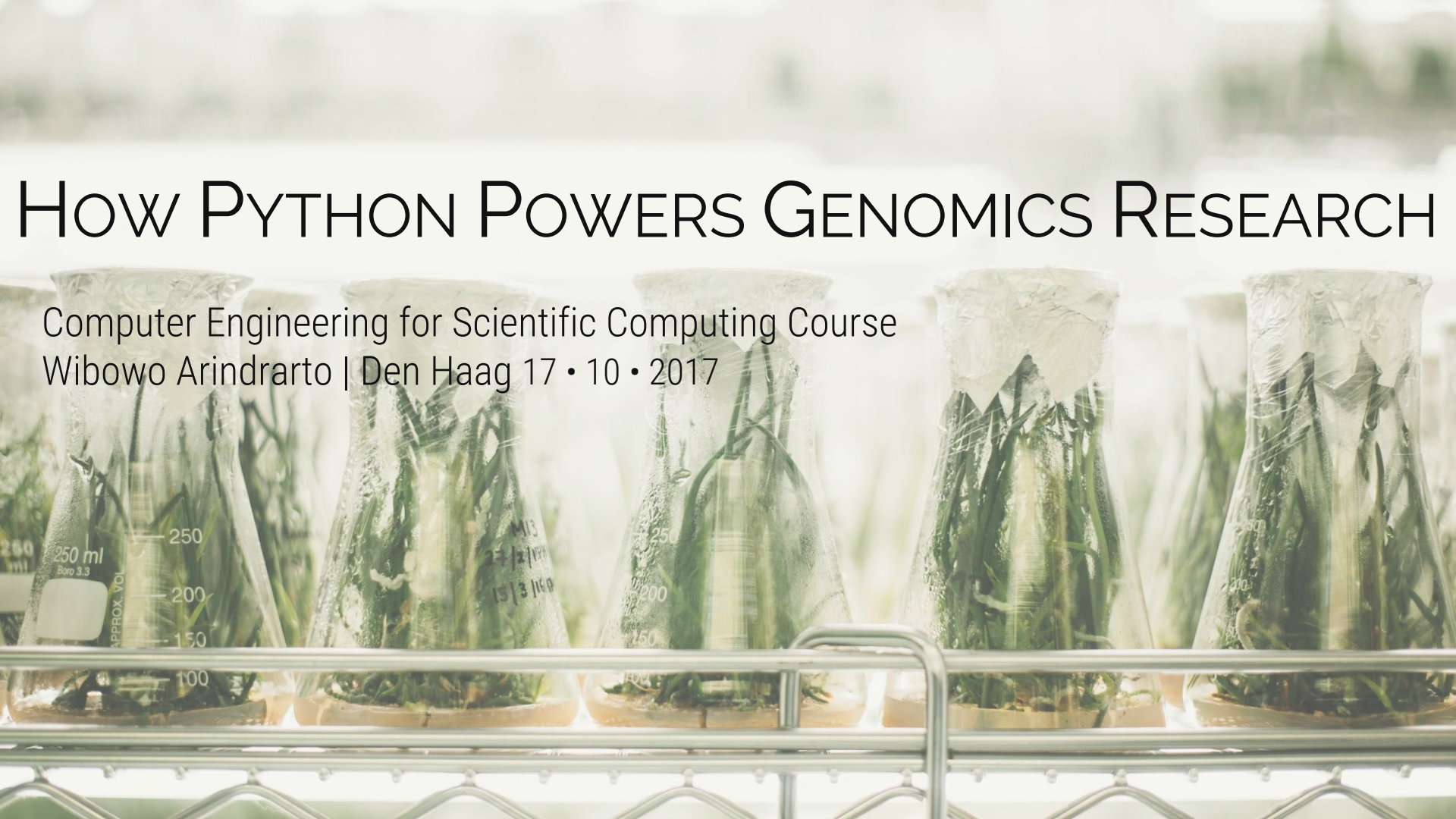
How Python Powers Genomics Research

Computer Engineering for Scientific Computing Course

Wibowo Arindrarto | Den Haag 17 • 10 • 2017



genomics, n.

Oxford English Dictionary

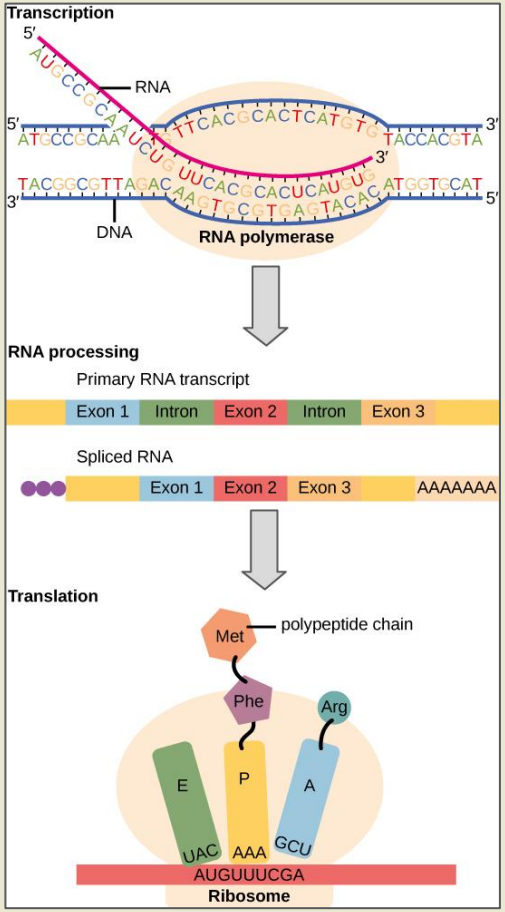
The branch of molecular biology concerned with the structure, function, evolution, and mapping of genomes.

Nature Publishing Group

The study of the full genetic complement of an organism (the genome). It employs recombinant DNA, DNA sequencing methods, and bioinformatics to sequence, assemble, and analyse the structure and function of genomes.

