# Homework 3

**Instructions.** This homework is due at the beginning of lab, Wednesday, September 20th. You can edit this file and copy/paste into it by opening it in Libreoffice on Linux, PDFfiller Google App, Acrobat, Microsoft Word and others. If you use the internet for information, please assess the verity of the source and document it as a source. Please turn in a pdf version of the written solutions, as well as a zip file of all code you use to solve the problems. Please name each piece of code by the question it is intended to answer or otherwise clearly indicate to me how to run the code to get each answer.

1. In the following, suppose you have read data from a Next Generation Sequencing (NGS) experiment sequencing your own DNA delivered in a Fasta file. Learn more about one of the technologies that can do generate such data under Large Genome Sequencing at the Illumina Whole Genome Sequencing website.

   (a) **T** (True/False) The proportion of nucleotide A in a read is a statistic.

   **Solution:** A read is the sequence of nucleotides produced when some sequencing technology attempts to sequentially "read" the nucleotides along a DNA fragment (from your genome, in this case). Reads are discussed, for example, on the Wikipedia entry DNA sequencing. Since the proportion of A in the read is computed from the data, in this case, a read, and requires no additional information, it is a *statistic*.

   (b) **F** (True/False) The proportion of nucleotide A in your *FOXP2* gene is a statistic.

   **Solution:** Your *FOXP2* gene is a contiguous set of nucleotides in your genome, but because you do not observe your genome directly (only the error-containing reads), this sequence, and hence the proportion of A in it, is not a statistic. It is an unknown population parameter, and though you can *estimate* it from reads of the *FOXP2* gene, you cannot know *for sure* what it is.

   (c) If you perform statistical inference using these data, what is the population you can extend your conclusions to? **all fragments of your genome: together, your genome**

   (d) One kind of inference that might be of interest is to determine whether you are a mutant. Suppose there are $n = 17$ reads covering a particular site in your genome that is known to be mutated from C to A in some individuals. If you observe $X = 4$ A nucleotides and $n - X = 13$ C at this site, can you assume you are mutant? Keep in mind that NGS data contain errors and you are diploid. (Also assume you are "mutant" as soon as you have at least one copy of the mutant nucleotide.) Please show all your work, including your choice of test statistic, your specific null hypothesis $H_0$, including any assumptions it makes, and the calculation of a $p$-value to support your conclusion.

**Solution:** Diploids are one of three possible genotypes at this locus: AA (homozygous mutant), AC (heterozygous mutant), or CC (normal). I allowed you to ignore the possibility you are AA.

**Vague null and test statistic.** A useful vague null hypothesis is "I am a heterozygous mutant." If I am a heterozygous mutant, then I expect half the reads to have an A and half to have a C *if there are no errors.* If I am not a mutant, then I expect no reads to have A. If I am a homozygous mutant, then I expect all reads to have A. Even if there are rare errors, I can still expect that small numbers of A-containing reads indicate for the conclusion that I am not a mutant, while large numbers of A-containing reads indicate for the conclusion that I am a homozygous mutant. Overall, the count of A-containing reads is a test statistic that is sensitive to the truth of my vague null.

**Specific null.** Let $X_i$ indicate if the $i$th read (of $n$) covering the site is A. If reads are independent, then $X = \sum_{i=1}^{n} X_i$ is a Binomial random variable with $n$ trials and probability of success $p = 0.5$ under the now specific null hypothesis

$$H_0 : X \sim \text{Bin}(n, 0.5).$$

**Data and $p$-value.** I observe $x = 4$ when $n = 17$. If I can assume that I am *not* a homozygous mutant, then the $p$-value is

$$P(X \leq 4) = 0.0245209.$$

This is a small number, so I can fairly safe conclude I am not a mutant. The chance I am wrong in this conclusion is less than 3%.

**The rest of this answer.** This question is challenging, because I did not explicitly provide the test statistic, nor the vague, nor the specific null hypothesis. That's tough, but solving open-ended problems like this is exactly how you make yourself valuable to employers. (In this case, imagine a sequencing company that wants to diagnose genetic mutations using NGS technology.) I demonstrate several different model building processes below, including extensions or details for the above answer. Several of you tried $z$-, $t$-tests, or otherwise forced a normal distribution on the test statistic. All of these are poor models for this problem, but if you implemented it correctly, you could earn points. You may be asked to suggest or defend test statistics $T$ or null hypotheses $H_0$ on exams in simple situations.

**Allowing errors.** The above null hypothesis assumes, vaguely, that errors are rare. This is an imprecision in the specific null hypothesis. What it really assumes is that there is no bias in the error, such that neither A nor C gains by the error process. To specifically accommodate the reality of errors, we could hypothesize a specific $H_0$ where all three possible errors occur with probability $p_e/3$, and the nucleotide counts are distributed as a Multinomial with $n = 17$ trials and probabilities $(0.5 - p_e/3, 0.5 - p_e/3, p_e/3, p_e/3)$. Many bioinformaticians have made exactly this assumption for NGS. Then, we have an unknown model parameter $p_e$. You can estimate it from the data, but I cannot assume you have the tools to do the estimation: if you got this far, you could legitimately ask how to do the estimation. (It turns out the estimate is $\hat{p}_e = 0$, which gets we back to the same test.)

