Understanding Genetic Variation in the Repetitive Sequence Content of *Drosophila melanogaster*

Ellison, Christopher E. Ph.D ❋❖ & Rele, Chinmay P. ❋◇

ABSTRACT

Between 40% and 70% of eukaryotic genomes are composed of repetitive sequence, which influences genome organisation, gene expression, and genome evolution. We chose to use the Drosophila Genome Reference Panel (DGRP), a collection of strains of Drosophila melanogaster caught in the wild and highly inbred in our analysis as hey have priorly been sequenced. We needed to identify high repeat regions and identify how exactly they interact with the rest of the genome. We still don't know how the repetitive portions of the genome vary among individuals as we cannot accurately align the short Illumina reads to the repetitive parts of a genome assembly. We found that the copy number of a lot of genomic repeats correlated with the copy number of mtDNA. Most of these pairs were highly positively correlated with each other, while a few had negative correlation.

❋ *Department of Genetics; Rutgers University, New Brunswick*
❖ *Assistant Professor*
◇ *Undergraduate Student*

For documents and more data, please contact chinmay.rele@rutgers.edu

INTRODUCTION

REPEAT ELEMENTS

Repeat sequences are abundant in every genome *(de Koning et al. 2011)*. They can be beneficial from those such as the poly-A tail at the end of genes to prevent mRNA degradation *(Shapiro and von Sternberg 2005)*, to micro-satellites, which have been characterised with protein-encoding genes *(Vieira 2016)*.

Between 40% and 70% of eukaryotic genomes are composed of repetitive sequence, which influences genome organisation, gene expression, and genome evolution *(Biscotti et al. 2015)*. There are three main type of repetitive sequences: (1) terminal repeats, (2) tandem repeats, and (3) interspersed repeats. Terminal repeats exist in the form of telomeric repeat regions, which prevent degrading the chromosome *(O'Sullivan and Karlseder 2010)*. Tandem repeats are those that have been extended during replication, which can be the cause of many genetic diseases such as ASD. They may also fracture certain chromosomes due to hyper-long repeat expansion *(Sutherland and Richards 1995)*.

TRANSPOSONS

By far, the most interesting form of repeat elements are transposable. Transposable Elements (TEs) or transposons were first identified by Barbara McClintock in the mid 1950s in maize *(McClintock 1956)*. They move in two fashions — by a Copy/Paste mechanism or by a Cut/Paste mechanism. In the former, Class I transposons, or retrotransposons, move via an RNA intermediate. They are first transcribed into an RNA, which is then reverse transcribed into DNA, which is then inserted into the genome at a new position *(Corces and Geyer 1991)*. Class II elements, DNA transposons have a Cut/Paste locomotive mechanism, i.e., they do not require an RNA intermediate. They are excised via a transposase, which binds to the local inverted repeats, excises the copy, and relocates it to another genomic locus *(Muñoz-López and García-Pérez 2010)*.

It is obvious by the mechanisms that Class I transposons double in copy number every time they move, whereas those Class II TEs transpose via a non-replicative mechanism, they must increase copy number based on indirect mechanisms that rely on host machinery *(Feschotte and Pritham 2007)*. TEs are usually much longer than simple repeat elements, and thus, need to be identified in a different way.

SEQUENCING PRACTICES

The most common form of sequencing is Illumina sequencing. It consists of taking many short reads, and aligning them to a reference genome in order to make small contig sequences with a high amount of overlap or coverage *(Quail et al. 2012)*. There are two wants of forming the genome of the organism from here: (1) by aligning each contig to a reference sequence (allowing for some error rate due to SNPs or other small mutations); or (2) by aligning the contigs to themselves to form the genomic sequence. However, both of these approaches cannot properly align repetitive sequences that are larger than the read length of Illumina (about 300bp) *(Nakamura et al. 2011)*. These repetitive sequences get clustered to a single locus and knowing their true location on the 2D chromosome is almost impossible solely using Illumina sequencing.

Oxford Nanopore's MinION™, seems to overcome this problem. It has much longer read lengths (maxing out at about 15kb)*(Jain et al. 2016)*, but also has problems of its own. It tends to have a much higher error rate when calling bases. So, this is a method better suited for calling where the repeat elements are, and not the particular sequence of repeat elements (which can be sequenced properly using Illumina).

