Contents

OH Stats Tutorial: A Beginner's Guide

Welcome!

This tutorial will walk you through statistical analysis of Occupational Health (OH) data using `oh_stats`. **No advanced statistics background required** - we'll explain everything as we go.

The OH data ecosystem consists of two packages: - **oh_parser**: Extracts and validates data from OH profile JSON files - **oh_stats**: Performs statistical analysis on the extracted data

These packages work with **multi-modal data** from OH profiles, including: - **Sensor data**: EMG, accelerometers, heart rate monitors, posture sensors - **Questionnaires**: COPSOQ, MUEQ, ROSA, IPAQ, OSPAQ, NPRS - **Daily self-reports**: Workload ratings, pain assessments

By the end, you'll understand: - What exactly you're analyzing (the unit of analysis) - Why we use Linear Mixed Models (and why t-tests won't work here) - How to run a complete analysis for any modality - How to interpret your results - What numbers matter and what they mean - How to handle common edge cases

Table of Contents

0. What Are We Actually Analyzing?

Before diving into statistics, let's be crystal clear about **what we're analyzing**.

The OH Profile: Multi-Modal Health Data

An OH (Occupational Health) profile contains multiple types of data collected from workers:

Data Type	Examples	Measurement Scale
Sensor metrics	EMG, accelerometer, heart rate	Continuous (numeric)
Questionnaires	COPSOQ, MUEQ, ROSA, IPAQ, OSPAQ	Ordinal (scales) or Continuous (composite scores)
Daily self-reports	Workload, pain ratings	Ordinal (Likert 1-5, NPRS 0-10)
Environmental	Temperature, noise levels	Continuous

The Unit of Analysis: Subject x Day

Your primary unit of analysis is **one subject on one day**.

```
Each row in your analysis = one person's measurements for one day
```

Key points: - Each subject contributes **daily aggregated metrics** (e.g., average EMG over the whole day) - The data are **naturally unbalanced** - some subjects have 3 days, others have 5 - You analyze **each modality separately** (EMG models don't mix with posture or HR models)

Why This Matters

Most statistics textbooks assume: 1. Every observation is independent [X] 2. You have the same number of observations per group [X]

Your data violates both assumptions **by design** - and that's fine! We just need the right tools.

Where oh_parser Ends and oh_stats Begins

```
oh_parser                           oh_stats
---------                           --------
Load JSON profiles                  Prepare analysis tables
Extract ANY modality metrics -->    Check data quality
Clean & validate data               Fit statistical models
                                    Correct for multiple testing
                                    Generate reports
```

oh_parser extracts and validates your data from JSON profiles.
oh_stats answers your research questions about that data.

1. The Problem: Why Can't We Just Use T-Tests?

The Setup

Imagine you're studying muscle fatigue in office workers. You measure their EMG (muscle activity) every day for a week. Your question: **Does muscle activity change over the week?**

Your data looks like this:

```
Subject  Day  EMG_value
------  ---  ---------
Alice     1       10.2
Alice     2        9.8
Alice     3        8.5
Alice     4        7.2
Alice     5        6.9
Bob       1       15.1
Bob       2       14.8
Bob       3       13.2
...
```

Why T-Tests Fail Here

The Problem: Alice's measurements are all related to each other. If Alice naturally has low muscle activity, ALL her measurements will be lower. Bob might naturally have high activity, so ALL his will be higher.

T-tests and basic ANOVA assume **every measurement is independent** - like flipping a coin. But Alice's Day 2 is NOT independent from Alice's Day 1. They're both "Alice measurements."

What happens if we ignore this?

When we pretend related measurements are independent, we get: - **False confidence**: Our p-values are too small - **False discoveries**: We "find" effects that aren't real - **Unreliable results**: Different samples give wildly different answers

A Simple Analogy

Imagine you want to know if a coin is fair. You flip it 100 times and get 52 heads. That's reasonable - probably fair.

Now imagine you flip it 10 times, but you COUNT EACH FLIP 10 TIMES. You now have "100 observations" but really only 10 independent flips. If you got 6 heads in those 10 real flips, you'd report 60 heads in "100 flips" - and wrongly conclude the coin is biased!

That's exactly what happens when you use t-tests on repeated measures data.

2. The Solution: Linear Mixed Models

The Key Idea

Linear Mixed Models (LMMs) solve this by recognizing TWO sources of variation:

1. **Between-subject variation**: Alice vs Bob differences (some people naturally have higher/lower values)
2. **Within-subject variation**: Day-to-day changes for the same person

```
Total variation = Between-subject + Within-subject
```

How It Works (Simplified)

The model says:

```
EMG_value = Overall_average + Day_effect + Subject's_personal_baseline + Random_noise
```

- **Overall_average**: The typical EMG value across everyone
- **Day_effect**: How much Day 2, 3, 4, etc. differ from Day 1 (this is what we want to test!)
- **Subject's_personal_baseline**: Alice is naturally 3 units lower, Bob is 5 units higher, etc.
- **Random_noise**: Unexplained day-to-day fluctuations

By explicitly modeling the "personal baseline," we correctly account for the fact that measurements from the same person are related.

