STATS 506 Final Project

Lingzhi Hao

GitHub Repository: https://github.com/Lingzhi-Hao/STATS-506-Final-Project

```
set.seed(42)

library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(e1071)
```

```
Warning: package 'e1071' was built under R version 4.5.2
```

```
library(nortest)
```

```
Warning: package 'nortest' was built under R version 4.5.2
```

```
library(ggplot2)
```

```
Attaching package: 'ggplot2'
```

```
The following object is masked from 'package:e1071':

    element
```

```r
library(patchwork)
```

Set number of repetitions for Monte Carlo simulations, general sample size grid, and threshold of convergence.

```r
reps <- 5000
n_values <- c(5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 100)

p_threshold <- 0.05
skew_threshold <- 0.1
kurt_threshold <- 0.2
```

1. **Uniform distribution**

   Conduct Monte Carlo simulations to generate Z, and calculate the A2, p, Abs_Skew, and Abs_Kurtosis.

   ```r
   a_uni <- 0
   b_uni <- 1

   mu_uni <- (a_uni + b_uni) / 2
   sigma_uni <- sqrt((b_uni - a_uni)^2 / 12)

   results_uniform <- data.frame(
     n = integer(),
     A2 = numeric(),          # Anderson-Darling statistic
     p = numeric(),
     Abs_Skew = numeric(),   # Absolute Skewness
     Abs_Kurtosis = numeric() # Absolute Excess Kurtosis
   )

   for (n in n_values) {
     sample_means <- numeric(reps)

     for (i in 1:reps) {
       samples <- runif(n, min = a_uni, max = b_uni)
   ```

```r
    sample_means[i] <- mean(samples)
  }

  # Calculate the standardized Z-statistic
  Z <- (sample_means - mu_uni) / (sigma_uni / sqrt(n))


  # Anderson-Darling Statistic
  Z_sub <- sample(Z, 500)
  ad <- ad.test(Z_sub)
  ad_stat <- ad$statistic
  p_value <- ad$p.value

  # Absolute Skewness
  skewness_val <- abs(e1071::skewness(Z))

  # Absolute Excess Kurtosis
  kurtosis_val <- abs(e1071::kurtosis(Z))

  # Store results for the current n
  results_uniform <- rbind(results_uniform, data.frame(
    n = n,
    A2 = ad_stat,
    p = p_value,
    Abs_Skew = skewness_val,
    Abs_Kurtosis = kurtosis_val
  ))
}

# Print the resulting table
print(results_uniform)
```

```
     n        A2          p    Abs_Skew Abs_Kurtosis
A    5 0.2719243 0.66995886 0.030469707   0.23185444
A1  10 0.7751119 0.04385430 0.042952465   0.17957260
A2  15 0.6223121 0.10464140 0.012113975   0.06818970
A3  20 0.3498371 0.47193461 0.041649046   0.03213142
A4  25 0.3527750 0.46470065 0.015642698   0.12843418
A5  30 0.3717617 0.42030696 0.034574717   0.04938775
A6  35 0.7380144 0.05415956 0.061490205   0.03888959
A7  40 0.3992937 0.36271373 0.015539385   0.07110972
A8  45 0.4226622 0.31953984 0.005418398   0.10803816
```

```
A9    50 0.2606493 0.70793826 0.006251157   0.02090600
A10   60 0.6104598 0.11194789 0.006767555   0.06786130
A11   70 0.5478184 0.15797037 0.010194028   0.05690354
A12   80 0.4188092 0.32632117 0.040065966   0.07098296
A13 100 0.4266866 0.31259380 0.014613957   0.11051374
```

Find the smallest sample size n*.

```
n_star_uniform <- results_uniform %>%
  filter(
    p >= 0.05,
    Abs_Skew <= skew_threshold,
    Abs_Kurtosis <= kurt_threshold
  ) %>%
  arrange(n) %>%
  slice(1) %>%
  pull(n)

n_star_uniform
```

```
[1] 15
```

Regenerate Z using sample size n*.

```
set.seed(42)

sample_means_star <- numeric(reps)

for (i in 1:reps) {
  samples <- runif(n_star_uniform, min = a_uni, max = b_uni)
  sample_means_star[i] <- mean(samples)
}

Z_star_uni <- (sample_means_star - mu_uni) / (sigma_uni / sqrt(n_star_uniform))
```

