Disco/Hadoop MapReduce

Benjamin Zaitlen
Continuum Analytics

PyCon 2013

CONTINUUM
ANALYTICS

What is Large Data?

CONTINUUM
ANALYTICS

What is Large Data?

- Can't fit into Excel

CONTINUUM ANALYTICS

What is Large Data?

- Can't fit into Excel
  - Increase Memory

CONTINUUM ANALYTICS

What is Large Data?

- Can't fit into Excel
    - Increase Memory
- Can't fit into R

CONTINUUM ANALYTICS

What is Large Data?

- Can’t fit into Excel
  - Increase Memory
- Can’t fit into R
  - Increase Memory

What is Large Data?

- Can’t fit into Excel
  - Increase Memory
- Can’t fit into R
  - Increase Memory
- Can’t fit into Memory

What is Large Data?

- Can't fit into Excel
  - Increase Memory
- Can't fit into R
  - Increase Memory
- Can't fit into Memory
  - Increase Memory

What is Large Data?

- Can’t fit into Excel
  - Increase Memory
- Can’t fit into R
  - Increase Memory
- Can’t fit into Memory
  - Increase Memory
- Can’t fit on a single disk

CONTINUUM ANALYTICS

What is Large Data?

- Can't fit into Excel
  - Increase Memory
- Can't fit into R
  - Increase Memory
- Can't fit into Memory
  - Increase Memory
- Can't fit on a single disk
  - Distributed Filesystem: SAN, HDFS/DDFS, AWS: S3, Redshift, etc.

MapReduce

Framework to help solve the problem of distributed computation for distributed data

MapReduce

Framework to help solve the problem of distributed computation for distributed data

- A mass of data: records

MapReduce

Framework to help solve the problem of distributed computation for distributed data

- A mass of data: records
- Split/**Map** records into key-values pairs

MapReduce

Framework to help solve the problem of distributed computation for distributed data

- A mass of data: records
- Split/**Map** records into key-values pairs
- Collect/Partition kv pairs (Optional Sort)

CONTINUUM ANALYTICS

MapReduce

Framework to help solve the problem of distributed computation for distributed data

- A mass of data: records
- Split/**Map** records into key-values pairs
- Collect/Partition kv pairs (Optional Sort)
- Buckets are passed to **Reduce** function

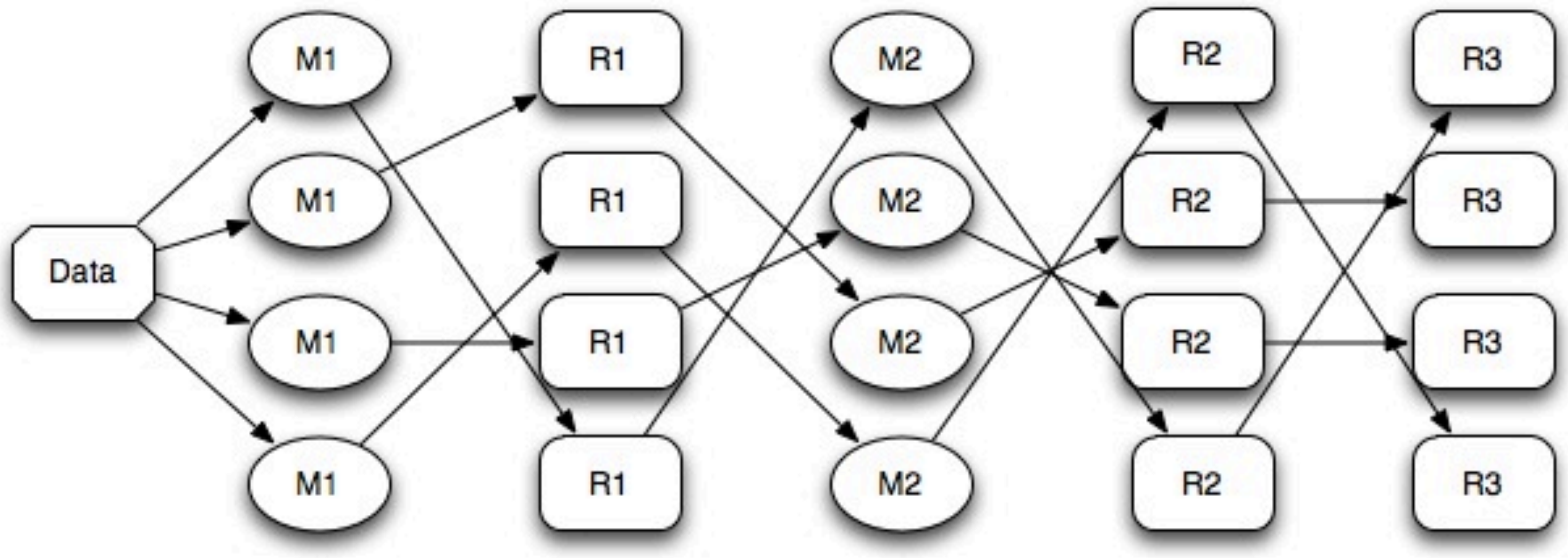
MapReduce

Framework to help solve the problem of distributed computation for distributed data

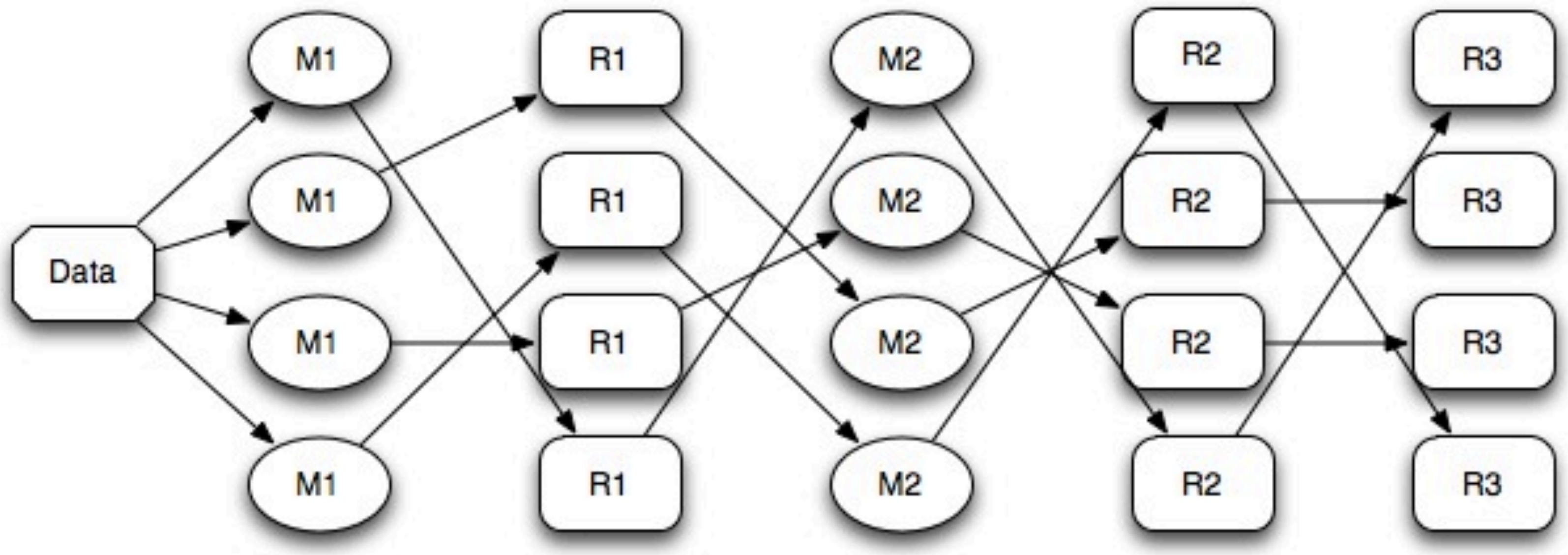
- A mass of data: records
- Split/**Map** records into key-values pairs
- Collect/Partition kv pairs (Optional Sort)
- Buckets are passed to **Reduce** function
- Result is returned

