Chapter 2

Materials and methods

First Draft, Sunday 17th March, 2019

2.1 Killifish husbandry and sample preparation methods

Male turquoise killifish (*Nothobranchius furzeri*, GRZ-AD strain) from a single hatching were raised under standard husbandry conditions [1] and housed from four weeks post-hatching in individual 2.8 L tanks connected to a water recirculation system. Fish received 12 h of light per day on a regular light/dark cycle, and were fed blood worm larvae and brine shrimp nauplii twice a day during the week and once a day during the weekend [1, 2].

Sacrificed fish (Table D.24) were killed by anaesthetisation in $1.5\,\mathrm{g\,L^{-1}}$ Tricaine solution in room-temperature tank water [3], then flash-frozen in liquid nitrogen and ground to a homogenous powder with a pestle in a liquid-nitrogen-filled mortar. The powder was mixed thoroughly and stored at $-80\,^{\circ}\mathrm{C}$ prior to RNA isolation.

2.2 Biochemistry and molecular biology methods

2.2.1 Standard methods

2.2.1.1 PCR

The polymerase chain reaction is a well-established method for rapid amplification of a DNA sequence through repeated cycles of denaturation, priming and replication by a high-temperature-tolerant DNA polymerase enzyme [4]. Unless otherwise specified, all PCRs in this chapter were performed using $2\times$ Kapa HiFi HotStart ReadyMix PCR Kit (Appendix A.1) according to the manufacturer's instructions. Briefly, for a 25 µl reaction, 12.5 µl Kapa ReadyMix was combined with 12.5 µl total of template, nuclease-free water, and primers; these volumes were scaled linearly for reactions of different volumes. The mixture was then heated in a thermocycler as follows:

Step	**Temperature** [°C]	**Duration** [s]	**Cycles**
Initial denaturation	95	180	1
Denaturation	98	20	
Annealing	T_a [a]	15	n_c [a]
Extension	72	t_{ext} [a]	
Final extension	72	$t_{ext} \times 4$ [a]	1

[a] Annealing temperature (T_a), extension time (t_{ext}) and cycle number (n_c) determined separately for each reaction.

2.2.1.2 Nucleic-acid purification with SeraSure magnetic beads

Nucleic-acid isolation, size-selection and concentration in the IgSeq library preparation protocol (and elsewhere where necessary) was performed using SeraSure SPRI (solid phase reversible immobilization) bead preparations [5–8]. In SPRI, paramagnetic beads bind DNA in the presence of polyethylene glycol (PEG), with the affinity of the beads for DNA depending on the concentration of PEG in the binding buffer. As a result, the range of nucleic-acid sequence lengths retained by SPRI bead purification depends primarily on the concentration of PEG, which in turn depends on the relative volume of SeraSure bead suspension added to a sample; the higher the concentration, the shorter the minimum fragment length retained during the purification process. In combination with a magnetic rack to remove the DNA-bound beads from suspension, this allows DNA of the desired size range to be isolated from a solution and resuspended in the desired volume of fresh buffer.

To prepare 50 ml of SeraSure bead suspension for DNA (or DNA:RNA heteroduplex) isolation, a stock of SeraMag beads (Appendix A.2) was vortexed thoroughly, then 1 ml was transferred to a new tube. This tube was then transferred to a magnetic rack and incubated at room temperature for 1 min, then the supernatant was removed and replaced with 1 ml TET buffer (Appendix A.3) and the tube was removed from the rack and vortexed thoroughly. This washing process was repeated twice more, for a total of three washes in TET. A fourth cycle was used to replace the TET with incomplete SeraBind buffer (iSB, Appendix A.3). The vortexed 1 ml aliquot of beads in iSB was then transferred to a conical tube containing 28 ml iSB and mixed by inversion. To add the PEG, 20 ml 50 % (w/v) PEG 8000 solution was dispensed slowly down the side of the conical tube, bringing the total volume to 49 ml. Finally, this was brought to 50 ml by adding 250 µl 10 % (w/v) Tween 20 solution and 750 µl autoclaved water to complete the SeraSure bead suspension.

To perform a bead cleanup, an aliquot of prepared SeraBind solution was vortexed thoroughly to completely resuspend the beads, then the appropriate relative volume of SeraSure suspension was added to a sample, mixing thoroughly by gentle pipetting. The sample was incubated at room temperature for 5 min to allow the beads to bind the DNA, then transferred to a magnetic rack and incubated for a further 5 min to draw as many beads as possible out of suspension. The supernatant was removed and discarded and replaced with 80 % ethanol, to a volume sufficient to completely submerge the bead pellet. The sample was incubated for 0.5-1 min, then the ethanol was replaced and incubated for a further 0.5-1 min. The second ethanol wash was removed, and the tube left on the rack until the bead pellet was almost, but not completely, dry, after which it was removed from the rack. The bead pellet was resuspended in a suitable volume of elution buffer (EB, Appendix A.3) then incubated at room temperature for at least 5 minutes to allow the nucleic-acid molecules to elute from the beads.

Unless otherwise specified, the beads from a cleanup were left in a sample during subsequent applications. To remove beads from a sample, the sample was mixed gently but thoroughly to

resuspend the beads, incubated for an extended time period (at least 10 min) to maximise nucleic-acid elution, then transferred to a magnetic rack and incubated for 2-5 min to remove the beads from suspension. The supernatant (containing the eluted nucleic-acid molecules) was then transferred to a new tube, and the beads discarded.

2.2.1.3 Phenol-chloroform extraction and ethanol precipitation of DNA

Phenol-chloroform extraction is a well-established method for removing proteins and other hydrophobic or amphipathic contaminants from nucleic-acid solutions [9]. By thoroughly mixing an aqueous solution of nucleic acid with an organic solvent (phenol), hydrophilic contaminants are dissolved into the organic phase while proteins are denatured at the organic/aqueous boundary. The addition of chloroform further encourages the denaturation of proteins, as well as improving separation of the aqueous and organic phases following centrifugation. The effect on nucleic acids depends upon the pH of the phenol: under acidic conditions, the negatively-charged DNA phosphate backbone is neutralised and so primarily dissolves in the organic phase, while RNA is kept in the aqueous phase through hydrogen bonding between water and exposed bases; under neutral or basic conditions, both DNA and RNA are retained in the aqueous phase [9].

To remove protein from isolated DNA samples, each sample was diluted to 500 µl in nuclease-free water and mixed with 500 µl of equilibrated (non-acidic) phenol:chloroform:isoamyl alcohol (PCI) mixture (Appendix A.2) in a fume hood. The sample/PCI mixture was shaken vigorously by had for 15 s to thoroughly mix the different components, then centrifuged in a benchtop centrifuge (5 min, room temperature, top speed). Again in a fume hood, the mixed sample was angled at 45°, and the upper aqueous phase containing the DNA was removed and transferred to a new tube while the lower organic phase was discarded. A second aliquot of 500 µl PCI was added and the sample was mixed, centrifuged and separated as before. Finally, in order to remove residual phenol in the sample, 500 µl pure chloroform was added to the newly-separated aqueous phase and the sample was once again mixed, centrifuged and separated.

Following this final round of separation, the DNA in the aqueous phase was precipitated by ethanol precipitation, another widely-used protocol exploiting the ability of alcohols to precipitate nucleic acids in the presence of monovalant cations [10]. 0.1 volumes of 3 M sodium acetate solution was added to each sample, followed by 2.5 volumes of fresh 100 % ethanol. The mixture was mixed gently by inversion, then incubated for 1-3 h at $-80\,^{\circ}$C or at $-20\,^{\circ}$C overnight. The suspension of precipitated DNA was pelleted through centrifugation in a benchtop centrifuge (30 min, 4 °C, top speed). The supernatant was discarded and replaced with 500 µl chilled 70 % ethanol, and the sample centrifuged again (5 min, 4 °C, top speed). After this, the supernatant was again discarded, and the samples allowed to air-dry before being resuspended in 30-50 µl EB (Appendix A.3).

2.2.1.4 Guanidinium thiocyanate-phenol-chloroform extraction of RNA

Acid guanidinium thiocyanate-phenol-chloroform extraction is a technique for purifying RNA samples closely related to the phenol-chloroform-extraction method described in Section 2.2.1.3 [9]. By using acid rather than equilibrated phenol, DNA in the sample is dissolved in the organic rather than the aqueous phase and so is removed from the aqueous solution along with proteins and other contaminants [9, 11]. Meanwhile, the addition of guanidinium thiocyanate, a chaotropic agent which disrupts hydrogen bonds in aqueous solution, strongly encourages the rapid denaturation of proteins and so helps protect the RNA from degradation by RNase enzymes [9, 11].

To isolate total RNA from homogenised killifish tissues, 1 ml of QIAzol lysis reagent (Appendix A.2, containing acid phenol and guanidinium thiocyanate) was added to 0.1 *g* of tissue, mixed gently but thoroughly by inversion, then incubated at room temperature for 5 min to allow the QIAzol to penetrate the tissue. 0.2 volumes of chloroform was added and the mixture was shaken vigorously for 15 s, then incubated at room temperature for 3 min. The mixture was then centrifuged (15 min, 4 °C, 12 000 *g*). Angling the tube at 45°, the upper aqueous phase containing the RNA was removed and transferred to a new tube, while the lower organic phase was discarded.

Following phase separation, the RNA was precipitated using isopropanol precipitation, which works on the same principles as ethanol precipitation but requires lower relative volumes of alcohol [10]. 0.5 volumes of room-temperature isopropanol was added to each sample, mixing gently by inversion and incubating for 5 min at room temperature. The suspension was centrifuged (10 min, 4 °C, 12 000 *g*) and the supernatant discarded. 1 volume of freshly prepared 75 % ethanol was added and the tube was vortexed briefly and centrifuged again (5 min, 4 °C, 7500 *g*). The supernatant was discarded and the RNA pellet allowed to air-dry for 5-10 min, then resuspended in 50 µl EB (Appendix A.3). The concentration and quality of the resulting total-RNA solution were assayed with the Qubit 2.0 flourometer (Thermo Fisher, RNA BR assay kit) and TapeStation 4200 (Agilent, RNA tape), respectively, according to the manufacturer's instructions.

2.2.2 Library size-selection with the BluePippin

The BluePippin (Sage Science) is a DNA size-selection system based on agarose gel electrophoresis, which uses timed switching between positively-charged electrodes at a forked gel channel in an agarose cassette to redirect DNA of a desired size range into a separate lane from the rest of the sample [12]. The timing of switching is determined based on the size range input by the user and calibrated using flourescent internal standards, which are added to the sample during the sample preparation process and designed to run well ahead of the possible size ranges for that cassette type. The combination of the choice of cassette and the choice of standards determines which fragment lengths can be effectively isolated using the machine.

For the experiments described in this thesis, a 1.5 % cassette with R2 markers were used, enabling size selection of targets in the range of 250–1500 bp [12]. Machine calibration and testing, cassette preparation, and protocol design were performed in accordance with the BluePippin documentation and instructions given by the machine software. During this process, the elution wells of the lanes to be used in the size-selection run were emptied and refilled with 40 µl of electrophoresis buffer (Appendix A.2), then sealed for the duration of the run, and a broad-range size-selection protocol with a target range of 400 to 800 bp was specified. 30 µl of sample was then combined with 10 µl of loading solution (Appendix A.2) and vortexed to mix, then 40 µl of buffer was removed from the appropriate loading well and replaced, slowly, with the prepared sample mixture. The protocol was started and run until the final elution was complete. Finally, the eluted samples were removed from the elution wells of the appropriate lanes, and the unused lanes of the cassette were re-sealed for future use.

2.2.3 BAC isolation and sequencing

All BAC clones that were sequenced for this research were provided by the FLI in Jena as plate or stab cultures of transformed *E. coli*, which were replated and stored at 4 °C when not in use. Prior to isolation, the clones of interest were cultured overnight in at least 100 ml LB medium to produce a large liquid culture. The cultures were then transferred to 50 ml conical tubes and centrifuged (10-25 min, 4 °C, 3500 *g*) to pellet the cells. After pelleting, the supernatant was carefully discarded and the cells were resuspended in 18 ml buffer P1 (Appendix A.3).

After resuspension, the cultures underwent alkaline lysis [13] to release the BAC DNA and precipitate genomic DNA and cellular debris. 18 ml lysis buffer P2 (Appendix A.3) was added to each tube, which was then mixed gently but thoroughly by inversion and incubated at room temperature for 5 min. 10 ml ice-chilled neutralisation buffer P3 (Appendix A.3) was added to precipitate genomic DNA and cellular debris, and each tube was mixed gently but thoroughly by inversion and incubated on ice for 15 min. The tubes were then centrifuged (20-30 min, 4 °C, 12 000 *g*) to pellet cellular debris and the supernatant transferred to new conical tubes. This process was repeated at least two more times, until no more debris was visible in any tube; this repeated pelleting was necessary to minimise contamination in each sample, as the normal column- or paper-based filtering steps used during alkaline lysis protocols result in the loss of the BAC DNA.

Following alkaline lysis, the BAC (and residual genomic) DNA in each sample underwent isopropanol precipitation: 0.6 volumes of room-temperature isopropanol was added to the clean supernatant in each tube, followed by 0.1 volumes of 3 M sodium acetate solution. Each tube was mixed well by inversion, incubated for 10-15 min at room temperature, then centrifuged (30 min, 4 °C, 12 000 *g*) to pellet the DNA. The supernatant was discarded and the resulting DNA smear was "resuspended" in 1 ml 100 % ethanol and transferred to a 1.5 ml tube, which was re-centrifuged (5 min, 4 °C, top speed) to obtain a concentrated pellet. Finally, the peletted samples were resuspended in EB

(Appendix A.3) and purified of proteins and RNA using phenol:chloroform extraction and ethanol precipitation (Section 2.2.1.3).

The resuspended BAC isolates were sent to the Cologne Center for Genomics, where they underwent Illumina Nextera XT library preparation and were sequenced on an Illumina MiSeq sequencing machine (MiSeq Reagent Kit v3, 2x300bp reads).

2.2.4 Immunoglobulin sequencing of killifish samples

2.2.4.1 RNA template quantification and quality control

Total RNA from whole-body killifish samples was isolated as described in Section 2.2.1.4; gut RNA from microbiota transfer experiments [2] was already prepared and available. Quantification of RNA samples was performed with the Qubit 2.0 flourometer (RNA BR assay kit), while quality control and integrity measurement was performed using the TapeStation 4200 (RNA tape), both according to the manufacturer's instructions.

2.2.4.2 Reverse-transcription and template switching

Reverse transcription of total RNA and template switching for IgSeq library preparation was performed using SMARTScribe Reverse Transcriptase (Appendix A.1), in line with the protocol specified in [14] (Procedure, steps 5-9). Briefly, 750 ng total RNA from a killifish sample was combined with 2 µl 10 µM gene-specific primer (GSP), homologous with the second CH exon of *N. furzeri IGHM* (Appendix B.2, designed using Primer3 [15]). The reaction volume was brought to a total of 8 µl with nuclease-free water, and the resulting mixture was incubated for 2 minutes at 70 °C to denature the RNA, then cooled to 42 °C to anneal the GSP [14].

Following annealing, the RNA-primer mixture was combined with 12 µl of reverse-transcription master-mix (Table 2.1), including the reverse-transcriptase enzyme and template-switch adapter (Appendix B.1, sequence provided in [14]). The complete reaction mixture was incubated at for 1 h at 42 °C for the reverse-transcription reaction, then mixed with 1 µl of uracil DNA glycosylase (UDG, Appendix A.1) and incubated for a further 40 min at 37 °C to digest the template-switch adapter. Finally, the reaction product was purified using SeraSure beads (Section 2.2.1.2) at 0.7 × concentration, eluting in 16.5 µl clean elution buffer (EB, Appendix A.3).

2.2.4.3 PCR amplification and adapter addition

Following reverse-transcription, UDG digestion, and cleanup, the reaction mixture underwent three successive rounds of Kapa PCR (Section 2.2.1.1, Table 2.2) each of which was followed by a further round of bead cleanups (Section 2.2.1.2, Table 2.3). The first of these PCR reactions added a second

Table 2.1: Master-mix components for SMARTScribe reverse transcription, per sample

Volume [µl]	Component	Concentration	Reference
2	SMARTScribe reverse transcriptase	$100\,\mathrm{U\,\mu l^{-1}}$	Appendix A.1
4	SMARTScribe first-strand buffer	$5\times$	Appendix A.2
2	SmartNNNa barcoded TSA	10 µM	Appendix B.1
2	DTT	20 mM	Appendix A.2
2	dNTP mix	10 µM per nucleotide	Appendix A.2
0.5	RNasin RNase inhibitor	$40\,\mathrm{U\,\mu l^{-1}}$	Appendix A.1

strand to the reverse-transcribed cDNA and amplified the resulting DNA molecules; the second added partial Illumina sequencing adapters and further amplified the library, and the third added complete Illumina adapters (Appendix B.3, including i5 and i7 indices [16]). Primer sequences (Appendix B.2) homologous to the template-switch adapter (M1SS and M1S) were provided by [14], while those homologous to the $C_\mu 1$ constant-region exon (IGHC-B and IGHC-C) were designed using Primer3 [15].

Table 2.2: Details of PCR protocols for *N. furzeri* immunoglobulin sequencing

PCR	Protocol details			Primers		Volumes (µl)[b]			
	# cycles	T_m (°C)	t_{ext} (s)	F	R	Template	Primers (each)	Kapa	H_2O
1	18	65	15	IGHC-B	M1SS	10.5	1 (×2)	12.5	0
2	13	65	15	M1S+P2	IGHC-C+P1	1	0.5 (×2)	12.5	10.5
3	7 to 12[c]	**68**	15	D50*[a]	D7**[a]	2	**0.75** (×2)	12.5	9

[a] PCR3 primers selected as appropriate for each library's assigned indices; see Appendix B.3 for more information.
[b] If the number of samples to be sequenced was small, all volumes of PCR 3 were doubled for a 50 µl total PCR volume.
[c] See Table 2.4 for specific cycle numbers used in each experiment.

2.2.4.4 Library pooling, size selection and sequencing

Following PCR3 and its attendant bead cleanup, the total concentration of each library was assayed with the Qubit 2.0 (DNA HS assay kit), while the size distribution of each library was obtained using the TapeStation 4200 (D1000 tape). To obtain the concentration of complete library molecules in each case (as opposed to primer dimers or other off-target bands), the ratio between the concentration of the desired library band (c. 620-680 bp) and the total concentration of the sample in the TapeStation assay was calculated for each TapeStation lane, and the total concentration of each library as measured by the Qubit was multiplied by this number to obtain an estimate of the desired figure:

Table 2.3: Details of bead cleanups during *N. furzeri* immunoglobulin sequencing

Stage[a]	**Sample volume**	**Beads volume (µl)**		**Elution volume (µl)**[c]
		µl	×[b]	
RT	21	14.7	0.7	16.5
PCR 1	25	17.5	0.7	25
PCR 2	25	17.5	0.7	15
PCR 3	25[d]	20[d]	0.8	15[d]
Pooling	Varies[e]	Varies[e]	2.5	35

[a] Each bead cleanup takes place immediately *after* its corresponding stage.
[b] Bead volumes are usually given as multiples of the sample volume.
[c] All elutions performed in the specified volume of elution buffer (EB, Appendix A.3).
[d] If PCR 3 reaction volume differs from 25 µl, bead and elution volumes are rescaled proportionally to sample volume as appropriate.
[e] Samples are pooled in equimolar ratio.

Table 2.4: PCR cycle numbers used for *N. furzeri* immunoglobulin sequencing

Experiment	**Number of PCR cycles**			**Reference**
	PCR 1	PCR 2	PCR 3	
Pilot	18	13	9	Section 4.4
Ageing	18	13	8	Section 4.5
Gut	18	13	12	Section 4.6

$$\text{Library concentration}(L) = \text{Qubit concentration} \times \frac{\text{TapeStation concentration [main band]}}{\text{TapeStation concentration [total]}} \quad (2.1)$$

All the libraries for a given experiment were then pooled, such that the estimated concentration L of each library in the final pooled sample was equal and the total mass of nucleic acid in the pooled sample was at least 240 ng. The pooled libraries underwent a final bead cleanup (Section 2.2.1.2, Table 2.3) to concentrate the samples, then underwent size selection with the BluePippin (Section 2.2.2, 1.5 % DF Marker R2, broad 400-800bp). The size-selected pooled samples underwent a final round of quality control (as above) to confirm their collective concentration (at least 1.5 nM) and size distribution (one peak at c. 620-680 bp). Finally, the pooled and size-selected libraries were sequenced externally on an Illumina MiSeq System (MiSeq Reagent Kit v3, 2x300bp reads, 30% PhiX spike-in), either at the Cologne Center for Genomics (pilot and ageing experiments) or with Admera Health (gut experiment).

2.3 Computational and analytic methods

Version details for software used in this section are given in Table D.1.

2.3.1 General data processing and pipeline structure

Unless otherwise specified, processing and analysis of biological data was performed using standard Bioconductor [17] packages: Biostrings [18] and BSgenome [19] for biological sequence data, GenomicRanges [20] for sequence ranges, and genbankr [21] and rentrez [22] for GenBank files.

Smith-Waterman and Needleman-Wunsch exhaustive alignments [23, 24] were performed using the `pairwiseAlignment` function from Biostrings; percentage sequence identities were computed using the `pid` function from the same package.

Processing of tabular data was performed using the Tidyverse suite of tools, especially readr [25], dplyr [26], tidyr [27] and stringr [28]. Snakemake [29] was used to design and run data-processing pipelines.

Unless otherwise specified, standard statistical operations and comparisons were performed using built-in functions from the stats package in R [30]: `wilcox.test` for two-sample Mann-Whitney *U* tests, `kruskal.test` for multisample Kruskal-Wallis analysis-of-variance tests, `lm` for linear models, `glm` for generalised linear models, `cor` for Pearson product-moment correlation coefficients, `hclust` for hierarchical clustering, and `pcoa` for principal co-ordinate analysis.

2.3.2 Data visualisation

Unless otherwise specified, data were visualised using ggplot2 [31]. Chromosome ideograms, locus structure visualisations, and sashimi plots were constructed using Gviz [32]. Cluster dendrograms and phylogenetic trees were drawn with ggtree [33], using utilities from ape [34] and tidytree [35]. Sequence logos were drawn with ggseqlogo [36].

2.3.3 BAC insert assembly

2.3.3.1 Identifying BAC candidates for the *Nothobranchius furzeri IGH* locus

The first group of candidate BAC clones to be used in the *N. furzeri* locus assembly was identified by searching for scaffolds in a previous assembly of the *N. furzeri* genome (`NotFur1`, GenBank accession GCA_000878545.1 [37]) that contained either *IGH* gene fragments (`GapFilledScaffold_8761`, `8571`, `16121`) or genes homologous to those flanking the *IGH* locus in stickleback and medaka (GapFilledScaffold_2443 and 292). Subsequences from these scaffolds were sent to Kathrin Reichwald at

the FLI in Jena, who identified four BAC clones (193A03, 276N03, 209K12, 181N10) with sequenced ends close to the query sequences.

