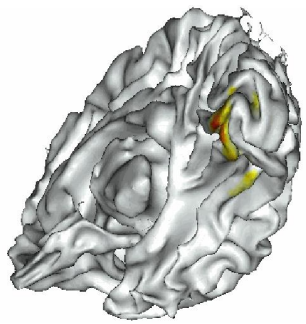
# The AnalyzeFMRI package

Pierre Lafaye de Micheaux

Department of Mathematics and Statistics
Université de Montréal

UseR 2010 Tutorial
Gaithersburg, USA
July, 20 2010
http://www.neurostatistician.eu/AnalyzeFMRI



# Reading and writing files

A necessary step in every fMRI analysis with **R** is to be able to read/write data files.

Two formats are widely used by the neuroimaging community :

- ANALYZE 7.5
- NIFTI-1

# The ANALYZE 7.5 format

The ANALYZE (7.5) format image consists of two files :

`.img` : consists of one long stream of voxel intensities

`.hdr` : header that contains information about the .img file

**Note :** SPM (96/99/2) uses the basic ANALYZE 7.5 format, but with three modifications / extensions : scale factor (`funused1`), voxel coordinates of image origin (`originator`) and a `.mat` file. See : `imaging.mrc-cbu.cam.ac.uk/imaging/FormatAnalyze`

# Summary of an ANALYZE header file

```
> f.analyze.file.summary(system.file("example.hdr",
                                     package="AnalyzeFMRI"))

  File name: /usr/lib/R/library/AnalyzeFMRI/example.img
Data Dimension: 4-D
   X dimension: 64
   Y dimension: 64
   Z dimension: 21
Time dimension: 1 time points
Voxel dimensions: 4 mm x 4 mm x 6 mm
      Data type: signed short (16 bits per voxel)
```

# Reading of an ANALYZE header file : f.read.analyze.header()

```
Description:
  Reads the ANALYZE image format .hdr header file
  into a list.
Usage:
  f.read.analyze.header(file)
Arguments:
  file: The .hdr file to be read
Value:
  A list containing the information in the fields
  of the .hdr file.
```

# Reading ANALYZE header files

For a full description of the header file, please try :

```
res <- f.read.analyze.header(system.file("example.hdr",
                                         package="AnalyzeFMRI"))

names(res)

res

?f.read.analyze.header
```

# Reading parts of ANALYZE .img files

Without loading the whole file into **R** memory :

```
f.read.analyze.slice()  Read one slice from a .img file

f.read.analyze.slice.at.all.timepoints()
                        Reads a slice at all time points
                        from a .img file

f.read.analyze.tpt()    Read in a volume at one time point

f.read.analyze.ts()     Read in one voxel time series
```

# Reading whole or part of ANALYZE .img files : examples

You can try :

```
file <- system.file("example.img",package="AnalyzeFMRI")

# Full img file:
f.read.analyze.volume(file)

# Parts of img file:
f.read.analyze.slice(file,slice=10,tpt=1)
f.read.analyze.slice.at.all.timepoints(file,slice=10)
f.read.analyze.tpt(file,tpt=1)
f.read.analyze.ts(file,x=30,y=30,z=10)
```

# Writing ANALYZE .hdr files

```
f.basic.hdr.list.create(X, file.hdr)
     creates basic .hdr list in ANALYZE format

f.write.list.to.hdr(L,file)
     Writes a list of attributes to a .hdr file
```

# Writing ANALYZE .img/.hdr pair files

```
f.write.analyze(mat,file,size,pixdim,vox.units,
                                cal.units,originator)
```

writes an array to a .img/.hdr pair in ANALYZE format

# The NIFTI format

NIFTI-1 is adapted from the widely used ANALYZE 7.5 file format. NIFTI-1 uses the "empty space" in the ANALYZE 7.5 header to add several new features. These innovations include :

- Affine coordinate definitions relating voxel index (i,j,k) to spatial location (x,y,z) ;
- Codes to indicate spatio-temporal slice ordering for fMRI ;
- "Complete" set of 8-128 bit data types ;
- Standardized way to store vector-valued datasets over 1-4 dimensional domains ;
- Codes to indicate data "meaning" ;
- A standardized way to add "extension" data to the header ;
- Dual file (`.hdr` & `.img`) or single file (`.nii`) storage ;

and many more goodies.

# Summary of a NIFTI header file

```
> f.nifti.file.summary(system.file("example-nifti.img",
                                   package="AnalyzeFMRI"))

File name:/usr/lib/R/library/AnalyzeFMRI/example-nifti.img
  Data Dimension: 3-D
     X dimension: 53
     Y dimension: 63
     Z dimension: 46
  Time dimension: 1 time points
Voxel dimensions: 3  x 3  x 3
       Data type:  (8 bits per voxel)
```

# Reading NIFTI header files : f.read.nifti.header()

```
Description:
  Reads the NIFTI image format .hdr header
  file into a list
Usage:
  f.read.nifti.header(file)
Arguments:
  file: The .hdr file to be read
Value:
  A list containing the information in the fields
  of the .hdr file.
```

# Reading NIFTI header files

For a full description of the header file, please try :

```
res<-f.read.nifti.header(system.file("example-nifti.img",
                                     package="AnalyzeFMRI"))

names(res)

res

?f.read.nifti.header
```

# Reading ANALYZE or NIFTI header files : `f.read.header()`

The function `f.read.header()` can read **both** ANALYZE `.hdr` or NIFTI `.hdr` (or `.nii`) header files.

```
Description:
    Reads the ANALYZE or NIFTI image format .hdr (or .nii)
    header file into a list. The format type is determined
    by first reading the magic field.
Usage:
    f.read.header(file)
Arguments:
  file: The .hdr (or .nii) file to be read
```

# Reading parts of NIFTI .img (.nii) files

```
f.read.nifti.slice()  Read one slice from a .img (or .nii)
                      in NIFTI format

f.read.nifti.slice.at.all.timepoints()
                      Read a slice at all time points
                      from a NIFTI .img (or .nii) file

f.read.nifti.tpt()    Read in a volume at one time point

f.read.nifti.ts()     Read in one voxel time series
```

# Reading whole or part of NIFTI .img (.nii) files

You can try :

```
file <- system.file("example-nifti.img",
                    package="AnalyzeFMRI")

# Whole .img (or .nii) file:
f.read.nifti.volume(file)

# Parts of .img (.nii) file:
f.read.nifti.slice(file,slice=10,tpt=1)
f.read.nifti.slice.at.all.timepoints(file,slice=10)
f.read.nifti.tpt(file,tpt=1)
f.read.nifti.ts(file,x=30,y=30,z=10)
```

# Reading whole ANALYZE or NIFTI .img (.nii) files

```
f.read.volume()


Description:
     Reads the ANALYZE or NIFTI format image
     file into an array. Autodetects format type.
Usage:
     f.read.volume(file)
Arguments:
    file: The location of the image file to be read
```

# Writing NIFTI .hdr files

```
f.basic.hdr.nifti.list.create(dim.mat, file.hdr)
     creates basic .hdr list in NIFTI format

f.write.list.to.hdr.nifti(L,file)
     writes a .hdr file in NITI format

f.complete.hdr.nifti.list.create(many parameters)
     Creates a complete list that can be used to
     write a .hdr file or the header part of
     a .nii file
```

# Writing NIFTI .img/.hdr or .nii files

```
f.write.nifti(mat,file,size,L,nii)
  Creates a .img/.hdr pair of files or a .nii file
  from a given array

     mat: An array
    file: Name of the file to be written, without
          .img or .hdr suffix
    size: "float" (for 4 byte floats), "int" (2 byte
          integers) or "char" (1 byte integers).
       L: header.If NULL, the list is created by the
          function else it should be provided.
     nii: should we write only one .nii file or a
          .hdr/.img pair of files
```

# The `magicfield`

`magicfield()` `Get magicfield from the header of an image file`

We read the magic field of a file to determine in which type it is (ANALYZE or NIFTI). The magic field can be either one of :

- `""` for an ANALYZE .hdr/.img pair ;
- `"ni1"` for a NIFTI-1 .hdr/.img pair ;
- `"n+1"` for a NIFTI-1 .nii file.

