Caenorhabditis tRNA allelic variation analysis (github.com/paabylab/trnavcfalleles) Code details

Avery Davis Bell
Original: September 2025 | Updated: February 2026

This document describes inputs and outputs (or other relevant aspects) of all scripts in this repository that are run on their own – *i.e.,* if a script is only called via an included workflow, that script may not be fully documented here (but you can see workflow doc to figure out its use case). Scripts are grouped into their general categories, but be aware outputs of some scripts are used for multiple types of analyses.

All script paths here are internal to https://github.com/paabylab/trnavcfalleles (relative paths)

Table of Contents

Related to generating initial information about tRNA genes, alleles

generate_alleles/trnavcfvariants.nf

Workflow that generates all unique tRNA allele sequences; runs tRNAScan-SE to characterize them; collects summaries for this.

- Parameters/inputs

Category	Flag for script (in script as params.<this)	Description

Data input	--runinfo	/ tab-delimited file, one row per sample/species to run. Columns:
		// id, sample/species ID (used for output file naming)
		// trnasout, highest-confidence tRNAscan-SE .out filepath OR full - just keep track of whether you're including pseudogenes or not!
		// trnass, highest-confidence tRNAscan-SE secondary structure (usually .SS) out filepath
		// trnabed, tRNA bed file path
		// vcf, path to VCF for this species - used to find all the tRNA variants
		// vcftbi, path to VCF .tbi index
		// refstrain, ref strain to strip out before data summarization

Data output	--out	
software	--vcfconda	path to conda environment set up to run scripts that import pyvcf
software	--pyscriptdir	Directory containing python scripts run here - get_strain_variants.py etc
software	--rscriptdir	Directory containing R scripts run here (e.g. getstrainxtrnacalls.R)

- Processes

Name	Description	Any saved outputs? (all in output dir / ID of this species/sample) And relevant notes
subsetvcf	Slices vcf to only be tRNA regions (otherwise collectvarstrnas is quite slow) // AND Edits VCF so python vcf reader can work with it (unzips; removes "malformed filter lines", etc)	no (*but definitely resume this, it's big)*
collectvarstrnas	Runs get_strain_variants.py (Collects variants in tRNAs	in /data/ subdir _strain_variants.txt: variants in tRNAs

	(important preprocessing step for next scripts)) VCFs have to be unzipped here [there is built in for .gz but encoding issues seem to happen]	get_strain_variants.log: log file, *but also has record of who has missing genotypes. might be useful.*
buildaltseqs	Runs build_alt_sequences.py (Generates strain-specific tRNA sequences)	in /data/ subdir _strain_trnas_gen.txt, _strain_trnas.fa, _strain_trnas_info.txt
straintrnascanse	Run tRNAscan-SE on strain-specific FASTA	in /data/trnascan subdir: .out: main output file .SS: secondary structure .stats: about tRNAscan-SE run .isospecific.out: tRNA isotype info
missingness	Run catalogmissingness_trnavcfscripts.R to get record of where missing VCF genotypes are	in /data/ subdir: _trna_variant_info_wmissingness_pervariant.txt _trna_variant_info_wmissingness_pertrna.txt _trna_variant_info_wmissingness_perstrain.txt In /summaries/ subdir: _trna_variant_info_wmissingness_pertrna.pdf _trna_variant_info_wmissingness_perstrain.pdf
strainxtrna	Run getstrainxtrnacalls.R to get final data and summaries	in /summaries/ subdir - all generated files: _referencestrain<refstrain>_pertRNAalleles.txt (if ref strain provided, included) _strainsxtrnas_alleles_wmissing.txt.gz (***this is data though too)*** _alleleinfo_counts_wmissing.txt _strain_alleletype_counts_wmissing.txt _strain_alleletype_counts_wmissing_variablegenesonly.txt _bygeneup_tRNA_counts_wmissing.txt _bygeneup_codon_counts_wmissing.txt _bygeneup_AA_counts_wmissing.txt

generate_alleles/catalogmissingness_trnavcfscripts.R

Gets which strains have missing calls per variant, per tRNA

- Inputs (also get by running script with --help)

```
-v, --varinfo      Path to *_strain_variants.txt output of
                   get_strain_variants.py. **strains are inferred
                   from this
-t, --trnainfo     Path to *_strain_trnas_info.txt output of
                   build_alt_sequences.py
-b, --baseoutname  Base name for all output files [default: out]
-o, --outdir       Outer output directory. Sub-directories will be
                   created internally. **NB: if you provide getwd()
                   here (quote wrapped), current directory will be
                   used
```

- Outputs
  - *_trna_variant_info_wmissingness_pervariant.txt: variant information annotated with which tRNA it is in. Columns:

Chr	chromosome variant & tRNA are on
Pos	variant position
Ref	variant ref allele
Alt	variant alt allele
tRNA	tRNA ID (e.g. chrI.trna1)
Start	tRNA start
End	tRNA end
Strand	tRNA strand
AA	tRNA AA
Codon	tRNA codon
nhomRef	# strains with ref allele
nhomAlt	# strains with alt allele
nMissingOrHet	# strains with missing (or het called) genotypes
missingOrHet	comma-separated list of strains with missing (or het called) genotypes. If none for this variant, entry is 'None'

  - *_trna_variant_info_wmissingness_pertrna.txt : tRNA-level summary: number and identity of strains with missing-ness variants overlapping tRNA (or not where there aren't variants/missing variants). Here, missing/het is for *any or all* of the overlapped variants

- *Use this to annotate next level summaries, probably*
- Columns:
-

tRNA	tRNA ID (e.g. chrI.trna1)
Chr	chromosome
Start	tRNA start
End	tRNA end
Strand	tRNA strand
AA	tRNA AA
Codon	tRNA codon
nVCFVars	# variants from VCF were in this tRNA
nMissingOrHet	# strains with missing (or het called) genotypes
missingOrHet	comma-separated list of strains with missing (or het called) genotypes. If none for this variant, entry is 'None'

- *_trna_variant_info_wmissingness_perstrain.txt: quick strain level summary of number of tRNAs in that strain with missing variant calls. Columns:

  # strain, strain ID

  # nvars_missing, number of variants in tRNAs that had missing (or het) calls in this strain

  # ntrnas_missingvars, number of distinct tRNAs with one or more variants with missing (or het) calls in this strain

  # ntrnas_nomissingvars, number of tRNAs with no missing/het calls - just total number input minus previous number

- *quick summary plots*
  - *_trna_variant_info_wmissingness_pertrna.pdf - histogram showing number strains with missing calls (x axis) vs number tRNAs (y axis). Each bin is a single count.
  - *_trna_variant_info_wmissingness_perstrain.pdf - scatter plot showing how *many reference* tRNAs each strain has with any missing called variants (and total # missing-called variants on y axis)

generate_alleles/getstrainxtrnacalls.R

Getting strains x tRNA genes matrix with allele ID, including if there was any missingness
NB: 'Lost' is the given classification if : as pseudo gene (low inf but tRNAscan-SE does this) or if tRNAscan-SE doesn't find the generated fasta seq

- *How allele classifications are done:*

Classification	Criteria
Lost	not found by tRNAscan-SE, or found and called a pseudo gene (or secondary or undetermined isotype)
Altered	AA and alleleCM don't match (isotype switch) [or IPD in note - these are the same]
Invariant	only one allele in population AND not altered/Lost. **Deprecated - now we have a separate column for VariableInPop T/F, can use this to interpret other classifications**
Best	isotype matches what is expected and has the best infernal score for this tRNA should best be highest or lowest? highest based on what's classified as pseudo. [only confusion was in an old plot interpretation]
Functional	isotype matches what is expected, has a functional infernal score for this tRNA that is less than the Best score (Infernal > 20 but really these should all be flagged in note if fail this)

- Inputs (also get by running script with --help)

```
-t, --trnainfo     Path to *_strain_trnas_info.txt output of
                   build_alt_sequences.py
-m, --missinfo     Path to
                   *_trna_variant_info_wmissingness_pertrna.txt
                   output of catalogmissingness_trnavcfscripts.R
--trnascanout      Path to strain-specific fasta tRNAscan-SE output
                   (.out file)
--trnafasta        Path to strain_trnas.fa from
                   build_alt_sequences.py (used to check allele
                   names)
-r, --refstrain    ID of reference strain(s) to strip out of final data; comma-separated if more than one
-b, --baseoutname  Base name for all output files [default: out]
-o, --outdir       Outer output directory. Sub-directories will be
                   created internally. **NB: if you provide getwd()
                   here (quote wrapped), current directory will be
                   used
```

- Outputs
  - *Main data*
    - If reference strain is known and provided, it is removed from final data and saved separately:
      - _referencestrain<refstrain>_pertRNAalleles.txt
      - Columns: tRNA (tRNA gene ID), <refstrain>_allele (allele ref strain has there - generally Reference one, ha!)
    - *_strainsxtrnas_alleles_wmissing.txt.gz: *gzipped* Strains x tRNAs matrix with allele identified for each strain at each tRNA; NA if strain had any missing variant calls at that tRNA
      - columns are strain then each tRNA
  - *Data summaries (excludes reference strain duplication where reference strain ID provided)*
    - _alleleinfo_counts_wmissing.txt: **allele info & counts of strains with each allele.** Columns:

tRNA	tRNA ID
allelename	name of allele with date in this row
AA	amino acid the following DNA codon codes for (from tRNAscan-SE)
Codon	specific codon (from tRNAscan-SE) [DNA: reverse comp of the RNA code]
Infernal	infernal score (from tRNAscan-SE)
AlleleCM	expected amino acid based on backbone [*best guess, haven't found clear def of this field]* (from tRNAscan-SE) yep as far as I can tell - "The sequences were then grouped according to isotype, this time primarily based on which isotype-specific covariance model yielded the highest score for each sequence (regardless of the anticodon sequence). " (2021 tRNAscan-SE paper)
IsotypeScore	(from tRNAscan-SE)
Note	(from tRNAscan-SE)
VariableInPop	T or F, does this allele vary in the population
Classification	Lost, Altered, Functional, Best - see classification table above
n.allele	# strains with this allele

n.called	# strains with any allele called here (same for all alleles within gene)
n.missing	# strains with missing genotype calls for any variants in this tRNA (same for all alleles within gene)

- Updated 5/12/25 to make sure ones with no codon called by tRNAScan-SE are classified as pseudo/lost (some with 'Undet' amino acid/not-found codon were coming through as 'Altered')

- *_strain_alleletype_counts_wmissing.txt: strainwise count summary (*less useful for initial mutational variation piece, more for possible fitness differences/mutational burdens).*
  long format data with one row per strain, amino acid combination (and all together). Columns:
  - # strain,
  - # AA, amino acid alleles/tRNAs are combined for - if 'all', that's top-level strain summary
  - # <one for each classification - Altered, lost, best, functional> number of alleles called <this classification> for this amino acid in this strain
  - # nNA, number of alleles not called due to genotype missingness

    NB: [per AA is the ones actually coded for -- not the alleleCM. could be delivering wrong ones or...?]