VARIATION ANALYSIS

Between 40% and 70% of eukaryotic genomes are composed of repetitive sequence, which influences genome organisation, gene expression, and genome evolution. Repetitive sequences can be in the form of repetitive genomic structures such as terminal repeats, tandem repeats or interspersed repeats. The latter differ from tandem repeats in that instead of coming directly after one another, they are non-adjacently dispersed throughout the genome. Most non-adjacent repeat sequences are in the form of TEs, as their primary function is to replicate and move around the genome in a pseudo-random fashion.

We chose to use the Drosophila Genome Reference Panel (DGRP), a collection of strains of Drosophila melanogaster caught in the wild and highly inbred in our analysis as hey have priorly been sequenced.

Illumina sequencing is the most common and cheapest form of sequencing available. However, despite it having high coverage values, making it ideal for SNP analysis, it suffers from having short read lengths, and thus, reads of repeat elements cannot be properly aligned to a contig and are usually ignored from genomic analyses. Understanding how these sequences vary at the population level is important for understanding their evolutionary dynamics.

RESULTS and DISCUSSION

R.1 HEATMAP

We found a series of repeat elements that were highly correlated with each other. These included the telomeric repeats, ribosomal DNA, satellite sequences (with very high correlation of Responder elements), and 1.688 Satellite sequences.

