

Next-Generation Genomics Analysis Using Spark and ADAM

Timothy Danford

AMPLab, Tamr Inc.



1. To learn something about *your* genome...



2. Take a sample...



3. Put it in a sequencer...



4. Which produces a text file!



```
>read1  
AATGACCGATAGAAA  
>read2  
AATGACTCACCATAA  
>read3  
TCGACGATAATTTAC
```

5. "Bioinformatics" is the analysis which computationally *reverses* this process and tells us about your genome!

Bioinformatics **today** is workflows and files

Sequencing: **clustered**

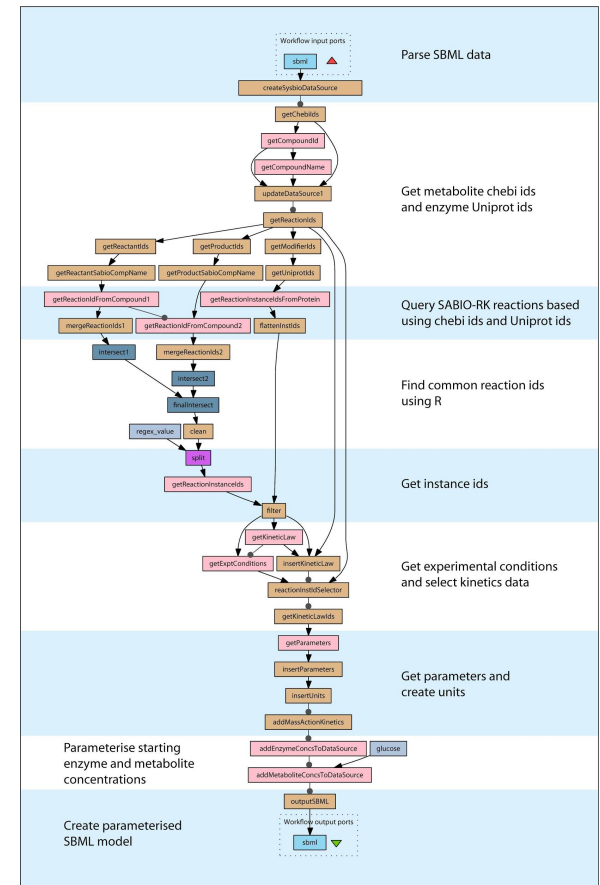
Data size: terabytes-to-**petabytes**

Location: local disk, **AWS**

Updates: periodic, **batch** updates

Sharing: **copying**

- Intermediate files are often retained
- Most data is stored in custom file formats
- “Workflow” and “pipeline” systems are everywhere





Tony Cox
@The_Soup_Dragon

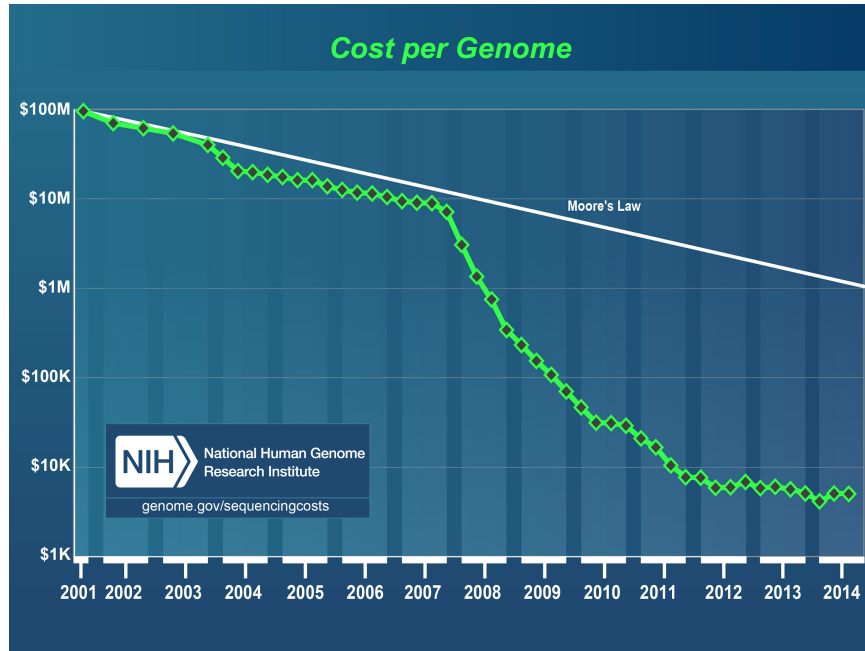


Follow

Not wishing to be outdone by Amazon, Sanger Institute develops drone delivery system for BAM files.