- *_strain_alleletype_counts_wmissing_variablegenesonly.txt: same as above [columns *may* end up with different orders though], but only genes that have more than one observed allele in population used for summary - so can look at counts of just variable ones
- Summaries of number of alleles, genes, etc for each tRNA, codon, AA. *Note for all, AA is the one actually coded for from codon - so one mutated away from its original codon is counted in its new group*
  - *_bygeneup_tRNA_counts_wmissing.txt. one row per tRNA gene. Columns:

tRNA, gene ID
AA, AA codon codes for
Codon, actual codon in tRNA
VariableInPop, is gene variable in population
nAlleles, # alleles observed in population
anyMissingCalls, T or F, any alleles have any missing genotype calls in sequence
Best, number alleles classified as best
Functional, number alleles classified as functional
Altered, number alleles classified as altered
Lost, number alleles classified as altered

  - *_bygeneup_codon_counts_wmissing.txt. one row per triplet codon - the one that's actually CALLED, not the backbone! Columns:

AA, AA codon codes for
Codon, actual codon in tRNA
nGenes, # genes that have this codon
VariableInPop, is ANY gene for this codon variable in population
nAlleles, # alleles observed in population across any gene
nGenesMissingCalls, how many genes have alleles with any missing genotype calls (./. in VCF)
Best, number alleles classified as best across all genes
Functional, number alleles classified as functional across all genes
Altered, number alleles classified as altered across all genes
Lost, number alleles classified as altered across all genes

- *_bygeneup_AA_counts_wmissing.txt. one row per amino acid - the one that's actually CALLED, not the backbone! (from the codon)

AA, AA summarized here
nCodons, # codons for this AA in this genome
nGenes, # genes that have this AA
VariableInPop, is ANY gene for this AA variable in population
nAlleles, # alleles observed in population across any gene for this AA
nGenesMissingCalls, how many genes have alleles with any missing genotype calls (./. in VCF)
Best, number alleles classified as best across all genes for this AA
Functional, number alleles classified as functional across all genes for this AA
Altered, number alleles classified as altered across all genes for this AA
Lost, number alleles classified as altered across all genes for this AA

Related to comparing variants in tRNAs and protein-coding exons

variant_characterization/trnaothervcfgtcounts

Workflow (in several pieces) that gets genotype counts for variants of interest (those in tRNA regions and protein-coding exons). Calls some internal scripts in bin/ directory here.

- Parameters
  -

Flag	Default	Description

--runinfo	""	tab-delimited file, one row per sample/species to run. Columns:
		// id, sample/species ID (used for output file naming)
		// vcf, path to whole genome vcf to subset
		// vcftbi, path to .tbi index of above
		// trnabed, path to bed file for all tRNA genes of interest, name in 4th column
		// gtf, path to GTF file containing (at least) all protein-coding genes' exons

--out	""	output directory (multiple files created internally)

- Processes
  -

Name	**Description**	**Any saved outputs? Where?**
gtfexon2bed	Get bed files for exons (NOT from MtDNA)	*Yes, exon length information here:* *protcodexons.bed
bedsubvcfcts	Subset VCF based on bed file, keeping region name, then get per-genotype counts *Run 1x for tRNAs, 1x for prot-coding exons* ***Just for biallelic sites***	YES: *tRNA_gt_counts.txt.gz, gt counts for all tRNA variants *protcodexonic_gt_counts.txt.gz, gt counts for all exonic variants *Format: has header; columns are CHROM POS NAME REF ALT <gene name> <count homref> <count hom alt> <count 'het'> <'count missing'>*

variant_characterization/analyzetrnavarsfs.R

Analysis script that analyzes/plots minor allele frequencies, computes Tajima's *D*, etc from genotype counts of variants in protein-coding exons and tRNAs (downstream of trnaothervcfgtcounts workflow). Note this is for selfing, isogenic species (het samples are assumed to have a genotyping issue).

- Notes
  - This FINDS overlapping exons and restricts to just the longest of these (so not overcounting gene length but including all the variants)
  - Tajima's D doesn't natively deal well with missingness (different N); instead of doing a whole lot of math to model this really well, I'm going to restrict to the sites with >80% and >90% coverage
- Inputs (also get by running with --help)

```
-s, --speciesf     File containing information on all species to
                   process here. Columns infilename (exactly how all
                   files have this species in their name),
                   displayname (name that should be used for plot
                   outputs etc), shortname (no-spaces name for ouptut
                   files, sorting, etc - either shorter than or same
                   as infilename, probably).  In order you'd like
                   plots to be in!. **If not all files exist for each
                   species, only does analyses that it can for each
                   species**
-t, --trnabed      EXAMPLE path to bed files of tRNA gene locations
                   (4th column is name), as used to generate allele
                   counts input.  Where species ID/species specific
                   info is, put SAMP instead
-p, --protbed      EXAMPLE path to bed files of protein-coding exon
                   locations (4th column is name), as used to
                   generate allele counts input.  Where species
                   ID/species specific info is, put SAMP instead
--trnavarcts       EXAMPLE path to file with one row per variant in
                   any tRNA with genotype counts. Columns are CHROM,
                   POS, ID, REF, ALT, gene_id, n_homref, n_homalt,
                   n_het, n_miss Where species ID/species specific
                   info is, put SAMP instead
--protvarcts       EXAMPLE path to file with one row per variant in
                   any protein-coding exo with genotype counts.
                   Columns are CHROM, POS, ID, REF, ALT, gene_id,
                   n_homref, n_homalt, n_het, n_miss Where species
```

ID/species specific info is, put SAMP instead

-b, --baseoutname Base name for all output files [default: out]

-o, --outdir Outer output directory. Sub-directories will be created internally as needed. **NB: if you provide getwd() here (quote wrapped), current directory will be used

- Outputs
  - Data
    - In subdirectory data/
    - *Simply related to gene locations/curation*
      - *_bedrecords_lengthmergedinfo.txt.gz: All unique bed records as in input, but with annotations:

Geneclass	Protein or tRNA
Bp_length	End - start length
Overlaps_any_row	T if this exon overlaps another
Overlaps_any_gene	T if any exon in this gene overlaps another

      - *_bedrecords_mergedoverlappingexons.txt.gz: Bed records that are combined for used in downstream analysis. All overlapping protein-coding exons are merged (so can get better length metrics to match with other metrics). Standard bed/descriptive info plus:

Geneclass	Protein or tRNA
Bp_length	End - start length *of the merged region*
Overlaps_any_row	T if this exon overlaps another [if so it's merged]
Overlaps_any_gene	T if any exon in this gene overlaps another
N_merged_row	NaN if no overlap (this exact info was in input) number of exons that were merged if there were overlaps

      - *_bedrecords_combinedpergene.txt.gz: per-gene bed records for downstream analyses. Exons are combined for each gene, length is their length's sums (hence merging above). As above but start and end are first and last exon start/end, has had_merged_exon T or F (instead of n_merged_row etc). Also has n_exons record

- *_pervariant_counts_maf.txt.gz: Per-variant information - from input, but annotated and combined. Columns:
    -

displayname	from bed inputs & associated outputs
gene_id	
shortname	
geneclass	
chr	
start	
end	
bp_length	
n_exons	
overlaps_any_gene	
had_merged_exons	
CHROM	variant information (from input VCF processing)
POS	
REF	
ALT	
ID	
n_homref	
n_homalt	
n_het	
n_miss	
nStrainsWithGT	# strains with non-missing, non(bad) het GTs
pStrainsWithGT	Proportion of strains with non-missing, non(bad) het GTs (for filtering out sites for Tajima's D)
MAF_allstrains	Minor allele freq: minor count divided by total number strains

MAF_obsgts	Minor allele freq: minor count divided by number of strains with genotype here

- *_genotypecoverage_nonmissing_perset.txt: proportion of variants with different amounts of coverage across the gene sets
- *_tajimasd_pergeneset_estimates.txt: Tajima's D estimates and underlying numbers, estimates that went into it (number segregating sites, uncorrectied Pi and Watterson's theta)

- Related to minor allele frequency of each variant
  - *In subdirectory mafrelated/*
  - *_mafdistplots.pdf: lots of plots of the distribution of minor allele frequency in tRNA variants & protein coding gene variants (*no filtering on gene class)*
  - *_mafMWres.txt: MW test results comparing MAF distributions
  - *_proportionsingletonvariants.txt: proportion of each gene class's variants that are singletons; chi-sq on this

Related to secondary structure of tRNA (variant positions relative to secondary structure, secondary structure-aware alignments, etc)

secstruct_related/get_strain_variants_relpos.py

Gets variants internal to tRNAs (like get_strain_variants.py), but also includes their relative position (e.g. base number in acceptor stem).

Notes/caveats/method

- This WILL NOT assign a variant to multiple tRNAs - if tRNAs overlap, have a problem
    - *(laziness founded in biology*)
- If tRNA secondary structure isn't well parsed, info about relative position will be NA'ed out
- If tRNA has an intron, relative position is for *non-intronic length*
- Indels sometimes cause things to break
- *** see also notes for secstruct_related/secstruct2seqpieces.py
- Logic used to assign bases to structures (combined with secondary structure paired/unpaired calls):
- What about using KNOWN lengths of tRNA structures to guide? [using this in coding]
    - Require GOOD EVIDENCE to vary from this (except in variable loop)
    - For good looking tRNA:
    -

acceptor stem	1 non paired (base 73, often) 7 paired nts [but this can vary or have mismatches]
possible linker	0-…2?
d arm (paired)	usually looks like 4 paired
d loop	8-10? *varies?*
possible linker	0? 1+?
anticodon arm (paired)	5 paired nts [usually]
anticodon loop	easy to ID - has anticodon & other unpaired nts surrounding
variable loop	3+
t arm (paired)	4?
t loop (unpaired)	however many

    - SO, I can take this as baseline, and vary for 'nice' tRNAs that clearly have longer paired regions (*but not for pairing failures!)*
    - Possible structures [using this in coding]

Number (in tRNA gene)	What	optional or required?
1	acceptor stem (left)	required
2	acc-d linker	optional
3	d arm (left)	required
4	d loop	required

5	d arm (right)	reauired
6	d-ant linker	optional
7	anticodon arm (left)	required
8	anticodon loop (left of actual anticodon)	required
9	anticodon itself	required
10	anticodon loop (right of actual anticodon)	required
11	anticodon arm (right)	required
12	variable loop	required
13	t arm (left)	required
14	t loop	required
15	t arm (right)	required
16	t-acc linker	optional
17	acceptor stem (right)	required
18	acceptor stem overhang	required

Inputs (get by running with --help)

-trnass trnas.SS Path to tRNAscan-SE .SS file for tRNAs to process here
(reference tRNAs). Should have TRUE genomic
coordinates to match those in VCF

-vcf variants.vcf Path to VCF file for species whos tRNAs are provided

-out outfile.txt.gz Path to output file. If .gz, will be gzipped

-forcepseud [True, False]
During secondary structure processing: If gene is
pseudo with codon listed as NNN as 0-0, try to force
processing by trying codon at 34-36. (Default : True)

Outputs

- log file
- flags of when there were issues
- Main output file: one line per variant in VCF that was in one of the tRNA locations provided.
  Columns:
  chrom: chromosome
  pos: position of **VCF variant**
  ref: ref allele of variant - as in VCF
  alt: alt allele of variant - as in VCF
  nHomRef: # samples in VCF with 0/0 gts
  nHomAlt: # samples in VCF with 1/1 gts
  nNotMissingHet: # samples in VCF with 0/0 or 1/1 gts (not 0/1, missing)

tRNA: tRNA gene name
tRNA_strand: strand of tRNA
tRNA_pos: position of variant **relative to tRNA non intronic bases
structure: structure of tRNA this base is in
substructure: substructure of tRNA this base is in
nSubStructure: count substructure is from L to R (of all substructures)
substructure_pos: position of variant **within its substructure
nMissingHet: # samples in VCF with 0/1 or missing gts
homRef: comma-separated samples with 0/0 gts
homAlt: comma-separated samples with 1/1 gts
missingOrHettRNA: comma-separated samples with 1/1 or ./. gts

secstruct_related/trna2salignments.nf

Wrote python script that will break each tRNA allele into its secondary structure components (where possible); now want to run this and then align each structure component taking into account structure using 4sale.

