# Surveying Features of Spherical Viruses

Gabe Orosan-Weine
Advisor: Dr. David Wilson
Department of Physics

# Contents

# 1 Introduction

## 1.1 Viruses

There are estimated to be $10^{31}$ viruses on earth, infecting organisms across all domains of life. Viruses contain genetic material - DNA or RNA, each of which can be single or double stranded - but rely on host cell machinery (to varying degrees) in order to replicate; they cannot reproduce alone and are thus typically classified as nonliving. The capsid of a virus, meaning the protein shell around the viral genome, is closely related to the function of the virus, containing spike proteins that dictate how the virus binds to and enters target cells. Understanding the relationship between the capsid of the virus and its other properties is of medical relevance for pharmaceutical and vaccine design, and could be useful for many other biological inquiries.

Viruses are categorized into three main groups: helical viruses, spherical viruses (those that have icosahedral symmetry), and complex viruses. Spherical viruses have a protein capsid shell that exhibits icosahedral symmetry (with protrusions called spikes) surrounding the genetic material. This paper will focus on the properties of spherical viruses.

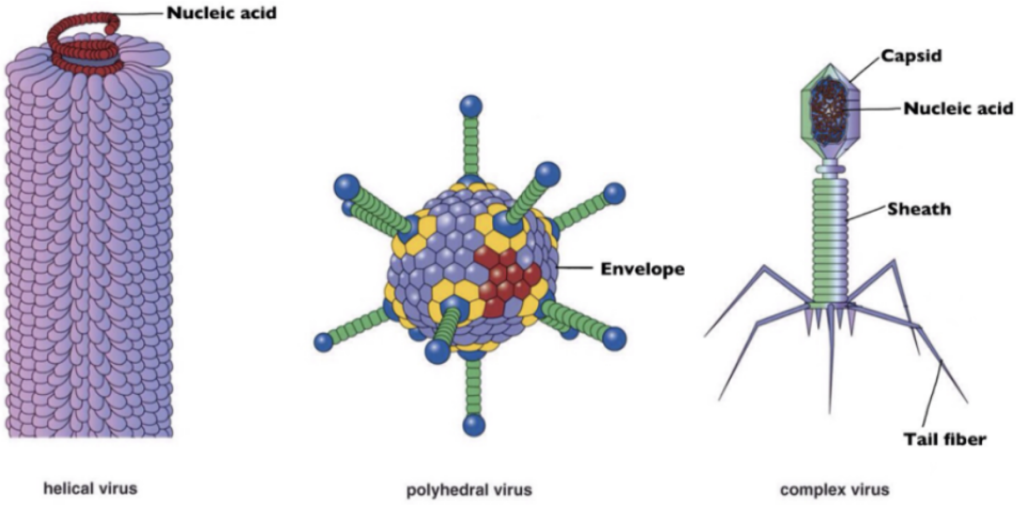



Figure 1. The three main types of viruses. Viruses can be Helical, polyhedral (including spherical viruses) or complex (Evans, 2020). The typical bacteriophage structure pictured is one of many types of complex viruses.

## 1.2 Triangulation Number

One of the main ways spherical capsids are further characterized is by their triangulation number (T-Number), describing the number of proteins that form a virus's Asymmetric Unit (AU), which is then tiled 60 times to form the full capsid shell.

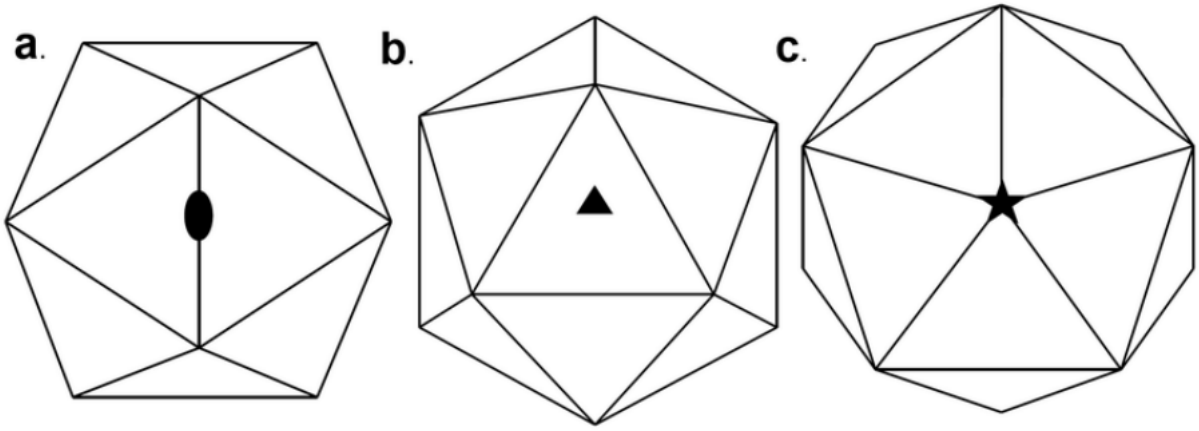
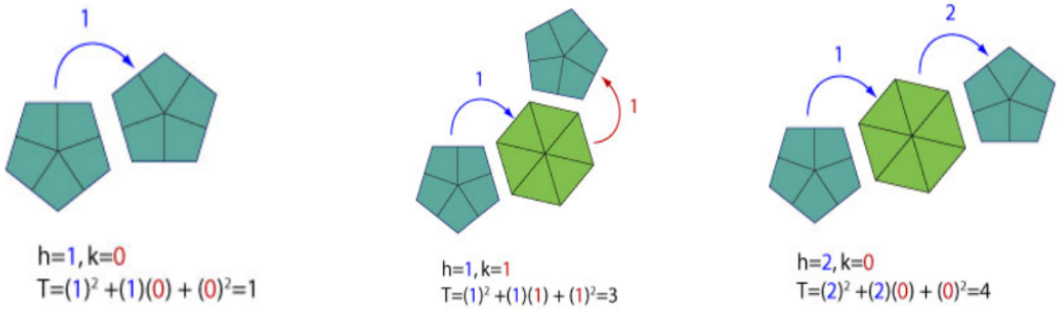



Figure 2. The respective two, three, and five-fold axes of symmetry on a regular icosahedron (Evans, 2020).

These shells then have 60 rotational symmetries across 5-Fold, 3-Fold, and 2-Fold axes reflecting the 60 asymmetric unit pieces. The symmetries, shown in Figure 2, denote how many rotations you can do before you end up back where you started.




(a) T = 1 (b) T = 3 (c) T = 4

Figure 3. (a) T = 1 (b) T = 3 (c) T = 4 In Figure 3.a, pentagons are touching so it only takes one step to get to the next; $T = 1^2 + (1)(0) + (0)$ = 1. In Figure 3.b, h and k are both equal to 1 because you must take one step in both directions, giving T=3. In the final case, you must take two steps in the same direction which results in T=4.

Formally, T-Number is calculated by $T = h^2 + hk + k^2$, where h and k are the number of hexamers separating two pentamers in each direction. Since h and k must be whole numbers, T-Numbers are somewhat restricted; for example there are no T5 or T6 viruses.

## 1.3 Point Arrays/Gauge Points

While T-Number is a useful classification tool, it does not take into account different radial levels or packing of genetic material inside, and thus does not have the ability to delineate some structural properties of interest. Previous research has shown the ability of point arrays (formed by rotation and extending the icosahedral shell in predetermined ways) to identify geometric constraints that must be obeyed by the capsid (Wilson 2020). The next step was to investigate how point arrays of spherical virus capsids relate to protrusions and genome composition; it was found that the locations of spike protein protrusions correspond with certain gauge points of point arrays and some protrusion points are only used by RNA viruses (Wilson & Roof, 2021).

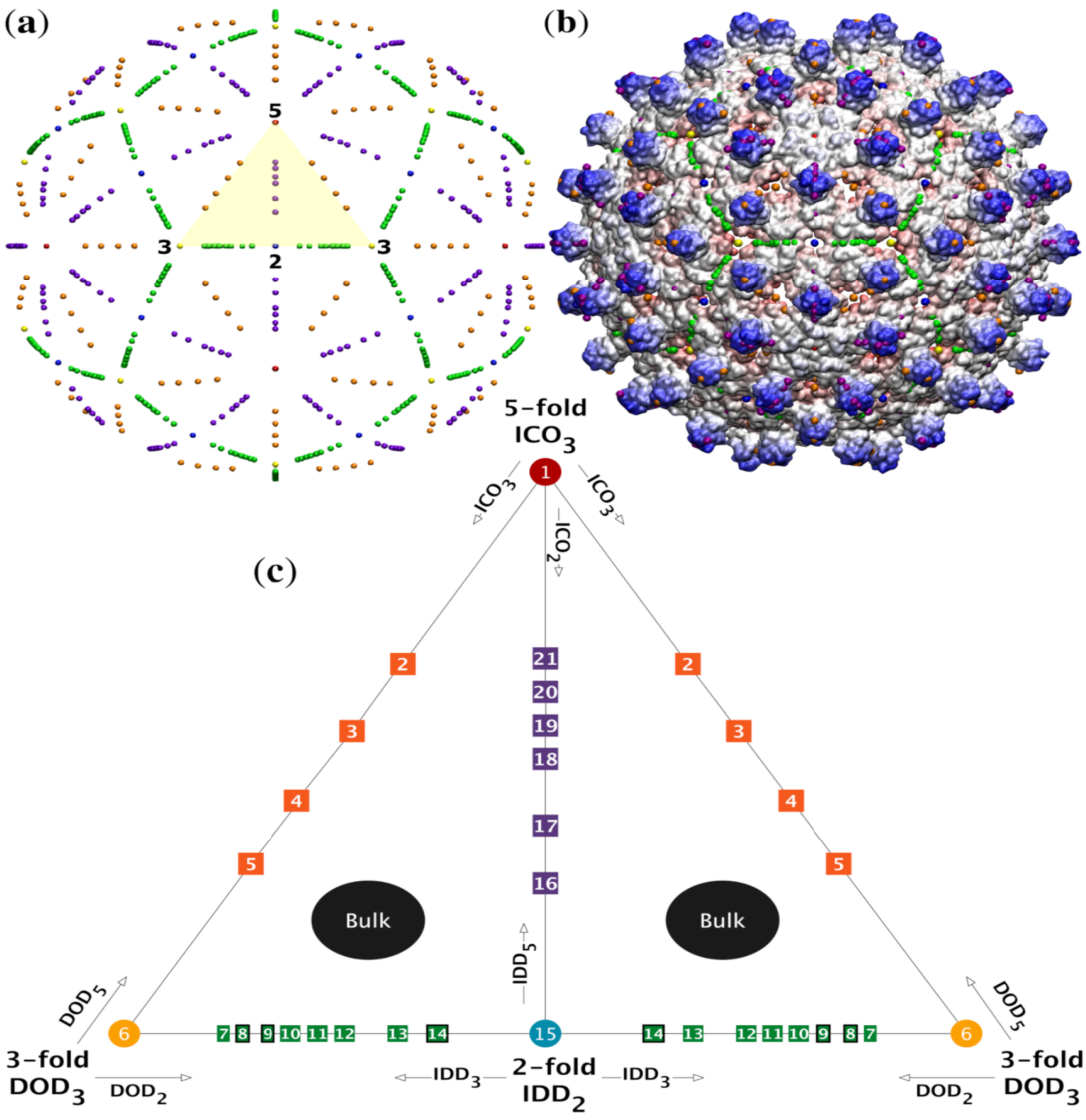



Figure 4. Gauge point numbering, associated with points on the 2-, 3-, and 5-fold axes of symmetry on the AU of a viral capsid (Wilson 2020).

The numbering scheme used is illustrated in Figure 4, showing the gauge point number associated with spike protrusions at each point along the AU's axes of symmetry. We hypothesize that certain gauge point groups are associated with other important characteristics of viruses such as protrusion amino acid composition, capsid size, and triangulation number.

# 2 Project Overview

The main research goal was to investigate the relationship between gauge points and other capsid properties. Data was collected from online databases: ViperDB (VIrus Particle ExploreR database), RCSB (Research Collaboratory for Structural Bioinformatics), and SCOP (Structural Classification of Proteins). I wrote code to streamline and automate this process, analyze capsid data, and compile a database, then I made a website to visualize the results.

Dr. Wilson already had a MatLab script called *SC_frankencode.m* (which I will refer to as *GP.m*) to find gauge points given a capsid's PDB (Protein Data Bank) coordinates from ViperDB, so I wrote scripts to perform other analysis on PDB and point array files. The first thing I worked on was developing a script called *find_aas.py* to detect which amino acids were nearby (<5Å) each point in a given point array and write the results to an Excel file. I then made a shell script to loop over a folder of point arrays and later used pyinstaller to create an executable of the Python script, as well as a corresponding adapted PA-folder script.

Next I created a pipeline that downloaded all of the capsid coordinate files from ViperDB and performed all the steps involved in collecting the gauge point and amino acid data. This consisted of: creating a list of all PDB IDs using the API, downloading each of their coordinate files, running makeicos.pl to create the full capsid of the AU in the coordinate file, extract_coords.pl to get the XYZ coordinates, pdb_indo.pl to load the capsid, my version of GP.m which writes the output to an Excel file (along with the points of the 5 closest Point Arrays to protrusions), and find_aas.py on each point array. This produced ~1,200 Excel files (available here - full_*.xlsx files contain amino acid data, others contain gauge point data).

I made an iPython notebook file called xlfiles_json.ipynb to compile a database with the results, additional data from scraping SCOP, RCSB, and ViperDB, and the output of some other analysis scripts (detailed in the data section below). The last stage of my work has been focused on making a website where you can search, filter, and create visualizations for information stored in the database. Data visualization was done with D3.js (Data-Driven Documents). All of the scripts and data used for the website can be found here.

# 3 Data Collection

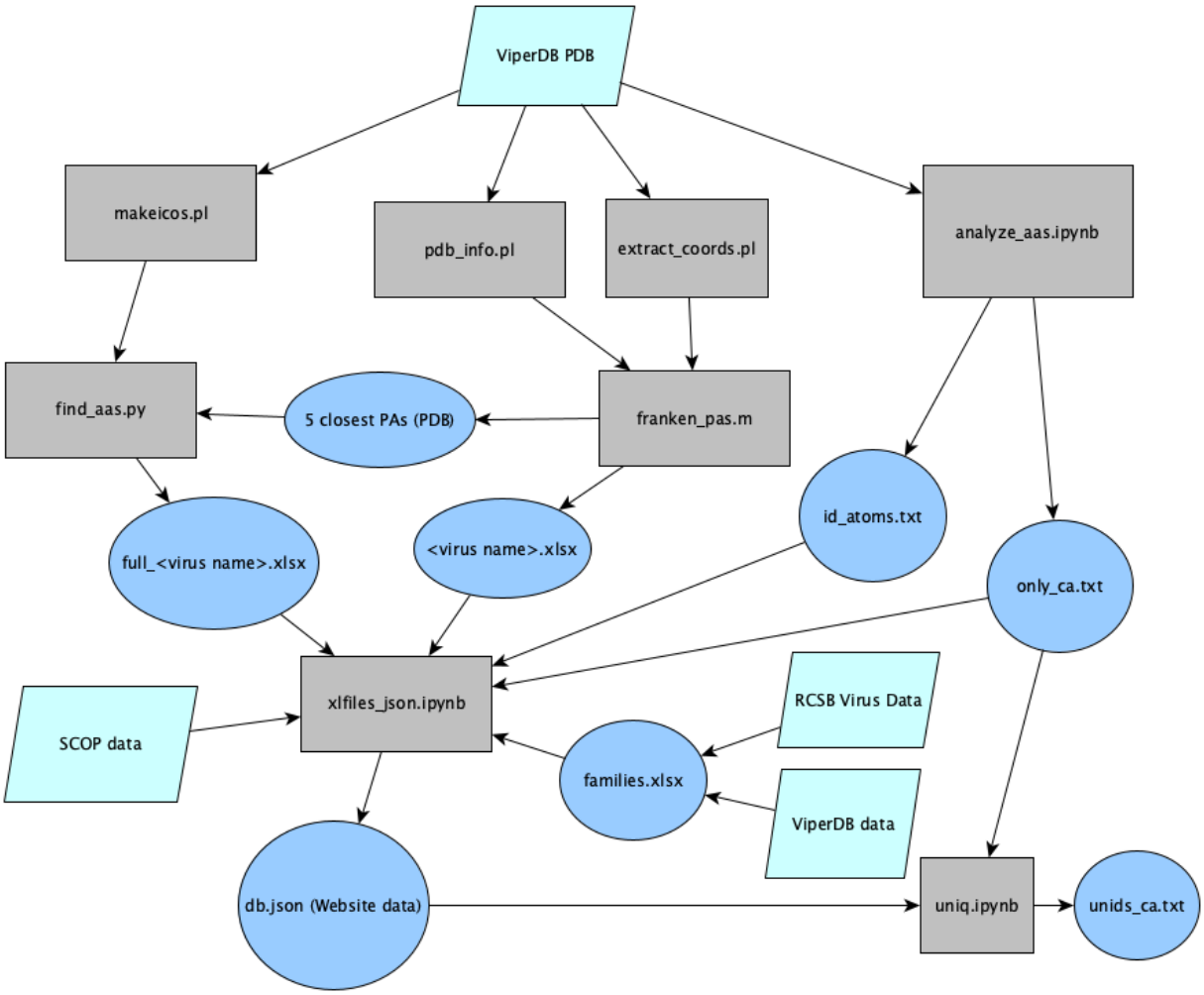



Figure 5. Data collection and processing steps going from the ViperDB, SCOP, and RCSB data to the website database file (and other outputs).

## 3.1 Finding Structurally Relevant Amino Acids

The purpose of this script (*find_aas.py*) is to take in two input files: one with the x, y, and z coordinates of the full capsid of a virus (the result of *makeicos.pl* taking the ViperDB PDB coordinates and tiling it 60 times to form the full capsid) , and the other with coordinates of the points of a relevant point array (PA), and compile a list of the closes amino acid residue to each PA point for each protein chain in the capsid. In order to accomplish this, I first parsed the coordinates into a dictionary mapping each chain to the coordinates of each constituent atom. I also parsed the coordinates of the PA file

into a list of [x,y,z] point coordinates. Next, I looped through each chain and used the function scipy.spatial.distance.cdist (SciPy is a Python library for scientific computing) to compute the distances from each member point for the chain to each point in the point array. I stored the minimum distance, along with the corresponding residue and atom, and looped through the sorted distances to find one where the distance was less than 5 and the residue was not the same as the closest one, storing that residue if it existed or N/A otherwise. Finally, I wrote the results to an xlsx file, with a row for each PA point and 5 columns for each PA chain.

Because I made use of SciPy, pandas, NumPy, and openpyxl (all Python libraries which must be installed by the user), I used pyinstaller to convert the script into an executable file find_aas bundled with all the dependencies mentioned. However, because the executable is slightly slower than the Python script given the bundled dependencies, I ended up using the Python script in the automated pipeline discussed below.

