BIONODE.IO

Introduction

github.com/bionode-hack/discussions

gitter.im/bionode-hack/discussions

WHAT

Modular and universal bioinformatics

WHAT

Modular and universal bioinformatics

- Tries to do one thing well

WHAT

Modular and universal bioinformatics

- Tries to do one thing well
- Provides highly reusable code and tools

WHAT

Modular and universal bioinformatics

- Tries to do one thing well
- Provides highly reusable code and tools
- Scales by using Streams

WHAT

Modular and universal bioinformatics

- Tries to do one thing well
- Provides highly reusable code and tools
- Scales by using Streams
- Runs everywhere

HOW

Using Node.js

HOW

Using Node.js

- Highly modular

HOW

Using Node.js

- Highly modular
- Very open community on GitHub

Modules count

Modules count

HOW

Using Node.js

- Highly modular
- Very open community on GitHub

HOW

Using Node.js

- Highly modular
- Very open community on GitHub
- Provides native implementation of Streams

HOW

Using Node.js

- Highly modular
- Very open community on GitHub
- Provides native implementation of Streams
- Run same JavaScript code on browser or CLI

Run same JavaScript code on browser or CLI

3. ↣ fish: fish /Users/bmpvieira/Local/bionode-fasta

```
    ✓ should return an array of Objects

  Use parser to read file by passing filename (add path to results)
    ✓ should return a Buffer for each sequence
    ✓ should return an Object for each sequence
    ✓ should return an Object for each sequence (shortcut version)

  Use parser to read file by passing filename and get results with call
results)
    ✓ callback error should be null
    ✓ should return a Buffer with all sequences objects separated by ne
    ✓ callback error should be null
    ✓ should return an array of Objects
    ✓ callback error should be null
    ✓ should return an array of Objects

  Use parser to read file by passing filename (add path to results) (re
    ✓ should return a Buffer for each sequence
    ✓ should return an Object for each sequence
    ✓ should return an Object for each sequence (shortcut version)

  Use parser to read file by passing filename and get results with call
results) (read gzipped file)
    ✓ callback error should be null
    ✓ should return a Buffer with all sequences objects separated by ne
    ✓ callback error should be null
    ✓ should return an array of Objects
    ✓ callback error should be null
    ✓ should return an array of Objects

  Errors should be caught
    ✓ should return a ENOENT error for non-existing path

  tests 40
  pass  40


  Pass!
~/L/bionode-fasta  feature/browser-testing-working  npm test
```

Run same JavaScript code on browser or CLI

3. ↣ fish: fish /Users/bmpvieira/Local/bionode-fasta

```
    ✓ should return an array of Objects

  Use parser to read file by passing filename (add path to results)
    ✓ should return a Buffer for each sequence
    ✓ should return an Object for each sequence
    ✓ should return an Object for each sequence (shortcut version)

  Use parser to read file by passing filename and get results with call
results)
    ✓ callback error should be null
    ✓ should return a Buffer with all sequences objects separated by ne
    ✓ callback error should be null
    ✓ should return an array of Objects
    ✓ callback error should be null
    ✓ should return an array of Objects

  Use parser to read file by passing filename (add path to results) (re
    ✓ should return a Buffer for each sequence
    ✓ should return an Object for each sequence
    ✓ should return an Object for each sequence (shortcut version)

  Use parser to read file by passing filename and get results with call
results) (read gzipped file)
    ✓ callback error should be null
    ✓ should return a Buffer with all sequences objects separated by ne
    ✓ callback error should be null
    ✓ should return an array of Objects
    ✓ callback error should be null
    ✓ should return an array of Objects

  Errors should be caught
    ✓ should return a ENOENT error for non-existing path

  tests 40
  pass  40


  Pass!
~/L/bionode-fasta  feature/browser-testing-working  npm test
```

