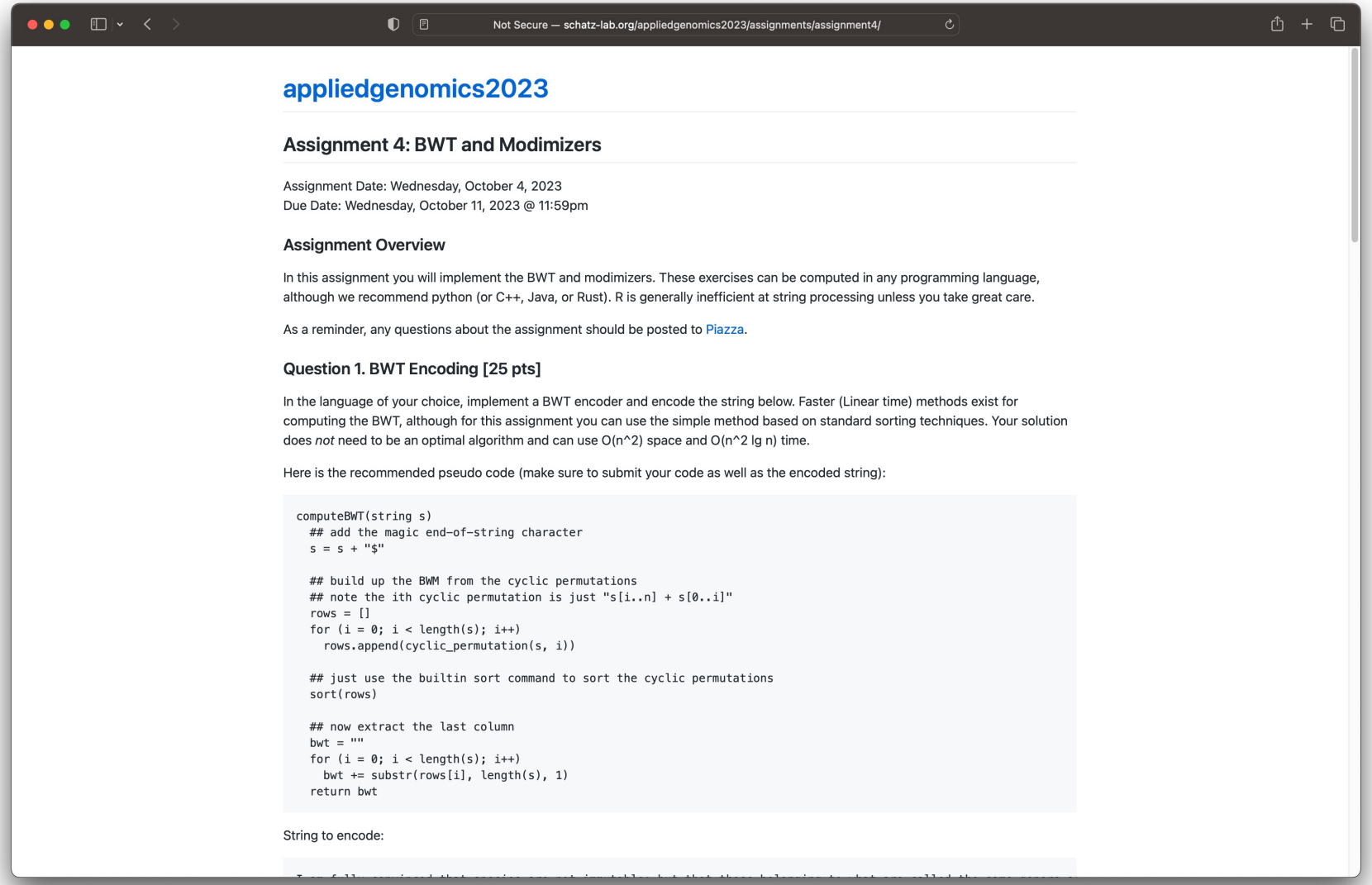
Intro to ML & Bedtools

Michael Schatz

October 11, 2023

Lecture 13. Applied Comparative Genomics

Assignment 4: BWT and Modimizers Due Wednesday Oct 11 by 11:59pm



appliedgenomics2023

Assignment 4: BWT and Modimizers

Assignment Date: Wednesday, October 4, 2023
Due Date: Wednesday, October 11, 2023 @ 11:59pm

Assignment Overview

In this assignment you will implement the BWT and modimizers. These exercises can be computed in any programming language, although we recommend python (or C++, Java, or Rust). R is generally inefficient at string processing unless you take great care.

As a reminder, any questions about the assignment should be posted to Piazza.

Question 1. BWT Encoding [25 pts]

In the language of your choice, implement a BWT encoder and encode the string below. Faster (Linear time) methods exist for computing the BWT, although for this assignment you can use the simple method based on standard sorting techniques. Your solution does *not* need to be an optimal algorithm and can use O(n^2) space and O(n^2 lg n) time.

Here is the recommended pseudo code (make sure to submit your code as well as the encoded string):

