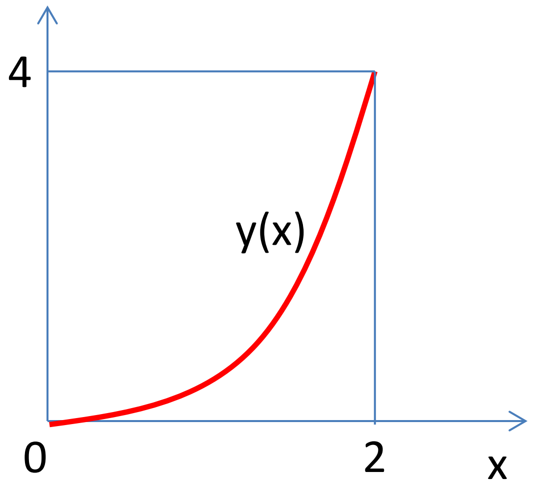
Modeling Uncertainty

Phys 281 – Class 9

Grant Wilson

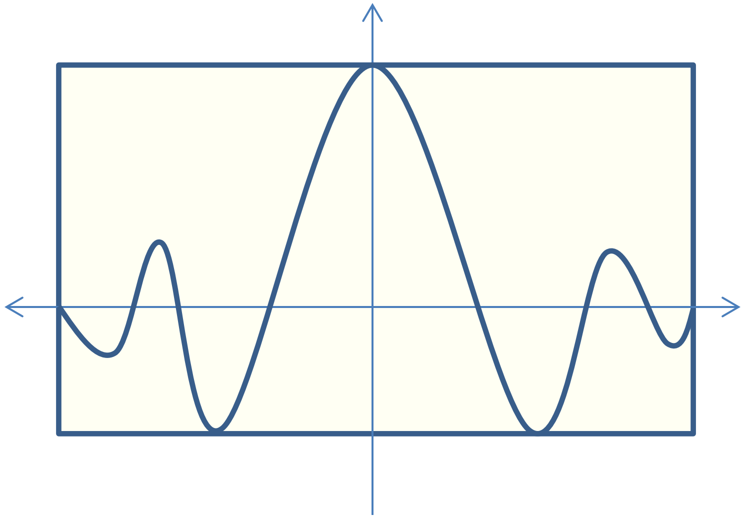
Class 08 Exercises – P1.1

```
#1.1 - Integrate x^2 from 0 to 2 using MC integration
#pick x and y points at random to make npts-ordered pairs
npts = int(1e5)
x1 = np.random.uniform(0,2,npts)
y1 = np.random.uniform(0,4,npts)
box_area = 4.*2.

#here is the function to integrate
def func(x):  return x**2

#need the values of y(x1) to compare to y1
yfunc = func(x1)

#now count up the y1 that fall between 0 and y(x1)
count = 0.
for i in range(npts):
   if (y1[i]<yfunc[i]):
      count = count+1

#the integral is the fraction of the points below y(x1) times the box area
int1 = count/npts * box_area
int_romberg = integrate.romberg(func,0.,2.)
print "Problem 1"
print '    I1 = ', int1
print '    I_romberg = ', int_romberg
```



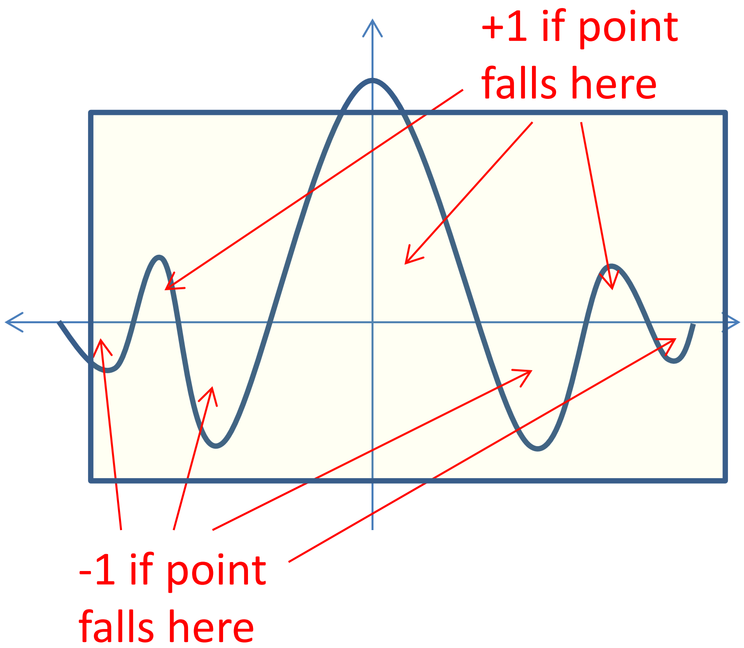
Class 08 Exercises – P1.2

```
#1.2 - Integrate sin(x)/x from -pi to pi using MC integration
npts = int(1e5)

#pick x and y points at random to make npts-ordered pairs
x1 = np.random.uniform(-4.*np.pi,4.*np.pi,npts)
y1 = np.random.uniform(-0.25,1,npts)
box_area = 8.*np.pi*1.25

#here is the function to integrate
def func2(x):
    return np.sin(x)/x

#need the values of y(x1) to compare to y1
yfunc = func2(x1)

#now count up the y1 that fall between y(x1) and th x-axis.  Points
#that fall above the x-axis should get a +1 while points that fall
#below the x-axis should get a -1.  Points that fall outside of
#the function should get 0
count = 0.
for i in range(npts):
    if ((y1[i]<yfunc[i]) and (y1[i]>0)):
        count = count+1
    elif ((y1[i]>yfunc[i]) and (y1[i]<0)):
        count = count-1
```

