SexFindR

Release 2021

Phil Grayson

Feb 25, 2022

CONTENTS

These pages contain the documentation necessary to run SexFindR on your species of interest. The pages are meant to used in conjunction with the folders on Github (https://github.com/phil-grayson/SexFindR), which contain the same structure. Commands for the analysis of tiger pufferfish (*Takifugu rubripes*) are present throughout to reproduce these analyses as a test case. When relevant, software versions and links to the source code are presented when the software is mentioned the first time.

Commands were run using a combination of SLURM systems and standard Linux servers. When possible, configurations are provided for the SLURM systems in `.sh` scripts (these are submitted with `sbatch` commands). Commands that finish quickly, or commands that could only be run locally, are simply provided as command line input. `R` code is also provided for relevant steps. File paths, module loads, SLURM accounts, etc. may need to be modified for your specific use case.

Please reach out to Dr. Phil Grayson (`phil.d.grayson <at> gmail.com`) or open a GitHub issue if you experience difficulty running any part of the `SexFindR` workflow.

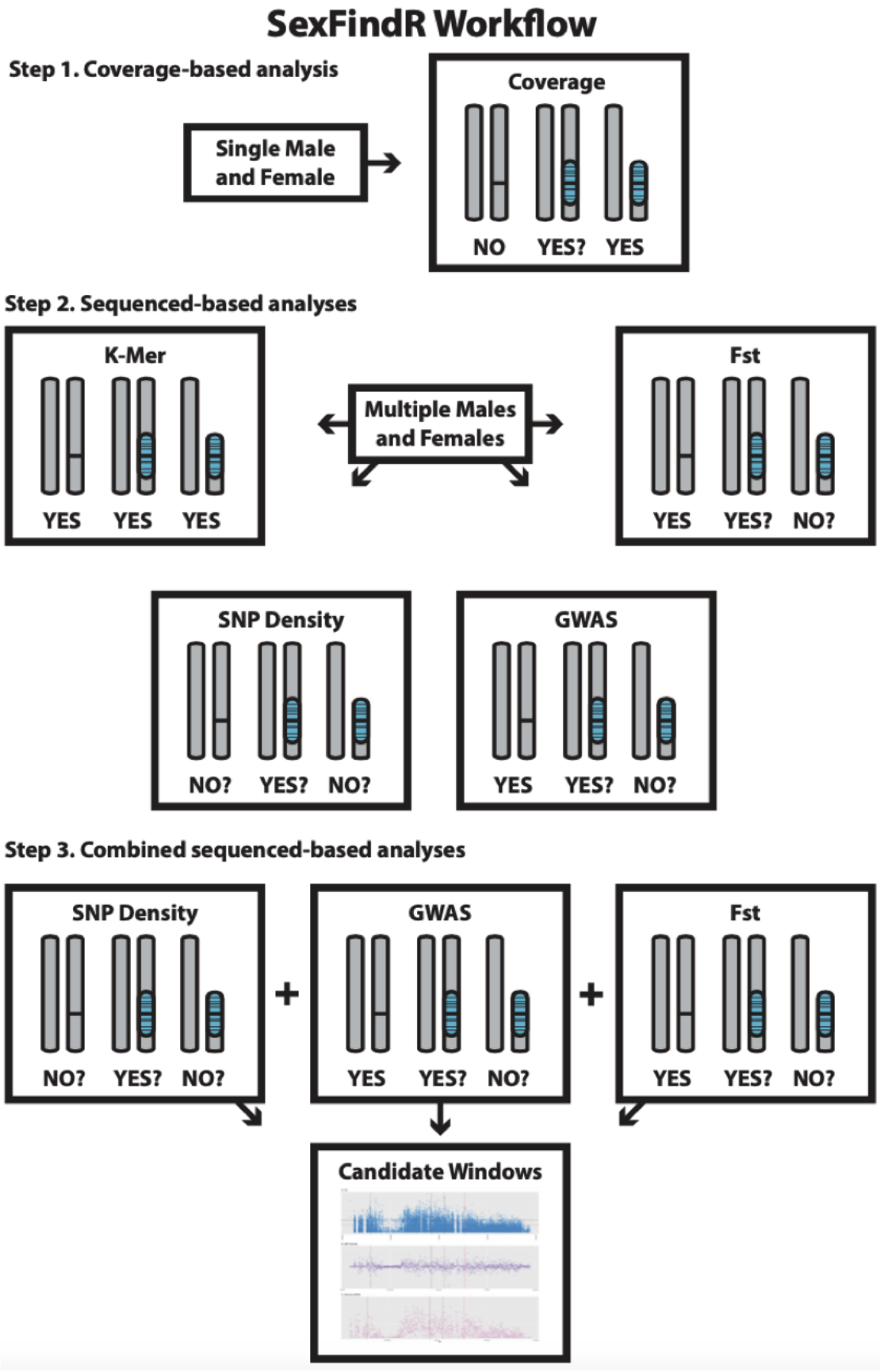

SexFindR Workflow
Step 1. Coverage-based analysis
Single Male and Female
Coverage
NO
YES?
YES
Step 2. Sequenced-based analyses
K-Mer
YES
YES
YES
Multiple Males and Females
Fst
YES
YES?
NO?
SNP Density
NO?
YES?
NO?
GWAS
YES
YES?
NO?
Step 3. Combined sequenced-based analyses
SNP Density
NO?
YES?
NO?
+
GWAS
YES
YES?
NO?
+
Fst
YES
YES?
NO?
Candidate Windows

Figure 1. SexFindR workflow in three steps

CHAPTER

ONE

CONTENTS

1.1 Step 0. Mapping and variant calling

Prior to running Step 1 (Coverage-based analysis) or any of the analyses from Step 2 (Sequence-based analyses) except the reference-free k-mer analysis, you will need to map your reads to a reference genome.

In the SexFindR paper, we used `Bowtie2` (v 2.3.4.3; http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) for read mapping and we provide some scripts and configuration for this within the GitHub repo and below. If you already have mapped reads from `bwa-mem` or another widely-used algorithm, please feel free to use those.

For all the sequence-based analyses mapped to a reference genome, SNP calling is also required. In the SexFindR paper, we used `Platypus` (commit 3e72641; https://github.com/andyrimmer/Platypus) to jointly call SNPs across all samples, and we provide some scripts and configurations for this within the GitHub and below. Again, if you have already called SNPs for your samples using `GATK` or another widely-used algorithm, please feel free to use those.

`Platypus` requires that the genome file be indexed with `samtools` (v1.10; https://github.com/samtools/samtools). We also filtered the raw `vcf` to only include sites that `PASS` the quality filters. This requires `bcftools` (v1.9; https://github.com/samtools/bcftools). `vcftools` is also used to filter for biallelic sites (v0.1.14; https://github.com/vcftools/vcftools).

