Yet another method for calculating information from neural data.

P.E. Latham, S.M. Carcieri, A.L. Jacobs and S. Nirenberg

University of California at Los Angeles

52.7
SFN 2000

Mutual information, *I*, gives us a powerful tool answering the question:

<u>what aspects of a spike train are important?</u>

$$I = -\sum_{R} P(R) \log P(R) + \sum_{S} P(S) \sum_{R} P(R|S) \log P(R|S),$$

$$= H(R) - H(R|S)$$

S = stimulus,
R = response (typically spike trains),
$H(R)$ = entropy of responses,
$H(R|S)$ = conditional entropy.

For example:

S **= pictures shown to a monkey**
R **= neuronal response in V1, consisting of:**

A. ***R*** **= spike count in 300 ms window** — I **(bits) =** $\boxed{1.2} \pm \boxed{1.0}$

B. ***R*** **= precise spike times in 300 ms window** — I **(bits) =** $\boxed{3.4} \pm \boxed{2.6}$

Spike timing is important

Well, maybe not ...

The moral: to make a meaningful comparison, one must

A. **Accurately estimate entropy,**
B. **Provide error bars.**

The holy grail:

- **Given data, D, find an unbiased estimator, $\hat{H}(D)$, such that**

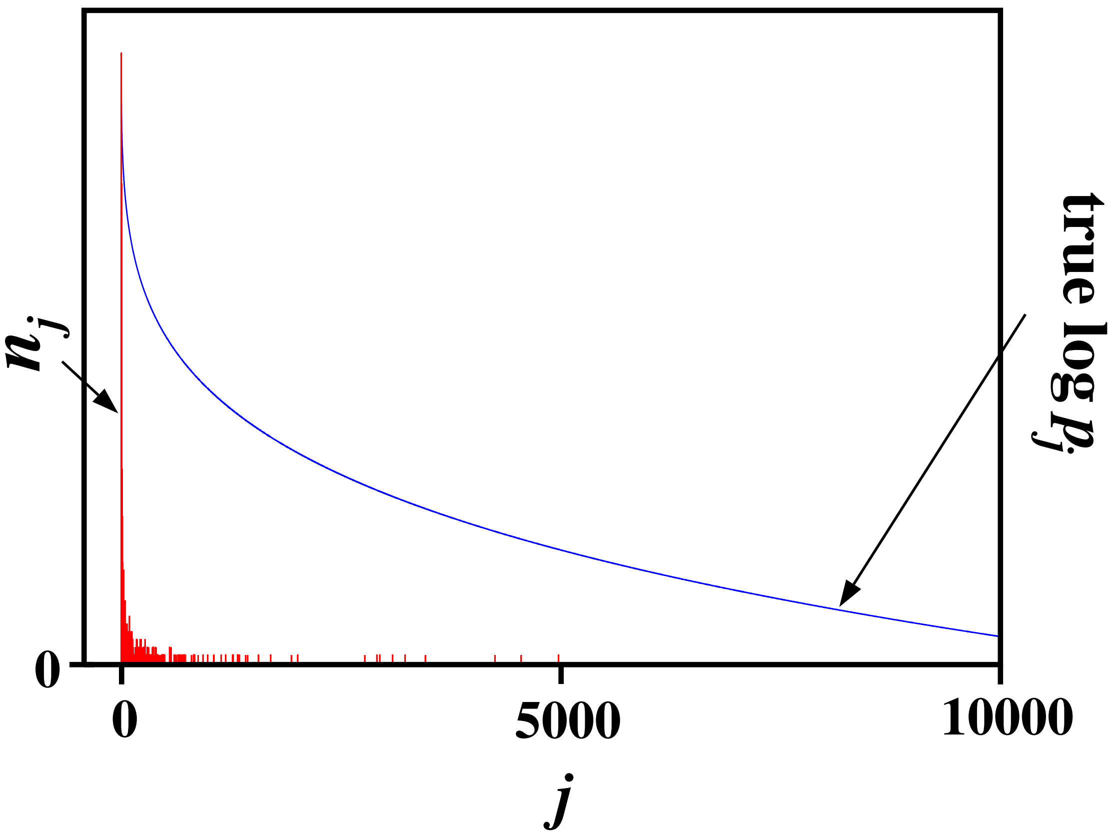
$$\langle \hat{H}(D) \rangle = H_{\mathrm{true}}$$

  where $\langle \ldots \rangle$ indicates an average over experiments.

- **Compute the variance of the estimator.**

Why is this a hard problem?