Photo by Alex Krivec

The Central Dogma

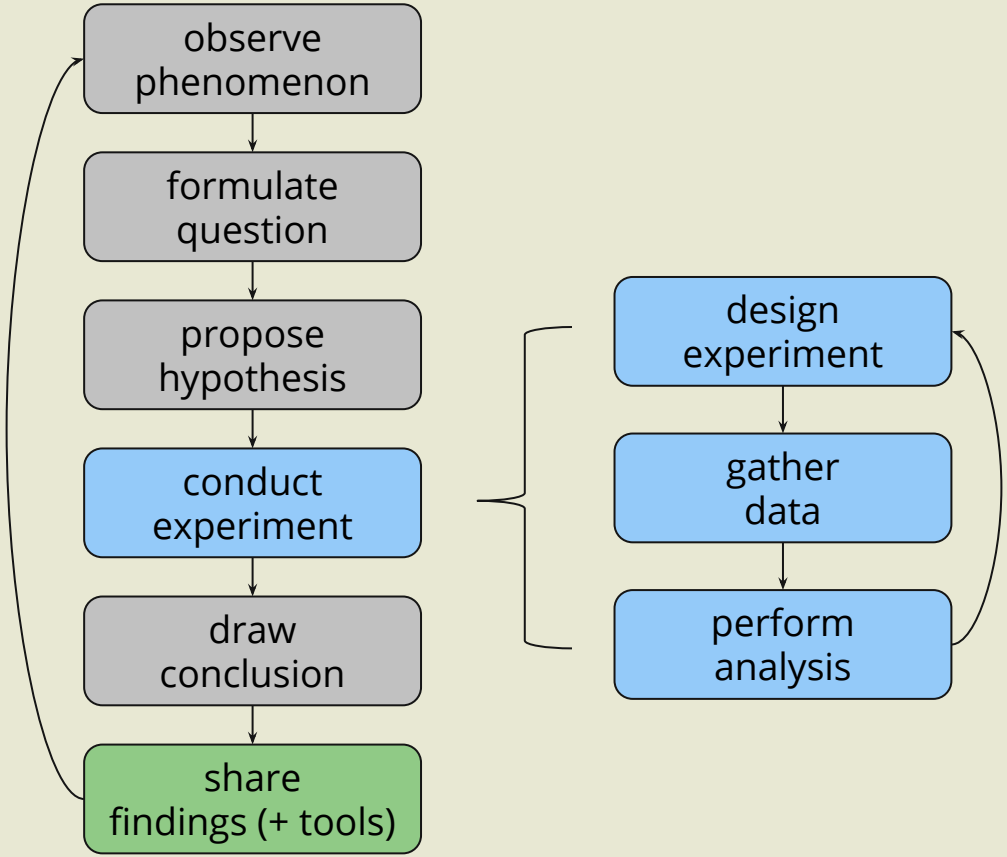


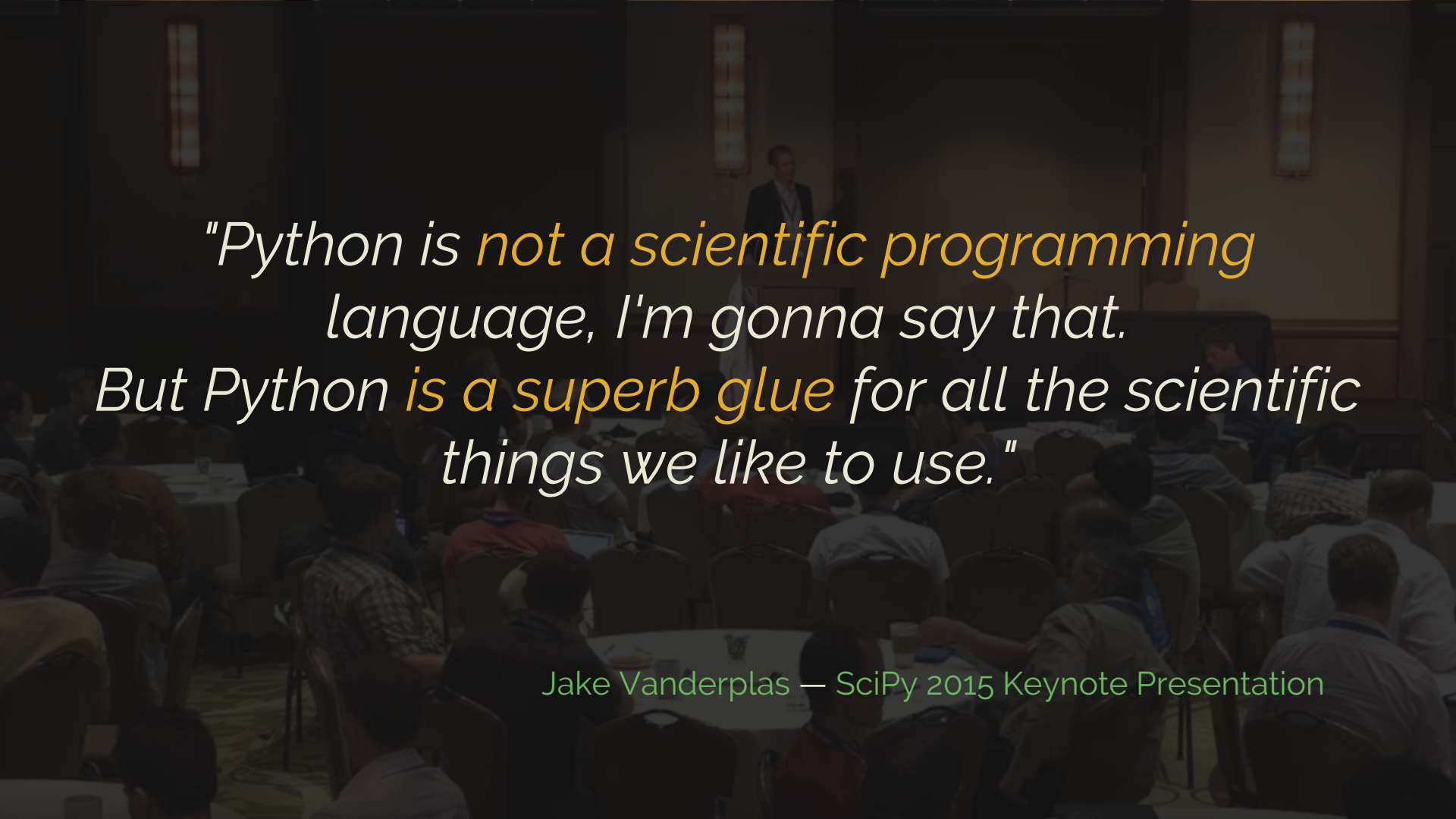
⇒ Genetic information is stored in sequences of chemical units:
- → nucleotides form DNA and RNA
- → amino acids form protein

⇒ The Central Dogma provides a framework of how information flows in biological systems
- → exceptions and nuances exist

Figure from the Biology OpenStax book

The Scientific Process: Where Python Fits





"Python is not a scientific programming language, I'm gonna say that.
But Python is a superb glue for all the scientific things we like to use."