1.1.1 Download raw reads from NCBI

Requires `fasterq-dump` from the `SRA-tools` (https://github.com/ncbi/sra-tools). In my experience, `fasterq-dump` can be fairly flakey depending on your system and connection, so this download might need to be repeated multiple times to get all the samples. You might want to separate out the `acc_list.txt` files or add something like `if [ ! -f ${file}_1.fastq ]` to the code below to avoid downloading the same file repeatedly.

```
for file in $(cat acc_list.txt); do echo $file; date; fasterq-dump $file -e 36; done
```

1.1.2 Download the genome

```
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/901/000/725/GCF_901000725.2_fTakRub1.2/
↪GCF_901000725.2_fTakRub1.2_genomic.fna.gz
gunzip GCF_901000725.2_fTakRub1.2_genomic.fna.gz
```

1.1.3 Create a bowtie2 index

```
bash bowtie2_makeindex_linux.sh GCF_901000725.2_fTakRub1.2_genomic.fna fugu &> index_
↪outerr.txt &
```

1.1.4 Map reads to the genome

For each sample, the reads must be mapped using a command like:

```
bash bowtie2_16_linux.sh SRR8585991_* fugu &> bt2_SRR8585991_outerr.txt &
```

`bowtie2_16_long.sh` is also included as a reference on SLURM systems.

1.1.5 Call variants using Platypus

Variants are jointly called (all at once) through the use of a bam list (e.g., 15M_14F_bams.txt). The following was run on a SLURM system:

```
samtools faidx GCF_901000725.2_fTakRub1.2_genomic.fna
sbatch platypus_all_region_1day.sh 15M_14F_bams.txt GCF_901000725.2_fTakRub1.2_genomic.
↪fna
```

1.1.6 Filter for variant calls that PASS quality filters

This script will keep only those variants that have `PASS` in the `FILTER` field, removing low quality calls.

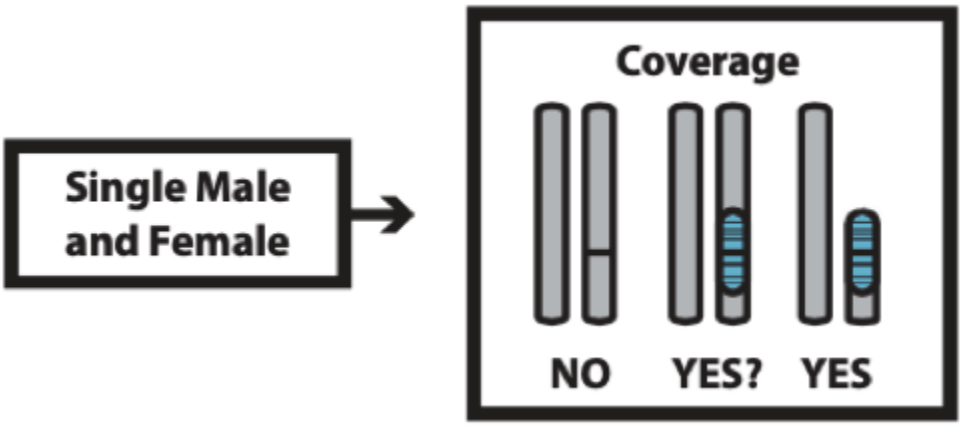
```
sbatch bcf_filter.sh fugu_14M_13F.vcf
```

1.1.7 Filter for biallelic sites

Some downstream analyses (e.g., `Fst`) require only biallelic sites to work properly (e.g., 0/0, 0/1, 1/1). Execute the following to filter for only biallelic sites:

```
vcftools --vcf filtered_PASS_fugu_14M_13F.vcf --max-alleles 2 --stdout --recode --recode-
↪INFO-all | gzip -c > biallelic_filtered_PASS_fugu_14M_13F.vcf.gz
```

1.2 Step 1. Coverage-based analysis



To determine if there are large sex-specific regions present in your species of interest, a coverage-based analysis is first carried out.

1.2.1 Running DifCover

`DifCover` (https://github.com/timnat/DifCover) requires only a single male and single female `bam` file aligned to a reference genome.

Example `DifCover` command with 1 male (`SRR8585998_1.fastq.bam`) versus 1 female (`SRR8585999_1.fastq.bam`).

```
bash run_difcover.sh SRR8585998_1.fastq.bam SRR8585999_1.fastq.bam 1 >> difCov_Male98_
↪over_Female99_outerr.txt 2>&1
```

Note that you will need to modify `run_difcover.sh` to have a proper path for `FOLDER_PATH` on your system, and the `1` is the library-specific Adjustment Coefficient (`AC`) to account for differences in sequencing depth between samples. The correct `AC` value can be determined by taking the ratio of modal depths for the samples of interest through `samtools stats` as follows:

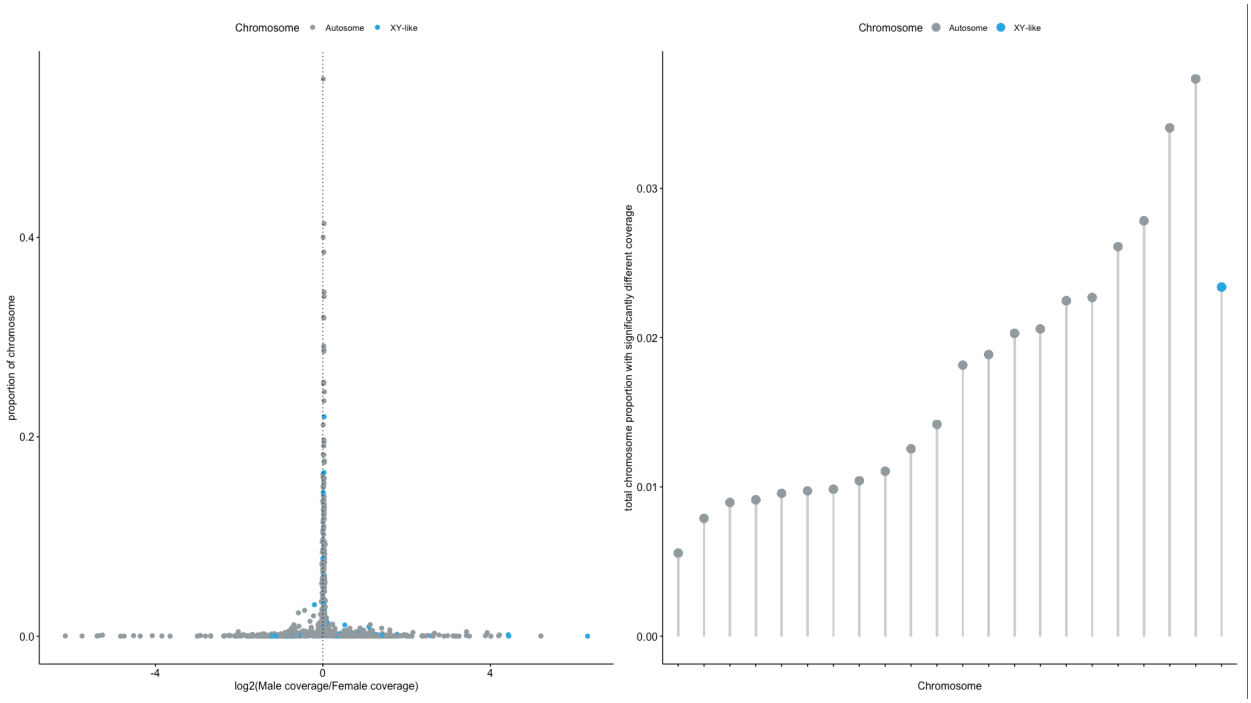
```
chmod u+rwx samtools_Modaldepth.sh
for file in $(ls *q.bam); do ./samtools_Modaldepth.sh $file; done
```

Once you have a modal depth for each sample you are interested in, `AC` is (modal coverage of sample 2 / modal coverage of sample 1). Of note, I have found that although this works well for some samples, others report a modal depth of 1, which is not helpful for determining `AC`. In these cases, I have have mixed success simply using the ratio of the `bam` file sizes for `AC`. This can be dialed in after the analysis as well by plotting and examining the center of the distribution (which should lie at 0 for autosomes if `AC` is properly set).