**Note :** output can also be `"nix"` or `"n+x"`, x=0,...,9, for (future) NIFTI-x files.

# Conversion from ANALYZE to NIFTI

```
analyze2nifti()  Create a NIFTI file from an ANALYZE file
```

# File storage

The image file consists of "raw" voxel intensity values that are stored sequentially. The image file's voxel order is defined as follows :

- The file x-dimension (i) is defined as the dimension that changes most rapidly (i.e., each sequential voxel will fall in a different column and will therefore have a different x coordinate).
- The file y-dimension (j) changes more slowly than the x-dimension and more quickly than the z-dimension.
- The file z-dimension (k) is defined as the dimension that changes most slowly (e.g., all of the voxels for a given z-plane are stored before any of the voxels of the next z-plane).

# Be careful !

Real-world coordinates do not necessarily correspond to voxel coordinates, i.e. how the image is stored in the file !

For instance, while real-world $x$ coordinates might be **increasing** as you move from the left side of the brain to the right side, the actual associated voxel index $i$ in the volume could be **decreasing**. In this case, the image is said to be stored in Right-to-Left format.

This is an example of a **left-handed** coordinate system and is the convention for ANALYZE format files.

# Be careful !

The opposite movement of voxel coordinates and real-world coordinates in the $x$ dimension is what leads people to talk about ANALYZE files as being **stored in "radiological" format**.

However, images can be **displayed** radiologically or neurologically irrespectively of the format that the data are stored in. In fact, neurological and radiological conventions only relate to visualization of axial images.

In the ANALYZE and NIFTI formats, voxel data are retrieved in an **R** array such that the location (in the file) of voxel $(i, j, k)$ is :

$$i + (j-1) \times dim[2] + (k-1) \times dim[2] \times dim[3],$$

where $1 \leq i \leq dim[2]$, $1 \leq j \leq dim[3]$, $1 \leq k \leq dim[4]$.

The ANALYZE convention is a **left-handed** coordinate system (radiological convention) :

$$i \leftrightarrow x \text{ (}\mathbf{R}\text{ to }\mathbf{L}\text{)}, \quad j \leftrightarrow y \text{ (}\mathbf{P}\text{ to }\mathbf{A}\text{)}, \quad k \leftrightarrow z \text{ (}\mathbf{I}\text{ to }\mathbf{S}\text{)}.$$

The default NIFTI convention is a **right-handed** system (neurological convention) :

$$i \leftrightarrow x \text{ (}\mathbf{L}\text{ to }\mathbf{R}\text{)}, \quad j \leftrightarrow y \text{ (}\mathbf{P}\text{ to }\mathbf{A}\text{)}, \quad k \leftrightarrow z \text{ (}\mathbf{I}\text{ to }\mathbf{S}\text{)}.$$

# Some basic conventions

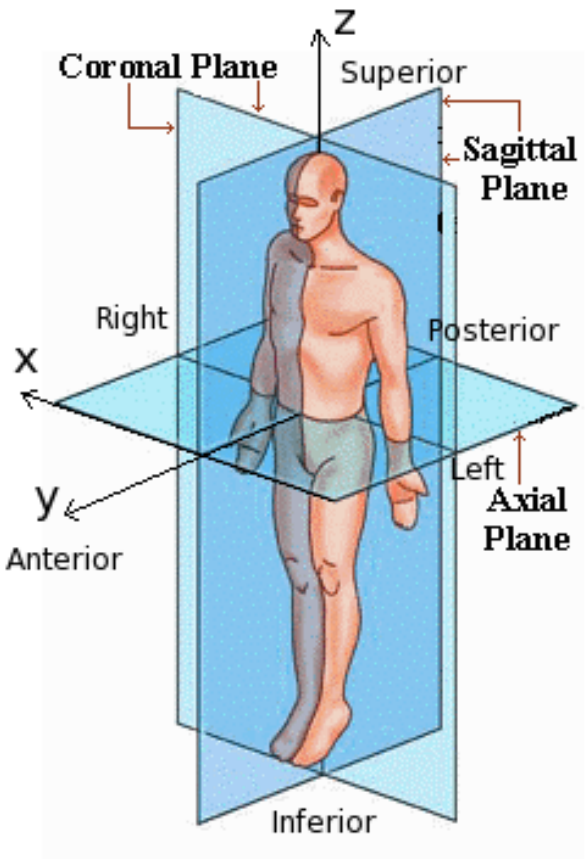
Note that directions are given relative to the patient. For example :

- The left side of the brain is denoted by negative $x$ coordinates, and regions in the right side of the brain are given by positive $x$ coordinates. Thus, real-world $x$ coordinates increase from **L**eft to **R**ight.
- Real-world $y$ coordinates increase from **P**osterior to **A**nterior.
- Real-world $z$ coordinates increase from **I**nferior to **S**uperior.

This is the way that the Talairach and Tournoux atlas is organized. This is an example of a **right-handed** real-world coordinate system : thumb = **R**, 2nd finger = **A**, middle finger = **S**.

# Some basic conventions

## Neurologist RAS axes



# Three possible transformations

**Note :** The NIFTI format contains header information that allows the $x, y, z$ directions of the brain to be assigned to other $x, y, z$ arbitrary patient directions. This means that we assign directly $(i, j, k)$ to special transformed $(x, y, z)$ real-world coordinates. This is done via :

- **Method 1 :** Scaling ;
- **Method 2 :** Rigid body transform (scaling+rotation+shift) ;
- **Method 3 :** Affine transformation (scaling+shearing+ rotation+shift), i.e. lines that are parallel before transformation remain parallel after transformation.

# Method 1 : `qform.code` $= 0$ and `sform.code` $= 0$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} pixdim[2] \times i \\ pixdim[3] \times j \\ pixdim[4] \times k \end{pmatrix}$$

where

$$\begin{cases} pixdim[2] : & \text{voxel width in mm.} \\ pixdim[3] : & \text{voxel height in mm.} \\ pixdim[4] : & \text{slice thickness (interslice distance) in mm.} \end{cases}$$

**Note :** This is what is done for standard ANALYZE files, i.e. no orientation is given or assumed.

# Method 2 : qform.code $> 0$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}}_{R\text{:rotation matrix}} \underbrace{\begin{pmatrix} pixdim[2] \times i \\ pixdim[3] \times j \\ qfac \times pixdim[4] \times k \end{pmatrix}}_{Scaling} + \underbrace{\begin{pmatrix} qoffset.x \\ qoffset.y \\ qoffset.z \end{pmatrix}}_{Translation}$$

We have $qfac = pixdim[1] = \pm 1$. The $R$ (proper) rotation matrix is encoded into NIFTI files using quaternions parameters (quatern.b, quatern.c and quatern.d), see later.

# Method 3 : `sform.code` $> 0$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{bmatrix} srow.x[1] & srow.x[2] & srow.x[3] & srow.x[4] \\ srow.y[1] & srow.y[2] & srow.y[3] & srow.y[4] \\ srow.z[1] & srow.z[2] & srow.z[3] & srow.z[4] \end{bmatrix}}_{\text{affine matrix}} \begin{pmatrix} i \\ j \\ k \\ 1 \end{pmatrix}$$

**Note :** SPM (old versions) uses external `.mat` files (in addition to the Analyze `.img`/`.hdr`) to store affine transformations.

# The function ijk2xyz()

```
ijk2xyz(ijk=c(1,1,1),method=2,L)

Arguments:
 ijk: matrix. Each column of ijk should contain a voxel
      index coordinates (i,j,k) to be mapped to its (x,y,z)
      real coordinates in some other space
 method: 1 (qform.code=sform.code=0),2 (qform.code>0,
      rigid transformation) or 3 (sform.code>0, affine
      transformation).
 L: header list of a NIFTI file
```

# The function xyz2ijk()

```
xyz2ijk(xyz=c(1,1,1),method=2,L)

Arguments:
 xyz: matrix. Each column of xyz should contain a voxel
      real world index coordinates (x,y,z) to be mapped
      to its (i,j,k) voxel index coordinates in the
      dataset
 method: 1 (qform.code=sform.code=0),2
      (qform.code>0, rigid transformation) or 3
      (sform.code>0, affine transformation).
 L: header list of a NIFTI file
```

The quaternion representation $(a, b, c, d)$ is chosen for its compactness in representing rotations. We have $a^2 = 1 - b^2 - c^2 - d^2$ and

$$R = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2bd + 2ac \\ 2bc + 2ad & a^2 + c^2 - b^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 + d^2 - c^2 - b^2 \end{bmatrix}$$

Note that rotations stored in NIFTI headers (using quaternions) are always proper ($det(R) = 1$). Improper rotation is the combination of an ordinary rotation of 3D Euclidean space, that keeps the origin fixed, with an inversion (x goes to -x for example). An improper rotation of an object thus produces a rotation of its mirror image. Note also that the `qfac=pixdim[1]` parameter is necessary to handle the case of improper rotations.