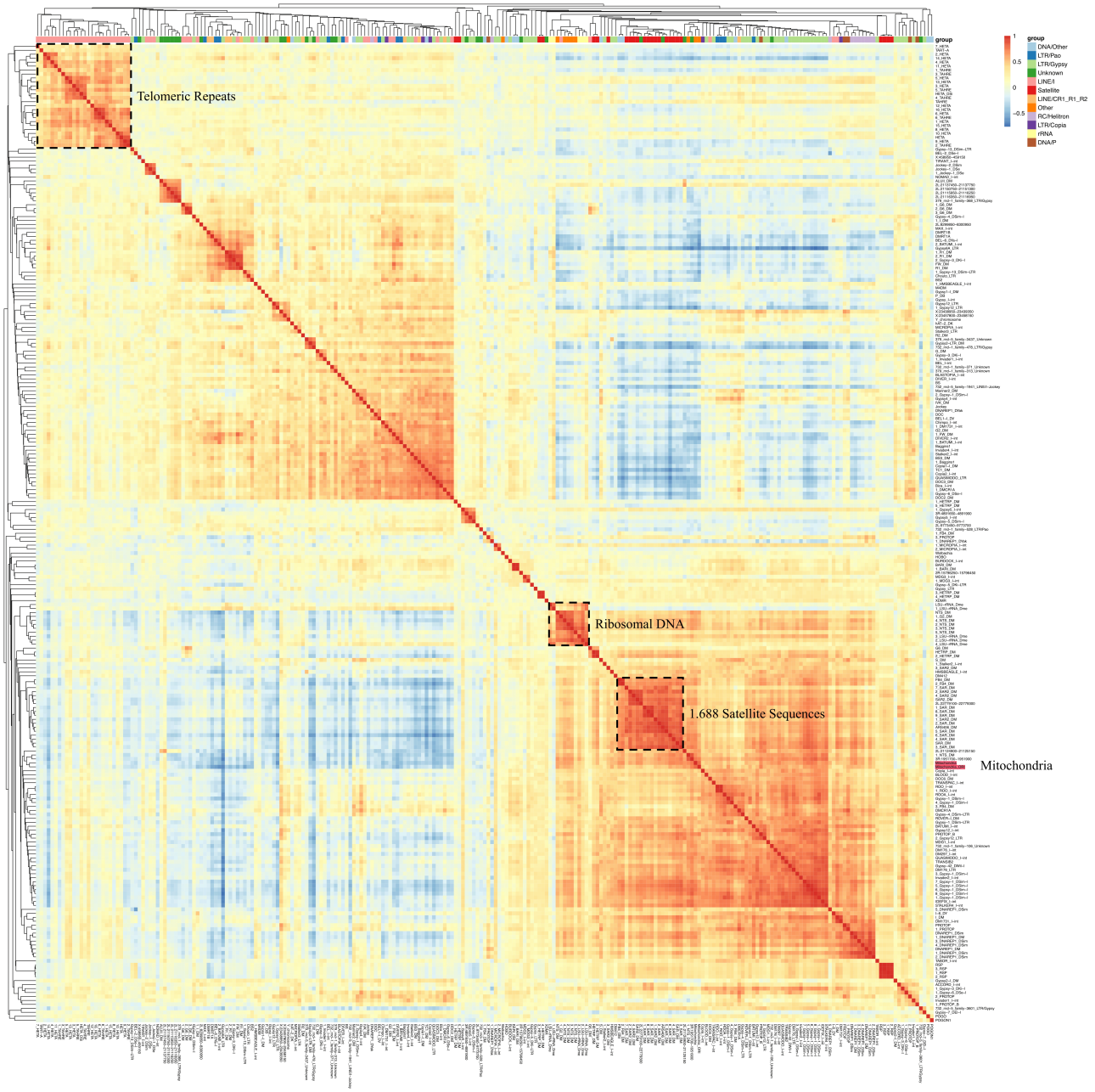

Fig. 01: Heatmap of Spearman correlation matrix of repeat elements copy number vs. repeat element copy number averaged across all DGRP individuals.

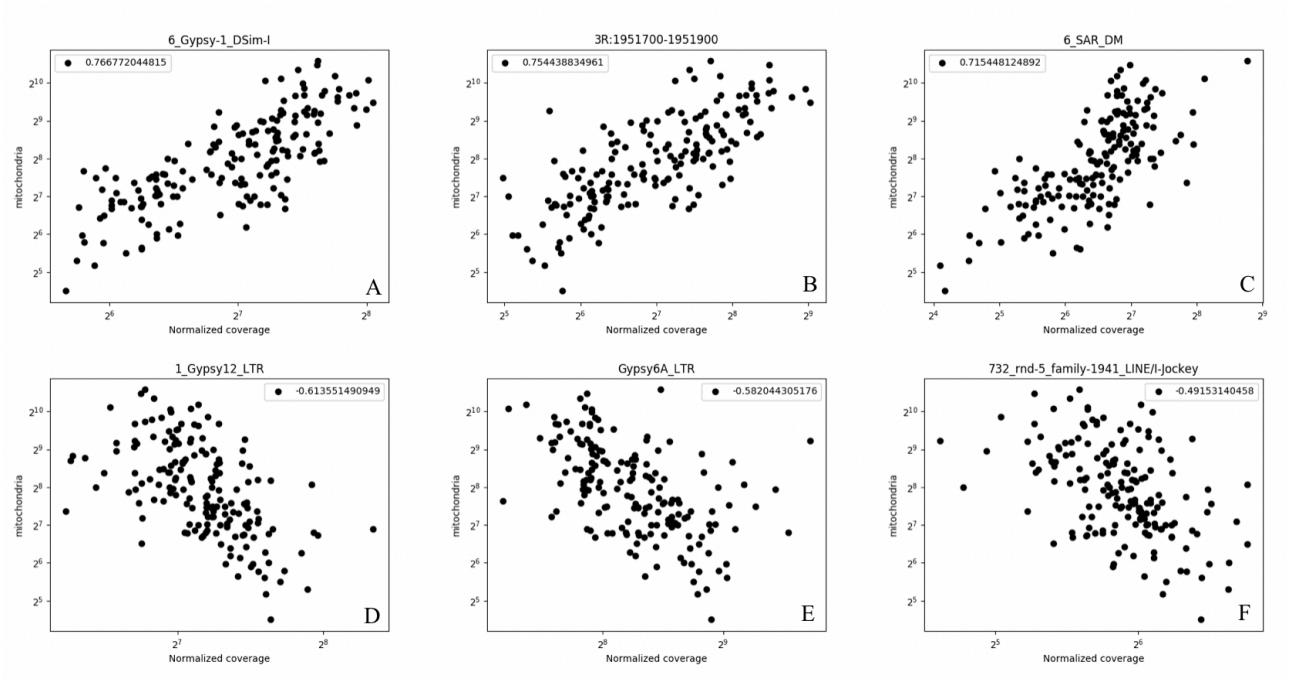

Fig. 02: Coverage of mtDNA with the repeat (A) 6_Gypsy-1_DSim-I, (B) Unidentified repeat at 3R:1951700-1951900, (C) 6_SAR_DM, (D) 1_Gypsy12_LTR, (E) Gypsy6A_LTR, and (F) I-Jockey; with corresponding Spearman correlation coefficient (ρ). These repeats are emphasised to show high positive (A, B, C) or high negative (D, E, F) correlation.

