Bootstrapping and multiple comparison corrections in EEG

Literate script to demonstrate how the bootstrap works and how we can use it for multiple comparison correction in EEG. Some of the code relies on LIMO EEG. A large part was inspired by G. Rousselet tutorial on bootstrap (in R). The code below (and data) can be found here.

Cyril Pernet - June 2019

The bootstrap is a well-established early computer-age inferential method (Efron, 1979; Efron & Hastie, 2016; Efron & Tibshirani, 1994). The bootstrap is based on the idea that using only the data at hand can sometimes give better results than making unwarranted assumptions about the populations we're trying to estimate. The core mechanism of the bootstrap is sampling with replacement from the data, which is a form of data-driven simulation.

The bootstrap idea

The idea is to make inferences using the data at hand only, avoiding making assumptions about the underlying distribution, observations are coming from by sampling with replacement the data.

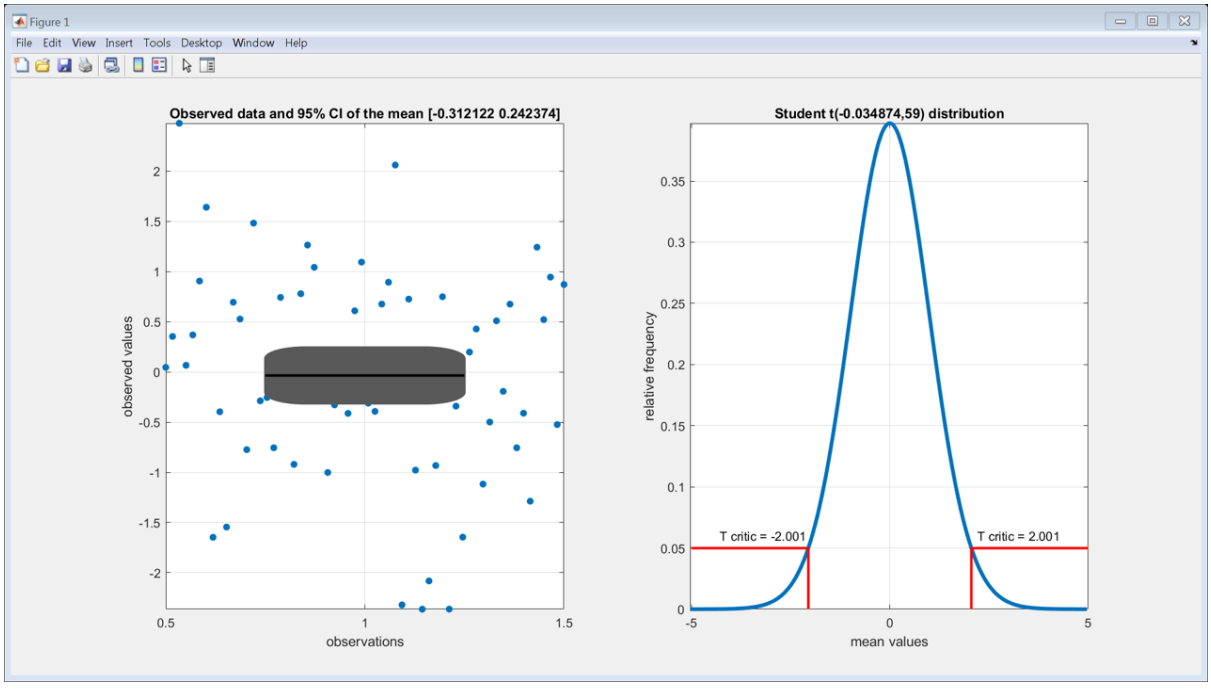
```
% imagine we do an experiment, and have 60 observations
clear variables
N = 60; obs = randn(N,1);

% We can now compute the mean
avg_obs = mean(obs);
```

The theory tells us that, for normally distributed data, the mean follows a Student t-distribution with N-1 degrees of freedom. A confidence interval (CI) for the mean is an interval that covers the population mean with a of error alpha, here 5% for a 95%CI (Morey ey al. 2016)

```
alpha_value = 5/100;
Tcritic     = tinv(alpha_value/2,N-1); % divide by two 2.5% each side of the distribution
se          = std(obs)/sqrt(N); % standard error
CI          = [avg_obs-abs(Tcritic)*se avg_obs+abs(Tcritic)*se]; % same as
[~,~,CI]=ttest(obs)
```