# Various geometrical utilities

```
mat34.to.TRSZ()     Affine 4x4 (or 3x4) matrix to
                    Translation, Rotation, Shear and Scale
mat34.to.TZSR()     Affine 4x4 (or 3x4) matrix to
                    Translation, Scale, Shear and Rotation
nifti.quatern.to.mat44()
                    Quaternion (etc..) to affine 4x4 matrix
Q2R()               Quaternion to (proper) rotation
R2Q()               (proper) Rotation to quaternion
```

# Various utilities

`diminfo2fps()` Extract freq.dim, phase.dim and slice.dim fields from the one byte dim.info field of a NIFTI header file.

`fps2diminfo()` Encode freq.dim, phase.dim and slice.dim fields into the one byte dim.info field of a NIFTI header file

`st2xyzt()` Encode space and time dimensions fields into the one byte xyzt.units field of a NIFTI header file.

`xyzt2st()` Extract space and time dimensions fields from the one byte xyzt.units field of a NIFTI header file.

# Various utilities

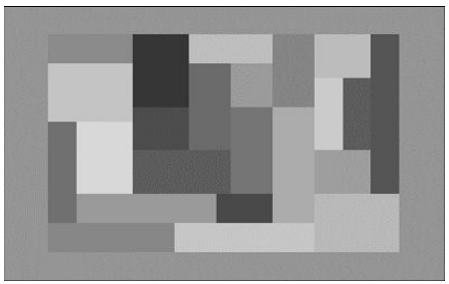
`fourDto2D()` Transforms a 4D image into a 2D image matrix of size tm x vm

`threeDto4D()` To read tm functionnal images files in ANALYZE or NIFTI format, and concatenate them to obtain one 4D image file in Analyze (hdr/img pair) or Nifti format (hdr/img pair or single nii) which is written on disk.

`twoDto4D()` This function transforms a 2D matrix of size tm x vm containing images in each row into a 4D array image.

# An fMRI experiment

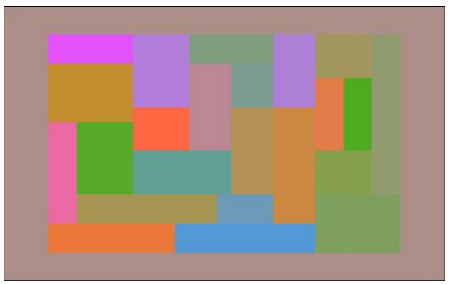
## Stimulus



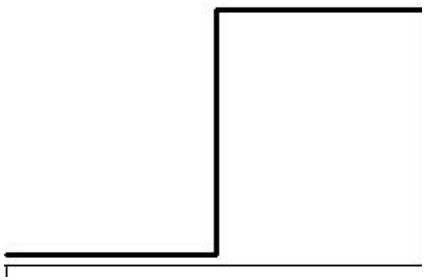
2 alternating periods : activity/ “rest”

# An fMRI experiment

Stimulus

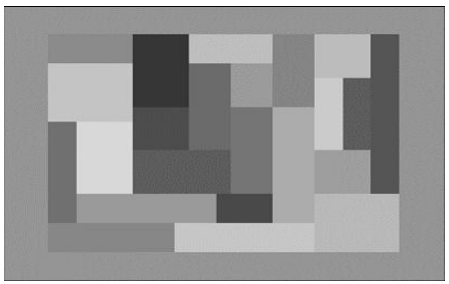


2 alternating periods : activity/ “rest”

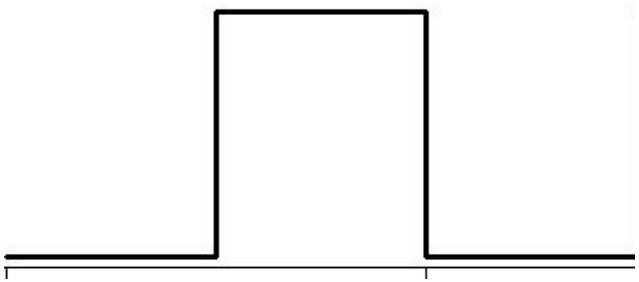


# An fMRI experiment

Stimulus

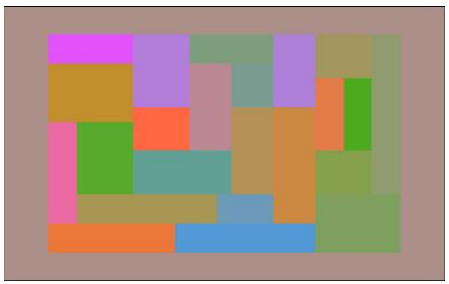


2 alternating periods : activity/ "rest"

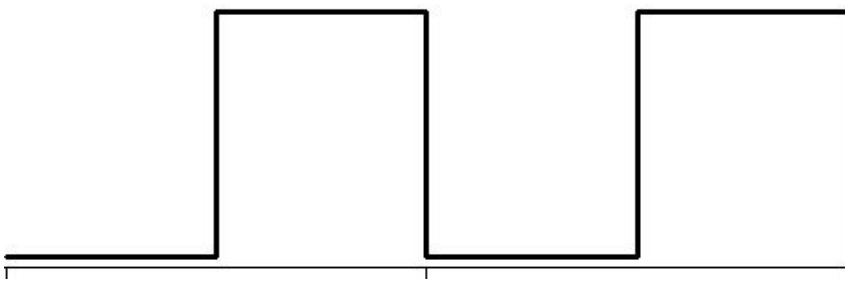


# An fMRI experiment

Stimulus

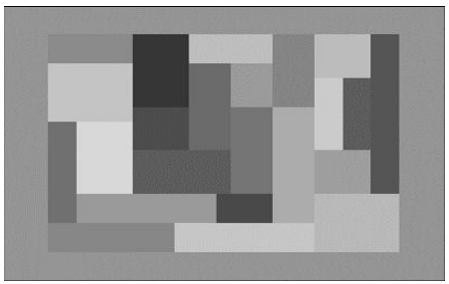


2 alternating periods : activity/ "rest"

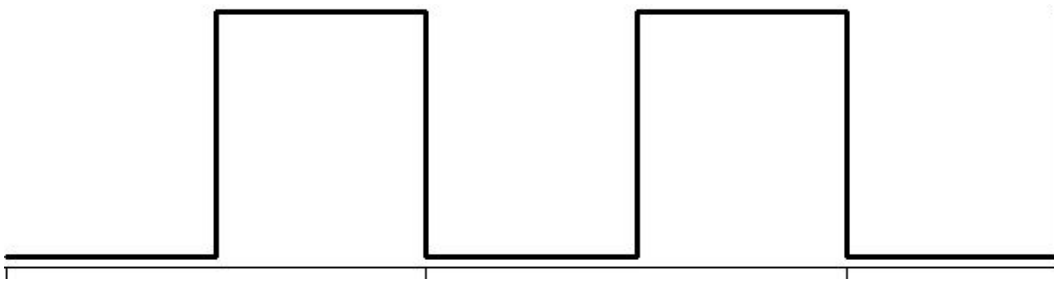


# An fMRI experiment

Stimulus

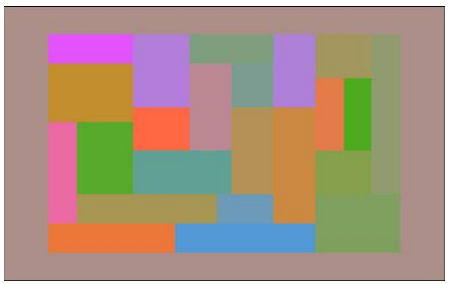


2 alternating periods : activity/ "rest"

