Data overview and Donsker Test Explained

Contents

Inter group comparison

Data loading

We are using R and we will need, in addition to the standard `R` distribution the library `hdf5r` since the data are distributed using the HDF5 format. This means that a typical reader will most likely have to install the library and this is done by typing in `R` command line:`install.packages("hdf5r")`. Once this is done, and we need to do it once only, we start by loading the library:

```
library(hdf5r)
```

We can now load the data assuming that `R` was started in the directory where the data `beta_values.h5` were downloaded:

```
fin = H5File$new("beta_values.h5","r")
```

A very "rough" way to get information on this file is:

```
fin
```

```
## Class: H5File
## Filename: /home/xtof/Data/Vera/Permut/beta_values.h5
## Access type: H5F_ACC_RDONLY
## Attributes: README
## Listing:
##      name  obj_type dataset.dims dataset.type_class
##       CTL H5I_GROUP         <NA>               <NA>
##  Patients H5I_GROUP         <NA>               <NA>
```

We see that the file contain 2 "groups" and has a "README" attribute. Let's check the content of the latter:

```
cat(h5attr(fin,"README"))
```

```
## Data from two groups: "CTL" made of 12 subjects and "Patients" made of 13 subjects.
## Each of the two groups is subdivided into subjects: subject01, subject02,...,subject11,subject12 and
## Each of these sub-groups contains three datasets: low, medium and high corresponding to the three exp
## Each data set is a matrix of doubles between -1 and +1 by construction (these are Spearman's correla
## The rows of each data set correspond to the trial index (28 trials in low and medium load and 25 tria
```

We will need to access to both the `CTL` and the `Patients` groups so let's make some "shortcuts" to them:

```
fin_CTL = fin[["CTL"]]
fin_Patients = fin[["Patients"]]
```

We can see the names of the groups in the CTL set with:

```
names(fin_CTL)
```

```
##  [1] "subject01" "subject02" "subject03" "subject04" "subject05"
##  [6] "subject06" "subject07" "subject08" "subject09" "subject10"
## [11] "subject11" "subject12"
```

In the same way we get the names of the Patients groups with:

```
names(fin_Patients)
```

```
##  [1] "subject01" "subject02" "subject03" "subject04" "subject05"
##  [6] "subject06" "subject07" "subject08" "subject09" "subject10"
## [11] "subject11" "subject12" "subject13"
```

So we have indeed two "groups" in HDF5 parlance: "CTL" with 12 cases and "Patients" with 13 cases.

In order to have a smoother R handling of these data, we create three lists, one for each experimental "load": low, medium and high. Each of these lists is a list of matrices starting with the 12 controls and ending with the 13 patients.

low load list

In R, the name of our list will be responseL_list:

```
responseL_list = list()
for (i in 1:12) {
    if (i < 10) {
        name_in = paste0("subject0",i,"/low")
        name_out = paste0("CTL0",i)
    } else {
        name_in = paste0("subject",i,"/low")
        name_out = paste0("CTL",i)
    }
    responseL_list[[name_out]] = fin_CTL[[name_in]][,]
}
for (i in 1:13) {
    if (i < 10) {
        name_in = paste0("subject0",i,"/low")
        name_out = paste0("Patient0",i)
    } else {
        name_in = paste0("subject",i,"/low")
        name_out = paste0("Patient",i)
    }
    responseL_list[[name_out]] = fin_Patients[[name_in]][,]
}
```

We must be careful in subsequent use of these data since R is "column major" the matrix got transposed upon loading. The trials are now along the columns.

medium load list

In R, the name of our list will be responseM_list:

```
responseM_list = list()
for (i in 1:12) {
    if (i < 10) {
        name_in = paste0("subject0",i,"/medium")
        name_out = paste0("CTL0",i)
```

```
    } else {
        name_in = paste0("subject",i,"/medium")
        name_out = paste0("CTL",i)
    }
    responseM_list[[name_out]] = fin_CTL[[name_in]][,]
}
for (i in 1:13) {
    if (i < 10) {
        name_in = paste0("subject0",i,"/medium")
        name_out = paste0("Patient0",i)
    } else {
        name_in = paste0("subject",i,"/medium")
        name_out = paste0("Patient",i)
    }
    responseM_list[[name_out]] = fin_Patients[[name_in]][,]
}
```

`high` load list

In R, the name of our list will be `responseH_list`:

```
responseH_list = list()
for (i in 1:12) {
    if (i < 10) {
        name_in = paste0("subject0",i,"/high")
        name_out = paste0("CTL0",i)
    } else {
        name_in = paste0("subject",i,"/high")
        name_out = paste0("CTL",i)
    }
    responseH_list[[name_out]] = fin_CTL[[name_in]][,]
}
for (i in 1:13) {
    if (i < 10) {
        name_in = paste0("subject0",i,"/high")
        name_out = paste0("Patient0",i)
    } else {
        name_in = paste0("subject",i,"/high")
        name_out = paste0("Patient",i)
    }
    responseH_list[[name_out]] = fin_Patients[[name_in]][,]
}
```

Closing the file

Once we are done with this "data loading" task, we can close the HDF5 file:

```
fin$close_all()
```

Low load case

Getting a readable overview of the data

We look next at the individual trials but, in order to get "readable" graphs we first reorder the voxels (rows of the matrices) such that the mean value (across trials) of the voxels increases–we want to be able to see the difference

