Course: Binp39

Credits: 30

Project name: Integrative Bioinformatics for Rational AAV Vector Design

Student: Hooi Min Tan Grahn

Supervisor: Marcus Davidsson
Email: Marcus.Davidsson@raaven.se

Integrative Bioinformatics for Rational AAV Vector Design

Abstract

This research project seeks to understand the complex molecular mechanisms behind capsid modification and tropism in Adeno-associated virus (AAV) vectors using a comprehensive bioinformatics approach. The specific objectives include developing a bulk sequencing workflow to identify recurring patterns within computationally segmented amino acid (aa) sequences (7-aa-long polypeptides) in cells following AAV vector administration. By utilizing mRNA Illumina sequencing, the project will investigate the most effective aa patterns that replicate in tissue samples post-virus infection and their effects on capsid modification and DNA virus tropism integration. The bioinformatics analyses in this study include several key methodologies. First, sequencing reads undergo preprocessing, which involves quality trimming to remove low-quality bases and artifacts, ensuring high fidelity in downstream analysis. Following this, the reads are aligned and mapped to a customized reference library database tailored for the study, which allows for accurate matching and analysis of the sequences of interest. Once the reads are mapped, barcode tracking is used to quantify the frequencies of specific amino acids within the sequences, facilitating a detailed assessment of amino acids profile present in the viral capsid proteins across different samples. This step is essential for identifying which amino acids are most commonly associated with certain modifications or tropism changes in the viral capsid. Finally, recurring sequence patterns of peptides are identified to uncover motifs or sequence signatures that may influence capsid properties. These recurring patterns can provide insights into how specific peptide sequences contribute to the virus's ability to change its tropism, ultimately aiding in the understanding of viral behavior and enhancing the design of more effective AAV vectors. By integrating these bioinformatics methods, this research endeavors to provide comprehensive insights into the molecular intricacies underlying capsid modification and tropism. These insights hold immense potential for the rational design and optimization of AAV vectors for diverse applications in gene therapy and biotechnology, thereby contributing significantly to advancements in the field.

Availability and Implementation: This project is built using Bash, Python, Awk and R. All scripts used in this project are available for download from https://github.com/hooimin7/BINP39_project

Contact: ho4588ta-s@student.lu.se

Introduction

Adeno-associated virus (AAV)-mediated gene delivery represents a promising strategy for advancing the next generation of clinical gene therapies [1,2]. The recombinant AAV (rAAV) is one of the leading vectors for gene therapy applications that deliver gene-editing enzymes, antibodies, RNA interference molecules and peptides to anatomical sites where viruses accumulate and exert therapeutic effects in AAV gene therapy [3]. rAAV vectors are limited by the need for host-cell synthesis of double-stranded DNA from their single-stranded genomes. To overcome this, researchers developed self-complementary rAAV (scAAV) vectors that bypass this step by packaging DNA dimers, allowing them to form double-stranded DNA upon uncoating [4,5]. In general, the single-stranded DNA (ssDNA) genome of wild-type AAV2 (wtAAV2) is approximately 4.7 kb in length. AAV serotypes are classified in a separate genus within the *Parvoviridae* family, known as *Dependovirus,* because productive infection generally requires either co-infection with a helper virus (such as adenovirus or herpes simplex virus) or the induction of cellular stress [6]. In the context of AAV, serotypes refer to different naturally occurring or engineered variants of AAV that have unique capsid proteins. These capsid differences give each serotype specific properties, such as the ability to infect certain types of cells or tissues. This makes certain AAV serotypes more suitable for targeting specific tissues in gene therapy applications, such as the brain, liver, or muscle [7,8]. Other findings suggested that AAV2 capsids could tolerate ligand insertions. The researchers inserted a 14-amino-acid targeting peptide (L14) into six regions of the AAV2 capsid protein, successfully generating three mutants displayed L14 on their surface, opening possibilities for targeted gene therapy [9]. The AAV capsid is assembled from viral structural proteins encoded by the cap gene. Capsid assembly is facilitated by an assembly-activating protein, followed by the packaging of the viral genome into the capsid, a process mediated by replication proteins flanked by two 145 bp long inverted terminal repeats (ITRs) [10] (Fig 1A). To enhance the development of AAV vectors and advance capsid engineering toward late-stage clinical trials, Marcus and colleagues developed a method called barcoded rational AAV vector evolution (BRAVE), which incorporates the advantages of rational design [3]. Rational design leverages existing knowledge of AAV biology and structure, typically targeting surface-exposed regions of the AAV capsid [11]. The number of variants tested is relatively small, often in the tens or hundreds, as the designs are guided by specific hypotheses or known mechanisms. Additionally, rational design can be enhanced with computational tools to improve the precision of the modifications [3]. A previous study addressed the need for error-free barcode clustering in *in vivo* applications like cellular fate mapping [12]. While various clustering methods have been attempted on samples with unknown diversity, the study validated the use of the message passing clustering algorithm in the Starcode software by leveraging high-diversity, barcoded fragment libraries. The findings indicated that while this algorithm was robust for clustering degenerate barcodes, it could produce false clusters if the Levenshtein distance threshold was too high. The study recommended validating clustering with paired-end sequencing data from an *in vitro* plasmid library to establish a subset of “valid barcodes,” minimizing false clustering in the *in vivo* data [13]. High-resolution lineage tracking using

DNA barcodes could detect rare beneficial mutations, estimate their timing, and assess fitness effects. By tracking multiple lineages simultaneously, this method provided detailed insights into population dynamics [14]. Our study further refines AAV production plasmid to connect capsid structure with an *in vivo* molecular barcode. The barcode was inserted into the 3′ untranslated region (UTR) of green fluorescence protein within a gutted self-complementary AAV (scAAV) genome while the AAV2 Rep/Cap genes were expressed from the same plasmid but positioned outside the ITRs, thereby linking the capsid structure with the barcode. In this system, peptides (seven amino acid) or fragment were incorporated at the N587 position of the VP1 capsid protein (Fig 1B). My responsibility in this project involves reproducing the pipeline [3] and tailoring the bioinformatics workflow to handle three distinct plasmid libraries. Libraries p005 and p007 were designed using rational design principles, whereas p006 was engineered by incorporating degenerate NNK codons to introduce amino acid randomization (Fig 1C). The NNK configuration is a method used in genetic mutagenesis to efficiently encode all 20 amino acids with only one stop codon, reducing the total sequence library size. In this approach, "K" can be either "T" or "G," resulting in a 32-fold degeneracy (32 unique codons). By using NNK for mutagenizing target regions, we can screen a significantly smaller, manageable library with diverse capsid variants, optimizing the discovery of improved activity without the need for an excessively large sequence pool [15].