Most viral capsids have multiple PAs of interest, so I wrote a shell script called run_pas to run find_aas with a capsid for every point array file in a designated folder, and adapted the find_aas.py script so that if the Excel file being written to (named after the capsid) already exists, then the results would be written to a new sheet so that the result of the script being run on a folder is an Excel file with sheets named after each PA and storing their respective results.

### 3.2 Automated Pipeline

I wrote a series of scripts that downloaded, extracted, and collected amino acid and gauge point data for all spherical viral capsids on ViperDB. First was a script that collects all the PDB IDs through ViperDB's web api and downloads the file with their coordinates and AA residues (dl_ids.py). To do this, I wrote a small module named req_util.py, containing variables and functions to form requests by T-Number, genus, or family and used this to compile a text document with each PDB ID. Next, I used a Python library called Selenium to automate the process of downloading the coordinate files: opening the URL associated with a given PDB ID and clicking on the right download button. I ended up having to add a delay of 1 second because the download didn’t go through if I immediately had it navigate away from the page, but this was not a major concern as the time spent downloading is marginal compared to the time required

for the analysis scripts to run on each capsid. Finally, this script calls pdb.sh, which unzips the downloaded file, changes the ending to pdb (from ViperDB) and runs makeicos.pl and extract_coords.pl to compile the full capsid coordinate file (ViperDB only gives you the coordinates for the AU) used later in find_aas.py and the XYZ file used in GP.m.

Next was using pas_from_ids.sh to actually run the analysis scripts - 1) GP.m which writes the results to an Excel file named after the virus it is called on, and writes the point coordinates of the 5 closest PAs to pdb files and 2) find_aas.py, which determines the distance between capsid amino acids and point array points. Before I could do this, I had to write and execute a MatLab script which loaded the necessary virus info - output from running pdb_info.pl on the capsid coordinate file. Next, I ran GP.m by using the MatLab CLI (command-line interface), and finished by calling group_pas.sh, which executes find_aas.py on each point array recorded for the capsid and moves the resulting Excel files into a directory called xlfiles. This process is repeated for each line in the input file (containing the PDB IDs) so that all viruses on ViperDB could be analyzed by using a file containing all the IDs of viruses downloaded using the process described above.

### 3.3 Other Generated Data

The notebook analyze_aas.ipynb was made to create id_atoms.txt by counting the number of lines in each ViperDB PDB file, which (multiplied by 60) is the atom field used in the website (id_atoms.txt), and create only_ca.txt with a list of capsids which only have C alpha residues (~80), which was used in the creation of the unique ID list described below.

The notebook uniq.ipynb was made to compile a list of capsids with unique combinations of fields T-Number, genome, family, and genus. This is to try to reduce the effect of the bias introduced by the scope of virus research. Scientists tend to study viruses that impact humans, and viruses that are easy to study (easy to keep alive and grow in lab conditions, among other things). Among viruses with overlapping fields, the ones with the lowest resolution were chosen. By default, all viruses with only C alpha residues are removed. The output is written to unids_ca.txt and uniq_lres_ca_removed.xlsx

To allow for customizable analysis of amino acid data with specific point arrays, fields, and PDB IDs, custom_pa.ipynb was created. It is set up by default to work with the IDs produced from uniq.ipynb (detailed above) stored in unids.txt. It allows you to select whether you want to only consider the closest point in the top row of the closest PA (as in the main dataset), or for example the closest point in the top row of the top 5 closest PAs - getting the closest AA for each. Additionally, you can choose a field - say, T-Number - and have it create a separate Excel file for each possible value of that field. Other data like the total counts of each AA in the full capsid are visualized with pie charts.

I used count_aas.py to create a json file (aas.json) with the counts of each AA in the full capsid for each virus, along with some accessory amino acid results which are included in a separate endpoint in Firebase.

### 3.4 Web-Scraped Data

The main pieces of information that I decided to collect for each capsid consisted of properties listed on ViperDB (average_radius, family, etc.), RCSB (weight, polymer counts, etc.), and fold data from SCOP.

For ViperDB and RCSB information, I used the respective web APIs to get any data that seemed relevant, using functions from req_util.py in iPython notebooks (family_stats.ipynb and vdb_parser.ipynb, which are no longer working due to API changes). For the SCOP data, I mainly worked from the downloadable file from SCOP's download page containing all of the classifications for each capsid. From Nasir and Caetano-Anollés, 2017, Dr.Wilson and I compiled a list of the translation from the fold names used in SCOP to the names used in Nasir & Caetano-Anollés, 2017, and Krupovic & Koonin, 2017. This was used to compile an Excel file I called families.xlsx with sheets named after each family with viruses on ViperDB, containing the PDB and corresponding data for each virus in the family. Some corrections were made where values weren’t sensible (such as decimal values for genome).

### 3.5 Creating the Database

I made xlfiles_json.ipynb to create db.json, the file used as the website's database. This file takes inputs from much of the data collected so far: the xlfiles directory, families.xlsx, SCOP downloadable files, id_atoms.txt, and aas.json. The output from find_aas.py is used to create the amino acid fields on the website (described in the next section), and output from franken_pas.m is used to get gauge points. I also compile some other information: all the fields (families, genuses, etc.) which at least one virus in the dataset had - this serves as the list of filters which are shown on the website - and a list of discrete (tnumber, gauge_point, etc.) and continuous metrics (average radius, weight) to be used for the options for the axes for different kinds of graphs. That, along with the data coming from other input files, is stored in a JSON file called db.json, which contains all the API endpoints for the website.

# 4 Website Usage

Figure 6. QR code for the website (https://pa-project-66d90.web.app/). Scan to try it out!

I used Firebase, a Google Cloud service, for my database and website hosting because I have some experience with it and it gives you an easy interface to design a web application for many users for free. The main features of the website are: a search feature that allows you to query the information stored for any specific PDB ID, scatter plots, bar graphs, pie charts, and heatmaps. Because of the way I stored all data as singular fields for each virus, and there are only ~1200 spherical capsids on ViperDB, I decided to load all viruses and all associated data into the web app as soon as it opens so that any filters applied or axis change would not require a new query.

## 4.1 IDs

The 'ids included' option above the graph buttons allows you to limit the dataset to certain PDB_IDs. By default, all ids are included - the full dataset. 'unique' refers to a specific list of 187 capsids with unique combinations of fields T-Number, genome, family, and genus (prioritizing the capsids with lowest resolution, and excluding those with C alpha only residues in their PDB files) which can be found here. Additionally the 'custom' option lets you upload your own newline-delimited text file with PDB IDs to include (all others will be filtered out).

## 4.2 Search

Clicking on the magnifying glass in the top left lets you search for the data of a specific virus capsid by its PDB id. Simply type in the id (make sure it is in the ids that

make up the dataset - see above) and you will get each of the fields and values, separated by a colon, from the capsid object.

## 4.3 Filters

After creating a graph, a filter bar along the top will be created. There is a filter dropdown checklist for each property of viral capsids in the database. By default, all values are included (even missing), but if you check a certain value then only viruses with that value will be graphed; checking multiple values will let viruses with any of the checked values be included. This means clicking select all will exclude the capsids which have that field missing.

Also on the filter bar you will find the download button, which generates a semicolon & newline-delimited (semicolon for separating coordinates, newline for points) text file with the data used to generate the graph (varies by the graph, but for example the scatter plot gives you 'x-coordinate;y-coordinate;value;id' on each line corresponding to one point).

## 4.4 Fields

What follows is a description of all of the data fields available on the website, each starting with a bullet point. They are grouped into three categories: ViperDB data, RCSB data, SCOP fold data, and generated data

### ViperDB

- average_radius (Å): The 'Ave' Radius value on Viper for the capsid
- tnumber: The triangulation number (T-Number) can be thought of as describing how many proteins form the asymmetric unit of a capsid
- resolution (Å): Resolution of the shell coordinate reconstruction
- family: Family of the virus
- genus: Genus of the virus

- genome: Genome of the virus (for uniformity, viruses were grouped into either ssRNA, ssDNA, dsRNA, dsDNA, or NA meaning other)
- atoms: Number of lines in the PDB coordinate file multiplied by 60 to account for the full capsid.

## RCSB

- weight (kDa): Total Structure Weight - molecular weight in all non-hydrogen atoms. Hydrogen atoms are included for the charged state in ARG, HIS & LYS residues.
- deposited_polymer_monomer_count: Deposited Residue Count - Number of all polymer monomer residues
- polymer_molecular_weight_maximum: Maximum molecular weight of polymers
- polymer_molecular_weight_minimum: Minimum molecular weight of polymers

## SCOP

- fold: SCOP download files were used to map viruses to their folds described in Nasir & Caetano-Anollés, 2017. This was done by finding the string corresponding to the folds in Nasir, finding all the SCOP IDS which map to each string, then getting PDB IDS of viruses containing those SCOP IDS.

## Generated data

- most_common_aa: The most frequent AA in the full capsid (generated using makeicos.pl
- gauge_point: The gauge point of the capsid
- closest_aa: The AA with lowest distance from any point in any of the top 5 PAs (point arrays)
- other_aa: Another AA which was within 5 Angstroms of closest_aa (NA if none)
- closest_gp_aa: The AA with lowest distance from the gauge point in the closest PA

- other_gp_aa: Another AA which was within 5 Angstroms of closest_gp_aa (NA if none)
- common_gauge_aa: The most common AA from those with distance less than 5 (even if not closest - including other_aa) from any point from the top PA

### 4.5 Graphs

**Fields that can be viewed/graphed on the website**

| Discrete | Continuous |
|---|---|
| fold | weight |
| tnumber | atoms |
| genome | deposited_polymer_monomer_counts |
| family | polymer_molecular_weight_maximum |
| most_common_aa | polymer_molecular_weight_minimum |
| closest_gp_aa | average_radius |
| other_gp_aa | resolution |
| common_gauge_aa | (count) (for non-scatter) |
| closest_aa | |
| other_aa | |
| genus | |
| gauge_point | |

**Graphs that can be made on the website**

| | |
|---|---|
| Scatter | Continuous vs Continuous |
| Bar | Continuous vs Discrete |
| Pie | Discrete |
| Heatmap | Discrete vs Discrete (shading by continuous) |

Figure 6. Graphing options on the website. The variables are grouped as either discrete or continuous, and are shown as options for scatter, bar, pie, and heatmap graph axes accordingly.

There are 4 graph options: scatter plot, bar graph, pie chart, and heatmap. When a button is clicked to create a graph, random parameters are chosen and the graph is constructed. For each parameter - for example, the x-axis metric - a dropdown is shown with the current value selected. There is a download button which allows the user to download a txt file with the data used to construct the graph and an info button which toggles a text block describing the graph shown and how to adjust it.

**Scatter Plot**

The function which creates a scatter plot takes in two continuous metrics to plot, a “primary” discrete metric to color data points by, and an object containing the selected filters. The object is created by calling loadFilters, a function which returns true for each checked class in each category. For every key in the filter object (corresponding to categories where at least one class was checked), every virus which does not have the category registered or whose class for that category is not in the list of checked classes is filtered out. The graph element is created with the filtered data, titled with the x and y-axis metrics and the number of data points, and with dropdowns initialized with the given metrics on each axis. When a user clicks on the scatter plot button initially, I decided to choose randomly from the list of continuous variables, loaded from the properties endpoint in Firebase. I also choose a random field to color the data by (ex. T1 capsids have blue dots, 3 have red, etc.), with viruses that don’t contain the field being filtered by shown as black unless the user used the checkbox dropdown to filter for certain values. A legend is created for the colors that represent each value of the chosen field.

**Bar Graph**

The main difference between the bar graph and scatter plot is that the x-axis is a discrete metric, and the y-axis can be either count or the average of a continuous metric. The x-axis dropdown thus contains all discrete metrics in the database, and whichever one is selected then looks to the filter dropdown checkboxes to determine which classes in that category to include (all are included if nothing is checked).

**Pie Chart**

The function to create a pie chart takes in a single category and plots the count of each class within that family. Because of space concerns, the slices on the pie chart are not labeled but the downloadable data contains the class label and value for each slice.

**Heatmap**

To create the heatmap, a list of classes has to be supplied for both the x and y axes; this is done the same way as the bar chart where it is random at first and then the item selected in the axis dropdown turns the filters for that category into the classes used in the heatmap. Then, the average of the selected continuous variable is calculated and used to determine the degree of shading for each box corresponding to the intersection of a class for each axis.

# 5 Results & Discussion

This section begins by surveying some fields of interest across all capsids, then we look at frequency, average atoms and average_radius for 1) closest_gp_aa, 2) gauge points, 3) genome 4) T-Number using bar charts, and finally use heatmaps to investigate potential relationships between these fields. The pie and scatter charts include all of the ~1200 capsids in the database and the other sections compare data between both capsid sets.

## 5.1 General Results

I used pie charts to visualize the frequency of values in the following fields: closest_gp_aa, family, fold, gauge_point, genome, and tnumber.

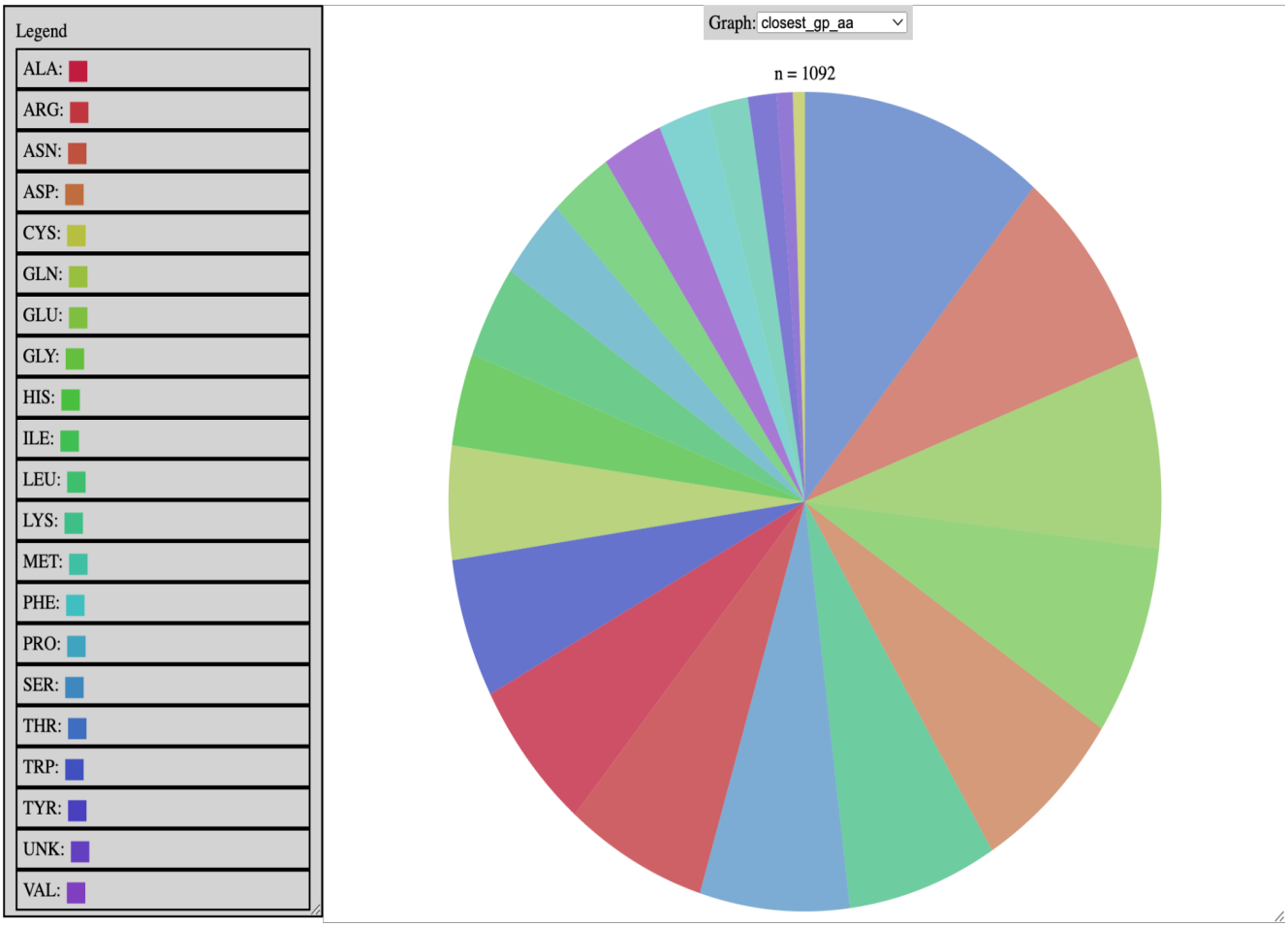



Figure 7. The amino acid closest to the gauge point for all capsids. The most common closest_gp_aa values in the full datasets are threonine (121), asparagine (89), and glutamic acid (83).

The most common closest_gp_aa in the full dataset is threonine, followed by asparagine and glutamic acid; there are many amino acids which are around the 70-90 capsid mark (about 6% of the capsids with GP data). Interestingly, two of the top three amino acids (threonine and asparagine) have polar uncharged side chains. This could be because of the role of spike proteins of interaction with the polar heads in target cell membranes.

