Jupyter Notebook Execution Report

Name: Kimberly Eder
Project SubTitle: Analysis
Date: December 11, 2025

Cell 1: ■ Markdown

01 Map Sessions

Overview

This notebook matches experiment CSV files with their corresponding EEG recording files based on Unix timestamps. The workflow includes:

1. **File Discovery**: Scans the Data directory for experiment CSV files and EEG CSV files

2. **Timestamp Extraction**: Efficiently extracts the first and last timestamps from each file

3. **Time Matching**: Compares timestamps to find the best EEG recording match for each experiment session

4. **Validation**: Calculates time offset and coverage to verify match quality

5. **Mapping Export**: Creates a session mapping CSV linking experiment files to EEG recordings

Input:

- Experiment CSV files: `01_human-llm-alignment_YYYY-MM-DD_HHhMM.SS.mmm.csv`

- EEG CSV files: `EEG_data_YYYY-MM-DD_HHhMM.SS.mmm.csv`

Output: `session_mapping.csv` with columns:

- `experiment_file`: Name of the experiment CSV file

- `eeg_file`: Name of the matched EEG CSV file (or 'NO MATCH')

- `time_offset_min`: Time difference between experiment and EEG start in minutes

- `coverage`: Percentage of experiment duration covered by EEG recording

Note: This notebook uses an optimized file reading method (seeking to the last line) for 10-100x faster timestamp extraction compared to loading entire files.

1. Import Libaries

Cell 2: ■ Code

```
import pandas as pd
import numpy as np
from pathlib import Path
from datetime import datetime
import glob
```

Cell 3: ■ Markdown

2. Find all Files

Cell 4: ■ Code

```
data_dir = Path('./Data')
asc_files_dir = Path('./asc_files')
```

```
# Find all experiment CSV files (with complete data) and ASC eye-tracking files
exp_files = sorted([f for f in data_dir.glob('01_human-llm-alignment_*.csv')])
eye_files = sorted([g for g in data_dir.glob('*.asc')])
```

```
# Find all EEG and Eyetracking files
eeg_csv_files = sorted(data_dir.glob('EEG_data_*.csv'))
eyetracking_edf_files = sorted(data_dir.glob('*.EDF'))
```

```
print(f"Found:")
print(f" Experiment CSVs: {len(exp_files)}")
print(f" EEG CSV Files: {len(eeg_csv_files)}")
print(f" Eye-Tracking EDF Files: {len(eyetracking_edf_files)}")
print(f" Eye-Tracking ASC Files: {len(eye_files)}")
```

Output:

```
Found:
  Experiment CSVs: 107
  EEG CSV Files: 33
  Eye-Tracking EDF Files: 9
  Eye-Tracking ASC Files: 20
```

Cell 5: ■ Markdown

3. Extract Timestamps from Experiment Files

Cell 6: ■ Code

```
def get_experiment_timestamps(csv_file):
"""Extract start and end time from experiment CSV."""
try:
df = pd.read_csv(csv_file, engine='python', on_bad_lines='skip')
```

```
# Get el_recording.started_Unix timestamp
if 'el_recording.started_Unix' not in df.columns:
return None
```

```
start_unix = df['el_recording.started_Unix'].dropna()
if len(start_unix) == 0:
return None
```

```
start = start_unix.iloc[0]
```

```
# Determine number of trials from ScenarioLoop (more accurate than counting rows)
n_trials = 0
if 'ScenarioLoop.thisN' in df.columns:
max_trial_n = df['ScenarioLoop.thisN'].dropna()
if len(max_trial_n) &gt; 0:
# thisN is 0-indexed, so max + 1 = total trials
n_trials = int(max_trial_n.max()) + 1
```

```
# Fall back to counting AI_Response events if ScenarioLoop not available
if n_trials == 0 and 'AI_Response.started' in df.columns:
ai_response_times = df['AI_Response.started'].dropna()
n_trials = len(ai_response_times)
```

```
# Skip files with very few trials (likely test runs)
if n_trials &lt; 10:
return None
```

```
# Calculate duration and end time
if 'AI_Response.started' in df.columns:
ai_response_times = df['AI_Response.started'].dropna()
```

```
if len(ai_response_times) &gt; 0:
duration = ai_response_times.max()
else:
duration = 3000 # default 50 min estimate
else:
duration = 3000 # default estimate
```

```
end = start + duration + 300 # +5 minutes buffer
```

```
return {
'file': csv_file.name,
'start_unix': start,
'end_unix': end,
'duration': duration,
'n_trials': n_trials,
'date': datetime.fromtimestamp(start).strftime('%Y-%m-%d %H:%M:%S')
}
```

```
except Exception as e:
print(f"Error at {csv_file.name}: {e}")
return None
```

```
# Collect info for all experiment files
exp_info = []
for f in exp_files:
info = get_experiment_timestamps(f)
if info:
exp_info.append(info)
```

```
df_exp = pd.DataFrame(exp_info)
print(f"\nComplete experiments: {len(df_exp)}")
```

```
# Show all experiments (not just first 10)
df_exp
```

