hw2

January 31, 2024

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import ttest_1samp
```

[2]:
```python
df = pd.read_csv('hw2.csv', header = None)

samples = df.shape[0]
trials = df.shape[1]
total_intervals = samples - 1

print(f'Samples: ', samples)
print(f'Trials: ', trials)
print(f'Interval Between Samples: ', total_intervals)
```

```
Samples: 201
Trials: 20
Interval Between Samples: 200
```

[3]:
```python
df
```

[3]:
```
           0         1         2        3        4         5         6  \
0    0.53767  1.833900 -2.258800  0.86217  0.31877 -1.307700 -0.433590
1    0.67150 -1.207500  0.717240  1.63020  0.48889  1.034700  0.726890
2   -0.10224 -0.241450  0.319210  0.31286 -0.86488 -0.030051 -0.164880
3   -1.08910  0.032557  0.552530  1.10060  1.54420  0.085931 -1.491600
4    1.41930  0.291580  0.197810  1.58770 -0.80447  0.696620  0.835090
..       …         …         …        …        …         …         …
196  0.92949  1.793000 -1.183200 -0.11118 -0.65677  1.879700 -0.896080
197 -2.81980 -2.081000 -0.044753  1.11800 -1.64950  0.678670  0.494320
198  0.25879  0.603660  2.221000 -1.65410  0.68041  0.135830 -0.037987
199 -0.47897 -0.519100 -0.294660 -0.16055  1.02350  0.015895 -0.482660
200  1.30560  0.983970 -1.251400 -0.17975 -0.74341  0.233240  2.101300

           7         8        9       10        11       12        13  \
0    0.34262  3.578400  2.76940 -1.34990  3.034900  0.72540 -0.063055
1   -0.30344  0.293870 -0.78728  0.88840 -1.147100 -1.06890 -0.809500
2    0.62771  1.093300  1.10930 -0.86365  0.077359 -1.21410 -1.113500
```

```
3   -0.74230 -1.061600  2.35050 -0.61560  0.748080 -0.19242  0.888610
4   -0.24372  0.215670 -1.16580 -1.14800  0.104870  0.72225  2.585500
..       …         …        …        …        …         …         …
196  1.33150 -0.624570  0.77668 -1.32430  1.500300 -2.20290  0.322510
197 -0.58847 -0.023925  2.19130 -1.40060  0.480100  0.20755  0.322080
198 -0.69314 -0.127530  0.68036  0.42621 -1.605700  0.90621  0.249510
199  0.63467 -1.364400  0.59862 -0.20650  2.139400 -0.64881  0.423590
200 -0.87667  1.948800 -0.46527 -0.65192  0.609630  0.70912  0.279800

           14        15         16        17         18         19
0    0.714740 -0.20497 -0.124140  1.48970  1.409000  1.417200
1   -2.944300  1.43840  0.325190 -0.75493  1.370300 -1.711500
2   -0.006849  1.53260 -0.769670  0.37138 -0.225580  1.117400
3   -0.764850 -1.40230 -1.422400  0.48819 -0.177380 -0.196050
4   -0.666890  0.18733 -0.082494 -1.93300 -0.438970 -1.794700
..        …        …         …        …        …         …
196  0.421000 -0.82070 -0.966460  0.60935 -1.036200 -0.028238
197 -0.005881  0.28139 -0.251340 -1.69350 -0.584020  0.234440
198  1.894200 -1.11240 -0.722040 -1.75220  0.003612  0.995140
199  0.117870 -1.08080  0.875530 -2.22630 -1.946100 -1.309600
200 -1.467500 -0.69132 -0.867990 -0.51251 -0.782090  0.425790

[201 rows x 20 columns]
```

1. What is the sampling rate of this recording?

0.1 Answer:

[4]:
```
time_range = 10  #-5s to +5s
sampling_rate = total_intervals / time_range
print('Sampling Rate: ', sampling_rate)
```