**Allowing homozygous mutants.** If I might also be a homozygous mutant, then the $p$-value is the above *plus* the sum of the probability of all large A counts that are as extreme or more extreme than the observed $x = 4$ in the right tail. How extreme is the observed outcome? It has probability $P(X = 4) = 0.018158$. Thus, any outcome $y$ in the sample space $\{0, 1, \ldots, 17\}$ for which

$$P(X = y) \leq P(X = 4)$$

is "as or more extreme." We can sum up the relevant probabilities with the Python code:

```
import scipy.stats as stats

x = 4
p_success = 0.5
n = 17
p_value = stats.binom.cdf(x, n=n, p=p_success)
for y in range(x+1, n+1):
if stats.binom.pmf(x, n=n, p=p_success) >= \
stats.binom.pmf(y, n=n, p=p_success):
p_value += stats.binom.pmf(y, n=n, p=p_success)
print "P-value", p_value
```

The answer is

```
## [1] "P-value 0.0490417480469"
```

I still conclude with reasonable confidence that I am not a heterozygous mutant. My chance of a mistake in making this conclusion is less than 5%. Formally, I have not excluded the possibility that I am a homozygous mutant, yet the data are highly suggestive that I am a non-mutant.

***Need to know:*** *There is an important lesson here. It is not always possible to clearly delineated which values of $T$ are "as extreme or more than the observed test statistic," even when we know the $T$ is sensitive to the truth of $H_0$. One way to identify which $T$ are "as extreme or more than the observed test statistic" is to compare the probability of the observed test statistic $t_0$, i.e. $P(T = t_0)$, to the* *probability of any other value $t$, assuming $H_0$ is true. If $P(T = t_0) \geq P(T = t)$, then $t$ is among those test statistics that are "as extreme or more extreme than $t_0$". When you are computing the tail area in the $z$- or $t$-test, you are integrating over all values of $z$ or $t$ that have sampling density less than that of the observed value.*

**Starting with "I am not mutant" as the vague null.** We may have chosen "I am not a mutant" as our vague null. If our genome were observed without error, then all reads would contain C and the presence of even one A definitively proves we *are* a mutant, $p$-value 0. The reality is that NGS reads are error-prone, so it is possible to be nonmutant and still have A's in our reads. How do we perform inference in this situation?

If we modify our null hypothesis to be "I am not a mutant, so all observed non-C's are produced by error," then we need to build a model for the error process. As suggested above, we may reasonably assume that error C $\rightarrow$ A occurs independently in each read with the same error probability $p_e$, so the specific null hypothesis is

$$H_{0a} : X \sim \text{Bin}(17, p_e), 0 \leq p_e \leq 1.$$

As before, we need to estimate $p_e$. In this case, we would estimate

$$\hat{p}_e = \frac{4}{17} = 0.2352941,$$

and the $p$-value is

$$P(X \geq 4) = 0.593064.$$

We would be foolish to reject $H_0$ because we have better than even odds of being wrong!

However, we would also have failed to reject $H_0$ if $X = 15$, when common sense suggests we really should reject $H_0$ then. But what is this common sense we are using? We have adopted null $H_{0a}$, and we assume that large $X$ indicate against $H_{0a}$, but it is no longer true. You are using information–errors are rare–that is not actually encoded in our null hypothesis, and *this information* is behind our "common sense." If we did encode it, such as requiring a small error probability $p_e$, as in

$$H_{0b} : X \sim \text{Bin}(17, p_e), p_e \leq 0.01,$$

then our estimate would change to

$$\hat{p}_e = 0.01,$$

since this is the closest of the possible values to $\frac{4}{17}$. Now, $X \sim \text{Bin}(17, 0.01)$, and the $p$-value is

$$P(X \geq 4) = 2.1444754 \times 10^{-5},$$

resounding evidence against not being a mutant.

You may also introduce "common sense" by assuming all errors are equally likely and considering a richer output, the counts of all nucleotides at the site: $(X_{\text{A}}, X_{\text{C}}, X_{\text{G}}, X_{\text{T}})$. Now, the null is

$$H_{0c} : (X_{\text{A}}, X_{\text{C}}, X_{\text{G}}, X_{\text{T}}) \sim \text{Multinomial}(p_e/3, 1 - p_e, p_e/3, p_e/3),$$

with $p_e$ to be estimated from the data. The estimate is still $\hat{p}_e = 4/17$. This estimation is beyond our pay grade too, but there is an interesting simulation ahead that we *are* capable of doing, so I continue. If we use $T = \max\{X_{\text{A}}, X_{\text{G}}, X_{\text{T}}\}$ as our test statistic, it should get larger as either (1) mutations are not all equally likely or (2) we actually have the mutation. There is no theoretical sampling distribution for this statistic, but we can simulate it. I'll use `R` because I can do it inline, but I expect you to be able to do it in `Python`.

```
B <- 1000   # number of replicates
p.e <- 4/17 # estimate
X <- rmultinom(B, size=17, prob=c(p.e/3, 1-p.e, p.e/3, p.e/3))
X <- X[-2,] # remove count of C
Y <- apply(X, 2, max) # find max of others
p.value <- mean(Y >= 4) # compute p-value as fraction w/ higher max
```

The $p$-value is 0.124, which is suggestive, but not really convincing evidence against our assumption that we are not a mutant, and it could also just indicate that our equal error rate assumption is wrong.

**Final conclusion.** It is not possible to reject the assumption that we are not a mutant, *unless* we get or use extra information about $p_e$ and the error process. It appears relatively easier to prove we are not a heterozygous mutant, but only because the usual assumptions made about errors still leave the 50/50 split between A and C intact. Overall, for these data and typical assumptions about errors (they are rare and not biased toward any nucleotide), the most likely explanation is that we are, in fact, not a mutant. However, there does appear to be something fishy about the data. Why so many A errors, but not others? Maybe we are not a diploid at the locus! That is another kind of mutation: duplication.