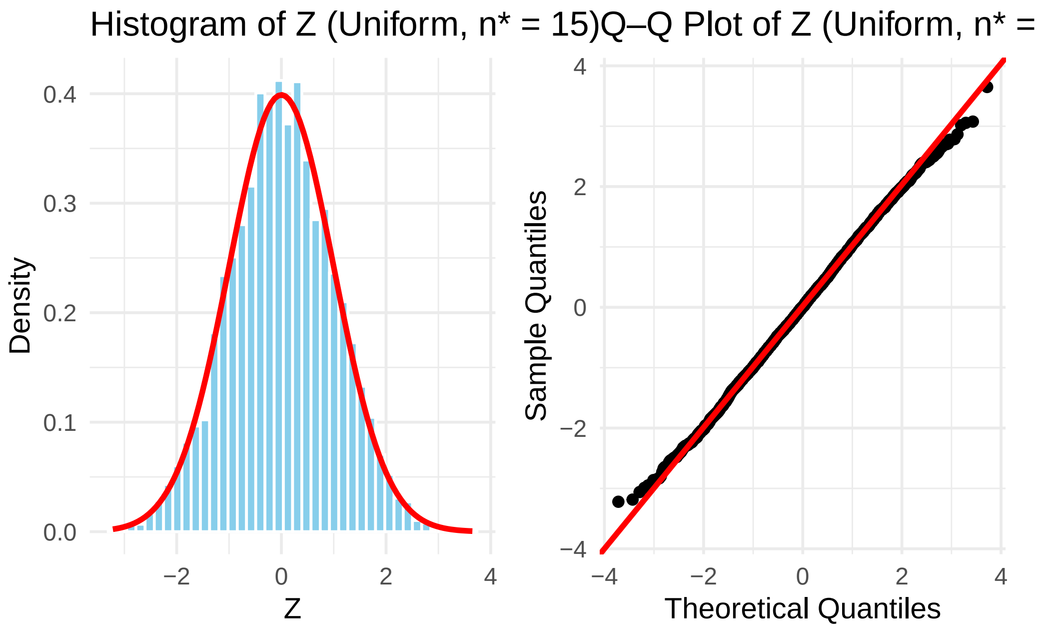
Define a function to plot the histogram and Q-Q plot of Z.

```
plot_hist_qq <- function(Z_star, dist_name = "", n_star = NULL, bins = 40) {
  df_plot <- data.frame(Z = Z_star)

  title_suffix <- ""
  if (dist_name != "") title_suffix <- dist_name
  if (!is.null(n_star)) {
    title_suffix <- paste0(title_suffix, ifelse(title_suffix == "", "", ", "),
                           "n* = ", n_star)
```

```r
  }
  if (title_suffix != "") title_suffix <- paste0(" (", title_suffix, ")")

  p_hist <- ggplot(df_plot, aes(x = Z)) +
    geom_histogram(aes(y = after_stat(density)),
                   bins = bins, fill = "skyblue", color = "white") +
    stat_function(fun = dnorm, color = "red", linewidth = 1) +
    labs(title = paste0("Histogram of Z", title_suffix),
         x = "Z", y = "Density") +
    theme_minimal()

  p_qq <- ggplot(df_plot, aes(sample = Z)) +
    stat_qq() +
    stat_qq_line(color = "red", linewidth = 1) +
    labs(title = paste0("Q-Q Plot of Z", title_suffix),
         x = "Theoretical Quantiles", y = "Sample Quantiles") +
    theme_minimal()

  p_hist + p_qq
}
```

Plot the histogram and Q-Q plot to see its closeness to normality.

```r
plot_hist_qq(Z_star_uni, dist_name = "Uniform", n_star = n_star_uniform)
```



Since the n* calculated above is very small, refine the sample size grid to find a more precise n*.

```
n_values_uni <- c(5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)

results_uniform_refine <- data.frame(
  n = integer(),
  A2 = numeric(),
  p = numeric(),
  Abs_Skew = numeric(),
  Abs_Kurtosis = numeric()
)

for (n in n_values_uni) {
  sample_means <- numeric(reps)

  for (i in 1:reps) {
    samples <- runif(n, min = a_uni, max = b_uni)
    sample_means[i] <- mean(samples)
  }

  Z <- (sample_means - mu_uni) / (sigma_uni / sqrt(n))

  Z_sub <- sample(Z, 500)
```

```r
  ad <- ad.test(Z_sub)

  results_uniform_refine <- rbind(results_uniform_refine,
    data.frame(
      n = n,
      A2 = ad$statistic,
      p = ad$p.value,
      Abs_Skew = abs(skewness(Z)),
      Abs_Kurtosis = abs(kurtosis(Z))
  ))
}

print(results_uniform_refine)
```

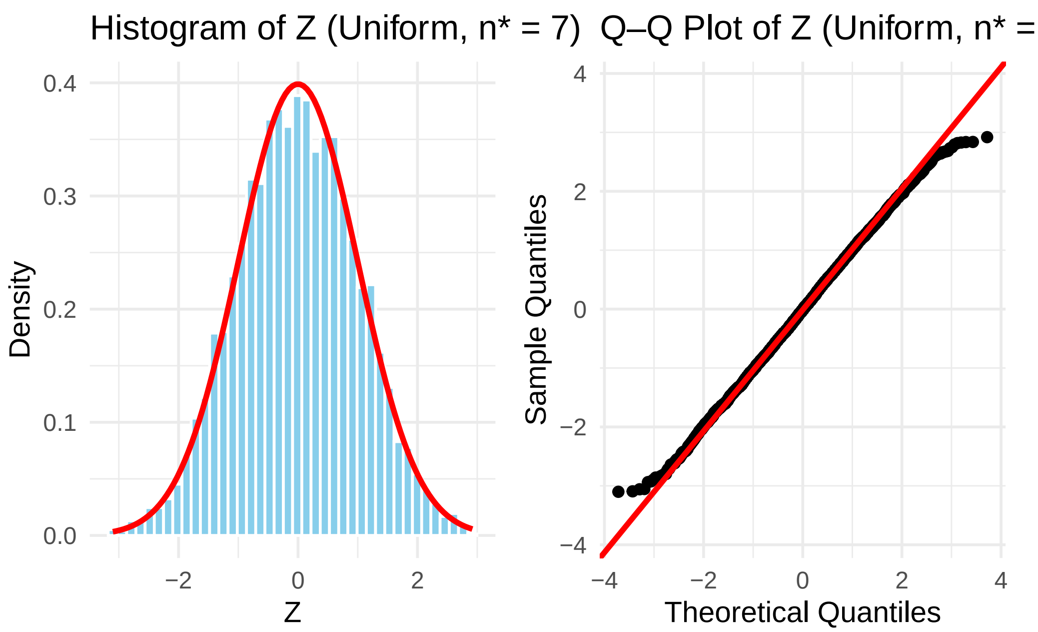
```
     n        A2          p   Abs_Skew Abs_Kurtosis
A    5 0.5705819 0.13832642 0.02330459   0.28942227
A1   6 0.8360669 0.03100621 0.02207999   0.26158005
A2   7 0.6346649 0.09753393 0.01802869   0.19084264
A3   8 0.3769694 0.40881829 0.01992096   0.14543440
A4   9 0.4339356 0.30042675 0.02330854   0.15543053
A5  10 0.1308610 0.98211166 0.02502674   0.11001927
A6  11 0.2970789 0.58970701 0.02964784   0.07089911
A7  12 0.1979061 0.88682635 0.03669547   0.08751041
A8  13 0.4015081 0.35840718 0.02212739   0.06202901
A9  14 0.1733253 0.92726609 0.03511768   0.13113398
A10 15 0.5443130 0.16121335 0.02491716   0.18410541
```

```r
n_star_uniform_refined <- results_uniform_refine %>%
  filter(
    p >= 0.05,
    Abs_Skew <= 0.1,
    Abs_Kurtosis <= 0.2
  ) %>%
  arrange(n) %>%
  slice(1) %>%
  pull(n)

n_star_uniform_refined
```

```
[1] 7
```

```r
set.seed(42)
```

```
sample_means_star <- numeric(reps)

for (i in 1:reps) {
  samples <- runif(n_star_uniform_refined, min = a_uni, max = b_uni)
  sample_means_star[i] <- mean(samples)
}

Z_star_uni_refined <- (sample_means_star - mu_uni) / (sigma_uni / sqrt(n_star_uniform_re

plot_hist_qq(Z_star_uni_refined, dist_name = "Uniform", n_star = n_star_uniform_refined)
```