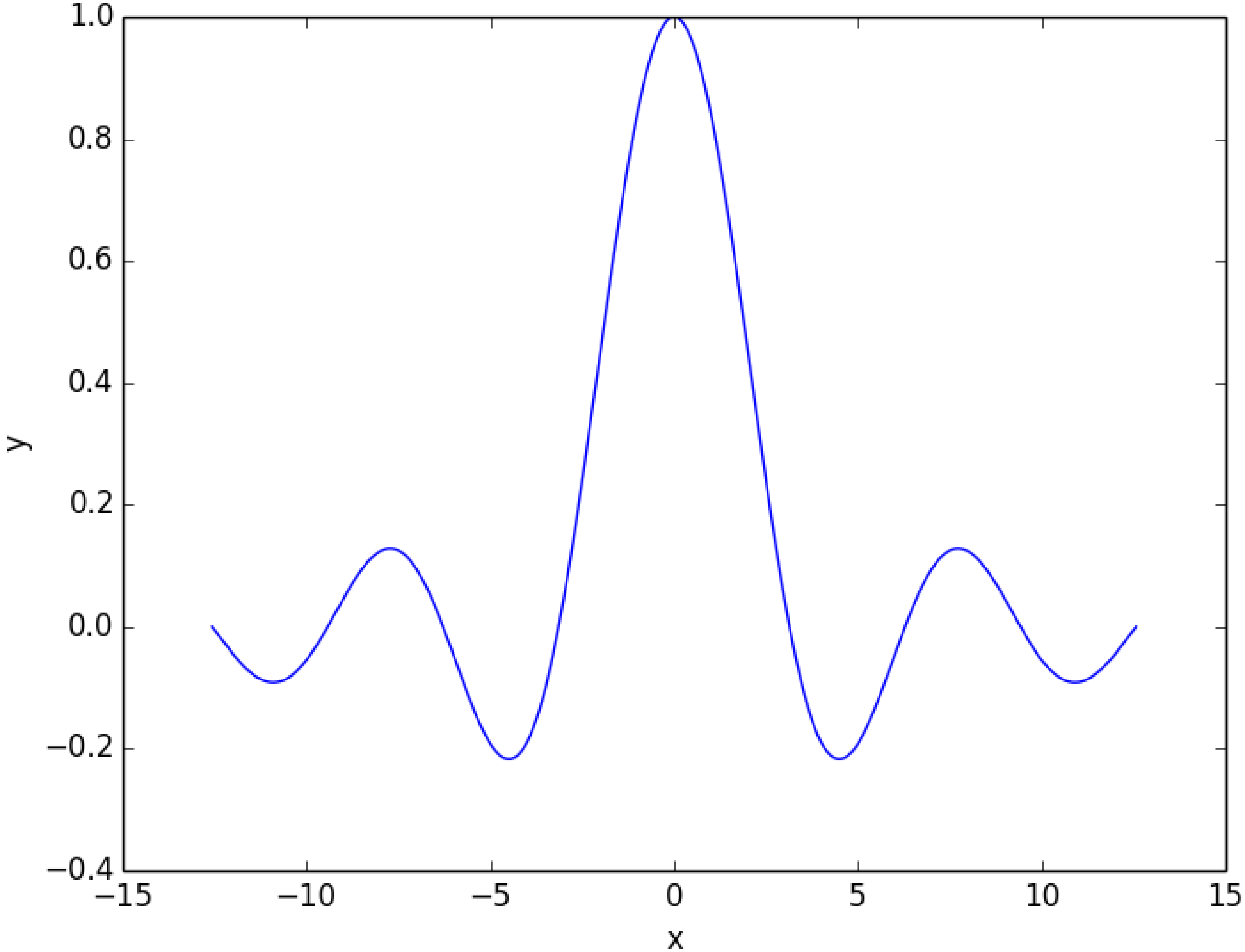


Class 08 Exercises – P1.2

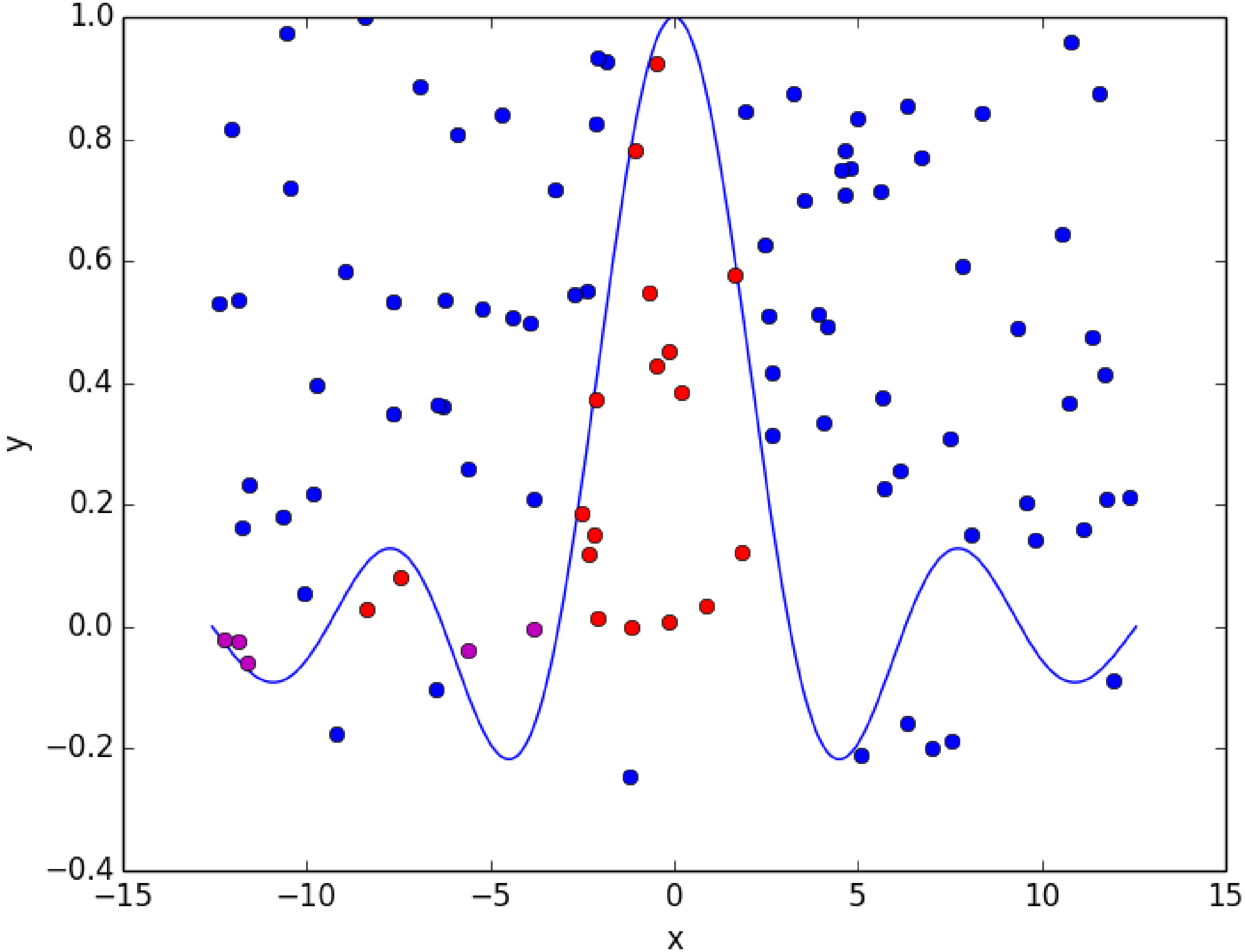
```
#1.2 - Integrate sin(x)/x from -pi to pi using MC integration
npts = int(1e5)

#pick x and y points at random to make npts-ordered pairs
x1 = np.random.uniform(-4.*np.pi,4.*np.pi,npts)
y1 = np.random.uniform(-0.25,1,npts)
box_area = 8.*np.pi*1.25

#here is the function to integrate
def func2(x):
  return np.sin(x)/x

#need the values of y(x1) to compare to y1
yfunc = func2(x1)

#now count up the y1 that fall between y(x1) and th x-axis.  Points
#that fall above the x-axis should get a +1 while points that fall
#below the x-axis should get a -1.  Points that fall outside of
#the function should get 0
count = 0.
for i in range(npts):
  if ((y1[i]<yfunc[i]) and (y1[i]>0)):
    count = count+1
  elif ((y1[i]>yfunc[i]) and (y1[i]<0)):
    count = count-1
```



