Surface Statistics using Caret

Jörn Diedrichsen

Department of Biomedical Engineering / Department of Neuroscience

Johns Hopkins University

720 Rutland Ave, 419 Traylor Building

Baltimore, MD, 21205-2195

phone: (410) 614-8266

Email: jdiedric@bme.jhu.edu

Web: http://www.bme.jhu.edu/~jdiedric/surface_stats.htm

Toolbox available at: http://www.bme.jhu.edu/~jdiedric/surface_stats.htm

Version 1.3: 09/03/05

1. Introduction

Because the neo-cortex is a folded sheet, it appears natural to treat statistical maps on the neo-cortex (fMRI-contrasts, anatomical measures, connectivity etc.) in a 2D representation. One question that arises when evaluating maps of statistical values is how to calculate p-values of certain events on these maps that correct for multiple comparisons. Furthermore, often we would like to attach a p-value not only for the height of a statistical value, but also for the extent of a cluster above a certain height-threshold. To do this, one obvious factor that has to be kept in mind is the spatial dependence of the data (also often called spatial smoothness). Figure 1 shows two Gaussian random fields with different levels of smoothness. These images were generated by using the same random data and smoothing them with a Gaussian kernel of either 3 pixel or 10 pixel width. As you will probably see, the expected random statistics are different for these maps. For example, in the left map you would expect to see more maxima or clusters, where as in the right map you should see fewer, but bigger clusters. Also peak values might be lower in the smoother map, as extreme values are smoothed more.

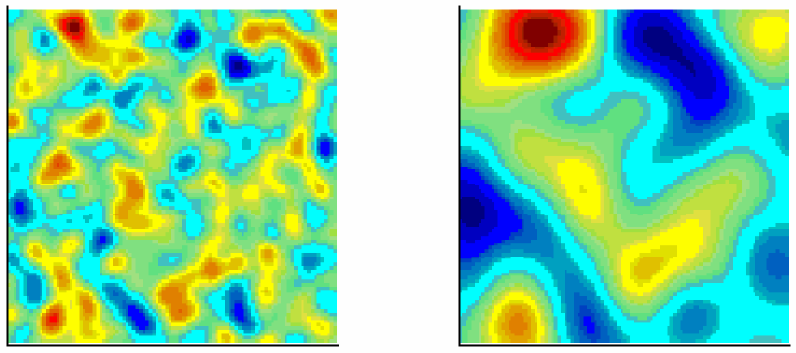

Figure 1. A Gaussian random field with different levels of smoothness. The Graphs were generated by taking a 100 x 100 matrix of independent Gaussian random variables and smoothing them with either a 3 pixel (left side) or 10 pixel wide Gaussian kernel.

The theory of how statistics behave on a random map of different characteristics has been worked out very nicely in the Random field theory (Adler, 1981), an approach that was first applied to brain imaging data by Worsley and Friston (Friston et al., 1994; Worsley et al., 1996). This approach has become an accepted standard for the brain imaging community. Really, the only thing that needed to be done was to implement this theory for 2D-surfaces and write the code for caret.

The technical note will outline how to do statistics for functional (or anatomical) data on a surface with caret. We start with a set of maps that are aligned to the same surface. These are typically the values from functional contrasts from different participants (e.g. movement v. rest). I assume that these individual statistics have been already calculated in 3D using a package like SPM or AFNI and then mapped to a flat surface. Using these individual data, we then can calculate a group statistics (2). We then will estimate the smoothness of the statistical map (3). One challenge here is that the surface is irregularly tessellated with triangles, rather than having a regular lattice-structure such as 3D functional data. Finally I will describe functions and algorithms used to apply Random field theory to arrive at p-values for certain peaks and cluster sizes (4). Finally, we can restrict our search region to specific regions of interest and correct for it (5). I will mention the theory where necessary, but for a more comprehensive treatment I recommend some very well-written introductory papers to the problem (Friston et al., 1994; Worsley, 1996a; Worsley, 1996b; Worsley et al., 1996).

What this technical note does not address: As David van Essen pointed out, one of the big issues with surface-based methods is the question of how confident we are that an activity observed at location (x,y) really comes from this location. In 3D we rarely think about this issue (although we probably should!). But in 2D, distances on the map do not represent distances in the space the data was acquired in. For example, activity observed on one bank of a sulcus may be mapped to the neighboring bank of the sulcus. Many factors play into this question, making a theoretical treatment very difficult: How well is the functional data aligned to the individual anatomical data? What is the resolution of the functional data? What mapping algorithm is used? Was the activation mapped to a standard surface (e.g. the colin brain) or was it mapped to the individual's surface and then aligned with spherical alignment to a standard surface (which in my mind should increase your confidence on locations of activation)? Anyhow, this is a complex question which I do not address here.

