BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

APLICACIÓN DEL MÉTODO BOOTSTRAP PARA LA OBTENCIÓN DEL ERROR ESTÁNDAR, EN UNA MEZCLA DE NORMALES OBTENIDA MEDIANTE MAXIMIZACIÓN DE LA ESPERANZA (EM)

TESIS PARA OBTENER EL GRADO DE:

LICENCIADO EN INGENIERIA DE CIENCIAS DE LA COMPUTACIÓN

PRESENTA: RUBEN ILHAN MARTÍNEZ MANCILLA

ASESOR 1: DR. GERARDO MARTÍNEZ GUZMÁN

ASESOR 2: MC. MARIANO LARIOS GÓMEZ

MARZO 2021

Dedicatoria

Para mi equipo, los que tienen mi sangre y por la que daría cada gota.

Agradecimientos

Agradezco a mi director de tesis el Dr. Gerardo Martinez Guzmán que me dio las herramientas necesarias para desarrollar esta tesis y que me motivo para entregar un trabajo serio, sin el no hubiera sido posible.

También agradezco a mi asesor MC. Mariano Larios Gómez por el apoyo brindado para pulir este trabajo.

Gracias a la BUAP que es la universidad del pueblo y a todos mis maestros que se esforzaron día con día para transmitir el conocimiento.

Gracias a mi patria porque es lo que me motiva a ser un profesional.

Gracias a mi mamá que fue la inversora principal de mi carrera, gracias porque nunca dejaste de creer en mí y por presentarme a tu Dios que me dio vida cada día de los 7 años que me tarde en terminar esto.

Agradecimientos especiales para Adriana Luna por que tuvo el duro trabajo de ser mi esposa y me esta enseñando a leer, a Silvio porque me motivo a perder el miedo y a ser ejemplo y a Sophie porque la mejor medicina para la ansiedad es la ternura de su cara.

A mi hermano Jonathan por todas las cenas de domingo.

A mi tía Miriam que me dio rutinas de ejercicio para generar serotonina sin necesidad de medicinas.

A mi tía Vero que fue niñera de mis hijos muchas veces y a su familia que apoyó, gracias Loreli, Felipe y Valentina.

A mi prima Diana que fue la única que me espero para terminar la carrera.

A mi perro Chase que me ayudo a sacudirme el estrés cuando jugaba con el frisbee y a Moka que no he podido darle todo lo que merece, pero me quiere.

A la familia de mi esposa, mi suegra Lucia, su hermana Angelica y la tía Candy que muchas veces me sacaron de apuros cuando no podía ir a la universidad.

A Quecha, que fue la primera niñera.

ÍNDICE

APLICACIÓN DEL MÉTODO BOOTSTRAP PARA LA OBTENCIÓN DEL ERROR ESTÁNDAR, EN UNA MEZCLA DE NORMALES OBTENIDA MEDIANTE MAXIMIZACIÓN DE LA ESPERANZA (EM)

MARZO 2020

I. MARCO TEÓRICO

Resumen

El cáncer de mama es un tumor sólido y según GLOBOCAN es la cuarta causa de muerte por cáncer en general y es el tipo de cáncer más frecuente en mujeres y el más letal. El cáncer es un proceso celular transformativo y de esta manera, las células cancerosas pueden diferir morfológicamente de las células que les dieron origen [11].

El diagnóstico de células cancerosas a partir de biopsias se basa principalmente en el análisis de cambios morfológicos de la estructura nuclear como el aumento del tamaño nuclear [12], que ocurre posiblemente debido a la desregulación del ciclo celular, así como del crecimiento celular [13]. El incremento del tamaño nuclear se observa en biopsias de pacientes con diagnóstico benigno y maligno.

En este trabajo se hace un análisis de la variable (radius mean) relacionada con el incremento del tamaño nuclear en pacientes con diagnóstico benigno y maligno. En este análisis se prueba, mediante el algoritmo de aprendizaje no supervisado llamado maximización de la esperanza (EM) que dicha variable tiene un comportamiento tipo mezcla de normales con dos componentes. Tal algoritmo es capaz de discriminar los datos en dos grupos (malignos y benignos). Dicho modelo proyecta una clasificación con un porcentaje alto de coincidencia con los datos observados.

Sin embargo, el cálculo del error estándar mediante las técnicas tradicionales es demasiado complicado, por lo que en este trabajo se implementa el método de bootstrap paramétrico para el cálculo del error estándar que usa el remuestreo intensivo.

Introducción

El bootstrap fue introducido en 1979 por Bradley Efron. Los métodos bootstrap son una clase de métodos Monte Carlo no paramétricos que pretenden estimar propiedades de un estimador tales como su sesgo o su error estándar o estimar la distribución de una población mediante remuestreo. Los métodos de remuestreo tratan una muestra observada como una población finita, y a partir de ella, generan muestras aleatorias para estimar características poblacionales y hacer inferencia sobre la población muestreada. A menudo, estos métodos se usan cuando no se conoce la distribución de la población objetivo, de modo que la muestra es la única información disponible.

El término bootstrap puede referirse a bootstrap no paramétrico o bootstrap paramétrico. Los métodos de Monte Carlo que implican el muestreo a partir de una distribución que no se especifica se conoce como bootstrap no paramétrico. Si la distribución de probabilidad está completamente especificada, el muestreo se conoce como bootstrap paramétrico. En esta tesis se trabajará con una distribución completamente determinada, una mezcla de normales, donde interesa calcular el error estándar de los estimadores de parámetros.

De acuerdo con lo anterior, la distribución de la población finita representada por la muestra puede ser vista como una pseudopoblación con características similares a las de la verdadera población, por lo tanto, generando repetidamente muestras aleatorias de esta pseudopoblación, se puede estimar la distribución muestral de un estadístico. Cabe señalar que las estimaciones bootstrap de una distribución de muestreo son análogas a la idea de estimación de la densidad. El histograma de una muestra proporciona una estimación de la forma de la función de densidad. El histograma no es la densidad, pero desde el punto de vista no paramétrico puede ser visto como una estimación razonable de la misma.

Planteamiento del problema

El enfoque tradicional de la inferencia estadística se basa en modelos idealizados y suposiciones. A menudo, las expresiones para medidas de precisión, como el error estándar, se basan en la teoría asintótica y no están disponibles para muestras pequeñas, ver [4]. Una alternativa moderna a la aproximación tradicional es el método bootstrap, presentado por Efron (1979), ver [2] y [3]. El bootstrap es un método de remuestreo intensivo, que es ampliamente aplicable y permite el tratamiento de más modelos realistas.

En este trabajo se desarrollará un problema donde a partir de una distribución de probabilidad completamente especificada a saber, una mezcla de distribuciones normales, se encuentra una estimación de sus parámetros desconocidos, mediante el método estadístico llamado maximización de la esperanza (EM), y posteriormente realizamos el cálculo del error estándar de estos estimadores mediante la técnica bootstrap.

Las distribuciones de mixturas finitas se han empleado para la modelización de datos heterogéneos, puesto que en muchos casos no es suficiente explicar la distribución de unos datos mediante una única distribución estadística, es necesaria la utilización de una combinación de distribuciones. Estas combinaciones comúnmente son descritas mediante los modelos de mixturas, ver [8] y [9], los cuales se definen por los parámetros de cada componente y las proporciones en las que cada una de ellas contribuye a la distribución general.

Este concepto conduce al agrupamiento del conjunto de observaciones en grupos con algunas características comunes. Las distribuciones mixtas pueden ser estimadas mediante muchas técnicas, tales como métodos gráficos, el método de los momentos, de máxima verosimilitud, aproximaciones bayesianas y análisis de componentes principales por mencionar algunas.

El algoritmo EM (Maximización de la Esperanza), ver [7], es una herramienta habitual iterativa para la estimación de máxima verosimilitud de las distribuciones mixtas. La idea es introducir una variable indicadora multinomial que identifica la pertenencia a un grupo de cada observación del conjunto de datos. El algoritmo EM fue expuesto por Arthur Dempster, Nan Laird y Donald Rubin de la Royal Statistical Society en una publicación de 1977 [1].

Análisis de los datos

Los datos que se tienen para el estudio es una muestra de 569 mujeres, cada una de ellas con el ID que identifica a la persona, el diagnostico si el tumor es benigno o maligno y el valor de la variable radius_mean (mean of distances from center to points on the perimeter).

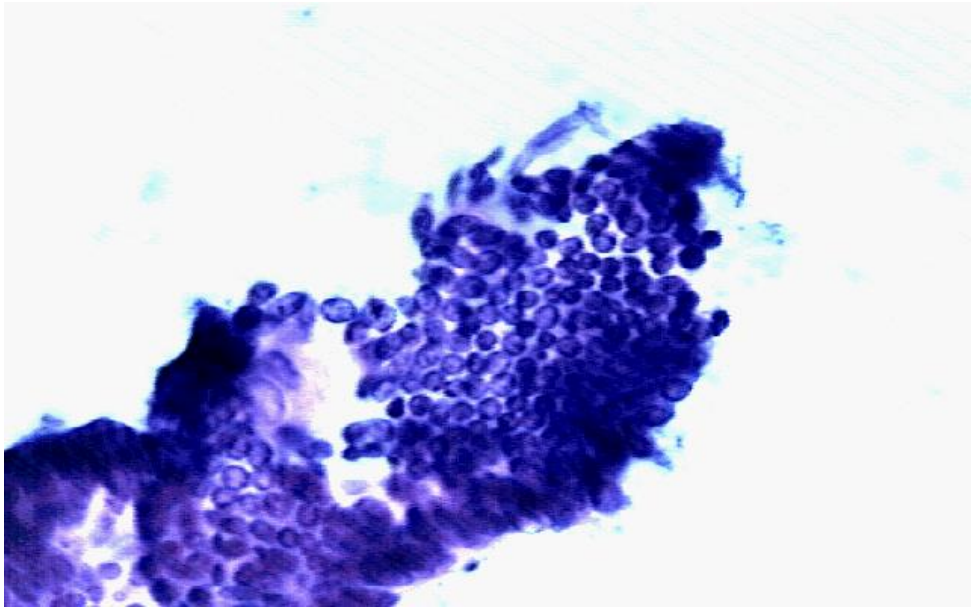


Imagen 1. 92_751 Radius mean: 7.76

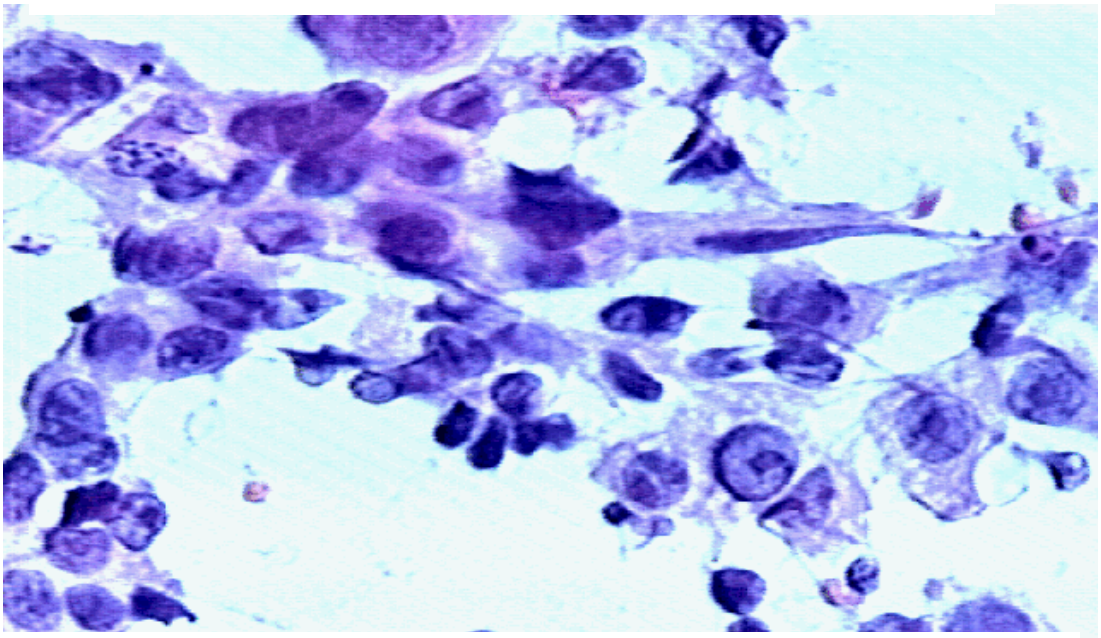


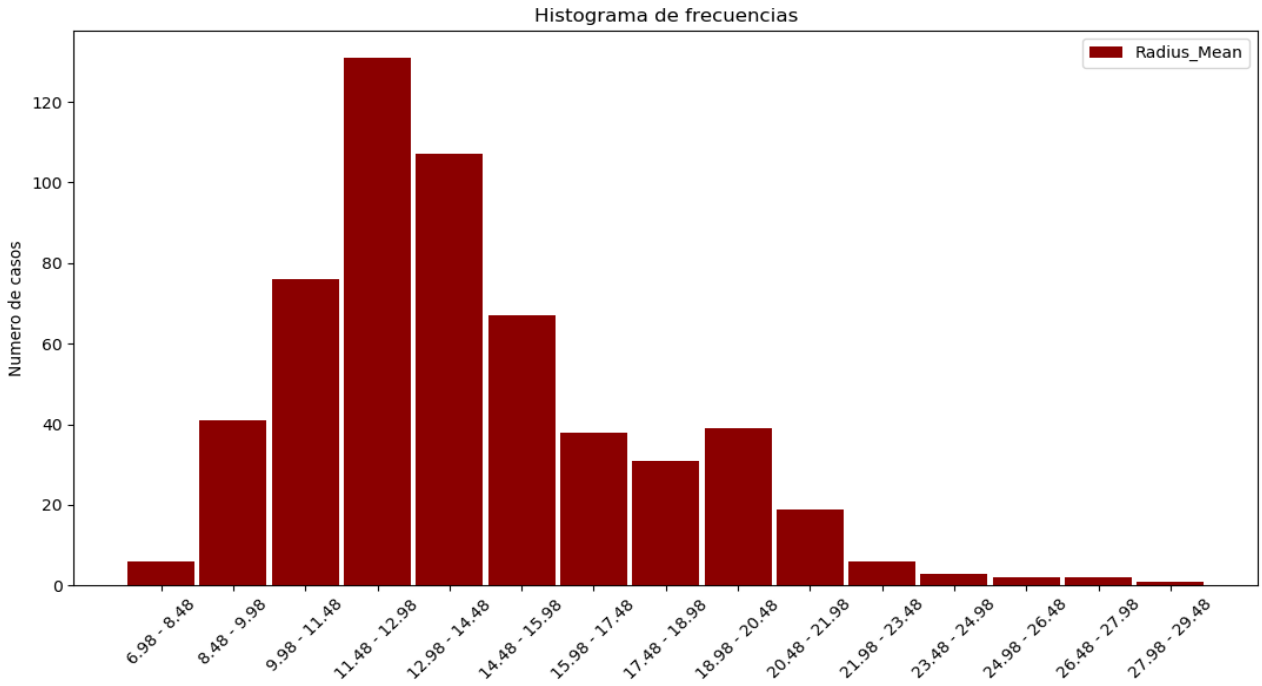
Imagen 2. 92_7241 Radius mean: 20.6

Del total de la muestra existen 357 casos que resultaron benignos y 212 de los casos maligno, ver [5]. Se muestran algunos datos de la base en la Tabla 1.

ID	Diagnóstico Benigno (B), Maligno (M)	radius_mean
8612399	**M**	**18.46**
86135501	**M**	**14.48**
86135502	**M**	**19.02**
861597	**B**	**12.36**
861598	**B**	**14.64**
861648	**B**	**14.62**
861799	**M**	**15.37**
861853	**B**	**13.27**
862009	**B**	**13.45**
862028	**M**	**15.06**
86208	**M**	**20.26**
86211	**B**	**12.18**
862261	**B**	**9.787**
862485	**B**	**11.6**
862548	**M**	**14.42**
862717	**M**	**13.61**
862722	**B**	**6.981**
862965	**B**	**12.18**
862980	**B**	**9.876**
862989	**B**	**10.49**

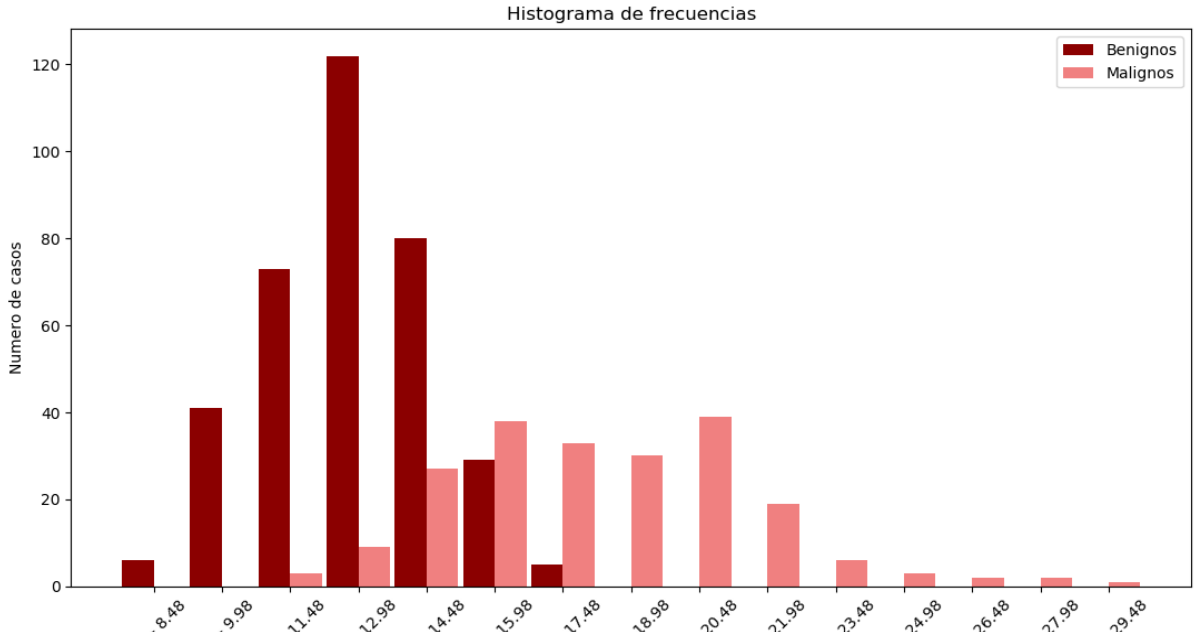
Tabla 1. En esta tabla se muestran algunos datos de la base de datos original.

Un Histograma de frecuencias de la variable radius_mean se presenta en la Gráfica 1.



Gráfica 1.Histograma de frecuencias de la variable radius mean

Si consideramos la variable radius_mean y tomamos en cuenta que, de las 569 observaciones totales, 357 observaciones indican la ausencia de células cancerosas, mientras que 212 observaciones muestran la presencia de células cancerosas, el histograma de frecuencias de la variable con el diagnóstico se presenta en la Gráfica 2.



Gráfica 2. Histograma con diagnóstico

El análisis en este trabajo tiene como objetivo determinar en qué medida la variable radius_mean es útil para predecir si el cáncer de seno es benigno o maligno. Se desarrollará el modelo de la mezcla de normales y se aplicará a la variable radius_mean para establecer si este modelo puede contribuir para el logro de esta investigación, ver [10].