The ICC: How Much Does "Who You Are" Matter?

The **Intraclass Correlation (ICC)** tells you what proportion of the variation is due to between-subject differences.

```
ICC = Between-subject variation / Total variation
```

How to interpret ICC:

ICC Value	Meaning
0.0 - 0.2	Subjects are pretty similar; most variation is day-to-day
0.2 - 0.5	Moderate clustering; both sources matter
0.5 - 0.8	Strong clustering; who you are matters a lot
0.8 - 1.0	Very strong; almost all variation is between people

In our EMG data, ICC is typically 0.4-0.6, meaning about half the variation is "Alice vs Bob" and half is "day-to-day changes."

If ICC is high, you REALLY need mixed models. T-tests would be very wrong.

3. Your First Analysis: Step by Step

Step 1: Load Your Data and Discover What's Available

```python
from oh_parser import load_profiles
from oh_stats import (
    get_profile_summary,
    discover_sensors,
    discover_questionnaires,
    prepare_daily_emg,
    prepare_sensor_data
)

# Load the OH profiles
profiles = load_profiles("/path/to/OH_profiles")

# FIRST: See what data is available (recommended!)
print(get_profile_summary(profiles))

# Output might look like:
# OH Profile Summary (42 subjects)
# ==================================================
#
# SENSOR DATA:
#   emg: 15 metrics
#   heart_rate: 8 metrics
#   noise: 6 metrics
#
# SINGLE-INSTANCE QUESTIONNAIRES:
#   personal: 31 fields
#   biomechanical: 73 fields
#
# DAILY QUESTIONNAIRES:
#   workload: 6 fields

# For more detail on a specific sensor:
sensors = discover_sensors(profiles)
print(sensors['heart_rate'])  # See available heart rate metrics

# Option A: Use convenience function for EMG (common case)
ds = prepare_daily_emg(profiles, side="both")

# Option B: Use generic function for ANY sensor
ds = prepare_sensor_data(
    profiles,
    sensor="heart_rate",
    base_path="sensor_metrics.heart_rate",
    level_names=["date"],
    value_paths=["HR_BPM_stats.*", "HR_ratio_stats.*"],
)

# See what you have
print(f"Number of observations: {len(ds['data'])}")
print(f"Number of subjects: {ds['data']['subject_id'].nunique()}")
print(f"Number of outcomes: {len(ds['outcome_vars'])}")
```

What `side="both"` means: - EMG is measured on left AND right sides - `"both"` keeps them as separate rows (more data, but left/right from same day are related) - `"average"` averages them together (simpler, fewer statistical issues)

Understanding the AnalysisDataset

The `ds` you just created is an **AnalysisDataset** - a sealed container that holds everything needed for analysis:

```
# What's inside ds:
ds['data']          # The actual data (pandas DataFrame, long format)
ds['outcome_vars']  # List of outcome column names
ds['id_var']        # Clustering variable (usually 'subject_id')
ds['time_var']      # Time variable (usually 'day_index')
ds['sensor']        # Which sensor ('emg', 'posture', etc.)
ds['level']         # Aggregation level ('daily', 'hourly')
```

Why "long format" matters:

In long format, each row is ONE observation:

```
subject_id  day_index  side   EMG_intensity.mean_percent_mvc
----------  ---------  ----   ------------------------------
S001        1          left   8.2
S001        1          right  9.1
S001        2          left   7.8
S001        2          right  8.5
S002        1          left   12.3
...
```

This is what mixed models expect. The `AnalysisDataset` ensures all our functions speak the same language.

Step 2: Check Your Data Quality (ALWAYS DO THIS!)

Before any modeling, check for problems:

```
from oh_stats import summarize_outcomes, check_variance, missingness_report

# Pick an outcome to analyze
outcome = "EMG_intensity.mean_percent_mvc"

# Basic summary
summary = summarize_outcomes(ds, [outcome])
print(summary)

# Check for missing data
missing = missingness_report(ds, [outcome])
print(f"Missing: {missing['pct_missing'].iloc[0]:.1f}%")

# Check for "dead" variables (all same value)
variance = check_variance(ds, [outcome])
if variance['is_degenerate'].iloc[0]:
    print("WARNING: This variable has no variation - can't analyze it!")
```

What to look for: - **Missing data > 10%**: Investigate why. Is it random or systematic? -

Degenerate variables: If 95% of values are the same, there's nothing to analyze - **Extreme skewness**: Values like skewness > 2 suggest you might need a transformation

