Hypothesis Testing

- Έστω https://twitter.com/meganinlisbon/status/1101870079858409478 (https://twitter.com/meganinlisbon/status/1101870079858409478)

```
I presented the math for this at the #cosyne19 diversity lunch today.

Success rates for first authors with known gender:
Female: 83/264 accepted = 31.4%
Male: 255/677 accepted = 37.7%

37.7/31.4 = a 20% higher success rate for men
```

Υπάρχει όντως προκατάληψη εναντίων των γυναικών σε αυτό το συνέδριο;

In [7]:
```
female_yes = 83
female_total = 264

male_yes = 255
male_total = 677

total = female_total + male_total
accepted = female_yes + male_yes
failed = total - accepted

print ('total papers:', total)
print ('Accepted papers:', accepted)
print ('Failed papers::', fail)

female_success_rate = female_yes/female_total
male_success_rate = male_yes/male_total

diff = male_success_rate - female_success_rate
diff
```

```
total papers: 941
Accepted papers: 338
Failed papers:: 603
```

Out[7]:
```
0.06226780358981243
```

Η διαφορά μεταξύ του acceptance rate (ποσοστό των papers που έγιναν δεκτά) μεταξύ των ανδρών και γυναικών είναι 0.06226780358981243 . Άραγε αυτό έγινε κατά τύχη, ή λόγω προκατάληψης;

Αρχικά κοιτάμε να δούμε ποια είναι η πιθανότητα να έγινε κατά τύχη. Για να το κάνουμε αυτό πρέπει να βρούμε το εξής:

Βάζουμε σε ένα κουτί 338 μαύρες μπάλες (failed papers) και 603 άσπρες μπάλες (accepted papers). Στη συνέχεια ανακατέβουμε το κουτί και παίρνουμε 264 τυχαίες μπάλες (female papers) και τις βάζουμε στο δοχείο A και τις υπόλοιπες 677 μπάλες (male papers) τις βάζουμε στο δοχείο B. Μετά μετράμε το ποσοστό των μπαλών που είναι άσπρες στο κουτί A και το ποσοστό των μπαλών που είναι άσπρες στο κουτί B. Τέλος υπολογίζουμε τη διαφορά αυτών των ποσοστών και ελέγχουμε αν αυτή η διαφορά είναι μεγαλύτερη ή ίση με αυτή που παρατηρήσαμε..

Δηλαδή βρίσκουμε τη πιθανότητα αυτή η "ανισότητα" που βρήκαμε να ήταν τυχαία αν υποθέσουμε ότι ΔΕΝ υπάρχει προκατάληψη

Ας το κάνουμε αυτό σε python:

In [30]:
```
import random

def random_papers():

    box = [True] * accepted + [False] * failed
    random.shuffle(box)

    female_papers = box[:female_total]
    male_papers = box[female_total:]

    female_acceptance_rate = sum(female_papers)/female_total
    male_acceptance_rate = sum(male_papers)/male_total

    return (male_acceptance_rate - female_acceptance_rate) > diff
```

Ας κάνουμε 100.000 πειράματα:

In [35]:
```
tries = 100_000
sum (random_papers() for x in range(tries))/tries
```

Out[35]:
```
0.0326
```

Το αποτέλεσμα είναι 0.0326. Το οποίο μπορεί να "ερμηνευθεί" ως εξής: Η πιθανότητα αυτή η διαφορά που υπολογίσαμε (0.06226780358981243) να μην οφείλεται σε κάποια προκατάληψη αλλά.. στη τύχη είναι μικρότερη από 1 στις 20. Ένα από τα πιο συνηθισμένα όρια που θέτουμε για να πούμε ότι "κάτι παίζει" και ότι αυτό που μετρήσαμε δεν είναι τυχαίο είναι το 0.05 (1/20).

Άρα βγάζουμε το συμπέρασμα ότι όντως υπήρχε προκατάληψη σε αυτό το συνέδριο.