Run same JavaScript code on browser or CLI

FASTA file format
> X-gene
ATGCGTACTGCATCATG
ACGTACTGCATTCATGC
GTGCTAGGGGTTTACGT

Run same JavaScript code on browser or CLI

FASTA file format
> X-gene
ATGCGTACTGCATCATG
ACGTACTGCATTCATGC
GTGCTAGGGGTTTACGT

JSON (JavaScript Object Notation)
{
"id":"X-gene",
"seq":"ATGCGTACTGCATCAT
GACGTACTGCATTCATGCGTG
CTAGGGGTTTACGT"
}

Run same JavaScript code on browser or CLI

localhost:49330/__testling

localhost:49330/__testling?show=true

```
# Use parser to read file by passing filename
ok 4 should return a Buffer for each sequence
ok 5 should return an Object for each sequence
ok 6 should return an Object for each sequence (shortcut version
# Use parser to read file by passing filename and get results wi
ok 7 callback error should be null
ok 8 should return a Buffer with all sequences objects separated
ok 9 callback error should be null
ok 10 should return an array of Objects
ok 11 callback error should be null
```

Elements Network Sources Timeline Profiles Resources Audits Cons

<top frame> Preserve log

```
# Use parser to read file by passing filename (add path to results) (rea
ok 31 should return a Buffer for each sequence
ok 32 should return an Object for each sequence
ok 33 should return an Object for each sequence (shortcut version)
# Use parser to read file by passing filename and get results with callb
ok 34 callback error should be null
ok 35 should return a Buffer with all sequences objects separated by new
ok 36 callback error should be null
ok 37 should return an array of Objects
ok 38 callback error should be null
ok 39 should return an array of Objects
# Errors should be caught
ok 40 should return a ENOENT error for non-existing path

1..40
# tests 40
# pass  40

# ok
```

Console Search Emulation Rendering

3. ↣ fish: fish /Users/bmpvieira/Local/bionode-fasta

```
  ✓ should return an array of Objects

 Use parser to read file by passing filename (add path to results)
  ✓ should return a Buffer for each sequence
  ✓ should return an Object for each sequence
  ✓ should return an Object for each sequence (shortcut version)

 Use parser to read file by passing filename and get results with call
results)
  ✓ callback error should be null
  ✓ should return a Buffer with all sequences objects separated by ne
  ✓ callback error should be null
  ✓ should return an array of Objects
  ✓ callback error should be null
  ✓ should return an array of Objects

 Use parser to read file by passing filename (add path to results) (re
  ✓ should return a Buffer for each sequence
  ✓ should return an Object for each sequence
  ✓ should return an Object for each sequence (shortcut version)

 Use parser to read file by passing filename and get results with call
results) (read gzipped file)
  ✓ callback error should be null
  ✓ should return a Buffer with all sequences objects separated by ne
  ✓ callback error should be null
  ✓ should return an array of Objects
  ✓ callback error should be null
  ✓ should return an array of Objects

 Errors should be caught
  ✓ should return a ENOENT error for non-existing path

 tests 40
 pass  40


 Pass!
~/L/bionode-fasta  feature/browser-testing-working  npm test
```

Run same JavaScript code on browser or CLI

localhost:49330/__testling

localhost:49330/__testling?show=true

```
# Use parser to read file by passing filename
ok 4 should return a Buffer for each sequence
ok 5 should return an Object for each sequence
ok 6 should return an Object for each sequence (shortcut version
# Use parser to read file by passing filename and get results wi
ok 7 callback error should be null
ok 8 should return a Buffer with all sequences objects separated
ok 9 callback error should be null
ok 10 should return an array of Objects
ok 11 callback error should be null
```

Elements Network Sources Timeline Profiles Resources Audits Cons

<top frame> Preserve log

```
# Use parser to read file by passing filename (add path to results) (rea
ok 31 should return a Buffer for each sequence
ok 32 should return an Object for each sequence
ok 33 should return an Object for each sequence (shortcut version)
# Use parser to read file by passing filename and get results with callb
ok 34 callback error should be null
ok 35 should return a Buffer with all sequences objects separated by new
ok 36 callback error should be null
ok 37 should return an array of Objects
ok 38 callback error should be null
ok 39 should return an array of Objects
# Errors should be caught
ok 40 should return a ENOENT error for non-existing path

1..40
# tests 40
# pass  40

# ok
```