- Parameters

Category	Flag for script (in script as params.<this)	Default value (if highlighted, need to provide)	Description

data	--runinfo	""	tab-delimited file, one row per sample/species to run. Columns:
			// id, sample/species ID (used for output file naming)
			// trnass, tRNAscan-SE .ss filepath for ALL alleles
			// specprefix, species prefix (no spaces) for FASTA seq naming for combining across species

data	--out	""	Outer/parent output directory (multiple directories created internally)
data	--alignmentout	"all_species_alleles_foursale_aligned.xfasta.gz"	File name for final alignment file
software param	--gapopen	0	-gapopen passed to foursale/clustal22
software param	--gapextend	0	-gapextend passed to foursale/clustal22
software param	--combfull	True	-full for fastax concat'ing: True or False: keep all sequences, like full/outer join
software param	--combfullfill	"N"	-fill for fastax concat'ing: "fill with N bases/residues for IDs missing in some files when using -full"

software param	--replaceU	True	-replaceU for fastax concat'ing: True or False: replace any Us in seq with Ts
software	--pyscriptdir		directory containing script secstructrelated/secstruct2seqpieces.py; utilityscripts/*concatxfastas.py*
software	--foursalerunjar		tring to put after java to get foursale to run. Should include all cp dependencies.
software	--clustalw2		/ fully articulated path to clustalw directory that runs with foursale (passed to foursale)
software	--params.vcfconda		path to conda environment set up to run scripts that import pyvcf (don't actually need pyvcf, but this one has other dependencies I use)

- Processes

Name	**Description**	**Any saved outputs? Where?**
secstruct2seqpieces	runs secstruct2seqpieces.py	ALL ouptuts of this script saved for future reference In directory: allelesecstruct (subdirectories for each species)
combinexfastas	Combines X fastas across species (within secondary structure pieces)	NA
foursale	Runs foursale for each secondary structure alignment piece	*no; these are intermediates*
combinealignments	Combines aligned outputs across secondary structures *With custom script: concatxfastas.py* *(seqkit works for fasta but not xfasta)*	Alignment in directory: fullalignments

secstruct_related/secstruct2seqpieces.py

Script that identifies sequence elements for each tRNA so, for example, broken up segments can be separately aligned (within secondary structure)

- NOTES on classifications
  - If stems have a mismatch, they are assigned the typical known length (in some cases, this could mean a linker or a bit of a different stem is included with them). *Am working to flag when things like this might be the case*
  - There ARE cases where borders are off by one - especially T arm vs. acceptor stem, for example [tricky cases where first acceptor stem base is unpaired, gets classified as having 2 bp overhang instead, etc]
  - See also notes for secstruct_related/get_strain_variants_relpos.py
- Inputs (can also get by running script with --help)

  -trnass trnas.SS Path to tRNAscan-SE .SS file for tRNAs to process here

  -outdir outdir Output directory

  -out outfilestem Prefix for output files

  -forcepseud [True, False]

  If gene is pseudo with codon listed as NNN as 0-0, try to force processing by trying codon at 34-36. (Default : True)

  -fasta [True, False] Write a fasta for each sensible secondary structure chunk of tRNAs. [default : True]

  -xfasta [True, False]

  Write a X fasta (where line after sequence is secondary structure in (). format) for each sensible secondary structure chunk of tRNAs. [default : True]

  -fanameprefix species_prefix

  Optional prefix for all sequences in output fasta/xfasta files - e.g., species identifier if files will be compined across multiple species.

  -maskanticodon [True, False]

  N out anticodons in output fasta/X fasta? (they are saved in their own FASTA) [default : True]
- Outputs
  - *_tRNA2struct_info.txt.gz: Information on tRNAs, one row per tRNA. NB doesn't include global stuff like position, strand as this doesn't make sense for strain-specific sequences that might be provided [has to come from elsewhere]. [doesn't include ones where couldn't split the seq well; DOES include ones with 'fake' codon positions if they were pseudogenes with 000 and NNN as codon in .SS file and script could make it work calling codon at 34-36]

    Columns (DESCRIPTIONS of them can be found in summary file):
    tRNA: gene name
    PossiblePseudogene: T or F
    Intron: T or F
    NoFlags: T or F. Not flagged at all - followed all rules perfectly as far as we can tell
    TooShort: T or F
    CodonIssue: T or F

ReqStructuresMissing: number of required tRNA cloverleaf structures missing in this one (should be 0)
RemainingBases: number of bases in tRNA not assigned to a structure (should be 0)
AnticodonArmIssue: T or F. Anticodon arm stem not as expected (could be just one unpaired base, or worse)
AccStemIssue: T or F. Acceptor stem not as expected (could be just one unpaired base, or worse)
DArmIssue: T or F. D arm not as expected (could be just one unpaired base, or worse)
DLoopIssue: T or F. Unable to identify D loop, VERY WEIRD & UNTESTED [*can happen when fake out pseudogenes to try to process them]*
TArmIssue: T or F. T arm not as expected (could be just one unpaired base, or worse)
TLoopIssue: T or F. Unable to identify T loop, VERY WEIRD & UNTESTED

- *_tRNA2struct_summary.txt: summary of number of tRNAs with various characteristics. Columns:
  Category: flag category (intron, codon issue, etc)
  Description: description of category (as defined in this script)
  N: number of tRNAs flagged for this category
- *_tRNA2struct.txt.gz: For each base in tRNA, what structure does it map to. For all tRNAs where this was doable (NOT necessarily all tRNAs), one line per base in non-intronic length of that tRNA [1 indexed].
  - Columns:
    - tRNA: gene name
    - pos.tRNA: position in the tRNA (1-length of tRNA) this row describes
    - Structure: overall structure this position is assigned to (e.g. 'D' for D arm)
    - SubStructure: substructure this position is assigned to (e.g. 'arm_L' for L/first part of paired arm before D loop)
    - num.structure: number this substructure is in the tRNA (with acceptor stem L starting at 1)
    - pos.substructure: position this base is within its substructure [for comparing across tRNA]
    - nucleotide: nt at this structure (from sequence in input SS file)
    - secstruct: sec structure in . >< format at this structure (from sec structure in input SS file)
- FASTA related (if fasta or x fasta or both specified by inputs): ***excludes any it couldn't predict for***
  - *_<secondary structure descrip>.<x>fasta: fasta/x fasta for each piece of each tRNA. Name is tRNA name optionally prefixed (e.g. with species name) if input dictates. Files written (<secondary structure descrip>):

accstemL2d	acceptor stem to d loop [half], not including any d
darm2ant	d arm to anticodon arm, not including any anticodon
antarm	anticodon arm *with actual anticodon masked to N if specified*
anticodon	Actual anticodon (for checking; for use later if N'ed it out)

varloop	variable loop (everything between anticodon arm and t arm)
tarm2acc	d arm to acceptor stem, not including any acceptor stem
accstemR2end	second half acceptor stem plus any overhang

Related to variation in tRNA flanking regions

flank_specific/trnaflankvcfvars.nf

Workflow that gets the variants in tRNA gene flanking regions, notes which strains have which variants.

- Parameters/inputs

Category	Flag for script (in script as params.<this)	Default value (if highlighted, need to provide)	Description
data	--runinfo	""	tab-delimited file, one row per sample/species to run. Columns: // id, sample/species ID (used for output file naming) // trnasout, tRNAscan-SE .out filepath // trnabed, tRNA bed file path // vcf, path to VCF for this species - used to find all the tRNA variants // vcftbi, path to VCF .tbi index
data	--out		Outer/parent output directory (one per sample/species ID created internally)
data	--totalflank	40	flank length to get for tRNA flanking bed. **Just for output bed file, this does NOT change how Corinne's script works
data	--innerflank	20	flank length for inner tRNA flank for output bed file (outer is this to total). **Just for output bed file, this does NOT change how Corinne's script works
software	--vcfconda		path to conda environment set up to run scripts that import pyvcf
software	--wormtrnarepo		Directory containing gitrepo python scripts run here - updatedinitial/get_strain_variants-flank.py and utilityscripts/bed2flanks.py

- Processes

<table>
<tr><th>Name</th><th>Description</th><th>Any saved outputs? (all in output dir / ID of this species/sample) And relevant notes</th></tr>
<tr><td>bed4vcf</td><td>Generate bed file that includes tRNAs and generous flanking regions</td><td>no</td></tr>
<tr><td>subsetvcf</td><td>slice VCF to be only tRNA + flanking regions; Edits VCF so python vcf reader can work with it (unzips; removes "malformed filter lines", etc)</td><td>no</td></tr>
<tr><td>strainflankvars</td><td>Runs Corinne's get_strain_variants-flank.py</td><td>*variants-flank.txt, info on variants in flanks (in Corinne's format)
*variants-flank.log, log file</td></tr>
<tr><td>bedflanks</td><td>Makes bed file with exact flank regions of interest (split into inner, outer, 5', 3' and named as such)
Generated from tRNA bed file

python script does this: flank_specific/bed2flanks.py
Inputs:
-h, --help show this help message and exit
-bed regions.bed Path to bed file containing regions for which to get flanking regions (e.g., tRNA genes)
-nflank N Length of desired total flanking region (on either side of region, not combined) in bp
-ninner N Length of desired INNER flanking region (on either side of region, not combined) in bp. Bases between this and nflank will be classified as outer flanking regions
-out out.bed Output file to contain flank regions bed</td><td>_trnas_flanks.bed: BED file with 6 flank regions per input tRNA:
<table><tr><td>Name</td><td>What</td></tr><tr><td>flank_total_5</td><td>Total flank length (in workflow defaults to 40) on 5' end</td></tr><tr><td>flank_inner_5</td><td>inner flank (in workflow defaults to 20) on 5' end</td></tr><tr><td>flank_outer_5</td><td>outer flank (in workflow defaults to 20) from total flank end to inner flank</td></tr><tr><td><same as each of above but _3></td><td><same as each above for 3' end></td></tr></table></td></tr>
</table>

		Idea here is that can use this to feed to various summary programs

flank_specific/flankvariation.R

Transforms data for downstream use; makes flank variant plots (doesn't include gene body)

- Notes on classifications, metrics, etc
  - For splitting tRNAs into active or inactive, active is ones where ANY strain has non-lost alleles. Inactive is where ALL strains have lost alleles (technically, all alleles classified as 'lost'). I don't actually think this is useful; more useful would be to use predicted gene activity a la Thornlow 2020
- Path/location
  - In wormtrna gitrepo
  - wormtrna/trnavcfvars/flankvariation.R
- Inputs (also get by running script with --help)

```
-s, --speciesf     File containing information on all species to
                   process here. Columns infilename (exactly how all
                   files have this species in their name),
                   displayname (name that should be used for plot
                   outputs etc), shortname (no-spaces name for ouptut
                   files, sorting, etc - either shorter than or same
                   as infilename, probably).  In order you'd like
                   plots to be in!
-f, --flankvars    EXAMPLE Path to file with tRNA flank variant info
                   output by get_strain_variants-flank.py.  Where
                   species ID/species specific info is, put SAMP
                   instead
-t, --trnainfoal   Path to file containing tRNA per allele
                   information including original codon & allele as
                   well as (potential) remolded one. -
                   *alleleinfo_counts_worigcodonetc.txt output of
                   show_mutational_variation.R
--trnainfog        EXAMPLE Path to file containing tRNA per gene
                   information including name & strand. -
                   *strain_trnas_info.txt output of
                   build_alt_sequences.py Where species ID/species
                   specific info is, put SAMP instead
-l, --lflank       Start and end of basepairs away from gene used in
                   input for 5' flank, comma-separated. E.g -40,0
```

[default: -40,0]

-r, --rflank Start and end of basepairs away from gene used in input for 3' flank, comma-separated. E.g 0,40 [default: 0,40]

-b, --baseoutname Base name for all output files [default: out]

-o, --outdir Output directory path. if getwd(), current will be used [default: out]

- Outputs
  - *_perpositiontRNAvarianceprop.txt.gz: key data (proportion of tRNAs with variation at each position) in looooong format. Columns:

Name	description
displayname	species name
strand	strand restriction (or not) for data in these rows - e.g., all tRNAs on + strand
genes	Genes restriction (or not) for data in these rows - all, active, or inactive (inactive =all alleles classified as lost)
varfreq	this is done for all variants (subset as other columns suggest) AND for only vars with allele freq < 0.05
mutclass	all, any.SNV, other [indels], or specific base change this data is for
. alleles	- for multi-mutation classes
. relpos	relative position in flank. **All tRNAs are now in terms of + direction - 3' or 5' call came from original strandedness, but relative position is for + direction**
flank	3' or 5'. **All tRNAs are now in terms of + direction - 3' or 5' call came from original strandedness, but relative position is for + direction**
n.tRNAs.var,	# tRNAs with variation at this position
n.tRNAs.invar,	# tRNAs withOUT variation at this position. NB missing not taken into account here - it's on a per tRNA not allele freq/per strain basis
p.tRNAs.var	proportion of tRNAs (missing excluded) that have variant at this position
low95ci.tRNAs.var	binomial 95% CI on proportion here - lower bound
high05ci.tRNAs.var	binomial 95% CI on proportion here - upper bound

  - *_locVpropvartRNAs_*<species>.pdf - Location relative to tRNA gene vs. proportion of tRNA genes that have mutation in that specific region. (Corrected for strandedness - all are plotted here where relevant). Each page has tRNA gene sets (all, active, inactive i.e.

