# SamSrf for pRF models

**IMPORTANT NOTE: As of SamSrf X, this manual has been deprecated! Most of the information in this cookbook probably still applies, but some of it is obsolete and has been superseded by more efficient ways to do the analysis.** We have corrected any glaring errors we could find in this as we could find but cannot guarantee that there aren't any more and if so, whether we can fix them. There are only so 24 hours in a day.

**The manual for the SamSrfAnalysis GUI explains the basics of what you need for an analysis.** Naturally, you can still run *SamSrf* functions directly in Matlab using the wrapper functions in **SamSrf/Models** and this is still the recommended approach for advanced users. The **ModelHelp** and **FuncHelp** tools can tell you more up-to-date information about how that works.

## BASIC INFORMATION

This cookbook is a generic manual indicating how to conduct a simple population receptive field (pRF) experiment using SPM, Freesurfer and SamSrf for data analysis. Currently, SamSrf stands for **Seriously Annoying Matlab Surfer**. If you come up with a better acronym that meets Sam's approval, you will win a prize and the acronym will be changed in your honour.

Note that this guide has been written in part from the perspective of a UNIX-based system user (e.g. Linux, MacOS). If you are working in Windows, working through a virtual machine may be necessary although as of Windows X it is possible to run the UNIX terminal scripts directly in Windows. This guide assumes basic knowledge of MATLAB, SPM, and of the UNIX command line.

**Important note to users of SamSrf versions prior to 6: A lot of things have changed in SamSrf 6. In particular, the GUI has been removed.** Most people will use scripting sooner or later for most of their analysis pipeline and the GUI started to become increasingly complex and it was not worth the effort supporting it any longer. There are now a few GUI tools available for displaying maps and delineating ROIs, and a documentation GUI for the parameters you specify in the Model structure.

Required dependencies:
- **MATLAB 64-bit** *(version R2020a or later*
  *earlier versions & 32-bit versions have not been tested)*
- **Statistics** and **Optimization** *toolboxes from MATLAB (only for some functions)*

Not required but we use:
- **SPM8** or **SPM12** *for pre-processing (but you could use any other package)*
  *& if using SamSrf for surface projection*
  *There is also a patch for loading NII files*
  *using MATLAB's native functions instead.*
- **FreeSurfer 6.0.0** *or later – not strictly required but it's what we use for surface models*

**Note that SamSrf saves data MAT files in format 7.3.** This should not be a problem for most people but **if your Matlab cannot read the MAT files** generated by SamSrf you need to change the default setting in *File->Preferences->General->MAT-Files*

**Overview of pRF pipeline**

Before delving into SamSrf analysis proper, let's look at the overall plan for data analysis in pRF modelling. The analysis takes place in three component streams: pre-processing in an fMRI package of your choice (e.g. SPM, FSL), surface reconstruction in FreeSurfer, and pRF modelling in SamSrf. Between these streams, the whole analysis can be broken down into 5 main steps:

1. fMRI pre-processing (in any package)
2. Anatomical reconstruction (FreeSurfer)
3. Surface-projection of functional data (either SamSrf or FreeSurfer)
4. pRF modelling with inputs from all previous steps (SamSrf)
5. Displaying results, area delineation, and quantitative analysis (FreeSurfer or SamSrf)

# CHAPTER 1: ACQUIRING DATA

In order to perform a pRF experiment you will need to acquire the following MRI scans:

**Echo-planar imaging (EPI)**
- One or more runs of a visual mapping stimulus
  - In this example, we use 2 runs of geometric bars in the following configuration:
    1. Bars traversing the visual field in four different directions
    2. The same bars but sweeping in a different order
- (Optional) You may collect one run of sparse visual stimulation for estimating hemodynamic response functions (HRF) but we will normally use the canonical HRF
- (Optional) You may want to collect B0 field maps to correct distortion in your EPI scans (probably more important for 3T and 7T data than 1.5T)

**Structural images**
- A high-resolution T1-weighted anatomical image with the full 32-channel head-coil on.
- (Optional) If acquiring EPI scans with the top part of the head-coil off, a short T1-weighted anatomical image also with the top coil section off. Generally, when reusing an older anatomical scan for your participant, it is advisable to acquire a fast structural to aid coregistration with the old scan on which your FreeSurfer reconstruction is based.

# CHAPTER 2: DIRECTORY STRUCTURE

For each of your subjects, you will need to create an organised directory structure to perform the analysis. See Figure 1:
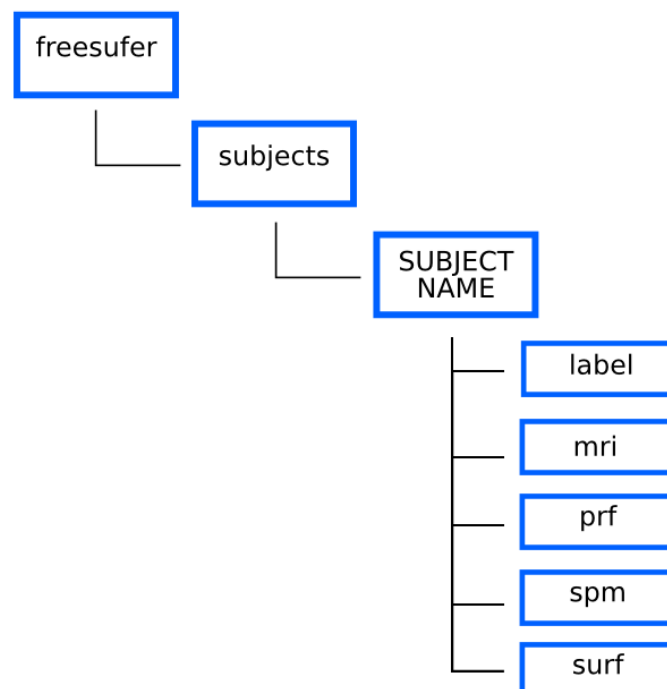


*Figure 1*

The analysis for each subject is contained within a directory specific to that subject. Within the subject directory you will find:

> *label* contains the surface labels that FreeSurfer creates during reconstruction/parcellation
> *mri* contains the volume information and the original T1-weighted image reconstructed
> *surf* contains the surface reconstruction information from FreeSurfer
> *spm* contains the files for the pre-processing steps will reside (you can choose this name)
> *prf* is the population receptive field analysis directory (you can choose this name)
> *anatomy* is the folder where the anatomical meshes for the subject will be stored

**Note that most of your directory structure will be created when you run recon-all in Freesurfer (step 3), except the folders *spm, anatomy,* and *prf*. The folders *spm* and *prf* will need to be created manually and you can choose the names for these.** (In theory, you can also keep those folders somewhere else if you so desire, but this cookbook assumes you leave them with the FreeSurfer data). The folder *anatomy* is created when you surface-project data.

If you wish to carry out more than one pRF experiment in a single subject (e.g. two conditions), then a sensible step might be to create two folders (prf_1 and prf_2) instead of one, and treat them independently from chapter 4 onwards. Alternatively, you can also keep data from different conditions in the same folder but name the analyses with different, descriptive names.

**Note that the subjects folder shown here is inside the Freesurfer installation. However, you can use any arbitrary folder as your *subjects* folder,** e.g. when you have different projects and you don't want the data to be mixed. To change the *subjects* folder you need edit *SetUpFreeSurfer.csh*, which is located inside your main Freesurfer folder. You will find a line that says something like:

> setenv SUBJECTS_DIR /usr/local/freesurfer/subjects

You can change the path to wherever you want to keep your data. However, you have to make sure that various folders FreeSurfer wants (from the original *freesurfer/subjects* folder) are present.

