

Problema 1

(Problema en Ecolog  a) Sea X_1, X_2, \dots, X_m variables aleatorias donde X_i denota el n  mero de individuos de una especie en cierta regi  n. Suponga $X_i|N, p \sim \text{Binomial}(N, p)$, entonces

$$f(\bar{x}|N, p) = \prod_{i=1}^m \frac{N!}{x_i!(N - x_i)!} p^{x_i} (1 - p)^{N - x_i}$$

Asumiendo la distribuci  n a priori $p \sim \text{Beta}(\alpha, \beta)$ y $N \sim h(\cdot)$, donde h es una dist. discreta en $\{0, 1, 2, \dots, N_{\max}\}$, se tiene definida la distribuci  n posterior $f(N, P|\bar{x})$.

A partir del algoritmo MH, simule valores de la distribuci  n posterior usando un kernel h  brido.

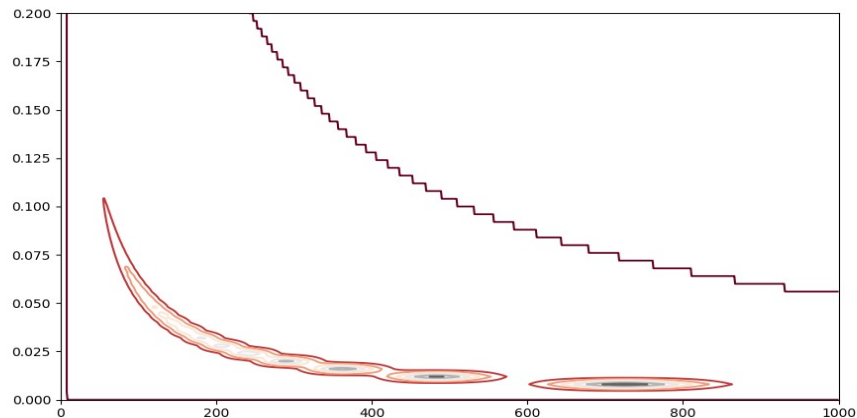
Lo primero que estudiaremos ser   la forma expl  cita de la distribuci  n posterior, de donde desp  es podremos determinar una factorizaci  n para un kernel Gibbs para p y un posiblemente otras propuestas. Tenemos que,

$$\pi(N, p|\bar{x}) = f(\bar{x}|N, p)f(N)f(p) \quad (1)$$

$$= \prod_{i=1}^m \frac{N!}{x_i!(N - x_i)!} p^{x_i} (1 - p)^{N - x_i} \frac{p^{\alpha-1} (1 - p)^{\beta-1}}{\text{Beta}(\alpha, \beta)} \frac{1}{N_{\max} + 1} \quad (2)$$

$$\propto p^{(\alpha + \sum_i x_i) - 1} (1 - p)^{(\beta + mN - \sum_i x_i) - 1} \prod_{i=1}^m \frac{N!}{x_i!(N - x_i)!} \quad (3)$$

Visualizemos las curvas de nivel de nuestra distribuci  n posterior para darnos una idea de la forma de las muestras que esperamos simular.



Notemos que se pueden observar distintas modas en la distribución posterior, de modo que en las muestras posiblemente observaremos 'islas'. Posiblemente nuestra simulación pueda converger a una sola de las modas.

Ahora notemos que de aquí podemos factorizar la distribución posterior de la siguiente manera,

$$\pi(N, p | \bar{x}) \propto f(p | N, \bar{x}) f(N | \bar{x}),$$

obteniendo la siguiente una propuesta Gibbs con la siguiente distribución.

(Propuesta 1) Gibbs Sampler - prob .3

$$p' | N, \bar{x} \sim \text{Beta}(\alpha + \sum_{i=1}^m x_i, \beta + mN - \sum_{i=1}^m x_i)$$

Siendo esta una propuesta Gibbs, no rechaza ninguna de las propuestas generadas. Podemos separar en dos categorías las propuestas que intuitivamente (sin un argumento formal, solo cómo comentario) representan la 'exploración' del soporte de la distribución (explorar regiones poco visitadas donde podría encontrarse un región donde se acepten las poropuestas) y la 'explotación' donde sabemos que nuestras propuestas serán casi siempre aceptadas, aunque no exploren mucho otras regiones. De esta manera decimos que como la propuesta de Gibbs se concentra en lugares 'conocidos' de la distribución, entonces es de 'explotación'.

