Modelos Lineales Doblemente Generalizados con enfoque Bayesiano: Teoría y aplicaciones en R, OpenBugs y Stan

Hernan David Avila Perico

Universidad Nacional de Colombia
Facultad de Ciencias
Bogotá, Colombia
2020

Modelos Lineales Doblemente Generalizados con enfoque Bayesiano: Teoría y aplicaciones en R, OpenBugs y Stan

Hernan David Avila Perico

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de:
Magister en Ciencias - Estadística

Director(a):
Ph.D. Edilberto Cepeda Cuervo

Línea de Investigación:
Inferencia Bayesiana

Grupo de Investigación:
Inferencia Bayesiana

Universidad Nacional de Colombia
Facultad de Ciencias Bogotá, Colombia
2020

¿Por qué me interrogas sobre el abolengo? Cual la generación de hojas, así es la de los hombres. Esparse el viento las hojas por el suelo y la selva, reverdeciendo, produce otras al llegar la primavera, de igual suerte, una generación de hombres nace y otra perece.

Homero, La Iliada, Canto VI - 147

Agradecimientos

Toda mi gratitud al profesor Edilberto Cepeda Cuervo, quien con mucha diligencia y paciencia ha guiado el desarrollo de este trabajo. Adicionalmente, una mención especial al profesor Alvaro Mauricio Montenegro quien sugirió la incorporación de Stan para las aplicaciones y simulaciones, en una primera instancia el alcance estaba limitado a OpenBUGS.

Resumen

En este trabajo se presentan los Modelos Lineales Doblemente Generalizados como una técnica que permite en modelamiento de datos cuando existe dispersión variable. Este documento presenta consideraciones teóricas y aspectos relacionados a la estimación bayesiana en los lenguajes Stan y OpenBUGS a través de simulaciones y aplicaciones desarrolladas en R usando el paquete *R2OpenBUGS* y la distribución *rstan*. Se presentan ejemplos de DGLM como la regresión normal heteroscedástica, la regresión gamma, la regresión beta con simulaciones y aplicaciones. Además, en el capitulo 3 se plantean modelos para datos de conteo basados en la distribución beta binomial y binomial negativa y finalmente, se presenta una extensión para modelamiento de datos longitudinales basados en la distribución normal multivariada. Se espera que este documento sirva de guia para la comprensión y utilización de estos modelos en un contexto bayesiano.

Palabras clave: (Modelos Lineales Doblemente Generalizados, regresión normal heteroscedástica, regresión gamma, regresión beta, regresión beta binomial, regresión binomial negativa, datos longitudinales, sobredisperión, dispersión variable, stan, OpenBUGS).

Abstract

This work presents Double Generalized Linear Models as a technique which allows data modeling when there is variable dispersión. This document contains theoretical aspects and details about bayesian estimation using Stan and OpenBUGS programming languages through simulations and aplications developed in R using package *R2OpenBUGS* and *rstan* distribution. This document contains examples for heteroscedastic normal regression, gamma regression, beta regression with simulations and applications. Furthermore, models for count data based on beta binomial and negative binomial distribution are presented in chapter 3. Finally an extension for longitudinal data based on multivariate normal distribution is exposed. The expectancy is this document to be a guide for understanding and use of this model in a bayesian context.

Keywords: Double Generalized Lineas Models, DGLM, heteroscedastic normal regression, gamma regression, beta regression, beta binomial regression, negative binomial regression, logitudinal data, overdispersion, varaible dispersion, stan, OpenBUGS)

Contenido

Lista de Figuras

Lista de Tablas

1. Introducción

En el estudio y modelamiento de datos en diversas áreas del saber los Modelos Lineales Generalizados, en adelante GLM, son unas de las técnicas más utilizadas, difundidas y estudiadas. Los GLM, fueron introducidos por [Nelder & Wedderburn, 1972] y de manera general, pueden ser entendidos como una extensión de las técnicas de regresión lineal a la familia exponencial uniparamétrica, de ahí el nombre *generalizados* [Gelfand & Dalal, 1990] [Dey et al., 1997].

En su planteamiento, los GLM cuentan con tres componentes: primero, una componente aleatoria que es la variable a modelar, y que por ser una variable aleatoria se le puede asignar una distribución, la cual se encuentra en la familia exponencial uniparamétrica. Segundo, una componente sistemática, que describe los cambios en la variable aleatoria como una función de las covariables, es decir, atribuye parte de la variación de la variable objeto de estudio a las variaciones en las covariables mediante el uso de un predictor lineal. Finalmente, la tercera componente de los GLM es la función de enlace, la cual, conecta la componente aleatoria con la componente sistemática del modelo [McCullagh, 1983].

Si bien los GLM son muy populares y su uso está ampliamente difundido, existe una extensión de los GLM muy útil en contextos de dispersión variable ya que le asigna un predictor lineal al parámetro de dispersión, a esta técnica se le ha dado el nombre de Modelos Lineales Doblemente Generalizados. Los DGLM pueden ser entendidos como una generalización de los GLM a la familia exponencial biparamétrica en donde se le asigna un predictor lineal a cada uno de los parámetros, aunque, de manera incluso más general, puedan ser entendido como una generalización a distribuciones biparamétricas tal como se muestra en [Cepeda, 2001b] y en [Cepeda & Cifuentes, 2017] en donde, entre otros modelos, se propone la regresión beta, con modelado conjunto de los parámetros de media y dispersión y las regresiones beta-binomial y binomial negativa también con modelado conjunto para los parámetros de media y dispersión respectivamente.

Teniendo en cuenta lo anterior, la diferencia entre los GLM y los DGLM es que los segundos buscan modelar no sólo la relación de un parámetro, usualmente la media, con un conjunto de covariables, sino que buscan modelar la relación de dos parámetros, usualmente la media y el parámetro de dispersión con un conjunto de covariables [Smyth & Verbyla, 1999]. Un ejemplo de contraste entre los GLM y los DGLM es la regresión lineal normal, en la

cual, se puede adoptar un enfoque de varianza constante u homoscedástico o un enfoque de dispersión variable o heteroscedástico. La principal diferencia entre estos dos enfoques es que el primero, en el marco de los GLM, asume que la varianza es conocida y el proceso de modelamiento consiste en estimar un predictor lineal para la estructura de la media. En contraste, en el marco de los DGLM, la regresión normal heteroscedástica asume un predictor lineal para la media y otro para la varianza [1] [Aitkin, 1987, Cepeda & Gamerman, 2000].

Un objetivo de este documento es presentar algunos de los fundamentos teóricos de los DGLM y su uso, para lo cual, este texto incluye una exposición de aspectos teóricos generales e implementa varias simulaciones y ejemplos de su aplicación. Este trabajo adopta un enfoque bayesiano y muestra la manera de hacer uso de los DGLM a través de OpenBugs y Stan. OpenBUGS se adopta debido a que es un referente importante en cuanto a software y lenguaje de programación de estimación bayesiana. Stan, por su parte, desde su lanzamiento en 2012 ha ganado mucha importancia en el modelamiento Bayesiano.

OpenBugs fue desarrollado por el departamento de Bioestadística de la Universidad de Cambridge y su mayor bondad se basa en dos aspectos fundamentales. Primero, por ser un lenguaje de programación probabilístico, es muy sencillo llevar a cabo la programación de un modelo ya que para esto sólo es necesario establecer las distribuciones a priori de los parámetros y la función de verosimilitud. Segundo, la estimación en OpenBUGs es usualmente rápida y muy eficiente, lo cual permite llevar a cabo estimaciones en tiempos razonables [Spiegelhalter et al., 2007].

Stan, por su parte, es un lenguaje de programación probabilístico para estadística bayesiana escrito sobre C++, que permite realizar computación estadística y que en los últimos años ha tenido gran difusión. Entre sus aspectos más notables se tiene que usa mecánica Hamiltoniana en los algoritmos HMCMC y non-Uturn Sampler, además de inferencia variacional y algoritmos de optimización avanzados como BFGS de memoria limitada.

Un aspecto importante a considerar, es que tanto OpenBUGS como Stan se integran con R, el cual, es un software y lenguaje de programación altamente conocido y difundido. OpenBUGS se integra con R a través del paquete R2OpenBugs [Sturtz et al., 2005] el cual transmite datos entre *R* y *OpenBUGS*, razón por la cual es necesario disponer de los dos programas de manera independiente. El enfoque de la integración de Stan con R es un poco diferente al de OpenBUGS ya que Stan se integra con R a través de *rstan* [Carpenter et al., 2017], no obstante, a diferencia de *R2OpenBUGS*, *rstan* es más bien una distribución de *Stan* diseñada para *R*. Esto último quiere decir que no es necesario tener disponibles *Stan* y *R*

[1] Usualmente, en un contexto de modelamiento de datos no se conoce con certeza el valor de los parámetros a estimar, por lo que cuando se lleva a cabo el modelamiento solo del parámetro de media usualmente se estima el parámetro de varianza constante para todas las observaciones.

sino que es suficiente con instalar *rstan* sobre *R* para utilizar este lenguaje. Teniendo en cuenta lo anterior, este documento trabaja principalmente desde *R* aprovechando sus fortalezas en cuanto a su amplia difusión, manejo y generación de datos e integración con Stan y OpenBUGS.

El presente trabajo está ordenado en 4 capítulos. En el primero, a manera de introducción se presenta el objeto de estudio y se profundiza brevemente en aspectos teóricos relacionados con GLM, además de presentar de manera general aspectos referentes a Stan y OpenBUGS, su funcionamiento, sus principales diferencias y la programación de modelos en estos dos lenguajes. Posteriomente, en el segundo capítulo se presentan los DGLM y se trabajan algunos ejemplos basados en la distribución normal, la distribución gamma y la distribución beta. Además, se presentan los códigos para la programación de estos modelos y simulaciones y aplicaciones. En el capítulo 3 se presentan modelos para datos de conteo con el modelo Beta Binomial y Binomial Negativo. Para cada uno de estos modelos se presentan aspectos teóricos del modelo y cuestiones relativas a estimación de los mismos y adicionalmentese presentan simulaciones y aplicaciones.

Una vez desarrollados los modelos anteriores, se incluye a manera de extensión, una sección que muestra el modelamiento de datos longitudinales, los cuales pueden ser definidos como observaciones repetidas de uno o varios individuos a lo largo del tiempo [Weiss, 2005].Lo anterior, dado que en el modelamiento de datos longitudinales un componente de suma importancia es modelar la varianza y covarianza entre las observaciones de cada uno de los individuos. Para su modelamiento, en este capítulo se consideran algunas estructuras de covarianza y se desarrollan ejemplos de simulación y algunas aplicaciones.

Finalmente, se incluyen los apéndices A y B en donde se profundiza brevemente en aspectos teóricos de estimación bayesiana y en evaluación de modelos respectivamente.

1.1. Introducción a los Modelos Lineales Generalizados - GLM

Los GLM son una técnica estadística que puede entenderse como una extensión de regresión normal a variables cuya distribución se encuentra en la familia exponencial uniparamétrica, lo cual, lo hace una de las técnicas más utilizadas debido a que en esta familia se encuentran distribuciones tan conocidas como la distribución binomial, poisson, gamma o inversa gaussiana [Nelder & Wedderburn, 1972, McCullagh, 1983].

Definición 1.1.1 ***Modelos Lineales Generalizados - GLM:*** *Sean Y_i $i = 1, ..., n$ variables aleatorias independientes, Los modelos lineales generalizados, GLM, están definidos*

por las siguientes tres compomnentes:

- *La componente aleatoria: Los componentes Y_i tienen distribución en la familia exponencial, con $E(Y_i) = \mu$ y $V(y_i) = \sigma_i^2$.*
- *La componentes sistemática: El predictor lineal $\eta_1 = X_i\beta$.*
- *La función de enlace: La fución de enlace conecta la parte sistemática con la componente aletoria de manera que $h_1 = g(\mu_i)$.*

La primera componente, es la componente aleatoria del modelo, es la variable cuyo comportamiento se quiere modelar y que en este caso se encuentra en la familia exponencial. Se considera un vector $Y = (y_1, y_2, ..., y_n)^T$ donde se asume independencia entre los componentes del vector Y, esta independencia es muy común como supuesto en el caso de datos de corte transversal y longitudinal, aunque en los datos longitudinales, como se expondrá más adelante, es muy importante modelar la dependencia entre las observaciones de un mismo individuo[2]. La función de probabilidad de cada uno de los componentes de este vector se debe poder expresar como lo muestra la ecuación 1-1 para que dicha distribución se encuentre en la familia exponencial.

$$f(y_i|\theta, \phi) = exp[(y_i\theta - b(\theta))/a(\phi) + c(\theta, \phi)]I_Y(y_i) \tag{1-1}$$

La ecuación 1-1 tiene dos parámetros, θ y ϕ donde se tiene que si se desconocen los dos parámetros se tiene la familia exponencial biparamétrica y si se conoce ϕ pero se desconoce θ se tiene la familia exponencial con parámetro canónico. En el caso de los GLM se tiene que ϕ se asume conocido [McCullagh, 1983].

La segunda componente de un glm, es la componente sistemática, la cual, hace referencia al predictor lineal generado por las covariables X. Este predictor lineal contiene la información que el conjunto de covariables X aporta en la variable dependiente Y, o dicho de otra manera, establece una relación en la forma como las covariables X influyen en el comportamiento de las variable de estudio, esta componente se denota como $\eta = X\beta$ donde $\eta = (\eta_1, ..., \eta_n)^T$, $\beta = (\beta_1, ..., \beta_p)^T$ y X es una matriz de $n*p$ llamada matriz de covariables o matriz de diseño.

La tercera componente de un glm es la función de enlace, la cual tiene por objeto unir la componente sistemática y la componente aleatoria y es parte fundamental del modelamiento en la medida en que define la relación entre los parámetros de la ecuación 1-1 y el predictor lineal presentado anteriormente. De esta manera, se tiene que para la i-esima observación

[2]la independencia entre los componentes del vector también permite, en el caso de la estimación clásica, asumir la verosimilitud total como la productoria de las verosimilitudes de cada observación

$\eta_i = h(\mu_i)$ donde es común asumir que $h(.)$ es una función monótona y dos veces diferenciable, esto ultimos es de suma importancia ya que permitir el uso de métodos de maximización, a través de algoritmos como Newton-Raphson o Scoring de Fisher, en la estimación clásica de estos modelos.

Cuando, en la anterior ecuación se asume que $h(\mu_i) = \theta_i$ se tiene el enlace canónico, el cual, por ejemplo corresponde a la función logit para distribución binomial, a la función identidad para la distribución normal, la función logarítmica para la distribución poisson o el recíproco para la familia gamma [McCullagh, 1983].

Ejemplo 1.1.1 *Si se considera que $y_i \sim N(\mu_i, \sigma)$ con σ conocida se tiene que el GLM considerando el enlace canónico viene dado por la siguiente expresión:*

$$h(\mu_i) = \mu_i = X_i\beta \quad \forall i = 1, 2, ..., n$$

Donde X_i es el conjunto de covariables pertenecientes a la i-ésima observación y β es un vector de parámetros a estimar. En este ejemplo se aprecian los tres componentes del modelo, y_i es la componente aleatoria con distribución normal, $X_i\beta$ es la componente sistemática del modelo y $h(\mu_i)$ es la función de enlace, la cual, en este caso es la función identidad.

Ejemplo 1.1.2 *Si $y_i \sim Bin(n, p_i)$ se tiene que el enlace canónico en este caso es el enlace logit, por lo tanto, el modelo sería:*

$$h(p_i) = log(\frac{p_i}{1 - p_i}) = X_i\beta \quad \forall i = 1, 2, ..., n$$

Donde X_i es el conjunto de covariables pertenecientes a la i-ésima observación y β es un vector de parámetros a estimar. En este ejemplo se aprecian los tres componentes del modelo, y_i es la componente aleatoria con distribución binomial, $X_i\beta$ es la componente sistemática del modelo y $h(\mu_i)$ es la función de enlace, la cual, en este caso es la función logit.

Ejemplo 1.1.3 *Si $y_i \sim P(\mu_i)$ el enlace canónico para la distribución de poisson es el enlace logarítmico y el modelo estaría dado por la expresión*

$$h(\mu_i) = log(\mu_i) = X_i\beta \quad \forall i = 1, 2, ..., n$$

Donde X_i es el conjunto de covariables pertenecientes a la i-ésima observación y β es un vector de parámetros a estimar. En este ejemplo se aprecian los tres componentes del modelo, y_i es la componente aleatoria con distribución poisson, $X_i\beta$ es la componente sistemática del modelo y $h(\mu_i)$ es la función de enlace, la cual, en este caso es la función logaritmo.

1.2. Stan y OpenBUGS

En esta sección se presentan los dos lenguajes utilizados en este trabajo, Stan y OpenBUGS. En la primera sección se presentan aspectos relacionados con Stan su funcionamiento y la forma de programar modelos en este leguaje. En la segunda sección se muestran estos mismos temas para OpenBUGS.

1.2.1. Stan

Stan: Descripción y funcionamiento general

Stan es un lenguaje de programación probabilístico enfocado al modelamiento bayesiano lo que significa que está específicamente enfocado a generar muestras de distribuciones a posteriori de los parámetros de un modelo. Stan, tiene este nombre en honor a Stanislaw Ulam, quien fue un pionero en el desarrollo de métodos de montecarlo para el muestreo de distribuciones [Carpenter et al., 2017].

El muestreo de distribuciones a posteriori en Stan involucra el el uso de diversas herramientas como lo son algoritmos de Markov Chain Monte Carlo con mecánica Hamiltoniana, la cual, es una de las principales ventajas de Stan. El uso de inferencia variacional para aproximación de distribuciones a posteriori y algoritmos de gradiente para maximización de verosimilitud basados en diferenciación automática.

Básicamente, y de manera muy simplificada e intuitiva, cuando un modelo es programado en Stan y es compilado corretamente, la ejecución general del modelo tiene multiples procesos. En primera medida, se puede considerar la aproximación de la función a posteriori mediante el método de inferencia variacional (IV), el cual, es un método aplicable en estimación bayesiana que se presenta como alternativa a los métodos mas utilizados de MCMC y al algoritmo de Esperanza Maximización (EM). La ventaja que ofrecce este método es que se desempeña muy bien cuando se tienen conjuntos de datos muy grandes con una dimensionalidad alta y un número de parámetros grande. El enfoque del método de IV consiste en lograr la aproximación de una distribución a posteriori a través de una segunda distribución llamada distribución de aproximación, al tiempo que se minimiza alguna medida de discrepancia entre las dos distribuciones.

Una vez apoximada la distribución a posteriori, Stan lleva a cabo el muestreo de la misma incorporando Mecánica Hamiltoniana mediante algoritmos conocidos como HMC (Hamiltonian Monte Carlo) y NUTS (Non U turn Sampler). El uso de la mecánica Hamiltoniana para el muestreo de una distribución permite un muestreo más eficiente de la distribución a posteriori mediante una exploración más completa del espacio muestral. La analogía consiste en ver las generaciones de nuevas muestras en un algoritmo MCMC como el movimiento de

la variable que se está muestreando y aplicar los principios de mecánica hamiltoniana a dicho proceso.

Algunas referencias para profundizar en tópicos específicos del funcionamiento y modelamietno de Stan son [Carpenter et al., 2017] y [Stan Development Team, 2016], en el primero es una manual general que describe el funcionamiento de Stan y desarrolla tópicos diversos en cuanto al uso de este lenguaje, el segundo es el manual de uso oficial de Stan.

Para los lectores que deseen profundizar en aspectos teóricos de los algoritmos implementados en Stan se recomiendan [Chib & Greenberg, 1995], [Neal et al., 2011] y [Hoffman & Gelman, 2014] para profundizar en los algoritmos de Metropolis Hastings, uso de mecanica hamiltoniana en muestreo de distribuciones a posteriori y el algoritmo NUTS. Finalmente, el trabajo de [Blei et al., 2017] es una buena guia de estudio para los métodos de inferencia variacional.

Estructura General de un programa en Stan

Tras haber expuesto como funciona Stan y haber ilustrado como funcionan los algoritmos que utiliza, es necesario explicar como se lleva a cabo la programación en este lenguaje, es decir, la estructura básica de un programa en Stan, la cual, en general puede constar de 7 bloques de códigos, algunos de los cuales son opcionales.

```
functions{

}
data{

}
transformed data{

}
parameters{

}
transformed parameters{

}
model{

}
generated quantities{
```

```
}
```

En el primer bloque es el bloque de declaración de funciones en el cual, como su nombre lo indica, se deben declarar todas las funciones que el desarrollador requiera para el programa incluyendo la definición de nuevas distribuciones de probabilidad. El segundo bloque de código es el bloque de declaración de datos, en el cual, se establecen los datos requeridos para el modelo. En el tercer bloque de código, se llevan a cabo las transformaciones necesarias sobre los datos declarados en el segundo bloque.

El bloque de parámetros declara los parámetros a estimar en el modelo y en el quinto bloque, el bloque de parámetros transformados, se llevan a cabo todo tipo de transformaciones necesarias sobre los parámetros, lo cual, incluye también combinaciones lineales muy usuales en contextos de regresión.

Los últimos bloques de códigos son el bloque donde se declara el modelo y el bloque donde se generan cantidades. En el bloque de declaración del modelo está el centro de la estimación del programa, en este bloque, se declara la función de verosimilitud y las distribuciones a priori de los parámetros. En el último bloque de código, se generan cantidades de interés como por ejemplo muestras de la variable dependiente.

Existen algunos aspectos importantes a tener en cuenta al momento de escribir el código en Stan. Primero, todas las secciones del código son opcionales, es decir, el codigo puede ser creado por ejemplo sin necesidad de declarar funciones de usuario o generar cantidades. Segundo, aunque no es lo más recomendado no es obligatorio seguir estrictamente cada uno de los bloques de código, por ejemplo, los parámetros pueden transformarse en el bloque de modelos, esto, aunque posible no es la práctica más recomendada.

1.2.2. OpenBUGS

OpenBUGS: Descripción y Funcionamiento General

OpenBUGS es la versión gratuita de WinBUGS, el cual es un software estadístico desarrollado por el departamento de bioestadística de la Universidad de Cambridge y basado en el proyecto BUGS que es el acrónimo para Bayesian inference Using Gibbs Sampling. OpenBUGS lleva a cabo el modelamiento estadístico usando metodos de MCMC (Markov Chain Monte Carlo) y permite trabajar con modelos de casi cualquier complejidad de manera fácil ya que utiliza modelos de grafos dirigidos acíclicos (DGA) permitiendo la división de estructuras complejas en secuencias de estructuras mas simples [Lunn et al., 2012].

Algunas referencias para profundizar en tópicos específicos del funcionamiento y modelamiento de OpenBUGS son [Lunn et al., 2012] y [Spiegelhalter et al., 2007]. El primero de

ellos es un manual general que desarrolla tópicos diversos y el segundo es el manual de uso oficial de OpenBUGS.

Estructura General de un programa en OpenBUGS

Una de las ventajas de trabajar con OpenBUGS es su facilidad de parametrización, ya que al ser un lenguaje de programación probabilítico su programación se basa solamente en establecer las relaciones necesarias entre las variables del modelo. En el caso de estimación bayesiana, la programación se reduce a establecer la función de verosimilitud y las distribuciones a priori de los parámetros a estimar.

Comparativamente con Stan, implementar un modelo en OpenBUGS es más sencillo, sin embargo, también es menos flexible. Teniendo en cuenta esto, la parametrización de un modelo en OpenBugs se debe tener en cuenta lo siguiente:

- Es necesario crear un modelo que tenga la función de verosimilitud y las distribuciones a priori para cada uno de los parámetros a estimar, este modelo es el aspecto clave para poder realizar una estimación en OpenBugs ya que básicamente contiene toda la información de la estrucutra de la estimación, las cuales, para el caso de una estimación bayesiana son la función de verosimilitud y las distribuciones a priori.
- Los datos de todas las variables, deben ser suministrados en formato lista. Para el caso de las matrices, es necesario tener en cuenta que OpenBugs hace el llenado de las mismas por filas.
- Existen aspectos complementarios de la estimación que son de suma importancia como el número de iteraciones, el número de cadenas y el número de iteraciones que se eliminaran por ser consideradas parte del calentamiento del modelo. Estos aspectos cuando se manejan desde openBugs se configuran en la interfaz gráfica del software, sin embargo, cuando se maneja desde R, son parámetros adicionales de una función de R.

El siguiente fragmento de código muestra la estructura básica de un modelo en OpenBUGS en el cual se establece la función de verósimilitud, las distribuciones a priori de los parámetros y otros cálculos importantes para el modelo.

```
model{
    L # Función de Verosimilitud
    P # Distribuciones a priori
    0 #Otro cálculos relevantes para el modelo como cantidades generadas
}
```

Esta declaración del modelo consta de una función de verosimilitud, las distribuciones a priori y los otros cálculos relevantes para el modelo.

1.2.3. Stan y OpenBUGS, algunas diferencias a considerar

Stan y OpenBUGS son catalogados como dos lenguajes de programación probabilístico, esto quiere decir que la programación del modelo se lleva a cabo mediante la especificación de sus componentes principales y todo el proceso de estimación se realiza de manera automática mediante las técnicas de estimación previamente consideradas por el lenguaje. Este tipo de paradigma de programación permite entre otras cosas una gran facilidad en la programación de modelos [Carpenter et al., 2017].

Tanto Stan como OpenBUGS son leguajes de programación probabilístico enfocados a estimación Bayesiana. Sin embargo, tal como se muestra en este apartado existen diferencias sustanciales entre los dos lenguajes. Una primera diferencia a considerar radica en el hecho que OpenBUGS es un lenguaje interpretado mientra que Stan es compilado.

Una segunda diferencia a resaltar surge del hecho que existen diferencias importantes en el orden en el que es permitido escribir el código, por ejemplo, dado que Stan se traduce a C++ es indispensable hacer las declaraciones antes de usar cualquier variable, por su parte, OpenBUGS se ejecuta acorde al modelo de grafos acíclicos de manera que cada una de las variables está disponible al momento de su utilización.

Una tercera diferencia importante radica en que OpenBUGS utiliza algoritmos de MCMC para llevar a cabo la actulización de los parámetros uno a la vez, por su parte, Stan utiliza algoritmos de HMC y lleva a cabo la actualización de los parámetros todos a la vez.

1.2.4. Ejecución desde R: rstan y R2OpenBUGS

En la sección anterior, se presentaron las generalidades del funcionamiento y de la parametrización de un modelo en Openbugs y en Stan. Ahora es necesario presentar la manera como se alimentan los datos en los dos lenguajes y la forma como se ejecutan códigos de estos lenguajes desde R.

La integración entre R y OpenBUGS se logra a través de la libreria R2OpenBUGS, la cual, es una envoltura (Wrapper en Inglés) que permite el manejo de OpenBUGS desde R, para lo cual, se debe tener instalado previamente OpenBUGS [Sturtz et al., 2010]. Esta libreria permite la inicialización de OpenBUGS, la parametrización del modelo a través de un archivo con extensión *.txt*. Además, la carga de datos a través de una lista y la definición de valores

iniciales a traves de listas de objetos de R, finalmente, se configuran aspectos relacionados al proceso de estimación bayesiana como el número de cadenas, el numero de iteraciones entre otros.

Stan, por su parte, tiene diferentes integraciones o distribuciones que permiten su uso desde diferentes lenguajes de programación o software como R, Python, Julia, Matlab o Stata, en este caso, se usará *rstan*. Lo anterior quiere decir que al contrario de OpenBUGS que para su uso se instala OpenBUGS y *R2OpenBUGS* para enviar comandos desde *R* a *OpenBUGS*, en el caso de *rstan* este funciona de manera nativa sin que sea necesario instalar algún otro complemento.

EL siguiente fragmento de código muestra el uso de la función bugs del paquete *R2OpenBUGS* [Sturtz et al., 2010], con la cual, se lleva a cabo la estimación del modelo. En este ejemplo, la función bugs toma los datos de las variables X y Y y los valores iniciales de los objetos *beta*0 y *sigma*0, el modelo, en este caso, lo toma del archivo *modelo.txt* y establece que los parametros a estimar son *beta* y *sigma*; como complemento, establece el número de cadenas en 1 (parámetro n.chains), el número de iteraciones en 10.000 (parámetro n.iter) y la cantidad de iteraciones a descartar en 5.000 (parámetro n.burnin).

```
modelo = R2OpenBUGS::bugs(data = list("x", "y"),
                          inits = list("beta0","sigma0"),
                          model.file="modelo.txt",
                          parameters= c("beta","sigma"),
                          n.chains = 1,
                          n.iter=1000,
                          n.burnin = 5000)
```

El siguiente fragmento de código muestra como llevar a cabo la estimación de un modelo utilizando Stan a través de la función *stan* del paquete rstan [Guo et al., 2016]. Al igual que en la ejecuión de OpenBUGS la función stan toma los datos de las variables X y Y. El modelo, lo toma del archivo *modelo.stan* y establece que los parametros a estimar son *beta* y *sigma*. Como complemento, establece el número de cadenas en 1 (parámetro chains), el número de iteraciones en 10.000 (parámetro iter) y la cantidad de iteraciones a descartar en 5000 (parámetro warmup).

```
modelo = rstan::stan(data=list("x","y"),
                     file = "un_modelo.stan",
                     pars= c("beta","sigma"),
                     chains = 1,
```

```
iter=10000,
warmup = 5000)
```

Como es posible observar en los dos extractos de código existen similitudes importantes en la ejecución de los modelos de stan y OpenBUGS desde R. Primero, los datos se alimentan desde una lista de R. Segundo, tanto Stan como OpenBUGS leen los modelos desde archivos externos, en stan es un archivo con extensión *.stan* y en OpenBUGS es un archivo *.txt*, en estos archivos, está escrito un modelo con cada una de las secciones que se explicaron en las secciones 1.2.1 y 1.2.2. Finalmente, en los dos ejemplos se muestra el numero de iteraciones totales y el número de iteraciones a descartar en el calentamiento de cada una de las cadenas.

Interfaz Gráfica de OpenBUGS

Además de la ejecución desde *R*, es importante señalar que para sistema operativo Windows, *OpenBUGS* cuenta con interfaz gráfica de usuario -*GUI*, la cual, es una forma de usar esta herramienta y es muy útil sobre todo para aquellos usuarios que no tienen una formación en programación en *R*. En este apartado, se presentará una breve descripción de como utilizar la interfaz de OpenBUGS.

Al igual que en el caso de R, en la GUI, también es necesario declarar el modelo y alimentar los datos, en este caso, tanto el modelo como los datos se escriben en el editor que se despliega al navegar a $File > New$, cada uno en un editor por separado.

Una vez se tienen escritos tanto el modelo como los datos, se debe navegar a la herramienta de especificación del modelo en $Model > Specification$. En la ventana que se despliega se debe verificar el modelo, para esto, se activa la ventana en la cual se escribió el mismo y se verifica el modelo en la herramienta de especificación. Para la carga de los datos, se procede de la misma manera, es decir, se activa la ventana en que se escribieron los datos y en la ventana de especificación del modelo se activa la opción de cargar datos. Una vez que estan cargados los datos y verificado el modelo se especifican en la misma ventana la cantidad de cadenas. Finalmente, se compila el modelo en esta misma ventana.

Una vez compilado el modelo, se procede a declarar los parámetros que se quieren monitorear en el proceso de muestreo. Para esto se debe navegar al menu inference $Inference > Samples$, lo cual despliega la herramienta de monitoreo de muestreo. En esta herramienta se deben agregar los elementos para los cuales se quiere llevar a cabo el monitoreo del muestreo, para esto, se escribe el nombre en el recuadro *node* y luego pulsar el boton *set*.

Finalmente, para llevar a cabo el muestreo se debe navegar a $Model > Update$, lo cual des-

pliega la herramienta de actualización de datos. En esta herramienta se selecciona la cantidad de muestras en cada uno de los ciclos y cada actualización genera tantas muestras como las establecidas en el recuadro *updates*.

Después de haber generado las muestras, con el fin de obtener los resultados de este proceso es necesario desplegar nuevamente la herramienta de monitoreo de muestreo, para esto, se navega a $Inference > Samples$. En esta herramienta se generan resumenes de estimación seleccionando los parámetros en el recuadro *node* y las cadenas que se quieren reportar y el intervalo de iteraciones que se quiere reportar en los recuadros *begin* y *end*.

En resumen, para llevar a cabo la estimación básica de un modelo se siguen los siguientes pasos:

1. Declarar el modelo y cargar los datos en $Model > Specification$ a partir del modelo y los datos creados en el editor que se despliega en $File > New$.
2. Establecer los parámetros a estimar en $Inference > Samples$.
3. Realizar el muestreo en $Model > Update$.
4. Obtener los resultados del muestreo en $Inference > Samples$.

Para declarar el modelo y cargar los datos, en este documento se utiliza el editor que se despliega en $File > New$ y una vez escritos en este editor, tanto los datos como el modelo se cargan en la interfaz de especificación del modelo. El siguiente fragmento de código muestra un ejemplo del elemento modelo tal como se puede declarar en OpenBUGS.

```
model
    {
    for (i in 1:N)
    {
        mu[i] <- f(x*beta) ;
        y[i] ~ verosimiltud(mu[i]);
    }

    beta_0 ~ prior_1(0);
    beta_1 ~ prior_2(0);
    beta_2 ~ prior_3(0);
    }
```

Para cargar los datos a OpenBUGS desde el editor, existen varios formatos a tener en cuenta, por ejemplo, el siguiente código muestra la carga de un vector *mi_vector*, un escalar *mi_escalar* y una matriz *mi_matriz*. Es importante recalcar que OpenBUGS carga los datos

de matrices por filas, lo cual quiere decir que en este ejemplo los 5 primeros datos de cada matriz corresponden a la primera fila, los siguientes 4 a la segunda fila y así sucesivamente dado que *mi* tiene 30 filas y 5 columnas como lo muestra $.Dim = c(30, 5)$.

```
list(
    mi_escalar =22,
    mi_vector=c(34.8, 71.5, 90.5, 78, 11.24),
    mi_matriz=structure(
             .Data=c(
                   12, 45, 27, 56, 89,
                   .
                   .
                   .
                   56, 12, 75, 98, 77),
             .Dim=c(30, 5)
          )
    )
```

Ejemplo 1.2.1 ***GLM con la distribución poisson en Stan, OpenBUGS y R****: Teniendo en cuenta la estructura de modelo presentada en el 1.1.3 se mostrará como usar Stan y OpenBUGS para llevar a cabo la estimación de este modelo. Primero, se simularon* $n = 100$ *datos con la estructura para la media* $ln(\mu_i) = 1 - 0{,}05x_{1,i} + 0{,}2x_{2,i}\ \forall i = n, ..., 50$*, donde* x_1 *se generan de una distribución uniforme continua en el intervalo* $(0, 10)$ *y* x_2 *se genera de una distribución uniforme* $(10, 20)$*. Los datos de la variable dependiente* y_i *se simularon de una distribución poisson* $y_i \sim Poisson(\mu_i)$ *tal como muestra el siguiente código de R.*

```
library('rstan')
library('R2OpenBUGS')
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())


n <- 100

beta_0 <- 1
beta_1 <- -0.05
beta_2 <- 0.2


x_1 <- runif(n, 0,10)
x_2 <- runif(n, 10,20)
```

```
mu_i <- exp(beta_0 + beta_1*x_1 + beta_2*x_2)
y <- rpois(n,mu_i)
```

Para la estimación en Stan se utiliza el siguiente código, en el cual, se declaran las variables tanto dependientes como independendientes en el apartado data. En la sección parameters se declaran los parámetros a estimar y en la sección tranformedparameters se lleva a cabo la declaración del parámetro de media y el cálculo del mismo como una función de los parámetros β y las covariables X. Finalmente, en la sección model se asignan las prior de los parámetrosn beta las cuales en este caso son $N(0, 10)$ y la verosimilitud poisson a la variable y.

```
data{
    int<lower=0> N;
    real x_1[N];
    real x_2[N];
    int<lower=0> y[N];
}
parameters{
    real beta_0;
    real beta_1;
    real beta_2;
}
transformed parameters{
    real mu[N];
    for(i in 1:N){
        mu[i] = exp(beta_0 + beta_1*x_1[i] + beta_2*x_2[i]);
    }
}
model{
    beta_0 ~ normal(0,10);
    beta_1 ~ normal(0,10);
    beta_2 ~ normal(0,10);

    for (i in 1:N){
        y[i] ~ poisson_lpmf(mu[i]);
    }
}
```

Para la estimación en OpenBUGS se utiliza el siguiente código el cual es mucho más sencillo que el de Stan. En este código se presenta la asignación de la función de verosimilitud poison a la variable y y las prior normales para los parámetros beta, las cuales en este caso son

$N(0,0{,}1)$. Además se calcula el parámetro de media mu mediante la combinación lineal de las variables mencionadas anteriormente.

```
model
{
for (i in 1:N)
{
    mu[i] <- exp(beta_0 + beta_1*x_1[i] + beta_2*x_2[i]) ;
    y[i] ~ dpois(mu[i]);
}

beta_0 ~ dnorm(0,0.1);
beta_1 ~ dnorm(0,0.1);
beta_2 ~ dnorm(0,0.1);
}
```

El siguiente código de R ejecuta los dos programas anteriores mediante las funciones stan y bugs de los paquetes rstan y R2OpenBUGS. Para el caso de Stan el anterior código se guardó en el archivo poisson.stan y en el caso de OpenBUGS se guardó en el archivo poisson.txt. En este caso ambas funciones fueron alimentadas desde la lista data, la cual contiene los datos de las covariables, la variable dependiente y el número de observaciones.

```
data = list(x_1 = x_1, x_2 = x_2, y=y, N=n)

model_stan <- rstan::stan(file = 'poisson.stan', data = data,
                      iter = 10000, pars = c('beta_0','beta_1', 'beta_2'),
                      chains = 1)



model_ob <- R2OpenBUGS::bugs(data=data,
                             inits = NULL,
                             model.file = 'poisson.txt',
                             parameters= c('beta_0', 'beta_1', 'beta_2'),
                             n.chains = 1,
                             n.iter=500000)
```

2. Modelos Lineales Doblemente Generalizados

Tal como se consideró anteriormente, en presencia de dispersión variable, es conveniente llevar a cabo el modelamiento de la misma a través de variables explicativas. Por ejemplo, en el modelo normal, en el cual, la media de la variable y_i se puede modelar de manera que $\mu_i = X_i\beta$ y la varianza de la variable aleatoria puede ser modelada mediante la estructura $g(\sigma_i^2) = z_i\gamma$, donde $g(.)$ es una función de valor real que debe garantizar que σ_i^2 sea positivo, z_i es un vector de variables explicativas y γ es un vector de parámetros a estimar [Aitkin, 1987, Cepeda, 2001a].