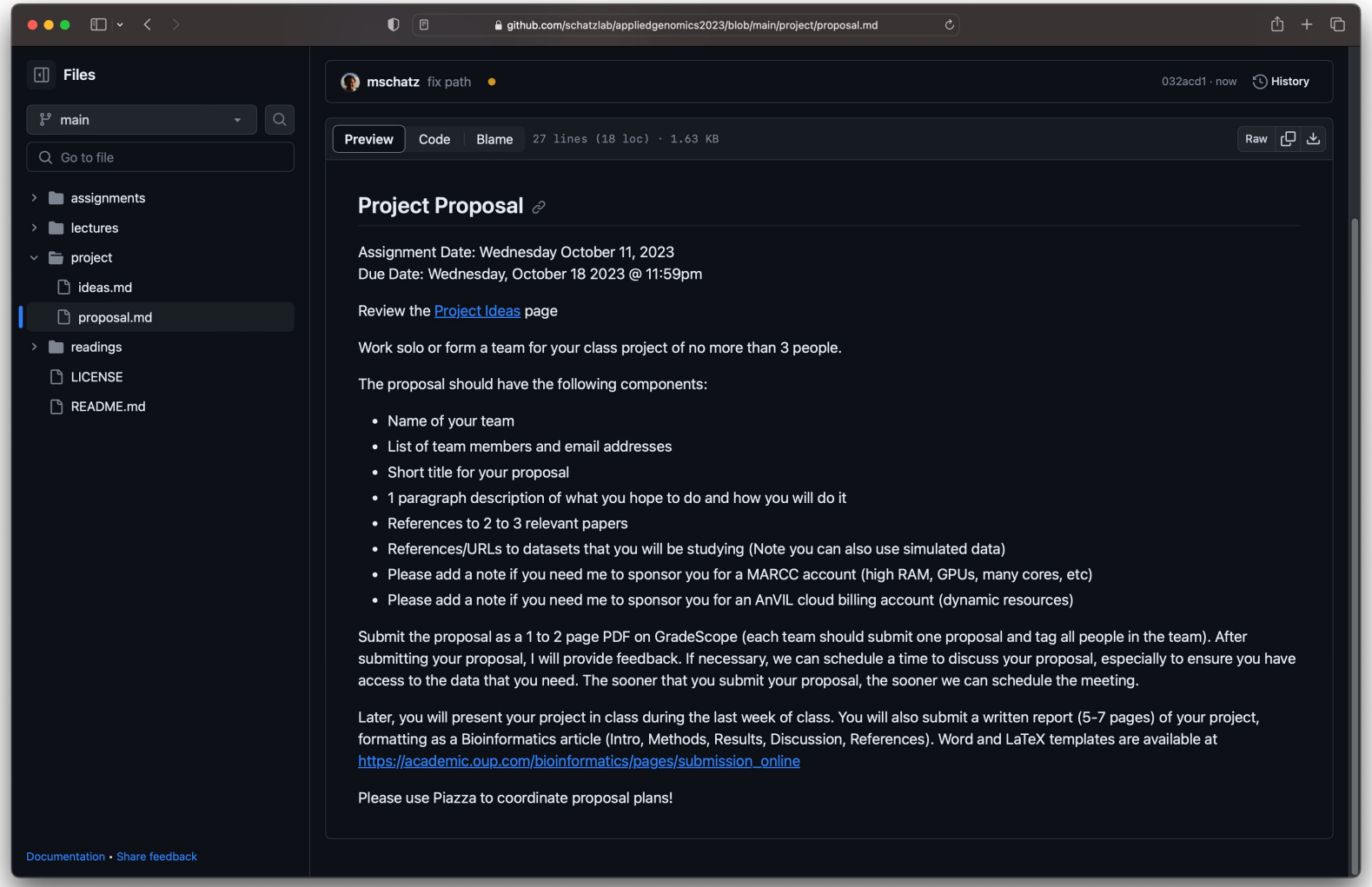
```
computeBWT(string s)
  ## add the magic end-of-string character
  s = s + "$"

  ## build up the BWM from the cyclic permutations
  ## note the ith cyclic permutation is just "s[i..n] + s[0..i]"
  rows = []
  for (i = 0; i < length(s); i++)
    rows.append(cyclic_permutation(s, i))

  ## just use the builtin sort command to sort the cyclic permutations
  sort(rows)

  ## now extract the last column
  bwt = ""
  for (i = 0; i < length(s); i++)
    bwt += substr(rows[i], length(s), 1)
  return bwt
```

String to encode:

https://github.com/schatzlab/appliedgenomics2023/tree/main/assignments/assignment4

Check Piazza for questions!

Project Proposal
Due Wednesday Oct 18 by 11:59pm



https://github.com/schatzlab/appliedgenomics2023/blob/main/project/proposal.md

Check Piazza for questions!

Algorithm Overview

1. Split read into segments

Read

CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA

Read (reverse complement)

TACAGGCCTGGGTAAAATAAGGCTGAGAGCTACTGG

Policy: extract 16 nt seed every 10 nt

Seeds

+, 0: CCAGTAGCTCTCAGCC

+, 10: TCAGCCTTATTTTACC

+, 20: TTTACCCAGGCCTGTA

-, 0: TACAGGCCTGGGTAAA

-, 10: GGTAAAATAAGGCTGA

-, 20: GGCTGAGAGCTACTGG

2. Lookup each segment and prioritize

Seeds

+, 0: CCAGTAGCTCTCAGCC

+, 10: TCAGCCTTATTTTACC

+, 20: TTTACCCAGGCCTGTA

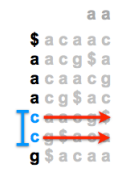
-, 0: TACAGGCCTGGGTAAA

-, 10: GGTAAAATAAGGCTGA

-, 20: GGCTGAGAGCTACTGG

→

Ungapped alignment with FM Index



→

Seed alignments (as B ranges)

{ [211, 212], [212, 214] }

{ [653, 654], [651, 653] }

{ [684, 685] }

{ }

{ }

{ [624, 625] }

3. Evaluate end-to-end match

Extension candidates

SA:684, chr12:1955

SA:624, chr2:462

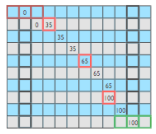
SA:211: chr4:762

SA:213: chr12:1935

SA:652: chr12:1945

→

SIMD dynamic programming aligner



→

SAM alignments

```
r1   0    chr12    1936     0
     36M  *    0    0
     CCAGTAGCTCTCAGCCTTATTTTACCCAGGCCTGTA
     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
     AS:i:0    XS:i:-2   XN:i:0
     XM:i:0    XO:i:0    XG:i:0
     NM:i:0    MD:Z:36   YT:Z:UU
     YM:i:0
...
```

(Langmead & Salzberg, 2012)

Similarity metrics

- Hamming distance
  - Count the number of substitutions to transform one string into another

```
MIKESCHATZ
||X||XXXX|
MICESHATZZ
     5
```

- Edit distance
  - The minimum number of substitutions, insertions, or deletions to transform one string into another

```
MIKESCHAT-Z
||X||X|||X|
MICES-HATZZ
     3
```