## CHAPTER 3: PREPROCESSING

1. **fMRI pre-processing**
   Copy your data to SUBJECT/spm and perform your standard pre-processing for fMRI images in your software of choice.

   In the case of data acquired at the FIL, we recommend the following steps in SPM8:
   - DICOM import*
   - Bias correction (only when using 32-channel head coil)
   - Remove dummies
   - Optional: Create B0 field maps
   - Realign & unwarp (with or without B0 field maps)
   - Optional: Slice-timing correction (Karl Friston will tell you it's useless)
   - Co-registration to the structural
   - Optional: Smoothing (but this is best done on surface at cortical projection stage)
   - 4D merge (in some centres this is the normal output from the scanner)

An expanded explanation of these steps is provided in Appendix A.

From your pre-processing, you should end up with a single 4D file for each stimulation run in NiFTI format. In this example, we have 3 files: 1$^{st}$ run of fMRI (bars_1), 2$^{nd}$ run of fMRI (bars_2) and the HRF scan.

You also need to convert the T1-weighted structural from .hdr/.img to .nii.

## 2. Run recon-all
Locate the T1-weighted file and run the automatic reconstruction:

    recon-all –i /path/to/T1.nii -s SUBJECT -all

Note SUBJECT is going to be the subject name in the *freesurfer/subjects* directory.

Additionally you can use the -label_v1 option in this command to derive an anatomical prediction of V1 (based on Hinds et al., 2008, NeuroImage). This does not work on all versions of FreeSurfer. We tested SamSrf extensively on 6.0.0 but we always recommend using the latest version.

## 3. Inspect surface
Check that the reconstruction worked in each hemisphere with tksurfer. Example:

    tksurfer SUBJECT lh inflated

Instead you can also do this in SamSrf, using the SurfaceProjection tool and by projecting data to the surface and opening that in DisplayMaps.

## 4. Create occipital labels
This is optional but advised for saving time. There are two ways to create the occipital ROI.
1. The default in SamSrf is now to do this automatically in MATLAB using the MakeOccRoi command (in *SamSrf/Utils*). This reads in the surface data created by recon-all and restricts the occipital ROI label to the posterior vertices.
2. Originally, we created these labels manually in *tksurfer*. For the sake of completeness we leave these instructions in: Open the surface file as in step 3 and draw a wide region encompassing the occipital lobe.
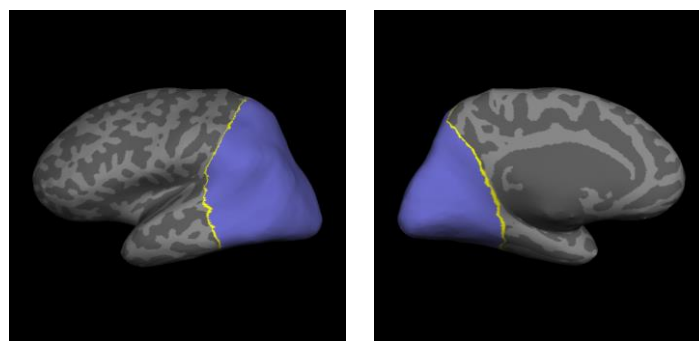   Save label with meaningful names, e.g. 'lh_occ' and 'rh_occ' in the *prf* directory



*Figure 2*

5

# CHAPTER 4: SAMSRF PROCESSING

**You can also read in GII (GIfTI) or MGH files directly, e.g. surface data projected using FreeSurfer with** mri_vol2surf**. If you are using this functionality you can select the MGH/GII files you created for each functional run using mri_vol2surf in** SurfaceProjection**.** If you do this, you probably already know how to use FreeSurfer functions. In fact, we generally use this procedure ourselves because it is much faster than our functions. **This is also the only way to use the standalone executable version of SamSrf.** Nevertheless, for completeness we continue below with the instructions using SamSrf's own surface projection functionality, **but we recommend using other approaches**.

**Important: for structural data with resolution better than 1mm (and some 7T data in general) the SamSrf functions currently do not work – so you <u>must</u> use FreeSurfer's functions or another way of surface projection instead!**

5. **Check structural registration**
   In MATLAB, navigate to where you want the pRF data to live (usually in *SUBJECT_NAME/prf*)
   Now launch SurfaceProjection
   Select the main folder for your FreeSurfer subject (labelled SUBJECT NAME in Figure 1)
   After a moment of loading the files a SPM check registration window will open
   A red line indicating the grey-white matter boundary will appear in both hemispheres

   *Note: since SamSrf 6.30 you can also use Matlab's native NII reading functionality by using the NatMatlabNii4SamSrf patch. This cannot display the surfaces interactively like SPM so you will have to make do with what it can do. Only use this if you know what you're doing.*

   Look at the match-up between the anatomy and the red overlay as shown in Figure 3. It should define a boundary between the white and grey matter that follows the sulcal pattern of the cortex:
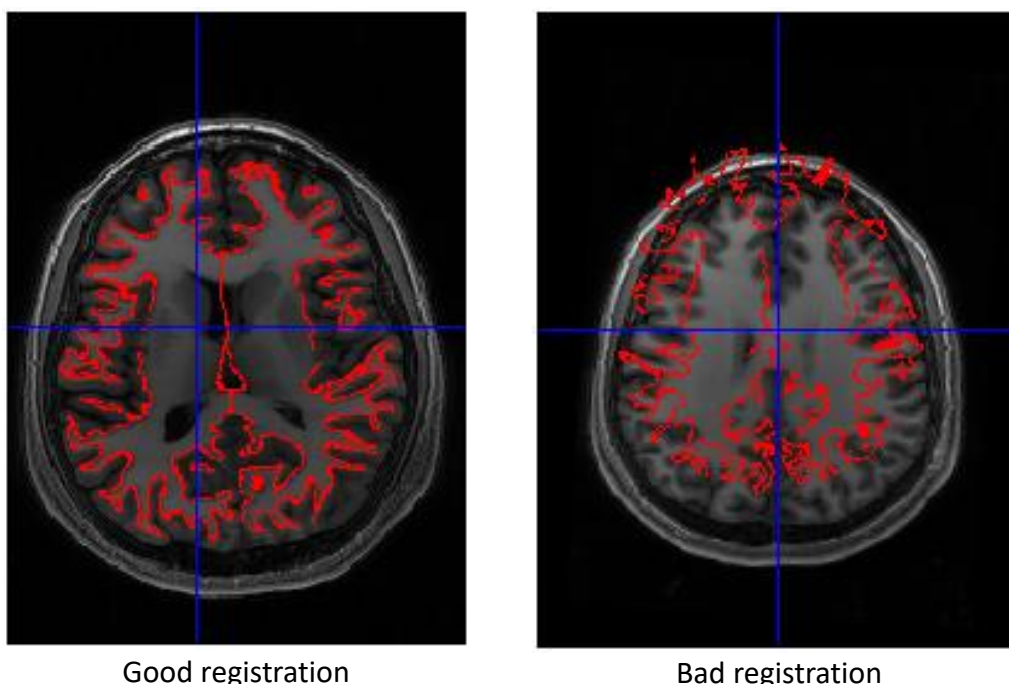


Good registration                    Bad registration

*Figure 3A*

There are various reasons why the link between FreeSurfer's coordinate system and SPM may fail. For example, if you fiddled with the affine transformation of the structural scan you used for FreeSurfer reconstruction (or if you resliced it or used it as source image in coregistration), then the alignment will be broken.