En este capítulo se presentan modelos de regresión en los cuales se lleva a cabo el modelamiento de parámetros en la familia exponencial biparamétrica mediante el uso de estructuras de regresión para los parámetros de media y un parámetro relacionado con la dispersión de la variable. El objetivo de este capítulo es presentar este tipo de modelos e ilustrar la programación de estos en Stan y OpenBUGS.

La literatura aporta diversos referentes para el estudio de los DGLM. [Smyth, 1989] plantea una extensión al trabajo de [McCullagh, 1983] en el cual ya se consideraban algunas maneras de manejar conjuntos de datos con sobre o subdispersión. En [Smyth, 1989] se plantea que en el caso en el que los conjuntos de datos presenten dispersión variable una manera de afrontarla consiste en llevar a cabo el modelamiento de esta, es decir, aplicar una estructura de regresión no solo sobre la media sino también sobre la dispersión del conjunto de datos, en este trabajo, esta propuesta de modelamiento es planteada y aplicada para modelos de la familia exponencial. Posteriormente, [Smyth & Verbyla, 1999] plantea métodos de estimación con máxima verosimilitud restringida para esta familia de modelos.

Una metodología bayesiana fue propuesta por [Dey et al., 1997] para la estimación de parámetros ortogonales en la familia exponencial biparamétrica presentada en la ecuación 2-2 y una parametrización ortogonal de la misma, la cual, además permite simplificar la estimación clásica tal como lo muestra [Cepeda, 2016]. Ampliaciones y complementos a esta metodología fueron presentadas por [Cepeda, 2001a, Cepeda & Gamerman, 2005] en donde se desarrollan propuestas bayesianas para diversos tipos de modelos en la familia exponencial biparamétrica y para datos de medidas repetidas, además, una extensión de DGLM a modelos lineales

generalizados con efectos aleatorios en la familia exponencial biparametrica es propuesta por [Cepeda et al., 2014].

En este capítulo se define la familia exponencial biparamétrica y se muestran algunas distribuciones que pertenecen a esta. A continuación, se definen los DGLM y se muestran algunos ejemplos de los mismos, concretamente, se presenta el modelo de regresión normal, el modelo de regresión gamma y el modelo de regresión beta. Finalmente, se presentan ejemplos de simulación y aplicaciones en los cuales se ilustra la programación en Stan y OpenBUGS y la utilización de estos modelos en contextos aplicados.

2.1. Familia Exponencial Biparamétrica

La familia exponencial la presenta [McCullagh, 1983] como el conjunto de distribuciones cuya función de densidad puede ser expresada mediante la siguiente forma:

$$f(y|\theta,\phi) = exp\{[y\theta - b(\theta)]/a(\phi) + c(\theta,\phi)\}I_{(Y)}(y) \tag{2-1}$$

En donde $a(.)$, $b(.)$ y $c(.)$ son funciones específicas y en caso de que y sea una variable aleatoria continua la ecuación 2-2 respresenta una función de densidad y en el caso de que y sea una variable discreta la ecuación 2-2 representa un función de probabilidad de masa. La función $I_{(Y)}(y)$ es una función indicadora de y en Y, es decir, la función $I_{(Y)}(y)$ toma valor 1 si $y \in Y$ y 0 en cualquier otro caso. Esta función determina el dominio de la misma ya que la función $f(y|\theta,\tau)$ sea diferente de cero $I_{(Y)}(y)$ debe ser igual a 1.

En esta definición se tiene que si ϕ es conocido la función de de densidad representa la familia exponencial con parámetro canónico y si ϕ es desconocido se tiene la familia exponencial biparamétrica [McCullagh, 1983]. En esta definición ϕ tiene una connotación de parámetro de dispersión, lo cual, tiene ventajas en el modelamiento debido a que en un contexto de regresión el efecto de las covariables sobre el parámetro de dispersión es mucho más interpretable que por ejemplo con respecto a un parámetro de forma.

Otra definición de la familia exponencial biparamétrica se da en los trabajos de [Efron, 1986, Gelfand & Dalal, 1990] se define a la familia exponencial biparamétrica como la familia de distribución cuya función de densidad se puede expresar de la siguiente manera:

$$f(y|\theta,\tau) = b(y)exp(\theta y + \tau T(y) - \rho(\theta,\tau))I_{(Y)}(y) \tag{2-2}$$

Diversos autores han estudiado esta familia de distribuciones encontrando propiedades interesantes dentro de las cuales se resalta que por ejemplo [Dey et al., 1997] muestra que si la ecuación 2-2 es integrable con respecto a $y \in Y$, y si $T(y)$ es convexa, para una media

dada se tiene que $V(y)$ es creciente con respecto a τ, de donde se tiene la noción que τ es un parámetro de dispersión. Esto resulta muy interesante en el contexto de los GLM ya que este vínculo permite llevar a cabo el modelamiento de la varianza $V(y)$ a través del parámetro de dispersión τ e incluso, se puede modelar el parámetro τ manteniendo interpretabilidad.

Otro aspecto importante a considerar es que bajo condiciones de regularidad de Cramér-Rao, [Zacks, 2014] al reparametrizar la función en términos de la media μ y la dispersión τ estos dos parámetros son ortogonales en el sentido de [Cox & Reid, 1987] y [Barndorff, 2014]. Esta ortogonalidad de parámetros implica que la matríz de información de Fisher es una matriz diagonal, lo cual, facilita el trabajo al llevar a cabo la estimación mediante métodos de máximización de verosimilitud ya que incluso se pueden llevar a cabo de manera separada tal como lo muestra [Cepeda, 2001b].

La tabla **2-2** algunos ejemplos de distribuciones que pertenecen a la familia exponencial definida en la ecuación 2-2. En este caso, tal como lo muestra [Cepeda, 2016] la distribución beta pertenece a esta familia con la reparametrización $z = log[y/(1-y)]$.

Distribución	θ	τ	T(y)	$\rho(\theta,\tau)$
Normal	μ/σ^2	$1/2\sigma^2$	-y^2	$0.5\theta^2\sigma^2 - 0{,}5log(2\pi\sigma^2)$
Gamma	-α/μ	$\alpha-1$	log(y)	$\log(\ \Gamma(\alpha)(-\theta)^{-\alpha})$
Beta	$\mu\phi$	ϕ	log(1-y)	$\log[\Gamma(\phi)/\Gamma(\mu\phi)\Gamma(\phi(1-\phi))]$

Tabla 2-1.: Distribuciones Familia Exponencial 1

En complemento a las distribuciones presentadas anteriormente, cuando $\tau = 0$ se tiene la familia exponencial uniparamétrica. La distribución de Poisson es un ejemplo de distribución que pertenece a esta familia con $\theta = log(\lambda)$ y $\rho(\theta,\tau) = exp(\theta)$. La distribución binomial también pertenece a esta familia con $\theta = logit(p)$ y $\rho(\theta,\tau) = nlog(1/(1+exp(\theta)))$.

En cuanto a la definición de la familia exponencial dada por la ecuación 2-1, [McCullagh, 1983] muestra algunas distribuciones que pertenecen a esta distribución entre las cuales se encuentran la distribución Normal, la distribución Gamma, la distribución de Poisson, la distribución binomial y la distribución inversa Gausiana tal como lo muestra la tabla **2-2**.

2.2. Modelos Lineales Doblemente Generalizados

Como fue explicado anteriormente, los DGML pueden ser entendidos como una generalización de los GLM para el caso en el que existe dispersión variable, por lo tanto, acorde con

Distribución	θ	b(θ)	a(ϕ)	C(y, ϕ)
Normal	μ	θ^2	$2\sigma^2$	-0.5($\frac{y^2}{\mu^2}$ + ln($2\pi\sigma^2$))
Gamma	$-\frac{1}{\mu}$	ln(-$\theta\kappa$)	$\frac{1}{\kappa}$	$(\kappa-1)ln(y) - ln(\Gamma(\kappa))$
Binomial	logit(p)	ln(p) -θ	$\frac{1}{m}$	$ln\binom{m}{y}$
Poisson	ln(λ)	exp(θ)	1	-ln(y!)

Tabla 2-2.: Distribuciones Familia Exponencial 2

[Smyth & Verbyla, 1999] puede ser entendido como un conjunto de dos submodelos en el cual por un lado se modela la media de la variable y por el otro se modela la dispersión o algún otro parámetro de interés. De manera un poco mas formal, tal como lo plantean [Cepeda, 2001a, Cepeda, 2016] si se tienen $y_i \ \forall i = 1, ..., n$ variables aleatorias independientes los DGLM, de manera análoga a los GLM, tienen tres componentes.

Definición 2.2.1 ***Modelos Lineales Doblemente Generalizados - DGLM:*** *Sean Y_i $i = 1, ..., n$ variables aleatorias independientes, Los modelos lineales doblemente generalizados, DGLM, están definidos por las siguientes tres compomnentes:*

- *Las componentes aleatorias: Los componentes Y_i tienen distribución en la familia exponencial biparamétrica, con $E(Y_i) = \mu$ y $V(y_i) = \sigma_i^2$.*
- *Las componentes sistemáticas: El predictor lineal $\eta = (\eta_1, \eta_2)$ está determinado de manera que $\eta_1 = X_i\beta$ y $\eta_2 = Z_i\gamma$.*
- *Las funciones de enlace: Las funciones de enlace conectan la parte sistemática con la componente aletoria $h = (h_1, h_2)$ de manera que $h_1 = g(\mu_i)$ y $h_2 = (\tau_i)$.*

La primera componente del modelo, la componente aleatoria es el vector $Y = (y_1, ..., y_n)$ con elementos independientes con distribución en la familia exponencial biparamétrica. Por independencia se tiene que $Cov(y_i, y_j) = 0$, sin embargo, como los elementos no son identicamente distribuidos se puede atribuir a cada uno de ellos $E(y_i) = \mu_i$ y $Var(y_i) = \sigma_i^2$.

La segunda componente del modelo son las componentes sistemáticas, conformadas por los predictores lineales que componen el vector $scherzobeethoven9\eta = (\eta_1, \eta_2)$ y se define $\eta_1 = X\beta$ y $\eta_2 = Z\gamma$ donde X es una matriz de dimensión $n * p$ y Z es una matriz de dimensión $n * l$, β es un vector $p * 1$ y γ es un vector $l * 1$. Las matrices X y Z contienen en cada una de sus filas los valores de las covariables de cada uno de las n observaciones. Las variables en X modelarán un componente, usualmente la media y las variables Z modelarán otra componente, usualmente la dispersión.

La tercera componente del modelo son las funciones de enlace, las cuales, en este caso conectan las componentes sistemáticas con las componentes aleatorias, es decir, $h_1(\theta) = \eta_1$ y $h_2(\tau) = \eta_2$, en este caso, se está modelando el comportamiento de los parámetros θ y τ, además, h_1 y h_2 son funciones monótonas y diferenciables.

Ejemplo 2.2.1 ***Modelo de regresión Normal:*** *Considérese que y_i para $i = 1, ..., n$ una muestra de las variables aleatorias de interés y $x_i = (x_{i,1}, ..., x_{i,s})$ y $z_i = (z_{i,1}, ..., z_{i,k})$ para $i = 1, ..., n$ dos vectores de covariables y $\beta = (\beta_1, ..., \beta_s)$ y $\gamma = (\gamma_1, ..., \gamma_k)$ dos vectores de parámetros de regresión. Con las parametrizaciones de las ecuaciones 2-3 y 2-4 el modelo para la regresión normal heteroscedástica queda definido por las siguientes tres componentes:*

- *Componente aleatoria: La componente aleatoria esta dada por una muestra aleatoria de tamaño n de una variable aleatoria $y_i \sim N(\mu_i, \sigma_i^2)$ para $i = 1, ..., n$.*

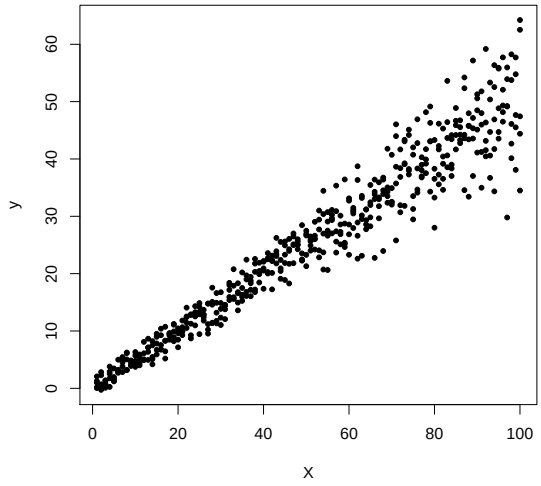
- *Dos componentes sistemáticas $\eta_{1,i} = X_i^T\beta$ y $\eta_{2,i} = Z_i^T\gamma$ para $i = 1, ..., n$.*

- *Dos funciones de enlace $h(.)$ y $g(.)$ de manera que $h(\mu_i) := \eta_{1,i}$ y $g(\sigma^2) := \eta_{2,i}$ para $i = 1, ..., n$. Donde dado que se tiene una variable aleatoria con distribución normal $h(\mu_i)$ debe tener rango en los reales, donde lo usual, es usar la función identidad dado que también es el enlace canónico para esta distribución; respecto a la función de enlace $g(\sigma^2)$ esta debe garantizar que σ^2 sea positivo, por lo que es muy usual utilizar el enlace logarítmico [Agresti, 2015].*

Ejemplo 2.2.2 ***Un ejemplo gráfico:*** *A manera de ejemplo considere la gráfica **2-1**. En ella se aprecia claramente que la variable dependiente y tiene una correlación positiva con la variable independiente. Observando más detenidamente, se observa tambien una relación positiva entre los valores de x y la amplitud del rango en que se observa la varaible y, es decir, entre mayor sea x más dispersos serán los valores de la variable dependiente.*

Estos datos fueron generados a partir de una distribución normal con un modelo para la media $\mu_i = 0{,}5x_i$ y un modelo de la varianza $ln(\sigma_i^2) = 0{,}04x_i$ donde $y_i \sim N(0{,}5x_i, exp(0{,}04x_i))$. En este ejemplo se aprecian claramente las componentes de un DGLM, la variable y es la componente aleatoria del modelo, segundo, los dos predictores lineales $\eta_1 = 0{,}5x_i$ el de la media y $\eta_2 = 0{,}04x_i$ el de la varianza. Finalmente las dos funciones de enlace, la función identidad para el modelo de la media y el enlace logarítmico para el modelo de la varianza.

Figura 2-1.: Gráfico de dispersión de Datos con Varianza Variable



2.2.1. Regresion Normal Heteroscedástica

La distribución Normal es quizá la distribución más representativa o conocida en el modelamiento de datos y quizá la más importante en ámbitos aplicados, es muy utilizada en el modelamiento de diversos fenómenos cuando se tienen variables de valor real. Su importancia o representatividad se debe entre otras cosas al teorema del límite central, al principio de regresión a la media y al amplio desarrollo de métodos de estimación e inferencia estadística.

Una variable aleatoria con distribución normal tiene la siguiente función de densidad:

$$f(y, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp \frac{1}{2\sigma^2}(y-\mu)^2] \tag{2-3}$$

Con esta parametrización se tiene que $E(y) = \mu$ y $Var(y) = \sigma^2$. Una parametrización alternativa consiste en expresar la función de densidad en términos del valor esperado μ y la precisión τ, tal como en la ecuación 2-4.

$$f(y, \mu, \tau) = \sqrt{\frac{\tau}{2\pi}} exp(-\frac{\tau}{2}(y-\mu)^2) \tag{2-4}$$

Modelo de Regresión Normal Heteroscedástica

Considerese que y_i para $i = 1, ..., n$ una muestra de las variables aleatorias de interés y $x_i = (x_{i,1}, ..., x_{i,s})$ y $z_i = (z_{i,1}, ..., z_{i,k})$ para $i = 1, ..., n$ dos vectores de covariables y

$\beta = (\beta_1, ..., \beta_s)$ y $\gamma = (\gamma_1, ..., \gamma_k)$ dos vectores de parámetros de regresión. Con las parametrizaciones de las ecuaciones 2-3 y 2-4 el modelo para la regresión normal heteroscedástica queda definido por las siguientes tres componentes:

- Componente aleatoria: La componente aleatoria esta dada por una muestra aleatoria de tamaño n de una variable aleatoria $y_i \sim N(\mu_i, \sigma_i^2)$ para $i = 1, ..., n$.

- Dos componentes sistemáticas $\eta_{1,i} = X_i^T \beta$ y $\eta_{2,i} = Z_i^T \gamma$ para $i = 1, ..., n$.

- Dos funciones de enlace $h(.)$ y $g(.)$ de manera que $h(\mu_i) := \eta_{1,i}$ y $g(\sigma^2) := \eta_{2,i}$ para $i = 1, ..., n$. Donde dado que se tiene una variable aleatoria con distribución normal $h(\mu_i)$ debe tener rango en los reales, donde lo usual, es usar la función identidad dado que también es el enlace canónico para esta distribución; respecto a la función de enlace $g(\sigma^2)$ esta debe garantizar que σ^2 sea positivo, por lo que es muy usual utilizar el enlace logarítmico [Agresti, 2015].

Para el modelo planteado en la ecuación 2-4 el planteamiento del modelo es exactamente igual al anterior dado que el parámetros de precisión τ tambien debe ser positivo, por lo tanto, es usual utilizar el enlace logarítmico. Diversos trabajos en torno al modelamiento de la varianza en modelos de regresión normal han surgido a lo largo del tiempo. [Harvey, 1976] y [Aitkin, 1987] plantearon un modelo de regresion normal heteroscedástico con una función de enlace logarítmica para el modelo de la varianza. [Smyth & Verbyla, 1999] generaliza este modelos para la familia exponencial y propone métodos de quasi verosimilitud para su estimación y [Verbyla, 1993] también propone estimación por máxima verosimilitud y máxima verosimilitud residual para este mismo modelo. [Smyth & Verbyla, 1999] propone estimaciones por máxima verosimilitud para diversos modelos entre ellos la distribución normal. En el ámbito bayesiano, [Cepeda, 2001b] y [Cepeda, 2001a] presentan propuestas para la estimación bayesiana del modelo normal heteroscedástico, posteriormente [Cepeda & Achcar, 2010] introduce estructuras de regresión no lineales para este mismo modelo.

2.2.2. Regresión Gamma

La distribución gamma es una distribución muy útil en el modelamiento de muchos fenómenos cuando se tienen variables aleatorias restringidas sobre los reales positivos. Los modelos de regresión gamma han sido utilizados en diversos contextos, algunas aplicaciones incluyen desde el análisis de sobrevida para presentar funciones de riesgo como lo muestra

[Zhang et al., 2014], modelación de ingresos en economía [Salem & Mount, 1974], modelamiento de tiempos en epidemiologia [Bateson, 2009] y series de tiempo para modelamiento de flujos en [Cepeda et al., 2012a].

Distribución Gamma y Modelos de Regresión Gamma

Una variable aletoria Y con distribución Gamma, es decir $Y \sim \Gamma(\alpha, \kappa)$ tiene la siguiente función de densidad de probabilidad:

$$f(y|\kappa,\theta) = \frac{1}{\Gamma(\kappa)\theta^{\kappa}} y^{\kappa-1} exp(-\frac{y}{\theta}) I_{(0,\infty)}(y) \tag{2-5}$$

Donde $\kappa > 0$ es un parámetro de forma y $\theta > 0$ es un parámetro de escala y $y \in (0, \infty)$. Con esta parametrización, la variable y tiene valor esperado y varianza dados por $E(y) = \kappa\theta$ y $Var(y) = \kappa\theta^2$. Otra parametrización muy conocida de la distribución gamma se basa en los parámetros de forma y tasa, es decir, una variable aleatoria con distribución gamma tal que $y \sim \Gamma(\alpha, \lambda)$ tiene la siguiente función de densidad:

$$f(y|\alpha,\lambda) = \frac{1}{\Gamma(\alpha)} \lambda^{\alpha} y^{\alpha-1} exp(-\lambda y) I_{(0,\infty)}(y) \tag{2-6}$$

Donde $\alpha > 0$ es un parámetro de forma y $\lambda > 0$ es un parámetro de tasa. En el caso de esta forma de la función de densidad de una variable aleatoria con distribución gamma se tiene que $E(y) = \frac{\alpha}{\lambda}$ y $Var(y) = \frac{\alpha}{\lambda^2}$. Además de lo anterior, es importante resaltar que si $\lambda = \frac{1}{\theta}$, las ecuaciones 2-5 y 2-6 son equivalentes.

Las anteriores parametrizaciones son muy conocidas y por lo general son las que se encuentran en referencias de esta distribución, no obstante, con el fin de lograr mayor interpretación la función de densidad Gamma se puede reparametrizar en función de la media y la precisión tal como lo muestra la ecuación 2-7 partiendo de la ecuación 2-5 con $E(y) = \mu = \kappa\theta$.

$$f(y|\kappa,\mu) = \frac{1}{\Gamma(\kappa)} (\frac{\kappa}{\mu})^{\kappa} y^{\kappa-1} exp(-\frac{\kappa}{\mu} y) I_{(0,\infty)}(y) \tag{2-7}$$

Donde la función de densidad queda parametrizada con respecto a la media μ y a la forma κ y por lo tanto, $E(y) = \mu$ y $Var(y) = \frac{\mu^2}{\kappa}$. De la expresión de la varianza se aprecia que para un valor fijo de μ la relación entre κ y $V(y)$ es inversa para un valor dado de μ, por lo tanto, es posible darle a κ una connotación de parámetro de precisión. Esta misma parametrización se obtiene si se toma de la ecuación 2-6 y se hace que $E(y) = \mu = \frac{\alpha}{\beta}$ y por lo tanto $\beta = \frac{\alpha}{\mu}$.

Modelo para media y precisión de distribución Gamma

Considerese que y_i para $i = 1, ..., n$ una muestra de las variables aleatorias de interés y $x_i = (x_{i,1}, ..., x_{i,s})$ y $z_i = (z_{i,1}, ..., z_{i,k})$ para $i = 1, ..., n$ dos vectores de covariables y $\beta = (\beta_1, ..., \beta_s)$ y $\gamma = (\gamma_1, ..., \gamma_k)$ dos vectores de parámetros de regresión. Con la parametrización de las ecuaciones 2-7 el modelo para la regresión Gamma queda definido de la siguiente manera:

- Componente aleatoria: La componente aleatoria esta dada por una muestra aleatoria de tamaño n de una variable aleatoria $y_i \sim \Gamma(\mu_i, \kappa_i)$ para $i = 1, ..., n$.

- Dos componentes sistemáticas $\eta_{1,i} = X_i^T \beta$ y $\eta_{2,i} = Z_i^T \gamma$ para $i = 1, ..., n$.

- Dos funciones de enlace $h(.)$ y $g(.)$ de manera que $h(\mu_i) := \eta_{1,i}$ y $g(\sigma^2) := \eta_{2,i}$ para $i = 1, ..., n$. Donde dado que se tiene una variable aleatoria con distribución gamma, $h(\mu_i)$ y $g(\kappa_i)$ debe garantizar que μ_i y κ_i tomen valores positivos, por lo tanto, en este caso es muy usual utilizar el enlace logarítmico [McCullagh, 1983] y [Agresti, 2015].

El modelo de regresión gamma se remonta a los desarrollos propuestos por [Nelder & Wedderburn, 1972] y [McCullagh, 1983] como un modelo lineal generalizado con un parámetro de dispersión. [Smyth & Verbyla, 1999] considera el modelo gamma dentro de los DGLM y propone estimaciones por máxima verosimilitud para esta familia de modelos. En el ámbito de estimación bayesiana [Cepeda, 2001a] plantea la estimación del modelo gamma dentro de la familia exponencial biparamétrica en el contexto bayesiano teniendo una parametrización con respecto a la media y a un parámetro de forma al que se le puede dar una implicación de parámetro de dispersión. [Cepeda & Corrales, 2015] desarrolla el paquete *Gammareg* para software *R*, en el cual, se utiliza el método de fisher scoring para llevar a cabo la estimación de los parámetros de media y forma siguiendo los planteamientos teóricos realizador por [Cepeda, 2001a].

2.2.3. Regresion Beta

Uno de los problemas de modelamiento de datos más comunes involucra datos de naturaleza binaria, es decir, datos que tienen dos categorías y cuyo modelamiento se enfoca principalmente en estimar la probabilidad de pertenecer a la categoría de referencia o modelar la proporción de individuos que poseen una característica de interés. Este tipo de datos es usualmente modelado mediante la distribución binomial si los datos son agrupados o Bernoulli si no lo son, sin embargo, como se observará a lo largo de este capítulo, la distribución beta permite mayor flexibilidad en el modelamiento de este tipo de datos ya que como lo explica [Cepeda, 2001b] y [Ferrari & Cribari, 2004] los datos binarios usualmente presentan dispersión variable.

Distribución Beta y modelos de regresión Beta

Una variable aleatoria Y con distribución Beta, es decir $Y \sim Beta(\alpha, \beta)$ tiene la siguiente función de densidad de probabilidad:

$$f(y|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} y^{\alpha-1}(1-y)^{\beta-1} I_{(0,1)}(y) \tag{2-8}$$

Donde $y \in [0, 1]$, $\alpha > 0$ y $\beta > 0$ y $B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha,\beta)}$ donde $\Gamma(.)$ es la función Gamma y los parámetros α y β son dos parámetros de forma. Con esta paramterización, la variable aletoria y tiene el siguiente valor esperado y varianza:

$$E[y] = \mu = \frac{\alpha}{\alpha + \beta}$$

$$Var[y] = \sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

En el contexto específico de regresión beta es posible expresar la función de densidad de probabilidad en términos de su media y un parámetro de precisión, ya que estos parámetros tiene mayor interpretabilidad que los parámetros de forma de la ecuación 2-8. Para esto se hace $\phi = \alpha + \beta$ y por lo tanto $\alpha = \mu\phi$, $q = \phi(1-\mu)$ y $\sigma^2 = \frac{\mu(1-\mu)}{\phi+1}$, en donde se asocia que ϕ es un parámetro de precisión en el sentido que, para una media dada, tiene una relación inversa con la varianza. Con esta reparametrización la función de densidad de probabilidad de una variable aleatoria y con distribución beta, es decir $y \sim (\mu, \phi)$, es la siguiente:

$$f(y|\mu, \phi) = \frac{1}{B(\phi\mu, \phi(1-\mu))} y^{\mu\phi-1}(1-y)^{\phi(1-\mu)-1} I_{(0,1)}(y) \tag{2-9}$$

La anterior ecuación tiene la ventaja de estar expresada en términos de los parámetros de valor esperado y precisión, los cuales son más fáciles de interpetar que los parámetros α y β de la ecuación , por lo tanto, aunque más compleja, la anterior parametrización resulta muy útil en el contexto de un DGLM ya permite modelar directamente los parámetros de media y precisión, los cuales, como se expuso anteriormente son de suma importancia en contextos aplicados.

Modelo para media y Precisión de la distribución beta

Considérese que y_i para $i = 1, ..., n$ una muestra de las variables aleatorias de interés y $x_i = (x_{i,1}, ..., x_{i,s})$ y $z_i = (z_{i,1}, ..., z_{i,k})$ para $i = 1, ..., n$ dos vectores de covariables y $\beta = (\beta_1, ..., \beta_s)$ y $\gamma = (\gamma_1, ..., \gamma_k)$ dos vectores de parámetros de regresión. Con la parametrización de las ecuaciones 2-9 y el modelo para la regresión beta queda definido de la siguiente manera:

- Componente aleatoria: La componente aleatoria esta dada por una muestra aleatoria de tamaño n de una variable aleatoria $y_i \sim Beta(\mu_i, \phi_i)$ para $i = 1, ..., n$.

- Dos componentes sistemáticas $\eta_{1,i} = X_i^T\beta$ y $\eta_{2,i} = Z_i^T\gamma$ para $i = 1, ..., n$.

- Dos funciones de enlace $h(.)$ y $g(.)$ de manera que $h(\mu_i) := \eta_{1,i}$ y $g(\sigma^2) := \eta_{2,i}$ para $i = 1, ..., n$. Donde dado que se tiene una variable aleatoria con distribución beta, $h(\mu_i)$ debe garantizar que μ_i tome valores en el intervalo $(0, 1)$, por lo cual, lo más usual es utilizar el enlace logit, aunque existen otras alternativas como probit y log-log complementario. Para la función de enlace $g(\phi_i)$ se debe considerar que ϕ debe ser positivo, por lo tanto, en este caso es muy usual utilizar el enlace logarítmico como se muestra en [McCullagh, 1983] y [Agresti, 2015].

Para el modelo planteado en la ecuación 2-8 el planteamiento del modelo es exactamente igual al anterior salvo por el hecho que las funciones de enlace en este caso deben garantizar que los parámetros α y β sean positivos, por lo que en ese caso lo usual sería utilizar el enlace logarítmico en ambos casos.

El modelo de regresión beta fue planteado por [Cepeda, 2001a] como un modelo en el cual se consideran dos estructuras de regresión uno para el parámetro de media y otra para el parámetro de dispersión, en este trabajo, se plantea tanto la estimación clásica como la estimación bayesiana de este modelo. [Ferrari & Cribari, 2004] presenta el modelo de regresión beta con dispersión constante y estimaciones de máxima verosimilitud posteriormente [Vasconcellos & Cribari, 2005] y [Simas et al., 2010] proponen formas de corregir el sesgo de estimación por máxima verosimilitud al modelo propuesto por [Ferrari & Cribari, 2004]. Un trabajo complementario se encuentra en [Cepeda, 2016] donde se muestra la estimación bayesiana y clásica para el modelo de regresión beta con estructura de regresión para los parámetros de media y dispersión.

En el ámbito aplicado [Cepeda & Núñez, 2013] utiliza la distribución beta para estudiar la calidad de la educación en colombia y lleva a cabo una ampliación del modelo de regresión beta con componentes espaciales. [Cribari & Zeileis, 2010] desarrolla el paquete *betareg* para software R, el cual lleva a cabo estimaciones por máxima verosimilitud para modelos de regresión para los dos parámetros de la distribución.

2.3. Estimación Modelos Lineales Doblemente Generalizados Utilizando OpenBUGS y Stan

2.3.1. Programación del modelo en Stan

La programación de una familia de modelos como el definido en la sección 2.2 en Stan requiere primero definir los datos a utilizar en el modelo. En este caso, estos datos correponden a la variable respuesta y, las covariables X cada una declarada como un vector de tamańo N donde N es la cantidad de datos. Posterior a la declaración de los datos se presentan los parámetros a estimar, en este caso, corresponden al conjunto de parámetros del predictor lineal de la media, es decir, los parámetros β y al predictor lineal de la dispersión, es decir, los parámetros γ.

En la sección de parámetros transformados se llevan a cabo las modificaciones necesarias sobre los parámetros β y γ, en este caso se declaran los parámetros de la media y la dispersión teniendo en cuenta las funciones de enlace específicas de cada modelo y el predictor lineal. En la sección del modelo, se declaran las distribuciones a priori de los paámetros β y γ y se asigna la verosimilitud a la variable dependeinte y.

Finalmente en la sección de cantidades generadas se lleva a cabo la predicción de los valores de la variable dependiente y a través de la generación de muestras de la distribución predictiva y se calcula la log verosimilitud.

El siguiente fragmento de código de Stan es un pseudo código que muestra lo explicado anteriormente. Más adelante en la sección 2.4 se desarrollan simulaciones y aplicaciones específicas que permitirán ilustrar mejor la programación de estos modelos en Stan.

```
data{
    int<lower=0> N;
    real x[N];
    real y[N];
}
parameters {
    real beta;
    real gamma;
}
transformed parameters{
    real[N] g(media) = X * beta;
    real[N] h(dispersion) = X * gamma;
```

```
     }
model{
    beta ~ distribucion_prior();
    gamma ~ distribucion_prior();
    for (i in 1:N){
        y[i] ~ verosimilitud(media[i], dispersion[i]);
            }
        }
generated quantities{
    vector[N] y_test;
    vector[N] log_lik;
    for (i in 1:N){
        y_test[i] = generador_aleatorio(media[i], dispersion[i]);
        log_lik[i] = log_verosimilitud(y[i] | media[i], dispersion[i]);
    }
}
```

2.3.2. Programación del modelo en OpenBUGS

Tal como se presentó en la sección 1.2.2, la programacón de una familia de modelos como el definido en la sección 2.2 es más sencilla en OpenBUGS que en Stan. Aunque a diferencia de Stan en OPenBUGS no hay secciones de código delimitadas, en este trabajo se seguirá un orden en el cual lo primero es definir la función de verosimilitud y los calculos del parámetro de media y dispersión teniendo en cuenta las funciones de enlace. Posteriormente, se asignan las distribuciones a priori a los parámetros γ y β y finalmente, se declara la variable y_est la cual es la predicción de la variable dependiente.

El siguiente fragmento es un pseudo código que ilustra lo planteado anteriormente, más adelante en la sección 2.4 se desarrollan simulaciones y aplicaciones específicas que permitirán ilustrar mejor la programación de estos modelos en OpenBUGS.

```
model
{
    for (i in 1:N)
    {
    g(media[i]) <- beta*x
    h(precision[i]) <- gamma*x
    y[i] ~ verosimilitud(media[i], precision[i])
    }
    beta ~ distribucion_prior(0,0.0001);
    gamma ~ distribucion_prior(0,0.0001);
```

```
    for (i in 1:N)
    {
    y_est[i] ~ generador_aleatorio(media[i], tau[i])
    }
}
```

2.4. Simulaciones y Aplicaciones

2.4.1. Estudio de Simulación para la regresión Normal Heteroscedástica

Para este estudio de simulación se toma como referencia el trabajo de [Cepeda & Gamerman, 2000], se generaron 4 covariables denominadas x_1, x_2, x_3 y x_4 para un total de $n = 400$ observaciones. La variable x_1 se generó de manera que $x_{i,1} = 1 \quad \forall i = 1, ..., n$, esto con el fin de generar un valor de intercepto. Las otras covariables se generaron de manera aleatoria $x_{2,i} \sim U(0, 400)$, $x_{3,1} \sim U(10, 23)$ y $x_{4,i} \sim U(0, 10)$ $\forall i = 1, ..., n$. La variable respuesta y_i proviene de una distribución normal $y_i \sim N(\mu_i, \sigma_i^2) \quad \forall i = 1, ..., n$ de manera que $\mu_i = -35 + 0{,}35x_{2,i} - 1{,}7x_{3,1}$ y $ln(\sigma_i^2) = -8 + 0{,}026x_{2,i} - 0{,}4x_{4,i}$.

En esta simulación están claramente identificados los tres componentes de un proceso generador de datos que corresponde con el expuesto en la sección 2.2.1. En primera medida, está la componente aleatoria que en este caso corresponde las n observaciones de la variable y. Las componentes sistemáticas dadas por las estructuras de la media $\beta_1 + \beta_2 x_{2,i} - \beta_3 x_{3,1}$ y la varianza $\gamma_1 + \gamma_2 x_{2,i} + \gamma_3 x_{4,i}$ de y_i. Finalmente las funciones de enlace en este caso para la media es la función identidad y para la varianza el enlace logarítmico.

