phase5_assumption_testing.R

abdulbasir

2025-12-09

```r
# phase 5: assumption testing
#
# sample filtering, data transformation, and assumption checking including
# variance homogeneity tests and normality assessments

# sourcing configuration and utilities
source("config.R")
source("utils.R")

# loading required libraries
suppressPackageStartupMessages({
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(viridis) # adding colorblind-friendly palette
})

print_section_header("Phase 5: Assumption Testing")
```

```
##
## ======================================================================
## Phase 5: Assumption Testing
## ======================================================================
```

```r
# 1. loading classified data
input_file = file.path(RESULTS_DIR, "fda_analysis_clean_classified.csv")
classified_data = load_csv(input_file)

# 2. filtering analysis sample
analysis_data = classified_data %>%
  filter(therapeutic_area != "Uncertain")

cat(paste("Filtered to analysis sample:", nrow(analysis_data), "drugs\n"))
```

```
## Filtered to analysis sample: 1038 drugs
```

```r
cat(paste("Excluded uncertain classifications:", nrow(classified_data) - nrow(analysis_data), "\n"))
```

```
## Excluded uncertain classifications: 297
```

```r
# 3. creating factor variables
# CRITICAL BUG FIX: simplify review designation BEFORE creating factor
analysis_data = analysis_data %>%
  mutate(
    review_type_simplified = if_else(
      grepl("Priority", `Review Designation`),
      "Priority",
      "Standard"
    ),
    therapeutic_area_factor = factor(
      therapeutic_area,
      levels = c("Other", "Oncology")
    ),
    review_type_factor = factor(
      review_type_simplified,
      levels = c("Standard", "Priority")
    ),
    regulatory_era_factor = factor(
      assign_regulatory_era(`Approval Year`),
      levels = c("Pre-PDUFA", "Early-PDUFA", "Mid-PDUFA", "Post-FDASIA"),
      ordered = TRUE
    )
  )

cat("Factor variables created:\n")
```

```
## Factor variables created:
```

```r
cat("  therapeutic_area_factor:", paste(levels(analysis_data$therapeutic_area_factor), collapse = ", ")
```

```
##   therapeutic_area_factor: Other, Oncology
```

```r
cat("  review_type_factor:", paste(levels(analysis_data$review_type_factor), collapse = ", "), "\n")
```

```
##   review_type_factor: Standard, Priority
```

```r
cat("  regulatory_era_factor:", paste(levels(analysis_data$regulatory_era_factor), collapse = ", "), "\
```

```
##   regulatory_era_factor: Pre-PDUFA, Early-PDUFA, Mid-PDUFA, Post-FDASIA
```

```r
# 4. extracting response variables
analysis_data = analysis_data %>%
  mutate(
    review_time_days_response = review_time_days,
    log_review_time_days_response = log(review_time_days)
  )

missing_review = sum(is.na(analysis_data$review_time_days_response))
missing_log = sum(is.na(analysis_data$log_review_time_days_response))

cat("Response variables extracted:\n")
```

```
## Response variables extracted:
```

```