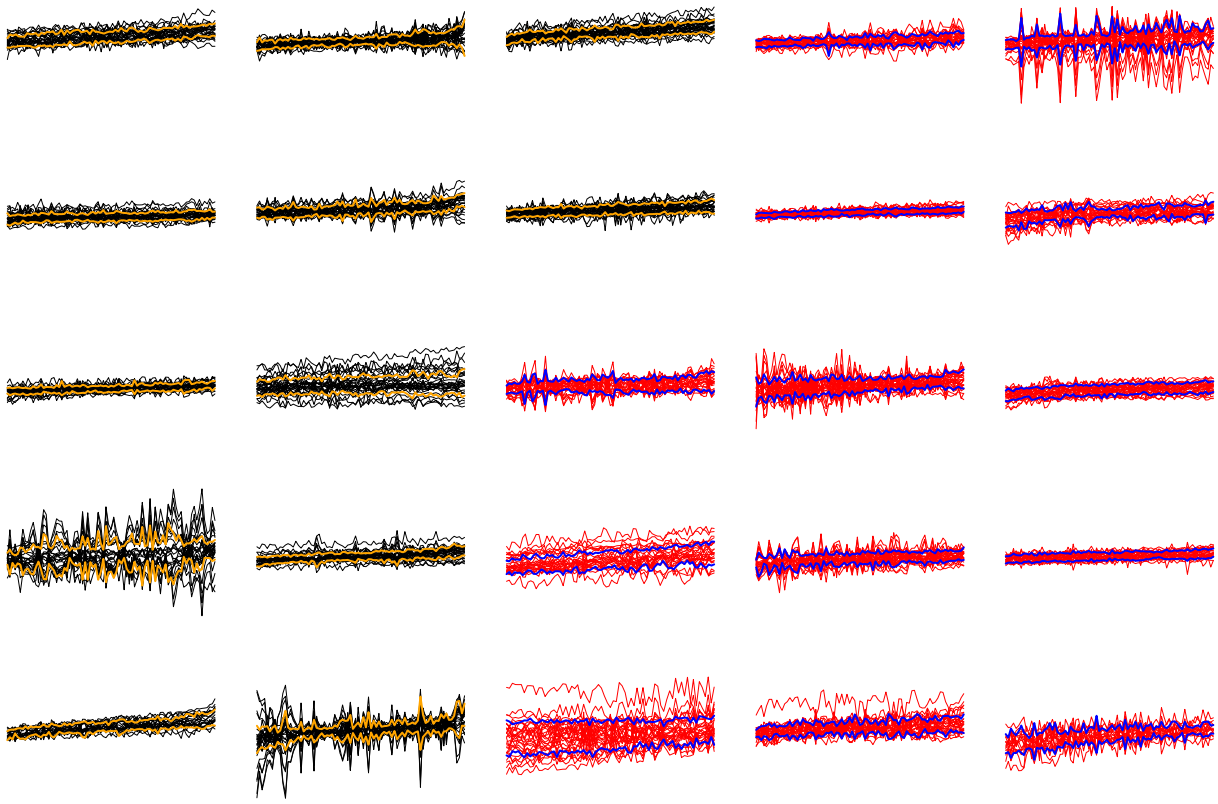

Figure 1: Raw data for control subjects (black) and patients (red). On each sub graph, each line corresponds to one trial. The abscissa gives the voxel number (after sorting of the latter according to the mean value; this explains the positive trend from left to right). For each subject the median +/- the MAD has been added (in orange for controls and in blue for patients).

between an across trial variability shared by all the voxels and voxels exhibiting a high variability–.

```
layout(matrix(1:25,nr=5))
domain = range(unlist(lapply(responseL_list,as.vector))[!is.na(unlist(lapply(responseL_list,as.vector))]
par(mar=c(1,1,0,0))
lapply(1:length(responseL_list),function(idx) {
    m = responseL_list[[idx]]
    mean_vox = apply(m,1,mean,na.rm=TRUE)
    new_vox_idx = sort.int(mean_vox,index.return = TRUE)$ix
    m = m[new_vox_idx,]
    m_med = apply(m,1,median,na.rm=TRUE)
    m_mad = apply(m,1,mad,na.rm=TRUE)
    matplot(m,type="l",col=ifelse(idx<=12,1,2),lwd=0.4,lty=1,ylim=domain,axes=FALSE)
    m = cbind(m_med-m_mad,m_med+m_mad)
    matlines(m,type="l",col=ifelse(idx<=12,"orange","blue"),lwd=1,lty=1,ylim=domain)
})
```

Here, the control subjects are in black, the patients in red. Numbers are increasing along rows, then along column. That is, the last control subject (subject 12) is on the third column, second row. For each subject the voxels have been sorted along increasing mean values. *All ordinates are on the same scale.* The median +/- the mad (`median absolute deviation`) has been added to each graph.

We see clearly qualitatively different behaviors:

- Some controls (row 3 column 1) and patients (row 2 column 4) exhibit very regular responses.
- Some exhibit a rather strong across trial variability evenly shared among all the voxels (row 3 column 2 for a control and row 5 column 3 for a patient).
- Some exhibit bith very variable voxels and very regular ones (row 5 column 2 for a control and row 1 column 5 for a patient).

The data set is not huge but there do not seem to be a clear cut separation between controls and patients based on the above three criteria.

Getting the correlation matrices

We now computed for each subject the inter trial correlation using the Spearman correlation coefficient. This is done immediately with:

```
cor_list = lapply(responseL_list,function(m) cor(m,method="spearman"))
```

Now each element of `cor_list` is a 28x28 matrix (since we have 28 trials). Some of the elements of these matrices are `NA` (Not Available since some of the trials were themselves `NA`).

We now have two collections of (symmetric) matrices one for the 12 controls and one for the 13 patients. **The question we somehow want to address is are these two groups identical insofar as these correlation matrices are concerned?**

We will start by creating a mean correlation matrix and a variance correlation matrix for each of the two groups. Here "mean correlation matrix" means a matrix whose *(i,j)* element is the mean of the *(i,j)* element for each member of a given group, the same goes for the "variance correlation matrix" (not to be confused with a "variance-covariance matrix"). To get these two matrices for each of the two groups we start by creating a "matrix stack" for each group. A matrix stack is a 3D object (a 3D `array` for `R`) where the third dimension is the subject index–you can visualize that as literally making a stack of matrices–this intermediate step facilitates the subsequent calculation:

```
ctl_stack = array(0,dim=c(28,28,12))
patient_stack = array(0,dim=c(28,28,13))
for (i in 1:12) ctl_stack[,,i] = cor_list[[i]]
for (i in 1:13) patient_stack[,,i] = cor_list[[i+12]]
ctl_mean = apply(ctl_stack,c(1,2),mean)
ctl_var = apply(ctl_stack,c(1,2),var)
patient_mean = apply(patient_stack,c(1,2),mean)
patient_var = apply(patient_stack,c(1,2),var)
```