***Need to know.*** *The overall lesson is that our assumptions about the data generation process matter a lot. We need to justify our assumptions, and we cannot do that unless we become an expert in the applied field, consult with someone who is, or collect data to become an expert when no one yet is.*

2. In this question we will identify the nucleotide substitution that is occurring in the nonpermissive cells from the virus experiment. Suppose $\mathrm{N}_t$ is the target nucleotide and it is mutated to $\mathrm{N}_m \neq \mathrm{N}_t$, both in $\{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$. Your goal is, therefore, to infer $\mathrm{N}_t$ and $\mathrm{N}_m$.

   (a) For your **specific null hypothesis** $H_0$, you assume there is *no* mutation occurring, that therefore the virus genomes from permissive and nonpermissive cells are identically distributed, that the sequenced 500 bp fragments are independent and uniformly (every possible starting position is equally likely) drawn from the larger virus genome, and that the nucleotides in the virus genome (and by implication each fragment) are iid Multinoulli($p_{\mathrm{A}}, p_{\mathrm{C}}, p_{\mathrm{G}}, p_{\mathrm{T}}$), where $p_{\mathrm{N}}$ is the probability of nucleotide N. Raise two biological criticisms of this model, identifying how it may fail to mimic real biological data, other than the interesting possibility that mutations may be *actually* happening.

**Solution:**

1. As discussed in class, it is not reasonable to assume that the nucleotides are independent because they encode for proteins and other biological functions that rely on the concurrent signal from multiple nucleotides.

   **More.** For example, the amino acids in proteins are encoded with triples of nucleotides. In particular, you now know that ATG (in the DNA) encodes for Met and the first amino acid in a protein. This induces a dependence, because if you have drawn A, then T, biology is going to favor G at the next position in order to insure the genome carries out its function. The preference for G, which you can communicate with probabilities as

$$P(X_3 = \mathrm{G} \mid X_1 = \mathrm{A}, X_2 = \mathrm{T}) > P(X_3 = \mathrm{G}),$$

   for three contiguous nucleotides $X_1, X_2, X_3$, clearly contradicts independence, since (mutual) independence implies, among other things, that

$$P(X_3 = \mathrm{G} \mid X_1 = \mathrm{A}, X_2 = \mathrm{T}) = P(X_3 = \mathrm{G}).$$

2. It is also not reasonable to assume the sites are "identically distributed." If you could align sequences from a virus genome, you will see that they mostly agree (have the same nucleotide) at the same site. In the mutant example, question 2, all normal humans share a C at the site in question. Therefore, the random variable present at that site in a randomly sampled individual may be more like Multinoulli($0.002, 0.997, 0.0005, 0.0005$), which will not be the same as a model for a site where we all tend to be A!

**Others.** You could in fact go through each of the assumptions listed in the specific $H_0$ and critique each one. The two listed above are the easiest critiques.

- **No mutation.** Even if there is no mutation caused by a novel process, mutations do occur as a natural part of the virus life cycle. The permissive and nonpermissive cells obviously differ somehow, but if this difference exerts a selective pressure on the virus, the result could be that certain mutants are *selected* in the nonpermissive cells. The pattern may look just like an active mutation process operating in the nonpermissive cells. We cannot eliminate this possibility through statistical methods.

- **Virus genomes are identically distributed.** This is the assumption violated in the above selection example, but if it is violated for any reason, it will produce fragments that are also not identically distributed, which is discussed below.

- **Fragments are independent.** If the viruses and cells did not interact in the petri dish, this assumption would be true, but plaques form when a single virus infects one cell, replicates, and then infects all the cells around it. Thus, all genomes from cells in or near this plaque are likely descendents of a single virus: they are not independent. If the first virus is partially mutated, but still functional, then all descendents will share the same mutations. We will count $m$ (the number of descendents) A $\rightarrow$ C mutations when there was actually only one. Our confidence in the conclusion that mutation A $\rightarrow$ C is happening will be falsely inflated. This serious problem is solved by consulting with the biologist and making sure only one round of infection is allowed.

- **Fragments are identically distributed.** If there is inhomogeneity of nucleotide content in the genome and the genome is not uniformly sampled (next assumption), then the fragments will not be identically distributed. This reality is not a serious problem if the permissive and nonpermissive samples are not differentially affected by this assumption violation.

- **The genome is uniformly sampled.** This assumption is demonstratably not true. Fragments are often caused by shearing the DNA, and there are preferential shear sites. This problem can be difficult to correct, but assuming it affects permissive and nonpermissive cells the same way, there should be little effect. If the shear sites change because of mutation and there is inhomogeneity of nucleotide distribution across sites, then there will be false signal. This is very hard to overcome and most bioinformaticians would simply ignore it at their peril.

Though our simulator does not replicate the complexity admitted here, if the assumption violations are identical in the permissive and nonpermissive samples, then estimating the $p$-value by simulation will probably be robust to these assumption violations. The most alarming possibility is that the assumption violations differentially impact the permissive and nonpermissive samples. We should discuss this possibility with the biologist.

(b) Let $n_{n\mathrm{N}}$ be the total number of $\mathrm{N} \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ nucleotides in the fragments obtained from nonpermissive cells and $n_{p\mathrm{N}}$ the corresponding count for permissive cells. Let $n_n$ and $n_p$ be the total number of nucleotides in the reads from nonpermissive and permissive cells, respectively. The test statistic you will be using in this problem is

$$T = \sum_{\mathrm{N}\in\{\mathrm{A,C,G,T}\}} \left( \frac{n_{n\mathrm{N}}}{n_n} - \frac{n_{p\mathrm{N}}}{n_p} \right)^2 .$$

i. What numeric value will this test statistic be close to if $H_0$ is true? Why?

