Grado en Matemática Computacional

Trabajo de Final de Grado

Detección de comportamientos anómalos mediante mixturas de distribuciones von Mises y una extensión del algoritmo CUSUM

Autor:
Diego Lacomba Fañanás

Tutora académica:
Amelia Simó Vidal

Fecha de lectura: julio de 2023
Curso académico 2022/2023

Resumen

El trabajo de fin de grado recogido en esta memoria surge a partir del proyecto realizado por el alumno Diego Lacomba durante su estancia en prácticas externas. El proyecto consiste en el desarrollo de un sistema de monitorización en interiores, con el objetivo de detectar anomalías en el comportamiento de la persona monitorizada. Este sistema de monitorización permitirá estudiar el desarrollo de una enfermedad degenerativa, cambios producidos por una medicación, o simplemente supervisar a una persona con dependencia como puede ser una persona de edad avanzada.

Esta memoria incluye los fundamentos teóricos en los que se sustenta el proyecto y su implementación. Nos centramos únicamente en la monitorización del tránsito entre las diferentes habitaciones de la casa.

El modelo teórico propuesto para las transiciones se compone de un conjunto de mixturas de distribuciones von Mises, donde cada una de ellas representa una transición posible entre dos habitaciones. Los datos empleados se han obtenido de un *dataset* público. La estimación de los parámetros de las mixturas se ha realizado con el algoritmo EM y la detección de anomalías mediante el algoritmo CUSUM. Para comprobar el correcto ajuste del modelo con los parámetros obtenidos por el algoritmo EM se ha implementado un test de bondad de ajuste de Kolmogorov-Smirnov para mixturas de distribuciones von Mises. Por otro lado, para comprobar la correcta implementación de CUSUM, se han realizado simulaciones y mostrado los resultados.

Palabras clave

Modelo de comportamiento, distribución von Mises, Modelos de mixturas, Algoritmo EM, CUSUM.

Keywords

Behavior model, von Mises distribution, Mixture models, EM algorithm, CUSUM.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

El proyecto presentado en esta memoria forma parte de un proyecto mucho más ambicioso de la Cátedra Cuatroochenta de Inteligencia Artificial, Salud y Bienestar de la Universitat Jaume I, creada en junio de 2021 y promovida por la empresa Cuatroochenta. Esta cátedra tiene como rango de actuación fomentar la enseñanza, investigación, diseminación de conocimiento e innovación en el sector de la tecnología para promover la salud de las personas. Dicho proyecto tiene como principal objetivo obtener un modelo de comportamiento de una persona en su propio hogar, con tal de establecer el patrón de comportamiento habitual y detectar posibles discrepancias y anomalías con el paso de tiempo. Algunos casos de personas que podrían beneficiarse del mismo serían, por ejemplo, personas de avanzada edad, con alguna enfermedad degenerativa, o que se estén sometiendo a alguna medicación.

Para llevar a cabo este objetivo, inicialmente se pretendía hacer uso de un sistema de monitorización en interiores desarrollado durante la estancia en prácticas, con el que se monitorizaría a una persona en su hogar y se obtendrían los datos necesarios. El sistema de monitorización está compuesto por un conjunto de balizas repartidas por la casa que miden la intensidad de la señal *bluetooth* de una pulsera inteligente con la que se equipa a la persona monitorizada, y que cada cierto periodo de tiempo envían las mediciones a un servidor mediante peticiones HTTP.

En el presente Trabajo de Fin de Grado nos centramos únicamente en la monitorización de las transiciones entre las diferentes habitaciones de la vivienda. Además, por la limitación de tiempo, para la recogida de datos reales haremos uso de *datasets* públicos. En [1] hay diversos *datasets* con los que poder trabajar, entre los cuales encontramos el empleado en este proyecto.

El *dataset* empleado esta compuesto por una tabla de observaciones ordenadas por hora y fecha que corresponden a las activaciones y desactivaciones de los sensores que están repartidos por la casa. Con una depuración previa podemos construir la secuencia de habitaciones que recorre la persona y, por ende, los momentos en los que tiene lugar una transición entre dos habitaciones.

El modelo propuesto consiste en un conjunto de mixturas de distribuciones von Mises que modelizan las transiciones entre dos habitaciones. El hogar monitorizado cuenta con 5 habitaciones, por lo tanto tendremos un total de 20 mixturas, una para cada transición.

Una vez depurados los datos, tiene lugar el entrenamiento del modelo, es decir, la estimación de los parámetros de las mixturas. A continuación, mediante un test de bondad de ajuste, se comprobó si los modelos propuestos modelizan correctamente los datos y finalmente se implementó el algoritmo de detección de cambios. La estimación de los parámetros se realizará con el algoritmo EM, el test de bondad de ajuste será Kolmogorov-Smirnov y el proceso de detección de cambios en los parámetros se realizará con una extensión del algoritmo CUSUM.

La primera parte de esta memoria recoge los fundamentos teóricos en los que se sustenta el proyecto. Comenzando con una introducción a la estadística circular, con sus antecedentes históricos, algunos ejemplos de aplicación y las distribuciones de probabilidad más conocidas entre las que encontramos la distribución von Mises. Continuando con una sección reservada para las mixturas de distribuciones de probabilidad, incluyendo los métodos de estimación de parámetros más conocidos y reservando un apartado exclusivamente para el algoritmo EM, el empleado en este proyecto. También encontramos una sección para el test de bondad de ajuste de Kolmogorov-Smirnov y finalmente la teoría relativa a la detección de anomalías, concretamente el algoritmo CUSUM y la extensión empleada.

La segunda parte de la memoria contiene la implementación del modelo, haciendo referencias al código desarrollado en Julia y R que se encuentra adjunto en el Anexo. Por último, se comentan los resultados obtenidos, las conclusiones del trabajo y algunas propuestas de trabajo futuro.

1.2. Objetivos

El objetivo principal de este trabajo consiste en detectar posibles discrepancias y anomalías en el comportamiento de una persona dentro de su hogar, en particular respecto a los tránsitos entre habitaciones. Este objetivo principal se puede desglosar en los siguientes objetivos específicos:

- Búsqueda y depuración de datos de *datasets* públicos.
- Diseño de modelos de comportamiento mediante mixturas de distribuciones von Mises, para cada una de las posibles transiciones.
- Entrenamiento de los modelos.
- Comprobación de la bondad del ajuste.
- Implementación de un método para la detección de anomalías.

Capítulo 2

Fundamentos Teóricos

En este capítulo introduciremos los fundamentos teóricos en los que se sustenta este trabajo. Como hemos comentado en el apartado anterior, los datos con los que trabajaremos corresponden a la variable momento del día en que se observa un cierto suceso, en particular una transición de una habitación a otra. Son datos que se pueden representar en un círculo unitario, por ende, podemos considerarlos datos de naturaleza circular y por esta razón comenzaremos el capítulo con una introducción a este tipo de datos, sus antecedentes, ejemplos reales y algunas de sus distribuciones más conocidas, profundizando más en la que emplearemos para modelizar la transición entre habitaciones del sujeto monitorizado, la distribución von Mises. Con tal de obtener un modelo más preciso, será necesario el uso de mixturas de distribuciones. En este capítulo también introducimos la teoría de estos modelos junto a métodos de estimación de sus parámetros, entre los que encontramos el algoritmo EM. Finalmente, introduciremos la teoría correspondiente a la detección de anomalías. En nuestro modelo, una anomalía realmente es una variación en los parámetros de las distribuciones. Para detectar estos cambios existe un método llamado CUSUM, el cuál también introduciremos en esta sección.

2.1. Estadística circular

2.1.1. Introducción

La Estadística Circular es una rama de la estadística que se enfoca en el análisis de datos que se pueden representar en un círculo unitario. Estos datos son conocidos como datos circulares y se obtienen a partir de variables circulares o direccionales. Estos se caracterizan por no tener un valor cero real, ya que podemos ubicarlo de forma arbitraria en cualquier punto del círculo. Otra característica es la designación subjetiva de valores máximos y mínimos. No existen magnitudes cuando trabajamos con este tipo de datos, un ángulo no es mayor que otro, simplemente indican direcciones diferentes. También se caracterizan por desenvolverse en una escala finita, entre 0^{o} y 360^{o}, o entre 0 y 2π. En la Figura 2.1 tenemos un ejemplo de como representar este tipo de datos.

La estadística circular nace de la necesidad de tratar este tipo de datos, con ciertas carac-

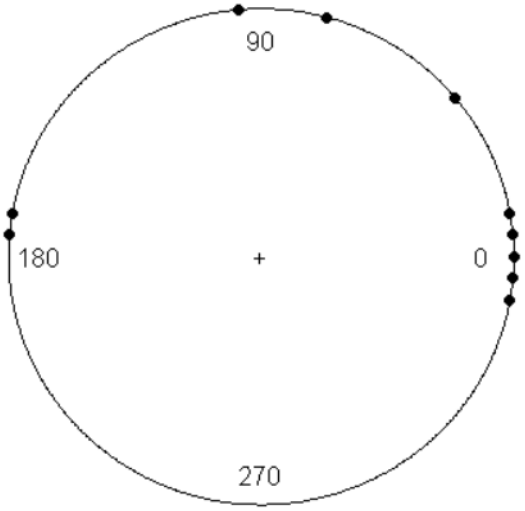


Figura 2.1: Ejemplo de gráfico representando observaciones de una variable tipo circular.

terísticas que implican que no se pueda hacer uso de los procedimientos típicos de la estadística en la recta real. La aplicación de la estadística circular la encontraremos en una amplia variedad de disciplinas debido a la naturaleza de los datos circulares, los cuales presentan una gran versatilidad, representando direcciones, orientaciones o fenómenos cíclicos que puedan tener lugar en un momento del día, semana, mes, etc.

Los datos circulares se pueden obtener a partir de diversos instrumentos de medición, como la brújula, el reloj, el transportador de ángulos o el teodolito. Por ejemplo, las direcciones de viento, las direcciones migratorias de las aves, el vuelo de las abejas hacia la colmena, o como en nuestro caso particular, el momento de transición de una habitación a otra por parte del sujeto monitorizado. Por supuesto, cuando trabajamos con datos direccionales en el espacio tridimensional y no en el plano, es más correcto hablar de estadística direccional. En este trabajo nos limitaremos a hablar de estadística circular, ya que los datos manipulados se pueden representar en el círculo de radio 1.

2.1.2. Antecedentes históricos

Desde finales del primer milenio ya se tiene conocimiento de la existencia de gráficos de naturaleza circular. Ya en el siglo X o XI se realizaban estudios astrológicos reflejados en gráficos circulares y también se conocen gráficos del mismo tipo en estudios sobre el campo magnético de la tierra en 1510 y 1701. Pero no fue hasta mediados del siglo XVIII que comenzó el análisis de datos circulares, cuando se estudiaron las separaciones angulares de las estrellas y se comenzó a hablar de distribuciones uniformes de trayectorias o de la necesidad de analizar los datos direccionales de forma diferente a los ordinarios. Un siglo después, en la guerra de Crimea, se empleó una primera referencia de lo que actualmente se conoce como diagrama de rosa para analizar los registros sanitarios. Podemos ver esta referencia en la Figura 2.2.

El siglo XX aparecieron una gran variedad de aplicaciones en otras disciplinas científicas

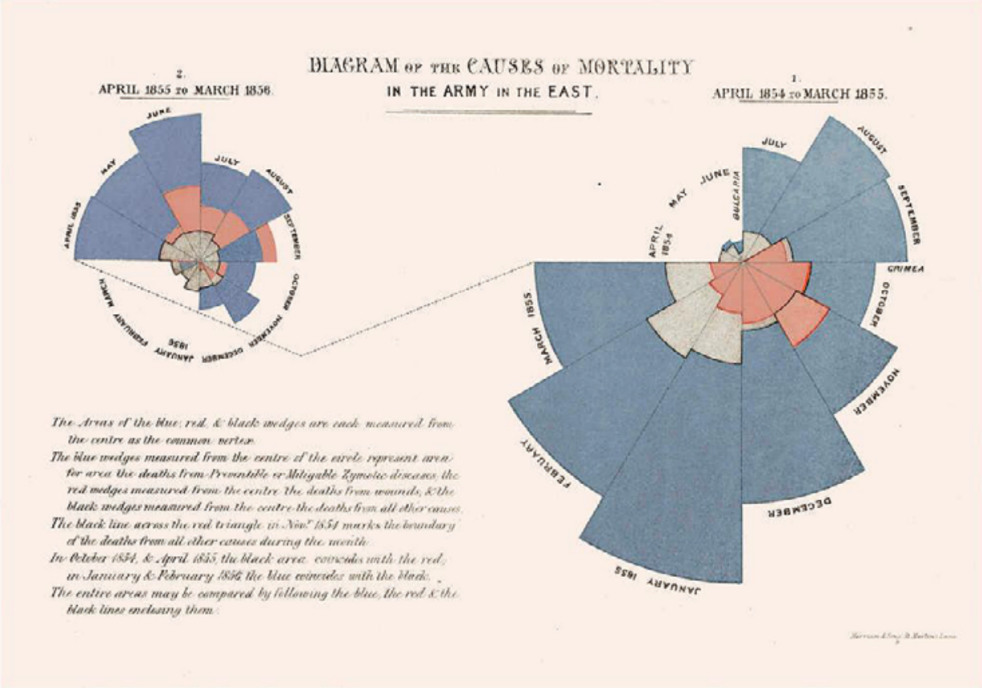


Figura 2.2: Diagrama precursor del actual diagrama de rosa (1854-1856). Gráfico obtenido de [2].

como la Biología y la Geología. Como las contribuciones de Schmidt sobre la representación de datos circulares o la de von Mises con la distribución que lleva su nombre y que emplearemos en este trabajo.

