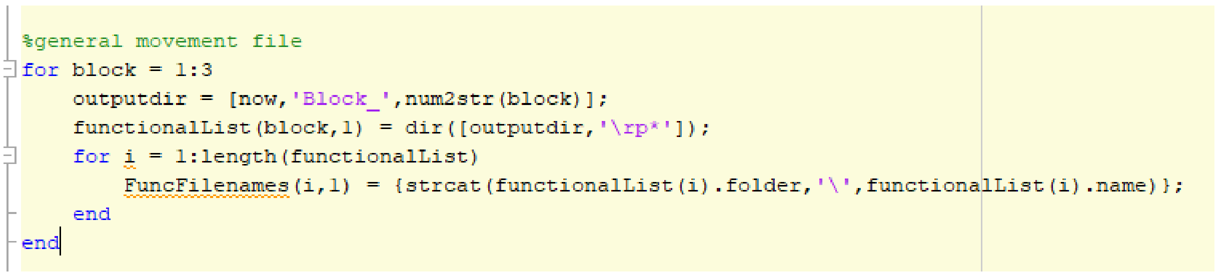
1st Level: Specification, Concatenation, Estimation and Contrasts

To concatenate runs in one GLM, you will need files specifying onsets and durations that seamlessly contain all blocks/runs in an ongoing manner. To produce those files extract the onsets for each event type you want to add into your GLM as a regressor, specify as well the durations (you can take the real duration of the event to get a better estimate or if the real duration is really short, you can model impulse durations of 0). An example MCF file (with: names, onsets and durations of all events/conditions of interest) can be found under **MCF_Example_file.m**. Make sure that the onsets of each event type are ongoing in time – this may require cropping out the time that spent during breaks between runs.

Before specifying the first level, make sure to write out a general movement file (.txt) for each participant over all runs. The *rp files can be found in the directory where the output of all the preprocessing steps is found.

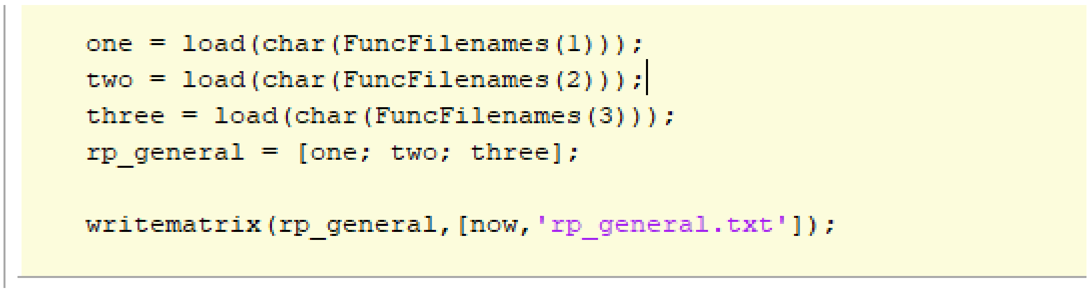
*If you did not run a realignment via movement regressors but used a field map for unwarping, run a realignment first to get the *rp files.*

1. **Get the *rp files from each run: write the paths in one struct**



```
%general movement file
for block = 1:3
    outputdir = [now,'Block_',num2str(block)];
    functionalList(block,1) = dir([outputdir,'\rp*']);
    for i = 1:length(functionalList)
        FuncFilenames(i,1) = {strcat(functionalList(i).folder,'\',functionalList(i).name)};
    end
end
```

2. **Concatenate the different *rp Files into one general *rp File and write this general *rp file to a .txt file**

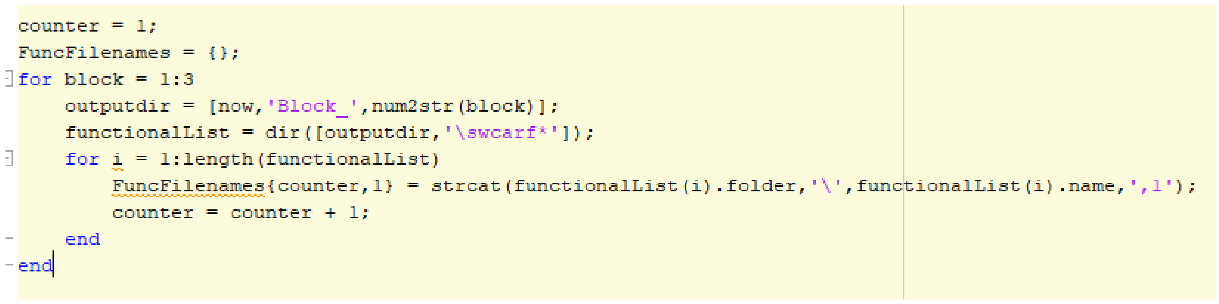


```
one = load(char(FuncFilenames(1)));
two = load(char(FuncFilenames(2)));
three = load(char(FuncFilenames(3)));
rp_general = [one; two; three];

writematrix(rp_general,[now,'rp_general.txt']);
```

