# Week4.R

shadankhan

2024-08-19

```
# Ques 1 Define the string with your name, unit name, and task name
my_string <- "Name: Shadan Khan, Unit: Statistical Data Analysis, Task: Calculas Exploratory Week 4"

# Print the string
print(my_string)
```

```
## [1] "Name: Shadan Khan, Unit: Statistical Data Analysis, Task: Calculas Exploratory Week 4"
```

```
#Ques 2

data<-read.csv('weather (2).csv')

#printing head

head(data)
```

```
##   origin year month day hour  temp  dewp humid wind_dir wind_speed wind_gust
## 1    EWR 2013     1   1    1 39.02 26.06 59.37      270   10.35702        NA
## 2    EWR 2013     1   1    2 39.02 26.96 61.63      250    8.05546        NA
## 3    EWR 2013     1   1    3 39.02 28.04 64.43      240   11.50780        NA
## 4    EWR 2013     1   1    4 39.92 28.04 62.21      250   12.65858        NA
## 5    EWR 2013     1   1    5 39.02 28.04 64.43      260   12.65858        NA
## 6    EWR 2013     1   1    6 37.94 28.04 67.21      240   11.50780        NA
##   precip pressure visib            time_hour
## 1      0   1012.0    10 2013-01-01T06:00:00Z
## 2      0   1012.3    10 2013-01-01T07:00:00Z
## 3      0   1012.5    10 2013-01-01T08:00:00Z
## 4      0   1012.2    10 2013-01-01T09:00:00Z
## 5      0   1011.9    10 2013-01-01T10:00:00Z
## 6      0   1012.4    10 2013-01-01T11:00:00Z
```

```
#Ques 3  Print variables names/types, select wind_speed and 1 other variable for the next tasks.

str(data)
```

```
## 'data.frame':    26115 obs. of  15 variables:
##  $ origin    : chr  "EWR" "EWR" "EWR" "EWR" ...
##  $ year      : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ hour      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ temp      : num  39 39 39 39.9 39 ...
##  $ dewp      : num  26.1 27 28 28 28 ...
##  $ humid     : num  59.4 61.6 64.4 62.2 64.4 ...
##  $ wind_dir  : int  270 250 240 250 260 240 240 250 260 260 ...
##  $ wind_speed: num  10.36 8.06 11.51 12.66 12.66 ...
##  $ wind_gust : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ precip    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pressure  : num  1012 1012 1012 1012 1012 ...
##  $ visib     : num  10 10 10 10 10 10 10 10 10 10 ...
##  $ time_hour : chr  "2013-01-01T06:00:00Z" "2013-01-01T07:00:00Z" "2013-01-01T08:00:00Z" "2013-01-01
```

```
selected_data <- data[c("temp", "wind_speed")]

head(selected_data)
```

```
##    temp wind_speed
## 1 39.02   10.35702
## 2 39.02    8.05546
## 3 39.02   11.50780
## 4 39.92   12.65858
## 5 39.02   12.65858
## 6 37.94   11.50780
```

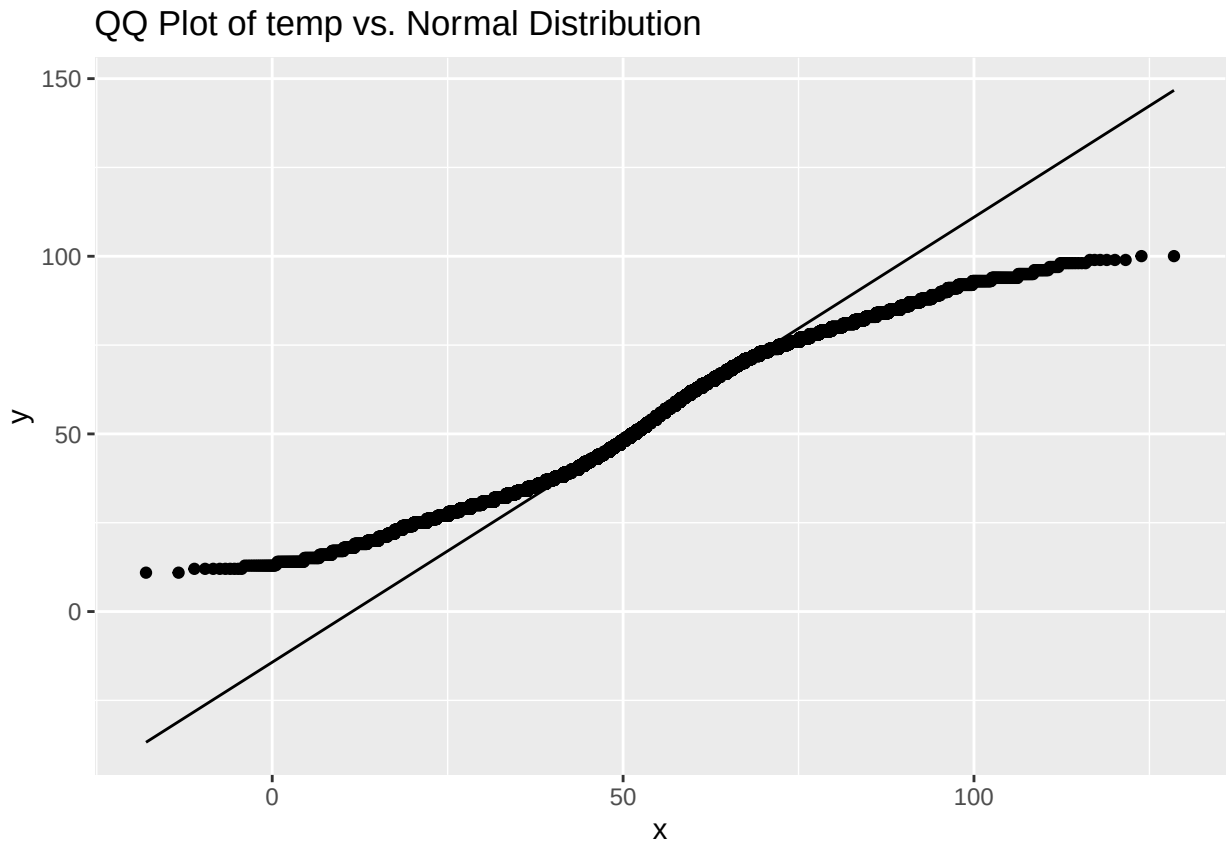
```
# Load necessary libraries
library(ggplot2)
library(fitdistrplus)
```

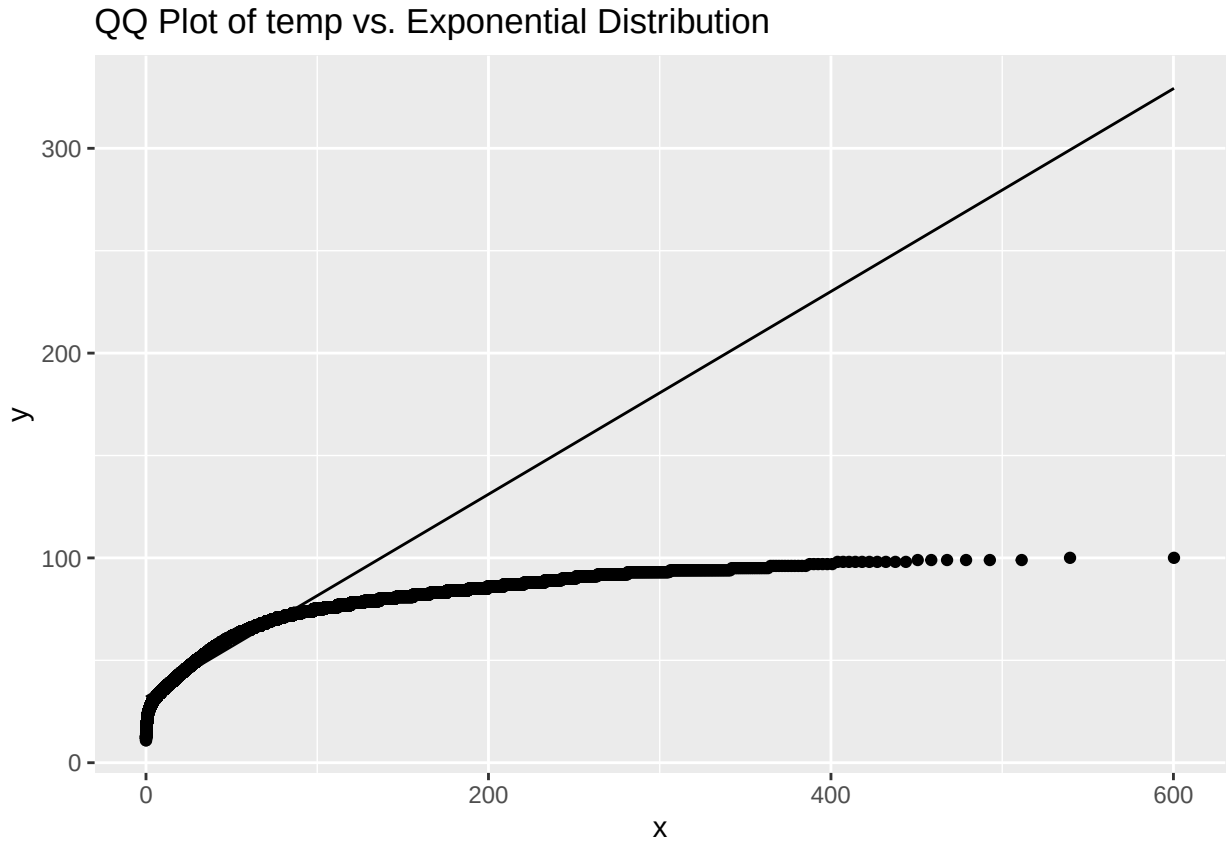
```
## Loading required package: MASS
```