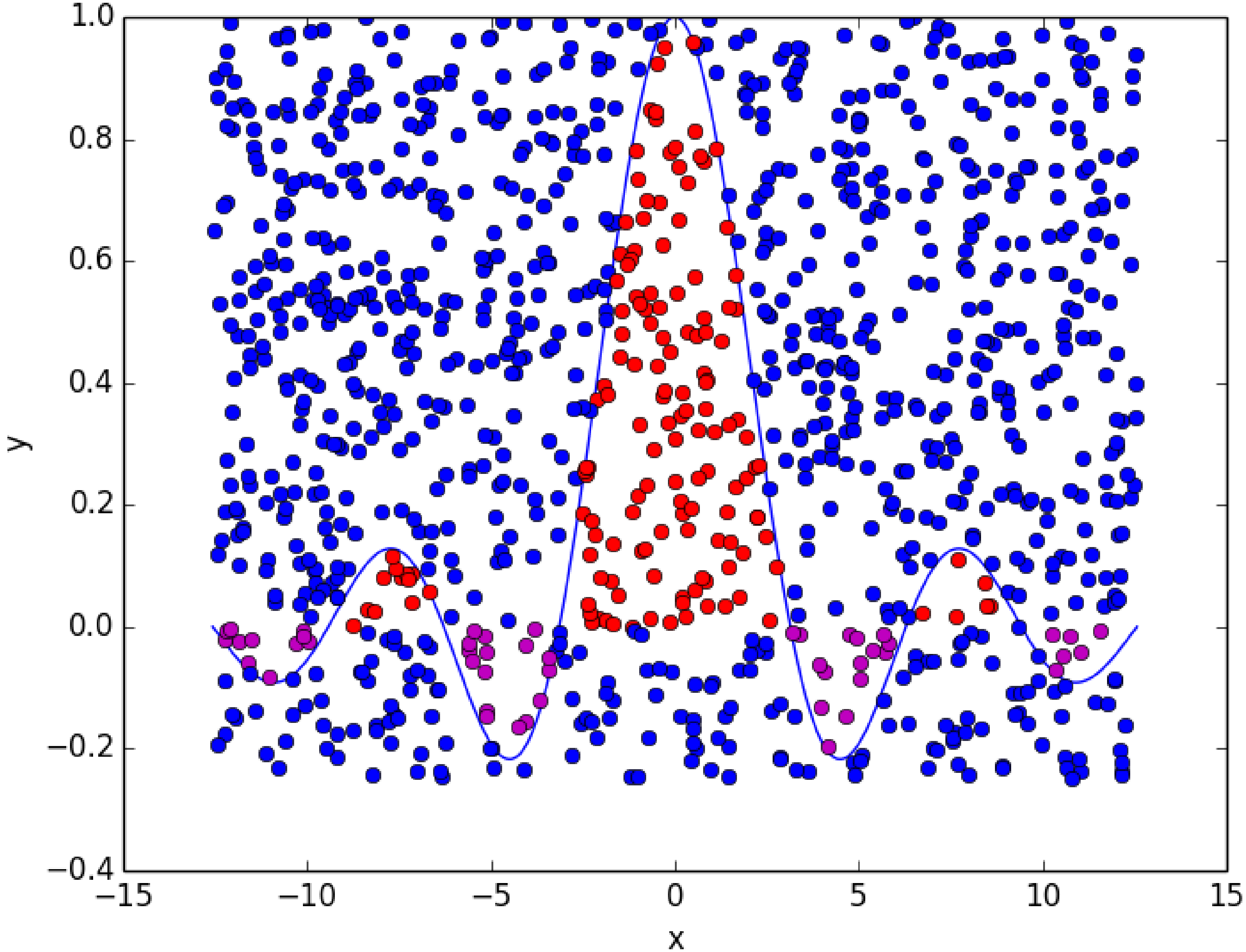
Function to be integrated



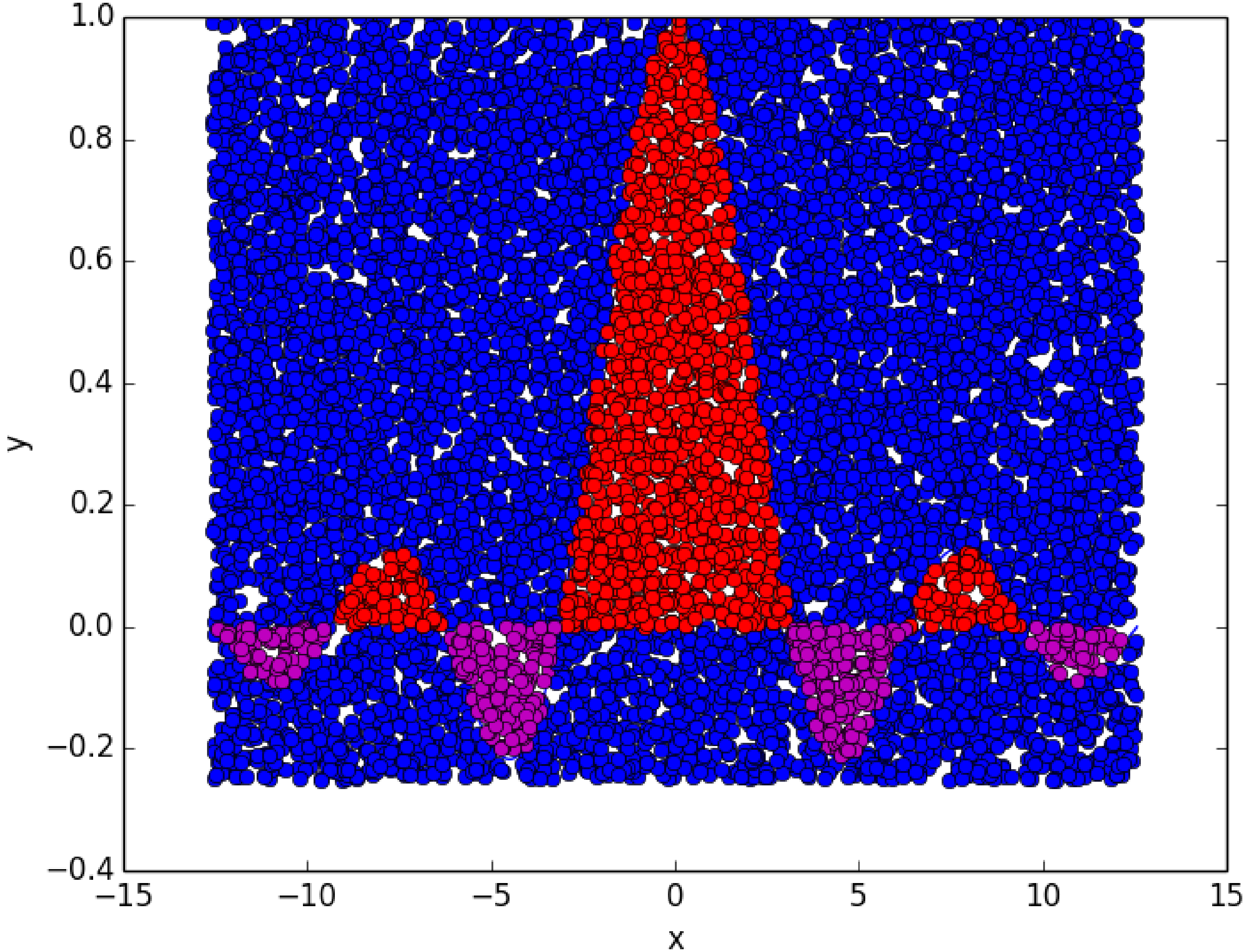
100 points



1000 points



10,000 points



Class 08 Exercises – P2

What needs to be done for P2 is interesting enough that I think I'll make a video short for the approach for the answer. In the meantime, here's the answer:

- Suppose you want to integrate some function $f(\mathbf{x})$ over some subspace $\mathcal{C}$ of an n-dimensional space spanned by $\mathbf{x}$ (note use of bold rather than using a vector sign).

$$I = \int_{\mathcal{C}} \mathrm{d}\mathbf{x}\ f(\mathbf{x}) \tag{4}$$

- Monte Carlo integration estimates the integral as

$$\int_{\mathcal{C}} \mathrm{d}\mathbf{x}\ f(\mathbf{x}) = V\langle f\rangle \pm V\sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}} \tag{5}$$

where we define the moments taken over N sample points as

$$\begin{aligned} \langle f\rangle &\equiv \frac{1}{N}\sum_{i=0}^{N-1} f(\mathbf{x}_i) \\ \langle f^2\rangle &\equiv \frac{1}{N}\sum_{i=0}^{N-1} f^2(\mathbf{x}_i) \end{aligned}$$

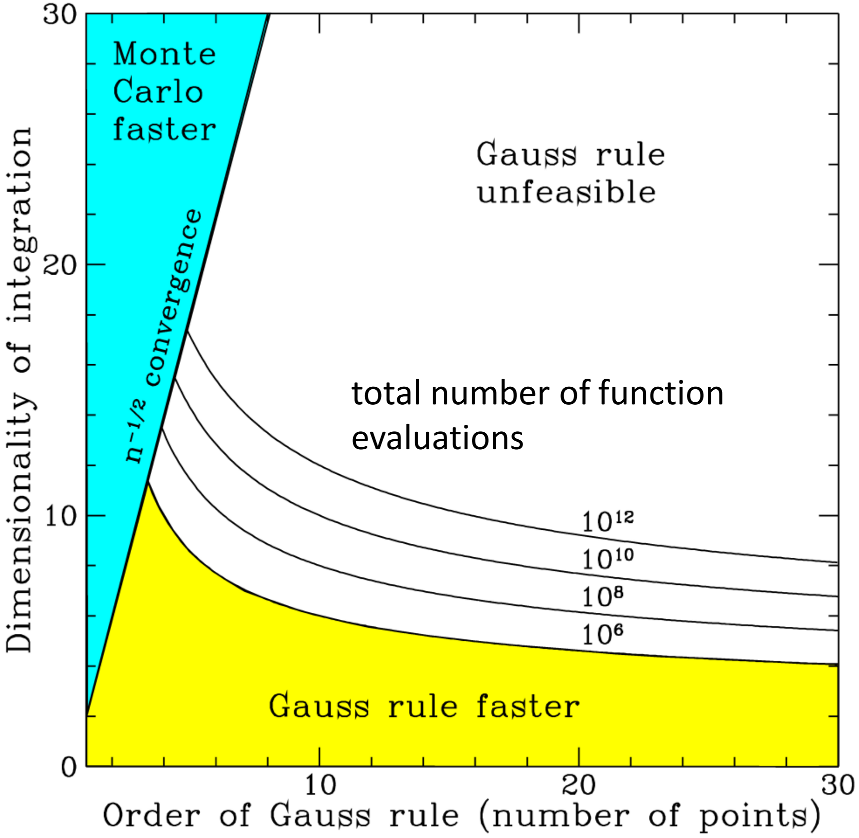
and the volume V encloses the subspace $\mathcal{C}$.

Choosing your integration approach

Monte Carlo Integration is best used when

1. the dimension of the problem is greater than 10
2. there is no easy way to write down the limits of the integration

Otherwise, go with Gaussian Quadrature



Modeling Uncertainty

Phys 281 – Class 9

Grant Wilson

Simulating probabilities

- What does a 1 in 5 chance of something occurring mean?
- How could you simulate that?

Disease Simulation

Given a positive test result, what is the probability that you've actually got the disease?

#here is the information about the disease and test

prevalence = 0.1

false_positive_rate = 0.2

false_negative_rate = 0.05

#We could calculate this using probability theory, or we could simulate
#the process. Let's do the simulation.

#set up the trials, one per test

ntrials = 10000

index = np.arange(ntrials)

Disease Simulation

```
#diseased is an array of one element for each trial
#(think of them as different people in the community)
#diseased = 1 implies that the person really does have the disease
#diseased = 0 implies that the person does not have the disease
diseased = np.zeros(ntrials)
u = np.random.rand(ntrials)
for i in index:
```

Disease Simulation

```
#diseased is an array of one element for each trial
#(think of them as different people in the community)
#diseased = 1 implies that the person really does have the disease
#diseased = 0 implies that the person does not have the disease
diseased = np.zeros(ntrials)
u = np.random.rand(ntrials)
for i in index:
    if (u[i] <= prevalence):
        diseased[i] = 1
```

Disease Simulation

```
#at this point, all the diseased=1 people have it, diseased=0 people are clear
#run through them again and apply the test
#test_result = 1 implies they have it
#test_result = 0 implies they don't
test_result = np.zeros(ntrials)
for i in index:
   if (diseased[i] == 1):

   if (diseased[i] == 0):
```

Disease Simulation

```
#at this point, all the diseased=1 people have it, diseased=0 people are clear
#run through them again and apply the test
#test_result = 1 implies they have it
#test_result = 0 implies they don't
test_result = np.zeros(ntrials)
for i in index:
    if (diseased[i] == 1):
        #this person has it, what does the test say?
        u = np.random.rand()
        if (u <= false_negative_rate):
            test_result[i] = 0
        else:
            test_result[i] = 1

    if (diseased[i] == 0):
```

accounts for false negatives

Disease Simulation

```
#at this point, all the diseased=1 people have it, diseased=0 people are clear
#run through them again and apply the test
#test_result = 1 implies they have it
#test_result = 0 implies they don't
test_result = np.zeros(ntrials)
for i in index:
    if (diseased[i] == 1):
        #this person has it, what does the test say?
        u = np.random.rand()
        if (u <= false_negative_rate):
            test_result[i] = 0
        else:
            test_result[i] = 1

    if (diseased[i] == 0):
        #this person doesn't have it, what does the test say?
        u = np.random.rand()
        if (u <= false_positive_rate):
            test_result[i] = 1
        else:
            test_result[i] = 0
```

accounts for false negatives

accounts for false positives

Disease Simulation

```
#count up the different kinds of people
diseased_tested_positive = 0.
diseased_tested_negative = 0.
not_diseased_tested_positive = 0.
not_diseased_tested_negative = 0.
number_diseased = 0
for i in index:
    if (diseased[i] == 1 and test_result[i] == 1):
            diseased_tested_positive += 1
    if (diseased[i] == 1 and test_result[i] == 0):
            diseased_tested_negative += 1
    if (diseased[i] == 0 and test_result[i] == 1):
            not_diseased_tested_positive += 1
    if (diseased[i] == 0 and test_result[i] == 0):
            not_diseased_tested_negative += 1
```

Disease Simulation

```
#now, given a positive test result, what's the probability you're diseased
probability_diseased_given_positive_test =
diseased_tested_positive/(diseased_tested_positive +
not_diseased_tested_positive)*100.

probability_not_diseased_given_positive_test =
not_diseased_tested_positive/(not_diseased_tested_positive+diseased_tested_positi
ve)*100.

print "Probability of being diseased given a positive test:
"+str(probability_diseased_given_positive_test)+'%'

print "Probability of not being diseased given a positive test:
"+str(probability_not_diseased_given_positive_test)+'%'
```

Disease Simulation: Results

False Positive Rate	False Negative Rate	P(sick|+)	P(okay|+)
20%	5%	35%	65%
10%	5%	50%	50%
1%	5%	92%	8%
0.1%	5%	99%	1%

Probability of being diseased and getting a positive test result.

Probability of being okay and getting a positive test result.

Exercise #1

- Download the code: epidemiology.py from the class Moodle site.

- Modify the program to print out the probability of being sick given a negative test result.

Measurement Errors

A quick quiz:

1. What type of number does adding a float and an integer yield?

Measurement Errors

A quick quiz:

1. What type of number does adding a float and an integer yield?
2. What type of number does adding a finite and an infinite number yield?