```
Sampling Rate: 20.0
```

2. What is the total duration, T, of the recording?

[5]:
```
T = time_range
T
```

[5]:
```
10
```

3. Using the Nyquist sampling theorem, what is the maximum frequency (F_{max}) which can be accurately represented with this sampling rate?

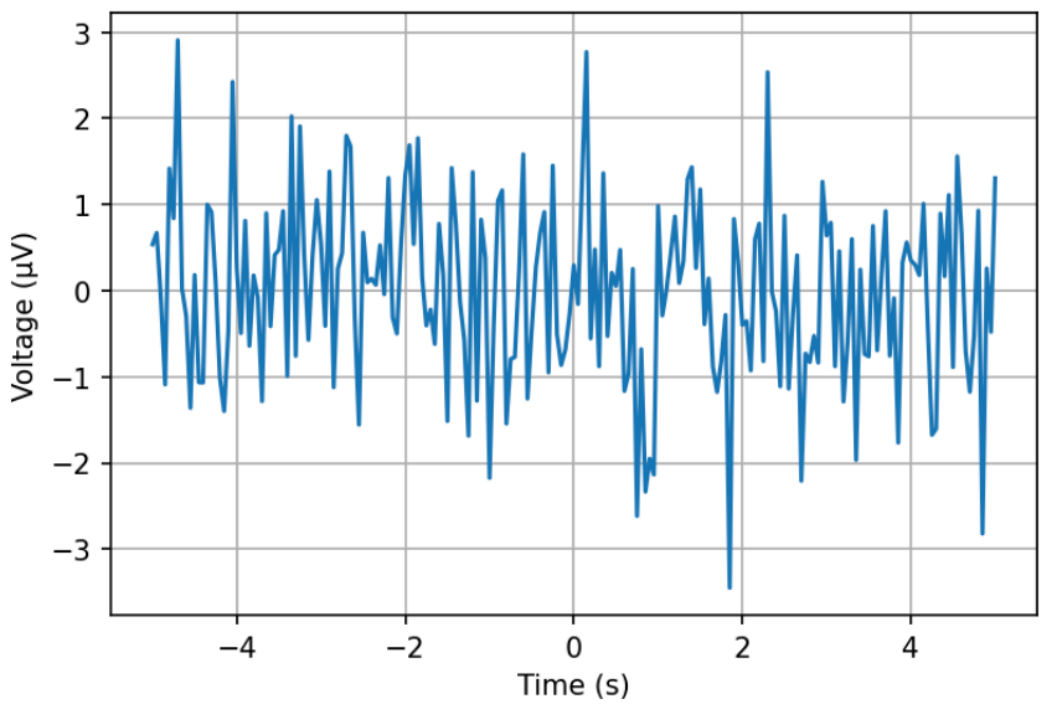
[6]:
```
f_max = sampling_rate / 2
f_max
```

[6]:
```
10.0
```

Next, make a plot (using Python) showing the voltage on the first trial (i.e. the first column of the dataset) as a function of time. Be sure to label the axes and indicate the units (time in seconds, voltage in microvolts µV) . You don't need to submit your plot or code; you will just use it to answer the following questions.

Hint: If you use Python np.arange, note that np.arange(t1,t2) does include the final time point (t2). Thus np.arange(0,5) is [0,1,2,3,4]. Thus, to make a vector of integers running from t1 to t2 (inclusive), you could use np.arange(t1,t2+1).

Your plot should like this:



4. What is the minimum voltage recorded during the pre-stimulus period (i.e. in the 5 s before the stimulus is presented)?

0.2 Answer:

```
[7]: sample_times = np.linspace(-5, 5, num = samples)
     pre_stimulus = df[sample_times < 5]
     minimum_voltage = pre_stimulus.min().min()
     minimum_voltage
```