Methods

Total RNA was extracted from brain tissue, primary neurons, and HEK293T cells, then treated with DNase I to remove any DNA contamination. The RNA was reverse transcribed to cDNA, which was amplified by PCR using specific primers. Barcode-containing PCR products were purified, followed by an Illumina adapter PCR and a Nextera XT Index PCR, with additional purification steps in between. The final PCR products were sequenced using the Illumina NextSeq 500/550. Three AAV batches were produced. After production, purification, and titration, those batches were treated with DNase I and Proteinase K. The viral lysates then underwent two rounds of PCR to add Illumina-compatible sequences and NexteraXT indexes, followed by purification. The indexed samples were sequenced on the Illumina NextSeq500/550.

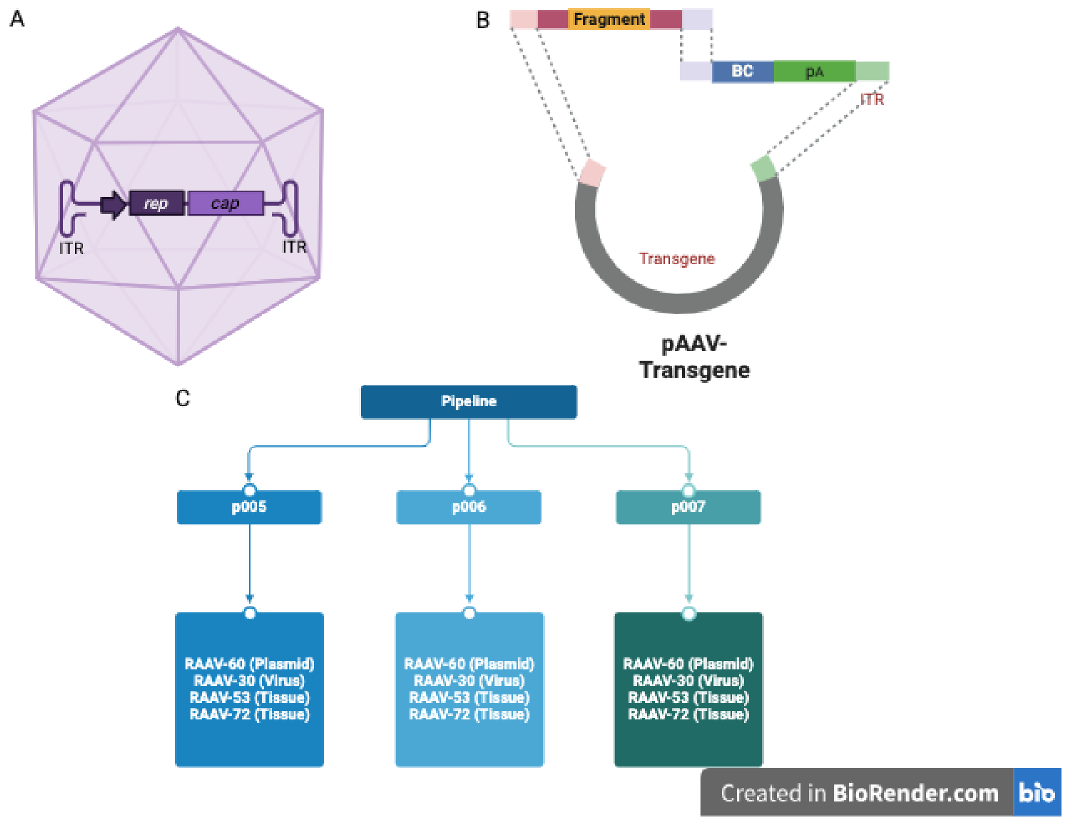


Fig. 1 Adeno-associated virus (AAV) vector genome and workflow structure. A) The AAV genome is linear, comprising single-stranded DNA (ssDNA) that is approximately 4.7 kilobases (kb) in length. It is flanked by two inverted terminal repeats (ITRs), each 145 nucleotides long, located at the ends of the genome. These ITRs flank two viral genes: rep, which is responsible for replication, and cap, which encodes the structural capsid proteins. B) The BRAVE rational design process shows how the fragment (21 nucleotides) and barcode (BC) are positioned closer together after plasmid preparation. In brief, the oligonucleotide pool was assembled into an AAV production backbone containing the cis-acting AAV2 Rep/Cap and CMV-GFP flanked by ITRs. Simultaneously, a 24-bp random molecular BC was inserted into the 3′ UTR of the GFP gene. To create a look-up table linking

each BC to its corresponding peptide, a portion of the plasmid preparation was treated with Cre-recombinase *in vitro*. C) Workflow structure and data resources.

