Some Tests Based on the Binomial Distribution (Conover Chapter Three)

STAT 345-01: Nonparametric Statistics [*]

Fall Semester 2018

Contents

[*]Copyright 2018, John T. Whelan, and all that

Thursday 6 September August 2018 – Read Section 3.1 of Conover; refer to Chapter 2 of Hollander

1 Tests for Binomial Proportion

An important problem in inference is the modelling of a **binomial experiment**, a series of independent identically-distributed observations X_i which each have two possible outcomes, labelled "success" ($X_i = 1$) and "failures" ($X_i = 0$). The invididual observations are called **Bernoulli trials**, with probability distribution $P(X_i = 1) = p$, $P(X_i = 0) = 1 - p$. This is sometimes referred to as "sampling with replacement" since it can be thought of as drawing an item from a population in which a proportion p are in the "success" category, and then putting the item back into the population so that the proportion doesn't change. In the limit of a large population, the opposite situation of "sampling with replacement" can be approximated as a binomial experiment, since the removal of up to n items won't change the population proportion much. You've probably learned a bit about binomial experiments in basic statistics, but they're often included in the field of nonparametric statistics because the model requires minimal assumptions.

The test statistic $Y = \sum_{i=1}^{n} X_i$ is a binomial random variable with n trials and probability p of success on each trial. It has a discrete sampling distribution

$$p(y) = \frac{n!}{(n-y)!y!} p^y (1-p)^{n-y} \qquad y = 0, 1, \ldots, n \tag{1.1}$$

Since the binomial distribution has $E(Y) = np$ and $\text{Var}(Y) = np(1-p) \equiv npq$, for large n, when the Central Limit Theorem kicks in, the statistic

$$Z = \frac{Y - np}{\sqrt{np(1-p)}} \tag{1.2}$$

is approximately standard normal, which simplifies most of the constructions.

1.1 Hypothesis Tests

There are three typical hypothesis tests for the proportion in a binomial experiment, all based around a hypothesized null value p^* for the parameter p:

1. An **upper-tailed test** (one-tailed), where the null hypothesis is H_0: $p = p^*$ (or $p \leq p^*$) and the alternative hypothesis is H_1: $p > p^*$.
2. A **lower-tailed test** (one-tailed), where the null hypothesis is H_0: $p = p^*$ (or $p \geq p^*$) and the alternative hypothesis is H_1: $p < p^*$.
3. A **two-tailed test**, where the null hypothesis is H_0: $p = p^*$ and the alternative hypothesis is H_1: $p \neq p^*$.

1.1.1 Large Samples

When n is large (typically np^* and $n(1-p^*)$ both more than around 10, we can use the normal approximation and define the test statistic

$$Z = \frac{Y - np^*}{\sqrt{np^*(1-p^*)}} \tag{1.3}$$

which is standard normal if H_0 is true. That means a test with significance α can be obtained by rejecting H_0 if $Z > z_{1-\alpha}$ for an upper-tailed test, if $Z < -z_{1-\alpha}$ for a lower-tailed test, and $|Z| < z_{1-\alpha/2}$ for a two-tailed test.

Similarly, the p-value for a one-tailed test is the probability that a standard normal random variable will exceed the actual observed value of $z = \frac{y-np^*}{\sqrt{np^*(1-p^*)}}$:

$$P = P(Z \geq z | p = p^*) = 1 - \Phi\left(\frac{y - np^*}{\sqrt{np^*(1-p^*)}}\right) \tag{1.4}$$

and for a two-tailed test, it's the probability that Z will be farther from zero than z, i.e.,

$$P = P(|Z| \geq |z| \, | p = p^*) = 1 - 2\Phi\left(\left|\frac{y - np^*}{\sqrt{np^*(1-p^*)}}\right|\right) \tag{1.5}$$

1.1.2 General One-Tailed

If the sample size is not large enough to use the normal approximation, we just need to use an appropriate threshold on the number of successes. For instance, if we define a test which rejects H_0: $p = p^*$ in favor of H_1: $p > p^*$ whenever $y > k$, this test will have significance

$$\alpha = P(Y{>}k|p{=}p^*) = 1 - P(Y{\leq}k|p{=}p^*) = 1 - B(k; n, p^*) \tag{1.6}$$

where $B(y;n,p)$ is the cdf of the binomial distribution. Conover tabulates this in Appendix Table A3, but we could also get it from `scipy.stats.binom.cdf()` or equivalently $1-B(k;n,p^*)$ from `scipy.stats.binom.sf()`:

```
from __future__ import division
import numpy as np
from scipy import stats
n = 20
pstar = 0.2
mydist = stats.binom(n,pstar)
mydist.sf(10)
k = np.arange(n+1)
mydist.sf(k)
mydist.sf(6)
mydist.sf(7)
```