La siguiente propuesta que consideramos es la utilizando la distribución a priori de p :

(Propuesta 2) - prob .1

$$p' \sim \text{Beta}(\alpha, \beta)$$

Despues de varios experimentos notamos que el porcentaje de rechazo de esta propuesta siempre es alrededor del 95 %, por lo que es muy poco eficiente, aunque podría explorar regiones del soporte de la distribución posterior. Siendo una propuesta de 'exploración' y le asignamos una probabilidad de .3 de ser escogida como kernel.

Una propuesta que directamente podemos aplicar a nuestra problema es el de una caminata aleatoria con un paso 'Normal' alrededor de nuestro parámetro actual p , tomando una valor de σ fijo,

(Propuesta 3 - Caminata Aleatoria p) - prob .1

$$p' | N, p \sim \mathcal{N}(p, .01)$$

Como escogimos un valor de $\sigma = .01$ (tamaño de paso) pequeño, puesto que despues de varios experimentos notamos que rechaza alrededor del 15 % de la propuestas generadas despues

del **burn-in**. De modo que consideramos a esta propuesta como de 'explotaci  n'.

Siendo esta las propuestas de kernel para el par  metro p . Ahora presentaremos las propuestas que utilizamos para el par  metro N .

(Propuesta 4) - prob .1

$$N' \sim \mathcal{U}_d\{0, 1, \dots, N_{max}\}$$

Esta propuesta es la obtenida al usar la distribuci  n a-priori del par  metro N . Despues de varios experimentos notamos que se rechaza alrededor del 95 % de estas propuestas (igual que la propuesta 'a-priori' de p).

(Propuesta 5) - prob .1

$$N'|N, p \sim N + \text{Poisson}(\lambda := N(1 - p))$$

Escogimos un propuesta Poisson con par  metro $\lambda = N(1 - p)$ esto inspirado te  ricamente por la propuesta Gibbs de un modelo 'Binomial-Beta-Poisson' que similar al de nuestra posterior y emp  ricamente por que rechaza menos propuestas que usar par  metros como $\lambda = N$ o $\lambda = Np$.

(Propuesta 6 - Caminata Aleatoria N) - prob .2

$$N' = N + \epsilon, \mathbb{P}(\epsilon = 1) = \frac{1}{2} = \mathbb{P}(\epsilon = -1)$$

Esta propuesta es de 'explotaci  n' pues el tama  o de paso es muy peque  o, de modo que en regiones conocidas con 'pasos peque  os' se aceptan casi todas la propuestas.

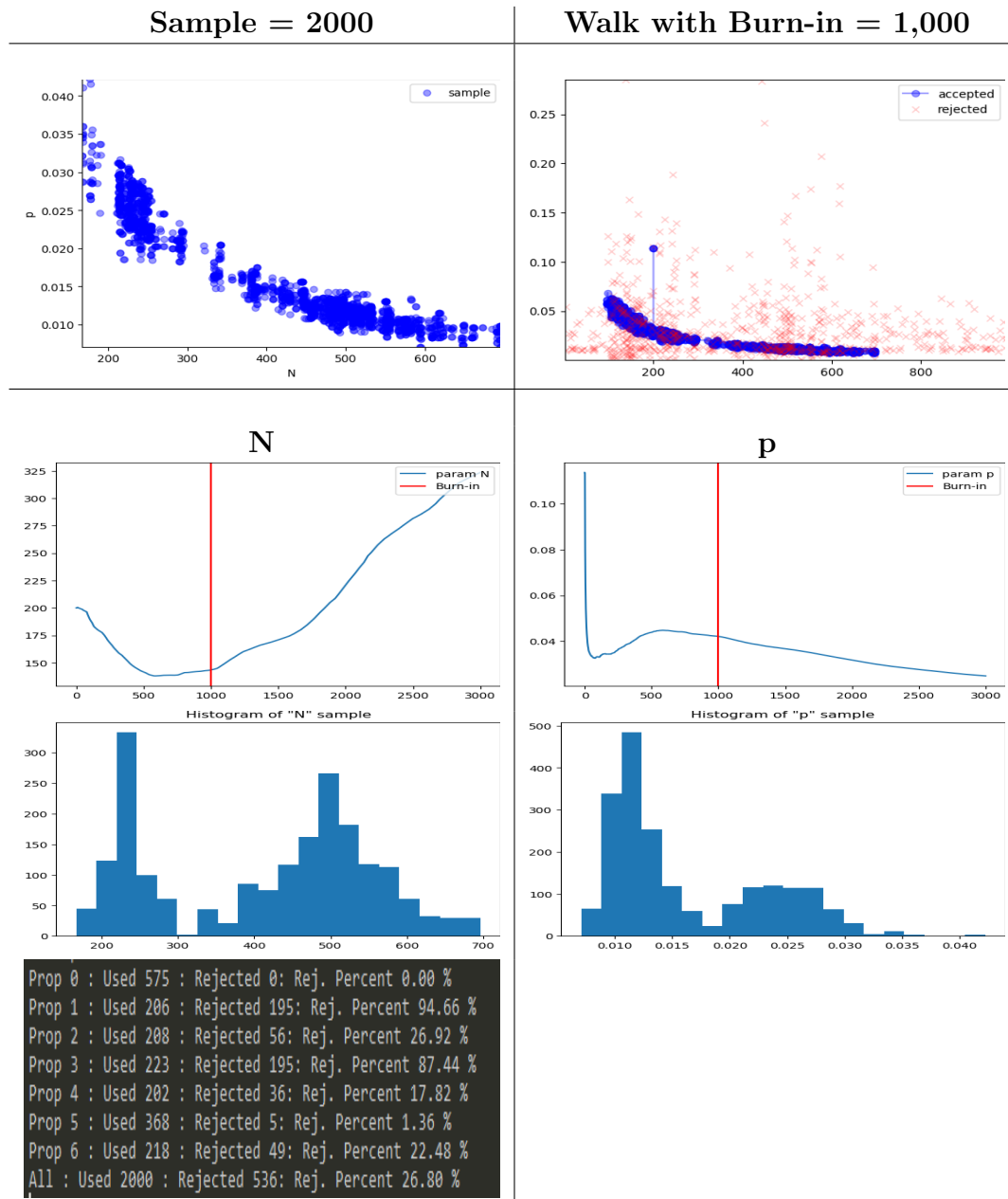
Y finalmente nuestra **propuesta 7 (prob .1)** es la combinaci  n de las propuestas de caminata aleatoria, osea usamos las **propuestas 3 y 6** al mismo tiempo.