2. **Chi-square distribution**

```
set.seed(42)
df_chi <- 50

mu_chi <- df_chi
sigma_chi <- sqrt(df_chi * 2)

# Data frame to store results
results_chisq <- data.frame(
  n = integer(),
  A2 = numeric(),        # Anderson-Darling statistic
  p = numeric(),
```

```
  Abs_Skew = numeric(),  # Absolute Skewness
  Abs_Kurtosis = numeric() # Absolute Excess Kurtosis
)

for (n in n_values) {
  sample_means <- numeric(reps)

  for (i in 1:reps) {
    samples <- rchisq(n, df = df_chi)
    sample_means[i] <- mean(samples)
  }

  # Calculate the standardized Z-statistic)
  Z <- (sample_means - mu_chi) / (sigma_chi / sqrt(n))


  # Anderson-Darling Statistic
  Z_sub <- sample(Z, 500)
  ad <- ad.test(Z_sub)
  ad_stat <- ad$statistic
  p_value <- ad$p.value

  # Absolute Skewness
  skewness_val <- abs(e1071::skewness(Z))

  # Absolute Excess Kurtosis
  kurtosis_val <- abs(e1071::kurtosis(Z))

  # Store results for the current n
  results_chisq <- rbind(results_chisq, data.frame(
    n = n,
    A2 = ad_stat,
    p = p_value,
    Abs_Skew = skewness_val,
    Abs_Kurtosis = kurtosis_val
  ))
}

print(results_chisq)
```

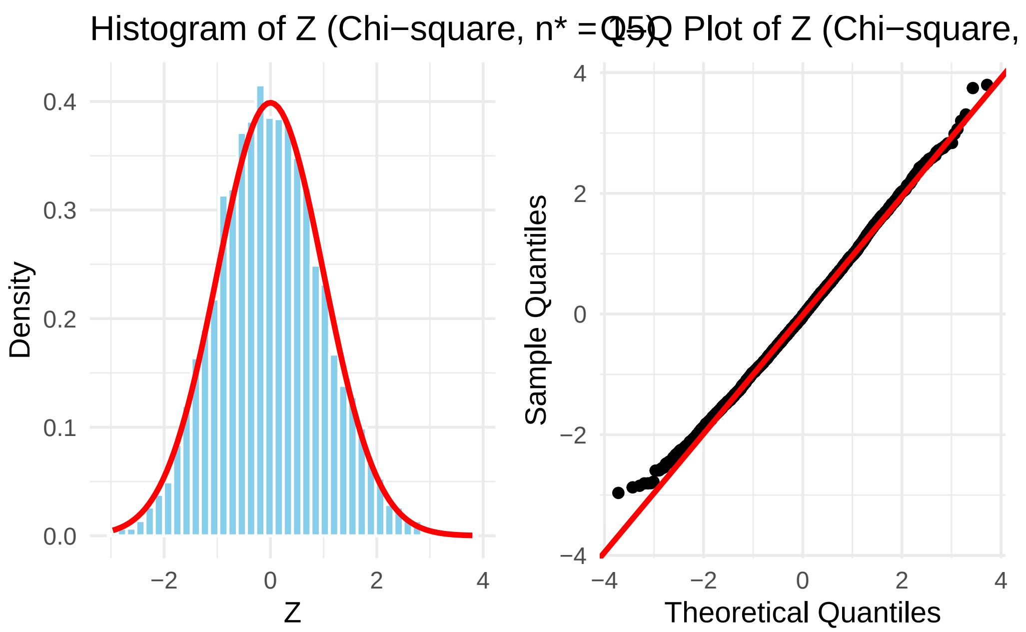
```
      n        A2          p   Abs_Skew Abs_Kurtosis
A     5 0.5732752 0.13615715 0.16682553  0.078957428
```

```
A1   10 0.5184759 0.18706591 0.15443982  0.025527800
A2   15 0.6572291 0.08577527 0.08419590  0.006810319
A3   20 0.1373528 0.97664084 0.06570533  0.022526609
A4   25 0.6300811 0.10011309 0.11527464  0.037322333
A5   30 0.9481903 0.01639279 0.10881077  0.214079339
A6   35 0.1774887 0.92010070 0.11183483  0.179111418
A7   40 0.4875011 0.22303582 0.04332635  0.024304017
A8   45 0.4534281 0.26981261 0.09484053  0.063136891
A9   50 0.5412078 0.16413704 0.09646852  0.066979971
A10  60 0.4702079 0.24576267 0.04163882  0.012670401
A11  70 0.2156375 0.84605489 0.03950165  0.004065862
A12  80 0.5299010 0.17519851 0.05077990  0.028444477
A13 100 0.5503404 0.15567425 0.05312449  0.029361179
```

```r
n_star_chi <- results_chisq %>%
  filter(
    p >= 0.05,
    Abs_Skew <= skew_threshold,
    Abs_Kurtosis <= kurt_threshold
  ) %>%
  arrange(n) %>%
  slice(1) %>%
  pull(n)

n_star_chi
```

```
[1] 15
```

```r
set.seed(42)

sample_means_star <- numeric(reps)

for (i in 1:reps) {
  samples <- rchisq(n_star_chi, df = df_chi)
  sample_means_star[i] <- mean(samples)
}

Z_star_chi <- (sample_means_star - mu_chi) / (sigma_chi / sqrt(n_star_chi))

plot_hist_qq(Z_star_chi, dist_name = "Chi-square", n_star = n_star_chi)
```



