AG Neuroimaging & Neuroengineering of Experimental Stroke
University of Cologne, Faculty of Medicine
and
University Hospital Cologne, Dept. of Neurology, Germany

Installation and User Guide

v1.1 Last Updated: 02.08.2021
Leon Scharwächter, Niklas Pallast, Felix Schmitt, Markus Aswendt

AIDAconnect

Table of Contents

1 Getting Started

1.1 Introduction

AIDAconnect is a comprehensive collection of MATLAB scripts for post-processing of mouse brain MRI data, whole brain connectivity, and neural network analysis using graph theory. It was developed for the automated analysis of experimental stroke imaging data (Pallast et al. 2020; Aswendt et al. 2020)but will work for a multitude of neuroimaging applications. Most basic graph theoretical functions, e.g. for calculating the degree, node strength, and shortest path were implemented using the Brain Connectivity Toolbox (Rubinov and Sporns 2009)The registration of the MRI data with the Allen Mouse Brain atlas (Wang et al. 2020) is the foundation of AIDAconnect to allow cross-study comparisons. The given tools are provided for anatomical T2-weighted MRI (T2w), diffusion tensor imaging (DTI) and resting-state functional MRI (rs-fMRI) data and are currently designed for creating weighted-undirected graphs only.

NOTE: This manual consists of three almost identical parts. Each part corresponds to one imaging technique (T2w, DTI, rs-fMRI). For this reason, you do not need to read the whole manual if your data only consists of one of these types.

1.2 Requirements

1.2.1 Data format and file structure

AIDAconnect requires the data to be in NIfTI-1 Data Format (https://nifti.nimh.nih.gov/nifti-1/) and saved according to a specific file structure *(see Figure 1*). Each group folder is divided into the respective subject folders containing the MRI data as well as a separate folder “Physio” which contains the related physiological monitoring data (breathing, anaesthetic dosage and temperature). Every MRI subject folder contains the pre-processed imaging files sorted into sub-folders, e.g. Localizer (to locate the animal in the scanner), T2, DTI, and fMRI. For the whole pre-processing, we recommend our software pipeline AIDAmri (Pallast et al. 2019) (https://github.com/maswendt/AIDAmri), which will automatically save the data in the presented folder structure. However, if you are using other MRI sequences than listed here or a different pre-processing pipeline, you need to adjust the folders manually.

In this version of AIDAconnect, several analysis functions are restricted to two groups. If you want to analyze more groups, you can load them sequentially and run the function again. If your dataset consists of only one group, you can simply specify this group twice.

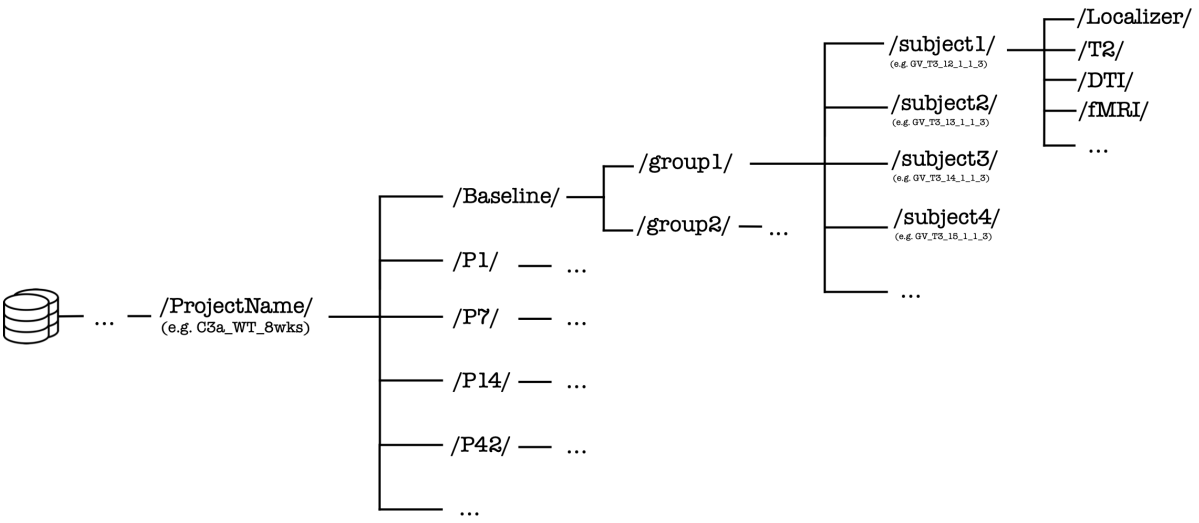


Figure 1: File structure required for AIDAconnect. The parent project folder contains all data sorted according to the measurement days – here baseline, and post stroke day 1 – 42 (P1-42). The time point folders contain the data of all experimental groups – here group 1 and 2. These groups contain the data from all subjects at the specified time point.

1.2.2 Atlas registration

The analysis is solely region-based, thus will only work with mouse MRI data registered with an atlas. With AIDAmri, the data is registered with a custom parental version of the Allen Mouse Brain atlas, ccf v3 (Wang, et al., 2020), composed of 98 regions, split between hemispheres. For more details how to generate a custom version of the atlas, see https://github.com/maswendt/AIDAmri. All necessary files are provided together with AIDAconnect. If a different atlas is used the corresponding acronym list and all related links in the code have to be adapted.

1.2.3 MATLAB

All processing steps were developed for MATLAB (tested for versions R2018a and R2019b). AIDAconnect requires the following MATLAB packages (free to download and use for academics):

- **Brain Connectivity Toolbox (BCT)** https://sites.google.com/site/bctnet/ (Rubinov and Sporns 2009)
- **FSLNets:** https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FSLNets
    - **NOTE:** AIDAconnect requires some modified FSL scripts. Therefore, the download of the latest FSLNets version is optional. The related files are located in the Tools folder.
- **NIfTI:** https://mathworks.com/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image

It is important to store the extracted packages' folders in AIDAconnect/Tools with the folder names as shown in Figure 2:

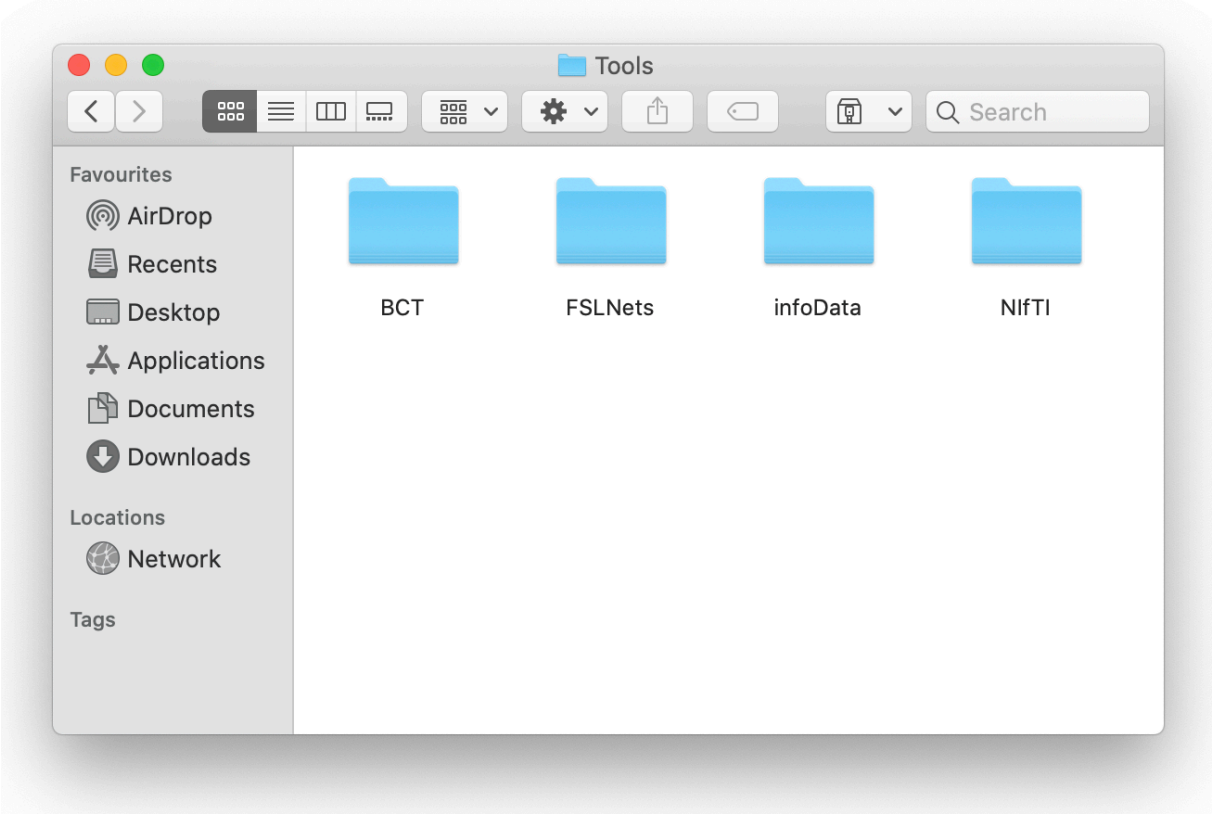


Figure 2: Necessary toolboxes and packages for AIDAconnect to work. Please make sure that the folder names match the names shown in this figure, otherwise AIDAconnect can not access the files.

1.2.4 List of BCT functions

Function	Measure	Network*
`clustering_coef_wu.m`	Clustering coefficient	WU
`participation_coef.m`	Participation coefficient	BU, BD, WU, WD
`degrees_und.m`	Degree	BU, WU
`strengths_und.m`	Node strength	WU
`betweenness_wei.m`	Betweenness centrality	WU, WD
`eigenvector_centrality_und.m`	Eigenvector centrality	BU, WU
`efficiency_wei.m`	Efficiency	WU, WD
`density_und.m`	Density	BU, WU
`transitivity_wu.m`	Transitivity	WU
`efficiency_wei.m`	Efficiency	WU, WD
`assortativity_wei.m`	Assortativity	WU, WD
`modularity_und.m`	Modularity	BU, WU

*BU, binary undirected networks; WU, weighted undirected networks; BD, binary directed networks; WD, weighted directed networks.

2 Analysis of T2-weighted MRI data

2.1 First steps

To work with T2-weighted data, all data have to be consolidated into new MAT-files using *mergeT2data_input.m*. All MATLAB-files used for T2-analysis are stored in *AIDAconnect/T2*. We strongly recommend to change the working path in MATLAB to the path of AIDAconnect (a dialog box will ask you when you run the script). The following lines of *mergeT2data_input.m* have to be adapted to each new dataset to run the script:

- Line 8: path to the project folder (automatically created by AIDAmri),
- Line 11: list of the days of measurement in ascending order,
- Line 14: names of all groups,
  Line 17: output path.