Following sequencing and assembly of these BAC inserts, a further group of BACs was identified using a second, independent genome assembly (GenBank accession GCA_001465895.2, [38]) and the database of BAC end sequences, which by then were publicly available. The assembled BAC sequences were found to map within or near a large, gapped region on synteny group 3 of this genome assembly, and BACs were selected that either intruded into this gapped region or had end sequences that mapped to another scaffold aligning to the assembled BAC inserts (scaffold01427, scaffold02214, scaffold01820). In total, 11 further BACs were sequenced and assembled in this second round (223M21, 162F04, 220O06, 248A22, 165M01, 206K13, 154G24, 208A08, 277J10, 109B21, 216D12).

2.3.3.2 Sequence trimming, filtering and correction

Demultiplexed, adapter-trimmed MiSeq sequencing data were uploaded by the sequencing provider to Illumina BaseSpace and accessed via the Illumina utility program BaseMount. Reads from each library were trimmed with `Trimmomatic` [39] to remove adapter sequences, trim low-quality sequence, and discard any trimmed reads below a minimum lengh:

```
trimmomatic PE -phred33 <forward_reads_fastq>
   ↪ <reverse_reads_fastq> <output_paths>
   ↪ ILLUMINACLIP:<adapter_directory>/TruSeq3-PE.fa:2:30:10
   ↪ LEADING:20 TRAILING:20 SLIDINGWINDOW:4:30 MINLEN:36
```

Following this, the trimmed reads were filtered to remove *E. coli* genomic DNA and other contaminants by aligning them using `Bowtie2` [40] and retaining read pairs that did not align concordantly:

```
bowtie2 --very-sensitive-local --local --reorder --un-conc
   ↪ <output_prefix> -x <ecoli_genome_index_path> -1
   ↪ <forward_reads_fastq> -2 <reverse_reads_fastq> -S
   ↪ <sam_file_prefix>
```

Before sequence assembly, the filtered reads then underwent correction, to reduce the impact of errors occurring during the library preparation and sequencing process. In order to increase the reliability of the resulting scaffolds and reduce the impact of ideosyncracies of any given correction tool, the reads were corrected in parallel using two different programs; QuorUM [41]:

```
quorum -d -q "33" -p <output_path> <interleaved_reads_files>
```

and BayesHammer (the built-in correction tool of the SPAdes genome-assembly software [42, 43]):

```
spades.py -1 <forward_reads_fastq> -2 <reverse_reads_fastq> -o
   ↪ <output_path> --disable-gzip-output
   ↪ --only-error-correction --careful --cov-cutoff auto -k
   ↪ 21,33,55,77,99,127 --phred-offset 33
```

2.3.3.3 Sequence assembly and scaffolding

Each pair of independently-corrected reads files was then passed to SPAdes [42] for *de novo* genome assembly:

```
spades.py -1 <forward_reads_fastq> -2 <reverse_reads_fastq> -o
   ↪ <output_path> --disable-gzip-output --only-assembler
   ↪ --careful --cov-cutoff auto -k 21,33,55,77,99,127
   ↪ --phred-offset 33
```

Following assembly, any *E. coli* scaffolds resulting from residual contaminating reads were identified by aligning scaffolds to the *E. coli* genome using BLASTN [44, 45], and scaffolds containing significant matches were discarded. The remaining scaffolds were then scaffolded using SSPACE [46], using jumping libraries from the two killifish genome assemblies mentioned in Section 2.3.3.1 [37, 38]:

```
SSPACE_Standard_v3.0.pl -x 0 -k 5 -a 0.7 -n 15 -z 200 -g 1 -p 0
   ↪ -l <jumping_library_config_file> -s
   ↪ <spades_scaffolds_file>
```

In order to guarantee the reliability of the assembled scaffolds, the assemblies produced with BayesHammer- and QuorUM-corrected reads were compared, and scaffolds were broken into segments whose contiguity was agreed on between both assemblies. To integrate these fragments into a contiguous insert assembly, points of agreement between BAC assemblies from the same genomic region (e.g. two scaffolds from one assembly aligning concordantly to one scaffold from another) and between BAC assemblies and genome scaffolds, were used to combine scaffolds where possible. Any still-unconnected scaffolds were assembled together through pairwise end-to-end PCR (Section 2.2.1.1, with one primer each on the end of each scaffold designed using Primer3 [15]) and Sanger sequencing (Eurofins).

2.3.4 Locus characterisation and assembly

2.3.4.1 Collating reference sequences

Most publications presenting characterisations of *IGH* loci do not provide easy-to-use databases of trimmed and curated gene segments, and the data that is available is often partial and heterogeneous between publications. In order to obtain standardised reference databases for locus characterisation, further analysis was performed on publically-available data from three reference species with previously-characterised *IGH* loci: medaka (*Oryzias latipes*) [47], zebrafish (*Danio rerio*) [48] and three-spined stickleback (*Gasterosteus aculeatus*) [49, 50], as described below. Following automatic sequence extraction, the reference sequences were checked manually for any severely pathological (e.g. out-of-frame) sequences and edited before being used for inference in novel loci.

Medaka

GenBank files of the annotated medaka *IGH* locus were downloaded from the supplementary information of the medaka locus paper ([47], additional file 6) and corrected to make them parsable by genbankr. Locus sequence and annotation ranges were extracted from these GenBank files into FASTA and tab-separated tabular formats, respectively, and segment annotations were renamed to match the naming conventions used in other species. VH, DH, JH and constant-region-exon nucleotide sequences were extracted from the locus sequence using these annotations. Amino-acid sequences for VH, JH and constant-region sequences were obtained automatically by identifying the reading frames which minimised the number of STOP codons in each sequence.

Stickleback

Limited sequence information on the *IGH* locus in stickleback, including VH segments and bulk (non-exon-separated) constant regions was provided in a GenBank file in the locus characterisation paper for medaka ([47], additional file 6), while additional sequence information (including DH and JH nucleic-acid sequences and amino-acid sequences of constant-region exons) was extracted manually from one of the stickleback locus papers ([49], Figure S1 to S4) into FASTA files. As with medaka, the GenBank reference file was downloaded, corrected and parsed to yield a FASTA file of the locus sequence and tab-separated tabular files of annotation ranges. VH sequences were extracted from the locus sequence using these annotation ranges and translated as specified for medaka above; JH sequences provided by [49] were translated such that the final nucleotide formed the last position of the final codon.

To obtain nucleic-acid sequences of the constant-region exons, the amino-acid sequences from [49] were aligned to the locus sequence with TBLASTN [51], with a query coverage threshold of 40% and a maximum of three HSPs per query sequence:

```
tblastn -query <ch_aa_fasta> -subject <gac_locus_fasta>
   ↪ -qcov_hsp_perc 40 -max_hsps 3 -outfmt '<output_format>' >
   ↪ <output_path>
```

with the following standardised tabular output format:

```
6 qseqid sseqid pident qcovhsp length mismatch gapopen gaps
   ↪ sstrand qstart qend sstart send evalue bitscore qlen slen
```

To filter out alignments across subloci, any alignment of an exon upstream of the annotated boundaries of its corresponding bulk constant region (whose ranges were specified in the GenBank file) was discarded; the alignment with the highest score for each exon was then used to extract the corresponding nucleic-acid sequence from the locus. In order to control for any errors, either during manual extraction of locus sequences from the paper or in the paper itself, these nucleic-acid sequences were then re-translated to generate new amino-acid sequences, again using the translation frame producing the fewest STOP codons; these sequences were then used in place of the reference files in downstream applications.

Zebrafish

GenBank files corresponding to the zebrafish *IGH* locus were provided (without segment annotations) on GenBank by [48]; this publication also provided detailed co-ordinates for the VH, DH and JH segments (but not constant exons) on these sequences. Aligned amino-acid sequences were provided for the exons of *IGHM* and *IGHZ*, but no detailed information about *IGHD* exons could be found for these sequences; as a result, reference information about *IGHD* was not used from this species.

As with stickleback, the amino-acid sequences provided were aligned to the locus sequences with TBLASTN to identify and extract exon nucleic-acid sequences, which were then translated using the frame yielding the fewest STOP codons for each sequence. VH sequences were obtained using the ranges provided in [48] and translated in the same manner. DH and JH nucleotide sequences were obtained directly from [48]; as with stickleback, JH amino-acid sequences were obtained by translating the nucleotide sequences in the frame such that the final nucleotide formed the last position of the final codon.

2.3.4.2 Identifying putative locus sequences

In order to identify sequences in a genome assembly potentially containing part of an *IGH* locus, reference VH, JH and constant-region nucleotide and amino-acid sequences were mapped to the assembly using BLAST [44, 45]. Nucleotide sequences were aligned to the locus using the relatively permissive blastn algorithm (as opposed to e.g. megablast or dc-megablast):

```
blastn -tastk blastn -query <reference_exon_fasta> -subject
   ↪ <locus_fasta> -outfmt "<output_format>"
```

Protein sequences, meanwhile, were aligned using the standard blastp algorithm:

```
\noindent blastp -query <reference_exon_fasta> -subject
   ↪ <locus_fasta> -outfmt '<output_format>'
```

In both cases, the tabular output format specified in Section 2.3.4.1 was used, to provide a predictable format for downstream processing of BLAST alignment tables.

Following alignment of reference sequences, overlapping alignments to reference segments of the same segment type, isotype (if applicable) and exon number (if applicable) were collapsed together, keeping track of the number of collapsed alignments and the best E-values and bitscores obtained for each alignment group. Alignment groups with a very poor maximum E-value (> 0.001) were discarded, as were groups consisting of fewer than two alignments and groups overlapping with a much better alignments to a different sequence type, where "much better" was defined as a bitscore difference of at least 33. Following resolution of conflicts, VH and CH alignments underwent a second filtering step of increased stringency, requiring a minimum E-value of 10^{-10} to be retained.

Following alignment filtering, scaffolds containing surviving alignments to at least two distinct segment types (where VH, JH, and each type of constant-region exon each counted as one segment type), or alignments to one segment type covering at least 1% of the scaffold's total length were retained as potential locus scaffolds. To reduce computational runtime spent processing irrelevant sequence on long scaffolds, each candidate scaffold so identified was trimmed to 100kb before the first putative gene segment and 100kb after the last one; in the case of *Nothobranchius furzeri* and *Xiphophorus maculatus*, these ranges were further reduced following more thorough segment characterisation (Section 2.3.4.4).

The exact reference sequences used for this extraction process differed depending on the genome being analysed. For *Nothobranchius furzeri* (Section 3.2), the reference sequences extracted from medaka, stickleback and zebrafish (Section 2.3.4.1) were used; for *Xiphophorus maculatus* (Section 3.3), gene segments inferred for *N. furzeri* were also included; and for other species (Section 3.4), the reference sequences plus those inferred for both *N. furzeri* and *X. maculatus* were used.

2.3.4.3 Locus sequence finalisation

In the case of both *Nothobranchius furzeri* and *Xiphophorus maculatus*, a single chromosome (chromosome 6 in *N. furzeri*, chromosome 16 in *X. maculatus*) was identified as bearing the *IGH* locus in that species. In the case of *X. maculatus*, this was the only segment-bearing scaffold identified in the genome, and the completed locus sequence was obtained by simply trimming the chromosomal sequence at either end of the segment-bearing region. In contrast, multiple scaffolds from the *N.*

furzeri genome were also identified as bearing at least one potential *IGH* segment (Table 3.1). In order to identify which of these were in fact part of the locus and integrate them into a contiguous sequence, BAC candidates identified and assembled as described in Section 2.3.3 were incorporated into the assembly.

To do this, all assembled BAC inserts were screened for *IGH* locus segments in the same manner described for genome scaffolds (Section 2.3.4.2). Passing BACs (Table 3.2) were aligned to the candidate genome scaffolds with BLASTN and integrated manually together, giving priority in the event of a sequence conflict to (i) any sequence containing a gene segment missing from the other, and (ii) the genome scaffold sequence if neither sequence contained such a segment. BACs and scaffolds which could not be integrated into the locus sequence in this way were discarded as orphons.

2.3.4.4 Locus segment characterisation

Detailed characterisation of *IGH* gene segments was performed on finished *IGH* locus sequences for *Xiphophorus maculatus* and *Nothobranchius furzeri*, and on isolated candidate scaffolds for other species, using the same reference segment databases used to identify candidate scaffolds for that species in Section 2.3.4.2. The specific methods used depended on segment type.

VH

To identify VH segments on newly characterised loci, reference VH segments were used to construct a multiple-sequence alignment with PRANK [52]:

```
prank -d=<reference_vh_db> -o=<output_path> -gaprate=0.00001
   ↪ -gapext=0.00001 -F -termgap
```

The resulting alignment was used as an input to NHMMER [53–56], which constructs a Hidden Markov Model from a multiple-sequence alignment and uses it to identify matching sequences in a reference sequence:

```
nhmmer --dna --notextw --tblout <output_path> -T 80
   ↪ <vh_alignment> <locus_sequence_path>
```

where `-T 80` specified the minimum alignment score required to report a match. The resulting match table was used to identify candidate ranges in the locus sequence corresponding to VH segments; these ranges were extended by 9bp at either end to account for boundary errors, and the corresponding nucleotide sequences were extracted to a FASTA file. Each sequence was then checked and refined manually: 3' ends were identified by the start of the RSS heptamer sequence (consensus `CACAGTG` hesse1989rss), if present, while 5' ends and FR/CDR boundaries were identified using IMGT/DomainGapAlign [57] with the default settings. Where necessary, IMGT/DomainGapAlign was also used to IMGT-gap the VH segments in accordance with the IMGT unique numbering [58].

An initial amino-acid sequence for each VH segment was produced automatically from the extracted nucleotide sequence by identifying the reading frame which minimised the number of STOP codons in the sequence; this worked well for most segments. VH amino-acid sequences were then refined (and in a few cases re-translated) using the manually-refined nucleotide sequences, including end-refinement and FR/CDR boundary identification.

Following extraction and manual curation, VH segments were grouped into families based on their pairwise sequence identity. In order to assign segments to families, the nucleotide sequence of each VH segment in a locus was aligned to each other segment using Needleman-Wunsch global alignment, and the resulting matrix of pairwise sequence identities was used to perform single-linkage heirarchical clustering on the VH segments. The resulting dendrogram was cut at 80% sequence identity to obtain VH families. These families were then numbered based on the order of the first-occurring VH segment from that family in the first *IGH* sublocus in which the family is represented, and each VH segment was named based on its parent sublocus, its family, and its order among elements of that family in that sublocus (**????????????**).

JH

As with VH segments, JH segments were identified by building a multiple-sequence alignment with PRANK and using it to construct an HMM with NHMMER; the parameters used were the same as for VH segments, except that there was no minimum score for NHMMER to report a sequence match (`-T 0` instead of `-T 80`). The resulting sequence ranges were extended by 20 bp on either end and extracted into FASTA format. These sequences were then trimmed automatically by identifying the RSS heptamer sequence at the 5' end and the splice junction motif (`GTA`) at the 3' end, then checked and refined manually.

IgBLAST [59] identifies CDR3 boundaries for recombined IGH VDJ sequences using an auxiliary file specifying the reading frame of each JH segment, along with the co-ordinate of the conserved `TGG` codon (corresponding to the conserved W118 residue in the recombined sequence [60]) marking the CDR3/FR4 boundary. An auxiliary file for the inferred JH segments was generated automatically by searching for the conserved sequence using a series of regular-expression patterns of decreasing stringency (Table 2.5), taking the first match in each sequence as the desired residue; this determined both the reading frame and the W118 sequence co-ordinate. Once generated, the auxiliary file was then used to determine the reading frame for automatically translating the JH sequences; both the auxiliary file and amino-acid FASTA file were then edited to incorporate any manual refinements made to the JH nucleotide sequences.

Table 2.5: Regex patterns used to search for conserved W118 residues in JH sequences during AUX file generation

#	Pattern
1	TGGGBNNNNGBN
2	TGGGBNNNGBN
3	TGGGBNNNNNGBN
4	TGGGBNNNNNNGBN
5	TGGGBN

Curated JH sequences were named based on their order within their parent sublocus and, where applicable, on whether they were upstream of IGHZ or IGHM constant regions (Tables D.8 and D.19).

DH

Unlike VH and JH gene segments, DH segments are too short and variable to be found effectively using an HMM-based search strategy. Instead, DH segments in assembled loci were located using their distinctive pattern of flanking recombination signal sequences (Section 1.2.2): an antisense RSS in 5', then a short D-segment, then a sense RSS in 3'. Potential matches to this pattern were searched for using FUZZNUC from the EMBOSS collection of bioinformatics tools [61], with a high error tolerance to account for deviations from the conserved sequence in either or both of the RSSs:

```
fuzznuc -pattern
   ↪ 'GGTTTTTGTN(10,14)CACTGTGN(1,25)CACAGTGN(10,14)ACAAAAACC'
   ↪ -pmismatch 8 -rformat gff -outfile <output_path>
   ↪ <locus_sequence_path>
```

This generated a GFF file [62] of permissive matches, representing potential DH segments; these were then grouped by sequence co-ordinate, and higher-mismatch candidates overlapping with a lower-mismatch alternative were discarded.

Orientation of DH segments based on their own sequence is challenging, as the segments themselves have no clear conserved structure and the flanking RSSs are rotationally symmetric. To overcome this problem and orientate the DH segments on the locus, the table of DH candidate ranges was combined with previously-identified (and easier to orientate) VH and JH ranges. Each DH candidate was then orientated based on the orientations of its flanking segments: segments with an oriented segment immediately upstream or downstream adopted the orientation of that segment, while segments with contradictory orientation information were discarded. This process was repeated until all DH segments had either been orientated or discarded.

After orientation, the DH ranges were used to extract DH sequences in FASTA format from the locus sequence; these sequences then underwent a second, more stringent filtering step, in which sequences lacking the most conserved positions in each RSS [63] were discarded [64]:

```
grep -B 1
   ↪ '[ACTG]\{{25,27\}}TG[ACTG]\{{1,25\}}CA[ACTG]\{{25,27\}}'
   ↪ <dh_fasta> | sed '/^--$'/d > <output_fasta>
```

Finally, the identified DH candidates were checked manually, candidates without good RSS sequences were discarded, and flanking RSS sequences were trimmed to obtain the DH segment sequences themselves. As with JH, these were numbered based on their order within their parent sublocus and, when applicable, on whether they were upstream of IGHZ or IGHM constant regions (Tables D.5 and D.17).

CH

To detect and identify constant-region exons in the characterised loci, constant-region nucleotide and protein sequences from reference species were mapped to the locus sequence using BLAST [44, 45], in the same manner described for putative locus scaffolds in Section 2.3.4.2. Following alignment of reference sequences, overlapping alignments to reference segments of the same isotype and exon number were collapsed together, keeping track of the number of collapsed alignments and the best E-values and bitscores obtained for each alignment groups. Alignment groups with a very poor maximum E-value (> 0.001) were discarded, as were groups overlapping with a much better alignments to a different isotype or exon type, where "much better" was defined as a bitscore difference of at least 16.5. Where conflicting alignments to different isotypes or exon types co-occurred without a sufficiently large difference in bitscore, both alignment groups were retained for manual resolution of exon identity.

Following resolution of conflicts, alignment groups underwent a second filtering step of increased stringency, requiring a minimum E-value of 10^{-8} and at least two aligned reference exons over all reference species to be retained. Each surviving alignment group was then converted to a sequence range, extended by 10 bp at each end to account for truncated alignments failing to cover the ends of the exon, and used to extract the corresponding exon sequence into FASTA format. These sequences then underwent manual curation to resolve conflicting exon identities, assign exon names and perform initial end refinement based on putative splice junctions.

In order to validate intron/exon boundaries and investigate splicing behaviour among *IGH* constant-region exons in *N. furzeri* and *X. maculatus*, published RNA-sequencing data (Table D.2) were aligned to the annotated locus using STAR [65]. In both cases, reads files from multiple individuals were concatenated and aligned together, in order to make the intron/exon boundary changes in mapping behaviour as clear as possible.

Before aligning the RNA-seq reads, each locus underwent basic repeat masking, using the built-in zebrafish repeat parameters from RepeatMasker [66]:

```
RepeatMasker -species danio -dir <masked_locus_dir> -s
   ↪ <unmasked_locus_path>
```

After masking, a STAR genome index was generated from each locus:

```
STAR --runMode genomeGenerate --genomeDir
   ↪ <star_index_directory_path> --genomeFastaFiles
   ↪ <masked_locus_path> --genomeSAindexNbases <sa_index>
```

where the `--genomeSAindexNbases` option determines the size of the suffix-array index and is dependent on the length of the reference sequence being indexed:

$$\text{SA index size (bits)} = \left\lfloor \frac{\log_2(\text{length of reference sequence})}{2} - 1 \right\rfloor \tag{2.2}$$

Following index generation, the RNA-seq reads were mapped to the generated index as follows:

```
STAR --genomeDir <star_index_directory_path> --readFilesIn
   ↪ <input_reads> --outFilterMultimapNmax 5 --alignIntronMax
   ↪ 10000 --alignMatesGapMax 10000
```

where the `--outFilterMultimapNmax` option excludes read pairs mapping to more than five distinct co-ordinates in the reference sequence, the `--alignIntronMax` option excludes reads spanning predicted introns of more than 10 kb, and the `--alignMatesGapMax` option excludes read pairs mapping more than 10 kb apart. Following alignment, the resulting SAM files were processed into sorted, indexed BAM files using SAMtools [67] and visualised with Integrated Genomics Viewer (IGV, [68, 69]) to determine intron/exon boundaries of predicted exons, as well as the major splice isoforms present in each dataset.

In order to reduce time and memory requirements for generating alignment figures (Figures 3.6 and 3.14), secondary alignments were performed on truncated loci consisting only of the IGHM/D or (where present) IGHZ constant regions, plus a few flanking kilobases on each side. In these cases, the additional parameters constraining multimapping, intron length and mate distance were not necessary due to the much shorter and less repetitive reference sequence.

For species other than *N. furzeri* or *X. maculatus*, intron/exon boundaries were predicted manually based on BLASTN and BLASTP alignments to closely-related species and the presence of conserved splice-site motifs (`AG` at the 5' end of the intron, `GT` at the 3' end [70]). In cases where no 3' splice site was expected to be present (e.g. for CM4 or TM2 exons), the nucleotide exon sequence was terminated at the first canonical polyadenylation site (`AATAAA` if present, otherwise one of `ATTAAA`, `AGTAAA` or `TATAAA` [71]), while the amino-acid sequence was terminated at the first STOP codon. In many cases, it was not possible to locate a TM2 exon due to its very short conserved coding sequence (typically only 2 to 4 amino-acid residues [48, 49].

2.3.4.5 Synteny analysis

Synteny between subloci in the *N. furzeri* locus was analysed using DECIPHER's standard synteny pipeline [72], which searches for chains of exact *k*-mer matches within two sequences:

```
DBPath <- tempfile()
DBConn <- dbConnect(SQLite(), DBPath)
```

```
Seqs2DB(seqs = <sublocus_1_sequence>, type = "XStringSet",
   ↪ dbFile = DBConn, identifier = "IGH1", verbose = FALSE)
Seqs2DB(seqs = <sublocus_2_sequence>, type = "XStringSet",
   ↪ dbFile = DBConn, identifier = "IGH2", verbose = FALSE)

dbDisconnect(DBConn)

SyntenyObject <- FindSynteny(dbFile = DBPath, verbose = FALSE)
```

Cross-locus sequence comparisons between gene segments were performed analogously to the comparisons involved in VH family assignment, with `pairwiseAlignment` and `pid` from Biostrings.