As a quick way to visualize the results we can build a matrix with a lower triangular part given by the patient mean and an upper triangular part given by the control mean (we include in this matrix only the minimum between the two groups of the maximum number of "available" trials):

```
ctl_patient_mean = ctl_mean[1:16,1:16]
for (i in 2:16) for (j in 1:(i-1)) ctl_patient_mean[i,j] = patient_mean[i,j]
image(1:16,1:16,ctl_patient_mean,col=grey.colors(n=256,start=0,end=1),
      xlab="Trial index",ylab="Trial index",asp=1)
```

To go further than this (hard to interpret) display since our null hypothesis is: *the correlation matrices (for the successful trials) of the two groups are identical*, we can look at the properly normalized difference matrix. For each matrix element *(i,j)* we have a mean, $\widehat{\beta}_{i,j}^{CTL}$ and variance, $\widehat{\sigma}_{i,j}^{2\,CTL}$, value (computed on n^{CTL} subjects) for the control group and a mean, $\widehat{\beta}_{i,j}^{PAT}$ and variance, $\widehat{\sigma}_{i,j}^{2\,PAT}$, value (computed on n^{PAT} subjects) for the patient group. If our null hypothesis is correct:

$$\frac{\widehat{\beta}_{i,j}^{CTL} - \widehat{\beta}_{i,j}^{PAT}}{\sqrt{\widehat{\sigma}_{i,j}^{2\,CTL}/n^{CTL} + \widehat{\sigma}_{i,j}^{2\,PAT}/n^{PAT}}}$$

should (approximately) follow a Gaussian distribution–in fact, more precisely a t distribution–with a null mean

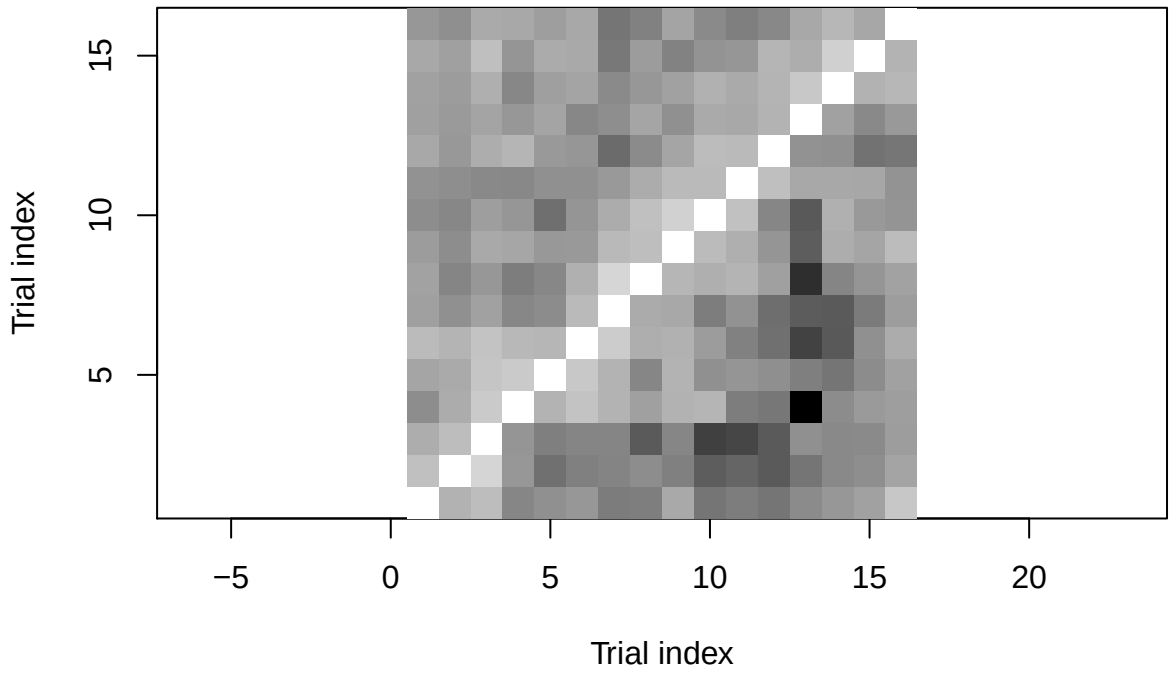

Figure 2: Above first bisector, the mean correlation matrix for patients; below first bisector, the mean correlation matrix for controls.

value and a variance equal to 1. If we decide to go for the "proper" t distribution then number of degrees of freedom is given by:

$$\nu \approx \frac{\left(\widehat{\sigma}^{2\,CTL}_{i,j}/n^{CTL} + \widehat{\sigma}^{2\,PAT}_{i,j}/n^{PAT}\right)^2}{\widehat{\sigma}^{4\,CTL}_{i,j}/\left(n^{2CTL}(n^{CTL})-1\right) + \widehat{\sigma}^{4\,PAT}_{i,j}/\left(n^{2PAT}(n^{PAT})-1\right)}\,.$$

We see that the number of effective degrees of freedom can be led to change to variations in empirical variance estimators.

We can compute this normalized difference matrix and we get:

```
D = (ctl_mean[1:16,1:16]-patient_mean[1:16,1:16])/sqrt(ctl_var[1:16,1:16]/12+patient_var[1:16,1:16]/13)
image(1:16,1:16,D,col=grey.colors(n=256,start=0,end=1),
      xlab="Trial index",ylab="Trial index",asp=1)
```

The question becomes: **Can we assume that one of the triangular parts of this matrix was generated by drawing independently for each element a random number distribution with mean value 0 and variance 1?**

An identity test that does not assume more than zero mean and identical variances

What we need is Donsker's theorem also known as the *invariance principle* (Pinsky and Karlin, *An Introduction to Stochastic Modeling*, Academic Press, 2011, pp 396-397) that says: Let $X_1, X_2, \ldots$ be a sequence of IID random variables with mean 0 and variance 1. Let $S_n \equiv \sum_{i=1}^{n} X_i$. The stochastic process $S = (S_n)_{n\in\mathbb{N}}$ is known as a random walk. Define the rescaled random walk by:

$$W^{(n)}(t) \equiv \frac{S_{\lfloor nt \rfloor}}{\sqrt{n}}, \quad t \in [0,1]\,.$$

Then $W^{(n)}(t)$ converges in distribution towards a *standard Brownian motion process* (here $\lfloor nt \rfloor$ is the largest integer smaller or equal to nt).