cat(sprintf("  review_time_days_response: %d values, %d missing\n",
            nrow(analysis_data), missing_review))
```

```
##   review_time_days_response: 1038 values, 0 missing
```

```
cat(sprintf("  log_review_time_days_response: %d values, %d missing\n",
            nrow(analysis_data), missing_log))
```

```
##   log_review_time_days_response: 1038 values, 0 missing
```

```
cat(sprintf("  Range (raw): [%d, %d]\n",
            min(analysis_data$review_time_days_response, na.rm = TRUE),
            max(analysis_data$review_time_days_response, na.rm = TRUE)))
```

```
##   Range (raw): [18, 3497]
```

```
cat(sprintf("  Range (log): [%.2f, %.2f]\n",
            min(analysis_data$log_review_time_days_response, na.rm = TRUE),
            max(analysis_data$log_review_time_days_response, na.rm = TRUE)))
```

```
##   Range (log): [2.89, 8.16]
```

```
cat(paste("No missing values:", ifelse(missing_review == 0 && missing_log == 0, "yes", "no"), "\n"))
```

```
## No missing values: yes
```

```
# 5. validating cell counts
crosstab = table(
  analysis_data$therapeutic_area_factor,
  analysis_data$review_type_factor
)

cat("Cell count validation:\n")
```

```
## Cell count validation:
```

```
print(addmargins(crosstab))
```

```
##
##             Standard Priority  Sum
##   Other          372      367  739
##   Oncology        75      224  299
##   Sum            447      591 1038
```

```
onc_priority = crosstab["Oncology", "Priority"]
onc_standard = crosstab["Oncology", "Standard"]
other_priority = crosstab["Other", "Priority"]
other_standard = crosstab["Other", "Standard"]

cat(sprintf("\nOncology-Priority: %d\n", onc_priority))
```

```
##
## Oncology-Priority: 224
```

```
cat(sprintf("Oncology-Standard: %d\n", onc_standard))
```

```
## Oncology-Standard: 75
```

```
cat(sprintf("Other-Priority: %d\n", other_priority))
```

```
## Other-Priority: 367
```

```
cat(sprintf("Other-Standard: %d\n", other_standard))
```

```
## Other-Standard: 372
```

```
cat(sprintf("Minimum cell count: %d\n", min(onc_priority, onc_standard, other_priority, other_standard))
```

```
## Minimum cell count: 75
```

```
# 6. extracting cell data for variance analysis
other_standard_raw = analysis_data %>%
  filter(therapeutic_area_factor == "Other", review_type_factor == "Standard") %>%
  pull(review_time_days_response)

other_priority_raw = analysis_data %>%
  filter(therapeutic_area_factor == "Other", review_type_factor == "Priority") %>%
  pull(review_time_days_response)

oncology_standard_raw = analysis_data %>%
  filter(therapeutic_area_factor == "Oncology", review_type_factor == "Standard") %>%
  pull(review_time_days_response)

oncology_priority_raw = analysis_data %>%
  filter(therapeutic_area_factor == "Oncology", review_type_factor == "Priority") %>%
  pull(review_time_days_response)

other_standard_log = analysis_data %>%
  filter(therapeutic_area_factor == "Other", review_type_factor == "Standard") %>%
  pull(log_review_time_days_response)

