BCBio 444: Bioinformatics Analysis

Karin S. Dorman

Department of Statistics
Department of Genetics, Development & Cell Biology
Program in Bioinformatics & Computational Biology
Iowa State University

September 14, 2017

Introductions

- Introductions: highest course in biology, math, statistics, computer science, major(s).
- syllabus

Introduction

Bioinformatics is like finding a needle in a haystack where every piece of hay looks like a needle. And the needle is cancer.

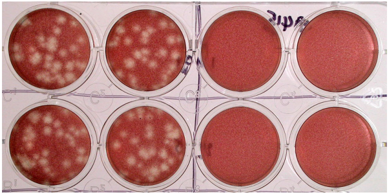
– darkhelmet41290 [reddit]

A Motivating Example

Much of lower-level bioinformatics and this course is about learning how to identify and use computational tools to answer standard questions, but it will not take long before you encounter data that looks different from standard types of data or biological questions unlike those that have known procedures to answer.

We will start with a motivating example that demonstrates how you can use the general-purpose tools of bioinformatics to put together your own methodology and answer for a example dataset.

Discovery: antiviral function



Suppose you have discovered a novel biological process that attacks and destroys some viruses. You have been able to grow a susceptible virus in two types of cells with respective to this novel process, one permissive and the other not. You *hypothesize* that the nonpermissive cells actively mutate the virus genome, rendering them nonfunctional. You suspect the mysterious function is specific, targeting and mutating one type of nucleotide base $\mathrm{N}_t \in \Omega = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ in the virus to another, wrong nucleotide base $\mathrm{N}_m \in \Omega$ with $\mathrm{N}_m \neq \mathrm{N}_t$.

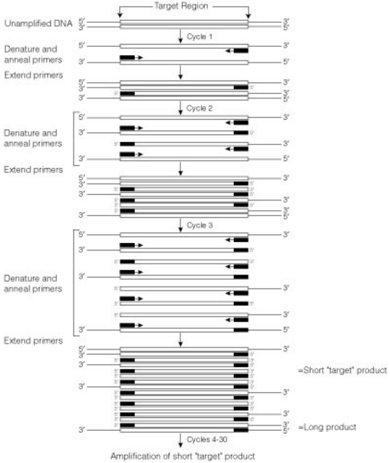
Your Goal

Your first goal is to determine what are N_t and N_m, in other words, what is the mutation that this novel biological process is inducing?

An Experiment

The susceptible virus in your study *integrates* its genome into host cells. You use this fact to design an experiment.

- Grow 1 plate each of (non)permissive cells.
- Add 10 moi of virus to each plate and incubate.
- Collect the cells and isolate the DNA.
- Amplify the virus genome using PCR (see right).
- Fragment and sequence.



Data in Fasta Format

Virus from nonpermissive cells:

```
>n.1
AAGGACCCTGTGCATAAAGTATATTATGACCCATCAAAAGACTTAATAGCAGAGATACA
GAAGCAAAGACAAGACCAATAGACATATCAGATTTATCAAGAACCATTTAAAAATCTGA
AAACAAGGAAATATGCAAGAAAAAAGTCTGCTCACAC...
>n.2
AAAAATAACATGGTAGAGCAGATGCATACAGATATAGTCAGTCTATAAGAACAAAGCCT
AAAGCCATGTGTAAAGTTAACCCCTCTCTGCGTTACTTTACATTGTAACAATGTCACAG
GGAACATCACAGAGAGAATCAGAGAAGAAAAAAAAAA...
...
```

Virus from permissive cells:

```
>p.1
GACCCTTATCCCGAACCCAAGGGAACCCGACAGGCCAGGAAGAATCGAAGAAGAAGGTG
GAGAGCAAGACAAAGAGAGATCCGTGCGATTAGTGAGCGGATTCTTAGCACTTGCCTGG
GACGACCTACGGAGCCTGTGCCTCTTCAGCTACCACC...
>p.2
ACCTAGTGTGAACAATGAGACACCAGGAATTAGATATCAGTACAATGTGCTTCCACAAG
GATGGAAAGGATCACCAGCAATATTCCAAAGTAGCATGACAAAAATCTTAGAACCTTTC
AGAAAGCAAAATCCAGAAATAACTATCTATCAATACA...
...
```

Questions

It could be that one or a few specific N_t nucleotides in the genome that are critical for virus function are being targeted for mutation. It is also possible that random N_t nucleotides are mutated until eventually virus function is disrupted.