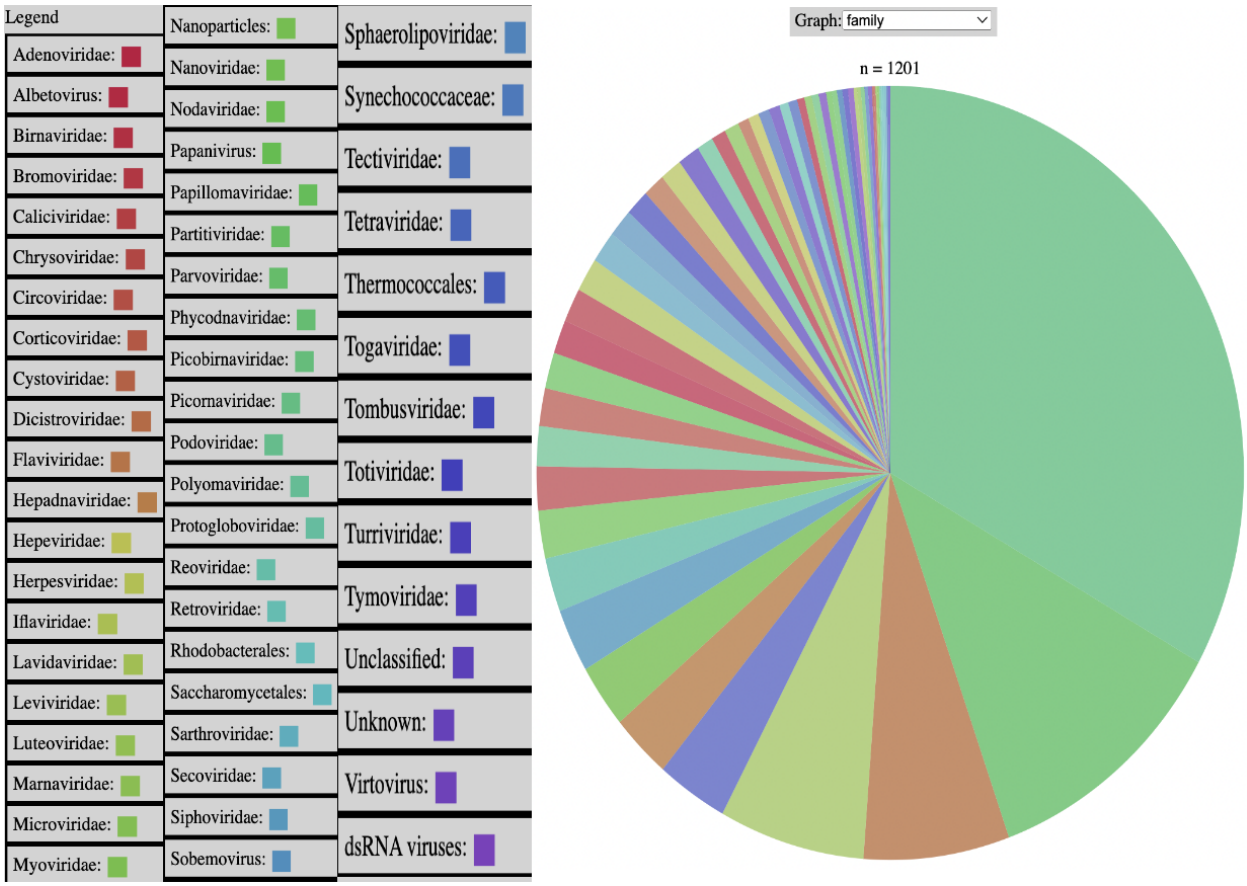



Figure 8. Virus families from ViperDB. The majority of spherical virus capsids listed are either Picornaviridae (398), Parvoviridae (137), or Flaviviridae (80).

The Picornaviridae (398 capsids), Parvoviridae (137 capsids), and Flaviviridae (80) make up over half of the dataset alone, but there are 57 families represented in total. Each of these top three families have single stranded genomes, with Parvoviridae having ssDNA and the other two having positive ssRNA. This suggests a somewhat significant bias in the dataset; averages will usually tend towards characteristics of the dominant families, which is one of the motivations for the unique capsid set.

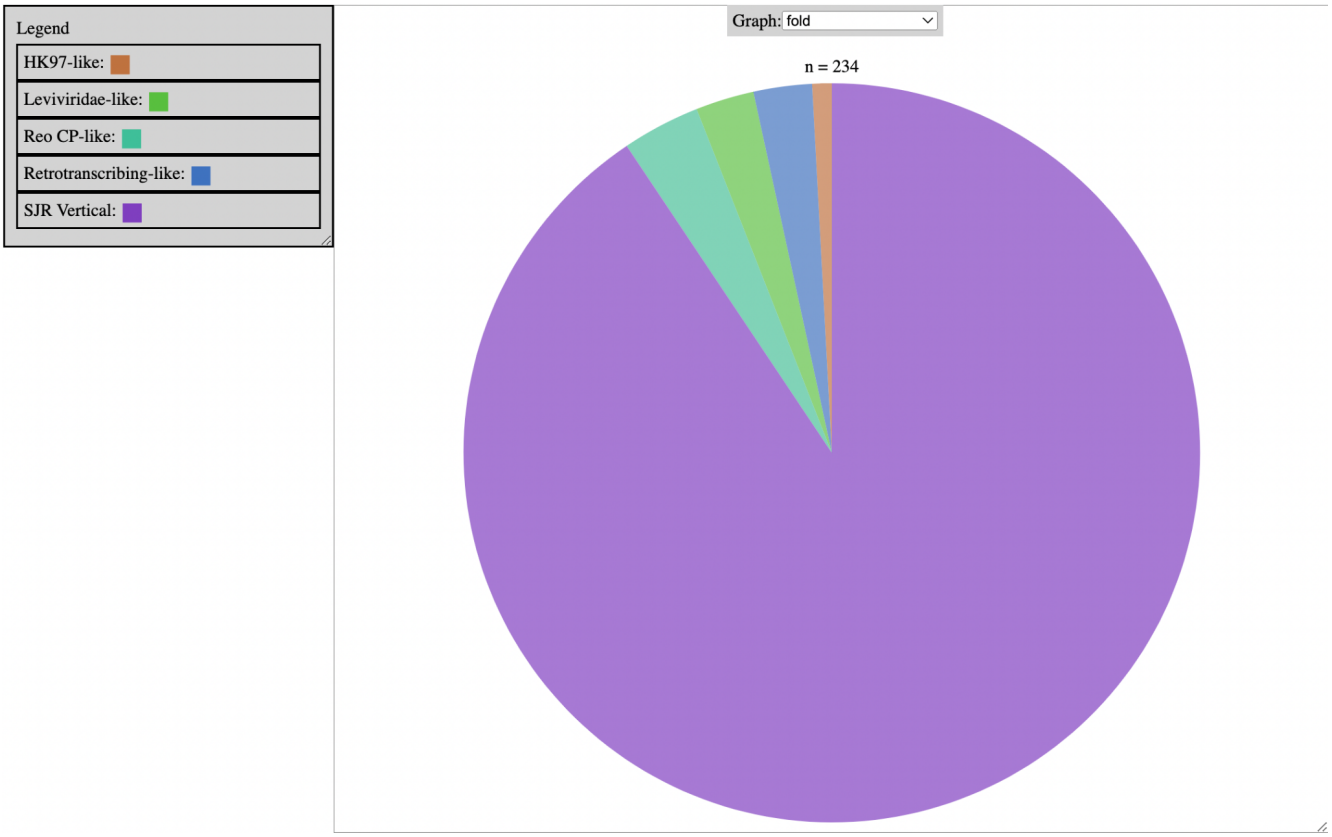



Figure 9. SCOP fold classification, if listed. SJR (Single Jelly Roll) was the dominant fold for capsids in the SCOP dataset.

By far the most common fold is SJR Vertical, covering 212 out of the 234 capsids with fold data in SCOP. There are 4 other fold categories found, each with less than 10 members. Unfortunately, very few of the capsids on ViperDB had folds listed in the SCOP downloadable files other than “Nucleoplasmin-like/VP (viral coat and capsid proteins)”, which translates to SJR Vertical (Nasir & Caetano-Anollés, 2017). 23 capsids which did not have SCOP IDs referenced by Nasir & Caetano-Anollés, but did have the associated SCOP translations - this is possible because SCOP uses the same string of text at multiple levels of classification - were added to try to get more data.

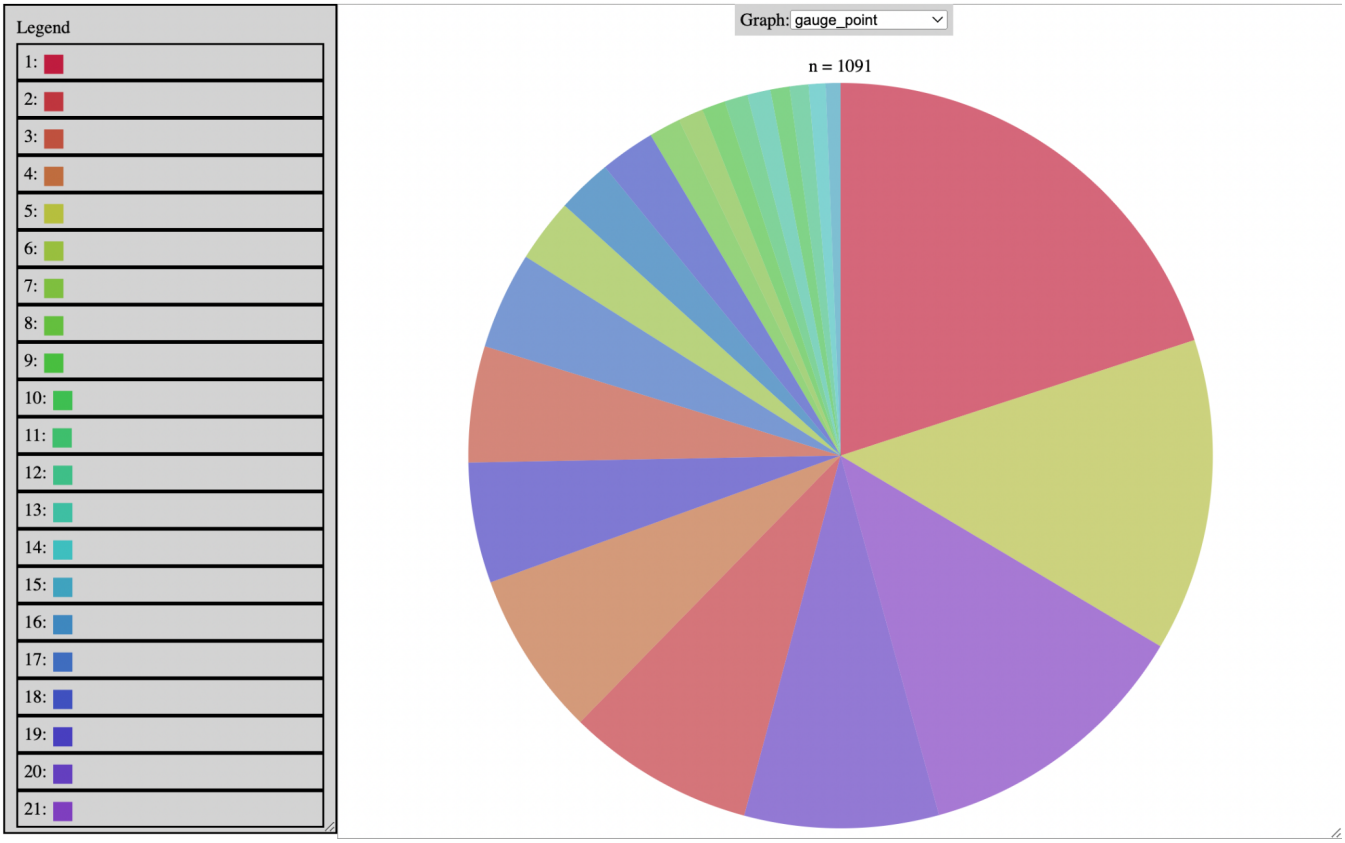



Figure 10. Gauge point frequency. Top gauge points were 1 (214 capsids), 5 (148 capsids), and 21 (133 capsids). This places most protrusions at the center of a pentagonal face (1), just below it (21), or at the lowest point along the left and right sides of the AU.

The most common gauge points were 1 (214 capsids), 5 (148 capsids), and 21 (133 capsids), making up around 45% of the capsids with GP data. All gauge points had at least one capsid, but the distribution is very skewed, with gauge points 1-6 and 16-21 making up over 90% of capsids. This corresponds to a scarcity of protrusions along the two-fold axis of symmetry. It is worth noting that most frequent gauge points are those which lead to the least number of protrusions across the AU. With GP 1, for example, since it is at the center of the pentagon there is only one protrusion per pentagon, whereas having a protrusion along the two-fold axis would mean 10 protrusions per pentagon.

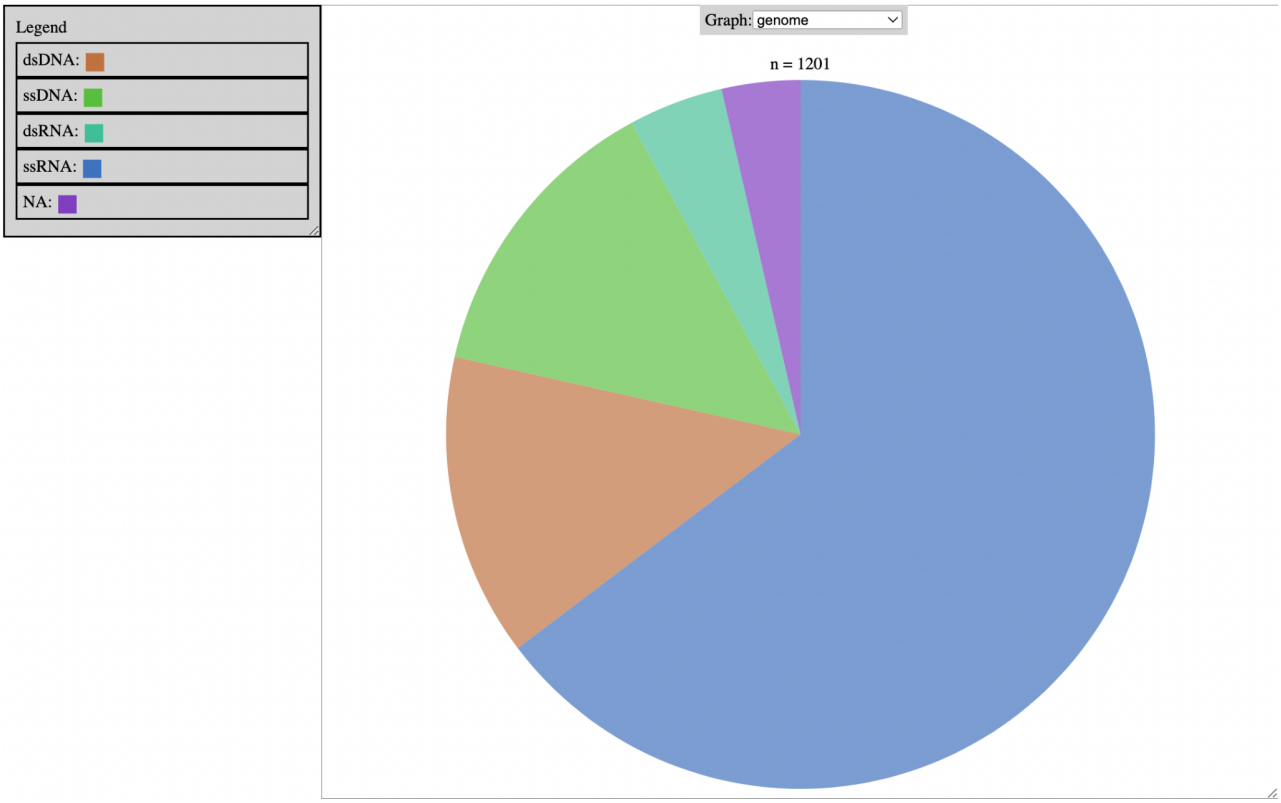



Figure 11. Virus genome type. Capsids without genome data on ViperDB, or with values that did not fit into the above classification, were set to ‘NA’.

Almost ⅔ of capsids in the dataset had ssRNA genomes, with only 52 dsRNA capsids. Note that ViperDB uses a wider range of labels; this list was compressed manually to the 5 categories displayed: dsDNA, ssDNA, dsRNA, ssRNA, NA (missing/other). Genome type is known to be related to the size of capsids, with double stranded (especially dsDNA) genome capsids being larger.

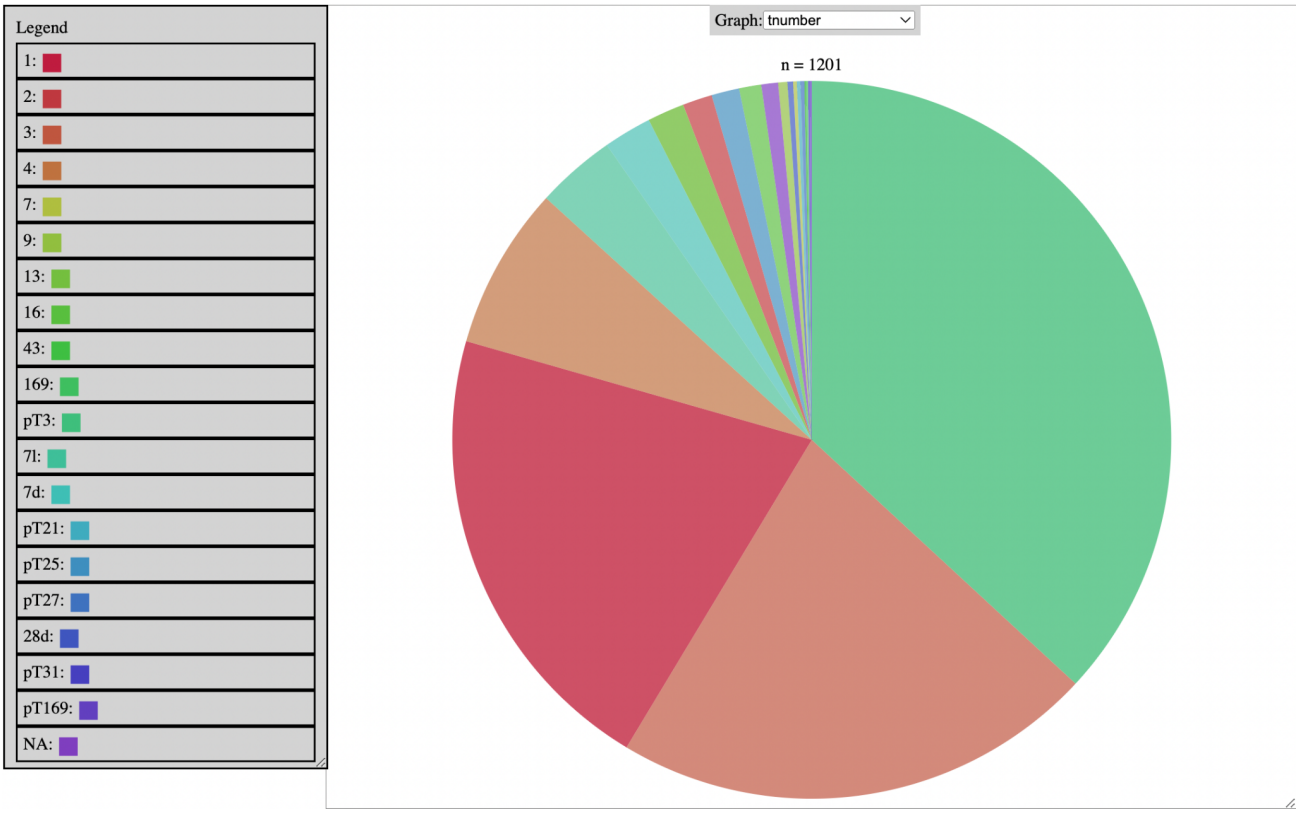



Figure 12. T-Number frequency. Most capsids had T-number pT3 (pseudo-T3), which mimic T3 capsid structures with three domains of a single protein.

More than ¾ of capsids had T-number pT3 (443), 3 (261), or 1 (250). This could reflect a bias in the dataset towards simpler viruses, which is part of the motivation for cultivating the unique id set used for the results below.

Of the continuous metrics which I collected data for, atoms and average_radius seemed to be the most reliable, as data from RCSB was found to vary more than we expected. Resolution is also an interesting field as it largely reflects the precision with which capsids have been observed, with lower resolution values indicating more precision. A natural question that arose was whether there is a relationship between the number of atoms and the resolution, or average radius.