CONTINUUM ANALYTICS

MapReduce Workflow

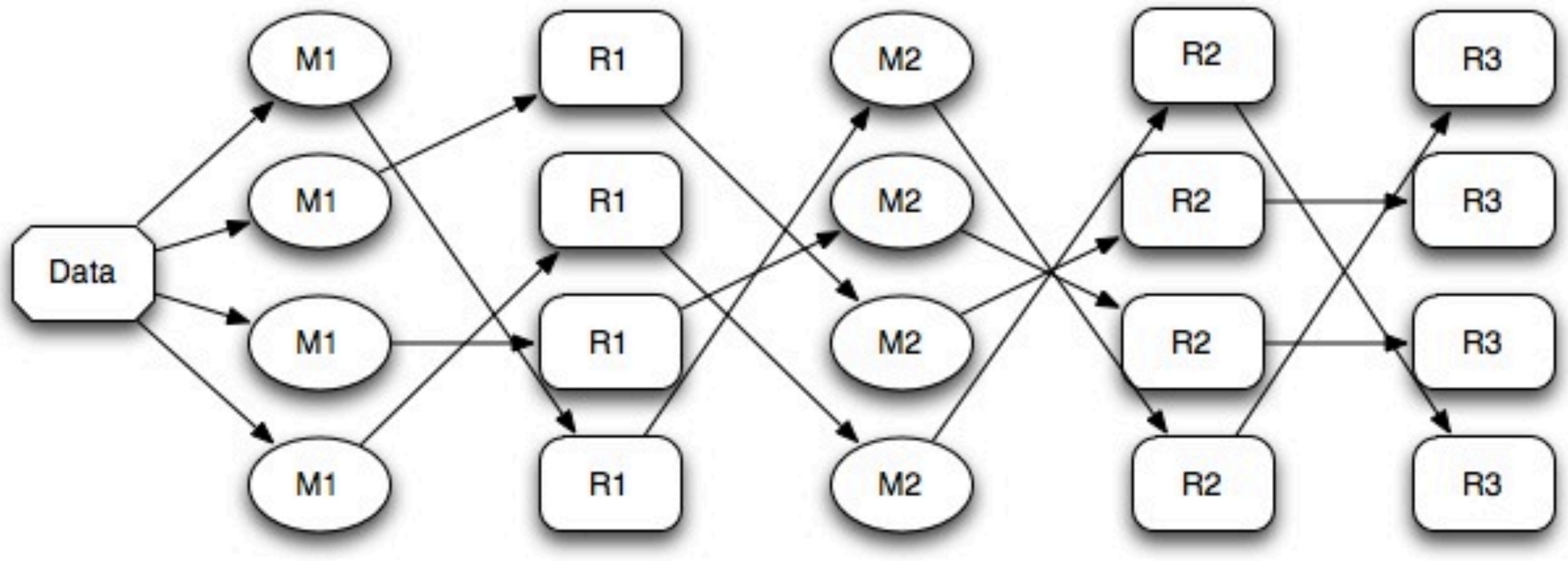


MapReduce Workflow



- Push code to data

MapReduce Workflow



- Push code to data
- Lots of network traffic

MapReduce it's a Party

CONTINUUM ANALYTICS

MapReduce it's a Party

CONTINUUM ANALYTICS

MapReduce it's a Party

CONTINUUM
ANALYTICS

MapReduce it’s a Party

- Disco: Python + Erlang

MapReduce it's a Party

- Disco: Python + Erlang
- Hadoop: Java-Dumbo (Python Streaming)