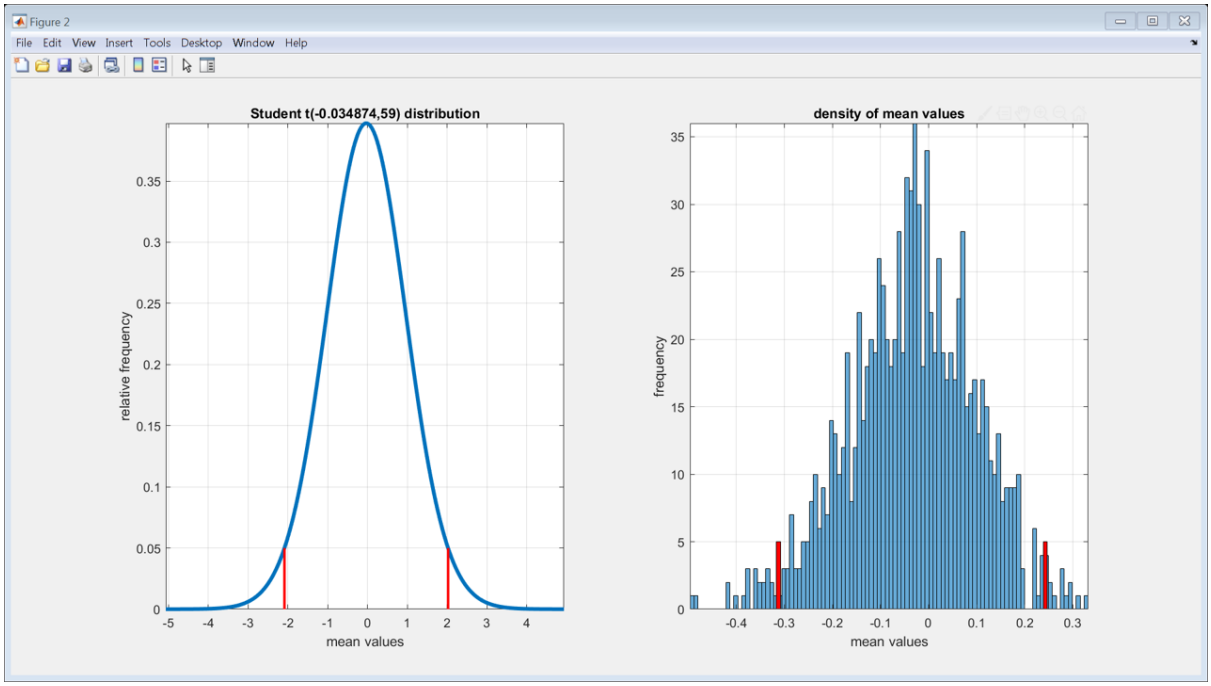
Let's illustrate what it means

```
figure('units','normalized','outerposition',[0 0 1 1])
subplot(1,2,1); scatter([0.5:1/(N-1):1.5],obs,30,'filled');
box on; grid on; axis tight; xlabel('observations'); ylabel('observed values')
hold on; rectangle('Position',[0.75 CI(1) 0.5 diff(CI)],'Curvature',[0.4
0.4],'LineWidth',2,...
    'FaceColor',[0.35 0.35 0.35],'EdgeColor',[0.35 0.35 0.35]);
plot([0.75:0.1:1.25],repmat(avg_obs,1,6),'k','LineWidth',2);
title(sprintf('Observed data and 95%% CI of the mean [%g %g]',CI(1),CI(2)))

subplot(1,2,2);
sample = [-5:.01:5]+avg_obs;
theoretical_distribution = tpdf(sample,N-1);
plot(sample,theoretical_distribution,'LineWidth',3)
box on; grid on; axis tight; xlabel('mean values'); ylabel('relative frequency')
title(sprintf('Student t(%g,%g) distribution',avg_obs,N-1))
hold on; [~,position] = min(abs(theoretical_distribution-alpha_value));
plot([-5:.01:sample(position)],repmat(theoretical_distribution(position),1,...
    length([-5:.01:sample(position)])),'r','LineWidth',2);
plot([5:-.01:abs(sample(position))],repmat(theoretical_distribution(position),1,...
    length([5:-.01:abs(sample(position))])),'r','LineWidth',2);
plot(repmat(sample(position),1,6),[0:.01:alpha_value],'r','LineWidth',2);
plot(repmat(abs(sample(position)),1,6),[0:.01:alpha_value],'r','LineWidth',2);
text(-4.3,0.06,['T critic = ' num2str(Tcritic)]);
text(2.2,0.06,['T critic = ' num2str(abs(Tcritic))]);
```



We can compute the same thing without assuming a Student t distribution and simply computing the distribution of sample means, as if we were collecting data from many experiments - ie by bootstrapping.

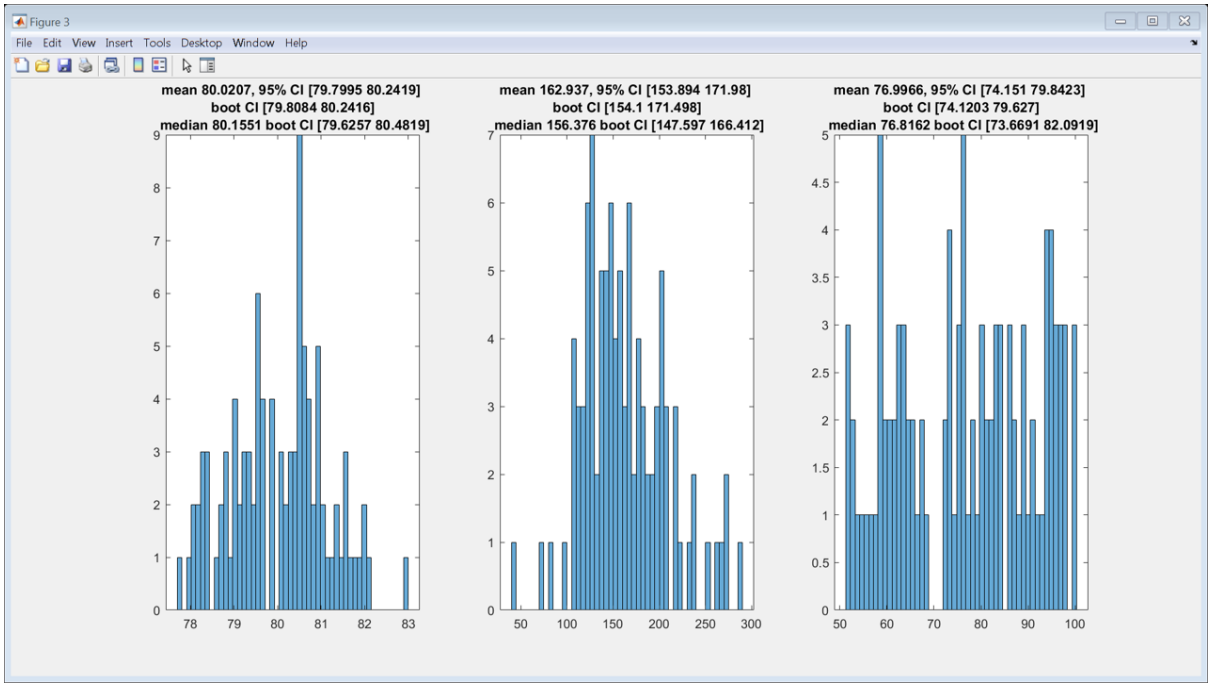
```
Nboot     = 1000;
resamples = randi(N,N,Nboot); % pick at random out of the N observations
avg_boot  = sort(mean(obs(resamples))); % the Nboot means from sampling
CI_boot   = [avg_boot(alpha_value/2*Nboot) avg_boot(Nboot-alpha_value/2*Nboot)];

% Let's compare
figure('units','normalized','outerposition',[0 0 1 1])
subplot(1,2,1); plot(sample+mean(avg_boot),theoretical_distribution,'LineWidth',3)
box on; grid on; axis tight; xlabel('mean values'); ylabel('relative frequency')
hold on;
plot(repmat(sample(position)+mean(avg_boot),1,6),[0:.01:alpha_value],'r','LineWidth',2);
plot(repmat(-1*(sample(position))+mean(avg_boot),1,6),[0:.01:alpha_value],'r','LineWidth',2);
title(sprintf('Student t(%g,%g) distribution',avg_obs,N-1))
subplot(1,2,2); h = histogram(avg_boot, 100);
title('density of mean values'); xlabel('mean values'); ylabel('frequency')
hold on; bar(CI(1),5,h.BinWidth,'r');
bar(CI(2),5,h.BinWidth,'r');
axis tight; grid on; box on
```