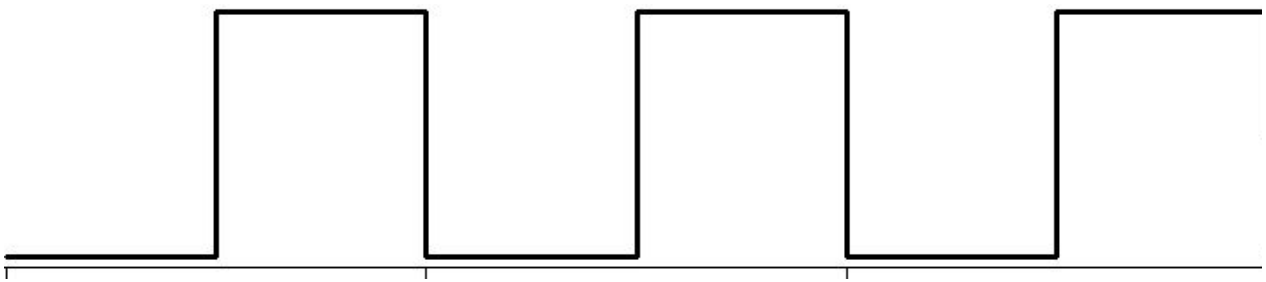


# An fMRI experiment

Stimulus

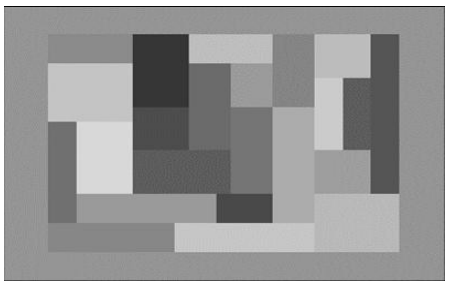


2 alternating periods : activity/ "rest"

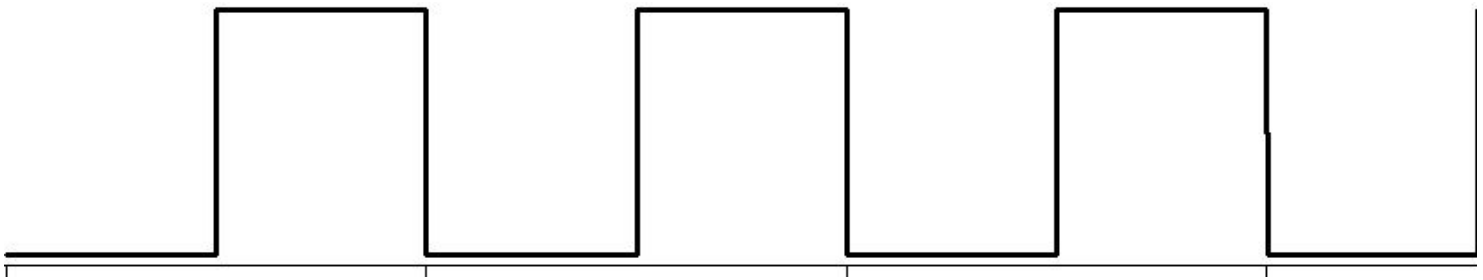


# An fMRI experiment

Stimulus

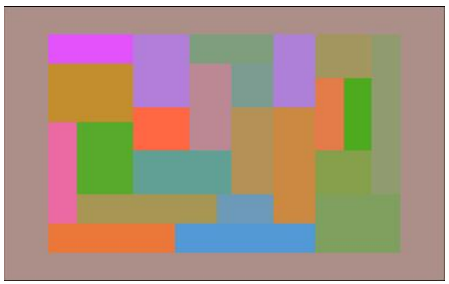


2 alternating periods : activity/ “rest”

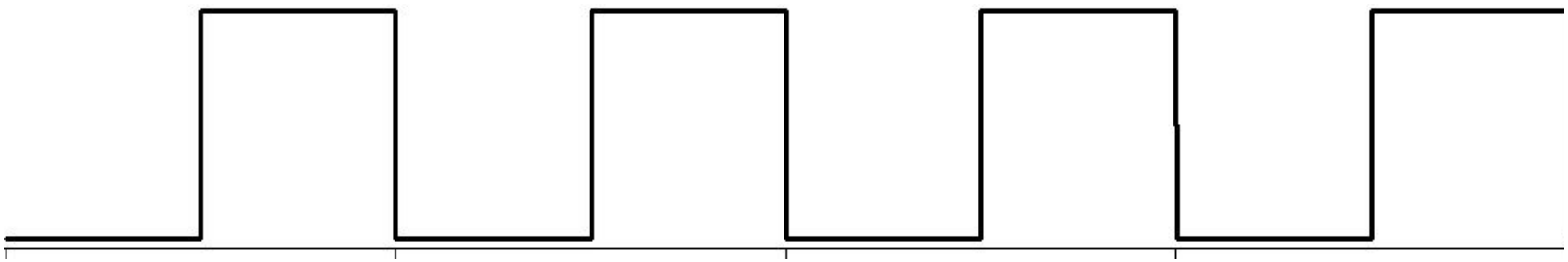


# An fMRI experiment

Stimulus



2 alternating periods : activity/ "rest"



# The problem of blind source separation

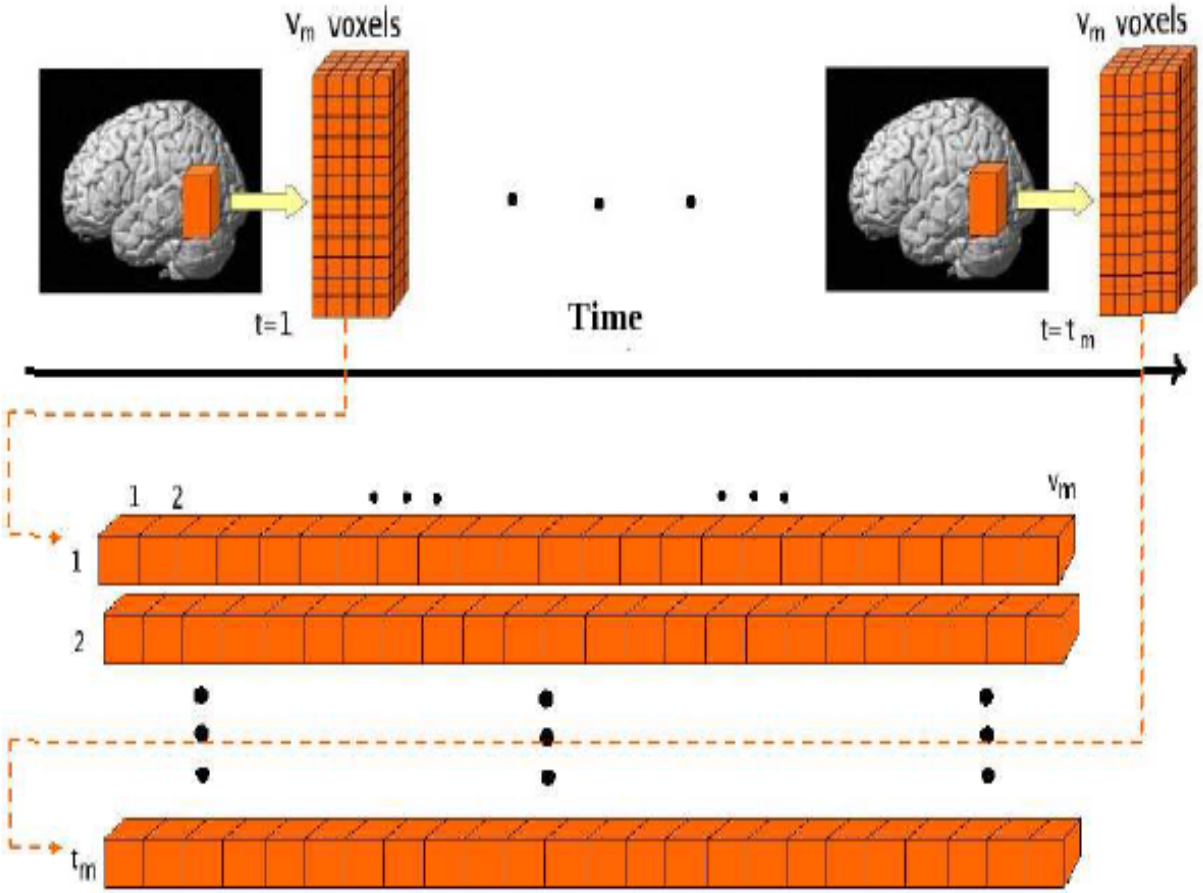
- Each voxel contains a **mixing** of several **source signals** : *heart rate*, *eyes movement*, *respiratory cycle*, *stimulus of the experiment*, ...
- These signals are mixed in different proportions depending on the voxel. The weighting changes with time.
- Thus, we are facing here a **source separation problem**.

