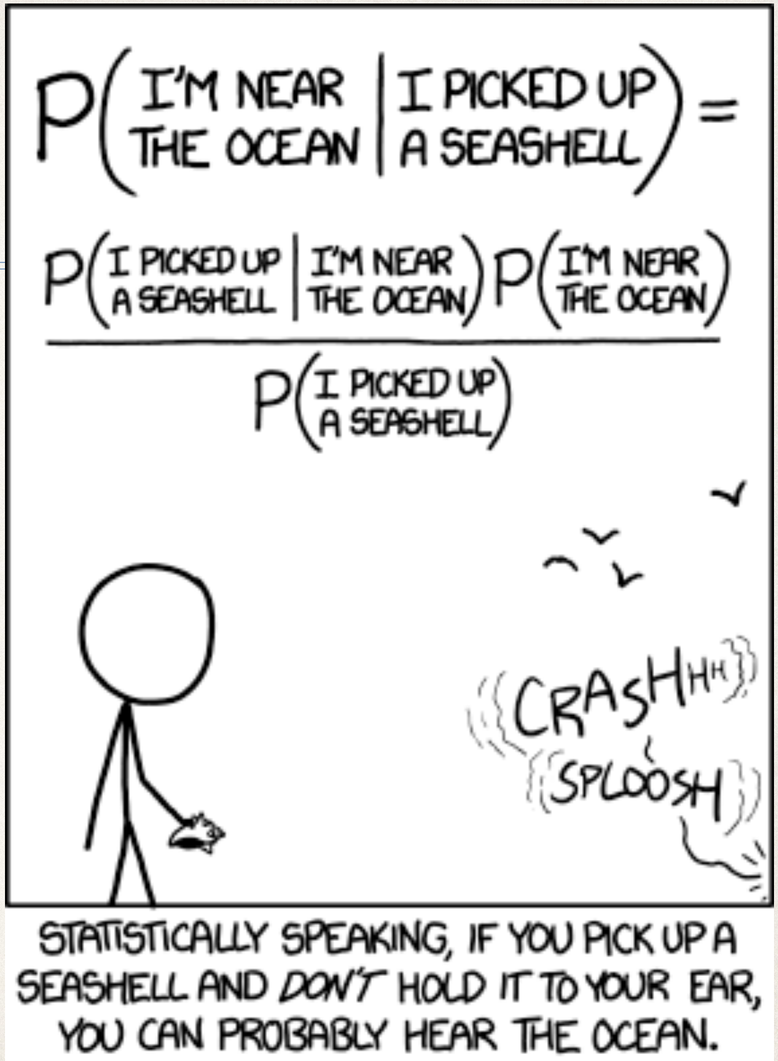
Probability Distributions

Statistics and Data Science

Spring 2025



http://xkcd.com/1236/

Goals for today: you should be able to...

- **Lecture 7 / 8 notebook:**
  - Choose appropriate priors for Gaussian parameters
- Explain what we mean by a statistic
- **Lecture 9 notebook:**
  - Identify and apply major statistics (mean, mode, median, standard deviation, etc.)

Review: Priors for the Normal Distribution

prob(params | data) = prob(data | params) prob(params) / prob(data)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

- Typically, for an uninformative prior we assume the values of the two parameters of a Gaussian are independent, so *prob(μ, σ) = prob(μ) prob(σ)*
- Jeffreys priors are:

  prob(μ) = 1

  prob(σ) = 1/σ

Bayesian use of Gaussians

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

- Suppose we have some measured value for x, where we expect x to come from a Gaussian of some unknown μ, but with known σ=2, e.g., $x \sim N(\mu,2^2)$.
- What is the posterior for μ given a measurement (say, x=5), using the Jeffreys prior $prob(\mu) = 1$?

```
x=5
sigma=2
mu=np.linspace(-10,10,201)
likelihood = ???
prior= ???
plt.plot(mu,likelihood,label='Likelihood')
plt.plot(mu,likelihood*prior,label='Posterior')
plt.legend()
```

You should find this is just a Gaussian, centered at our measured value, with standard deviation sigma!

