Beyond MLR Lab 5: multi-level analysis of cross-sectional observational data

Preliminary setup

In this lab, we will use the R packages `ggplot2`, `dplyr`, `emmeans`, `lmerTest`. You can use the script below to automatically install and load them using the `pacman` package.

```r
# Check whether pacman is available and install if needed
options(repos = c(CRAN = "https://cloud.r-project.org"))
if (!requireNamespace("pacman", quietly = TRUE)) install.packages("pacman")

# Use pacman to install (if needed) and load the required packages
pacman::p_load(dplyr, ggplot2, emmeans, lmerTest)
```

Chicago AirBnB Data

The Chicago Airbnb dataset was compiled by Trinh and Ameri as part of a course project at St. Olaf College and is included with the book Beyond Multiple Linear Regression by Roback and Legler (2021). It contains information on 1,561 Airbnb listings in Chicago, including details such as nightly price, overall satisfaction rating, number of reviews, and various other listing characteristics. Additionally, the dataset includes neighborhood-level information, such as ratings for walkability, access to public transit, and bikeability, providing valuable context for the listings' locations.

The dataset is available on Brightspace under the name `airbnb.csv`. It contains the following variables:

- price: the nightly price of the listing (in USD)
- overall_satisfaction: the listing's average rating, on a scale from 1 to 5
- reviews: number of user reviews the listing has
- room_type: the type of listing (eg: Shared room)

- accommodates: number of guests the listing accommodates
- bedrooms: the number of bedrooms the listing has
- minstay: the minimum number of nights to stay in the listing
- neighborhood: the neighborhood in which the listing is located
- WalkScore: the neighborhood's rating for walkability (0 - 100)
- TransitScore: the neighborhood's rating for access to public transit (0 - 100)
- BikeScore: the neighborhood's rating for bikeability (0 - 100)

Exploratory Data Analysis

Let's load the data and use the `str()` function to inspect the structure of the dataset.

```r
# Load the Chicago AirBnB data. Note that the call below assumes the csv file is placed in tl
chicago_airbnb <- read.csv("data/airbnb.csv")

# Convert all character variables into factors
chicago_airbnb <- chicago_airbnb |>
  mutate(across(where(is.character), as.factor))

# Inspect the structure of the dataset
str(chicago_airbnb)
```

This initial inspection makes clear that the Airbnb dataset has a hierarchical structure: 1,561 individual listings are nested within 43 neighborhoods. While the dataset also includes a `district` variable that groups neighborhoods into 9 broader districts, for educational purposes we will treat this as a two-level hierarchical structure with listings (level 1) nested within neighborhoods (level 2), ignoring the higher-level clustering of neighborhoods into districts.

Grouping structure

To get a better feeling for the grouping structure, we create a summary table showing the number of listings within each neighborhood and visualize it with a bar chart:

```r
# Create a summary table showing listings per neighborhood
neighborhood_summary <- chicago_airbnb |>
  group_by(neighborhood) |>
  summarise(num_listings = n(), .groups = "drop") |>
  arrange(desc(num_listings))

# Create a bar chart showing listings per neighborhood (top 20)
neighborhood_summary |>
```

```r
  top_n(20, num_listings) |>
  ggplot(aes(x = reorder(neighborhood, num_listings), y = num_listings)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  coord_flip() +
  labs(title = "Number of Listings per Neighborhood (Top 20)",
       x = "Neighborhood",
       y = "Number of Listings") +
  theme_minimal()
```

Explanation:

- `group_by(neighborhood)`: groups the data by neighborhood so that we can count listings within each neighborhood.
- `n()`: counts the total number of listings within each neighborhood.
- `arrange(desc(num_listings))`: sorts the table by the number of listings in descending order.
- `top_n(20, num_listings)`: selects the 20 neighborhoods with the most listings for visualization.
- `reorder(neighborhood, num_listings)`: reorders the neighborhood factor levels based on the values of `num_listings`, arranging neighborhoods from lowest to highest number of listings for easier interpretation.

