Basic Info

Title

Longread Sequencing - Evaluating Integrity of Consensus Sequence Generation from Reads Binned by Unique Molecular Identifiers (UMIs)

Team Members

- Caleb Cranney
    - Email: caleb.cranney@utah.edu
    - uID: u0704188
- Alexander Millar
    - Email: alexander.millar@utah.edu
    - uID: u0740821

Project Repository

https://github.com/illato/conseq

Background and Motivation

Both teammates are in the Biomedical Informatics (BMI) master's program at the University of Utah. In considering possible projects, it was agreed that we wanted to focus on a project with bioinformatics applications. Our proposed project focuses on long-read nanopore sequencing, a relatively new topic pertinent to the future of genetic sequencing. In addition to being a generally exciting topic with numerous visualization possibilities, we both felt this would be beneficial to our future careers.

Teammate Caleb Cranney currently works in the Biochemistry lab of Justin English (see here). Among the various current projects of the lab is VEGAS, a protocol for synthetically designing biomolecules via directed evolution. In essence, viruses with a specific protein are placed under evolutionary pressure to improve the efficacy of that protein. By the end of the protocol, we have the genetic blueprint for a "better" protein. We are interested in studying the changes that occurred in this process to improve the protein. In order to do so, the English Lab is developing a protocol for extracting and sequencing viruses throughout this process, allowing lab members to compare genetic changes over time from the original protein state to the final protein state. The end goal is to effectively create a phylogenetic tree and track major, widespread changes in the gene of interest. However, the sequencing process has several major obstacles, which Caleb was originally hired to help mitigate and that will be the focus of this project.

Nanopore sequencing is the ideal sequencing method for VEGAS. In addition to being less costly and more convenient, it can sequence a single entire virion genome in one pass. This is an ideal situation where we anticipate extracting viruses in various states of genetic modification. Illumina short-read sequencing would require stitching together portions of these genomes after the fact to find an original consensus, and as such the subtle differences we are trying to identify would be lost. Nanopore sequencing, while ideal, is also notoriously inaccurate. Several protocols exist to mitigate this problem. Generally, these methods involve duplicating the reads before sequencing them, then comparing the duplicates computationally to generate the "original," accurate consensus sequence. Most methodologies rely on a reference genome or sequence to compile the consensus. However, relying on the reference genome risks overlooking small changes that may have been introduced since the virus diverged, and is therefore not viable for the VEGAS strategy. Thus, the English lab has been developing a protocol and software for *de novo* consensus sequence generation (see this paper for the protocol their current approach is based on). This includes marking individual virion genomes with Unique Molecular Identifier (UMI) tags, in addition to duplicating and sequencing them. Caleb has written software for binning sequences by UMI tag in his work, as well as a candidate consensus sequencing algorithm, and is now in the process of evaluating its applicability to VEGAS. Alexander and Caleb both believe the application of interactive data visualization will be substantially beneficial in the evaluation, refinement, and furthering of this work.

Our proposed project focuses on evaluating the validity of consensus sequencing algorithms. See Project Objectives for more details.

Project Objectives

As stated previously, nanopore sequencing is notoriously inaccurate. This is augmented by the reality that these errors are often systematic – specific sequences are more likely than others to have specific sequencing errors. Thus when a consensus sequence is generated it would be hugely beneficial to look at common discrepancies between 1) each of the “binned” sequences that were used to generate the final consensus and 2) the final consensus itself. For example, for a consensus sequence that was generated by comparing 100 binned sequences, what is the most common discrepancy between these sequences and the final result? If 30 of those sequences had a C->A mutation at base pair 67, then we may want to look at that location more carefully in future experiments that rely on the same genome.

In general, we are hoping to create a visual that explores differences between the binned sequences and their generated consensus sequence. Some questions that could be answered are:

- What are the most common differences between the binned sequences and the consensus?
- Are specific error types more likely than others (insertion, deletion, mutation)?
- Are there specific base pairs that are more prone to errors?
- Are there entire areas of the consensus sequence that have a higher density of errors than others?
- Are specific error types concentrated in these base pairs/areas?

All of the above questions focus on the generation of a consensus from a single collection of binned sequences. However, in a real experiment, we would generate multiple consensus sequences, one for each of numerous bins of sequences. In a VEGAS experiment, these consensus sequences would be descended from the same original viral genome. So, in addition to being able to view individual bins, there would be benefit to seeing how these errors exist across multiple bins. Some questions that could be answered are:

- Are there any consensus sequences that have a unique concentration of errors in a given area?
- Are there any areas where the consensus sequences tend to have similar error occurrence patterns?
- Are there any areas where the consensus sequences tend to have dissimilar error occurrence patterns?

Data

Data for this project is generated by members of the English Lab and is located on their Lab computer. The consensus sequence algorithm element of the project is still in development, so we are not performing directed evolution experiments as of yet. Test data, however, is still generated from virion genomes as would apply in a VEGAS directed evolution experiment. These genomes each have a unique barcode inserted at a specific locus. This was largely done to verify the binning process succeeded, however, given that this locus is expected to vary substantially from consensus to consensus sequence it also benefits any attempts to visualize areas of frequent disagreement.

Data Processing

We do not expect substantial data cleanup. The consensus sequence algorithm Caleb developed relies on tracking differences between the consensus sequence and an alignment of the binned sequences. While running this program, it would be relatively trivial to write these differences out to a CSV-delimited file.

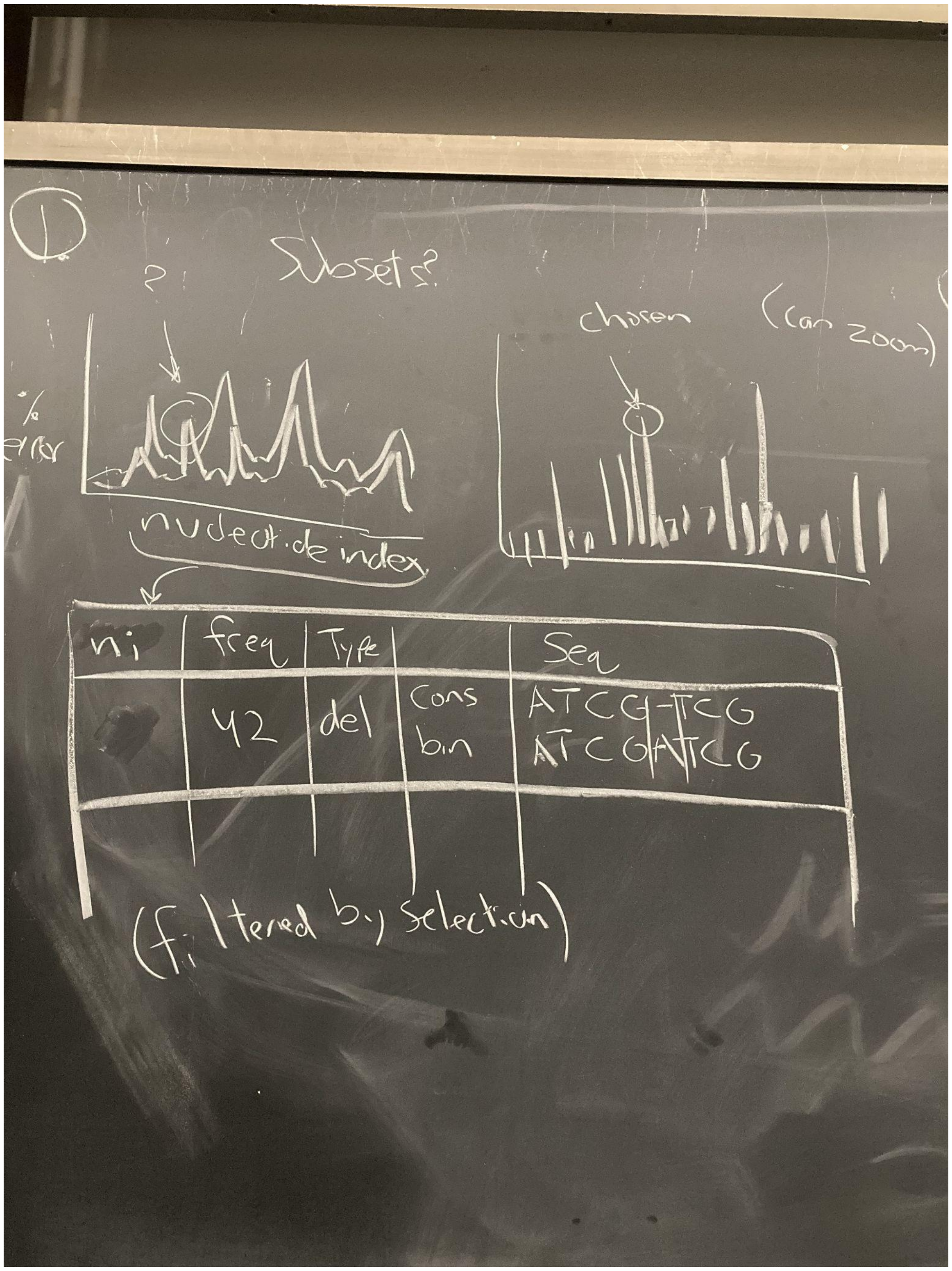
The current plan for the data format is to use a table with the following columns:

- ‘start’: The index of the consensus sequence where the aligned binned sequence begins to diverge.
- ‘end’: The index of the consensus sequence where the aligned binned sequence ends diverging.
- ‘insert’: The portion of the binned sequence that, when inserted between the ‘start’ and ‘end’ indices, would make the consensus sequence and binned sequence identical at this locus.
- ‘binNum’: An identifier for the specific consensus sequence. Lower bin numbers indicate the consensus sequence was generated from a larger bin of sequences.
- ‘seqID’: An identifier for the specific bin sequence. This is in case we come up with a useful idea for visualizing individual bin sequences against their consensus sequence, though we may get rid of this column if we decide it is a waste of memory. See Optional Features.