Bayesian interpretation of the measurement

- So if we measure x=5, with a known uncertainty (σ) of 2, we'd expect μ to be within 2 units of 5 (i.e., <1 σ away) 68% of the time, within 4 (=2 σ) 95% of the time, etc.

Bayesian estimate for σ

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

- Now suppose we have some measured value for x, where we expect x to come from a Gaussian of known mean $\mu=0$, but with unknown σ.
- What is the likelihood for σ and posterior for σ given a single measurement (say, $x=5$), with prior $prob(\sigma) = 1/\sigma$? **This time, be sure to normalize the posterior distribution to have integral 1.**

```
x=5
mu=0
sigma=np.linspace(0.,50.,501)+1.E-3 # want to avoid dividing 1/0
likelihood=???
prior = ???
norm = ???

plt.plot(sigma,likelihood,label='Likelihood')
plt.plot(sigma,likelihood*prior/norm,label='Posterior')
plt.legend()
```

Statistics

Statistics!

- Suppose we have not just one measurement, but several independent ones, $x_1, x_2 \ldots x_N$, which we know are all drawn from the same Normal distribution, $x_i \sim N(\mu, \sigma^2)$ with known σ; and we want to determine what our best guess at the value of μ is.
- We want to apply:

$$prob(params \mid data) = prob(data \mid params)\, prob(params) \,/\, prob(data)$$

with:

$$prob(params) = 1$$

and

$$prob(data \mid params) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

Maximizing the likelihood

- Then: *prob(params | data)* $\propto \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$
- To maximize the posterior (i.e., choose the value of μ with greatest probability), we can just maximize the likelihood,

$$L \propto \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

- Note that *ln y* is a strictly increasing function of *y*; i.e., the bigger *y* is, the bigger *ln y* is. So the value of μ which maximizes *L* maximizes *ln L* as well.

$$ln\ L = \sum_i -\frac{(x_i-\mu)^2}{2\sigma^2} + C$$

At a maximum, $\partial\ ln\ L/\partial\ \mu = 0$, so:

$$\sum_i \frac{2(x_i-\mu)}{2\sigma^2} = 0$$, so $N\ \mu = \sum x_i$

So the most probable value of μ is: $\mu_L = \sum x_i / \mathrm{N}$

- This is a number we can calculate just given the data; i.e., a ***statistic***.
- Hopefully it is one you have seen before: the ***mean*** of the data (i.e., the mean of the x_i).
- The mean is an indicator of ***location***; and for data drawn from a Gaussian distribution all with the same sigma, it is the "best" estimate of the parameter μ (for some definition of "best")!
- Since we *maximized* $\ln L = \sum_i -\frac{(x_i - \mu)^2}{2\sigma^2} + C$

  we chose the value of μ which *minimizes* the sum of the squares of the deviations from μ; this is a '***least-squares***' estimate for μ

Weighted means

- Suppose we had taken the data to all be different measurements of the same property, which should always have the same mean μ, but each measurement x_i is drawn from a Gaussian with a different σ, σ_i. In that case, we would have found the value of μ which maximizes

$$ln\ L = \sum_i -\frac{(x_i-\mu)^2}{2\sigma_i^2} + C$$

- We then find:

$$\mu_L = \frac{\sum_i \frac{x_i}{\sigma_i^2}}{\sum_i \frac{1}{\sigma_i^2}}$$

- This is an example of a ***weighted mean***; it is still a combination of the data, and hence a statistic. In general, we can produce arbitrarily weighted means by