Fue el trabajo de Fisher [3] el que propició el desarrollo de numerosos procedimientos y distribuciones direccionales, como la estimación puntual, los contrastes de hipótesis de una y varias muestras, o la regresión. Posteriormente se desarrollaron muchos más métodos estadísticos como, por ejemplo, el análisis de la varianza y pruebas paramétricas y no paramétricas.

En la década de los 90, la publicación del libro *Directional Statistics* de Mardia y Jupp [4], que también se emplea como referencia en este trabajo, y los avances tecnológicos que permitieron el desarrollo de métodos intensivos de computación y descriptivos más potentes, impulsaron la investigación moderna en estadística direccional.

Algunos ejemplos de casos prácticos en distintos campos de la ciencia son:

- En Medicina. En los vectocardiogramas la información sobre la actividad eléctrica de una corazón durante un latido se describe como una órbita casi plana en un espacio tridimensional. También la incidencia de las muertes a causa de una enfermedad en diferentes momentos del año proporciona datos circulares. En la Figura 2.3 podemos ver representado el ciclo de temblores semanales causados por epilepsia de un conjunto de pacientes. En ese estudio [5] se pretendía demostrar que la epilepsia se manifiesta de forma periódica.

- En Biología. Se utiliza estadística circular en el estudio de los movimientos de los animales. Algunas de las preguntas de interés en estos estudios son si las direcciones que toman los

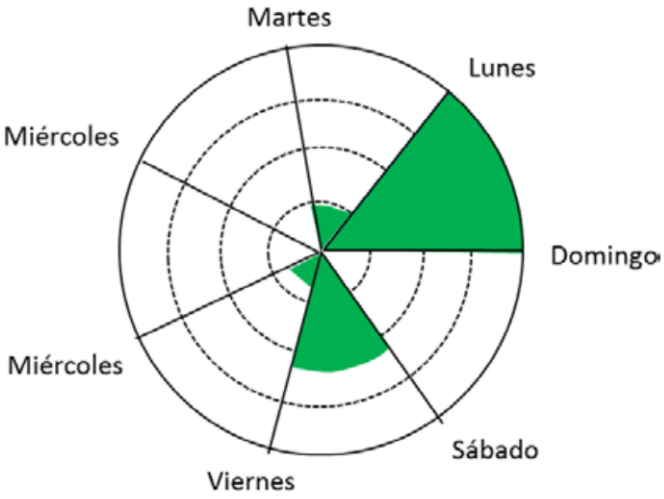


Figura 2.3: Temblores semanales causados por la epilepsia en un conjunto de pacientes. Gráfico obtenido de [2].

animales se distribuyen uniformemente en el círculo o si los animales tienden a ir hacia una dirección específica. En [6] se demuestra que la orientación de los nidos de las aves está influenciada por efectos ambientales como el viento o la lluvia y también por la posición geográfica. En la Figura 2.4 encontramos un gráfico con los datos de la orientación de los nidos de *Furnarius rufus*.

- En Ciencias de la Tierra. Encontramos una gran variedad de aplicaciones considerando que la superficie de la tierra es aproximadamente una esfera. El epicentro de un terremoto o la estimación relativa de las rotaciones de las placas tectónicas son casos donde se trabaja con datos direccionales. Por otro lado, la atmósfera de la tierra es otro lugar donde aparecen este tipo de datos. Como pueden ser datos de interés para la Meteorología la dirección del viento, la frecuencia diaria de tormentas o las veces del año que tiene lugar una lluvia intensa. En [7] se estudia la concentración de polen de dos tipos de planta en una ciudad. Concluyendo con que las horas de la mañana son las de menor exposición. En la Figura 2.5 se puede observar el gráfico correspondiente.

2.1.3. Medidas descriptivas

Antes de comenzar con las distribuciones de probabilidad circulares explicaremos como obtener algunas de las medidas de posición y dispersión, como son la media y la varianza, para una muestra de ángulos.

El *ángulo medio* nos indica la orientación media de una muestra de n datos circulares. Si aplicásemos la conocida media aritmética no obtendríamos el valor deseado, por ejemplo, si la aplicamos a los ángulos 15^{o} y 345^{o} obtendríamos 180^{o} como resultado, cuando realmente la dirección media sería 0^{o} como podemos ver en la Figura 2.6. De manera que la media aritmética no es una buena herramienta para obtener direcciones medias ya que depende de donde "cortamos" el círculo. La alternativa es tratar los datos como vectores unitarios.

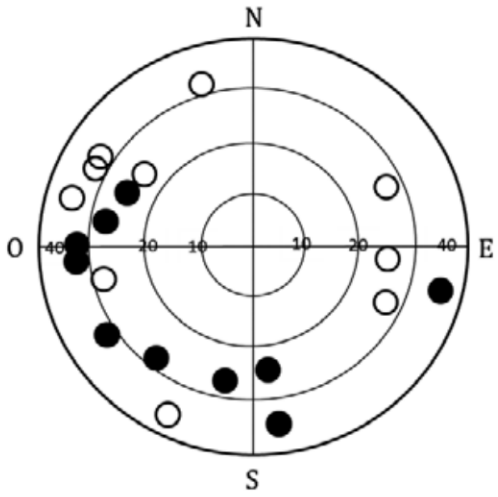


Figura 2.4: Orientación de los nidos de *Furnarius rufus*. Gráfico obtenido de [2].

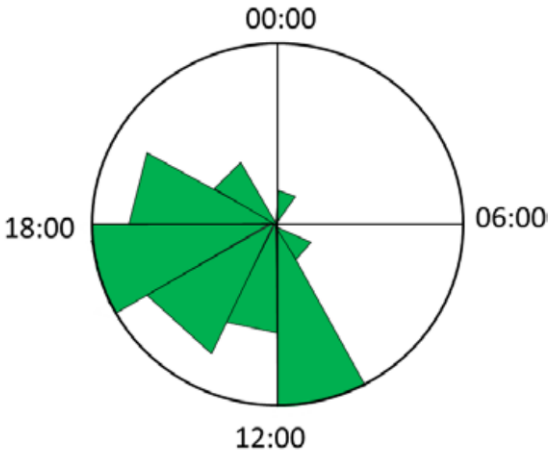


Figura 2.5: Concentración diaria de polen en una ciudad. Gráfico obtenido de [2].

Sean φ_i con $1 \leq i \leq n$ una muestra de n ángulos. Podemos obtener las coordenadas cartesianas en el circulo unitario de la siguiente forma:

$$x_i = cos(\varphi_i),\ y_i = sin(\varphi_i).$$

A continuación podemos obtener las coordenadas cartesianas medias:

$$\bar{x} = \frac{\sum_{i=1}^{n} cos(\varphi_i)}{n},\ \bar{y} = \frac{\sum_{i=1}^{n} sin(\varphi_i)}{n}.$$

Se define el *ángulo medio* $\bar{\varphi}$:

$$\bar{\varphi} = arctan(\frac{\bar{y}}{\bar{x}}),\ si\ \bar{x} > 0, \bar{y} > 0,$$
$$\bar{\varphi} = \pi + arctan(\frac{\bar{y}}{\bar{x}}),\ si\ \bar{x} < 0,$$
$$\bar{\varphi} = 2\pi + arctan(\frac{\bar{y}}{\bar{x}}),\ si\ \bar{x} > 0, \bar{y} < 0.$$

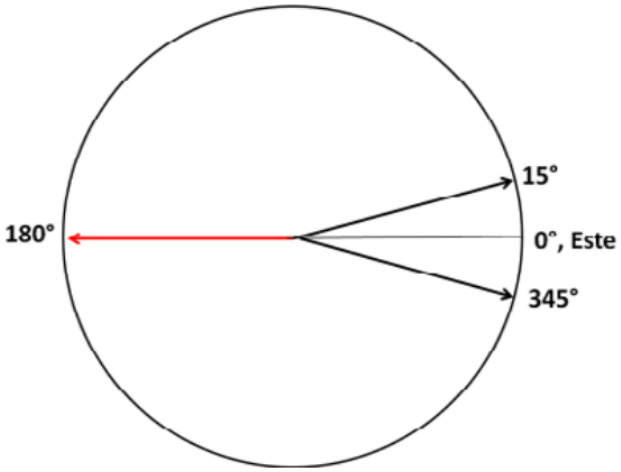


Figura 2.6: Media aritmética de dos ángulos. Gráfico obtenido de [2].

Si $\bar{y} = \bar{x} = 0$ el ángulo medio no está definido.

La *longitud del vector medio* resultante es $R = \sqrt{\bar{x_i}^2 + \bar{y_i}^2}$ y tomará valores comprendidos entre 0 y 1, es decir, $(\bar{x}, \bar{y})$ no estará, en general, en el circulo unitario. Por este motivo, esta media es conocida como *media extrínseca*, puesto que se calcula fuera del círculo y después se "proyecta" al mismo. Para espacios curvados en general existe otra definición alternativa de media, que sería la equivalente a la media en un espacio Euclídeo, ésta se denomina *media de Frechet*, pero queda fuera del alcance de este trabajo [8].

Del desarrollo anterior, se puede suponer que cuando los datos están más dispersos las medias cartesianas tienden a ser más pequeñas y por lo tanto la longitud del vector medio también disminuye. Y por su puesto, en caso de que los datos estén más concentrados, ocurre todo lo contrario. Por esta razón se puede considerar a R como una medida de concentración y de aquí se define la medida de dispersión $V = 1 - R$ como la *varianza circular*, tomando valores entre [0,1].

Hay otras medidas como la *desviación circular estándar*, la *distancia angular*, la *asimetría* o la *curtosis* que vamos a omitir en este trabajo.

2.1.4. Distribuciones de probabilidad

Al igual que en la estadística clásica, es necesario el uso de modelos estadísticos con el fin de conocer el comportamiento de los datos y poder aplicar las técnicas que más se ajusten. A diferencia de las distribuciones lineales, una distribución circular es aquella donde la probabilidad total está concentrada en una circunferencia de radio 1. De esta forma podemos asignar probabilidades a las diferentes direcciones en un rango entre $[0, 2\pi]$ (o $[-\pi, \pi]$). Sea φ un ángulo cualquiera, la función de densidad $f(\varphi)$ de una variable circular tiene las siguientes propiedades:

(I) $f(\varphi) \geq 0$.

(II) $\int_0^{2\pi} f(\varphi)d\varphi = 1$

(III) $f(\varphi) = f(\varphi + 2\pi k), \forall k \in \mathbb{Z}$.

Podemos definir la función de distribución $F(\varphi)$ entre dos ángulos φ_1 y φ_2 de la siguiente manera:

$$F(\varphi_2) - F(\varphi_1) = \int_{\varphi_1}^{\varphi_2} f(\varphi)d\varphi.$$

Estableciendo un ángulo φ_0 como "inicio" podemos definirla como sigue:

$$F(\varphi_1) = \int_{\varphi_0}^{\varphi_1} f(\varphi)d\varphi.$$

Denotaremos por μ, ρ y ν a las correspondientes versiones poblacionales de $\bar{\varphi}$, R y V. Procedemos a presentar las distribuciones circulares más conocidas, reservando una sección para la distribución von Mises.

Distribución uniforme

La distribución uniforme es la análoga a la correspondiente en la estadística en la recta real, donde todos los valores que puede tomar la variable observada tienen la misma "probabilidad" de aparecer. Su función de densidad es:

$$f(\varphi) = \frac{1}{2\pi},\ 0 \leq \varphi < 2\pi.$$

Intengrando obtenemos su función de distribución:

$$F(\varphi) = \frac{\varphi}{2\pi},\ 0 \leq \varphi < 2\pi.$$

Por lo tanto, para $\alpha \leq \beta \leq \alpha + 2\pi$

$$P(\alpha < \varphi \leq \beta) = F(\beta) - F(\alpha) = \frac{\beta - \alpha}{2\pi}.$$

En esta distribución el ángulo medio no está definido, pues la longitud del vector medio resultante es 0 y por lo tanto $\nu = 1$.

Distribución cardioide

Modificando la función de densidad de la distribución uniforme nace la distribución cardioide $C(\mu, \rho)$. El nombre viene de la forma de su función de densidad, que es como la curva cardioide, la generada por la traza de un punto del perímetro de un circulo que rota sobre otro circulo fijo del mismo radio (Figura 2.7). La función de densidad es:

$$f(\varphi) = \frac{1}{2\pi}\left\{1 + 2\rho cos(\varphi - \mu)\right\}, \quad 0 \leq \varphi < 2\pi, \quad |\rho| < \frac{1}{2},$$

y su función de distribución:

$$F(\varphi) = \frac{\rho}{\pi} sin(\varphi - \mu) + \frac{\varphi}{2\pi}, \quad 0 \leq \theta < 2\pi.$$

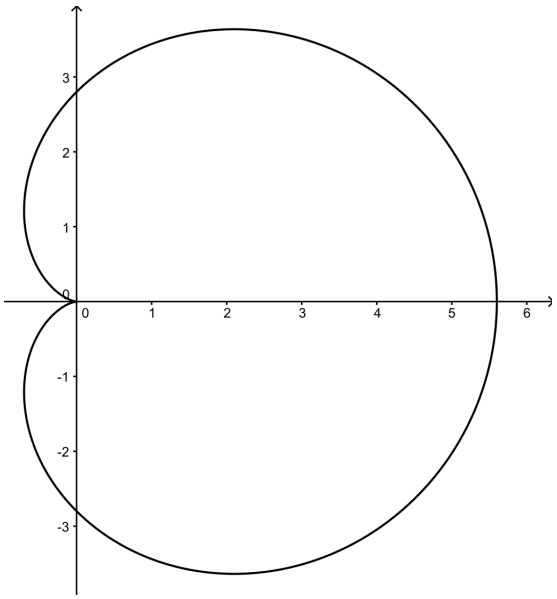


Figura 2.7: Curva cardioide

Distribuciones envueltas (*wrapped*)

Podemos transformar cualquier función de densidad en la línea real "envolviéndola" alrededor de la circunferencia de radio 1.

