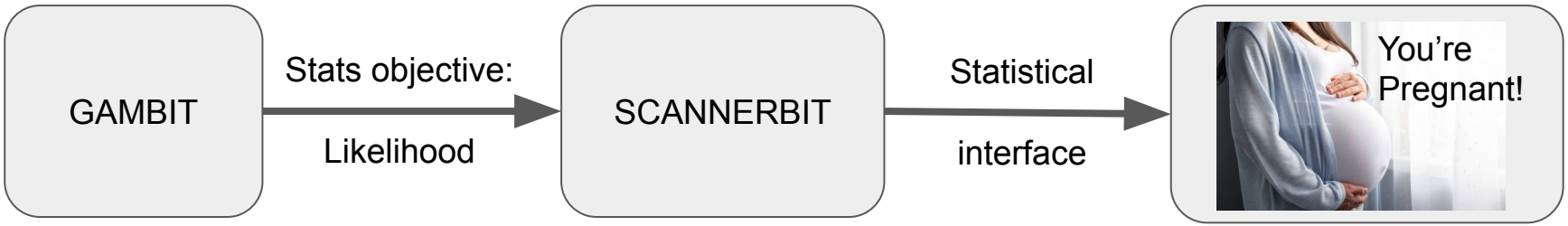
ScannerBit

Workgroup update

Gregory Martinez, *Andrew Fowlie*, Anders Kvellested, Will Handley

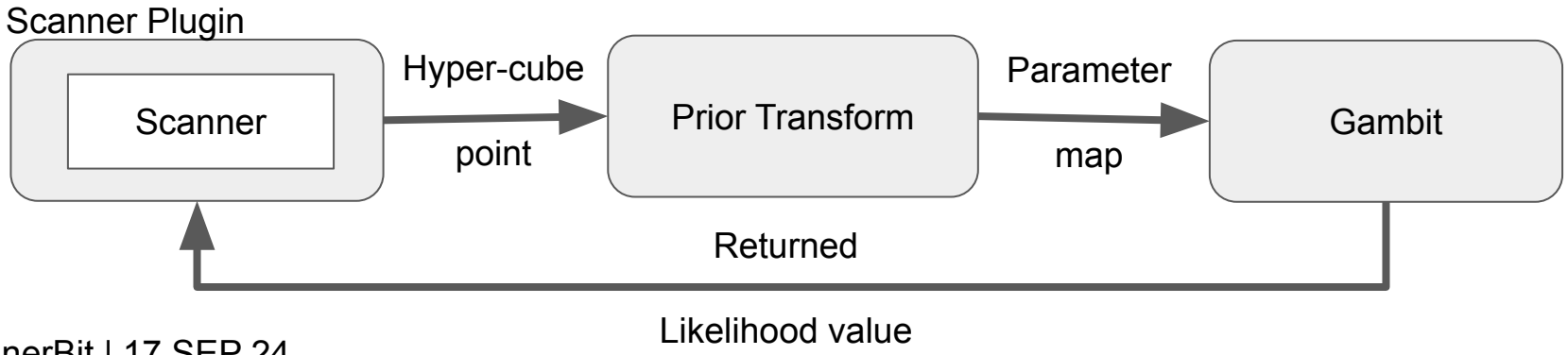
What it ScannerBit?

- Gambit's statistical inference bit
- Four active members currently – More are definitely welcome.
- Perfect for anyone who doesn't care about "science".
- ScannerBit is practically in its own environment where Gambit treated as a black box.



