Modelos Bayesiano usando PyMC

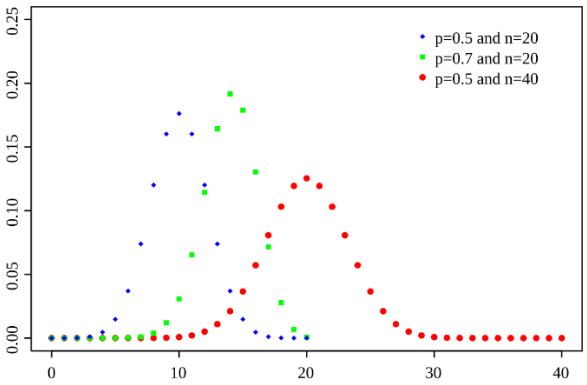
PyMC es una librería de programación probabilística para Python que permite a sus usuarios ajustar modelos Bayesianos usando varios métodos numéricos, entre los cuales destacan los métodos de Montecarlo basados en cadenas de Márkov (MCMC por sus siglas en inglés) y *variational inference* (VI).

En el video Fernando comenta que la estadística bayesiana tiene la ventaja de ser más intuitiva ya que se enfoca en la confianza de las observaciones.

El **Teorema de Bayes** permite actualizar nuestras creencias o probabilidades sobre un evento (A) después de considerar una nueva evidencia o información (B). Es decir, parte de lo que sabemos sobre el evento original para inferir la nueva probabilidad teniendo en cuenta la nueva evidencia. Aquí nos apoyamos de PyMC ya que nos evita la parte analítica del problema.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Distribución binomial: Donde hay N experimentos que resultan en un éxito o fracaso con probabilidad p, por mencionar un ejemplo el lanzamiento de una moneda.



Problema: El video ilustra los modelos bayesianos con un ejemplo sobre la cantidad de gente contagiada de SARS-CoV-2 en Santiago, Chile, a través de un test de anticuerpos.

Se toma una muestra aleatoria de 50 personas a los que se les aplica dicho test y se obtiene que 40 son negativas y 10 positivas. Una vez dicho esto, se proponen 3 modelos:

1. **Se asume que los test realizados son correctos**

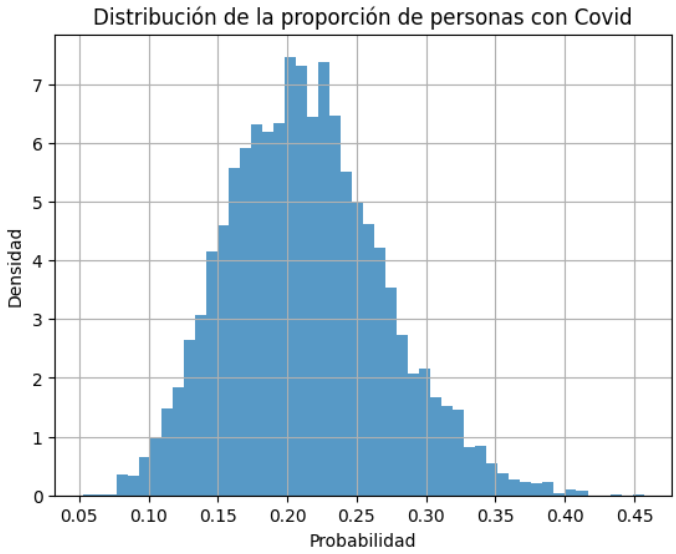
   Se asume que el 20% de la muestra tuvo Covid

```
tests_totales = 50
tests_positivos = 10
```

```
with pm.Model() as modelo_test_perfecto:
  prob = pm.Uniform(name='prob', # Variables a estimar
                    lower=0,
                    upper=1)
  casos_positivos = pm.Binomial(name='casos_positivos', # Definicion de los datos a estimar
                                n=tests_totales,
                                p=prob,
                                observed=tests_positivos)
  trace_test_perfecto = pm.sample(3000) # Saca 3000 muestras
                                        # Aqui se define el modelo matématico
```

Se utiliza la distribución uniforme porque primero consideramos que todos los valores son igual de probables. (Priori)

El objetivo del modelo es **estimar la probabilidad de éxito** en las pruebas. Es decir, se quiere saber cuál es la probabilidad de que, al realizar una prueba, el resultado sea positivo (como los 10 que observamos).