Modelo de mixturas con g componentes

Las distribuciones mixtas son utilizadas para la modelización de datos que en muchas situaciones experimentales pueden interpretarse como procedentes de dos o más subpoblaciones. La obtención de estas componentes conduce a la estimación de los parámetros de la mixtura, ver [9] y [6].

Algoritmo EM

En el desarrollo del algoritmo EM, proporcionamos una formulación paramétrica para la representación del modelo. En lo sucesivo, mediante $Y = (Y_1, Y_2, \ldots, Y_n)$, se denotará a una muestra aleatoria de tamaño n, donde Y_i es un vector aleatorio q −dimensional con función de densidad de probabilidad $f(y_i)$ donde $y_i \in R^q$. Así $y = (y_1, y_2, \ldots, y_n)$ representa a una muestra observada o realización de Y, donde y_i constituye un valor observado del vector aleatorio Y_i.

Definición 1. Si la función de densidad de una variable aleatoria Y_i es de la forma

$$f(y_i|\boldsymbol{\psi}) = \sum_{k=1}^{g} \pi_k f_k(y_i|\boldsymbol{\theta_k}) \qquad y_i \in R^q$$

se dice que posee una distribución de mixtura finita con *g* componentes, con un vector de parámetros

$$\psi = (\pi_1, \ldots, \pi_g, \boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_g}).$$

Aquí, $f_k(y_i|\boldsymbol{\theta_k}),\ k = 1,2,\ldots,g$, denotan las densidades de las componentes de la mixtura con parámetros $\boldsymbol{\theta_k}$ y parámetros de peso $\pi_1, \ldots, \pi_g$. Se asume también que las funciones $f_k(y_i|\boldsymbol{\theta_k})$ pueden pertenecer a diferentes familias paramétricas.

Para que la mixtura sea una función de densidad los pesos deben cumplir las condiciones

$$0 \leq \pi_k \leq 1 \quad k = 1, \ldots, g \quad y \quad \sum_{k=1}^{g} \pi_k = 1$$

Note que en la condición anterior uno de los pesos resulta redundante (uno de ellos se expresa en términos de los demás).

Para mixturas cuyas componentes pertenecen a otras familias de densidades, puede consultarse Moharir (1992) para mixturas de distribuciones Poisson; Falls (1970), para mixturas Weibull; Blischke (1962), para mixturas binomiales.

Definición 2. Sea $y = (y_1, y_2, \ldots, y_n)$ observaciones independientes de una variable aleatoria, cuya función de densidad $f(y|\boldsymbol{\psi})$ es una mixtura, entonces la funcion

$$L(\boldsymbol{\psi}|y) = \prod_{i=1}^{n} f(y_i|\boldsymbol{\psi}) = \prod_{i=1}^{n} \sum_{k=1}^{g} \pi_k f_k(y_i|\boldsymbol{\theta_k})$$

recibe el nombre de función de verosimilitud de la mixtura.

Tomando logaritmo natural en $L(\boldsymbol{\psi}|y)$ obtenemos su función log-verosimilitud

$$l(\boldsymbol{\psi}|y) = \log L(\boldsymbol{\psi}|y) = \log \prod_{i=1}^{n} \left\{ \sum_{k=1}^{g} \pi_k f_k(y_i|\boldsymbol{\theta_k}) \right\}$$
$$= \sum_{i=1}^{n} \log \left\{ \sum_{k=1}^{g} \pi_k f_k(y_i|\boldsymbol{\theta_k}) \right\}$$

Para calcular el estimador de máxima verosimilitud $\widehat{\boldsymbol{\psi}}$ es común utilizar el logaritmo de la función de verosimilitud, pues recordemos que la función y el logaritmo de la función bajo ciertas condiciones de regularidad toman en el mismo punto su máximo. Por lo tanto, debemos resolver la ecuación de verosimilitud,

$$\frac{\partial}{\partial \boldsymbol{\psi}} \sum_{i=1}^{n} \log \left\{ \sum_{k=1}^{g} \pi_k f_k(y_i|\boldsymbol{\theta_k}) \right\} = 0$$

Debido a la presencia del logaritmo de una suma, es difícil la resolución de la ecuación, por tal motivo se requiere otro tipo de procedimiento.

Sea $y = (y_1, y_2, \ldots, y_n)$ una muestra observada de tamaño n, a la que denominaremos vector de datos incompletos, correspondientes a una realización de Y, con función de densidad $f(y|\boldsymbol{\psi})$, donde $\boldsymbol{\psi}$ es el vector de parámetros a estimar. Ahora considere la variable $Z = (Z_1, Z_2, \ldots, Z_n)$ que denominaremos latente, que representa a los datos no observados y cuya realización es $z = (z_1, z_2, \ldots, z_n)$. Entonces el vector aleatorio $X = (Y, Z)$ recibe el nombre de vector de datos completos y su

realización es $x_1 = (y_1, z_1), x_2 = (y_2, z_2), \dots, x_n = (y_n, z_n)$ de tal forma que a cada realización y_i le corresponde siempre una z_i.

En este contexto podemos suponer que Z_i representa una variable indicadora binaria g −dimensional cuyo elemento j −esimo Z_{ij} indica la pertenencia de la observación y_i a la componente j −esima de la mixtura donde $i = 1,2 \dots, n$ y $j = 1,2, \dots, g$. Así, podemos definir Z_{ij} como,

$$Z_{ij} = z_{ij} = \begin{cases} 1 & Si\ \ y_i - proviene\ de\ la\ componente\ j - esima. \\ 0 & en\ otro\ caso. \end{cases}$$

Podemos representar al vector de datos incompletos y como un vector transpuesto

$$y' = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

y representar en una matriz a los datos completos

$$x' = (y', z') = \begin{pmatrix} y_1 & z_1 \\ y_2 & z_2 \\ \vdots & \vdots \\ y_n & z_n \end{pmatrix} = \begin{pmatrix} y_1 & z_{11} & \dots & z_{1g} \\ y_2 & z_{21} & \dots & z_{2g} \\ \vdots & \vdots & \vdots & \vdots \\ y_n & z_{n1} & \dots & z_{ng} \end{pmatrix}$$

Cada realización z_i toma $(g - 1)$ valores iguales a cero y un único valor igual a uno.

Dada la naturaleza categórica de la variable Z_i al indicar la pertenencia de los puntos muestrales a una componente u otra de la mixtura, se puede interpretar que los pesos π_k es la probabilidad a priori de que la observación y_i pertenezca a la población k, lo que hace suponer que Z_i sigue una distribución multinomial de una sola realización sobre g categorías con probabilidades $\pi = (\pi_1, \pi_2, \dots, \pi_g)$, es decir,

$$P(Z_i = z_i) = \binom{1}{z_{i1}, z_{i2}, \dots, z_{ig}} {\pi_1}^{z_{i1}} {\pi_2}^{z_{i2}} \dots {\pi_g}^{z_{ig}} = \prod_{k=1}^{g} {\pi_k}^{z_{ik}}$$

Donde

$$\sum_{k=1}^{g} z_{ik} = 1 \qquad \sum_{k=1}^{g}\sum_{i=1}^{n} z_{ik} = n$$

Función log verosimilitud de los datos completos

Usando el hecho de que,

$$f_k(y_i|z_{ik} = 1) = f_k(y_i|\boldsymbol{\theta_k}).$$

Y desarrollando la distribución de Y_i con todos los estados posibles de z_k tenemos

$$f_k(y_i, z_i) = f_k\big(Y_i = y_i, Z_{i1} = z_{i1}, \dots, Z_{ig} = z_{ig}\big) =$$

$$f_k(Y_i = y_i\big|Z_{i1} = z_{i1}, \dots, Z_{ig} = z_{ig})P(Z_{i1} = z_{i1}, \dots, Z_{ig} = z_{ig}$$

$$= \left\{\prod_{k=1}^{g} f_k(y_i|\boldsymbol{\theta_k})^{z_{ik}}\right\}\left\{\prod_{k=1}^{g} \pi_k{}^{z_{ik}}\right\} = \prod_{k=1}^{g}[\pi_k f_k(y_i|\boldsymbol{\theta_k})]^{z_{ik}}$$

De aquí la función de verosimilitud conjunta para todos los valores observados y y todos los valores no observados z es

$$\prod_{i=1}^{n}\prod_{k=1}^{g}[\pi_k f_k(y_i|\boldsymbol{\theta_k})]^{z_{ik}}$$

Aplicando la función logaritmo tenemos

$$l(\boldsymbol{\psi}|y, z) = \log L(\boldsymbol{\psi}|y, z) = \sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik} log[\pi_k f_k(y_i|\boldsymbol{\theta_k})]$$

$$\sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik}[log\pi_k + log f_k(y_i|\boldsymbol{\theta_k})]$$

$$\sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik} log\pi_k + \sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik} log f_k(y_i|\boldsymbol{\theta_k})$$

Aquí

$$\boldsymbol{\psi} = \big(\pi_1, \dots, \pi_g, \boldsymbol{\theta_1}, \dots, \boldsymbol{\theta_g}\big).$$

Para el caso de mixtura Gaussiana con parámetros $\boldsymbol{\theta_k} = (\mu_k, {\sigma_k}^2)$, tenemos

$$l(\boldsymbol{\psi}|y,z) = \sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik} log\pi_k + \sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik} log\varphi(y_i|\mu_k, {\sigma_k}^2) \; =$$

$$\sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik} log\pi_k - \frac{n}{2} log 2\pi - \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik}\left[log{\sigma_k}^2 + \frac{(y_i - \mu_k)^2}{{\sigma_k}^2}\right]$$

Agrupamiento del modelo de mixturas Gaussianas

Una vez definidas las variables Z_{ik} puede introducirse ahora el concepto de agrupamiento sobre los datos observados. Uno de los propósitos de los modelos mixtos es el de proporcionar una partición de los datos en g grupos, siendo g un numero previamente establecido. Como el k-esimo peso de la mixtura π_k lo interpretamos como la probabilidad a priori de que una observación muestral pertenezca a la población k, entonces,

$$P(z_{ik} = 1) = \pi_k \qquad k = 1,2,..,g.$$

Esta suposición se hace porque podemos obtener información acerca de los datos ausentes mediante la observación de las elecciones que se realizaron. Por ejemplo, si los ingresos de una persona no están disponibles, pero se observa que la persona ha comprado un Mercedes, se puede inferir que es probable que los ingresos de dicha persona estén por encima de la media.

Calculamos las probabilidades de los datos ausentes condicionada a las elecciones observadas en la muestra, mediante el teorema de Bayes,

$$P(z_{ik} = 1|Y_i = y_i) = \frac{P(z_{ik} = 1)P(Y_i = y_i|z_{ik} = 1)}{P(Y_i = y_i)}$$

$$= \frac{\pi_k f_k(y_i|\boldsymbol{\theta_k})}{\sum_{k=1}^{g}\pi_k f_k(y_i|\boldsymbol{\theta_k})}.$$

Si definimos $h(z|y,\boldsymbol{\psi})$ como la densidad de los datos ausentes condicionada a las elecciones observadas en la muestra. Entonces,

$$h(z|y,\boldsymbol{\psi}) = P(z_{ik} = 1|Y_i = y_i) = \frac{\pi_k f_k(y_i|\boldsymbol{\theta_k})}{\sum_{k=1}^{g}\pi_k f_k(y_i|\boldsymbol{\theta_k})}.$$

El procedimiento EM es iterativo y comienza con un valor inicial de los parámetros $\boldsymbol{\psi}^0$ y en cada iteración se van actualizando de la manera siguiente. Definimos una nueva función en $\boldsymbol{\psi}$ que se relaciona con la función de verosimilitud pero que utiliza la distribución condicionada $h(z|y,\boldsymbol{\psi})$, esta nueva función es

$$\mathcal{E}(\boldsymbol{\psi}|\boldsymbol{\psi^0}) = E\left[l(\boldsymbol{\psi}|y,z)\middle|Y=y,\boldsymbol{\psi^0}\right]$$

$$\mathcal{E}(\boldsymbol{\psi}|\boldsymbol{\psi^0}) = E\left[\sum_{i=1}^{n}\sum_{k=1}^{g} z_{ik} log[\pi_k f_k(y_i|\boldsymbol{\theta_k})]\middle|Y=y,\boldsymbol{\psi^0}\right]$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{g} E\left[z_{ik}\middle|Y_i=y_i,\boldsymbol{\psi^0}\right][log\pi_k + log f_k(y_i|\boldsymbol{\theta_k})]$$

Sin embargo,

$$E\left[z_{ik}\middle|Y_i=y_i,\boldsymbol{\psi^0}\right] = P(z_{ik}=1|Y_i=y_i,\boldsymbol{\psi}^0)$$

$$= \frac{f_k(Y_i=y_i|z_{ik}=1)P(z_{ik}=1)}{P(Y_i=y_i)}\Bigg|_{\boldsymbol{\psi^0}}$$

$$= \frac{\pi_k f_k(y_i|\boldsymbol{\theta_k})}{\sum_{k=1}^{g}\pi_k f_k(y_i|\boldsymbol{\theta_k})}\Bigg|_{\boldsymbol{\psi^0}} = \hat{\tau}_{ik}{}^{(0)}$$

Por lo tanto,

$$\mathcal{E}(\boldsymbol{\psi}|\boldsymbol{\psi^0}) = \sum_{i=1}^{n}\sum_{k=1}^{g} \hat{\tau}_{ik}{}^{(0)}[log\pi_k + log f_k(y_i|\boldsymbol{\theta_k})]$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{g} \hat{\tau}_{ik}{}^{(0)} log\pi_k + \sum_{i=1}^{n}\sum_{k=1}^{g} \hat{\tau}_{ik}{}^{(0)} log f_k(y_i|\boldsymbol{\theta_k}).$$

Después del cálculo anterior se realiza la maximización de la función $\mathcal{E}$ respecto de $\boldsymbol{\psi}$. Esta maximización se realiza en dos partes dado que π_k aparece únicamente en el primer sumando y $\boldsymbol{\theta_k}$ lo hace el segundo sumando.

Comenzamos con la maximización del primer sumando. Para este caso usamos los multiplicadores de Lagrange

$$\frac{\partial}{\partial \pi_k}\left(\sum_{i=1}^{n}\sum_{k=1}^{g}\hat{\tau}_{ik}{}^{(0)}\, log\pi_k + \lambda\left[\sum_{k=1}^{g}\pi_k - 1\right]\right) = 0$$

$$\sum_{i=1}^{n}\hat{\tau}_{ik}{}^{(0)}\frac{1}{\pi_k} + \lambda = 0$$

$$\sum_{i=1}^{n}\hat{\tau}_{ik}{}^{(0)} = -\lambda\pi_k$$

Tomando la suma sobre k en ambos lados de la última igualdad obtenemos,

$$n = \sum_{i=1}^{n}\sum_{k=1}^{g}\hat{\tau}_{ik}{}^{(0)} = \sum_{k=1}^{g} -\lambda\pi_k = -\lambda$$

Lo cual implica que

$$\hat{\pi}_k{}^{(1)} = \pi_k = \frac{1}{n}\sum_{i=1}^{n}\hat{\tau}_{ik}{}^{(0)}$$

Para la maximización del segundo sumando respecto de $\boldsymbol{\theta_k}$ depende de la función de densidad $f_k(y_i|\boldsymbol{\theta_k})$ que en nuestro caso son densidades Gaussianas, entonces

$$logf_k(y_i|\boldsymbol{\theta_k}) = log\varphi(y_i|\mu_k, \sigma_k{}^2) =$$

$$= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\frac{(y_i - \mu_k)^2}{\sigma_k{}^2}$$

$$= -\frac{1}{2}\log(2\pi) - log\sigma^2 - \frac{1}{2}\frac{(y_i - \mu_k)^2}{\sigma_k{}^2}$$

Empezamos derivando respecto de μ

$$\frac{\partial}{\partial \mu_k}\sum_{i=1}^{n}\sum_{k=1}^{g}\hat{\tau}_{ik}{}^{(0)}\left(-\frac{1}{2}\log(2\pi) - log\sigma_k - \frac{1}{2}\frac{(y_i - \mu_k)^2}{\sigma_k{}^2}\right) = 0$$

$$2\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} \left(\frac{y_i - \mu_k}{2\sigma_k{}^2} \right) = 0$$

$$\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} y_i = \sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} \mu_k$$

$$\hat{\mu}_k{}^{(1)} = \frac{\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} y_i}{\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)}}$$

Para obtener el estimador de $\sigma_k{}^2$ tenemos,

$$\frac{\partial}{\partial \sigma_k{}^2} \sum_{i=1}^{n} \sum_{k=1}^{g} \hat{\tau}_{ik}{}^{(0)} \left(-\frac{1}{2}\log(2\pi) - \frac{log\sigma_k{}^2}{2} - \frac{1}{2}\frac{(y_i - \mu_k)^2}{\sigma_k{}^2} \right) = 0$$

$$-\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} \frac{1}{2\sigma_k{}^2} + \sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} \frac{(y_i - \mu_k)^2}{2(\sigma_k{}^2)^2} = 0$$

$$\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} \frac{(y_i - \mu_k)^2}{\sigma_k{}^2} = \sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)}$$

$$\sigma_k{}^2 = \frac{\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} (y_i - \mu_k)^2}{\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)}}$$

Usando la estimación de μ_k , obtenemos una estimación de σ_k,

$$\hat{\sigma}_k{}^{(1)} = \sqrt{\frac{\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)} \left(y_i - \hat{\mu}_k{}^{(1)}\right)^2}{\sum_{i=1}^{n} \hat{\tau}_{ik}{}^{(0)}}}$$

Valores iniciales

Los valores iniciales sobre los que el algoritmo comienza a iterar, y que son implementados en muchas situaciones, se toman dividiendo la muestra en g particiones y sobre cada una de ellas se calcula la media de las observaciones. Estos valores se representan por $\hat{\mu}_1{}^{(0)}, \hat{\mu}_2{}^{(0)}, \ldots, \hat{\mu}_g{}^{(0)}$. Respecto a los pesos todos se toman

semejantes según $\pi_1{}^{(0)} = \pi_2{}^{(0)} = \cdots = \pi_g{}^{(0)} = 1/g$. Existen otras formas de tomar los valores iniciales, por ejemplo, ver Finch (1989).

Criterio de parado

Para detener las iteraciones vamos a considerar la diferencia relativa

$$l(\boldsymbol{\psi}|y) = \sum_{i=1}^{n} \log\left\{\pi_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(y_i-\mu_1)^2}{2\sigma_1^2}} + \pi_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(y_i-\mu_2)^2}{2\sigma_2^2}}\right\}$$

$$\frac{\left|l(\boldsymbol{\psi}|y)_{n+1} - l(\boldsymbol{\psi}|y)_n\right|}{l(\boldsymbol{\psi}|y)_n} \qquad n = 1,2,3$$

Que se utiliza más por su adimensionalidad. Y se detiene el proceso cuando el valor máximo de dicha diferencia es menor que 10^{-7}.

$$\frac{\left|l(\boldsymbol{\psi}|y)_{n+1} - l(\boldsymbol{\psi}|y)_n\right|}{l(\boldsymbol{\psi}|y)_n} < 0.0000001$$

Cálculo del error estándar mediante el método Bootstrap

El enfoque tradicional de la inferencia estadística se basa en modelos idealizados y suposiciones. A menudo, las expresiones para medidas de precisión, como el error estándar, se basan en la teoría asintótica y no están disponibles para muestras pequeñas, ver [4]. Una alternativa moderna a la aproximación tradicional es el método bootstrap, presentado por Efron (1979), ver [2] y [3]. El bootstrap es un método de remuestreo intensivo, que es ampliamente aplicable y permite el tratamiento de más modelos realistas.