Bioinformatics **tomorrow** will (need to) be distributed, incremental



<http://www.genome.gov/sequencingcosts/>

Sequencing: **ubiquitous**

Data size: petabytes-to-**exabytes?**

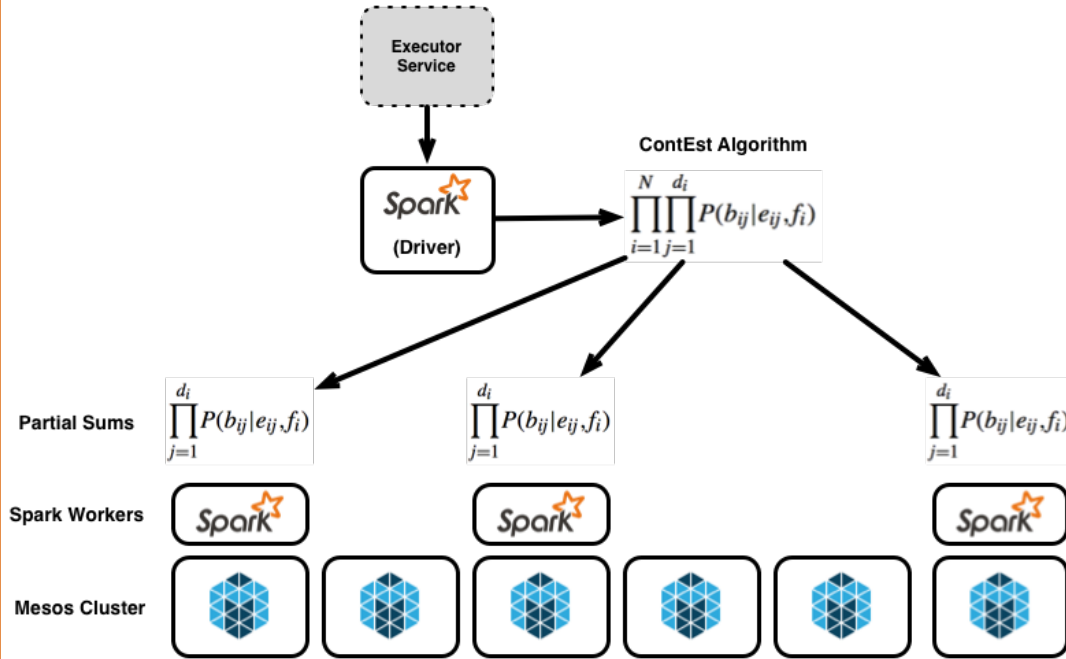
Location: **distributed**

Updates: **continuous**

Sharing: **???**

- How can we take advantage of the “natural” parallelizability in many of these computations?

Spark takes advantage of **shared parallelism** throughout a pipeline



- Many genomics analyses are naturally parallelizable
- Pipelines can often share parallelism between stages
- No intermediate files
- Separate implementation concerns:
 - parallelization and scaling **in the platform**
 - let methods developers **focus on methods**