1.2.2 Analyzing DifCover Results

An example output file `sample1_sample2.ratio_per_w_CC0_a10_A219_b10_B240_v1000_l500.log2adj_1.DNAcopyout` is included in the GitHub repository alongside the R script `Fugu_M98_F99_DifCover.R` used to generate the figures in the main body of the paper. Given that fugu has only a single SNP that is fixed between males and females, coverage does not produce any outlier windows for this species.

As an example of a species that does exhibit significant differences in depth, the `DifCover` plot for chicken is also provided below. The code for chicken can be found in `Supplemental Code` on GitHub.



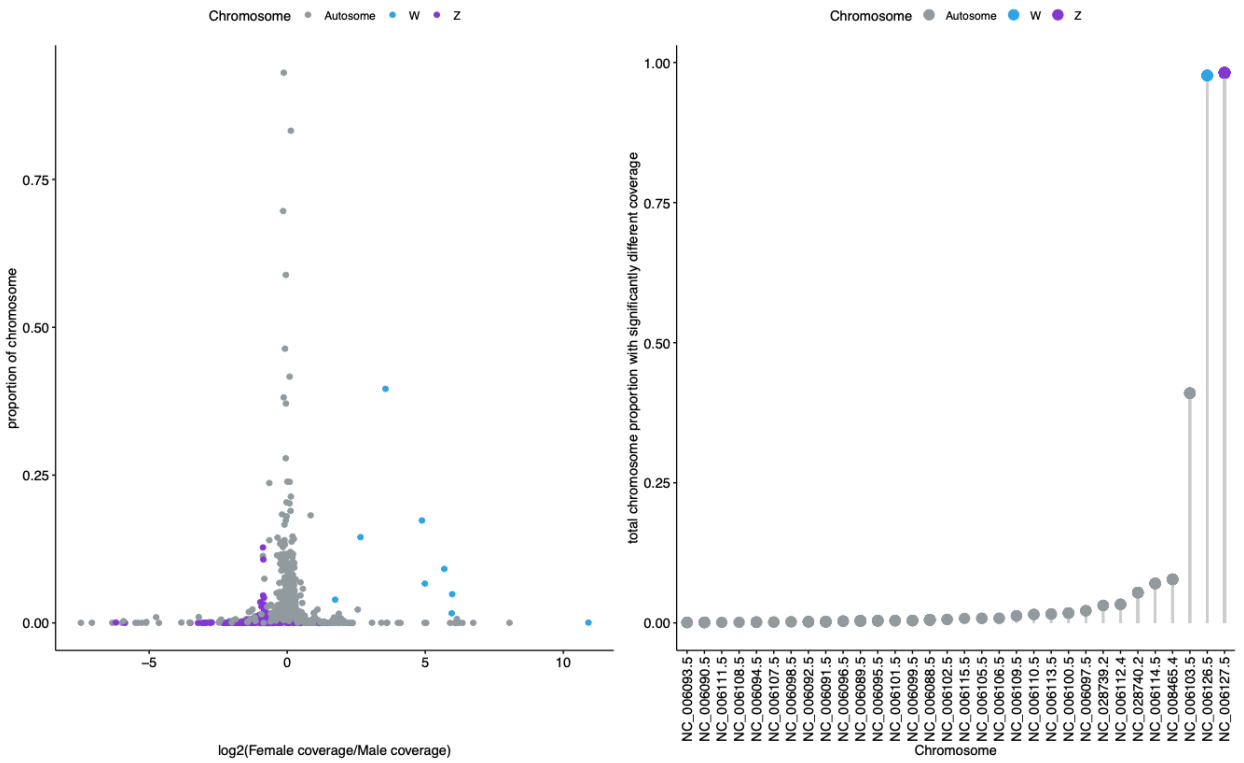
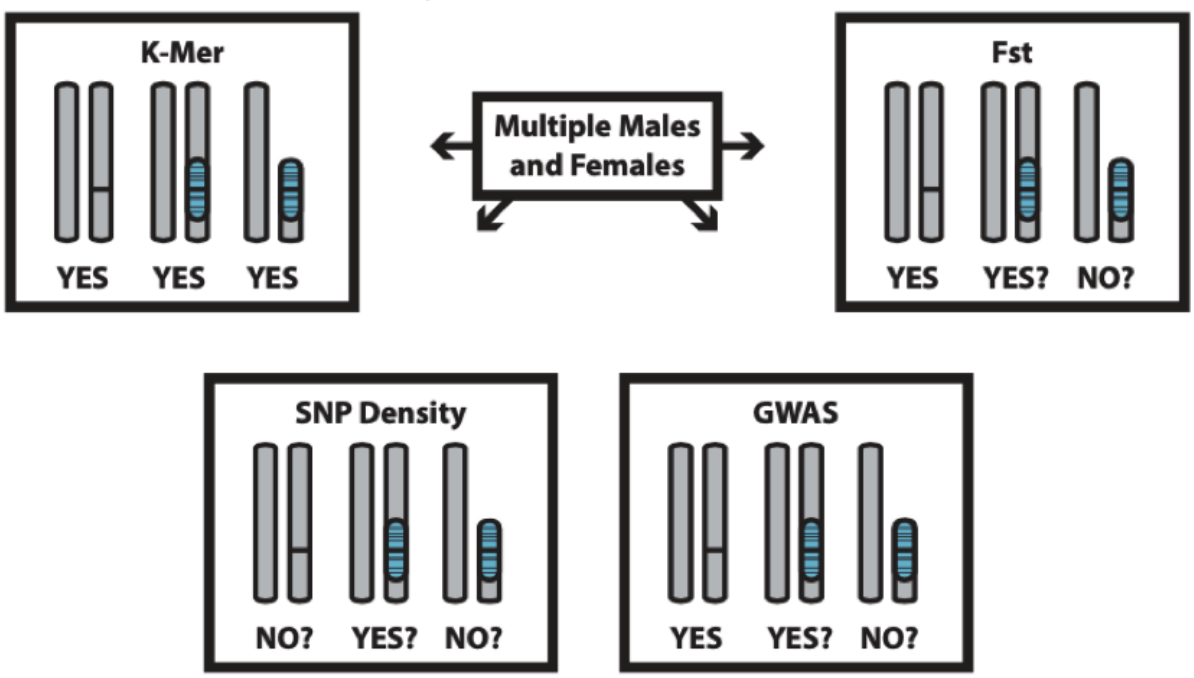


Figure 2. DifCover fails (top) to identify coverage differences between a single male and single female in the fugu, indicating that there are no large regions of recombination suppression in the fugu genome (and/or suggesting that the fugu sex determining region is very young in evolutionary time). DifCover succeeds (bottom) at identify the Z and W chromosomes in chicken using only information on depth of coverage between a single male and single female

1.3 Step 2. Sequence-based analyses



1.3.1 SNP Density

In nascent sex-linked sequences, we would expect to see differences in overall SNP density between males and females due to Y reads still mapping to the X (or W to the Z). This is due to the divergent evolutionary trajectories of the X and Y following recombination suppression on the Y. SNP density can identify regions where male or female-specific SNPs are still segregating within their respective populations.