Fastq files preprocessing

Hierarchy of the files in this study were Plasmid Library (RAAV-60), Virus library (RAAV-30), Tissues mRNAs (RAAV-53 and RAAV-72). RAAV-60 consisted of three different library batches, in which they were p005, p006, and p007 (Fig 2). Each batch consisted of an Illumina sequencing dataset of 2x151 bp paired-end reads were utilized. An Illumina sequencing dataset of 1x151 bp paired-end reads were used for both virus library and tissue mRNAs. Each batch was quality control with FastQC (version 0.11.9-Java-11) before and after a trimming procedure named quality trim (BBMap, bbduk2.sh,

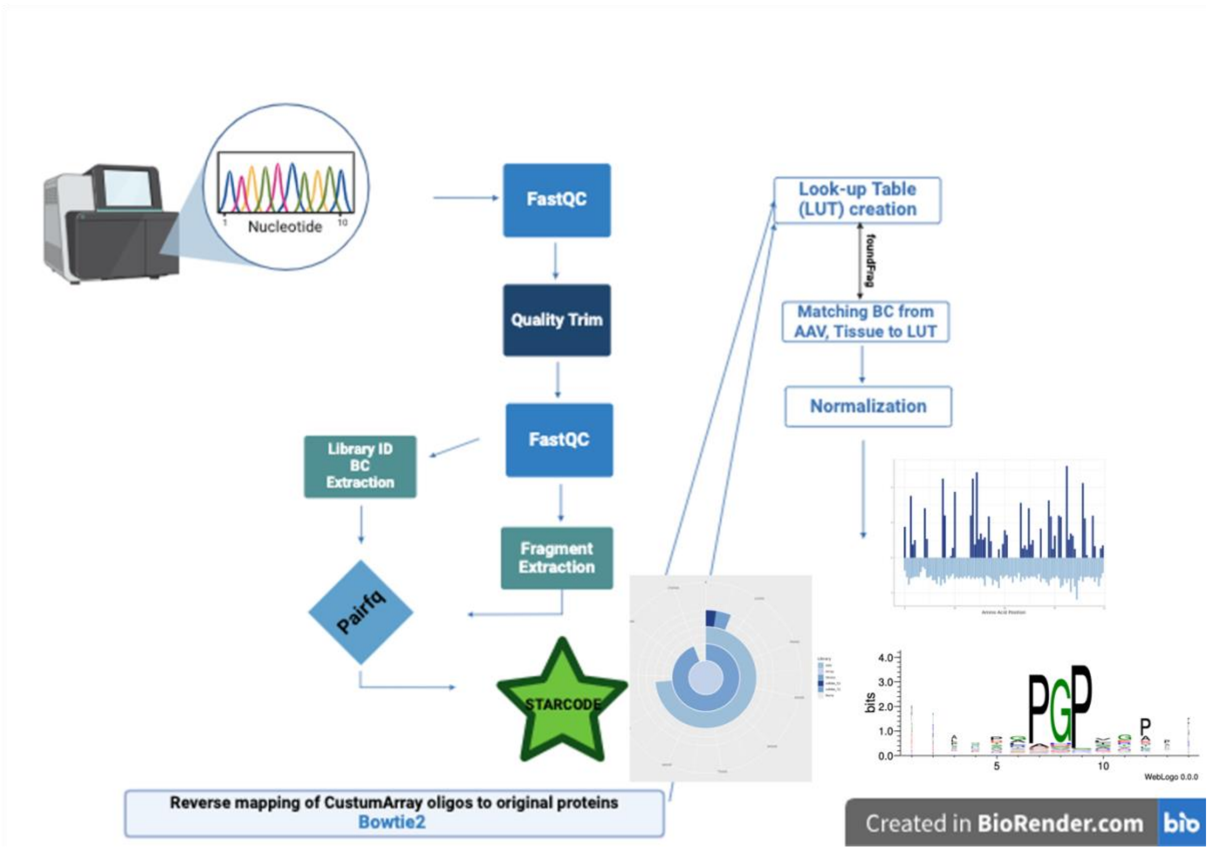


Fig 2. Comprehensive Bioinformatics Tools and Data Analysis Workflow.

An overview of the data processing pipeline, including FastQC analysis of fastq reads before and after quality trimming; extraction of library IDs, barcodes (BC), and fragments; pairing BC with fragments using pairfq; BC reduction via Starcode; mapping custom array oligos back to the original proteins using Bowtie2; creation of a Look-Up Table for identifying fragments derived from the viral library and infected tissue samples (foundFrag); normalization of all reads using various computational methods; and subsequent data analysis to generate outputs like circular bar plots, pairwise plots, and motif creation.

https://github.com/BioInfoTools/BBMap/blob/master/sh/bbduk2.sh) [16]. Given the analysis is highly sensitive to sequence length, I performed mild trimming of nucleotides with quality scores below 15.

Then, I also used bbduk2.sh for fragments, barcodes and library IDs extraction. I extracted the sequences which were surrounded by the matching 13-16 (k=16, mink=13) nucleotide sequences from the known reference for each plasmid. I allowed for 2 nucleotides mismatches (hammingdistance=2, qhdist=2). The fragment was required to be 21 nucleotides length (minlength=21 maxlength=21), barcode was 23-24 nucleotides length (minlength=23 maxlength=24), library ID was 3 nucleotides (bbduk2.sh and python script). Fragment sequences were extracted from R2-fastq quality trimmed files. It was done when R2 fragment was trimmed from the left side (lliteral) and from the right side (rliteral). Barcode sequences were extracted from R1-fastq quality trimmed files. R1 was reverse complement to the reference sequences. Therefore, all the reference sequences used for barcode extraction were also reverse complement to the reference sequences. R1 was trimmed from the right side (rliteral), then, and R1 was trimmed from the left side. Library IDs were extracted from the R1-fastq quality trimmed files. Library ID was trimmed from the left side (lliteral). I concatenated the fastq files of samples (fragments 5a with fragments 5b; barcodes 5a with barcodes 5b; library IDs 5a with library IDs 5b) where more than one file from the same sample was available (in which it applied to both plasmid library of p005 and p007).

Sync paired-end fastq files

After fragments, barcodes and library IDs extraction (by bbduk2.sh), these fastq files were paired with one another using (**Pairfq** version 0.17.0). There were several methods in pairfq and I used makepairs method to pair the forward and reverse reads and wrote the singletons to separate files. The extracted fragments were paired with the extracted barcodes, then the library IDs were paired with the extracted barcodes that had been paired once with the extracted fragments. Following by that step, the paired barcodes that had been paired twice were paired again with the extracted fragments that had once paired earlier. Hence after three times of pairing, each fragment, barcode and library ID with similar identifiers were listed in a table.

