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 a, uno de los principales desarrolladores del popular IDE[1] `RStudio`.

Para instalar el paquete `bpwpm` de forma remota, basta con instalar un paquete adicional y correr una línea, ver tabla 2. 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("https://github.com/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, en la tabla 2 se presenta un ejemplo *mínimiamente funcional* de como se puede correr un modelo `bpwpm`.

```
mod <- bpwpm(datos y parámetros)
summary(mod)
plot(mod)
plot_2D(mod) # Si los datos están en 2D

# Datos X y Y en muestra o fuera de ella
mod_res <- predict(mod,X,Y)
summary(mod_res)
plot(mod_res)
```

Tabla 2: Ejemplo mínimamente funcional

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 `bpwpm_gibbs.R`.

`bpwpm_gibbs`($\mathbf{y}, X, M, J, K, \ldots$): 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.

Función *predict.bpwpm*

Encontrada en el archivo `predict_funcs.R`.

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

Funciones matemáticas

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

`calculate_Phi`(X, M, J, K, d, τ): calcula la expansión de bases $\Psi_j(X, \mathcal{P})$ para $j = 1, \ldots, d$ y cada una de las observaciones con base en los parámetros

M, J y K.

`calc_F`(Ψ, w, d,`intercept`): calcula la matriz $F = \Psi\mathbf{w}$ donde cada columna representa la transformación f_j.

`log_loss`($\mathbf{y}, \hat{\mathbf{p}}, \ldots$): calcula la función *log-loss* dados los valores reales $\mathbf{y}$ y las probabilidades ajustadas $\hat{p}$.

`mode`(x): calcula la moda de un vector x.

`calc_proy`(F, β): calcula el vector de medias para cada observación $f = F\beta$.

`model_proy`($\tilde{X}$, `params`): calcula la función de proyección f para un conjunto de datos $\tilde{X}$, pueden ser con los que se entrenaron los parámetros o un conjunto de datos nuevos.

`post_probs`($\tilde{X}$, `params`): calcula la probabilidad posterior de la clasificación.

`acurracy`($\tilde{y}, \hat{\mathbf{p}}$): calcula la precisión total del modelo.

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

`ergodic_mean`(`mcmc_chain`): calcula la media ergódica de una cadena MCMC.

Funciones útiles

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

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

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

`post_params(bpwpm, thin, burn_in, type)`: dado un objeto de la clase `bpwpm`, la función hace la estimación puntual de los parámetros del modelo β y $\mathbf{w}$ utilizando el tipo de estimación puntual (media, moda 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 habilitan el análisis gráfico de los datos y el modelo de forma rápida y sencilla. Existen 3 funciones importante que toman el papel *envoltorio*[2] para las demás: `plot.bpwpm, plot.bpwpm_predictions` y `plot_2D`. Encontradas en el archivo `plot_funcs.R`.

2. wrapper function

`plot.bpwpm(object, n)`: para un objeto del tipo `bpwpm`, grafica las trazas y los histogramas para los parámetros β y cada w_j.

`plot_chains(mcmc_chains, n, title)`: grafica la traza de una cadena MCMC.

`plot_hist(mcmc_chains, n, title)`: grafica el histograma de una cadena MCMC.

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

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

`plot_2D`($\mathbf{y}, X$ `bpwpm_params`, n, `alpha`): dados los datos en 2D, se realizan 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.

`plot.2D_data`($\mathbf{y}, X$): grafica de puntos de los datos originales usando el paquete `ggplot2`.

`plot.2D_proj`($\mathbf{y}, X$`bpwpm.params, n, alpha`): grafica la proyección de f en el espacio de covariables $\mathcal{X}$ para datos bivariados. Bueno para identificar las regiones de clasificación y visualizar los resultados del modelo.

`plot.3D_proj`(X, `bpwpm.params, n`): usando el paquete de `lattice`, se crea una gráfica de malla para representar la función f en 3D, únicamente se puede utilizar cuando se tengan datos en 2D. El parámetro n controla la finura de la malla.

`plot.ergodic_mean`(`bpwpm, thin, burn_in`): grafica la media ergódica para las cadenas de un `bpwpm` para todos los parámetros β y w_j.

Funciones de resumen

Estas tres funciones, son métodos S3[3] 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 β y $\mathbf{w}$.

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

`summary.bpwpm_prediction`(`objeto y parámetros`): resume e imprime la in-

3. 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.

formación 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 entrenados.