Jake Vanderplas — SciPy 2015 Keynote Presentation

Gathering data

⇒ parsing various file formats

⇒ pulling data from remote resources

Analyzing results

⇒ performing computation
(interfacing with external library)

⇒ orchestrating jobs into analysis pipelines

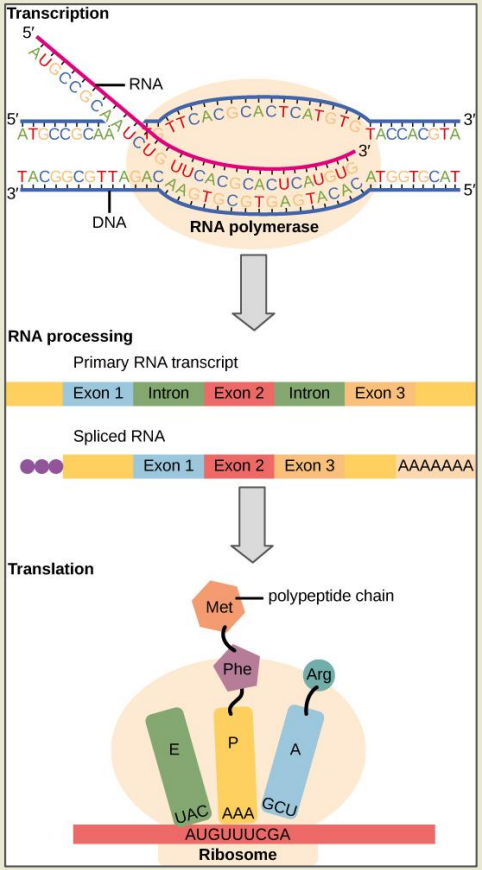
Data + results exploration

⇒ static & interactive visualization

Publishing + reproducibility

⇒ defining + setting up computing environments

Task: Count output of a sequencing run



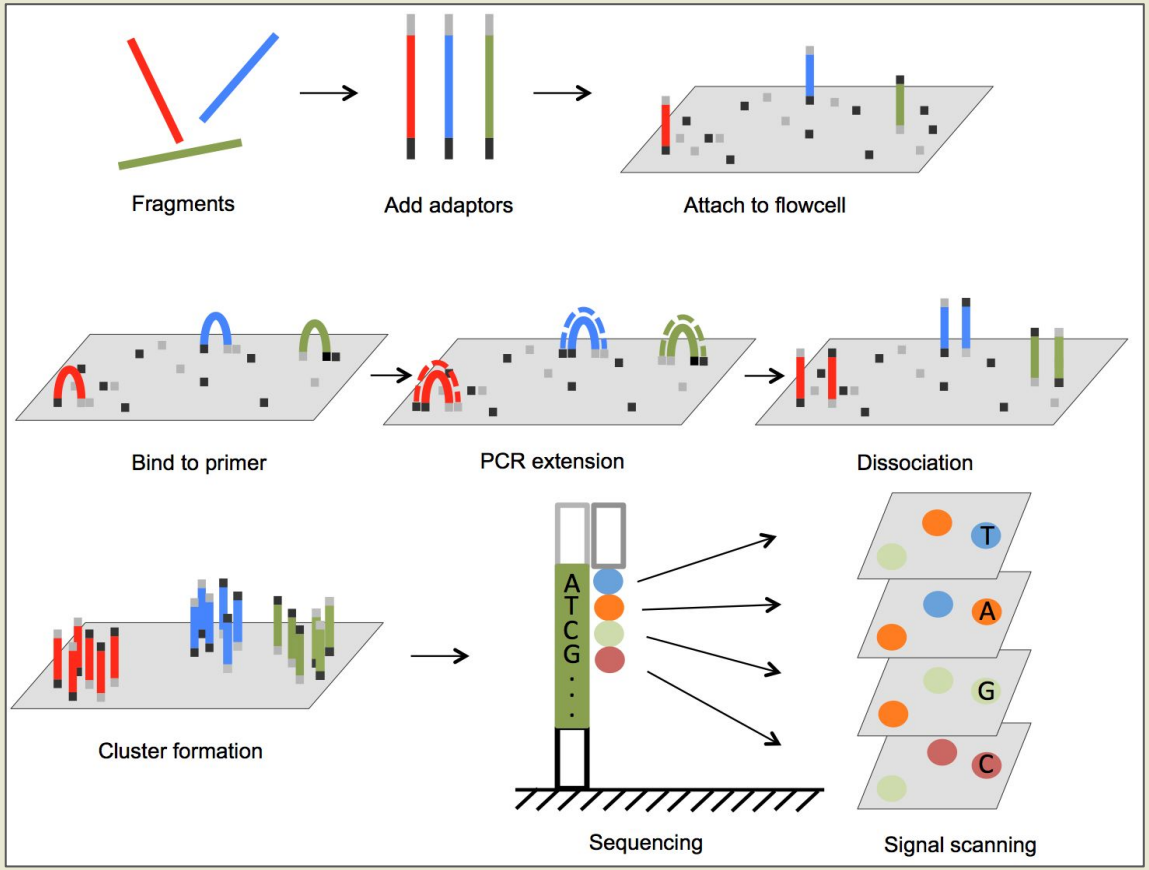


Figure from Lu, *et al.* (2016). doi: 10.5772/61657

Task: Count output of a sequencing run

FASTQ

one format for storing sequencing reads

```
@HISEQ:113:C6ALHANXX:5:1101:1491:2097 1:N:0:GTCCGC
CTGGCCTGGTGGGCGCGGTGACCATGGTCAGCTGNNNNNNNNNNNNCCTGACCCACCTCCN
+
CCCCCGGGGGGGCGDGGGGGGFGEFGFGGGGGGG############===EFGGGGGGEFF#
@HISEQ:113:C6ALHANXX:5:1101:1570:2114 1:N:0:GTCCGC
CCGCCATCTTCAGCAAACCCTGATGAAGGCTACAANNNNNNNNNAAGTACCCACGTAAAGA
+
CCBCCGGGGGGGGGGGGGFGBGGGGGGFEGGGGFG########===FGGGGGGGGGDFFG
@HISEQ:113:C6ALHANXX:5:1101:5598:2329 1:N:0:GTCCGC
CAATCACCTGGGCGCTGGAGGTGGCTTTGGCCCTGTAGCAGATGATGGCTATGGAGTTTCC
+
CCCCCFGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGGGGGEEGGGGGGGGGG=FDFG
@HISEQ:113:C6ALHANXX:5:1101:5695:2342 1:N:0:GTCCGC
GTAGCAAATTCACTAAACTTTTGTGTTCAGAGTTAAATTGTTCTCAGTACTTTCAATGTAG
+
3<<:0@FGGGGGGGGGGGCBGGFF:EFG1FFFF1:<CGGGGGCG:1E1=<BFDF<:@FGG
```

