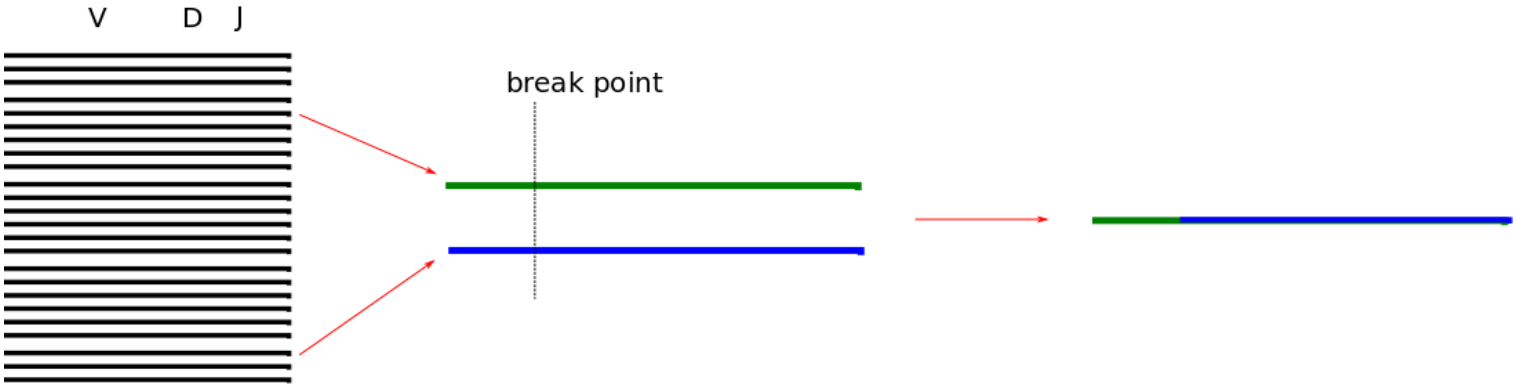
Chimera detection in BCRs

Duncan Ralph

now

Simulate PCR chimeras

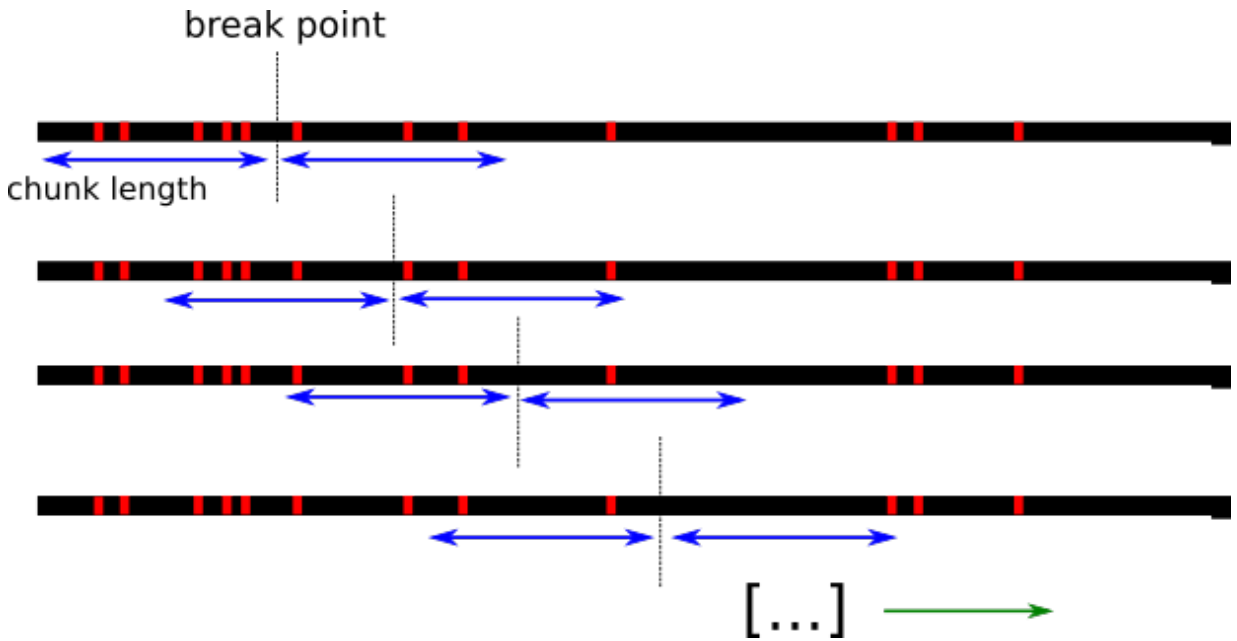


- Take a few thousand BCR sequences
- For each sequence A, choose a random break point
- Randomly choose a different sequence B from the same sample
- Replace the part of the sequence A before the break point with the corresponding part of sequence B
- https://github.com/psathyrella/partis/blob/dev/bin/add-chimeras.py

Detect chimeric sequences (1)

- Look for a position where the SHM rates on either side are very different
- Define a characteristic chunk length
  - Calculate SHM rates within this length on either side of potential break point