Barcodes reduction

The paired extracted barcodes further clustered into a “consensus” barcode using **starcode** algorithm (starcode version 1.4) [17]. I set the starcode parameter for the clustering algorithm that allowed the maximum distance between items in a cluster to 1 and the algorithm should perform 5 repetitions or rounds of clustering. In essence, this workflow clusters all peptide reads with highly similar barcodes into a single "multiple-alignment". This method, when used restrictively, has been previously demonstrated to outperform other reduction techniques by avoiding false clustering of barcodes [13].

Sequence alignment

The paired fragments were aligned using BLAST (version Blast: 2.15.0+, https://anaconda.org/bioconda/blast) against to the reference fragments with gene names information. The reference fragments were used to make the customized BLAST database. I ran the BLAST nucleotide search tool and I set up several BLASTn parameters. I limit the number of target sequences to 25, set the word size for the BLAST search to 11 and specified the output format as comma-separated values (-outfmt 10). The BLAST output file was then used for analyzing the percentage of alignment, the bitscores and mismatches. BLAST was performed to determine the origin, and the oversampling of sequencing (using all reads with the same barcode). The information was incorporated into the same table as paired fragments and paired barcodes based on the Look-up Table number (LUTnr).

Creating Look-up Table (LUT)

I created the LUT using the R language (version 4.3.2) and this R-based analysis framework was a parallelized implementation of the MapReduce programming philosophy [18–20], introduced by Google in 2004. The MapReduce model suggested to distribute and parallel complex job process by splitting the job into multiple tasks through the use of map and reduce stages [21].

Short sequencing reads alignment to reference genes and SAM/BAM files indexing

The purpose of using Bowtie2 (version Bowtie2: 2.5.4, https://anaconda.org/bioconda/bowtie2) was to align the reference fragments with gene names to the NCBI gene sequences which was collected in an excel file at lab. The Bowtie 2 index was made from a set of sequencing read files and output a set of alignments in SAM format. SAMtools (version 1.18) was used to convert the SAM file to a BAM file and read the alignment data from the files in sorted BAM file into a GAlignments object. These objects represented genomics alignments. The GAlignments object included metadata columns, such as mapping quality and CIGAR strings.

Adeno-associated virus batches production (RAAV-30) using the same plasmid pool (RAAV-60)

RAAV-30 consisted of three different libraries of DNA viruses which derived from three plasmid libraries of RAAV-60. They were p005-AAV-01, p006-AAV-02 and p007-AAV-03. The final PCR products were sequenced from the 3'-end, producing a single 100 bp read that covered the barcode sequences. Each virus batch was checked for quality scores (FastQC) before and after the quality trim (bbduk2.sh). MultiQC (Version 1.14) showed that 1% of the raw reads were incorrect and had adapter contamination. I did quality trimming of raw reads expecting of Q>20. I had tried to use trimmomatic (version 0.39) to trim away the Ilumina adapters and low-quality nucleotides. Another FastQC was done to confirm the sequences quality. I then compared the trimmed result produced by trimmomatic and by bbduk2.sh, I chose to continue using the trimmed raw reads from bbduk2.sh as it had preserved the sequence length albeit more stringent quality scores. Next, I extracted barcodes from the quality trimmed fastq sequences using bbduk2.sh. I trimmed fastq sequences from the right side using a 13-16 nucleotides k-mer allowing for 1-2 mismatches. The sequences were reverse complement to the template, therefore, the k-mer sequences were also reverse complement. The fastq sequences from the left side were trimmed using a 16 nucleotides k-mer allowing for 1-2 mismatches. The sequences were reverse complement to the template, therefore, the k-mer sequences were also reverse complement. Library IDs were extracted with bbduk2.sh. Similarly, library IDs s were trimmed from the left side (lliteral). Then, the extracted library IDs and the extracted barcodes were paired together (pairfq) based on their shared identifiers. The paired barcodes were then clustered using starcode reduction.

Barcodes and Library IDs extraction of tissue from DNA viruses (RAAV-53) and (RAAV-72)

RAAV-53 and RAAV-72 were mRNA sequences that consisted of barcodes and library IDs, which could be derived from three different plasmid library (RAAV-60: p005, p006 or p007). RAAV-53 had 20 samples and RAAV-72 had 8 samples with different reads depth. Similar content of scripts was used in RAAV-60 but modified to handle multiple samples for RAAV-53 and RAAV-72 in each directory. Fastqc data showed that some of the samples had lower quality scores. Hence a more stringent parameters were done using the bbduk2.sh quality trimming for every tissue samples. Each sample in RAAV-53 and RAAV-72 were extracted with specific sequence templates used for barcodes extraction in each plasmid library RAAV-60: p005, p006 and p007. A series of barcode reduction were done for each sample.

Data analysis

This workflow encompasses RNA counting, alignment, normalization, and data visualization across various RAAV datasets (RAAV-60, RAAV-30, RAAV-53, RAAV-72). It included executing RNA counting scripts and performing alignment, barcode, and fragment frequency analyses. Chimeric barcodes were aligned for RAAV-60 samples (p005, p006, p007), with fragment and barcode frequency tables generated. The pipeline calculated alignment percentages. Complete library range objects were created by combining fragments for selected RAAV samples. RNA counting data was saved as .rds files, optimized for large datasets, while library counts were normalized in smaller chunks with iterative read count updates. The workflow also included data visualization, with circular barplots illustrating unique peptides, pairwise and accumulated gene coverage plots. For pairwise gene plot, RNA counts were log2-transformed to stabilize variance. The data was divided into bins representing amino acid (AA) positions across the gene, with each bin containing information on barcode counts (BC). This binning highlighted patterns along the AA sequence for different experimental groups. Finally, the barcode count and corresponding peptide amino acid sequences were input into the Hammock tool (v_1.2.0), [22]. Consensus motifs were visualized using WebLogo (v 0.0.0), and output formatting was handled with GPL Ghostscript 9.18 (2015-10-05) in Bioconda environment.