other_priority_log = analysis_data %>%
  filter(therapeutic_area_factor == "Other", review_type_factor == "Priority") %>%
```

```r
  pull(log_review_time_days_response)

oncology_standard_log = analysis_data %>%
  filter(therapeutic_area_factor == "Oncology", review_type_factor == "Standard") %>%
  pull(log_review_time_days_response)

oncology_priority_log = analysis_data %>%
  filter(therapeutic_area_factor == "Oncology", review_type_factor == "Priority") %>%
  pull(log_review_time_days_response)

cat("Cell data extracted:\n")
```

```
## Cell data extracted:
```

```r
cat("Raw scale (review_time_days):\n")
```

```
## Raw scale (review_time_days):
```

```r
cat(sprintf("  Other-Standard: n=%3d\n", length(other_standard_raw)))
```

```
##   Other-Standard: n=372
```

```r
cat(sprintf("  Other-Priority: n=%3d\n", length(other_priority_raw)))
```

```
##   Other-Priority: n=367
```

```r
cat(sprintf("  Oncology-Standard: n=%3d\n", length(oncology_standard_raw)))
```

```
##   Oncology-Standard: n= 75
```

```r
cat(sprintf("  Oncology-Priority: n=%3d\n", length(oncology_priority_raw)))
```

```
##   Oncology-Priority: n=224
```

```r
cat("\nLog scale (log_review_time_days):\n")
```

```
##
## Log scale (log_review_time_days):
```

```r
cat(sprintf("  Other-Standard: n=%3d\n", length(other_standard_log)))
```

```
##   Other-Standard: n=372
```

```r
cat(sprintf("  Other-Priority: n=%3d\n", length(other_priority_log)))
```

```
##   Other-Priority: n=367
```

```r
cat(sprintf("  Oncology-Standard: n=%3d\n", length(oncology_standard_log)))
```

```
##   Oncology-Standard: n= 75
```

```r
cat(sprintf("  Oncology-Priority: n=%3d\n", length(oncology_priority_log)))
```

```
##   Oncology-Priority: n=224
```

```r
# 7. calculating cell variances
var_other_standard_raw = var(other_standard_raw)
var_other_priority_raw = var(other_priority_raw)
var_oncology_standard_raw = var(oncology_standard_raw)
var_oncology_priority_raw = var(oncology_priority_raw)

var_other_standard_log = var(other_standard_log)
var_other_priority_log = var(other_priority_log)
var_oncology_standard_log = var(oncology_standard_log)
var_oncology_priority_log = var(oncology_priority_log)

variances_raw = c(var_other_standard_raw, var_other_priority_raw,
                  var_oncology_standard_raw, var_oncology_priority_raw)
variance_ratio_raw = max(variances_raw) / min(variances_raw)

variances_log = c(var_other_standard_log, var_other_priority_log,
                  var_oncology_standard_log, var_oncology_priority_log)
variance_ratio_log = max(variances_log) / min(variances_log)

cat("Cell variances:\n")
```

```
## Cell variances:
```

```r
cat("Raw scale (review_time_days):\n")
```

```
## Raw scale (review_time_days):
```

```r
cat(sprintf("  Other-Standard: %12.0f\n", var_other_standard_raw))
```

```
##   Other-Standard:       280855
```

```r
cat(sprintf("  Other-Priority: %12.0f\n", var_other_priority_raw))
```

```
##   Other-Priority:       198042
```

```r
