Streamlined neuroimaging analysis with enhanced infrastructure and surface API integration

DOI 10.5281/zenodo.8397156 RRID SCR_001362

Yasmin Mzayek, Himanshu Aggarwal, Pierre Bellec, Ahmad Chamma, Alexandre Cionca, Jelle Roelof, Dalenberg, Jérôme Dockès, Mathieu Dugré, Elizabeth DuPre, Rémi Gau, Nicolas Gensollen, Mathias Goncalves, Anne-Sophie Kieslinger, Alisha Kodibagkar, Steven Meisler, François Paugam, Julio A. Peraza, Jean-Baptiste Poline, Patrick Sadil, Taylor Salo, Kevin Sitek, Maximilian Cosmo Sitter, Alexis Thual, Mohammad Torabi, Konrad Wagstyl, Hao-Ting Wang, Michelle Wang, Bertrand Thirion and Nilearn contributors

What is Nilearn ?

- Python package for analysis of brain images
  - Connectivity analysis (resting-state)
  - Decoding (MVPA)
  - GLM (stats)
  - Plotting volumetric and surface data
- Well documented
- Supportive community
- Open-source and community driven

Releases 0.10.2 - 0.10.4

- support for CompCor confounds from fmriprep
- easily load fmriprep confounds for GLM analysis
- improved HTML reports
- fixes for many old (and new) bugs
- improved codebase (formating, linting, automation)

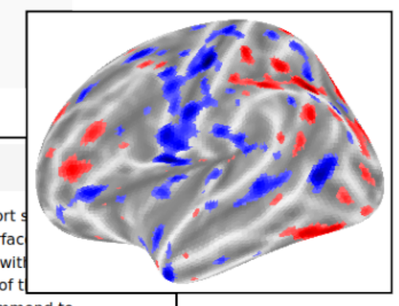
New surface API

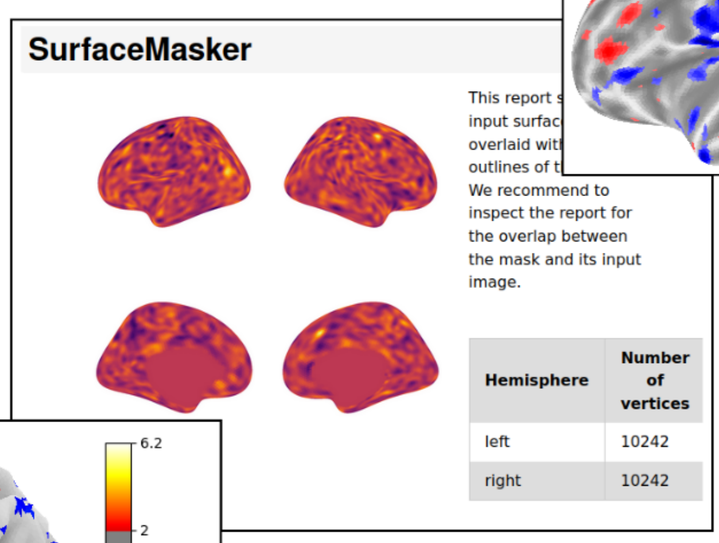
```
# experimental module with surface datasets
>>> from nilearn.experimental import surface
>>> surf_time_series = surface.fetch_nki()[0]
>>> print(f"NKI image: {surf_time_series}")
NKI image: <SurfaceImage (895, 20484)>
```

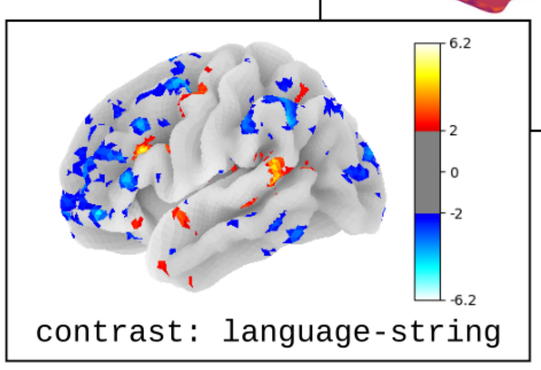
```
# mesh and data in the same surface object
>>> print(surf_time_series.data.parts)
{'left': array([[-0.4938, ...] ...],
       dtype=object),
 'right': array([[-0.5474, ...] ...],
        dtype=object)}

>>> print(surf_time_series.mesh.parts)
{'left': <FileMesh with 10242 vertices>,
 'right': <FileMesh with 10242 vertices>}
```

```
# surface maskers to perform signal extraction
>>> masker = surface.SurfaceMasker()
>>> data = masker.fit_transform(surf_time_series)
>>> print(f"Masked data shape: {data.shape}")
Masked data shape: (895, 20484)

>>> first_data = data[0]
>>> first_img = masker.inverse_transform(first_data)
>>> print(f"First timepoint: {first_data}")
First timepoint: <SurfaceImage (20484,)>
```

```
# easy plotting and reporting
>>> from nilearn.experimental import plotting
>>> plotting.plot_surf(first_img, ...)
>>> report = masker.generate_report()
```








```
# quickly run GLM on surface data
>>> from nilearn.glm.first_level import FirstLevelModel
>>> glm = FirstLevelModel(t_r).fit(surface_time_series,
                                   events)
>>> z_scores = glm.compute_contrast("language-string")
>>> plotting.plot_surf_stat_map(stat_map=z_scores,
                                hemi="left", ...)
```

Future directions

- Run decoding directly on surfaces
- Input / output for surface data
- Improved user-guide

Join the community!!!

- Check the documentation
  nilearn.github.io
- Ask questions on
  neurostars.org/tag/nilearn
- Contribute GitHub
  github.com/nilearn/nilearn
- Weekly drop-in hour, Wednesday 4pm UTC
  meet.jit.si/nilearn-drop-in-hours