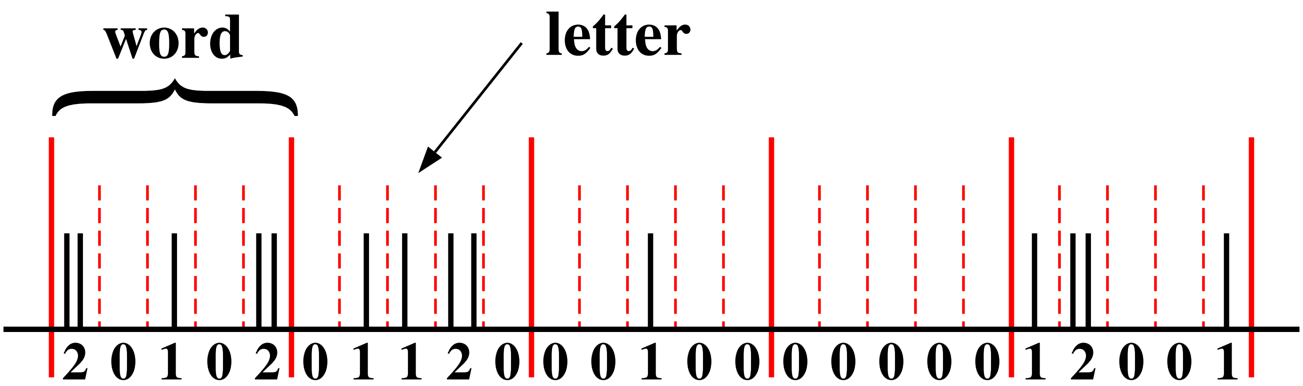
<u>The</u> <u>short</u> <u>answer:</u>
when you collect data from a distribution with long tails, you don't sample the tails.



$\hat{H}_{\text{naive}} = -\sum (n_j/N) \log(n_j/N) = 5.8$ **bits** (N=1000)

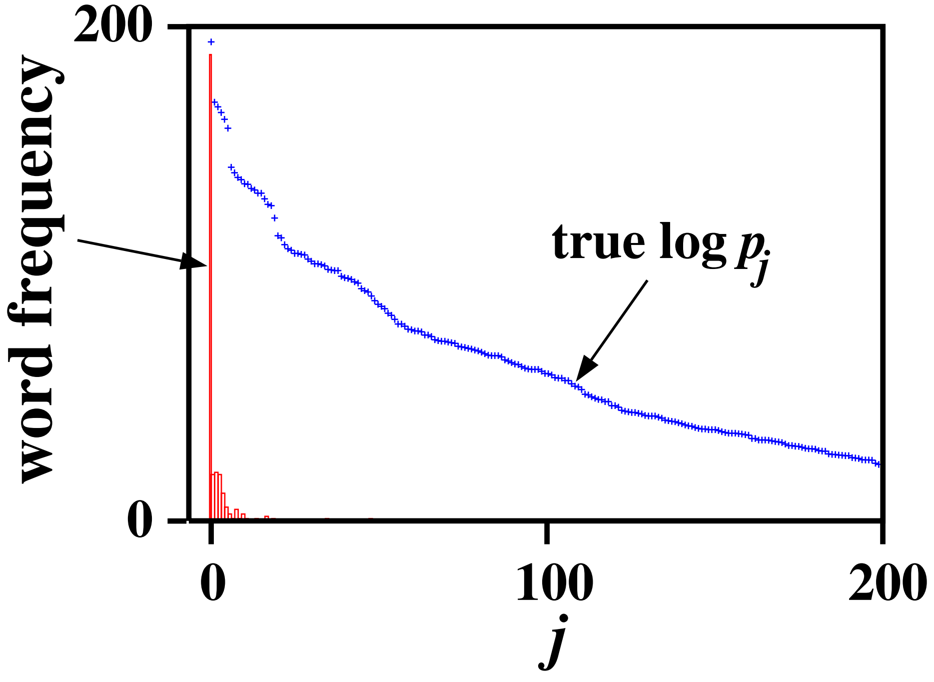
$H_{\text{true}} = -\sum p_j \log p_j = 8.6$ **bits**

An example



construct histogram:

word	frequency
00000	47
00001	18
00010	29
...	
23114	1

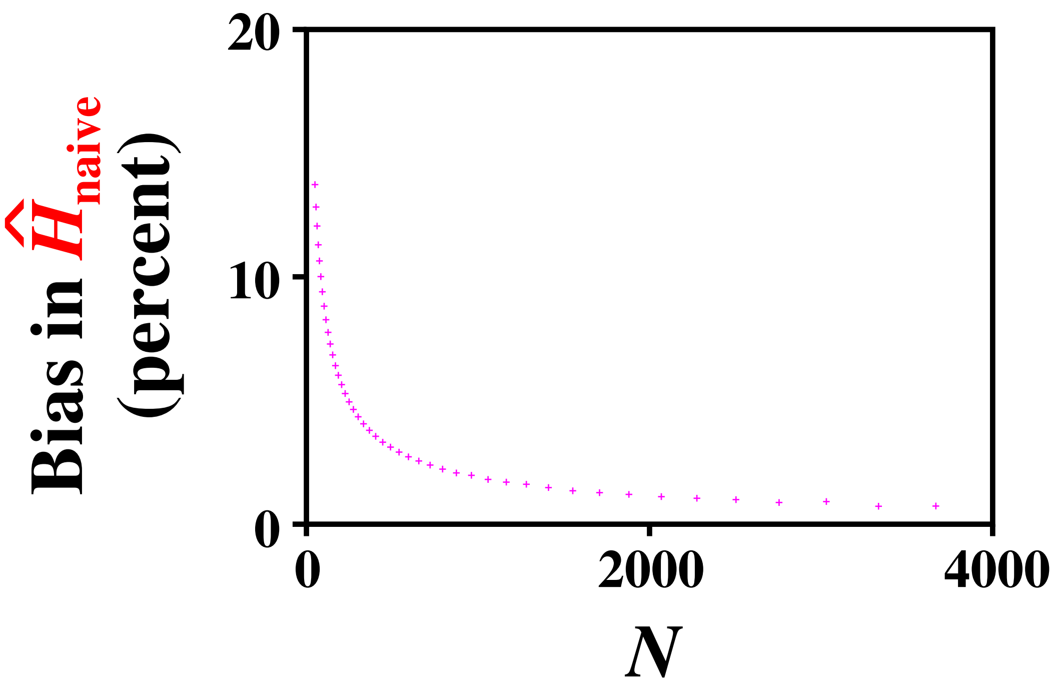


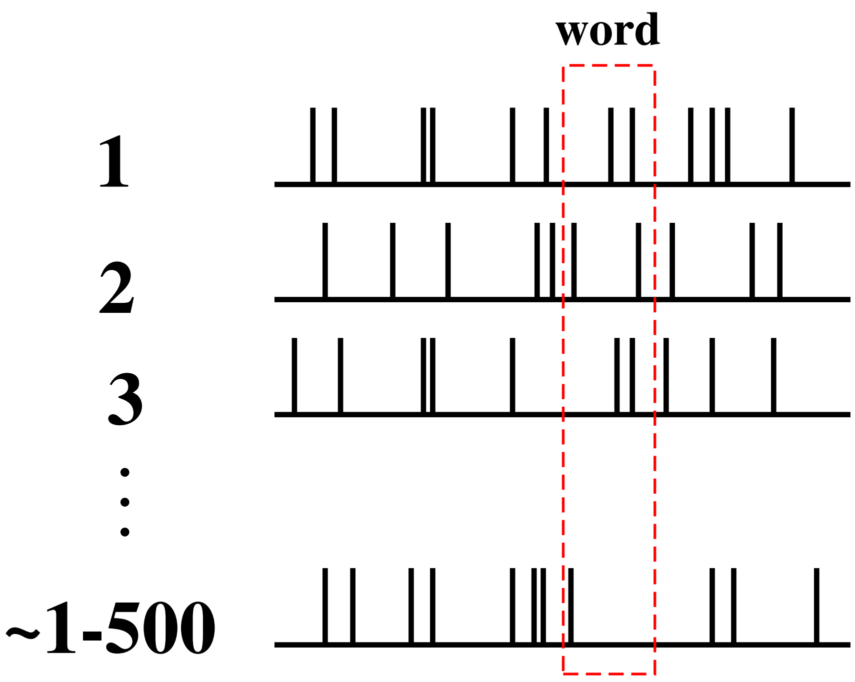
$\hat{H}_{naive}$ = 1.9 bits (N=300)

H_{true} = 2.4 bits

Bias in naive estimate of entropy

50 ms words, 10 ms letters, inhomogeneous Poisson process, mean rate = 10 Hz





For conditional entropy, you need to present the same stimulus many times. "Many" corresponds to several hundred. That's because each stimulus is several seconds long, and recordings last ~hours.

<u>Observations:</u>

1. **Small errors in entropy can cause large errors in mutual information, since mutual information is the <u>difference</u> between two entropies.**

2. **There is <u>no</u> unbiased estimator of entropy, since a finite amount of data does not tell you about the tails of the distribution.**

3. **There is <u>no</u> estimator with a consistent <u>upward bias</u>, since the entropy can be infinite (e.g., $p_j \sim 1/j\log^2 j$).**

<u>What one can do:</u>

1. **<u>Know your data</u> -- if you know something about the tails of the distribution, i.e., how p_j behaves as $j \to \infty$, there is hope of coming up with a decent estimator.**

2. **Strategy: find an estimator that provide, on average, <u>lower</u> and <u>upper</u> bounds on the entropy.**

rigorous **model dependent**

Our estimators

Data:

n_j = number of occurrences of element j (e.g., word j).

Definitions:

$N = \sum_j n_j$ = total number of elements sampled.

M = number of non-zero n_j .

M_1 = number of $n_j = 1$.