Supongamos que $X = \{x_1, x_2, \dots, x_n\}$ es una muestra aleatoria observada de una distribución con función de distribución $F(x)$. Si a partir de X se selecciona aleatoriamente X^* entonces.

$$P[X^* = x_i] = \frac{1}{n}, \qquad i = 1,2, \dots, n$$

El remuestreo genera una muestra aleatoria $X^* = \{x_1^*, x_2^*, \dots, x_n^*\}$ mediante el muestreo con reemplazamiento de X. Las variables aleatorias X_j^* son

independientes e idénticamente distribuidas de manera uniforme en el conjunto de $\{x_1, x_2, \dots, x_n\}$

La función de distribución empírica, $F_n(x)$, es un estimador de $F(x)$. Puede probarse que $F_n(x)$ es un estadístico suficiente de $F(x)$, es decir, toda la información sobre $F(x)$ contenida en la muestra está también contenida en $F_n(x)$. Aun más, $F_n(x)$ es en sí misma la función de distribución de una variable aleatoria, a saber, la variable aleatoria que se distribuye de manera uniforme en el conjunto $X = \{x_1, x_2, \dots, x_n\}$. Por tanto, la función de distribución empírica $F_n(x)$ es la función de distribución de X^*.

Así, en bootstrap, pueden considerarse dos aproximaciones. Por una parte, F_n es una aproximación de F_X y, por otra, la función de distribución empírica F_n^* de las réplicas bootstrap es una aproximación de $F_n(x)$. El remuestreo a partir de X equivale a generar muestras aleatorias de la distribución $F_n(x)$. Las dos aproximaciones pueden ser representadas mediante el diagrama

$$\begin{array}{ccccc} F & \rightarrow & X & \rightarrow & F_n \\ F_n & \rightarrow & X^* & \rightarrow & X_n^* \end{array}$$

Para generar una muestra aleatoria bootstrap mediante remuestreo de X, basta generar n números enteros aleatorios $\{i_1, i_2, \dots, i_n\}$ uniformemente distribuidos en $1, \dots, n$ y seleccionar la muestra bootstrap $X^* = \{x_{i_1}, x_{i_2}, \dots, x_{i_n}\}$.

Estimación Bootstrap del error estándar

El algoritmo bootstrap, para estimar el error estándar de un estimador $\hat{\theta} = s(X)$ del parámetro θ se puede obtener mediante los siguientes pasos:

De la muestra inicial $X_1, X_2, \dots, X_n$, genere B muestras de arranque independientes,

$$\begin{array}{rcl} X^{*(1)} & \sim & X_1^{*(1)}, \dots, X_n^{*(1)} \\ X^{*(2)} & \sim & X_1^{*(2)}, \dots, X_n^{*(2)} \\ & \dots & \\ X^{*(B)} & \sim & X_1^{*(B)}, \dots, X_n^{*(B)} \end{array}$$

Evaluar,

$$\hat{\theta}^{*(b)} = s\left(X^{*(b)}\right) \quad ; \quad b = 1, \dots, B$$

Estimamos el error estándar $se(\hat{\theta})$ por la desviación estándar de las repeticiones B,

$$\widehat{se}_{boot}(\hat{\theta}) = \left[\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\theta}^{*(b)} - \hat{\theta}^{*(.)}\right)^2\right]^{1/2}$$

Donde

$$\hat{\theta}^{*(.)} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*(b)}$$

II. IMPLEMENTACIÓN DEL ALGORITMO

El algoritmo fue codificado en Python 3 y diseñado para trabajar en Linux, la programación está orientado a objetos, cuenta con dos clases, la primera denominada "*Ventana*" encargada de la interfaz (desarrollada con la librería Tkinter) y la segunda llamada *"MaximizacionEsperanza".*

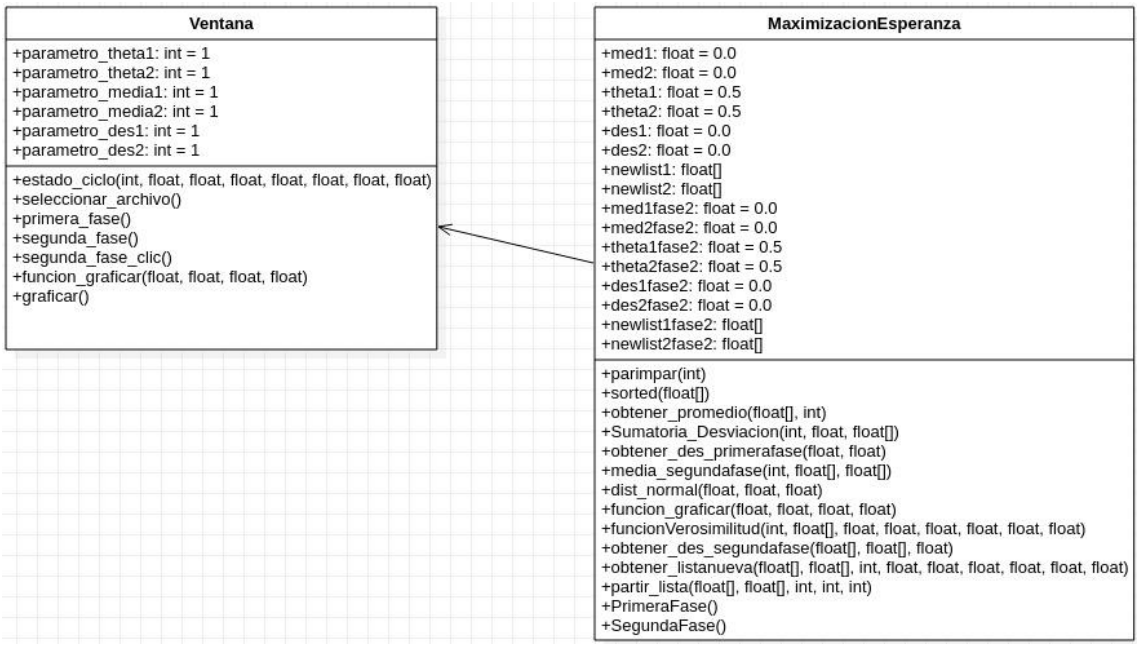


Imagen 3. Diagrama de Clases del Algoritmo Maximización de la Esperanza (EM)

En la Imagen 3 se muestra el diagrama de clases donde se exponen los atributos y métodos codificados para la implementación del algoritmo en un lenguaje de alto nivel, tales métodos se explican uno a uno en el siguiente apartado para un mejor entendimiento.

En la Imagen 4, el diagrama de Casos de Uso muestra la interacción entre el usuario final y el software obtenido después de codificar el algoritmo de EM, como se observa, el actor Usuario Final es capaz de ingresar los datos de arranque del algoritmo, después solicita la ejecución de la primera fase obteniendo resultados en la pantalla, lo que habilita la opción de ejecutar segunda fase y obtener las gráficas finales resultantes de la ejecución del algoritmo EM. El diagrama muestra la interacción esperada entre el usuario final y el producto de software resultante.

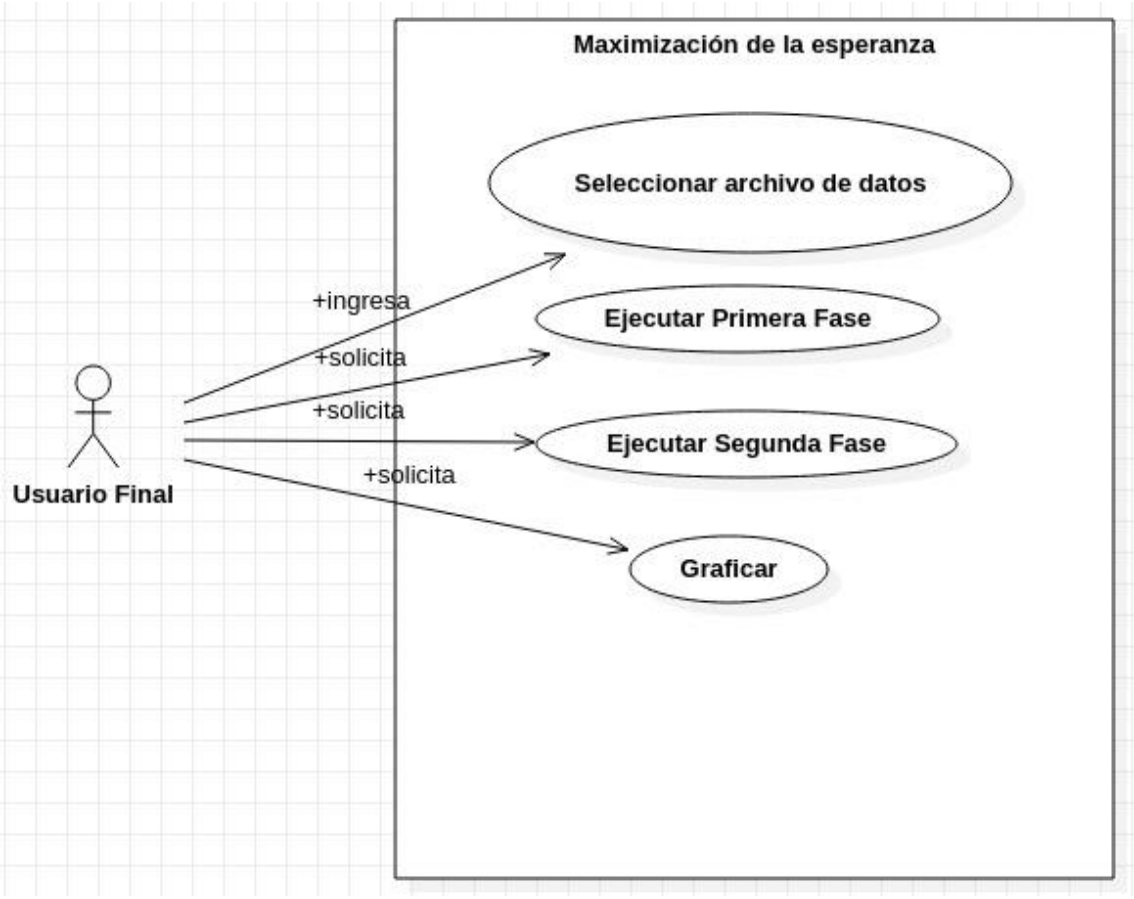


Imagen 4. Diagrama de Casos de Uso del Algoritmo Maximización de la Esperanza (EM)

Código del algoritmo EM

El algoritmo se divide en dos fases, la primera tiene los siguientes valores iniciales de theta:

$$\boldsymbol{\pi_1 = \pi_2 = 0.5}$$

Y mediante cálculos que se describen a continuación obtiene valores de los estimadores $\hat{\mu}_1^{(0)}, \hat{\mu}_1^{(0)}, \hat{\sigma}_1^{(0)}, \hat{\sigma}_2^{(0)}, \pi_1^{(0)}$ y $\pi_2^{(0)}$.

Primera fase

```
#Identificamos con base en el número de elementos de la lista como se va a
realizar la división de elementos
        tipo=self.parimpar(count_elementos)
#Dividimos la lista ordenada en dos partes iguales en caso de ser par
        if tipo == 1:
            val_x= val_y =int(count_elementos/2)
#En caso de que el número de elementos sea impar se divide entre dos y a la
segunda lista se le asigna +1 elementos
        else:
```

```
            val_x= int(count_elementos/2)
            val_y= val_x + 1
#de este modo definimos los arreglos en donde se guardaran los datos de la
lista.
        x = [0] * val_x
        y = [0] * val_y


#Una vez que definimos como dividir la lista original procedemos a ordenar los
elementos para almacenarlos en los arreglos x[] y y[]
        cs=sorted(z)
#con la función partir lista llenamos los dos arreglos
        self.partir_lista(cs,x,0,val_x,0)
        self.partir_lista(cs,y,val_x,count_elementos,1)
```

A continuación, se presentan las funciones usadas en esta primera parte de codificación:

```
def partir_lista (self,lista_origen,lista_destino,lim_izq,lim_der,mitad):
#esta función recibe la lista ordenada, el arreglo destino, límites del
arreglo y si es la primera o segunda mitad.
        k=0
        if mitad == 0:
            for k in range(lim_izq,lim_der):
                lista_destino[k]=lista_origen[k]
            return lista_destino
        if mitad == 1:
            for k in range(lim_izq,lim_der):
                lista_destino[k-lim_izq]=lista_origen[k]
            return lista_destino


def parimpar(self,count_e):
#esta función recibe el número de elementos de la lista
        if count_e % 2 == 0:
            return 1
        else:
            return 0
```

La siguiente parte de código detalla cómo se obtienen medias, desviaciones y thetas actualizados para iniciar la segunda fase.

Para obtener las medias sacamos el promedio de cada una de las listas:

$$\hat{\mu}_1{}^{(0)} = \frac{1}{n_1}\sum_{s=1}^{n_1} y_s$$

$$\hat{\mu}_2{}^{(0)} = \frac{1}{n_2}\sum_{s=1}^{n_2} y_s$$

Lo cual se codifico de la manera siguiente:

```
#Obtenemos las dos medias de la primera fase calculando el promedio de cada
nueva lista.
        med1=self.obtener_promedio(x,val_x)
        med2=self.obtener_promedio(y,val_y)
```

usando la función:

```
def obtener_promedio (self,lista,tam_lista):
#esta función recibe la lista ordenada y el numero de elementos del arreglo.
        suma=0.0
        for k in range(0,tam_lista):
            suma=lista[k]+suma
        suma=suma/tam_lista
        return suma
```

Obtenemos las dos desviaciones usando las medias anteriores:

$$\hat{\sigma}_1^{(0)} = \sqrt{\frac{1}{n_1 - 1}\sum_{s=1}^{n_1}\left(y_s - \hat{\mu}_1^{(0)}\right)^2}$$

$$\hat{\sigma}_2^{(0)} = \sqrt{\frac{1}{n_2 - 1}\sum_{s=1}^{n_2}\left(y_s - \hat{\mu}_2^{(0)}\right)^2}$$

Lo cual se codifico de la manera siguiente:

```
#Obtenemos las dos desviaciones haciendo uso de las medias obtenidas
previamente.
        suma1= self.Sumatoria_Desviacion(val_x,med1,x)
        suma2= self.Sumatoria_Desviacion(val_y,med2,y)
        div_des1=1.0/float(val_x -1)
        div_des2=1.0/float(val_y -1)
        des1=self.obtener_des_primerafase(div_des1,suma1)
        des2=self.obtener_des_primerafase(div_des2,suma2)
```

usando las funciones:

```
def Sumatoria_Desviacion (self,tam_lista, media,lista):
#esta función recibe el numero de elementos del arreglo, la media y la lista
ordenada.
        suma=0.0
        for k in range(0,tam_lista):
            suma=pow((lista[k]-media),2)+suma
        return suma
def obtener_des_primerafase(self,a,b):
```

```
#esta función recibe el resultado de 1/elementos de la mitad-1 y el resultado
de la sumatoria_desviacion.
        desviacion=0.0
        desviacion=(a*b)**0.5
        return desviacion
```

Obtenemos dos nuevas listas, cada una del tamaño de la lista original las cuales usaremos también en la segunda fase:

$$\hat{\tau}_1^{(0)} = \frac{\pi_1 \, N(y_i \,|\hat{\mu}_1^{(0)} \hat{\sigma}_1^{(0)})}{\pi_1 \, N(y_i \,|\hat{\mu}_1^{(0)} \hat{\sigma}_1^{(0)}) + \pi_2 \, N(y_i \,|\hat{\mu}_2^{(0)} \hat{\sigma}_2^{(0)})}$$

$$\hat{\tau}_2^{(0)} = \frac{\pi_2 \, N(y_i \,|\hat{\mu}_2^{(0)} \hat{\sigma}_2^{(0)})}{\pi_1 \, N(y_i \,|\hat{\mu}_1^{(0)} \hat{\sigma}_1^{(0)}) + \pi_2 \, N(y_i \,|\hat{\mu}_2^{(0)} \hat{\sigma}_2^{(0)})}$$

donde: $N(y_i \,|\mu_1 \, \sigma_1)$ es la fórmula de la distribución normal

$$f_{(x)} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Lo cual se codifico de la manera siguiente:

```
#para obtener los dos nuevos valores de theta creamos dos nuevas listas usando
la lista original ordenada y los parámetros media y desviación obtenidos
previamente así como los thetas iniciales 0.5.

self.obtener_listanueva(cs,newlist1,count_elementos,theta1,theta2,des1,des2,me
d1,med2)

self.obtener_listanueva(cs,newlist2,count_elementos,theta2,theta1,des2,des1,me
d2,med1)
```

usando la función:

```
def obtener_listanueva
(self,lista_origen,lista_destino,tam_lista,t1,t2,d1,d2,m1,m2):
#esta función recibe la lista ordenada, el arreglo destino, el número de
elementos del arreglo,los dos parámetros theta, los dos parámetros media y los
dos parámetros desviación.
        for k in range(0,tam_lista):

            lista_destino[k]=(t1*self.dist_normal(lista_origen[k],m1,d1))/(((t
            1*self.dist_normal(lista_origen[k],m1,d1))+
                ((t2*self.dist_normal(lista_origen[k],m2,d2)))))
```

Por último, obtenemos dos nuevos thetas los cuales usaremos en la segunda fase:

$$\pi_1^{(0)} = \frac{1}{n} \sum_{s=1}^{n} \tau_{i1}$$

$$\pi_2{}^{(0)} = \frac{1}{n}\sum_{s=1}^{n} \tau_{i2}$$

Lo cual se codifico de la manera siguiente:

```
#Finalmente Obtenemos los dos nuevos valores de theta.
        theta1=self.obtener_promedio(newlist1,count_elementos)
        theta2=self.obtener_promedio(newlist2,count_elementos)
```

usando la función:

```
def obtener_promedio (self,lista,tam_lista):
#esta función recibe la lista ordenada y el numero de elementos del arreglo.
        suma=0.0
        for k in range(0,tam_lista):
            suma=lista[k]+suma
        suma=suma/tam_lista
        return suma
```

Segunda fase

En esta fase, el algoritmo empieza a iterar y se detiene exactamente en la iteración donde se cumple,

$$\frac{\left|l(\boldsymbol{\psi}|y)_{n+1} - l(\boldsymbol{\psi}|y)_n\right|}{l(\boldsymbol{\psi}|y)_n} \quad n = 1,2,3$$

$$\frac{\left|l(\boldsymbol{\psi}|y)_{n+1} - l(\boldsymbol{\psi}|y)_n\right|}{l(\boldsymbol{\psi}|y)_n} \quad < 10^{-7}$$

$$l(\boldsymbol{\psi}|y) = \sum_{i=1}^{n} \log\left\{\pi_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(y_i-\mu_1)^2}{2\sigma_1^2}} + \pi_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(y_i-\mu_2)^2}{2\sigma_2^2}}\right\}.$$

los valores de theta $\pi_1{}^{(0)} = 0.497$, $\pi_2{}^{(0)} = 0.502$ y las dos listas $\hat{\tau}_1{}^{(0)}$ y $\hat{\tau}_2{}^{(0)}$ obtenidos en la primera fase serán usados en la primera corrida de esta fase y en la siguiente corrida i los obtenidos en la corrida $i-1$.