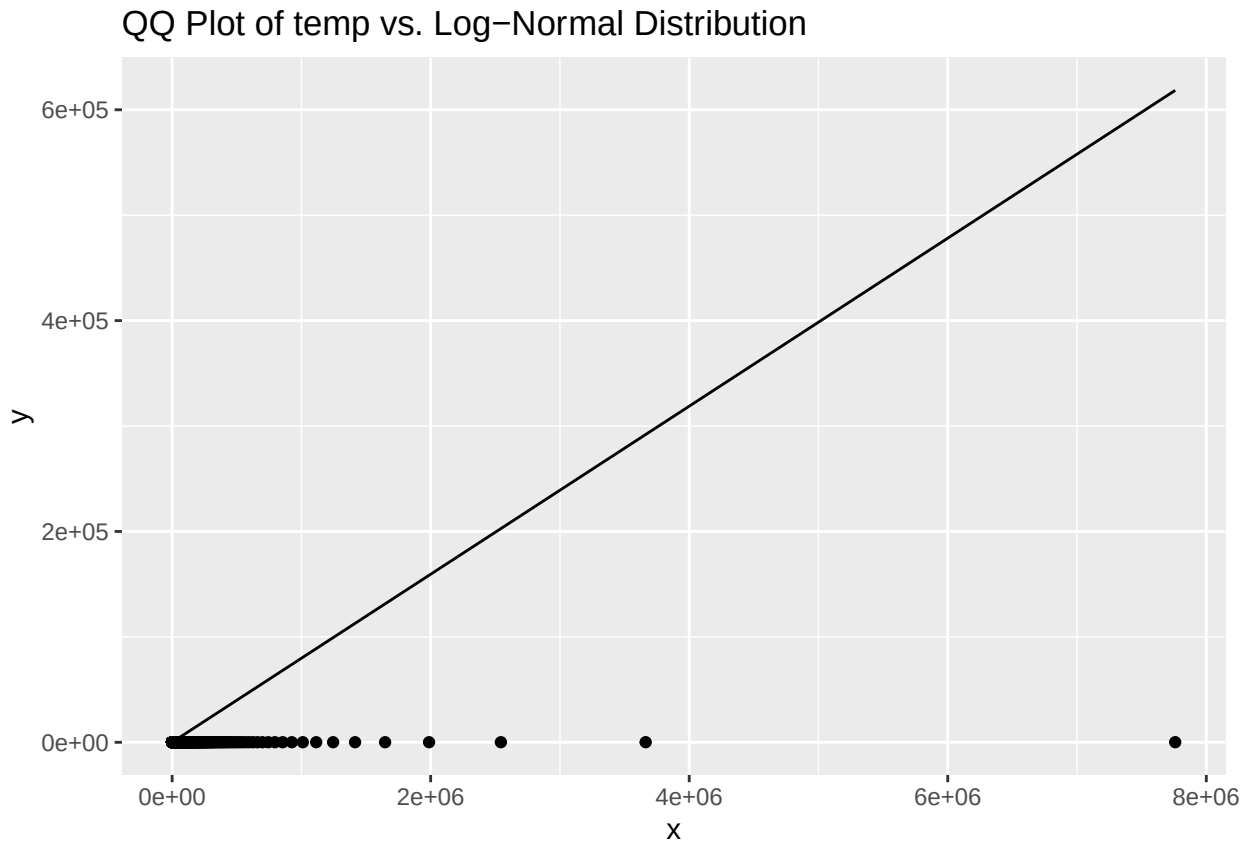
```
## Loading required package: survival
```

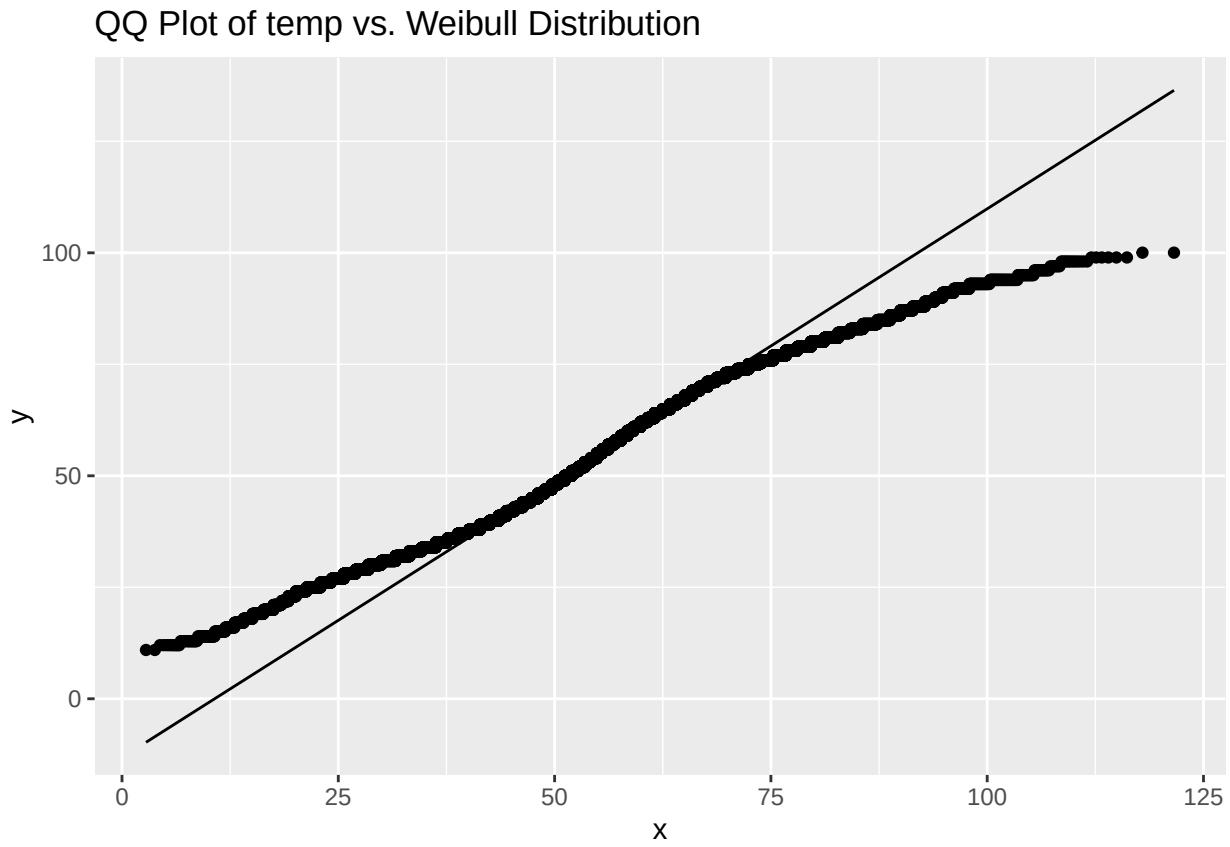
```
#Ques 4 Provide QQ plots and histograms for the selected 2 variables

generate_qq_plots <- function(data, variable_name) {
  # Extract the specified variable data
  variable_data <- data[[variable_name]]
  # Remove NA values
  variable_data <- variable_data[!is.na(variable_data)]
  # Create a data frame with the variable data
  df <- data.frame(variable_data = variable_data)
  # Calculate parameters for the distributions
  mean_var <- mean(variable_data, na.rm = TRUE)
  sd_var <- sd(variable_data, na.rm = TRUE)

  # QQ Plot against a Normal Distribution
  qqplot_normal <- ggplot(df, aes(sample = variable_data)) +
    stat_qq(distribution = qnorm, dparams = list(mean = mean_var, sd = sd_var)) +
    stat_qq_line(distribution = qnorm, dparams = list(mean = mean_var, sd = sd_var)) +
    ggtitle(paste("QQ Plot of", variable_name, "vs. Normal Distribution"))
  print(qqplot_normal)
```