- How can we use the data to distinguish these two hypotheses?
- How can we detect which, if any, nucleotide is targeted and how it is mutated?
- Why might the experiment not work?

Relevant data

I argue that the relevant data in the Fasta files for answering the previous slide's questions are the nucleotide counts.

Cell type	A	C	G	T
Nonpermissive cells	$n_{n\mathrm{A}}$	$n_{n\mathrm{C}}$	$n_{n\mathrm{G}}$	$n_{n\mathrm{T}}$
Permissive cells	$n_{p\mathrm{A}}$	$n_{p\mathrm{C}}$	$n_{p\mathrm{G}}$	$n_{p\mathrm{T}}$

- Will the counts $\boldsymbol{n}_{n\cdot} = (n_{n\mathrm{A}}, n_{n\mathrm{C}}, n_{n\mathrm{G}}, n_{n\mathrm{T}})$ and $\boldsymbol{n}_{p\cdot}$ be identical?
- Why will they vary?
- How can we determine which nucleotide, if any, is mutated and how it is mutated?
- When can we conclude that, yes, for example, the novel mechanism *does* mutate A to C?

Detecting signal

This is an example of detecting a signal. We can use two big ideas from statistics to help:

- Statistical hypothesis testing (*e.g.* z-test, t-test).
- Resampling to quantify variability.

First, let's review (or learn) basic probability & statistics...

Foundations of Probability

- **random experiment**: a repeatable process whose outcome cannot be predicted beforehand, but will be observed after the experiment is complete
- **outcome**: one possible output of a random experiment
- **sample space**: the set of possible outcomes of a random experiment
- **event**: a set of outcomes
- **probability**: given a random experiment, a measure of how likely an event is, in the range $[0, 1]$

In order to determine the probability of events, one must hypothesize a model. This is where the bioinformatics team needs to work together when developing new bioinformatics methods. Quantitative scientists propose models; biologists tear them down. Teamwork!

Examples – Probability

- *Experiment*: toss a coin; *outcome*: head (H), *sample space*: $\Omega = \{H, T\}$; *event*: $E = \{H\}$, *probability*: $P(\{H\}) = P(\{T\}) = 0.5$.
- *Experiment*: sequence a 15 bp fragment of mRNA; *outcome*: ACCGAGGTCTCTAAA; *sample space*:

$$\Omega = \underline{\hspace{6cm}};$$

event: $E = \{\text{YYYYYYYYYYYYYYY}\}$; *probability*:

$$P(\{\text{ACCGAGGTCTCTAAA}\}) = \underline{\hspace{6cm}}$$

$$P(\{\text{YYYYYYYYYYYYYYY}\}) = \underline{\hspace{6cm}}$$

Example – Nucleotide Counts/1

- **random experiment**: (1) collect data according to biological experiment, (2) count the nucleotides in the fasta files and store as count vectors $\boldsymbol{n}_p$ and $\boldsymbol{n}_n$.
- **outcome**: We will see an example on the next slides...
- **sample space**:

$$\Omega = \left\{(\boldsymbol{n}_p, \boldsymbol{n}_n) : n_{hi} \in \{0, \mathbb{Z}^+\}, h \in \{p, n\}, i \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}\right\} .$$

- **event**:

$$E = \left\{(\boldsymbol{n}_p, \boldsymbol{n}_n) \in \Omega : n_{p\mathrm{A}} > n_{n\mathrm{A}}\right\} .$$

- **probability:**

What can we do for the probability?

Example – Nucleotide Counts/2

This is a silly model for the count data $\boldsymbol{n}_n$ and $\boldsymbol{n}_p$ as demonstrated in R code.

```r
# generate two samples of 100 nucleotides by
# flipping a fair, 4-sided coin:
n.n.data <- rmultinom(n = 100, size = 1,
  prob = rep(0.25, 4))
n.p.data <- rmultinom(n = 100, size = 1,
  prob = rep(0.25, 4))
n.n <- rowSums(n.n.data)
n.p <- rowSums(n.p.data)
n.n

## [1] 29 25 19 27

n.p

## [1] 21 21 26 32
```

Example – Nucleotide Counts/3

Cell type	A	C	G	T
Nonpermissive cells	29	25	19	27
Permissive cells	21	21	26	32

- If you had to guess the target nucleotide N$_t$ and mutated nucleotide N$_m$ were from this data, what would you choose?
- In this case, the two rows of data are generated under identical conditions: there is no actual difference!
- So, how can we be sure a difference we see is real?