The example in Figure 3A should be pretty obvious but note that things can be a lot more subtle, as in Figure 3B. Note that the surface is only off by a few millimetres:
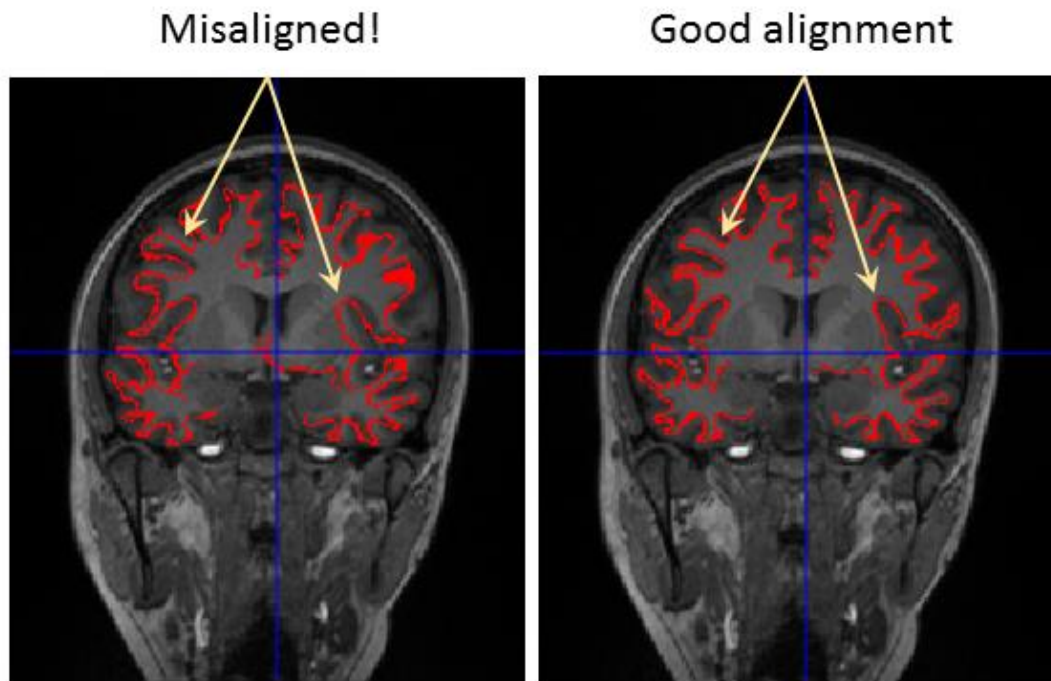


*Figure 3B*

This tool is also helpful for inspecting the surface reconstruction has worked properly (instead of step 3 above). If the red surface doesn't cover the whole cortex but parts of the cortex are outside of it, then you will have to fix the reconstruction. This may be due to too sharply defined structural images. Subtly smoothing (~1mm kernel) the structural in SPM can help in this situation.

Note that if you use mri_vol2surf then step 5 is only to ensure the surfaces are correct.

● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●

### What to do if registration is bad

In previous versions of SamSrf, there was a hack where you could provide a coregistration file that ensured the alignment between FreeSurfer's and SPM's space coordinates works. As of SamSrf verson 6.3 this has been removed because we believe to have fixed it. **Generally, for T1 data with better than 1mm resolution and/or ultra-high field data you should really use FreeSurfer's functions – and we recommend using those in any case!**

● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●

## 6. Create surface data files

SurfaceProjection will then automatically ask you for the other required files:

Select FreeSurfer subject folder (e.g. *freesurfer/subjects/SUBJECT/*)
    This will look for the .nii file from freesurfer that was used for reconstruction in
    *freesurfer/subjects/SUBJECT/mri/NAME_OF_T1.nii* or its orig subfolder.

Select functionals: choose all the <u>pre-processed</u> 4D NiFTI images, e.g. bars_1.nii, bars_2.nii
Note that if you use mri_vol2surf then select the MGH files for your functional runs instead.

You can also provide the *Cortical sampling steps* as a vector. The number is in proportions of cortical thickness. If you have more than one value here, then the activity of different steps will be averaged. If you don't enter anything, a default of 0.5 is used. Note that if you use mri_vol2surf then this step is not available as you have already done the cortical sampling.

Next there is a dialog asking you whether you want to *average*, *concatenate* or *separate* runs. Obviously, for averaging the stimulus protocol must then be identical in all those runs. For concatenation you may run into problems if there are too many runs to put into one matrix which will depend on the amount of RAM in your machine. Finally, you can keep individual runs separate (the projection will be done separately for each then and thus take much longer). For most purposes, averaging or concatenating are more useful but we left this function in for completeness' sake.

Next there is a dialog asking whether you want to use temporal normalisation. For functional data analysis, normalisation is necessary but you may also want to project all sorts of other data (e.g. structural data, GLM betas, t-statistics, etc) so then you will not want to use it.

Finally, there is also a menu allowing you to specify the projection method. This menu only appears if you defined more than one cortical sampling step. By default, this is the mean across cortical sampling steps but it can also be the median, minimum, maximum, or you can turn it off so all sampling steps are saved (this is useful for plotting the data along the vertex normal – **currently the main analysis functions in SamSrf however do not support such data)**.

This will take a few minutes (or longer if you have many cortical sampling steps). Once it's done, SamSrf will output all the files you gave it in .mat format with the prefix 'lh_' or 'rh_' appended to indicate which hemisphere it belongs to. The SurfaceProjection tool does this automatically for both hemispheres.

**Remember to do this for both the functional runs from pRF mapping <u>as well as</u> the HRF measurement run (if you have one).** The functions used to extract the HRF also work in surface space so they need this data. **You can also combine surface projected data of both hemispheres into a bilateral (bi_*.mat) file using the samsrf_bilat_srf.m function.**

*Note:* as of SamSrf 6.1 the surface projection stage also calculates the noise ceiling of your data – but only when using averaged runs. This is based on splitting the data into odd and even runs and so works best when having an even number of runs. The noise ceiling gives you an upper bound of what goodness-of-fit your pRF model can possibly achieve with

these data. It is stored in Srf.Noise_Ceiling in raw data files but after model fitting it is moved into Srf.Data.

7. **Fit HRF**
   **This is optional: you can simply use a canonical HRF or fit the HRF using your pRF data**

   In MATLAB, run this function to fit an HRF to the data from your HRF run. Assuming your HRF run is called lh_uabfHrf.mat and the TR of that run was 2.55 seconds you run:

   samsrf_extract_hrf('lh_uabfHrf', 2.55)

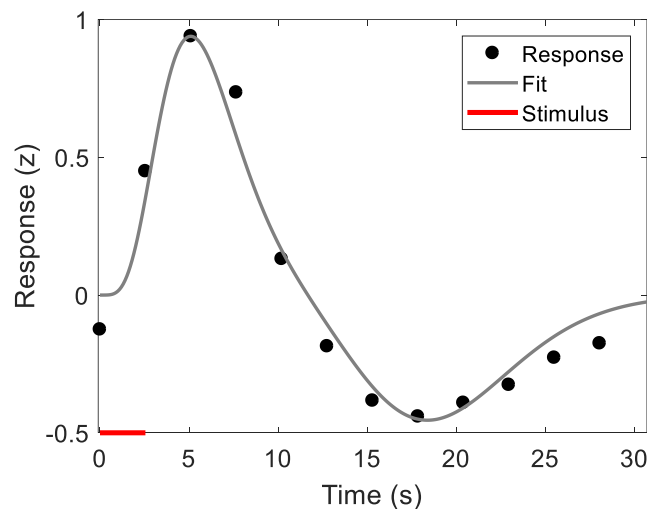   Check the curve-fit is sensible, for example (Figure 4):



*Figure 4*

*Fix bad HRF:*
This step is rarely ever necessary. If your HRF data contain outliers that skew the HRF fit, you can fix this using the samsrf_fix_broken_hrf function.