Let's have a quick look at different distributions - and since this is that simple, let's get both the mean and the median confidence intervals

```
figure('units','normalized','outerposition',[0 0 1 1]);
resample = randi(100,100,1000);
for d=1:3
    if d == 1
        % normal data as before - does this really exist?
        data = sort(randn(100,1))+80;
    elseif d == 2
        % lognornal - classic reaction times
        No = normrnd((200+randn(1)*50),25*abs(randn(1)),[100,1]);
        Ex = exprnd(75,[100,1]) + (150+randn(1)*50);
        data = (No+Ex)/2;
```

```
    else
        % uniform
        unif = sort(rand(200,1)).*150;
        data = unif(end-100:end);
        data = data.*100/max(data); % pretend max is 100
    end
    boot = sort(mean(data(resample)));
    boot2 = sort(median(data(resample)));
    subplot(1,3,d); histogram(data,50); [~,~,tCI]=ttest(data);
    title(sprintf('mean %g, 95%% CI [%g %g] \n boot CI [%g %g] \n median %g boot CI [%g
%g]',...
        mean(data), tCI(1), tCI(2), boot(25), boot(975), median(data), boot2(25),
boot2(975)))

end
```

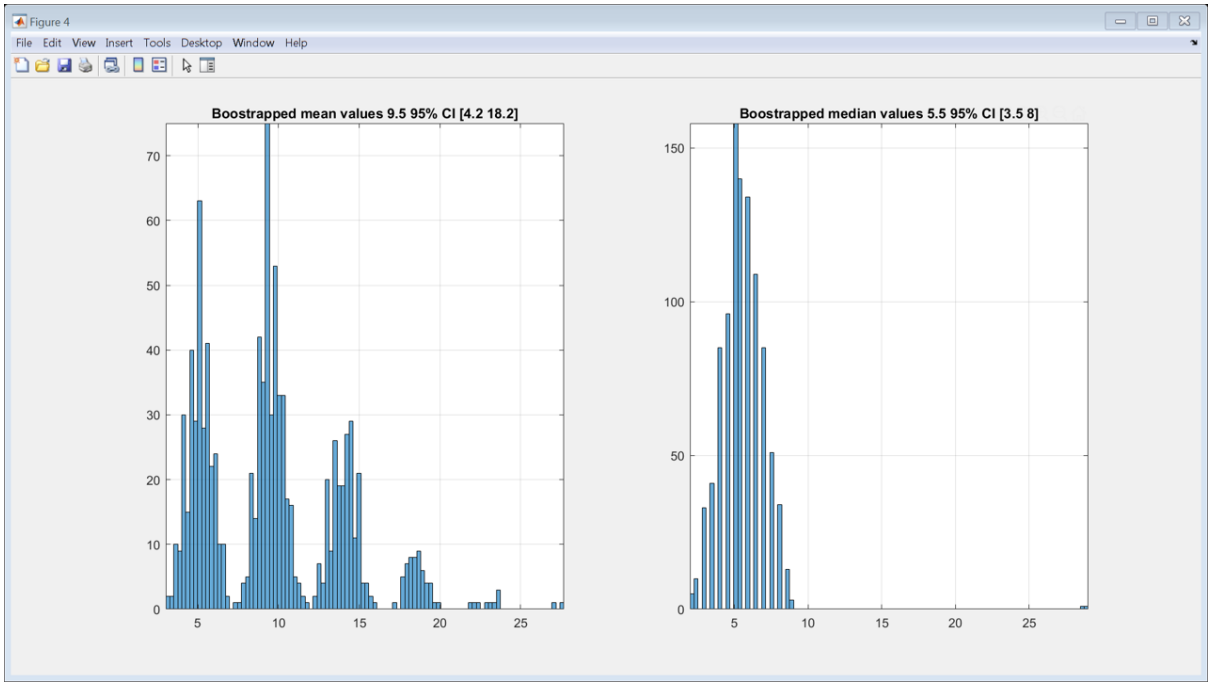


Bootstrap does not bring robustness

In statistics, the term robust means resistant to outliers. The mean is the least robust location estimator with a breakdown point of 0 while the median is the most robust with a breakdown point of 50% (i.e. doesn't change up to 50% of outliers).

```
A=[1:9 50]; Nboot = 1000;
% rather that completly random, the resampling can be constrained, here to
% at least 4 unique values ; which ensures some distribution in the data
index = 1;
while index<Nboot
    tmp = randi(10,10,1);
    if length(unique(tmp))>4
        bootsamples(:,index) = tmp;
        index = index+1;
    end
end
```

```
means = sort(mean(A(bootsamples)));
medians = sort(median(A(bootsamples)));
low = floor(alpha_value*Nboot); high = Nboot-low;
figure('units','normalized','outerposition',[0 0 1 1])
subplot(1,2,1); histogram(means, 100); axis tight; grid on; box on
title(['Boostrapped mean values ' num2str(mean(A)) ' 95% CI [' num2str(means(low)) ' '
num2str(means(high)) ']'])
subplot(1,2,2); histogram(medians, 100); axis tight; grid on; box on
title(['Boostrapped median values ' num2str(median(A)) ' 95% CI [' num2str(medians(low)) ' '
num2str(medians(high)) ']'])
```



Bootstrap is not assumption free

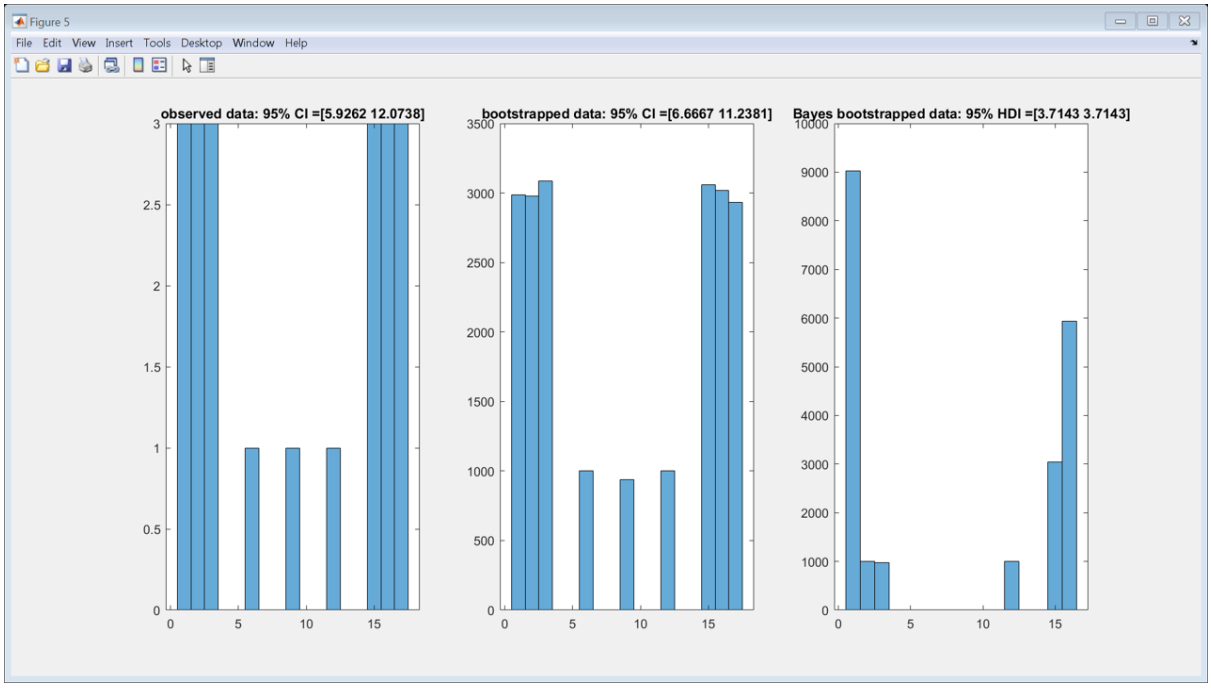
when we resample, we sample from the observed data only

```
A = [1 1 1 2 2 2 3 3 3 6 9 12 15 15 15 16 16 16 17 17 17];
resamples = randi(length(A),length(A),Nboot); % pick at random out of the N observations
```

