Learning SciPy for Numerical and Scientific Computing

Francisco Blanco-Silva

University of South Carolina

Some Basic Principles in Scientific Computing

A personal perspective

Scientific Computing is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

Some Basic Principles in Scientific Computing

A personal perspective

Scientific Computing is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- Trust the Mathematicians: High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!

Some Basic Principles in Scientific Computing

A personal perspective

Scientific Computing is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- Trust the Mathematicians: High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!
- Trust the Engineer, Biologist, Chemist, Physicist, ...: There are different ways to solve any problem. Rather than dismiss a different point of view, embrace it, work it out, explore the source of the problem, and look for connections with other techniques.

Some Basic Principles in Scientific Computing

A personal perspective

Scientific Computing is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- Trust the Mathematicians: High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!
- Trust the Engineer, Biologist, Chemist, Physicist, ...: There are different ways to solve any problem. Rather than dismiss a different point of view, embrace it, work it out, explore the source of the problem, and look for connections with other techniques.
- Trust the Computer Scientist: Writing low-level code from scratch seldom guarantees best results.

Some Basic Principles in Scientific Computing

A personal perspective

Scientific Computing is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- Trust the Mathematicians: High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!
- Trust the Engineer, Biologist, Chemist, Physicist, ...: There are different ways to solve any problem. Rather than dismiss a different point of view, embrace it, work it out, explore the source of the problem, and look for connections with other techniques.
- Trust the Computer Scientist: Writing low-level code from scratch seldom guarantees best results.
- Find a reliable way to communicate through software.

Some Basic Principles in Scientific Computing

A personal perspective

Scientific Computing is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- Trust the Mathematicians: High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!
- Trust the Engineer, Biologist, Chemist, Physicist, ...: There are different ways to solve any problem. Rather than dismiss a different point of view, embrace it, work it out, explore the source of the problem, and look for connections with other techniques.
- Trust the Computer Scientist: Writing low-level code from scratch seldom guarantees best results.
- Find a reliable way to communicate through software.
- Big guns: Solving these problems usually require massive amounts of calculations and are often executed on supercomputers or distributed computing platforms.

Computational Resources

Software

- Optimized Algebra Libraries
  - `BLAS`
  - `LAPACK`

Computational Resources

Software

- Optimized Algebra Libraries
  - `BLAS`
  - `LAPACK`
- Computer Algebra systems
  - `MATLAB`
  - `Mathematica`
  - `SciLab`
  - `GNU Octave`

Computational Resources

Software

- Optimized Algebra Libraries
  - `BLAS`
  - `LAPACK`
- Computer Algebra systems
  - `MATLAB`
  - `Mathematica`
  - `SciLab`
  - `GNU Octave`
- Computer Languages
  - `C`
  - `Fortran`
  - `Perl`
  - `R`
  - `julia`
  - `Python`

Computational Resources

Software

- Optimized Algebra Libraries
  - `BLAS`
  - `LAPACK`
- Computer Algebra systems
  - `MATLAB`
  - `Mathematica`
  - `SciLab`
  - `GNU Octave`
- Computer Languages
  - `C`
  - `Fortran`
  - `Perl` (with `PDL`)
  - `R`
  - `julia`
  - `Python`

Computational Resources

Software

- Optimized Algebra Libraries
  - BLAS
  - LAPACK
- Computer Algebra systems
  - MATLAB
  - Mathematica
  - SciLab
  - GNU Octave
- Computer Languages
  - C
  - Fortran
  - Perl (with PDL)
  - R
  - julia
  - Python
    - with PyLab: ipython + NumPy + **SciPy** + matplotlib