Como distribuci  n inicial de nuestro algoritmo MH nos quedamos con la propuesta presentada en el enunciado, puesto que en los resultados se han mostrado diferentes comportamientos a explorar con estos puntos iniciales.

$$p \sim \mathcal{U}(0, 1), \quad N \sim \mathcal{U}_d\{\max(x_i), \dots, N_{max}\}$$

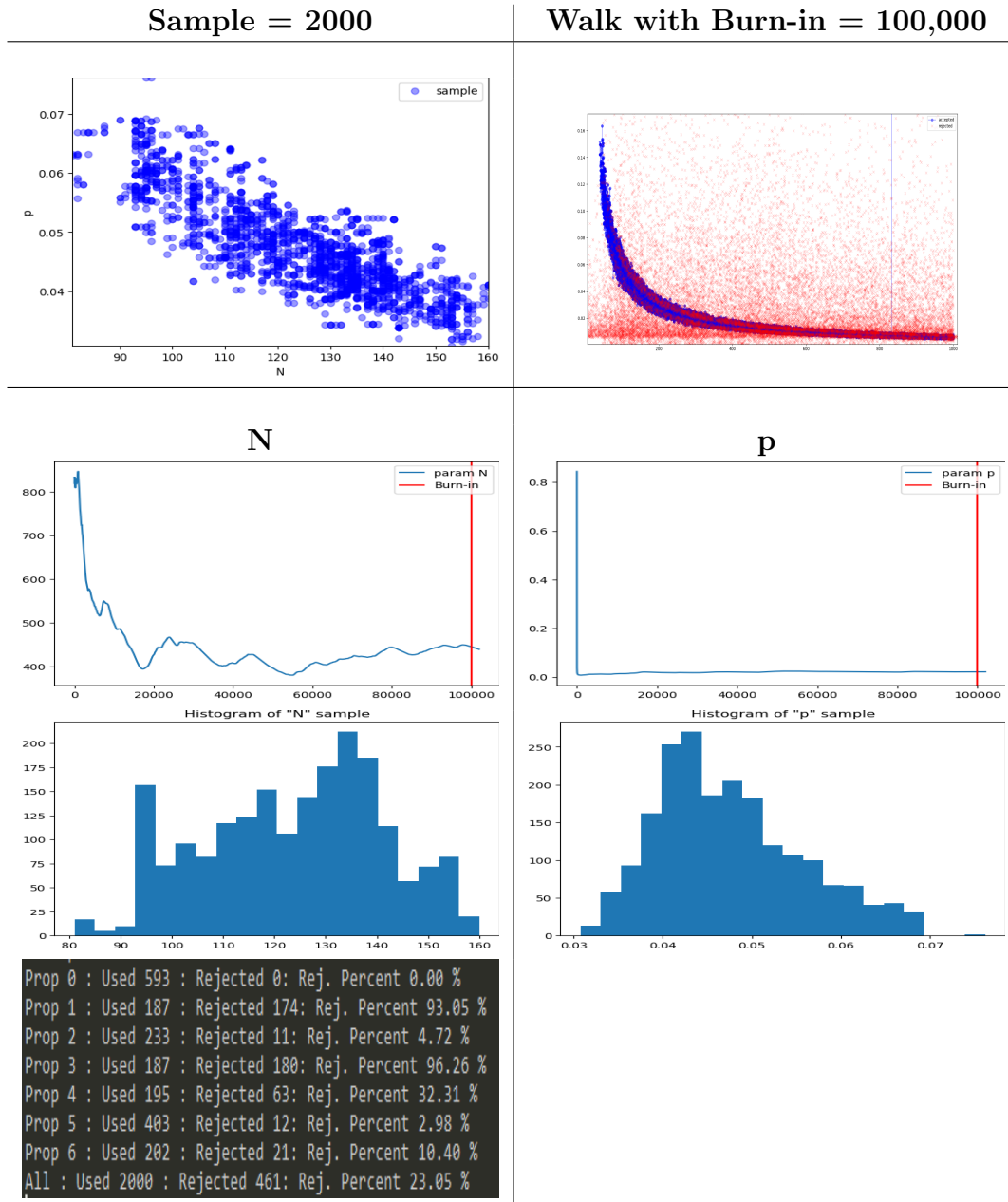
Algo que debemos mencionar sobre la nuestra implementaci  n particular del algoritmo de **Metropolis-Hastings** es que todo se trabaja en el espacio **logaritmico**, aprovechando las implementaciones de **scipy** de la **logpdf** de muchas distribuciones. e.g. **beta.logpdf**(x, α, β). La misma raz  n de aceptaci  n se calcula con el logartimo de las densidades y al final solo se calcula su exponencial para determinar la aceptaci  n o el rechazo. Esto, como se mencion   en clase, evita problemas num  ricos puesto que el **soporte num  rico** de las densidades ser   inmensamente mayor. Todo el c  digo de esta implementaci  n se encuentra en el archivo **metropolis_hastings_hybrid_kernels.py**.