Answer: We need to know the probability of every outcome. If the observed outcome is very unlikely, then we suspect there is some process, such as mutation, driving the pattern in the data.

Review – Probability

- **Vocabulary.** random experiment, sample space, outcome, probability
- A model (or hypothesis) is necessary to compute probabilities.
- Most scientific experiments are random, at least there is measurement error: Outcomes contain noise.
- Scientific experiments are designed to answer questions or test hypotheses.
- Some noise can look like a meaningful pattern: *e.g.* it looked like G→A mutation in the simulated count data.
- The triumvirate of bioinformatics:
    - Biological knowledge/cleverness will determine the right experiment & visible pattern to confirm the hypothesis;
    - Computers will help us extract the pattern;
    - Statistics (and computers) will help us *distinguish the pattern (signal) from the noise*.

Statistical Inference

Statistical inference is the process of deducing facts about the **population** based on a **simple random sample** (constituting **data**) from the population. There are two types of statistical inference:

- estimation
- hypothesis testing

Examples – Population/Sample

- Population: ISU students; Sample: this class (Is it a *random* sample? If I want to deduce the mathematical skills of ISU students, can I use you guys as the sample?)
- Population: mRNA in a cancer cell; Sample: a random set of mRNA from a random set of cancer cells from a random tumor
- What is the sample in the scientific experiment our biologist undertook? What is the population?

We observe properties of the **sample** to draw conclusions about the *unobservable* **population** while accounting for the *randomness/noise* of sampling.

Statistic

A **statistic** is any function of a sample that requires nothing more than the sample to compute.

Example – Statistic

Which of these are statistics?

- Population: ISU students; Sample: this class. The average height of students in this room.
- Population: ISU students; Sample: this class. The number of inches each student's height differs from the mean ISU student height.
- Population: provirus genome fragments in permissive cells; Sample: our fasta file.

$$n_{pC}$$

- Population: provirus genome fragments in permissive *and* nonpermissive cells; Sample: our fasta file*s*.

$$n_{pC} - n_{nC}$$

Random variable

Definition (random variable)

A random variable $X : \Omega \rightarrow E$ is a function that maps outcomes $\omega \in \Omega$ of a random experiment to a subset of the real line $E \subset \Re$.

Examples:

- **Discrete random variables**: Bernoulli, Multinoulli, Binomial, Multinomial, Geometric, Hypergeometric, Negative Binomial, Poisson.
- **Continuous random variables**: Uniform, Normal (or Gaussian), Exponential, Gamma, Beta, Chi-Squared, *t*, *F*, Laplace, Cauchy, Dirichlet, Multivariate Gaussian.

Review – Statistical inference

Statistical inference is the process of deducing facts about the **population** based on a **simple random sample** (constituting **data**) from the population.

- estimation
- hypothesis testing

To perform statistical inference, we compute **statistics** on samples. Some statistics are useful for estimation: they are called **estimators**. Other statistics are useful for hypothesis testing: they are called **test statistics**.

Statistical hypothesis testing I

- Identify one, two or more hypotheses. A hypothesis is a model for reality. In statistical hypothesis testing, we must ultimately define a really, really precise model. In fact, it must be a precisely defined procedure capable of generating the **test statistic** (see below) computed from your data sample.
- In *frequentist hypothesis testing*, we focus on one particular hypothesis called the **null hypothesis**, denoted by H_0. If we have many hypotheses, we would test each in turn.
- Then, we choose a **test statistic** that is *sensitive to the truth of* H_0, that *signals the validity of* H_0.

Vague vs. Specific Hypotheses

A challenge is that you usually start with a vague hypothesis, such as “One nucleotide is mutated to another nucleotide” or “Gene A has nothing to do with cancer”. These hypotheses are nowhere near sufficient mimics of the experiment able to generate faux data. Here are some examples of vague vs. specific hypotheses.

