El desarrollo de un paquete computacional para la estimación de los parámetros del modelo resulto ser la tarea más retadora y cautivante de del trabajo. El código, conecta la teoría abstracta del modelo y la aterriza en un terreno práctico y tangible. Durante este esfuerzo, se encontró la necesidad de desarrollar funcionalidad adicional para probar cada parte del modelo e ir visualizando paso a paso las correspondientes proyecciones. Al final, la capacidad y flexibilidad del paquete quedan reflejadas en la practicidad de su uso pues con pocas líneas de código se puede correr una realización del modelo y visualizar ciertos componentes.

El paquete se desarrolló en el lenguaje abierto de programación estadístico `R` por dos razones: por la familiaridad con la que previamente se contaba y por la facilidad del lenguaje para manejar objetos matemáticos como vectores, matrices y listas. Asimismo el paquete se escribe siguiendo **wickham2015rpck**, uno de los principales desarrolladores del popular IDE `RStudio`[1].

Para instalar el paquete `bpwpm` de forma remota, basta con instalar un paquete adicional y correr una línea, ver código 1. Posterior a esto, toda las las capacidades y funcionalidad del modelo con las que se crearon las gráficas de este trabajo deberían de ser asequibles para el usuario.

```
install.package("devtools")
devtools::install_github("PaoloLuciano/bpwpm2")
```

Tabla 1: Descarga del paquete

1. *Integrated Development Enviroment*

La idea de desarrollar un paquete es poder usar el modelo de una forma sencilla, así como validar y explorar las realizaciones del modelo sin tener que programar demasiado. Por lo tanto se presenta un ejemplo *mínimiamente funcional* de como se puede correr un modelo `bpwpm`, ver código 2. La base de datos viene incluida en el paquete y corresponde a la del ejemplo 2.

0.1. Listado de funciones

Se trata de respetar la notación usada dentro del trabajo para su fácil comprensión. Sin embargo, como parte del desarrollo del paquete, este también cuenta con una documentación más detallada que se puede acceder directamente desde la terminal o un IDE.

Función *bpwpm_gibbs*

Encontrada en el archivo `bpwpm2.R`.

> `bpwpm_gibbs`$(\mathbf{y}, \mathbf{X}, M, J, K, \ldots)$: esta es la función principal del paquete que realiza la simulación de la cadena de Markov usando los datos para el entrenamiento y los parámetros especificados.

```r
        # Ejemplo minimo del paquete bpwpm2

# 0. Lectura de datos y preparacion del modelo
library(bpwpm2)
# set.seed(135491) incluir para obtener replicabilidad

datos <- bpwpm2::NormalBiVarMod
Y <- datos$Y # Consitencia en la notacion
de este trabajo
X <- datos[,2:3]
plot_2D_data(Y,X) # Visualizacion de los datos

# Parametros del modelo
M <- 3 # Controla el grado de los polinomios
J <- 3 # Controla el numero de nodos
K <- 2 # Controla la suavidad
draws <- 10000 # N sim

# 1. Modelo
model <- bpwpm2_gibbs(  Y = Y, X = X,
        M = M, J = J,
        K = K, draws = draws)
burn_in <- 7500
thin <- 0
predictions <- predict(model, Y, X, burn_in, thin, 'mean')

# 2. Analisis y visualizaciones
summary(model)
plot(model, n = draws, m = 5)
summary(predictions)
plot(predictions)
plot_2D(Y, X, predictions, n = 20, alpha = 0.9)
```

Tabla 2: Ejemplo mínimo funcional

Función *predict.bpwpm*

Encontrada en el archivo `predict_funcs.R`.

> `predict.bpwpm`(`object`, $\tilde{y}$, $\tilde{\mathbf{X}}$,`thin`, `burn-in`, `type`, ...): función genérica de la clase S3 que realiza predicciones de un nuevo conjunto de datos $\tilde{\mathbf{X}}$ dado un objeto de la clase `bpwpm` y los prueba contra las etiquetas reales $\tilde{y}$.[2]

Funciones matemáticas

Funciones auxiliares relacionadas con procedimientos matemáticos más complejos. Encontradas en el archivo `math_utils.R`.

> `calculate_Psi`($\mathbf{X}, M, J, K, n, d, \tau$): calcula matriz $\widetilde{\Psi}(\mathbf{X})$ con base en los parámetros M, J y K. Los componentes de la matriz son las expansiones de bases de cada covariable para cada una de las observaciones $i = 1, \ldots, n$.

> `calc_F`($\Psi, \boldsymbol{\beta}, d$): calcula una matriz F donde cada columna representa la transformación f_j. Este calculo se obtiene de separar $\widetilde{\Psi}(\mathbf{X})$ en sus correspondientes componentes.

2. Los métodos S3 (precisando, clases S3) son funciones que crean objetos y son la estructura que permite que `R` sea más análogo a un lenguaje orientado a objetos tradicional como lo sería `Python`. Son útiles pues detectan automáticamente la clase del objeto con el que se las llama, ya sea un modelo o cualquier otra estructura soportada por `R` y actúan de forma automática según el tipo. Algunos ejemplos son las funciones `print` y `summary`.

`ergodic_mean(mcmc_chain)`: calcula la media ergódica por componentes de una cadena de Markov.

`log_loss`$(\mathbf{y}, \hat{\mathbf{p}}, \ldots)$: calcula la función *log-loss* dados los valores reales $\mathbf{y}$ y las probabilidades ajustadas $\hat{p}$. La versión implementada controla por el posible error de máquina para probabiliades muy cercanas a cero o a uno.