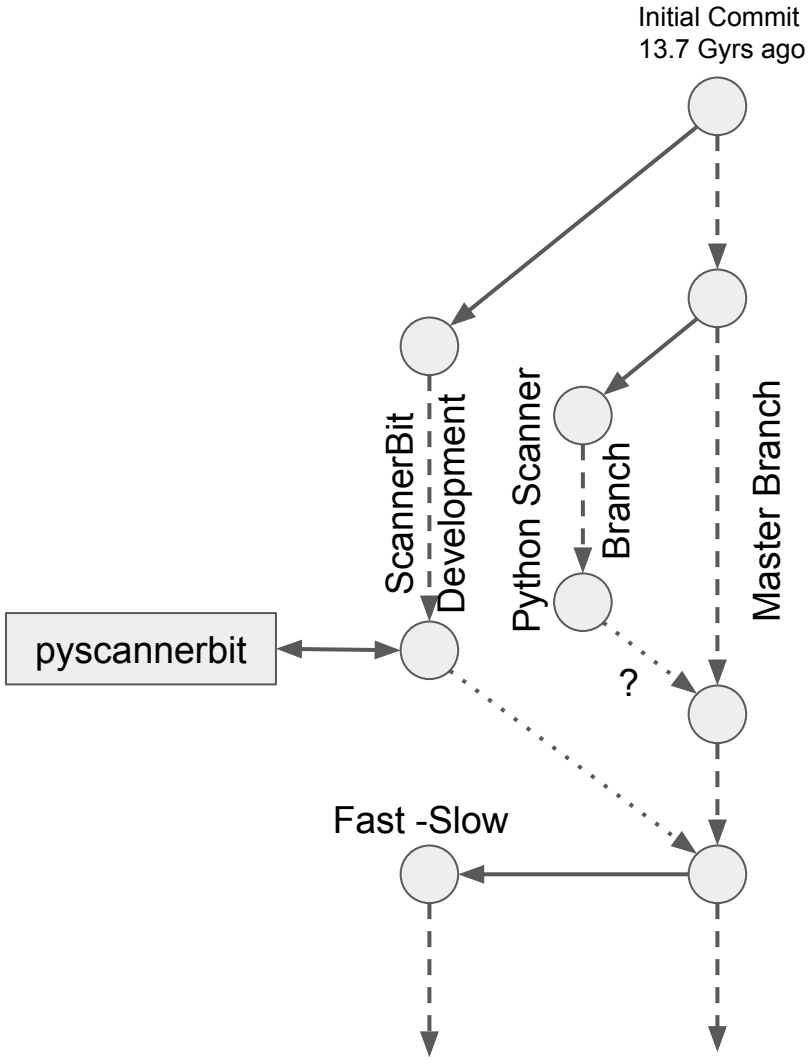
Quick overview

- ScannerBit “scans” in an unit hyper-cube.
- Each scanned hyper-cube point is converted to parameter points via the prior probability density
- Each scanner is contained within a self-contained plugin that is compiled separately from Gambit.