2. Calculating statistics

Calculating statistical maps from raw data is pretty straight-forward. So far I have implemented one-sample t-tests for group analysis, as well as a regression analysis. However, the code is laid out in such a way, that any linear model (ANOVA, ANCOVA) will be easy to implement in the future. In general we have *n* observation (subjects or sessions) at *N* location (nodes) on a map, stored in different columns of a metric file. Now we have a general linear model for the data y for each subject *i* and position **x**:

$$y_i(\mathbf{x}) = x_i^T \beta(\mathbf{x}) + \varepsilon_i(\mathbf{x}) \tag{1}$$

We can now calculate the standard estimates for β and resulting t-values for specific contrasts. For a one-sample t-test all x's are 1 and the β-weights are just the mean activation for each Node. The resulting data structure is called a cSPM (caret statistical parameter map, do distinguish it from the similar structure in SPM). To generate a one-sample t-test from the columns [cols] from the metric file <filename> call:

```
cSPM=caret_getcSPM ('onesample-t','data','filename',[cols],…);
```

The data-argument can be repeated, if you want to concatenate columns of different metric files. To get a map that does a simple or multiple regression analysis on the data:

```
cSPM=caret_getcSPM ('regression',x,'data','filename',[cols],…);
```

Where x is a *n x p* Regression matrix .

The cSPM-structure can be saved as a metric-file using the command.

```
caret_savecSPM('filename',cSPM);
```

This metric contains *1….n* columns for the data, the *1..p* beta-values (for the one-sample t-test that's only the mean), the Residual variance, and the statistics (One for every contrasts, by default a t-test for each beta value of the hypothesis b=0). I recommend saving the cSPM structure in matlab, to retain the other fields (STAT, df, etc) as well.

3. Estimating smoothness

3.1. Algorithm

To apply Gaussian field theory to obtain cluster-wise p-values, we need to estimate the smoothness of the statistical map, using the residuals of the model. The necessary theory is laid out in Worsley (Worsley, 1996a). The roughness of a map $\mathbf{\Lambda}$, a *DxD* matrix, is defined as the variance of the first derivative of the noise in each dimension of the map. *D* is the dimensionality of the space the data is in (2 for a surface).

$$\mathbf{\Lambda} = \mathrm{var}\left(\frac{\partial \varepsilon(\mathbf{x})}{\partial \mathbf{x}} \right) \qquad (2)$$

Where $\mathbf{x}$ is a Dx1 vector that indicates the coordinate-location of the point.

Assume we have a linear model of the data at each position (we made n observations). We can approximate the noise term ε through the residuals $r_i(\mathbf{x}) = y_i(\mathbf{x}) - x_i^T \hat{\beta}(\mathbf{x})$ of the model. These residuals are normalized for each position $\mathbf{x}$ separately:

$$u_i = r_i \left(\sum_{i=1}^{n} r_i^2 \right)^{-1/2} \qquad (3)$$

It then can be shown (Worsley, 1996a) that

$$\hat{\lambda}_{jk} = \frac{n-p-1}{n-p} \frac{1}{N} \sum_{\mathbf{x}} \sum_{i=1}^{n} \frac{\partial u_i(\mathbf{x})}{\partial \mathbf{x}_j} \frac{\partial u_i(\mathbf{x})}{\partial \mathbf{x}_k} \qquad (4)$$

is an unbiased estimator for the th entry of $\mathbf{\Lambda}$. *p* is the rank of the linear model, thus *n-p* is the degrees of freedom.

Now how do we find the local derivatives of the maps in each dimension? This is easy on a regular lattice-representation (like a voxel-based volume), the estimates of the derivatives in directions x,y, and z are just the differences between neighboring voxels in this direction. However, caret-surfaces are irregularly tessellated.

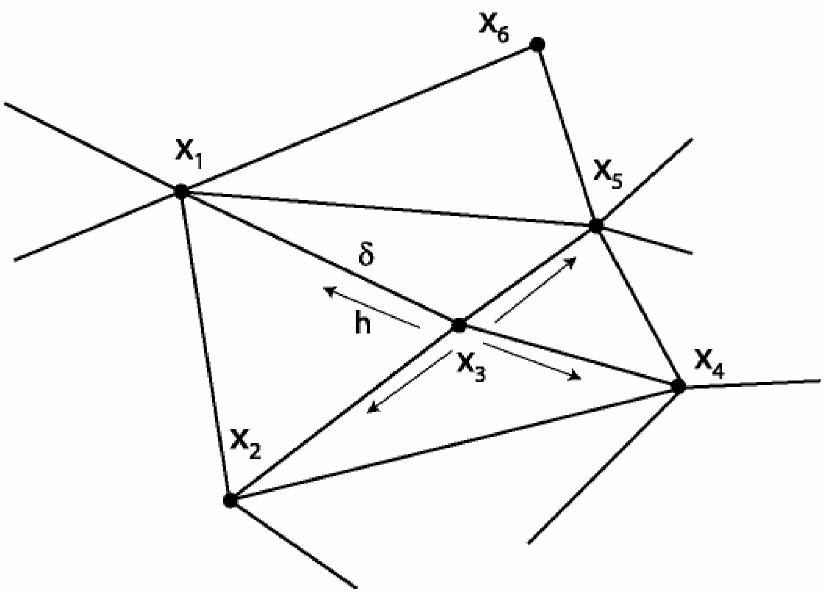

Figure 2. Estimating smoothness on irregularly tessellated surfaces. The smoothness is estimated in each direction h and finally integrated using a linear model.