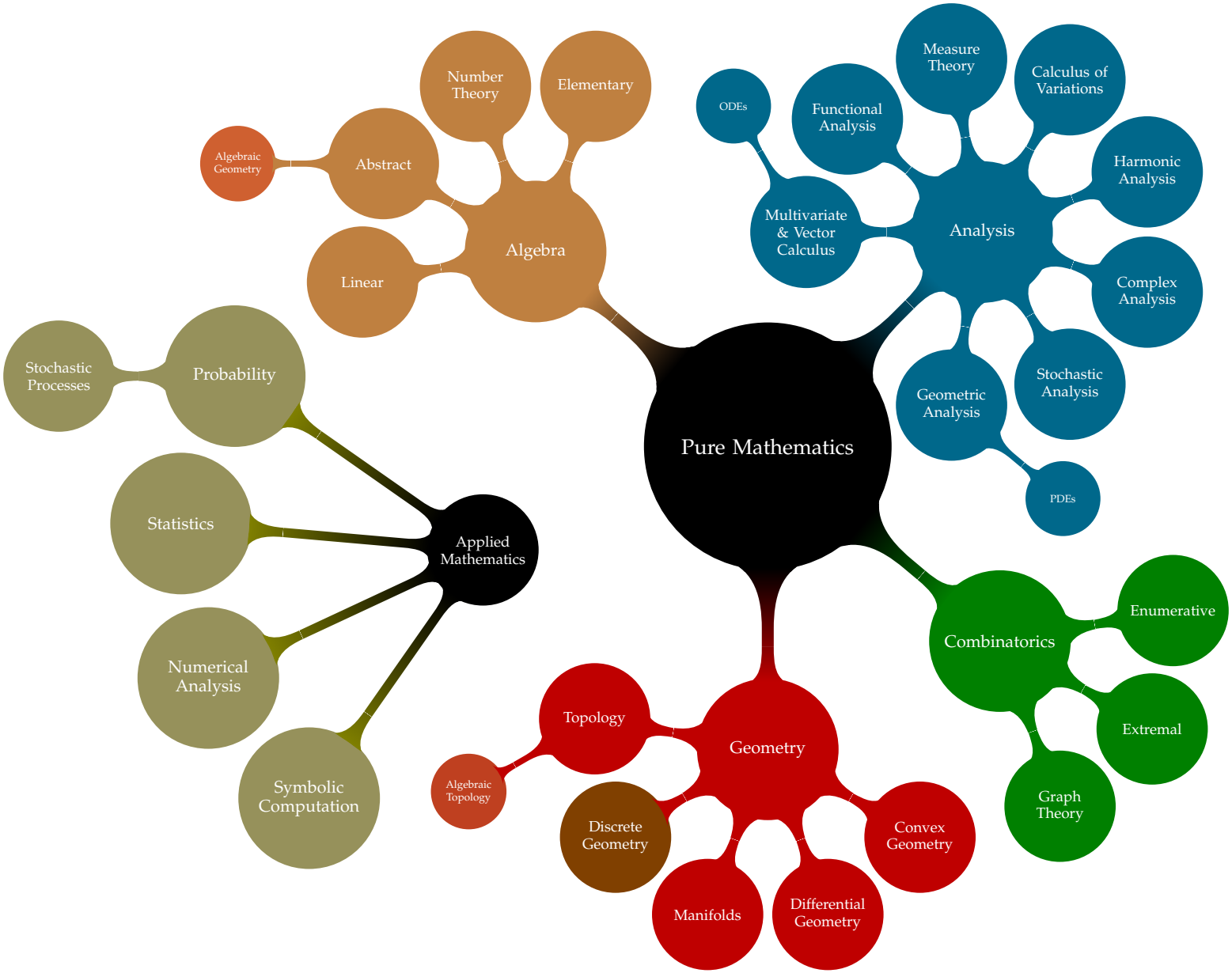
Computational Resources

Software

- Optimized Algebra Libraries
  - BLAS
  - LAPACK
- Computer Algebra systems
  - MATLAB
  - Mathematica
  - SciLab
  - GNU Octave
- Computer Languages
  - C
  - Fortran
  - Perl (with PDL)
  - R
  - julia
  - Python
    - with PyLab: ipython + NumPy + **SciPy** + matplotlib
    - with scikits and Pandas on top of that