- **Vague.** The sample $x_1, x_2, \ldots, x_n$ is iid Normally distributed. **Specific.** $x_1, x_2, \ldots, x_n \overset{\text{iid}}{\sim} \mathcal{N}(0, 3)$; the mean and variance must be specified.
- **Vague.** The sequences are random. **Specific.** Each nucleotide is an iid random choice from set $\{\text{A}, \text{C}, \text{G}, \text{T}\}$ with probabilities $p_{\text{A}} = 0.21, p_{\text{C}} = 0.13, p_{\text{G}} = 0.37, p_{\text{T}} = 0.29$.

By the way, iid stands for “independent and identically distributed,” and random variables are iid if they are independent and share the same distribution.

Simulation

Once we have a **specific null hypothesis** H_0 that truly mimics the real experiment and can be used like a recipe to generate the test statistic T, then you are said to be able to **simulate** T using the H_0 model. Your ability to program in `Python` or any other language combined with your modeling skills gives you the ability to simulate data, and it is a super power!

Review Hypothesis Testing [lab]

1. From a **vague null hypothesis**, identify a test statistic T that is sensitive to the truth of the null hypothesis.
2. Work the hypothesis into a **specific null hypothesis** H_0 that is a **model** for the scientific experiment and can simulate test statistics T. You probably need to iterate steps 1-2 before finalizing your test statistic T and specific null hypothesis H_0.
3. Compute the ***p*-value**, or probability of observing a test statistic T as or more extreme than the observed test statistic t_0 when H_0 is true. Make a decision, if necessary. Otherwise, just report the p-value as your evidence for/against H_0.

Example – *z*-test/1

The **null hypothesis** tested by the *z*-test is

$$H_0 : x_1, \ldots, x_n \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2), \text{ where } \mu = \mu_0 \text{ and } \sigma \text{ is known.}$$

The *z*-test uses the *z* **test statistic**, namely

$$z = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}} \sim \mathcal{N}(0, 1),$$

where

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

is the **sample mean.**

Why does this **test statistic** signal the validity of H_0?

Exercise – z-test/2 [lab]

Is our model/hypothesis specified with enough detail to simulate a test statistic?

- Write a program that simulates $x_1, x_2, \ldots, x_n$ according to H_0 with $\mu_0 = 3$, $\sigma = 1$, and $n = 10$. Use these data to simulate a test statistic z.
    - See function `random.gauss()` in library `random`.
- Write a program that simulates z directly.
- Which program is more efficient?

Exercise – *z*-test/2

Is our model/hypothesis specified with enough detail to simulate a test statistic?

- Write a program that simulates $x_1, x_2, \ldots, x_n$ according to H_0 with $\mu_0 = 3$, $\sigma = 1$, and $n = 10$. Use these data to simulate a test statistic z.
    - See function `random.gauss()` in library `random`.
- Write a program that simulates z directly.
- Which program is more efficient?
- So, yes, we can use `Python` to simulate a test statistic under H_0. In the second case, we are relying on results proven in Stat 341/2.

Solution – *z*-test/1 [lab]

The following code simulates the data $x_1, x_2, \ldots, x_n$:

```
import random

n = 10    # sample size
mu_0 = 3  # hypothesized
sigma = 1 # known

x = []    # simulate x
for i in range(n):
   x.append(random.gauss(mu = mu_0, sigma = sigma))
x_bar = sum(x) / n       # compute sample mean
z = (x_bar - mu_0)/sigma # compute z

print "H0: mu =", mu_0  # print stuff
print "Data are", x
print "Mean is", x_bar
print "Z statistic is", z
```

Solution – *z*-test/2 [lab]

The following code simulates the test statistic *z* directly:

```
import random

# simulate N(0,1)
z = random.gauss(mu = 0, sigma = 1)
print "Z statistic is", z
```

Clearly, this approach is much more efficient: less time to write the code, less time to run the code. In some sense, statisticians are builders of efficient algorithm; they use mathematical proofs to provide shortcuts.

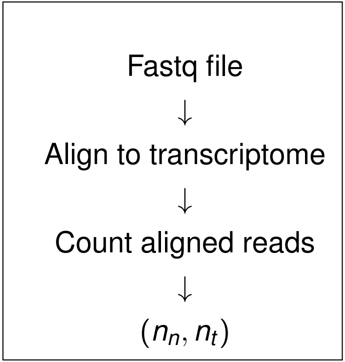
Simulating Data is Flexible

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

$$\downarrow$$

$$T(\boldsymbol{x}) = z = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}}$$

Since the data processing pipeline is computer-based, you can simulate at any level that is convenient and apply the pipeline to simulate the test statistic.