Parquet+Avro lets us **compile** our file formats

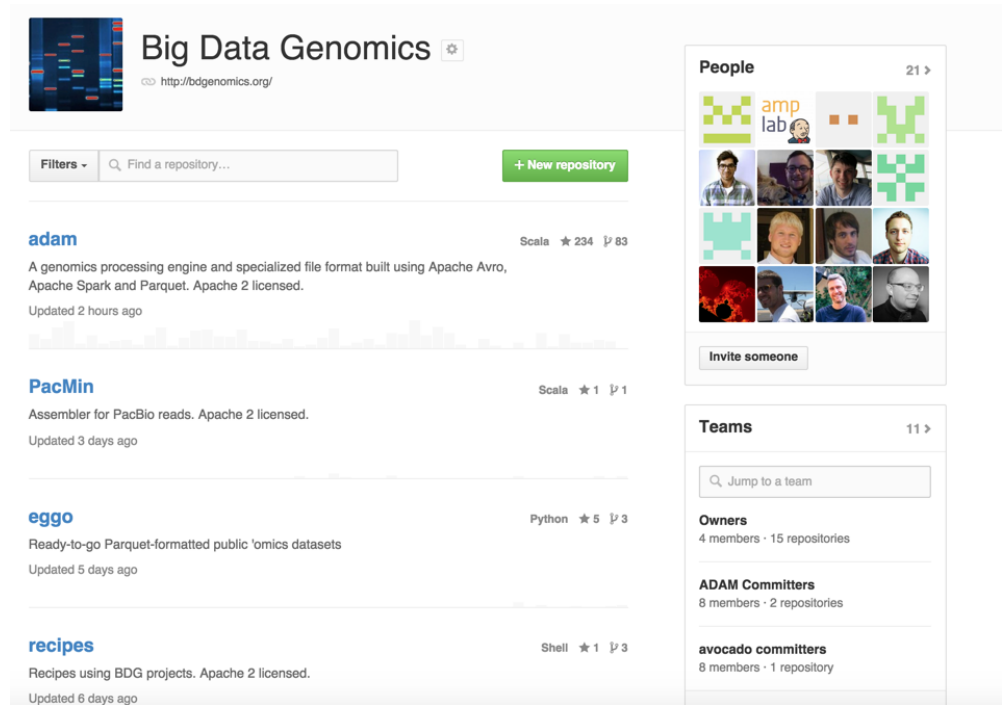
- Instead of defining **custom** file formats for each data type and access pattern...
- Parquet creates a compressed format for each Avro-defined data model.
- Improvement over existing formats¹
 - 20-22% for BAM
 - ~95% for VCF

```
record Variant {  
  union { null, Contig } contig = null;  
  union { null, long } start = null;  
  union { null, long } end = null;  
  
  union { null, string } referenceAllele = null;  
  union { null, string } alternateAllele = null;  
  union { null, StructuralVariant } svAllele = null;  
  
  union { boolean, null } isSomatic = false;  
}
```

¹compression % quoted from 1K Genomes samples

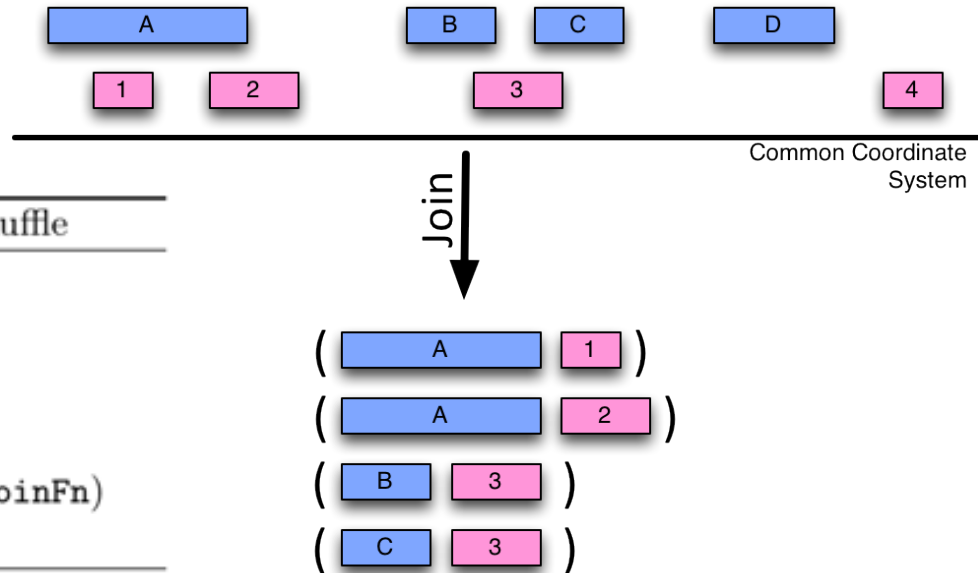
ADAM is “Spark for Genomics”

- Hosted at Berkeley and the AMPLab
- Apache 2 Licensed
- Contributors from universities, biotechs, and pharmas
- Today: core spatial primitives, variant calling
- Future: RNA-seq, cancer genomics tools