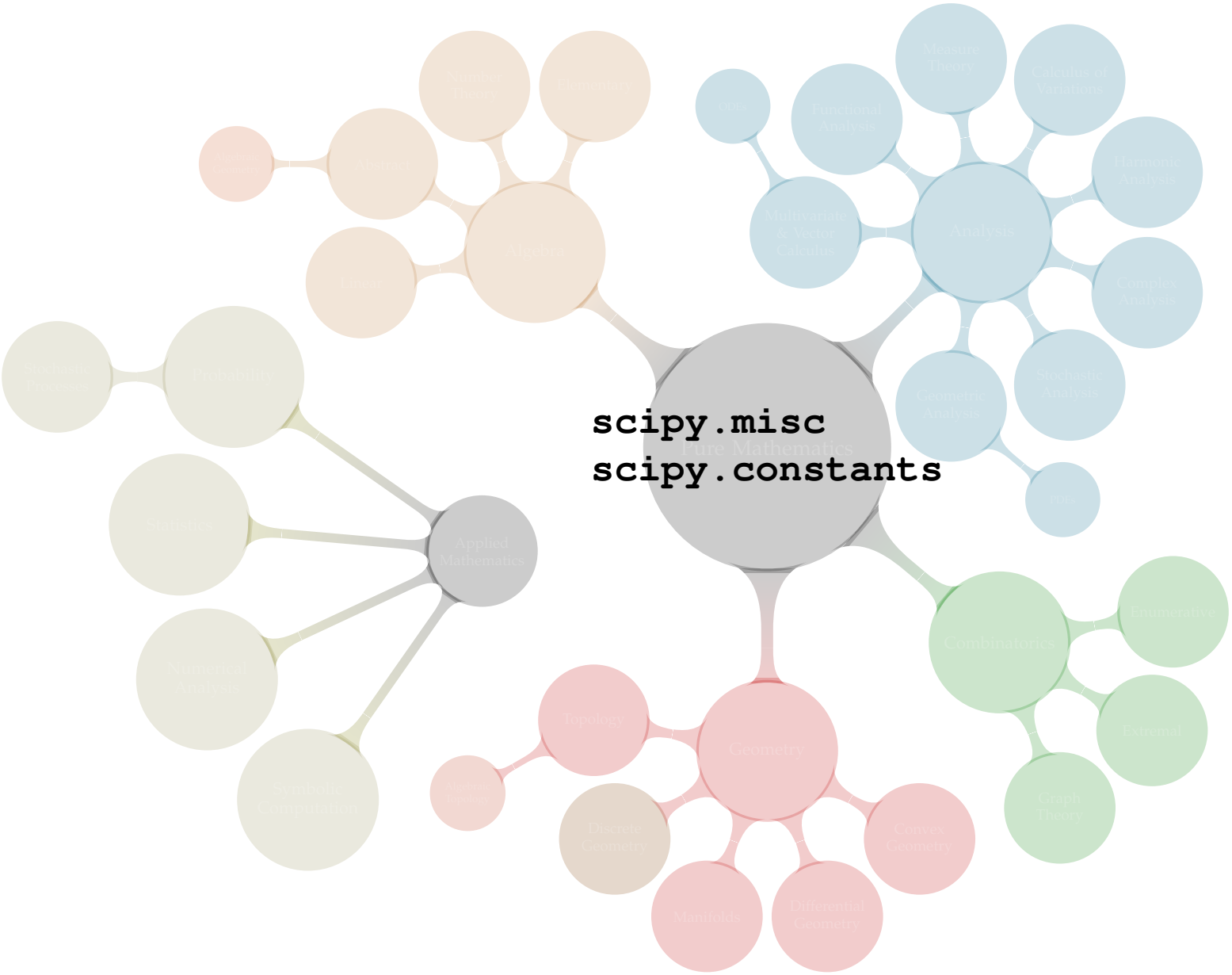
The Structure of SciPy

Similarity to the different areas of Mathematics



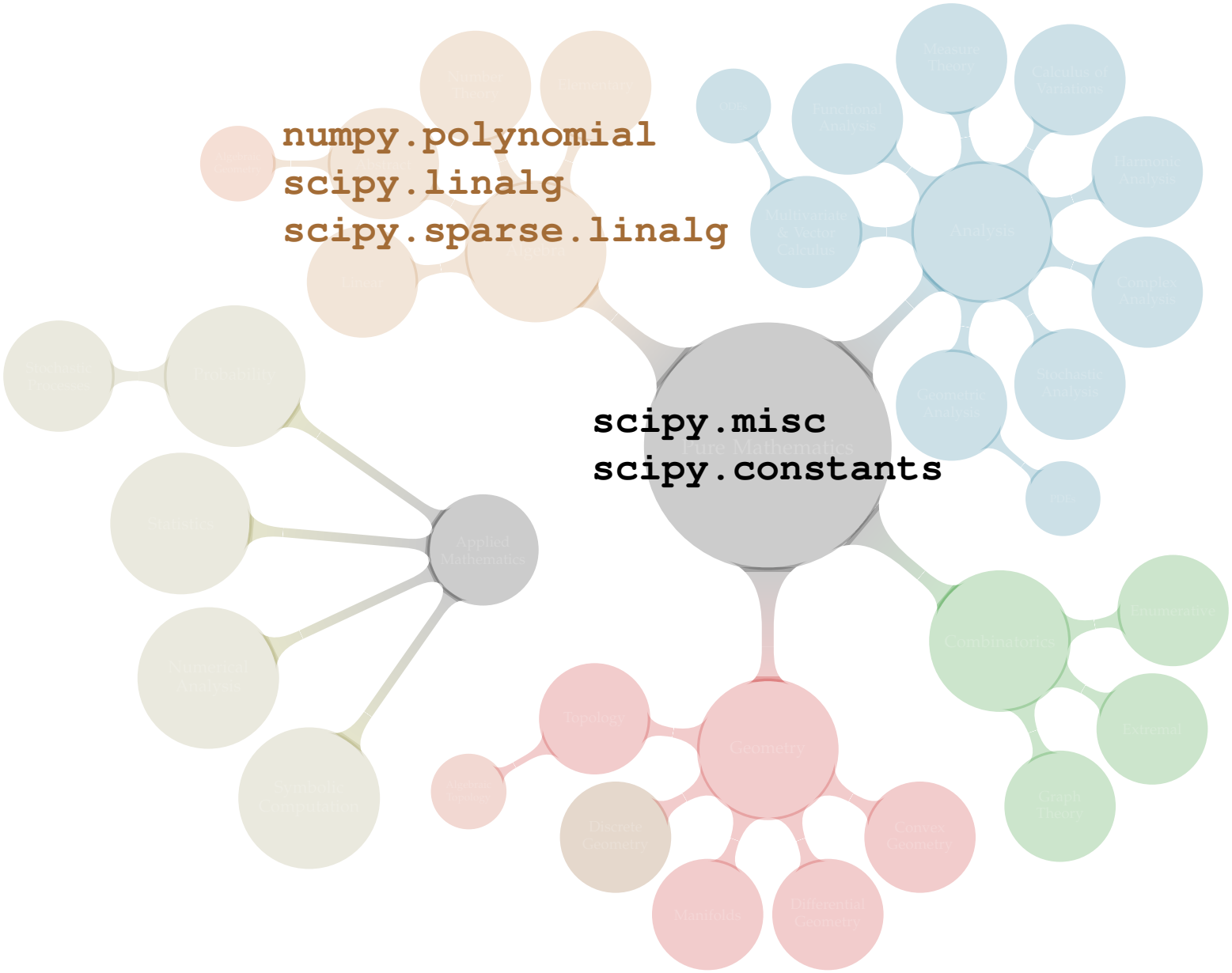
The Structure of SciPy

Similarity to the different areas of Mathematics



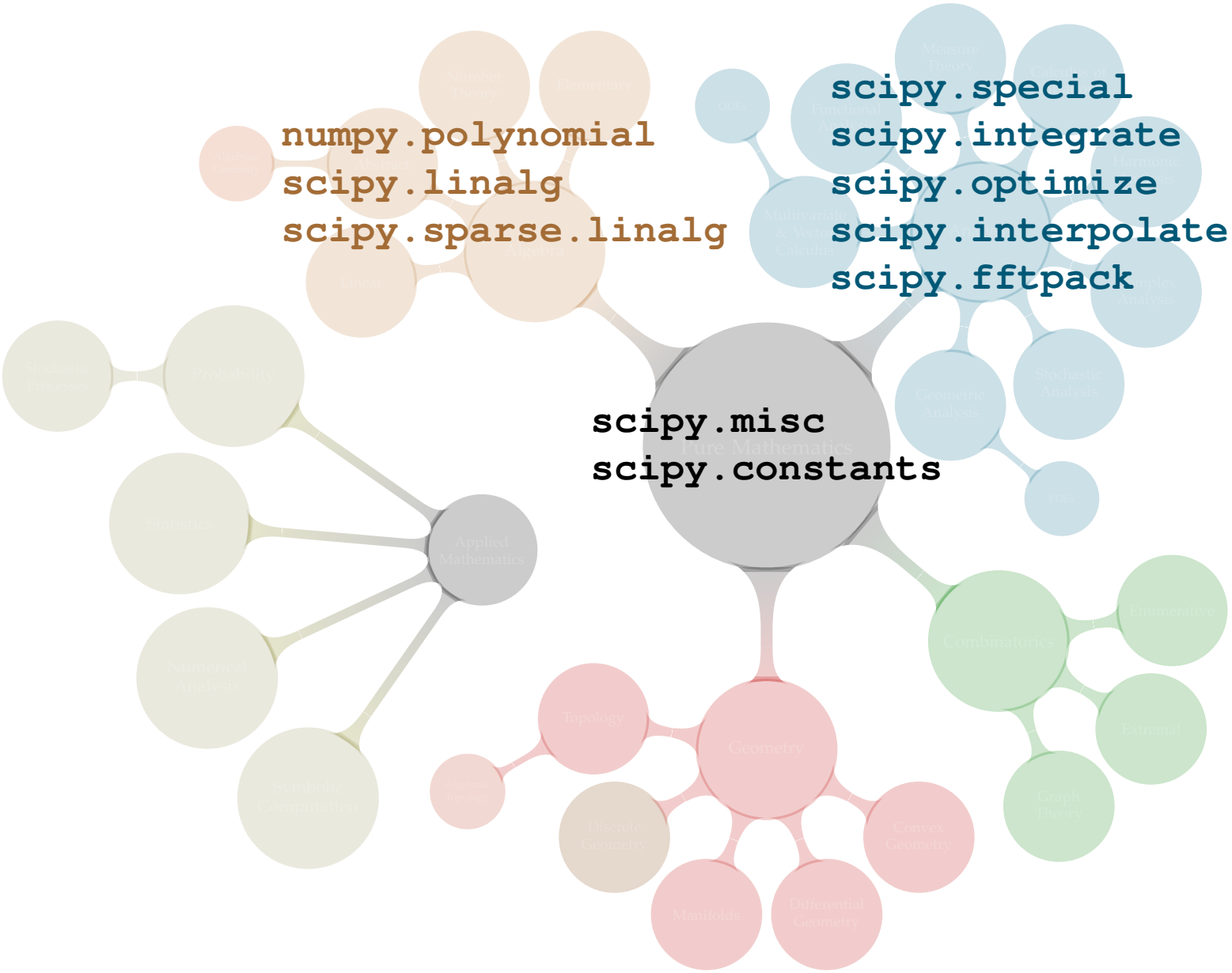
The Structure of SciPy

Similarity to the different areas of Mathematics