We assumed that there would be high correlation between repeat elements of the same class, and either low or regular correlation of repeat elements across classes. This did not hold true as there seems to be relatively high correlation between Satellite, Helitron and Gypsy repeat elements.

Focusing on mitochondrial repeats, we found that the copy number of a lot of genomic repeats correlated with the copy number of mtDNA; this should not be the case as mitochondria have their own genome as compared to the host. Most of these pairs were highly positively correlated, while others showed more negative correlation *[Table 1]*.

High negative correlation		High positive correlation	
Corr	Name of Repeat Element	Corr	Name of Repeat Element
-0.6135515	1_Gypsy12_LTR	0.7916619	Mitochondria_ORI
-0.5820443	Gypsy6A_LTR	0.766772	6_Gypsy-1_DSim-I
-0.4915314	732_rnd-5_family-1941_LINE/I-Jockey	0.7544388	3R:1951700-1951900
-0.4826265	DIVER2_I-int	0.7532853	5_Gypsy-1_DSim-I
-0.4776319	QUASIMODO_LTR	0.7306136	8_Gypsy-1_DSim-I
-0.4674989	DMRT1A	0.7155331	7_Gypsy-1_DSim-I
-0.4484729	1_BATUMI_I-int	0.7154481	6_SAR_DM
-0.4471379	2_BATUMI_I-int	0.7129666	DM297_I-int
-0.4461737	DOC3_DM	0.7122574	4_SAR_DM
-0.4335121	BS3_DM	0.7048639	3_SAR_DM

Table 1: Spearman correlation of mitochondrial copy number vs. copy number of following repeats across all DGRP strains.

FUTURE DIRECTION

We currently have the genome sequence of RAL-379 which has a high copy number of mitochondrial DNA (mtDNA). We need to sequence RAL-83, which has a low copy number of mtDNA and see what explains the pattern of correlation between genomic DNA and mtDNA.

We are currently creating de novo genome assemblies from the longer reads generated by the Nanopore MinION. This approach will allow us to identify the genomic location and copy number of each repeat element, which we will compare to the copy number inferred by aligning Illumina sequences to the consensus sequence of each repeat. With this study, we hope to learn about the evolutionary dynamics of repeat sequences and the forces that control their copy number in the genome.

METHODS

M.1 MOTIVATION

We needed to identify high repeat regions and identify how exactly they interact with the rest of the genome. The main motivation is to estimate the evolutionary biology of these repeat regions. This can be done by comparing the repetitive sequence content between individuals and seeing which repeats are correlated with the other repeats.

We still don't know how the repetitive portions of the genome vary among individuals as we cannot accurately align the short Illumina reads to the repetitive parts of a genome assembly. This is because (1) each repeat is present at many loci in the genome, and (2) other individuals have additional repeat indels and expansion of tandem repeats.

We hypothesise that TEs and other repeats that use similar transposition/amplification mechanisms will have copy numbers that are correlated among individuals.

M.2 PROTOCOL

In order to compare repeat copy number among individuals, we needed to get the consensus sequence for every possible repetitive element in the Drosophila genome. We started analysis with sequences of known TEs.

We ran *RepeatModeler*, which tries to find novel repeats, on the sequence data of the DGRP strains. We than ran a home-brew pipeline to isolate repeats with high coverage [*A01-A07*]. In this pipeline: we found windows of 50bp that do not have simple repeats (such as poly-A elements, and other tandem repeat elements with low complexity), filtered them and collapsed adjacent windows. We filtered them again based on collapsed window size. After isolating these high coverage windows, we intersected the coverage values of these windows across all individuals.

We then collapsed duplicate sequences using *UCLUST*, collapsed tandem repeats and filtered based on window size. We chose the minimum window size to be 200bp. We then aligned them to Illumina sequences from each of the 173 DGRP strains and counted the coverage (the number of Illumina reads that aligned with the repeats deciphered above).

We calculated the Spearman coefficient of the coverage of each of these repeat elements to each other; after which we normalised these alignments to adjust for sequencing bias and correct for sequencing depth variation among individuals. *deepTools* was used to do this. We then made a heatmap of the data [*Fig 01*].

ACKNOWLEDGEMENTS

LAB MEMBERS

We would also like to specially thank Weihuan Cao for constant input with improving procedures, preparing embryos, extracting nuclei and generally rearing the fly strains that have been used for the study.