2.3.5 Phylogenetic trees

2.3.5.1 Species tree construction and annotation

Information about the interrelationships of most of the teleost taxa discussed in this thesis was obtained from the comprehensive teleost phylogeny of Hughes *et al.* [73], while additional, higher-resolution information on the interrelationships of African killifishes missing from that tree was provided by Cui *et al.* [74]. As no single published tree covered all the species of interest, a simple cladogram of relatioships was constructed manually from the information provided by these two sources. Annotations (e.g. of clade membership or isotype status) were added using tidytree [35].

2.3.5.2 Phylogenetic inference on *IGH* locus sequences

Three phylogenetic trees were inferred from molecular data of *IGHZ* gene segments: one on the VH segments on *Nothobranchius furzeri* and *Xiphophorus maculatus*, one on CH exons from all species, and one on *IGHZ* constant-regions of *IGHZ* bearing species. In all cases, a sequence FASTA database was assembled from the relevant species. As identical sequences can cause problems during phylogenetic analysis, entries with completely identical sequences were then collapsed together into a single FASTA sequence, which was relabelled with the names of all its parent sequences.

A multiple-sequence alignment of the remaining sequences was then constructed with PRANK:

```
prank -d=<ch_fasta> -o=<output_prefix> DNA -termgap
```

The resulting alignment was passed to the maximum-likelihood phylogenetic inference program `RAxML` ([75–77], version 8.2.12), using the SSE3-enabled parallelised version of the software, the standard GTR-Gamma nucleotide substitution model, and built-in rapid bootstrapping:

```
raxmlHPC-PTHREADS-SSE3 -f a -m GTRGAMMA -s <ch_prank_alignment>
   ↪ -w <output_dir> -N <n_bootstrap_replicates> -x
   ↪ <bootstrap_seed> -p <parsimony_seed> -n <output_suffix>
```

Finally, the bootstrap-annotated `RAxML_bipartitions` file was inspected and rooted manually in Figtree [78], before being annotated and visualised in R with tidytree and ggtree, respectively.

VH-segment tree

In order to build a phylogenetic tree of VH segments from the *N. furzeri* and *X. maculatus IGH* loci, all VH sequences from those loci were labelled with their origin species and combined together into a single FASTA file. Sequences with more than 25% missing characters were discarded prior to PRANK alignment. During tree-inference with RAxML, 100 bootstrap replicates were used.

CH exon tree

To build a phylogenetic tree of CH exons, nucleotide sequences of constant exons from all species involved in this study were labelled with their origin species and combined into a single FASTA file, which was then filtered to discard transmembrane exons, secretory tails, and sequences with more than 25% missing characters. In addition, CM4 nucleotide sequences were trimmed to the coding region, removing the 3'-UTR. As with the VH-segment tree described above, 100 bootstrap replicates were used during tree-inference with RAxML. As the outgroup among CH exon groups is unknown, the tree was visualised in unrooted format (Figure 3.18).

IGHZ tree

To investigate the evolution of *IGHZ*, the C_ζ1-4 exons from each *IGHZ* constant region found in any of the analysed genomes were concatenated together into a single sequence and labelled with the source species and constant region. In the event of partial constant regions missing one or more C_ζ exons, the remaining exons were concatenated together in the usual order. Following database processing and alignment, RAxML tree-inference was run using 1000 bootstrap replicates, in order to increase the reliability and precision of the support values obtained. During tree visualisation, nodes with bootstrap support of less than 65 % were collapsed into polytomies.

2.3.6 Ig-Seq data pre-processing

Unless otherwise specified, pre-processing utilities used in the following sections were provided by the pRESTO [79] and Change-O [80] suites of Ig-Seq processing tools.

2.3.6.1 Sequence uploading and annotation

Demultiplexed, adapter-trimmed MiSeq sequencing data were uploaded by the sequencing provider to Illumina BaseSpace and accessed via the Illumina utility program BaseMount. Library annotation

information (fish ID, sex, strain, age at death, death weight, etc.) were added to the read headers of each library FASTQ file:

```
ParseHeaders add -f <field_keys> -u <field_values> -s
   ↪ <input_reads_file>
```

The replicate and individual identity of each library were then added as additional annotations, by concatenating sets of other annotations together as specified by the user:

```
ParseHeaders.py merge -f <field_keys> -k INDIVIDUAL --act cat
   ↪ -s <annotated_reads_file>
ParseHeaders.py merge -f <field_keys> -k REPLICATE --act cat -s
   ↪ <annotated_reads_file>
```

Following annotations, reads from different libraries were pooled together, then split by replicate identity:

```
SplitSeq.py group -s {input} -f REPLICATE s <pooled_reads_file>
```

This pooling and re-splitting process enables all reads considered to be a single replicate to be processed together even if sequenced separately, maximising the effectiveness of UMI-based pre-processing while also allowing all replicates to be processed in parallel.

2.3.6.2 Quality control, primer masking and UMI extraction

After pooling and re-splitting, the raw read set underwent quality control, discarding any read with an average Phred score [81] of less than 20:

```
FilterSeq quality -q 20 -s <input_reads_file>
```

Following quality filtering, the reads underwent processing to identify and remove invariant primer sequences. To do this, known primer sequences were aligned to each read from a fixed starting position, and the best match on each read was identified and trimmed. Initially, the primer sequences from the third PCR step of the library prep protocol were used, trimming off primer sequences corresponding to part of the constant $C_\mu 1$ exon and the 5' invariant part of the template-switch adapter:

```
MaskPrimers score --mode cut --start 0 -s <3prime_read_file> -p
   ↪ <CM1_primer_file>;
MaskPrimers score --mode cut --start 0 -s <5prime_read_file> -p
   ↪ <TSA_primer_file>
```

Following this, the forward reads underwent a second round of masking using the 3' invariant part of the TSA sequence (CTTGGGG), and the intervening 16 bases were extracted and recorded in each read header as that read's unique molecular identifier (UMI):

```
MaskPrimers score --mode cut --barcode --start 16 --maxerror
   ↪ 0.5 -s <masked_5prime_read_file> -p
   ↪ <TSA_3prime_sequence_file>
```

As the match sequence for this second round of masking is shorter and more error-prone than the primer sequences used in the first round, an increased mismatch tolerance (`--maxerror 0.5`) was permitted, increasing the number of reads with successfully-extracted UMIs.

2.3.6.3 Barcode error handling

In order to reduce the level of barcode errors in each dataset, primer-masked IgSeq reads underwent barcode clustering, in which reads with the same replicate identity and highly similar UMI sequences were grouped together into the same molecular identifier group (MIG). To do this, 5'-reads were clustered by UMI sequence using CD-HIT-EST [82, 83] with a 90% sequence identity cutoff, with cluster identities being recorded in a new CLUSTER field in each read header [84]:

```
SplitSeq group -s <masked_reads_file> -f REPLICATE;
ClusterSets barcode -f BARCODE -k CLUSTER --cluster cd-hit-est
   ↪ --prefix B --ident 0.9 -s <split_reads_file>
```

In order to split any genuinely distinct MIGs accidentally united by this process, as well as to reduce the level of "natural" barcode collisions, the reads then underwent a second round of clustering, this time separately on the read sequences within each barcode cluster using VSEARCH (an open-source alternative to USEARCH [85, 86]). This time, the cluster dendrogram was cut at 75% total sequence identity, and each subcluster was separated into its own distinct MIG [84]:

```
ClusterSets set -f CLUSTER -k CLUSTER --cluster vsearch
   ↪ --prefix S --ident 0.75 -s <clustered_reads_file>
```

These clustering thresholds (90% for barcode clustering, 75% for barcode splitting) were identified empirically as the values that maximise the number of reads passing downstream quality checks in turquoise-killifish data.

The cluster annotations from these two clustering steps were combined into a single annotation, uniquely identifying each MIG in each replicate. These annotations were further modified to designate the replicate identity of each read, giving each MIG a unique annotation across the entire dataset. These annotations were copied to the reverse reads, such that each read pair had a matching MIG

annotation, and reads without a mate (due to differential processing of the two reads files) were discarded:

```
ParseHeaders collapse -s <clustered_5prime_reads> -f CLUSTER
   ↪ --act cat;
ParseHeaders merge -f REPLICATE CLUSTER -k RCLUSTER --act set
   ↪ -s <annotated_5prime_reads>;
PairSeq -1 <reannotated_5prime_reads> -2 <3prime_reads> --1f
   ↪ BARCODE CLUSTER RCLUSTER --coord illumina
```

2.3.6.4 Consensus-read generation and pair merging

Following barcode clustering, the IgSeq forward reads were grouped based on cluster identity, and the reads in each cluster grouping were aligned and collapsed into a consensus read sequence:

```
BuildConsensus --bf RCLUSTER --cf <header_fields> --act
   ↪ <copy_actions> --maxerror 0.1 --maxgap 0.5 -s
   ↪ <annotated_reads_file>
```

Positions at which at least half the aligned reads in the MIG had a gap character were deleted from the consensus (`--maxgap 0.5`), while MIGs with a mismatch rate from the consensus of more than 10% were discarded from the dataset (`--maxerror 0.1`). The resulting FASTQ file contained a single consensus sequence for each cluster annotation, labelled with its CONSCOUNT (the total number of reads contributing to that consensus sequence), the number of reads allocated to each barcode in the cluster, and various header fields (`<header_fields>`) propagated from the contributing reads by summing or concatenating the values from each contributing read (`<copy_actions>`). An identical consensus-read-generation step was performed on the reverse reads.

After consensus-read generation had been performed for both 5' and 3' reads, the annotations attached to each read were again unified across read pairs with matching cluster identities, and consensus reads without a mate of the same cluster identity were dropped:

```
PairSeq -1 <5prime_consensus_reads> -2 <3prime_consensus_reads>
   ↪ -coord presto
```

Following consensus-read generation and annotation unification, consensus-read pairs with matching cluster annotations were aligned and merged into a single contiguous sequence. Where possible, this was done by simply aligning the two mate sequences against each other; where this was not possible (e.g. due to the lack of a significant sequence overlap) the consensus reads were instead aligned with BLASTN [44, 45] to a reference database of VH sequences to generate a merged sequence, with `N`

characters used to separate pairs that aligned in a non-overlapping manner on the same VH segment [79]:

```
AssemblePairs sequential --coord presto --scanrev --aligner
   ↪ blastn --rc tail --1f <header_fields> -1
   ↪ <5prime_consensus_reads> -2 <3prime_consensus_reads> -r
   ↪ <vh_fasta_file>
```

In either case, annotation fields were copied to the new merged sequence from the forward consensus read, with the fields to be copied specified by `--1f <header_fields>`. Sequence pairs for which both alignment approaches failed were discarded.

2.3.6.5 Collapsing identical sequences and singleton removal

To quantify the abundance of each unique sequence present in each sample, merged consensus sequences with identical insert sequences but distinct MIG assignments were collapsed together into a single FASTQ entry, recording the number, size and UMI makeup of contributing MIGs in each case in the sequence header alongside any existing annotation information:

```
CollapseSeq --inner --cf <header_fields> --act <copy_actions>
   ↪ -n 20 -s <merged_consensus_seqs>
```

The collapsed sequences from each replicate identity in the dataset were then concatenated into a single file for easier downstream processing.

As sequences represented by only a single read across all MIGs in the dataset (so-called *singleton* sequences, with a `CONSCOUNT` of no more than 1) could not be corrected by UMI clustering or consensus building, they are not considered reliable for downstream processing and analysis [84]; as a result, they were here identified and separated from the other collapsed sequences:

```
SplitSeq group -f CONSCOUNT --num 2 -s
   ↪ <collapsed_consensus_seqs>
```

Finally, the non-singleton sequences so identified were converted into FASTA format with seqtk [87] for downstream processing:

```
seqtk seq -a <non_singleton_consensus_seqs> >
   ↪ <presto_fasta_output>
```

2.3.6.6 Assigning VDJ identities with IgBLAST

To assign VH, DH and JH identities to the corrected, collapsed consensus sequences produced by the pRESTO pipeline, gene segment databases were aligned to the FASTA output from Section 2.3.6.5 with IgBLAST [59]. To do this, each reference file was converted into a BLAST database with `makeblastdb`, and the output FASTA file was aligned to these databases with `igblastn`:

```
makeblastdb -parse_seqids -dbtype nucl -in <vh_reference_fasta>
   ↪ -out <vh_db_prefix>;
makeblastdb -parse_seqids -dbtype nucl -in <dh_reference_fasta>
   ↪ -out <dh_db_prefix>;
makeblastdb -parse_seqids -dbtype nucl -in <jh_reference_fasta>
   ↪ -out <jh_db_prefix>;
igblastn -ig_seqtype Ig -domain_system imgt -query
   ↪ <presto_fasta_output> -out <igblast_output>
   ↪ -germline_db_V <vh_db_prefix> -germline_db_D
   ↪ <dh_db_prefix> -germline_db_J <jh_db_prefix>
   ↪ -auxiliary_data <jh_aux_file> -outfmt '7 std qseq sseq
   ↪ btop'
```

A JH auxiliary file (Section 2.3.4.4) was used to indicate the reading frame and CDR3 boundary co-ordinate of each JH sequence in the reference database (`-auxiliary_data <jh_aux_file>`).

2.3.6.7 Clonotype inference with Change-O

Following V/D/J identity assignment, the output FASTA file from Section 2.3.6.5, raw reference segment databases from Section 2.3.4.4 and segment assignments from Section 2.3.6.6 were used to construct a tab-delimited Change-O sequence database:

```
MakeDb igblast --regions --scores --failed --partial
   ↪ --asis-calls --cdr3 -i <igblast_output> -s
   ↪ <presto_fasta_output> -r <vh_reference_fasta>
   ↪ <dh_reference_fasta> <jh_reference_fasta>
```

where `--failed` indicates that invalid sequences should be included in a separate database rather than discarded outright, `--regions` and `--cdr3` indicate that the database should include comprehensive FR and CDR annotations, `--scores` indicates that the database should include alignment score metrics, `--partial` indicates that sequences with incomplete V/D/J alignments (e.g. those without an unambiguous V- or J-assignment) should not automaticall qualify as failed, and `--asis-calls` instructs the program to accept assignments to V/D/J databases with non-standard name formatting. Following database construction, each entry was given a unique name on the basis of its replicate

identity and ordering, and the database was filtered to exclude sequences with a V-alignment score of less than 100 (Section 4.4.1).

In order to compute the appropriate distance threshold for clonotype assignment, each sequence was assigned a nearest-neighbour Hamming distance within the repertoire, using the related R packages SHazaM and Alakazam [80]:

```
tab <- readChangeoDb(<named_db_path>) %>% mutate(ROW = seq(n()))
dist_pass <- distToNearest(tab, model = "ham", normalize =
   ↪ "len", fields = "INDIVIDUAL", first = FALSE)
dist_fail <- tab %>% filter(! ROW %in% dist_pass$ROW)
   ↪ %>% mutate (DIST_NEAREST = NA)
writeChangeoDb(bind_rows(dist_pass, dist_fail),
   ↪ <db_output_path>)
```

where `model = "ham"` indicates that a single-nucleotide Hamming distance metric is to be used, `normalize = "len"` that distances should be normalised by total sequence length, `first = FALSE` determines how to handle ambiguous V/J calls, and `fields = "INDIVIDUAL"` that only distances between sequences from the same source individual should be considered.

Following assignment of nearest-neighbour distances, a distance threshold for clonotyping was computed by fitting a pair of unimodal distributions to the nearest-neighbour distribution over all sequences and selecting the threshold that minimises the maximises the average of sensitivity and specificity when assigning a point to one of these distributions (Section 4.4.2) [88]. As with nearest-neighbour distance assignment, this was done in R using SHazaM and Alakazam; all four possible models (fitting either a normal or gamma distribution to each of the two peaks) were tried, and the one with the highest maximum likelihood was used to compute the threshold value:

```
tab <- readChangeoDb(<changeo_db_path_with_distances>)
models <- c("gamma-gamma", "gamma-norm", "norm-gamma",
   ↪ "norm-norm")
thresholds <- numeric(length(models))
likelihoods <- numeric(length(models))
for(n in 1:length(models)){
  obj <- tryCatch(findThreshold(as.numeric(tab$DIST_NEAREST),
   ↪ method = "gmm", model = "hmm", cutoff = "opt"), error =
   ↪ function(e) return(e$message), warning = function(w)
   ↪ return(w$message))
  thresholds[n] <- ifelse(isS4(obj), obj@threshold, NA)
  likelihoods[n] <- ifelse(isS4(obj), obj@loglk, NA)
}
```

```
if (!all(is.na(thresholds)))
    ↪ write(thresholds[last(which(likelihoods ==
    ↪ max(likelihoods, na.rm = TRUE))], <threshold_output_path>)
```

Following inference of the correct distance threshold, clonotype inference was performed on the sequence database by grouping sequences by V- and J-assignment and CDR3 length, computing pairwise Hamming distances between each pair of CDR3 sequences in each group, and performing single-linkage clustering on the resulting distance matrix [89], with a maximum of one permitted ambiguous `N` character in the CDR3 sequence:

```
DefineClones --act set --model ham --sym min --norm len
    ↪ --failed -d <changeo_db> --dist
    ↪ <cluster_distance_threshold> --gf INDIVIDUAL --link
    ↪ single --maxmiss 1
```

where `--act set` tells the program how to handle ambiguous V/D/J assignments, `--model ham` specifies the clustering metric as the pairwise Hamming distance; `--norm len` indicates that the Hamming distances should be normalised by sequence length; `--sym min` specifies that, in the event of asymmetric `A -> B` and `B -> A` distances (e.g. arising from length normalisation) the minimum distance should be used; `--dist` specifies the distance threshold at which to cut the clustering dendrogram; and `--gf INDIVIDUAL` states that only sequences from the same individual can belong to the same clone.

Finally, the clonotype numbers assigned to each individual were combined with the group ID of each clonotype to give a unique ID for each clonotype in the dataset.

2.3.6.8 Germline inference and segment annotation

After threshold determination and clonotyping, a so-called "full-length germline sequence" is constructed for each sequence. To do this for a given sequence, germline V/J sequences are trimmed of deleted positions and concatenated together, separated by a masked region with length corresponding to the inserted nucleotides and remaining DH sequence:

```
CreateGermlines -g dmask --cloned -d <clonotyped_changeo_db> -r
    ↪ <vh_reference_fasta> <dh_reference_fasta>
    ↪ <jh_reference_fasta> --failed
```

where `-g dmask` indicates that DH nucleotides should be masked in the germline sequence, and `--failed` indicates that sequences that fail germline assignment should be retained in a separate database. Importantly, `--cloned` indicates that sequences from the same clone should recieve the same germline assignment, based on a simple majority rule among sequences in the clone; this process

also enables assignment of unambiguous segment identities to ambiguously-assigned sequences within larger clones, improving segment calls in the dataset.

Following germline inference, each sequence in the dataset was annotated according to whether or not it possessed V/D/J assignments and whether these assignments were ambiguous (i.e. whether multiple possible assignments were given rather than just one), and combined VJ and VDJ assignments were obtained by concatenating the individual segment assignments as appropriate. The processed sequence databases were then passed on to downstream analysis pipelines as outlined in Section 2.3.7.

2.3.6.9 Tracking read survival

Read survival during the pre-processing pipeline was tracked for each replicate at each stage from importation of raw reads to clonotyping (no sequences were lost during germline inference). During the pRESTO pipeline, read counting was performed by extracting sequence headers into a tab-delimited table

```
ParseHeaders.py table -s <input_seqs> -f INDIVIDUAL REPLICATE
   ↪ <other_fields>
```

with other fields (e.g. CONSCOUNT) as appropriate to the stage in the pipeline. Change-O databases, meanwhile, were already in tab-delimited tabular format. Following conversion to tabular format where necessary, read survival for each replicate was assessed by aggregating the CONSCOUNT fields of each sequence assigned to each replicate; prior to consensus-building, each sequence was assigned a CONSCOUNT of 1, making this equivalent to simply counting the number of sequences.

2.3.7 Downstream analysis of antibody repertoires

2.3.7.1 Clonal counts and inter-replicate correlations

Following the completion of the pipeline from Section 2.3.6, the number and proportion of sequences successfully assigned clonotypes was evaluated by counting the number of unique sequences in the final Change-O databases with missing (NA) clonal identities. The size of each clone in the dataset was found by counting the number of unique sequences assigned to that clone, either in total or for each replicate separately; a clone was designated to be present in a replicate if at least one sequence in that replicate was assigned to that clone. Following inference of clone sizes in each replicate, the correlation between replicates was computed using a simple Pearson's product-moment correlation coefficient (implemented in `cor` in R) comparing their respective vectors of clone sizes.

2.3.7.2 Zipf approximation of rank:frequency distributions

To obtain the rank:frequency distributions of clones in a Change-O database, the size of each clone in each repertoire (Section 2.3.7.1) was divided by the total number of unique sequences in that repertoire to obtain a relative frequency for each clone; these frequencies were then ranked in descending order within each repertoire. The resulting distributions could then be plotted on a log:log plot to visualise the underlying distribution. Best-fit Zipf distributions could then be obtained for each repertoire using maximum-likelihood estimation, either on all clones or after excluding some number of the largest clones in each repertoire:

```
lzipf <- function(f, s){
  # Compute the negative log-likelihood of a set of frequencies
  # under a Zipf distribution with parameter s
  s * sum(f * log(1:length(f))) + sum(f) * log(H(length(f), s))
}

dzipf <- function(r, s, N){
  # Return the predicted frequency of a rank r under a Zipf
  # distribution for a population with total size N
  (1/(r^s))/H(N, s)
}

compute_zipf_slope <- function(frequencies, n_exclude){
  m <- mle(function(s) lzipf(frequencies[-(1:n_exclude)], s),
   ↪ start = list(s=1))
  return(m@coef[["s"]])
}

# Add Zipf slope and predicted clonal frequencies to a
# pre-computed clone table
clone_table <- clntab %>% group_by(INDIVIDUAL) %>%
   ↪ arrange(CLONE_RANK) %>% mutate(S =
   ↪ compute_zipf_slope(CLONE_SIZE, n_exclude), EXP_FREQUENCY
   ↪ = dzipf(CLONE_RANK, S, n()), EXP_SIZE = sum(CLONE_SIZE) *
   ↪ EXP_FREQUENCY))
```

The resulting expected frequency from the fitted Zipf distribution could then be overlaid on the actual observed frequency to compare the fit of the inferred distribution to the actual clonal repertoire (Section 4.4.2). Meanwhile, the ranks and frequencies computed as part of the estimation process could be used to compute the observed P20 of the repertoire (i.e. the sum of frequencies of all clones

with rank 20 or less), as well as the expected P20 (i.e. the sum of frequences for those clones predicted by the fitted Zipf distribution).

2.3.7.3 Rarefaction analysis for inter-experiment comparison

Inter-individual comparisons of metrics such as clonal counts, P20, or the proportion of large clones in the repertoire make the implicit assumption that the sampling process (in particular, the sampling effort) used to obtain each repertoire was similar between the individuals being compared. When this is not the case – when the number of cycles used to amplify each library or the number of sequencing reads per library differs between individuals, for example – sample composition is liable to differ systematically between repertoires, making comparisons using such metrics unreliable. In the case of immune repertoire sequencing, the very large number of small clones in most immune repertoires [90] means that increased sampling depth is likely to lead to larger clonal counts, lower P20 values, and smaller proportions of large clones.