La estimación se lleva a cabo asignando la distribución a priori normal a cada uno de los parámetros $\gamma_j, \beta_j \quad \forall j = 1, 2, 3$ con media 0 y varianza 10^6, es decir, la distribución a priori para los parámetros es $\gamma_j, \beta_j \sim N(0, 10^6) \quad \forall j = 1, 2, 3$.

Parametrización del modelo Normal Heteroscedástico en Stan

Tras haber generado los datos con la estructura expuesta anteriormente se lleva a cabo la programación del modelo en Stan, la cual, sigue la estructura expuesta en la sección 3.2.1. Primero, se declaran los datos a utilizar en el modelo, en este caso corresponden a las variables Y, X_1, X_2, X_3, X_4 y la cantidad de obervaciones N. Posterior a la declaración de los datos se declaran los parámetros a estimar, en este caso son $\beta_i, \gamma_i \quad \forall i = 1, 2, 3$.

En la sección de parámetros transformados se declaran los parámetros de media y desviación estandar y se lleva a cabo su cálculo teniendo en cuenta las funciones de enlace presentadas anteriormente. En la sección del modelo, se declaran las distribuciones a priori normales de los paámetros $\beta_i, \gamma_i \forall i = 1, 2, 3$ y se asigna la log verosimilitud normal a la variable dependiente. Finalmente en la sección de cantidades generadas se lleva a cabo la predicción de los valores de la variable dependiente Y y la log verosimilitud.

```
data{
    int<lower=0> N;
    real x_1[N];
    real x_2[N];
    real x_3[N];
    real x_4[N];
    real y[N];
}
parameters {
    real beta_1;
    real beta_2;
    real beta_3;
    real gamma_1;
    real gamma_2;
    real gamma_3;
}
transformed parameters{
    real[N] mu;
    real[N] sigma;
    for (i in 1:N){
        mu[i] = beta_1*x_1[i]+beta_2*x_2[i] + beta_3*x_3[i];
        sigma[i] = sqrt(exp(gamma_1*x_1[i]+gamma_2*x_2[i]+
                   gamma_3*x_4[i]));
         }
     }
model{
    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
    beta_3 ~ normal(0,100);
    gamma_1 ~ normal(0,100);
    gamma_2 ~ normal(0,100);
    gamma_3 ~ normal(0,100);
    for (i in 1:N){
         y[i] ~ normal_lpdf( mu[i], sigma[i]);
```

```
            }
        }
    generated quantities{
        vector[N] y_test;
        vector[N] log_lik;
        for (i in 1:N){
            y_test[i] = normal_rng(mu[i], sigma[i]);
            log_lik[i] = normal_lpdf(y[i] | mu[i], sigma[i]);
        }
    }
```

El siguiente código muestra la ejecución del modelo normal heteroscedástico en *stan* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *stan* de la distribución *rstan*. Esta función toma el código de *stan* del archivo *normal.stan* y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud log_lik y de la distribución predictiva de la variable dependiente y_test.

```
data = list(x_1 = x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, y=y, N=n)

model <- rstan::stan(file = 'normal.stan', data = data,
                iter = 3000, pars = c('beta_1','beta_2', 'beta_3',
                                      'gamma_1', 'gamma_2', 'gamma_3',
                                      'log_lik', "y_test"),
                chains = 1)
```

Parametrización del modelo Normal Heteroscedástico en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud normal y se definen los predictores lineales, en este caso, está el predictor lineal de la media y la varianza, sin embargo, la parametrización de la distribución normal en OpenBUGS se hace con los parámetros de media y precisión $\tau = 1/\sigma^2$, razón por la cual, se hace una tranformación sobre el parámetro de varianza en el código.

Posteriormente, se asignan las distribuciones a priori normales par los parámetros $\beta_i, \gamma_i \forall i = 1, 2, 3$. Finalmente, se declara la variable y_est la cual es la predicción de la variable dependiente.

```
model
```

```
    {
        for (i in 1:N)
        {
        media[i] <- beta_1*x_1[i]+beta_2*x_2[i] + beta_3*x_3[i]
        sigma_2[i] <- exp(gamma_1*x_1[i] + gamma_2*x_2[i]+ gamma_3*x_4[i])
        tau[i] <- 1/sigma_2[i]
        y[i] ~ dnorm(media[i], tau[i])
        }
        beta_1 ~ dnorm(0,0.0001);
        beta_2 ~ dnorm(0,0.0001);
        beta_3 ~ dnorm(0,0.0001);
        gamma_1 ~ dnorm(0,0.0001);
        gamma_2 ~ dnorm(0,0.0001);
        gamma_3 ~ dnorm(0,0.0001);
        for (i in 1:N)
        {
        y_est[i] ~ dnorm(media[i], tau[i])
        }
    }
```

El siguiente código muestra la ejecución del modelo normal heteroscedástico en *OpenBUGS* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *bugs* del paquete *R2OpenBUGS*, esta función toma el código de *OpenBUGS* del archivo $model_normal.txt$ y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_test, la media *media* y el parámetro de dispersión τ.

```
data = list(x_1 = x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, y=y, N=n)

modelo_normal =R2OpenBUGS::bugs(data,
                 inits = NULL,
                 model.file="model_normal.txt",
                 parameters= c("beta_1", "beta_2", "beta_3",
                               "gamma_1", "gamma_2", "gamma_3",
                               "y_test", "media", "tau"),
                 n.chains = 3,
                 n.iter=1000
)
```

La tabla **2-3** muestra el resumen de la estimación realizada para este modelo utilizando Stan, la cual es bastante cercana a los valores reales de los parámetros con que se simularon

los datos. Además se puede ver que los valores reales del parámetro están contenidos en los intervalos de credibilidad comprendidos entre el 2,5 % y el 97,5 % de los valores muestreados de la distribución a posteriori.

Tabla 2-3.: Resumen de Estimación Modelo Normal Heteroscedástico usando Stan

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
β_1	-35.00	0.00	0.00	-35.01	-35.00	-35.00	-34.99	-34.99	11762.21	1.00
β_2	0.35	0.00	0.00	0.35	0.35	0.35	0.35	0.35	19858.29	1.00
β_3	-1.70	0.00	0.00	-1.70	-1.70	-1.70	-1.70	-1.70	11986.12	1.00
γ_1	-8.11	0.00	0.18	-8.46	-8.23	-8.12	-7.99	-7.75	2091.50	1.00
γ_2	0.03	0.00	0.00	0.03	0.03	0.03	0.03	0.03	5486.16	1.00
γ_3	-0.39	0.00	0.02	-0.43	-0.40	-0.39	-0.37	-0.34	1607.09	1.00
lp__	409.38	0.03	1.76	405.13	408.44	409.73	410.69	411.79	4353.19	1.00

La tabla **2-4** muestra la estimación de la correlación entre los parámetros de la simulación, es importante resaltar que en la correlación entre los parámetros de la estrucutra de la media, es decir, los parámetros β y los parámetros de la estructura de la varianza, es decir, los parámetros γ es baja. Así mismo, en concordancia con [Cepeda & Gamerman, 2000] existe una fuerte correlación entre los parámetros β_1 y β_3 y los parámetros γ_1 y γ_2 y γ_1 y γ_3.

Tabla 2-4.: Correlación a posteriori

	β_1	β_2	β_3	γ_1	γ_2	γ_3
β_1	1					
β_2	-0.147	1				
β_3	-0.962	-0.053	1			
γ_1	-0.072	-0.011	0.077	1		
γ_2	0.025	0.064	-0.039	-0.633	1	
γ_3	0.087	-0.057	-0.08	-0.613	-0.072	1

2.4.2. Estudio de Simulación para la regresión Gamma

Para este estudio de simulación se generaron 4 covariables denominadas x_1, x_2, x_3 y x_4 para un total de $n = 500$ observaciones. La variable x_1 se generó de manera que $x_{i,1} = 1 \quad \forall i = 1, ..., n$, esto con el fin de generar un valor de intercepto. Las otras covariables se generaron de manera aleatoria de manera que $x_{2,i} \sim U(0, 30)$, $x_{3,1} \sim U(0, 15)$ y $x_{1,i} \sim U(10, 20)$ $\forall i = 1, ..., n$. La variable respuesta y_i se proviene de una distribución gamma con función de densidad dada por la ecuación 2-7, $y_i \sim \Gamma(\mu_i, \alpha_i) \quad \forall i = 1, ..., n$, de manera que

$ln(\mu_i) = -0{,}15 + 0{,}2x_{2,i} - 0{,}3 + x_{3,1}$ y $ln(\alpha_i) = -0{,}3x_{1,i} + 0{,}1x_{2,i} + 0{,}3x_{4,i}$.

En esta simulación están claramente identificados los tres componentes de un proceso generador de datos que corresponde con el expuesto en la sección 2.2.2. En primera medida, está la componente aleatoria que en este caso corresponde las n observaciones de la variable y. Las componentes sistemáticas dadas por las estructuras de la media $\beta_1 + \beta_2 x_{2,i} - \beta_3 x_{3,1}$ y la precisión $\gamma_1 + \gamma_2 x_{2,i} + \gamma_3 x_{4,i}$ de y_i. Finalmente las funciones de enlace en este caso para la media y la precision son el enlace logarítmico.

La estimación se lleva a cabo asignando la distribución a priori normal a cada uno de los parámetros $\gamma_j, \beta_j \quad \forall j = 1, 2, 3$ con media 0 y varianza 10^4, es decir, la distribución a priori para los parámetros es $\gamma_j, \beta_j \sim N(0, 10^4) \quad \forall j = 1, 2, 3$.

Parametrización del modelo Gamma en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud gamma y se definen los predictores lineales para la media y el parámetro de precisión y las transformaciones necesarias de los parámetros dado que en OpenBUGS la distribución gamma no está parametrizada con respecto a la media y la dispersión sino como lo muestra la ecuación 2-5.

Posteriormente, se asignan las distribuciones a priori normales para los parámetros γ y β. En último lugar, se declara la variable y_est la cual es la predicción de la variable dependiente.

```
model
{
    for (i in 1:N)
    {
        alpha[i] <- exp(gamma_1*x_1[i] + gamma_2*x_2[i] + gamma_3*x_4[i]);
        media[i] <- exp(beta_1*x_1[i] + beta_2*x_2[i] + beta_3*x_3[i]);
        mu[i] <- alpha[i]/media[i];
        y[i] ~ dgamma(alpha[i],mu[i]);
    }
    beta_1 ~ dnorm(0,0.0001);
    beta_2 ~ dnorm(0,0.0001);
    beta_3 ~ dnorm(0,0.0001);
    gamma_1 ~ dnorm(0,0.0001);
    gamma_2 ~ dnorm(0,0.0001);
    gamma_3 ~ dnorm(0,0.0001);
```

```
        for (i in 1:N)
        {
            alpha_est[i] <- exp(gamma_1*x_1[i] + gamma_2*x_2[i] + gamma_3*x_4[i]);
            media_est[i] <- exp(beta_1*x_1[i] + beta_2*x_2[i] + beta_3*x_3[i]);
            mu_est[i] <- alpha_est[i]/media_est[i];
            y_est[i] ~ dgamma(alpha_est[i],mu_est[i]);
        }
    }
```

El siguiente código muestra la ejecución del modelo gamma en *OpenBUGS* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *bugs* del paquete *R2OpenBUGS*, esta función toma el código de *OpenBUGS* del archivo *gamma.txt* y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
data <- list(y=y, x_1=x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, N = N)

modelo_gamma = R2OpenBUGS::bugs(data=data,
                                inits = NULL,
                                model.file = "gamma.txt",
                                parameters= c("beta_1", "beta_2", "beta_3","gamma_1",
"gamma_2", "gamma_3",
                                "y_est"),
                                n.chains = 3,
                                n.iter=10000)
```

Parametrización del modelo Gamma en Stan

Tras haber generado los datos con la estructura expuesta anteriormente se lleva a cabo la programación del modelo en Stan, la cual, sigue la estructura expuesta en la sección 3.2.1. Primero, se declaran los datos a utilizar en el modelo, en este caso corresponden a las variables Y, X_1, X_2, X_3, X_4 y la cantidad de obervaciones N. Posterior a la declaración de los datos se declaran los parámetros a estimar, en este caso son $\beta_i, \gamma_i \quad \forall i = 1, 2, 3$.

En la sección de parámetros transformados se declaran y calculan los parámetros de media y precisión y se transforman estos parámetros dado que en Stan la distribución Gamma no esta parametrizada con respecto a la media y a la precisión sino como en la ecuación 2-5.

En la sección del modelo, se declaran las distribuciones a priori normales de los paámetros $\beta_i, \gamma_i \forall i = 1, 2, 3$ y se asigna la log verosimilitud gamma a la variable dependiente. Finalmente en la sección de cantidades generadas se lleva a cabo la predicción de los valores de la variable dependiente Y y la log verosimilitud.

```
data{
    int<lower=0> N;
    real x_1[N];
    real x_2[N];
    real x_3[N];
    real x_4[N];
    real y[N];
}
parameters {
    real beta_1;
    real beta_2;
    real beta_3;
    real gamma_1;
    real gamma_2;
    real gamma_3;
}
transformed parameters{
    real mu[N];
    real alpha[N];
    real theta[N];
    for (i in 1:N){
         alpha[i] = exp(gamma_1*x_1[i] + gamma_2*x_2[i] + gamma_3*x_4[i]);
         mu[i] = exp(beta_1*x_1[i]+beta_2*x_2[i] + beta_3*x_3[i]);
         theta[i] = alpha[i]/mu[i];
    }
}
model{
    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
    beta_3 ~ normal(0,100);
    gamma_1 ~ normal(0,100);
    gamma_2 ~ normal(0,100);
    gamma_3 ~ normal(0,100);
    for (i in 1:N){
        y[i] ~ gamma_lpdf(alpha[i],theta[i]);
            }
```

```
}
generated quantities{
  vector[N] y_test;
  vector[N] log_lik;
  for (i in 1:N){
      y_test[i] = gamma_rng(alpha[i],theta[i]);
      log_lik[i] = gamma_lpdf(y[i] | alpha[i],theta[i]);
      }
}
```

El siguiente código muestra la ejecución del modelo gamma en *stan* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *stan* de la distribución *rstan*. Esta función toma el código de *stan* del archivo *gamma.stan* y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud log_ik y de la distribución predictiva de la variable dependiente y_test.

```
data <- list(y=y, x_1=x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, N = N)

modelo_gamma = rstan::stan(data=data,
                           file = "gamma.stan",
                           pars= c("beta_1", "beta_2", "beta_3", "gamma_1", "gamma_2",
"gamma_3", "y_test", "log_lik"),
                           chains = 4,
                           iter=10000
                           )
```

La tabla **2-5** muestra el resumen del resultado de la estimación realizada para este modelo utilizando Stan, en la cual, se tiene que la estimación es bastante cercana a los valores de los parámetros. Además, se puede observar que los valores reales del parámetro están contenidos en los intervalos de credibilidad comprendidos entre el 2,5 % y el 97,5 % de los valores muestreados de la distribución a posteriori.

La tabla **2-6** muestra la estimación de la correlación entre los parámetros de la simulación, es importante resaltar que la correlación entre los parámetros de la estructura de la media, es decir, los parámetros β y los parámetros de la estructura de la varianza, es decir, los parámetros γ es muy baja. Así mismo, existe una fuerte correlación entre los interceptos y los otros parámetros.

Tabla 2-5.: Resumen de Estimación Modelo Gamma

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
β_1	0.15	0.00	0.01	0.14	0.15	0.15	0.16	0.16	11998.98	1.00
β_2	0.20	0.00	0.00	0.20	0.20	0.20	0.20	0.20	13397.16	1.00
β_3	-0.30	0.00	0.00	-0.30	-0.30	-0.30	-0.30	-0.30	19200.64	1.00
γ_1	-0.65	0.00	0.33	-1.32	-0.88	-0.65	-0.42	-0.00	7633.27	1.00
γ_2	0.10	0.00	0.01	0.08	0.09	0.10	0.10	0.11	11345.72	1.00
γ_3	0.32	0.00	0.02	0.28	0.31	0.32	0.34	0.36	7856.58	1.00
lp__	253.07	0.02	1.75	248.86	252.14	253.41	254.35	255.45	7365.29	1.00

Tabla 2-6.: Correlación a posteriori modelo Gamma

	β_1	β_2	β_3	γ_1	γ_2	γ_3
beta_1	1					
beta_2	-0.83	1				
beta_3	-0.485	0	1			
gamma_1	-0.005	-0.016	0.002	1		
gamma_2	0.043	-0.051	0.003	-0.364	1	
gamma_3	-0.009	0.034	-0.002	-0.943	0.081	1

2.4.3. Estudio de Simulación para la regresión Beta

Para este estudio de simulación se generaron 4 covariables denominadas x_1, x_2, x_3 y x_4 para un total de $n = 500$ observaciones. La variable x_1 se generó de manera que $x_{i,1} = 1 \quad \forall i = 1, ..., n$, esto con el fin de generar un valor de intercepto. Las otras covariables se generaron de manera aleatoria de manera que $x_{2,i} \sim U(0, 30)$, $x_{3,1} \sim U(0, 15)$ y $x_{1,i} \sim U(10, 20)$ $\forall i = 1, ..., n$. La variable respuesta y_i se proviene de una distribución Beta con función de densidad dada por la ecuación 2-9, $y_i \sim \Gamma(\mu_i, \alpha_i) \quad \forall i = 1, ..., n$, de manera que $ln(\mu_i) = 0{,}2 + 0{,}1x_{2,i} - 0{,}1x_{3,1}$ y $ln(\alpha_i) = 0{,}2 + 0{,}1x_{2,i} + 0{,}1x_{4,i}$.

En esta simulación están claramente identificados los tres componentes de un proceso generador de datos que corresponde con el expuesto en la sección 2.2.3. En primera medida, está la componente aleatoria que en este caso corresponde las n observaciones de la variable y. Las componentes sistemáticas dadas por las estructuras de la media $\beta_1 + \beta_2 x_{2,i} + \beta_3 x_{3,1}$ y la precisión $\gamma_1 + \gamma_2 x_{2,i} + \gamma_3 x_{4,i}$ de y_i. Finalmente las funciones de enlace en este caso para la media es el enlace logit y para la precisión es el enlace logarítmico.

La estimación se lleva a cabo asignando la distribución a priori normal a cada uno de los parámetros $\gamma_j, \beta_j \quad \forall j = 1, 2, 3$ con media 0 y varianza 10^4, es decir, la distribución a priori para los parámetros es $\gamma_j, \beta_j \sim N(0, 10^4) \quad \forall j = 1, 2, 3$.

Parametrización del modelo Beta en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud beta y se definen los predictores lineales. En este caso específico, es necesario llevar a cabo ciertas transformaciones de variables dado que en OpenBUGS la parametrización de la distribución beta no es con respecto a la media y la dispersión sino que coincide con la de la ecuación 2-8.

Posteriormente, se asignan las distribuciones a priori normales para los parámetros $\gamma_j, \beta_j \quad \forall j = 1, 2, 3$ y por último se declara la variable y_est la cual es la predicción de la variable dependiente.

```
model
{
for(i in 1:N){
    y[i] ~ dbeta(a[i],b[i])
    a[i] <-((1-mu[i])*mu[i]*mu[i]- mu[i]*phi[i])/phi[i]
    b[i]<-(1-mu[i])*(mu[i]-mu[i]*mu[i]-phi[i])/phi[i]
    mu[i] <- exp(beta_1+ beta_2*x_1[i])/(1+exp(beta_1+ beta_2*x_1[i]))
    phi[i] <-exp(gamma_1+gamma_2*x_1[i])
}
beta_1 ~ dnorm(0,0.00001)
beta_2 ~ dnorm(0,0.00001)
gamma_1 ~ dnorm(0,0.00001)
gamma_2 ~ dnorm(0,0.00001)
for(i in 1:N){
    y_est[i] ~ dbeta(a[i],b[i])
    }
}
```

El siguiente código muestra la ejecución del modelo beta en *OpenBUGS* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *bugs* del paquete *R2OpenBUGS*, esta función toma el código de *OpenBUGS* del archivo *beta.txt* y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
data = list(x_1 = x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, y=y, N=n)
```

```
modelo_normal =R2OpenBUGS::bugs(data,
                  inits = NULL,
                  model.file="beta.txt",
                  parameters= c("beta_1", "beta_2", "beta_3",
                                "gamma_1", "gamma_2", "gamma_3",
                                "y_est"),
                  n.chains = 2,
                  n.iter=10000
)
```

Parametrización del modelo Beta en Stan

Tras haber generado los datos con la estructura expuesta anteriormente se lleva a cabo la programación del modelo en Stan, la cual, sigue la estructura expuesta en la sección 3.2.1. Primero, se declaran los datos a utilizar en el modelo, en este caso corresponden a las variables Y, X_1, X_2, X_3, X_4 y la cantidad de obervaciones N. Posterior a la declaración de los datos se declaran los parámetros a estimar, en este caso son $\beta_i, \gamma_i \quad \forall i = 1, 2, 3$.

En la sección de parámetros transformados se declaran y calculan los parámetros de media y precisión y se transforman estos parámetros dado que en Stan la distribución Beta no esta parametrizada con respecto a la media y a la precisión sino como en la ecuación 2-8. En la sección del modelo, se declaran las distribuciones a priori normales de los paámetros $\beta_i, \gamma_i \forall i = 1, 2, 3$ y se asigna la log verosimilitud beta a la variable dependiente. Finalmente en la sección de cantidades generadas se lleva a cabo la predicción de los valores de la variable dependiente Y y la log verosimilitud.

```
data{
    int<lower=0> N;
    real x_1[N];
    real x_2[N];
    real x_3[N];
    real x_4[N];
    real y[N];
}
parameters {
    real beta_1;
    real beta_2;
    real beta_3;
    real gamma_1;
```

```
    real gamma_2;
    real gamma_3;
}
transformed parameters{
   real mu[N];
   real phi[N];
   real alpha[N];
   real omega[N];
   for (i in 1:N){
       mu[i] = exp(beta_1*x_1[i]+beta_2*x_2[i]+
                   beta_3*x_3[i])/(1+exp(beta_1*x_1[i]+
                   beta_2*x_2[i]+beta_3*x_3[i]));
       phi[i] = exp(gamma_1*x_1[i] + gamma_2*x_2[i] + gamma_3*x_4[i]);
       omega[i] = phi[i]-phi[i]*mu[i];
       alpha[i] = phi[i]-omega[i];
   }
}
model{
    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
    beta_3 ~ normal(0,100);
    gamma_1 ~ normal(0,100);
    gamma_2 ~ normal(0,100);
    gamma_3 ~ normal(0,100);
    for (i in 1:N){
         y[i] ~ beta_lpdf(alpha[i], omega[i]);
            }
}
generated quantities{
    vector[N] y_test;
    vector[N] log_lik;
    for (i in 1:N){
        y_test[i] = beta_rng(alpha[i], omega[i]);
        log_lik[i] = beta_lpdf(y[i] | alpha[i], omega[i])
    }
}
```

El siguiente código muestra la ejecución del modelo gamma en *stan* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *stan* de la distribución *rstan*. Esta función toma el código de *stan* del archivo *beta.stan* y declara el número de iteraciones y el número de

cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud log_lik y de la distribución predictiva de la variable dependiente y_test.

```
data = list(x_1 = x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, y=y, N=n)

model <- rstan::stan(file = 'normal.stan', data = data,
                     iter = 10000, pars = c('beta_1','beta_2', 'beta_3',
                                            'gamma_1', 'gamma_2', 'gamma_3',
                                            'log_lik', "y_test"),
                     chains = 2)
```

La tabla **2-7** muestra el resumen del resultado de la estimación realizada para este modelo utilizando Stan, en la cual, se tiene una estimación a traves de la media a posteriori que es bastante cercana a los valores reales de los parámetros con que se simularon los datos, de igual manera, se puede ver que los valores reales del parámetro están contenidos en los intervalos de credibilidad comprendidos entre el 2,5 % y el 97,5 % de los valores muestreados de la distribución a posteriori aunque para el parámetro γ_1 el intervalo contiene el valor cero.

La tabla **2-8** muestra la estimación de la correlación entre los parámetros de la simulación, es importante resaltar que en la correlación entre los parámetros de la estrucutra de la media, es decir, los parámetros β y los parámetros de la estructura de la varianza, es decir, los parámetros γ. Así mismo, en concordancia con [Cepeda & Gamerman, 2000] existe una fuerte correlación entre los parámetros β_1 y β_3 y los parámetros γ_1 y γ_2 y γ_1 y γ_3.

Tabla 2-7.: Resumen de Estimación Modelo Beta

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
β_1	0.24	0.00	0.08	0.07	0.18	0.24	0.29	0.40	10854.20	1.00
β_2	0.10	0.00	0.00	0.09	0.09	0.10	0.10	0.10	12718.35	1.00
β_3	0.10	0.00	0.01	0.08	0.09	0.10	0.10	0.11	14588.10	1.00
γ_1	0.11	0.00	0.33	-0.55	-0.11	0.11	0.34	0.76	10760.69	1.00
γ_2	0.10	0.00	0.01	0.08	0.09	0.10	0.10	0.11	15288.65	1.00
γ_3	0.10	0.00	0.02	0.06	0.09	0.10	0.12	0.14	11212.42	1.00
lp__	853.83	0.02	1.75	849.55	852.89	854.17	855.12	856.21	7961.49	1.00

2.4.4. Aplicación con datos de Árboles de Cerezo

En esta aplicación se consideran los datos de árboles de cereza, este conjunto de datos consta de 31 observaciones y 3 variables [Ryan et al., 1976]. La primera describe el diámetro

Tabla 2-8.: Correlación a posteriori Modelo Beta

	β_1	β_2	β_3	γ_1	γ_2	γ_3
β_1	1					
β_2	-0.765	1				
β_3	-0.564	0.07	1			
γ_1	0.028	-0.048	0.06	1		
γ_2	-0.244	0.365	0.014	-0.288	1	
γ_3	0.06	-0.065	-0.028	-0.93	-0.025	1

del árbol en pulgadas, la segunda es la altura del árbol en pulgadas y la tercera el volumen de madera producida por el mismo en pulgadas cúbicas. Este conjunto de datos ha sido considerados por [Cepeda & Gamerman, 2000] en donde se considera el volumen del árbol como la variable dependiente en función del diámetro y la altura del árbol planteando un modelo lineal para la media de la raíz cúbica del volúmen de madera y un modelo con enlace logarítmico para la varianza de dicha variable, es decir, $V_i^{1/3} \sim N(\mu_i, \sigma_i^2)$ con $\mu_i = \beta_0 + \beta_1 * h_i + \beta_2 * d_i$ y $\sigma_i^2 = exp(\gamma_0 + \gamma_1 * h_i + \gamma_2 * d_i)$ $\forall i = 1, ..., 31$. Como distribuciones a priori del conjunto de parámetros γ_i y *beta* se seleccionaron distrobuciones normales $N(0, 10^k)$, en este documento se reportan los resultados para $k = 4$.

Parametrización del modelo en Stan

Para parametrizar el modelo en Stan, en la sección de datos se declaran tres variables *girth*, *heigh* y la variable dependiente *volume*, cada una como un vector de tamaño N que es la cantidad de observaciones del conjunto de datos. Los parámetros a estimar son el conjunto de parámetros β y γ los cuales son los predictores linerales del parámetro de media y de desviación estandar respectivamente.

Posterior a la declaración de parámetros se lleva a cabo la transformación de los mismos, en este caso, se lleva a cabo el calculo de los parámetros de media y desviación estandar. En la siguiente sección se lleva a cabo la declaración del modelo, en la cual se asignan las distribuciones prior para cada uno de los parámetros β y γ y se asigna a cada una de las observaciones de la variable *volume* la función de log verosimilitud de la distri bución normal.

Por último, se lleva a cabo el cálculo de la log verosimilitud y la predicción de la variable dependiente en la sección de cantidades generadas.

```
data{
    int<lower=0> N;
    real volume[N];
```

```
    real girth[N];
    real height[N];
}
parameters {
    real beta_0;
    real beta_1;
    real beta_2;
    real gamma_0;
    real gamma_1;
    real gamma_2;
}
transformed parameters{
   real mu[N];
   real sigma[N];
   for (i in 1:N){
       mu[i] = beta_0 + beta_1*height[i] + beta_2*girth[i];
       sigma[i] = sqrt(exp(gamma_0+  gamma_1*height[i]+gamma_2*girth[i]));
   }
}
model{
    beta_0 ~ normal(0,100);
    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
    gamma_0 ~ normal(0,100);
    gamma_1 ~ normal(0,100);
    gamma_2 ~ normal(0,100);
    for (i in 1:N){
        volume[i] ~ normal_lpdf( mu[i], sigma[i]);
            }
}
generated quantities{
vector[N] y_test;
vector[N] log_lik;
for (i in 1:N){
    y_test[i] = normal_rng(mu[i], sigma[i]);
    log_lik[i] = normal_lpdf(volume[i] | mu[i], sigma[i]);
    }
}
```

Parametrización del modelo en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud normal y se definen los predictores lineales, en este caso, está el predictor lineal de la media y la varianza, sin embargo, la parametrización de la distribución normal en OpenBUGS se hace con los parámetros de media y precisión $\tau = 1/\sigma^2$, razón por la cual, se hace una tranformación sobre el parámetro de varianza en el código.

Posteriormente, se asignan las distribuciones a priori normales par los parámetros γ y β. Finalmente, se declara la variable y_est la cual es la predicción de la variable dependiente.

```
model
{
    for (i in 1:N)
    {
        media[i] <- beta_0 + beta_1*height[i] + beta_2*girth[i];
        sigma_2[i] <- exp(gamma_0+ gamma_1*height[i]+gamma_2*girth[i]);
        tau[i] <- 1/sigma_2[i];
        volume[i] ~ dnorm(media[i], tau[i]);
    }

    beta_0 ~ dnorm(0,0.00001);
    beta_1 ~ dnorm(0,0.00001);
    beta_2 ~ dnorm(0,0.00001);
    gamma_0 ~ dnorm(0,0.00001);
    gamma_1 ~ dnorm(0,0.00001);
    gamma_2 ~ dnorm(0,0.00001);
    for (i in 1:N)
    {
        y_est[i] ~ dnorm(media[i], tau[i])
    }
}
```

Ejecución del modelo desde R

El siguiente fragmento de código de muestra el tratamiento de datos en R.

```
library('rstan')
library('R2OpenBUGS')
```

```
options(mc.cores = parallel::detectCores())
rstan::rstan_options(auto_write = TRUE)

rm(list = ls())
data("trees")

df_trees <- trees
df_trees$Volume <- (df_trees$Volume)^(1/3)
rm(trees)

N <- nrow(df_trees)
data = list(girth = df_trees$Girth,
              height = df_trees$Height,
              volume = df_trees$Volume,
              N = N)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $OpenBUGS$. El modelo se estima mediante la función $bugs$ del paquete $R2OpenBUGS$, esta función toma el código de $OpenBUGS$ del archivo $normal_trees.txt$ y declara el número de iteraciones en 10.000 y el número de cadenas en 4. En cuanto a los parámetros a estimar se tiene que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
modelo_normal_ob =R2OpenBUGS::bugs(data,
                                   inits = NULL,
                                   model.file="normal_trees.txt",
                                   parameters= c("beta_0", "beta_1", "beta_2",
                                                 "gamma_0", "gamma_1", "gamma_2",
                                                 "y_est"),
                                   n.chains = 4,
                                   n.iter=10000
)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $Stan$. El modelo se estima mediante la función $stan$ de la distribución $rstan$. Esta función toma el código de $stan$ del archivo $normal_trees.stan$ y declara el número de iteraciones en 100.000 y el número de cadenas en 4. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud log_lik y de la distribución predictiva de la variable dependiente y_test. Para facilitar la convergencia en los parámetros de control al parámetro adapt$_deltaseleasignaelvalorde$0,9.

```
modelo_normal = rstan::stan(data=data,
```

```
                    file = "normal_trees.stan",
                    pars= c("beta_0", "beta_1", "beta_2",
                            "gamma_0", "gamma_1", "gamma_2",
                            "log_lik", "y_test"),
                    chains = 4,
                    iter=100000,
                    control=list(adapt_delta = 0.9)
)
```

La tabla **2-9** muestra el resumen de la estimación de este modelo en Stan, la cual, muestra que los resultados son coherentes con los presentados por [Cepeda & Gamerman, 2000] y que se tienen desviaciones estandar del estimador de la media de la distribución a posteriori mucho menores, no obtante, el intervalo de credibilidad del 95 % que arroja la estimación también muestra que el cero está contenido en el intervalo para el parámetro γ_2. De la misma manera, la tabla **2-10** muestra una correlación importante entre las muestras de la distribución a posteriori de los parámetros β_0 y β_1 y γ_0 y γ_1.

Tabla 2-9.: Resumen de Estimación Modelo Normal para Datos de Arboles

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
beta_0	-0.09	0.00	0.15	-0.39	-0.19	-0.09	0.00	0.20	95608.35	1.00
beta_1	0.01	0.00	0.00	0.01	0.01	0.01	0.02	0.02	87356.23	1.00
beta_2	0.15	0.00	0.01	0.14	0.15	0.15	0.15	0.16	122959.73	1.00
gamma_0	-6.60	0.01	1.98	-10.49	-7.92	-6.59	-5.27	-2.71	94223.84	1.00
gamma_1	0.05	0.00	0.03	-0.00	0.03	0.05	0.07	0.11	90568.75	1.00
gamma_2	0.02	0.00	0.07	-0.11	-0.03	0.02	0.06	0.16	132371.29	1.00
lp__	34.24	0.01	1.82	29.82	33.27	34.58	35.58	36.74	67734.84	1.00

Tabla 2-10.: Correlación a posteriori Modelo Normal Datos Arboles

	beta_0	beta_1	beta_2	gamma_0	gamma_1	gamma_2
beta_0	1					
beta_1	-0.92	1				
beta_2	0.179	-0.542	1			
gamma_0	-0.011	-0.036	0.122	1		
gamma_1	-0.04	0.081	-0.123	-0.907	1	
gamma_2	0.12	-0.114	0.024	-0.03	-0.389	1

Como los intervalos de credibilidad para los parámetros β_0 y γ_2 contienen el valor cero, se estima un modelo excluyendo estos parámetros, este modelo es llamado modelo 2. Los resultados del modelo 2 de esta estimación están contenidos en la tabla **2-11**. La estimación

de este modelo muestra que las dos variables tienen un efecto positivo sobre la media de la variable volumen y la variable altura tiene un efecto positivo sobre la varianza del volumen.

Tabla 2-11.: Resumen de Estimación Modelo Normal para Datos de Arboles Modelo 2

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
beta_1	0.01	0.00	0.00	0.01	0.01	0.01	0.01	0.02	7269.89	1.00
beta_2	0.15	0.00	0.01	0.14	0.15	0.15	0.15	0.16	7175.04	1.00
gamma_0	-6.58	0.03	1.95	-10.45	-7.88	-6.58	-5.26	-2.76	5434.29	1.00
gamma_1	0.05	0.00	0.03	0.00	0.04	0.05	0.07	0.10	5438.85	1.00
lp__	35.00	0.02	1.45	31.31	34.30	35.32	36.06	36.80	5808.55	1.00

2.4.5. Aplicación sobre datos de calificaciones de Lectura

En esta sección se presenta una aplicación con datos de calificaciones de lectura recolectados por [Pammer & Kevan, 2007] y utilizados por [Smithson & Verkuilen, 2006, Cepeda & Cifuentes, 2017]. Estos datos muestran la calificación obtenida en el test de lectura, la presencia o no de dislexia y el coeficiente intelectual no verbal estandarizado para un conjunto de 44 niños, la calificación obtenida por cada uno de los niõs fue estandarizada por [Smithson & Verkuilen, 2006] para que estuviera contenida en el intervalo $[0, 1]$. Este conjunto de datos fue utilizado por [Cepeda & Cifuentes, 2017] para ajustar un modelo de regresión beta parametrizado como en la ecuación 2-9 con las siguientes funciones para la media y el parámetro de dispersi'on.

$$logit(\mu_i) = \beta_0 + \beta_1 DIS_i + \beta_2 IQ_i + \beta_3 INT_i \tag{2-10}$$

$$log(\phi[i]) = \gamma_0 + \gamma_1 DIS_i + \gamma_2 IQ_i \tag{2-11}$$

Donde la variable $INT_i = IQ_i * DIS_i$. Para llevar a cabo la estimación de este modelo el autor utilizó el paquete betareg R introducido por [Zeileis et al., 2016] y también estimación bayesiana. En este trabajo, la estimación bayesiana se llevará a cabo teniendo en cuenta la misma función de verosimilitud y como distribuciones a priori del conjunto de parámetros γ_i y β_i se seleccionaron distribuciones normales $N(0, 10^k)$, en este documento se reportan los resultados para $k = 4$.

Parametrización del modelo en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud normal y se definen los predictores lineales con las variables iq, dys, $inter$. En este caso, está el predictor lineal de la media y la precision, sin embargo, la parametrización de la distribución

beta en OpenBUGS no se hace con los parámetros de media y precisión, razón por la cual, se hace una tranformación sobre el parámetro de varianza en el código.

Posteriormente, se asignan las distribuciones a priori normales par los parámetros γ y β. Finalmente, se declara la variable y_est la cual es la predicción de la variable dependiente.