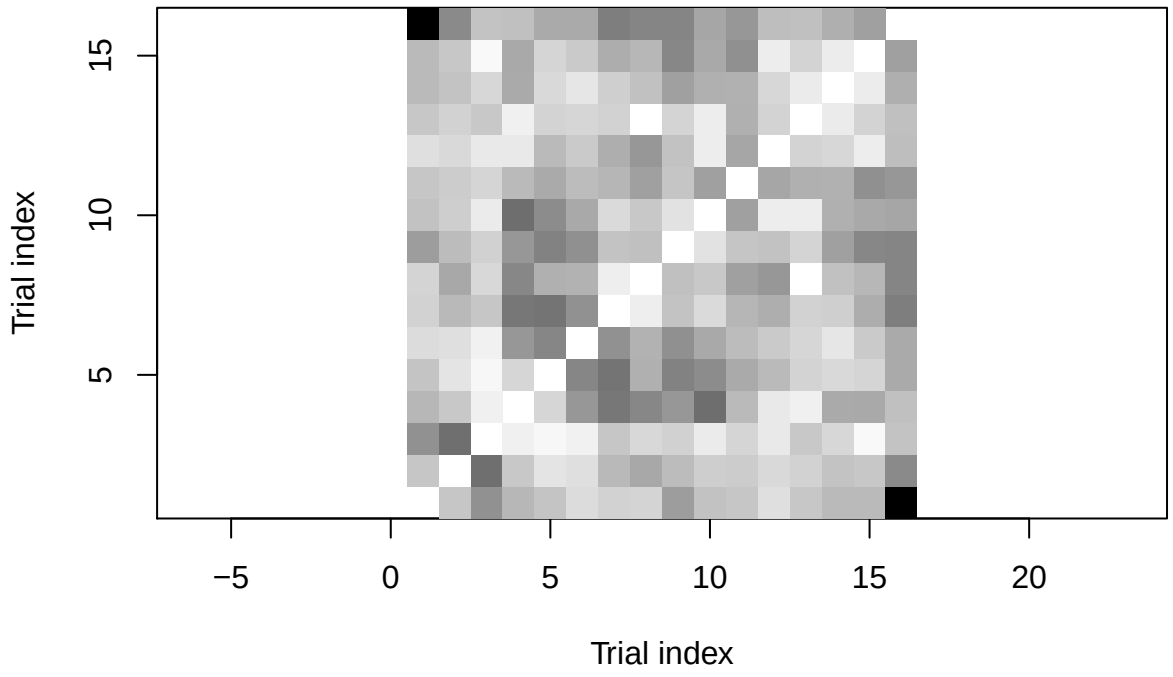

Figure 3: Normalized difference of the control and patient mean correlation matrices.

Here the role of the X_i is played by the elements of our difference matrix `D`. So we have to transform the upper triangular part into a vector (for instance by pasting the relevant portion of rows one after the other), compute the cumulative sum of the vector elements and plot the rescaled version of this cumulative sum. If what is obtained does not look like the realization a *standard Brownian motion process* we can reject the null hypothesis.

This means that we must be able to recognize the realization of a *standard Brownian motion process* when we see one. This sounds not obvious but luckily in a (not exactly "easy to read") paper, Kendall, Marin and Robert (Brownian Confidence Bands on Monte Carlo Output, *Statistics and Computing* **17**:1-10, 2007) show how to construct a minimal surface domain that will contain *in their totality* a given fraction of the realizations of a standard Brownian motion process. Since the expression of boundary of this minimal surface is slightly cumbersome to work with, they recommend to work with boundaries of the form $a + b\sqrt{t}$ (giving domains with an almost minimal surface), adjusting a and b to get the desired coverage probability. Precise values of a and b can be found in Pouzat, Chaffiol and Bar-Hen (Homogeneity and identity tests for unidimensional Poisson processes with an application to neurophysiological peri-stimulus time histograms, 2015, hal-01113126v2); for a coverage probability of 0.95 one should use $a = 0.3$ and $b = 2.348$, for a coverage probability of 0.99, one should use $a = 0.312$ and $b = 2.891$.

On our data that gives:

```
Z = numeric(16*15/2)
i = 1
for (j in 2:16) {
    for (k in 1:(j-1)) {
        Z[i] = D[j,k]
        i = i+1
    }
}
plot((1:120)/120,cumsum(Z)/sqrt(120),type="l",
     xlab="Normalized (pseudo) time",
     ylab="Rescaled cumsum",ylim=c(-5,5),
     lwd=2)
xx = seq(0:500)/500
```