Fastq file

$\downarrow$

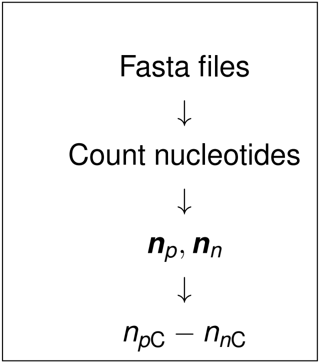
Align to transcriptome

$\downarrow$

Count aligned reads

$\downarrow$

(n_n, n_t)



Fasta files

$\downarrow$

Count nucleotides

$\downarrow$

$\boldsymbol{n}_p, \boldsymbol{n}_n$

$\downarrow$

$n_{pC} - n_{nC}$

Example – *t*-test/1

The null hypothesis tested by the *t*-test is

$H_0 : x_1, \ldots, x_n \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2),$ where $\mu = \mu_0$ and σ is unknown.

The *t*-test uses the *t* test statistic, namely

$$t = \frac{\overline{x} - \mu_0}{s/\sqrt{n}} \sim t_n,$$

where

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2$$

is the **sample variance.**

Example – t-test/2

- Is t sensitive to the truth of H_0?
- Is the model/hypothesis specified with enough detail to simulate the test statistic? How would you do it in `Python`?

Example – HW1 SCLC/1

Which of these are observed test statistics?

- (n_n, n_t), where n_n is the number of reads mapping to the gene of interest in the normal cells, and n_t is the number of reads mapping to the gene of interest in the tumor cells.
- $n_t - n_n$
- Indicator random variable,

$$\mathbb{1}\left\{n_t > n_n\right\} = \begin{cases} 1 & \text{if } n_t > n_s \\ 0 & \text{otherwise.} \end{cases}$$

The quantity (n_n, n_t) is not an observed test statistic because it is bivariate. A test statistic maps the sample to the real line (not the real plane).

Example – HW1 SCLC/2

- Colin suggested we model $N_t - N_n$ as the difference between two Poisson random variables. Where did the idea come from? If $X \sim \text{Poisson}(\lambda)$, then you learn in Stat 341 that

  $$P(X = x) = \frac{e^{-\lambda}\lambda^x}{x!}, x \in \{0, 1, 2, \ldots\}.$$

  The range of X is the counting numbers. The statistics N_t and N_n also have the same range, so why not "model" $N_t - N_n$ as the difference in two Poisson random variables. Equivalently, assume

  $$N_t \sim \text{Poisson}(\lambda_t) \qquad N_n \sim \text{Poisson}(\lambda_n).$$

  This is not the only model you could think of.

- The test statistic $\mathbb{1}\{N_t > N_n\}$ is a Bernoulli random variable:

  $$\mathbb{1}\{N_t > N_n\} \sim \text{Bernoulli}(p)$$

  for some $p \in [0, 1]$.

Example – HW1 SCLC/3

Our model is still not precise enough: we cannot simulate data if we do not know the values of the **parameters**. In some cases, we can argue for specific values. In others, we must turn to **parameter estimation** to produce values.

- **Poisson**. Exercise in lab (next slide).
- **Bernoulli**. If the gene in question has nothing to do with SCLC, then its expression should not be higher or lower in the tumor compared to the normal tissue. The contrapositive statement may make more sense to you: If the gene has higher or lower expression in the tumor, then evidently it is related to cancer! If the contrapositive is true, then so is the first statement. Q.E.D. Thus, we conclude $p = 0.5$ under the **specific null** H_0, and

$$\mathbb{1}\left\{N_{it} > N_{in}\right\} \sim \text{Bernoulli}(0.5) \text{ and } \sum_{i=1}^{14} \mathbb{1}\left\{N_{it} > N_{in}\right\} \sim \text{Bin}(14, 0.5),$$

where (N_{it}, N_{in}) are the counts for the ith sampled patient.