8. **Create apertures**
   Next you need the apertures to tell the model where the stimulus was in each volume.

   If you used a standard retinotopy design using some of the SamPenDu lab's scripts, you should be able to do this using the CreateApertures function and load the results files containing the apertures. If you used your own stimulus protocol, you will have to generate them in your own way.

   Critically, the apertures should be stored in a .mat file in a variable called *ApFrm*.
   This is a 3D matrix where columns and rows correspond to the x and y coordinates of pixels, and the third dimension is time. There is a frame for every volume. Usually these are binary images (1 for stimulus, 0 for background) but that is not essential.

   **Note: For forward-modelling, we require vectorised apertures** (using VectoriseApertures), **which will save them with the _vec suffix. As of SamSrf 10, the fitting algorithm will however do this automatically, so you need not worry about it anymore.**

9

9. **Inspect apertures**
   Once generated/saved, you may want to check that the apertures look as they should.
   Run the ViewApertures tool.
   Select the aperture file (should be named something like *aps_\*.mat*).
   Click through the display to make sure everything is correct.
   You can also click *Play Movie* to play it automatically.

# CHAPTER 5: pRF MODEL ANALYSIS

Unlike in previous versions, since SamSrf 6 there is no longer a GUI for running the analysis. More people scripted their analysis pipeline anyway and the GUI wasn't really that convenient. We now have wrapper scripts that define the parameters of the analysis. You can adapt them for your batch scripts to completely automate the process.

Example scripts are kept in the *SamSrf/Models* folder. There are scripts for the basic pRF models you might want to use, such as the standard 2D Gaussian, the Difference-of-Gaussians, various asymmetric pRFs, as well as 1D tuning curve models. You can also create your own – but this is way outside the scope of this cookbook. Here we will describe what you need to do to run a standard 2D Gaussian model defined in the script Standard_2d_Gaussian_pRF.m

The idea is that you take a copy of these example scripts, put that in a place related to your analysis, such as the parent folder where you keep all your subject data or a folder where you keep your other analysis scripts. You then modify the script to define the parameters of your analysis. It might also be advisable to give your script a unique name that reflects the project. For instance, in a study on pRFs in aging we might want to call this function Aging_Study_2d_pRF.m
In the script, you define a structure called *Model* which contains various parameters. If that seems scary, don't fret because most of these you won't need to worry about unless you define your own pRF model. You can simply skip forward to step **15-18** and then on to **20**.

**Importantly, you can use the ModelHelp tool to learn about all the possible model parameters in the various analyses and how to use them.**

10. **Define the pRF model**
    *Model.Prf_Function* is a MATLAB functional handle to the pRF model function. For the standard 2D Gaussian pRF model this is defined like this:

    > @(P,ApWidth)   prf_gaussian_rf(P(1), P(2), P(3), ApWidth)

    The handle has two inputs: *P*, which contains the actual model parameters to be fitted, and *ApWidth*, which is an internally required variable defining the stimulus coordinates.
    The maximum number of parameters is ten. You can see how other models are defined in the other example scripts in *SamSrf/Models*. **You don't need to worry about this part unless you want to define your own pRF models.**

11. **Model name**
    *Model.Name* defines the name of the pRF model, which is what the data file will be called. In our example script, this is 'pRF_Gaussian' so the final map file for the left hemisphere

10

will be called *lh_pRF_Gaussian.mat.* You can change this if you want.

### 12. Parameter names

*Model.Param_Names* is a cell array defining the names of the parameters. In a retinotopy model, the first two will be the horizontal and vertical coordinate of the pRF centre, so we call them *x0* and *y0*. For the pRF size (standard deviation of the Gaussian), we define the name as *Sigma*. The standard 2D Gaussian model only has three parameters so we can stop there. If we fit more parameters, they need to be named here. **The beta (amplitude) and intercept (baseline) are fit separately, after the pRF parameters have been estimated.**

### 13. Determine scaled parameters

Depending on the receptive field model you use, some or all the parameters are relative to the eccentricity of your mapping stimulus. Some parameters (like x and y coordinates or pRF size) are in this space and must therefore be scaled. The model fitting will reject parameter estimates twice the *Model.Scaling_Factor* (see below).

*Model.Scaled_Param* is a Boolean vector and it toggles whether a parameter is scaled or not. In the case of the standard 2D Gaussian model, all three parameters are scaled so it is [1 1 1]. But in the case of some other models, such as the oriented asymmetric 2D pRF, one of the parameters is the orientation of the pRF and this is obviously *not* scaled. **Critically, this vector must be of equal length as the cell array in step 12.**

### 14. Determine positive-only parameters

The model fitting function will look at the fitted (optimised) parameters in turn and reject those with absurd estimates. Any parameters that are scaled (see 13. *Model.Scaled_Param*) will automatically be rejected if the estimate is greater than twice the scaling factor. However, some parameters should also only be positive, e.g. the pRF size or tuning width cannot be negative.

*Model.Only_Positive* is a Boolean vector that toggles whether a parameter must be positive. In the case of the standard 2D Gaussian model, this is only the pRF size (Sigma).

### 15. Define scaling factor

Once we know which parameters will be scaled, we also need to define by how much it will be scaled. In two-dimensional retinotopic models, the scaling factor is the maximal eccentricity of the mapping stimulus (usually the radius of the mapped area). In one-dimensional tuning curve models (for example, tonotopy) this would be the half-width of the stimulus space because the middle is defined as 0.
**You must define this critical parameter because this will inevitably differ for your setup!**

### 16. Define the TR

You will also need to define the TR of your pulse sequence in seconds. This is critical because the convolution of the predicted time series with the HRF depends on the TR.
**This is also a critical parameter that depends on your experimental setup!**

### 17. Specify the HRF

In *Model.Hrf* you will also need to specify which HRF to use. We typically use the canonical HRF in which case we leave this empty []. If you estimated the HRF as described earlier in step 7, you can provide here the file name of the estimated HRF. Or you can provide the

HRF directly as a vector, where each component corresponds to one TR. Finally, if you don't want any HRF to be used (as you might in some situations) you must set this to 1. **Whatever you do, you need to specify this, even if it just [] to use the canonical.**

18. **Specify aperture file**

Finally, in *Model.Aperture_File* you will need to specify the file containing the apertures. Since this will typically depend on your experimental setup, **you must define this**. If you used the exact same stimulus design for each participant, you could just keep the aperture file in a common folder (with your model script) and provide the full path name here. **Apertures must be vectorised and contain both the ApFrm and ApXY variables!** You can use VectoriseApertures to convert movie apertures into vectorised ones.