Console Search Emulation Rendering

3. ↣ fish: fish /Users/bmpvieira/Local/bionode-fasta

```
  ✓ should return an array of Objects

 Use parser to read file by passing filename (add path to results)
  ✓ should return a Buffer for each sequence
  ✓ should return an Object for each sequence
  ✓ should return an Object for each sequence (shortcut version)

 Use parser to read file by passing filename and get results with call
results)
  ✓ callback error should be null
  ✓ should return a Buffer with all sequences objects separated by ne
  ✓ callback error should be null
  ✓ should return an array of Objects
  ✓ callback error should be null
  ✓ should return an array of Objects

 Use parser to read file by passing filename (add path to results) (re
  ✓ should return a Buffer for each sequence
  ✓ should return an Object for each sequence
  ✓ should return an Object for each sequence (shortcut version)

 Use parser to read file by passing filename and get results with call
results) (read gzipped file)
  ✓ callback error should be null
  ✓ should return a Buffer with all sequences objects separated by ne
  ✓ callback error should be null
  ✓ should return an array of Objects
  ✓ callback error should be null
  ✓ should return an array of Objects

 Errors should be caught
  ✓ should return a ENOENT error for non-existing path

 tests 40
 pass  40


 Pass!
~/L/bionode-fasta  feature/browser-testing-working  npm test
```

- Run same JavaScript code on browser or CLI

JavaScript not suitable for heavy scientific computation?

C++ preferred?

Run C++ from JavaScript

Icons made by Freepik and Icomoon from www.flaticon.com and this diagram made by BusFaster Ltd are licensed under

All code is under MIT.

#include nbind in C++ & call effortlessly from JavaScript without changes.

C++11 or later

ES5 / ES6 / TypeScript

compile to asm.js

compile to native binary

GCC
Clang
Visual Studio

asm.js

for native

fallback

native

run on major browsers

run on server or desktop

ORIGIN

During my PhD at

ORIGIN

During my PhD at

- Involved in biological web projects that need JS

Involved in biological web projects that need JS

wurmlab.github.io

Involved in biological web projects that need JS

wurmlab.github.io

Involved in biological web projects that need JS

wurmlab.github.io

GMOD

Involved in biological web projects that need JS

wurmlab.github.io

CHECK SEQUENCE TYPE

Takes a sequence string and checks if it's DNA, RNA or protein. Follows IUPAC notation which allows ambiguous sequence notation. In this case the sequence is labelled as ambiguous nucleotide rather than amino acid sequence.

```
seq.checkType("ATGACCCTGAGAAGAGCACCG");
=> "dna"
seq.checkType("AUGACCCUGAAGGUGAAUGAA");
=> "rna"
seq.checkType("MAYKSGKRPTFFEVFKAHCSDS");
=> "protein"
seq.checkType("AMTGACCCTGAGAAGAGCACCG");
=> "ambiguousDna"
seq.checkType("AMUGACCCUGAAGGUGAAUGAA");
=> "ambiguousRna"
```

```
seq.checkType = function(sequence) {
  var acgMatch = sequence.match(/[ACG]/i);
  var tMatch = sequence.match(/[T]/i);
  var nMatch = sequence.match(/[N]/i);
  var uMatch = sequence.match(/[U]/i);
  var potentialNucleotideMatch = sequence.match(/[WSMKRYBDHV]/i);
  var proteinMatch = sequence.match(/[EFIJLOPQZX\*]/i);
  if (proteinMatch) {
    return "protein";
  } else if (acgMatch && !potentialNucleotideMatch && !uMatch) {
    return "dna";
  } else if (acgMatch && potentialNucleotideMatch && !uMatch) {
    return "ambiguousDna";
  } else if (acgMatch && !potentialNucleotideMatch && uMatch && !tMatch) {
    return "rna";
  } else if (acgMatch && potentialNucleotideMatch && uMatch && !tMatch) {
    return "ambiguousRna";
  }
}
```

bionode-seq

REVERSE SEQUENCE

Takes sequence string and returns the reverse sequence.

```
seq.reverse("ATGACCCTGAAGGTGAA");
=> "AAGTGGAAGTCCCAGTA"
```

```
seq.reverse = function(sequence) {
  return sequence.split('').reverse().join('')
}
```