Exercise – Poisson [lab]

Tissue	**Counts**													
Tumor	28	33	23	26	24	29	31	26	28	20	24	25	26	26
Normal	13	14	21	17	10	9	19	24	15	20	11	17	25	23
$n_t - n_n$	15	19	2	9	14	20	12	2	13	0	13	8	1	3

- **Finish defining the specific hypothesis H_0.** Focusing just on the patient 1 data, what should the values of λ_t and λ_n be if the gene is not associated with SCLC? How can we estimate them? (Hint: The λ parameter of the Poisson random variable X is the mean count, $\mathbb{E}[X]$.)
- **Simulation.** If the test statistic $T = N_t - N_n$, can you use `Python` to simulate a value of T under our **specific hypothesis**?
  - Plan the algorithm.
  - Implement.

Solution – Poisson [lab]

```
import numpy.random

# input data
observed_cnts = [28,13]

# estimate lambda
lambda_hat = float(sum(observed_cnts))
   /len(observed_cnts)

# simulate data
simulated_cnts = numpy.random.poisson(
   lam = lambda_hat, size = 2)

# compute T
T = simulated_cnts[0] - simulated_cnts[1]

print "Observed test statistic is",\
   observed_cnts[0] - observed_cnts[1]
print "Simulated test statistic is", T
```

Exercise – Mutated Provirus/1 [lab]

Cell type	A	C	G	T
Nonpermissive cells	29	25	19	27
Permissive cells	21	21	26	32

Our **vague hypothesis** is that some specific nucleotide is being mutated to some other specific nucleotide.

- What **test statistics** T could we use?
- What **specific model/hypothesis** H_0 could we use to simulate T in a realistic way?

Exercise – Mutated Provirus/2

- $T = \frac{N_{nA}}{N_{pA}}$ or $T = N_{nA} - N_{pA}$ are sensitive to the truth of H_0 if the target nucleotide N$_t$ or mutant nucleotide N$_m$ is A.
  - If $n_n = n_p$, then under H_0, it is reasonable to assume (and there is good theory to back it up if nucleotides are independent) $N_{nA}, N_{pA} \overset{\text{iid}}{\sim}$ Poisson(λ) for some λ. We can estimate $\hat{\lambda} = \frac{N_{nA}+N_{pA}}{2}$ as the sample mean of the observed counts.
  - If $n_n \neq n_p$, then $N_{nA} \sim$ Poisson(λ_n) is independent of $N_{pA} \sim$ Poisson(λ_p). However, there remains a relationship between λ_n and λ_p. In fact, $\lambda_n = n_n\lambda$ and $\lambda_p = n_p\lambda$, where λ is the expected increase in A count per observed nucleotide. Thus, since $\hat{\lambda}_n = N_{nA}$ and $\hat{\lambda}_p = N_{pA}$, we have $\hat{\lambda} = \frac{N_{nA}+N_{pA}}{n_n+n_p}$.

Exercise – Mutated Provirus/3

- $T = \frac{N_{nA}/n_n}{N_{pA}/n_p}$ or $T = N_{nA}/n_n - N_{pA}/n_p$ are sensitive to the truth of H_0 if the target nucleotide N$_t$ or mutant nucleotide N$_m$ is A.
  - If n_n and n_p are large, then it is reasonable to assume (and there is good theory to back it up if nucleotides are independent) $\frac{N_{nA}}{n_n} \sim \mathcal{N}(\mu, \sigma_n^2)$ and $\frac{N_{pA}}{n_p} \sim \mathcal{N}(\mu, \sigma_p^2)$. We can estimate $\hat{\mu} = \frac{N_{nA}}{2n_n} + \frac{N_{pA}}{2n_p}$ and $\sigma_n^2 = \frac{\hat{\mu}(1-\hat{\mu})}{n_n}$ and $\sigma_p^2 = \frac{\hat{\mu}(1-\hat{\mu})}{n_p}$.
  - I am using the Central Limit Theorem to derive these results. Very, very important theorem.