The screenshot displays the GitHub profile for 'Big Data Genomics' (bdgenomics.org). The main repository, 'adam', is highlighted, described as a genomics processing engine and specialized file format built using Apache Avro, Apache Spark, and Parquet. It is Apache 2 licensed and has 234 stars and 83 forks. Other repositories listed include 'PacMin' (Scala, 1 star, 1 fork), 'eggo' (Python, 5 stars, 3 forks), and 'recipes' (Shell, 1 star, 3 forks). On the right, the 'People' section shows 21 team members, including the 'AMP Lab' logo and several individual profiles. Below that, the 'Teams' section lists 'Owners' (4 members, 15 repositories), 'ADAM Committers' (8 members, 2 repositories), and 'avocado committers' (8 members, 1 repository).

ADAM includes core genomics primitives: distributed join-by-overlap



Algorithm 2 Partition And Join Regions via Shuffle

```
left ← input dataset; left side of join  
right ← input dataset; right side of join  
partitions ← left.getPartitions()  
left ← left.repartitionAndSort(partitions)  
right ← right.repartitionAndSort(partitions)  
joined ← left.zipPartitions(right, partitionJoinFn)  
return joined
```

"Yet Another Workflow Engine??"



Sébastien Boisvert @sebhtml · Feb 27

Bioinformatics has a lot of "workflow engines" - a new one is born every day
slideshare.net/TimothyDanford... #cloud #spark



Titus Brown

@ctitusbrown

Follow

@sebhtml you know what we need? A meta workflow engine!



Does Bioinformatics Need Another “Workflow Engine?”

- **No**: it has **a few** already, it will require **rewriting** all our software, we should focus on **methods** instead.
- **Yes**: we need to move to **commodity computing**, start planning for a day when **sharing is not copying**, write methods that **scale** with more resources
- Most importantly: separate “**developing a method**” from “**building a platform**,” and allow different developers to work separately on **both**

Thanks to...

Matt Massie
Frank Nothaft
Uri Laserson
Carl Yeksigian

And thank you!
Questions?

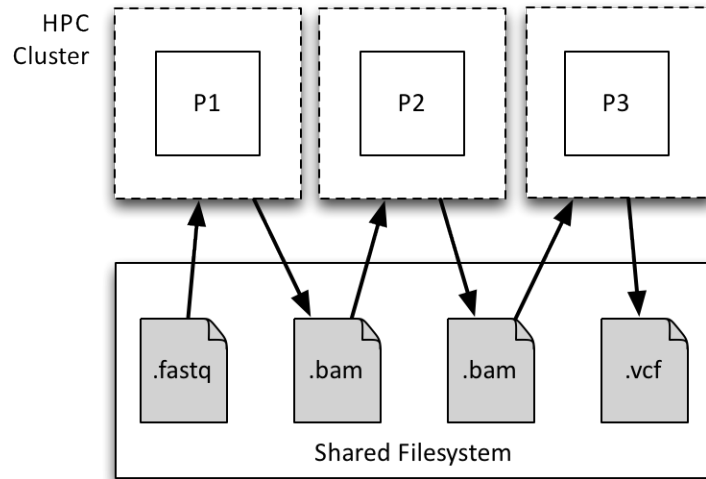
NCI Cloud Pilot Project



```
record Sample {  
  
    Participant participant;  
  
    string sampleId;  
    union { null, string } tcgaSampleBarcode = null;  
    string sampleType;  
    union { null, string } vial = null;  
    union { null, string } tcgaSubmissionDate = null;  
    string preservationMethod = null;  
  
}
```

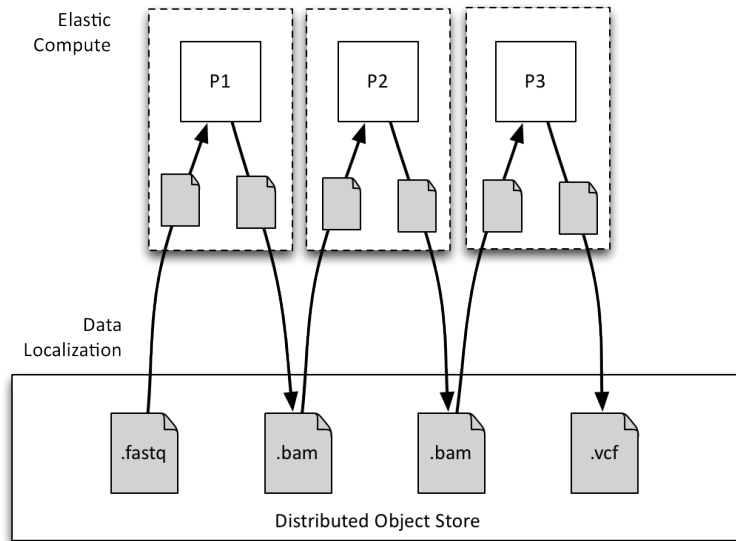
Bioinformatics today is carried out on a traditional cluster