Task: Count output of a sequencing run

FASTQ

one format for storing sequencing reads

```
@HISEQ:113:C6ALHANXX:5:1101:1491:2097 1:N:0:GTCCGC
CTGGCCTGGTGGGCGCGGTGACCATGGTCAGCTGNNNNNNNNNNNNCCTGACCCACCTCCN
+
CCCCCGGGGGGGCGDGGGGGGFGEFGFGGGGGGG############===EFGGGGGGEFF#
@HISEQ:113:C6ALHANXX:5:1101:1570:2114 1:N:0:GTCCGC
CCGCCATCTTCAGCAAACCCTGATGAAGGCTACAANNNNNNNNNAAGTACCCACGTAAAGA
+
CCBCCGGGGGGGGGGGGGFGBGGGGGGFEGGGGFG########===FGGGGGGGGGDFFG
@HISEQ:113:C6ALHANXX:5:1101:5598:2329 1:N:0:GTCCGC
CAATCACCTGGGCGCTGGAGGTGGCTTTGGCCCTGTAGCAGATGATGGCTATGGAGTTTCC
+
CCCCCFGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGGGGGEEGGGGGGGGGG=FDFG
@HISEQ:113:C6ALHANXX:5:1101:5695:2342 1:N:0:GTCCGC
GTAGCAAATTCACTAAACTTTTGTGTTCAGAGTTAAATTGTTCTCAGTACTTTCAATGTAG
+
3<<:0@FGGGGGGGGGGGGCBGGFF:EFG1FFFF1:<CGGGGGCG:1E1=<BFDF<:@FGG
```

Task: Count output of a sequencing run

```python
import sys


fname = sys.argv[1]
count = 0
with open(fname, "r") as src:
    for idx, line in enumerate(src):
        if idx % 4 == 0 and line.startswith("@"):
            count += 1
print("There are", count, "records")
```

```
$ python count.py reads.fq↲
There are 4 records
```

Task: Count output of a sequencing run

```python
import sys


fname = sys.argv[1]
count = 0
with open(fname, "r") as src:
    for idx, line in enumerate(src):
        if idx % 4 == 0 and line.startswith("@"):
            count += 1
print("There are", count, "records")
# what about malformed files? what if we want to do more?
```

```
$ python count.py reads.fq↲
There are 4 records
```

Task: Count output of a sequencing run

https://biopython.org

```python
import sys
from Bio import SeqIO


fname = sys.argv[1]
count = 0
for _ in SeqIO.parse(fname, "fastq"):
    count += 1
print("There are", count, "records")
```

```
$ python count.py reads.fq↵
There are 4 records
```

Task: Count output of a sequencing run

https://biopython.org

```
import sys
from Bio import SeqIO


fname = sys.argv[1]
count = sum(1 for _ in SeqIO.parse(fname, "fastq"))
print("There are", count, "records")
```

```
$ python count.py reads.fq↵
There are 4 records
```

Task: Count output of a sequencing run

https://biopython.org

```
import sys
from Bio import SeqIO


fname = sys.argv[1]
count = sum(1 for record in SeqIO.parse(fname, "fastq")
            if "N" in record.seq)
print("There are", count, "records with ambiguous bases")
```

```
$ python count.py reads.fq↲
There are 2 records with ambiguous bases
```

Task: Parse various file formats

GenBank
annotated
sequence format

```
LOCUS       NM_000207                469 bp    mRNA    linear   PRI 03-OCT-2017
DEFINITION  Homo sapiens insulin (INS), transcript variant 1, mRNA.
ACCESSION   NM_000207
VERSION     NM_000207.2
KEYWORDS    RefSeq.
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
            Catarrhini; Hominidae; Homo.
...
FEATURES             Location/Qualifiers
...
     exon            43..246
                     /gene="INS"
                     /gene_synonym="IDDM; IDDM1; IDDM2; ILPR; IRDN; MODY10"
                     /inference="alignment:Splign:1.39.8"
ORIGIN
        1 agccctccag gacaggctgc atcagaagag gccatcaagc agatcactgt ccttctgcca
       61 tggccctgtg gatgcgcctc ctgcccctgc tggcgctgct ggccctctgg ggacctgacc
      121 cagccgcagc ctttgtgaac caacacctgt gcggctcaca cctggtggaa gctctctacc
      181 tagtgtgcgg ggaacgaggc ttcttctaca cacccaagac ccgccgggag gcagaggacc
      241 tgcaggtggg gcaggtggag ctgggcgggg gccctggtgc aggcagcctg cagcccttgg
      301 ccctggaggg gtccctgcag aagcgtggca ttgtggaaca atgctgtacc agcatctgct
      361 ccctctacca gctggagaac tactgcaact agacgcagcc cgcaggcagc cccacacccg
      421 ccgcctcctg caccgagaga gatggaataa agcccttgaa ccagcaaaa
//
```

https://www.ncbi.nlm.nih.gov/nuccore/NM_000207

Task: Parse various file formats from external resources

```
LOCUS       NM_000207                469 bp    mRNA    linear   PRI 03-OCT-2017
DEFINITION  Homo sapiens insulin (INS), transcript variant 1, mRNA.
...
```

GenBank
annotated
sequence format

biopython

```python
#!/usr/bin/env python

import sys
from Bio import Entrez, SeqIO

# not compulsory but recommended
Entrez.email = "mail@site.net"

gene_id = sys.argv[1]
handle = Entrez.efetch(db="nucleotide", id=gene_id, rettype="gb", retmode="text")
record = SeqIO.read(handle, "genbank")
print("Record of {!r}".format(record.description),
      "has", len(record.seq), "nucleotides",
      "and", sum(1 for ft in record.features if ft.type == "exon"), "exons")
```

```
$ python fetch_and_parse.py NM_000207↲
Record of 'Homo sapiens insulin (INS), transcript variant 1, mRNA' has 469 nucleotides and 3 exons
```

