ReadMe – Granger Causality Analysis

Content

[1. Overview – What is this file about? 1](#_Toc522211645)

[2. Fix broken AVI Index 1](#_Toc522211646)

[1) Problem description: 1](#_Toc522211647)

[2) How to repair the broken index: 1](#_Toc522211648)

[3) The result: 2](#_Toc522211649)

[3. Convert AVI to MP4 2](#_Toc522211650)

[1) Problem description: 2](#_Toc522211651)

[2) How to convert: 2](#_Toc522211652)

[4. Sanity checks 3](#_Toc522211653)

# Overview – What is this file about?

This user guide contemplates the code that was used for granger causality analysis of the widefield imaging data of the Multisensory project. It gives detailed instructions on how to use my code in conjunction with Anil Seth’s *Multivariate Granger Causality MATLAB Toolbox* (MVGC), it elaborates on pitfalls when using the toolbox and challenges of our particular project.

## Conceptual overview

Granger causality (GC) is a statistical inference method that can be used for directed functional connectivity analysis, i.e. it can help to infer whether a certain brain region may have influenced another during a task. Specifically, Granger causality is identical to Transfer-Entropy in the case of truly Gaussian variables. Generally, if brain region *X* “G-causes” brain region *Y,* it means that “**the past of *X* contains information that helps to predict the future of *Y* above the information already in the past of *Y* itself**” (Seth, 2015, Granger Causality Analysis in Neuroscience and Neuroimaging). We use GC to investigate whether functional connectivity differs in behaving mice between 1) different external stimuli (visual, somatosensory, both, none), 2) different task types (visual and somatosensory task) and 3) different responses to the stimuli.

# Granger Causality computation

## Data organization

The wide-field calcium imaging data, initially recorded in *dcimg* file format, needs to be preprocessed and registered before GC analysis comes into play. During preprocessing a series of 200 samples (sampling rate 20 Hz and sample 31 was the onset of the stimulus) of images of size *256 x 256* were created for every trial and saved in a separate *mat* file (trial 1 🡪 *dFF\_t1.mat*). The data was stored on the hard disks *F* and *I* in the structure:

*hard\_disk\_id:/data/registered/date\_id/mouse\_id/session\_id/trial\_id.mat*

For the future, the usage of the *F* drive should be avoided since it is very slow (copying 600GB to the server (*W*) took around a week!).

## Data preparation

As main ingredient, the MVGC toolbox requires a matrix of shape *num\_regions x num\_observations x num\_trials*. Instead of running GC analysis on a pixel-wise basis (*num\_regions = 256\*256)*, 33 regions of interests (ROI) were defined, inspired by the Allen Mouse Common Coordinate Framework (CCF). By task design: *num\_observations* = 200. *num\_trials* = the total number of trials fulfilling a certain criterion. Tested criteria were:

1. External stimulus
   1. visual\_stim In this trial, a visual stimulus was presented on a screen
   2. sensory\_stim In this trial, a somatosensory stimulus was applied to the whiskers
   3. multi\_stim Both, visual and sensory stimuli were applied
   4. no\_stim No stimulus was applied
2. Task type
   1. visual\_task The mouse’s task was to lick in response to visual stimuli only
   2. sensory\_task The mouse’s task was to lick in response to somatosensory stimuli only
3. Task-response-alignments
   1. hit The mouse licked and this was the response expected by task design
   2. miss The mouse should have licked but did not do so
   3. false\_alarm The mouse licked although it should not have done so
   4. correct\_rejection The mouse correctly did not lick

More criteria such as high/low performance sessions could be tested in the future. The assembly of these data matrices is achieved by the class **gc\_gathering** which receives the ID of a mouse (data for the following mice has been recorded: 1110r, 1113rr, 5212r, 5627rr). It loops over all sessions of this mouse, opens the *.mat* file of each trial (widefield time series) and warps it to the standard Allen atlas. It then loops over all ROI, computes a single value for each frame (by averaging all pixels in the ROI) and saves the resulting matrix as a struct (next to a few meta variables such as the task type, the behavior and the stimulus associated to each trial).

## Streamlining GC analysis (hyperparameter setting)

Next, the script **gca\_script** can be used to automatize the GC analysis. In its first section, the user can specify meta variables such as the sampling rate of the data and which conditions to analyse for which mouse. The user can also set *momax,* i.e. the maximum model order that could be applied if the multivariate autoregressive (MVAR) model is fitted during the analysis. For example, if the model order is *M*, the MVAR would look like:

Where *X* and *Y* are time series representing 2 ROI, *A* is a *2 x 2 x M* matrix with the learned GC coefficients and and ­ are the residuals respectively. The actual model order is computed in a regularization procedure based on BIC/AIC. should be set carefully such that it corresponds to the maximal number of frames which a signal may need to travel from one brain area to another. For this project we set (i.e. 300ms since sampling rate is 20 Hz). The user can also set time\_onset, a parameter that controls that all frames [time\_onset, 200 are used for the analysis. Frame 31 corresponds to the stimulus onset and frame 71 to the start of the response phase. Hence, e.g. frames < 31 cannot be predominantly driven by the task type so it may be beneficial to set time\_onset = 31 if the investigated condition is “task type” or ”external stimulus” (and 71 if “task-response-alignment”).

