Modeling spontaneous brain activity in Python
Scientific progress and software challenges

Gaël Varoquaux, INRIA and Neurospin
Spontaneous brain activity
Why study spontaneous brain activity?

- Explains 90% of the signal in task-driven experiments
- Window on intrinsic brain architecture
- Unique biomarker to study brain pathologies on impaired patients
Scientific challenge
To develop in collaboration with neuroscientists new statistical tools to learn probabilistic models of spontaneous brain activity
1. Spatial patterns of brain activity
2. Beyond activation maps
3. Inter-subject comparisons
4. From models to software tools?
Spatial patterns of brain activity
Conventional brain mapping

- Study of stimuli response
- Mass-univariate statistics:
  for each voxel \( \mathbf{X} = \beta \mathbf{Y} + \mathbf{E} \)
- Group inference: subject-variability model on \( \beta \)
Conventional brain mapping – software

**Nipy:** NeuroImaging in Python
Berkeley, Stanford, Neurospin . . .

**Vision:** Open code shared between labs

**Progress:**
- Statistical models implemented 😊
- API difficult to use 😞
- Good Input/Output code 😊
- Preprocessing not implemented 😞

**Roadblocks:**
- Different teams ⇒ different visions
- Scientists can’t justify time on “solved problems”
Spatial correlation maps of spontaneous activity

- Biswal 1995: strong correlation between activity in left and right motor cortex at rest

- Later: seed-based correlation mapping

The human brain is intrinsically organized into dynamic, anticorrelated functional networks (Fox 2005)

How many? How to choose seeds? 😞
Independent component analysis

\[ B = \mathbf{M} \cdot S \]

- **B**: observed images
- **M**: mixing matrix
- **S**: sources

Minimize mutual information between patterns \( S \).
Independent component analysis

\[ B = M \cdot S \]

- \( B \): observed images
- \( M \): mixing matrix
- \( S \): sources

Minimize mutual information between patterns \( S \).

No noise model \( \Rightarrow \) Lack of reproducibility + Fits noise
Independent component analysis

\[ \text{sources} \ 
\begin{array}{c}
\text{voxels} \\
\text{B} \\
\text{=} \\
\text{[M]} \cdot \\
\text{S} \\
\text{=} \\
\text{sources} \\
\text{voxels}
\end{array} \ 
\]

- **B**: observed images
- **M**: mixing matrix
- **S**: sources

Minimize mutual information between patterns \( S \).

No noise model

⇒ Lack of reproducibility + Fits noise 😞
Model subject-to-subject variability

\[ A_1 \neq A_2 \]

Multivariate random effects model:

\[ Y_s = \text{loadings} \times P_s + \text{intra-subject noise} \]

\[ \{ P_s \} = \text{loadings} \times B + \text{inter-subject variability} \]

\[ B = M \times A \]

\[ \Rightarrow \text{Group-level networks} \]
Model subject-to-subject variability

Reproducibility across random groups

<table>
<thead>
<tr>
<th></th>
<th>no CCA</th>
<th>CCA + ICA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subspace</td>
<td>.36 (.02)</td>
<td>.71 (.01)</td>
</tr>
<tr>
<td>One-to-one</td>
<td>.36 (.02)</td>
<td>.72 (.05)</td>
</tr>
</tbody>
</table>

Varoquaux, NeuroImage 2010
Efficient Python implementation (CanICA)

Problem to solve:

1. \( \mathbf{Y}_s = \text{loadings} \times \mathbf{P}_s + \ldots \)  
   - PCA: SVD

2. \( \{\mathbf{P}_s\} = \text{loadings} \times \mathbf{B} + \ldots \)  
   - CCA: SVD

3. \( \mathbf{B} = \mathbf{M} \times \mathbf{A} \)  
   - ICA: iterations

\( \vdash \text{Recomputed many times across random groups} \)

Step 2 and 3: Small data size  \( \Rightarrow \) not bottleneck

Step 1: Independent problems per subject  
\( \Rightarrow \) Parallel runs and caching of the results

Joblib: Python functions as pipeline jobs

Goals: remove dataflow and persistence problems from algorithmic code
Spatial patterns of brain activity

- New algorithms for spatial decomposition of spontaneous activity with explicit model of group-variability
- Separation of concerns in code: algorithms ≠ dataflow
Beyond activation maps
Segmenting sparse regions

\[ \text{sources} \quad \text{voxels} \quad B \quad = \quad [M] \cdot \text{sources} \quad \text{voxels} \quad S \]
Segmenting sparse regions

\[ B = \mathbf{M} \cdot \mathbf{S} \]

Interesting sources \( S \) sparse

- \( Q \): Gaussian noise

⇒ Null hypothesis: centered normal distribution.
A full-brain parcellation

Visual system

- **V1**
  - map 0, reproducibility: 0.54
  - **V1**

- **V1-V2**
  - map 1, reproducibility: 0.52
  - **V1-V2**

- **extrastriate**
  - map 3, reproducibility: 0.47
  - **extrastriate**

- **superior parietal**
  - map 25, reproducibility: 0.34
  - **superior parietal**
A full-brain parcellation