Sea X una variable aleatoria continua en la recta real con función de densidad $f(x)$. Definimos la variable transformada $X_w = X(mod\ 2\pi)$ y decimos que X_w es una variable aleatoria circular con la distribución envuelta correspondiente. La correspondiente función de densidad envuelta f_w de X_w viene dada por:

$$f_w(\varphi) = \sum_{k=-\infty}^{\infty} [f(\varphi + 2\pi k)], \quad 0 \leq \varphi < 2\pi.$$

Y su función de distribución F_w es:

$$F_w(\varphi) = \sum_{k=-\infty}^{\infty} [F(\varphi + 2\pi k) - F(2\pi k)], \quad 0 \leq \varphi < 2\pi.$$

2.1.5. Distribución von Mises circular

La distribución de von Mises circular es considerada como la distribución normal circular, se trata de una distribución unimodal y simétrica con dos parámetros, μ y κ. Donde μ es la dirección media y κ el parámetro de concentración. Su función de densidad viene dada por:

$$f(\varphi) = \frac{1}{2\pi I_0(\kappa)} exp^{\kappa cos(\varphi - \mu)}, \quad 0 \leq \varphi < 2\pi, \tag{2.1}$$

donde I_0 es la función de Bessel modificada de primer tipo y orden 0:

$$I_0(\kappa) = \frac{1}{2\pi} \int_0^{2\pi} exp^{\kappa cos(\varphi)}\, d\varphi.$$

Notad que las distribuciones von Mises con parámetros $(\mu + \pi, \kappa)$ y $(\mu, -\kappa)$ son la misma distribución, por lo que tomamos siempre $\kappa \geq 0$.

En la Figura 2.8 tenemos la función de densidad para una von Mises con $\mu = 0$ y $\kappa = 10$. Su función de densidad en la recta tiene forma de campana de Gauss, de ahí su relación con la distribución normal.

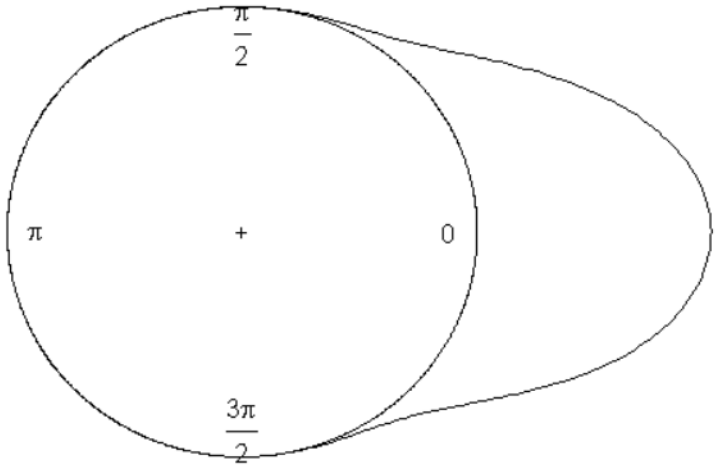


Figura 2.8: Densidad de una distribución von Mises con $\mu = 0$ y $\kappa = 10$

En la Figura 2.8 vemos también que la moda coincide con μ y que el valor más alejado de la moda se encuentra en $\varphi = \mu + \pi$. La proporción entre la densidad de la moda y este otro punto viene dada por $e^{2\kappa}$, de ahí que valores altos en el parámetro κ impliquen mayor concentración de los datos alrededor de la moda o ángulo medio y para $\kappa = 0$ la von Mises equivale a una distribución uniforme. En la Figura 2.9 se muestra la densidad para una distribución von Mises con $\mu = 0$ y $\kappa = 0.5, 1, 2, 4$.

Hay que hacer notar que, aunque la distribución de von Mises se puede definir también de forma general en la hiperesfera unitaria en $\mathbb{R}^d$, en esta memoria utilizaremos únicamente la circular, es decir, la definida en el círculo unitario.

2.2. Mixturas de distribuciones de probabilidad

2.2.1. Introducción

Para poder modelizar nuestras variables de interés vamos a usar modelos de mixturas [9]. Modelizar los momentos de transición de una habitación a otra únicamente con una distribución von Mises no se aproxima a la realidad. Por ejemplo, tiene sentido esperar que la probabilidad de ir de cualquier habitación de la casa a la cocina aumente entorno a las diferentes horas de comer y disminuya en las horas intermedias. En capítulos posteriores veremos esto reflejado

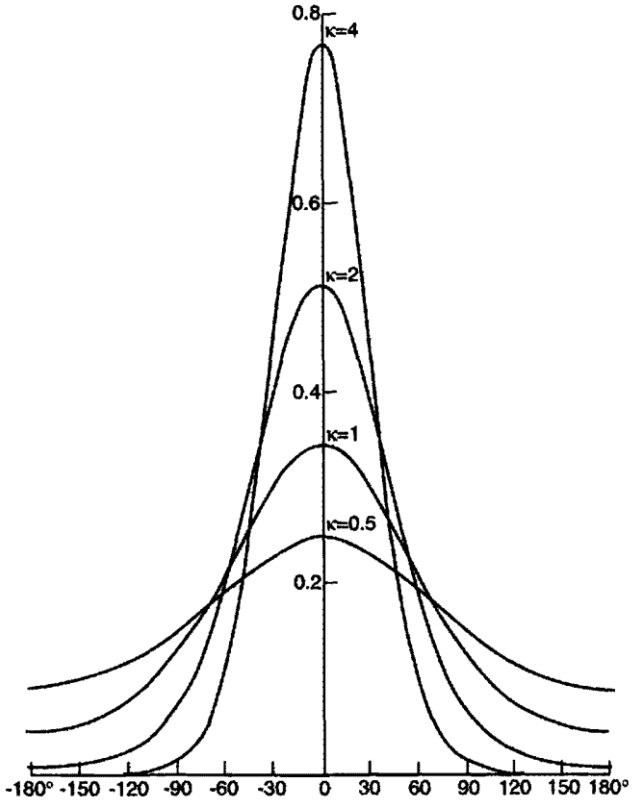


Figura 2.9: Densidad de una distribución von Mises para distintos valores de kappa. Gráfico obtenido de [4].

en los datos, cuando ajustemos la función de densidad correspondiente a la muestra. En esta sección explicaremos que es una mixtura de distribuciones, mostraremos algunos ejemplos para observar como se comporta la función de densidad resultante y comentaremos algunos métodos de estimación de parámetros, profundizando en el que emplearemos nosotros, el conocido algoritmo EM.

Un modelo de mixturas se puede definir como una combinación de distribuciones donde cada una tiene una probabilidad asignada y con el requisito de que la suma de probabilidades de cada distribución sea 1. Es decir, la variable aleatoria que sigue este modelo, puede tomar cualquiera de las distribuciones que componen la mixtura con una probabilidad asignada a cada una. La función de densidad de una mixtura tiene la forma

$$f(x|k, \boldsymbol{w}, \theta) = \sum_{r=1}^{k} w_r f_r(x|\theta_r), \tag{2.2}$$

donde k es el número de distribuciones que componen la mixtura, $\boldsymbol{w} = (w_1, ..., w_k)$ son las probabilidades asignadas a cada una de las k componentes tal que $\sum_{r=1}^{k} w_r = 1$ y θ es el vector con los parámetros que como mínimo tendrá dimensión k. En la Figura 2.10 podemos ver las correspondientes densidades de mixturas de distribuciones normales, primera fila con $k = 2$, segunda fila con $k = 5$, tercera fila con $k = 25$ y última fila con $k = 50$. Vemos como estos modelos presentan la flexibilidad de aproximar situaciones con una estructura compleja.

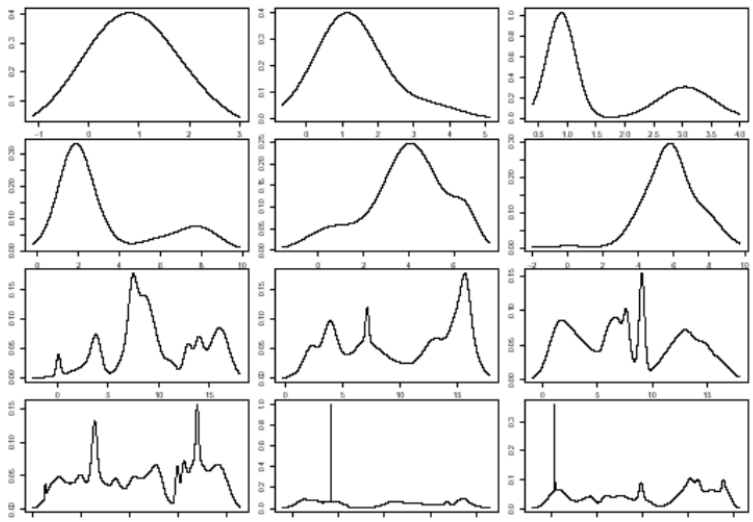


Figura 2.10: Densidades resultantes de mixturas de distribuciones normales con distinto número de componentes.

Mixtura de von Mises

Introducimos a continuación el modelo con el que se trabajará en este proyecto: una mixtura de distribuciones von Mises. En este caso, el vector de parámetros sería $\theta = (\mu_1, \kappa_1, ..., \mu_k, \kappa_k)$ y $f_r(\varphi|\mu_r, \sigma_r^2)$ sería la función de densidad en (2.1), sustituyendo ésta en (2.2) nos queda la función de densidad correspondiente a la mixtura de la siguiente forma:

$$f(\varphi|\theta) = \sum_{r=1}^{k} w_r \frac{exp(\kappa_r cos(\varphi - \mu_r))}{2\pi I_0(\kappa_r)} \quad 0 \leq \varphi < 2\pi. \tag{2.3}$$

Considerando que $\int_{\pi}^{\pi} exp(\kappa cos(\varphi - \mu)) = 2\pi I_0(\kappa)$ con tal de que la probabilidad total sea igual a 1.

En la Figura 2.11 se puede ver el resultado de la función de densidad para $k = 5$ con $\theta = ((-4.6, 1.39), (-29.25, -65.72), (-240.77, 339.36), (-9.96, -5.79), (4.31, -3.78))$ y $\boldsymbol{w} = (0.54, 0.115, 0.2, 0.07, 0.075)$.

2.2.2. Estimación de parámetros. Algoritmo EM

El primer paso que debemos realizar en nuestro proyecto es el entrenamiento del modelo, que en este caso consiste únicamente en estimar los parámetros de las mixturas que modelizan cada una de las posibles transiciones. En concreto, estimar los parámetros de (2.2) que más se ajusten a los datos, es decir: el número k que índica el número de componentes de la mixtura; el vector $\boldsymbol{w} = (w_1, ..., w_k)$ que indica la probabilidad de cada componente y el vector $\theta = (\mu_1, \kappa_1, ..., \mu_k, \kappa_k)$ con los parámetros de cada distribución.

Uno de los métodos más empleados para este fin es el método de máxima verosimilitud

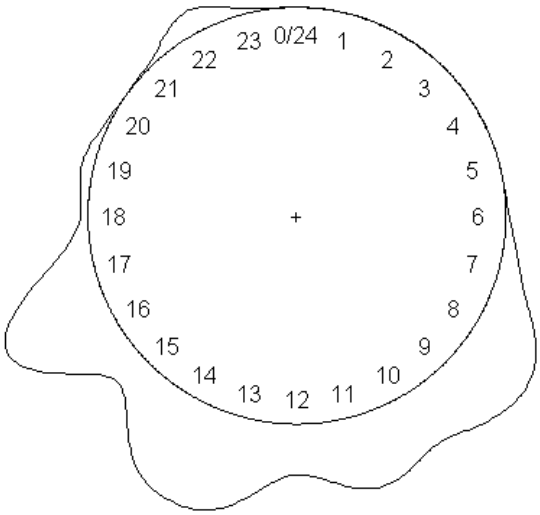


Figura 2.11: Mixtura de von Mises con 5 componentes.

(MLE, del inglés *Maximum Likelihood Estimation*). El método de máxima verosimilitud estima los parámetros del modelo maximizando la verosimilitud de los datos observados.

Sea la muestra $x_1, x_2, ..., x_n$, con x_i independientes y de la misma población, su función de densidad conjunta es

$$f(x_1, ..., x_n|\theta) = f(x_1|\theta)...f(x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta),$$

donde θ es el vector de parámetros de la distribución.

Cuando θ es conocido, la función anterior determina la probabilidad de la muestra. En cambio, cuando la muestra es conocida y el parámetro θ es desconocido, a esta función se le denomina función de verosimilitud y se denota por $l(\theta) = f(x_1, ..., x_n|\theta)$.

El valor que maximiza la función de verosimilitud es el estimador máximo verosímil (EMV), es decir, $EMV(\theta) = \hat{\theta} = \arg\max_\theta l(\theta)$.

El EMV se puede obtener habitualmente mediante cálculo analítico mientras que en otros casos se requiere de optimización numérica. Para resolverlo de forma analítica, si la función de verosimilitud es diferenciable y su máximo no ocurre en un extremo de su dominio de definición, obtendremos el máximo resolviendo el sistema de ecuaciones:

$$\begin{cases} \frac{\partial l(\theta)}{\partial \theta_1} = 0, \\ ... \\ \frac{\partial l(\theta)}{\partial \theta_r} = 0. \end{cases}$$

La mayoría de funciones de verosimilitud tienen una forma exponencial. Esto hace que sea más fácil obtener el EMV hallando el máximo del logaritmo de la verosimilitud, que es lo que se conoce como función soporte y se define como $L(\theta) = log(l(\theta))$.