This analysis requires `vcftools` and `R`.

Running SNP Density

We use `vcftools SNPDensity` to calculate SNP Density for each sample in 10kb windows across the genome. This requires individual `vcf` files for each sample. The individual files are created and the `SNPDensity` calculations carried out using the commands below.

```
mkdir individual_SNP_density

for file in $(cat snpDen_females.txt); do sbatch SNPdensity.sh $file individual_SNP_
↪density/Female_${file%%.*}.vcf biallelic_filtered_PASS_fugu_14M_13F.vcf.gz; sleep 0.1;␣
↪done

for file in $(cat snpDen_males.txt); do sbatch SNPdensity.sh $file individual_SNP_
↪density/Male_${file%%.*}.vcf biallelic_filtered_PASS_fugu_14M_13F.vcf.gz; sleep 0.1;␣
↪done
```

Here, `snpDen_males.txt` and `snpDen_females.txt` reference files are simply lists of the sample IDs (as found in the `vcf` header) that contain the sample from those sexes. The submission script above appends `Male` or `Female` to

the SNPDensity output file, which we then use to parse these output files in R.

Analyzing SNP Density

SNP density can be analyzed from the R script SNPdensity_permutations_fugu.R. This file runs the user through reading in the individual SNP densities calculated above, then calculates the true male and female means (and differences) for each 10 kb window, before carrying out a small permutation test. This permutation test generates the p-values that allows us to determine the correct order for SNP Density windows in Step 3. A larger permutation test was carried out for both the lamprey and fugu in the main manuscript to further analyze the SNP Density data. Code and documentation for this test is included in the Supplemental Code folder on the GitHub repo.

1.3.2 Fst

Fst is an index of allelic fixation in populations, and so, if there are high levels of Fst within discrete genomic regions when comparing males to females, this would suggest that those regions differ between males and females and, therefore, are not recombining. For the SexFindR workflow, Fst is calculated from a biallelic vcf file using vcftools.

Example command

```
vcftools --gzvcf biallelic_filtered_PASS_fugu_14M_13F.vcf.gz --weir-fst-pop snpDen_males.
↪txt --weir-fst-pop snpDen_females.txt --out biallelic_fst
```

Here, we make use of the same snpDen_males.txt and snpDen_females.txt reference files from the SNP Density step to assign individuals to their correct "population".

1.3.3 Genome Wide Association Study (GWAS)

GWAS identifies associations between a phenotype and a genotype (Klein et al. 2005). Similar to Fst, by carrying out a GWAS with the male and female populations as two different phenotypes, it is possible to identify SNPs that are strongly or weakly associated with sex. These SNPs can be fixed, or nearly fixed, in either males or females.

This analysis requires vcftools and a combination of plink (v 1.07-x86_64; https://www.cog-genomics.org/plink/1.9/) and GEMMA (v0.98.1; https://github.com/genetics-statistics/GEMMA). We tested numerous GWAS algorithms and this one performed best across the sex chromosome divergence continuum.

Filtering the vcf

We use vcftools to convert the vcf to plink format, remove indels, remove sites with >50% missing data, and sites where the minor allele frequency is less than 5% or more than 95%.

```
vcftools --vcf filtered_PASS_fugu_14M_13F.vcf --plink --remove-indels --max-missing 0.5 -
↪-max-maf 0.95 --maf 0.05 --out gwas_fugu
```

Running the GWAS

As outlined in the GEMMA manual (https://github.com/genetics-statistics/GEMMA/blob/master/doc/manual.pdf), the program requires input files in the plink binary format. We first run plink (as recommended) to generate these files and then supply these files to GEMMA alongside the option -lm 2 to specify a likelihood ratio test.

```
/home/pgrayson/programs/plink-1.07-x86_64/plink --file gwas_fugu --pheno sex_fugu_meta.
↪txt --make-bed --out gwas_fugu_plink --noweb --allow-no-sex
~/programs/gemma-0.98.1-linux-static -bfile gwas_fugu_plink -lm 2 -o fugu_gemma_out
```

sex_fugu_meta.txt is included in the GitHub repository as an example file. It contains 3 tab-delimited columns that contain the sample name (repeated in column 1 and column 2) and the phenotype (1 for male, 2 for female - or vice versa). An example is below:

```
SRR8585991_1.fastq      SRR8585991_1.fastq      1
SRR8585992_1.fastq      SRR8585992_1.fastq      2
SRR8585993_1.fastq      SRR8585993_1.fastq      2
```

Analysing the output

The commands above will generate fugu_gemma_out.assoc and fugu_gemma_out.log.txt. The .assoc file contains the GWAS results, including the genome position (rs) the number of individuals that had a call (n_obs) or missing data (n_mis) at that site, the allele frequency (af), and the p-value for the likelihood ratio test (p_lrt). This file can be searched and parsed to identify the top candidates from this analysis.

1.3.4 kmersGWAS

The kmersGWAS approach described below is possibly the most powerful single analysis within the SexFindR protocol. Apart from not requiring a reference genome, this single method has proved capable of detecting sex-linked sequences across the entire sex chromosome evolutionary continuum, from highly degenerate mammalian systems to takifugu, which has a single base that is heterozygous is males and homozygous in females.

After testing a few other k-mer association algorithms as well as a number of specific options within this algorithm, I settled on the following workflow given its consistent results across the sex chromosome divergence continuum. This protocol requires kmersGWAS (v0.2 beta; https://github.com/voichek/kmersGWAS), ABYSS (v2.2.5; https://github.com/bcgsc/abyss), plink (v 1.07-x86_64; https://www.cog-genomics.org/plink/1.9/), R and blast+ (v2.10.0; https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.10.0/). I have presented the necessary steps and configurations for the fugu analysis below, but it should be noted that run times and memory usage vary greatly with genome size, coverage, and sample number. For fewer than 30 samples of fugu, the config provided in the SLURM scripts should be sufficient, but some of the kmersGWAS steps required days to run on large machines (200 GB+ memory) for lamprey, and other species.

Directory structure

kmersGWAS requires a specific directory structure and has many separate steps. These are presented below, but more information can be found in the kmersGWAS manual here (https://github.com/voichek/kmersGWAS/blob/master/manual.pdf).