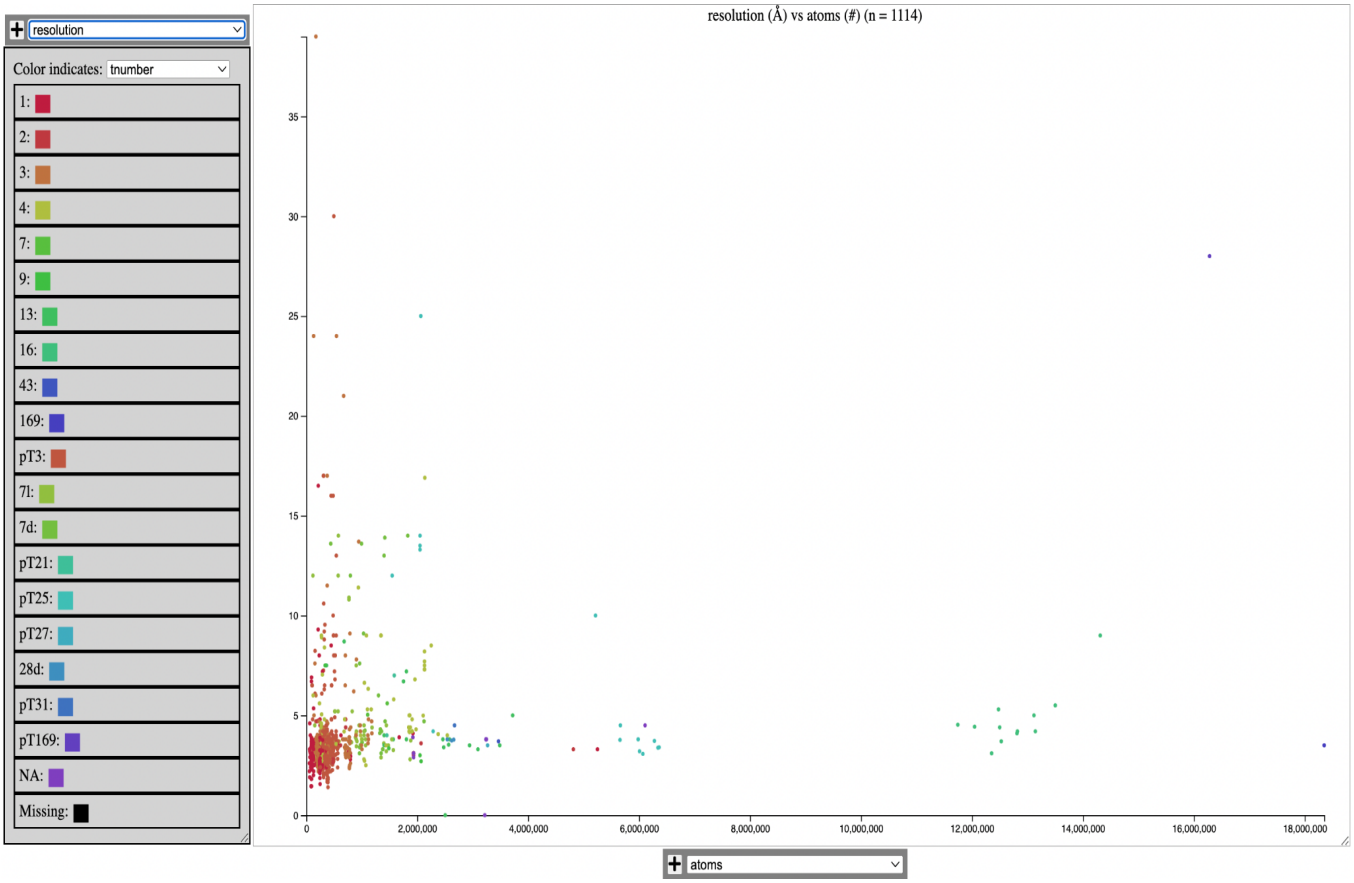



Figure 13. Resolution vs atoms, colored by T-Number. Resolution indicates how finely the structure can be resolved, so higher resolution provides less information. There does not appear to be a relationship between resolution and atoms.

There is not a significant relationship between atoms and resolution. Capsids with more atoms tend to have higher T-Numbers; the group of points slightly to the right of the center are T16 capsids. This tells us that there's at least not an obvious bias in our low-resolution unique set towards larger or smaller capsids.

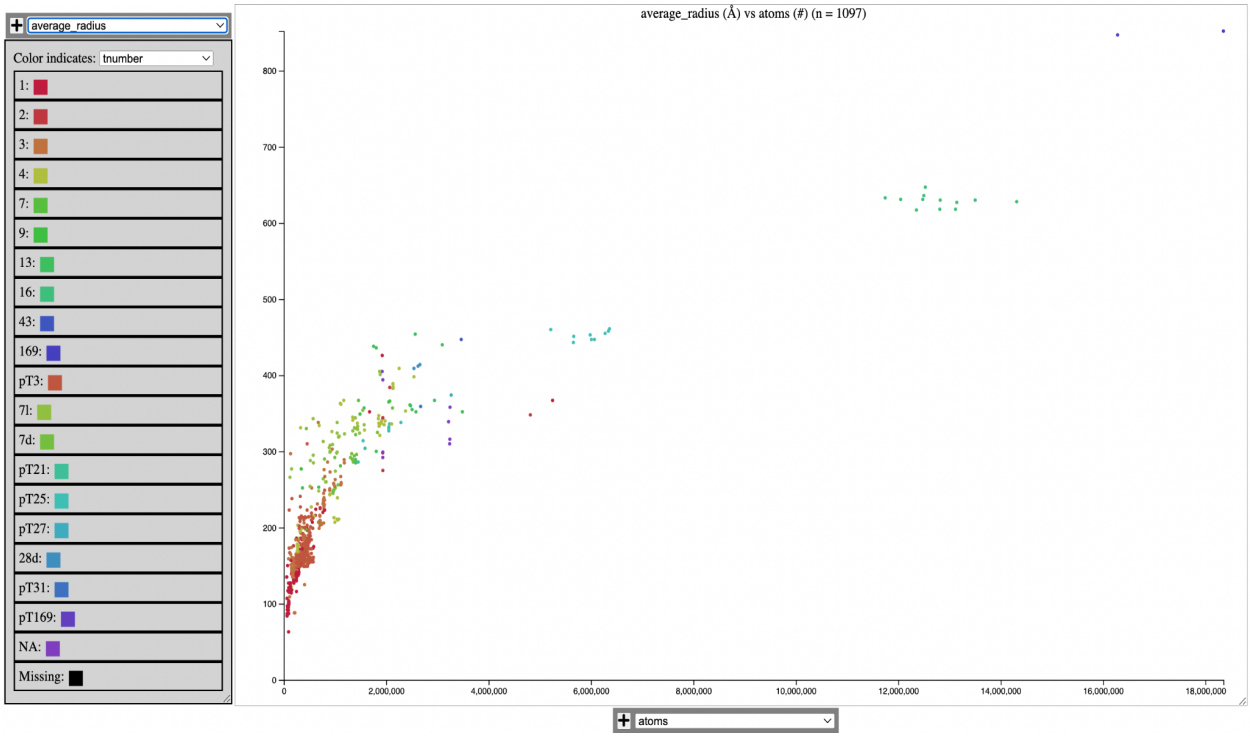



Figure 14. Average_radius vs atoms, colored by tnumber. It looks like average_radius scales with the second or third root of the number of atoms in the full capsid.

There seems to be some relationship between average radius and number of atoms in the full capsid. Presumably, this relationship would be something like a cube root as the volume of the capsid scales as the cube of its radius.

## 5.2 Closest Amino Acid

The primary field generated from find_aas.py data is closest_gp_aa: the closest amino acid (AA) to the gauge point (GP). This could relate to capsid formation, structure, and binding to the cell membrane of target cells.

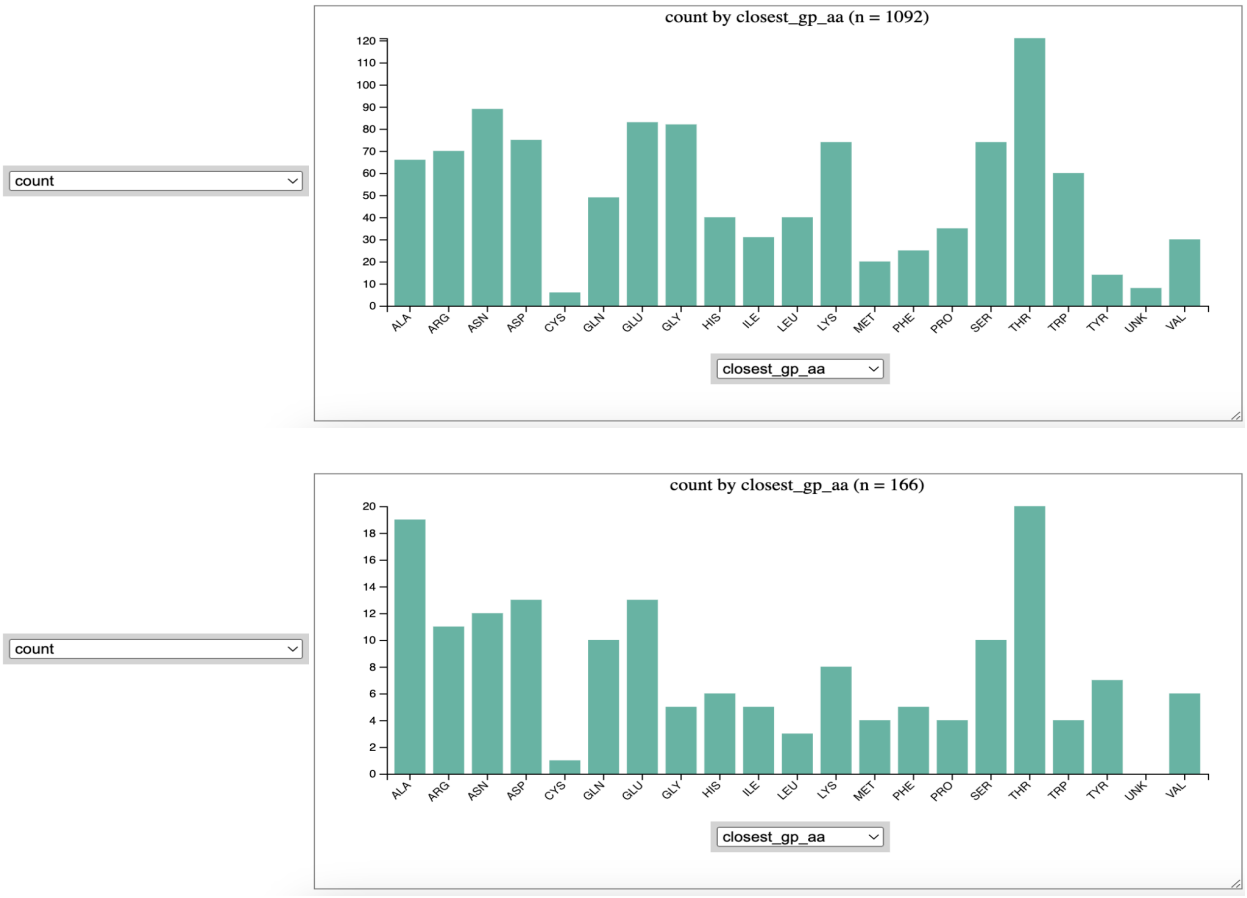



Figure 15. Count by closest_gp_aa in all vs unique capsids. Threonine is the most common, and cysteine the least.

In both sets, threonine (with a polar uncharged side chain) is the most common closest_gp_aa, but in the unique set alanine (having a hydrophobic side chain) is the second-most common, whereas in the full set asparagine (polar uncharged) and glutamic acid (negative) were the second and third most common GP AAs respectively. This variation in chemical properties suggests a structural reason for their location on the water-dominated exterior rather than an electrochemical one. Additionally, cysteine is notably uncommon even though it is a nucleophile along with glutamic acid, forming

bonds by donating electrons. In both sets, threonine makes up about ⅛ of the GP AAs - 121/1092 and 20/166 for the full and unique sets respectively.

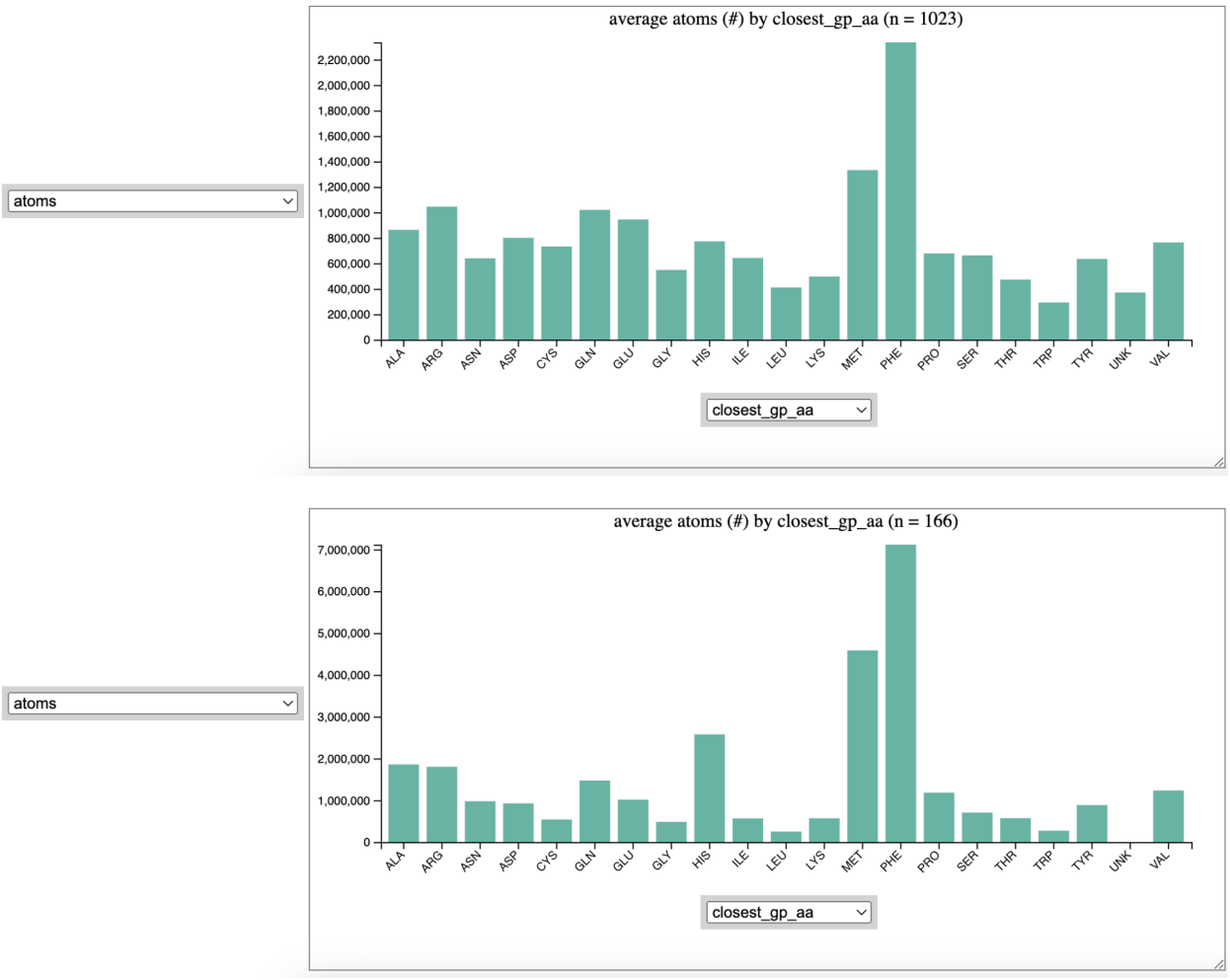



Figure 16. Average atoms by closest_gp_aa. There is a surprising amount of variance among the average number of atoms in each group; capsids with phenylalanine closest to the gauge point have much higher number of atoms on average than the others.

There seems to be a very high average number of atoms in capsids with phenylalanine and (less so) methionine as the AA closest to their GP. This is followed by histidine in the unique capsid set, but the only pattern that is really preserved in the full set is the higher number of atoms for phenylalanine and methionine. Phenylalanine has a hydrophobic side chain, and has been established as an amino acid that could be important in virus structure.

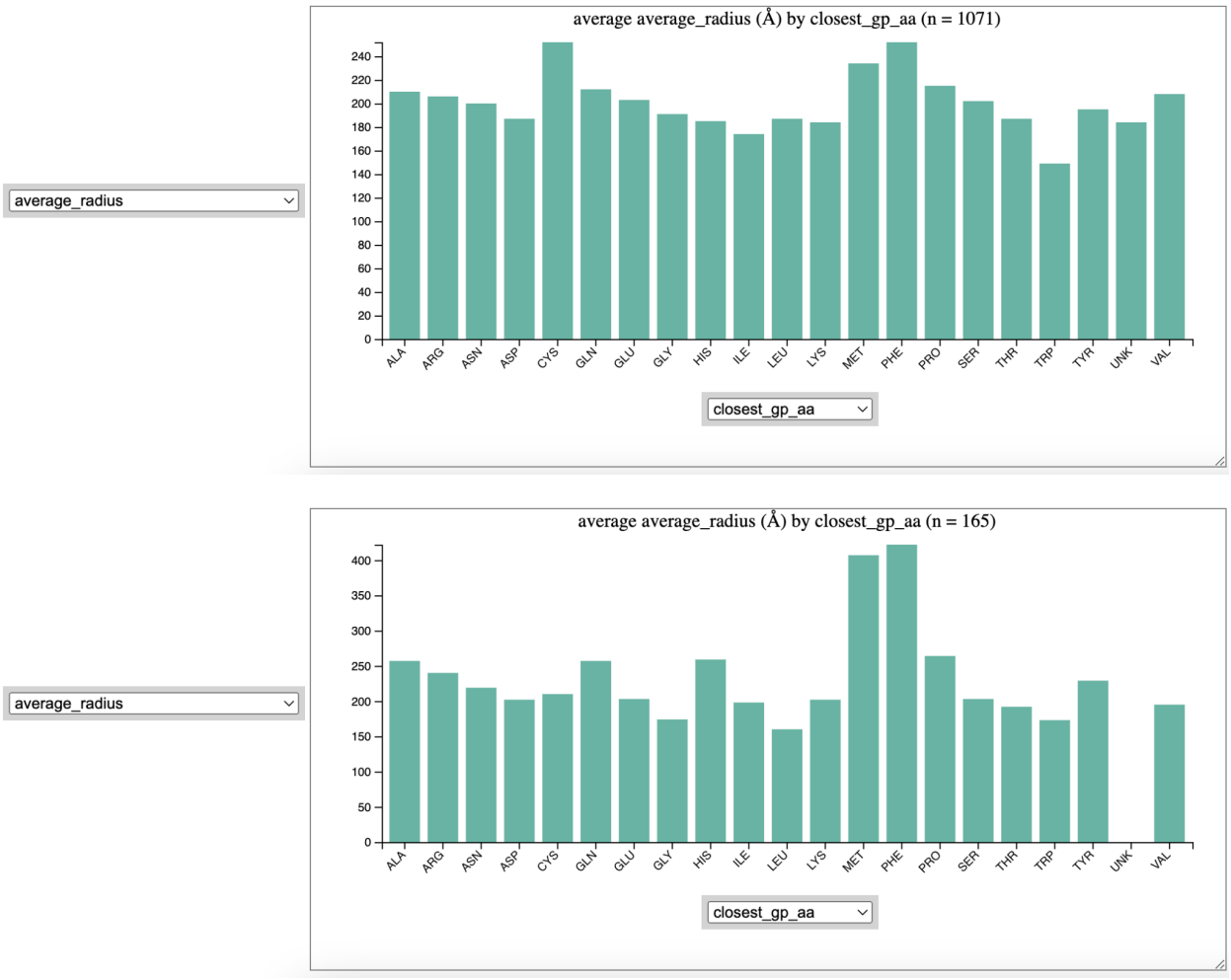



Figure 17. Average average_radius by closest_gp_aa. In the unique set (bottom), capsids where the closest AA to the gauge point is phenylalanine or methionine tend to have larger average_radius values.