# The problem of blind source separation

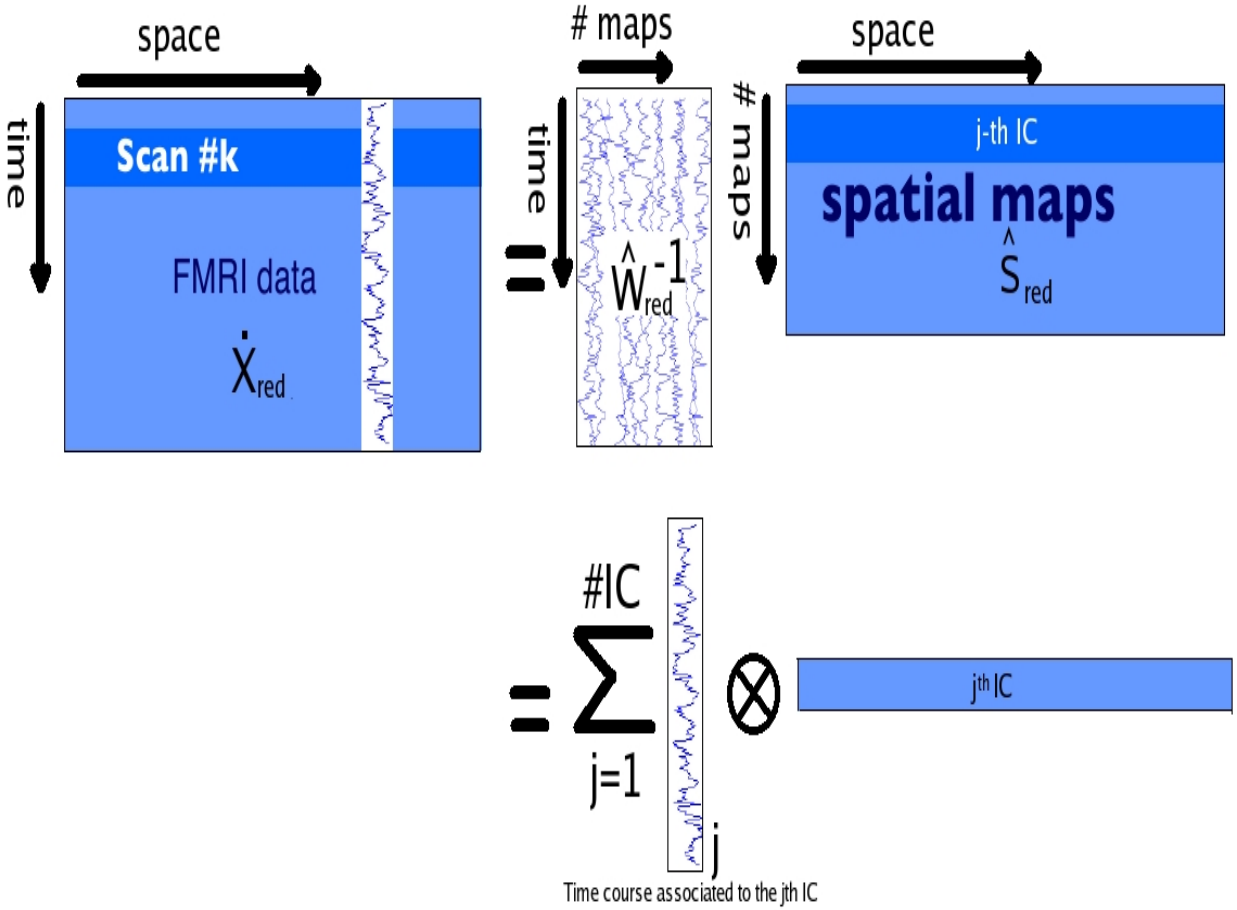
- Each voxel contains a **mixing** of several **source signals** : *heart rate*, *eyes movement*, *respiratory cycle*, *stimulus of the experiment*, ...
- These signals are mixed in different proportions depending on the voxel. The weighting changes with time.
- Thus, we are facing here a **source separation problem**.