While not expecting too much cleanup, we will be considering ways to compress this data for use in the visualization. There are many individual differences between each binned sequence and their consensus, and having a CSV file to the order of millions of rows would not be out of scope for some projects. One example for how we may do that is to group all identical errors (same ‘start’, ’end’, and ‘insert’ values) and add a column that lists bin numbers where that error was identified, substantially shortening the table. One potential problem with this approach is the computational expense of getting all the errors for a particular bin. As we implement features that view the data from different angles, we will continue to evaluate the implications on size/performance balance. At scale, a database (e.g., SQLite) may be a more practical solution, but this is outside the scope of this project – the primary focus being visualization.

Visualization Design

Prototype Visual 1:



The first prototype was designed primarily with function in mind - searching for and obtaining specific data in a format scientists are familiar with. There are three components to this prototype visual, a line graph (top left), bar chart (top right), and a table (bottom).

a) The line graph is a high-level representation of all consensus sequences and the frequency of errors in their subsequent binned sequences. By showing all error frequencies together and normalizing their portrayal by using percentages, one could identify consensus sequences that had areas with an unusual concentration of errors, or sections where consensus sequences frequently did/did not have errors.
    i) There would be a line for each consensus.
        1) Interactively, one could select a line, which would then populate the bar plot and table with all data pertaining to that consensus and its binned sequences.
    ii) The x-axis would be the sequence index where an error could be found (index of a specific nucleotide).
    iii) The y-axis is the percentage of binned sequences with an error at the given index (number of sequences with a discrepancy at that index / the total number of sequences). This would likely need to be a percentage because the specific number of errors would be strongly impacted by the number of sequences in that bin, requiring a normalizing approach like percentage.

b) The bar graph drills down into a specific consensus and the discrepancies between it and the bin of sequences used to create it.
    i) Each bar represents errors at a specific index of the consensus.
        1) Interactively, selecting a bar would filter the table to only show errors found at that index. Selecting the bar would cause it to change color to show it was selected. You could probably select more than one bar, and the table would show them together.
        2) Optional: We could color these bars based on the specific nucleotide at that index, one each for adenosine, thymine, cytosine, and guanine (ATCG).
    ii) The x-axis would be the sequence index where an error could be found (index of a specific nucleotide).
    iii) The y-axis is the count of errors at the given index. This could alternatively be shown as a percentage as in a)iii), or a second y-axis could be added to the right side of the histogram.
    iv) Optional: Filtering errors by error type (insertion, mutation, or deletion), shortening the bars and possibly changing their color.
    v) Optional: In line with iv), we could possibly do a stacked bar chart with three possible colors for each error type.

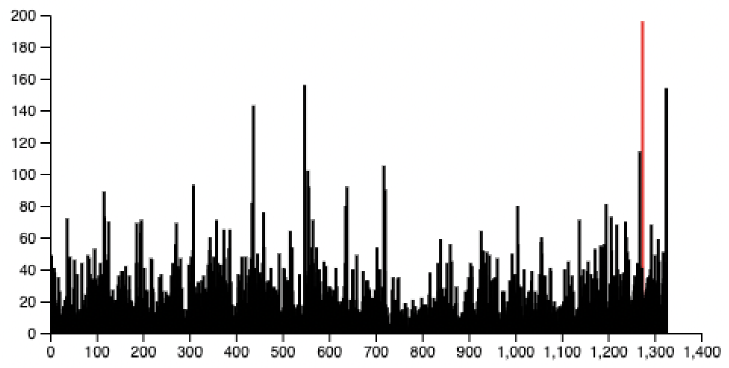
c) The table
    i) Each row would refer to a specific error at a specific index of the consensus sequence where there is a discrepancy with one or more sequences in the corresponding bin. There would likely need to be five columns. The first three would need to be sortable.

1) Nucleotide Index: The index of the error. When a specific bar is selected, all values in this column would reflect the selected index.
2) Error Type: Indicates if the error is an insertion, mutation, or deletion. For clarity, these would each have a specific color font that would match the changes in the sequence comparison later.
3) Frequency: Indicates the frequency of the specific error.
4) Placeholder column: include text to differentiate what is found in the consensus sequence and what is in the binned sequence for this error.
5) Sequence Comparison: Showing an alignment of the consensus and binned sequence to show what the error looks like.
    (a) This is a must-have for the project - somewhere, we need to be able to drill down and clearly show the error in question.
    (b) To highlight the difference, a color change for the differential area would likely be required.

ii) Because this table would be pretty long, we'd likely need a cutoff of how many were presented at any given time. Say, it shows 20 rows, then provides the option of showing the next 20, etc.

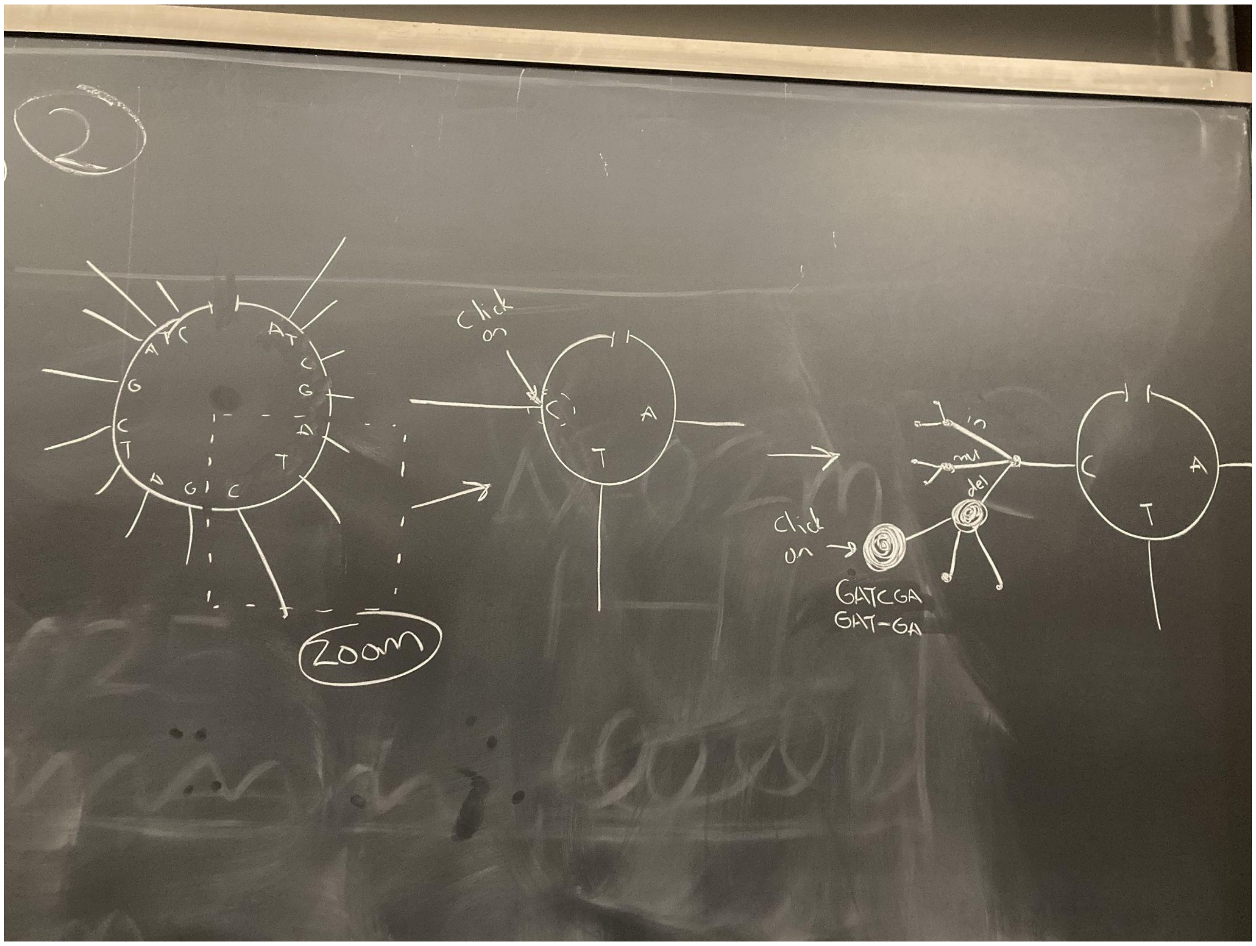
We started tinkering with what this may look like and created the following graphic for the bar chart (b) and the table (c), with the bar at index 1271 of the bar chart selected.