```
[7]: -3.4915
```

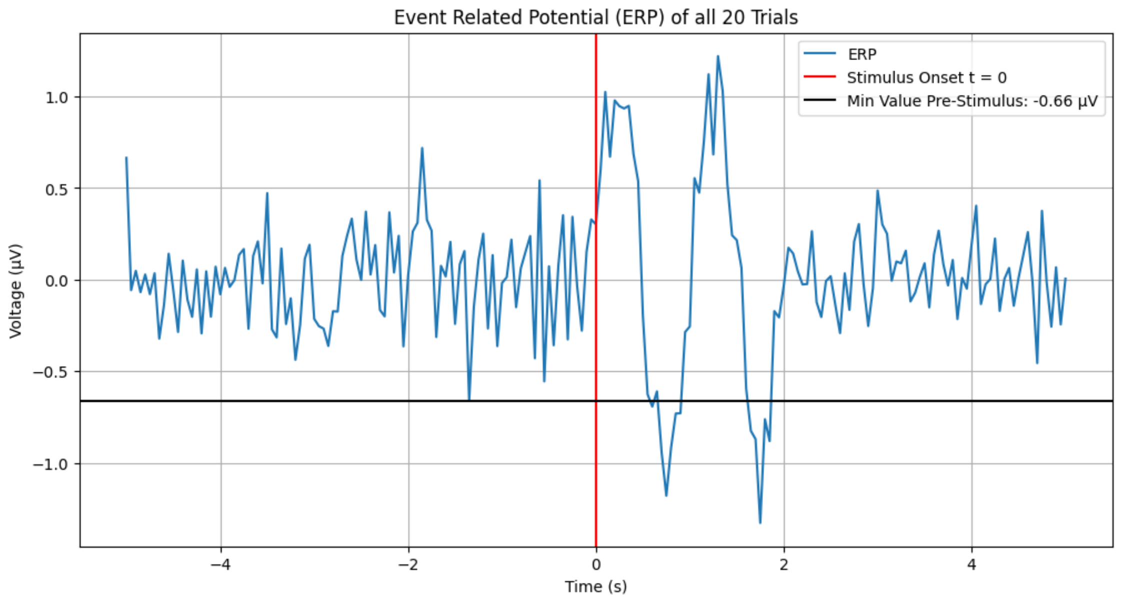
5. Plot the average of all 20 trials (voltage vs. time). This is the "Event Related Potential" (ERP). What is the approximate minimum value of the ERP during the pre-stimulus period?

0.3 Answer:

```
[8]: average_of_trials = df.mean(axis = 1)
     erp_pre_stimulus = average_of_trials[sample_times < 0].min()
     erp_pre_stimulus
```

```
[8]: -0.6616441
```

```
[31]: plt.figure(figsize=(12, 6))
      plt.plot(sample_times, average_of_trials, label = 'ERP')
      plt.xlabel('Time (s)')
      plt.ylabel('Voltage (µV)')
      plt.title('Event Related Potential (ERP) of all 20 Trials')
      plt.axvline(x = 0, color = 'red', linestyle = '-', label = 'Stimulus Onset t =␣
       ↪0')
      plt.axhline(y = erp_pre_stimulus, color = 'black', linestyle = '-', label =␣
       ↪f'Min Value Pre-Stimulus: {erp_pre_stimulus:.2f} µV')
      plt.legend()
      plt.grid(True)
      plt.show()
```



```
[31]: -0.6616441
```

6. What is the range of the voltage (maximum value – minimum value) during the post-stimulus period (i.e. during the 5 s following the stimulus presentation)?

```
[9]: post_stimulus = average_of_trials[sample_times > 0]
     maximum_voltage = post_stimulus.max()
```

```
minimum_voltage = post_stimulus.min()
range_of_voltage = maximum_voltage - minimum_voltage
range_of_voltage
```

```
[9]: 2.547851
```

7. Which of the following statements is a valid interpretation of your plots?