Every information needs to be entered as a string with quotation marks. After the execution of the script, MAT-files will be created into the specified output path and can be used for AIDAconnect analysis. The generated files are named after the specified days, in which all information are saved as MATLAB-structs for each particular day. The script will create another MATLAB-struct called *inputT2,* which stores the information entered in the script before. In this version, AIDAconnect/T2 does not use this struct for further analysis. Additionally, the acronym list for the regions is loaded to the Workspace.

mergeT2data_input.m

This script consolidates all data from the respective groups and days into single MAT-files by adjusting the input information. Please hit 'Run' after the information is entered.

```
1   %% First Steps
2   % Consolidates all information of the processed T2-weighted MRI-data.
3   % Please specify all information below and hit 'Run'.
4
5   %% Specifications
6
7   % Path to the processed image data folder (e.g. proc_data)
8   inputT2.in_path = "/Users/Username/Documents/Projects/proc_data";
9
10  % Observation days (e.g. "P1" etc.)
11  inputT2.days = ["Baseline","P1","P7","P14","P21","P28"];
12
13  % Groups (e.g. "Sham" etc.)
14  inputT2.groups = ["SP_T1.2.2","SP_T1.2.4"];
15
16  % Output path
17  inputT2.out_path = "/Users/Username/Documents/Projects/proc_data/outputT2";
18
19  %% Do not modify the following lines
```

After successful processing the following MAT-files are generated, which are ready for the import into MATLAB *(see red lines in Figure 3)*. The structs stored in the MAT-files can be loaded with the command `load()` in MATLAB, by double-clicking or per drag&drop. For loading multiple MAT-files, we recommend to use an abbreviation, e.g. the group and day:

```
group1_P1 = load('/…/ProjectName/outPath/T2/group1/P1.mat');
group1_P7 = load('/…/ProjectName/outPath/T2/group1/P7.mat');
```

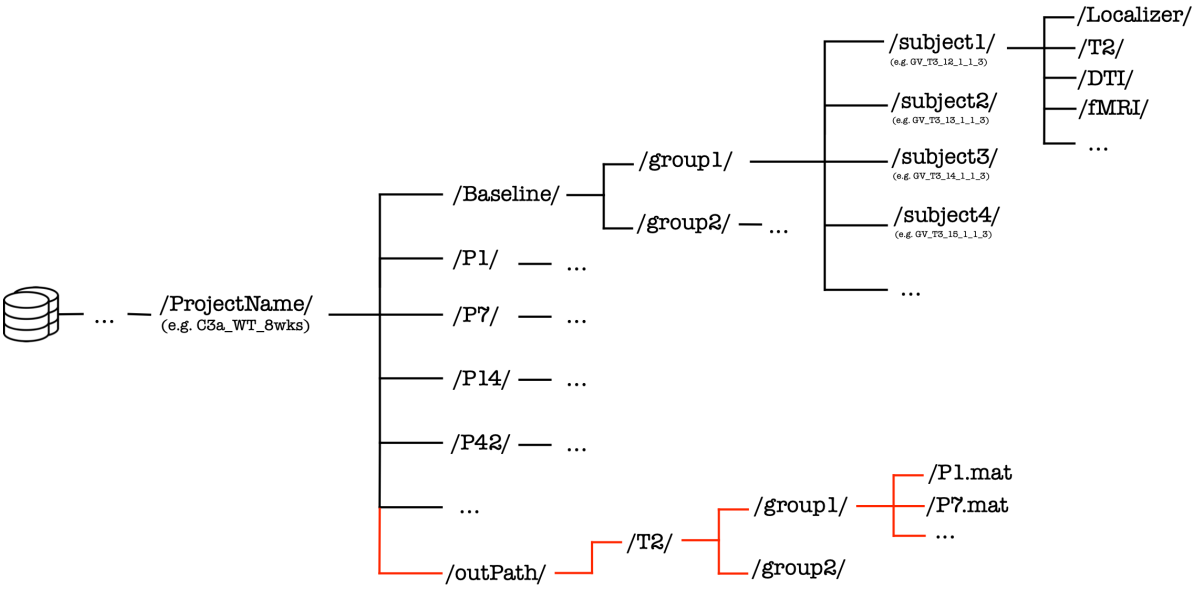


Figure 3: File structure after the consolidation of the T2-weighted data using mergeT2data_input.m. The red lines indicate the new generated path, which contains the new MAT-files sorted by group and day.

2.2 Information stored in the files

The properties are calculated on the basis of the previously-defined Regions of Interest (ROI) - in our example stroke masks to delineate the hyperintense stroke region on T2w-MRI. Each MAT-file consists of a struct with the name *infoT2* and contains the following fields *(see Table below)*. The size of some fields depends on the number of subjects X. The X is only a placeholder for the number of animals in the group (see next page).

The following fields contain general information about the loaded struct (e.g. P7.mat), information about the lesion size and the affected regions. Every field can be read by calling the field name:

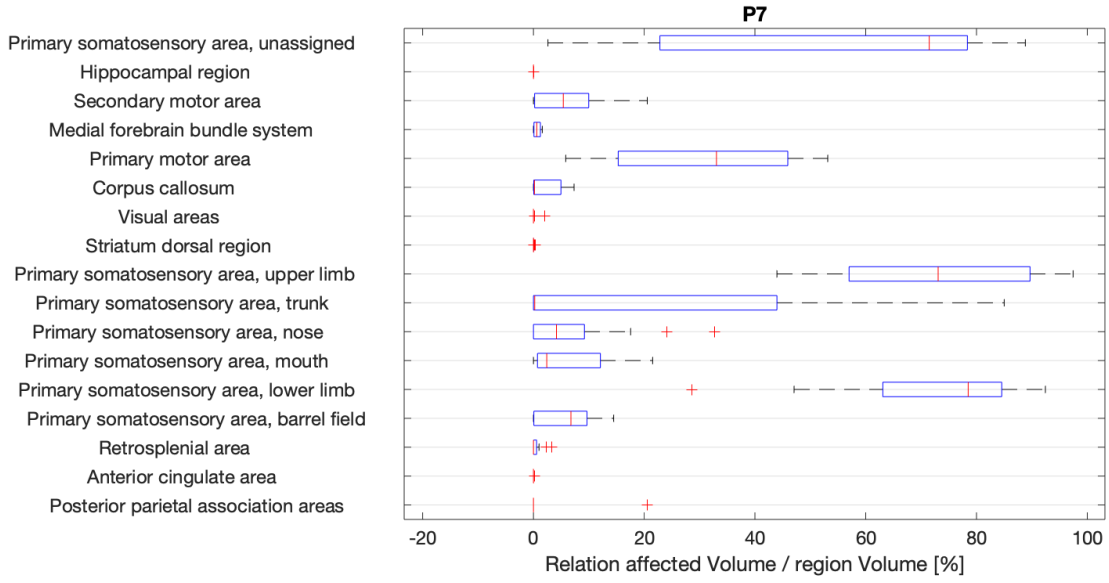
Field Name	Description	Output Type
`infoT2.group`	Group name	`(string)`
`infoT2.day`	Day of data acquisition	`(string)`
`infoT2.names`	Subject names within the group	`{X×1 cell}`
`infoT2.labels`	49 atlas labels of the regions (not split between hemispheres)	`[49×1 string]`
`infoT2.lesionSize_mm`	Lesion size in mm for each subject	`[X×1 double]`
`infoT2.lesionSize_percent`	Lesion size in percent (related to the masked whole brain size) for each subject	`[X×1 double]`
`infoT2.affectedRegions_percent`	Lesion volume in percent (related to the region size) for each of the 49 regions for each subject	`[X×49 double]`

2.3 T2w-MRI-analysis functions

`plotStrokeSize_Percent(path).m`

The lesion volume in percent for each of the 49 regions can be displayed by transferring the corresponding path (as String) of the MAT-file.

`Example: plotStrokeSize_Percent('/Volumes/MRI/proc_data/outputT2/Group1/P1.mat')`



3 Analysis of DTI data

3.1 First steps

To work with DTI data, all data have to be consolidated into new MAT-files using *mergeDTIdata_input.m*. All MATLAB-files used for DTI-analysis are stored in *AIDAconnect/DTI*. We strongly recommend to change the working path in MATLAB to the path of AIDAconnect (a dialog box will ask you when you run the script). The following lines of *mergeDTIdata_input.m* have to be adapted to each new dataset to run the script:

- Line 8: path to the project folder (automatically created by AIDAmri),
- Line 11: list of the days of measurement in ascending order,
- Line 14: names of all groups,
- Line 17: AIDAconnect includes all fibers larger than a given threshold,
- Line 20: output path.

Every information needs to be entered as a string with quotation marks. As the threshold is a decimal number between 0 and 1, the neglected fibers are those smaller than the mean number of fibers multiplied by the threshold. After the execution of the script, MAT-files will be created into the specified output path and can be used for AIDAconnect analysis. The generated files are named after the specified days, in which all information are saved as MATLAB-structs for each particular day. The script will create another MATLAB-struct called *inputDTI,* which stores the information entered in the script before.

mergeDTIdata_input.m

This script consolidates all data from the respective groups and days into single MAT-files by adjusting the input information. Please hit 'Run' after the information is entered.

```
1   %% First Steps
2   % Consolidates all graph-theoretical measures of the processed DTI-data.
3   % Please specify all information below and hit 'Run'.
4
5   %% Specifications
6
7   % Path to the processed image data folder (e.g. proc_data)
8   inputDTI.in_path = "/Users/Username/Documents/Projects/proc_data";
9
10  % Observation days (e.g. "P1" etc.)
11  inputDTI.days = ["Baseline","P7","P14","P28","P42","P56"];
12
13  % Groups (e.g. "Sham" etc.)
14  inputDTI.groups = ["Treatment_C3a","Treatment_PBS"];
15
16  % Threshold (0-1)
17  thres = 0;
18
19  % Output path
20  inputDTI.out_path = "/Users/Username/Documents/Projects/proc_data/outputDTI";
21
22  %% Do not modify the following lines
```

Additionally, this script loads the acronym list for all regions and creates the MATLAB-cell-arrays *graphCell*, *matrixValues* and *ids* that contain graph-theoretical measures, connectivity matrices and the subject names for all groups and days. You can manually investigate these files to see the respective properties (see Chapter 3.2 Information stored in the files and Chapter 3.3 DTI-analysis functions). *inputDTI* and *graphCell* are needed as input parameters for several analysis functions. For the usage of these files in a new MATLAB-session, just run this script again. As long as the output path already exists, the files will not be merged again and only *inputDTI* and the cell arrays will be created. For connectivity analysis, a graph theoretical model is built by AIDAconnect using the Brain Connectivity Toolbox (Rubinov & Sporns, 2010).