Note that this means it's not possible to get a test with an arbitrary significance α. For instance, if $n = 20$ and $p^* = 0.2$, a test which rejects H_0 if $y > 7$ has $\alpha \approx 0.032$ and one which rejects H_0 if $y > 6$ has $\alpha \approx 0.087$, so we cannot find a test with a significance of exactly 0.05.

For the P-value of an upper one-sided hypothesis test, we just take

$$P = P(Y \geq y|p=p^*) = 1 - B(y-1;n,p^*) \tag{1.7}$$

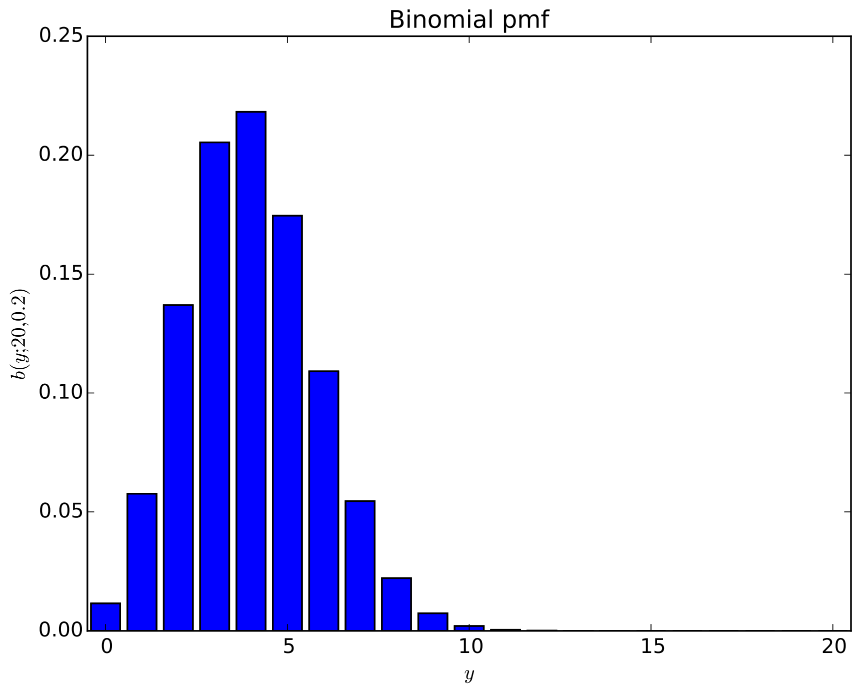
Again, only certain specific P-values are possible.

1.1.3 General Two-Tailed

Things get interesting for two-tailed tests. Not only do we have the discreteness of the binomial distribution, but it is also in general asymmetric, which means we have to give some thought to how much probability to put on each tail of the distribution. For instance, suppose $n = 20$ and $p^* = 0.2$. Then

```
bar(k,mydist.pmf(k),align='center');
xlim(-0.5,n+0.5)
xlabel(r'$y$');
ylabel(r'$b(y;%d,%.1f)$' % (n,pstar));
title('Binomial pmf');
```

gives us the pmf



Conover somewhat unhelpfully tells us that if we want to devise a test with confidence α, we should reject if $y \leq t_1$ or $y > t_2$ where $P(Y \leq t_1|p=p^*) = \alpha_1$ and $P(Y > t_2|p=p^*) = \alpha_2$, where α_1 and α_2 are each "approximately half" of α, and then doesn't provide a small-n example. Let's work through the details.