Task: Parse various file formats

SAM/BAM
genome alignment format

```
@HD VN:1.5 SO:coordinate
@SQ SN:chrQ LN:45
r002  99  ref  7  30  8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG EFGGFGGFGFGFGFEEE
r002 147  ref 37  30  9M         =  7 -39 CAGCGGCAT         GEFEEFGGE NM:i:1
....
```

VCF
variant call format

```
##fileformat=VCFv4.1
##INFO=<ID=HOM,Number=1,Type=Integer,Description="Number of samples called homozygous-variant">
##INFO=<ID=NC,Number=1,Type=Integer,Description="Number of samples not called">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO          FORMAT  Sample1
chr1    14464   .       A       T       .       PASS    HOM=1;NC=0    GT:GQ   1/1:184
chr1    633365  .       C       CCA     .       PASS    HOM=1;NC=0    GT:GQ   1/1:145
...
```

Task: Parse various file formats + interface with faster external libraries

SAM/BAM
genome alignment format

```
@HD VN:1.5 SO:coordinate
@SQ SN:chrQ LN:45
r002  99  ref  7  30  8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG EFGGFGGFGFGFGFEEE
r002 147  ref 37  30  9M         =  7 -39 CAGCGGCAT         GEFEEFGGE NM:i:1
....
```

⇒ has its own compression and encoding format

⇒ unpacking requires considerably more CPU

⇒ parseable using pysam, a wrapper around htslib, a C library.

⇒ pysam uses Cython to talk to htslib and define its own functions

⇒ Cython is also used in some parts of numpy

Task: Interface with faster external libraries

```python
# pure Python
def fib(n):
    a, b = 0.0, 1.0
    for i in range(n):
        a, b = a + b, a
    return a
```

```
fib(0) : 590 ns    (1x)
fib(90): 12,852 ns (1x)
```

```cython
# pure Cython
def fib(int n):
    cdef int i
    cdef double a = 0.0, b = 1.0
    for i in range(n):
        a, b = a + b, a
    return a
```

```
fib(0) : 90 ns  (~6.5x)
fib(90): 258 ns (~49.8x)
```

```c
// pure C
double cfib(int n) {
  int i;
  double a = 0.0, b = 1.0, tmp;
  for (i = 0; i < n; ++i) {
    tmp = a; a = a + b; b = tmp;
  }
  return a;
}
```

```
fib(0) : 2 ns   (295x)
fib(90): 164 ns (~78.3x)
```

```cython
# Cython wrapping C
cdef extern from "fib.h":
    double cfib(int n)


def fib(int n):
    return cfib(n)
```

Code and benchmarks adapted from the Cython - A Guide for Python Programmers book by Kurt W. Smith

Task: Interface with faster external libraries

```
# pure Python
def fib(n):
    a, b = 0.0, 1.0
    for i in range(n):
        a, b = b, a + b
    return a
```

python

fib(0) : 590 ns (1x)
fib(90): 12,852 ns (1x)

```
# pure Cython
def fib(int n):
    cdef int i;
    cdef double a = 0.0, b = 1.0, tmp;
    for i in range(n):
        a, b = b, a + b
    return a
```

cython → compiled to → C → compiled to → .so → imported by → python

fib(0) : 90 ns (~6.5x)
fib(90): 258 ns (~49.8x)

```
// pure C
double cfib(int n) {
  int i;
  double a = 0.0, b = 1.0, tmp;
  for (i = 0; i < n; ++i) {
    tmp = a; a = a + b; b = tmp;
  }
  return a;
}
```

C → x → python

fib(0) : 2 ns (295x)
fib(90): 164 ns (~78.3x)

```
# Cython wrapping C
cdef extern from "fib.h":
    double cfib(int n)

def fib(int n):
    return cfib(n)
```

python

Task: Interface with faster external libraries

Sequence alignment using the Smith-Waterman algorithm

	ϵ	T	A	T	A	T	G	C	G	G	C	G	T	T	T
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	0	2	2	0	2	0	0	0
G	0	0	0	0	0	0	2	0	2	4	0	2	0	0	0
T	0	2	0	2	0	2	0	0	0	0	0	0	4	2	2
A	0	0	4	0	4	0	0	0	0	0	0	0	0	0	0
T	0	2	0	6	0	6	0	0	0	0	0	0	2	2	2
G	0	0	0	0	2	0	8	2	2	2	0	2	0	0	0
C	0	0	0	0	0	0	2	10	4	0	4	0	0	0	0
T	0	2	0	2	0	2	0	4	6	0	0	0	2	2	2
G	0	0	0	0	0	0	4	0	6	8	2	2			
G	0	0	0	0	0	0	2	0	2	8	4	4			
C	0	0	0	0	0	0	0	4	0	2	10	4			
G	0	0	0	0	0	0	2	0	6	2	4	12	6	0	0
C	0	0	0	0	0	0	0	4	0	2	4	6	8	2	0
T	0	2	0	2	0	2	0	0	0	0	0	0	8	10	4
A	0	0	4	0	4	0	0	0	0	0	0	0	2	4	6

```
y :  TATATGC-GGCGTTT
        |||||  ||||
x :  GGTATGCTGGCGCTA
```