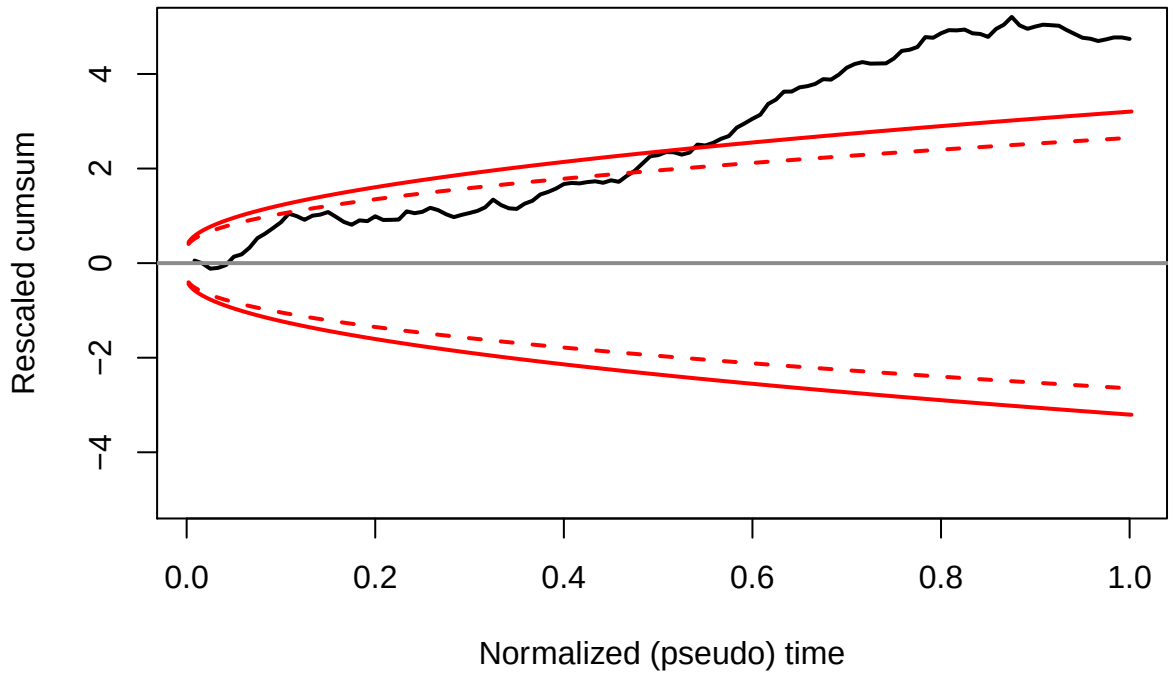

Figure 4: Rescaled cumulative sum of the row portions of the upper triangular part of matrix `D`. Red continuous: boundary of the 0.99 domain; red dashed: boundary of the 0.95 domain.

```
lines(xx,0.312+2.891*sqrt(xx),lwd=2,col=2)
lines(xx,-0.312-2.891*sqrt(xx),lwd=2,col=2)
lines(xx,0.3+2.348*sqrt(xx),lwd=2,col=2,lty=2)
lines(xx,-0.3-2.348*sqrt(xx),lwd=2,col=2,lty=2)
abline(h=0,col="grey55",lwd=2)
```

Here the observed path leaves the minimal surface region(s) and we can **reject the null hypothesis at the 0.99 level**.

High load case

We repeat the analysis in the "high load" case.

Inspect individual trials

```
layout(matrix(1:25,nr=5))
domain = range(unlist(lapply(responseH_list,as.vector))[!is.na(unlist(lapply(responseH_list,as.vector))]
par(mar=c(1,1,0,0))
lapply(1:length(responseH_list),function(idx) {
    m = responseH_list[[idx]]
    mean_vox = apply(m,1,mean,na.rm=TRUE)
    new_vox_idx = sort.int(mean_vox,index.return = TRUE)$ix
    m = m[new_vox_idx,]
    m_med = apply(m,1,median,na.rm=TRUE)
    m_mad = apply(m,1,mad,na.rm=TRUE)
    matplot(m,type="l",col=ifelse(idx<=12,1,2),lwd=0.4,lty=1,ylim=domain,axes=FALSE)
    m = cbind(m_med-m_mad,m_med+m_mad)
    matlines(m,type="l",col=ifelse(idx<=12,"orange","blue"),lwd=1,lty=1,ylim=domain)
})
```