AMAREL

The authors acknowledge the Office of Advanced Research Computing (OARC) at Rutgers, The State University of New Jersey for providing access to the Amarel cluster and associated research computing resources that have contributed to the results reported here. URL: http://oarc.rutgers.edu

Bibliography

Biscotti, M. A., E. Olmo and J. S. Heslop-Harrison, 2015 Repetitive DNA in eukaryotic genomes. Chromosome Res 23: 415-420.

Corces, V. G., and P. K. Geyer, 1991 Interactions of retrotransposons with the host genome: the case of the gypsy element of Drosophila. Trends in Genetics 7: 86-90.

de Koning, A. P. J., W. Gu, T. A. Castoe, M. A. Batzer and D. D. Pollock, 2011 Repetitive Elements May Comprise Over Two-Thirds of the Human Genome. PLOS Genetics 7: e1002384.

Feschotte, C., and E. J. Pritham, 2007 DNA Transposons and the Evolution of Eukaryotic Genomes. Annual review of genetics 41: 331-368.

Jain, M., H. E. Olsen, B. Paten and M. Akeson, 2016 The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. Genome biology 17: 239-239.

McClintock, B., 1956 Controlling elements and the gene. Cold Spring Harbor symposia on quantitative biology 21: 197-216.

Muñoz-López, M., and J. L. García-Pérez, 2010 DNA Transposons: Nature and Applications in Genomics. Current Genomics 11: 115-128.

Nakamura, K., T. Oshima, T. Morimoto, S. Ikeda, H. Yoshikawa et al., 2011 Sequence-specific error profile of Illumina sequencers. Nucleic Acids Res 39.

O'Sullivan, R. J., and J. Karlseder, 2010 Telomeres: protecting chromosomes against genome instability. Nat Rev Mol Cell Biol 11: 171-181.

Quail, M. A., M. Smith, P. Coupland, T. D. Otto, S. R. Harris et al., 2012 A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC Genomics 13: 341.

Shapiro, J. A., and R. von Sternberg, 2005 Why repetitive DNA is essential to genome function. Biol Rev Camb Philos Soc 80: 227-250.

Sutherland, G. R., and R. I. Richards, 1995 Simple tandem DNA repeats and human genetic disease. Proceedings of the National Academy of Sciences 92: 3636-3641.

Vieira, M. L. C., 2016 Microsatellite markers: what they mean and why they are so useful. 39: 312-328.

deepTools: Fidel Ramírez, Friederike Dündar, Sarah Diehl, Björn A. Grüning, and Thomas Manke. deepTools: a flexible platform for exploring deep-sequencing data. Nucl. Acids Res. first published online May 5, 2014 doi:10.1093/nar/gku365

DGRP: Mackay, T. F. C., et. al. (2012). "The Drosophila melanogaster Genetic Reference Panel." Nature 482(7384): 173-178.

USEARCH: Edgar,RC (2010) Search and clustering orders of magnitude faster than BLAST, Bioinformatics 26(19), 2460-2461. doi: 10.1093/bioinformatics/btq461

RepeatModeler: http://www.repeatmasker.org/RepeatModeler/

Pheatmap: https://CRAN.R-project.org/package=pheatmap

eXpress: https://pachterlab.github.io/eXpress/overview.html

APPENDIX

A01 01_STRAIN_COV.SH

Run using:

```
sbatch 01_strain_cov.sh
```

```
#!/bin/bash

#SBATCH --partition=genetics_1             # Partition (job queue)
#SBATCH --requeue                        # Return job to the queue if preempted
#SBATCH --job-name=01strain_cov          # Assign an short name to your job
#SBATCH --nodes=1                        # Number of nodes you require
#SBATCH --ntasks=1                       # Total # of tasks across all nodes
#SBATCH --cpus-per-task=1                # Cores per task (>1 if multithread tasks)
#SBATCH --mem=20G                       # Real memory (RAM) required (MB)
#SBATCH --time=00:03:00                  # Total run time limit (HH:MM:SS)
#SBATCH --output=slurm.%j.out     # STDOUT output file
#SBATCH --error=slurm.%j.err      # STDERR output file (optional)
#SBATCH --export=ALL                     # Export you current env to the job env

cd /scratch/cpr74/2018Spring/coverage_per_strain/
new_strains_2018_02_DGRP732_assembly

# Make small windows of 50 bp each.
bedtools makewindows -g DGRP732.final_pilon.genome -w 50 > ./
windows_50bp_732_pilon.genome
sleep 3

# Keep windows that do not overlap the simple repeats.
bedtools intersect -v -a windows_50bp_732_pilon.genome -b
DGRP732.final_pilon.simple_repeats.bed > ./
windows_without_low_complexity_repeats.bed
```