We could, however, imagine that missing values exist, simply not observed the current sample! Using such priors is called Bayesian bootstrap Rubin (1981)

```
n = length(A);
Bayes_resamples = NaN(size(resamples));
for boot=1:Nboot % bootstrap loop
    theta = exprnd(1,[n,1]);
    weigths = theta ./ repmat(sum(theta),n,1);
    Bayes_resamples(:,boot) = (datasample(A,n,'Replace',true,'Weights',weigths));
end

% let's check CI
[~,~,tCI] = ttest(A); % from observed data, assume normality
tmp       = sort(mean(A(resamples))); % build your distribution of means
bootCI    = [tmp(low) tmp(high)]; % take 2.5% 97.5% of this distribution
```

```
tmp       = sort(mean(A(Bayes_resamples))); % sample from hypergeometric distribution
ci        = 1:low; % build floor(alpha_value*Nboot) CI
ciWidth   = tmp(ci+high) - tmp(ci); % all intervals in the range
[~,index] = find(ciWidth == min(ciWidth)); % densest centile
HDI       = [tmp(index) tmp(index+high)]; % highest density interval

% let's see how different this is
figure('units','normalized','outerposition',[0 0 1 1])
subplot(1,3,1); histogram(A);
title(['observed data: 95% CI =[' num2str(tCI(1)) ' ' num2str(tCI(2)) ']'])
subplot(1,3,2); all = A(resamples); histogram(all(:));
title(['bootstrapped data: 95% CI =[' num2str(bootCI(1)) ' ' num2str(bootCI(2)) ']'])
subplot(1,3,3); all = A(Bayes_resamples); histogram(all(:));
title(['Bayes bootstrapped data: 95% HDI =[' num2str(HDI(1)) ' ' num2str(HDI(2)) ']'])
```



application to ERP

```
electrode = 60;
limo_add_plots('cond1-2_single_subjects_Mean',1,electrode);
title('condition 1 - all subjects')
limo_add_plots('cond1-2_single_subjects_Mean',2,electrode);
title('condition 2 - all subjects')
```

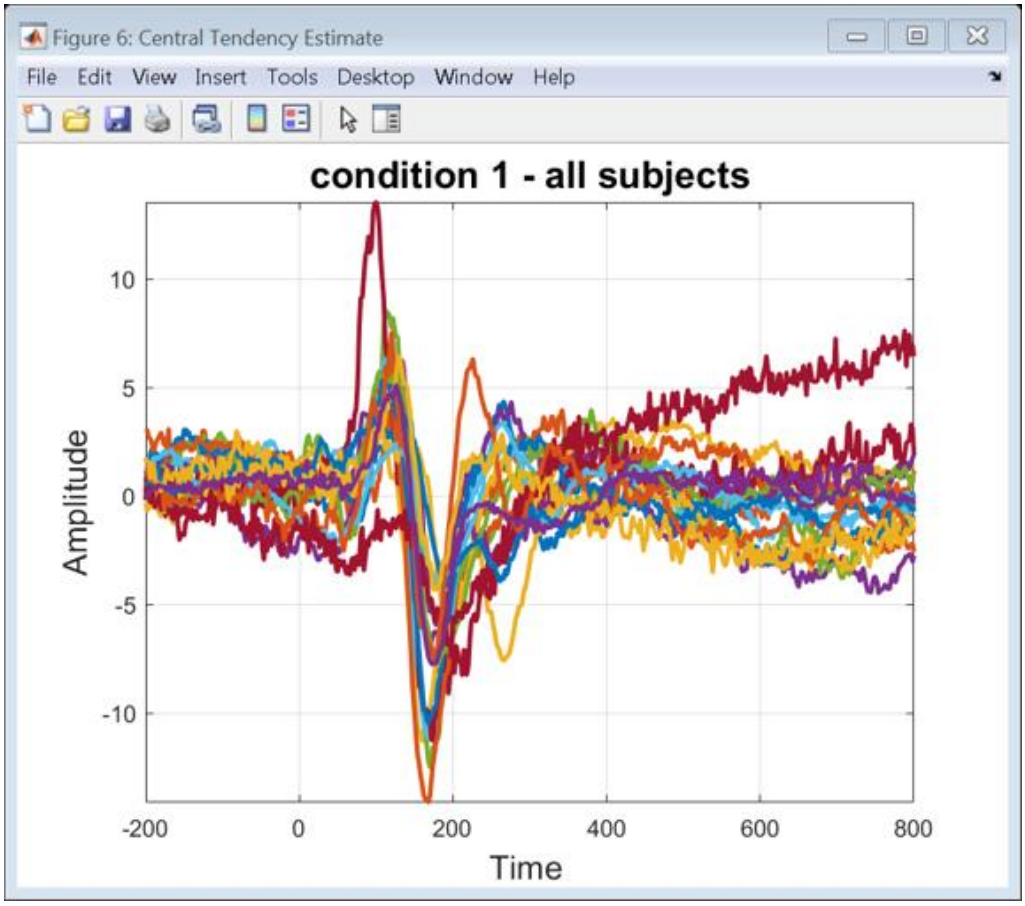


Figure 6: Central Tendency Estimate
File Edit View Insert Tools Desktop Window Help
condition 1 - all subjects
Amplitude
10
5
0
-5
-10
-200
0
200
400
600
800
Time

