# HERA Memo 61: Properly Normalized $\chi^2$/DoF in Redundant-Baseline Calibration

## Josh Dillon, January 11, 2019

In order to assess the errors in redundant calibration, we need to know what we should expect when the errors are caused by pure noise.

The goal of redundant calibration is to take a set of observed visibilities $V_{ij}^{\text{obs}}$ and find a set of gains $g_i$ and unique-baseline visibilities $V_{i-j}$ that minimizes

$$\chi^2 \equiv \sum_{i>j} \frac{\left|V_{ij}^{\text{obs}} - g_i g_j^* V_{i-j}\right|^2}{\sigma_{ij}^2}$$

where $\sigma_{ij}^2$ is the per-baseline noise variance. Since $\chi^2$ is a random variable, we would like to know its mean (or equivalently the number of degrees of freedom $DoF$ that would give us a mean $\chi^2/DoF$ of 1) and its probabily distribution.

## A classic example

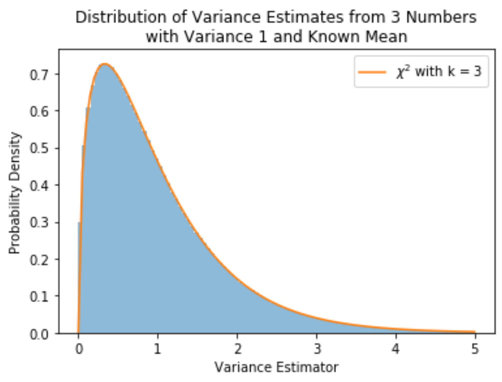
First, let's simulate a much simpler estimator. We know that the the estimates of the variance of $d$ identically and guassian-distributed random numbers with unit variance is $\chi^2$ -distributed (https://en.wikipedia.org/wiki/Chi-squared_distribution) with $k = d$ degrees of freedom Let's look at $d = 3$.

In [1]:
```
import numpy as np
from scipy import stats
from copy import deepcopy
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(21)
```

In [2]:
```
d = 3
mean = 1
data = mean + np.random.randn(d,10000000)
variance_estimators = np.sum((data - mean)**2, axis=0)/d
print 'Mean of Variance Estimators:', np.mean(variance_estimators)

x = np.linspace(0,5,100)
plt.hist(np.sum((data - mean)**2, axis=0)/d, x, density=True, alpha=.5)
plt.plot(x, stats.chi2.pdf(x*d, d)*d, label="$\chi^2$ with k = {}".format(d))
plt.title('Distribution of Variance Estimates from {} Numbers\nwith Variance 1 and Known Mean'.format(d))
plt.xlabel('Variance Estimator'); plt.ylabel('Probability Density')
plt.legend()
plt.show()
```

```
Mean of Variance Estimators: 0.9999686634312153
```

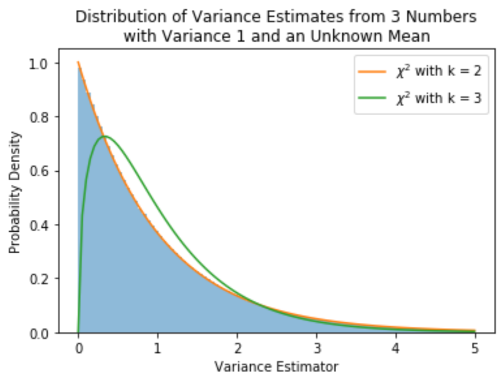


However, if we don't already know the mean and have to fit for it simultaneously, the unbiased estimator of the variance (https://en.wikipedia.org/wiki/Variance#Sample_variance) is a random variable that also follows a $\chi^2$ distribution but with one fewer degree of freedom.

```
In [3]: d = 3
        mean = 1
        data = mean + np.random.randn(d,10000000)
        variance_estimators = np.sum((data - np.mean(data, axis=0))**2, axis=0)/(d-1.)
        print 'Mean of Variance Estimators:', np.mean(variance_estimators)

        x = np.linspace(0,5,100)
        plt.hist(variance_estimators, x, density=True, alpha=.5)
        for k in [d-1, d]:
            plt.plot(x, stats.chi2.pdf(x*k, k)*k, label="$\chi^2$ with k = {}".format(k))
        plt.title('Distribution of Variance Estimates from {} Numbers\nwith Variance 1 and an Unknown Mean'.format
        (d))
        plt.xlabel('Variance Estimator'); plt.ylabel('Probability Density')
        plt.legend()
        plt.show()
```

```
Mean of Variance Estimators: 0.9996494038014742
```



