06429438  6.750910 0.5460520       3.632966       3.0869139
## 10 0.07243558  7.605736 0.4021237       3.941740       3.5396160
## 11 0.09562712 10.040848 0.9420930       5.974728       5.0326348
## 12 0.10597633 11.127514 0.5226367       6.119301       5.5966646
## 13 0.10734540 11.271266 2.5502277       5.740417       3.1901891
## 14 0.11346912 11.914258 0.7163552       3.833180       3.1168251
## 15 0.11498608 12.073538 0.3798203       3.276106       2.8962855
```

Using the previous data, we proceed to plot:

```
plot(1, type="n",
     main = "Auto insurance simulation: Total amount of money at time t",
     xlab=expression(t),
     ylab="Total amount of money",
     xlim=c(0, 1),
     ylim=c(min(total_amount_after_claim)-0.5,
            max(total_amount_before_claim)+0.5))
for(i in 1:(nrow(claim_sim)-1)){
  segments(x0 = claim_sim$t[i],
           y0 = claim_sim$PostClaim_Total[i],
           x1 = claim_sim$t[i+1],
           y1 = claim_sim$PreClaim_Total[i+1])

  points(claim_sim$t[i], claim_sim$PostClaim_Total[i], pch = 19)
  points(claim_sim$t[i+1], claim_sim$PreClaim_Total[i+1])
}
abline(h = 0)
```

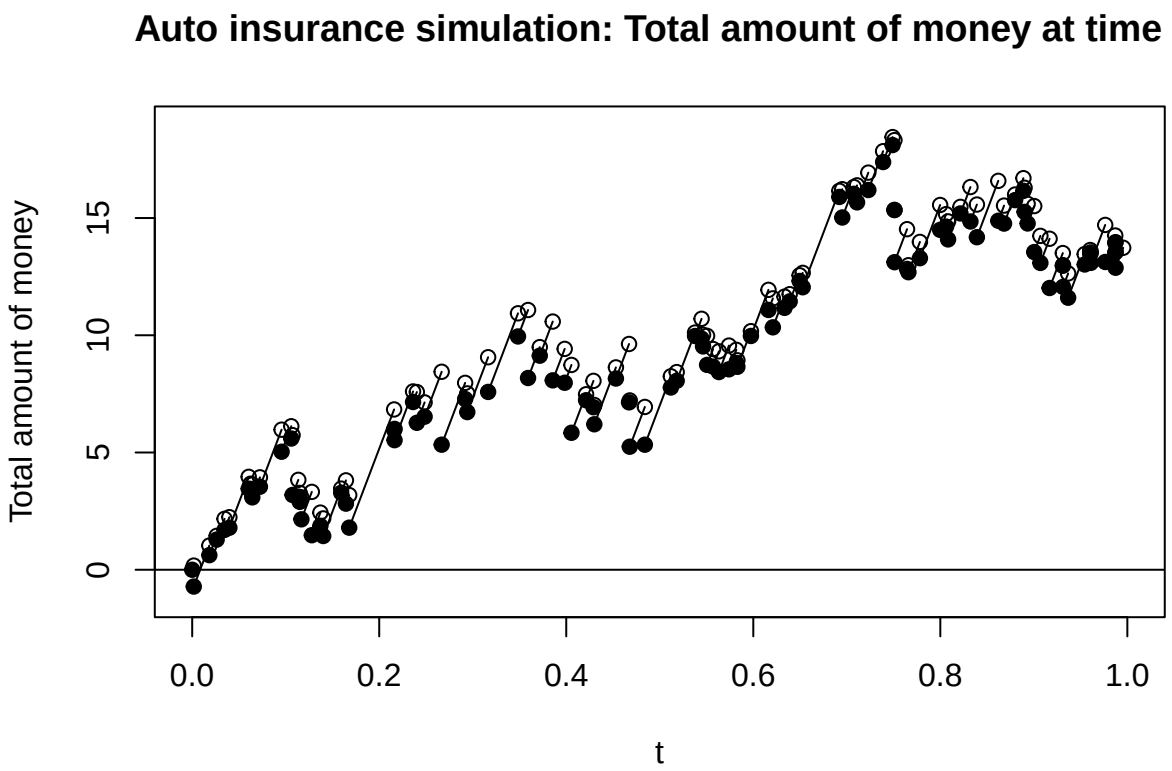