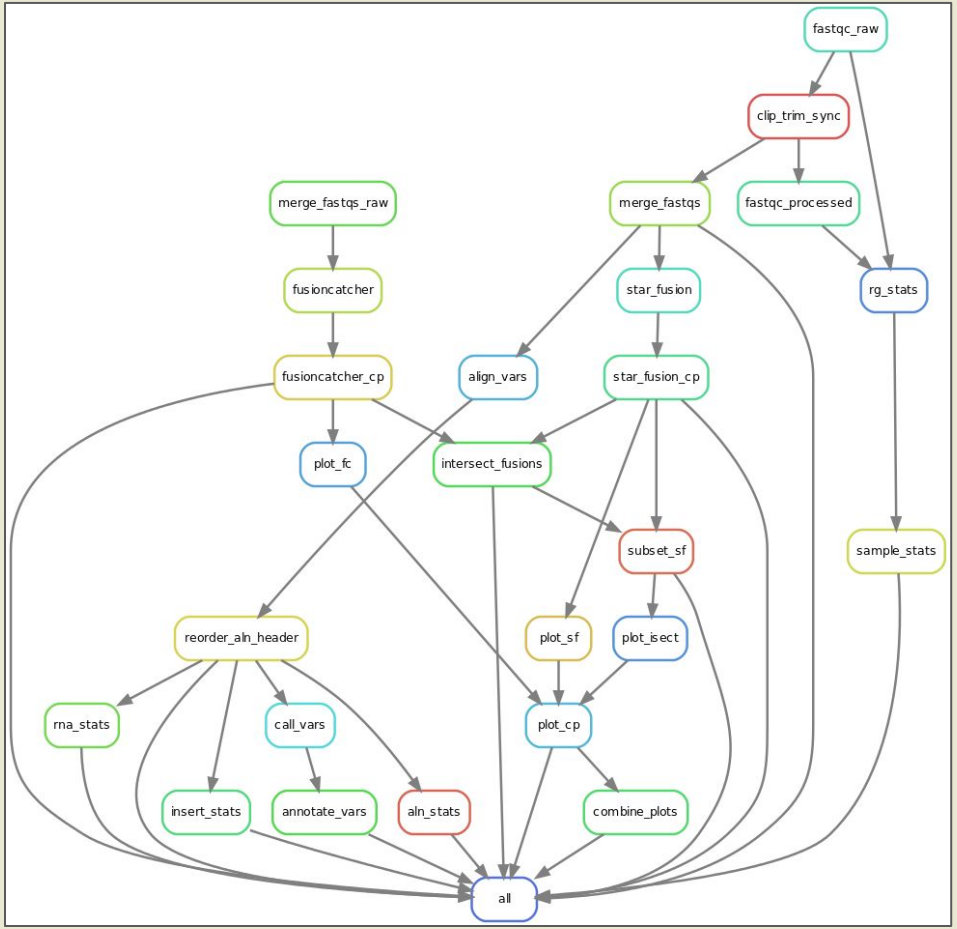
$s(a, b)$

	A	C	G	T	-
A	2	-4	-4	-4	-6
C	-4	2	-4	-4	-6
G	-4	-4	2	-4	-6
T	-4	-4	-4	2	-6
-	-6	-6	-6	-6	

cutadapt, a Python command-line tool for cleaning reads uses Cython to implement this algorithm

Figure taken from teaching materials by Ben Langmead

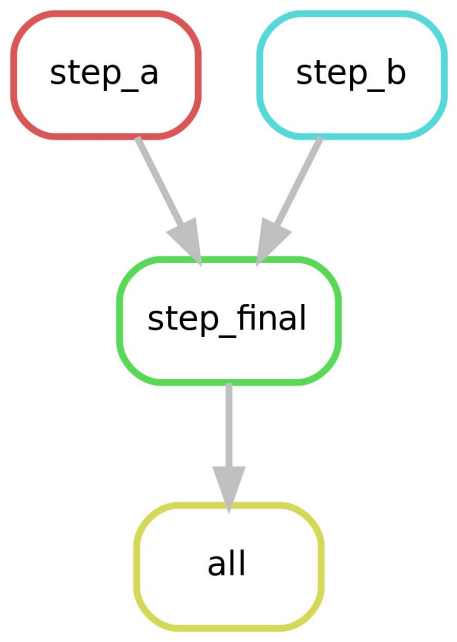
Task: Orchestrate jobs into data analysis pipelines



- ⇒ (relatively) simple workflow that analyzes RNA-seq samples of leukemia patients
- ⇒ nodes indicate execution of a command-line tool
- ⇒ edges indicate job dependencies
- ⇒ real world use involves many more steps and is run on multiple inputs

Task: Orchestrate jobs into data analysis pipelines

Snakemake

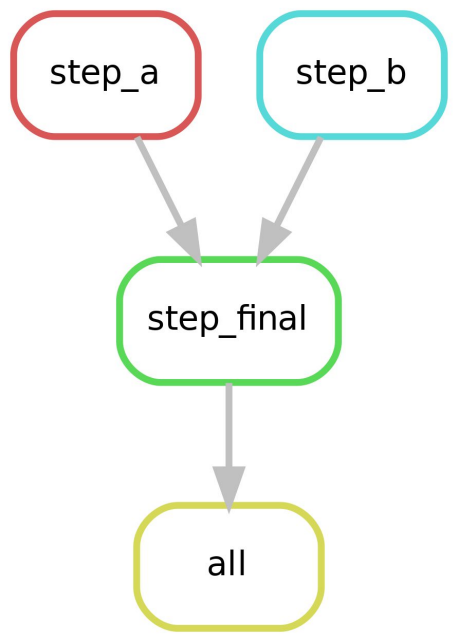



```
rule all:
    input:
        "final_output.txt"

rule step_a:
    output:
        "output_a.txt"
    shell:
        """echo 'step a' > {output}"""

rule step_b:
    output:
        "output_b.txt"
    shell:
        """echo 'step b' > {output}"""

rule step_final:
    input:
        first="output_a.txt",
        second="output_b.txt"
    output:
        "final_output.txt"
    shell:
        """cat {input.first} {input.second} > {output}"""
```

https://snakemake.readthedocs.io

Task: Orchestrate jobs into data analysis pipelines

Snakemake




```
rule all:
    input:
        "final_output.txt"

rule step_a:
    output:
        "output_a.txt"
    shell:
        """echo 'step a' > {output}"""
```

```
@workflow.rule(name="all", lineno=1, snakefile="/path/to/Snakefile")
@workflow.input("final_output.txt")
@workflow.norun()
@workflow.run
def __rule_all(input, output, params, ...):
    pass


@workflow.rule(name="step_a", lineno=11, snakefile="/path/to/Snakefile")
@workflow.output("output_a.txt")
@workflow.shellcmd("""echo 'step a' > {output}""")
@workflow.run
def __rule_all(input, output, params, ...):
    shell("""echo 'step a' > {output}""", bench_record=bench_record)
```

https://snakemake.readthedocs.io

Task: Explore and visualize results

Jupyter Notebook + pandas

jupyter demo Last Checkpoint: Last Tuesday at 2:21 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Code

In [1]:
```
%pylab --no-import-all inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
import pandas as pd

df = pd.read_csv("metrics.tsv", sep="\t")
df
```