Figure 2 shows an example of this. The surface is tessellated with non-overlapping triangles, each one between 3 Nodes. Each Nodes has a coordinate **x**, either in 3-D (fiducial) or in 2D (flat) space. Each pair of Nodes is connected by an Edge with length δ and direction **h**, a D-dimensional unit-vector. Then, each difference in between the residuals of neighboring Nodes is providing us with an estimate of derivative in that direction, namely of $\mathbf{h}^{\mathrm{T}}\mathbf{\Lambda}\mathbf{h}$.

$$\Delta = \frac{n-p-1}{n-p}\sum_{i=1}^{n}\left(\left(u_i\left(\mathbf{x}+\delta\mathbf{h}\right)-u_i\left(\mathbf{x}\right)\right)/\delta\right)^2 \tag{5}$$

Finally we can use linear regression to obtain estimates for all the unique entries of the symmetric matrix $\mathbf{\Lambda}$ averaged over all M pairs of neighboring nodes (Worsley, 1996a):

$$\begin{bmatrix} \mathbf{\Lambda}_{11} \\ \mathbf{\Lambda}_{22} \\ \mathbf{\Lambda}_{12} \end{bmatrix} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)\mathbf{H}^{\mathrm{T}}\mathbf{\Delta}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_{1,1}^2 & \mathbf{h}_{2,1}^2 & \mathbf{h}_{1,1}\mathbf{h}_{2,1} \\ \ldots & \ldots & \ldots \\ \mathbf{h}_{1,M}^2 & \mathbf{h}_{1,1}^2 & \mathbf{h}_{1,M}\mathbf{h}_{2,M} \end{bmatrix} \tag{6}$$

$$\mathbf{\Delta} = \begin{bmatrix} \Delta_1 \\ \ldots \\ \Delta_M \end{bmatrix}$$

For three dimensions, the matrices would be expanded correspondingly. Smoothness is often expressed in terms of FWHM which is:

$$FWHM = \sqrt{\frac{4\log 2}{\det(\mathbf{\Lambda})}} \tag{7}$$

3.2. Implementation

This algorithm is implemented in the function:

```
cSPM.FWHM= caret_estsmoothness(cSPM,S);
```

Where S is a surface structure that contains the information of a coord-file and topo-file. You can generate this structure by calling.

```
S=caret_getsurface('coordfilename','topofilename');
```

4. P-values

4.1. Theory

The theory of cluster-wise p-values on statistical maps is well laid out (Friston et al., 1994; Worsley et al., 1996). The center-piece of the theory are three random variables. First, *N* is the area (in mm^2, etc) that has a statistical value (for example a *t*-value) above a certain threshold *u*. Secondly, *m* is the number of clusters that are above that threshold. Finally, n_i is the area of the i^th cluster.

The expected value of *N* is simply the size of the area where the statistical value of interest (*Z)* is above a threshold *u*. Thus, it is the size of the search area times the probability of *Z*>*u*. For the expected numbers of clusters, we use an approximation based on the expected Euler characteristic

(EC) of the area above threshold, the so-called excursion set (Worsley, 1996b). For high threshold, the EC is counting the numbers of unconnected excursion sets, or clusters. The expected EC can be calculated for each statistical map, and depends on the type of the statistical map (e.g. a *t*-map or a *F*-map), the threshold *u*, the smoothness of the map, and the size and shape of the search region.

To calculate the expected EC for a certain threshold u, we first have to describe our search region in terms of Resel-counts. A Resel is a surface patch of the size FWHM*FWHM. For a Surface the Resel-counts are defined as:

R_0: EC of the search region

R_1: ½ Circumference of the search region / FWHM

R_2: The area of the search region / FWHM*FWHM

From **R**, the type of statistics and the threshold, the expected EC can be calculated (Worsley et al., 1996). This gives us the expected number of clusters (*E(m)*) on the surface. Finally the expected cluster size is approximately *E(n)=E(N)/E(m)*, assuming independence of *N* and *m*.

To get p-values of certain event, we need not only the expected values of these random variables, but also their distribution. For high values of the threshold *u* and the special case of a 2-D surface we also have a simple results for the distribution of *m* and *n*: The numbers of clusters are approximately Poisson distributed with mean *E(m)*:

$$P(m=c) \approx \frac{1}{c!} E(m)^c \, e^{-E(m)} \tag{8}$$

The size of each cluster follows approximately an exponential distribution:

$$E(n>k) \approx e^{-\frac{k}{E(n)}} \tag{9}$$

(Note that in general $n^{2/D}$ but not n is exponentially distributed, (Adler, 1981; Friston et al., 1994)).

Here we are! We just calculate the expected value for the three random variables *N*, *m*, and *n* for a certain height-threshold, search region, smoothness and statistics and we know their approximate distributions. Now we can calculate three different types of p-values: First, we can calculate for how likely it is to observe at least one voxel that is above a certain value for *t*-, *Z*-, *Chi*-square and *F*-statistics. This would give us a p-value for a single voxel above a certain threshold adjusted for multiple comparisons (a so-called corrected p-value).

$$p(m>0)=1-e^{-E(m)} \tag{10}$$

Secondly, we can also calculate the probability how likely we would observe one cluster of a certain size above a threshold u. This gives us a cluster-wise p-value.

$$p(n_{\max}>k)=1-\exp\left(-E(m)\exp\left(\frac{-k}{E(n)}\right)\right) \tag{11}$$

Finally, we can even calculate how likely *c* clusters of a certain size *k* (or higher) are, that is, if we care to do so…. (Set-level inference, Friston et al., 1994).