**Solution:** Because the specific $H_0$ in 2(a) assumes there is no change in the viral genomic content being sequenced in permissive and nonpermissive cells, specifically $p_{n\mathrm{N}} = p_{p\mathrm{N}} = p_{\mathrm{N}}$ for all $\mathrm{N} \in \mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}$, we expect $\frac{n_{n\mathrm{N}}}{n_n}$ to be similar to $\frac{n_{p\mathrm{N}}}{n_p}$ because $\frac{n_{n\mathrm{N}}}{n_n} \approx p_{n\mathrm{N}}$ and $\frac{n_{p\mathrm{N}}}{n_p} \approx p_{n\mathrm{N}}$. Hence $T \approx 0$ under the specific $H_0$.
Many of you stated that the counts $n_{n\mathrm{N}} \approx n_{p\mathrm{N}}$ should be similar, but there is nothing in the specific $H_0$ that leads to this conclusion, even though it is *approximately* true in your actual data. Look to your specific $H_0$, *not your data*, to answer this question.

ii. How will the value of the test statistic change if $H_0$ is not true?

**Solution:** If $H_0$ is not true because the nonpermissive cells are mutating C to A, for example, then we expect $\frac{n_{n\mathrm{A}}}{n_n}$ to be substantially larger than $\frac{n_{p\mathrm{A}}}{n_p}$. There will be a corresponding, negative shift in the proportion of C. The end result is that the summand will be substantially non-zero for $\mathrm{N}_t$ and $\mathrm{N}_m$, and $T$ will increase in value.

**More.** It turns out that if there is dependence among the nucleotides, $T$ will still remain close to 0. (Try and come up with a counterexample. For example, consider complete dependence, where every fragment is made up of completely identical nucleotides, but the reads are still independent and draw the first nucleotide independently.)
So, this test statistic is robust to violations of independent nucleotides, but it is still sensitive to assumption violations that differentially affect

permissive and nonpermissive samples.

iii. Compute and report the observed value $t_0$ of the test statistic. What set of values of random variable $T$ are as or more extreme than $t_0$? (Hint: Write a function to compute $t_0$ from sequence or count data for later reuse.)

**Solution:** I provide the following `Python` code to accomplish the task. I learned a few things about `Python`. For example, to finish the next Part c, I wanted to be able to access the functions I wrote for this part without re-executing the code to compute $t_0$. I found help on stackoverflow. I assumed all nucleotide sequences are upper case, which may have been true in the files, but to be sure, I used the string method `upper()`. Otherwise, I used what I learned from the `Python` unit.

```
import sys

##
# Read fasta file.
#
# @param filename       name of fasta file
# @return               fasta dictionary (names : sequences)
def read_fasta( filename ):
        file = open(filename)
        sequence = ""
        fasta_dict = {}
        for line in file:
                if line.startswith(">"):
                        line = line.strip(">").strip()
                        if len(sequence) > 0:   # store previous sequence
                                fasta_dict[name] = sequence.upper()
                        name = line
                        sequence = ""
                else:
                        sequence += line.strip()
        file.close()
        if len(sequence) > 0:   # store last sequence
                fasta_dict[name] = sequence.upper()
        return fasta_dict

##
# Count nucleotides in a fasta dictionary (names : sequences).
#
# @param fasta_dict     the fasta dictionary
# @return               vector of counts in order A, C, G, T
def count_nucs(fasta_dict):
        nuc_cnts = [0] * 4
        for n, s in fasta_dict.iteritems():
                nuc_cnts[0] += s.count('A')
                nuc_cnts[1] += s.count('C')
                nuc_cnts[2] += s.count('G')
                nuc_cnts[3] += s.count('T')
        return nuc_cnts

def compute_T(cntA, cntB):
        nA = sum(cntA)
        nB = sum(cntB)
        t_0 = 0
        for i in range(len(cntA)):
                t_0 += (cntA[i]/float(nA) -
                        cntB[i]/float(nB))**2
        return t_0

script_name = sys.argv[0]

if len(sys.argv) != 3:
        print "Usage: ", script_name, "permissive_filename",\
                "nonpermissive_filename"
        sys.exit()

perm_filename = sys.argv[1]
nonperm_filename = sys.argv[2]

perm_fasta = read_fasta(perm_filename)
nonperm_fasta = read_fasta(nonperm_filename)
perm_cnts = count_nucs(perm_fasta)
nonperm_cnts = count_nucs(nonperm_fasta)

##
# Wrap rest of content in main() so that importing this file to use
# the functions does not execute this code!
def main():
        t_0 = compute_T(perm_cnts, nonperm_cnts)
        print t_0

if __name__ == "__main__":
        main()
```

The output is:

```
## 0.00406990314928
```

Evidence against the specific null hypothesis $H_0$ is provided by values of $T \geq 0.00406990314928$.

(c) Propose estimates $\hat{p}_{\mathrm{A}}, \hat{p}_{\mathrm{C}}, \hat{p}_{\mathrm{G}}$, and $\hat{p}_{\mathrm{T}}$ of the model parameters $p_{\mathrm{A}}, p_{\mathrm{C}}, p_{\mathrm{G}}$, and $p_{\mathrm{T}}$ under the model assumptions stated in Part (a).