Si lo graficamos observamos la distribución de la proporción de las personas con Covid y efectivamente, que el 20% es el valor más probable.

2. El test da a veces falsos positivos

El laboratorio que creo el test hizo 100 pruebas y nos informa que la tasa de falsos positivos es de 10%. Por lo que se supone que, de los 10 test positivos, 5 son falsos. Se modifica el modelo:

```
with pm.Model() as modelo_fp:
  prob_cov = pm.Uniform(name='prob_cov', # Variables a estimar
                        lower=0,
                        upper=1)
  prob_fp = 0.1 # Probabilidad de falsos negativos
  prob_test_positivo = prob_cov + (1-prob_cov)*prob_fp
  casos_positivos = pm.Binomial(name='casos_positivos',
                                n=tests_totales,
                                p=prob_test_positivo,
                                observed=tests_positivos)
  modelo_fp = pm.sample(3000)
```

Ajusta la estimación de la probabilidad de infección en un escenario donde las pruebas pueden fallar, a través de la ecuación de la probabilidad de un test positivo que es igual a la probabilidad de tener Covid mas la probabilidad de no tener Covid, por la probabilidad de un falso positivo.

3. No tenemos un número exacto de falsos positivos (Incertidumbre).

Aquí la probabilidad de fasos negativos ya no es 0.1 sino es una variable con incertidumbre que podemos poner como una distribución bimomial.

```
lab_fp_observados = 10
lab_tests_hechos = 100

with pm.Model() as modelo_incetidumbre:
  prob_fp = pm.Uniform(name='prob_fp',
                       lower=0,
                       upper=1)

  test_falsos_positivos = pm.Binomial(name='test_falsos_positivos',
                                      n=lab_tests_hechos,
                                      p=prob_fp,
                                      observed=lab_fp_observados)

  prob_cov = pm.Uniform(name='prob_cov',
                        lower=0,
                        upper=1)
  prob_test_positivo = prob_cov + (1-prob_cov)*prob_fp
  casos_positivos = pm.Binomial(name='casos_positivos',
                                n=tests_totales,
                                p=prob_test_positivo,
                                observed=tests_positivos)
  trace_modelo_incertidumbre = pm.sample(3000)
```

Resultados: Como en el video, al calcular la probabilidad de que menos del 20% de las personas este infectadas, nos dieron estos resultados:

Modelo 1 (Test perfecto): 41.98%

Aquí no se consideran errores en las pruebas, el modelo es más conservador en su estimación, lo que explica el valor más bajo.

Modelo 2 (Falsos positivos): 88.5%

Al incluir los falsos positivos, algunos de los resultados positivos observados en las pruebas pueden no corresponder a personas realmente infectadas. Esto hace que el modelo infiera que la verdadera probabilidad de infección podría ser más baja.

Modelo 3 (Incertidumbre): 87.85%

En este modelo a pesar de que se agrego la incertidumbre de los falsos positivos no afecta significativamente la estimación general de la prevalencia de la infección, lo cual es un indicio de que el número observado de falsos positivos está bien ajustado.

Conclusiones: Los modelos bayesianos son extremadamente útiles para manejar incertidumbre y errores en los datos. En este caso, al incorporar la posibilidad de falsos positivos y la incertidumbre en su tasa, podemos obtener estimaciones más realistas de los infectados de Covid.

La flexibilidad de los modelos bayesianos permite ajustar los resultados observados con base en creencias previas, lo que es especialmente valioso en situaciones con poca información o donde las pruebas tienen un margen de error.