4.2. Implementation

`Table=caret_list(S,cSPM,u,k,[contrast]);`

This function implements the generation of cluster-wise and corrected p-values (and is named after the corresponding function in SPM). This function consists of the following steps: First it calculates the area, circumference and Euler Characteristic of the Search area S (if this hasn't been done yet). It also checks if the smoothness of the statistical map has been estimated and does so, if necessary. As a next step it will find the clusters of super-threshold values, using the routine `caret_clusters`. This routine relies on the fact that the surface structure S has a list for each Node with its neighbor-nodes. `caret_neighborhod` is the function that does this calculation. After the clusters have been found, caret_P is called to calculate the expected number of clusters, and a list of corrected and cluster-wise p-values is generated and printed on the screen. The table is also returned as a data structure. Here a description of the sub-functions in particular:

`S=caret_calcarea(S)`

This function calculates the area of all tiles on the surface and assigns 1/3 of the area of each tile to a neighboring Node. This function deals with some of the topological errors in a surface (see Topology -> Topology error report). For example nodes that have no surface area connected to them will be called `bad_nodes` and will be ignored. The function then calculates the circumference by checking how often Edges are occurring throughout the Set of Tiles. Edges that are on the circumference will occur once, Edges within the surface twice. Thus the length of the circumference can easily be found. In case of topological artifacts, Edges can sometime occur three times (degenerated links), in which case the Tiles causing the defect will be ignored.
The EC of the search area is then

EC = #Nodes - # of unique Edges + # of Tiles.

Note that a full flat map should have EC = 1, a flat map with a hole should have an EC of 0 and a spherical surface without any holes should have an EC of 2.

```
Index = caret_clusters(S,Z>u)
```

This function finds the clusters above a certain threshold: it uses a recursive dynamic programming algorithm. So for big clusters you probably have to increase your recursion limit to a couple of thousand. Index is a variable that holds a assigned number of the cluster that each Node belongs to. Unfortunately, for this algorithm to work, we need a list of all the Neighbors of each node. These are actually saved for some `topo`-files, but can be generated with `S=caret_calcneighbor(S)`, which is computationally expensive. (Caret does have a function that does this more efficiently). However, once you have calculated the neighborhood, you can save your surface as a mat-file. Now, caret_clusters will work very fast using this saved information.

```
[P,p,Em,En,EN]=caret_P(c,k,u,df,STAT,R)
```

Finally returns the corrected (P) and uncorrected (p)-value of observing c clusters of at least size k above threshold u, given a statistics of form `‘T’,’F’,’X’,’Z’` with df degrees of freedom. Of course this function is shamelessly stolen from SPM.

Here are some examples for the usage of caret_P:

`caret_P(1,0,Z,df,STAT,1)` :uncorrected p value

`caret_P(1,0,Z,df,STAT,R)` :corrected p value based on height Z

`caret_P(1,k,u,df,STAT,R)` :cluster-wise p value based on extent k at u

Caret-list is generating a tab-delimited table that can be printed with

```
caret_list(‘txtList’,Table);
```

The list presents all the clusters, the area of the cluster (size of all the Nodes in that cluster) , the maximum of the cluster, the uncorrected and corrected P-value of that maximum, and a cluster-wise p-value based on extend and threshold. The Footer of the Table contains useful information, as the expectations over m,n, and N, smoothness, etc.

5. Small volume correction

Often we want p-values not over a whole hemisphere, but want to restrict our attention to some pre-specified region of interest (ROI). These can be defined anatomically or through other (but please independent!!!!) contrasts on the same data.
The small-volume correction is applied by creating a subset of the original surface, based on a functional criterion or a paint file that indicates the location of a ROI.
Subsets of surfaces can be generated by
```
[SubS]=caret_surfacesubset(S,boolean);
```
Where `boolean` is a logical value, either generated by a functional threshold, an anatomical paint-file, a combination of those, etc….
When generating the subset of the surface, the function discards Nodes that have no Tiles (and thus no surface area) attached to them. The area and Neighborhood of the subsurface has to be again calculated using `caret_calcarea` and `caret_calcneighbor`.

6. Monte Carlo Studies

To test the validity of the assumption, I ran the following Monte Carlo study.

