Chapter 4: inference

We start again by loading the data and filtering it so that we restrict it to properties that host at most 6 people. Also, as in Chapter 3, we again create a variable `review_scores_rating_standardized` with the standardized review score.

```
# Load the datasets from the RData file
load("../dataCreated/listings_clean.RData")

listings_clean_filtered <- listings_clean %>%
  filter(accommodates <= 6)

# Standardize review_scores_rating
listings_clean_filtered <- listings_clean_filtered %>%
  mutate(review_scores_rating_standardized =
           (review_scores_rating - mean(review_scores_rating, na.rm = TRUE)) /
           sd(review_scores_rating, na.rm = TRUE))
```

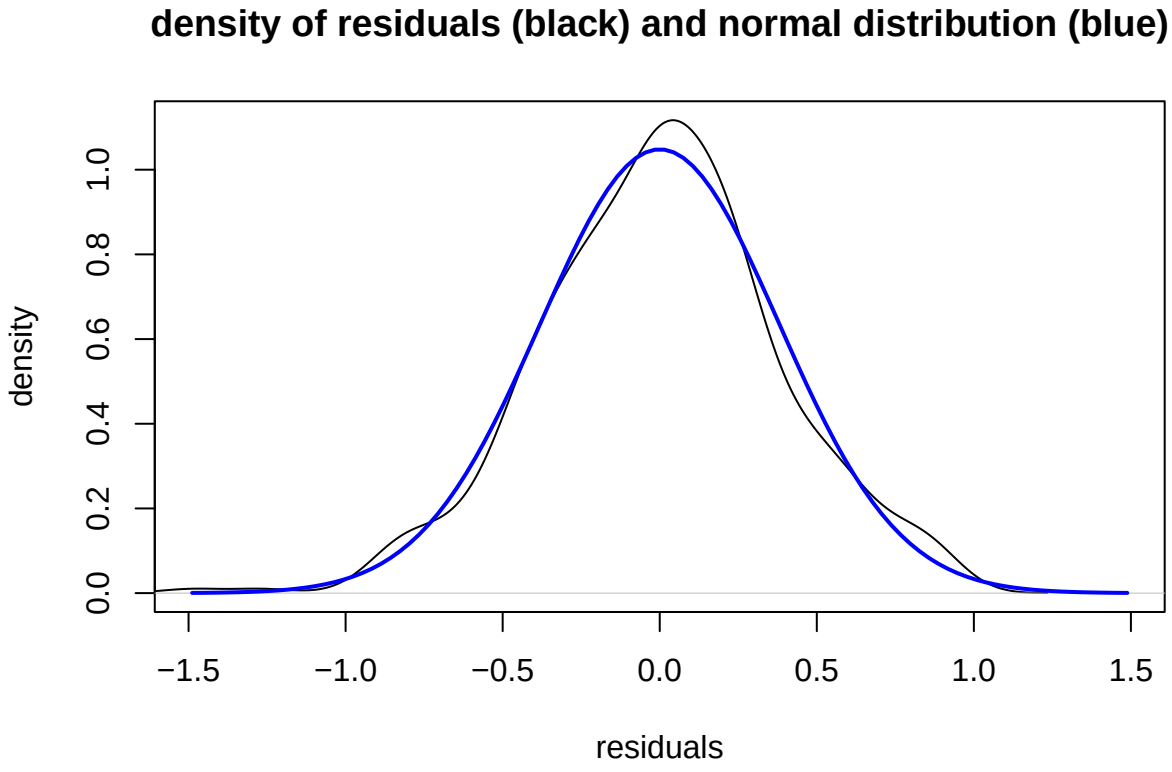
Our point of departure for Chapter 4 is the richer model from Chapter 3 where we regress the log price on `review_scores_rating`, `accommodates`, and 4 neighborhood characteristics.

```
estimatesFullModel <- lm(log(price) ~ review_scores_rating_standardized + accommodates + Centrality + Qu
summary(estimatesFullModel)
```

```
##
## Call:
## lm(formula = log(price) ~ review_scores_rating_standardized +
##     accommodates + Centrality + Quietness + Coolness + Fanciness,
##     data = listings_clean_filtered)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48845 -0.24997  0.01026  0.23530  0.94151
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       3.10015    0.49275   6.292    8e-10 ***
## review_scores_rating_standardized  0.05000    0.01887   2.650  0.00837 **
## accommodates                       0.21641    0.01480  14.625  < 2e-16 ***
## Centrality                         0.08591    0.04406   1.950  0.05186 .
## Quietness                          0.12896    0.04451   2.897  0.00397 **
## Coolness                           0.09606    0.07834   1.226  0.22079
## Fanciness                         -0.14765    0.05382  -2.744  0.00634 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3834 on 414 degrees of freedom
## Multiple R-squared:  0.3494, Adjusted R-squared:   0.34
## F-statistic: 37.05 on 6 and 414 DF,  p-value: < 2.2e-16
```