Step 3: Fit the Model

```python
from oh_stats import fit_lmm

# Fit a Linear Mixed Model
result = fit_lmm(ds, outcome)

# Did it work?
if result['converged']:
    print("Model fitted successfully!")
else:
    print("WARNING: Model had problems converging")
    print(result['warnings'])
```

What happens behind the scenes: 1. The model estimates the overall average 2. It estimates how each day differs from Day 1 3. It estimates each subject's personal baseline 4. It calculates how confident we can be in these estimates

Step 4: Look at the Results

```python
# Print a nice summary
from oh_stats import summarize_lmm_result
print(summarize_lmm_result(result))

# See the coefficients
print(result['coefficients'])
```

We'll explain how to interpret these in the next section!

4. Understanding Your Results

The Coefficients Table

When you run a model, you get a table like this:

```
             term   estimate  std_error  p_value
        Intercept      9.406      1.035    0.000
C(day_index)[T.2]     -0.411      0.825    0.618
C(day_index)[T.3]     -0.028      0.839    0.973
C(day_index)[T.4]     -1.931      0.840    0.022  <-- Significant!
C(day_index)[T.5]     -1.643      0.975    0.092
 C(side)[T.right]      0.902      0.550    0.101
```

How to read this:

Column	What it means
term	What's being compared
estimate	The size of the difference

Column	What it means
std_error	How uncertain we are (smaller = more confident)
p_value	Probability this is just random chance (smaller = more likely real)

Interpreting each row:

- **Intercept (9.406)**: The average EMG on Day 1, Left side
- **C(day_index)[T.2] = -0.411**: Day 2 is 0.411 units LOWER than Day 1 (p=0.618, not significant)
- **C(day_index)[T.4] = -1.931**: Day 4 is 1.931 units LOWER than Day 1 (p=0.022, significant!)
- **C(side)[T.right] = 0.902**: Right side is 0.902 units HIGHER than left (p=0.101, not significant)

What Does "Significant" Mean?

The **p-value** answers: "If there were NO real effect, how often would we see data this extreme?"

p-value	Interpretation
$p < 0.01$	Strong evidence of a real effect
$p < 0.05$	Moderate evidence (conventional threshold)
$p < 0.10$	Weak evidence, worth noting but not conclusive
$p > 0.10$	Not enough evidence to claim an effect

IMPORTANT: $p < 0.05$ does NOT mean "95% sure the effect is real." It means "if there's no effect, we'd see this only 5% of the time." These are different statements!

The Random Effects

```
print(result['random_effects'])
# Output: {'group_var': 24.05, 'residual_var': 23.88, 'icc': 0.502}
```

- **group_var (24.05)**: Variance between subjects (how much Alice differs from Bob)
- **residual_var (23.88)**: Variance within subjects (day-to-day noise)
- **icc (0.502)**: 50% of variation is between subjects

This ICC of 0.50 tells us: Mixed models were definitely the right choice! Half of all variation is just "who the person is" - ignoring this would give wrong answers.

5. The Multiple Testing Problem

The Problem

You're analyzing 20 different EMG outcomes. Even if NONE of them have real effects, you'll probably find at least one "significant" result just by chance!

Why? If $p < 0.05$ means "5% chance when there's no effect," then: - Test 1 outcome: 5% chance of false positive - Test 20 outcomes: About 64% chance of AT LEAST ONE false positive!

This is called the **multiple testing problem**.

The Solution: Correction Methods

We adjust our p-values to account for testing multiple outcomes:

```
from oh_stats import fit_all_outcomes, apply_fdr

# Fit models for multiple outcomes
results = fit_all_outcomes(ds, max_outcomes=10)

# Apply FDR correction
fdr_results = apply_fdr(results)
print(fdr_results)
```

Output:

```
                             outcome    p_raw  p_adjusted  significant
EMG_intensity.iemg_percent_seconds   0.0003      0.0007         True
     EMG_intensity.max_percent_mvc   0.0001      0.0007         True
                EMG_apdf.active.p10   0.0180      0.0286         True
                EMG_apdf.active.p50   0.0712      0.0712        False
```

What changed: - `p_raw`: Original p-value from each model - `p_adjusted`: P-value corrected for multiple testing (always bigger) - `significant`: Is p_adjusted < 0.05?

Two Types of Correction

Method	Controls	When to use
FDR (False Discovery Rate)	Expected proportion of false discoveries	Exploratory analysis, many outcomes
FWER (Family-Wise Error Rate)	Chance of ANY false positive	Confirmatory, few primary outcomes

Our strategy: 1. Use **FDR** across all outcomes (discovery phase) 2. Use **FWER (Holm)** for post-hoc comparisons within significant outcomes

Critical Detail: Which P-Value Feeds FDR?