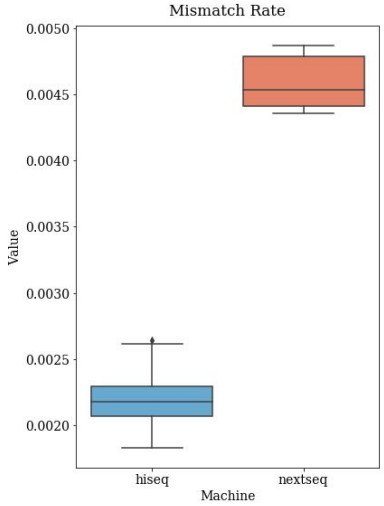
Out[2]:

	BAD_CYCLES	CATEGORY	LIBRARY	MEAN_READ_LENGTH	PCT_ADAPTER	PCT_CHIMERAS	PCT_PF_READS	PCT_PF_READS_ALIGNED	F
0	0	PAIR	NaN	123.808637	0.000018	0.003954	1	0.955509	
1	0	PAIR	NaN	123.275902	0.000044	0.003001	1	0.933713	
2	0	PAIR	NaN	123.761940	0.000056	0.002839	1	0.949851	
3	0	PAIR	NaN	124.171084	0.000073	0.002497	1	0.923459	
4	0	PAIR	NaN	123.907422	0.000061	0.002907	1	0.952802	
5	0	PAIR	NaN	123.680499	0.000216	0.003006	1	0.942098	
6	0	PAIR	NaN	123.871832	0.000019	0.003200	1	0.945502	
7	0	PAIR	NaN	123.921686	0.000032	0.002847	1	0.944738	
8	0	PAIR	NaN	123.522577	0.000228	0.002555	1	0.955324	
9	0	PAIR	NaN	123.548251	0.000080	0.003031	1	0.952169	
10	0	PAIR	NaN	123.948549	0.000063	0.002366	1	0.943659	

Task: Explore and visualize results

Jupyter Notebook + pandas + seaborn / matplotlib

In [3]:
```
import matplotlib
import seaborn as sns

def compare(y, ylabel="Value", title="Comparison by Machine", title_dist=1.01):
    font = {'weight' : 'normal', 'size': 14, 'family': 'serif'}
    matplotlib.rc('font', **font)
    pylab.rcParams['figure.figsize'] = (6.0, 9.0)

    ax = sns.boxplot(x="MACHINE", y=y, data=df)
    boxes = ax.artists
    for i, box in enumerate(boxes):
        if i == 0:
            box.set_facecolor("#68AACF")
        else:
            box.set_facecolor("#E58368")
    ax.set(ylabel=ylabel, xlabel="Machine")
    if title is not None:
        ax.set_title(title, y=title_dist)

compare("PF_MISMATCH_RATE", title="Mismatch Rate")
```



Task: Explore and visualize results

Jupyter Notebook + pandas + seaborn / matplotlib + ipywidgets

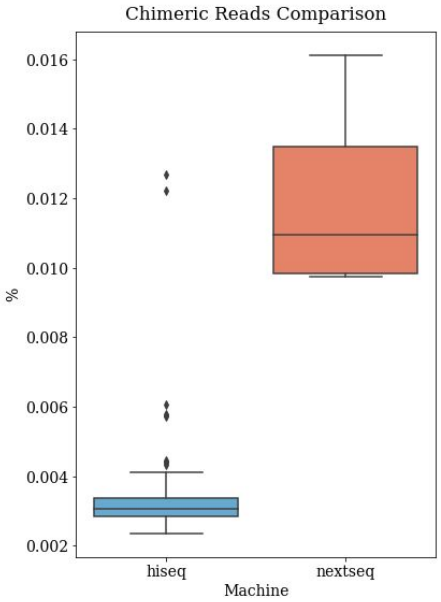
```
In [4]: from ipywidgets import interact
        _ = interact(compare, y=list(df.columns), ylabel="Value", title="Comparison Plot", title_dist=(1, 1.1, 0.01))
```