cat(sprintf("  Oncology-Standard: %12.0f\n", var_oncology_standard_raw))
```

```
##   Oncology-Standard:       218286
```

```r
cat(sprintf("  Oncology-Priority: %12.0f\n", var_oncology_priority_raw))
```

```
##   Oncology-Priority:        79776
```

```r
cat(sprintf("  Variance ratio: %12.2f\n", variance_ratio_raw))
```

```
##   Variance ratio:         3.52
```

```r
cat("\nLog scale (log_review_time_days):\n")
```

```
##
## Log scale (log_review_time_days):
```

```r
cat(sprintf("  Other-Standard: %12.4f\n", var_other_standard_log))
```

```
##   Other-Standard:       0.3489
```

```r
cat(sprintf("  Other-Priority: %12.4f\n", var_other_priority_log))
```

```
##   Other-Priority:       0.5546
```

```r
cat(sprintf("  Oncology-Standard: %12.4f\n", var_oncology_standard_log))
```

```
##   Oncology-Standard:       0.3765
```

```r
cat(sprintf("  Oncology-Priority: %12.4f\n", var_oncology_priority_log))
```

```
##   Oncology-Priority:       0.3233
```

```r
cat(sprintf("  Variance ratio: %12.2f\n", variance_ratio_log))
```

```
##   Variance ratio:         1.72
```

```r
# 8. running Levene's test - raw scale
combined_raw = c(other_standard_raw, other_priority_raw,
                 oncology_standard_raw, oncology_priority_raw)
groups_raw = rep(c("other_standard", "other_priority", "oncology_standard", "oncology_priority"),
                 times = c(length(other_standard_raw), length(other_priority_raw),
                           length(oncology_standard_raw), length(oncology_priority_raw)))

if (!require(car, quietly = TRUE)) {
  install.packages("car", quiet = TRUE)
  library(car)
}
```

```
##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```
levene_raw_result = leveneTest(combined_raw ~ groups_raw, center = median)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to factor.
```

```
levene_raw_stat = levene_raw_result$`F value`[1]
levene_raw_p = levene_raw_result$`Pr(>F)`[1]

cat(sprintf("Levene's test - raw scale: F=%.4f, p=%.4f\n", levene_raw_stat, levene_raw_p))
```

```
## Levene's test - raw scale: F=19.5679, p=0.0000
```

```
if (levene_raw_p < ALPHA) {
  cat("Result: reject homogeneity (p < 0.05), variances significantly different\n")
} else {
  cat("Result: fail to reject homogeneity (p >= 0.05), variances not significantly different\n")
}
```

```
## Result: reject homogeneity (p < 0.05), variances significantly different
```

```
# 9. running Levene's test - log scale
combined_log = c(other_standard_log, other_priority_log,
                 oncology_standard_log, oncology_priority_log)
groups_log = rep(c("other_standard", "other_priority", "oncology_standard", "oncology_priority"),
                 times = c(length(other_standard_log), length(other_priority_log),
                           length(oncology_standard_log), length(oncology_priority_log)))

levene_log_result = leveneTest(combined_log ~ groups_log, center = median)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to factor.
```

```
levene_log_stat = levene_log_result$`F value`[1]