(REVERSE) COMPLEMENT SEQUENCE

Takes a sequence string and optional boolean for reverse, and returns its complement.

```
seq.complement("ATGACCCTGAAGGTGAA");
=> "TACTGGGACTTCCACTT"
seq.complement("ATGACCCTGAAGGTGAA", true);
=> "TTCACCTTCAGGGTCAT"
//Alias
seq.reverseComplement("ATGACCCTGAAGGTGAA");
=> "TTCACCTTCAGGGTCAT"
```

```
seq.complement = function(sequence, reverse) {
  var reverse = reverse || false
  var sequenceType = seq.checkType(sequence)
  var getComplementBase = seq.createComplementBase(sequenceType)
  if (reverse) {
    return sequence.split('').reverse().map(getComplementBase).join('')
  }
  else {
    return sequence.split('').map(getComplementBase).join('')
  }
}
```

ORIGIN

During my PhD at

- Involved in biological web projects that need JS

ORIGIN

During my PhD at

- Involved in biological web projects that need JS
- Had to find and get TB of data online

Compare genetic diversity of social vs solitary species

Compare genetic diversity of social vs solitary species

Compare genetic diversity of social vs solitary species

Maze of data sources

Maze of data sources

Had to find and get TB of data online

All Insects and Rodents

TOTAL: 167

Species w/ Genome @NCBI

Had to find and get TB of data online

Social (red) - Solitary (blue)

TOTAL: 167

Blue solitary, Red social

How to get the URLs for genomic dataset?

for example raw data of genome assembly:

ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000188075.1_Si_gnG

fish /home/bmpvieira/project-phd — docker • -bash — 131×42

```
~/project-phd  bionode-ncbi urls assembly ants | head -n 1 | json
```


```
{
  "uid": "140471",
  "structure": {
    "dir": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_assembly_structure/"
  },
  "report": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_assembly_report.txt"
  },
  "stats": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_assembly_stats.txt"
  },
  "genomic": {
    "fna": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_rna_from_genomic.fna.gz",
    "gbff": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_genomic.gbff.gz",
    "gff": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_genomic.gff.gz"
  },
  "table": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_feature_table.txt.gz"
  },
  "protein": {
    "faa": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_protein.faa.gz",
    "gpff": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_protein.gpff.gz"
  },
  "rm": {
    "out": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_rm.out.gz",
    "run": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_rm.run"
  },
  "wgsmaster": {
    "gbff": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_wgsmaster.gbff.gz"
  },
  "README": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/README.txt"
  },
  "hashes": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/annotation_hashes.txt"
  },
  "md5checksums": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/md5checksums.txt"
  }
}
```


```
~/project-phd
```

fish /home/bmpvieira/project-phd — docker • -bash — 131×42

```
~/project-phd  bionode-ncbi urls assembly ants | head -n 1 | json
{
  "uid": "140471",
  "structure": {
    "dir": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_assembly_structure/"
  },
  "report": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_assembly_report.txt"
  },
  "stats": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_assembly_stats.txt"
  },
  "genomic": {
    "fna": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_rna_from_genomic.fna.gz",
    "gbff": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_genomic.gbff.gz",
    "gff": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_genomic.gff.gz"
  },
  "table": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_feature_table.txt.gz"
  },
  "protein": {
    "faa": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_protein.faa.gz",
    "gpff": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_protein.gpff.gz"
  },
  "rm": {
    "out": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_rm.out.gz",
    "run": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_rm.run"
  },
  "wgsmaster": {
    "gbff": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/GCA_000611835.1_CerBir1.0_wgsmaster.gbff.gz"
  },
  "README": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/README.txt"
  },
  "hashes": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/annotation_hashes.txt"
  },
  "md5checksums": {
    "txt": "http://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000611835.1_CerBir1.0/md5checksums.txt"
  }
}
~/project-phd
```