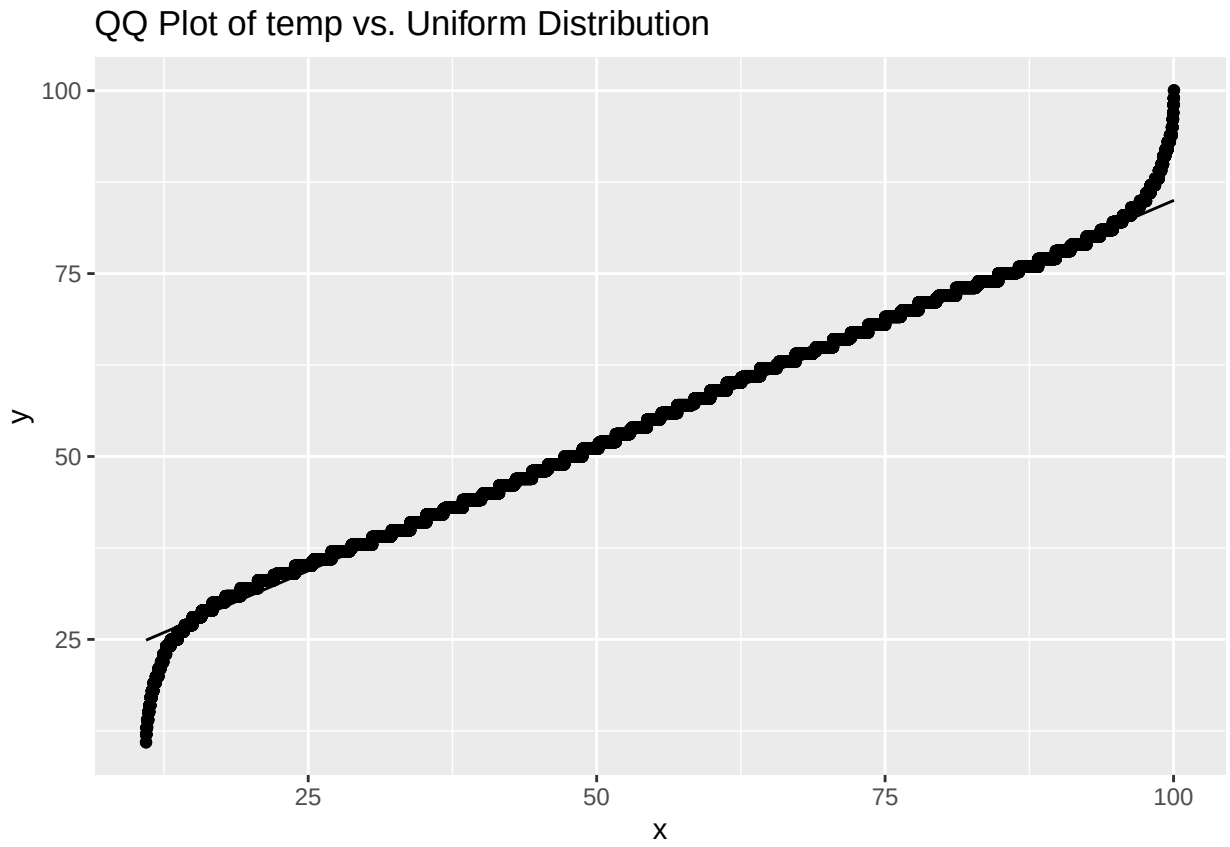
```r
  # QQ Plot against an Exponential Distribution
  rate_exp <- 1 / mean_var
  qqplot_exp <- ggplot(df, aes(sample = variable_data)) +
    stat_qq(distribution = qexp, dparams = list(rate = rate_exp)) +
    stat_qq_line(distribution = qexp, dparams = list(rate = rate_exp)) +
    ggtitle(paste("QQ Plot of", variable_name, "vs. Exponential Distribution"))
  print(qqplot_exp)

  # QQ Plot against a Log-Normal Distribution
  qqplot_lognorm <- ggplot(df, aes(sample = variable_data)) +
    stat_qq(distribution = qlnorm, dparams = list(meanlog = log(mean_var), sdlog = log(sd_var))) +
    stat_qq_line(distribution = qlnorm, dparams = list(meanlog = log(mean_var), sdlog = log(sd_var))) +
    ggtitle(paste("QQ Plot of", variable_name, "vs. Log-Normal Distribution"))
  print(qqplot_lognorm)

  # Estimate shape and scale parameters for Weibull distribution
  fit_weibull <- tryCatch({
    fitdistrplus::fitdist(variable_data[variable_data > 0], "weibull")
  }, error = function(e) {
    warning("Weibull fitting failed: ", e$message)
    return(NULL)
  })

  if (!is.null(fit_weibull)) {
    shape <- fit_weibull$estimate["shape"]
    scale <- fit_weibull$estimate["scale"]
    # QQ Plot against a Weibull Distribution
    qqplot_weibull <- ggplot(df, aes(sample = variable_data)) +
      stat_qq(distribution = qweibull, dparams = list(shape = shape, scale = scale)) +
      stat_qq_line(distribution = qweibull, dparams = list(shape = shape, scale = scale)) +
      ggtitle(paste("QQ Plot of", variable_name, "vs. Weibull Distribution"))
    print(qqplot_weibull)
  } else {
    print("Weibull distribution fitting could not be performed due to data issues.")
  }

  # QQ Plot against a Uniform Distribution
  qqplot_uniform <- ggplot(df, aes(sample = variable_data)) +
    stat_qq(distribution = qunif, dparams = list(min = min(variable_data, na.rm = TRUE), max = max(varia
    stat_qq_line(distribution = qunif, dparams = list(min = min(variable_data, na.rm = TRUE), max = max
    ggtitle(paste("QQ Plot of", variable_name, "vs. Uniform Distribution"))
  print(qqplot_uniform)
}

generate_qq_plots(data,"temp")
```