3. **Log-normal distribution**

```
set.seed(42)

mu_l <- 0
sigma_l <- 0.3

mu_log <- exp(mu_l + sigma_l^2 / 2)
sigma_log <- sqrt((exp(sigma_l^2) - 1) * exp(2 * mu_l + sigma_l^2))

# Data frame to store results
results_lognormal <- data.frame(
  n = integer(),
  A2 = numeric(),        # Anderson-Darling statistic
  p = numeric(),
  Abs_Skew = numeric(),  # Absolute Skewness
  Abs_Kurtosis = numeric() # Absolute Excess Kurtosis
)

for (n in n_values) {
  sample_means <- numeric(reps)

  for (i in 1:reps) {
    samples <- rlnorm(n, meanlog = mu_l, sdlog = sigma_l)
```

```
    sample_means[i] <- mean(samples)
  }

  # Calculate the standardized Z-statistic)
  Z <- (sample_means - mu_log) / (sigma_log / sqrt(n))

  Z_sub <- sample(Z, 500)
  ad <- ad.test(Z_sub)
  ad_stat <- ad$statistic
  p_value <- ad$p.value

  # Absolute Skewness
  skewness_val <- abs(e1071::skewness(Z))

  # Absolute Excess Kurtosis
  kurtosis_val <- abs(e1071::kurtosis(Z))

  # Store results for the current n
  results_lognormal <- rbind(results_lognormal, data.frame(
    n = n,
    A2 = ad_stat,
    p = p_value,
    Abs_Skew = skewness_val,
    Abs_Kurtosis = kurtosis_val
  ))
}

print(results_lognormal)
```

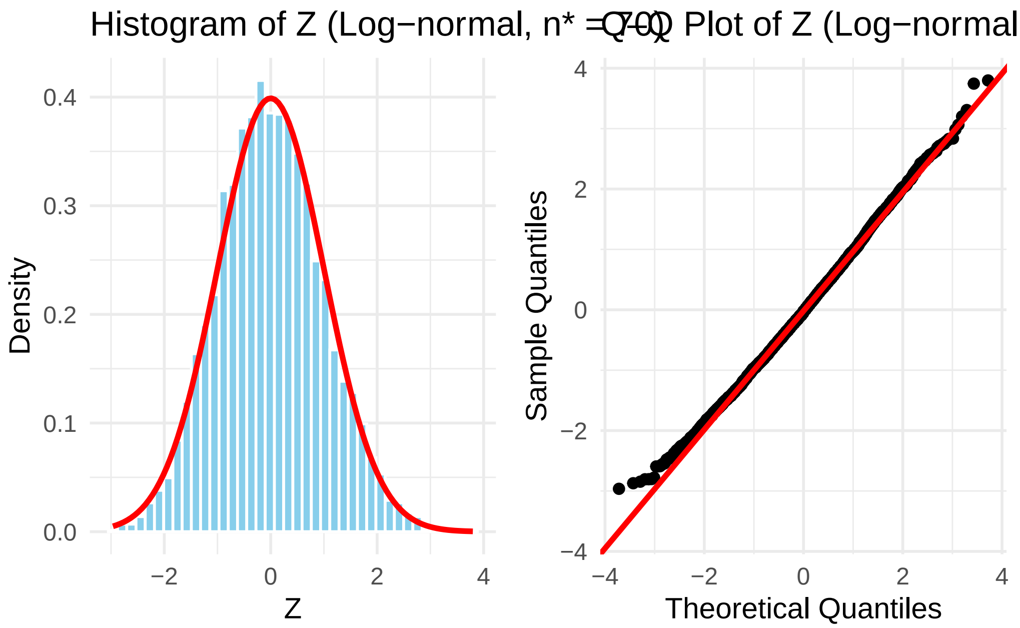
```
     n        A2            p   Abs_Skew Abs_Kurtosis
A    5 1.7875663 0.0001409223 0.40408599 0.4677737463
A1  10 0.9062295 0.0208073344 0.33392700 0.3036954367
A2  15 0.3986557 0.3639631639 0.22411472 0.0093324992
A3  20 0.2600767 0.7098642902 0.19508766 0.0807188900
A4  25 0.9531801 0.0159345326 0.14056976 0.0004771369
A5  30 0.8502094 0.0286104266 0.19543414 0.0716011012
A6  35 0.6116176 0.1112121989 0.23668660 0.1708179978
A7  40 0.8575859 0.0274352903 0.12370493 0.0150989175
A8  45 0.7291306 0.0569677630 0.14074703 0.0144546453
A9  50 0.3512971 0.4683271642 0.12849942 0.0373065522
A10 60 0.2780741 0.6494665205 0.10177115 0.0221296640
A11 70 0.2746755 0.6607532812 0.09368066 0.0202819878
```

```
A12  80 0.4002863 0.3607776124 0.10560299 0.0671626944
A13 100 0.7733946 0.0442848162 0.10236579 0.1194804410
```

```r
n_star_log <- results_lognormal %>%
  filter(
    p >= 0.05,
    Abs_Skew <= skew_threshold,
    Abs_Kurtosis <= kurt_threshold
  ) %>%
  arrange(n) %>%
  slice(1) %>%
  pull(n)

n_star_log
```

```
[1] 70
```

```r
set.seed(42)

sample_means_star <- numeric(reps)

for (i in 1:reps) {
  samples <- rlnorm(n_star_log, meanlog = mu_l, sdlog = sigma_l)
  sample_means_star[i] <- mean(samples)
}

Z_star_log <- (sample_means_star - mu_log) / (sigma_log / sqrt(n_star_log))

plot_hist_qq(Z_star_chi, dist_name = "Log-normal", n_star = n_star_log)
```