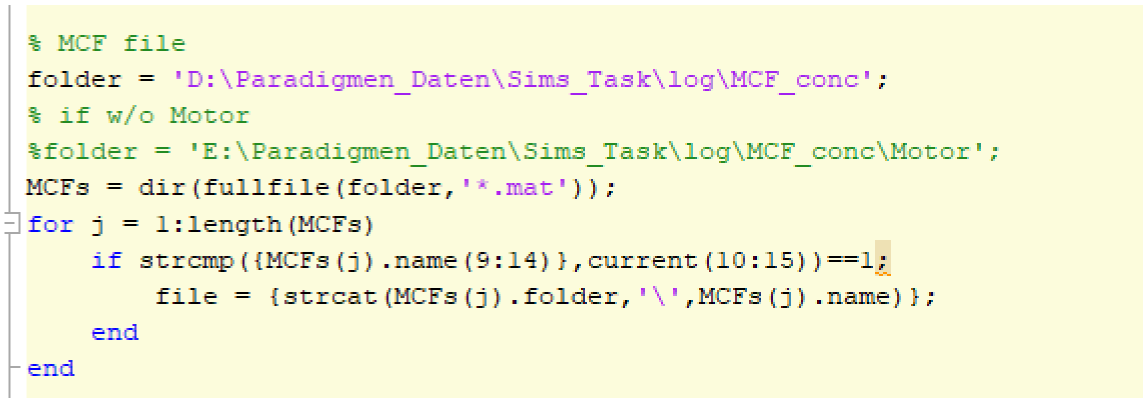
Now you can proceed specifying the first level:

1. **Get all your functional files from all your runs/blocks and write them in one struct to be assessed later:**



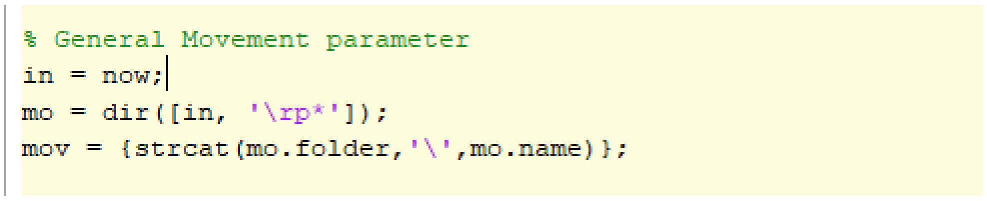
```
counter = 1;
FuncFilenames = {};
for block = 1:3
    outputdir = [now,'Block_',num2str(block)];
    functionalList = dir([outputdir,'\swcarf*']);
    for i = 1:length(functionalList)
        FuncFilenames{counter,1} = strcat(functionalList(i).folder,'\',functionalList(i).name,',1');
        counter = counter + 1;
    end
end
```

2. **Get your MCF files (where you specified the onsets and durations for each event of interest)**