Output:

```
Complete experiments: 24
                                                 file  ...                 date
```

```
0   01_human-llm-alignment_2025-11-17_11h36.44.912...  ...  2025-11-17 11:38:44
1   01_human-llm-alignment_2025-11-20_13h16.37.791...  ...  2025-11-20 13:17:08
2   01_human-llm-alignment_2025-11-20_15h02.07.504...  ...  2025-11-20 15:04:29
3   01_human-llm-alignment_2025-11-20_16h25.12.833...  ...  2025-11-20 16:25:37
4   01_human-llm-alignment_2025-11-24_14h13.05.529...  ...  2025-11-24 14:15:13
5   01_human-llm-alignment_2025-11-27_09h44.35.888...  ...  2025-11-27 09:48:52
6   01_human-llm-alignment_2025-11-27_10h18.29.349...  ...  2025-11-27 10:25:33
7   01_human-llm-alignment_2025-11-27_10h49.01.727...  ...  2025-11-27 10:49:15
8   01_human-llm-alignment_2025-11-27_11h29.15.053...  ...  2025-11-27 11:29:51
9   01_human-llm-alignment_2025-11-27_12h50.01.880...  ...  2025-11-27 12:51:00
10  01_human-llm-alignment_2025-12-01_09h17.38.489...  ...  2025-12-01 09:18:20
11  01_human-llm-alignment_2025-12-01_10h17.52.885...  ...  2025-12-01 10:19:04
12  01_human-llm-alignment_2025-12-01_12h45.09.945...  ...  2025-12-01 12:49:43
13  01_human-llm-alignment_2025-12-01_14h10.59.014...  ...  2025-12-01 14:12:49
14  01_human-llm-alignment_2025-12-01_16h21.26.160...  ...  2025-12-01 16:22:29
15  01_human-llm-alignment_2025-12-02_13h26.02.964...  ...  2025-12-02 13:26:22
16  01_human-llm-alignment_2025-12-04_11h37.33.132...  ...  2025-12-04 11:40:50
17  01_human-llm-alignment_2025-12-04_13h15.42.235...  ...  2025-12-04 13:16:14
18  01_human-llm-alignment_2025-12-04_14h00.44.858...  ...  2025-12-04 14:01:30
19  01_human-llm-alignment_2025-12-04_14h43.42.571...  ...  2025-12-04 14:44:32
20  01_human-llm-alignment_2025-12-05_13h41.20.156...  ...  2025-12-05 13:43:12
21  01_human-llm-alignment_2025-12-05_14h03.40.086...  ...  2025-12-05 14:04:20
22  01_human-llm-alignment_2025-12-05_14h27.12.562...  ...  2025-12-05 14:27:53
23  01_human-llm-alignment_2025-12-05_14h43.55.570...  ...  2025-12-05 14:44:23

[24 rows x 6 columns]
```