  - Shorter: better sensitivity, longer: better specificity
  - Played around with 15-150, 75 seems good

Detect chimeric sequences (cont'd)



- For each position in sequence (further than chunk len from ends)
  - Get SHM rate of chunk len before, and chunk len after
  - Calculate absolute difference
- Position at which this absolute difference is a maximum is the most likely chimera break point for this sequence

Max abs difference examples

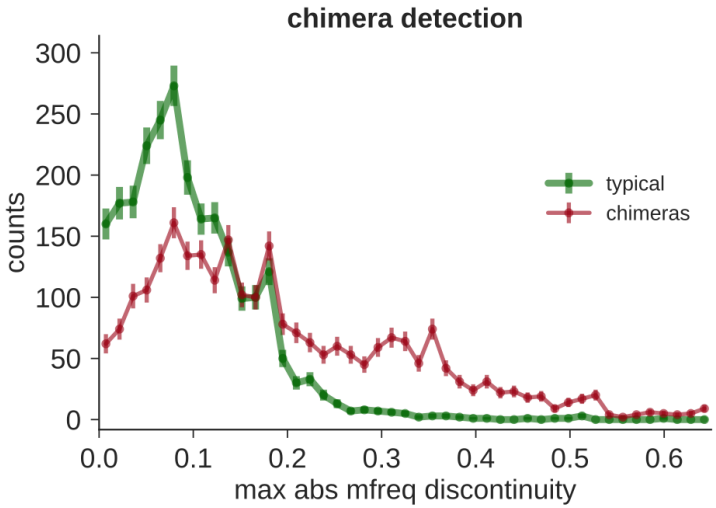
0.5: Clearly chimeric

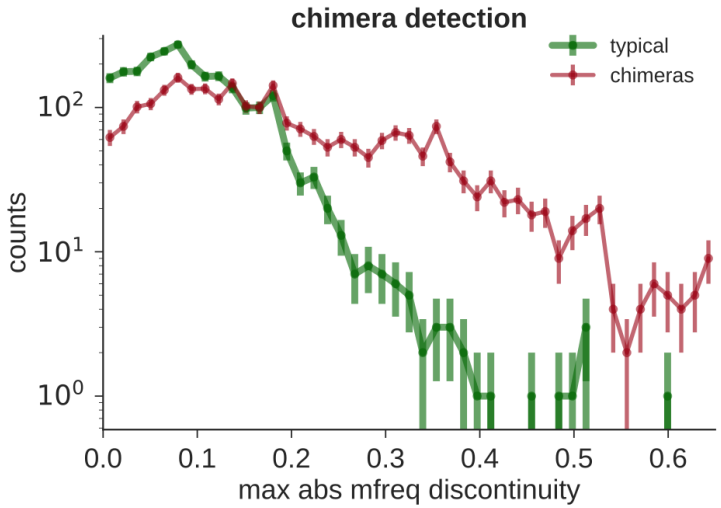
0.3: kinda marginal

0.2: don't look chimeric

Distributions on simulation

- Compare sample with no chimeras to the same sample modified to be 100% chimeras (linear + log)