Question

Which neighborhoods have the most Airbnb listings? Is there substantial variation in the number of listings across neighborhoods?

Outcome variable

Next, we create a histogram to visualize the distribution of the outcome variable `price`:

```r
# Create a histogram of price
ggplot(chicago_airbnb, aes(x = price)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Listing Prices",
       x = "Nightly Price (USD)",
       y = "Frequency") +
  theme_minimal()
```

While normality of the outcome is not strictly required for a mixed-effects model, transforming a right-skewed variable like price can help stabilize variance and linearize relationships. We therefore create a new variable, `log_price`, to store the log-transformed prices:

```r
# Log-transform the price variable
chicago_airbnb <- chicago_airbnb |>
  mutate(log_price = log(price))
```

Variance components

We also want to get a sense of the variability in the listing prices within and between neighborhoods. For this, we are going to randomly select 10 neighborhoods and create a scatter plot with the log-transformed price on the y-axis and the neighborhood on the x-axis:

```r
# Set seed for reproducibility
set.seed(123)

# Randomly select 10 unique neighborhoods
random_neighborhoods <- sample(unique(chicago_airbnb$neighborhood), size = 10)

# Filter the data to include only the randomly selected neighborhoods
random_neighborhood_data <- chicago_airbnb |>
  filter(neighborhood %in% random_neighborhoods)

# Calculate the average log_price for each neighborhood
neighborhood_avg_price <- random_neighborhood_data |>
  group_by(neighborhood) |>
  summarise(avg_log_price = mean(log_price, na.rm = TRUE), .groups = "drop")

# Calculate the overall grand mean across all neighborhoods
grand_mean <- mean(chicago_airbnb$log_price, na.rm = TRUE)

# Create scatter plot with jittered observations
ggplot(random_neighborhood_data, aes(x = reorder(neighborhood, log_price, FUN = mean), y = l
  geom_hline(yintercept = grand_mean, linetype = "solid", color = "black", linewidth = 1) +
  geom_jitter(alpha = 0.3, width = 0.2, size = 1, color = "gray60") +
  geom_point(data = neighborhood_avg_price, aes(x = neighborhood, y = avg_log_price),
             size = 4, shape = 18, color = "red") +
  labs(title = "Variation in listing prices: within and between neighborhoods",
       subtitle = "Points = individual listings; Diamonds = neighborhood means; Solid line =
       x = "Neighborhood (ordered by mean log price)",
       y = "Log-transformed price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Question

What does this plot suggest about the variability in listing prices within and between neighborhoods?

Covariate effects

Before fitting models with covariates, it is useful to explore the relationships between potential predictors and the outcome variable. Let us examine how some subject-level characteristics relate to listing prices.

First, we will look at the relationship between the number of reviews and log-transformed price:

```r
# Scatterplot with smooth line for reviews vs log_price
ggplot(chicago_airbnb, aes(x = reviews, y = log_price)) +
  geom_point(color = "gray60") +
  geom_smooth(method = "lm", se = FALSE, color = "steelblue", linewidth = 1.5) +
  labs(title = "Relationship between number of reviews and listing price",
       x = "Number of reviews",
       y = "Log-transformed price") +
  theme_minimal()
```

Next, let us examine how listing prices vary across different room types:

```r
# Boxplot for room_type vs log_price
ggplot(chicago_airbnb, aes(x = room_type, y = log_price, fill = room_type)) +
  geom_boxplot(alpha = 0.7) +
  labs(title = "Listing prices by room type",
       x = "Room type",
       y = "Log-transformed price") +
  theme_minimal() +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1))
```

Coding exercise

Create similar visualizations to explore the relationships between:

1. Overall satisfaction rating and log-transformed price (scatterplot with smooth line)
2. Number of bedrooms and log-transformed price (boxplot)

What patterns do you observe? Do higher satisfaction ratings and more bedrooms tend to be associated with higher or lower prices?

Random intercept model

To model the variability in listing prices within and between neighborhoods, we start by fitting a random intercept model to account for the nesting of listings within neighborhoods. This model is specified as follows:

```r
# Fit the random intercept model
random_intercept_model <- lmer(log_price ~ 1 + (1 | neighborhood), data = chicago_airbnb)
summary(random_intercept_model)
```

Question

Based on the estimated variance components, what can you conclude about the relative contributions of within-neighborhood and between-neighborhood variability to the total variability in listing prices?

Extending the random intercept model with subject-level variables

As a second step, we extend the random intercept model with subject-level variables. To facilitate the interpretation of the model coefficients, we start by setting the coding scheme for categorical variables to effects coding. We also center the numerical variables by subtracting the mean value from each observation. This step is important to reduce multicollinearity, improve numerical stability, and make the intercept more interpretable.

```r
# Use effects coding for the categorical variables
options(contrasts = c("contr.sum", "contr.poly"))

# Center the continuous subject-level variables
chicago_airbnb <- chicago_airbnb |>
  mutate(overall_satisfaction_c = scale(overall_satisfaction, scale = FALSE),
         accommodates_c = scale(accommodates, scale = FALSE),
         bedrooms_c = scale(bedrooms, scale = FALSE),
         minstay_c = scale(minstay, scale = FALSE))
```

Next, we fit the random intercept model with the subject-level variables included:

```r
# Fit the random intercept model with subject-level variables
random_intercept_model_L1variables <- lmer(log_price ~ overall_satisfaction_c + room_type + a
summary(random_intercept_model_L1variables)

# Display the coding scheme for the room_type variable
contrasts(chicago_airbnb$room_type)

# Obtain the ANOVA table for the subject-level variables
anova(random_intercept_model_L1variables)
```

Question

How does centering the continuous variables affect the interpretation of the intercept?

Question

How does the inclusion of individual-level variables affect the estimated variance components? More specifically, does the inclusion of individual-level variables affect the within-neighborhood variance, the between-neighborhood variance, or both? Can you explain why?

Question

Which of the individual-level variables are significantly associated with listing prices? How do you interpret the coefficients for these variables?

Including context-level variables

As a third step, we extend the previously fitted model with context-level variables. We start by centering the context-level variables:

```r
# Center the context-level variables
chicago_airbnb <- chicago_airbnb |>
  mutate(WalkScore_c = scale(WalkScore, scale = FALSE),
         TransitScore_c = scale(TransitScore, scale = FALSE),
         BikeScore_c = scale(BikeScore, scale = FALSE))
```

Next, we fit the random intercept model with both subject-level and context-level variables included:

```r
# Fit the random intercept model with subject-level and context-level variables
random_intercept_model_L1L2variables <- lmer(log_price ~ overall_satisfaction_c + room_type
summary(random_intercept_model_L1L2variables)

# Obtain the ANOVA table
anova(random_intercept_model_L1L2variables)
```

Question

How does the inclusion of context-level variables affect the estimated variance components? More specifically, does the inclusion of context-level variables affect the within-neighborhood variance, the between-neighborhood variance, or both? Can you explain why?

Model selection using stepwise elimination

Not all the context-level variables included in the model are significant predictors of listing prices. To obtain a more parsimonious model, we can use the `step()` function from the `lmerTest` package to perform a backward elimination of the fixed-effect terms.

The `step()` function implements **backward elimination** in two stages:

1. **First stage (random effects)**: By default, `step()` first eliminates random-effect terms using likelihood ratio tests (via the `ranova()` function)
2. **Second stage (fixed effects)**: Then it eliminates fixed-effect terms by repeatedly calling `drop1()`, which uses F-tests

Backward elimination process for fixed effects:

1. Starting with the full covariate model (after any random effects elimination), `drop1()` computes F-tests for all marginal fixed-effect terms (i.e., terms that can be dropped while respecting the hierarchy of terms in the model)
2. `step()` identifies the term with the highest (least significant) p-value from the F-tests
3. If that p-value exceeds the significance threshold (default $= 0.05$ for fixed effects), the term is removed
4. The process repeats with the new reduced model until no more terms can be removed

The F-tests used by `drop1()` are based on Satterthwaite's approximation for the denominator degrees of freedom, which is appropriate for mixed models.

Controlling what step() eliminates

The `drop1()` function is specifically designed for testing **fixed effects**. The `step()` function, by default, can eliminate both **random effects** (using likelihood ratio tests) and **fixed effects** (using F-tests). However, this behavior can be controlled using the `reduce.random` and `reduce.fixed` parameters:

- `reduce.random = TRUE` (default): allows elimination of random effects
- `reduce.fixed = TRUE` (default): allows elimination of fixed effects

In this lab, we focus on selecting fixed effects for our random intercept model, so we will set `reduce.random = FALSE` to prevent any changes to the random effect structure. Testing different random effect structures (such as whether to include random slopes) will be covered in more detail in the next lab on longitudinal data analysis.