In Chapter 4, we assume normality of the error term. To check whether this is (approximately) appropriate, we first construct a plot of the density of the residuals and superimpose a normal distribution.

```
# Calculate residuals
residualsFullModel <- residuals(estimatesFullModel)

# Find the maximum absolute value of the residuals
max_abs_residual <- max(abs(residualsFullModel))

# Plot the density of the residuals, setting the x-axis limits symmetrically around 0
plot(density(residualsFullModel),
     main = "density of residuals (black) and normal distribution (blue)",
     xlab = "residuals",
     ylab = "density",
     xlim = c(-max_abs_residual, max_abs_residual))  # Symmetric x-axis limits

# Superimpose a normal distribution
x_vals <- seq(-max_abs_residual, max_abs_residual, length = 100)
y_vals <- dnorm(x_vals, mean = mean(residualsFullModel), sd = sd(residualsFullModel))
lines(x_vals, y_vals, col = "blue", lwd = 2)
```



That looks promising. We also perform two formal tests for normality. First, the so-called Shapiro-Wilk test.

```
# Perform the Shapiro-Wilk test for normality on the residuals
shapiro_test <- shapiro.test(residualsFullModel)

# Display the test results
shapiro_test
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  residualsFullModel
## W = 0.99381, p-value = 0.08395
```

We also implement the Jarque-Bera test for normality.

```
library(tseries)
jarque.bera.test(residualsFullModel)
```

```
##
##  Jarque Bera Test
##
## data:  residualsFullModel
## X-squared = 4.2494, df = 2, p-value = 0.1195
```

For both, we cannot formally reject the null hypothesis of normality at the 5% level of significance.[1]

We now perform a t-test for the hypothesis that the coefficient of `Centrality` is zero, with the alternative that it is not.

```
model_summary <- summary(estimatesFullModel)

# Extract the coefficient estimate and standard error for Centrality
coef_centrality <- model_summary$coefficients["Centrality", "Estimate"]
se_centrality <- model_summary$coefficients["Centrality", "Std. Error"]

# Calculate the t-statistic
t_stat_centrality <- coef_centrality / se_centrality

# Extract the number of observations and coefficients
n_obs <- nobs(estimatesFullModel)  # Number of observations
n_coeffs <- length(coef(estimatesFullModel))  # Number of coefficients (including intercept)

# Degrees of freedom
df <- n_obs - n_coeffs

# Explanation
cat("The degrees of freedom are", df, "because there are", n_obs, "observations and", n_coeffs, "coeffi
```

```
## The degrees of freedom are 414 because there are 421 observations and 7 coefficients.
```

```
# Calculate the p-value manually using the t-distribution for a two-sided test
p_value_centrality_manual <- 2 * (1 - pt(abs(t_stat_centrality), df))

# Display the t-statistic, degrees of freedom, and manually calculated p-value for Centrality
t_stat_centrality
```

```
## [1] 1.949922
```

```
p_value_centrality_manual
```

```
## [1] 0.05186001
```