19. **Define search space**

You can also define the parameters of the search space used for the coarse fitting (extensive grid search) stage. For each of the five parameters you define a vector of the points of the search grid for that parameter, so this vector determines both the range and the granularity of the search space. **Unlike in previous versions, since SamSrf 9 the search grid must be defined in stimulus space (e.g. degrees of visual angle)!** You could define it in unit space and multiple the scaled parameters by the *Model.Scaling_Factor* (see example scripts in SamSrf/Models)

For most people, there probably isn't much of a reason to change these parameters. If you make the granularity too fine the search space will become unwieldy and you may even run out of memory. The following parameters can be defined. These are just names for the model engine. The actual names in the final map you already defined above in step 12. The parameters can mean several different things, depending on the pRF model. In models with fewer than ten parameters you can just omit unneeded ones (they are internally set to 0).

| | |
|---|---|
| *Model.Param1:* | The horizontal coordinates of 2D pRF centres |
| | or the polar angle location of 2D pRF centres |
| | or the peak location of a 1D tuning curve mode |
| *Model.Param2:* | The vertical coordinates of 2D pRF centres |
| | or the eccentricity of 2D pRF centres |
| | or the tuning width of a 1D tuning curve model |
| *Model.Param3:* | pRF size, usually the standard deviation of a Gaussian pRF |
| *Model.Param4:* | Second pRF size in the DoG pRF or elliptical pRFs |
| *Model.Param5:* | The rare fifth parameter. In the DoG model the amplitude ratio, in oriented model this is the orientation, etc... |
| *Model.Param6-10:* | More complex models will have even more parameters although it becomes questionable how accurate these model fits will be. Ten parameters is probably more than the algorithm can realistically cope with... |

20. **Run model fit**

Now we are ready to run the model fitting. This is done using the generic samsrf_fit_prf function which takes the following inputs: the Model structure you specified above, a cell array with the SamSrf data files we created in Chapter 4, and (optionally) the region of interest label we created in step 4 (see the examples in SamSrf/Models). More complex pRF models or 1D tuning curves are run in a similar way. You can hopefully deduce a lot of that

12

from the example scripts and by referring to the ModelHelp tool.

**OPTIONAL MODEL PARAMETERS**
In addition to these standard parameters there are quite a few optional parameters, and analyses like reverse correlation of connective field modelling require different parameters. Please refer to the ModelHelp tool for more information about these.

21. **Post-processing pRF map**
     The pRF fitting function does not use any smoothed data unless you tell it to smooth the coarse fit or you presmooth your data before running the model fit.

     You may also wish to consider denoising the map. Beta parameters can be very diagnostic about rubbish data. If the Beta is 0 or negative, or if it is massively large, this is a sign of useless data. Using the samsrf_denoisemap function you can set the $R^2$ of such vertices to 0 and thus filter them out. Similar filtering may also be necessary based on implausible Sigma estimates (for example really tiny pRFs below the resolution of your stimulus or which are not biologically plausible).

     Further, for delineation or even just making figures you may wish to have smoother maps. You may also want to calculate field sign maps or the cortical magnification factor. To do all of that we can load the pRF map file and run the function samsrf_surfcalcs. In addition, for more complex models you may need to do other post-processing, for example calculating the FWHM of the DoG model or ordering orientation parameters. This can also be done at this point and some of the example scripts show how you might do such post-processing.


## RECAP: WHAT FILES DO I NEED TO RUN MODEL FITTING?

Here is you need to have in the *prf* folder (change *lh* to *rh* for right hemisphere):
     **Apertures file** with same length as the whole time series: (e.g. *aps_Bars.mat*)
     Surface data files for **pRF mapping runs** (e.g. *lh_Bars1.mat*, *lh_Bars2.mat*, etc)
     (Optional) HRF parameter file from an **HRF run** created in step 7 (e.g. *hrf_lh_Hrf.mat*)
     (Optional) **ROI label** file to restrict analysis created in step 4 (e.h. *lh_occ.label*)


## INTERMISSION: WHAT HAPPENS DURING MODEL FITTING?

First the surface data files are loaded into memory. If a region-of-interest was chosen the mask is loaded. Next, the search space is generated (see the *Introduction to pRF models* section). This used to take a long time on earlier machines but nowadays this is probably fine for most people. For more complex models (e.g. the Difference-of-Gaussians pRF) this takes longer because the search space is more complex. If you have the exact same order of conditions for each subject you can reuse the same search space (src_*.mat files) for each though and thus save time.

Next, the search space predictions are convolved with the haemodynamic response function. Then the coarse fitting starts. On modern computers and MATLAB versions this should take no more than a few minutes. Then the fine fitting commences and this can take from a half hour to a day or even a week, depending on what model you're fitting, which procedure you chose, and which

MATLAB version you are running. Finally, the maps are saved in a .mat file. **Note that the Model variable in which you defined the model parameters (such as the search space dimensions and the HRF) is also saved in that file! <u>Be aware of this if you are reloading the .mat file!</u>**

## CHAPTER 6: RENDERING MAPS

### 22. Displaying maps

To inspect maps and save figures you can use the DisplayMaps tool. This will ask you for a pRF map file in SamSrf .mat format and then load this up. By default this limits the view to an occipital ROI (you can change this option in *SamSrf_defaults.mat*).
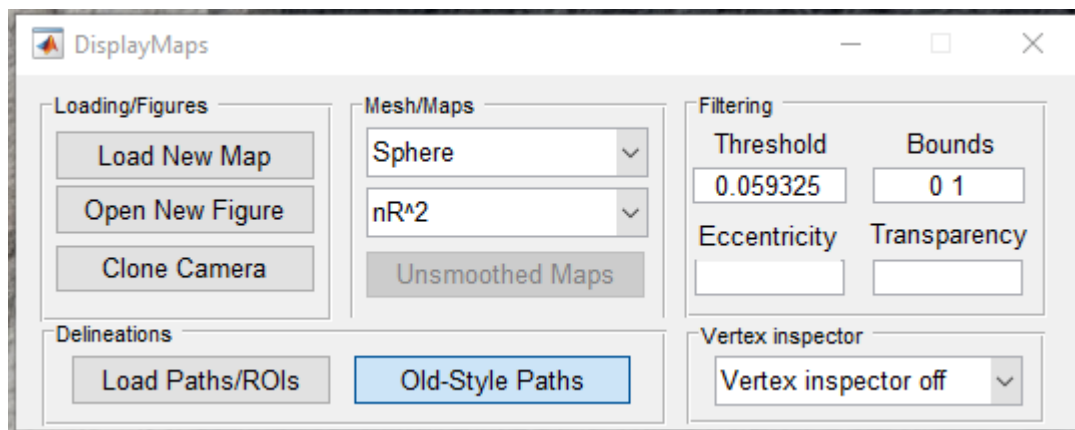


*Figure 5*

You can load any of the maps stored in your SamSrf file. The Polar Angle and Eccentricity maps need to be calculated as the file only stores Cartesian X and Y variables.
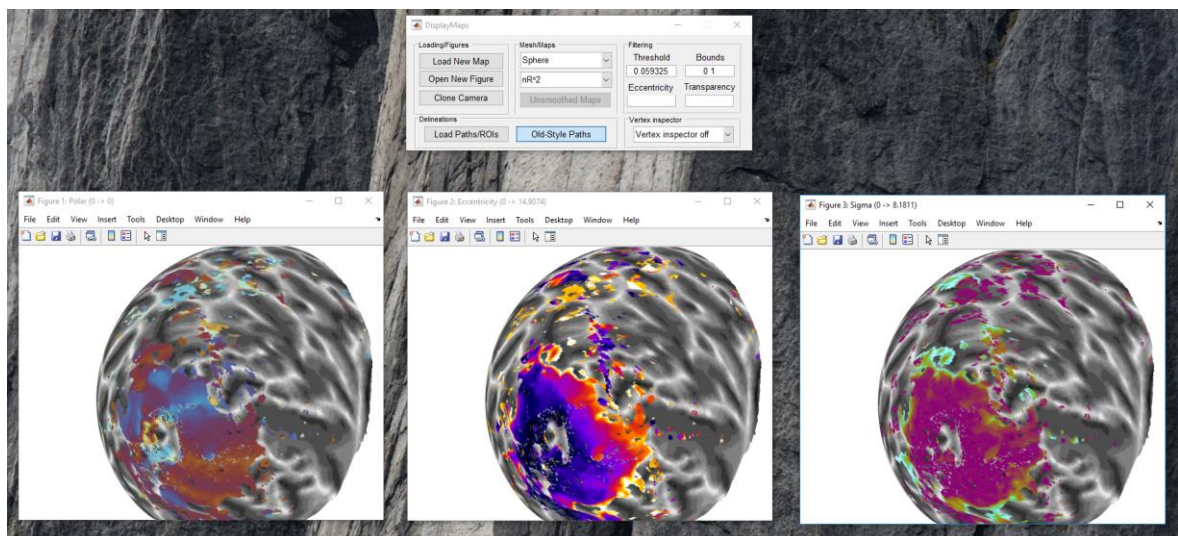