NO non-pseud alleles) x All mutations or mutations with MAF < 0.05. Then, each page is a different strand and/or different mutation class/way of coloring mutation classes

Analyses of tRNA gene and allele data

analysis/show_mutational_variation.R

Big 'starter' script that does a lot of summarizing and plotting of tRNA mutational variation

- Inputs (also get by running with --help)

```
-s, --speciesf     File containing information on all species to
                   process here. Columns infilename (exactly how all
                   files have this species in their name),
                   displayname (name that should be used for plot
                   outputs etc), shortname (no-spaces name for ouptut
                   files, sorting, etc - either shorter than or same
                   as infilename, probably).  In order you'd like
                   plots to be in!
-a, --alleleinfo   EXAMPLE path to *alleleinfo_counts_wmissing.txt
                   output of getstrainxtrnacalls.R. Where species
                   ID/species specific info is, put SAMP instead
-t, --trnasgen     EXAMPLE path to *strain_trnas_gen.txt output of
                   build_alt_sequences.py. Where species ID/species
                   specific info is, put SAMP instead
--trnainfo         EXAMPLE path to
                   *_bygeneup_tRNA_counts_wmissing.txt output of
                   getstrainxtrnacalls.R. (Columns about tRNA; counts
                   of alleles w/ various classifications.)  Where
                   species ID/species specific info is, put SAMP
                   instead
-c, --chrlens      File containing lengths of chromosomes to use for
                   plotting; chromosomes/contigs not included from
                   those analyses/plots. Columns displayname
                   (matching species info in speciesf), Chr, Length
-b, --baseoutname  Base name for all output files [default: out]
-o, --outdir       Outer output directory. Sub-directories will be
                   created internally. **NB: if you provide getwd()
                   here (quote wrapped), current directory will be
                   used
```

- Outputs
  - Updated data

- *_alleleinfo_counts_worigcodonetc.txt: Same as --alleleinfo inputs, but with added columns to map the codon to AlleleCM instead of AA (so can do plots grouping things by estimated original codon/AA, not what the DNA seq codes for). New columns:
  -

freq.ofcalled	Frequency this allele is of all called alleles
freq.ofallinclmissing	Frequency this allele is of all strains
Codon.orig	what the codon was at the original amino acid this tRNA codes for - usually same as Codon, but in case of isotype switches, this helps group it with AlleleCM
AlleleCM.orig	what is the likely AlleleCM in original amino acid this tRNA codes for - so that all alleles from same gene can be grouped even in the (rare, I think) cases where AlleleCM differs
all.altered	usually F; or T - all alleles were flagged as isotype switches here; useful to know about this. CAN be this AND have classification be 'Lost'

  - (5/12/25 considering updating to make sure ones with no codon ID'd by tRNAscan-SE are classified as Lost…but that might should happen upstream….yep, added this upstream)
- Plots showing allele information
  - *All in subdirectory /alleleinfoplots*
  - *tRNA_allele_barplots.pdf: stacked bar plot where each gene is a bar; faceted by codon and amino acid
    - ***note this is for 'coded for' AA, not alleleCM/gene group
  - *<species>_tRNA_allele_stackedpies_<AASplit or AAPlusCodonSplit - are genes grouped within amino acid only/merged across codon, or faceted down to codon level>.pdf: Stacked pie charts, one pie per gene. Pages have different plot versions, gene sets as specified in titles and by looking at plot; each PDF has all for one species.
    - Here, genes are grouped with the 'original' codon they should go to and the 'original' **AlleleCM** for that gene.
    - if prop is < 0.03, pie slice is inflated (and total pie inflated with it) - just for visualization!
- Plots showing tRNA location (+ possibly other info)
  - *All in subdirectory /genelocplusplots*
  - These are made using *strain_trnas_gen.txt output of build_alt_sequences.py: number of alleles etc from there at least at first!! [not currently intersecting with classification etc]]
  - *_tRNA_locVsNAllelesEtc.pdf: location on chromosome of tRNA vs. number of alleles that tRNA has. Faceted by species. 16 versions - with and without

histogram showing N genes pileup; faceted & free y axes; normal and log scale y axes; pseudogenes noted and not
        - In areas of chromosome that one species has but not other - at the end- the species that doesn't go that long has grayed out region
    - *_tRNA_locByAA.pdf- Shows counts in 500kb along chromosomes split by amino acid. So can compare locations. 4 plots per species: all genes; excluding genes pseudo in ref; highlighting genes pseudo in ref; and highlighting genes with any pseudo alleles
        - Note: this is just COUNTS - harder to compare amino acids by eye since they have different n genes, but doing the proportion within amino acid is harder than worth the time *right now*
        - Right now, there's an 'NA' amino acid row added - this is just to get the coordinates right, didn't want to add counts of fake genes to real data. *Gotta be a better way…*
    - *Testing (and plotting) if tRNA genomic location distributions differ based on multiple characteristics*
        - Characteristics tested:

Pseudo. in ref.?
Any pseudo/lost alleles?
Variable in population?
Any strains have any missing genotype calls?

- *_tRNA_groupings_location_KStests.txt: results from Kolmogorov-Smirnov tests of genomic position distributions for each characteristic above, per chromosome and overall (overall = genomic coordinates; chromosomes added together). *P-values are also noted on plots.* Specific columns:
    -

displayname	species
Chr	chromosome or "all_combined" for overall genome distribution
tested	description of characteristic tRNA genes were split on - to test T and F location distributions against each other
ks.D.stat	test statistic
ks.p.value	p value

- *_tRNA_groupings_location_KStests.pdf: two plots per characteristic, faceted by chromosome and species, showing the distributions of genes TRUE for the characteristic vs. FALSE for the characteristic. KS per-chromosome p values annotated on facets; overall p-values are in subtitles. *Dark gray rectangles denote areas of the chromosome that don't exist in that species - i.e., where another species has a longer chromosome*
- Related to examining if pseudogene alleles in reference are functional elsewhere
    - *_pseudoinref_genes.txt - one row per gene that is pseudo in ref, contains total number alleles and number that might be functional and strain numbers for different allele classifications

- *_pseudoinref_summary.txt - per-species summary: n_pseudo_in_ref (# genes pseudo in ref), n_pseudo_in_ref_mult_alleles (# of these that are variable in population), n_pseudo_in_ref_has_fn_allele (# of these that have putatively functional allele)
- *_pseudoinref_details_putativelyfunc.txt -input (alleleinfo file) info for each allele from genes that are pseudo in ref but also have putatively functional alleles

analysis/genebodyvariation.R

Does some data re-formatting and plotting of variants internal to gene body (developed with secondary structure aware methods).

- Notes
  - First wanted to get prop tRNAs with variant each bp, but with different lengths this gets quite gnarly. Instead, going to get rate in each *substructure* [normalized to # *canonical* bases in that substructure

    # Get each site....a little tricky here...do within each structure instead, perhaps...

    # [if convert to certain # bps, how deal with when one tRNA has longer/shorter structure than canonical? average it out across the rest of the bases?]

    # kinda cleaner to do per structure then later (when plotting) assign 'canonical' len to each structure??
- Inputs (also get by running script with --help)

```
-s, --speciesf     File containing information on all species to
                   process here. Columns infilename (exactly how all
                   files have this species in their name),
                   displayname (name that should be used for plot
                   outputs etc), shortname (no-spaces name for ouptut
                   files, sorting, etc - either shorter than or same
                   as infilename, probably).  In order you'd like
                   plots to be in!
-g, --genevars     EXAMPLE Path to file with tRNA gene body info
                   output by get_strain_variants_relpos.py Where
                   species ID/species specific info is, put SAMP
                   instead
-t, --trnainfoal   Path to file containing tRNA per allele
                   information including original codon & allele as
                   well as (potential) remolded one. -
                   *alleleinfo_counts_worigcodonetc.txt output of
                   show_mutational_variation.R
--trnainfog        EXAMPLE Path to file containing tRNA per gene
                   information including name & strand. -
                   *strain_trnas_info.txt output of
                   build_alt_sequences.py Where species ID/species
                   specific info is, put SAMP instead
--ssflags          Path to file containing per-allele information
                   categorizing how well sec structure assignment
```

went ('info' file output of
secstruct2seqpieces.py) Where species ID/species
specific info is, put SAMP instead

--trnasecstruct Path to file with information on how tRNA sec
structures in --genevars input are arranged, etc.
Columns: structure (as in --genevars),
substructure (as in --genevars), nSubStructure (as
in --genevars), canonicalnbp - length of this
structure in a 72/73 bp tRNA [for plotting],
structure_plot and substructure_plot: categories &
names prettified/simplified for how you'd like to
plot them; structure_plot_level &
substructure_plot_level: RANKINGS of unique
structure_plot and substructure_plot for ordering
in plot

-b, --baseoutname Base name for all output files [default: out]

-o, --outdir Output directory path. if getwd(), current will be
used [default: out]

- Outputs
  - *_persecstructtRNAvarianceprop.txt.gz: Key data. Long! All structures x all species x all mutation types information. Columns:
    - displayname, species info
    - strand, Strand analysis is narrowed to for tRNA genes (or all)
    - genes, all, active, or inactive tRNAs
    - secstructflags, Was this analysis restricted to genes without any 'issues' in the python script calling their secondary structure? (Sometimes issues are no big, like a mismatch in acceptor stem; other times they could in theory throw off more of the structure)
    - mutclass - name of mutclass here
    - alleles - "-" if doesn't make a difference; Ref > Alt or Major > Minor for specific mutation classes
    - structure, substructure, nSubStructure, canonicalnbp - from --trnasecstruct input; information about which tRNA secondary structure piece is described in this row
    - n.tRNAs.var, # tRNAs with one or more variants in this substructure
    - n.variants, absolute # variants observed in this substructure
    - n.tRNAs.invar, # tRNAs with no variants recorded in this substructure
    - <p, low95ci, high95ci>.tRNAs.var.any - proportion & 95% CI (binomial) for number variant tRNAs over all. Not corrected by structure length.
    - <p, low95ci, high95ci>.varsPerBp - proportion & 95% CI (binomial) for number variants in region normalized to length of that region from input (may be

imperfect, but closer to per-bp number). Multiple variants from same tRNA count where relevant.