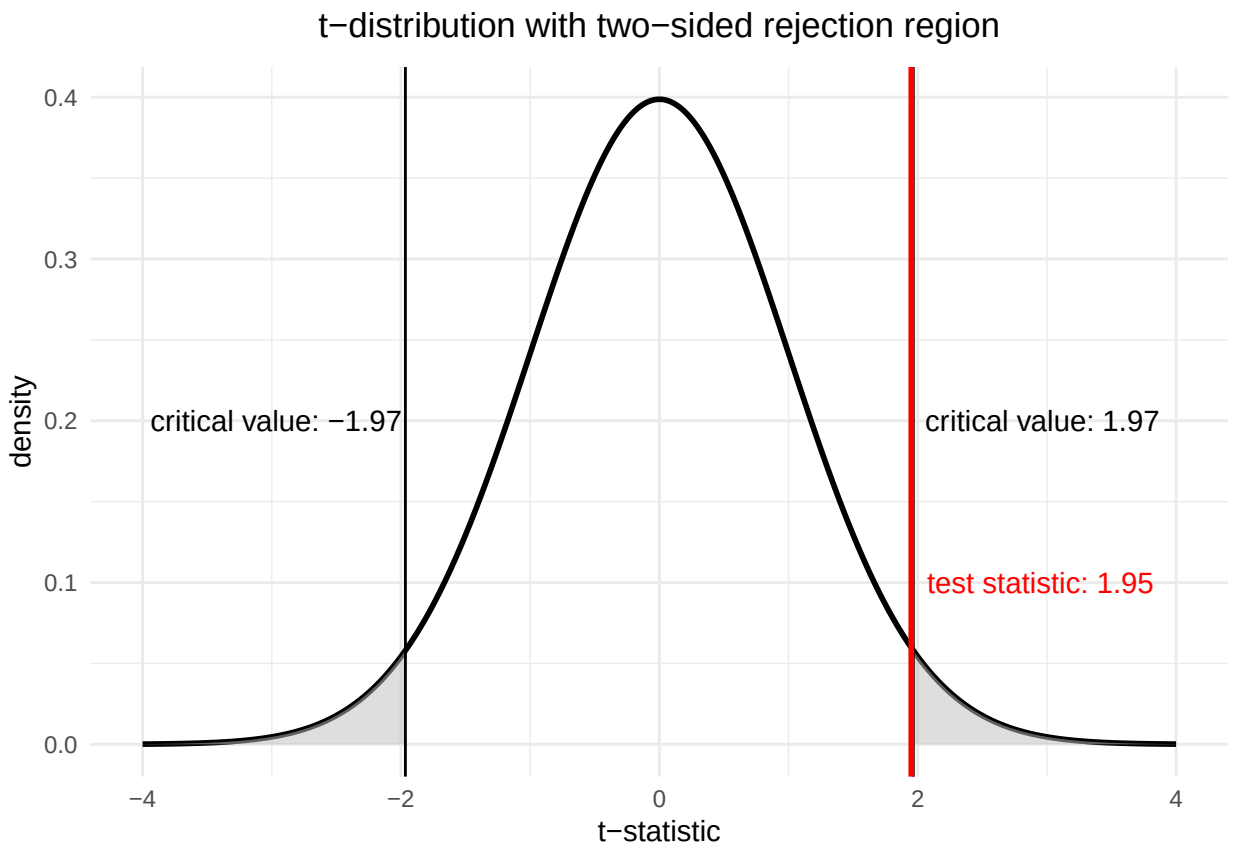
```
# Critical value for the two-sided test (5% significance level)
alpha_two_sided <- 0.05
critical_value_two_sided <- qt(1 - alpha_two_sided / 2, df)
cat("The critical value for the two-sided test at the 5% level is", critical_value_two_sided, "\n")
```

[1] Later, in Chapter 5, we will actually learn that we do not need that if we instead make use of asymptotic results.

```
## The critical value for the two-sided test at the 5% level is 1.965711
```