MapReduce it’s a Party

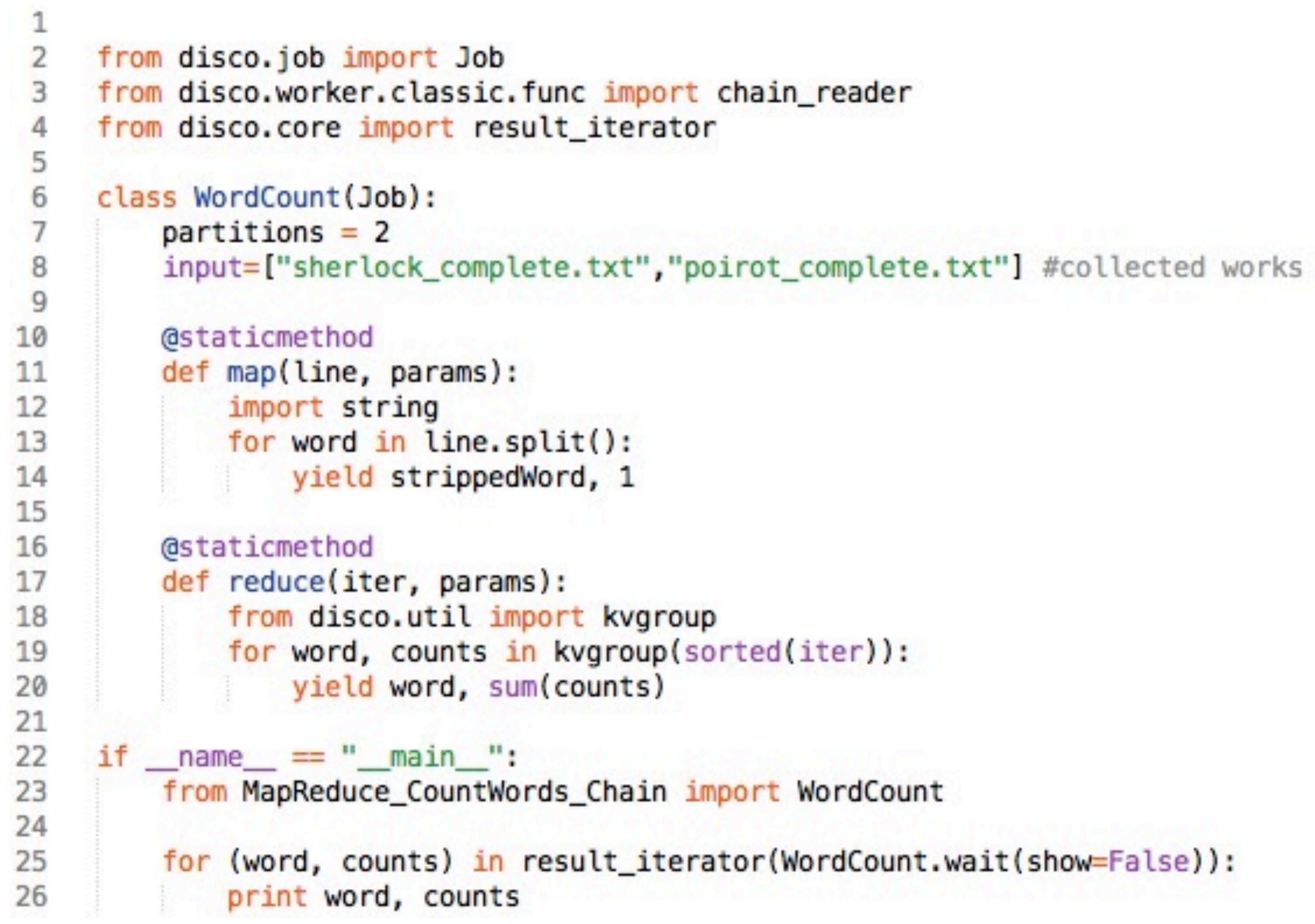
- Disco: Python + Erlang
- Hadoop: Java-Dumbo (Python Streaming)
- Distributed FileSystem: DDFS

MapReduce it’s a Party

- Disco: Python + Erlang
- Hadoop: Java-Dumbo (Python Streaming)
- Distributed FileSystem: DDFS
- Bring your friends
  - NumPy
  - SciPy
  - pandas
  - scikits-learn
  - OpenCV