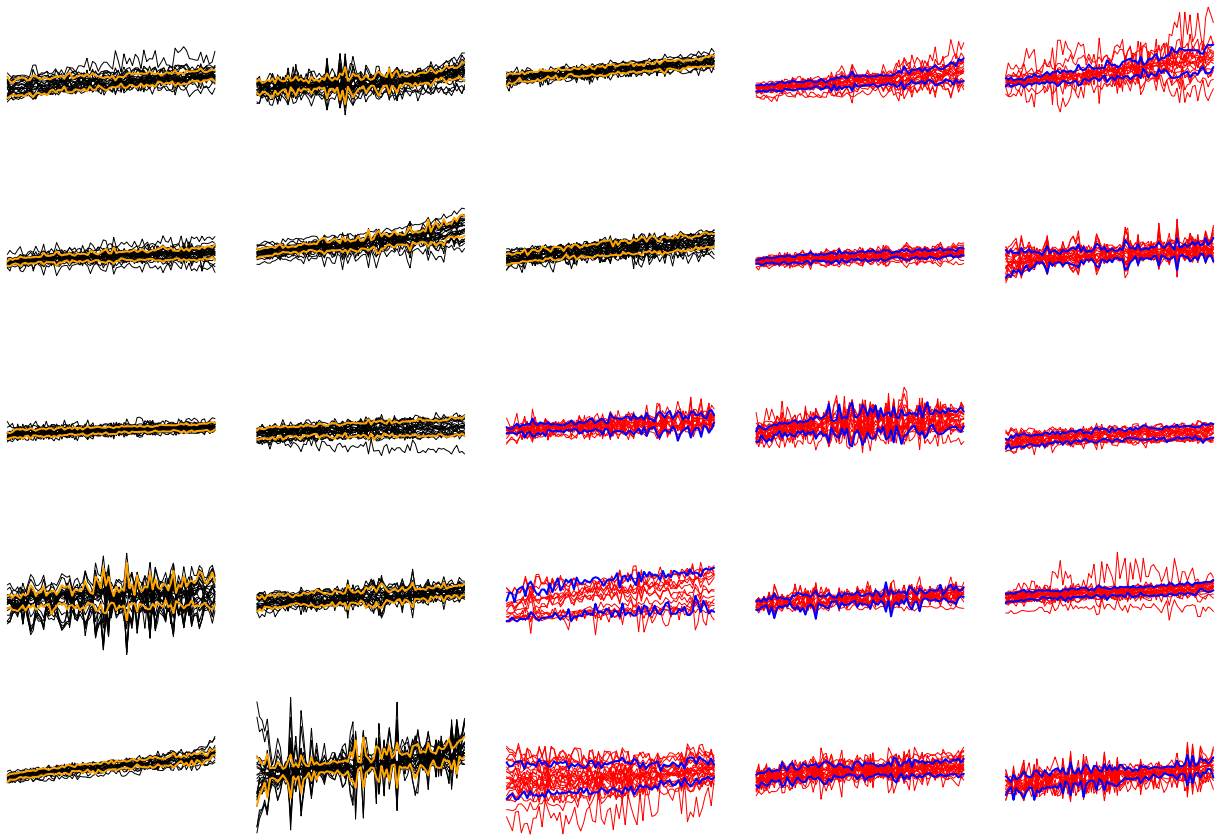

Figure 5: Raw data for control subjects (black) and patients (red) in the high load case. On each sub graph, each line corresponds to one trial. The abscissa gives the voxel number (after sorting of the latter according to the mean value; this explains the positive trend from left to right). For each subject the median +/- the MAD has been added (in orange for controls and in blue for patients).

Get the correlation matrices, their mean and variance matrices

```
corH_list = lapply(responseH_list,function(m) cor(m,method="spearman"))
```

```
nt = 25
ctlH_stack = array(0,dim=c(nt,nt,12))
patientH_stack = array(0,dim=c(nt,nt,13))
for (i in 1:12) ctlH_stack[,,i] = corH_list[[i]]
for (i in 1:13) patientH_stack[,,i] = corH_list[[i+12]]
ctlH_mean = apply(ctlH_stack,c(1,2),mean)
ctlH_var = apply(ctlH_stack,c(1,2),var)
patientH_mean = apply(patientH_stack,c(1,2),mean)
patientH_var = apply(patientH_stack,c(1,2),var)
```

Do Donsker's test

```
DH = (ctlH_mean[1:15,1:15]-patientH_mean[1:15,1:15])/sqrt(ctlH_var[1:15,1:15]/12+patientH_var[1:15,1:15]
ZH = numeric(15*14/2)
i = 1
for (j in 2:15) {
    for (k in 1:(j-1)) {
        ZH[i] = DH[j,k]
        i = i+1
    }
}
plot((1:105)/105,cumsum(ZH)/sqrt(105),type="l",
     xlab="Normalized (pseudo) time",
     ylab="Rescaled cumsum",ylim=c(-5,10),
     lwd=2)
xx = seq(0:500)/500
lines(xx,0.312+2.891*sqrt(xx),lwd=2,col=2)
lines(xx,-0.312-2.891*sqrt(xx),lwd=2,col=2)
lines(xx,0.3+2.348*sqrt(xx),lwd=2,col=2,lty=2)
lines(xx,-0.3-2.348*sqrt(xx),lwd=2,col=2,lty=2)
abline(h=0,col="grey55",lwd=2)
```

We can again **reject the null hypothesis at the 0.99 level**.

Medium load case

We repeat the analysis in the "medium load" case.

Inspect individual trials

```
layout(matrix(1:25,nr=5))
domain = range(unlist(lapply(responseM_list,as.vector))[!is.na(unlist(lapply(responseM_list,as.vector))]
par(mar=c(1,1,0,0))
lapply(1:length(responseM_list),function(idx) {
    m = responseM_list[[idx]]
    mean_vox = apply(m,1,mean,na.rm=TRUE)
    new_vox_idx = sort.int(mean_vox,index.return = TRUE)$ix
    m = m[new_vox_idx,]
    m_med = apply(m,1,median,na.rm=TRUE)
    m_mad = apply(m,1,mad,na.rm=TRUE)
```

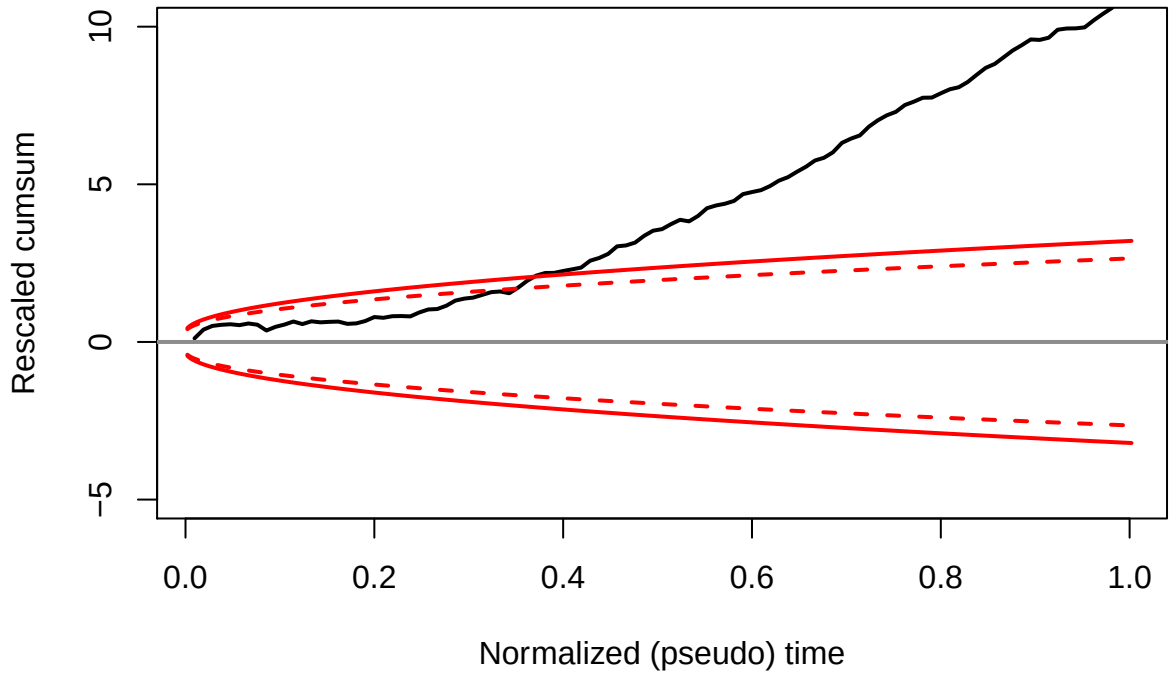

Figure 6: Rescaled cumulative sum of the row portions of the upper triangular part of matrix DH. Red continuous: boundary of the 0.99 domain; red dashed: boundary of the 0.95 domain.

```
    matplot(m,type="l",col=ifelse(idx<=12,1,2),lwd=0.4,lty=1,ylim=domain,axes=FALSE)
    m = cbind(m_med-m_mad,m_med+m_mad)
    matlines(m,type="l",col=ifelse(idx<=12,"orange","blue"),lwd=1,lty=1,ylim=domain)
})
```