We can also draw the distribution of the test statistic under the null, the value we get, and the rejection region.

```r
# Example test statistic from previous model (replace with your actual value)
test_stat <- t_stat_centrality  # Replace with the calculated t-statistic

# Generate values for the t-distribution
x_vals <- seq(-4, 4, length = 1000)
y_vals <- dt(x_vals, df)

# Calculate the critical values for the two-sided test
alpha_two_sided <- 0.05
critical_value <- qt(1 - alpha_two_sided / 2, df)

# Create a data frame for plotting
data <- data.frame(x = x_vals, y = y_vals)

# Plot the t-distribution with rejection regions and test statistic
ggplot(data, aes(x = x, y = y)) +
  geom_line(color = "black", linewidth = 1) +  # density line in black
  geom_area(data = subset(data, x < -critical_value), aes(y = y), fill = "gray", alpha = 0.5) +  # left
  geom_area(data = subset(data, x > critical_value), aes(y = y), fill = "gray", alpha = 0.5) +  # right
  labs(title = expression("t-distribution with two-sided rejection region"),  # Math 't'
       x = "t-statistic", y = "density") +
  geom_vline(xintercept = -critical_value, linetype = "solid", color = "black", linewidth = 0.5) +  # l
  geom_vline(xintercept = critical_value, linetype = "solid", color = "black", linewidth = 0.5) +  # ri
  geom_vline(xintercept = test_stat, linetype = "solid", color = "red", linewidth = 1) +  # test statis
  # Adjusting the position of the annotations:
  annotate("text", x = -critical_value - 1.0, y = 0.2, label = paste("critical value:", round(-critical
  annotate("text", x = critical_value + 1.0, y = 0.2, label = paste("critical value:", round(critical_va
  annotate("text", x = test_stat + 1.0, y = 0.1, label = paste("test statistic:", round(test_stat, 2)),
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  # center the title
```



Formally, we do not reject the null hypothesis at the 5% level because the red line is in-between the two black lines.

Next, we do a one-sided t-test for the alternative hypothesis that the coefficient is positive

```
# Calculate the one-sided p-value
p_value_centrality_one_sided <- 1 - pt(t_stat_centrality, df)

# Display the t-statistic and one-sided p-value for Centrality (positive alternative)
t_stat_centrality
```

```
## [1] 1.949922
```

```
p_value_centrality_one_sided
```

```
## [1] 0.02593001
```

```
# Explanation
cat("For the alternative hypothesis that the coefficient on Centrality is positive, the one-sided p-valu
```

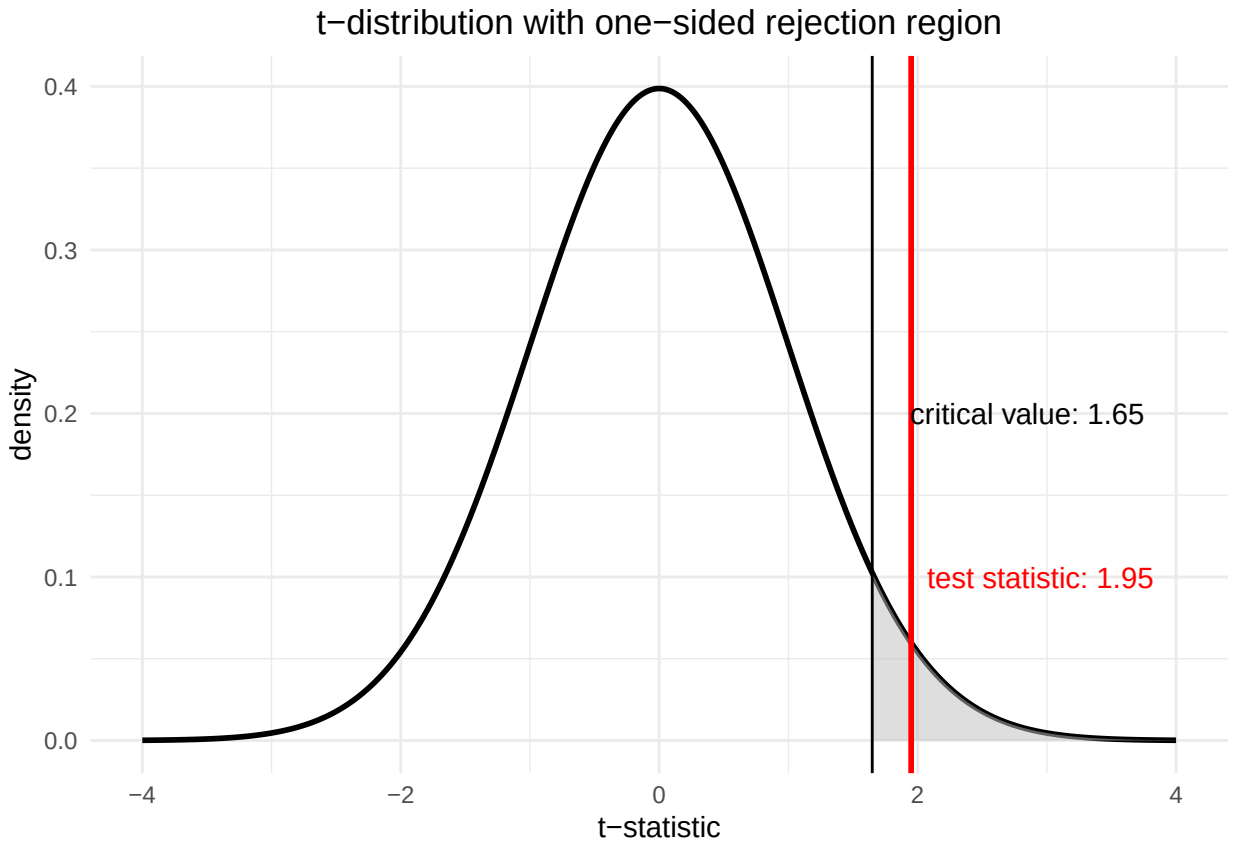
```
## For the alternative hypothesis that the coefficient on Centrality is positive, the one-sided p-value
```

```
# Critical value for the one-sided test (5% significance level)
alpha_one_sided <- 0.05
critical_value_one_sided <- qt(1 - alpha_one_sided, df)
cat("The critical value for the one-sided test at the 5% level is", critical_value_one_sided, "\n")
```

```
## The critical value for the one-sided test at the 5% level is 1.648543
```

