NeuroMiner Short Tutorial Workbook

NeuroMiner Tutorial Part 1:

Topics: introduction to NeuroMiner and data entry

The goal of this part is to get you familiar with NeuroMiner and begin an analysis using the COBRE dataset of individuals with schizophrenia and controls. At the end of this tutorial you will be familiar with the NeuroMiner menu system, be able to enter your data into NeuroMiner with different formats (e.g., spreadsheet and neuroimaging), and be able to run basic classification analysis and explore results. You should be able to answer to the following questions:

- What is NeuroMiner?
- Which data formats are accepted in NM? How do I enter data to NM?
- How do I set up a basic classification analysis in NeuroMiner?

Contents

Exercise 1: NeuroMiner and the NM structure

NeuroMiner was developed by Prof. Nikolaos Koutsouleris (LMU). Using a light-weight and interactive text-based menu system, NeuroMiner allows the user to: load their data easily (e.g., using spreadsheets, NifTi images, or SPM structures) build a variety of cross-validation frameworks for classification and regression problems that have become a gold standard in the field (e.g., repeated nested cross-validation, leave-site-out cross-validation); apply a range of preprocessing strategies (e.g., scaling, filtering, dimensionality reduction, etc.); choose and combine cutting-edge supervised algorithms (e.g., support vector machine, elastic net, random forest, etc.); apply feature selection procedures (e.g., wrappers), data fusion techniques, and stacked generalization; apply learned models to new data (external validation).

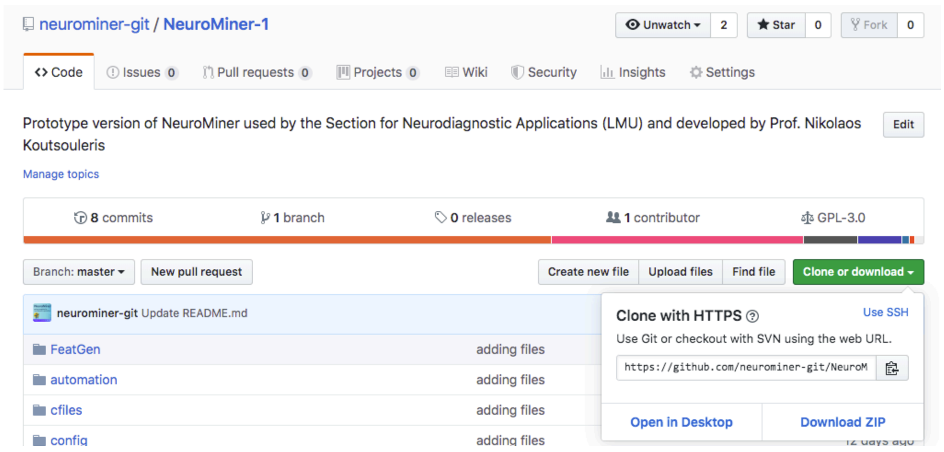
Things To Keep in Mind:

- NeuroMiner works exclusively from the MATLAB command line and it has a text-based interface. This means that all analyses are implemented by selecting options from text menus and entering parameters when asked to do so.
- For comments, questions, or bug reports send an Email to: email.neurominer@gmail.com.
- Please take note of the warnings and errors
- Please read the manual

If you want to use it outside of the tutorial then these questions are important:

- Do you have MATLAB2018a or newer installed?
- If you want to use MRI imaging, do you have SPM8 or SPM12 installed?
- Do you have at least 4GB of RAM on your computer?

Do you have NeuroMiner on your computer? If not, please download it from https://github.com/neurominer-git/NeuroMiner-1 . Go to „Clone or Download“ and then select „Download ZIP“



Prototype version of NeuroMiner used by the Section for Neurodiagnostic Applications (LMU) and developed by Prof. Nikolaos Koutsouleris

neurominer-git Update README.md

FeatGen	adding files
automation	adding files
cfiles	adding files
config	adding files

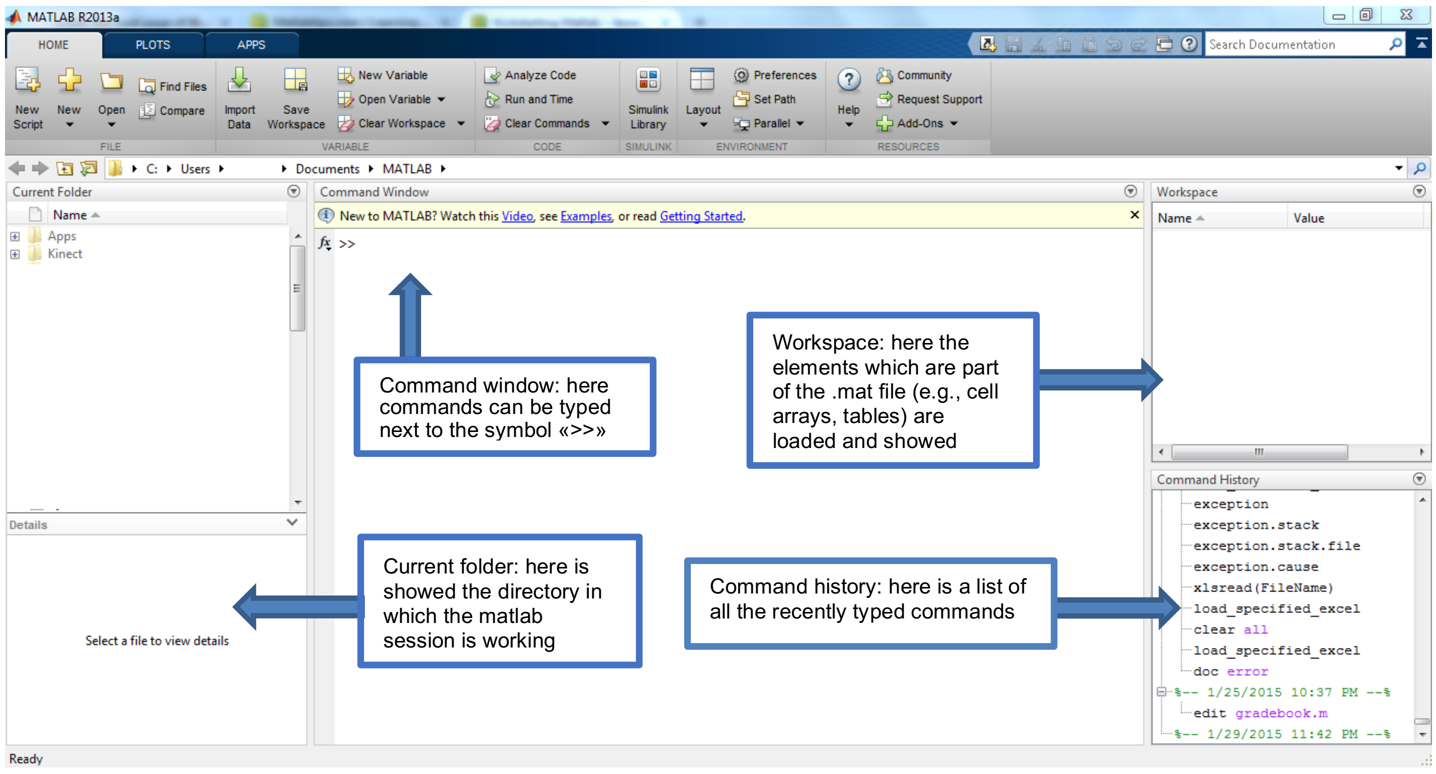
Use Git or checkout with SVN using the web URL.

`https://github.com/neurominer-git/NeuroM`

The goal of this exercise to introduce you to the MATLAB display, how to load NeuroMiner, and what the NM structure is.

Basic Points about MATLAB

- A .mat file is a MATLAB specific file type that contains data used in a MATLAB environment. NeuroMiner needs to access these types of data and stores all results and settings using this format.
- The default MATLAB display shows a few main areas that can be important for a NeuroMiner analysis. On the following page you will find a picture highlighting the main parts of the MATLAB GUI



New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
>>
```

Command window: here commands can be typed next to the symbol «>>»

Workspace: here the elements which are part of the .mat file (e.g., cell arrays, tables) are loaded and showed

Current folder: here is showed the directory in which the matlab session is working

Command history: here is a list of all the recently typed commands

```
exception
exception.stack
exception.stack.file
exception.cause
xlsread(FileName)
load_specified_excel
clear all
load_specified_excel
doc error
%-- 1/25/2015 10:37 PM --%
edit gradebook.m
%-- 1/29/2015 11:42 PM --%
```

Step 1: Adding the NeuroMiner Path

Within the MATLAB environment, software has to be added to the search path in order for it to work properly. You can add NeuroMiner permanently to your path by:

1. Going to "File->Set Path" from within MATLAB or type "pathtool" at the MATLAB prompt.
2. Using the "Add" button to add your NM folder to the MATLAB path.
3. Clicking "Save" so that this path is used in future MATLAB sessions.

You can also add the path from the command line by typing:

```
>> addpath('/YOUR_PATH_TO_NEUROMINER')
```

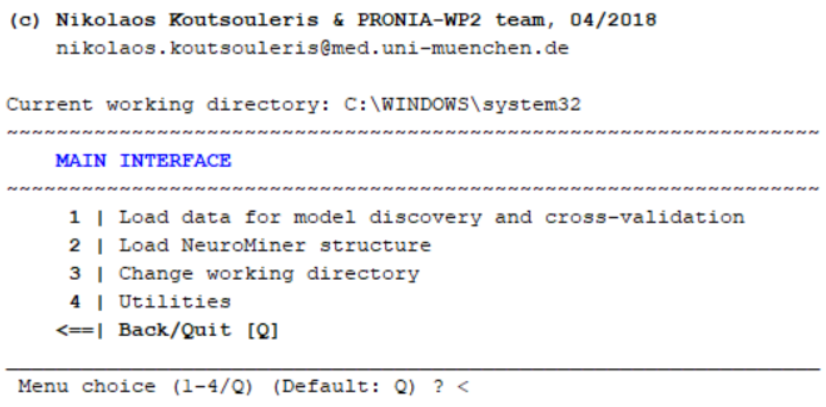
Where "YOUR_PATH_TO_NEUROMINER" is the path to where you downloaded NeuroMiner (e.g., '/Users/Dom/Applications/NeuroMiner').

Step 2: Opening NeuroMiner

Open NeuroMiner by typing

```
>>nm
```

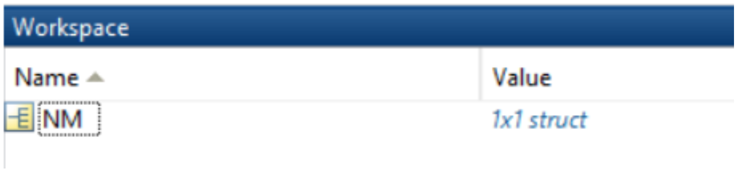
If this is the first time you have loaded NeuroMiner then you will need to also direct it to some important paths for SPM12 and FreeSurfer using the pop-up menus. These programs are not packaged with NeuroMiner, but are used to process NIFTI or FreeSurfer surface data. Then you will see the following picture:



```
 (c) Nikolaos Koutsouleris & PRONIA-WP2 team, 04/2018
     nikolaos.koutsouleris@med.uni-muenchen.de

Current working directory: C:\WINDOWS\system32
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MAIN INTERFACE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         1 | Load data for model discovery and cross-validation
         2 | Load NeuroMiner structure
         3 | Change working directory
         4 | Utilities
       <==| Back/Quit [Q]
_________________________________________________________________
 Menu choice (1-4/Q) (Default: Q) ? <
```

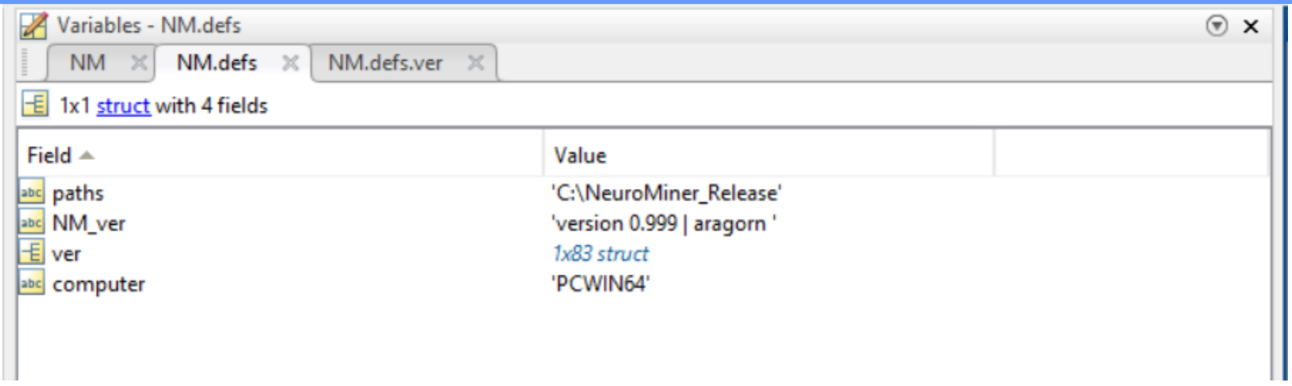
Step 3: understanding the NM structure

Whenever you start NeuroMiner, it will create a MATLAB structure called "NM" in the MATLAB workspace (visible after you have exited NeuroMiner).



Name	Value
NM	1x1 struct

The "NM" structure is the primary variable that NeuroMiner uses to everything about your analyses, including data, settings, analyses, and results. After the first creation of the structure when you start NM, the structure will be updated with all the data and settings that you will enter through the menu system.


Variables - NM.defs

NM | NM.defs | NM.defs.ver

1x1 struct with 4 fields

Field	Value
paths	'C:\NeuroMiner_Release'
NM_ver	'version 0.999 | aragorn '
ver	*1x83 struct*
computer	'PCWIN64'

If you were to exit NeuroMiner and then enter it again without deleting the NM structure, it will automatically find it in the workspace and all your settings will be retained. However, if you close MATLAB then you will lose the settings (i.e., the NM variable will be lost from the workspace) and so it is recommended to periodically save the NM structure.

Questions:

What is a MATLAB structure?

What is the NM structure? What does the NM.defs variable contain?

What is stored in the NM structure?

Is the NM structure saved automatically?

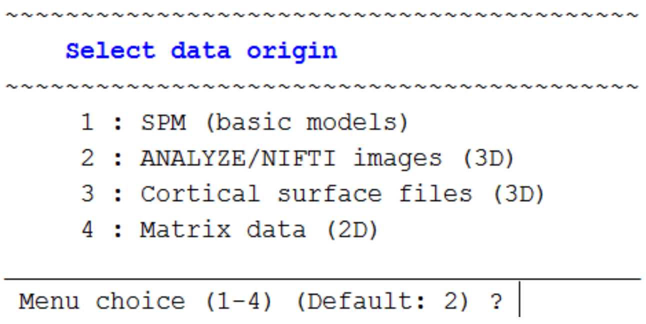
Exercise 2: data entry with spreadsheet

In this section, we will show how to import data stored in spreadsheet. First, exit NeuroMiner and then clear the current NM structure by typing

>> clear

Then, enter again NeuroMiner (>>nm) and choose:

>>**1: load data for model discovery and cross-validation**. From the new menu, then type >> **2: Select data input format**. The following selection will appear :



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Select data origin
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     1 : SPM (basic models)
     2 : ANALYZE/NIFTI images (3D)
     3 : Cortical surface files (3D)
     4 : Matrix data (2D)

 Menu choice (1-4) (Default: 2) ?
```

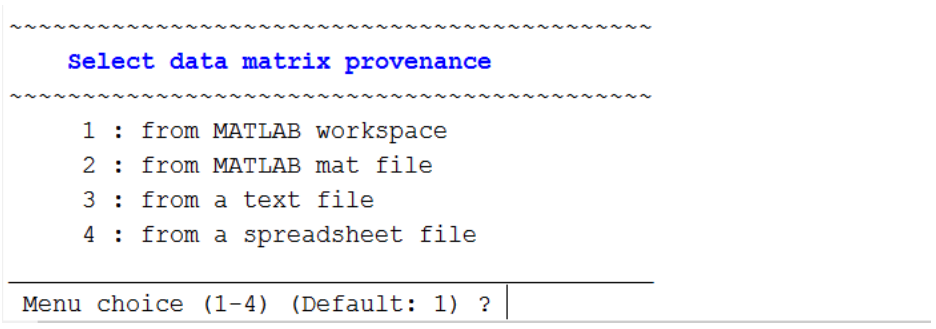
As you can see, a user can enter data coming from the SPM toolbox, NIFTI images, cortical surface files from Freesurfer, or matrix data. We want to select option "4: Matrix data (2D)". Matrix data is any kind of data that is stored in a 2 dimensional (i.e., n x m) matrix. These matrices can be of any size and can come from any source. A simple source might be from a clinical spreadsheet (e.g., in excel) or database where the rows represent cases and the columns are psychological, psychiatric, or cognitive variables. However, any 2D matrix data can be entered—e.g., genetics, functional imaging, structural imaging, physiological readouts, or any other matrix. For example, a common matrix from imaging could be a vectorised connectivity matrix from a resting-state study. Thus, this option to enter matrix data is the most flexible of all options. Press

>> 4 : Matrix data (2D)

You will now be redirected to the NM data workspace manager. Please note how the workspace manager has now changed and it includes more options than before. This reflects the **adaptive menu system of NeuroMiner where the options in each menu will change based on your previous input**.

Matrix data can be entered in a variety of different ways and these can be chosen by selecting:

>> 3: define data provenance



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Select data matrix provenance
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     1 : from MATLAB workspace
     2 : from MATLAB mat file
     3 : from a text file
     4 : from a spreadsheet file