After successful processing the following MAT-files are generated, which are ready for the import into MATLAB *(see red lines in Figure 4).* The content of the MAT-files can be loaded with the command `load()` in MATLAB, by double-clicking or per drag&drop. For loading multiple MAT-files, we recommend to use an abbreviation, e.g. the group and day:

```
group1_P1 = load('/…/ProjectName/outPath/DTI/group1/P1.mat');
group1_P7 = load('/…/ProjectName/outPath/DTI/group1/P7.mat');
```

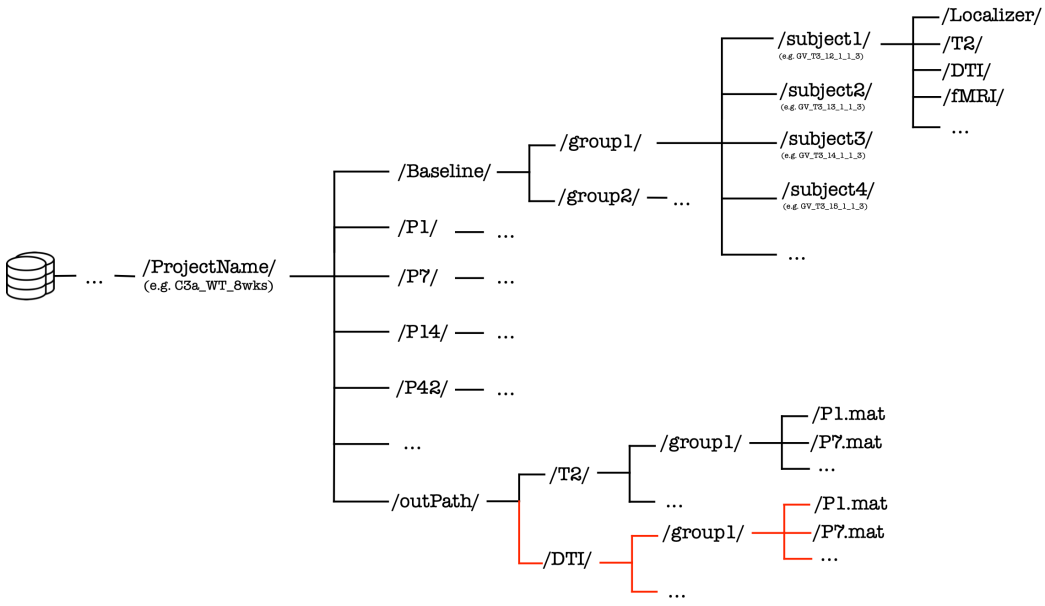


Figure 4: File structure after the consolidation of the DTI-data using mergeDTIdata_input.m. The red lines indicate the new generated path, which contains the new MAT-files sorted by group and day.

3.2 Information stored in the files

The properties are calculated on the basis of the output of DSIstudio (http://www.dsi-studio.labsolver.org). Each MAT-file consists of a struct with the name *infoDTI* and contains the following structure *(see tables on the next page).* The content varies with the number of subjects X. The X is only a placeholder for the number of animals in the group. The following fields contain general information about the loaded struct (e.g. P7.mat). Every field can be read by calling the field name.

Field Name	Description	Output Type
infoDTI.group	Group Name	(string)
infoDTI.day	Day of data acquisition	(string)
infoDTI.names	Subject names within the group	{X×1 cell}
infoDTI.matrix	Connectivity Matrix for 98x98 atlas regions for each subject	[98x98xX double]
infoDTI.labels	Labels of the 98 atlas regions	{98×1 cell}

The following fields below contain the diffusivity and anisotropy parameters of the DTI measurement. The diffusivity along the principal axis (λ_1) is the Axial Diffusivity (AD). Fractional Anisotropy (FA0) is an index for the amount of diffusion asymmetry within a voxel. The Mean Diffusivity (MD) is the average diffusivity of all three eigenvalues. Radial Diffusivity (RD) is the mean orthogonal diffusivity regarding the principal axis (λ_1).

$$FA0 = \sqrt{\frac{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_1 - \lambda_3)^2}{2(\lambda_1^2 + \lambda_2^2 + \lambda_3^2)}}$$

$$MD = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} \qquad RD = \frac{\lambda_2 + \lambda_3}{2}$$

Field Name	Description	Output Type
infoDTI.AD	AD values of all subjects for each 98 regions	[Xx98 double]
infoDTI.FA0	FA0 values of all subjects for each 98 regions	[Xx98 double]
infoDTI.MD	MD values of all subjects for each 98 regions	[Xx98 double]
infoDTI.RD	RD values of all subjects for each 98 regions	[Xx98 double]

The following fields contain local parameters of the graph-theoretical model:

Field Name	Description*	Output Type
infoDTI.clustercoef	The weighted clustering coefficient is the fraction of triangles around a node for all 98 regions	[Xx98 double]
infoDTI.participation-coef	The participation coefficient measures if the connections of a node are clustered into one module or are distributed over all modules	[Xx98 double]
infoDTI.degrees	Node degree is the number of links connected to the nodes of the 98 regions for each subject	[Xx98 double]
infoDTI.strengths	Node strength is the sum of weights of links connected to the nodes of the 98 regions for each subject	[Xx98 double]
infoDTI.betw_centrality	Node betweenness centrality is the fraction of all shortest paths the network that contain a given node. Nodes with high values of betweenness centrality participate in a large number of shortest paths of the 98 regions for each subject	[Xx98 double]
infoDTI.eign_centrality	Eigenvector centrality is a self-referential measure of centrality: nodes have high eigenvector centrality if they connect to other nodes that have high eigenvector centrality	[Xx98 double]

Field Name	Description*	Output Type
infoDTI.localEfficiency	The local Efficiency is the global Efficiency (see below) computed on node neighborhoods and is related to the clustering coefficient	[Xx98 double]

The following fields contain global parameters of the graph-theoretical model:

Field Name	Description*	Output Type
infoDTI.density	Density is the fraction of present connections to possible connections for each subject	[Xx1 double]
infoDTI.transitivity	Transitivity is the ratio of 'triangles to triplets' in the network for each subject	[Xx1 double]
infoDTI.efficiency	The global efficiency is the average of inverse shortest path length for each subject	[Xx1 double]
infoDTI.assortativity	A positive assortativity coefficient indicates that nodes tend to link to other nodes with the same or similar strength	[Xx1 double]
infoDTI.modularity	Modularity is the subdivision of the network into dense modules	[Xx1 double]
infoDTI.charPathLength	The characteristic path length is the average shortest path length between all pairs of nodes in the network for each subject	[Xx1 double]
infoDTI.smallWorldness	Indicates if a node can be reached from another random node by a small number of hops. A network fulfils this property when S > 1	[Xx1 double]

* Descriptions based on Brain Connectivity Toolbox (Rubinov & Sporns, 2010)

3.3 DTI-analysis functions

[graphCell,matrixValues,ids] = graphAnalysis_DTI(inputDTI).m

This function is used by mergeDTIdata_Input.m and is not meant to be used manually. It builds the graph-theoretical model. This model represents the regions as nodes and the fibres as edges, while the number of fibres determine the edge weight.

Output descriptions:

graphCell{G, D} contains Edges and Nodes for each group and day obtained by the use of graph theory. You can manually investigate this cell array: G represents the number of groups and D the number of days. Rows stand for the groups and columns for the days. Each cell represents a graph for a group and a day that can be called, e.g. graphCell{1, 2} for group 1 and time point 2. This graph has two attributes: Edges and Nodes. You can call them via dot notation: graphCell{1, 2}.Edges and graphCell{1, 2}.Nodes.

matrixValues{G, D} contains the mean correlation among each region for each group and day over all subjects. To see a specific correlation matrix, replace G and D by the desired number, e.g. matrixValues{1, 2} for group 1 and time point 2.

ids{G, D} contains all subject names for all groups and days. For example, you can see the subject names of the first group at the second day by ids{1, 2}. You can address a specific entry of the resulting cell array, e.g. ids{1, 2}{3} for the third entry of the first group at the second day.

Usage: – No manual usage –

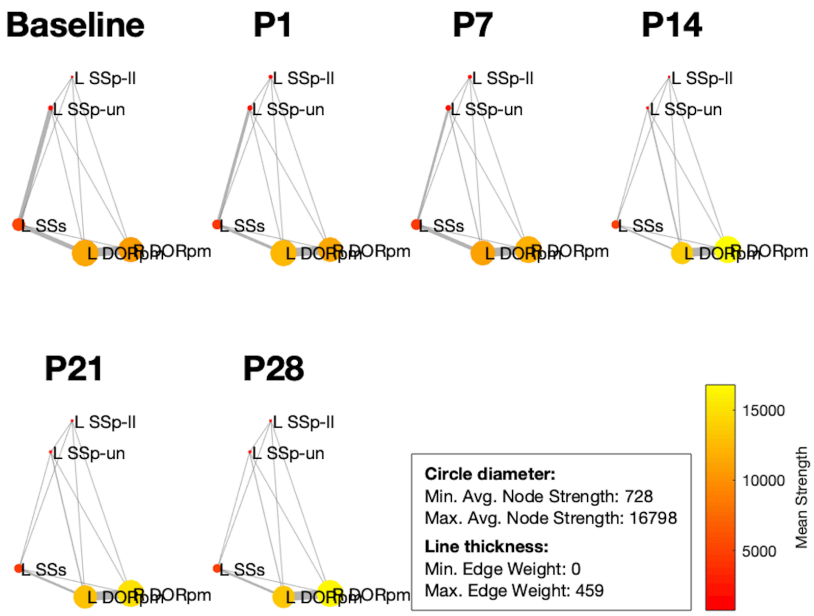
...

Output Value	Description	Output Type
`graphCell`	Each entity in *graphCell* contains Edges and Nodes for each group G and each day D	`{G×D cell array}`
`matrixValues`	Each entity in *matrixValues* contains the mean connectivity matrix values of each group G and each day D	`{G×D cell array}`
`ids`	Each entity in *ids* contains the names of each subject for each group G and each day D	`{G×D cell array}`

`plotSelectedRegions(inputDTI, graphCell).m`

This function represents the graph model with regions as nodes and edges as the number of fibres. The nodes have the anatomically correct position in the display because the focal points of each region were previously assigned to a node. Each node is color-coded according to its degree or strength. Which regions should be displayed can be selected using the selectedRegions variable in line 10. The first argument is the struct inputDTI. The second argument is the cell array graphCell. Remember that in a new session both parameters have to be created manually again (see mergeDTIdata_input.m) The plots are called up according to the groups in infoDTI.groups.