Pero como hemos comentado, en algunas ocasiones el estimador máximo verosímil no puede obtenerse analíticamente. Por ejemplo, en el caso particular de una distribución de von Mises, la estimación del parámetro μ por máxima verosimilitud se corresponde con la obtención del ángulo medio de la muestra, cuyo procedimiento se ha comentado en la sección 2.1.3. En cambio, la estimación del parámetro κ por máxima verosimilitud resulta bastante complicada, puesto que habría que resolver la ecuación:

$$\frac{I_1(\kappa)}{I_0(\kappa)} = R.$$

donde I_1 es ahora la función de Bessel modificada de primer tipo y orden 1.

En [2] (pag. 74) y en [4] (pags. 85-86) podemos encontrar las siguientes aproximaciones:

$$\begin{aligned}
R < 0,53 \quad & \kappa \approx 2R + R^3 + \frac{5}{6}R^5, \\
0,53 \leq R < 0,85 \quad & \kappa \approx -0,4 + 1,39R + \frac{0,43}{1-R}, \\
R > 0,85 \quad & \kappa \approx \frac{1}{2(1-R) + (1-R)^2 - (1-R)^3}, \\
R > 0,9 \quad & \kappa \approx \frac{1}{2(1-R)}.
\end{aligned}$$

En otras ocasiones maximizar la función de verosimilitud resulta muy costoso computacionalmente, por ejemplo, en el caso de una mixtura de k componentes. La función de densidad viene dada en la Ecuación 2.2 y su función de verosimilitud sería:

$$l(x|\boldsymbol{w}, \theta) = \prod_{i=1}^{n} \sum_{r=1}^{k} w_r f_r(x_i|\theta_r), \tag{2.4}$$

y el número de parámetros a estimar $k(1 + dim(\theta_i))$.

Para disminuir el coste computacional en el contexto de la estimación de parámetros de mixturas se utiliza lo que se conoce como estructura de variables latentes o faltantes y se estiman los parámetros utilizando el algoritmo EM que explicamos a continuación.

Algoritmo EM

El algoritmo EM (del inglés, *Expectation-Maximization*) [10] es un método iterativo para encontrar estimadores máximo verosímiles (locales) de los parámetros de un modelo estadístico, cuando el modelo depende de variables no observables, o también llamadas, latentes. La idea detrás del algoritmo es bastante intuitiva y natural y ha sido ampliamente utilizado en diversas áreas de la ciencia y la ingeniería.

Como hemos comentado, las situaciones habituales de uso del algoritmo EM son aquellas con datos incompletos o faltantes, pero hay otras situaciones donde el problema se puede modelizar a uno más simple asumiendo la existencia de datos faltantes. Este es el caso del contexto de este

proyecto, donde un modelo de mixturas se puede describir de una forma más simple, asumiendo que cada observación tiene su variable latente especificando la componente de la mixtura a la que pertenece.

Sean $y_1, y_2, ..., y_n$ los datos observables, con función de densidad de probabilidad $f(y;\theta)$, donde $\theta = (\theta_1, ..., \theta_d)$ es el vector de parámetros desconocidos. Consideramos $z_1, z_2, ..., z_n$ el conjunto de datos no observables y llamamos $x_i = (y_i, z_i)$ al vector de datos completos (observables y no observables) con función de densidad $f_C(x;\theta)$. Se define la función log-verosimilitud de los datos completos:

$$L_C(\theta) = log(\prod_{i=1}^{n} f_C(x_i|\theta)) = log(\prod_{i=1}^{n} f_C(y_i, z_i|\theta)).$$

Encontrar el EMV requiere maximizar esta función respecto a todos los parámetros, pero esto no es posible porque los valores z_i son desconocidos. Una posibilidad sería maximizar la marginal de los datos observados, pero esto tampoco es posible porque no podemos obtener la distribución marginal debido a que esta depende de los parámetros desconocidos.

El algoritmo EM aborda el problema en dos pasos. El primer paso, denominado paso E, consiste en obtener la función log-verosimilitud **esperada** respecto a la actual distribución de z, dado y junto a una estimación previa $\theta^{(0)}$. El segundo paso, denominado paso M, consiste en **maximizar** la función resultante del paso anterior respecto a θ sobre todo el espacio paramétrico Ω, obteniendo así una nueva estimación $\theta^{(1)}$, que será empleada para realizar el paso E de nuevo. Así que tras establecer un valor previo $\theta^{(0)}$, los pasos E y M se repiten alternadamente hasta que se llega a un punto de convergencia donde la diferencia $L(\theta^{(h+1)}) - L(\theta^{(h)})$ es despreciable. A continuación los formulamos bajo la iteración $(h+1)$:

Paso E: Consiste en el cálculo de:

$$Q(\theta;\theta^{(h)}) = E_{Z\sim f(z|Y,\theta^{(h)})}[L_C(\theta)] = \int_{-\infty}^{\infty} log(\prod_{i=1}^{N} f_C(z, y_i|\theta)) \cdot f(z|y,\theta^{(h)})\, dz.$$

Paso M: Consiste en la maximización de $Q(\theta, \theta^{(h)})$, es decir,

$$\theta^{(h+1)} = \arg\max_{\theta} Q(\theta, \theta^{(h)}).$$

2.2.3. Agoritmo EM para la estimación de los parámetros de mixturas de distribuciones de von Mises

En este apartado nos centramos en nuestro caso particular de mixturas de distribuciones de von Mises con k componentes. Para convertirlo en un problema de datos faltantes definimos el vector de datos faltantes no observables $z_i = (z_{i1}, ..., z_{ik})$ tal que

$$z_{ij} = \begin{cases} 1, \text{ si } y_i \text{ pertenece a la componente j,} \\ 0, \text{ resto.} \end{cases}$$

La función log-verosimilitud completa seria:

$$L_C(\theta) = log(\prod_{i=1}^{n}\sum_{j=1}^{k} z_{ij} w_j f(y_i|\theta_j)) = \sum_{i=1}^{n} log(\sum_{j=1}^{k} z_{ij} w_j f(y_i|\theta_j)),$$

pero teniendo en cuenta que $\forall i$ hay un único j con $z_{ij} = 1$ la podemos expresar:

$$L_C(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{k} z_{ij} log(w_j f(y_i|\theta_j)).$$

Paso E

El paso E es similar al de cualquier otro modelo de mixturas:

$$Q(\theta;\theta^{(h)}) = E_Z[L_C(\theta)|y,\theta^{(h)}] = E_Z[\sum_{i=1}^{n}\sum_{j=1}^{k} z_{ij} log(w_j f(y_i|\theta_j))|y,\theta^{(h)}],$$

y teniendo en cuenta que la función es lineal en los datos no observables:

$$Q(\theta;\theta^{(h)}) = \sum_{i=1}^{n}\sum_{j=1}^{k} log(w_j f(y_i|\theta_j)) E_Z[z_{ij}|y,\theta^{(h)}].$$

Por tanto el paso E solo consiste en el cálculo de $E_Z[z_{ij}|y,\theta^{(h)}]$, es decir la esperanza condicionada de Z_{ij} dado el vector de datos observables, que, por ser una variable de tipo Bernoulli, es la probabilidad a posteriori de que el elemento i-ésimo de la muestra, con valor observado y_i, pertenezca a la componente j de la mixtura, que aplicando Bayes sería:

$$E_Z[z_{ij}|y,\theta^{(h)}] = \frac{w_j^{(h)} f(y_i|\theta_j^{(h)})}{\sum_{r=1}^{k} w_r^{(h)} f(y_i|\theta_r^{(h)})}, \quad i = 1,...,n \; j = 1,...,k.$$

Si llamamos $z_{ij}^{(h)} = E_Z[z_{ij}|y,\theta^{(h)}]$, tendríamos que:

$$Q(\theta;\theta^{(h)}) = \sum_{i=1}^{n}\sum_{j=1}^{k} log(w_j f(y_i|\theta_j)) z_{ij}^{(h)}.$$

Paso M

En este paso tenemos en cuenta que si z_{ij} fuera observable el EMV de las proporciones de cada mixtura $\boldsymbol{w} = (w_1,...,w_k)$ se obtendría simplemente calculando:

$$\hat{w}_j = \sum_{i=1}^{n} \frac{z_{ij}}{n}.$$

Así pues el paso M empieza simplemente estimando para $j = 1,...,k$:

$$w_j^{(h+1)} = \sum_{i=1}^{n} \frac{z_{ij}^{(h)}}{n}.$$

Y a continuación se estiman los parámetros de cada mixtura por separado, cuya estimación ya depende del modelo concreto de mixturas. El caso particular de mixturas de distribuciones von Mises, que es nuestro caso, lo podemos encontrar en [11]. En este artículo, los autores explican la principal función de ajuste para mixturas finitas de distribuciones von Mises-Fisher del paquete de R **movMF** creado por ellos. Este paquete ha sido empleado en la implementación de este proyecto.

Para una componente concreta j, las estimaciones $\mu_j^{(h+1)}$ y $\kappa_j^{(h+1)}$ se obtendría aplicando el procedimiento explicado en la sección 2.2.2, pero usando la siguiente expresión para el cálculo de las coordenadas en el procedimiento de la sección 2.1.3:

$$\overline{v}_j^{(h)} = \frac{\sum_{i=1}^{N} z_{ij}^{(h)} v_i}{\sum_{i=1}^{N} z_{ij}^{(h)}},$$

con $(v_i = cos(y_i), sin(y_i))$.

2.3. Test de bondad de ajuste

Las pruebas o contrastes de bondad de ajuste son empleadas para observar si la muestra difiere mucho de un cierto modelo teórico, es decir, resumen la discrepancia entre los datos esperados y los observados. En nuestro caso serán usados para ver si podemos asumir que nuestros datos siguen un modelo de mixturas de distribuciones von Mises. Si se acepta la hipótesis, podremos asumir que la mixtura ajustada a nuestra transición representa correctamente su comportamiento.

La prueba de Kolmogorov-Smirnov (a partir de ahora K-S) [12] será la empleada en este proyecto. Otro contraste de bondad de ajuste ampliamente empleado es el contraste de Chi-cuadrado, pero al querer contrastar una población que sigue un modelo de variable continua es más conveniente emplear K-S. Esta prueba compara las frecuencias acumuladas observadas con las frecuencias acumuladas esperadas, es decir, las obtenidas del modelo que queremos contrastar. El contraste de hipótesis tiene la siguiente forma:

$$\begin{cases} H_0 : X \backsim mixVM(\boldsymbol{w}, \varphi),\ donde\ \boldsymbol{w} = (w_1, ..., w_r), \varphi = (\mu_1, \kappa_1, ..., \mu_r, \kappa_r), \\ H_1 : \text{no } H_0. \end{cases}$$

El primer paso del procedimiento es ordenar la muestra que queremos contrastar, por ende, esta prueba solo puede utilizarse para variables numéricas. Una vez tengamos la muestra ordenada obtenemos la frecuencia acumulada de cada observación. Esto es, para una muestra $(x_1, ..., x_n)$ de tamaño n, $F(x_i) = \frac{1}{n} i, \quad i = 1, ..., n$. Continuamos calculando las frecuencias acumuladas bajo la hipótesis nula H_0, $F_0(x_1), ..., F_0(x_n)$. Para una mixtura de von Mises:

$$F_0(x_i) = \sum_{j=1}^{r} w_j \int_0^{x_i} f(x_i|\mu_j, \kappa_j), \tag{2.5}$$

Ahora obtenemos el estadístico de contraste $D = sup|F(x_i) - F_0(x_i)|$ y la región de rechazo accediendo a las tablas $R = \{D : D \geq k\}$ donde k cumple $P(D > k) = \alpha$ siendo α el nivel de significación escogido. Para un nivel de significación de 0.05 y un tamaño muestral mayor que 40, $k = 1.36/\sqrt{n}$. Si el estadístico de contraste D se encuentra fuera de la región de rechazo podremos asumir que el modelo está bien ajustado.

2.4. Algoritmo CUSUM

El principal objetivo de este proyecto es detectar anomalías en el comportamiento de una persona. Hemos modelizado el comportamiento de la persona con una serie de mixturas de von Mises, por lo que un cambio en el comportamiento se traduce en un cambio en los parámetros de alguna o algunas mixturas. Existe una amplia variedad de algoritmos de detección de cambio, pero en este trabajo nos limitaremos al método CUSUM. Entre el resto de algoritmos seguro que hay otros perfectamente válidos o incluso mejores que CUSUM en este contexto. El estudio e implementación de estos algoritmos se propone como trabajo futuro del proyecto. La documentación respecto a CUSUM empleada para este proyecto se encuentra en [13]. En [14], el algoritmo CUSUM se ha usado en un problema de detección de anomalías muy parecido al planteado en este proyecto.

La metodología de detección de cambios puede realizarse en línea (en inglés, *on-line*) o fuera de línea (en inglés, *off-line*). La detección se realiza en línea cuando se desea detectar la aparición del cambio lo antes posible, en esta metodología se monitoriza continuamente y se detiene cuando el estadístico de monitorización cumple cierta regla de parada. En la detección fuera de línea se toma una muestra completa y se detecta a posteriori. El tipo de algoritmos usados para detectar anomalías en el comportamiento son, habitualmente, del tipo fuera de línea.

Consideramos una muestra $(x_1, ..., x_n)$ con función de densidad de probabilidad $f_{\theta_0}(x)$, donde cada observación corresponde a un momento de transición entre dos habitaciones concretas, o dicho de otra forma, una muestra asociada a una de las mixturas de nuestro modelo, como puede ser la que representa la transición entre el dormitorio y el baño. Se trata de observaciones obtenidas de la monitorización de la persona, que si no ha sufrido ninguna alteración, siguen el correspondiente modelo de mixtura que ha sido ajustado en el entrenamiento.

Supongamos que conocemos el valor del parámetro θ_1 tras la alteración, el método de detección de anomalías CUSUM se basa en definir las sumas:

$$S_n = \sum_{i=1}^{n} s_i, \text{ donde } s_i = ln\frac{f_{\theta_1}(x_i)}{f_{\theta_0}(x_i)}. \tag{2.6}$$

Notar que S_n puede interpretarse como el logaritmo del cociente de las verosimilitudes en θ_0 y en θ_1.