First, consider the problem of finding the two-tailed P-value when $y = 8$. We can find that $P(Y \geq 8) = P(Y > 7.5) \approx 0.0321$ with

```
mydist.sf(7.5)
```

You might naïvely think that the two-tailed P-value should be twice this, but we're asking for the null probability of Y having a value at least as extreme as 8, which we can reasonably take to mean a value which is no more likely than 8, i.e.,

$$P = \sum_{p(y)\le p(8)} p(y) \tag{1.8}$$

```
mypmf = mydist.pmf(k)
unlikely = mypmf <= mypmf[8]
print(unlikely)
k[unlikely]
mypmf[0]
mypmf[unlikely].sum()
```

So we see that the set of values at least as unlikely than 8 is $Y \le 0$ or $Y \ge 8$. But because $P(Y \le 0) = 0.0116$, the P-value is 0.0437, which is less than twice the upper tail probability. Fortunately, this behavior is exactly what's coded in Python

```
stats.binom_test(8,n,pstar)
```

or R (as `binom.test(8,20,0.2)`).

To get a test with significance approximately 0.05, we just need to look at the possible P-values. First we sort the possible numbers of successes in decreasing order of null probability:

```
np.argsort(mydist.pmf(k))[::-1]
```

The next least probable value after 8 is 7:

```
unlikely7 = mypmf <= mypmf[7]
mypmf[unlikely7].sum()
```

The P-value for $y = 7$ is 0.0982. So for a test which rejects if $Y \le 0$ or $Y \ge 8$, the significance is 0.0437; for one which rejects if $Y \le 0$ or $Y \ge 7$, it is 0.0982.

1.2 Confidence Intervals

Rather than proposing some fixed proportion p^* and defining hypothesis tests or P-values, we can instead use the observed y to construct a confidence interval on p. A confidence interval at confidence level $1-\alpha$ should be constructed in such a way as to have a probability $1-\alpha$ of containing the true value of p.

If the sample size n is large enough, we can define the point estimate $\hat{p} = y/n$, which is a realization of the corresponding statistic $\hat{p} = Y/n$, and note that the Central Limit Theorem tells us the statistic $\frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$ is approximately standard normal. If we use this to construct a confidence interval, we end up having to solve a quadratic equation for p and get a somewhat complicated expression called the Wilson score interval which you may have seen in Devore. But if n is large enough, you can also assume that the statistic $\frac{\hat{p}-p}{\sqrt{\hat{p}(1-\hat{p})/n}}$ is standard normal as well, and get an approximate confidence interval on p with endpoints

$$\hat{p} \pm z_{1-\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{1.9}$$

in direct analogy with the usual normal confidence interval.

If n is not large, Conover basically punts and says to look it up in the back of the book. But in fact there are several different ways of defining small-n and intermediate-n confidence intervals for binomial proportion.[1] The one Conover uses is called the Clopper-Pearson interval. It's defined as the set of all proportions p^* such that

$$P(Y \le y | p = p^*) > \frac{\alpha}{2} \qquad \text{and} \qquad P(Y \ge y | p = p^*) > \frac{\alpha}{2} \tag{1.10}$$

[1]More information than you could possibly want is on the relevant Wikipedia page `https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval`

This would be straightforward but laborious to work out experimentally, and it would be understandable to fall back to looking it up in the table. It turns out, though, that a bit of mathematical manipulatin allows one to express the endpoints in terms of the percentiles of a beta distribution, so we can define a function that computes it on the fly even though it's not built in to scipy:

```
def ClopperPearsonCI(CL,n,x):
    tailprob = 0.5*(1.-CL)
    lower = stats.beta.ppf(tailprob,x,n-x+1)
    upper = stats.beta.isf(tailprob,x+1,n-x)
    return (lower,upper)
```

We can then test it with

```
CIlo, CIhi = ClopperPearsonCI(.95,20,8)
CIlo
CIhi
print(stats.binom(n,CIhi).cdf(8.5))
print(stats.binom(n,CIlo).sf(7.5))
```

As a final comment, note that the way we constructed this confidence interval was somewhat different than the approach used for the P-value, since we required equal tail probabilities. An interesting exercise/challenge is to define a confidence interval, given n and y, as all of the proportions for which the P-value is greater than 0.05. You can then check the coverage of this confidence interval for some true proportion p as in the homework.

Tuesday 11 September August 2018 – Read Section 3.2 of Conover; refer to Chapter 1 of Higgins

2 Inference About Population Quantiles

We now consider the first of a couple of truly nonparametric inferences which turn out to be mathematically equivalent to the binomial test. Suppose we have a sample $x_1, \ldots x_n$ which is drawn from an unspecified distribution. We wish to make a statement about the p^* quantile x_{p^*}, for a specified p^*, given the data $\{x_i\}$.