Cell 7: ■ Markdown

4. Extract EEG Timestamps

Cell 8: ■ Code

```
def get_eeg_csv_timestamps(csv_file):
"""Extract start and end time from EEG CSV."""
try:
```

```
# Use tail method for last line (much faster)
with open(csv_file, 'r') as f:
# First line (Header)
header = f.readline().strip().split(',')
# Second line (first data line)
first_line = f.readline().strip().split(',')
```

```
# Last line with tail-like method
f.seek(0, 2) # Go to end of file
file_size = f.tell()
```

```
# Read last ~2000 bytes (should contain multiple lines)
offset = min(2000, file_size)
f.seek(file_size - offset)
lines = f.readlines()
last_line = lines[-1].strip().split(',')
```

```
# Find Time column index
time_idx = header.index('Time')
```

```
start_time = float(first_line[time_idx])
end_time = float(last_line[time_idx])
```

```
return {
'file': csv_file.name,
'start_unix': start_time,
'end_unix': end_time,
'duration': end_time - start_time,
'date': datetime.fromtimestamp(start_time).strftime('%Y-%m-%d %H:%M:%S')
}
except Exception as e:
print(f"Error at {csv_file.name}: {e}")
return None
```

```
# Collect EEG info
eeg_info = []
for f in eeg_csv_files:
info = get_eeg_csv_timestamps(f)
```

```
if info:
eeg_info.append(info)
```

```
df_eeg = pd.DataFrame(eeg_info)
print(f"\nEEG CSV files: {len(df_eeg)}")
df_eeg
```

Output:

```
EEG CSV files: 33
                       file    start_unix  ...     duration                 date
0   EEG_data_1763373596.csv  1.763374e+09  ...  3808.189084  2025-11-17 10:59:57
1   EEG_data_1763640940.csv  1.763641e+09  ...  2106.895494  2025-11-20 13:15:41
2   EEG_data_1763647289.csv  1.763647e+09  ...   443.182222  2025-11-20 15:01:30
3   EEG_data_1763652280.csv  1.763652e+09  ...   110.268764  2025-11-20 16:24:41
4   EEG_data_1763989917.csv  1.763990e+09  ...   293.910738  2025-11-24 14:11:58
5   EEG_data_1763990890.csv  1.763991e+09  ...  2062.123921  2025-11-24 14:28:10
6   EEG_data_1764232442.csv  1.764232e+09  ...  1310.406297  2025-11-27 09:34:03
7   EEG_data_1764235080.csv  1.764235e+09  ...  3514.616303  2025-11-27 10:18:01
8   EEG_data_1764239330.csv  1.764239e+09  ...   310.408256  2025-11-27 11:28:50
9   EEG_data_1764244046.csv  1.764244e+09  ...  2472.474455  2025-11-27 12:47:27
10  EEG_data_1764576993.csv  1.764577e+09  ...  1894.647223  2025-12-01 09:16:33
11  EEG_data_1764580647.csv  1.764581e+09  ...  1283.412992  2025-12-01 10:17:27
12  EEG_data_1764583881.csv  1.764584e+09  ...   320.109914  2025-12-01 11:11:22
13  EEG_data_1764589375.csv  1.764589e+09  ...  2254.201440  2025-12-01 12:42:55
14  EEG_data_1764594621.csv  1.764595e+09  ...  1056.743533  2025-12-01 14:10:22
15  EEG_data_1764597232.csv  1.764597e+09  ...  1326.188432  2025-12-01 14:53:52
16  EEG_data_1764602406.csv  1.764602e+09  ...   667.398611  2025-12-01 16:20:06
17  EEG_data_1764604070.csv  1.764604e+09  ...   356.496075  2025-12-01 16:47:51
18  EEG_data_1764678104.csv  1.764678e+09  ...    11.870370  2025-12-02 13:21:44
19  EEG_data_1764678674.csv  1.764679e+09  ...  1979.629943  2025-12-02 13:31:15
20  EEG_data_1764844642.csv  1.764845e+09  ...  2120.028801  2025-12-04 11:37:23
21  EEG_data_1764846902.csv  1.764847e+09  ...    58.567348  2025-12-04 12:15:03
22  EEG_data_1764847136.csv  1.764847e+09  ...    94.347673  2025-12-04 12:18:57
23  EEG_data_1764850359.csv  1.764850e+09  ...  1068.055175  2025-12-04 13:12:39
24  EEG_data_1764851452.csv  1.764851e+09  ...    60.728332  2025-12-04 13:30:53
```

```
25  EEG_data_1764851971.csv  1.764852e+09  ...   509.587128  2025-12-04 13:39:32
26  EEG_data_1764853208.csv  1.764853e+09  ...  1443.646410  2025-12-04 14:00:09
27  EEG_data_1764855733.csv  1.764856e+09  ...   396.729711  2025-12-04 14:42:14
28  EEG_data_1764856407.csv  1.764856e+09  ...   632.648865  2025-12-04 14:53:28
29  EEG_data_1764938463.csv  1.764938e+09  ...  1218.628298  2025-12-05 13:41:04
30  EEG_data_1764939806.csv  1.764940e+09  ...  1153.772549  2025-12-05 14:03:27
31  EEG_data_1764941223.csv  1.764941e+09  ...   742.663375  2025-12-05 14:27:04
32  EEG_data_1764942219.csv  1.764942e+09  ...   936.712946  2025-12-05 14:43:40

[33 rows x 5 columns]
```