Ahora estudiemos los resultados de usar el algoritmo de MH con Kerneles H bridos, escogiendo cada de las propuestas como se mencion  anteriormente con la notaci n **prob k**. Realizaremos 2 experimentos donde en cada uno obtendremos una muestra de tama o 2000, pero con diferentes valores de **burn-in** 1000 y 100000.



En este primero caso podemos observar que nuestra algoritmo no ha convergido a ninguna de las modas que observamos inicialmente en las curvas de nivel y se encuentra explorando

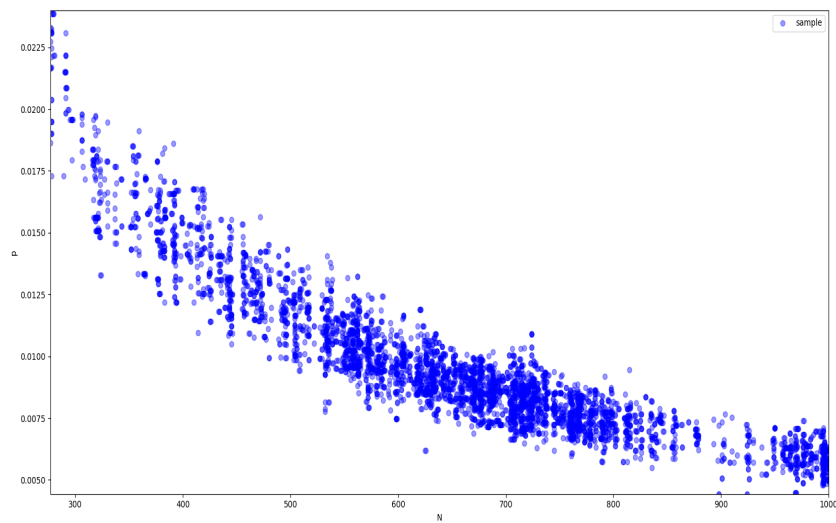
distintas regiones de alta densidad. En las gr  ficas de 'burn-in' donde observamos el cambio de la media, claramente notamos como cambian la distribuci  n a N y p en distintas regiones. Tambi  n en el histograma de muestras individuales observamos claramente dos modas. La eficacia de nuestras propuestas en conjunto es del 74 %, puesto que pesamos las propuestas usando nuestra intuici  n de que hace cada una de ellas. A estos resultados nos refer  amos cuando comentabamos la tasa de rechazo al definir nuestras propuestas.