Bar Charts



Nucleotide Index	Error Type	Frequency	Sequence Comparison	
1271	deletion	174	Consensus Sequence: Bin Sequence:	`CCTCCCCCGTGCCTT` `CCTCCCC-GTGCCTT`
1271	mutation	12	Consensus Sequence: Bin Sequence:	`CCTCCCCCGTGCCTT` `CCTCCCCTGTGCCTT`
1271	insertion	3	Consensus Sequence: Bin Sequence:	`CCTCCCC-CGTGCCT` `CCTCCCCGCGTGCCT`
1271	insertion	2	Consensus Sequence: Bin Sequence:	`CCTCCCC-CGTGCCT` `CCTCCCCACGTGCCT`
1271	mutation	2	Consensus Sequence: Bin Sequence:	`CTCCCCCGTGCCTTC` `CTCCCCA-TGCCTTC`
1271	deletion	1	Consensus Sequence: Bin Sequence:	`CTCCCCCGTGCCTTC` `CTCCCC--TGCCTTC`

Prototype Visual 2



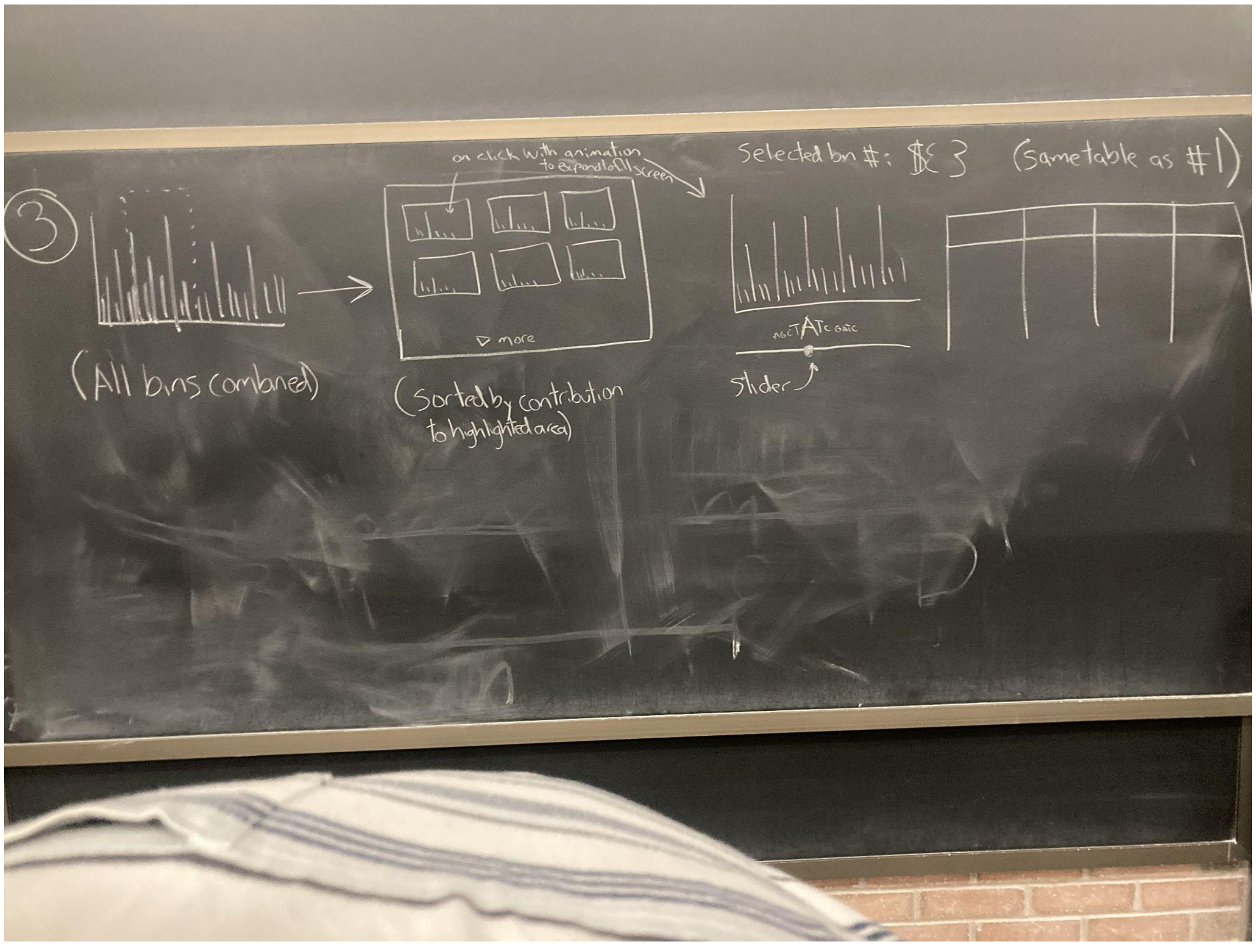
This visual was designed for portraying a single bin of sequences/consensus in a way that encapsulated all the data of b) and c) of prototype visual 1. There are three new representations introduced, namely a circular representation of the consensus sequence (left), a dynamic method for zooming in on particular subsections of that sequence (middle), and a tree representation of specific errors and their frequency (right).

a) The circle represents a kind of circular representation of the bar chart in 1.b. The circle itself is made of the nucleotides of the consensus sequences, with bars extending from the outside of the circle indicating frequency of errors (y-axis of the bar chart).
    i) Optional: We could color these bars based on the specific nucleotide at that index, one each for adenosine, thymine, cytosine, and guanine (ATCG).
b) The diagram in the middle of the above drawing would be formed from the dotted-line square drawn in a) above. Interactively drawing this square, all nucleotides outside of the

square are filtered out, and the selected portions would expand to fill the circle. The nucleotide font size would likely increase to fill the circle as best as possible.

c) The diagram on the right was a method we invented for portraying the frequency of specific errors at a given index. After clicking on a nucleotide (‘C’ was selected above), the bar previously used to indicate error frequency would be replaced by a tree. We may constrain the number of nucleotide indices that can have a tree expanded at a given time, possibly even to a single index. In this case, clicking on another index’s bar would collapse any open tree. A tree would have the following characteristics:
    i) The area of each node is used to express the frequency of all collective errors of the nodes in child branches. Therefore later nodes are either the same size as the parent or smaller.
    ii) A single ‘trunk’ from the circle to the first node. This merely extends it away from the remainder of the circle.
    iii) Up to three branches separating at the first node. These are for separating into error types of insertions, mutations, and deletions, respectively. The three nodes at the end of this branch and all children nodes would be colored by error type. For an example of coloring please see our final visual drawing.
    iv) A single branch is then created for each specific error identified at that nucleotide, ending with a node that represents the frequency of that specific error.

Prototype Visual 3


3
on click with animation to expand to fill screen
Selected bin #: ${ }
(Same table as #1)
▷ more
AGCTATCGATC
(All bins combined)
(Sorted by contribution to highlighted area)
Slider

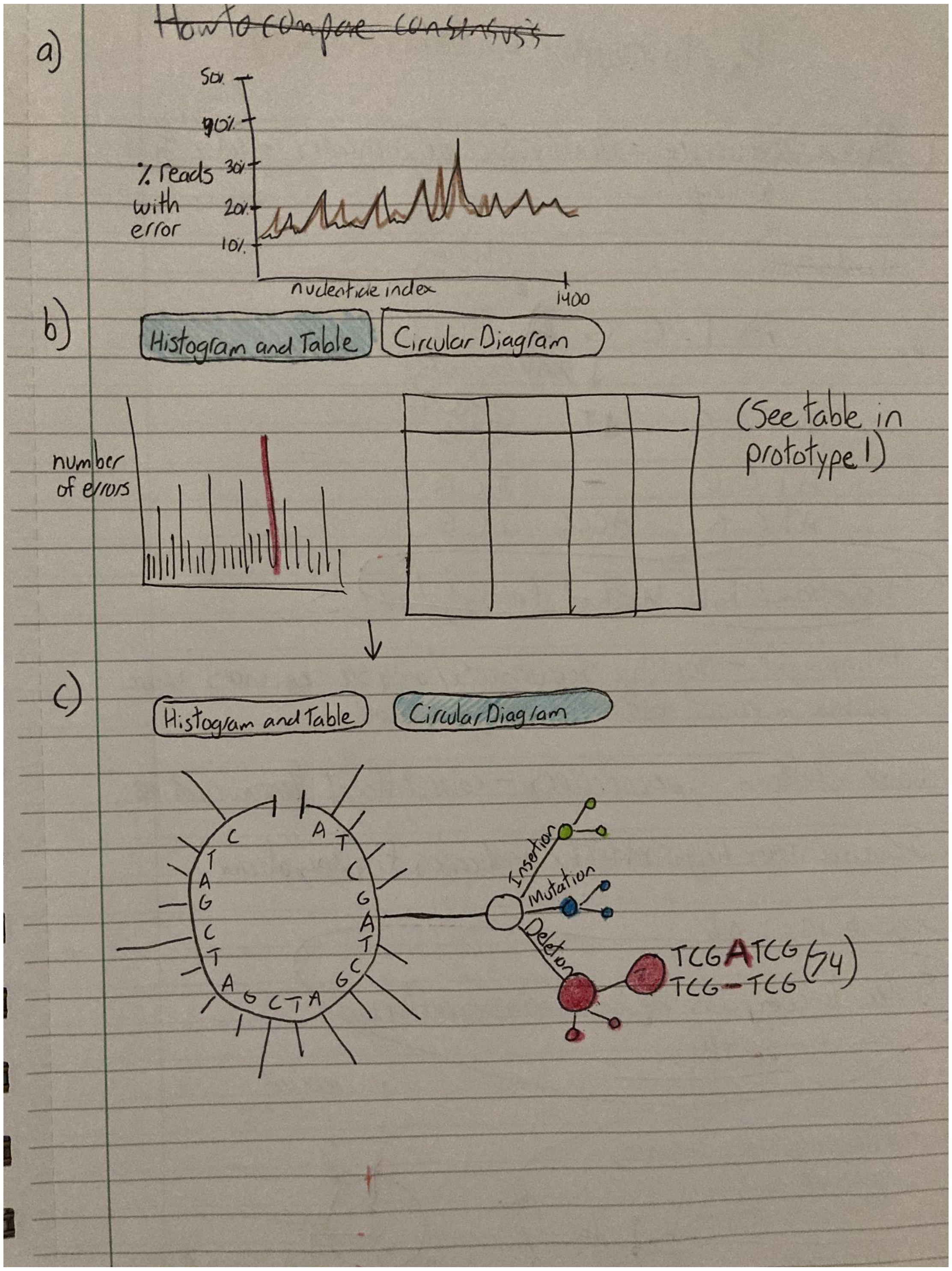
This prototype was developed to address two primary concerns. One, we felt we hadn't found a good way to visualize a consensus sequence in its entirety out of zooming options. With over a thousand base pairs or 'ticks' in a theoretical bar chart, these have not scaled well with our ideas. Second, we wonder if the line chart described in 1.a would become too cluttered to be meaningful. If there's a line for each consensus, and there are over 500 consensus sequences, there may be too much data for a line graph to reliably capture. So, this visual has the following characteristics:

a) The initial view will show all the errors from all the bins combined into one bar chart (left). These would share all characteristics of the bar chart in 1.b, but represent all bins instead of just one. This would reduce the clutter of a line graph. However, to inspect the individual contributions of specific bins, one can highlight a portion of the peaks to trigger the second view.
b) The second view is a grid of boxplots (middle left). After selecting a section of bars in the initial view, the program would determine which bins had on average the greatest percentage contribution of errors for the selected section. Each bin boxplot would be shown, starting with the top six 'contributing' bins.