2.1 Hypothesis Tests

For a hypothesis test, we wish to evaluate the a hypothesis H_0 which specifies some value x^* for the p^* quantile, H_0: $x_{p^*} = x^*$. As usual, the alternative hypothesis can be upper-tailed (H_1: $x_{p^*} > x^*$), lower-tailed (H_1: $x_{p^*} < x^*$) or two-tailed (H_1: $x_{p^*} \neq x^*$).

If the sampling distribution is assumed to be a continuous distribution $f(x)$, so that there's zero probability that X will be exactly x^*, the statement $x_{p^*} = x^*$ is equivalent to

$$P(X \leq x^*) = p^* \tag{2.1}$$

We can thus construct the test statistic T which is the number of values in the random sample which are below x^*; under the null hypothesis, this will be a binomial random variable with n trials and probability of success p. We then apply the binomial methods from last week. For instance, let's take a random sample of size $n = 40$ and do a two-tailed test of the hypothesis that the 60th percentile of the sampling distribution is 1.

```
from __future__ import division
import numpy as np
from scipy import stats
n = 40
np.random.seed(1)
xi = stats.invgamma(2).rvs(size=n)
xi
p = 0.6
xpstar = 1.
teststat = np.sum(xi <= xpstar)
teststat
mu = n*p; mu
sigma = np.sqrt(n*p*(1-p)); sigma
z = (teststat-mu)/sigma; z
pvalz = 2*stats.norm.sf(z); pval
nulldist = stats.binom(n,p)
k = np.arange(n+1)
nullpmf = nulldist.pmf(k)
unlikely = (nullpmf <= nullpmf[teststat])
k[unlikely]
pval = np.sum(nullpmf[unlikely]); pval
```

For the one-sided case, things proceed in a relatively straightforward way, but one-tailed tests are in the opposite direction for the binomial and quantile tests. E.g., if the alternative hypothesis is $x_{p^*} > x^*$, that is equivalent to $P(X \le x^*) < p^*$.

If we consider the possiblity of a discrete sampling distribution, we have to worry about the possibility of one of the data values equalling exactly x^*. That means there are two possibilities for our "obvious" test statistic: T_1 which is the number of data values with $x_i \le x^*$ and T_2 is the number with $x_i < x^*$. Note that this means $T_1 \ge T_2$

If we think about the null hypothesis H_0: $x_{p^*} = x^*$ in light of the definition of the quantile x_p:

$$P(X < x^*) \le p^* \qquad \text{and} \qquad P(X > x^*) \le 1 - p^* \tag{2.2}$$

which can also be written

$$P(X \le x^*) \ge p^* \qquad \text{and} \qquad P(X < x^*) \le p^* \tag{2.3}$$

we see it's actually a composite hypothesis. So large values of T_1 (the number of values for which $X_i \le x^*$) are consistent with H_0, as are small values of T_2 (the number of values for which $X_i < x^*$). So we need to reject H_0 if t_1 is too small or t_2 is too large. In each case, the null distribution will be $\text{Bin}(n, p^*)$. It might seem odd to have the same null distribution for two different statistics, but this comes from the definition of significance for composite null hypothesis, i.e., you take the form of the null hypothesis which gives the highest false alarm probability, which in this case is that each binomial probability is p^*.

```
n = 20
np.random.seed(1)
xi = stats.binom(4,0.4).rvs(size=n)
xi
p = 0.5
xpstar = 2
teststat1 = np.sum(xi <= xpstar); teststat1
teststat2 = np.sum(xi < xpstar); teststat2
stats.binom(n,p).sf(teststat2-0.5)
nulldist = stats.binom(n,p)
k = np.arange(n+1)
nullpmf = nulldist.pmf(k)
unlikely = (nullpmf <= nullpmf[teststat2])
np.sum(nullpmf[unlikely])
2.*stats.binom(n,p).sf(teststat2-0.5)
```

For a one-sided test, e.g., H_0: $x_{p^*} = x^*$ versus H_1: $x_{p^*} < x^*$, you use the appropriate test statistic for the appropriate direction. So in this case H_1 is

$$H_1 : P(X < x^*) > p^* \tag{2.4}$$

and we want to reject H_0 if t_2 is too high.