```
[10]: trial_one = df.iloc[:, 0]
amplitude_of_trial_one = trial_one.max() - trial_one.min()

print('\nAmplitude of the ERP: ', range_of_voltage)
print('\nAmplitude of the Raw Signal from Trial 1: ', amplitude_of_trial_one)

if range_of_voltage < amplitude_of_trial_one:
    print('\nThe amplitude of the ERP (max-min value) is smaller than the␣
 ↪amplitude of the raw signal from trial 1 because averaging reduces the␣
 ↪effect of noise.')
else:
    print('\nThe amplitude of the ERP is not smaller than the amplitude of the␣
 ↪raw signal from trial 1 because...')
```

```
Amplitude of the ERP:  2.547851

Amplitude of the Raw Signal from Trial 1:  6.3582

The amplitude of the ERP (max-min value) is smaller than the amplitude of the
raw signal from trial 1 because averaging reduces the effect of noise.
```

0.4 Answer:

The amplitude of the ERP (max-min value) is smaller than the amplitude of the raw signal from trial 1 because averaging reduces the effect of noise.

8. Which of the following statements is a valid interpretation of your plots?

0.5 Answer:

The signal to noise ratio (SNR) is greater for the ERP than for the raw data on trial 1 because the noise is reduced.

Notes: - ERP is the average of all trials which reduces random noise in individual trials. Averaging does not amplify actual signal, which will make amplitdue of ERP smaller compared to raw signal because of the noise reduction.

0.5.1 From hw1:

SNR Formula: SNR $(dB) = 20\log_{10}\left(\frac{\text{Total Signal RMS}}{\sigma\,(\text{standard deviation})}\right) = 20\log_{10}\left(\frac{\frac{A}{\sqrt{2}}\cdot\sqrt{5000\cdot r}}{10}\right) = 20\log_{10}\left(\frac{\frac{0.01}{\sqrt{2}}\cdot\sqrt{5000\cdot r}}{10}\right) = 20\left(\log\left(\frac{0.01}{\sqrt{2}}\cdot\sqrt{5000}\right)+\log\left(10\right)\right) = \log\left(\frac{r^{10}}{1048576}\right) - 20$

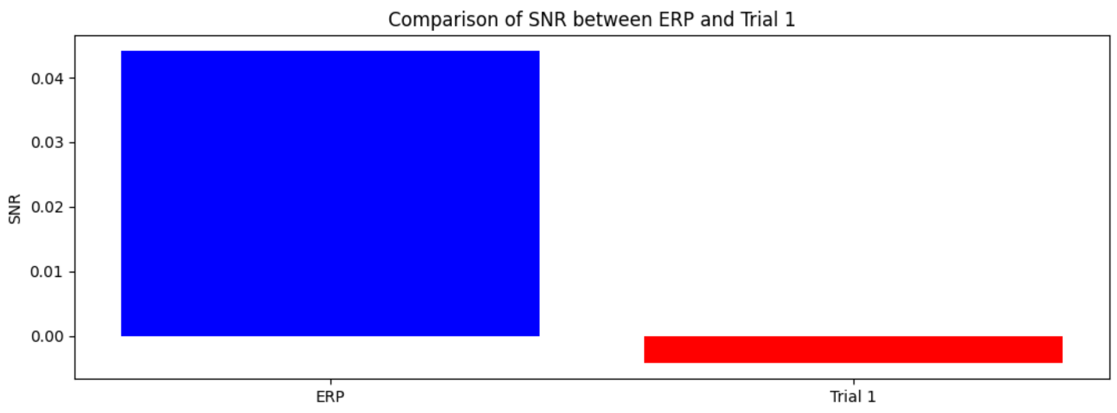
"'python neurons = 5000 sd = 10 A = 0.05 #original is A = 0, 0.05 is to not divide by 0 responsive_neurons_r = np.linspace(0, 1, 800) total_signal_RMS = (A / np.sqrt(2)) * np.sqrt(neurons * responsive_neurons_r) SNR = 20 * np.log10(total_signal_RMS / sd)

plt.figure(figsize = (8, 4)) plt.plot(neurons * responsive_neurons_r, SNR) plt.xlabel('Number of Responsive Neurons') plt.ylabel('SNR (dB)') plt.title('Signal-to-Noise Ratio as a Function of Responsive Neurons') plt.grid(True) plt.show()