Generally, you want the following:

1. Files present in a top level directory

2. Individual directories created within the top level directory corresponding to the base name of each sample (e.g., top level directory contains `SRR8585991_1.fastq.gz`, `SRR8585991_2.fastq.gz`, and the `SRR8585991` directory)
3. Within the base name subdirectories (e.g., `SRR8585991`), there are execution scripts and an `input_files.txt` file that contains the paths to the `fastq` files (e.g., `../SRR8585991_1.fastq.gz` and `../SRR8585991_2.fastq.gz` each on their own line).

Running kmersGWAS

The following is a set of example commands to set up the directory structure for the fugu samples, but these will need to be made specific to your file names, path, etc. Many of these steps were executed on a SLURM system and the execution scripts will need to be modified to match your system as well. This process begins with the `fastq` files and one copy of each of the execution scripts in the top level (`fugu_kmerGWAS`) directory. All commands are executed from that top level directory. I have included an example of all necessary text files in the GitHub repository to check against as well.

```
cd /home/pgrayson/scratch/fugu_kmerGWAS
for file in $(ls --color=none *_1.*gz); do baseName=$(echo $file | awk -F'[_]' '{print
↪$1}'); mkdir $baseName; done
ls --color=none -d */ > dirlist.txt
sed 's/\///g' dirlist.txt > clean_dirlist.txt
for dir in $(cat clean_dirlist.txt); do cd $dir; echo $dir; ls --color=none ../*${dir}
↪*gz > input_files.txt; cd ..; done
for dir in $(cat clean_dirlist.txt); do cp step1_kmerGWAS.sh $dir; done
for dir in $(cat clean_dirlist.txt); do cd $dir; sbatch step1_kmerGWAS.sh; cd ..; done
```

Once those first jobs complete, you can run the following (again with modification to the execution script):

```
for dir in $(cat clean_dirlist.txt); do cp strand_kmerGWAS.sh $dir; done
for dir in $(cat clean_dirlist.txt); do cd $dir; sbatch strand_kmerGWAS.sh; cd ..; done
```

We next need the individuals k-mers list file. As above, these should be generated following the `kmersGWAS` manual. An example set of commands used to generate this file for the fugu samples is provided below:

```
cd /home/pgrayson/
ll scratch/fugu_kmerGWAS/ | tail -n +2 | awk '{printf "/scratch/fugu_kmerGWAS/%s/kmers_
↪with_strand\t%s\n", $NF,$NF}' > kmers_list_paths.txt
grep -v .gz kmers_list_paths.txt | grep SRR > kmers_clean_list_path.txt
sed 's/\/scratch/\/home\/pgrayson\/scratch/g' kmers_clean_list_path.txt > final_kmers_
↪clean_list_path.txt
mv final_kmers_clean_list_path.txt ~/scratch/fugu_kmerGWAS
```

Example execution scripts are provided to combine the k-mers and create the k-mers table.

```
sbatch combine_kmersGWAS.sh
sbatch create_table_kmersGWAS.sh
```

Once these steps are completed, we generate `plink` binary files using the `kmers_table_to_bed` function. The `phenotype.pheno` file here contains two tab-delimited columns, `accession_id` and `phenotype_value` with the sample ID in `accession_id` and 1 or 2 in the `phenotype value` column for the male or female phenotype. `-b` here is batch size. Depending on the power of your machine and the number of samples included in your analysis, you might have to produce smaller batches (I ran on a Linux machine with 256 Gb of memory and had to batch other species with more samples). Batches here are the number of k-mers within the analysis for `plink`.

```
~/programs/kmerGWAS/bin/kmers_table_to_bed -t kmers_table -k 31 -p phenotype.pheno --maf␣
↪0.05 --mac 5 -b 1000000000 -o fugu_kmerGWAS_plink
```

Running plink

Next, we run these files through `plink` to obtain `p-values` for each k-mer, representing the association between k-mer presence and phenotypic sex. If you had to batch your k-mers (above), you will also need to execute this command for batch the other batches (e.g., fugu_kmerGWAS_plink.1, fugu_kmerGWAS_plink.2, etc.). Make sure to also append something new to the `--out` name (e.g., `fugu_kmers1`, `fugu_kmers2`), so that previous results are not overwritten.

```
~/programs/plink-1.07-x86_64/plink --noweb --bfile fugu_kmerGWAS_plink.0 --allow-no-sex -
↪-assoc --out fugu_kmers
```

The resulting outfile (e.g., `fugu_kmers.assoc`) can explored and parsed based on the `P` column to identify and pull out the k-mers that have the highest association with sex. An example for fugu to obtain the k-mers with the most significant `p-values` is:

```
awk '$9 < 0.000000000001' fugu_kmers.assoc > most_significant_fugu_assoc.txt
```

Results from different batches can be concatenated with `cat fugu_kmers*.assoc` or something similar following analysis in `plink` (although a simple `cat` will also repeat the header line from each batch, so be aware that you will need to take this into account using `tail` or parsing the output file from `cat` to remove the extra header lines).

Running ABYSS

Once you are happy with the filtered k-mer set, you can use `ABYSS` to assemble those k-mers into small contigs for `blastn` analysis (if a reference genome exists). A basic `python` script (`plink_to_abyss_kmers.py`) is included to parse the filtered `plink` output into `ABYSS`-ready input. The `ABYSS` input should be in fasta format, with the `p-value` as the sample ID and the k-mer as the sequence. e.g.,

```
>2.005e-13
AAAAAAAAAAAATCATTTCCCACCTCATCAA
>2.005e-13
AAAAAAAAAAATCATTTCCCACCTCATCAAT
>2.005e-13
AAAAAAAAAATCATTTCCCACCTCATCAATC
...
```

To generate this file, the following should work:

```
python plink_to_abyss_kmers.py most_significant_fugu_assoc.txt fugu_plink_abyss_input.txt
```

If your output does not match the example above, you might need to change the index positions in the python script to correctly grab the k-mer and the p-value columns (given different `plink` versions). This input file is then run through ABYSS to assemble small contigs.

```
ABYSS -k25 -c0 -e0 fugu_plink_abyss_input.txt -o fugu_plink_abyss_output.txt
```

Running blastn

If you have a reference genome, you might be interested in where the enriched k-mers map to. The easiest way to check this is through a `blastn` analysis.