$$\mu_L = \sum w_i\, x_i \;/\; \sum w_i$$

Other measures of location: the mode

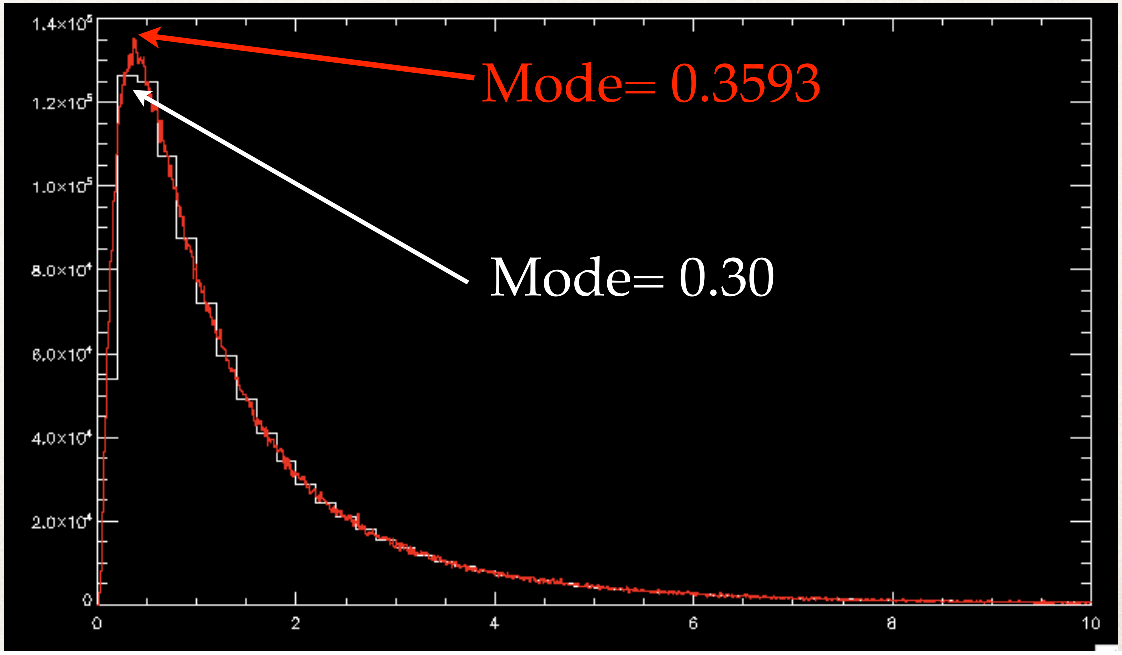
- There are two other common measures of location besides the mean:

1) the ***mode*** is the most common value of the data. Note it will depend on binning!

  - The mode of an image is a good representation of its background level.