**Solution:** If $H_0$ is true, then every nucleotide observed in the permissive and nonpermissive cells come from the same Multinoulli distribution. We can estimate $p_{\mathrm{N}}$ by counting the number of occurrences of N in both datasets, and dividing by the total number of nucleotides observed across both datasets. Fortunately, we have already computed the numbers we need to do that. The following code will use those functions and compute the estimates. To complete this code, I decided I wanted to output the numbers with only three decimal places (so they would fit on the page). I accomplished that with formatted output: under "The Pythonic Way". I also learned the "comma trick" to avoid having `print` output a newline.

```
import sys

# import some functions we already wrote
from hw3q2biii import read_fasta, count_nucs, perm_cnts, nonperm_cnts

print perm_cnts
print nonperm_cnts

perm_n = sum(perm_cnts)
nonperm_n = sum(nonperm_cnts)
print perm_n, nonperm_n
p_estimate = [0] * 4
for i in range(4):
        p_estimate[i] = (perm_cnts[i] + nonperm_cnts[i]) /\
                float(perm_n + nonperm_n)

for p in p_estimate:
        # formatted output to display only 3 decimal places
        print "{0:.3f}".format(p),      # trailing comma trick
print
```

```
## 0.388 0.172 0.217 0.223
```

(d) Use Monte Carlo simulation to estimate the $p$-value.

**Solution:** For this solution, I needed a way to sample A, C, G, and T with the probabilities estimated in Part c. I searched and found `numpy.random.choice` for this purpose.

```
import numpy as np
from hw3q2c import *
from hw3q2biii import compute_T
```

```
def simulate_fasta(fasta_dict, nuc_probs):
        sim_fasta_dict = {}
        for n, s in fasta_dict.iteritems():
                seq_len = len(s)
                nucleotides = np.random.choice(['A','C','G','T'],
                        size=seq_len, p=nuc_probs)
                sim_fasta_dict[n] = "".join(nucleotides)
        return sim_fasta_dict


def main():
        # using data loaded from hw3q2c
        t_0 = compute_T(perm_cnts, nonperm_cnts)

        B = 1000
        p_value = 0

        for b in range(B):
                perm_sim_fasta = simulate_fasta(perm_fasta,\
                        p_estimate)
                nonperm_sim_fasta = simulate_fasta(nonperm_fasta,\
                        p_estimate)
                perm_sim_cnt = count_nucs(perm_sim_fasta)
                nonperm_sim_cnt = count_nucs(nonperm_sim_fasta)
                perm_sim_n = sum(perm_sim_cnt)
                nonperm_sim_n = sum(nonperm_sim_cnt)
                T_sim = compute_T(perm_sim_cnt, nonperm_sim_cnt)
                if (T_sim > t_0):       # large values disprove H0
                        p_value += 1

        print "Test statistic: t0 =", t_0
        print "P-value is estimated as", p_value/float(B), "(",\
                p_value, "times out of", B, ")"
        print "Detect the mutation: nonpermissive - permissive counts"
        for i in range(len(perm_cnts)):
                print "{0:.3f}".format(nonperm_cnts[i] - perm_cnts[i]),

if __name__ == "__main__":
        main()
```

The output, without the repeated output from Part c, is

```
## Test statistic: t0 = 0.00406990314928
## P-value is estimated as 0.0 ( 0 times out of 1000 )
## Detect the mutation: nonpermissive - permissive counts
## 483.000 -129.000 -563.000 15.000
```

We conclude with $p$-value estimated to be less than 0.001 (not 0: can you see why?) that mutation is happening. Studying the count differences, we conclude that most likely G is getting mutated to A. We have just discovered how APOBEC3G does its damage.

**More.** This code is by no means the most efficient. In particular, sums of 500 Multinoulli$(\hat{p}_{\mathrm{A}}, \hat{p}_{\mathrm{C}}, \hat{p}_{\mathrm{G}}, \hat{p}_{\mathrm{T}})$ random vectors follow the Multinomial$[500, (\hat{p}_{\mathrm{A}}, \hat{p}_{\mathrm{C}}, \hat{p}_{\mathrm{G}}, \hat{p}_{\mathrm{T}})$ distribution, which we could use instead of `numpy.random.choice`, although I'm not sure it would be more efficient. If we recognize this fact and remember the Central Limit Theorem, we can use statistical theory to motivate a slight variation on test statistic $T$ whose $p$-value can be computed using a chi-squared sampling distribution, which would avoid a lot of wasted calculations. That's the benefit of studying statistics or consulting with a statistician, but the bigger skill is understanding and accounting for randomness in samples while making inferences.

3. In this question, you will identify whether there are any dinucleotide motifs (a dinucleotide is any pair of contiguous nucleotides, such as AC) that are specifically targeted by the mutation mechanism. For example, if C $\to$ A is targeted in the AC dinucleotide context, then mutations will occur specifically as AC $\to$ AA. All but one part is bonus because although many of you, especially those who have come through Stat 330 (or Stat 341), should know how to do the involved calculations, I cannot assume you can.

   (a) **Bonus.** Show mathematically that if nucleotide sites are independent and mutation $\mathrm{N}_t \to \mathrm{N}_m$ independently strikes $\mathrm{N}_t$ nucleotides with probability $p$, then the proportion of $\mathrm{N}_t\mathrm{N}_t$ dinucleotides will be altered in the mutated genomes relative to nonmutated genomes. Thus, you cannot check for changes in dinucleotide proportions to determine there is a dinucleotide motif.