Here is the corresponding plot.

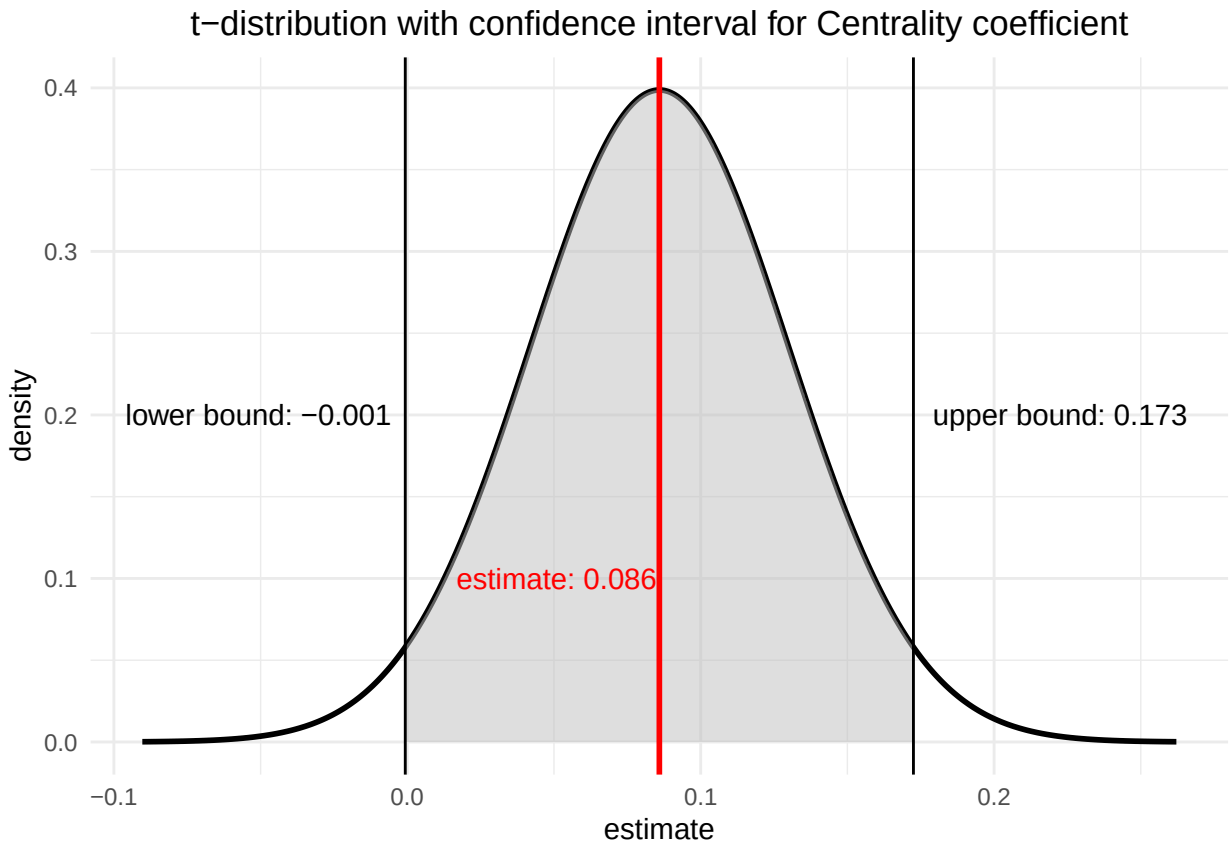
```r
# Example test statistic from previous model (replace with your actual value)
test_stat <- t_stat_centrality  # Replace with the calculated t-statistic

# Generate values for the t-distribution
x_vals <- seq(-4, 4, length = 1000)
y_vals <- dt(x_vals, df)

# Calculate the critical value for the one-sided test
alpha_one_sided <- 0.05
critical_value_one_sided <- qt(1 - alpha_one_sided, df)

# Create a data frame for plotting
data <- data.frame(x = x_vals, y = y_vals)

# Plot the t-distribution with one-sided rejection region and test statistic
ggplot(data, aes(x = x, y = y)) +
  geom_line(color = "black", linewidth = 1) +  # density line in black
  geom_area(data = subset(data, x > critical_value_one_sided), aes(y = y), fill = "gray", alpha = 0.5) +
  labs(title = "t-distribution with one-sided rejection region",
       x = "t-statistic", y = "density") +
  geom_vline(xintercept = critical_value_one_sided, linetype = "solid", color = "black", linewidth = 0.5
  geom_vline(xintercept = test_stat, linetype = "solid", color = "red", linewidth = 1) +  # test statis
  annotate("text", x = critical_value_one_sided + 1.2, y = 0.2, label = paste("critical value:", round(
  annotate("text", x = test_stat + 1.0, y = 0.1, label = paste("test statistic:", round(test_stat, 2)),
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  # center the title
```



Now we reject the null at the 5% level, because the red line is to the right of the critical value.

We can also construct a plot for the 95% confidence interval for the coefficient on 'Centrality'.

```
# Extract the coefficient estimate and standard error for Centrality from your regression results
estimate <- coef(estimatesFullModel)["Centrality"]  # Coefficient estimate for Centrality