10^6 points

$x \sim e^{N(0,1)}$



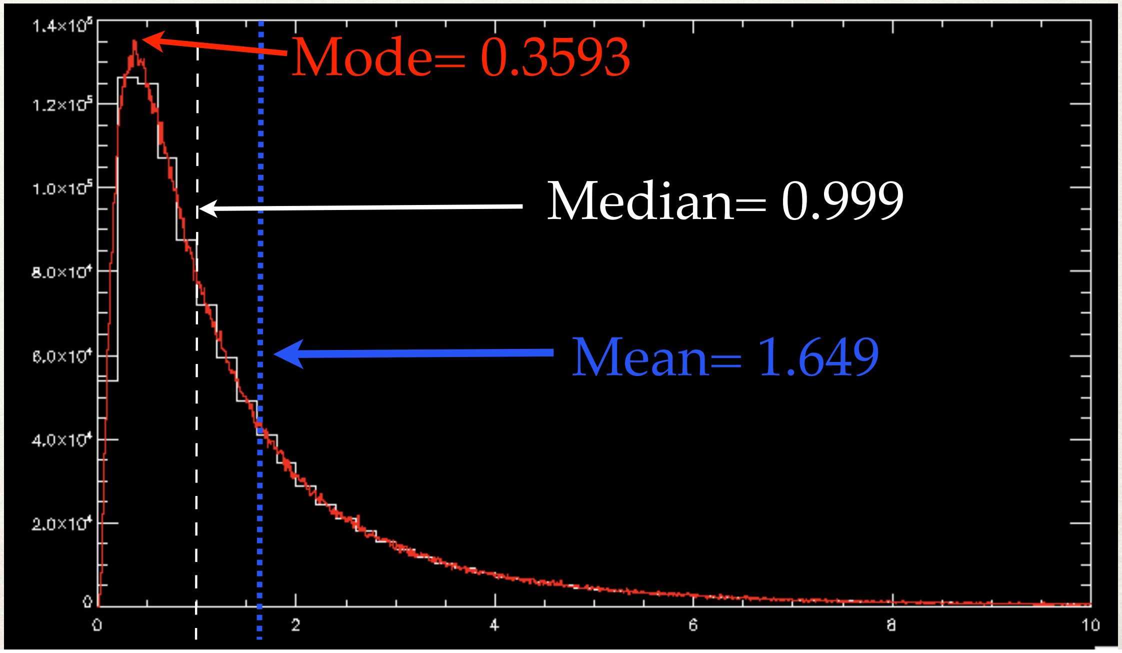
Other measures of location: the median

2) the ***median*** is the element of the data that is larger than 50% of the data and smaller than 50%; i.e., the 'middle' value.

- For a Gaussian with large N, mean, median, and mode should all occur at the same place. That isn't true for all PDFs:

10^6 points

$x \sim e^{N(0,1)}$



Determining mean & median in Python: Lecture 9 notebook

Let's make some log-normally distributed data:

```
data=np.exp(random.randn(100_000))
```

The function `np.mean()` returns the mean of an array:

- `print( np.mean(data) )`
- `print( data.mean() )`

The function `np.median()` returns the median of an array:

```
print( np.median(data) )
```

Determining mode in Python

- The value of the mode depends on how data are binned. E.g.:

```
print(f'Unrounded: {stats.mode(data)}')
```

does not give very useful results.

- If we want bins that correspond to some decimal place, we can do this by rounding and then using stats.mode():

```
data_r = np.round(data,decimals=2)
print(f'Rounded: {stats.mode(data_r)}' )
```

Determining mode in Python

- Otherwise, we can use np.histogram to determine the mode:

```
bins = np.linspace(-0.005,10.005,1002)
counts,edges=np.histogram(data,bins=bins)
```

- **`np.histogram()`** creates histograms in the same manner (with the bins and range keywords) as **`plt.hist()`**, but doesn't plot them. To determine the mode, we find the index in the counts array that corresponds to the maximum, and the corresponding bin center:

```
whmax=np.argmax(counts)
mode=(edges[whmax]+edges[whmax+1])/2
print(mode)
```

Let's turn that into a function...

```
def mode2(data,**kwargs):
# note: provide bins and (optionally) range keywords to not use
# defaults of np.histogram (10 bins, full range)
    counts,edges=np.histogram(data,**kwargs)
    whmax=np.argmax(counts)
    mode=(edges[whmax]+edges[whmax+1])/2
    return(mode)
```

`**kwargs` passes along a variable-length list of all the keywords passed to a routine; `*args` would pass a variable-length list of all non-keyword inputs.

Applying the mode2() function

- **Test it out:**

```
print( mode2(data) )
```

- **Try at least 3 different binnings; see how the mode changes.**

Statistics for the spread of values

- Suppose we have several independent measurements, $x_1, x_2 \ldots x_N$, which we know are all drawn from the same Normal distribution, $x_i \sim N(\mu, \sigma^2)$ with known μ; and we want to determine what our best guess at the value of σ is. We have:

$$prob(params \mid data) = prob(data \mid params)\, prob(params) \,/\, prob(data)$$

with Jeffreys prior:

$$prob(params) = 1/\sigma$$

and likelihood:

$$prob(data \mid params) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

Maximizing the likelihood, again

- Then: *prob(params | data)* $\propto \sigma^{-(N+1)} \prod_i \mathrm{e}^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$
- If we ignore the prior, we would maximize the likelihood:

$$L \propto \sigma^{-N} \prod_i \mathrm{e}^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