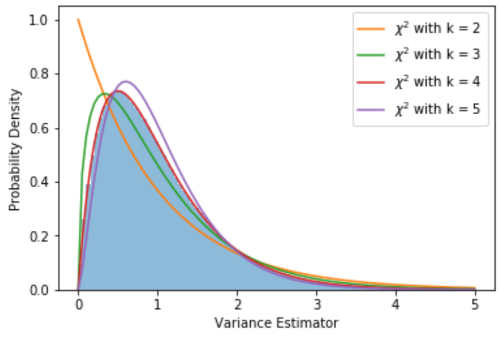
## A more complex example

Now let's consider a complex random variable with unit variance (split evenly between the real and imaginary parts, which are uncorrelated) in the case where we need to simultaneously estimate the mean.

```
In [4]: d = 3
        data = 1.0/2**.5 * np.random.randn(d,10000000) + 1.0j/2**.5 * np.random.randn(d,10000000)
        variance_estimators = np.sum(np.abs(data - np.mean(data, axis=0))**2, axis=0)/(d-1.)
        print 'Mean of Variance Estimators:', np.mean(variance_estimators)

        x = np.linspace(0,5,100)
        plt.hist(variance_estimators, x, density=True, alpha=.5)
        for k in [d-1, d, 2*d - 2, 2*d - 1]:
            plt.plot(x, stats.chi2.pdf(x*k, k)*k, label="$\chi^2$ with k = {}".format(k))
        plt.xlabel('Variance Estimator'); plt.ylabel('Probability Density')
        plt.legend()
        plt.show()
```

```
Mean of Variance Estimators: 1.000170932182834
```



While the naive extension of the variance estimator in the above case produces an unbiased estimate, the variance of that estimator has been doubled when we go from real to complex variables (consistent with this reference) (http://www.dsp-book.narod.ru/DSPMW/60.PDF). It is still $\chi^2$-distributed, but the $k$ parameter that describes the distribution is now 4 rather than 2.

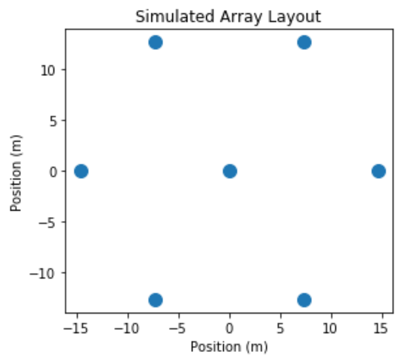
Put another way, **while we still need to subtract a degree of freedom to account for simultaneously solving for the mean when normalizing the variance estimator $DoF = (d-1) = 2$, we need to double that number $k = (2d-2) = 4$ to describe the resulting $\chi^2$ distribution.**

## Generalizing to redundant-baseline calibration

So, how does this generalize to $\chi^2/DoF$ for redundant calibration? Let's take a simple example where the number of degrees of freedom is small in order to make any differences are readily apparent.

In [5]:
```
from hera_sim.antpos import hex_array
from hera_cal.datacontainer import DataContainer
from hera_cal import redcal
```

In [6]:
```
# Set up array
antpos = hex_array(2, split_core=False, outriggers=0)
plt.scatter(np.array(antpos.values())[:,0], np.array(antpos.values())[:,1], s=100)
plt.gca().set_aspect('equal')
plt.title('Simulated Array Layout')
plt.xlabel('Position (m)'); plt.ylabel('Position (m)')
plt.show()
```



In [7]:
```
# Get redundancies and set up array
reds = redcal.get_reds(antpos)
freqs = np.linspace(100e6, 200e6, 1024, endpoint=False)
times = np.linspace(0, 600./60/60/24, 600, endpoint=False)
df = np.median(np.diff(freqs))
dt = np.median(np.diff(times)) * 3600. * 24
```

In [8]:
```
# Simulate redundant data with noise
noise_var = .01
g, tv, d = redcal.sim_red_data(reds, shape=(len(times),len(freqs)))
n = DataContainer({bl: np.sqrt(noise_var/2) * (np.random.randn(*vis.shape) + 1j * np.random.randn(*vis.sha
pe)) for bl, vis in d.items()})
noisy_data = n + d
```

