Python Reference

Factorial

Compute $n!$ as an integer. This example computes 20!:

```
import math
print(math.factorial(20))
```

Choose

As of Python 3.8, you can compute $\binom{n}{m}$ from the math module. This example computes $\binom{10}{5}$:

```
import math
print(math.comb(10, 5))
```

Natural Exponent

Calculate e^x. For example this computes e^3

```
import math
print(math.exp(3))
```

SciPy Stats Library

SciPy is a free and open source library for scientific computing that is built on top of NumPy. You may find it helpful to use SciPy to check the answers you obtain in the written section of your problem sets. NumPy has the capability of drawing samples from many common distributions (type `help(np.random)` in the python interpreter), but SciPy has the added capability of computing the probability of observing events, and it can perform computations directly on the probability mass/density functions.

Binomial

Make a Binomial Random variable X and compute its probability mass function (PMF) or cumulative density function (CDF). We love the scipy stats library because it defines all the functions you would care about for a random variable, including expectation, variance, and even things we haven't talked about in CS109, like entropy. This example declares $X \sim \text{Bin}(n = 10, p = 0.2)$. It calculates a few statistics on X. It then calculates $P(X = 3)$ and $P(X \leq 4)$. Finally it generates a few random samples from X:

```
from scipy import stats
X = stats.binom(10, 0.2) # Declare X to be a binomial random variable
print(X.pmf(3))          # P(X = 3)
print(X.cdf(4))          # P(X <= 4)
print(X.mean())          # E[X]
print(X.var())           # Var(X)
print(X.std())           # Std(X)
print(X.rvs())           # Get a random sample from X
print(X.rvs(10))         # Get 10 random samples form X
```

From a **terminal** you can always use the "help" command to see a full list of methods defined on a variable (or for a package):

```
from scipy import stats
X = stats.binom(10, 0.2) # Declare X to be a binomial random variable
help(X)                  # List all methods defined for X
```

Poisson

Make a Poisson Random variable Y. This example declares $Y \sim \text{Poi}(\lambda = 2)$. It then calculates $P(Y = 3)$:

```
from scipy import stats
Y = stats.poisson(2)  # Declare Y to be a poisson random variable
print(Y.pmf(3))       # P(Y = 3)
print(Y.rvs())        # Get a random sample from Y
```

Geometric

Make a Geometric Random variable X, the number of trials until a success. This example declares $X \sim \text{Geo}(p = 0.75)$:

```
from scipy import stats
X = stats.geom(0.75)  # Declare X to be a geometric random variable
print(X.pmf(3))       # P(X = 3)
print(X.rvs())        # Get a random sample from Y
```

Normal

Make a Normal Random variable A. This example declares $A \sim N(\mu = 3, \sigma^2 = 16)$. It then calculates $f_Y(0)$ and $F_Y(0)$. **Very Important!!!** In class, the second parameter to a normal was the variance (σ^2). In the scipy library, the second parameter is the standard deviation (σ):

```
import math
from scipy import stats
A = stats.norm(3, math.sqrt(16)) # Declare A to be a normal random variable
print(A.pdf(4))                  # f(3), the probability density at 3
print(A.cdf(2))                  # F(2), which is also P(Y < 2)
print(A.rvs())                   # Get a random sample from A
```

Exponential

Make an Exponential Random variable B. This example declares $B \sim \text{Exp}(\lambda = 4)$:

```
from scipy import stats
# `λ` is a common parameterization for the exponential,
#  but `scipy` uses `scale` which is `1/λ`
B = stats.expon(scale=1/4)
print(B.pdf(1))             # f(1), the probability density at 1
print(B.cdf(2))             # F(2) which is also P(B < 2)
print(B.rvs())              # Get a random sample from B
```

Beta

Make an Beta Random variable X. This example declares $X \sim \text{Beta}(\alpha = 1, \beta = 3)$:

```
from scipy import stats
X = stats.beta(1, 3)  # Declare X to be a beta random variable
print(X.pdf(0.5))     # f(0.5), the probability density at 1
print(X.cdf(0.7))     # F(0.7) which is also P(X < 0.7)
print(X.rvs())        # Get a random sample from X
```