- The value of σ which maximizes *L* maximizes ln *L* as well:

$$\ln L = \sum_i -\frac{(x_i-\mu)^2}{2\sigma^2} - N \ln \sigma + C$$

- At a maximum, $\partial \ln \mathrm{L} / \partial \sigma = 0$, so:

$$\frac{1}{\sigma^3} \sum_i (x_i-\mu)^2 - \frac{N}{\sigma} = 0 \text{, so } N\sigma^2 = \sum_i (x_i-\mu)^2$$

So the most probable value of σ^2 is: $\sigma_L^2 = (\sum (x_i - \mu)^2) / N$

- This is another statistic you should have seen before.
- σ_L^2 is the ***variance*** or ***mean-square deviation*** of a set of the data (NOT of a distribution...)
- σ_L is the ***standard deviation*** or ***root-mean-square (RMS) deviation*** of the data.
- The standard deviation is an indicator of ***spread***; for data drawn from a Gaussian distribution with known mean all with the same sigma, it is the "best" estimate of the parameter σ (for some definition of "best")!
- We can also write $\sigma_L^2 = \langle x^2 \rangle - (\langle x \rangle)^2$

Other measures of spread

- The variance is the mean of $(x_i-\mu)^2$, and for data drawn from a Gaussian distribution provides a direct estimate of the σ parameter. We can write down a few, similar quantities that also measure spread:

1) The *average absolute deviation* or *average deviation*: $<|x_i - <x>|>$
   - For a Normal distribution, the expectation value of this quantity is $\sqrt{(2/\pi)}$ times σ, or $0.7979 \times \sigma$

2) The *median absolute deviation*, or MAD: $\mathrm{median}(|x_i - \mathrm{median}(x)|)$
   - For a Normal distribution, the expectation value of this quantity is $0.6745 \times \sigma$

Interquartile range

3) The interquartile range, or IQR:

- IQR= 75th percentile value - 25th percentile value
- = median of highest 50% of values - median of lowest 50% of values

- For a Normal distribution, the IQR = 1.349 x σ

Scale measures in Python

- The Standard Deviation of an array is calculated by the Python function `np.std()`.
- **Important**: You generally would want to call it with the keyword `ddof=1`, which calculates:

$$\sigma_s^2 = \frac{\Sigma(x_i - < x >)^2}{N - 1}$$

instead of $\sigma_L^2 = (\sum (x_i-\mu)^2) / N$.

- σ_s is known as the ***sample standard deviation***. It differs by a factor of N/(N-1) from what we derived before ("Bessel's correction"); this is substantial for small N, negligible for large N.
- This factor corrects for the fact that <x> is the value of x which minimizes the sum of the square of the deviations; i.e., it minimizes σ^2.
  - If we measure σ about that point, we get a value which must be biased low. For our earlier derivation, instead we assumed we knew μ, so we didn't have that problem.

Scale measures in Python

- **Try it out:**

```
print( np.std(data),np.std(data,ddof=1) )
print( np.std(np.log(data)),np.std(np.log(data),ddof=1) )
```

- We need to do some work to calculate the average absolute deviation and normalize it to match sigma for a Gaussian:

```
normavgabsdev = np.mean(np.abs(data-data.mean()))/0.7979
mnlog = np.mean(np.log(data) )
normavgabsdev_log = np.mean(np.abs( np.log(data)-mnlog) )/0.7979
```

Rank-based measures

- We can also calculate MAD (and its normalization) by hand:

```
meddata=np.median(data)
normmad = np.median(np.abs(data-meddata))/0.6745
normmad_log = np.median(abs(np.log(data)-np.log(meddata)))/0.6745
```

- Alternatively, we can use `scipy.stats.median_abs_deviation()` with `scale='normal'` (NOT the default):