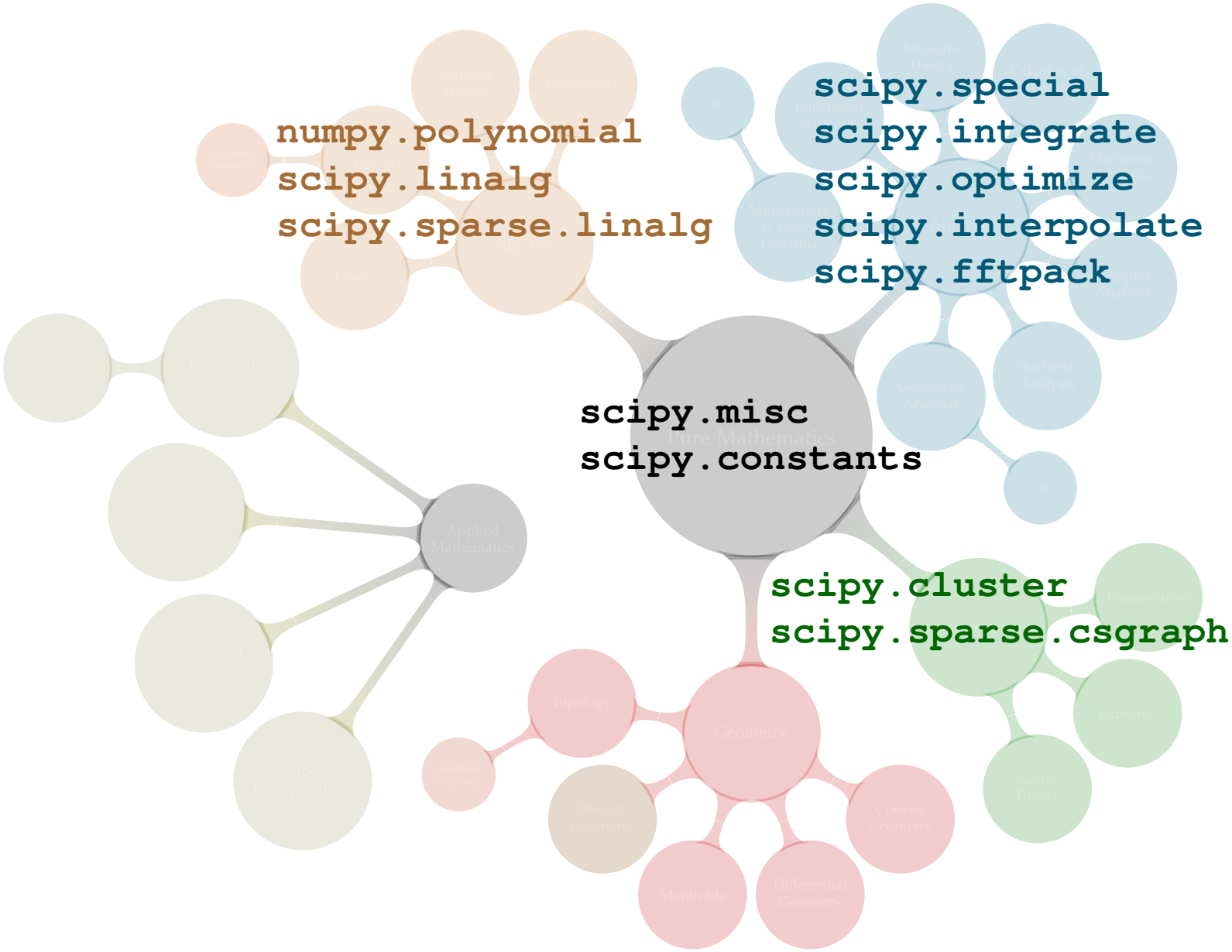
The Structure of SciPy

Similarity to the different areas of Mathematics



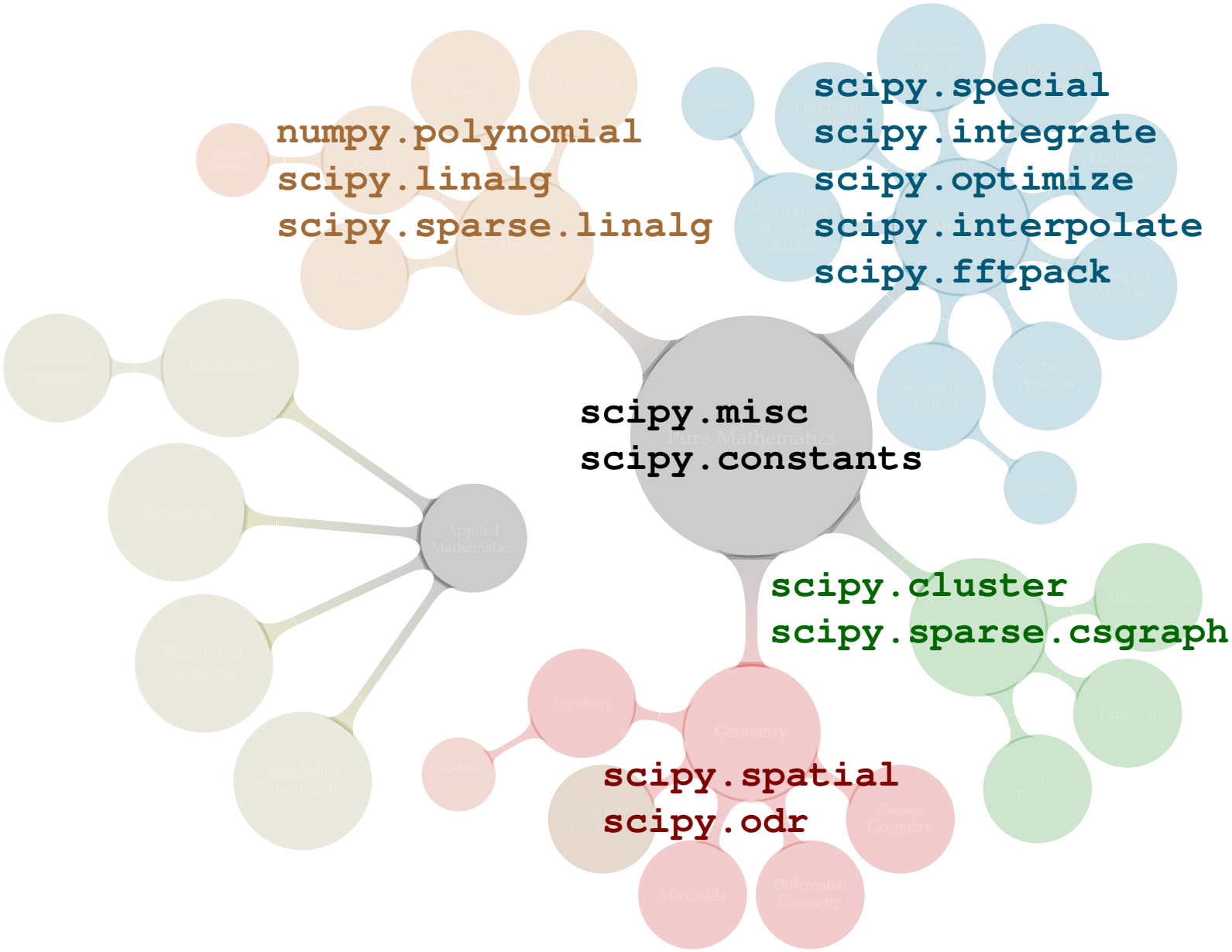
The Structure of SciPy

Similarity to the different areas of Mathematics



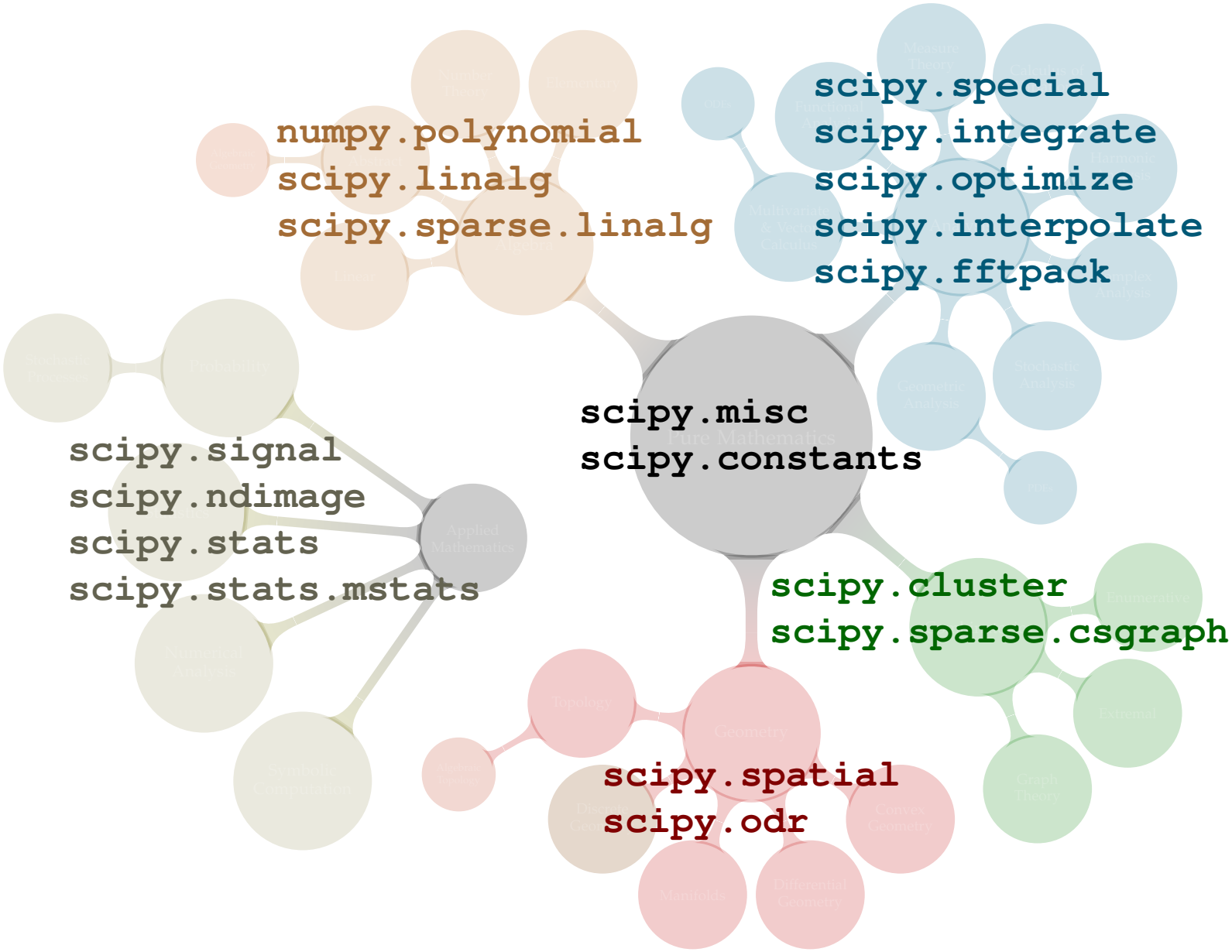
The Structure of SciPy

Similarity to the different areas of Mathematics



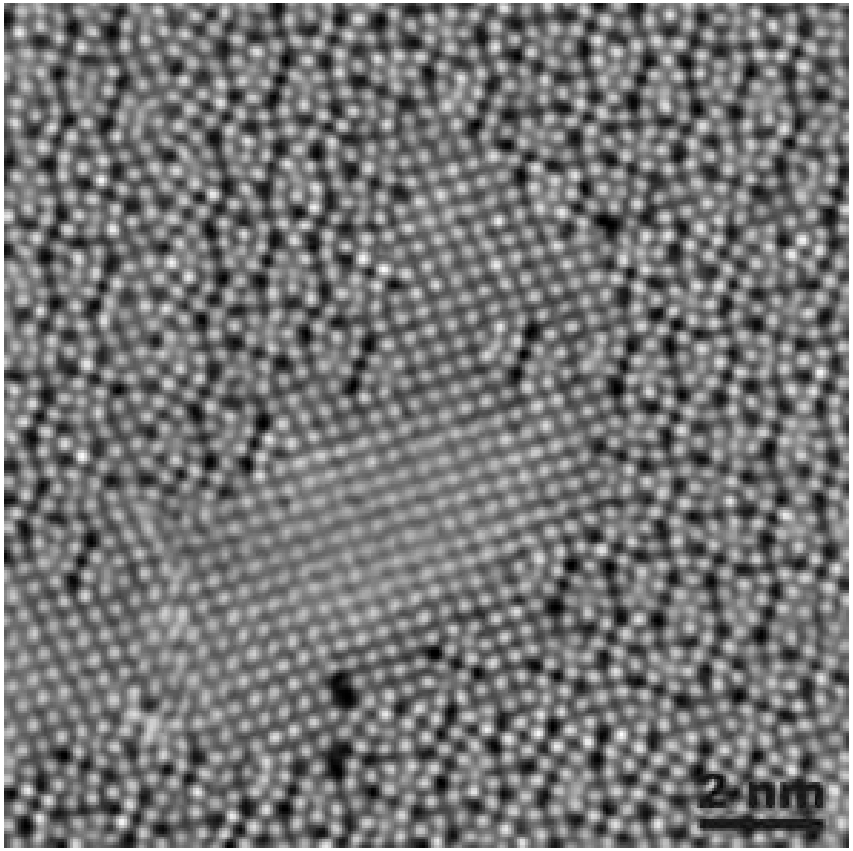