Figure 17 shows phenylalanine and methionine have the largest average_radius values in the unique set, but not in the whole capsid set. More research could be done into the relationship between amino acids and capsid size to determine the significance of this relationship, as it may simply be a sample size problem - only 4 and 5 capsids bad closest_gp_aa methionine and phenylalanine, respectively.

## 5.3 Gauge Point

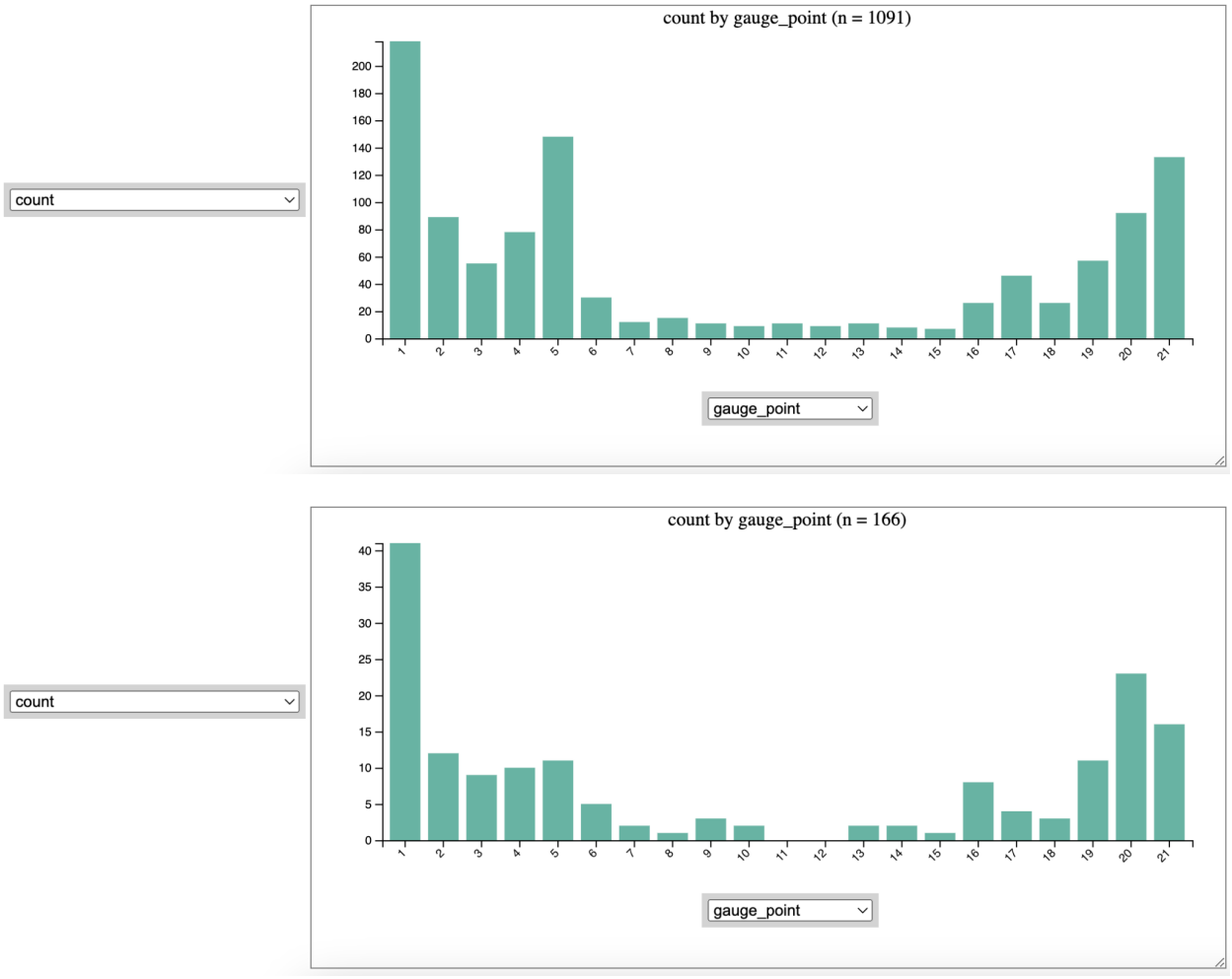



Figure 18. Count by gauge_point in (a) all capsids vs (b) unique capsids. Gauge point 1 is the most common, with clusters around 1-5 and 16-21.

As shown in Figure 18, gauge points were clustered around 1-5 and 16-21, with 1 being the most common in both sets. Gauge points show largely the same pattern when restricting to unique capsids as in the whole dataset. There are no capsids with gauge points 11 or 12 in the unique set.

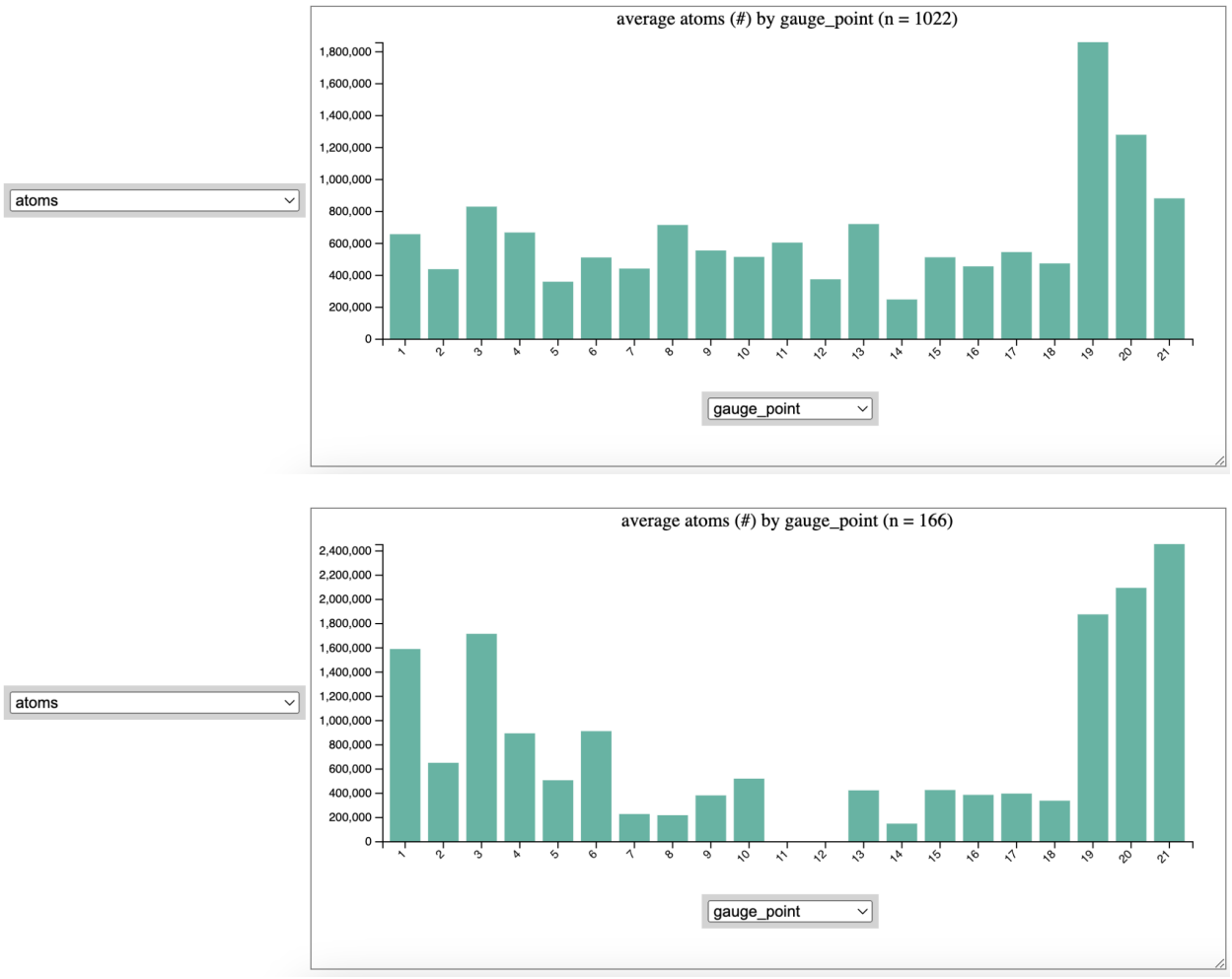



Figure 19. Average atoms in all vs unique capsids. There are more atoms, on average, in capsids with gauge points 19, 20, and 21.

In both capsid sets there seem to be many more atoms in capsids with GP 19, 20, and 21, followed by 1 and 3, although the GPs 1 and 3 stick out much less in the full capsid set. But in the unique set, these GP groups have at least 9 capsids in them, with 1 having 41, so this pattern could be indicative of a feature of gauge point 1 capsids that simply gets diluted when certain groups of gauge point 1 capsids get included in the full set. There are slight dips at gauge points 5 and 14 in both sets, and a somewhat notable peak at GP 19 in the full set.

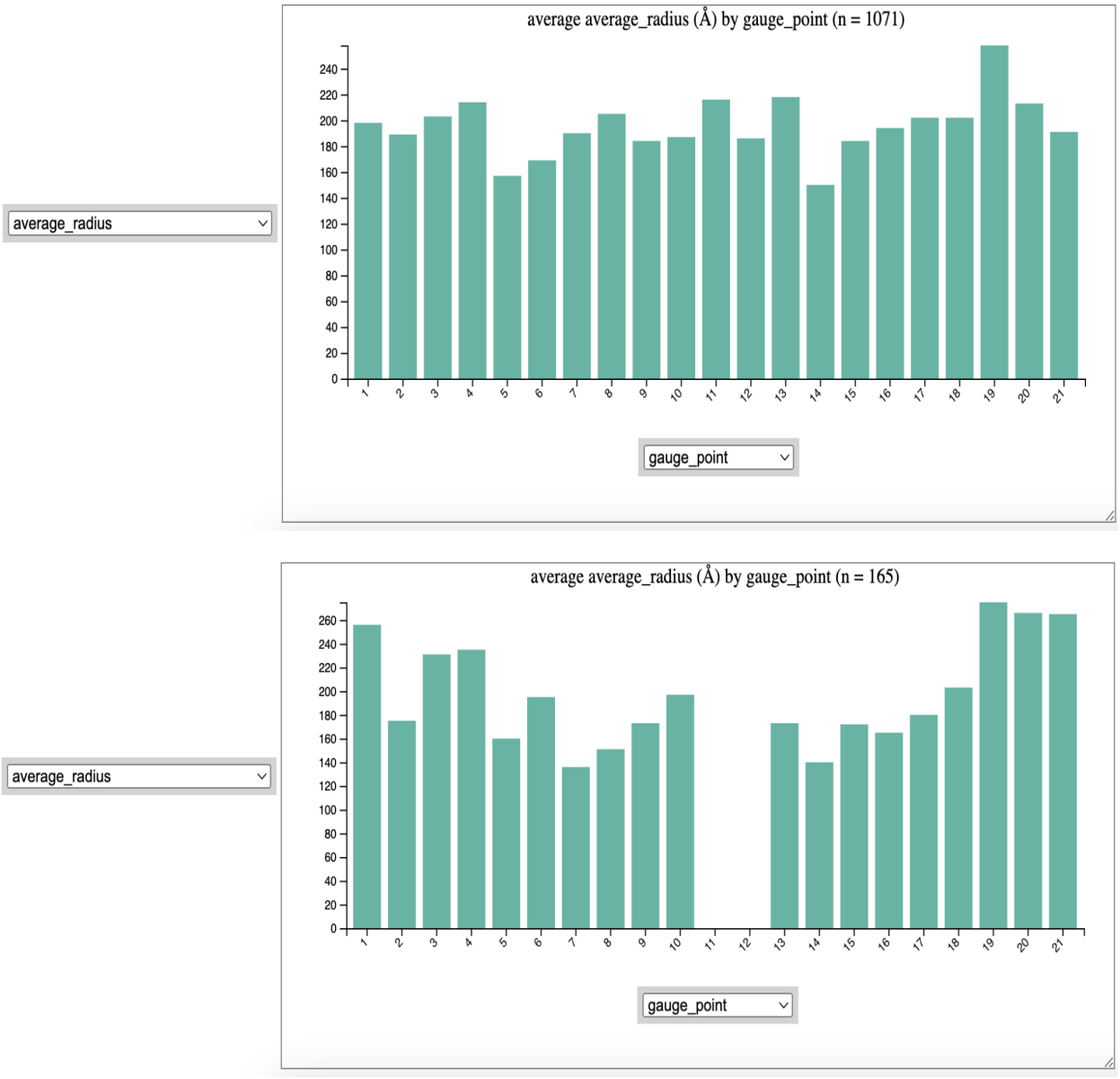



Figure 20. average_radius by gauge point for all vs unique capsids. Curiously, there is not as clear of a relationship between gauge point and average_radius as with atoms.

There does not seem to be as much of a relationship between gauge point and average radius as with atoms; it seems more balanced across the central GPs but still exhibits similar peaks on gauge points 1, 19, 20, and 21 in the unique set. However, there are still noteworthy dips at gauge points 5 and 14, especially in the full set.

## 5.4 Genome

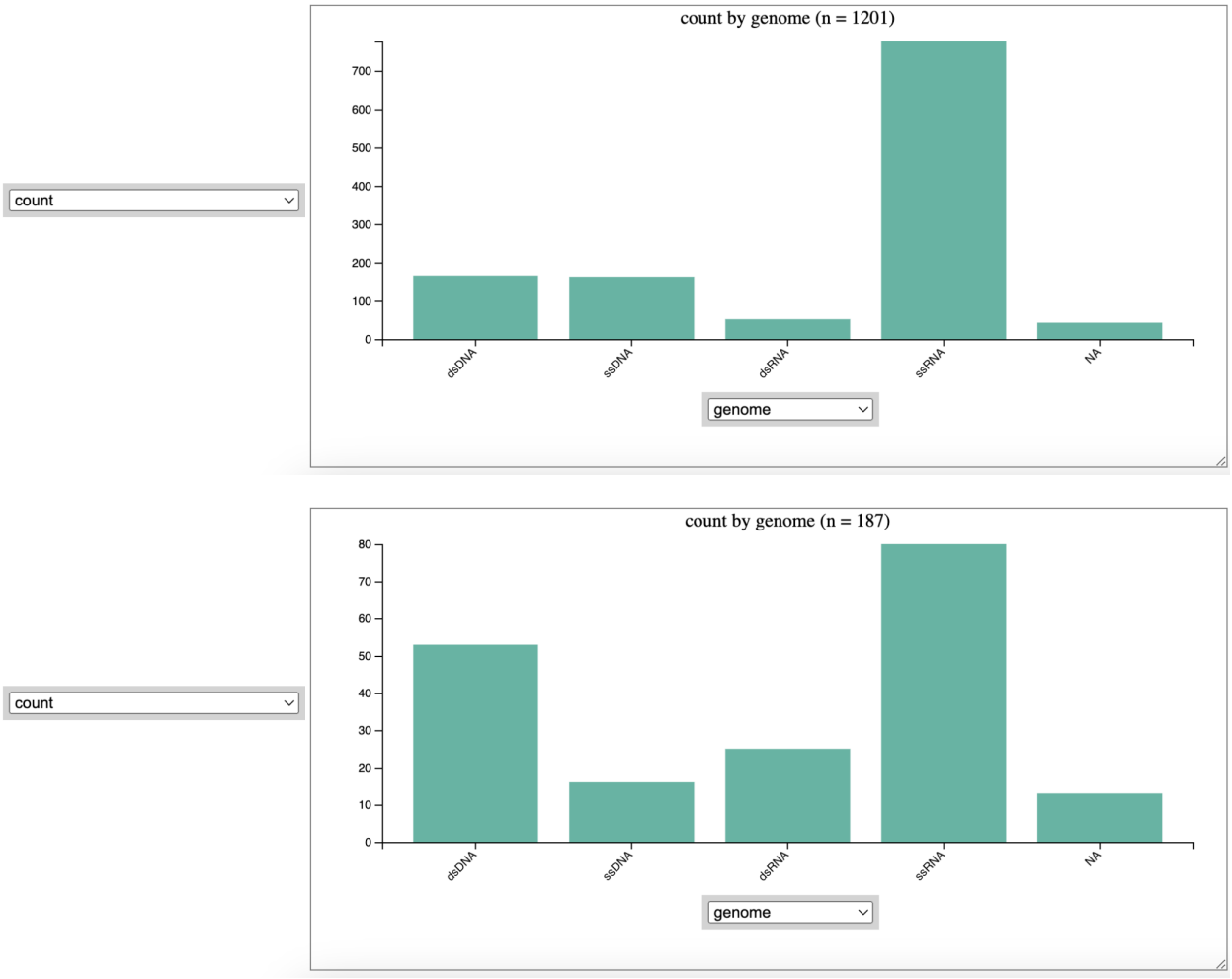



Figure 21. Count by genome. Capsids with genomes not listed, or with invalid values, were set to ‘NA’. ssRNA is by far the most common, followed by dsDNA capsids, which make up a much larger proportion of the unique set.

While there are similar numbers of dsDNA and ssDNA capsids in the whole dataset, there are significantly more dsDNA capsids in the unique set. ssRNA capsids make up the majority of all capsids. ssRNA viruses make up almost 43% of the unique set, followed closely by dsDNA with around 28%.