4. **Poisson distribution**

```
set.seed(42)
lambda_p <- 15

mu_p <- lambda_p
sigma_p <- sqrt(lambda_p)

# Data frame to store results
results_poisson <- data.frame(
  n = integer(),
  A2 = numeric(),         # Anderson-Darling statistic
  Abs_Skew = numeric(),   # Absolute Skewness
  p = numeric(),
  Abs_Kurtosis = numeric() # Absolute Excess Kurtosis
)

for (n in n_values) {
  sample_means <- numeric(reps)

  for (i in 1:reps) {
    samples <- rpois(n, lambda = lambda_p)
    sample_means[i] <- mean(samples)
  }
```

```r
  # Calculate the standardized Z-statistic)
  Z <- (sample_means - mu_p) / (sigma_p / sqrt(n))


  # Anderson-Darling Statistic
  Z_sub <- sample(Z, 500)
  ad <- ad.test(Z_sub)
  ad_stat <- ad$statistic
  p_value <- ad$p.value

  # Absolute Skewness
  skewness_val <- abs(e1071::skewness(Z))

  # Absolute Excess Kurtosis
  kurtosis_val <- abs(e1071::kurtosis(Z))

  # Store results for the current n
  results_poisson <- rbind(results_poisson, data.frame(
    n = n,
    A2 = ad_stat,
    p = p_value,
    Abs_Skew = skewness_val,
    Abs_Kurtosis = kurtosis_val
  ))
}

print(results_poisson)
```

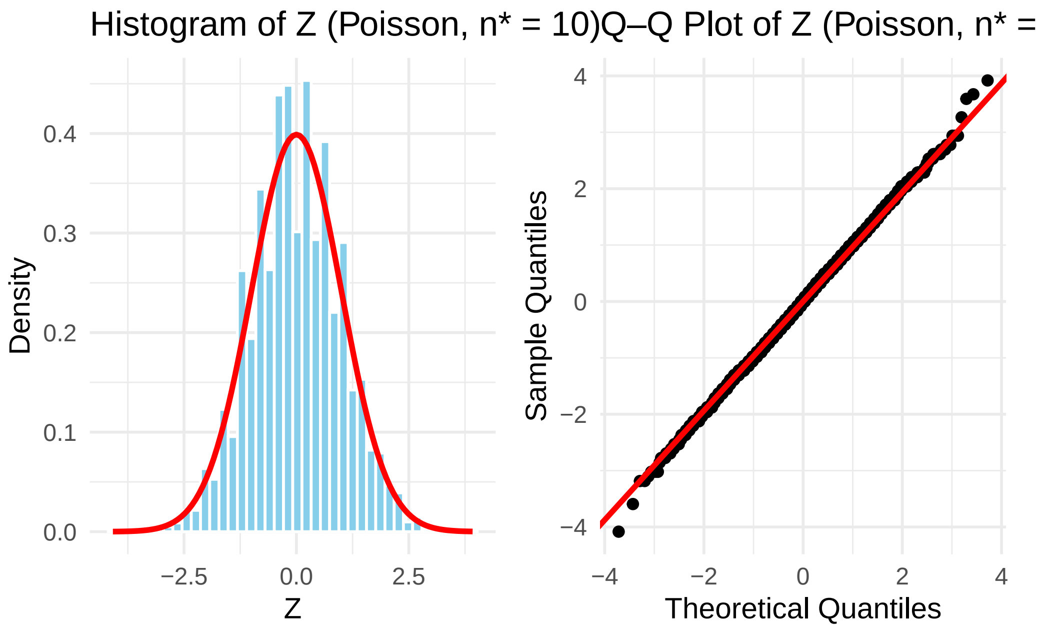
```
      n        A2          p    Abs_Skew Abs_Kurtosis
A     5 0.9662105 0.01479741 0.110842505  0.028532041
A1   10 0.6043094 0.11593850 0.089676570  0.065558470
A2   15 0.3050317 0.56738202 0.056902292  0.051423449
A3   20 0.2886831 0.61521497 0.094090646  0.007618793
A4   25 0.4482180 0.27770457 0.044931701  0.054883114
A5   30 0.2526782 0.73458018 0.112562308  0.107066934
A6   35 0.7520704 0.04999679 0.028465812  0.007990664
A7   40 0.5862116 0.12616739 0.028415772  0.028638058
A8   45 0.6301768 0.10005857 0.096716376  0.005445658
A9   50 0.1949354 0.89115796 0.014950469  0.005423548
A10  60 0.4994533 0.20846769 0.083551359  0.016322852
A11  70 0.5969414 0.11839840 0.008434018  0.080878251
A12  80 0.4541727 0.26870159 0.048218473  0.024274672
```

```
A13 100 0.7882637 0.04069312 0.020314057  0.103921708
```

```r
n_star_p <- results_poisson %>%
  filter(
    p >= 0.05,
    Abs_Skew <= skew_threshold,
    Abs_Kurtosis <= kurt_threshold
  ) %>%
  arrange(n) %>%
  slice(1) %>%
  pull(n)

n_star_p
```

```
[1] 10
```

```r
set.seed(42)

sample_means_star <- numeric(reps)

for (i in 1:reps) {
  samples <- rpois(n_star_p, lambda = lambda_p)
  sample_means_star[i] <- mean(samples)
}

Z_star_p <- (sample_means_star - mu_p) / (sigma_p / sqrt(n_star_p))

plot_hist_qq(Z_star_p, dist_name = "Poisson", n_star = n_star_p)
```