```
model
{
    for(i in 1:N){
        score[i] ~ dbeta(a[i],b[i]);
        exp_mu[i] <- exp(beta_0+ beta_1*dys[i] + beta_2*iq[i] +
                     beta_3*inter[i]);
        mu[i] <- exp_mu[i]/(1+exp_mu[i]);
        sigma_2[i] <-exp(gamma_0+gamma_1*dys[i] + gamma_2*iq[i]);

        b[i] <- sigma_2[i]-sigma_2[i]*mu[i];
        a[i] <- sigma_2[i]-b[i];
        }
    beta_0 ~ dnorm(0,0.0001)
    beta_1 ~ dnorm(0,0.0001)
    beta_2 ~ dnorm(0,0.0001)
    beta_3 ~ dnorm(0,0.0001)
    gamma_0 ~ dnorm(0,0.0001)
    gamma_1 ~ dnorm(0,0.0001)
    gamma_2 ~ dnorm(0,0.0001)
    for(i in 1:N){
        y_est[i] ~ dbeta(a[i],b[i])
    }
}
```

Parametrización del modelo en Stan

Para parametrizar el modelo en Stan, en la sección de datos se declaran cuatro variables *iq*, *dys*, *inter* y la variable dependiente *score*, cada una como un vector de tamaño N que es la cantidad de observaciones del conjunto de datos. Los parámetros a estimar son el conjunto de parámetros β y γ los cuales son los predictores linerales del parámetro de media y de precisión respectivamente.

Posterior a la declaración de parámetros se lleva a cabo la transformación de los mismos, en este caso, se lleva a cabo el cálculo de los parámetros de media y dispersión. En este caso, al igual que en el ejemplo de simulación las transformaciones sobre los predictores lineales son

mayores dada la parametrización de la distribución beta en Stan.

En la siguiente sección se lleva a cabo la declaración del modelo, en la cual se asignan las distribuciones prior para cada uno de los parámetros β y γ y se asigna a cada una de las observaciones de la variable *score* la función de log verosimilitud de la distri bución beta.

Por último, se lleva a cabo el cálculo de la log verosimilitud y la predicción de la variable dependiente en la sección de cantidades generadas.

```
data{
    int<lower=0> N;
    real score[N];
    real iq[N];
    real dys[N];
    real inter[N];
}
parameters {
    real beta_0;
    real beta_1;
    real beta_2;
    real beta_3;
    real gamma_0;
    real gamma_1;
    real gamma_2;
}
transformed parameters{
   real mu[N];
   real exp_mu[N];
   real phi[N];
   real alpha[N];
   real omega[N];
   for (i in 1:N){
       exp_mu[i] = exp(beta_0+ beta_1*dys[i] + beta_2*iq[i] +  beta_3*inter[i]);
       mu[i] = exp_mu[i]/(1+exp_mu[i]);
       phi[i] = exp(gamma_0+gamma_1*dys[i] + gamma_2*iq[i]);
       omega[i] = phi[i]-phi[i]*mu[i];
       alpha[i] = phi[i]-omega[i];
       }
}
model{
    beta_0 ~ normal(0,100);
```

```
        beta_1 ~ normal(0,100);
        beta_2 ~ normal(0,100);
        beta_3 ~ normal(0,100);
        gamma_0 ~ normal(0,100);
        gamma_1 ~ normal(0,100);
        gamma_2 ~ normal(0,100);
        for (i in 1:N){
            score[i] ~ beta_lpdf(alpha[i], omega[i]);
                }
    }
    generated quantities{
      vector[N] log_lik;
      vector[N] y_est;
      for (i in 1:N){
              log_lik[i] = beta_lpdf(score[i] | alpha[i], omega[i]);
              y_est[i] = beta_rng(alpha[i], omega[i]);
              }
    }
```

Ejecución del modelo desde R

El siguiente fragmento de código de muestra el tratamiento de datos en *R*.

```
library('rstan')
library('betareg')
library('R2OpenBUGS')

options(mc.cores = parallel::detectCores())
rstan::rstan_options(auto_write = TRUE)

rm(list = ls())
data("ReadingSkills")


df_reading <- ReadingSkills

rm(ReadingSkills)
df_reading$dyslexia <- ifelse(df_reading$dyslexia=='yes',1,0)
df_reading$inter <- df_reading$dyslexia*df_reading$iq
N <- nrow(df_reading)
```

```
data = list(score = df_reading$accuracy,
            iq = df_reading$iq,
            dys = df_reading$dys,
            inter = df_reading$inter,
            N = N)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $OpenBUGS$. El modelo se estima mediante la función $bugs$ del paquete $R2OpenBUGS$, esta función toma el código de $OpenBUGS$ del archivo $beta\_reading\_acc.txt$ y declara el número de iteraciones en 10.000 y el número de cadenas en 2. En cuanto a los parámetros a estimar se tiene que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
modelo_beta_ob =R2OpenBUGS::bugs(data_x,
                                 model.file="beta_reading_acc.txt",
                                 parameters = c("beta_0", "beta_1","beta_2",
                                 "beta_3", "gamma_0", "gamma_1",
                                 "gamma_2","y_test"),
                                n.chains = 2,
                                n.iter=10000
)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $Stan$. El modelo se estima mediante la función $stan$ de la distribución $rstan$. Esta función toma el código de $stan$ del archivo $beta\_reading\_acc.stan$ y declara el número de iteraciones en 10.000 y el número de cadenas en 1. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud log_lik y de la distribución predictiva de la variable dependiente y_est. Para facilitar la convergencia en los parámetros de control al parámetro adapt$_d$$eltaseleasignaelvalorde$0,95.

```
    modelo_beta = rstan::stan(data=data_x,
                              file = "beta_reading_acc.stan",
                              pars= c("beta_0", "beta_1", "beta_2",
                                      "beta_3", "gamma_0", "gamma_1",
                                      "gamma_2", "log_lik", "y_est"),
                              chains = 1,
                              iter=10000,
                              control=list(adapt_delta = 0.95)
    )
```

Los resultados de la estimación de este modelo se observan en al tabla **2-12** mientras que la correlación entre las muestras de la distribución a posteriori de los parámetros está contenida en la tabla **2-13**. Comparativamente, los resultados obtenidos con muy similares a los de [Cepeda & Cifuentes, 2017]. En cuanto a la interpretabilidad, el modelo para la media para aquellos estudiantes que no tienen dislexia es $logit(\mu_i) = 1{,}88 + 0{,}97IQ_i$, es decir, la variable IQ tiene un efecto positivo sobre la calificación promedio de los estudiantes que no tienen dislexia. Ahora, para los estudiantes con dislexia ($DIS_i = 1$) el modelo de la media es $logit(\mu_i) = 1{,}88 - 1{,}49 + 0{,}97IQ_i - 1{,}06INT_i$ lo cual dado que para este grupo de usuarios $DIS_i = 1$ y que $INT_i = DIS_iIQ_i$ es igual a $logit(\mu_i) = 0{,}39 - 0{,}97IQ_i - 1{,}06IQ_i = 0{,}39 - 0{,}09IQ_i$, lo cual muestra que para los estudiantes con dislexia la variable IQ tiene un efecto negativo en la calificación promedio de lectura. En cuanto al modelo del parámetro de precisión ϕ se aprecia que la variable IQ tiene efectos posistivos en este tanto para los dos grupos de estudiantes y dado un nivel de IQ la dispersión de los datos es menor (la presición es mayor) en el grupo con dislexia.

Tabla 2-12.: Resumen de Estimación Modelo Beta para Datos de Lectura

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
beta_0	1.88	0.00	0.30	1.26	1.68	1.88	2.08	2.45	58108.42	1.00
beta_1	-1.49	0.00	0.31	-2.08	-1.70	-1.50	-1.29	-0.87	58453.52	1.00
beta_2	0.97	0.00	0.35	0.27	0.74	0.98	1.21	1.63	49920.46	1.00
beta_3	-1.06	0.00	0.36	-1.73	-1.30	-1.07	-0.82	-0.33	49959.48	1.00
gamma_0	1.52	0.00	0.34	0.83	1.30	1.53	1.75	2.14	66791.31	1.00
gamma_1	3.23	0.00	0.61	2.00	2.83	3.25	3.65	4.38	56208.93	1.00
gamma_2	1.05	0.00	0.45	0.14	0.74	1.06	1.36	1.91	53740.18	1.00
lp__	62.20	0.01	1.96	57.50	61.13	62.54	63.65	65.00	65545.55	1.00

Tabla 2-13.: Correlación a posteriori Modelo Beta Datos Lectura

	beta_0	beta_1	beta_2	beta_3	gamma_0	gamma_1	gamma_2
beta_0	1						
beta_1	-0.984	1					
beta_2	-0.564	0.553	1				
beta_3	0.553	-0.544	-0.986	1			
gamma_0	0.601	-0.591	-0.294	0.293	1		
gamma_1	-0.466	0.46	0.499	-0.502	-0.707	1	
gamma_2	-0.311	0.303	0.753	-0.752	-0.361	0.64	1

2.4.6. Aplicación sobre datos de crecimiento de Drosophyla Melanogaster

En esta sección se presenta una aplicación con datos de crecimiento de moscas de la fruta en relación a la temperatura, estos datos ya fueron abordados por [McCullagh, 1983, Cepeda et al., 2016]. El conjunto de datos completo es un conjunto de 2 variables, la temperatura y la duración en horas en cada uno de los estados, a su vez, hay 4 estados: el estado embrionario, el de larva-huevo, el de larva y el de crisálida. En este tarbajo, al igual que en el trabajo de [Cepeda et al., 2016], se considera solamente el estado embrionario.

Para este conjunto de datos se propone una regresión gamma con la parametrización propuesta en la ecuación 2-7 donde los parámetros se modelan de la siguiente manera:

$$ln(\mu_i) = \beta_0 + \beta_1 T_i + \beta_2 \frac{1}{T_i} \tag{2-12}$$

$$ln(\kappa_i) = \gamma_0 + \gamma_1 T_i + \gamma_2 \frac{1}{T_i} \tag{2-13}$$

En este trabajo, la estimación bayesiana se llevará a cabo teniendo en cuenta la función de verosimilitud gamma y como distribuciones a priori del conjunto de parámetros γ_i y β_i se seleccionaron distribuciones normales $N(0, 10^k)$, para este caso se reportan los resultados para $k = 4$.

Parametrización del modelo en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud gamma y se definen los predictores lineales con las variables *temp*, *temp_inv*. En este caso, está el predictor lineal de la media y la precision, sin embargo, la parametrización de la distribución gamma en OpenBUGS no se hace con los parámetros de media y precisión razón por la cual, se hacen necesarias algunas transformaciones sobre estos parámetros.

Posteriormente, se asignan las distribuciones a priori normales par los parámetros γ y β. Finalmente, se declara la variable *y_est* la cual es la predicción de la variable dependiente.

```
model
{
    for (i in 1:N)
    {
    alpha[i] <- exp(gamma_0 + gamma_1*temp[i] + gamma_2*temp_inv[i]);
    media[i] <- exp(beta_0 + beta_1*temp[i] + beta_2*temp_inv[i]);
```

```
        mu[i] <- alpha[i]/media[i];
        dur[i] ~ dgamma(alpha[i],mu[i]);

        }
        beta_0 ~ dnorm(0,0.0001);
        beta_1 ~ dnorm(0,0.0001);
        beta_2 ~ dnorm(0,0.0001);
        gamma_0 ~ dnorm(0,0.0001);
        gamma_1 ~ dnorm(0,0.0001);
        gamma_2 ~ dnorm(0,0.0001);


        for (i in 1:N)
        {
            y_est[i] ~ dgamma(alpha[i],mu[i]);
        }
    }
```

Parametrización del modelo en Stan

Para parametrizar el modelo en Stan, en la sección de datos se declaran cuatro variables *temp*, *temp_inv* y la variable dependiente *dur*, cada una como un vector de tamaño N que es la cantidad de observaciones del conjunto de datos. Los parámetros a estimar son el conjunto de parámetros β y γ los cuales son los predictores linerales del parámetro de media y de precisión respectivamente.

Posterior a la declaración de parámetros se lleva a cabo la transformación de los mismos, en este caso, se lleva a cabo el cálculo de los parámetros de media y dispersión. En este caso, al igual que en el ejemplo de simulación las transformaciones sobre los predictores lineales son mayores dada la parametrización de la distribución gamma en Stan.

En la siguiente sección se lleva a cabo la declaración del modelo, en la cual se asignan las distribuciones prior para cada uno de los parámetros β y γ y se asigna a cada una de las observaciones de la variable *dur* la función de log verosimilitud de la distri bución gamma.

Por último, se lleva a cabo el cálculo de la log verosimilitud y la predicción de la variable dependiente en la sección de cantidades generadas.

```
data{
int<lower=0> N;
real temp[N];
real dur[N];
real temp_inv[N];
}
parameters {
    real beta_0;
    real beta_1;
    real beta_2;
    real gamma_0;
    real gamma_1;
    real gamma_2;
}
transformed parameters{
    real mu[N];
    real alpha[N];
    real theta[N];

    for (i in 1:N){
         alpha[i] = exp(gamma_0 + gamma_1*temp[i] +    gamma_2*temp_inv[i]);
         mu[i] = exp(beta_0 + beta_1*temp[i] +    beta_2*temp_inv[i]);
         theta[i] = alpha[i]/mu[i];
    }
}
model{
    beta_0 ~ normal(0,100);
    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
    gamma_0 ~ normal(0,100);
    gamma_1 ~ normal(0,100);
    gamma_2 ~ normal(0,100);

    for (i in 1:N){
        dur[i] ~ gamma_lpdf(alpha[i],theta[i]);
            }

}
generated quantities{
  vector[N] log_lik;
```

```
      vector[N] y_est;
      for (i in 1:N){
          log_lik[i] = gamma_lpdf(dur[i] | alpha[i],theta[i]);
          y_est[i] = gamma_rng(alpha[i],theta[i]);
          }
    }
```

Ejecución del modelo desde R

El siguiente fragmento de código de muestra el tratamiento de datos en R.

```
library('rstan')
library('loo')
library(R2OpenBUGS)
options(mc.cores = parallel::detectCores())
rstan::rstan_options(auto_write = TRUE)

rm(list = ls())


delta <-  58.64
temp <- c(14.95, 16.16, 16.19, 17.15,
       18.20, 19.08, 20.07, 22.14,
       23.27, 24.09, 24.81, 24.84,
       25.06, 25.06, 25.08, 26.92,
       27.68, 28.89, 28.96, 29.00,
       30.05, 30.80, 32.00)

dur <- c(67.5, 57.1, 56.0, 48.4,
           41.2, 37.80, 33.33, 26.50,
           24.24, 22.44, 21.13, 21.05,
           20.39, 20.41, 19.45, 18.77,
           17.79, 17.38, 17.26, 17.18,
           16.81, 16.97, 18.20)

temp_inv <- 1/(temp)

N <- length(dur)

data = list(temp = temp,
```

```
dur = dur,
temp_inv = temp_inv,
N = N)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $OpenBUGS$. El modelo se estima mediante la función $bugs$ del paquete $R2OpenBUGS$, esta función toma el código de $OpenBUGS$ del archivo $gamma\_flies.txt$ y declara el número de iteraciones en 10.000 y el número de cadenas en 1. En cuanto a los parámetros a estimar se tiene que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
modelo_gamma_ob =R2OpenBUGS::bugs(data_x,
                                  inits = NULL,
                                  model.file="gamma_flies.txt",
                                  parameters= c("beta_0","beta_1","beta_2", "gamma_0","ga
                                  "y_est"),
                                  n.chains = 1,
                                  n.iter=10000
)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $Stan$. El modelo se estima mediante la función $stan$ de la distribución $rstan$. Esta función toma el código de $stan$ del archivo $gamma_flies.stan$ y declara el número de iteraciones en 10.000 y el número de cadenas en 1. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud log_lik y de la distribución predictiva de la variable dependiente y_est. Para facilitar la convergencia en los parámetros de control al parámetro adapt$_delta$ se le asigna el valor de 0,95 y al parámetro max$_treedepth$ el valor de 15.

```
modelo_gamma = rstan::stan(data=data,
                           file = "gamma_flies.stan",
                           pars= c("beta_0", "beta_1", "beta_2",
                                   "gamma_0", "gamma_1", "gamma_2",
                                   "log_lik", "y_est"),
                           chains = 1,
                           iter=10000,
                           control=list(adapt_delta = 0.95, max_treedepth = 15)
)
```

Los resultados de la estimación de este modelo se presentan en la tabla **2-14** y la correlación de la muestra de la distribución a posteriori está contenida en la tabla **2-15**, en general,

salvo por el parámetro β_2, las estimaciones son muy similares a las de [Cepeda et al., 2016]. El modelo de la media muestra que el componente lineal y el componente racional tienen un efecto importante en el tiempo de permanencia en el estado embrionario.

Tabla 2-14.: Resumen de Estimación Modelo Gamma para Datos de Drosophyla

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
beta_0	3.11	0.00	0.07	2.97	3.07	3.11	3.16	3.26	6805.53	1.00
beta_1	-0.27	0.00	0.01	-0.28	-0.27	-0.27	-0.27	-0.26	7050.18	1.00
beta_2	-224.92	0.08	6.96	-238.61	-229.51	-224.92	-220.35	-211.09	6828.67	1.00
gamma_0	8.92	0.02	2.15	4.60	7.51	8.95	10.41	13.04	9843.46	1.00
gamma_1	-0.06	0.00	0.12	-0.28	-0.13	-0.06	0.02	0.17	9814.49	1.00
gamma_2	-15.25	1.02	99.72	-213.18	-81.40	-14.73	51.57	177.81	9489.93	1.00

Tabla 2-15.: Correlación a posteriori Modelo Gamma Datos Drosophyla

	beta_0	beta_1	beta_2	gamma_0	gamma_1	gamma_2
beta_0	1					
beta_1	0.921	1				
beta_2	0.966	0.99	1			
gamma_0	-0.029	-0.036	-0.034	1		
gamma_1	0.067	0.079	0.076	-0.305	1	
gamma_2	0.046	0.052	0.05	0.429	0.723	1

3. Modelamiento de media y dispersión con Datos de Conteo

El modelamiento de datos aquellos que tienen naturaleza de conteo tienen una importante especial dado que muchos fenómenos tienen esta naturaleza, por ejemplo, la cantidad de personas que llegan a una fila o a un punto de atención, la cantidad de personas contagiadas en una epidemia, la cantidad de elementos defectuosos en un sistema de producción, la cantidad de artículos defectuosos en una determinada cantidad de artículos en una inspección de calidad, entre otros. En lo referente al modelameiento de este tipo de datos, es muy usual encontrar propuestas a través de la distribución de poisson y la distribución binomial, no obstante, existen problemas con este tipo de enfoque cuando los datos presentan discrepancias entre la distribución teórica y la distribución empírica [Cepeda & Cifuentes, 2017].

Por lo presentado anteriormente, diversos trabajos se han planteado con el fin de lograr un modelamiento más apropiado en estos contextos, incluso, desde el mismo plantemiento de los GLM, [Wedderburn, 1974] propuso estimaciones por quasi verosimilitud para casos en que se presentan sobre o sub dispersión.[Williams, 1982] y [Breslow, 1984] estudiaron el fenómeno de sobredispersión para los casos binomial y poisson respectivamente e incorporaron componentes de sobredispersión en las estimaciones de máxima verosimilitud.

En el contexto de los DGLM, [Cepeda & Cifuentes, 2017] propone un modelo beta binomial y un modelo binomial negativo para datos de conteo con estructuras de regresión para los dos parámetros de cada distribución y los incorpora en los DGLM, además, presenta metodologás bayesiana y clásica para las estimación de estos modelos [1]. En el ámbito aplicado [Cepeda et al., 2012b] utiliza el modelo beta binomial para una aplicación en la cual se modelan datos de mortalidad infantil en colombia. [Hauer, 2001] utiliza un modelo binomial negativo en contraposición a un modelo de poisson para modelar datos de cantidades de accidentes de tránsito y [Pham et al., 2010] utiliza un modelo beta binomial para el modelamiento de conteos en casos de detección de cáncer.

[1] En este caso, dado que estas distribuciones no se encuentran en la familia exponencial se puede inferir que el termino *generalizados* hace referencia a una familia biparamétrica.

3.1. Modelamiento de Media y Dispersión para datos de conteo

3.1.1. Modelo de Regresión Beta Binomial

La distribución Beta Binomial puede ser entendida como una distribución que modela un fenómeno Binomial, en el cual, el parámetro p que hace referencia a la probabilidad de éxito en el experimento Benoulli no es fijo sino que es una variable aleatoria con distribución Beta. Es decir, la distribución Beta binomial puede ser entendida como la distribución de una variable aleatoria X que condicionada a un parámetro p tiene distribución binomial, cuando dicho parámetro tiene distribución beta. Esta distribución tiene la ventaja de permitir un mejor modelamiento de los datos cuando existen ciertos tipos de discrepancias entre la varianza teórica y la varianza empírica de los mismos [Cepeda & Cifuentes, 2017].

Distibución Beta Binomial

Una variable aleatoria y con distribución beta binomial $y \sim BB(\mu, \phi)$ tiene la siguiente función de densidad de probabilidad:

$$f(y) = \binom{m}{y} \frac{\beta(y + \mu\phi, m - y + \phi(1 - \mu))}{\beta(\mu\phi, \phi(1 - \mu))} \tag{3-1}$$

De donde se tiene que $E(y) = m\mu$ y $V(y) = m\mu(1 - \mu)(\frac{\phi+m}{\phi+1})$.

Modelo de Regresión Beta Binomial

Considerese que y_i para $i = 1, ..., n$ una muestra de las variables aleatorias de interés y $x_i = (x_{i,1}, ..., x_{i,s})$ y $z_i = (z_{i,1}, ..., z_{i,k})$ para $i = 1, ..., n$ dos vectores de covariables y $\beta = (\beta_1, ..., \beta_s)$ y $\gamma = (\gamma_1, ..., \gamma_k)$ dos vectores de parámetros de regresión, el modelo de datos de conteo con sobredipersión basado en la regresión Beta Binomial de la ecuación 3-1 queda defindo de la siguiente manera:

- Una componente aleatoria dada por las variables $y_i \sim BB(\mu_i, \phi_i)$ distribuidas de manera independiente.

- Dos componentes sistemáticas $\eta_{1,i} = X_i^T \beta$ y $\eta_{2,i} = Z_i^T \gamma$.

- Dos funciones de enlace $h(.)$ y $g(.)$, de manera que $h(\mu_i) := \eta_{1,i}$ y $g(\phi_i) := \eta_{2,i}$. en este caso, la función $h(\mu_i)$ debe garantizar que tanto μ como $g(\phi_i)$ tomen valores en el

intervalo $(0, 1)$, algunas de las opciones más comunes son las funciones logit, probit y log-log complementario [McCullagh, 1983].

3.1.2. Modelo de Regresión Binomial Negativa

Distribución Binomial Negativa

Así como la distribución beta binomial puede ser entendida como una distribución compuesta entre una distribución binomial y una distribución beta, la distribución Binomial negativa puede verse como una distribución compuesta, en este caso, de una distribución poisson y una distribución gamma. De manera más explícita, se tiene que una variable y condicionada por λ tiene una distribución poisson de parámetro λ donde λ es a su vez una variable aleatoria con distribución gamma.

De manera más clara, si se toma la parametrización de la función gamma como en la ecuación 2-7, es decir, $\lambda \sim \Gamma(\mu, \kappa)$ donde μ es la media de la variable y κ es un parámetro de forma o como se mostró un parámetro de dispersión. Con esta parametrización de la función gamma se tiene que la función de densidad de y cuando y distribuye binomial negativa, $y \sim NB(\mu, \kappa)$, viene dada por:

$$f(y) = (\frac{\kappa}{\mu})^{\kappa} \frac{\Gamma(y+\kappa)}{y!\Gamma(\kappa)(1+\frac{\mu}{\kappa})^{(y+\kappa)}} \tag{3-2}$$

Donde $E(y) = \mu$ y $Var(y) = \frac{\mu(\mu+\kappa)}{\kappa}$. Otra opción para parametrizar esta distribución es expresarla en términos de su media y su varianza, lo cual es posible si se asume que el parámetro λ tiene una distribución gamma con media μ y varianza σ^2, haciendo que $\kappa = \mu^2/(\sigma^2 - \mu)$ la función de densidad de y cuando y tiene distribución binomial negativa, $y \sim NB(\mu, \sigma^2)$ viene dada por:

$$f(y) = (\frac{\mu}{\sigma^2})^{\frac{\mu^2}{\sigma^2-\mu}} \frac{\Gamma(y+\frac{\mu^2}{\sigma^2-\mu})}{y!\Gamma(\frac{\mu^2}{\sigma^2-\mu})(1+\frac{\sigma^2}{(\sigma^2-\mu)})^y} \tag{3-3}$$

Con la anterior parametrización la variable y tiene media μ y varianza σ^2.

Respecto a la interpretación, considerese una sucesión de experimentos bernoulli independientes e idénticos, es decir, con igual probabilidad de éxito, una variable aleatoria con esta distribución es el número de éxitos hasta que ocurra un número determinado de fracasos. A manera de ejemplo, el lanzamiento de un dado en el cual se considera un fracaso obtener tres puntos y se quiere obtener tres puntos por ejemplo l veces, en este caso $r = l$, la variable

X definida como la cantidad de éxitos hasta que el número tres se repitió l veces tiene una distribución binomial negativa y es equivalente al total de intentos n menos el numero de fracasos, en este caso $r = l$.

De la ecuación 3-2 es posible obtener la forma más conocida de la distribución binomial negativa, tal como lo ilustra [Cepeda & Cifuentes, 2017], haciendo que $p = \frac{\mu}{\mu+\kappa}$, se tiene que $\mu = \frac{\alpha p}{1-p}$ donde p es la probabilidad de éxito en cada intento, con esta reparametrización la función de probabilidad está dada por la ecuación 3-4.

$$f(y) = \frac{\Gamma(y+\kappa)}{\Gamma(\kappa)}\frac{1}{y!}(1-p)^{\kappa}p^{y} \tag{3-4}$$

Un aspecto importante a tener en cuenta radica en el hecho que la anterior presentación de la distribución binomial negativa la variable que tiene esta distribución es el número de éxitos hasta completar determinada cantidad de fracasos. No obstante, tal como se puede ver en [DeGroot & Schervish, 2012], esta distribución también puede verse de dos maneras alternativas: primera, puede verse como el número de intentos hasta obtener determinada cantidad de éxitos o fracasos en una sucesión de eventos Bernoulli. Segundo, también corresponde a una variable aleatoria que muestra la cantidad de fracasos hasta obtener determinada cantidad de éxitos en una sucesión de experimentos bernoulli independientes, esta última definición es muy popular y es la implementación que se tiene en R, Stan y OpenBUGS, por lo tanto, es importante presentarla en esta sección. La ecuación 3-5 corresponde con esta forma de ver la variable aleatoria binomial Negativa $Y \sim NB(\kappa, p)$.

$$f(y) = \frac{1}{y!}\frac{\Gamma(y+\kappa)}{\Gamma(\kappa)}(1-p)^{y}p^{\kappa} \tag{3-5}$$

Una variable aleatoria con esta función de probabilidad tiene valor esperado y varianza $E(y) = \mu = \frac{(1-p)\kappa}{p}$ $V(y) = \frac{(1-p)\kappa}{p^2}$, reparametrizando la función de probabilidad en términos de la media se tiene la función de probabilidad de la ecuación 3-6, es decir, una variable aleatoria y con distribución binomial negativa como la de dicha ecuación tiene valor esperado $E(y) = \mu$ y $Var(y) = \frac{\mu^2}{\kappa} + \mu$, donde al igual que en algunas distribuciones presentadas anteriormente, dado un valor de la media μ, el parámetro κ guarda una relación negativa con la varianza, por lo tanto, tiene una noción de ser parámetro de precisión [Cepeda & Cifuentes, 2017].

$$f(y) = \frac{1}{y!}\frac{\Gamma(y+\kappa)}{\Gamma(\kappa)}(\frac{\mu}{\mu+\kappa})^{y}(\frac{\kappa}{\kappa+\mu})^{\kappa} \tag{3-6}$$

Modelo de Regresión Binomial Negativa

Considérese que y_i para $i = 1, ..., n$ una muestra de las variables aleatorias de interés y $x_i = (x_{i,1}, ..., x_{i,s})$ y $z_i = (z_{i,1}, ..., z_{i,k})$ para $i = 1, ..., n$ dos vectores de covariables y $\beta = (\beta_1, ..., \beta_s)$ y $\gamma = (\gamma_1, ..., \gamma_k)$ dos vectores de parámetros de regresión. Con la parametrización de la ecuación 3-6 el modelo de regresión para datos de conteo con sobredispersión basado en la distribución Binomial Negativa queda definido de la siguiente manera:

- Una componente aleatoria dada por las variables $y_i \sim NB(\mu_i, \kappa_i)$ distribuidas de manera independiente.

- Dos componentes sistemáticas $\eta_{1,i} = X_i^T\beta$ y $\eta_{2,i} = Z_i^T\gamma$.

- Dos funciones de enlace $h(.)$ y $g(.)$, de manera que $h(\mu_i) := \eta_{1,i}$ y $g(\kappa_i) := \eta_{2,i}$. en este caso, las funciones de enlace $h(\mu_i)$ y $g(\kappa_i)$ deben garantizar que μ y κ tomen valores positivos, por lo cual es muy usual utilizar el enlace logarítmico.

Para la reparametrización $NB(\mu, \sigma^2)$ el modelo se puede plantear de la misma manera salvo que las funciones de enlace ya no se habla de $g(\kappa)$ sino de $g(\sigma^2)$.

3.2. Modelamiento para datos de conteo utilizando OpenBUGS y Stan

3.2.1. Programación del modelo en Stan

Tal como se presentó en la sección 3.2.1, el siguiente fragmento de código muestra la parametrización general de un modelo en Stan. En este fragmento se aprecian las secciones del código para la programación de este modelo en Stan. La única diferencia a resaltar con lo espuesto anteriormente radica en las funciones de verosimilitud a utilizar en este tipo de modelos.

En la sección de modelos y en la sección de cantidades generadas se ebe usar las funciones que hacen referencia a la función beta binomial y binomial negativa respectivamente, las cuales en este caso son las funciones *beta_binomial_lpmf* y *beta_binomial_rng* para el caso de la distribución beta binomial y *neg_binomial_lpmf* y *neg_binomial_rng* para el caso de la distribución binomial negativa.

El siguiente fragmento de código de Stan es un pseudo código que muestra lo explicado anteriormente. Más adelante en la sección 3.3 se desarrollan simulaciones y aplicaciones específicas que permitirán ilustrar mejor la programación de estos modelos en Stan.

```
data{
    int<lower=0> N;
    real x[N];
    real y[N];
}
parameters {
    real beta;
    real gamma;
}
transformed parameters{
    real[N] g(media) = X * beta;
    real[N] h(dispersion) = X * gamma;

     }
model{
    beta ~ distribucion_prior();
    gamma ~ distribucion_prior();
    for (i in 1:N){
        y[i] ~ verosimilitud(media[i], dispersion[i]);
            }
        }
generated quantities{
    vector[N] y_test;
    vector[N] log_lik;
    for (i in 1:N){
        y_test[i] = generador_aleatorio(media[i], dispersion[i]);
        log_lik[i] = log_verosimilitud(y[i] | media[i], dispersion[i]);
    }
}
```

3.2.2. Programación del modelo en OpenBUGS

Tal como se presentó en la sección 3.2.2, el siguiente fragmento de código muestra la programación del modelo en OpenBUGS, en este se aprecia la misma estructura expuesta en esa sección. Para el caso del modelamiento de datos de conteo es necesario hacer la aclaración de las funciones de verosimilitud a utilizar las cuales en este caso correspoden a la función *debetabin* para el caso de la distribución beta binomial. En la sección 3.3se desarrollan simulaciones y aplicaciones específicas que permitirán ilustrar mejor la programación de estos modelos en OpenBUGS.

```
model
```

```
{
    for (i in 1:N)
    {
    g(media[i]) <- beta*x
    h(precision[i]) <- gamma*x
    y[i] ~ verosimilitud(media[i], precision[i])
    }
    beta ~ distribucion_prior(0,0.0001);
    gamma ~ distribucion_prior(0,0.0001);

    for (i in 1:N)
    {
    y_est[i] ~ generador_aleatorio(media[i], tau[i])
    }
}
```

3.3. Simulaciones y aplicaciones

En esta sección se llevarán a cabo estudios de simulación sobre el modelo de regresión Beta Binomial, lo anterior más allá de buscar una comparación en cuanto a eficiencia en la estimación utilizando Stan u OpenBUGS o pretende difundir el uso de esta técnica mediante herramientas de uso común y conocido como lo son Stan, R y OpenBUGS. Adicionalmente, se llevarán a cabo apliaciones sobre conjuntos de datos diversos con el fin de ilustrar el uso de esta técnica para problemas del mundo real, para esto, se ilustrará la forma como se parametrizaron los modelos en las dos herramientas y se comparaán los resultados obtenidos con los de otros autores.

3.3.1. Estudio de simulación para el Modelo Beta Binomial

Para este estudio de simulación se generaron 4 covariables denominadas x_1, x_2, x_3 y x_4 para un total de $n = 500$ observaciones y para el parámetro m, el número de intentos en el experimento se estableció un valor de 20, $m = 20$. La variable x_1 se generó de manera que $x_{i,1} = 1 \quad \forall i = 1, ..., n$, esto con el fin de generar un valor de intercepto. Las otras covariables se generaron de manera aleatoria de manera que $x_{2,i} \sim U(0, 30)$, $x_{3,1} \sim U(0, 15)$ y $x_{1,i} \sim U(10, 20)$ $\forall i = 1, ..., n$. La variable respuesta y_i se proviene de una distribución Beta Binomial con función de densidad dada por la ecuación 3-1, $y_i \sim (m, \mu_i, \gamma_i) \quad \forall i = 1, ..., n$, de manera que $logit(\mu_i) = -0{,}15+0{,}2x_{2,i}-0{,}3+x_{3,1}$ y $logit(\phi_i) = \gamma_1 x_{1,i}+\gamma_2 x_{2,i}+\gamma_3 x_{4,i}$ y $m = 20$.

En esta simulación están claramente identificados los tres componentes de un proceso generador de datos que corresponde con el expuesto en la sección 3.1.1. En primera medida, está

la componente aleatoria que en este caso corresponde las n observaciones de la variable y. Las componentes sistemáticas dadas por las estructuras de la media $\beta_1 + \beta_2 x_{2,i} - \beta_3 x_{3,1}$ y la estructura $\gamma_1 + \gamma_2 x_{2,i} + \gamma_3 x_{4,i}$ para el parámetro ϕ. Finalmente las funciones de enlace en este caso tanto para la media como para el parámetro ϕ.

La estimación se lleva a cabo asignando la distribución a priori normal a cada uno de los parámetros $\gamma_j, \beta_j \quad \forall j = 1, 2, 3$ con media 0 y varianza 10^4, es decir, la distribución a priori para los parámetros es $\gamma_j, \beta_j \sim N(0, 10^4) \quad \forall j = 1, 2, 3$.

Parametrización del modelo Beta Binomial en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud beta binomial y se definen los predictores lineales para los parámetros de media y precisión, además, se llevan a cabo las transformaciones necesarias sobre estos parámetros dado que la función Beta Binomial en OpenBUGS no se parametriza con respecto a la media y la precisión.

Posteriormente, se asignan las distribuciones a priori normales para los parámetros γ y β. En último lugar, se declara la variable y_est la cual es la predicción de la variable dependiente.

```
model
{
for (i in 1:N)
{
y[i] ~ dbetabin(alpha[i],beta[i])
mu[i] <- exp(beta_1*x_1[i]+ beta_2*x_2[i]+beta_3*x_3[i])/(1+exp(beta_1*x_1[i]+
beta_2*x_2[i]+beta_3*x_3[i]));
phi[i] <- exp(gamma_1*x_1[i] + gamma_2*x_2[i] + gamma_3*x_4[i]);
beta[i] <- phi[i]-phi[i]*mu[i];
alpha[i] <- phi[i]*beta[i];
}

beta_1 ~ dnorm(0,0.0001);
beta_2 ~ dnorm(0,0.0001);
beta_3 ~ dnorm(0,0.0001);
gamma_1 ~ dnorm(0,0.0001);
gamma_2 ~ dnorm(0,0.0001);
gamma_3 ~ dnorm(0,0.0001);

for (i in 1:N)
```

```
{
y_est[i] ~ dbetabin(alpha[i],beta[i]);
}
}
```

El siguiente código muestra la ejecución del modelo beta binomial en *OpenBUGS* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *bugs* del paquete *R2OpenBUGS*, esta función toma el código de *OpenBUGS* del archivo $beta_binomial.txt$ y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
data <- list(y=y, m=m, x_1=x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, N = N)

modelo_normal =R2OpenBUGS::bugs(data,
                 model.file="beta_binomial.txt",
                 parameters= c("beta_1", "beta_2", "beta_3",
                               "gamma_1", "gamma_2", "gamma_3",
                               "y_est"),
                 n.chains = 3,
                 n.iter=200000
)
```

Parametrización del modelo Beta Binomial en Stan

Para la configuración del modelo en Stan, en la primera seccińe se declaran los datos a utilizar en el modelo, en este caso corresponden a las variables $Y, X_1, X_2, X_3 y X_4$ y la cantidad de obervaciones N. Posterior a la declaración de los datos se declaran los parámetros a estimar, en este caso, corresponden al conjunto de parámetros del predictor lineal de la media y al predictor lineal del parámetro de precisión.