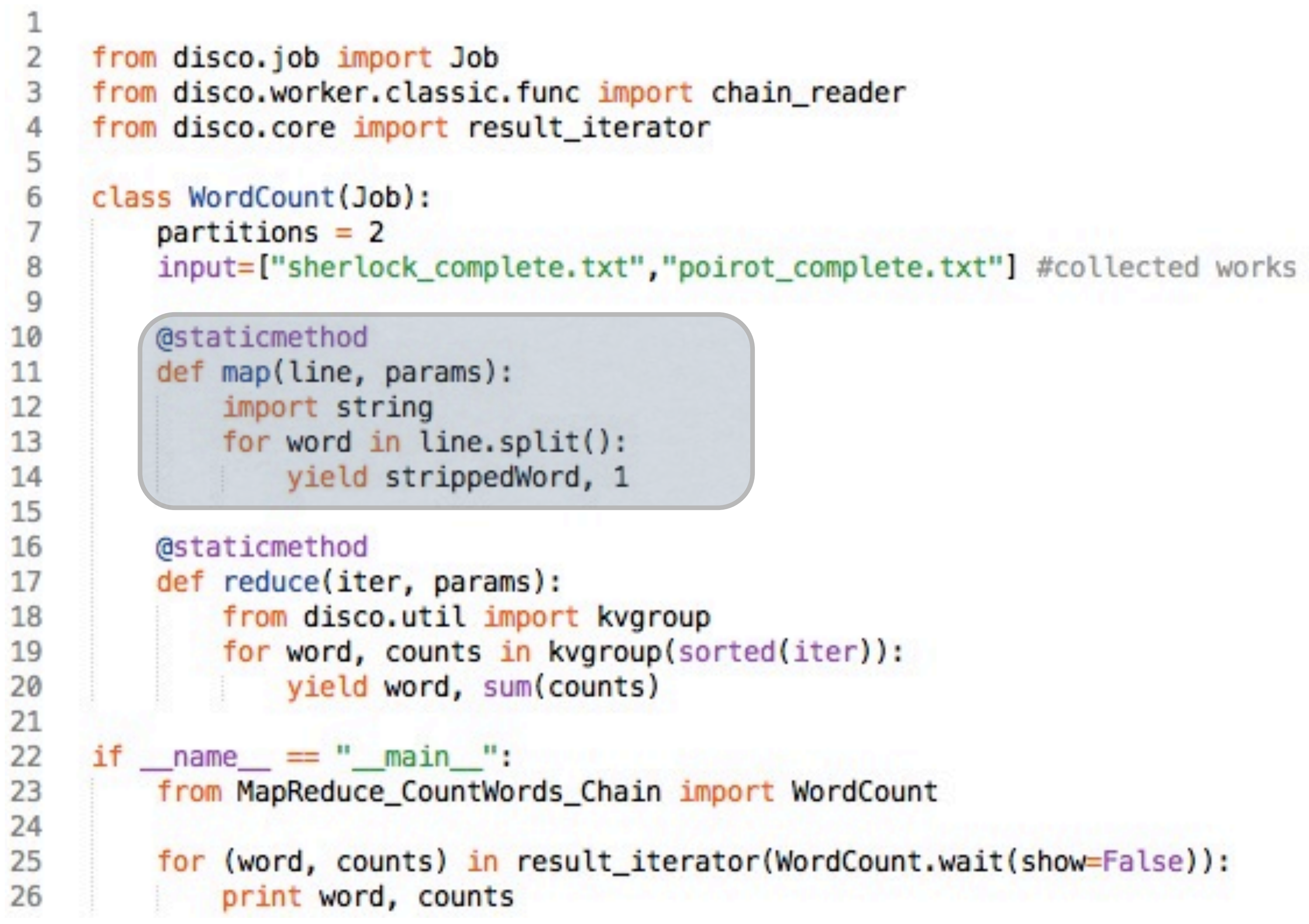
CONTINUUM ANALYTICS

Canonical Example



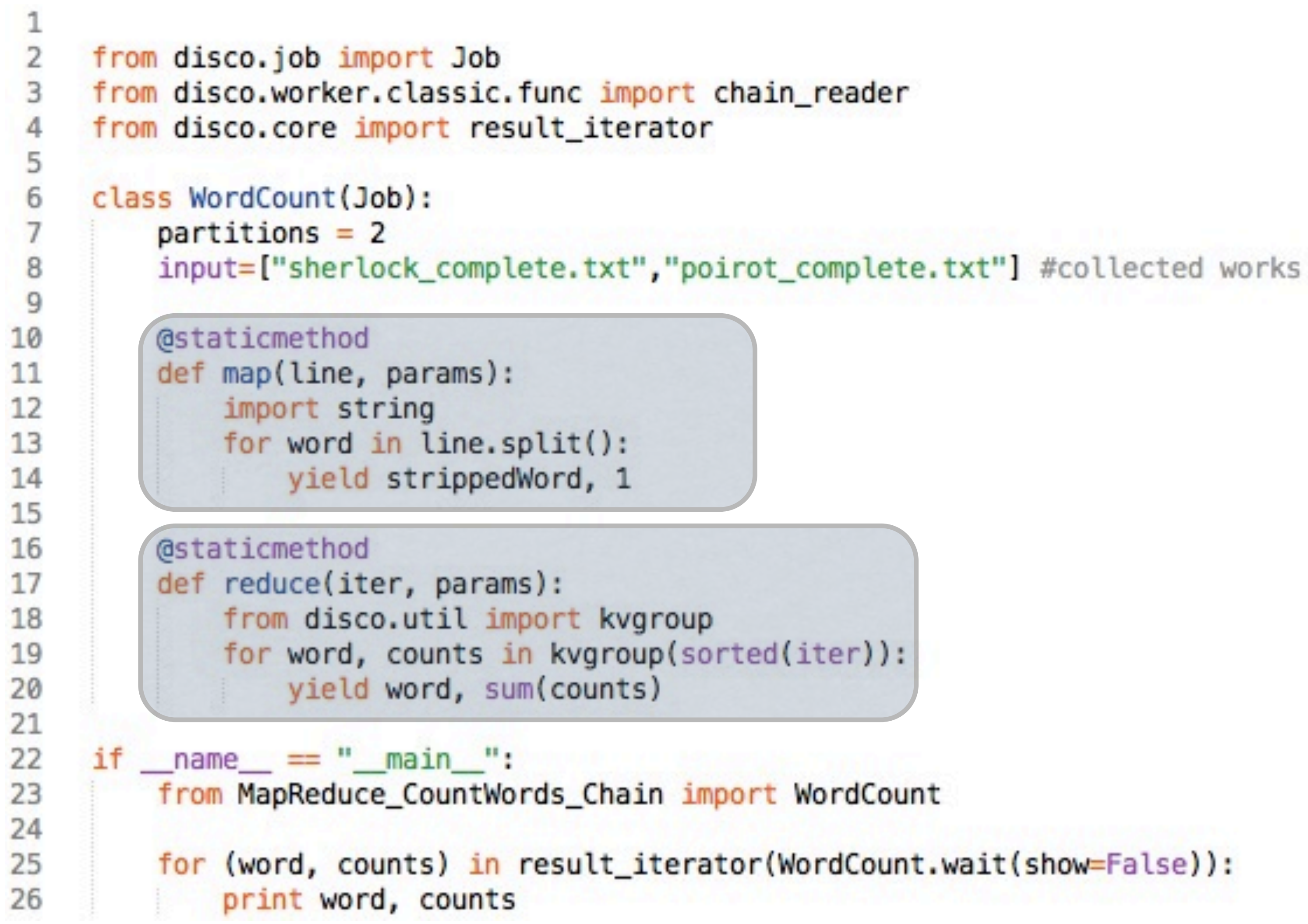
```python
from disco.job import Job
from disco.worker.classic.func import chain_reader
from disco.core import result_iterator

class WordCount(Job):
    partitions = 2
    input=["sherlock_complete.txt","poirot_complete.txt"] #collected works

    @staticmethod
    def map(line, params):
        import string
        for word in line.split():
            yield strippedWord, 1

    @staticmethod
    def reduce(iter, params):
        from disco.util import kvgroup
        for word, counts in kvgroup(sorted(iter)):
            yield word, sum(counts)

if __name__ == "__main__":
    from MapReduce_CountWords_Chain import WordCount

    for (word, counts) in result_iterator(WordCount.wait(show=False)):
        print word, counts
```

Canonical Example



```python

from disco.job import Job
from disco.worker.classic.func import chain_reader
from disco.core import result_iterator

class WordCount(Job):
    partitions = 2
    input=["sherlock_complete.txt","poirot_complete.txt"] #collected works

    @staticmethod
    def map(line, params):
        import string
        for word in line.split():
            yield strippedWord, 1

    @staticmethod
    def reduce(iter, params):
        from disco.util import kvgroup
        for word, counts in kvgroup(sorted(iter)):
            yield word, sum(counts)

if __name__ == "__main__":
    from MapReduce_CountWords_Chain import WordCount

    for (word, counts) in result_iterator(WordCount.wait(show=False)):
        print word, counts
```

