Chapter 20

Creation and Analysis of a Virome: Using CRISPR Spacers

Michelle Davison and Devaki Bhaya

Abstract

Advances in sequencing technology have allowed for the study of complex and previously unexplored microbial and viral populations; however, linking host–phage partners using in silico techniques has been challenging. Here, we describe the flow-through for creation of a virome, and its subsequent analysis with the viral assembly and analysis module "Viritas," which we have recently developed. This module allows for binning of contigs based on tetranucleotide frequencies, putative phage-host partner identification by CRISPR spacer matching, and identification of ORFs.

Key words CRISPR-Cas, Next-generation sequencing, Viral genome assembly, Metagenomics, Phage annotation, Bioinformatics

1 Introduction

Pioneering work by several groups [1–4] using high-throughput sequencing technologies has provided a glimpse of microbial and viral diversity in a range of different environments. However, the lack of fully assembled viral and corresponding host genomes has limited the analysis of such data. Despite recent progress in handling regions of uneven coverage [5–7], tackling complex populations [8–10], development of comprehensive metagenomic analysis pipelines [11, 12], as well as ORF identification and calling in viromes [13], robust assembly and analysis of viral sequence data remains a significant obstacle.

For viruses to initiate successful infections, they must mutate to evade the host CRISPR defense system, which relies on a close match between acquired spacer and incoming viral sequence [13, 14]. Because new spacers are being acquired into host CRISPR arrays they are useful markers, both for the analysis of host–phage relationships, and to provide a time-line of past viral infections [15]. The spacers of the CRISPR-Cas adaptive immunity system, therefore, can also be a tool that can be utilized to "bin" novel viral

Magnus Lundgren et al. (eds.), *CRISPR: Methods and Protocols*, Methods in Molecular Biology, vol. 1311,
DOI 10.1007/978-1-4939-2687-9_20,

sequences. [13]. We recently investigated viral populations in the hot spring polymicrobial mat community of Yellowstone National Park. By using CRISPR spacers identified in full genome sequences, as well as from metagenomic reads, we were able to successfully bin and identify potential viral-host partners [13, 16]. The processes to perform these analyses are described in this chapter.

2 Materials

1. 1× Tris–EDTA: 10 mM Tris–HCl, 1 mM EDTA, pH 8.0.
2. Nitrocellulose filters, 0.45 μm and 0.2 μm (Nalgene, Thermo Scientific).
3. Microfuge Tube Polyallomer 1.5 mL ultramicrocentrifuge tubes.
4. Illustra GenomiPhi V2 kit, GE.
5. Beckman TL-100 Ultracentrifuge, TLA 100.3.
6. 10× PCR buffer (Qiagen).

3 Methods

3.1 Creation of a Virome

3.1.1 Enrichment of Viral Particles from an Environmental Sample of Interest

Resuspend microbial mat sample in 50 mL 1× Tris–EDTA by vigorous vortexing (*see* **Note 1**). Pellet down intact cells and cellular debris at 6,000 rpm for 10 min in a Sorvall GS5C. Aspirate the supernatant, and pass sequentially through 0.45 μm and 0.2 μm filters to remove any remaining cells and/or cellular debris. Aliquot filtered supernatant into 1.5 mL microfuge tubes and ultracentrifuge at 50,000 rpm or 1 h for concentrate viral particles. Carefully remove all but 5–10 μL of supernatant. Use 1 μL of enrichment as template for amplification reaction with ϕ29 polymerase as per kit instructions (Illustra GenomiPhi V2 kit; *see* **Note 2**) (Fig. 1).

3.1.2 Qualitative Determination of Bacterial and Viral Ratios in Amplified Viral Sample

Assemble 25 μL PCRs with the following 8× Mastermix recipe: 20 μL 10× PCR Buffer, 111.0 μL of double distilled water, 10.0 μL of DMSO, 16.0 μL dNTP (2.5 mM stock), 20.0 μL forward primer, and 20.0 μL reverse universal bacterial 16S primers [27], 1.0 μL of ϕ29 amplified viral template reaction and 2.0 μL of Taq Polymerase. Visualize PCRs by gel electrophoresis in 0.8 % agarose with ethidium bromide (0.0002 μg/μL) run at 100 mV for 20 min.

3.1.3 Sequencing of Viral Enrichment

Pool duplicate ϕ29 amplified viral templates and carry out appropriate Next-Generation Sequencing (e.g., 454 sequencing, Illumina, PacBio) based on read length needs, acceptable error rates, and coverage depth requirements (*see* **Note 3**).