Simulation Setup: The code simulates an insurance company's financial status over a period of time (t). It involves generating claim times and amounts, and calculating the total amount of money before and after each claim.

Claim Times: Generated using a Poisson process, which models the times at which claims occur.

Claim Amounts: Simulated using a gamma distribution, representing the monetary value of each claim.

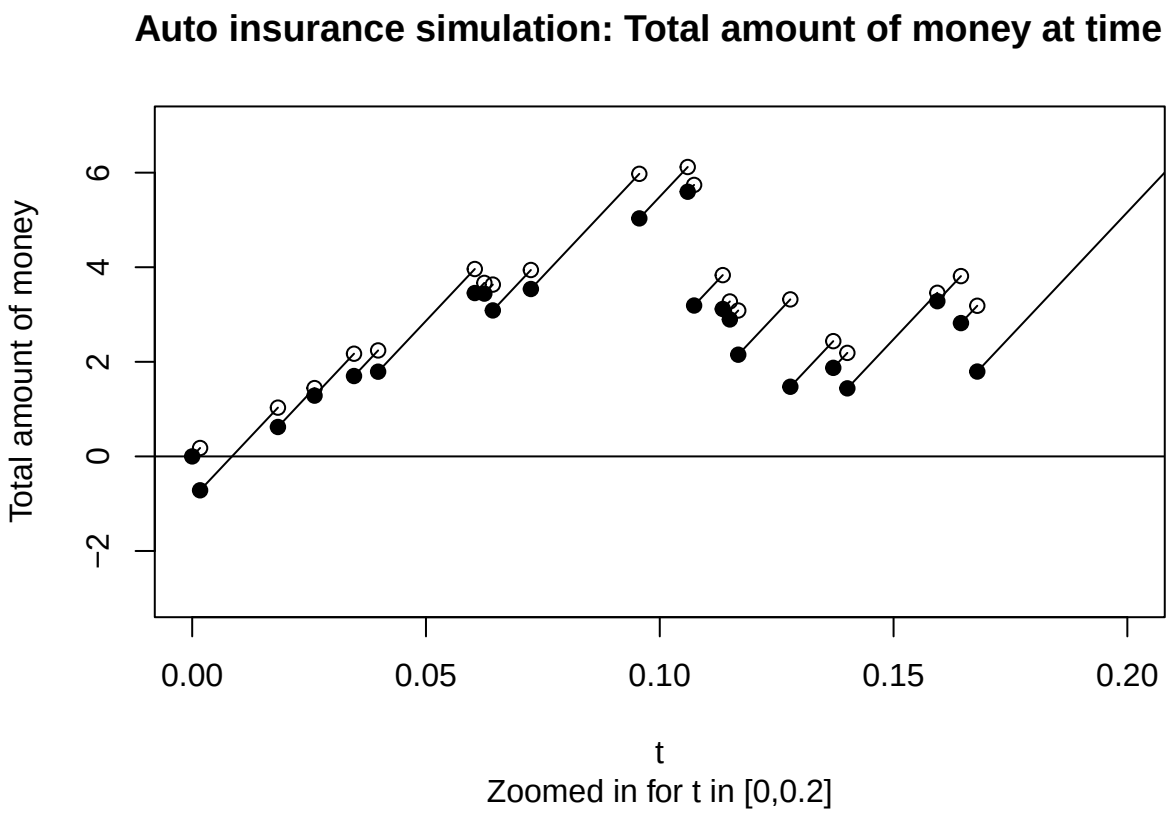
Premiums Collected: Calculated as a function of time, assuming a constant rate of premium collection.

Total Amount Before and After Claims: Calculated to show the company's financial status at each point in time.



We also include a zoomed in version of the plot to show the behavior of the discontinuities:

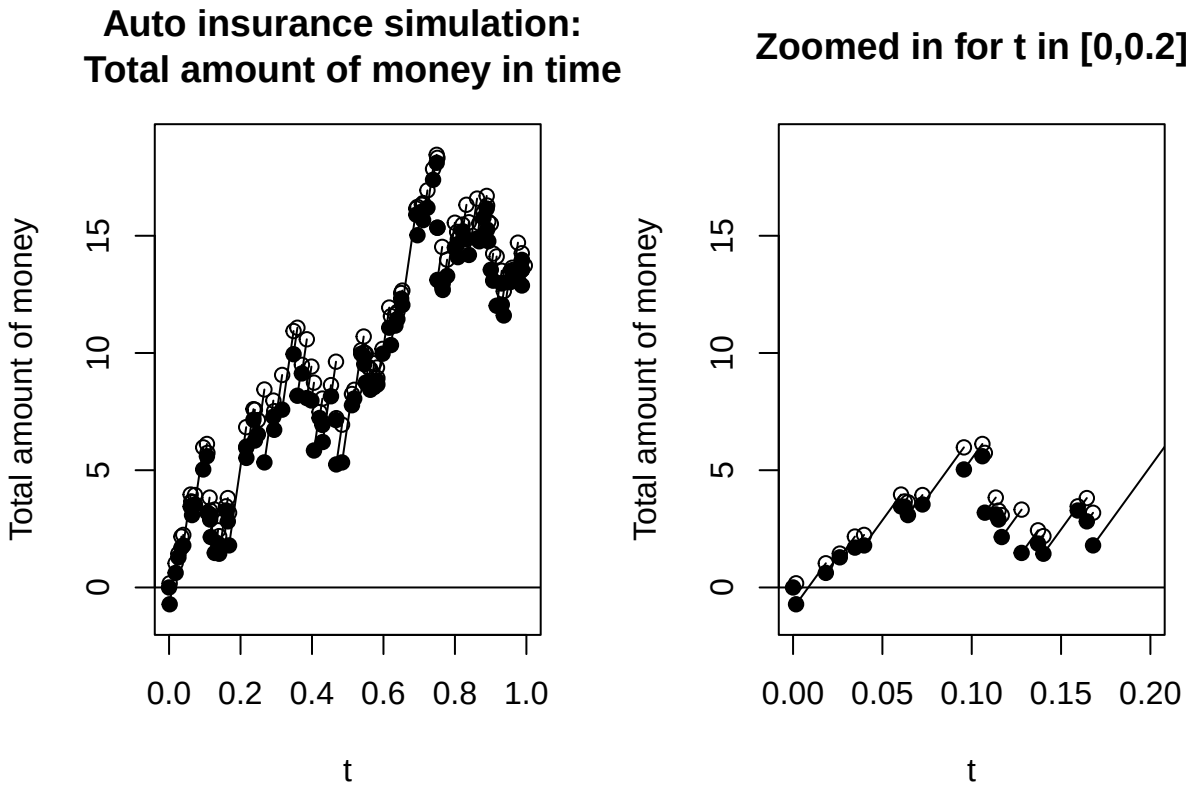
```
plot(1, type="n",
     main = "Auto insurance simulation: Total amount of money at time t",
     sub = "Zoomed in for t in [0,0.2]",
     xlab = expression(t),
     ylab = "Total amount of money",
     xlim = c(0, 0.2),
     ylim = c(-3, 7))
for(i in 1:(nrow(claim_sim)-1)){
  segments(x0 = claim_sim$t[i],
           y0 = claim_sim$PostClaim_Total[i],
           x1 = claim_sim$t[i+1],
           y1 = claim_sim$PreClaim_Total[i+1])