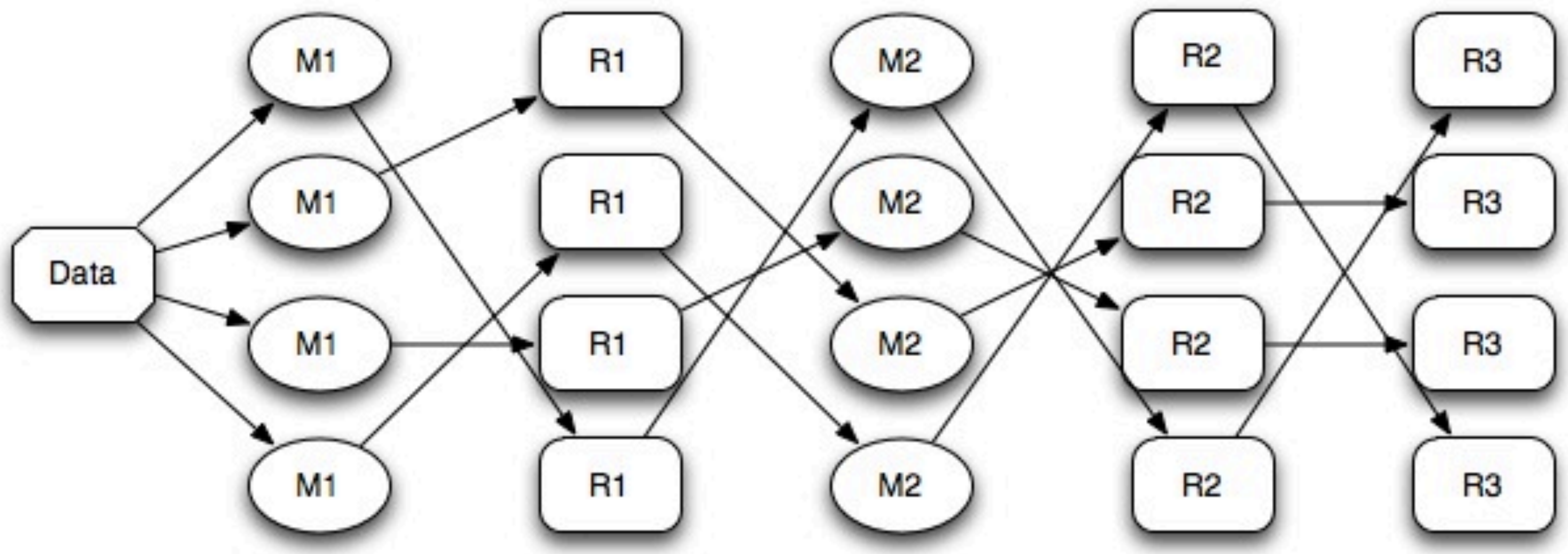
Canonical Example



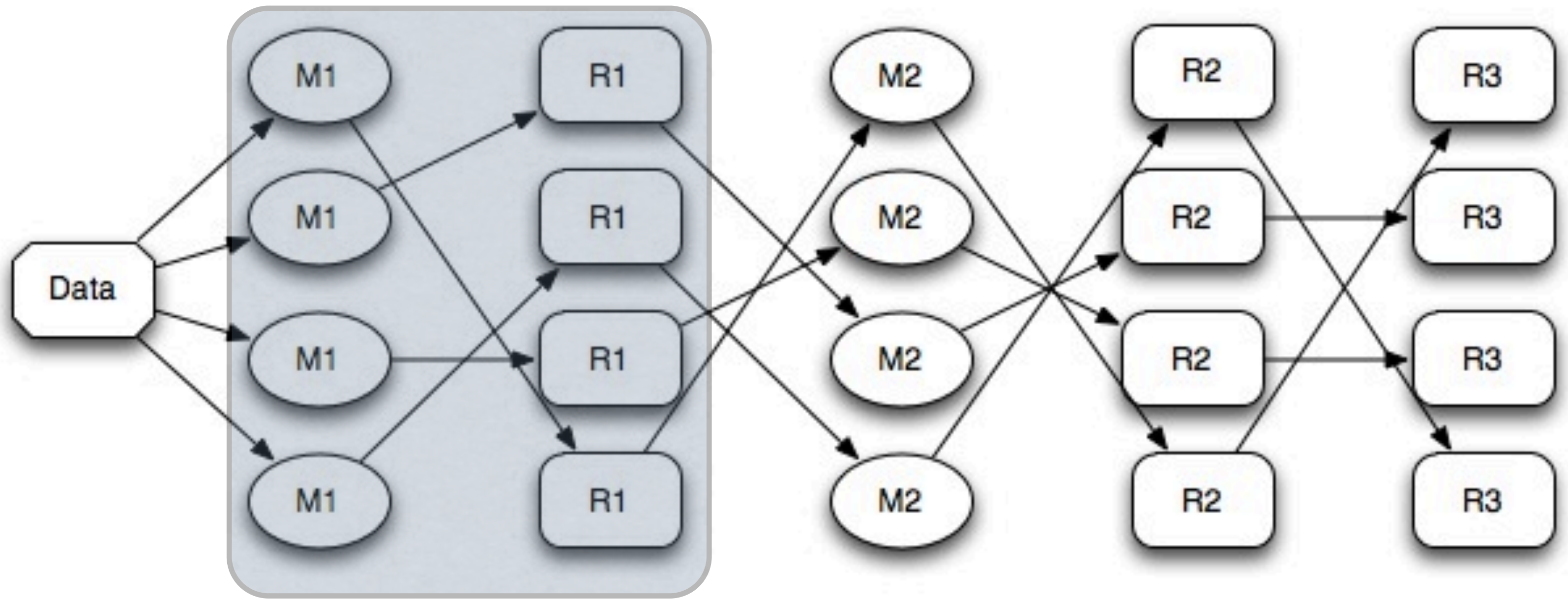
```python

from disco.job import Job
from disco.worker.classic.func import chain_reader
from disco.core import result_iterator

class WordCount(Job):
    partitions = 2
    input=["sherlock_complete.txt","poirot_complete.txt"] #collected works

    @staticmethod
    def map(line, params):
        import string
        for word in line.split():
            yield strippedWord, 1

    @staticmethod
    def reduce(iter, params):
        from disco.util import kvgroup
        for word, counts in kvgroup(sorted(iter)):
            yield word, sum(counts)

if __name__ == "__main__":
    from MapReduce_CountWords_Chain import WordCount

    for (word, counts) in result_iterator(WordCount.wait(show=False)):
        print word, counts
```