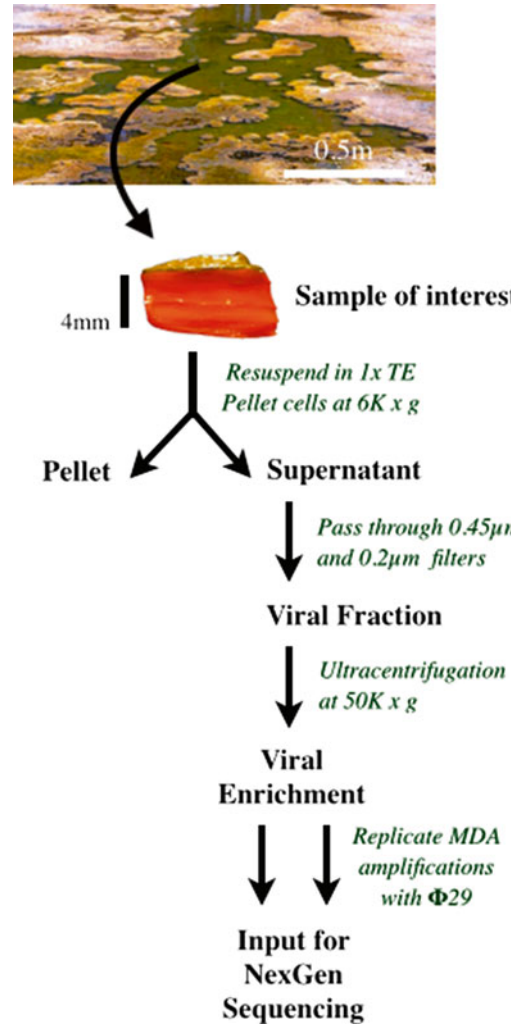

Fig. 1 Schematic workflow of viral sample preparation for next-generation sequencing

3.2 Identification of CRISPR Spacers

3.2.1 Identification of CRISPR Spacers Present in Metagenome Reads and Extraction of Novel CRISPR Spacers

To identify reads containing a CRISPR array, pipeline metagenome reads (either from publically available datasets, or optimally a metagenome generated in tandem from the host of interest) through CRISPRFinder to extract both spacers and repeats, then subject repeats to a BLASTN (e-value $<=10^{-5}$) against the nr database to determine the species from which they originated [17–19] (*see* **Note 4**). CRISPR spacer identification can be done with either raw or assembled reads, however, when performed with assembled reads some resolution will be lost as spacers identified will be consensus sequences.

3.3 Assembly of Viral Reads

3.3.1 Preprocessing of Reads for Viral Assembly

To remove amplification artifacts and low quality/complexity reads from 454 data, use the program prinseq to preprocess viral reads prior to assembly [20]. Reads smaller than 50 bp should be filtered out before further analysis (*see* **Note 5**).

3.3.2 Read Filtration (Removing Host/ Bacterial DNA)

To find and filter out all hits to known bacterial genomes, and recruit reads to known phage sequences, use NUCmer [21] with the following parameters: --maxmatch –c 30 –l 30 --coords).

3.3.3 Genome Assembly

To assemble genomes, firstly, reads confidently classified by FCP (e-value <=10^{-5}) as bacterial or archaea should be removed. Then, run an assembly with Newbler version 2.5p1 (params: -minlen 45 -m -cpu 16 -mi 90 -ml 40 -ud -l 1000 -g -e 3 -a 300 -large) (*see* **Note 6**).

3.3.4 Celera Assembler Virome Assembly

The Celera Assembler (Celera Genomics, 1999, tuned for metagenomic assembly, utgErrorRate = 0.030, utgBubblePopping = 1, cgwDistanceSampleSize = 10, doToggle = 1, toggleNumInstances = 0, toggleUnitigLength = 1000) should be run on the dataset, for verification of the Newbler output by identifying contigs in disagreement, in addition to improving the virome assembly in terms of contig length and fragmentation.

3.3.5 Phage Annotation Pipeline

The phage assembly module, Viritas, exists within the metagenomic assembly pipeline MetAMOS [10] to assemble, bin, annotate and characterize viral sequencing data. The pipeline consists of MetaGeneMark [21] for ORF prediction, PHMMER [22, 28] for homology detection using an annotated phage protein DB (Phantome & EBI), Repeatoire for internal repeat misassembly identification, NUCmer/PROmer for spacer hits to known CRISPR spacers, and tRNAscan for tRNAs [23] frequently found in phages. The specific python command lines for each program are detailed below:

MetaGeneMark

/gmhmmp -o %s.orfs -m MetaGeneMark_v1.mod

tRNAscan

tRNAscan-SE -o trna.out –B

HMMER 3

phmmer --cpu 10 -E 0.1 -o %s.phm.out --tblout %s.phm.tbl --notextw %s.faa ./DBS/allprots.faa"%(prefix,prefix,prefix)