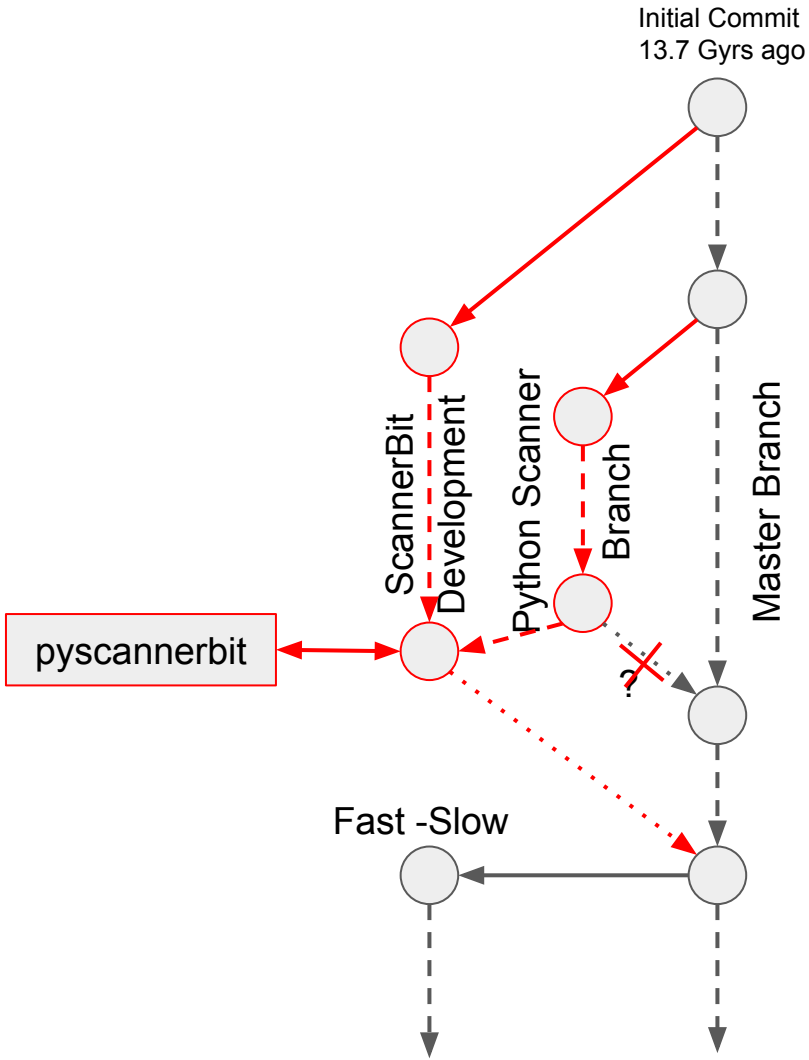
Plans from last meeting

- Merge Python Scanner Branch
  - Issues with base class.
  - Finalize interface.
  - Conversion to Eigen has begun.
- Merge ScannerBit_develoment
  - Has bunch of scanners.
  - Issues since it interfaces with pyscannerbit.
- Fast-Slow development
  - Convert to Eigen vectors.
  - Allow sub-hyper-cube inputs to likelihoods
  - Group parameters.



Plans from last meeting

- Merge Python Scanner Branch
  - Issues with base class.
  - Finalize interface.
  - Conversion to Eigen has begun.
- Merge ScannerBit_develoment
  - Has bunch of scanners.
  - Issues since it interfaces with pyscannerbit.
- Fast-Slow development
  - Convert to Eigen vectors.
  - Allow sub-hyper-cube inputs to likelihoods
  - Group parameters.



ScannerBit_development is no more

- ScannerBit_development has been merged into master and deleted.
- Python Scanner Tutoral, see last talk:
  - https://docs.google.com/presentation/d/1MbQU9ArGEk2TILbULb4v3t(CSmISPzMnzvmVZN1Y0hY/edit#slide=id.g22f02707f39_0_31
  - See ScannerBit/src/scanners/python/plugins/grid.py
  - … or anything in ScannerBit/src/scanners/python/plugins
- New scanners:
  - Jswarm, nessai, dynasty, ultranest, zeus, nautilus, emcee, pocomc, and a bunch of scipy scanners
- Python interface (for pyscannerbit)
  - In ScannerBit/python
  - Need -DWITH_PYTON_SCANNERBIT=True cmake option
  - Build with make scannerbit_python_interface
  - See ScannerBit/python/example.py

Python Plugin Example

```
import scipy.optimize
import scanner_plugin as splug
from utils import copydoc, version

class DifferentialEvolution(splug.scanner):
    """
    Differential evolution optimizer from scipy.
    """

    __version__ = version("scipy")

    def __init__(self, **kwargs):
        super().__init__(use_mpi=True, use_resume=False)

    @copydoc(scipy.optimize.differential_evolution)
    def run(self):
        bounds = [(0., 1.)] * self.dim
        res = scipy.optimize.differential_evolution(
            self.loglike_hypercube, bounds, **self.run_args)
        print(res)

__plugins__ = { “scipy_differential_evolution”: DifferentialEvolution}
```