_____________________________________________
 Menu choice (1-4) (Default: 1) ?
```

Then type:

>> 4: from a spreadsheet file

This will then display a graphical user interface (GUI) where you can select the data. In this case we want to select the NM_cobre_cogneuro.xls in the …/DataTutorial/DataImport/Spreadsheet directory. You will then be redirected to the NM data workspace manager where you must define different elements of the spreadsheet, including the group and case identifiers.

To define the groups, select the option to "**4: specify column header containing the label data**". Then specify the name of the column as "group"; i.e., you need to type:

>> 4

>> group

Now **"5: define the column headers containing the case IDs"** and then type **'ID'** and press enter; i.e., you need to type:

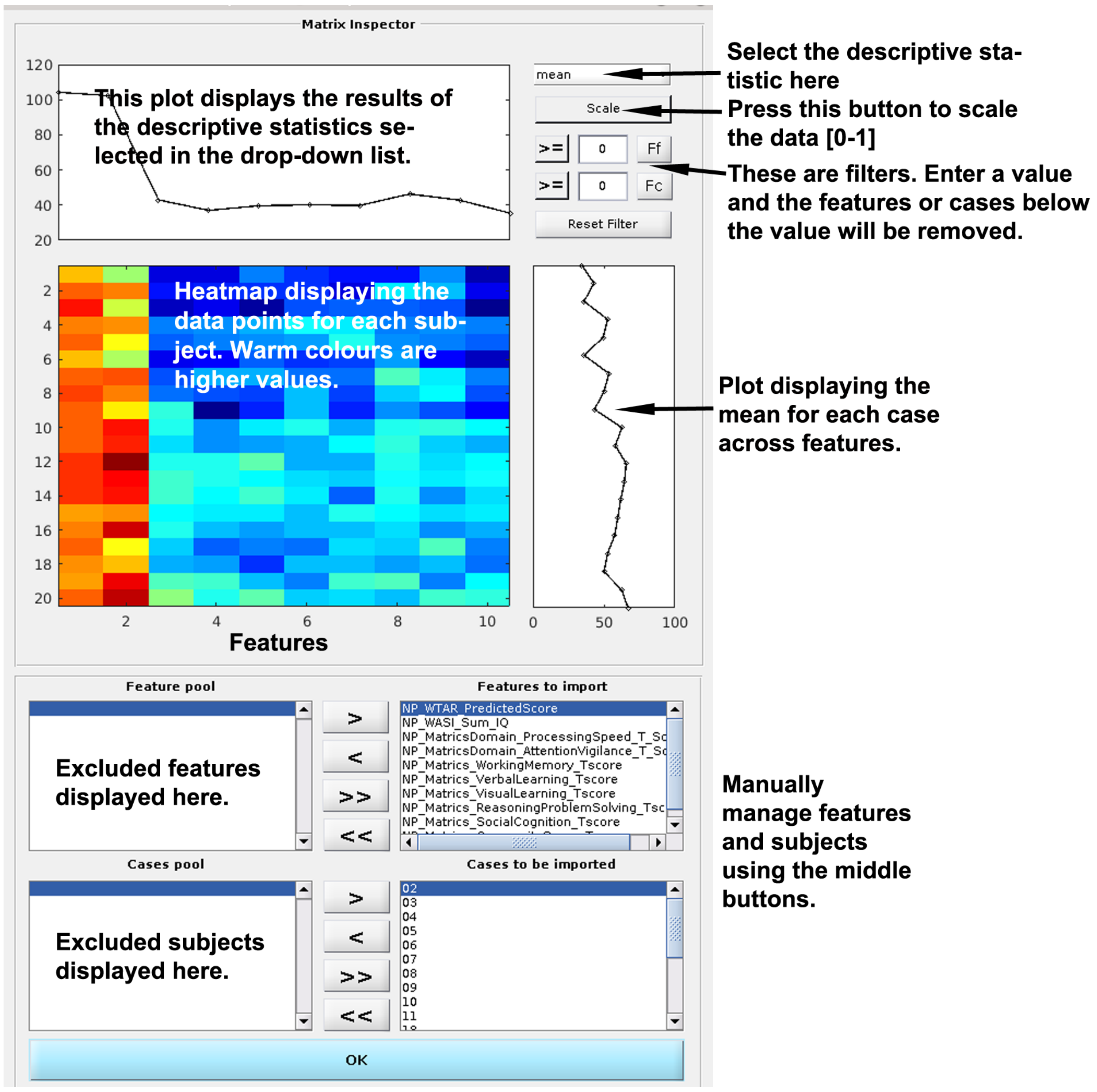
>> 5

>> ID

The data are now entered and you can inspect the features that will be imported. The menu should also change to include the option to "inspect matrix data and select features for import".

Inspecting Data in NeuroMiner

It is critical to understand your data before you do any analyses. This can be done in standard ways (e.g., descriptive statistics) outside of NeuroMiner. However, NeuroMiner also has a data inspector that you can use to understand basic aspects of your data. To see this, select "**6: Inspect matrix data and select features for import**" and the following toolbox GUI will be displayed.



This plot displays the results of the descriptive statistics selected in the drop-down list.

Heatmap displaying the data points for each subject. Warm colours are higher values.

Select the descriptive statistic here

Press this button to scale the data [0-1]

These are filters. Enter a value and the features or cases below the value will be removed.

Plot displaying the mean for each case across features.

Excluded features displayed here.

Excluded subjects displayed here.

Manually manage features and subjects using the middle buttons.

Features to import:

NP_WTAR_PredictedScore
NP_WASI_Sum_IQ
NP_MatricsDomain_ProcessingSpeed_T_Sc
NP_MatricsDomain_AttentionVigilance_T_Sc
NP_Matrics_WorkingMemory_Tscore
NP_Matrics_VerbalLearning_Tscore
NP_Matrics_VisualLearning_Tscore
NP_Matrics_ReasoningProblemSolving_Tsc
NP_Matrics_SocialCognition_Tscore

Cases to be imported:

02
03
04
05
06
07
08
09
10
11

At the top-right of the window, descriptive statistics can be displayed by choosing an option from the drop-down list: the mean, median, variance, interquartile range, %(Nan) (i.e., ”Not a Number” missing values), kurtosis, or skewness. This selection will change the following display images:

- At the top of the inspector, the values (i.e., mean, median, variance, etc.) for each feature **across individuals** are displayed as a line plot (e.g., the mean for the feature)
- In the middle of the inspector a heatmap is shown (cases x features), which shows the actual score for each individual in the sample. Hot colours are high values and cool colours are low values. The matrix is scaled across features.
- To the right of the heatmap, there is another line plot displaying the values for each individual **across features**.

Because features are often not in the same scale, directly below the drop-down box there is an option to “Scale”. This will scale each feature to between 0 to 1. This means that the features will be more comparable, but you have to consider whether it is valid to do this**.** Keep in-mind that this is only a display and the values won’t change in the actual data that will be used for analyses.

It can happen that there are features with unrealistically high values. We recommend to do all your data cleaning outside of NeuroMiner and make sure that your data is valid, but there may be a reason to filter variables based on some kind of numeric range. For example, you may want to filter out all cases with values above 1 and this can be done using the filter boxes below the scaling option. You can then enter a filter to exclude either F(Feats) or F(Cases) based on a value. When a feature filter is used, a red line will appear that signifies the filter value and features above this value will be excluded and moved to the ”Feature pool” at the bottom left of the screen.

You can also manually add or subtract features using the arrows (<) between the feature pool and the features to import. Once features have been understood, filtered, or removed, press OK and return to the main menu.

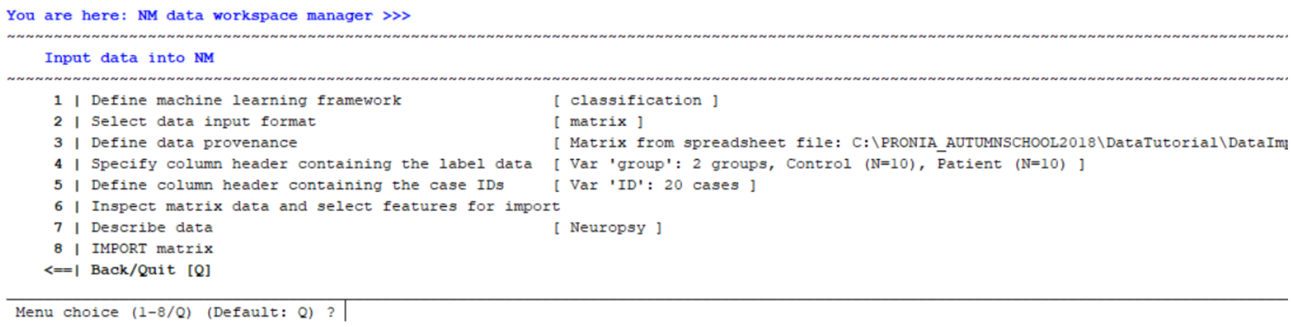
Make sure that you spend some time with the inspector. Then once you are happy that you have an idea of the data press the “ok” button at the bottom of the figure in the middle.

You will be redirected to the NM data workspace manager where you then need to “**7: Describe your data**” and type:

>> cobre_cog_neuro

This is a simple name that will be stored in the NM structure and is useful when you have multiple types of data. An important aspect of NeuroMiner is that multiple modalities can be entered and later combined (i.e., fused and stacked).

The NeuroMiner interface should look similar to the following:



```
You are here: NM data workspace manager >>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Input data into NM
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     1 | Define machine learning framework               [ classification ]
     2 | Select data input format                        [ matrix ]
     3 | Define data provenance                          [ Matrix from spreadsheet file: C:\PRONIA_AUTUMNSCHOOL2018\DataTutorial\DataIm
     4 | Specify column header containing the label data [ Var 'group': 2 groups, Control (N=10), Patient (N=10) ]
     5 | Define column header containing the case IDs    [ Var 'ID': 20 cases ]
     6 | Inspect matrix data and select features for import
     7 | Describe data                                   [ Neuropsy ]
     8 | IMPORT matrix
    <==| Back/Quit [Q]