y: PCT_CHIMERAS

ylabel: %

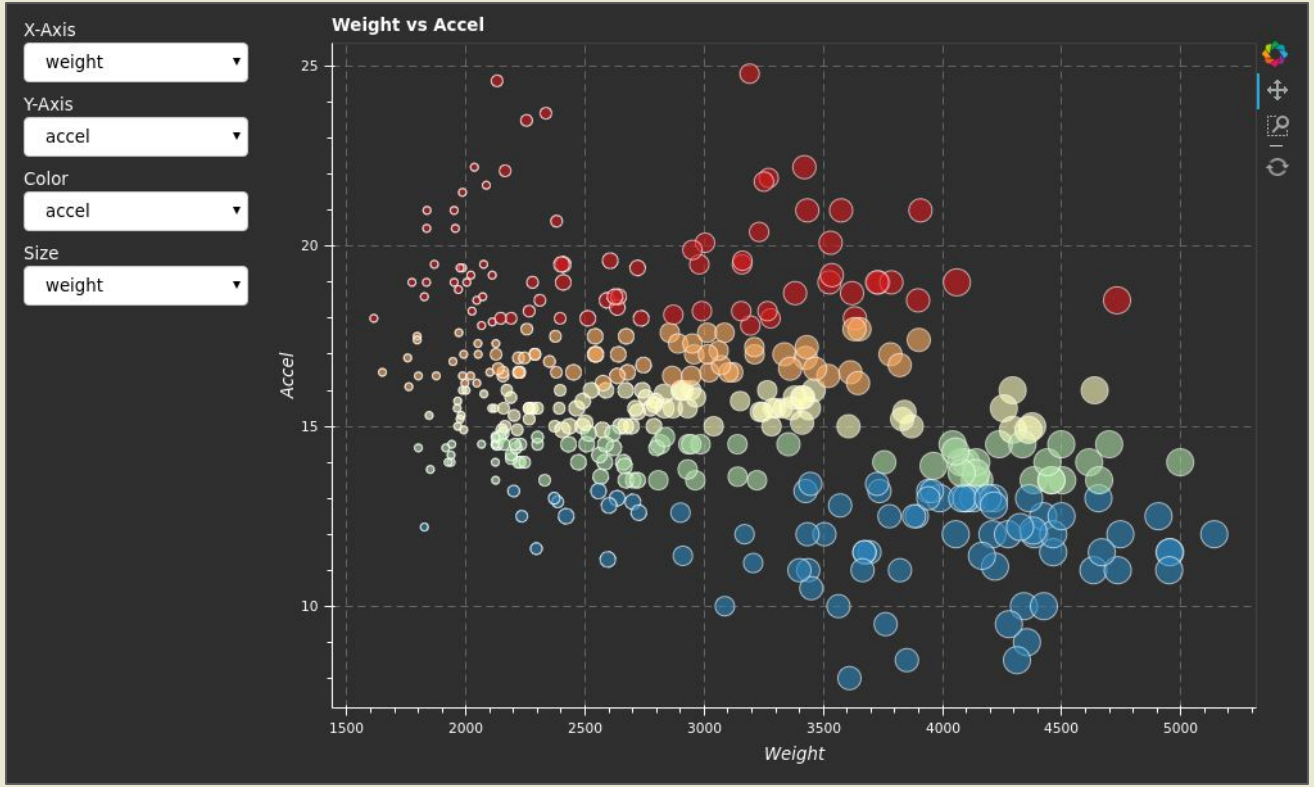
title: Chimeric Reads Comparison

title_dist: 1.01



Task: Explore and visualize results

Bokeh for exposing interactive plots



https://github.com/bokeh/bokeh/tree/master/examples/app/crossfilter

Task: Publish & share tools

BIOCONDA®

https://bioconda.github.io/

Problem

⇒ Tools and scripts often complex dependencies

⇒ "*It runs on my machine!*"

(bio)conda

⇒ Applies ideas from Python virtualenv to generic command-line tools and libraries

⇒ bioconda is one distribution channel built on conda (which is built with Python)

⇒ "`conda install my_awesome_package`"

Highlights

⇒ YAML for defining environments

⇒ YAML for defining build steps

⇒ automated deployment using a continuous integration platform

⇒ automated Docker container builds

Production Use: Ebola Virus Evolution

Cell

CellPress

Volume 161, Issue 7, 18 June 2015, Pages 1516-1526

Article

Ebola Virus Epidemiology, Transmission, and Evolution during Seven Months in Sierra Leone

Daniel J. Park [1, 21], Gytis Dudas [2, 21], Shirlee Wohl [1, 3, 21], Augustine Goba [4, 21], Shannon L.M. Whitmer [5, 21], Kristian G. Andersen [6], Rachel S. Sealfon [1, 7], Jason T. Ladner [8], Jeffrey R. Kugelman [8], Christian B. Matranga [1], Sarah M. Winnicki [1, 3], James Qu [1], Stephen K. Gire [1, 3], Adrianne Gladden-Young [1], Simbirie Jalloh [4], Dolo Nosamiefan [1], Nathan L. Yozwiak [1, 3], Lina M. Moses [9] ... Pardis C. Sabeti [1, 3, 22]

Source Code: https://github.com/broadinstitute/viral-ngs

- ⇒ analysis pipeline developed using Snakemake [source]
- ⇒ parts of analysis published as Python notebooks [source]
- ⇒ various Python scripts published in GitHub
- ⇒ package available in Bioconda

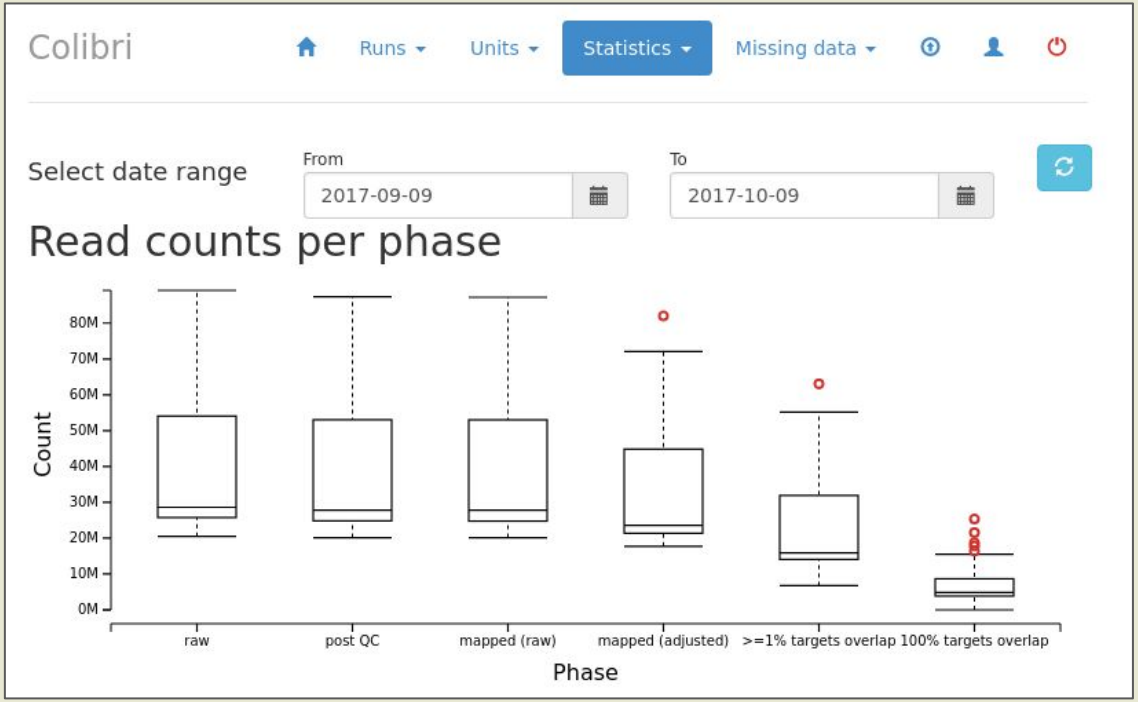
Park DJ, *et al.* (2015). doi:10.1016/j.cell.2015.06.007

Production Use: IsoSeq Data Exploration

- ⇒ web service for exploring feature correlation data
- ⇒ Uses the Flask web framework + Bokeh visualization framework

TranscriptGraph - developed by Guy Allard for Anvar *et al.* (2017). doi.org/10.1101/165092

Production Use: Automated Pipeline Monitoring



⇒ web service for monitoring and launching data analysis pipeline runs

⇒ uses the Flask web framework

Colibri - developed by Wibowo Arindrarto & Sander Bollen
for the Clinical Genetics Department, LUMC

Thank You!

Credits

⇒ Front photo by chuttersnap

⇒ This photo by Wayne Robinson