Cell 9: ■ Markdown

5. Match Experiment ↔ EEG Based on Timestamps

Cell 10: ■ Code

```python
def find_matching_eeg(exp_row, df_eeg, min_coverage_percent=10,
allow_multiple=True):
"""Find matching EEG file(s) for an experiment.
```

```
Args:
exp_row: Experiment info
df_eeg: DataFrame with EEG info
min_coverage_percent: Minimum coverage percentage to consider a match (default:
15%)
allow_multiple: If True, can combine multiple EEG segments if there are gaps
```

```
Returns:
dict with match info, or None if no match found
"""
exp_start = exp_row['start_unix']
exp_end = exp_row['end_unix']
exp_duration = exp_end - exp_start
```

```python
# Find EEG files whose time range overlaps with the experiment
matches = []
for idx, eeg_row in df_eeg.iterrows():
```

```
eeg_start = eeg_row['start_unix']
eeg_end = eeg_row['end_unix']
```

```
# Calculate overlap
overlap_start = max(exp_start, eeg_start)
overlap_end = min(exp_end, eeg_end)
overlap_duration = max(0, overlap_end - overlap_start)
```

```
coverage_percent = (overlap_duration / exp_duration) * 100
```

```
# Require minimum coverage
if coverage_percent &gt;= min_coverage_percent:
# Calculate time offset (negative = EEG starts after experiment)
time_offset = exp_start - eeg_start
```

```
matches.append({
'eeg_file': eeg_row['file'],
'offset_seconds': time_offset,
'offset_minutes': time_offset / 60,
'coverage_percent': coverage_percent,
'coverage': f'{coverage_percent:.1f}%',
'coverage_seconds': overlap_duration,
'overlap_minutes': overlap_duration / 60,
'is_complete': eeg_end &gt;= exp_end,
'eeg_start': eeg_start,
'eeg_end': eeg_end
})
```

```
if not matches:
return None
```

```
# If allow_multiple and no single file covers &gt;80%, try combining multiple segments
if allow_multiple:
best_single = max(matches, key=lambda x: x['coverage_seconds'])
```

```
if best_single['coverage_percent'] &lt; 80:
# Try to find adjacent EEG files that can be combined
matches_sorted = sorted(matches, key=lambda x: x['eeg_start'])
```

```
if len(matches_sorted) &gt; 1:
# Check if we can combine consecutive segments to improve coverage
total_coverage = sum([m['coverage_seconds'] for m in matches_sorted])
combined_coverage_percent = (total_coverage / exp_duration) * 100
```

```
if combined_coverage_percent &gt; best_single['coverage_percent']:
# Return combined info
eeg_files = [m['eeg_file'] for m in matches_sorted]
return {
'eeg_file': ' + '.join(eeg_files), # Mark as combined
'offset_minutes': matches_sorted[0]['offset_minutes'],
'coverage': f'{combined_coverage_percent:.1f}%',
'coverage_seconds': total_coverage,
'overlap_minutes': total_coverage / 60,
'is_complete': matches_sorted[-1]['is_complete'],
'is_combined': True,
'n_segments': len(eeg_files)
}
```

```
return best_single
else:
# Choose match with best coverage (longest overlap)
return max(matches, key=lambda x: x['coverage_seconds'])
```

```
# Match all experiments
session_map = []
for idx, exp in df_exp.iterrows():
match = find_matching_eeg(exp, df_eeg, min_coverage_percent=15,
allow_multiple=True)
session_map.append({
'experiment_file': exp['file'],
'experiment_date': exp['date'],
'n_trials': exp['n_trials'],
'exp_duration_min': exp['duration'] / 60,
'eeg_file': match['eeg_file'] if match else 'NO MATCH',
'time_offset_min': match['offset_minutes'] if match else None,
'coverage': match['coverage'] if match else None,
'overlap_min': match['overlap_minutes'] if match else None,
'is_complete': match['is_complete'] if match else False,
'is_combined': match.get('is_combined', False) if match else False,
'n_segments': match.get('n_segments', 1) if match else None
})
```

```
df_sessions = pd.DataFrame(session_map)
print(f"\nSession Mapping (≥15% coverage, with multiple segment support):")
print(f" Matched: {df_sessions['eeg_file'].ne('NO MATCH').sum()}")
print(f" Unmatched: {df_sessions['eeg_file'].eq('NO MATCH').sum()}")
print(f" Combined (multiple segments): {df_sessions['is_combined'].sum()}")
print(f" Complete coverage: {df_sessions['is_complete'].sum()}")
```

```
# Show combined sessions
combined_sessions = df_sessions[df_sessions['is_combined'] == True]
if len(combined_sessions) &gt; 0:
print(f"\n■■ Sessions using multiple EEG segments:")
for idx, row in combined_sessions.iterrows():
print(f" {row['experiment_file']}: {row['eeg_file']} ({row['coverage']})")
```

```
df_sessions
```