You first need to create a blastdb:

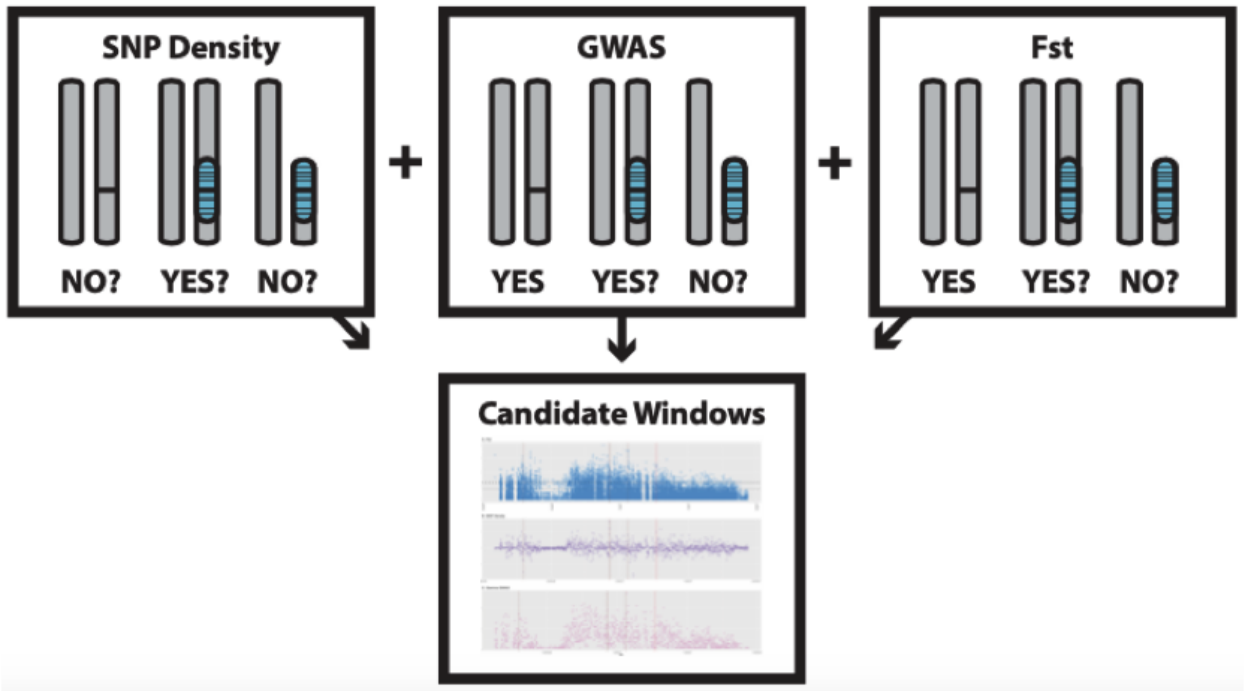
```
makeblastdb -in GCF_901000725.2_fTakRub1.2_genomic.fna -dbtype nucl
```

Then you can run blastn:

```
blastn -query fugu_plink_abyss_output.txt -db GCF_901000725.2_fTakRub1.2_genomic.fna -
↪outfmt 6
```

The output file from blastn will provide top candidate regions for each contig that was assembled from the k-mers. In the case of fugu, there are only 4 contigs that all blast to NC_042303.1, but in poplar and the golden monkey, these blast outputs were parsed with an `R` script (e.g., `poplar_kmerGWAS_blast_results.R`) to visualize which genomic regions the sex-assocaited k-mers map to. This script and the necessary input files have been included in the GitHub repo for poplar.

1.4 Step 3. Combined sequence-based analysis



If your species of interest has a genetic basis for sex determination, but you have not been able to identify a consistent signal within your genomic data using Steps 1 and 2, the scripts for Step 3, Combined sequence-based analysis, should be able to help. If you have not identified a compelling candidate by the end of Step 3, I would suggest that there is not a genetic signal for sex determination in the genomic data you have analyzed, which was the case for sea lamprey (see the main manuscript).

In Step 3, non-overlapping signals from the reference-based methods are converted to 10kb windows and combined to increase the user's power to focus on specific genomic regions. Once candidate windows are identified in Step 3, the

user can return to GWAS, SNP density, and Fst results to determine if fixed or nearly-fixed differences exist between the sexes in the top candidate windows.

Once again, the example code Fugu_SexFindR.R and necessary input files are included in the GitHub repository in order to run the analysis for fugu and generate the plot in the SexFindR figure in the manuscript.

For the GWAS and Fst input data, parsing of the input data is carried out both in the R script, and through Python scripts and bash commands.

To generate gemma_window_count.txt which is required for the R script from fugu_gemma_sort_cut.txt (which is created in the R script from test_gemma_out.assoc.txt.zip), do the following on the command line:

```
sort -k1,1 -k2n fugu_gemma_sort_cut.txt > coordinated_sorted_gemma_cut.txt
python window_count.py coordinated_sorted_gemma_cut.txt gemma_window_count.txt 10000
```

This creates a window-based count of top GWAS hits (top 5%) within each 10 kb region across the genome.

Similarly, to generate Fstugu_window_count.txt from Fstugu_sort_cut.txt (created in R from biallelic_fst.weir.fst.zip),

```
sort -k1,1 -k2n Fstugu_sort_cut.txt > coordinated_sorted_Fstugu_cut.txt
python window_count.py coordinated_sorted_Fstugu_cut.txt Fstugu_window_count.txt 10000
```

This creates a window-based count of top Fst hits (top 5%) within each 10 kb region across the genome.

The SNP Denisty data is already in 10 kb windows from the SNP Density analysis (Step 2) and is able to be used without modifications outside of R.

With all these scripts and input data, you should be able to recreate this plot from the main manuscript in R.

Figure 3. SexFindR Step 3 combined results identified the fugu SDR (red dotted line) alongside 4 additional candidate

windows (thin black dotted lines), all on NC_042303.1. Regions of high interest are darkly colored, whereas regions of low interest are colored in grey.

1.5 Supplemental Code

This section is a work in progress. Code for the analysis of lamprey and the other species from the main manuscript on the GitHub repository and documentation (when necessary or helpful) will be added.

1.5.1 Running VCF Search

The VCF Search described in the supplements was carried out on fugu as a positive control and on lamprey to look for fixed or nearly-fixed differences between males and females across the entire dataset as well as within individual lakes. In-house `Python` scripts were written for this purpose and can be found in the `VCF Search` folders for lamprey and fugu. The `Python` scripts require an uncompressed vcf, which is too large to include on GitHub, but I have subsampled the fugu vcf to include only the chromosome that contains the SDR so that the fugu script can be used for demonstation purposes.

Subsetting the fugu `vcf` file:

```
grep "^#" ~/Desktop/fugu/filtered_PASS_fugu_14M_13F.vcf > fugu_14M_13F.vcf
grep -v "^#" ~/Desktop/fugu/filtered_PASS_fugu_14M_13F.vcf | grep "NC_042303.1" >> fugu_
↪14M_13F.vcf
gzip fugu_14M_13F.vcf
```

Running VCF Search

```
gunzip fugu_14M_13F.vcf.gz
python3 fugu_vcf_search.py fugu_14M_13F.vcf out_fugu_vcf_search_test.txt
```

1.5.2 Running 100k Permutations

For SNP density, the `R` code provided runs 1000 permutations, but I wanted to run more. For fugu and lamprey, custom `Python` and `R` code was written to carry out the permuations and pivot the resulting table to a useful format. Below is the example code for the lamprey run.

```
cd ~/SexFindR/Supplemental_Code/Lamprey/SNP\ Density/
unzip SNPdensity_rows_location.txt.gz
for run in $(cat runs.txt); do python SNP_density_permuter.py SNPdensity_rows_location.
↪txt perm_${run}.txt 2500 $run & sleep 0.1; done

cat out_true_SNPden.txt perm* > true_plus_100k_perms.txt

# using R to pivot the dataframe
R
long <- read_tsv("true_plus_100k_perms.txt")
pivot <- t(long)
output <- as.data.frame(pivot) %>% rownames_to_column()
write_tsv(output,"trans_true_plus_100k_perms.txt",col_names = F)

python3 SNP_density_p_generator.py trans_true_plus_100k_perms.txt SNP_density_100k_
↪permutations_p_values.txt &
```

1.6 Supplemental Text

This page contains two sections that were not included in the SexFindR manuscript or supplements, but could be of use for others.