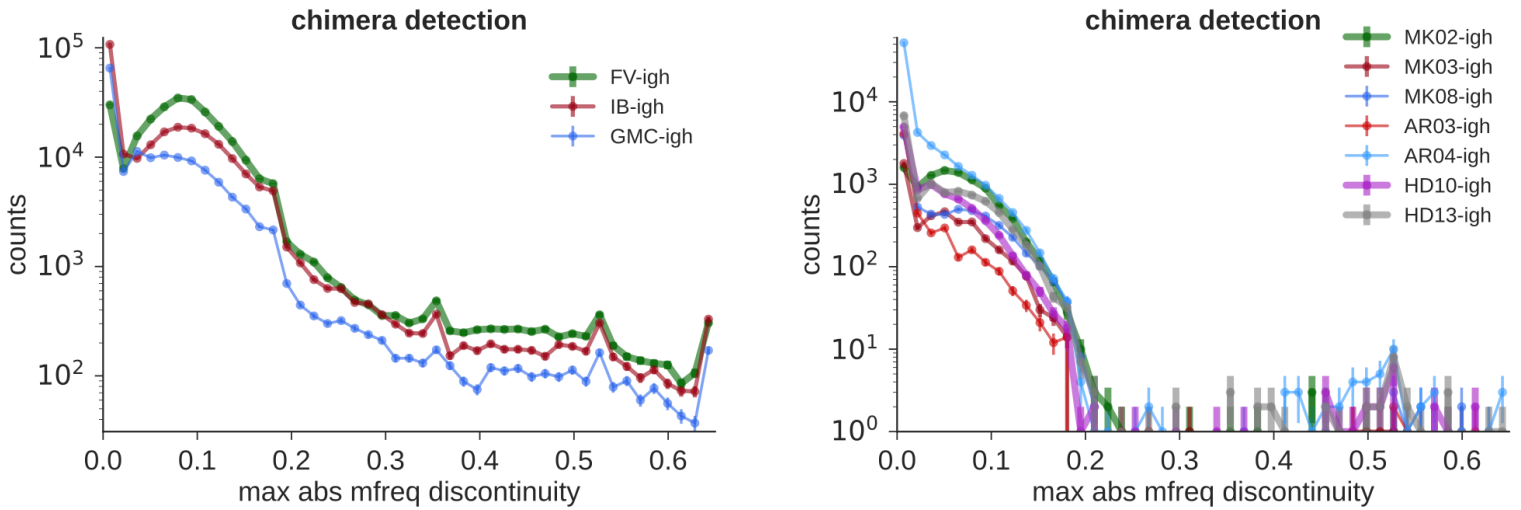




- No hope of detecting chimeras with
  - Break point within CDR3
  - Break point close to either end
  - Very similar template sequences

Distributions on data

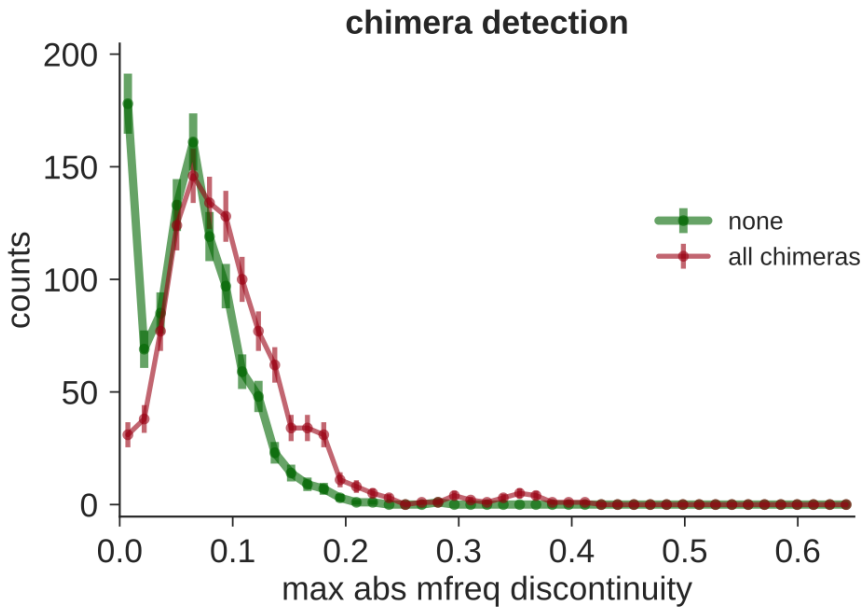
- Samples from two studies that I had lying around