```r
# Perform stepwise selection to identify the most important predictors
# Set reduce.random = FALSE to only eliminate fixed effects (not random effects)
step(random_intercept_model_L1L2variables, reduce.random = FALSE)
```

Question

Which individual-level and context-level variables are retained in the final model after the stepwise selection procedure?

Exploring cross-level interactions

Finally, we explore the possibility of cross-level interactions between individual-level and context-level variables.

The relationship between the overall satisfaction rating of an individual listing and its price may depend on neighborhood characteristics such as their walkability and access to public transit. For instance, higher satisfaction ratings might have a stronger effect on prices in less walkable or transit-accessible neighborhoods, where positive reviews could help compensate for the disadvantages of limited walkability or transit options. In contrast, in neighborhoods with high walkability or access to public transit ratings, these location-based amenities might already drive prices, reducing the added impact of satisfaction ratings.

To test whether the `overall_satisfaction × WalkScore` interaction is significant, we can fit a model with the interaction and use `drop1()` or `anova()` to test its contribution:

```r
# Fit model with WalkScore interaction
model_interaction_walk <- lmer(log_price ~ overall_satisfaction_c * WalkScore_c + room_type

# Test the interaction term
drop1(model_interaction_walk)
```

The `drop1()` output will show whether the interaction term `overall_satisfaction_c:WalkScore_c` significantly contributes to the model. Look for the p-value in the `Pr(>F)` column for this interaction term.

Question

Based on the F-test from `drop1()`, is the `overall_satisfaction × WalkScore` interaction statistically significant? What does this tell you about whether the effect of satisfaction on price depends on neighborhood walkability?

Interpreting the interaction

If the interaction is significant, we can use the `emmeans` package to compute estimated marginal means (EMMs) at specific combinations of satisfaction ratings and walkability scores. These estimated marginal means represent the model's predicted log prices at particular values of the predictors, holding all other variables constant at their reference levels. By calculating EMMs at representative values (such as low, average, and high levels), we can create a clear visualization of how the effect of satisfaction on price varies across different walkability levels.

```r
# Use representative values based on the data distribution
# For interpretability, we'll use low (Q1), medium (mean = 0), and high (Q3) values
satisfaction_low <- quantile(chicago_airbnb$overall_satisfaction_c, 0.25, na.rm = TRUE)
satisfaction_high <- quantile(chicago_airbnb$overall_satisfaction_c, 0.75, na.rm = TRUE)

walkscore_low <- quantile(chicago_airbnb$WalkScore_c, 0.25, na.rm = TRUE)
walkscore_high <- quantile(chicago_airbnb$WalkScore_c, 0.75, na.rm = TRUE)

# Create a 3x3 grid using low (Q1), average (0), and high (Q3) values
predictions_grid <- emmeans(model_interaction_walk,
                            specs = ~ overall_satisfaction_c * WalkScore_c,
                            at = list(overall_satisfaction_c = c(satisfaction_low, 0, satisfa
                                      WalkScore_c = c(walkscore_low, 0, walkscore_high)),
                            lmer.df = "satterthwaite")
```