`Example: plotSelectedRegions(inputDTI, graphCell)`



`regionDetectionAnalysis.m`

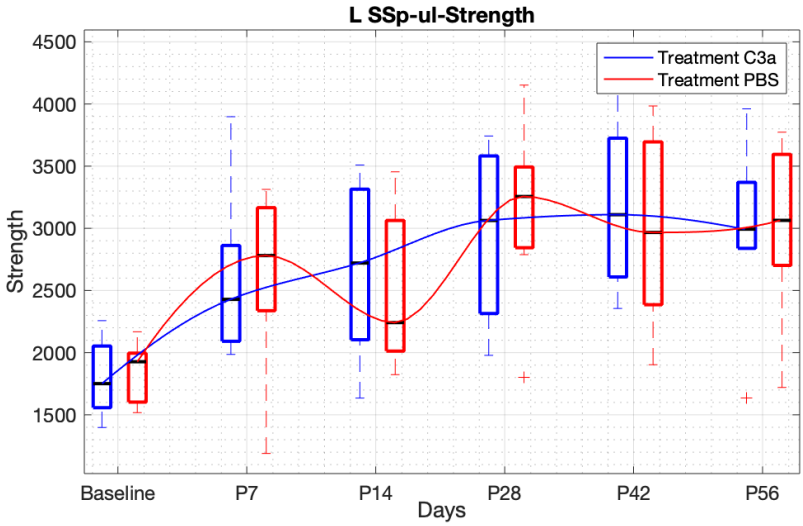
This script plots the difference between two connectivity matrices. Only regions whose connectivity is higher than a certain threshold are shown. You can specify the matrices inside the script in line 12 and 13 and the threshold in line 10.

`Usage: Fill in the paths to both matrices, enter a threshold and hit 'Run'.`

plotLocalParameter(inputDTI, graphCell, strNodeName, strParameter).m

This function uses graphCell to calculate the properties ('Degree', 'Strength', 'Eigenvector', 'Betweenness', 'Efficiency', 'Clusteringcoefficient', 'Participationcoefficient', 'AD', 'FA0', 'MD', 'RD') of the region strNodeName based on the information by inputDTI. The selected property is displayed for all groups over all days. Remember that in a new session the input parameter inputDTI and the cell array graphCell have to be created manually again (see mergeDTIdata_input.m). The selected property is represented by the argument strParameter.

```
Example: plotLocalParameter(inputDTI, graphCell, 'L SSp-ul', 'Strength')
```



shortestPath(inputDTI, day2examine, startNode, endNode, groupAverage).m

This function displays the shortest path from one region to another. The shortest path is calculated as an inverse Dijkstra's Algorithm, meaning that this function seeks for the shortest path with the hightest sum of fibers representing the best strengthened path. The first argument is inputDTI. Remember that in a new session this parameter has to be created manually again (see mergeDTIdata_input.m). day2examine refers to the day to analyze (as an integer, position of inputDTI.days). startNode and endNode correspond to the regions (as strings) whose shortest path should be determined. groupAverage is an optional input argument: You can either choose to display the mean shortest path for all groups (=1, default value) or choose to display the shortest path for each subject of the groups (=0)

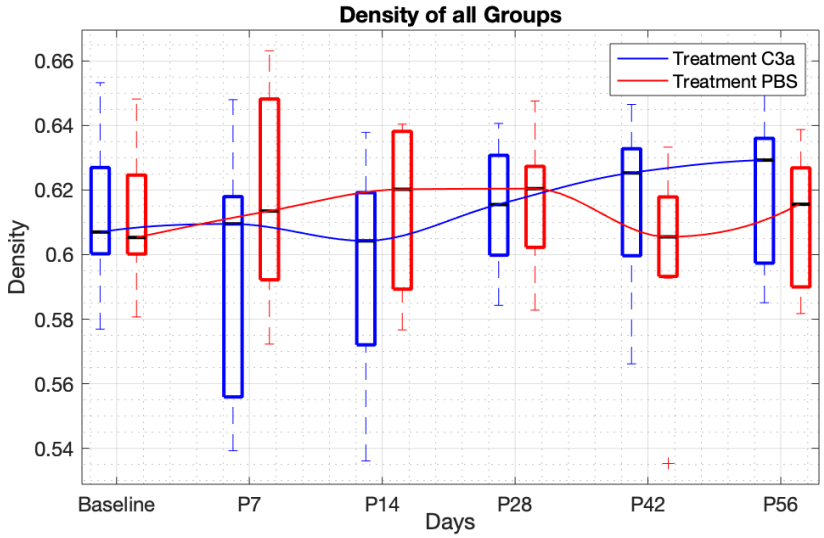
The output is displayed in the Command Window of MATLAB. The output parameter 'Path Length' refers to the distance of the indicated nodes calculated on the basis of the inverted number of fibers. A lower value corresponds to a stronger connection.

```
Example: shortestPath(inputDTI, 1, 'R sAMY', 'L SSs')
```

plotGlobalParameter(inputDTI, graphCell, strParameter).m

This function uses inputDTI and graphCell to display global graph properties of the whole network. The selected parameter ('Density', 'Transitivity', 'Efficiency', 'Assortativity', 'Modularity', 'charPathLength' or 'smallWorldness') is displayed for all groups over all days. Remember that in a new session the input parameter inputDTI and the cell array graphCell have to be created manually again (see mergeDTIdata_input.m). The selected property is represented by the argument strParameter.

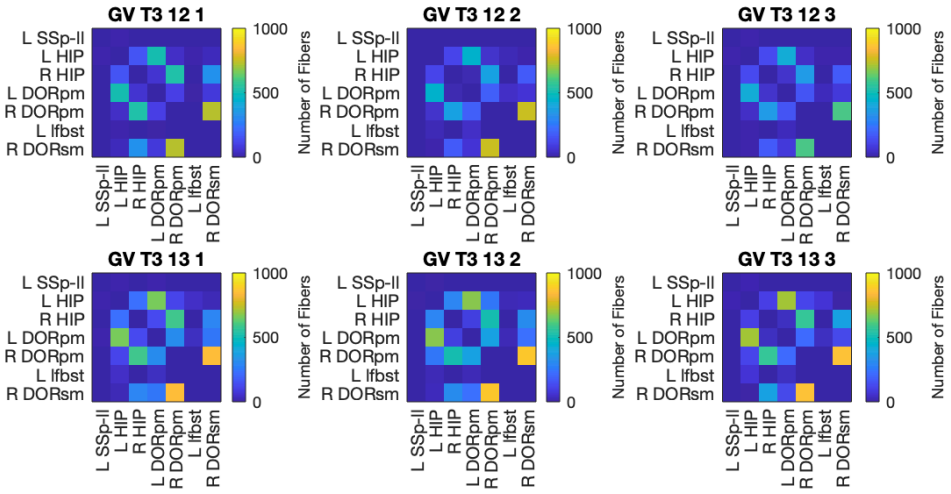
Example: `plotGlobalParameter(inputDTI, graphCell, 'Density')`



plotFiberCountMatrices.m

This script displays scatter plots of the number of fibers of selected regions (figure 1) and their mean values and standard deviation (figure 2). Please specify the path to a processed MAT-file and the selected regions in line 8 and 9 of the script. You can also adjust the colour range of the plot in line 10.

Usage: Fill in the specifications and hit 'Run'.



analyzeLostConnections(inputDTI, graphCell, selectedRegion, day1, day2).m

This function finds all connections of a selected region per group that are not present in the second time point but do have connection weights in the first time point. The first two arguments are inputDTI and graphCell. Remember that in a new session both structs have to be created manually again (see mergeDTIdata_input.m). The third argument defines the region to examine. The fourth and fifth arguments specify the time points. You can specify a weighting threshold in line 17 to neglect lost connections that only had small weights. This threshold represents the number of fibers. The output is displayed in the Command Window of MATLAB for all groups.

```
Example: analyzeLostConnections(inputDTI, graphCell, 'L SSp-ll', 'Baseline',
'P7')
```

analyzeMostConnected(inputDTI, graphCell, RegionPF, day, out_path).m

*This function finds the most connected regions of a seed region per group for a selected time point. The first two arguments are inputDTI and graphCell. Remember that in a new session both structs have to be created manually again (see mergeDTIdata_input.m). The third argument defines the region to examine. Please enter the region without a specification of the hemisphere (L-, R-). The plot will automatically display both sides. The fourth argument specifies the time point. The last argument is the output path where the table and/or the figures can be saved as a *.csv-file and *.pdf-files. If you keep this argument empty, no table/figure will be exported. You can define the number of the most connected regions in line 12 of the function and further options in line 13 (size of nodes/edges), 17 (export table) and 18 (export figures).*

```
Example: analyzeMostConnected(inputDTI, graphCell, 'SSs', 'P1')
```

analyzeNewConnections(inputDTI, graphCell, selectedRegion, day1, day2).m

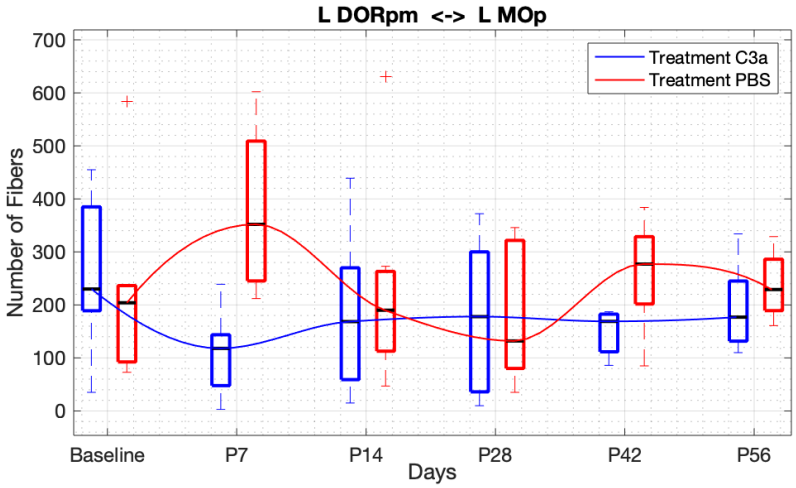
This function finds all connections of a selected region per group that are not present in the first time point but do have connection weights in the second time point. The first two arguments are inputDTI and graphCell. Remember that in a new session both structs have to be created manually again (see mergeDTIdata_input.m). The third argument defines the region to examine. The fourth and fifth arguments specify the time points. You can specify a weighting threshold in line 17 to neglect new connections with small weights. This threshold represents the least number of fibers. The output is displayed in the Command Window of MATLAB for all groups.

```
Example: analyzeNewConnections(inputDTI, graphCell, 'L SSp-ll', 'Baseline', 'P7')
```

plotConnectionWeight(inputDTI, graphCell, startP, endP).m

This function represents the change of the edge weight over time for all groups. inputDTI and graphCell represent the first two input values in order to draw the information from the corresponding study. Remember that in a new session both input parameters have to be created manually again (see mergeDTIdata_input.m). startP and endP represent the regions whose edge weight should be analyzed. Please pass the regions as strings.