The Structure of SciPy

Similarity to the different areas of Mathematics



SciPy in action

Extract the structural model of a molecule of $Nb_4W_{13}O_{47}$

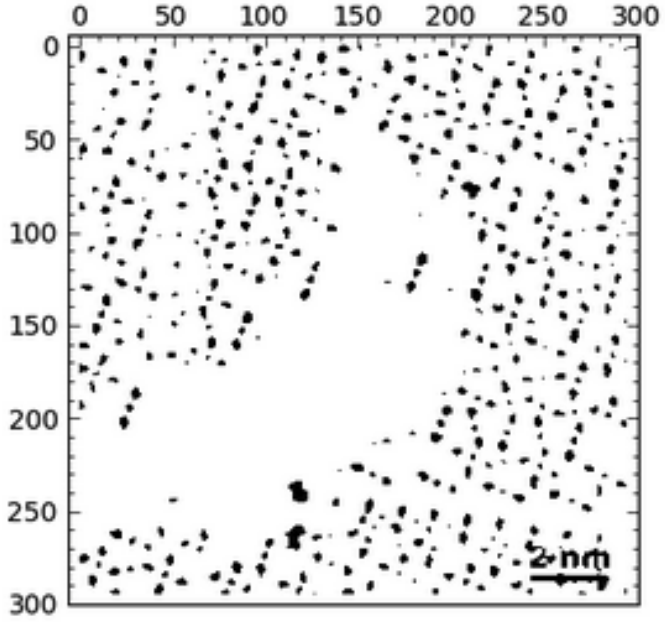


SciPy in action

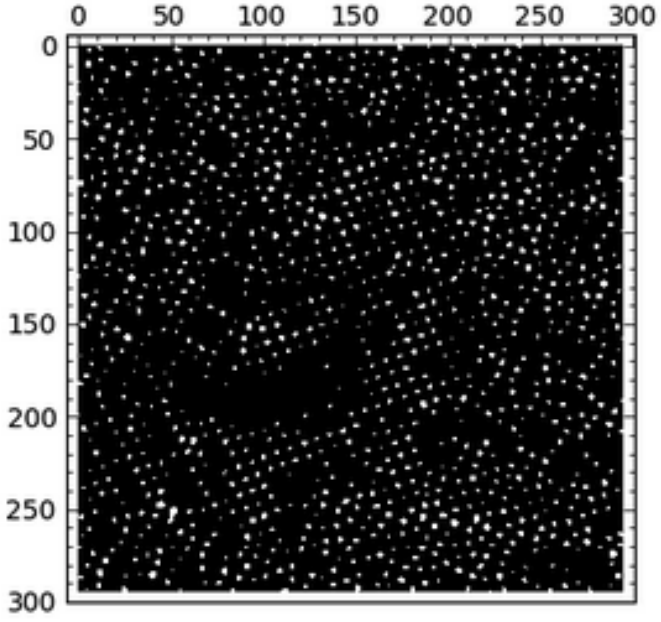
Computation of structural models

We take the following (naïve) approach:

- Segmentation of the atoms by thresholding and morphological operations.



img>0.2



img>0.7

SciPy in action

Computation of structural models

We take the following (naïve) approach:

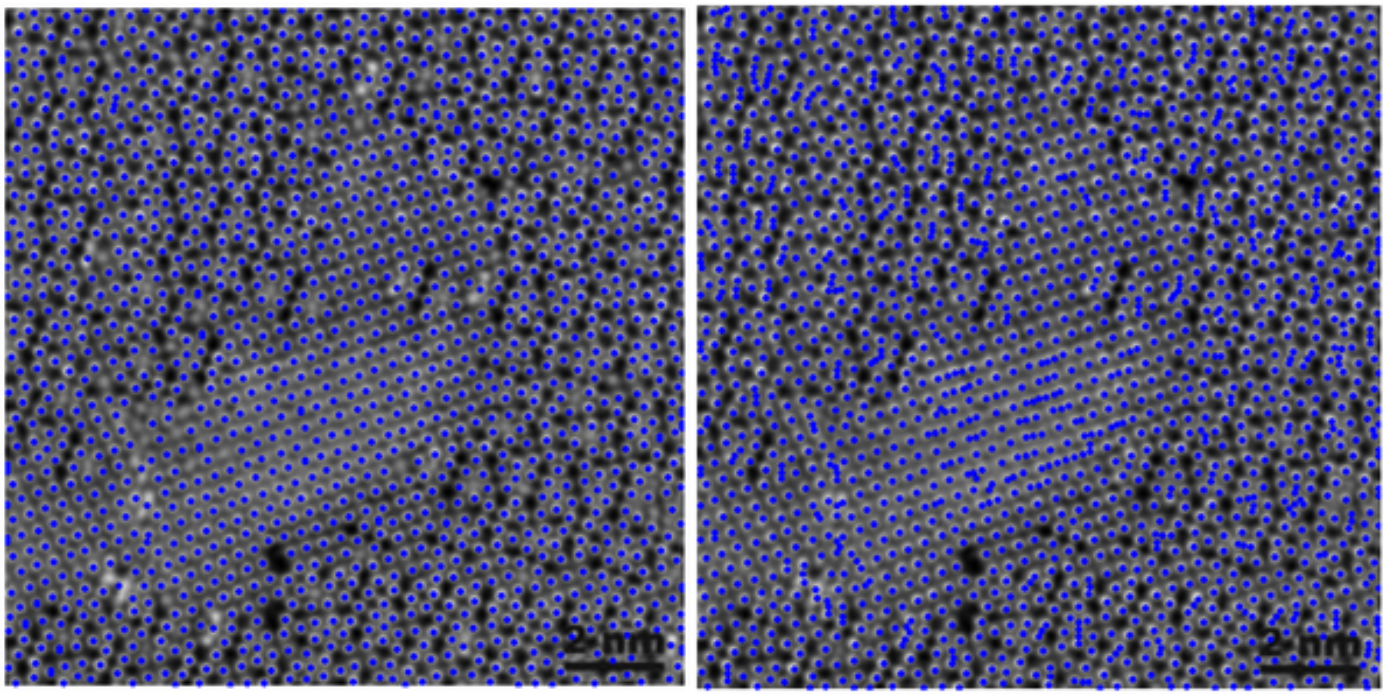
- Segmentation of the atoms by thresholding and morphological operations.
- Connected component labeling to extract each atom for posterior examination.