Se puede intuir que si la observación x_i es más probable bajo θ_0 el valor de s_i será negativo, y en caso contrario positivo. En virtud de esto, si tenemos una muestra con un cambio en un instante $j \in \{1, 2, ..., n\}$, S_n tendrá una tendencia decreciente antes del cambio y una tendencia creciente después. En la Figura 2.12 podemos observar un ejemplo donde se ha introducido

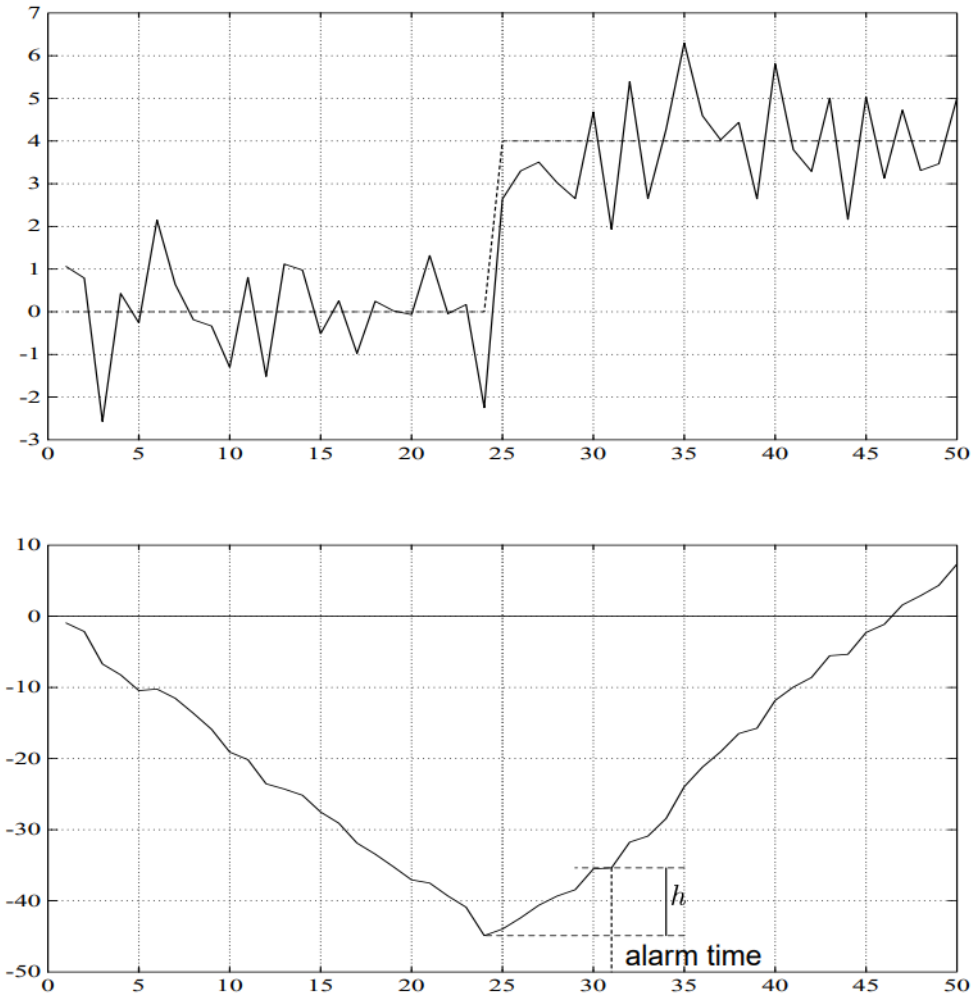


Figura 2.12: Ejemplo de la evolución de S_n tras un cambio en la media de una variable Gaussiana.

un cambio en la media de una secuencia Gaussiana a partir de la observación 24. En la parte superior vemos los valores que toma la variable y en la parte inferior la evolución de S_n.

En este caso, el algoritmo CUSUM se basa en buscar ese cambio de tendencia, pero como también podemos observar, aunque la tendencia sea decreciente, hay fluctuaciones positivas. Por esta razón es necesario definir un umbral h para aceptar un cambio en los parámetros, que habitualmente se busca a partir de la experimentación.

En nuestro problema de detección de anomalías, el parámetro θ_1 es desconocido y por tanto no podemos estudiar el comportamiento de S_n. Además, consideramos que es importante obtener el valor del umbral de una manera no visual sino basada en probabilidades. Por este motivo, plantearemos el método de CUSUM basado en un contraste de hipótesis. En particular, vamos a plantear el problema como un problema de contrastes de hipótesis múltiples.

Supongamos que queremos contrastar que el instante de cambio es $j \in \{1, 2, ..., n\}$, es decir, un cambio en los parámetros a partir de la observación x_j, antes del cambio el vector de parámetros θ de la mixtura es θ_0 conocido, y después del cambio un $\theta_1 \neq \theta_0$ desconocido. El

contraste de hipótesis es el siguiente:

$$\begin{cases} H_0 : \theta^{(i)} = \theta_0 \quad i = 1, ..., n \\ H_1 : \theta^{(i)} = \theta_0 \quad i = 1, ..., (j-1); \quad \theta^{(i)} \neq \theta_0 \quad i = j, ..., n, \end{cases} \tag{2.7}$$

donde $\theta^{(i)}$ denota el vector de parámetros de la mixtura en la observación i-ésima.

Ahora consideramos el estadístico de la razón de verosimilitud generalizado asociado al contraste 2.7:

$$\Lambda_1^n(j) = \frac{\prod_{i=1}^{j-1} f_{\theta_0}(x_i) \prod_{i=j}^{n} f_{\theta_1}(x_i)}{\prod_{i=1}^{n} f_{\theta_0}(x_i)} = \frac{\prod_{i=j}^{n} f_{\theta_1}(x_i)}{\prod_{i=j}^{n} f_{\theta_0}(x_i)},$$

y dos veces su logaritmo:

$$S_j^n = 2 \sum_{i=j}^{n} ln \frac{f_{\theta_1}(x_i)}{f_{\theta_0}(x_i)} = 2 \sum_{i=j}^{n} (ln(f_{\theta_1}(x_i)) - ln(f_{\theta_0}(x_i))). \tag{2.8}$$

Como el parámetro θ_1 es desconocido, en el test de la razón de verosimilitudes tenemos que sustituirlo por su estimador máximo verosímil bajo H_1, es decir, el EMV obtenido a partir de las observaciones entre j y n.

La región de rechazo del test de la razón de verosimilitudes para un nivel de significación α sería, asumiendo independencia:

$$R_1(j) = \{S_j^n : S_j^n >_{1-\alpha} \chi_p^2\},$$

siendo ${}_{1-\alpha}\chi_p^2$ el percentil $1-\alpha$ de una distribución Chi-cuadrado con p grados de libertad, con p la dimensión del vector de parámetros en los que se quiere detectar el cambio.

El contraste anterior lo podemos plantear para cada posible instante de cambio j, es decir, tendremos un estadístico de contraste S_j^n y su correspondiente $\hat{\theta}_1^{(j)} = EMV(\theta_1)$ para $j = 1, ..., n$.

Supongamos que el instante de cambio real es τ, si calculamos S_j^n con $j < \tau$ tendremos que s_i será menor que 0 generalmente para $i = j, ..., (\tau - 1)$, ya que se tratan de observaciones más ajustadas a θ_0, por lo que $S_j^n < S_\tau^n$. Por otro lado, si calculamos S_j^n con $j > \tau$, es trivial que $S_\tau^n = \sum_{i=\tau}^{j-1} s_i + S_j^n$ y s_i será mayor que 0 generalmente para $i = \tau, ..., j$, ya que se tratan de observaciones más ajustadas a θ_1, por tanto $S_j^n < S_\tau^n$. En la sección 3.3 encontramos los resultados donde se plasma lo que acabamos de comentar.

Una forma meramente descriptiva de determinar el instante de cambio τ sería observando el gráfico. El cambio de tendencia que se aprecia parece suficiente como para determinar que ha ocurrido un cambio en el punto máximo. Pero también se observan algunas fluctuaciones previas que en algunos contextos podrían dar lugar a conclusiones erróneas, de ahí el interés en encontrar un umbral h que nos permita tomar la decisión correcta con la máxima precisión posible.

Si consideramos el instante de cambio como desconocido, el estadístico de contraste de la razón de verosimilitudes sería:

$$\Lambda^n = sup_j S_j^n.$$

Para obtener el valor del umbral h deberíamos conocer la distribución de probabilidad de este estadístico bajo H_0, pero desafortunadamente esta distribución no es, en general, conocida.

En [15] podemos encontrar valores asintóticos bajo ciertas condiciones sin asumir independencia. Como su complejidad queda fuera del alcance de este proyecto, vamos a simplificar el problema asumiendo independencia y simplemente vamos a aproximar el valor del umbral usando la desigualdad de Boole (la utilizada en la corrección tipo Bonferroni) si fijamos un nivel de significación α, tenemos:

$$\alpha = P(\Lambda^n > h) = P(\bigcup_{j=1}^{N} \{S_j^n > h\}) \leq \sum_{j=1}^{n} P(S_j^n > h) = nP(S_j^n > h) = nP(\chi_p^2 > h),$$

tomando h como el percentil $1-\alpha_0/n$ de una distribución Chi-cuadrado con p grados de libertad:

$$\alpha_0/n = P(\chi_p^2 > h),$$

tendríamos garantizado que $\alpha \leq \alpha_0$.

Como se trata de una aproximación, en el siguiente capítulo comprobaremos con datos simulados que funciona bastante bien en la práctica y es suficiente para nuestro problema.

Una vez aceptado que ha habido un cambio, el Estimador Máximo Verosímil del instante de cambio es:

$$\tau = \arg\max_j S_j^n.$$

Capítulo 3

Implementación del modelo

3.1. Datos

Lo ideal para la obtención de los datos necesarios para nuestro proyecto sería una monitorización 100 % efectiva en la que supiéramos sin error alguno todos los movimientos que hace una persona en su hogar. El problema reside en que conseguir ésto sin violar por completo la privacidad de una persona es bastante complicado o imposible. La consecuencia de esto es un número muy reducido de *datasets* públicos disponibles. Afortunadamente en [1] tenemos diversos *datasets* con los que poder trabajar. Entre todos los disponibles se escogió el conjunto de datos denominado Milán, por diversos motivos, aunque también presenta algunos inconvenientes. Hay que tener en cuenta que la mayoría de estos *datasets* fueron recogidos para un objetivo diferente al nuestro, con ellos se pretendía estudiar las actividades del día a día. Por ello los datos incluyen los inicios y finales de un conjunto de actividades predefinido. El *dataset* viene acompañado de un plano del hogar con la ubicación de los sensores y de un fichero explicando brevemente sus características. En la Figura 3.1 podemos ver el plano.

Uno de los principales motivos por los que se escogió este *dataset* fue el número reducido de habitaciones que presenta. Cuantas más habitaciones el número de transiciones posibles crece y en consecuencia hay que trabajar con más mixturas. Por otro lado, el *dataset* contiene suficientes observaciones con las que trabajar, aunque hay un inconveniente que implica un depurado exhaustivo que reduce bastante el número de observaciones. Este inconveniente es que la persona monitorizada convive con un perro y que sus hijos la visitaban frecuentemente. Esto produce mucha suciedad en los datos, ya que por factores externos se producía la activación de ciertos sensores en otros lugares de la casa.

Los datos consisten en una tabla de activaciones y desactivaciones de sensores ordenadamente por hora y día. Cada observación esta compuesta por:

- Fecha
- Hora
- Id del sensor: en la Figura 3.1 se pueden ver el id de cada sensor.

- Activación o desactivación: campo que indica se la observación se corresponde a la activación o desactivación de un sensor.
- Actividad: Indica la actividad que comienza o finaliza en esa observación, o nada en caso de que no comience ni finalice ninguna actividad.
- Inicio o fin: En caso de que haya un valor en el campo de "Actividad" indica si se corresponde al inicio o al final.

En el cuadro 3.1 podemos ver una representación de los datos con su formato.

Fecha	Hora	ID	On-Off	Actividad	Inicio-Fin
2009-10-16	00:01:04.000059	M028	OFF		
2009-10-16	00:01:06.000046	M021	ON	Bed_to_Toilet	Inicio
2009-10-16	00:01:07.000064	M013	ON		
2009-10-16	00:01:08.000081	M028	ON		
2009-10-16	00:01:09.000028	M013	OFF		
...	...	...	...	...	...
2009-10-16	00:08:50.000081	M028	OFF		
2009-10-16	00:08:55.000040	M025	ON	Bed_to_Toilet	Fin
2009-10-16	00:09:23.000032	M013	ON		

Cuadro 3.1: Representación de los datos del dataset Milán.

La suciedad de los datos mencionada previamente, junto a la necesidad de asociar a cada sensor su habitación correspondiente, implica un proceso de depurado por pasos que será explicado en la próxima sección.

3.2. Entrenamiento

El entrenamiento de nuestro modelo está compuesto por tres fases principales. La primera es la fase de depurado, donde preparamos nuestros datos para la siguiente fase de estimación de parámetros. Necesitaremos los instantes donde se produce una transición entre dos estancias en habitaciones distintas. Consideraremos una estancia cuando el tiempo que permanece la persona monitorizada en su habitación supera los 2 minutos. Una vez tengamos todas las transiciones bien clasificadas podremos estimar los parámetros de cada mixtura y luego someter los modelos generados a un test o contraste de bondad de ajuste con el que comprobaremos que se ajusten bien a los datos.