Example:

$n_j = (4, 5, 0, 1, 4, 1, 0, 2)$

$N = 17$

$M = 6$

$M_1 = 2$

Estimators (LB = lower bound; UB = upper bound):

$$\hat{H}_{LB} = \underbrace{-\sum (n_j/N) \log (n_j/N)}_{\text{naive}} + \underbrace{(M-1)/2N \ln 2}_{\text{Treves and Panzeri correction (NC 7, 1995)}}$$

$$\hat{H}_{UB} = \hat{H}_{LB} + \underbrace{M_1 \log N/N}_{\text{pulled more or less out of thin air.}}$$

$$\text{bias} = \langle \hat{H} \rangle - H_{\text{true}}$$

$$= \sum_{\{n\}} H(\mathrm{n})\, P(\mathrm{n}|\mathrm{p}) - \left(-\sum_j p_j \log p_j\right)$$

$$\mathrm{P(n|p)} = \text{multinomial} = N! \prod_j p_j^{n_j} / n_j!$$

<u>A little algebra:</u>

$$\langle \hat{H}_{LB} \rangle - \mathrm{H}_{\text{true}} = N^{-1} \sum_j f(p_j)$$

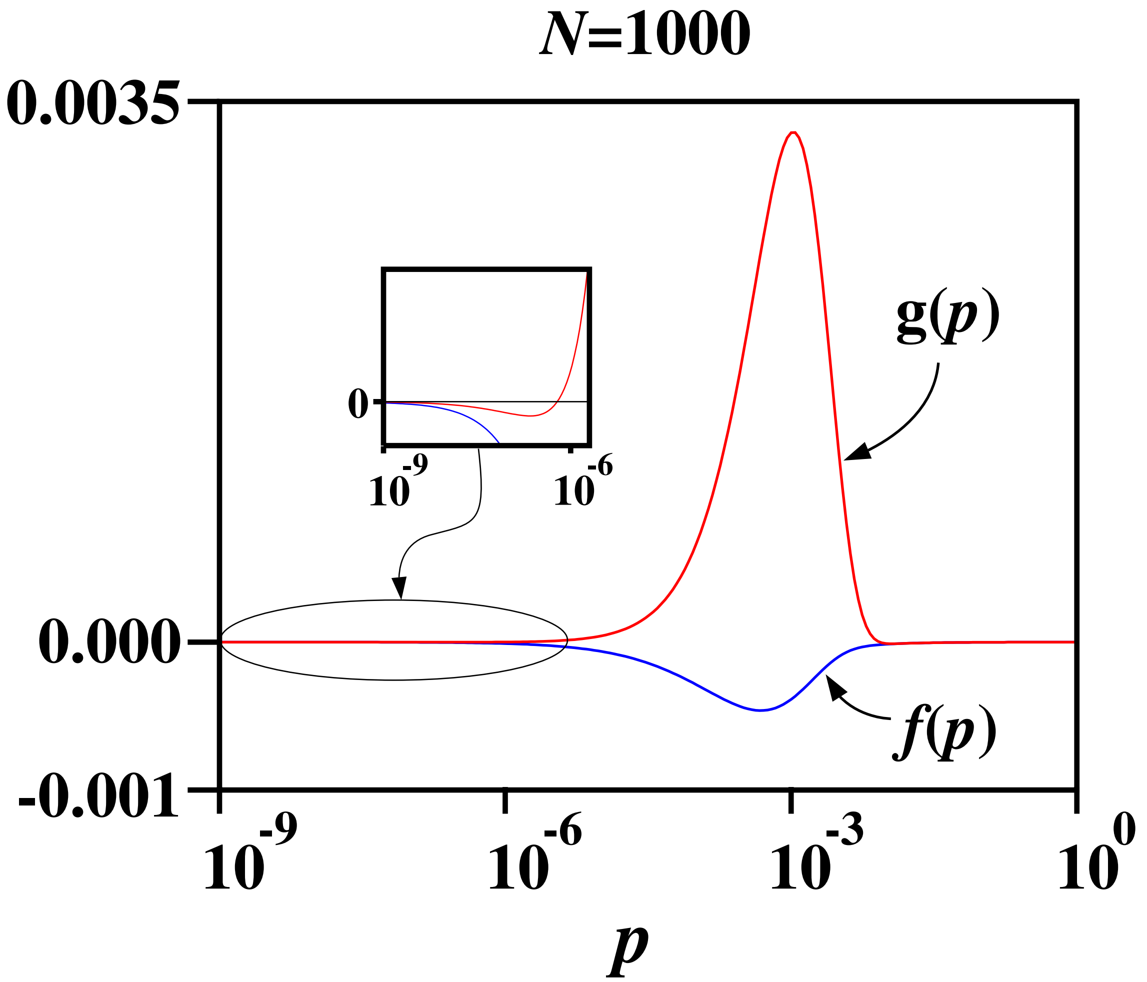
$$\langle \hat{H}_{UB} \rangle - H_{\text{true}} = N^{-1} \sum_j g(p_j)$$

$$f(p) = \sum_n n \log (n/N)\, P(n|p) + [1-p-\exp(N\log(1-p))]/2 \ln 2$$

$$g(p) = f(p) + (\log N)\, Np \exp((N-1)\log(1-p))$$

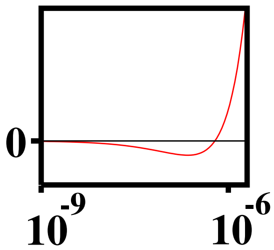
$$\text{binomial} = p^n (1-p)^{N-n} N!/n!(N-n)!$$