Output:

```
Session Mapping (≥15% coverage, with multiple segment support):
  Matched: 21
  Unmatched: 3
  Combined (multiple segments): 5
  Complete coverage: 3

■■  Sessions using multiple EEG segments:
    01_human-llm-alignment_2025-12-01_14h10.59.014.csv: EEG_data_1764594621.csv + EEG_data_176459
    01_human-llm-alignment_2025-12-01_16h21.26.160.csv: EEG_data_1764602406.csv + EEG_data_176460
    01_human-llm-alignment_2025-12-04_13h15.42.235.csv: EEG_data_1764850359.csv + EEG_data_176485
    01_human-llm-alignment_2025-12-04_14h43.42.571.csv: EEG_data_1764855733.csv + EEG_data_176485
    01_human-llm-alignment_2025-12-05_13h41.20.156.csv: EEG_data_1764938463.csv + EEG_data_176493
                                      experiment_file  ... n_segments
```

```
0   01_human-llm-alignment_2025-11-17_11h36.44.912...  ...         1.0
1   01_human-llm-alignment_2025-11-20_13h16.37.791...  ...         1.0
2   01_human-llm-alignment_2025-11-20_15h02.07.504...  ...         NaN
3   01_human-llm-alignment_2025-11-20_16h25.12.833...  ...         NaN
4   01_human-llm-alignment_2025-11-24_14h13.05.529...  ...         1.0
5   01_human-llm-alignment_2025-11-27_09h44.35.888...  ...         1.0
6   01_human-llm-alignment_2025-11-27_10h18.29.349...  ...         1.0
7   01_human-llm-alignment_2025-11-27_10h49.01.727...  ...         1.0
8   01_human-llm-alignment_2025-11-27_11h29.15.053...  ...         NaN
9   01_human-llm-alignment_2025-11-27_12h50.01.880...  ...         1.0
10  01_human-llm-alignment_2025-12-01_09h17.38.489...  ...         1.0
11  01_human-llm-alignment_2025-12-01_10h17.52.885...  ...         1.0
12  01_human-llm-alignment_2025-12-01_12h45.09.945...  ...         1.0
13  01_human-llm-alignment_2025-12-01_14h10.59.014...  ...         2.0
14  01_human-llm-alignment_2025-12-01_16h21.26.160...  ...         2.0
15  01_human-llm-alignment_2025-12-02_13h26.02.964...  ...         1.0
16  01_human-llm-alignment_2025-12-04_11h37.33.132...  ...         1.0
17  01_human-llm-alignment_2025-12-04_13h15.42.235...  ...         2.0
18  01_human-llm-alignment_2025-12-04_14h00.44.858...  ...         1.0
19  01_human-llm-alignment_2025-12-04_14h43.42.571...  ...         2.0
20  01_human-llm-alignment_2025-12-05_13h41.20.156...  ...         2.0
21  01_human-llm-alignment_2025-12-05_14h03.40.086...  ...         1.0
22  01_human-llm-alignment_2025-12-05_14h27.12.562...  ...         1.0
23  01_human-llm-alignment_2025-12-05_14h43.55.570...  ...         1.0

[24 rows x 11 columns]
```