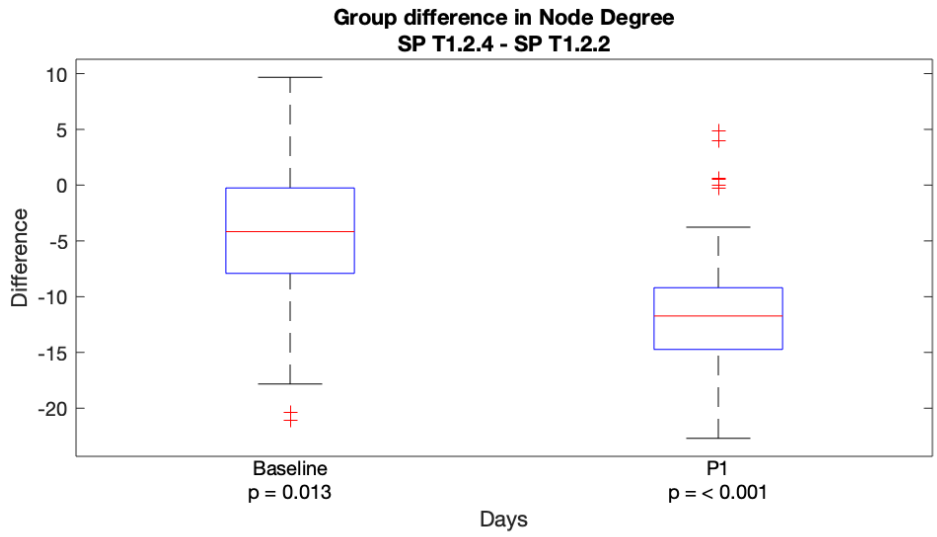
```
Example: plotConnectionWeight(inputDTI, graphCell, 'L DORpm', 'L MOp')
```



compareTotalDegree(inputDTI, graphCell, day1, day2, out_path).m

*This function collects all degrees of all nodes and compares them with a paired t-test on selected (two) days. The first argument is the struct inputDTI. The second argument is the cell array graphCell. Remember that in a new session both parameters have to be created manually again (see mergeDTIdata_input.m). The third and fourth argument represent the selected days as an integer (position of inputDTI.days in ascending order). In the last argument you may specify an output path where the plot can be saved as a *.pdf-file. If you keep this argument empty, no plot will be exported.*

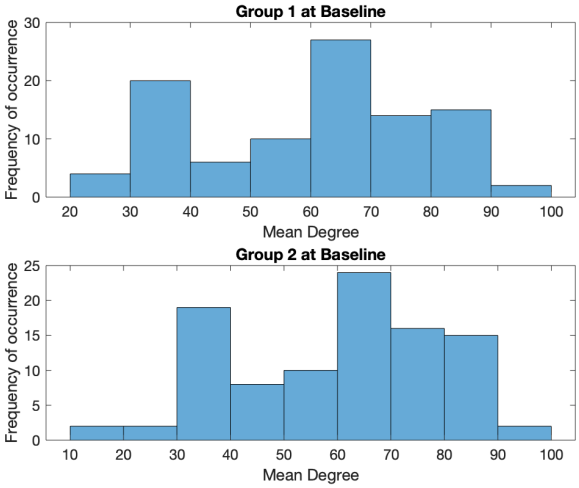
```
Example: compareTotalDegree(inputDTI, graphCell, 1, 2)
```



plotDistribution(inputDTI, graphCell, strParameter, day).m

This function displays the distribution of either the node degree or the node strength for all groups as a histogram for a given day. This way, you can investigate how many regions of the brain have degree/strength values between a certain range. inputDTI and graphCell represent the first two input values in order to draw the information from the corresponding study. Remember that in a new session both input parameters have to be created manually again (see mergeDTIdata_input.m). The third argument defines the graph parameter and accepts either 'Degree' or 'Strength'. The fourth parameter specifies the time point (as in inputDTI.days). Please define the parameter and the time point as strings. You can furthermore display all region's acronyms that have degree/strength values in a specific range if you enter the lower and upper bound in lines 16 and 17 of the function. The output is displayed in the command window of MATLAB.

```
Example: plotDistribution(inputDTI, graphCell, 'Degree', 'Baseline')
```



plotSignificantMatrix(inputDTI, day, out_path).m

This function checks for differences in connectivity between two groups for a certain day. It tests all connections in the connectivity matrix for all 98 regions and plots a matrix which shows a 1 (yellow pixel) if a significant difference between the groups is detectable (all connections with p-Value < 0.05 are displayed as 1 in the plot). The first argument is the struct inputDTI. Remember that in a new session this struct has to be created manually again (see mergeDTIdata_input.m). The second argument is the day to examine. In the last argument you may specify an output path where the plot can be saved as significantConn.pdf. If you keep this argument empty, no plot will be exported.

```
Example: plotSignificantMatrix(inputDTI, 'P28')
```

3.3.1 Used data structure

You typically want to ask yourself: What kind of graphs do exist in group 1 on the second day? You can investigate graph parameters using graphCell{G, D}. The output of a specific call is divided into two main parameters: Nodes (regions) and Edges (connections).

Nodes have the following properties:

```
Name, XCoord, YCoord, ZCoord, allMatrix, allDegree, allStrength, allEigenvector,
allBetweenness, FA0, AD, MD, RD
```

Edges have the following properties:

```
EndNodes, Weight
```

These properties can be read out using dot notation, e.g.:

```
graphCell{1,2}.Nodes.allDegree
graphCell{1,2}.Edges.Weight
```

to display the Node Degree and the Weights of all Edges of the first group on the second time point.

4 Analysis of rs-fMRI data

4.1 First steps

To work with rs-fMRI data, all data have to be consolidated into new MAT-files using *mergeFMRIdata_input.m*. All MATLAB-files used for rs-FMRI-analysis are stored in *AIDAconnect/fMRI*. We strongly recommend to change the working path in MATLAB to the path of AIDAconnect (a dialog box will ask you when you run the script). The following lines of *mergeFMRIdata_input.m* have to be adapted to each new dataset to run the script:

- Line 8: path to the project folder (automatically created by AIDAmri),
- Line 11: list of the days of measurement in ascending order,
- Line 14: names of all groups,
- Line 17: AIDAconnect includes all correlations larger than a given threshold,
- Line 20: output path.

Every information needs to be entered as a string with quotation marks. After the execution of the script, MAT-files will be created into the specified output path and can be used for AIDAconnect analysis. The generated files are named after the specified days, in which all information are saved as MATLAB-structs for each particular day. The script will create another MATLAB-struct called *inputFMRI,* which stores the information entered in the script before.

mergeFMRIdata_input.m

This script consolidates all data from the respective groups and days into single MAT-files by adjusting the input information. Please hit 'Run' after the information is entered.

```
1   %% First Steps
2   % Consolidates all graph-theoretical measures of the processed rsfMRI-data.
3   % Please specify all information below and hit 'Run'.
4
5   %% Specifications
6
7   % Path to the processed image data folder (e.g. proc_data)
8   inputFMRI.in_path = "/Users/Username/Documents/Projects/proc_data";
9
10  % Observation days (e.g. "P1" etc.)
11  inputFMRI.days = ["Baseline","P7","P14","P28","P42","P56"];
12
13  % Groups (e.g. "Sham" etc.)
14  inputFMRI.groups =  ["Treatment_C3a","Treatment_PBS"];
15
16  % Threshold (0-1)
17  thres = 0.1;
18
19  % Output path
20  inputFMRI.out_path = "/Users/Username/Documents/Projects/proc_data/outputFMRI";
21
22  %% Do not modify the following lines
```

Additionally, this script loads the acronym list for all regions and creates the MATLAB-cell-arrays *graphCell*, *matrixValues* and *ids* that contain graph-theoretical measures, correlation matrices and the subject names for all groups and days. You can manually investigate these files to see the respective properties (see Chapter 4.2 Information stored in the files and Chapter 4.3 rs-fMRI-analysis functions).

inputFMRI and *graphCell* are needed as input parameters for several analysis functions. For the usage of these files in a new MATLAB-session, just run this script again. As long as the output path already exists, the files will not be merged again and only *inputFMRI* and the cell arrays will be created. For connectivity analysis, a graph theoretical model is built by AIDAconnect using the Brain Connectivity Toolbox (Rubinov & Sporns, 2010).

After successful processing the following MAT-files are generated, which are ready for the import into MATLAB *(see red lines in Figure 5)*. The structs stored in the MAT-files can be loaded with the command `load()` in MATLAB, by double-clicking or per drag&drop. For loading multiple MAT-files, we recommend to use an abbreviation, e.g. the group and day:

```
group1_P1 = load('/…/ProjectName/outPath/fMRI/group1/P1.mat');
group1_P7 = load('/…/ProjectName/outPath/fMRI/group1/P7.mat');
```