- Hand-written task creation
- Dependence on file formats, which may be
 - poorly-defined,
 - contain optional fields, or
 - have multiple alternatives for a single data type
- End up rewriting workflow engines, task restart
- Poor support for multiple-sample analysis



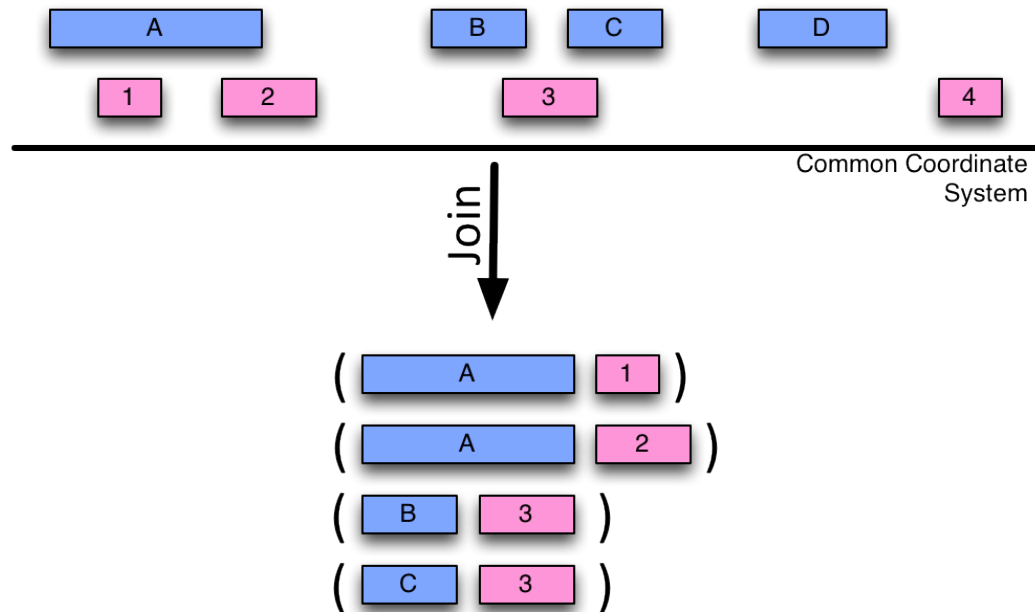
Translation to the cloud is constrained by existing tools

- Pretty much everyone lifts bioinformatics tools into the cloud in the same way
- Virtualization (VMs, Docker) for abstracting and encapsulating tasks
- Parallelization dictated by file format

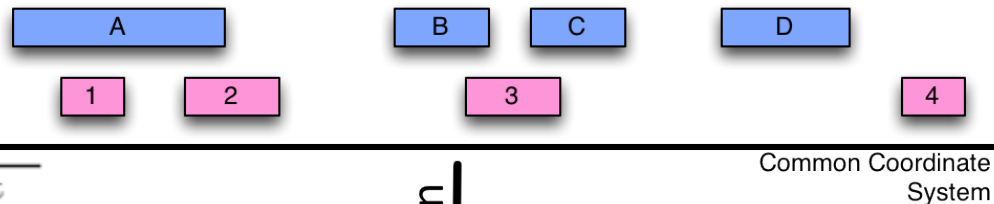


Spark can easily express genomics primitives: join by genomic overlap

1. Calculate disjoint regions based on left (blue) set
2. Partition both sets by disjoint regions
3. Merge-join within each partition
4. (Optional) aggregation across joined pairs