Results

The output from this workflow included a list of barcodes, their corresponding peptides (nucleotide sequences translated into amino acids), and the position of each peptide within the original protein. Each plasmid library was associated with specific library IDs: p005 used (TCG), p006 used (TAC), and p007 used (CTA). Most of the results presented here were derived from p007 library. For plasmid libraries p005 and p007, 98% of the sequences were aligned with the original sequences established in the lab. Since plasmid library p006 was designed using degenerate NNK, it was not subjected to BLAST analysis.

For plasmid library p005, a total of 113,150,190 reads were processed, identifying 9,202,839 unique original barcodes. After starcode reduction, 5,434,737 unique barcodes remained. Out of these, 2,191,113 barcodes appeared only once, classified as single-read barcodes. Clean multi-read barcodes, which appeared only once from multiple reads, totaled 38,162. There were also 3,205,462 chimeric multi-read barcodes, appearing more than once from multiple reads.

Clean multi-read and Chimeric multi-read barcodes referred to different types of barcodes identified in the sequence analysis based on how often they appeared and their consistency across reads. Clean Multi-Read Barcodes represented multi-read barcodes where all reads (sequences) for a given barcode were consistent. They indicated reliable, non-chimeric sequences, where the reads aligned well without discrepancies. Chimeric Multi-

Read Barcodes appeared multiple times with different read sequences, meaning that the reads associated with a barcode were not fully consistent. This chimerism suggested that these barcodes might have combined sequences, often arising from sequencing or amplification errors.

For plasmid library p006, a total of 20,251,305 reads were processed, resulting in the detection of 1,261,887 unique original barcodes. After starcode reduction, this number decreased to 592,798 unique barcodes. There were 306,521 single-read barcodes. Clean multi-read barcodes amounted to 5,813, and 280,464 chimeric multi-read barcodes were observed. For plasmid library p007, 123,875,291 reads were processed, resulting in the identification of 6,747,020 distinct original barcodes. Following starcode reduction, 3,333,752 unique barcodes remained. Of these, 2,685,026 were single-read barcodes. Clean multi-read barcodes, appearing only once, totaled 14,545, while chimeric multi-read barcodes amounted to 634,181. Data processing steps similar to those applied to plasmid libraries were also performed on viral libraries and tissue samples, though the results are not presented here.

The efficiency of fragment matching and identification could be traced back to the original 294 proteins using the lookup table (LUT) and consensus motifs identified through the Hammock method (Fig. 2). The p006 library contained 340,909 unique peptides, with 18,517 from the virus library, 395 from tissue-infected samples (RAAV-53), and 42 from another tissue-infected sample (RAAV-72) (Fig 3A). The array contained 158,788 entries, created in the lab. Within the p007 library, 117,148 unique peptides were identified. The virus library contained 116,881 unique peptides, with 53,294 detected in infected tissue (RAAV-53) and 85,392 in infected tissue (RAAV-72), as shown in Fig. 3B.

From the *in vivo* screening of tissue-infected samples, the top 25 peptides were identified from proteins associated with multiple barcodes and detected across different samples (Fig 4). Peptides from the far-left region (highest peak) in peptide A were hypothesized to exhibit both highly efficient retrograde transport and local infectivity in the rat brain [3], making them valuable candidates for further testing (Fig 4A, 4B).

A) B)

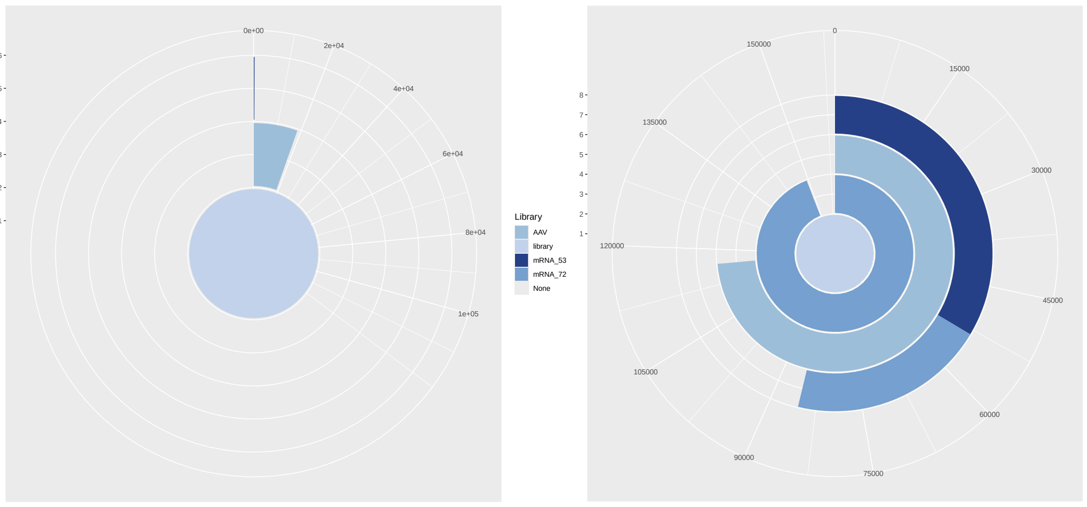


Fig 3. The circular bar plot displays the absolute quantities of unique peptides recovered at each stage of the BRAVE assay. A) p006 library. B) p007 library.