En el segundo caso desp  s de haber hecho 'burn-in' por mucho tiempo, 100000 iteraciones,

observamos que al final converge nuestra moda en particular con $N \approx 130$ y $p \approx .05$. Esto no quiere decir que m  s adelante no vaya a explorar de nuevo otra regi  n. De la caminata completa donde tambi  n se gr  fic  n las propuestas rechazadas notamos que se explor   todas las regiones que observamos en las curvas de nivel. La eficacia de nuestras propuestas en conjunto es del 77%, puesto que pesamos las propuestas usando nuestra intuici  n de que hace cada una de ellas.

Ahora estudiemos una muestra de tama  o 5000 al haber usado 2000 iteraciones de burn-in que representa la exploraci  n de todas las regiones de 'alta probabilidad' que observamos en las curvas de nivel de la distribuci  n posterior.



Notamos que hay una relaci  n entre los valores simulados de N y p . Esto intuitivamente se entiende por que estamos ajustando un modelo **Binomial** a nuestros 20 datos $\{X_i\}$ y como su media es $\mathbb{E}[X] = Np$, tenemos que hay una relaci  n **inversamente proporcional** entre los par  metros N y p . E.g Si hacemos m  s ensayos Bernoulli, N grande, entonces p debe ser peque  o por que sabemos que la especie que estamos observando no es muy abundante. De esto anterior podr  amos hacer una regresi  n para ajustar una curva.

Ahora para hablar de la distribuci  n de los puntos de nuestra muestra notamos que hay mucha m  s densidad en los valores.

Problema 2

(Estudio de mercado) Se tiene un producto y se realiza una encuesta con el fin de estudiar cu  nto se consume dependiendo de la edad. Sea Y_i el monto de compra y X_i la covariable la cual representa la edad.

Suponga que $Y_i \sim Po(\lambda_i)$ (distribuci  n Poisson con intensidad λ_i)

$$\lambda_i = cg_b(x_i - a)$$

para g_b la siguiente funci  n de liga

$$g_b(x_i - a) = \exp\left(-\frac{x_i^2}{2b_2}\right)$$

Usando Metropolis-Hastings simule de la distribuci  n posterior.

Escribamos expl  citamente la distribuci  n posterior de los par  metros a , b y c de nuestra regressi  n Poisson, suponiendo que nuestros valores de a , b y c estan en el soporte de sus distribuciones a-priori respectivas,

$$\pi(a, b, c | \bar{y}, \bar{x}) = f(\bar{y}, \bar{x} | a, b, c) f(a) f(b) f(c) \quad (4)$$

$$= \left(\prod_{i=1}^m \frac{\lambda_i^{y_i} e^{-\lambda_i}}{y_i!} \right) f(a) f(b) f(c) \quad (5)$$

$$\propto \left(\prod_{i=1}^m \frac{\lambda_i^{y_i} e^{-\lambda_i}}{y_i!} \right) \left(e^{-\frac{(a-35)^2}{25^2}} \right) \left(b^3 e^{-\frac{2b}{5}} \right) \left(c^2 e^{-\frac{3c}{950}} \right) \quad (6)$$

$$\propto \left(e^{-(\sum_i \lambda_i) - (\sum_i y_i (x_i - a)^2 / 2b^2)} c^{(\sum_i \lambda_i)} \right) \left(e^{-\frac{(a-35)^2}{25^2}} \right) \left(b^3 e^{-\frac{2b}{5}} \right) \left(c^2 e^{-\frac{3c}{950}} \right) \quad (7)$$

De donde despues podr  amos hacer una factorizaci  n para obtener una propuesta Gibbs si encontraramos una forma de alguna densidad conocida.

A continuaci  n presentaremos las propuestas que utilizamos para cada par  metro a , b y c que utilizamos en nuestro algoritmo :