In order to compare the clonal richness and P20 values of antibody repertoires from different IgSeq experiments in the turquoise killifish in a more reliable manner, I performed rarefaction analysis [91] on these repertoires at the level of unique sequences. For each of a range of sample sizes (from 10^2 to 10^4 unique sequences), sequence entries in the pre-processed Change-O database were repeatedly subsampled without replacement from the repertoire of each individual in each experiment, for a total of 20 iterations per sample size. For each experiment, individual, and sample size, the number of small (fewer than 5 unique sequences), large (5 or more unique sequences) and total clones for each individual, as well as the P20, was computed for each subsample, and the mean and standard deviation of each measurement was computed across subsamples. The resulting rarefaction curves (of clonal count or P20 vs sample size) could then be plotted and used to compare repertoires of different experiments at any given sample size (Section 4.6).

2.3.7.4 Hill diversity spectra

The procedure for computing Hill diversity spectra (Appendix C) from a Change-O sequence database was adapted (and substantially expanded) from the code for the same purpose provided in the R package Alakazam [80, 92]. To begin with, columns in the database were specified designating how to divide the entries into an outer group (e.g. an age group), a finer inner group (e.g. an individual), and an even-finer "clone" group (by default the clonal identity, but could also be a V(D)J identity or any other set of sequence categories). Counts of unique sequences were then computed for each "clone" within each inner group (`clone_tab`), and aggregated to find total sequence counts for each inner group (`group_tab`):

```
clone_tab <- data %>% group_by_(.dots = c(outer_group,
   ↪ inner_group, clone_field)) %>% dplyr::summarize(COUNT =
   ↪ n())
group_tab <- clone_tab %>% group_by_(.dots = c(outer_group,
   ↪ inner_group)) %>% dplyr::summarize_(SEQUENCES =
   ↪ interp(~sum(x, na.rm = TRUE), x = as.name("COUNT")))
```

The size of the bootstrap replicates (`NSAM`) for each sample was then computed as the minimum across all inner groups, and a table of `nboot` bootstrap replicates of "clonal" frequencies, each of size `NSAM`, was computed for each outer-group/inner-group combination through multinomial resampling:

```
bootstrap_abundance <- function(nboot, clone_tab, group_tab,
   ↪ outer_group_name, outer_group_value, inner_group_name,
   ↪ inner_group_value, clone_field){
        # Generate independent bootstrap samples for a given
        # group combination in a clone table
        gtab <- group_tab[group_tab[[outer_group_name]] ==
                outer_group_value &
                group_tab[[inner_group_name]] ==
                inner_group_value,]
        ctab <- clone_tab[clone_tab[[inner_group_name]] ==
                inner_group_value &
                clone_tab[[outer_group_name]] ==
                outer_group_value,]
        # Get sequence number for resampling
        n <- gtab$NSAM
        # Get abundances of each clone in group
        abund_obs <- ctab$COUNT
        # Infer abundances of observed and unseen clones using
        # Chao's method (functions provided in alakazam)
        p1 <- adjustObservedAbundance(abund_obs)
        p2 <- inferUnseenAbundance(abund_obs)
        # Adjusted vector of clonal frequencies
        abund_inf <- c(p1, p2)
        # Specify clone names for known and unknown clones
        n1 <- ctab[[clone_field]]
        n2 <- paste("UNKNOWN", inner_group_value,
                seq_along(p2), sep = "_")
        names_inf <- c(n1, n2)
```

```
        # Use inferred clone distribution and resampling size to
        # generate independent bootstrap samples through
        # multinomial sampling
        sample_mat <- rmultinom(nboot, n, abund_inf)
        sample_tab <- melt(sample_mat, varnames =
                c(clone_field, "ITER"), value.name = "N")
        sample_tab[[clone_field]] <-
                names_inf[sample_tab[[clone_field]]]
        sample_tab[[outer_group_name]] <- outer_group_value
        sample_tab[[inner_group_name]] <- inner_group_value
        return(sample_tab)
}

compute_inner_bootstrap <- function(nboot, clone_tab,
   ↪ group_tab, outer_group_name, outer_group_value,
   ↪ inner_group_name, clone_field){
        # For each inner group in an outer group, compute
        # bootstrap counts and return a concatenated table
        bootstraps <- tibble()
        gtab <- group_tab[group_tab[[outer_group_name]] ==
                outer_group_value,]
        for (g in gtab %>% pull(inner_group_name) %>% unique){
                sample_tab <- bootstrap_abundance(nboot,
                        clone_tab, group_tab,
                        outer_group_name, outer_group_value,
                        inner_group_name, g, clone_field)
                bootstraps <- bind_rows(bootstraps, sample_tab)
        }
        return(bootstraps)
}

compute_outer_bootstrap <- function(nboot, clone_tab,
   ↪ group_tab, outer_group_name, inner_group_name,
   ↪ clone_field){
        # Compute inner bootstrap values for each outer group
        # and collate values in a single table
        bootstraps <- tibble()
        for (G in group_tab %>% pull(outer_group_name) %>%
```

```
                unique){
                sample_tab <- compute_inner_bootstrap(nboot,
                        clone_tab, group_tab,
                        outer_group_name, G,
                        inner_group_name, clone_field)
                bootstraps <- bind_rows(bootstraps, sample_tab)
        }
        return(bootstraps)
}

bootstraps <- compute_outer_bootstrap(nboot, clone_tab,
   ↪ group_tab, outer_group, clone_field)
```

Following bootstrap generation, individual Hill diversity numbers (Appendix C.1.3) could be computed for each inner group for each bootstrap replicate (using alakazam's built-in `calcDiversity` function [80, 92]) for each diversity order in a vector of orders `Q`:

```
bs_tab <- bootstraps %>% group_by_(.dots = c(outer_group,
   ↪ inner_group, "ITER"))
div_tab_bs_solo <- tibble()
for (q in Q){
        dt <- summarise(bs_tab, Q = q,
                D = calcDiversity(N, q = q), N_GROUP = 1)
        div_tab_bs_solo <- bind_rows(div_tab_bs_solo, dt)
}
```

Similarly, different sorts of aggregate diversity spectrum (Appendix C.2) could be computed for each outer group:

```
# Gamma diversity (simple diversity over each outer group0
bs_tab_gamma <- bootstraps %>% group_by_(.dots =
   ↪ c(group_within, clone_field, "ITER")) %>% summarise(N =
   ↪ sum(N)) %>% group_by_(.dots = c(outer_group, "ITER"))
div_tab_bs_gamma <- tibble()
for (q in Q){
        dt <- summarise(bs_tab_gamma, Q = q,
                D = calcDiversity(N, q = q))
        div_tab_bs_gamma <- bind_rows(div_tab_bs_gamma, dt)
}
```

```
# Alpha diversity (average diversity across inner groups
# within each outer groups)
bs_tab_alpha <- bootstraps %>% group_by_(.dots = c(outer_group,
   ↪ inner_group, "ITER"))
div_tab_split <- tibble()
for (q in Q){
        dt <- summarise(bs_tab_alpha, Q = q,
                D = calcDiversity(N, q = q))
        div_tab_split <- bind_rows(div_tab_split, dt)
}
div_tab_bs_alpha <- div_tab_bs_alpha %>% group_by_(.dots =
   ↪ c(outer_group, "ITER", "Q")) %>% summarise(D =
   ↪ ifelse(dplyr::first(Q) != 1,
   ↪ mean(D^(1-dplyr::first(Q)))^(1/(1-dplyr::first(Q))),
   ↪ exp(mean(log(D)))), N_GROUP = n())

# Beta diversity (gamma/alpha)
div_tab_bs_beta <- full_join(div_tab_bs_gamma,
   ↪ div_tab_bs_alpha, by = c(outer_group, "ITER", "Q"),
   ↪ suffix = c("_GAMMA", "_ALPHA")) %>% mutate(D=
   ↪ D_GAMMA/D_ALPHA) %>% select(-D_GAMMA, -D_ALPHA)
```

Finally, each diversity spectrum was grouped by diversity order and summarised across bootstrap replicates, to obtain means and standard-deviations diversity (and therefore 95 % confidence intervals) at each diversity order. Solo, alpha and gamma spectra could then be plotted (and compared between groups or against other diversity metrics) directly from these summarised tables; however, as the beta diversity of an outer group depends on the number of inner groups it contains (`N_GROUP`), beta spectra of outer groups of different sizes needed to be rescaled to a common range (0 for minimum possible beta-diversity, 1 for maximum) prior to plotting in order to be comparable (Appendix C.2.3):

```
beta_rescaled <- <summarised_beta_diversity> %>% mutate(D =
   ↪ (D-1)/(N_GROUP-1), D_UPPER = (D_UPPER-1)/(N_GROUP-1),
   ↪ D_LOWER = (D_LOWER-1)/(N_GROUP-1), D_SD =
   ↪ D_SD/(N_GROUP-1)) %>% select(-N_GROUP)
```

2.3.7.5 Repertoire Dissimilarity Index (RDI)

The Repertoire Dissimilarity Index (RDI) [93] is a pairwise distance metric between immune repertoires, based on the relative prevalence of different gene segments (or combinations of segments)

in each repertoire. To compute a set of pairwise RDIs between a group of repertoires, the number of sequences assigned to each segment-choice identity is first counted for each repertoire. These counts vectors are then downsampled without replacement to the size of the repertoire with the fewest unique sequences, then normalised to sum to some constant factor. The normalised counts are then transformed with the inverse hyperbolic sine function, which is roughly linear for values close to zero and logarithmic for values greater than one [93]; this transformation puts greater weight on rare segment-choice categories to prevent them being dominated by changes in the most prevalent categories [93]. Finally, the distance between each pair of repertoires is calculated as the Euclidean distances between their respective transformed counts vectors. This process is then repeated multiple times with independent downsamplings, and the final RDI between each pair of repertoires is given as the arithmetic mean across all iterations.

In this thesis, RDIs were computed based on the VJ-assignment of each sequence in each repertoire. To compute VJ-RDIs between repertoires in a collated Change-O database, the database was first filtered to remove sequences with missing or ambiguous V- or J-calls. The database columns denoting VJ-identity and repertoire membership (by replicate for the pilot dataset, by individual for the ageing and gut dataset) were then extracted, and used to construct a VJ-counts table in the format specified by the rdi package [93] in R:

```
# Extract VJ and ID columns
genes <- pull(<filtered_changeo_db>, <vj_call_field>)
annots <- as.character(pull(filtered_changeo_db>,
   ↪ <repertoire_id_field>))

# Create counts table
counts <- calcVDJcounts(genes = genes, seqAnnot = annots,
   ↪ simplifyNames = TRUE, splitCommas = FALSE)
```

The RDI between each pair of repertoires could then be computed as described above using the provided `calcRDI` function from the rdi package [93]:

```
rdi <- calcRDI(counts, nIter = 100)
```

where `nIter = 100` specifies that the RDI measurements should be averaged over 100 iterations.

Following RDI computation, hierarchical (average-linkage) clustering was performed on the resulting distance matrix, and the dendrogram of the resulting clustering structure was visualised using ggtree. Principal co-ordinate analysis (PCoA) was performed using standard R `pcoa` function, then processed into standard tabular format for visualisation:

```
# Perform PCoA
pc <- pcoa(rdi)
```

```
# Percentage of variation in each dimension
pc_var <- pc$values %>% pull(Broken_stick) %>% (function(x)
   ↪ round(x*100, 1))

# Extract co-ordinates in first two axes into data frame
pc_tab <- tibble(REPERTOIRE = rownames(as.matrix(rdi)), PCO1 =
   ↪ pc$vectors[,1], PCO2 = pc$vectors[,2])
```

2.3.7.6 Analysing generative repertoire diversity with IGoR

The processes generating the primary antibody repertoire include VDJ recombination, P-insertion/deletion at the ends of the selected gene segments, and N-insertion between them (Section 1.2.2). The enormous potential diversity of sequences generated by these processes makes it impossible to accurately model the generative processes by explicitly assigning probabilities to each possible sequence in the primary repertoire, as in any given sample of rearranged sequences the observed probability of the vast majority of possible sequences will be zero [94]. The program IGoR bypasses this problem by instead explicitly estimating the probability distributions of each contributing stochastic process separately, producing a generative model that can be used to estimate the entropy of the generative process, or to generate new, simulated sequences drawn from the same probability distribution [94]. In this thesis, I used IGoR to estimate these processes in turquoise killifish, in order to investigate the diversity and composition of the killifish generative repertoire and the changes that take place in the generative process with age.

Preparing sequence databases for IGoR

Before fitting a generative model with IGoR, the pre-processed Change-O database must be further processed to exclude functional sequences and (to the greatest extent possible) sequences produced by affinity maturation rather than primary sequence generation.

To begin with, the pre-processed Change-O database of all sequences in all repertoires for a given experiment is further processed with SHazaM [80] in R to collapse each clone down to a single strict-majority-rule consensus sequence:

```
# First exclude sequences with missing clonal assignments
changeo_db <- changeo_db[!is.na(changeo_db[[clone_field]]),]

# Collapse each clone to a single consensus sequence
changeo_db_collapsed <- collapseClones(changeo_db, cloneColumn
   ↪ = "CLONE", sequenceColumn = "SEQUENCE_IMGT",
   ↪ germlineColumn = "GERMLINE_IMGT_D_MASK", method =
```

```
    ↪ "mostCommon", includeAmbiguous = FALSE,
    ↪ breakTiesStochastic = FALSE, expandedDb = NULL)
```

In order to assign functional statuses to these new consensus sequences, they were then extracted into FASTA format, assigned V/J identities and CDR3 boundaries with IgBLAST, then re-imported into Change-O database format, after which functional sequences could be identified and discarded:

```
# Extract consensus DB into FASTA Format
ConvertDb fasta -d <consensus_db_path> -o
    ↪ <consensus_fasta_path> --if SEQUENCE_ID --sf SEQUENCE_OUT
    ↪ --mf INDIVIDUAL REPLICATE <other fields>

# Assign VDJ identities etc. with IgBLAST
makeblastdb -parse_seqids -dbtype nucl -in <vh_reference_fasta>
    ↪ -out <vh_db_prefix>;
makeblastdb -parse_seqids -dbtype nucl -in <dh_reference_fasta>
    ↪ -out <dh_db_prefix>;
makeblastdb -parse_seqids -dbtype nucl -in <jh_reference_fasta>
    ↪ -out <jh_db_prefix>;
igblastn -ig_seqtype Ig -domain_system imgt -query
    ↪ <consensus_fasta_path> -out <igblast_output_path>
    ↪ -germline_db_V <vh_db_prefix> -germline_db_D
    ↪ <dh_db_prefix> -germline_db_J <jh_db_prefix>
    ↪ -auxiliary_data <jh_aux_file> -outfmt '7 std qseq sseq
    ↪ btop'

# Re-import into Change-O, with new VDJ assignments and
# functional status
MakeDb igblast --regions --scores --partial --asis-calls --cdr3
    ↪ -i <igblast_output> -s <consensus_fasta_path> -r
    ↪ <vh_reference_fasta> <dh_reference_fasta>
    ↪ <jh_reference_fasta>

# Split functional from nonfunctional sequences
ParseDb split -f FUNCTIONAL -d <new_consensus_db>
```

These operations produced a pooled database of nonfunctional, semi-naïve sequences from all individuals in all sample groups. Before running IGoR on these databases, they then needed to be split by INDIVIDUAL (for individual models) or sample group annotation (for pooled models):

```
ParseDb split -d <nonfunctional_consensus_db> -f <split_field>
```

Finally, each split database was then re-extracted into FASTA format for importation by IGoR:

```
ConvertDb fasta -d <split_nonfunctional_db> -o
   ↪ <split_nonfunctional_fasta> --if SEQUENCE_ID --sf
   ↪ SEQUENCE_INPUT
```

Model inference with IGoR

Following sequence preparation, the model-inference process by IGoR took place in several steps. First, the sequences from the previous section were converted into the correct format:

```
igor -set_wd <working_directory> -read_seqs
   ↪ <split_nonfunctional_fasta>
```

These sequences were then aligned to reference V/D/J databases, using V/J "anchor" files to specify the CDR3 boundaries for each V/J identity:

```
igor -set_wd <working_directory> -set_genomic --V
   ↪ <vh_reference_fasta> --D <vh_reference_fasta> --J
   ↪ <vh_reference_fasta> -set_CDR3_anchors --V
   ↪ <v_anchor_file> --J <j_anchor_file> -align --all
```

These alignments could then be used to infer a generative model for the repertoire, by fitting probability distributions for V/D/J-choice, N-insertions, P-insertions, and deletions:

```
igor -set_wd <working_directory> -set_custom_model
   ↪ <default_model> -set_genomic --V <vh_reference_fasta> --D
   ↪ <vh_reference_fasta> --J <vh_reference_fasta>
   ↪ -set_CDR3_anchors --V <v_anchor_file> --J <j_anchor_file>
   ↪ -infer
```

The inferred model was then *evaluated*, to generate parseable parameter and probability-distribution files:

```
igor -set_wd <working_directory> -load_last_inferred -evaluate
   ↪ -output --scenarios 5 --Pgen --coverage VJ_gene
```

Finally, having inferred a generative model for each individual and pooled repertoire with IGoR, the inferred gene-choice, insertion and deletion probability distributions, along with the inferred entropy of each component of the model, were extracted from IGoR's output files with [python]pygor, a Python package supplied as part of the IGoR program.

References

1. Dodzian, J. *et al.* A Protocol for Laboratory Housing of Turquoise Killifish (Nothobranchius furzeri). *Journal of Visualized Experiments : JoVE* (2018).
2. Smith, P. *et al.* Regulation of life span by the gut microbiota in the short-lived African turquoise killifish. *eLife* **6,** e27014 (2017).
3. Carter, K. M., Woodley, C. M. & Brown, R. S. A review of tricaine methanesulfonate for anesthesia of fish. *Reviews in Fish Biology and Fisheries* **21,** 51–59 (2011).
4. Paul, N., Shum, J. & Le, T. In *RT-PCR Protocols: Second Edition* (ed King, N.) 301–318 (Humana Press, Totowa, NJ, 2010).
5. Hawkins, T. L. *et al.* DNA purification and isolation using a solid-phase. *Nucleic Acids Research* **22,** 4543–4544 (1994).
6. DeAngelis, M. M., Wang, D. G. & Hawkins, T. L. Solid-phase reversible immobilization for the isolation of PCR products. *Nucleic acids research* **23,** 4742–4743 (1995).
7. Lennon, N. J. *et al.* A scalable, fully automated process for construction of sequence-ready barcoded libraries for 454. *Genome Biology* **11,** R15 (2010).
8. Fisher, S. *et al.* A scalable, fully automated process for construction of sequence-ready human exome targeted capture libraries. *Genome Biology* **12,** R1 (2011).
9. Zumbo, P. *Phenol-chloroform extraction* Weill Cornell Medical College (2012), 7.
10. Zumbo, P. *Ethonal precipitation* Weill Cornell Medical College (2012), 7.
11. Chomczynski, P. & Sacchi, N. The single-step method of RNA isolation by acid guanidinium thiocyanate–phenol–chloroform extraction: twenty-something years on. en. *Nature Protocols* **1,** 581–585 (2006).
12. Magloire, A. *BluePippin DNA Size Selection System Operations Manual* Accessed: 2019-02-24. Sage Science Inc. (2016).
13. Birnboim, H. C. & Doly, J. A rapid alkaline extraction procedure for screening recombinant plasmid DNA. *Nucleic Acids Research* **7,** 1513–1523 (1979).
14. Turchaninova, M. A. *et al.* High-quality full-length immunoglobulin profiling with unique molecular barcoding. *Nature Protocols* **11,** 1599–1616 (2016).
15. Untergasser, A. *et al.* Primer3—new capabilities and interfaces. *Nucleic Acids Research* **40,** e115 (2012).
16. *Illumina Adapter Sequences (1000000002694)* version 9. Illumina (2018).
17. Huber, W. *et al.* Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods* **12,** 115–121 (2015).
18. Pagès, H. *et al. Biostrings: Efficient manipulation of biological strings* R package version 2.0 (2017).

19. Pagès, H. *BSgenome: Software infrastructure for efficient representation of full genomes and their SNPs* R package version 1.50.0 (2018).
20. Lawrence, M. *et al.* Software for Computing and Annotating Genomic Ranges. *PLoS Computational Biology* **9** (8 2013).
21. Becker, G. & Lawrence, M. *genbankr: Parsing GenBank files into semantically useful objects* R package version 1.10.0 (2018).
22. Winter, D. J. rentrez: an R package for the NCBI eUtils API. *The R Journal* **9,** 520–526 (2 2017).
23. Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **48,** 443–453 (1970).
24. Smith, T. F. & Waterman, M. S. Identification of common molecular subsequences. *Journal of Molecular Biology* **147,** 195–197 (1981).
25. Wickham, H., Hester, J. & Francois, R. *readr: Read Rectangular Text Data* R package version 1.2.1 (2018).
26. Wickham, H. *et al. dplyr: A Grammar of Data Manipulation* R package version 0.7.8 (2018).
27. Wickham, H. & Henry, L. *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions* R package version 0.8.2 (2018).
28. Wickham, H. *stringr: Simple, Consistent Wrappers for Common String Operations* R package version 1.3.1 (2018).
29. Köster, J. & Rahmann, S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics* **28,** 2520–2522 (2012).
30. R Core Team. *R: A Language and Environment for Statistical Computing* R Foundation for Statistical Computing (Vienna, Austria, 2018).
31. Wickham, H. *ggplot2: Elegant Graphics for Data Analysis* (Springer-Verlag New York, 2016).
32. Hahne, F. & Ivanek, R. In (eds Mathé, E. & Davis, S.) 335–351 (Springer New York, New York, NY, 2016).
33. Yu, G. *et al.* Two methods for mapping and visualizing associated data on phylogeny using ggtree. *Molecular Biology and Evolution* (2018).
34. Paradis, E. & Schliep, K. ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics* **35,** 526–528 (2019).
35. Yu, G. *tidytree: A Tidy Tool for Phylogenetic Tree Data Manipulation* R package version 0.2.0 (2018).
36. Wagih, O. *ggseqlogo: A 'ggplot2' Extension for Drawing Publication-Ready Sequence Logos* R package version 0.1 (2017).
37. Valenzano, D. R. *et al.* The African Turquoise Killifish Genome Provides Insights into Evolution and Genetic Architecture of Lifespan. *Cell* **163,** 1539–1554 (2015).
38. Reichwald, K. *et al.* Insights into Sex Chromosome Evolution and Aging from the Genome of a Short-Lived Fish. *Cell* **163,** 1527–1538 (2015).
39. Bolger, A. M., Lohse, M. & Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* **30,** 2114–2120 (2014).
40. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nature Methods* **9,** 357–359 (2012).
41. Marçais, G., Yorke, J. A. & Zimin, A. QuorUM: An Error Corrector for Illumina Reads. *PLoS ONE* **10,** e0130821 (2015).