Motor system

Map 4, reproducibility: 0.47

Map 21, reproducibility: 0.36

Map 32, reproducibility: 0.30
A full-brain parcellation
Between-regions connectivity

**Correlation matrix** \( \Sigma \)

- Default mode
- Task maintenance
- Executive control
- Ventral fronto-parietal
- Fronto-parietal control system
- Dorsal attentional
- Somato-motor
- Primary areas
- Visual
Change of representation
Understanding complex data requires interactive visualization with high level concepts

Mayavi: Python 3D visualization
Inter-subject comparisons
Ischemic stroke: ■ Temporary interruption of blood flow
■ Affects 1 person out of 100 every year for people > 55 years
■ Causes focal lesions of varying consequences

How does brain reorganize after stroke?

Prognostic based on intrinsic brain activity?
Probabilistic covariance modeling

Probabilistic model of data

- Covariance = $2^{nd}$ moment of observed data

⇒ Specifies a probability distribution

Test the likelihood of data in a covariance model
Probabilistic covariance modeling

Probabilistic model of data

- Covariance = 2\textsuperscript{nd} moment of observed data
- \Rightarrow \text{Specifies a probability distribution}

Test the likelihood of data in a covariance model

Covariances variations in healthy population

Which one of the above has a large cortical lesion?
Probabilistic model of data

- Covariance = 2\textsuperscript{nd} moment of observed data

⇒ Specifies a probability distribution

Test the likelihood of data in a covariance model

Covariances variations in healthy population

Which one of the above has a large cortical lesion?
Modeling variability of covariance
Modeling variability of covariance

Varoquaux, MICCAI 2010
Modeling variability of covariance

Varoquaux, MICCAI 2010
Modeling variability of covariance

\( \mathcal{P}(d\Sigma) \): probability density in tangent space

Varoquaux, MICCAI 2010
Modeling variability of covariance

\( P(d\Sigma) \): probability density in tangent space

**Controls**
- Patient

**Controls**
- Patient

**Patient**
- Controls

**Patient**
- Controls

Log-likelihood

Tangent space

controls

patients

Varoquaux, MICCAI 2010
Finding the cause of the difference

Between which regions is connectivity modified?

**Ill-posed problem**
- Non-local effects

⇒ Many differences causes give the same observations

**Our suggestion**
- Pair-wise partial correlations
- In tangent space: almost independent
- Draw random groups of healthy controls to tabulate their variability
Finding the cause of the difference
Research code in clinical settings

- Applications give rise to non-trivial mathematical problems
- Need to interact with neurologists
- Round-trips are costly: neurologists should use our code, modify our code
From models to software tools?
The hidden costs of releasing software

- Gap from paper to software:
  - Remove duplication
  - Write documentation
  - Make usable APIs
  - Write tests
  - Fix corner cases

Cost of code

- Complexity scales as the square of project size
  - Woodfield 1979, *an experiment on unit increase in problem complexity*

Cost of users

- Backward compatibility
- Support for multiple installations and versions
- Bug reports, feature request, mailing list support

Maintenance cost \( \sim (\# \text{ lines})^2 \sqrt{\# \text{ users}} \)
Better code

- High-level coding and abstractions
  - numpy arrays: abstract out memory and pointers
  - traits Model+View: hide dialogs and events
  - joblib: factor out persistence

- Common libraries
  - scipy, Mayavi, ... 

Project management decisions

- 80/20 rule
- Not every research code should be released
- Focus on documentation and installation
Software as building blocks for new science

Segregated, functionally-specialized, packages
- Answer a specific problem
- Limit dependencies

Reusable projects
- Useful for a different purpose than the original one
- Libraries (no control of point of entry)
- Standard data structures
- Most often simple
- BSD licensed
4 Mayavi: making 3D visualization reusable

Pipelines: from data sources to visualization objects

- Simple API: `mlab.contour3d(x, y, z, data)`

- Building pipelines by function calls: `mlab.pipeline.iso_surface(mlab.pipeline.contour(src))`

- GUI + automatic script generation
Mayavi: making 3D visualization reusable

260 lines of code!
Mayavi: making 3D visualization reusable

- 260 lines of code!
- All dialogs are components: we expose our internals
- Visualizations included Traits view
- Easy update of data
Dataflow pipeline: *succession of processing steps executed on demand*

**joblib:**
- Lazy-revaluation
- Persistence
- Parallel processing
- Logging

*All with functions (seemingly)*
from joblib import Memory

mem = Memory(cachedir='~/tmp/joblib')

import numpy as np

a = np.vander(np.arange(3))
square = mem.cache(np.square)
b = square(a)

d = square(a)

# The above call did not trigger an evaluation
Towards Quantitative modeling of spontaneous brain activity

- Requires probabilistic models and state-the-art machine learning tools
- Algorithms and software development hand in hand with neurologists for applications
- Need a high-level stack of software tools general purpose with separation of concerns