This is subtle but important. When you see the coefficients table:

```
              term   estimate  p_value
C(day_index)[T.2]     -0.411    0.618   <-- NOT this p-value!
C(day_index)[T.3]     -0.028    0.973
C(day_index)[T.4]     -1.931    0.022
C(day_index)[T.5]     -1.643    0.092
```

We do NOT use these individual coefficient p-values for FDR correction.

Instead, we use the **omnibus Likelihood Ratio Test (LRT)** p-value. This tests:

“Does including ‘day’ improve the model AT ALL?”

```
# The LRT compares two models:
# Full model:    EMG ~ day + side + (1|subject)
# Reduced model: EMG ~ side + (1|subject)

# Access it from the result:
print(result['fit_stats']['lrt_pvalue'])  # This feeds FDR!
```

Why the LRT, not coefficient p-values?

- Coefficient p-values test “Day 2 vs Day 1”, “Day 3 vs Day 1”, etc. - that’s many tests per outcome!
- LRT asks one question per outcome: “Is there ANY day effect?”
- FDR needs ONE p-value per outcome to work correctly

The two-layer system:

```
Layer 1 (across outcomes): FDR on LRT p-values
   "Which outcomes show any day effect?"
        ↓
Layer 2 (within outcome): Holm on post-hoc contrasts
   "Which specific days differ?" (only for significant outcomes)
```

6. Checking If Your Analysis Is Valid

Models make assumptions. If these are badly violated, results might be wrong.

The Main Assumptions

1. **Residuals are roughly normal**: The “leftover” variation after modeling should be bell-shaped
2. **Residuals have constant variance**: The spread of residuals shouldn’t change with fitted values
3. **Independence (within clusters)**: After accounting for subjects, remaining variation is random

How to Check

```
from oh_stats import residual_diagnostics

diag = residual_diagnostics(result)

# Quick summary
print(f"Normality test p-value: {diag['normality_p']:.4f}")
print(f"Number of outliers: {diag['n_outliers']}")
```

Interpreting Diagnostics

Normality test (Shapiro-Wilk): - p > 0.05: Residuals look normal (good!) - p < 0.05: Some departure from normality

BUT DON'T PANIC! - LMMs are fairly robust to mild normality violations - With large samples, the test is overly sensitive - Visual checks (QQ plots) are often more useful

What to do if assumptions are violated: 1. Try a transformation (LOG for skewed data) 2. Check for outliers and investigate them 3. Consider if the violation is severe enough to matter

Visual Checks (Recommended)

```
import matplotlib.pyplot as plt
from scipy import stats

fig, axes = plt.subplots(1, 2, figsize=(10, 4))

# QQ Plot: Points should follow the diagonal line
stats.probplot(diag['standardized'], dist="norm", plot=axes[0])
axes[0].set_title("QQ Plot (should be a straight line)")

# Residuals vs Fitted: Should be a random cloud around zero
axes[1].scatter(diag['fitted'], diag['residuals'], alpha=0.5)
axes[1].axhline(y=0, color='r', linestyle='--')
axes[1].set_xlabel("Fitted Values")
axes[1].set_ylabel("Residuals")
axes[1].set_title("Residuals vs Fitted (should be random cloud)")

plt.tight_layout()
plt.show()
```

What good plots look like: - **QQ Plot**: Points follow the diagonal line (some wobble at edges is OK) - **Residuals vs Fitted**: Random scatter around zero, no funnel shape or curves