*Figure 6*

You can also load labels or paths previously saved. These can be in FreeSurfer *.label* format (ASCII files with a list of vertices) or paths in *.mat* files. You can also load paths stored in a delineation file saved by the DelineationTool

You can select any of the following cortical meshes to display: *Inflated, Pial, White*, and

14

*Sphere* (plus some other options derived from these). The spherical mesh is the default because it allows you to see a lot more cortex in the same view, but it may not be ideal for knowing what you're looking at (especially outside of occipital cortex).

The *Threshold* is automatically adjusted to give you a pre-defined p-value for your $R^2$. You can also manually adjust this to your liking. Similarly, you can adjust the *Eccentricity* range as a two-element vector (e.g. 1 8 to only show eccentricities between 1° and 8°). The *Bounds* determines how the maps are clipped. For $R^2$ maps this value determines at which level the hottest/colds colours are displayed. For eccentricity maps, this value determines beyond which eccentricity the maximum/minimum colour is used etc.

Finally, you can also set the *Transparency*. This determines the transparency of the map. By default this is 0.1, which means that that all model fits above the 10th percentile are fully opaque but below that the transparency is ramped down to the threshold. You can also set this to a negative value. In that case, the whole map has the same transparency, so for instance -0.5 means that the map is 50% transparent.

You can use the *Open New Figure* button to create a new main figure. This is the only figure that is updated by anything you do in the tool. So, you should first load the maps you want and then open a new figure for any further maps you want to look at simultaneously. If you have several figures open (e.g. the polar, eccentricity, sigma and $R^2$ maps) and want them all to show the same view, you click the *Clone Camera* button. This means all the other plots will adopt the same view as your main figure.

You can load a completely new map using the *Load New Map* button.

### 23. Renderer lighting

The pretty lighting can be nice for figures but does not seem to work on all computers and may result in a display bug. On some computers, the display is also very slow when the lighting is turned on. You can turn the lighting in the figures on and off with this command:

```
samsrf_lighting('off')
samsrf_lighting('on')
```

Lighting is turned off by default. For pretty figures, you may want to **first** adjust the camera angle and then turn on lighting.

### 24. How to delineate regions

A detailed description of how to delineate visual regions (including the higher extra-striate cortex) is the subject of a separate cookbook. In short, visual regions are bounded by the reversals in the polar and eccentricity maps. For the early regions (V1-V3) the polar map is fully sufficient. For higher regions (e.g. V4 and in particular the V3A/B complex and VO-1/2) it helps to also look at the eccentricity map because most of these regions have foveal representations separate from the early visual areas. A field sign map (showing whether or not a map is a mirror image of visual space) can also be useful, although the clarity of this map can vary a lot. Finally, the beta (amplitude) map can be very helpful in identifying bad data, in particular off-the-edge artifacts in parts of the visual field that were not stimulated. The AutoDelineation tool allows you to create some automatically drawn borders for regions

15

based on an atlas. These are far from perfect and must be corrected/completed, but they improve the efficiency and reproducibility of delineation.

To help you with the delineation Figure 7 shows the colour wheels. They can also be found in the *SamSrf/Colours* folder in your SamSrf installation or you can display them in MATLAB using the function

samsrf_colourcode



*Polar wheel*
*Left hemisphere*

*Polar wheel*
*Right hemisphere*

*Eccentricity wheel*

*Figure 7*
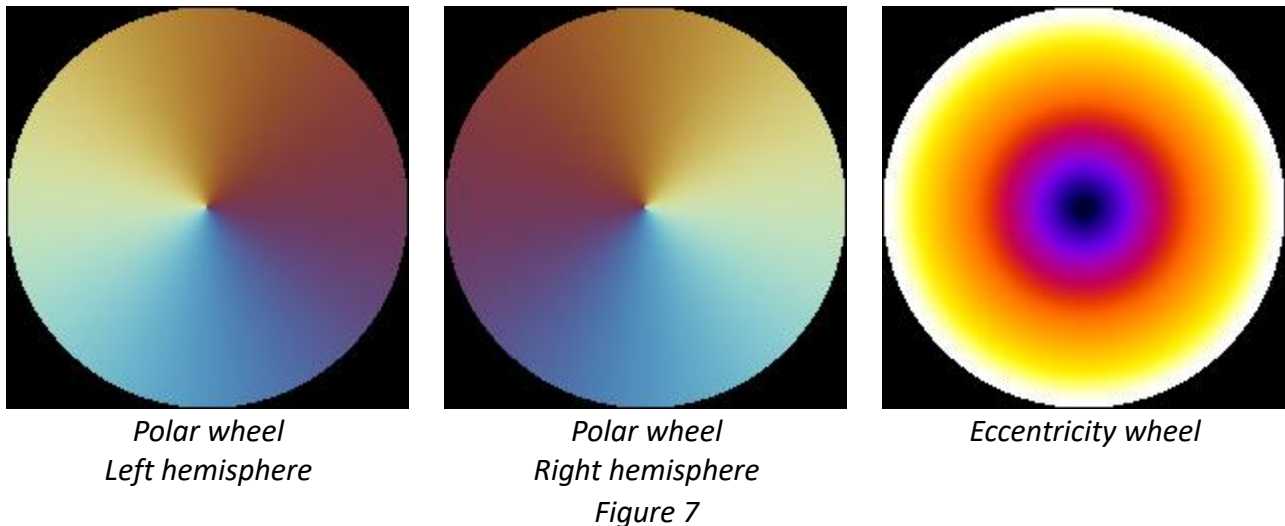
### 25. Exploring the results

To save the variables that generate the above graphs, you need to run these command line functions and repeat for each ROI:

```
load lh_pRF_Gaussian
Srf = samsrf_expand_srf(Srf)
```

This loads the pRF data into the workspace as a variable Srf. **The second line is critical to expand the data set and add in the anatomical meshes! Without that, none of the other functions may run properly,** although all native functions in theory should do that automatically (see also Appendix B).