The next section of **gca\_script** retrieves the data matrices assembled by **gc\_gathering** and calls the function **gc\_analysis** which computes the pairwise Granger causality of all ROI and saves the results in a separate folder.

## The analysis

Please be careful when changing **gc\_analysis**, it may result in errors or even wrong estimations from the MVGC toolbox (the current values are verified to work). Frequent problems with the MVGC toolbox are extensively documented in the comments of the respective functions.

## Model order estimation (information criteria)

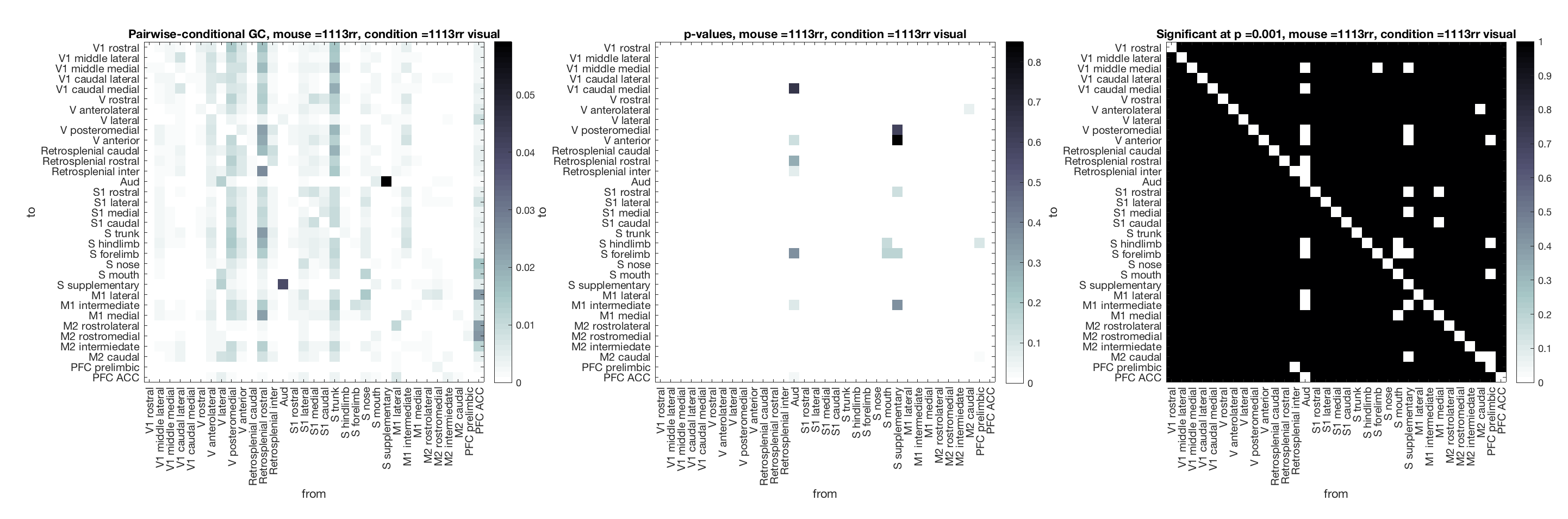
First, the suitable model order is estimated from the data by computing the Akaike Information Criterion (AIC) as a tradeoff between explained variance and model complexity. For all observed cases in the current project . The MATLAB function for this is **tsdata\_to\_infocrit.**

A frequently occurring error in practice is that the ROI covariance matrix is not positive definite (but only positive semi-definite). This happens if the rank of the covariance matrix is not full which is usually caused by a missing ROI in the original data matrix (i.e. a zero row). Therefore **gca\_script** performs a routine excluding unused ROIs.

## Model estimation

Next, the data, the selected model order and the regression mode (OLS) are given to the function **tsdata\_to\_var** which fits the VAR model by estimating the coefficient matrix *A* of shape *num\_roi x num\_roi x m*. It also returns the covariance residuals. These 2 matrices are then fed to **var\_to\_autocov** which computes the autocovariance sequence. In the next step, the pairwise Granger causalities between all ROI are computed from the autocovariance matrix by means of **autcov\_to\_pwcgc**. This call may take up to an hour depending on the size of the autocovariance sequence and the machine.

In the final step, the resulting pairwise GC matrix (shape *num\_rois x num\_rois*) is plotted, next to a matrix with the *p*-values of a *F*-test with the null hypothesis that the pairwise GC coefficient is 0 and a significance matrix with *p‑*values thresholded (including a multiple comparison correction, per default the most conservative, i.e. Bonferroni correction)



Author: Jannis Born, jborn@ethz.ch