6.1. Methods

I generated an artificial caret-surface that consists of a rectangle regularly tessellated with even-sided triangles. Each Node is connected to 6 Neighbor Nodes. The surface has a size of 100 x 86.6 mm, consists of 9950 Nodes and 19503 Tiles, each of the size 0.43 mm^2. In comparison, the left PALS surface consists of 73730 Nodes, and 141908 Tiles with an average size of 0.65 mm^2. On this surface, I generated sets of 10 normally distributed random fields that were smoothed with a Gaussian kernel with FWHM of 3, 6, 9, 12, and 15 mm. For each set I calculated a t-test with 9 df to test the Null-hypothesis mean=0. I repeated this 250 times for each smoothness-value. For each SPM, I estimated the smoothness, calculated expected and observed values for N, m, and n, and assigned voxel-wise and cluster-wise corrected p-values to the excursion set at threshold values of $t>3.5$ and $t>5.5$. Because the underlying data is random with known smoothness, the estimated values should fit the observed values, and events with a probability of $p<0.05$ should occur on less than 5% of the generated maps

6.2. Results

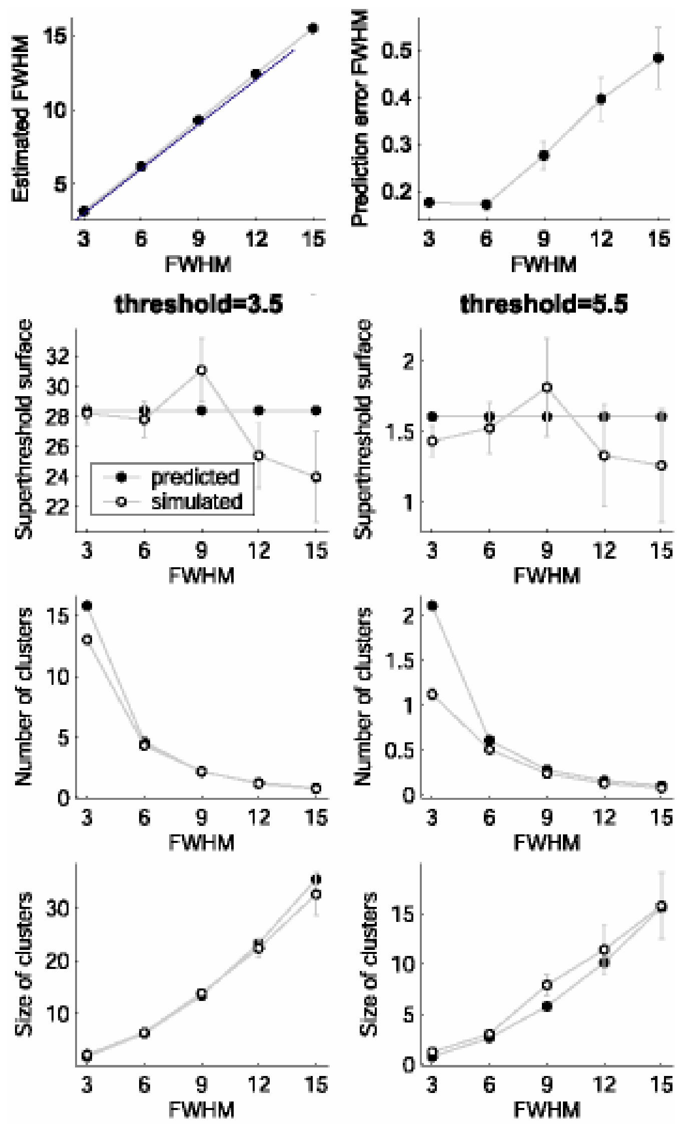

Figure 3. Result from the Monte Carlo study. First row: Estimated smoothness and error in the estimated smoothness. Second row: estimated and observed amount of superthreshold surface (N), for a threshold of 3.5 and 5.5. Third row: estimated and observed number of clusters (m). Forth row: estimated and observed size of clusters (n).

Figure 3 shows the results of one run of this Simulation. The estimated smoothness fits the actual smoothness quite well, however, there is a small but significant overestimation for the degree of smoothness for bigger FWHM. This is most likely caused by the relatively small surface we have. Naturally, the estimated and observed superthreshold surface is matching (second row). However, because the surface is relatively small, the observed values are quite noisy for higher smoothness, as the number of independent observations per surface (resels) goes down. The third row shows the estimated and observed number of superthreshold clusters. The estimate for *m* is too high for

very low smoothness of the map, but accurate for higher smoothness. Finally (forth row), the size of the clusters is relatively accurate, but shows violations for the lowest level of smoothness.
I found one violation of the threoretical assumption made in Section 4.1. The random variables *N* and *m* were not independent, but showed a correlation of 0.6 on average. I assume that this correlation will only get lower with bigger surface-sizes, but probably remain substantial for a surface of the size of the human cortex. Given this violation, will the p-values be accurate?

I tested this by counting the number of surfaces, that showed at least one excursion set that had a corrected voxel-wise p-value of smaller than 0.05 or a cluster-wise p-value of smaller than 0.05. If the p-values are accurate, this number should not exceed 0.05.

Table 1. False positive rates for the Monte Carlo study. Table indicates the Proportion of maps that show at least one maximum that had a p-value of smaller than 0.05, either the corrected voxel-wise or cluster-wise p-value at different levels of threshold.