$\Longrightarrow$ **Independent Component Analysis (ICA)**

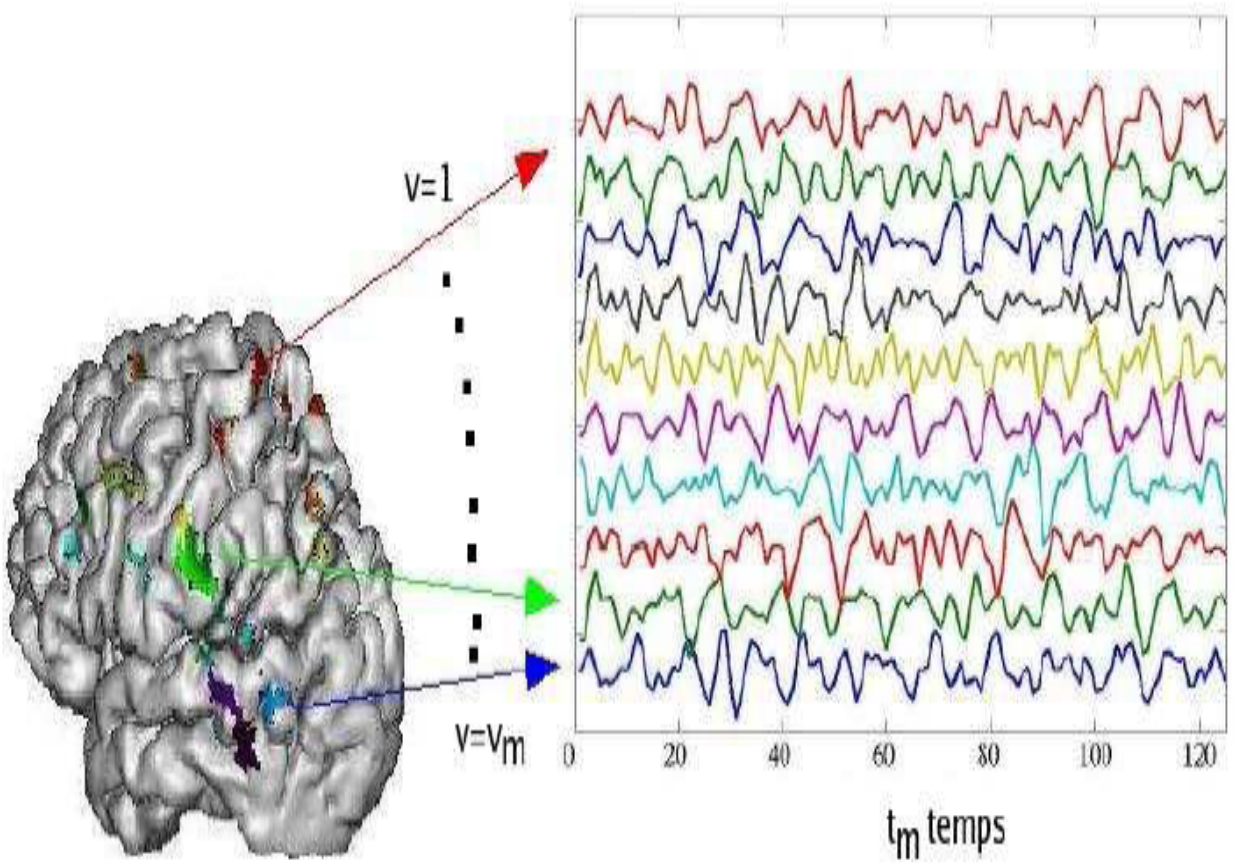
# Spatial decomposition



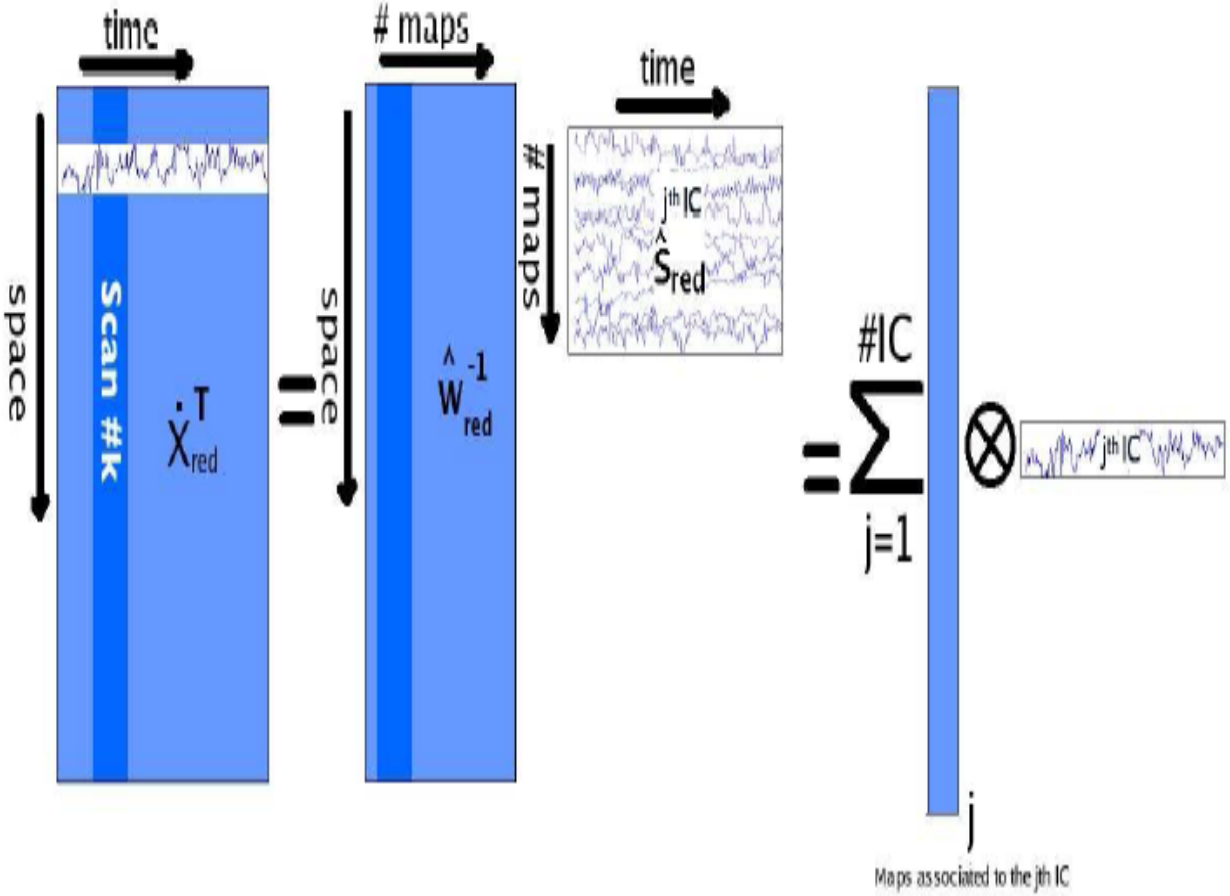
# Spatial decomposition



# Temporal decomposition



# Temporal decomposition



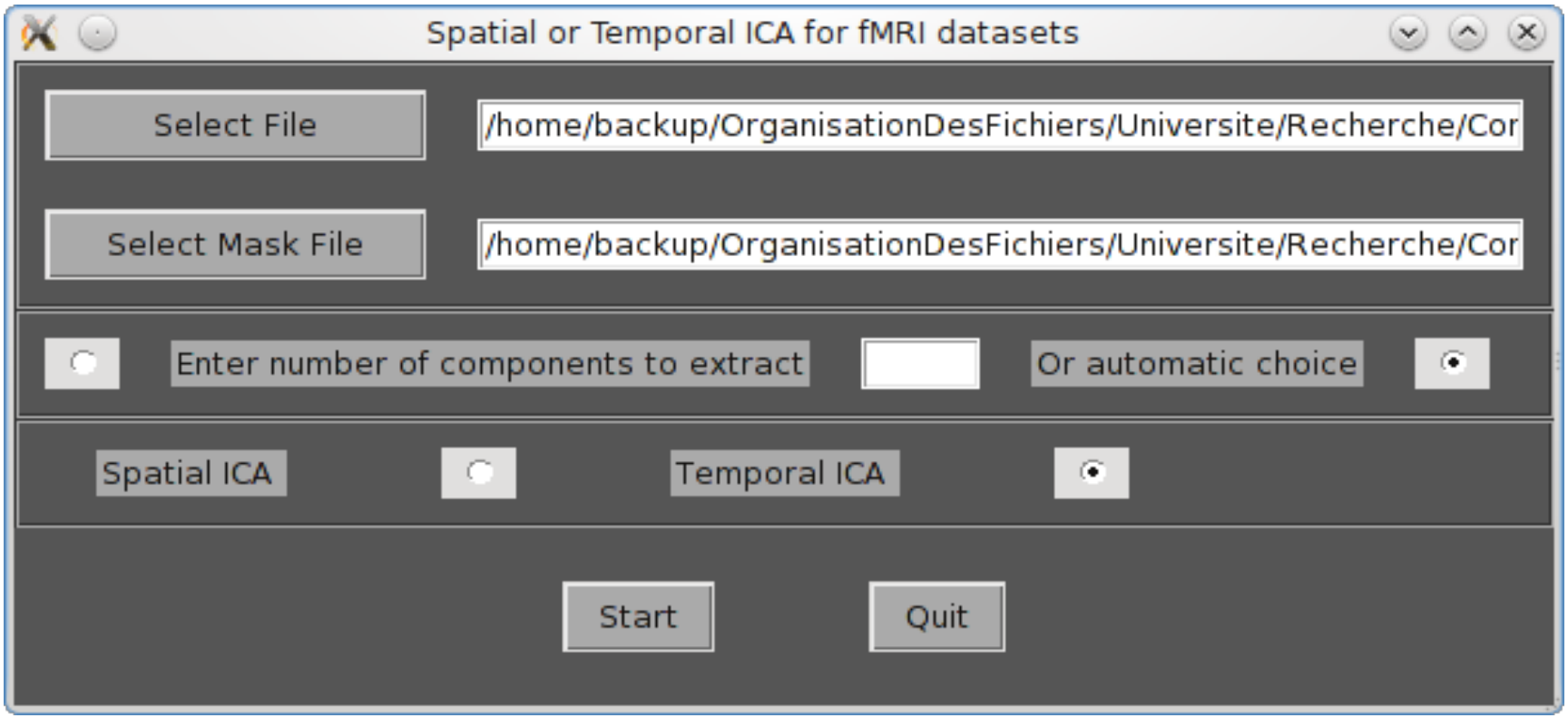
# R functions to perform ICA

`ICAspat()` To perform a spatial ICA

`ICAtemp()` To perform a temporal ICA

`f.icast.fmri.gui()` The GUI provides a quick and easy to use interface for applying spatial or temporal ICA to fMRI NIFTI datasets.