- Left (3 subjects, Laserson 2014) big tail of very likely chimeras (see backup)
- Right (7 subjects, Vander Heiden 2017) almost no obvious chimeras
- Presumably reflects a difference in sequencing or preprocessing

Dual-barcode samples from Chaim

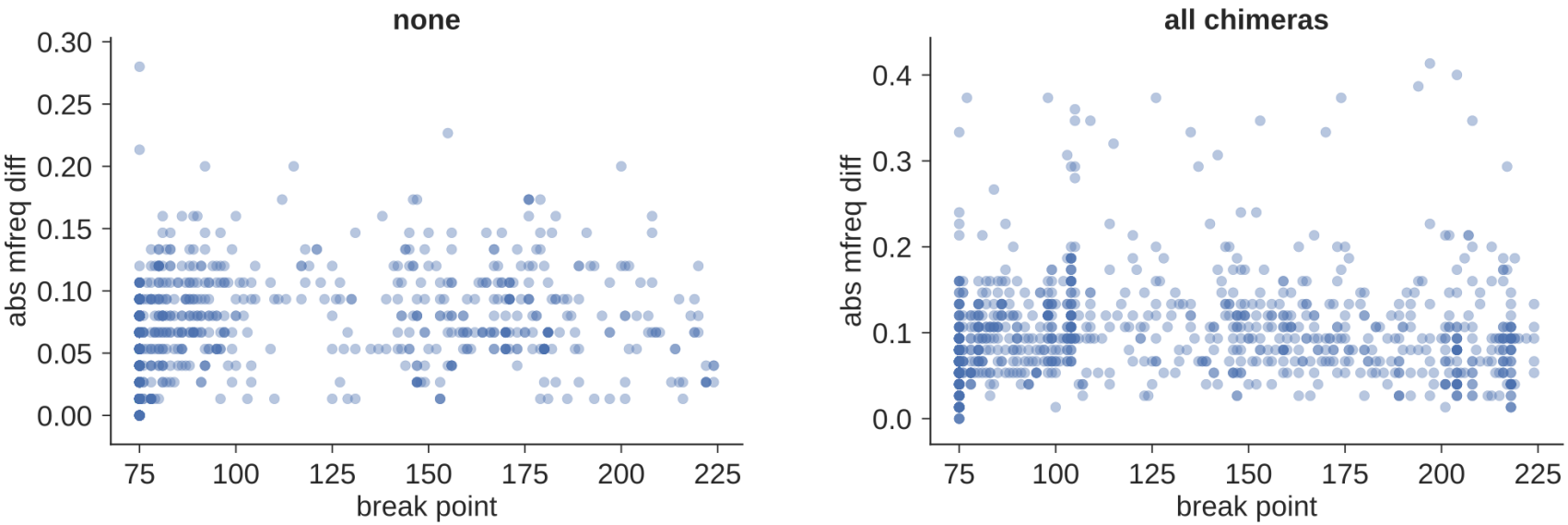
- Two 1000-sequence samples
  - one entirely chimeras
  - one with none



- Fewer obvious chimeras than in simulation, but otherwise similar

Why no obvious chimeras?

- Maybe break point selection in these samples is super biased?
  - Simulation assumed randomly chosen break point



- Probably not: likely chimeras (near top) are pretty evenly distributed across sequence
  - same holds for chunk len down to 15

Why no obvious chimeras?