Haciendo uso de la lista original de datos y las dos nuevas listas creadas en la primera fase (o en la corrida anterior desde la segunda iteración) obtenemos dos nuevas medias $\hat{\mu}_1{}^{(i)}$ y $\hat{\mu}_2{}^{(i)}$.

$$\hat{\mu}_1{}^{(i)} = \frac{\sum_{s=1}^{n}(\hat{\tau}_{1s}{}^{(i)} y_s)}{\sum_{s=1}^{n} \hat{\tau}_{1s}{}^{(i)}}$$

$$\hat{\mu}_2{}^{(i)} = \frac{\sum_{s=1}^{n}(\hat{\tau}_{2s}{}^{(i)} y_s)}{\sum_{s=1}^{n} \hat{\tau}_{2s}{}^{(i)}}$$

Lo cual se codifico de la manera siguiente:

```
# Obtenemos las dos medias
        med1fase2=self.media_segundafase(count_elementos,w,z)
        med2fase2=self.media_segundafase(count_elementos,w,x)
```

usando la función:

```
def media_segundafase (self,tam_lista,lista_inicial,lista_nueva):
#esta función recibe el número de elementos del arreglo, la lista ordenada y
la lista nueva creada en la corrida anterior.

        suma=0.0
        suma2=0.0
        for k in range(0,tam_lista):
                suma =lista_nueva[k]*lista_inicial[k]+suma
                suma2=lista_nueva[k]+suma2
                total=suma/suma2
        return total
```

Obtenemos dos nuevas desviaciones haciendo uso de las dos medias que previamente se obtuvieron

$$\hat{\sigma}_1{}^{(i)} = \sqrt{\frac{\sum_{s=1}^{n} \hat{\tau}_{1s}{}^{(i)} (y_s - \hat{\mu}_1{}^{(i)})^2}{\sum_{s=1}^{n} \hat{\tau}_{1s}{}^{(i)}}}$$

$$\hat{\sigma}_2{}^{(i)} = \sqrt{\frac{\sum_{s=1}^{n} \hat{\tau}_{2s}{}^{(i)} (y_s - \hat{\mu}_2{}^{(i)})^2}{\sum_{s=1}^{n} \hat{\tau}_{2s}{}^{(i)}}}$$

Lo cual se codifico de la manera siguiente:

```
# Obtenemos las dos desviaciones haciendo uso de las medias obtenidas
previamente
        des1fase2=self.obtener_des_segundafase(z,w,med1fase2)
        des2fase2=self.obtener_des_segundafase(x,w,med2fase2)
```

usando la función:

```
def obtener_des_segundafase (self,lista_nueva,lista_inicial,med):
#esta función recibe la lista nueva creada en la corrida anterior, la lista
original ordenada y la media creada en esta corrida.

        suma=0.0
        suma2=0.0
        for k in range(0,count_elementos):
```

```
            suma=(lista_nueva[k]*(lista_inicial[k]-med)**2)+suma
            suma2=lista_nueva[k]+suma2

        total=(suma/suma2)**0.5
        return total
```

Obtenemos dos nuevas listas, cada una del tamaño de la lista original las cuales usaremos también en la siguiente corrida:

$$\hat{\tau}_1^{(i)} = \frac{\pi_1^{(i)}\, N(y_i \,|\hat{\mu}_1^{(0)}\hat{\sigma}_1^{(0)})}{\pi_1^{(0)}\, N(y_i \,|\hat{\mu}_1^{(0)}\hat{\sigma}_1^{(0)}) + \pi_2^{(0)}\, N(y_i \,|\hat{\mu}_2^{(0)}\hat{\sigma}_2^{(0)})}$$

$$\hat{\tau}_2^{(i)} = \frac{\pi_2^{(0)}\, N(y_i \,|\hat{\mu}_2^{(0)}\hat{\sigma}_2^{(0)})}{\pi_1^{(0)}\, N(y_i \,|\hat{\mu}_1^{(0)}\hat{\sigma}_1^{(0)}) + \pi_2^{(0)}\, N(y_i \,|\hat{\mu}_2^{(0)}\hat{\sigma}_2^{(0)})}$$

donde: $N(y_i \,|\mu_1\, \sigma_1)$ es la fórmula de la distribución normal

$$f_{(x)} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Lo cual se codifico de la manera siguiente:

```
#para obtener los dos nuevos valores de theta creamos dos nuevas listas usando
la lista original ordenada y los parámetros media y desviación obtenidos
previamente así como los thetas iniciales 0.5.

self.obtener_listanueva(w,newlist1fase2,count_elementos,y[0],y[1],des1fase2,de
s2fase2,med1fase2,med2fase2)
self.obtener_listanueva(w,newlist2fase2,count_elementos,y[1],y[0],des2fase2,de
s1fase2,med2fase2,med1fase2)
```

usando la función:

```
def obtener_listanueva
(self,lista_origen,lista_destino,tam_lista,t1,t2,d1,d2,m1,m2):
#esta función recibe la lista ordenada, el arreglo destino, el número de
elementos del arreglo,los dos parámetros theta, los dos parámetros media y los
dos parámetros desviación.
        for k in range(0,tam_lista):

            lista_destino[k]=(t1*self.dist_normal(lista_origen[k],m1,d1))/(((t
            1*self.dist_normal(lista_origen[k],m1,d1))+
                ((t2*self.dist_normal(lista_origen[k],m2,d2)))))
```

Por último, obtenemos dos nuevos valores para las variables thetas los cuales usaremos en la siguiente corrida:

$$\pi_1^{(i)} = \frac{1}{n}\sum_{s=1}^{n} \tau_{i1}$$

$$\pi_2{}^{(i)} = \frac{1}{n}\sum_{s=1}^{n} \tau_{i2}$$

Lo cual se codifico de la manera siguiente:

```
# Finalmente Obtenemos los dos nuevos valores de tetha
        theta1fase2=self.obtener_promedio(newlist1fase2,count_elementos)
        theta2fase2=self.obtener_promedio(newlist2fase2,count_elementos)
```

usando la función:

```
def obtener_promedio (self,lista,tam_lista):
#esta función recibe la lista ordenada y el numero de elementos del arreglo.
        suma=0.0
        for k in range(0,tam_lista):
            suma=lista[k]+suma
        suma=suma/tam_lista
        return suma
```

cada dos corridas calculamos la función de verosimilitud que se usara para calcular la diferencia relativa la cual es nuestro criterio de parado para detener el algoritmo:

```
#Si la corrida es par (i) guardamos los parámetros en un arreglo para usarlos
con la corrida siguiente (i+1) y calcular la diferencia relativa.
        if k % 2==0:
            compara[0]=med1fase2
            compara[1]=theta1fase2
            compara[2]=des1fase2
            compara[3]=med2fase2
            compara[4]=theta2fase2
            compara[5]=des2fase2
#Si la corrida es impar (i+1) calculamos las funciones de verosimilitud de
#esta y de la corrida (i) para obtener la diferencia relativa y en caso de
#ser menor que la cota parar el ciclo.
            if k % 2==1:

valor1=self.funcionVerosimilitud(count_elementos,w,theta1fase2,med1fase2,des1f
ase2,theta2fase2,med2fase2,des2fase2)

valor2=self.funcionVerosimilitud(count_elementos,w,compara[1],compara[0],compa
ra[2],compara[4],compara[3],compara[5])
            diferencia=abs(valor2-valor1)/(valor2)
            if diferencia <cota:
                band=k
                miVentana.estado_ciclo(998-
band,100,theta1fase2,theta2fase2,med1fase2,med2fase2,des1fase2,des2fase2)
                break
```

usando la función:

```
def funcionVerosimilitud (self,e,arreglo,t1,t2,m1,m2,d1,d2):
```

```
#La función recibe la cantidad de datos, la lista original de n datos
ordenada, los dos parámetros theta, los dos parámetros media y los dos
parámetros desviación.
        suma=0.0
        for k in range(0,e):
         suma=math.log1p((t1*(1/(((2*math.pi)**0.5)*d1))*((2.718281)**(-
((arreglo[k]-
m1)**2)/(2*d1**2))))+(t2*(1/(((2*math.pi)**0.5)*d2))*((2.718281)**(-
((arreglo[k]-m2)**2)/(2*d2**2)))))+suma
        return suma
```

Código del algoritmo Bootstrap

Generamos b listas nuevas basándonos en la lista original:

Se generan *n* números aleatorios, *n* es el número de elementos de la lista original.

```
def generar_num_alea ( lim1, lim2, array,x):
#limite izq, limite der, arreglo de salida y cantidad de numeros generados.

    for k in range(0,x):
        array[k]=random.uniform(lim1,lim2)
```

Después se generan el mismo número de intervalos iguales de medida *x* la cual es el resultado de la división $\frac{1}{n}$ y se asocia cada número generado a cada intervalo, si el numero aleatorio pertenece al intervalo i entonces se inserta el numero en la posición i de la lista original en la lista nueva.

```
def asociar ( array_entrada,x,salto):
#arreglo de entrada, cantifad de numeros a colocar, tamano intervalo.

    for j in range(0,x):#cantidad de numeros
        for k in range(0,x):#cantidad intervalos
            if array_entrada[j]>salto*k and
(array_entrada[j]<(salto*k)+salto):
                print z[k]

cantidad_num=569
intervalo=0.00175746
arreglo = [0] * cantidad_num
z = [0] * cantidad_num
for j in range(0, 569):
    z[j]=float(raw_input())
generar_num_alea(0,1,arreglo,cantidad_num)
asociar(arreglo,cantidad_num,intervalo)
```

Una vez generadas las *b* listas con el script *generador_listas.sh* aplicamos el algoritmo EM a cada una con el script *correrx1000.sh* lo cual nos dará *b* de cada uno de los siguientes parámetros $\hat{\mu}_1^{(i)}, \hat{\mu}_1^{(i)}, \hat{\sigma}_1^{(i)}, \hat{\sigma}_2^{(i)}, \pi_1^{(i)}$ y $\pi_2^{(i)}$ los cuales obtendremos de

la última corrida de cada lista en la que paro el algoritmo mediante la función de verosimilitud usando los scripts *obtener_valores_finales.sh* para recuperar los datos de la última corrida y *obtener_parametros_finales.sh* para separar los parámetros en archivos diferentes.

Con estos datos podemos obtener el error estándar de cada parámetro mediante el algoritmo bootstrap.

```
def promedio (e,oro):
  suma=0.0
  for k in range(0,e):
   suma=oro[k]+suma
  return suma/1000
def calculoError (e,arreglo,prom):
  suma=0.0
  for k in range(0,e):
   suma=(arreglo[k]-prom)**2+suma
  suma2=(suma/999)**0.5
  return suma2
z = [0] * 1000
p = [0] * 1000
t = [0] * 2000
m = [0] * 2000
d = [0] * 2000
con1=0
con2=0
#leemos archivo de entrada "t_finales_exit.txt"
f = open("t_finales_exit.txt")

for j in range(0, 2000):
  t[j] = float(f.readline())
f.close()

#leemos archivo de entrada "m_finales_exit.txt"
f = open("m_finales_exit.txt")

for j in range(0, 2000):
  m[j] = float(f.readline())
f.close()
#leemos archivo de entrada "d_finales_exit.txt"
f = open("d_finales_exit.txt")

for j in range(0, 2000):
  d[j] = float(f.readline())
f.close()
#llenamos el arreglo z con las tetha1
for j in range(0, 2000):
    if (j % 2)==0:
        z[con1]=t[j]
        con1=con1+1
#llenamos el arreglo p con las tetha2
for k in range(0, 2000):
    if k % 2==1:
        p[con2]=t[k]
        con2=con2+1
```

```
#obtenemos el promedio de z
print("tetha1y2")
prom1=promedio(1000,z)
print(prom1)
#obtenemos el promedio de p
prom2=promedio(1000,p)
print(prom2)
#calculamos el error y lo imprimimos en el archivo de salida "error.txt"
print("bootstrap")
S1=calculoError(1000,z,prom1)
print(S1)
S2=calculoError(1000,p,prom2)
print(S2)
con1=0
con2=0
#llenamos el arreglo z con las media1
for j in range(0, 2000):
  if (j % 2)==0:
    z[con1]=m[j]
    con1=con1+1
#llenamos el arreglo p con las media2
for k in range(0, 2000):
  if k % 2==1:
    p[con2]=m[k]
    con2=con2+1

#obtenemos el promedio de z
print("media1y2")
prom1=promedio(1000,z)
print(prom1)
#obtenemos el promedio de p
prom2=promedio(1000,p)
print(prom2)
#calculamos el error y lo imprimimos en el archivo de salida "error.txt
print("bootstrap")
S1=calculoError(1000,z,prom1)
print(S1)
S2=calculoError(1000,p,prom2)
print(S2)
con1=0
con2=0
#llenamos el arreglo z con las des1
for j in range(0, 2000):
  if (j % 2)==0:
    z[con1]=d[j]
    con1=con1+1
#llenamos el arreglo p con las des2
for k in range(0, 2000):
  if k % 2==1:
    p[con2]=d[k]
    con2=con2+1

#obtenemos el promedio de z
print("des1y2")
prom1=promedio(1000,z)
print(prom1)
#obtenemos el promedio de p
```

```
prom2=promedio(1000,p)
print(prom2)
#calculamos el error y lo imprimimos en el archivo de salida "error.txt"
print("bootstrap")
S1=calculoError(1000,z,prom1)
print(S1)
S2=calculoError(1000,p,prom2)
print(S2)
```

III. EJECUCIÓN DEL ALGORTIMO (EM) EN UNA INSTANCIA DE DATOS Y OBTENCIÓN DEL ERROR ESTANDAR MEDIANTE BOOTSTRAP

Si el algoritmo es capaz de discriminar por medio de la variable de estudio, si el tumor es maligno o benigno, debemos considerar una mezcla de normales con dos componentes, por lo que dividimos nuestra base de datos en dos grupos iguales previamente ordenandos de menor a mayor, como el número de elementos es de $n = 569$, consideramos dos grupos, uno con los primeros $n_1 = 284$ elementos y el segundo con los $n_2 = 285$ elementos restantes.

Los valores iniciales que se determinan con base en los valores de la variable radius_mean al finalizar la primera fase son:

$$\pi_1{}^{(0)} = 0.497 \pi_2{}^{(0)} = 0.502$$

$$\hat{\mu}_1{}^{(0)} = \frac{1}{n_1}\sum_{s=1}^{n_1} y_s = 11.46 \quad \hat{\sigma}_1{}^{(0)} = \left(\frac{1}{n_1 - 1}\sum_{s=1}^{n_1}\left(y_s - \hat{\mu}_1{}^{(0)}\right)^2\right)^{1/2} = 1.33$$

$$\hat{\mu}_2{}^{(0)} = \frac{1}{n_2}\sum_{s=n_1}^{n} y_s = 16.80 \quad \hat{\sigma}_2{}^{(0)} = \left(\frac{1}{n_2 - 1}\sum_{s=n_1}^{n}\left(y_s - \hat{\mu}_2{}^{(0)}\right)^2\right)^{1/2} = 2.97$$

Bajo estas condiciones usando la lista original el algoritmo paro en la corrida 373, oobteniendo los siguientes valores para los estimadores:

$\boldsymbol{\theta_k}$	Valor de $\widehat{\boldsymbol{\theta}}_{\boldsymbol{k}}$
π_1	0.68
π_2	0.32
μ_1	12.40
μ_2	17.90
σ_1	1.86
σ_2	3.25

Tabla 2. Valores obtenidos en la corrida 373

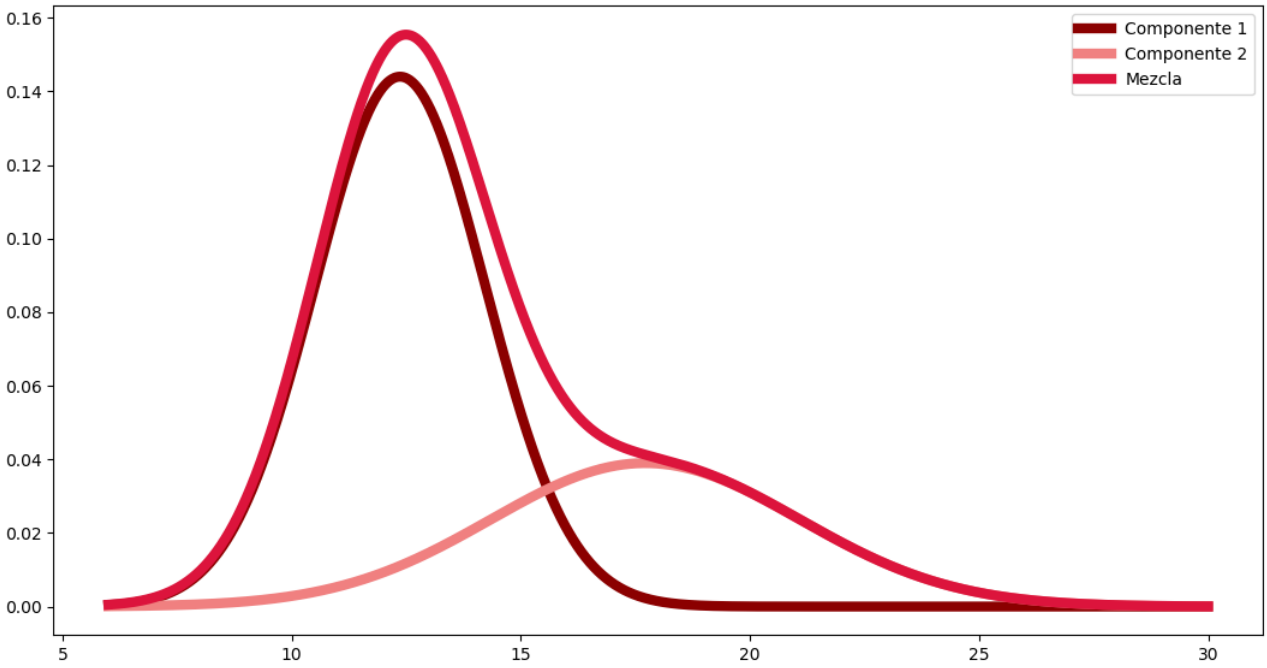
Con estos valores tenemos la forma exacta de la mezcla de normales y con los valores de la última iteración, podemos calcular las componentes, algunos datos de la lista original se muestran en la tabla 3, la tabla completa se anexa al final.

ID	Diagnóstico Benigno (B), Maligno (M)	radius_mean	Mezcla de Normales Benigno (1) Maligno (2)
8612399	**M**	**18.46**	**2**
86135501	**M**	**14.48**	**1**
86135502	**M**	**19.02**	**2**
861597	**B**	**12.36**	**1**
861598	**B**	**14.64**	**1**
861648	**B**	**14.62**	**1**
861799	**M**	**15.37**	**1**
861853	**B**	**13.27**	**1**
862009	**B**	**13.45**	**1**
862028	**M**	**15.06**	**1**
86208	**M**	**20.26**	**2**
86211	**B**	**12.18**	**1**
862261	**B**	**9.787**	**1**
862485	**B**	**11.6**	**1**
862548	**M**	**14.42**	**1**
862717	**M**	**13.61**	**1**
862722	**B**	**6.981**	**1**
862965	**B**	**12.18**	**1**
862980	**B**	**9.876**	**1**
862989	**B**	**10.49**	**1**

Tabla 3.Datos de la lista original para corroborar resultados

Podemos observar que existen diferencias, por ejemplo, en los registros $2,7,10,15\ y\ 16$ la mezcla de normales los clasifica en la componente uno (benignos), sin embargo, en el diagnostico estos registros están clasificados en la componente de malignos.

En la gráfica 3 se muestra la forma de la mezcla de normales y sus dos componentes; componente 1 (benignos) y componente 2 (malignos).



