A Minimal/Incomplete Statistical Toolbox

Set things:

A $\subset$ B means that the existence of A implies B $A^c = 1 - A$

$A = (A \cap B^c) \cup (A \cap B)$

$P(A \cap B^c) = P(A) - P(A \cap B)$

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$ (recursively applicable, last term is 0 if A & B are mutually exclusive)

$P(A|B) = P(A \cap B)/P(B)$

Independence: $P(A|B) = P(A); P(A \cap B) = P(A)P(B)$

Baye's Theorem: $P(B_j|A) = \frac{P(B_j)P(A|B_j)}{\sum_{i=1}^{k} P(B_i)P(A|B_i)}$

Cominations/Permutations:

$${}_nC_r = \frac{n!}{r!(n-r)!};\ {}_nP_r = \binom{n}{r} = \frac{n!}{(n-r)!}$$

Standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

$$\sigma = \sqrt{\sum_{i=1}^{N} p_i(x_i - \mu)^2}, \mu = \sum_{i=1}^{N} p_i x_i$$

$$\sigma = \sqrt{int_x(x - \mu)^2 p(x)dx}, \mu = \int_X xp(x)dx$$

Error Propagation

f = A+B	$\sigma_f^2 = \sigma_A^2 + \sigma_B^2$
f = aA	$\sigma_f^2 = a^2\sigma_A^2$
f = aA $\pm$ bB	$\sigma_f^2 = a^2\sigma_A^2 + b^2\sigma_B^2 \pm 2ab\sigma_{AB}$
f = AB	$f^2((\frac{\sigma_A}{A})^2 + (\frac{\sigma_A}{A})^2 + 2\frac{\sigma_{AB}}{AB})$
f = aA^b	$\sigma_f^2 = (abA^{b-1}\sigma_A)^2 = (\frac{fb\sigma_A}{A})^2$
f = a $log_b(cA)$	$\sigma_f^2 = (a\frac{\sigma_A}{Aln(b)})^2$
f = a^{bA}	$\sigma_f^2 = f^2(bln(a)\sigma_A)^2$

Normal Distribution

$$Y = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2} = \frac{1}{\sqrt{2\pi}} e^{-0.5*Z^2}$$

Incidentally, this is can also be expressed as a sepcial case of the Sérsic function: $\Sigma(r) = \Sigma_e exp(-\kappa((\frac{r}{r_e})^{1/n} - 1))$ (including diff between 2 means and sigmas, as well as z-score and hypothesis testing)

Binomial Distribution

probability of success, failure = p, q. (q = 1-p)

For n trials, x successes: $\binom{n}{x} p^x q^{n-x} = \frac{n!}{(n-x)!} p^x q^{n-x}$

$\mu_p = n \cdot p$

$\sigma = \sqrt{n \cdot p \cdot q} = \sqrt{n \cdot p(1-p)}$

Alternatively, finding a distribution given P successes and Q failures in N trials:

$\sigma = \sqrt{PQ/N}; \mu = P$ (p = P/N; q = Q/N)

Multiple distributions interacting:

$\mu_{a-b} = \mu_a - \mu_b$ $\sigma_{a-b} = \sqrt{\sigma_a^2 + \sigma_b^2}$

(including diff between 2 means and sigmas, as well as z-score and hypothesis testing)

Binomial test: find the probability of getting P or more successes, given a probability of p. (Just sum up the probabilities, treat as a p-value. This is an exact test, though for large sample sizes χ^2 will also work. One-tailed is obvious enough, though two-tailed is more subtle. Do you want the total deviation to be greater or less than the expected value? Do you want the deviation in both directions to be as likely or less likely than the expected value?) scipy.stats.binom_test(P, N, p, alternative='greater') scipy.stats.binom_test(P, N, p, alternative='two-sided')

Fisher's exact test – for finding correlations between 2 binomial variables. (need to include calculation example, 1 vs 2 tailed) You can arrange your data into a 2x2 grid, and want to know if there is a correlation between the amount in each category in your 2 different samples. There's a trick with a hypergeometric distribution – you only need to consider cases with the name marginal sums. (Marginal sums: rows, columns, total.)

array 1	array 2
A	B
C	D

A+B+C+D = N $p = \frac{(\frac{A+B}{A})(\frac{C+D}{C})}{(\frac{N}{A+C})} = \frac{(A+B)!(C+D)!(A+C)!(B+D)!}{A!B!C!D!N!}$ Significance requires finding all p-values more extreme than this. (obvious enough for one tailed, but sutble for two tailed) scipy.stats.fisher_exact(array$_1$, array$_2$)

Poisson Distribution

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

x = 0, 1, 2, ...; $\mu = \lambda = \sigma^2$

This is self-similiar, so if you have an occurance rate over one timescale you can easily multiply to get occurance probabilities over another.

Kolmovogorov-Smirnov Test 2 functions, function plus point collection, or 2 point collections. Last is most useful. from scipy import stats stats.ks_2samp(array$_1$, array$_2$)

(students T-test) TBD (pearsons chi-squared test) TBD Statistical tests in general follow a 2-step process: A test statistic is calculated, using whatever parameters are considered relevant. This statistic is then compared with the distribution of that statistic given the parameters under the null hypothesis (eg: same means, same variances, etc).

F-test compare 2 fits to determine which is better. Or rather, given their Chi-squared (or similiar sum of squares) values, number of parameters (or degrees of freedom), etc, the probability of one fit being better than another for certain assumptions. $F-statistic == frac(SS_1 - SS_2)$ 1 - p-value == scipy.stats.f.cdf(F-statistic, degrees of freedom for fit 1, degrees of freedom for fit 2) This use gives a 1-tailed test. To avoid the 1-p-value annoyance, consider sf instead of cdf. (A high F-statistic points to low p-value and vice-versa. Note that scipy's f-test gives you the probability that the null hypothesis can be rejected, not accepted!)

Degrees of Freedom show up fairly regularly in different statistical tests. They often correspond to number of parameters (or data points) minus one. Or data - parameters - 1.