CONTINUUM
ANALYTICS

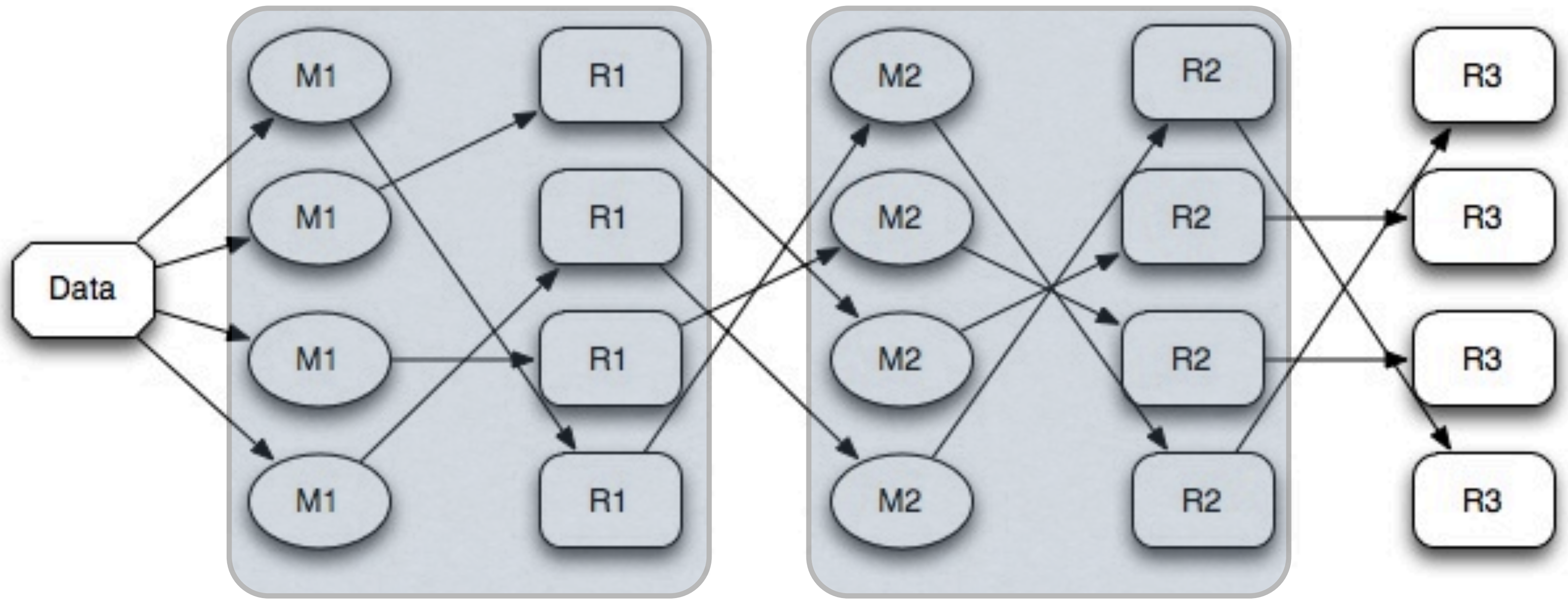
Chain Jobs



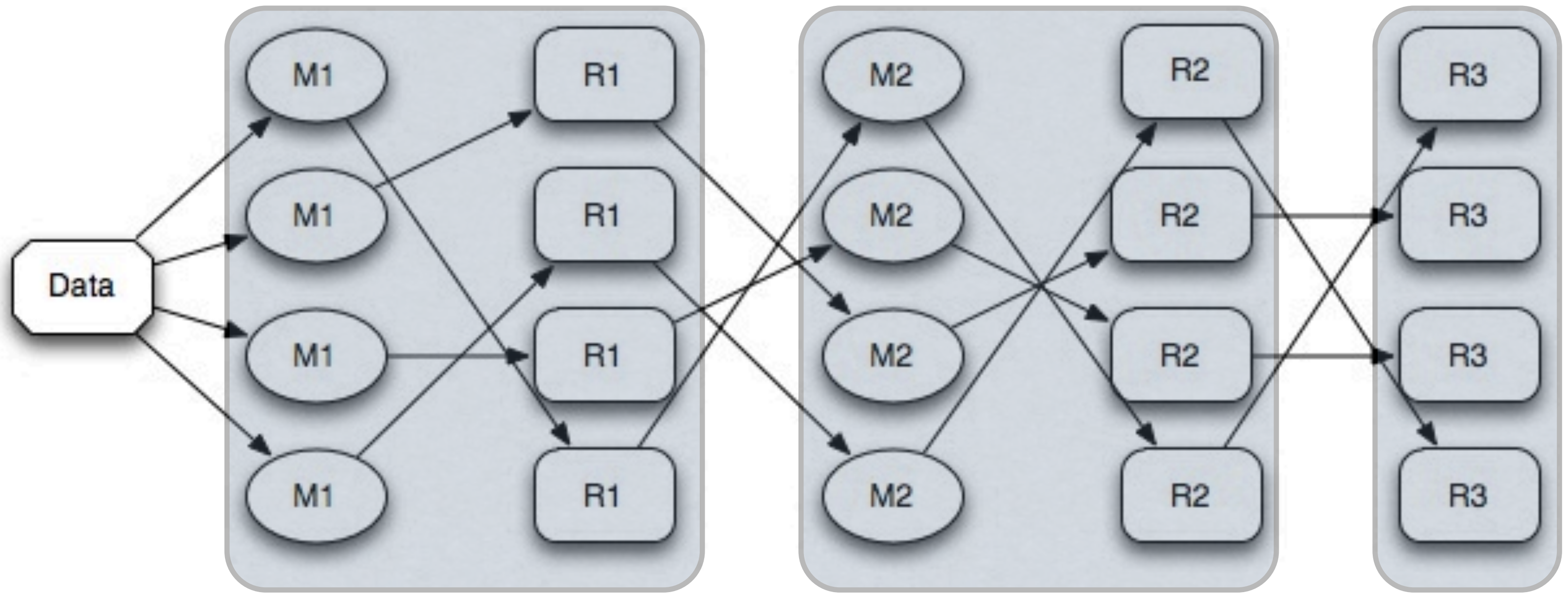
Chain Jobs



Chain Jobs



Chain Jobs



MapReduce Thoughts

CONTINUUM
ANALYTICS

MapReduce Thoughts

- Data Cleansing
    - Everyone's pain point

MapReduce Thoughts

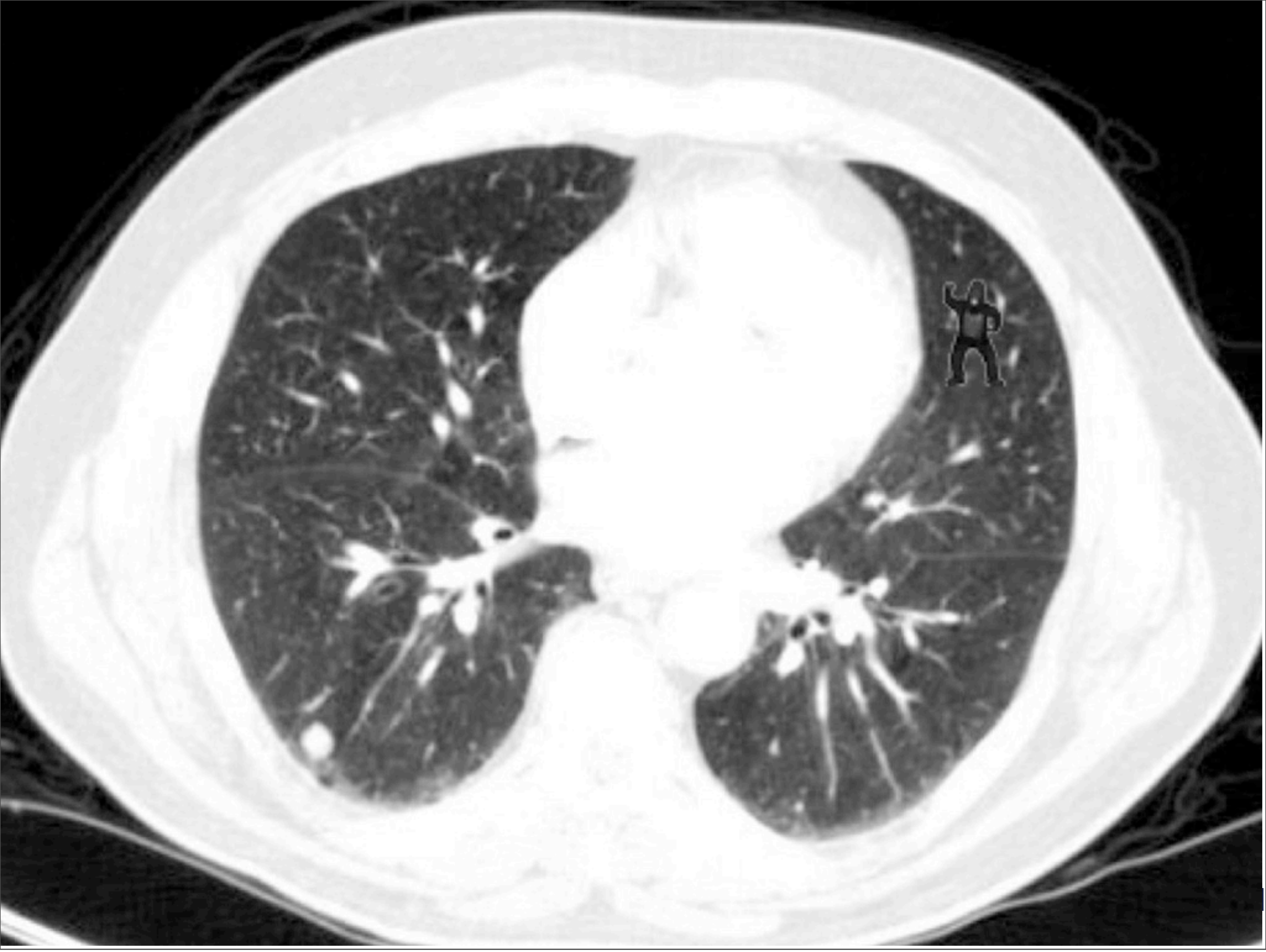
- Data Cleansing
  - Everyone's pain point
- Task Deconstruction
  - Good for code management
  - Hides -- in a good way -- data management