(Propuesta 1) - Caminata Aleatoria - prob .3

$$a' | a, b, c \sim \mathcal{N}(a, .01)$$

(Propuesta 2) - Caminata Aleatoria - prob .3

$$b' | a, b, c \sim \mathcal{N}(b, .01)$$

(Propuesta 3) - Caminata Aleatoria - prob .4

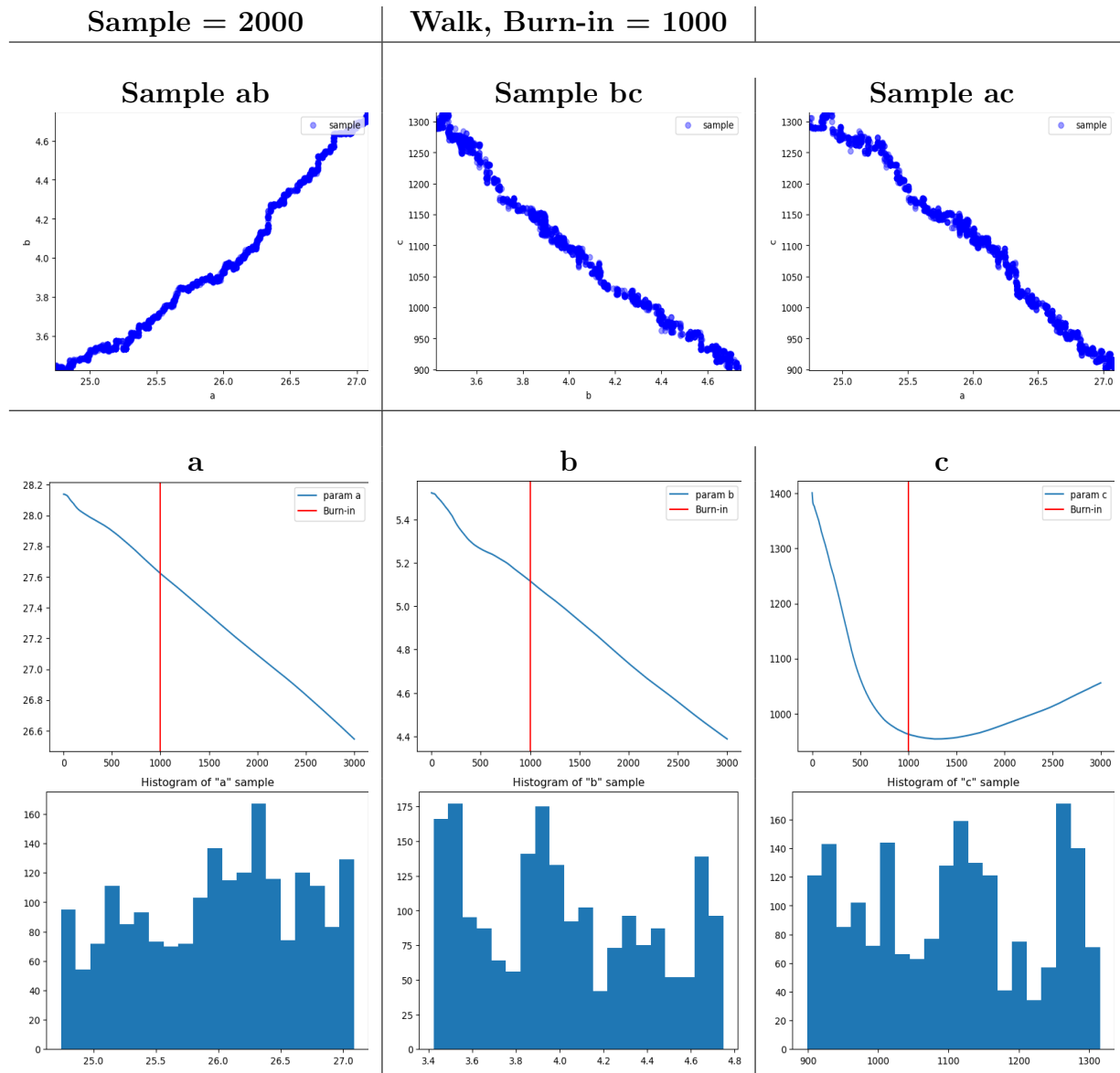
$$c'|a, b, c \sim \mathcal{N}(c, 10)$$

Escogimos una varianza más alta para aproximar la forma de la distribución de este parámetro que al ser en **pesos** tiene una varianza más grande que la **edad**. Los parámetros de σ en cada una de las propuestas fueron escogidos despues de varios experimentos donde estos mostraron mejores resultados.

Como distribución inicial de nuestro algoritmo MH usamos la siguiente propuesta, que fue determinada despues de varios experimentos para poder observar sin demasiadas iteraciones iniciales la convergencia.

$$a \sim \mathcal{U}(20, 30), \quad b \sim \mathcal{U}(0, 10), \quad c \sim \mathcal{U}(1300, 1500)$$

Ahora estudiaremos los experimentos para un tamaño de muestra 2000 con diferentes valores de **burn-in = 1000, 10,000**.