# Using `f.icast.fmri.gui()` GUI

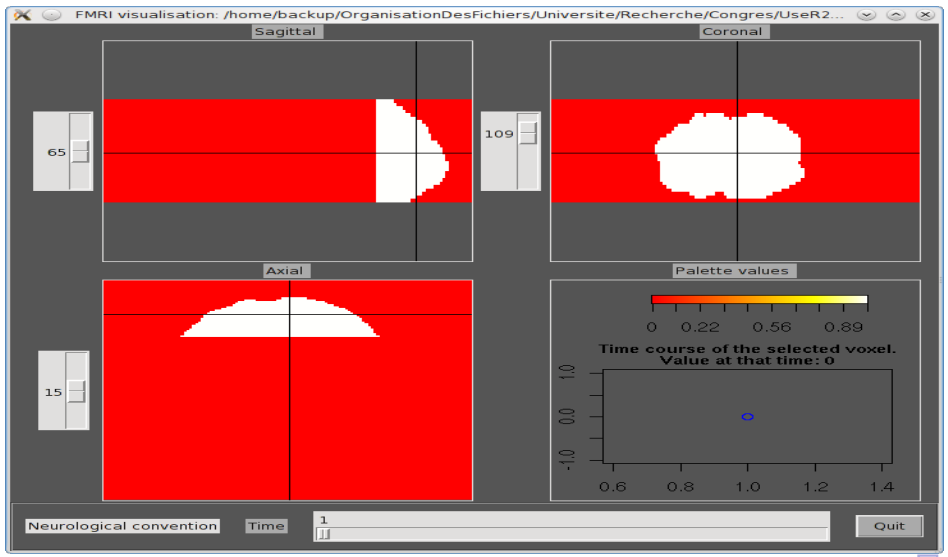


# Using `f.icast.fmri.gui()` GUI

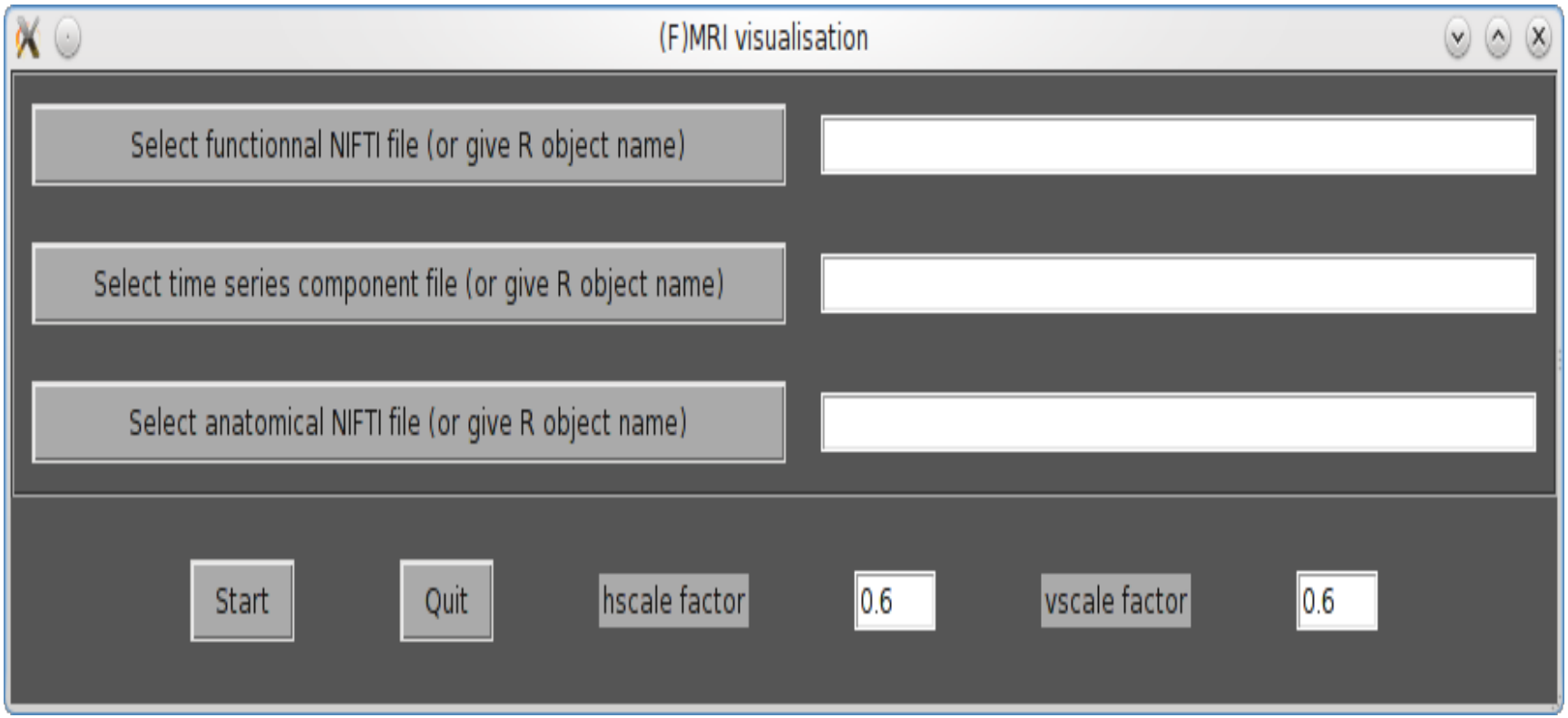
Here are the two files created if we choose to perform tICA :

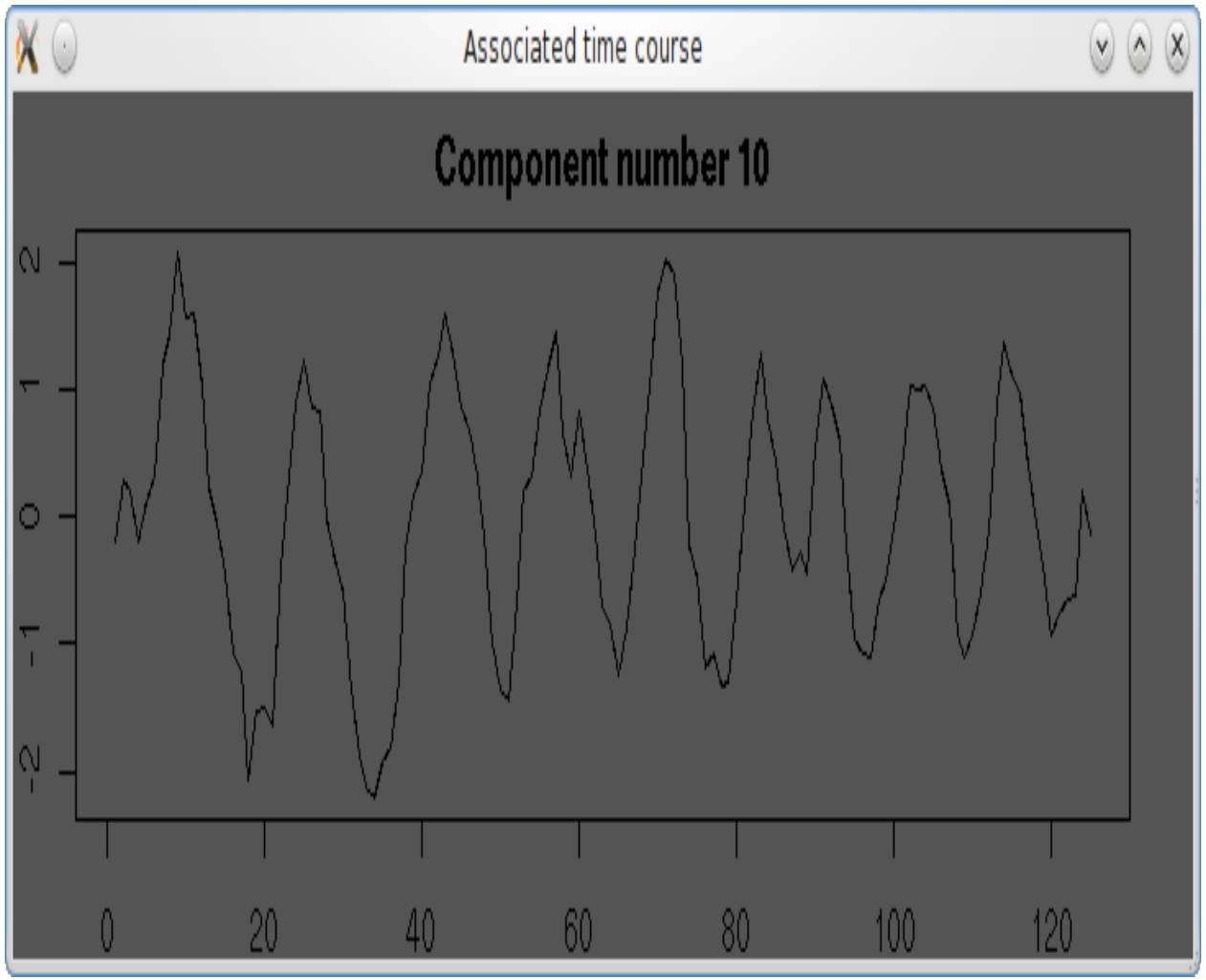
`map293_mond_ICAt.nii`
`map293_mond-ICAt-time-series.dat`