se <- summary(estimatesFullModel)$coefficients["Centrality", "Std. Error"]  # Standard error for Centra
df <- df.residual(estimatesFullModel)  # Degrees of freedom for the model

# Generate t-distribution values
x_vals <- seq(-4, 4, length = 1000)  # Standardized t-values
y_vals <- dt(x_vals, df)

# Scale by the standard error and shift by the estimate
x_shifted <- x_vals * se + estimate  # Shifted x-values for the t-distribution

# Calculate the 95% confidence interval
alpha <- 0.05
critical_value_ci <- qt(1 - alpha / 2, df)  # Critical value for the 95% confidence level
lower_bound <- estimate - critical_value_ci * se  # Lower bound of CI
upper_bound <- estimate + critical_value_ci * se  # Upper bound of CI

# Create a data frame for plotting
data <- data.frame(x = x_shifted, y = y_vals)

# Plot the t-distribution with confidence interval and test statistic
ggplot(data, aes(x = x, y = y)) +
  geom_line(color = "black", linewidth = 1) +  # density line in black
  geom_area(data = subset(data, x > lower_bound & x < upper_bound), aes(y = y), fill = "gray", alpha = (
  labs(title = "t-distribution with confidence interval for Centrality coefficient",
       x = "estimate", y = "density") +
  geom_vline(xintercept = lower_bound, linetype = "solid", color = "black", linewidth = 0.5) +  # lower
  geom_vline(xintercept = upper_bound, linetype = "solid", color = "black", linewidth = 0.5) +  # upper
  geom_vline(xintercept = estimate, linetype = "solid", color = "red", linewidth = 1) +  # test statist
  annotate("text", x = lower_bound - 0.05, y = 0.2, label = paste("lower bound:", format(round(lower_bou
  annotate("text", x = upper_bound + 0.05, y = 0.2, label = paste("upper bound:", format(round(upper_bou
  annotate("text", x = estimate - 0.035, y = 0.1, label = paste("estimate:", format(round(estimate, 3),
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  # center the title
```



