Appendix A: Python code for change point analysis

```
import matplotlib.pyplot as plt
import numpy as np
import numba as nb
import pandas as pd


@nb.njit
def sample_idx(weights):
    """ Sample an index with probability proportional to its weight. """
    weights = weights / np.sum(weights)
    u = np.random.uniform(0, 1)
    partial_sum = weights[0]
    idx = 0
    while u > partial_sum:
        idx += 1
        partial_sum += weights[idx]
    return idx


@nb.jit(fastmath=True)
def binomial_changepoint_model_samples(
    x1: np.ndarray,
    x2: np.ndarray,
    init_t1: np.float64,
    init_t2: np.float64,
    init_k: np.int64,
    n_iters: np.int64 = 100_000,
    seed: np.int64 = 0,
) -> np.ndarray:
    """
    Run the Gibbs sampler for the Binomial changepoint model.

    This function will run a Gibbs sampler for the binomial changepoint model where the
    number of occurrences of one variant over time is given by `x1` and the number of occurrences
    of the alternative variant over the same time frame is given by `x2`.

    The Binomial changepoint model assumes that X1 ~ Binom(n, t1) before the changepoint. That is,
    for each period of time <= k. Here, n is the total number of occurences of either variant
    (i.e. n = X1 + X2), X1 is the number of occurrences of one variant (and, therefore, X2
    is the number of occurrences of the alternative variant), and k is the index at which the
    change occurred.

    The Binomial changepoint model assumes that X2 ~ Binom(n, t2) after the changepoint. That is,
    for each period of time > k.

    Parameters
    ----------
    x1: np.ndarray
        An array containing, over a given time frame, the number of occurrences of one variant.
        E.g. x1[0] is the number of occurrences of near in 1771, x1[1] is the number of
        occurrences of near in 1772, and so on.
    x2: np.ndarray
        An array containing, over a given time frame, the number of occurrences of the variant
        that is alternative to the variant which is tracked in `x1`. E.g. x2[0] is the number of
        occurrences of nearly in 1771, x2[1] is the number of occurrences of nearly in 1772, and
```

```
    so on.
  init_t1: np.float64
    An initial value for the Binomial probability parameter before the changepoint. That is,
    a value for `t1` at which to start the Gibbs sampler.
  init_t2: np.float64
    An initial value for the Binomial probability parameter after the changepoint. That is,
    a value for `t2` at which to start the Gibbs sampler.
  init_k: np.int64
    An initial value for the index at which the change occurred.
  n_iters: np.int64
    The number of iterations for which to run the Gibbs sampler.
  seed: np.int64
    The seed for the random number generator. Ensures reproducibility.

  Returns
  -------
  t1_out: np.ndarray
    The sampled values for t1.
  t2_out: np.ndarray
    The sampled values for t2.
  k_out: np.ndarray
    The sampled values for k.
  """
  n_periods = len(x1)
  total_occ = x1 + x2
  t1_out = np.empty(n_iters + 1, dtype=np.float64)
  t2_out = np.empty(n_iters + 1, dtype=np.float64)
  k_out = np.empty(n_iters + 1, dtype=np.int64)
  ws = np.empty(n_periods, dtype=np.float64)
  t1_out[0], t2_out[0], k_out[0] = init_t1, init_t2, init_k
  np.random.seed(seed)
  for i in range(n_iters):
    t1, t2, k = t1_out[i], t2_out[i], k_out[i]
    sum_x_before = np.sum(x1[:k + 1])
    sum_occ_before = np.sum(total_occ[:k + 1])
    sum_all_x = np.sum(x1)
    sum_all_occ = np.sum(total_occ)
    t1 = np.random.beta(sum_x_before + 1.0, sum_occ_before - sum_x_before + 1.0)
    t2 = np.random.beta(
      sum_all_x - sum_x_before + 1.0,
      (sum_all_occ - sum_occ_before) - (sum_all_x - sum_x_before) + 1.0
    )
    for j in range(n_periods):
      sum_x_before = np.sum(x1[:j + 1])
      sum_occ_before = np.sum(total_occ[:j + 1])
      log_weight = (
        sum_x_before * (np.log(t1) - np.log(t2)) +
        (sum_occ_before - sum_x_before) * (np.log(1.0 - t1) - np.log(1.0 - t2))
      )
      ws[j] = np.exp(log_weight)
    k = sample_idx(ws)
    t1_out[i + 1], t2_out[i + 1], k_out[i + 1] = t1, t2, k
  return t1_out, t2_out, k_out


data = pd.read_csv('./data.csv')
data['total'] = data['abortion'] + data['miscarriage']
```

```
data['proportion'] = data['abortion'] / data['total']


x1 = data['abortion'].values
x2 = data['miscarriage'].values
init_k = 10
init_date = 1985
before_mask = (data['year'] <= init_date)
after_mask = (data['year'] > init_date)
init_t1 = data.loc[before_mask, 'proportion'].mean()
init_t2 = data.loc[after_mask, 'proportion'].mean()


out = binomial_changepoint_model_samples(x1, x2, init_t1, init_t2, init_k, 100_000, 12345)


years = data['year'].values
idxs, counts = np.unique(out[-1], return_counts=True)


plt.bar(years[idxs], counts / counts.sum())
plt.hist(out[0], bins=500)
plt.hist(out[1], bins=500)
```