Posterior a las declaraciones anteriores, se llevan a cabo las transformaciones de parámetros necesarias, en este caso se declaran los parámetros de media y precisionón , además, se llevan a cabo las transformaciones necesarias sobre estos parámetros dado que la función Beta Binomial en OpenBUGS no se parametriza con respecto a la media y la precisión.

En la sección del modelo, se declaran las distribuciones a priori de los parámetros β y γ las cuales en este caso corresponden a distribuciones normales centradas en cero y con una

varianza de 10^6, posteriormente, se lleva a cabo el cálculo de los valores de *mu* y *sigma* y se asigna la verosimilitud normal a la variable y que en este caso corresponde a la distribución beta binomial.

Finalmente en la sección de cantidades generadas se lleva a cabo la predicción de los valores de la variable dependiente y y la log verosimilitud.

```
data{
    int<lower=0> N;
    int<lower=0> m;
    real x_1[N];
    real x_2[N];
    real x_3[N];
    real x_4[N];
    int<lower=0> y[N];
}
parameters {
    real beta_1;
    real beta_2;
    real beta_3;
    real gamma_1;
    real gamma_2;
    real gamma_3;
}
transformed parameters{
    real<lower=0> mu[N];
    real<lower=0> phi[N];
    real<lower=0> exp_mu[N];
    real<lower=0> exp_phi[N];
    real<lower=0> alpha[N];
    real<lower=0> beta[N];

    for (i in 1:N){
         exp_phi[i] = exp(gamma_1*x_1[i]+gamma_2*x_2[i] +gamma_3*x_4[i]);
         exp_mu[i] = exp(beta_1*x_1[i]+beta_2*x_2[i] + beta_3*x_3[i]);

         mu[i] = exp_mu[i]/(1+exp_mu[i]);
         phi[i] = exp_phi[i]/(1+exp_phi[i]);
         alpha[i] = mu[i]*phi[i];
```

```
          beta[i] = phi[i]-phi[i]*mu[i];
    }
}
model{

    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
    beta_3 ~ normal(0,100);
    gamma_1 ~ normal(0,100);
    gamma_2 ~ normal(0,100);
    gamma_3 ~ normal(0,100);
    for (i in 1:N){
        y[i] ~ beta_binomial_lpmf( m, alpha[i], beta[i]);
            }
}
generated quantities{
vector[N] y_test;
vector[N] log_lik;

for (i in 1:N){
    y_test[i] = beta_binomial_rng(m, alpha[i],beta[i]);
    log_lik[i] ~ beta_binomial_lpmf(y[i]| m, alpha[i], beta[i]);
    }
}
```

El siguiente código muestra la ejecución del beta binomial heteroscedástico en *stan* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *stan* de la distribución *rstan*. Esta función toma el código de *stan* del archivo *beta_binomial.stan* y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud *log_lik* y de la distribución predictiva de la variable dependiente *y_test*.

```
data <- list(y=y, m=m, x_1=x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, N = N)

modelo_beta_binomial = rstan::stan(data=data,
                           file = "beta_binomial.stan",
                           pars= c("beta_1", "beta_2","beta_3",
                           "gamma_1","gamma_2","gamma_3",
                           "y_test", "log_lik"),
                           chains = 3,
```

```
                    iter=100
                    )
```

La tabla **3-1** muestra el resumen del resultado de la estimación realizada para este modelo utilizando Stan, en la cual, se tiene una estimación a traves de la media a posteriori que es bastante cercana a los valores reales de los parámetros con que se simularon los datos, de igual manera, se puede ver que los valores reales del parámetro están contenidos en los intervalos de credibilidad comprendidos entre el 2,5 % y el 97,5 % de los valores muestreados de la distribución a posteriori.

Tabla 3-1.: Resumen de Estimación Modelo Beta Binomial

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
beta_1	0.33	0.00	0.20	-0.06	0.20	0.33	0.46	0.72	12091.22	1.00
beta_2	0.06	0.00	0.01	0.04	0.05	0.06	0.06	0.08	13991.46	1.00
beta_3	-0.05	0.00	0.02	-0.08	-0.06	-0.05	-0.03	-0.01	14320.38	1.00
gamma_1	-0.64	0.01	0.85	-2.29	-1.21	-0.65	-0.09	1.09	10902.80	1.00
gamma_2	-0.02	0.00	0.02	-0.06	-0.03	-0.02	-0.01	0.02	17046.69	1.00
gamma_3	0.05	0.00	0.05	-0.05	0.02	0.05	0.09	0.16	11343.27	1.00
lp__	-1035.32	0.02	1.78	-1039.67	-1036.27	-1034.98	-1034.03	-1032.87	8133.93	1.00

La tabla **3-2** muestra la estimación de la correlación entre los parámetros de la simulación, es importante resaltar que la correlación entre los parámetros de la estructura de la media, es decir, los parámetros β y los parámetros de la estructura de la varianza, es decir, los parámetros γ es muy baja. Así mismo, existe una fuerte correlación entre los interceptos y los otros parámetros.

	beta_1	beta_2	beta_3	gamma_1	gamma_2	gamma_3
beta_1	1					
beta_2	-0.555	1				
beta_3	-0.71	-0.043	1			
gamma_1	0.058	0.03	-0.072	1		
gamma_2	-0.133	0.179	0.071	-0.316	1	
gamma_3	-0.013	-0.069	0.052	-0.925	-0.016	1

Tabla 3-2.: Correlación a posteriori modelo Beta Binomial

3.3.2. Estudio de simulación para el Modelo Binomial Negativo

Para este estudio de simulación se generaron 4 covariables denominadas x_1, x_2, x_3 y x_4 para un total de $n = 500$ observaciones y para el parámetro m, el número de intentos en el experimento se estableció un valor de 20, $m = 20$. La variable x_1 se generó de manera que $x_{i,1} = 1 \quad \forall i = 1, ..., n$, esto con el fin de generar un valor de intercepto. Las otras covariables se generaron de manera aleatoria de manera que $x_{2,i} \sim U(0, 30)$, $x_{3,1} \sim U(0, 15)$ y $x_{1,i} \sim U(10, 20)$ $\forall i = 1, ..., n$. La variable respuesta y_i se proviene de una distribución Beta Binomial con función de densidad dada por la ecuación 3-1, $y_i \sim (m, \mu_i, \gamma_i) \quad \forall i = 1, ..., n$, de manera que $logit(\mu_i) = 0{,}2 + 0{,}1x_{2,i} + 0{,}1 + x_{3,1}$ y $logit(\phi_i) = 0{,}5x_{1,i} + 0{,}1x_{2,i} + 0{,}1x_{4,i}$.

En esta simulación están claramente identificados los tres componentes de un proceso generador de datos que corresponde con el expuesto en la sección 3.1.2. En primera medida, está la componente aleatoria que en este caso corresponde las n observaciones de la variable y. Las componentes sistemáticas dadas por las estructuras de la media $\beta_1 + \beta_2 x_{2,i} - \beta_3 x_{3,1}$ y la precisión $\gamma_1 + \gamma_2 x_{2,i} + \gamma_3 x_{4,i}$ de y_i. Finalmente las funciones de enlace en este caso para la media y la precision son el enlace logarítmico.

La estimación se lleva a cabo asignando la distribución a priori normal a cada uno de los parámetros $\gamma_j, \beta_j \quad \forall j = 1, 2, 3$ con media 0 y varianza 10^4, es decir, la distribución a priori para los parámetros es $\gamma_j, \beta_j \sim N(0, 10^4) \quad \forall j = 1, 2, 3$.

Parametrización del modelo Binomial Negativo en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud binomial negativo y se definen los predictores lineales para los parámetros de media y precisión, además, se llevan a cabo las transformaciones necesarias sobre estos parámetros dado que la distribución Binomial Negativa en OpenBUGS no se parametriza con respecto a la media y la precisión.

Posteriormente, se asignan las distribuciones a priori normales para los parámetros γ y β. En último lugar, se declara la variable *y_est* la cual es la predicción de la variable dependiente.

```
model
{
for (i in 1:N)
{
y[i] ~ dbetabin(alpha[i],beta[i])
mu[i] <- exp(beta_1*x_1[i]+ beta_2*x_2[i]+beta_3*x_3[i])/
         (1+exp(beta_1*x_1[i]+beta_2*x_2[i]+beta_3*x_3[i]));
```

```
phi[i] <- exp(gamma_1*x_1[i] + gamma_2*x_2[i] + gamma_3*x_4[i]);
beta[i] <- phi[i]-phi[i]*mu[i];
alpha[i] <- phi[i]*beta[i];
}

beta_1 ~ dnorm(0,1);
beta_2 ~ dnorm(0,1);
beta_3 ~ dnorm(0,1);
gamma_1 ~ dnorm(0,1);
gamma_2 ~ dnorm(0,1);
gamma_3 ~ dnorm(0,1);

for (i in 1:N)
{
y_est[i] ~ dbetabin(alpha[i],beta[i]);
}
}
```

El siguiente código muestra la ejecución del modelo normal heteroscedástico en *OpenBUGS* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *bugs* del paquete *R2OpenBUGS*, esta función toma el código de *OpenBUGS* del archivo $binomial_negativo.txt$ y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
data = list(x_1 = x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, y=y, N=N)

modelo_binomial_negativo =R2OpenBUGS::bugs(data_x,
                 inits = list(0,0,0),
                 model.file="binomial_negativo.txt",
                 parameters= c("beta_1", "beta_2", "beta_3",
                 "gamma_1", "gamma_2", "gamma_3", "y_est"),
                 n.chains = 3,
                 n.iter=10000
)
```

Parametrización del modelo Binomial Negativo en Stan

Para la configuración del modelo en Stan, en la primera secciń se declaran los datos a utilizar en el modelo, en este caso corresponden a las variables $Y, X_1, X_2, X_3 y X_4$ y la cantidad de obervaciones N. Posterior a la declaración de los datos se declaran los parámetros a estimar, en este caso, corresponden al conjunto de parámetros del predictor lineal de la media y al predictor lineal del parámetro de precisión.

Posterior a las declaraciones anteriores, se llevan a cabo las transformaciones de parámetros necesarias, en este caso se declaran los parámetros de media y dispersión, además, se llevan a cabo las transformaciones necesarias sobre estos parámetros dado que la función Binomial Negativa en OpenBUGS no se parametriza con respecto a la media y la precisión.

En la sección del modelo, se declaran las distribuciones a priori de los parámetros β y γ las cuales en este caso corresponden a distribuciones normales centradas en cero y con una varianza de 10^6, posteriormente, se lleva a cabo el cálculo de los valores de *mu*, *phi*, *alpha* y *beta* se asigna la verosimilitud normal a la variable y que en este caso corresponde a la distribución binomial Negativa.

Finalmente en la sección de cantidades generadas se lleva a cabo la predicción de los valores de la variable dependiente y la log verosimilitud.

```
data{
    int<lower=0> N;
    real x_1[N];
    real x_2[N];
    real x_3[N];
    real x_4[N];
    int<lower=0> y[N];
}
parameters {
    real beta_1;
    real beta_2;
    real beta_3;
    real gamma_1;
    real gamma_2;
    real gamma_3;
}
transformed parameters{
    real<lower=0> mu[N];
```

```
    real<lower=0> phi[N];
    real<lower=0, upper=1> probabilidad[N];
    real<lower=0> beta[N];

    for (i in 1:N){
         phi[i] = exp(gamma_1*x_1[i]+gamma_2*x_2[i]+gamma_3*x_4[i]);
         mu[i] = exp(beta_1*x_1[i]+beta_2*x_2[i]+beta_3*x_3[i]);
         probabilidad[i] = phi[i]/(mu[i]+phi[i]);
         beta[i] = probabilidad[i]/(1-probabilidad[i]);

             }
}
model{
    beta_1 ~ normal(0,10000);
    beta_2 ~ normal(0,10000);
    beta_3 ~ normal(0,10000);
    gamma_1 ~ normal(0,10000);
    gamma_2 ~ normal(0,10000);
    gamma_3 ~ normal(0,10000);

    for (i in 1:N){
        y[i] ~  neg_binomial_lpmf(phi[i], beta[i]);
            }

}
generated quantities{
vector[N] y_test;
vector[N] log_lik;
for (i in 1:N){
    log_lik[i] = neg_binomial_lpmf(y[i] | phi[i], beta[i]);
    y_test[i] = neg_binomial_rng(phi[i], beta[i]);
    }
}
```

El siguiente código muestra la ejecución del modelo normal heteroscedástico en *stan* desde *R*. En este fragmento se aprecia primero la manera en que se alimentan los datos a través de la lista *data*. El modelo se estima mediante la función *stan* de la distribución *rstan*. Esta función toma el código de *stan* del archivo *binomial_negativo.stan* y declara el número de iteraciones y el número de cadenas. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud *log_lik* y de la

distribución predictiva de la variable dependiente *y_test*.

```
data <- list(y=y, m=m, x_1=x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, N = N)

modelo_binomial_negativo = rstan::stan(data=data_x,
                                       file = "binomial_negativo.stan",
                            pars= c("beta_1", "beta_2", "beta_3",
                            "gamma_1", "gamma_2", "gamma_3",
                            "y_test","log_lik"),
                            chains = 4,
                            iter=20000)
```

La tabla **3-3** muestra el resumen del resultado de la estimación realizada para este modelo utilizando Stan, en la cual, se tiene una estimación a través de la media a posteriori que es bastante cercana a los valores reales de los parámetros con que se simularon los datos, de igual manera, se puede ver que los valores reales del parámetro están contenidos en los intervalos de credibilidad comprendidos entre el 2,5 % y el 97,5 % de los valores muestreados de la distribución a posteriori.

Tabla 3-3.: Resumen de Estimación Modelo Binomial Negativo

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
beta_1	0.16	0.00	0.05	0.06	0.12	0.16	0.19	0.26	6584.24	1.00
beta_2	0.10	0.00	0.00	0.10	0.10	0.10	0.10	0.11	7983.25	1.00
beta_3	0.10	0.00	0.00	0.09	0.10	0.10	0.10	0.11	8585.84	1.00
gamma_1	0.93	0.03	1.80	-2.26	-0.02	0.94	1.87	4.01	3457.13	1.00
gamma_2	0.09	0.00	0.07	-0.00	0.06	0.09	0.11	0.17	1833.30	1.00
gamma_3	0.09	0.00	0.12	-0.10	0.02	0.09	0.16	0.34	3664.27	1.00
lp__	-1388.01	0.04	2.10	-1393.26	-1389.00	-1387.58	-1386.50	-1385.31	2787.23	1.00

La tabla **3-4** muestra la estimación de la correlación entre los parámetros de la simulación, es importante resaltar que la correlación entre los parámetros de la estructura de la media, es decir, los parámetros β y los parámetros de la estructura de la varianza, es decir, los parámetros γ es muy baja. Así mismo, existe una fuerte correlación entre los interceptos y los otros parámetros.

3.3.3. Aplicación Datos Publicaciones de Estudiantes de PHD Distribución Binomial Negativa

En esta sección se presenta una aplicación con datos de [Long, 1990], los cuales, fueron trabajados por [Young, 2018] con el fin de estudiar algunos determinantes de la cantidad de

Tabla 3-4.: Correlación a posteriori modelo Binomial Negativo

	beta_1	beta_2	beta_3	gamma_1	gamma_2	gamma_3
beta_1	1					
beta_2	-0.814	1				
beta_3	-0.602	0.102	1			
gamma_1	-0.009	-0.016	0.047	1		
gamma_2	0.103	-0.091	-0.083	-0.071	1	
gamma_3	-0.038	0.045	0.012	-0.887	-0.24	1

publicaciones de estudiantes de doctorado.

El conjunto de datos se compone de una variable que tiene la cantidad de articulos publicados, el genero del estudiante (fem), el número de hijos menores de 5 años (kid), una medida del prestigio del programa doctoral (phd) y la cantidad de artículos publicada por el asesor del estudiante (ase).

Para este conjunto de datos [Young, 2018] ajusta un modelo para la media basado en una distribución de poisson y otro basado en la distribución binomial negativa con parámetro de de precisión constante, en este trabajo se complemetará el modelo binomial negativo con una estructura de regresión para el parámetro de dispersión tal como lo muestran las siguientes ecuaciones:

$$ln(\mu_i) = \beta_0 + \beta_1 fem_i + \beta_2 mar_i + \beta_3 kid_i + \beta_4 phd_i + \beta_5 ase_i \tag{3-7}$$

$$ln(\kappa_i) = \gamma_0 + \gamma_1 fem_i + \gamma_2 mar_i + \gamma_3 kid_i + \gamma_4 phd_i + \gamma_5 ase_i \tag{3-8}$$

En este trabajo, la estimación bayesiana se llevará a cabo teniendo en cuenta la función de verosimilitud binomial negativa y como distribuciones a priori del conjunto de parámetros γ_i y β_i se seleccionaron distribuciones normales $N(0, 10^k)$, para este caso se reportan los resultados para $k = 4$.

Parametrización del modelo en OpenBUGS

La parametrización del model anterior en Openbugs se observa en el siguiente fragmento de código. En la primera parte se lleva a cabo la definición de la función de verosimilitud binomial Negativa y se definen los predictores lineales con las variables fem, mar, kid, phd, men. En este caso, está el predictor lineal de la media y la precisión, sin embargo, al igual que en casos anteriores, dado que la parametrización de la distribución Binomial Negativa en

OpenBUGS no se hace con los parámetros de media y precisión se hacen necesarias algunas transformaciones sobre estos parámetros.

Posteriormente, se asignan las distribuciones a priori normales par los parámetros γ y β. Finalmente, se declara la variable y_est la cual es la predicción de la variable dependiente.

```
model
{
    for (i in 1:N)
    {
      vector_medias[i] <-exp(beta_0+ beta_1*fem[i]+
                             beta_3*kid[i]+beta_5*men[i]);
      vector_alphas[i] <-  trunc( exp(gamma_4*phd[i])+1);
      vector_probabilidad[i] <- vector_alphas[i]/(vector_medias[i]
                                +vector_alphas[i]);
      pub[i] ~ dnegbin(vector_probabilidad[i], vector_alphas[i]);
    }
    beta_0 ~ dnorm(0,0.0001);
    beta_1 ~ dnorm(0,0.0001);
    beta_3 ~ dnorm(0,0.0001);
    beta_5 ~ dnorm(0,0.0001);
    gamma_4 ~ dnorm(0,0.001);
}
```

Parametrización del modelo en Stan

Para parametrizar el modelo en Stan, en la sección de datos se declaran cuatro variables fem, mar, kid, phd, men y la variable dependiente pub, cada una como un vector de tamaño N que es la cantidad de observaciones del conjunto de datos. Los parámetros a estimar son el conjunto de parámetros β y γ los cuales son los predictores linerales del parámetro de media y de precisión respectivamente.

Posterior a la declaración de parámetros se lleva a cabo la transformación de los mismos, en este caso, se lleva a cabo el cálculo de los parámetros de media y dispersión. Al igual que en el ejemplo de simulación las transformaciones sobre los predictores lineales son mayores dada la parametrización de la distribución binomial negativa en Stan.

En la siguiente sección se lleva a cabo la declaración del modelo, en la cual se asignan las distribuciones prior para cada uno de los parámetros β y γ y se asigna a cada una de las observaciones de la variable dur la función de log verosimilitud de la distri bución binomial

negativa.

Por último, se lleva a cabo el cálculo de la log verosimilitud y la predicción de la variable dependiente en la sección de cantidades generadas.

```
    data{
    int<lower=0> N;
    int<lower=0>  pub[N];
    real fem[N];
    real mar[N];
    real kid[N];
    real phd[N];
    real men[N];

}
parameters {
    real beta_0;
    real beta_1;
    real beta_3;
    real beta_5;
    real gamma_4;

}
transformed parameters{
    real<lower=0> mu[N];
    real<lower=0> phi[N];
    real<lower=0, upper=1> probabilidad[N];
    real<lower=0> beta[N];

    for (i in 1:N){

          mu[i] = exp(beta_0+ beta_1*fem[i]+ beta_3*kid[i]+ beta_5*men[i]);
          phi[i] = exp(gamma_4*phd[i]);
          probabilidad[i] = phi[i]/(mu[i]+phi[i]);
          beta[i] = probabilidad[i]/(1-probabilidad[i]);

              }
}
model{
    beta_0 ~ normal(0,100);
```

```
    beta_1 ~ normal(0,100);
    beta_3 ~ normal(0,100);
    beta_5 ~ normal(0,100);
    gamma_4 ~ normal(0,100);

    for (i in 1:N){
        pub[i] ~  neg_binomial_lpmf(phi[i], beta[i]);
            }

}
generated quantities{
vector[N] log_lik;
vector[N] y_est;
for (i in 1:N){
    log_lik[i] = neg_binomial_lpmf(pub[i] | phi[i], beta[i]);
    y_est = neg_binomial_rng(phi[i], beta[i]);
    }
}
```

Tabla 3-5.: Resumen de Estimación Modelo Binomial Negativo para Datos de Publicaciones

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
beta_0	0.40	0.00	0.07	0.27	0.36	0.40	0.45	0.53	2698.06	1.00
beta_1	-0.22	0.00	0.07	-0.37	-0.27	-0.22	-0.17	-0.08	2970.53	1.00
beta_3	-0.14	0.00	0.05	-0.24	-0.17	-0.14	-0.11	-0.05	3575.35	1.00
beta_5	0.03	0.00	0.00	0.02	0.03	0.03	0.03	0.03	3832.50	1.00
gamma_4	0.25	0.00	0.04	0.18	0.23	0.25	0.28	0.33	3932.64	1.00

Ejecución del modelo desde R

El siguiente fragmento de código de muestra el tratamiento de datos en *R*.

```
library('rstan')
library('vcdExtra')
library('loo')
library('R2OpenBUGS')
```

```
options(mc.cores = parallel::detectCores())
rstan::rstan_options(auto_write = TRUE)

rm(list = ls())
data('PhdPubs')
df_publicaciones <- PhdPubs

rm(PhdPubs)

N <- nrow(df_publicaciones)

data_x = list(pub = df_publicaciones$articles,
              fem = df_publicaciones$female,
              mar = df_publicaciones$married,
              kid = df_publicaciones$kid5,
              phd = df_publicaciones$phdprestige,
              men = df_publicaciones$mentor,
              N = N)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $OpenBUGS$. El modelo se estima mediante la función *bugs* del paquete $R2OpenBUGS$, esta función toma el código de $OpenBUGS$ del archivo *binomial_negativa_publicaciones.txt* y declara el número de iteraciones en 50.000 y el número de cadenas en 2. En cuanto a los parámetros a estimar se tiene que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
modelo_binomial_neg_ob =R2OpenBUGS::bugs(data_x,
                                  inits = NULL,
  model.file="binomial_negativa_publicaciones.txt",
                                  parameters=c("beta_0",
                                               "beta_1",
                                               "beta_3",
                                               "beta_5",
                                               "gamma_4"

                                  ),
                                  n.chains = 2,
                                  n.iter=50000
)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $Stan$. El modelo se estima mediante la función $stan$ de la distribución $rstan$. Esta función toma el código de $stan$ del archivo $binomial_negativa_publicaciones.stan$ y declara el número de iteraciones en 10.000 y el número de cadenas en 1. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud log_lik y de la distribución predictiva de la variable dependiente y_est. Para facilitar la convergencia en los parámetros de control al parámetro $adapt\_delta$ se le asigna el valor de 0.8 y al parámetro $max\_treedepth$ el valor de 10.

```
modelo_binomial_neg  = rstan::stan(data=data_x,
                          file = "binomial_negativa_publicaciones.stan",
                          pars= c("beta_0",
                                  "beta_1",
                                  "beta_3",
                                  "beta_5",
                                  "gamma_4",
                                  "log_lik",
                                  "y_est"
                                  ),
                          chains = 1,
                          iter=10000,
                          control=list(adapt_delta = 0.8, max_treedepth = 10)
)
```

Los resultados de la estimación bayesiana se presentan en la tabla **3-5**, en ella se muestra el modelo estimado solo para aquellos parámetros para los cuales el intervalo de credibilidad del 95 % no contiene el valor cero. En el modelo de la media la variable mar que muestra si está casado o no el estudiante y la variable phd que muestra una calificación del prestigio del programa no parecen ser sinificativos, no obstante, en [Young, 2018] no se presenta estimación del error estandar de la estimación de los parámetros, razón por la cual, no es posible hacer la comparación en terminos de intervalos de credibilidad para estos parámetros. En esta estimación , las variables fem y kid tienen un efecto negativo sobre la cantidad de articulos publicados, por su parte, la variable ase que cuenta la cantidad de publicaciones del asesor tiene un efecto positivo en la cantidad de publicaciones de los estudiantes. En cuanto a la estructura del parámetro de precisión, parece haber un efecto positivo entre la variable phd y la precisión de los datos.

La tabla **3-6** muestra la correlación estimada entre las muestras de las distribuciones a posteriori de los parámetros, en esta, se puede apreciar correlaciones altas entre los parámetros

del predictor lineal de la media.

Tabla 3-6.: Correlación a posteriori Modelo Binomial Negativo Datos Publicaciones

	beta_0	beta_1	beta_3	beta_5	gamma_4
beta_0	1				
beta_1	-0.625	1			
beta_3	-0.441	0.248	1		
beta_5	-0.55	0.118	-0.039	1	
gamma_4	0.084	-0.003	-0.013	-0.133	1

3.3.4. Aplicación Datos de Zanahorias Distribución Beta Binomial

En esta sección se presenta una aplicación con datos de [Phelps, 1982], los cuales, fueron trabajados por [McCullagh, 1983] con el fin de mostrar el efecto que un tratamiento con insecticida produce en la cantidad de zanahorias que presentan daños por insectos. El conjunto de datos se compone de una variable que es el logaritmo de la dosis recibida *dose* y el cociente entre la cantidad de zanahorias en que se observaron daños *obs* y la cantidad de zanahorias totales observadas m para tres grupos diferentes lv_1 lv_2 y lv_3. Cabe destacar que paraq ningún nivel de dosis ni para ningún grupo la cantidad de zanahorias observadas es la misma, razón por la cual es posible pensar en un modelo beta binomial en el que la variable a modelar es la cantidad de zanahorias dañada y el total de experimentos es la cantidad de zanahorias observadas para cada dosis.

Para este conjunto de datos [McCullagh, 1983] ajusta un modelo para las proporciones basado en la dosis de insecticida recibida y el grupo al que pertenece la observación. En este trabajo se adopta un enfoque basado en la distribución beta binomial con una estructura de regresión para el parámetro de media y otra para el parámetro de precisión tal como se definió en el modelo 3.1.1. Las siguientes ecuaciones muestran los predictores lineales del modelo.

$$logit(\mu_i) = \gamma_1 dose_i + \gamma_2 lv\_1_i + \gamma_3 lv\_2_i; \tag{3-9}$$

$$logit(\phi_i) = \beta_1 dose_i + \beta_2 lv\_1_i + \beta_3 lv\_2_i; \tag{3-10}$$

En este trabajo, la estimación bayesiana se llevará a cabo teniendo en cuenta la función de verosimilitud beta binomial y como distribuciones a priori del conjunto de parámetros γ_i y β_i se seleccionaron distribuciones normales $N(0, 10^k)$, para este caso se reportan los resultados para $k = 4$.

Parametrización del modelo en Stan

Para parametrizar el modelo en Stan, en la sección de datos se declaran cuatro variables *dose*, lv_1, lv_2, lv_3 y la variable dependiente *obs*, cada una como un vector de tamaño N que es la cantidad de observaciones del conjunto de datos. Es necesario tener en cuenta que la distribución binomial negativa en Stan modela la cantidad de fracasos hasta obtener un numero derminado de exitos, razón por la cual, es necesario transformar la variable *obs* para que muestra la cantidad de zanahorias no afectadas y no la cantidad de zanahorias afectadas . Los parámetros a estimar son el conjunto de parámetros β y γ los cuales son los predictores linerales del parámetro de media y de precisión respectivamente.

Posterior a la declaración de parámetros se lleva a cabo la transformación de los mismos, en este caso, se lleva a cabo el cálculo de los parámetros de media y dispersión. Al igual que en el ejemplo de simulación las transformaciones sobre los predictores lineales son mayores dada la parametrización de la distribución beta binomial en Stan.

En la siguiente sección se lleva a cabo la declaración del modelo, en la cual se asignan las distribuciones prior para cada uno de los parámetros β y γ y se asigna a cada una de las observaciones de la variable *obs* la función de log verosimilitud de la distri bución beta binomial.

Por último, se lleva a cabo el cálculo de la log verosimilitud y la predicción de la variable dependiente en la sección de cantidades generadas.

```
    data{
    int<lower=0> N;
    int<lower=0> m[N];
    int<lower=0> obs[N];
    real dose[N];
    real lv_1[N];
    real lv_2[N];
    real lv_3[N];
    }
parameters {
    real beta_1;
    real beta_2;
    real beta_3;
    real beta_4;
    real gamma_1;
    real gamma_2;
    real gamma_3;
```

```
    real gamma_4;
}
transformed parameters{
    real<lower=0> mu[N];
    real<lower=0> phi[N];
    real<lower=0> exp_mu[N];
    real<lower=0> exp_phi[N];
    real<lower=0> alpha[N];
    real<lower=0> beta[N];

    for (i in 1:N){
         exp_phi[i] = exp(gamma_1*dose[i]+gamma_2*lv_1[i]+gamma_3*lv_2[i]+gamma_4*lv_3
         exp_mu[i] = exp(beta_1*dose[i]+beta_2*lv_1[i]+beta_3*lv_2[i]+beta_4*lv_3[i]);
         mu[i] = exp_mu[i]/(1+exp_mu[i]);
         phi[i] = exp_phi[i]/(1+exp_phi[i]);
         alpha[i] = mu[i]*phi[i];
         beta[i] = phi[i]-phi[i]*mu[i];
    }
}
model{
    beta_1 ~ normal(0,10);
    beta_2 ~ normal(0,10);
    beta_3 ~ normal(0,10);
    beta_4 ~ normal(0,10);
    gamma_1 ~ normal(0,10);
    gamma_2 ~ normal(0,10);
    gamma_3 ~ normal(0,10);
    gamma_4 ~ normal(0,10);

    for (i in 1:N){
        obs[i] ~  beta_binomial_lpmf(m[i], alpha[i], beta[i]);
            }

}
generated quantities{
vector[N] log_lik;
vector[N] y_test;
for (i in 1:N){
    log_lik[i] = beta_binomial_lpmf(obs[i] | m[i], alpha[i],beta[i]);
    y_test[i] = beta_binomial_rng(m[i], alpha[i],beta[i]);
```

```
        }
}
```

Ejecución del modelo desde R

El siguiente fragmento de código de muestra el tratamiento de datos en R.

```
library('rstan')
library('loo')
options(mc.cores = parallel::detectCores())
rstan::rstan_options(auto_write = TRUE)

rm(list = ls())

log_dose <- rep(c(1.52, 1.64, 1.76, 1.88,
                  2.00, 2.12, 2.24, 2.36),
                3)

m <- c(35, 42, 50, 42, 35, 42, 32, 28,
       38, 40, 33, 39, 47, 42, 35, 35,
       34, 38, 36, 35, 49, 40, 22, 31)

level <- c(1,1,1,1,1,1,1,1,
           2,2,2,2,2,2,2,2,
           3,3,3,3,3,3,3,3)

observados <- c(10, 16, 8, 6, 9, 9, 1, 2,
                17, 10, 8, 8, 5, 17, 6, 4,
                10, 10, 5, 3, 2, 1, 3, 2)

no_observados <- m-observados

df_zanahorias <- data.frame(observados, no_observados, m, log_dose, level)
df_zanahorias$level_1 <- ifelse(df_zanahorias$level == 1, 1, 0)
df_zanahorias$level_2 <- ifelse(df_zanahorias$level == 2, 1, 0)
df_zanahorias$level_3 <- ifelse(df_zanahorias$level == 3, 1, 0)

N <- nrow(df_zanahorias)
```

```
data = list(m = df_zanahorias$m,
            obs = df_zanahorias$no_observados,
            dose = df_zanahorias$log_dose,
            lv_1 = df_zanahorias$level_1,
            lv_2 = df_zanahorias$level_2,
            lv_3 = df_zanahorias$level_3,
            N = N)
```

El siguiente código muestra la ejecución del modelo desde R utilizando $OpenBUGS$. El modelo se estima mediante la función *bugs* del paquete $R2OpenBUGS$, esta función toma el código de $OpenBUGS$ del archivo $beta\_binomial\_zanahorias.txt$ y declara el número de iteraciones en 10.000 y el número de cadenas en 4. En cuanto a los parámetros a estimar se tiene que además de los parámetros β y γ se obtienen muestras de la distribución predictiva de la variable dependiente y_est.

```
parametros = c( "beta_1",
                "gamma_1",
                "y_est"
)
```

```
modelo_zanahorias  = rstan::stan(data=data_x,
                                 file = "beta_binomial_zanahorias.stan",
                                 pars= parametros,
                                 chains = 4,
                                 iter=10000,
                                 control=list(adapt_delta = 0.99,
                                              max_treedepth = 10))
```

El siguiente código muestra la ejecución del modelo desde R utilizando $Stan$. El modelo se estima mediante la función *stan* de la distribución *rstan*. Esta función toma el código de *stan* del archivo $beta_binomial_zanahorias.stan$ y declara el número de iteraciones en 10.000 y el número de cadenas en 1. En cuanto a los parámetros a estimar se tiene el que además de los parámetros β y γ se obtienen muestras de la logverosimilitud log_lik y de la distribución predictiva de la variable dependiente y_est. Para facilitar la convergencia en los parámetros de control al parámetro $adapt\_delta$ se le asigna el valor de 0.99 y al parámetro $max\_treedepth$ el valor de 10.

```
parametros = c( "beta_1",
                "gamma_1",
                "log_lik", "y_test"
)


modelo_zanahorias  = rstan::stan(data=data_x,
                                 file = "beta_binomial_zanahorias.stan",
                                 pars= parametros,
                                 chains = 4,
                                 iter=10000,
                                 control=list(adapt_delta = 0.99,
                                              max_treedepth = 10))
```

Los resultados de la estimación bayesiana se presentan en la tabla **3-7**, en ella se muestra el modelo estimado solo para aquellos parámetros para los cuales el intervalo de credibilidad del 95 % no contiene el valor cero. El modelo de la media la variable *obs* muestra que un aumento de la dosis tiene un efecto positivo en la cantidad de zanahorias que no presentaron da´ nos por insctos, lo cual, es intuituvamente correcto, por su parte, la misma variable tiene un efecto positivo en la dispersión de esta variable.

Tabla 3-7.: Resumen de Estimación Modelo Beta Binomial para Datos de zanahorias

	mean	se_mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	n_eff	Rhat
beta_1	0.35	0.00	0.13	0.09	0.26	0.35	0.44	0.61	6638.02	1.00
gamma_1	8.92	0.09	5.71	1.65	4.44	7.66	12.22	22.75	3860.53	1.00

La tabla **3-8** muestra la correlación estimada entre las muestras de las distribuciones a posteriori de los parámetros, en esta, se puede apreciar correlaciones altas entre los parámetros del predictor lineal de la media.

Tabla 3-8.: Correlación a posteriori Modelo BB Datos Zanahorias

	beta_1	gamma_1
beta_1	1	
gamma_1	0.016	1

4. Modelamiento de datos longitudinales

En esta sección, a manera de complemento, se tratará el modelamiento de datos longitudinales como una extensión del trabajo realizado hasta este punto. Lo anterior, dado que en el modelamiento de datos longitudinales un componente de suma importancia es modelar la varianza y covarianza entre las observaciones de cada uno de los individuos.

Los datos longitudinales pueden ser definidos como observaciones repetidas de uno o varios individuos a lo largo del tiempo [Weiss, 2005]. Este tipo de datos es de suma importancia en diversas áreas del saber, por ejemplo, en finanzas, los precios de un conjunto de acciones generalmente se analizan a lo largo de un periodo de tiempo para determinar sus comportamientos conjuntos, establecer su relación con variables macroeconomicas o tratar de predecir valores futuros; en medicina, para evaluar el efecto de un medicamento se pueden realizar estudios donde se someten pacientes a un tratamiento y se observa su evolución en comparación con un grupo de pacientes sin tratamiento; en economía, se pueden observar medidas macreoconómicas de un conjunto de paises con el fin de evaluar diversas teorías o el efecto de choques importantes como la caida del mercado de valores [Fitzmaurice & Ravichandran, 2008].