7. Complete Example with Interpretation

Let's walk through a real analysis from start to finish:

```
"""
Research Question: Does EMG intensity change over the monitoring week?
"""

from oh_parser import load_profiles
from oh_stats import (
    prepare_daily_emg,
    summarize_outcomes,
    check_variance,
    fit_lmm,
    apply_fdr,
    residual_diagnostics,
)

# ============================================================
# STEP 1: Load and Prepare Data
# ============================================================
profiles = load_profiles("/path/to/OH_profiles")
```

```python
ds = prepare_daily_emg(profiles, side="both")

print(f"Loaded {ds['data']['subject_id'].nunique()} subjects")
print(f"Total observations: {len(ds['data'])}")

# ============================================================
# STEP 2: Data Quality Check
# ============================================================
outcome = "EMG_intensity.mean_percent_mvc"

summary = summarize_outcomes(ds, [outcome])
print(f"\nOutcome: {outcome}")
print(f"  Mean: {summary['mean'].iloc[0]:.2f}")
print(f"  SD: {summary['std'].iloc[0]:.2f}")
print(f"  Missing: {summary['pct_missing'].iloc[0]:.1f}%")

variance = check_variance(ds, [outcome])
if variance['is_degenerate'].iloc[0]:
    print("  WARNING: Variable is degenerate!")
else:
    print("  Variance check: OK")

# ============================================================
# STEP 3: Fit the Model
# ============================================================
result = fit_lmm(ds, outcome)

print(f"\nModel Results:")
print(f"  Converged: {result['converged']}")
print(f"  N observations: {result['n_obs']}")
print(f"  N subjects: {result['n_groups']}")
print(f"  ICC: {result['random_effects']['icc']:.3f}")

# ============================================================
# STEP 4: Interpret Coefficients
# ============================================================
print(f"\nDay Effects (compared to Day 1):")
coef = result['coefficients']
for _, row in coef.iterrows():
    if 'day_index' in row['term']:
        day = row['term'].split('[T.')[1].rstrip(']')
        sig = "*" if row['p_value'] < 0.05 else ""
        print(f"  Day {day}: {row['estimate']:+.2f} (p={row['p_value']:.3f}) {sig}")

# ============================================================
# STEP 5: Check Diagnostics
# ============================================================
diag = residual_diagnostics(result)
print(f"\nDiagnostics:")
print(f"  Normality p-value: {diag['normality_p']:.4f}")
print(f"  Outliers detected: {diag['n_outliers']}")

# ============================================================
# STEP 6: Plain English Summary
# ============================================================
print("\n" + "="*50)
print("PLAIN ENGLISH SUMMARY")
print("="*50)

icc = result['random_effects']['icc']
print(f"""
We analyzed EMG mean %MVC across {result['n_groups']} subjects over 5 days.

KEY FINDINGS:

1. CLUSTERING: ICC = {icc:.2f}
   - {icc*100:.0f}% of variation is between subjects (personal baselines)
   - This confirms we needed a mixed model, not simple t-tests

2. DAY EFFECTS:
""")

# Find significant days
sig_days = []
for _, row in coef.iterrows():
    if 'day_index' in row['term'] and row['p_value'] < 0.05:
        day = row['term'].split('[T.')[1].rstrip(']')
        sig_days.append((day, row['estimate'], row['p_value']))

if sig_days:
    for day, est, p in sig_days:
        direction = "lower" if est < 0 else "higher"
        print(f"   - Day {day} was {abs(est):.2f} units {direction} than Day 1 (p={p:.3f})")
else:
    print("   - No significant differences between days")

print(f"""
3. INTERPRETATION:
   - The ICC of {icc:.2f} means individual differences are substantial
   - {"Some days showed significant changes" if sig_days else "No clear trend over the week"}
   - Results account for repeated measures within subjects
""")
```

Example Output

```
Loaded 37 subjects
Total observations: 320

Outcome: EMG_intensity.mean_percent_mvc
  Mean: 9.13
  SD: 6.99
  Missing: 0.0%
  Variance check: OK

Model Results:
  Converged: True
  N observations: 320
  N subjects: 37
  ICC: 0.502

Day Effects (compared to Day 1):
  Day 2: -0.41 (p=0.618)
  Day 3: -0.03 (p=0.973)
  Day 4: -1.93 (p=0.022) *
  Day 5: -1.64 (p=0.092)

Diagnostics:
  Normality p-value: 0.0023
  Outliers detected: 2

==================================================
PLAIN ENGLISH SUMMARY
==================================================

We analyzed EMG mean %MVC across 37 subjects over 5 days.

KEY FINDINGS:

1. CLUSTERING: ICC = 0.50
   - 50% of variation is between subjects (personal baselines)
   - This confirms we needed a mixed model, not simple t-tests

2. DAY EFFECTS:
   - Day 4 was 1.93 units lower than Day 1 (p=0.022)

3. INTERPRETATION:
   - The ICC of 0.50 means individual differences are substantial
   - Some days showed significant changes
   - Results account for repeated measures within subjects
```

8. Working with Different Data Types

The OH profile contains diverse data types that require different statistical approaches.

8.1 Sensor Data (Continuous)

Sensor data like EMG, accelerometer, and heart rate metrics are typically **continuous** - they can take any numeric value within a range.

```
from oh_stats import (
    discover_sensors,
    prepare_daily_emg,
    prepare_sensor_data
```

```python
)

# First: Discover what sensors are available
sensors = discover_sensors(profiles)
print(sensors.keys())  # e.g., ['emg', 'heart_rate', 'noise', 'environment']

# EMG data (convenience function for the most common case)
emg_ds = prepare_daily_emg(profiles, side="both")

# Generic sensor data (works for ANY sensor!)
# Just provide the path and structure

# Heart rate data
hr_ds = prepare_sensor_data(
    profiles,
    sensor="heart_rate",
    base_path="sensor_metrics.heart_rate",
    level_names=["date"],
    value_paths=["HR_BPM_stats.*", "HR_ratio_stats.*"],
)

# Noise data
noise_ds = prepare_sensor_data(
    profiles,
    sensor="noise",
    base_path="sensor_metrics.noise",
    level_names=["date"],
    value_paths=["Noise_statistics.*"],
)

# Environment data
env_ds = prepare_sensor_data(
    profiles,
    sensor="environment",
    base_path="sensor_metrics.environment",
    level_names=["date"],
    value_paths=["*"],  # Get all environment metrics
)
```

Statistical treatment: Standard Linear Mixed Models work well for continuous outcomes. The same analysis approach works regardless of which sensor the data came from - it's the **data type** (continuous, proportion, etc.) that determines the statistical method.