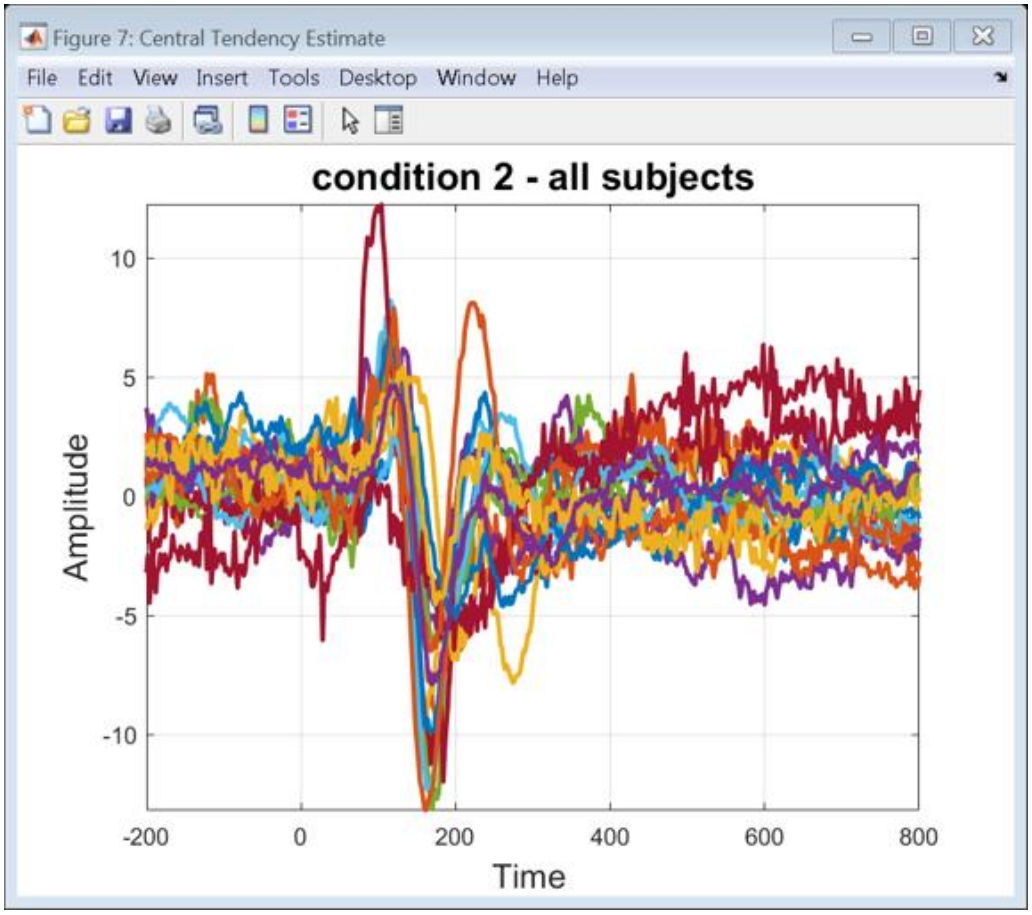
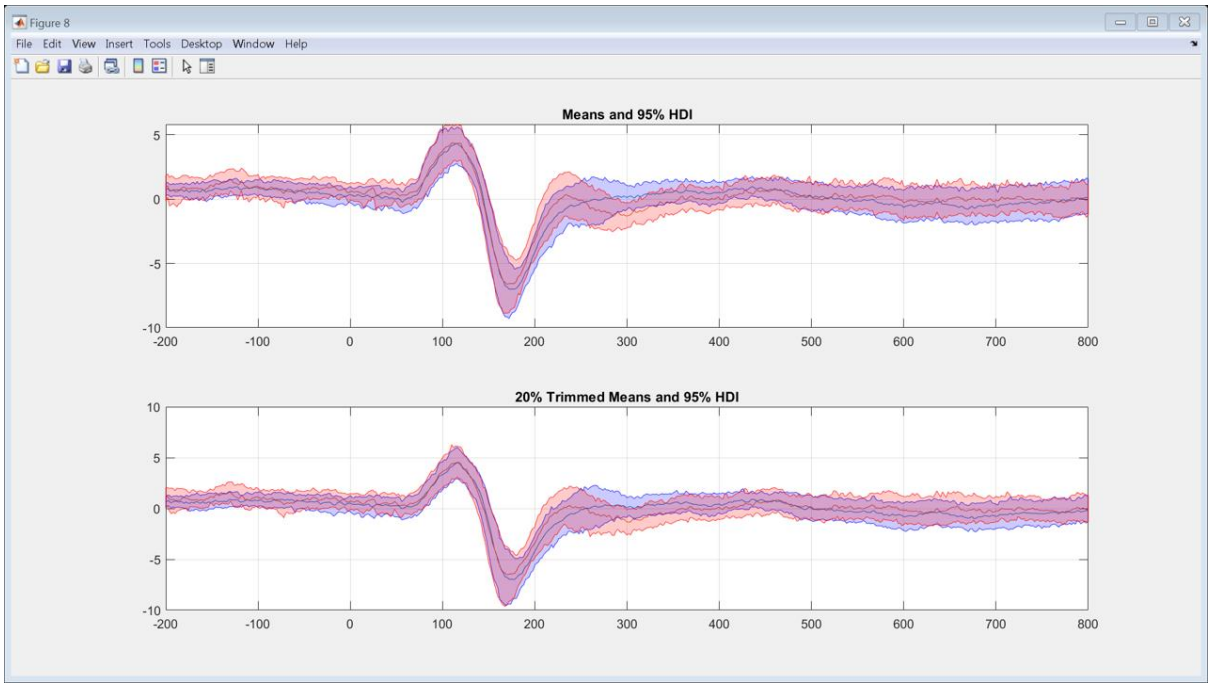