Dynamic Programming Matrix

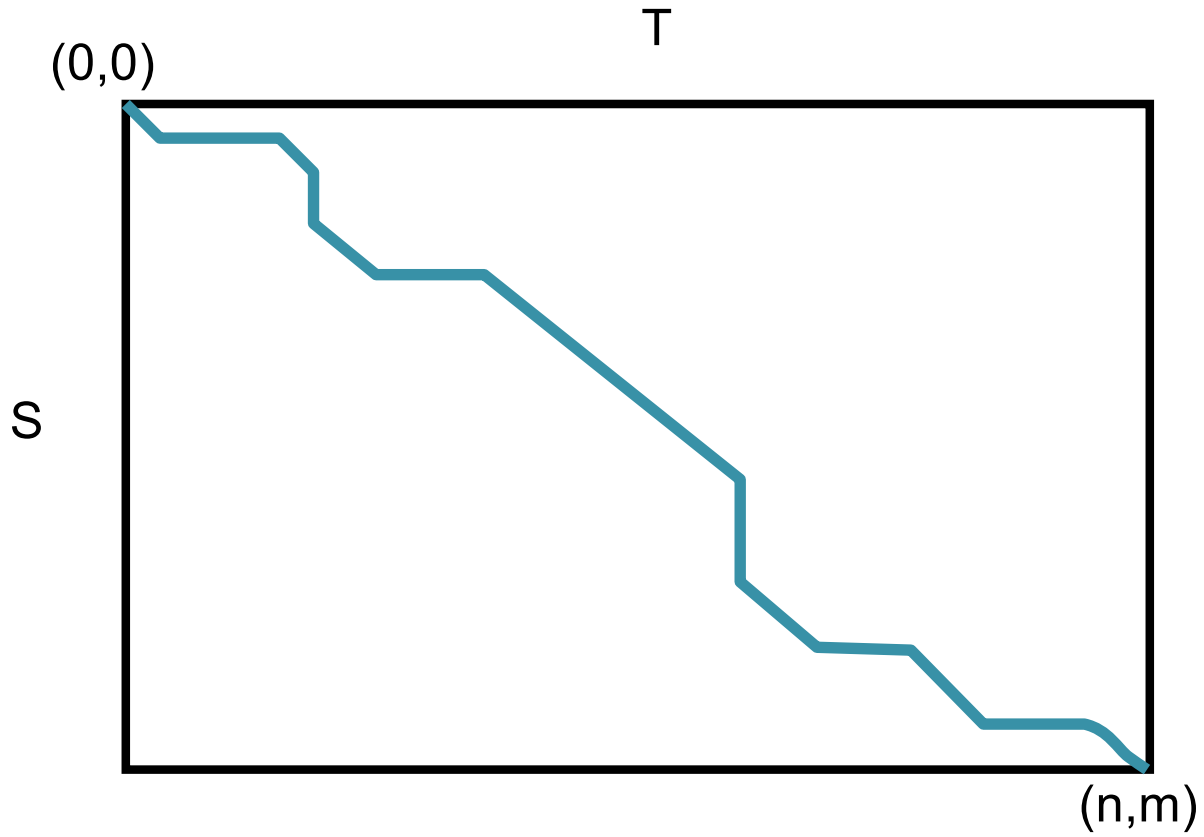
		A	C	A	C	A	C	T	A
	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
G	2	1	1	2	3	4	5	6	7
C	3	2	1	2	2	3	4	5	6
A	4	3	2	1	2	2	3	4	5
C	5	4	3	2	1	2	2	3	4
A	6	5	4	3	2	1	2	3	3
C	7	6	5	4	3	2	1	2	3
A	8	7	6	5	4	3	2	2	2

D[AGCACACA,ACACACTA] = 2

```
AGCACAC-A
|*|||||*|
A-CACACTA
```

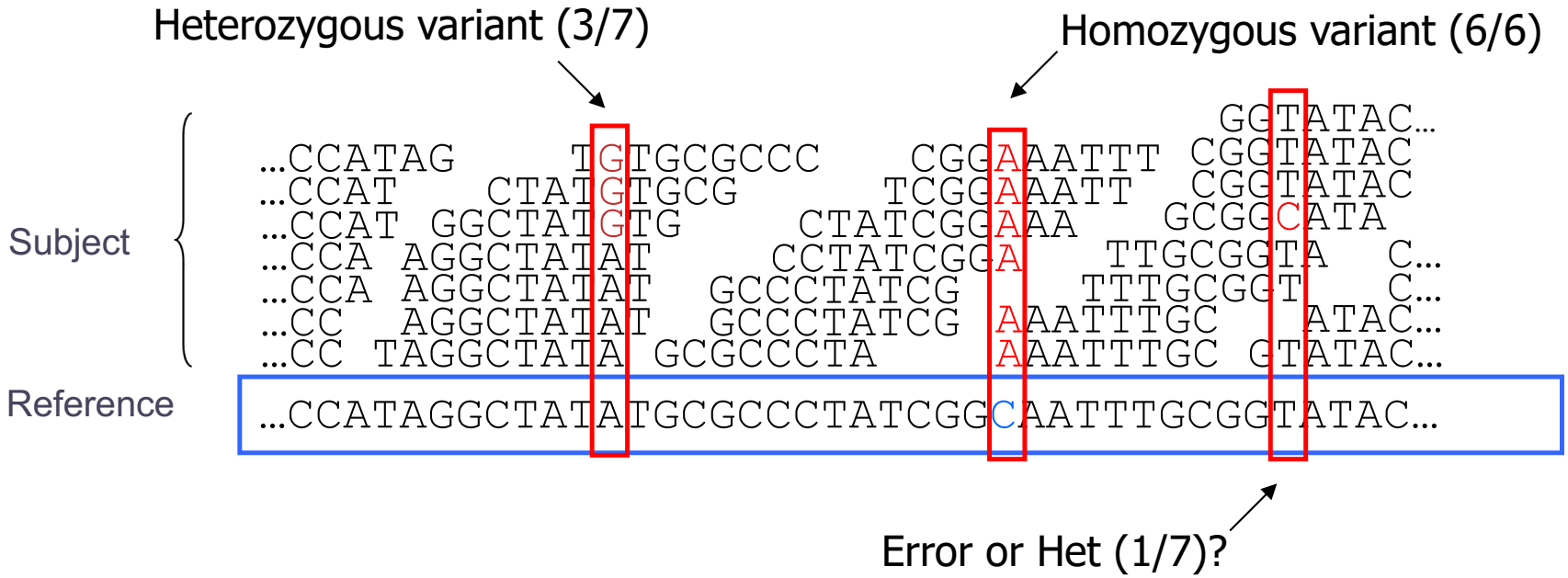
[Can we do it any better?]