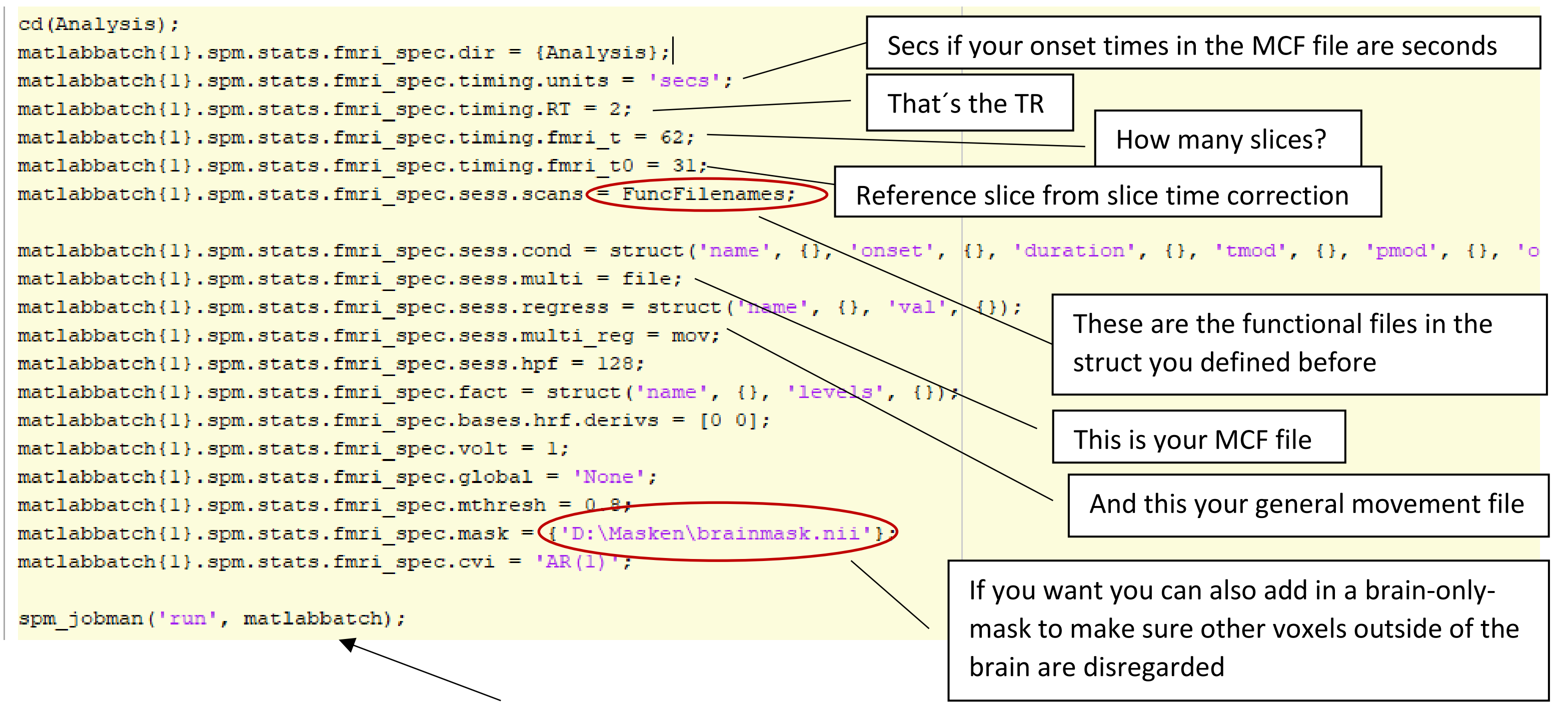
```
% MCF file
folder = 'D:\Paradigmen_Daten\Sims_Task\log\MCF_conc';
% if w/o Motor
%folder = 'E:\Paradigmen_Daten\Sims_Task\log\MCF_conc\Motor';
MCFs = dir(fullfile(folder,'*.mat'));
for j = 1:length(MCFs)
    if strcmp({MCFs(j).name(9:14)},current(10:15))==1;
        file = {strcat(MCFs(j).folder,'\',MCFs(j).name)};
    end
end
```

3. **Get your General Movement Parameter file**



```
% General Movement parameter
in = now;
mo = dir([in, '\rp*']);
mov = {strcat(mo.folder,'\',mo.name)};
```

4. And specify your first level



```
cd(Analysis);
matlabbatch{1}.spm.stats.fmri_spec.dir = {Analysis};
matlabbatch{1}.spm.stats.fmri_spec.timing.units = 'secs';
matlabbatch{1}.spm.stats.fmri_spec.timing.RT = 2;
matlabbatch{1}.spm.stats.fmri_spec.timing.fmri_t = 62;
matlabbatch{1}.spm.stats.fmri_spec.timing.fmri_t0 = 31;
matlabbatch{1}.spm.stats.fmri_spec.sess.scans = FuncFilenames;

matlabbatch{1}.spm.stats.fmri_spec.sess.cond = struct('name', {}, 'onset', {}, 'duration', {}, 'tmod', {}, 'pmod', {}, 'o
matlabbatch{1}.spm.stats.fmri_spec.sess.multi = file;
matlabbatch{1}.spm.stats.fmri_spec.sess.regress = struct('name', {}, 'val', {});
matlabbatch{1}.spm.stats.fmri_spec.sess.multi_reg = mov;
matlabbatch{1}.spm.stats.fmri_spec.sess.hpf = 128;
matlabbatch{1}.spm.stats.fmri_spec.fact = struct('name', {}, 'levels', {});
matlabbatch{1}.spm.stats.fmri_spec.bases.hrf.derivs = [0 0];
matlabbatch{1}.spm.stats.fmri_spec.volt = 1;
matlabbatch{1}.spm.stats.fmri_spec.global = 'None';
matlabbatch{1}.spm.stats.fmri_spec.mthresh = 0.8;
matlabbatch{1}.spm.stats.fmri_spec.mask = {'D:\Masken\brainmask.nii'};
matlabbatch{1}.spm.stats.fmri_spec.cvi = 'AR(1)';

spm_jobman('run', matlabbatch);
```