- plots
    - *NB: not perfect yet - showing 1bp is tricky & depends on how the thing works…*
    - 1 per species per any variants vs. variants corrected per bp. Pages and facets have all other varieties!
    - when mutations shown combined, 95% Cis plotted - as thin black lines above/below the main line

analysis/flankandgenebodyvar_visualize.R

Puts flank region variation and gene body variation together – downstream of analysis/genebodyvariation.R and flank_specific/flankvariation.R

- Inputs (also get by running script with --help)

```
-f, --flankpertrna     Flank variant location, proportion of tRNAs
                       info: *_perpositiontRNAvarianceprop.txt.gz
                       output of flankvariation.R
-g, --genevarspertrna  Within gene body variant info/proportion of
                       tRNAs: *_persecstructtRNAvarianceprop.txt.gz
                       output of genebodyvariation.R
-t, --trnasecstruct    Path to file with information on how tRNA sec
                       structures are arranged as used in
                       genebodyvariation.R call. Columns: structure
                       (as in --genevars), substructure (as in
                       --genevars), nSubStructure (as in --genevars),
                       canonicalnbp - length of this structure in a
                       72/73 bp tRNA [for plotting], structure_plot
                       and substructure_plot: categories & names
                       prettified/simplified for how you'd like to
                       plot them; structure_plot_level &
                       substructure_plot_level: RANKINGS of unique
                       structure_plot and substructure_plot for
                       ordering in plot
-s, --speciesf         File containing information on all species
                       processed in preceeding script. Columns
                       infilename (exactly how all files have this
                       species in their name), displayname (name that
                       should be used for plot outputs etc - in other
                       input files), shortname (no-spaces name for
                       ouptut files, sorting, etc - either shorter
                       than or same as infilename, probably).  In
                       order you'd like plots to be in!
-b, --baseoutname      Base name for all output files [default: out]
-o, --outdir           Output directory path. if getwd(), current
                         will be used [default: out]
```

- Outputs
  - A bunch of plots!
  - All tRNA gene body values are normalized to canonical n bp in that structure

- All genes included regardless of secondary structure flagging

analysis/trna_location_analyses.R

Performs various analyses of tRNA location with regard to protein coding gene location; genomic recombination domains; other tRNAs; etc

- Inputs (also get by running with --help)

```
-s, --speciesf    File containing information on all species to
                  process here. Columns infilename (exactly how all
                  files have this species in their name),
                  displayname (name that should be used for plot
                  outputs etc), shortname (no-spaces name for ouptut
                  files, sorting, etc - either shorter than or same
                  as infilename, probably).  In order you'd like
                  plots to be in!. **If not all files exist for each
                  species, only does analyses that it can for each
                  species**
-t, --trnasgen    EXAMPLE path to *strain_trnas_gen.txt output of
                  build_alt_sequences.py. Where species ID/species
                  specific info is, put SAMP instead
--trnainfo        EXAMPLE path to
                  *_bygeneup_tRNA_counts_wmissing.txt output of
                  getstrainxtrnacalls.R. (Columns about tRNA; counts
                  of alleles w/ various classifications.)  Where
                  species ID/species specific info is, put SAMP
                  instead
-g, --genelocs    EXAMPLE path to no-header, 3 column file
                  specifying all genes' locations for a species.
                  Columns: chr, start, end (NOT header'ed!)  Where
                  species ID/species specific info is, put SAMP
                  instead
-c, --chrdomains  EXAMPLE path to chromosome domain info files (tip,
                  arm, center) - rename them if you need to for this
                  convention to work.  Should have columns chr,
                  domain, subdomain, start, end.  Where species
                  ID/species specific info is, put SAMP instead
--chrlens         File containing lengths of chromosomes to use for
                  plotting; chromosomes/contigs not included from
                  those analyses/plots. Columns displayname
                  (matching species info in speciesf), Chr, Length
-b, --baseoutname Base name for all output files [default: out]
-o, --outdir      Outer output directory. Sub-directories will be
                  created internally. **NB: if you provide getwd()
                  here (quote wrapped), current directory will be
                  used
```

- Outputs
  - Misc - not location realted, but made here - possibly useful for other scripts
    - *_identicalseqgroupinfo.txt: Information on all seqeunces ***from input fastas*** *(may not match other inputs)*:
      - displayname, species
      - tRNA, tRNAscan-SE tRNA sequence ID, with 'chr' stripped if it was present
      - seq, sequence of that tRNA
      - seqlabel, unique sequence number - same sequences have the same number
      - nwseqlabel, number tRNAs with this same sequence in ref genome
    - *_tRNAgeneinfo_combined.txt: summary of all tRNA info pulled in or generated here. *NA when a tRNA didn't have that info, e.g. some seqs for elegans and domain for tropicalis*. Columns:

displayname: species

tRNA: tRNA name
seqlabel: **new,** what sequence GROUP is this tRNA in (all with this same label have same sequence)
nwseqlabel: **new,** how many other tRNAs in this ref genome have the same sequence
AA: amino acid *in reference* (from strain_gens)
Codon: codon *in reference* (from strain_gens)
Chr: chromosome of gene
Start: start pos of gene
End: end pos of gene
Strand: strand of gene
pos: midpoint of gene - for plotting one pos
domain: chromosomal recomb domain of midpoint where available
subdomain: "" but subdomain - L, r, etc
VariableInPop: multiple alleles, T or F
nAlleles: # alleles, from my summary
numStrains: # strains per allele
anyMissingCalls: any strains have missing calls
Best: # best alleles
Functional: # functional alleles
Altered: # switch alleles (PRELIM, may not match with full analysis later)
Lost: # pseud/lost alleles (PRELIM, may not match with full analysis later)

- *_tRNAgenecounts_refsequniqueness.txt: Summary of how many genes of different categories there are - total, with ref seq in fasta, UNIQUE gene sequences in genome, and this for genes with no pseud alleles, not all pseud alleles (*from early calling*)

- General gene location related
  - *_genecounts_per500kb_trnasandall.txt: For 500kb in the genome *(last bin on chr can be from 0.5-1.5 this bin length),* how many (number AND proportion) tRNA, non-tRNA genes are there. Columns:
    - displayname, species
    - description, which tRNA genes used here : gene set - all possible tRNAs (including pseud), functional tRNAs (at least one non-lost allele), Pseudo tRNAs (no non-pseud alleles)
    - genes, which genes are counted in this row? all (from background input geneset - prot coding genes expected), tRNA (tRNAs that also meet criteria in description),
    - chr, bin chromosome
    - start, bin start
    - end, bin end
    - bin.mid, bin midpoint
    - n, number of given genes here
    - p, proportion of all genes of the given category that are in this bin
    - p_low95ci, lower 95% binomail CI on p
    - p_high95ci, upper 95% binomial CI on p
  - *_allGsvstRNA_per500kb_plots.pdf: plots of above - line plots showing genes per 500kb; various gene sets, with and without 95% CI. Has domains overlaid.
- *Chromosome recombination domain related*

- All in subdirectory domain_analyses. **Only generated for species that had all necessary inputs.**
- *_geneCounts_subdomains_withperkb.txt: for multiple gene sets, counts of genes and number genes per Kb for each genomic subdomain. Columns:
    -

description	gene set - all possible tRNAs (including pseud), functional tRNAs (at least one non-lost allele), Pseudo tRNAs (no non-pseud alleles)
displayname	species nice format
shortname	species short format
chr	
domain	tip, arm, or center
subdomain	left or right, if appropriate
level	"subdomain" (can ignore)
genes	which genes are counted in this row? all (from background input geneset - protein coding genes in general), tRNA (tRNAs that also meet criteria in description),
N	number of genes in this region for these characteristics
N_per_kb	N/kb length of this region

- *_geneCounts_domainsup_withperkb.txt: for multiple gene sets, counts of genes in domains combined across chromosomes and genome wide (so, all arms on that chromosome or all arms genome wide; on a chromosome, center is the same for here or subdomain). *Columns are a subset of those above.*
- *_tRNAGeneProportion_subdomains.txt: what proportion of genes in each region are tRNAs. As above, tRNAs are also subset by *description* column. Number columns here are p (proportion), p_low95ci and p_high95ci (binomial 95% confidence intervals on proportion)
- *_tRNAGeneProportion_domainsup.txt: as above, but what proportion of genes in larger/summarized windows (e.g. all arms summed) are tRNAs. *Probably less useful.*
- *_withingeneset_regionchisq_allchrs.txt: chi-sq result for distribution of genes *within each category* across chromosomal domains, summed each chromosome. I.e., do arm and center proportion differ *within tRNAs*
- *_withingeneset_regionchisq_perchr.txt: chi-sq results as above (*within each category*), but done for each chromosome domain/subdomain set separately
- *_betweengeneset_regionchisq.txt: chi-sq results **testing proportion tRNAs in each domain vs. non-tRNAs in each domain.** Overall and per chromosome. For all tRNAs, functional only, pseudol only (*not testing pseudo & not against each other though)*
    - ***this is what would use to annotate plots, for example***
- *_genestrnas_perkbprops_indomains.pdf: shows genes per kb of tRNA, non-tRNA genes; proportion of genes on each chromosome that are in each arm

○ Related to distance among genes

- *Notes*
  - NOTE: codon here is from first file (_strain_trnas_gen.txt): identity of amino acid assigned by tRNAscan-SE *in reference genome*
  - NOTE, this is for all the seqs tRNAscan-SE IDs - NOT cleaned up narrower number from gtrnadb (for *C. elegans*, this means its my FASTA, not trnascan-se's)
- All in directory /pairwisedistance/
- *_tRNApairdistances.txt.gz: allll the data: *all* pairwise distances between tRNAs in input here (>816k currently!!), with characterizations of the 'type' of pair it is for splitting/analysis. Columns:
  - displayname, species
  - genepair, tRNA gene names paired (- separated, sorted) - so can match with any OTHER info about genes you might decide to add later…
  - pseud, 0, 1, or 2 - how many of the genes compared here had all alleles classified as Lost after first pass. ***early/imperfect pseud call***
  - AA, amino acid pair (from AA column of input) [sorted]
  - Codon, codon pair (from AA column of input) [sorted]
  - chr, chromosome (one if same chr, 2 sorted if not)
  - isoacceptor, T if pair is same AA, F if not
  - isodecoder, T if pair is same Codon, F if not
  - sameseq, T if pair has same sequence (seqlabel), F if not
  - lonelyseq, 0, 1, or 2 - how many of the genes are the only one of their classification that has it's seq ID (nwseqlabel = 1)
  - samechr, T or F: was pair on same chromosome
  - distance, distance in bp between genes. *arbitrarily 30Mb here if on diff chromosomes*
- *_tRNApairtypes_nsumm.txt: summary of the number of pairs with different classifications of interest. For all tRNAs (pseud & not) and excluding those with all pseud alleles.
  -

Different isoacceptor	not same AA; obligately also different isodecoder and different sequence
Same isoacceptor	any two genes with same AA
Same isoacceptor, different isodecoder	two genes have same AA, different codons [subset of isoacceptor]
Same isodecoder	same codon [subset of isoacceptor]
Same isodecoder, different sequence	same codon, sequence otherwise differs [subset of isodecoder]
Same sequence	genes have identical sequence [subset of isodecoder]

- *_tRNApairdistance_propsamechr_stats.txt: Results from proportion test comparing proportion of one set of gene pairs that are on the same chromosome vs another. Species, genes, sets, then stat test info (estimates, p values) shown. Comparisons done:
  -

set1name	set2name
Different isoacceptor	Same isoacceptor
Same isoacceptor, different isodecoder	Same isodecoder
Same isodecoder, different sequence	Same sequence
Different sequence	Same sequence

- *_tRNApairdistance_MWcompare_stats.txt: results from Mann-Whitney/Wilcox test testing distance distribution for tRNA pairs in different sets (see above - 'Comparisons done'). For all chromosomes combined but excluding chromosome-different pairs; each chromosome individually (to make sure chromosomal distribution not underlying differences); all combined where diff chromosome pairs arbitrarily assigned 30Mb apart [can mess up medians!]
- _tRNApairdistance_propsamechr_plot.pdf: for each of the gene sets/subsets (set1 or 2), plots proportion of genes on the same chromosome. *No Cis for now.*
- *_tRNApairdistance_vsseqdiv.pdf: hex tile plot of sequence divergence (new-ish input) between alignments vs. pairwise distance (for pairs where both had structure alignments). Many pages: all together vs split by chr; log scale distance vs not; different seq divergence metrics
- *Modeling distance on difference between sequence alignments*
  - *_tRNApairdistance_seqdiffs_glms_gammalog.RData: Quick save out of all the actual model objects in case want details long term. Named by all the info about them (ugly), described in other outputs. Outer list is of species - multiple models in each species.
  - *_tRNApairdistance_seqdiffs_glmsummary_gammalog.txt: The models run (descriptions, formula, etc) and key summary criteria - dispersion, AIC, and effects for the seq diff parameter, where included. Columns:
    - Displayname, species these results are for

      seqdiffdescrip: how are tRNA sequence alignment differences counted
      seqdiffmeasure: name of data column corresponding to above
      description: description of this model
      data.slice: any data reduction done (same chromosome; pseud in or out)
      use.formula: formula used for model
      model.AIC: AIC of model
      model.dispersion: dispersion of model (resid deviance squared / resid, close to 1 suggests good fit)
      seqdiff.beta: Estimate of seqdiff effect (raw/as is)
      seqdiff.beta.se: SE of seq diff effect
      seqdiff.tval: T value of seq diff effect
      seqdiff.pval: p value of seq diff effect
      seqdiff.foldchange: Each unit change in seq diff has this *multiplicative* effect on pairwise distance (exp(beta, which was in log scale))
      seqdiff.percentchange: percent greater pairwise diff is for each seq diff unit change (in % format - 0.1% means 0.1%, not 10%)

- *_tRNApairdistance_seqdiffs_glm_gammalog_plots_<species>.pdf: plots of the GLM models: *ignore for super nulls!*

  # Residuals vs Fitted: Look for patterns (should be random scatter).

Normal Q-Q: Deviance residuals should roughly follow a straight line.
Scale-Location: Checks homoscedasticity.
Residuals vs Leverage: Identifies influential points.