5. **t-distribution**

```
set.seed(42)
df_t <- 10

mu_t <- 0
sigma_t <- sqrt(df_t / (df_t - 2))

# Data frame to store results
results_t <- data.frame(
  n = integer(),
  A2 = numeric(),         # Anderson-Darling statistic
  p = numeric(),
  Abs_Skew = numeric(),   # Absolute Skewness
  Abs_Kurtosis = numeric() # Absolute Excess Kurtosis
)

for (n in n_values) {
  sample_means <- numeric(reps)

  for (i in 1:reps) {
    samples <- rt(n, df = df_t)
    sample_means[i] <- mean(samples)
  }
```

```r
  # Calculate the standardized Z-statistic)
  Z <- (sample_means - mu_t) / (sigma_t / sqrt(n))


  # Anderson-Darling Statistic
  Z_sub <- sample(Z, 500)
  ad <- ad.test(Z_sub)
  ad_stat <- ad$statistic
  p_value <- ad$p.value

  # Absolute Skewness
  skewness_val <- abs(e1071::skewness(Z))

  # Absolute Excess Kurtosis
  kurtosis_val <- abs(e1071::kurtosis(Z))

  # Store results for the current n
  results_t <- rbind(results_t, data.frame(
    n = n,
    A2 = ad_stat,
    p = p_value,
    Abs_Skew = skewness_val,
    Abs_Kurtosis = kurtosis_val
  ))
}

print(results_t)
```

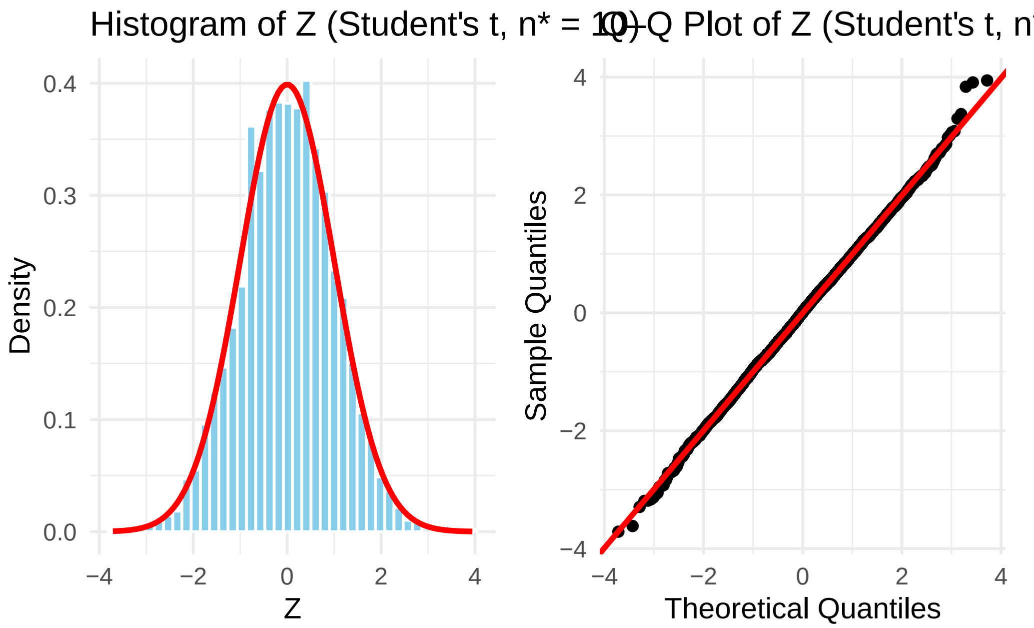
```
      n        A2           p    Abs_Skew Abs_Kurtosis
A     5 0.2992406 0.583441745 0.007059050   0.28019625
A1   10 0.2092252 0.862235415 0.006055159   0.11480775
A2   15 0.4595249 0.260837049 0.022523763   0.13194381
A3   20 0.3930243 0.375161587 0.085108370   0.02075104
A4   25 0.7306693 0.056471140 0.013614309   0.10696467
A5   30 0.3781581 0.406235878 0.005349626   0.10069553
A6   35 0.6453108 0.091797554 0.010369816   0.03089532
A7   40 0.8358449 0.031045386 0.007186639   0.17742978
A8   45 0.3157161 0.540992524 0.008619828   0.12330334
A9   50 0.5030617 0.204243589 0.054645003   0.06915963
A10  60 1.1006513 0.006896097 0.056223564   0.03625932
A11  70 0.2578312 0.717403179 0.015041960   0.04101110
A12  80 0.3612596 0.444365408 0.017049477   0.07823047
```

```
A13 100 0.3067460 0.562848264 0.010429298   0.03561274
```

```r
n_star_t <- results_t %>%
  filter(
    p >= 0.05,
    Abs_Skew <= skew_threshold,
    Abs_Kurtosis <= kurt_threshold
  ) %>%
  arrange(n) %>%
  slice(1) %>%
  pull(n)

n_star_t
```

```
[1] 10
```

```r
set.seed(42)

sample_means_star <- numeric(reps)

for (i in 1:reps) {
  samples <- rt(n_star_t, df = df_t)
  sample_means_star[i] <- mean(samples)
}

Z_star_t <- (sample_means_star - mu_t) / (sigma_t / sqrt(n_star_t))

plot_hist_qq(Z_star_t, dist_name = "Student's t", n_star = n_star_t)
```