A02 02_STRAIN_COV.SH

Run using:

```
sbatch --array 1-175 02_strain_cov.sh
```

```
#!/bin/bash

#SBATCH --partition=genetics_1             # Partition (job queue)
#SBATCH --requeue                        # Return job to the queue if preempted
#SBATCH --job-name=02strain_cov          # Assign an short name to your job
#SBATCH --nodes=1                        # Number of nodes you require
#SBATCH --ntasks=1                       # Total # of tasks across all nodes
```

```
#SBATCH --cpus-per-task=1            # Cores per task (>1 if multithread tasks)
#SBATCH --mem=30G                   # Real memory (RAM) required (MB)
#SBATCH --time=03:00:00              # Total run time limit (HH:MM:SS)
#SBATCH --output=slurm.%j.out     # STDOUT output file
#SBATCH --error=slurm.%j.err      # STDERR output file (optional)
#SBATCH --export=ALL                 # Export you current env to the job env

cd /scratch/cpr74/2018Spring/coverage_per_strain/
new_strains_2018_02_DGRP732_assembly

# Get coverage for windows and keep windows with coverage above 500. Also keep
file with all coverages.
bedtools coverage -mean -a windows_without_low_complexity_repeats.bed -b
$SLURM_ARRAY_TASK_ID.reduced.bam > ./cov_$SLURM_ARRAY_TASK_ID.txt
sleep 3
cat ./cov_$SLURM_ARRAY_TASK_ID.txt | awk '$4 > 100' | sort -k1,1 -k2,2n > ./
high_cov_$SLURM_ARRAY_TASK_ID.txt
```

A03 03_STRAIN_COV.SH

Run using:

```
sbatch --array 1-175 03_strain_cov.sh
```

```
#!/bin/bash

#SBATCH --partition=genetics_1            # Partition (job queue)
#SBATCH --requeue                    # Return job to the queue if preempted
#SBATCH --job-name=03strain_cov          # Assign an short name to your job
#SBATCH --nodes=1                    # Number of nodes you require
#SBATCH --ntasks=1                   # Total # of tasks across all nodes
#SBATCH --cpus-per-task=1            # Cores per task (>1 if multithread tasks)
#SBATCH --mem=20G                   # Real memory (RAM) required (MB)
#SBATCH --time=03:00:00              # Total run time limit (HH:MM:SS)
#SBATCH --output=slurm.%j.out     # STDOUT output file
#SBATCH --error=slurm.%j.err      # STDERR output file (optional)
#SBATCH --export=ALL                 # Export you current env to the job env

cd /scratch/cpr74/2018Spring/coverage_per_strain/
new_strains_2018_02_DGRP732_assembly

# Sort the windows and use bedtools to merge adjacent windows. Average coverage
values using '-c' and '-o' options.
cat high_cov_$SLURM_ARRAY_TASK_ID.txt | sort -k1,1V -k2,2n | bedtools merge -c 4
-o mean > ./merged_$SLURM_ARRAY_TASK_ID.bed
```