Ένας στατιστικός θα το υπολόγιζε με αυτό τον τρόπο:

In [36]:
```
import numpy as np
import scipy.stats.distributions as dist

total_accepted_rate = (female_yes + male_yes) / total

variance = total_accepted_rate * (1 - total_accepted_rate)

standard_error = np.sqrt(variance * ((1 / female_total) + (1 / male
_total)))

best_estimate = male_success_rate - female_success_rate

hypothesized_estimate = 0.0

test_stat = (best_estimate-hypothesized_estimate) / standard_error

dist.norm.cdf(-np.abs(test_stat))
```

Out[36]:
```
0.03683152398331601
```

Το ποιον τρόπο θα επιλέξετε είναι δικό σας θέμα! Αλλά: ο πρώτος (ονομάζεται και permutation test: https://en.wikipedia.org/wiki/Resampling_(statistics (https://en.wikipedia.org/wiki/Resampling_(statistics))) είναι πιο ευκολονόητος και διαισθητικός για αυτούς που έχουν εμπειρία στον προγραμματισμό! Ο 2ος βέβαια είναι πιο γρήγορος και πιο ακριβής (αν το κάνετε σωστά). Μία μέση λύση είναι να κάνετε το πρώτο για να έχετε μία ιδέα το που κυμαίνεται το p-value και μετά να το επιβεβαιώσετε με τον 2ο τρόπο.

Παράδειγμα 2

Αυτό το παράδειγμα είναι "κλεμμένο" από εδώ: https://www.youtube.com/watch?v=5Dnw46eC-0o (https://www.youtube.com/watch?v=5Dnw46eC-0o) **Η παρακολούθηση αυτού του βίντεο είναι υποχρεωτική!**

Σε ένα paper (https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0009546) οι συγγραφείς εξέτασαν την υπόθεση: "Η κατανάλωση μπύρας αυξάνει την ελκυστικότητά μας στα κουνούπια που προκαλούν ελονοσία". Για να το ελέγξουν, χωρίσανε μία ομάδα από ανθρώπους σε δύο γκρουπ. Στη 1η δώσανε να πιούν μπύρα και στη δεύτερη νερό. Στη συνέχεια για κάθε άνθρωπο ελέγξαν το πλήθος από κουνούπια που προσέλκησε. Τα δεδομένα είναι:

In [37]:
```
beer = [
    29, 19, 20,
    20, 23, 17,
    21, 24, 31,
    26, 28, 20,
    27, 19, 25,
    31, 24, 28,
    24, 29, 21,
    21, 18, 27,
    20
]

water = [
    21, 19, 13,
    22, 15, 22,
    15, 22, 20,
    12, 24, 24,
    21, 19, 18,
    16, 23, 20,
]
```

Ας ελέγξουμε για αρχή τη διαφορά στον μέσο όρο:

In [38]:
```
beer_average = sum(beer)/len(beer)
water_average = sum(water)/len(water)
diff = beer_average - water_average
diff
```

Out[38]:
```
4.457777777777778
```

Ποια είναι η πιθανότητα αυτή η διαφορά να προήλθε κατά τύχη; Κάνουμε ότι κάναμε και πριν. Υποθέτουμε ότι είτε πιει κάποιος μπύρα είτε νερό θα συγκεντρώσει το ίδιο πλήθος από κουνούπια. Αρα αν πάρουμε τυχάια `len(beer)` άτομα θα έχουν τον ίδιο μέσο όρο κουνουπιών με τους υπόλοιπους `len(water)` . Επίσης υποθέτουμε ότι αυτή η διαφορά πυ μετρήσαμε (4.458) συμβαίνει "πολύ συχνά" (δηλαδή ποιο συχνά από 1/20) κατά τύχη:

In [52]:
```
def random_mosquitos():
    box = beer + water
    random.shuffle(box)
    random_beer = box[:len(beer)]
    random_water = box[len(beer):]

    random_average_beer = sum(random_beer) / len(random_beer)
    random_average_water = sum(random_water) / len(random_water)

    return (random_average_beer - random_average_water) >= diff
```

In [54]:
```
tries = 100_000
sum (random_mosquitos() for x in range(tries)) / tries
```

Out[54]:
```
0.00042
```

Ας το υπολογίσουμε τώρα με την αναλυτική μέθοδο:

In [51]:
```
from scipy.stats import ttest_ind
ttest_ind(beer, water).pvalue/2
```