**Solution:** Let $p_{n\mathrm{N}}$ and $p_{p\mathrm{N}}$ be the true proportion of nucleotide N in nonpermissive and permissive cells, respectively. Let $p_{n\mathrm{N}_1\mathrm{N}_2}$ and $p_{p\mathrm{N}_1\mathrm{N}_2}$ be the true proportion of dinucleotide $\mathrm{N}_1\mathrm{N}_2$ in nonpermissive and permissive cells, respectively.

$$\begin{aligned}
p_{n\mathrm{N}_t\mathrm{N}_t} &= p_{n\mathrm{N}_t}p_{n\mathrm{N}_t} && \text{independence in nonpermissive} \\
&= \left[(1-p)p_{p\mathrm{N}_t}\right]\left[(1-p)p_{p\mathrm{N}_t}\right] && \mathrm{N}_t \xrightarrow{\text{wp } p} \mathrm{N}_m \text{ independently} \\
&= (1-p)^2 p_{p\mathrm{N}_t}p_{p\mathrm{N}_t} && \text{algebra} \\
&= (1-p)^2 p_{p\mathrm{N}_t\mathrm{N}_t} && \text{independence in permissive}
\end{aligned}$$

Then, $p_{n\mathrm{N}_t\mathrm{N}_t} = p_{p\mathrm{N}_t\mathrm{N}_t}$ if and only if $p = 0$.

(b) **Bonus.** Show that if the dinucleotide AC is targeted for mutation to AA with probability $p$, then even if the nucleotides of the provirus were independent in the permissive cells, they will not be independent in the nonpermissive cells. So, non-independence is a signal that could used to detect dinucleotide motifs.

**Solution:** We will show by counterexample. Assume $p_{p\mathrm{A}} = p_{p\mathrm{C}} = 0.25$, $p = 0.1$ and A $\to$ C only in dinucleotide AC. Notice that $p_{p\mathrm{A}}$ and $p_{p\mathrm{AA}}$ and similar proportions are the proportions *before* mutation, so the proportions *after* mutation, like $p_{n\mathrm{AA}}$, are a function of those before mutation.

$$\begin{aligned}
p_{n\mathrm{AA}} &= p_{p\mathrm{AA}} + pp_{p\mathrm{AC}} + pp_{p\mathrm{ACA}} && \text{Law of Total Probability}\\
&= p_{p\mathrm{A}}p_{p\mathrm{A}} + pp_{p\mathrm{A}}p_{p\mathrm{C}} + pp_{p\mathrm{A}}^2 p_{p\mathrm{C}} && \text{independence of nucleotides}\\
&= p_{p\mathrm{A}}(p_{p\mathrm{A}} + pp_{p\mathrm{C}} + pp_{p\mathrm{A}}p_{p\mathrm{C}}) && \text{algebra}\\
&= 0.25 \times (0.25 + 0.10 \times 0.25 + 0.1 \times 0.25^2) && \text{substitute values}\\
&= 0.0703125 && \text{compute}\\
&\neq p_{n\mathrm{A}}^2
\end{aligned}$$

because

$$p_{n\mathrm{A}} = p_{p\mathrm{A}} + pp_{p\mathrm{AC}} = p_{p\mathrm{A}} + pp_{p\mathrm{A}}p_{p\mathrm{C}} = p_{p\mathrm{A}}(1 + pp_{p\mathrm{C}}) = 0.25 \times (1 + 0.1 \times 0.25) = 0.25625,$$

so

$$p_{n\mathrm{A}}^2 = 0.0656641.$$

(c) **Bonus.** Discuss the appropriateness of the following test statistic for testing whether CC is a targeted motif (to AC). What value do you anticipate if there is no mutation? What value do you anticipate if there is C $\to$ A mutation, but no dinucleotide targeting? What value do you anticipate if CC is targeted to AC?

$$T = \frac{\frac{N_{n\mathrm{AC}} n_n^2}{(n_n - m_n) N_{n\mathrm{A}} N_{n\mathrm{C}}}}{\frac{N_{p\mathrm{AC}} n_p^2}{(n_p - m_p) N_{p\mathrm{A}} N_{p\mathrm{C}}}}, \tag{1}$$

where $N_{n\mathrm{AC}}$ and $N_{p\mathrm{AC}}$ are the counts of overlapping occurrences of dinucleotide AC in the provirus genomes of nonpermissive and permissive cells, respectively. In addition, $m_n$ and $m_p$ are the numbers of sequences in the nonpermissive and permissive FASTA files, respectively. [This was corrected from the original posting; the effect of the error is not large.]

**Solution:** We can see after a bit of staring that $T$ is an estimate of

$$\frac{\frac{p_{n\mathrm{AC}}}{p_{n\mathrm{A}} p_{n\mathrm{C}}}}{\frac{p_{p\mathrm{AC}}}{p_{p\mathrm{A}} p_{p\mathrm{C}}}}$$

obtained by plugging in sample proportions. If nucleotides are independent in both cell types, then both the numerator and denominator should be near 1, and the overall ratio should also be near one. If AC co-occur much more strongly in nonpermissive cells, perhaps because it was also created by mutation of CC, then we expect the numerator to be large. If CC fail to occur much in nonpermissive cells, perhaps because it was mutated to AC, then we expect the numerator to be small. Thus, $T$ is sensitive to the truth of dinucleotide-targeted mutation.