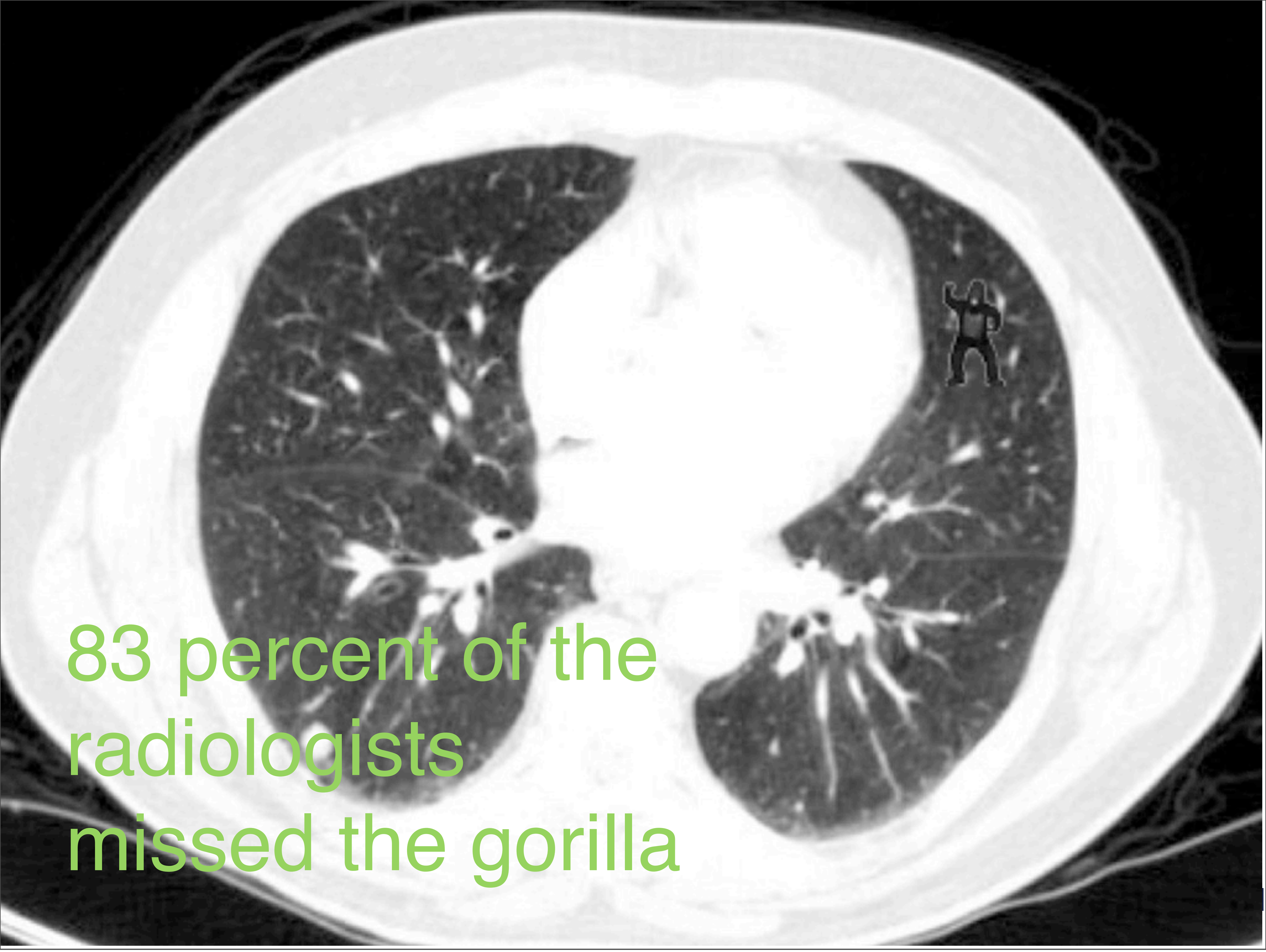
CONTINUUM
ANALYTICS

MapReduce Thoughts

- Data Cleansing
  - Everyone's pain point
- Task Deconstruction
  - Good for code management
  - Hides -- in a good way -- data management
- Can Be Inefficient
  - Network traffic
  - Job organization