Figure 7: Central Tendency Estimate
File Edit View Insert Tools Desktop Window Help
condition 2 - all subjects
Amplitude
10
5
0
-5
-10
-200
0
200
400
600
800
Time

set the proper time

```
load('cond1-2_single_subjects_Mean');
timevect    = Data.limo.data.start:(1000/Data.limo.data.sampling_rate):Data.limo.data.end;  %
in msec

% compute Mean of two conditions
[est,HDI]   = limo_central_estimator(squeeze(Data.data(electrode,:,1,:)),'Mean',95/100);
[est2,HDI2] = limo_central_estimator(squeeze(Data.data(electrode,:,2,:)),'Mean',95/100);

% plot
figure('units','normalized','outerposition',[0 0 1 1]); subplot(2,1,1)
plot(timevect,est); hold on; plot(timevect,est2); grid on
fillhandle = patch([timevect fliplr(timevect)], [HDI(1,:),fliplr(HDI(2,:))], [0 0 1]);
set(fillhandle,'EdgeColor',[0 0 1],'FaceAlpha',0.2,'EdgeAlpha',0.8);%set edge color
fillhandle = patch([timevect fliplr(timevect)], [HDI2(1,:),fliplr(HDI2(2,:))], [1 0 0]);
set(fillhandle,'EdgeColor',[1 0 0],'FaceAlpha',0.2,'EdgeAlpha',0.8);%set edge color
title('Means and 95% HDI')

% same using trimmed means
[est,HDI]   = limo_central_estimator(squeeze(Data.data(electrode,:,1,:)),'Trimmed
mean',95/100);
[est2,HDI2] = limo_central_estimator(squeeze(Data.data(electrode,:,2,:)),'Trimmed
mean',95/100);

subplot(2,1,2); plot(timevect,est); hold on; plot(timevect,est2); grid on
fillhandle = patch([timevect fliplr(timevect)], [HDI(1,:),fliplr(HDI(2,:))], [0 0 1]);
set(fillhandle,'EdgeColor',[0 0 1],'FaceAlpha',0.2,'EdgeAlpha',0.8);%set edge color
fillhandle = patch([timevect fliplr(timevect)], [HDI2(1,:),fliplr(HDI2(2,:))], [1 0 0]);
set(fillhandle,'EdgeColor',[1 0 0],'FaceAlpha',0.2,'EdgeAlpha',0.8);%set edge color
title('20% Trimmed Means and 95% HDI')
```

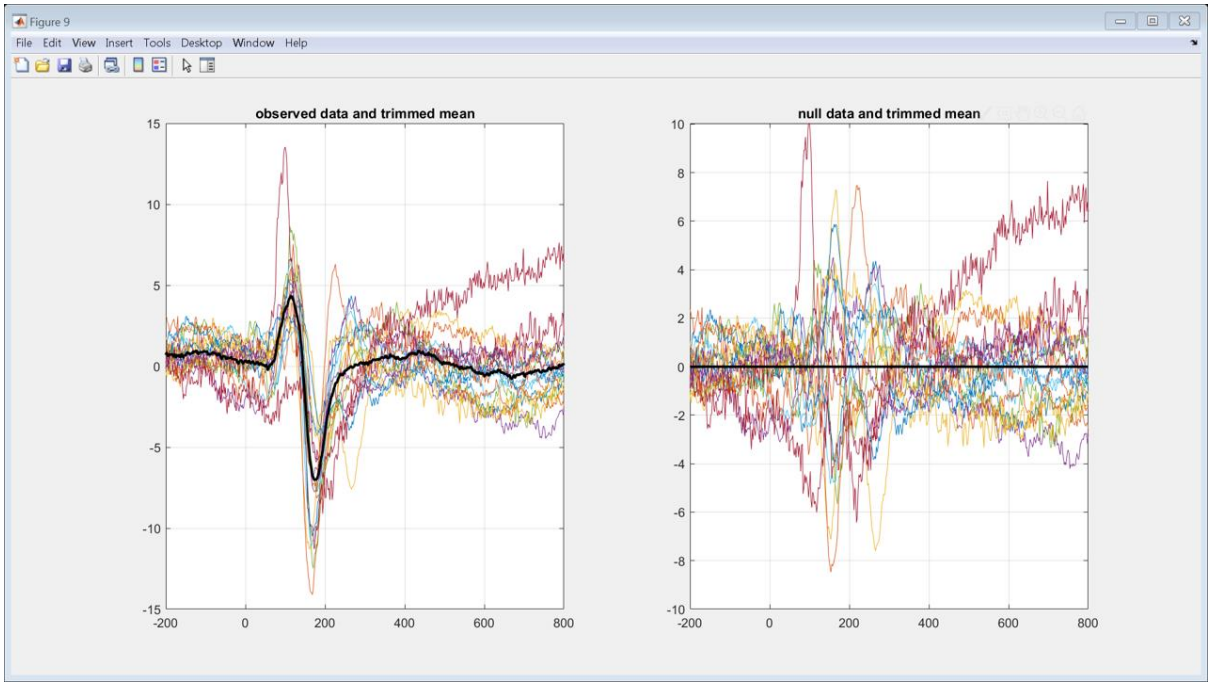


The beauty of using HDI is that 1 - this is the confidence interval of the mean (not the alpha prob. to include the population mean interval) and 2 - since this is Bayesian, you can accept the null (rather than only fail to reject).

estimating H0 using bootstrap

The same way as we can compute the distribution of mean values, we can do it for the null - for instance, a one-sample t-test testing if the trimmed mean(data)=0

```
% let's get the trimmed mean again with 95% CI
data = squeeze(Data.data(:,:,1,:));
[t,tmdata,trimci,se,p,tcrit,df]=limo_trimci(data, 20/100, 5/100, 0);
test = limo_trimmed_mean(data);

% step 1: make null data (the trimmed mean = 0 --> remove TM from each subject)
null = data - repmat(tmdata,[1 1 size(data,3)]);

figure('units','normalized','outerposition',[0 0 1 1]);
subplot(1,2,1); plot(timevect,squeeze(data(electrode,:,:)));
grid on; hold on; plot(timevect,squeeze(tmdata(electrode,:)),'k','LineWidth',2);
title('observed data and trimmed mean');
subplot(1,2,2); plot(timevect,squeeze(null(electrode,:,:)));
grid on; hold on; [~,tmnull]=limo_trimci(null, 20/100, 5/100, 0);
plot(timevect,squeeze(tmnull(electrode,:)),'k','LineWidth',2);
title('null data and trimmed mean');
```