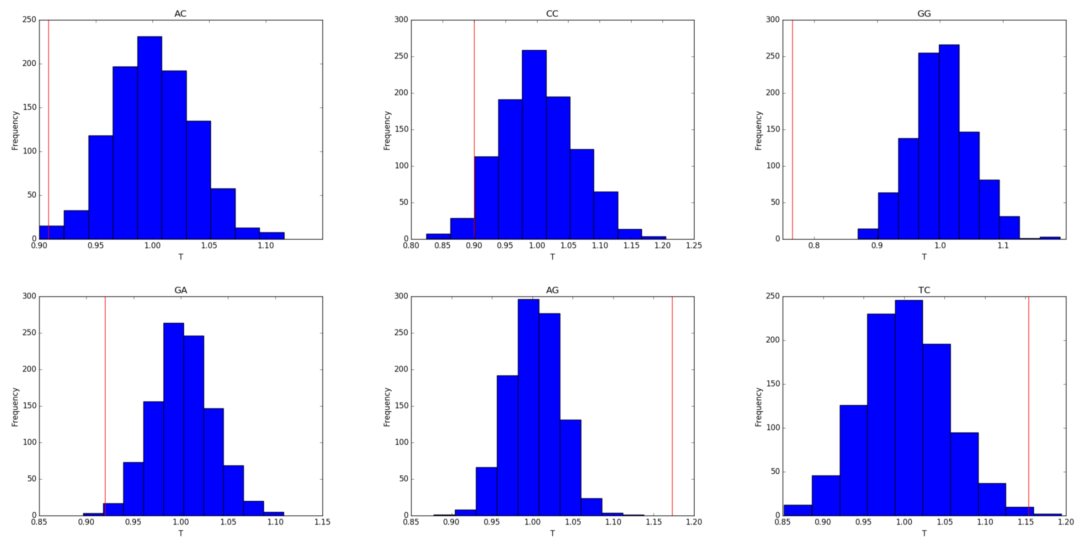
If nucleotides are independent in the permissive cells, then the denominator is superfluous: it should usually be $\sim 1$. However, if nucleotides are not independent because the sequences are biologically functional, then this test statistic should still be near 1 if the level of dependence has not changed much. The hope is that this statistic $T$ will therefore be resistant to violations of the independence assumption while still picking up the signal of new dependence induced by dinucleotide-targeted mutation. Furthermore, we can argue our simulator, which does not account for dependence, should still validly assess significance because its failure to mimick the dependence is (hopefully) of little consequence for this test statistic.

(d) Using the specific null hypothesis $H_0$ from the previous question 2 (that assumes no mutation), simulate many $T$ from Eq. 1 and plot a histogram of the values. (Yes, you can do this in Python!) Do you think there is evidence that CC is targeted? What about a more appropriate dinucleotide, motivated by your results from question 2?

**Solution:** The following code implements a `Python` solution. It also has some code for answering the final Part e.

```
import numpy as np
from hw3q2c import *
from hw3q2d import simulate_fasta

def count_dinuc(fasta_dict, dinuc):
        cnt = 0
        for s in fasta_dict.values():
                for i in range(len(s)-1):
                        if s[i:i+2] == dinuc:
                                cnt += 1
        return cnt

def compute_T(mononuc_cntsA, mononuc_cntsB, dinuc_cntA, dinuc_cntB,\
        dinuc, mA, mB):
        n_A = sum(mononuc_cntsA)
        n_B = sum(mononuc_cntsB)
        nucs_idx = ["ACGT".index(dinuc[0]), "ACGT".index(dinuc[1])]
        T_num = float(dinuc_cntA)*n_A**2 / (n_A-mA) /\
                mononuc_cntsA[nucs_idx[0]] / mononuc_cntsA[nucs_idx[1]]
        T_den = float(dinuc_cntB)*n_B**2 / (n_B-mB) /\
                mononuc_cntsB[nucs_idx[0]] / mononuc_cntsB[nucs_idx[1]]
        return T_num/T_den



def simulate_T(perm_fasta, nonperm_fasta, dinuc):
        perm_cnt_dinuc = count_dinuc(perm_fasta, dinuc)
        nonperm_cnt_dinuc = count_dinuc(nonperm_fasta, dinuc)
        m_n = len(nonperm_fasta)
        m_p = len(perm_fasta)
        t_0 = compute_T(nonperm_cnts, perm_cnts, nonperm_cnt_dinuc,\
                perm_cnt_dinuc, dinuc, m_n, m_p)

        B = 1000
        p_value = 0
        T_hist = []

        for b in range(B):
                perm_sim_fasta = simulate_fasta(perm_fasta,\
                        p_estimate)
                nonperm_sim_fasta = simulate_fasta(nonperm_fasta,\
                        p_estimate)
                perm_sim_cnt = count_nucs(perm_sim_fasta)
                nonperm_sim_cnt = count_nucs(nonperm_sim_fasta)
                perm_cnt_dinuc = count_dinuc(perm_sim_fasta, dinuc)
                nonperm_cnt_dinuc = count_dinuc(nonperm_sim_fasta, dinuc)
                T_hist.append(compute_T(nonperm_sim_cnt, perm_sim_cnt,
                        nonperm_cnt_dinuc, perm_cnt_dinuc, dinuc, m_n,\
                        m_p))

        return [T_hist, t_0]

import matplotlib
matplotlib.use('Agg')   # make png plot
import matplotlib.pyplot as pp

# using data loaded from hw3q2c
dinucs = ["AC","CC","GG","GA","AG","TC"]
T_hist = {}
for dinuc in dinucs:
        T_hist[dinuc] = simulate_T(perm_fasta, nonperm_fasta, dinuc)
        p_value = 0
        if (T_hist[dinuc][1] < 1):
                for i in range(len(T_hist[dinuc][0])):
                        if T_hist[dinuc][0][i] <= T_hist[dinuc][1]:
                                p_value += 1
        else:
                for i in range(len(T_hist[dinuc][0])):
                        if T_hist[dinuc][0][i] >= T_hist[dinuc][1]:
                                p_value += 1
        print "Reject H0 for", dinuc, "with p-value", 2*float(p_value) /\
                len(T_hist[dinuc][0])

        pp.figure()
        pp.hist(T_hist[dinuc][0])
        pp.axvline(x = T_hist[dinuc][1], color='r')
        pp.title(dinuc)
        pp.xlabel("T")
        pp.ylabel("Frequency")
        pp.savefig('hw3q3d' + dinuc + '.png')
```

The histogram plots for various tested dinucleotides are plotted below.