- *_tRNApairdistance_seqdiffs_glmgammalog_mcfaddensr2.txt: McFadden's pseudo-R2 : proportion deviance explained by fuller model vs. nuller model. Some rows are for isolating contribution of seq diff, others for chrom vs null
- *_tRNApairdistance_seqdiffs_glms_gammalog_anovas.RData: ANOVAs for JUST the ones that include seq diff (since terms added sequentially anyway). In case want the full info at some point; summaries & stats etc in next outputs
- *_tRNApairdistance_seqdiffs_glmgamma_anovapdev.txt: ANOVA results and prop/percent deviance explained for each model where seq divergence was one of the factors. Columns:
    - displayname: species
    - seqdiffdescrip: which measure of sequence difference used here
    - description: model description, including if pseud included or not
    - Df: Df from ANOVA (on glm)
    - Deviance: from ANOVA (on glm)
    - Resid. Df: from ANOVA (on glm)
    - Resid. Dev: from ANOVA (on glm)
    - Pr(>Chi): from ANOVA (on glm)
    - prop.modeltermdeviance: Proportion of the deviance explained by the terms in the model that this term explains (*not super useful)*
    - percent.modeltermdeviance: Percent of the deviance explained by the terms in the model that this term explains (*not super useful)*
    - prop.totaldeviance: Proportion of the TOTAL deviance that this term explains
    - percent.totaldeviance: Percent of the TOTAL deviance that this term explains

analysis/trna_location_hypdiv.R

Examine distribution of tRNA and protein coding genes with regard to hyperdivergent regions. Excludes mtDNA.

- *Inputs (also get by running with --help)*

```
-t, --trnainfo      Path to *_tRNAgeneinfo_combined.txt output of
                    trna_location_analyses.R. For species combined
-s, --species       Species that hypdiv info provided here is for
                    (writing at least first version to do one species
                    at a time). As specified in displayname column of
                    tRNA info file. [default: C. elegans]
--hypdivbed         Path to hyperdivergent regions bed file from
                    CaeNDR release matching the date, species used
                    here [default:
                    20220216_c_elegans_divergent_regions_strain.bed]
-p, --pgenelocs     Path to no-header, 3 column file specifying all
                    PROTEIN CODING (non tRNA) genes' locations for
                    this species.  Columns: chr, start, end (NOT
                    header'ed!)
-b, --baseoutname   Base name for all output files [default: out]
-o, --outdir        Outer output directory. Sub-directories will be
                    created internally. **NB: if you provide getwd()
                    here (quote wrapped), current directory will be
                    used
```

- *Outputs*
  - *_trnainfo_whypdivoverlap.txt: input tRNA info with new hypdiv.overlap column that contains T if overlaps hypdiv region in *any* strain, F otherwise
  - *_protcgeneinfo_whypdivoverlap.txt.gz: input protein coding gene info with new hypdiv.overlap column that contains T if overlaps hypdiv region in *any* strain, F otherwise
  - *_tRNAsVsProtCod_hypdivregions_chisq.txt: Chi-sq test results comparing hyperdivergent region overlap between tRNAs (different sets) and protein-coding regions. Lots of columns, all of which characterize this, give proportions, etc. Bonferroni correction is for number of tRNA sets separately compared against distinct protein coding set.

analysis/pairwisediffsrefseq.R

Script that gets sequence 'distance' between reference genome sequence alignments (number of nucleotides or gaps that don't match and number unique stretches of same). Needed for location analysis script; a bit circular as currently it also uses an output of that script.

- *Inputs (also get by running script with --help)*

```
-s, --speciesf     File containing information on all species
                   processed in preceeding script(s). Columns
                   infilename (exactly how all files have this
                   species in their name), displayname (name that
                   should be used for plot outputs etc - in other
                   input files), shortname (no-spaces name for ouptut
                   files, sorting, etc - either shorter than or same
                   as infilename, probably).  In order you'd like
                   plots to be in!
-x, --xfasta       Foursale alignment xfasta for ALL tRNA alleles in
                   ALL species of interest
-i, --introninfo   EXAMPLE path to *_tRNA2struct_info.txt.gz output
                   file of secstruct2seqpieces.py, used here to ID
                   genes with suspected introns vs not.  where sample
                   info goes, put SAMP instead (as in speciesf)
--seqgroupinfo     *_identicalseqgroupinfo.txt output of
                   trna_location_analyses.R: information on all
                   seqeunces from ref genome - columns: displayname,
                   species tRNA, tRNAscan-SE tRNA sequence ID, with
                   'chr' stripped if it was present seq, sequence of
                   that tRNA seqlabel, unique sequence number - same
                   sequences have the same number nwseqlabel, number
                   tRNAs with this same sequence in ref genome
-b, --baseoutname  Base name for all output files [default: out]
-o, --outdir       Output directory path. if getwd(), current will be
                   used [default: out]
```

- Outputs
  - *_alignedseqdiffs_uniqueseqpairs.txt.gz: the relevant data! For each pair of *unique reference* tRNA alignments (after 4sale etc) in each species, provides:
    - genepair, tRNA genes used here, '-' separated
    - seqlabelpair, seq label of the genes compared here (so can match with other genes), '-' separated
    - **intron,** *added later,* number of genes in pair with intron [introns are excluded from alignment so any diffs in intron won't be captured here]

    - ntNoMatch, number of elements in alignment [nts] that were NOT identical
    - ntMatch, number of elements in alignment [nts] that were identical
    - stretchNoMatch, number of distinct STRETCHES of alignments that don't match - i.e., if elements are F F F T, this is 1 stretch non-match, 3 nts non match
    - stretchMatch, number of distinct STRETCHES of alignments that match...not sure this will be used, but might as well have
- **NOTE** this is for all reference seqs where I was able to assign all tRNA secondary structures - NOT all reference sequences in the seqgroupinfo input
- *_alignedseqdiffs_uniqueseqpairs_hists.pdf: histograms of number of gene pairs with different numbers/measures of mis-matches

analysis/charisotypeswitching.R

Script that works to characterize/understand isotype switching in these species - including when its polymorphic vs. not; just getting short clear numbers of when it happens.

- Inputs (also get by running script with --help)

```
-s, --speciesf     File containing information on all species to
                   process here. Columns infilename (exactly how all
                   files have this species in their name),
                   displayname (name that should be used for plot
                   outputs etc), shortname (no-spaces name for ouptut
                   files, sorting, etc - either shorter than or same
                   as infilename, probably).  In order you'd like
                   plots to be in!
-a, --alleleinfo   Path to *alleleinfo_counts_worigcodonetc.txt
                   output of show_mutational_variation.R. For all
                   species in species info file combined.
-g, --genevars     EXAMPLE Path to file with tRNA gene body info
                   output by get_strain_variants_relpos.py Where
                   species ID/species specific info is, put SAMP
                   instead
--anticodontable   Path to anticodon table with columns dnaAnticodon
                   and AminoAcid. Should map directly to tRNA
                   anticodons here, including optional readings of
                   CAT as Met or iMet and of TCA, TTA, CTA as SeC
-b, --baseoutname  Base name for all output files [default: out]
-o, --outdir       Outer output directory. Sub-directories will be
                   created internally. **NB: if you provide getwd()
                   here (quote wrapped), current directory will be
                   used
```

- Outputs
  - *_nisotypeswitchsummary_all.txt: for all amino acids/codons combined, how many isotype switches are in each species and related numbers. Columns:
    - displayname: species
    - nSwitchAls: total number alleles observed with isotype switch
    - nGenesWithSwitches: total number of genes that have at least one allele with isotype switch
    - nGenesWSw.invariant: # genes with at least one isotype switch allele that only have one allele in population
    - nGenesWSw.variant: # genes with at least one isotype switch allele that only have multiple alleles in population

    - nGenesWSw.allSw: # genes with at least one isotype switch allele where ALL alleles have isotype switches
    - nGenesWSw.notallSw: # genes with at least one isotype switch where NOT ALL alleles have isotype switches. *Note these are obligately genes that do not have all.lost alleles*
    - nGenesWSw.allSw.variant: # genes where all alleles have isotype switches but there are multiple alleles in population
    - nAlleles: total number alleles
    - nAllelesNotAllPseud: total number alleles from genes that have some functional alleles
    - nAllelesNonSwitch: total number alleles that do NOT have isotype switch
    - nAllelesNonSwitchNotAllPseud: total number alleles that do NOT have isotype switch and are from genes that are NOT all pseud alleles
    - nGenes: total n genes
    - nGenesNotAllPseud: total n genes not having all pseud alleles
    - nGenesNoSwitches: total n genes having no isotype switch alleles
    - nGenesNoSwitchesNonPseud: total n genes having no isotype switch alleles that aren't all pseud alleles
    - nGenesNoSwitchesNonPseud.variant: total n genes with no isotype switch alleles and not all pseud alleles that have multiple alleles in the population
    - nGenesNoSwitchesNonPseud.variant: total n genes with no isotype switch alleles and not all pseud alleles that only have one detected allele in the population
- *_longformNsPs_from_genes.txt: Key values in long format as numbers and proportions. For each amino acid ORIGINALFROM this AlleleCM. *For plotting*. Columns:
    - from - "all": overall numbers
    - specific amino acid - numbers describe total number mutating from this amino acid (alleleCM)
    - specific codon - numbers describe number mutating from this specific codon
    - whatn - description of number in this row. All are number of genes, genes with all-pseud alleles excluded.
        - all switch - invariant: all alleles are switches, only one allele
        - all switch - variant: all alleles are switches, multiple alleles
        - some switches, some not - alleles have these characteristics
        - no switch - invariant: only one allele observed in population (*though some strains could have missing calls!!)*, not isotype switch
        - no switch - variant: no isotype switches gene count with this alleleCM; multiple alleles observed in population
    - displayname - species
    - n - number genes fitting these criteria
    - totalN - number genes summed across whatns for the other criteria (species, amino acid or whatever)
    - p - n/totalN
    - p.low95ci - binomial lower 95% bound on proportion
    - p.high95ci - binomial upper 95% on proportion
- *_longformNsPs_to_genes.txt: as above, but for going TO amino acids/codons in long form