REPEATOIRE

./repeatoire --minreplen=20 --z=11 --extend=0 --allow-redundant=0 --sequence=t1.fna --output=reps.out >& test.out

PROMER

promer --maxmatch -c 4 -l 4 --coords t1.fna ./CRISPR/INPUTCRISPRs.fasta > t1.out 2> t2.err

promer --maxmatch -c 4 -l 4 --coords t1.fna ./CRISPR/INPUTCRISPRhits.fasta > t1.out 2> t2.err

show-coords -I 85 -o -k -L 30 -T -c -r out.delta > out.coords

3.3.6 Tetranucleotide Analysis

To group contigs based on tetranucleotide frequency, the following python script was used on the assembled contigs over 1 kb in length:

```
#count_tetramers.py
#takes an assembly as input, calculates tetramer frequencies stepwise across each contig
#reports frequences in *.tetra output file and provides OBS/EXP values for each tetramer
 import os,sys,string,operator,math
#create tetramer hash
def getTetramerDict():
    tdict = {}
    nts = ('A','G','C','T')
    for c1 in nts:
        for c2 in nts:
            for c3 in nts:
                for c4 in nts:
                    tmer = c1+c2+c3+c4
                    ids = [tmer,getRC(tmer)]
                    ids = sorted(sorted(ids), key=str.upper)
                    tdict[ids[0]] = 0
    #print len(tdict.keys())
    return tdict
#get reverse complement
def getRC(tmer):
    rcs = ""
    for nt in tmer:
        if nt == "T":
            rcs += "A"
        elif nt == "A":
            rcs += "T"
        elif nt == "G":
            rcs += "C"
        elif nt == "C":
            rcs += "G"
        elif nt == "N":
            rcs += "N"
    if len(rcs) != 4:
        print "problem in tmer size"
        sys.exit(1)
    rcs = rcs[::-1]
    return rcs
verbose = False
try:
    f = open(sys.argv[1],'r')
except IndexError:
    print "usage: calc_tetramers.py contigs.fa [verbose=0]"
    sys.exit(0)
```

```
except IOError:
    print "Input file not found!"
    sys.exit(0)
try:
    verbose = int(sys.argv[2])
except IndexError:
    pass
except TypeError:
    pass
data = f.read()
data = data.split(">")[1:]
outf = open(sys.argv[1]+".tetra",'w')
outf.write("ContigID,")
cdict = getTetramerDict()
for tmer in cdict.keys()[:-1]:
    outf.write(tmer+",")
outf.write(cdict.keys()[-1]+"\n")
nts = ("A","G","C","T")
top_mers = {}
for seq in data:
    hdr,ctg = seq.split("\n",1)
    outf.write(hdr.replace("\n","").split("\t")[0].
    split(" ")[0]+",")
    ctg = ctg.replace("\n","")
    ctg = string.upper(ctg)
    s1 = 0
    s2 = s1+4
    tdict = getTetramerDict()
        while s1+4 < len(ctg):
        tmer = ctg[s1:s2]
        tmer = string.upper(tmer)
        if "N" in tmer:
            s1 +=1
            s2 +=1
            continue
        notok = 0
        for char in tmer:
            if char not in nts:
                notok = 1
                break
        if notok:
            s1 +=1
            s2 +=1
            continue
            try:
            tdict[tmer]
        except KeyError:
            tmer = getRC(tmer)
```

```
    tdict[tmer] +=1
    s1 +=1
    s2 +=1
    sum = 0.0
  for tmer in tdict.keys()[:-1]:
    #outf.write("%.5f"%((float(tdict[tmer])*4.0)/
    float(len(ctg)))+",")
    if len(ctg)-4 > 0:
      outf.write("%.5f"%((float(tdict[tmer]))/
      float(len(ctg)-4))+",")
      sum += (float(tdict[tmer]))/float(len(ctg)-4)
  tmer = tdict.keys()[-1]
  #outf.write("%.5f"%((float(tdict[tmer])*4.0)/
  float(len(ctg)))+"\n")
  if len(ctg)-4 > 0:
    outf.write("%.5f"%((float(tdict[tmer]))/
    float(len(ctg)-4))+"\n")
    sum += (float(tdict[tmer]))/float(len(ctg)-4)
  #calc background GC freq
  acnt = ctg.count("A")
  tcnt = ctg.count("T")
  ccnt = ctg.count("C")
  gcnt = ctg.count("G")
  afreq = float(acnt)/float(len(ctg))
  gfreq = float(gcnt)/float(len(ctg))
  cfreq = float(ccnt)/float(len(ctg))
  tfreq = float(tcnt)/float(len(ctg))
  if not verbose:
    continue
  #only run this if verbose
  print hdr
  sorted_x = sorted(tdict.iteritems(), key=operator.itemgetter
  (1))
  sorted_x.reverse()
  print "Tetramer\tCount\tObs/Exp"
  for tup in sorted_x[0:5]:
    if tup[1] > 0:
                nfreqs = 1
      for nt in tup[0]:
        if nt == "A":
          nfreqs *= afreq
        elif nt == "T":
          nfreqs *= tfreq
        elif nt == "G":
          nfreqs *= gfreq
        elif nt == "C":
          nfreqs *= cfreq
      exp = nfreqs*(len(ctg)-4)
```

```
            if exp < 1:
                exp = 1
            #nfreqs = afreq*gfreq*cfreq*tfreq
                    printtup[0]+"\t"+str(tup[1])+"\t%f"%(float(tup[1])/
                    (exp)),
            if float(tup[1])/(exp) > 2:
                print "*"
            else:
                print
    outf.close()
    print "done! tetramer freqs found in: ", sys.argv[1]+".tetra"
```