Cell 11: ■ Markdown

6. Save Session Mapping

Cell 12: ■ Code

```python
# Save as CSV
df_sessions.to_csv('./session_mapping.csv', index=False)
print("Session mapping saved: ./session_mapping.csv")
```

```python
# Show only matched sessions
df_matched = df_sessions[df_sessions['eeg_file'] != 'NO MATCH'].copy()
print(f"\n{len(df_matched)} complete sessions for analysis:")
df_matched
```

Output:

```
Session mapping saved: ./session_mapping.csv

21 complete sessions for analysis:
                                      experiment_file  ... n_segments
0   01_human-llm-alignment_2025-11-17_11h36.44.912...  ...        1.0
1   01_human-llm-alignment_2025-11-20_13h16.37.791...  ...        1.0
4   01_human-llm-alignment_2025-11-24_14h13.05.529...  ...        1.0
5   01_human-llm-alignment_2025-11-27_09h44.35.888...  ...        1.0
6   01_human-llm-alignment_2025-11-27_10h18.29.349...  ...        1.0
7   01_human-llm-alignment_2025-11-27_10h49.01.727...  ...        1.0
9   01_human-llm-alignment_2025-11-27_12h50.01.880...  ...        1.0
10  01_human-llm-alignment_2025-12-01_09h17.38.489...  ...        1.0
11  01_human-llm-alignment_2025-12-01_10h17.52.885...  ...        1.0
12  01_human-llm-alignment_2025-12-01_12h45.09.945...  ...        1.0
13  01_human-llm-alignment_2025-12-01_14h10.59.014...  ...        2.0
14  01_human-llm-alignment_2025-12-01_16h21.26.160...  ...        2.0
15  01_human-llm-alignment_2025-12-02_13h26.02.964...  ...        1.0
16  01_human-llm-alignment_2025-12-04_11h37.33.132...  ...        1.0
17  01_human-llm-alignment_2025-12-04_13h15.42.235...  ...        2.0
18  01_human-llm-alignment_2025-12-04_14h00.44.858...  ...        1.0
19  01_human-llm-alignment_2025-12-04_14h43.42.571...  ...        2.0
20  01_human-llm-alignment_2025-12-05_13h41.20.156...  ...        2.0
21  01_human-llm-alignment_2025-12-05_14h03.40.086...  ...        1.0
22  01_human-llm-alignment_2025-12-05_14h27.12.562...  ...        1.0
23  01_human-llm-alignment_2025-12-05_14h43.55.570...  ...        1.0

[21 rows x 11 columns]
```