QQ Plot of temp vs. Normal Distribution
150
100
50
0
y
0
50
100
x

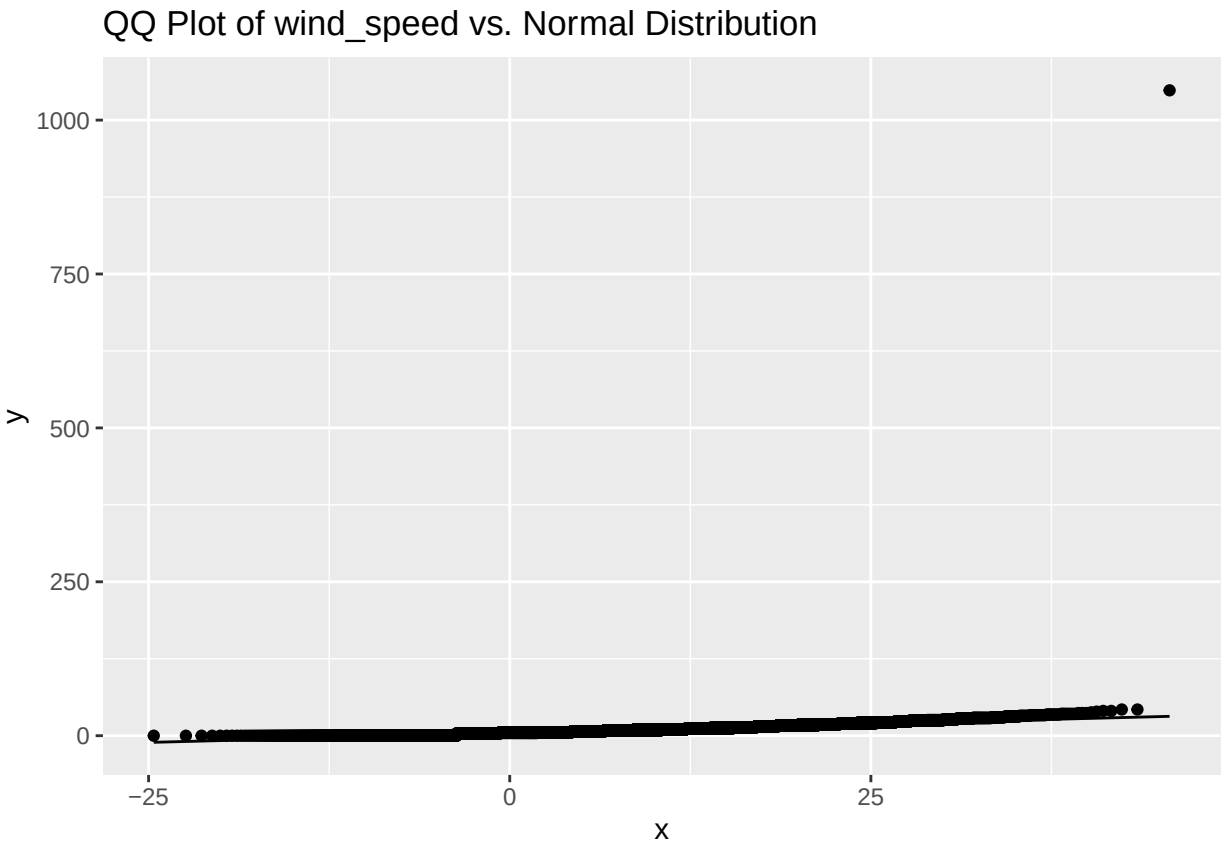

QQ Plot of temp vs. Exponential Distribution
300
200
100
0
y
0
200
400
600
x

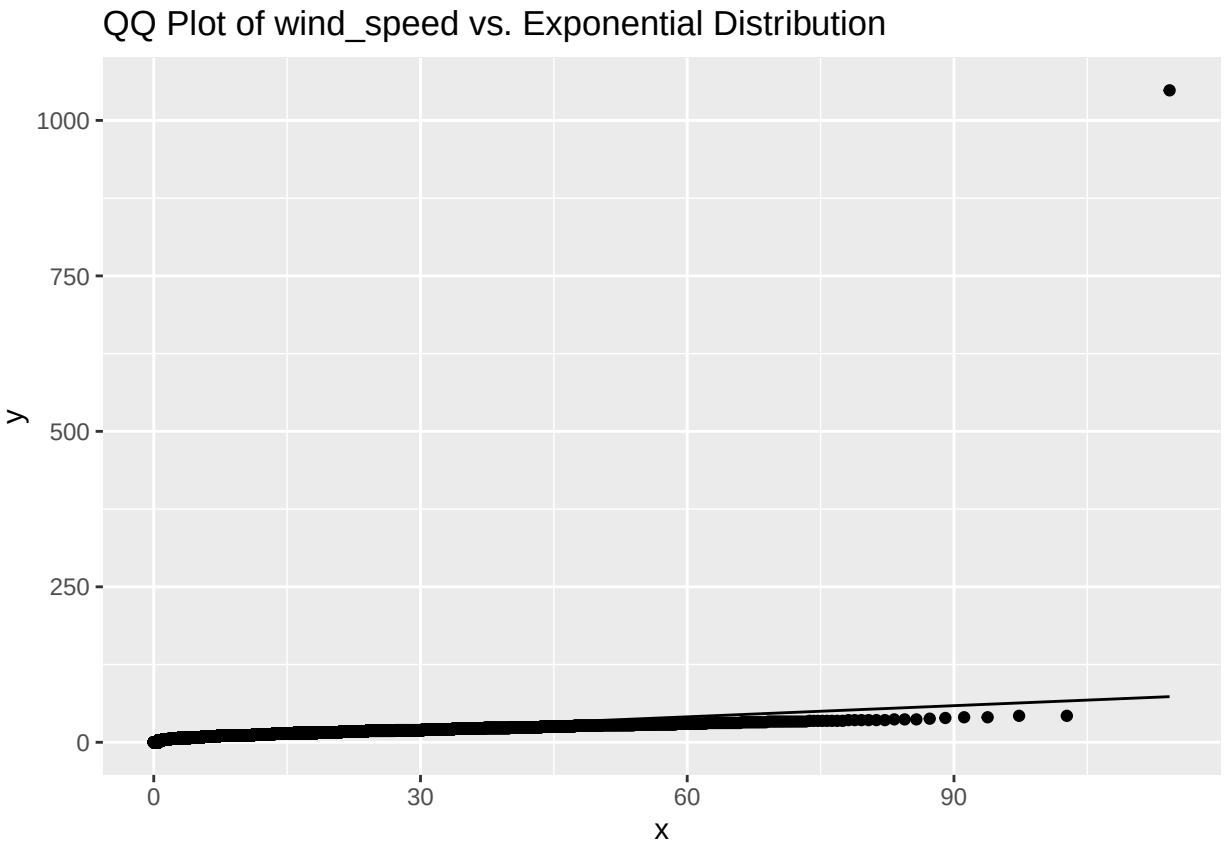

QQ Plot of temp vs. Log−Normal Distribution
6e+05
4e+05
2e+05
0e+00
y
0e+00
2e+06
4e+06
6e+06
8e+06
x