levene_log_p = levene_log_result$`Pr(>F)`[1]

cat(sprintf("Levene's test - log scale: F=%.4f, p=%.4f\n", levene_log_stat, levene_log_p))
```

```
## Levene's test - log scale: F=8.6675, p=0.0000
```

```
if (levene_log_p < ALPHA) {
  cat("Result: reject homogeneity (p < 0.05), variances significantly different\n")
} else {
  cat("Result: fail to reject homogeneity (p >= 0.05), variances not significantly different\n")
}
```

```
## Result: reject homogeneity (p < 0.05), variances significantly different
```

```
# 10. fitting raw model for residuals
model_raw = lm(review_time_days_response ~ therapeutic_area_factor + review_type_factor,
               data = analysis_data)
residuals_raw = residuals(model_raw)

cat("Model: review_time_days ~ therapeutic_area + review_type\n")
```

```
## Model: review_time_days ~ therapeutic_area + review_type
```

```r
cat(sprintf("N observations: %d\n", length(residuals_raw)))
```

```
## N observations: 1038
```

```r
cat(sprintf("Residual mean: %.4f\n", mean(residuals_raw)))
```

```
## Residual mean: 0.0000
```

```r
cat(sprintf("Residual std: %.2f\n", sd(residuals_raw)))
```

```
## Residual std: 450.69
```

```r
# 11. fitting log model for residuals
model_log = lm(log_review_time_days_response ~ therapeutic_area_factor + review_type_factor,
               data = analysis_data)
residuals_log = residuals(model_log)

cat("Model: log(review_time_days) ~ therapeutic_area + review_type\n")
```

```
## Model: log(review_time_days) ~ therapeutic_area + review_type
```

```r
cat(sprintf("N observations: %d\n", length(residuals_log)))
```

```
## N observations: 1038
```

```r
cat(sprintf("Residual mean: %.6f\n", mean(residuals_log)))
```

```
## Residual mean: -0.000000
```

```r
cat(sprintf("Residual std: %.4f\n", sd(residuals_log)))
```

```
## Residual std: 0.6458
```

```r
# 12. generating normality visualizations - raw scale
shapiro_raw = shapiro.test(residuals_raw)
w_stat_raw = shapiro_raw$statistic
p_value_raw = shapiro_raw$p.value

png(file.path(FIGURES_DIR, "qq_plot_raw_scale.png"), width = 14 * DPI, height = 5 * DPI, res = DPI)
par(mfrow = c(1, 2))

# qq-plot
qqnorm(residuals_raw, main = "QQ-Plot: Raw Scale Residuals", cex.main = 1.2, font.main = 2)
qqline(residuals_raw, col = "red", lwd = 2)

# histogram with normal overlay
```

```r
hist(residuals_raw, breaks = 40, freq = FALSE, col = rgb(0.27, 0.51, 0.71, 0.7),
     border = "black", main = "Histogram: Raw Scale Residuals",
     xlab = "Residuals (days)", ylab = "Density", cex.main = 1.2, font.main = 2)
mu = mean(residuals_raw)
sigma = sd(residuals_raw)
x = seq(min(residuals_raw), max(residuals_raw), length = 100)
lines(x, dnorm(x, mu, sigma), col = "red", lwd = 2)
legend("topright", legend = "Normal curve", col = "red", lwd = 2)

dev.off()
```

```
## pdf
##   2
```

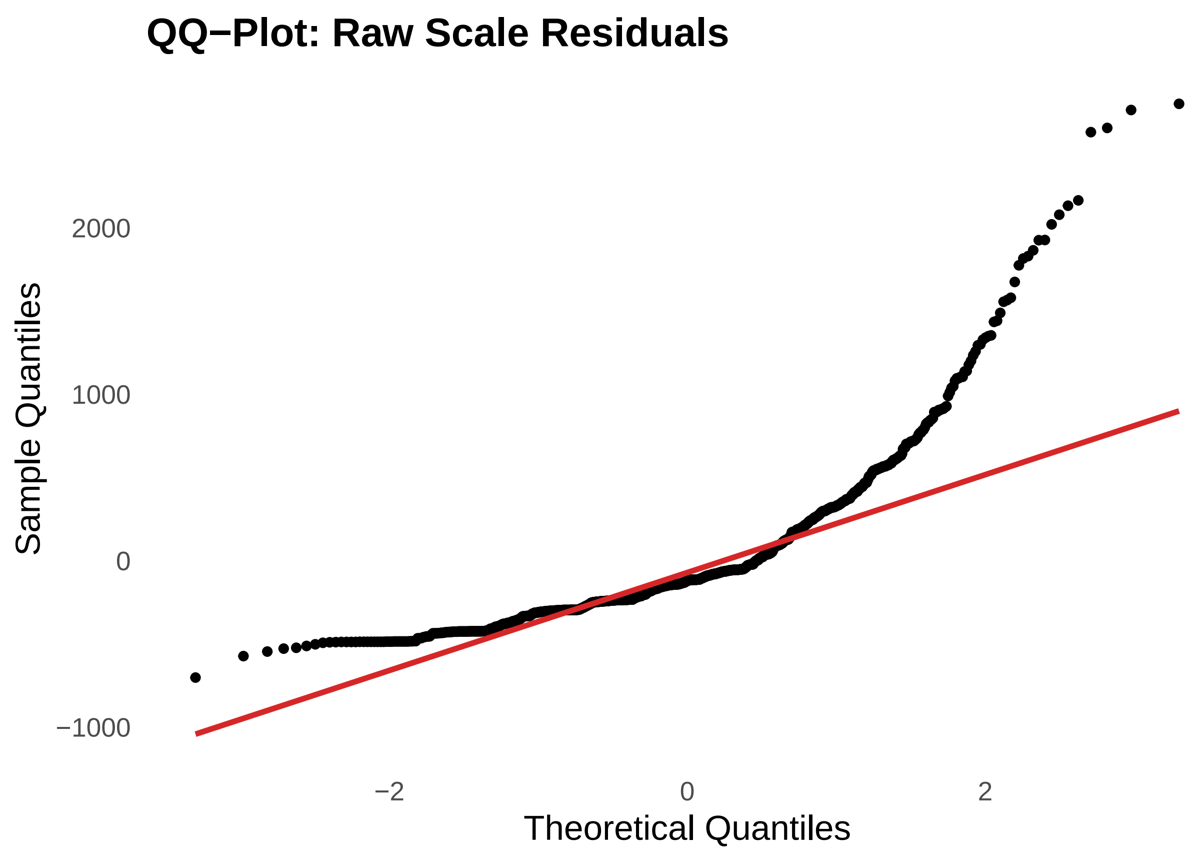
```r
cat(sprintf("Shapiro-Wilk test: W=%.6f, p=%.4e\n", w_stat_raw, p_value_raw))
```

```
## Shapiro-Wilk test: W=0.784619, p=6.9301e-35
```