Python Plugin Example

Pugin Interface:
- __init__(**args)
- run()
- __version__
- __plugins__

```
import scipy.optimize
import scanner_plugin as splug
from utils import copydoc, version

class DifferentialEvolution(splug.scanner):
    """
    Differential evolution optimizer from scipy.
    """

    __version__ = version("scipy")

    def __init__(self, **kwargs):
        super().__init__(use_mpi=True, use_resume=False)

    @copydoc(scipy.optimize.differential_evolution)
    def run(self):
        bounds = [(0., 1.)] * self.dim
        res = scipy.optimize.differential_evolution(
            self.loglike_hypercube, bounds, **self.run_args)
        print(res)

__plugins__ = { “scipy_differential_evolution”: DifferentialEvolution}
```

Python Plugin Example

Pugin Interface:
- __init__(**args)
- run()
- __version__
- __plugins__

```
import scipy.optimize
import scanner_plugin as splug
from utils import copydoc, version

class DifferentialEvolution(splug.scanner):
    """
    Differential evolution optimizer from scipy.
    """

    __version__ = version("scipy")

    def __init__(self, **kwargs):
        super().__init__(use_mpi=True, use_resume=False)

    @copydoc(scipy.optimize.differential_evolution)
    def run(self):
        bounds = [(0., 1.)] * self.dim
        res = scipy.optimize.differential_evolution(
            self.loglike_hypercube, bounds, **self.run_args)
        print(res)

__plugins__ = { “scipy_differential_evolution”: DifferentialEvo
```

scanner base class:
- loglike_hypercube
- loglike_physical
- run_args
- dim
- Printer
- Init arguments:
  - use_mpi,
  - use_resume

Python Plugin Example

Pugin Interface:
- __init__(**args)
- run()
- __version__
- __plugins__

```
import scipy.optimize
import scanner_plugin as splug
from utils import copydoc, version

class DifferentialEvolution(splug.scanner):
    """
    Differential evolution optimizer from scipy.
    """

    __version__ = version("scipy")

    def __init__(self, **kwargs):
        super().__init__(use_mpi=True, use_resume=False)

    @copydoc(scipy.optimize.differential_evolution)
    def run(self):
        bounds = [(0., 1.)] * self.dim
        res = scipy.optimize.differential_evolution(
            self.loglike_hypercube, bounds, **self.run_args)
        print(res)

__plugins__ = { “scipy_differential_evolution”: DifferentialEvo
```

Documentation
- Diagnostics will use docs strings
- Will separate __init__ and run doc strings

scanner base class:
- loglike_hypercube
- loglike_physical
- run_args
- dim
- Printer
- Init arguments:
  - use_mpi,
  - use_resume

Python Plugin Example

Pugin Interface:
- __init__(**args)
- run()
- __version__
- __plugins__

```
import scipy.optimize
import scanner_plugin as splug
from utils import copydoc, version

class DifferentialEvolution(splug.scanner):
    """
    Differential evolution optimizer from scipy.
    """

    __version__ = version("scipy")

    def __init__(self, **kwargs):
        super().__init__(use_mpi=True, use_resume=False)

    @copydoc(scipy.optimize.differential_evolution)
    def run(self):
        bounds = [(0., 1.)] * self.dim
        res = scipy.optimize.differential_evolution(
            self.loglike_hypercube, bounds, **self.run_args)
        print(res)

__plugins__ = { “scipy_differential_evolution”: DifferentialEvo
```

Utils:
- version
- @copydoc
- mpi

Documentation
- Diagnostics will use docs strings
- Will separate __init__ and run doc strings

scanner base class:
- loglike_hypercube
- loglike_physical
- run_args
- dim
- Printer
- Init arguments:
  - use_mpi,
  - use_resume

Python Plugin Example

Pugin Interface:
- __init__(**args)
- run()
- __version__
- __plugins__

```
import scipy.optimize
import scanner_plugin as splug
from utils import copydoc, version

class DifferentialEvolution(splug.scanner):
    """
    Differential evolution optimizer from scipy.
    """

    __version__ = version("scipy")

    def __init__(self, **kwargs):
        super().__init__(use_mpi=True, use_resume=False)

    @copydoc(scipy.optimize.differential_evolution)
    def run(self):
        bounds = [(0., 1.)] * self.dim
        res = scipy.optimize.differential_evolution(
            self.loglike_hypercube, bounds, **self.run_args)
        print(res)

__plugins__ = { “scipy_differential_evolution”: DifferentialEvo
```