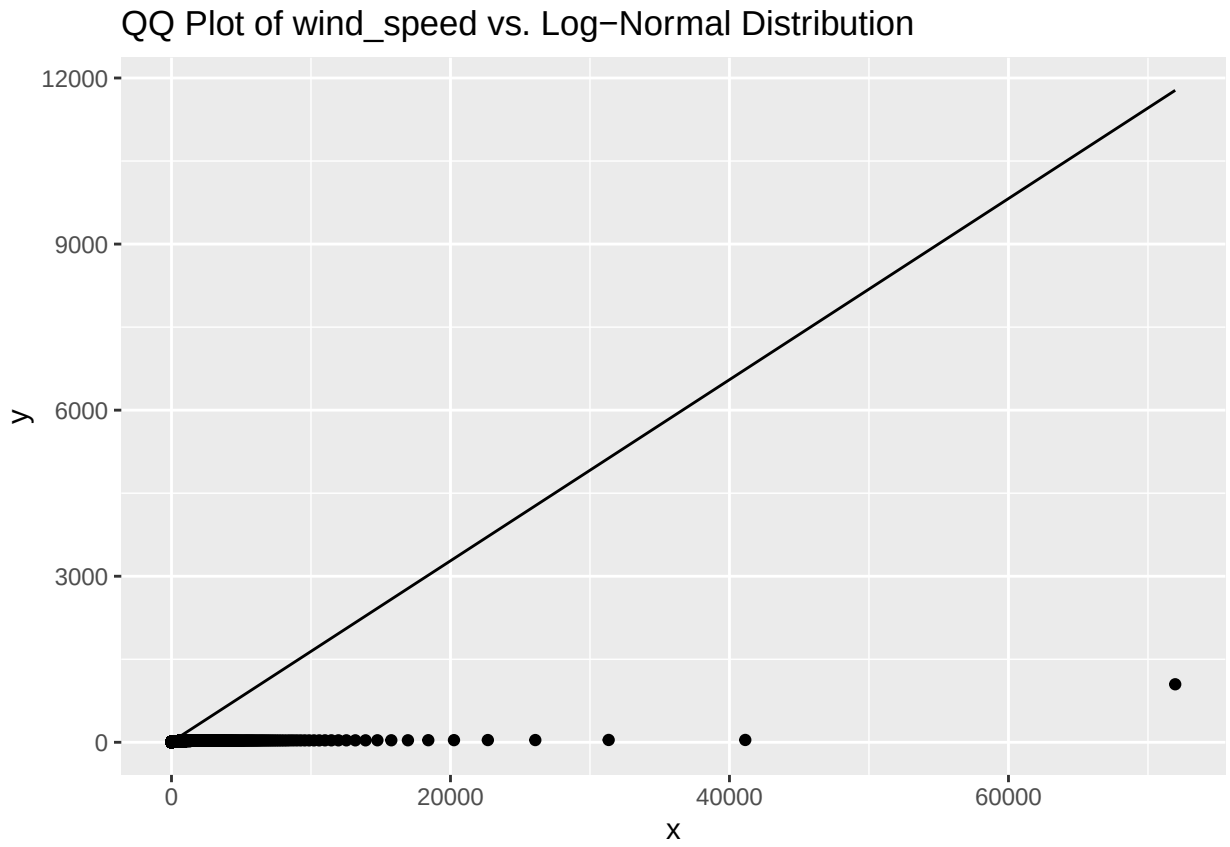

QQ Plot of temp vs. Weibull Distribution
y
x
0
50
100
0
25
50
75
100
125

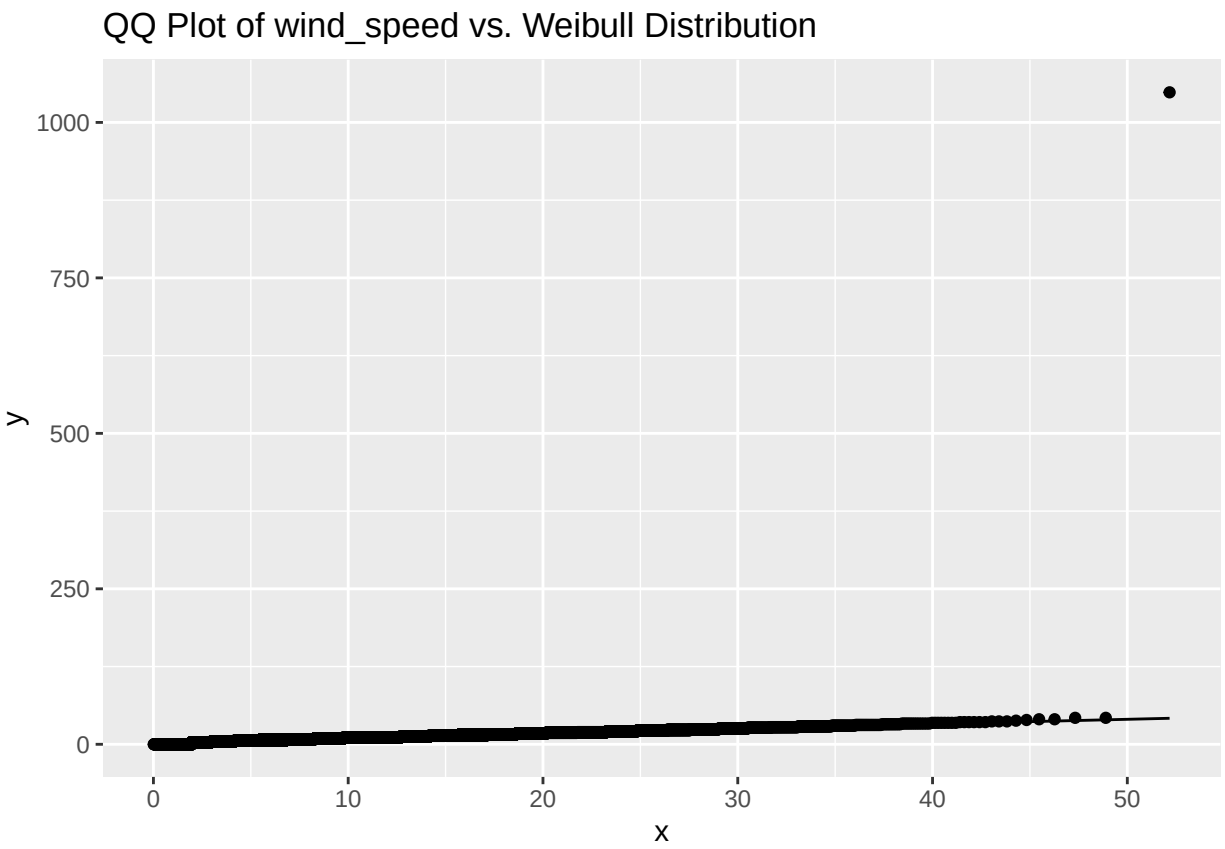


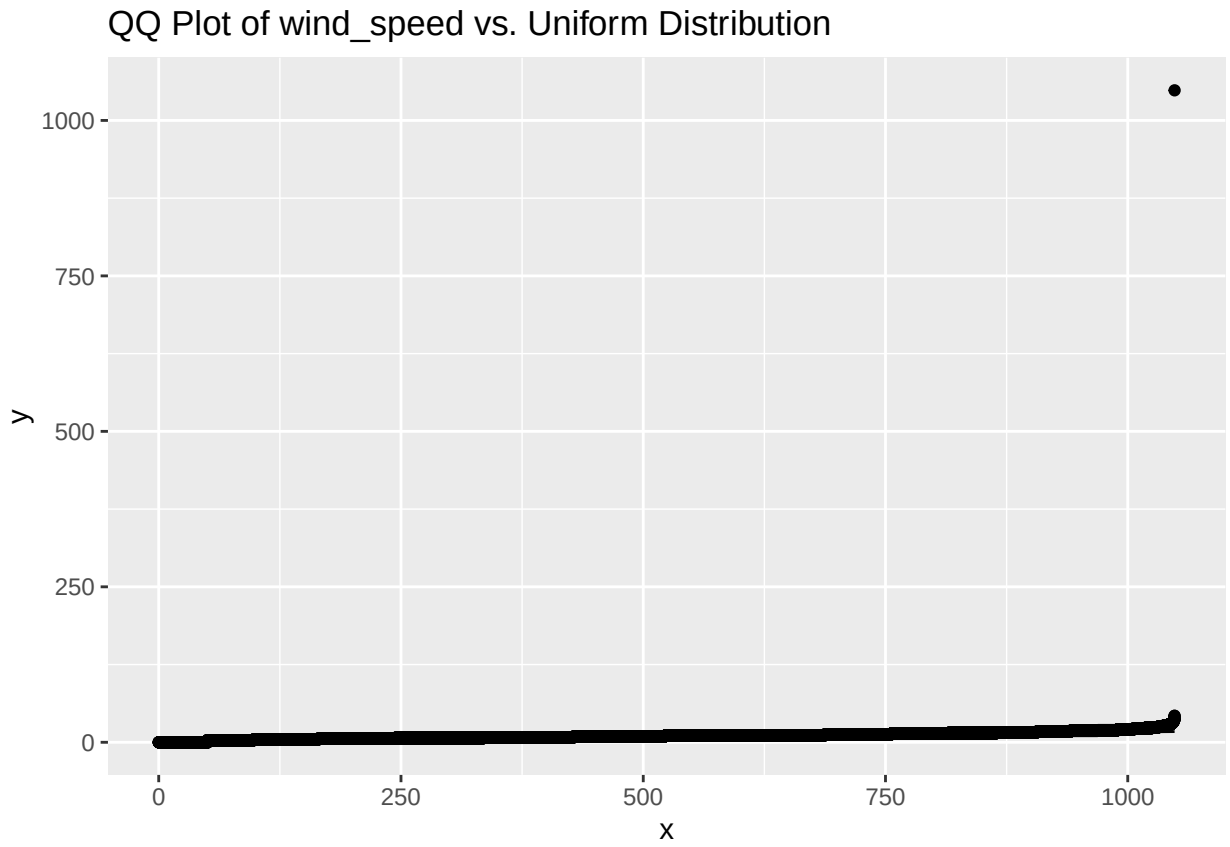

```
#Generate the QQ plot without removing the outliers
generate_qq_plots(data,"wind_speed")
```


QQ Plot of wind_speed vs. Normal Distribution
1000
750
500
250
0
y
−25
0
25
x

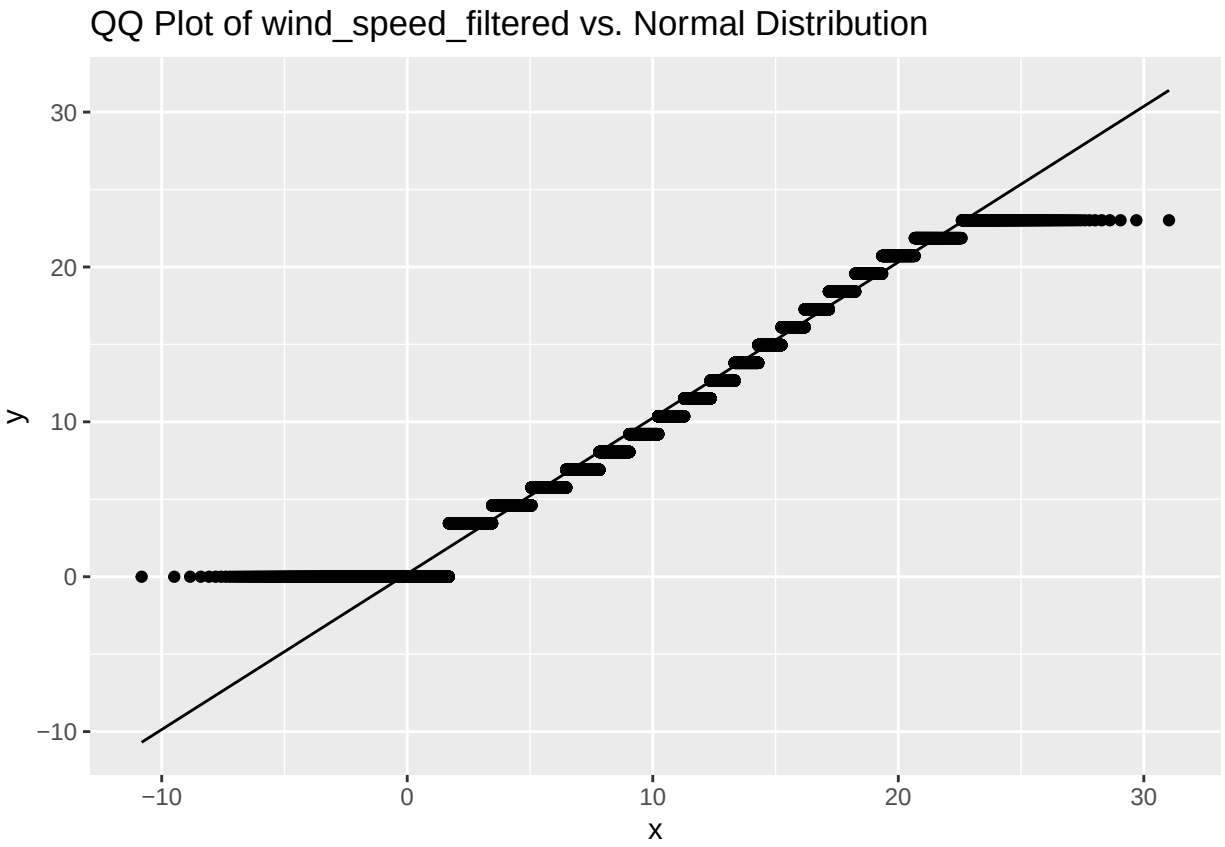

QQ Plot of wind_speed vs. Exponential Distribution
1000
750
500
250
0
y
0
30
60
90
x

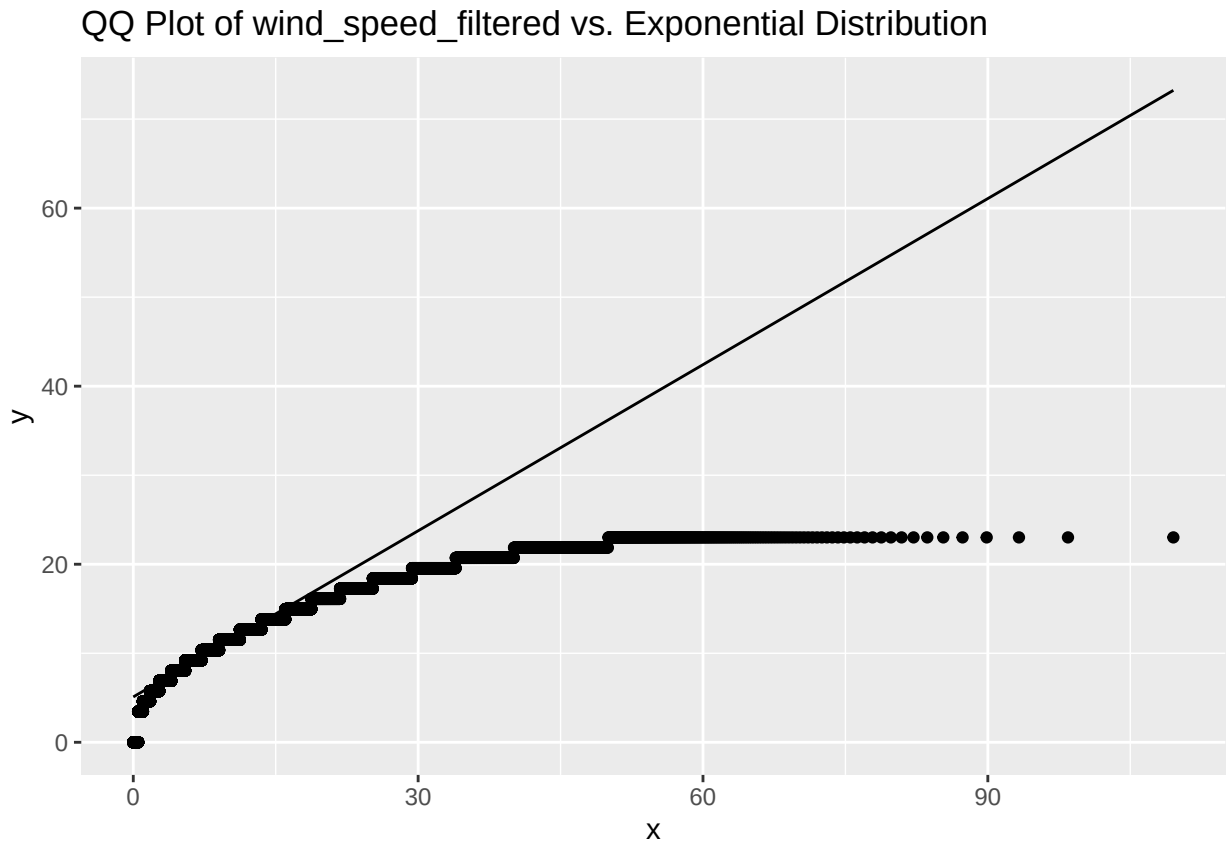

QQ Plot of wind_speed vs. Log−Normal Distribution
12000
9000
6000
3000
0
y
0
20000
40000
60000
x