Upper and lower bounds



Observations:

1. $g(p)$ and $f(p)$ have a min and max at $p \approx 1/N$.
2. $f(p)$ is never positive.
3. $g(p)$ goes negative at p $\approx 1/N^2$.



Does this little negative portion on $g(p)$ (a supposed upper bound) matter?

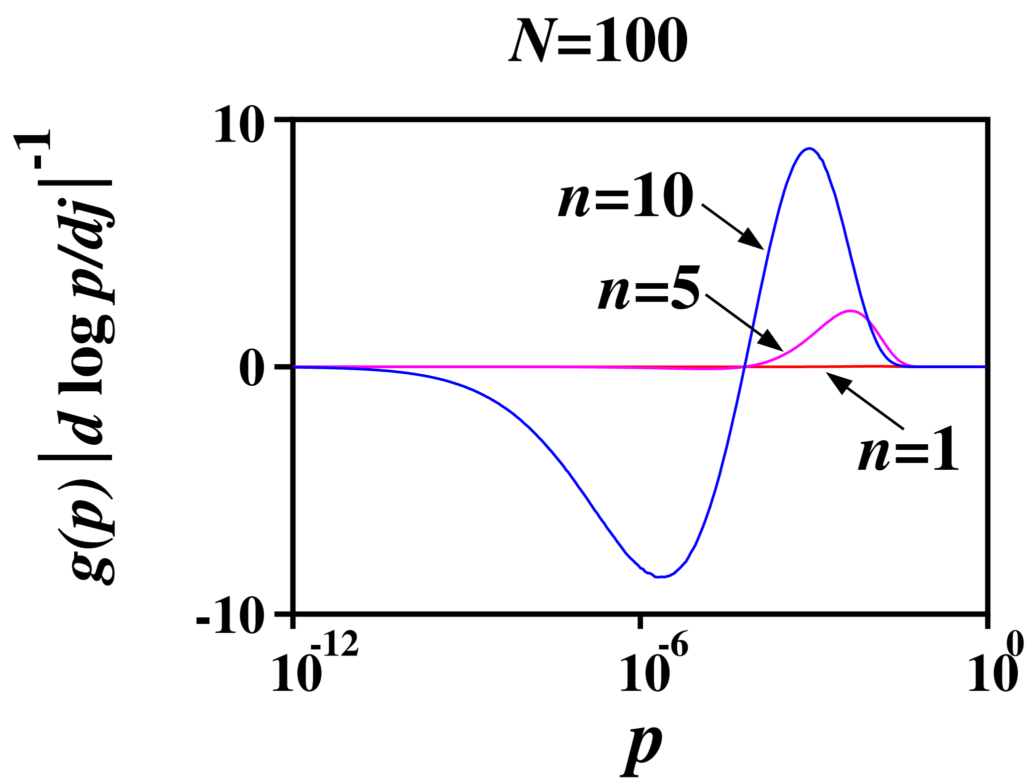
Recall: $$\langle \hat{H}_{UB} \rangle - \mathrm{H}_{\mathrm{true}} = N^{-1} \sum_j g(p_j)$$

$$\sum_j g(p_j) \rightarrow \int dj\, g(p_j) = \int d \log p \;\; g(p) \left| d \log p/dj \right|^{-1}$$