We see that 0 lies inside the confidence interval (which is directly related to us not rejecting the null that the coefficient is zero at the 5% level).

Finally, we conduct an F-test for the joint significance of the coefficients on `Centrality`, `Quietness`, `Coolness`, and `Fanciness`.

```
# Fit the reduced model excluding Centrality, Quietness, Coolness, and Fanciness
reduced_model <- lm(log(price) ~ review_scores_rating_standardized + accommodates, data = listings_clea

# Fit the full model (already done)
full_model <- estimatesFullModel

# Extract the sum of squared residuals (SSR) from both models
SSR_reduced <- sum(residuals(reduced_model)^2)
SSR_full <- sum(residuals(full_model)^2)

# Extract the number of observations (n) and number of coefficients in full and reduced models
n <- nobs(full_model)
k_full <- length(coef(full_model))  # Number of parameters in the full model
k_reduced <- length(coef(reduced_model))  # Number of parameters in the reduced model

# Calculate the number of restrictions (q), i.e., the number of coefficients tested jointly
q <- k_full - k_reduced

# Calculate the F-statistic manually
F_stat <- ((SSR_reduced - SSR_full) / q) / (SSR_full / (n - k_full))

# Calculate the p-value for the F-statistic
```

```r
p_value <- 1 - pf(F_stat, q, n - k_full)

# Display the F-statistic and p-value
F_stat
```

```
## [1] 2.935027
```

