Yale

Runtime Verification of Scientific Software

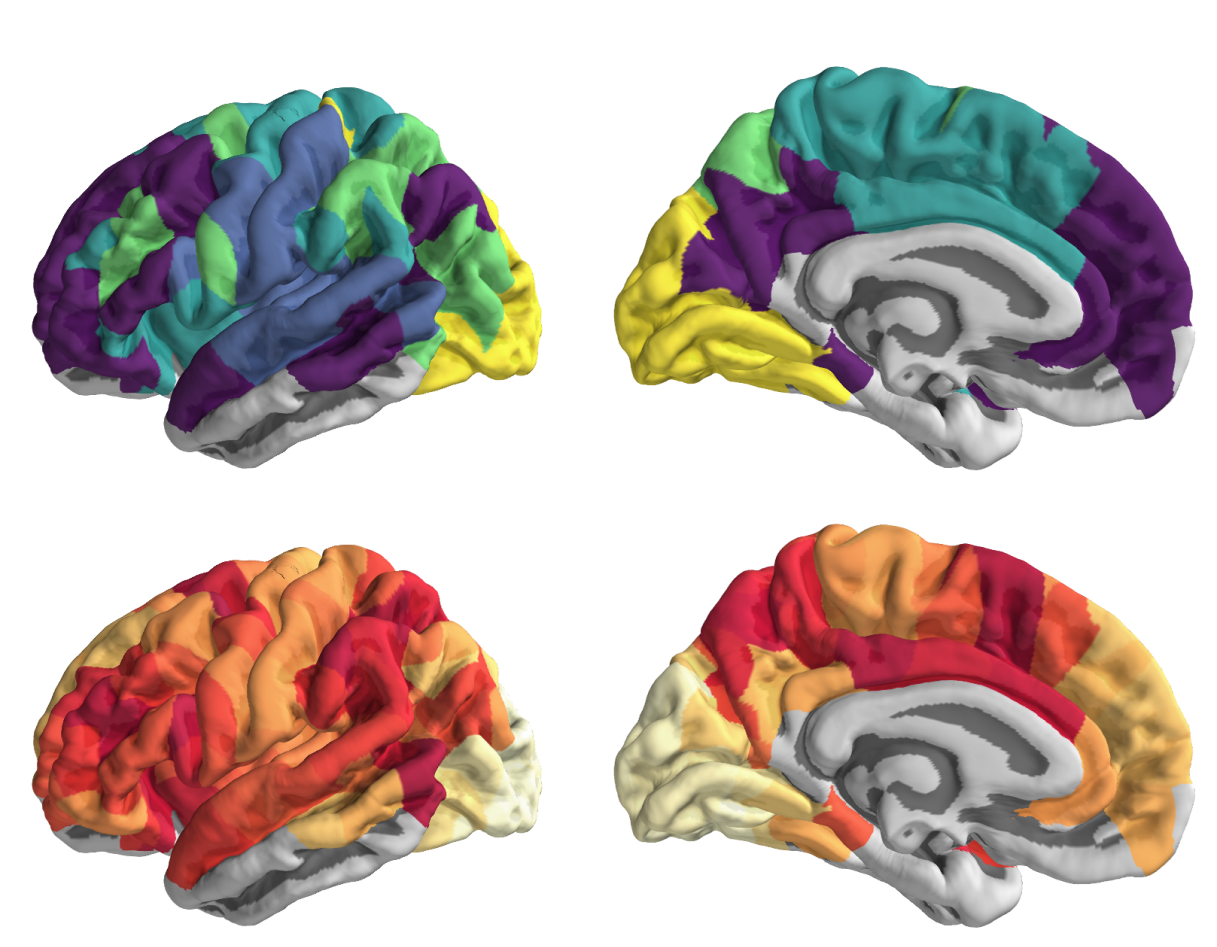
Maxwell Shinn, Clarence Lehman, and Ruzica Piskac

October 31, 2018

Motivation

```
def graph_measure(filename):
  timeseries = load_from_csv(filename)
  corr_matrix = corr_coef(timeseries)
  normalized = fisher_transform(corr_matrix)
  G = matrix_to_graph(normalized)
  return graph_clustering(G)
```

Expected result



Actual result

Motivation

```
def graph_measure(filename):
  timeseries = load_from_csv(filename)
  corr_matrix = corr_coef(timeseries)
  normalized = fisher_transform(corr_matrix)
  G = matrix_to_graph(normalized)
  return graph_clustering(G)
```

Motivation

```
def graph_measure(filename):
  timeseries = load_from_csv(filename)
  corr_matrix = corr_coef(timeseries) diag = 1+1e-10
  normalized = fisher_transform(corr_matrix)
  G = matrix_to_graph(normalized)
  return graph_clustering(G)
```

Motivation

```
def graph_measure(filename):
  timeseries = load_from_csv(filename)
  corr_matrix = corr_coef(timeseries) diag = 1+1e-10
  normalized = fisher_transform(corr_matrix) = NaN
  G = matrix_to_graph(normalized)
  return graph_clustering(G)
```

Motivation

```
def graph_measure(filename):
  timeseries = load_from_csv(filename)
  corr_matrix = corr_coef(timeseries) diag = 1+1e-10
  normalized = fisher_transform(corr_matrix) = NaN
  G = matrix_to_graph(normalized) NaN → 0
  return graph_clustering(G)
```

Motivation

```
def graph_measure(filename):
  timeseries = load_from_csv(filename)
  corr_matrix = corr_coef(timeseries) diag = 1+1e-10
  normalized = fisher_transform(corr_matrix) = NaN
  G = matrix_to_graph(normalized) NaN → 0
  return graph_clustering(G) ???
```

Our tool

Our tool

- In scientific software, we don't know what the output should be

Our tool

- In scientific software, we don't know what the output should be
- Python library for scientific software

Our tool

- In scientific software, we don't know what the output should be
- Python library for scientific software
- Checks entry and exit conditions

Our tool

- In scientific software, we don't know what the output should be
- Python library for scientific software
- Checks entry and exit conditions
- Conditions specified with refinement types

Our tool

- In scientific software, we don't know what the output should be
- Python library for scientific software
- Checks entry and exit conditions
- Conditions specified with refinement types
- Specification serves as documentation

Our tool

- In scientific software, we don't know what the output should be
- Python library for scientific software
- Checks entry and exit conditions
- Conditions specified with refinement types
- Specification serves as documentation
- Automated offline testing