42. Bankevich, A. *et al.* SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *Journal of Computational Biology* **19,** 455–477 (2012).
43. Nikolenko, S. I., Korobeynikov, A. I. & Alekseyev, M. A. BayesHammer: Bayesian clustering for error correction in single-cell sequencing. *BMC Genomics* **14,** S7 (2013).
44. Altschul, S. F. *et al.* Basic local alignment search tool. *Journal of Molecular Biology* **215,** 403–410 (1990).
45. Altschul, S. F. *et al.* Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* **25,** 3389–3402 (1997).
46. Boetzer, M. *et al.* Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics* **27,** 578–579 (2011).
47. Magadán-Mompó, S., Sánchez-Espinel, C. & Gambón-Deza, F. Immunoglobulin heavy chains in medaka (*Oryzias latipes*). *BMC Evolutionary Biology* **11,** 165 (2011).
48. Danilova, N. *et al.* The immunoglobulin heavy-chain locus in zebrafish: identification and expression of a previously unknown isotype, immunoglobulin Z. *Nature Immunology* **6,** 295–302 (2005).
49. Bao, Y. *et al.* The immunoglobulin gene loci in the teleost *Gasterosteus aculeatus*. *Fish & Shellfish Immunology* **28,** 40–48 (2010).
50. Gambón-Deza, F., Sánchez-Espinel, C. & Magadán-Mompó, S. Presence of an unique IgT on the IGH locus in three-spined stickleback fish (*Gasterosteus aculeatus*) and the very recent generation of a repertoire of VH genes. *Developmental & Comparative Immunology* **34,** 114–122 (2010).
51. Gertz, E. M. *et al.* Composition-based statistics and translated nucleotide searches: Improving the TBLASTN module of BLAST. *BMC Biology* **4,** 41 (2006).
52. Löytynoja, A. In *Multiple Sequence Alignment Methods* (ed Russell, D. J.) 155–170 (Humana Press, Totowa, NJ, 2014).
53. Wheeler, T. J. & Eddy, S. R. nhmmer: DNA homology search with profile HMMs. *Bioinformatics* **29,** 2487–2489 (2013).
54. Eddy, S. R. Accelerated Profile HMM Searches. *PLOS Computational Biology* **7,** e1002195 (2011).
55. Eddy, S. R. In *Genome Informatics 2009* 205–211 (PUBLISHED BY IMPERIAL COLLEGE PRESS and DISTRIBUTED BY WORLD SCIENTIFIC PUBLISHING CO., 2009).
56. Eddy, S. R. A Probabilistic Model of Local Sequence Alignment That Simplifies Statistical Significance Estimation. *PLOS Computational Biology* **4,** e1000069 (2008).
57. Ehrenmann, F. & Lefranc, M.-P. IMGT/DomainGapAlign: IMGT Standardized Analysis of Amino Acid Sequences of Variable, Constant, and Groove Domains (IG, TR, MH, IgSF, MhSF). *Cold Spring Harbor Protocols* **2011,** pdb.prot5636 (2011).
58. Lefranc, M.-P. *et al.* IMGT unique numbering for immunoglobulin and T cell receptor variable domains and Ig superfamily V-like domains. *Developmental & Comparative Immunology* **27,** 55–77 (2003).
59. Ye, J. *et al. IgBLAST: an immunoglobulin variable domain sequence analysis tool* 2013.
60. Lefranc, M.-P. Immunoglobulins: 25 Years of Immunoinformatics and IMGT-ONTOLOGY. *Biomolecules* **4,** 1102–1139 (2014).
61. Rice, P., Longden, I. & Bleasby, A. EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics* **16,** 276–277 (2000).
62. Stein, L. Generic feature format version 3. *Sequence Ontology Project,* 1–18 (2010).

63. Hesse, J. E. *et al.* V(D)J recombination: a functional definition of the joining signals. *Genes & Development* **3,** 1053–1061 (1989).
64. Et al., A. M. *GNU Grep 3.0* Accessed: 2018-11-30. Free Software Foundation, Inc (2017).
65. Dobin, A. *et al.* STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29,** 15–21 (2013).
66. Smith, A., Hubley, R. & Green, P. *RepeatMasker Open-4.0.(2013-2015)* 2016.
67. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)* **25,** 2078–2079 (2009).
68. Robinson, J. T. *et al.* Integrative genomics viewer. *Nature Biotechnology* **29,** 24–26 (2011).
69. Thorvaldsdóttir, H., Robinson, J. T. & Mesirov, J. P. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Briefings in Bioinformatics* **14,** 178–192 (2013).
70. Shapiro, M. B. & Senapathy, P. RNA splice junctions of different classes of eukaryotes: sequence statistics and functional implications in gene expression. *Nucleic acids research* **15,** 7155–7174 (1987).
71. Ulitsky, I. *et al.* Extensive alternative polyadenylation during zebrafish development. *Genome Research,* gr.139733.112 (2012).
72. Wright, E. S. Using DECIPHER v2.0 to Analyze Big Biological Sequence Data in R. *The R Journal* **8,** 352–359 (2016).
73. Hughes, L. C. *et al.* Comprehensive phylogeny of ray-finned fishes (Actinopterygii) based on transcriptomic and genomic data. *Proceedings of the National Academy of Sciences,* 201719358 (2018).
74. Cui, R. *et al.* Genome-wide relaxation of selection underlies the evolution of annual killifishes (Submitted).
75. Stamatakis, A., Ludwig, T. & Meier, H. RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics* **21,** 456–463 (2005).
76. Stamatakis, A. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics* **22,** 2688–2690 (2006).
77. Stamatakis, A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* **30,** 1312–1313 (2014).
78. Rambaut, A. *FigTree v1. 4* 2012.
79. Vander Heiden, J. A. *et al.* pRESTO: a toolkit for processing high-throughput sequencing raw reads of lymphocyte receptor repertoires. *Bioinformatics* **30,** 1930–1932 (2014).
80. Gupta, N. T. *et al.* Change-O: a toolkit for analyzing large-scale B cell immunoglobulin repertoire sequencing data. *Bioinformatics* **31,** 3356–3358 (2015).
81. Ewing, B. & Green, P. Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities. en. *Genome Research* **8,** 186–194 (1998).
82. Li, W. & Godzik, A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* **22,** 1658–1659 (2006).
83. Fu, L. *et al.* CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* **28,** 3150–3152 (2012).
84. Vander Heiden, J. A. Personal communication. 2018.
85. Edgar, R. C. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics* **26,** 2460–2461 (2010).

86. Rognes, T. *et al.* VSEARCH: a versatile open source tool for metagenomics. *PeerJ* **4,** e2584 (2016).
87. Li, H. seqtk: A fast and lightweight tool for processing sequences (Broad Inst., Cambridge, MA) (2016).
88. Nouri, N. & Kleinstein, S. H. Optimized Threshold Inference for Partitioning of Clones From High-Throughput B Cell Repertoire Sequencing Data. *Frontiers in Immunology* **9** (2018).
89. Gupta, N. T. *et al.* Hierarchical Clustering Can Identify B Cell Clones with High Confidence in Ig Repertoire Sequencing Data. en. *The Journal of Immunology* **198,** 2489–2499 (2017).
90. Mora, T. & Walczak, A. Quantifying lymphocyte receptor diversity. *arXiv:1604.00487 [q-bio].* arXiv: 1604.00487 (2016).
91. Gotelli, Nicholas J. & Colwell, Robert K. Quantifying biodiversity: procedures and pitfalls in the measurement and comparison of species richness. *Ecology Letters* **4,** 379–391 (2001).
92. Stern, J. N. H. *et al.* B cells populating the multiple sclerosis brain mature in the draining cervical lymph nodes. *Science Translational Medicine* **6,** 248ra107–248ra107 (2014).
93. Bolen, C. R. *et al.* The Repertoire Dissimilarity Index as a method to compare lymphocyte receptor repertoires. *BMC Bioinformatics* **18,** 155 (2017).
94. Marcou, Q., Mora, T. & Walczak, A. M. High-throughput immune repertoire analysis with IGoR. en. *Nature Communications* **9,** 561 (2018).
95. Robert K. Peet. The measurement of species diversity. *Annual Review of Ecology and Systematics* **5,** 285–307 (1974).
96. Berger, W. H. & Parker, F. L. Diversity of Planktonic Foraminifera in Deep-Sea Sediments. en. *Science* **168,** 1345–1347 (1970).
97. Simpson, E. H. Measurement of Diversity. en. *Nature* **163,** 688 (1949).
98. Lou Jost. Entropy and diversity. *Oikos* **113,** 363–375 (2006).
99. Caruso, T. *et al.* en. In *Biodiversity and Conservation in Europe* (eds Hawksworth, D. L. & Bull, A. T.) 35–43 (Springer Netherlands, Dordrecht, 2008).
100. Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal* **27,** 379–423 (1948).
101. Mora, T. & Walczak, A. M. Rényi entropy, abundance distribution, and the equivalence of ensembles. en. *Physical Review E* **93** (2016).
102. Hill, M. O. Diversity and Evenness: A Unifying Notation and Its Consequences. en. *Ecology* **54,** 427–432 (1973).
103. Miho, E. *et al.* Computational Strategies for Dissecting the High-Dimensional Complexity of Adaptive Immune Repertoires. English. *Frontiers in Immunology* **9** (2018).
104. Lou Jost. Partitioning diversity into independent alpha and beta components. *Ecology* **88,** 2427–2439 (2007).

Appendix A

Solutions and buffers

A.1 Enzymes

Enzyme	Concentration	Manufacturer	Product code
KAPA HiFi HotStart ReadyMix PCR Kit	2 ×[a]	Kapa Biosystems	KR0370
SMARTScribe Reverse Transcriptase	100 U µl^{-1}	Clontech Laboratories	639537
RNasin RNase inhibitor	40 U µl^{-1}	Promega	N2511
Uracil DNA glycosylase (UDG)	5 U µl^{-1}	NEB	M0280S
RNase A	100 mg ml^{-1}	QIAGEN	19101

[a] KAPA HiFi HotStart DNA Polymerase present at 0.04 U µl^{-1}.

A.2 Non-enzyme reagents and components

Reagent	Concentration	Manufacturer	Product code
SMARTScribe first-strand buffer	5 ×	Clontech Laboratories	639537[a]
Dithiothreitol (DTT)	20 mM	Clontech Laboratories	639537[a]
dNTP mix	10 µM each[b]	NEB	N0447L
1 µm Sera-Mag Magnetic SpeedBeads	50 mg ml^{-1}	GE Healthcare	65152105050250
QIAzol Lysis Reagent	1 ×	QIAGEN	79306
BluePippin electrophoresis buffer	1 ×	Sage Science	BDF1510[c]
BluePippin R2 loading solution / marker mix	1 ×	Sage Science	BDF1510[c]
Roti-Phenol/chloroform/isoamyl alcohol	1 ×	Roth	A156.2

[a] Supplied with SMARTScribe Reverse Transcriptase (Appendix A.1).

[b] i.e. 10 µM each of dATP, dGTP, dCTP and dTTP.

[c] Supplied with BluePippin 1.5 % agarose dye-free cassettes.

A.3 Prepared buffers

Name	Purpose	Composition	pH	Storage temperature
TET	Washing SeraMag beads	• 10 mM Tris base • 1 mM Na_2-EDTA • 0.05 % (v/v) Tween 20	8.0[a]	Room temperature
iSB	Preparing SeraSure bead suspension	• 4.2 M NaCl • 16.8 mM Tris base • 1.68 mM Na_2-EDTA	8.0[a]	Room temperature
EB	Buffering nucleic-acid solutions	• 10 mM Tris-HCl	8.5[a]	Room temperature
P1	Resuspending cultured *E. coli* cells	• 50 mM Tris-HCl • 10 mM Na_2-EDTA • 100 µg ml^{-1} RNase-A[b]	8[a]	4 °C[c]
P2	Cell lysis	• 200 mM Sodium hydroxide • 1 % (v/v) Sodium dodecyl sulfate	–	Room temperature
P3	Precipitation of cell lysate	• 3 M Potassium acetate	5.5[d]	Room temperature

[a] Adjust to the required pH with hydrochloric acid (HCl).

[b] Appendix A.1.

[c] Can be stored at room temperature before addition of RNase-A.

[d] Adjust to the required pH with glacial acetic acid.

Appendix B

Primers and oligonucleotides

B.1 Template-switch adapter oligos for reverse transcription

Name	Sequence	Source
SmartNNNa	AAGCAGUGGTAUCAACGCAGAGUNNNNUNNNNUNNNNUCTTrGrGrGrG	[14]

B.2 PCR and reverse-transcription primers

Name	Sequence	Purpose	Source[a]
RT1	TGGTCTTGCCAGCTGGTGATTTCCGCC	IgSeq $C_\mu 2$ reverse-transcription primer	–
M1SS	AAGCAGTGGTATCAACGCA	IgSeq PCR1 forward primer	[14]
IGH-B	CCACATGGCACCAGAGGAAAC	IgSeq PCR1 reverse primer	–
M1S+P2	GTGACTGGAGTTCAGACGTGTGCTCTTC–CGATCTCAGTGGTATCAACGCAGAG	IgSeq PCR2 forward primer	[14]
IGH-C+P1	ACACTCTTTCCCTACACGACGCTCTTC–CGATCTATGGCACCAGAGGAAACACAAC	IgSeq PCR2 reverse primer	–

[a] Items without a specified source were designed by the author using Primer3 [15].

[b] Reverse-transcription

B.3 Illumina TruSeq adaptor sequences

B.3.1 P1/i5 adaptor sequences

Base sequence: AATGATACGGCGACCACCGAGATCTACAC**NNNNNNNN**–
ACACTCTTTCCCTACACGACGC

Index sequences:

Name	Index Sequence[a]
D501	AGGCTATA
D502	GCCTCTAT
D503	AGGATAGG
D504	TCAGAGCC
D505	CTTCGCCT
D506	TAAGATTA
D507	ACGTCCTG
D508	GTCAGTAC

[a] [16]

B.3.2 P2/i7 adaptor sequences

Base sequence: ACAAGCAGAAGACGGCATACGAGAT**NNNNNNNN**–
GTGACTGGAGTTCAGACGTGTGC

Index sequences:

Name	Index Sequence[a]
D701	CGAGTAAT
D702	TCTCCGGA
D703	AATGAGCG
D704	GGAATCTC
D705	TTCTGAAT
D706	ACGAATTC
D707	AGCTTCAG
D708	GCGCATTA
D709	CATAGCCG
D710	TTCGCGGA
D711	GCGCGAGA
D712	CTATCGCT

[a] [16]

Appendix C

Hill numbers and antibody repertoire diversity

The question of how to measure the diversity of clones or sequences within an antibody repertoire, as well as the degree of divergence in composition between pairs or groups of repertoires, closely parallels related questions in ecology, information theory, and elsewhere. Over time, a great many different conceptions of diversity have been developed in these fields [95], each with its own cohort of measurement and approximation methods. Many of these diversity indices, however, can be unified into a common framework of so-called "true diversities" based on Hill numbers, providing a more intuitive and comprehensive insight into the diversity structure of a population. In this appendix, I describe the concepts and motivations underlying this conception of population of diversity, both for individual (unitary) populations (Appendix C.1) and for more complex populations with internal subpopulation structure (Appendix C.2).

C.1 Diversity in unitary populations

C.1.1 Terminology

Though in this appendix I use terminology derived from the ecological literature, the concepts and measures discussed here can be applied to any situation in which a set (or *population*) of elements (*individuals*) is partitioned among some number of mutually-exclusive categories (*species*). Considered abstractly, these terms and denotations could be used to refer to coloured balls in an urn, species in a rainforest, or sequences in a repertoire.

Let X be a unitary population of individuals, each of which is assigned to some species s from a set of possible species S. The term "unitary" here denotes that X has no internal structure except the

species identity of its constitutent individuals. Let n_s denote the number of individuals in X belonging to s, and N denote the total number of individuals in X. Then

$$p_s = \frac{n_s}{N} \tag{C.1}$$

denotes the proportion (or *relative frequency*) of individuals in X belonging to s, or equivalently the probability that a randomly-selected individual from X belongs to s. The *species richness* of P is the total number of species $|S|$, while the *evenness* of P is the degree to which different species are similar in their p_s values: a population containing one very abundant species and $n-1$ very rare species, for example, is much less even than a population containing n equally-abundant species.

C.1.2 Simple diversity indices

The diversity of a unitary population X of some total size N is generally considered to increase with both the richness and evenness of X. Different measures of within-population diversity, however, place different amounts of weight on the richness of X compared to its evenness when evaluating its diversity. At one extreme, the species richness itself is used as a measure of diversity, albeit one that entirely ignores evenness; other commonly-used diversity measures, such as Simpson's index, Shannon entropy, and the Berger-Parker index are affected to varying degrees by both richness and evenness [95, 96]. In addition to their relative emphasis on richness vs evenness, different indices will also place different amounts of weight on common vs rare species when computing diversity: at the extremes, species richness gives the same amount of weight to all species regardless of frequency, while the Berger-Parker index gives zero weight to all species except the most common (Appendix C.1.2.2). As a result, any given diversity index captures a different aspect of the diversity structure of any population of interest.

C.1.2.1 Simpson's index

One of the oldest diversity indices incorporating both species richness and evenness is Simpson's index, which measures the probability that two randomly selected individuals from a population are from the same species [97]. For a finite population (or sample) X with a set of possible species S, Simpson's index is calculated as follows:

$$L(X) = \frac{\sum_{s \in S} n_s(n_s - 1)}{N(N-1)} \tag{C.2}$$

As the population size tends to infinity, this value simplifies to

$$L^*(X) = \frac{\sum_{s\in S} n_s^2}{N^2} = \sum_{s\in S} p_s^2 \tag{C.3}$$

As originally formulated, Simpson's index can be considered a *dominance index* (also known as a concentration index [97]), measuring the degree to which a population is dominated by a small number of species; as such, a higher Simpson's index indicates a less-diverse population. Conversely, the complement $1 - L(P)$ of Simpson's index, known as the Gini-Simpson index [98], represents the "probability of interspecific encounter" [95], and is also widely used as a diversity index in the literature.

C.1.2.2 The Berger-Parker index

The Berger-Parker index is a very simple measure of diversity, given by the relative frequency p_x of the most abundant species x in the population [96, 99]. Like Simpson's index, the Berger-Parker is a dominance index, measuring the degree to which the population is dominated by the single largest species. Despite its simplicity, the Berger-Parker is a true diversity index, responding to both the richness and the evenness of a population, and can be used to distinguish different diversity structures in real populations [99]. As it focuses only on the most common species in each population, it also has the particular advantage of being much less vulnerable to sampling bias than other diversity metrics, especially compared to the species richness itself [96].

C.1.2.3 Entropic diversity indices

In information theory, the Shannon entropy $H(Y)$ of a random variable Y provides a measure of the unpredictability of that variable, and therefore of the degree to which its value can be predicted in advance [100]. A variable which can take only a single value has zero entropy (it is perfectly predictable), while one which can take all values in its state space with equal probability has maximum entropy (it is maximally unpredictable) for that state space. All else being equal, a variable which can take a larger number of possible values has greater entropy than one which can take fewer values; hence, the entropy of a variable increases with both the richness and evenness of its state space [100].

Although the concept of Shannon entropy was developed in the context of electronic communication, it can be extended quite naturally to ecology, where the process of sampling species from a population represents an information source and each species observation represents an output. In this case, the Shannon entropy represents "the amount of uncertainty that exists regarding the species of an individual selected at random from the population" [95]. The Shannon entropy of a population X is given by

$$H(X) = -\sum_{s \in S} p_s \cdot \log p_s \tag{C.4}$$

Any base logarithm can be used, though bases 2 and e are most common; in these bases, the units of Shannon entropy are the bit ("binary digit") and nat ("natural unit"), respectively.

A generalisation of the Shannon entropy, also used as a diversity measure, is the Rényi entropy [101]:

$$H_q(X) = \frac{1}{1-q} \log \sum_{s \in S} p_i^q \tag{C.5}$$

where q is denoted the *order* of the entropy measure. The Rényi entropy is undefined at $q = 1$, but reduces to the Shannon entropy (with the same base of logarithm) in the limit as $q \to 1$ [101]. As p_i is always between 0 and 1, raising the relative frequencies to a power greater than 1 downweights rarer species (with smaller p_i) relative to more common ones; as a result, higher-order Rényi entropies put more emphasis on the most common species compared to Shannon entropy when computing population diversity, while Rényi entropies with order less than 1 put more weight on rarer species; at $q = 0$, H_0 is simply the logarithm of the species richness $|S|$.

C.1.3 Effective species richness and true diversity

Appendix C.1.2 discusses several commonly-used diversity indices, including Simpson's index, the Gini-Simpson index, Shannon entropy (& other Rényi entropies), and the Berger-Parker index; many other diversity indices are possible. These indices differ importantly in their forms, numeric ranges and biological interpretation, as well as their response behaviours to different changes in population composition [95, 98]. For example, Simpson's index and the Berger-Parker index range from 0 to 1, with lower values indicating greater population diversity, while Shannon- and Rényi-entropy measures range from 0 to infinity, with higher values indicating greater diversity (in log-scale). The use of different diversity indices can therefore yield importantly different results when used to compare different populations: comparing two populations using only one such measure will capture only part of the diversity structure of those populations, while comparing them using two or more raw indices will often yield results that are difficult to accurately interpret [95, 98].

Fortunately, different diversity indices can be transformed into a common framework by considering, for each index, the number D of *equally-common* species that would be needed to produce the same diversity value using that index. This transformation gives an estimation, for each index, of the "effective species richness" of the population: if we corrected for differences in species abundance while holding the diversity index constant, how rich would the resulting population be? This effective

richness can itself be considered a diversity measure; one that is comparable across diversity indices in a way the raw indices are not [98].

For many important diversity indices, the equation for the effective richness D takes a common form, known as the Hill number or "true diversity" for some parameter q [98, 102]:

$$ {}^{q}D(X) = \left(\sum_{s \in S} p_s^q \right)^{\frac{1}{1-q}} \tag{C.6} $$

Indices whose effective richnesses take this form (Table C.1) include species richness ($q = 0$), Rényi entropy in base e ($q = q$), Shannon entropy in base e ($q \to 1$), Simpson's index ($q = 2$) and the Berger-Parker index ($q \to \infty$) among others [95, 98, 102, 103]. As with the Rényi entropy, the parameter q, known here as the *diversity order*, describes the degree to which rare species are downweighted compared to common ones when calculating the Hill number: at one extreme ($q = 0$, the species richness), all species are given equal weight regardless of their frequency, while at the other ($q \to \infty$, the reciprocal Berger-Parker index) only the most common species in the sample is considered. As a result of this downweighting, higher-order indices are also less sensitive to undersampling than lower-order ones; this is particularly important to keep in mind in cases, like immune repertoires, where the number of rare species is extremely large and undersampling of rare species is pervasive [90].

The use of Hill numbers in diversity estimation enables many different, widely-used diversity measurements to be considered and compared in a common framework, giving a description of population diversity that is easy to interpret biologically and compare across different populations [98]. Typically, the limit at $q \to 1$ is substituted for the (undefined) value at $q = 1$, giving a function that is both continuous and monotonically decreasing:

$$ {}^{q}D(X) = \begin{cases} \left(\sum_{s \in S} p_s^q \right)^{\frac{1}{1-q}} & q \neq 1 \\ \exp\left(-\sum_{s \in S} p_s \cdot \ln p_s \right) & q = 1 \end{cases} \tag{C.7} $$

This equation can then be easily used to construct diversity profiles (or *spectra*) spanning many different orders of diversity [103]; since each value of q captures a different aspect of a population's diversity structure, these profiles can be much more informative than any single metric when analysing and comparing the diversity structure of populations.

