Synthetic HTS Generation

Hackathon Plan

```python

# --- TESTING ---
# synthetic read pair:
r: AlignedSegment = pysam.AlignedSegment()
r.query_name = 'read1'
r.query_sequence = 'CTGDAAAACC' * 10
r.query_qualities = pysam.qualitystring_to_array(input_str='AAAAAAAAAA' * 10)
r.flag = 0x43
r.reference_id = 0
r.reference_start = 100
r.next_reference_start = 100
r.mapping_quality = 20
r.cigarstring = '100M'
r.set_tag(tag='MC', value='100M')

```

Why

- Creating test data for algorithm development and maintenance, or exploratory research, is long winded
- Though most bioinformatics data are ultimately primitives, they have unique constraints
- While some options are available for generating empirically modelled sequence data, they have no capacity for more artificial constraints, e.g. “give me 100 valid reads of this reference **which contain 1 or more 3bp deletions**”

Current landscape

- `faker-biology` can generate semi-random nucleotide strings, but no parameterisation
- `wgsim`, `pbsim`, `ART` can generate entire alignments for specific sequencing types
- `NEAT` can learn and generate simulated sequence data via a CLI, including a mutational profile

But no tool provides scalable, constrained, on-the-fly HTS data generation

Hackathon Output

- `Core constrained synthetic read generators` – composable units up to a read or fragment, allowing for edit injection either onto reference pre-simulation or reads post-simulation
- `Python interface` – to generator/s and compositions thereof
- (stretch) `Prototype CLI` – to output data with a structure defined by an input schema and config

Fast, loose, synthetic data