Contigs less than 1 kb in length often result in "noisy" signatures and should be excluded from further analysis. Use the gplots heatmap.2 R function to visualize frequencies based on a hierarchical clustering of tetranucleotide frequency of the assembled contigs [24].

3.4 Mapping of CRISPR Spacers onto Viral Reads

Identified CRISPR spacers are mapped back to viral reads with BLASTN. To be considered a 'putative' match, a spacer had to have at least 85 % identity spanning 70 % of the spacer length [13]. Exclude any reads containing host CRISPR arrays from this BLAST, to prevent artificial inflation of CRISPR hit numbers from spacers matching with 100 % identity to themselves.

3.5 Conclusion

We have outlined a method by which viromes can be analyzed, using a multitiered approach with broad applicability. Novel virome sequence can be conservatively assembled with the metAMOS pipeline, similar contigs can then be further clustered based on tetranucleotide signatures, and potential viral-host relationships elucidated by using CRISPR-spacer matching.

4 Notes

1. This particular protocol was optimized for use with the microbial mat communities of Yellowstone National Park; however, it can be tailored to suit any sample of interest. Volume of 1× TE should be enough liquid to thoroughly resuspend sample; keep in mind some liquid will be ultimately be lost in the filtering step. For example, we used 50 mL to resuspend an 8 mm circular plug 2 mm deep.
2. A DNase step prior to amplification was intentionally omitted in this case, as it was unknown if the viral particles were intact. Use of DNase on damaged particles can result in complete loss of viral DNA [25].
3. Multiple replicates are highly advised to mediate the effect of ϕ29 polymerase bias in multiple displacement amplification (MDA) reactions [26].

4. CRISPRFinder: Always manually curate identified spacers to remove any spurious spacer calls, such as repeat-rich sequences, that are not associated with CRISPR loci.
5. All assembly parameters described were optimized for use with Roche 454 Titanium sequencing generated reads.
6. By stringently filtering out all hits to known bacterial and archaeal genomes phage, and prophage genes contained on host genomes will be excluded.

Acknowledgments

DB acknowledges funding support from the NSF (MCB#1024755) and the Carnegie Institution of Science. This protocol is a modification of the Viritas pipeline described in Davison et al (in prep) which was carried out in collaboration with Mihai Pop, a the University of Maryland and his group. Viritas has been incorporated into the metAMOS pipeline (https://github.com/marbl/metAMOS).