Gráfica 3. Mezcla de normales y componentes por separado

Utilizando el algoritmo bootstrap para 1000 remuestreos obtenemos los errores estándar de cada uno de los parámetros como se muestra en el siguiente cuadro.

$\boldsymbol{\theta_k}$	Valor de $\widehat{\boldsymbol{\theta}}_{\boldsymbol{k}}$	$\widehat{\boldsymbol{se}}_{\boldsymbol{boot}}(\widehat{\boldsymbol{\theta}})$
$\boldsymbol{\pi_1}$	0.68	0.05
$\boldsymbol{\pi_2}$	0.32	0.05
$\boldsymbol{\mu_1}$	12.40	0.15
$\boldsymbol{\mu_2}$	17.90	0.68
$\boldsymbol{\sigma_1}$	1.86	0.10
$\boldsymbol{\sigma_2}$	3.25	0.45

Tabla 4. Error estándar

Conclusiones

Se puede observar que, de un total de 569 observaciones, 357 que representan el 62.7% del total, indican la ausencia de células cancerosas, mientras que 212 que representan el 37.3% del total muestran la presencia de células cancerosas. Con el modelo de mezcla de normales, se encontró que, del total de observaciones, 420 que representan el 73.8%, indican la ausencia de células cancerosas, mientras que 149 que representan 26.2% del total, muestran la presencia de células cancerosas.

En los datos encontrados mediante la mezcla de normales, se observa que en el caso de tumores benignos existe una coincidencia de 349 casos, que representan el 97.8% del total de casos benignos, con una diferencia de 8 casos. Mientras que el caso de los malignos existe una coincidencia de 141 casos que representa el 66.5% del total de los casos malignos, con una diferencia de 71 caso.

Se concluye que el modelo en el caso de tumores benignos tiene una predicción con buena aceptación, mientras que en el caso de tumores malignos la predicción no sería muy buena. En este último caso se debe considerar que en el estudio del Dr. Wolberg nos indica que "El porcentaje es inusualmente grande; el conjunto de datos no representa en este caso una distribución típica de análisis médico. Por lo general, tendremos una cantidad considerable de casos que representan negativos versus una pequeña cantidad de casos que representan tumores positivos (malignos)". Esta afirmación puede corroborar la diferencia que se encontró en los casos malignos.

BIBLIOGRAFÍA

[1] Dempster, A., Laird, N. y Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B (Methodological), 39(1): 1-38.

[2] Efron, B. (1979). Bootstrap methods: another look at the jackknife. Annals of Statistics 7, 1-26.

[3] Efron, B. and Tibshirani, R.J. (1993). An Introduction to the Bootstrap, Chapman & Hall, New York.

[4] Jamshidian, M. y Jennrich, R.I. 2000. Standard errors for EM estimation. Journal of the Royan Statistical.

[5]https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

[6] Levine, R., Casella, G., (2001). Implementation of the monte carlo EM algorithm. Journal of Computational and Graphical Statistics 10, 422-439.

[7] McCulloch, C.E. (1998). Review of "EM Algorithm and Extensions". Journal of the American Statistical Association 93:403-404.

[8] McLachlan, G. y Peel, D. 2000. Finite Mixture Models. Wiley Series in Probability and Statistics, New York.

[9] Mengerse, K., Robert, C. y Titterington, D. 2011. Mixtures: Estimation and Applications. Wiley Series in Probability and Mathematical Statistics.

[10] Schlattmann, P. 2009. Medical Applications of Finite Mixture Models. Springer, Berlin Heidelberg.

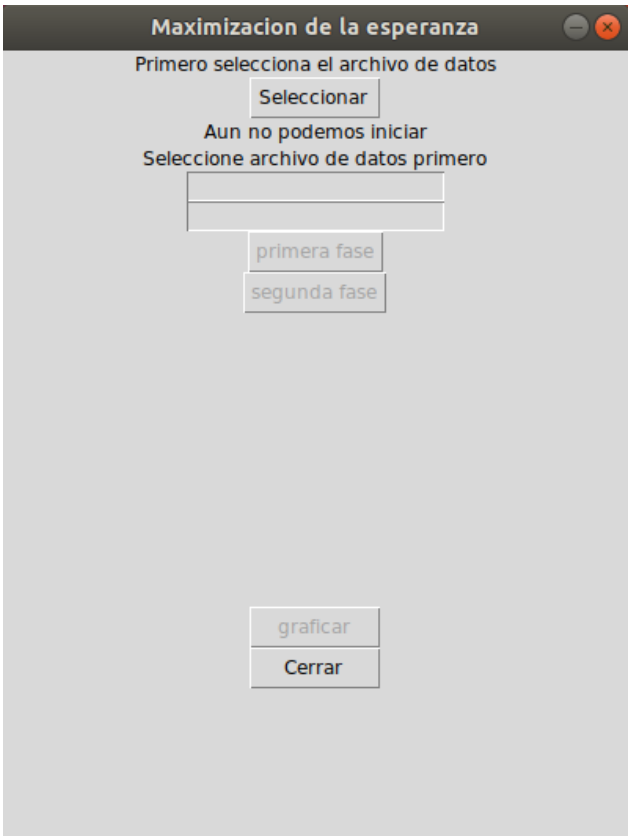
[11] White, A. C., & Lowry, W. E. (2015). Refining the role for adult stem cells as cancer cells of origin. Trends in cell biology, 25(1), 11–20. https://doi.org/10.1016/j.tcb.2014.08.008

[12] Denais, C., & Lammerding, J. (2014). Nuclear mechanics in cancer. Advances in experimental medicine and biology, 773, 435–470. https://doi.org/10.1007/978-1-4899-8032-8_20

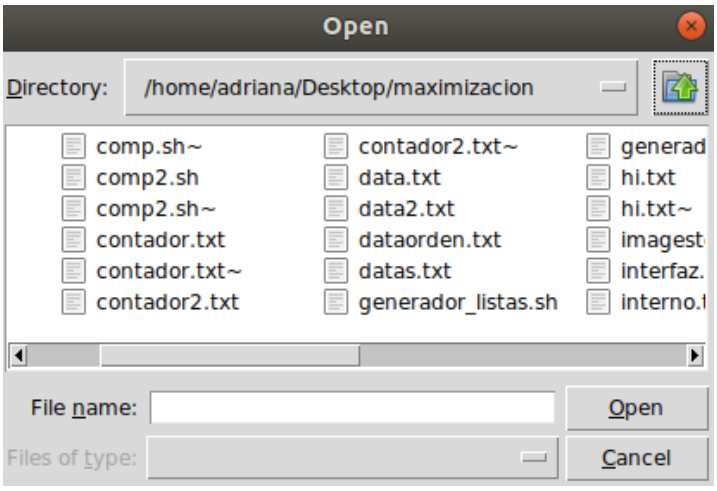
[13] Hanahan, D., & Weinberg, R. A. (2011). Hallmarks of cancer: the next generation. Cell, 144(5), 646–674. https://doi.org/10.1016/j.cell.2011.02.013

Anexo 1. Manual del usuario del programa

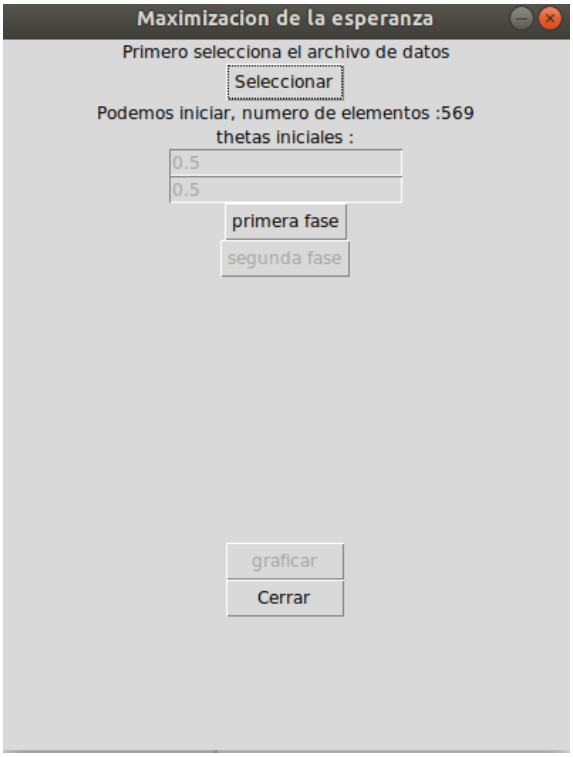
La ventana principal muestra 5 botones “seleccionar”, “primera fase”, “segunda fase”, “graficar” y “cerrar”.
Todos los botones aparecen inhabilitados excepto Seleccionar.



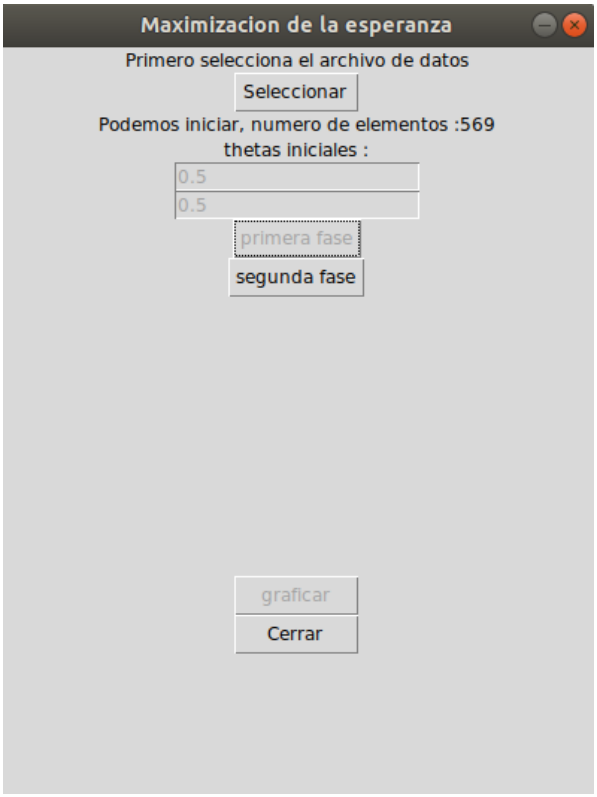
Al dar clic en Seleccionar se abre una ventana para elegir un archivo de datos numéricos, en este caso la lista original.



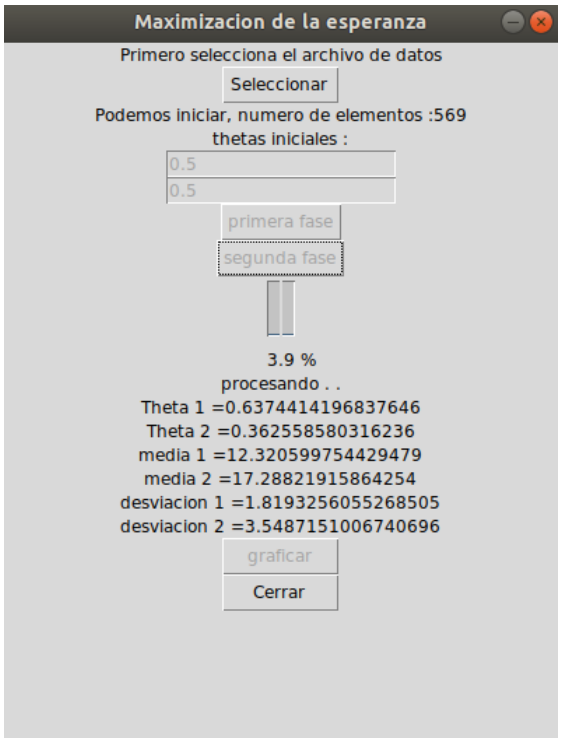
Cuando tenemos seleccionado el archivo se habilita el botón de “primera fase”.



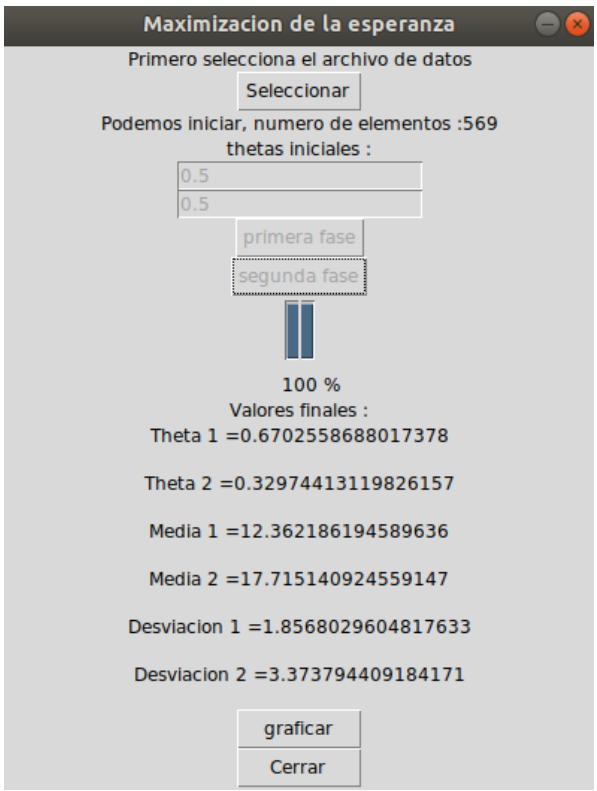
Cuando damos clic en “primera fase” se habilita el botón “segunda fase” y se ejecuta la primera fase del algoritmo EM.



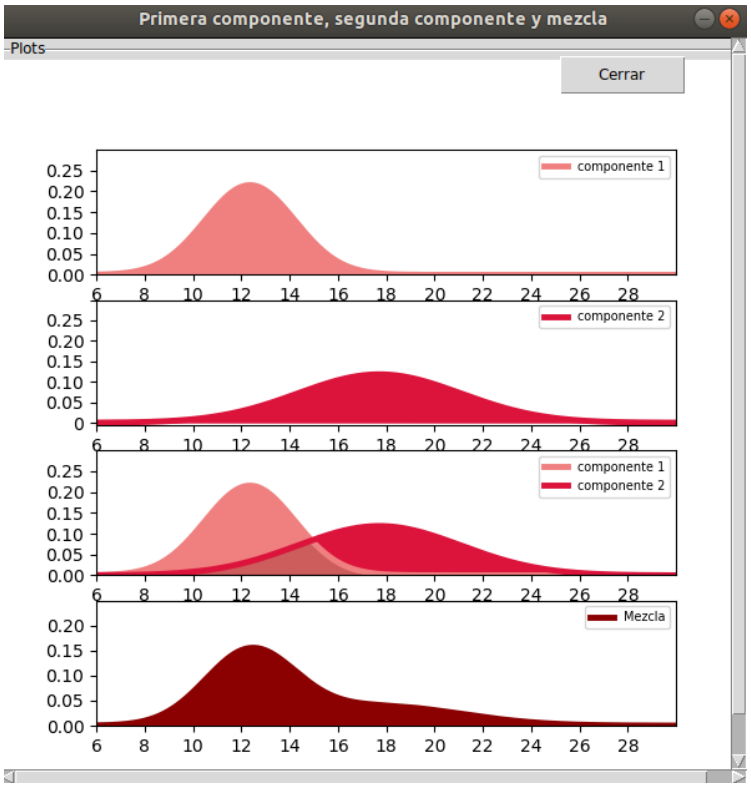
Al dar clic en “segunda fase” el algoritmo empieza a iterar y finalizara cuando se cumpla el criterio de parado.



Al terminar la ejecución se habilita el botón “graficar”.



Para visualizar las gráficas damos clic en “graficar” y se abrirá otra ventana con las gráficas generadas por el algoritmo.



Anexo 2. SCRIPTS

generador_listas.sh

```
#!/bin/bash

  echo Script para generar 1000 listas nuevas.

  echo data.txt es el archivo de entrada y nuevalista.txt sera cada lista de
salida.

  sleep 3s
for (( i=0; i<1000; i++ ))
do
python listanueva.py <data.txt > nueva_lista$i.txt
done
```

correrx1000.sh

```
#!/bin/bash

  echo Script para correr 1000 veces el algoritmo.

for (( j=0; j<1000; j++ ))
do
python3 maximizacion.py <listas_nuevas/nueva_lista$j.txt | tee sophie.txt
cp -r interno.txt
/home/adriana/Desktop/maximizacion/sininterfaz/internos/interno$j.txt

rm -r interno.txt
done
cp -r paro.txt paroverosimilitud.txt
rm -r paro.txt
```

obtener_valores_finales.sh

```
#!/bin/bash

  echo Script para obtener valores finales

  echo 1000 listas originadas aleatoriamente

for (( i=1; i<1001; i++ ))
do
let j=i-1
g="$i"
p="p"
x="$(sed -n "$i p" paroverosimilitud.txt)" #donde paro con la funcion de
verosimilitud, no es la corrida 1000 necesariamente.
let xx=x*13+1
let yy=xx+11
```

```
y="$(sed -n "$xx,$yy p"
/home/adriana/Desktop/maximizacion/sininterfaz/internos/interno$j.txt)" #datos
de la corrida x de cada lista
echo -e "lista $j \n$y" >> datos_finales.txt #escribimos los datos de la
corrida 1000 en el archivo de salida "datos_finales"
echo -e "\n" >> datos_finales.txt
done
```

obtener_parametros_finales.sh

```
#!/bin/bash

  echo Script para obtener las 2000 medias 2000 desviaciones y 2000 thetas

  echo med1,med1.1,med2,med2.1,med3,med3.1...


#obtenemos parametros del archivo de entrada "datos_finales.txt" que contiene
los datos de la corrida 1000
y="$(sed -n '4,5p' datos_finales.txt)" #primero de la lista 0
echo -e "$y" >> parametros_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
echo -e "$y" >> t_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
y="$(sed -n '8,9p' datos_finales.txt)" #primero de la lista 0
echo -e "$y" >> parametros_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
echo -e "$y" >> m_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
y="$(sed -n '12,13p' datos_finales.txt)" #primero de la lista 0
echo -e "$y" >> parametros_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
echo -e "$y" >> d_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
let "n=15"
#despues de la lista 1 a la 1000
for (( i=1; i<1000; i++ ))
do
let "o=n*$i+4"
let "m=n*$i+5"
y="$(sed -n ''$o','$m' p' datos_finales.txt)"
echo -e "$y" >> parametros_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
echo -e "$y" >> t_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
let "o=n*$i+8"
let "m=n*$i+9"
y="$(sed -n ''$o','$m' p' datos_finales.txt)"
echo -e "$y" >> parametros_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
echo -e "$y" >> m_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
let "o=n*$i+12"
let "m=n*$i+13"
y="$(sed -n ''$o','$m' p' datos_finales.txt)"
```

```
echo -e "$y" >> parametros_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"
echo -e "$y" >> d_finales.txt #los escribimos en archivo de salida
"parametros_finales.txt"

done

cp parametros_finales.txt parametros_finales_exit.txt
rm -r parametros_finales.txt
cp m_finales.txt m_finales_exit.txt
rm -r m_finales.txt
cp d_finales.txt d_finales_exit.txt
rm -r d_finales.txt
cp t_finales.txt t_finales_exit.txt
rm -r t_finales.txt
```