  points(claim_sim$t[i], claim_sim$PostClaim_Total[i], pch = 19)
  points(claim_sim$t[i+1], claim_sim$PreClaim_Total[i+1])
}
abline(h = 0)
```



And we add bith in a single figure as shown in the exercise sheet:

```
par(mfrow = c(1,2))

plot(1, type="n",
     main = "Auto insurance simulation: \n Total amount of money in time",
     xlab=expression(t),
     ylab="Total amount of money",
     xlim=c(0, 1),
     ylim=c(min(total_amount_after_claim)-0.5,
            max(total_amount_before_claim)+0.5))
for(i in 1:(nrow(claim_sim)-1)){
  segments(x0 = claim_sim$t[i],
           y0 = claim_sim$PostClaim_Total[i],
           x1 = claim_sim$t[i+1],
           y1 = claim_sim$PreClaim_Total[i+1])

  points(claim_sim$t[i], claim_sim$PostClaim_Total[i], pch = 19)
  points(claim_sim$t[i+1], claim_sim$PreClaim_Total[i+1])
}
abline(h = 0)

plot(1, type="n",
     main = "Zoomed in for t in [0,0.2]",
     xlab = expression(t),
     ylab = "Total amount of money",
     xlim = c(0, 0.2),
     ylim=c(min(total_amount_after_claim)-0.5,
            max(total_amount_before_claim)+0.5))
```

```r
for(i in 1:(nrow(claim_sim)-1)){
  segments(x0 = claim_sim$t[i],
           y0 = claim_sim$PostClaim_Total[i],
           x1 = claim_sim$t[i+1],
           y1 = claim_sim$PreClaim_Total[i+1])

  points(claim_sim$t[i], claim_sim$PostClaim_Total[i], pch = 19)
  points(claim_sim$t[i+1], claim_sim$PreClaim_Total[i+1])
}
abline(h = 0)
```



Analysis of the Plots

Understanding Financial Dynamics: The plots show how the total amount of money changes over time as claims are paid out and premiums are collected. Each drop in the graph represents a claim being paid, and the rising segments indicate periods where premiums are being collected without any claims.

Importance of Visualization: These visualizations are crucial for understanding the dynamics of an insurance company's cash flow. They illustrate the impact of claims on the company's financial position over time and highlight periods of higher risk when the total amount of money decreases significantly.

Zoomed-In View: The zoomed-in plot provides a closer look at the financial fluctuations in a specific time frame, offering detailed insights into the company's short-term financial stability.