8.2 Questionnaire Data

Questionnaires come in different flavors:

Questionnaire	Measurement	Scale	Analysis Approach
COPSOQ	Psychosocial dimensions	0-100 (computed)	Continuous LMM
MUEQ	Ergonomic risk factors	0-100 (computed)	Continuous LMM

Questionnaire	Measurement	Scale	Analysis Approach
ROSA	Office strain assessment	1-10 ordinal	Treat as continuous*
IPAQ	Physical activity	MET-min/week	LOG transform + LMM
OSPAQ	Sitting/standing time	% proportions	LOGIT transform + LMM
NPRS	Pain rating	0-10 ordinal	Treat as continuous*

*Ordinal scales with 5+ levels are commonly treated as continuous in practice.

```python
from oh_stats import (
    prepare_baseline_questionnaires,
    prepare_daily_pain,
    prepare_daily_workload
)

# Single-instance questionnaires (baseline)
baseline_ds = prepare_baseline_questionnaires(profiles, questionnaire_type="copsoq")

# Daily repeated questionnaires
pain_ds = prepare_daily_pain(profiles)
workload_ds = prepare_daily_workload(profiles)

# Analyze as usual
result = fit_lmm(pain_ds, "nprs_neck")
```

8.3 Composite Scores from Likert Items

Many questionnaires compute dimension scores from multiple Likert items:

```python
from oh_stats import compute_composite_score

# Example: COPSOQ Quantitative Demands from 3 items
df['copsoq_quant_demands'] = compute_composite_score(
    df,
    items=['q1_workload', 'q2_time_pressure', 'q3_pace'],
    reverse_items=['q2_time_pressure'],  # Some items are reverse-coded
    scale_max=5,
    output_scale=(0, 100),  # Rescale to 0-100
    missing_rule="half"  # Require at least half the items
)
```

8.4 Proportion Data (Bounded 0-1)

Some outcomes are proportions (e.g., % time in a posture, OSPAQ sitting percentage):

```python
from oh_stats import fit_lmm, TransformType

# OSPAQ percentages are bounded [0, 1]
# The LOGIT transform handles this
result = fit_lmm(
    ospaq_ds,
    "ospaq_sitting_pct",
    transform=TransformType.LOGIT  # Automatic for registered proportions
)
```

What LOGIT does: Transforms bounded [0,1] data to unbounded (-inf, +inf) for proper LMM fitting.

8.5 Multi-Modal Analysis

Combine sensor data with questionnaires to answer questions like: "Does perceived workload predict muscle activity?"

```python
from oh_stats import align_sensor_questionnaire

# Merge EMG with daily workload on subject x date
merged_ds = align_sensor_questionnaire(emg_ds, workload_ds)

# Now analyze EMG with workload as a predictor
result = fit_lmm(
    merged_ds,
    outcome="EMG_intensity.mean_percent_mvc",
    fixed_effects=["day_index", "workload_composite"],
    day_as_categorical=False
)
```

8.6 Summary: Choosing the Right Approach

Data Type	Example	Recommended Approach
Continuous	EMG %MVC	Standard LMM
Right-skewed	IPAQ MET-min	LOG1P transform + LMM
Proportions	OSPAQ %, rest_percent	LOGIT transform + LMM
Ordinal 5+ levels	ROSA, NPRS	Treat as continuous (pragmatic)
Ordinal <5 levels	Likert items	Compute composite, then LMM
Binary	Yes/No outcomes	Logistic GLMM (future)

9. Quick Reference Card

Minimal Workflow

```python
from oh_parser import load_profiles
from oh_stats import (
    get_profile_summary,
    discover_sensors,
    prepare_daily_emg,
    prepare_sensor_data,
    fit_lmm,
    apply_fdr,
    fit_all_outcomes
)

# 1. Load and Discover
profiles = load_profiles("/path/to/data")
print(get_profile_summary(profiles))  # See what's available

# 2. Prepare ANY sensor data (generic approach)
ds = prepare_sensor_data(
    profiles,
    sensor="your_sensor",
    base_path="sensor_metrics.your_sensor",
    level_names=["date"],
    value_paths=["metric_pattern.*"],
)

# Or use convenience functions for common sensors
ds = prepare_daily_emg(profiles, side="both")

# 3. Single outcome
result = fit_lmm(ds, "your_outcome")
print(result['coefficients'])

# 4. Multiple outcomes with correction
results = fit_all_outcomes(ds, max_outcomes=10)
fdr = apply_fdr(results)
print(fdr[fdr['significant']])
```