SciPy in action

Computation of structural models

We take the following (naïve) approach:

- Segmentation of the atoms by thresholding and morphological operations.
- Connected component labeling to extract each atom for posterior examination.
- Computation of the centers of mass of each label identified as an atom.

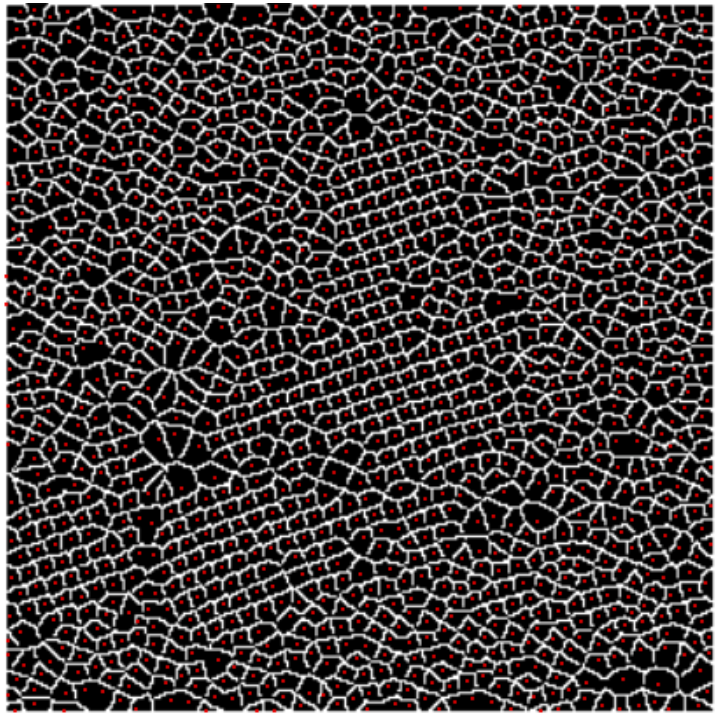


SciPy in action

Computation of structural models

We take the following (naïve) approach:

- Segmentation of the atoms by thresholding and morphological operations.
- Connected component labeling to extract each atom for posterior examination.
- Computation of the centers of mass of each label identified as an atom.
- Computation of the Voronoi diagram of the lattice formed by the previous points.



SciPy in action

Computation of structural models

```
# Preamble
import numpy
import scipy
from scipy.ndimage import binary_opening, label, center_of_mass,
     distance_transform_edt
# Load the image
img = scipy.misc.imread('NbW-STEM.png')
# Apply a threshold to segment atoms
BWatoms = (img > 0.62)
# Perform a binary operation to eliminate outliers
BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
# Segmentation of each atom
structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
segmentation,segments = label(BWatoms, structuring_element)
# Computation of centers of mass of each atom
coords = center_of_mass(img, segmentation, range(1,segments+1))
xcoords = array([x[0] for x in coords])
ycoords = array([x[1] for x in coords])
# Compute the Voronoi diagram of the lattice
L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
     return_indices=True)
Voronoi = segmentation[L1,L2]
```

SciPy in action

Computation of structural models

```
# Preamble
import numpy
import scipy
from scipy.ndimage import binary_opening, label, center_of_mass,
     distance_transform_edt
# Load the image
img = scipy.misc.imread('NbW-STEM.png')
# Apply a threshold to segment atoms
BWatoms = (img > 0.62)
# Perform a binary operation to eliminate outliers
BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
# Segmentation of each atom
structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
segmentation,segments = label(BWatoms, structuring_element)
# Computation of centers of mass of each atom
coords = center_of_mass(img, segmentation, range(1,segments+1))
xcoords = array([x[0] for x in coords])
ycoords = array([x[1] for x in coords])
# Compute the Voronoi diagram of the lattice
L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
     return_indices=True)
Voronoi = segmentation[L1,L2]
```

SciPy in action

Computation of structural models

```
# Preamble
import numpy
import scipy
from scipy.ndimage import binary_opening, label, center_of_mass,
     distance_transform_edt
# Load the image
img = scipy.misc.imread('NbW-STEM.png')
# Apply a threshold to segment atoms
BWatoms = (img > 0.62)
# Perform a binary operation to eliminate outliers
BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
# Segmentation of each atom
structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
segmentation,segments = label(BWatoms, structuring_element)
# Computation of centers of mass of each atom
coords = center_of_mass(img, segmentation, range(1,segments+1))
xcoords = array([x[0] for x in coords])
ycoords = array([x[1] for x in coords])
# Compute the Voronoi diagram of the lattice
L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
     return_indices=True)
Voronoi = segmentation[L1,L2]
```

SciPy in action

Computation of structural models

```
# Preamble
import numpy
import scipy
from scipy.ndimage import binary_opening, label, center_of_mass,
     distance_transform_edt
# Load the image
img = scipy.misc.imread('NbW-STEM.png')
# Apply a threshold to segment atoms
BWatoms = (img > 0.62)
# Perform a binary operation to eliminate outliers
BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
# Segmentation of each atom
structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
segmentation,segments = label(BWatoms, structuring_element)
# Computation of centers of mass of each atom
coords = center_of_mass(img, segmentation, range(1,segments+1))
xcoords = array([x[0] for x in coords])
ycoords = array([x[1] for x in coords])
# Compute the Voronoi diagram of the lattice
L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
     return_indices=True)
Voronoi = segmentation[L1,L2]
```

SciPy in action

Computation of structural models

```
# Preamble
import numpy
import scipy
from scipy.ndimage import binary_opening, label, center_of_mass,
     distance_transform_edt
# Load the image
img = scipy.misc.imread('NbW-STEM.png')
# Apply a threshold to segment atoms
BWatoms = (img > 0.62)
# Perform a binary operation to eliminate outliers
BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
# Segmentation of each atom
structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
segmentation,segments = label(BWatoms, structuring_element)
# Computation of centers of mass of each atom
coords = center_of_mass(img, segmentation, range(1,segments+1))
xcoords = array([x[0] for x in coords])
ycoords = array([x[1] for x in coords])
# Compute the Voronoi diagram of the lattice
L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
     return_indices=True)
Voronoi = segmentation[L1,L2]
```