3.2.1. Depurado

Como se ha mencionado previamente, el depurado debe transformar los datos con el formato en el cuadro 3.1 en conjuntos de transiciones. Cada conjunto de transiciones corresponderá a una transición entre dos habitaciones y por lo tanto de cada conjunto obtendremos una mixtura

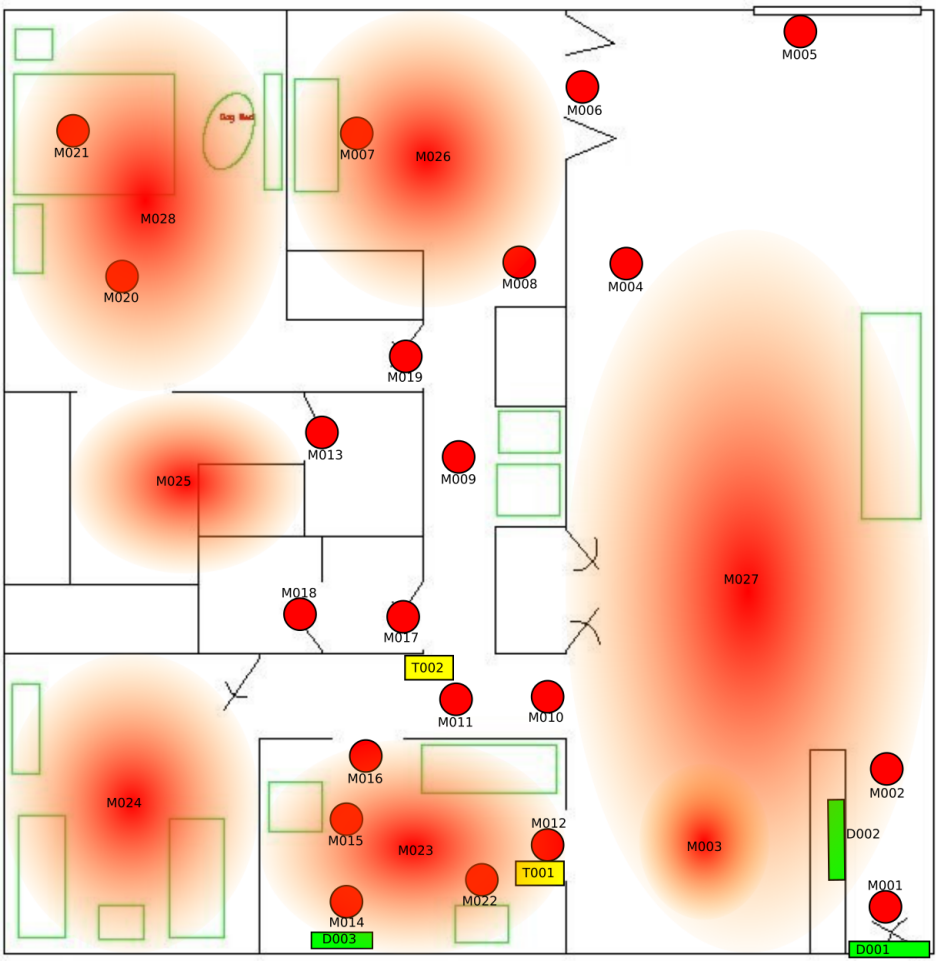


Figura 3.1: Plano con la disposición de los sensores de donde se obtuvieron los datos para el proyecto.

que lo modelice. En concreto, trabajaremos con 5 habitaciones: dormitorio, baño, cocina, salón y televisión. En el plano de la Figura 3.1 podemos ver más habitaciones que las mencionadas. Con la finalidad de reducir el número de transiciones se ha decidido unir los dos baños en uno y se ha eliminado una habitación donde no había prácticamente estancias.

El proceso de depurado consistirá en los siguientes pasos:

- **Paso 1:** Asociar a cada observación su habitación correspondiente.
- **Paso 2:** Eliminar todas las observaciones que no pertenezcan a una estancia de al menos 2 minutos.
- **Paso 3:** Clasificar las transiciones por ficheros.

En la sección A.2 del Anexo se encuentra una explicación más extendida del procedimiento de depurado junto a parte del código implementado. El resultado son 20 ficheros, uno para cada transición (5×4 transiciones posibles), con todos los instantes en los que ha ocurrido esa transición. Los instantes se han transformado de hora:minutos a radianes y de radianes a puntos en el círculo unitario, ya que es el formato que emplea el paquete **movMF** para ajustar una mixtura de von Mises.

Por último, mencionar que en primer lugar se implementó el depurado de los datos con el lenguaje R, pero el resultado fue una proceso muy lento debido a que no se consideraron las propiedades de R como lenguaje. En concreto, R tiene carencias debido a que debe comprobar el tipo de la variable cada vez que accede a ella. Esto produce un gran coste a la hora de realizar tareas repetitivas como hacer una acción dentro de un bucle. La solución a esto es transformar el código para trabajar con operaciones vectorizadas, ya que todos los elementos de un vector tienen el mismo tipo.

Como alternativa decidí emplear Julia para la implementación de todo el modelo. Julia es un lenguaje orientado a la obtención de códigos de alto rendimiento y eficiencia, el cuál no requiere vectorizar el código para obtener buenos resultados. Combina la velocidad de C, el dinamismo de Ruby, la usabilidad de Python, la estadística de R y el álbegra lineal de Matlab. Tiene el inconveniente de que no existen tantas librerías y paquetes como en el entorno de R, pero afortunadamente se pueden hacer llamadas a éste desde Julia.

3.2.2. Estimación de parámetros

Como mencionamos en la sección 2.2.2, haremos uso del paquete de R **movMF** para ajustar a cada conjunto de transiciones una mixtura de von Mises mediante el algoritmo EM. El paquete incluye funciones para obtener la densidad, calcular la probabilidad acumulada, hacer simulaciones y ajustar los parámetros de una mixtura de distribuciones von Mises.

Para estimar los parámetros haremos uso de la función *movMF()* a la cual le pasaremos como argumentos el conjunto de datos, el número de componentes que queremos que tenga la mixtura y el número de ejecuciones que deseamos que se realicen para obtener el mejor ajuste. El método permite otras opciones, como especificar la variante del algoritmo EM que deseamos

emplear o escoger el método para resolver el parámetro de concentración. Nos limitaremos a emplear las opciones por defecto y se propone el estudio del resto para trabajos futuros.

En la Figura 3.2 podemos ver el código con el que obtenemos, para el mismo conjunto de datos, mixturas de 2 a 6 componentes. Para luego escoger la mixtura con mejor BIC (del inglés, *Bayesian Information Criterion*) [16], un criterio para la selección de un conjunto de modelos sobre los mismos datos.

```
library(movMF)
vMs = lapply(2:6, function(K) movMF(as.matrix(coords), k=K, control=list(nruns = 30)))
options = sapply(vMs, BIC) # BIC para cada mixtura
options_sorted = sort(options,index.return=TRUE)
id_sorted = options_sorted$ix # El primer valor son los componentes-1 de la mejor mixtura
vM = movMF(as.matrix(coords), k=id_sorted[1]+1, control=list(nruns = 30))
```

Figura 3.2

La convergencia del algoritmo EM no nos garantiza llegar a la solución óptima. De hecho, como veremos en la próxima sección, hay tres casos donde no encontraremos una mixtura que ajuste bien los datos. Esto se debe a que son las únicas tres transiciones con menos de 40 observaciones. En estos tres casos el algoritmo de estimación converge pero obtenemos componentes con muy pocos datos y la estimación del parámetro de concentración de estas componentes o birn resulta muy elevado o bien no se supera el test de bondad de ajuste. En la sección A.3 del Anexo se encuentra el código implementado.

3.2.3. Test de bondad de ajuste

Por último, para concluir con el entrenamiento y almacenar los parámetros de nuestro modelo, tenemos que comprobar que las mixturas se ajustan correctamente. Para ello se tuvo que implementar en Julia, junto a llamadas al entorno de R, el test de Kolmogorov-Smirnov, debido a que no existe ninguna librería ni en Julia ni en R que tenga implementado este test para mixturas de distribuciones von Mises. En la práctica, este paso está combinado con el paso de estimación de parámetros, de forma que cuando obtenemos la mixtura con mejor BIC, la sometemos al test de bondad de ajuste, y en caso de no pasarlo, obtenemos la siguiente mixtura con mejor BIC.

Comentar que para obtener las frecuencias acumuladas bajo el modelo estimado $F_0(x_i)$ se tuvieron que obtener las probabilidades acumuladas de cada componente usando el paquete circular [17] y promediar, puesto que el paquete **movMF** no las lleva implementadas. Es importante también comentar que para poder calcular las frecuencias acumuladas ninguna de sus componentes debe tener un parámetro de concentración muy elevado. En la máquina que empleamos el límite de *kappa* se estableció en 700, ya que cuando este era superior la máquina no conseguía obtener las frecuencias acumuladas.

Como se ha mencionado previamente, de las 20 transiciones posibles, hubieron 3 que no se consiguieron ajustar por que tenían un tamaño de muestra pequeño, por lo que el algoritmo no conseguía converger a unos parámetros aceptables. Las mixturas a las que conseguía converger tenían alguna componente con un parámetro de concentración mayor a 700 o no pasaban el test de bondad de ajuste. Esto implica no poder detectar anomalías en estas tres transiciones, que

fueron: dormitorio-tv, cocina-dormitorio, tv-dormitorio. Todo el código implementado para la estimación de parámetros y los test de bondad de ajuste se encuentran en A.3.

3.3. Detección de anomalías

Como se ha comentado en el Capítulo 1, por cuestión de tiempo, no nos es posible trabajar con datos reales de sujetos monitorizados, y menos aún sujetos con alguna anomalía. Por este motivo, para comprobar la correcta implementación del método CUSUM explicado en la sección 2.4 nos vamos a tener que basar en simulaciones. Tras generar una secuencia de transiciones con los parámetros estimados, la idea es introducir un cambio en los parámetros de alguna mixtura y volver a generar una secuencia que adjuntaremos a la anterior.

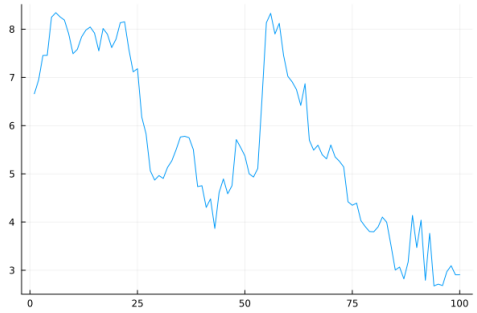
A continuación se presentan algunos de los resultados obtenidos. Con estos resultados se pretende demostrar que nuestra aproximación funciona correctamente dentro de nuestro contexto. Se han realizado pruebas para distintos tamaños muestrales, cambiando los parámetros a los que se le introduce el cambio e introduciendo cambios grandes y leves. Los resultados los representaremos con los gráficos obtenidos del cálculo de S_j^n para $j = 1, ..., n$.

Asumimos que se pueden producir cambios en las proporciones y en los parámetros de las componentes de la mixtura. También vamos a asumir que no se puede producir un cambio en el número de componentes de la mixtura. La transición escogida para realizar las pruebas es "salon-cocina" ya que era la que tenía un tamaño muestral más grande para la estimación de sus parámetros. La mixtura correspondiente a esta transición tiene 3 componentes, por lo que para el calculo del umbral haremos uso de una Chi-cuadrado con 5 grados de libertad. También mencionar que todas las pruebas se han realizado para un nivel de significación de 0.05.

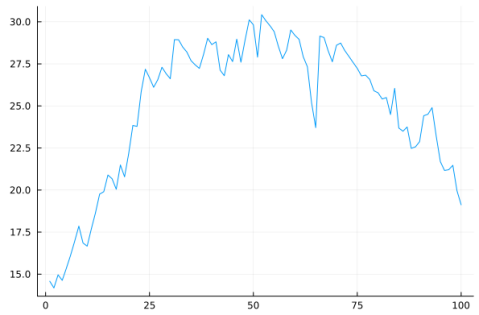
Comenzamos con los resultados para un tamaño muestral $n = 100$ con el cambio introducido en la observación 50. El umbral correspondiente es $h = 22.1$. En la Figura 3.3a tenemos el caso en el que no hay ningún cambio introducido. Se puede observar como no se supera el umbral para ningún $j = 1, ..., n$ como era de esperar. A la derecha, en la Figura 3.3b encontramos el resultado de modificar las proporciones de las componentes. Vemos como se supera el umbral a partir antes de $j = 25$ y que se alcanza el máximo en la observación 52.

Finalmente, en la segunda fila, tenemos los resultados de modificar la media de una componente. En la Figura 3.3c tenemos el resultado de sumar 0.3 radianes a la media, mientras que en la Figura 3.3d sumamos 0.8 radianes a la media. En ambos casos se supera el umbral necesario para considerar que ha tenido lugar un cambio, pero podemos apreciar que cuando el cambio es mayor el umbral se supera por mucho más. En ambos casos el máximo se alcanza entorno al momento de cambio.

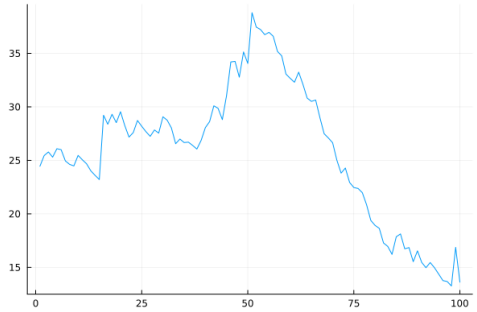
Continuamos con los resultados para un tamaño muestral $n = 50$ con el cambio introducido en la observación 35. El umbral correspondiente es $h = 20.5$. Al igual que en el caso anterior, en la Figura 3.4a podemos ver que el comportamiento es el esperado cuando no se introduce ningún cambio. También obtenemos un resultado satisfactorio en la Figura 3.4b cuando modificamos las proporciones de las componentes, superando el umbral y alcanzando el máximo entorno al momento de cambio.



