Fast and Efficient Mixed-effects Analysis (FEMA)

User Guide

Release v2.0

Apr 2022

Fast **E**fficient **M**ixed-effects **A**nalysis (FEMA)

FEMA is a tool to run mass univariate linear mixed effects analysis to estimate both fixed and random effects in a fast and efficient manner (<60s for 150,000 voxels). This can be done at the whole brain level using vertexwise and voxelwise data, and connectivity matrices from the ABCD study. FEMA also includes an external data option (.txt format) which can include columms of imaging or behavioral data.

FEMA: Fast and efficient mixed-effects algorithm for population-scale whole brain imaging data

Chun Chieh Fan, Clare E Palmer, John Iverson, Diliana Pecheva, Wesley K. Thompson, Donald Hagler, Terry L. Jernigan, Anders M. Dale

doi: https://doi.org/10.1101/2021.10.27.466202

Preprint on bioRxiv:

https://www.biorxiv.org/content/10.1101/2021.10.27.466202v1.full.pdf

Installation

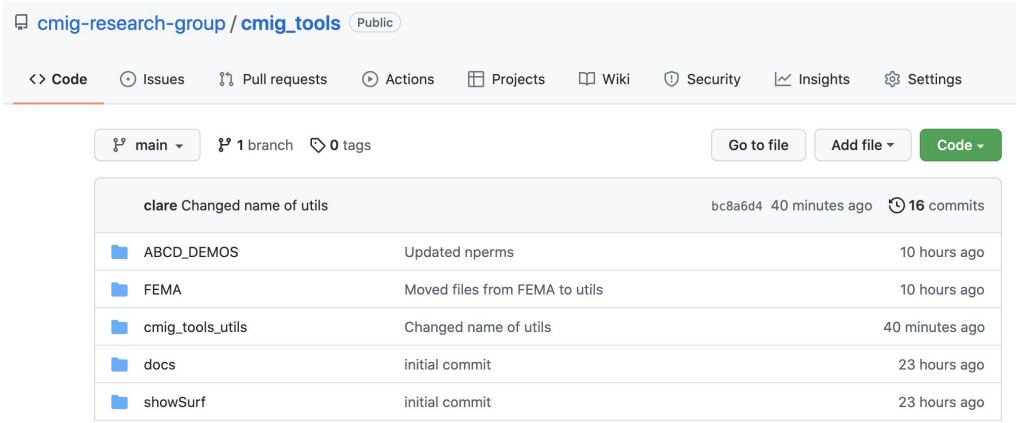
All of the code to run FEMA can be found in our cmig_tools package on GitHub:
https://github.com/cmig-research-group/cmig_tools

Dependencies
To run threshold free cluster enhancement you will also need to download PALM (Winkler et al, 2014) and include this in your MATLAB path:
https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/PALM/UserGuide#Downloading_and_installing_PALM

For ABCD Investigators only
We have included demos and ABCD specific processing functions to aid analysis of ABCD data. This requires a mirror of all concatenated imaging (NOT derived data) and non-imaging data files on your own server in the same directory structure. See ABCD internal confluence for more information.

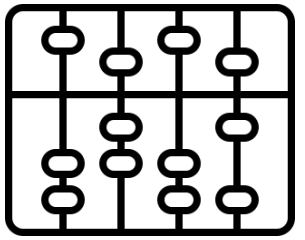
CMIG GitHub

To download the package from GitHub go to
https://github.com/cmig-research-group



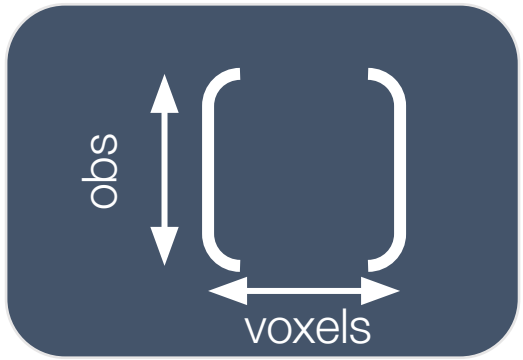
Or type enter the following command into your terminal
git clone https://github.com/cmig-research-group/cmig_tools.git
To up date your FEMA version, from the terminal cd into your FEMA directory and enter the following command:
git pull

Dependant variable format

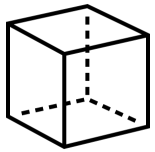


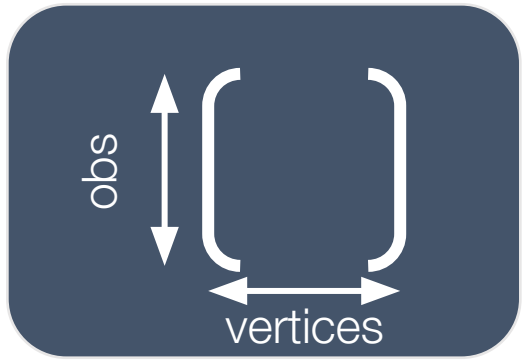
The dependent variable is parsed to FEMA as a 2D or 3D matrix in the following format:

- rows = observations
- columns = voxels/vertices/data points/nodes

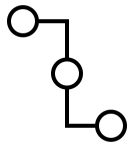


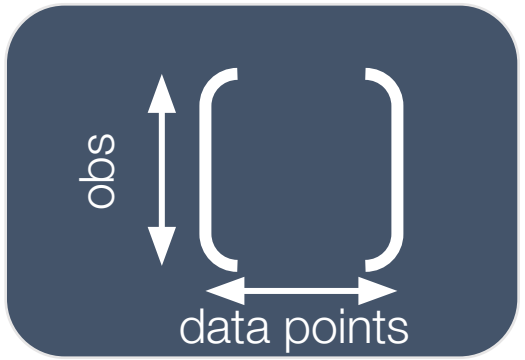
voxelwise



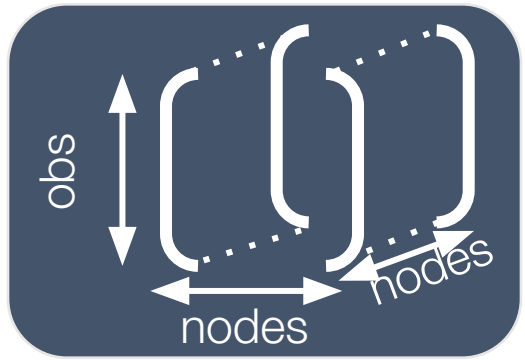


vertexwise

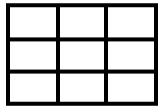




external

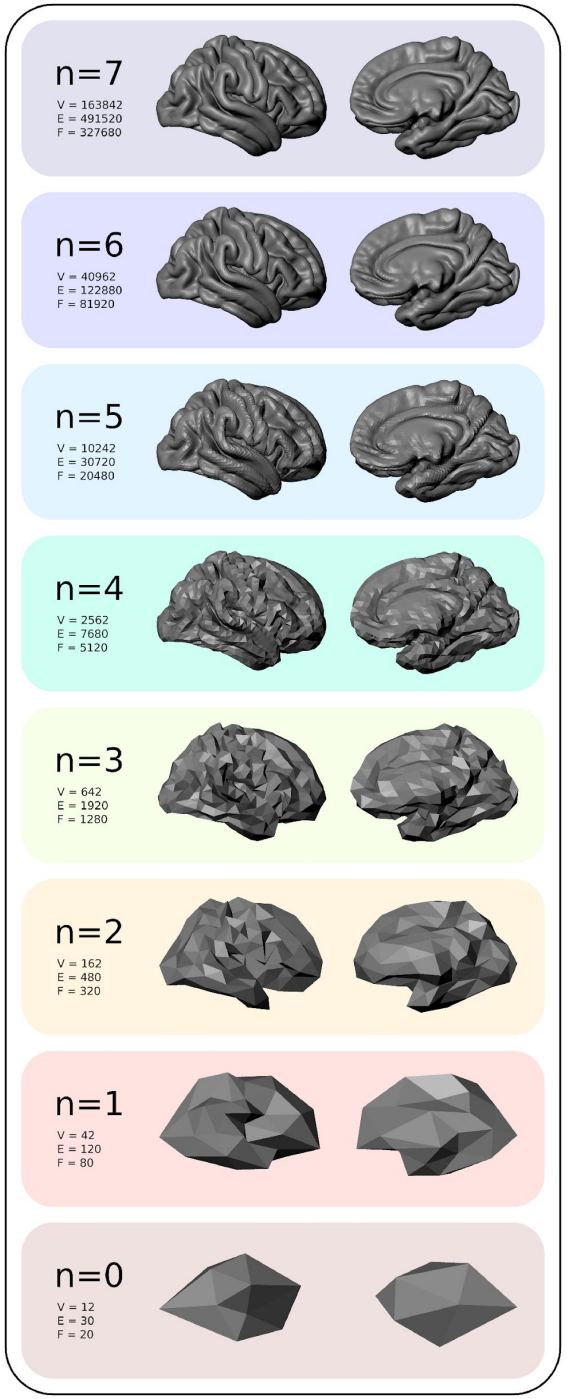


corrmat

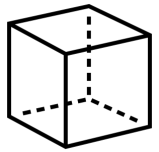


Vertexwise data

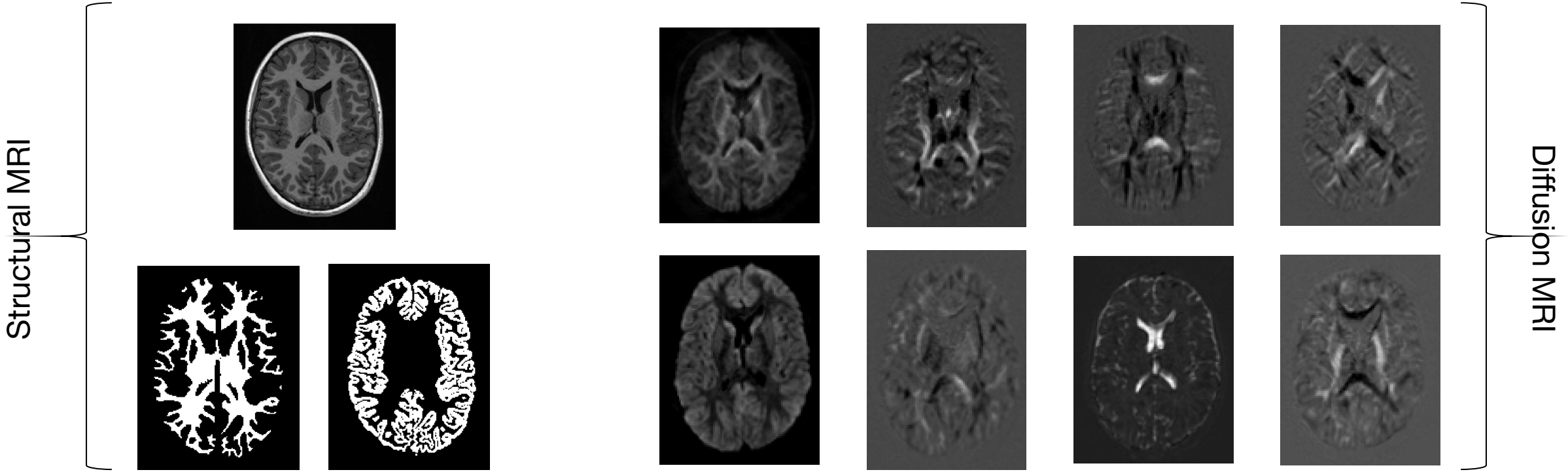
- Vertexwise data are saved at different resolutions.
- The *icosahedral number* defines the resolution, higher n $\Longrightarrow$ higher resolution



Voxelwise data (ABCD data)