Anexo 3. Tabla 3 completa

ID		Variable (Eje X)	Comp_1	Comp_2	Mixtura					Num. de 1 y 2
862722	B	6.98	0.002	0.000	0.002	0.93770462	0.06229538	FALSE	1	1
921362	B	7.69	0.006	0.000	0.006	0.95437982	0.04562018	FALSE	1	2
921092	B	7.73	0.006	0.000	0.007	0.95502223	0.04497777	FALSE	1	3
92751	B	7.76	0.006	0.000	0.007	0.95553108	0.04446892	FALSE	1	4
85713702	B	8.20	0.011	0.000	0.012	0.9614024	0.0385976	FALSE	1	5
871001502	B	8.22	0.012	0.000	0.012	0.96165272	0.03834728	FALSE	1	6
91805	B	8.57	0.018	0.001	0.018	0.96486206	0.03513794	FALSE	1	7
894047	B	8.60	0.018	0.001	0.019	0.96505637	0.03494363	FALSE	1	8
858981	B	8.60	0.018	0.001	0.019	0.96506374	0.03493626	FALSE	1	9
858477	B	8.62	0.018	0.001	0.019	0.96520934	0.03479066	FALSE	1	10
872113	B	8.67	0.020	0.001	0.020	0.96557987	0.03442013	FALSE	1	11
864496	B	8.73	0.021	0.001	0.021	0.96594138	0.03405862	FALSE	1	12
903483	B	8.73	0.021	0.001	0.022	0.96599204	0.03400796	FALSE	1	13
9010333	B	8.88	0.024	0.001	0.025	0.96682354	0.03317646	FALSE	1	14
859711	B	8.89	0.025	0.001	0.025	0.96687576	0.03312424	FALSE	1	15
864726	B	8.95	0.026	0.001	0.027	0.96718396	0.03281604	FALSE	1	16
89346	B	9.00	0.027	0.001	0.028	0.96741334	0.03258666	FALSE	1	17
859471	B	9.03	0.028	0.001	0.029	0.96753867	0.03246133	FALSE	1	18
894329	B	9.04	0.029	0.001	0.030	0.96759304	0.03240696	FALSE	1	19
859196	B	9.17	0.032	0.001	0.033	0.96807942	0.03192058	FALSE	1	20
915186	B	9.27	0.035	0.001	0.036	0.96836385	0.03163615	FALSE	1	21
917092	B	9.30	0.036	0.001	0.037	0.96843446	0.03156554	FALSE	1	22
924342	B	9.33	0.037	0.001	0.039	0.96852626	0.03147374	FALSE	1	23
905539	B	9.40	0.040	0.001	0.041	0.96866102	0.03133898	FALSE	1	24
905978	B	9.41	0.040	0.001	0.041	0.96867613	0.03132387	FALSE	1	25
925236	B	9.42	0.041	0.001	0.042	0.96870872	0.03129128	FALSE	1	26
901034301	B	9.44	0.041	0.001	0.042	0.96873105	0.03126895	FALSE	1	27
859464	B	9.47	0.042	0.001	0.043	0.96877722	0.03122278	FALSE	1	28
8510824	B	9.50	0.043	0.001	0.045	0.96883144	0.03116856	FALSE	1	29
882488	B	9.57	0.046	0.001	0.047	0.9689	0.0311	FALSE	1	30
898143	B	9.61	0.047	0.002	0.049	0.96893074	0.03106926	FALSE	1	31
9113778	B	9.67	0.050	0.002	0.051	0.96896093	0.03103907	FALSE	1	32
9113514	B	9.67	0.050	0.002	0.051	0.96896124	0.03103876	FALSE	1	33
915276	B	9.68	0.050	0.002	0.052	0.96896354	0.03103646	FALSE	1	34
923169	B	9.68	0.050	0.002	0.052	0.96896524	0.03103476	FALSE	1	35
875099	B	9.72	0.052	0.002	0.053	0.96896949	0.03103051	FALSE	1	36
8710441	B	9.73	0.052	0.002	0.054	0.96896921	0.03103079	FALSE	1	37
868999	B	9.74	0.052	0.002	0.054	0.96896866	0.03103134	FALSE	1	38
8910996	B	9.74	0.053	0.002	0.054	0.96896822	0.03103178	FALSE	1	39
907145	B	9.74	0.053	0.002	0.054	0.96896822	0.03103178	FALSE	1	40
9112712	B	9.76	0.053	0.002	0.055	0.96896614	0.03103386	FALSE	1	41
864033	B	9.78	0.054	0.002	0.056	0.96896037	0.03103963	FALSE	1	42
862261	B	9.79	0.054	0.002	0.056	0.96895681	0.03104319	FALSE	1	43
917897	B	9.85	0.057	0.002	0.059	0.96892315	0.03107685	FALSE	1	44
862980	B	9.88	0.058	0.002	0.060	0.96889932	0.03110068	FALSE	1	45
879804	B	9.88	0.058	0.002	0.060	0.96889932	0.03110068	FALSE	1	46
872608	B	9.90	0.059	0.002	0.061	0.96887161	0.03112839	FALSE	1	47
907367	B	10.03	0.065	0.002	0.067	0.96868948	0.03131052	FALSE	1	48
897880	B	10.05	0.066	0.002	0.068	0.96865186	0.03134814	FALSE	1	49
874158	B	10.08	0.067	0.002	0.069	0.96859091	0.03140909	FALSE	1	50
924964	B	10.16	0.071	0.002	0.073	0.96840166	0.03159834	FALSE	1	51
858970	B	10.17	0.071	0.002	0.073	0.96837525	0.03162475	FALSE	1	52
8812844	B	10.18	0.072	0.002	0.074	0.96834822	0.03165178	FALSE	1	53
8811779	B	10.20	0.072	0.002	0.075	0.96829231	0.03170769	FALSE	1	54
894604	B	10.25	0.075	0.002	0.077	0.96814161	0.03185839	FALSE	1	55
898677	B	10.26	0.075	0.002	0.078	0.96810959	0.03189041	FALSE	1	56
90317302	B	10.26	0.075	0.002	0.078	0.96810959	0.03189041	FALSE	1	57
922840	B	10.26	0.075	0.002	0.078	0.96810959	0.03189041	FALSE	1	58

924934	B	10.29	0.077	0.003	0.079	0.96800972	0.03199028	FALSE	1	59
922577	B	10.32	0.078	0.003	0.081	0.96790412	0.03209588	FALSE	1	60
88147101	B	10.44	0.084	0.003	0.087	0.96742324	0.03257676	FALSE	1	61
884437	B	10.48	0.086	0.003	0.089	0.9672417	0.0327583	FALSE	1	62
907409	B	10.48	0.086	0.003	0.089	0.9672417	0.0327583	FALSE	1	63
862989	B	10.49	0.086	0.003	0.089	0.96719462	0.03280538	FALSE	1	64
892657	B	10.49	0.086	0.003	0.089	0.96719462	0.03280538	FALSE	1	65
864292	B	10.51	0.087	0.003	0.090	0.96709841	0.03290159	FALSE	1	66
892399	B	10.51	0.087	0.003	0.090	0.96709841	0.03290159	FALSE	1	67
901315	B	10.57	0.090	0.003	0.093	0.96679317	0.03320683	FALSE	1	68
909777	B	10.57	0.090	0.003	0.093	0.96679317	0.03320683	FALSE	1	69
8910251	B	10.60	0.091	0.003	0.094	0.96663107	0.03336893	FALSE	1	70
88466802	B	10.65	0.094	0.003	0.097	0.96634655	0.03365345	FALSE	1	71
871642	B	10.66	0.094	0.003	0.097	0.96628747	0.03371253	FALSE	1	72
8910720	B	10.71	0.097	0.003	0.100	0.96598097	0.03401903	FALSE	1	73
869254	B	10.75	0.098	0.003	0.102	0.96572226	0.03427774	FALSE	1	74
87127	B	10.80	0.101	0.004	0.104	0.96538158	0.03461842	FALSE	1	75
90745	B	10.80	0.101	0.004	0.104	0.96538158	0.03461842	FALSE	1	76
923465	B	10.82	0.102	0.004	0.105	0.96523983	0.03476017	FALSE	1	77
923748	B	10.86	0.104	0.004	0.107	0.96494677	0.03505323	FALSE	1	78
911391	B	10.88	0.104	0.004	0.108	0.96479539	0.03520461	FALSE	1	79
871149	B	10.90	0.105	0.004	0.109	0.96464072	0.03535928	FALSE	1	80
891936	B	10.91	0.106	0.004	0.110	0.96456214	0.03543786	FALSE	1	81
904971	B	10.94	0.107	0.004	0.111	0.9643214	0.0356786	FALSE	1	82
855563	M	10.95	0.108	0.004	0.112	0.96423946	0.03576054	FALSE	1	83
919537	B	10.96	0.108	0.004	0.112	0.96415667	0.03584333	FALSE	1	84
909411	B	10.97	0.109	0.004	0.113	0.96407302	0.03592698	FALSE	1	85
901836	B	11.04	0.112	0.004	0.116	0.96346303	0.03653697	FALSE	1	86
905520	B	11.04	0.112	0.004	0.116	0.96346303	0.03653697	FALSE	1	87
89827	B	11.06	0.113	0.004	0.117	0.96328074	0.03671926	FALSE	1	88
904302	B	11.06	0.113	0.004	0.117	0.96328074	0.03671926	FALSE	1	89
91544002	B	11.06	0.113	0.004	0.117	0.96328074	0.03671926	FALSE	1	90
871641	B	11.08	0.113	0.004	0.118	0.96309481	0.03690519	FALSE	1	91
9013838	M	11.08	0.113	0.004	0.118	0.96309481	0.03690519	FALSE	1	92
90769601	B	11.13	0.116	0.004	0.120	0.96261378	0.03738622	FALSE	1	93
923780	B	11.13	0.116	0.004	0.120	0.96261378	0.03738622	FALSE	1	94
901011	B	11.14	0.116	0.005	0.120	0.96251475	0.03748525	FALSE	1	95
87106	B	11.15	0.116	0.005	0.121	0.96241476	0.03758524	FALSE	1	96
906290	B	11.16	0.117	0.005	0.121	0.96231381	0.03768619	FALSE	1	97
925311	B	11.20	0.118	0.005	0.123	0.96190029	0.03809971	FALSE	1	98
88203002	B	11.22	0.119	0.005	0.124	0.9616876	0.0383124	FALSE	1	99
897132	B	11.22	0.119	0.005	0.124	0.9616876	0.0383124	FALSE	1	100
897137	B	11.25	0.120	0.005	0.125	0.96136102	0.03863898	FALSE	1	101
8913049	B	11.26	0.121	0.005	0.126	0.96125011	0.03874989	FALSE	1	102
91789	B	11.26	0.121	0.005	0.126	0.96125011	0.03874989	FALSE	1	103
901549	B	11.27	0.121	0.005	0.126	0.96113818	0.03886182	FALSE	1	104
903011	B	11.27	0.121	0.005	0.126	0.96113818	0.03886182	FALSE	1	105
868871	B	11.28	0.122	0.005	0.127	0.96102521	0.03897479	FALSE	1	106
8910748	B	11.29	0.122	0.005	0.127	0.96091119	0.03908881	FALSE	1	107
883852	B	11.30	0.122	0.005	0.127	0.96079612	0.03920388	FALSE	1	108
859465	B	11.31	0.123	0.005	0.128	0.96067999	0.03932001	FALSE	1	109
88199202	B	11.32	0.123	0.005	0.128	0.96056279	0.03943721	FALSE	1	110
8911230	B	11.33	0.124	0.005	0.129	0.96044452	0.03955548	FALSE	1	111
864018	B	11.34	0.124	0.005	0.129	0.96032517	0.03967483	FALSE	1	112
916221	B	11.34	0.124	0.005	0.129	0.96032517	0.03967483	FALSE	1	113
905502	B	11.36	0.125	0.005	0.130	0.9600832	0.0399168	FALSE	1	114
89143601	B	11.37	0.125	0.005	0.130	0.95996057	0.04003943	FALSE	1	115
865137	B	11.41	0.127	0.005	0.132	0.95945885	0.04054115	FALSE	1	116
873843	B	11.41	0.127	0.005	0.132	0.95945885	0.04054115	FALSE	1	117
84348301	M	11.42	0.127	0.005	0.132	0.95933058	0.04066942	FALSE	1	118
868682	B	11.43	0.127	0.005	0.133	0.95920117	0.04079883	FALSE	1	119
869218	B	11.43	0.127	0.005	0.133	0.95920117	0.04079883	FALSE	1	120
861103	B	11.45	0.128	0.005	0.133	0.95893886	0.04106114	FALSE	1	121

89296	B	11.46	0.128	0.006	0.134	0.95880596	0.04119404	FALSE	1	122
898690	B	11.47	0.129	0.006	0.134	0.95867186	0.04132814	FALSE	1	123
911685	B	11.49	0.129	0.006	0.135	0.9584001	0.0415999	FALSE	1	124
88518501	B	11.50	0.130	0.006	0.135	0.95826242	0.04173758	FALSE	1	125
925291	B	11.51	0.130	0.006	0.136	0.95812351	0.04187649	FALSE	1	126
85759902	B	11.52	0.130	0.006	0.136	0.95798339	0.04201661	FALSE	1	127
884689	B	11.52	0.130	0.006	0.136	0.95798339	0.04201661	FALSE	1	128
893988	B	11.54	0.131	0.006	0.137	0.95769943	0.04230057	FALSE	1	129
921385	B	11.54	0.131	0.006	0.137	0.95769943	0.04230057	FALSE	1	130
906539	B	11.57	0.132	0.006	0.138	0.95726408	0.04273592	FALSE	1	131
862485	B	11.60	0.133	0.006	0.139	0.95681722	0.04318278	FALSE	1	132
893061	B	11.60	0.133	0.006	0.139	0.95681722	0.04318278	FALSE	1	133
911320501	B	11.60	0.133	0.006	0.139	0.95681722	0.04318278	FALSE	1	134
906616	B	11.61	0.133	0.006	0.139	0.95666567	0.04333433	FALSE	1	135
911366	B	11.62	0.134	0.006	0.140	0.9565128	0.0434872	FALSE	1	136
9112366	B	11.63	0.134	0.006	0.140	0.95635861	0.04364139	FALSE	1	137
863031	B	11.64	0.134	0.006	0.140	0.95620308	0.04379692	FALSE	1	138
899187	B	11.66	0.135	0.006	0.141	0.95588797	0.04411203	FALSE	1	139
91903901	B	11.67	0.135	0.006	0.141	0.95572837	0.04427163	FALSE	1	140
913512	B	11.68	0.135	0.006	0.142	0.9555674	0.0444326	FALSE	1	141
919812	B	11.69	0.136	0.006	0.142	0.95540504	0.04459496	FALSE	1	142
893783	B	11.70	0.136	0.006	0.142	0.95524129	0.04475871	FALSE	1	143
868223	B	11.71	0.136	0.006	0.143	0.95507613	0.04492387	FALSE	1	144
874373	B	11.71	0.136	0.006	0.143	0.95507613	0.04492387	FALSE	1	145
89864002	B	11.71	0.136	0.006	0.143	0.95507613	0.04492387	FALSE	1	146
8912055	B	11.74	0.137	0.007	0.143	0.95457209	0.04542791	FALSE	1	147
91550	B	11.74	0.137	0.007	0.143	0.95457209	0.04542791	FALSE	1	148
8711561	B	11.75	0.137	0.007	0.144	0.95440119	0.04559881	FALSE	1	149
91858	B	11.75	0.137	0.007	0.144	0.95440119	0.04559881	FALSE	1	150
857343	B	11.76	0.137	0.007	0.144	0.95422883	0.04577117	FALSE	1	151
892189	M	11.76	0.137	0.007	0.144	0.95422883	0.04577117	FALSE	1	152
869691	M	11.80	0.138	0.007	0.145	0.95352456	0.04647544	FALSE	1	153
904357	B	11.80	0.138	0.007	0.145	0.95352456	0.04647544	FALSE	1	154
874662	B	11.81	0.139	0.007	0.145	0.95334473	0.04665527	FALSE	1	155
853612	M	11.84	0.139	0.007	0.146	0.95279601	0.04720399	FALSE	1	156
8810528	B	11.84	0.139	0.007	0.146	0.95279601	0.04720399	FALSE	1	157
891703	B	11.85	0.140	0.007	0.147	0.95260999	0.04739001	FALSE	1	158
9111596	B	11.87	0.140	0.007	0.147	0.9522332	0.0477668	FALSE	1	159
8811523	B	11.89	0.140	0.007	0.148	0.95185001	0.04814999	FALSE	1	160
8911164	B	11.89	0.140	0.007	0.148	0.95185001	0.04814999	FALSE	1	161
905686	B	11.89	0.140	0.007	0.148	0.95185001	0.04814999	FALSE	1	162
869476	B	11.90	0.141	0.007	0.148	0.95165598	0.04834402	FALSE	1	163
864685	B	11.93	0.141	0.007	0.149	0.95106399	0.04893601	FALSE	1	164
904647	B	11.93	0.141	0.007	0.149	0.95106399	0.04893601	FALSE	1	165
857374	B	11.94	0.141	0.007	0.149	0.95086332	0.04913668	FALSE	1	166
8912909	B	11.94	0.141	0.007	0.149	0.95086332	0.04913668	FALSE	1	167
899147	B	11.95	0.142	0.007	0.149	0.95066096	0.04933904	FALSE	1	168
9110720	B	11.99	0.142	0.008	0.150	0.94983434	0.05016566	FALSE	1	169
8612080	B	12.00	0.143	0.008	0.150	0.94962331	0.05037669	FALSE	1	170
9111843	B	12.00	0.143	0.008	0.150	0.94962331	0.05037669	FALSE	1	171
895299	B	12.03	0.143	0.008	0.151	0.94897955	0.05102045	FALSE	1	172
9113816	B	12.04	0.143	0.008	0.151	0.94876134	0.05123866	FALSE	1	173
857155	B	12.05	0.143	0.008	0.151	0.94854131	0.05145869	FALSE	1	174
90250	B	12.05	0.143	0.008	0.151	0.94854131	0.05145869	FALSE	1	175
871122	B	12.06	0.143	0.008	0.151	0.94831943	0.05168057	FALSE	1	176
898678	B	12.06	0.143	0.008	0.151	0.94831943	0.05168057	FALSE	1	177
918465	B	12.07	0.144	0.008	0.151	0.94809569	0.05190431	FALSE	1	178
903554	B	12.10	0.144	0.008	0.152	0.94741314	0.05258686	FALSE	1	179
912193	B	12.16	0.145	0.008	0.153	0.94599575	0.05400425	FALSE	1	180
86211	B	12.18	0.145	0.008	0.153	0.94550732	0.05449268	FALSE	1	181
862965	B	12.18	0.145	0.008	0.153	0.94550732	0.05449268	FALSE	1	182
894090	B	12.18	0.145	0.008	0.153	0.94550732	0.05449268	FALSE	1	183
866714	B	12.19	0.145	0.008	0.153	0.94526004	0.05473996	FALSE	1	184