Dado que los datos longitudinales son medidas repetidas de individuos a lo largo del tiempo, es importante señalar que su análisis se centra en el modelamiento de la media de los individuos y de la correlación temporal de la observaciones de los mismos, este último componente es de suma importancia debido a que es de suponerse que existe algun tipo de dependencia entre las diferentes observaciones de un mismo individuo. Teniendo en cuenta lo anterior, una forma general de modelar estos datos consiste en establecer una estructura para la media de las observaciones de los individuos y modelar la matriz de varianzas y covarianzas de las observaciones [Fitzmaurice & Ravichandran, 2008].

Respecto al enfoque anterior, en la literatura referente al modelamiento de datos longitudinales existen diversas propuestas para modelar la matriz de varianzas y covarianzas, como por ejemplo adoptar un enfoque de varianza estructurado o no estructurado [Weiss, 2005, Fitzmaurice & Ravichandran, 2008]. El enfoque estructurado permite considerar estructuras de correlación estacionarias o no entre observaciones a través del tiempo así como también varianzas homogeneas o no con respècto al tiempo. En el caso de estructuras de covarianza estacionarias se tienen por ejemplo las SC, modelos autorregresivos de orden p $Ar(p)$ o estructuras de autorregresivas de media movil $ARMA(p, q)$ [Castillo et al., 2020].

En lo referente a la estimación de modelos de datos longitudinales, respecto a la estimación clásica se tiene que el enfoque de máxima verosimilitud restringida resulta apropiado en la mayoria de los casos ya que los parámetros que componen las estructuras de matrices de covarianzas tienen rangos definidos [Weiss, 2005]. En lo referente al enfoque Bayesiano, el cual es el adoptado en este trabajo, es importante destacar que en cuanto al modelamiento con matrices de covarianzas no estructuradas lo usual es utilizar como distribución a priori una distribución Wishart tal como lo propusieron [Brown et al., 2001].

En lo referente al modelamiento de matrices de covarianza con estructuras sencillas como por ejemplo las mencionadas anteriormente, el modelamiento para la estructura de la media puede hacerse con distribuciones a priori normales para los parámetros del predictor lineal, para el caso de los parámetros de la matriz de covarianzas es usual tomar como prior distribuciones de dominio en los reales positivos para los parámetros de la varianza y distribuciones con restricciones en el intervalo $(0, 1)$ para los parámetros de correlación [Hui & Berger, 1983]. También el modelamiento con estructuras de regresió tanto para la media como para la matríz de varianzas y covarianzas ha sido desarrollado en el contexto bayesiano, en el trabajo de [Cepeda, 2001a] se utilizan prior normales para los regresores de la media y la varianza con un modelo normal mutivariado.

En este capítulo se trata primero qué son exactamente los datos longitudinales para posteriomente plantear un modelo basado en la distribución normal multivariada. A continuación, se presentan algunas estructuras en la matriz de covarianzas. Finalmente, se presentarán algunos estudios de simulación con el fin de ilustrar la manera como se debe parametrizar este tipo de modelos en herramientas como Stan y OpenBUGS y se muestran algunas aplicaciones con grupos de datos reales.

4.1. Modelo General para Datos Longitudinales

En este caso, se considera que existen N individuos que han sido observados en T tiempos consecutivos, para el caso de datos longitudinales no es necesario ni que las observaciones se den en periodos consecutivos ni siquiera en los mismos periodos de tiempo, sin embargo, en este documento se sigue el enfoque en el cual todas los individuos son observados el mismo numero de veces en los mismos periodos de tiempo.

El vector Y_i para $i = 1, ..., N$ contiene todas las observaciones del i-esimo individuo y por lo tanto es un vector de dimension $1 * T$ porque el individuo fue observado T veces.
El vector X_i para $i = 1, ..., N$ contienen todas las covariables del i-esimo individuo, si se supone que existen s covariables, entonces el vector sería de dimension $1 * s$ en cada momento

del tiempo. Por ejemplo, si se observaran tres covariables cada que se observa la variable de interés se tienen que al final de 4 mediciones se tienen 12 medidas de las covariables.

Utilizando una notacioón matricial se tiene que $Y_i = (Y_{i,1}, Y_{i,2}, ..., Y_{i,T})$ es el vector de observaciones del i-esimo individuo y $t_i = (t_{i,1}, t_{i,2}, ..., ti, T)$ es el vector que contiene los tiempos en los cuales se tomaron las mediciones para el i-esimo individuo. Además, $X = (X_1, X_2, ..., X_i, ..., X_n)$ con $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,s})$ es un conjunto de covariables de cada uno de los individuos. Además, sea $\sigma^2 V = \Sigma$ es una matriz diagonal por bloques con bloques no nulos donde $\sigma^2 V_0$ de tamaño $T * T$, donde el i-ésimo bloque es la matriz de varianzas y covarianzas de las observaciones del i-ésimo individuo. Cabe destacar que al ser una matriz por bloques, implica que la covarianza entre las observaciones de diferentes individuos es cero.

Ahora, teniendo en cuenta que se plantea que las mediciones realizadas entre individuos distintos son independientes entre sí, y se piensa en la media de cada individuo como una combiación lineal de funciones de las covariables, asumiendo que las respuestas están distribuidas de manera normal multivariante se plantea el siguiente modelo:

$$Y \sim NM_T(X\beta, \Sigma(\alpha)) \tag{4-1}$$

Donde $Y \sim NM_T(\mu, \Sigma)$ es la distribución normal multivariante de T componentes y μ y Σ son el vector de medias y la matriz de varianzas y covarianzas respectivamente. β es un vector de parámetros de dimensión p fijos y desconocidos y α hace referencia al conjunto de parámetros que componen la matriz Σ. Profundizando un poco sobre el vector de parámetros α, se debe tener en cuenta varias cosas, primero, como se verá más adelante, la cantidad de componentes de α en un modelo de covarianza no estructurado puede ser grande en relación al total de observaciones ya que incluye todas las covarianzas entre cada una de las obsevaciones de cada uno de los individuos. Segundo, cada uno de los componentes del vector α , puede ser incluso la combinación lineal de por ejemplo otro grupo de covariables, Z, tal y como se mostró en los modelos de las secciones anteriores y como lo plantea [Cepeda & Gamerman, 2004].

4.1.1. Parametrización del Modelo General en OpenBUGS

Una vez expuesto el modelo general de datos longitudinales basado en la distribución normal multivariada, se explica su parametrización en OpenBUGS, la cual, consiste en la declaración de la verosimilitud correspondiente a la distribución normal multivariante y las distribuciones a priori para los parámetros de las estructura de la media $\beta_i \forall 1 = 1, ..., s$ y para los parámetros α de los cuales depende la matriz de varianzas y covarianzas.

El siguiente, es el pseudo código con el cual se lleva a cabo la parametrización del modelo expuesto en la sección 4.1.

```
model
{
// En la primera parte se declara la función de verosimilitud
for (i in 1:N)
    {
    y[i,1:Tiempo] ~ dmnorm(media,precision)
    for(t in 1:Tiempo){
        }
    }
// Declaración de distribuciones a priori
beta ~ distribucion_prior();
alpha ~ distribucion_prior();

//Transformaciones necesarias
media <-f(X*beta)
precision <- h(alpha);
}
```

Parametrización del modelo mediante la interfaz Gráfica de Open BUGS

En el apartado 1.2.4 se presentó la manera como utilizar la interfaz gráfica de *OpenBUGS* para estimar modelos en esta herramienta. A manera de resumen los siguientes son los pasos para estimar un modelo en *OpenBUGS* mediante la interfaz gráfica:

1. Declarar el modelo y cargar los datos en $Model > Specification$ a partir del modelo y los datos creados en el editor que se despliega en $File > New$.
2. Establecer los parámetros a estimar en $Inference > Samples$.
3. Realizar el muestreo en $Model > Update$.
4. Obtener los resultados del muestreo en $Inference > Samples$.

La declaración del modelo en OpenBUGS se hace mediante la definición del mismo en el editor $File > New$ y la interfaz de especificación del modelo $Model > Specification$. A manera de ejemplo el siguiente fragmento de código parametriza un modelo en OpenBus, en este modelo existen N individuos que tienen observaciones que se modelarán con una distribución normal T variada, lo cual, compone la función de verosimilitud. También se declaran las distribuciones *prior* y las transformaciones necesarias para llevar a cabo la estimación del modelo.

```
model
{
// En la primera parte se declara la función de verosimilitud
for (i in 1:N)
    {
    y[i,1:Tiempo] ~ dmnorm(media,precision)
    for(t in 1:Tiempo){
        }
    }
// Declaración de distribuciones a priori
beta ~ distribucion_prior();
alpha ~ distribucion_prior();

//Transformaciones necesarias
media <-f(X*beta)
precision <- h(alpha);
}
```

Una vez presentada la forma como se debe parametrizar el modelo en OpenBUGS es necesario mostrar como se deben presentar los datos. El siguiente fragmento de código muestra un ejemplo de como presentar matrices de datos en OpenBugs. En este caso, se cargan dos matrices, la matriz y, la matriz x y la matriz z, las cuales, contienen las observaciones de la variabe dependiente y las variables independientes respectivamente y tienen en ambos casos dimension $10*4$. Es importante recalcar que OpenBUGS carga los datos de matrices por filas, lo cual quiere decir que en este ejemplo los T primeros datos de cada matriz corresponden a la primera fila, los siguientes T a la segunda fila y así sucesivamente. En este caso, las matrices están definidas para $T = 4$ y $N = 10$, de esta manera, la i-èsima fila de la matriz y es el vector de las observaciones de la variable dependiente del iésimo individuo que se llevaron a cabo en T momentos, por su parte, la i-ésima fila de la matriz x y de la matriz z es el vector de observaciones de la covariable x y la covariable z observado en T momentos diferentes.

```
list(
    y = structure(
    .Data = c(
    4.39,  9.00, 13.87, 19.66,
    4.71, 10.11, 14.40, 19.21,
    .
    .
    .
```

```
    4.43,  9.97, 14.95, 20.04),.Dim = c(10, 4)
    ),
    x = structure(
    .Data = c(
    29.68673, 33.15317, 79.65242, 73.78261,
    79.24622, 12.87030, 27.85970, 96.05825,
    .
    .
    .
    55.01871, 77.93981, 99.83065, 72.14452),.Dim = c(10, 4)
    ),
    z = structure(
    .Data = c(
    2.867, 3.157, 9.24, 3.78,
    7.462, 2.030, 2.90, 9.05,
    .
    .
    .
    55.01871, 77.93981, 99.83065, 72.14452),.Dim = c(10, 4)
    )
)
```

Finalmente, en caso de querer especificar los valores iniciales de los parámetros, es decir, los valores con los cuales comenzará el proceso iterativo, se pueden suministrar en otra lista como lo muestra el siguiente fragmento de código, el cual contiene un ejemplo de inicio para un parametro *beta* y un parámetro *alpha*.

```
list(beta=c(10,0,0,10), alpha=c(0))
```

Una vez que se ha especificado y compilado el modelo y cargado las observaciones y si se desea los valores iniciales de los parámetros, se pueden establecer otros parámetross de interés de la estimación como el numero de cadenas, la cantidad de iteraciones y la cantidad de iteraciones que serán tomadas como estabilización en el proceso de estimación.

4.1.2. Parametrización del Modelo General en Stan

Tras haber presentado la parametrización de este modelo general en OpenBUGS, se presenta la parametrización del modelo en Stan, la cual, como se ha mostrado hasta este punto en varias oportunidades, es un poco más elaborada.

La primera sección del código declara los datos a utilizar en el modelo. Para este caso específico, se debe especificar la cantidad de unidades observacionales N, la cantidad de observaciones T, la variable dependiente y las covariables, las cuales, se declaran como matrices de N filas y T columnas. La segunda sección declara los parámetros a estimar en el modelo, es decir, los parámetros β y α.
El siguiente es un pseudo código donde se ilustra la declaración de estas dos secciones de código.

```
data{
    int<lower=0> N;
    int<lower=0> T;
    matrix[N,T] y;
    matrix[N,T] X;
}
parameters {
    real[N] beta;
    real alpha;
    }
```

La tercera sección de este código es la sección de parámetros transformados, en ella especificamente se llevará a cabo el calculo de los predictores lineales necesarios para los modelos y el calculo de la matriz de varianzas y covarianzas a utilizar. En la siguiente sección se declara el modelo como tal, el cual al igual que en OpenBUGS consiste en declarar la función de verosimilitud normal multivariada y las distribuciones a apriori de los parámetros a estimar.

La sección de parámetros transformados y la sección de declaración del modelo se muestran en el siguiente pseudo código:

```
transformed parameters{

    matrix[T,T] varianza = f(alpha);
    matrix[N,T] media = h(X*beta);
}
model{
    beta ~ distribucion_prior();
    alpha ~ distribucion_prior();
    for (i in 1:N){
        y[i,1:T] ~ multi_normal_lpdf(media[i,1:T], varianza);
        }
}
```

Finalmente, la última sección a tener en cuenta en este modelo es la sección de cantidades generadas, en este caso, se ilustra como calcular la log verosimilitud del modelo y la predicción de la variable independiente.

```
generated quantities{
matrix[N] y_test;
vector[N] log_lik;
for (i in 1:N){
    y_test[i] = multi_normal_rng(media[i,1:T], varianza);
    log_lik[i] = multi_normal_lpdf(y[i]| media[i,1:T], varianza);
}
}
```

4.2. Modelos de datos longitudinales con Covarianza Estructurada y No Estucturada

Tal como ha sido mencionado anteriormente, en el contexto de los datos longitudinales el modelamiento de la matriz de varianzas y covarianzas es de suma importancia, para lo cual, existen básicamente dos enfoques. Primero, el modelo de covarianza no estructurado en el cual, como su nombre lo indica, no se asume ninguna estructura de varianza compartida por todos los individuos y cada uno de los parámetros de correlación debe ser estimado para cada individuo. Segundo, el modelo de covarianza estructurado que básicamente asume que hay una estructura de covarianza común para todos los individuos, con lo que se gana eficiencia en la estimación ya que el número de parámetros a estimar se reduce drásticamente [Weiss, 2005].

4.2.1. Algunas estructuras de Covarianza

En esta sección se presentan las estructuras de varianzas y covarianzas utilizadas en este trabajo, entre las cuales se incluyen los modelos de simetría compuesta, el modelo autorregresivo de orden 1, y los modelos antedependientes estructurados. Para profundizar este tipo de estructuras algunas referencias relevantes son el capítulo 8 en [Weiss, 2005] y el capítulo 6 en [Hedeker & Gibbons, 2006].

Estructura de Covarianza de Simetría Compuesta - SC:

El modelo de covarianza de simetría compuesta asume que las observaciones tienen la misma varianza y que la correlación entre observaciones es la misma sin importar la distancia temporal entre las mismas, es decir, $cov_{i,j}$ es constante para todo $i \neq j$ y $var_j = \sigma^2$ tambien

es constante para todas las observaciones del individuo i.

La siguiente es la matriz de varianzas y covarianzas de un modelo de simetría compuesta, en ella, se tienen dos parámetros σ^2 y ρ que corresponden cada uno a la varianza de la cada una de las observaciones del individuo i y a la correlación entre dos observaciones cualquiera de dicho individuo, para $T = 5$, en ella se ve claramente que la correlación no varía con la distancia temporal entre las observaciones de los individuos.

$$\mathbf{Cov}(\mathbf{Y_i}) = \sigma^2 \begin{bmatrix} 1 & \rho & \rho & \rho & \rho \\ & 1 & \rho & \rho & \rho \\ & & 1 & \rho & \rho \\ & & & 1 & \rho \\ & & & & 1 \end{bmatrix}$$

En otras palabras, $[Cov]_{i,j} = \sigma^2 \quad \forall i = j$ y $[Cov]_{i,j} = \rho\sigma^2 \quad \forall i \neq j$. En el modelo de simetría compuesta los parámetros de la matriz de varianzas y covarianzas a estimar son σ^2 y ρ.

Estructura de Covarianza Autorregresivo de Orden 1 - AR(1)

El modelo autorregresivo de orden 1 es muy parecido al modelo de Simetría Compuesta, asume que las observaciones tienen la misma varianza y que la correlación entre observaciones disminuye acorde a la distancia temporal de las mismas, es decir, la correlación entre dos observaciones entre el tiempo i y j esta dada por la siguiente expresión.

$$\rho_{i,j} = \rho^{|i-j|}$$

En esta ecuación se ve la similitud entre el modelo AR(1) y el modelo SC, los dos estiman los mismos parámetros σ^2 y ρ pero el modelo AR(1) transforma las correlaciones de tal forma que a mayor distancia temporal disminuye su valor dado que $|\rho| < 1$. Teniendo en cuenta lo anterior la matriz de varianzas y covarianzas para el modelo AR(1) con $T = 5$ es la siguiente:

$$\mathbf{Cov}(\mathbf{Y_i}) = \sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & \rho^4 \\ & 1 & \rho & \rho^2 & \rho^3 \\ & & 1 & \rho & \rho^2 \\ & & & 1 & \rho \\ & & & & 1 \end{bmatrix}$$

En otras palabras, de manera general, $[Cov]_{i,j} = \sigma^2 \quad \forall i = j$ y $[Cov]_{i,j} = \sigma^2 * \rho^{|i-j|} \quad \forall i \neq j$. En el modelo autorregresivo de orden , al igual que en el de simetría compuesta, los parámetros de la matriz de varianzas y covarianzas a estimar son σ^2 y ρ.

Estructura de Covarianza Autorregresiva de Media Móvil de Orden 1, 1 - ARMA(1,1):

El modelo ARMA(1,1) es una generalización de los modelos SC y AR(1), este modelo considera la siguiente matríz de varianzas y covarianzas, la cual tiene tres parámetros σ^2, ϕ y ρ. La siguiente, es la matriz de varianzas y covarianzas para el modelo ARMA(1,1) con $T = 5$, en el caso en el que ϕ y ρ son iguales se tiene el modelo AR(1) y en el caso en que ρ es igual a 1 se tiene el modelo SC. Un caso especial a tener en cuenta es el que se presenta cuando el parámetro ρ es igual a cero, en ese caso, se tiene el modelo de media móvil de orden 1 - MA(1).

$$\mathbf{Cov(Y_i)} = \sigma^2 \begin{bmatrix} 1 & \phi & \phi\rho & \phi\rho^2 & \phi\rho^3 \\ & 1 & \phi & \phi\rho & \phi\rho^2 \\ & & 1 & \phi & \phi\rho \\ & & & 1 & \phi \\ & & & & 1 \end{bmatrix}$$

En otras palabras, $[Cov]_{i,j} = \sigma^2 \quad \forall i = j$ y $[Cov]_{i,j} = \sigma^2 * \phi * \rho^{|i-j|-1} \quad \forall i \neq j$. De esta generalización se ve claramente que si $\phi = \rho$ se tiene la mtriz de covarianzas del modelo AR(1) y si $rho = 1$ se tiene la matriz de modelo de simetría compuesta. En el caso del modelo ARMA(1,1) los parámetros de la matriz de varianzas y covarianzas a estimar son σ^2, ρ y ϕ.

Para facilitar la interpretación del modelo ARMA(1,1) se puede tener en cuenta que este tambien puede ser escrito de la siguiente manera:

$$Y_{i,j} = \rho * Y_{i,j-1} + \phi * \epsilon_{i,j-1} + \epsilon_{i,j}$$

La anterior ecuación, muestra que la j-esima observación del i-esimo individuo depende tanto de la observación como del error en el momento anterior, por lo tanto, los parámetros ρ y ϕ son respectivamente la correlación entre la observación actual y la observación anterior y entre la observación actual y el error anterior.

Estructura de Covarianza Antedependiente Estructurada de Orden 1 - ADS(1):

Los modelos de antedependecia son una generalización de los modelos autorregresivos en los cuales se permite correlaciones no constantes en algunos ordenes, en el caso del modelo ADS(1) esa correlación no constante se da con el rezago 1 [Weiss, 2005]. Por lo anterior, para este modelo, teniendo que $T = 5$ la matriz de varianzas y covarianzas viene dada por:

$$\mathbf{Cov}(\mathbf{Y_i}) = \sigma^2 \begin{bmatrix} 1 & \rho_1 & \rho_1\rho_2 & \rho_1\rho_2\rho_3 & \rho_1\rho_2\rho_3\rho_4 \\ & 1 & \rho_2 & \rho_2\rho_3 & \rho_2\rho_3\rho_4 \\ & & 1 & \rho_3 & \rho_3\rho_4 \\ & & & 1 & \rho_4 \\ & & & & 1 \end{bmatrix}$$

Donde se tiene que ρ_j es la correlación entre la observacion j y $j+1$. [Castillo, 2017] plantea que para $j = 1, ..., T$ primero, los parámetros de correlación están determinados por $\rho_j = \rho^{f(j,\lambda)-f(j+1,\lambda)}$ y segundo, la función $f(j,\lambda)$ estará determinada por:

$$f(j,\lambda) = \begin{cases} (j^\lambda - 1)/\lambda & si \quad \lambda \neq 0 \\ log(j) & si \quad \lambda = 0 \end{cases}$$

En el modelo ADS(1) los parámetros de la matriz de varianzas y covarianzas a estimar son σ^2, ρ y λ.

Estructura de Covarianza Antedependiente Estructurada de Orden 2 - ADS(2):

En el caso del modelo ADS(2), teniendo en cuenta lo planteado para el modelo ADS(1), la matríz de varianzas y covarianzas teniendo $T = 5$ para este modelo es la siguiente,

$$\mathbf{Cov} = \sigma^2 \begin{bmatrix} 1 & \sqrt{\psi}\rho_1 & \sqrt{\psi}\rho_1\rho_2 & \sqrt{\psi}\rho_1\rho_2\rho_3 & \sqrt{\psi}\rho_1\rho_2\rho_3\rho_4 \\ & 1 & \sqrt{\psi}\rho_2 & \sqrt{\psi}\rho_2\rho_3 & \sqrt{\psi}\rho_2\rho_3\rho_4 \\ & & 1 & \sqrt{\psi}\rho_3 & \sqrt{\psi}\rho_3\rho_4 \\ & & & 1 & \sqrt{\psi}\rho_4 \\ & & & & 1 \end{bmatrix}$$

Donde se tienen las mismas condiciones que en el modelo anterior y los parámetros a estimar son σ^2, ρ, λ y ψ.

4.3. Simulaciones y aplicaciones

En esta sección se llevarán a cabo simulaciones para datos longitudinales con el modelo expuesto en la ecuación 4.1, para lo cual, se adoptará un modelo general para la media y se utilizará la matriz de varianzas y covarianzas para cada una de las estructuras expuestas anteriormente. El objetivo de esta sección es mostrar como llevar a cabo la parametrización de este tipo de modelos en OpenBUGS y Stan, para lo cual, se seguirá la propuesta bayesiana de varios de estos modelos hecha por [Castillo, 2017].

Para este estudio de simulación se considera que existen $N = 50$ individuos todos observados en $T = 4$ momentos del tiempo consecutivos, por cuestiones de simplicidad en este estudio se considera el caso en que las observaciones se dan en periodos consecutivos en los mismos periodos de tiempo. Como se explicó anteriormente el vector $Y_i \quad \forall i = 1, ..., N$ contiene las 4 observaciones del i-ésimo individuo y por lo tanto es un vector de dimension $1 * 4$.

Además de lo anterior, se tiene que cada uno de los individuos en este caso tiene dos covariables, T y T^2, las cuales, hacen referencia al periodo en el cual se lleva a cabo la observación y el cuadrado de dichos periodos respectivamente. Existe también un vector de parámetros β a estimar de manera que $E[Y_{i,t}] = t * \beta_1 + t^2\beta_2$, es decir, la media es modelada con una estructura cuadrática con respecto al tiempo.

Por lo tanto, teniendo el cuenta el enfoque de la ecuación 4.1 se tiene que $Y_i \sim NM_4(t_i\beta_1 + t_i^2\beta_2, \Sigma_i)$ donde $t_i\beta_1 + t_i^2\beta_2$ es el vector de valo esperado de dimensión $1 * 4$ y σ_i es una matriz de varianzas y covarianzas de dimensión $4 * 4$ que corresponde con alguna de las estructuras expuestas en la sección 4.2.1. A su vez, t_i es un vector $t_i = (1, 2, 3, 4)$ y $t_i^2 = (1, 4, 9, 16)$. La simulación de la media se llevó a cabo con valores de $\beta_1 = 1$ y $\beta_2 = 2$.

4.3.1. Simulación Modelo de Simetría Compuesta

Para llevar a cabo la simulación del modelo de simetría compuesta se asume una estructura de covarianzas como la expuesta en la sección 4.2.1, para esto, se generan valores de la matriz de varianzas y covarianzas con $\gamma = 0{,}25$ y $\rho = 0{,}5$ con $\sigma^2 = 1/\gamma$. Como distribuciones a priori se asigna una distribución $U(0, 1)$ ya que es un parámetro de correlación y en datos longitudinales es usual ver correlación positiva entre las observaciones de los individuos. Para el parámetro γ se asume una distribución a priori $\Gamma(3, 2)$ y para los parámetros $\beta_i \forall i = 1, 2$ se asume una distribución normal $N(0, 10^4)$.

La función de verosimilitud en este modelo esta dada por la siguiente expresión:

$$L(\beta, \sigma^2, \rho|Y) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{T}{2}}|\Sigma(\sigma^2, \rho)|^{\frac{1}{2}}} exp[-\frac{1}{2}(y_i - x_i\beta)^T\Sigma(\sigma^2, \rho)^{-1}(y_i - x_i\beta)]$$

Donde la matriz $\Sigma(\sigma^2, \rho)$ corresponde a la matriz de varianzas y covarianzas del modelo de simetría compuesta explicado en la sección 4.2.1.

El siguiente código de R muestra la simulación de datos para el modelo planteado anteriormente:

```
generarCov <- function(Tiempo, rho, sig2, phi){
  covi <- diag(1,Tiempo)
```

```
  covi[covi == 0] <- phi
  for(i in 1:Tiempo){
    for(j in 1:Tiempo){
      if (i != j) {
        covi[i,j] <- covi[i,j]*(rho^(abs(i-j)-1))
      }
    }
  }
  return(as.matrix(sig2*covi))
}
```

```
N <- 50
Tiempo <- 4
gamma <- 0.25
sig2 <- 1/gamma
rho <- 1
phi <- 0.5
beta_simulacion <- c(1,2)
Cov_matriz <- generarCov(Tiempo = Tiempo, rho = rho, sig2 = sig2, phi = phi)
x <- matrix(rep(c(1:Tiempo),N), ncol=Tiempo, nrow = N, byrow=T)
x_2 <- x^2

vector_media <- beta_simulacion[1] * x[1,] + beta_simulacion[2] *  x_2[1,]

y <- MASS::mvrnorm(N, vector_media, Cov_matriz)
```

Parametrización del modelo en OpenBUGS

Tal como se ilustró en el apartado que habla de la parametrización de los modelos en OpenBUGS implica la definición de una entidad llamada modelo, la cual, contiene los tres componentes principales del modelo en OpenBUGS: la función de verosimilitud, las distribuciones a priori y los otros calculos necesarios para llevar a cabo la estimación del modelo. En la primera parte se declara la función de verosimilitud la cual corresponde a la función normal multivariada. En la segunda parte se declaran las distribuciones a priori de los parámetros. Finalmente, se actualizan calculos necesarios para llevar a cabo la estimación del modelo, las cuales en este caso, corresponden a calcular e invertir la matriz de covarianzas.

```
    model
```

```
{
    for (i in 1:N)
    {
        y[i,1:Tiempo] ~ dmnorm(media[i,1:Tiempo],precision[1:Tiempo,1:Tiempo])
        for(t in 1:Tiempo){
            media[i,t] <- beta_1*x[i,t] + beta_2 * x_2[i,t]
        }
    }

    beta_1 ~ dnorm(0,100)
    beta_2 ~ dnorm(0,100)
    gamma~ dgamma(3,2)
    rho ~ dunif(0,1)

    sigma_2 <- 1/gamma

    for(c in 1:Tiempo){
    for (f in 1:Tiempo){
        covarianza_Y[c,f] <- (1- step(abs(c-f)-1))*sigma_2 +
                             step(abs(c-f)-1)*sigma_2*rho
    }
    }
    precision[1:Tiempo,1:Tiempo] <- inverse(covarianza_Y[1:Tiempo,1:Tiempo])
}
```

Parametrización del modelo en Stan

Tal como fue presentado anteriormente, la parametrización de los modelos en Stan es más compleja que en OpenBUGS. En este caso, la parametrización del modelo implica primero una declaración de la variables a utilizar, tal como lo muestra el siguiente fragmento de código en donde se declara la cantidad de individuos o unidades observacionales N, la cantidad de observaciones para todos lo individuos T, las cuales en este caso son las mismas para cada uno de los individuos. Adicionalmente se declara la variable independiente Y y las dos matrices de covariables X y X_2 las cuales son declaradas como matrices de tamaño $N * T$.

```
data{
    int<lower=0> N;
    int<lower=0> T;
    matrix[N,T] y;
    matrix[N,T] X;
    matrix[N,T] X_2;
```

```
}
```

Una vez declaradas las variables a utilizar, en la sección parámetros se declaran los parámetros a estimar, en este caso, se declara el conjunto de parámetros β para la estructura de la media y los parámetros γ y ρ de la estructura de covarianza.

```
parameters {
    real beta_1;
    real beta_2;
    real gamma;
    real<lower=0, upper=1> rho;
}
```

Después de declarar los parámetros se llevan a cabo todas las transformaciones necesarias sobre los mismos, en este caso, consiste en llevar a cabo el calculo de la matriz de covarianzas.

```
transformed parameters{
    real sigma_2 = 1/gamma;
    matrix[T,T] varianza;
    matrix[N,T] media;
    for (i in 1:N){
        for(t in 1:T){
            media[i,t] = beta_1*X[i,t] + beta_2 * X_2[i,t];
           }
    }

    for (i in 1:T){
        for(t in 1:T){
            if (i == t){
                varianza[i,t] = sigma_2;
            } else {
                varianza[i,t] = sigma_2*rho;
            }
           }
        }
}
```

Finalmente, se lleva a cabo la declaración del modelo, el cual, en este caso al igual que en el caso de OpenBUGS la decalración del modelo consiste en asignar las distribuciones a priori y la función de verosimilitud.

```
model{

    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
    gamma ~ gamma(1,1);
    rho ~ uniform(0,1);
    for (i in 1:N){
            y[i, 1:T] ~ multi_normal_lpdf(media[i,1:T], varianza);
        }
}
```

La tabla **4-1** muestra la salida de la estimación para el modelo en OpenBUGS, en ella se aprecia que en efecto, los valores estimados para los 4 parámetros son bastante cercanos a los valores reales. En la tabla **4-2** se ve una correlación alta entre los parámetros β_1 y β_2 y entre los parámetros γ y ρ.

Tabla 4-1.: Resumen de Estimación modelo de simetría compuesta

	mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	Rhat	n.eff
β_1	0.56	0.09	0.39	0.50	0.56	0.63	0.74	1.00	2400.00
β_2	2.06	0.03	2.01	2.04	2.06	2.08	2.11	1.00	2100.00
γ	0.26	0.03	0.19	0.23	0.26	0.28	0.32	1.00	20000.00
ρ	0.51	0.07	0.38	0.46	0.51	0.56	0.64	1.00	7000.00
deviance	784.84	7.13	773.80	779.70	783.80	789.00	801.40	1.00	20000.00

Tabla 4-2.: Correlación a posteriori Modelo SC

	β_1	β_2	γ	ρ
β_1	1			
β_2	-0.78	1		
γ	0.15	0.09	1	
ρ	-0.15	0.01	-0.64	1

Ejecución del modelo desde R

La ejecución del modelo desde R se lleva a cabo tal como lo muestra el siguiente fragmento de código.

```
data=list("N","Tiempo", "x", "x_2", "y")
```

```
modelo_SC = R2OpenBUGS::bugs(data,
                  inits = NULL,
                  model.file="sc_cuadratica.txt",
                  parameters= c("beta_1","beta_2" , "gamma","rho"),
                  n.chains = 2,
                  n.iter=60000
)

modelo_sc <- rstan::stan(file = 'sc_cuadratica.stan',
                      init = 0,
                      data = data_x,
                      iter = 100, pars = c('beta_1', 'beta_2','gamma', 'rho', 'log_lik'
                      chains = 1)
```

4.3.2. Simulación Modelo Autorregresivo de Orden 1

Para llevar a cabo la simulación del modelo Autorregresivo de Orden 1 se asume una estructura de covarianzas como la expuesta en la sección 4.2.1, para esto, se generan valores de la matriz de varinazas y covarianzas con $\gamma = 0{,}25$ y $\rho = 0{,}5$ con $\sigma^2 = 1/\gamma$. Como distribuciones a priori se asigna una distribución $U(0,1)$ para el parámetro *rho*. Para el parámetro γ se asume una distribución a priori $\Gamma(3,2)$ y para los parámetros $\beta_i \forall i = 1,2$ se asume una distribución normal $N(0,10^4)$.

La función de verosimilitud en este modelo esta dada por la siguiente expresión:

$$L(\beta,\sigma^2,\rho|Y) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{T}{2}}|\Sigma(\sigma^2,\rho)|^{\frac{1}{2}}} exp[-\frac{1}{2}(y_i - x_i\beta)^T\Sigma(\sigma^2,\rho)^{-1}(y_i - x_i\beta)]$$

Al igual que en el modelo de simetría compuesta y tal como se mostró en las secciones 4.2.1 y 4.2.1 la matriz de varianzas y covarianzas del modelo AR(1) depende de los mismos parámetros que de las del modelo SC. La matriz $\Sigma(\sigma^2,\rho)$ corresponde a la expuesta en la 4.2.1.