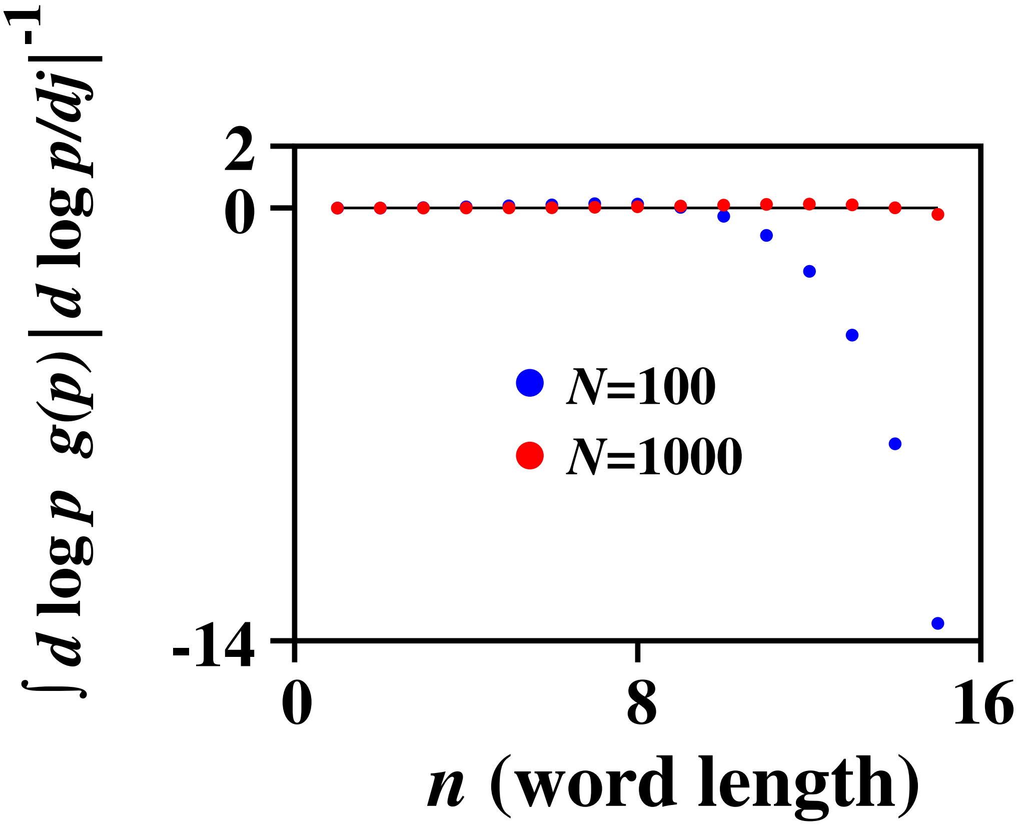
Everything depends on how p_j depends on j.

A reasonable model for the words/letters approach to computing entropy: if n is the number of letters (word length), and the spike train has no temporal correlations, then (see last page of poster):

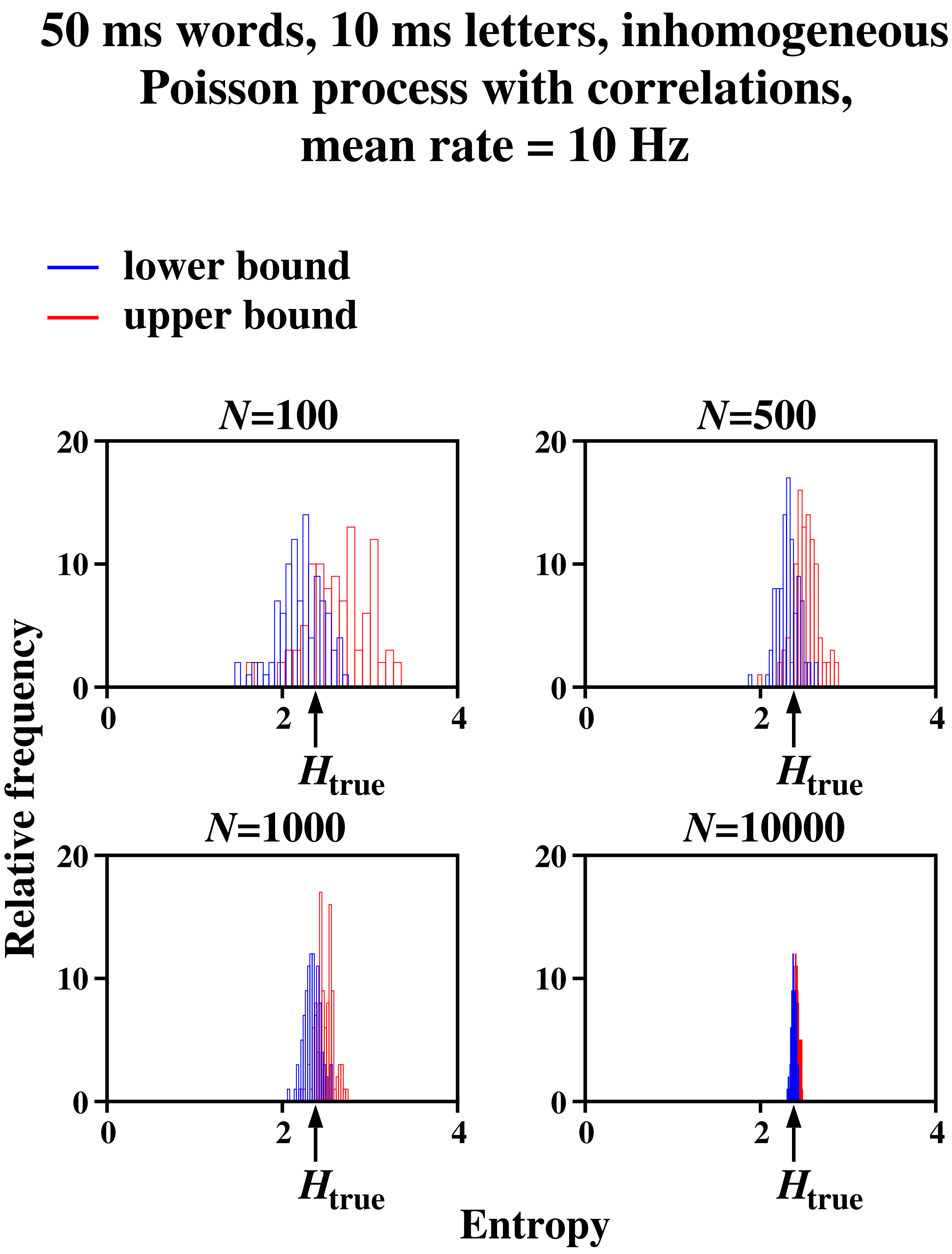
$$\boxed{p_j \approx p_0 \exp[-(p_0\, n!\, j)^{1/n}]}$$

