Statistical Analysis

Mariella Paul
paulm@cbs.mpg.de

MPI for Human and Cognitive Brain Sciences
Berlin School of Mind & Brain

Ways to do ERP statistics

- good way 1: hypothesis-driven
- good way 2: data-driven with correction for multiple testing
- “the bad way”: data-driven, no correction for multiple testing

Hypothesis-driven approach

- if you have a hypothesis about a specific ERP component, you can decide to only analyze the relevant time window / electrodes **a priori**
- this significantly reduces the amount of multiple testing
- e.g. if you expect an N400, you might choose to only analyze 300 – 600 ms TW and centro-parietal electrodes

Data-driven approach

- if you do not have an a priori hypothesis about time windows or channels, you should analyze all **channel x time pairs**
- with this approach, you run into the **multiple comparison problem**

The multiple comparison problem

The multiple comparison problem (MCP) applies to all kinds of data when you want to run more than one dependent statistical analysis

MCP: example

- assume you have accuracy data from an experiment with 3 different conditons, and you want to statistically compare the accuracies of each conditon
- you decide to run three tests: cond 1 vs cond 2, cond 2 vs cond 3 and cond 1 vs cond 3
- because these tests are dependent, you need to adjust your alpha
- **Bonferroni correction**: alpha/# of tests
- in this case: 0.05/3 = 0.017

MCP with EEG data

in EEG data we have A LOT of dependent data

- usually hundreds of trials (let's assume 100)
- consisting of time samples (let's assume 250)
- measured on dozens channels (let's assume 30)
- we would end up with 100*250*30 = 750 000 tests
- and with **Bonferroni correction**, would have to adjust alpha to 0.05 / 750 000 = 0.0000667 -> **no effect would ever be significant**!

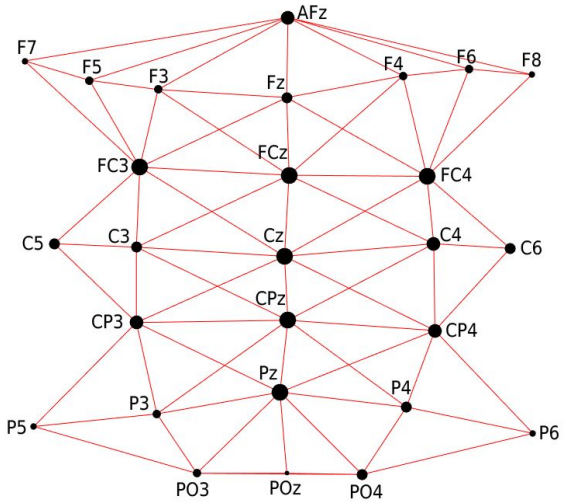
Cluster-based permutation test (CBPT)

- the cluster-based permutation test (Maris and Oostenveld, 2007) solves the MCP in an elegant way
- it creates clusters of channel x time pairs
- effects are only considered to be significant if they are **significant over several channel x time pairs**, i.e. if they last several milliseconds and are distributed over several electrodes

Maris and Oostenveld (2007) *J. Neurosci. Methods*

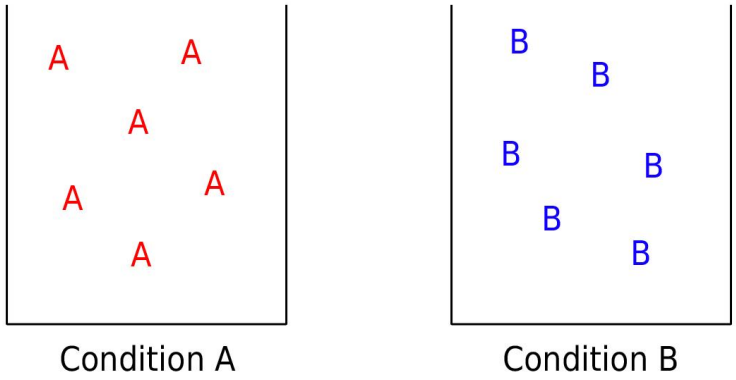
CBPT: real t-statistic

1. define neighborhood relationships for electrodes (needed for clustering)



CBPT: real t-statistic

1. define neighborhood relationships for electrodes (needed for clustering)
2. for each condition, collect all trials

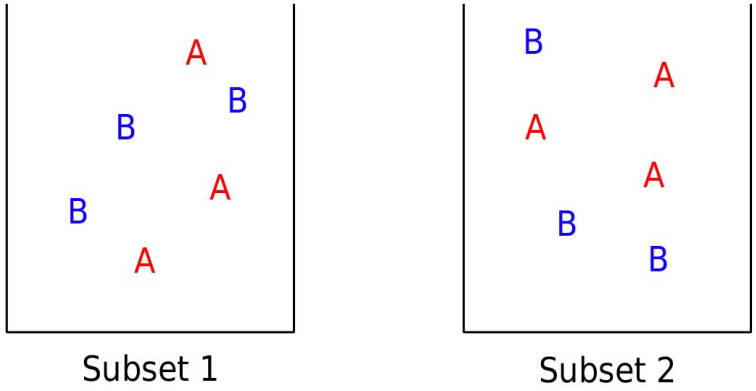


CBPT: real t-statistic

1. define neighborhood relationships for electrodes (needed for clustering)
2. for each condition, collect all trials
3. calculate t-values for channel-time pairs for these conditions and sum them over clusters (*real t-statistic*)

