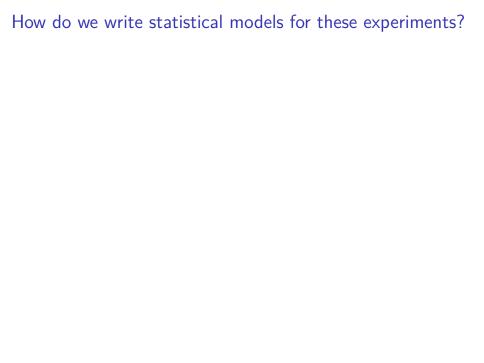
Lecture 17 - Translating statistical models into

code





Let's translate these models into code!

- ▶ There is a general recipe for using maximum likelihood in code
 - ▶ load data
 - write a custom likelihood function
 - maximize the likelihood, but actually we minimize the negative log likelihood, to estimate parameters

Loading data

We've done this a number of times now!

for R:

```
data=read.table('dataFile.txt',header=TRUE,sep='\t')
```

for Python:

```
import pandas
```

```
data=pandas.read_csv('dataFile.txt',header=0,sep='\t')
```

Defining custom functions

We need to generate a function that takes observations as input and returns a likelihood of our model given the data.

- ► First we define the function, give it a variable name, and specify arguments to be expected
- ▶ Inside the function we need to accomplish the following tasks:
 - define parameter values
 - calculate expected values based on the parameter values
 - calculate the likelihood given the observations and expected values (based on the model parameters)
 - return the likelihood value that we've calculated

Defining a custom function

```
import pandas
                              import numpy
                              from scipy.optimize
nllike<-function(p,x,y){
                              import minimize
     B0=p[1]
                              from scipy.stats import
     B1=p[2]
                              norm
     sig=p[3]
     pred=B0+B1*x
                              def nllike(p,obs):
   nll=-sum(dnorm(x=y,mean=
                                    B0=p[0]
   pred,sd=sigma,log=TRUE))
                                    B1=p[1]
     return(nll)
                                    sig=p[2]
}
                                    pred=B0+B1*obs.x
                                  nll=-1*norm(pred,sig).
                                    logpdf(obs.y).sum()
                                    return nll
```

better formatted code for R

```
nllike<-function(p,x,y){
   B0=p[1]
   B1=p[2]
   sigma=exp(p[3])
   pred=B0+B1*x
   nll=-sum(dnorm(x=y,mean=pred,sd=sigma,log=TRUE))
   return(nll)
}</pre>
```

better formatted code for Python

```
import pandas
import numpy
from scipy.optimize import minimize
from scipy.stats import norm
def nllike(p,obs):
  B0=p[0]
  B1=p[1]
  sig=p[2]
  pred=B0+B1*obs.x
  nll=-1*norm(pred, sig).logpdf(obs.y).sum()
  return nll
```

Optimization in general

- ► The development of algorithms behind optimization is a science unto itself.
- ▶ The goal of these alogrithms is to search a *n*-dimensional (where *n* is the number of parameters) space to find the combination of parameters that minimizes the negative log likelihood.
- ▶ There are three major classes of minimization routines:
 - grid search
 - derivative dependent
 - derivative independent

Optimization in code

We pass our custom function as an argument to a function that does minimization in order to estimate the most likely parameter values given our data.

```
guess=numpy.array([1,1,1])
guess=c(1,1,1)
fit=minimize(nllike,guess
fit=optim(par=guess,fn= ,method='Nelder-Mead',
nllike,x=x,y=y) options={'disp': True},
args=df)
```

Challenge: Replicate Kelly et al. 2014

Data available on Sakai in Week 9

Estimate the likelihood of their candidate models using your own code

Do your likelihoods support their inference?