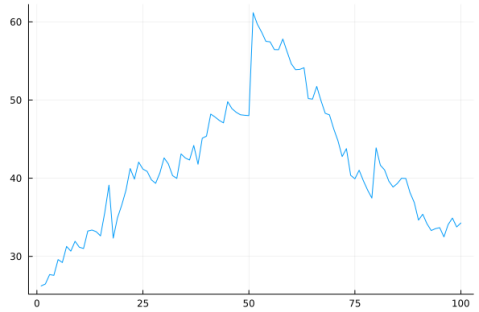
(a) Ningún cambio introducido.



(b) Modificación de las proporciones de las componentes.



(c) Cambio leve (+0.3 radianes) en la media de una componente.



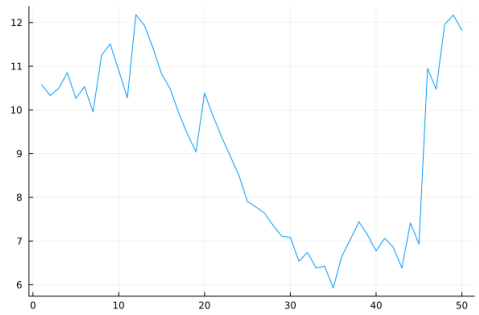
(d) Cambio grande (+0.8 radianes) en la media de una componente.

Figura 3.3: Distintos tipos de pruebas para un tamaño muestral de 100.

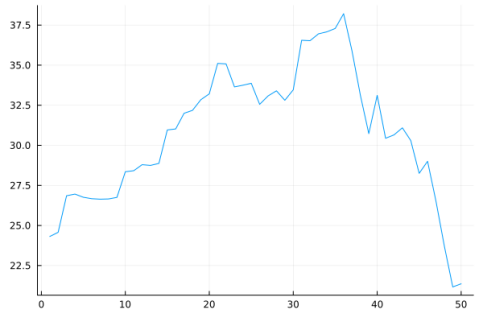
En la segunda fila tenemos el mismo contexto que para $n = 100$. En la Figura 3.4c también obtenemos un resultado satisfactorio. En cambio, en la Figura 3.4d, cuando el cambio es leve, vemos como apenas se supera el umbral para $j = 50$. Dado este resultado, una buena práctica sería seguir monitorizando para obtener un tamaño muestral mayor y volver a realizar la prueba.

En último lugar, los resultados para un tamaño muestral $n = 25$ con el cambio introducido en la observación 15. El umbral correspondiente es $h = 19.8$. Como era de esperar, para un tamaño muestral tan pequeño, los resultados no son tan satisfactorios como los anteriores, pero en general siguen siendo aceptables. Cuando no introducimos ningún cambio, como en la Figura 3.5a, el comportamiento es el esperado. Por otro lado, en la Figura 3.5c, cuando introducimos un cambio leve en la media de una componente, no se supera el umbral en ningún momento, por lo que no se detectaría el cambio. Para un cambio grande en la media (Figura 3.5d) y un cambio en las proporciones (Figura 3.5b) obtenemos resultados satisfactorios.

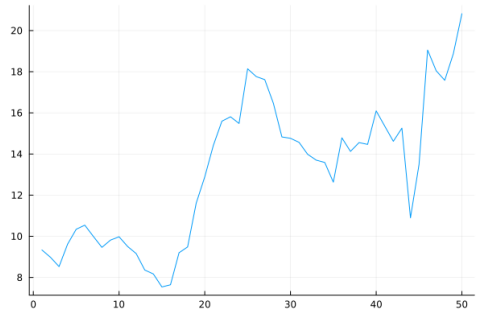
Los resultados mostrados anteriormente son una pequeña selección de un conjunto mucho más grande. En general, podemos asumir que el método detecta los cambios de forma fiable cuando el tamaño muestral no es pequeño. También consideramos que es importante considerar el factor visual para asumir el cambio o no, ya que en algunos resultados se ha superado el umbral pero el gráfico no se comportaba como lo esperado. En estas situaciones lo mejor es seguir monitorizando y realizar la prueba con una tamaño muestral mayor. En el Anexo A.4 se muestra una parte del código implementado.



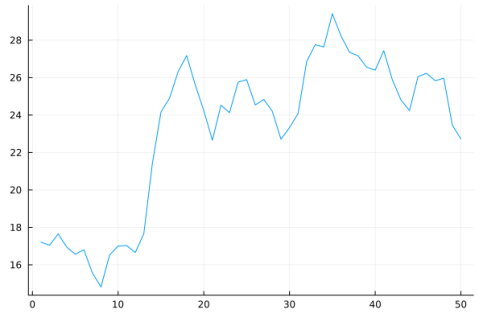
(a) Ningún cambio introducido.



(b) Modificación de las proporciones de las componentes.

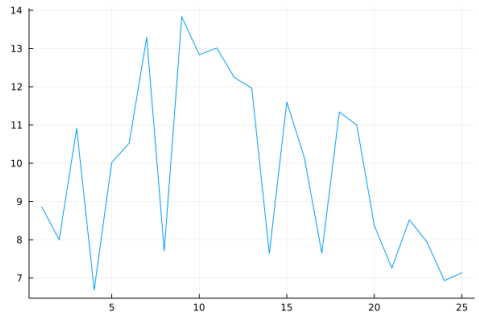


(c) Cambio leve (+0'3 radianes) en la media de una componente.

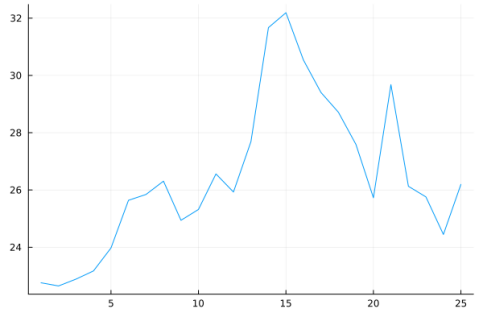


(d) Cambio grande (+0.8 radianes) en la media de una componente.

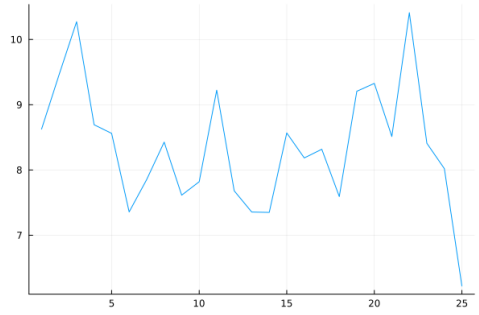
Figura 3.4: Distintos tipos de pruebas para un tamaño muestral de 50.



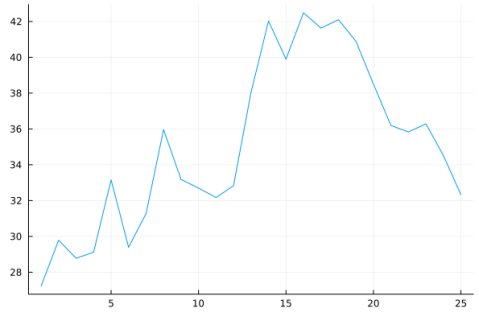
(a) Ningún cambio introducido.



(b) Modificación de las proporciones de las componentes.



(c) Cambio leve (+0.3 radianes) en la media de una componente.



(d) Cambio grande (+0.8 radianes) en la media de una componente.

Figura 3.5: Distintos tipos de pruebas para un tamaño muestral de 25.

Capítulo 4

Conclusiones

Para concluir, en esta última sección se presentan los resultados finales obtenidos junto a los posibles trabajos futuros a desarrollar. Además se incluyen las conclusiones obtenidas en el ámbito formativo y en el personal.

El trabajo realizado en esta memoria incluye todos los objetivos propuestos inicialmente. Como objetivo general tenemos la detección de posibles discrepancias y anomalías en el comportamiento de una persona dentro de su hogar, para el cual ha sido necesario la búsqueda y depurado de un *dataset* apto para nuestro contexto, el diseño de modelos de comportamiento mediante mixturas de distribuciones von Mises y la estimación de sus parámetros, la implementación de un test de bondad de ajuste y finalmente la implementación del método para la detección de anomalías. En cuanto a su aplicación práctica, como se ha comentado, aunque se hayan encontrado una serie de inconvenientes, como la falta de convergencia del algoritmo EM en tamaños muestrales pequeños, que habría que mejorar, estamos satisfechos y podemos afirmar que el método detecta los cambios de forma fiable cuando el tamaño muestral es mediano o grande.

Respecto a los posibles trabajos futuros, en primer lugar se propone la implementación de un sistema de monitorización en interiores. Un sistema ajustado a nuestras necesidades nos proporcionaría datos más limpios y nos permitiría obtener muestras de mayor tamaño con las que poder estimar mejor los parámetros de las mixturas. Por otro lado, se propone estudiar y probar las diversas alternativas que ofrece el paquete **movMF**, en concreto otras variantes del algoritmo EM disponibles y las opciones de estimación del parámetro de concentración. También se propone estudiar otras alternativas para el algoritmo de detección de anomalías debido a la gran variedad que existe de este tipo de algoritmos. Finalmente, también es de interés la posibilidad de hacer uso de una estructura que nos permita estudiar la probabilidad de una secuencia de transiciones, como puede ser una cadena de Markov. Con esto podríamos buscar anomalías desde una perspectiva más general y no desde una transición entre dos habitaciones concretas.

En el aspecto formativo, este proyecto me ha brindado la oportunidad de adquirir y aplicar diversas herramientas y metodologías que han enriquecido mi formación académica. Durante el desarrollo de este trabajo, he tenido la posibilidad de familiarizarme en el campo de la

estadística circular, que me ha permitido comprender mejor el comportamiento de los datos de naturaleza circular. Asimismo, he tenido la oportunidad de familiarizarme con el algoritmo EM para la estimación de parámetros, una herramienta fundamental y ampliamente empleada. Y por último, también me ha parecido importante haber podido familiarizarme con algoritmos de detección de cambios a los cuales encuentro de gran utilidad.

En el aspecto personal, durante la realización de este proyecto he mantenido unos niveles altos de interés y motivación que me han permitido disfrutar del trayecto. Considero que todo lo aprendido es de gran utilidad de cara a mi futuro. Gracias a Amelia Simó por el apoyo recibido en este paso final, estoy muy agradecido por la gran atención y preocupación que ha mostrado por hacer de este trabajo una enriquecedora y satisfactoria experiencia.

Bibliografía

[1] Washington State University. *Welcome to CASAS*. [Consulta: 19 de mayo de 2023]. URL: https://casas.wsu.edu/datasets.

[2] J L Pérez. *Introducción a la Estadística Circular*. Universidad de Extremadura, 2020.

[3] N Fisher. "Dispersion on a shpere". En: *Proceedings of the Royal Society, London* 125 (1953).

[4] K V Mardia y P E Jupp. *Directional Statistics*. Chichester: Wiley, 2000.

[5] Ph J Karoly et al. "Circadian and circaseptan rhythms in human epilepsy: a retrospective cohort study". En: *Lancet Neurology* 17 (2018), págs. 977-988.

[6] A A Schaaf et al. "Effect of geographical latitude and sun exposure on Rofous Hornero (Furnarius rufus) nest orientation". En: *Journal of Ornitology* 159 (2018), págs. 967-974.

[7] K Borycka e I Kasprzyk. "Hourly pattern of allergenic alder and birch pollen concetrations in the air: Spatial differentiation and the effect of meteorological conditions". En: *Atmosferic Environment* 182 (2018), págs. 179-192.

[8] X. Pennec. "Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements". En: *Journal of Mathematical Imaging and Vision* 25.1 (2006), págs. 127-154.

[9] David Michael Titterington, Adrian FM Smith y Udi E Makov. *Statistical analysis of finite mixture distributions*. Wiley, 1985.

[10] Geoffrey J McLachlan y Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

[11] Kurt Hornik y Bettina Grün. "movMF: An R package for fitting mixtures of von Mises-Fisher distributions." En: *Journal of Statistical Software* 58.10 (2014), pág. 4.

[12] Frank J Massey Jr. "The Kolmogorov-Smirnov test for goodness of fit". En: *Journal of the American statistical Association* 46.253 (1951), págs. 68-78.

[13] I Nikiforov y M Basseville. "Detection of abrupt changes: theory and application". En: *Englewood Cliffs, NJ: PTR Prentice Hall* (1993).

[14] Óscar Belmonte-Fernández et al. "Anomaly detection in activities of daily living with linear drift". En: *Cognitive Computation* 12.6 (2020), págs. 1233-1251.

[15] Jean Deshayes y Dominique Picard. "Off-line statistical analysis of change-point models using non parametric and likelihood methods". En: *Detection of abrupt changes in signals and dynamical systems*. Springer, 2005, págs. 103-168.

[16] Wikipedia. *Bayesian information criterion*. [Consulta: 19 de mayo de 2023]. URL: https://en.wikipedia.org/wiki/Bayesian_information_criterion.

[17] Ulric Lund, Claudio Agostinelli y Maintainer Claudio Agostinelli. “Package ‘circular’”. En: *Repository CRAN* 775 (2017).

[18] Diego Lacomba Fañanás. *Código implementado*. [Consulta: 5 de julio de 2023]. URL: https://github.com/diegolacomba/behaviour-model.

Anexo A

Código implementado

A.1. Introducción

En las próximas secciones se explican los aspectos que se consideran más importantes respecto al código implementado. El código completo se puede encontrar en [18].

A.2. Depurado

A continuación enumeramos por pasos el procedimiento realizado para obtener el depurado deseado de los datos. Cada paso se corresponde a un método cuya implementación se encuentra más adelante.