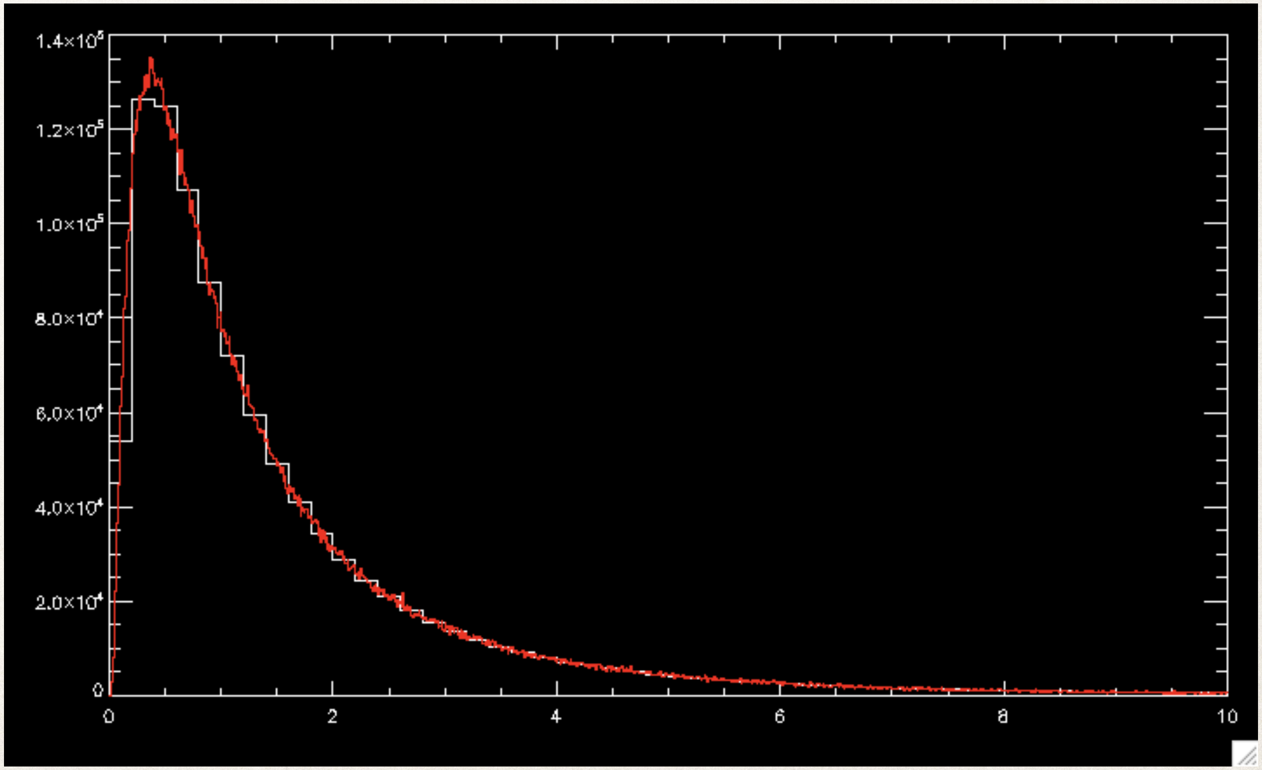
```
normmad_scipy = stats.median_abs_deviation(data,scale='normal')
```

- IQR requires us to use a new routine:

```
d25,d75 = np.percentile(data,[25,75])
normiqr = (d75-d25)/1.349
normiqr_log = (np.log(d75)-np.log(d25))/1.349
```

Results

- IF we normalize, all of these methods gave ~equal estimates for the true standard deviation for a Gaussian case (the log of our log-normal values)
- For the log-normal, the range is 0.89-2.17!
    - Compare to the true σ of the distribution = 2.16 !



Standard Deviation vs. Standard Error

- All of these methods estimate the **spread** of values that were drawn from some PDF.
    - The intrinsic spread will be the same no matter how many values we look at.
- Often, we are interested instead in **how accurately we have determined the mean** from some set of data: the "***standard error***".
    - In that case, the more data we have, the better-measured the mean should be.
- If we have 1 data point selected from *N(0,1)*, then that point will (of course) be spread around 0 as a Gaussian with sigma 1. What happens if we average N points all drawn from this Gaussian?