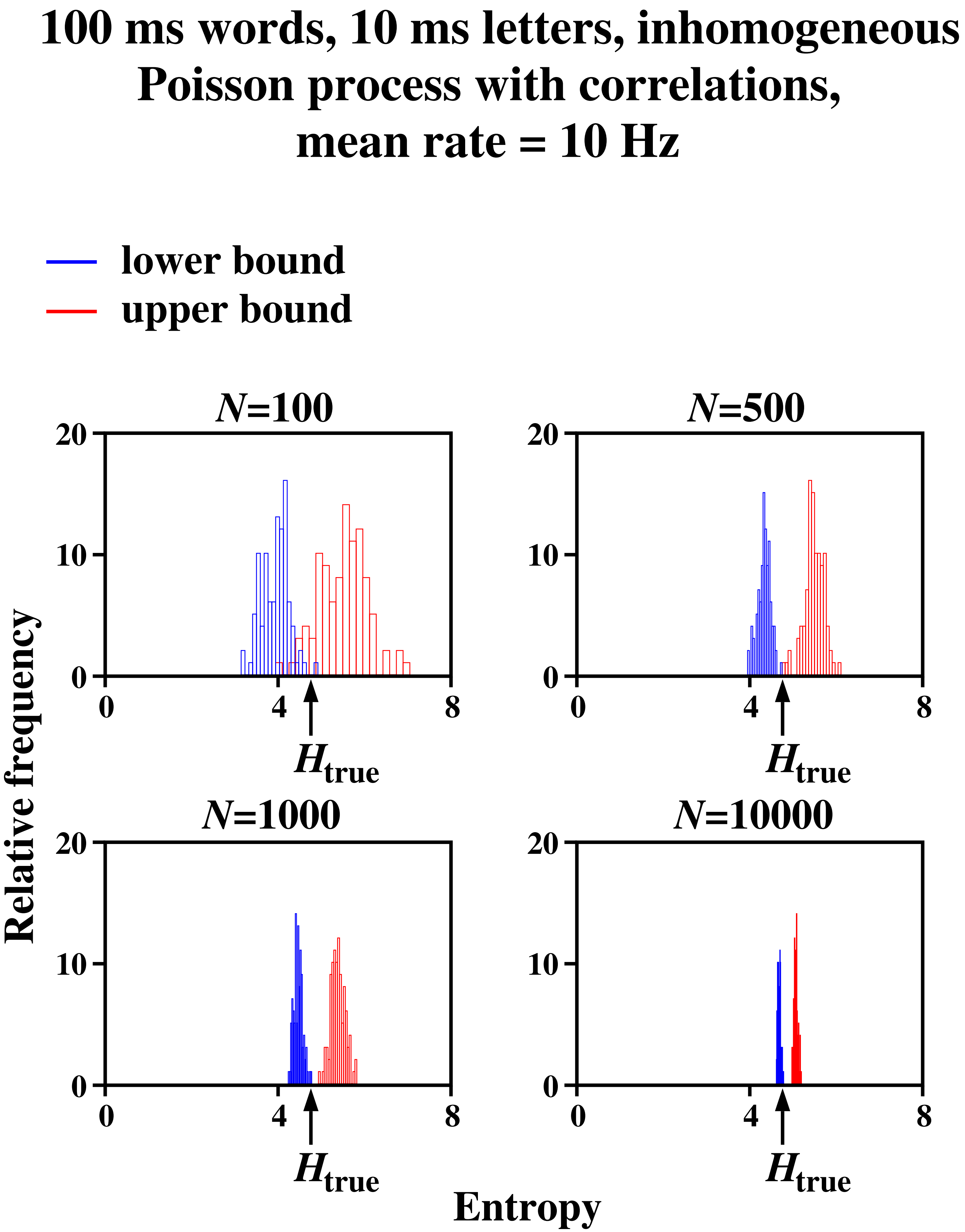


N=100
g(p) |d log p/dj|^-1
10
0
-10
n=10
n=5
n=1
10^-12
10^-6
10^0
p



∫ d log p g(p) |d log p/dj|^-1
2
0
-14
N=100
N=1000
0
8
16
n (word length)