- **1) input_rooms()**: Asignamos a cada observación su habitación correspondiente en función del código del sensor o de la actividad realizada. La implementación se encuentra en la Figura A.1.
- **2) remove_ADLintermediate()**: Eliminamos las observaciones que se encuentran entre el inicio y final de una actividad. La implementación se encuentra en la Figura A.2.
- **3) remove_unpaired()**: Eliminamos las observaciones que no pertenecen a lo que se considera como una estancia (mínimo dos observaciones seguidas en la misma habitación). La implementación se encuentra en la Figura A.3.
- **4) remove_intermediate()**: De forma similar al paso 2, eliminamos las observaciones entre el inicio y fin de una estancia. La implementación se encuentra en la Figura A.4. En este instante tenemos un *dataset* como el que vemos en la Figura A.1.
- **5) remove_short()**: Eliminamos las estancias que duran menos de 2 minutos. La implementación se encuentra en la Figura A.5.
- **6) genera_transiciones_{habitación}()**: Por último, para cada habitación obtenemos los instantes en los que ocurre cada una de las 4 posibles transiciones por separado,

```
# Asociar habitaciones a sensores y actividades
function input_rooms()
    # Añadimos nueva columna al dataframe
    data[:,:habitacion] = missings(String15, nrow(data))
    # Asociamos sensor a habitacion
    data[in(["M028"]).(data.ID),:habitacion] .= "dormitorio"
    data[in(["M021"]).(data.ID),:habitacion] .= "dormitorio"
    data[in(["M020"]).(data.ID),:habitacion] .= "dormitorio"
    data[in(["M019"]).(data.ID),:habitacion] .= "dormitorio"
    data[in(["M025"]).(data.ID),:habitacion] .= "baño"
    # [...]
    data[in(["M005"]).(data.ID),:habitacion] .= "salon"
    data[in(["M006"]).(data.ID),:habitacion] .= "salon"
    data[in(["M001"]).(data.ID),:habitacion] .= "salon"
    data[in(["M002"]).(data.ID),:habitacion] .= "salon"

    # Sustituimos "missing" por "" (necesario para la continuacion)
    data.description = replace(data.description, missing => "")
    data.activity = replace(data.activity, missing => "")

    # Asociamos actividad a habitacion
    data[in(["Bed_to_Toilet"]).(data.description),:habitacion] .= "baño"
    data[in(["Desk_Activity"]).(data.description),:habitacion] .= "television"
    data[in(["Dining_Rm_Activity"]).(data.description),:habitacion] .= "salon"
    data[in(["Eve_Meds"]).(data.description),:habitacion] .= "cocina"
    data[in(["Guest_Bathroom"]).(data.description),:habitacion] .= "baño"
    data[in(["Kitchen_Activity"]).(data.description),:habitacion] .= "cocina"
    data[in(["Leave_Home"]).(data.description),:habitacion] .= "salon"
    data[in(["Master_Bathroom"]).(data.description),:habitacion] .= "baño"
    data[in(["Watch_TV"]).(data.description),:habitacion] .= "television"
    data[in(["Sleep"]).(data.description),:habitacion] .= "dormitorio"
    data[in(["Read"]).(data.description),:habitacion] .= "salon"
    data[in(["Morning_Meds"]).(data.description),:habitacion] .= "cocina"
    data[in(["Master_Bedroom_Activity"]).(data.description),:habitacion] .= "dormitorio"

    # Eliminamos las observaciones sin asignación de habitación:
    dropmissing!(data,:habitacion,disallowmissing=true)
    filter!(row -> !(row.description == "" && row.OnOff == "OFF"), data)
end
```

Figura A.1: Método correspondiente al paso 1 del depurado de datos.

transformamos del formato hora:minutos a minutos, de minutos a radianes, de radianes a coordenadas en el circulo unitario y los almacenamos en un fichero. Es importante mencionar que consideramos una transición entre habitaciones cuando la diferencia temporal entre las observaciones es de menos de 10 minutos. La implementación para el caso de la habitación “dormitorio” se encuentra en la Figura A.6.

Como se ha mencionado en la sección 3.2.1, el resultado es un fichero para cada transición, haciendo un total de 20 ficheros.

```
# Asociar totalmente una actividad a una habitación
# Eliminar observaciones entre begin y end
function remove_ADLintermediate()
    activity = false

    # Marcamos la observaciones a eliminar
    for i in 1:nrow(data)
        if data[i,:activity] == "begin"
            activity = true
        elseif data[i,:activity] == "end"
            activity = false
        elseif activity
                data[i,:habitacion] = "remove"
        end
    end


    # Eliminar observaciones por condición:
    filter!(row -> !(row.habitacion == "remove"), data)
end
```

Figura A.2: Método correspondiente al paso 2 del depurado de datos.

```
# Eliminar observaciones individuales
# Es decir, si la habitacion es distinta de la observación anterior y siguiente
function remove_unpaired()
    # Primera observacion:
    if data[1,:habitacion] != data[2,:habitacion]
        data[1,:description] = "remove"
    end

    # Resto:
    prev = data[1,:habitacion]
    for i in 2:nrow(data)-1

        current = data[i,:habitacion]
        sig = data[i+1,:habitacion]

        # Si se diferencia de la anterior Y siguiente
        if (current != prev && current != sig)
            # Eliminamos intermedio
            data[i,:description] = "remove"
        end

        prev = current

    end

    # Eliminar observaciones por condición:
    filter!(row -> !(row.description == "remove"), data)
end
```

Figura A.3: Método correspondiente al paso 3 del depurado de datos.

```
# Eliminar observaciones intermedias (minimo tres seguidas)
# Resultado: parejas inicio-fin estancia
function remove_intermediate()
    prev = data[1,:habitacion]

    for i in 2:nrow(data)-1

        current = data[i,:habitacion]
        sig = data[i+1,:habitacion]

        # Si hay tres seguidas
        if (prev == current && prev == sig)
            # Eliminamos intermedio
            data[i,:habitacion] = "remove"
        else
            prev = current
        end

    end

    # Eliminar observaciones por condición:
    filter!(row -> !(row.habitacion == "remove"), data)
end
```

Figura A.4: Método correspondiente al paso 4 del depurado de datos.

```
# Eliminar estancias de menos de 2 minutos
function remove_short()
    # Eliminamos estancias de menos de 2 minutos
    for i in 1:2:nrow(data)
        minutes = duration(data[i,:HOUR],data[i+1,:HOUR])

        if minutes < 2
            data[i,:habitacion] = "remove"
            data[i+1,:habitacion] = "remove"
        end
    end

    # Eliminar observaciones por condición:
    filter!(row -> !(row.habitacion == "remove"), data)
end
```

Figura A.5: Método correspondiente al paso 5 del depurado de datos.

```
# ---------------------------- DORMITORIO -----------------------------------
function genera_transiciones_dormitorio()

    # bano
    dormitorio_bano_coords = rad2unitcircle(min2rad(hour2min(transiciones("dormitorio","baño"))))
    dormitorio_bano.x = dormitorio_bano_coords[1]
    dormitorio_bano.y = dormitorio_bano_coords[2]
    CSV.write("Transiciones\\dormitorio_bano.csv", dormitorio_bano)

    # COCINA
    dormitorio_cocina_coords = rad2unitcircle(min2rad(hour2min(transiciones("dormitorio","cocina"))))
    dormitorio_cocina.x = dormitorio_cocina_coords[1]
    dormitorio_cocina.y = dormitorio_cocina_coords[2]
    CSV.write("Transiciones\\dormitorio_cocina.csv", dormitorio_cocina)

    # TELEVISION
    dormitorio_tv_coords = rad2unitcircle(min2rad(hour2min(transiciones("dormitorio","television"))))
    dormitorio_tv.x = dormitorio_tv_coords[1]
    dormitorio_tv.y = dormitorio_tv_coords[2]
    CSV.write("Transiciones\\dormitorio_tv.csv", dormitorio_tv)

    # SALON
    dormitorio_salon_coords = rad2unitcircle(min2rad(hour2min(transiciones("dormitorio","salon"))))
    dormitorio_salon.x = dormitorio_salon_coords[1]
    dormitorio_salon.y = dormitorio_salon_coords[2]
    CSV.write("Transiciones\\dormitorio_salon.csv", dormitorio_salon)

end
```

Figura A.6: Método correspondiente al paso 6 del depurado de datos.

Fecha	Hora	Habitación
2009-10-16	00:01:08.000081	dormitorio
2009-10-16	03:55:48.000049	dormitorio
2009-10-16	03:55:53.000080	baño
2009-10-16	03:58:28.000002	baño
2009-10-16	03:58:42.000014	dormitorio
...	...	...
2009-10-16	10:40:11.000028	cocina
2009-10-16	10:50:21.000006	cocina
...	...	...

Cuadro A.1: Datos del dataset Milán depurados.

A.3. Estimación de parámetros y test de bondad de ajuste

En esta sección se explica brevemente la implementación correspondiente a la estimación de parámetros y al test de bondad de ajuste de Kolmogorov-Smirnov. El procedimiento es el siguiente:

- **Paso 1:** Estimar los parámetros para mixturas de 2 a 6 componentes y escoger la mixtura con mejor BIC.
- **Paso 2:** Calcular la frecuencia acumulada F.

```
# Obtención de EMV calculable y KSTest
function testParams(coords,radians)
    # Mixtura con mejor BIC
    @rput coords
    R"""
    library(movMF)
    vMs = lapply(2:6, function(K) movMF(as.matrix(coords), k=K, control=list(nruns = 30)))
    options = sapply(vMs, BIC)
    options_sorted = sort(options,index.return=TRUE)
    id_sorted = options_sorted$ix
    vM = movMF(as.matrix(coords), k=id_sorted[1]+1, control=list(nruns = 30))
    """
    @rget vM

    F = getF(radians)
    mixtura = getParams(vM)

    # Comprobar que sea posible el calculo de F0 (kappa<=700)
    i=1
    iter=1
    while (i <= size(mixtura)[1])
        if (mixtura[i,4] > 700)
            # Ninguna mixtura válida
            if (iter==5)
                print("Limit exceeded")
                return changeParams(0)
            end
            # Cambiamos a la siguiente mixtura con mejor BIC y reiniciamos
            mixtura = changeParams(iter)
            i=1
            iter+=1
        else
            i+=1
        end

    end
    # Actual mejor mixtura calculable
    save = mixtura
```

Figura A.7: Implementación en Julia de la estimación de parámetros y del test Kolmogorov-Smirnov (Parte 1).

- **Paso 3:** Comprobar que ninguna componente de la mixtura tiene un parámetro de concentración mayor de 700 para poder calcular la frecuencia acumulada bajo los parámetros estimados (F0). En caso de no cumplir con el requisito lo intentamos con la siguiente mixtura con mejor BIC. La implementación hasta este instante se encuentra en la Figura A.7. A partir de este instante la implementación se encuentra en la Figura A.8.

- **Paso 4:** Calcular F0.

- **Paso 5:** Ejecutar el test de bondad de ajuste. En caso de no superarlo, probar con la siguiente mixtura con mejor BIC. Volver a comprobar que sea posible el cálculo de F0. En caso de superarlo, almacenar los parámetros de la mixtura.

```
    # Una vez encontrada una mixtura adecuada,
    # Obtenemos frecuencias acumuladas bajo parámetros estimados
    F0 = getF0(radians, mixtura)

    # Test de bondad de ajuste Kolmogorov-Smirnov
    while (!runKS(F,F0))
        if (iter==5)
            print("Limit exceeded")
            return save
        end
        mixtura=changeParams(iter)
        iter+=1

        # Buscamos un kappa < 700
        i=1
        while (i <= size(mixtura)[1])
            if (mixtura[i,4] > 700)
                # Ninguna mixtura válida
                if (iter==5)
                    print("Limit exceeded")
                    return save
                end
                # Cambiamos a la siguiente mixtura con mejor BIC y reiniciamos
                mixtura = changeParams(iter)
                i=1
                iter+=1
            else
                i+=1
            end
        end
        F0 = getF0(radians, mixtura)
    end

    return mixtura

end
```

Figura A.8: Implementación en Julia de la estimación de parámetros y del test Kolmogorov-Smirnov (Parte 2).

A.4. Detección de anomalías con CUSUM

En esta sección se explica brevemente el procedimiento seguido para simular y detectar anomalías. El procedimiento se puede dividir en los siguientes pasos:

- **Paso 1:** Simular una secuencia introduciendo un cambio en los parámetros a partir de cierta observación.
- **Paso 2:** Cálculo de $\hat{\theta}_1^{(j)} = EMV(\theta_1)$ para $j = 1, ..., n$. Como se menciona en la sección 3.2.2, para estimar de forma efectiva los parámetros necesitamos un tamaño muestral mayor de 40. Por ello, si la muestra tiene un tamaño N, $n = N - 40$.
- **Paso 3:** Cálculo de S_j^n para $j = 1, ..., n$. En la Figura A.9 podemos ver la implementación de este paso.
- **Paso 4:** Obtención del estadístico de contraste Λ^n y cálculo del umbral h. En la Figura A.10 podemos ver la implementación de este paso.

```
# Calculo del estadístico de la razón de verosimilitud generalizado
# para cada contraste
CumSums = []
for j in 1:n
    muestra = muestra_coords[j:N,:]

    theta1 = Matrix(EMVs[j][:,5:6])
    alpha1 = EMVs[j][:,:alpha]

    @rput muestra
    @rput theta1
    @rput alpha1
    R"""
    p0 = dmovMF(muestra, theta, alpha)
    p1 = dmovMF(muestra, theta1, alpha1)
    """
    @rget p0
    @rget p1

    cumsum = 2*sum(log.(p1)-log.(p0))

    push!(CumSums,cumsum)
end
```

Figura A.9: Cálculo de los estadísticos de la razón de verosimilitud generalizados para cada contraste.

```
# Estadístico de contraste (supremo)
τ = sortperm(CumSums,rev=true)[1]
A = CumSums[τ]

# Cálculo del umbral de rechazo
α = 0.05
prob = α/n
p = componentes+(componentes-1)
@rput prob
@rput p
R"""
umbral = qchisq(p=prob, df=p, lower.tail=FALSE)
"""
@rget umbral
```

Figura A.10: Obtención del estadístico de contraste y cálculo del umbral.