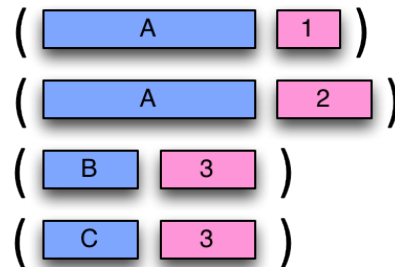


Spark can easily express genomics primitives: join by genomic overlap



Algorithm 1 Partition And Join Regions via Broadcast

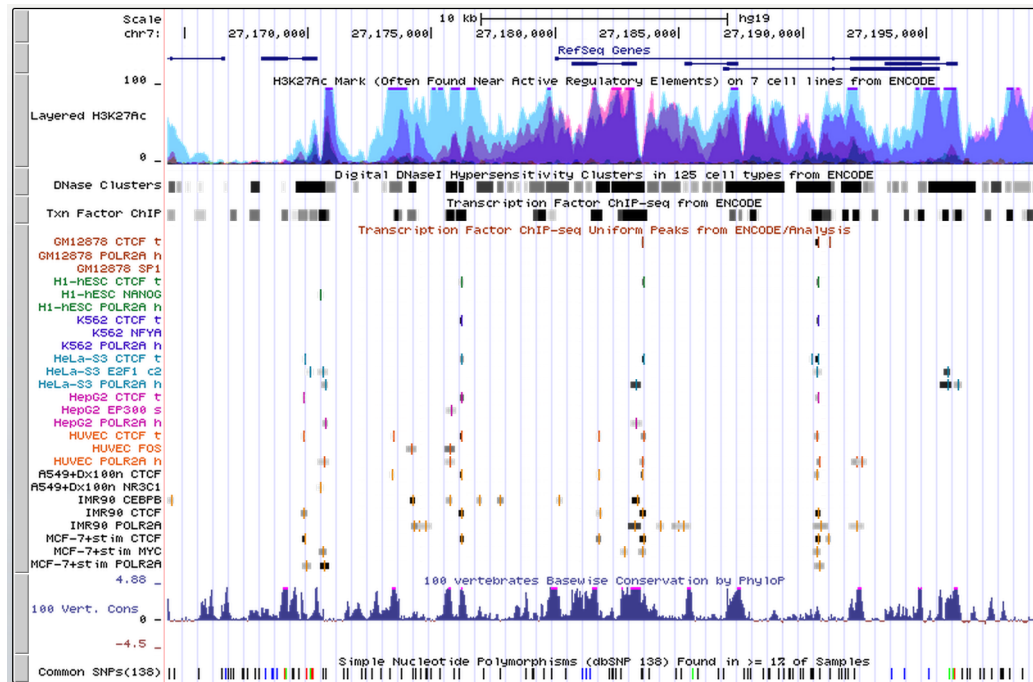
```
left ← input dataset; left side of join  
right ← input dataset; right side of join  
regions ← left.map(data ⇒ generateRegion(data))  
regions ← regions.groupBy(region ⇒ region.name)  
hulls ← regions.findConvexHull()  
hulls.broadcast()  
keyLeft ← left.keyBy(data ⇒ getHullId(data, hulls))  
keyRight ← right.keyBy(data ⇒ getHullId(data, hulls))  
joined ← keyLeft.join(keyRight)  
truePositives ← joined.filter(r1, r2 ⇒ r1.overlaps(r2))  
return truePositives
```



$$L(M_f^m) = P(\{b_i\} | \{e_i\}, r, m, f) = \prod_{i=1}^d P(b_i | e_i, r, m, f)$$

Spark RDDs and Partitioners allow declarative parallelization for genomics

- Genomics computation is parallelized in a small, standard number of ways
 - by position
 - by sample
- Declarative, flexible partitioning schemes are useful