Utils:
- version
- @copydoc
- mpi

Documentation
- Diagnostics will use docs strings
- Will separate __init__ and run doc strings

scanner base class:
- loglike_hypercube
- loglike_physical
- run_args
- dim
- Printer
- Init arguments:
  - use_mpi,
  - use_resume

Inifile interface is same

```
Scanner:
  use_scanner: scipy_differential_evolution

  scanners:
    scipy_differential_evolution:
      like: LogLike
      plugin: scipy_differential_evolution
      run:
        strategy: 'best1bin'
        maxiter: 1000
        popsize: 15
        tol: 0.01
        mutation: [0.5, 1]
        recombination: 0.7
        disp: false
        polish: true
        init: 'latinhypercube'
        atol: 0
        updating: 'immediate'
```

Python Interface

```
import sys
import ctypes
import yaml

flags = sys.getdlopenflags()
sys.setdlopenflags(flags | ctypes.RTLD_GLOBAL)

import ScannerBit as scan

def like(m):
    a = m["model1::x"]
    scan.print("my_param", 0.5)

    return -a*a/2.0

class prior:
    # see ScannerBit/python/example.py

with open("ScannerBit.yaml", 'r') as f:
    inifile = yaml.load(f)

myscan = scan.scan(True)
myscan.run(inifile=inifile, lnlike={"LogLike": like}, restart=True,
                    #prior =prior
                    )
```

Python Interface

Needs this to work:
- Set flags before importing ScannerBit

```
import sys
import ctypes
import yaml

flags = sys.getdlopenflags()
sys.setdlopenflags(flags | ctypes.RTLD_GLOBAL)

import ScannerBit as scan

def like(m):
    a = m["model1::x"]
    scan.print("my_param", 0.5)

    return -a*a/2.0

class prior:
    # see ScannerBit/python/example.py

with open("ScannerBit.yaml", 'r') as f:
    inifile = yaml.load(f)

myscan = scan.scan(True)
myscan.run(inifile=inifile, lnlike={"LogLike": like}, restart=True,
                    #prior =prior
                    )
```

Python Interface

Needs this to work:
- Set flags before importing ScannerBit

```
import sys
import ctypes
import yaml

flags = sys.getdlopenflags()
sys.setdlopenflags(flags | ctypes.RTLD_GLOBAL)

import ScannerBit as scan

def like(m):
    a = m["model1::x"]
    scan.print("my_param", 0.5)

    return -a*a/2.0

class prior:
    # see ScannerBit/python/example.py

with open("ScannerBit.yaml", 'r') as f:
    inifile = yaml.load(f)

myscan = scan.scan(True)
myscan.run(inifile=inifile, lnlike={"LogLike": like}, restart=True,
                    #prior =prior
                    )
```

Define Likelihood:
- Likelihood only needs to be callable
- Can optionally define prior in python

Python Interface

Needs this to work:
- Set flags before importing ScannerBit

```
import sys
import ctypes
import yaml

flags = sys.getdlopenflags()
sys.setdlopenflags(flags | ctypes.RTLD_GLOBAL)

import ScannerBit as scan

def like(m):
    a = m["model1::x"]
    scan.print("my_param", 0.5)

    return -a*a/2.0

class prior:
    # see ScannerBit/python/example.py

with open("ScannerBit.yaml", 'r') as f:
    inifile = yaml.load(f)

myscan = scan.scan(True)
myscan.run(inifile=inifile, lnlike={"LogLike": like}, restart=True,
                    #prior =prior
                    )
```