Cell 13: ■ Code

```python
print(f"\n{'='*80}")
print("DIAGNOSTIC: Why are some experiments excluded?")
print(f"{'='*80}")
```

```
print(f"\n✓ Total Experiment CSVs found: {len(exp_files)}")
```

```
# Check which ones were filtered out by n_trials &lt; 10
print(f"\n[Filter 1] Checking for test runs (n_trials &lt; 10)...")
excluded_by_trials = []
for f in exp_files:
try:
df = pd.read_csv(f, engine='python', on_bad_lines='skip')
n_trials = 0
if 'ScenarioLoop.thisN' in df.columns:
max_trial_n = df['ScenarioLoop.thisN'].dropna()
if len(max_trial_n) &gt; 0:
n_trials = int(max_trial_n.max()) + 1
```

```
if n_trials == 0 and 'AI_Response.started' in df.columns:
ai_response_times = df['AI_Response.started'].dropna()
n_trials = len(ai_response_times)
```

```
if n_trials &lt; 10:
excluded_by_trials.append({
'file': f.name,
'n_trials': n_trials,
'reason': 'Test run (&lt; 10 trials)'
})
except:
pass
```

```
if excluded_by_trials:
print(f" ■■ {len(excluded_by_trials)} experiments excluded (test runs):")
for item in excluded_by_trials:
print(f" - {item['file']}: {item['n_trials']} trials")
else:
print(f" ✓ No test runs detected")
```

```
print(f"\n✓ Experiments passed trial filter: {len(df_exp)}")
```

```
# Check coverage distribution
print(f"\n[Filter 2] Coverage distribution (15% minimum required)...")
df_sessions_with_coverage = df_sessions.copy()
df_sessions_with_coverage['coverage_num'] = pd.to_numeric(
df_sessions_with_coverage['coverage'].str.rstrip('%'), errors='coerce'
)
```

```
# All sessions with coverage info
coverage_available = df_sessions_with_coverage[df_sessions_with_coverage['eeg_file'] != 'NO MATCH'].copy()
```

```
print(f"\n Sessions with EEG data found:")
print(f" ✓ ≥15% coverage: {(coverage_available['coverage_num'] &gt;= 15).sum()}")
print(f" ■■ 10-15% coverage: {((coverage_available['coverage_num'] &gt;= 10) & (coverage_available['coverage_num'] &lt; 15)).sum()}")
print(f" ■■ 5-10% coverage: {((coverage_available['coverage_num'] &gt;= 5) & (coverage_available['coverage_num'] &lt; 10)).sum()}")
print(f" ■ &lt;5% coverage: {(coverage_available['coverage_num'] &lt; 5).sum()}")
```

```
# Show NO MATCH experiments
unmatched = df_sessions[df_sessions['eeg_file'] == 'NO MATCH']
if len(unmatched) &gt; 0:
print(f"\n ■ {len(unmatched)} experiments without matching EEG:")
for idx, row in unmatched.iterrows():
print(f" - {row['experiment_file']} ({row['n_trials']} trials, {row['exp_duration_min']:.1f} min)")
else:
print(f"\n ✓ All experiments have matching EEG")
```

```
# Calculate how many sessions would be available at different thresholds
print(f"\n" + "="*80)
print("SENSITIVITY ANALYSIS: Sessions available at different coverage thresholds")
print("="*80)
```

```
thresholds = [5, 10, 15, 20, 30, 50]
for thresh in thresholds:
# Create temp matching with different threshold
session_map_temp = []
```

```
for idx, exp in df_exp.iterrows():
match = find_matching_eeg(exp, df_eeg, min_coverage_percent=thresh)
session_map_temp.append({
'experiment_file': exp['file'],
'eeg_file': match['eeg_file'] if match else 'NO MATCH',
})
```

```
df_temp = pd.DataFrame(session_map_temp)
n_matched = df_temp['eeg_file'].ne('NO MATCH').sum()
print(f" Threshold ≥{thresh:2d}%: {n_matched:2d} sessions")
```

```
print(f"\n✓ Current setting (15%): {df_sessions['eeg_file'].ne('NO MATCH').sum()}
sessions")
print(f"✓ Sessions with 100% coverage: {(coverage_available['coverage_num'] ==
100).sum()}")
```

```
print(f"\n■ RECOMMENDATION:")
print(f" If you need more sessions, lower the threshold to 10% or check why")
print(f" some experiments don't have matching EEG files.")
print(f" Current EEG files available: {len(df_eeg)}")
```