	Proportion of maps with at least one p<0.05			
Smoothness	Corrected voxel-wise	Cluster (t>3.5)	Cluster (t>4.5)	Cluster (t>5.5)
3	0.008	0.084	0.052	0.064
6	0.048	0.092	0.068	0.056
9	0.028	0.08	0.072	0.052
12	0.036	0.06	0.032	0.04
15	0.036	0.04	0.04	0.04

For the lowest degree of smoothness, the voxel-wise p-value is too conservative (related to the overestimation of the mean number of clusters *Em*) and the cluster-wise p-values are too liberal (related to the underestimation of *En*).
For higher smoothness values this is much more accurate, however for lower threshold and lower levels of smoothness the cluster-wise p-values might be to liberal, while the voxel-wise p-values are a little bit too conservative. However, we see that the stability of the of cluster-wise p-values

6.3. Discussion

Overall the match between expected and observed parameter values for N, m, and n is encouraging. Despite violation of some assumptions (e.g. independence of N and m), the resulting p-values are relatively close. Only for the lowest level of smoothness, did I observe drastic inaccuracies in the p-values. This reinforces the notion that the maps have to have a certain amount of smoothness imposed on them through smoothing. This is best done in caret AFTER the mapping from the 3d data to avoid smoothing across sulcal boundaries. For the simulation it seems that the result are fairly accurate if one resel ($FWHM^2$) corresponds to ca 60 Tiles. That it for an average tile size f 0.65 for the cortical PALS surface, a smoothness of 6.1 or greater is sufficient. In my experience this value is reached by smoothing in caret for 4 iterations with strength of 0.5.

The Monte-carlo study also exposes weaknesses of this technique. The voxel-wise correction is always on the conservative side and sometimes can even be more conservative than the Bonferroni correction, which should provide a lower limit on the p-values. The cluster-wise p-values are more accurate, but the variance observed in the false-rejection rate is typical for this technique (Nichols and Hayasaka, 2003). Therefore, permutation statistics may yield more accurate results.

7. Which Surface?

Independent of the statistical method used, one important issue is which surface is used to calculate the area of the activated clusters. Given the geometric distortions of the flat map or spherical representation, reporting clustersizes on these would be subject to these distortions and produce invalid results. Specifically, areas on the fringe of the flat map would appear bigger than areas in the center of the flat map and therefore would become more easily significant. One solution would be to take the average fiducial map in some atlas space, e.g. the file `PALS.L.avrg.FIDUCIAL.SPM2.coord`. This map has the advantage to give the coordinates of the maxima in the chosen atlas space.

The problem with the average coordinate file is depicted in Figure 4. The figure shows the log-ratio of the area of each node, calculated on the average fiducial (`PALS.L.avrg.FIDUCIAL.SPM2-.coord`), and the average area of each node calculated on each individual fiducial (the fiducials of the 12 PALS-individuals). As can be seen, the average fiducial surface shows compression of (for example) the intraparietal sulcus. The reason for this is that sulci that show large inter-individual variability will be averaged out and the area contained in them will be compressed.

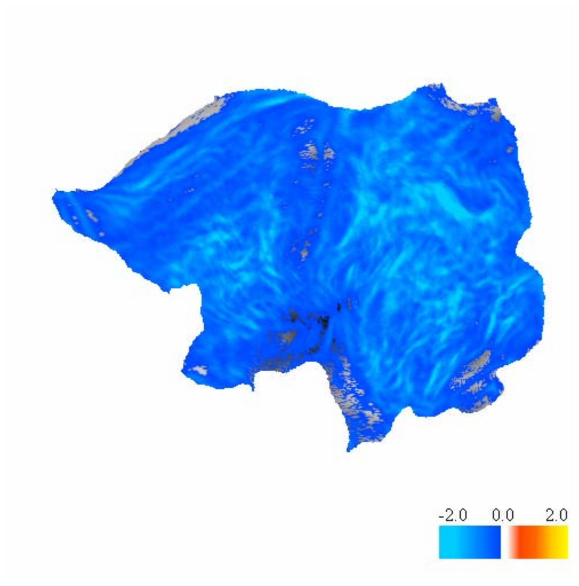

Figure 4. Geometric areal distortion of the average fiducial surface. Plotted is log (area of Node on average Fiducial Surface / average area of Node on individual Fiducial surfaces). There is an overall compression, especially pronounced in areas of high sulcal variability.

Therefore, the usage of the average coordinate file would lead to underestimation of activation regions in the intraparietal sulsus.

I suggest the following solution: The files PALS.L.Avrg.SPM2.mat and PALS.R.Avrg.SPM2.mat contain the average of the fiducials from the 12 PALS atlas surfaces. The coordinates of each Node are averaged here, as in the file `PALS.L.avrg.FIDUCIAL.SPM2.coord`. Instead of calculating the area of each tile using the average coordinate, however, the area of each tile is the average area of the individual surfaces. That is, the map is by definition distortion free. A tile with 1mm^2 surface on the average surface will have on average a 1mm^2 surface on the individual fiducial surfaces. We also compute the length of each Edge by averaging the length of each edge across participants. This comes at the sacrifice of an "inconsistent" surface. That is, If one were to recomputed the average area of the tiles from the average Node coordinates, (using `S=caret_calcarea(S,'force')`, the old distorted representation would result (Figure 4). The average PALS surfaces have been generated with the function `caret_avrgPALSsurface`.

8. Examples

In this example, we will use a data set for the left hemisphere that has been reported in (Diedrichsen et al., under revision). The study involved reaching movements under target-jump or visual rotation conditions. Each individual hemisphere was reconstructed using SureFit and then aligned to the PALS atlas surface. We will use the execution error contrast from Experiment 1, a statistical result reported in Table 2. The underlying data files can be found at http://www.bme.jhu.edu/~jdiedric/download/caret_example.zip .