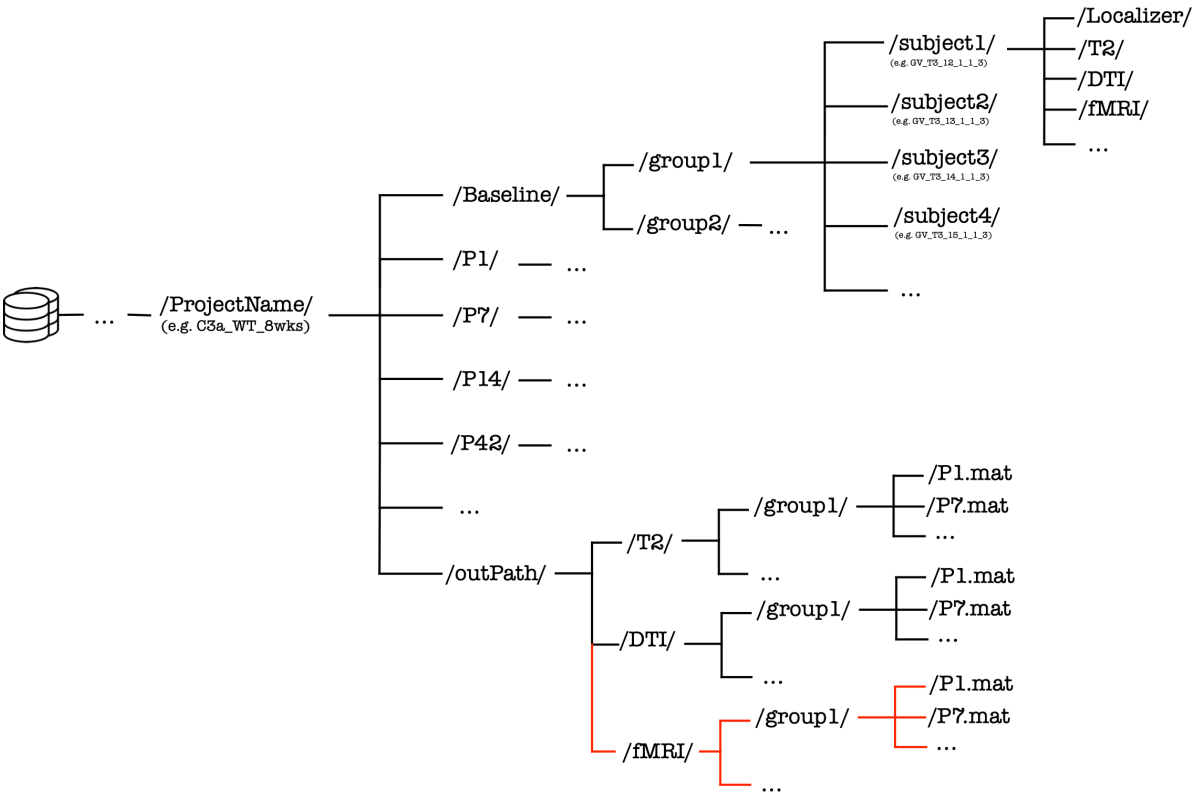


Figure 5: File structure after the consolidation of the fMRI-data using mergeFMRIdata_input.m. The red lines indicate the new generated path, which contains the new MAT-files sorted by group and day.

HINT: In this version of AIDAconnect, the absolute value of negative correlations is taken. This way, the impact of negative correlations is not fully discarded, but incorporated in the graph as a kind of correlation. This leads to correlation matrices that are in the range of 0 – 1.

4.2 Information stored in the files

The properties are calculated on the basis of the output of FSLnets (https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FSLNets). Each MAT-file consists of a struct with the name *infoFMRI* and contains the following structure *(see tables on the next page)*. The content varies with the number of subjects X. The X is only a placeholder for the number of animals in the group.

The following fields contain general information about the loaded struct (e.g. P7.mat). Every field can be read by calling the field name:

Field Name	Description	Output Type
infoFMRI.group	Group Name	(string)
infoFMRI.day	Day of data acquisition	(string)
infoFMRI.names	Subject names within the group	{X×1 cell}
infoFMRI.matrix	Connectivity Matrix for 98x98 atlas regions for each subject	[98x98xX double]
infoFMRI.labels	Labels of the 98 atlas regions	{98×1 cell}

The following fields contain local parameters of the graph-theoretical model:

Field Name	Description*	Output Type
infoFMRI.clustercoef	The weighted clustering coefficient is the fraction of triangles around a node for all 98 regions	[Xx98 double]
infoDTI.participation-coef	The participation coefficient measures if the connections of a node are clustered into one module or are distributed over all modules	[Xx98 double]
infoFMRI.degrees	Node degree is the number of links connected to the nodes of the 98 regions for each subject	[Xx98 double]
infoFMRI.strengths	Node strength is the sum of weights of links connected to the nodes of the 98 regions for each subject	[Xx98 double]
infoFMRI.betw_centrality	Node betweenness centrality is the fraction of all shortest paths the network that contain a given node. Nodes with high values of betweenness centrality participate in a large number of shortest paths of the 98 regions for each subject	[Xx98 double]
infoFMRI.eign_centrality	Eigenvector centrality is a self-referential measure of centrality: nodes have high eigenvector centrality if they connect to other nodes that have high eigenvector centrality	[Xx98 double]
infoFMRI.localEfficiency	The local Efficiency is the global Efficiency (see below) computed on node neighborhoods and is related to the clustering coefficient	[Xx98 double]

The following fields contain global parameters of the graph-theoretical model:

Field Name	Description*	Output Type
infoFMRI.density	Density is the fraction of present connections to possible connections for each subject	[Xx1 double]
infoFMRI.transitivity	Transitivity is the ratio of 'triangles to triplets' in the network for each subject	[Xx1 double]
infoFMRI.efficiency	The global efficiency is the average of inverse shortest path length for each subject	[Xx1 double]
infoFMRI.assortativity	A positive assortativity coefficient indicates that nodes tend to link to other nodes with the same or similar strength	[Xx1 double]
infoFMRI.modularity	Modularity is the subdivision of the network into dense modules	[Xx1 double]
infoFMRI.charPathLength	The characteristic path length is the average shortest path length between all pairs of nodes in the network for each subject	[Xx1 double]
infoFMRI.smallWorldness	Indicates if a node can be reached from another random node by a small number of hops. A network fulfils this property when S>1	[Xx1 double]
infoFMR.overalconnectivity	In addition to the fixed threshold method, the overall connectivity is calculated as one important additional measure for the comparison between networks with different average degree/number of nodes (Wijk, Stam, and Daffertshofer 2010).	[Xx1 double]

* Descriptions based on Brain Connectivity Toolbox (Rubinov & Sporns, 2010)

4.3 rs-fMRI-analysis functions

[graphCell,matrixValues,ids] = graphAnalysis_DTI(inputDTI).m

This function is used by mergeFMRIdata_Input.m and is not meant to be used manually. It builds the graph-theoretical model. AIDAconnect measures the correlation coeffient of the BOLD-Signals between regions to represent these regions as nodes and the fibres as edges.

Output descriptions:

graphCell{G, D} contains Edges and Nodes for each group and day obtained by the use of graph theory. You can manually investigate this cell array: G represents the number of groups and D the number of days. Rows stand for the groups and columns for the days. Each cell represents a graph for a group and a day that can be called, e.g. graphCell{1, 2} for group 1 and time point 2. This graph has two attributes: Edges and Nodes. You can call them via dot notation: graphCell{1, 2}.Edges and graphCell{1, 2}.Nodes.

matrixValues{G, D} contains the mean correlation among each region for each group and day over all subjects. To see a specific correlation matrix, replace G and D by the desired number, e.g. matrixValues{1, 2} for group 1 and time point 2.

ids{G, D} contains all subject names for all groups and days. For example, you can see the subject names of the first group at the second day by ids{1, 2}. You can address a specific entry of the resulting cell array, e.g. ids{1, 2}{3} for the third entry of the first group at the second day.

Usage: – No manual usage –

...

Output Value	Description	Output Type
graphCell	Each entity in *graphCell* contains Edges and Nodes for each group G and each day D	{G×D cell array}
matrixValues	Each entity in *matrixValues* contains the mean connectivity matrix values of each group G and each day D	{G×D cell array}
ids	Each entity in *ids* contains the names of each subject for each group G and each day D	{G×D cell array}

shortestPath(inputFMRI, day2examine, startNode, endNode, groupAverage).m

This function displays the shortest path from one region to another. The shortest path is calculated as an inverse Dijkstra's Algorithm, meaning that this function seeks for the shortest path with the hightest sum of connection weights representing the best strengthened path. The first argument is inputFMRI. Remember that in a new session this parameter has to be created manually again (see mergeFMRIdata_input.m). day2examine refers to the day to analyze (as an integer, position of inputFMRI.days). startNode and endNode correspond to the regions (as strings) whose shortest path should be determined. groupAverage is an optional input argument: You can either choose to display the mean shortest path for all groups (=1, default value) or choose to display the shortest path for each subject of the groups (=0)

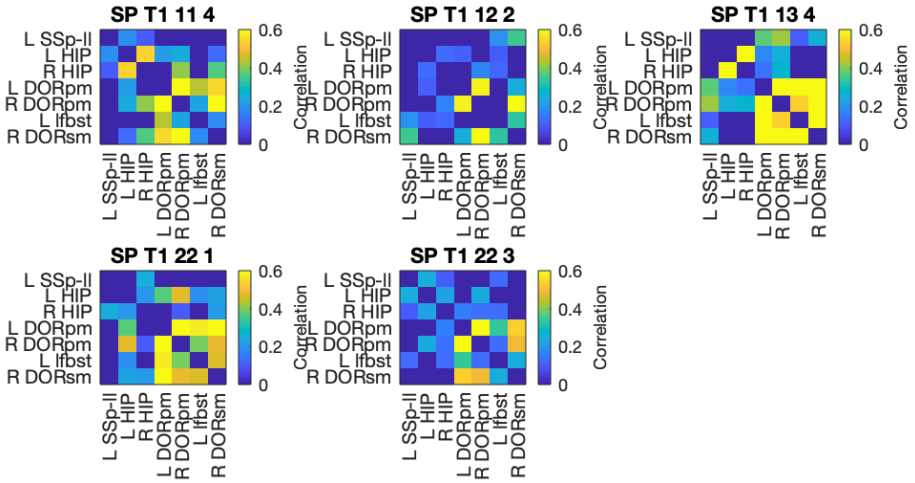
The output is displayed in the Command Window of MATLAB. The output parameter 'Path Length' refers to the distance of the indicated nodes calculated on the basis of the inverted correlation. A lower value corresponds to a stronger connection.

```
Example: shortestPath(inputFMRI, 1, 'R sAMY', 'L SSs')
```

plotCorrelationMatrices.m

This script displays scatter plots of the correlation of selected regions (figure 1) and their mean values and standard deviation (figure 2). Please specify the path to a processed MAT-file and the selected regions in line 8 and 9 of the script. You can also adjust the colour range of the plot in line 10.