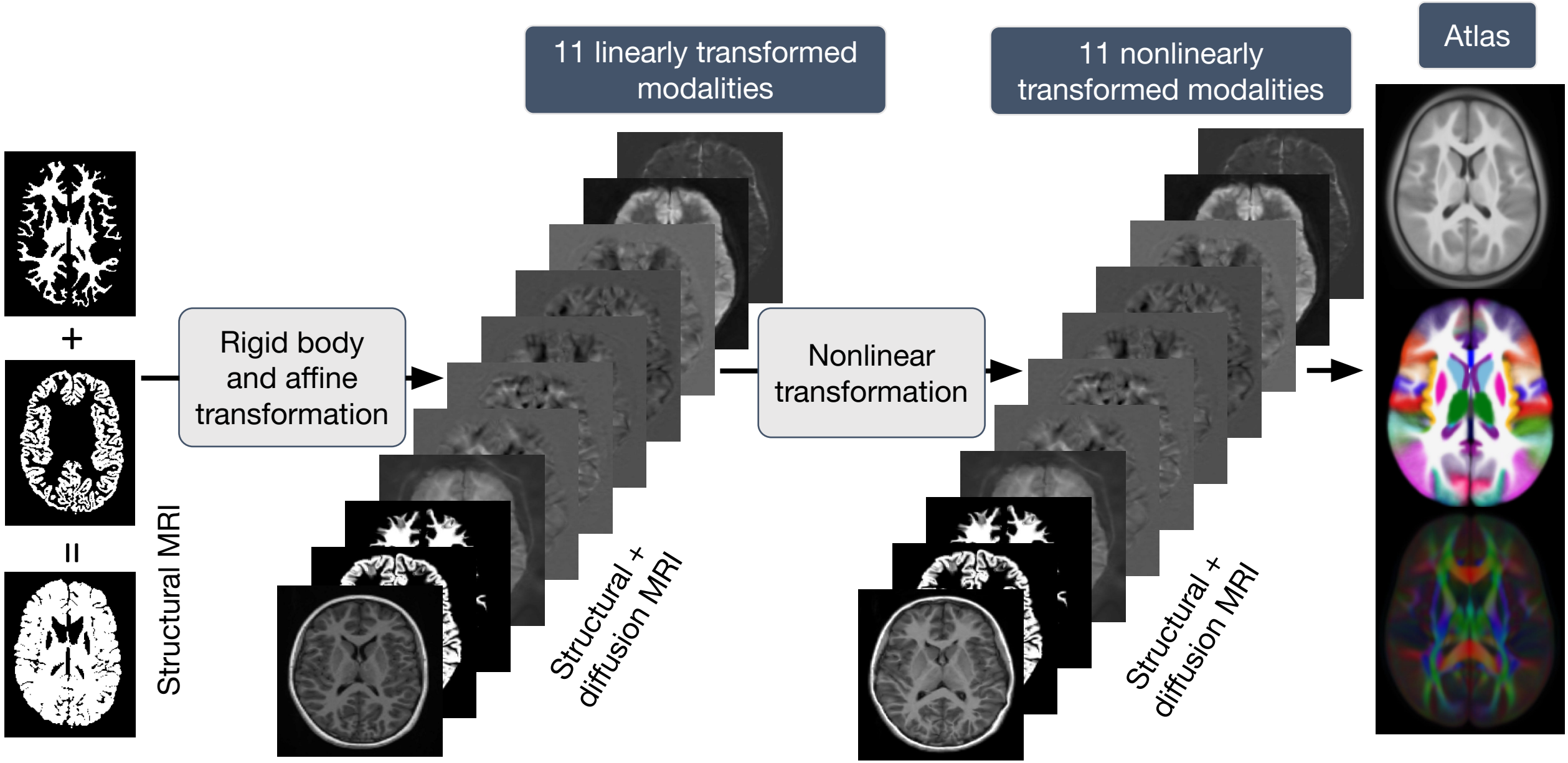
- To ensure anatomical correspondence across subjects, voxelwise data is aligned using Fast Efficient Multimodal Image Normalisation Tool (FEMINisT)



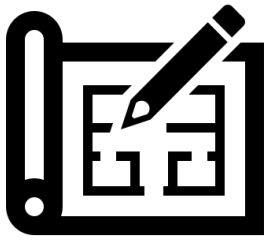
Fast Efficient Multimodal Image Normalisation Tool (FEMINisT)

- Subjects' images are transformed from the individual subject space to ABCD cohort-specific atlas space. The atlas was created by averaging all participants with imaging data that met QC in release 4.0.
- The initial rigid body and affine registration is performed on subjects' combined white matter and grey matter segmentation images.
- All 11 images are then transformed using the rigid body and affine transformations and all 11 images are used to determine the nonlinear deformations.
- This produced alignment across subject comparable to the best, most-widely used multimodal registration algorithm at a fraction of the computation time.

Fast Efficient Multimodal Image Normalisation Tool (FEMINisT)



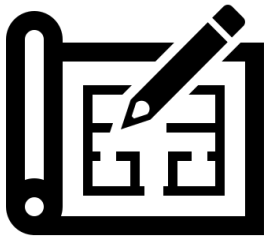
Creating your design matrix for FEMA



FEMA expects the following structure for the design matrix:

src_subject_id	eventname	rel_family_id	age	all predictors
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

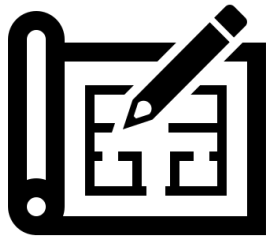
Creating your design matrix for FEMA



- The first 4 columns must be the src_subject_id, eventname, rel_family_id, age, as they appear in the ABCD tabulated data, and in that order.
- If any of those variables, age for example, are to be included in the design matrix as predictors they must be entered a second time, as part of the "all predictors".
- Within the FEMA package we provide two R scripts to help you make your design matrix:
  - `createMiniRDS_demo.R`
    - This script extracts variables of interest from the different instruments' `.txt` files on abcd-sync and saves them as an RDS file. You could skip this step if you wish to use your own script for this or use the complete RDS file from ABCD.
    - It has one dependency
      - `Loadtxt.R` to correctly load the tabulated .txt files from abcd-sync.
  - `makeDesign_demo.R`
    - Shows you how to run the script `makeDesign.R` which creates a design matrix in the correct FEMA format and checks for rank deficiency.
    - It has two dependencies.
      - `makeDEAPdemos.R` creates the demographic variables used in DEAP.
      - `makeDesign.R` which creates the final design matrix.