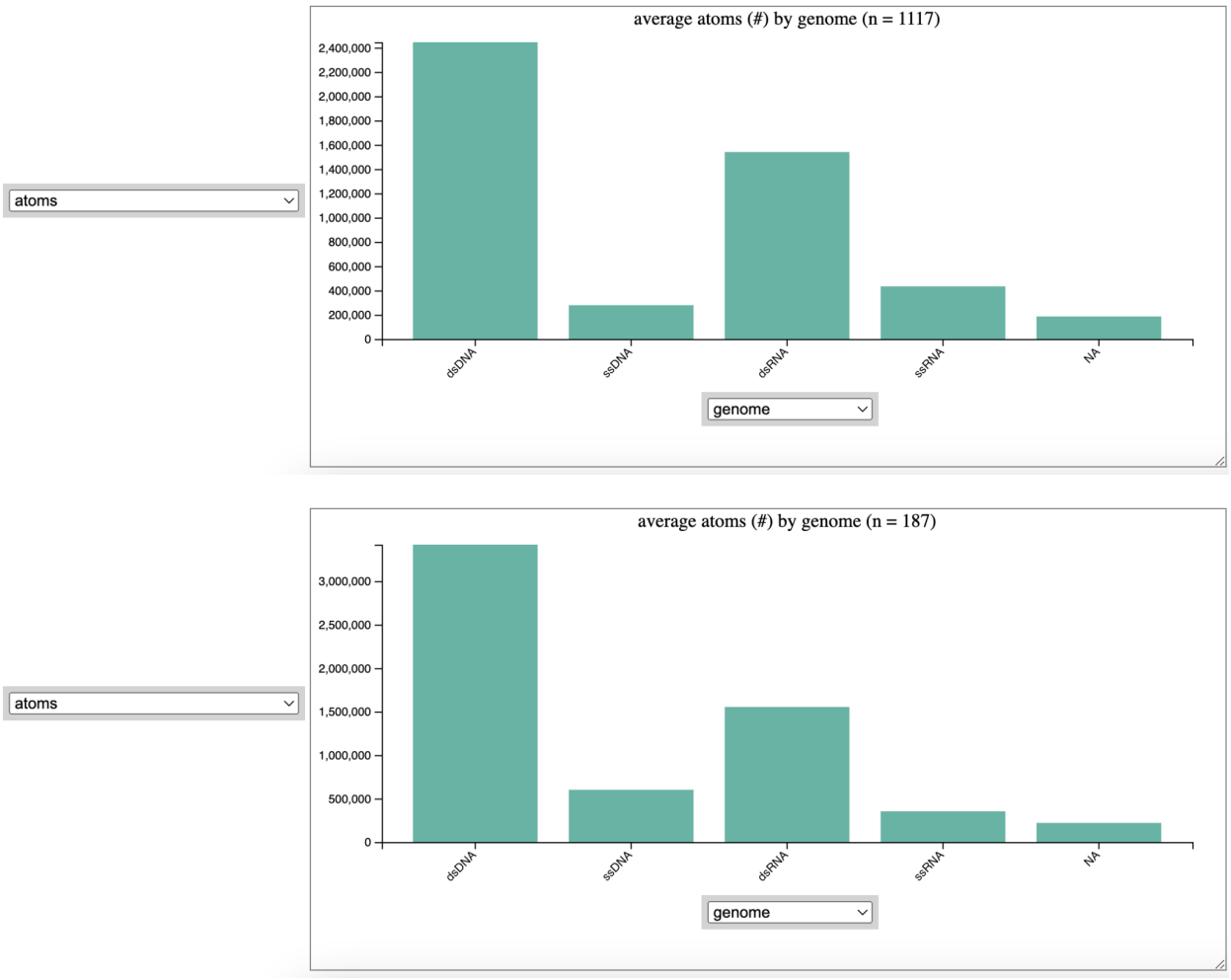



Figure 22. Average atoms by genome. As expected, dsDNA capsids have by far the most atoms.

Unsurprisingly, dsDNA capsids tend to be the largest; in both datasets the average number of atoms in dsDNa capsids is about double that of dsRNA capsids, with ssDNA and ssRNA capsids having even lower. NA capsids likely have low average atoms because not much information is known about them, so they may not have had full structures listed.

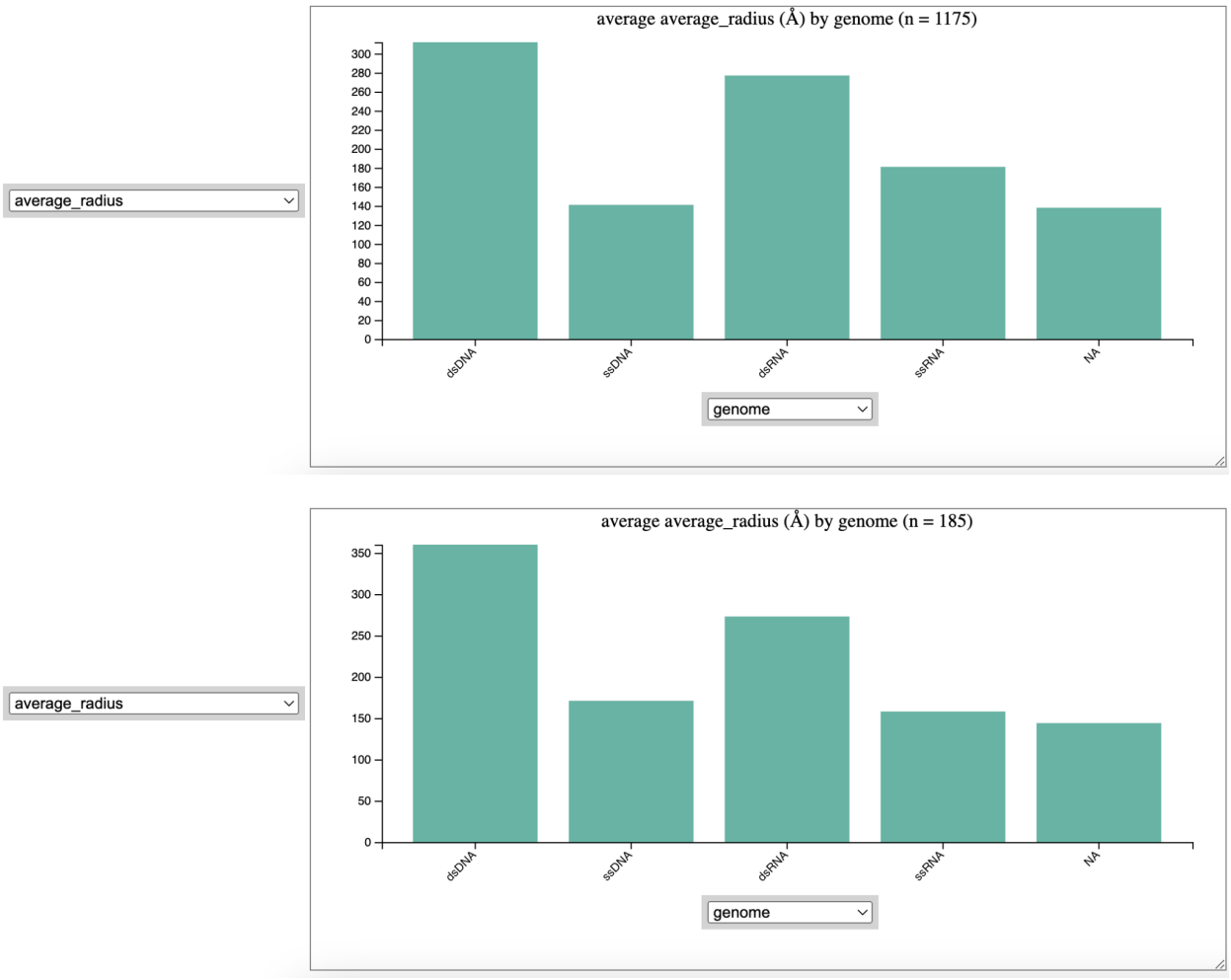



Figure 23. Average average_radius by genome. As we expected, dsDNA capsids tend to have larger average_radius values, but there is not as strong of a correlation as with genome and atoms.

We see similar correlations in average_radius by genome, with dsDNA capsids having the largest capsids, followed by dsRNA. Just as with atoms, there is a swap between ssDNA and ssRNA from the full to unique capsid set, but this is likely just the result of having relatively few capsids in the ssDNA group.

## 5.5 Triangulation Number

Along with traditional T-Numbers described in the introduction, there are pseudo-T (pT) number viruses, numbered after how many their protein domains are in the AU rather than how many separate proteins, along with categories like 7l and 7d for left and right-handed amino acids, respectively.

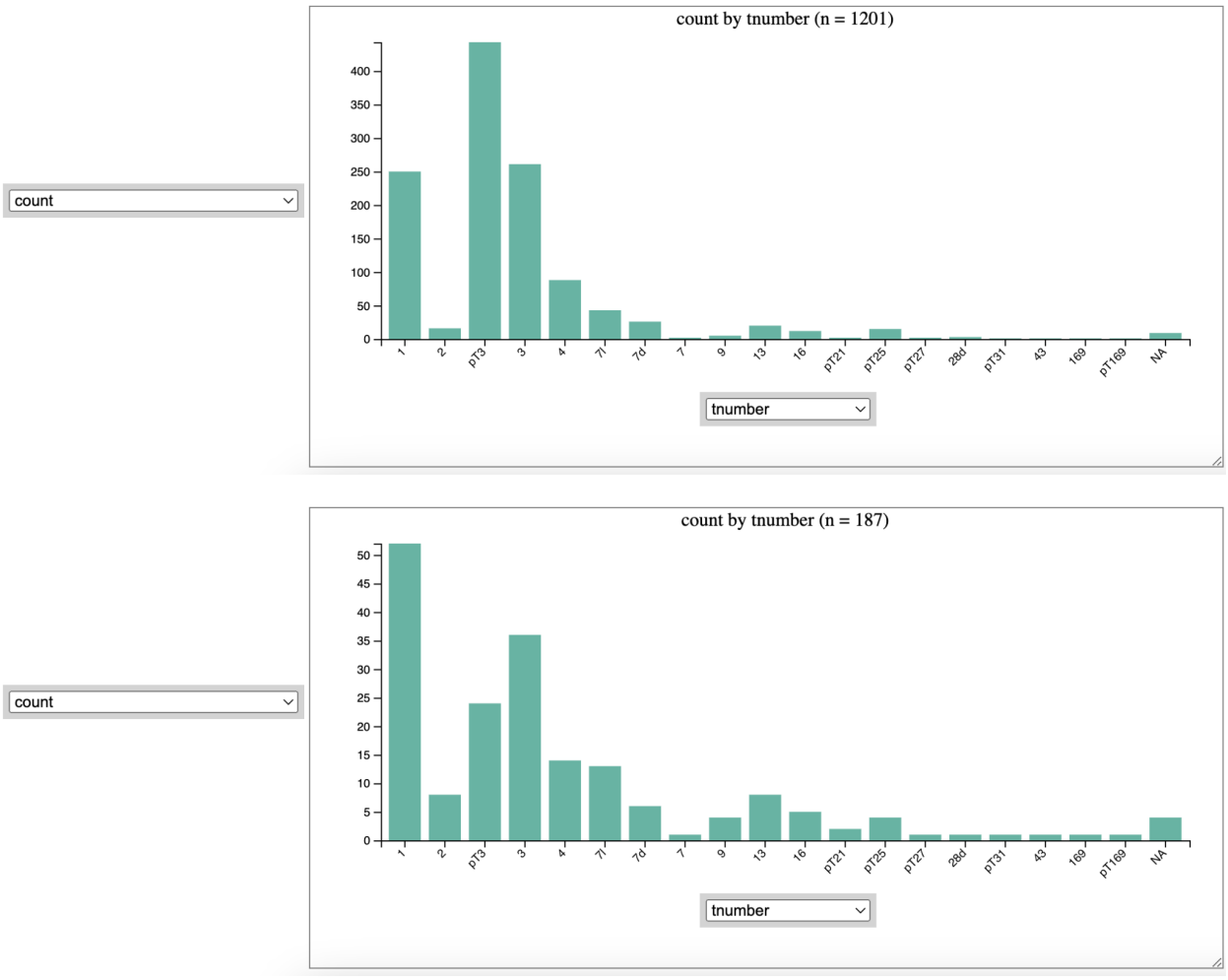



Figure 24. Count by tnumber in all vs unique capsids. pT3 is the most common T-Number in the full set, whereas T1 viruses were the most common in the unique set.

Interestingly, there are many more T1 viruses than pT3 in the unique set compared to in the full set where pT3 is the most common. Lower T-number values tend to be more common, with the exception of 2 - presumably because they are unique in

their capsid formation as you cannot get a T-number of two with the traditional tiling explained in the introduction.

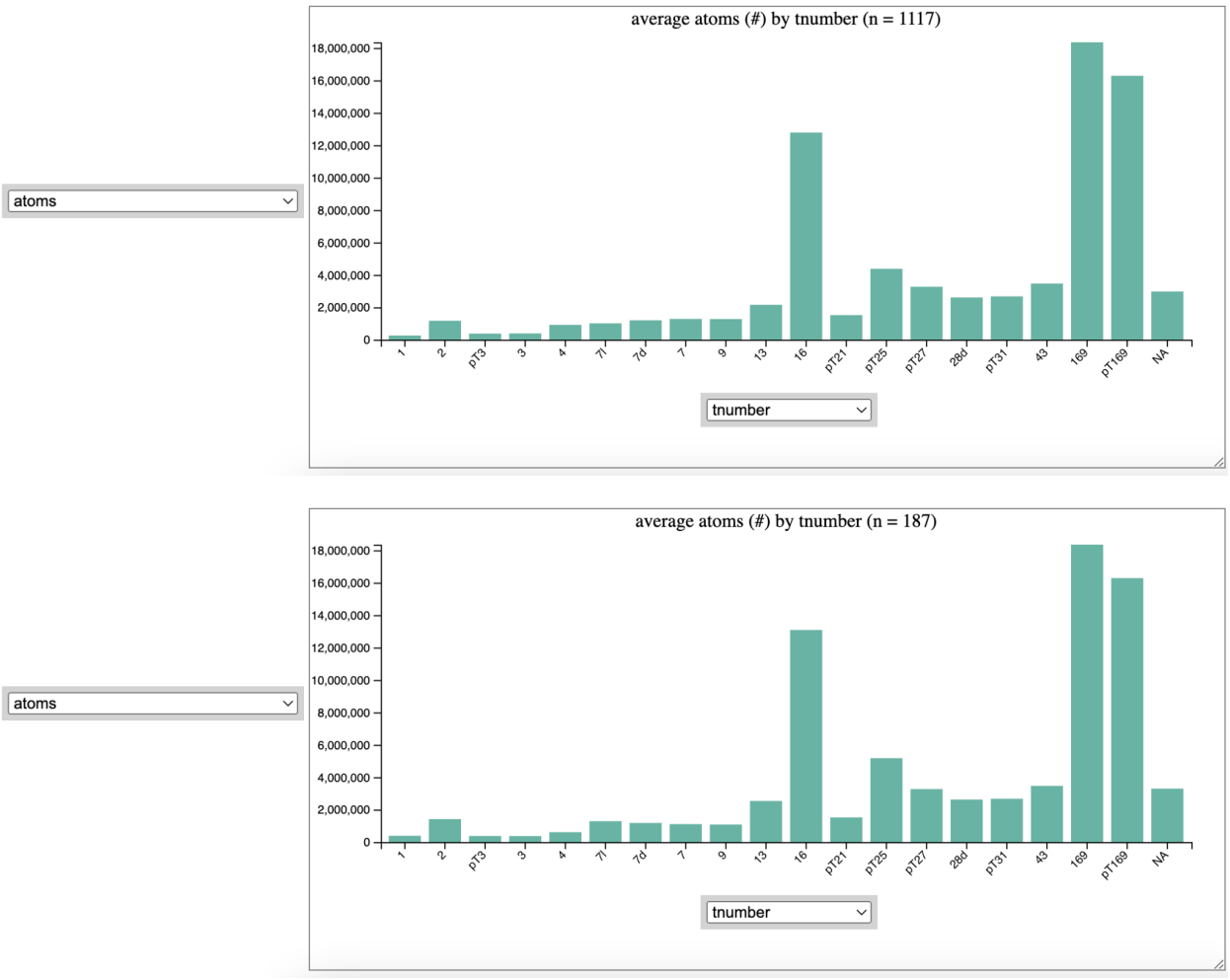



Figure 25. Average atoms by tnumber. Certain T-Numbers - 16, 169, and pT169 - look to have many more atoms on average than the other T-Numbers.

There seem to be many more atoms in T-Number 16, 169, and pT169 viruses. 16 may be in this category, and not other T-Number slightly higher, because of something about the geometry of capsids with even T-Numbers (note that 2 also has high atoms/average radius compared to other lower T-Numbers). There is also a slight dip for pT21 in both capsid sets, perhaps because of a similar quirk in its tiling. Alternatively, this could be because higher T-numbers are relatively scarce in the dataset, or tend to be single stranded viruses which are smaller.

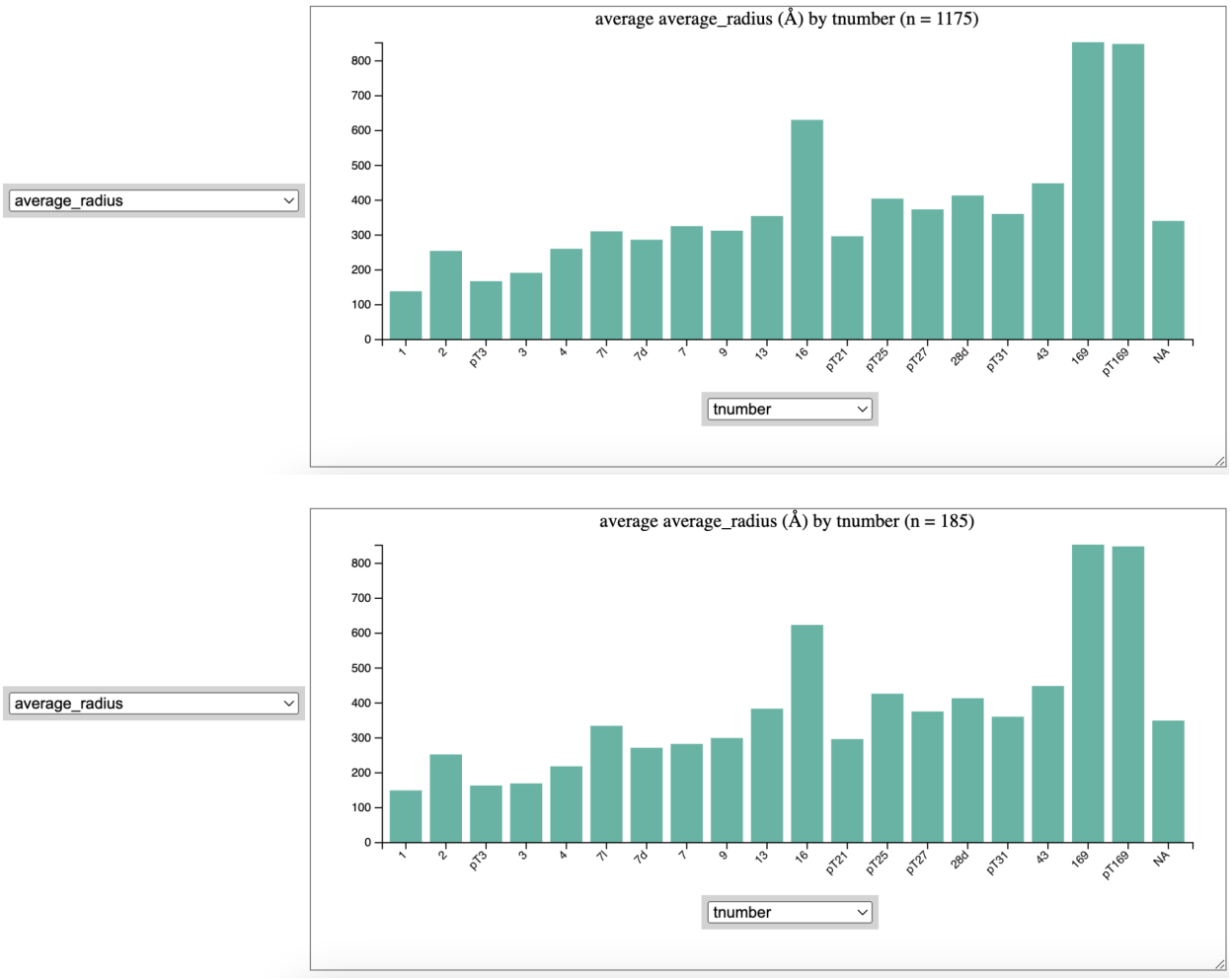



Figure 26. Average average_radius by tnumber. There is a similar pattern as with average atoms, though slightly less pronounced: T-Numbers 16, 169, and pT169 are higher.