step 2: bootstrap as usual and

```
TM_null = NaN(Nboot,length(timevect)); % record trimmead means under the null (percentille
bootatrap)
T_null  = NaN(Nboot,length(timevect)); % record t values under the null (percentille-t
bootatrap)
boot_table = limo_create_boot_table(data,Nboot);
parfor b=1:Nboot
    [tnull,tmnull] = limo_trimci(null(:,:,boot_table{electrode}(:,b)), 20/100, 5/100, 0);
    TM_null(b,:)   = tmnull(electrode,:);
    T_null(b,:)    = tnull(electrode,:);
end

% step 3: get p-values
% p-values are the prob under the null to observed a value equal or bigger
% than a threshold, we thus compute how many times the observed data are
% bigger than the null
pboot  = mean(squeeze(tmdata(electrode,:,:))>TM_null);
pboot  = min([pboot' 1-pboot'],[],2)';
ptboot = mean(squeeze(t(electrode,:,:))>T_null);
ptboot = min([ptboot' 1-ptboot'],[],2)';

% let make a figure
figure('units','normalized','outerposition',[0 0 1 1]);
plot(timevect,squeeze(tmdata(electrode,:))); hold on; grid on
fillhandle = patch([timevect fliplr(timevect)], ...
    [squeeze(trimci(electrode,:,1)),fliplr(squeeze(trimci(electrode,:,2)))], [0 0 1]);
set(fillhandle,'EdgeColor',[0 0 1],'FaceAlpha',0.2,'EdgeAlpha',0.8);%set edge color
plot(timevect,(squeeze(p(electrode,:))<0.05)-11,'rx'); hold on
plot(timevect,(pboot<0.05)-10.8,'go');
plot(timevect,(ptboot<0.05)-10.6,'k+');
axis([timevect(1) timevect(end) -10 6])
title('Trimmed mean with sig values based on t distrib (red) or percentile boot (green) or
percentile t-boot (black)')
```

multiple comparisons correction

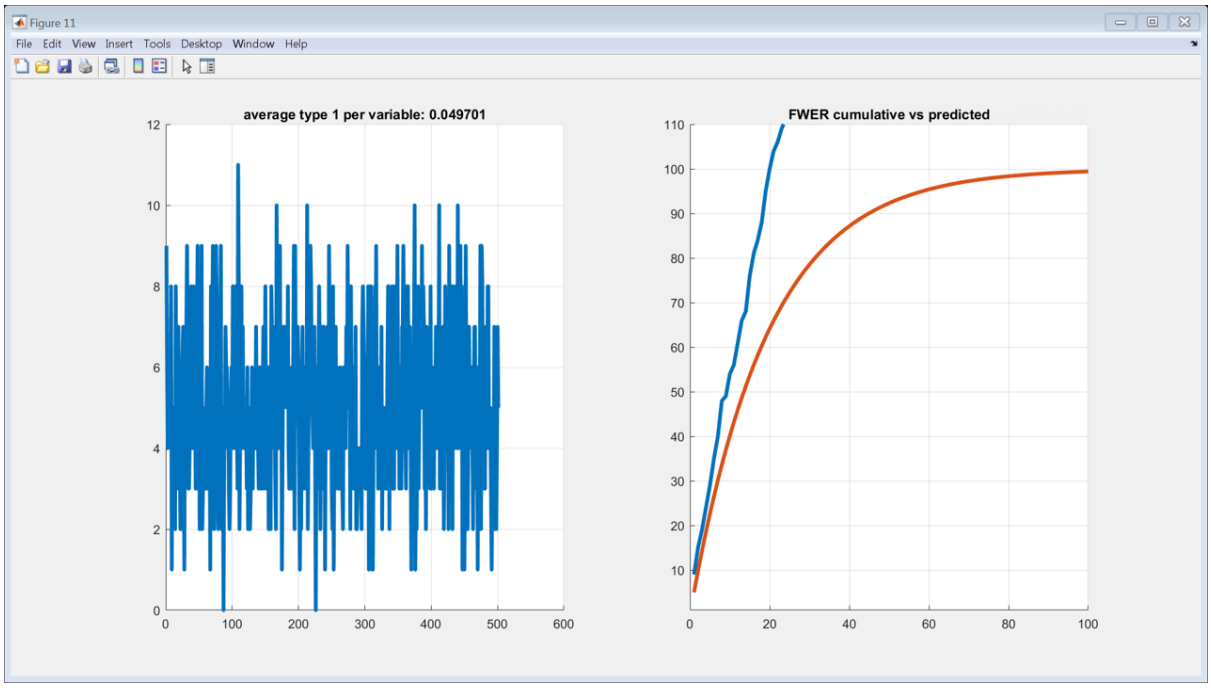
The issue we are facing now is that we have done 501 t-tests so we can expect 25 false positives, we just don't know which ones

```
% let's check if that's true
h = ttest(randn(18,length(timevect)));
fprintf('found %g false positives out of %g expected \n',sum(h),5/100*length(timevect))
```

```
found 22 false positives out of 25.05 expected
```

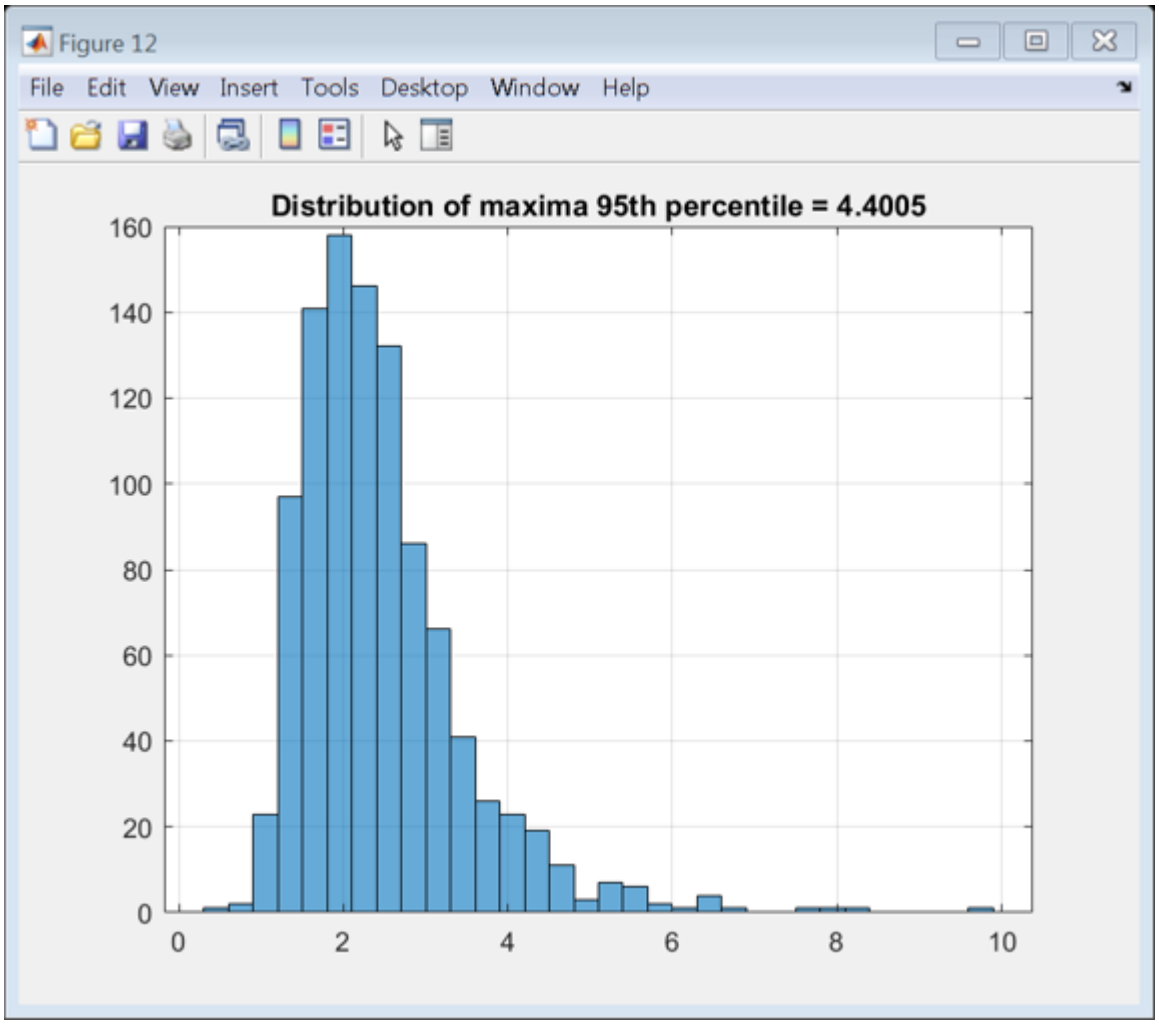
In EEG we talk about family-wise error (FWE) correction because tests are correlated with each other. The FWER = 1 - (1 - alpha)^n, so for 501 tests we reach 100%. Since the FWER is the probability to make at least one error, let's compute the distribution of maximum t value -- if the biggest effect is controlled then smaller effects are controlled too.