Thursday 13 September August 2018 – Read Section 3.4 of Conover; refer to Section 3.4 of Hollander

2.2 Confidence Intervals

To consider a confidence interval on a population quantile, it's helpful to first think about a point estimator for that quantile. For instance, an obvious estimator for the population median is the sample median. Sample quantiles can also be described in terms of **order statistics**. The kth order statistic of a sample $\{X_i\}$ is written $X^{(k)}$, and it's simply the kth value in the sorted list of the sample. So for a particular realization, if $x_1 = 1.3$, $x_2 = -2.1$, $x_3 = 3.4$, and $x_4 = 0.7$, we have $x^{(1)} = -2.1$, $x^{(2)} = 0.7$, $x^{(3)} = 1.3$, and $x^{(4)} = 3.4$. For a random sample, each order statistic is a random variable which depends on the whole sample.

The endpoints of a confidence interval on the quantile x_{p^*} will be order statistics; we define a $1 - \alpha$ confidence interval by

$$P(X^{(r)} \leq x_{p^*} \leq X^{(s)}) = 1 - \alpha \tag{2.5}$$

We can choose the integers r and s by considering the meanings of the inequalities $X^{(r)} \leq x_{p^*}$ and $x_{p^*} \leq X^{(s)}$. Assuming for simplicity that we're dealing with a continuous distribution, so that $P(X \leq x_{p^*}) = P(X \leq x_{p^*}) = p^*$ for each point in the sample, the statement $X^{(r)} \leq x_{p^*}$ means that at least r of the values in the sample are below x_{p^*}. On the other hand, $x_{p^*} \leq X^{(s)}$ means that less than s of the points in the sample are below x_{p^*}. So if Y is a $\text{Bin}(n, p^*)$ random variable which refers to the number of points in the sample below x_{p^*}, equation (2.5) can be written

$$P(r \leq Y < s) = \sum_{i=r}^{s-1} \binom{n}{i} (p^*)^i (1 - p^*)^{n-i} = 1 - \alpha \tag{2.6}$$

So we find the `r` and `s` which satisfy this (using statistical tables or `stats.binom.interval()`), and then pick out the corresponding order statistics as the ends of the confidence interval. As usual, if np^* and $n(1 - p^*)$ are large enough, we can us the normal approximation, along with $E(Y) = np^*$ and $\text{Var}(Y) = np^*(1 - p^*)$, and the continuity correction

$$P(r \leq Y < s) = P\left(r - \frac{1}{2} \leq Y \leq s - \frac{1}{2}\right) = 1 - \alpha \tag{2.7}$$

to write

$$r - \frac{1}{2} \approx np^* - z_{1-\alpha/2} \tag{2.8a}$$
$$s - \frac{1}{2} \approx np^* + z_{1-\alpha/2} \tag{2.8b}$$
$$\tag{2.8c}$$

```
from __future__ import division
import numpy as np
from scipy import stats
n = 40
np.random.seed(1)
xi = stats.invgamma(2).rvs(size=n)
xi
```

```
orderstats = np.sort(xi); orderstats
p = 0.6
alpha = 1. - 0.90
mydist = stats.binom(n,p)
r,sm1 = mydist.interval(1.-alpha)
r = int(r); r
s = int(sm1) + 1; s
mu = mydist.mean(); mu
sigma = mydist.std(); sigma
zcrit = stats.norm.isf(0.5*alpha); zcrit
rn = 0.5 + mu - zcrit * sigma; rn
sn = 0.5 + mu + zcrit * sigma; sn
(int(np.floor(rn)),int(np.ceil(sn)))
(r,s)
mydist.cdf(s-1) - mydist.cdf(r-1)
mydist.cdf(s-2) - mydist.cdf(r-1)
mydist.cdf(s-1) - mydist.cdf(r)
orderstats[r-1]
orderstats[s-1]
```

3 The Sign Test for Paired Data

Another test which uses binomial probabilities is the sign test, one of the oldest (and crudest) hypothesis tests. The context is **paired data**, which goes beyond the simplest sampling experiment to suppose our data consist of n pairs of values (x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n), drawn independently from the joint distribution $f(x, y)$, This means the joint distribution for the full data, seen as random vectors $\mathbf{X}, \mathbf{Y} \equiv \{X_i\}, Y_i\}$, is

$$f(\mathbf{x}, \mathbf{y}) = f(x_1, y_1)f(x_2, y_2) \cdots f(x_n, y_n) \tag{3.1}$$