bionode-ncbi


```
var bio = require('bionode')
```

bionode-ncbi


```
var bio = require('bionode')
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})

// Event pattern
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})

// Event pattern
bio.ncbi.urls('assembly', ‘ants’)
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})

// Event pattern
bio.ncbi.urls('assembly', ‘ants’)
.on('data', printGenomeURL)
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})

// Event pattern
bio.ncbi.urls('assembly', ‘ants’)
.on('data', printGenomeURL)
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})

// Event pattern
bio.ncbi.urls('assembly', ‘ants’)
.on('data', printGenomeURL)

function printGenomeURL(urls) {
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})

// Event pattern
bio.ncbi.urls('assembly', ‘ants’)
.on('data', printGenomeURL)

function printGenomeURL(urls) {
  console.log(urls[0].genomic.fna)
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})

// Event pattern
bio.ncbi.urls('assembly', ‘ants’)
.on('data', printGenomeURL)

function printGenomeURL(urls) {
  console.log(urls[0].genomic.fna)
})
```

bionode-ncbi


```
var bio = require('bionode')

// Callback pattern
bio.ncbi.urls('assembly', 'ants', function(urls) {
  console.log(urls[0].genomic.fna)
})

// Event pattern
bio.ncbi.urls('assembly', ‘ants’)
.on('data', printGenomeURL)

function printGenomeURL(urls) {
  console.log(urls[0].genomic.fna)
})
```

```
// Pipe pattern
```

bionode-ncbi


```
// Pipe pattern
var ncbi = require('bionode-ncbi')
```

bionode-ncbi


```
// Pipe pattern
var ncbi = require('bionode-ncbi')
var ndjson = require(‘ndjson')
```

bionode-ncbi


```
// Pipe pattern
var ncbi = require('bionode-ncbi')
var ndjson = require(‘ndjson')
```

bionode-ncbi


```
// Pipe pattern
var ncbi = require('bionode-ncbi')
var ndjson = require(‘ndjson')

ncbi.urls('genomes', 'ants')
```

bionode-ncbi


```
// Pipe pattern
var ncbi = require('bionode-ncbi')
var ndjson = require(‘ndjson')

ncbi.urls('genomes', 'ants')
.pipe(ndjson.stringify())
```

bionode-ncbi


```
// Pipe pattern
var ncbi = require('bionode-ncbi')
var ndjson = require(‘ndjson')

ncbi.urls('genomes', 'ants')
.pipe(ndjson.stringify())
.pipe(process.stdout)
```

bionode-ncbi


```
// Pipe pattern
var ncbi = require('bionode-ncbi')
var ndjson = require(‘ndjson')

ncbi.urls('genomes', 'ants')
.pipe(ndjson.stringify())
.pipe(process.stdout)
```

bionode-ncbi

bionode-ncbi

fish /home/bmpvieira — docker • -bash — 60×9

```
~ bionode-ncbi urls sra ants
```

bionode-ncbi

fish /home/bmpvieira — docker • -bash — 60×9

```
~ bionode-ncbi urls sra ants | json -ga0 url
```

bionode-ncbi

fish /home/bmpvieira — docker • -bash — 60×9

```
~ bionode-ncbi urls sra ants | json -ga0 url | head -n 1
```

bionode-ncbi

fish /home/bmpvieira — docker • -bash — 60×9

```
~ bionode-ncbi urls sra ants | json -ga0 url | head -n 1
http://ftp.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/
SRR/SRR327/SRR3272362/SRR3272362.sra
~
```