El siguiente código de R muestra la simulación de datos para el modelo planteado anteriormente.

```
generarCov <- function(Tiempo, rho, sig2, phi){
  covi <- diag(1,Tiempo)
  covi[covi == 0] <- phi
  for(i in 1:Tiempo){
```

```
    for(j in 1:Tiempo){
      if (i != j) {
        covi[i,j] <- covi[i,j]*(rho^(abs(i-j)-1))
      }
    }
  }
  return(as.matrix(sig2*covi))
}
```

```
N <- 50
Tiempo <- 4
gamma <- 0.25
sig2 <- 1/gamma
rho <- 1
phi <- 0.5
beta_simulacion <- c(1,2)
Cov_matriz <- generarCov(Tiempo = Tiempo, rho = rho, sig2 = sig2, phi = phi)
x <- matrix(rep(c(1:Tiempo),N), ncol=Tiempo, nrow = N, byrow=T)
x_2 <- x^2

vector_media <- beta_simulacion[1] * x[1,] + beta_simulacion[2] *  x_2[1,]

y <- MASS::mvrnorm(N, vector_media, Cov_matriz)
```

Parametrización del modelo en OpenBUGS

Tal como se ilustró en el apartado que habla de la parametrización de los modelos en OpenBUGS implica la definición de una entidad llamada modelo, la cual, contiene los tres componentes principales del modelo en OpenBUGS: la función de verosimilitud, las distribuciones a priori y los otros calculos necesarios para llevar a cabo la estimación del modelo. En la primera parte se declara la función de verosimilitud la cual corresponde a la función normal multivariada. En la segunda parte se declaran las distribuciones a priori de los parámetros. Finalmente, se actualizan calculos necesarios para llevar a cabo la estimación del modelo, las cuales en este caso, corresponden a calcular e invertir la matriz de covarianzas.

```
model
{
     for (i in 1:N)
```

```
        {
        y[i,1:Tiempo] ~ dmnorm(media[i,1:Tiempo],precision[1:Tiempo,1:Tiempo])
        for(t in 1:Tiempo){
        media[i,t] <- beta_1*x[i,t] + beta_2 * x_2[i,t]
        }
        }

        beta_1 ~ dnorm(0,1)
        beta_2 ~ dnorm(0,1)
        gamma~ dgamma(3,2)
        rho ~ dunif(0,1)
        sigma_2 <- 1/gamma
        for(c in 1:Tiempo){
        for (f in 1:Tiempo){
        covarianza_Y[c,f] <- sigma_2*pow(rho, abs(c-f))
        }
        }
        precision[1:Tiempo,1:Tiempo] <- inverse(covarianza_Y[1:Tiempo,1:Tiempo])
}
```

Parametrización del modelo en Stan

En este caso, la parametrización del modelo implica primero una declaración de la variables a utilizar, tal como lo muestra el siguiente fragmento de código en donde se declara la cantidad de individuos o unidades obsevacioneles N, la cantidad de observaciones para todos lo individuos T, las cuales en este caso son las mismas para cada uno de los individuos. Adicionalmente se declara la variable independiente Y y las dos matrices de covariables X y X_2 las cuales son declaradas como matrices de tamaño $N * T$.

```
data{
    int<lower=0> N;
    int<lower=0> T;
    matrix[N,T] y;
    matrix[N,T] X;
    matrix[N,T] X_2;
}
```

Una vez declaradas las variables a utilizar, en la sección parámetros se declaran los parámetros a estimar, en este caso, se declara el conjunto de parámetros β para la estructura de la media y los parámetros γ y ρ de la estructura de covarianza.

```
parameters {
```

```
    real beta_1;
    real beta_2;
    real gamma;
    real<lower=0, upper=1> rho;
}
```

Una vez declaradas las variables a utilizar, en la sección parámetros se declaran los parámetros a estimar, en este caso, se declara el conjunto de parámetros β para la estructura de la media y los parámetros γ y ρ de la estructura de covarianza. Despues de declarar los parámetros se llevan a cabo todas las transformaciones necesarias sobre los mismos, en este caso, consiste en llevar a cabo el calculo de la matriz de covarianzas.

```
transformed parameters{
    real sigma_2 = 1/gamma;
    matrix[T,T] varianza;
    matrix[N,T] media;
    for (i in 1:T){
        for(t in 1:T){
            if (i == t){
                varianza[i,t] = sigma_2;
            } else {
                varianza[i,t] = sigma_2*pow(rho, abs(i-t));
            }
           }
        }
    for (i in 1:N){
        for(t in 1:T){
            media[i,t] = beta_1*X[i,t] + beta_2 * X_2[i,t];
            }
       }
}
```

Finalmente, se lleva a cabo la declaración del modelo, el cual, en este caso al igual que en el caso de OpenBUGS la decalración del modelo consiste en asignar las istribuciones a priori y la función de verosimilitud.

```
model{
    real lpa[N];
    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
    gamma ~ gamma(3,2);
```

```
        rho ~ uniform(0,1);


        for (i in 1:N){
            y[i, 1:T] ~  multi_normal_lpdf(media[i,1:T], varianza);
            }

    }
```

Ejecución del modelo desde R

La ejecución del modelo desde R se lleva a cabo tal como lo muestra el siguiente fragmento de código.

```
data=list("N","Tiempo", "x","x_2" ,"y")

modelo_AR_1 = R2OpenBUGS::bugs(data,
                    inits = NULL,
                    model.file="ar1__cuadratica.txt",
                    parameters= c("beta_1", "beta_2", "gamma","rho"),
                    n.chains = 2,
                    n.iter= 100000)


modelo_ar_1 <- rstan::stan(file = 'ar_1_cuadratica.stan',
                           data = data_x,
                           iter = 10000, pars = c('beta_1', 'beta_2','gamma', 'rho'),
                           chains = 4)
```

La tabla **4-3** muestra la salida de la estimación para el modelo en OpenBUGS, en ella se aprecia que en efecto, los valores estimados para los 4 parámetros son bastante cercanos a los valores reales. En la tabla **4-4** se ve una correlación alta entre los parámetros β_1 y β_2 y entre los parámetros γ y ρ.

4.3.3. Simulación Modelo Antedependiente Estructurado de Orden 1

Para llevar a cabo la simulación del modelo Antedependiente de Orden 1 se asume una estructura de covarianzas como la expuesta en la sección 4.2.1, para esto, se generan valores de la matriz de varianzas y covarianzas con $\gamma = 0{,}25$ y $\rho = 0{,}5$ con $\sigma^2 = 1/\gamma$ y $lambda = -0{,}5$.

	mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	Rhat	n.eff
β_1	1.30	0.23	0.86	1.14	1.30	1.45	1.76	1.01	340.00
β_2	1.91	0.06	1.79	1.87	1.91	1.96	2.03	1.01	470.00
γ	0.29	0.03	0.23	0.27	0.29	0.31	0.35	1.00	1800.00
ρ	0.39	0.07	0.25	0.34	0.39	0.44	0.52	1.00	6200.00
deviance	797.87	2.85	794.30	795.80	797.20	799.20	805.20	1.00	5900.00

Tabla 4-3.: Resumen de Estimación modelo Autorregresivo de orden 1

	β_1	β_2	γ	ρ
β_1	1			
β_2	-0.96	1		
γ	0.01	-0.01	1	
ρ	0.01	-0.01	-0.41	1

Tabla 4-4.: Correlación a posteriori

Como distribuciones a priori se asigna una distribución $U(0,1)$. Para el parámetro γ se asume una distribución a priori $\Gamma(3,2)$ y para los parámetros $\beta_i \forall i = 1,2$ se asume una distribución normal $N(0,10^4)$.

El siguiente fragmento de código muestra la simulación de los datos del modelo en R.

```
N <- 50
Tiempo <- 4
gamma <- 0.25
sig2 <- 1/gamma
lambda_sim <- - 0.5
rho = 0.5
beta_simulacion <- c(1,2)

generar_cov_ante_1 = function(lambda, rho, tiempo,sigma_2){

  tiempos <- rep(1:(tiempo-1))
  if(lambda != 0){
    rhos <- rho^(((tiempos ^lambda)-1/lambda) - (((tiempos + 1)^lambda)-1/lambda))
  } else {
    rhos <- log(tiempos)
  }
```

```
  cov <- matrix(0, tiempo, tiempo)
  cov[1,1]<-1
  cov[1,2]<-rhos[1]
  cov[1,3]<-rhos[1]*rhos[2]
  cov[1,4]<-rhos[1]*rhos[2]*rhos[3]
  cov[2,1]<-rhos[1]
  cov[2,2]<-1
  cov[2,3]<-rhos[2]
  cov[2,4]<-rhos[2]*rhos[3]
  cov[3,1]<-rhos[1]*rhos[2]
  cov[3,2]<-rhos[2]
  cov[3,3]<-1
  cov[3,4]<-rhos[3]
  cov[4,1]<-rhos[1]*rhos[2]*rhos[3]
  cov[4,2]<-rhos[2]*rhos[3]
  cov[4,3]<-rhos[3]
  cov[4,4]<-1


  cov <- cov*sigma_2

  return(cov)

}

Cov_matriz <- generar_cov_ante_1(lambda_sim, rho, Tiempo, sig2)

x <- matrix(rep(c(1:Tiempo),N), ncol=Tiempo, nrow = N, byrow=T)
x_2 <- x^2
vector_media <- beta_simulacion[1] * x[1,] + beta_simulacion[2] *  x_2[1,]
y <- mvrnorm(N, vector_media, Cov_matriz)
```

Parametrización del modelo en OpenBUGS

```
    model
    {
        for (i in 1:N)
        {
        for(t in 1:Tiempo){
```

```
        media[i,t] <- beta_1*x[i,t] + beta_2 * x_2[i,t]
      }
    y[i,1:Tiempo] ~ dmnorm(media[i,1:Tiempo],precision[1:Tiempo,1:Tiempo])
    }
    beta_1 ~ dnorm(0,0.01)
    beta_2 ~ dnorm(0,0.01)
    gamma ~ dgamma(3,2)
    rho ~ dunif(0,1)
    lambda ~ dunif(-1,1)
    sigma_2 <- 1/gamma

    rho_1 <- pow(rho,  ((pow(2, lambda)-1)/(lambda) -
             (pow(1,lambda)-1)/(lambda)))
    rho_2 <- pow(rho,  ((pow(3, lambda)-1)/(lambda) -
             (pow(2,lambda)-1)/(lambda)))
    rho_3 <- pow(rho , ((pow(4, lambda)-1)/(lambda) -
             (pow(3,lambda)-1)/(lambda)))

    covarianza_Y[1,1]  <- sigma_2 * 1
    covarianza_Y[1,2]  <- sigma_2 * rho_1
    covarianza_Y[1,3]  <- sigma_2 * rho_1 *rho_2
    covarianza_Y[1,4]  <- sigma_2 * rho_1*rho_2*rho_3
    covarianza_Y[2,1]  <- sigma_2 * rho_1
    covarianza_Y[2,2]  <- sigma_2 * 1
    covarianza_Y[2,3]  <- sigma_2 * rho_2
    covarianza_Y[2,4]  <- sigma_2 * rho_2*rho_3
    covarianza_Y[3,1]  <- sigma_2 * rho_1*rho_2
    covarianza_Y[3,2]  <- sigma_2 * rho_2
    covarianza_Y[3,3]  <- sigma_2 * 1
    covarianza_Y[3,4]  <- sigma_2 * rho_3
    covarianza_Y[4,1]  <- sigma_2 * rho_1*rho_2*rho_3
    covarianza_Y[4,2]  <- sigma_2 * rho_2*rho_3
    covarianza_Y[4,3]  <- sigma_2 * rho_3
    covarianza_Y[4,4]  <- sigma_2 * 1

    precision[1:Tiempo,1:Tiempo] <- inverse(covarianza_Y[1:Tiempo,1:Tiempo])
}
```

El anterior fragmento de código muestra los cálculos necesarios para realizar la estimación del modelo ADS(1) en OpenBugs. La estructra del código es la misma que en los ejemplos

anteriores, y lo único que cambia es el cálculo de la matriz de covarianzas, la cual, en este caso depende de dos parámetros σ^2, ρ y λ.

$$L(\beta, \sigma^2, \rho, \lambda|Y) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{T}{2}}|\Sigma(\sigma^2, \rho, \lambda)|^{\frac{1}{2}}} exp[-\frac{1}{2}(y_i - x_i\beta)^{\Sigma}(\sigma^2, \rho, \lambda)^{-1}(y_i - x_i\beta)]$$

Parametrización del modelo en Stan

Al igual que en los casos anteriores, en el sigueinte fragmento de código se declara la cantidad de individuos o unidades observacionales N, la cantidad de observaciones para todos lo individuos T, las cuales en este caso son las mismas para todos. Adicionalmente se declara la variable independiente Y y las dos matrices de covariables X y X_2 las cuales son declaradas como matrices de tamaño $N * T$.

```
data{
    int<lower=0> N;
    int<lower=0> T;
    matrix[N,T] y;
    matrix[N,T] X;
    matrix[N,T] X_2;
}
```

Una vez declaradas las variables a utilizar, en la sección parámetros se declaran los parámetros a estimar, en este caso, se declara el conjunto de parámetros β para la estructura de la media y los parámetros γ, ρ, *phi* y *lambda* de la estructura de covarianza.

```
parameters {
    real beta_1;
    real beta_2;
    real gamma;
    real<lower=0, upper=1> rho;
    real<lower=0, upper=1> phi;
    real lambda;
}
```

Despues de declarar los parámetros se llevan a cabo todas las transformaciones necesarias sobre los mismos, en este caso, consiste en llevar a cabo el calculo de la matriz de covarianzas y los vectores de medias.

```
transformed parameters{
    real sigma_2 = exp(gamma);
    real rho_1;
    real rho_2;
```

```
    real rho_3;

    matrix[T,T] varianza;
    matrix[N,T] media;

    for (i in 1:N){
        for(t in 1:T){
            media[i,t] = beta_1*X[i,t] + beta_2 * X_2[i,t];
            }
        }

    rho_1 = pow(rho,  ((pow(2, lambda)-1)/(lambda) - (pow(1,lambda)-1)/(lambda)));
    rho_2 = pow(rho,  ((pow(3, lambda)-1)/(lambda) - (pow(2,lambda)-1)/(lambda)));
    rho_3 = pow(rho , ((pow(4, lambda)-1)/(lambda) - (pow(3,lambda)-1)/(lambda)));

    varianza[1,1]  = sigma_2 * 1;
    varianza[1,2]  = sigma_2 * rho_1;
    varianza[1,3]  = sigma_2 * rho_1 *rho_2;
    varianza[1,4]  = sigma_2 * rho_1*rho_2*rho_3;
    varianza[2,1]  = sigma_2 * rho_1;
    varianza[2,2]  = sigma_2 * 1;
    varianza[2,3]  = sigma_2 * rho_2;
    varianza[2,4]  = sigma_2 * rho_2*rho_3;
    varianza[3,1]  = sigma_2 * rho_1*rho_2;
    varianza[3,2]  = sigma_2 * rho_2;
    varianza[3,3]  = sigma_2 * 1;
    varianza[3,4]  = sigma_2 * rho_3;
    varianza[4,1]  = sigma_2 * rho_1*rho_2*rho_3;
    varianza[4,2]  = sigma_2 * rho_2*rho_3;
    varianza[4,3]  = sigma_2 * rho_3;
    varianza[4,4]  = sigma_2 * 1;

}
```

Finalmente, se lleva a cabo la declaración del modelo, el cual, en este caso al igual que en el caso de OpenBUGS la decalración del modelo consiste en asignar las distribuciones a priori y la función de verosimilitud.

```
model{
    beta_1 ~ normal(0,100);
    beta_2 ~ normal(0,100);
```

```
        gamma ~ gamma(3,2);
        rho ~ uniform(0,1);
        phi ~ uniform(-1,1);
        lambda ~ uniform(-1,1);
        for (i in 1:N){
            y[i, 1:T] ~ multi_normal_lpdf(media[i,1:T], varianza);
            }
    }
```

Ejecución del modelo desde R

La ejecución del modelo desde R se lleva a cabo tal como lo muestra el siguiente fragmento de código.

```
data=list("N", "Tiempo", "x", "x_2", "y")

modelo_ante_1 = R2OpenBUGS::bugs(data,
                     inits = NULL,
                     model.file="ante_1_1_cuadratica.txt",
                     parameters= c("beta_1", "beta_2","gamma","rho", "lambda"),
                     n.chains = 1,
                     n.iter=100000
)
```

```
modelo_ante_1 <- rstan::stan(file = 'ante_1_cuadratica.stan',
                                data = data_x,
                                chains = 1,
                                iter = 10000,
                                pars = c('beta_1', 'beta_2','gamma', 'rho', 'lambda'),
                                control = list(adapt_delta = 0.95)
                                 )
```

La tabla **4-5** muestra la salida de la estimación para el modelo en OpenBUGS, en ella se aprecia que en efecto, los valores estimados para los 4 parámetros son bastante cercanos a los valores reales. En la tabla **4-7** se ve una correlación alta entre los parámetros β_1 y β_2 y entre los parámetros γ y ρ y λ y ρ.

Tabla 4-5.: Resumen de Estimación modelo Antedependiente de orden 1

	mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	Rhat	n.eff
β_1	1.05	0.19	0.67	0.92	1.05	1.18	1.42	1.00	24000.00
β_2	1.99	0.04	1.92	1.96	1.99	2.01	2.06	1.00	23000.00
γ	0.26	0.04	0.18	0.23	0.26	0.29	0.35	1.00	49000.00
ρ	0.67	0.08	0.50	0.62	0.68	0.73	0.81	1.00	28000.00
λ	-0.44	0.28	-0.93	-0.65	-0.45	-0.25	0.15	1.00	32000.00
deviance	604.25	3.10	600.10	602.00	603.60	605.80	611.90	1.00	44000.00

Tabla 4-6.: Correlación a posteriori Modelo Antedependiente de Orden 1

	β_1	β_2	γ	ρ	λ
β_1	1				
β_2	-0.95	1			
γ	0.15	-0.13	1		
ρ	-0.13	0.1	-0.49	1	
λ	-0.01	-0.01	0.03	0.74	1

Tabla 4-7.: Correlación a posteriori

4.3.4. Simulación Modelo Antedependiente Estructurado de Orden 2

Para llevar a cabo la simulación del modelo Antedependiente de Orden 2 se asume una estructura de covarianzas como la expuesta en la sección 4.2.1, para esto, se generan valores de la matriz de varianzas y covarianzas con $\gamma = 0{,}25$, $\rho = 0{,}5$ con $\sigma^2 = 1/\gamma$, $\lambda = -0{,}5$ y $\delta = -0{,}5$ con $\phi = exp(delta)$. Como distribuciones a priori se asigna una distribución $U(0,1)$. Para el parámetro γ se asume una distribución a priori $\Gamma(3,2)$ y para los parámetros $\beta_i \forall i = 1,2$ se asume una distribución normal $N(0,10^4)$.

El siguiente fragmento de código muestra la simulación de los datos en R.

```
N <- 500
Tiempo <- 4
gamma <- 0.25
sig2 <- 1/gamma
lambda_sim <- -0.5
rho = 0.5
beta_simulación <- c(1,2)
delta <- -0.5
```

```
phi <- exp(delta)


generar_cov_ante_2 = function(lambda, rho, tiempo,sigma_2, phi){

  tiempos <- rep(1:(tiempo-1))
  if(lambda != 0){
    rhos <- rho^( (((tiempos)^lambda)-1/lambda) - (((tiempos+1)^lambda)-1/lambda))
  } else {
    rhos <- log(tiempos)
  }

  print(rhos)
  cov <- matrix(0, tiempo, tiempo)
  cov[1,1]<-1
  cov[1,2]<-sqrt(phi)*rhos[1]
  cov[1,3]<-sqrt(phi)*rhos[1]*rhos[2]
  cov[1,4]<-sqrt(phi)*rhos[1]*rhos[2]*rhos[3]
  cov[2,1]<-sqrt(phi)*rhos[1]
  cov[2,2]<-1
  cov[2,3]<-sqrt(phi)*rhos[2]
  cov[2,4]<-sqrt(phi)*rhos[2]*rhos[3]
  cov[3,1]<-sqrt(phi)*rhos[1]*rhos[2]
  cov[3,2]<-sqrt(phi)*rhos[2]
  cov[3,3]<-1
  cov[3,4]<-sqrt(phi)*rhos[3]
  cov[4,1]<-sqrt(phi)*rhos[1]*rhos[2]*rhos[3]
  cov[4,2]<-sqrt(phi)*rhos[2]*rhos[3]
  cov[4,3]<-sqrt(phi)*rhos[3]
  cov[4,4]<-1

  cov <- cov*sigma_2

  return(cov)

}

Cov_matriz <- generar_cov_ante_2(lambda_sim, rho, Tiempo, sig2, phi)
x <- matrix(rep(c(1:Tiempo),N), ncol=Tiempo, nrow = N, byrow=T)
x_2 <- x^2
```

```
vector_media <- beta_simulacion[1] * x[1,] + beta_simulacion[2] *  x_2[1,]
y <- MASS::mvrnorm(N, vector_media, Cov_matriz)
```

Parametrización del modelo en OpenBUGS

```
model
{
    for (i in 1:N)
    {
        y[i,1:Tiempo] ~ dmnorm(media[i,1:Tiempo],precision[1:Tiempo,1:Tiempo])
        for(t in 1:Tiempo){
            media[i,t] <- beta_1*x[i,t] + beta_2 * x_2[i,t]
        }
    }

    beta_1 ~ dnorm(0,0.01)
    beta_2 ~ dnorm(0,0.01)
    gamma ~ dgamma(3,2)
    rho ~ dunif(0,1)
    lambda ~ dunif(-1 ,1)
    phi ~ dunif(0,1)
    sigma_2 <- 1/gamma

    rho_1 <- pow(rho,  ((pow(2, lambda)-1)/(lambda) -
             (pow(1,lambda)-1)/(lambda)))
    rho_2 <- pow(rho,  ((pow(3, lambda)-1)/(lambda) -
             (pow(2,lambda)-1)/(lambda)))
    rho_3 <- pow(rho , ((pow(4, lambda)-1)/(lambda) -
             (pow(3,lambda)-1)/(lambda)))

    covarianza_Y[1,1]  <- sigma_2 * 1
    covarianza_Y[1,2]  <- sqrt(phi) * sigma_2 * rho_1
    covarianza_Y[1,3]  <- sqrt(phi) * sigma_2 * rho_1 *rho_2
    covarianza_Y[1,4]  <- sqrt(phi) * sigma_2 * rho_1*rho_2*rho_3
    covarianza_Y[2,1]  <- sqrt(phi) * sigma_2 * rho_1
    covarianza_Y[2,2]  <- sigma_2 * 1
    covarianza_Y[2,3]  <- sqrt(phi) * sigma_2 * rho_2
    covarianza_Y[2,4]  <- sqrt(phi) * sigma_2 * rho_2*rho_3
    covarianza_Y[3,1]  <- sqrt(phi) * sigma_2 * rho_1*rho_2
```

```
    covarianza_Y[3,2]  <- sqrt(phi) * sigma_2 * rho_2
    covarianza_Y[3,3]  <- sigma_2 * 1
    covarianza_Y[3,4]  <- sqrt(phi) * sigma_2 * rho_3
    covarianza_Y[4,1]  <- sqrt(phi) * sigma_2 * rho_1*rho_2*rho_3
    covarianza_Y[4,2]  <- sqrt(phi) * sigma_2 * rho_2*rho_3
    covarianza_Y[4,3]  <- sqrt(phi) * sigma_2 * rho_3
    covarianza_Y[4,4]  <- sigma_2 * 1

    precision[1:Tiempo,1:Tiempo] <- inverse(covarianza_Y[1:Tiempo,1:Tiempo])
}
```

En el caso del modelo ADS(2) la matriz de varianzas y covarianzas depende de 4 parámetrosσ^2, ϕ, ρ y λ. La estructura del código es la misma que en los ejemplos anteriores, la siguiente ecuación muestra la función de verosimilitud para este modelo, en ella, el cambio principal es el de la matriz de varianzas y covarianzas $\Sigma(\sigma^2, \rho, \phi, \lambda)$.

$$L(\beta, \sigma^2, \rho, \phi, \lambda|Y) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{T}{2}}|\Sigma(\sigma^2, \rho, \phi, \lambda)|^{\frac{1}{2}}} exp[-\frac{1}{2}(y_i - x_i\beta)^{\Sigma}(\sigma^2, \rho, \phi, \lambda)^{-1}(y_i - x_i\beta)]$$

Parametrización del modelo en Stan

Al igual que en los casos anteriores, en el sigueinte fragmento de código se declara la cantidad de individuos o unidades obsevacioneles N, la cantidad de observaciones para todos lo individuos T, las cuales en este caso son las mismas para todos. Adicionalmente se declara la variable independiente Y y las dos matrices de covariables X y X_2 las cuales son declaradas como matrices de tamaño $N * T$.

```
data{
    int<lower=0> N;
    int<lower=0> T;
    matrix[N,T] y;
    matrix[N,T] X;
    matrix[N,T] X_2;
}
```

Una vez declaradas las variables a utilizar, en la sección parámetros se declaran los parámetros a estimar, en este caso, se declara el conjunto de parámetros β para la estructura de la media y los parámetros γ, ρ, *phi* y *lambda* de la estructura de covarianza.

```
parameters {
    real beta_1;
    real beta_2;
```

```
    real gamma;
    real rho;
    real phi;
    real lambda;
}
```

Despues de declarar los parámetros se llevan a cabo todas las transformaciones necesarias sobre los mismos, en este caso, consiste en llevar a cabo el calculo de la matriz de covarianzas y los vectores de medias.

```
transformed parameters{
    real sigma_2 = 1/gamma;
    real rho_1;
    real rho_2;
    real rho_3;
    matrix[N,T] media;
    matrix[T,T] varianza;
    for (i in 1:N){
        for(t in 1:T){
            media[i,t] = beta_1*X[i,t] + beta_2 * X_2[i,t];
            }
    }
    rho_1 = pow(rho,  ((pow(2, lambda)-1)/(lambda) -
            (pow(1,lambda)-1)/(lambda)));
    rho_2 = pow(rho,  ((pow(3, lambda)-1)/(lambda) -
            (pow(2,lambda)-1)/(lambda)));
    rho_3 = pow(rho , ((pow(4, lambda)-1)/(lambda) -
            (pow(3,lambda)-1)/(lambda)));

    varianza[1,1]  = sigma_2 ;
    varianza[1,2]  = sqrt(phi) * sigma_2 * rho_1;
    varianza[1,3]  = sqrt(phi) * sigma_2 * rho_1 *rho_2;
    varianza[1,4]  = sqrt(phi) * sigma_2 * rho_1*rho_2*rho_3;
    varianza[2,1]  = sqrt(phi) * sigma_2 * rho_1;
    varianza[2,2]  = sigma_2;
    varianza[2,3]  = sqrt(phi) * sigma_2 * rho_2;
    varianza[2,4]  = sqrt(phi) * sigma_2 * rho_2*rho_3;
    varianza[3,1]  = sqrt(phi) * sigma_2 * rho_1*rho_2;
    varianza[3,2]  = sqrt(phi) * sigma_2 * rho_2;
    varianza[3,3]  = sigma_2;
    varianza[3,4]  = sqrt(phi) * sigma_2 * rho_3;
    varianza[4,1]  = sqrt(phi) * sigma_2 * rho_1*rho_2*rho_3;
```

```
        varianza[4,2]  = sqrt(phi) * sigma_2 * rho_2*rho_3;
        varianza[4,3]  = sqrt(phi) * sigma_2 * rho_3;
        varianza[4,4]  = sigma_2;
    }
```

Finalmente, se lleva a cabo la declaración del modelo, el cual, en este caso al igual que en el caso de OpenBUGS la decalración del modelo consiste en asignar las distribuciones a priori y la función de verosimilitud.

```
    model{
        beta_1 ~ normal(0,100);
        beta_2 ~ normal(0,100);
        gamma ~ gamma(3,2);
        rho ~ uniform(0,1);
        phi ~ uniform(-1,1);
        lambda ~ uniform(-1,1);
        for (i in 1:N){
            y[i, 1:T]  ~ multi_normal_lpdf(media[i,1:T], varianza);
            }
    }
```

Ejecución del modelo desde R

La ejecución del modelo desde R se lleva a cabo tal como lo muestra el siguiente fragmento de código.

```
data_x = list(y=y, N=N, T=Tiempo, X = x, X_2 = x_2 )

modelo_ante_2 <- rstan::stan(file = 'ante_2_cuadratica_prueba.stan',
                             init = 0.5,
                             data = data_x,
                             iter = 1000,
                             pars = c('beta_1', 'beta_2','gamma',
                                      'rho', 'lambda', 'phi'),
                             chains = 1)

modelo_ante_2 = R2OpenBUGS::bugs(data,
                     inits = NULL,
                     model.file="ante_2_1_cuadratica_prueba.txt",
```

```
                    parameters= c("beta_1", "beta_2","gamma","rho", "lambda",
"phi"),
                    n.chains = 1,
                    n.iter=100
)
```

La tabla **4-8** muestra la salida de la estimación para el modelo en OpenBUGS, en ella se aprecia que en efecto, los valores estimados para los 4 parámetros son bastante cercanos a los valores reales. En la tabla **4-7** se ve una correlación alta entre los parámetros β_1 y β_2 y entre los parámetros γ y ρ y λ y ρ.

Tabla 4-8.: Resumen de Estimación modelo Antedependiente de Orden 2

	mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	Rhat	n.eff
β_1	0.83	0.24	0.37	0.67	0.83	0.99	1.30	1.00	7300.00
β_2	2.03	0.06	1.93	2.00	2.04	2.07	2.14	1.00	7600.00
γ	0.23	0.03	0.17	0.21	0.23	0.25	0.30	1.00	12000.00
ρ	0.92	0.06	0.77	0.89	0.93	0.97	1.00	1.00	6000.00
λ	-0.44	0.28	-0.91	-0.64	-0.45	-0.21	0.11	1.00	8100.00
ϕ	0.52	0.11	0.33	0.44	0.51	0.59	0.77	1.00	2300.00
deviance	772.91	3.30	768.40	770.50	772.30	774.60	781.00	1.00	65000.00

Tabla 4-9.: Correlación a posteriori modelo Antedependiente de Orden 2

	beta_1	beta_2	gamma	rho	lambda	phi
beta_1	1					
beta_2	-0.9586	1				
gamma	-0.0032	0.0034	1			
rho	0.0062	-0.0093	-0.2136	1		
lambda	0.0077	-0.0084	-0.0229	0.4421	1	
phi	0.0053	-0.0015	-0.3384	-0.6421	-0.2479	1

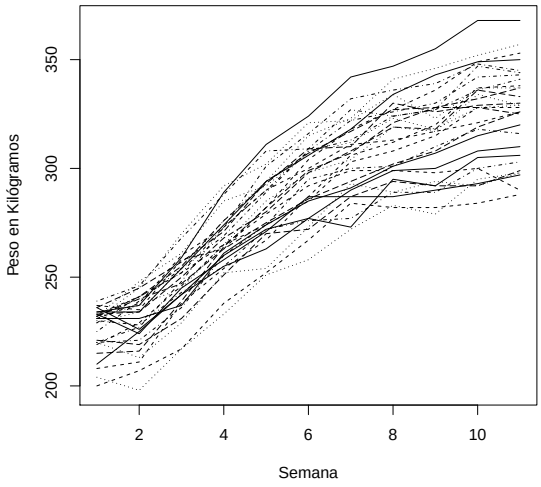
4.3.5. Aplicacion: Datos de Crecimiento Vacuno

En esta sección se presentará un ejemplo de aplicación de datos reales, para esto se utilizará el conjunto de datos de crecimiento de ganado vacuno de [Kenward, 1987] disponibles en [Zimmerman & Nunez, 2009]. Este conjunto de datos proviene de un experimento en el cual se toma el peso en kilogramos de dos grupos de vacas para evaluar la ganancia de peso de las mismas ante la presencia de un tratamiento A o un tratamiento B para control de

parásitos intestinales. Para este ejemplo se toma solamente el conjunto de datos de las vacas que fueron sometidas al tratamiento A, tal como lo hizo [Cepeda, 2001b].

La gráfica **4-1** muestra las trayectorias de cada una de las observaciones de estas vacas, en esta gráfica se muestra una tendencia creciente de los datos aunque con de crecimientos decrecientes.

Figura 4-1.: Gráfica de las trayectorias de cada uno de los individuos



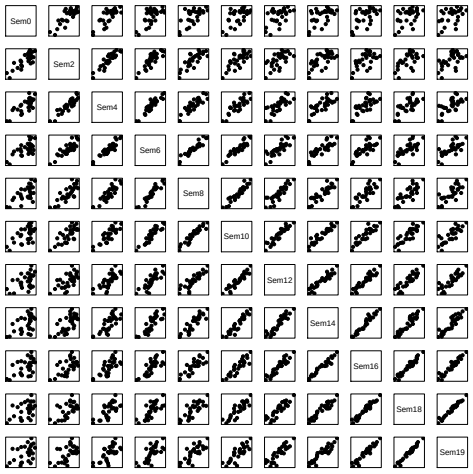
La gráfica **4-2** es un plot de dispersión entre las observaciones de diferentes momentos del tiempo, en ella se ve claramente que la correlación entre las observaciones disminuye a medida que aumenta la distancia temporal de los datos. Por lo anterior, se puede inferir que un modelo con covarianza que disminuye a través del tiempo es bastante apropiado para este tipo de datos.

Siguiendo el trabajo de [Cepeda, 2001b], la siguiente ecuación para modelar la media de las observaciones:

$$\hat{E}(Y_{i,t}) = \beta_0 + \beta_1 * t + \beta_2 * t^2 + \beta_3 * t^3 \tag{4-2}$$

Es decir, el valor esperado es un función cúbica del tiempo, con el vector t definido de la siguiente manera $t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9{,}5)$ donde los vectores t^2 y t^3 son el cuadrado y el cubo de los elementos del vector t respectivamente. Para la matriz de varianzas y covarianzas se utilizará el modelo AR(1) explicado en la sección 4.2.1.

Figura 4-2.: Gráficas de dispersión entre observaciones semanales



Parametrización del modelo en OpenBUGS

Para los datos de crcimiento vacuno, el siguiente es el código del modelo en OpenBUGS. En la primera parte del programa se declara la función de verosimilitud que en este caso corresponde a la función normal multivariada y se calcula el predictor lineal de la media. Posteriormente, se declaran distribuciones a priori normales para los parámetros β, gamma para el parámetro γ y uniforme para el parámetro ρ. Finalmente, se llevan a cabo las transformaciones necesarias para el modelo, las cuales incluyen el calculo de la matriz de covarianzas.

```
model
{
    for (i in 1:N)
    {
    y[i,1:Tiempo] ~ dmnorm(media[i,1:Tiempo],precision[1:Tiempo,1:Tiempo])
        for(t in 1:Tiempo){
            media[i,t] <- beta_0 + beta_1*x[i,t] + beta_2 * x_2[i,t]
                          + beta_3 * x_3[i,t]
        }
    }
    beta_0 ~ dnorm(0,0.001)
    beta_1 ~ dnorm(0,0.001)
    beta_2 ~ dnorm(0,0.001)
```

```
    beta_3 ~ dnorm(0,0.001)
    gamma~ dgamma(3,2)
    rho ~ dunif(0,1)
    sigma_2 <- 1/gamma
    for(c in 1:Tiempo){
    for (f in 1:Tiempo){
    covarianza_Y[c,f] <- sigma_2*pow(rho, abs(c-f))
    }
    }
    precision[1:Tiempo,1:Tiempo] <- inverse(covarianza_Y[1:Tiempo,1:Tiempo])
}
```

Parametrización del modelo en Stan

La programación en Stan del modelo propuesto para los datos de ganado Vacuno, se presenta en el siguiente código. En la sección de datos se declaran cuatro variables X, X_2 y X_3 y la variable dependiente y, cada una como una matriz de tamaño $N * T$ donde N es la cantidad de unidades observacionales y T es la cantidad de observaciones para cada individuo, las cuales, en este caso son las mismas para todos lo individuos. Los parámetros a estimar son el conjunto de parámetros β que confoman el predictor lineal de la media y los parámetros de precisión $gamma_sigma$ y de correlación rho.

Posterior a la declaración de parámetros se lleva a cabo la transformación de los mismos, en este caso, se lleva a cabo el cálculo de los parámetros de varianza $sigma_2$ y el cálculo de la matriz de covarianzas $varianza$ de tamaño $T*T$, además, el predictor lineal de la media.

En la siguiente sección se lleva a cabo la declaración del modelo, en la cual se asignan las distribuciones prior para cada uno de los parámetros y se asigna a cada una de las observaciones de la variable y la función de log verosimilitud de la distribución normal multivariada.