- Each script has its own in-line detailed annotations.

Creating your design matrix for FEMA



createMiniRDS_demo.R

Extract variables of interest from different instruments in the ABCD tabulated data

makeDesign_demo.R

Create design matrix in the correct FEMA format

makeDesign_demo.R

Set up for makeDesign

```r
#################################################################

# This script describes how to create design matrices in the format
# expected by the FEMA package. Users will need to save a local
# copy and modify paths and variables of interest appropriately.

#################################################################
# First need to make sure that R know where to find the makeDesign function
# which makes (obv) the matrices
source('/path/to/cmig_utils/r/makeDesign.R')

#################################################################
# The function makeDesign reads variables from an R data frame. This
# data frame can be loaded as the official ABCD RDS file or any other
# data frame, for example if you have saved individual instruments as a
# mini RDS as in the example script createMiniRDS.R

# Load the data
ndafile <- '/path/to/nda4.0_offrel.RDS'
nda <- readRDS(ndafile)

# It is STRONGLY recommended that you only include subjects which pass QC -
# for the imaging variable you are interested in testing.  For example this step
# filters using the imgincl_dmri_include variable from the abcd_imgincl01.txt instrument.
idx_dmri_inc <- which(nda$imgincl_dmri_include==1)
nda_dmri_inc <- nda[idx_dmri_inc,]

# Define the path to the directory where you would like to save out your design matrix
outpath <- '/path/to/your/fave/output/directory'

```

makeDesign_demo.R

Creating design matrix for **cross-sectional** analysis

```
31 ############################################################
32 # The following section describes how to set up a design matrix for CROSS-SECTIONAL
33 # analysis, ie to model the inter-subject variability associated with your predictors
34 # while accounting for repeated measures if you include multiple time points.
35
36 # Define the name of your design matrix file
37 fname <- 'designMat1.txt'
38 # and it's full path to your fave output directory
39 outfile <- paste0(outpath, '/', fname)
40
41 # makeDesign encodes continuous and categorical variables differently, therefore they are
42 # specified using different flags. Continuous variables are added using the "contvar" flag.
43 # Here we include age and the genetic PCs as continuous variables.
44 contvar <- c('interview_age', 'PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8','PC9', 'PC10')
45
46 # Categorical variables are dummy coded and added using the "catvar" flag. makeDesign automatically
47 # includes an intercept. For each categorical variable one category is defined as the reference category
48 # and that column is dropped. makeDesign also checks whether your matrix is rank deficient and if so
49 # will automatically drop further categories to avoid this. Here we include sex, scanner info and
50 # SES demographics as categorical.
51 catvar <- c('sex', 'high.educ', 'hisp', 'household.income', 'mri_info_deviceserialnumber', 'mri_info_softwareversion')
52
53 # The time points for which we wish to extract data are specified. You do not have to specify multiple
54 # time points, however if you do, specify them in chronoligical order ( this will be important for
55 # longitudinal modelling)
56 time <- c('baseline_year_1_arm_1', '2_year_follow_up_y_arm_1') # order matters! start with baseline!
57
58 # You can specify whether you wish to demean your continuous variables or not. the default is set to
59 # demean=TRUE. Other flags include "delta" for longitudinal modelling, "interact" to include interactions,
60 # "subjs" if you wish to filter by subject and "quadratic" if you wish to include a quadratic term
61 # (more details on these in the following sections). The defaults to these are set to null.
62
63 # Now run makeDesign!
64 makeDesign(nda_dmri_inc, outfile, time, contvar=contvar, catvar=catvar, delta=NULL, interact=NULL, subjs=NULL, demean=TRUE, quadratic=NULL)
65
```

makeDesign_demo.R

Creating design matrix for **longitudinal** analysis

```
66  ############################################################
67  # The following describes how to set up a design matrix for LONGITUDINAL analysis, ie to model the both
68  # the inter-subject and intra-subject variability associated with predictors of your choice across multiple
69  # time points. The following example creates a design matrix to model longitudinal changes associated with age.
70
71  # First specify a different name for your design matrix file
72  fname <- 'designMat2.txt'
73  outfile <- paste0(outpath, '/', fname)
74
75  # Seeing as we want to model variability associated with change in age, it is no longer included in
76  # the "contvar" flag. Age is now specified using the "delta" flag, which separates the age variable
77  # into its baseline value (saved as interview_age_base) and the change in age (saved as interview_age_delta)
78  delta <- c('interview_age');
79
80  # The other continuous variables are specified as before.
81  contvar <- c('PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8','PC9', 'PC10')
82
83  # Categorical variables are specified as before.
84  catvar <- c('sex', 'high.educ', 'hisp', 'household.income', 'mri_info_deviceserialnumber', 'mri_info_softwareversion')
85
86  # The time points for which we wish to extract data have to be specified in chronoligical order, currently
87  # it only works for two time points.
88  time <- c('baseline_year_1_arm_1', '2_year_follow_up_y_arm_1') # order matters! start with baseline!
89
90  # Now run makeDesign!
91  makeDesign(nda_dmri_inc, outfile, time, contvar=contvar, catvar=catvar, delta=delta, interact=NULL, subjs=NULL, demean=TRUE, quadratic=NULL)
92
```