To then plot data for a particular region of interest you can run:

```
samsrf_plot(Srf, 'Sigma', Srf, 'Eccentricity', [0 9], 'lh_V3', 0.5)
```

This produces a scatter plot for sigma in Srf against the eccentricity in Srf (Figure 8). The vector [0 9] determines the eccentricity range to be plotted. You can provide two different Srf variables here, e.g. when you want to compare different conditions based on a common eccentricity estimate. Here 'lh_V3' is the name of a ROI label for V3 in the left hemisphere. The 0.5 refers to the $_nR^2$ threshold to be used ($_nR^2$ is $R^2$ normalised by noise ceiling).
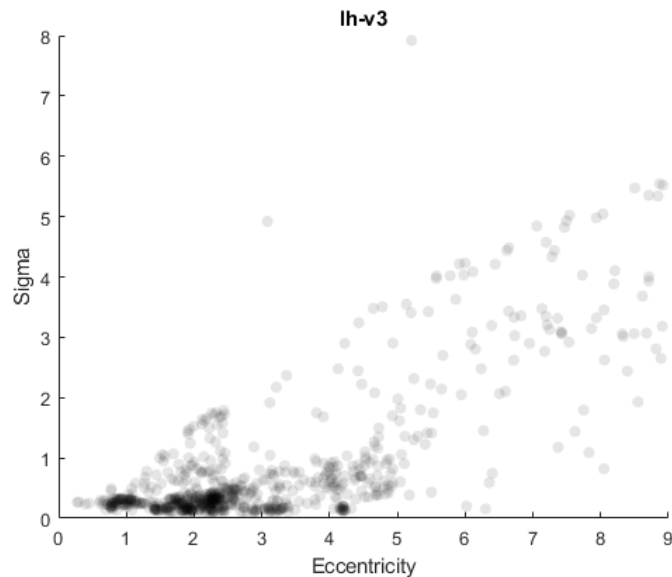
16

*Figure 8*

You can also use binned plots including polar plots when using polar angles for binning. **But please be aware of the dangers of binning analyses!**
(see https://www.biorxiv.org/content/10.1101/2020.12.15.422942v2)

There are many other functions available for analysing and quantifying the pRF results. For instance, you may want to plot the visual field coverage of V1 for pRFs with the $_nR^2$ threshold > 0.5, assuming a mapping stimulus with eccentricity of 9, and clipped to >10% of pRF coverage (Figure 9).
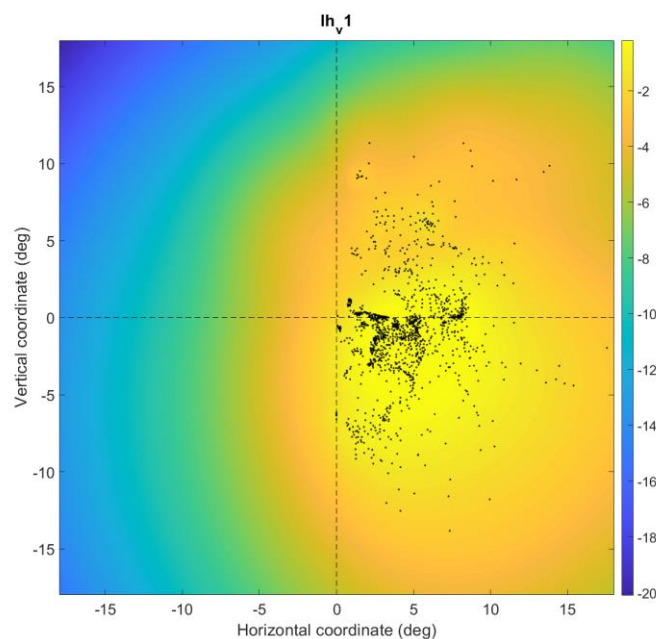
```
samsrf_vfcoverage(Srf, 9, 'lh_v1', 0.5, 0.1)
```



*Figure 9*

All functions contain `help` information that should explain how they work.

## APPENDIX A: FMRI PRE-PROCESSING

Sample pre-processing steps for fMRI data carried out in SPM8. **Note that these instructions are specific to data acquired in the FIL on a 3T TIM Trio scanner with a 32-channel coil.** If you didn't acquire your data there under these conditions, the procedure for you will most likely differ. A lot has changed since these notes were written, but they are still correct in principle.

**DICOM import**
      SPM > DICOM import
            Convert all your raw files to 3D NiFTI format, conserving the original filenames
      *prefix added: [f]*

**Bias correction**
      Perform bias correction on all the EPI images using spm_biascorrect
      *prefix added: [b]*

**Remove dummies**
      Place the first four volumes in a separate folder

**Create fieldmap vdm_* files**
      After DICOM import, fielmaps are divided into two folders; the first has two files (magnitude) and the second has a single file (phase).
      Batch script
            SPM > Tools > Fieldmap > Presubstracted phase and magnitude data
            Phase image: second field map folder (single file)
            Magnitude image: first file in first field map folder
            Blip direction: -1
            EPI-based field map?: no
            Jacobian modulation?: no
            Brain mask = no
            Match to the <u>first</u> volume in each EPI run

**Realign and unwarp**
      Batch script
            SPM > Spatial > Realign & Unwarp
`            1 subject = 1 module, each run in a separate session
            for each run, use the single session-matching vdm_ file
            Masking = yes
      *prefix added: [u]*

**Slice-timing correction**
      Batch script
            SPM > temporal > Slice timing
            On all bias-corrected, realigned and unwarped functionals
            Insert scan details (TR, TA, number of slices, slice order)
            Reference slice is middle one (15 of 30)
      *prefix added: [a]*

If you acquired a single structural, do a one-step coregistration. If you acquired a short and a long

structural, do a two-step coregistration. Note that it is important to select only 'estimate' and not 'estimate and reslice' in order to preserve the spatial origin from the fMRI images throughout.

**One-step coregistration**
> Batch script – Estimate
>> All functionals to high-resolution structural
>> Reference (target): high-resolution structural from Freesurfer folder
>> Source: $1^{st}$ aubf_* image of $1^{st}$ run
>> Other images: all other aubf_* volumes for all runs
>> One module per participant

**Two-step coregistration**
> Batch script
>> SPM>Spatial>Coreg>Estimate
>> All functionals to short structural with coil off
>> Reference (target): short structural
>> Source: $1^{st}$ aubf_* image of $1^{st}$ run
>> Other images: all other aubf_* volumes for all runs
>>
>> SPM>Spatial>Coreg>Estimate
>> All functionals to high-resolution structural with coil on
>> Reference (target): high-resolution structural from Freesurfer folder
>> Source: short structural
>> Other images: dependency on results from step above
>> One module per participant

**Check coregistration**
> SPM>CheckReg

**Optional: Smoothing (use only with the pre-smoothed receptive field model!)**
> All the coregistered, slice-timing corrected, unwarped, and bias-corrected volumes
>> Kernel size: 4mm
> *prefix added: [s]*

**4D merge**
> Combine smoothed images for <u>each run</u> using spm_file_merge
> Good idea to rename them here into the correct run, for example:
> - bars_1
> - bars_2
> - HRF

## APPENDIX B: STRUCTURE OF SAMSRF DATA FILES

### *Introduction*

You carefully designed your study, pre-processed your data and ran a model fit with SamSrf. Now, you can use the buttons in DisplayMaps to plot maps or used some functions to plot data. But what if you want to get your hands on the data themselves?

This guide covers the contents of the SamSrf results files and their potential uses. It assumes basic knowledge of the MATLAB environment and programming language. If you wish to learn MATLAB, you can head to MathWorks[1] or the excellent guide of MATLAB for Psychologists[2].

### *Navigation*

To access the SamSrf results file, open MATLAB and go into the subject folder where you ran the pRF fit. Here you will find a file called *lh_pRF_Gaussian.mat* or similar.
- 'lh' or 'rh' for left and right hemisphere, respectively, or 'bi' for bilateral files, or 'vol' for a volumetric data file (feature largely untested!)
- Or any arbitrary name for the actual data prior to any analysis (e.g. 'lh_Bars1')

Open the file by double clicking or typing:

*load('lh_pRF_Gaussian.mat')*

Once loaded, you will see 2 variables in workspace (some formats may contain more):
Model:          The pRF model specified as described in Chapter 5
Srf:            The main structure with all the data

By default, Srf variables are compressed and have their anatomical meshes removed. So before you can use them you must expand them (but all of the native SamSrf functions should in theory do that automatically). To do so run:

*[Srf, roi] = samsrf_expand_srf(Srf)*

Finally, before saving *.mat* files with a Srf you may wish to compress them again:

*Srf = samsrf_compress_srf(Srf, roi)*

This will remove the anatomical meshes from the *Srf*. Further, it also restricts the Srf to a region of interest defined by the vector *roi* (e.g. the occipital region). See s*amsrf_compress_srf* for details.