What Numbers to Report

Metric	What to report	Example
Sample size	N subjects, N observations	"37 subjects, 320 observations"
ICC	Value and interpretation	"ICC = 0.50, indicating substantial clustering"
Effect	Estimate with 95% CI	"Day 4 was -1.93 (95% CI: -3.58 to -0.28) lower"
Significance	p-value (corrected if multiple tests)	"p = 0.022" or "p_adj = 0.035"
Model fit	AIC for comparison	"AIC = 2023.1"

Reporting Template for Papers

Here's a template you can adapt for your Methods and Results sections:

Methods section: > Daily EMG metrics were analyzed using linear mixed models with day as a fixed effect > and random intercepts for subjects to account for repeated measurements. Given the > exploratory nature of the analysis across N outcomes, p-values were adjusted using > the Benjamini-Hochberg procedure to control the false discovery rate at 5%. Post-hoc > pairwise comparisons between days were corrected using the Holm method. Analyses > were performed using the oh_stats package (v0.3.0) in Python.

Results section: > We analyzed 320 observations from 37 subjects over 5 monitoring days. The intraclass > correlation was 0.50, indicating that 50% of the variance in EMG intensity was > attributable to between-subject differences, justifying the use of mixed models. > After FDR correction, 4 of 10 EMG outcomes showed significant day effects (all > p_adj < 0.05). For mean %MVC, Day 4 was significantly lower than Day 1 > (= -1.93, 95% CI: -3.58 to -0.28, p = 0.022). ### Decision Tree

```
START: Do you have repeated measures per subject?
  |
  +-- NO --> Use regular t-test or ANOVA
  |
  +-- YES --> Use Linear Mixed Model
              |
              How many outcomes are you testing?
              |
              +-- ONE --> Report p-value directly
              |
              +-- MULTIPLE --> Apply FDR correction
                               Report adjusted p-values
```

10. Glossary

Term	Plain English Definition
AIC	A score for comparing models. Lower = better fit. Only meaningful when comparing models on the same data.
Coefficient	The estimated size of an effect. E.g., "Day 4 is 1.93 units lower than Day 1."
Confidence Interval (CI)	A range that probably contains the true effect. "95% CI" means we're 95% confident the true value is in this range.
Converged	The model successfully found a solution. If FALSE, results may be unreliable.
FDR (False Discovery Rate)	A method to control false positives when testing many things. Allows some false positives but controls the proportion.
Fixed Effect	Something we're directly interested in measuring (e.g., day effect, side effect).
ICC (Intraclass Correlation)	What fraction of total variation is due to differences between subjects. High ICC = measurements within a subject are very similar.
Linear Mixed Model (LMM)	A statistical model that handles repeated measures by modeling both fixed effects (what we care about) and random effects (subject differences).
p-value	The probability of seeing your data if there were no real effect. Small p = evidence of real effect.
Random Effect	Variation we want to account for but not directly measure (e.g., each subject's personal baseline).
Residual	The "leftover" after the model's prediction. Good models have small, random residuals.

Term	Plain English Definition
Standard Error (SE)	How uncertain we are about an estimate. Smaller = more confident.
Transform	Converting data (e.g., LOG) to make it better behaved for modeling.

11. Edge Cases & Troubleshooting

Real data is messy. Here's how to handle common problems.

10.1 Missing Days / Unbalanced Data

The situation: Some subjects have 5 days of data, others have only 3.

Good news: LMMs handle this naturally! They use all available data and don't require balanced designs.

What to watch for: - Is missingness random or systematic? (e.g., do subjects drop out because they're injured?) - Very few observations per subject (< 3) may cause convergence issues

```python
# Check missingness patterns
missing = missingness_report(ds)
print(missing[missing['pct_missing'] > 10])  # Flag outcomes with >10% missing
```

10.2 Degenerate Outcomes (No Variance)

The situation: An outcome is nearly constant (e.g., 95% of values are zero).

The problem: No variance = nothing to model. The model literally can't estimate effects.

Solution: Exclude these outcomes from analysis.

```python
variance = check_variance(ds)
degenerate = variance[variance['is_degenerate']]
print(f"Degenerate outcomes to exclude: {list(degenerate['outcome'])}")
```

10.3 Convergence Failures

The situation: `result['converged'] = False`

What it means: The optimizer couldn't find a stable solution. Results are unreliable.