Settings:
- From inifile
- Or define dict in python

Define Likelihood:
- Likelihood only needs to be callable
- Can optionally define prior in python

Python Interface

Needs this to work:
- Set flags before importing ScannerBit

```
import sys
import ctypes
import yaml

flags = sys.getdlopenflags()
sys.setdlopenflags(flags | ctypes.RTLD_GLOBAL)

import ScannerBit as scan

def like(m):
    a = m["model1::x"]
    scan.print("my_param", 0.5)

    return -a*a/2.0

class prior:
    # see ScannerBit/python/example.py

with open("ScannerBit.yaml", 'r') as f:
    inifile = yaml.load(f)

myscan = scan.scan(True)
myscan.run(inifile=inifile, lnlike={"LogLike": like}, restart=True,
                #prior =prior
                )
```

Run
- Declare scan object and run
- Constructor option:
  - Initial MPI?
- Can get diagnostics via: myscan.diagnostics()

Settings:
- From inifile
- Or define dict in python

Define Likelihood:
- Likelihood only needs to be callable
- Can optionally define prior in python

Python Interface

Needs this to work:
- Set flags before importing ScannerBit

```
import sys
import ctypes
import yaml

flags = sys.getdlopenflags()
sys.setdlopenflags(flags | ctypes.RTLD_GLOBAL)

import ScannerBit as scan

def like(m):
    a = m["model1::x"]
    scan.print("my_param", 0.5)

    return -a*a/2.0

class prior:
    # see ScannerBit/python/example.py

with open("ScannerBit.yaml", 'r') as f:
    inifile = yaml.load(f)

myscan = scan.scan(True)
myscan.run(inifile=inifile, lnlike={"LogLike": like}, restart=True,
                #prior =prior
                )
```

Run
- Declare scan object and run
- Constructor option:
  - Initial MPI?
- Can get diagnostics via: myscan.diagnostics()

Settings:
- From inifile
- Or define dict in python

Define Likelihood:
- Likelihood only needs to be callable
- Can optionally define prior in python

PyScannerBit at end of life

- Github: https://github.com/bjfar/pyscannerbit
- Also … pip install pyscannerbit works too
- Pyscannerbit is no longer supported
- Last push on most recent ScannerBit version
- Haven’t really worked on it anyways
- Has commonality with Gambit-lite

Future Plans

- ScannerBit paper 2.0
  - Software paper with all the new scanners / python interface

Future Plans

- ScannerBit paper 2.0
  - Software paper with all the new scanners / python interface
- Emulation plugins?
  - Did some work … but …
  - MPI integration will be tricky

Future Plans

- ScannerBit paper 2.0
  - Software paper with all the new scanners / python interface
- Emulation plugins?
  - Did some work … but …
  - MPI integration will be tricky
- MLS
  - Sheng wanted to to experiment with this. But there was a time limit that probably passed
  - Note from meeting
    - To get observables, use only the capabilities tag to “pass_to_scanner”.
    - Put into yaml node for scanner
    - Store yaml node in likelihood container
    - **Unsure if this would work**
    - Tag yaml node to either module function name or capability name
  - https://drive.google.com/drive/folders/14SaDwQuvkSXNENQTyroA7dxo84t-6ctz

Future Plans

- ScannerBit paper 2.0
  - Software paper with all the new scanners / python interface
- Emulation plugins?
  - Did some work … but …
  - MPI integration will be tricky
- MLS
  - Sheng wanted to to experiment with this. But there was a time limit that probably passed
  - Note from meeting
    - To get observables, use only the capabilities tag to “pass_to_scanner”.
    - Put into yaml node for scanner
    - Store yaml node in likelihood container
    - **Unsure if this would work**
    - Tag yaml node to either module function name or capability name
  - https://drive.google.com/drive/folders/14SaDwQuvkSXNENQTyroA7dxo84t-6ctz
- Fast-Slow development
  - Convert to Eigen vectors.
  - Allow sub-hyper-cube inputs to likelihoods
  - Group parameters.