        - Note for 'to' Sup and SeC are *combined* - Sup is the AA code tRNAScan-SE gives tRNAs with SeC codon that have other AlleleCM (vs SeC that looks like a canonical SeC)
        - ****do NOT trust the Codon stuff - this was taken from data and data doesn't always know/show what original codon was*****
    - *_genes_swfixedvarclassifs.txt: per tRNA, gives CLASSIFICATION of gene:
        # no_switch_fixed: no isotype switching, totally fixed in pop
        # no_switch_variable: no isotype switching, multiple alleles in pop
        # all_switch_fixed: all are SAME isotype switch, totally fixed allele in pop
        # all_switch_variable: all are SAME isotype switch, allelic variation in pop
        # all_switch_diff_variable: all are isotype switches but there can be DIFF isotype switches (to 2 new AAs, for example)
        # variable_switch: some alleles are isotype switches, some are not (could be multiple iso sw types)
        # NA - other, like all pseud or whatever
- Columns are:
    - displayname, tRNA, AA, Codon, AlleleCM, all.lost - as in other files
    - nAls, total number alleles at this gene
    - nSwNotSwClearAlleles, total alleles EXCLUDING ones with Undet amino acids (some lost ones were mucking up classification)
    - nSw, number switch alleles (not Undet AA)
    - nNotSw, number non-switch alleles (not Undet AA)
    - gene.classif - classification *see above.* NA in some cases, e.g. all alleles are pseudogenes
    - flagSwNotRare: flag the special case (here just one!) where variable_switch is classifciation but the switch allele is not rare and still joined by functional-looking non-switch alleles
- *_alleles_swfixedvarclassifs.txt.gz: All allele info, as in input, with annotations added by this script. Key new columns are those described in above *_genes_swfixedvarclassifs.txt file.
    - **Good allele-level downstream data (later, variant info also gets added)**
- Related to allele frequency
    - *_allelefreq_switchVnot_statres.txt: results testing distribution of allele frequencies from alleles that are isotype switches ("1" in output column name) and are not ("2" in output column name); genes with all-pseud alleles excluded.
    - *_allelefreq_switchVnot_hists.pdf: histograms showing allele frequency distribution split by whether allele is an isotype switch or not (first: count, second: proportion). Genes with all-pseudo alleles excluded.
- Specific amino acid to-from, codon to-from
    - **All in inner directory: byaacodon/**
    - NOTE for codons the original codon had to be *inferred:* what is the codon of the major allele at that gene. *So possible a few of these are not the appropriate switch!*
        - AND if ALL alleles at a gene are this switch, original codon will have been interpreted to be this one….*SO WON'T SEE SWITCHES FOR THESE!!*
    - *Data*
        - General column names in these files that are repurposed/added to/subtracted from depending on the specific information in that file:
            - displayname, species - from input
            - fromTo, from.val-to.val in case want already concatenated
            - from, from value (from input)

    - to, to value (from input)
    - nAlleles, # alleles that are from-to
    - nGenesThisSw, # genes with at least one from-to allele
    - nGenesThisSw.invariant, # genes with at least one from-to allele with only one allele (this one) in pop
    - nGenesThisSw.variant, # genes with at least one from-to allele with multiple alleles in pop
    - nGenesThisSw.allThisSw, # gnees with at least one from-to allele and all alleles in population are THIS isotype switch
    - nGenesThisSw.allThisSw.variant, # genes with ALL same from-to alleles AND multiple alleles in pop
    - nGenesThisSw.notallThisSw, # genes with at least one from-to allele and all alleles in pop are NOT THIS isotype sw
    - nAllelesThisFrom, # alleles in population that have this from value (to ignored) - like all the ones with leucine allelecm, for example
    - nAllelesThisFrom.notAllPseud, # alleles in population that have this from value (to ignored) - like all the ones with leucine allelecm, for example - excluding ones with all pseud alleles
    - nGenesThisFrom, # genes in population that have any alleles with this from value (to ignored) - like all the ones with leucine allelecm, for example
    - nGenesThisFrom.notAllPseud, # genes in population that have any alleles with this from value (to ignored) - like all the ones with leucine allelecm, for example - excluding ones with all pseud alleles
    - nGenesThisFrom.noSwNoPseud.invariant, # genes with this from value that have NO isotype switches and NO allelic variation (and all non-pseud alleles)
    - nGenesThisFrom.noSwNoPseud.variant, # genes with this from value that have NO isotype switches and have allelicvariation (and all non-pseud alleles)
    - nAllelesThisTo, # as with from, but for ones with the provided To value (e.g. leucine amino acid)
    - nAllelesThisTo.notAllPseud, # as with from, but for ones with the provided To value (e.g. leucine amino acid)
    - nGenesThisTo, # as with from, but for ones with the provided To value (e.g. leucine amino acid)
    - nGenesThisTo.notAllPseud # as with from, but for ones with the provided To value (e.g. leucine amino acid)
    - nGenesThisTo.noSwNoPseud.invariant, # genes with this from value that have NO isotype switches and NO allelic variation (and all non-pseud alleles)
    - nGenesThisTo.noSwNoPseud.variant, # genes with this to value that have NO isotype switches and have allelicvariation (and all non-pseud alleles)
- *_aafromto.txt.gz, Summary of n alleles etc for each amino acid from-to (allelecm-aa) combination.
- *_aafrom.txt.gz, summary of n alleles etc that mutate FROM the same backbone, per amino acid. First columns are counts for when they switch isotypes, last columns are overall counts of genes with this alleleCM (as in earlier output)
- *_aato.txt.gz, as above, but for those that mutate TO the same called AA
- *_tripfromto.txt.gz, as aafromto but for every pair of triplets/codons instead of amino acids. Amino acids also specified with columns. Includes codon pairs that wouldn't result in the amino acid changing.
- *_trfrom.all.txt.gz, as in aafrom, but for each triplet/codon. Includes if switches to a codon that doesn't change amino acid.

        - *_trfrom.aasw.txt.gz, as trfrom.all but first columns ONLY count those that cause AA switch. (latter columns count all switched, invariant, not , etc)
        - *_trto.all.txt.gz, as in trpfrom.all but mutating TO specified triplet
        - *_trto.aasw.txt.gz as in trpfrom.aasw but mutating TO specified triplet (for those that cause AA switch.)
    - *Plots*
        - *_SwNotVarNotbyAAFrom.pdf: number, proportion of tRNA genes with each AlleleCM ('from') that have vs don't have switches, where switches are variant vs. invariant (including where switch is vs. isn't fixed in all alleles or one allele)
            - All here are for genes where not all alleles are pseud!
            - Only shows Aas that were original AA (AlleleCM) at least once (i.e., not 'Sup')
        - *_SwNotVarNotbyAATo.pdf: as above, but for going TO that amino acid (so, alleleCM unknown from these plots - for all the ones with a codon that matches the given amino acid, how many *switched to* that one vs *'were always'* that one)
        - *_AAFromToHeatMaps.pdf: heat maps showing count/prop of genes
- Understanding the mutations underlying switches
    - **All in directory: mutpaths/**
    - Analysis notes
        - this EXCLUDES cases where not all alleles have the same AlleleCM, and cases where all alleles are pseud
        - original codon for each gene as in input: majority allele codon (IIRC)
        - Sometimes there are anticodon matches that don't match predicted changes between anticodons - simply flagging these (indels is one obvious case, occasionally there are others). *Possible these are places where alleleCM call is wrong?*
    - *Data*
        - *_ns_anticodonbackbonemutgenes.txt: itty little summary - number of genes overall and per gene classification that have predicted anticodon mutation or backbone mutation [put in long format for plotting]. Columns:
            - displayname, species
            - gene.classif, 'all' for all combined or each one that is observed (all_switch_fixed etc)
            - nSw, total N genes in this classification with either anticodon or backbone mutation (summed two)
            - nAnticodonMut, number with different codons (predicted anticodon mutations)
            - nBackboneMut, number with same anticodon (predicted backbone mutations)
        - *_anticodonmutations_allelepredictedpaths.txt: for all cases (alleles) where the anticodon in the allele is not in one for the alleleCM (anticodon mutations), details observed mutations in the tRNA and if they match any predicted paths - how many mutations had to happen; do all alleles have the same mutations; etc. Columns:
            - displayname, tRNA, allelename, freq.ofallinclmissing - allele info as in input
            - gene.classif - gene classification (as from *_genes_swfixedvarclassifs.txt)

AA, from input (amino acid this allele's anticodon matches)
initialAA, from input (amino acid backbone)
observed.antic, actual observed anticodon in allele
proposed.init.antic, anticodon this one mutated FROM (multiple if can't tell apart) - predicted or observed when possible
nMutsFromInit, number of mutations in proposed initial anticodon to get to eventual observed anticodon
whichMutsFromInit, which spot(s) in anticodon had to mutate to get to observed (; separated if multiple possibilities)
varObsInData, was this variant observed in population variation data, T or F?
FLAG.anticodVarButNoMatch : if T, there WAS an anticodon mutation flagged but it didn't match data - have seen this when, e.g, indels in anticodon arms
nPossibleMutations, count of # mutations that could lead to what we see (can't narrow further)
varObsPos, if anticodon variant observed - its genomic position
varObs.tRNApos, if anticodon variant observed - its position in tRNA
varObs.ref, if anticodon variant observed - its ref allele
varObs.alt, if anticodon variant observed - its alt allele
varObs.anticPos, if anticodon variant observed - its position in anticodon
varObs.nHomAlt, if anticodon variant observed - n strains homozygous alt allele here
nNonAnticVariants, if this allele has variants NOT in anticodon, how many
structureNonAnticVariants, if this allele has variants NOT in anticodon, where are they (,-separated)
posNonAnticVariants, if this allele has variants NOT in anticodon, genomic position (,-separated)
consistentAnticMutAcrossAlleles, T if all alleles of this gene in input have same predicted [or obs] anticodon mutations; F otherwise

- *_anticodonmutations_pergenesummaries.txt: like above but summarizing number of alleles PER tRNA GENE with various characteristics
- *_anticodonmutations_perspeciessummaries.txt: OVERALL summaries of counts of alleles, genes with different anticodon mutation characteristics. For all gene classifications combined AND, more relevantly, within gene classification (fixed vs variable switch alleles, etc). Columns:
  -

displayname	species
gene.classif	classification for this row's numbers - 'all' is summed/combined across all
nAllelesWAntMut	# alleles that have predicted anticodon mutations

nAlsVarObsInDat	# alleles that have variants observed in VCF that match a predicted anticodon mutation (based on AA, alleleCM)
nAlsFLAG.anticodVarButNoMatch	# alleles that have variants observed in anticodon that DON'T match prediction (e.g., indels)
nAlsConsistentAnticMutAcrossAlleles	# alleles where all anticodon-switch alleles from the same gene have the same mutations (observed or predicted)
nAls1MutPath	# alleles with predicted 1 mutation between anticodons underlying switch
nAls2MutPath	"" for 2 mutations
nAls3MutPath	"" for 3 mutations
nAlsOnePossMut	# alleles where there's only one possibility traversing between the two anticodons
nAlsMoreOnePossMut	# alleles where there's more than one possibility to get between the two anticodons
nAlsNonAnticVars	# alleles where there are observed mutations in the tRNA outside the anticodon
nGenesWAntMut	# genes that have 1+ alleles with anticodon mutation
nGeneswAllelewVarObsInDat	# genes with ANY alleles with observed variants
nGenesAllAlleleVarObsInDat	# genes with ALL alleles having observed variants
nGenesAnyAlleleFLAG.anticodVarButNoMatch	# genes with ANY alleles having flag [see above]
nGenesConsistentAnticMutAcrossAlleles	# genes with all anticodon-switch alleles having the same anticodon mutation (predicted or observed)
nGenesAllAl1MutPath	# genes with ALL allele with one possibility between anticodons
nGenesAny1MutPath	# genes with ANY allele with one possibility for anticodon switch

nGenesAllAl2MutPath	"" for 2 mutations
nGenesAnyAl2MutPath	"" for 2 mutations
nGenesAllAlOnePossMut	# genes where ALL alleles have only one possibility to get between the two anticodons
nGenesAnyAlOnePossMut	# genes where ANY allele has one possibility to get between the anticodons
nGenesAllAlNonAnticVars	# genes where ALL alleles have tRNA mutations outside of anticodon (from VCF)
nGenesAnyAlNonAnticVars	# genes where ANY alleles have tRNA mutations outside of anticodon (from VCF)