makeDesign_demo.R

Creating design matrix with an **interaction**

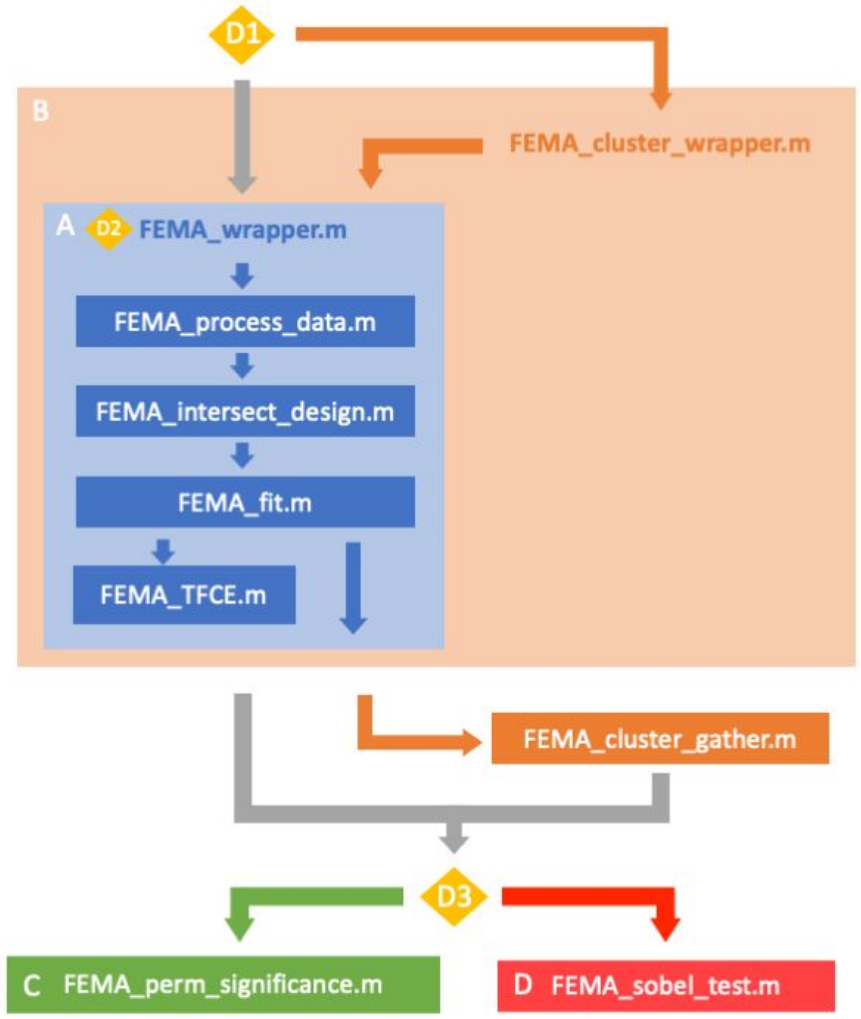
```
93  ##########################################################
94  # The following section describes how to create a design matrix which includes INTERACTIONS. This e
95  # example uses a cross-sectional model and includes an interaction between sex and age
96
97  # Again we specify a new file name
98  fname <- 'designMat3.txt'
99  outfile <- paste0(outpath, '/', fname)
100
101 # We specify the continuous variables. Seeing as we are not modeling longitudinal change with age,
102 # interview_age is back in with continuous variables. Both age and sex are included explicitly and
103 # seperately from the interaction terms that includes them both
104 contvar <- c('interview_age', 'PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8','PC9', 'PC10')
105
106 # The categorical variables are specified as before (sex is in there!)
107 catvar <- c('sex', 'high.educ', 'hisp', 'household.income', 'mri_info_deviceserialnumber', 'mri_info_softwareversion')
108
109 # The interaction term is specified using the convention "interview_age*sex"
110 interact <- c('interview_age*sex');
111
112 # The time points which we wish to extract are specified as before
113 time <- c('baseline_year_1_arm_1', '2_year_follow_up_y_arm_1') # order matters! start with baseline!
114
115 # Now run make Design!
116 makeDesign(nda_dmri_inc, outfile, time, contvar=contvar, catvar=catvar, delta=NULL, interact=interact, subjs=NULL, demean=TRUE, quadratic=NULL)
117
```

makeDesign_demo.R

Creating two design matrices for **mediation** analysis

```
# The following section describes how to create design matrices for a MEDIATION analysis.
# This code will produce two design matrices one that is termed '_reduced' and the other which
# which is termed '_full'. These are nested models. Here the mediator is body mass index (BMI).
# The FULL model will include all of the independent variables specified in contvar and catvar.
# The REDUCED model will have all of these excluding the mediator (BMI).
# The mediator is always placed as the penultimate column (before the intercept) in the deisgn matrix.
# Using this code ensures the two design matrices have the same number of subjects and are nested.

# Again we specify a new file name
fname <- 'designMat4.txt'
outfile <- paste0(outpath, '/', fname)

# We specify the continuous variables. Here we have additionally included
contvar <- c('interview_age', 'PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8','PC9', 'PC10','anthro_bmi_calc')

# The categorical variables are specified as before (sex is in there!)
catvar <- c('sex', 'high.educ', 'hisp', 'household.income', 'mri_info_deviceserialnumber', 'mri_info_softwareversion')

# The interaction term is specified using the convention "interview_age*sex"
interact <- NULL;

# The time points which we wish to extract are specified as before
time <- c('baseline_year_1_arm_1', '2_year_follow_up_y_arm_1') # order matters! start with baseline!

# Now run make Design!
makeDesign(nda_dmri_inc, outfile, time, contvar=contvar, catvar=catvar, delta=NULL, interact=interact, subjs=NULL, demean=TRUE, quadratic=NULL, mediator='anthro_bmi_calc')
```

FEMA Package Workflow



This flowchart will help you navigate through the FEMA functions and how to conduct mass univariate LME with imaging data using FEMA.