- Preprocessing removed the obvious chimeras?
- Chimera creation is biased toward similar sequences?
  - it is, but unclear how much
- Maybe max abs diff distribution is super sensitive to SHM rates?
  - No: small effect on the left end of the distribution (results not shown)
- Different chunk lengths?
  - No: distributions look basically the same (tried down to 15) (results not shown)

Conclusion I

- Backing up, the dual barcode samples don't change the big picture that much
- Simulation told us that some chimeras are easy to detect, and some are impossible
  - Depending on a few knobs, the impossible-to-detect ones were maybe half of them
- Some data samples have obvious chimeras (Laserson 2014), some don't (Vander Heiden 2017)
  - Could be from difference in chimera prevalence
  - Or from difference in chimera characteristics (e.g. break point selection)
  - Or difference in preprocessing
- In samples enriched for chimeras (Chaim/Sai dual barcodes)
  - The difference is only from chimera characteristics
  - Still loads of undetectable chimeras, same as simulation

Conclusion II

- We can detect obvious chimeras
  - But a large fraction (at least half) of chimeras are undetectable
- This ability is useful for, e.g, flagging samples that are likely to have a chimera problem
- But it's useless for, e.g, saying with any certainty whether an IARC submission results from chimeras
- It's of course possible that a more clever metric would be better at identifying chimeras
  - But seems unlikely you could do a whole lot better
  - SHM causes lots of adjacent mutations, V genes are super similar, break points in CDR3 are pretty hopeless

Backup

Implementation

- In partis at the moment:

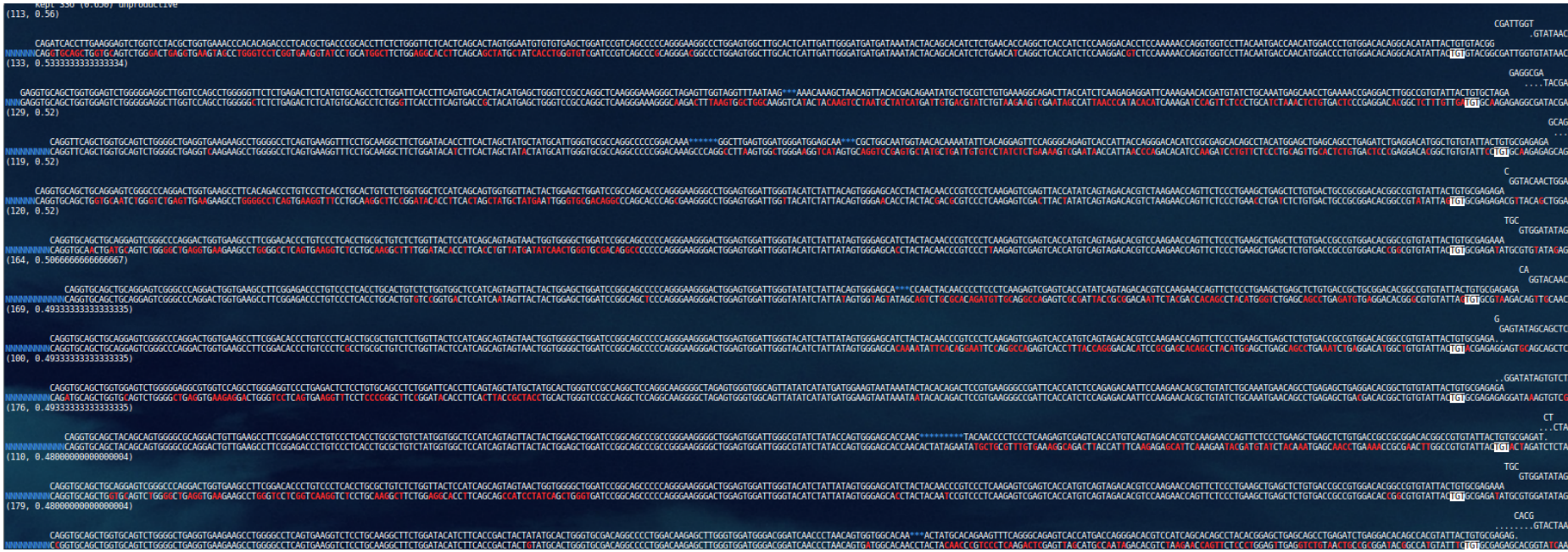
  https://github.com/psathyrella/partis/blob/dev/python/utils.py#L3422