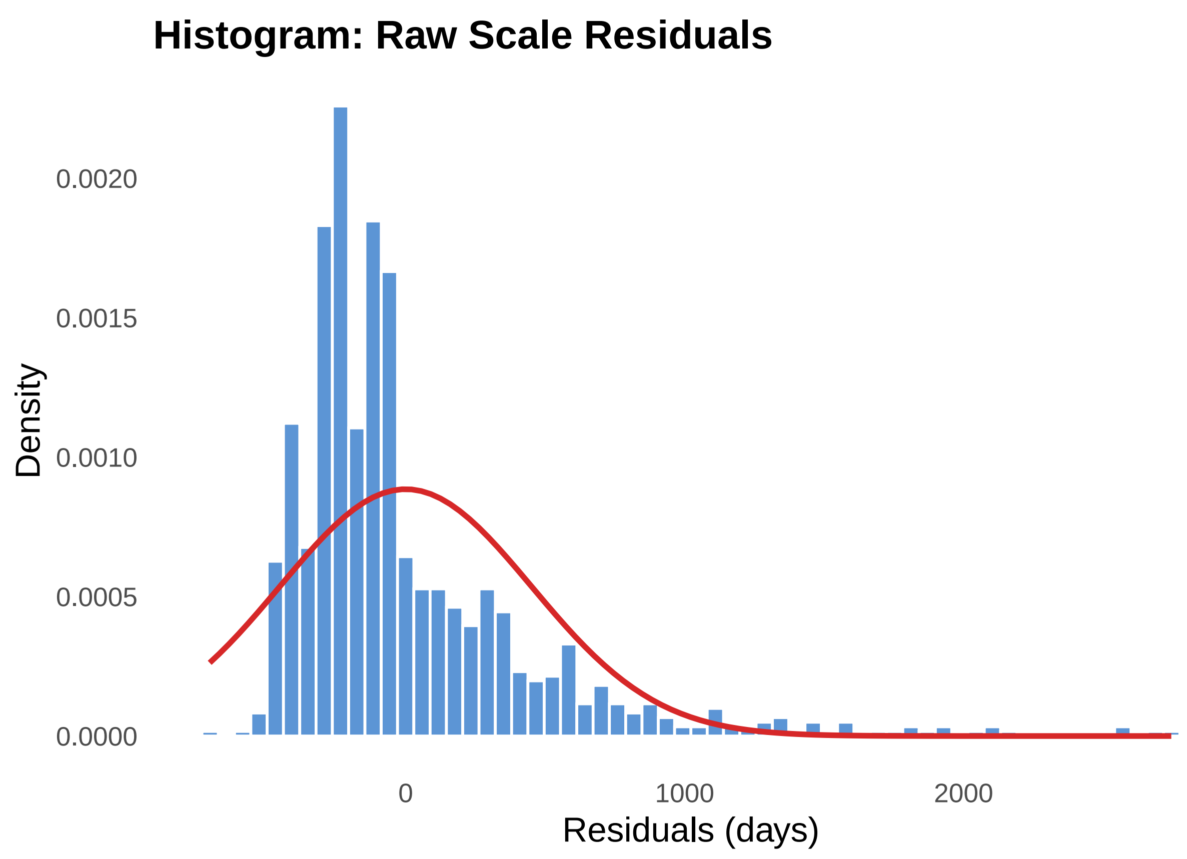
```r
cat(sprintf("Normality: %s at  =%.2f\n",
            ifelse(p_value_raw < ALPHA, "rejected", "not rejected"), ALPHA))
```

```
## Normality: rejected at  =0.05
```

```r
qq_hist_palette = viridis::viridis(3, option = "plasma")

raw_resid_df = data.frame(residuals_raw = residuals_raw)

qq_raw_plot = ggplot(raw_resid_df, aes(sample = residuals_raw)) +
  stat_qq(size = 1.2, color = "black") +
  stat_qq_line(color = "#d62728", linewidth = 1) +
  labs(
    title = "QQ-Plot: Raw Scale Residuals",
    x = "Theoretical Quantiles",
    y = "Sample Quantiles"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", size = 15),
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank()
  )

hist_raw_plot = ggplot(raw_resid_df, aes(x = residuals_raw)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 60,
                 fill = "#4b8ad1",
                 color = "white",
                 alpha = 0.9) +
  stat_function(fun = dnorm,
                args = list(mean = mean(residuals_raw), sd = sd(residuals_raw)),
                color = "#d62728",
                 linewidth = 1) +
  labs(
    title = "Histogram: Raw Scale Residuals",
    x = "Residuals (days)",
    y = "Density"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", size = 15),
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank()
  )

print(qq_raw_plot)
```



```
print(hist_raw_plot)
```



```r
ggsave(
  file.path(FIGURES_DIR, "qq_plot_raw_scale_gg.png"),
  plot = qq_raw_plot,
  width = 8,
  height = 6,
  dpi = DPI,
  bg = "white"
)

ggsave(
  file.path(FIGURES_DIR, "histogram_raw_scale_gg.png"),
  plot = hist_raw_plot,
  width = 8,
  height = 6,
  dpi = DPI,
  bg = "white"
)

# 13. generating normality visualizations - log scale
shapiro_log = shapiro.test(residuals_log)
w_stat_log = shapiro_log$statistic
p_value_log = shapiro_log$p.value

png(file.path(FIGURES_DIR, "qq_plot_log_scale.png"), width = 14 * DPI, height = 5 * DPI, res = DPI)
par(mfrow = c(1, 2))
```

```r
# qq-plot
qqnorm(residuals_log, main = "QQ-Plot: Log Scale Residuals", cex.main = 1.2, font.main = 2)
qqline(residuals_log, col = "red", lwd = 2)

# histogram with normal overlay
hist(residuals_log, breaks = 40, freq = FALSE, col = rgb(0.27, 0.51, 0.71, 0.7),
     border = "black", main = "Histogram: Log Scale Residuals",
     xlab = "Residuals (log scale)", ylab = "Density", cex.main = 1.2, font.main = 2)
mu = mean(residuals_log)
sigma = sd(residuals_log)
x = seq(min(residuals_log), max(residuals_log), length = 100)
lines(x, dnorm(x, mu, sigma), col = "red", lwd = 2)
legend("topright", legend = "Normal curve", col = "red", lwd = 2)

dev.off()
```

```
## pdf
##   2
```

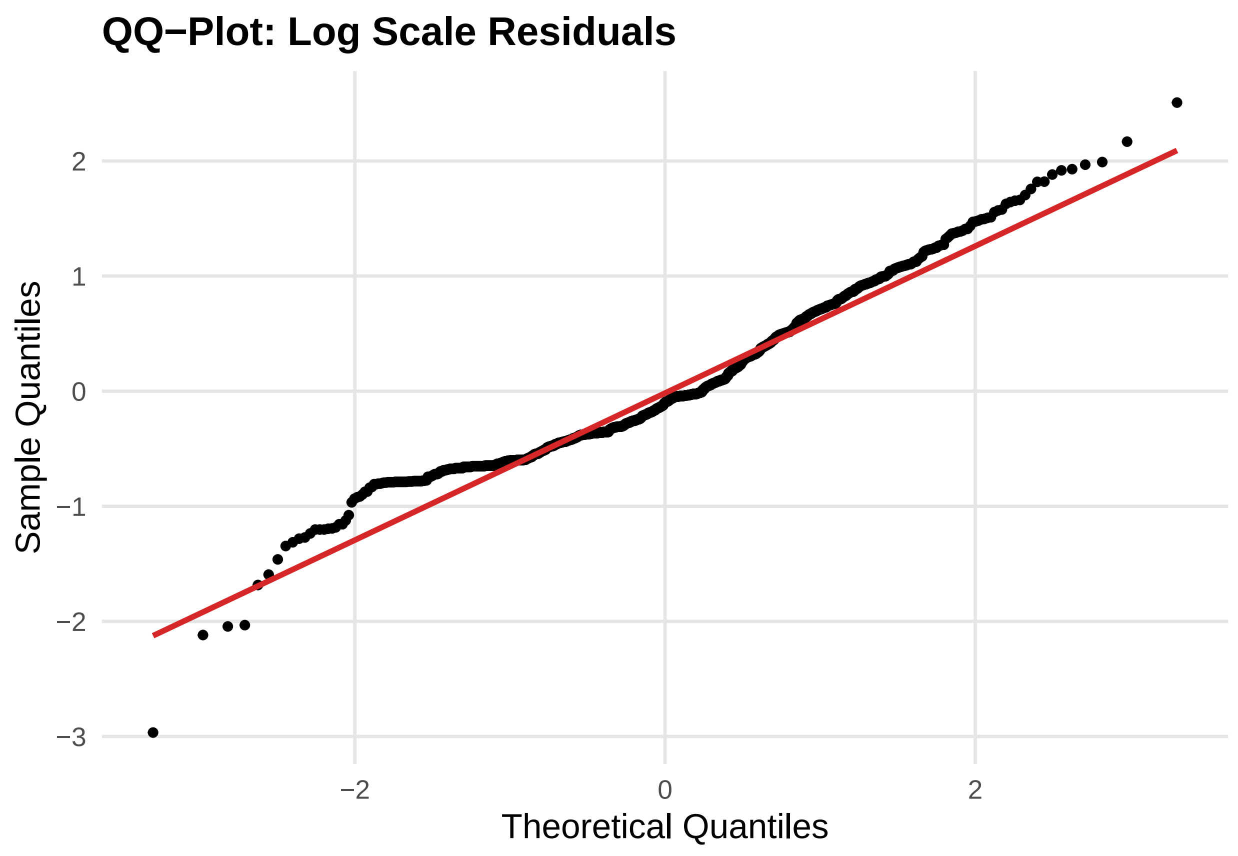
```r
cat(sprintf("Shapiro-Wilk test: W=%.6f, p=%.4e\n", w_stat_log, p_value_log))
```

```
## Shapiro-Wilk test: W=0.962001, p=8.4065e-16
```

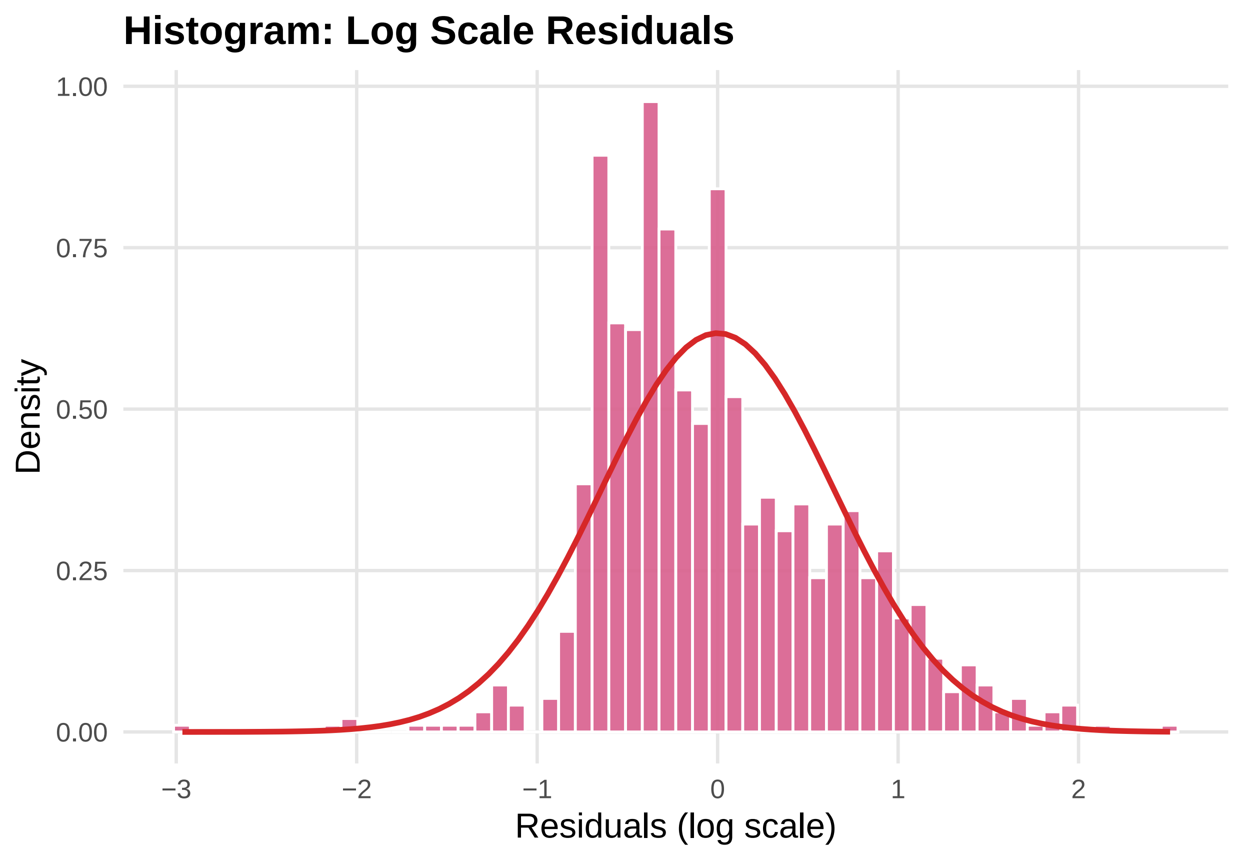
```r
cat(sprintf("Normality: %s at  =%.2f\n",