Output:

```
================================================================================
DIAGNOSTIC: Why are some experiments excluded?
================================================================================

✓ Total Experiment CSVs found: 107

[Filter 1] Checking for test runs (n_trials &lt; 10)...
  ■■ 83 experiments excluded (test runs):
    - 01_human-llm-alignment_2025-11-12_09h11.37.052.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_09h15.36.432.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_09h22.17.806.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_09h27.02.139.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_09h36.01.606.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_09h38.08.675.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_09h39.11.326.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_09h42.55.821.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_09h47.57.807.csv: 2 trials
    - 01_human-llm-alignment_2025-11-12_09h50.34.823.csv: 2 trials
    - 01_human-llm-alignment_2025-11-12_17h02.54.154.csv: 2 trials
    - 01_human-llm-alignment_2025-11-12_17h09.31.602.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_17h51.48.135.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_17h57.23.632.csv: 1 trials
    - 01_human-llm-alignment_2025-11-12_17h58.29.462.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_07h48.31.800.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_07h49.41.480.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_07h51.29.464.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_08h08.01.580.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_08h10.43.779.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_08h17.14.277.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_08h17.55.241.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_08h20.11.688.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_08h29.21.055.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_12h34.03.419.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_12h49.58.179.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_12h51.48.365.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_13h01.36.703.csv: 0 trials
    - 01_human-llm-alignment_2025-11-13_13h03.58.744.csv: 0 trials
    - 01_human-llm-alignment_2025-11-13_13h05.16.386.csv: 0 trials
    - 01_human-llm-alignment_2025-11-13_13h07.25.542.csv: 0 trials
    - 01_human-llm-alignment_2025-11-13_13h08.05.696.csv: 0 trials
    - 01_human-llm-alignment_2025-11-13_13h08.52.157.csv: 0 trials
    - 01_human-llm-alignment_2025-11-13_13h33.13.593.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_13h36.22.996.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h07.35.813.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h08.28.727.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h10.55.078.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h12.32.995.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h17.26.349.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h21.33.327.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h22.28.672.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h27.03.734.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h28.55.196.csv: 1 trials
    - 01_human-llm-alignment_2025-11-13_14h31.19.567.csv: 1 trials
    - 01_human-llm-alignment_2025-11-14_14h01.35.730.csv: 1 trials
    - 01_human-llm-alignment_2025-11-14_14h05.08.803.csv: 6 trials
    - 01_human-llm-alignment_2025-11-14_14h43.25.913.csv: 1 trials
    - 01_human-llm-alignment_2025-11-14_14h47.46.406.csv: 1 trials
    - 01_human-llm-alignment_2025-11-14_14h49.10.771.csv: 1 trials
    - 01_human-llm-alignment_2025-11-14_14h50.19.485.csv: 1 trials
    - 01_human-llm-alignment_2025-11-14_14h54.15.403.csv: 5 trials
    - 01_human-llm-alignment_2025-11-17_09h29.38.121.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h34.46.099.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h40.29.614.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h44.52.979.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h48.47.655.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h49.28.739.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h54.45.505.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h55.15.594.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h57.56.878.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_09h59.53.601.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h01.42.476.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h04.20.606.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h05.15.687.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h06.47.978.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h08.04.534.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h09.41.882.csv: 6 trials
    - 01_human-llm-alignment_2025-11-17_10h15.42.678.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h17.53.317.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h18.15.716.csv: 2 trials
    - 01_human-llm-alignment_2025-11-17_10h22.52.482.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h33.51.003.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_10h36.18.993.csv: 4 trials
    - 01_human-llm-alignment_2025-11-17_10h41.29.833.csv: 4 trials
    - 01_human-llm-alignment_2025-11-17_11h03.52.791.csv: 8 trials
```

```
    - 01_human-llm-alignment_2025-11-17_11h13.59.258.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_11h18.47.002.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_11h24.57.535.csv: 1 trials
    - 01_human-llm-alignment_2025-11-17_16h18.55.779.csv: 1 trials
    - 01_human-llm-alignment_2025-11-20_12h46.39.280.csv: 3 trials
    - 01_human-llm-alignment_2025-11-20_15h51.50.944.csv: 4 trials
    - 01_human-llm-alignment_2025-11-24_12h52.19.205.csv: 5 trials
✓ Experiments passed trial filter: 24
[Filter 2] Coverage distribution (15% minimum required)...
  Sessions with EEG data found:
    ✓ ≥15% coverage: 21
    ■■  10-15% coverage: 0
```

... (26 more lines truncated)

Cell 14: ■ Markdown

7. Summary

Next steps:

1. **Preprocessing**: All matched EEG files through preprocessing pipeline (01-04)

2. **ERP Analysis**: Calculate ERPs for each session (05_ERP_Analysis)

3. **Grand Average**: Combine all sessions for group ERPs

4. **Statistics**: Condition comparisons across all sessions