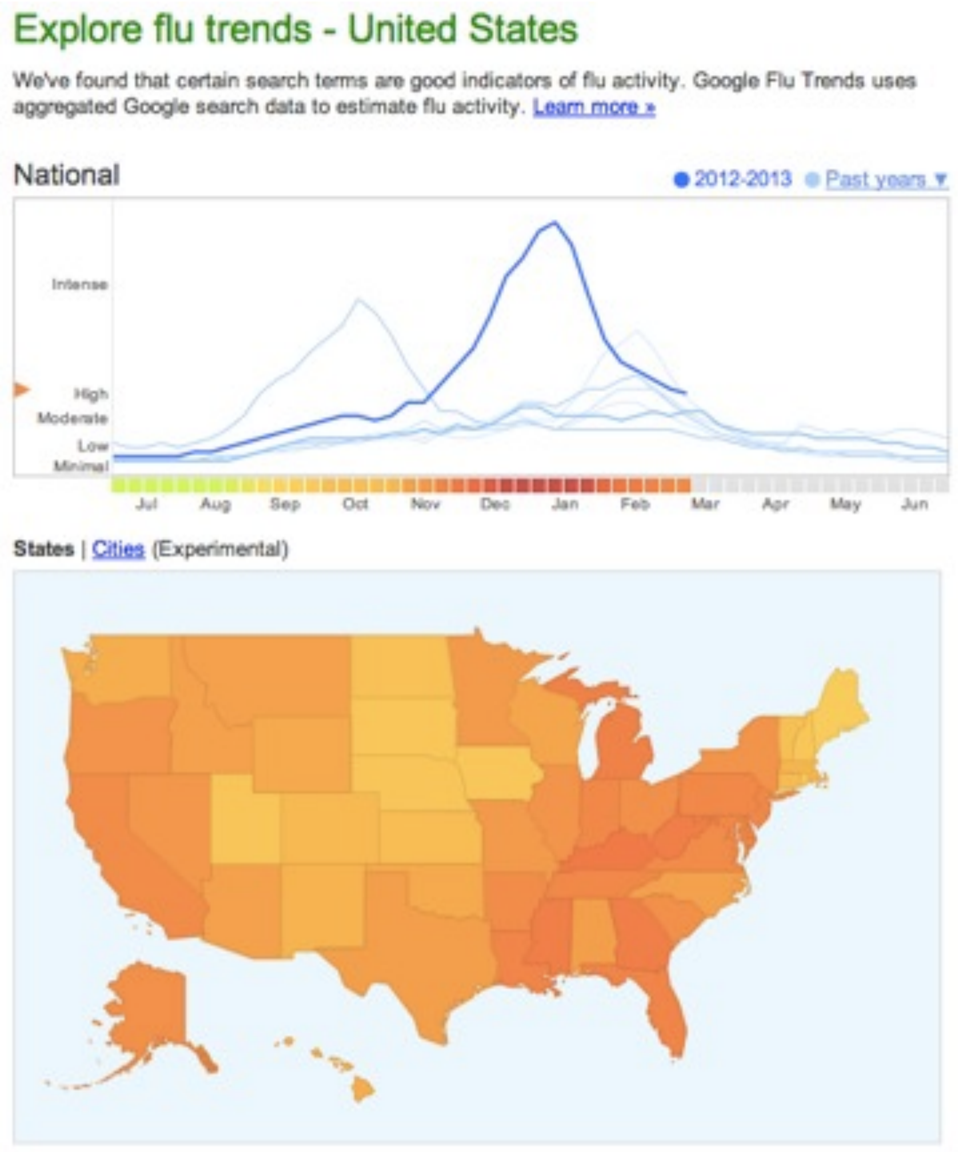
Data Thoughts





83 percent of the radiologists missed the gorilla

Google Flu


Explore flu trends - United States

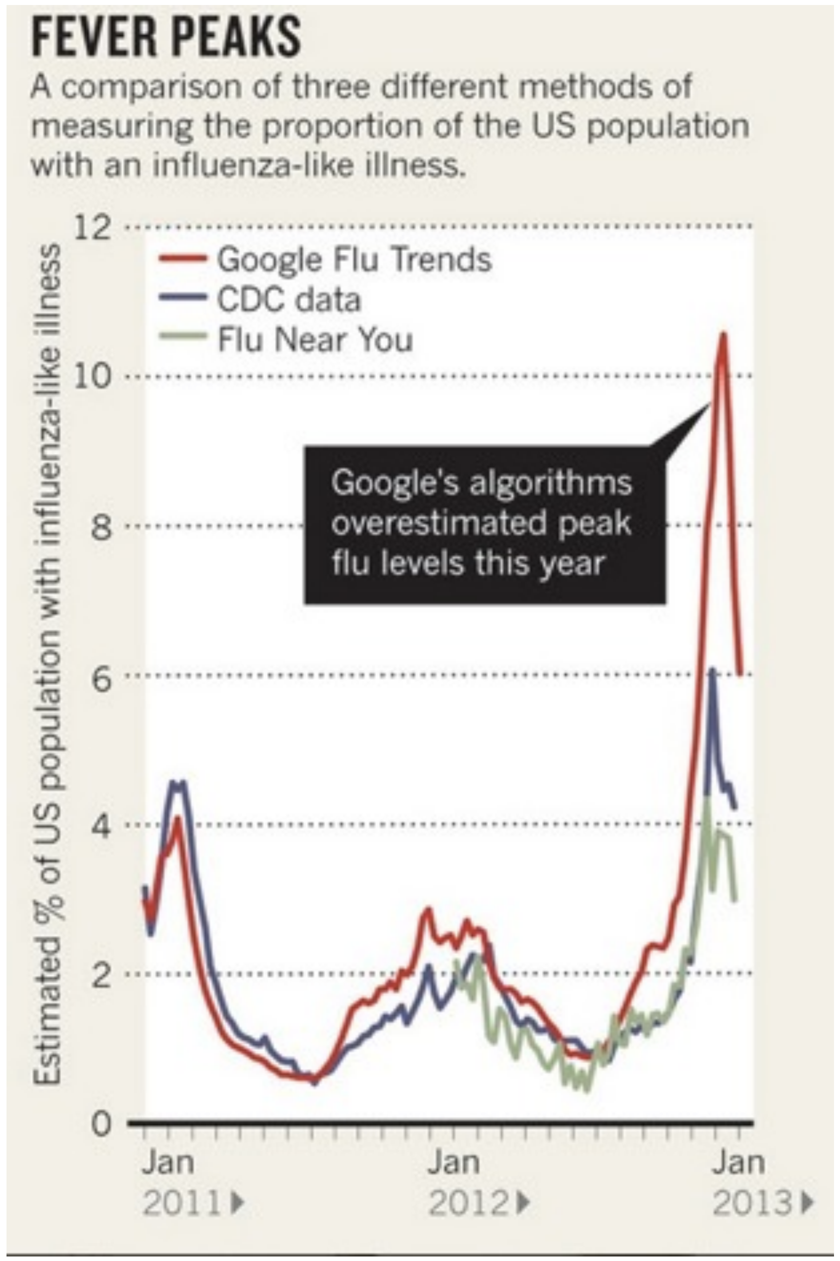
We've found that certain search terms are good indicators of flu activity. Google Flu Trends uses aggregated Google search data to estimate flu activity. Learn more »

National

- Data Mining
- Faster than CDC

CONTINUUM ANALYTICS

Google Get's Wrong



- Typically, prediction is great!
- This year not so much
- Google: No comment!
- Feedback mechanism from hype-up media

CONTINUUM ANALYTICS

Wrapping Up

- Invisible Gorillas will stay invisible
- Turnkey analytics is dangerous
- Good Analysis
    - Requires iterative exploration
    - Peer review and collaboration