```
[11]: #ERP SNR
mean_for_erp = average_of_trials.mean()
std_for_erp = average_of_trials.std()
snr_for_erp = mean_for_erp / std_for_erp

#Trial 1 SNR
mean_for_trial1 = trial_one.mean()
std_for_trial1 = trial_one.std()
snr_for_trial1 = mean_for_trial1 / std_for_trial1

print('\nSNR for ERP: ', snr_for_erp)
print('\nSNR for Trial 1: ', snr_for_trial1)

if snr_for_erp > snr_for_trial1:
    print('\nThe signal to noise ratio (SNR) is greater for the ERP than for␣
 ↪the raw data on trial 1 because the noise is reduced.')
else:
    print('nThe signal to noise ratio (SNR) is not greater for the ERP than for␣
 ↪the raw data on trial 1 because ...')

#side notes:
# neurons = ?
# A = ?
# sample_times = np.linspace(-5, 5, num = samples)
# total_signal_RMS = (A / np.sqrt(2)) * np.sqrt(neurons * sample_times)
# SNR = 20 * np.log10(total_signal_RMS / std_for_erp)
```

```
SNR for ERP:  0.04408699550554319

SNR for Trial 1:  -0.00425915551396173

The signal to noise ratio (SNR) is greater for the ERP than for the raw data on
trial 1 because the noise is reduced.
```

[28]:
```python
snr_values = [snr_for_erp, snr_for_trial1]
labels = ['ERP', 'Trial 1']

plt.figure(figsize = (12, 4))
plt.bar(labels, snr_values, color = ['blue', 'red'])
plt.ylabel('SNR')
plt.title('Comparison of SNR between ERP and Trial 1')
plt.show()
```



[13]:
```python
time_point_tzero = np.where(sample_times == 0)[0][0]
erp_for_tzero = df.iloc[time_point_tzero]
t_statistic, p_value = ttest_1samp(erp_for_tzero, 0)

print('T-Static: ', t_statistic)
print('P-Value: ', p_value)

if p_value > snr_for_trial1:
    print('\nERP mean voltage at time t=0 is not significantly different from 0␣
↪and null hypothesis ERP = 0 is not rejected.')
else:
    print('\nERP mean voltage at time t=0 is significantly different from 0 and␣
↪null hypothesis ERP = 0 is rejected.')
```