89511501	B	12.20	0.145	0.008	0.153	0.94501069	0.05498931	FALSE	1	185
9011495	B	12.21	0.145	0.008	0.154	0.94475926	0.05524074	FALSE	1	186
902975	B	12.21	0.145	0.008	0.154	0.94475926	0.05524074	FALSE	1	187
91544001	B	12.22	0.145	0.009	0.154	0.94450573	0.05549427	FALSE	1	188
877501	B	12.23	0.145	0.009	0.154	0.94425007	0.05574993	FALSE	1	189
8711003	B	12.25	0.145	0.009	0.154	0.94373235	0.05626765	FALSE	1	190
91376701	B	12.25	0.145	0.009	0.154	0.94373235	0.05626765	FALSE	1	191
905501	B	12.27	0.145	0.009	0.154	0.94320593	0.05679407	FALSE	1	192
9113846	B	12.27	0.145	0.009	0.154	0.94320593	0.05679407	FALSE	1	193
874839	B	12.30	0.146	0.009	0.155	0.94239972	0.05760028	FALSE	1	194
897374	B	12.30	0.146	0.009	0.155	0.94239972	0.05760028	FALSE	1	195
8610175	B	12.31	0.146	0.009	0.155	0.94212648	0.05787352	FALSE	1	196
87139402	B	12.32	0.146	0.009	0.155	0.94185097	0.05814903	FALSE	1	197
8712064	B	12.34	0.146	0.009	0.155	0.94129301	0.05870699	FALSE	1	198
875263	M	12.34	0.146	0.009	0.155	0.94129301	0.05870699	FALSE	1	199
904969	B	12.34	0.146	0.009	0.155	0.94129301	0.05870699	FALSE	1	200
91813702	B	12.34	0.146	0.009	0.155	0.94129301	0.05870699	FALSE	1	201
861597	B	12.36	0.146	0.009	0.155	0.94072571	0.05927429	FALSE	1	202
863270	B	12.36	0.146	0.009	0.155	0.94072571	0.05927429	FALSE	1	203
90251	B	12.39	0.146	0.009	0.155	0.93985687	0.06014313	FALSE	1	204
907915	B	12.40	0.146	0.009	0.155	0.93956241	0.06043759	FALSE	1	205
883539	B	12.42	0.146	0.009	0.155	0.9389661	0.0610339	FALSE	1	206
894335	B	12.43	0.146	0.010	0.155	0.93866421	0.06133579	FALSE	1	207
843786	M	12.45	0.146	0.010	0.155	0.93805284	0.06194716	FALSE	1	208
913063	B	12.45	0.146	0.010	0.155	0.93805284	0.06194716	FALSE	1	209
84501001	M	12.46	0.146	0.010	0.155	0.93774332	0.06225668	FALSE	1	210
892604	B	12.46	0.146	0.010	0.155	0.93774332	0.06225668	FALSE	1	211
914101	B	12.46	0.146	0.010	0.155	0.93774332	0.06225668	FALSE	1	212
87930	B	12.47	0.146	0.010	0.155	0.93743123	0.06256877	FALSE	1	213
914580	B	12.47	0.146	0.010	0.155	0.93743123	0.06256877	FALSE	1	214
894089	B	12.49	0.146	0.010	0.156	0.93679919	0.06320081	FALSE	1	215
901034302	B	12.54	0.145	0.010	0.156	0.93517235	0.06482765	FALSE	1	216
91505	B	12.54	0.145	0.010	0.156	0.93517235	0.06482765	FALSE	1	217
9010258	B	12.56	0.145	0.010	0.155	0.93450244	0.06549756	FALSE	1	218
8912521	B	12.58	0.145	0.010	0.155	0.93382129	0.06617871	FALSE	1	219
88147202	B	12.62	0.145	0.010	0.155	0.93242451	0.06757549	FALSE	1	220
911202	B	12.62	0.145	0.010	0.155	0.93242451	0.06757549	FALSE	1	221
86408	B	12.63	0.145	0.011	0.155	0.93206799	0.06793201	FALSE	1	222
914366	B	12.65	0.145	0.011	0.155	0.93134598	0.06865402	FALSE	1	223
89511502	B	12.67	0.144	0.011	0.155	0.93061185	0.06938815	FALSE	1	224
85922302	M	12.68	0.144	0.011	0.155	0.93024018	0.06975982	FALSE	1	225
906024	B	12.70	0.144	0.011	0.155	0.92948748	0.07051252	FALSE	1	226
891716	B	12.72	0.144	0.011	0.155	0.92872213	0.07127787	FALSE	1	227
90769602	B	12.72	0.144	0.011	0.155	0.92872213	0.07127787	FALSE	1	228
917080	B	12.75	0.143	0.011	0.154	0.92754992	0.07245008	FALSE	1	229
89382602	B	12.76	0.143	0.011	0.154	0.92715263	0.07284737	FALSE	1	230
9010598	B	12.76	0.143	0.011	0.154	0.92715263	0.07284737	FALSE	1	231
868202	M	12.77	0.143	0.011	0.154	0.92675201	0.07324799	FALSE	1	232
875093	B	12.77	0.143	0.011	0.154	0.92675201	0.07324799	FALSE	1	233
924084	B	12.77	0.143	0.011	0.154	0.92675201	0.07324799	FALSE	1	234
859487	B	12.78	0.143	0.011	0.154	0.92634804	0.07365196	FALSE	1	235
90401602	B	12.80	0.143	0.011	0.154	0.92552991	0.07447009	FALSE	1	236
873586	B	12.81	0.142	0.012	0.154	0.9251157	0.0748843	FALSE	1	237
881861	M	12.83	0.142	0.012	0.154	0.92427684	0.07572316	FALSE	1	238
911408	B	12.83	0.142	0.012	0.154	0.92427684	0.07572316	FALSE	1	239
905190	B	12.85	0.142	0.012	0.153	0.92342386	0.07657614	FALSE	1	240
8610908	B	12.86	0.141	0.012	0.153	0.92299201	0.07700799	FALSE	1	241
894855	B	12.86	0.141	0.012	0.153	0.92299201	0.07700799	FALSE	1	242
8910506	B	12.87	0.141	0.012	0.153	0.92255653	0.07744347	FALSE	1	243
908916	B	12.87	0.141	0.012	0.153	0.92255653	0.07744347	FALSE	1	244
917062	B	12.88	0.141	0.012	0.153	0.9221174	0.0778826	FALSE	1	245
924632	B	12.88	0.141	0.012	0.153	0.9221174	0.0778826	FALSE	1	246
884626	B	12.89	0.141	0.012	0.153	0.92167459	0.07832541	FALSE	1	247

8912284	B	12.89	0.141	0.012	0.153	0.92167459	0.07832541	FALSE	1	248
8913	B	12.89	0.141	0.012	0.153	0.92167459	0.07832541	FALSE	1	249
869224	B	12.90	0.141	0.012	0.153	0.92122807	0.07877193	FALSE	1	250
875878	B	12.91	0.140	0.012	0.153	0.9207778	0.0792222	FALSE	1	251
9047	B	12.94	0.140	0.012	0.152	0.91940421	0.08059579	FALSE	1	252
891670	B	12.95	0.140	0.012	0.152	0.91893865	0.08106135	FALSE	1	253
904689	B	12.96	0.139	0.012	0.152	0.91846917	0.08153083	FALSE	1	254
896864	B	12.98	0.139	0.012	0.151	0.91751837	0.08248163	FALSE	1	255
897604	B	12.99	0.139	0.013	0.151	0.91703698	0.08296302	FALSE	1	256
844981	M	13.00	0.138	0.013	0.151	0.91655155	0.08344845	FALSE	1	257
871001501	B	13.00	0.138	0.013	0.151	0.91655155	0.08344845	FALSE	1	258
9112594	B	13.00	0.138	0.013	0.151	0.91655155	0.08344845	FALSE	1	259
873357	B	13.01	0.138	0.013	0.151	0.91606204	0.08393796	FALSE	1	260
854941	B	13.03	0.138	0.013	0.151	0.91507066	0.08492934	FALSE	1	261
857810	B	13.05	0.137	0.013	0.150	0.91406256	0.08593744	FALSE	1	262
893548	B	13.05	0.137	0.013	0.150	0.91406256	0.08593744	FALSE	1	263
9010259	B	13.05	0.137	0.013	0.150	0.91406256	0.08593744	FALSE	1	264
8510653	B	13.08	0.136	0.013	0.149	0.91251845	0.08748155	FALSE	1	265
863030	M	13.11	0.136	0.013	0.149	0.9109351	0.0890649	FALSE	1	266
8810158	B	13.11	0.136	0.013	0.149	0.9109351	0.0890649	FALSE	1	267
9113455	B	13.14	0.135	0.013	0.148	0.90931151	0.09068849	FALSE	1	268
8711002	B	13.15	0.134	0.013	0.148	0.9087612	0.0912388	FALSE	1	269
914102	B	13.16	0.134	0.014	0.148	0.90820627	0.09179373	FALSE	1	270
85638502	M	13.17	0.134	0.014	0.147	0.90764667	0.09235333	FALSE	1	271
85715	M	13.17	0.134	0.014	0.147	0.90764667	0.09235333	FALSE	1	272
911320502	B	13.17	0.134	0.014	0.147	0.90764667	0.09235333	FALSE	1	273
884448	B	13.20	0.133	0.014	0.147	0.90593952	0.09406048	FALSE	1	274
89344	B	13.20	0.133	0.014	0.147	0.90593952	0.09406048	FALSE	1	275
9112367	B	13.21	0.133	0.014	0.147	0.90536088	0.09463912	FALSE	1	276
922296	B	13.21	0.133	0.014	0.147	0.90536088	0.09463912	FALSE	1	277
9113239	B	13.24	0.132	0.014	0.146	0.90359565	0.09640435	FALSE	1	278
861853	B	13.27	0.131	0.014	0.145	0.90178556	0.09821444	FALSE	1	279
8813129	B	13.27	0.131	0.014	0.145	0.90178556	0.09821444	FALSE	1	280
856106	M	13.28	0.130	0.014	0.145	0.90117204	0.09882796	FALSE	1	281
902727	B	13.28	0.130	0.014	0.145	0.90117204	0.09882796	FALSE	1	282
901041	B	13.30	0.130	0.014	0.144	0.89992951	0.10007049	FALSE	1	283
8611161	B	13.34	0.128	0.015	0.143	0.89738125	0.10261875	FALSE	1	284
865468	B	13.37	0.127	0.015	0.142	0.89541339	0.10458661	FALSE	1	285
9112085	B	13.38	0.127	0.015	0.142	0.8947464	0.1052536	FALSE	1	286
9010877	B	13.40	0.126	0.015	0.141	0.89339561	0.10660439	FALSE	1	287
915691	M	13.40	0.126	0.015	0.141	0.89339561	0.10660439	FALSE	1	288
87163	M	13.43	0.125	0.015	0.140	0.89132669	0.10867331	FALSE	1	289
855167	M	13.44	0.125	0.015	0.140	0.89062546	0.10937454	FALSE	1	290
862009	B	13.45	0.124	0.015	0.140	0.88991837	0.11008163	FALSE	1	291
9013579	B	13.46	0.124	0.015	0.139	0.88920535	0.11079465	FALSE	1	292
91813701	B	13.46	0.124	0.015	0.139	0.88920535	0.11079465	FALSE	1	293
912519	B	13.47	0.124	0.016	0.139	0.88848638	0.11151362	FALSE	1	294
855138	M	13.48	0.123	0.016	0.139	0.88776139	0.11223861	FALSE	1	295
857156	B	13.49	0.123	0.016	0.138	0.88703033	0.11296967	FALSE	1	296
893526	B	13.50	0.122	0.016	0.138	0.88629317	0.11370683	FALSE	1	297
90401601	B	13.51	0.122	0.016	0.138	0.88554985	0.11445015	FALSE	1	298
8610629	B	13.53	0.121	0.016	0.137	0.88404452	0.11595548	FALSE	1	299
8510426	B	13.54	0.121	0.016	0.137	0.88328242	0.11671758	FALSE	1	300
8812818	B	13.56	0.120	0.016	0.136	0.8817391	0.1182609	FALSE	1	301
8910499	B	13.59	0.119	0.016	0.135	0.87937554	0.12062446	FALSE	1	302
8911800	B	13.59	0.119	0.016	0.135	0.87937554	0.12062446	FALSE	1	303
862717	M	13.61	0.118	0.016	0.134	0.87776685	0.12223315	FALSE	1	304
866083	M	13.61	0.118	0.016	0.134	0.87776685	0.12223315	FALSE	1	305
922576	B	13.62	0.118	0.017	0.134	0.87695245	0.12304755	FALSE	1	306
857373	B	13.64	0.117	0.017	0.133	0.8753033	0.1246967	FALSE	1	307
88350402	B	13.64	0.117	0.017	0.133	0.8753033	0.1246967	FALSE	1	308
8812816	B	13.65	0.116	0.017	0.133	0.87446844	0.12553156	FALSE	1	309
9013594	B	13.66	0.116	0.017	0.133	0.87362665	0.12637335	FALSE	1	310

906878	B	13.66	0.116	0.017	0.133	0.87362665	0.12637335	FALSE	1	311
91903902	B	13.68	0.115	0.017	0.132	0.87192207	0.12807793	FALSE	1	312
9013005	B	13.69	0.115	0.017	0.132	0.87105917	0.12894083	FALSE	1	313
912558	B	13.70	0.114	0.017	0.131	0.87018913	0.12981087	FALSE	1	314
84458202	M	13.71	0.114	0.017	0.131	0.86931189	0.13068811	FALSE	1	315
917896	B	13.71	0.114	0.017	0.131	0.86931189	0.13068811	FALSE	1	316
84667401	M	13.73	0.113	0.017	0.130	0.86753559	0.13246441	FALSE	1	317
869931	B	13.74	0.113	0.017	0.130	0.86663643	0.13336357	FALSE	1	318
88411702	B	13.75	0.112	0.017	0.129	0.86572985	0.13427015	FALSE	1	319
875938	M	13.77	0.111	0.018	0.129	0.86389421	0.13610579	FALSE	1	320
891923	B	13.77	0.111	0.018	0.129	0.86389421	0.13610579	FALSE	1	321
90944601	B	13.78	0.111	0.018	0.128	0.86296504	0.13703496	FALSE	1	322
859983	M	13.80	0.110	0.018	0.128	0.86108371	0.13891629	FALSE	1	323
87880	M	13.81	0.109	0.018	0.127	0.86013144	0.13986856	FALSE	1	324
91504	M	13.82	0.109	0.018	0.127	0.85917137	0.14082863	FALSE	1	325
86561	B	13.85	0.108	0.018	0.126	0.85624369	0.14375631	FALSE	1	326
8911834	B	13.85	0.108	0.018	0.126	0.85624369	0.14375631	FALSE	1	327
909231	B	13.85	0.108	0.018	0.126	0.85624369	0.14375631	FALSE	1	328
8810987	M	13.86	0.107	0.018	0.125	0.85525179	0.14474821	FALSE	1	329
901028	B	13.87	0.107	0.018	0.125	0.85425179	0.14574821	FALSE	1	330
922297	B	13.87	0.107	0.018	0.125	0.85425179	0.14574821	FALSE	1	331
902976	B	13.88	0.106	0.018	0.125	0.85324363	0.14675637	FALSE	1	332
911673	B	13.90	0.105	0.018	0.124	0.85120262	0.14879738	FALSE	1	333
91227	B	13.90	0.105	0.018	0.124	0.85120262	0.14879738	FALSE	1	334
918192	B	13.94	0.104	0.019	0.122	0.84702014	0.15297986	FALSE	1	335
886452	M	13.96	0.103	0.019	0.121	0.84487774	0.15512226	FALSE	1	336
908489	M	13.98	0.102	0.019	0.121	0.8427006	0.1572994	FALSE	1	337
909410	B	14.02	0.100	0.019	0.119	0.83824021	0.16175979	FALSE	1	338
88249602	B	14.03	0.099	0.019	0.119	0.83710266	0.16289734	FALSE	1	339
909220	B	14.04	0.099	0.019	0.118	0.83595601	0.16404399	FALSE	1	340
925292	B	14.05	0.098	0.019	0.118	0.8348002	0.1651998	FALSE	1	341
903811	B	14.06	0.098	0.020	0.117	0.83363517	0.16636483	FALSE	1	342
89524	B	14.11	0.096	0.020	0.115	0.82766968	0.17233032	FALSE	1	343
8810955	M	14.19	0.092	0.020	0.112	0.81762392	0.18237608	FALSE	1	344
911654	B	14.20	0.091	0.021	0.112	0.81632361	0.18367639	FALSE	1	345
874858	M	14.22	0.090	0.021	0.111	0.81369262	0.18630738	FALSE	1	346
883270	B	14.22	0.090	0.021	0.111	0.81369262	0.18630738	FALSE	1	347
854268	M	14.25	0.089	0.021	0.110	0.80966934	0.19033066	FALSE	1	348
858986	M	14.25	0.089	0.021	0.110	0.80966934	0.19033066	FALSE	1	349
86409	B	14.26	0.088	0.021	0.109	0.80830754	0.19169246	FALSE	1	350
892214	B	14.26	0.088	0.021	0.109	0.80830754	0.19169246	FALSE	1	351
91979701	M	14.27	0.088	0.021	0.109	0.8069353	0.1930647	FALSE	1	352
8910721	B	14.29	0.087	0.021	0.108	0.80415923	0.19584077	FALSE	1	353
88143502	B	14.34	0.085	0.022	0.106	0.7970323	0.2029677	FALSE	1	354
9113156	B	14.40	0.082	0.022	0.104	0.78812003	0.21187997	FALSE	1	355
89143602	B	14.41	0.081	0.022	0.103	0.78659573	0.21340427	FALSE	1	356
862548	M	14.42	0.081	0.022	0.103	0.78506019	0.21493981	FALSE	1	357
89813	B	14.42	0.081	0.022	0.103	0.78506019	0.21493981	FALSE	1	358
86973702	B	14.44	0.080	0.022	0.102	0.78195512	0.21804488	FALSE	1	359
877500	M	14.45	0.079	0.022	0.102	0.78038551	0.21961449	FALSE	1	360
921386	B	14.47	0.079	0.023	0.101	0.77721189	0.22278811	FALSE	1	361
86135501	M	14.48	0.078	0.023	0.101	0.77560778	0.22439222	FALSE	1	362
865432	B	14.50	0.077	0.023	0.100	0.77236475	0.22763525	FALSE	1	363
911150	B	14.53	0.076	0.023	0.099	0.76741248	0.23258752	FALSE	1	364
911201	B	14.53	0.076	0.023	0.099	0.76741248	0.23258752	FALSE	1	365
84799002	M	14.54	0.075	0.023	0.098	0.76573816	0.23426184	FALSE	1	366
852763	M	14.58	0.073	0.023	0.097	0.75892197	0.24107803	FALSE	1	367
915940	B	14.58	0.073	0.023	0.097	0.75892197	0.24107803	FALSE	1	368
925277	B	14.59	0.073	0.023	0.096	0.757188	0.242812	FALSE	1	369
90291	M	14.60	0.072	0.023	0.096	0.75544199	0.24455801	FALSE	1	370
89382601	B	14.61	0.072	0.024	0.096	0.75368389	0.24631611	FALSE	1	371
861648	B	14.62	0.072	0.024	0.095	0.75191369	0.24808631	FALSE	1	372
861598	B	14.64	0.071	0.024	0.094	0.74833679	0.25166321	FALSE	1	373