    i) The selected portions would be highlighted in each boxplot of the grid, likely by changing their color.
    ii) There would be the option of revealing more boxplots after the initial six, either by expanding the bottom to reveal more than six boxplots or by being able to switch views to numbers seven through twelve, then thirteen through eighteen, etc.

c) The final view would be a barchart (middle right) and table (right), similar to what was described in prototype 1.b and 1.c. However, we altered the interactivity of the chart.
    i) Instead of interactively choosing a single bar in the bar chart, there would be a slider that ran along the bottom of the graph. Moving this slighter would highlight specific nucleotides by enlarging or magnifying them compared to their neighbors. The slider would only be able to highlight one nucleotide at a time, simultaneously magnifying the nucleotide and changing the color of the bar represented by that nucleotide (much like a mouseover event changing its color). When the slider is released, the table would filter according to the selected nucleotide index.

Final Visual



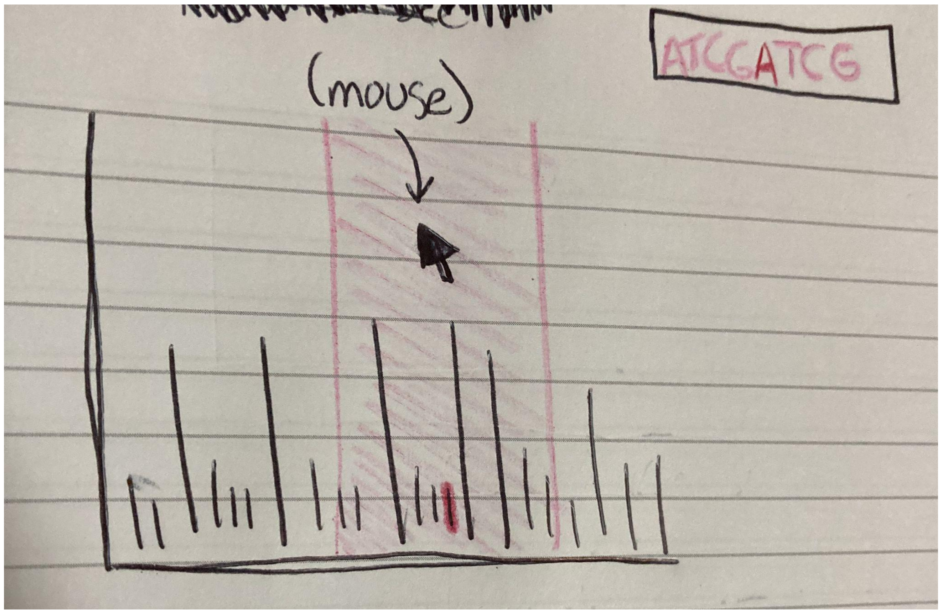
This visual summarizes several of our prototype ideas into a single graph. Note that we are allowing the user to choose between a “Histogram and Table” view as well as a “Circular Diagram” view, as described in 3.b and 3.c respectively.

a) We are currently settling on the line graph method (1.a) for showing the high-level consensus sequence comparisons, as that would allow us to select bins that have abnormal error patterns. This would be at the top of the visual.

b) This would be the histogram and table chart from 1.b and 1.c respectively. Some additions/alterations to the original idea described in prototype 1:
   i) Similar to prototype 2, we are going to implement a zooming function to the bar chart. Zooming will be performed using the mouse wheel or by brushing the overlay. Selection of bars in the histogram will be achieved by individual click or by brushing multiple bars, updating the table accordingly.
   ii) Instead of the slider described in prototype 3, we opted that a mouseover event that highlighted a subsection of 10-15 nucleotides would highlight sequences of interest more clearly. A box would appear over the 10-15 nucleotides around which the mouse was hovering, and those nucleotides would appear in a box over the histogram. See below image for a drawing.

c) This would be the circular graph described in prototype 2. The nucleotide highlight box described in 3.b.ii would apply here on mouse hover of the frequency bar.
   i) Note: We opted to not use the slider because it would not translate well to the circular diagram.
   ii) For all zoom/selection functions, there will be means to return to the initial state.

For the 3.b.ii description:



Must-Have Features

- Global view
    - Line chart
        - Line selection
            - Transition to Single view
        - Crowded Selection
            - Brush and Zoom
- Single view
    - Navigation/State
        - Back (to Global view) button
        - Reset (state/selection) button
    - Histogram and Table view
        - Histogram
            - Bar selection
                - Single Selected Color
                - Populates table
        - Table
            - Sortable
                - Index
                - Type
                - Frequency
            - Sequence Comparison
                - Discrepancy colored
    - Hybrid view
        - Radial
            - Brushing (2D) Subset Selection
                - Selection Spreads to Fill Radial
            - Bar Selection
                - Populates Tree
        - Tree/Network
            - Hierarchical Expand/Collapse
            - Leaf Selection
                - Frequency
                - Sequence Comparison

Optional Features

- Global view
  - Line chart
    - Line selection
      - Transition to Single view
    - Crowded Selection
      - Brush and Zoom
      - Brush/Pan and Zoom
- Single view
  - Navigation/State
    - Back (to Global view) button
    - Reset (state/selection) button
  - Histogram and Table view
    - Histogram
      - Bar selection
        - Single Selected Color
        - Populates table
      - Brush and Zoom
      - Brushing Bar selection
        - 1D (x-axis)
        - 2D (x-axis & Y-axis)
        - Nucleotide-Specific Selected Color
        - Populates table
      - Histogram Filtering
        - Error Type(s) (Insertion/Deletion/Mutation)
      - Bar Hover
        - Peek Nucleotide Sequence
    - Table
      - Sortable
        - Index
        - Type
        - Frequency
      - Sequence Comparison
        - Discrepancy colored
    - Hybrid view
      - Pan/Zoom
      - Radial
        - Brushing (2D) Subset Selection
          - Selection Spreads to Fill Radial
        - Bar Selection
          - Populates Tree
        - Nucleotide Labels
          - Scale to Fit
        - Bar Hover
          - Peek Sequence
      - Tree/Network
        - Hierarchical Expand/Collapse
        - Leaf Selection
          - Frequency
          - Sequence Comparison
  - Consensus vs Selected Bin Sequence(s) view

Project Schedule

- Week 10 – **27 Oct** 2022 @ 23:59
  - Project Peer Feedback – **25 Oct 2022**
  - Goals
    - Histogram – **CC**
      - Bar Selection – **CC**
      - Crowded Selection - Brush and Zoom – **AM**
  - Stretch Goals
    - Line Chart – **AM**
      - Line Selection – **AM**
    - Table – **CC**
      - Sortable – **CC**
      - Sequence Comparison – **CC**

- Week 11 – **3 Nov** 2022 @ 23:59
  - Goals
    - Line Chart – **AM**
      - Line Selection – **AM**
      - Crowded Selection - Brush and Zoom – **AM**
    - Histogram
      - Bar Hover – **CC**
        - Peek Nucleotide Sequence – **CC**
      - 1D Brushing Selection – **CC**
  - Stretch Goals
    - Radial – **CC**
    - Navigation – **AM**

- Week 12 – **10 Nov** 2022 @ 23:59
  - Schedule time-slot with Staff Member – **11 Nov 2022**
    - Project Review Meeting – Following Week
  - Goals
    - Radial
      - 2D Brushing Subset Selection – **CC**
      - Bar Selection – **AM**
        - Populate Tree – **AM**
    - Tree – **AM**
  - Stretch Goals
    - Radial
      - Bar Hover – **CC**
        - Peek Nucleotide Sequence – **CC**
    - Tree
      - Expand/Collapse – **AM**

- Week 13 – **17 Nov** 2022 @ 23:59
    - Project Review Meeting with Mentor – 14-18* **Nov 2022**
    - Goals
        - Radial – **CC**
        - Tree
            - Node
                - Scaling – **CC**
                - Coloring – **CC**
            - Leaf Selection – **AM**
                - Frequency – **AM**
                - Sequence Comparison – **AM**
    - Stretch Goals
        - Hybrid
            - Pan and Zoom – **AM**
            - Radial
                - Scaling – **CC**
                - Coloring – **CC**

- Week 14 – **24 Nov** 2022 @ 23:59
    - Goals
        - Histogram
            - 2D Brushing Selection – **AM**
            - Error Type Filtering – **CC**
    - Stretch Goals
        - Tree
            - Leaf Selection
                - Consensus vs Bin Sequence(s)
                    - Line Chart – **AM**
    - **Thanksgiving Break**