- Simple enough may as well paste it here

```
# ----------------------------------------------------------------------------------------
def get_chimera_max_abs_diff(line, iseq, chunk_len=75, max_ambig_frac=0.1, debug=False):
    naive_seq, mature_seq = subset_iseq(line, iseq, restrict_to_region='v')  # self.info[ui

    if ambig_frac(naive_seq) > max_ambig_frac or ambig_frac(mature_seq) > max_ambig_frac:
        if debug:
            print '  too much ambig %.2f %.2f' % (ambig_frac(naive_seq), ambig_frac(mature_
        return None, 0.

    if debug:
        color_mutants(naive_seq, mature_seq, print_result=True)
        print ' '.join(['%3d' % s for s in isnps])

    _, isnps = hamming_distance(naive_seq, mature_seq, return_mutated_positions=True)

    max_abs_diff, imax = 0., None
    for ipos in range(chunk_len, len(mature_seq) - chunk_len):
        if debug:
            print ipos

        muts_before = [isn for isn in isnps if isn >= ipos - chunk_len and isn < ipos]
        muts_after = [isn for isn in isnps if isn >= ipos and isn < ipos + chunk_len]
        mfreq_before = len(muts_before) / float(chunk_len)
        mfreq_after = len(muts_after) / float(chunk_len)

        if debug:
            print '    len(%s) / %d = %.3f' % (muts_before, chunk_len, mfreq_before)
            print '    len(%s) / %d = %.3f' % (muts_after, chunk_len, mfreq_after)

        abs_diff = abs(mfreq_before - mfreq_after)
        if imax is None or abs_diff > max_abs_diff:
            max_abs_diff = abs_diff
            imax = ipos

    return imax, max_abs_diff
```