 Menu choice (1-8/Q) (Default: Q) ?
```

If this is the case, select the option to "**8: IMPORT matrix**". Then you will be redirected to NeuroMiner with the following text in MATLAB shown in the image below. At this point, we can add another modality to the NM workspace, modify it, delete it, delete all data, add covariates, or finish data import for discovery and cross-validation analysis. Now we want to control for covariates, so we will select:

5: Add covariate(s) to NM workspace.

There are also a select number of confounding variables stored in a matlab matrix called Covars.mat that are usually used in imaging experiments (in our case, age and gender).

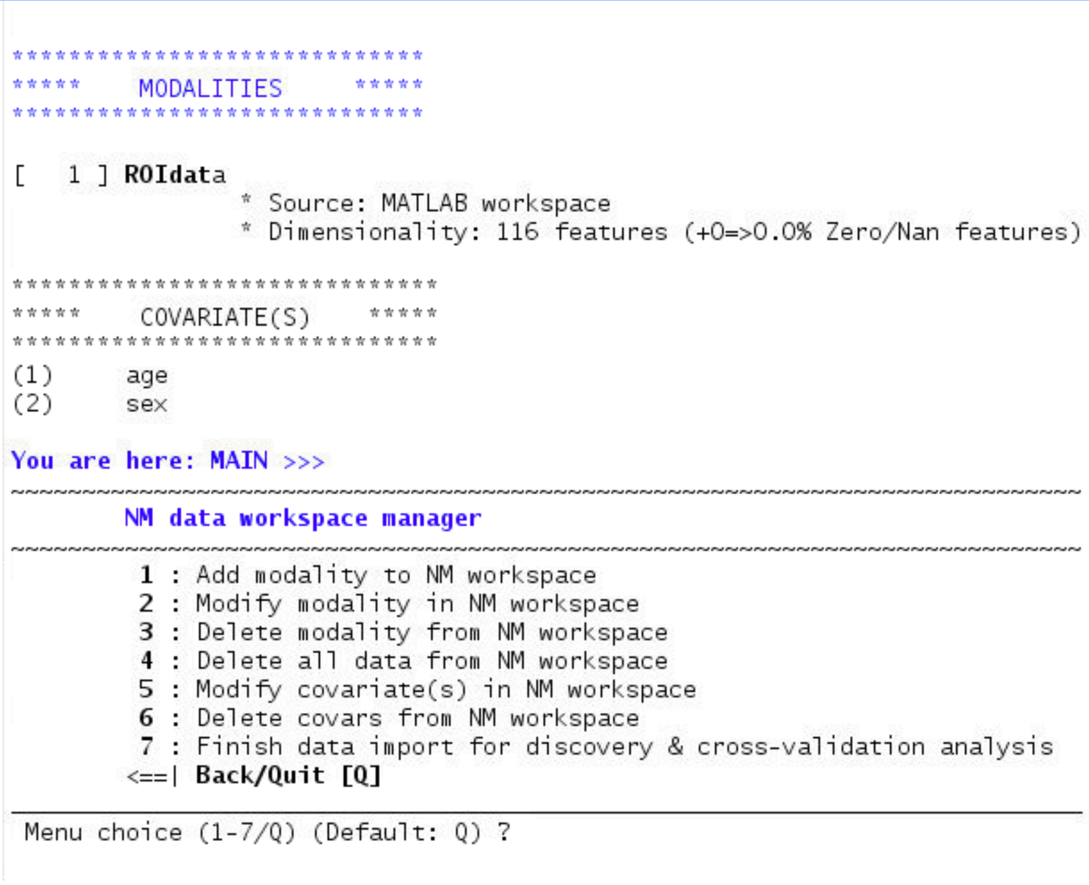
After pressing 5, we will type:

>> Covars

>> v

>> CovarsNames

You should now see the following screen:

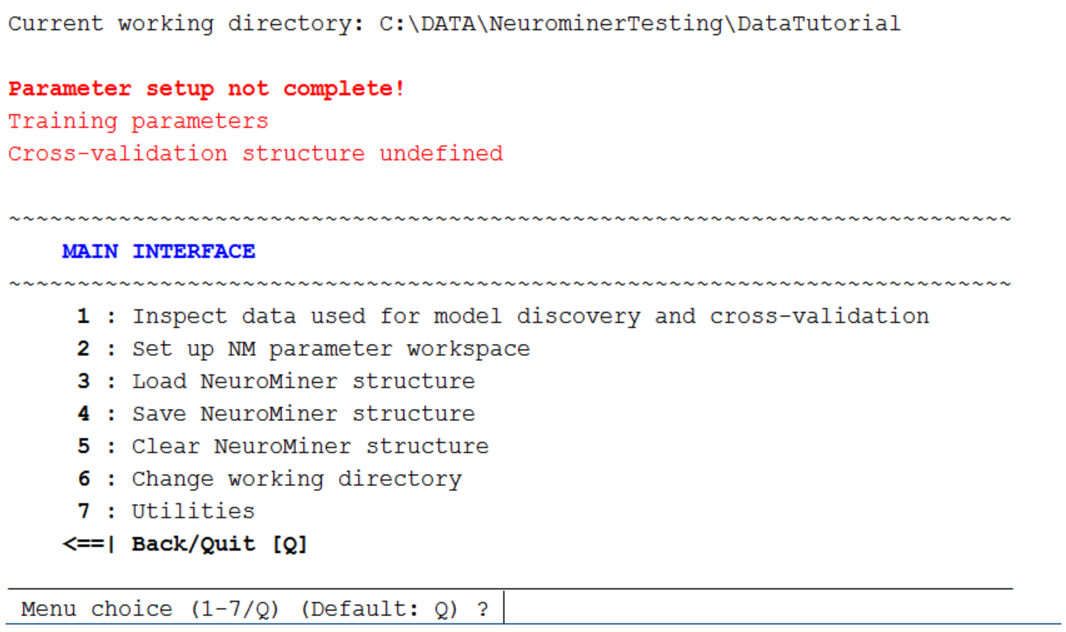


```
*******************************
*****    MODALITIES      *****
*******************************

[   1 ] ROIdata
                * Source: MATLAB workspace
                * Dimensionality: 116 features (+0=>0.0% Zero/Nan features)

********************************
*****    COVARIATE(S)    *****
********************************
(1)     age
(2)     sex

You are here: MAIN >>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        NM data workspace manager
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          1 : Add modality to NM workspace
          2 : Modify modality in NM workspace
          3 : Delete modality from NM workspace
          4 : Delete all data from NM workspace
          5 : Modify covariate(s) in NM workspace
          6 : Delete covars from NM workspace
          7 : Finish data import for discovery & cross-validation analysis
        <==| Back/Quit [Q]
_______________________________________________________________________________
 Menu choice (1-7/Q) (Default: Q) ?
```

The final step is to press:

>> 7: Finish data import for discovery & cross-validation analysis.

Once this is completed, you will be directed to the MAIN MENU of NeuroMiner, which should look like this:



```
Current working directory: C:\DATA\NeurominerTesting\DataTutorial

Parameter setup not complete!
Training parameters
Cross-validation structure undefined

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MAIN INTERFACE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        1 : Inspect data used for model discovery and cross-validation
        2 : Set up NM parameter workspace
        3 : Load NeuroMiner structure
        4 : Save NeuroMiner structure
        5 : Clear NeuroMiner structure
        6 : Change working directory
        7 : Utilities
      <==| Back/Quit [Q]
_________________________________________________________________________
 Menu choice (1-7/Q) (Default: Q) ?
```

This menu includes fundamental functions that are required to manage a NeuroMiner analysis. It adapts depending on the input. At the moment, the first thing we want to do is to

4: Save NeuroMiner structure

A pop-up should appear and you can create a folder entitled “cog_neuro” and then save the NM structure in this as “NM_struct”. This means that you will now have a copy of the NM structure saved for future use.

<u>Questions</u>

What matrix data types can be entered into NeuroMiner?

Why is it important to inspect the data prior to a machine learning analysis? What should we look for when we are doing this? What variables should we use?

Can multiple data types be entered into NeuroMiner? Why would we want to do this?

NeuroMiner Tutorial Part 2

Topics: Nested CV, preprocessing, algorithms, and hyperparameters

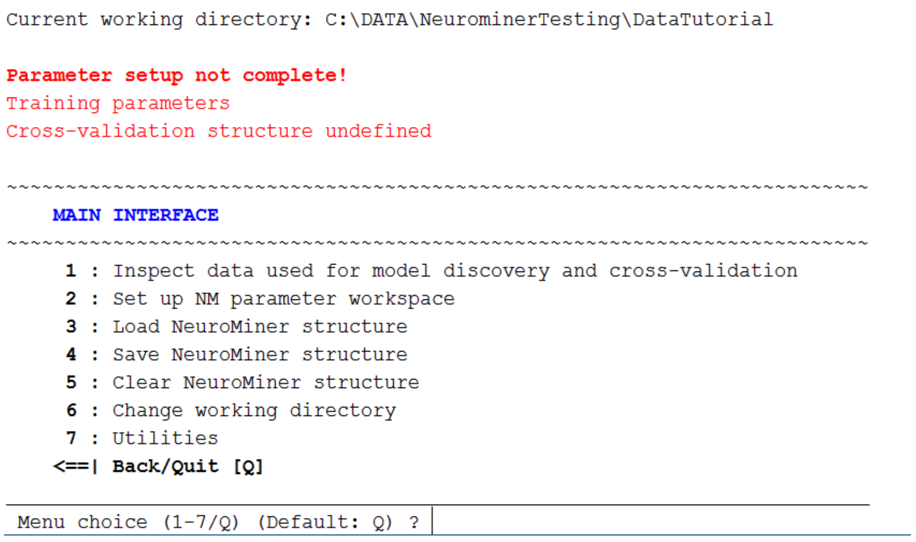
In this tutorial we will aim to classify individuals as either having schizophrenia or not from the image data that we have entered. In doing so, we will develop an understanding of repeated, nested, cross-validation, data preprocessing, algorithm selection, and hyperparameter optimisation. At the end of the tutorial, you should have a deeper appreciation of fundamental machine learning techniques and an appreciation of how to apply them using NeuroMiner.

Please note that we will guide the class through this exercise. The explanations and descriptions below are provided so that you can repeat this exercise later using the data that we have uploaded to www.pronia.eu/neurominer .

Contents

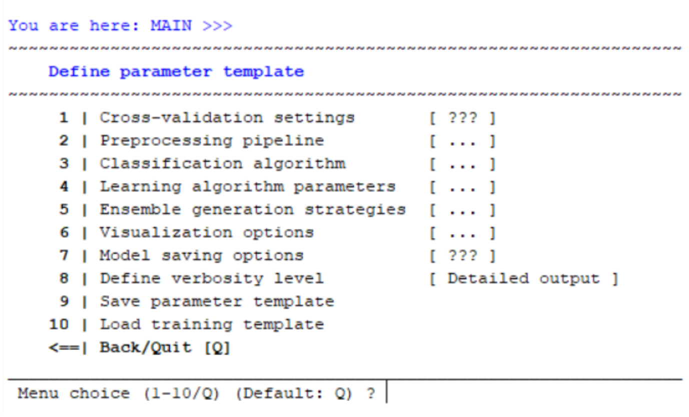
Exercise 1: Repeated, nested cross-validation

In this exercise, we will establish a cross-validation structure within NeuroMiner and discuss the importance of this critical step.