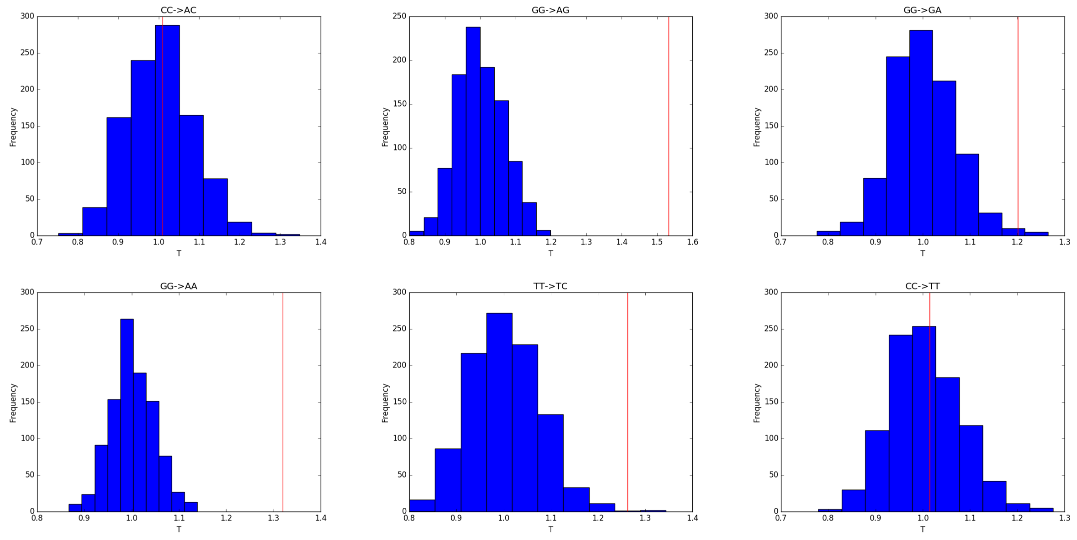
From these plots, we observe AC, GG, and GA are underrepresented in nonpermissive cells, while AG and TC are over-represented. The dinucleotide CC does not appear to change significantly, but since AC is also on the low side, it cannot be that CC $\rightarrow$ AC. The only plausible possibility from these comparisons is GG $\rightarrow$ AG, which is in fact the motif that is targeted by APOBEC3G. In Part e, we will see that the result is statistically significant as well, but clearly, this has not been an exhaustive study of all 16 possible dinucleotides.

**More.** Finally, there is a flaw with the proposed test statistic for answering the question: Which dinucleotide is targeted and how? Some motifs may show as mutated because they overlap with actually mutated motifs. For example, it is likely that the mutation GG $\to$ AA does *not* happen, but it is significant because AA can show up after targeting AGG at the GG dinucleotide. The proper way to deal with these false signals is to use a Markov chain model that handles the actual and induced dependencies (no one has done this!). And please note, machine learning techniques will *not* solve this problem. They may be very good at separating permissive and nonpermissive sequence, but they can provide very little insight into the mechanism driving the differences. Sometimes it matters to you or your client, sometimes it does not.

Here I propose a better test statistic that gets more directly at how dinucleotides get targeted. Let $T_{\text{AC}}$ be the statistic suggested in Eq. (1), then I propose

$$\frac{T_{\text{AC}}}{T_{\text{CC}}}$$

to test $H_0$ no mutation against CC $\to$ AC, with large values indicating that this is a likely targeting and small values arguing that AC $\to$ CC is the direction. The code needs slight modification, then we observe the following plots.



These plots suggest GG $\to$ AG and GG $\to$ AA are definitely happening, maybe GG $\to$ GA. The strong result for TT $\to$ TC is unexpected given that APOBEC3G should not mutate either of those nucleotides. Well, it literally mutates C $\to$ U, but we observe it as G $\to$ A in the opposite strand. I'm sparing you the details, but read all you like. It is fascinating. So is this a false positive? Possibly. There are many assumption violations (Question 2, Part (a)) we could explore. Data analysis is messy. The best thing you can do is never, ever grow complacent. Always be thinking.

(e) **Bonus.** Using your knowledge of the mutation from question 2, compute a $p$-value for the probability of obtaining a test statistic $T$ as or more extreme than the observed one for your favorite motif.

**Solution:** The distributions look reasonably symmetric around 1, so it is plausible to compute the $p$-value as twice the error in the tail:

$$\frac{2}{B} \times \begin{cases} \sum_{i=1}^{B} \mathbb{1}\left\{T^{(b)} \geq t_0\right\} & \text{if } t_0 > 1 \\ \sum_{i=1}^{B} \mathbb{1}\left\{T^{(b)} \leq t_0\right\} & \text{if } t_0 < 1. \end{cases}$$

The two cases determine which peak we are in, although 1 is not always the peak, and we should be careful about the calculations when $t_0 \approx 1$. On the other hand, if $t_0 \approx 1$, then we cannot reject $H_0$ anyway. Do you have any other ideas?

The output from the code from Part d reports the $p$-values for each motif.

```
## Reject H0 for AC with p-value 0.008
## Reject H0 for CC with p-value 0.07
## Reject H0 for GG with p-value 0.0
## Reject H0 for GA with p-value 0.006
## Reject H0 for AG with p-value 0.0
## Reject H0 for TC with p-value 0.004
```

Everything but CC looks fairly convincing, but clearly GG and AG have the biggest signal. By combining these significance result with interpretation as in the answer to Part d, we can draw some supported conclusions, though the ratio of $T$ statistics proposed in Part d are easier to interpret.