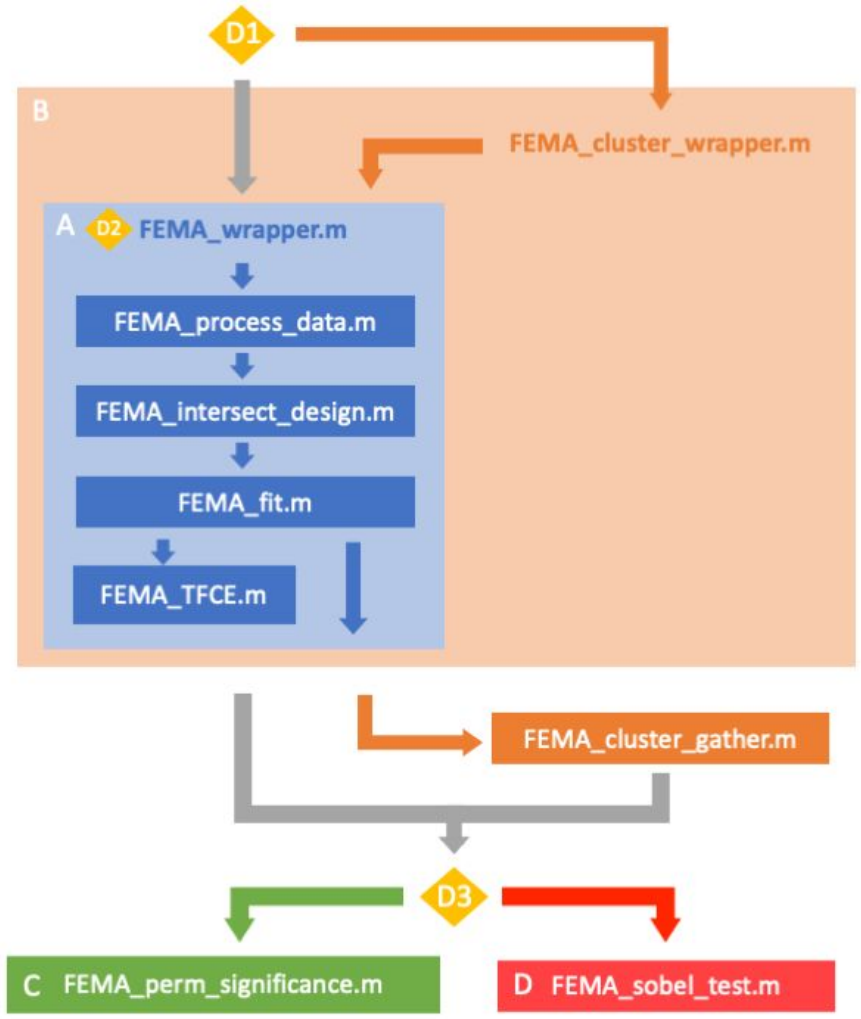
Yellow diamonds mark decision points. From top to bottom:

D1) Running on single computer vs cluster

D2) User specifies inputs to `FEMA_wrapper.m` depending on specific analysis

D3) The output from `FEMA_wrapper.m` can be used as input to different functions that run additional analyses beyond the mass univariate LME

FEMA Package Workflow



`FEMA_wrapper.m` is a wrapper function to simplify analysing data from the ABCD Study. FEMA_wrapper.m calls the three main functions required to run FEMA with ABCD data:

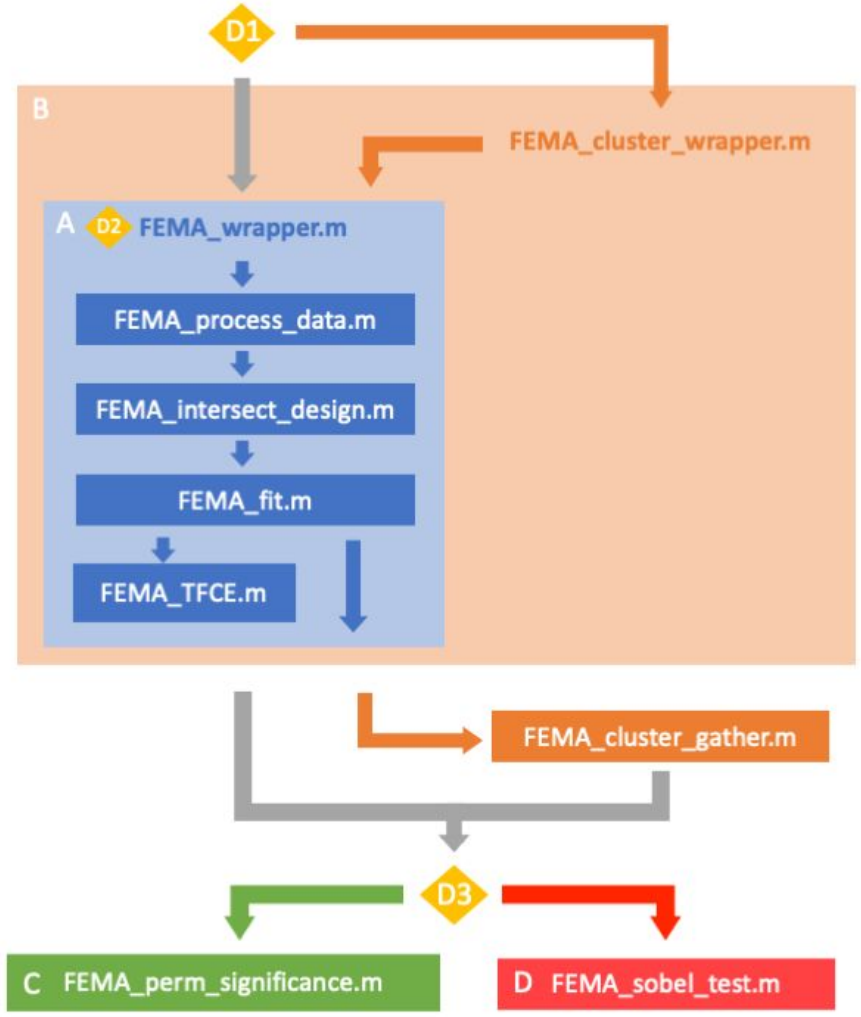
1. **`FEMA_process_data.m`** = loads imaging data (Y) and intersects with imaging info relevant for ABCD analysis

2. **`FEMA_intersect_design.m`** = loads design matrix (X) and intersects with imaging data (Y)

3. **`FEMA_fit.m`** = this is the core function that runs the linear mixed effects analysis. If you want to use FEMA to analyse other data not related to ABCD you can directly input data into FEMA_fit.m and write your own wrapper script to save the output.

`FEMA_wrapper.m` saves the output from `FEMA_fit.m` into a specified output directory.