```
# Display the 3x3 grid of predictions
print(predictions_grid)
```

Explanation of the `emmeans()` arguments:

- The first argument is the fitted model object (`model_interaction_walk`)
- `specs = ~ overall_satisfaction_c * WalkScore_c` specifies which variables to compute marginal means for, including their interaction
- `at = list(...)` specifies the exact values at which to evaluate the predictors. Here we use three levels for each variable: Q1 (low), 0 (average, since variables are centered), and Q3 (high)
- `lmer.df = "satterthwaite"` specifies the method for computing degrees of freedom and confidence intervals. We use Satterthwaite's approximation to be consistent with the default method used by `lmerTest` for F-tests
- All other predictors in the model (room type, accommodates, bedrooms, etc.) are automatically held at their reference levels or means

The output shows a 3 × 3 grid of estimated marginal means with their standard errors and confidence intervals, representing the estimated log price at each combination of satisfaction and walkability levels.

Now we can visualize these estimated marginal means using an interaction plot:

```
# Convert predictions to data frame for plotting
pred_df <- as.data.frame(predictions_grid)

# Add descriptive labels for the factor levels
pred_df$satisfaction_label <- factor(pred_df$overall_satisfaction_c,
                                     levels = c(satisfaction_low, 0, satisfaction_high),
                                     labels = c("Low (Q1)", "Average (Mean)", "High (Q3)"))

pred_df$walkscore_label <- factor(pred_df$WalkScore_c,
                                  levels = c(walkscore_low, 0, walkscore_high),
                                  labels = c("Low (Q1)", "Average (Mean)", "High (Q3)"))

# Create interaction plot
ggplot(pred_df, aes(x = satisfaction_label, y = emmean,
                    color = walkscore_label, group = walkscore_label)) +
  geom_point(size = 3) +
  geom_line(linewidth = 1) +
  labs(title = "Interaction between Satisfaction and Walkability",
       subtitle = "Estimated log prices at different combinations of satisfaction and walkabi
       x = "Overall Satisfaction Level",
      y = "Estimated Log Price",
      color = "Walkability Level") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

This visualization shows:

- **Three lines** representing neighborhoods with low, average, and high walkability
- **Slopes** indicating how satisfaction relates to price at each walkability level
- If the lines are **parallel**, the interaction effect is weak or non-significant
- If the lines have **different slopes**, it indicates that the satisfaction-price relationship varies by walkability

Question

Based on the interaction plot, how does the relationship between satisfaction ratings and listing prices differ across neighborhoods with varying walkability levels? Do satisfaction ratings matter more (have a stronger effect) in low-walkability or high-walkability neighborhoods?

Coding exercise

Following the example above, test whether the `overall_satisfaction × TransitScore` interaction is statistically significant.

1. Fit a model that includes the `overall_satisfaction × TransitScore` interaction
2. Use `drop1()` to test whether the interaction term is significant
3. Interpret the results: Is the interaction significant? What does this suggest about how satisfaction ratings relate to prices in neighborhoods with different levels of transit accessibility?

Testing strategy for mixed models

In this lab, we have used **F-tests** (via `step()` and `drop1()`) to test fixed effects in our mixed models. These tests:

- Use Satterthwaite's approximation for degrees of freedom
- Work with REML estimation (the default for `lmer()`, which gives better variance component estimates)
- Are appropriate for testing whether fixed effect terms should be included in the model

This approach provides a consistent workflow for testing and selecting fixed effects. For testing **random effects** (such as whether to include random slopes in addition to random intercepts), we use **likelihood ratio tests (LRT)**, which we will cover in the next lab on longitudinal data analysis.

Model diagnostics

The process of performing model diagnostics for the random intercept models fitted in this lab is similar to the one described in the previous labs. Therefore, we refer to those labs for more detailed instructions on assessing model assumptions.