1.6.1 1. Genetic Sexing of the Tiger Pufferfish (fugu)

In the tiger pufferfish (*Takifugu rubripes*), it had been previously shown that a single polymorphic site within the Amhr2 gene is consistently C/C in females and C/G in males (Kamiya et al. 2012). This SNP (NC_042303.1:12708100, hereafter referred to as fugu SDR) is reportedly the only fixed genomic difference between males and females of this species and is believed to be involved in sex determination. Recently, a group studying the evolution of salinity tolerance within the pufferfish clade, released a dataset containing whole-genome re-sequencing data for 29 tiger pufferfish (Zhang et al. 2020). Although these samples were deposited onto NCBI without phenotypic sex data, the strong support for the fugu SDR allowed us to genetically sex most of the samples using the following methods and criteria. The tiger pufferfish samples (table S4) were downloaded from NCBI SRA repository using fasterq-dump and mapped to the reference genome, fTakRub1.2 (https://www.ncbi.nlm.nih.gov/genome/63), using Bowtie2. The fugu SDR and surrounding alignment was pulled from the resulting BAM files and viewed in IGV. Of the 29 samples, 13 displayed C/C genotypes for all of their reads and were classified as females for subsequent analyses. Of the remaining 16 samples, given the expected 50/50 ratio of G:C for males, samples with an excess of C or an excess of G were removed from the analysis. To this end, we classified samples as males if at least 25% of their reads were G and at least 25% of their reads were C. This resulted in confident identification of 14 males and the exclusion of 2 samples; one which had 87% G-containing reads and one that had 10% G-containing reads. No other bases were reported at the fugu SDR in any of these samples, suggesting that sequencing errors within this region were minimal for these data. The genetically-assigned sexes are found in table S5.

Although there are numerous studies that demonstrate a perfect correlation between genetic and phenotypic sex through the fugu SDR in the tiger pufferfish (Kamiya et al. 2012; Matsunaga et al. 2014), there is at least one reported population from an aquaculture center where the gonads (testis or ovary) did not agree with the fugu SDR (Matsunaga et al. 2014). The purpose of the tiger pufferfish in our study is to determine whether the methods we employ are capable of identifying fixed differences between males and females, even in a very young sex determination system where a single SNP could govern phenotypic sex. Given this purpose, it is somewhat irrelevant whether the specific fugu samples downloaded from NCBI would have also followed a perfect correlation between genetic and phenotypic sex. We are primarily using these data to validate our ability to identify sex specific sequences as small as a single SNP to examine this possibility in sea lamprey. Although we could have generated simulated data for the same purpose, the fugu data is particularly valuable given that it represents true biological data from a living species.

1.6.2 2. The Value and Utility of kmersGWAS

As described in the main paper, k-mer based analyses have the potential to be incredibly powerful for uncovering sex-specific sequences. Since k-mer based analyses do not rely on an assembled genome for read mapping, they side-step many of the issues associated with standard analyses. One of the main reasons for carrying out coverage-based and population genomic analyses in Steps 1 and 2 of SexFindR is that the amount of divergence between an X and Y (or Z and W) will dictate the power of those different methods, and by only using one, you could easily miss an important signal. Coverage methods are key if there is sufficient divergence to prevent reads from one sex chromosome from mapping to the other, whereas standard population genomics methods like FST and GWAS have power only when the reads from both sex chromosomes map equally well to the reference, so that the fixed differences between males and females can be identified.

With k-mer based methods, this boundary does not exist, because mapping is not required to identify the sex-specific sequence. We already demonstrated that kmersGWAS was capable of identifying fixed differences in homomorphic systems (poplar and fugu), but we did not look at a system where coverage-alone would be sufficient to identify the sex chromosomes. To prove the utility of kmersGWAS across the divergence continuum, we downloaded re-sequencing

data from the golden snub-nosed monkey (*Rhinopithecus roxellana*) which has the standard placental mammalian XY system with a degenerate Y chromosome (table S4).

Sex chromosomes were initially identified by running DifCover for the male reference (SRR1040959) used in genome assembly (Zhou et al. 2014) against individuals that had been previously re-sequenced and sexed (Yu et al. 2016; Kuang et al. 2019). In the ASM756505v1 assembly, (GCA_007565055.1 on NCBI), the X chromosome is NC_044555.1 (chromosome 7) and the Y chromosome is NC_044570.1 (chromosome 22) given our results (fig. S9). Sex of the remaining samples was confirmed by running samtools depth on 10kb of sequence selected from the X, the Y and an autosome (NC_044554.1). Males were expected to display similar coverage between the X and Y, but twice that level of coverage on the autosome (i.e., 1X, 1Y, 2A). Females were expected to show similar coverage between the X and autosome, but little-to-no coverage on the Y. Sexed predicted using this method were compared to those that were previously published (Kuang et al. 2019), and samples that did not agree were discarded (table S11).