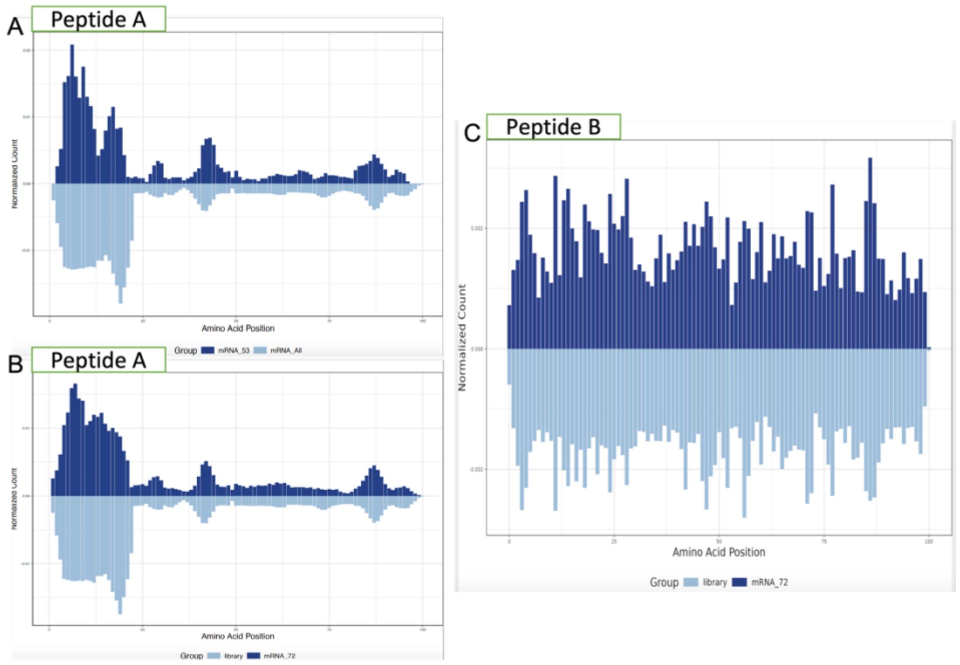


Fig 4. Relative Amino Acid Frequency Distribution. The height of each bar reflects the relative frequency of a specific amino acid, adjusted according to the total library, and aggregated from all peptides spanning that region.

The Hammock approach employed a hidden Markov model (HMM) to detect patterns across the dataset, grouping peptides with similar sequence homology. The y-axis, measured in bits, represented the information content at each position within the motif, indicating positional conservation or variability. Higher information content denoted conserved positions where specific bases or amino acids were preferred, while lower information content signified greater variability. This analysis provided insights into the conservation patterns within the motif and highlights key residues or bases that might be critical for function or recognition (Fig 5).

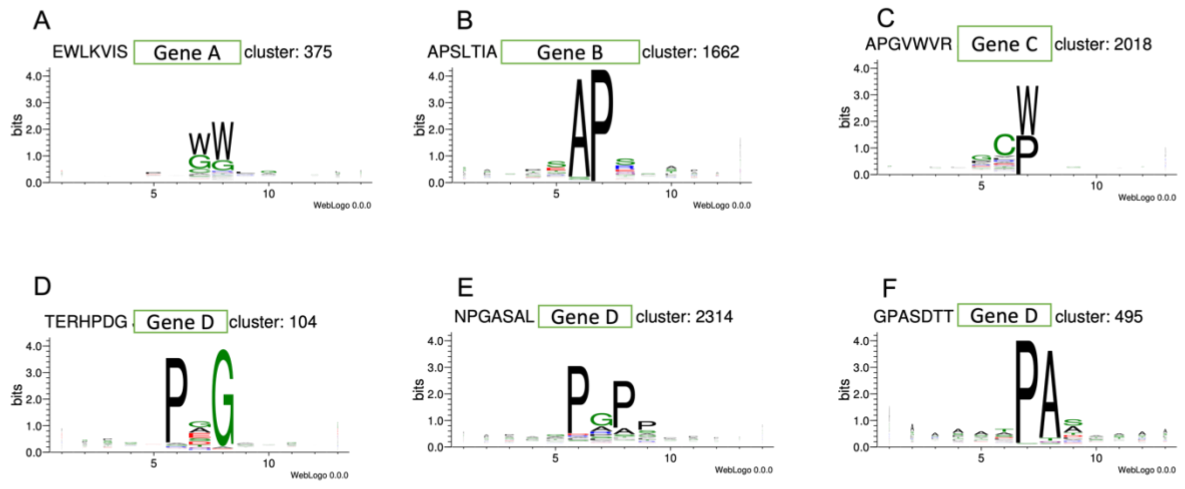


Fig 5. Bitmap plots of consensus motifs generated using the Hammock clustering approach, which utilizes a hidden Markov model (HMM) to align and identify recurring patterns across sequences. (A-C) Consensus motifs of distinct peptides derived from different proteins. (D-F) Consensus motifs of three distinct peptides from the same protein.

Discussion

During the data processing of the p005 and p007 libraries, the lookup table (LUT) required for data analysis, including fragment identification and normalization, consumed significant computer memory, making it challenging to process the entire dataset simultaneously. To improve performance and scalability, chunking - dividing the data into smaller, manageable pieces was applied. These chunks were created based on metadata containing GA alignment objects, simplifying data management and processing. A dynamic chunking strategy was implemented to accommodate varying data sizes [20]. This study utilized multiple programming languages, including Bash, Python, and R, with R used for data analysis and graph plotting. Some scripts incorporated the ‘future.apply’ and ‘plan’ packages in R to enable parallel processing [21], https://msu-icer.github.io/r-for-hpcc/parallelizing-r-code.html. When normalization was conducted, setting the future plan with workers = 8 (using multicore) succeeded, but increasing workers to 32 resulted in an out-of-memory error. This likely occurred because, with workers = 8, memory usage was spread across 8 processes, staying within the capacity of the Lunarc cosmos system's RAM. However, when workers = 32, the memory demands of 32 processes likely exceeded the system's RAM, causing the error (https://future.futureverse.org/reference/plan.html).