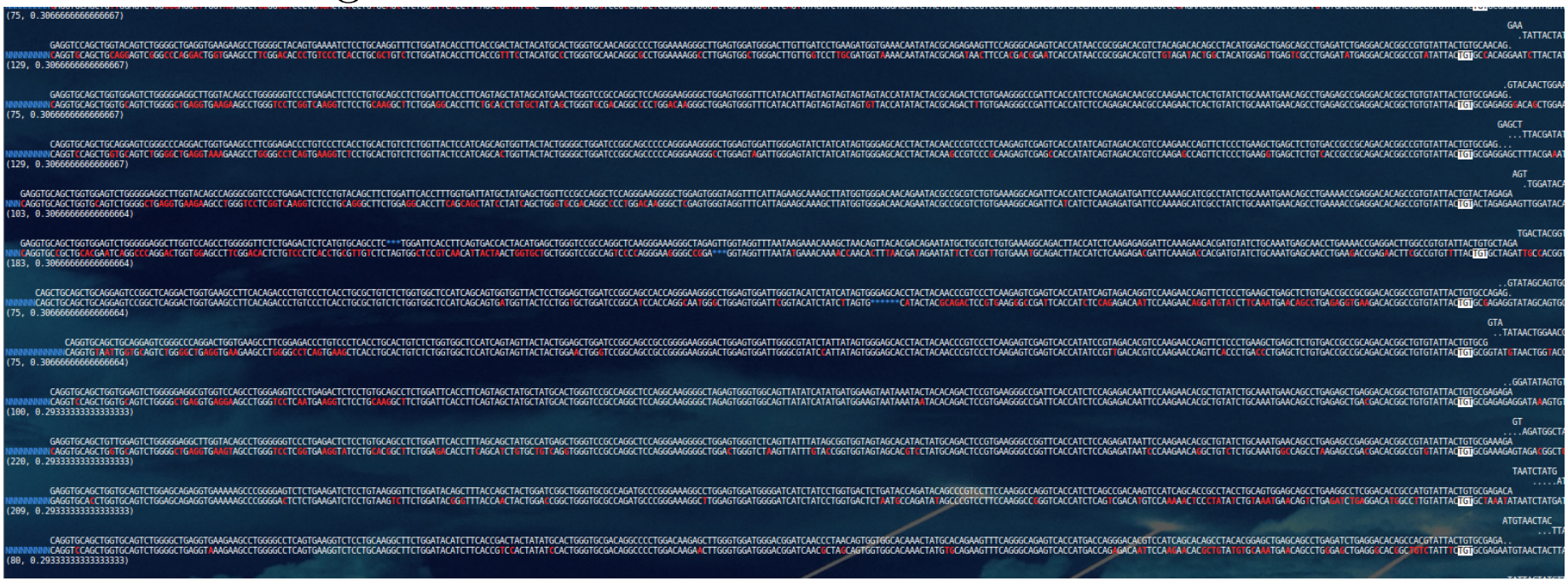
Max abs difference examples: 0.5

0.5: Clearly chimeric



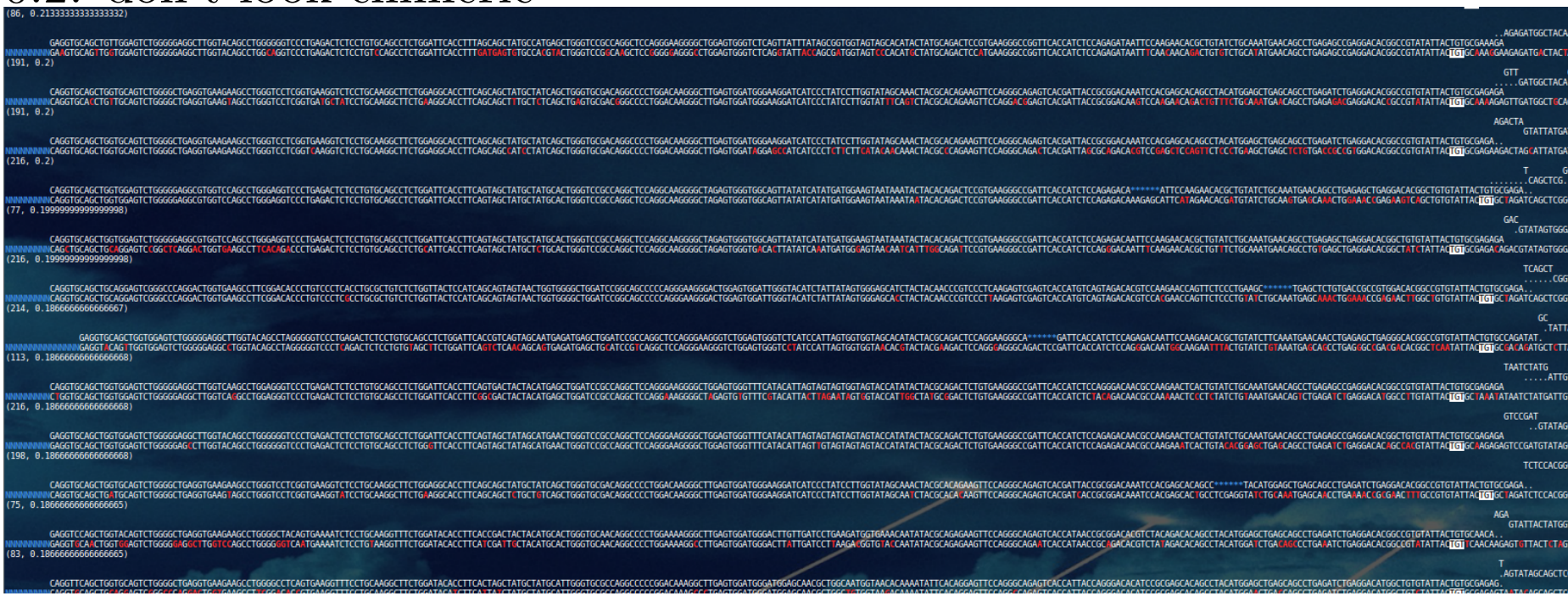
Max abs difference examples: 0.3

0.3: kinda marginal



Max abs difference examples: 0.2

0.2: don't look chimeric



High-abs diff data examples (Laserson 2014)

values around 0.8