- *_backbonemutations_allperallele.txt: For alleles that have same anticodon sequences but diff AA / AlleleCM (assumed backbone ones), summary of the actual variants observed there. Columns:
    - allele descriptors from input (same for each row of an allele): many
    - variant descriptors from input (different for each row of an allele): chrom, pos, ref, alt, nHomRef, nHomAlt, nNotMissingHet, nMissingHet, tRNA_strand, tRNA_pos, structure, substructure, nSubStructure, substructure_pos
- *_backbonemutations_summaryperallele.txt: Summary of above - summary of variants per allele, one row per allele. Columns:
    - allele descriptors from input (same for each row of an allele): many
    - # nVars, number variants (from VCF) observed in this allele
    - # nStructuresWithVars, number structures variants come from
    - # nSubstructsWithVars, number structure-substructure pairs variants come from
    - # nAcceptorVars, number variants from acceptor stem
    - # nDVars, number variants from d arm
    - # nTVars, number variants from t arm
    - # nVLVars, number variants from variable loop
    - # nAntArmVars, number variants from anticodon arm (arm, loop, actual anticodon)
    - # nOtherStructVars, number variants from any other called tRNA structure
    - # nNotCalledInStructVars, count where there are variants that I couldn't assign to a tRNA substruct
    - # varSubstructs, comma-separated list of structure-substructure pair variants in this allele are in
    - # flagNoObsVar, flag: T if all of these are 0s/ no observed variants
- *_backbonemutations_pergenesummaries.txt: For each gene, count of alleles (with backbone mutation) with various number, location backbone mutations. Column naming conventions follow above file.
- *_backbonemutations_perspeciessummaries.txt: OVERALL summaries of counts of genes with different backbone mutation characteristics. For all gene classifications combined AND, more relevantly, within gene classification (fixed vs variable switch alleles, etc). Columns:
    -

displayname	species

gene.classif	classification for this row's numbers - 'all' is summed/combined across all
nGenesBBMut	number genes with 1+ alleles with isotype-switch causing backbone mutation predicted
nGenesAllAlFlagNoObsVar	# genes with *all* bb mutation predicted alleles where there is NOT observed allele
nGenesAnyAlFlagNoObsVar	"" but for *any* 1+ alleles
nGenesMultBBMutAlleles	# genes with more than one allele predicted to have backbone mutation
nGenesAllAl1BbMut	# genes where *all* bb predicted alleles have 1 mutation
nGenesAnyAl1BbMut	"" but for *any* 1+ alleles
nGenesAllAlMultBbMut	# genes where *all* bb predicted alleles have >1 mutation
nGenesAnyAlMultBbMut	"" but for *any* 1+ alleles
nGenesAllAlAcceptorVars	# genes where *all* bb predicted alleles have 1+ predicted acceptor stem variants
nGenesAnyAlAcceptorVars	"" but for *any* 1+ alleles
nGenesAllAlDVars	# genes where *all* bb predicted alleles have 1+ predicted D arm variants
nGenesAnyAlDVars	"" but for *any* 1+ alleles
nGenesAllAlTVars	# genes where *all* bb predicted alleles have 1+ predicted T arm variants
nGenesAnyAlTVars	"" but for *any* 1+ alleles
nGenesAllAlVLVars	# genes where *all* bb predicted alleles have 1+ predicted variable loop variants
nGenesAnyAlVLVars	"" but for *any* 1+ alleles
nGenesAllAlAntArmVars	# genes where *all* bb predicted alleles have 1+ predicted anticodon arm variants
nGenesAnyAlAntArmVars	"" but for *any* 1+ alleles
nGenesAllAlOtherStructVars	# genes where *all* bb predicted alleles have 1+ predicted variants in other structures than above
nGenesAnyAlOtherStructVars	"" but for *any* 1+ alleles

nGenesAllAlNotCalledInStructVars	# genes where *all* bb predicted alleles have 1+ predicted variants that didn't' have location classifications (e.g. indels)
nGenesAnyAlNotCalledInStructVars	"" but for *any* 1+ alleles

- *_ns_anticodonbackbonemutgenes_obsVnot_varswitch.txt: cross-tabulation, for variable switch genes, of number with Anticodon vs Backbone mutations and of those how many I had any observed variants for. Note: anticodon 'observed' means observed *and* matches something predicted; backbone is just observed (more lenient/no functional test)

analysis/trna_trees_init.R

Script that makes neighbor-joining trees of tRNA allele multiple sequence alignments (made in secondary structure aware fashion), messes around with plotting. Unlike most scripts, Right now plenty in here is **hard coded** for current species. Would have to update for others.

- Inputs (get with --help, mostly just doing it w/in Rstudio though)

```
-s, --speciesf     File containing information on all species
                   processed in preceeding script. Columns infilename
                   (exactly how all files have this species in their
                   name), displayname (name that should be used for
                   plot outputs etc - in other input files),
                   shortname (no-spaces name for ouptut files,
                   sorting, etc - either shorter than or same as
                   infilename, probably).  In order you'd like plots
                   to be in!
-a, --alleleplus   *alleleinfo_counts_worigcodonetc.txt output of
                   show_mutational_variation.R, with all allele info
                   including Codon.orig, AlleleCM.orig,
                   VariableInPop, Classification, etc
-p, --pseudref     *_pseudoinref_genes.txt output of
                   show_mutational_variation.R, with info on if each
                   gene is all pseud
-x, --xfasta       Foursale alignment xfasta for ALL tRNA alleles in
                   ALL species of interest
-b, --baseoutname  Base name for all output files [default: out]
-o, --outdir       Output directory path. if getwd(), current will be
                   used [default: out]
```

- Outputs
  - *_trees.Rdata: saves workspace every time a big computation is done (and so can use the NJ trees again another time - generating them is somewhat slow)
  - *Plots - info*
    - Allele frequency when shown is number with this allele/total number strains (including those with missing), not proportion of called strains
  - *Plots - file names (?)*
    - *_phylotrees_allspeciesgenes.pdf: all variants of phylo (branches unequal lengths) including all species and all amino acids. Different colorways, different inclusion/exclusion of pseudogenes.

- *_cladotrees_allspeciesgenes.pdf: as above, but trees are cladograms (branch lengths not considered - all go to edge)
- *phylotrees_<species>.pdf - tree for alleles across Aas in each individual species. Gene, colorway subsets noted in title.
- *cladotrees_<species>.pdf - cladograms for alleles across AAs in each individual species. Gene, colorway subsets noted in title.
- *_trees_<specific amino acid>.pdf: Trees for specific amino acids (by that a.a.'s BACKBONE - all originally a leucine plotted, etc; excluding genes where all alleles are pseudo). First page is tree, second is cladogram.
    - **These for 4 that I picked specifically, later made for all amino acids**
- (added later) *_phylotrees_eachAA.pdf, Trees for specific amino acids (by that a.a.'s BACKBONE - all originally a leucine plotted, etc; excluding genes where all alleles are pseudo). Typical tree.
- (added later) *_cladotrees_eachAA.pdf, Trees for specific amino acids (by that a.a.'s BACKBONE - all originally a leucine plotted, etc; excluding genes where all alleles are pseudo). Cladogram!

analysis/investigate_compmuts.R

Brief look at where variants in tRNAs are: are they in stems that are paired? (does NOT get into if the mutations are actually in paired nucleotides, though.) Narrowing to paired stem regions specifically of acceptor, d, anticodon, t arms; Finding cases where there's at least one mutation in one side of the paired stem and one in the other. (if these), see if any are in the same strains. Excluding INDELs - too hard to manage & less reliable

- Inputs (also get by running script with --help)

```
-s, --speciesf       File containing information on all species to
                     process here. Columns infilename (exactly how
                     all files have this species in their name),
                     displayname (name that should be used for plot
                     outputs etc), shortname (no-spaces name for
                     ouptut files, sorting, etc - either shorter than
                     or same as infilename, probably).  In order
                     you'd like plots to be in!
-g, --genevars       EXAMPLE Path to file with tRNA gene body info
                     output by get_strain_variants_relpos.py Where
                     species ID/species specific info is, put SAMP
                     instead
-t, --trnasecstruct  Path to file with information on how tRNA sec
                     structures in --genevars input are arranged,
                     etc. Columns: structure (as in --genevars),
                     substructure (as in --genevars), nSubStructure
                     (as in --genevars), canonicalnbp - length of
                     this structure in a 72/73 bp tRNA [for
                     plotting], structure_plot and substructure_plot:
                     categories & names prettified/simplified for how
                     you'd like to plot them; structure_plot_level &
                     substructure_plot_level: RANKINGS of unique
                     structure_plot and substructure_plot for
                     ordering in plot
--trnainfo           Path to *_genes_swfixedvarclassifs.txt tRNA
                     gene-characterizing output of isotype switch
                     variation script. For species combined
--talleleinfo        Path to tRNA allelic information
                     (_alleles_swfixedvarclassifs.txt.gz)
--ssfile             Path to EXAMPLE file with tRNA secondary
```

structures (as output by tRNAscan-SE). For all alleles INCLUDING ref.

-b, --baseoutname Base name for all output files [default: out]

-o, --outdir Output directory path. if getwd(), current will be used [default: out]

- Outputs
  - *_strainSharingStemSNVPairs.txt: One row per pair of SNVs that are in same stem but on opposite sides - BEFORE checking for if they're actually paired or not. Lots of columns:
    - tRNA information for the tRNAs they're in, combined from input:
      - tRNA, AA, Codon, AlleleCM, all.lost, nAls, nSwNotSwClearAlleles, nSw, nNotSw, nTypesSw, gene.classif, flagSwNotRare
    - Structure information about both mutations
      - structure, shortname, chrom, tRNA_strand
    - Variant information on the first ('Left') mutation, from variant input
      - pos_1, ref_1, alt_1, nHomRef_1, nHomAlt_1, nNotMissingHet_1, nMissingHet_1, tRNA_pos_1, substructure_1, nSubStructure_1, substructure_pos_1
    - Variant information on the second ('Left') mutation, from variant input
      - pos_2, ref_2, alt_2, nHomRef_2, nHomAlt_2, nNotMissingHet_2, nMissingHet_2, tRNA_pos_2, substructure_2, nSubStructure_2, substructure_pos_2
    - Information on who shares these variants
      - nInCommon: number of strains that were homozygous alt for both mutations
      - whichInCommon: comma separated names of these strains
  - Per species summary of number of these potential pairs above, broken down into some tRNA categories (e.g. pseud vs not):
    - *strainSharingStemSNVPossPairs_summary_nbylost.txt: Number in pseud vs. not tRNAs
    - *strainSharingStemSNVPossPairs_summary_nbyswcl: Number by Isotype switch classification (non-pseud only)
    - *strainSharingStemSNVPossPairs_summary_nbysing.txt: Number by if 0, 1, or 2 of the SNVs in pair are singletons (split by pseud vs. not)
  - *_tRNAsNonPseudwstrainSharingStemSNVPairs_alleleinfo.txt for all the tRNAs with strain(s) with stem mutations (NOT checked if paired yet) in *putatively functional genes,* their allelic breakdown tagged with information about whether this allele captures the pair of SNVs, some of them, etc. Columns not in input:

    # is.reference, is this allele the reference
    # is.pair, this allele is one with at least one putatively comp. pair
    # n.pairs.in, count of N potentially compensatory pairs this allele harbors
    # has.pair.mut, T/F potential comp pair mutation (any!) seen in this allele
    # has.nonpair.mut, T/F other mutations that aren't in comp pair
    # n.muts, total N non-ref muts in allele
- *_tRNAsNonPseudwstrainSharingStemSNVPairs_alleleinfo_dotplot.pdf: plot of above plotting all alleles from a tRNA colored by pair mutation status, y axis is allele frequency