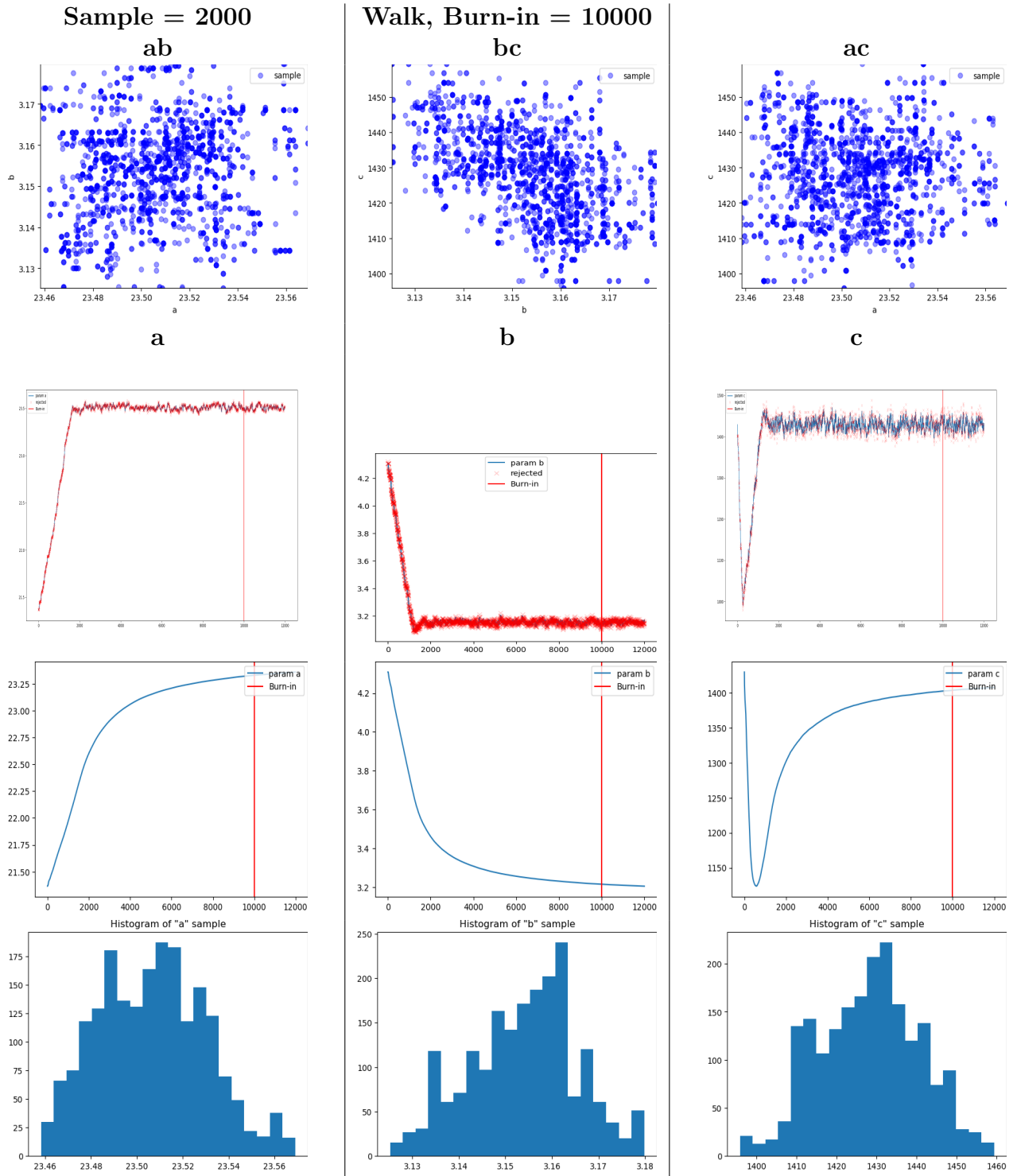
```

Prop 0 : Used 614 : Rejected 300: Rej. Percent 48.86 %
Prop 1 : Used 628 : Rejected 170: Rej. Percent 27.07 %
Prop 2 : Used 758 : Rejected 261: Rej. Percent 34.43 %
All : Used 2000 : Rejected 731: Rej. Percent 36.55 %

```

Notamos que con 1000 iteraciones de burn-in no converge a ning  n valor la media de nuestros par  metros. Esto claramente lo podemos observar en las muestras puesto que presentan m  s un comportamiento de seguir explorando el soporte de la distribuci  n posterior. Tambi  n podemos notar una eficiencia global del 64 % de nuestra cadena, siendo la primera propuesta

con un porcentaje de 48 % de rechazo la menos eficaz en esta etapa de la cadena cuando seguimos explorando.



```
Prop 0 : Used 582 : Rejected 86: Rej. Percent 14.78 %  
Prop 1 : Used 581 : Rejected 149: Rej. Percent 25.65 %  
Prop 2 : Used 837 : Rejected 229: Rej. Percent 27.36 %  
All : Used 2000 : Rejected 464: Rej. Percent 23.20 %
```