- Week 15 – **1 Dec** 2022 @ 23:59
    - Final Project Due – **2 Dec 2022**
    - Goals
        - Transitions/Scaling/Navigation/Misc – **AM**/**CC**
    - Stretch Goals
        - Consensus vs Bin Sequence(s)
            - Line Chart – **AM**
                - Subset/Filtering/Selection – **CC**

Process Book

Entry 1: Data Processing - Caleb

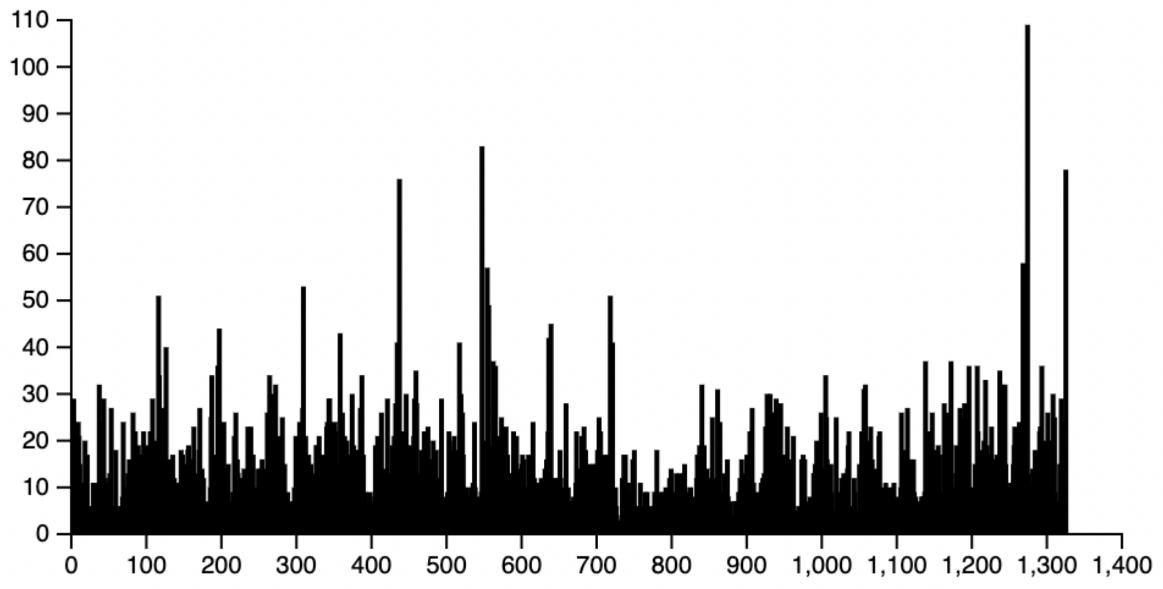
In my Python work code, I included the following function. It finds the differences between two sequences and returns them in a `(<start index>, <end index>, <insert>)` format. Each tuple indicates a change necessary for updating the first sequence (generally the consensus sequence) to look like the second sequence. This can be done by replacing whatever lies between the start/end index of the string with the insert string.

This function relies on the PairwiseAligner object from biopython.
```
from Bio.Align import PairwiseAligner

def find_aligned_differences(seq1, seq2):

    aligner = PairwiseAligner()
    alignments = aligner.align(seq1,seq2)
    indices = alignments[0].aligned
    a_idx, b_idx = list(indices[0]), list(indices[1])
    a_idx.insert(0, (0,0))
    b_idx.insert(0, (0,0))
    a_idx.append((len(seq1),len(seq1)))
    b_idx.append((len(seq2),len(seq2)))

    diffs = []

    for i in range(len(a_idx)-1):
    start = a_idx[i][1]
    end = a_idx[i+1][0]
    insert = seq2[b_idx[i][1]:b_idx[i+1][0]]
    if start == end and len(insert)==0: continue
    diffs.append((start, end, insert))
    return diffs
```

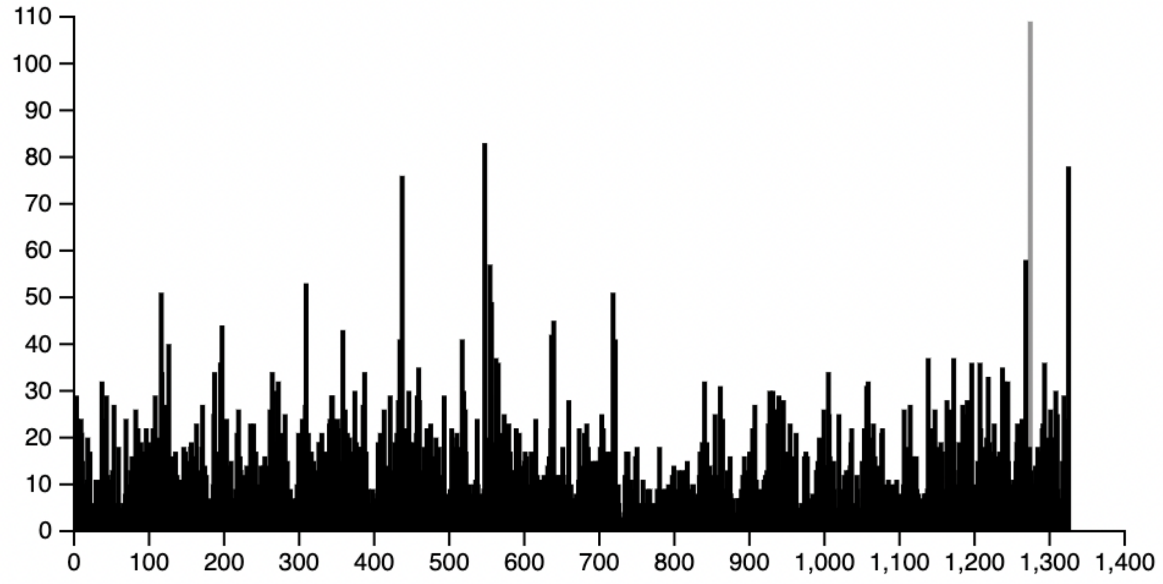
For the purposes of this project, I include with each tuple the bin number and sequence identifier, as previously described, and save them to a csv file.

Entry 2: Barplot Histogram - Caleb

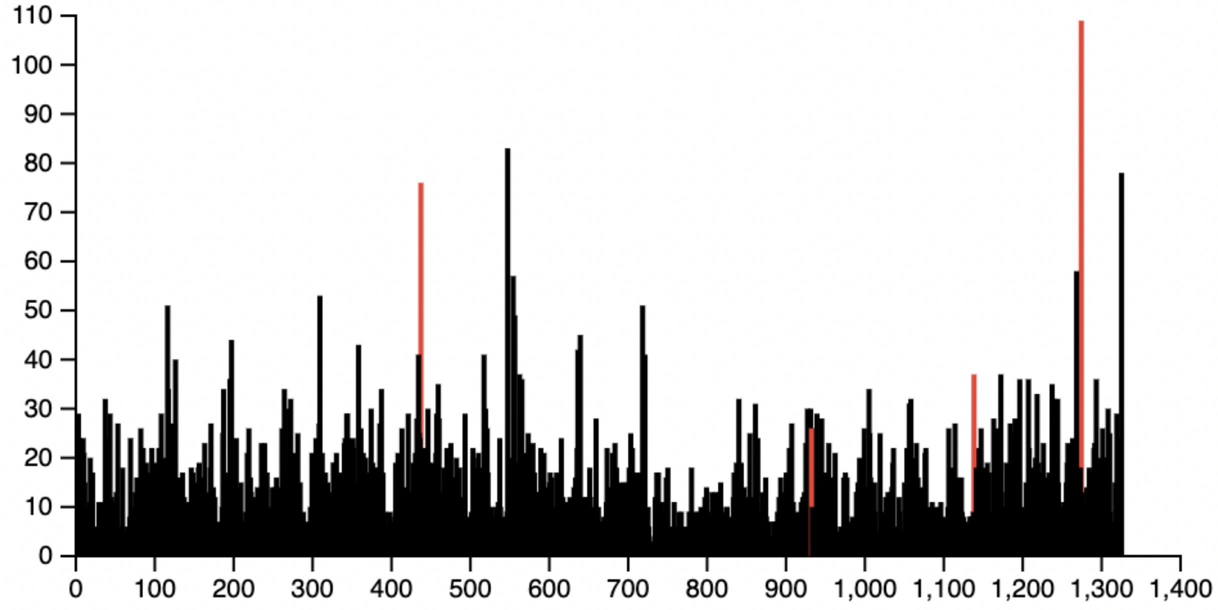
We've already put together a bar chart for homework 3, and I relied largely on my previous code and the solution code provided by the TAs to make an initial barchart. I kept the barchart in its own file, separate from the `setup.js` where I preprocessed the data. The data was grouped by the "start" index for each error - so it's a bit generalized, a flaw that will be covered by the table.



I made the graph interactive in two ways. One, lines turn grey in a mousover event, and turn back to black after a mouseout event. See the tallest peak below for an example.



Second, when a bar is clicked, it turns red. Clicking on a red bar will turn it black again. The intent is to filter the table by the selected bars, but for now I just added the coloring functionality. I highlighted a few just to illustrate.



TODO: I still need to label the axes. The y-axis is an error count, and the x-axis is nucleotide position.

TODO: There’s a gap between the last nucleotide (~1320) and the end of the graph. It should go right to the end.

TODO: I notice some of the red bars in the above example are semi-hidden by black bars - they don’t quite make it to the x-axis. I should probably adjust that so that smaller ones are more visible.

Entry 3: Table - Caleb

I wrote up some basic table code. The table has 5 columns, namely the nucleotide index, the error type, the frequency, an unlabeled description column, and the sequence comparison. All table values are text-based, but I did some tricky coloring for some of them. Data was grouped by [start, end, insert] values, and each row indicates a group.