50 ms words, 10 ms letters, inhomogeneous
Poisson process with correlations,
mean rate = 10 Hz
lower bound
upper bound
N=100
N=500
N=1000
N=10000
20
10
0
0
2
4
H_{true}
Relative frequency
Entropy



100 ms words, 10 ms letters, inhomogeneous Poisson process with correlations, mean rate = 10 Hz
lower bound
upper bound
N=100
N=500
N=1000
N=10000
Relative frequency
Entropy
H_true
0
4
8
10
20

Open questions:

1. **How well does this method work with real data? Answering this will require extensive simulations with surrogate data designed to match the properties of real data.**

2. **How well can we estimate the variance? Our current estimates for the variance tend to be too small by 10-30%. Better estimates are needed.**

3. **Are there better estimators for the upper and lower bounds than the ones presented here?**

The probability distribution of words, p_j (see pg. 12 of poster)

To estimate entropy from a finite data set, it is crucial to know the properties of the underlying probability distribution from which the data was sampled. Specifically, for a probability distribution, p_j, the entropy computed from a finite data set depends strongly on the behavior of p_j when j is large. (Here and in what follows we rank-order the p_j, so that $p_{j+1} \leq p_j$. Thus, large j corresponds to the tails of the distribution p_j.) As we showed in the poster, the estimate for an upper bound of the entropy becomes worse as the tails of the distribution become flatter. Moreover, for tails that are too flat, the upper bound can even turn into a lower bound, rendering the estimate useless.

Here we estimate the dependence of p_j on j when p_j are word probabilities, as calculated using the "direct" method [1] (see also poster page 5). We know of no rigorous way to estimate p_j in general, so we provide a really rough estimate. We do this by assuming that there are no temporal correlations in the spike train; that is, we approximate the spike count probability in each bin as independent of all other bins. Letting $p_k(l)$ be the probability of observing l spikes in bin k (out of n bins), the independence assumption allows us to write

$$p(l_1, ..., l_n) = \prod_{k=1}^{n} p_k(l_k) \tag{1}$$

where $p(l_1, ..., l_n)$ is the probability of observing l_k spikes in bin k, $k = 1, ..., n$, and each of the $p(l_1, ..., l_n)$ corresponds to a distinct p_j. To find out how p_j depends on j, we first solve the inverse problem – we find $j(p)$, the number of words with probability greater than p – and then invert $j(p)$ to find p_j.

The quantity $j(p)$ is the number of configurations of the l_k such that $p(l_1, ..., l_n) > p$. Taking the log of both sides and using Eq. (1), this expression becomes

$$\sum_k - \log p_k(l_k) < - \log p\,. \tag{2}$$

To determine the number of configurations of the l_k that satisfies Eq. (2), $j(p)$, it is useful to think of $(-\log p_1(l_1), -\log p_2(l_2), ..., -\log p_n(l_n))$ as points in an n-dimensional space. Then, $j(p)$ is simply the number of points below the plane $\sum_k x_k = -\log p$, where the x_k are coordinates in the same n-dimensional space as the $-\log p_k(l_k)$.

We now make a second approximation, that the points points $(-\log p_1(l_1), -\log p_2(l_2), ..., -\log p_n(l_n))$ are approximately evenly spaced; that is, $\log p_k(l) - \log p_k(l+1) = \Delta_k$, approximately independent of l. In that case, $j(p)$ is proportional to the volume bounded by the $n+1$ hyper-planes $\sum_k x_k = -\log p$ and $x_k = -\log p_k(0)$, $k = 1, ..., n$, with the constant of proportionality equal to $[n! \prod_k \Delta_k]^{-1}$. Defining $\prod_k \Delta_k \equiv p_0$, we thus have

$$j(p) = \frac{[-\log p + \sum_k \log p_k(0)]^n}{n! p_0}\,. \tag{3}$$

Inverting this expression and enforcing the normalization $\int_0^\infty dj\, p_j \approx 1$, we arrive at

$$p_j \approx p_0 \exp[-(p_0 n! j)^{1/n}]\,. \tag{4}$$

This expression, which is identical to the one on page 12 of the poster, should provide a *rough* estimate of p_j.

References

1. Strong, S.P., Koberle, R., de Ruyter van Steveninck, R.R. & Bialek, W. Entropy and information in neural spike trains. *Phys. Rev. Lett.* **80**, 197-200 (1998).