En el caso de 10,000 iteraciones de burn-in podemos observar que converge claramente a una región de alta densidad de la distribución posterior. Esto mismo se puede corroborar con los porcentajes de rechazo y de éxito para generar la muestra de tamaño 2000, donde observamos una eficacia del 77%. De modo, que nuestra propuesta es

De los resultados del último experimento sabemos que las medias de los parámetros convergen a valores alrededor de,

$$\hat{a} = 23.487388712622728, \quad \hat{b} = 3.1624298746162607, \quad \hat{c} = 1426.1552131989615$$

Tenemos confianza en estos valores, pues la distribución resultante que observamos en las gráficas *ab*, *bc* y *ac* muestra muy poca varianza. Por la forma en la que aplicamos nuestra regresión Poisson, hay una interpretabilidad clara de estos valores. El primero nos está diciendo que en nuestros datos la edad promedio es de **23 años** en un segmento de edad de ancho $2\hat{b} \approx 6$ de compradores y que el gasto promedio de este segmento de edades es $\hat{c} = 1426$ pesos.

Problema 3

Investiga y describe muy brevemente los softwares OpenBugs, Nimble, JAGS, DRAM, Rtwalk, Mcee Hammer, PyMCMC.

- **OpenBugs** : OpenBugs es la versión opensource del paquete de software BUGS (**B**ayesian inference **U**sing **G**ibbs **S**ampling) que como su nombre nos dice, sirve para efectuar inferencia bayesiana con Gibbs Sampling de modelos jerarquicos de relaciones entre los parámetros. Las relaciones entres los parámetros se definen basandose en la estuctura de un DAG (Directed Acyclic Graph), con un lenguaje especializado en el software para definirlas.
- **Nimble** : NIMBLE es una extensión del lenguaje de BUGS para definir estructuras jerarquicas de los parámetros de un modelo, pero sin restringirlo a su utilización con algoritmos de MCMC, sino explotar esta estructura con otros posibles algoritmos. Está implementado en R, por lo que está acompañado de todas las herramientas ya conocidas de este lenguaje.

- **JAGS** : Las siglas de JAGS significan Just Another Gibbs Sampler. Es un programa de análisis bayesiano de modelos jerárquicos usando simulación MCMC. No es muy distinto que BUGS. Los 3 objetivos que se tuvieron al crear JAGS fueron : 1) Que sea una implementación multiplataforma de BUGS, siendo este distribuido bajo una licencia GNU, 2) Que sea flexible y fácil de extender con otras distribuciones o samplers, 3) Que sea una herramienta para la exploración de ideas en modelación Bayesiana.
- **DRAM** : Las siglas de DRAM significan Delayed Rejection Adaptive Metropolis siendo esta una implementación en Matlab de un algoritmo de MCMC que combina las dos ideas de Adaptive Metropolis, Haario, et al. 2001, y Delayed Rejection. Adaptive Metropolis consiste en ajustar la covarianza en cada iteración cuando se usa una kernel normal, destruyendo la propiedad Markoviana de la cadena, pero preservando propiedades de ergodicidad. Delayed Rejection consiste en generar una segunda propuesta cuando se rechaza la primera propuesta, en vez de solo descartar la primera propuesta y mantenerse en la posición actual.
- **Rtwalk** : Es una implementación en R del algoritmo 't-walk' [2] que es un algoritmo de MCMC usado para muestrear de distribuciones continuas sin tener que escoger ningún hiperparámetro y asegurando la convergencia con algunas pequeñas restricciones simples. El algoritmo de t-walk mantiene en cada iteración dos puntos independientes en el espacio y se tiene una razón de aceptación igual a la de MH en el espacio producto. Se restringen las propuestas o 'pasos' a casos donde estas sean invariantes a escala y transformaciones afines.
- **Mcee Hammer** : **Emcee** es una implementación estable del sampler en conjunto (ensemble) para MCMC presentado por Goodman Weare (2010). Esta implementación es open source y está hecha en Python.
- **PyMCMC** : PyMC es un módulo de python que implementa varios modelos estadísticos Bayesianos y los algoritmos para ajustarlo. En particular se implementan algoritmos MCMC. Tiene implementadas muchas herramientas para medir qué tan bueno es el ajuste de los modelos y explota todas las capacidades de python usando lo más posible la librería **numpy**.

Referencias

- [1] (Springer Texts in Statistics) - Monte Carlo Statistical Methods (2004), Christian Robert, George Casella
- [2] A General Purpose Sampling Algorithm for Continuous Distributions (the t-walk), J. Andres Christen, Colin Fox