First, the nucleotide index would just indicate the start index of a group. This is a bit generalized - for example, if a deletion took out the three nucleotides after the start index, the credit would still go to the first one. I don’t see a way around that.

Second, the error type. I used the following code to determine if each value was a mutation, deletion, or insertion. I made a color scale for these three values.

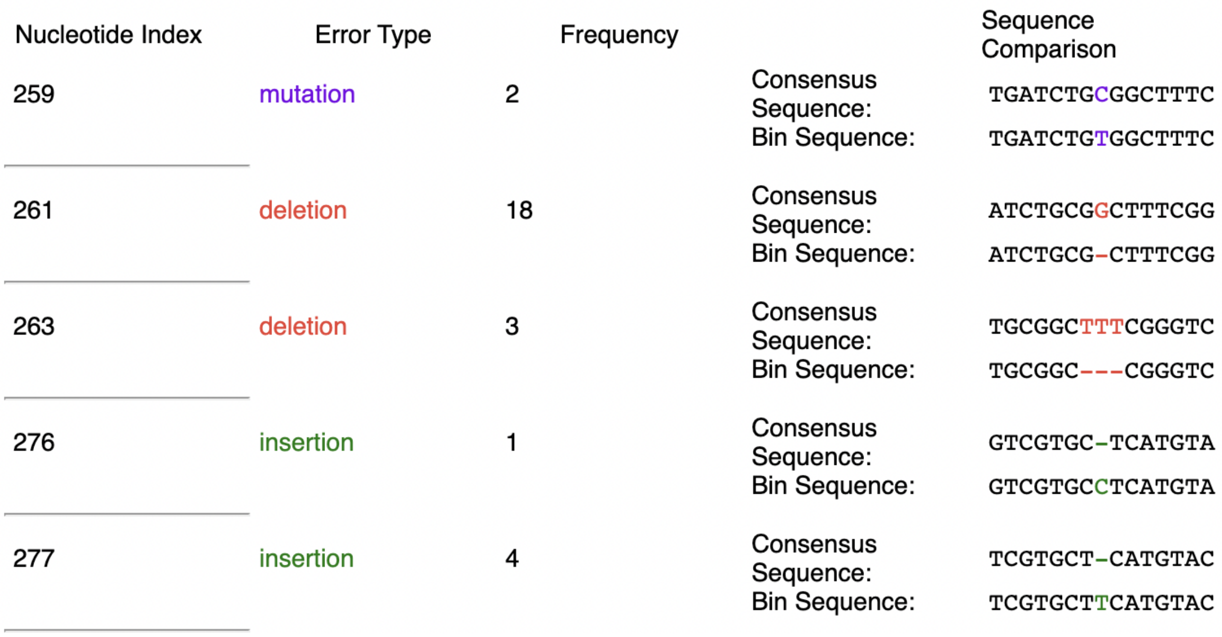
```
  determineErrorType(d){
          if (d.start === d.end) {return 'insertion';}
          else if (d.insert.length !== 0) { return 'mutation';}
          else {return 'deletion'};
  }
```

I haven’t found any edge cases where this function misassigns an error type, but I’ll keep an eye out.

Third, frequency. This is just the number of values in each group, pretty straightforward.

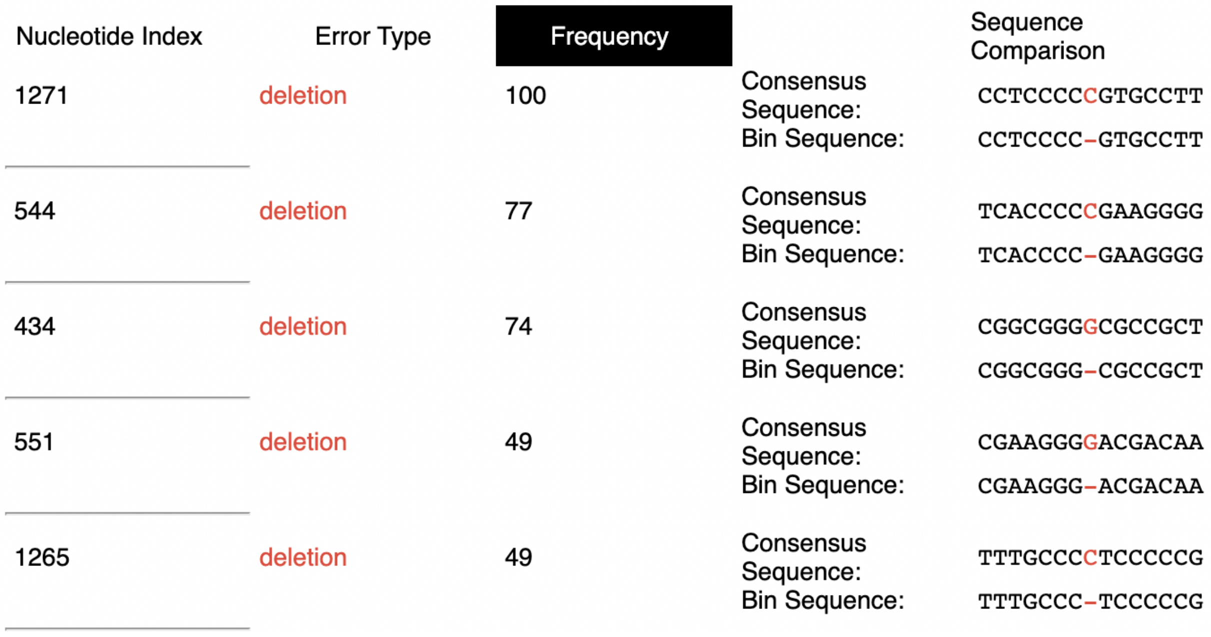
Fourth, sequence comparison. This was tricky - I wanted to highlight the difference between the consensus sequence and the read sequence, preferably altering the color so it matched the error type. I also needed to split each group into two rows so one sequence was right over the other, leaving the other column values blank in the second row, aside from the unlabeled column that distinguishes consensus from binned sequence. The letters of the sequences also needed to be the same width so you could have a 1:1 view of each nucleotide comparison.

Ultimately, I got it to work. The sequence text wound up being three <tspan> objects, with the middle <tspan> colored by error type. I used a font that had identical character widths.



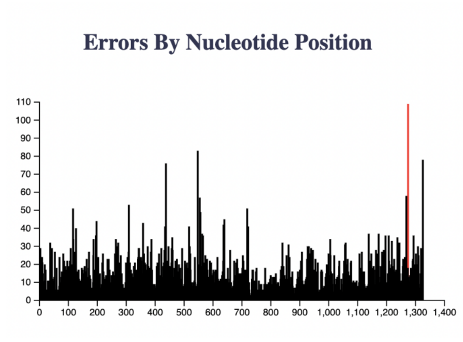
Nucleotide Index	Error Type	Frequency		Sequence Comparison
259	mutation	2	Consensus Sequence: Bin Sequence:	TGATCTGCGGCTTTC TGATCTGTGGCTTTC
261	deletion	18	Consensus Sequence: Bin Sequence:	ATCTGCGGCTTTCGG ATCTGCG-CTTTCGG
263	deletion	3	Consensus Sequence: Bin Sequence:	TGCGGCTTTCGGGTC TGCGGC---CGGGTC
276	insertion	1	Consensus Sequence: Bin Sequence:	GTCGTGC-TCATGTA GTCGTGCCTCATGTA
277	insertion	4	Consensus Sequence: Bin Sequence:	TCGTGCT-CATGTAC TCGTGCTTCATGTAC

I also included sorting functionality for all the labeled columns.



Nucleotide Index	Error Type	Frequency		Sequence Comparison
1271	deletion	100	Consensus Sequence: Bin Sequence:	CCTCCCCCGTGCCTT CCTCCCC-GTGCCTT
544	deletion	77	Consensus Sequence: Bin Sequence:	TCACCCCCGAAGGGG TCACCCC-GAAGGGG
434	deletion	74	Consensus Sequence: Bin Sequence:	CGGCGGGGCGCCGCT CGGCGGG-CGCCGCT
551	deletion	49	Consensus Sequence: Bin Sequence:	CGAAGGGGACGACAA CGAAGGG-ACGACAA
1265	deletion	49	Consensus Sequence: Bin Sequence:	TTTGCCCCTCCCCCG TTTGCCC-TCCCCCG

And, lastly, I created a constant object that would be shared between the barplot, setup, and table files. When one clicks on a bar in the barchart, in addition to highlighting in read it also filters the table to only include values from highlighted bars.



Nucleotide Index	Error Type	Frequency		Sequence Comparison
1271	deletion	100	Consensus Sequence:	CCTCCCCCGTGCCTT
			Bin Sequence:	CCTCCCC-GTGCCTT
1271	insertion	2	Consensus Sequence:	CCTCCCC-CGTGCCT
			Bin Sequence:	CCTCCCCGCGTGCCT
1271	mutation	5	Consensus Sequence:	CCTCCCCCGTGCCTT
			Bin Sequence:	CCTCCCCTGTGCCTT

TODO: I never did make a great line division between grouped rows. I have a small one that only reaches to the end of the first column, and it works, but it would probably work better if I could make it run entirely across.

TODO: On that note, the spacing still feels off to me. Maybe center the text everything, just like the header? Increase the width of the unlabelled column and decrease it for Frequency? Something to chew on.

TODO: It’s not necessary, but I couldn’t get the up/down sorting arrows to appear like we did for homework 4. That would be a benefit to the table.

Entry 4: Peer Feedback - Alex

See “Data Processing” above.