Out[51]:
```
0.0004205775278799323
```

Παράδειγμα 3

Ίσως το κυριότερο ερώτημα στις γενετικές έρευνες είναι: "υπάρχει συσχετισμός μεταξύ ενός αλληλόμορφου και στην εμφάνιση μιας ασθένειας"; Με την σημερινή τεχνογία μπορούμε να πάρουμε τον γονότυπο ενώς μεγάλου μέρους τον γονοδιόματος από έναν άνθρωπο. Αυτή η διαδικασία ονομάζεται "γονοτύπηση". Αν κάνουμε αυτή τη διαδικασία για ένα μεγάλο πλήθος από ανθρώπους, κάποιους από τους οποίους εμφανίζουν μία ασθένεια, τότε μπορούμε να απαντήσουμε στο παραπάνω ερώτημα. Οι έρευνες αυτού του είδους ονομάζονται και GWAS: Genome Wide Association Studies (https://en.wikipedia.org/wiki/Genome-wide_association_study). Ας δούμε λοιπόν πως μπορούμε να προσεγγίσουμε αυτό το ερώτημα. Για αρχή ας υποθέσουμε ότι έχουμε 10 ανθρώπους. 5 από αυτούς εμφανίζουν μία ασθένεια η οποία πιστεύουμε ότι έχει γενετική βάση και 5 δεν την εμφανίζουν. Ας πάρουμε τον γονότυπό τους σε μία περιοχή του γονιδώματος:

Δείγμα	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
Γονότυπος	A/A	A/a	A/A	A/A	A/A	A/A	A/a	A/a	a/a	a/a
Ασθένεια	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ

Και σε python:

In [55]:
```
genotypes = [('A','A'), ('A','a'), ('A','A'), ('A','A'), ('A','A'),
             ('A','A'), ('A','a'), ('A','a'), ('a','a'), ('a','a')]
phenotypes = ['NO', 'NO', 'NO', 'NO', 'NO', 'YES', 'YES', 'YES', 'Y
ES', 'YES']
```

Για αρχή φτιάχνουμε έναν πίνακα που δείχνει πόσα αλληλόμορφα υπάρχουν σε ασθενείς και μη ασθενείς. Αυτός ο πίνακας ονομάζεται contingency table (https://en.wikipedia.org/wiki/Contingency_table).

In [56]:
```
import pandas as pd

def create_contingency_table(genotypes, phenotypes):
    contingency = {'Disease': ['NO', 'YES', 'total']}
    for allele in ('A', 'a'):
        contingency[allele] = []
        total = 0
        for phenotype in ('NO', 'YES'):
            c = sum(g.count(allele) for g,p in zip(genotypes, pheno
types) if p==phenotype)
            contingency[allele].append(c)
            total += c
        contingency[allele].append(total)

    contingency = pd.DataFrame(contingency)
    contingency = contingency.set_index('Disease')
    contingency['total'] = contingency.apply(sum, axis=1)

    return contingency

contingency = create_contingency_table(genotypes, phenotypes)

contingency
```

Out[56]:

Disease	A	a	total
NO	9	1	10
YES	4	6	10
total	13	7	20

Φτιάχνουμε τώρα έναν άλλο πίνακα που περιέχει την αναμενόμενες τιμές του πινακα αν υποθέταμε ότι ο γονότυπος αυτός δεν έχει κάποια σχέση με την εκδήλωση της ασθένειας.

Το στοιχείο του πίνακα αυτού για το allele=X και phenotype=Y ισούται με:

$$E_{X,Y} = \frac{Total_X * Total_Y}{total}$$

In [58]:
```
def create_expected_contingency_table(observed):

    contingency_e = {'Disease': ['NO', 'YES', 'total']}
    for allele in ('A', 'a'):
        contingency_e[allele] = []
        total = 0
        for phenotype in ('NO', 'YES'):
            #c = sum(g.count(allele) for g,p in zip(genotypes, phen
otypes) if p==phenotype)

            c = observed.loc[phenotype]['total'] * observed.loc['to
tal'][allele]
            c = c/observed.loc['total']['total']

            contingency_e[allele].append(c)
            total += c
        contingency_e[allele].append(total)

    contingency_e = pd.DataFrame(contingency_e)
    contingency_e = contingency_e.set_index('Disease')
    contingency_e['total'] = contingency_e.apply(sum, axis=1)

    return contingency_e


contingency_e = create_expected_contingency_table(contingency)
contingency_e
```

Out[58]:

Disease	A	a	total
NO	6.5	3.5	10.0
YES	6.5	3.5	10.0
total	13.0	7.0	20.0

Παρατηρείστε ότι αν υποθέσουμε ότι η θέση αυτή δεν συσχετίζεται με την ασθένεια, τότε η κατανομή των αλληλόμορφων είναι ίδια για τις δύο κλάσεις (NO, YES) της ασθένειας.