Measurement Errors

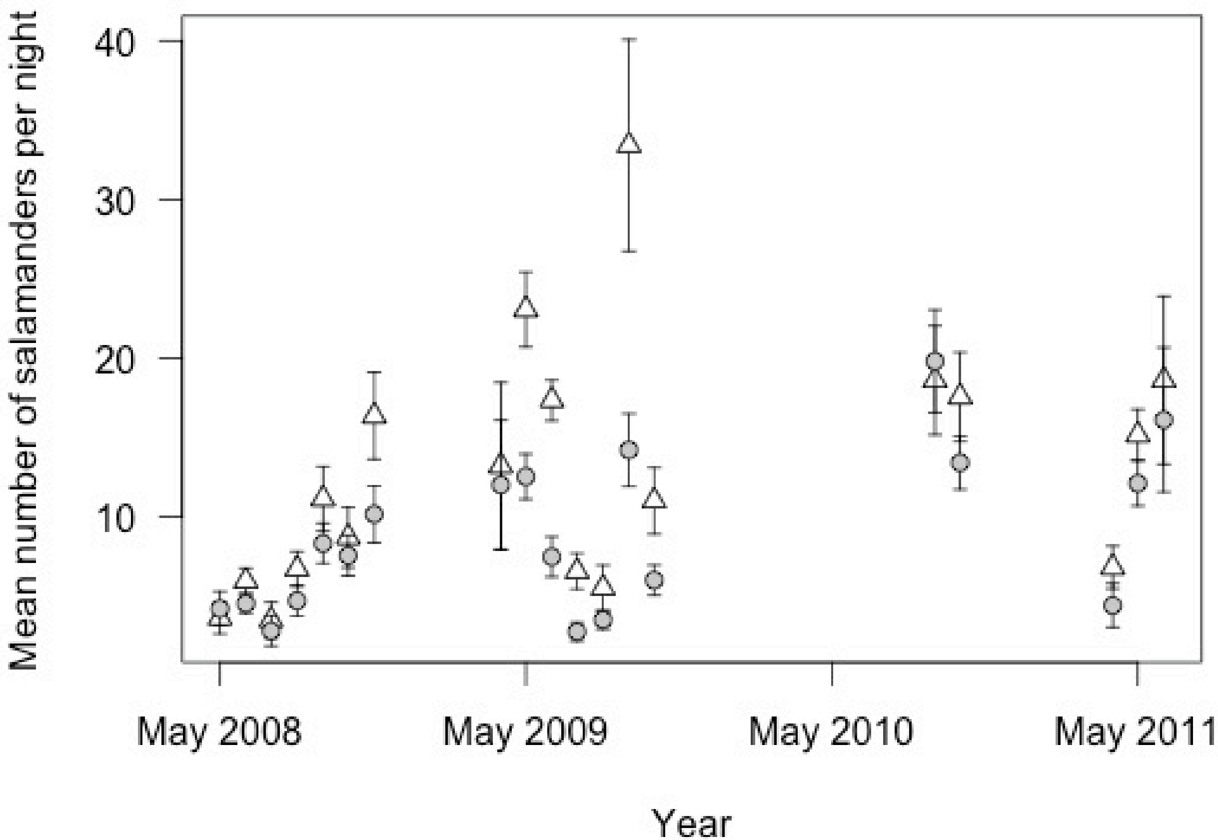
A quick quiz:

1. What type of number does adding a float and an integer yield?

2. What type of number does adding a finite and an infinite number yield?

3. What type of number does adding a random deviate and a “normal” number yield?

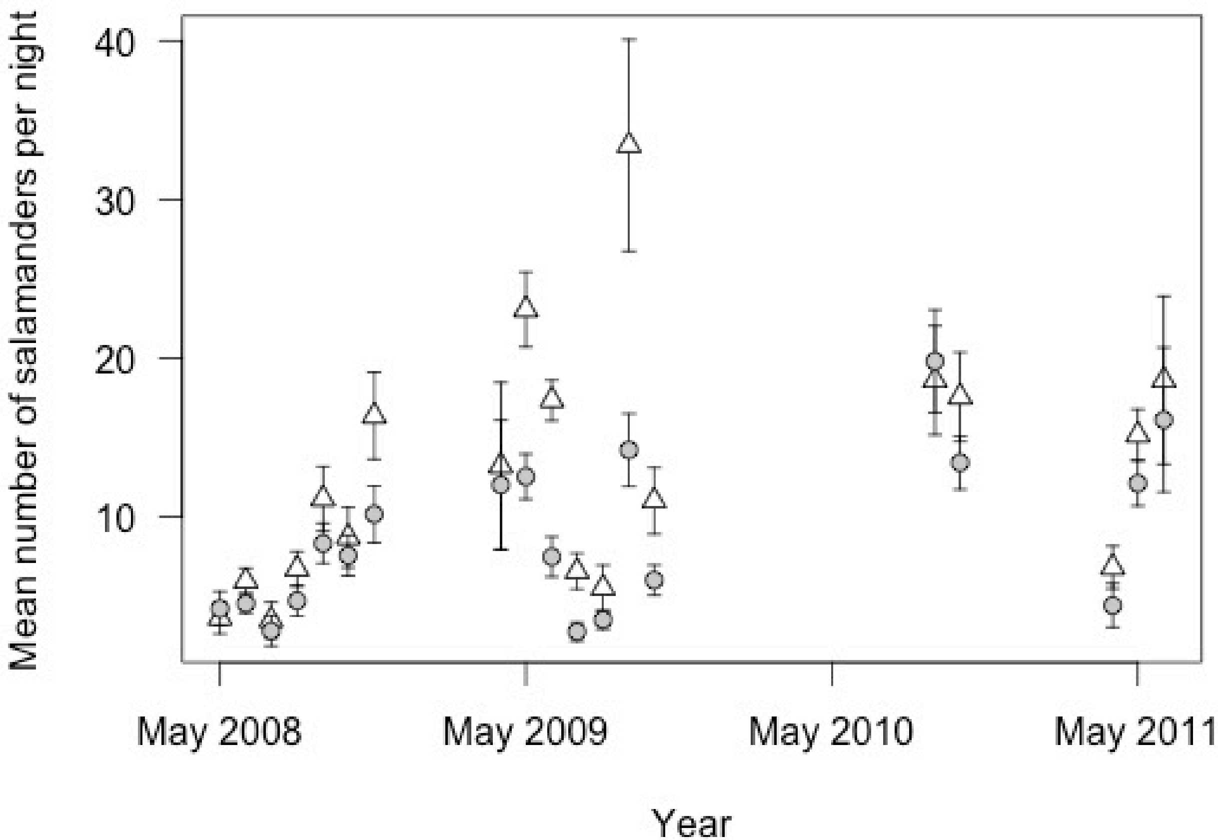
Measurement Errors



We've all seen plots like this before.

What does it mean to have an error bar?

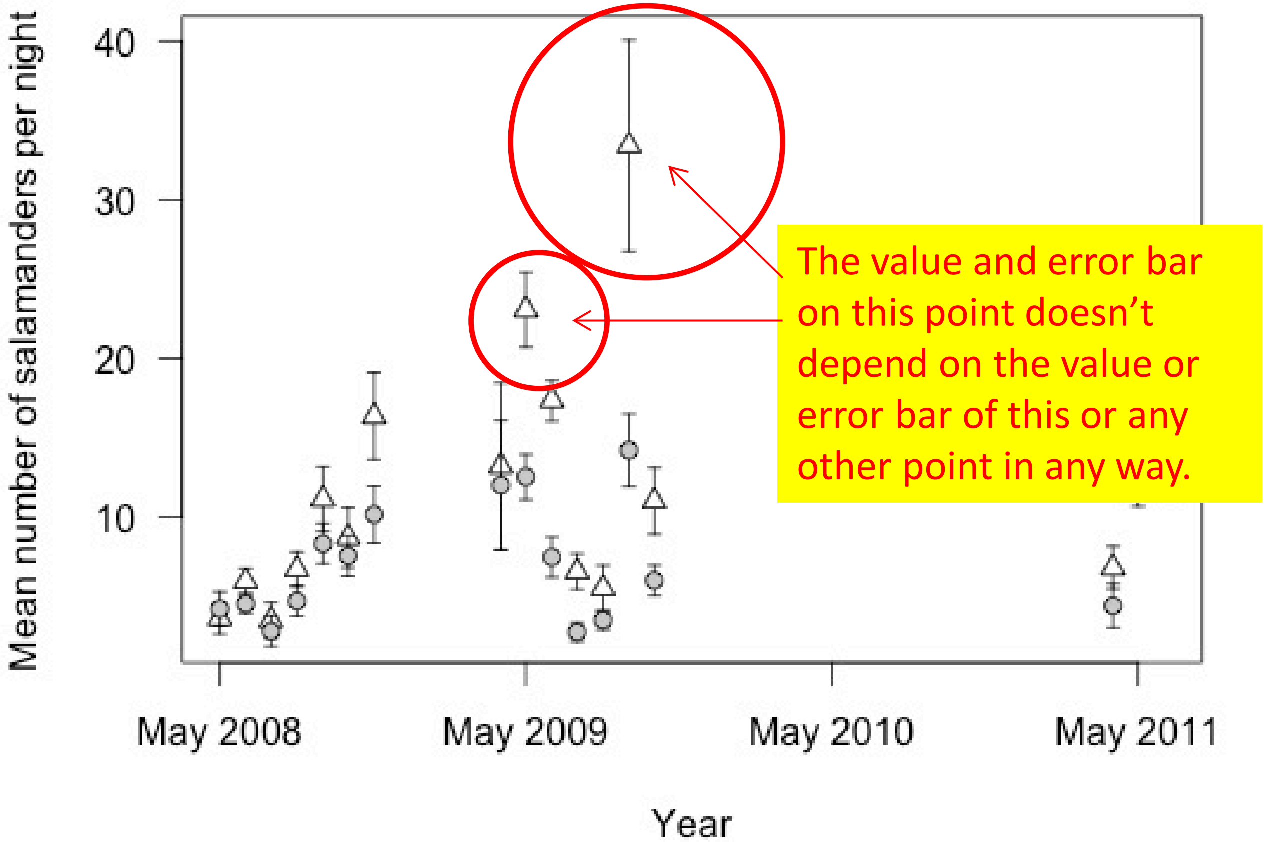
Measurement Errors



Key assumptions when looking at a plot like this:

1. The points are independent.
2. The errors are normally distributed.
3. The error bars are 1-sigma errors.

Measurement Errors



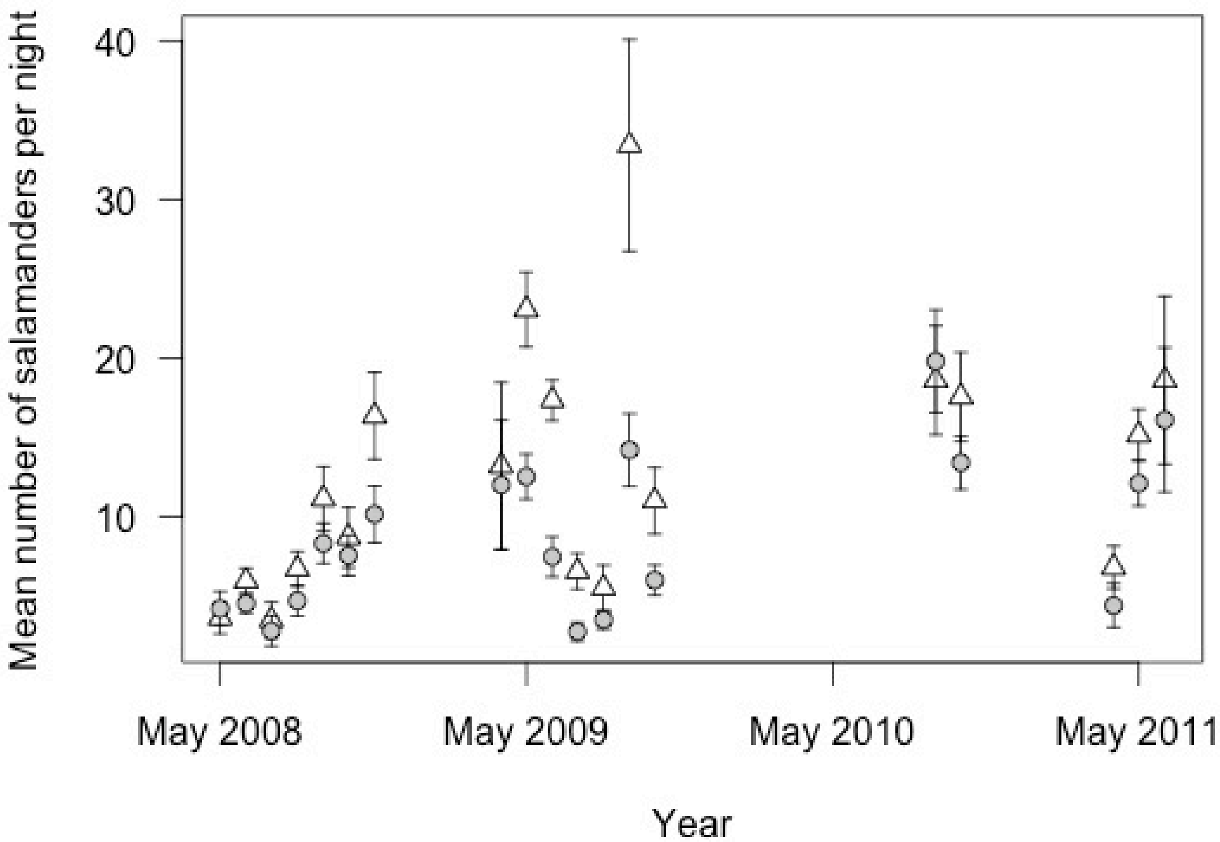
Key assumptions when looking at a plot like this:

1. The points are independent.

2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

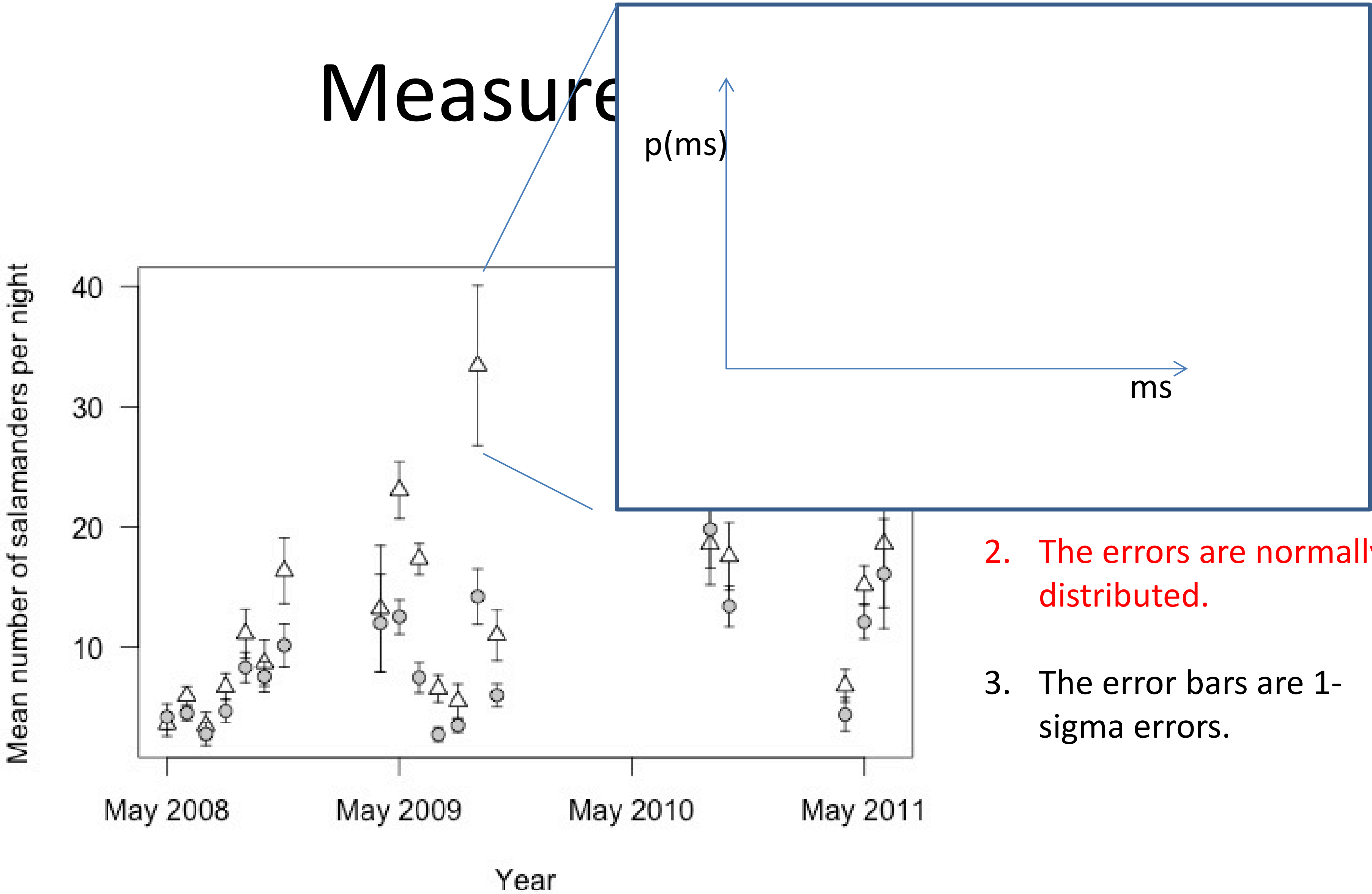
Measurement Errors



Key assumptions when looking at a plot like this:

1. The points are independent.
2. The errors are normally distributed.
3. The error bars are 1-sigma errors.

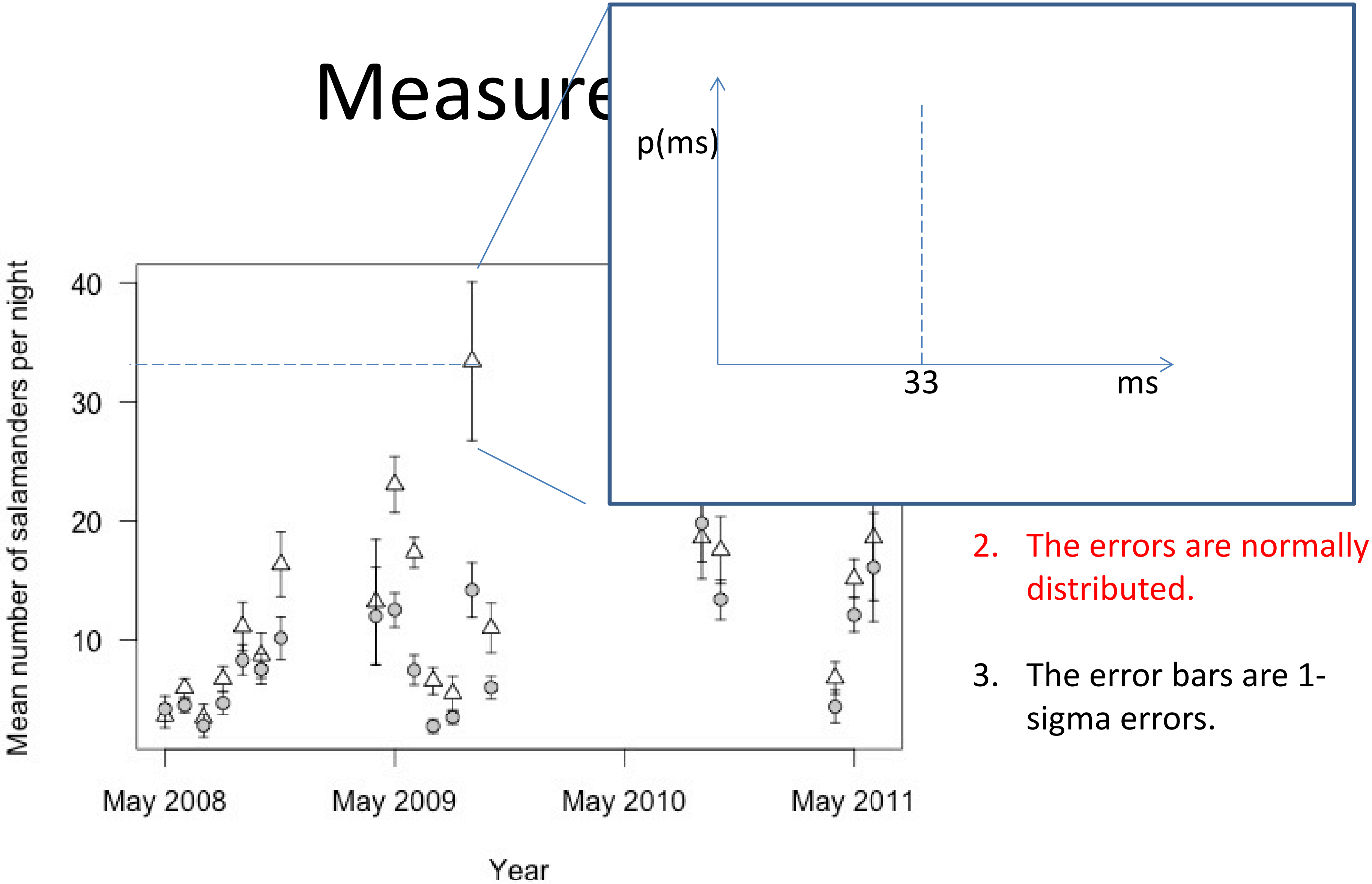
Measure



2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

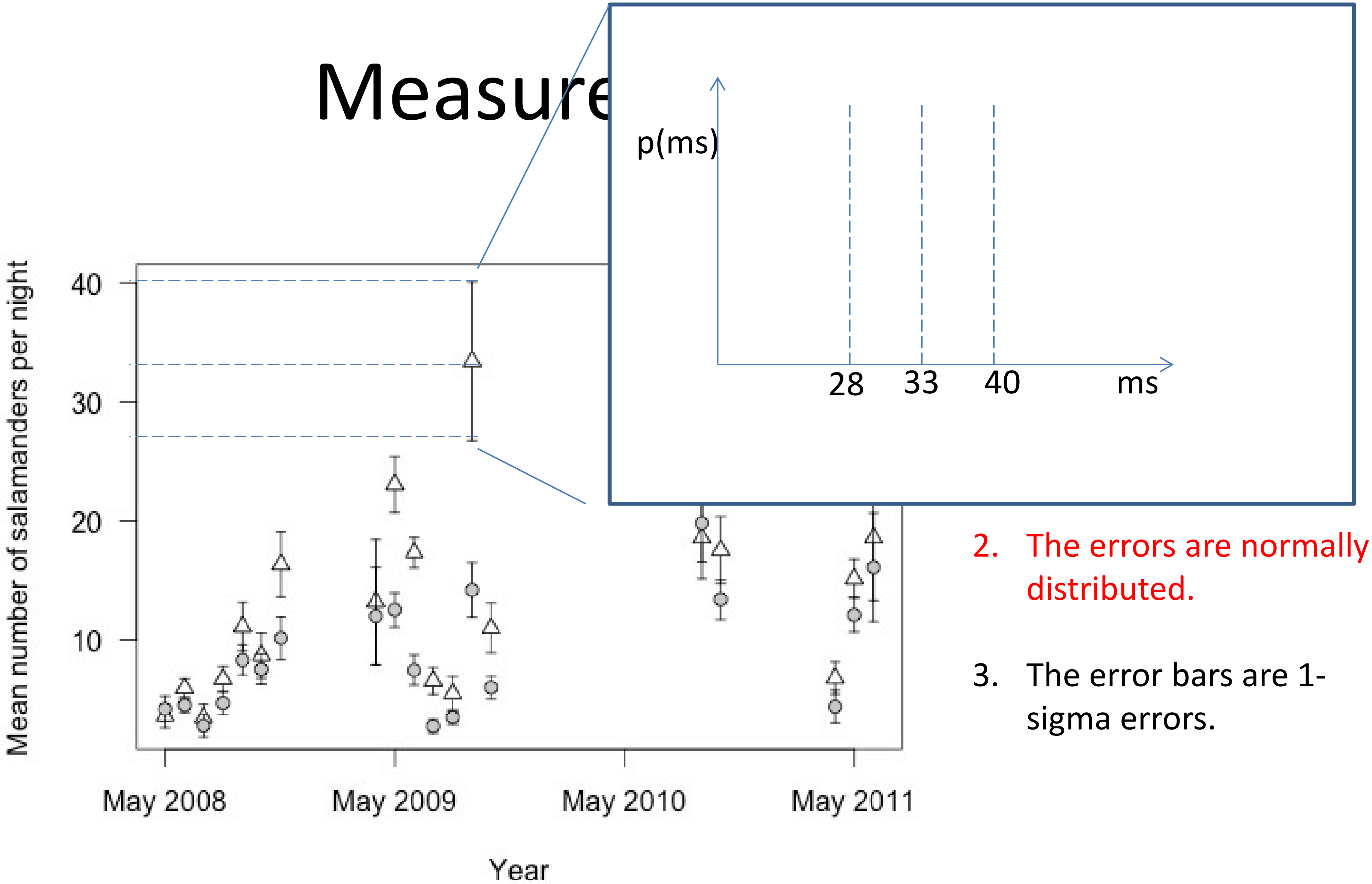
Measure



2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

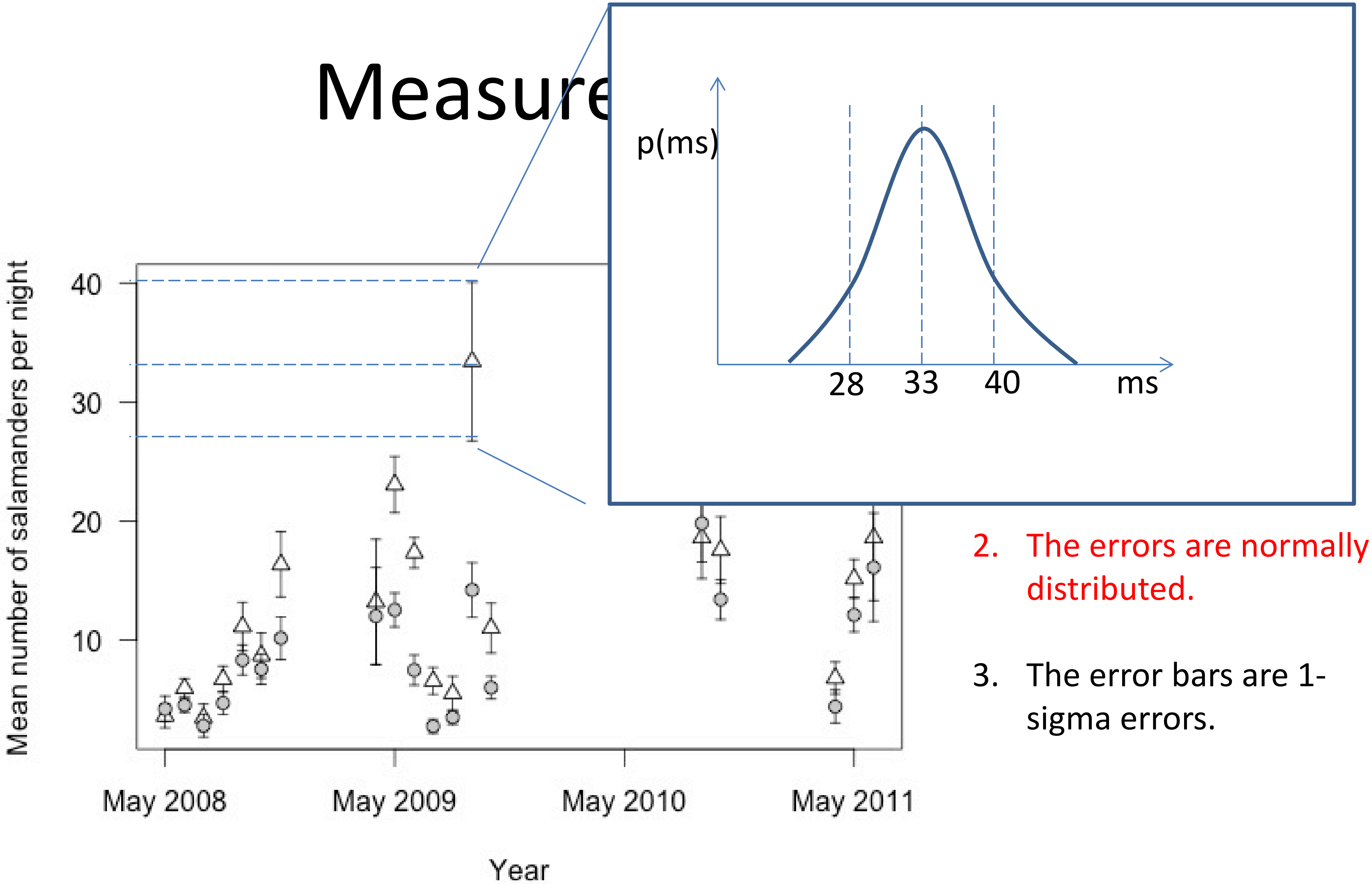
Measure



2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

Measure



2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

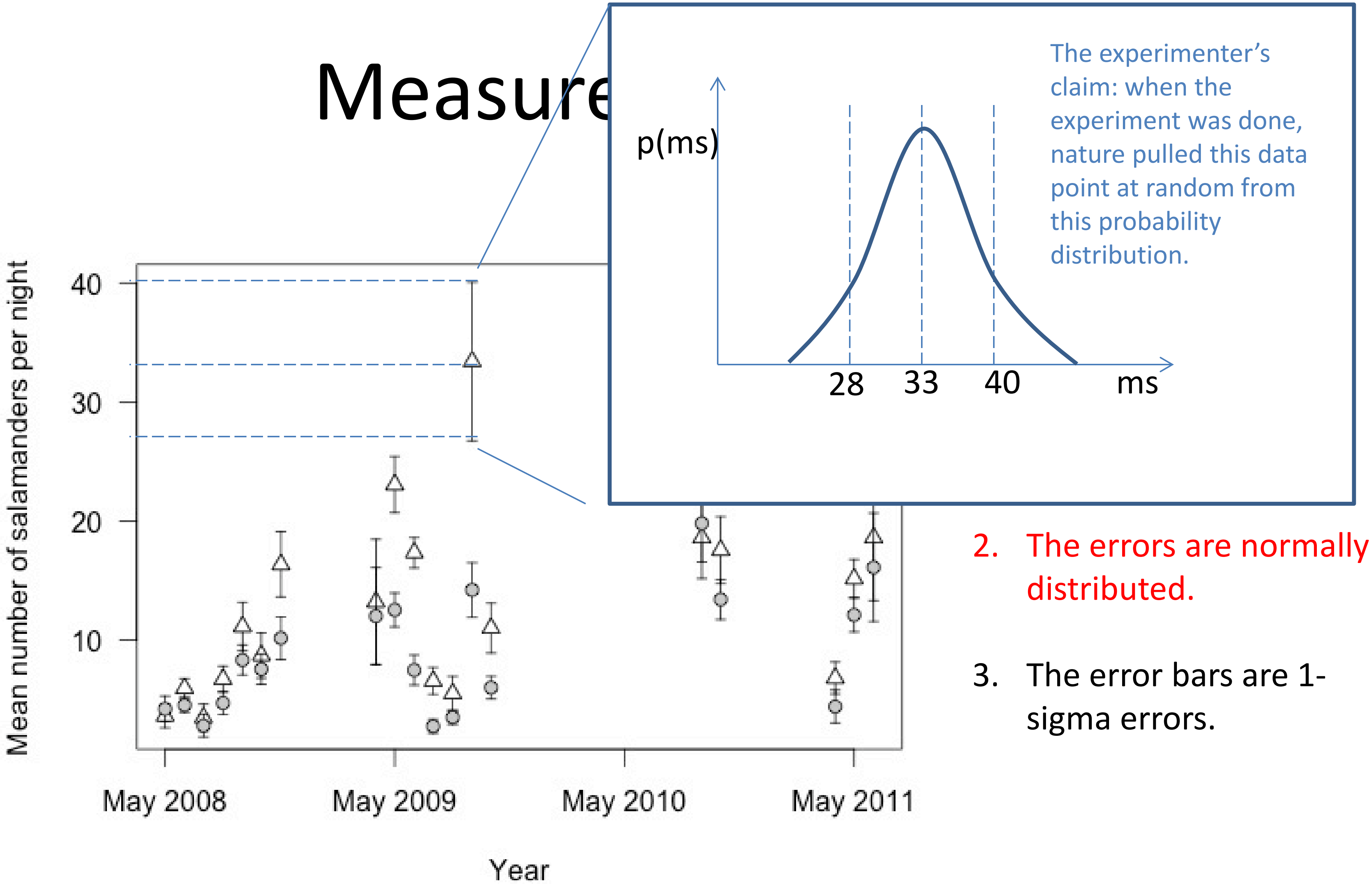
Aside: Why a Normal Distribution?

Answer: because of the Central Limit Theorem.

Central limit theorem: The distribution of an average tends to be Normal, even when the distribution from which the average is computed is decidedly non-Normal.

Connection to data and errors: Data points are *typically* made up of the average of many competing random processes. As a result, the central limit theorem states that the underlying probability distribution of each data point is drawn from a Gaussian (normal) distribution.