- Line Graph
  - Sam Himes
    - What is the purpose of looking at all bins at same time (line graph)
  - Sam Himes
    - Draw confidence interval area and only color lines that come outside this std dev/etc.
  - John Stanley
    - Consider making y-axis be Std Dev which would spread data more
  - John Stanley
    - Consider adding dot/point selection
  - Sam Himes
    - Consider a slider to increase/decrease confidence intervals (e.g., if 99% confidence already has a bunch of lines colored, adjusting to 95% might reduce to a few that lines are colored. Maybe standard deviation instead
      - (e.g., 1-std many lines colored, increase to 3-std and only a few are colored)
- Histogram/Bar Chart
  - John Stanley
    - Consider making y-axis be percent or adding second y for comparability among bins
- Radial/Tree
  - John Stanley
    - Be careful to avoid clutter
  - Sam Himes
    - Rotate on different selection
  - John Stanley/Sam Himes
    - Numbers of sum(count) in each insert/delete/mutate parent node if room

During peer feedback, we highlighted our difficulty in determining how to approach the global, multi-bin view. For this reason, we solicited as much feedback regarding the Line Graph as possible and John/Sam were happy to oblige, providing many plausible approaches we had yet to consider.

The primary obstacle with the global view is that, unlike the other views which are limited to one bin’s consensus sequence and error profile, the global view may have any number of bins’ consensus sequence error profiles.

In hindsight, after recent lecture material, their suggestions essentially boil down to integrating an analytic component into the view.

One implementation could be to add a global line with a confidence band along the nucleotide position and have an interactive component to adjust the extent of the band, using this as a filtering mechanism.

We will find out more as we progress through the implementation, but whatever the case, applying analytic methods will certainly be of value–even if the implementation differs slightly.

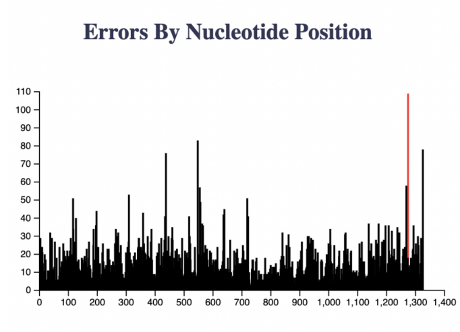
The only feedback regarding the implementation of the single-bin histogram/bar chart was comparability among bins, such as when switching between bins. The note here was to consider adding a second Y-axis that is based on percent rather than absolute magnitude. We intend to keep this in mind; however, if we find a decent solution to our global view implementation, this may be redundant.

Regarding the Radial/Tree view, the common concern was space constraints and clutter considerations. This will be one of our primary obstacles to achieving a clean feeling implementation; however, the suggestion to rotate the nucleotide sequence on selection of a different nucleotide position will help toward that end. In the case that a nucleotide position has already been selected and the tree for that location is expanded, selecting another position will collapse the existing tree, rotate the sequence, and expand the tree for the selection position in the same location the tree was previously.

This is a good idea and could give a grounded feel to the view that might otherwise be overwhelming and floating.

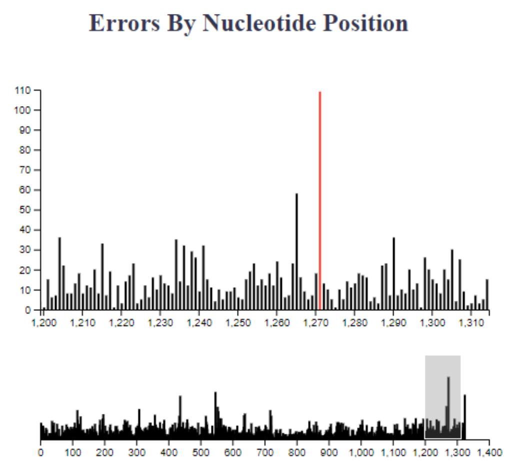
Sam and John provided excellent feedback, all of which we will keep in mind as we proceed.

Entry 5: Focus + Context (Zoom) Functionality - Alex



Nucleotide Index	Error Type	Frequency		Sequence Comparison
1271	deletion	100	Consensus Sequence:	CCTCCCCCGTGCCTT
			Bin Sequence:	CCTCCCC-GTGCCTT
1271	insertion	2	Consensus Sequence:	CCTCCCC-CGTGCCT
			Bin Sequence:	CCTCCCCGCGTGCCT
1271	mutation	5	Consensus Sequence:	CCTCCCCCGTGCCTT
			Bin Sequence:	CCTCCCCTGTGCCTT

When working with the focus-only sequence-error histogram, it was difficult to select nucleotide positions that were surrounded by higher error bars. This was one of the high value-added areas we wanted to address early on. Implementing a rescale of the X-axis on zoom would have been sufficient, but context would be lost while zoomed if we settled on this solution. For this reason, I implemented a zoom feature on the existing focus and added a context below. One additional benefit of the context view is the ability to move the brushed region to traverse the x-axis nucleotide position (rather than zoom out and/or click and drag repeatedly).



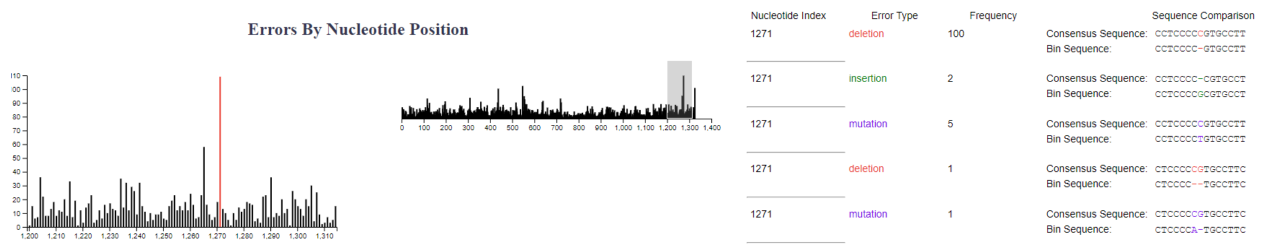
Nucleotide Index	Error Type	Frequency		Sequence Comparison
1271	deletion	100	Consensus Sequence:	CCTCCCCCGTGCCTT
			Bin Sequence:	CCTCCCC-GTGCCTT
1271	insertion	2	Consensus Sequence:	CCTCCCC-CGTGCCT
			Bin Sequence:	CCTCCCCGCGTGCCT
1271	mutation	5	Consensus Sequence:	CCTCCCCCGTGCCTT
			Bin Sequence:	CCTCCCCTGTGCCTT
1271	deletion	1	Consensus Sequence:	CTCCCCCGTGCCTTC
			Bin Sequence:	CTCCCC--TGCCTTC

One difficulty was the existing transitions on the nucleotide error bars. Aesthetically pleasing transitions existed for the case where data is loaded in, such as when switching to another sequence bin. However, when traversing the X-axis using the context brush, the X-axis would move and the bars would lag behind by the transition duration.

Upon investigating this, I found that I could mitigate the issue and resolve a performance issue with my initial F+C implementation. The MVP (minimum viable product) implementation of the F+C feature simply redrew the bars, which was highly inefficient. Selecting the appropriate rect elements and updating their x attribute was sufficient and reduced computational expense.

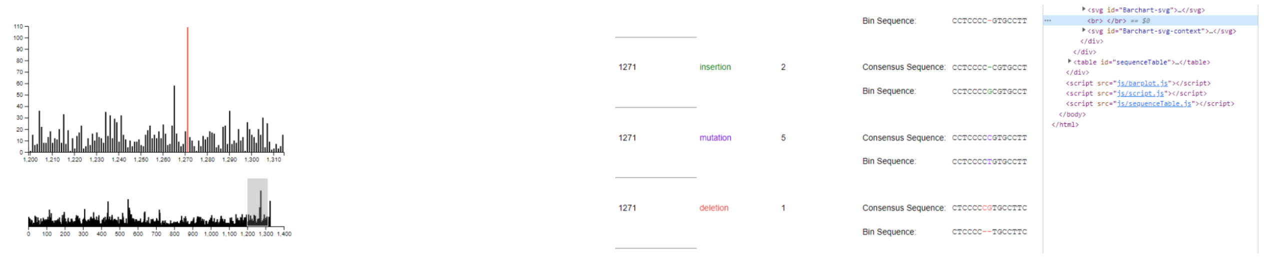
While working in this part of the code, I also refactored the filtering of all data upon bin change for a 3x yield in processing expense.

TODO: The filtering of sequences into bins could likely stand further improvement by creating a dictionary of bins that is sorted once. On launch, the default bin would be sorted immediately and the remaining bins can be asynchronously. Alternatively, we may be able to rollup the data by grouping by bin and then start/end/insert within bins. For the current single-bin view, the current implementation is not a hindrance, but as we implement the multi-bin view this issue may need to be addressed.



Nucleotide Index	Error Type	Frequency	Sequence Comparison
1271	deletion	100	Consensus Sequence: CCTCCCCCGTGCCTT Bin Sequence: CCTCCCC-GTGCCTT
1271	insertion	2	Consensus Sequence: CCTCCCC-CGTGCCT Bin Sequence: CCTCCCCGCGTGCCT
1271	mutation	5	Consensus Sequence: CCTCCCCCGTGCCTT Bin Sequence: CCTCCCCTGTGCCTT
1271	deletion	1	Consensus Sequence: CTCCCCCGTGCCTTC Bin Sequence: CTCCCC--TGCCTTC
1271	mutation	1	Consensus Sequence: CTCCCCCGTGCCTTC Bin Sequence: CTCCCCA-TGCCTTC

TODO: In order to reduce complexity and maximize code reuse, the focus and context were placed into separate SVGs. Now that the implementation is fairly mature, we may be in a better place to re-evaluate if this is still the case. The focus and context are currently in a container that will allow them to flex, given enough room, which causes the behavior seen above. Simply adding a break element between them resolves the issue. So, if we want to stay with the current design, this validates that we can use simple CSS or HTML to mitigate the undesirable behavior.