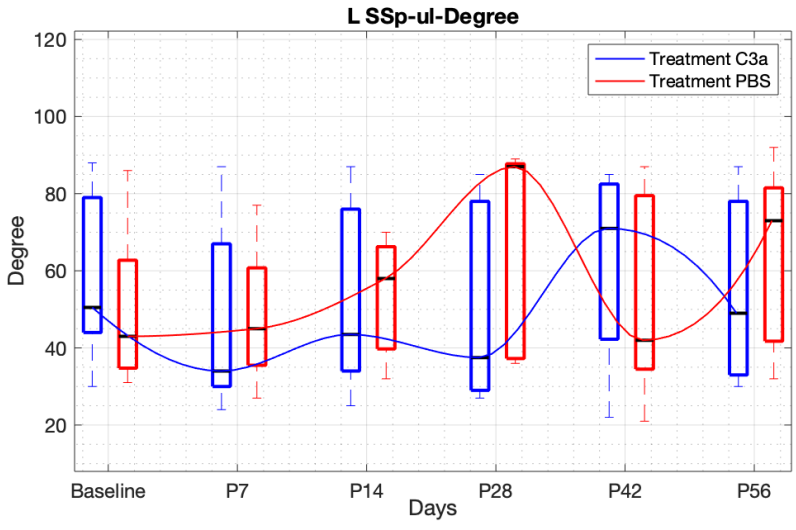
Usage: Fill in the specifications and hit 'Run'.



plotLocalParameter(inputFMRI, graphCell, strNodeName, strParameter).m

This function uses graphCell to calculate the properties ('Degree', 'Strength', 'Eigenvector', 'Betweenness', 'Clustercoefficient', 'Participationcoefficient', 'Efficiency') of the region strNodeName based on the information by inputFMRI. The selected property is displayed for all groups over all days. Remember that in a new session the input parameter inputFMRI and the cell array graphCell have to be created manually again (see mergeFMRIdata_input.m). The selected property is represented by the argument strParameter.

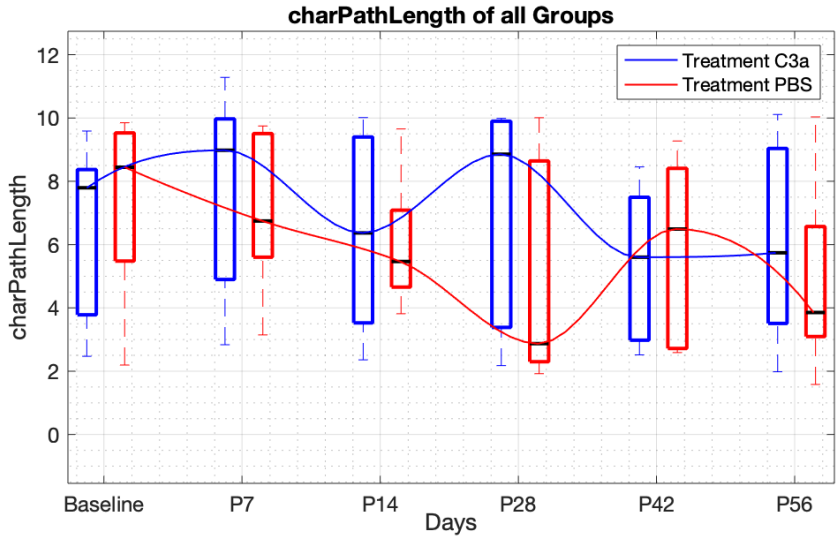
Example: plotLocalParameter(inputFMRI, graphCell, 'L SSp-ul', 'Degree')



plotGlobalParameter(inputFMRI, graphCell, strParameter).m

This function uses inputFMRI and graphCell to display global graph properties of the whole network. The selected parameter ('Density', 'Transitivity', 'Efficiency', 'Assortativity', 'Modularity', 'charPathLength' or 'smallWorldness', 'overallconnectivity') is displayed for all groups over all days. Remember that in a new session the input parameter inputFMRI and the cell array graphCell have to be created manually again (see mergeFMRIdata_input.m). The selected property is represented by the argument strParameter.

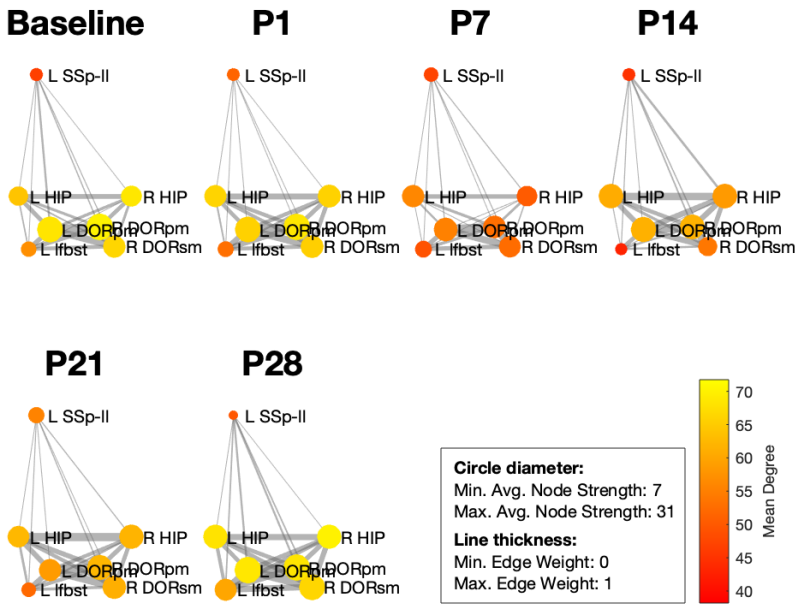
`Example: plotGlobalParameter(inputFMRI, graphCell, 'Transitivity')`



`plotSelectedRegions(inputFMRI, graphCell).m`

This function represents the graph model with regions as nodes and the correlations as a basis for the edge weights. The nodes have the anatomically correct position in the display because the focal points of each region were previously assigned to a node. Each node is color-coded according to its degree or strength. Which regions should be displayed can be selected using the selectedRegions variable in line 11. You can specify further display options in lines 14 and 15. The first argument is the struct inputFMRI. The second argument is the cell array graphCell. Remember that in a new session both parameters have to be created manually again (see mergeFMRIdata_input.m) The plots are called up according to the groups in infoFMRI.groups.

`Example: plotSelectedRegions(inputFMRI, graphCell)`



analyzeLostConnections(inputFMRI, graphCell, selectedRegion, day1, day2).m

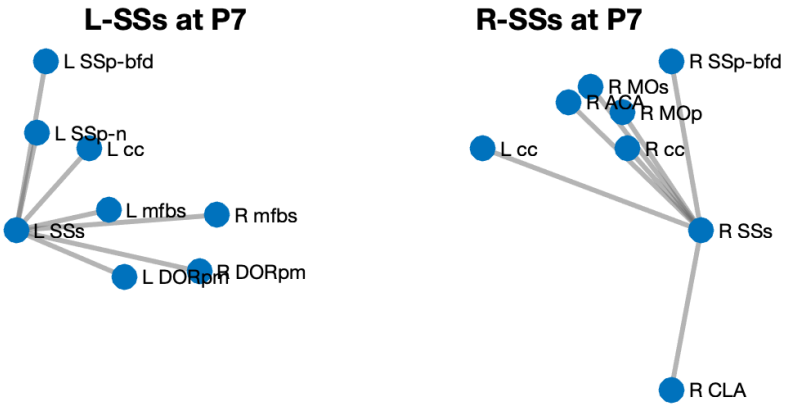
This function finds all connections of a selected region per group that are not present in the second time point but do have connection weights in the first time point. The first two arguments are inputFMRI and graphCell. Remember that in a new session both structs have to be created manually again (see mergeFMRIdata_input.m). The third argument defines the region to examine. The fourth and fifth arguments specify the time points. You can specify a weighting threshold in line 17 to neglect lost connections that only had small weights. In case of rsFMRI the connection weight is built up of correlations (0-1 = range of threshold). The output is displayed in the Command Window of MATLAB for all groups.

```
Example: analyzeLostConnections(inputFMRI, graphCell, 'L SSp-ll', 'Baseline',
'P7')
```

analyzeMostConnected(inputFMRI, graphCell, RegionPF, day, out_path).m

*This function finds the most connected regions of a seed region per group for a selected time point. The first two arguments are inputFMRI and graphCell. Remember that in a new session both structs have to be created manually again (see mergeFMRIdata_input.m). The third argument defines the region to examine. Please enter the region without a specification of the hemisphere (L-, R-). The plot will automatically display both sides. The fourth argument specifies the time point. The last argument is the output path where the table and/or the figures can be saved as a *.csv-file and *.pdf-files. If you keep this argument empty, no table/figure will be exported. You can define the number of the most connected regions in line 12 of the function and further options in line 13 (size of nodes/edges), 17 (export table) and 18 (export figures).*

Example: analyzeMostConnected(inputFMRI, graphCell, 'SSs', 'P7')



analyzeNewConnections(inputFMRI, graphCell, selectedRegion, day1, day2).m

This function finds all connections of a selected region per group that are not present in the first time point but do have connection weights in the second time point. The first two arguments are inputFMRI and graphCell. Remember that in a new session both structs have to be created manually again (see mergeFMRIdata_input.m). The third argument defines the region to examine. The fourth and fifth arguments specify the time points. You can specify a weighting threshold in line 17 to neglect new connections with small weights. In case of rsFMRI the connection weight is built up of correlations (0-1 = range of threshold). The output is displayed in the Command Window of MATLAB for all groups.

Example: analyzeNewConnections(inputFMRI, graphCell, 'L SSp-ll', 'Baseline', 'P7')

analyzeSameMostConnectedBothGroups(inputFMRI, RegionPF, day, out_path).m

*This function finds all regions, which are similar connected in both groups. Regarding these regions only, this function finds the most connected regions of a seed region for a selected time point. The first argument is inputFMRI. Remember that in a new session this struct has to be created manually again (see mergefMRIdata_input.m). The second argument defines the region to examine. The third argument specifies the time point. You can define the number of the most connected regions in line 20 of the function. The last argument is the output path where the table can be saved as a *.csv-file. If you keep this argument empty, no table will be exported.*

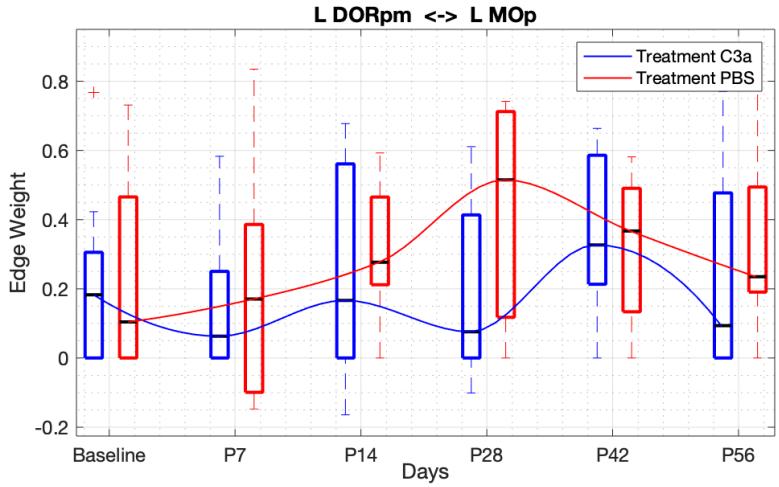
The output is displayed in the Command Window of MATLAB.