What to try: 1. **Simplify the model**: Remove interactions, use `side="average"` 2. **Check for degenerate outcomes**: Near-constant values cause problems 3. **Check sample size**: Need enough subjects (ideally 20+) and observations 4. **Look at warnings**: `result['warnings']` often explains the issue

```python
if not result['converged']:
    print("Convergence failed!")
    print("Warnings:", result.get('warnings', []))
    # Try simpler model
    ds_simple = prepare_daily_emg(profiles, side="average")
    result = fit_lmm(ds_simple, outcome)
```

10.4 EMG Left/Right Correlation (side="both")

The situation: You kept both sides as separate rows, but left and right from the same subject-day are correlated.

The problem: A subject-only random intercept is an approximation - it doesn't fully capture same-day correlations.

Three defensible strategies:

Strategy	Pros	Cons
`side="average"`	Simplest, no correlation issue	Loses side-specific information
Analyze sides separately	Clean interpretation	Doubles the number of tests
Keep `side="both"`	More power	Slight model misspecification

Recommendation: Start with `side="average"` for simplicity. Use `side="both"` for sensitivity analysis.

10.5 Skewed Distributions

The situation: Residuals are not normally distributed (e.g., right-skewed EMG data).

Don't panic! LMMs are fairly robust to moderate non-normality, especially with larger samples.

When to act: - Severe skewness (> 2) with small samples - Heavy ceiling/floor effects

Solutions:

```
import numpy as np

# Log transform (add small constant to handle zeros)
ds['data']['log_outcome'] = np.log1p(ds['data'][outcome])

# Square root transform (gentler than log)
ds['data']['sqrt_outcome'] = np.sqrt(ds['data'][outcome])
```

10.6 Outliers

The situation: A few extreme values are pulling the model.

How to identify:

```
diag = residual_diagnostics(result)
print(f"Number of outliers (|z| > 3): {diag['n_outliers']}")

# See which observations are outliers
outlier_idx = np.abs(diag['standardized']) > 3
print(ds['data'][outlier_idx])
```

What to do: 1. **Investigate**: Are they data errors or real extreme values? 2. **Sensitivity analysis**: Run with and without outliers 3. **Report both**: "Results were similar with outliers excluded (N=2)"

10.7 Likert/Ordinal Data

The situation: You have questionnaire items on a 1-5 scale.

The theoretical issue: Likert scales are ordinal, not continuous. The difference between 1->2 isn't necessarily the same as 4->5.

Practical guidance:

Distribution	Recommendation
Roughly symmetric, no ceiling/floor	Treat as continuous with LMM (common practice)
Heavy ceiling (most responses = 5)	Consider ordinal models or dichotomize
Heavy floor (most responses = 1)	Consider ordinal models or dichotomize

If treating as continuous: Always report medians and IQR alongside means.

10.8 Proportions (0-1 bounded)

The situation: Your outcome is a proportion (e.g., % time in a posture).

The problem: Values bounded at 0 and 1; residuals can't be normal at the extremes.

Pragmatic solution (current):

```
# Logit transform (handle 0 and 1 with small epsilon)
epsilon = 0.001
ds['data']['logit_prop'] = np.log(
    (ds['data'][outcome] + epsilon) / (1 - ds['data'][outcome] + epsilon)
)
```

Better solution (future): Beta regression for proportions.

10.9 Count Data

The situation: Your outcome is a count (e.g., number of posture transitions).

The problem: Counts are non-negative integers, often with many zeros.

Critical warning: Counts often scale with **wear time**. 10 transitions in 8 hours is NOT the same as 10 transitions in 4 hours!

Pragmatic solution:

```python
# Convert to rate, then log transform
ds['data']['rate'] = ds['data']['count'] / ds['data']['wear_hours']
ds['data']['log_rate'] = np.log1p(ds['data']['rate'])
```

Better solution (future): Poisson or negative binomial models with an offset for exposure.

Quick Troubleshooting Checklist

```
[ ] Model didn't converge?
    -> Try side="average", check for degenerate outcomes

[ ] Residuals look weird?
    -> Check for outliers, consider transformation

[ ] Unexpected results?
    -> Check missingness patterns, verify data quality

[ ] p-values all non-significant but you expected effects?
    -> Check ICC (high ICC = less power), check sample size

[ ] Too many significant results?
    -> Are you using FDR correction? Check for data leakage
```

Still Confused?

That's OK! Statistics is hard. Here are some resources:

1. **For the concepts**: Search "mixed models for repeated measures" on YouTube
2. **For the math**: Gelman & Hill "Data Analysis Using Regression and Multilevel/Hierarchical Models"
3. **For R users**: The `lme4` package documentation has great explanations

Or just run the code and focus on the **plain English summaries**. You don't need to understand every detail to get useful results - that's why we built this package!

OH Stats Tutorial v2.0 (for oh_stats v0.3.0) - January 2026