Table C.1: Summary of effective richness measures for some common diversity indices (adapted and expanded from [98])

Diversity index, $f(X)$	Effective species richness, ${}^qD(X)$	Diversity order, q
Species richness	${}^0D(X) = f(X)$	0
Shannon entropy (base e)	$\lim_{q\to 1}\ {}^qD(X) = \exp f(X)$	1
Simpson index	${}^2D(X) = \frac{1}{f(X)}$	2
Gini-Simpson index	${}^2D(X) = \frac{1}{1-f(X)}$	2
Renyi entropy (base e, order q)	${}^qD(X) = \exp f(X)$	q
Berger-Parker index	$\lim_{q\to\infty}\ {}^qD(X) = \frac{1}{f(X)}$	∞

C.2 Diversity in structured populations

Section C.1 discusses methods for analysing the diversity of a single, unitary population. In many cases, however, we are interested in groups of related populations, and the relative amount of variability within and between the populations in each group. In order to extend the mathematical framework of diversity measurement, and especially that of true diversities and Hill spectra, to this more complex case, some additional clarification of terminology is needed.

C.2.1 Terminology

In a *structured* population C, the individuals in C can be partitioned into some number M of disjoint unitary subpopulations $X_1, X_2, \ldots$, such that:

- Each subpopulation X has size N_X, with the total size of the whole population given by $N = \sum_{X\in C} N_X$.
- Each individual in a subpopulation X is assigned to a species s drawn from a set of possible species S_X, with the total species set for the population given by $S = \bigcup_{X\in C} S_X$.
- Each subpopulation can be assigned a relative statistical weight w_X; this could be equal for all populations ($w_X = 1/M$), proportional to each population's relative size $w_X = \frac{N_X}{N}$, or proportional to some other measure of each population's "importance" to the system.
- The relative frequency of a species s in a subpopulation X is given by $p_{X,s} = \frac{n_{X,s}}{N_X}$, where $n_{X,s}$ is the number of individuals in X belonging to s.

What is the diversity of C? There are several possible answers, depending on which features of the makeup of C are most salient:

1. The **gamma diversity** of C is the *total* diversity of the population when its subpopulations are pooled according to their weights; it represents the species diversity across the whole population, ignoring the subpopulation membership of individuals.

2. The **alpha diversity** of C is the diversity arising from differences in species identity among individuals *within* each subpopulation, and is given by a weighted *average* of the unitary diversities of those subpopulations; the appropriate weighting function depends on the order of diversity under consideration. In some sense, the alpha diversity of C can be thought of as the expected diversity of a single population drawn from C.

3. The **beta diversity** of C is the diversity arising from variability in species composition *among* the subpopulations in C; it is lowest when all subpopulations have identical species compositions and highest when they have no species in common.

The alpha and beta diversities of a population are independent; two different structured populations can have identical alpha and very different beta diversities, or vice versa, depending on the exact species compositions of their subpopulations. The alpha and beta diversity of a structured population also completely determine its gamma diversity [104]:

$$D_\gamma(C) = D_\alpha(C) \times D_\beta(C) \tag{C.8}$$

and therefore

$$D_\beta = \frac{D_\gamma}{D_\alpha} \tag{C.9}$$

This (Equation C.9) is typically the easiest way of computing the beta diversity of a given collection of populations.

In the rest of this section, I will primarily consider only subpopulations with equal sizes $N_X = \frac{N}{M}$ and equal weights $w = \frac{1}{M}$, as this is how they are used in the repertoire-diversity methods discussed in Section 2.3.7.4 and Chapter 4. Equal-sized and -weighted populations can be produced by downsampling each subpopulation to the same number of individuals (in the IgSeq case, by downsampling each repertoire to the same number of unique sequences) prior to calculating the diversity of the population.

C.2.2 Calculating alpha, beta, and gamma

The alpha, beta and gamma diversities of structured populations can be calculated analogously to the unitary true diversities discussed in Appendix C.1.3. In this framework, the alpha diversity represents the effective number of equally-abundant species present in the "average" subpopulation drawn from

C, while the gamma diversity represents the effective number of such species present in the population as a whole (ignoring subpopulation membership). Beta diversity also represents an effective number of groupings, but in this case the unit is subpopulations rather than species: given an alpha diversity value for C, the beta diversity gives the number of equally-weighted subpopulations, with no species in common, that would give rise to the gamma diversity of C.

Under this framework, the diversities of order q for a structured population C are given [104] by

$$ {}^{q}D_{\gamma}(C) = \left[\sum_{s \in S} \left(\frac{\sum_{X \in C} w_X p_{X,s}}{\sum_{X \in C} w_X} \right)^q \right]^{\frac{1}{1-q}} = \left[\frac{\sum_{s \in S} \left(\sum_{X \in C} w_X p_{X,s} \right)^q}{\left(\sum_{X \in C} w_X \right)^q} \right]^{\frac{1}{1-q}} \tag{C.10} $$

$$ {}^{q}D_{\alpha}(C) = \left[\frac{\sum_{X \in C} w_X^q \left(\sum_{s \in S_X} p_{X,s}^q \right)}{\sum_{X \in C} w_X^q} \right]^{\frac{1}{1-q}} = \left[\frac{\sum_{X \in C} \sum_{s \in S_X} (w_X p_{X,s})^q}{\sum_{X \in C} w_X^q} \right]^{\frac{1}{1-q}} \tag{C.11} $$

$$ {}^{q}D_{\beta}(C) = \frac{{}^{q}D_{\gamma}(C)}{{}^{q}D_{\alpha}(C)} = \left[\frac{\sum_{s \in S} \left(\sum_{X \in C} w_X p_{X,s} \right)^q}{\sum_{X \in C} \sum_{s \in S_X} (w_X p_{X,s})^q} \times \frac{\sum_{X \in C} w_X^q}{\left(\sum_{X \in C} w_X \right)^q} \right]^{\frac{1}{1-q}} \tag{C.12} $$

When all the subpopulations are equally-weighted (i.e. $w_X = w = \frac{1}{M}$ for all subpopulations), these formulae simplify considerably:

$$ {}^{q}D_{\gamma}(C) = \left[\frac{\sum_{s \in S} \left(\sum_{X \in C} w p_{X,s} \right)^q}{\left(\sum_{X \in C} w \right)^q} \right]^{\frac{1}{1-q}} = \left[\frac{w^q \sum_{s \in S} \left(\sum_{X \in C} p_{P,s} \right)^q}{M^q w^q} \right]^{\frac{1}{1-q}} = \left[\frac{\sum_{s \in S} \left(\sum_{X \in C} p_{X,s} \right)^q}{M^q} \right]^{\frac{1}{1-q}} \tag{C.13} $$

$$ {}^{q}D_{\alpha}(C) = \left[\frac{\sum_{X \in C} \sum_{s \in S_X} (w p_{X,s})^q}{\sum_{X \in C} w^q} \right]^{\frac{1}{1-q}} = \left[\frac{w^q \sum_{X \in C} \sum_{s \in S_X} (p_{X,s})^q}{M w^q} \right]^{\frac{1}{1-q}} = \left[\frac{\sum_{X \in C} \sum_{s \in S_X} (p_{X,s})^q}{M} \right]^{\frac{1}{1-q}} \tag{C.14} $$

$$ {}^{q}D_{\beta}(C) = \frac{{}^{q}D_{\gamma}(C)}{{}^{q}D_{\alpha}(C)} = \left[\frac{\sum_{s\in S}\left(\sum_{X\in C} p_{X,s}\right)^{q}}{\sum_{X\in C}\sum_{s\in S_X}(p_{X,s})^{q}} \times \frac{M}{M^{q}} \right]^{\frac{1}{1-q}} = \left[\frac{\sum_{s\in S}\left(\sum_{X\in C} p_{X,s}\right)^{q}}{M^{q-1}\sum_{X\in C}\sum_{s\in S_X}(p_{X,s})^{q}} \right]^{\frac{1}{1-q}} \tag{C.15} $$

These equations are valid for all values of $q \in \mathbb{R}$ except 1, providing a spectrum of alpha, beta or gamma diversity measures analogous to the diversity spectra provided for unitary populations in Appendix C.1.3. As in the unitary case, a special case needs to be made for $q = 1$ in order to make these functions continuous[1] [104]:

$$ \begin{aligned} {}^{1}D_{\gamma}(C) &= \lim_{q\to 1} {}^{q}D_{\gamma}(C) = \exp\left[-\sum_{s\in S}\left(\left[\sum_{X\in C} w p_{X,s}\right] \cdot \ln\left[\sum_{X\in C} w p_{X,s}\right] \right) \right] \\ &= \exp\left(-\sum_{s\in S} p_s \cdot \ln p_s \right) = {}^{1}D(C) \end{aligned} \tag{C.16} $$

$$ \begin{aligned} {}^{1}D_{\alpha}(C) &= \lim_{q\to 1} {}^{q}D_{\alpha}(C) = \exp\left[-\sum_{X\in C} w \sum_{s\in S_X} (p_{X,s} \cdot \ln p_{X,s}) \right] \\ &= \exp\left[\frac{1}{M}\sum_{X\in C}\left(-\sum_{s\in S_X} (p_{X,s} \cdot \ln p_{X,s}) \right) \right] = \exp\left[\frac{1}{M}\sum_{X\in C} \ln {}^{1}D(X) \right] \end{aligned} \tag{C.17} $$

$$ {}^{1}D_{\beta}(C) = \frac{{}^{1}D_{\gamma}(C)}{{}^{1}D_{\alpha}(C)} = \frac{\exp\left[-\sum_{s\in S} p_s \cdot \ln p_s\right]}{\exp\left[\frac{1}{M}\sum_{X\in C}\left(-\sum_{s\in S_X}(p_{X,s} \cdot \ln p_{X,s})\right)\right]} = \frac{\exp\left[\frac{1}{M}\sum_{X\in C} \ln {}^{1}D(X)\right]}{{}^{1}D(C)} \tag{C.18} $$

C.2.3 Rescaling beta diversity

As discussed in Appendix C.2.2, while alpha and gamma diversity are expressed in terms of an effective number of species (in an average subpopulation and the entire structured population, respectively), beta diversity is expressed in terms of an effective number of subpopulations. Since the effective number of subpopulations is determined in part by the actual number of subpopulations, this means that the beta diversity, unlike alpha and gamma diversity, is directly dependent on the number of subpopulations M. If two different structured populations contain different numbers of subpopulations,

[1] Note that, when $w = \frac{1}{M}$ and $N_X = \frac{N}{M}$ for all populations, $\sum_{X\in C} w p_{X,s} = \sum_{X\in C} \frac{1}{M}\frac{n_{X,s}}{N_X} = \frac{1}{M}\frac{M\sum_{X\in C} n_{X,s}}{N} = \frac{n_s}{N} = p_s$

it is therefore not possible to compare their beta diversity values directly; rather, the beta diversity spectra of the populations must be *rescaled* to a common range before such a comparison is performed.

The *minimum* beta diversity of a structured population obtains when all subpopulations have identical species competition (i.e. $p_{X,s} = p_s$ for all species s and subpopulations X). In this case, the beta diversity for the structured population is given by:

$$ {}^{q}D_{\beta\,\min}(C) = \left[\frac{\sum_{s\in S}\left(\sum_{X\in C} p_{X,s}\right)^{q}}{M^{q-1}\sum_{X\in C}\sum_{s\in S_X}(p_{X,s})^{q}} \right]^{\frac{1}{1-q}} = \left[\frac{\sum_{s\in S}(Mp_s)^{q}}{M^{q-1}\sum_{s\in S_X} Mp_s^{q}} \right]^{\frac{1}{1-q}} = \left[\frac{M^{q}\sum_{s\in S} p_s^{q}}{M^{q}\sum_{s\in S_X} p_s^{q}} \right]^{\frac{1}{1-q}} = 1 \quad \text{(C.19)} $$

The *maximum* beta diversity, meanwhile, obtains when there is no overlap in species between populations. In this case, for any given species, $n_{X,s}$ is equal to n_s for one subpopulation and 0 for all others, and therefore $(\sum_{X\in C} p_{X,s})^{q} = \sum_{X\in C}(p_{X,s})^{q} = p_s^{q}$ and $\sum_{X\in C}\sum_{s\in S_X}(p_{X,s})^{q} = \sum_{s\in S} p_s^{q}$. The beta diversity is therefore given by:

$$ {}^{q}D_{\beta\,\max}(C) = \left[\frac{\sum_{s\in S}\left(\sum_{X\in C} p_{X,s}\right)^{q}}{M^{q-1}\sum_{X\in C}\sum_{s\in S_X}(p_{X,s})^{q}} \right]^{\frac{1}{1-q}} = \left[\frac{\sum_{s\in S} p_s^{q}}{M^{q-1}\sum_{s\in S} p_s^{q}} \right]^{\frac{1}{1-q}} = \left[\frac{1}{M^{q-1}} \right]^{\frac{1}{1-q}} = M \qquad \text{(C.20)} $$

The same identities hold for the special case when $q = 1$:

$$ \begin{aligned} {}^{1}D_{\beta\,\min}(C) &= \frac{\exp\left[-\sum_{s\in S} p_s \cdot \ln p_s\right]}{\exp\left[\frac{1}{M}\sum_{X\in C}\left(-\sum_{s\in S_X}(p_{X,s}\cdot \ln p_{X,s})\right)\right]} = \frac{\exp\left[-\sum_{s\in S} p_s \cdot \ln p_s\right]}{\exp\left[\frac{1}{M}\sum_{X\in C}\left(-\sum_{s\in S}(p_s \cdot \ln p_s)\right)\right]} \\ &= \frac{\exp\left[-\sum_{s\in S} p_s \cdot \ln p_s\right]}{\exp\left[\frac{M}{M}\left(-\sum_{s\in S} p_s \cdot \ln p_s\right)\right]} = 1 \end{aligned} \qquad \text{(C.21)} $$

$$
\begin{aligned}
{}^{1}D_{\beta\max}(C) &= \frac{\exp\left[-\sum_{s\in S} p_s \cdot \ln p_s\right]}{\exp\left[\frac{1}{M}\sum_{X\in C}\left(-\sum_{s\in S_X}(p_{X,s}\cdot \ln p_{X,s})\right)\right]} \\
&= \exp\left[-\sum_{s\in S} p_s \cdot \ln p_s - \frac{1}{M}\sum_{X\in C}\left(-\sum_{s\in S_X}(p_{X,s}\cdot \ln p_{X,s})\right)\right] \\
&= \exp\left[H(C) + \frac{1}{M}\sum_{X\in C}\left(\sum_{s\in S_X}\frac{n_{X,s}}{N_X}\cdot \ln\frac{n_{X,s}}{N_X}\right)\right] \\
&= \exp\left[H(C) + \frac{1}{M}\sum_{X\in C}\left(\sum_{s\in S_X}\frac{Mn_{X,s}}{N}\cdot \ln\frac{Mn_{X,s}}{N}\right)\right] \\
&= \exp\left[H(C) + \sum_{X\in C}\left(\sum_{s\in S_X}\frac{n_{X,s}}{N}\cdot \ln\frac{Mn_{X,s}}{N}\right)\right] \\
&= \exp\left[H(C) + \left(\sum_{s\in S}\frac{n_s}{N}\cdot \ln\frac{Mn_s}{N}\right)\right] = \exp\left[H(C) + \left(\sum_{s\in S} p_s \cdot \ln(Mp_s)\right)\right] \\
&= \exp\left[H(C) + \left(\sum_{s\in S} p_s \cdot [\ln M + \ln p_s]\right)\right] \\
&= \exp\left[H(C) + \left(\sum_{s\in S} p_s \cdot \ln p_s\right) + \ln M \cdot \sum_{s\in S} p_s\right] = \exp\left[H(C) - H(C) + \ln M\right] \\
&= \exp\ln M = M
\end{aligned}
\tag{C.22}
$$

The beta diversity for a structured population with M subpopulations therefore ranges between 1 (identical composition) and M (maximally-divergent composition). The beta diversities of this population can thus be transformed onto a new scale from 0 to 1 as follows:

$$
{}^{q}D_{\beta\,\text{rescaled}} = \frac{{}^{q}D_{\beta} - {}^{q}D_{\beta\,\min}}{{}^{q}D_{\beta\,\max} - {}^{q}D_{\beta\,\min}} = \frac{{}^{q}D_{\beta} - 1}{M - 1} \tag{C.23}
$$

By transforming the beta diversity spectra of different structured populations onto this common scale, the inter-subpopulation variability of those populations can be meaningfully compared, even if they differ in the number of subpopulations they contain.

Appendix D

Supplementary tables

Table D.1: Software versions used in computational analyses

Program	Version
Basemount	0.15.96.2154
BLAST	2.7.1
Bowtie2	2.2.6
CD-HIT-EST	4.6.8
Change-O	0.4.5
EMBOSS (FUZZNUC)	6.6.0
FigTree	1.4.2
HMMER	3.2
IgBLAST	1.7.0
IGoR	1.3.0
IGV	2.3.68
IMGT/DomainGapAlign	4.9.2
PRANK	v.170427
pRESTO	0.5.10
Primer3	2.3.6
Python 2	2.7.14
Python 3	3.6.4
QuorUM	1.0.0
R	3.4.1/3.5.2
RAxML	8.2.12
RepeatMasker	4.0.6
SAMtools	1.9
sed	4.2.2
seqtk	1.3
Snakemake	5.3.0
SPAdes	3.6.1
SSPACE	3.0
STAR	2.5.2b
Trimmomatic	0.32
VSEARCH	2.8.0

Table D.2: RNA-sequencing datasets used for *IGH* constant-region exon refinement and isoform identification

Species	*N. furzeri*	*X. maculatus*
Tissues	Gut	Various[a]
BioProject Accession	PRJNA379208	PRJNA420092
SRA Run Accessions	SRR5344350 SRR5344343 SRR5344344 SRR5344345 SRR5344346 SRR5344347 SRR5344348 SRR5344349 SRR5344350	SRR6327069 SRR6327070 SRR6327071 SRR6327072 SRR6327073 SRR6327074 SRR6327075 SRR6327076 SRR6327077 SRR6327078 SRR6327079 SRR6327080 SRR6327081 SRR6327082 SRR6327083 SRR6327084 SRR6327085 SRR6327086 SRR6327087 SRR6327088 SRR6327089 SRR6327090 SRR6327091 SRR6327092 SRR6327093 SRR6327094
Source	[2]	Citation not given

[a] Tissues used for *X. maculatus* RNA-sequencing included brain, heart, liver, gut, skin or whole fish; see BioProject entry for details.

Table D.3: Co-ordinate table of constant-region exons in the *Nothobranchius furzeri IGH* locus.

Name	Isotype	Start	End	Length	Strand
IGH1M-1	M	130848	131144	297	+
IGH1M-2	M	131971	132312	342	+
IGH1M-3	M	132394	132705	312	+
IGH1M-4	M	132816	133288	473	+
IGH1M-TM1	M	134262	134413	152	+
IGH1M-TM2	M	138431	138819	389	+
IGH1D-1	D	139381	139689	309	+
IGH1D-2A	D	139774	140064	291	+
IGH1D-3A	D	140178	140489	312	+
IGH1D-4A	D	140572	140853	282	+
IGH1D-2B	D	145613	145909	297	+
IGH1D-3B	D	146000	146311	312	+
IGH1D-4B	D	146398	146676	279	+
IGH1D-5	D	146795	147124	330	+
IGH1D-6	D	147210	147527	318	+
IGH1D-7	D	147598	147885	288	+
IGH1D-TM1	D	148016	148164	149	+
IGH1D-TM2	D	148323	148504	182	+
IGH2D-TM2	D	187624	187803	180	-
IGH2D-TM1	D	187963	188111	149	-
IGH2D-7	D	188658	188945	288	-
IGH2D-6	D	189016	189333	318	-
IGH2D-5	D	189419	189748	330	-
IGH2D-4B	D	189867	190145	279	-
IGH2D-3B	D	190232	190543	312	-
IGH2D-2B	D	190636	190932	297	-
IGH2D-4A	D	195644	195925	282	-
IGH2D-3A	D	196008	196319	312	-
IGH2D-2A	D	196433	196723	291	-
IGH2D-1	D	196808	197116	309	-
IGH2M-TM2	M	198315	198506	192	-
IGH2M-TM1	M	199834	199985	152	-
IGH2M-4	M	200953	201425	473	-
IGH2M-3	M	201536	201847	312	-
IGH2M-2	M	201929	202270	342	-
IGH2M-1	M	203549	203845	297	-

Name	Start	End	Length	Strand	RSS Start	Heptamer	Spacer Length	Nonamer	RSS End	RSS Length	Comment
IGH1V1-01	1252	1540	289	+	1541	CACAGTG	22	ACAAAAACC	1578	38	
IGH1V1-02	3365	3656	292	+	3657	CACAGTG	22	ACAAAAACC	3694	38	
IGH1V2-01	5907	6201	295	+	6202	CACAGAA	15	ACAAAAACT	6232	31	
IGH1V1-03	13690	13964	275	+	13965	CACAGTG	22	ACAAAAACC	14002	38	
IGH1V3-01	14862	15162	301	+	15163	CACAGTG	23	ACAAAAACC	15201	39	
IGH1V2-02	17433	17730	298	+	17731	CACAATG	23	ACAAAAACC	17769	39	
IGH1V4-01p	24566	24837	272	+	24838	CGCAGTG	22	CCACAAACC	24875	38	Nonsense mutation
IGH1V1-04	37305	37596	292	+	37597	CACAGTG	22	ACAAAAACC	37634	38	
IGH1V2-03	48845	49139	295	+	49140	CACAGTG	23	TCAAAAACT	49178	39	
IGH1V1-05	49909	50197	289	+	50198	CACAGTG	22	ACAAAAACC	50235	38	
IGH1V5-01	51710	51998	289	+	51999	CACAGTG	22	ACAAAAACT	52036	38	
IGH1V2-04	56322	56616	295	+	56617	CACAGTG	23	ACAAAAACC	56655	39	
IGH1V6-01	57465	57762	298	+	57763	CACAGTG	21	ACTAAATCT	57799	37	
IGH1V1-06	59678	59966	289	+	59967	CACAGTG	22	ACAAAAACC	60004	38	
IGH1V4-02p	68017	68288	272	+	68289	TGCAGTG	22	TCACAAACC	68326	38	Nonsense mutation
IGH1V2-05	69787	70084	298	+	70085	CACAGTG	23	ACAAAAACC	70123	39	
IGH1V1-07	155485	155763	279	+	155764	CACAGTG	22	TCAAAACCC	155801	38	
IGH2V2-02	282620	282914	295	-	282915	CACAGTG	23	ACAAAAACC	282953	39	
IGH2V4-01p	284404	284675	272	-	284676	TGCAGTG	22	TCACAAACC	284713	38	Nonsense mutation
IGH2V5-01	288808	289096	289	-	289097	CACAGTG	22	ACAGAAACT	289134	38	
IGH2V1-03	289977	290271	295	-	290272	CACAGTG	22	ACAAAAACC	290309	38	
IGH2V1-02	293835	294126	292	-	294127	CACAGTG	22	ACAAAAACC	294164	38	
IGH2V2-01	303780	304074	295	-	304075	CAGGGCC	24	AGCACAAAG	304114	40	
IGH2V1-01	304926	305204	279	-	305205	CACAGTG	22	TCAAAACCC	305242	38	

Table D.4: Co-ordinate table of VH segments in the *Nothobranchius furzeri IGH* locus.