Two approaches were used to normalize RNA barcode counts. The first method involved processing smaller data chunks in a loop, updating the read count with each iteration before moving to the next chunk. The second method processed smaller chunks, saving each as a combined .rds file, and then proceeded with filtering out entries with an RNA count of 1 before performing normalization. Looping through chunks and processing smaller segments could reduce memory usage while still providing normalized counts. Saving intermediate results in files and filtering low-count entries before normalization, as in my second approach, could also reduce data load while improving normalization accuracy for downstream analysis.

When running Hammock, the error StringIndexOutOfBoundsException: String index out of range: 15 occurred. However, after setting the parameter --greedy_threshold = 15, the program executed smoothly. In the context of greedy clustering, this threshold generally controlled the allowable similarity or distance between sequences in the same cluster. A threshold of 15 likely specified that sequences within a cluster must meet a similarity standard, where differences between them did not exceed 15 units. This could represent a maximum allowed mismatch, an edit distance limit, or even a similarity cutoff of 85% (if interpreted as 100 - 15). Setting this threshold defined how loosely or strictly sequences were grouped, with lower values requiring closer similarity (more stringent clustering) and higher values allowing for more variation within clusters [22]. Despite technical challenges, the BRAVE strategy successfully identified peptide fragments associated with unique barcodes in targeted infected tissue samples.

The efficient recovery of barcodes from virally expressed mRNA ensured that only capsid variants successfully completing all critical steps of AAV infectivity. This readout modality was crucial, as certain capsid engineering versions might result in capsid sequestration on the cell surface, blocking internalization and causing misleading readouts. The BRAVE approach effectively filtered out such variants. Additionally, extracting barcodes from DNA in the injected library could provide insight into which capsid variants became sequestered in tissue (either before or after internalization and uncoating) without leading to successful transgene expression.

An added advantage of this method was the use of a stable bacterial library combined with barcode oversampling, enabling efficient error control during *in vivo* screening. This control was achieved through vector-linked peptide transcript identification on unique barcode counts (rather than mRNA expression levels) and the number of samples showing the same readout for each peptide.

The main limitation was the relatively small size and diversity of the libraries compared to those generated by methods like capsid shuffling, degenerate primers, or error-prone PCR. However, this limitation was balanced by the advantages of rational design, which ensured that only potentially functional sequences, without repetition or bias, were included in the library. Although this approach did not capture the full diversity of a completely random method, the increasing capacity of synthesis (currently in the range of 10^6) and decreasing costs continue to expand its potential applications.

Conclusion

AAV vector-based therapies are rapidly emerging as a promising approach for treating or even preventing neurological diseases. AAV gene therapy is particularly effective for well-validated CNS targets, especially those genetic targets that are difficult to address with conventional treatments. Clinical trials are currently assessing AAV vectors for delivering therapeutic proteins to the CNS. One key advantage of AAV gene transfer is its durability in non-dividing cells like neurons after a single administration, which is beneficial in scenarios where repeat dosing may be challenging or impractical.

The most critical factor in unlocking the full potential of AAV gene therapy for neurological diseases is effective delivery, largely determined by the capsid, which controls the specific tissues and cell types transduced. Each administration route, along with the chosen capsid, has unique characteristics and considerations, including dose requirements. It is unlikely that a single administration route will suit all CNS applications; rather, the optimal route for a given target and disease will depend on achieving the desired target distribution while minimizing exposure to non-target tissues [2,3].

In summary, recent progress in harnessing the potential of AAV gene therapy for neurological disorders has sparked significant interest and effort in this field. As improvements in AAV capsid optimization, vector delivery, and transgene design advance, the range of CNS disease applications and targets is expected to grow considerably.

Acknowledgments

I would like to express my gratitude to my supervisor, Marcus Davidsson, from the Company of rAAVen Therapeutics at Medicon Village, for his support and guidance.

Reference

1. Hocquemiller M, Giersch L, Audrain M, Parker S, Cartier N. Adeno-Associated Virus-Based Gene Therapy for CNS Diseases. Human Gene Therapy. 2016;27: 478–496. doi:10.1089/hum.2016.087

2. Deverman BE, Ravina BM, Bankiewicz KS, Paul SM, Sah DWY. Gene therapy for neurological disorders: progress and prospects. Nat Rev Drug Discov. 2018;17: 641–659. doi:10.1038/nrd.2018.110

3. Davidsson M, Wang G, Aldrin-Kirk P, Cardoso T, Nolbrant S, Hartnor M, et al. A systematic capsid evolution approach performed in vivo for the design of AAV vectors with tailored properties and tropism. Proc Natl Acad Sci USA. 2019;116: 27053–27062. doi:10.1073/pnas.1910061116

4. McCarty DM, Monahan PE, Samulski RJ. Self-complementary recombinant adeno-associated virus (scAAV) vectors promote efficient transduction independently of DNA synthesis. Gene Ther. 2001;8: 1248–1254. doi:10.1038/sj.gt.3301514

5. McCarty DM, Fu H, Monahan PE, Toulson CE, Naik P, Samulski RJ. Adeno-associated virus terminal repeat (TR) mutant generates self-complementary vectors to overcome the rate-limiting step to transduction in vivo. Gene Ther. 2003;10: 2112–2118. doi:10.1038/sj.gt.3302134

6. Srivastava A, Lusby EW, Berns KI. Nucleotide sequence and organization of the adeno-associated virus 2 genome. J Virol. 1983;45: 555–564. doi:10.1128/jvi.45.2.555-564.1983

7. Lopez-Gordo E, Chamberlain K, Riyad JM, Kohlbrenner E, Weber T. Natural Adeno-Associated Virus Serotypes and Engineered Adeno-Associated Virus Capsid Variants: Tropism Differences and Mechanistic Insights. Viruses. 2024;16: 442. doi:10.3390/v16030442

8. Pupo A, Fernández A, Low SH, François A, Suárez-Amarán L, Samulski RJ. AAV vectors: The Rubik's cube of human gene therapy. Molecular Therapy. 2022;30: 3515–3541. doi:10.1016/j.ymthe.2022.09.015