T-Numbers 16, 169, and pT169 have the largest average radii, not other large T-Numbers, replicating the pattern from the atoms vs tnumber. We also see a slight increase relative to its neighbors at 2, and a relative dip at pT21.

## 5.6 Heatmaps

We look at frequency of pairings across combinations of closest_gp_aa, gauge_point, genome, and T-Number in both unique and full capsid sets. We also consider other_gp_aa vs closest_gp_aa frequency, as this reflects which other amino acids were with 5 angstroms of the closest AA, if any.

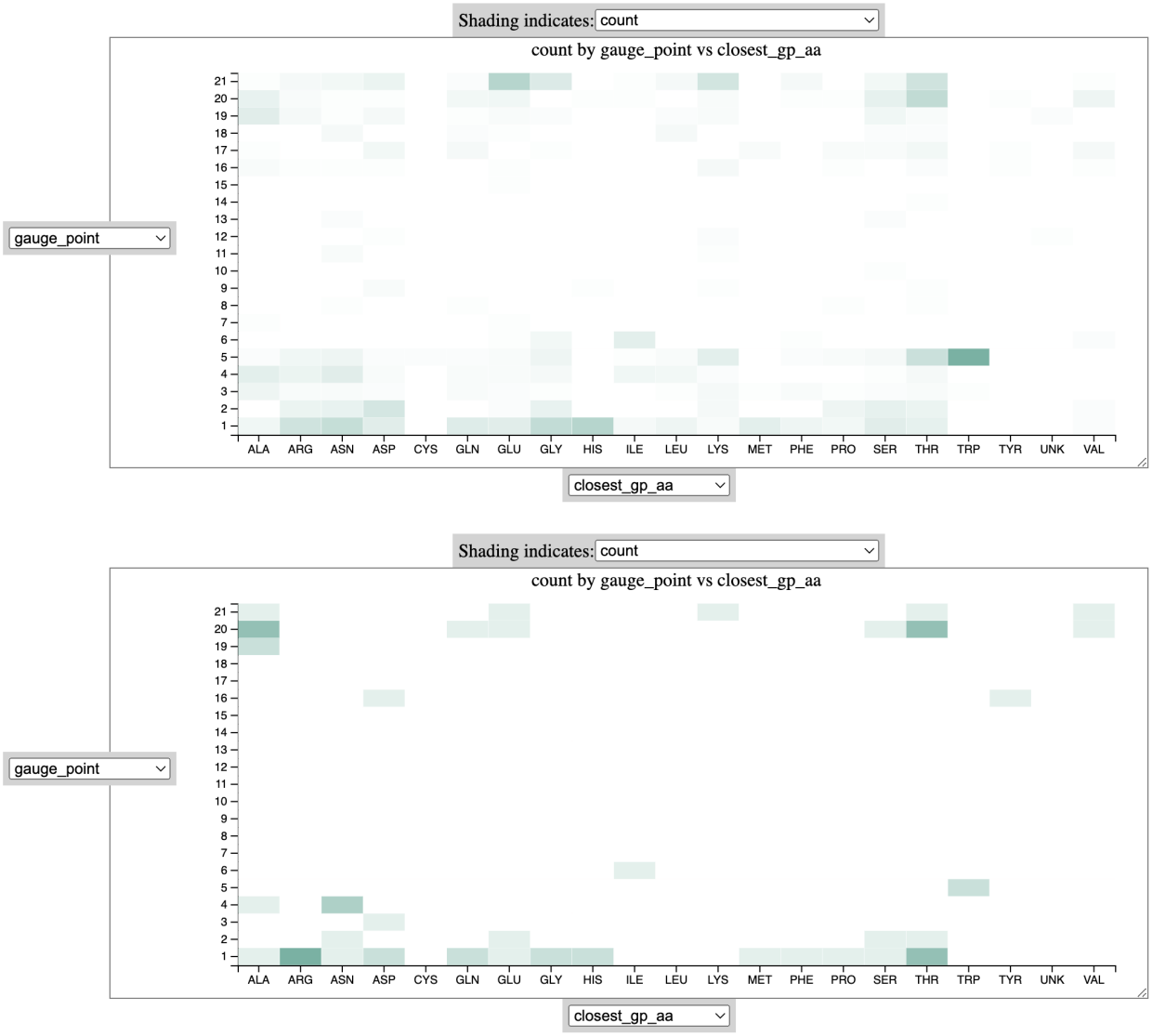



Figure 27. Count by gauge_point vs closest_gp_aa in all (top) vs unique (bottom) capsids. Particularly in the full set, there are many GP 5 capsids with closest GP AA tryptophan.

While tryptophan was not one of the most common closest_gp_aas, capsids with closest_gp_aa tryptophan tend to have gauge point 5 - this is one of the strongest patterns across unique and all capsids. Tryptophan is known to play a key role in some viruses (Medits et al. 2021), and tryptophan catabolism is a known strategy employed by the immune system. Isoleucine capsids are more likely to have GP 6, although the amount is relatively low - 12 in the full capsid set and 2 in unique.

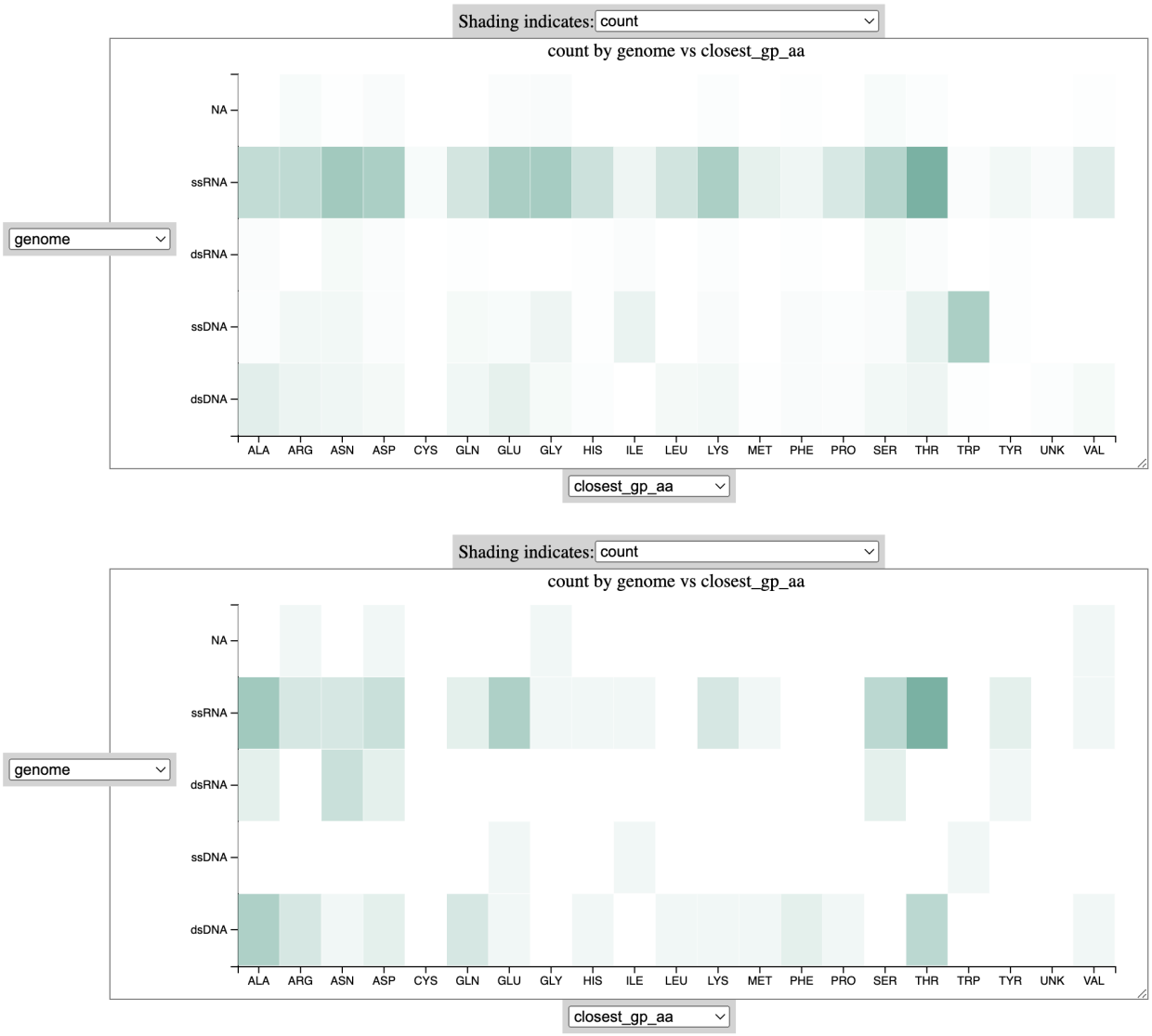



Figure 28. Count by genome vs closest_gp_aa in all (top) vs unique (bottom) capsids. Capsids with tryptophan as the amino acid closest to their gauge point tend to be ssDNA, even though ssDNA capsids make up a relatively small proportion of both the unique and full data sets compared to ssRNA.

Threonine capsids tend to be either dsDNA or (more often) ssRNA, with alanine capsids being more balanced between the two. Tryptophan capsids are almost exclusively ssDNA. Isoleucine seems to have more ssDNA capsids than average as well. This could relate to the distinctive gauge point profiles of these two amino acids.

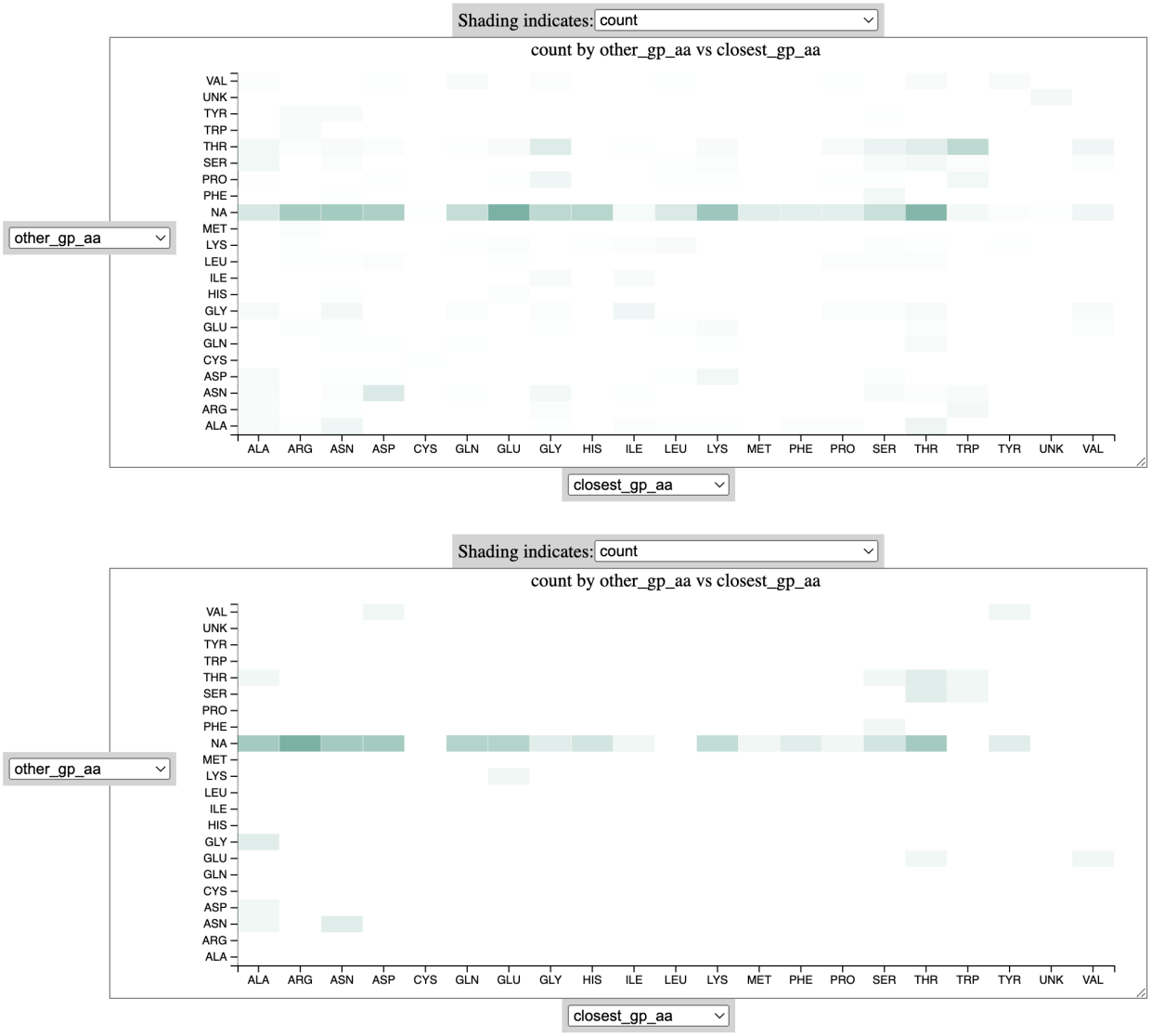



Figure 29. Count by other_gp_aa (the next closest residue within 5 angstroms of closest_gp_aa, if any) vs closest_gp_aa in all (top) vs unique (bottom) capsids. Other_gp_aa is set to NA if there are no amino acid residues within 5 angstroms.

Capsids with threonine for their closest_gp_aa tend to have the next closest residue within 5 angstroms be either another threonine or serine; alanine capsids have glycine, aspartic acid, asparagine, or threonine; arginine capsids tend to have no other

residue nearby. Threonine serine protein kinases are encoded by some DNA viruses, which could explain why they are paired together somewhat often. Tryptophan closest_gp_aa capsids tend to have other_gp_aa tryptophan or threonine, and rarely have no other amino acids near the gauge point.

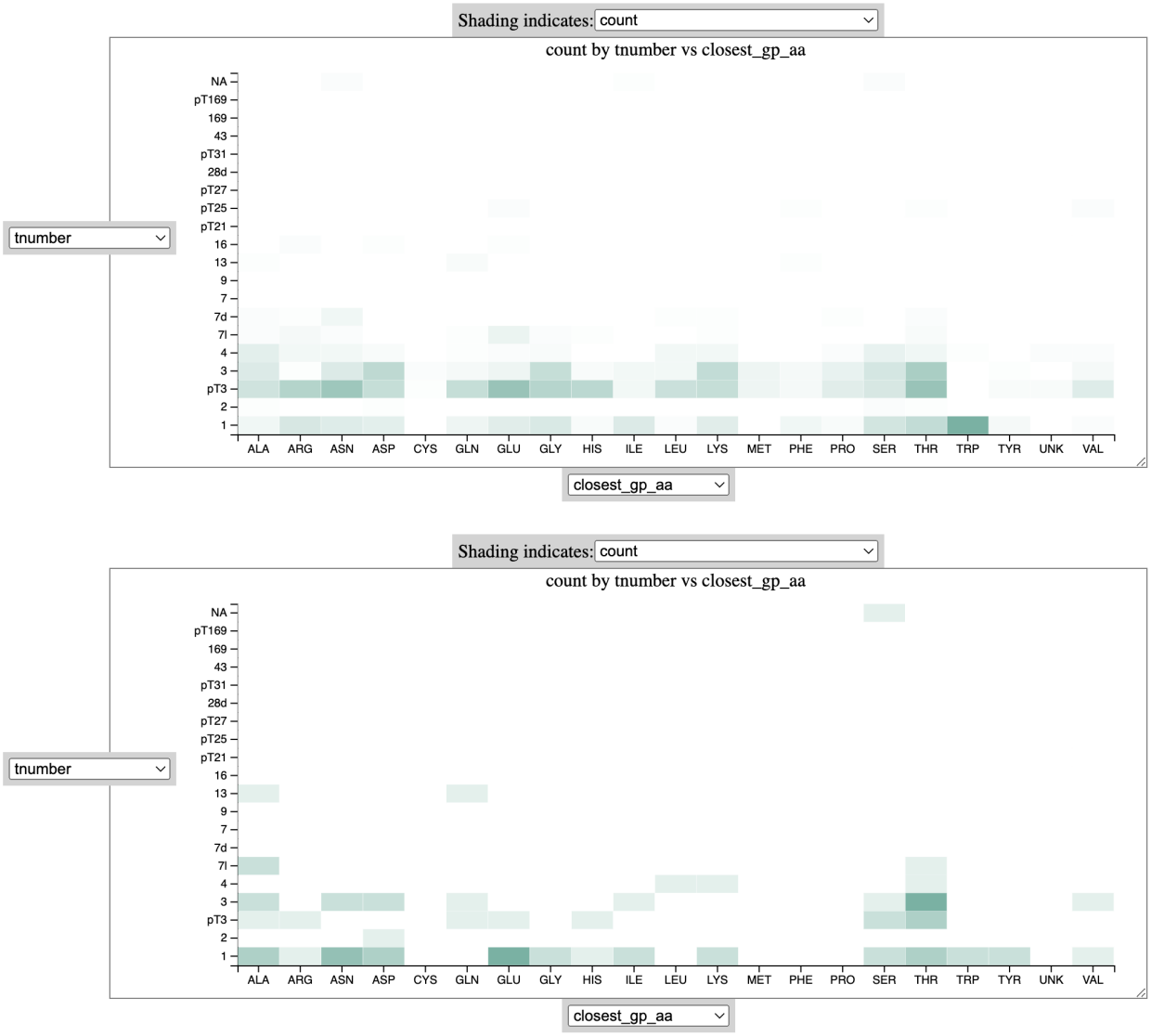



Figure 30. Count by tnumber vs closest_gp_aa in all (top) vs unique (bottom) capsids. The amino acid closest to the gauge point seems to have a weak relationship with T-Number - most are T1 or pT3 - but threonine capsids are often T3 in the unique set.