            ifelse(p_value_log < ALPHA, "rejected", "not rejected"), ALPHA))
```

```
## Normality: rejected at  =0.05
```

```r
log_resid_df = data.frame(residuals_log = residuals_log)

qq_log_plot = ggplot(log_resid_df, aes(sample = residuals_log)) +
  stat_qq(size = 1.2, color = "black") +
  stat_qq_line(color = "#d62728", linewidth = 1) +
  labs(
    title = "QQ-Plot: Log Scale Residuals",
    x = "Theoretical Quantiles",
    y = "Sample Quantiles"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", size = 15),
    panel.grid.minor = element_blank(),
    panel.grid.major = element_line(color = "gray90")
  )

hist_log_plot = ggplot(log_resid_df, aes(x = residuals_log)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 60,
                 fill = "#d95f8d",
                 color = "white",
                 alpha = 0.9) +
  stat_function(fun = dnorm,
                args = list(mean = mean(residuals_log), sd = sd(residuals_log)),
                color = "#d62728",
                linewidth = 1) +
  labs(
    title = "Histogram: Log Scale Residuals",
    x = "Residuals (log scale)",
    y = "Density"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", size = 15),
    panel.grid.minor = element_blank(),
    panel.grid.major = element_line(color = "gray90")
  )

print(qq_log_plot)
```



```r
print(hist_log_plot)
```



```r
ggsave(
  file.path(FIGURES_DIR, "qq_plot_log_scale_gg.png"),
  plot = qq_log_plot,
  width = 8,
  height = 6,
  dpi = DPI,
  bg = "white"
)

ggsave(
  file.path(FIGURES_DIR, "histogram_log_scale_gg.png"),
  plot = hist_log_plot,
  width = 8,
  height = 6,
  dpi = DPI,
  bg = "white"
)

# 14. variance homogeneity comparison
interpret_ratio = function(ratio) {
  if (ratio < 3.0) {
    return("acceptable (< 3.0)")
  } else if (ratio < 4.0) {
    return("borderline (3.0-4.0)")
  } else {
    return("problematic (> 4.0)")
  }
}

comparison_data = data.frame(
  Scale = c("Raw (review_time_days)", "Log (log_review_time_days)"),
  Variance_Ratio = c(variance_ratio_raw, variance_ratio_log),
  Interpretation = c(interpret_ratio(variance_ratio_raw), interpret_ratio(variance_ratio_log)),
  Levene_Statistic = c(levene_raw_stat, levene_log_stat),
  Levene_P_Value = c(levene_raw_p, levene_log_p),
  Levene_Result = c(
    ifelse(levene_raw_p < ALPHA, "reject homogeneity", "accept homogeneity"),
    ifelse(levene_log_p < ALPHA, "reject homogeneity", "accept homogeneity")
  )
)

print(comparison_data)
```

```
##                        Scale Variance_Ratio       Interpretation Levene_Statistic Levene_P_Value
## 1     Raw (review_time_days)       3.520534 borderline (3.0-4.0)        19.567888   2.416455e-12
## 2 Log (log_review_time_days)       1.715330   acceptable (< 3.0)         8.667458   1.105743e-05
##        Levene_Result
## 1 reject homogeneity
## 2 reject homogeneity
```

```r
cat("\nInterpretation:\n")
```

```
##
## Interpretation:
```

```r
cat(sprintf("Raw scale: ratio=%.2f (%s)\n", variance_ratio_raw, interpret_ratio(variance_ratio_raw)))