Averaging *n* data

- Let's try it, averaging 100 at a time:

```
nsims=int(1E5)
navg=100
data=random.randn(nsims,navg)
means=np.mean(data,axis=1)
```

- **Plot a histogram of the distribution of means, with bins 0.01 in size, over the range from -2 to +2**
- **Determine the standard deviation of the array of means**

Averaging n data

- What happens if we average 9 values at a time instead?

```
navg=9
data_9= ???
means_9=???
```

- **Overplot the histogram of the distribution of means, using plt.hist with the same binning as before.**
- **Determine the standard deviation of the array of means in each case**
- **Discuss: How does the scatter in the means scale?**

Results from averaging n data

- In each case, if we average n datapoints, the means are distributed as a Gaussian with the same mean as the true distribution (0) but spread :

$$\sigma_m = \frac{\sigma}{n^{1/2}}$$

- We could look at this as a consequence of the Central Limit Theorem:

  If you form averages M_n of samples of n from a population with finite mean and variance, then the distribution of $(M_n-\mu)/(\sigma/\sqrt{n})$ approaches a Gaussian with mean 0 and variance 1 as n goes to infinity.

- So the distribution of $M_n-\mu$ - which is the thing we just plotted (since $\mu=0$) - should be distributed as a Gaussian with mean 0 and variance σ^2/n, for large n.

The standard deviation of the mean

- In fact, the sum of 2 Gaussian-distributed variables will always be distributed as a perfect Gaussian, with $\sigma^2 = \sigma_1^2 + \sigma_2^2$ (where σ_1 and σ_2 are the standard deviations of the distributions the values x_1, x_2 are drawn from)
  - so the mean of n Gaussian-distributed variables will be distributed as a perfect Gaussian with variance $\sigma_{mean}^2 = \Sigma \frac{\sigma^2}{n^2}$ (using the fact that $N(\mu, \sigma^2) = \mu + \sigma N(0,1)$).

- We call $\sigma_m = \frac{\sigma}{n^{1/2}}$ the ***standard deviation of the mean*** or the ***standard error***
  - It is the RMS deviation of the **mean** of n data from the **true mean** of the distribution they come from.

The standard deviation of the mean

- We would expect (in the frequentist view) that 95% of the time the **true mean**, μ, will lie in the interval ($\langle x \rangle - 2\,\sigma_m$, $\langle x \rangle + 2\,\sigma_m$).* We can call that a ***95% confidence interval*** for μ.
- σ_m will **<u>always</u>** be smaller than (or equal to, for n=1) the sample standard deviation, which describes the spread of individual measurements
  - Instead, the standard error tells us how well we know the mean of the distribution
- The key thing to remember: as we acquire more data, **the standard deviation should not decrease**, as it describes the observed spread of individual values, but **our knowledge of the mean value does get better** from more data.

* IFF you know σ_m perfectly

Swimming in a sea of statistics

Estimators of location of data:

- Mean (`np.mean`)
- (Inverse-Variance) Weighted Mean (`np.average`)
- Mode (`mode2`)
- Median (`np.median`)

Estimators of spread of data:

- Sample Standard Deviation (`np.std`)
- Avg. Absolute Deviation
- Median absolute deviation (`scipy.stats.median_abs_deviation`)
- Interquartile Range (IQR, `scipy.stats.iqr`)

How do we determine the right statistic to use for our situation?

How should we choose amongst all these statistics?

- For data that really is distributed as a Gaussian, it is possible to show that the ordinary mean and sample standard deviation are the 'best' estimates of the true parameters μ and σ - for some definition of 'best'. What makes a statistic 'good' or 'better' than some other, anyway?

  1) We'd like our statistics to be ***unbiased*** - i.e., to have an expectation value equal to the parameter of interest, not offset from it. For a Normal distribution, $\langle x \rangle$ is unbiased, while σ_s has a modest (max. -20%) bias for small N.