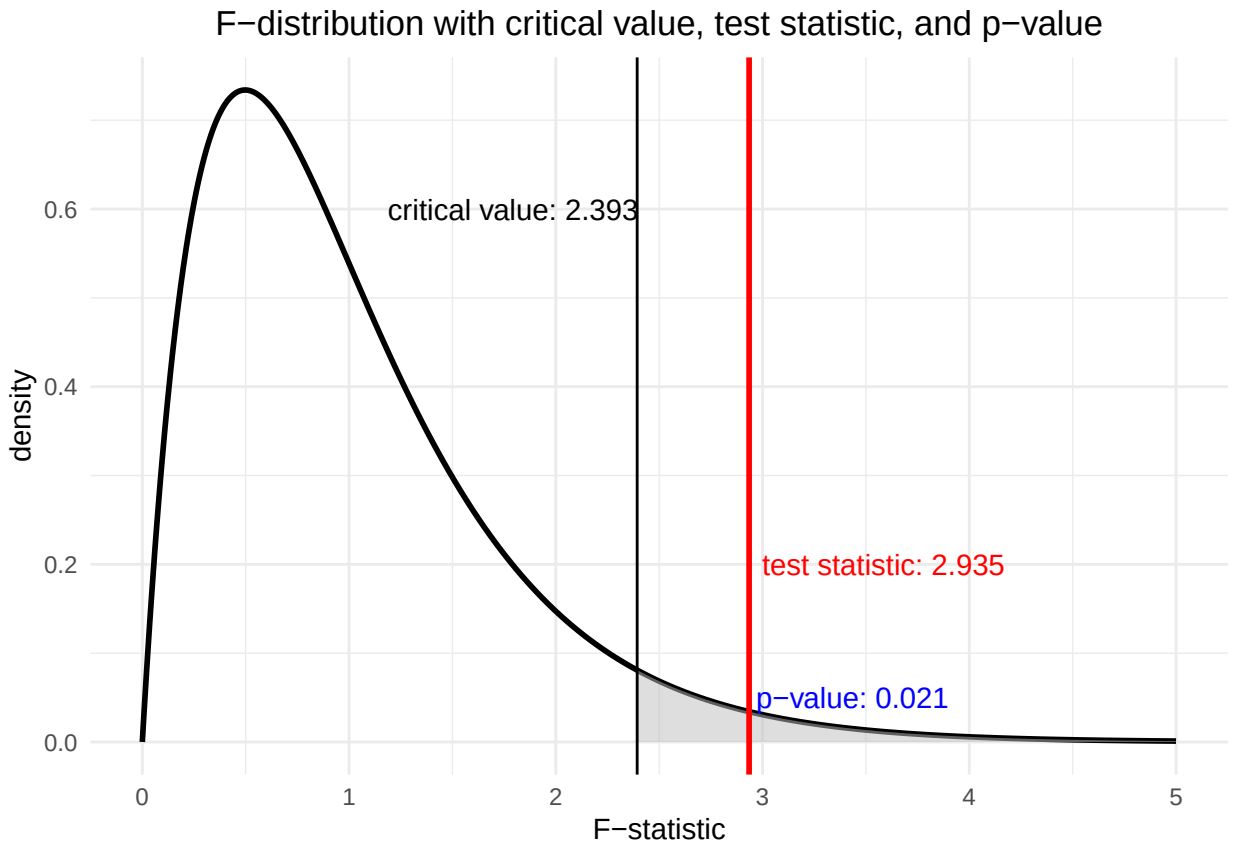
```r
p_value
```

```
## [1] 0.02054708
```

This shows that we reject the null hypothesis that all coefficients are zero at the 5% level.

We can also make a plot of the F-distribution to visualize the test.

```r
# Manually calculate the F-statistic and p-value (from previous steps)
# Assuming you've already calculated the F-statistic (F_stat) and degrees of freedom (q, n - k_full)
F_stat <- F_stat  # Use the calculated F-statistic
df1 <- q  # Numerator degrees of freedom (number of restrictions)
df2 <- n - k_full  # Denominator degrees of freedom (residuals from full model)

# Generate F-distribution values
x_vals <- seq(0, 5, length = 1000)
y_vals <- df(x_vals, df1, df2)

# Calculate the critical value for the 5% significance level
alpha <- 0.05
critical_value_f <- qf(1 - alpha, df1, df2)

# Calculate the p-value for the F-statistic
p_value_f <- 1 - pf(F_stat, df1, df2)

# Create a data frame for plotting
data <- data.frame(x = x_vals, y = y_vals)

# Plot the F-distribution with the critical value, test statistic, and p-value
ggplot(data, aes(x = x, y = y)) +
  geom_line(color = "black", linewidth = 1) +  # density line in black
  geom_area(data = subset(data, x > critical_value_f), aes(y = y), fill = "gray", alpha = 0.5) +  # rej
  labs(title = "F-distribution with critical value, test statistic, and p-value",
       x = "F-statistic", y = "density") +
  geom_vline(xintercept = critical_value_f, linetype = "solid", color = "black", linewidth = 0.5) +  #
  geom_vline(xintercept = F_stat, linetype = "solid", color = "red", linewidth = 1) +  # test statistic
  annotate("text", x = critical_value_f - 0.6, y = 0.6, label = paste("critical value:", format(round(c
  annotate("text", x = F_stat + 0.65, y = 0.2, label = paste("test statistic:", format(round(F_stat, 3)
  annotate("text", x = F_stat + 0.5, y = 0.05, label = paste("p-value:", format(round(p_value_f, 3), nsm
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  # center the title
```



The p-value in this plot corresponds to the gray area to the right of the test statistic line (in red). Since the test statistic is in the rejection region (to the right of the critical value), we reject the null hypothesis that all coefficients are zero at the 5% level.