8.1. Generating and preparing a surface

We will use the average fiducial surface, as described in section 7, generated by the m-file `caret_avrgPALSsurface`. To estimate the smoothness of the statistical map, however, we also need one example fiducial surface that is consistent, in which the relationship between nodes, edges and areas is preserved. We will get the surface by specifying coord and topo file, and estimate the area of the surface.

```
S=caret_getsurface('Human_Buck_Case2.L.F.RegToPALS_B12.LR.FIDUCIAL_SPM2.73730.c
oord',' Human.sphere_6.LEFT_HEM.73730.topo');
S=caret_calcarea(S);
save PALS.L.Subj2.surface.mat S
```

8.2. Making a statistical map

In the following code we will generate a statistical map for a one-sample t-test over columns 1-13 of a metric file. Each column represents the percent signal change at that Node for one individual participant. The metric file was generated by mapping the un-smoothed contrast for a first-level analysis onto the individual's fiducial surface. These fiducial surfaces were then aligned using 6 landmarks to the PALS surface and the deformation applied to the metric file. Each individual contrast was smoothed in 2D (4 iterations, strength 0.5).
After a one-sample t-test is calculated, we will estimate the smoothness of the map using the residuals from the t-test. The smoothness is stored in the field cSPM.FWHM and indicates a smoothness of ca. 6.2 mm. We then save the cSPM as a metric file, load it into caret and can see the t-values. I adjusted the threshold to t=3.61 on the caret-display (under the D-C/Matric-tab) and displayed positive and negative values, so that it matches my statistical threshold for the map in caret_list (Figure 5). Finally, a list of corrected and cluster-wise p-values is generated.

```
cSPM=caret_getcSPM('onesample_t','data','C57.L.full.perc21.metric',[1:13]);
load PALS.L.Subj2.surface.mat
cSPM.FWHM=caret_estsmoothness(cSPM,S);
cSPM.FWHM
```

```
ans =

    6.8958
    5.9447
    5.8035

caret_savecSPM('Contrast_21.metric',cSPM);
load PALS.Avrg.L.SPM2.mat
caret_list(S,cSPM,3.61,17,'sort_by','area','sign',-1);



STATISTICS: one-sample t-test: mean<0
==============================================================================
c      NumN   Area   max(Z) p(unc) p(cor) p(cl.) X(mm)  Y(mm)  Z(mm)
------------------------------------------------------------------------------
4        94   167.08 6.113  0.000  0.685  0.000  -55.16 -22.43 51.12
3        74   128.65 6.505  0.000  0.517  0.000  -44.74 -31.35 44.44
1        34   50.36  4.586  0.000  1.000  0.010  -31.88 -53.18 63.40
5        13   28.02  5.911  0.000  0.771  0.284  -43.57 17.30  43.01
2        16   17.44  7.078  0.000  0.315  0.834  -29.51 -45.02 66.19
------------------------------------------------------------------------------
Height threshold: T = 3.61, p = 0.002 (1.000)
Extent threshold: k = 17.00 sqmm, p = 0.067 (0.854)
Expected sqmm per cluster, <k> = 6.298
Expected number of clusters, <c> = 1.92
Expected surface above threshold (sqmm) = 180.02
Degrees of freedom = [1.0, 12.0]
Smoothness FWHM = 6.2 (mm)
Search vol: 100582 sqmm; 2619.7 resels
==============================================================================
```

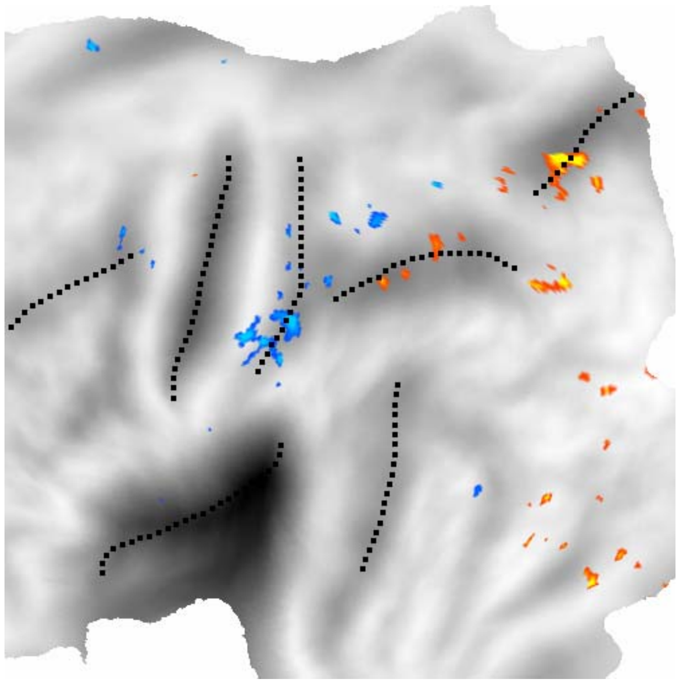

Figure 5. T-map calculated from data 'C57.L.full.perc21.metric', thresholded at t(12)>3.61.