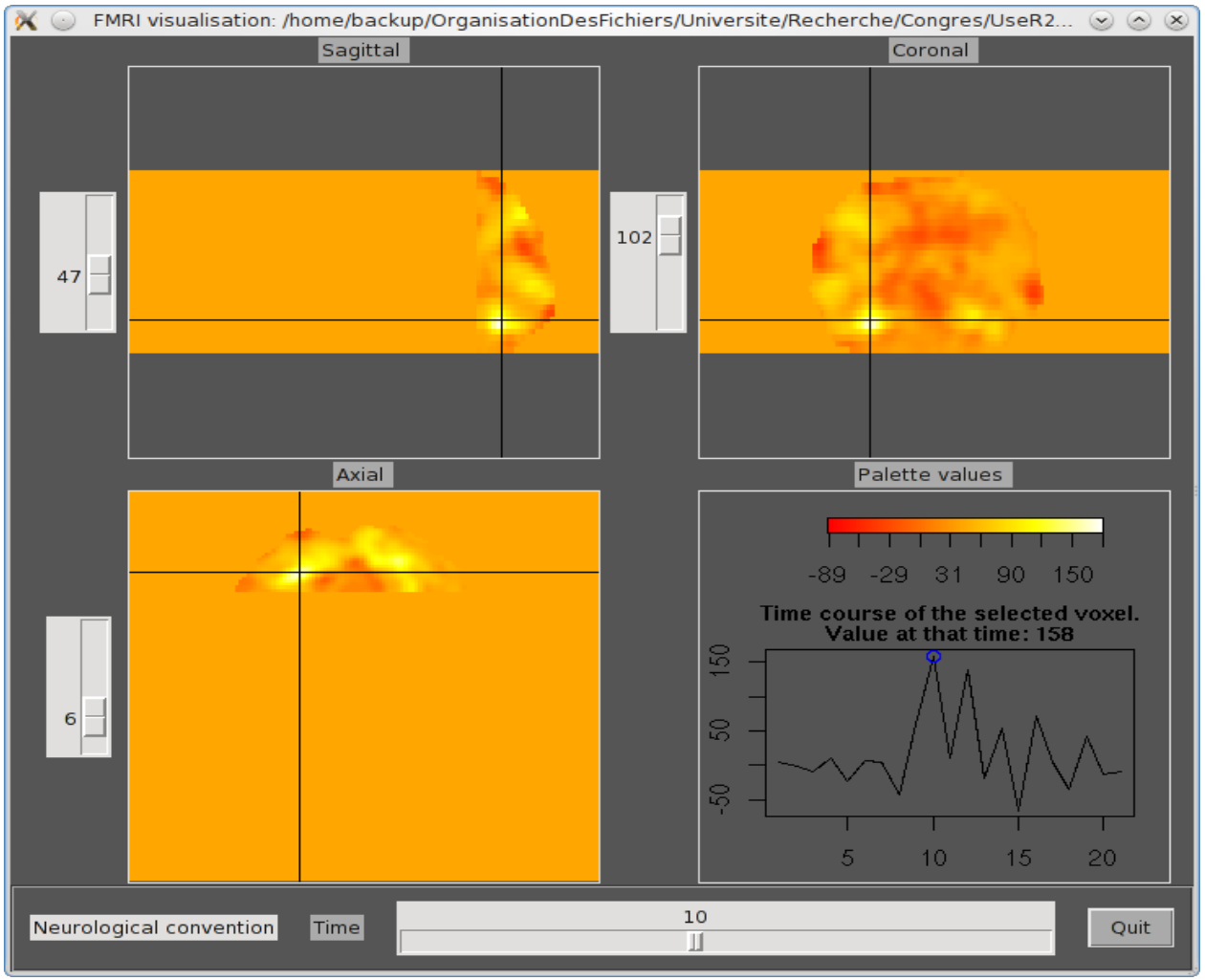
Computation time : less than 1mn with this mask/laptop.

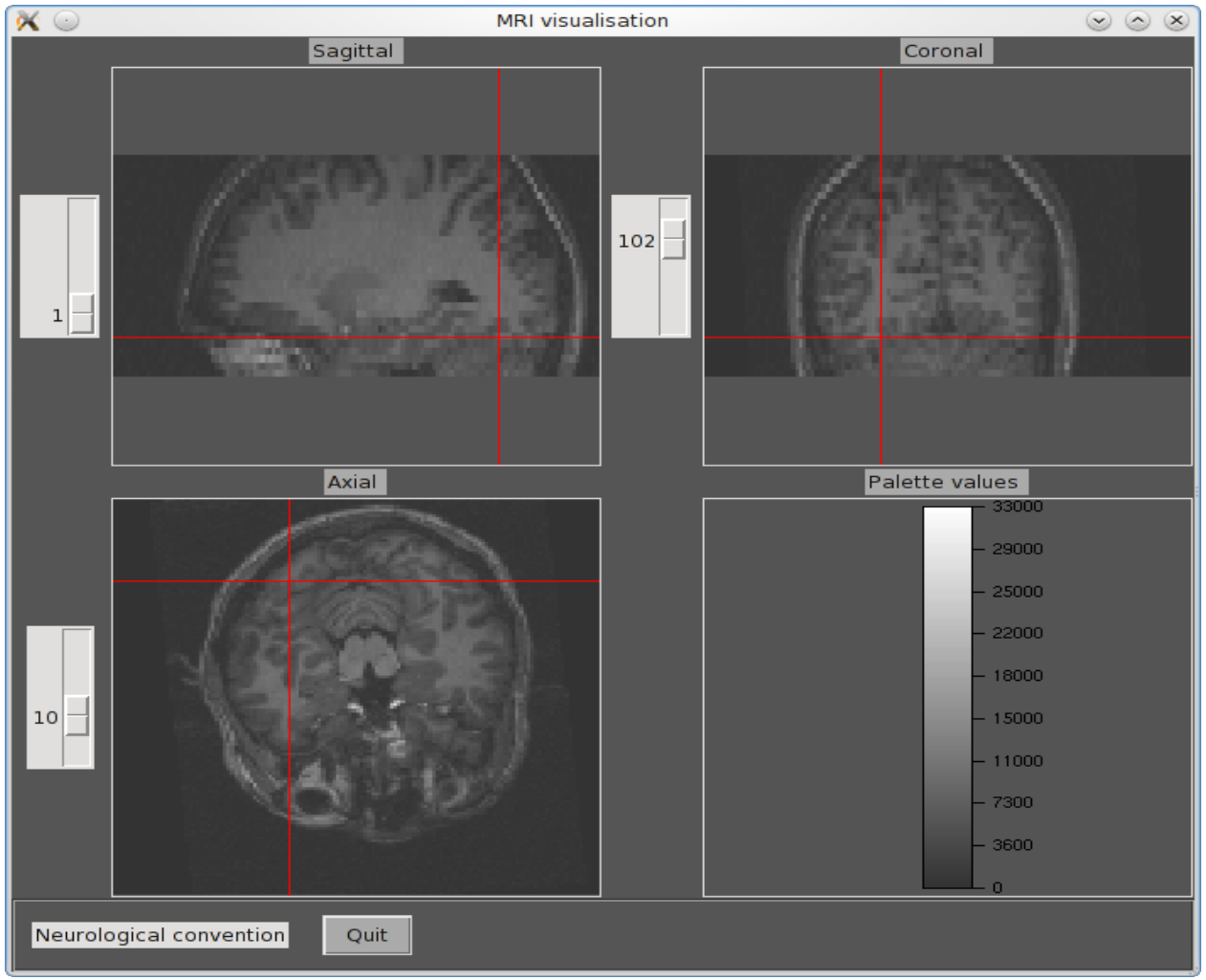


# Visualization of images using `f.plot.volume.gui()`




Associated time course
Component number 10
2
1
0
-1
-2
0
20
40
60
80
100
120


FMRI visualisation: /home/backup/OrganisationDesFichiers/Universite/Recherche/Congres/UseR2...
Sagittal
Coronal
47
102
Axial
Palette values
-89 -29 31 90 150
Time course of the selected voxel.
Value at that time: 158
150
50
-50
5 10 15 20
6
Neurological convention
Time
10
Quit


MRI visualisation
Sagittal
Coronal
1
102
Axial
Palette values
10
33000
29000
25000
22000
18000
15000
11000
7300
3600
0
Neurological convention
Quit

# Thank you for your attention