6. **Bernoulli distribution**

```
set.seed(42)
n_values_b <- c(5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 100, 120, 140, 160, 1
p_b <- 0.5

mu_b <- p_b
sigma_b <- sqrt(p_b * (1 - p_b))

# Data frame to store results
results_bern <- data.frame(
  n = integer(),
  A2 = numeric(),         # Anderson-Darling statistic
  p = numeric(),
  Abs_Skew = numeric(),   # Absolute Skewness
  Abs_Kurtosis = numeric() # Absolute Excess Kurtosis
)

for (n in n_values_b) {
  sample_means <- numeric(reps)

  for (i in 1:reps) {
    samples <- rbinom(n, size = 1, prob = p_b)
    sample_means[i] <- mean(samples)
```

```
  }

  # Calculate the standardized Z-statistic)
  Z <- (sample_means - mu_b) / (sigma_b / sqrt(n))


  # Anderson-Darling Statistic
  Z_sub <- sample(Z, 500)
  ad <- ad.test(Z_sub)
  ad_stat <- ad$statistic
  p_value <- ad$p.value

  # Absolute Skewness
  skewness_val <- abs(e1071::skewness(Z))

  # Absolute Excess Kurtosis
  kurtosis_val <- abs(e1071::kurtosis(Z))

  # Store results for the current n
  results_bern <- rbind(results_bern, data.frame(
    n = n,
    A2 = ad_stat,
    p = p_value,
    Abs_Skew = skewness_val,
    Abs_Kurtosis = kurtosis_val
  ))
}

print(results_bern)
```

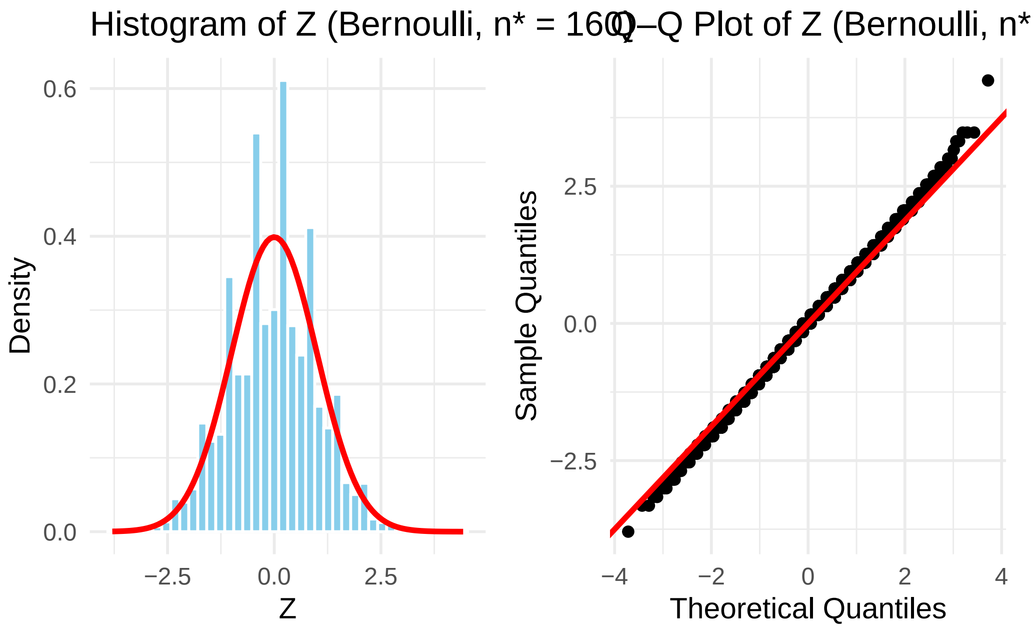
```
     n         A2            p     Abs_Skew Abs_Kurtosis
A    5 17.0519315 3.700000e-24 0.0137221358 0.4428615123
A1  10  9.1593908 3.145121e-22 0.0280026675 0.3135786158
A2  15  6.0919042 5.439402e-15 0.0177833860 0.1133788118
A3  20  4.8458272 5.239534e-12 0.0129738322 0.0749064654
A4  25  3.7765075 1.994563e-09 0.0394034483 0.0980778816
A5  30  3.2215858 4.429202e-08 0.0400593697 0.1254296363
A6  35  2.9421679 2.119345e-07 0.0227254909 0.1081763800
A7  40  2.3925165 4.648354e-06 0.0185496391 0.0119875567
A8  45  1.9870944 4.566968e-05 0.0316551601 0.0585186033
A9  50  1.8537974 9.693338e-05 0.0163120285 0.0841468003
A10 60  2.6235813 1.267437e-06 0.0057393579 0.0075611768
```

```
A11  70  1.5527354 5.317892e-04 0.0146243461 0.0749990222
A12  80  1.2465322 3.013955e-03 0.0065873009 0.0198181146
A13 100  0.9857350 1.324381e-02 0.0177255247 0.0012487091
A14 120  0.9079050 2.061012e-02 0.0084670545 0.0193763569
A15 140  0.9573162 1.556442e-02 0.0285185538 0.1139721315
A16 160  0.7170147 6.103411e-02 0.0424784173 0.0004635463
A17 180  0.6991906 6.755034e-02 0.0009594619 0.0386866163
A18 200  0.6543557 8.719000e-02 0.0365441370 0.0580936040
```

```r
n_star_b <- results_bern %>%
  filter(
    p >= 0.05,
    Abs_Skew <= skew_threshold,
    Abs_Kurtosis <= kurt_threshold
  ) %>%
  arrange(n) %>%
  slice(1) %>%
  pull(n)

n_star_b
```

```
[1] 160
```

```r
set.seed(42)

sample_means_star <- numeric(reps)

for (i in 1:reps) {
  samples <- rbinom(n_star_b, size = 1, prob = p_b)
  sample_means_star[i] <- mean(samples)
}

Z_star_b <- (sample_means_star - mu_b) / (sigma_b / sqrt(n_star_b))

plot_hist_qq(Z_star_b, dist_name = "Bernoulli", n_star = n_star_b)
```