A04 04_STRAIN_COV.SH

Run using:

```
sbatch 04_strain_cov.sh
```

```
#!/bin/bash

#SBATCH --partition=genetics_1             # Partition (job queue)
#SBATCH --requeue                       # Return job to the queue if preempted
#SBATCH --job-name=04strain_cov          # Assign an short name to your job
#SBATCH --nodes=1                       # Number of nodes you require
#SBATCH --ntasks=1                      # Total # of tasks across all nodes
#SBATCH --cpus-per-task=1               # Cores per task (>1 if multithread tasks)
#SBATCH --mem=200G                      # Real memory (RAM) required (MB)
#SBATCH --time=03:00:00                 # Total run time limit (HH:MM:SS)
#SBATCH --output=slurm.%j.out     # STDOUT output file
#SBATCH --error=slurm.%j.err      # STDERR output file (optional)
#SBATCH --export=ALL                    # Export you current env to the job env

cd /scratch/cpr74/2018Spring/coverage_per_strain/
new_strains_2018_02_DGRP732_assembly/

# Combine all of the high-coverage windows across all individuals, sort them by
chromosome and start coordinate, and then merge them to create a file containing
windows with high coverage in at least one individual.
cat merged_*.bed | sort -k1,1V -k2,2n | bedtools merge > ./all_merged.bed
sleep 3

# Filter this file by only keeping windows that are 200 bp or larger.
awk '$3-$2 > 199' all_merged.bed > ./large_windows.bed
sleep 3

# For each window, intersect it with the coverage windows from all individuals.
bedtools intersect -wo -filenames -a large_windows.bed -b cov_*.txt > ./
temp1.bed
sleep 3

# For each window, intersect it with the coverage windows from all individuals.
bedtools groupby -i temp1.bed -g 4 -c 1,2,3,8 -o distinct,distinct,distinct,mean
> ./temp2.bed
sleep 3
bedtools groupby -i temp2.bed -g 2,3,4 -c 1,5 -o collapse,collapse > ./
cov_per_strain.bed
sleep 3

# Convert from .bed to .table for pheatmaps
head -1 cov_per_strain.bed | awk '{print "STRAINS\t"$4}' | sed -r 's/\.txt//g' |
sed -r 's/cov_//g' | tr "," "\t" > cov_per_strain.table
sleep 1.5
awk '{print $1"_"$2"_"$3"\t"$5}' cov_per_strain.bed | tr "," "\t" | cut -f 1-174
>> cov_per_strain.table
```

```
sleep 1.5

# Convert header in .table to RAL_IDS via python.
python bam_to_ral.py

# Running R script with the commands in commands.R.
R --no-save < commands.R >& output.log
```

A05 05_STRAIN_COV_CLEANUP.SH

Run using:

```
sbatch 04_strain_cov_cleanup.sh
```

```
#!/bin/bash

#SBATCH --partition=genetics_1             # Partition (job queue)
#SBATCH --requeue                        # Return job to the queue if preempted
#SBATCH --job-name=05cleanup              # Assign an short name to your job
#SBATCH --nodes=1                        # Number of nodes you require
#SBATCH --ntasks=1                       # Total # of tasks across all nodes
#SBATCH --cpus-per-task=1                # Cores per task (>1 if multithread tasks)
#SBATCH --mem=20G                       # Real memory (RAM) required (MB)
#SBATCH --time=03:00:00                  # Total run time limit (HH:MM:SS)
#SBATCH --output=slurm.%j.out     # STDOUT output file
#SBATCH --error=slurm.%j.err      # STDERR output file (optional)
#SBATCH --export=ALL                     # Export you current env to the job env

cd /scratch/cpr74/2018Spring/coverage_per_strain/
new_strains_2018_02_DGRP732_assembly

rm ./windows_50bp_iso1.genome
rm ./windows_without_low_complexity_repeats.bed
rm ./high_cov_*.txt
rm ./cov_*.txt
rm ./merged_*.bed
rm ./all_merged.bed
rm ./large_windows.bed
rm ./temp1.bed
rm ./temp2.bed
rm ./cov_per_strain.bed
rm ./cov_per_strain.table
rm ./cov_per_RALstrain.table
rm ./output.log
```

A06 BAM_TO_RAL.PY

```
# bam_number_to_RAL.lookup
data = open('./bam_number_to_RAL.lookup', 'r')
lendata = len(open('bam_number_to_RAL.lookup', 'r').readlines())

lookup = []
for i in range(lendata):
    a = data.readline()
    lookup.append(a[:-1].split(' '))

# cov_per_strain_nosimplereps.table
data = open('./cov_per_strain.table', 'r')
lendata = len(open('./cov_per_strain.table', 'r').readlines())

to_change = data.readline()[:-1].split('\t')

changed = to_change

for i in range(len(changed)):
    for item in lookup:
        if changed[i] == item[1]:
            changed[i] = item[0]

with open('./cov_per_RALstrain.table', 'w') as f:
    print('\t'.join(changed), file=f)
    for i in range(lendata-1):
        print(data.readline()[:-1], file=f)
```

A07 COMMANDS.R

```
library(pheatmap)
read.table("cov_per_RALstrain.table",row.names=1,header=T)->D
pdf("DGRP732_assembly_100+cov_correlation.pdf",width=20,height=200)
pheatmap(log10(as.matrix(D)+1),clustering_distance_rows    =
"correlation",show_rownames=T,fontsize_row=6,clustering_distance_cols    =
"correlation",fontsize_col=6)
dev.off()
```