```
Current working directory: C:\DATA\NeurominerTesting\DataTutorial

Parameter setup not complete!
Training parameters
Cross-validation structure undefined

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MAIN INTERFACE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     1 : Inspect data used for model discovery and cross-validation
     2 : Set up NM parameter workspace
     3 : Load NeuroMiner structure
     4 : Save NeuroMiner structure
     5 : Clear NeuroMiner structure
     6 : Change working directory
     7 : Utilities
    <==| Back/Quit [Q]

 Menu choice (1-7/Q) (Default: Q) ?
```

After data is entered, you should see the display above. As you can see from the red text “Parameter setup is not complete!”. This means that NeuroMiner needs some basic parameter settings in order to conduct a machine learning analysis. We are going to set up a pooled cross validation design. Select the option to “2: set up NM parameter workspace”.



```
You are here: MAIN >>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Define parameter template
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     1 | Cross-validation settings        [ ??? ]
     2 | Preprocessing pipeline           [ ... ]
     3 | Classification algorithm         [ ... ]
     4 | Learning algorithm parameters    [ ... ]
     5 | Ensemble generation strategies   [ ... ]
     6 | Visualization options            [ ... ]
     7 | Model saving options             [ ??? ]
     8 | Define verbosity level           [ Detailed output ]
     9 | Save parameter template
    10 | Load training template
   <==| Back/Quit [Q]

 Menu choice (1-10/Q) (Default: Q) ?
```

NM has default options but there are some things that need to be defined before it can be run that are indicated by question marks “???”. First, we need to set-up the cross-validation settings and then we will set-up the preprocessing as well.

Understanding Repeated, Nested Cross-Validation

A central aspect of NeuroMiner is the ability to use nested cross-validation. This involves the separation of two cross-validation schemes: an inner cross-validation (CV1) and an outer cross-validation (CV2; Fig. 1 below). You can read about this in the NM manual ("Cross-validation settings"). In the inner CV1 cross-validation, features can be selected and parameters for models can be optimized. Then, these models are applied to the held-out information in the outer CV2 cross-validation folds. This separation of inner and outer-cross-validation means that we can optimize many aspects of the machine learning pipeline. You can change the cross-validation scheme based on your sample, features, and the research question.

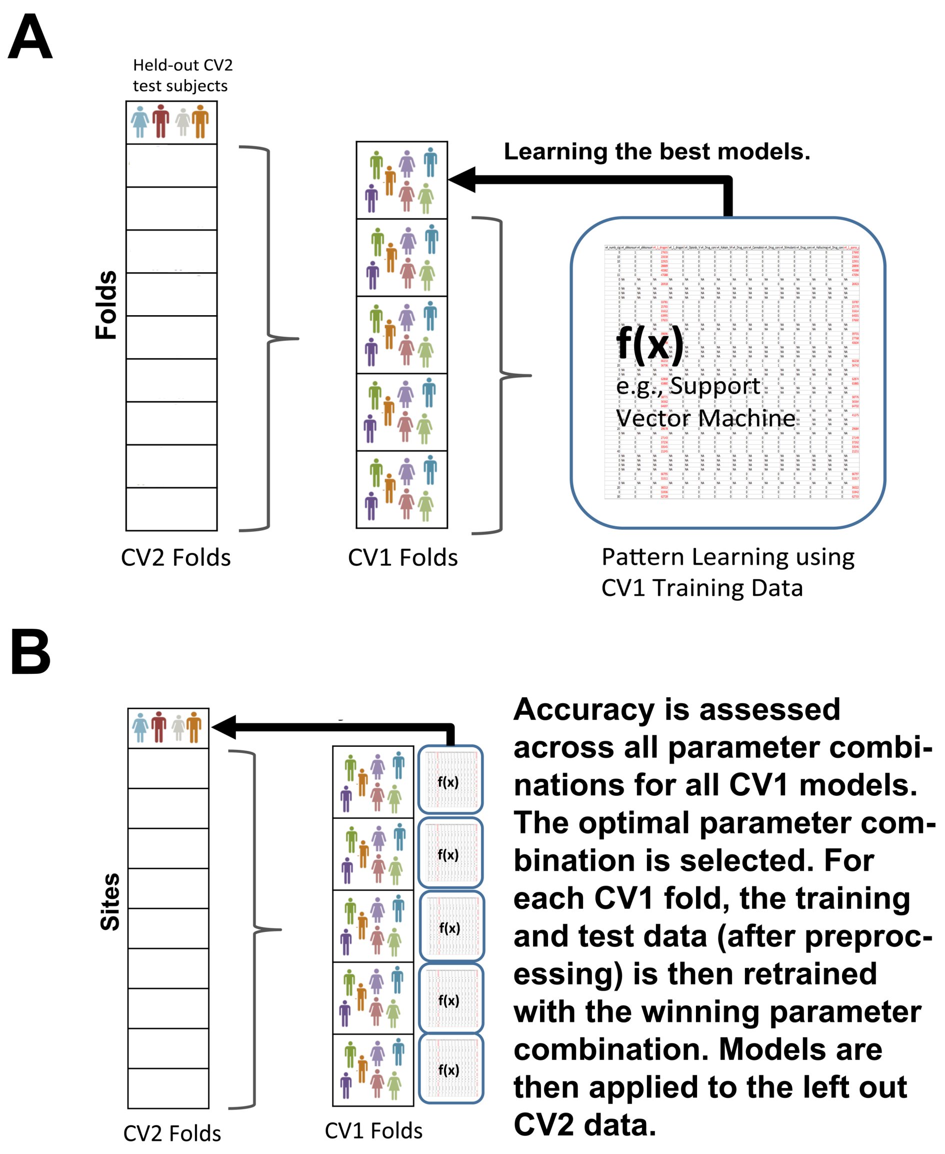

A
Held-out CV2 test subjects
Folds
Learning the best models.
f(x)
e.g., Support Vector Machine
CV2 Folds
CV1 Folds
Pattern Learning using CV1 Training Data
B
Sites
f(x)
CV2 Folds
CV1 Folds
Accuracy is assessed across all parameter combinations for all CV1 models. The optimal parameter combination is selected. For each CV1 fold, the training and test data (after preprocessing) is then retrained with the winning parameter combination. Models are then applied to the left out CV2 data.

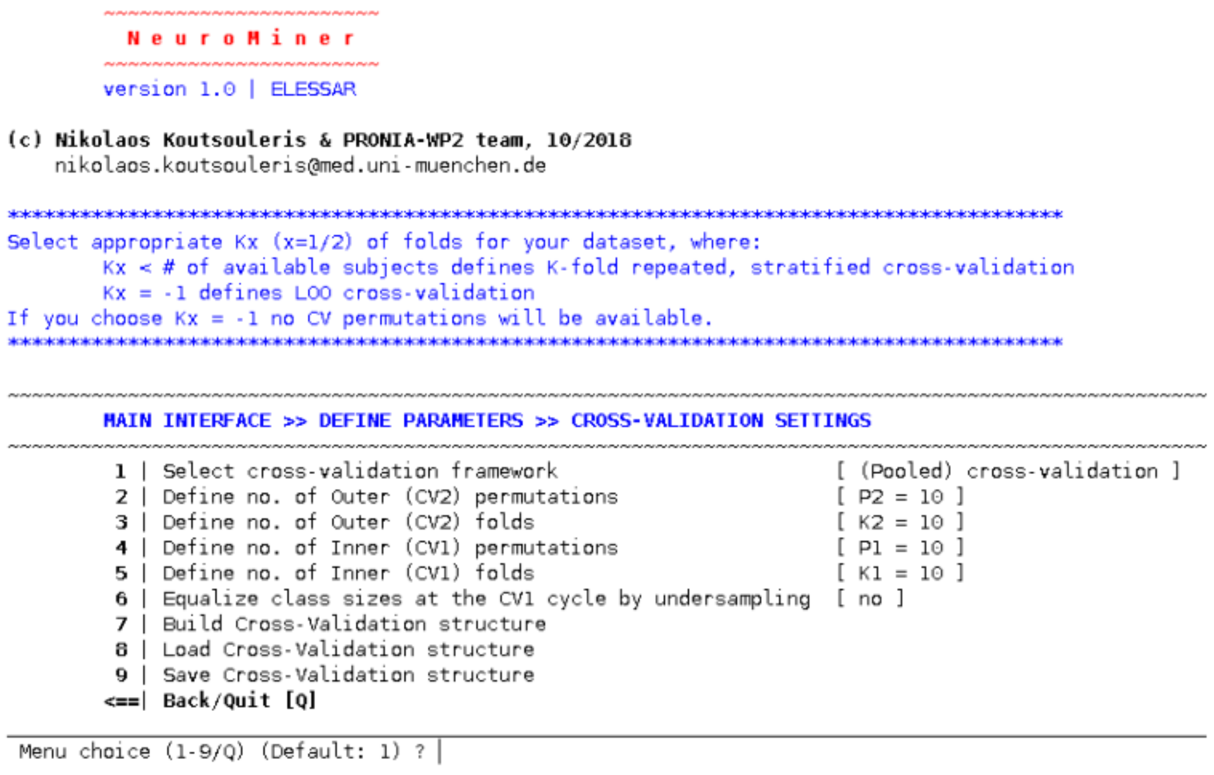
Figure 1. A) The data is split into k number of folds in the outer CV2 set and a CV2 test set is held-out. The rest of the data goes into the 'nest' where it is again subdivided into k-folds, a test fold is held-out, and the rest of the data is used for training. Models associated with each parameter combination (e.g., C or epsilon) are created in the CV1 training data

and applied to the CV1 test data. B) the results from all parameters or parameter combinations across all folds are assessed and the optimum is chosen (the user can define how the optimum is chosen). The preprocessed training and test data are then concatenated for each fold and these data are retrained using the winning parameter or parameter combination. Top % of parameters or parameter combinations can also be chosen by the user.

If you have successfully entered the data into NeuroMiner, do the following to see the cross-validation set-up:

Main Interface >> 2: Set up NM parameter workspace >> 1: Cross-validation settings

You should now see the following:



```
        ~~~~~~~~~~~~~~~~~~~~~~~
          N e u r o M i n e r
        ~~~~~~~~~~~~~~~~~~~~~~~
        version 1.0 | ELESSAR

(c) Nikolaos Koutsouleris & PRONIA-WP2 team, 10/2018
    nikolaos.koutsouleris@med.uni-muenchen.de

*****************************************************************************************
Select appropriate Kx (x=1/2) of folds for your dataset, where:
        Kx < # of available subjects defines K-fold repeated, stratified cross-validation
        Kx = -1 defines LOO cross-validation
If you choose Kx = -1 no CV permutations will be available.
*****************************************************************************************

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        MAIN INTERFACE >> DEFINE PARAMETERS >> CROSS-VALIDATION SETTINGS
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          1 | Select cross-validation framework                 [ (Pooled) cross-validation ]
          2 | Define no. of Outer (CV2) permutations            [ P2 = 10 ]
          3 | Define no. of Outer (CV2) folds                   [ K2 = 10 ]
          4 | Define no. of Inner (CV1) permutations            [ P1 = 10 ]
          5 | Define no. of Inner (CV1) folds                   [ K1 = 10 ]
          6 | Equalize class sizes at the CV1 cycle by undersampling [ no ]
          7 | Build Cross-Validation structure
          8 | Load Cross-Validation structure
          9 | Save Cross-Validation structure
        <==| Back/Quit [Q]
_________________________________________________________________________________________________
 Menu choice (1-9/Q) (Default: 1) ? |
```

The cross-validation menu by default is established with pooled cross-validation, which defines the folds randomly. However, you can choose to define the folds yourself by selecting “>> **1: Select cross-validation framework**” and entering the leave-group-out option. This is very useful if you have multiple sites in a study because it can measure inter-site generalizability. We will work with pooled cross-validation today as an introduction.

Much machine learning literature in medical and psychological sciences uses familiar frameworks, such as leave-one-out or simple k-fold cross-validation to test the generalizability of results. This holds one subject out, or a number of subjects out in k-fold, while the other subjects are used to create a model. The model is then tested on the held-out subject to assess the degree to which it generalizes to new cases. These methods are great, but they do not allow us to optimize the machine learning process without problems with overfitting—e.g., finding a non-linear decision boundary.

Unlike most psychological and psychiatric machine learning research, NeuroMiner was coded with repeated, nested, cross-validation as the default. The default settings include 10 permutations and 10 folds for both the outer cross-validation (CV2) cycle and the inner cross-validation (CV1) cycle. This means that the dataset will be randomly split into 10 CV2 folds, one of these folds will be held out, and the rest will go into the CV1 training cycle. The program will then cycle through all of the CV2 folds, and then it will repeat this entire process 10 times (i.e., 10 permutations). For each permutation, it will shuffle the participants before it creates the folds to include different participants in each fold. The nested cross-validation prevents **information leakage** between the training and test folds when we optimize parameters or feature selection (see later in the tutorial). Permuting the data makes the solutions more stable. Select the default settings by simply selecting

>> 7: Build Cross-Validation structure.

Tutor guided understanding of cross-validation.

After a short explanation of cross-validation by your tutor, please continue through these questions.

In the current dataset,

- Approximately, how many people will be included in:
  - a CV2 test fold?
  - a CV2 training set?
  - a CV1 test fold?
  - the CV1 training sets?
- What does this mean when we are creating models? Where are they created? How many people will be used to create the models?
- How many models are going to be calculated in total (keep in-mind the permutations!)? How does this affect the computational time?
- When would you need to use a different number of permutations or folds?

Exercise 2: Preprocessing and model selection

Preprocessing in NeuroMiner describes any processing steps that need to be applied to the data within each training fold before the data is forwarded to a multivariate learning method (e.g., SVM or random forest). These include steps such as smoothing images, scaling the data, dimensionality reduction, basic filtering, and many more functions. The parameters of these preprocessing steps are stored and the full preprocessing and testing analysis chain is applied to the test folds when it comes time to apply the models. This idea will be reinforced throughout the course, but for this analysis we only need to change a few simple things.

Main menu >> Set up parameter workspace >> Preprocessing pipeline

The default preprocessing settings for matrix data are to scale and remove (or 'prune') non-informative features. In addition to these steps, we would like to control for age and sex. In a classical analysis, we would regularly do this by regressing age and sex from the entire sample (i.e., including the cases and controls). However, there are known interactions between such variables as age and illness (e.g., the longer that someone is chronically ill the more that this will be reflected in brain changes). To avoid this problem, NeuroMiner gives us the ability to only model covariates in control subjects and then use this model to remove effects from cases using the method in **Dukart et al. (2011)**. This can be done with the following steps:

1: Add preprocessing step >> 1: Regress out nuisance covariates >> 1: Select covariates >> 1:2 >> 5: Define subgroup of training cases for computing betas >> 6: Provide Index to training cases for computing betas: ControlSubs

This will mean that age and sex is modelled with a linear regression in the control subjects, then the corrections for these variables will be applied to the entire dataset (i.e., including the individuals with schizophrenia). This means that we will not unintentionally remove the illness effects if there are interactions between illness and the covariates. It's a technique that has been used in multiple publications that are outlined in the manual.

After removal of the effects of covariates, you'll need to scale the data again because we need to do this for the SVM to work. To do this, return to the CV-PREPROCESSING PIPELINE menu and then select:

1: Add preprocessing step >> 4: Scale data >> Press Enter to return to the menu.

It is **critical to note that these steps will be performed within each of the CV1 training folds**. The pre-processing parameters will then be applied without modification to the CV1 test folds and the SVM to get the models. When the winning CV1 model is selected, the parameters will then be ultimately applied to the CV2 test folds (see **Figure 1**). Now go back to the "Define parameter template" screen by entering "Q" or pressing the enter key.

<u>Step 2: inspect the classification model</u>

Main menu >> Set up parameter workspace >> Classification algorithm

The default setting in NeuroMiner is to use a linear support vector machine (SVM) with accuracy (i.e., sensitivity + specificity) to assess the significance of the models. Accuracy is a good method of assessing the significance of the results if we don't care about whether the results are unbalanced (e.g., sensitivity = 90%, specificity = 15%), but we generally want to balance these two aspects of prediction. Thus, change the performance criterion to the following:

>> 1: Define ML model performance >> 4: Balanced Accuracy.

Notice that there are many other settings that could be used depending on your problem and many of these are outlined in Dwyer, Falkai, & Koutsouleris (2018).

Next we need to check the learning algorithm by selecting:

>> 2: Configure learning algorithm. You will see a range of different algorithms that can be used. There are also a range of other algorithms within option 8: matLearn. Today we're going to use a standard SVM.

Select **1: LIBSVM**. Here you will see settings regarding the support vector machine algorithm that **will be applied to each of the CV1 training folds**. NeuroMiner gives you the opportunity to change settings depending on the problem. If you are using unbalanced samples (i.e., where you have more people in one group than the other), then at this stage you could weight the hyperplane using option 7. This option is described in the manual. Currently, we are happy with these settings, so proceed and then **Select 1** for a linear kernel.

Step 3: inspect the machine learning parameters that are being used

Go to the "Learning algorithm parameters" screen by going to the parameter setup screen and selecting:

>> 4: learning algorithm parameters

The "Slack/Regularization parameter(s)" for an SVM is the C parameter. This is the thing we want to change to learn about hyperparameter optimisation, but for now we will just pick a number to simulate the situation where we are not using nested cross-validation. Select this option and then **change the setting to 0.01** and press enter. A researcher might pick this setting because they want to make sure that the models are highly generalizable to new samples. **However, it is mainly being used here for demonstrative purposes to show how hyperparameter optimisation works in a later step**.

Step 4: finish the parameter template and then run the preprocessing

Now go back to the "Define parameter template" screen. **Select 7: Model saving options**, do not save models, and then define a name; e.g., slack01. Then go back to the main menu by pressing "Q" or the enter key. Now initialize the analyses.

Initialisation of an analysis in NeuroMiner

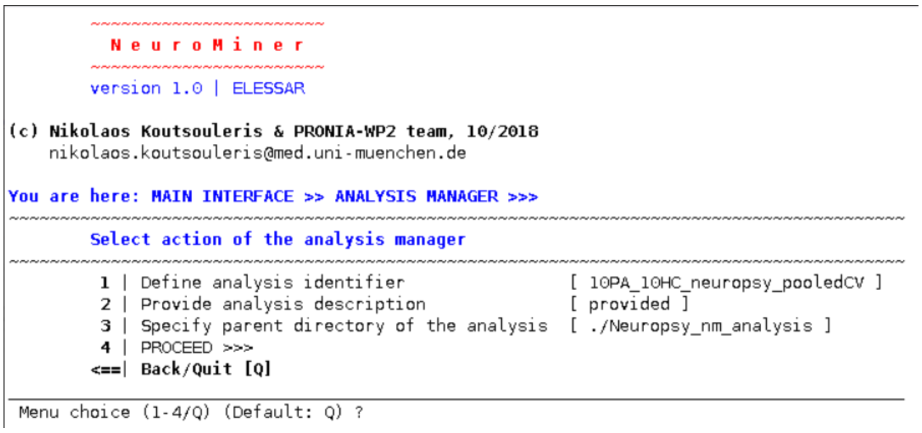
After the parameter template has been established, the settings can be saved into the NM structure through a process of "Initialization" on an analysis-by-analysis basis. This means that you can have multiple analyses within each NM structure and for each you would change the parameter template and re-initialise the analysis. Initialisation is a critical step in NeuroMiner.

To initialise an analysis, select the option to "**3: Initialize and manage analyses**". Then

>> 1: Define analysis identifier >> Enter a name that makes sense to this analysis. For example: cog_neuro_C01

>> 2: Provide analysis description >> Here you can enter a free text to describe your analysis in detail. Finish the description by pressing "." and then enter on a new line—i.e., the only thing on that line is a fullstop ".". This confuses a lot of people. If you need help ask.

>> 3: Specify parent directory of the analysis >> Select the directory where you want the analysis to be stored. You should then see the following.