Secs if your onset times in the MCF file are seconds

That´s the TR

How many slices?

Reference slice from slice time correction

These are the functional files in the struct you defined before

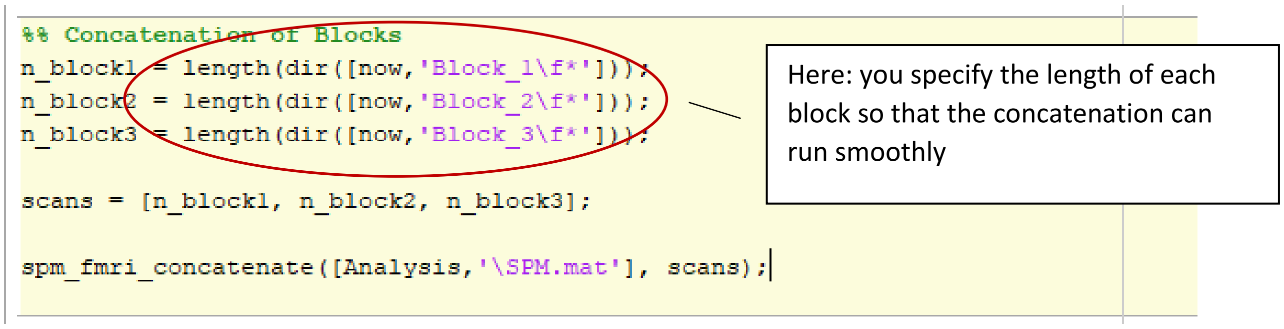
This is your MCF file

And this your general movement file

If you want you can also add in a brain-only-mask to make sure other voxels outside of the brain are disregarded

… and don´t forget to run your batch!

5. After Model specification you can now concatenate your blocks by using this script:



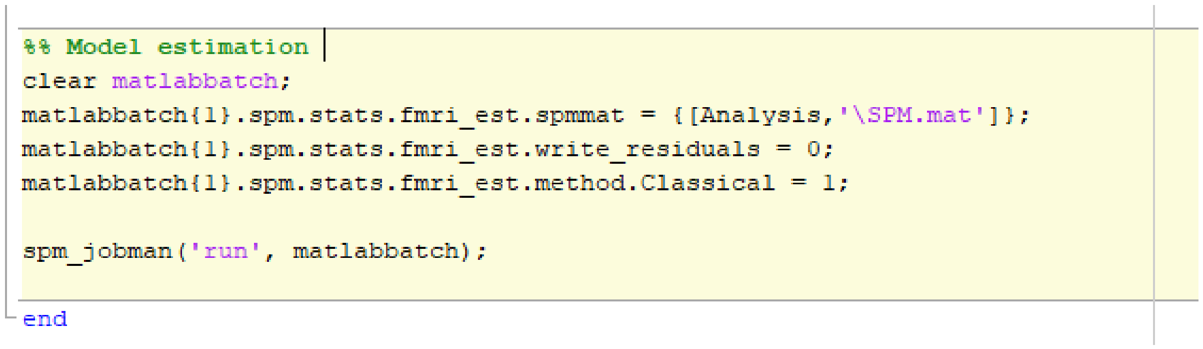
```
%% Concatenation of Blocks
n_block1 = length(dir([now,'Block_1\f*']));
n_block2 = length(dir([now,'Block_2\f*']));
n_block3 = length(dir([now,'Block_3\f*']));

scans = [n_block1, n_block2, n_block3];

spm_fmri_concatenate([Analysis,'\SPM.mat'], scans);
```

Here: you specify the length of each block so that the concatenation can run smoothly