```
T-Static:  1.2400628478672049
P-Value:  0.23004918230685234

ERP mean voltage at time t=0 is not significantly different from 0 and null
hypothesis ERP = 0 is not rejected.
```

9. Use a t-test to determine whether the ERP is statistically significant at the time point t=0, (the 100th sample or the 101th sample will also be accepted). That is, test the null hypothesis that the ERP = 0. What is the p-value?

Hint: You can perform a t-test using scipy.stats.ttest_1samp Links to an external site..

This is “1-sample t-test” because we are comparing the 20 trials with a population mean of 0. This function returns two values (the t-statistic, and the p-value).

What is the p-value? Provide your answer with 2 significant figures, e.g. 0.92 instead of 0.9 or 0.92314.

“‘python scipy.stats.ttest_1samp(a, popmean, axis=0, nan_policy=‘propagate’, alternative=‘two-sided’, *, keepdims=False)

0.6 Answer:

P-value = 0.23004918230685234, the null hypothesis that the ERP = 0 not rejected

10. Based on the p-value you found for t=0 s, is the mean voltage at the time of the stimulus onset significantly different from 0? That is, can we reject the null hypothesis?

0.7 Answer:

No, because the p-value is >0.05. This makes sense because there has not been time for the stimulus to influence brain activity. P-value = 0.23

Repeat the t-test at each of the 201 time points and store the p-value for each time point in a vector (list).

Make a plot showing the p-value as a function of time. Be sure to use a log-scale for the y-axis [Hint: plt.yscale(‘log’) Links to an external site.], which will help to see the magnitude of very small p-values. Add a line to the plot showing the threshold p=0.05 [Hint: plt.hlines Links to an external site.]

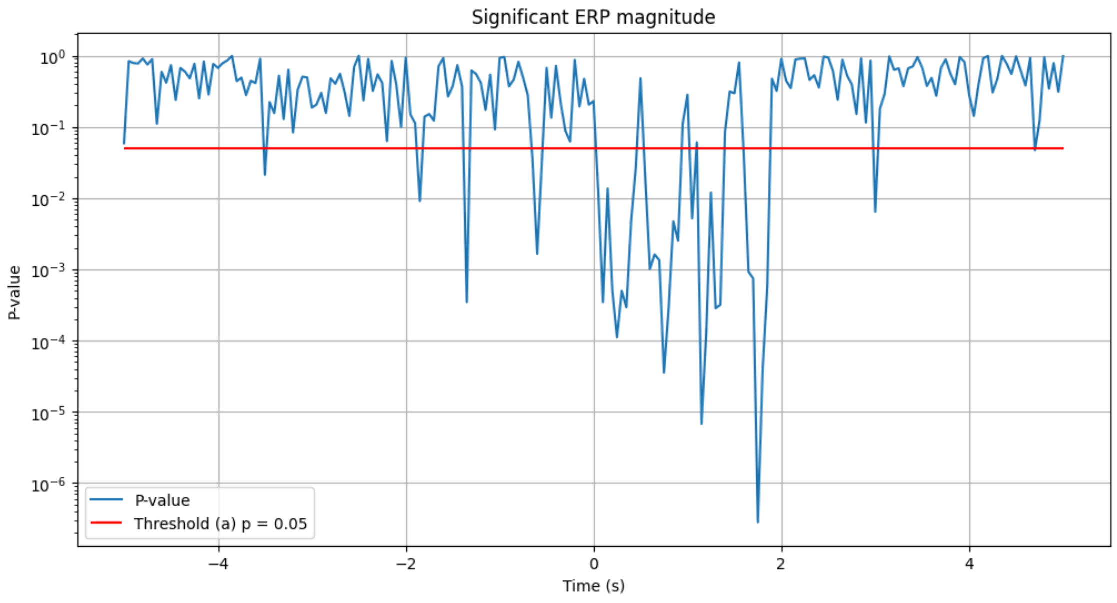
11. How many of the 201 timepoints have a significant ERP magnitude? For this question use the threshold p=0.05 (also called $a = 0.05$).

0.8 Answer:

37 timepoints have a significant ERP magnitude

[29]:
```
p_values = []

for index in range(samples):
    t_statistic, p_value = ttest_1samp(df.iloc[index], 0)
    p_values.append(p_value)

significant_timepoints = np.sum(np.array(p_values) < 0.05)

print('Timepoints that have a significant ERP magnitude',␣
 ↪significant_timepoints)

plt.figure(figsize = (12, 6))
plt.plot(sample_times, p_values, label = "P-value")
plt.yscale('log')
plt.hlines(0.05, xmin = -5, xmax = 5, colors = 'red', linestyles = '-', label =␣
  ↪'Threshold (a) p = 0.05')
plt.xlabel('Time (s)')
plt.ylabel('P-value')
plt.title('Significant ERP magnitude')
plt.legend()
plt.grid(True)
plt.show()
```

```
Timepoints that have a significant ERP magnitude 37
```



12. Consider the 100 pre-stimulus time points (t=-5 to 0). How many of these 100 have a significant ERP (using significance threshold 0.05)?

Hint: You can extract a subset of time points from a vector (numpy array) in Python by slicing. For example, if

t=np.array([-2,-1,0,1,2])

and pvals is a vector of p-values at each of these time points:

pvals=np.array([0.02,0.01,0.1,0.2,0.3])

then

pvals[t<0] = [0.02,0.01]

0.9 Answer:

6 Significant time points ERP pre-stimulus

[15]:
```python
pvalues_of_prestimulus100 = np.array(p_values)[:time_point_tzero]
significant_erp = np.sum(pvalues_of_prestimulus100 < 0.05)
print('Significant time points ERP pre-stimulus: ', significant_erp)
```