```
          ~~~~~~~~~~~~~~~~~~~~~~~
           N e u r o M i n e r
          ~~~~~~~~~~~~~~~~~~~~~~~
          version 1.0 | ELESSAR

(c) Nikolaos Koutsouleris & PRONIA-WP2 team, 10/2018
    nikolaos.koutsouleris@med.uni-muenchen.de

You are here: MAIN INTERFACE >> ANALYSIS MANAGER >>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        Select action of the analysis manager
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         1 | Define analysis identifier              [ 10PA_10HC_neuropsy_pooledCV ]
         2 | Provide analysis description            [ provided ]
         3 | Specify parent directory of the analysis  [ ./Neuropsy_nm_analysis ]
         4 | PROCEED >>>
       <==| Back/Quit [Q]

 Menu choice (1-4/Q) (Default: Q) ?
```

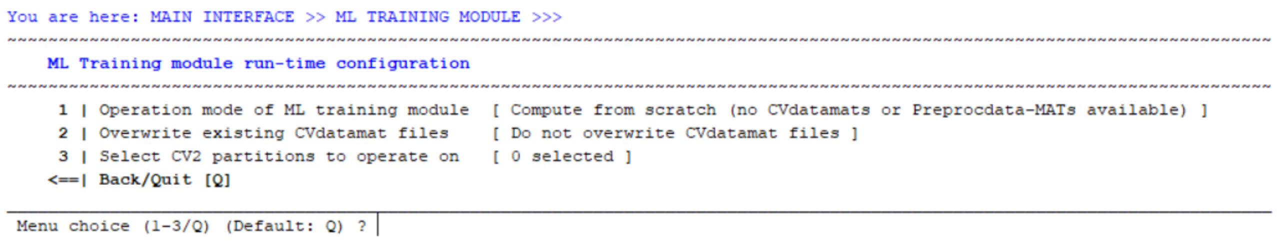
If this is all provided correctly, then you should be able to press the "**4: PROCEED**" option. This will save a folder with your folder name in your directory. This is where all the data will be stored during an analysis. Look at the directory. Know where it is. Then go back to the "**MAIN INTERFACE**".

Step 5: Training the classifiers and inspecting the results

Now that the data has been preprocessed, we can train the supervised classifiers with our C parameter that is set to 0.01.

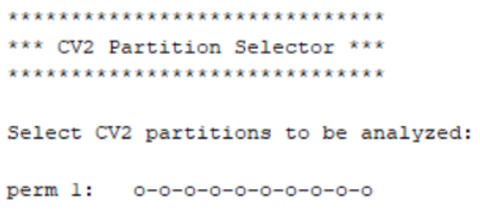
When your settings are initialised, you can preprocess and train your data and model. Sometimes it is useful to preprocess your data separately from training the models. Doing this will apply the settings you defined in your "Preprocessing pipeline" to the data within each fold that you have defined in your cross-validation settings. This is useful if you have multiple analytic strategies using the same preprocessing pipeline. For example, if you have the same preprocessing but then want to try an SVM in one analysis and a random forest in another analysis. However, here we just want to train the models from scratch and this will include the preprocessing as part of the pipeline. To do this, from the “MAIN INTERFACE” select the option to “5 Train supervised classifiers”:

You will be directed to the ML Training module run-time configuration, see below



```
You are here: MAIN INTERFACE >> ML TRAINING MODULE >>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    ML Training module run-time configuration
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      1 | Operation mode of ML training module  [ Compute from scratch (no CVdatamats or Preprocdata-MATs available) ]
      2 | Overwrite existing CVdatamat files    [ Do not overwrite CVdatamat files ]
      3 | Select CV2 partitions to operate on   [ 0 selected ]
    <==| Back/Quit [Q]
---------------------------------------------------------------------------------------------------------
 Menu choice (1-3/Q) (Default: Q) ?
```

We can use the defaults for all of these, but we need to “**3: Select CV2 partitions to operate on**”. This will show something like this



```
*****************************
*** CV2 Partition Selector ***
*****************************

Select CV2 partitions to be analyzed:

perm 1:   o-o-o-o-o-o-o-o-o-o
```

The option “Select CV2 partitions to operate on” is particularly important because you don’t have to run the analysis on all permutations and folds, but you can also choose to run the analysis in specific folds and permutations at the CV2 level. For example, you could select only the first permutation in order to check that everything is working before running the rest of the permutations. To do so keep in mind that:

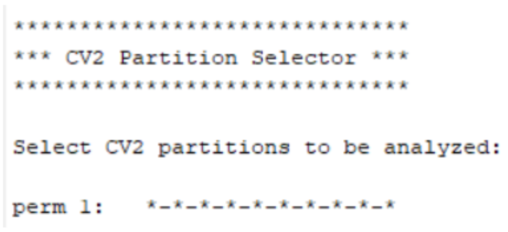
1. Perm 1: * * * * *. A star “*” indicates that the fold is selected, whereas the letter “o” indicates the fold is not selected; e.g., Perm 1: o o o o o.
2. All folds are selected by default in NeuroMiner, and thus you would need to de-select all folds first if you wanted to select specific perms or folds.
3. Columns represents folds, rows permutations.
4. The folds can be referenced by first indicating the permutation (e.g., 1st permutation) and then the fold position (e.g., 1st fold). For example, the second fold of the first permutation would be [1,2].
5. A fold range can also be selected in the following format [perm start, perm end, fold start, fold end]. For example, in pooled cross-validation designs, you may want to select the first two permutations and the first three folds in the following way: [1,2,1,3].

The figure therefore shows the CV2 structure of our analysis. As defined before, we have only 1 row because we choose to perform only 1 permutation (perm1), while we have 10 columns because we selected to split our CV2 cycle in 10 folds. The number of folds is usually based on the number of subjects that you have.

We want to select all CV2 partitions for training the classifier, so we type

>>1: Select all CV2 positions

Then, the partition selection will change and will look like this:



```
******************************
*** CV2 Partition Selector ***
******************************

Select CV2 partitions to be analyzed:

perm 1:    *-*-*-*-*-*-*-*-*-*
```

Then go back to the interface and press **4: PROCEED** in order to train.

Test your understanding:

- What is happening right now as it is training?
- What data is being scaled?
- How is the regression of covariates working?
- Once the analysis has been run, enter into the analysis directory that you defined during initialization and inspect the files there. What are these files?

If you are clear on the above points and it has finished, then go to the main menu and select the option to **7: Open NM Results Viewer**. A screen will appear. Please refer to the manual to understand the display results (section 4.9 Display training results, pg. 59). Make sure that you progress through the different results screens by using the drop down menu—e.g., the accuracy, the classification performance, the generalization error, and so on.

Questions:

- How many models are being created and applied to a single CV2 fold?

- How many models are being created overall?

- What is in the main results window?

- What is plotted on the Y-axis?

- What does the sensitivity, specificity, and balanced accuracy represent?

- What do the pie charts show?

Now we will optimize a hyperparameter of the SVM to see how this changes performance and alters the selection of models across cross-validation folds.

Exercise 3: Parameter optimisation within nested cross-validation

Once a nested, cross-validation pipeline is established then we can learn what the best models are in the inner CV1 cycle. In this exercise, we are going to focus on the C parameter associated with the SVM that we used yesterday and will use again today. The C parameter is critically important to classification using an SVM because it is a form of regularization that defines the degree of penalty that is applied when a case is misclassified and the decision boundary. High values of the C parameter mean that misclassifications will be heavily penalized during training and the decision boundary will be tightly fit around the data (mnemonic: "high **C** = high penal**TY**"; **Figure 2**). Theoretically, this means that the higher the C the more that the model will be representative of the sample, but potentially less representative of other samples—i.e., higher values of C should result in less generalizability. But the range of C depends on the problem and other hyperparameters that have to be optimized as you will find in the following two tutorials.

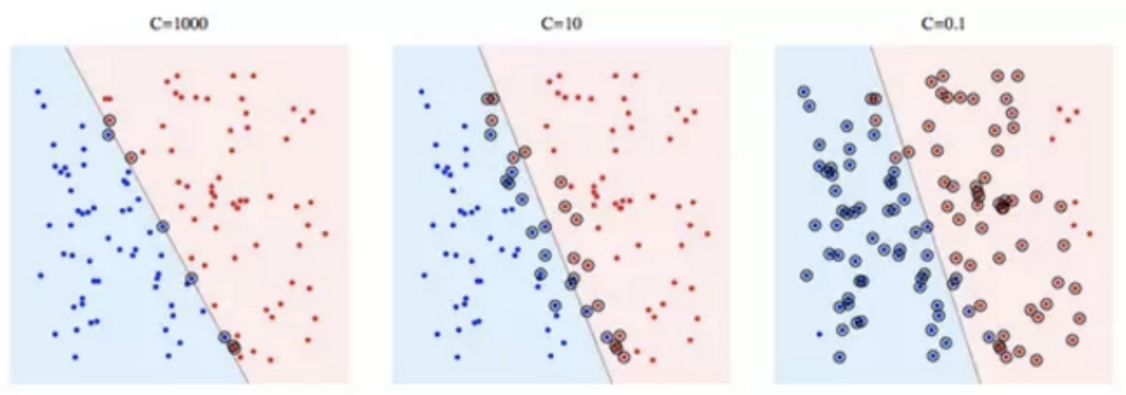


Figure 2. The C parameter. At high levels of C (left) a high penalty for misclassification will result less misclassifications and involve a harder decision boundary with less support vectors. As the C parameter is decreased, more errors are allowed, the soft boundary is relaxed, and there are more support vectors that are used.

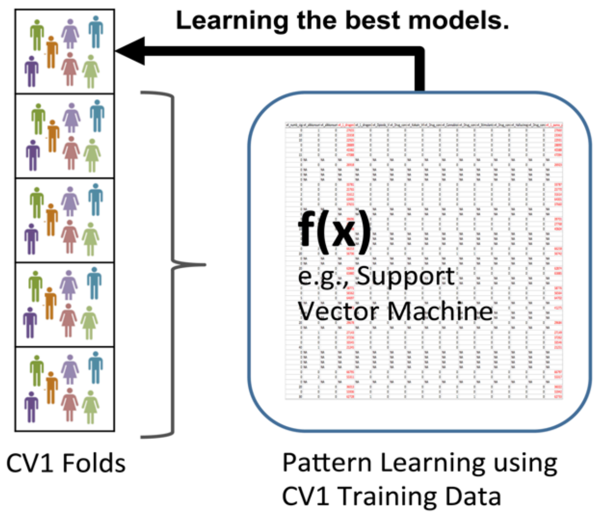
We are first going to run an analysis with only one C parameter and then we will run it again with multiple C parameters. Before we do this, we're going to establish all the settings as we did yesterday associated with the preprocessing and classification algorithms.

In this exercise, we will see how establishing a range for the hyperparameters changes the prediction. Before we do this, ask yourself what your hypothesis is. For example, if we optimize the C parameter (i.e., we determine what is the best hyperparameter across the models), then will it result in better or a worse accuracy when compared to no optimization?

Once you're ready to proceed, from the main menu do the following:

>> 2: Set up parameter workspace >> 4: Learning algorithm parameters >> 1: Define Slack/Regularization parameter(s) >> 2.^[-8:2:8]

On the first line of the parameters display, you should now see: "9 Params: 0.0039 (first) -> 256 (last)". Instead of just testing C=0.01, we are now going to apply all of these parameters to each inner **training** cross-validation set. After this they will all be applied to the inner **testing cross-validation fold** for that set. For this example, it will mean that for each CV1 test fold you will have 9 different models that quantify accuracy (Figure 3). It is critical to note that NeuroMiner will ultimately select winning models from the performance in the test folds of a nested CV design instead of the training folds. **Why would this be? Why is this important?**


Learning the best models.
f(x)
e.g., Support Vector Machine
CV1 Folds
Pattern Learning using CV1 Training Data

C parameters	0.004	0.016	0.062	0.25	1	4	16	64	256
Accuracy	51%	54%	58%	61%	68%	64%	60%	40%	50%

Figure 3. In a nested CV design, NeuroMiner learns each model in the CV1 training folds and then applies them to the test folds. This means that for each test fold, there will be as many performance measures (e.g., accuracy) as there are parameter combinations. NeuroMiner then assesses the performance of the models across all test folds using grid search. In the most basic set-up, NeuroMiner will then select the optimally performing

model, all CV1 preprocessed data from training and test folds will be retrained using this, and then all models will be applied to the held-out CV2 test fold.

NeuroMiner will repeat this process for all of the test folds in the inner, CV1 data and it will create a matrix of accuracies for each hyperparameter or parameter combination. For example, if there were 5 test folds then it might look like this:

C	**0.004**	**0.016**	**0.062**	**0.25**	**1**	**4**	**16**	**64**	**256**
Fold 1	51%	54%	58%	61%	70%	64%	60%	40%	50%
Fold 2	58%	64%	55%	60%	79%	70%	62%	61%	58%
Fold 3	50%	55%	56%	62%	77%	68%	65%	55%	56%
Fold 4	51%	54%	61%	65%	80%	70%	68%	60%	61%
Fold 5	52%	58%	62%	64%	73%	71%	65%	60%	55%

If the default settings are used, then NeuroMiner then finds the hyperparameter that results in the highest performance across all CV1 folds—in the example above, C=1 wins because it has the highest accuracy overall. There are multiple ways that you can determine a winner, please see the manual for details. After the winning parameter or parameter combination is chosen, then for each CV1 fold, NeuroMiner concatenates the preprocessed training and test data and then retrains these data with selected parameter or parameter combination. Finally, each model produced for each CV1 fold is then applied to outer CV2 test fold.

In the option "**Cross-parameter model selection process**" you can determine how the models are selected to be tested in the test CV1 fold. For example, you might decide that selecting the single optimum model is not a great idea because you have 70 parameters and you might have just found the best in that sample by chance but it won't generalise, so it might be better to select the top 5% of models because it may improve generalizability. However, since we only have 9 paramters we're going to keep the default setting and use the optimum model selection.

We should be happy with the change in settings now, so go back to the main menu and use the initialization menu that we have previously learnt about to initialize another analysis. Then go to the main menu, select the following:

>> 5: Train supervised classifiers (we don't have to preprocess again because we've already done this) >> 1: Choose analysis to work on >> 2: Go to next analysis >> 1: Select current analysis >> 2: Operation mode of ML training module >> 2: Compute using existing preprocdata-MATs >> 3: Specify PreprocData files >> 2: Manual file selection >> Select the files >> Save PreprocMaster file: no >> 6: PROCEED

Once it has finished return to the main menu and go to the "Open NM results viewer (cross-validation results)". In the top left corner, select the drop-down box that defines the analysis and select analysis 2. Interpret the results with reference to analysis 1 where you only used one hyperparameter (i.e., C=0.01).

<u>Visualisation</u>

Now you want to visualise the weights on the brain. Go back to the Main Interface

>> 6: "Visualise Classifiers" >> select the data that you've processed (datamats) and then hit **7: proceed.**

Viewing Results

Now the analysis is done and we can begin to review the results. Understanding results from NeuroMiner will take some time and be covered in further tutorials, but you can get an idea of the results by going to the "MAIN INTERFACE" and then selecting "**7: Open NM results viewer**".

The graphic interface of NeuroMiner will open. Explore the training results, the different options of the graphic interface and the visualization results. Refer to the NeuroMiner manual to explain the different results.

Answer the questions and then test your understanding:

- What does the C parameter do?
- What did we do when the models were training?
- How does the accuracy change without optimisation and with optimisation?
- What does each point in the test performance graphs represent?
- How does the ratio of the CV1 and CV2 test performance change over the parameters?
- Why does the CV1 and CV2 test performance change with reference to the C parameter?
- Why did we have to use nested cross-validation to do this?

- What are some other parameters that we can optimize?

- What is the danger of optimizing too many parameter combinations? How could we control for this?

If you've clear on these questions then it's excellent because now you understand hyperparameter optimization, which is a critical aspect of machine learning. We have optimized one parameter here using SVM, but it is critical to note that you could optimize many more parameters using this nested cross-validation design. For example, in NeuroMiner you could change the SVM from being linear to non-linear (e.g., an RBF) and then optimize the non-linear kernel parameters. Within NeuroMiner you can even optimise parameters of the preprocessing steps, such as smoothing, filtering, or image compression.

Summary and Discussion

We have covered nested cross-validation, preprocessing, SVM parameters, and parameter optimisation. The main take-home message from this is that we can use supervised machine learning to find the parameters that best predict the label of an individual. In this case it was schizophrenia, which means that we've built models that could be applied to new individuals to predict their schizophrenia diagnosis using their brain scan information. Because we have cross-validated our models, we can be more confident that this will generalize in the real-world (with some big caveats).

Please address any questions you might have and discuss the pros and cons of using this machine learning approach. One question to address is:

- **Can these techniques really create models that could be applied in hospital settings right now as decision aides?**
- **How could they be biased?**
- **Would it be ethical to use something like this in cases where the individual is at-risk but does not have the condition?**

As a side note, you have done very well if you understand and could implement even half of what we have covered today. Keep in-mind that for many of these concepts and topics, the way to learn them is through experience, reading, and repetition. If you have come from a different background (e.g., classical statistics) then it sometimes takes quite a while to get your head around a related, but different, paradigm.