Central Limit Theorem

Theorem (Central Limit Theorem)

Let $X_1, X_2, \ldots, X_n \overset{iid}{\sim} (\mu, \sigma^2)$ *with* $\sigma^2 < \infty$*. Then, the distribution of the sample mean is, to very good approximation as* $n \to \infty$*,*

$$\overline{X} \dot{\sim} \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right).$$

In particular, if $X_1, X_2, \ldots, X_n \overset{\text{iid}}{\sim} \text{Bernoulli}(p)$, then

$$\overline{X} \dot{\sim} \mathcal{N}\left(p, \frac{p(1-p)}{n}\right).$$

Statistical hypothesis testing II

- To make a decision, we compute the *p*-value, which is the probability of obtaining a test statistic T *as extreme or more extreme* than the observed test statistic t_0 when H_0 is true:

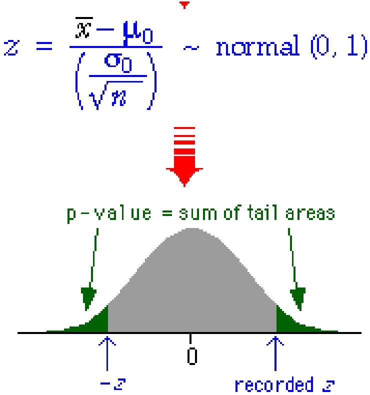
$$P\left(\{T \text{ as or more extreme than } t_0\} \mid H_0\right).$$

  If the *p*-value is p_0, then we can conclude H_0 is *not* true, and the chance of being wrong is p_0.

- If we can derive the **probability distribution** of the *test statistic* under H_0, then we can compute the *p*-value directly. The probability distribution of a test statistic (or any statistic) is called a **sampling distribution**. The theory classes in statistics derive tons of these sampling distributions. But *you* can simulate. You can benefit from, but you don't need theory.

The *p*-value is an example of a **conditional probability** (click to remind yourself what this is).

z test *p*-value


$z = \frac{\bar{x} - \mu_0}{\left(\frac{\sigma_0}{\sqrt{n}}\right)} \sim$ normal (0, 1)
p-value = sum of tail areas
0
-z
recorded z

This solution uses the **sampling distribution**, $\mathcal{N}(0, 1)$, of the *z* **test statistic** derived in a statistics class using mathematical theory. In such classes, you may look up these values in a book, or you may use `R` to find the green area in the above figure. You can also use `Python`.

Exercises – Compute p-values

- Compute the p-value for z-test statistic $z = 3.2$ in `Python`.
  - The `norm.cdf(x, loc, scale)` function in `scipy.stats` provides the area under the curve to the left of its first argument `x`.
- Compute the p-value for t-test statistic $t = 3.2$ in `Python`.
  - The `t.cdf(x, df)` function in `scipy.stats` provides the area under the curve to the left of its first argument `x`.

Solution – *p*-value for *z*-test

```
import math
from scipy.stats import norm

# determine observed test statistic
#z = norm.rvs(loc = 0,       # by simulation
#   scale = 1, size = 1)
z = 3.2             # number given on slides

# compute p-value
p_value = 2*norm.cdf(x = -math.fabs(z),
   loc = 0, scale = 1)


# print stuff
print "Z statistic is", z
print "P-value is", p_value
```

Solution – *p*-value for *t*-test

```
import math
from scipy.stats import t

# sample size necessary
n = 10

# determine observed t statistic
#t_stat = t.rvs(df = n,     # by simulation
#   size = 1)
t_stat = 3.2       # number given on slides
p_value = 2*t.cdf(x = -math.fabs(t_stat),
   df = n)


# print stuff
print "Student’s t statistic is", t_stat
print "P-value is", p_value
```

Decision Time/1

Can we reject the **vague null hypothesis**

$$H_0^{\text{(vague)}} : \mu = \mu_0?$$

Decision Time/2

Can we reject the **vague null hypothesis**

$$H_0^{\text{(vague)}} : \mu = \mu_0?$$

The *p*-values from both *z*- and *t*- tests are small (below 0.01), so we are quite confident to reject $H_0^{\text{(vague)}}$. We have a less than 1% chance of being wrong to reject $H_0^{\text{(vague)}}$.

However, we are actually rejecting our **specific null hypothesis** H_0. If any *one* of our assumptions was incorrect, it is possible the data are rejecting that assumption, not that $\mu = \mu_0$.

But What If...?

But what if you cannot derive (or you do not remember) the sampling distribution of your test statistic? Do we need to become full-fledged statisticians to compute a *p*-value?

No! (Use the simulation super power!)