Τώρα μπορούμε να δούμε αν η διαφορά μεταξύ του observed contingency table (O) και του expected contingency table (E) είναι στατιστικά σημαντική. Υπολογίζουμε το χ^2:

$$\chi^2 = \sum_{p=[YES,NO]} \sum_{al=[A,a]} \frac{\left(O_{p,al} - E_{p,al}\right)^2}{E_{p,al}}$$

In [59]:
```
def get_chi_square(O,E):
    ret = 0
    for p in ('NO', 'YES'):
        for al in ('A', 'a'):
            O_p_al = O.loc[p][al]
            E_p_al = E.loc[p][al]
            ret += (O_p_al-E_p_al)**2 / E_p_al

    return ret
```

In [61]:
```
chi_square = get_chi_square(contingency, contingency_e)
print ('chi square:', chi_square)
```

```
chi square: 5.4945054945054945
```

Ποια είναι η πιθανότητα να βρούμε αυτό το chi-square (ή μεγαλύτερο) αν υποθέσουμε ότι δεν υπάρχει σχέση μεταξύ της μετάλλαξης και της ασθένειας; Απλό: φτιάχνουμε έναν τυχαίο contingency table ο οποίως κάνει αυτή ακριβώς την υπόθεση:

In [82]:
```
genotypes = [('A','A'), ('A','a'), ('A','A'), ('A','A'), ('A','A'),
             ('A','A'), ('A','a'), ('A','a'), ('a','a'), ('a','a')]
phenotypes = ['NO', 'NO', 'NO', 'NO', 'NO', 'YES', 'YES', 'YES', 'Y
ES', 'YES']

phenotypes_random = phenotypes
random.shuffle(phenotypes_random)

random_contigency_table = create_contingency_table(genotypes, pheno
types_random)
random_contigency_table
```

Out[82]:

Disease	A	a	total
NO	5	5	10
YES	8	2	10
total	13	7	20

Και υπολογίζουμε το chi-square αυτού του τυχαίου πίνακα:

In [83]:
```
random_chi_square = get_chi_square(random_contigency_table, conting
ency_e)
random_chi_square
```

Out[83]:
```
1.978021978021978
```

Παρατηρούμε ότι σε ένα τυχαίο "ανακάτωμα" των φαινότυπων, το chi-square είναι μικρότερο από αυτό που παρατηρήσαμε (1.978<5.494). Ας δούμε λοιπόν ποια είναι η πιθανότητα να βρούμε chi-suare μεγαλύτερο από αυτό που παρατηρήσαμε:

In [93]:
```
def random_trial():
    random.shuffle(phenotypes_random)

    random_contigency_table = create_contingency_table(genotypes, p
henotypes_random)
    random_chi_square = get_chi_square(random_contigency_table, con
tingency_e)

    return random_chi_square > chi_square
```

In [94]:
```
trials = 10_000
sum(random_trial() for x in range(trials))/trials
```

Out[94]:
```
0.007
```

Αν λοιπόν θέσουμε ως όριο σημαντικότητας το 0.05 τότε βγάζουμε ως συμπέρασμα ότι p_value < 0.05, άρα το αλληλόμορφο a σε αυτή τη θέση συσχετίζεται με την ανάπτυξη της ασθένειας.

Ας το υπολογίσουμε τώρα και με τον αναλυτικό τρόπο:

In [100]:
```
from scipy.stats.distributions import chi2
p_value = chi2.sf(chi_square, 1) # 1 = degrees of freedom
print ('p_value=', p_value)
```

```
p_value= 0.01907632210177841
```

In [ ]:
```

```