  2) We'd like our statistics to be ***consistent*** - i.e., to lie in a narrower and narrower window around the correct value of some parameter for large N. An unbiased statistic is always consistent.

How should we choose amongst all these statistics?

3) A statistic should be ***impartial***: our conclusions should not depend on swapping the labels on the points/datasets (unless time is an important variable) or the units used.

- E.g., if we estimate the mean of sample A is higher than the mean of sample B by δ, using the same procedure with A and B reversed should yield $-\delta$.

4) We'd like our statistics to be ***efficient*** - to require as small a sample as possible to yield an accuracy within some threshold.

- Given a distribution, we can calculate the ***Asymptotic Relative Efficiency*** (ARE)*:*
  - If statistic A gives the same error with N_A data points as statistic B gives with N_B, the ARE of statistic A is the limit as N approaches ∞ of N_B/N_A . E.g., if N_A=1E6 yields the same errors as N_B=6E5, then statistic A has an ARE of 60%.

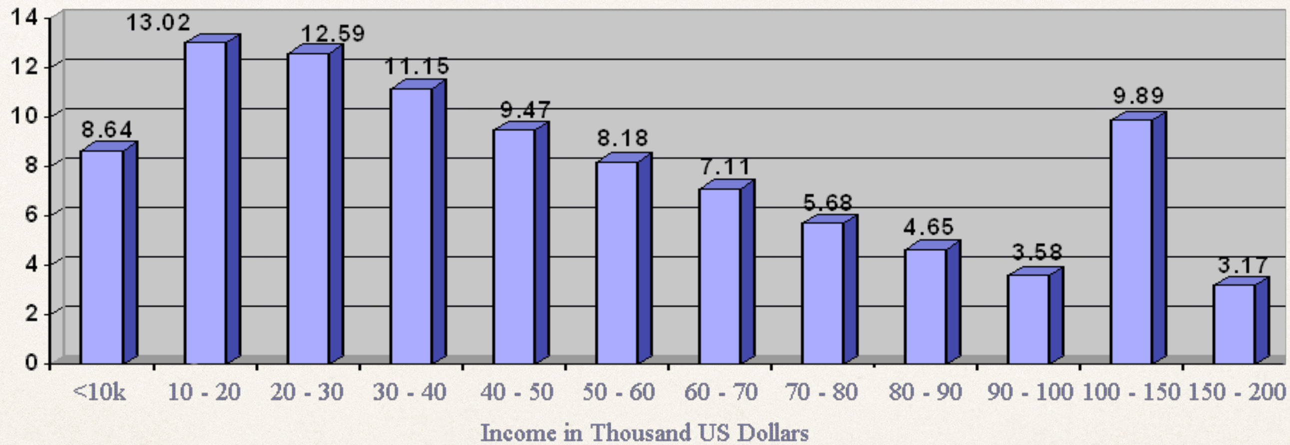
How should we choose amongst all these statistics?

5) A statistic should have ***closeness***: i.e., give a value as close as possible to the true value of some parameter of interest. However, there's lots of ways to measure closeness: do we minimize the RMS error? the average absolute deviation? etc.

- We can generalize this concept to say that a statistic should **minimize loss**, where the "***loss***" is the expectation value of some function over all possible samples.
- The estimators we derived from maximum likelihood for a Normal distribution would be equivalent to minimizing a loss given by $\sum_i -(x_i-\mu)^2 / (2\,\sigma_i^2)$. A different weighing of loss (e.g. one that depends linearly on deviations, rather than the square) would yield a different 'best' statistic.
- Some statistics minimize the maximum possible loss, instead of the expectation value; these are called ***Mini-max estimators***.

How should we choose?

6) Ideally, a statistic should be ***robust***: i.e., give the correct answer even if we have a non-Normal distribution (e.g., a Gaussian plus outliers). Although the ordinary mean has a high efficiency for normally-distributed data, **it is not robust**.



- This distribution has mean $60,528, median $44,389. Which is more representative of the population?
- What happens to each one if someone finds $10 billion stuffed in their couch?