```

```
## Raw scale: ratio=3.52 (borderline (3.0-4.0))
```

```r
cat(sprintf("Log scale: ratio=%.2f (%s)\n", variance_ratio_log, interpret_ratio(variance_ratio_log)))
```

```
## Log scale: ratio=1.72 (acceptable (< 3.0))
```

```r
cat("\nRecommendation: log scale shows better variance homogeneity\n")
```

```
##
## Recommendation: log scale shows better variance homogeneity
```

```r
# saving results
save_csv(comparison_data, file.path(RESULTS_DIR, "assumption_testing_summary.csv"))
```

```
## saving results to: /Users/abdulbasir/Downloads/Experimental AI/fda-oncology-approval-analysis/result
```

```
# 15. comprehensive normality report
cat("Normality testing - raw scale:\n")
```

```
## Normality testing - raw scale:
```

```
cat(sprintf("  W-statistic: %.6f\n", w_stat_raw))
```

```
##   W-statistic: 0.784619
```

```
cat(sprintf("  P-value: %s\n", format_p_value(p_value_raw)))
```

```
##   P-value: p<0.001
```

```
cat(sprintf("  Decision: normality %s at  =%.2f\n",
            ifelse(p_value_raw < ALPHA, "rejected", "not rejected"), ALPHA))
```

```
##   Decision: normality rejected at  =0.05
```

```
cat("\nNormality testing - log scale:\n")
```

```
##
## Normality testing - log scale:
```

```
cat(sprintf("  W-statistic: %.6f\n", w_stat_log))
```

```
##   W-statistic: 0.962001
```

```
cat(sprintf("  P-value: %s\n", format_p_value(p_value_log)))
```

```
##   P-value: p<0.001
```

```
cat(sprintf("  Decision: normality %s at  =%.2f\n",
            ifelse(p_value_log < ALPHA, "rejected", "not rejected"), ALPHA))
```

```
##   Decision: normality rejected at  =0.05
```

```
cat("\nRaw vs log scale comparison:\n")
```

```
##
## Raw vs log scale comparison:
```

```
w_diff = w_stat_log - w_stat_raw
cat(sprintf("  W-statistic improvement: %.6f (log - raw)\n", w_diff))
```

```
##   W-statistic improvement: 0.177383 (log - raw)
```

```
cat("  Log scale has substantially better normality\n")
```

```
##   Log scale has substantially better normality
```

```
cat("  Recommendation: use log scale for parametric analyses\n")
```

```
##   Recommendation: use log scale for parametric analyses
```

```
cat("  Alternative: permutation tests recommended for inference validation\n")
```

```
##   Alternative: permutation tests recommended for inference validation
```

```
# 16. saving analysis-ready dataset for phases 6-8
if ("regulatory_era" %in% names(analysis_data)) {
  analysis_data = analysis_data %>% select(-regulatory_era)
}

analysis_ready_file = file.path(RESULTS_DIR, "analysis_ready_dataset.csv")
save_csv(analysis_data, analysis_ready_file)
```

```
## saving results to: /Users/abdulbasir/Downloads/Experimental AI/fda-oncology-approval-analysis/results
```

```
cat(sprintf("\nSaved analysis-ready dataset to: %s\n", analysis_ready_file))
```

```
##
## Saved analysis-ready dataset to: /Users/abdulbasir/Downloads/Experimental AI/fda-oncology-approval-a
```

```
cat("\nPhase 5 complete\n")
```

```
##
## Phase 5 complete
```