SciPy in action

Computation of structural models

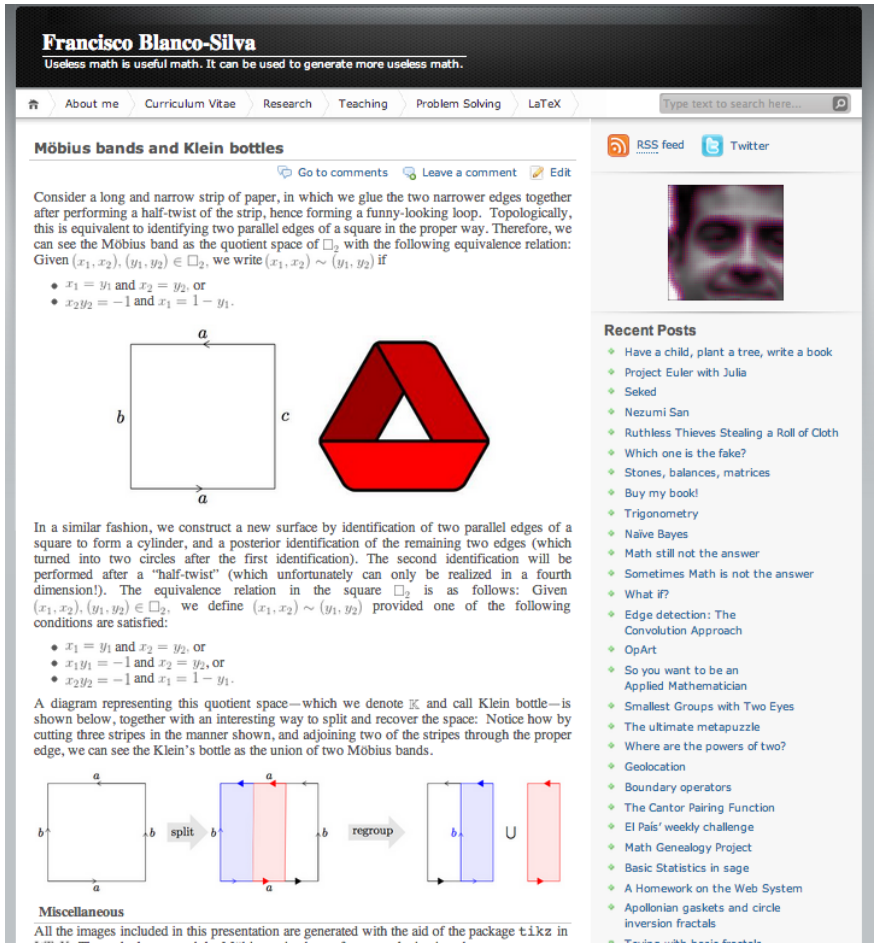
```
# Preamble
import numpy
import scipy
from scipy.ndimage import binary_opening, label, center_of_mass,
     distance_transform_edt
# Load the image
img = scipy.misc.imread('NbW-STEM.png')
# Apply a threshold to segment atoms
BWatoms = (img > 0.62)
# Perform a binary operation to eliminate outliers
BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
# Segmentation of each atom
structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
segmentation,segments = label(BWatoms, structuring_element)
# Computation of centers of mass of each atom
coords = center_of_mass(img, segmentation, range(1,segments+1))
xcoords = array([x[0] for x in coords])
ycoords = array([x[1] for x in coords])
# Compute the Voronoi diagram of the lattice
L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
     return_indices=True)
Voronoi = segmentation[L1,L2]
```

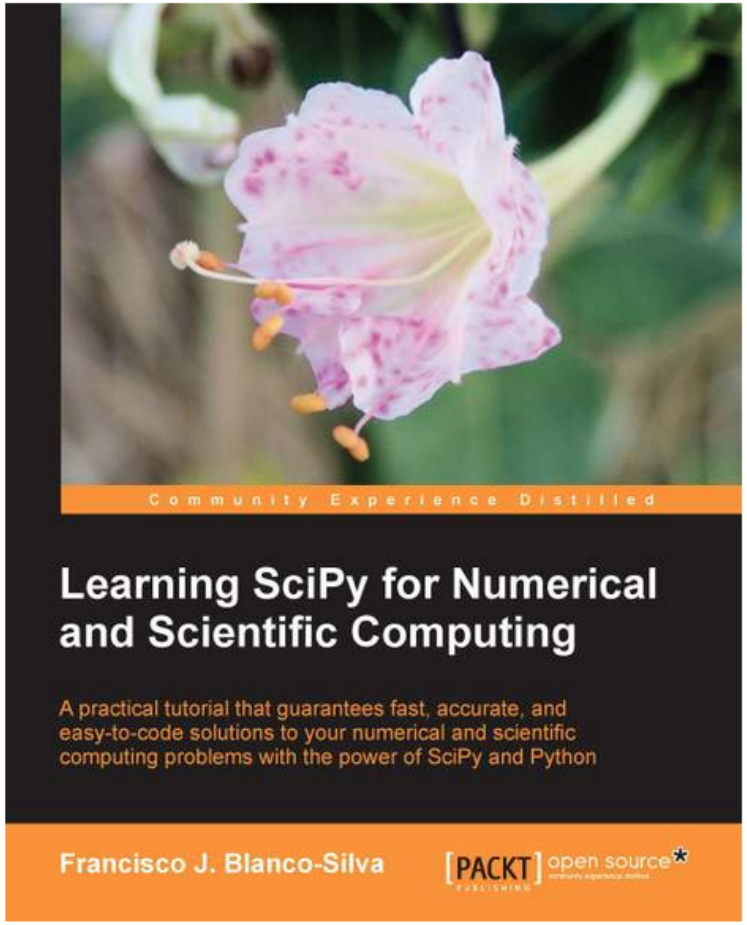
SciPy in action

Computation of structural models

```
# Preamble
import numpy
import scipy
from scipy.ndimage import binary_opening, label, center_of_mass,
     distance_transform_edt
# Load the image
img = scipy.misc.imread('NbW-STEM.png')
# Apply a threshold to segment atoms
BWatoms = (img > 0.62)
# Perform a binary operation to eliminate outliers
BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
# Segmentation of each atom
structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
segmentation,segments = label(BWatoms, structuring_element)
# Computation of centers of mass of each atom
coords = center_of_mass(img, segmentation, range(1,segments+1))
xcoords = array([x[0] for x in coords])
ycoords = array([x[1] for x in coords])
# Compute the Voronoi diagram of the lattice
L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
     return_indices=True)
Voronoi = segmentation[L1,L2]
```

For more information, examples, ideas, ...





`blancosilva.wordpress.com`