Table D.5: Co-ordinate table of DH segments in the *Nothobranchius furzeri IGH* locus.

Name	Start	NT Sequence	End	Length	Strand
IGH1D01	25782	ATACGTACTTTCGTGGTATATAGAGA	25807	26	+
IGH1D02	76700	GATATCTGGGTGGGGG	76715	16	+
IGH1D03	77027	TGAAATGATTAC	77038	12	+
IGH1D04	77476	TCGCGTAGCGGC	77487	12	+
IGH1D05	78717	GAAACCACGGCAGC	78730	14	+
IGH1D06	79049	TTTATAGCGGCTAC	79062	14	+
IGH1D07	80417	CAGACTGGAGA	80427	11	+
IGH1D08	81362	TTCATGGCAGCCAC	81375	14	+
IGH1D09	82067	CAGACTGGAGC	82077	11	+
IGH1D10	84282	TGGGGTGGCAGC	84293	12	+
IGH2D04	263497	CAGACTGGAGA	263507	11	-
IGH2D03	270243	TTTATAGCGGCTAC	270256	14	-
IGH2D02	270878	GAAACCACGGCAGC	270891	14	-
IGH2D01	271749	GACTTTTACTAC	271760	12	-

Table D.6: Co-ordinate table of DH 5'-RSSs in the *Nothobranchius furzeri IGH* locus.

Name	5'-RSS Start	Nonamer	Spacer Length	Heptamer	5'-RSS End	Length
IGH1D01	25754	GGTTGTTGT	12	CACTGTG	25781	28
IGH1D02	76672	AGTTTTTGA	12	CACAGTG	76699	28
IGH1D03	76999	TGTTGTTGT	12	CACAGTG	77026	28
IGH1D04	77448	AGTTTTTGT	12	CACGGTG	77475	28
IGH1D05	78688	GATGTTTTT	13	CACAGTG	78716	29
IGH1D06	79021	TGTTTTTGT	12	CGCTGTG	79048	28
IGH1D07	80389	AGTTTTGGT	12	CACAGTG	80416	28
IGH1D08	81334	TGTTTTTGT	12	CGCTGTG	81361	28
IGH1D09	82039	AGTTTTGGT	12	CACAGTG	82066	28
IGH1D10	84254	TCATTCATT	12	CACTGTG	84281	28
IGH2D04	263469	AGTTTTGGT	12	CACAGTG	263496	28
IGH2D03	270215	TGTTTTTGT	12	CGCTGTG	270242	28
IGH2D02	270850	TGTTTTTGT	12	CACAGTG	270877	28
IGH2D01	271721	AGTTTTTAT	12	CATGGTG	271748	28

Table D.7: Co-ordinate table of DH 3'-RSSs in the *Nothobranchius furzeri IGH* locus.

Name	3'-RSS Start	Heptamer	Spacer Length	Nonamer	3'-RSS End	Length
IGH1D01	25808	CACAGTG	12	ACAAAAACC	25835	28
IGH1D02	76716	CACAGTG	12	ACAAAAACC	76743	28
IGH1D03	77039	CACTGTG	11	AATATAACC	77065	27
IGH1D04	77488	CACAGCG	12	ACATAAAAC	77515	28
IGH1D05	78731	CACAGCG	12	ACAAAAGCC	78758	28
IGH1D06	79063	CACTGTG	12	ACAAGATCC	79090	28
IGH1D07	80428	CACAACG	12	ACAAAAACC	80455	28
IGH1D08	81376	CACTGTG	12	ACAAAATCC	81403	28
IGH1D09	82078	CACAATG	12	ACAAAAACC	82105	28
IGH1D10	84294	CACAGTG	12	ACAAAAACC	84321	28
IGH2D04	263508	CACAACG	12	ACAAAAACC	263535	28
IGH2D03	270257	CACTGTG	12	ACAAGATCC	270284	28
IGH2D02	270892	CACAGCG	12	ACAAAAGCC	270919	28
IGH2D01	271761	CACAATG	12	ACAAAAACC	271788	28

Name	Start	NT Sequence	AA Sequence	End	Length	Strand
IGH1J01	26187	GTGCTTTAGACAACTGGGGAAAAGGAACGGAGGTTACTGTTCAACCTG	ALDNWGKGTEVTVQP	26234	48	+
IGH1J02	128176	ATGACTACTTTGACTACTGGGGAAAAGGAACAATGGTGACGGTCACATCAG	DYFDYWGKGTMVTVTS	128226	51	+
IGH1J03	128354	ACCGTGGGGTAAAGGGACAACAGTCACGGTCAAAACAG	PWGKGTTVTVKT	128391	38	+
IGH1J04	128533	ACGGTGCTCTTGACTACTGGGGTAAAGGGACCGCAGTCACTGTAACATCAG	GALDYWGKGTAVTVTS	128583	51	+
IGH1J05	128887	ACAACGCTTTTGACTACTGGGGAAAAGGAACAACGGTCACCGTCACTTCAG	NAFDYWGKGTTVTVTS	128937	51	+
IGH1J06	129346	CTACGATGCTTTTGACTACTGGGGGAAAAGGACGATGGTCACGTCACTTCAG	YDAFDYWGKRTMVTSLQ	129397	52	+
IGH1J07	129635	TTAACTGGGCTTTCGACTACTGGGGAAAAGGGACGATGGTAACGGTGACTTCAG	NWAFDYWGKGTMVTVTS	129688	54	+
IGH1J08	129965	TTACCACGCAGCTTTGGACTACTGGGGAAAAGGGACGACGGTCACCGTCACCTCAG	YHXALDYWGKGTTVTVTS	130020	56	+
IGH1J09	130612	TCTACGCTGCTTTTGACTACTGGGGTAAAGGTACAACGGTAACCGTTTCATCAG	YAAFDYWGKGTTVTVSS	130665	54	+
IGH2J08	204031	TCTACGCTGCTTTTGACTACTGGGGTAAAGGTACAACGGTAACCGTTTCATCAG	YAAFDYWGKGTTVTVSS	204084	54	-
IGH2J07	204673	TTACCACGCAGCTTTGGACTACTGGGGAAAAGGGACGACGGTCACCGTCACCTCAG	YHXALDYWGKGTTVTVTS	204728	56	-
IGH2J06	205005	ATAACTGGGCTTTCGACTACTGGGGAAAAGGGACGATGGTAACGGTGACTTCAG	NWAFDYWGKGTMVTVTS	205058	54	-
IGH2J05	205296	CTACGATGCTTTTGACTACTGGGGGAAAAGGACGATGGTCACGTCACTTCAG	YDAFDYWGKRTMVTSLQ	205347	52	-
IGH2J04	205756	ACAACGCTTTTGACTACTGGGGAAAAGGAACAACGGTCACCGTCACTTCAG	NAFDYWGKGTTVTVTS	205806	51	-
IGH2J03	206111	ATGGTGCTTTTGACTACTGGGGTAAAGGGACCGCAGTCACTGTAACATCAG	GAFDYWGKGTAVTVTS	206161	51	-
IGH2J02	206303	ACCGTGGGGTAAAGGGACAACAGTCACGGTCAAAACAG	PWGKGTTVTVKT	206340	38	-
IGH2J01	206466	ATGACTACTTTGACTACTGGGGAAAAGGAACAATGGTGACGGTCACATCAG	DYFDYWGKGTMVTVTS	206516	51	-

Table D.8: Co-ordinate table of JH segments in the *Nothobranchius furzeri IGH* locus.

Name	RSS Start	Nonamer	Spacer Length	Heptamer	RSS End	RSS Length
IGH1J01	26196	TGTTTTTGT	23	CACTGTG	26186	39
IGH1J02	128188	AGTGTTTGT	23	CACTGTG	128175	39
IGH1J03	128353	TGTTTATTT	23	CACTGTG	128353	39
IGH1J04	128545	GGTTTTTGT	23	CACTGTG	128532	39
IGH1J05	128899	GGTTTTAGT	23	TACTGTG	128886	39
IGH1J06	129360	TCTTCTTGT	22	TACTTTG	129345	38
IGH1J07	129650	AGTTTTTGT	23	TACTGTG	129634	39
IGH1J08	129983	AGTTTTAGT	22	TACTGTG	129964	38
IGH1J09	130628	CGTTTTTAT	22	CACTGTG	130611	38
IGH2J08	204047	CGTTTTTAT	22	CACTGTG	204030	38
IGH2J07	204691	AGTTTTAGT	22	TACTGTG	204672	38
IGH2J06	205020	AGTTTTTGT	23	TACTGTG	205004	39
IGH2J05	205310	TCTTCTTGT	22	TACTTTG	205295	38
IGH2J04	205768	GGTTTTAGT	23	TACTGTG	205755	39
IGH2J03	206123	GGTTTTTGT	23	CACTGTG	206110	39
IGH2J02	206302	TGTTTATTT	23	CACTGTG	206302	39
IGH2J01	206478	AGTGTTTGT	23	CACTGTG	206465	39

Table D.9: Co-ordinate table of JH RSSs in the *Nothobranchius furzeri IGH* locus.

Table D.10: Co-ordinate table of constant-region exons in the *Xiphophorus maculatus IGH* locus.

Name	Isotype	Start	End	Length	Strand
IGHZ1-1	Z	3380	3667	288	+
IGHZ1-2	Z	3814	4098	285	+
IGHZ1-3	Z	4195	4497	303	+
IGHZ1-4	Z	4934	5263	330	+
IGHZ1-S	Z	5264	5459	196	+
IGHZ1-TM1	Z	6345	6490	146	+
IGHZ1-TM2	Z	6645	7043	399	+
IGHZ2-1	Z	256059	256337	279	+
IGHZ2-2	Z	256453	256734	282	+
IGHZ2-3	Z	256893	257171	279	+
IGHZ2-4	Z	257319	257636	318	+
IGHZ2-S	Z	257637	257850	214	+
IGHZ2-TM1	Z	258059	258213	155	+
IGHZ2-TM2	Z	258410	258629	220	+
IGHM-1	M	279664	279960	297	+
IGHM-2	M	280880	281224	345	+
IGHM-3	M	281321	281629	309	+
IGHM-4	M	281789	282291	503	+
IGHM-TM1	M	282910	283034	125	+
IGHM-TM2	M	285028	285740	713	+
IGHD-1	D	285902	286219	318	+
IGHD-2A	D	286310	286597	288	+
IGHD-3A	D	286814	287128	315	+
IGHD-4A	D	287250	287534	285	+
IGHD-2B	D	288876	289166	291	+
IGHD-3B	D	289262	289576	315	+
IGHD-4B	D	289680	289964	285	+
IGHD-5	D	290052	290381	330	+
IGHD-6	D	290472	290789	318	+
IGHD-7	D	290865	291152	288	+
IGHD-TM1	D	291286	291434	149	+
IGHD-TM2	D	291541	291642	102	+

Name	Start	End	Length	Strand	RSS Start	Heptamer	Spacer Length	Nonamer	RSS End	RSS Length	Comment
IGHV01-01	1159	1450	292	+	1451	CACAGTG	23	GTAAAAACC	1489	39	
IGHV02-01	10534	10825	292	+	10826	CACAGTG	23	ACAAAACCC	10864	39	
IGHV02-02	11961	12261	301	+	12262	CACTGTG	23	ACAAAAACT	12300	39	
IGHV02-03	13319	13616	298	+	13617	CACAGTG	23	ACACAAACT	13655	39	
IGHV03-01	15440	15734	295	+	15735	CACAGTG	22	ACAAAAACT	15772	38	
IGHV02-04	16618	16908	291	+	16909	CACAGTG	23	ACAAAAACC	16947	39	
IGHV02-05	17522	17822	301	+	17823	CACTGTG	22	ACAAAAACT	17860	38	
IGHV02-06	18881	19178	298	+	19179	CACAGTG	23	ACACAAACT	19217	39	
IGHV03-02	21000	21294	295	+	21295	CACAGTG	22	ACAAAAACT	21332	38	
IGHV02-07	22179	22467	289	+	22468	CACAGTG	23	ACAAAAACC	22506	39	
IGHV02-08p	24234	24514	281	+	24515	CACAGTG	23	ACAAAAACT	24553	39	Frameshift
IGHV04-01	25359	25659	301	+	25660	CACAGTG	23	ACAAAAACT	25698	39	
IGHV04-02	27066	27366	301	+	27367	CACAGTG	23	ACAAAAACA	27405	39	
IGHV02-09	28669	28958	290	+	28959	CACAGTG	23	ACAAAAACC	28997	39	
IGHV02-10p	30460	30741	282	+	30742	CACAATG	23	ACAAAACTC	30780	39	Frameshift
IGHV02-11	32395	32681	287	+	32682	CACAGTG	23	ACAAAAACC	32720	39	
IGHV03-03	33663	33957	295	+	33958	CACTGTG	22	ACAAAAACT	33995	38	
IGHV02-12	35012	35299	288	+	35300	CACAGTG	23	ACAAAAACC	35338	39	
IGHV03-04	36281	36575	295	+	36576	CACTGTG	22	ACAAAAACT	36613	38	
IGHV02-13	37639	37931	293	+	37932	CACAGTG	23	ACAAAAACT	37970	39	
IGHV02-14	39019	39311	293	+	39312	CACAGTG	23	ACAAAAACT	39350	39	
IGHV03-05	41008	41302	295	+	41303	CACAGTG	22	ACAAAAACT	41340	38	
IGHV02-15	42660	42952	293	+	42953	CACAGTG	23	ACAAAAACT	42991	39	
IGHV03-06	45081	45375	295	+	45376	CACAGTG	22	ACAAAAACT	45413	38	
IGHV02-16	46732	47024	293	+	47025	CACAGTG	23	ACAAAAACT	47063	39	

Table D.11: Co-ordinate table of VH segments in the *Xiphophorus maculatus IGH* locus, part 1.

Name	Start	End	Length	Strand	RSS Start	Heptamer	Spacer Length	Nonamer	RSS End	RSS Length	Comment
IGHV03-07	48618	48912	295	+	48913	CACAGTG	22	ACAAAAACT	48950	38	
IGHV02-17	50323	50611	289	+	50612	CACAGTG	23	ACAAAAACC	50650	39	
IGHV03-08	51890	52184	295	+	52185	CACAGTG	22	ACAAAAACT	52222	38	
IGHV03-09p	53026	53274	249	+	53275						3'-truncated, no RSS
IGHV02-18	54462	54747	286	+	54748	CACAGTG	23	ACAAAAACC	54786	39	
IGHV02-19p	55729	55866	138	+	55867	CACAGTG	23	ACAAAAACC	55905	39	3'-truncated
IGHV03-10	57371	57662	292	+	57663	CACAGTG	22	ACAAAAACT	57700	38	
IGHV02-20p	58698	58986	289	+	58987	CACAGTG	23	ATAAAAACC	59025	39	Nonsense mutation
IGHV03-11	59940	60234	295	+	60235	CACAGTG	22	ACAAAAACT	60272	38	
IGHV02-21	61249	61537	289	+	61538	CACAGTG	23	ATAAAAACC	61576	39	
IGHV03-12	62491	62785	295	+	62786	CACAGTG	22	ACAAAAACT	62823	38	
IGHV02-22	63801	64089	289	+	64090	CACAGTG	23	ATAAAAACC	64128	39	
IGHV03-13	65043	65337	295	+	65338	CACAGTG	22	ACAAAAACT	65375	38	
IGHV02-23	66354	66640	287	+	66641	CACAGTG	23	ACAAAAACT	66679	39	
IGHV03-14	68452	68743	292	+	68744	CACTATG	22	ACAAAACTC	68781	38	
IGHV02-24	70101	70389	289	+	70390	CACAGTG	23	ACAAAAACC	70428	39	
IGHV03-15	72206	72501	296	+	72502	CACAGTG	22	ACAAAAACT	72539	38	
IGHV02-25	73484	73772	289	+	73773	CACAGTG	23	ACAAAAACC	73811	39	
IGHV03-16	75799	76090	292	+	76091	CACAGTG	22	ACAAAAACT	76128	38	
IGHV03-17	77773	78067	295	+	78068	CACAGTG	22	ACAAAAACT	78105	38	
IGHV02-26	79001	79289	289	+	79290	CACAGTG	23	ACAAAAACC	79328	39	
IGHV03-18	80492	80784	293	+	80785	CACAGTG	22	ACAAAAACT	80822	38	
IGHV02-27p	81799	82082	284	+	82083	CACAGTG	23	ACAAAAACC	82121	39	Frameshift
IGHV03-19	83736	84030	295	+	84031	CACAGTG	22	ACAAAAACT	84068	38	
IGHV02-28p	85093	85381	289	+	85382	CACAGGG	23	GCAAAAACC	85420	39	Nonsense mutation

Table D.12: Co-ordinate table of VH segments in the *Xiphophorus maculatus IGH* locus, part 2.

Name	Start	End	Length	Strand	RSS Start	Heptamer	Spacer Length	Nonamer	RSS End	RSS Length	Comment
IGHV02-29	86225	86505	281	+	86506	CACAGTG	23	ATAAAAACC	86544	39	
IGHV03-20	87419	87713	295	+	87714	CACAGTG	22	ACAAAAACT	87751	38	
IGHV03-21	94532	94826	295	+	94827	CACAGTG	23	ACAAAAACC	94865	39	
IGHV03-22	96192	96489	298	+	96490	CACAGTG	23	ACAAAAACC	96528	39	
IGHV03-23	98068	98368	301	+	98369	CACAGTG	23	ACAAAAACC	98407	39	
IGHV03-24	99482	99779	298	+	99780	CACAGTG	23	ACAAAAACC	99818	39	
IGHV03-25	101639	101936	298	+	101937	CACAGTG	23	ACAAAAACC	101975	39	
IGHV05-01p	102818	103096	279	+	103097	CAGAAGC	0	ACAAAAACT	103112	16	Frameshift
IGHV03-26	104098	104389	292	+	104390	CACAGTG	23	ACAAAATCC	104428	39	
IGHV06-01	105551	105831	281	+	105832	CACAGTG	23	ACAAAAACC	105870	39	
IGHV03-27	107274	107571	298	+	107572	CACAGTG	23	ACAAAAACC	107610	39	
IGHV03-28	108775	109072	298	+	109073	CACAGAG	23	ACAAAAACC	109111	39	
IGHV03-29	110372	110672	301	+	110673	CACAGTG	23	ACAAAAACC	110711	39	
IGHV07-01	111565	111856	292	+	111857	CACAATG	23	ACAAAAACT	111895	39	
IGHV08-01p	113033	113330	298	+	113331	CACAGAG	23	CCAAGAACC	113369	39	Nonsense mutation
IGHV09-01	115512	115800	289	+	115801	CACAGTG	22	ACAAAAACT	115838	38	
IGHV10-01	117078	117379	302	+	117380	CACAGTG	22	ACATAAACT	117417	38	
IGHV11-01	119462	119760	299	+	119761	CACAGTG	23	ACAAAAACT	119799	39	
IGHV03-30	126125	126416	292	+	126417	CACAGTG	22	ACAAAAACC	126454	38	
IGHV03-31	127109	127400	292	+	127401	CACAGTG	23	GCAAAAACC	127439	39	
IGHV12-01	128489	128786	298	+	128787	CACAGTG	23	ACAAAAACC	128825	39	
IGHV02-30	135711	136000	290	+	136001	CACAGTG	22	ACAAAAACA	136038	38	
IGHV13-01	136757	137057	301	+	137058	CACAGTG	23	ACAAAAACT	137096	39	
IGHV02-31	138344	138637	294	+	138638	CACAGTG	23	ACAAAAATC	138676	39	
IGHV02-32	140024	140315	292	+	140316	CACTGTG	23	ACAAAAACT	140354	39	

Table D.13: Co-ordinate table of VH segments in the *Xiphophorus maculatus IGH* locus, part 3.

Name	Start	End	Length	Strand	RSS Start	Heptamer	Spacer Length	Nonamer	RSS End	RSS Length	Comment
IGHV02-33	142332	142620	289	+	142621	CACAGTG	23	ACAAAAACA	142659	39	
IGHV02-34	144334	144625	292	+	144626	CACAGTG	23	ACAAAAACT	144664	39	
IGHV02-35	145740	146031	292	+	146032	CACAGTG	23	ACAAAAAAT	146070	39	
IGHV02-36	146903	147194	292	+	147195	CACAGTG	23	ACAAAAACT	147233	39	
IGHV02-37	147839	148138	300	+	148139	CACAGTG	23	ACAAAAATC	148177	39	
IGHV02-38p	150504	150797	294	+	150798	CACAATA	23	ACAAAAACC	150836	39	Nonsense mutation
IGHV02-39	152249	152537	289	+	152538	CACAGTA	23	ACAAAAACC	152576	39	
IGHV14-01	154075	154374	300	+	154375	CACAGTG	23	ACAAAAAGT	154413	39	
IGHV02-40	155433	155709	277	+	155710	CACAGTG	23	ACAAAAACC	155748	39	
IGHV02-41	156583	156870	288	+	156871	CACAGTG	23	ACAAAAACC	156909	39	
IGHV02-42	163977	164269	293	+	164270	CACAGTG	23	ACAAAACCC	164308	39	
IGHV03-32	165416	165708	293	+	165709	CACAGTG	22	ACAAAAACA	165746	38	
IGHV02-43	166994	167293	300	+	167294	CACAATG	23	ACAGAAACT	167332	39	
IGHV12-02	169602	169900	299	+	169901	CACAGTG	23	ACAAAAACC	169939	39	
IGHV02-44	171452	171752	301	+	171753	CACTGTG	23	GCAAAAACT	171791	39	
IGHV02-45	173096	173384	289	+	173385	CTCAGTG	23	ACAAAAACC	173423	39	
IGHV02-46	174714	175009	296	+	175010	CACAGTG	23	ACAAAAACT	175048	39	
IGHV02-47	176396	176697	302	+	176698	CACAGTG	23	ACAAAAACT	176736	39	
IGHV12-03	178422	178719	298	+	178720	CACAGTG	23	ACAAAAACA	178758	39	
IGHV12-04	181245	181543	299	+	181544	CACAGTG	23	ACAAAAACC	181582	39	
IGHV02-48p	182977	183236	260	+	183237	CACAGGT	8	ACAAAAACT	183260	24	5'-truncated
IGHV02-49p	184323	184611	289	+	184612	CACAGTG	23	ACAAAAACC	184650	39	Nonsense mutation
IGHV02-50	185946	186244	299	+	186245	CACAGTG	23	ACAAAAACT	186283	39	
IGHV02-51	187624	187925	302	+	187926	CACAGTG	23	ACAAAAACT	187964	39	
IGHV12-05	190987	191284	298	+	191285	CACAGTG	23	ACAAAAACA	191323	39	

Table D.14: Co-ordinate table of VH segments in the *Xiphophorus maculatus IGH* locus, part 4.