Global Alignment Schematic



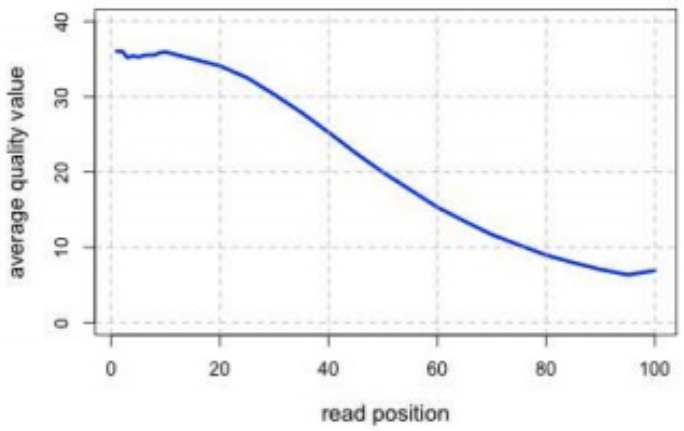
- A high quality alignment will stay close to the diagonal
  - If we are only interested in high quality alignments, we can skip filling in cells that can't possibly lead to a high quality alignment
  - Find the global alignment with at most edit distance d: O(2dn)

Genotyping Theory



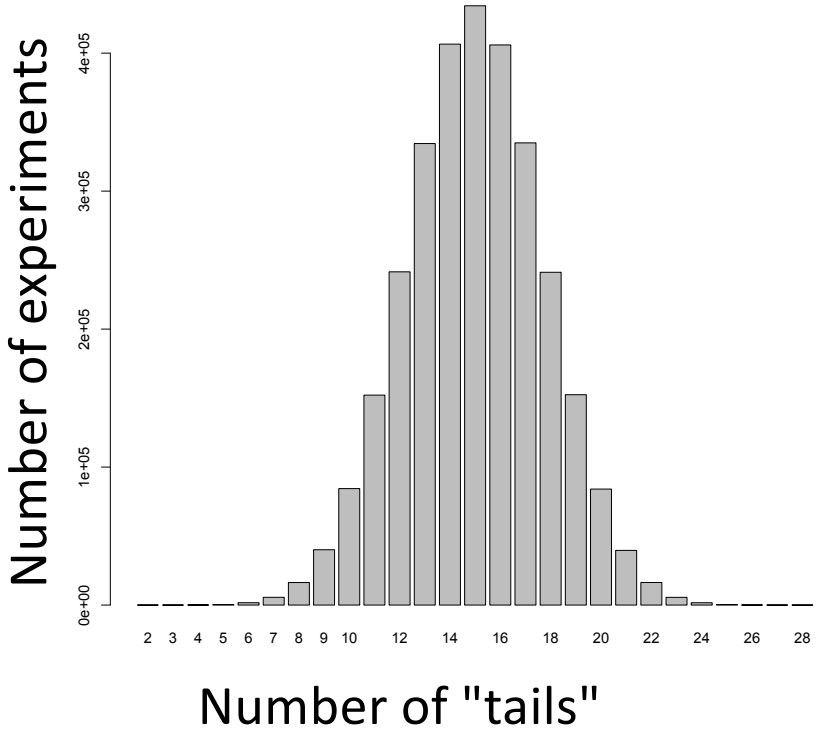
- If there were no sequencing errors, identifying SNPs would be very easy: any time a read disagrees with the reference, it must be a variant!

- Sequencing instruments make mistakes
  - Quality of read decreases over the read length



- A single read differing from the reference is probably just an error, but it becomes more likely to be real as we see it multiple times

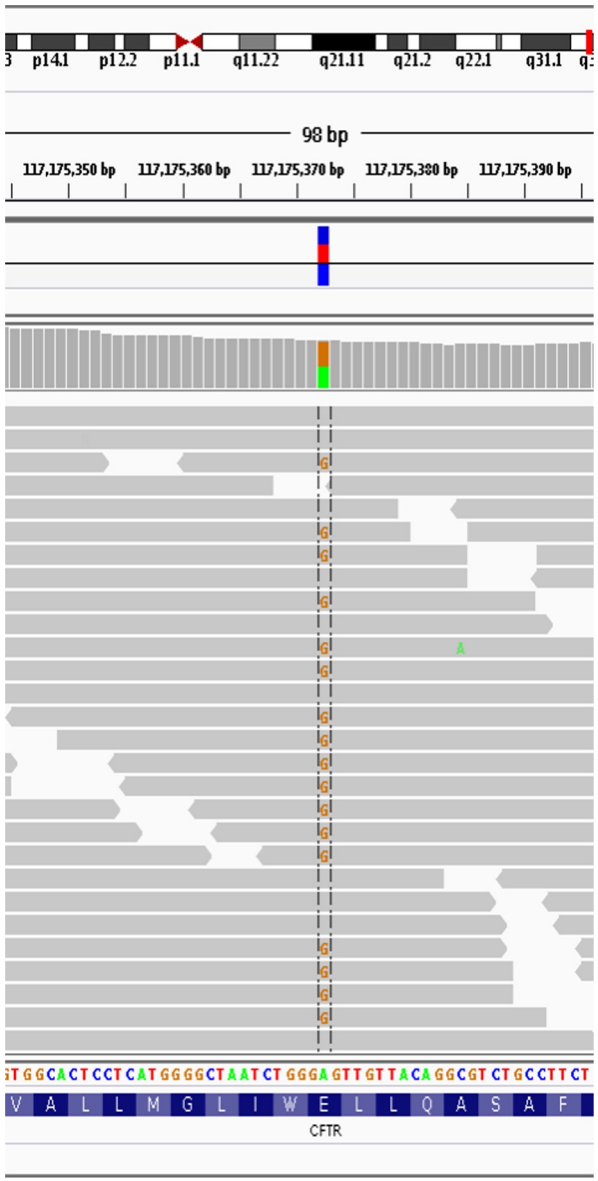
So, with 30 tosses (reads), we are much more likely to see an even mix of alternate and reference alleles at a heterozygous locus in a genome



This is why at least a "30X" (30 fold sequence coverage) genome is recommended: it confers sufficient power to distinguish heterozygous alleles and from mere sequencing errors

P(3/30 het) <?> P(3/30 err)