			Bin Sequence: CCTCCCC-GTGCCTT
1271	insertion	2	Consensus Sequence: CCTCCCC-CGTGCCT Bin Sequence: CCTCCCCGCGTGCCT
1271	mutation	5	Consensus Sequence: CCTCCCCCGTGCCTT Bin Sequence: CCTCCCCTGTGCCTT
1271	deletion	1	Consensus Sequence: CTCCCCCGTGCCTTC Bin Sequence: CTCCCC--TGCCTTC

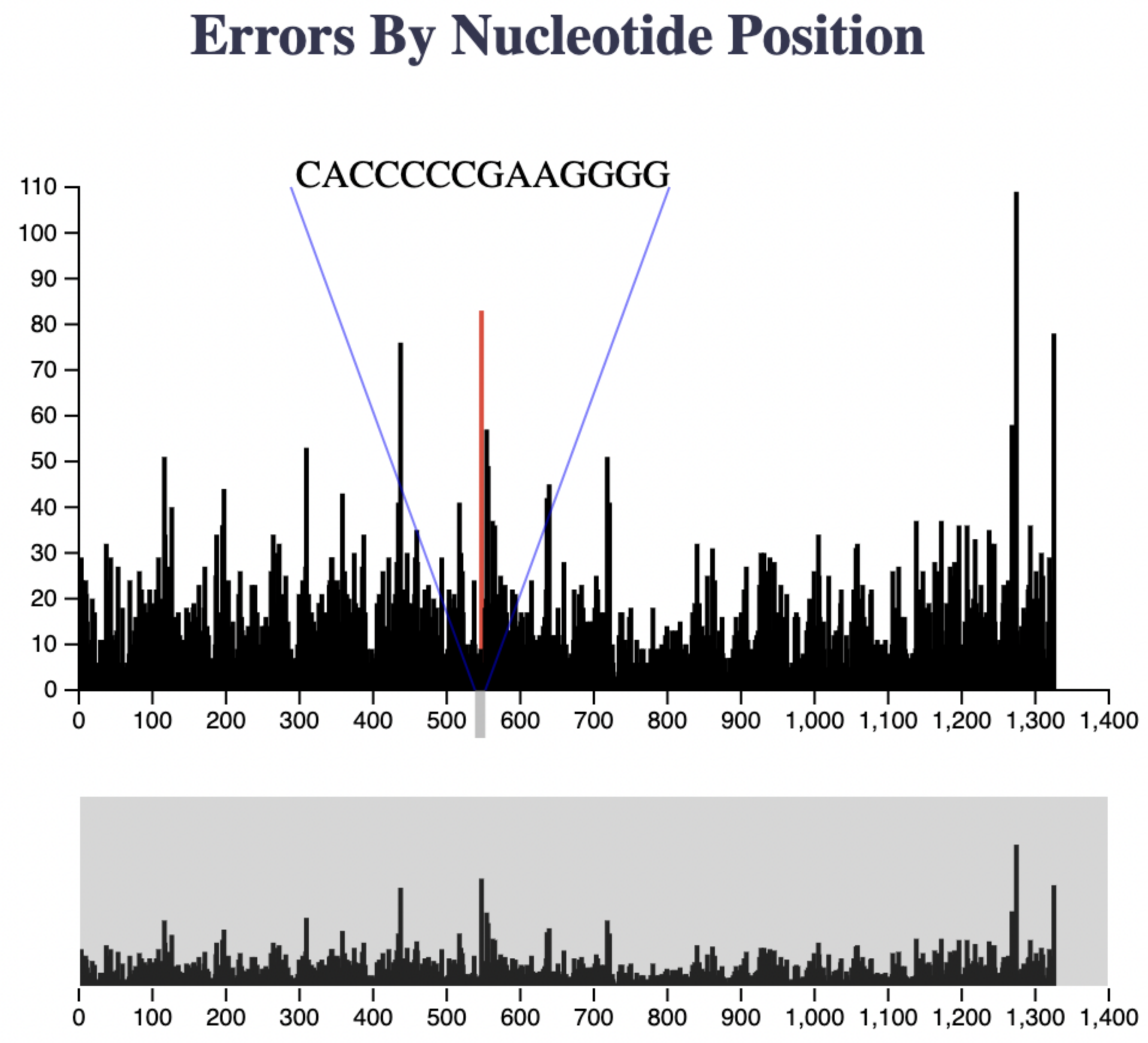
```
        ▸<svg id="Barchart-svg">…</svg>
...      <br> </br> == $0
        ▸<svg id="Barchart-svg-context">…</svg>
        </div>
      </div>
     ▸<table id="sequenceTable">…</table>
    </div>
    <script src="js/barplot.js"></script>
    <script src="js/script.js"></script>
    <script src="js/sequenceTable.js"></script>
  </body>
</html>
```

Entry 6: Sequence Text Highlighter - Caleb

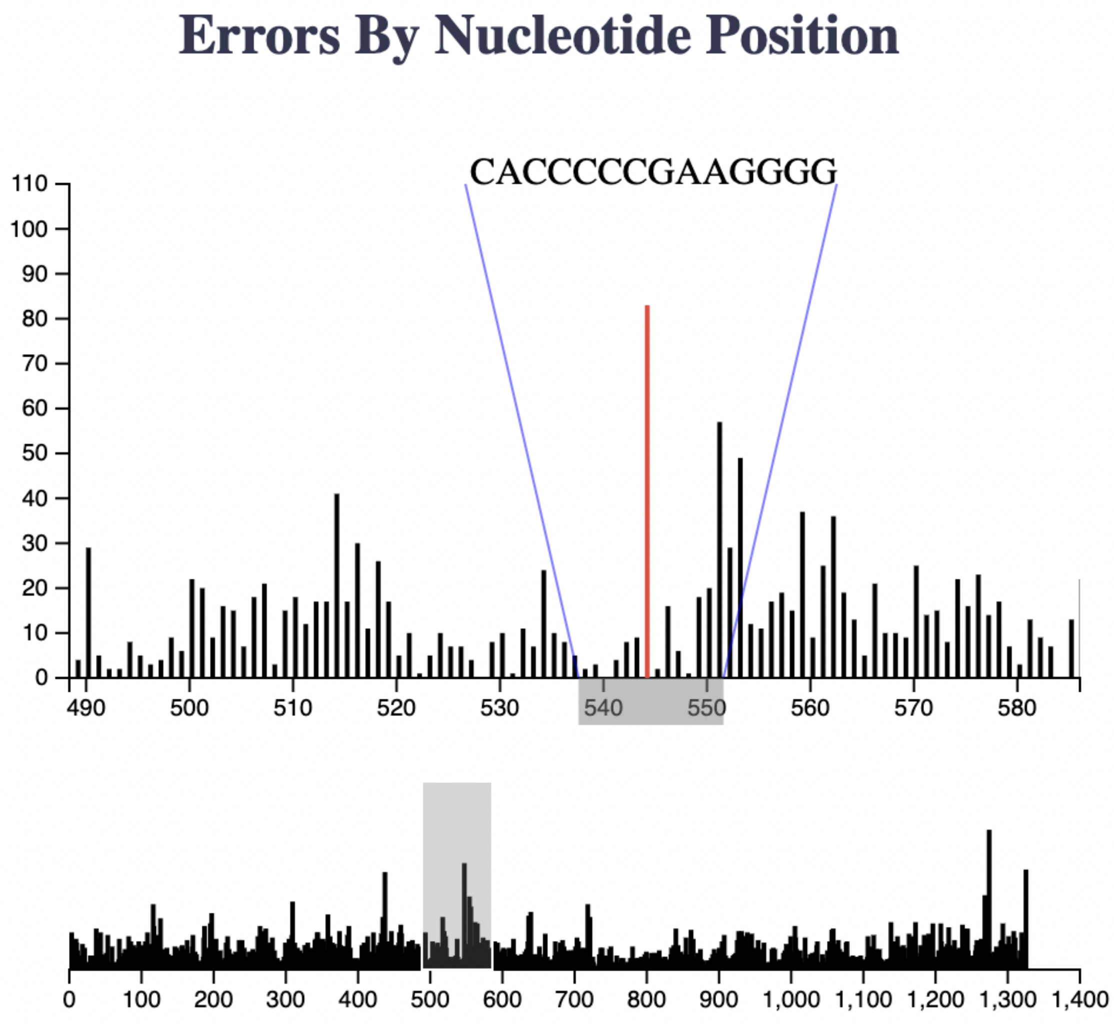
This is more of a rough draft attempt - it does what I want it to do, but it doesn't look… professional. Still mulling it over.

I wanted to include functionality that would highlight what a sequence looks like over a chosen section of the barchart. Each x-axis value has a corresponding nucleotide in the sequence, and in exploring the data one would want to know what that nucleotide is in addition to the surrounding nucleotides. You can click on it and see the subsequence in the table… but it doesn't scratch the immediate itch of "what am I looking at right now?"

So I added the following functionality. A grey, transparent box runs along the bottom, highlighting the nucleotides depicted. Blue lines expand to the top, like a magnification illustration. And the given sequence is conveyed at the top.



This view is perhaps more clear when you zoom in substantially.



I feel like it’s still not very professional-looking, but am running out of ideas on how to chang that.

TODO: Review this functionality with peers/the TA. Does anyone have any ideas on how to make it look better?