- Avro to define data models
- Parquet for serialization format
- Still need to answer design questions
 - how wide are the schemas?
 - how much do we follow existing formats?
 - how do carry through projections?

```
@namespace("edu.berkeley.cs.amplab.adam.avro")
protocol ADAM {

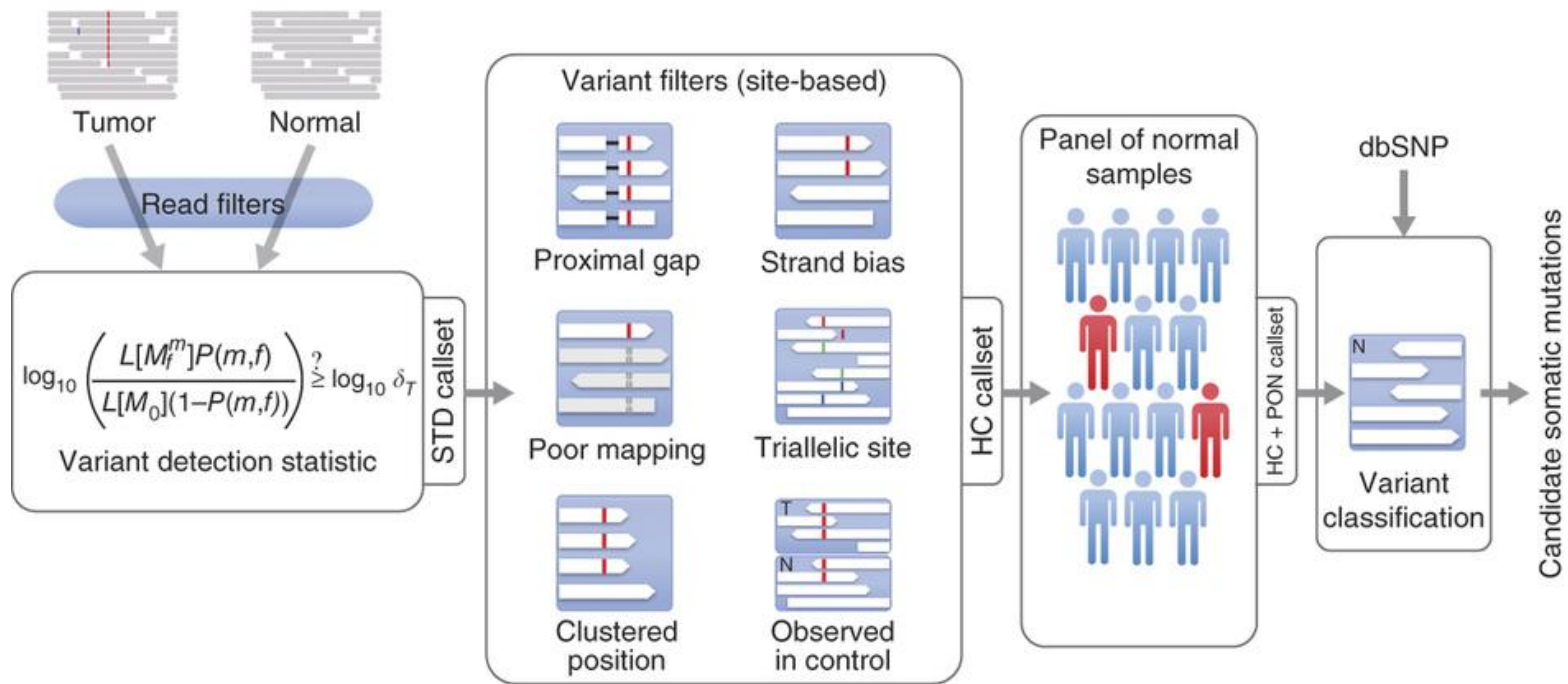
  record ADAMRecord {

    union { null, string } referenceName = null;
    union { null, int } referenceId = null;

    // 0-based reference position start
    union { null, long } start = null;

    union { null, int } mapq = null;
    union { null, string } readName = null;
    union { null, string } sequence = null;
    union { null, string } mateReference = null;
    union { null, long } mateAlignmentStart = null;
    union { null, string } cigar = null;
    union { null, string } qual = null;
    union { null, string } recordGroupName = null;
    union { null, int } recordGroupId = null;

    // Read flags (all default to false)
    union { boolean, null } readPaired = false;
    union { boolean, null } properPair = false;
    union { boolean, null } readMapped = false;
    union { boolean, null } mateMapped = false;
    union { boolean, null } readNegativeStrand = false;
    union { boolean, null } mateNegativeStrand = false;
    union { boolean, null } firstOfPair = false;
    union { boolean, null } secondOfPair = false;
    union { boolean, null } primaryAlignment = false;
    union { boolean, null } failedVendorQualityChecks = false;
    union { boolean, null } duplicateRead = false;
```



Cibulskis et al. *Nature Biotechnology* **31**, 213–219 (2013)

Slide title

Text

Slide Title

- Bullet
- Bullet
- Bullet