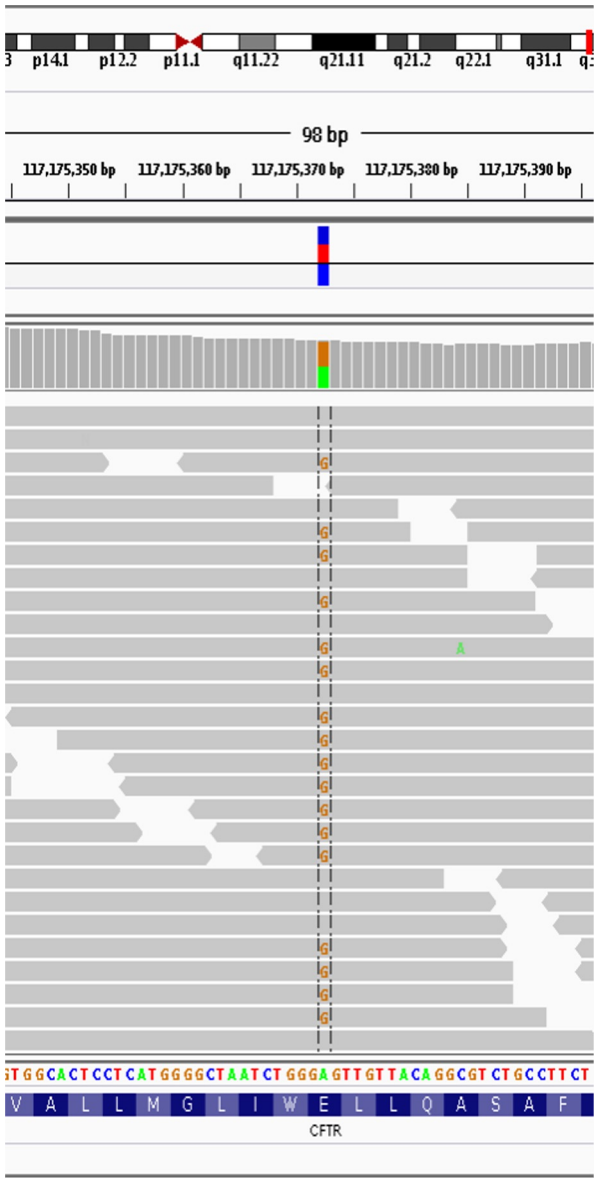
What information is needed to decide if a variant exists?



- Depth of coverage at the locus
- Bases observed at the locus
- The base qualities of each allele
- The strand composition
- Mapping qualities
- Proper pairs?
- Expected polymorphism rate

https://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-10-r99

Bayesian SNP calling



$$P(SNP|Data) = \frac{P(Data|SNP) * P(SNP)}{P(Data)}$$

http://www.nature.com/ng/journal/v23/n4/full/ng1299_452.html

PolyBayes: The first statistically rigorous variant detection tool.

letter

A general approach to single-nucleotide polymorphism discovery

Gabor T. Marth[1], Ian Korf[1], Mark D. Yandell[1], Raymond T. Yeh[1], Zhijie Gu[2], Hamideh Zakeri[2], Nathan O. Stitziel[1], LaDeana Hillier[1], Pui-Yan Kwok[2] & Warren R. Gish[1]