CBPT: random t-statistic

1. create a subset 1 and randomly draw as many trials from both conditions as there are in condition A; put the rest of the trials in subset 2



CBPT: random t-statistic

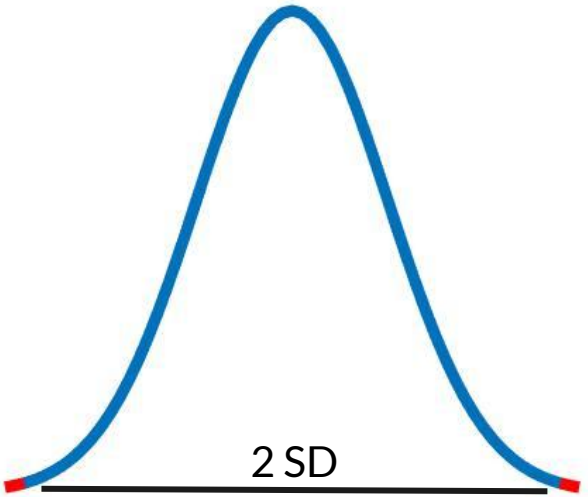
1. create a subset 1 and randomly draw as many trials from both conditions as there are in condition A; put the rest of the trials in subset 2
2. calculate t-values for cluster-time pairs for these subsets and sum them over clusters (random t-statistic)

CBPT: random t-statistic

1. create a subset 1 and randomly draw as many trials from both conditions as there are in condition A; put the rest of the trials in subset 2
2. calculate t-values for cluster-time pairs for these subsets and sum them over clusters (random t-statistic)
3. repeat steps 1 and 2 a large number of times (resulting many random t-statistics)

CBPT: compute significance

compare real t-statistic with random t-statistics:
if real t-statistic is 2 or more standard deviations (SD) away from the median of random t-statistics, consider the real t-statistic to be significant



t-statistics are normally distributed

„the bad way“: visual inspection

- some papers only analyze certain time windows based on “visual inspection”
- this usually means they looked at their data and chose the time window that looked like it might contain a significant effect

What's the problem with this approach?

CBPT in FieldTrip

- we'll perform a cluster-based permutation test in FieldTrip using ft_timelockstatistics
- but first, which conditions do we want to compare?

Open the FieldTrip Tutorial on CBPT to work along

Neighborhood

- the first step of the cluster-based permutation test involves defining neighborhoods
- let's do this in FieldTrip using `ft_prepare_neighbours` with `cfg.method = 'triangulation'`

Specify the cfg

- before we use `ft_timelockstatistics`, we need to set up the cfg. Specify:
  - the channels
  - the latency
- `cfg.method` determines how we create the random t-statistics
- we'll use `cfg.method = 'montecarlo'`
- this determines what other cfg options we have
- look at the reference documentation of `ft_statistics_montecarlo` to see what our options are

Specify the cfg

- `cfg.correctm` determines the way ft_timelockstatistics corrects for multiple testing and should be `'cluster'`
- `cfg.minnbchan` determines the minimum number of channels to be included in a cluster; we'll use 2 since we have relatively few electrodes
- `cfg.neighbours`- this is where you should specify the neighborhood structure we defined earlier
- `cfg.tail` defines one- or two-sided test; we want a two-sided test

Specify the cfg

- `cfg.alpha` determines the alpha value of the statistical test **per tail**. Given that we're running a two-sided test, what alpha do we need?
- `cfg.numrandomization`determines how many random t-statistics are computed; this should be a large number, for now, let's take a 1000
- what settings are there for cfg.statistics and which one do we want?
    - dependent samples are within-subject, independent samples between-subject

Design config

- in `cfg.design`, you need to specify a matrix describing the design:
- e.g. imagine an experiment with 4 trials in the first condition and 3 trials in the second condition:

  ```
  design = [1 1 1 1 2 2 2]
  ```
- use the `trial` field of your data structures to compute the number of trials in each condition

ft_timelockstatistics and plotting

- now we've finally got the cfg set up and can use `ft_timelockstatistics`!
- you can check the fields `poscluster` and `negcluster` of the output of `ft_timelockstatistics` to see if there are any significant clusters
- plot the results by adapting the code for plotting at the bottom of the FieldTrip Tutorial on CBPT
- as a last step, use `ft_analysispipeline` to output the analysis pipeline