Resampling to Compute *p*-Value

- The outcome of our scientific experiment has been distilled down to the **observed test statistic** t_0, which captures the signal in the data indicating the truthfullness of $H_0^{\text{(vague)}}$.
- The reason we need the **sampling distribution** of the test statistic T is to understand how it varies because of the randomness of the experiment. We need to understand this variation to know if the signal exceeds the usual variation.
- If we could repeat the **scientific experiment** many times, then we could observe this variation directly.
- We have used our modeling skills to build a **specific null model** H_0 that mimics the scientific experiment and simulates random test statistics T.
- Let's use it and repeatedly *simulate* the random experiment *in silico*.

Estimating the p-value

The p-value is the probability of obtaining T as or more extreme than the observed test statistic t_0. It is the *success probability* of a Bernoulli random variable that indicates the event $\{T$ as or more extreme than $t_0\}$. You can estimate this probability from repeated observations of the test statistic $T^{(1)}, T^{(2)}, \ldots$ as the proportion of $T^{(i)}$ as or more extreme than the observed t_0. Thus,

$$P(\{T \text{ as more more extreme than } t_0\} \mid H_0) \approx \frac{1}{B}\sum_{i=1}^{B} \mathbb{1}\left\{T^{(i)} > t_0\right\},$$

where B is the number of simulations you did.

Exercise – Resampling z test

Use `Python` to determine how rare the test statistic $z = 3.2$ is using simulation. In other words, assume you do *not* have `norm.cdf(x, loc, scale)` available to you.

Solution – Resampling *z* test

```
import math
from scipy.stats import norm

z_0 = 3.2    # observed test statistic
B = 10000    # number of resamples

# resample test statistics
z_resample = norm.rvs(loc = 0, scale = 1, size = B)
# you could also resample x1,...,xn
# iid~ N(mu0, sigma) & compute z

# count as or more extreme events
as_or_more_extreme = 0
for i in range(B):
   as_or_more_extreme += math.fabs(z_resample[i])\
       >= math.fabs(z_0)

print "Observed Z statistic is", z_0
print "Estimate p-value from", B, "simulations:",\
   as_or_more_extreme/float(B)
```

Resampling – Monte Carlo Simulation

(General) Algorithm: Mimic the randomness/uncertainty of the random experiment using a computer.

- **Input**: the observed data $\boldsymbol{x} = (x_1, \ldots, x_n)$, a large number $B \in \mathbb{Z}^+$ for the number of times to repeat, and a model H_0 (constructed and confirmed with a biologist).
- Loop B times: at iteration i
  - Generate a **simulated** data set $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$
  - Compute and store the test statistic: $T^{(i)} = T(\boldsymbol{x}^{(i)})$.
  - (If your simulator directly simulates the test statistic rather than some upstream data, then you obtain $T^{(i)}$ in one step.)
- Compute the *observed test statistic*: $t_0 = T(\boldsymbol{x})$.
- **Output:** Compute the p-value as the proportion of simulation samples where $T^{(i)}$ is as or more extreme (shows more signal) than the observed test statistic t_0.

Review – Hypothesis Testing

1. From a **vague null hypothesis**, identify a test statistic T that is sensitive to the truth of the null hypothesis.
2. Work the hypothesis into a **specific null hypothesis** H_0 that is a **model** for the scientific experiment and can simulate test statistics T. You probably need to iterate steps 1-2 before finalizing your test statistic T and specific null hypothesis H_0.
3. Compute the ***p*-value**, or probability of observing a test statistic T as or more extreme than the observed test statistic t_0 when H_0 is true using statistical theory or simulation. Make a decision, if necessary, reporting the chance of making an error if you reject H_0. Otherwise, just report the p-value as your evidence for/against H_0.

Important Concepts

- variation and noise in samples. Why do data in samples vary?
- random experiment, sample space, outcome, probability. Can you identify the these components of a “random experiment” in the elements of a scientific experiment?
- population, sample, statistical inference. What kind of conclusions can you draw from a statistical inference?
- random variable, sampling distribution. What are the random variable function’s range and domain? Can you identify one example of a sampling distribution of a famous test statistic?
- test statistic. What is one thing a test statistic *must* accomplish to be useful for inference?
- resampling. How can resampling help you assess whether a detected signal is significant?
- *p*-values. What does a *p*-value tell you?