913102	B	14.64	0.071	0.024	0.094	0.74833679	0.25166321	FALSE	1	374
848406	M	14.68	0.069	0.024	0.093	0.74103624	0.25896376	FALSE	1	375
906564	B	14.69	0.068	0.024	0.092	0.73918036	0.26081964	FALSE	1	376
857793	M	14.71	0.067	0.024	0.092	0.73543152	0.26456848	FALSE	1	377
921644	B	14.74	0.066	0.024	0.091	0.72971523	0.27028477	FALSE	1	378
89869	B	14.76	0.065	0.025	0.090	0.72584212	0.27415788	FALSE	1	379
859283	M	14.78	0.064	0.025	0.089	0.72191905	0.27808095	FALSE	1	380
9110944	B	14.80	0.063	0.025	0.088	0.71794591	0.28205409	FALSE	1	381
915664	B	14.81	0.063	0.025	0.088	0.71594054	0.28405946	FALSE	1	382
87556202	M	14.86	0.061	0.025	0.086	0.70572537	0.29427463	FALSE	1	383
908469	B	14.86	0.061	0.025	0.086	0.70572537	0.29427463	FALSE	1	384
864729	M	14.87	0.060	0.025	0.086	0.70364465	0.29635535	FALSE	1	385
914333	B	14.87	0.060	0.025	0.086	0.70364465	0.29635535	FALSE	1	386
907914	M	14.90	0.059	0.026	0.085	0.69732716	0.30267284	FALSE	1	387
911384	B	14.92	0.058	0.026	0.084	0.69305277	0.30694723	FALSE	1	388
868826	M	14.95	0.057	0.026	0.083	0.68654731	0.31345269	FALSE	1	389
86973701	B	14.95	0.057	0.026	0.083	0.68654731	0.31345269	FALSE	1	390
8915	B	14.96	0.057	0.026	0.083	0.68435385	0.31564615	FALSE	1	391
8712291	B	14.97	0.056	0.026	0.082	0.68214793	0.31785207	FALSE	1	392
8712853	B	14.97	0.056	0.026	0.082	0.68214793	0.31785207	FALSE	1	393
855133	M	14.99	0.055	0.026	0.082	0.6776988	0.3223012	FALSE	1	394
905557	B	14.99	0.055	0.026	0.082	0.6776988	0.3223012	FALSE	1	395
88147102	B	15.00	0.055	0.026	0.081	0.67545564	0.32454436	FALSE	1	396
914862	B	15.04	0.053	0.027	0.080	0.66635972	0.33364028	FALSE	1	397
91594602	M	15.05	0.053	0.027	0.080	0.66405508	0.33594492	FALSE	1	398
862028	M	15.06	0.052	0.027	0.079	0.66173825	0.33826175	FALSE	1	399
9010018	M	15.08	0.052	0.027	0.079	0.65706819	0.34293181	FALSE	1	400
857438	M	15.10	0.051	0.027	0.078	0.6523499	0.3476501	FALSE	1	401
866458	B	15.10	0.051	0.027	0.078	0.6523499	0.3476501	FALSE	1	402
879523	M	15.12	0.050	0.027	0.077	0.64758376	0.35241624	FALSE	1	403
905680	M	15.13	0.050	0.027	0.077	0.64518288	0.35481712	FALSE	1	404
9012568	B	15.19	0.047	0.028	0.075	0.63053206	0.36946794	FALSE	1	405
925622	M	15.22	0.046	0.028	0.074	0.62305182	0.37694818	FALSE	1	406
8810436	B	15.27	0.044	0.028	0.073	0.61036255	0.38963745	FALSE	1	407
873885	M	15.28	0.044	0.028	0.072	0.60779215	0.39220785	FALSE	1	408
852973	M	15.30	0.043	0.029	0.072	0.60261954	0.39738046	FALSE	1	409
886776	M	15.32	0.043	0.029	0.071	0.59740519	0.40259481	FALSE	1	410
8511133	M	15.34	0.042	0.029	0.071	0.59214995	0.40785005	FALSE	1	411
861799	M	15.37	0.041	0.029	0.070	0.58419236	0.41580764	FALSE	1	412
8670	M	15.46	0.038	0.030	0.067	0.55981215	0.44018785	FALSE	1	413
87164	M	15.46	0.038	0.030	0.067	0.55981215	0.44018785	FALSE	1	414
915460	M	15.46	0.038	0.030	0.067	0.55981215	0.44018785	FALSE	1	415
903507	M	15.49	0.037	0.030	0.067	0.55152821	0.44847179	FALSE	1	416
90602302	M	15.50	0.036	0.030	0.066	0.54875067	0.45124933	FALSE	1	417
88725602	M	15.53	0.035	0.030	0.066	0.54037152	0.45962848	FALSE	1	418
889403	M	15.61	0.033	0.031	0.064	0.51771453	0.48228547	FALSE	1	419
887181	M	15.66	0.031	0.031	0.062	0.50335293	0.49664707	FALSE	1	420
873701	M	15.70	0.030	0.031	0.061	0.49177108	0.50822892	TRUE	2	1
867387	B	15.71	0.030	0.031	0.061	0.48886437	0.51113563	TRUE	2	2
912600	B	15.73	0.029	0.031	0.061	0.48303885	0.51696115	TRUE	2	3
8812877	M	15.75	0.029	0.032	0.060	0.47719853	0.52280147	TRUE	2	4
899667	M	15.75	0.029	0.032	0.060	0.47719853	0.52280147	TRUE	2	5
84610002	M	15.78	0.028	0.032	0.060	0.46841393	0.53158607	TRUE	2	6
864877	M	15.78	0.028	0.032	0.060	0.46841393	0.53158607	TRUE	2	7
846381	M	15.85	0.026	0.032	0.058	0.44783351	0.55216649	TRUE	2	8
845636	M	16.02	0.022	0.033	0.055	0.39777151	0.60222849	TRUE	2	9
896839	M	16.03	0.022	0.033	0.055	0.39483996	0.60516004	TRUE	2	10
8610404	M	16.07	0.021	0.034	0.054	0.38314517	0.61685483	TRUE	2	11
869104	M	16.11	0.020	0.034	0.054	0.37151146	0.62848854	TRUE	2	12
84862001	M	16.13	0.020	0.034	0.053	0.365722	0.634278	TRUE	2	13
854039	M	16.13	0.020	0.034	0.053	0.365722	0.634278	TRUE	2	14
905189	B	16.14	0.019	0.034	0.053	0.36283489	0.63716511	TRUE	2	15
86730502	M	16.16	0.019	0.034	0.053	0.35707704	0.64292296	TRUE	2	16

901303	B	16.17	0.019	0.034	0.053	0.35420673	0.64579327	TRUE	2	17
8912280	M	16.24	0.017	0.034	0.052	0.33429551	0.66570449	TRUE	2	18
911916	M	16.25	0.017	0.035	0.052	0.33147944	0.66852056	TRUE	2	19
895633	M	16.26	0.017	0.035	0.052	0.32867109	0.67132891	TRUE	2	20
8953902	M	16.27	0.017	0.035	0.051	0.32587066	0.67412934	TRUE	2	21
915452	B	16.30	0.016	0.035	0.051	0.31751895	0.68248105	TRUE	2	22
9012315	M	16.35	0.015	0.035	0.050	0.30377637	0.69622363	TRUE	2	23
87281702	M	16.46	0.013	0.036	0.049	0.27442436	0.72557564	TRUE	2	24
9010872	B	16.50	0.013	0.036	0.049	0.26408565	0.73591435	TRUE	2	25
926954	M	16.60	0.011	0.036	0.048	0.23910777	0.76089223	TRUE	2	26
852552	M	16.65	0.011	0.036	0.047	0.22711481	0.77288519	TRUE	2	27
913535	M	16.69	0.010	0.037	0.047	0.21777113	0.78222887	TRUE	2	28
854253	M	16.74	0.010	0.037	0.046	0.20641452	0.79358548	TRUE	2	29
8712729	M	16.78	0.009	0.037	0.046	0.19759365	0.80240635	TRUE	2	30
8711216	B	16.84	0.008	0.037	0.046	0.1848124	0.8151876	TRUE	2	31
879830	M	17.01	0.007	0.038	0.045	0.15159809	0.84840191	TRUE	2	32
85382601	M	17.02	0.007	0.038	0.045	0.14978362	0.85021638	TRUE	2	33
881972	M	17.05	0.006	0.038	0.044	0.14443302	0.85556698	TRUE	2	34
89742801	M	17.06	0.006	0.038	0.044	0.14268037	0.85731963	TRUE	2	35
911296201	M	17.08	0.006	0.038	0.044	0.13922129	0.86077871	TRUE	2	36
852631	M	17.14	0.006	0.038	0.044	0.12921178	0.87078822	TRUE	2	37
889719	M	17.19	0.005	0.038	0.044	0.12128775	0.87871225	TRUE	2	38
859717	M	17.20	0.005	0.038	0.044	0.11974795	0.88025205	TRUE	2	39
909445	M	17.27	0.005	0.039	0.043	0.10938334	0.89061666	TRUE	2	40
888570	M	17.29	0.005	0.039	0.043	0.10655335	0.89344665	TRUE	2	41
8860702	M	17.30	0.005	0.039	0.043	0.10515994	0.89484006	TRUE	2	42
888264	M	17.35	0.004	0.039	0.043	0.09840594	0.90159406	TRUE	2	43
881094802	M	17.42	0.004	0.039	0.043	0.0895338	0.9104662	TRUE	2	44
88330202	M	17.46	0.004	0.039	0.043	0.08476058	0.91523942	TRUE	2	45
8712766	M	17.47	0.004	0.039	0.042	0.08360015	0.91639985	TRUE	2	46
877989	M	17.54	0.003	0.039	0.042	0.07583593	0.92416407	TRUE	2	47
853201	M	17.57	0.003	0.039	0.042	0.07269573	0.92730427	TRUE	2	48
9113538	M	17.60	0.003	0.039	0.042	0.06966442	0.93033558	TRUE	2	49
8711202	M	17.68	0.003	0.039	0.042	0.06209432	0.93790568	TRUE	2	50
9110732	M	17.75	0.002	0.039	0.042	0.05605353	0.94394647	TRUE	2	51
91376702	B	17.85	0.002	0.039	0.041	0.04830101	0.95169899	TRUE	2	52
90439701	M	17.91	0.002	0.039	0.041	0.04411027	0.95588973	TRUE	2	53
8911163	M	17.93	0.002	0.039	0.041	0.04278571	0.95721429	TRUE	2	54
865128	M	17.95	0.002	0.039	0.041	0.04149609	0.95850391	TRUE	2	55
842302	M	17.99	0.002	0.039	0.041	0.03901886	0.96098114	TRUE	2	56
90524101	M	17.99	0.002	0.039	0.041	0.03901886	0.96098114	TRUE	2	57
914062	M	18.01	0.002	0.039	0.041	0.0378298	0.9621702	TRUE	2	58
9110127	M	18.03	0.001	0.039	0.041	0.03667286	0.96332714	TRUE	2	59
8610637	M	18.05	0.001	0.039	0.041	0.03554732	0.96445268	TRUE	2	60
877159	M	18.08	0.001	0.039	0.041	0.03391641	0.96608359	TRUE	2	61
857392	M	18.22	0.001	0.039	0.040	0.02715341	0.97284659	TRUE	2	62
894326	M	18.22	0.001	0.039	0.040	0.02715341	0.97284659	TRUE	2	63
844359	M	18.25	0.001	0.039	0.040	0.02587227	0.97412773	TRUE	2	64
874217	M	18.31	0.001	0.039	0.040	0.02347214	0.97652786	TRUE	2	65
916799	M	18.31	0.001	0.039	0.040	0.02347214	0.97652786	TRUE	2	66
867739	M	18.45	0.001	0.039	0.039	0.01863651	0.98136349	TRUE	2	67
8612399	M	18.46	0.001	0.039	0.039	0.01832855	0.98167145	TRUE	2	68
914769	M	18.49	0.001	0.039	0.039	0.01743232	0.98256768	TRUE	2	69
852781	M	18.61	0.001	0.038	0.039	0.0142341	0.9857659	TRUE	2	70
853401	M	18.63	0.001	0.038	0.039	0.01375676	0.98624324	TRUE	2	71
857010	M	18.65	0.001	0.038	0.039	0.01329419	0.98670581	TRUE	2	72
86517	M	18.66	0.001	0.038	0.039	0.01306831	0.98693169	TRUE	2	73
897630	M	18.77	0.000	0.038	0.038	0.01080682	0.98919318	TRUE	2	74
8911670	M	18.81	0.000	0.038	0.038	0.01007848	0.98992152	TRUE	2	75
908445	M	18.82	0.000	0.038	0.038	0.00990365	0.99009635	TRUE	2	76
859575	M	18.94	0.000	0.037	0.038	0.0080139	0.9919861	TRUE	2	77
866203	M	19.00	0.000	0.037	0.037	0.00720037	0.99279963	TRUE	2	78
86135502	M	19.02	0.000	0.037	0.037	0.00694677	0.99305323	TRUE	2	79

855625	M	19.07	0.000	0.037	0.037	0.00634877	0.99365123	TRUE	2	80
8611792	M	19.10	0.000	0.037	0.037	0.00601343	0.99398657	TRUE	2	81
8912049	M	19.16	0.000	0.036	0.037	0.00539185	0.99460815	TRUE	2	82
846226	M	19.17	0.000	0.036	0.037	0.0052943	0.9947057	TRUE	2	83
877486	M	19.18	0.000	0.036	0.037	0.0051984	0.9948016	TRUE	2	84
8711803	M	19.19	0.000	0.036	0.036	0.00510413	0.99489587	TRUE	2	85
857637	M	19.21	0.000	0.036	0.036	0.00492038	0.99507962	TRUE	2	86
854002	M	19.27	0.000	0.036	0.036	0.00440565	0.99559435	TRUE	2	87
884180	M	19.40	0.000	0.035	0.035	0.00345875	0.99654125	TRUE	2	88
89122	M	19.40	0.000	0.035	0.035	0.00345875	0.99654125	TRUE	2	89
913505	M	19.44	0.000	0.035	0.035	0.00320832	0.99679168	TRUE	2	90
886226	M	19.45	0.000	0.035	0.035	0.00314844	0.99685156	TRUE	2	91
88119002	M	19.53	0.000	0.035	0.035	0.00270593	0.99729407	TRUE	2	92
892438	M	19.53	0.000	0.035	0.035	0.00270593	0.99729407	TRUE	2	93
88649001	M	19.55	0.000	0.035	0.035	0.00260485	0.99739515	TRUE	2	94
90312	M	19.55	0.000	0.035	0.035	0.00260485	0.99739515	TRUE	2	95
871201	M	19.59	0.000	0.034	0.034	0.00241329	0.99758671	TRUE	2	96
9111805	M	19.59	0.000	0.034	0.034	0.00241329	0.99758671	TRUE	2	97
898431	M	19.68	0.000	0.034	0.034	0.00202981	0.99797019	TRUE	2	98
84300903	M	19.69	0.000	0.034	0.034	0.00199096	0.99800904	TRUE	2	99
885429	M	19.73	0.000	0.034	0.034	0.00184246	0.99815754	TRUE	2	100
866674	M	19.79	0.000	0.033	0.033	0.00163923	0.99836077	TRUE	2	101
8811842	M	19.80	0.000	0.033	0.033	0.00160749	0.99839251	TRUE	2	102
849014	M	19.81	0.000	0.033	0.033	0.00157634	0.99842366	TRUE	2	103
916838	M	19.89	0.000	0.033	0.033	0.00134693	0.99865307	TRUE	2	104
89263202	M	20.09	0.000	0.031	0.031	0.00090399	0.99909601	TRUE	2	105
926682	M	20.13	0.000	0.031	0.031	0.00083391	0.99916609	TRUE	2	106
894618	M	20.16	0.000	0.031	0.031	0.00078476	0.99921524	TRUE	2	107
8610862	M	20.18	0.000	0.031	0.031	0.00075355	0.99924645	TRUE	2	108
908194	M	20.18	0.000	0.031	0.031	0.00075355	0.99924645	TRUE	2	109
9011494	M	20.20	0.000	0.031	0.031	0.00072352	0.99927648	TRUE	2	110
86208	M	20.26	0.000	0.030	0.030	0.00064012	0.99935988	TRUE	2	111
84358402	M	20.29	0.000	0.030	0.030	0.00060193	0.99939807	TRUE	2	112
887549	M	20.31	0.000	0.030	0.030	0.00057769	0.99942231	TRUE	2	113
895100	M	20.34	0.000	0.030	0.030	0.00054307	0.99945693	TRUE	2	114
901088	M	20.44	0.000	0.029	0.029	0.00044141	0.99955859	TRUE	2	115
91930402	M	20.47	0.000	0.029	0.029	0.00041464	0.99958536	TRUE	2	116
883263	M	20.48	0.000	0.029	0.029	0.00040606	0.99959394	TRUE	2	117
88206102	M	20.51	0.000	0.028	0.028	0.00038135	0.99961865	TRUE	2	118
919555	M	20.55	0.000	0.028	0.028	0.00035062	0.99964938	TRUE	2	119
842517	M	20.57	0.000	0.028	0.028	0.00033616	0.99966384	TRUE	2	120
881046502	M	20.58	0.000	0.028	0.028	0.00032914	0.99967086	TRUE	2	121
91485	M	20.59	0.000	0.028	0.028	0.00032227	0.99967773	TRUE	2	122
927241	M	20.60	0.000	0.028	0.028	0.00031553	0.99968447	TRUE	2	123
901288	M	20.64	0.000	0.028	0.028	0.0002899	0.9997101	TRUE	2	124
88995002	M	20.73	0.000	0.027	0.027	0.00023932	0.99976068	TRUE	2	125
926125	M	20.92	0.000	0.026	0.026	0.00015882	0.99984118	TRUE	2	126
884948	M	20.94	0.000	0.025	0.025	0.00015205	0.99984795	TRUE	2	127
873593	M	21.09	0.000	0.024	0.024	0.0001094	0.9998906	TRUE	2	128
911157302	M	21.10	0.000	0.024	0.024	0.000107	0.999893	TRUE	2	129
851509	M	21.16	0.000	0.024	0.024	9.3674E-05	0.99990633	TRUE	2	130
9012795	M	21.37	0.000	0.022	0.022	5.8478E-05	0.99994152	TRUE	2	131
926424	M	21.56	0.000	0.021	0.021	3.79E-05	0.9999621	TRUE	2	132
903516	M	21.61	0.000	0.020	0.020	3.3773E-05	0.99996623	TRUE	2	133
9011971	M	21.71	0.000	0.020	0.020	2.6778E-05	0.99997322	TRUE	2	134
8910988	M	21.75	0.000	0.019	0.019	2.4391E-05	0.99997561	TRUE	2	135
9012000	M	22.01	0.000	0.018	0.018	1.3194E-05	0.99998681	TRUE	2	136
86355	M	22.27	0.000	0.016	0.016	7.0437E-06	0.99999296	TRUE	2	137
915143	M	23.09	0.000	0.011	0.011	8.9283E-07	0.99999911	TRUE	2	138
88299702	M	23.21	0.000	0.010	0.010	6.5273E-07	0.99999935	TRUE	2	139
8712289	M	23.27	0.000	0.010	0.010	5.5752E-07	0.99999944	TRUE	2	140
878796	M	23.29	0.000	0.010	0.010	5.289E-07	0.99999947	TRUE	2	141
89812	M	23.51	0.000	0.009	0.009	2.9467E-07	0.99999971	TRUE	2	142

865423	M	24.25	0.000	0.006	0.006	3.8446E-08	0.99999996	TRUE	2	143
91762702	M	24.63	0.000	0.005	0.005	1.2963E-08	0.99999999	TRUE	2	144
8611555	M	25.22	0.000	0.003	0.003	2.267E-09	1	TRUE	2	145
899987	M	25.73	0.000	0.002	0.002	4.7557E-10	1	TRUE	2	146
873592	M	27.22	0.000	0.001	0.001	3.7145E-12	1	TRUE	2	147
911296202	M	27.42	0.000	0.001	0.001	1.874E-12	1	TRUE	2	148
8810703	M	28.11	0.000	0.000	0.000	1.6663E-13	1	TRUE	2	149