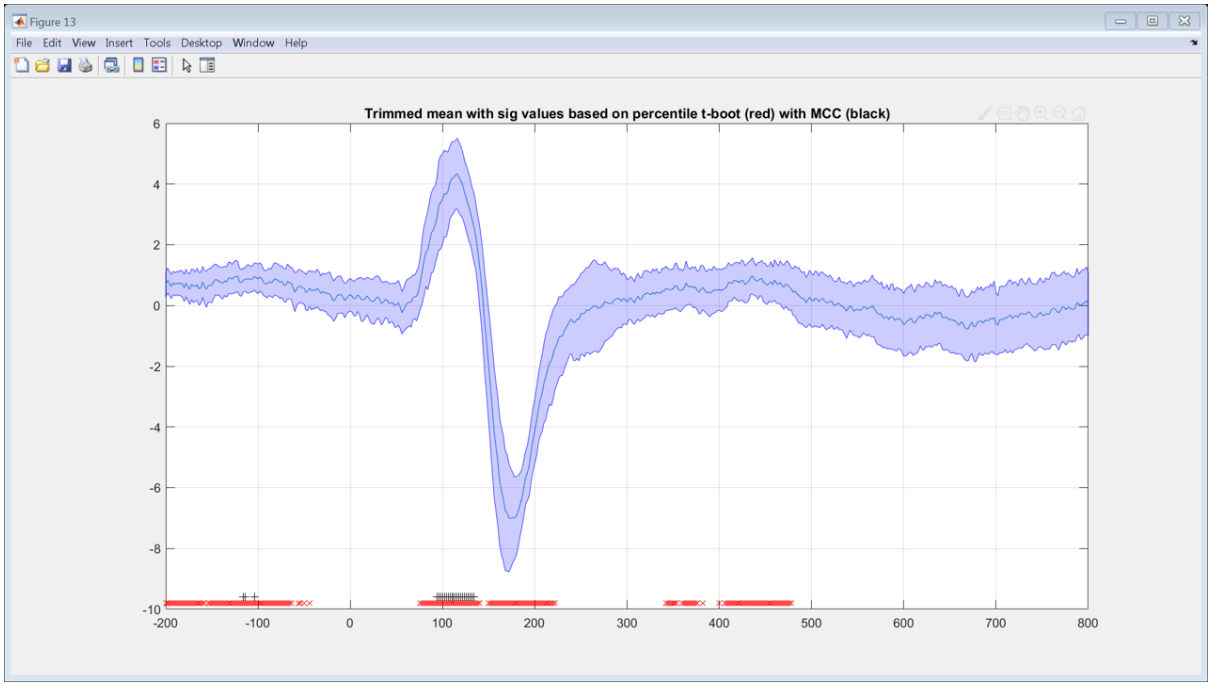
```
for simulation = 1:100
    h(simulation,:) = ttest(randn(18,length(timevect)));
end
E = mean(h); % on average how many errors per variable

figure('units','normalized','outerposition',[0 0 1 1]);
subplot(1,2,1); plot(E.*100,'LineWidth',3); grid on; box;
title(['average type 1 per variable: ' num2str(mean(E))])
subplot(1,2,2); plot(cumsum(E).*100,'LineWidth',3); grid on; box;
for n=1:length(timevect); FWER(n) = 1 - (1 - 0.05)^n; end
hold on;  plot(FWER.*100,'LineWidth',3); axis([0 100 1 110])
title('FWER cumulative vs predicted');
```



let's see what happens to our ERP using maximum statistics

```
Tmax_null = sort(max(T_null,[],2)); % this time we record t-values
high      = round((1-alpha_value).*Nboot);
threshold = Tmax_null(high); % that's the value we need to go other for
                             % the largest effect anywhere to be significant
figure; histogram(Tmax_null); hold on; grid on
title(['Distribution of maxima 95th percentile = ' num2str(threshold)])

figure('units','normalized','outerposition',[0 0 1 1]);
plot(timevect,squeeze(tmdata(electrode,:))); hold on; grid on
fillhandle = patch([timevect fliplr(timevect)], ...
    [squeeze(trimci(electrode,:,1)),fliplr(squeeze(trimci(electrode,:,2)))], [0 0 1]);
set(fillhandle,'EdgeColor',[0 0 1],'FaceAlpha',0.2,'EdgeAlpha',0.8);%set edge color
plot(timevect,(ptboot<0.05)-10.8,'rx');
plot(timevect,(squeeze(t(electrode,:))>threshold)-10.6,'k+');
axis([timevect(1) timevect(end) -10 6])
title('Trimmed mean with sig values based on percentile t-boot (red) with MCC (black)')
```





So far, we have considered each time point as independent and even didn't consider space (over all electrodes) - Dedicated methods have been developped like cluster-mass and tfce do do just that, see Pernet et al 2015 for an overview of these, more powerful, approaches. In short, instead of looking at the maximum of all data point, electrodes and time points are clustered and we look at the maximum of clusters under null.

Published with MATLAB® R2018b