Name	Start	End	Length	Strand	RSS Start	Heptamer	Spacer Length	Nonamer	RSS End	RSS Length	Comment
IGHV02-52	192570	192868	299	+	192869	CACAGTG	19	CTGAAAACC	192903	35	
IGHV12-06	193608	193906	299	+	193907	CACAGTG	23	ACAAAAACA	193945	39	
IGHV02-53	195271	195572	302	+	195573	CACAGTG	23	ACAAAAACC	195611	39	
IGHV15-01	204396	204693	298	+	204694	CACAATC	23	ACAAAAACT	204732	39	
IGHV13-02	206203	206503	301	+	206504	CACAGTG	23	ACAAAAACT	206542	39	
IGHV16-01	207726	208020	295	+	208021	CACAGTG	22	ACAAAAACT	208058	38	
IGHV13-03	208477	208777	301	+	208778	CACAGTA	23	ACAAAAACT	208816	39	
IGHV03-33	209921	210215	295	+	210216	CACGGTG	22	ACGAAAACT	210253	38	
IGHV17-01	211322	211625	304	+	211626	CACAGTA	23	ACAAAAACC	211664	39	
IGHV15-02p	214600	214860	261	+	214861						3’-truncated, no RSS
IGHV18-01	215671	215962	292	+	215963	CACACTG	23	ACAAAAACC	216001	39	
IGHV19-01	217874	218174	301	+	218175	CACAGTG	23	ACAAAAACT	218213	39	
IGHV03-34	219368	219668	301	+	219669	CACAGTG	23	ACAAAAACA	219707	39	
IGHV20-01	220329	220632	304	+	220633	CACAGTG	23	ACAAAAATT	220671	39	
IGHV02-54p	228547	228838	292	+	228839	CACACTG	23	ACAACCCCC	228877	39	Nonsense mutation
IGHV02-55	229963	230267	305	+	230268	CACAGCG	23	ACAAAAAAA	230306	39	
IGHV03-35	231630	231928	299	+	231929	CACAGTG	23	ACAAAAACC	231967	39	
IGHV21-01p	233069	233230	162	+	233231						Nonsense mutation, 3’-truncated, no RSS
IGHV22-01p	234954	235102	149	+	235103	CACAGTG	23	TCAAAAACT	235141	39	5’-truncated
IGHV02-56	236029	236330	302	+	236331	CACAGTG	23	ACAAATACT	236369	39	
IGHV03-36p	238122	238413	292	+	238414	CACAATG	23	ACAGAATCC	238452	39	Nonsense mutation
IGHV11-02p	240281	240579	299	+	240580	CACAGTG	24	ACAAAAACT	240619	40	Nonsense mutation
IGHV09-02	241878	242166	289	+	242167	CACAGTG	22	ACAAAAACT	242204	38	
IGHV23-01	243867	244164	298	+	244165	CACAGTG	23	ACAAAATCC	244203	39	
IGHV02-57	245524	245813	290	+	245814	CACCATA	22	ACAAAATCC	245851	38	

Table D.15: Co-ordinate table of VH segments in the *Xiphophorus maculatus IGH* locus, part 5.

Table D.16: Co-ordinate table of DH segments in the *Xiphophorus maculatus IGH* locus.

Name	Start	NT Sequence	End	Length	Strand
IGHDZ01	2243	GTGGGCAGGAGGCTATGC	2260	18	+
IGHDZ02	119768	AGG	119770	3	+
IGHDZ03	128794	ACTAAAGG	128801	8	+
IGHDZ04	129907	ATCGGG	129912	6	+
IGHDZ05	158017	ATATATGGGGG	158027	11	+
IGHDZ06	197791	ATATACTGGGGTGG	197804	14	+
IGHDZ07	222022	ATGGACTGGGGGG	222034	13	+
IGHDZ08	247941	GTGATTACGGCTACGGGGC	247959	19	+
IGHDZ09	249514	TTATGGGCTGGGGAG	249528	15	+
IGHDZ10	253752	TGGGTGGGGC	253761	10	+
IGHDM01	267392	TATACAGTGGCAAC	267405	14	+
IGHDM02	268498	CAGTATAGCAAC	268509	12	+
IGHDM03	268836	TACAATGGCAAC	268847	12	+
IGHDM04	269694	TAAACAGTGGCTAC	269707	14	+

Table D.17: Co-ordinate table of DH 5'-RSSs in the *Xiphophorus maculatus IGH* locus.

Name	5'-RSS Start	Nonamer	Spacer Length	Heptamer	5'-RSS End	Length
IGHDZ01	2215	GGTTTTTGT	12	CACTGTG	2242	28
IGHDZ02	119739	TGTATTACT	13	CACAGTG	119767	29
IGHDZ03	128766	TTTACTTCT	12	CACAGTG	128793	28
IGHDZ04	129879	GGTTTTTGT	12	CACAGTG	129906	28
IGHDZ05	157989	AGTTTTTGT	12	CACAGTG	158016	28
IGHDZ06	197763	GGTTTTTGC	12	TACTGTG	197790	28
IGHDZ07	221994	GGTTTTTGT	12	CGCTGTG	222021	28
IGHDZ08	247913	TGTTTTTGT	12	ATCTGTG	247940	28
IGHDZ09	249486	AGTTTTTGT	12	TGTGGTG	249513	28
IGHDZ10	253724	AGTTTTTGT	12	TGTAGTG	253751	28
IGHDM01	267364	AGTTTTTGT	12	TACAGTG	267391	28
IGHDM02	268470	TGTTTTTGT	12	CACAGTG	268497	28
IGHDM03	268808	AGTTTTTGC	12	TACTGTG	268835	28
IGHDM04	269666	CGTTTTTGT	12	CATTGTG	269693	28

Table D.18: Co-ordinate table of DH 3'-RSSs in the *Xiphophorus maculatus IGH* locus.

Name	3'-RSS Start	Heptamer	Spacer Length	Nonamer	3'-RSS End	Length
IGHDZ01	2261	CACTAAG	12	ACAAAAAGT	2288	28
IGHDZ02	119771	CAAAATG	13	ACAAAAACT	119799	29
IGHDZ03	128802	CAGAGAA	8	ACAAAAACC	128825	24
IGHDZ04	129913	CACAATG	12	TCAAAAACC	129940	28
IGHDZ05	158028	CACAGAG	12	ACAAAAACC	158055	28
IGHDZ06	197805	CACACAG	12	ACAAAAACC	197832	28
IGHDZ07	222035	CACAGAG	12	ACAAAAACC	222062	28
IGHDZ08	247960	CACAATA	12	ACAAAAACC	247987	28
IGHDZ09	249529	CACAATG	12	ACAAAAACC	249556	28
IGHDZ10	253762	CACAGTA	12	ACAAAAACC	253789	28
IGHDM01	267406	CACAGTG	12	GCAAAAACC	267433	28
IGHDM02	268510	CACAGTG	12	ACAGAAACC	268537	28
IGHDM03	268848	CACAGTG	12	ACAAAAACC	268875	28
IGHDM04	269708	CACTGTG	12	ACAAAATCA	269735	28

Name	Start	NT Sequence	AA Sequence	End	Length	Strand
IGHJZ01	2653	ATGCCTTAGATTACTGGGGTGAAGGGACCAGAGTCACAGTGACTTCAG	ALDYWGEGTRVTVTS	2700	48	+
IGHJZ02	120639	ATTACGCTCTTGACTACTGGGGAGCAGGAACCAAAGTTACTGTAAAGCCAG	YALDYWGAGTKVTVKP	120689	51	+
IGHJZ03	130376	ACTACGGCTTTGATTACTGGGGAGACGGAACTGAAGTTACTGTTGAACCAG	YGFDYWGDGTEVTVEP	130426	51	+
IGHJZ04	158408	AGATTTAGACTACTGGGGTAATGGAACAACAGTCACGGTTCTACCAG	DLDYWGNGTTVTVLP	158454	47	+
IGHJZ05	198186	ATTATGGTTTTGACTACTGGGGAGACGGAACCACAGTCACTGTTAGTCCAG	YGFDYWGDGTTVTVSP	198236	51	+
IGHJZ06	222417	ATGCTTTTGACGTCTGGGGTAAAGGAACCACAGTTACTGTTGTACCAG	AFDVWGKGTTVTVVP	222464	48	+
IGHJZ07	254130	ATGTTTTTGACTACTGGGGTAAAGGGACTGATGTCACAGTATCTCCAG	VFDYWGKGTDVTVSP	254177	48	+
IGHJM01	276014	ACGGCTACTTCGACTACTGGGGGAAAGGAACACAAGTCACAGTGACTTCTG	GYFDYWGKGTQVTVTS	276064	51	+
IGHJM02	276284	CCACTACTTTGACTACTGGGGAAAAGGAACCACGGTTACCGTCACTTCAG	HYFDYWGKGTTVTVTS	276333	50	+
IGHJM03	276654	ACAATGCTTTTGACTACTGGGGAAAAGGAACTACGGTAACAGTAACATCAG	NAFDYWGKGTTVTVTS	276704	51	+
IGHJM04	276999	ACTACGCTTTTGACTACTGGGGAAAAGGAACAATGGTCACTGTCACTTCAG	YAFDYWGKGTMVTVTS	277049	51	+
IGHJM05	277322	ACAACTGGGCTTTTGACTACTGGGGAGCAGGAACCATGGTAACAGTAACATCAG	NWAFDYWGAGTMVTVTS	277375	54	+
IGHJM06	277672	CTACGGTGCTTTTGACTACTGGGGTAAAGGGACTACAGTCACCGTCACTTCAG	YGAFDYWGKGTTVTVTS	277724	53	+
IGHJM07	278150	CTACGATGCTTTTGACTATTGGGGGAAAGGAACAACAGTCACCGTCATCACTTCAG	YDAFDYWGKGTTVTVITS	278205	56	+
IGHJM08	278606	TTACTACTACGCTTTTGACTATTGGGGAAAAGGGACAATGGTCACCGTCACTTCAG	YYYAFDYWGKGTMVTVTS	278661	56	+

Table D.19: Co-ordinate table of JH segments in the *Xiphophorus maculatus IGH* locus.

Name	RSS Start	Nonamer	Spacer Length	Heptamer	RSS End	RSS Length
IGHJZ01	2662	TGTTTTTGT	23	CACTGTG	2652	39
IGHJZ02	120651	TGTTTTTGT	23	CACTGTG	120638	39
IGHJZ03	130388	TGTTTTTGT	23	CACCGTG	130375	39
IGHJZ04	158416	GGTTTTTGT	23	CACTGTG	158407	39
IGHJZ05	198198	GGTTTTTGT	23	CACTGTG	198185	39
IGHJZ06	222426	TGTTTTTGT	23	CACTGTG	222416	39
IGHJZ07	254139	GGTTTTTGT	23	CACTGTG	254129	39
IGHJM01	276026	TGTATTTGT	23	CACTGTG	276013	39
IGHJM02	276295	TATTTTTGC	23	CACCGTG	276283	39
IGHJM03	276666	TGTTTTTGT	23	TACTGTG	276653	39
IGHJM04	277011	TGTTTTAGT	23	TACTGTG	276998	39
IGHJM05	277338	GGTTTTTGT	22	TACTGTG	277321	38
IGHJM06	277687	GCTTTTTAT	22	CACTGTG	277671	38
IGHJM07	278168	CCTTTTTAC	22	CACTGTG	278149	38
IGHJM08	278624	GCTTTTTAA	22	CACTGTG	278605	38

Table D.20: Co-ordinate table of JH RSSs in the *Xiphophorus maculatus IGH* locus.

Species	Scaffold(s)	Region	Isotype	Known Exons [1]	Complete?	Pseudo-exons	Comments
Nothobranchius orthonotus	scf33878	IGHM1	M	1,2,3,TM1	**No**	–	CM4 missing (missing sequence)
Nothobranchius orthonotus	scf33878	IGHD1	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Nothobranchius orthonotus	scf34438	IGHM2	M	1,2,3,4,TM1	Yes	–	
Nothobranchius orthonotus	scf34438, scf33917	IGHD2	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Nothobranchius orthonotus	scf33917	IGHD3	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Nothobranchius orthonotus	scf33917	IGHD4	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Nothobranchius orthonotus	scf9255, scf26119, scf33917	IGHD5	D	3,4,2,3,4,5,6,7,TM1	**No**	–	CD1 & CD2A missing (missing sequence)
Nothobranchius orthonotus	scf27951, scf33789	IGHM3	M	1,2,3,4,TM1	Yes	–	
Nothobranchius orthonotus	scf27951, 32033	IGHD6	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Nothobranchius orthonotus	scf32137, scf21286	IGHM4	M	1,2,3,4,TM1	Yes	–	
Nothobranchius furzeri	chr6 [2]	IGH1M	M	1,2,3,4,TM1	Yes	–	
Nothobranchius furzeri	chr6 [2]	IGH1D	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Nothobranchius furzeri	chr6 [2]	IGH2M	M	1,2,3,4,TM1	Yes	–	
Nothobranchius furzeri	chr6 [2]	IGH2D	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Aphyosemion australe	scf373	IGHM	M	1,2,3,4,TM1	Yes	–	
Aphyosemion australe	scf373	IGHD	D	1,2,3,4,5,6,7,TM1	Yes	–	
Callopanchax toddi	scf107	IGHZ1	Z	1,2,3,4,TM1	Yes	–	
Callopanchax toddi	scf107	IGHZ2	Z	1,2,3,4,TM1	Yes	–	
Callopanchax toddi	scf1209	IGHZ3	Z	1,2,3,4,TM1	Yes	–	
Callopanchax toddi	scf1209	IGHM1	M	1	**No**	–	Isolated CM1 exon
Callopanchax toddi	scf945	IGHZ4	Z	1,2,3,4,TM1	Yes	–	
Callopanchax toddi	scf945	IGHM2	M	1,2,3,4,TM1	Yes	–	
Callopanchax toddi	scf945	IGHD1	D	1,2,3,4,5,6,7,TM1	Yes	1,4,5	Frameshift mutations in CD1, CD4 & CD5
Callopanchax toddi	scf265	IGHM3	M	1,2,3,4,TM1	Yes	–	
Callopanchax toddi	scf265	IGHD2	D	1,5,7,TM1	**No**	–	CD2-4 & CD5-6 missing (not in sequence)

[1] Excluding TM2 and secretory exons.

[2] Expanded *IGH* locus sequency from Section 3.2.

Table D.21: *IGH* constant regions in cyprinidontiform fish, part 1.

Species	Scaffold(s)	Region	Isotype	Known Exons [1]	Complete?	Pseudo-exons	Comments
Pachypanchax playfairii	scf547	IGHZ	Z	1,2,3,4,TM1	Yes	–	
Pachypanchax playfairii	scf125	IGHM1	M	1,2,3,4,TM1	Yes	–	
Pachypanchax playfairii	scf125	IGHD	D	1,2,3,4,5,6,7,TM1	Yes	–	
Pachypanchax playfairii	scf547	IGHM2	M	1	**No**	–	Isolated CM1 exon
Austrofundulus limnaeus	NW_013954375.1	IGHZ	Z	TM1	**No**	TM1	Isolated TM1 exon with frameshift mutation
Austrofundulus limnaeus	NW_013952673.1	IGHM	M	1,2,3,4,TM1	Yes	–	
Austrofundulus limnaeus	NW_013952673.1, NW_013956335.1	IGHD	D	1,2,3,4,5,6,7,TM1	Yes	–	
Kryptolebias marmoratus	NW_016094348.1	IGHZ1	Z	1,2,3,4,TM1	Yes	–	
Kryptolebias marmoratus	NW_016094348.1	IGHZ2	Z	1,4,TM1	**No**	–	CZ2 & CZ3 missing (not in sequence)
Kryptolebias marmoratus	NW_016094301.1	IGHM1	M	1,2,3,4,TM1	Yes	–	
Kryptolebias marmoratus	NW_016094301.1	IGHD1	D	1,2,3,4,5,6,7,TM1	Yes	–	
Kryptolebias marmoratus	NW_016094277.1	IGHM2	M	1,2,3,4,TM1	Yes	–	
Kryptolebias marmoratus	NW_016094277.1	IGHD2	D	1,2,3,4,5,6,TM1	**No**	–	CD7 missing (not in sequence)
Poecilia reticulata	NC_024338.1	IGHZ1	Z	1,2,3,4	**No**	–	TM1 missing (missing sequence)
Poecilia reticulata	NC_024338.1	IGHZ2	Z	1,2,3,4,TM1	Yes	–	
Poecilia reticulata	NC_024338.1	IGHM	M	1,2,3,4,TM1	Yes	–	
Poecilia reticulata	NC_024338.1	IGHD	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Poecilia formosa	NW_006800081.1	IGHZ1	Z	1,2,3,4,TM1	Yes	–	
Poecilia formosa	NW_006800081.1	IGHZ2	Z	1,2,3,4,TM1	Yes	–	
Poecilia formosa	NW_006800081.1	IGHZ3	Z	1,2,3,4,TM1	Yes	–	
Poecilia formosa	NW_006800081.1	IGHM	M	1,2,3,4,TM1	Yes	–	
Poecilia formosa	NW_006800081.1	IGHD	D	1,2,3,4,5,6,7,TM1	Yes	–	
Xiphophorus maculatus	NC_036458	IGHZ1	Z	1,2,3,4,TM1	Yes	–	
Xiphophorus maculatus	NC_036458	IGHZ2	Z	1,2,3,4,TM1	Yes	–	
Xiphophorus maculatus	NC_036458	IGHM	M	1,2,3,4,TM1	Yes	–	

[1] Excluding TM2 and secretory exons.

Table D.22: *IGH* constant regions in cyprinidontiform fish, part 2.

Species	Scaffold(s)	Region	Isotype	Known Exons [1]	Complete?	Pseudo-exons	Comments
Xiphophorus maculatus	NC_036458	IGHD	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Fundulus heteroclitus	NW_012234561.1	IGHZ1	Z	1,2,3,4,TM1	Yes	–	
Fundulus heteroclitus	NW_012230737.1	IGHZ2	Z	4,TM1	**No**	–	CZ1 to CZ3 missing (missing sequence)
Fundulus heteroclitus	NW_012234542.1	IGHM	M	1,2,3,4,TM1	Yes	–	
Fundulus heteroclitus	NW_012234542.1	IGHD	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Cyprinodon variegatus	NW_015154250.1, NW_015151047.1	IGHZ	Z	1,2,3,4,TM1	Yes	–	
Cyprinodon variegatus	NW_015151047.1	IGHM	M	1,2,3,4,TM1	Yes	–	
Cyprinodon variegatus	NW_015151047.1	IGHD	D	1,2,3,4,2,3,4,5,6,7,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHM1	M	1,2,3,4,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHD1	D	1,2,3,4,6,7,TM1	Yes	7	Nonsense mutation in CD7
Oryzias latipes	NC_019866.2	IGHM2	M	1,2,3,4,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHD2	D	1,2,3,4,6,7,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHM3	M	1,2,3,4,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHD3	D	1,2,3,4,6,7,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHM4	M	1,2,3,4,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHD4	D	2,7,TM1	**No**	–	CD1 & CD3-6 missing (not in sequence)
Oryzias latipes	NC_019866.2	IGHM5	M	1,2,3,4,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHD5	D	1,2,3,4,6,7,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHM6	M	1,2,3,4,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHD6	D	1,2,3,4,6,7,TM1	Yes	–	
Oryzias latipes	NC_019866.2	IGHD7	D	1,2,3,6	**No**	–	CD4, CD5, CD7 and TM1 missing (not in sequence)

[1] Excluding TM2 and secretory exons.

Table D.23: *IGH* constant regions in cyprinidontiform fish, part 3.

Table D.24: Turquoise killifish used in IgSeq validation and ageing experiment. All fish are GRZ-AD strain and male.

Group	#	Fish ID[1]	Death weight (g)	Hatch date	Sacrifice date	Age (days)	Age (weeks)
1	1	4194	1.24	2016-05-09	2016-06-17	39	5.57
1	2	4107	1.39	2016-05-09	2016-06-17	39	5.57
1	3	4127	1.29	2016-05-09	2016-06-17	39	5.57
1	4	4204	1.35	2016-05-09	2016-06-17	39	5.57
1	5	4189	1.43	2016-05-09	2016-06-17	39	5.57
1	6	4160	0.68	2016-05-09	2016-06-17	39	5.57
1	7	4164	1.57	2016-05-09	2016-06-17	39	5.57
1	8	4171	1.40	2016-05-09	2016-06-17	39	5.57
1	9	4200	1.42	2016-05-09	2016-06-17	39	5.57
1	10	4131	1.27	2016-05-09	2016-06-17	39	5.57
2	1	4159	1.37	2016-05-09	2016-07-04	56	8.00
2	2	4179	1.47	2016-05-09	2016-07-04	56	8.00
2	3	4152	1.33	2016-05-09	2016-07-04	56	8.00
2	4	4132	1.35	2016-05-09	2016-07-04	56	8.00
2	5	4177	1.22	2016-05-09	2016-07-04	56	8.00
2	6	4158	1.51	2016-05-09	2016-07-04	56	8.00
2	7	4182	1.12	2016-05-09	2016-07-04	56	8.00
2	8	4202	1.54	2016-05-09	2016-07-04	56	8.00
2	9	4143	1.28	2016-05-09	2016-07-04	56	8.00
2	10	4201	1.55	2016-05-09	2016-07-04	56	8.00
3	1	4155	2.06	2016-05-09	2016-07-21	73	10.43
3	2	4193	1.92	2016-05-09	2016-07-21	73	10.43
3	3	4170	1.80	2016-05-09	2016-07-21	73	10.43
3	4	4135	1.65	2016-05-09	2016-07-21	73	10.43
3	5	4190	1.87	2016-05-09	2016-07-21	73	10.43
3	6	4099	1.94	2016-05-09	2016-07-21	73	10.43
3	7	4198	1.49	2016-05-09	2016-07-21	73	10.43
3	8	4024	1.73	2016-05-09	2016-07-21	73	10.43
3	9	4044	1.53	2016-05-09	2016-07-21	73	10.43
3	10	4117	1.57	2016-05-09	2016-07-21	73	10.43
4	1	4173	2.20	2016-05-09	2016-09-14	128	18.29
4	2	4197	2.40	2016-05-09	2016-09-14	128	18.29

[1] grz-AD_..._E

Fish ID	Age at death (weeks)	Treatment group	RNA Integrity Number (RIN)	Date of library prep	Sequenced?	Reason for exclusion
1271	16	ABX_16	7.7	2018-11-20	Yes	–
1274	16	ABX_16	7.3	2018-12-01	Yes	–
1309	16	ABX_16	6.9	2018-12-01	Yes	–
dash	16	ABX_16	6.8	2018-12-01	Yes	–
1015	16	SMT_16	7.0	2018-11-20	Yes	–
1298	16	SMT_16	5.5	2018-11-20	Yes	–
1301	16	SMT_16	5.9	2018-11-20	Yes	–
402	16	WT_16	7.0	2018-11-20	Yes	–
938	16	WT_16	7.8	2018-11-20	Yes	–
940	16	WT_16	7.2	2018-11-20	Yes	–
1403	6	YI_6	5.5	2018-11-20	Yes	–
1409	6	YI_6	6.4	2018-12-01	Yes	–
1412	6	YI_6	6.7	2018-11-20	Yes	–
1414	6	YI_6	5.5	2018-11-20	Yes	–
1009	16	YMT_16	7.0	2018-11-20	Yes	–
1026	16	YMT_16	6.3	2018-11-20	Yes	–
1305	16	YMT_16	7.3	2018-11-20	Yes	–
999	16	YMT_16	7.1	2018-12-01	Yes	–
400	16	WT_16	7.1	–	No	Not enough RNA for library prep
1005	16	SMT_16	4.4	–	No	RNA integrity too low

Table D.25: Turquoise killifish used in IgSeq gut experiment. All fish are GRZ-Bellemans strain and male.