Bayesian posterior probability

Base call + Base quality

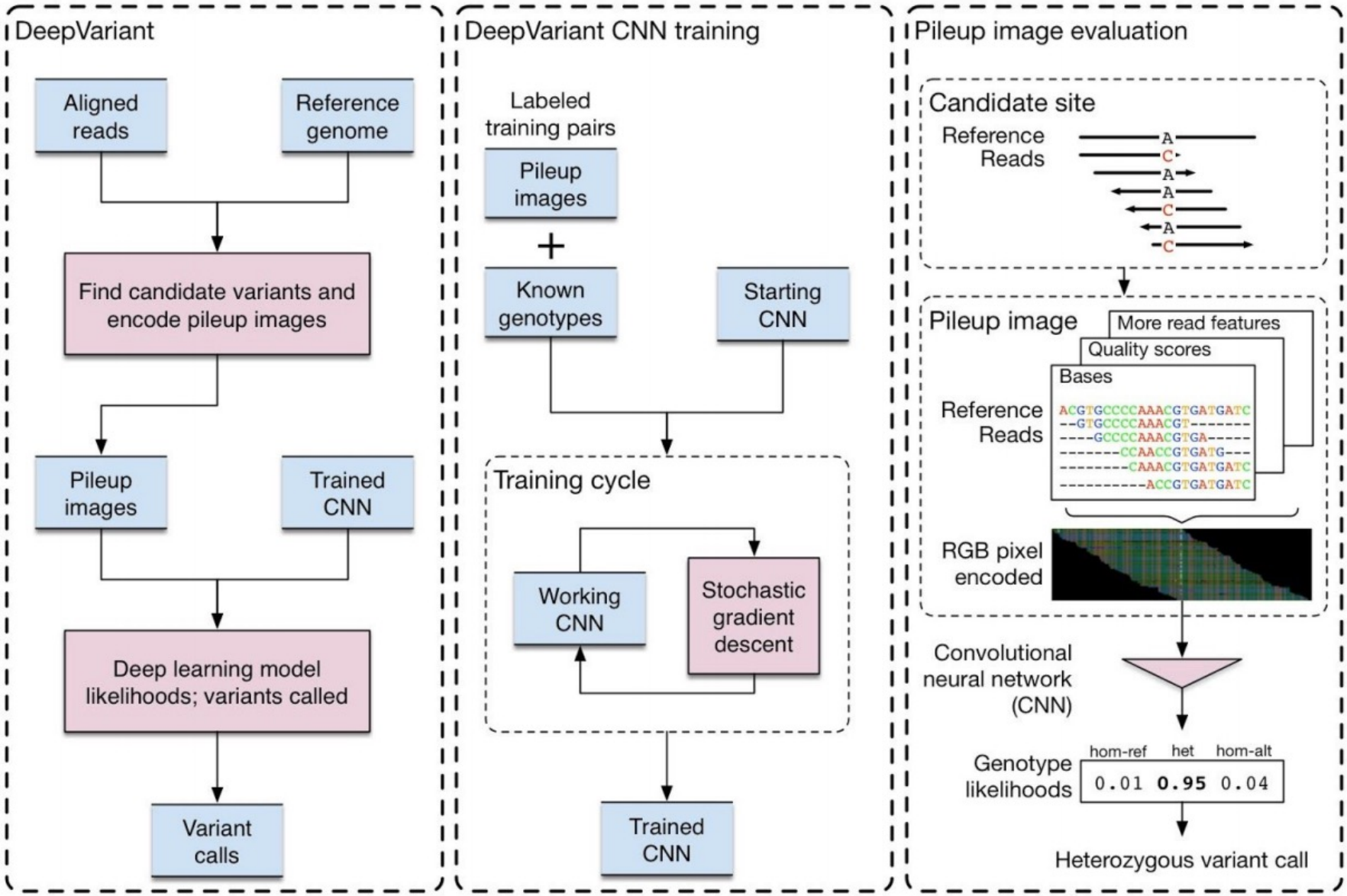
Expected (prior) polymorphism rate

$$P(SNP) = \sum_{all\ variable\ S} \frac{\frac{P(S_1 \mid R_1)}{P_{Prior}(S_1)} \cdot \ldots \cdot \frac{P(S_N \mid R_N)}{P_{Prior}(S_N)} \cdot P_{Prior}(S_1,\ldots,S_N)}{\sum_{S_{i_1} \in [A,C,G,T]} \ldots \sum_{S_{i_N} \in [A,C,G,T]} \frac{P(S_{i_1} \mid R_1)}{P_{Prior}(S_{i_1})} \cdot \ldots \cdot \frac{P(S_{i_N} \mid R_1)}{P_{Prior}(S_{i_N})} \cdot P_{Prior}(S_{i_1},\ldots,S_{i_N})}$$

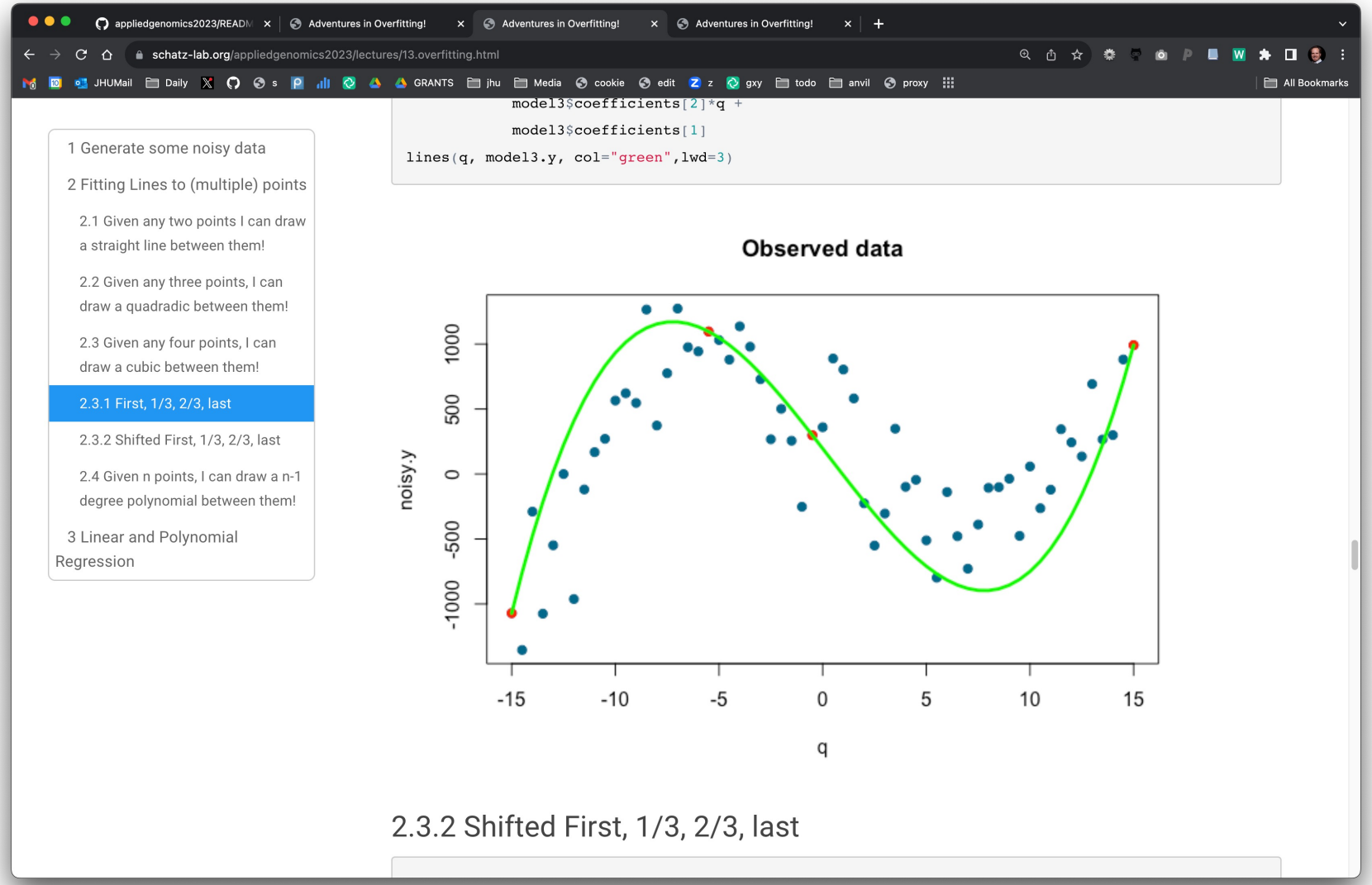
Probability of observed base composition (should model sequencing error rate)

http://www.nature.com/ng/journal/v23/n4/full/ng1299_452.html

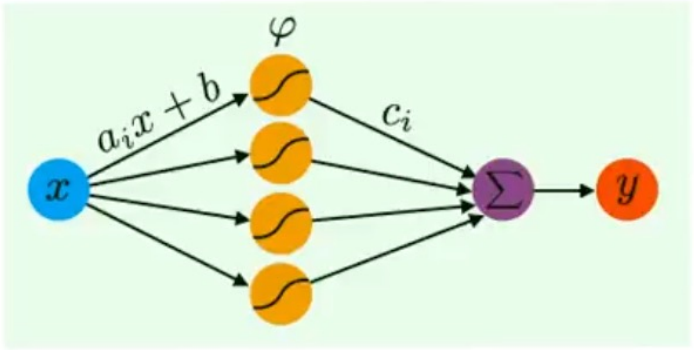
DeepVariant



Creating a universal SNP and small indel variant caller with deep neural networks
Poplin et al. (2018) Nature Biotechnology. https://www.nature.com/articles/nbt.4235