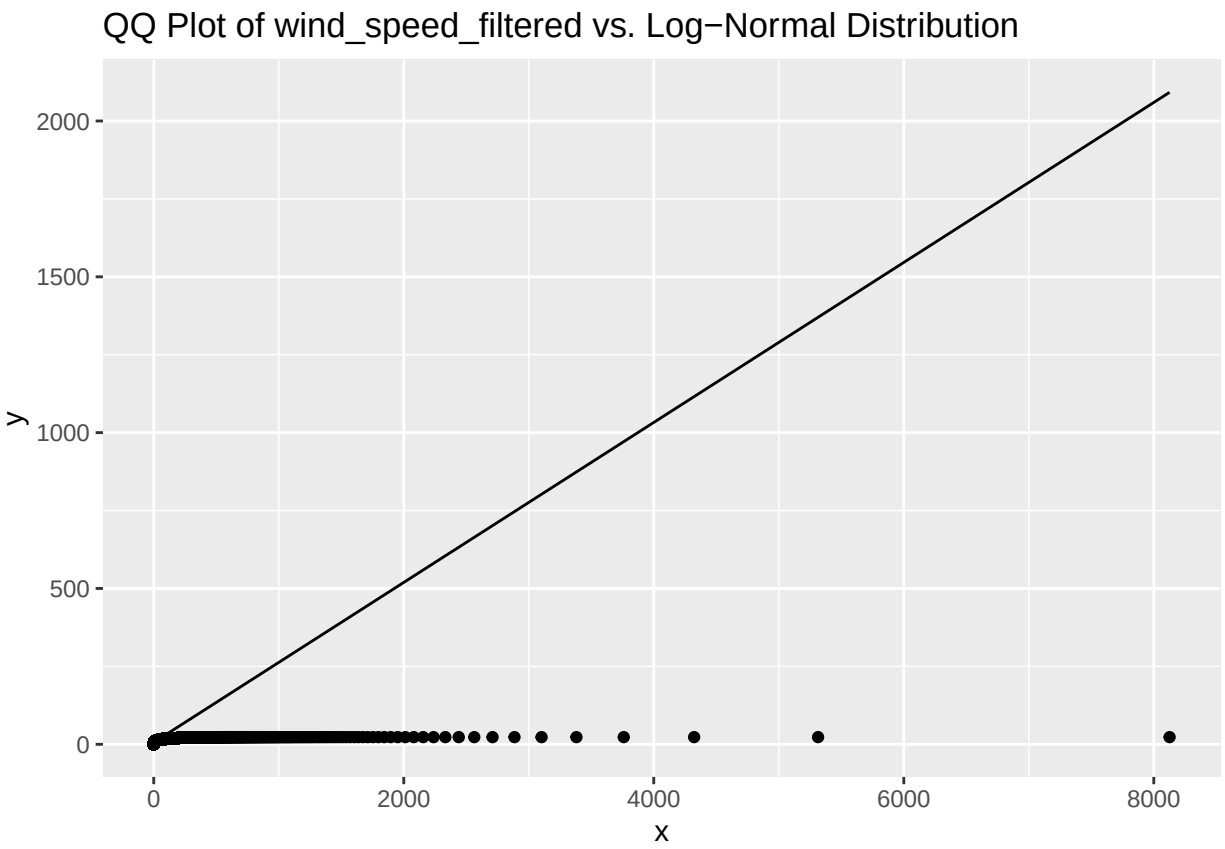

QQ Plot of wind_speed vs. Weibull Distribution
1000
750
500
250
0
y
0
10
20
30
40
50
x

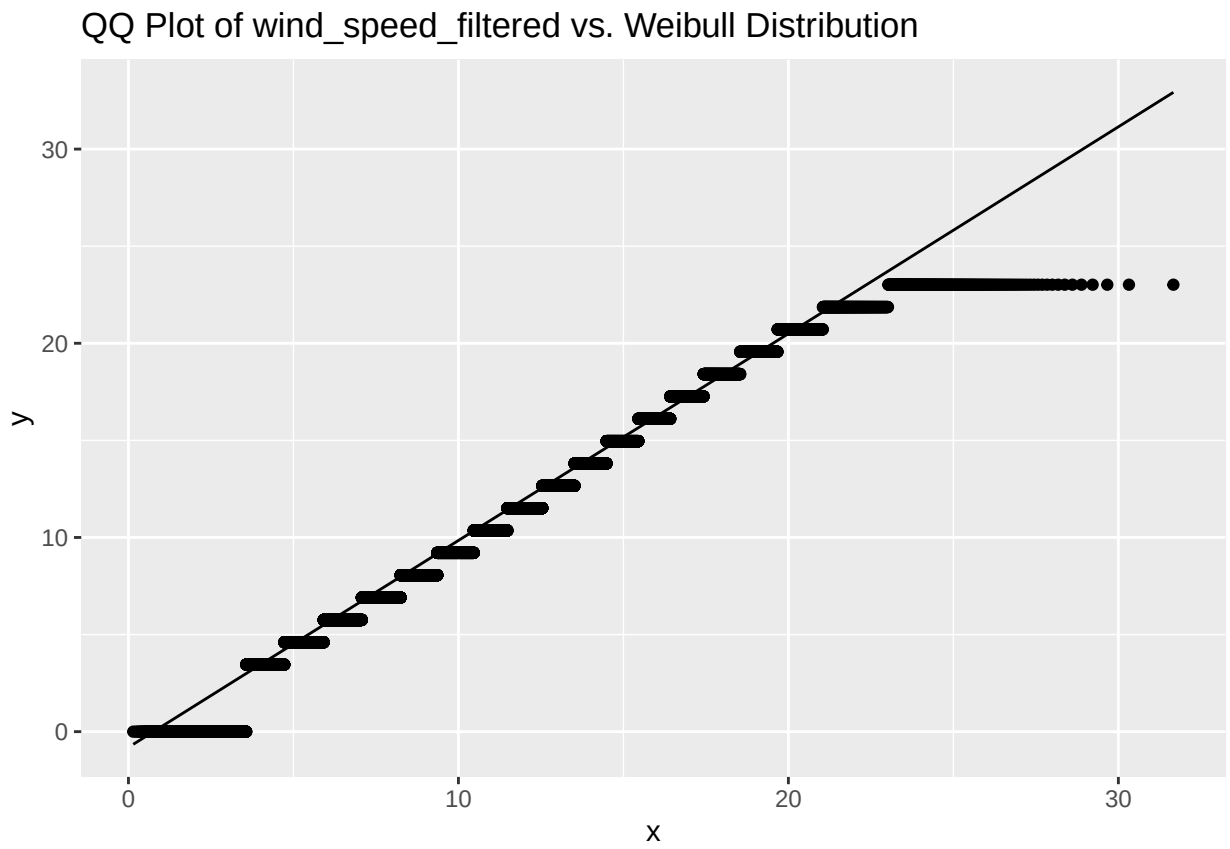


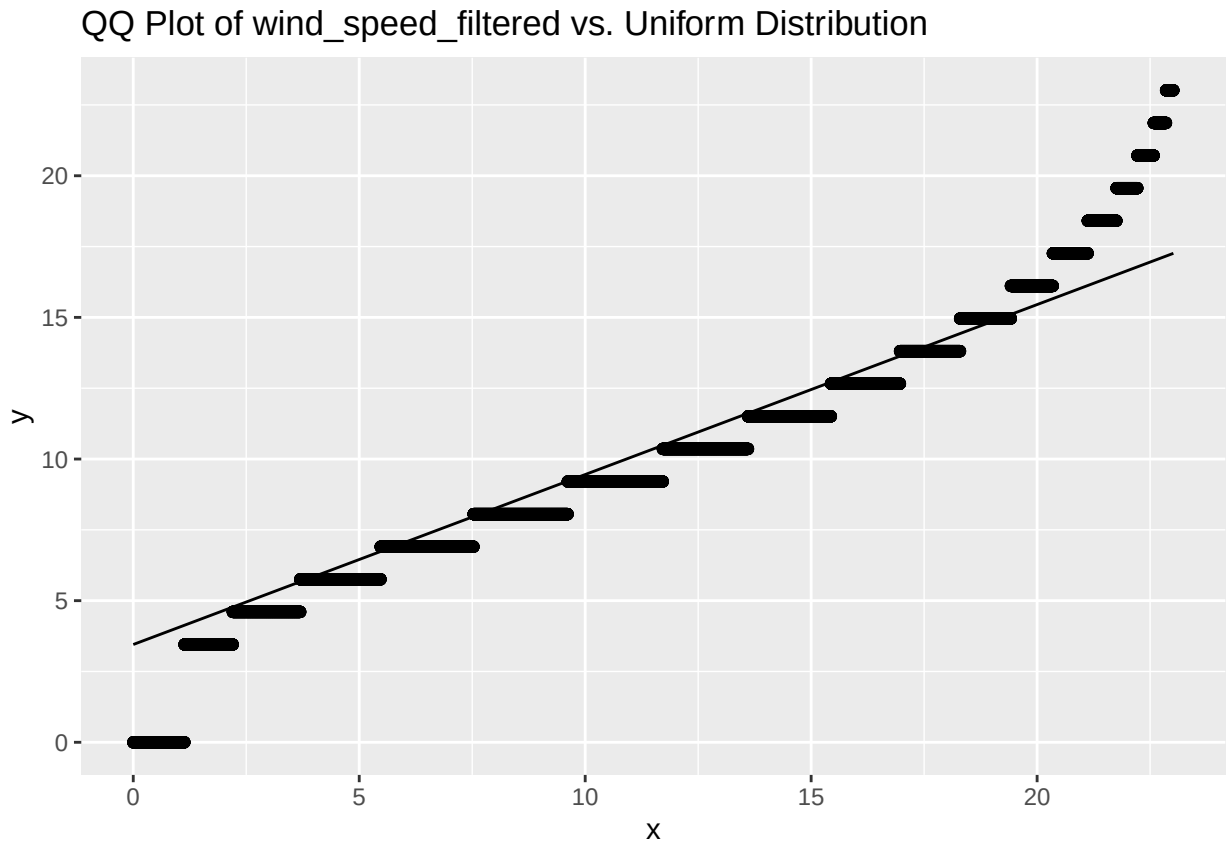

```r
# We now remove the outliers and plot the QQ plot
wind_speed <- data$wind_speed
wind_speed <- wind_speed[!is.na(wind_speed) & !is.nan(wind_speed)]
# Remove outliers using the IQR method
Q1 <- quantile(wind_speed, 0.25, na.rm = TRUE)
Q3 <- quantile(wind_speed, 0.75, na.rm = TRUE)
IQR <- Q3 - Q1
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR
wind_speed_filtered<- wind_speed[wind_speed >= lower_bound & wind_speed <= upper_bound]
# Create a data frame for wind_speed_filtered
wind_speed_filtered_df <- data.frame(wind_speed_filtered = wind_speed_filtered)
# Call the generate_qq_plots function
generate_qq_plots(wind_speed_filtered_df, "wind_speed_filtered")
```


