# ORGANISATION DU DÉVELOPPEMENT DE SCIKIT-LEARN, NILEARN ET JOBLIB

LOÏC ESTÈVE

# PLAN

- Présentation générale des trois projets
- Présentation des outils
- Aspects documentation et gestion de communauté

# SCIKIT-LEARN

Rendre accessible le machine learning pour les non-experts

```
from sklearn import SVM

classifier = svm.SVC()
classifier.fit(X_train, y_train)
y_test = classifier.predict(X_test)
```

Package agnostique par rapport au domaine d'application

~20 core contributors, 550 total, 2000 github emails last month

# JOBLIB

Calcul parallèle (joblib.Parallel)

```
from math import sqrt
from joblib import Parallel, delayed
Parallel(n_jobs=2)(delayed(sqrt)(i ** 2) for i in range(10))
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0]
```

Caching (joblib.Memory)

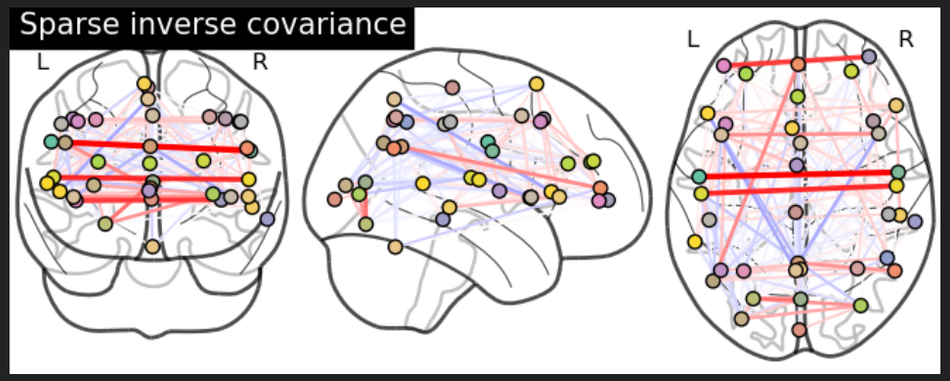
```
from joblib import Memory

mem = Memory(cachedir='.')

cached_func = mem.cache(func)
cached_func(a)  # computes func using 'a' as input
cached_func(a)  # fetches result from store
```

~5 core contributors, 40 total, ~150 github emails last month

# NILEARN




machine learning pour la neuroimagerie en python

~5-10 core contributors, ~40 total, 550 github emails last month

# PROCESSUS

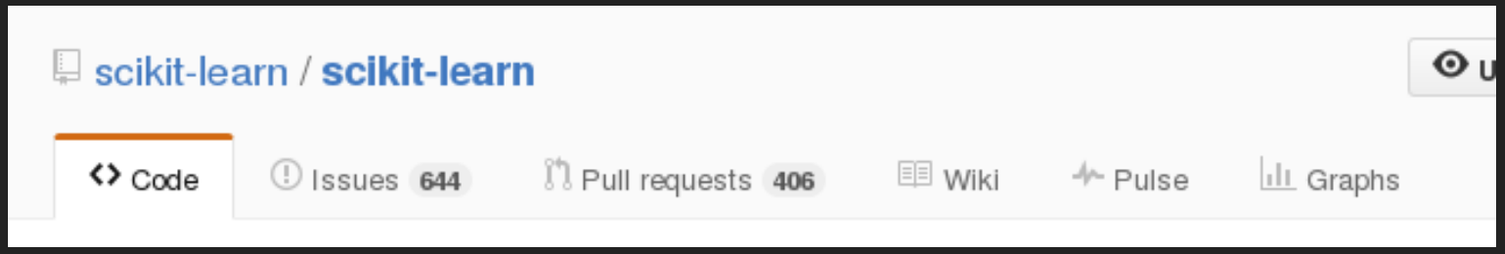
- github PR (convention [MRG] in the title) a besoin de l'accord de deux core développeurs avant d'être `mergée
- documentation + test
- Conventions de style (PEP8 entre autre), d'API
- cycle de déprécation de deux releases
- release majeure tous les 6 à 12 mois

# OUTILS

- github
- nosetests + coverage pour tests unitaires
- Integration continue : Travis/AppVeyor/CircleCI

# DOCUMENTATION

# GESTION DE COMMUNAUTÉ




- bien définir les limites du projects: règle du 20/80, bleeding edge vs boring
- contrôler la technicité du projet pour baisser la barrière de contribution
- sprints: recrutement de nouveaux collaborateurs, réunir physiquement des core developers pour discussion des directions plus à long terme

# QUESTIONS ?