`accurracy`: calcula la precisión del modelo definida como el número total de predicciones correctas entre el número total de observaciones.

`contingency_table`$(\tilde{\mathbf{y}}, \hat{\mathbf{p}})$: calcula la matriz de contingencia para las predicciones.

`calculate_eta`(F): calcula el vector de medias para cada observación $\boldsymbol{\eta} = \widetilde{\Psi}(\mathbf{X})\boldsymbol{\beta}$. La implementación computacional real, únicamente es la suma por renglones de la matriz F respetando la aditividad del modelo.

`model_eta`($\tilde{\mathbf{X}}$, `params`): calcula la función de proyección η para un conjunto de datos $\tilde{\mathbf{X}}$, los cuales pueden ser los mismos con los que se entrenó el modelo o un conjunto de datos nuevos.

`post_probs`($\tilde{\mathbf{X}}$, `params`): calcula la probabilidad posterior de la clasificación para un conjunto de datos nuevo o el mismo que se usó en el entrenamiento.

Funciones útiles

Funciones auxiliares para la simplificación de procesos en las funciones de más alto nivel. Encontradas en el archivo `utils.R`.

`thin_chain`(`mcmc_chain, burn_in, thin`): dada una matriz de una cadena MCMC esta función auxiliar recorta y adelgaza la cadena.

`thin_bpwpm`(`bpwpm, burn_in, thin`): adelgaza todos los parámetros de un modelo `bpwpm` y regresa un objeto del mismo tipo.

`post_params`(`bpwpm, burn_in, thin, type`): dado un objeto de la clase `bpwpm`, la función hace la estimación puntual de los parámetros del modelo $\hat{\boldsymbol{\beta}}$ utilizando el tipo de estimación puntual (media o mediana) especificado en `type`. Además, recorta y adelgaza la cadena y regresa un objeto conteniendo todos los parámetros con clase `bpwpm_params`.

Funciones gráficas

Funciones que facilitan el análisis gráfico de los datos y el modelo de forma rápida y sencilla. Existen 3 funciones importante que toman el papel de *envoltorio*[3] pues ayudan llamando a otras subfunciones para visualizar una a una las gráficas relevantes: `plot.bpwpm, plot.bpwpm_predictions` y `plot_2D`. Prácticamente todas las fun-

3. *wrapper function*

ciones dependen de el paquete libre para graficación en `R`: `ggplot2` que se instala en paralelo con el paquete de este trabajo. Todas estas funciones se encuentran en el archivo `plot_funcs.R`.

`plot.bpwpm`(`object, n`): para un objeto del tipo `bpwpm`, grafica las trazas y los histogramas para los parámetros estimados $\hat{\boldsymbol{\beta}}$. `n` controla el número de observaciones por graficar de la cadena muestreada original.

`plot_ergodic_mean`(`object, n`): grafica la média ergódica de una cadena de Markov muestreada.

`plot_chains`(`mcmc_chains, n`): grafica la traza de una cadena de Markov muestreada.

`plot_hist`(`mcmc_chains, n`): grafica el histograma de una cadena de Markov muestreada.

`plot.bpwpm_predictions`(`object, ...`): dado un objeto del tipo `bpwpm_prediction`, grafica las f_j del modelo ya estimado. Se usa como *envoltorio* para la función `plot_each_F`.

`plot_each_F`($\mathbf{y}, \mathbf{X}, F$): grafica cada una de las las funciones f_j.

`plot_2D`($\mathbf{y}, \mathbf{X}$ `bpwpm_params`, n, `alpha`): dados los datos en 2D ($d = 2$), se dibujan todas las gráficas posibles para este caso particular de los modelos.

El parámetro `alpha` controla la transparencia de los puntos proyectados y el parámetro n la finura de la malla en la representación en tres dimensiones.

`plot_2D_data`($\mathbf{y}, \mathbf{X}$): grafica de puntos de los datos originales.

`plot_2D_proj`($\mathbf{y}, \mathbf{X}$, `bpwpm.params, n, alpha`): grafica la proyección de η en el espacio de covariables $\mathbf{X}$ para datos bivariados. Bueno para identificar las regiones de clasificación y visualizar los resultados del modelo.

`plot_3D_proj`($\mathbf{X}$, `bpwpm.params, n`): usando el paquete de `lattice`, se crea una gráfica de malla para representar la función $\boldsymbol{\eta}(\mathbf{X})$ en 3D. El parámetro n controla la finura de la malla.

`plot_3D_proj`($\mathbf{y}$, `bpwpm.params`): grafica la proyección de los puntos al transformarlos no linealmente en f_j.

Funciones de resumen

Estas tres funciones son métodos S3 para la rápida exploración de los objetos del paquete. Se encuentran en el archivo `summary_funcs.R`.

`summary.bpwpm`(`objeto y parámetros`): imprime en pantalla la información sobre la llamada del modelo y los principales resúmenes numéricos de las cadenas para los parámetros $\boldsymbol{\beta}$.

`summary.bpwpm_params`(`objeto y parámetros`): imprime la estimación sobre los parámetros posteriores $\hat{\boldsymbol{\beta}}$.

`summary.bpwpm_prediction`(`objeto y parámetros`): resume e imprime la información sobre una predicción con el modelo. Esto es: la precisión, la medida *log-loss*, el tipo de estimación puntual usada, la tabla de contingencia, los nodos y los parámetros posteriores.