7. **Mixed-normal distribution**

```
set.seed(42)
n_values_m = c(100, 120, 150, 180, 200, 250, 300)
w1 <- 0.9
w2 <- 1 - w1
mu1 <- 2
mu2 <- -2
sd1 <- 1
sd2 <- 1

mu_m <- w1 * mu1 + w2 * mu2
sigma_m <- sqrt(w1 * (mu1^2 + sd1^2) + w2 * (mu2^2 + sd2^2) - mu_m^2)

rmixnorm <- function(n, w1, mu1, sd1, w2, mu2, sd2) {
  component <- sample(c(1, 2), size = n, replace = TRUE, prob = c(w1, w2))
  samples <- numeric(n)
  n1 <- sum(component == 1)
  samples[component == 1] <- rnorm(n1, mean = mu1, sd = sd1)
  n2 <- sum(component == 2)
  samples[component == 2] <- rnorm(n2, mean = mu2, sd = sd2)

  return(samples)
}
```

```
# Data frame to store results
results_mixnorm <- data.frame(
  n = integer(),
  A2 = numeric(),        # Anderson-Darling statistic
  p = numeric(),
  Abs_Skew = numeric(),  # Absolute Skewness
  Abs_Kurtosis = numeric() # Absolute Excess Kurtosis
)

for (n in n_values_m) {
  sample_means <- numeric(reps)

  for (i in 1:reps) {
    samples <- rmixnorm(n, w1, mu1, sd1, w2, mu2, sd2)
    sample_means[i] <- mean(samples)
  }

  # Calculate the standardized Z-statistic)
  Z <- (sample_means - mu_m) / (sigma_m / sqrt(n))


  # Anderson-Darling Statistic
  Z_sub <- sample(Z, 500)
  ad <- ad.test(Z_sub)
  ad_stat <- ad$statistic
  p_value <- ad$p.value

  # Absolute Skewness
  skewness_val <- abs(e1071::skewness(Z))

  # Absolute Excess Kurtosis
  kurtosis_val <- abs(e1071::kurtosis(Z))

  # Store results for the current n
  results_mixnorm <- rbind(results_mixnorm, data.frame(
    n = n,
    A2 = ad_stat,
    p = p_value,
    Abs_Skew = skewness_val,
    Abs_Kurtosis = kurtosis_val
  ))
```

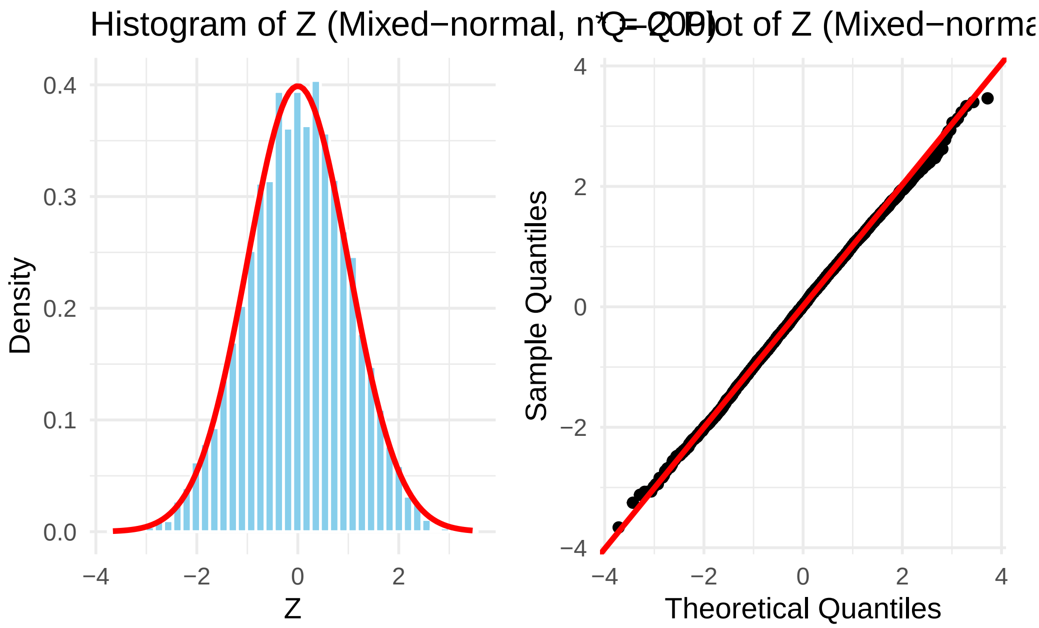
```r
}

print(results_mixnorm)
```

```
     n        A2         p    Abs_Skew Abs_Kurtosis
A  100 0.1237805 0.9869139 0.107730252   0.05161265
A1 120 0.3685627 0.4275077 0.161009738   0.02226480
A2 150 0.1503903 0.9622968 0.116148523   0.04535298
A3 180 0.6281105 0.1012428 0.133639712   0.02007044
A4 200 0.2895117 0.6126237 0.099653017   0.05059053
A5 250 0.3747760 0.4136221 0.006716221   0.04347650
A6 300 0.3622303 0.4420906 0.009796090   0.01231563
```

```r
n_star_m <- results_mixnorm %>%
  filter(
    p >= 0.05,
    Abs_Skew <= skew_threshold,
    Abs_Kurtosis <= kurt_threshold
  ) %>%
  arrange(n) %>%
  slice(1) %>%
  pull(n)

n_star_m
```

```
[1] 200
```

```r
set.seed(42)

sample_means_star <- numeric(reps)

for (i in 1:reps) {
  samples <- rmixnorm(n_star_m, w1, mu1, sd1, w2, mu2, sd2)
  sample_means_star[i] <- mean(samples)
}

Z_star_m <- (sample_means_star - mu_m) / (sigma_m / sqrt(n_star_m))

plot_hist_qq(Z_star_m, dist_name = "Mixed-normal", n_star = n_star_m)
```


Histogram of Z (Mixed−normal, n*=200)
Density
0.4
0.3
0.2
0.1
0.0
−4
−2
0
2
Z
Q−Q Plot of Z (Mixed−norma
Sample Quantiles
4
2
0
−2
−4
−4
−2
0
2
4
Theoretical Quantiles