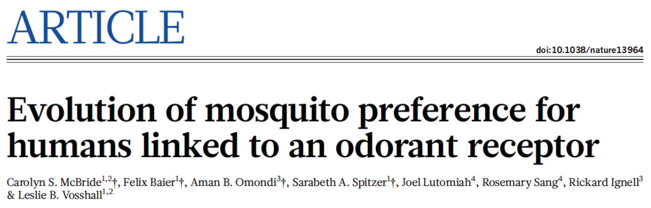
First we read a paper that we thought was interesting...



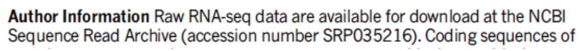
ARTICLE

doi:10.1038/nature13964

**Evolution of mosquito preference for humans linked to an odorant receptor**

Carolyn S. McBride[1,2]†, Felix Baier[1]†, Aman B. Omondi[3]†, Sarabeth A. Spitzer[1]†, Joel Lutomiah[4], Rosemary Sang[4], Rickard Ignell[3] & Leslie B. Vosshall[1,2]

On the last page of the paper you can find where they archived their data...usuall the SRA at NCBI:



**Author Information** Raw RNA-seq data are available for download at the NCBI Sequence Read Archive (accession number SRP035216). Coding sequences of

We hit the link in the paper for SRP035216 (or go to NCBI SRA and search for this project. This lands us on a page with lots of individual libraries. Near the top right we see a "send to" option/link. Hit that and then select "Run Selector" -> Go. This takes us to a screen where we can bulk download run information, including the names of the runs (SRR's). This is what we will use next to bulk download the raw sequences.

We select our desired runs, download the Accession List.

We start making commands to handle each library in our workflow. We can write simple scripts to generate the code by looping through the lines of our Accession List. The command fastq-dump is from a toolset published by NCBI for accessing their SRA data, called SRA Tools. The --split-3 option allows you to handle both paired end and one way sequence data (i.e., it is ok if there are either R1/R2 or only a single, fastq files). The -X 1000 says you don't want the whole library, just the first 1000 spots from the lane. We are using this option because it is faster and we want to make sure stuff works first.
The following code will write download instructions for each line (library name) in SRR_Acc_List.txt and then #append them to a file called rnaseqcommands.txt. NOTE: you can reuse this same loop structure to make all #of your commands specific to each library you want to map and analyze.
***module load sratools (check name/version using module avail)

```
while read i; do
        echo fastq-dump --split-3  -X 1000 $i;
done <SRR_Acc_List.txt >> rnaseqcommands.txt
```

The next part is to map your reads, but first you need to download, unzip and index your genome and gff (the latter doesn't need to be indexed). You can use curl and gunzip to do the first parts. The following is an example for using the bowtie2 algorithm to index your genome. "bowtie2-build" is the command to build the index. The -f is just specifying your FASTA file to index, in this case the long name beginning with "GCF". The last part, Aegypti_genome is just what you want the resulting index files to be called.
***module load bowtie2 (check name/version using module avail)

```
bowtie2-build -f GCF_000004015.4_AaegL3_genomic.fna Aaegypti_genome
```

The last part is to map the reads, organize them, and then count them. This can be done with the following command sets.

This is the mapping command. You specify the GFF file to help speed things up, and the -p is for the number of threads and the -o specifies an output directory (which is being newly made. The last two things specified are the indexed genome to map to and the fastq reads to map. (guess what, module load tophat first)

```
tophat2 -G GCF_000004015.4_AaegL3_genomic.gff -p 10 -o SRR1103959 Aaegypti_genome SRR1103959.fastq
```

The samtools commands will make and organize files that will be useful for checking alignments manually and for use by other algorithms. (oh yes, don't forget, you need to module load samtools too!!!)

```
samtools sort -n SRR1103959/accepted_hits.bam SRR1103959_sn
samtools view -o SRR1103959_sn.sam SRR1103959_sn.bam
samtools sort SRR1103959/accepted_hits.bam SRR1103959_s
samtools index SRR1103959_s.bam
```

Due to incompatibility issues, the first part loads an older version of python (by unloading the current version). Then the command htseq-count goes through the bam files and counts the number of reads mapped to each feature in the GFF file. The -s allows you to specify the strand (which we don't want), the -a is for alignment quality. The --idattr allows you to select which name you use to call your feature. Here it is set to "Dbxref" (database cross reference) which gives multiple gene names. You could also set this to "ID" and you will get the gene ID. See the last column in the GFF file for the options. Lastly you give the mapping SAM file and the gff file, redirecting the output to a file with the extension ".count" so you know what it is.

```
module unload python/2.7 and module load htseq-count
htseq-count -s no -a 4 --idattr=Dbxref SRR1103959_sn.sam GCF_000004015.4_AaegL3_genomic.gff > SRR1103960.count
```