6. And then you need to estimate your model

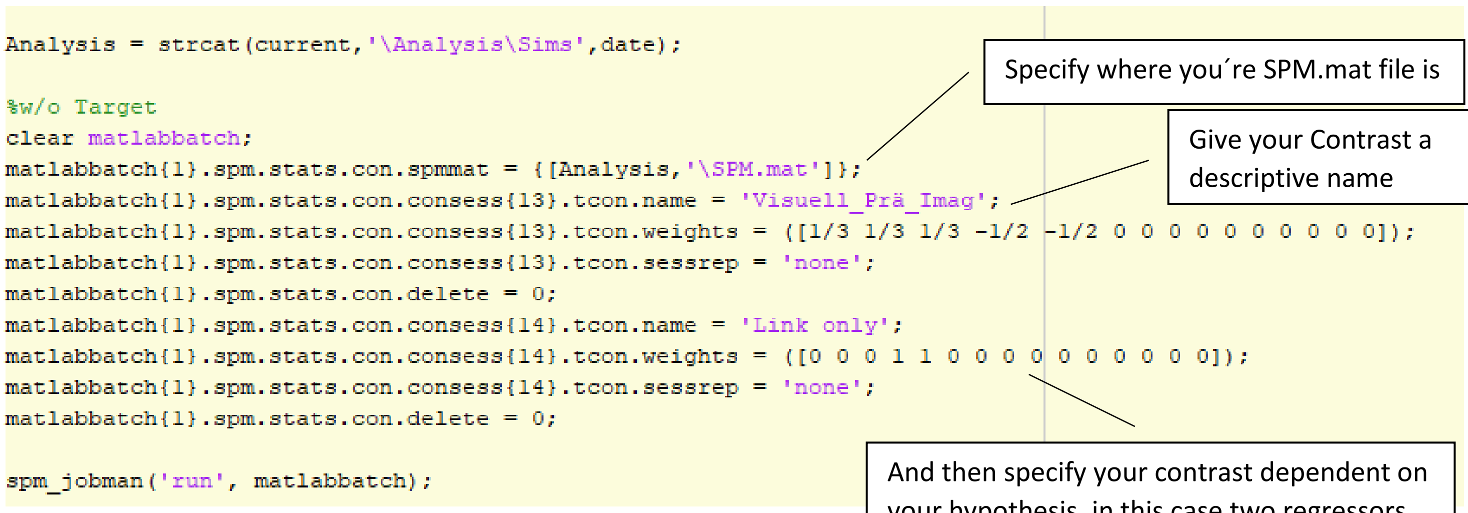


```
%% Model estimation
clear matlabbatch;
matlabbatch{1}.spm.stats.fmri_est.spmmat = {[Analysis,'\SPM.mat']};
matlabbatch{1}.spm.stats.fmri_est.write_residuals = 0;
matlabbatch{1}.spm.stats.fmri_est.method.Classical = 1;

spm_jobman('run', matlabbatch);

end
```

7. **Then you can proceed to calculate contrasts you´re interested in and want to take to the second level**



```
Analysis = strcat(current,'\Analysis\Sims',date);

%w/o Target
clear matlabbatch;
matlabbatch{1}.spm.stats.con.spmmat = {[Analysis,'\SPM.mat']};
matlabbatch{1}.spm.stats.con.consess{13}.tcon.name = 'Visuell_Prä_Imag';
matlabbatch{1}.spm.stats.con.consess{13}.tcon.weights = ([1/3 1/3 1/3 -1/2 -1/2 0 0 0 0 0 0 0 0 0 0]);
matlabbatch{1}.spm.stats.con.consess{13}.tcon.sessrep = 'none';
matlabbatch{1}.spm.stats.con.delete = 0;
matlabbatch{1}.spm.stats.con.consess{14}.tcon.name = 'Link only';
matlabbatch{1}.spm.stats.con.consess{14}.tcon.weights = ([0 0 0 1 1 0 0 0 0 0 0 0 0 0 0]);
matlabbatch{1}.spm.stats.con.consess{14}.tcon.sessrep = 'none';
matlabbatch{1}.spm.stats.con.delete = 0;

spm_jobman('run', matlabbatch);
```

Specify where you´re SPM.mat file is

Give your Contrast a descriptive name

And then specify your contrast dependent on your hypothesis, in this case two regressors were set 1 to test their effect against baseline

When you are done with calculating the contrasts, you can take them to the second level to run group analyses ☺