Table of Contents

OH Stats Tutorial: A Beginner's Guide

Welcome!

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

By the end, you'll understand: - Why we use Linear Mixed Models (and why t-tests won't work here) - How to run a complete analysis - How to interpret your results - What numbers matter and what they mean

---

Table of Contents

---

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

```
from oh_parser import load_profiles
from oh_stats import prepare_daily_emg

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

# Prepare for analysis (this creates a clean dataset)
```

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

# 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)

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

```
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

```
# 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
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

---

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")
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:.3
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
   - 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. Quick Reference Card

Minimal Workflow

```python
from oh_parser import load_profiles
from oh_stats import prepare_daily_emg, fit_lmm, apply_fdr, fit_all_outcomes

# 1. Load
profiles = load_profiles("/path/to/data")
ds = prepare_daily_emg(profiles, side="both")

# 2. Single outcome
result = fit_lmm(ds, "EMG_intensity.mean_percent_mvc")
print(result['coefficients'])

# 3. 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”

Metric	What to report	Example
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"

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
```

---

9. 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).

Term	Plain English Definition
Residual	The "leftover" after the model's prediction. Good models have small, random residuals.
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.

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 v1.0 - January 2026