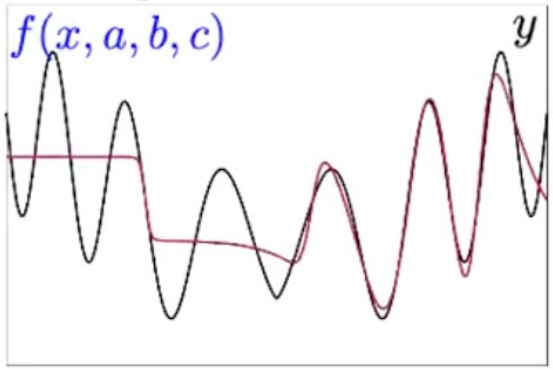
https://schatz-lab.org/appliedgenomics2023/lectures/13.overfitting.html



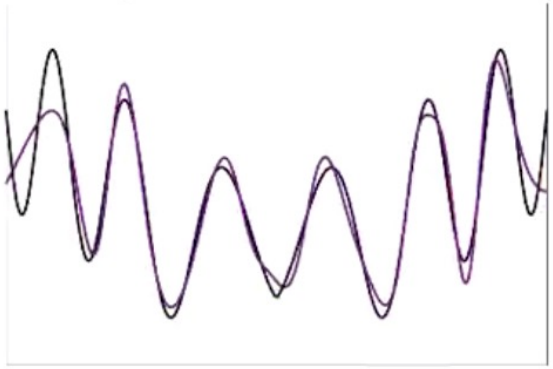
1 hidden layer perceptron:

$$y \approx f(x, a, b, c) \stackrel{\text{def.}}{=} \sum_{i=1}^{p} c_i \varphi(a_i x + b_i)$$

$p = 6$ neurons



$p = 20$ neurons



Approximation by Superpositions of a Sigmoidal Function*

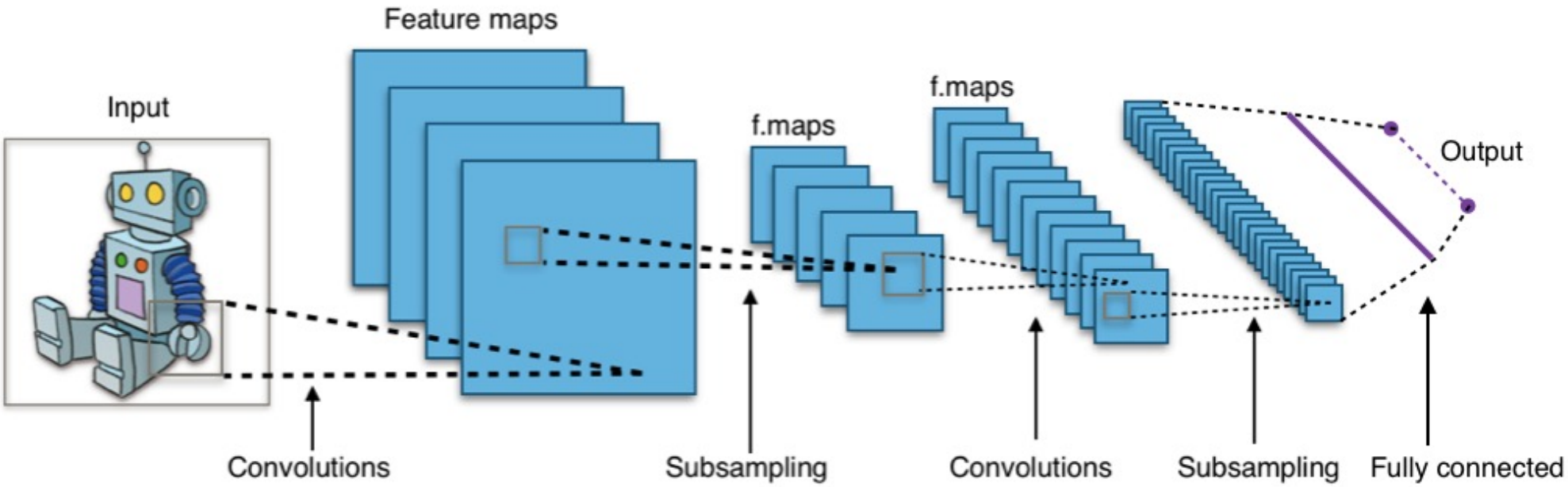
G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

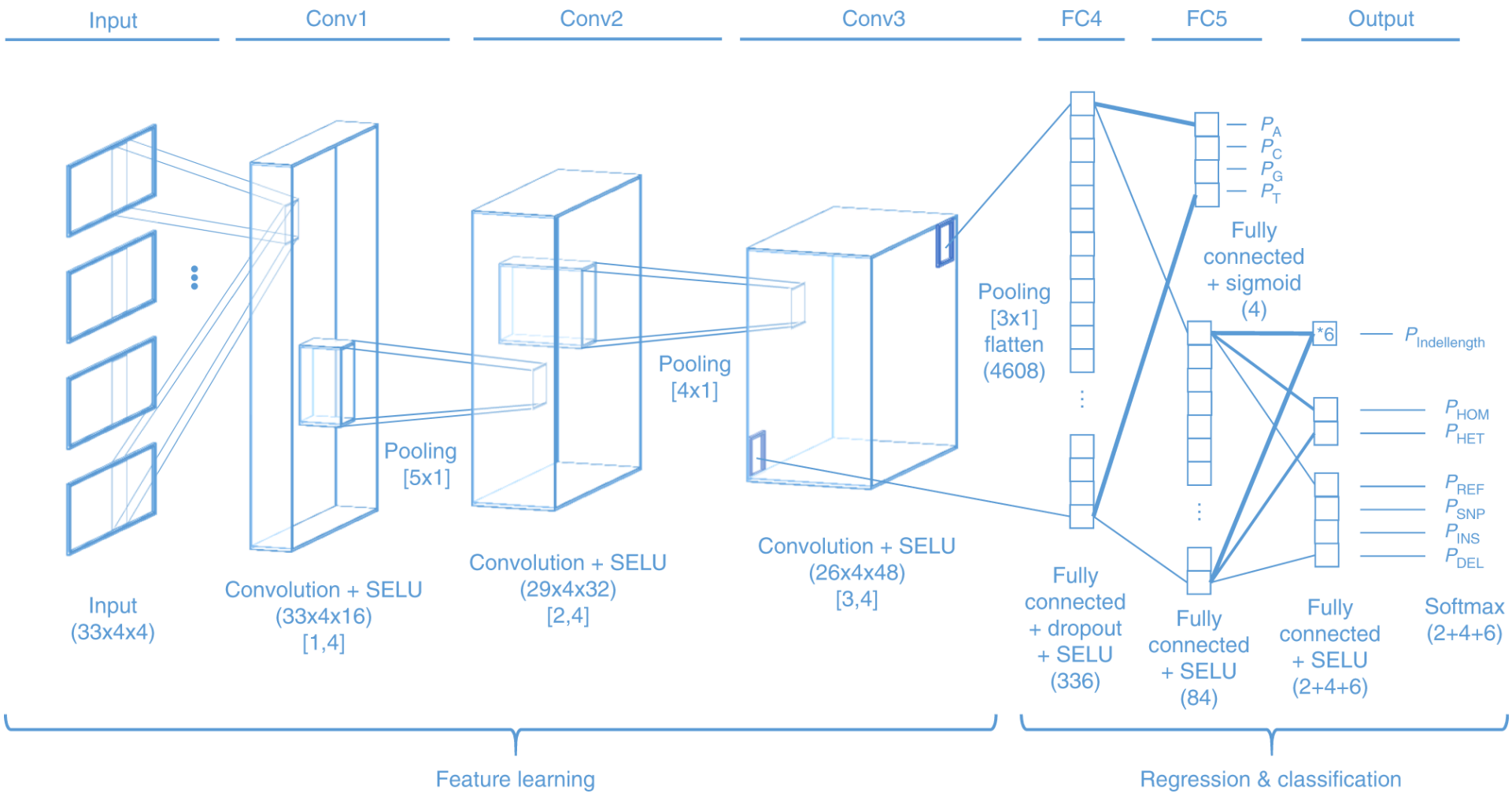
Key words. Neural networks, Approximation, Completeness.