References

1. Emerson JB, Thomas BC, Andrade K, Heidelberg KB, Banfield JF (2013) New approaches indicate constant viral diversity despite shifts in assemblage structure in an Australian hypersaline lake. Appl Environ Microbiol 79.21(2013):6755–6764
2. Pride DT, Schoenfeld T (2008) Genome signature analysis of thermal virus metagenomes reveals Archaea and thermophilic signatures. BMC Genomics 9:420
3. Fancello L, Raoult D, Desnues C (2012) Computational tools for viral metagenomics and their application in clinical research. Virology 434:162–174
4. Sullivan MB, Huang KH, Ignacio-Espinoza JC, Berlin AM, Kelly L, Weigele PR, DeFrancesco AS, Kern SE, Thompson LR, Young S et al (2010) Genomic analysis of oceanic cyanobacterial myoviruses compared with T4-like myoviruses from diverse hosts and environments. Environ Microbiol 12: 3035–3056
5. Chitsaz H, Yee-Greenbaum JL, Tesler G, Lombardo M-J, Dupont CL, Badger JH, Novotny M, Rusch DB, Fraser LJ, Gormley NA et al (2011) Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. Nat Biotech 29:915–921
6. Peng Y, Leung HC, Yiu S, Chin FY (2012) IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics 28:1420–1428
7. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, Lesin VM, Nikolenko SI, Pham S, Prjibelski AD, Pyshkin AV, Sirotkin AV, Vyahhi N, Tesler G, Alekseyev MA, Pevzner PA (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol 19:455–477
8. Boisvert S, Raymond F, Godzaridis E, Laviolette F, Corbeil J (2012) Ray Meta: scalable de novo metagenome assembly and profiling. Genome Biol 13:R122
9. Namiki T, Hachiya T, Tanaka H, Sakakibara Y (2011) MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. In: Proceedings of the 2nd ACM conference on bioinformatics, computational biology and biomedicine. pp. 116–124. Chicago, Illinois: ACM; 2011
10. Peng Y, Leung HCM, Yiu SM, Chin FYL (2011) Meta-IDBA: a de novo assembler for metagenomic data. Bioinformatics 27:i94–i101
11. Treangen T, Koren S, Astrovskaya I, Sommer D, Liu B, Pop M (2011) MetAMOS: a metagenomic assembly and analysis pipeline for AMOS. Genome Biol 12:P25
12. Kultima JR, Sunagawa S, Li J, Chen W, Chen H, Mende DR, Arumugam M, Pan Q,

Liu B, Qin J et al (2012) MOCAT: a metagenomics assembly and gene prediction toolkit. PLoS One 7:e47656

13. Heidelberg JF, Nelson WC, Schoenfeld T, Bhaya D (2009) Germ warfare in a microbial Mat community: CRISPRs provide insights into the Co-evolution of host and viral genomes. PLoS One 4:e4169
14. Deveau H, Barrangou R, Garneau JE, Labonte J, Fremaux C (2008) Phage response to CRISPR-encoded resistance in *Streptococcus thermophilus*. J Bacteriol 190:1390
15. Bhaya D, Davison M, Barrangou R (2011) CRISPR-Cas systems in bacteria and archaea: versatile small RNAs for adaptive defense and regulation. Annu Rev Genet 45:273–297
16. Davison M, Treangen TJ, Koren S, Gosrani S, Pop M, Bhaya, D. Analysis of virome diversity in a polymicrobial community *Manuscript submitted for publication*.
17. Bhaya D, Grossman AR, Steunou A-S, Khuri N, Cohan FM, Hamamura N, Melendrez MC, Bateson MM, Ward DM, Heidelberg JF (2007) Population level functional diversity in a microbial community revealed by comparative genomic and metagenomic analyses. ISME J 1:703–713
18. Grissa I, Vergnaud G, Pourcel C (2007) The CRISPRdb database and tools to display CRISPRs and to generate dictionaries of spacers and repeats. BMC Bioinformat 8:172
19. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215:403–410
20. Schmieder R, Edwards R (2011) Quality control and preprocessing of metagenomic datasets. Bioinformatics 27:863–864
21. Kurtz S, Phillippy A, Delcher A, Smoot M, Shumway M, Antonescu C, Salzberg S (2004) Versatile and open software for comparing large genomes. Genome Biol 5:R12
22. Zhu W, Lomsadze A, Borodovsky M (2010) Ab initio gene identification in metagenomic sequences. Nucleic Acids Res 38:e132–e132
23. Lowe TM, Eddy SR (1997) TRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. Nucleic Acids Res 25:0955–0964
24. Langfelder P, Horvath S (2008) WGCNA: an R package for weighted correlation network analysis. BMC Bioinformat 9:559
25. Emerson JB, Thomas BC, Andrade K, Allen EE, Heidelberg KB, Banfield JF (2012) Dynamic viral populations in hypersaline systems as revealed by metagenomic assembly. Appl Environ Microbiol 78:6309–6320
26. Ballantyne KN, van Oorschot RAH, Muharam I, van Daal A, John Mitchell R (2007) Decreasing amplification bias associated with multiple displacement amplification and short tandem repeat genotyping. Anal Biochem 368:222–229
27. Sundquist A, Bigdeli S, Jalili R, Druzin M, Waller S, Pullen K, El-Sayed Y, Taslimi MM, Batzoglou S, Ronaghi M (2007) Bacterial flora-typing with targeted, chip-based Pyrosequencing. BMC Microbiol 7:108
28. Eddy SR (2011) Accelerated profile HMM searches. PLoS Comput Biol 7:e1002195