In [9]:
```
# Set up autocorrelations so that the predicted noise variance is the actual simulated noise variance
for antnum in antpos.keys():
    noisy_data[(antnum, antnum, 'xx')] = np.sqrt(noise_var * dt * df)
noisy_data.freqs = deepcopy(freqs)
noisy_data.times_by_bl = {bl[0:2]: deepcopy(times) for bl in noisy_data.keys()}
```

In [10]:
```
# Perform redundant calibration
cal = redcal.redundantly_calibrate(noisy_data, reds)
```

Now the expected number of degrees of freedom, following Zheng et al. (2014) (https://arxiv.org/pdf/1405.5527.pdf), is the number of observations minus the number of gains and the number of unique visibilities. In our case, that would be $DoF = 21 - 9 - 7 = 5$. That's how our equation for $\chi^2/DoF$ is normalized in `hera_cal.redcal` (https://github.com/HERA-Team/hera_cal/blob/f96c89bdb0fa08d16c31f10a3624dbce09943b04/hera_cal/redcal.py#L950) as of the writing of this memo (though expect it to change shortly) (https://github.com/HERA-Team/hera_cal/blob/master/hera_cal/redcal.py). However we see right away that there's a problem.

In [11]:
```
print 'Mean ChiSq per "degree of freedom":', np.mean(cal['chisq']['Jxx'])
zheng_dof = len(d) - len(tv)- len(g)
print 'Mean ChiSq:', np.mean(cal['chisq']['Jxx']) * zheng_dof
```

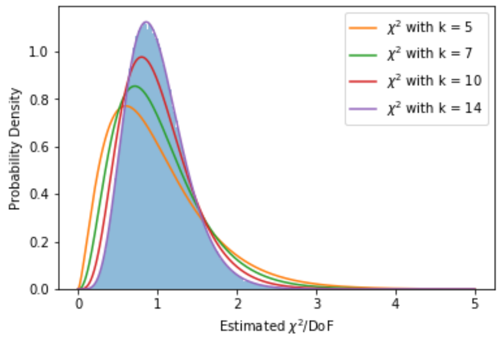
```
Mean ChiSq per "degree of freedom": 1.3969062584875176
Mean ChiSq: 6.9845312924375875
```

This suggests that the $DoF$ is actually 7, not 5. What accounts for the extra two degrees of freedom? Well, the number of parameters constrained by redundant calibration is not simply the number of gains plus the number of visibilities. It's actually fewer, because redundant calibration has 4 degeneracies (three phase and one amplitude) (https://arxiv.org/abs/1712.07212). Since we're considering complex numbers, this is equivalent to two extra $DoF$ in the denominator.

Now, what about the distribution of $\chi^2$?

In [12]:
```
correct_dof = zheng_dof + 2

x = np.linspace(0,5,500)
chisq_corrected = cal['chisq']['Jxx'] * zheng_dof / correct_dof
plt.hist(chisq_corrected.flatten(), x, density=True, alpha=.5)
for k in [zheng_dof, correct_dof, zheng_dof*2, correct_dof*2]:
    plt.plot(x, stats.chi2.pdf(x*k, k)*k, label="$\chi^2$ with k = {}".format(k))
plt.legend()
plt.xlabel('Estimated $\chi^2$/DoF'); plt.ylabel('Probability Density')
plt.show()
```



So, as we saw from the case of the distribution of complex variance estimates, the distribution of $\chi^2/DoF$ is a $\chi^2(k)$ function with $k = 2\ DoF$.

## Summary

We can summarize our findings as follows.

**The probability distribution for a given $\chi^2$ as a function of frequency and time and defined as**

$$\chi^2/DoF \equiv \sum_{i>j} \frac{\left|V_{ij}^{\text{obs}} - g_i g_j^* V_{i-j}\right|^2}{\sigma_{ij}^2}/DoF$$

**has a mean 1 when $DoF = N_{\text{obs}} - (N_{\text{ants}} + N_{\text{ubl}} - 2)$. Furthermore, that $\chi^2/DoF$ follows an analytic $\chi^2(k)$ distribution with $k = 2\ DoF$.**

**This differs from the formula presented in Equation 7 of Zheng et al. (2014) (https://arxiv.org/pdf/1405.5527.pdf), which would lead to very slight overestimates of $\chi^2$/DoF in that paper. It is difficult to check Figure 11 to see if the "perfect calibration" curve is broader than it should have been because $k$ was off by roughly a factor of 2. That part of the figure is now dubious.**

## Acknowledgements

Thanks to Adrian Liu for feedback on a draft of this memo.