9. Girod A, Ried M, Wobus C, Lahm H, Leike K, Kleinschmidt J, et al. Genetic capsid modifications allow efficient re-targeting of adeno-associated virus type 2. Nat Med. 1999;5: 1052–1056. doi:10.1038/12491

10. Bennett A, Mietzsch M, Agbandje-McKenna M. Understanding capsid assembly and genome packaging for adeno-associated viruses. Future virology. 2017;12: 283. doi:10.2217/fvl-2017-0011

11. Colón-Thillet R, Jerome KR, Stone D. Optimization of AAV vectors to target persistent viral reservoirs. Virology Journal. 2021;18: 85. doi:10.1186/s12985-021-01555-7

12. Naik SH, Perié L, Swart E, Gerlach C, van Rooij N, de Boer RJ, et al. Diverse and heritable lineage imprinting of early haematopoietic progenitors. Nature. 2013;496: 229–232. doi:10.1038/nature12013

13. Davidsson M, Diaz-Fernandez P, Schwich OD, Torroba M, Wang G, Björklund T. A novel process of viral vector barcoding and library preparation enables high-diversity library generation and recombination-free paired-end sequencing. Sci Rep. 2016;6: 37563. doi:10.1038/srep37563

14. Blundell JR, Levy SF. Beyond genome sequencing: Lineage tracking with barcodes to study the dynamics of evolution, infection, and cancer. Genomics. 2014;104: 417–430. doi:10.1016/j.ygeno.2014.09.005

15. Siloto RMP, Weselake RJ. Site saturation mutagenesis: Methods and applications in protein engineering. Biocatalysis and Agricultural Biotechnology. 2012;1: 181–189. doi:10.1016/j.bcab.2012.03.010

16. Bushnell B. BBMap: A Fast, Accurate, Splice-Aware Aligner. 2014 [cited 16 Oct 2024]. Available: https://escholarship.org/uc/item/1h3515gn

17. Zorita E, Cuscó P, Filion GJ. Starcode: sequence clustering based on all-pairs search. Bioinformatics. 2015;31: 1913–1919. doi:10.1093/bioinformatics/btv053

18. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM. 2008;51: 107–113. doi:10.1145/1327452.1327492

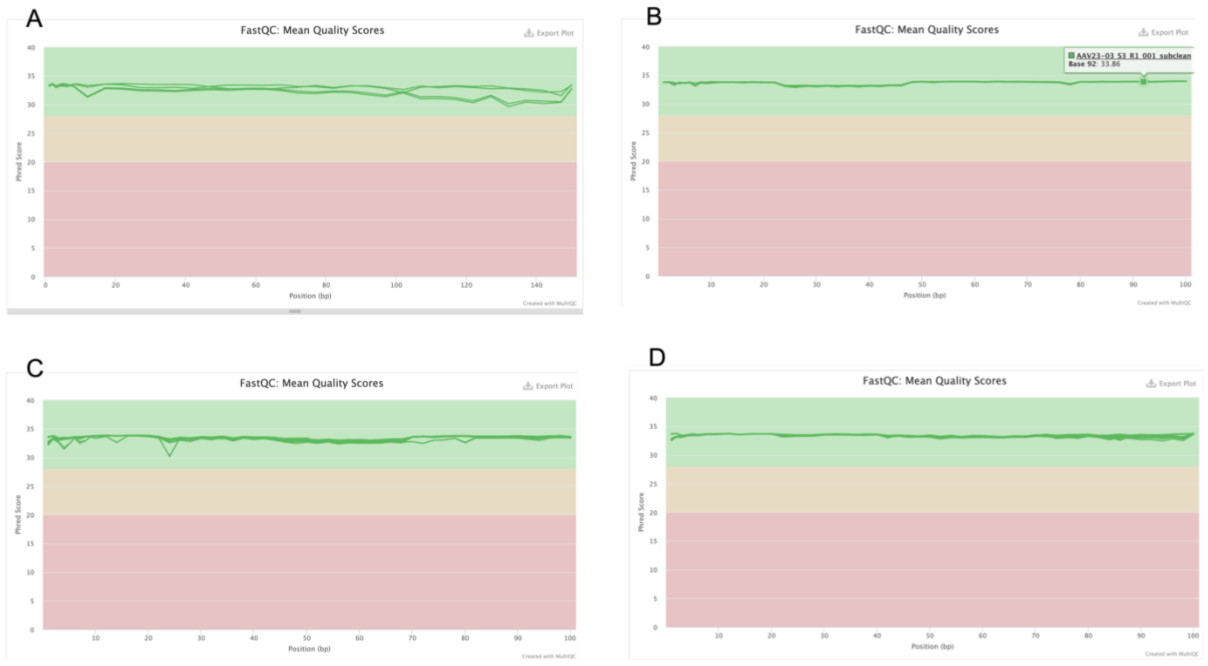
19. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, et al. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 2010;20: 1297–1303. doi:10.1101/gr.107524.110

20. Kumar P, Paul RK, Roy HS, Yeasin Md, Ajit, Paul AK. Big Data Analysis in Computational Biology and Bioinformatics. In: Mandal S, editor. Reverse Engineering of Regulatory Networks. New York, NY: Springer US; 2024. pp. 181–197. doi:10.1007/978-1-0716-3461-5_11

21. Bengtsson H. future.apply: Apply Function to Elements in Parallel using Futures. 2018. p. 1.11.3. doi:10.32614/CRAN.package.future.apply

22. Krejci A, Hupp TR, Lexa M, Vojtesek B, Muller P. Hammock: a hidden Markov model-based peptide clustering algorithm to identify protein-interaction consensus motifs in large datasets. Bioinformatics. 2016;32: 9–16. doi:10.1093/bioinformatics/btv522

Supplementary data



Supplementary Fig 1. MultiQC after quality trimming. A) RAAV-60, B) RAAV-30, C) RAAV-53, D) RAAV-72.