Get the correlation matrices, their mean and variance matrices

```
corM_list = lapply(responseM_list,function(m) cor(m,method="spearman"))
```

```
nt = 28
ctlM_stack = array(0,dim=c(nt,nt,12))
patientM_stack = array(0,dim=c(nt,nt,13))
for (i in 1:12) ctlM_stack[,,i] = corM_list[[i]]
for (i in 1:13) patientM_stack[,,i] = corM_list[[i+12]]
ctlM_mean = apply(ctlM_stack,c(1,2),mean)
ctlM_var = apply(ctlM_stack,c(1,2),var)
patientM_mean = apply(patientM_stack,c(1,2),mean)
patientM_var = apply(patientM_stack,c(1,2),var)
```

Do Donsker's test

```
DM = (ctlM_mean[1:15,1:15]-patientM_mean[1:15,1:15])/sqrt(ctlM_var[1:15,1:15]/12+patientM_var[1:15,1:15]
ZM = numeric(15*14/2)
i = 1
for (j in 2:15) {
    for (k in 1:(j-1)) {
        ZM[i] = DM[j,k]
```

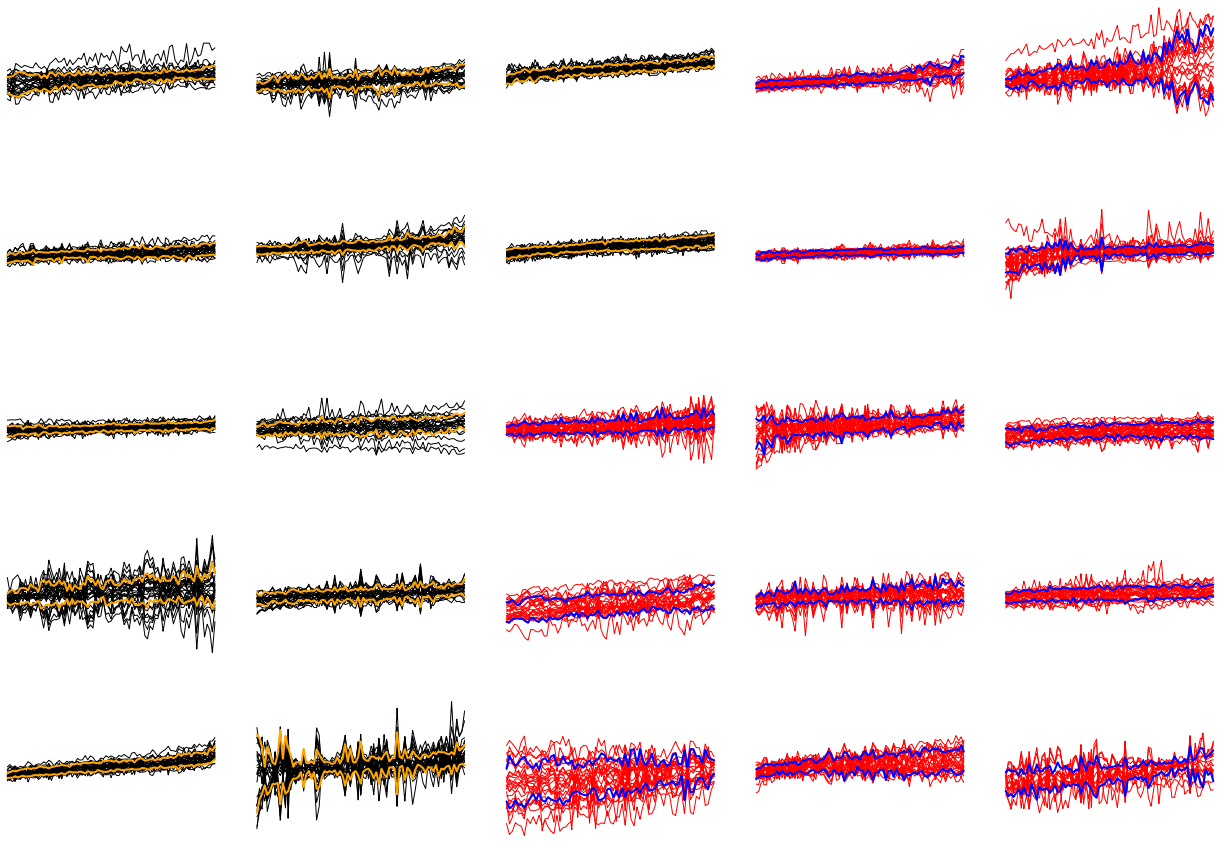

Figure 7: Raw data for control subjects (black) and patients (red) in the high load case. On each sub graph, each line corresponds to one trial. The abscissa gives the voxel number (after sorting of the latter according to the mean value; this explains the positive trend from left to right). For each subject the median +/- the MAD has been added (in orange for controls and in blue for patients).