Example: analyzeSameMostConnectedBothGroups(inputFMRI, 'SSs', 'P21')

plotConnectionWeight(inputFMRI, graphCell, startP, endP).m

This function represents the change of the edge weight over time for all groups. inputFMRI and graphCell represent the first two input values in order to draw the information from the corresponding study. Remember that in a new session both input parameters have to be created manually again (see mergeFMRI-data_input.m). startP and endP represent the regions whose edge weight should be analyzed. Please pass the regions as strings.

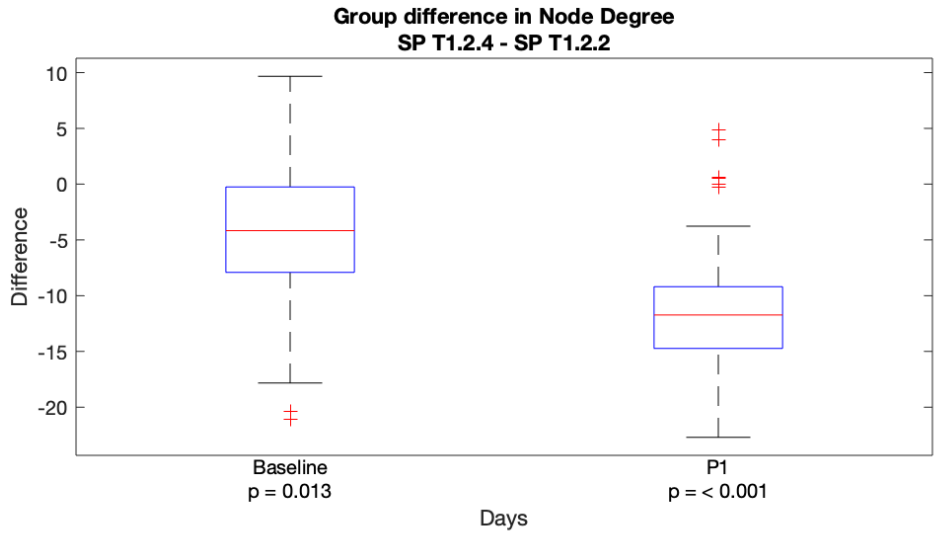
Example: plotConnectionWeight(inputFMRI, graphCell, 'L DORpm', 'L MOp')



compareTotalDegree(inputFMRI, graphCell, day1, day2, out_path).m

*This function collects all degrees of all nodes and compares them with a paired t-test on selected (two) days. The first argument is the struct inputFMRI. The second argument is the cell array graphCell. Remember that in a new session both parameters have to be created manually again (see mergeFMRIdata_input.m). The third and fourth argument represent the selected days as an integer (position of inputFMRI.days in ascending order). In the last argument you may specify an output path where the plot can be saved as a *.pdf-file. If you keep this argument empty, no plot will be exported.*

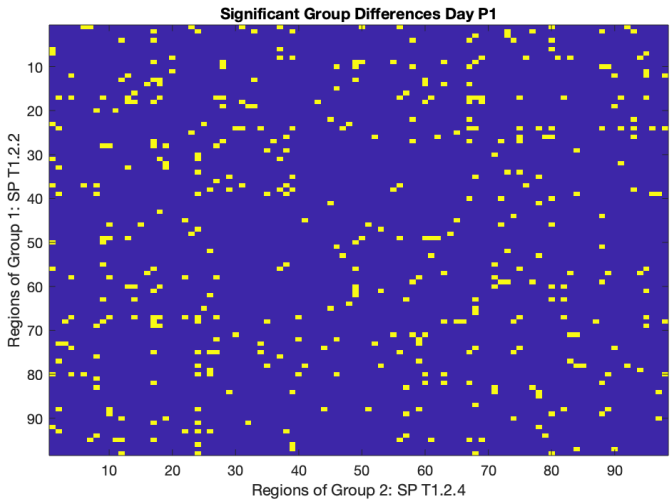
Example: compareTotalDegree(inputFMRI, graphCell, 1, 2)



plotSignificantMatrix(inputFMRI, day, out_path).m

This function checks for differences in connectivity between two groups for a certain day. It tests all connections in the connectivity matrix for all 98 regions and plots a matrix which shows a 1 (yellow pixel) if a significant difference between the groups is detectable (all connections with p-Value < 0.05 are displayed as 1 in the plot). The first argument is the struct inputFMRI. Remember that in a new session this struct has to be created manually again (see mergeFMRIdata_input.m). The second argument is the day to examine. In the last argument you may specify an output path where the plot can be saved as significantConn.pdf. If you keep this argument empty, no plot will be exported. It may help to maximize the window to read

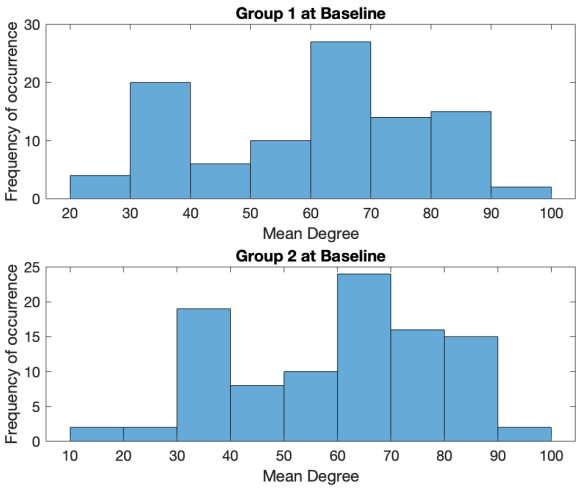
Example: plotSignificantMatrix(inputFMRI, 'P1')



plotDistribution(inputFMRI, graphCell, strParameter, day).m

This function displays the distribution of either the node degree or the node strength for all groups as a histogram for a given day. This way, you can investigate how many regions of the brain have degree/strength values between a certain range. inputFMRI and graphCell represent the first two input values in order to draw the information from the corresponding study. Remember that in a new session both input parameters have to be created manually again (see mergeFMRIdata_input.m). The third argument defines the graph parameter and accepts either 'Degree' or 'Strength'. The fourth parameter specifies the time point (as in inputFMRI.days). Please define the parameter and the time point as strings. You can furthermore display all region's acronyms that have degree/strength values in a specific range if you enter the lower and upper bound in lines 16 and 17 of the function. The output is displayed in the command window of MATLAB.

```
Example: plotDistribution(inputFMRI, graphCell, 'Degree', 'Baseline')
```



4.3.1 Used data structure

You typically want to ask yourself: What kind of graphs do exist in group 1 on the second day? You can investigate graph parameters using graphCell{G, D}. The output of a specific call is divided into two main parameters: Nodes (regions) and Edges (connections).

Nodes have the following properties:

```
Name, XCoord, YCoord, ZCoord, allMatrix, allDegree, allStrength, allEigenvector,
allBetweenness
```

Edges have the following properties:

```
EndNodes, Weight
```

These properties can be read out using dot notation, e.g.:

```
graphCell{1,2}.Nodes.allDegree
graphCell{1,2}.Edges.Weight
```

to display the Node Degree and the Weights of all Edges of the first group on the second time point.

5 References

Aswendt, Markus, Niklas Pallast, Frederique Wieters, Mayan Baues, Mathias Hoehn, and Gereon R Fink. 2020. “Lesion Size- and Location-Dependent Recruitment of Contralesional Thalamus and Motor Cortex Facilitates Recovery after Stroke in Mice.” *Translational Stroke Research*, 1–11. doi:10.1007/s12975-020-00802-3.

Pallast, Niklas, Michael Diedenhofen, Stefan Blaschke, Frederique Wieters, Dirk Wiedermann, Mathias Hoehn, Gereon R Fink, and Markus Aswendt. 2019. “Processing Pipeline for Atlas-Based Imaging Data Analysis of Structural and Functional Mouse Brain MRI (AIDAmri).” *Frontiers in Neuroinformatics* 13: 42. doi:10.3389/fninf.2019.00042.

Pallast, Niklas, Frederique Wieters, Marieke Nill, Gereon R Fink, and Markus Aswendt. 2020. “Graph Theoretical Quantification of White Matter Reorganization after Cortical Stroke in Mice.” *NeuroImage* 217: 116873. doi:10.1016/j.neuroimage.2020.116873.

Rubinov, Mikail, and Olaf Sporns. 2009. “Complex Network Measures of Brain Connectivity: Uses and Interpretations.” *NeuroImage* 52 (3): 1059–69. doi:10.1016/j.neuroimage.2009.10.003.

Wang, Quanxin, Song-Lin Ding, Yang Li, Josh Royall, David Feng, Phil Lesnar, Nile Graddis, et al. 2020. “The Allen Mouse Brain Common Coordinate Framework: A 3D Reference Atlas.” *Cell*. doi:10.1016/j.cell.2020.04.007.

Aswendt, Markus, Niklas Pallast, Frederique Wieters, Mayan Baues, Mathias Hoehn, and Gereon R Fink. 2020. “Lesion Size- and Location-Dependent Recruitment of Contralesional Thalamus and Motor Cortex Facilitates Recovery after Stroke in Mice.” *Translational Stroke Research*, 1–11. doi:10.1007/s12975-020-00802-3.

Pallast, Niklas, Michael Diedenhofen, Stefan Blaschke, Frederique Wieters, Dirk Wiedermann, Mathias Hoehn, Gereon R Fink, and Markus Aswendt. 2019. “Processing Pipeline for Atlas-Based Imaging Data Analysis of Structural and Functional Mouse Brain MRI (AIDAmri).” *Frontiers in Neuroinformatics* 13: 42. doi:10.3389/fninf.2019.00042.

Pallast, Niklas, Frederique Wieters, Marieke Nill, Gereon R. Fink, and Markus Aswendt. 2020. “Graph Theoretical Quantification of White Matter Reorganization after Cortical Stroke in Mice.” *NeuroImage* 217: 116873. doi:10.1016/j.neuroimage.2020.116873.

Rubinov, Mikail, and Olaf Sporns. 2009. “Complex Network Measures of Brain Connectivity: Uses and Interpretations.” *NeuroImage* 52 (3): 1059–69. doi:10.1016/j.neuroimage.2009.10.003.

Wang, Quanxin, Song-Lin Ding, Yang Li, Josh Royall, David Feng, Phil Lesnar, Nile Graddis, et al. 2020. “The Allen Mouse Brain Common Coordinate Framework: A 3D Reference Atlas.” *Cell*. doi:10.1016/j.cell.2020.04.007.

Wijk, Bernadette C. M. van, Cornelis J. Stam, and Andreas Daffertshofer. 2010. “Comparing Brain Networks of Different Size and Connectivity Density Using Graph Theory.” *PLoS ONE* 5 (10): e13701. doi:10.1371/journal.pone.0013701.