Approximation by superpositions of a sigmoidal function

Cybenko, G. (1989) Mathematics of Control Signal Systems doi: 10.1007/BF02551274



https://training.galaxyproject.org/training-material/topics/statistics/tutorials/CNN/tutorial.html



A multi-task convolutional deep neural network for variant calling in single molecule sequencing
Luo et al. (2019) Nature Communication. https://doi.org/10.1038/s41467-019-09025-z

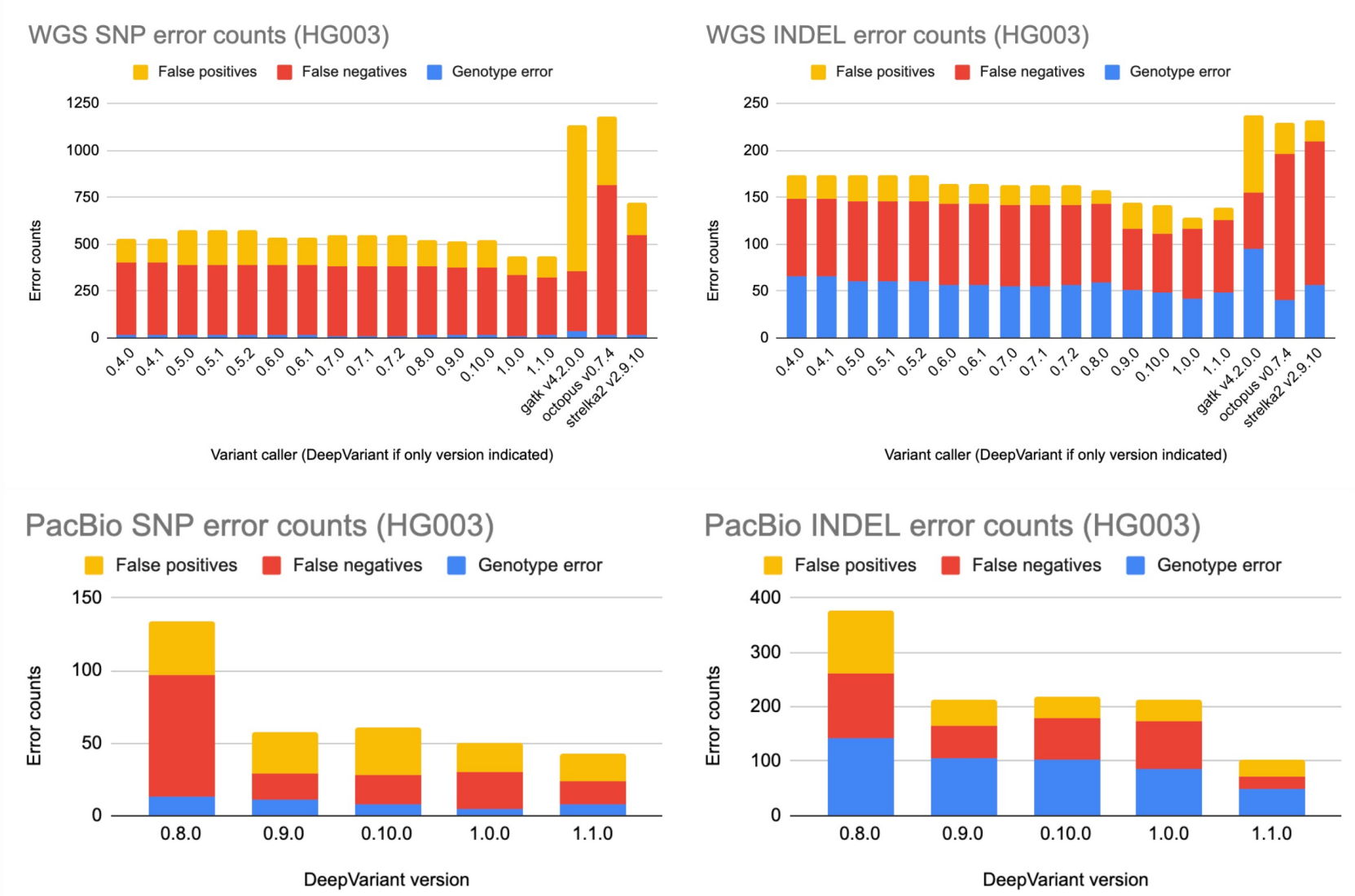


Figure 2: Error counts over the years for HG003. For Illumina WGS, we use a HiSeqX PCR-free dataset at 30x coverage. For PacBio HiFi, we use the same BAM as the one in our case study.

DeepVariant over the years

https://google.github.io/deepvariant/posts/2021-06-08-deepvariant-over-the-years/