The null hypothesis H_0 is that the random variable $X - Y$ is drawn from a distribution with zero median, i.e., each X_i is as likely to be larger than its associated Y_i as smaller. A standard parametric test for the difference in the *means* of X and Y is the paired t-test, based on the sample mean $\frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)$; if the distribution is symmetric–i.e., $f(x, y) = f(\mu_1 - [x - \mu_1], \mu_2 - [y - \mu_2])$–the (population) means will be the same as the (population) medians. But even then, the test can be thrown off by "outliers"; if there's a small chance that one $x_i - y_i$ is very positive or very negative, it will dominate the test statistic. So instead, the sign test basically tosses out all information about the data except whether each x_i is larger or smaller than its corresponding y_i. (This means the test is also useful when we don't have quantitative data about the size of the differences.)

If the original distribution is discrete, it's also possible that some of the differences will be neither positive nor negative, i.e., for some values of i it may turn out that $x_i = y_i$. So in principle the data will include n_+ pairs where $x_i > y_i$, n_- pairs where $x_i < y_i$, and n_0 pairs where $x_i = y_i$. There are many different approaches to how to handle these zero differences[2] but for the moment, we'll focus on the simplest one, which is to ignore them and only consider the data corresponding to non-zero differences. That means we have n_+ out of $n_\pm = n_+ + n_-$ pairs for which the difference is positive. Under the null hypothesis (in which the difference has zero median), the statistic N_+ is, for a given value of $n_\pm$,[3] a binomial $\text{Bin}(n_\pm, \frac{1}{2})$ random variable, so we can construct a binomial test to choose thresholds on n_+. If $n_\pm$ is

[2]For a summary see Coakley and Heise, "Versions of the Sign Test in the Presence of Ties", *Biometrics*, **52** 1242 (1996), available on campus as `http://www.jstor.org/stable/2532840` and off campus as `http://www.jstor.org.ezproxy.rit.edu/stable/2532840`, and more recently Bian, McAleer and Wong, "A trinomial test for paired data when there are many ties", *Mathematics and Computers in Simulation*, **81**, 1153 (2011), Available on campus as `http://dx.doi.org/10.1016/j.matcom.2010.11.002` and off campus as `http://dx.doi.org.ezproxy.rit.edu/10.1016/j.matcom.2010.11.002`

[3]In a broader sense, $N_\pm$ is also a random variable, which depends on

large enough, we can use the normal approximation, using the mean

$$E(N_+|n_\pm) = \frac{n_\pm}{2} \tag{3.2}$$

and standard deviation

$$\sqrt{\mathrm{Var}(N_+|n_\pm)} = \sqrt{\frac{n_\pm}{4}} = \frac{\sqrt{n_\pm}}{2} \tag{3.3}$$

Things are somewhat simplified by the fact that the assumed probability is $\frac{1}{2}$. In particular, the sampling distribution for N_+ is symmetric even if we are not in the large-sample regime, so e.g., the two-sided p-value is just twice the corresponding one-sided p-value.

In the numerical demo, we'll read in data from the space-separated file `notes03_signtest.dat`, available from `http://ccrg.rit.edu/~whelan/courses/2018_3fa_STAT_345/data/`

```
from __future__ import division
import numpy as np
from scipy import stats
xi, yi = np.loadtxt('notes03_signtest.dat',unpack=True)
xi
yi
n = len(xi); n
jitx = stats.norm(scale=0.05).rvs(n)
jity = stats.norm(scale=0.05).rvs(n)
plot(xi+jitx,yi+jity,'k.')
xlim(0.5,6.5)
ylim(0.5,6.5)
plot([0.5,6.5],[0.5,6.5])
nplus = np.sum(xi>yi); nplus
nminus = np.sum(xi<yi); nminus
npm = nplus + nminus; npm
stats.binom(npm,0.5).cdf(nplus)
stats.binom(npm,0.5).sf(npm-nplus-0.5)
mu = 0.5*npm; mu
sigma = 0.5*np.sqrt(npm); sigma
z = (nplus - mu)/sigma; z
stats.norm.cdf(z)
```

the number of ties N_0, but we can consider all of the relevant probabilities to be conditional upon the observed value $n_\pm$.