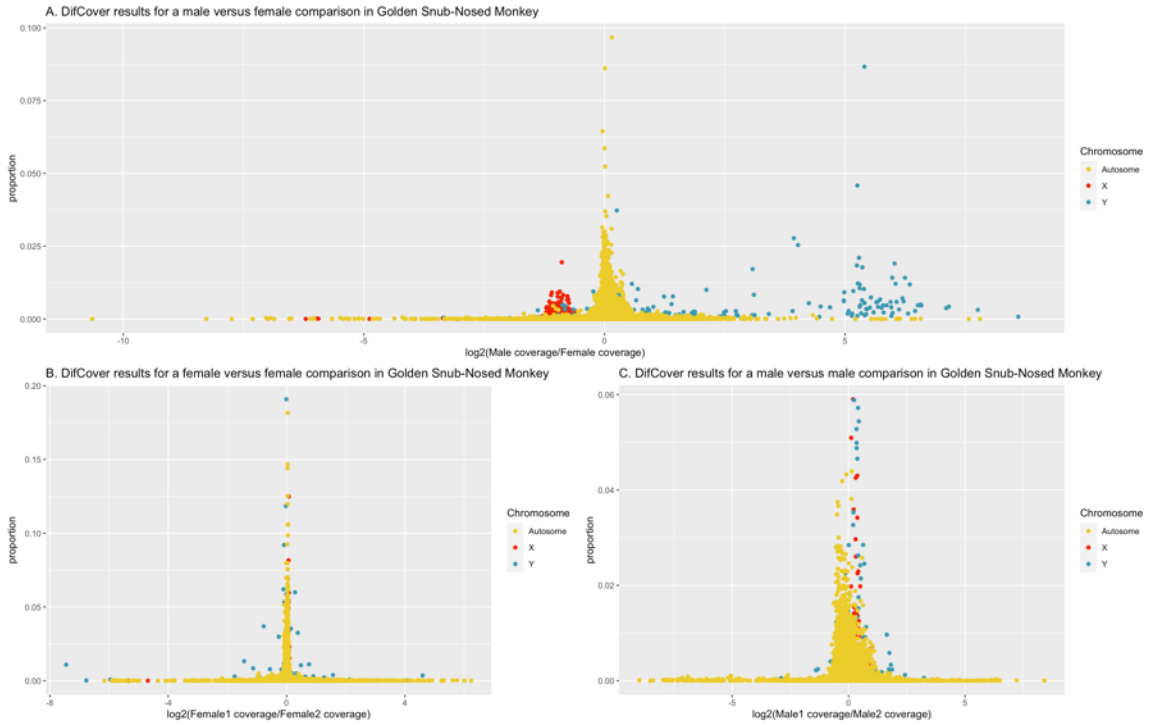


Fig. S9. DifCover analysis of the golden snub-nosed monkey. Each plot shows log2(coverage difference) on the X axis and the proportion of the full chromosome that that specific region occupies on the Y. Only full chromosomes (not scaffolds) are analyzed. The X and Y chromosomes are highlighted in red and teal, respectively, and all autosomes are coloured in yellow. A: male/female comparison with low male coverage on the X and high male coverage on the Y. B and C: female/female and male/male comparison, respectively, with no significant differences between X and Y chromosomes between the individuals of the same sex.

Once the sexing was completed, kmersGWAS was carried out using 12 males and 13 females. The top p-value achieved in this analysis was 1.537e-12 which appeared for 1,688,851 k-mers that were fixed in males and absent in females. These k-mers were assembled with ABYSS into 73,400 short contigs and blastn was used to map these contigs to the reference genome. 36,699 of these contigs found a position in the assembled genome using blastn, and this position was the Y chromosome for over 90% of the contigs (fig. S10).

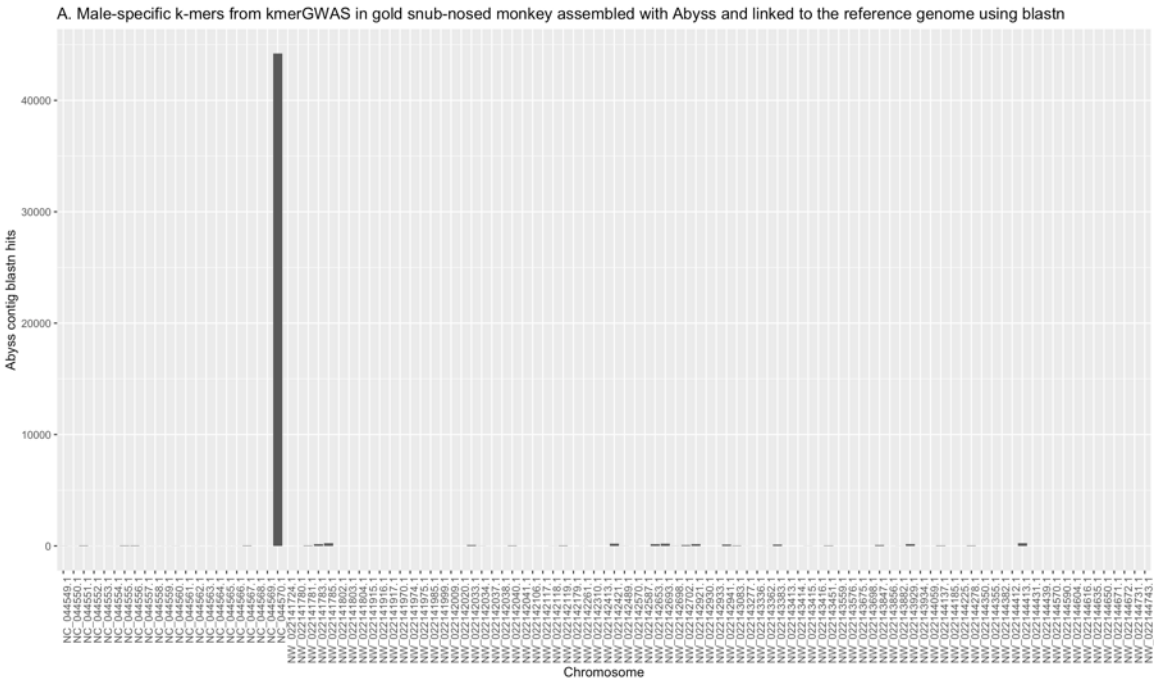


Fig. S10. kmersGWAS identified the Y chromosome in golden snub-nosed monkey within the class of k-mers that produced the smallest p-value. These male-specific k-mers map back to the Y chromosome almost exclusively.

Given this result, we have demonstrated that kmersGWAS is able to identify sex-specific across the entire divergence continuum, ranging from a single sex-specific SNP in fugu to a degenerate sex-specific Y chromosome in the golden snub-nosed monkey. kmersGWAS is a fast and straightforward method that does not rely on a reference genome and it is quite powerful for the detection of sex chromosomes.

1.6.3 References

Kamiya T, Kai W, Tasumi S, Oka A, Matsunaga T, Mizuno N, Fujita M, Suetake H, Suzuki S, Hosoya S, et al. 2012. A trans-species missense SNP in Amhr2 is associated with sex determination in the tiger Pufferfish, Takifugu rubripes (Fugu). PLoS Genet. 8(7). doi:10.1371/journal.pgen.1002798.

Kuang WM, Ming C, Li HP, Wu H, Frantz L, Roos C, Zhang YP, Zhang CL, Jia T, Yang JY, et al. 2019. The Origin and Population History of the Endangered Golden Snub-Nosed Monkey (Rhinopithecus roxellana). Mol Biol Evol. 36(3):487–499. doi:10.1093/molbev/msy220.

Matsunaga T, Ieda R, Hosoya S, Kuroyanagi M. 2014. An efficient molecular technique for sexing tiger pufferfish (fugu) and the occurrence of sex reversal in a hatchery population. :933–942. doi:10.1007/s12562-014-0768-0.

Yu L, Wang GD, Ruan J, Chen Y Bin, Yang CP, Cao X, Wu H, Liu YH, Du ZL, Wang XP, et al. 2016. Genomic analysis of snub-nosed monkeys (Rhinopithecus) identifies genes and processes related to high-altitude adaptation. Nat Genet. 48(8):947–952. doi:10.1038/ng.3615.

Zhang H, Hou J, Liu H, Zhu H, Xu G, Xu J. 2020. Adaptive evolution of low-salinity tolerance and hypoosmotic regulation in a euryhaline teleost, Takifugu obscurus. Mar Biol. 167(7):1–12. doi:10.1007/s00227-020-03705-x. https://doi.org/10.1007/s00227-020-03705-x. Zhou X, Wang B, Pan Q, Zhang J, Kumar S, Sun X, Liu Z, Pan H, Lin Y, Liu G, et al. 2014. Whole-genome sequencing of the snub-nosed monkey provides insights into folivory and evolutionary history. Nat Genet. 46(12):1303–1310. doi:10.1038/ng.3137.