QQ Plot of wind_speed_filtered vs. Normal Distribution
30
20
10
0
−10
y
−10
0
10
20
30
x

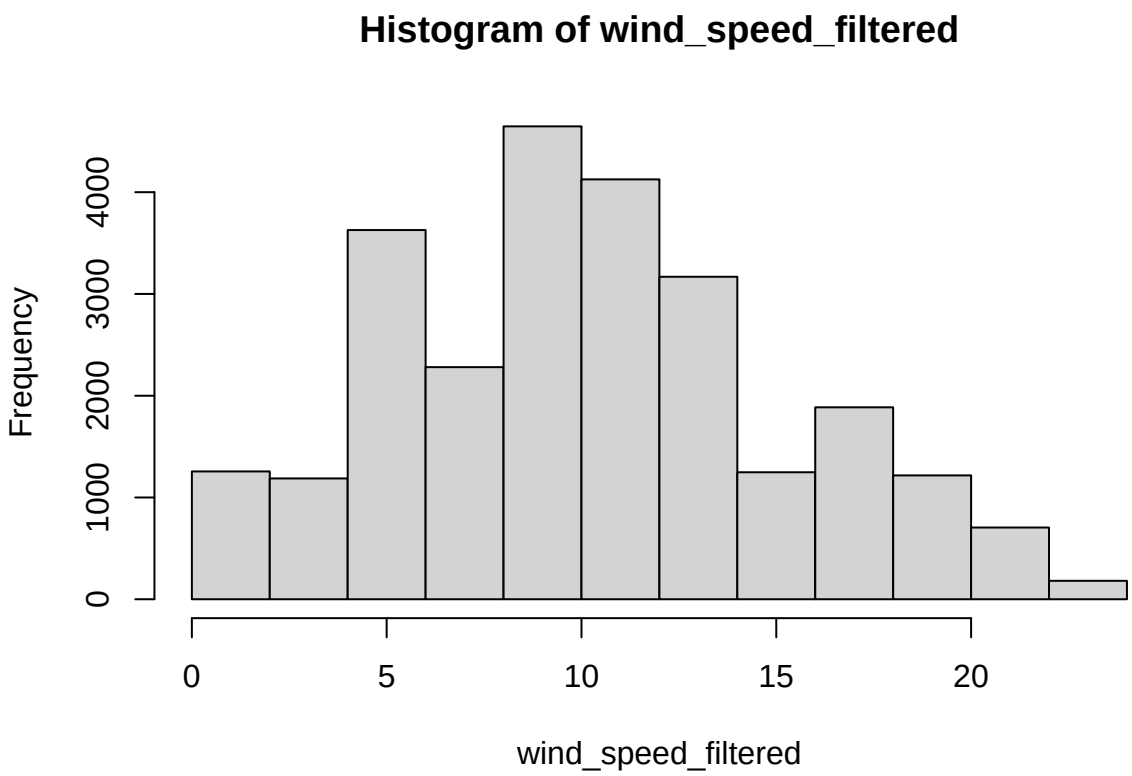

QQ Plot of wind_speed_filtered vs. Exponential Distribution
60
40
20
0
y
0
30
60
90
x


QQ Plot of wind_speed_filtered vs. Log−Normal Distribution
2000
1500
1000
500
0
y
0
2000
4000
6000
8000
x


QQ Plot of wind_speed_filtered vs. Weibull Distribution
30
20
10
0
y
0
10
20
30
x

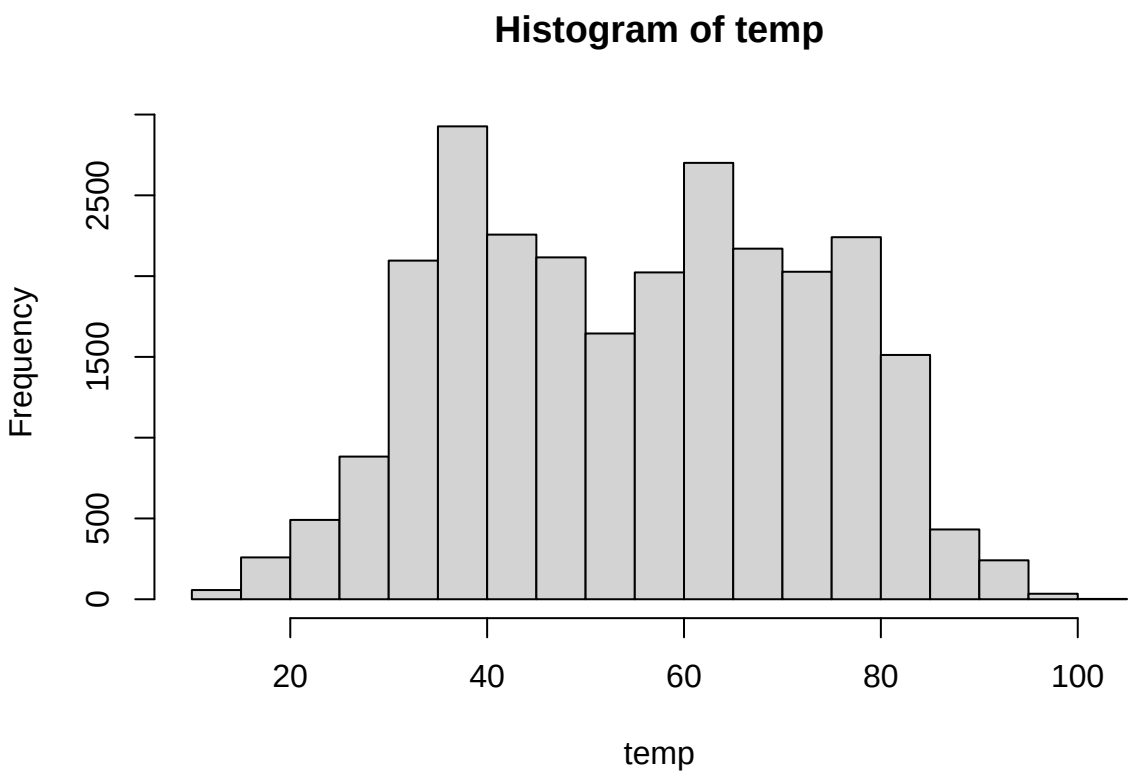



```
#We now plot the histograms
print(hist(wind_speed_filtered))
```