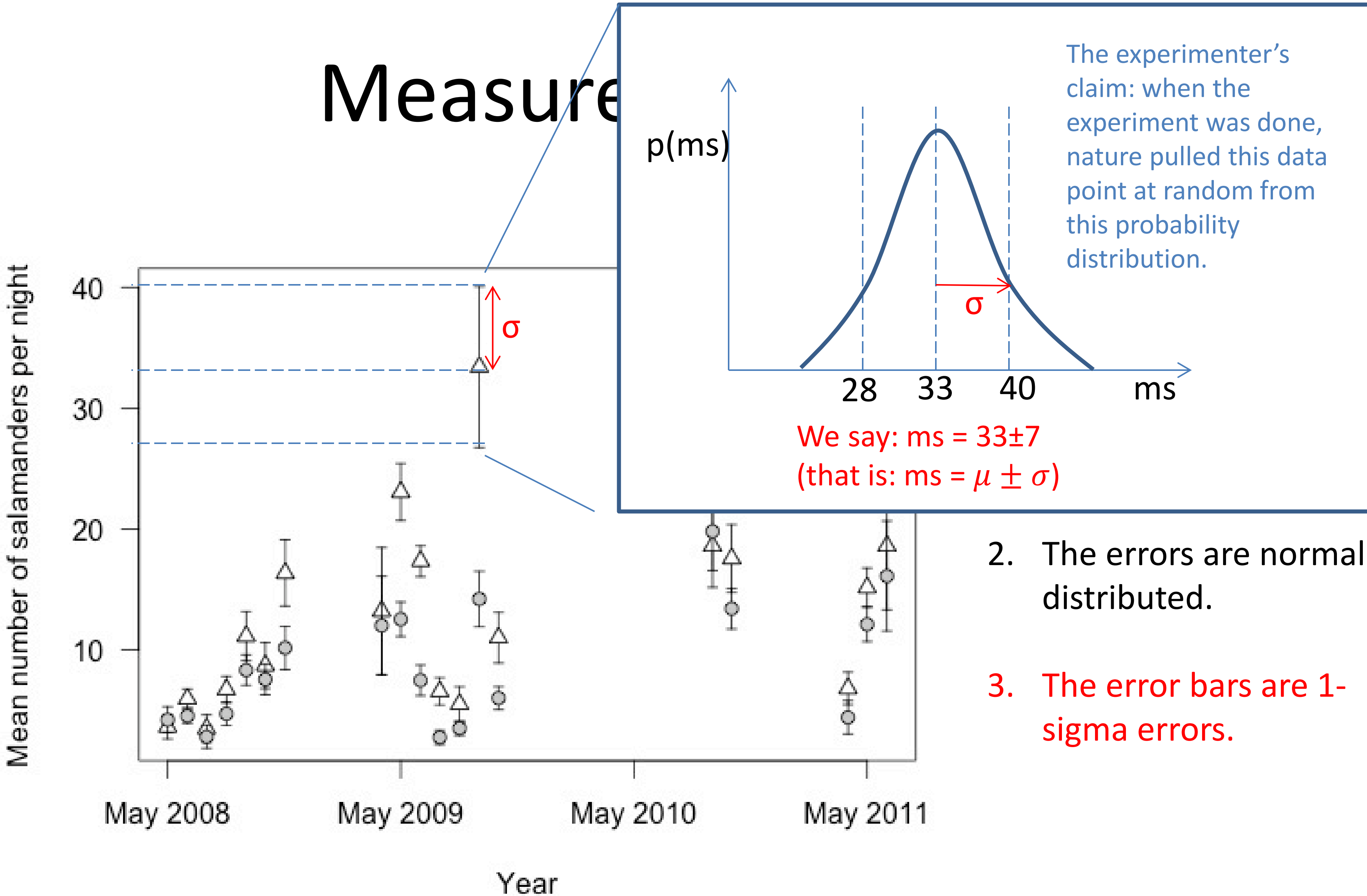
Measure



2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

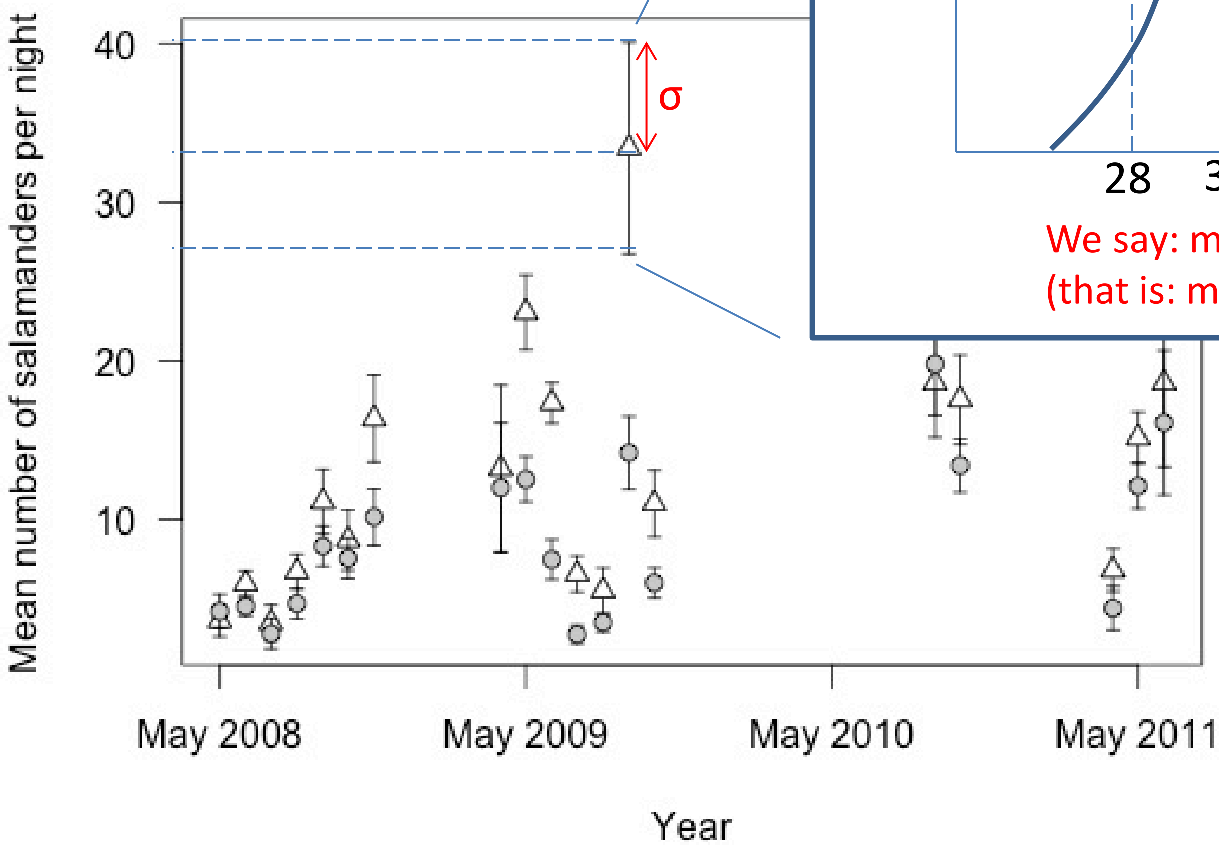
Measure

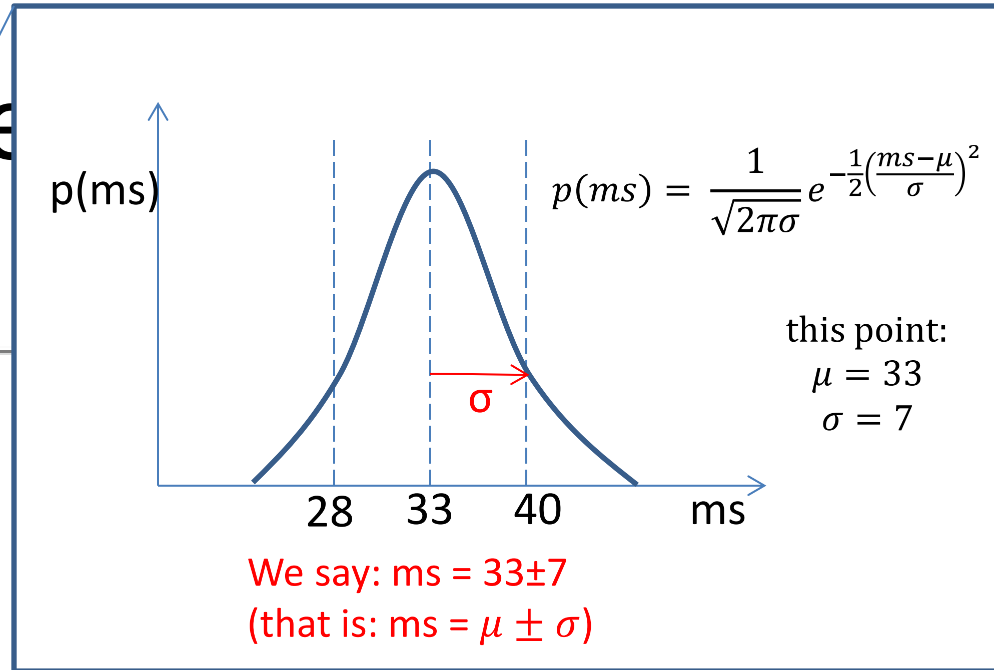


2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

Measure

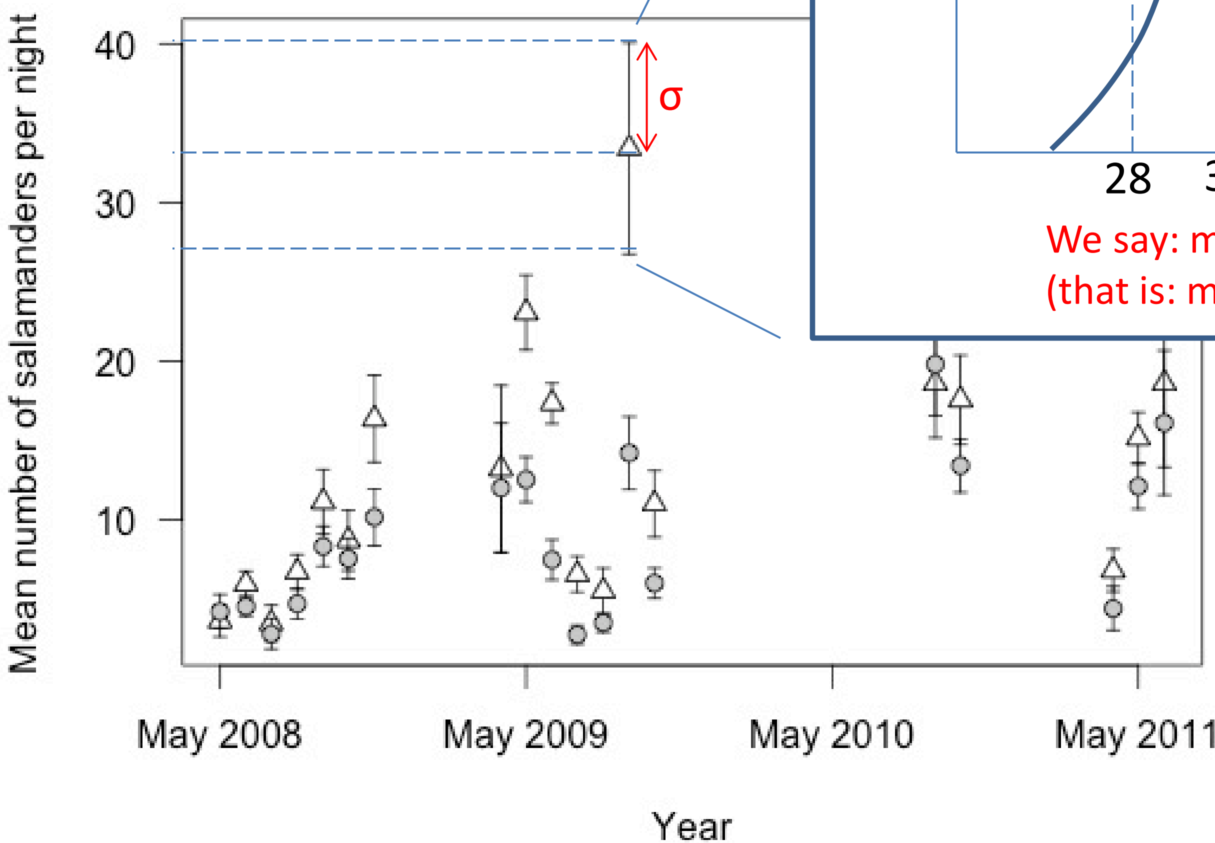


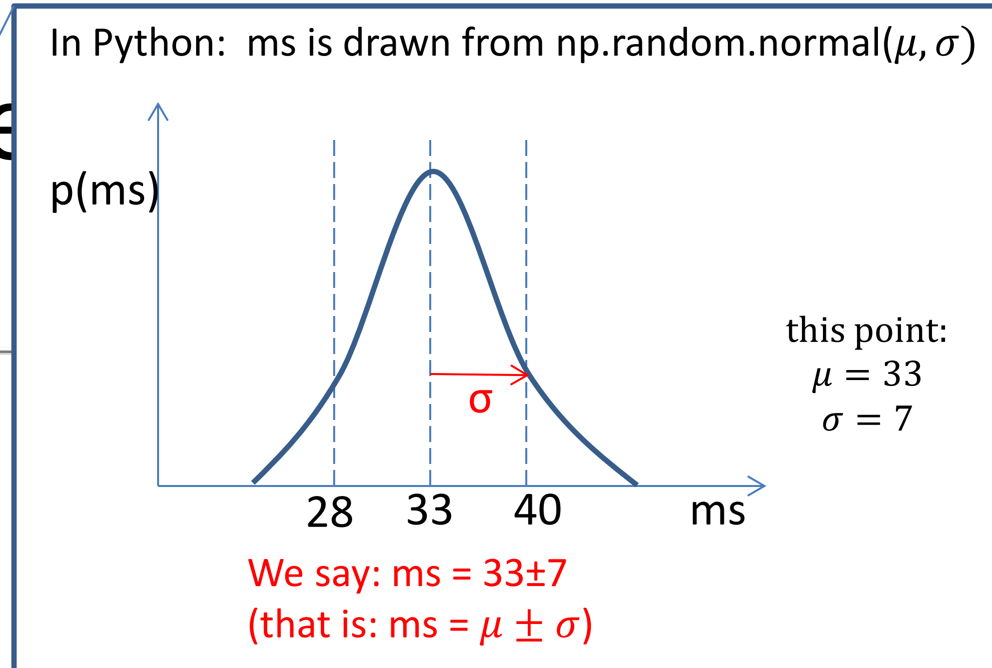


2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

Measure





2. The errors are normally distributed.

3. The error bars are 1-sigma errors.

Connection to Propagation of Uncertainties

- In the discussion in the notes we had the measurement sets [u] and [v] which yielded:
  - $u = \bar{u} \pm \sigma_u$; and
  - $v = \bar{v} \pm \sigma_v$

- If, knowing this, we want to draw values from [u] and [v] we can simply do:
  - usim = np.random.normal($\bar{u}$,σ_u)
  - vsim = np.random.normal($\bar{v}$,σ_v)

Exercise #2

- Given measurements of u and v as follows:
  - $u = 10 \pm 4$
  - $\mathrm{v} = 16 \pm 6$

  Use your simulation tools to find the value and error for $f(u, v) = u + v$.

  [Hint: the standard deviation (sigma) of a normally distributed set of data in a numpy array can be found with the array's own std() method.
  ex, my_gaussian_data.std()]

Simulating propagation of errors

- This is enormously powerful.
- Through simulations we can now easily model how errors propagate through incredibly nonlinear systems.
- Of course, the simulations are only as valid as the input assumptions!

Exercise #3

- A student models her data using a standard power-law model of the form

$$I(\nu) = A\left(\frac{\nu}{\nu_0}\right)^{\alpha}$$

where $\nu_0 = 10 \pm 3, A = 5 \pm 0.5, \alpha = 0.9 \pm 0.2.$

What is the value and error of I at $\nu = 5.5$?