FEMA Package Workflow



`FEMA_cluster_wrapper.m` is a helper function to create independent batch jobs to run `FEMA_wrapper.m` on a cluster. Each of these jobs will be an independent instantiation of `FEMA_wrapper.m`, which can then be run in parallel. This was designed to speed up running many permutations.

`FEMA_cluster_gather.m` is a function that concatenates several different FEMA outputs saved in different directories into a single output file.

When running several thousand permutations the output files can get very large >20GB. Therefore, the user can specify variables of interest to `FEMA_wrapper.m` using the input argument `colsinterest` and permuted statistics will only be saved for those variables.

Running LME using `FEMA_wrapper.m`

There are several different **input options** for `FEMA_wrapper.m` that can be used to specify the analysis.

To run a standard mass univariate LME analysis these are the inputs for `FEMA_wrapper.m`:

- **fstem_imaging** = name of imaging variable to analyse (e.g. thickness_ic5_sm256,FA, MD)
- **fname_design** = cell array with path to file with design matrix saved (if want to batch can add multiple filepaths as separate rows within fname_design as a cell array)
- **dirname_out** = cell array with path to output directory (if batching design matrices must include separate output directory as rows within dirname_out as a cell array)
- **dirname_tabulated** = path to tabulated data directory with NDA downloaded txt files
- **dirname_imaging** = path to imaging data directory
- **datatype** = which type of data to analyse (e.g. vertex, voxel, corrmat, external)
- **contrasts** = vector of contrasts for specified analysis (e.g. [1 -1 0 … 0] will calculate different between columns 1 and 2 in the design matrix)
- **ranknorm** = whether to inverse rank normalize the dependent variable
- **ico** = icosahedral number for vertexwise analysis (default, 5)

`FEMA_wrapper.m`will save an output .mat file which will include mass univariate z statistics, betas, beta SE and -log10 p values. The output will also include mass univariate random effect estimates:

- **sig2tvec** = total residual variance
- **sig2mat** = the proportion of total residual variance explained by each random effect. E.g. the default random effects are family (F), subject (S; for repeated measures) and E (unexplained residual). Each of these values will be represented as a row in this matrix (in the order specified in `FEMA_wrapper.m`). They will sum to 1.

For a demo see `~/github/FEMA/DEMOS/` **`FEMA_wrapper_demo.m`**

Running LME using `FEMA_wrapper.m`

Saving outputs

FEMA_wrapper.m saves the output from each model in it's own directory specified by dirname_out. Depending on the analysis run a sub-directory will be created to avoid over-writing results from the same design matrix. A a data structure, `save_params`, is saved with each output. This includes all the variables needed to save FEMA output files:

- **outdir** = output directory to save significance results
- **fstem_imaging** = name of vertex/voxel-mapped phenotype (DV) (e.g., 'thickness-sm16', 'FA')
- **datatype** = 'voxel','vertex','external', 'corrmat'
- **synth** = whether synthesized data were used

This data structure can be input to subsequent functions in the FEMA pipeline in order to save outputs in the same directories with the same file naming convention. If `save_params` is not passed to other functions, the output from those functions will not be saved.

FEMA functions can also be used interactively and results output into the MATLAB workspace by specifying the correct output arguments for each function.

Running LME using `FEMA_wrapper.m`

Resampling in FEMA

FEMA uses a residual, wild bootstrap resampling procedure that can be used to generate a null distribution in order to calculate standard errors and confidence intervals that are not dependent on the shape of the empirical distribution (Wu, 1986). This is useful if the dependent variable is not normally distributed or if the model is heteroskedastic. The code generates a synthetic y (dependent variable) by multiplying residuals by a random 'wild' weight and computing y with this equation: $y_i^* = \hat{y}_i + \hat{\varepsilon}_i v_i$

The wild bootstrap can generate an empirical null distribution or an empirical non-null distribution. The choice of this will depend on the analysis being performed:

- For Threshold Free Cluster Enhancement (TFCE) use null wildbootstrap ('wildbootstrap').
- For mediation analysis using Sobel's test use non-null wildbootstrap ('wildbootstrap-nn').

To run wild bootstrap use the following inputs to `FEMA_wrapper.m`:

- **nperms** = number of permutations
- **PermType** = 'wildbootstrap' or 'wildboostrap-nn'
- **colsinterest** = vector of column indices for variables of interest in design matrix (all permuted output will be truncated to only save permutations for these variables in order to reduce the output file size)

Resampling will also produce an output variable, `colnames_interest,` which will list the independent variable names for which the permuted statistics have been saved. Permuted statistics will only be saved for the independent variables (columns in design matrix) specified by `colsinterest.colnames_interest` can then be passed to subsequent functions so these variable labels are saved with each output.

For a demo see `~/github/FEMA/DEMOS/` **`FEMA_resampling_demo.m`**

Running LME using `FEMA_wrapper.m`

Estimating p-values from resampled null distribution

Permutation tests use the resampled empirical null distribution to estimate the likelihood of observing a given unpermuted statistic irrespective of the shape of the distribution and avoiding any assumptions of normality (Winkler et al., 2014). The rank of the observed statistic within the empirical null distribution is used to estimate the p-value. However, the precision of this p-value is limited by the number of permutations computed (min(p) = 1/nperms). In some cases, the unpermuted statistic is much larger than the maximum permuted statistic. In order, to determine a more precise p-value without having to run a very high number of permutations (which can be computationally expensive), a known statistical distribution can be fit to the tail of cumulative distribution function (CDF) of the permuted statistics in order to predict the expected p-value for much larger test statistics in the extremes of the null distribution. Here we have implemented code to compute both rank p-values of the observed statistics based on empirical null distribution and extrapolated p-values based on fitting a known statistical distribution to the CDF of the permuted statistics. **Extrapolated p-values are dependent on the fit of the known distribution to the CDF of the permuted statistics, therefore plots must be checked to ensure a good fit before extrapolated p-values are reported. The user must specify if they want to estimate extrapolated p-values; this is not default.**

To estimate p-values from permuted statistics use the following inputs to `FEMA_perm_significance.m`:

- **statmat_perm** = matrix of permuted statistics (IVs x voxels x perms) → statmat_perm(:,:,1) are the unpermuted statistics
- **stattype** = type of permuted statistics, e.g. 'z', 'tfce'
- **colnames_interest** = names of IVs used to produce statistics in statmat_perm
- *Optional input arguments:*
  - **save_params =** naming parameters needed to save outputs
  - **mask =** imaging mask (important to include for vertexwise data)
  - **extrapolate** = whether to estimate extrapolated p-values (default, 0)
  - **makeplots** = whether to make plots of the fit of the distribution (default, 0)

For a demo see `~/github/FEMA/DEMOS/` **`FEMA_resampling_demo.m`**

Running LME using `FEMA_wrapper.m`

Threshold free cluster enhancement (TFCE)

TFCE is an alternative to supra-threshold cluster statistics to determine the significance of vertex/voxelwise effects taking into account the correlation among spatially contiguous variables (Smith & Nichols, 2009). The benefit of TFCE is that it does not require an initial threshold to be specified manually by the user. TFCE alters the raw test statistic to account for the amount of cluster-like spatial support there is at each voxel. Each voxels new score is a weighted sum of all the local clustered signal without the need for any thresholding. P-values can then be estimated from the TFCE voxel or vertexwise statistics via permutation testing.

The FEMA code uses helper functions from the FSL PALM toolbox (Winkler et al., 2014) (https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/PALM). This must be added to your MATLAB path to run TFCE with FEMA.

To run TFCE use the following inputs to `FEMA_wrapper.m`:

- **nperms** = number of permutations
- **tfce** = logical indicator, if this is set to 1, then TFCE will be run
- **colsinterest** = vector of column indices for variables of interest in design matrix (all permuted output will be truncated to only save permutations for these variables in order to reduce the output file size)

For a demo see `~/github/FEMA/DEMOS/` **`FEMA_resampling_demo.m`**

To increase efficiency, for each permutation, the TFCE statistics are calculated within `FEMA_wrapper.m`. The p-values can then be estimated using the output from `FEMA_wrapper.m` using the function `FEMA_perm_significance`. This is because the p-values need to be estimated from the full distribution of permuted statistics and allows `FEMA_wrapper` to be run in parallel on a cluster and the outputs concatenated into a single file before estimating p-values.

Running LME using `FEMA_wrapper.m`

Mediation analysis

Sobel's test is designed to demonstrate whether the relationship between two variables of interest changes when another variable (the mediator) is added into the model (Sobel, 1989). If this change is large enough, the added variable is thought to mediate the relationship between the two other variables of interest. Here we have implemented Sobel's test at the vertex and voxel level.

FEMA expects two design matrices that are nested models with and without the mediator of interest. FEMA will then test whether the addition of the mediator in the full model mediates the mass univariate associations between the brain (DV) and the independent variable (IV) of interest.

To run a mediation analysis first run `FEMA_wrapper.m` using the following inputs to generate permuted statistics for the reduced and full models:

- **mediation =** 1
- **nperms =** number of permutations
- **PermType =** 'wildboostrap-nn' (non-null resampling procedure)
- **colsinterest** = vector of column indices for variables of interest in design matrix (all permuted output will be truncated to only save permutations for these variables in order to reduce the output file size)

It is important to run *`FEMA_wrapper.m`* with mediation set to 1 and both design matrices as inputs, because this will ensure the same resampling scheme is used to produce the permuted statistics for each model by using the same random seed for the wild bootstrap procedure. Sobel's test should not be run on output files from `FEMA_wrapper.m` that were <u>not</u> estimated from the same instance of `FEMA_wrapper.m`.

Running LME using `FEMA_wrapper.m`

Mediation analysis

Sobel's test can then be run on the output from `FEMA_wrapper.m` using `FEMA_sobel_test.m` *with the following inputs:*

- **FEMA_outfile_reduced** = path with filename for FEMA output for reduced model
- **FEMA_outfile_full** = path with filename for FEMA output for full model
- **IVname** = variable name of independent variable of interest for mediation model
- **alpha** = alpha level for calculation of confidence intervals (e.g. alpha=0.05 ~ 95% CI)
- *Optional input arguments:*
  - **save_params =** naming parameters needed to save outputs
  - **mask =** imaging mask (will save in output file if included)

This function will calculate the vertex/voxelwise mediation effects, Sobel's t-statistics and estimated p-values. Resampling is used to estimate the standard error of the mediation effect. This bootstrapping procedure does not rely on assumptions of normality.

For a demo see `~/github/FEMA/DEMOS/` **`FEMA_resampling_demo.m`**

Running on a cluster (for nperms>>1) `FEMA_wrapper.m`

- If running a great number of permutations (*n>100*), it is much faster to run on a cluster, if available. The number of permutations can be broken down by node, as the computations are independent, hence achieving a speedup of almost *n_nodes*.
- All in one approach:
  - Run the whole pipeline ‘hands-off’, with the ability to specify most required inputs (e.g. datatype, modalities, imaging phenotypes, etc.)
  - Use the `submit_cluster[.py]` tool (either Python or compiled binary)
  - For instructions or an example use, run: `python3 submit_cluster.py -h` or `./submit_cluster -h`
- Proceeding block by block:
  - The different steps can also be run independently if needed
  - Submit jobs (calls to `FEMA_wrapper`) to your cluster following the tool of your choice
  - Run `FEMA_cluster_gather`
  - If not specified as part of `FEMA_cluster_gather`, one can run `FEMA_perm_significance` to get TFCE or any other statistics significance
  - Follow each file’s documentation for more details

Visualization of Results

To visualize vertexwise FEMA output use the ‘showSurf’ package

- For details follow the ~/github/FEMA/DEMOS/**FEMA_showSurf_demo.m**

To visualize voxelwise FEMA output use the ‘showVol’ package

- For details follow the ~/github/FEMA/DEMOS/**FEMA_showVol_demo.m**

The voxelwise imaging mask, ‘vol_mask’, is saved with each FEMA_wrapper.m output. To convert an already masked vector of statistics into a 3D volume to view with showVol use the following line of code:

```
3D_volume = fullvol(vector, vol_mask)
```