Por último, se lleva a cabo el cálculo de la log verosimilitud y la predicción de la variable dependiente en la sección de cantidades generadas.

```
data{
    int<lower=0> N;
    int<lower=0> T;
    matrix[N,T] y;
    matrix[N,T] X;
```

```
    matrix[N,T] X_2;
    matrix[N,T] X_3;
}
parameters {
    real beta_0;
    real beta_1;
    real beta_2;
    real beta_3;
    real gamma_sigma;
    real rho;
    }
transformed parameters{
    real sigma_2 = 1/gamma_sigma;
    matrix[T,T] varianza;
    matrix[N,T] media;
    for (i in 1:T){
        for(t in 1:T){
            if (i == t){
                varianza[i,t] = sigma_2;
            } else {
                varianza[i,t] = sigma_2*pow(rho, abs(i-j));
            }
           }
        }

    for (i in 1:N){
        for(t in 1:T){
            media[i,t] = beta_0 + beta_1*X[i,t] + beta_2*X_2[i,t] + beta_3*X_3[i,t];
            }
        }
}
model{
    beta_0 ~ normal(200,10);
    beta_1 ~ normal(0,1);
    beta_2 ~ normal(0,1);
    beta_3 ~ normal(0,1);
    gamma_sigma ~ gamma(10,1);
    rho ~ uniform(0,1);
    for (i in 1:N){
        y[i, 1:T] ~ multi_normal_lpdf(media[i,1:T], varianza);
```

```
            }
    }
```

Para ejecutar el modelo anterior desde R se siguen los comandos en el siguiente fragmento de código. La primera parte muestra la importación de datos .La lista data contiene las matrices de la variable dependiente y, el numero de individuos N, la cantidad de observaciones $Tiempo$, la matriz de tiempos x, su cuadrado x_2y su cubo x_3. Finalmente, mediante los comandos $rstan :: stan$ y $R2OpenBUGS :: bugs$ se ejecutan los códigos presentados anteriormente.

```
vacas <- read.csv('./datos/vacas.txt', sep = ";", row.names = NULL)

y <- as.matrix(vacas[,2:ncol(vacas)])
x <- matrix(rep(c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9.5), nrow(vacas)),  ncol = ncol(vacas
-1 , nrow = nrow(vacas), by = 1)
x_2 <- x^2
x_3 <- x^3
N <- nrow(y)
Tiempo <- ncol(y)


data = list(y=y, N=N, T=Tiempo, X = x, X_2 = x_2, X_3 = x_3)


modelo_ar_1_vacas_stan <- rstan::stan(file = 'ar_1_vacas.stan',
                                      iter = 10000,
                                      pars = c('beta_0', 'beta_1', 'beta_2',
                                               'beta_3','gamma_sigma', 'rho'),
                                      data = data,
                                      chains = 4,
                                      control = list(adapt_delta = 0.99)
                                      )

modelo_ar_1_vacas_ob = R2OpenBUGS::bugs(data,
                           inits = NULL,
                           model.file="ar1_vacas.txt",
                           parameters= c("beta_0","beta_1", "beta_2",
                                         "beta_3", "gamma","rho"),
                           n.chains = 2,
                           n.iter=100000)
```

Para ejecutar este mismo modelo desde la intefaz gráfica de OpenBUGS los datos se pueden alimentar, como se mostró anteriormente, a través del editor de texto con la estructura presentada en el siguiente fragmento de código. En este se aprecian los escalares $Tiempo = 11$ y $N = 30$ que muestran la cantidad de observaciones por individuo y la cantidad de individuos. La variable a modelar y se incluye en la matriz del mismo nombre, la cual tiene dimensión $Tiempo * N$, y las matrices x, x_2 y x_3 son las matrices que muestran el tiempo, el tiempo al cuadrado y el tiempo al cubo respectivamente, los cuales, son los regresores a utilizar en este ejemplo.

```
list(
    Tiempo = 11,
    N = 30,
    y = structure(
    .Data = c(233,224,245,258,271,287,287,287,290,293,297,
              231,238,260,273,290,300,311,313,317,321,326,
              232,237,245,265,285,298,304,319,317,334,329,
              239,246,268,288,308,309,327,324,327,336,341,
              215,216,239,264,282,299,307,321,328,332,337,
              236,226,242,255,263,277,290,299,300,308,310,
              219,229,246,265,279,292,299,299,298,300,290,
              231,245,270,292,302,321,322,334,323,337,337,
              230,228,243,255,272,276,277,289,289,300,303,
              232,240,247,263,275,286,294,302,308,319,326,
              234,237,259,289,311,324,342,347,355,368,368,
              237,235,258,263,282,304,318,327,336,349,353,
              229,234,254,276,294,315,323,341,346,352,357,
              220,227,248,273,290,308,322,326,330,342,343,
              232,241,255,276,293,309,310,330,326,329,330,
              210,225,242,260,272,277,273,295,292,305,306,
              229,241,252,265,274,285,303,308,315,328,328,
              204,198,217,233,251,258,272,283,279,295,298,
              220,221,236,260,274,295,300,301,310,318,316,
              233,234,250,268,280,298,308,319,318,336,333,
              234,234,254,274,294,306,318,334,343,349,350,
              200,207,217,238,252,267,284,282,282,284,288,
              220,213,229,252,254,273,293,289,294,292,298,
              225,239,254,269,289,308,313,324,327,347,344,
              236,245,257,271,294,307,317,327,328,328,325,
              231,231,237,261,274,285,291,301,307,315,320,
```

```
            208,211,238,254,267,287,306,312,320,337,338,
            232,248,261,285,292,307,312,323,318,328,329,
            233,241,252,273,301,316,332,336,339,348,345,
            221,219,231,251,270,272,287,294,292,292,299
    ),.Dim = c(N, Tiempo)),
    x = structure(
    .Data = c( 0,1,2,3,4,5,6,7,8,9,9.5,
               0,1,2,3,4,5,6,7,8,9,9.5,
               0,1,2,3,4,5,6,7,8,9,9.5,
               .
               .
               .
               0,1,2,3,4,5,6,7,8,9,9.5
        ),.Dim = c(N, Tiempo)),
    x_2 = structure(
    .Data = c( 0,1,4,9,16,25,36,49,64,81,90.25,
               0,1,4,9,16,25,36,49,64,81,90.25,
               .
               .
               .
               0,1,4,9,16,25,36,49,64,81,90.25
        ),.Dim = c(N, Tiempo)),
    x_3 = structure(
    .Data = c( 0,1,8,27,64,125,216,343,512,729,857.375,
               0,1,8,27,64,125,216,343,512,729,857.375,
               .
               .
               .
               0,1,8,27,64,125,216,343,512,729,857.375
        ),.Dim = c(N, Tiempo)),
)
```

La tabla **4-10** muestra el resultado de la estimación para el modelo planteado anteriormente los cuales están en concordancia con los resultados obtenidos en [Cepeda, 2001b]. En cuanto a la convergencia de las cadenas se ve claramente del indicador *Rhat* que las cadenas alcanzaron la convergencia. Por otra parte, la tabla **4-11** muestra la correlación entre parámetros, la cual, se manifiesta alta entre los grupos de parámetros de la media y los grupos de parámetros de la varianza entre sí.

Tabla 4-10.: Resumen de Estimación modelo Autorregresivo de Orden 1 para Datos de Ganado Vacuno

	mean	sd	2.5 %	25 %	50 %	75 %	97.5 %	Rhat	n.eff
beta_0	224.33	2.96	218.40	222.40	224.40	226.30	230.10	1.00	200000.00
beta_1	8.99	1.23	6.61	8.16	8.99	9.82	11.45	1.00	64000.00
beta_2	1.51	0.31	0.89	1.30	1.51	1.73	2.11	1.00	86000.00
beta_3	-0.14	0.02	-0.18	-0.16	-0.14	-0.13	-0.10	1.00	140000.00
gamma	0.0039	0.0007	0.0027	0.0035	0.0039	0.0044	0.0053	1.00	200000.00
rho	0.91	0.02	0.87	0.90	0.91	0.92	0.94	1.00	200000.00
deviance	2251.20	3.75	2246.00	2248.00	2251.00	2253.00	2260.00	1.00	24000.00

Tabla 4-11.: Correlación a posteriori Parámetros Modelo Ganado Vacuno

	beta_0	beta_1	beta_2	beta_3	gamma	rho
beta_0	1					
beta_1	-0.2257	1				
beta_2	0.0506	-0.8888	1			
beta_3	-0.0278	0.7736	-0.9707	1		
gamma	0.1025	0.0386	-0.0329	0.0236	1	
rho	-0.0924	-0.0588	0.0523	-0.0421	-0.8946	1

5. Conclusiones y recomendaciones

En este trabajo se trataron tanto aspectos teóricos como aplicaciones de Modelos Lienales Doblemente Generalizados. En primera medida se presentaron aspectos teóricos generales como su definición, su relación con los GLM y aspectos teóricos relacionados a la estimación bayesiana en *Stan* y *OpenBUGS*. A manera de ejemplo se trataron algunos modelos que abarcan la generalidad de los requerimientos de modelación como lo son el modelo normal heteroscedástico y el modelo gamma para variables continuas, la regresión beta para datos de proporciones y modelos de datos de conteo basados en la regresión beta binomial y binomial negativa, finalmente, a manera de extensión se desarrolla un modelo para datos longitudinales basado en la distribución normal multivariada con matriz de varianzas y covarianzas estructurada.

Para cada uno de estos modelos se llevo a cabo una explicación teórica que incluyó entre otras aspectos relacionados a la función de distribución del modelo y cuando fue necesario, algunas reparametrizaciones con el fin de lograr mayor interpretabilidad de los parámetros. Específicamente, para los modelos gamma, beta, beta binomial y binomial negativa se hicieron reparametrizaciones para expresar la función de densidad en términos de la media y el parámetro de precisión.

En cuando a la estimación bayesiana se prestaron los aspectos generales para cada uno de los modelos y se llevó a cabo la programación de los modelos en stan y OpenBUGS. Comparativamente, se ha evidenciado que la programación del modelo en Stan aunque más compleja permite mayor flexibilidad. En los casos en que las parametrizaciones de las distribuciones en Stan y OpenBUGs no coincidian con los parámetros de media y dispersión se llevaron a cabo las transformaciones necesarias sobre los predictores lineales.

Finalmente, para cada uno de los modelos presentados anteriormente se realizaron simulaciones y aplicaciones en las cuales, se ilustró la programación en OpenBUGS y Stan. Estas aplicaciones contienen datos de diversas áreas y permiten ilustrar de una manera mucho más clara la utilización de los DGLM a través de estos dos leguajes, los cuales, son unas de las herramientas más populares para estimación bayesiana.

A. Anexo: Principios de estimación bayesiana

Dado el enfoque bayesiano de este documento en este apéndice se presentan los principios fundamentales en los cuales se basa la estimación bayesiana. En la estimación clásica, el enfoque de máxima verosimilitud consiste en encontrar un conjunto de parámetros que maximicen la verosimilitud, es decir, que hagan que los datos sean más verosímiles o creíbles. En este sentido, en el enfoque clásico se considera que toda la información relavante acerca de los parámetros está contenida en los datos, en contraste, la estadística bayesiana considera que los parámetros de interés son realizaciones de variables aleatorias y por lo tanto provienen de alguna distribución. Por lo anterior, en el enfoque bayesiano, también es importante incorporar al modelo información de esta distribución.

Tal como lo indica su nombre, la estadística bayesiana se basa en los postulados de teorema de Bayes. Para un conjunto de datos y y un conjunto de parámetros θ la probabilidad conjunta $p(y, \theta)$ se expresa de la siguiente manera:

$$p(y, \theta) = p(y|\theta)p(\theta) = p(\theta|y)p(y) \tag{A-1}$$

De donde se obtiene la siguiente expresión, la cual, es precisamente el teorema de Bayes:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{A-2}$$

Los componentes de la ecuación A-2 son bastante interesantes en el contexto del modelamiento de datos. Primero, la expresión $p(\theta|y)$ es la probabilidad de un valor determinado de los parámetros dado que se observó un conjunto de datos y, esta distribución es llamada distribución *a posteriori* o distribución *posterior*. Esta distribución tiene mucho sentido intuitivo en el sentido que claramente es la distribución que se espera encontrar ya que en efecto, lo que se quiere encontrar es el valor de los parámetros θ dado que se oberva un conjunto de datos .

La ecuación anterior muestra que la distribución *a posteriori* es el producto de $p(\theta)$ y $p(y|\theta)$. La expresión $p(y|\theta)$ es la probabilidad del conjunto de datos dado un valor θ, es decir, la verosimilitud. A su vez, $p(\theta)$ es la probabilidad de que θ tome un determinado valor, en este

caso, esta función es llamada distribución *prior* o distribución *a priori*.

Por su parte $p(y)$ es llamada *probabilidad de los datos* o *evidencia* y tiene como objetivo estandarizar la distribución $p(\theta|y)$. $p(y)$ puede ser obtenida mediante la marginalización de la distribución $p(y, \theta) = p(y|\theta)p(\theta)$, es decir $p(y) = \int p(y, \theta)\delta\theta = \int p(y|\theta)p(\theta)\delta\theta$, lo cual no es más que $E_\theta[P(y|\theta)]$. Esto muestra que en realidad $p(y)$ puede ser vista como un valor esperado de la verosimilitud inducido por θ [McElreath, 2020].

Uno de los aspectos importantes a considerar en el análisis bayesiano es la distribución $p(\theta)$ o distribución a priori ya que en gran parte es este componente el que enriquece este tipo de análisis. Respecto a la elección de la prior, dos visiones pueden ser consideradas. Primero, la prior puede respresentar las creencias que el investigador posee acerca de los parámetros, es decir, la elección de la prior puede ser una distribución que condense toda la información previa ya sea empírica o teórica que el investigador tiene acerca del comportamiento del parámetro. Segundo, la prior puede ser entendida como un componente más del modelo y por lo tanto, su validez puede ser constatada mediante algún tipo de criterio estadístico. En la práctica, la elección de la prior usualmente involucra un enfoque mixto entre el criterio subjetivo y el enfoque netamente estadístico [McElreath, 2020].

La ecuación A-2 muestra una interacción entre la verosimilitud y la distribución prior que permite obtener la distribución del parámetro θ. Sin embargo, en muchos casos la distribución a posteriori de los parámetros es intratable de manera analítica y es por esto que se acude a los métodos numéricos como una manera de aproximar $p(\theta|y)$.

Entre los métodos disponibles para llevar a cabo la aproximación de la distribución a posteriori destacan los métodos de Markov Chain Monte Carlo (MCMC), a los cuales pertenecen por ejemplo el muestreador de Gibbs, el algoritmo de Metropolis Hastings y el Non U-turn smapler (NUTS). El enfoque general de estos métodos consisten en obtener muestras de la distribución a posteriori repetidamente de manera que a partir de esas muestras se pueden hacer estimaciones sobre los parámetros. Una guia para profundizar en los algoritmos mencionados pueden ser los trabajos de [Chib & Greenberg, 1995, Neal et al., 2011, Hoffman & Gelman, 2014].

B. Anexo: Evaluación de modelos

Una vez obtenidas las muestras de la distribución a posteriori de los parámetros, es decir, una vez estimado el modelo, es necesario llevar a cabo los análisis del ajuste del mismo, en esta sección, se presentan algunas métricas de desempeño de modelo y de comparación de modelos.

Para la evaluación de modelos se consideran algunas medidas de discrepancia entre las predicciones del modelo y los datos observados, medidas de ajuste de valores extremos, el p valor bayesiano y algunas estadísticas para comparación de modelos como lo son el criterio WAICC y el criterio Loo.

Estadísticas de Discrepancia

Entre las estadísticas de discrepancia se encuentran la estadística de la deviance y los residuales de pearson estandarizados. Suponiendo que $y = (y_1, ..., y_n)$ son los valores observados de la variable dependiente y que $E(y_i) = \mu_i$ y $V(y_i) = \sigma_i^2$ se tiene que:

Residual de Pearson Estandarizado:

$$T(y|\theta) = \sum_{i}^{n} \frac{(y_i - \mu_i)^2}{\sigma_i^2} \tag{B-1}$$

Deviance:

$$T(y|\theta) = -2 \sum_{i}^{n} f(y_i|\theta) \tag{B-2}$$

Donde se asume que θ es el verdadero valor del parámetro o una estimación muy precisa del mismo. Es de resaltar que en el contexto de los DGLM, para el caso del residual de pearson estandarizado σ_i^2 es una función del predictor lineal γz_i dado que, como se muestra dependiendo del modelo, este predictor es el predictor para la varianza o para el parámetro de precisión.

P valor Bayesiano

Para el p valor bayesiano la idea es poder comparar $T(y)$ sobre los datos observados, y, con respecto al mismo estadístco con los datos replicados por el modelo, y^{rep}. Con esta comparación la idea es poder estimar la probabilidad de obtener valores consistentemente más extremos.

P valor Bayesiano:

$$P_{value} = P[T(y^{rep}|\theta) > T(y|\theta)] \tag{B-3}$$

La cual se puede calcular mediante $P[.] = \frac{1}{m}\sum_j^m I_{T(y^{rep}|\theta)>T(y|\theta)}$ donde m es la cantidad de muestras de la distribución a posteriori obtenidas. Algunos ejemplos para el uso de este indicador pueden ser:

P valor Bayesiano para el mínimo:
Para el caso del ajuste sobre el extremo inferior de los datos y, el estadístico $T(.)$ puede ser el mínimo, por lo tanto, la probabilidad a estimar es la siguiente:

$$P_{min} = P[min(y^{rep}|\theta) > min(y|\theta)] \tag{B-4}$$

P valor Bayesiano para el máximo:
Para el caso del ajuste sobre el extremo superior de los datos y, el estadístico $T(.)$ puede ser el máximo, por lo tanto, la probabilidad a estimar es la siguiente:

$$P_{max} = P[max(y^{rep}|\theta) > max(y|\theta)] \tag{B-5}$$

P valor Bayesiano para la log verosimilitud:
Para el caso en el que $T(.)$ sea la log verosimilitud sea l la función de log verosimilitud, se tiene que:

$$P_{modelo} = P[l(y^{rep}|\theta) > l(y|\theta)] \tag{B-6}$$

El criterio de decisión para el indicador P_{value} es que cuando alguno de estos valores tienda al extremo de 0 o 1, se puede concluir que existen problemas de ajustes en el modelo ya que consistentemente el estadístico basado en y^{rep} es o mayor o menor que el basado en y.

Validación Cruzada - loo

La validación cruzada es una técnica que permite observar el ajuste del modelo a los datos fuera de la muestra con la que el modelo fue estimado, esto permite una evaluación más

exacta de la capacidad de generalización del modelo. El método implementado en este documento es el método loo que significa *dejar uno afuera* (en inglés *leave one out*). Este método consiste en obtener las estimaciones de los parámetros con una muestra que excluye una observación y luego ver el ajuste del modelo al dato excluido. Este proceso se lleva a cabo de manera repetitiva y al final se obtiene una estimación del ajuste del modelo.

Para el calculo de este estimador en Stan se utiliza la función $rstan :: loo$, la cual, se alimenta con una matriz de log verosimilitud y computa el método basandose en lo propuesto por [Vehtari et al., 2017]. En este caso, el método de cálculo es una aproximación de la siguiente ecuación:

$$l(y|\theta_{-i}) = \sum_{i}^{n} l(y_i|\theta_{-i}) \tag{B-7}$$

Donde $l(y_i|\theta_{-i})$ es la log verosimilitud de y_i dado que θ se estimó sin la observación y_i.

Criterio de Información ampliamente aplicable - WAIC

El WAIC es un criterio de información que permite manejar la incertidumbre en las predicciones muestra a muestra a través del calculo de la varianza de la log verosimilitud de cada observación [McElreath, 2020]. El WAIC se define como:

$$WAIC = -2(lppd - p_{WAIC}) \tag{B-8}$$

Donde lppd es el logaritmo de la densidad predictiva puntual y se define como $lppd = \sum_{i}^{n} log(Pr(y_i))$ con $Pr(y_i)$ siendo la verosimilitud promedio de la i-ésima observación, es decir, lppd es el total a través de las observaciones del logaritmo de la verosimilitud promediada a través de todas las muestras obtenidas. Por su parte $p_{WAIC}) = \sum_{i}^{n} V(y_i)$ siendo $V(y_i)$ la varianza de la logverosimilitud de la i-ésima observación.

Para el calculo de este estimador en Stan se utiliza la función $loo :: waic$, la cual, se alimenta con una matriz de log verosimilitud [Vehtari et al., 2018].

B.0.1. Implementación en Stan

El siguiente fragmento de código muestra la implementación del modelo en Stan con todos los cálculos necesarios para los indicadores anteriormente mencionados para el modelo normal heteroscedástico.

```
data{
    int<lower=0> N;
    real x_1[N];
```

```
    real x_2[N];
    real x_3[N];
    real x_4[N];
    real y[N];
}
transformed data{

//  Calculo de minimo y maximo para uso en p valor bayesiano
real y_min;
real y_max;
y_min = min(y);
y_max = max(y);

}
parameters {
    real beta_1;
    real beta_2;
    real beta_3;
    real gamma_1;
    real gamma_2;
    real gamma_3;
}
transformed parameters{
    real mu[N];
    real sigma[N];
    for (i in 1:N){
        mu[i] = beta_1*x_1[i]+beta_2*x_2[i] + beta_3*x_3[i];
        sigma[i] = sqrt(exp(gamma_1*x_1[i]+gamma_2*x_2[i]+
                   gamma_3*x_4[i]));
         }
     }
model{
    beta_1 ~ normal(0,1000);
    beta_2 ~ normal(0,1000);
    beta_3 ~ normal(0,1000);
    gamma_1 ~ normal(0,1000);
    gamma_2 ~ normal(0,1000);
    gamma_3 ~ normal(0,1000);
    for (i in 1:N){
         y[i] ~ normal_lpdf( mu[i], sigma[i]);
```

```
            }
        }
    generated quantities{
        // Vector que contiene las r\'eplicas
        vector[N] y_test;
        // Vector que contiene la log verosimilitud de los datos originales
        vector[N] log_lik;
        // Vector que contiene la log verosimilitud de los datos replicados
        vector[N] log_lik_test;
        // vector que contiene los residuales de pearson
        vector[N] pearson_res;

        real p_min;
        real p_max;
        real p_model;
        real pearson;
        real deviance;


        for (i in 1:N){
            y_test[i] = normal_rng(mu[i], sigma[i]);
            log_lik[i] = normal_lpdf(y[i] | mu[i], sigma[i]);
            log_lik_test[i] = normal_lpdf(y_test[i] | mu[i], sigma[i]);
            pearson_res[i] = (y_test[i]-mu[i])^2/sigma[i]^2;

        }

        p_min = step(min(y_test)-y_min);
        p_max = step(max(y_test)-y_max);
        p_model = step(sum(log_lik)-sum(log_lik_test));
        pearson = sum(pearson_res);
        deviance = -2*sum(log_lik);

    }
```

Para ejecutar este modelo desde R se ejecuta el siguiente comando:

```
data = list(x_1 = x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, y=y, N=n)

model <- rstan::stan(file = 'modelo.stan', data = data,
```

```
                    iter = 1000, pars = c('beta_1','beta_2', 'beta_3',
                                          'gamma_1', 'gamma_2', 'gamma_3',
                                          'p_min', 'p_max', 'p_model',
                                          'pearson', 'deviance', 'log_lik',
                                          'log_lik_test'),
                    chains = 1)
```

Finalmente, para obtener los indicadores *loo* y *waic* desde R se extrae la log verosimilitud a través del comando *loo* :: *extract_log_lik*(*model*) y posteriormente se usan los comandos *rstan* :: *loo*(*log_lik*) y *loo* :: *waic*(*log_lik*) para calcular sus valores, tal como lo muestra el siguiente fragmento de código:

```
log_lik = loo::extract_log_lik(model, parameter_name = 'log_lik')

loo = rstan::loo(log_lik)
waic = loo::waic(log_lik)
```

B.0.2. Implementación en OpenBUGS

Algunas limitaciones de OpenBugs no permiten hacer la anterior implementación de manera tan sencilla como en Stan, sin embargo, a través de la generación demuestras del modelo y del calculo de la log verosimilitud se pueden hacer los mismos calculos en R. Partiendo del modelo normal heteroscedástico para el cual se tiene el siguiente código de OpenBUgs, en el cual, se han agregado las lineas para el cálculo de la log-verosimilitud tanto para los datos obervados como para las réplicas, *lok_lik* y *lok_lik_test* respectivamente.

```
    model
        {
        for (i in 1:N)
        {

        media[i] <- beta_1*x_1[i]+beta_2*x_2[i] + beta_3*x_3[i]
        sigma_2[i] <- exp(gamma_1*x_1[i] + gamma_2*x_2[i]+ gamma_3*x_4[i])
        tau[i] <- 1/sigma_2[i]
        y[i] ~ dnorm(media[i], tau[i])

        }

        beta_1 ~ dnorm(0,0.00001);
```

```
        beta_2 ~ dnorm(0,0.00001);
        beta_3 ~ dnorm(0,0.00001);
        gamma_1 ~ dnorm(0,0.00001);
        gamma_2 ~ dnorm(0,0.00001);
        gamma_3 ~ dnorm(0,0.00001);


        for (i in 1:N)
        {
        y_test[i] ~ dnorm(media[i], tau [i]);
        log_lik[i] <- -log(sigma_2[i]*sqrt(2*3.1416))-
                        (1/(2*sigma_2[i]))*pow(y[i]-media[i],2);
        log_lik_test[i] <- -log(sigma_2[i]*sqrt(2*3.1416))-
                        (1/(2*sigma_2[i]))*pow(y_test[i]-media[i],2);
        pearson_res[i] <- pow((y_test[i]-media[i]),2)/sigma_2[i];

        }
    }
```

Ahora, en R se lleva a cabo el calculo de los p valor bayesianos y los indicadores *loo* y $WAIC$.

```
data = list(x_1 = x_1, x_2 = x_2, x_3 = x_3, x_4 = x_4, y=y, N=n)

modelo_normal =R2OpenBUGS::bugs(data,
                inits = NULL,
                model.file="modelo.txt",
                parameters= c("beta_1", "beta_2", "beta_3",
                              "gamma_1", "gamma_2", "gamma_3",
                              "y_test", "log_lik",  "log_lik_test",
                              "res_pearson"),
                n.chains = 3,
                n.iter=10000
)


matriz_sims <- modelo_normal$sims.matrix

# Se extraen las observaciones de las replicas, y la log verosimilitud
# de las replicas y los datos.
y_test <- matriz_sims[,substr(colnames(matriz_sims),1,6) =="y_test"]
```

```
log_lik <- matriz_sims[,substr(colnames(matriz_sims),1,8) =="log_lik["]
log_lik_test <- matriz_sims[,substr(colnames(matriz_sims),1,12) =="log_lik_test"]
res_pearson <-  matriz_sims[,substr(colnames(matriz_sims),1,11) =="pearson_res"]



# Se obtiene el minimo y el maximo para cada iteración.
min_y_test <- apply(y_test,1,min)
max_y_test <- apply(y_test,1,max)

#se obtiene la verosimilitud total para cada iteración.
log_y <- apply(log_lik,1,sum)
log_y_test <- apply(log_lik_test,1,sum)
pearson <- apply(res_pearson,1,sum)


y_min <- min(y)
y_max <- max(y)

# Se calcula el p valor bayesiano

p_min <- mean(y_min > min_y_test)
p_max <- mean(y_max > max_y_test)
p_model <- mean(log_y > log_y_test)
pearson <- mean(pearson)
deviance <- -2*(mean(log_y))

# Se calcula loo y WAIC
loo = rstan::loo(log_lik)
waic = loo::waic(log_lik)
```

Bibliografía

[Agresti, 2015] Agresti, A. (2015). *Foundations of linear and generalized linear models*. John Wiley & Sons.

[Aitkin, 1987] Aitkin, M. (1987). Modelling variance heterogeneity in normal regression using glim. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 36(3), 332–339.

[Barndorff, 2014] Barndorff, O. (2014). *Information and exponential families: in statistical theory*. John Wiley & Sons.

[Bateson, 2009] Bateson, T. (2009). Gamma regression of interevent waiting times versus poisson regression of daily event counts: Inside the epidemiologist's toolbox–selecting the best modeling tools for the job. *Epidemiology*, 20(2), 202–204.

[Blei et al., 2017] Blei, D., Kucukelbir, A., & McAuliffe, J. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518), 859–877.

[Breslow, 1984] Breslow, N. (1984). Extra-poisson variation in log-linear models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 33(1), 38–44.

[Brown et al., 2001] Brown, P. J., Kenward, M. G., & Bassett, E. E. (2001). Bayesian discrimination with longitudinal data. *Biostatistics*, 2(4), 417–432.

[Carpenter et al., 2017] Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1).

[Castillo, 2017] Castillo, E. (2017). *Modelos Para Datos Longitudinales: Una Propuesta Bayesiana*. PhD thesis, Universidad Nacional de Colombia-Sede Bogotá.

[Castillo et al., 2020] Castillo, E., Cepeda, E., & Núnez, V. (2020). Bayesian structured antedependence model proposals for longitudinal data. *SORT*, 44(1), 1–30.

[Cepeda, 2001a] Cepeda, E. (2001a). *Modelagem da variabilidade em modelos lineares generalizados*. PhD thesis, Tese de D. Sc., IM–UFRJ, Rio de Janeiro, RJ, Brasil.

[Cepeda, 2001b] Cepeda, E. (2001b). Variability modeling in generalized linear models. *Unpublished Ph. D. Thesis. Mathematics Institute, Universidade Federal do Rio de Janeiro*.

[Cepeda, 2016] Cepeda, E. (2016). Double generalized linear models. Disponible en http://www.redeabe.org.br/novosinape2016/minicursos/minicurso2.pdf. Recuperado el dia 10 de Marzo de 2019.

[Cepeda & Achcar, 2010] Cepeda, E. & Achcar, J. (2010). Heteroscedastic nonlinear regression models. *Communications in Statistics—Simulation and Computation®*, 39(2), 405–419.

[Cepeda et al., 2012a] Cepeda, E., Andrade, M., & Achcar, J. (2012a). A seasonal and heteroscedastic gamma model for hydrological time series: A bayesian approach. In *AIP Conference Proceedings*, volume 1490 (pp. 97–107).: American Institute of Physics.

[Cepeda & Cifuentes, 2017] Cepeda, E. & Cifuentes, M. V. (2017). Double generalized beta-binomial and negative binomial regression models. *Revista Colombiana de Estadística*, 40(1), 141–163.

[Cepeda & Corrales, 2015] Cepeda, E. & Corrales, M. (2015). Gamma regression models with the gammareg r package1.

[Cepeda et al., 2016] Cepeda, E., Corrales, M., Cifuentes, M., & Zarate, H. (2016). On gamma regression residuals.

[Cepeda & Gamerman, 2000] Cepeda, E. & Gamerman, D. (2000). Bayesian modeling of variance heterogeneity in normal regression models. *Brazilian Journal of Probability and Statistics*, (pp. 207–221).

[Cepeda & Gamerman, 2004] Cepeda, E. & Gamerman, D. (2004). Bayesian modeling of joint regressions for the mean and covariance matrix. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 46(4), 430–440.

[Cepeda & Gamerman, 2005] Cepeda, E. & Gamerman, D. (2005). Bayesian methodology for modeling parameters in the two parameter exponential family. *Revista Estadística*, 57(168-169), 93–105.

[Cepeda et al., 2014] Cepeda, E., Migon, H. S., Garrido, L., & Achcar, J. A. (2014). Generalized linear models with random effects in the two-parameter exponential family. *Journal of Statistical Computation and Simulation*, 84(3), 513–525.

[Cepeda & Núñez, 2013] Cepeda, E. & Núñez, V. (2013). Spatial double generalized beta regression models: extensions and application to study quality of education in colombia. *Journal of Educational and Behavioral Statistics*, 38(6), 604–628.

[Cepeda et al., 2012b] Cepeda, E., Quintero, A., & Núñez, V. (2012b). Estimating infant mortality in colombia: some overdispersion modelling approaches. *Journal of Applied Statistics*, 39(5), 1011–1036.

[Chib & Greenberg, 1995] Chib, S. & Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4), 327–335.

[Cox & Reid, 1987] Cox, D. & Reid, N. (1987). Parameter orthogonality and approximate conditional inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 49(1), 1–18.

[Cribari & Zeileis, 2010] Cribari, F. & Zeileis, A. (2010). Beta regression in r. *Journal of Statistical Software, Articles*, 34(2).

[DeGroot & Schervish, 2012] DeGroot, M. & Schervish, M. (2012). *Probability and statistics*. Pearson Education.

[Dey et al., 1997] Dey, D., Gelfand, A., & Peng, F. (1997). Overdispersed generalized linear models. *Journal of Statistical Planning and Inference*, 64(1), 93–107.

[Efron, 1986] Efron, B. (1986). Double exponential families and their use in generalized linear regression. *Journal of the American Statistical Association*, 81(395), 709–721.

[Ferrari & Cribari, 2004] Ferrari, S. & Cribari, F. (2004). Beta regression for modelling rates and proportions. *Journal of applied statistics*, 31(7), 799–815.

[Fitzmaurice & Ravichandran, 2008] Fitzmaurice, G. & Ravichandran, C. (2008). A primer in longitudinal data analysis. *Circulation*, 118(19), 2005–2010.

[Gelfand & Dalal, 1990] Gelfand, A. & Dalal, S. (1990). A note on overdispersed exponential families. *Biometrika*, 77(1), 55–64.

[Guo et al., 2016] Guo, J., Lee, D., Sakrejda, K., Gabry, J., Goodrich, B., De Guzman, J., Niebler, E., Heller, T., & Fletcher, J. (2016). Rstan: the r interface to stan. *R package version*, 2(1).

[Harvey, 1976] Harvey, A. C. (1976). Estimating regression models with multiplicative heteroscedasticity. *Econometrica: Journal of the Econometric Society*, (pp. 461–465).

[Hauer, 2001] Hauer, E. (2001). Overdispersion in modelling accidents on road sections and in empirical bayes estimation. *Accident Analysis & Prevention*, 33(6), 799–808.

[Hedeker & Gibbons, 2006] Hedeker, D. & Gibbons, R. (2006). *Longitudinal data analysis*, volume 451. John Wiley & Sons.

[Hoffman & Gelman, 2014] Hoffman, M. & Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1), 1593–1623.

[Hui & Berger, 1983] Hui, S. L. & Berger, J. O. (1983). Empirical bayes estimation of rates in longitudinal studies. *Journal of the American Statistical Association*, 78(384), 753–760.

[Kenward, 1987] Kenward, M. (1987). A method for comparing profiles of repeated measurements. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 36(3), 296–308.

[Long, 1990] Long, S. (1990). The origins of sex differences in science. *Social forces*, 68(4), 1297–1316.

[Lunn et al., 2012] Lunn, D., Jackson, C., Best, N., Thomas, A., & Spiegelhalter, D. (2012). *The BUGS book: A practical introduction to Bayesian analysis*. CRC press.

[McCullagh, 1983] McCullagh, P. (1983). *Generalized linear models*. Routledge.

[McElreath, 2020] McElreath, R. (2020). *Statistical rethinking: A Bayesian course with examples in R and Stan*. CRC press.

[Neal et al., 2011] Neal, R. et al. (2011). Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11), 2.

[Nelder & Wedderburn, 1972] Nelder, J. & Wedderburn, R. (1972). Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3), 370–384.

[Pammer & Kevan, 2007] Pammer, K. & Kevan, A. (2007). The contribution of visual sensitivity, phonological processing, and nonverbal iq to children's reading. *Scientific Studies of Reading*, 11(1), 33–53.

[Pham et al., 2010] Pham, T. V., Piersma, S. R., Warmoes, M., & Jimenez, C. R. (2010). On the beta-binomial model for analysis of spectral count data in label-free tandem mass spectrometry-based proteomics. *Bioinformatics*, 26(3), 363–369.

[Phelps, 1982] Phelps, K. (1982). Use of the complementary log-log function to describe dose-response relationships in insecticide evaluation field trials. In *GLIM 82: Proceedings of the International Conference on Generalised Linear Models* (pp. 155–163).: Springer.

[Ryan et al., 1976] Ryan, T., Joiner, B., & Ryan, B. (1976). Minitab student handbook.

[Salem & Mount, 1974] Salem, A. & Mount, T. (1974). A convenient descriptive model of income distribution: the gamma density. *Econometrica: journal of the Econometric Society*, (pp. 1115–1127).

[Simas et al., 2010] Simas, A., Barreto, W., & Rocha, A. V. (2010). Improved estimators for a general class of beta regression models. *Computational Statistics & Data Analysis*, 54(2), 348–366.

[Smithson & Verkuilen, 2006] Smithson, M. & Verkuilen, J. (2006). A better lemon squeezer? maximum-likelihood regression with beta-distributed dependent variables. *Psychological methods*, 11(1), 54.

[Smyth, 1989] Smyth, G. (1989). Generalized linear models with varying dispersion. *Journal of the Royal Statistical Society: Series B (Methodological)*, 51(1), 47–60.

[Smyth & Verbyla, 1999] Smyth, G. & Verbyla, A. (1999). Double generalized linear models: approximate reml and diagnostics. In *Statistical Modelling: Proceedings of the 14th International Workshop on Statistical Modelling* (pp. 66–80).

[Spiegelhalter et al., 2007] Spiegelhalter, D., Thomas, A., Best, N., & Lunn, D. (2007). Openbugs user manual, version 3.0. 2. *MRC Biostatistics Unit, Cambridge*.

[Stan Development Team, 2016] Stan Development Team (2016). Stan modeling language users guide and reference manual. *Technical report*.

[Sturtz et al., 2005] Sturtz, S., Ligges, U., & Gelman, A. (2005). R2winbugs: a package for running winbugs from r.

[Sturtz et al., 2010] Sturtz, S., Ligges, U., & Gelman, A. (2010). R2openbugs: a package for running openbugs from r. *URL http://cran. rproject. org/web/packages/R2OpenBUGS/vignettes/R2OpenBUGS. pdf*.

[Vasconcellos & Cribari, 2005] Vasconcellos, K. & Cribari, F. (2005). Improved maximum likelihood estimation in a new class of beta regression models. *Brazilian Journal of Probability and Statistics*, (pp. 13–31).

[Vehtari et al., 2017] Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and computing*, 27(5), 1413–1432.

[Vehtari et al., 2018] Vehtari, A., Gelman, A., & Gabry, J. (2018). loo: Efficient leave-one-out cross-validation and waic for bayesian models. *R package version*, 2(0), 1003.

[Verbyla, 1993] Verbyla, A. (1993). Modelling variance heterogeneity: residual maximum likelihood and diagnostics. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(2), 493–508.

[Wedderburn, 1974] Wedderburn, R. (1974). Quasi-likelihood functions, generalized linear models, and the gauss—newton method. *Biometrika*, 61(3), 439–447.

[Weiss, 2005] Weiss, R. (2005). *Modeling longitudinal data*. Springer Science & Business Media.

[Williams, 1982] Williams, D. (1982). Extra-binomial variation in logistic linear models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 31(2), 144–148.

[Young, 2018] Young, D. (2018). *Handbook of regression methods*. CRC Press.

[Zacks, 2014] Zacks, S. (2014). *Parametric statistical inference: basic theory and modern approaches*, volume 4. Elsevier.

[Zeileis et al., 2016] Zeileis, A., Cribari, F., Gruen, B., Kosmidis, I., Simas, A. B., Rocha, A. V., & Zeileis, M. A. (2016). Package 'betareg'. *R package*.

[Zhang et al., 2014] Zhang, Y., Chen, M., Ibrahim, J. G., Zeng, D., Chen, Q., Pan, Z., & Xue, X. (2014). Bayesian gamma frailty models for survival data with semi-competing risks and treatment switching. *Lifetime data analysis*, 20(1), 76–105.

[Zimmerman & Nunez, 2009] Zimmerman, D. & Nunez, V. (2009). *Antedependence models for longitudinal data*. CRC Press.