The bottom of the table list expected values of k (average size of clusters – above or below size threshold), m (number of clusters ABOVE the size threshold), and N (total expected surface above threshold) for the given height threshold of T=3.61. It also gives uncorrected (and corrected) p-values for the thresholds used. The estimated smoothness of the map was 6.2 mm, generated by smoothing of the data (Neighbor, 4 iteration, strength 0.5), plus the inherent smoothness due to the resolution of the functional images. The search volume is the total hemisphere (100582 sqcm) or 2619 resels (one resel is FWHM*FWHM sqmm big).
Three maxima are significant by themselves: Two sites in the post-central sulcus and a site in the superior parietal cortex.

8.3. Using small-volume correction

In this example and in the article, we will restrict our search region to a subset, namely to regions that show higher activity during reaching movements than during rest. This contrast is stored in the file `C57.L.full.perc23.metric`.
In general, the search region can be constricted by any criterion, based either on an anatomically defined ROI (indicated by a paint file), or by a functional contrast. The only condition is that the restriction of the search region has to be independent (!) to the contrast of main interest.

First we need to generate a statistical map of the movement > rest contrast:

```
cSPM=caret_getcSPM('onesample_t','data','C57.L.full.perc23.metric',[1:1
```

The we load the average fiducial surface and generate a subset of it. The edges of this subset are then recalculated and the Neighborhood relationships as well for faster clustering. Be aware that we do not recalculate the area of the Nodes - otherwise we would have to have used `SubS=caret_calcarea(SubS,'force')` – but just take over the average area from the individual surfaces.

```
load FIDUCIAL_AVRG_surface.mat
SubS=caret_surfacesubset(S_L,cSPM.con.Z>0);
SubS=caret_calcarea(SubS);
SubS=caret_calcneighbor(SubS);
```

We could have also produced a Subset based on a anatomical ROI:

```
M=caret_load('my_ROI_paint_file.paint');
SS=caret_surfacesubset(S,M.data(:,ROI)==1);
```

Using the functional defined Surface subset, however, the result becomes:

```
caret_list(Sub,cSPM,3.61,17,'sort_by','area','sign',-1);
```

```
STATISTICS: one-sample t-test: mean>0
================================================================================
  c    NumN   Area   max(Z) p(unc) p(cor) p(cl.) X(mm)  Y(mm)  Z(mm)
--------------------------------------------------------------------------------
  4      94   167.08 6.113  0.000  0.476  0.000  -55.16 -22.43 51.12
  3      74   128.65 6.505  0.000  0.333  0.000  -44.74 -31.35 44.44
  1      34   50.36  4.586  0.000  0.989  0.002  -31.88 -53.18 63.40
  2      16   17.44  7.078  0.000  0.189  0.511  -29.51 -45.02 66.19
--------------------------------------------------------------------------------
Height threshold: T = 3.61, p = 0.002 (1.000)
Extent threshold: k = 17.00 sqmm, p = 0.046 (0.539)
Expected sqmm per cluster, <k> = 5.517
Expected number of clusters, <c> = 0.77
Expected surface above threshold (sqmm) = 93.06
Degrees of freedom = [1.0, 12.0]
Smoothness FWHM = 6.2 (mm)
Search vol: 51994 sqmm; 1354.2 resels
================================================================================
```

Compared to the last list, the search region has about halved, and cluster 5 (which was not significant anyway) has disappeared from the list, as it is outside of the search region.
In the study (Diedrichsen et al., under revision), I corrected not only for the search region of one hemisphere, but corrected for the search region of both hemispheres. I generated Sub-surfaces (positive movement against rest contrast) for the left and the right surfaces, and then added the EC from both areas:

```
SS_L; % Subsurface of left hemisphere
SS_R; % Subsurface of right hemisphere
A=SS_L.A + SS_R.A; % Combined surface of both hemispheres
SS_L.A =A;
SS_R.A=A;
```

9. Disclaimer

These functions are work in progress. I will be very happy for feedback, bug-reports, tests and proposals for extensions (also for comments to improve this manuscript). The code and this document are freely distributed but should be regarded as copyrighted, distributed under the terms of the GNU General Public License. The code and example data set can be downloaded from http://www.bme.jhu.edu/~jdiedric/download/caret.

10.References

Adler RJ (1981) The geometry of random fields. New York: Wiley & Sons.

Diedrichsen J, Hashambhoy YL, Rane T, Shadmehr R (under revision) Neural correlates of reach errors. J Neurosci.

Friston KJ, Worsley KJ, Frackowiak RSJ, Mazziotta JC, Evans AC (1994) Assessing the significance of focal activations using their spatial extent. Human Brain Mapping 1:214-220.

Nichols T, Hayasaka S (2003) Controlling the familywise error rate in functional neuroimaging: a comparative review. Stat Methods Med Res 12:419-446.

Worsley KJ (1996a) An unbiased estimator for the roughness of a multivariate Gaussian random field. In. Montreal: Department of Mathematics and Statistics, McGill University.

Worsley KJ (1996b) The geometry of random images. Chance 9:27-40.

Worsley KJ, Marrett S, Neelin P, Vandal AC, Friston KJ, Evans AC (1996) A unified statistical approach for determining significant voxels in images of cerebral activation. Human Brain Mapping 12:900-918.