Had to find and get TB of data online

Social (red) - Solitary (blue)

TOTAL: 167

Blue solitary, Red social

Had to find and get TB of data online

Remove non-wild samples

Had to find and get TB of data online

Remove missing WGS

TOTAL: 88

Usable reads (genomic, wgs, etc)

Had to find and get TB of data online

Remove low coverage

Had to find and get TB of data online

Remove unusable for PSMC

TOTAL: 45

30x manually curated (no haploids, p
inbreds, etc)

ORIGIN

During my PhD at

- Involved in biological web projects that need JS
- Had to find and get TB of data online

ORIGIN

During my PhD at

- Involved in biological web projects that need JS
- Had to find and get TB of data online
- Had to build complicated bioinformatic pipelines

DATA PROCESSING

samples UIDs
search samples
dat samples
dat reads
start with "S. invicta"
search reads
triplify to sampleOf
papers UIDs
search papers
dat papers
link to papers
triplify to paperOfR
reads UIDs
triplify to projectOf
link to projects
projects UIDs
search projects
dat projects
assemblies UIDs
search assemblies
dat assemblies
link to assemblies
triplify to assemblyOf

Data analysis pipeline

Metadata pipeline

Metadata pipeline

`ncbi`

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)

fork1
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)

fork1
.pipe(tool.extractProperty('uid'))
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)

fork1
.pipe(tool.extractProperty('uid'))
.pipe(ncbi.link('sra', 'pubmed'))
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)

fork1
.pipe(tool.extractProperty('uid'))
.pipe(ncbi.link('sra', 'pubmed'))
.pipe(ncbi.search('pubmed'))
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)

fork1
.pipe(tool.extractProperty('uid'))
.pipe(ncbi.link('sra', 'pubmed'))
.pipe(ncbi.search('pubmed'))
.pipe(fork2)
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)

fork1
.pipe(tool.extractProperty('uid'))
.pipe(ncbi.link('sra', 'pubmed'))
.pipe(ncbi.search('pubmed'))
.pipe(fork2)
.pipe(dat.papers)
```

Metadata pipeline


```
ncbi
.search('sra', 'Solenopsis invicta')
.pipe(fork1)
.pipe(dat.reads)

fork1
.pipe(tool.extractProperty('expxml.Biosample.id'))
.pipe(ncbi.search('biosample'))
.pipe(dat.samples)

fork1
.pipe(tool.extractProperty('uid'))
.pipe(ncbi.link('sra', 'pubmed'))
.pipe(ncbi.search('pubmed'))
.pipe(fork2)
.pipe(dat.papers)
```

4. ↣ fish: fish /Users/bmpvieira/Local/bionode-pipelines/pipeline-fetch

```
~/L/b/pipeline-fetch ▶ node pipeline-fetch.js
```

4. ↣ fish: fish /Users/bmpvieira/Local/bionode-pipelines/pipeline-fetch

```
~/L/b/pipeline-fetch ▶ node pipeline-fetch.js
```

bionode-waterwheel

A Streaming Workflow Engine for bioinformatics and other big data explorations

Google Summer of Code 2016

OIBIF

```
const samples = task({
  input: {
    db: 'sra',
    accession: config.sraAccession
  },
  output: '**/*.sra'
}, ({ input }) => ncbi.download(input.db, input.accession) )

const fastqDump = task({
  input: new File('**/*.sra'),
  output: [1, 2].map(n => new File(`*_${n}.fastq.gz`))
}, ({ input }) => shell(`fastq-dump --split-files --skip-technical --gzip ${input}`) )
```

BIONODE COMMUNITY

WurmLab
.github.io

Queen Mary
University of London

DNA
digest

dat
DATA

BioJS

ACKNOWLEDGMENTS

Organisers

DNA digest

Community

bionode.io

Sponsor

repositive.io

Research group

WurmLab .github.io

Queen Mary University of London

Venue

CAMPUS LONDON

Friends

dat DATA