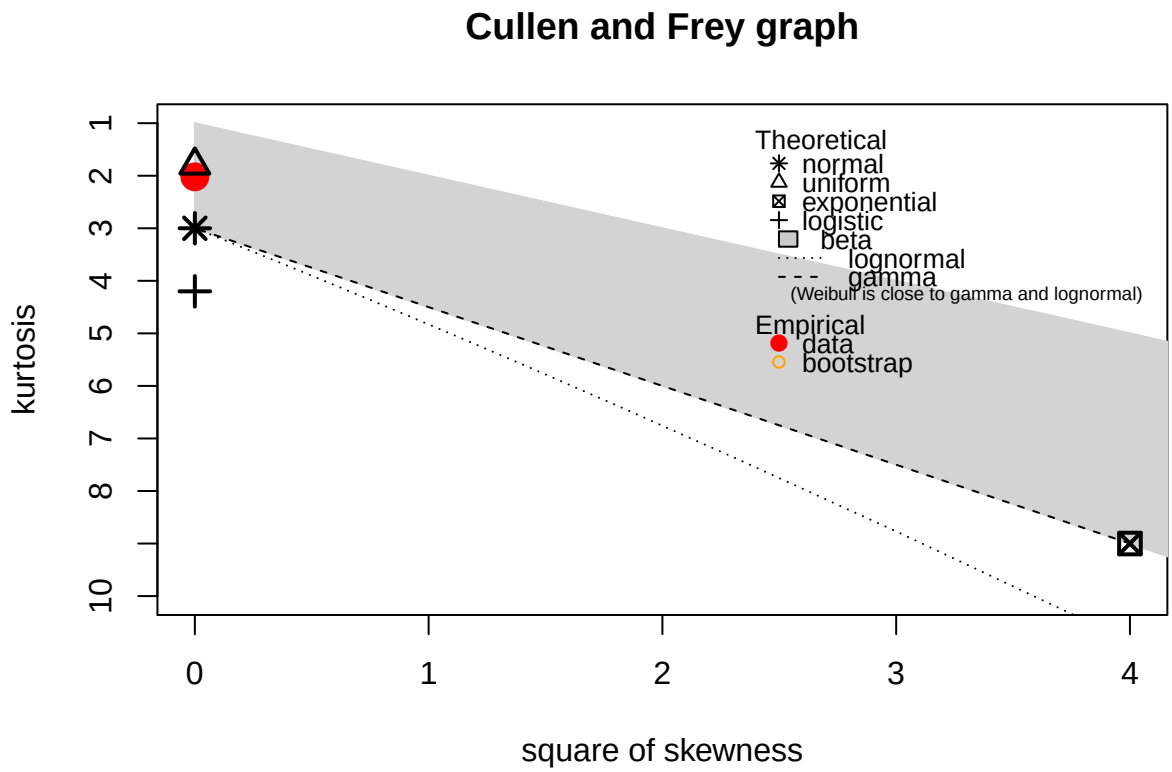

```
## $breaks
##  [1]  0  2  4  6  8 10 12 14 16 18 20 22 24
##
## $counts
##  [1] 1256 1187 3628 2281 4647 4126 3169 1248 1886 1217  705  181
##
## $density
##  [1] 0.02459755 0.02324625 0.07105088 0.04467118 0.09100701 0.08080373
##  [7] 0.06206181 0.02444088 0.03693549 0.02383377 0.01380674 0.00354471
##
## $mids
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23
##
## $xname
## [1] "wind_speed_filtered"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```r
#we not plot the temperature histogram.
temp <- data$temp
print(hist(temp))
```




```
## $breaks
##  [1]  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95 100
## [20] 105
##
## $counts
##  [1]   57  259  491  883 2096 2927 2257 2116 1645 2023 2701 2170 2027 2241 1512
## [16]  432  241   34    2
##
## $density
##  [1] 4.365474e-04 1.983610e-03 3.760435e-03 6.762656e-03 1.605269e-02
##  [6] 2.241709e-02 1.728575e-02 1.620587e-02 1.259861e-02 1.549360e-02
## [11] 2.068622e-02 1.661944e-02 1.552424e-02 1.716321e-02 1.158000e-02
## [16] 3.308570e-03 1.845753e-03 2.603967e-04 1.531745e-05
##
## $mids
##  [1]  12.5  17.5  22.5  27.5  32.5  37.5  42.5  47.5  52.5  57.5  62.5  67.5
## [13]  72.5  77.5  82.5  87.5  92.5  97.5 102.5
##
## $xname
## [1] "temp"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```r
#Ques 5 For one of the variables, generate simulated data using an appropriate distribution and
#compare the Cullen and Frey graph with that for the original distribution. Comment on
#any notable differences.
#Note: You will need to determine the required parameters appropriate for your
#distribution (and how to estimate them from the original data). A list of these can be
#found by searching for 'distributions' in the R Studio help tab. Remember that, e.g.,
#rgamma(...), runif(...) will be the corresponding function that generates the random
#data.

library(fitdistrplus)

# We extract temperature data and remove NAs
temp <- data$temp
temp <- temp[!is.na(temp)]

# Step 1: We will analyze the Original Data
# Fit a Normal distribution to the original temperature data
fit_temp <- fitdist(temp, "norm")

# We will put Cullen and Frey graph for the original data
descdist(temp, boot = 1000)
```



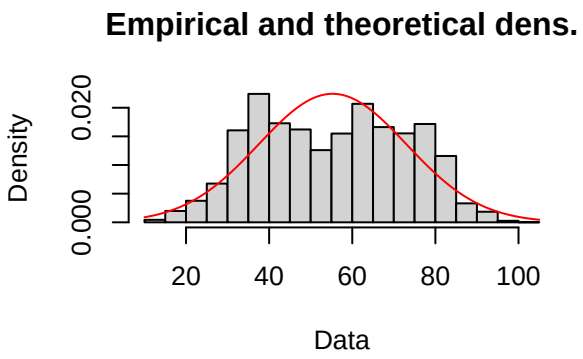

```
## summary statistics
## ------
## min:  10.94   max:  100.04
```

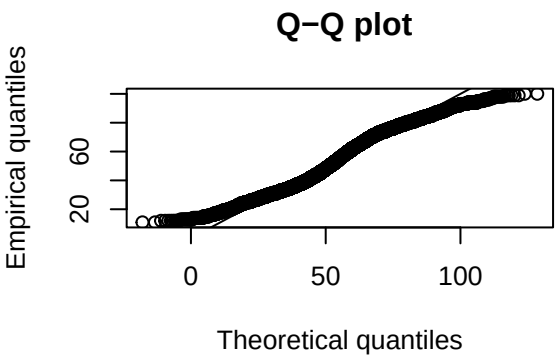
```
## median:  55.4
## mean:  55.26039
## estimated sd:  17.78785
## estimated skewness:  -0.006526353
## estimated kurtosis:  2.021424
```

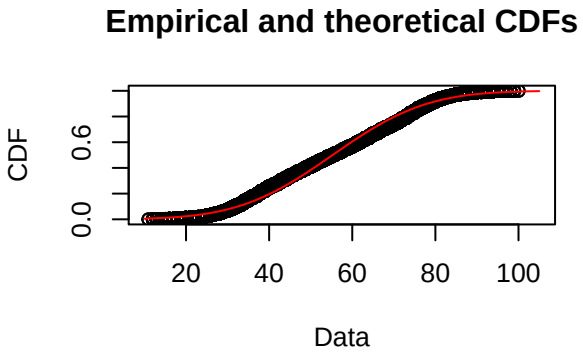
```r
# Step 2: We will simulate Data Based on the Fitted Distribution
# Estimate parameters from the fitted Normal distribution
mean_temp <- fit_temp$estimate["mean"]
sd_temp <- fit_temp$estimate["sd"]

# We will simulate data using the Normal distribution
simulated_temp <- rnorm(length(temp), mean = mean_temp, sd = sd_temp)

# We will simulate data using the Normal distribution
simulated_temp <- rnorm(length(temp), mean = mean_temp, sd = sd_temp)

#Ques 6 For each variable, use the fitdist() function in the "fitdistrplus" package to obtain an
#appropriate MLE fit of your data and display the estimates as well as the empirical vs
#theoretical comparison plots.

library(fitdistrplus)
library(ggplot2)

# Here the function to fit distributions, to find the best fit, and create a plot of the best fit
find_best_distribution <- function(data, variable_name) {
  # Extract the data for the specified variable, removing NA/NaN values
  variable_data <- data[[variable_name]]
  variable_data <- variable_data[!is.na(variable_data) & !is.nan(variable_data)]

  # Add a constant to ensure all values are positive for distributions that require it
  constant_add <- 100
  variable_data_positive <- variable_data + constant_add
  # Fit different distributions to the data
  fit_weibull <- fitdist(variable_data_positive, "weibull")
  fit_normal <- fitdist(variable_data, "norm")
  fit_lognormal <- fitdist(variable_data_positive, "lnorm")
  fit_exponential <- fitdist(variable_data_positive, "exp")
  fit_uniform <- fitdist(variable_data, "unif")

  # Extract negative log-likelihoods
  nll_weibull <- -fit_weibull$loglik
  nll_normal <- -fit_normal$loglik
  nll_lognormal <- -fit_lognormal$loglik
  nll_exponential <- -fit_exponential$loglik
  nll_uniform <- -fit_uniform$loglik
  # Create a data frame to summarize and compare
  nll_summary <- data.frame(
    Distribution = c("Weibull", "Normal", "Lognormal", "Exponential", "Uniform"),
    NLL = c(nll_weibull, nll_normal, nll_lognormal, nll_exponential, nll_uniform)
  )
  # Identify the distribution with the minimum NLL (best fit)
  best_fit <- nll_summary[which.min(nll_summary$NLL), ]
  # Print the summary and the best fit
```

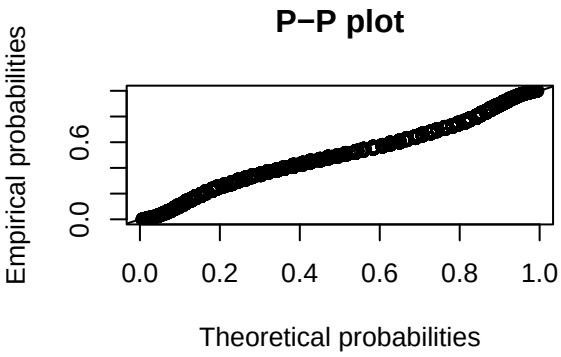
```r
  print(nll_summary)
  print(paste("Best fit distribution:", best_fit$Distribution))
  # Plot the best-fitting distribution
  if (best_fit$Distribution == "Weibull") {
    plot(fit_weibull)
  } else if (best_fit$Distribution == "Normal") {
    plot(fit_normal)
  } else if (best_fit$Distribution == "Lognormal") {
    plot(fit_lognormal)
  } else if (best_fit$Distribution == "Exponential") {
    plot(fit_exponential)
  } else if (best_fit$Distribution == "Uniform") {
    plot(fit_uniform)
  }
  return(list(
    nll_summary = nll_summary,
    best_fit_distribution = best_fit$Distribution
  ))
}

find_best_distribution(data, "temp")
```

```
##   Distribution      NLL
## 1      Weibull 112600.9
## 2       Normal 112223.2
## 3    Lognormal 112340.2
## 4  Exponential 157861.8
## 5      Uniform 117245.6
## [1] "Best fit distribution: Normal"
```




```
## $nll_summary
##   Distribution      NLL
## 1      Weibull 112600.9
## 2       Normal 112223.2
## 3    Lognormal 112340.2
## 4  Exponential 157861.8
## 5      Uniform 117245.6
##
## $best_fit_distribution
## [1] "Normal"
```