---

[1]    http://www.mathworks.co.uk/academia/student_center/tutorials/launchpad.html
[2]    http://antoniahamilton.com/matlab.html

### *Contents of 'Srf'*

The variable Srf is a structure containing several sub-variables. Note that this list only describes the most basic fields you might see when running a forward-model pRF analysis. Other analyses will contain additional fields:

**Srf.Version:** [scalar] Contains the version number used by the analysis. As of SamSrf 6, this is first added during surface projection but then updated again during pRF model fitting. Also, this has only been introduced in version 4.0 so if an analysed map file doesn't have this field it was analysed with an older version. This field is important to ensure data are comparable (e.g. Sigmas before version 4 were measured incorrectly). But ideally you shouldn't have to use any files created by versions prior to 7…

**Srf.Structural:** [string] Full path location of the structural image used as reference in alignment and for cortical reconstruction in FreeSurfer.

**Srf.Functional:** [string or cell array] If this is a pRF map, this contains a description of analysis carried out (e.g. which pRF model) and details of smoothing (if any) and name of ROI restricting the analysis (if any). If this is a data file, this contains the name of the functional NII file the data came from. If this is an average, it is a cell array of all the file names that were averaged.

**Srf.Hemisphere:** [string] Hemisphere contained in this file. This should match the filename. Please note that you can also create bilateral fields containing both hemispheres, in which case this will be 'bi'. You can also run volumetric analyses in which case this will be 'vol'.

**Srf.Cortex_Steps:** [scalar] Cortical sampling steps, as defined in the SamSrf GUI. This defines the voxel positions and number that each vertex is sampled from in volumetric space and is in units of cortical thickness. Therefore, the default 0.5 would mean that each vertex contains the data from that voxel that is located 50% along the normal vector from the vertex on the grey-white matter surface and the corresponding vertex on the pial surface. You could also choose more voxels though by setting this to a vector, e.g. [0:.2:1] would sample six voxels along this vector between grey-white and pial surface in steps of 20% of the cortical thickness. Taking more than one voxel probably only makes sense for relatively high-resolution data. Note that when you use FreeSurfer's surface projection functions this field is uninformative.

**Srf.Meshes:** [char] Contains a link to refer to the folder and file name where the anatomical mesh data are stored. This is not mandatory but is used by default when analysing data since SamSrf 6. It means that when you have more than one file with functional data projected to the surface, you only need to have one file for the anatomical meshes rather than duplicating this across files. Obviously, if you only have one file (e.g. for a public data repository) you might just share that with all the necessary data in the same file.

**Srf.Vertices:** [*m x 3* matrix] An index list of every vertex. Each row corresponds to the X, Y and Z coordinates of a single point in native space using FreeSurfer's internal coordinate system. The vertex coordinates correspond to the grey-white matter surface, so gyri and sulci are folded. May be stored in a separate file for anatomical meshes (see *Srf.Meshes*).

**Srf.Faces:** [*m x 3* matrix] Face values for every vertex. The surface meshes are constructed from thousands of tiny triangles, called *faces*. Each face is defined by three vertices and they are defined by the vertex indices in this list, one face per row. May be stored in a separate file for anatomical meshes (see *Srf.Meshes*).

**Srf.Normals:** [*m x 3* matrix] Each rows contains the vector pointing from a vertex on the grey-white matter surface to the corresponding vertex on the pial surface. This is used by surface projection to determine the voxels to sample from. May be stored in a separate file for anatomical meshes (see *Srf.Meshes*).

**Srf.Inflated/Sphere/Pial:** [*m x 3* matrix] Like Srf.Vertices but contains the coordinates model for the inflated, sphere, or pial surface mesh instead of the white-grey matter boundary. May be stored in a separate file for anatomical meshes (see *Srf.Meshes*).

**Srf.Curvature/Area/Thickness:** [*1 x m* vector] Contains the curvature, surface area, or thickness of the cortical surface model for each vertex. May be stored in a separate file for anatomical meshes (see *Srf.Meshes*).

**Srf.Data:** [*k x m* matrix] This contains all the resulting values of the analysis. If this field *Srf.Values* (see below) is present, the rows correspond to the k value descriptions in *Srf.Values* (e.g. $R^2$, $X_0$, $Y_0$, Sigma, etc..) and each column corresponds to a single vertex. So, if you are interested in, say, the $R^2$ values of every single vertex, you can index the first row and every column of *Srf.Data*. If *Srf.Values* is not present this is probably a file containing the data prior to analysis and each row of *Srf.Data* contains one volume. If you applied surface smoothing post-analysis, this variable contains the smoothed data. The un-smoothed data is stored in *Srf.Raw_Data*.

**Srf.Rule:** [char] Defines how the voxels in *Srf.Voxels* relate to the data for each vertex in *Srf.Data*. This only makes a difference if you have more than one cortical sampling step. The default is 'Mean' which means that each vertex is the arithmetic mean of the sampled voxels. Other options are 'Sum' for the sum, or 'Median' for the median (see *samsrf_vol2srf* for details). This is obviously only relevant for the actual surface projection. The analysed data files inherit this field as this tells you how surface projection was done.

**Srf.Noise_Ceiling:** [*1 x m* vector] Contains the noise ceiling if it has been calculated. This is only the case for raw data files. If the noise ceiling exists, the model fitting procedure will add this to Srf.Data and so there won't be a separate field in Srf for that in those files. At the moment, surface projection functions (*samsrf_vol2srf* and *samsrf_mgh2srf*) will automatically calculate the noise ceiling if there is an even number of runs that are being averaged. This is because the noise ceiling is based on split-half reliability data. When you concatenate runs or have more complicated designs, you may need to derive the noise ceiling yourself.

**Srf.Y:** [*n x m* matrix] The preprocessed time series for each vertex after z-normalizing each run and concatenating them (or in the case of CF analysis, after global mean correction). Each column is for a vertex, each row is for a volume (time point). This only exists for analysed data files.

**Srf.X:** [*n x m* matrix] The *neural* (rather than metabolic, so it is not convolved with the HRF!) time series for each vertex predicted by the pRF model. Therefore, $R^2$ can be calculated by comparing *Srf.X* to *Srf.Y*, after convolving Srf.X with the HRF. **Important: Just to complicate things, when running only a coarse fit, this field contains convolved time series!**

**Srf.Values:** [*k x 1* cell array] Description of the values stored in *Srf.Data*. Each row in this variable corresponds to the matching row in *Srf.Data*, so if the 4th row is 'Sigma', all the values along the 4th row in *Srf.Data* are Sigma values. Used for indexing. As of SamSrf 6, this is mandatory (and automatically added by *samsrf_expand_srf*, if it doesn't exist).

**Srf.Raw_Data:** [*p x m* matrix] If you applied surface smoothing, this variable will contain the non-smoothed data. If no smoothing is applied, this variable will not exist. Because values like the CMF are calculated after smoothing, they are not contained in this field.

**Srf.Roi:** [vector] A list of vertex indices that indicate whether any given vertex is inside the ROI used to limit the analysis. If used, the number of columns in *Srf.Data* should match the length of *Srf.Roi.* If this is the case, use *samsrf_expand_srf* to expand the data. This is essential before running any further analysis scripts on the data.