```
Significant time points ERP pre-stimulus:  6
```

13. Consider the 100 post-stimulus time points (t>0). How many of these have a significant ERP (using significance threshold 0.05)?

0.10 Answer

> 31 Significant time points ERP post-stimulus

[16]:
```python
significant_pre_stimulus = np.sum(significant_erp < 0.05)
pvalues_of_poststimulus100 = np.array(p_values)[time_point_tzero + 1:]
significant_erp_post = np.sum(pvalues_of_poststimulus100 < 0.05)
print('Significant time points ERP post-stimulus: ', significant_erp_post)
```

```
Significant time points ERP post-stimulus:  31
```

14. Recall that the p-value is defined as the probability that the data would be observed if the null hypothesis were true. In this case, that means that the p-value is the probability of observing the data if the average ERP is 0 and each of the 20 data points (across trials) is a Gaussian-distributed (normal) random noise value.

During the pre-stimulus period, the null hypothesis (ERP=0) is true by definition: The stimulus has not yet been presented, so the average ERP must be 0.

What is the average number of false positives you would expect to detect during the pre-stimulus period using a p-value threshold 0.05?

0.11 Answer:

> 5 > False Positives Expected during pre-stimulus period = time_points x False Positive Probability = 100% x 0.05 = 5%

15. What is the reason that we expect to have false positive detections?

0.12 Answer:

> We are testing multiple hypotheses (100 time points), and each time point has a 5% chance of a false-positive detection. > Notes: False positives are expected because of the statistical chance of incorrectly rejecting null hypothesis when making multiple comparisons

16. The Bonferroni correction Links to an external site. procedure adjusts the statistical significance threshold to a lower (more stringent) value to reduce the risk of false positives. Using the Bonferroni procedure, what threshold should we use to ensure there is 5% chance of finding even a single false positive? In other words, there is a 95% chance of having no false positives.

0.13 Answer:

$0.0002487562189054727 = 0.0002$

```
[17]: bonferroni_threshold = 0.05 / samples
      bonferroni_threshold
```

```
[17]: 0.0002487562189054727
```

```
[18]: p_values = [ttest_1samp(df.iloc[i], 0)[1] for i in range(samples)]
      significant_erp_pre_bonferroni = np.sum((np.array(p_values)[:time_point_tzero]␣
       ↪< bonferroni_threshold))
      significant_erp_post_bonferroni = np.sum(pvalues_of_poststimulus100 <␣
       ↪bonferroni_threshold)

      print('\nTime points during the pre-stimulus period (t<0) that have a␣
       ↪significant ERP (i.e. ERP 0) according to this Bonferroni-corrected criterion:
       ↪ ', significant_erp_pre_bonferroni)
      print('\nTime points during the post-stimulus period (t>0) that have a␣
       ↪significant ERP (i.e. ERP 0) according to this Bonferroni-corrected criterion:
       ↪ ', significant_erp_post_bonferroni)
```

```
Time points during the pre-stimulus period (t<0) that have a significant ERP
(i.e. ERP 0) according to this Bonferroni-corrected criterion:  0

Time points during the post-stimulus period (t>0) that have a significant ERP
(i.e. ERP 0) according to this Bonferroni-corrected criterion:  6
```

17. How many time points during the pre-stimulus period (t<0) have a significant ERP (i.e. ERP 0) according to this Bonferroni-corrected criterion?

0.14 Answer:

0

18. How many time points during the post-stimulus period (t>0) have a significant ERP (i.e. ERP 0) according to this Bonferroni-corrected criterion?

0.15 Answer:

6