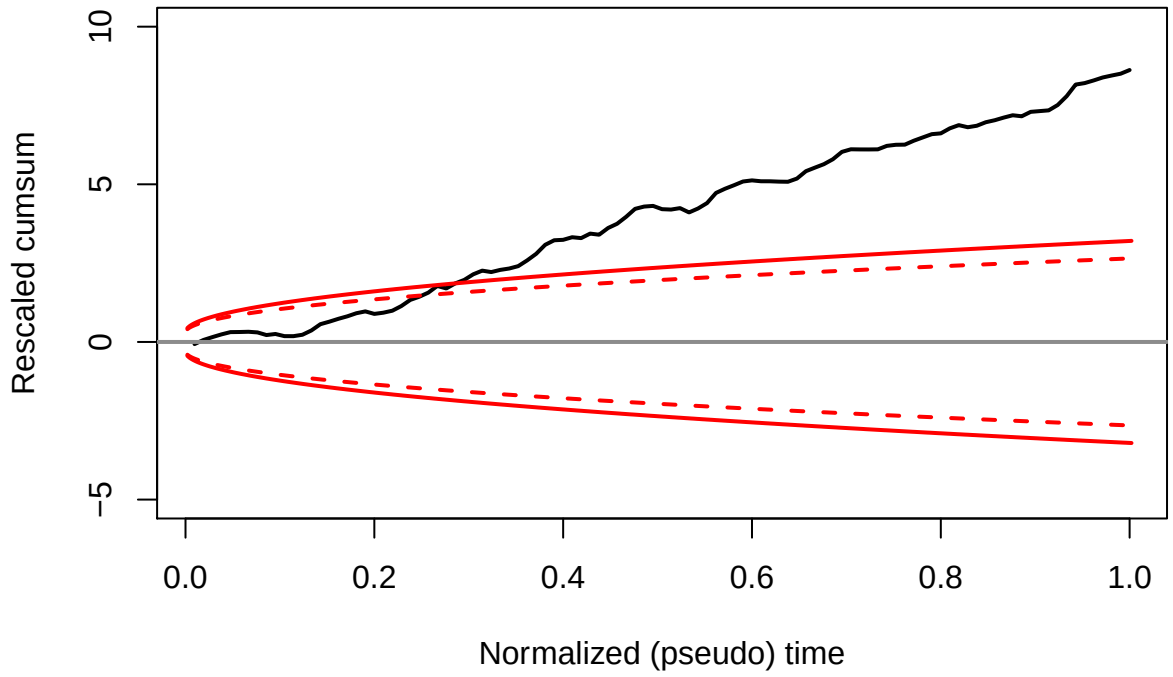

Figure 8: Rescaled cumulative sum of the row portions of the upper triangular part of matrix `DM`. Red continuous: boundary of the 0.99 domain; red dashed: boundary of the 0.95 domain.

```
        i = i+1
    }
}
plot((1:105)/105,cumsum(ZM)/sqrt(105),type="l",
     xlab="Normalized (pseudo) time",
     ylab="Rescaled cumsum",ylim=c(-5,10),
     lwd=2)
xx = seq(0:500)/500
lines(xx,0.312+2.891*sqrt(xx),lwd=2,col=2)
lines(xx,-0.312-2.891*sqrt(xx),lwd=2,col=2)
lines(xx,0.3+2.348*sqrt(xx),lwd=2,col=2,lty=2)
lines(xx,-0.3-2.348*sqrt(xx),lwd=2,col=2,lty=2)
abline(h=0,col="grey55",lwd=2)
```

We can again **reject the null hypothesis at the 0.99 level**.

Low vs High comparison for each subject

Function definitions

The difficulty here is to obtain the variance needed to normalize our differences of correlation coefficients (we got previously from the empirical variance computed across the population members). If we trust wikipedia the standard error on the correlation coefficient is $0.6326/\sqrt{n-1}$, where n is here the number of voxels in the ROI of a given patient (I have to check that). We can then define a function taking a CTL or patient index and doing the job of returning the `Z` vector as follows:

```
get_Z = function(index,CTL=TRUE) {
  ## Get the number of voxels in the ROI
```

```
  n_vox = dim(responseL_list[[index+ifelse(CTL,0,12)]])[1]
  if (CTL) {
    ## Find out the number of good trials in Low condition
    n_good_L = dim(ctl_stack)[1] - sum(is.na(ctl_stack[1,,index]))
    ## Find out the number of good trials in High condition
    n_good_H = dim(ctlH_stack)[1] - sum(is.na(ctlH_stack[1,,index]))
    ## Get the minimum
    n_good = min(n_good_L,n_good_H)
    D = (ctlH_stack[1:n_good,1:n_good,index]-ctl_stack[1:n_good,1:n_good,index])/(sqrt(2)*0.6325/sqrt(n_
  } else {
    ## Find out the number of good trials in Low condition
    n_good_L = dim(patient_stack)[1] - sum(is.na(patient_stack[1,,index]))
    ## Find out the number of good trials in High condition
    n_good_H = dim(patientH_stack)[1] - sum(is.na(patientH_stack[1,,index]))
    ## Get the minimum
    n_good = min(n_good_L,n_good_H)
    D = (patientH_stack[1:n_good,1:n_good,index]-patient_stack[1:n_good,1:n_good,index])/(sqrt(2)*0.6325
  }
  i = 1
  for (j in 2:n_good) {
      for (k in 1:(j-1)) {
          Z[i] = D[j,k]
          i = i+1
      }
  }
 Z
}
```

We now define a function that check if the Z trajectory remains within the 0.95 and 0.99 domains:

```
within_domain = function(Z) {
  n = length(Z)
  XX = (1:n)/n
  YY = cumsum(Z)/sqrt(n)
  U95 = 0.3+2.348*sqrt(XX)
  U99 = 0.312+2.891*sqrt(XX)
  c(all(abs(YY)<=U95),all(abs(YY)<=U99))
}
```

Using the functions

For the CTL

```
sapply(1:12, function(i) within_domain(get_Z(i)))
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11]
## [1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      [,12]
## [1,] FALSE
## [2,] FALSE
```

They all exhibit differences low vs high at both confidence levels.

For the patients

```
sapply(1:13, function(i) within_domain(get_Z(i,FALSE)))
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11]
## [1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      [,12] [,13]
## [1,] FALSE FALSE
## [2,] FALSE FALSE
```

They all exhibit differences low vs high at both confidence levels.