Capsids with GP AA threonine seem to have T-Number 3 or pt3 most of the time, with almost all others having T-Number 3 more often. Tryptophan capsids are almost

exclusively T-Number 1. Otherwise, there seems to be a fairly even T-Number distribution over closest_gp_aa.

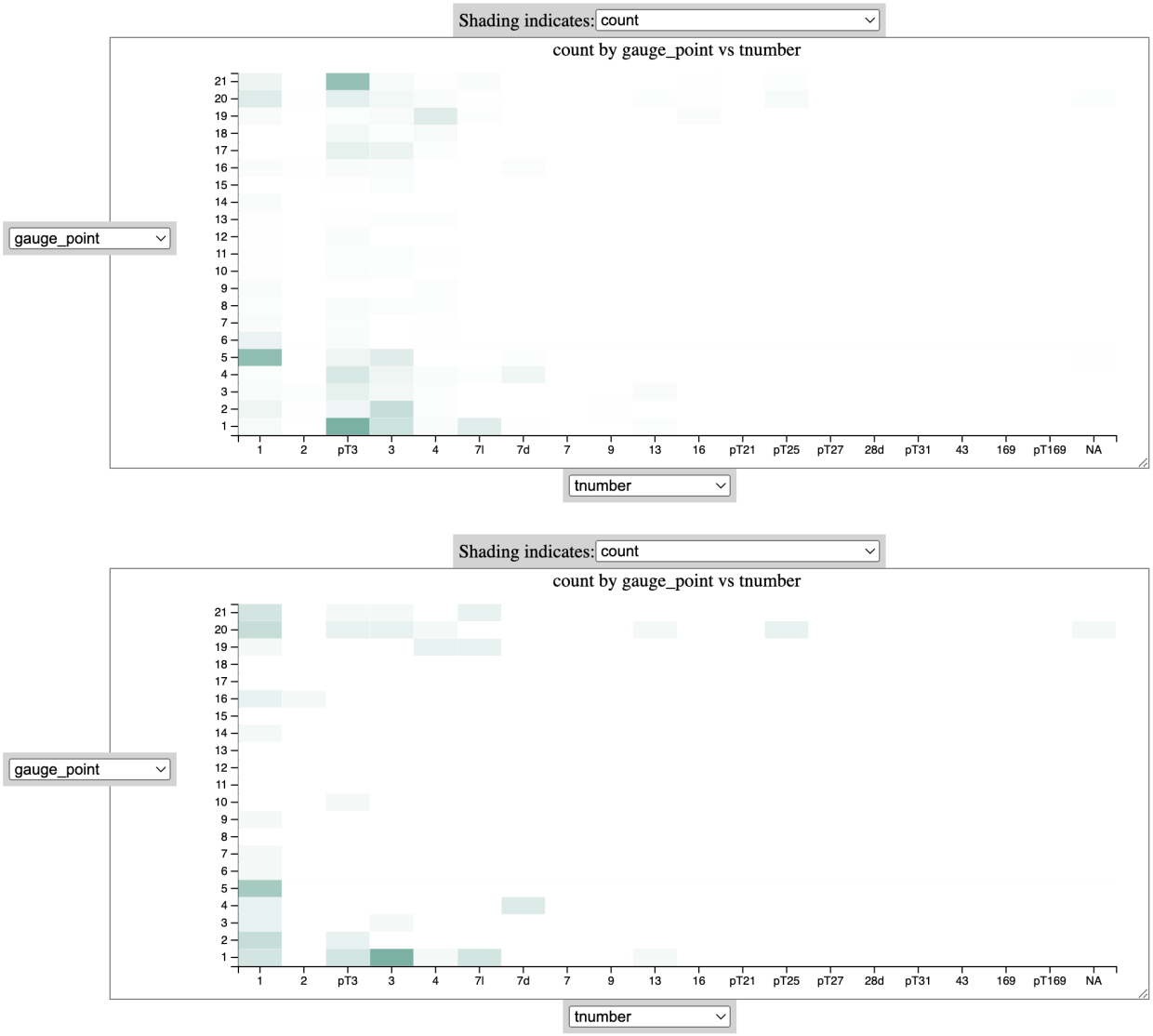



Figure 31. Count by gauge_point vs tnumber in all (top) vs unique (bottom) capsids. In the unique set, the majority of T3 capsids have gauge point 1, while T1 capsids tend to have gauge point 5.

pT3 capsids are almost all either gauge point 1 or 21, although this pattern does not completely transfer to the unique capsid set. Capsids with T-Number 1 are more likely to have gauge point 5. The majority of T-number 7d capsids have gauge point 4, but the frequency of T-Number 7d is relatively low - 15 in the full set and 4 in unique. With the exception of 7l, beyond T-Number 3, capsids tend to have higher gauge points. T-number 4 capsids tend towards gauge point 19.

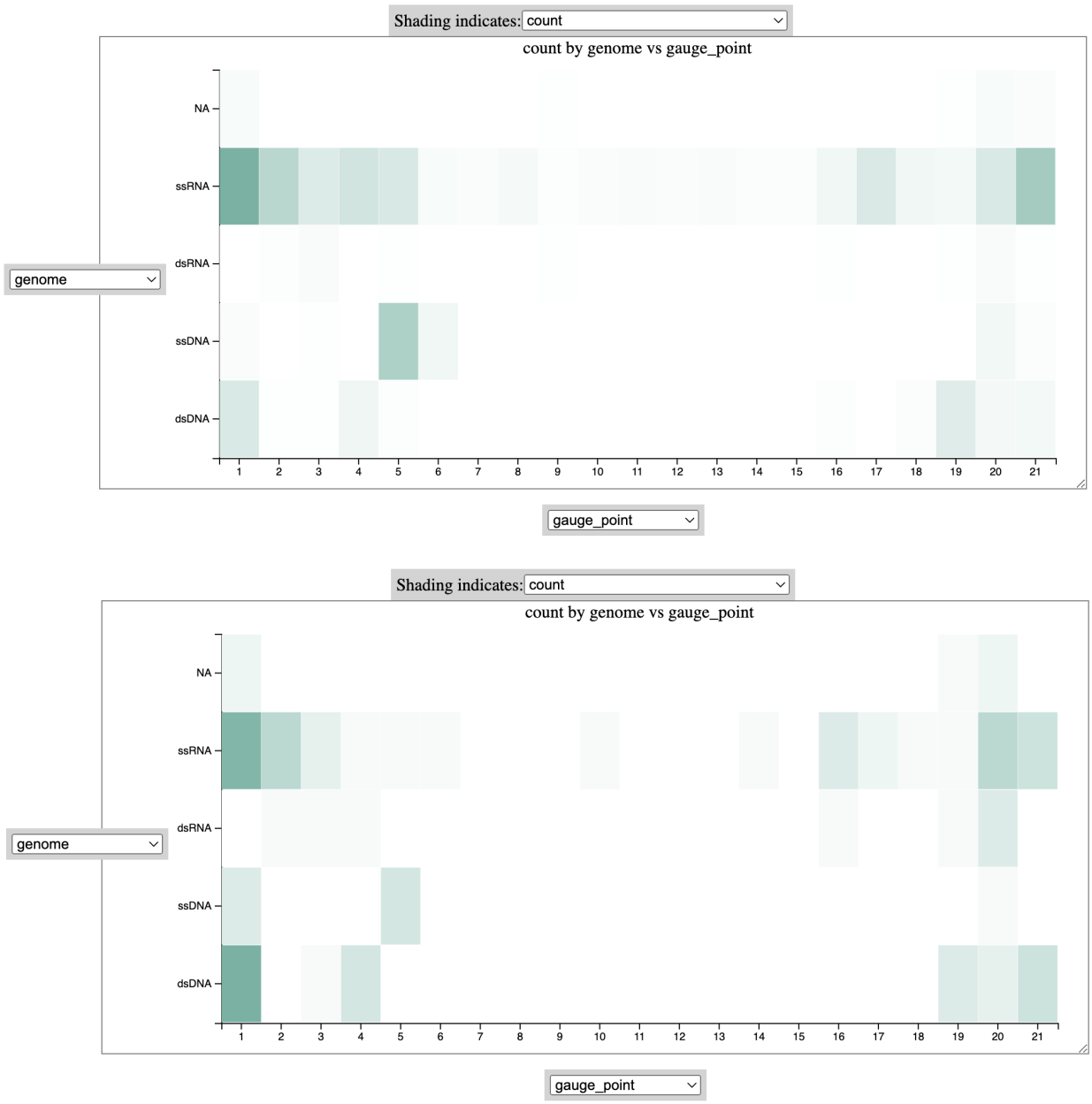



Figure 32. Heatmap of count by genome vs gauge_point in all (top) vs unique (bottom) capsids. Gauge point 5 capsids are quite frequently ssDNA, while gauge points 19-21 tend to have double-stranded genomes, which is congruent with the relationship we observed with the number of atoms earlier.

Gauge point 1 capsids tend to be either dsDNA or ssRNA; GP 18-21 tend to have slightly more double stranded capsids, but still a fair number of ssRNA. In the unique capsid set, there appears to be more of a concentration of dsDNA at gauge point 1. In both sets, ssDNA capsids tend to have gauge point 5. Although the number of

dsDNA viruses is fairly low in both sets, they never have gauge point 1 - the most common gauge point for other capsids.

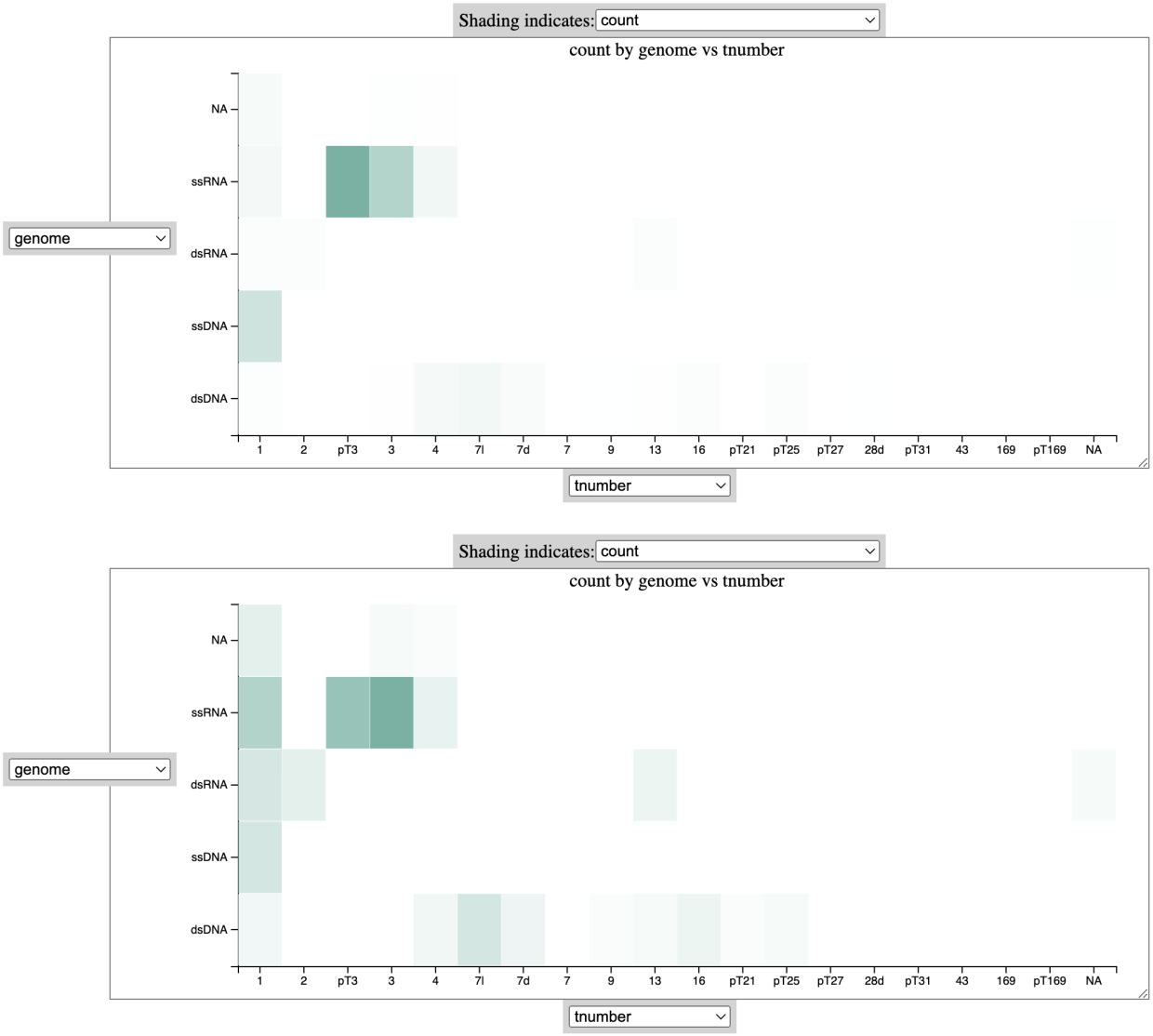



Figure 33. Count by genome vs tnumber in all (top) vs unique (bottom) capsids. As we would expect, lower T-Numbers tend to have single stranded genomes, associated with smaller capsid size.

It seems that ssRNA contains many of the T-Number pT3 and 3 capsids in both capsid sets. In the full set, most of the T-Number 1s are ssDNA. Higher T-Numbers, with the exception of 13, are mostly dsDNA. T-number 2 capsids are almost exclusively dsRNA, with the single exception of the virus with ID 3iyl.

## 5.7 Conclusion

Understanding the relationships between properties of spherical viruses could give us important guidance in drug and vaccine design, among other things. Gauge points and nearby amino acids offer novel insights into how viruses form and bind to cell membranes. There are many potential connections to examine, so the main goal of this project was to build data analysis pipelines and visualization tools to investigate them.

Specifically, we looked at relationships between gauge points, nearby amino acids, genome, and T-number. We were able to see correlations between atoms and gauge points, which could indicate which types of capsids use certain protrusion sites. We also found certain amino acids that tend to be grouped together, and correlate with capsids that have certain genomes and gauge points. We looked at connections between gauge points and genomes, as well as T-Numbers, which may be intermediate variables that can shed light on the unclear relationship between gauge points and capsid size. Overall, we showed that there are revealing correlations among traditional groupings of viruses like T-Number and genome type, and structural characteristics could be more predictive of capsid composition than chemical properties in some cases.

Future research will be benefited by having access to more capsid data in ViperDB, and could likely look at other attributes relating to the genome and capsid charge to make more sense of our preliminary amino acid findings. Additionally, fold data could be investigated manually or pulled from other sources, as this could have interesting correlations with gauge points but we had so few non-SJR folds that it is hard to say. By making the website, data, and methods available, hopefully these results can be expanded on to lead to a better understanding of the structure and function of viruses.

# Works Cited


Krupovic M, Koonin EV. Multiple origins of viral capsid proteins from cellular ancestors. Proc Natl Acad Sci U S A. 2017 Mar 21;114(12):E2401-E2410. doi: 10.1073/pnas.1621061114. Epub 2017 Mar 6. PMID: 28265094; PMCID: PMC5373398.

Medits, I.; Heinz, F.X.; Stiasny, K. An Absolutely Conserved Tryptophan in the Stem of the Envelope Protein E of Flaviviruses Is Essential for the Formation of Stable Particles. *Viruses* **2021**, *13*, 1727. https://doi.org/10.3390/v13091727

Nasir A, Caetano-Anollés G. Identification of Capsid/Coat Related Protein Folds and Their Utility for Virus Classification. Front Microbiol. 2017 Mar 10;8:380. doi: 10.3389/fmicb.2017.00380. PMID: 28344575; PMCID: PMC5344890.

Voyles, Evan. Decomposition of Virus Normal Modes into Spherical Harmonics. 2020 Feb 12. https://github.com/ejovo13/viruses/blob/master/+ejovo/media/SIP.pdf

Wilson DP, Roof DA. Viral Phrenology. Viruses. 2021 Oct 30;13(11):2191. doi: 10.3390/v13112191. PMID: 34834999; PMCID: PMC8618131.

Wilson DP. Unveiling the Hidden Rules of Spherical Viruses Using Point Arrays. Viruses. 2020 Apr 20;12(4):467. doi: 10.3390/v12040467. PMID: 32326043; PMCID: PMC7232142.

# Appendix

## find_aas.py

The purpose of this script is to take in two input files: one with the x,y,z coordinates of the full capsid of a virus, and the other with coordinates of the points of a relevant point array (PA), and compile a list of the closes amino acid residue to each PA point for each protein chain in the capsid. In order to accomplish this, I first parsed the coordinates into a dictionary mapping each chain to the coordinates of each constituent atom. I also parsed the coordinates of the PA file into a list of [x,y,z] point coordinates. Next, I looped through each chain and used the function scipy.spatial.distance.cdist (scipy is a python library for scientific computing) to compute the distances from each member point for the chain to each point in the point array. I stored the minimum distance, along with the corresponding residue and atom, and looped through the sorted distances to find one where the distance was less than 5 (or any given number) and the residue was not the same as the closest one, storing that residue if it existed or N/A otherwise. Finally, I wrote the results to an xlsx file, with a row for each PA point and 5 columns for each PA chain.

Most viral capsids have multiple PAs of interest, so I also wrote a shell script callen run_pas to run find_aas.py with a capsid for every point array file in a designated folder, and adapted the find_aas.py script so that if the excel file being written to (named after the capsid) already exists, then the results would be written to a new sheet so that the result of the script being run on a folder is an excel file with sheets named after each PA and storing their respective results.

Additionally, since in the script I made use of scipy, pandas, numpy, and openpyxl (all python libraries which must be installed by the user), I used pyinstaller to convert the script into an executable file find_aas_exec bundled with all the dependencies mentioned (and adapted the run_pas script accordingly, naming the new one run_pas_exec.sh). However, because the executable is slightly slower than the python script given the bundled dependencies, I ended up using the python script in the automated pipeline.

## uniq.ipynb

uniq.ipynb was made to compile a list of capsids with unique combinations of fields T-number, genome, family, and genus. This is to try to reduce the effect of the bias introduced by the scope of virus research. Scientists tend to study viruses that impact humans, and viruses that are easy to study (easy to keep alive and grow in lab conditions, among other things). Among viruses with overlapping fields, the ones with the lowest resolution were chosen. By default, all viruses with only C alpha residues are removed. The output is written to unids_ca.txt and uniq_lres_ca_removed.xlsx which are in the data directory.